



POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Bartosz Kowalski, s15154

Jan Talarek, s26710

Kompleksowy Rejestr Eksploracji Terenowych „KRET”

Praca inżynierska

Promotor:

dr inż. Mariusz Trzaska, prof. PJATK

Warszawa, lipiec 2026

Streszczenie

Celem niniejszej pracy jest zaprojektowanie i zaimplementowanie wieloplatformowego systemu wspomagającego zarządzanie pracami geologicznymi w terenie. System ułatwia współpracę pomiędzy dokumentatorem a geologami terenowymi, zapewnia cyfryzację danych pozyskiwanych w terenie (również w trybie *offline*), zautomatyzowane sprawdzanie danych, prezentację na mapie oraz generowanie raportów wynikowych. Rezultatem pracy jest działający prototyp składający się z serwera napisanego w języku Java z wykorzystaniem szkieletu aplikacyjnego Spring Boot, aplikacji internetowej napisanej w języku TypeScript z użyciem biblioteki React oraz aplikacji mobilnej zbudowanej przy pomocy technologii React Native. Za przechowywanie danych odpowiada relacyjna baza danych PostgreSQL z rozszerzeniem przestrzennym PostGIS.

Słowa kluczowe:

geologia inżynierska, badania geologiczne, prace terenowe, zarządzanie projektem geologicznym, aplikacja mobilna, aplikacja internetowa, tryb *offline*, GIS, Java, Spring Boot, Hibernate, PostgreSQL, PostGIS, TypeScript, React, React Native, Tailwind CSS, Leaflet, MinIO, Liquibase, REST API.

Spis treści

1. WSTĘP	6
1.1. Cel pracy	6
1.2. Zakres prac	6
1.3. Organizacja pracy	7
1.4. Zespół	7
1.5. Podział prac	8
1.6. Wprowadzenie do problematyki	8
2. STAN SZTUKI	10
2.1. Specyfikacja prac geologii inżynierskiej w Polsce	10
2.2. GeoStar (Soft-Projekt, Wrocław)	12
2.3. Geosoft GEO DB	14
2.4. GEO5	17
2.5. GIMSGEO	19
2.6. GeoAid	20
2.7. Rozwiązania otwartoźródłowe (QField, Mergin Maps, BGS·SIGMA)	23
2.7.1. QField	23
2.7.2. Mergin Maps	23
2.7.3. BGS·SIGMA	24
2.8. Generyczne systemy zarządzania pracami terenowymi	25
2.9. Porównanie systemów	26
2.10. Analiza luk i pozycjonowanie systemu KRET	28
2.11. Podsumowanie	29
3. PROPOZYCJA SYSTEMU KRET	30
3.1. Cele i założenia	30
3.2. Słownik pojęć	30
3.3. Użytkownicy systemu	31
3.4. Wymagania funkcjonalne	32
3.5. Wymagania нефункционалне	40
3.6. Diagram przypadków użycia	42
3.6.1. Diagram przypadków użycia dla Gościa systemu	42
3.6.2. Diagram przypadków użycia dla Geologa terenowego	43
3.6.3. Diagram przypadków użycia dla Dokumentatora	45
3.6.4. Diagram przypadków użycia dla Administratora	46
3.7. Omówienie wybranych przypadków użycia	47
3.7.1. Rejestracja użytkownika (Gość)	47
3.7.2. Tworzenie nowego otworu (Dokumentator/Admin)	47
3.7.3. Wypełnianie formularza wiercenia (Geolog terenowy)	48
3.7.4. Importowanie wiercenia z pliku (Dokumentator/Administrator)	49
3.7.5. Praca w trybie offline	50
3.7.6. Weryfikacja i zatwierdzenie badania (Dokumentator/Administrator)	51
3.7.7. Eksport karty otworu	51
3.8. Diagram klas	52
3.9. Diagramy stanów	54
3.9.1. Diagram stanów otworu	54
3.9.2. Diagram stanów sprzętu	55
3.10. Diagramy aktywności	55
3.10.1. Diagram aktywności dla rejestracji użytkownika	55
3.10.2. Diagram aktywności dla dodawania nowego otworu	57
3.10.3. Diagram aktywności dla tworzenia wiercenia	58
3.10.4. Diagram aktywności dla weryfikacji formularza wiercenia	59
3.10.5. Diagram aktywności dla eksportu karty otworu	60
4. TECHNOLOGIE I NARZĘDZIA	61

4.1. Java i Spring Boot	61
4.2. PostgreSQL i PostGIS	62
4.3. React.....	63
4.4. TypeScript.....	64
4.5. Tailwind CSS	65
4.6. React Native i Expo.....	65
4.7. Leaflet i mapy.....	67
4.8. MinIO.....	68
4.9. Liquibase.....	68
4.10. Lombok	69
4.11. Spring Security i JWT	70
4.12. Apache POI	70
4.13. Konteneryzacja i wdrożenie	71
4.14. IntelliJ IDEA	72
4.15. Draw.io.....	73
4.16. GitHub i GitHub Actions	73
4.17. GitHub Projects.....	75
4.18. OpenAPI i Swagger.....	75
5. IMPLEMENTACJA.....	76
5.1. Architektura systemu i infrastruktura wdrożeniowa.....	76
5.1.1. Architektura serwerowa	77
5.1.2. Konteneryzacja z Docker.....	77
5.1.3. Środowisko produkcyjne.....	78
5.1.4. Ciągła integracja z GitHub Actions.....	79
5.1.5. Dokumentacja API z Open API	80
5.2. Model PostGIS i dane przestrzenne	81
5.3. Uwierzytelnianie i zarządzanie użytkownikami.....	83
5.3.1. Autoryzacja oparta na tokenach.....	83
5.3.2. Odświeżanie sesji.....	86
5.3.3. Podział ról i uprawnień	87
5.3.4. Profile użytkowników i awatary.....	88
5.3.5. System zaproszeń i rejestracji.....	89
5.4. Przechowywanie i udostępnianie plików	90
5.5. Zarządzanie projektami i otworami.....	94
5.5.1. Cykl życia projektu	94
5.5.2. Zarządzanie otworami	94
5.5.3. Przypisywanie geologów do projektów i otworów.....	95
5.5.4. Import otworów	97
5.6. Wizualizacja mapowa i nawigacja	101
5.7. Formularze badań geologicznych.....	102
5.7.1. Formularz wiercenia	102
5.7.2. Formularz sondowania.....	104
5.7.3. Pola niestandardowe	105
5.7.4. Wzorce formularzy.....	105
5.7.5. Automatyczny zapis formularzy	106
5.7.6. Dokumentacja fotograficzna i zdjęcia 360	107
5.7.7. Automatyczne sprawdzanie i oddawanie badań do sprawdzenia	108
5.7.8. Usuwanie badań	109
5.8. Automatyczne przypisywanie dokumentatorów.....	109
5.9. System dyskusji.....	111
5.10. Powiadomienia.....	112
5.11. System eksportu danych.....	114
5.12. Baza wiedzy	117
5.13. Statystyki i wizualizacje.....	118
5.13.1. Statystyki projektowe	118
5.13.2. Profil geologa.....	120
5.13.3. Wizualizacje na kalendarzu	120

5.14. System zarządzania sprzętem	120
5.15. Architektura zorientowana na pracę offline i synchronizacja	122
5.16. Współdzielona biblioteka modułów	124
5.16.1. Adapter trwałego przechowywania	124
5.16.2. Przechwytywanie odświeżania tokenu	125
5.16.3. Dystrybucja i wersjonowanie	125
6. PREZENTACJA SYSTEMU	127
6.1. System zaproszeń i logowanie.....	127
6.2. Zarządzanie projektami	129
6.3. Mapa otworów i zlecenia terenowe	133
6.4. Formularze badań geologicznych	140
6.4.1. Formularz wiercenia	141
6.4.2. Formularz sondowania	144
6.4.3. Pola niestandardowe	146
6.4.4. Firmowe wzorce formularzy	147
6.5. Weryfikacja dokumentacji i dyskusje	147
6.6. Statystyki i kalendarz	151
6.7. Baza wiedzy	153
6.8. Zarządzanie sprzętem.....	158
6.9. Eksport danych.....	162
6.10. Import otworów	167
7. TESTY I EWALUACJA	169
7.1. Testy automatyczne.....	169
7.2. Testy manualne	171
7.3. Wyniki testów	173
8. PODSUMOWANIE I WNIOSKI	174
8.1. Osiągnięcia projektu.....	174
8.2. Napotkane trudności.....	174
8.3. Ograniczenia.....	175
8.4. Propozycje dalszego rozwoju	175
9. BIBLIOGRAFIA.....	176
10. SPIS RYSUNKÓW	179
11. SPIS TABEL.....	183
12. SPIS LISTINGÓW	184

1. Wstęp

Zdecydowana większość geologów w Polsce pracujących w terenie przy dokumentacji odwiertów opiera się na papierowych rozwiązaniach dla sporządzenia badania, których zawartość jest następnie ręcznie przetwarzana przez oprogramowanie biurowe. Proces ten jest czasochłonny, podatny na błędy i utrudnia koordynację pracy zespołów terenowych.

System KRET (Kompleksowy Rejestr Eksploracji Terenowych) powstał w odpowiedzi na te wyzwania. Jako zintegrowane narzędzie umożliwia cyfrową dokumentację terenową od momentu zlecenia wykonania badania po eksport danych do oprogramowania branżowego.

1.1. Cel pracy

Celem niniejszej pracy inżynierskiej jest zaprojektowanie oraz implementacja systemu KRET, elastycznego narzędzia informatycznego wspierającego badania geologiczne. System składa się z trzech głównych komponentów: aplikacji internetowej przeznaczonej dla dokumentatorów i zarządców systemu; aplikacji mobilnej dedykowanej geologom pracującym w terenie; interfejsu REST zapewniającego integrację między komponentami.

Głównym założeniem projektu jest zastąpienie papierowej dokumentacji terenowej cyfrowym procesem. System ma na celu usprawnienie procesu zbierania, sprawdzania i analizy danych, oraz poprawę koordynacji pracy zespołów w firmie zajmującej się terenowymi badaniami geologicznymi.

Główną motywację do realizacji projektu stanowi zidentyfikowana luka na rynku oprogramowania geologicznego. Istniejące rozwiązania są skoncentrowane na specyficznych aspektach pracy geologa – dokumentacji, wizualizacji i analizy danych – natomiast żadne rozwiązanie dostępne na rynku nie oferuje zintegrowanego systemu usprawniającego zbieranie danych oraz koordynację sprawdzania z analizą danych przez zespoły w spójny sposób.

1.2. Zakres prac

Zakres prac obejmuje zaprojektowanie i implementację systemu KRET w czterech komponentach:

- API – serwer udostępniający interfejs API zaimplementowany w języku Java z wykorzystaniem szkieletu aplikacyjnego Spring Boot. Odpowiada za logikę biznesową, zarządzanie danymi w bazie PostgreSQL z rozszerzeniem PostGIS.
- Web – aplikacja internetowa zaimplementowana w języku JavaScript z wykorzystaniem TypeScript i biblioteki React. Przeznaczona dla administratorów i dokumentatorów. Umożliwia zarządzanie projektami, tworzenie zleceń na mapie, przeglądanie statystyk, zarządzanie bazą wiedzy oraz weryfikację dokumentacji geologicznej.
- Mobile – aplikacja mobilna zaimplementowana w technologii React Native z wykorzystaniem szkieletu aplikacyjnego Expo. Dedykowana geologom terenowym. Główną funkcję stanowi możliwość wypełniania formularzy przypisanych badań (wiercenia lub sondowania), nawigacji do zleceń, dokumentacji fotograficznej oraz pracy w trybie *offline*.
- Shared – współdzielona biblioteka zawierająca moduły kodu TypeScript wykorzystywane przez Web i Mobile. Przechowuje między innymi: typy danych, stałe, logikę sprawdzania danych, logikę zarządzania stanem formularzy oraz logikę klienta API.

System obejmuje następujące moduły funkcjonalne:

- zarządzanie projektami, utworami i użytkownikami;

- system zleceń terenowych z wizualizacją na mapie;
- formularze badań geologicznych (wiercenie i sondowanie) z polami niestandardowymi, dynamicznymi szablonami i dokumentacją fotograficzną z trybem zdjęć 360;
- system automatycznego przypisywania dokumentatorów do sprawdzenia dokumentacji badań;
- system dyskusji dla efektywnej komunikacji przy sprawdzaniu badania;
- bazę wiedzy;
- statystyki i profil geologa z wizualizacjami wykonanych oraz planowanych prac na kalendarzu;
- rozszerzalny system eksportu danych;
- system zaproszeń i rejestracji;
- architekturę zorientowaną na pracę *offline*.

Poza zakresem niniejszej pracy jest: integracja z oprogramowaniem GeoStar wykraczająca poza eksport plików CSV, wdrożenie produkcyjne na skalę komercyjną, wdrożenie aplikacji mobilnej na iOS (prototyp mobilny został zrealizowany na platformę Android).

1.3. Organizacja pracy

Realizacja projektu przebiegała zgodnie z metodyką Kanban umożliwiającą elastyczne zarządzanie zadaniami i iteracyjne dostarczanie kolejnych funkcjonalności. Do zarządzania procesem wytwórczym wykorzystano platformę GitHub, na której rozwijane były cztery repozytoria kodu odpowiadające komponentom opisanym w poprzednim rozdziale.

Zarządzanie zadaniami odbywało się za pośrednictwem GitHub Projects w formie tablicy Kanban. Każde zadanie było reprezentowane przez zgłoszenie (ang. *issue*) zawierające opis wymagań, priorytet oraz przypisanie członka zespołu. Praca nad kodem przebiegała wspólnie dzięki narzędziu Git i oferowanym przez nie modelem rozgałęzień (ang. *branching*) – każda funkcjonalność rozwijana była na dedykowanej gałęzi, a scalenie z gałęzią główną wymagało utworzenia żądania scalenia (ang. *pull request*) i przejścia procesu oceny kodu przez drugiego członka zespołu.

Spotkania z promotorem odbywały się regularnie podczas zjazdów na uczelni co dwa tygodnie. Na każdym spotkaniu prezentowany był postęp prac oraz deklarowany był zakres zadań do realizacji na kolejny zjazd. Prototyp systemu został wdrożony na serwerze Oracle, co umożliwiło bieżącą prezentację postępów z prac.

Istotnym elementem procesu wytwórczego były konsultacje z dokumentatorami i geologami pracującymi w terenie. Jan Talarek, współpracujący na co dzień z osobami z branży geologicznej, prowadził rozmowy mające na celu zebranie wymagań, weryfikację proponowanych rozwiązań oraz sprawdzenie użyteczności zaimplementowanych funkcjonalności. Dzięki temu zespół miał stały dostęp do informacji zwrotnych od potencjalnych przyszłych użytkowników systemu.

1.4. Zespół

Projekt został zrealizowany przez dwuosobowy zespół:

- Bartosz Kowalski (s15154) – lider zespołu, programista. Odpowiadał za architekturę systemu, część projektowania modelu danych oraz implementacji warstwy serwerowej. Ponadto odpowiadał za implementację aplikacji mobilnej oraz współdzielonej biblioteki modułów. Posiada dziewięcioletnie doświadczenie komercyjne w projektowaniu i wdrażaniu aplikacji, co pozwoliło na zapewnienie wysokiej jakości procesu projektowego oraz zastosowanie sprawdzonych wzorców architektonicznych.

- Jan Talarek (s26710) – programista, odpowiedzialny za kontakt z branżą geologiczną. Odpowiadał za implementację aplikacji internetowej, systemu eksportu/importu danych oraz uczestniczył w implementacji kluczowych modułów serwera. Kluczową rolą Jana był kontakt z geologami: zbieranie wymagań funkcjonalnych, weryfikacja poprawności merytorycznej funkcjonalności systemu. Współpraca z osobami z branży na co dzień pozwoliła na stworzenie systemu odpowiadającego rzeczywistym potrzebom.

Prace wspólne obejmowały: opracowanie konspektu i diagramów UML (diagram przypadków użycia, diagram klas, diagram aktywności), testowanie systemu, a także przygotowanie tekstu niniejszej pracy inżynierskiej.

1.5. Podział prac

Praca została wykonana w zespole dwuosobowym. Tabela 1 przedstawia podział prac pomiędzy autorów, obejmujący zarówno wykonanie prototypu, jak i opracowanie tekstu niniejszej pracy.

Tabela 1. Podział prac pomiędzy autorów. Źródło: opracowanie własne.

Obszar	Bartosz Kowalski (s15154)	Jan Talarek (s26710)
Prototyp	Architektura systemu; API serwerowe (Java, Spring Boot): moduły domenowe, uwierzytelnianie i kontrola dostępu, powiadomienia, statystyki, testy automatyczne; aplikacja mobilna (React Native); biblioteka współdzielona @geokret/shared; wdrożenie i infrastruktura	Aplikacja internetowa (Next.js, React); moduły serwerowe: import danych, eksport danych (w tym kart otworów wraz z szablonami eksportu), system zaproszeń
Tekst pracy	Rozdziały 1, 4, 6, 7, 8; rozdziały 2 i 3 (wspólnie), w tym analityczny diagram klas oraz redakcja końcowa rozdziału 3; podrozdziały 5.1, 5.2, 5.3, 5.4, 5.7, 5.8, 5.10, 5.16, 5.17	Rozdziały 2 i 3 (wspólnie), w tym diagramy stanów i aktywności oraz pierwotna wersja tekstu rozdziału 3; podrozdziały 5.5, 5.6, 5.9, 5.11, 5.12, 5.13, 5.14, 5.15

1.6. Wprowadzenie do problematyki

Praca geologa terenowego obejmuje szereg czynności związanych z rozpoznaniem budowy geologicznej badanego podłoża gruntowego. Podstawową metodą badawczą jest wykonanie odwiertów geologicznych, podczas których geolog dokumentuje napotkane warstwy gruntu określając ich właściwości: litologię, miąższość, wilgotność, plastyczność oraz kolor. W zależności od rodzaju zlecenia wykonywane są również sondowania dynamiczne mające na celu określenie stopnia zagęszczenia gruntów niespoistych oraz wytrzymałość gruntów spoistych. W trakcie badań pobierane są próbki gruntu przeznaczone do dalszych badań laboratoryjnych.

Przebieg prac rozpoczyna się od otrzymania zlecenia zawierającego lokalizację punktów badawczych. Geolog udaje się w teren, wykonuje odwierty przy pomocy wiertnicy, a następnie dokumentuje napotkane warstwy geologiczne. Proces ten wymaga znajomości norm klasyfikacji gruntów, umiejętności rozpoznawania rodzajów gruntów i zapisu dokładnych parametrów każdej

warstwy. Po zakończeniu badania w terenie dane są przekazywane do dokumentatora, który weryfikuje poprawność zapisów i przygotowuje ostateczną dokumentację dla projektu.

W praktyce dokumentacja terenowa prowadzona jest w przeważającej mierze na papierowych formularzach lub w arkuszach kalkulacyjnych. Dane te są następnie ręcznie wprowadzane do specjalistycznego oprogramowania biurowego, takiego jak GeoStar, służącego do generowania końcowej dokumentacji geologicznej. Taki sposób pracy wiąże się z szeregiem problemów. Po pierwsze, ręczne przepisywanie danych jest czasochłonne i podatne na błędy. Po drugie, brak sprawdzania danych w terenie powoduje ryzyko, że nieprawidłowości w danych wykrywane są dopiero na etapie dokumentacji biurowej, co potencjalnie może wymusić powtórzenie badania. Po trzecie, koordynacja pracy zespołów geologów oraz monitorowanie postępu realizacji zleceń odbywa się w sposób nieformalny, bez wsparcia dedykowanych narzędzi informatycznych.

Dodatkowym wyzwaniem jest specyfika pracy terenowej. Geolodzy pracują w miejscach, które mogą być w strefie ograniczonej lub całkowicie niedostępnej łączności sieciowej. Narzędzie przeznaczone do pracy w terenie musi uwzględniać architekturę zorientowaną na pracę *offline* umożliwiającą pełną funkcjonalność w środowisku bez dostępu do Internetu, z automatyczną synchronizacją danych po przywróceniu łączności.

System KRET powstał w odpowiedzi na opisane wyzwania, oferując spójne rozwiązanie obejmujące cyfrową dokumentację, zarządzanie zleceniami, sprawdzanie danych w czasie rzeczywistym oraz eksport danych do formatów zgodnych z oprogramowaniem branżowym.

2. Stan sztuki

Niniejszy rozdział omawia narzędzia informatyczne wspierające prace terenowe geologii inżynierskiej. Analiza obejmuje dominujące produkty komercyjne, rozwiązania otwartoźródłowe, pomocnicze aplikacje terenowe oraz generyczne systemy zarządzania pracami terenowymi używane w pokrewnych branżach. Rozdział kończy się tabelą porównawczą i wskazaniem luk, które wypełnia projektowany system KRET.

2.1. Specyfikacja prac geologii inżynierskiej w Polsce

Geologia inżynierska to dyscyplina zajmująca się rozpoznawaniem warunków geologicznych i geotechnicznych podłoża pod inwestycje budowlane, między innymi drogi, mosty, budynki, linie kolejowe. Wynikiem prac jest dokumentacja geologiczno-inżynierska (DGI) na podstawie, której projektant może zaprojektować odpowiednie posadowienie budynku.

Typowy proces realizacji projektu obejmuje kilka faz. Pierwszą stanowi przygotowanie projektu robót geologicznych, w którym określa się rozmieszczenie otworów badawczych oraz rodzaje przewidywanych badań terenowych. Następnie zespół geologów wykonuje prace polowe, a w tym: wiercenie otworów, sondowanie dynamiczne lub statyczne. Wyniki dokumentowane są w karcie otworu badawczego, zawierającej profil litologiczny rozpoznanych warstw, opis ich miąższości, wilgotności i stanu gruntu, głębokość zwierciadła wody gruntowej oraz dodatkowe informacje o pobranych próbkach. Po pracach polowych następuje etap weryfikacji, w którym geolog dokumentator sprawdza poprawność karty i zatwierdza ją do dalszego opracowania. Kończącym artefaktem jest dokumentacja składana w organach administracji geologicznej oraz przekazywana inwestorowi.

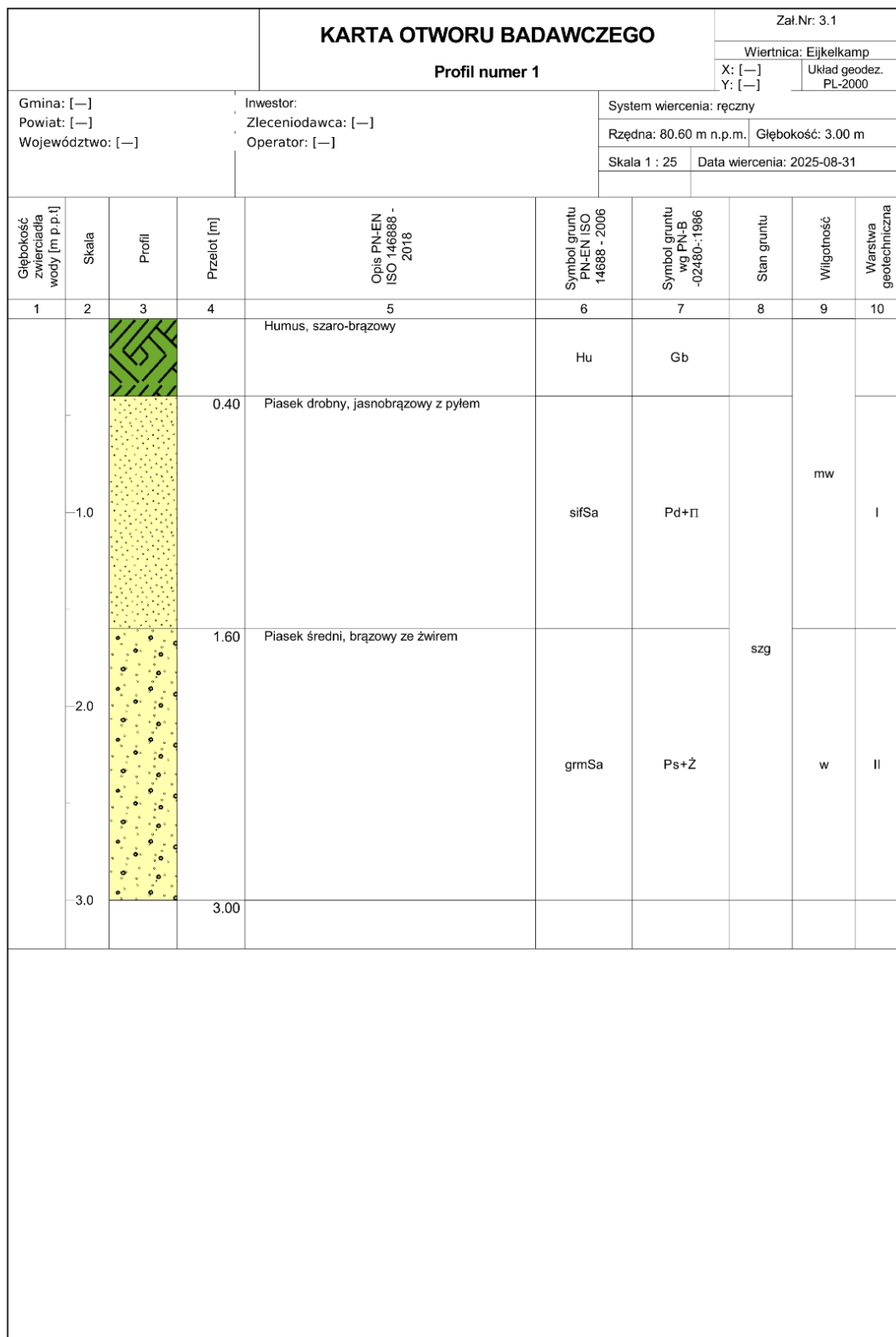
W procesie tym uczestniczą zatem dwie wyraźnie wyróżnione role: geolog terenowy, odpowiedzialny za prace polowe i pierwotne wypełnienie karty otworu, oraz dokumentator, weryfikujący poprawność dokumentacji i nadzorujący jej ostateczne opracowanie.

Praca w terenie cechuje się specyficznymi ograniczeniami infrastrukturalnymi i technologicznymi. Wiercenia są często prowadzone na placach budowy, w lasach oraz na obszarach niezurbanizowanych, gdzie zasięg sieci komórkowej bywa ograniczony lub nieosiągalny. Warunki atmosferyczne takie jak opady, błoto niskie temperatury wymagają stosowania urządzeń o podwyższonej odporności mechanicznej i wodoszczelności oraz interfejsów dostosowanych do obsługi w rękawicach roboczych. Te wymagania determinują projektowanie rozwiązań informatycznych dedykowanych pracy terenowej i odróżniają je od typowych aplikacji biurowych.

Wynikiem prac terenowych jest karta otworu badawczego. Jest to formalnie ustrukturyzowany dokument, którego zawartość regulują polskie normy geotechniczne. Przykładową kartę otworu wygenerowaną przez program GeoStar przedstawia Rysunek 1.

Strukturę karty otworu regulują polskie normy klasyfikacyjne: PN-EN ISO 14688:2018 [1] dla gruntów oraz PN-EN ISO 14689:2018 [2] dla skał. Norma PN-B-02480:1986 [3], choć formalnie zastąpiona, jest nadal powszechnie stosowana w praktyce. Z tego powodu karty otworów generuje się dwukolumnowo, prezentując symbole gruntu w obu konwencjach. Całość procesu dokumentowania prac geologicznych regulowana jest ustawą z dnia 9 czerwca 2011 roku – Prawo geologiczne i górnicze [4], która określa zakres obowiązków wykonawcy oraz wymagania formalne wobec dokumentacji składanej w organach administracji geologicznej.

Karta zawiera kolumny opisujące głębokość zwierciadła wody gruntowej, skalę głębokości otworu, profil litologiczny, symbole gruntów według dwóch norm, a także informacje o stanie gruntu, jego wilgotności oraz przypisanym numerze warstwy geotechnicznej. Niezależnie od formy pierwotnego wypełnienia, końcowy dokument musi zostać przygotowany w formacie zgodnym z wymaganiami organów administracji geologicznej oraz normami branżowymi.



Rysunek wykonano programem "GeoStar" zgodnie z PN-B-02480:1986

Rysunek 1. Karta otworu badawczego wygenerowana przez program GeoStar (dane identyfikujące zostały zanonimizowane). Źródło: Opracowanie własne na podstawie programu GeoStar [5].

2.2. GeoStar (Soft-Projekt, Wrocław)

GeoStar jest najpowszechniej stosowanym polskim systemem informatycznym wspierającym dokumentację prac geologii inżynierskiej. Producentem oprogramowania jest firma Soft-Projekt Jan Szymański z siedzibą w Wrocławiu [5], działająca na rynku od lat 90. XX wieku. Aktualną wersją produktu jest GeoStar 9i, wydany w 2024 roku. Wsparcie wersji 6i zostało zakończone 1 września 2024 roku, a wsparcie obsługi systemów Windows 7/8/8.1 zostało zakończone 1 stycznia 2024 roku.

System jest klasyfikowany w sylabusach akademickich kierunków geologicznych jako standard branżowy w zakresie opracowań dokumentacyjnych oraz graficznych [8], a wśród jego użytkowników wymieniany jest zarówno Państwowy Instytut Geologiczny, przy tworzeniu atlasów geologiczno-inżynierskich [9], jak i komercyjne biura geotechniczne, takie jak Geokrak z Krakowa [10].

Architektura systemu opiera się na aplikacji desktopowej dla systemów Windows 10 i 11, korzystającej z relacyjnej bazy danych (domyślnie Firebird 4, opcjonalnie MySQL, MariaDB lub Oracle [5]). Interfejs organizuje pracę wokół drzewa dokumentów geologicznych zawierającego otwory, sondowania i przekroje. Dedykowana aplikacja mobilna nie wchodzi w skład pakietu. Główny widok pakietu w wersji geologiczno-inżynierskiej przedstawia Rysunek 2.



Rysunek 2. Główny widok pakietu GeoStar 8i Ultimate (wersja 8.5, wydanie 2023) w konfiguracji geologiczno-inżynierskiej. Źródło: Opracowanie własne na podstawie programu GeoStar [5].

Producent oferuje również dodatkowy moduł GeoStar poprzez Internet, udostępniający kartę otworu w przeglądarce internetowej w trybie tylko do odczytu. Rozwiązanie to nie umożliwia wprowadzania danych z poziomu urządzenia mobilnego ani z poziomu placu budowy, lecz pozwala dyspozytorowi biurowemu na wgląd w dokumentację bez konieczności instalacji aplikacji na komputerze.

Funkcjonalność systemu obejmuje generowanie kart dokumentacyjnych otworów wiertniczych zgodnych z polskimi normami klasyfikacji gruntów [1][3] oraz kart sondowań geotechnicznych, w tym sondowań dynamicznych (DPL, DPM, DPH, DPSH) oraz statycznych CPT i CPTu obsługiwanych przez moduł CPT-Star. System pozwala na tworzenie przekrojów geologicznych oraz drogowych według kilometrażu inwestycji liniowej, zawiera słowniki litologii i stratygrafii zgodne z wytycznymi

Mechanizm eksportu danych wspiera szeroki zakres formatów. Rysunki przekrojów oraz kart można wyprowadzać do formatów DXF (akceptowanego przez systemy wspomagania projektowania), EMF, WMF oraz PDF. Dane tabel są eksportowane w formatach TXT, CSV, XLS, XML oraz w binarnym formacie GDB stanowiącym natywną strukturę bazy GeoStar [5]. Od listopada 2022 roku producent oferuje dodatkowo dedykowany moduł eksportu w międzynarodowym formacie AGS 4.1 (Association of Geotechnical and Geoenvironmental Specialists, brytyjska organizacja branżowa), wymaganym w dużych inwestycjach infrastrukturalnych realizowanych w Polsce. W 2025 moduł ten otrzymał nominację do branżowej nagrody TYTAN w obszarze geoinżynierii [7]. Cena modułu AGS nie jest publikowana w cenniku producenta i podlega indywidualnej negocjacji [6].

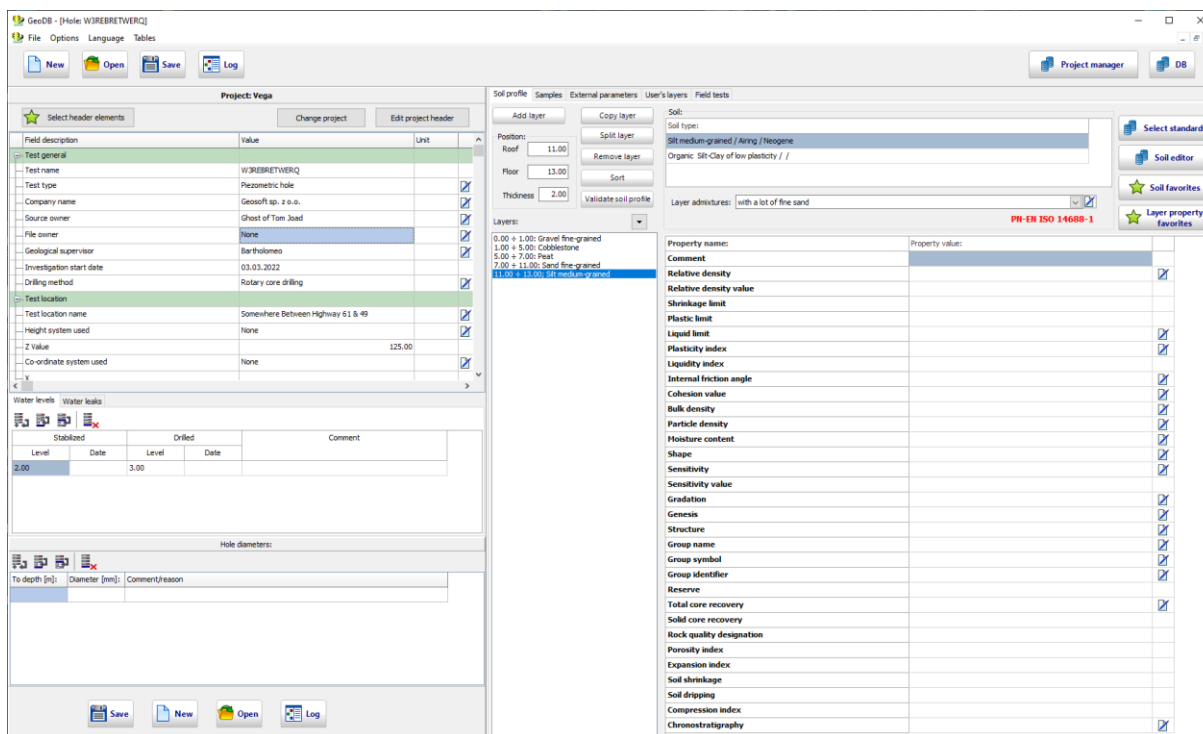
- Lite Small Business za 4 600 zł netto;
- Standard za 8 800 zł netto;
- Professional za 11 500 zł netto;
- Ultimate za 13 500 zł netto.

13

Mimo dominującej pozycji rynkowej GeoStar wykazuje istotne ograniczenia z perspektywy współczesnego procesu prac geologicznych, system nie posiada warstwy mobilnej umożliwiającej wprowadzanie danych bezpośrednio w terenie. Geolog dokumentuje obserwacje na papierze lub w arkuszu kalkulacyjnym, a następnie przepisuje je w biurze do oprogramowania przez komputer, co prowadzi do podwójnej pracy oraz ryzyka błędów w transkrypcji. System nie udostępnia również mechanizmów zarządzania zleceniami, przydziału zadań geologom terenowym ani podziału pracy weryfikacyjnej między dokumentatorów. W szczególności brak otwartego interfejsu programistycznego REST uniemożliwia integrację GeoStar z zewnętrznymi systemami zarządzania pracami terenowymi, co wymusza model wymiany danych oparty na ręcznym imporcie i eksporcie plików.

2.3. Geosoft GEO DB

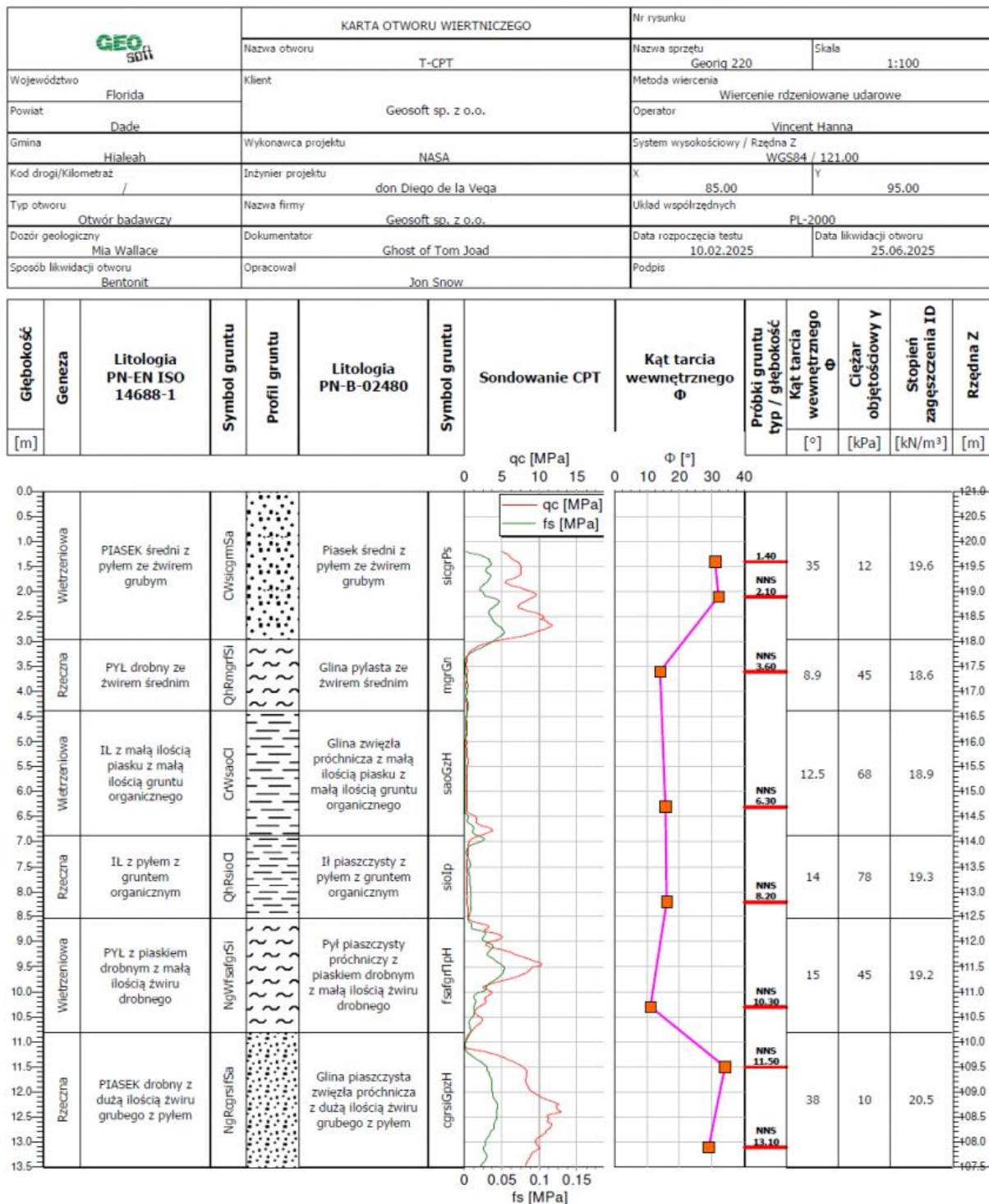
Drugim obok firmy Soft-Projekt polskim producentem oprogramowania geotechnicznego jest Geosoft Sp. z o.o. z siedzibą w Wrocławiu. Spółka działająca na rynku od 1989 roku [11]. Centralnym elementem oferty jest pakiet GEO DB. Jest to baza danych geologicznych i geotechnicznych umożliwiająca rejestrację informacji z otworów wiertniczych, sondowań oraz badań laboratoryjnych, a także generowanie kart otworów, raportów oraz przekrojów geologicznych. Główne okno programu przedstawia Rysunek 4.



Rysunek 4. Główne okno programu Geosoft GEO DB. Źródło: [11].

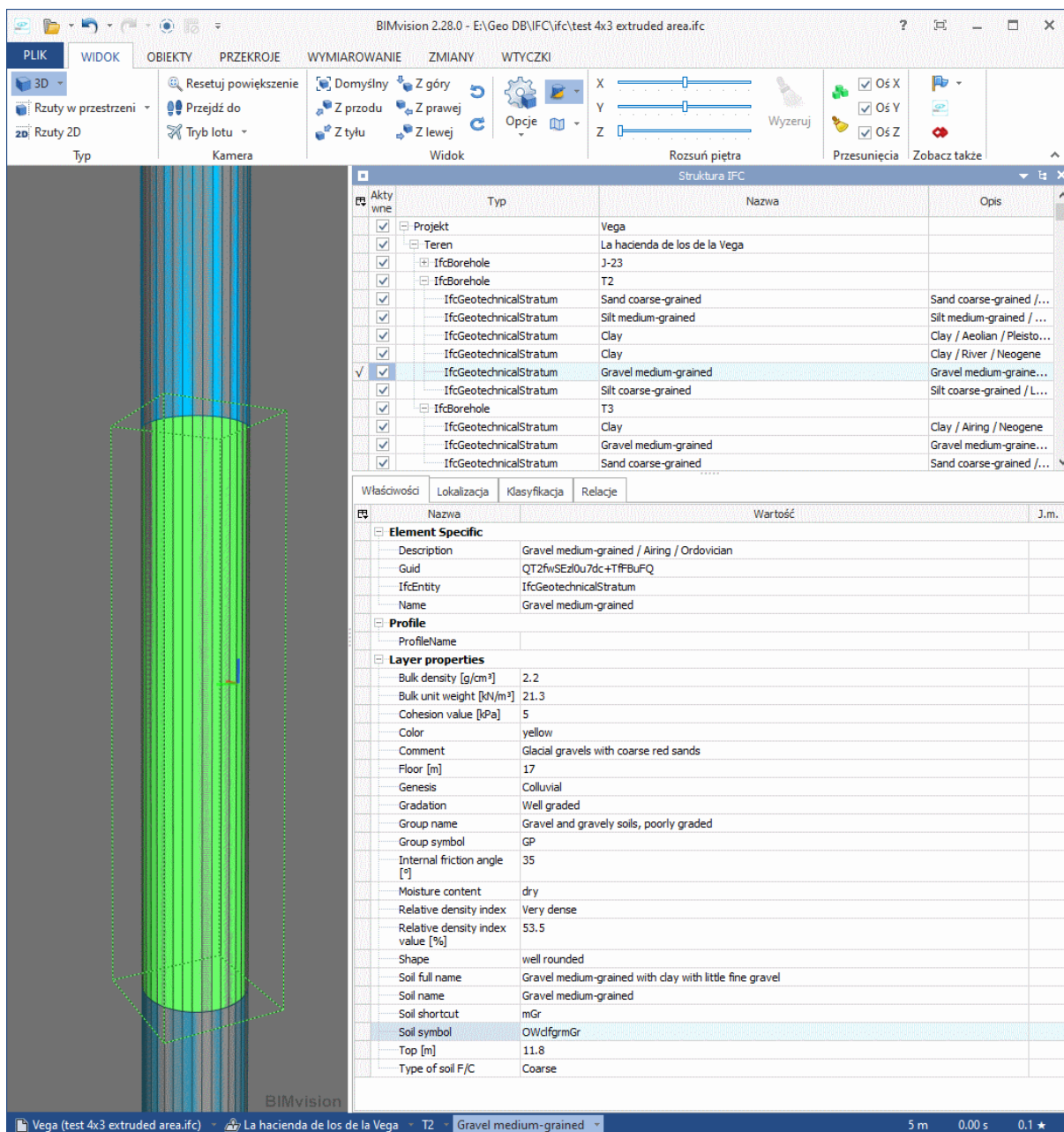
Oferta firmy Geosoft nie ogranicza się do bazy GEO DB. Pakiet uzupełniają moduły wyspecjalizowane: do interpretacji sondowań statycznych, do tworzenia przekrojów geologicznych i map, a także moduł do opracowywania wyników badań sejsmicznych [11]. W odróżnieniu od pakietów GeoStar mechanizm wymiany danych obejmuje również formaty wspomaganie projektowania oraz wbudowaną obsługę formatów danych geotechnicznych AGS i GEF, a także format IFC wykorzystywany w metodyce modelowania informacji o budowlu [11]. Całość działa jako aplikacja dla Windows i nie obejmuje dedykowanej warstwy przeznaczonej na urządzenia mobilne mogącej służyć do wprowadzania danych w terenie.

Karta otworu wytwarzana przez GEO DB jest w pełni konfigurowalna i może być formatowana z profilem litologicznym wypełnianym kolorami lub szrafurami zgodnymi z wytycznymi branżowymi dla projektów drogowych i kolejowych [11]. System pozwala na równoczesne generowanie karty zgodnie z dwiema normami klasyfikacyjnymi: PN-B-02480:1986 [3] oraz PN-EN ISO 14688:2018 [1], co jest cechą oczekiwaną przez polskich klientów ze względu na utrzymującą się dwutorowość praktyki branżowej. Przykład karty otworu opisanej według obu norm jednocześnie przedstawia Rysunek 5.



Rysunek 5. Karta otworu wygenerowana przez Geosoft GEO DB z równoczesną klasyfikacją według norm PN-EN ISO 14688-1 i PN-B-02480. Źródło: [11].

Istotnym wyróżnikiem Geosoft względem GeoStar jest natywne wsparcie międzynarodowych formatów wymiany danych geotechnicznych: standardu AGS, formatu GEF (ang. *Geotechnical Exchange Format*) oraz IFC (ang. *Industry Foundation Classes*, otwarty format wymiany danych w technologii modelowania informacji o budowlach *Building Information Modeling*, BIM). Wsparcie tych formatów jest wbudowane w standardową funkcjonalność programu, bez konieczności zakupu dodatkowych modułów [11], w odróżnieniu od konkurencyjnego GeoStar, gdzie eksport AGS wymaga oddzielnej, płatnej licencji. Wynik eksportu pojedynczego otworu do formatu IFC, otwarty w przeglądarce BIM, prezentuje Rysunek 6.



Rysunek 6. Plik IFC wygenerowany w Geosoft GEO DB i otwarty w przeglądarce BIMvision – ilustracja natywnego wsparcia standardu BIM. Źródło: [11].

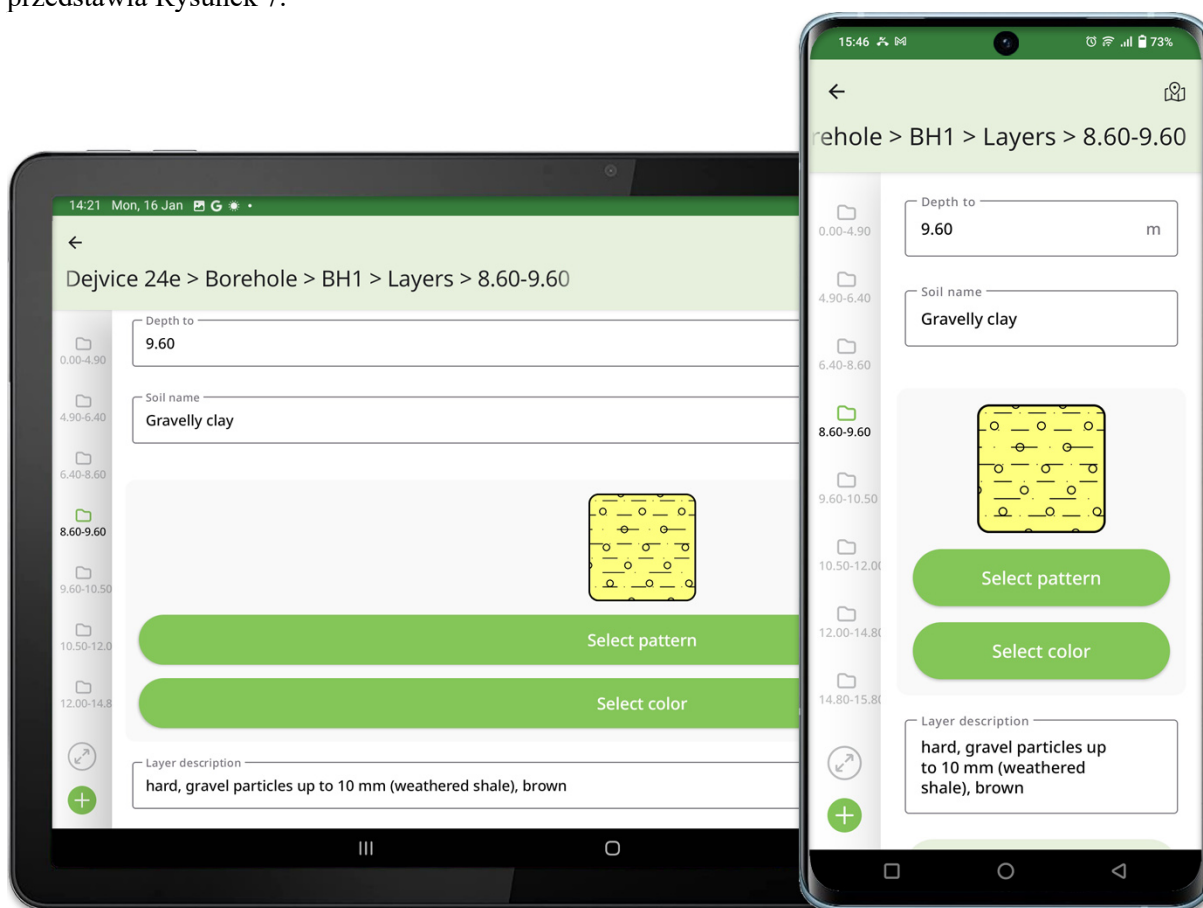
Geosoft podobnie jak GeoStar pozostaje aplikacją desktopową dla systemów Windows, bez warstwy mobilnej wprowadzania danych w terenie ani mechanizmów zarządzania zleceniami i przydziału zadań. Producent koncentruje swoją ofertę na profesjonalnej dokumentacji geotechnicznej

dla dużych projektów infrastrukturalnych z wykorzystaniem technologii BIM. Cennik nie jest publikowany na stronie producenta i ustalany jest indywidualnie z klientem. Oprogramowanie integruje się ze środowiskami CAD oraz wspiera opracowania zgodne z Eurokodem 7 [11].

2.4. GEO5

GEO5 jest jedynym istotnym międzynarodowym konkurentem na polskim rynku oprogramowania dla geologii inżynierskiej. Producentem jest czeska firma Fine spol. s r.o. z siedzibą w Pradze, oferująca w Polsce swoje produkty za pośrednictwem dystrybutora MMGEO z Warszawy [12]. W odróżnieniu od polskich producentów koncentrujących się na dokumentacji geologicznej, GEO5 stanowi szeroki ekosystem programów obejmujący zarówno moduły dokumentacyjne, jak i specjalistyczne narzędzia obliczeniowe geotechniki (analiza stateczności zboczy, projektowanie ścian oporowych, fundamentów, pali).

Najistotniejszym wyróżnikiem GEO5 z perspektywy niniejszej pracy jest dedykowana aplikacja mobilna GEO5 Data Collector dostępna bezpłatnie na platformach Android oraz iOS [12]. Aplikacja umożliwia gromadzenie danych z badań polowych, opisów warstw, pomiarów, zdjęć, lokalizacji GPS i przysyłanie ich do programu desktopowego Stratigraphy w celu dalszego opracowania. Jest to w dużej mierze funkcjonalność wykraczająca poza możliwości polskich konkurentów, ponieważ ani GeoStar ani Geosoft nie posiadają warstwy mobilnej. Widok formularza edycji warstwy litologicznej w terenie przedstawia Rysunek 7.

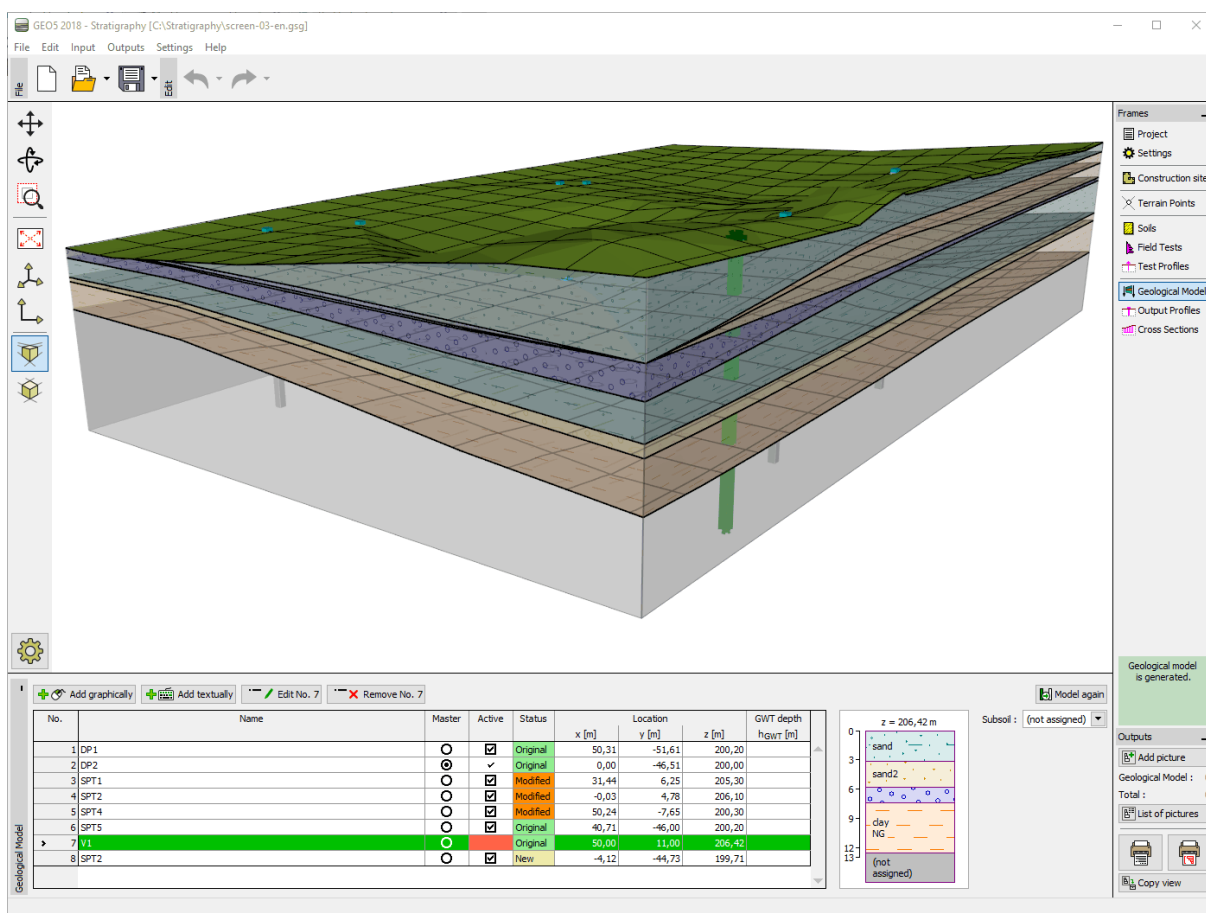


Rysunek 7. Aplikacja mobilna GEO5 Data Collector – edycja warstwy litologicznej w terenie. Źródło: [12].

Eksport danych odbywa się natywnie do formatów PDF, Microsoft Word, BIM/IFC oraz AGS, bez konieczności zakupu dodatkowych modułów. Producent oferuje wbudowane szablony kart otworów z możliwością tworzenia własnych, co umożliwia dostosowanie dokumentacji do wymagań poszczególnych krajów i klientów [12].

Model cenowy GEO5 jest oparty na rocznej subskrypcji, w odróżnieniu od dominującego na polskim rynku modelu wieczystej licencji. W cenniku obowiązującym od 1 listopada 2025 roku Pakiet Geologiczny, obejmujący moduły Stratigraphy, Logs, Cross Sections oraz Earthworks, kosztuje 1750 zł rocznie w wariantcie Personal, 2 280 zł w wariantcie Business oraz 3 500 zł w wariantcie Enterprise [13]. Pełny pakiet Professional zawierający wszystkie programy GEO5 jest dostępny w cenie 8 000 euro rocznie. Od edycji 2025 producent stosuje Fine Cloud License, licencje zarządzane są w chmurze, a program weryfikuje je przy starcie, wymagając połączenia z Internetem, możliwe jest tymczasowe lub trwale przeniesienie licencji na komputer, umożliwiające pracę bez dostępu do sieci [12].

Mimo posiadania aplikacji mobilnej GEO5 nie pełni roli systemu zarządzania pracami terenowymi w sensie operacyjnym. Data Collector jest aplikacją, która wymaga ekosystemu desktopowego i synchronizacji z programem Stratigraphy. Środowisko desktopowe programu GEO5 Stratigraphy wraz z wygenerowanym modelem 3D podłoża prezentuje Rysunek 8. Nie udostępnia mechanizmów przydziału zadań geologom, nie posiada mapy zleceń ze statusami otworów ani otwartego interfejsu programistycznego umożliwiającego integrację z zewnętrznymi systemami zarządzania zespołem.

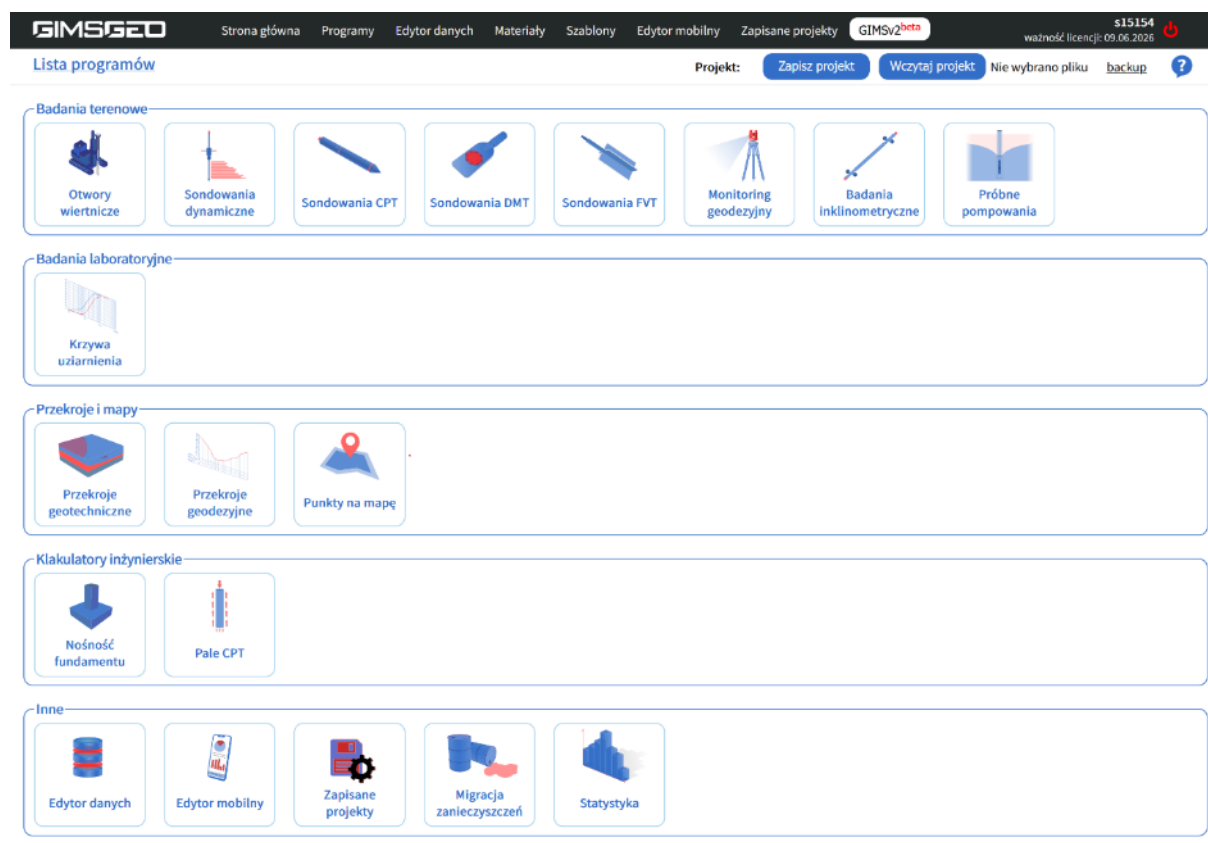


Rysunek 8. Program GEO5 Stratigraphy – trójwymiarowy model podłoża gruntowego wygenerowany z danych otworów badawczych. Źródło: [12].

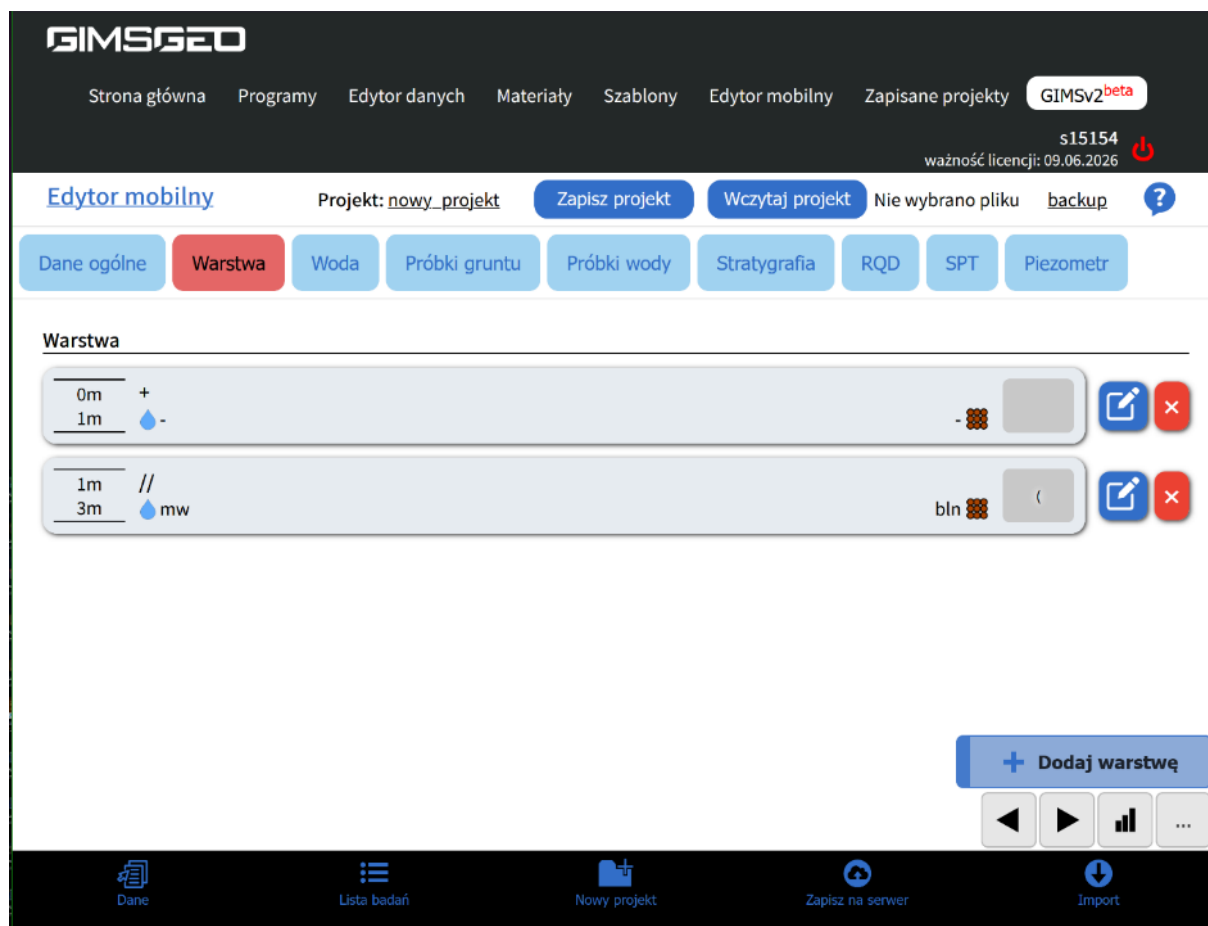
2.5. GIMSGEO

GIMSGEO (Geological Information Management System) jest internetową platformą informatyczną dla geologii inżynierskiej i geotechniki, rozwijaną przez Łukasza Zawadzkiego z Warszawy w ramach jednoosobowej działalności gospodarczej [14]. W odróżnieniu od opisanych wcześniej produktów desktopowych, GIMSGEO funkcjonuje jako aplikacja internetowa dostępna z poziomu przeglądarki i nie wymaga instalacji oprogramowania na komputerze użytkownika. Widok głównego menu aplikacji z listą dostępnych modułów przedstawia Rysunek 9.

System składa się z modułów obejmujących między innymi prowadzenie kart otworów wiertniczych, sondowań, badania inklinometryczne, monitoring geodezyjny oraz generowanie przekrojów geotechnicznych. Wszystkie moduły działają na wspólnej bazie danych, co eliminuje konieczność wielokrotnego wprowadzania tych samych informacji [14]. Karta otworu wytwarzana jest na podstawie konfigurowalnych szablonów i może zawierać wyniki badań laboratoryjnych oraz wykresy parametrów gruntu. Edytor mobilny aplikacji w widoku edycji warstw otworu prezentuje Rysunek 10.



Rysunek 9. Główny widok aplikacji internetowej GIMSGEO z listą dostępnych modułów. Źródło: Opracowanie własne na podstawie programu GIMSGEO [14].



Rysunek 10. Moduł "Edytor mobilny" GIMSGEO w widoku przeglądarki mobilnej. Źródło: Opracowanie własne na podstawie programu GIMSGEO [14].

GIMSGEO oferowany jest w modelu subskrypcyjnym w cenie 490 zł brutto miesięcznie za pojedyncze stanowisko, co odpowiada kwocie 5 880 zł brutto rocznie. Producent udostępnia również bezpłatną wersję testową na okres 14 dni oraz wycenę indywidualną dla zespołów wieloosobowych [14]. System udostępnia edytor danych dostępny z poziomu przeglądarki na urządzeniu mobilnym, jednak nie wspiera pracy w trybie *offline*. Pełna funkcjonalność wymaga stałego połączenia z Internetem, co stanowi istotne ograniczenie w warunkach pracy terenowej geologa, gdzie zasięg sieci bywa nieosiągalny. Producent nie udostępnia również publicznej dokumentacji interfejsu programistycznego REST umożliwiającego integrację z systemami zewnętrznymi.

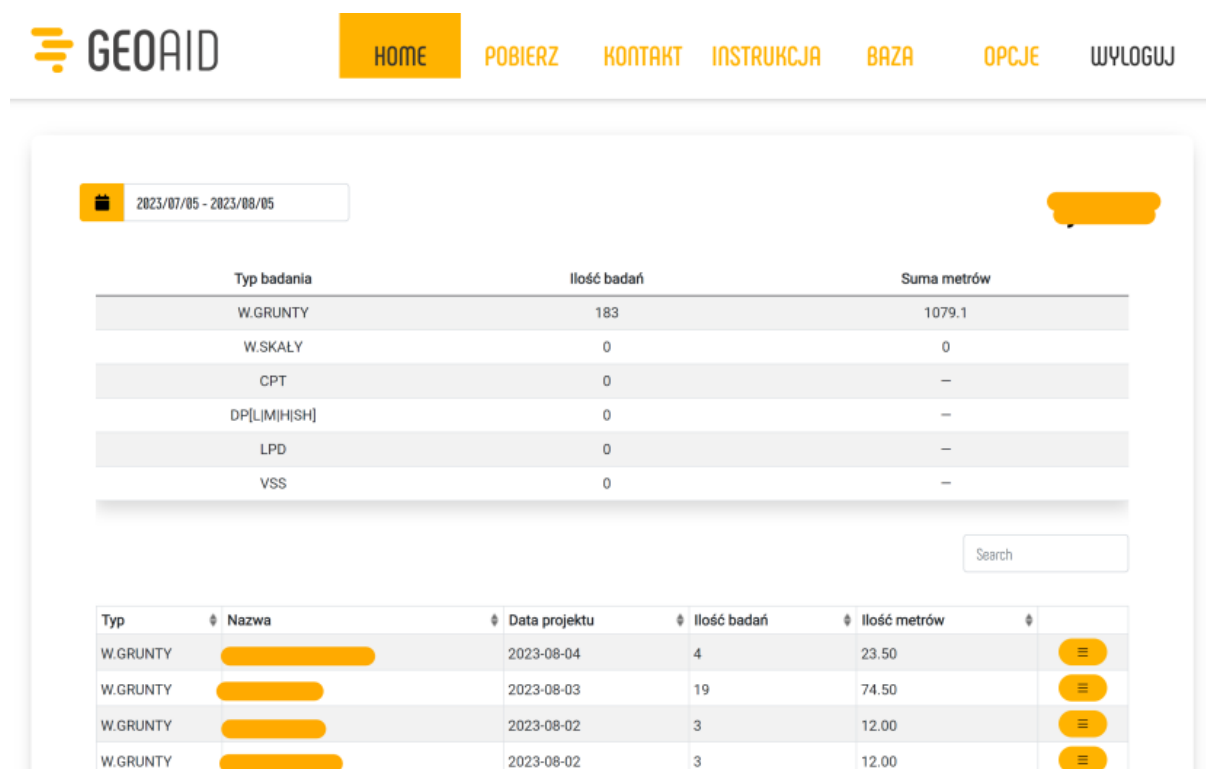
2.6. GeoAid

GeoAid jest produktem najbardziej zbliżonym funkcjonalnie do projektowanego systemu KRET spośród wszystkich rozwiązań omawianych w tym rozdziale. Producentem jest spółka GeoAid Sp. z o.o. z siedzibą w Krasnem, powstała w 2021 roku jako wyodrębniony projekt z biura geologicznego Drimgeo, sfinansowany ze środków unijnych w ramach Programu Operacyjnego Inteligentny Rozwój [15]. Podobnie jak KRET, system łączy internetowy panel zarządzania pracami z aplikacją mobilną przeznaczoną do wprowadzania danych w terenie, pozycjonując się jako rozwiązanie operacyjne dla biur geologii inżynierskiej, a nie jedynie narzędzie biurowe.

Aplikacja mobilna GeoAid dostępna jest wyłącznie dla systemu Android, producent nie oferuje wersji dla systemu iOS. Rekomendowanym sprzętem są wzmocnione tablety polowe Samsung Galaxy

Tab Active 2 lub Tab Active 3, dostosowane do pracy w trudnych warunkach atmosferycznych, posiadające rysik pozwalający na obsługę w zabłoconych rękawicach roboczych oraz konstrukcję umożliwiającą umycie urządzenia pod bieżącą wodą [15]. Synchronizacja danych pomiędzy terenem a biurem odbywa się automatycznie po nawiązaniu połączenia internetowego.

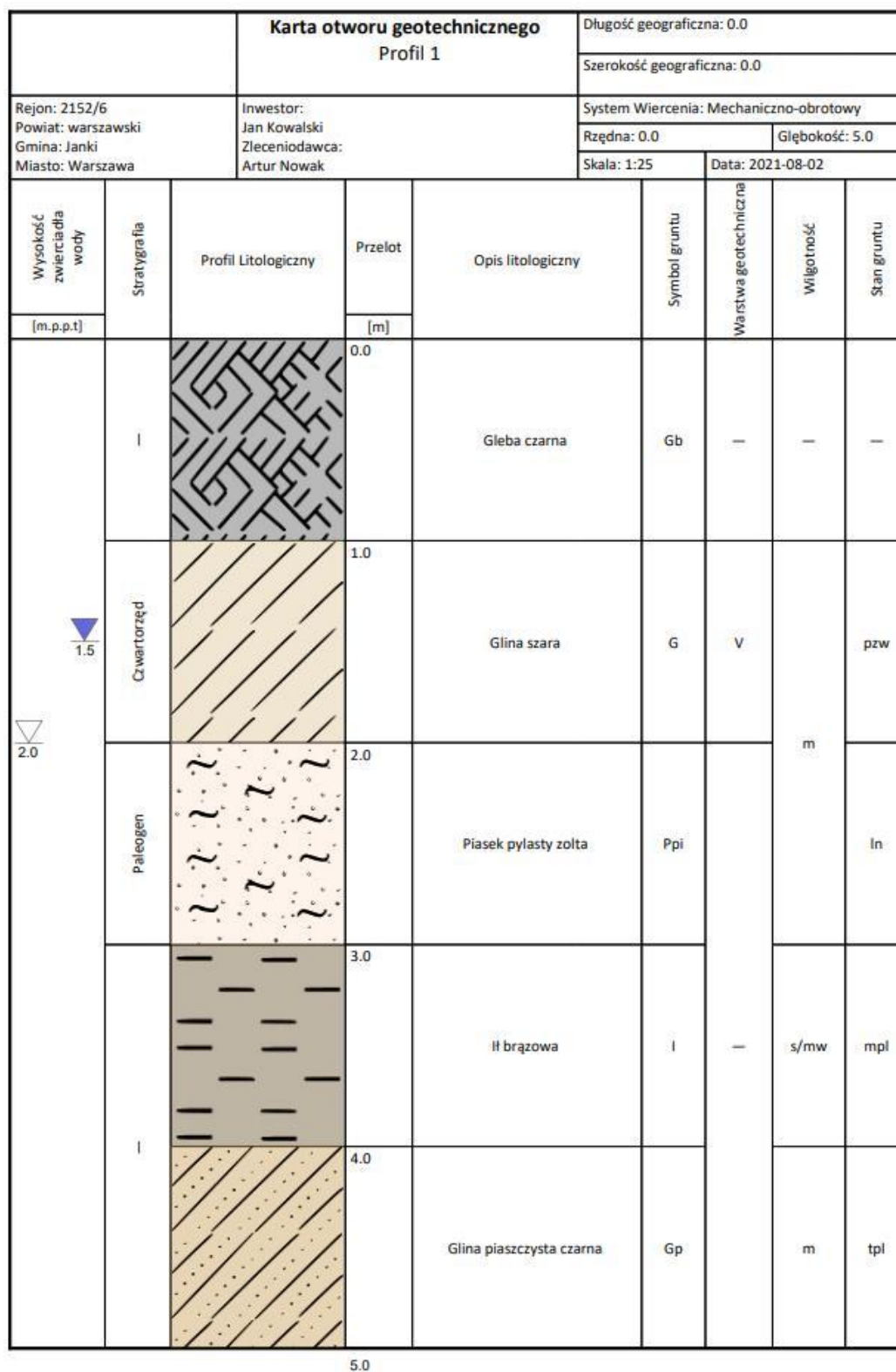
Funkcjonalność systemu obejmuje definiowanie projektów geologicznych, wprowadzanie punktów badań, dokumentację wierceń oraz sondowań. We wrześniu 2024 roku producent wprowadził funkcję mapy pokazującej postęp prowadzonych badań, dostępną w aplikacji mobilnej, jak i w panelu internetowym, co umożliwia kierownikom prac zdalne śledzenie postępów projektu. W sierpniu 2025 roku dodano warstwy z usług sieciowych (Web Map Service, WMS) zawierające ewidencję gruntów oraz przebieg sieci uzbrojenia podziemnego, co ułatwia geologowi terenowemu lokalizację badań względem elementów infrastruktury [15]. Interfejs panelu GeoAid prezentuje Rysunek 11, a kartę otworu wygenerowaną przez platformę GeoAid Rysunek 12.



Rysunek 11. Panel GeoAid – widok statystyki i listy zrealizowanych badań. Źródło: [15].

Eksport wyników terenowych odbywa się w trzech formatach: GDB (binarny format bazy danych), DXF oraz PDF [15]. Cennik nie jest publikowany na stronie producenta i ustalany jest indywidualnie po kontakcie z firmą.

Mimo zbliżonej koncepcji architektonicznej GeoAid wykazuje istotne ograniczenia, które wpływają na jego zastosowalność i pozycję rynkową. Brak wersji aplikacji dla systemu iOS wymusza na klientach inwestycje wyłącznie w urządzenia z systemem Android. Producent nie udostępnia również publicznej dokumentacji interfejsu programistycznego REST, co ogranicza możliwości integracji systemu z zewnętrznymi narzędziami. Funkcjonalność mapy pokazującej postęp prac, choć stanowi istotny krok w kierunku zarządzania operacyjnego, nie obejmuje w pełni cyklu życia zlecenia od jego utworzenia, przez przydział poszczególnym zadań geologom terenowym, aż do weryfikacji dokumentacji przez dokumentatora. To są mechanizmy, które dopiero KRET wprowadza jako spójny proces operacyjny.



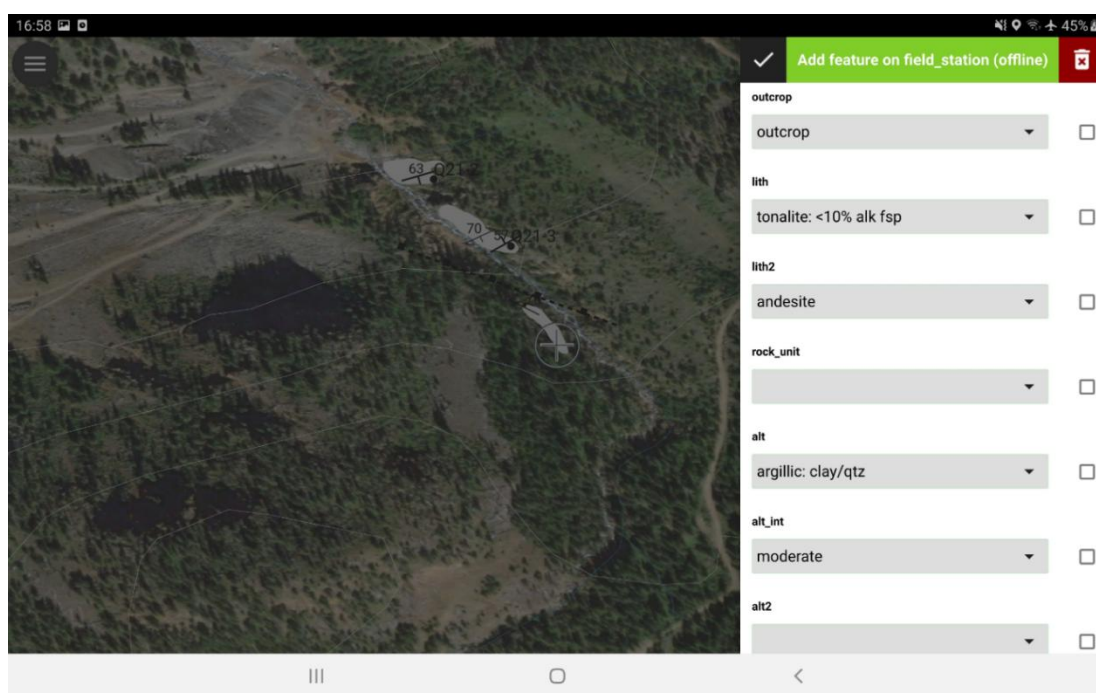
Rysunek 12. Karta otworu geotechnicznego wygenerowana przez platformę GeoAid. Źródło: [15].

2.7. Rozwiązania otwartoźródłowe (QField, Mergin Maps, BGS·SIGMA)

Niezależnie od polskiego rynku komercyjnego rozwijają się otwartoźródłowe narzędzia wspierające terenowe mapowanie geologiczne, wykorzystywane głównie przez instytucjonalne służby geologiczne oraz w środowisku akademickim. Trzy z nich zasługują na omówienie ze względu na wysoki poziom dojrzałości technologicznej oraz sprawdzone wdrożenia w skali międzynarodowej.

2.7.1. QField

QField, rozwijany przez szwajcarską firmę OPENGIS.ch i udostępniany na licencji GPL (General Public License), jest mobilną aplikacją bazującą na otwartym systemie informacji geograficznej QGIS [16]. Pozwala na otwieranie w terenie projektów GIS przygotowanych wcześniej na komputerze biurowym i wprowadzanie danych pomiarowych w trybie *offline*, z późniejszą synchronizacją po nawiązaniu łączności. Aplikacja jest dostępna zarówno dla systemu Android, jak i iOS. Aplikację mobilną przedstawia Rysunek 13.

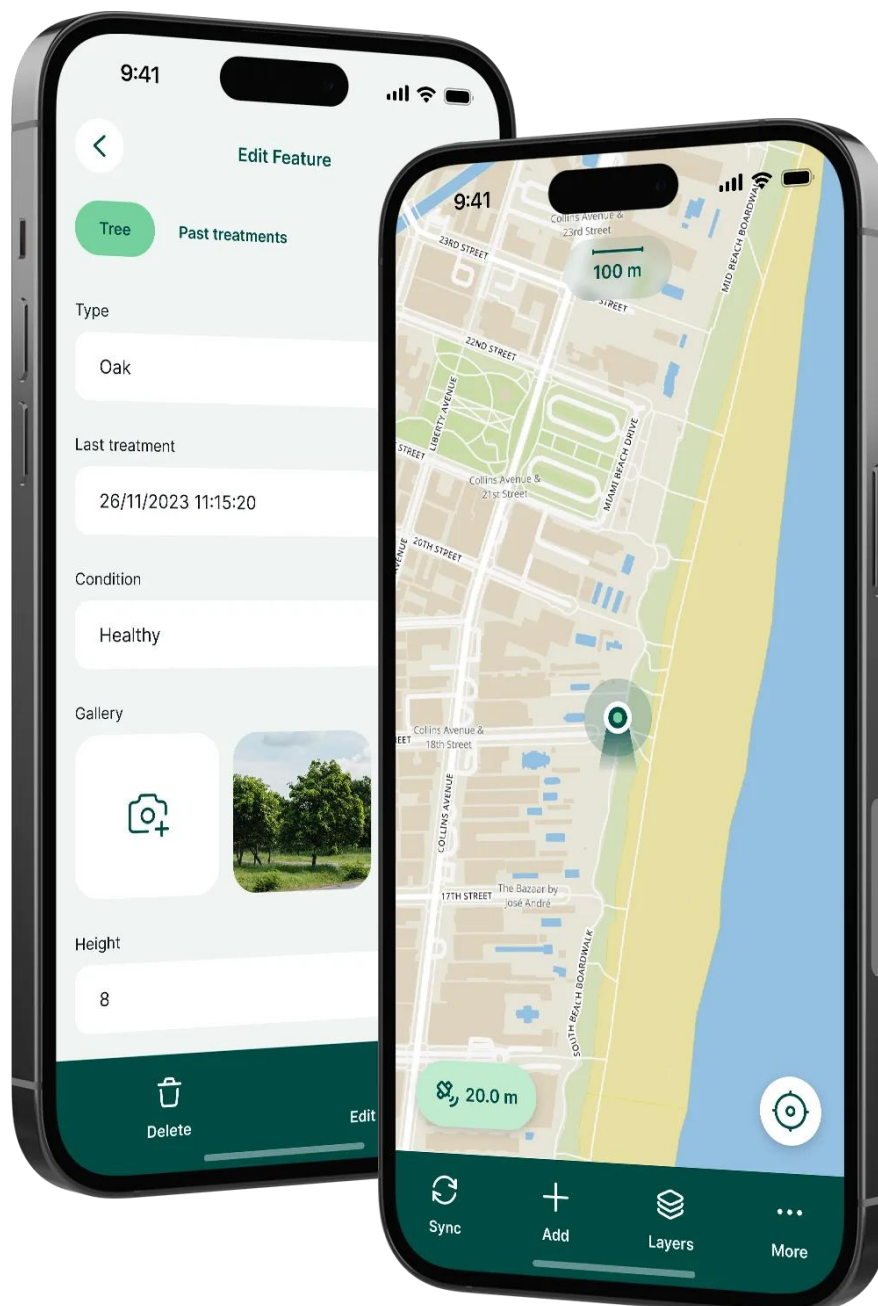


Rysunek 13. Aplikacja mobilna QField wykorzystywana do mapowania geologicznego – formularz wprowadzania obserwacji geologa strukturalnego. Źródło: [16].

2.7.2. Mergin Maps

Mergin Maps to platforma rozwijana przez brytyjską firmę Lutra Consulting, stanowiąca usługę synchronizacji projektów QGIS pomiędzy komputerem biurowym a urządzeniami mobilnymi. Platforma obsługuje pracę zespołową wielu geologów w terenie z mechanizmami łączenia zmian wprowadzanych równolegle przez różnych użytkowników [17].

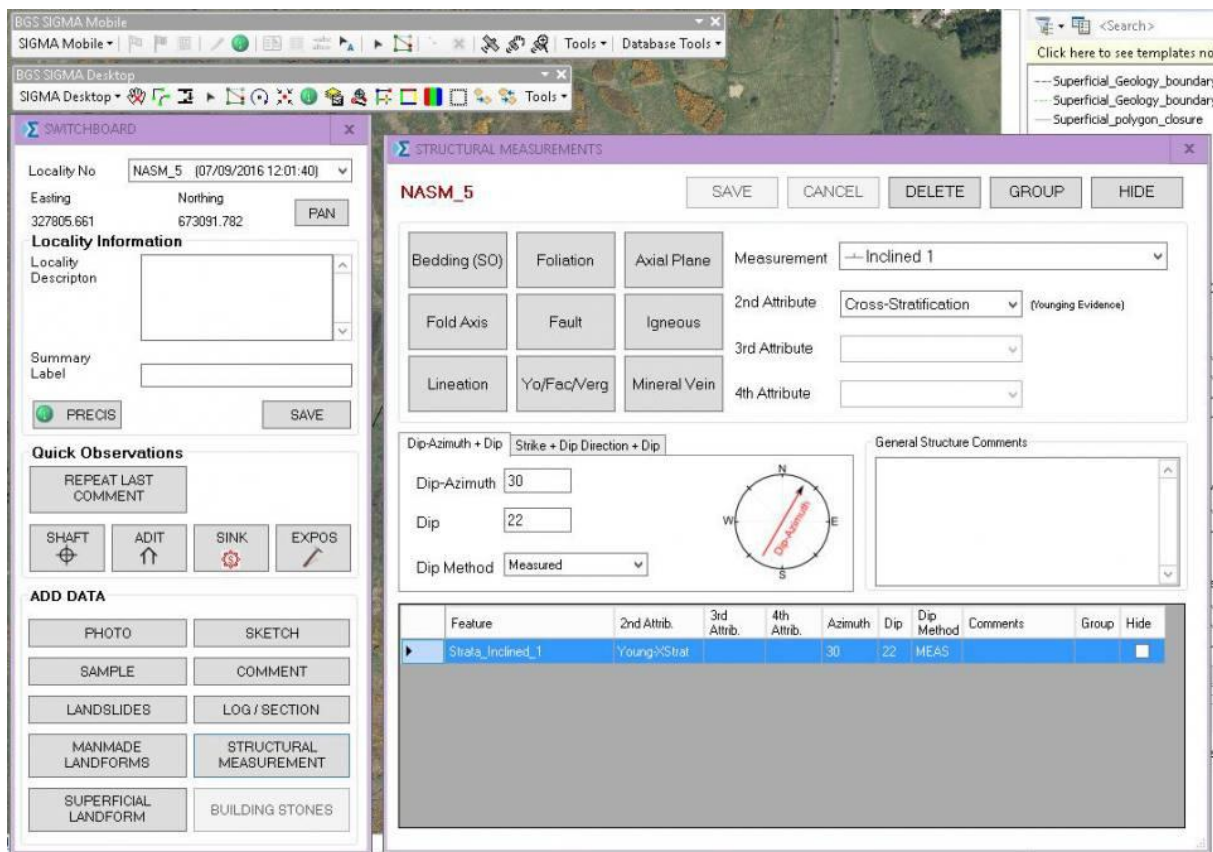
W marcu 2024 roku, w odpowiedzi na zakończenie przez firmę Esri wsparcia dla produktu ArcGIS Desktop, brytyjska państwowa służba geologiczna British Geological Survey podjęła decyzję o migracji systemu mapowania terenowego na rozwiązanie otwartoźródłowe oparte o QGIS oraz Mergin Maps. W ciągu pierwszego roku eksploatacji nowego systemu udokumentowano ponad 2000 lokalizacji terenowych [17]. Interfejs aplikacji mobilnej przedstawia Rysunek 14.



Rysunek 14. Aplikacja mobilna Mergin Maps – edycja atrybutów obiektu terenowego oraz widok mapy z lokalizacją. Źródło: [17].

2.7.3. BGS·SIGMA

BGS·SIGMA stanowi historycznie najbardziej zaawansowany instytucjonalny system mapowania geologicznego, rozwijany przez British Geological Survey od 2001 roku jako zestaw narzędzi opartych na ESRI ArcGIS. System został udostępniony bezpłatnie międzynarodowym partnerom w liczbie około 3000 licencji, wykorzystywanym przez inne służby geologiczne, uniwersytety i firmy poszukiwawcze [18]. Wraz z opisaną wyżej migracją British Geological Survey na platformę QGIS rozwój BGS·SIGMA został zakończony. Produkt znajduje się obecnie w fazie schyłkowej. Interfejs systemu BGS·SIGMA przedstawia Rysunek 15.



Rysunek 15. Interfejs systemu BGS-SIGMA – zasobniki BGS-SIGMA Mobile i BGS-SIGMA Desktop w środowisku ESRI ArcMap z otwartym oknem pomiarów strukturalnych. Źródło: [18].

Mimo wysokiej dojrzałości technologicznej żadne z opisanych rozwiązań nie odpowiada specyfice polskiej geologii inżynierskiej. Wszystkie trzy narzędzia są zorientowane na kartowanie geologiczne w skali regionalnej, rozpoznawanie i rejestrację jednostek litostratygraficznych na potrzeby tworzenia map geologicznych, a nie na dokumentowanie pojedynczych otworów wiertniczych zgodnie z polskimi normami klasyfikacyjnymi PN-EN ISO 14688 oraz PN-B-02480. Nie generują kart otworu w formacie wymaganym przez organy administracji geologicznej, nie obsługują polskiego procesu pracy obejmującego rozdzielenie ról geologa terenowego i dokumentatora ani nie integrują się z polskimi standardami wymiany danych geotechnicznych. Stanowią zatem inspirację technologiczną w zakresie architektury zorientowanej na pracę *offline* oraz synchronizacji projektów, lecz nie są bezpośrednią konkurencją dla rozwiązań dedykowanych polskiej praktyce branżowej.

2.8. Generyczne systemy zarządzania pracami terenowymi

Mechanizmy operacyjne projektowane w systemie KRET takie jak automatyczny przydział zadań pracownikom, mapa zleceń ze statusami, konfigurowalne szablony formularzy, otwarty interfejs programistyczny nie są wynalazkiem oryginalnym. Stanowią one standardowy zestaw funkcji oprogramowania klasy „zarządzanie pracami terenowymi” (ang. *field service management*, FSM), wykorzystywanego przez przedsiębiorstwa działające w sektorze usług serwisowych, instalacyjnych oraz utrzymaniowych.

Światowym liderem rynku FSM jest Salesforce Field Service, rozwijany jako rozszerzenie platformy Salesforce. Produkt udostępnia konsolę dyspozytora prezentującą zlecenia w postaci wykresu Gantta, mechanizm przydziału zadań na podstawie konfigurowalnych reguł uwzględniających

kompetencje pracownika, jego lokalizację oraz dostępność, a także aplikację mobilną dla systemów Android i iOS obsługującą tryb *offline* [19]. Platforma oferuje również edytor pozwalający na budowanie konfigurowalnych formularzy do zbierania danych terenowych, integrację z głównym ekosystemem Salesforce oraz publiczny interfejs programistyczny REST.

ServiceTitan jest przykładem rozwiązania branżowego, dedykowanego usługom domowym takim jak ogrzewanie, wentylacja, klimatyzacja, hydraulika oraz elektryka. System udostępnia funkcje planowania i dyspozycji pracy personelu terenowego oraz mechanizm Dispatch Pro wykorzystujący uczenie maszynowe do automatycznego dopasowania pracownika do zlecenia na podstawie historycznych danych [20].

Mimo dojrzałości funkcjonalnej i obecności wszystkich mechanizmów operacyjnych charakterystycznych dla projektowanego systemu KRET, rozwiązania te nie odpowiadają specyfice polskiej geologii inżynierskiej. Są to systemy uniwersalne, zorientowane na obsługę zleceń konsumenckich i serwisowych, które nie używają polskich norm klasyfikacji gruntów, nie generują kart otworu w formacie wymaganym przez organy administracji geologicznej, nie obsługują rozdzielania ról geologa terenowego i dokumentatora ani specyfiki dokumentowania profilu litologicznego.

Adaptacja takiego systemu do potrzeb biura geologicznego wymagałaby kosztownej konfiguracji oraz integracji z zewnętrznymi narzędziami branżowymi, której koszt w praktyce przerasta budżety małych i średnich przedsiębiorstw geologicznych stanowiących większość polskiego rynku.

2.9. Porównanie systemów

Zestawienie wszystkich omówionych w niniejszym rozdziale rozwiązań wraz z projektowanym systemem KRET przedstawiają Tabela 2 oraz Tabela 3. Pierwsza z nich obejmuje cechy merytoryczne związane z domeną zastosowania produktu – architekturę systemu, dostępność warstwy mobilnej oraz zgodność z polskimi normami dokumentacyjnymi i międzynarodowym standardem wymiany danych geotechnicznych. Druga przedstawia cechy operacyjne związane z zarządzaniem pracami terenowymi - wsparcie pracy w trybie *offline*, dostępność mapowego widoku zleceń, mechanizmy automatycznego przydziału zadań oraz otwartość integracyjną.

Tabela 2. Porównanie systemów pod względem cech dokumentacyjnych. Źródło: Opracowanie własne.

System	Producent / kraj	Architektura	Karta wg PN	AGS
GeoStar	Soft-Projekt, PL	aplikacja stacjonarna + przeglądarka (odczyt)	tak	tak (płatny)
GeoSoft GEO DB	Geosoft, PL	aplikacja stacjonarna	tak	tak
GEO5	Fine Software, CZ	aplikacja stacjonarna + mobilna (Android, iOS)	częściowo	tak
GIMSGEO	Łukasz Zawadzki, PL	aplikacja internetowa + mobilna	tak	nie
GeoAid	GeoAid Sp. z o.o., PL	aplikacja internetowa + mobilna (Android)	tak	nie
QField	OPENGIS.ch, CH	aplikacja mobilna (Android, iOS)	nie	nie
Mergin Maps	Lutra Consulting, UK	aplikacja stacjonarna QGIS + mobilna (Android, iOS)	nie	nie

System	Producent / kraj	Architektura	Karta wg PN	AGS
BGS·SIGMA	British Geological Survey, UK	aplikacja stacjonarna + tablet (Windows)	nie	nie
Salesforce Field Service	Salesforce, US	aplikacja internetowa + mobilna (Android, iOS)	nie	nie
ServiceTitan	ServiceTitan, US	aplikacja internetowa + mobilna (Android, iOS)	nie	nie
KRET	PJATK (projekt inżynierski)	aplikacja internetowa + mobilna (Android, iOS)	tak	planowany

Pierwsze zestawienie ujawnia wyraźny podział środowiskowy: polskie produkty geotechniczne (GeoStar, GeoSoft, GIMSGEO, GeoAid) oraz GEO5 generują karty otworu zgodne z normami klasyfikacyjnymi obowiązującymi w polskiej geologii inżynierskiej, podczas gdy rozwiązania zachodnie i otwartoźródłowe nie obsługują tej warstwy dokumentacyjnej.

Wsparcie międzynarodowego standardu AGS pozostaje natomiast domeną wyłącznie produktów dedykowanych geotechnice – nie występuje w generycznych systemach zarządzania pracami terenowymi. Drugie zestawienie ukazuje natomiast komplementarną perspektywę zarządzania operacyjnego, w której role producentów ulegają zasadniczemu odwróceniu.

Tabela 3. Porównanie systemów pod względem cech operacyjnego zarządzania pracami terenowymi.

Źródło: Opracowanie własne.

System	Tryb offline	Mapa zleceń	Automatyczne rozdzielanie zadań	REST API
GeoStar	nie	nie	nie	nie
GeoSoft GEO DB	nie	nie	nie	nie
GEO5	częściowo	nie	nie	nie
GIMSGEO	częściowo	nie	nie	nie
GeoAid	tak	częściowo	nie	nie
QField	tak	nie	nie	tak
Mergin Maps	tak	nie	nie	tak
BGS·SIGMA	tak	nie	nie	nie
Salesforce Field Service	tak	tak	tak	tak
ServiceTitan	tak	tak	tak	tak
KRET	tak	tak	tak	tak

Zestawienie w obu tabelach prowadzi do kluczowej obserwacji stanowiącej punkt wyjścia dalszej analizy: na polskim rynku żaden istniejący produkt nie łączy generowania karty otworu zgodnej z polskimi normami klasyfikacyjnymi z pełnym zestawem mechanizmów operacyjnego zarządzania pracami terenowymi. Polskie produkty geotechniczne (GeoStar, Geosoft, GIMSGEO, GeoAid) generują karty otworu zgodne z normami, lecz nie udostępniają mechanizmów charakterystycznych dla

oprogramowania zarządzania pracami terenowymi. Zachodnie systemy zarządzania pracami terenowymi (Salesforce Field Service, ServiceTitan) udostępniają wszystkie mechanizmy operacyjne, lecz nie obsługują polskiej dokumentacji geologicznej. Otwartoźródłowe narzędzia globalne (QField, Mergin Maps, BGS·SIGMA) sprawdziły się w mapowaniu geologicznym, lecz nie odpowiadają specyfice polskiej geologii inżynierskiej. Te obserwacje stanowią podstawę szczegółowej analizy luk rynkowych przedstawionej w kolejnym podrozdziale.

2.10. Analiza luk i pozycjonowanie systemu KRET

Przedstawiony przegląd dziewięciu rozwiązań, pięciu komercyjnych produktów polskich i czeskich, trzech otwarto źródłowych narzędzi globalnych oraz dwóch generycznych systemów zarządzania pracami terenowymi, pozwala na zidentyfikowanie pięciu istotnych luk w obecnej ofercie rynkowej dla geologii inżynierskiej w Polsce.

Pierwsza luka dotyczy braku zintegrowanego systemu internetowo-mobilnego obsługującego pełny cykl pracy biura geologicznego od przyjęcia zlecenia, przez dokumentację terenową, weryfikację dokumentatora, aż po eksport końcowy. Polskie systemy desktopowe (GeoStar, GeoSoft, GIMSGEO) koncentrują się na warstwie biurowej, a GeoAid, najbliższy koncepcyjnie, ogranicza się do platformy Android i nie obejmuje pełnego procesu z rozdziałem ról.

Druga luka dotyczy braku otwartych interfejsów programistycznych REST w polskich produktach geotechnicznych. Żaden z opisanych systemów nie udostępnia publicznej dokumentacji API umożliwiającej integrację z zewnętrznymi systemami. W praktyce uniemożliwia to budowanie złożonych przepływów pracy łączących oprogramowanie geologiczne z innymi narzędziami przedsiębiorstwa.

Trzecia luka dotyczy rozszerzalności mechanizmów eksportu. Istniejące systemy oferują zamknięty, sztywny zestaw formatów wyjściowych – bez możliwości definiowania własnych zestawów pól, własnych mapowań kolumn czy własnych formatów wymiany. W kontekście rosnących wymagań projektów infrastrukturalnych, gdzie różni zlecniodawcy oczekują różnych struktur danych, brak rozszerzalności staje się istotnym ograniczeniem operacyjnym.

Czwarta luka dotyczy automatyzacji przydziału pracy weryfikacyjnej. Mechanizmy klasy „każdy z każdym” (ang. *round-robin*) służące do automatycznego rozdzielania zadań między dokumentatorów na podstawie ich obciążenia w danym okresie. Takie procesy są standardem w oprogramowaniu zarządzania pracami terenowymi, lecz nie zostały zaadaptowane w żadnym z produktów geotechnicznych dostępnych na polskim rynku. W praktyce oznacza to ręczne przydzielanie zleceń przez kierownika prac, co stanowi ograniczenie organizacyjne w biurach prowadzących wiele równoległych projektów.

Piąta luka dotyczy integracji mapowych zleceń ze statusami otworów w pełnym przepływie pracy. GeoAid wprowadził w 2024 roku mapę postępu prowadzonych badań, lecz funkcja ta nie obejmuje pełnego cyklu życia zlecenia od jego utworzenia, przez przydzielenie geologowi, dokumentację terenową, weryfikację dokumentatora, aż po dostarczenie dokumentacji końcowej. Pozostałe systemy w ogóle nie udostępniają mapowego widoku zleceń ze statusami.

Projektowany system KRET adresuje wszystkie pięć zidentyfikowanych luk, jednocześnie świadomie unikając konkurencji w obszarach, w których polski rynek dysponuje już dojrzałymi rozwiązaniami. Kluczowe pozycjonowanie produktu należy bowiem rozumieć następująco: KRET nie zastępuje pakietów takich jak GeoStar czy Geosoft GEO DB w zakresie generowania kart otworu zgodnych z polskimi normami klasyfikacyjnymi. Jest to obszar dojrzały, w którym polskie biura geologiczne mają wieloletnie doświadczenie i sprawdzone rozwiązania.

System KRET wnosi warstwę dotychczas nieobecną w polskiej ofercie: operacyjne zarządzanie pracami terenowymi i dokumentacją obejmujące cały cykl życia zlecenia, mobilne wprowadzanie

danych z poziomu telefonu lub tabletu w trybie *offline*, automatyczny przydział pracy oraz otwarty interfejs programistyczny umożliwiający integrację z innymi narzędziami biurowymi. Strategicznym kierunkiem rozwoju systemu pozostaje integracja eksportowa z dominującymi narzędziami branżowymi oraz wsparcie standardu AGS dla projektów infrastrukturalnych dużej skali, co pozwala pozycjonować KRET jako uzupełnienie istniejącego ekosystemu, a nie jego substytut.

2.11. Podsumowanie

W niniejszym rozdziale przedstawiono specyfikę prac geologii inżynierskiej w Polsce, scharakteryzowano dziewięć istniejących rozwiązań informatycznych wspierających proces dokumentowania badań podłoża gruntowego oraz porównano je z projektowanym systemem KRET. Analiza pokazała, że polski rynek dysponuje dojrzałymi narzędziami w zakresie generowania dokumentacji zgodnej z normami klasyfikacyjnymi, lecz pozostaje niedoinwestowany w obszarze operacyjnego zarządzania pracami terenowymi. System KRET pozycjonuje się świadomie w tej luce rynkowej, traktując dokumentację zgodną z normami jako obszar wymagający współpracy z istniejącymi narzędziami branżowymi, a nie konkurencji z nimi. Zidentyfikowane luki oraz wynikająca z nich strategia produktu stanowią punkt wyjścia dla sformułowania wymagań funkcjonalnych i niefunkcjonalnych systemu, przedstawionych w kolejnym rozdziale.

3. Propozycja systemu KRET

W niniejszym rozdziale przedstawiono autorską propozycję systemu informatycznego KRET. Omówiono jego cele i założenia, słownik pojęć oraz użytkowników, a następnie zebrano wymagania funkcjonalne i нефункционалне. Koncepcję systemu zobrazowano za pomocą diagramów przypadków użycia, diagramu klas, diagramów stanów oraz diagramów aktywności. Zgromadzona w ten sposób specyfikacja stanowiła podstawę projektu i ułatwiła zrozumienie problemów oraz wyzwań napotkanych podczas implementacji aplikacji internetowej i mobilnej.

3.1. Cele i założenia

Celem niniejszej pracy inżynierskiej jest zaprojektowanie oraz implementacja aplikacji biznesowej wspierającej branżę geologiczną w Polsce. System KRET ma usprawnić koordynację prac dla geologów terenowych i dokumentatorów realizujących wspólne projekty badawcze oraz uporządkować obieg danych pozyskiwanych w terenie. Przyjęto, że oprogramowanie powinno zapewniać:

- gromadzenie i sprawdzanie danych wprowadzanych w terenie oraz ich weryfikację przez przełożonego dokumentatora, co podnosi jakość wyników projektu i sprzyja wymianie wiedzy w zespole;
- eksport wyników prac do formatów wymieniających z zewnętrznym oprogramowaniem branżowym. Od plików .csv, po gotowe do druku karty otworów w formacie pdf;
- pracę w dwóch trybach: biurowym w aplikacji internetowej oraz terenowym w aplikacji mobilnej;

Ze względu na zróżnicowany charakter pracy geologów inżynierskich zostało założone, że KRET nie ogranicza się do aplikacji internetowej. Aplikacja mobilna pozwala uzupełniać dane projektu niezależnie od miejsca prowadzenia prac, a praktyczne granice wyznacza jedynie obszar realizowanego projektu. Założono również możliwość pracy w warunkach ograniczonego zasięgu sieci komórkowej, w trybie *offline*, który szczegółowo opisano w podrozdziale 3.5.

3.2. Słownik pojęć

W niniejszym podrozdziale zebrano pojęcia dziedzinowe używane w dalszej części pracy. Definicje porządkują terminologię geologiczno-inżynierską oraz nazewnictwo przyjęte w systemie KRET.

- **Badanie** – czynność rozpoznawcza wykonywana w obrębie otworu. W systemie wyróżniono dwa rodzaje badania: wiercenie i sondowanie;
- **Baza wiedzy** – moduł aplikacji służący do gromadzenia informacji, poradników, dobrych praktyk i pomocniczej literatury dla pracowników firmy;
- **Dyskusja** – komunikacja między dokumentatorem a geologiem terenowym, prowadzona w panelu wiercenia lub sondowania. Służy zgłaszaniu uwag i poprawek do konkretnego pola wiercenia lub sondowania;
- **Karta otworu** – końcowy dokument wynikowy, stanowiący graficzne podsumowanie wiercenia. Zawiera właściwości otworu (współrzędne, dane administracyjne) oraz warstwy litologiczne wykonanego w nim wiercenia;

- **Otwór** – punktowe rozpoznanie głębokościowe o danych współrzędnych. Podstawowa jednostka badawcza w terenie, do której przypisuje się dane administracyjne oraz badania. Otwór może posiadać do 2 badań: wiercenie i/lub sondowanie;
- **Pole niestandardowe** – dodatkowe pole formularza badania, które uprawniony użytkownik (administrator lub dokumentator) może utworzyć i przypisać do firmy lub konkretnego projektu. Umożliwia dostosowanie aplikacji do wymagań danej firmy geologicznej, na przykład poprzez dodanie specyficznego parametru gruntu nieuwzględnionego standardowo w systemie;
- **Pomiar** – liczba uderzeń sondy (N) na odcinku penetracji, która stanowi podstawową jednostkę sondowania dynamicznego. Na podstawie tego wyniku można wyznaczyć stan zagęszczenia warstwy litologicznej;
- **Profil litologiczny** – opisowe lub graficzne przedstawienie układu warstw litologicznych w danym wierceniu;
- **Projekt** – zbiór zaplanowanych prac terenowych i analitycznych dla konkretnej inwestycji (np. budowy drogi);
- **Sondowanie** – rodzaj badania sprawdzającego zagęszczenie gruntu. System wspiera sondowania dynamiczne, w których mierzy się liczbę uderzeń N na odcinek penetracji (10 cm dla sond DPL, DPM i DPH. 20 cm dla sondy DPSH);
- **Warstwa litologiczna** – wydzielenie grupy skał lub gruntów tego samego rodzaju. Oprócz nazwy litologicznej system odnotowuje między innymi wilgotność, plastyczność, miąższość oraz obecność zwierciadła wody;
- **Wiercenie (odwiert)** – rodzaj badania, którego wynikiem jest profil litologiczny otworu;
- **Wzorzec formularza** – szablon formularza badania zawierający wybrany zestaw pól. Podczas prowadzenia badań przyspiesza wprowadzanie powtarzalnych danych przez geologów terenowych;
- **Zdjęcie 360** – zestaw co najmniej czterech fotografii dokumentujących wiercenie i próby gruntów.

3.3. Użytkownicy systemu

System KRET przeznaczony jest dla czterech typów użytkowników, różniących się zakresem uprawnień. Uprawnienia mają charakter przyrostowy. Każda kolejna rola dziedziczy możliwości ról o węższym zakresie. Uporządkowane od najmniejszego do największego zakresu role przedstawiają się następująco:

- **Gość** – użytkownik niezarejestrowany, przeglądający stronę internetową aplikacji. Może zapoznać się z ofertą systemu oraz zarejestrować konto, o ile posiada aktywne zaproszenie;
- **Geolog terenowy** – pracownik wykonujący badania i odwierty bezpośrednio w terenie, wprowadzający pierwotne dane. Pracuje przede wszystkim w aplikacji mobilnej;
- **Geolog dokumentator** – pracownik analizujący dane wprowadzone przez geologów terenowych. Tworzy nowe projekty, przydziela zadania geologom terenowym, weryfikuje poprawność wprowadzonych danych oraz generuje raporty końcowe;
- **Administrator** – użytkownik o najszerszych uprawnieniach w obrębie danej firmy. Zarządza kontami przypisanymi do firmy (dodaje je i usuwa). Odpowiada za konfigurację systemu oraz posiada wszystkie uprawnienia pozostałych ról.

3.4. Wymagania funkcjonalne

System KRET musi realizować szereg zadań, aby spełnić wymagania biznesowe oraz prawidłowo wspierać procesy geologiczno-inżynierskie. Wymagania funkcjonalne określają, w jaki sposób użytkownicy i oprogramowanie współpracują ze sobą.

Ze względu na rozbudowany charakter aplikacji wymagania funkcjonalne podzielono na logiczne moduły odpowiadające poszczególnym obszarom systemu. Podział ten ułatwia poruszanie się po specyfikacji oraz wskazuje, jakie operacje przysługują każdej z ról: administratorowi, dokumentatorowi i geologowi terenowemu. Każda tabela w niniejszym podrozdziale zawiera nazwę akcji, jej krótki opis, role uprawnione do jej wykonania oraz warunki początkowe, które muszą zostać spełnione przed jej rozpoczęciem.

Moduł panelu użytkownika stanowi punkt wejścia do systemu, udostępniając raporty, statystyki, listę projektów oraz powiadomienia. Wymagania funkcjonalne tego modułu przedstawia Tabela 4.

Tabela 4. Wymagania funkcjonalne modułu panelu użytkownika. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Zobacz pełne raporty	Wyświetla raporty i statystyki	Administrator, Dokumentator	–
2	Zmień okres raportów	Wyświetla raporty i statystyki z danego okresu	Administrator, Dokumentator	–
3	Stwórz nowy projekt	Pozwala na utworzenie nowego projektu	Administrator, Dokumentator	–
4	Filtruj projekty	Wyświetl projekty, które spełniają wybrane warunki	Administrator, Dokumentator	–
5	Wyświetl projekt	Wyświetl wybrany projekt	Administrator, Dokumentator	Projekt istnieje
6	Wyświetl powiadomienia	Wyświetl powiadomienia powiązane z użytkownikiem	Administrator, Dokumentator, Geolog terenowy	Istnieje co najmniej jedno powiadomienie
7	Usuń powiadomienie	Usuń wybrane powiadomienie	Administrator, Dokumentator, Geolog terenowy	Istnieje co najmniej jedno powiadomienie

Moduł panelu projektu skupia operacje na pojedynczym projekcie: jego konfigurację, zarządzanie polami i wzorcami w jego obrębie, przypisywanie zasobów oraz pracę z otworami. Wymagania funkcjonalne tego modułu przedstawia Tabela 5.

Tabela 5. Wymagania funkcjonalne modułu panelu projektu. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Edytuj projekt	Edytuj właściwości projektu	Administrator, Dokumentator	–
2	Eksportuj projekt	Eksportuj dane dotyczące projektu	Administrator, Dokumentator	–
3	Wyświetl pola niestandardowe	Wyświetl pola niestandardowe wykorzystywane w danym projekcie	Administrator, Dokumentator	–
4	Dodaj pole niestandardowe	Dodaj pole niestandardowe projektu	Administrator, Dokumentator	–
5	Edytuj pole niestandardowe	Edytuj pole niestandardowe projektu	Administrator, Dokumentator	Istnieje pole niestandardowe
6	Usuń pole niestandardowe	Usuń pole niestandardowe projektu	Administrator, Dokumentator	Istnieje pole niestandardowe
7	Włącz pole firmowe	Włącz pole firmowe do całego projektu	Administrator, Dokumentator	Istnieje pole firmowe
8	Wyłącz pole firmowe	Wyłącz pole firmowe projektu	Administrator, Dokumentator	Istnieje pole firmowe
9	Wyświetl wzorce formularzy	Wyświetl listę wszystkich wzorców formularzy, które można wykorzystywać w danym projekcie	Administrator, Dokumentator	–
10	Dodaj wzorec formularzy	Dodaj wzorec formularzy do badania	Administrator, Dokumentator	–
11	Usuń wzorec formularzy	Usuń wzorec formularzy	Administrator, Dokumentator	Istnieje wzorec formularzy
12	Edytuj wzorec formularzy	Edytuj wzorec formularzy	Administrator, Dokumentator	Istnieje wzorec formularzy

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
13	Przypisz geologów	Przypisz geologów do projektu	Administrator, Dokumentator	Firma posiada geologów
14	Zarezerwuj sprzęt	Zarezerwuj sprzęt do danego projektu na określony czas	Administrator, Dokumentator	–
15	Importuj dane	Importuj plik csv do projektu. Zawartością mogą być otwory, wiercenia czy też sondowania	Administrator, Dokumentator	–
16	Dodaj nowy otwór	Stwórz nowy otwór w projekcie	Administrator, Dokumentator	–
17	Edytuj otwór	Edytuj wybrany otwór	Administrator, Dokumentator	Otwór istnieje
18	Usuń otwór	Usuń wybrany otwór	Administrator, Dokumentator	Otwór istnieje
19	Wyświetl otwór	Wyświetl wybrany otwór	Administrator, Dokumentator	Otwór istnieje
20	Drukuj karty otworów	Drukuj wybrane wiercenie	Administrator, Dokumentator	Otwór istnieje

Moduł panelu otworu obejmuje przypisywanie pracowników do otworu oraz tworzenie i modyfikację przypisanych do niego badań (wierceń i sondowań). Wymagania funkcjonalne tego modułu przedstawia Tabela 6.

Tabela 6. Wymagania funkcjonalne modułu panelu otworu. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Edytuj otwór	Edytuj wybrany otwór	Administrator, Dokumentator	Otwór istnieje
2	Przypisz geologów	Przypisz geologów do otworu	Administrator, Dokumentator	Firma musi posiadać innych geologów
3	Przypisz dokumentatora	Przypisz dokumentatora do sprawdzania otworu	Administrator	Otwór istnieje
4	Importuj odwiert z pliku	Importuj plik csv w celu stworzenia wiercenia	Administrator, Dokumentator	Otwór istnieje

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
5	Dodaj odwiert	Dodaj odwiert do otworu	Administrator, Dokumentator	Otwór istnieje
6	Edytuj odwiert	Edytuj odwiert w otworze	Administrator, Dokumentator	Odwiert istnieje
7	Usuń odwiert	Usuń odwiert z otworu	Administrator, Dokumentator	Odwiert istnieje
8	Importuj sondowanie z pliku	Importuj plik csv w celu stworzenia sondowania	Administrator, Dokumentator	Otwór istnieje
9	Dodaj sondowanie	Dodaj sondowanie do otworu	Administrator, Dokumentator	Otwór istnieje
10	Edytuj sondowanie	Edytuj sondowanie w otworze	Administrator, Dokumentator	Sondowanie istnieje
11	Usuń sondowanie	Usuń sondowanie z otworu	Administrator, Dokumentator	Sondowanie istnieje

Moduł panelu odwiertu pozwala wypełniać profil litologiczny wiercenia, zarządzać warstwami oraz dokumentacją fotograficzną, a także prowadzić dyskusje na temat zawartości formularza. Wymagania funkcjonalne tego modułu przedstawia Tabela 7.

Tabela 7. Wymagania funkcjonalne modułu panelu odwiertu. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Wyświetl dyskusje	Wyświetl dyskusje przypisane do tego odwiertu	Administrator, Dokumentator, Geolog terenowy	Odwiert istnieje
2	Dodaj dyskusje	Dodaj dyskusje do określonego pola w wierceniu	Administrator, Dokumentator, Geolog terenowy	Odwiert istnieje
3	Weź udział w dyskusji	Odpowiedz na komentarz	Administrator, Dokumentator, Geolog terenowy	Dyskusja istnieje
4	Resetuj odwiert	Przywróć odwiert do stanu pierwotnego	Administrator, Dokumentator, Geolog terenowy	Odwiert istnieje

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
5	Wyświetl wzorce formularzy	Wyświetl wszystkie wzorce formularzy przypisanego do firmy	Administrator, Dokumentator	Wzorce formularzy muszą istnieć
6	Wypełnij dane w formularzu	Wypełnij wszystkie obowiązkowe dane do odwiertu	Administrator, Dokumentator, Geolog terenowy	Odwiert istnieje
7	Dodaj warstwę litologiczną	Dodaj warstwę litologiczną do odwiertu	Administrator, Dokumentator, Geolog terenowy	Odwiert istnieje
8	Edytuj warstwę litologiczną	Edytuj warstwę litologiczną w wierceniu	Administrator, Dokumentator, Geolog terenowy	Warstwa litologiczna istnieje
9	Usuń warstwę litologiczną	Usuń wybraną warstwę litologiczną z wiercenia	Administrator, Dokumentator, Geolog terenowy	Warstwa litologiczna istnieje
10	Dodaj zwykłe zdjęcie	Dodaj zdjęcie do konkretnej warstwy litologicznej	Administrator, Dokumentator, Geolog terenowy	Warstwa litologiczna istnieje
11	Dodaj zdjęcia 360	Dodaj zestaw co najmniej 4 zdjęć do warstwy litologicznej	Administrator, Dokumentator, Geolog terenowy	Warstwa litologiczna istnieje

Moduł panelu sondowania pozwala wprowadzać pomiary sondowania dynamicznego oraz prowadzić dyskusje. Wymagania funkcjonalne tego modułu przedstawia Tabela 8.

Tabela 8. Wymagania funkcjonalne modułu panelu sondowania. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Wyświetl dyskusje	Wyświetl dyskusje przypisane do tego sondowania	Administrator, Dokumentator, Geolog terenowy	Sondowanie istnieje
2	Dodaj dyskusje	Dodaj dyskusje do określonego pola sondowania	Administrator, Dokumentator, Geolog terenowy	Sondowanie istnieje
3	Weź udział w dyskusji	Odpowiedz na komentarz	Administrator, Dokumentator, Geolog terenowy	Dyskusja istnieje

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
4	Resetuj sondowanie	Przywróć sondowanie do stanu pierwotnego	Administrator, Dokumentator, Geolog terenowy	Sondowanie istnieje
5	Dodaj pomiar	Dodaj pomiar do sondowania	Administrator, Dokumentator, Geolog terenowy	Sondowanie istnieje
6	Edytuj pomiar	Edytuj pomiar w sondowaniu	Administrator, Dokumentator, Geolog terenowy	Sondowanie istnieje
7	Usuń pomiar	Usuń pomiar z sondowania	Administrator, Dokumentator, Geolog terenowy	Sondowanie istnieje
8	Wypełnij dane w formularzu	Wypełnij wszystkie obowiązkowe dane do sondowania	Administrator, Dokumentator, Geolog terenowy	Sondowanie istnieje

Moduł pól niestandardowych umożliwia uprawnionym użytkownikom definiowanie dodatkowych pól formularzy badań, dostosowanych do specyfiki firmy. Wymagania funkcjonalne tego modułu przedstawia Tabela 9.

Tabela 9. Wymagania funkcjonalne modułu pól niestandardowych. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Dodaj pole	Dodaj pole niestandardowe firmy	Administrator, Dokumentator	–
2	Edytuj pole	Edytuj pole niestandardowe firmy	Administrator, Dokumentator	Pole istnieje
3	Usuń pole	Usuń pole niestandardowe firmy	Administrator, Dokumentator	Pole istnieje
4	Wyświetl listę pól	Wyświetl listę pól niestandardowych firmy	Administrator, Dokumentator	Pole istnieje

Moduł wzorców formularzy pozwala definiować szablony formularzy badań, przyspieszające wprowadzanie powtarzalnych danych przez geologów terenowych. Wymagania funkcjonalne tego modułu przedstawia Tabela 10.

Tabela 10. Wymagania funkcjonalne modułu wzorców formularzy. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Dodaj wzorzec	Dodaj wzorzec formularza firmy	Administrator, Dokumentator	–
2	Edytuj wzorzec	Edytuj wzorzec formularza firmy	Administrator, Dokumentator	Wzorzec istnieje
3	Usuń wzorzec	Usuń wzorzec formularza firmy	Administrator, Dokumentator	Wzorzec istnieje
4	Wyświetl listę wzorców	Wyświetl listy dostępnych wzorców formularzy	Administrator, Dokumentator	Istnieje co najmniej jeden wzorzec
5	Wybierz rodzaj formularza	Użyj filtra do wybrania, które rodzaje formularzy mają się wyświetlić	Administrator, Dokumentator	Istnieje co najmniej jeden wzorzec

Moduł sprzętu obsługuje ewidencję sprzętu firmy oraz jego rezerwacje na potrzeby projektów. Wymagania funkcjonalne tego modułu przedstawia Tabela 11.

Tabela 11. Wymagania funkcjonalne modułu sprzętu. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Wyświetl wszystkie sprzęty	Wyświetl wszystkie sprzęty w firmie, w której pracuje użytkownik	Administrator, Dokumentator	Istnieje co najmniej jeden sprzęt
2	Sprawdź historie statusu	Sprawdź historie wybranego sprzętu	Administrator, Dokumentator	Sprzęt musi być wybrany
3	Edytuj sprzęt	Edytuj wybrany sprzęt z firmy	Administrator, Dokumentator	Sprzęt musi być wybrany
4	Usuń sprzęt	Usuń wybrany sprzęt z firmy	Administrator, Dokumentator	Sprzęt musi być wybrany
5	Dodaj sprzęt	Dodaj nowy sprzęt do firmy	Administrator, Dokumentator	–
6	Wyświetl rezerwacje użytkownika	Wyświetl wyłącznie rezerwacje użytkownika	Administrator, Dokumentator	–

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
7	Wyświetl wszystkie rezerwacje	Wyświetl wszystkie rezerwacje sprzętu w firmie	Administrator, Dokumentator	–
8	Zarezerwuj sprzęt	Zrób rezerwację na dany sprzęt	Administrator, Dokumentator	–

Moduł bazy wiedzy gromadzi artykuły, poradniki i dobre praktyki, porządkowane za pomocą kategorii oraz tagów. Wymagania funkcjonalne tego modułu przedstawia Tabela 12.

Tabela 12. Wymagania funkcjonalne modułu bazy wiedzy. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Wyświetl kategorie	Wyświetl wszystkie rodzaje kategorii w bazie wiedzy	Administrator, Dokumentator, Geolog terenowy	–
2	Dodaj kategorie	Dodaj nową kategorię firmową do bazy wiedzy	Administrator, Dokumentator	–
3	Edytuj kategorie	Edytuj istniejącą kategorię firmową w bazie danych	Administrator, Dokumentator	Kategoria istnieje
4	Usuń kategorie	Usuń kategorię firmową z bazy wiedzy	Administrator, Dokumentator	Kategoria istnieje
5	Wyświetl tagi	Wyświetl wszystkie rodzaje tagów	Administrator, Dokumentator, Geolog terenowy	–
6	Dodaj tag	Dodaj nowy tag do bazy wiedzy	Administrator, Dokumentator	–
7	Edytuj tag	Edytuj istniejący tag w bazie wiedzy	Administrator, Dokumentator	Tag istnieje
8	Usuń tag	Usuń tag z bazy wiedzy	Administrator, Dokumentator	Tag istnieje
9	Dodaj artykuł	Dodaj nowy artykuł do bazy wiedzy	Administrator, Dokumentator	–

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
10	Wyświetl artykuły bazy wiedzy	Wyświetl wszystkie artykuły bazy wiedzy	Administrator, Dokumentator, Geolog terenowy	Istnieje co najmniej jeden artykuł
11	Filtruj artykuł w bazie wiedzy	Wyświetl artykuł zależnie od kategorii, rodzaju	Administrator, Dokumentator, Geolog terenowy	Istnieje co najmniej jeden artykuł

Moduł zaproszeń obsługuje zapraszanie nowych pracowników do aplikacji oraz zarządzanie wysłanymi zaproszeniami. Wymagania funkcjonalne tego modułu przedstawia Tabela 13.

Tabela 13. Wymagania funkcjonalne modułu panelu zaproszeń. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1	Wyślij zaproszenia	Pozwala na zaproszenie dodatkowej osoby do aplikacji	Administrator, Dokumentator	—
2	Wyświetl zaproszenia	Pozwala na wyświetlenie zaproszeń	Administrator, Dokumentator	Istnieje co najmniej jedno zaproszenie
3	Filtruj zaproszenia	Wyświetl zaproszenia, które spełniają wybrane warunki	Administrator, Dokumentator	Istnieje co najmniej jedno zaproszenie
4	Odnów zaproszenie	Odnów konkretne wygasłe zaproszenie	Administrator, Dokumentator	Istnieje co najmniej jedno wygasłe zaproszenie

3.5. Wymagania нефункционалне

Wymagania нефункционалне definiują atrybuty jakościowe systemu KRET. Stanowią standardy, które aplikacja musi spełnić niezależnie od konkretnych funkcji biznesowych. Ich spełnienie przekłada się na aspekty szczególnie istotne dla geologów: bezpieczeństwo danych, wydajność, komfort pracy w terenie oraz przewidywalność interfejsu. Ze względu na specyfikę pracy geologa terenowego, która często odbywa się w warunkach ograniczonego zasięgu sieci, do określania wymagań przyjęto konkretne miary umożliwiające ich późniejszą weryfikację. Pełny zestaw wymagań нефункционалных systemu KRET, pogrupowanych według aspektów jakościowych, przedstawia Tabela 14.

Tabela 14. Wymagania niefunkcjonalne. Źródło: Opracowanie własne.

Lp.	Aspekt	Opis
1	Użyteczność	Aplikacja internetowa spełnia założenia projektowania responsywnego (RWD) i poprawnie wyświetla się w przeglądarkach opartych na silnikach Chromium, Gecko oraz WebKit w aktualnych wersjach.
2	Użyteczność	Interfejs aplikacji mobilnej jest dostosowany do warunków polowych. Przyciski, etykiety i formularze mają rozmiar umożliwiający obsługę jedną ręką oraz odczyt przy zmiennych warunkach oświetleniowych.
3	Użyteczność	Po każdej operacji modyfikującej dane system informuje użytkownika o jej powodzeniu lub błędzie wraz ze wskazaniem pól wymagających poprawy.
4	Użyteczność	Interfejs użytkownika dostępny jest w języku polskim.
5	Wydajność	Czas odpowiedzi serwera na typowe żądanie nie przekracza 2 sekund przy stabilnym łączu internetowym.
6	Wydajność	Maksymalny rozmiar pojedynczego przesyłanego zdjęcia wynosi 15 MB, a pojedynczego żądania 150 MB. Aplikacja mobilna stosuje kolejkę przesyłania plików, ograniczając ryzyko przeciążenia serwera przy przesyłaniu wielu plików.
7	Wydajność	Maksymalny rozmiar pliku CSV importowanego do projektu wynosi 32 MB.
8	Bezpieczeństwo	Hasła użytkowników przechowywane są w bazie danych w postaci skrótów wygenerowanych algorytmem BCrypt.
9	Bezpieczeństwo	Uwierzytelnianie oparte jest na tokenach JWT. Token dostępowy ma ważność 15 minut, token odświeżający 7 dni.
10	Bezpieczeństwo	Komunikacja pomiędzy aplikacjami klienckimi a serwerem odbywa się wyłącznie z wykorzystaniem protokołu HTTPS (TLS 1.2 lub TLS 1.3). Żądania kierowane na port HTTP są przekierowywane na HTTPS.
11	Bezpieczeństwo	System egzekwuje kontrolę dostępu opartą na rolach (administrator, dokumentator, geolog terenowy) oraz na przypisaniu do projektu. Użytkownik widzi wyłącznie zasoby swojej firmy.
12	Kompatybilność i przenośność	Dane projektu mogą być eksportowane w sześciu formatach: CSV, XLSX, JSON, GEOJSON, AGS4 oraz XML, co zapewnia interoperacyjność z zewnętrznym oprogramowaniem branżowym.
13	Kompatybilność i przenośność	Karty otworów dostępne są w postaci graficznej z konfigurowalnym układem (skala wydruku, nagłówek, stopka, profil litologiczny) oraz mogą zostać wydrukowane lub zapisane do pliku PDF za pośrednictwem przeglądarki internetowej.
14	Kompatybilność i przenośność	Aplikacja mobilna dostępna jest dla systemów Android i iOS.
15	Kompatybilność i przenośność	Specyfikacja interfejsu API udostępniona jest w standardzie OpenAPI, co umożliwia jej wykorzystanie przez systemy zewnętrzne.

Lp.	Aspekt	Opis
16	Dostępność i niezawodność	Aplikacja mobilna umożliwia pracę w trybie <i>offline</i> . Wprowadzone dane przechowywane są lokalnie i synchronizowane z serwerem po przywróceniu łącza.
17	Dostępność i niezawodność	Operacje wykonane w trybie <i>offline</i> trafiają do lokalnej kolejki synchronizacji, która podejmuje do 3 prób ponowienia w przypadku nieudanej transmisji, zapobiegając utracie pracy użytkownika.
18	Skalowalność	Serwer systemu jest bezstanowy (sesje JWT, brak stanu sesji po stronie serwera), co umożliwia skalowanie horyzontalne przez uruchomienie wielu instancji aplikacji za wspólnym serwerem pośredniczącym.
19	Skalowalność	System wspiera obsługę wielu firm jednocześnie zapewniając pełną izolację danych pomiędzy firmami na poziomie warstwy dostępu do danych.

3.6. Diagram przypadków użycia

Aby możliwie najlepiej zwizualizować interakcje pomiędzy użytkownikami systemu KRET, a oferowanymi funkcjonalnościami przygotowano cztery diagramy przypadków użycia, po jednym dla każdej z ról opisanych w podrozdziale 3.3. Każdy diagram przedstawia granice systemu oraz zakres uprawnień przysługujący danemu aktorowi. Przy tworzeniu diagramów wykorzystano notację UML, obejmującą asocjacje pomiędzy aktorem a przypadkami użycia, relacje typu <<include>> (włączenie zachowania). Aktor podrzędny dziedziczy wszystkie przypadki użycia aktora nadrzędnego, dzięki czemu diagramy aktorów o większych uprawnieniach prezentują wyłącznie nowe, dodatkowe funkcjonalności.

Zgodnie z dobrymi praktykami modelowania, w wielu przypadkach zrezygnowano z osobnego wyróżniania przypadków „dodaj”, „edytuj” i „usuń”. Operacje te połączono pod wspólną nazwą „modyfikuj”. Na diagramach zawarto wyłącznie przypadki istotne biznesowo, pomijając pomocnicze operacje czysto techniczne (na przykład odświeżanie tokenu uwierzytelniającego).

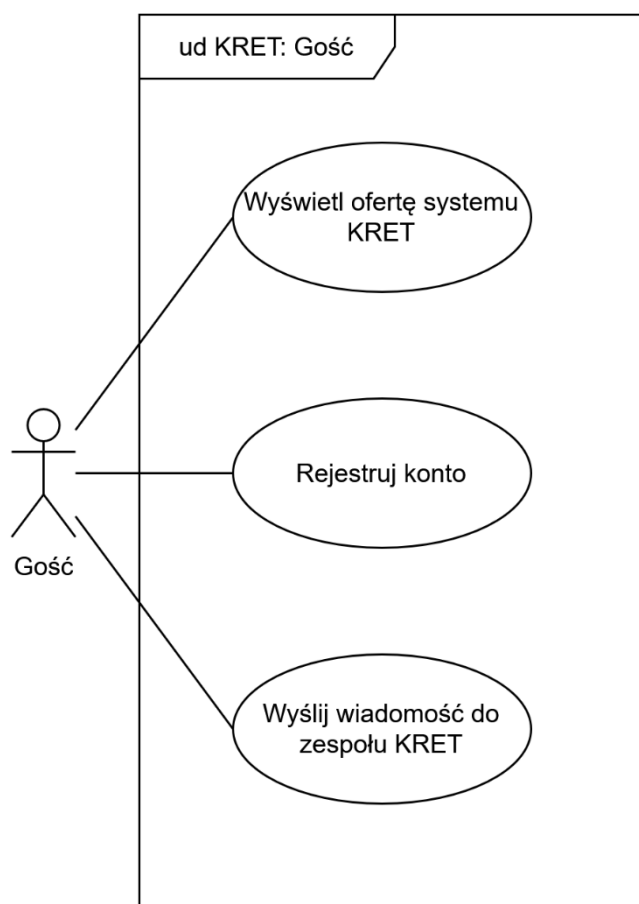
Hierarchia aktorów systemu KRET, uporządkowana od najwęższego do najszerszego zakresu uprawnień, przedstawia się następująco: Gość → Geolog terenowy → Dokumentator → Administrator. Diagramy przypadków użycia dla poszczególnych ról przedstawiają Rysunek 16, Rysunek 17, Rysunek 18, Rysunek 19.

3.6.1. Diagram przypadków użycia dla Gościa systemu

Gość to użytkownik niezarejestrowany, korzystający ze strony internetowej aplikacji KRET bez uprzedniego logowania. Zakres dostępnych dla niego funkcjonalności jest ograniczony do operacji informacyjnych oraz wstępnej obsługi procesu rejestracji.

Wszystkie operacje dostępne dla Gościa służą celom informacyjnym oraz inicjacji procesu rejestracji. Konto użytkownika może zostać założone wyłącznie na podstawie aktywnego zaproszenia otrzymanego pocztą elektroniczną od osoby posiadającej wyższe uprawnienia w systemie, co opisano szczegółowo w scenariuszu 3.7.1. Takie podejście pozwala kontrolować dostęp do platformy i zapobiega rejestracji osób spoza firmy.

Wysyłanie wiadomości do zespołu KRET realizowane jest za pośrednictwem formularza kontaktowego dostępnego na stronie głównej i pełni funkcję pomocniczą.



Rysunek 16. Diagram przypadków użycia dla Gościa. Źródło: Opracowanie własne.

3.6.2. Diagram przypadków użycia dla Geologa terenowego

Geolog terenowy to pracownik wykonujący badania bezpośrednio w terenie i wprowadzający pierwotne dane do systemu. Pracuje głównie z aplikacją mobilną, choć ma również dostęp do aplikacji internetowej. Diagram (Rysunek 17) rozszerza zakres uprawnień gościa o operacje związane z wypełnianiem badań oraz komunikacją z dokumentatorem.

Kluczową odpowiedzialnością geologa terenowego jest wprowadzanie wyników pomiarów i wierceń w obrębie przypisanych do niego otworów. Wypełnianie formularza wiercenia obejmuje (poprzez relacje <<include>>) modyfikację warstw litologicznych oraz dodawanie dokumentacji fotograficznej, w tym zdjęć panoramicznych. Resetowanie badania umożliwia rozpoczęcie pracy od nowa, gdy pierwotne dane wymagają całkowitej rewizji. Zdjęcia 360 (zestaw co najmniej czterech fotografii) służą do pełnego udokumentowania profilu litologicznego oraz prób gruntów wykonanych podczas wiercenia.

Komunikacja z dokumentatorem realizowana jest poprzez udział w dyskusjach prowadzonych na poziomie konkretnych pól formularza wiercenia lub sondowania. Pozwala to na precyzyjne odnoszenie się do zgłaszanych uwag oraz dokumentowanie historii poprawek w obrębie danego badania. Skuteczna komunikacja w obrębie dyskusji wpływa bezpośrednio na jakość końcowej dokumentacji geologicznej. Geolog terenowy może również otrzymywać powiadomienia o przypisaniu do nowego badania, prośbie o odpowiedź w dyskusji lub o zmianie statusu otworu, nad którym pracuje. Funkcjonalność eksportu wyników pracy ograniczona jest do zakresu badań przypisanych do danego geologa.

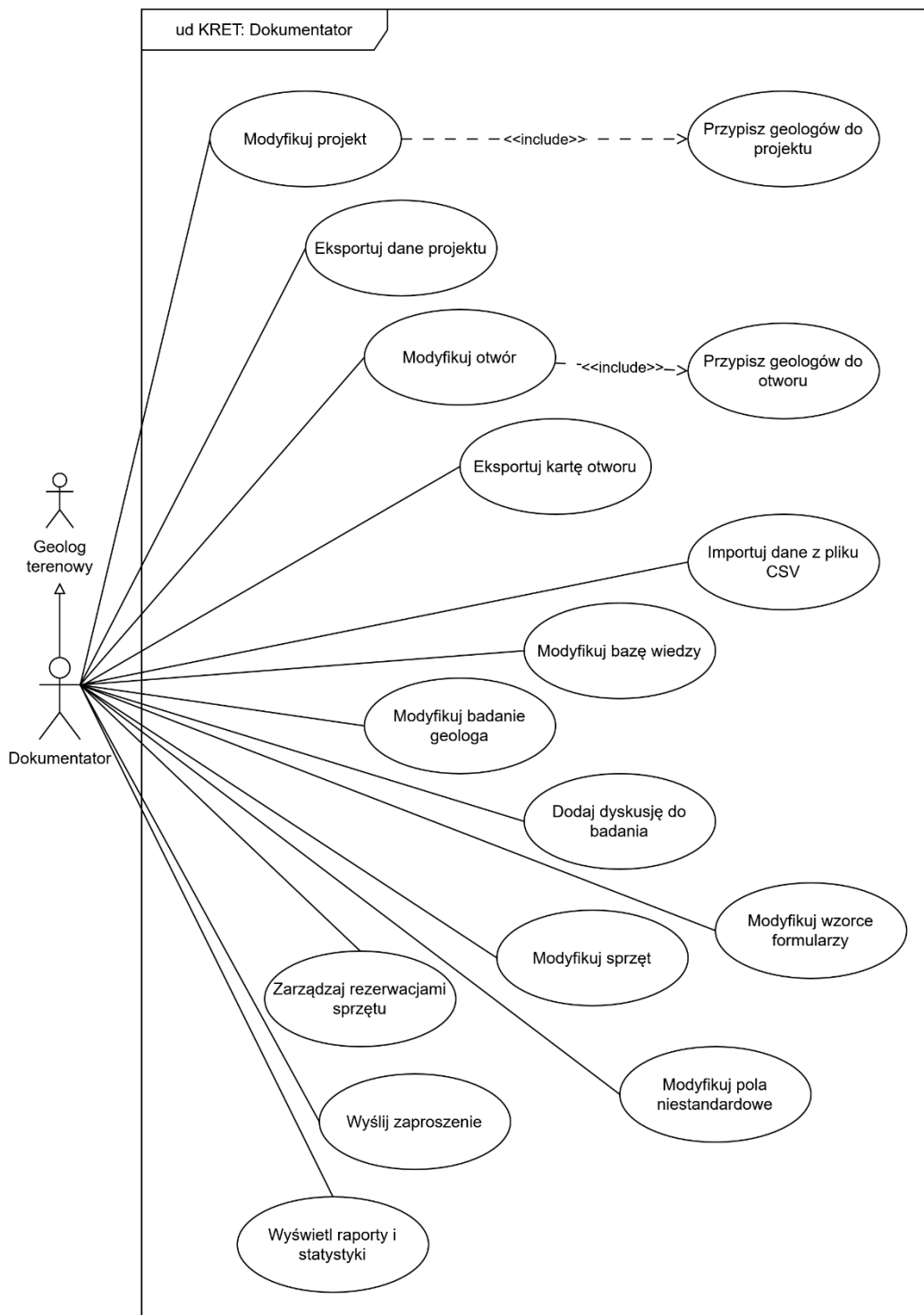
Diagram przypadków użycia dla geologa terenowego przedstawia Rysunek 17.



Rysunek 17. Diagram przypadków użycia dla geologa terenowego. Źródło: Opracowanie własne.

3.6.3. Diagram przypadków użycia dla Dokumentatora

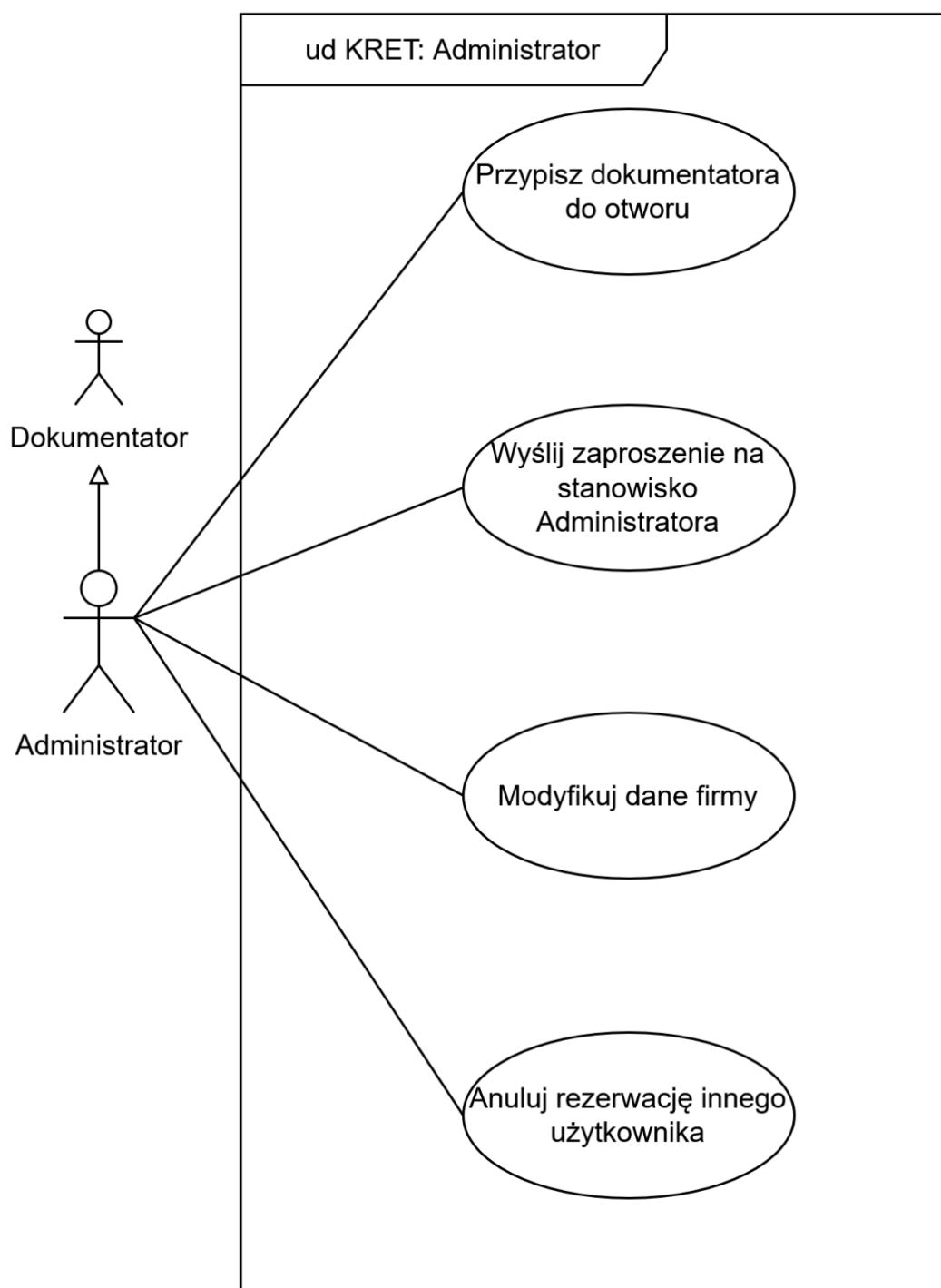
Dokumentator analizuje dane wprowadzone przez geologów terenowych, tworzy projekty oraz weryfikuje poprawność dokumentacji. Diagram (Rysunek 18) rozszerza zakres uprawnień geologa terenowego o operacje związane z zarządzaniem projektami oraz konfiguracją systemu na poziomie firmy. Diagram przypadków użycia dla Dokumentatora przedstawia Rysunek 18.



Rysunek 18. Diagram przypadków użycia dla Dokumentatora. Źródło: Opracowanie własne.

3.6.4. Diagram przypadków użycia dla Administratora

Administrator posiada najszerszy zakres uprawnień w obrębie danej firmy. Odpowiada za konfigurację systemu, zarządzanie kontami użytkowników oraz operacje wykraczające poza pojedynczy projekt. Diagram (Rysunek 19) rozszerza zakres uprawnień dokumentatora o funkcje administracyjne.



Rysunek 19. Diagram przypadków użycia dla Administratora. Źródło: Opracowanie własne.

3.7. Omówienie wybranych przypadków użycia

W niniejszym podrozdziale przedstawiono scenariusze wybranych przypadków użycia, opisujące sekwencje działań podejmowanych przez aktora w celu zrealizowania konkretnej interakcji z systemem KRET. Każdy scenariusz prezentuje główny przebieg zdarzeń wraz z możliwymi przepływami alternatywnymi obejmującymi przypadki błędów oraz wariantów postępowania. Scenariusze te stanowią uzupełnienie diagramów przedstawionych w podrozdziale 3.6 i precyzyjnie określają oczekiwane zachowanie systemu, ułatwiając proces projektowania, implementacji oraz testowania.

Do szczegółowego omówienia wybrano siedem przypadków użycia kluczowych z perspektywy realizowanego procesu biznesowego: rejestrację użytkownika, tworzenie nowego otworu, wypełnianie formularza wiercenia, import wiercenia z pliku, pracę w trybie *offline*, weryfikację badań przez dokumentatora oraz eksport karty otworu. Scenariusze tych przypadków przedstawia Tabela 15, Tabela 16, Tabela 17, Tabela 18, Tabela 19, Tabela 20 i Tabela 21.

3.7.1. Rejestracja użytkownika (Gość)

Scenariusz przedstawia proces rejestracji nowego konta w systemie KRET przez Gościa, który otrzymał wcześniej zaproszenie pocztą elektroniczną. Cel przypadku: utworzenie konta i uzyskanie dostępu do funkcji systemu. Warunkiem początkowym jest brak wcześniejszej rejestracji oraz posiadanie aktywnego zaproszenia. Scenariusz przepływu zdarzeń przedstawia Tabela 15.

Tabela 15. Scenariusz przepływu zdarzeń przypadku „Rejestracja użytkownika”. Źródło: Opracowanie własne.

Nazwa	Rejestracja użytkownika	
Aktorzy	Gość	
Główny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
1	Gość otwiera wiadomość elektroniczną zawierającą link aktywacyjny	2
2	Gość klika w link i przechodzi do formularza rejestracji	3
3	Gość wprowadza obowiązkowe dane do formularza	4
4	System sprawdza poprawność wprowadzonych danych	5
5	System tworzy nowe konto i informuje o pomyślnej rejestracji	6
6	Użytkownik loguje się do aplikacji	—
Alternatywny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
4a	System wykrywa nieprawidłowe dane w formularzu	4.1a
4.1a	System wyświetla komunikat o błędzie i wskazuje pola do poprawy	3

3.7.2. Tworzenie nowego otworu (Dokumentator/Admin)

Scenariusz opisuje proces utworzenia nowego otworu w obrębie wybranego projektu przez dokumentatora lub administratora. Cel przypadku: dodanie nowego punktu rozpoznania głębokościowego do projektu wraz z jego danymi administracyjnymi i przypisaniem do geologa terenowego. Warunkiem początkowym jest zalogowanie użytkownika oraz istnienie wybranego projektu. Scenariusz przepływu zdarzeń przedstawia Tabela 16.

Tabela 16. Scenariusz przepływu zdarzeń przypadku „Tworzenie nowego otworu”. Źródło: Opracowanie własne.

Nazwa	Tworzenie nowego otworu	
Aktorzy	Dokumentator, Administrator	
Główny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
1	Użytkownik otwiera wybrany projekt	2
2	Użytkownik otwiera moduł „Dodaj otwór"	3
3	Użytkownik wypełnia formularz otworu (nazwa, współrzędne, dane administracyjne obszaru, przypisany geolog)	4
4	System sprawdza wprowadzone dane	5
5	System zapisuje otwór w bazie danych i potwierdza utworzenie	6
6	Użytkownik może dodać badanie do nowo utworzonego otworu	–
Alternatywny przepływ zdarzeń:		
3a	Użytkownik pomija wprowadzenie danych administracyjnych	3.1a
3.1a	System uzupełnia dane administracyjne automatycznie na podstawie współrzędnych za pomocą zewnętrznego interfejsu	4
4a	System wykrywa nieprawidłowe dane w formularzu	4.1a
4.1a	System wyświetla komunikat o błędzie i wskazuje pola do poprawy	3

3.7.3. Wypełnianie formularza wiercenia (Geolog terenowy)

Scenariusz przedstawia proces wprowadzania szczegółowego profilu litologicznego do otworu przez geologa terenowego pracującego z aplikacją mobilną. Cel przypadku: utworzenie kompletnej dokumentacji wiercenia obejmującej warstwy litologiczne, parametry gruntu oraz dokumentację fotograficzną. Warunkiem początkowym jest istnienie otworu przypisanego do geologa oraz zalogowanie użytkownika. Scenariusz przepływu zdarzeń przedstawia Tabela 17.

Tabela 17. Scenariusz przepływu zdarzeń przypadku „Wypełnianie formularza wiercenia”. Źródło: Opracowanie własne.

Nazwa	Wypełnianie formularza wiercenia
Aktorzy	Geolog terenowy

Główny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
1	Geolog wybiera otwór, do którego chce dodać wiercenie	2
2	Geolog wybiera wzorzec formularza	3
3	Geolog wypełnia formularz (warstwy litologiczne, głębokości, pola niestandardowe, dokumentacja fotograficzna)	4
4	System sprawdza wprowadzone dane	5
5	System zapisuje wiercenie w bazie danych i potwierdza zapis	–
Alternatywny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
3a	Aplikacja traci połączenie z serwerem	3.1a
3.1a	Aplikacja przechodzi w tryb <i>offline</i> , zapisując dane w pamięci lokalnej urządzenia	3.2a
3.2a	Po przywróceniu łączy dane zostają automatycznie zsynchronizowane z serwerem	4
3b	Geolog dodaje zdjęcia o dużym rozmiarze	3.1b
3.1b	Aplikacja tworzy kolejkę wgrywania plików i kompresuje zdjęcia przed wysłaniem	4
4a	System wykrywa nieprawidłowe dane w formularzu	4.1a
4.1a	System wyświetla komunikat o błędzie i wskazuje pola do poprawy	3

3.7.4. Importowanie wiercenia z pliku (Dokumentator/Administrator)

Scenariusz opisuje proces zaimportowania danych wiercenia z pliku CSV przez dokumentatora lub administratora. Cel przypadku: szybkie wprowadzenie danych wiercenia do otworu bez ręcznego wypełniania formularza. Warunkiem początkowym jest istnienie otworu oraz zalogowanie użytkownika. Scenariusz przepływu zdarzeń przedstawia Tabela 18.

Tabela 18. Scenariusz przepływu zdarzeń przypadku „Importowanie wiercenia z pliku”. Źródło: Opracowanie własne.

Nazwa	Importowanie wiercenia z pliku	
Aktorzy	Dokumentator, Administrator	
Główny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej

		aktywności
1	Użytkownik wybiera otwór, do którego chce przypisać wiercenie	2
2	Użytkownik otwiera moduł „Importuj odwiert z pliku”	3
3	Użytkownik wybiera ustawienia importu (separator, kodowanie znaków)	4
4	Użytkownik wskazuje plik z dysku	5
5	System wyświetla zawartość pliku w formie tabeli, oznaczając ewentualne błędy	6
6	Użytkownik wybiera opcję „Importuj”	7
7	System zapisuje wiercenie w bazie danych	–
Alternatywny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
5a	System wykrywa uszkodzony lub nieprawidłowy plik	5.1a
5.1a	System wyświetla komunikat o błędzie pliku	4
5b	System wykrywa pojedyncze błędne rekordy w tabeli	5.1b
5.1b	Błędne warstwy litologiczne zostają pominięte, pozostałe są importowane poprawnie	7
5c	Użytkownik wybiera opcję powrotu do wyboru pliku	4

3.7.5. Praca w trybie offline

Scenariusz przedstawia ciągłość pracy geologa terenowego w miejscach o ograniczonym zasięgu sieci komórkowej. Cel przypadku: umożliwienie nieprzerwanego wprowadzania danych w terenie niezależnie od dostępności łącza. Warunkiem początkowym jest utrata zasięgu oraz aktywny tryb *offline* w aplikacji. Scenariusz przepływu zdarzeń przedstawia Tabela 19.

Tabela 19. Scenariusz przepływu zdarzeń przypadku „Praca w trybie offline”. Źródło: Opracowanie własne.

Nazwa	Praca w trybie <i>offline</i>	
Aktorzy	Geolog terenowy	
Główny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
1	Geolog pracuje lokalnie, wypełniając formularze wierceń lub sondowań oraz dodając dokumentację fotograficzną	2
2	System zapisuje wszystkie zmiany w lokalnej bazie danych urządzenia	3

3	Po przywróceniu połączenia system automatycznie synchronizuje dane z serwerem	—
Alternatywny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
3a	Synchronizacja nie powiedzie się	3.1a
3.1a	System ponawia próbę synchronizacji do trzech razy, zachowując dane lokalne	3

3.7.6. Weryfikacja i zatwierdzenie badania (Dokumentator/Administrator)

Scenariusz opisuje proces weryfikacji danych badania wprowadzonych przez geologa terenowego przez dokumentatora lub administratora. Cel przypadku: zapewnienie jakości zebranej dokumentacji geologicznej poprzez kontrolę przełożonego. Warunkiem początkowym jest złożenie badania przez geologa terenowego oraz przypisanie dokumentatora do jego weryfikacji. Scenariusz przepływu zdarzeń przedstawia Tabela 20.

Tabela 20. Scenariusz przepływu zdarzeń przypadku „Weryfikacja i zatwierdzenie badania”. Źródło: Opracowanie własne.

Nazwa	Weryfikacja i zatwierdzenie badania	
Aktorzy	Dokumentator, Administrator	
Główny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
1	Dokumentator otrzymuje powiadomienie o przypisaniu do weryfikacji badania	2
2	Dokumentator otwiera szczegóły wiercenia lub sondowania	3
3	Dokumentator dodaje dyskusję do pola formularza wymagającego uwagi	4
4	System wysyła powiadomienie do przypisanego geologa terenowego	–
Alternatywny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
3a	Dokumentator stwierdza poprawność wszystkich danych	3.1a
3.1a	Dokumentator zatwierdza badanie bez dodawania dyskusji	–

3.7.7. Eksport karty otworu

Scenariusz przedstawia proces przygotowania gotowej do druku karty wierceń w formacie PDF przez dokumentatora lub administratora. Cel przypadku: wygenerowanie końcowego dokumentu wynikowego stanowiącego graficzne podsumowanie wiercenia. Warunkiem początkowym jest istnienie zatwierdzonych otworów zawierających wiercenia w określonym projekcie. Scenariusz przepływu zdarzeń przedstawia Tabela 21.

Tabela 21. Scenariusz przepływu zdarzeń przypadku „Eksport karty otworu”. Źródło: Opracowanie własne.

Nazwa	Eksport karty otworu	
Aktorzy	Dokumentator, Administrator	
Główny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
1	Użytkownik otwiera moduł „Eksportuj karty otworów"	2
2	Użytkownik wybiera otwór do eksportu	3
3	System generuje kartę otworu na podstawie danych wiercenia oraz parametrów otworu	4
4	Użytkownik konfiguruje układ karty (skala, nagłówek, stopka, profil litologiczny)	5
5	Użytkownik pobiera plik PDF z gotową kartą wiercenia	—
Alternatywny przepływ zdarzeń:		
Lp.	Nazwa aktywności	Numer następnej aktywności
3a	W otworze brakuje części danych	3.1a
3.1a	Użytkownik uzupełnia brakujące pola bezpośrednio w widoku karty (zmiany nie są zapisywane w systemie)	4

3.8. Diagram klas

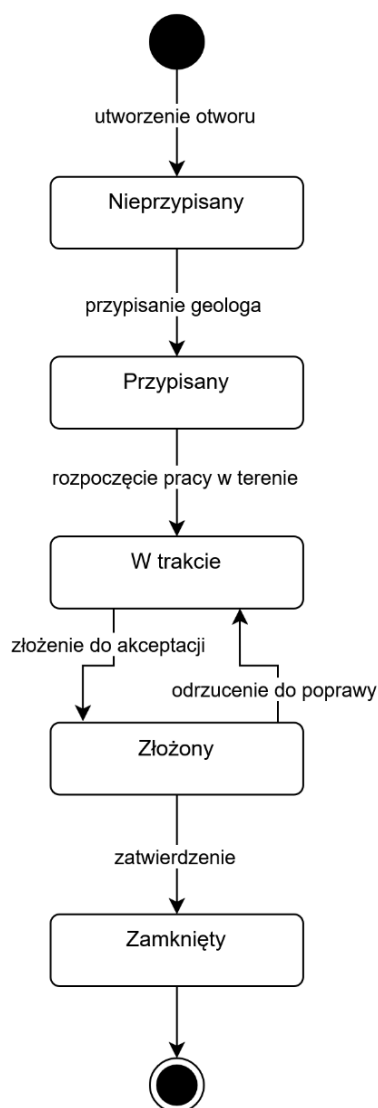
W niniejszym podrozdziale przedstawiono analityczny diagram klas systemu KRET, prezentujący strukturę domeny biznesowej oraz powiązania pomiędzy jej kluczowymi elementami. Diagram wykorzystuje klasyczną notację UML 2.x i pomija szczegóły implementacyjne (adnotacje mapowania obiektowo-relacyjnego, identyfikatory techniczne oraz pola audytu zostały świadomie usunięte, aby wyeksponować logikę biznesową aplikacji). Typy wyliczeniowe występujące w atrybutach klas (status projektu, status otworu, rola użytkownika, rodzaj litologii i pozostałe) zostały pominięte na diagramie analitycznym, ponieważ ich pełne specyfikacje przedstawiono w projektowej fazie modelowania (rozdział 5). Analityczny diagram klas systemu KRET przedstawia Rysunek 20.

3.9. Diagramy stanów

W niniejszym podrozdziale przedstawiono diagramy stanów dla wybranych klas systemu KRET o najbardziej rozbudowanym cyklu życia. Diagramy obrazują, przez jakie stany przechodzą obiekty od momentu utworzenia do zakończenia ich aktywnego życia w systemie oraz jakie zdarzenia powodują przejścia pomiędzy poszczególnymi stanami. Do szczegółowego omówienia wybrano dwie klasy: Otwór oraz Sprzęt, których stany w istotny sposób wpływają na przebieg procesu biznesowego.

3.9.1. Diagram stanów otworu

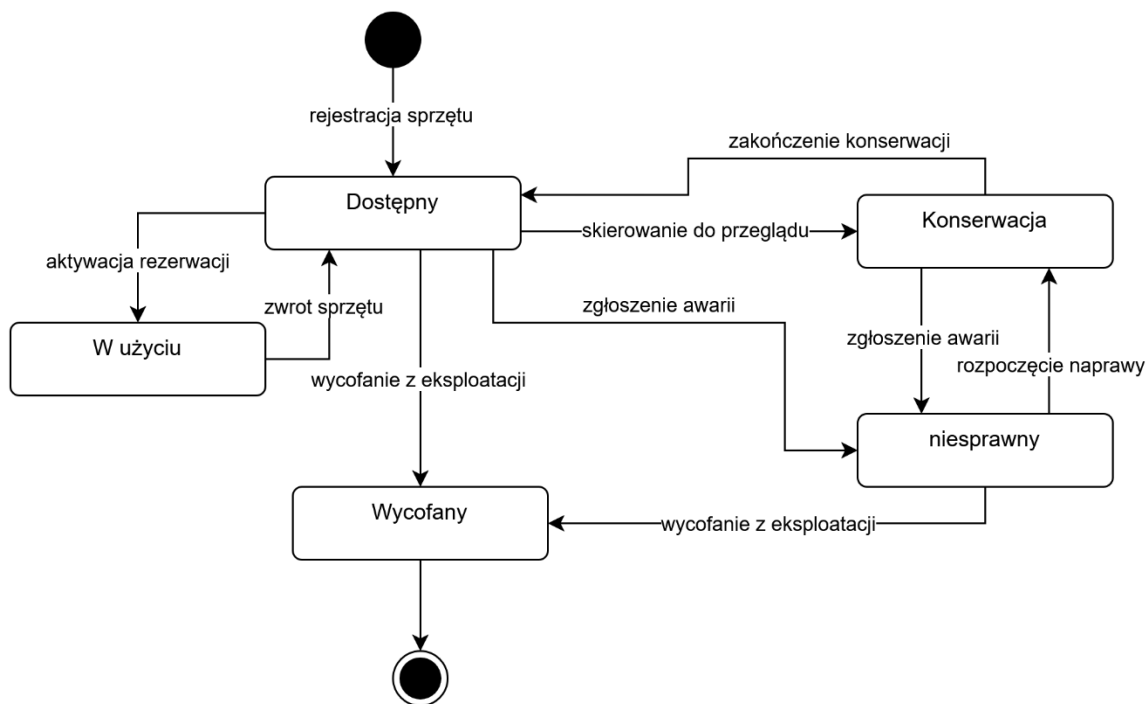
Otwór jest centralnym obiektem pracy geologa terenowego i przechodzi przez kilka stanów odzwierciedlających kolejne etapy jego dokumentowania. Od utworzenia przez dokumentatora, poprzez przypisanie geologa, wprowadzenie danych w terenie, weryfikację, aż po końcowe zamknięcie. Diagram stanów przedstawia Rysunek 21.



Rysunek 21. Diagram stanów klasy Otwór. Źródło: Opracowanie własne.

3.9.2. Diagram stanów sprzętu

Sprzęt geologiczny w systemie KRET podlega zarządzaniu statusem operacyjnym, który decyduje o jego dostępności do rezerwacji. Zmiany statusu mogą następować zarówno automatycznie (w wyniku aktywacji lub zakończenia rezerwacji), jak i manualnie (w trakcie konserwacji lub po awarii). Diagram stanów sprzętu przedstawia Rysunek 22.



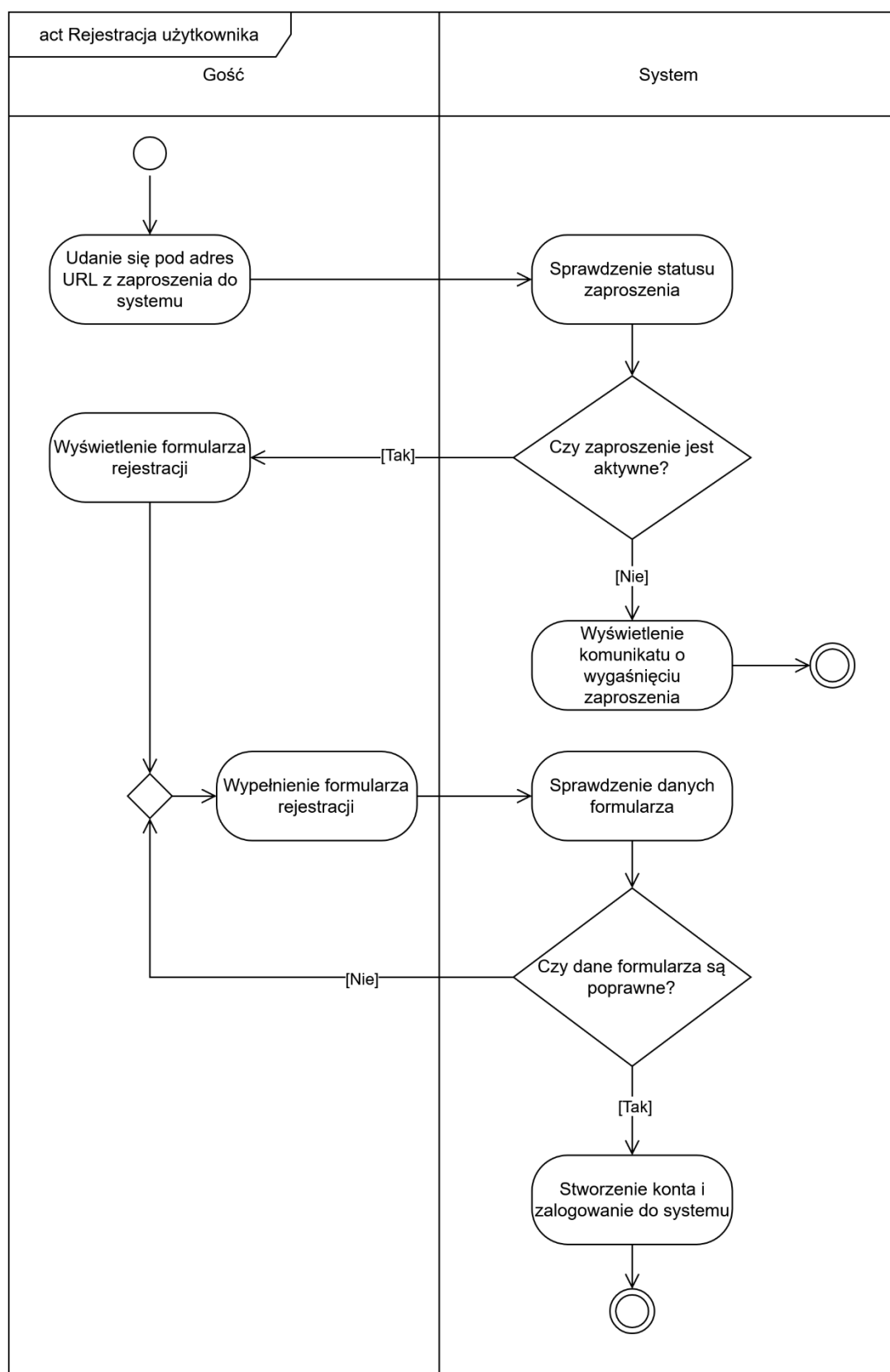
Rysunek 22. Diagram stanów klasy Sprzęt. Źródło: Opracowanie własne.

3.10. Diagramy aktywności

W niniejszym podrozdziale przedstawiono diagramy aktywności dla wybranych przypadków użycia systemu KRET, obrazujące najważniejsze procesy biznesowe oraz interakcje pomiędzy aktorami a systemem. Do szczegółowego omówienia wybrano pięć procesów najistotniejszych z perspektywy codziennej pracy użytkowników: rejestrację użytkownika, dodawanie nowego otworu, tworzenie wiercenia, weryfikację formularza wiercenia oraz eksport karty otworu.

3.10.1. Diagram aktywności dla rejestracji użytkownika

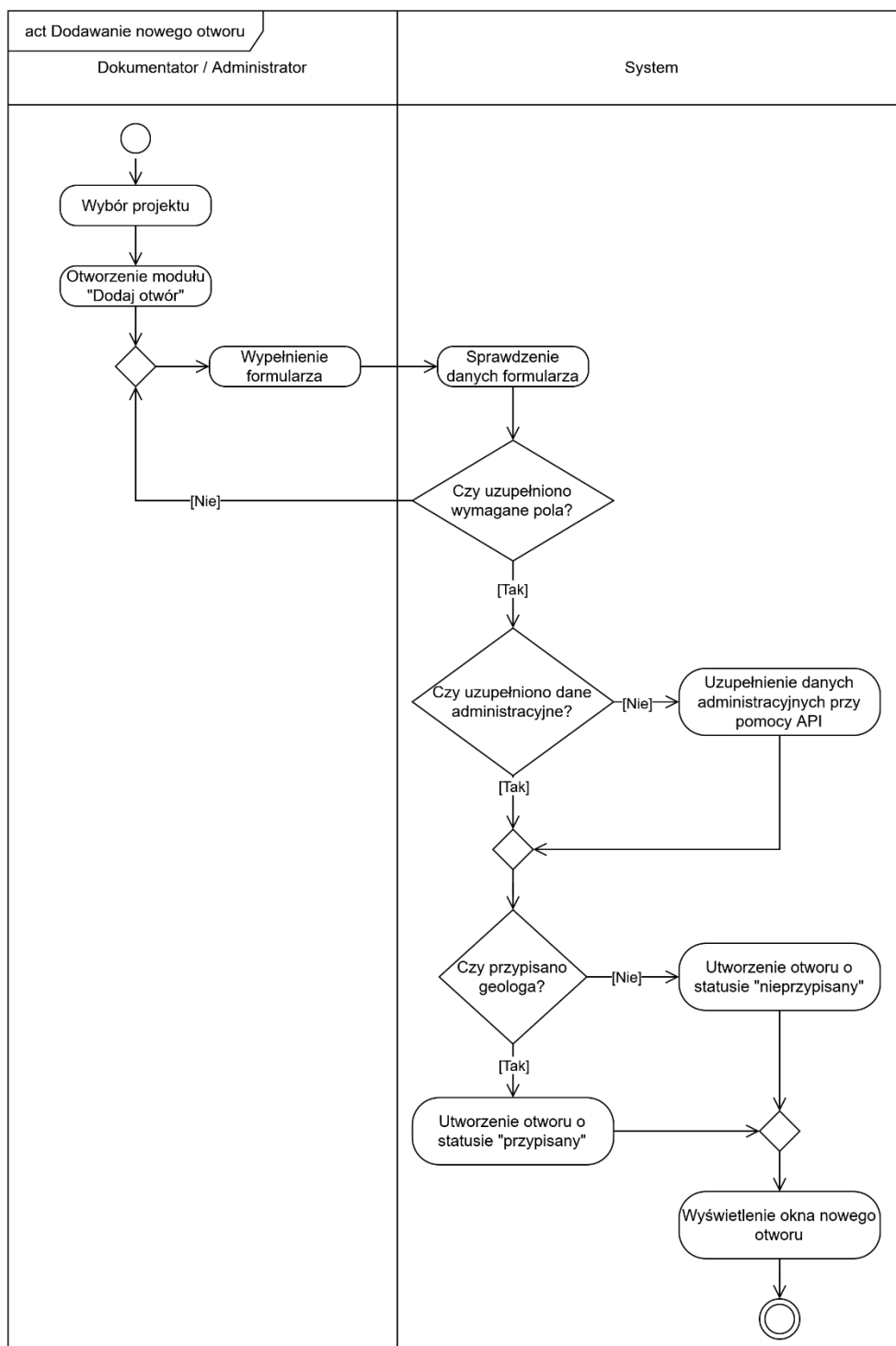
Rejestracja użytkownika w systemie KRET odbywa się wyłącznie na zaproszenie. Administrator lub dokumentator wysyła zaproszenie zawierające unikalny odnośnik, po którego otwarciu gość wypełnia formularz rejestracyjny i otrzymuje uprawnienia zgodne z rolą wskazaną w zaproszeniu. Diagram aktywności procesu rejestracji użytkownika przedstawia Rysunek 23.



Rysunek 23. Diagram aktywności – rejestracja użytkownika. Źródło: Opracowanie własne.

3.10.2. Diagram aktywności dla dodawania nowego otworu

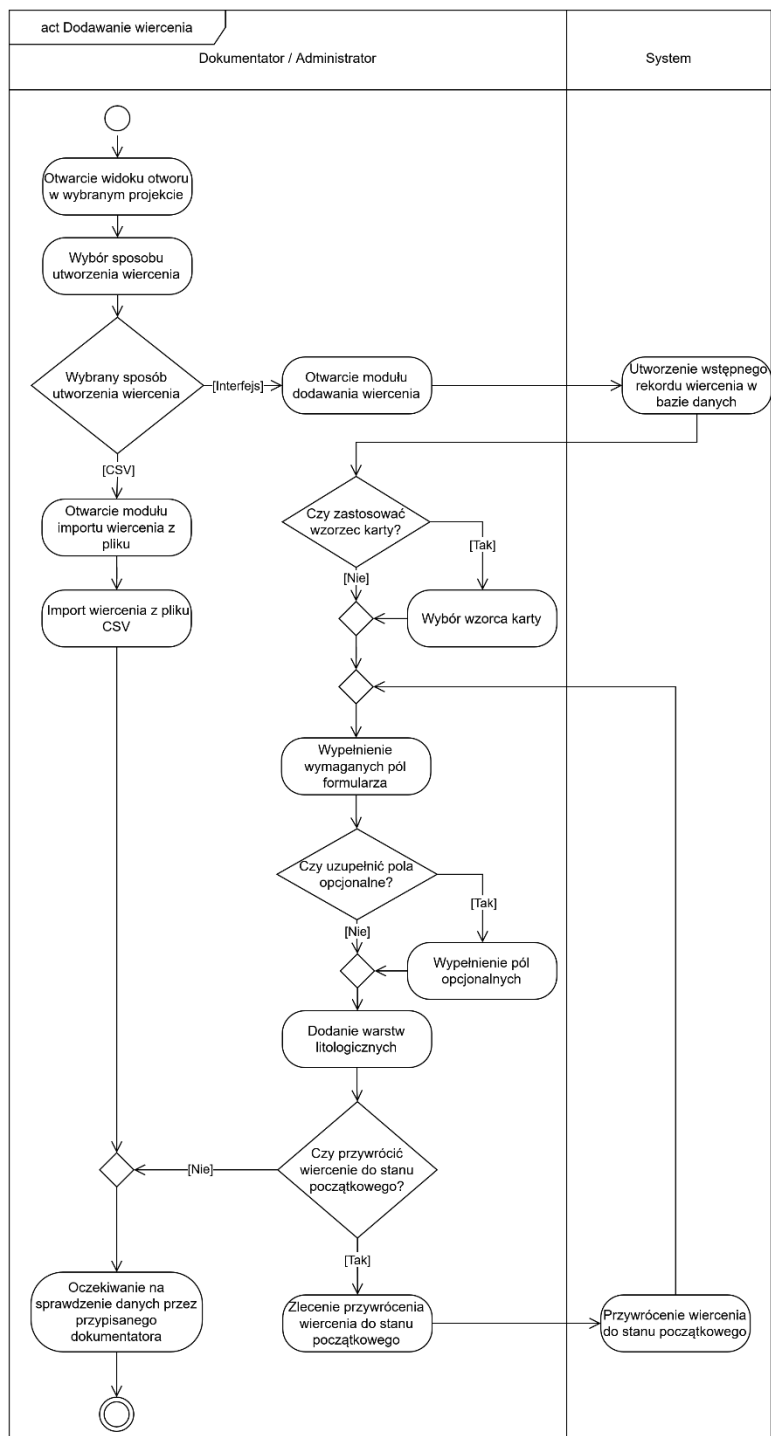
Dodawanie nowego otworu w systemie KRET inicjuje dokumentator, który definiuje lokalizację otworu na mapie oraz określa jego podstawowe atrybuty. Otwór tworzony jest w stanie „Nieprzypisany” i wymaga późniejszego przypisania geologa terenowego, który podejmie pracę w terenie. Diagram aktywności procesu dodawania nowego otworu przedstawia Rysunek 24.



Rysunek 24. Diagram aktywności – dodawanie nowego otworu. Źródło: Opracowanie własne.

3.10.3. Diagram aktywności dla tworzenia wiercenia

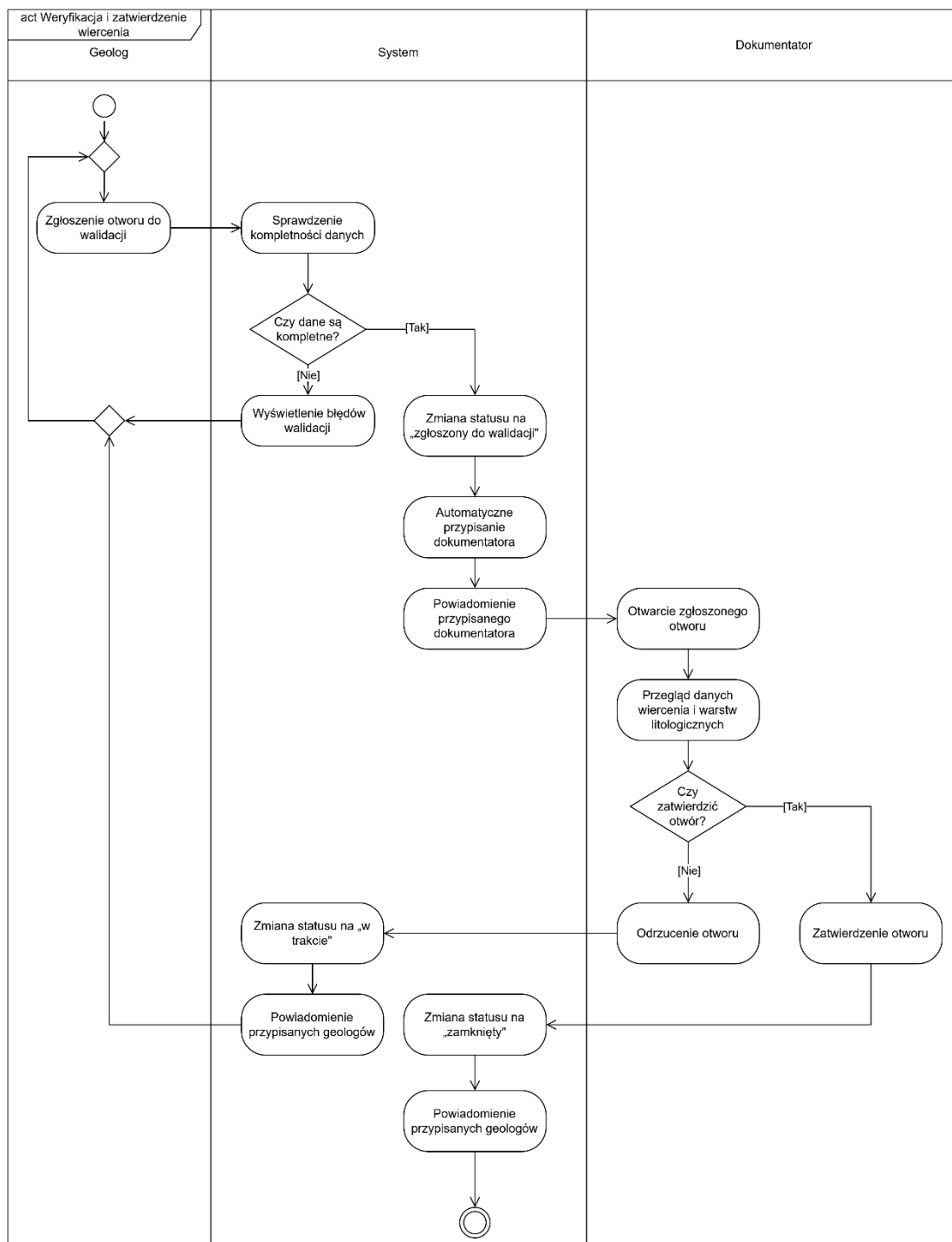
Tworzenie wiercenia jest centralnym procesem pracy geologa terenowego. Obejmuje wprowadzanie kolejnych warstw litologicznych, dokumentowanie ich parametrów oraz załączanie zdjęć stref. Proces realizowany jest w trybie *offline* ze względu na ograniczoną dostępność łączności w terenie, a wprowadzone dane synchronizowane są z serwerem po odzyskaniu połączenia. Diagram aktywności procesu tworzenia wiercenia przedstawia Rysunek 25.



Rysunek 25. Diagram aktywności – tworzenie wiercenia. Źródło: Opracowanie własne.

3.10.4. Diagram aktywności dla weryfikacji formularza wiercenia

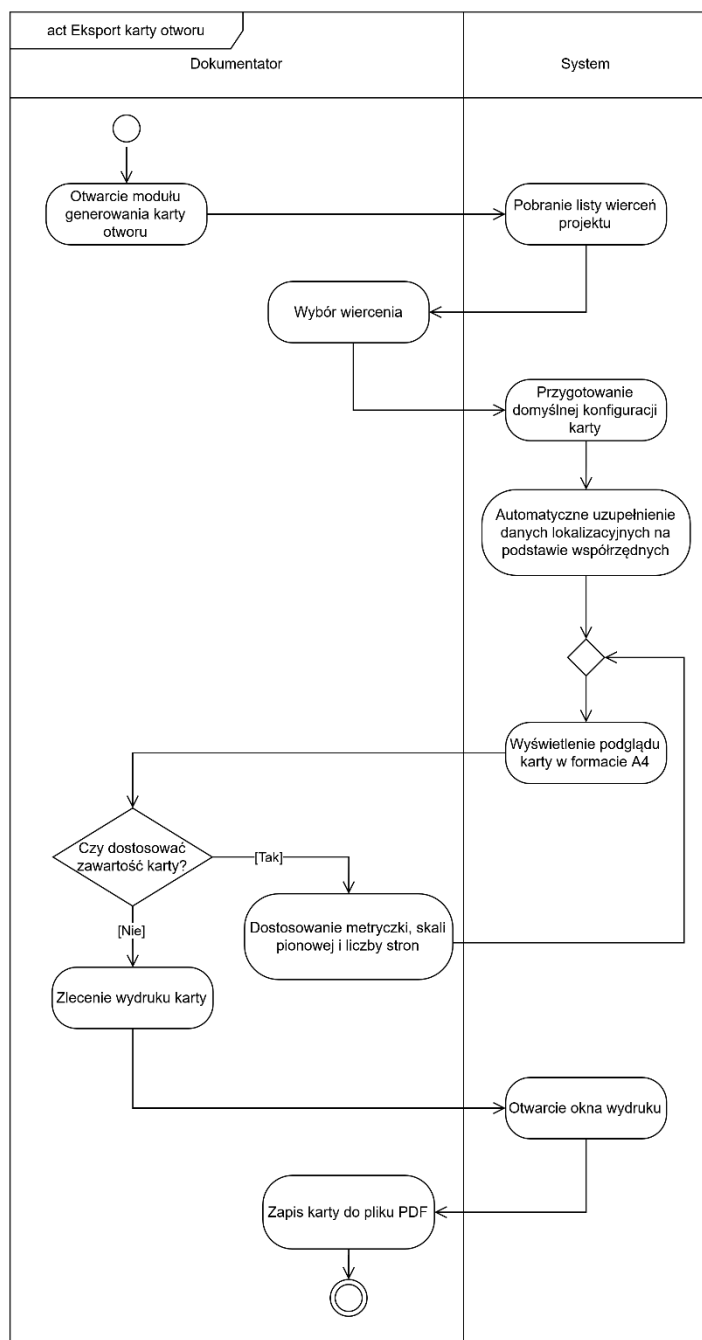
Po zakończeniu pracy w terenie geolog składa wiercenie do akceptacji, przekazując odpowiedzialność za jego weryfikację dokumentatorowi. Dokumentator analizuje wprowadzone dane, opcjonalnie inicjuje dyskusje na temat poszczególnych pól formularza, a następnie podejmuje decyzję o zatwierdzeniu lub odrzuceniu wiercenia do poprawy. Diagram aktywności procesu weryfikacji formularza wiercenia przedstawia Rysunek 26.



Rysunek 26. Diagram aktywności – weryfikacja i zatwierdzenie wiercenia. Źródło: Opracowanie własne.

3.10.5. Diagram aktywności dla eksportu karty otworu

Eksport karty otworu umożliwia wygenerowanie dokumentu PDF zawierającego kartę otworu badawczego, na którą składają się metryczka z danymi identyfikacyjnymi i lokalizacyjnymi otworu, graficzny profil litologiczny oraz tabelaryczne zestawienie warstw wraz z ich parametrami geotechnicznymi, w tym symbolami gruntu zgodnymi z normami PN-EN ISO oraz PN-B, stanem i wilgotnością gruntu, głębokością zwierciadła wody i przynależnością do warstwy geotechnicznej. Karta jest generowana dynamicznie z wykorzystaniem mechanizmu wydruku wbudowanego w przeglądarkę. Przed wygenerowaniem dokumentu użytkownik może dostosować zawartość metryczki, skalę pionową profilu oraz liczbę stron. Diagram aktywności procesu eksportu karty otworu przedstawia Rysunek 27.



Rysunek 27. Diagram aktywności – eksport karty otworu. Źródło: Opracowanie własne.

4. Technologie i narzędzia

Celem tego rozdziału jest zaprezentowanie stosu technologicznego, w którym projekt został zrealizowany. Dla każdej z opisanych technologii wskazano jej główne cechy oraz uzasadniono wybór w kontekście wymagań systemu.

4.1. Java i Spring Boot

Java to obiektowy i statycznie typowany język programowania opracowany przez firmę Sun Microsystems w 1995 roku, a obecnie rozwijany przez Oracle Corporation [23]. Jedną z kluczowych cech języka jest niezależność od platformy sprzętowej. Kod źródłowy kompilowany jest do kodu bajtowego wykonywanego przez maszynę wirtualną Java (JVM – *Java Virtual Machine*). Dzięki temu rozwiązaniu aplikacje napisane w Javie mogą działać w różnych systemach operacyjnych bez konieczności modyfikacji kodu aplikacji.

W projekcie KRET wykorzystano wersję Java 21, która jest wydaniem z długoterminowym wsparciem (ang. *Long-Term Support*, LTS) opublikowanym we wrześniu 2023 [23]. Wybór wersji LTS podyktowany był stabilnością oraz gwarancją wieloletniego wsparcia ze strony producenta oraz szeroką kompatybilnością z bibliotekami i narzędziami dostępnymi w ekosystemie Java.

Spring Boot to szkielet aplikacyjny oparty na platformie Spring, którego celem jest uproszczenie procesu wytwarzania aplikacji w języku Java [24]. Ten szkielet aplikacyjny realizuje zasadę konwencji ponad konfiguracją (ang. *convention-over-configuration*), czyli dostarcza domyślne ustawienia i rozwiązania dla większości typowych scenariuszy, jednocześnie umożliwiając ich nadpisanie w razie potrzeby. Spring Boot eliminuje konieczność czasochłonnej, ręcznej konfiguracji serwera aplikacji. Aplikację można uruchomić jako samodzielny proces z wbudowanym serwerem HTTP.

W Projekcie wykorzystano Spring Boot w wersji 3.4.1 wraz z następującymi modułami [24]:

- Spring Web – obsługa żądań HTTP i budowa interfejsu REST;
- Spring Data JPA – warstwa dostępu do bazy danych z wykorzystaniem mechanizmu mapowania obiektowo-relacyjnego realizowanego przez Hibernate;
- Spring Security – mechanizmy uwierzytelniania i autoryzacji (Opisane w rozdziale 4.13);
- Spring Validation – sprawdzanie danych wejściowych z wykorzystaniem adnotacji;
- Spring Actuator – monitorowanie stanu aplikacji i udostępnianie diagnostycznych punktów końcowych;
- Spring Mail – wysyłanie wiadomości e-mail wykorzystanie do obsługi systemu zaproszeń;
- Thymeleaf – silnik szablonów służący do generowania treści wiadomości e-mail w systemie zaproszeń.

Do budowania projektu wykorzystano Apache Maven [25]. Serwerowy kod aplikacji KRET został zorganizowany zgodnie z wzorcem architektury warstwowej (ang. *layered architecture*). Wydzielono następujące pakiety:

- *controller* – obsługa żądań HTTP;
- *service* – logika biznesowa;
- *repository* – dostęp do danych;
- *entity* – encje bazodanowe;
- *dto* – obiekty transferu danych;
- *config* – konfiguracja aplikacji;

- *security* – mechanizmy bezpieczeństwa;
- *exception* – obsługa wyjątków.

Listing 1 przedstawia przykładową definicję kontrolera REST w Spring Boot. Widoczne są charakterystyczne adnotacje: `@RestController` definiująca klasę jako kontroler REST oraz `@RequestMapping` określające ścieżkę bazową, pod którą udostępniany jest zasób. Adnotacja `@GetMapping` kieruje żądania metody HTTP GET do konkretnej metody kontrolera, a `@PathVariable` wiąże fragment ścieżki URL z parametrem metody.

Listing 1. Przykładowa definicja kontrolera REST w Spring Boot. Źródło: [24]

```
@RestController
@RequestMapping("/api/items")
@RequiredArgsConstructor
public class ItemController {

    private final ItemService itemService;

    @GetMapping("/{id}")
    public ResponseEntity<Item> getItem(@PathVariable Long id) {
        Item item = itemService.findById(id);
        return ResponseEntity.ok(item);
    }
}
```

Wybór języka Java oraz szkielet aplikacyjny Spring Boot podyktowany był kilkoma czynnikami. Java oferuje dojrzały ekosystem bibliotek, silne typowanie sprzyjające wczesnemu wykrywaniu błędów oraz szeroką społeczność zapewniającą dostępność wsparcia i dokumentacji. Spring Boot dodatkowo przyspiesza proces wytwarzania oprogramowania dzięki wstępnej konfiguracji i zestawowi gotowych modułów. Istotnym czynnikiem była również znajomość ekosystemu Java wyniesiona z programu studiów na kierunku Informatyka PJATK.

4.2. PostgreSQL i PostGIS

PostgreSQL to otwarty źródłowy, obiektowo-relacyjny system zarządzania bazą danych, rozwijany przez społeczność od 1996 roku [26]. System ten jest zgodny ze standardem SQL i jednocześnie oferuje zaawansowane mechanizmy wykraczające poza typowe bazy relacyjne takie jak obsługa typów zdefiniowanych przez użytkownika, dziedziczenie tabel rozbudowany system indeksów oraz kontrola współbieżności z wykorzystaniem wielowersyjności. W projekcie wykorzystano PostgreSQL w wersji 15 uruchamiany jako kontener Docker [26].

PostGIS to rozszerzenie bazy danych PostgreSQL dodające obsługę danych przestrzennych [27]. Umożliwia przechowywanie obiektów geometrycznych, wykonywanie zapytań przestrzennych oraz tworzenie indeksów przestrzennych przyspieszających wyszukiwanie. W systemie KRET rozszerzenie to wykorzystywane jest do przechowywania lokalizacji otworów badawczych jako obiektów typu `geometry (Point, 4326)`, gdzie 4326 oznacza układ współrzędnych WGS 84, powszechnie stosowany w systemach GPS i aplikacjach kartograficznych. Na kolumnie lokalizacji utworzono indeks przestrzenny GIST, umożliwiający w przyszłości rozszerzenie systemu o zapytania przestrzenne, takie jak wyszukiwanie otworów w zadanym obszarze geograficznym.

Integracja PostGIS z warstwą aplikacji odbywa się za pośrednictwem biblioteki Hibernate Spatial, która rozszerza mechanizm ORM szkieletu aplikacyjnego Hibernate o obsługę typów geometrycznych [28]. Dzięki temu lokalizacje otworów są mapowane na obiekty klasy `Point` z biblioteki JTS (Java Topology Suite) i mogą być przetwarzane bezpośrednio w kodzie Java.

Listing 2 przedstawia przykładową encję bazodanową z polem geometrycznym typu punkt. Anotacja `@Column` z parametrem `columnDefinition` definiuje typ kolumny jako punkt w układzie WGS 84, a pole typu `Point` z biblioteki JTS przechowuje współrzędne geograficzne. Hibernate Spatial obsługuje konwersję między obiektami JTS, a typami PostGIS w sposób przezroczysty dla programisty.

Listing 2. Mapowanie pola geometrycznego w encji JPA z wykorzystaniem Hibernate Spatial. Źródło: [28]

```
@Entity
@Table(name = "places")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Place {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String name;

    @Column(
        columnDefinition = "geometry(Point,4326)",
        nullable = false
    )
    private Point location;
}
```

Wybór PostgreSQL podyktowany był jego otwartością, dojrzałością oraz dostępnością rozszerzenia PostGIS, które jest niezbędne w kontekście aplikacji geologicznej operującej na danych przestrzennych. Alternatywne systemy bazodanowe oferują ograniczone wsparcie dla danych geoprzestrzennych w porównaniu z PostGIS.

4.3. React

React to biblioteka języka JavaScript służąca do implementacji interfejsów użytkownika, rozwijana przez firmę Meta od 2013 roku [29]. Podstawową koncepcją React jest podział interfejsu na komponenty. Komponenty to samodzielne wielokrotnie wykorzystywane fragmenty interfejsu, z których każdy zarządza własnym stanem i logiką renderowania. React stosuje deklaratywny model programowania, czyli programista opisuje, jak interfejs powinien wyglądać dla danego stanu, a biblioteka samodzielnie aktualizuje drzewo DOM przeglądarki.

Kluczowym mechanizmem React jest wirtualny DOM. Jest to lekka reprezentacja rzeczywistego drzewa DOM przechowywana w pamięci. Przy każdej zmianie stanu komponentu React porównuje nowy wirtualny DOM z poprzednim i aplikuje jedynie różnice do rzeczywistego DOM, co minimalizuje kosztowne operacje na drzewie dokumentu.

W projekcie KRET wykorzystano React w wersji 19 do budowy aplikacji internetowej [29]. Interfejs podzielono na komponenty funkcyjne z wykorzystaniem mechanizmu haków, takich jak `useState` do zarządzania stanem lokalnym czy `useEffect` do obsługi efektów ubocznych. Komponenty komunikują się z serwerem przez żądania HTTP realizowane za pomocą biblioteki Axios.

Wybór React był podyktowany jego dominującą pozycją wśród bibliotek interfejsu użytkownika, rozbudowanym ekosystemem oraz znajomością tej technologii w zespole. Listing 3 przedstawia przykładowy komponent funkcyjny w React. Widoczne są charakterystyczne elementy biblioteki:

deklaracja interfejsu definiującego właściwości komponentu (`GreetingProps`), komponent funkcyjny z destrukuryzacją właściwości oraz składnia JSX łącząca logikę z deklaratywnym opisem struktury interfejsu.

Listing 3. Przykładowy komponent funkcyjny w React z typami TypeScript. Źródło: [29]

```
interface GreetingProps {
  name: string;
}

export const Greeting: React.FC<GreetingProps> = ({ name }) => {
  return (
    <div>
      <h1 className="text-xl font-semibold mb-2">Hello, {name}!</h1>
      <p className="text-muted-foreground">Welcome to React.</p>
    </div>
  );
};
```

Next.js to szkielet aplikacyjny oparty na bibliotece React, rozwijany przez firmę Vercel [30]. Rozszerza możliwości React o mechanizmy niezbędne do budowy pełnoprawnych aplikacji internetowych, takie jak system routingu oparty na strukturze plików, renderowanie po stronie serwera, generowanie stron statycznych oraz obsługę komponentów serwerowych.

W projekcie KRET wykorzystano Next.js w wersji 16 z mechanizmem App Router, który organizuje routing aplikacji na podstawie struktury katalogów [30]. Zastosowano grupy tras (ang. *route groups*) do logicznego podziału aplikacji na trzy sekcje:

- (*auth*) – ekrany logowania i rejestracji;
- (*logged*) – ekrany dostępne po zalogowaniu;
- (*marketing*) – strony publiczne.

Trasy dynamiczne, takie jak `/projects/[id]/holes/[holeId]` umożliwiają nawigację do konkretnych zasobów na podstawie ich identyfikatorów. Next.js automatycznie generuje ścieżki URL na podstawie nazw katalogów i plików *page.tsx*.

W konfiguracji projektu włączono kompilator React, który automatycznie optymalizuje ponowne renderowania komponentów. W trybie deweloperskim skonfigurowano również mechanizm przepisywania adresów, przekierowujący żądania do serwera API, co eliminuje problemy z polityką CORS podczas lokalnego testowania.

Wybór Next.js podyktowany był potrzebą posiadania gotowego systemu routingu, wsparcia dla renderowania serwerowego oraz rozwiniętego ekosystemu narzędzi budowania i wdrażania aplikacji React.

4.4. TypeScript

TypeScript to język programowania rozwijany przez firmę Microsoft, będący nadzbiorem języka JavaScript [31]. Rozszerza JavaScript o statyczny system typów, który umożliwia wykrywanie błędów na etapie kompilacji, zanim kod zostanie uruchomiony w przeglądarce. Kod TypeScript jest konwertowany (ang. *transpile*) do czystego JavaScript, co oznacza pełną kompatybilność z istniejącymi bibliotekami.

Kluczowe elementy języka wykorzystane w projekcie KRET to interfejsy definiujące kształt obiektów, typy unii ograniczające dopuszczalne wartości, właściwości opcjonalne oraz typy generyczne. Mechanizmy te pozwalają na precyzyjne opisanie kontraktów między komponentami systemu co znacząco ułatwia współpracę w zespole i zmniejsza ryzyko błędów związanych z niezgodnością danych.

W Projekcie KRET TypeScript w wersji 5 wykorzystano zarówno w aplikacji internetowej jak i mobilnej, a także w bibliotece współdzielonej [31]. Listing 4 przedstawia fragment definicji typów z biblioteki współdzielonej. Typ `UserRole` ogranicza dopuszczalne wartości do trzech ról, a interfejs `User` definiuje kształt obiektu zwracanego przez API.

Listing 4. Przykładowe definicje typów w TypeScript: typ unii oraz interfejs. Źródło: [31].

```
export type UserRole = 'ADMIN' | 'USER' | 'GUEST';

export interface User {
  id: number;
  firstName: string;
  lastName: string;
  email: string;
  role: UserRole;
  active: boolean;
}
```

Wybór TypeScript podyktowany był potrzebą zapewnienia bezpieczeństwa typów. Dzięki definicjom typów w współdzielonej bibliotece, aplikacja internetowa i mobilna korzystają z tych samych kontraktów danych, co eliminuje rozbieżności i błędy między platformami.

4.5. Tailwind CSS

Tailwind CSS to szkielet aplikacyjny oparty na podejściu *utility-first*, w którym tworzenie stylów elementów interfejsu odbywa się za pomocą predefiniowanych klas narzędziowych [32]. Przypisywanych bezpośrednio w znacznikach HTML lub JSX. W odróżnieniu od tradycyjnych szkieletów aplikacyjnych CSS, takich jak Bootstrap, Tailwind nie narzuca gotowych komponentów wizualnych. Zamiast tego udostępnia niskopoziomowe klasy odpowiadające konkretnym właściwościom CSS, takim jak marginesy, kolory, rozmiary czcionek czy układ elementów w kontenerze.

Przykładem zastosowania w projekcie KRET jest Listing 3, w którym klasy takie jak `text-xl`, `font-semibold`, `mb-2` czy `text-muted-foreground` definiują wygląd komponentu bezpośrednio w kodzie JSX. Podejście to eliminuje konieczność tworzenia osobnych plików CSS i nadawania nazw klasom, co przyspiesza proces implementacji interfejsu.

W projekcie wykorzystano Tailwind CSS w wersji 4 wraz z rozszerzeniem `@tailwindcss/typography` odpowiedzialnym za styl komponentów tekstowych, takich jak artykuły w bazie wiedzy [32]. Konfiguracja szkieletu aplikacyjnego ogranicza się do wskazania plików źródłowych, w których Tailwind CSS wyszukuje wykorzystane klasy, co pozwala na automatyczne usunięcie nieużywanych stylów z wynikowego pliku CSS.

Wybór Tailwind CSS podyktowany był szybkością prototypowania interfejsu, spójnością wizualną wynikającą z ograniczonego zestawu predefiniowanych wartości oraz dobrą integracją z ekosystemem React i Next.js.

4.6. React Native i Expo

React Native to szkielet aplikacyjny do tworzenia aplikacji mobilnych z wykorzystaniem JavaScript i biblioteki React, rozwijany przez firmę Meta [33]. W odróżnieniu od rozwiązań hybrydowych opartych na technologii WebView, React Native korzysta z natywnych komponentów systemu operacyjnego – elementy interfejsu takie jak `View`, `Text` czy `TouchableOpacity` są tłumaczone na odpowiednie im widżety platformy Android lub iOS. Dzięki temu aplikacja zachowuje

wygląd i zachowanie charakterystyczne dla danego systemu, jednocześnie umożliwiając współdzielenie kodu między platformami Android i iOS.

Expo to platforma narzędziowa zbudowana wokół React Native, która upraszcza proces wytwarzania, budowania i dystrybucji aplikacji mobilnych [34]. Narzędzie to udostępnia zestaw gotowych modułów zapewniających dostęp do funkcji urządzenia, takich jak aparat fotograficzny, lokalizacja GPS, system plików czy powiadomienia wypychane (ang. *push*), bez konieczności ręcznej konfiguracji kodu natywnego. W projekcie KRET wykorzystano Expo SDK w wersji 54 wraz z następującymi modułami [34]:

- expo-location – dostęp do lokalizacji GPS urządzenia, wykorzystywany do wyświetlania pozycji użytkownika na mapie;
- expo-camera – obsługa aparatu fotograficznego do dokumentowania warstw geologicznych;
- expo-notifications – obsługa powiadomień wypychanych informujących o przypisaniu do projektu, nowych dyskusjach i zmianach statusów;
- expo-image-picker – wybór zdjęć z galerii urządzenia;
- expo-document-picker – wybór plików z pamięci urządzenia.

Nawigacja między ekranami w aplikacji mobilnej zrealizowana została za pomocą biblioteki React Navigation, z podziałem na stos nawigacyjny dla przejść między ekranami oraz nawigacją dolną dla głównych sekcji aplikacji.

Listing 5 przedstawia przykładowy komponent React Native. Widoczne są elementy charakterystyczne dla tej technologii: natywne komponenty (TouchableOpacity, Text) tłumaczone do widżetów odpowiedniej platformy oraz obiekt stylów tworzony za pomocą StyleSheet.create.

Listing 5. Przykładowy komponent React Native z wykorzystaniem natywnych elementów interfejsu.
Źródło: [33]

```
import { Text, TouchableOpacity, StyleSheet } from 'react-native';

const Button = ({ label, onPress }) => {
  return (
    <TouchableOpacity onPress={onPress} style={styles.button}>
      <Text style={styles.label}>{label}</Text>
    </TouchableOpacity>
  );
};

const styles = StyleSheet.create({
  button: {
    paddingVertical: 12,
    paddingHorizontal: 24,
    backgroundColor: '#007AFF',
    borderRadius: 8,
  },
  label: {
    color: 'FFFFFF',
    fontSize: 16,
    fontWeight: '600',
    textAlign: 'center',
  },
});
```

Budowanie aplikacji mobilnej odbywa się za pośrednictwem usługi EAS Build (Expo Application Services Build), która kompiluje projekt w chmurze i generuje gotowe pliki instalacyjne dla systemów Android oraz iOS. Eliminuje to konieczność konfigurowania lokalnego środowiska budowania, w szczególności Xcode na macOS wymaganego do kompilacji aplikacji iOS.

Wybór React Native z Expo podyktowany był możliwością współdzielenia kodu i kompetencji z aplikacją internetową (wspólna znajomość React i TypeScript), a także dostępnością gotowych modułów Expo przyspieszających proces wytwórczy obsługi funkcji natywnych urządzenia.

4.7. Leaflet i mapy

Wyświetlanie danych przestrzennych na mapie stanowi kluczowy element interfejsu użytkownika systemu geologicznego. W projekcie KRET zastosowano dwa rozwiązania mapowe osobne dla aplikacji internetowej i mobilnej. Wynika to z różnic technologicznych między platformami.

W aplikacji mobilnej wykorzystano bibliotekę `react-native-maps` w wersji 1.20.1, która odpowiada za wyświetlanie natywnych map Google Maps (Android) lub Apple Maps (iOS) [36]. Biblioteka ta oferuje komponenty takie jak `MapView`, `Marker` i `Callout` umożliwiające wyświetlanie lokalizacji otworów badawczych, pozycji GPS użytkownika oraz nawigację do wybranych punktów.

W aplikacji internetowej wykorzystano bibliotekę Leaflet w wersji 1.9.4 wraz z nakładką React Leaflet 5.0, która integruje Leaflet z komponentami React [35]. Leaflet to otwarta biblioteka JavaScript do wyświetlania interaktywnych map w przeglądarkach internetowych, obsługująca warstwy kafelkowe, znaczniki, wyskakujące okienka oraz zdarzenia użytkownika, takie jak przeciąganie i przybliżanie. Jako źródło danych kartograficznych wykorzystano OpenStreetMap, udostępniający darmowe kafelki mapowe [37].

Listing 6 przedstawia przykładowy komponent interaktywnej mapy z wykorzystaniem React Leaflet. Komponent `MapContainer` wstępnie ładuje mapę z określonym środkiem i poziomem przybliżenia, `TileLayer` ładuje kafelki z OpenStreetMap, a komponent `Marker` umieszcza znacznik w wybranej lokalizacji wraz z wyskakującym okienkiem opisowym.

Listing 6. Przykładowa interaktywna mapa z wykorzystaniem React Leaflet i OpenStreetMap. Źródło: [35].

```
import { MapContainer, TileLayer, Marker, Popup } from 'react-leaflet';

export const SimpleMap = () => {
  return (
    <MapContainer
      center={[52.2297, 21.0122]}
      zoom={13}
      style={{ height: '400px', width: '100%' }}
    >
      <TileLayer
        url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
        attribution='© OpenStreetMap contributors'
      />
      <Marker position={[52.2297, 21.0122]}>
        <Popup>Warszawa</Popup>
      </Marker>
    </MapContainer>
  );
};
```

Zastosowanie dwóch różnych bibliotek mapowych wynika z ograniczeń platformowych. Leaflet działa wyłącznie w środowisku przeglądarki, natomiast `react-native-maps` korzysta z natywnych silników kartograficznych urządzenia mobilnego. Wspólna warstwa danych zapewnia spójność informacji przestrzennych między platformami.

4.8. MinIO

MinIO to otwarty serwer pamięci obiektowej zgodny z protokołem Amazon S3 [38]. Umożliwia przechowywanie plików w postaci obiektów zorganizowanych w zasobnikach (ang. *buckets*), oferując interfejs HTTP do operacji zapisu, odczytu i usuwania plików. W odróżnieniu od tradycyjnego systemu plików, pamięć obiektowa jest skalowalna, niezależna od systemu operacyjnego i łatwa do integracji z aplikacjami za pośrednictwem API.

W systemie KRET serwer MinIO przechowuje następujące kategorie plików:

- zdjęcia warstw geologicznych;
- logotypy firm geologicznych;
- awatary użytkowników;
- załączniki artykułów bazy wiedzy;
- pliki powiązane z polami niestandardowymi.

Dostęp do plików realizowany jest za pomocą mechanizmu podpisanych adresów URL. Serwer API generuje tymczasowy adres URL z ograniczonym czasem ważności, który pozwala aplikacjom klienckim na bezpośrednie pobranie pliku z MinIO bez konieczności przesyłania go przez serwer API. Mechanizm ten odciąża API i zmniejsza zużycie przepustowości.

Listing 7 przedstawia podstawową konfigurację klienta MinIO w aplikacji Spring Boot. Bean `MinioClient` tworzony jest z wykorzystaniem wzorca projektowego budowniczego, z podaniem adresu serwera oraz danych uwierzytelniających.

Listing 7. Konfiguracja klienta MinIO w aplikacji Spring Boot. Źródło: [38].

```
@Configuration
public class StorageConfig {

    @Bean
    public MinioClient minioClient() {
        return MinioClient.builder()
            .endpoint("http://localhost:9000")
            .credentials("accessKey", "secretKey")
            .build();
    }
}
```

MinIO uruchamiany jest jako kontener Docker z dedykowanym wolumenem danych zapewniającym trwałość plików. Wybór MinIO podyktowany był zgodnością z protokołem S3, prostotą wdrożenia w środowisku kontenerowym oraz brakiem konieczności korzystania z płatnych usług chmurowych do przechowywania plików. Dodatkowym atutem jest możliwość bezproblemowej migracji na komercyjne rozwiązanie Amazon S3 po wyjściu z fazy prototypu, bez konieczności modyfikacji kodu aplikacji.

4.9. Liquibase

Liquibase to narzędzie do zarządzania migracjami schematu bazy danych [39]. Umożliwia definiowanie zmian struktury bazy danych w postaci plików konfiguracyjnych (XML, YAML, JSON lub SQL), które są następnie wykonywane automatycznie przy uruchomieniu aplikacji. Każda zmiana rejestrowana jest w tabeli śledzenia wersji schematu, co gwarantuje, że dana migracja zostanie wykonana dokładnie jeden raz w danym środowisku wdrożeniowym.

W projekcie KRET zdefiniowano ponad 30 plików migracji, numerowanych sekwencyjnie w formacie licznym od 0, na przykład *000-enable-postgis.xml*. Kolejne pliki obejmują tworzenie tabel, dodawanie kolumn, modyfikację ograniczeń, zakładanie indeksów oraz zmiany w relacjach między tabelami. Plik nadrzędny *db.changelog-master.xml* zawiera listę wszystkich migracji w kolejności ich wykonania. Dodatkowo zdefiniowano plik z danymi testowymi, uruchamiany wyłącznie w kontekście środowisk oznaczonych jako deweloperskie.

Listing 8 przedstawia przykładowy zbiór zmian Liquibase dodający kolumnę do istniejącej tabeli wraz z ograniczeniami nakładającymi wymóg unikalności i wartości niepustej. Widoczna jest struktura zmian z unikalnym identyfikatorem oraz blok rollback umożliwiający wycofanie migracji w przypadku problemów.

Listing 8. Przykładowy changeset Liquibase dodający kolumnę do tabeli. Źródło: [39].

```
<changeSet id="001-add-email-column" author="developer">
  <comment>
    Add email column to users table
  </comment>
  <addColumn tableName="users">
    <column name="email" type="varchar(255)">
      <constraints nullable="false" unique="true"/>
    </column>
  </addColumn>
  <rollback>
    <dropColumn tableName="users" columnName="email"/>
  </rollback>
</changeSet>
```

Wybór Liquibase podyktowany był potrzebą wersjonowania zmian schematu bazy danych w repozytorium kodu, co zapewnia powtarzalność wdrożeń i eliminuje ręczne modyfikację struktury bazy danych na środowisku produkcyjnym.

4.10. Lombok

Lombok to biblioteka języka Java, która automatycznie generuje powtarzalny kod na etapie kompilacji za pomocą adnotacji [40]. W typowej aplikacji Java znaczną część kodu źródłowego stanowią metody dostępowe (getter i setter), konstruktory, metody equals, hashCode i toString, które nic nie wnoszą do logiki biznesowej, ale są niezbędne przez konwencje języka Java. Lombok eliminuje konieczność ręcznego pisania tego kodu, generując go automatycznie na podstawie adnotacji umieszczonych nad polem lub klasą.

W projekcie KRET wykorzystano następujące adnotacje Lombok:

- @Getter, @Setter – automatyczne generowanie metod dostępowych pól klas;
- @Data – połączenie @Getter, @Setter, @ToString, @EqualsAndHashCode i @RequiredArgsConstructor w jednej adnotacji;
- @Builder – implementacja wzorca projektowego budowniczego, ułatwiającego tworzenie obiektów z wieloma parametrami;
- @NoArgsConstructor, @AllArgsConstructor – generowanie konstruktora bezargumentowego oraz konstruktora ze wszystkimi polami;
- @RequiredArgsConstructor – generowanie konstruktora dla pól oznaczonych jako final, wykorzystywany do wstrzykiwania zależności w Spring Boot;
- @Slf4j – automatyczne dodanie pola loggera do klasy.

Zastosowanie Lombok jest widoczne w wcześniejszych listingach – Listing 1 wykorzystuje `@RequiredArgsConstructor`, a Listing 2 stosuje `@Getter`, `@Setter`, `@NoArgsConstructor` i `@AllArgsConstructor` w encji bazodanowej.

Wybór Lombok podyktowany został znaczącym zmniejszeniem objętości kodu źródłowego i poprawą jego czytelności. Adnotacje Lombok nie wpływają na działanie aplikacji w czasie wykonania, ponieważ cały kod generowany jest na etapie kompilacji.

4.11. Spring Security i JWT

Spring Security to szkielet aplikacyjny zapewniający mechanizmy uwierzytelniania i autoryzacji w aplikacjach opartych na platformie Spring [41]. Oferuje konfigurowalny łańcuch filtrów, który przechwytuje każde żądanie HTTP i weryfikuje tożsamość użytkownika przed przekazaniem jego żądania do właściwego kontrolera.

W projekcie KRET uwierzytelnianie oparte jest na tokenach JWT (JSON Web Token) [21] co oznacza, że serwer nie przechowuje informacji o sesjach użytkowników. Każde żądanie zawiera token w nagłówku `Authorization`, który jest samodzielnie weryfikowany przez filtr bezpieczeństwa. Zastosowano schemat dwóch tokenów:

- *Access token* – krótkotrwały token JWT podpisywany algorytmem HMAC, zawierający identyfikator użytkownika i czas ważności;
- *Refresh token* – długotrwały token przechowywany w bazie danych, umożliwiający uzyskanie nowego *access token* bez ponownego logowania.

Przy każdym odświeżeniu tokena poprzedni *refresh token* jest unieważniany i generowany jest nowy, co ogranicza ryzyko nieautoryzowanego dostępu w przypadku przechwycenia tokena. Hasła użytkowników przechowywane są w postaci zaszyfrowanej z wykorzystaniem algorytmu BCrypt [42].

Listing 9 przedstawia przykładową konfigurację łańcucha filtrów bezpieczeństwa w Spring Security. Widoczne jest wyłączenie zabezpieczenia CSRF, zdefiniowanie końcówek dostępnych publicznie bez uwierzytelniania oraz wymóg uwierzytelnienia dla pozostałych żądań HTTP.

Listing 9. Przykładowa konfiguracja łańcucha filtrów w Spring Security. Źródło: [41].

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
        .csrf(csrf -> csrf.disable())
        .authorizeHttpRequests(authz -> authz
            .requestMatchers("/public/**").permitAll()
            .anyRequest().authenticated())
        .sessionManagement(session ->
            session.sessionCreationPolicy(SessionCreationPolicy.STATELESS));
    return http.build();
}
```

Wybór Spring Security z tokenami JWT podyktowany był bezstanową architekturą serwera API, która umożliwia łatwe skalowanie horyzontalne oraz integrację z aplikacją internetową, jak i mobilną bez konieczności zarządzania sesjami po stronie serwera.

4.12. Apache POI

Apache POI to biblioteka języka Java umożliwiająca tworzenie i odczytywanie plików w formatach pakietu Microsoft Office, w szczególności arkuszy kalkulacyjnych (XLSX) oraz

dokumentów Word (DOCX) [43]. Biblioteka udostępnia interfejs obiektowy do operacji na arkuszach, wierszach i komórkach, co pozwala na generowanie złożonych dokumentów bez konieczności korzystania z oprogramowania Microsoft Office.

W systemie KRET biblioteka Apache POI wykorzystywana jest do eksportu danych geologicznych do formatu XLSX. Funkcjonalność eksportu umożliwia wygenerowanie skoroszytu Excel zawierającego arkusze z danymi otworów badawczych, odwiertów, warstw geologicznych, sondowań oraz pomiarów, z uwzględnieniem nazw typów litologicznych i ich symboli zgodnych z normami geotechnicznymi. Użytkownik może konfigurować zakres eksportowanych pól, ich kolejność oraz nazwy kolumn.

Listing 10 przedstawia podstawowe operacje generowania arkusza Excel przy użyciu biblioteki Apache POI. Widoczne jest tworzenie skoroszytu (ang. *workbook*), arkusza (ang. *sheet*), wiersza nagłówkowego oraz wiersza z danymi, a także zapis wyniku do pliku w formacie XLSX.

Listing 10. Tworzenie arkusza Excel z wykorzystaniem biblioteki Apache POI. Źródło: [43].

```
Workbook workbook = new XSSFWorkbook();
Sheet sheet = workbook.createSheet("Dane");

Row header = sheet.createRow(0);
header.createCell(0).setCellValue("ID");
header.createCell(1).setCellValue("Nazwa");
header.createCell(2).setCellValue("Wartość");

Row row = sheet.createRow(1);
row.createCell(0).setCellValue(1);
row.createCell(1).setCellValue("Pozycja A");
row.createCell(2).setCellValue(100.50);

try (FileOutputStream out = new FileOutputStream("output.xlsx")) {
    workbook.write(out);
}
workbook.close();
```

Wybór Apache POI podyktowany był koniecznością generowania plików Excel, które stanowią powszechny format wymiany danych w branży geologicznej. Alternatywne podejście polegające na eksporcie wyłącznie do formatu CSV nie zapewniłoby możliwości podziału dokumentu na arkusze, formatowania ani umieszczania wielu kategorii danych w jednym pliku.

4.13. Konteneryzacja i wdrożenie

Konteneryzacja to technika wirtualizacji na poziomie systemu operacyjnego, w której aplikacja wraz ze wszystkimi zależnościami uruchamiana jest w izolowanym środowisku zwanym kontenerem. W projekcie KRET zastosowano platformę Docker do konteneryzacji warstwy serwerowej oraz narzędzie Docker Compose do orkiestracji wielu powiązanych kontenerów [44].

Środowisko produkcyjne składa się z następujących kontenerów:

- geokret-api – aplikacja Spring Boot;
- geokret-db – baza danych PostgreSQL;
- geokret-minio – serwer pamięci obiektowej MinIO;
- geokret-minio-init – kontener inicjalizujący, tworzący zasobnik i dane testowe;
- geokret-nginx – serwer pośredniczący NGINX (profil produkcyjny);
- geokret-certbot – automatyczne odnawianie certyfikatów SSL Let's Encrypt (profil produkcyjny).

Obrazy aplikacji Spring Boot budowane są zazwyczaj z wykorzystaniem wieloetapowego procesu (ang. *multi-stage build*). Listing 11 przedstawia przykładowy plik Dockerfile, w którym pierwszy etap kompiluje aplikację z użyciem JDK, a drugi tworzy lekki obraz produkcyjny oparty o JRE. Obraz produkcyjny uruchamiany jest z użyciem dedykowanego użytkownika systemowego, co zwiększa bezpieczeństwo kontenera.

Listing 11. Przykładowy wieloetapowy plik Dockerfile dla aplikacji Spring Boot. Źródło: [44].

```
# Etap budowania
FROM eclipse-temurin:21-jdk-alpine AS build
WORKDIR /app
COPY pom.xml ./
COPY src ./src
RUN ./mvnw package -DskipTests

# Etap produkcyjny
FROM eclipse-temurin:21-jre-alpine
WORKDIR /app
RUN addgroup -S app && adduser -S app -G app
USER app:app
COPY --from=build /app/target/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Docker Compose umożliwia uruchomienie całego środowiska jednym poleceniem. Kontenery komunikują się za pośrednictwem wewnętrznej sieci Docker, a dane bazy danych PostgreSQL i MinIO przechowywane są w wolumenach zapewniających trwałość pomiędzy ponownymi uruchomieniami aplikacji. Dla środowiska produkcyjnego zdefiniowano profil o nazwie `prod`, który uruchamia dodatkowo NGINX [45] jako serwer proxy z zakończeniem SSL oraz Certbot do automatycznego odnawiania certyfikatów Let's Encrypt [46].

Warstwa serwerowa wdrożona jest na instancji Oracle Cloud Infrastructure [47], natomiast aplikacja internetowa hostowana jest na platformie Vercel [30], która zapewnia automatyczne wdrożenia z repozytorium GitHub [50]. Dystrybucja aplikacji mobilnej odbywa się za pośrednictwem usługi EAS Build opisanej w podrozdziale 4.6.

4.14. IntelliJ IDEA

IntelliJ IDEA to zintegrowane narzędzie programistyczne dla języka Java, rozwijane przez firmę JetBrains [48]. Oferuje zaawansowane mechanizmy uzupełniania kodu, refaktoryzacji, debugowania oraz integracji z systemami kontroli wersji i narzędziami budowania aplikacji. W projekcie KRET IntelliJ IDEA wykorzystywano do pracy nad warstwą serwerową, a wbudowana obsługa Maven umożliwiała zarządzanie zależnościami i uruchamianie aplikacji bezpośrednio z poziomu IDE. Wersja Ultimate oferuje również wsparcie dla języków JavaScript i TypeScript, dzięki czemu to samo środowisko wykorzystywano do pracy nad aplikacją internetową, mobilną oraz biblioteką współdzieloną. Eliminowało to konieczność przełączania się między różnymi edytorami. Środowisko wspiera również wtyczki rozszerzające funkcjonalność, w tym wtyczkę Lombok odpowiedzialną za prawidłową obsługę adnotacji opisanych w rozdziale 4.12.

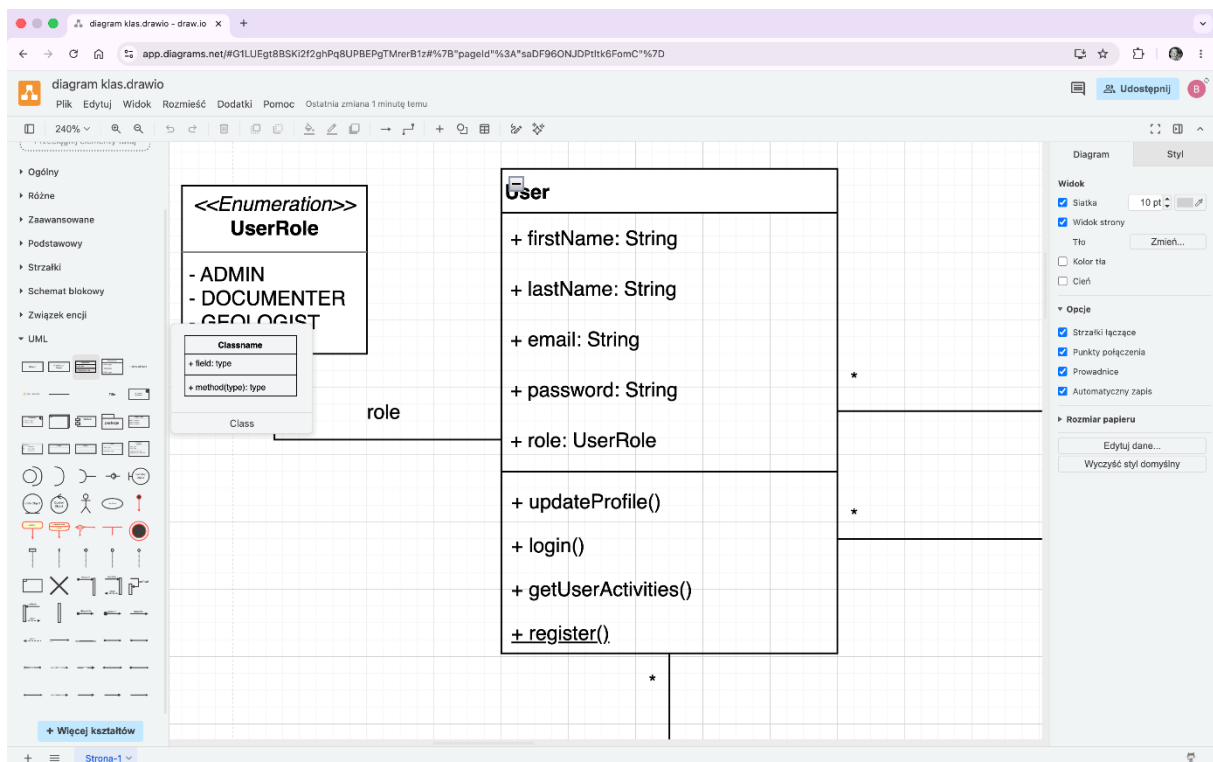
Wybór IntelliJ IDEA podyktowany był jego dominującą pozycją środowisk programistycznych dla języka Java, rozbudowanym wsparciem dla szkieletu aplikacyjnego Spring Boot, obsługą JavaScript i TypeScript w jednym narzędziu oraz dostępnością bezpłatnej licencji edukacyjnej dla studentów.

4.15. Draw.io

Draw.io to bezpłatne narzędzie do tworzenia diagramów i schematów działające w przeglądarce internetowej [49]. Umożliwia projektowanie diagramów klas UML, schematów architektury, diagramów klas, przepływów czy aktywności z wykorzystaniem biblioteki gotowych kształtów i połączeń.

W projekcie KRET narzędzie Draw.io wykorzystano do tworzenia diagramów dokumentacyjnych zamieszczonych w niniejszej pracy, takich jak schemat architektury systemu, diagramy klas czy diagramy aktywności. Pliki diagramów w formacie XML przechowywane są w repozytorium projektu, co umożliwia ich wersjonowanie i edycję przez członków zespołu.

Wybór Draw.io podyktowany był jego bezpłatną dostępnością, prostotą obsługi oraz możliwością eksportu diagramów do formatów graficznych wymaganych w pracy dyplomowej. Interfejs programu Draw.io przedstawia Rysunek 28.



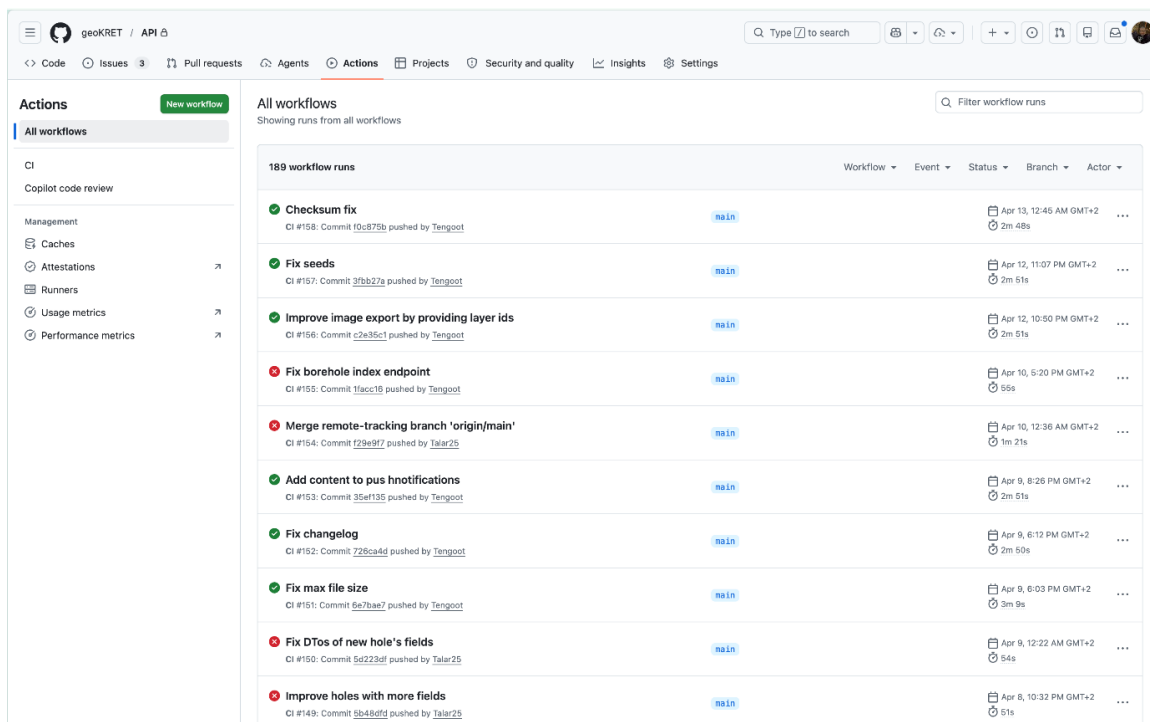
Rysunek 28. Edytor diagramów Draw.io z otwartym diagramem klas systemu KRET. Źródło: opracowanie własne.

4.16. GitHub i GitHub Actions

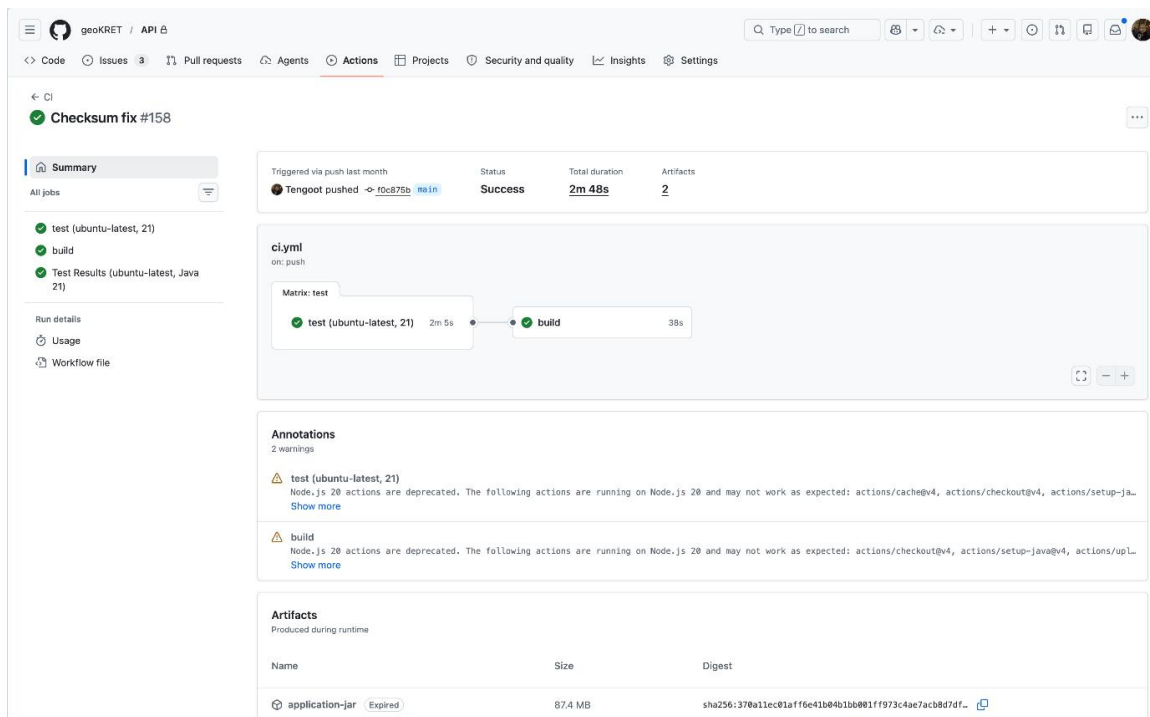
GitHub to platforma hostingowa dla repozytoriów Git, oferująca mechanizmy współpracy zespołowej takiej jak zgłoszenia (ang. *issues*), żądania scalenia (ang. *pull requests*) oraz przeglądanie i ocena kodu (ang. *code reviews*) [50]. W projekcie KRET kod źródłowy podzielony jest na cztery repozytoria: API, WEB, Mobile oraz Shared, zarządzane w ramach organizacji na GitHub.

GitHub Actions to wbudowany mechanizm ciągłej integracji i ciągłego dostarczania (ang. *Continuous Integration / Continuous Delivery, CI/CD*) umożliwiający automatyczne wykonywanie zadań w reakcji na zdarzenia w repozytorium. W projekcie KRET skonfigurowano potok CI dla

repozytorium API, który przy każdym żądaniu scalenia oraz wypchnięciu zmian do gałęzi głównej automatycznie kompiluje projekt i uruchamia testy jednostkowe. Pozwala to na wczesne wykrycie błędów przed scaleniem zmian do głównej gałęzi kodu. Rysunek 29 i Rysunek 30 przedstawiają interfejs GitHub Actions.



Rysunek 29. Historia uruchomień GitHub Actions w repozytorium API systemu KRET. Źródło: opracowanie własne.



Rysunek 30. Szczegóły przebiegu GitHub Actions w repozytorium API systemu KRET. Źródło: Opracowanie własne.

4.17. GitHub Projects

GitHub Projects to narzędzie do zarządzania pracą zespołową zintegrowane z platformą GitHub [50]. Umożliwia tworzenie tablic zadań w widoku Kanban lub tabeli, z możliwością przypisywania zadań do członków zespołu, definiowania terminów oraz śledzenia postępu prac. Zadania powiązane są bezpośrednio z GitHub issues w repozytoriach, co zapewnia spójność między planowaniem, a kodem źródłowym.

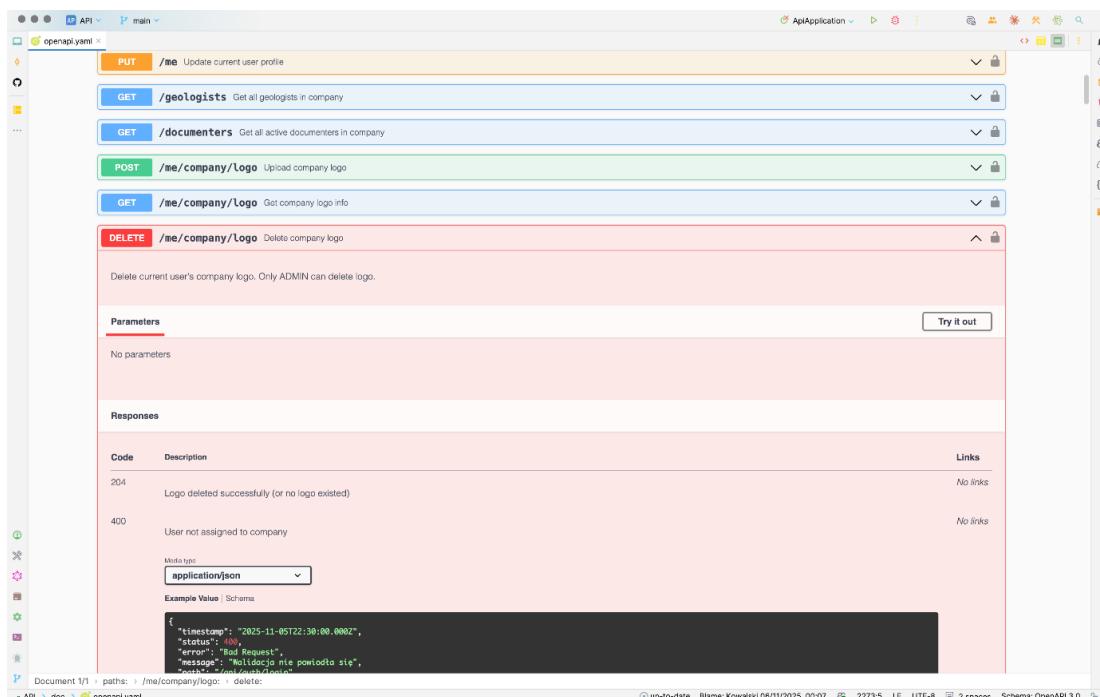
W projekcie KRET narzędzie GitHub Projects wykorzystano do planowania pracy pomiędzy zjazdami, przydzielania zadań między członków zespołu oraz monitorowania postępu realizacji poszczególnych funkcjonalności systemu.

Wybór GitHub Projects podyktowany był integracją z repozytoriami kodu na platformie GitHub, co eliminowało konieczność korzystania z zewnętrznego narzędzia do zarządzania projektem.

4.18. OpenAPI i Swagger

Swagger to zestaw narzędzi służących do projektowania, dokumentowania i wizualizacji interfejsów REST API w formacie YAML lub JSON, umożliwiający formalne zdefiniowanie dostępnych końcówek API, parametrów żądań oraz schematów danych [22][51]. Kluczowym elementem zestawu Swagger jest Swagger UI, który jest przeglądarkowym interfejsem umożliwiającym przeglądanie dostępnych końcówek API, ich parametrów oraz przykładowych odpowiedzi w czytelnej formie graficznej.

W projekcie KRET specyfikacja API utrzymywana jest ręcznie w postaci pliku *openapi.yaml* przechowywanego w repozytorium kodu API [22]. Swagger UI uruchamiany jest lokalnie na stacjach roboczych członków zespołu i służy do przeglądania dokumentacji oraz testowania końcówek podczas prac deweloperskich. Nie jest udostępniany razem z aplikacją serwerową w środowisku produkcyjnym. Interfejs narzędzia Swagger przedstawia Rysunek 31.



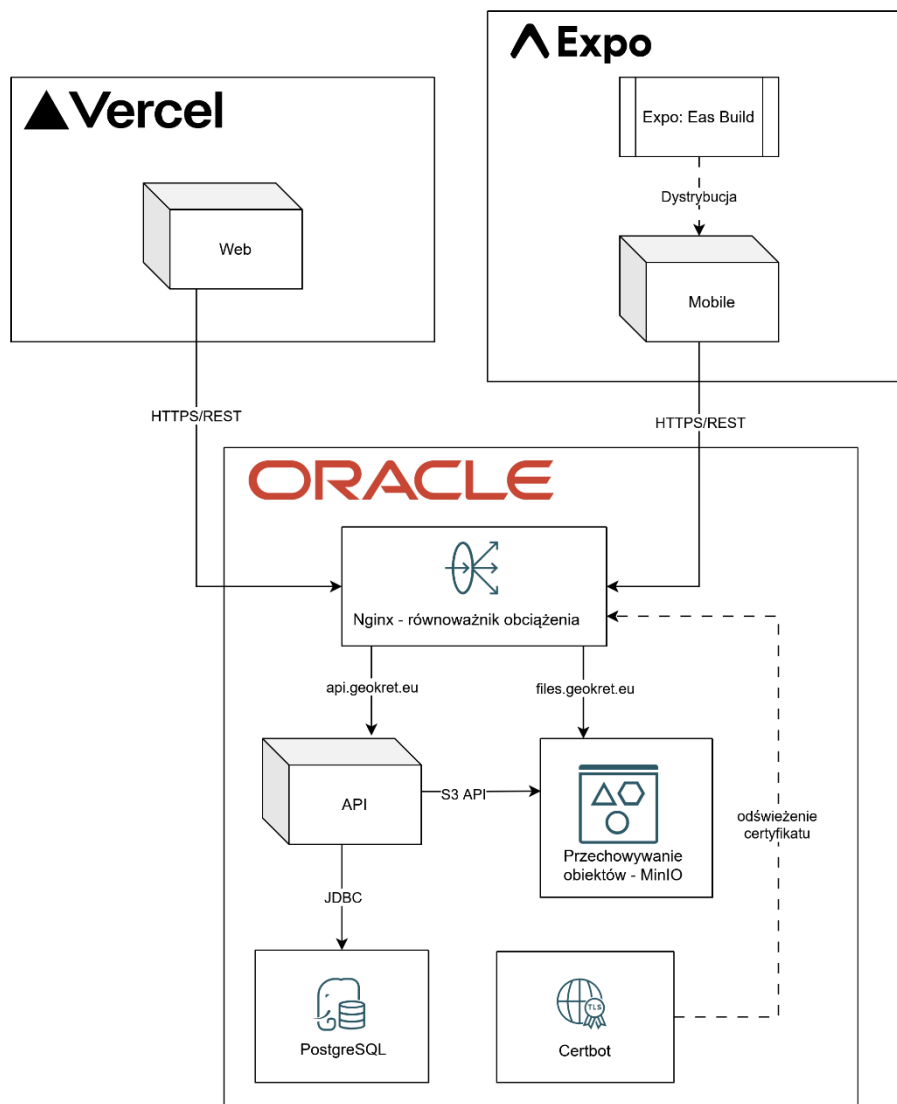
Rysunek 31. Dokumentacja interfejsu API systemu KRET wygenerowana w Swagger UI. Źródło: opracowanie własne.

5. Implementacja

Niniejszy rozdział opisuje kluczowe rozwiązania zastosowane podczas implementacji systemu KRET. Każdy podrozdział koncentruje się na jednym aspekcie technicznym prezentując strukturę kodu, decyzje projektowe oraz fragmenty kodu źródłowego i diagramy tam, gdzie ma to znaczenie.

5.1. Architektura systemu i infrastruktura wdrożeniowa

System KRET zaprojektowano jako architekturę rozproszoną złożoną z czterech komponentów komunikujących się przez interfejs REST oraz współdzielących definicje typów za pośrednictwem biblioteki TypeScript. Serwer uruchamiany jest w środowisku kontenerowym na pojedynczej instancji w chmurze Oracle. Aplikacja internetowa udostępniana jest na platformie Vercel, a aplikacja mobilna budowana i dystrybuowana przez usługę Expo Application Services. Schemat tej architektury wraz z podziałem ról poszczególnych komponentów przedstawia Rysunek 32.



Rysunek 32. Diagram architektury wdrożeniowej systemu KRET. Źródło: Opracowanie własne.

5.1.1. Architektura serwerowa

Warstwa serwerowa została zaimplementowana zgodnie z wzorcem architektury omówionym w rozdziale 4.1. Pakiet `controller` obsługuje żądania `http`, deleguje sprawdzenie danych wejściowych do mechanizmu `Bean Validation` oraz zwraca obiekty transferu danych (ang. *Data Transfer Object*, `DTO`). Pakiet `service` zawiera logikę biznesową. Każdy serwis odpowiada za jeden obszar funkcjonalny (na przykład `HoleService` obsługuje cykl życia otworu, a `BoreholeService` pozwala na zarządzanie wierceniami). Pakiet `repository` udostępnia interfejsy rozszerzające `JpaRepository` z biblioteki `Spring Data`, dzięki czemu większość operacji na bazie danych generowana jest automatycznie z sygnatur metod. Pakiet `entity` zawiera encje JPA opisujące tabele bazy danych. Pakiet `security` mieści filtr `JWT` przechwytyjący żądania zanim dotrą do kontrolerów.

Komunikacja między aplikacjami klienckimi, a serwerem odbywa się wyłącznie za pośrednictwem interfejsu `REST` udostępnionego pod ścieżkami zaczynającymi się od `/projects`, `/files`, `/auth`, `/notifications` i innych.

Serwer nie zarządza sesjami `HTTP`. Każde żądanie identyfikowane jest na podstawie tokenu `JWT` przekazanego w nagłówku `Authorization`, co opisano szczegółowo w podrozdziale 5.3. Konfiguracja `Spring Boot` wymusza tryb bezstanowy, wyłącza ochronę `CSRF` i włącza moduł `CORS` pozwalający na żądania z aplikacji internetowej.

5.1.2. Konteneryzacja z Docker

Środowisko serwerowe definiowane jest w pliku `docker-compose.yml` jako pięć usług połączonych wewnętrzną siecią o nazwie `geokret-network`. Na 5 usług składa się: aplikacja serwerowa, baza danych, pamięć obiektowa `MinIO`, serwer pośredniczący `Nginx` oraz narzędzie `Certbot`. Serwer pośredniczący i `Certbot` są uruchamiane wyłącznie w profilu produkcyjnym. Listing 12 przedstawia fragment tej konfiguracji obejmujący najważniejsze usługi.

Listing 12. Fragment konfiguracji Docker Compose. Źródło: Opracowanie własne.

```
services:
  app:
    build: .
    container_name: geokret-api
    restart: unless-stopped
    environment:
      - SPRING_PROFILES_ACTIVE=prod
      - DATABASE_URL=jdbc:postgresql://db:5432/geokretapi
      - JWT_SECRET=${JWT_SECRET}
      - MINIO_ENDPOINT=http://minio:9000
      - MINIO_PUBLIC_ENDPOINT=${MINIO_PUBLIC_ENDPOINT:-}
    depends_on:
      db:
        condition: service_healthy
      minio:
        condition: service_healthy
    healthcheck:
      test: ["CMD", "wget", "--spider",
             "http://localhost:8080/actuator/health"]
      interval: 30s
      retries: 3
      mem_limit: 768m

  db:
    image: imresamu/postgis:15-3.5-alpine
    container_name: geokret-db
    environment:
      - POSTGRES_DB=geokretapi
      - POSTGRES_USER=geokret
      - POSTGRES_PASSWORD=${DB_PASSWORD}
```

```

volumes:
  - postgres_data:/var/lib/postgresql/data

minio:
  image: minio/minio:latest
  container_name: geokret-minio
  command: server /data --console-address ":9001"
  volumes:
    - minio_data:/data

nginx:
  image: nginx:alpine
  profiles: [prod]
  ports: ["80:80", "443:443"]
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf:ro
    - ./ssl:/etc/nginx/ssl:ro

```

W konfiguracji zastosowano kilka istotnych usprawnień. Aplikacja serwerowa uruchamiana jest pod dedykowanym użytkownikiem systemowym `spring`, co ogranicza skutki potencjalnej eskalacji uprawnień wewnątrz kontenera. Parametr `-XX:MaxRAMPercentage=75.0` razem z flagą `-XX:+UseContainerSupport` instruuje maszynę wirtualną Javy, by respektowała limity pamięci ustawione na poziomie kontenera. Bez tej konfiguracji, aplikacja potencjalnie mogłaby przekroczyć przydzielony limit i zostać zakończona przez system operacyjny. Wbudowany mechanizm `HEALTHCHECK` raz na jakiś czas odpytuje punkt końcowy modułu `Actuator`, dzięki czemu Docker potrafi automatycznie wykryć awarię aplikacji i podjąć działania naprawcze, np. ponowne uruchomienie kontenera.

5.1.3. Środowisko produkcyjne

Środowisko produkcyjne zostało wdrożone na maszynie wirtualnej w chmurze Oracle Cloud Infrastructure z systemem Ubuntu. Ruch przychodzący z Internetu obsługiwany jest przez serwer pośredniczący Nginx. Nginx odpowiada za:

- terminację/zakończenie SSL z wykorzystaniem certyfikatów Let's Encrypt;
- rozdział ruchu między aplikacją serwerową, a serwerem MinIO;
- przekierowanie ruchu HTTP na HTTPS;
- udostępnienie ścieżki wymaganej przez protokół ACME do sprawdzania domeny przy odnawianiu certyfikatów;
- obsługę nagłówków pośredniczących przekazywanych do aplikacji.

Odnawianie certyfikatów SSL realizowane jest przez kontener Certbot uruchamiany co 12 godzin. Certyfikaty oraz dane wyzwań ACME współdzielone są między kontenerami Nginx i Certbot za pomocą wolumenów Docker.

Aplikacja internetowa udostępniana jest niezależnie za pośrednictwem platformy Vercel. Aplikacja mobilna budowana jest w chmurze przez usługę Expo Application Services (EAS) Build, której konfigurację zdefiniowano w pliku `eas.json`. Dystrybucja prototypu odbywa się przez wewnętrzny mechanizm udostępniania kompilacji w usłudze EAS.

5.1.4. Ciągła integracja z GitHub Actions

W repozytorium API skonfigurowano potok ciągłej integracji (ang. *Continuous Integration*, CI) uruchamiany automatycznie przy każdym wypchnięciu zmian do gałęzi głównej oraz przy utworzeniu żądania scalenia. Konfigurację przepływu pracy (ang. *workflow*) narzędzia GitHub Actions przedstawia Listing 13.

Listing 13. Plik konfiguracyjny dla ciągłej integracji GitHub Actions. Źródło: Opracowanie własne.

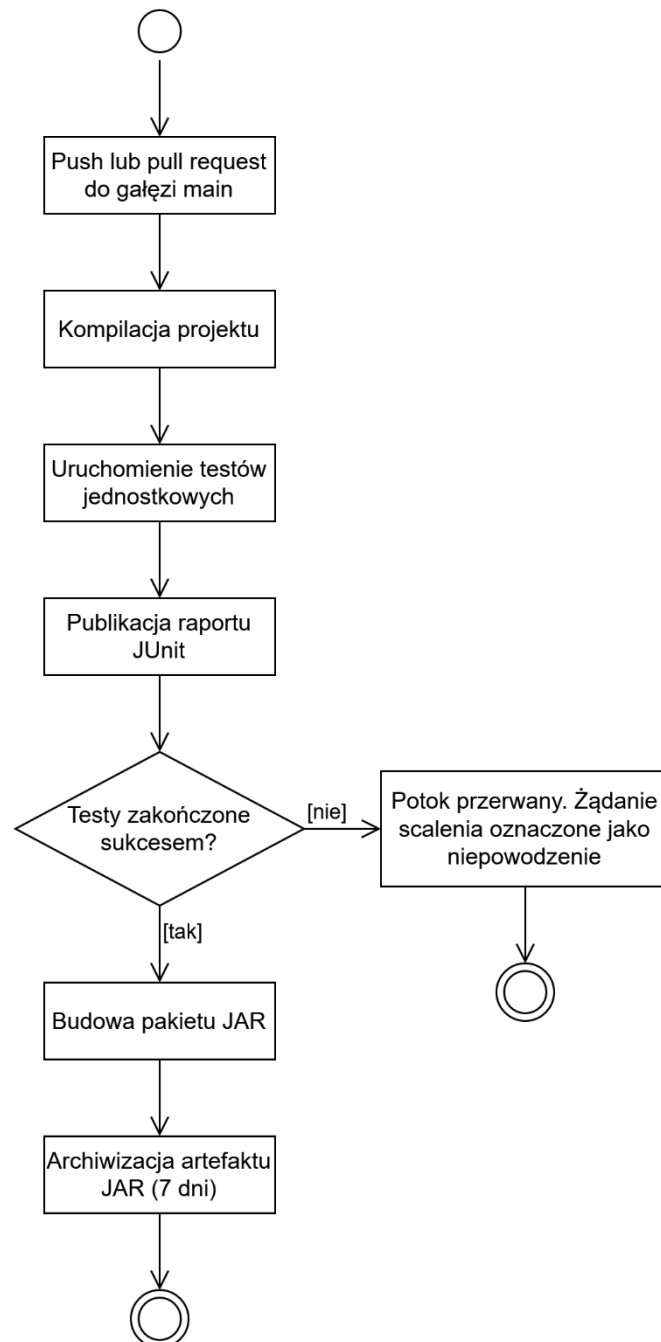
```
name: CI
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up JDK 21
        uses: actions/setup-java@v4
        with:
          java-version: 21
          distribution: 'temurin'
          cache: 'maven'
      - name: Cache Maven packages
        uses: actions/cache@v4
        with:
          path: ~/.m2/repository
          key: ${ runner.os }-maven-${ hashFiles('**/pom.xml') }
      - name: Build with Maven
        run: ./mvnw clean compile
      - name: Run tests
        run: ./mvnw test
      - name: Publish test report
        if: always()
        uses: dorny/test-reporter@v1
        with:
          path: target/surefire-reports/*.xml
          reporter: java-junit

  build:
    runs-on: ubuntu-latest
    needs: test
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-java@v4
        with:
          java-version: 21
          distribution: 'temurin'
      - name: Build package
        run: ./mvnw clean package -DskipTests
      - uses: actions/upload-artifact@v4
        with:
          name: application-jar
          path: target/*.jar
```

Potok składa się z dwóch zadań. Zadanie test kompiluje projekt, uruchamia testy jednostkowe oraz publikuje raport w formacie zgodnym z JUnit, który zostaje wyświetlony jako część widoku żądania scalenia. Zadanie build uruchamiane jest tylko po pomyślnym przejściu testów i generuje plik JAR aplikacji, który jest archiwizowany jako artefakt dostępny przez 7 dni.

Schemat działania potoku przedstawia Rysunek 33.



Rysunek 33. Diagram potoku ciągłej integracji w repozytorium API. Źródło: Opracowanie własne.

5.1.5. Dokumentacja API z Open API

Interfejs REST udostępniony przez aplikację serwerową dokumentowany jest ręcznie w pliku *openapi.yaml* zgodnym ze specyfikacją OpenAPI 3.0.3. Plik jest przechowywany w repozytorium API obok kodu źródłowego, dzięki czemu zmiany w interfejsie wersjonowane są wspólnie z implementacją.

Specyfikacja definiuje udostępnione końcówki, parametry żądań, modele danych odpowiedzi i kody błędów.

Decyzja o ręcznym utrzymaniu specyfikacji podyktowana była dwoma czynnikami. Po pierwsze, ręczna specyfikacja stanowi źródło prawdy niezależnie od kodu, co ułatwia jej współdzielenie z zewnętrznymi konsumentami interfejsu oraz dokumentowanie wymagań przed rozpoczęciem implementacji. Po drugie, automatyczne generatory dokumentacji często wytwarzają niepełne schematy dla złożonych typów danych (między innymi pól niestandardowych o dynamicznym typie wartości), co i tak wymagałoby ręcznego dopracowania końcowego dokumentu.

5.2. Model PostGIS i dane przestrzenne

W niniejszym podrozdziale skupiono się na implementacyjnym aspekcie obsługi danych przestrzennych, a w tym, sposobie odwzorowania współrzędnych geograficznych otworów badawczych na typ geometryczny w bazie danych PostgreSQL i mapowaniu między typem geometrycznym, a obiektowym modelem aplikacji, oraz mechanizmie indeksowania kolumny geometrycznej.

Schemat bazy danych zarządzany jest przez Liquibase, którego mechanizm migracji omówiono w rozdziale 4.9. Pierwsza migracja w łańcuchu *000-enable-postgis.xml* instaluje rozszerzenie PostGIS, jeśli nie zostało jeszcze zainstalowane dla danej instancji serwera. Sprawdzenie istnienia rozszerzenia w tabeli `pg_extension` zapobiega błędom przy ponownym uruchomieniu migracji na bazie.

Najistotniejszą tabelą z perspektywy obsługi danych przestrzennych jest tabela `holes`, w której kolumna `location` przechowuje położenie geograficzne otworu badawczego. Listing 14 przedstawia fragment migracji odpowiedzialnej za utworzenie tej tabeli, wraz założeniem indeksu przestrzennego GIST (ang. *Generalized Search Tree*) na kolumnie geometrycznej.

Listing 14. Fragment migracji Liquibase tworzącej tabelę otworów badawczych. Źródło: Opracowanie własne.

```
<changeSet id="007-create-hole-table" author="developer">
  <comment>Create holes table with PostGIS support</comment>
  <createSequence sequenceName="hole_id_seq"
    startValue="1" incrementBy="1"/>
  <createTable tableName="holes">
    <column name="id" type="BIGINT"
      defaultValueSequenceNext="hole_id_seq">
      <constraints primaryKey="true" primaryKeyName="pk_hole"/>
    </column>
    <column name="name" type="VARCHAR(1024)">
      <constraints nullable="false"/>
    </column>
    <column name="location" type="geometry(Point,4326)">
      <constraints nullable="true"/>
    </column>
    <column name="status" type="VARCHAR(50)">
      <constraints nullable="false"/>
    </column>
    <column name="project_id" type="BIGINT">
      <constraints nullable="false"/>
    </column>
  </createTable>
  <addForeignKeyConstraint
    baseTableName="holes" baseColumnNames="project_id"
    constraintName="fk_hole_project"
    referencedTableName="projects" referencedColumnNames="id"
    onDelete="CASCADE"/>
  <sql>CREATE INDEX idx_holes_location
    ON holes USING GIST (location);</sql>
</changeSet>
```

```

<rollback>
  <sql>DROP INDEX IF EXISTS idx_holes_location;</sql>
</rollback>
</changeSet>

```

Typ geometryczny `geometry (Point, 4326)` przechowuje współrzędne w układzie odniesienia WGS 84, powszechnie stosowanym do reprezentacji położen geograficznych w postaci par długość-szerokość geograficzna. Indeks GIST jest zbalansowaną strukturą drzewiastą umożliwiającą efektywne wyszukiwanie obiektów geometrycznych. W prototypie systemu indeks ten wspiera sprawdzenie unikalności położenia otworu w obrębie projektu, a jednocześnie stanowi przygotowanie schematu pod zapytania zakresowe wykorzystywane przy wizualizacji kartograficznej, których obsługę przewidziano w dalszym rozwoju systemu.

W warstwie aplikacyjnej mapowanie tabeli `holes` na obiekt Java realizowane jest przez encję JPA przedstawioną w Listing 15. Najistotniejszym elementem jest deklaracja pola `location` typu `Point` z biblioteki JTS, opisanego adnotacją `@Column` z parametrem `columnDefinition` ustawionym na `geometry (Point, 4326)`. Adnotacja ta instruuje Hibernate, by przy generowaniu zapytań DDL używał odpowiedniego typu PostGIS.

Listing 15. Fragment encji Hole z mapowaniem lokalizacji przestrzennej. Źródło: Opracowanie własne.

```

@Entity
@Table(name = "holes")
@Getter @Setter
@NoArgsConstructor @AllArgsConstructor
@Builder(toBuilder = true)
public class Hole {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE,
                    generator = "hole_id_seq")
    @SequenceGenerator(name = "hole_id_seq",
                      sequenceName = "hole_id_seq",
                      allocationSize = 1)
    private Long id;

    @NotBlank
    @Size(max = 1024)
    @Column(name = "name", nullable = false, length = 1024)
    private String name;

    @NotNull
    @Column(name = "location",
            columnDefinition = "geometry(Point,4326)",
            nullable = false)
    private Point location;

    @NotNull
    @Enumerated(EnumType.STRING)
    @Column(name = "status", nullable = false, length = 50)
    @Builder.Default
    private HoleStatus status = HoleStatus.UNASSIGNED;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "project_id", nullable = false)
    private Project project;

    public enum HoleStatus {
        UNASSIGNED, ASSIGNED, IN_PROGRESS, SUBMITTED, CLOSED
    }
}

```

Stan otworu reprezentowany jest przez wyliczenie `HoleStatus` przechowywane jako tekst w bazie, co zapewnia czytelność danych przy bezpośrednim podglądzie tabel oraz odporność na zmianę kolejności wartości wyliczenia.

Metoda przyjmuje obiekt klasy `Point` i sprawdza, czy w danym projekcie istnieje już otwór o identycznym położeniu. Wykorzystywana jest przy tworzeniu i modyfikacji otworu, oraz przy seryjnym imporcie otworów.

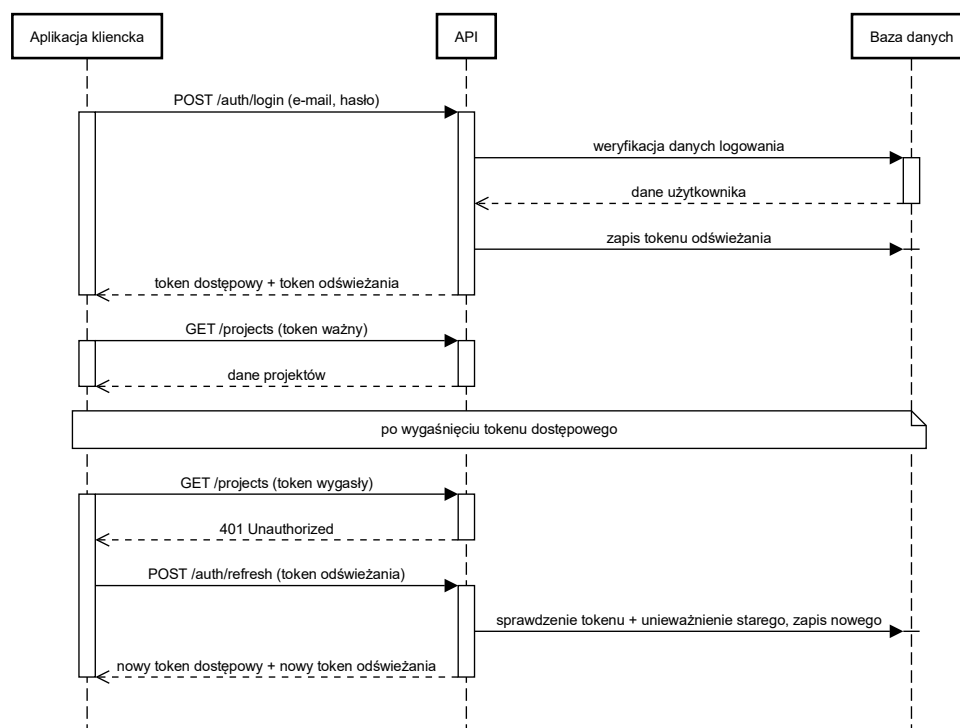
5.3. Uwierzytelnianie i zarządzanie użytkownikami

Mechanizm uwierzytelniania w systemie KRET oparty jest na dwóch uzupełniających się nawzajem elementach: krótkotrwałym tokenie dostępowym JSON Web Token (JWT) oraz długotrwałym tokenie odświeżania. Schemat dwóch tokenów pozwala utrzymać bezstanową architekturę serwera oraz zachować wygodę użytkownika, który nie loguje się przy każdej sesji.

5.3.1. Autoryzacja oparta na tokenach

Po pomyślnym uwierzytelnieniu użytkownika serwer generuje token JWT zawierający w polach: identyfikator użytkownika, datę wystawienia oraz datę wygaśnięcia. Czas życia tokenu dostępowego ustawiono na 15 minut, co stanowi kompromis między bezpieczeństwem, a liczbą wykonywanych operacji odświeżania. Krótki czas życia oznacza, że nawet w przypadku przechwycenia tokena okno potencjalnego nadużycia jest ograniczone, a wycofanie tokena nie wymaga utrzymywania po stronie serwera listy unieważnionych identyfikatorów.

Token podpisany jest algorytmem HMAC-SHA z kluczem odczytywanym ze zmiennej środowiskowej `JWT_SECRET`. Dla zachowania bezpieczeństwa zaleca się klucz o długości co najmniej 512 bitów. Konfiguracja Spring Boot zarządza wstrzykiwaniem klucza do warstwy aplikacji, dzięki czemu wartość ta nigdy nie pojawia się w kodzie źródłowym. Sekwencję żądań uwierzytelniających przedstawia Rysunek 34.



Rysunek 34. Diagram sekwencji uwierzytelniania żądań. Źródło: Opracowanie własne.

Listing 16 przedstawia kluczowy fragment metody login z klasy AuthService obsługujący logikę uwierzytelniania użytkownika i wystawiania pary tokenów. Faktyczna weryfikacja hasła oraz pobranie danych użytkownika delegowane są do mechanizmu Spring Security, który porównuje dostarczone hasło z wartością w bazie zaszyfowaną algorytmem BCrypt.

Listing 16. Fragment metody login w klasie AuthService. Źródło: Opracowanie własne.

```
@Transactional
public LoginResponse login(LoginRequest loginRequest) {
    try {
        Authentication authentication = authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(
                loginRequest.getEmail(),
                loginRequest.getPassword()));

        UserDetails userDetails =
            (UserDetails) authentication.getPrincipal();
        String token = jwtService.generateToken(userDetails);

        User user = userRepository.findByEmail(loginRequest.getEmail())
            .orElseThrow(() -> new UserNotFoundException(
                "User not found with email: " + loginRequest.getEmail()));

        RefreshToken refreshToken =
            refreshTokenService.createRefreshToken(user);

        log.info("User logged in successfully: {}", user.getEmail());
        return buildLoginResponse(user, token, refreshToken.getToken());
    } catch (BadCredentialsException e) {
        log.warn("Failed login attempt for email: {}",
            loginRequest.getEmail());
        throw new InvalidCredentialsException(
            "Nieprawidłowy adres e-mail lub hasło");
    } catch (AuthenticationException e) {
        log.warn("Authentication failed for email: {}",
            loginRequest.getEmail());
        throw new InvalidCredentialsException(
            "Nieprawidłowy adres e-mail lub hasło");
    }
}
```

Dzięki adnotacji @Transactional operacje bazodanowe wykonywane są w pojedynczej transakcji. Wyjątki uwierzytelnienia pozostają przechwycone, a wiadomość błędu ujednolicona w celu ukrycia szczegółów powodu odrzucenia dostarczonych danych. Ujednolicony komunikat utrudnia ataki polegające na wyliczaniu istniejących kont użytkowników.

Sprawdzenie każdego kolejnego żądania w sesji odbywa się w filtrze JwtTokenFilter zarejestrowanym w łańcuchu filtrów Spring Security co przedstawia Listing 17.

Listing 17. Konfiguracja łańcucha filtrów Spring Security. Źródło: Opracowanie własne.

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http)
    throws Exception {
    http
        .cors(cors -> cors.configurationSource(corsConfigurationSource))
        .csrf(csrf -> csrf.disable())
        .sessionManagement(session -> session
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .authorizeHttpRequests(authz -> authz
            .requestMatchers(
                "/auth/login",
                "/auth/refresh",
```

```

        "/auth/logout").permitAll()
    .requestMatchers(
        "/actuator/health",
        "/actuator/health/**",
        "/actuator/info").permitAll()
    .requestMatchers("/actuator/**").authenticated()
    .anyRequest().authenticated()
    .addFilterBefore(jwtTokenFilter,
        UsernamePasswordAuthenticationFilter.class);
    return http.build();
}

```

Konfiguracja wyłącza ochronę CSRF (zbędna przy uwierzytelnianiu bezstanowym), włącza obsługę CORS dla zdefiniowanych domen klienckich, ustawia politykę zarządzania sesjami na bezstanową oraz określa publiczne punkty końcowe takie jak logowanie, odświeżanie tokenu, wylogowanie oraz punkty kontroli stanu aplikacji dostarczone przez moduł Actuator. Filtr `JwtTokenFilter` rejestrowany jest przed `UsernamePasswordAuthenticationFilter`, dzięki czemu w przypadku obecności tokenu w nagłówku żądania kontekst bezpieczeństwa zostaje ustawiony przed jakimkolwiek mechanizmem uwierzytelniania z poziomu żądania od formularza.

Implementację `JwtTokenFilter` przedstawia Listing 18.

Listing 18. Filtr `JwtTokenFilter` sprawdzający tokeny dostępne. Źródło: Opracowanie własne.

```

@Component
@Slf4j
@RequiredArgsConstructor
public class JwtTokenFilter extends OncePerRequestFilter {

    private final JwtService jwtService;
    private final UserDetailsServiceImpl userDetailsService;

    @Override
    protected void doFilterInternal(
        @NonNull HttpServletRequest request,
        @NonNull HttpServletResponse response,
        @NonNull FilterChain filterChain)
        throws ServletException, IOException {
        try {
            String authHeader = request.getHeader("Authorization");
            String token = null;
            String username = null;

            if (authHeader != null && authHeader.startsWith("Bearer ")) {
                token = authHeader.substring(7);
                username = jwtService.extractUsername(token);
            }

            if (username != null
                && SecurityContextHolder.getContext()
                    .getAuthentication() == null) {
                if (jwtService.validateToken(token)) {
                    UserDetails userDetails = userDetailsService
                        .loadUserByUsername(username);
                    UsernamePasswordAuthenticationToken authToken =
                        new UsernamePasswordAuthenticationToken(
                            userDetails, null,
                            userDetails.getAuthorities());
                    authToken.setDetails(
                        new WebAuthenticationDetailsSource()
                            .buildDetails(request));
                    SecurityContextHolder.getContext()
                        .setAuthentication(authToken);
                }
            }
        }
    }
}

```

```

    }
    } catch (Exception e) {
        log.error("Error in JwtTokenFilter: {}", e.getMessage());
    }

    filterChain.doFilter(request, response);
}
}

```

Filtr rozszerza klasę `OncePerRequestFilter`, która gwarantuje pojedyncze wykonanie filtra dla każdego żądania. Algorytm filtrowania ma trzy etapy: odczyt tokenu z nagłówka `Authorization`, weryfikację podpisu i czasu ważności oraz ustawienie kontekstu uwierzytelnienia w obiekcie `SecurityContextHolder`. Po tym kontekst jest dostępny dla każdej z warstw aplikacji, co umożliwia pobranie aktualnego użytkownika przez wstrzyknięcie zależności lub serwis pomocniczy `SecurityContextService`. Wyjątki rzucone w trakcie sprawdzania tokena są łapane i logowane bez blokowania dalszego przepływu żądania. W przypadku nieprawidłowego tokenu mechanizm autoryzacji Spring Security odrzuca żądanie zwracając kod HTTP 401.

5.3.2. Odświeżanie sesji

Token odświeżania, w odróżnieniu od tokena dostępowego, jest identyfikatorem UUID przechowywanym w tabeli `refresh_tokens` wraz z odniesieniem do użytkownika, datą wygaśnięcia oraz flagą unieważnienia. Czas życia tokenu odświeżania wynosi 7 dni. W przeciwieństwie do tokenu dostępowego, który jest samowystarczającym tokenem JWT weryfikowanym wyłącznie na podstawie podpisu, token odświeżania jest wartością nieprzejrystą (ang. *opaque token*), której ważność za każdym razem sprawdzana jest w bazie danych. Listing 19 przedstawia kluczowe operacje na tokenach odświeżania zaimplementowane w klasie `RefreshTokenService`.

Listing 19. Operacje na tokenach odświeżania w klasie `RefreshTokenService`. Źródło: Opracowanie własne.

```

@Transactional
public RefreshToken createRefreshToken(User user) {
    RefreshToken refreshToken = RefreshToken.builder()
        .token(UUID.randomUUID().toString())
        .user(user)
        .expiresAt(Instant.now().plusMillis(refreshExpiration))
        .build();
    return refreshTokenRepository.save(refreshToken);
}

@Transactional(readOnly = true)
public RefreshToken validateRefreshToken(String token) {
    RefreshToken refreshToken = refreshTokenRepository.findByToken(token)
        .orElseThrow(() -> new InvalidRefreshTokenException(
            "Invalid refresh token"));

    if (refreshToken.isRevoked()) {
        throw new InvalidRefreshTokenException(
            "Token odświeżania został unieważniony");
    }
    if (refreshToken.getExpiresAt().isBefore(Instant.now())) {
        throw new InvalidRefreshTokenException(
            "Token odświeżania wygasł");
    }
    return refreshToken;
}

@Transactional
public void revokeRefreshToken(RefreshToken refreshToken) {
    refreshToken.setRevoked(true);
}

```

```
refreshTokenRepository.save(refreshToken);
}
```

W systemie KRET zastosowano rotację tokenów odświeżania. Oznacza to, że przy każdym pomyślnym odświeżeniu sesji poprzedni token zostaje unieważniony, a wygenerowany zostaje nowy. Mechanizm ten ogranicza skutki ewentualnego przechwycenia tokenu, ponieważ jego ponowne użycie po wykonanej rotacji powoduje odrzucenie żądania. Logika rotacji znajduje się w metodzie `refreshToken` klasy `AuthService` i polega na sekwencyjnym wywołaniu `validateRefreshToken`, `revokeRefreshToken` oraz `createRefreshToken`. Unieważnienie tokenu odświeżania analogicznie wykorzystuje metoda `logout` w `AuthService`, która unieważnia token odświeżania, co uniemożliwia dalsze przedłużanie sesji.

Po stronie aplikacji klienckich obsługa odświeżania zaimplementowana jest w bibliotece `@geokret/shared` jako mechanizm przechwytyjący odpowiedzi biblioteki Axios. Mechanizm przechwytyjący reaguje na kod odpowiedzi 401. Jeśli dane żądanie z odpowiedzią 401 nie było zapytaniem do końcówki uwierzytelniania, mechanizm przechwytyjący próbuje odświeżyć token i ponowić oryginalne żądanie z nowym tokenem dostępowym. Mechanizm ten szczegółowo omówiono w podrozdziale 5.16.

5.3.3. Podział ról i uprawnień

W systemie KRET wyróżniono trzy role użytkowników, zdefiniowane jako wyliczenie `UserRole` w encji `User`: `ADMIN`, `DOCUMENTER` oraz `GEOLOGIST`. Tabela 22 przedstawia podstawowe uprawnienia każdej z ról.

Tabela 22. Tabela uprawnień ról użytkowników w systemie KRET. Źródło: Opracowanie własne.

Operacja	ADMIN	DOCUMENTER	GEOLOGIST
Tworzenie i edycja projektów	tak	tak	nie
Tworzenie otworów	tak	tak	Nie
Przypisywanie geologów do otworów	tak	tak	nie
Wypełnianie formularzy badań	tak	tak	tylko przypisanych
Tworzenie dyskusji	tak	tak	tylko odpowiedzi
Weryfikacja i zamykanie otworów	tak	tak	nie
Zaproszenia administratorów	tak	nie	nie
Zarządzanie polami niestandardowymi i wzorcami	tak	tak	nie
Wgląd w statystyki firmy	tak	tak	tylko własne

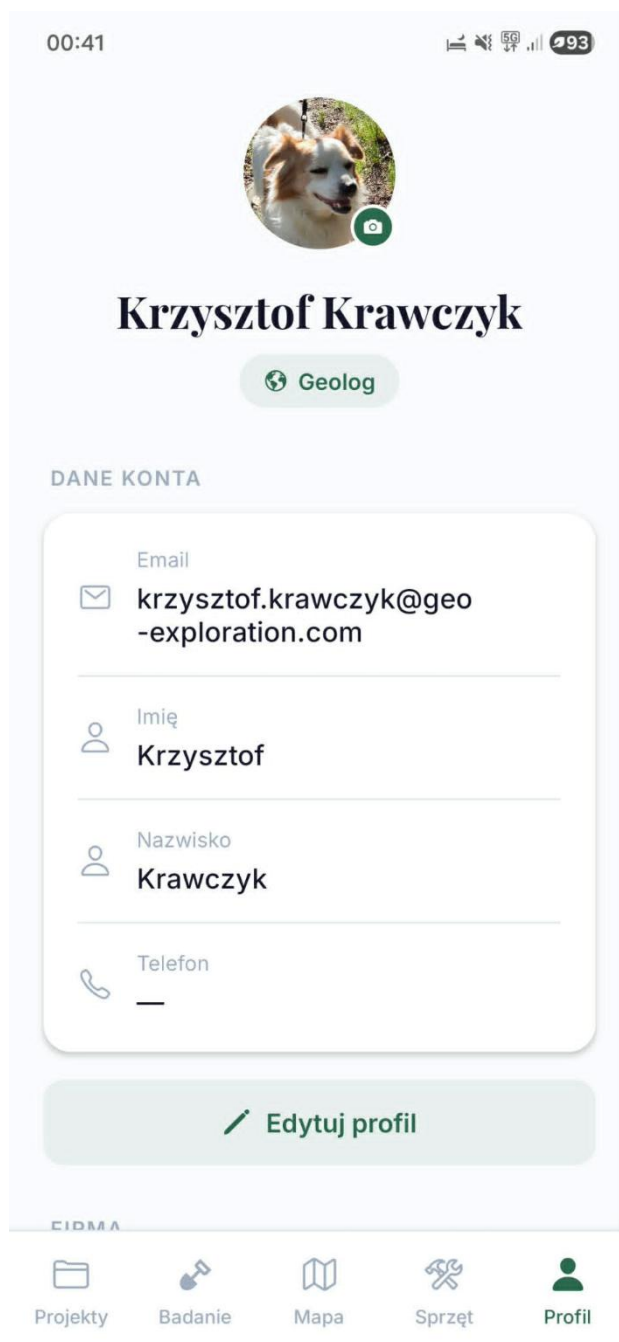
Egzekwowanie uprawnień zrealizowano w odrębnej klasie `AccessControlService`, której metody (`requireAdmin`, `requireAdminOrDocumenter`, `requireProjectAccessByCompany`, `requireProjectAssignment` i inne) wywoływane są przez serwisy biznesowe przed wykonaniem operacji wrażliwych. Każda z metod rzuca wyjątek `AccessDeniedException`, gdy rola lub powiązanie użytkownika nie spełnia wymagań danej operacji.

5.3.4. Profile użytkowników i awatary

Każdy użytkownik posiada profil zawierający imię, nazwisko, adres e-mail, numer telefonu oraz odniesienie do firmy, do której należy. Avatar przechowywany jest jako odniesienie do obiektu `StoredFile` zarządzanego przez serwis przechowywania plików omówiony w podrozdziale 5.4. Edycja danych profilu oraz awatara jest możliwa z poziomu aplikacji internetowej oraz mobilnej.

Adres e-mail jest unikatowy w obrębie systemu i służy jako identyfikator logowania oraz nie może być zmieniony po rejestracji. Hasło ustawiane jest w trakcie rejestracji przez zaproszenie i przechowywane wyłącznie w postaci skrótu wygenerowanego algorytmem BCrypt.

Profil użytkownika w aplikacji mobilnej przedstawia Rysunek 35.



Rysunek 35. Ekran profilu użytkownika w aplikacji mobilnej. Źródło: Opracowanie własne.

5.3.5. System zaproszeń i rejestracji

Rejestracja w systemie KRET odbywa się wyłącznie przez mechanizm zaproszeń. Otwarta rejestracja jest niedozwolona ze względu na model biznesowy systemu. Aplikacja jest przeznaczona dla pracowników konkretnych firm geologicznych, a nie otwartej społeczności. Administrator lub dokumentator wysyła zaproszenie na adres e-mail wskazanej osoby wraz z określeniem docelowej roli. Moduł ten prezentuje Rysunek 36. System generuje unikatowy token UUID dla każdego zaproszenia, zapisuje go w tabeli `invitations` z 48-godzinny terminem ważności i wysyła wiadomość zawierającą link aktywacyjny.

Rysunek 36. Ekran wysyłania zaproszenia w aplikacji internetowej. Źródło: Opracowanie własne.

Listing 20 przedstawia kluczowy fragment metody `createAndSendInvitation` z klasy `InvitationService`.

Listing 20. Tworzenie i wysyłka zaproszenia. Źródło: Opracowanie własne.

```
@Transactional(propagation = Propagation.REQUIRES_NEW)
public Invitation createAndSendInvitation(InvitationRequest request,
                                         Optional<User> inviterOptional) {
    String recipientEmail = request.getRecipientEmail();
    User.UserRole targetRole = request.getTargetRole();

    if (userRepository.findByEmail(recipientEmail).isPresent()) {
        throw new IllegalArgumentException(
            "Użytkownik z adresem e-mail '" + recipientEmail +
            "' już istnieje w systemie.");
    }
    if (invitationRepository.existsByEmailAndUsedAtIsNullAndExpiresAtAfter(
        recipientEmail, LocalDateTime.now())) {
        throw new IllegalArgumentException(
            "Istnieje już aktywne zaproszenie dla adresu e-mail: '" +
            recipientEmail + "'");
    }

    User inviter = inviterOptional.orElse(null);
    if (inviter != null && inviter.getRole() != User.UserRole.ADMIN
        && targetRole == User.UserRole.ADMIN) {
        throw new IllegalArgumentException(
            "Tylko administratorzy mogą zapraszać innych administratorów.");
    }
}
```

```

    }

    String token = UUID.randomUUID().toString();
    LocalDateTime expiresAt =
        LocalDateTime.now().plusHours(INVITATION_EXPIRY_HOURS);

    Invitation invitation = Invitation.builder()
        .email(recipientEmail)
        .token(token)
        .issuedAt(LocalDateTime.now())
        .expiresAt(expiresAt)
        .invitedBy(inviter)
        .intendedRole(targetRole)
        .build();

    invitationRepository.save(invitation);

    String registrationLink = frontendUrl + "/register?token=" + token;
    String inviterName = inviterOptional.map(User::getFirstName)
        .orElse(DEFAULT_INVITER_NAME);
    emailService.sendInvitationEmail(
        recipientEmail, inviterName, registrationLink, request.getMessage());

    return invitation;
}

```

Adnotacja obejmująca `@Transactional(propagation = Propagation.REQUIRES_NEW)` zapewnia uruchomienie metody w odrębnej transakcji niezależnej od transakcji wywołującej, co gwarantuje utrwalenie zaproszenia w przypadku, gdy w przepływie nadrzędnym wystąpi błąd.

Wysyłka wiadomości realizowana jest przez klasę `EmailService` wykorzystującą szablon `Thymeleaf`, w którym zmienne `inviterName`, `registrationLink` podstawiane są z kontekstu wywołania. Treść wiadomości zawiera powitanie z nazwą zapraszającego oraz przyciskiem kierującym do strony rejestracji.

Po kliknięciu w link aktywacyjny użytkownik trafia na stronę rejestracji, która wstępnie weryfikuje token zaproszenia. Po wprowadzeniu danych osobowych oraz hasła konto zostaje utworzone, przypisane do odpowiedniej firmy i roli zdefiniowanej w zaproszeniu, a token zaproszenia zostaje oznaczony jako wykorzystany.

Zaproszenia, które wygasły bez użycia, mogą zostać ponownie wysłane przez metodę `resendInvitation`. Operacja ta wymaga, aby poprzednie zaproszenie nie zostało wykorzystane oraz aby już wygasło, co nie pozwala na nieuzasadnione wielokrotne wysyłanie wiadomości do tej samej osoby w krótkim odstępie czasu.

5.4. Przechowywanie i udostępnianie plików

System KRET przechowuje znaczną liczbę plików: zdjęcia warstw geologicznych, logotypy firm, awatary użytkowników, załączniki artykułów bazy wiedzy oraz pliki powiązane z polami niestandardowymi typu plikowego. Wszystkie te pliki przechowywane są w serwerze pamięci obiektowej `MinIO` opisanym w podrozdziale 4.8, natomiast metadane pliku zapisywane są w tabeli `stored_files` w bazie `PostgreSQL`. Encje w aplikacji odwołują się do plików przez relacje `StoredFile`, dzięki czemu warstwa biznesowa pracuje z typowanymi obiektami zamiast z surowymi identyfikatorami obiektów `MinIO`.

`MinIO` posiada mechanizm uwierzytelniania opierający się na podpisach żądań zawierających nazwę hosta serwera. Powoduje to istotny problem techniczny w środowisku, w którym serwer

aplikacyjny komunikuje się z MinIO przez wewnętrzną sieć Docker (host *minio:9000*), natomiast aplikacje klienckie przez publiczny adres (*files.geokret.eu*). Adres URL z podpisem wygenerowany przez wewnętrznego klienta MinIO byłby ważny tylko dla hosta *minio:9000* i nie zadziałałby z poziomu przeglądarki użytkownika, Rozwiązaniem zastosowanym w systemie KRET jest skonfigurowanie dwóch oddzielnych instancji klienta *MinioClient* – wewnętrznego (zapis i odczyt przez serwer aplikacji) oraz publicznego (generowanie podpisanych adresów URL). Konfigurację obu klientów przedstawia Listing 21.

Listing 21. Konfiguracja dwóch klientów MinIO. Źródło: Opracowanie własne.

```
@Configuration
@Slf4j
public class MinioConfig {

    private final MinioProperties minioProperties;

    public MinioConfig(MinioProperties minioProperties) {
        this.minioProperties = minioProperties;
    }

    @Bean
    public MinioClient minioClient() {
        return MinioClient.builder()
            .endpoint(minioProperties.getEndpoint())
            .credentials(minioProperties.getAccessKey(),
                minioProperties.getSecretKey())
            .build();
    }

    @Bean
    public MinioClient minioClientForPresignedUrls() {
        String endpoint = minioProperties.getPublicEndpoint();
        if (endpoint == null || endpoint.isBlank()) {
            endpoint = minioProperties.getEndpoint();
            log.warn("MINIO_PUBLIC_ENDPOINT not set, " +
                "using internal endpoint for presigned URLs");
        }
        return MinioClient.builder()
            .endpoint(endpoint)
            .credentials(minioProperties.getAccessKey(),
                minioProperties.getSecretKey())
            .build();
    }
}
```

Metoda *minioClient* z adnotacją *@Bean* buduje klienta http, który wykorzystuje adres wewnętrzny i obsługuje wszystkie operacje zapisu oraz odczytu wykonane przez serwer aplikacji. Za to *minioClientForPresignedUrls* konfigurowany jest adresem publicznym i służy wyłącznie do generowania wstępnie podpisanych adresów URL (ang. *presigned URL*), które są eksponowane do aplikacji klienckich.

Wgrywanie pliku jest realizowane przez metodę *uploadFile* klasy *StorageService* przedstawioną na Listing 22. Metoda przesyła zawartość pliku do MinIO, a następnie zapisuje jego metadane w bazie danych.

Listing 22. Metoda wgrywania pliku do MinIO. Źródło: Opracowanie własne.

```
public StoredFile uploadFile(MultipartFile file, String directory) {
    String originalFilename = file.getOriginalFilename();
    String objectKey = generateObjectKey(directory, originalFilename);

    try (InputStream inputStream = file.getInputStream()) {
```

```

minioClient.putObject(
    PutObjectArgs.builder()
        .bucket(minioProperties.getBucket())
        .object(objectKey)
        .stream(inputStream, file.getSize(), -1)
        .contentType(file.getContentType())
        .build());

User currentUser = securityContextService.getCurrentUser();

StoredFile storedFile = StoredFile.builder()
    .originalFilename(originalFilename)
    .objectKey(objectKey)
    .bucketName(minioProperties.getBucket())
    .contentType(file.getContentType())
    .fileSize(file.getSize())
    .uploadedBy(currentUser)
    .build();

storedFile = storedFileRepository.save(storedFile);
log.info("Uploaded file: {} -> {}", originalFilename, objectKey);
return storedFile;
} catch (Exception e) {
    log.error("Failed to upload file: {}", originalFilename, e);
    throw new StorageException(
        "Nie udało się przesłać pliku: " + originalFilename, e);
}
}

```

Klucz obiektu generowany jest jako kombinacja danych katalogu (przykładowo *Projects/12/holes/45/boreholes/8/layers/3*), identyfikatora UUID oraz oryginalnej nazwy pliku, co zapobiega kolizjom przy wgrywaniu wielu plików o tej samej nazwie oraz pozwala na łatwe filtrowanie zasobów. Maksymalny rozmiar przesyłanego pliku wynosi 15 MB, a dozwolone formaty obrazów to JPEG, PNG, GIF oraz WebP.

Udostępnianie plików klientom realizowane jest za pomocą mechanizmu wstępnie podpisanych adresów URL. Listing 23 przedstawia metodę generującą taki adres dla pliku przechowywanego w MinIO.

Listing 23. Generowanie wstępnie podpisanego adresu URL. Źródło: Opracowanie własne.

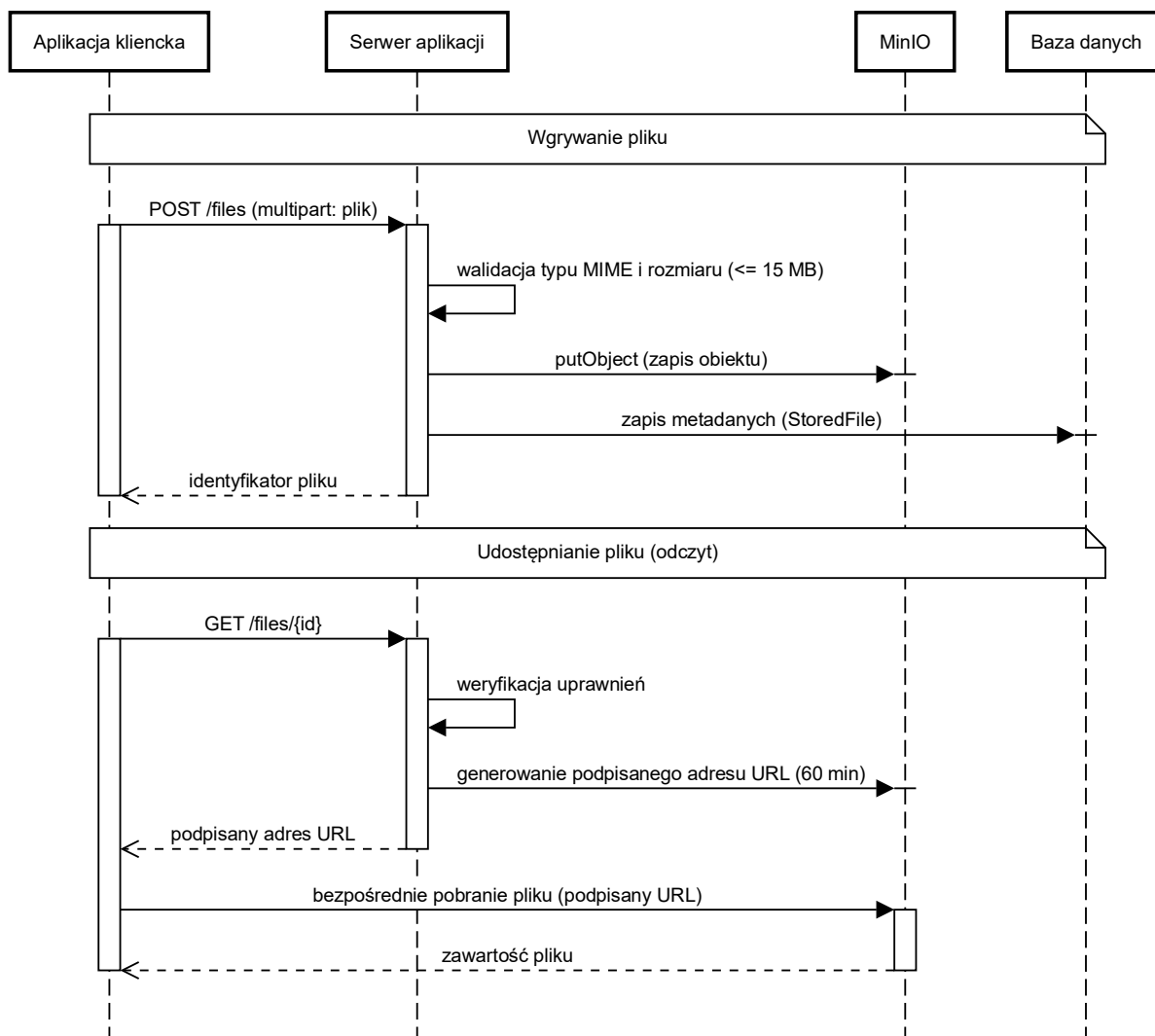
```

public String getPresignedUrl(StoredFile storedFile, int expiryMinutes) {
    try {
        return minioClientForPresignedUrls.getPresignedObjectUrl(
            GetPresignedObjectUrlArgs.builder()
                .bucket(storedFile.getBucketName())
                .object(storedFile.getObjectKey())
                .method(Method.GET)
                .expiry(expiryMinutes, TimeUnit.MINUTES)
                .build());
    } catch (Exception e) {
        log.error("Failed to generate presigned URL for: {}",
            storedFile.getObjectKey(), e);
        throw new StorageException(
            "Nie udało się wygenerować adresu URL pobierania", e);
    }
}

public String getPresignedUrl(StoredFile storedFile) {
    return getPresignedUrl(storedFile, 60);
}

```

Domyślny czas ważności podpisanego adresu wynosi 60 minut. Limit jest dopasowany dla wygody użytkowników i ograniczeń systemu. Pozwala na wygodne przeglądanie zdjęć, a jednocześnie ogranicza skutki ewentualnego ujawnienia adresu. Schemat całego przepływu przedstawia Rysunek 37.



Rysunek 37. Diagram sekwencji wgrywania i udostępniania pliku. Źródło: Opracowanie własne.

Udostępnianie plików do odczytu oparte jest na bezpośrednim transferze między aplikacją kliencką a MinIO, co oznacza, że serwer aplikacji nie pośredniczy w przesyłaniu danych binarnych przy pobieraniu, a jedynie generuje podpisany adres URL i weryfikuje uprawnienia. Wgrywanie pliku odbywa się natomiast przez serwer aplikacji, który odbiera plik, sprawdza jego typ i rozmiar, zapisuje go w MinIO oraz utrzuwa metadane w bazie. Takie rozdzielenie pozwala odciążyc serwer przy najczestszej operacji zachowujac jednocześnie kontrolę nad sprawdzaniem plików w momencie ich wgrywania.

5.5. Zarządzanie projektami i otworami

Moduły zarządzania projektami i otworami stanowią rdzeń funkcjonalny systemu KRET. Podrozdział opisuje mechanizmy obsługi badań terenowych – od utworzenia projektu, przez przypisanie wykonawców, po seryjny import danych.

5.5.1. Cykl życia projektu

Projekt stanowi encję nadrzędną dla większości danych w systemie – jest powiązany asocjacjami jeden-do-wielu z otworami, wzorcami formularzy, polami niestandardowymi oraz rezerwacjami sprzętu. Odwrotna zależność dotyczy jedynie firmy: jedna firma może prowadzić wiele projektów, lecz każdy projekt należy do dokładnie jednej firmy.

Cykl życia projektu odzwierciedla wewnętrzny typ wyliczeniowy `ProjectStatus`, obejmujący stany: przygotowanie → w trakcie → zakończony. Utworzenie projektu oraz zmiana jego stanu wymagają roli administratora lub dokumentatora. O każdym przejściu między stanami zainteresowani użytkownicy informowani są automatycznie za pośrednictwem `NotificationService`.

Operacje na projektach realizuje serwis `ProjectService`. Obok standardowego zapisu, odczytu i usunięcia serwis udostępnia metody biznesowe `assignUsersToProject` i `removeUsersFromProject`, a zmiany statusu odbywają się w ramach metody `updateProject`. Metody modyfikujące opatrzone są adnotacją `@Transactional` i korzystają z serwisów `SecurityContextService` oraz `AccessControlService` w celu weryfikacji tożsamości użytkownika i jego praw do modyfikacji projektu.

5.5.2. Zarządzanie otworami

Otwór jest podstawową jednostką organizacyjną badań w systemie KRET. Każde badanie, niezależnie od tego, czy jest wierceniem, czy sondowaniem, rozpoczyna się od utworzenia instancji tej encji. Otwór przechowuje metadane punktu badawczego: nazwę, współrzędne, dane administracyjne, przypisanych geologów i dokumentatora oraz aktualny status.

Cykl życia otworu definiuje wewnętrzny typ wyliczeniowy `HoleStatus`, obejmujący stany: nieprzypisany → przypisany → w trakcie → do sprawdzenia → zamknięty. Przejściami między stanami zarządza serwis `HoleService`, którego metody modyfikujące opatrzone są adnotacją `@Transactional` w celu zachowania spójności danych. Obok podstawowych operacji na danych (ang. CRUD – *create, read, update, delete*) serwis udostępnia metody biznesowe, m.in. `assignUsersToHole`, `removeUsersFromHole`, `submitHole`, `closeHole`, `rejectHole` oraz `manuallyAssignDocumenter`. Wspólny schemat ich działania obejmuje:

- sprawdzenie poprawności danych wejściowych;
- egzekwowanie uprawnień użytkownika przez `SecurityContextService` i `AccessControlService`;
- kontrolę przejść między stanami cyklu życia otworu.

Sposób realizacji tego schematu przedstawia Listing 24, zawierający metodę `assignUsersToHole`, która odpowiada również za pierwsze przejście otworu ze stanu nieprzypisanego do przypisanego.

Listing 24. Metoda `assignUsersToHole` serwisu `HoleService` odpowiedzialna za przypisywanie geologów do otworu. Źródło: Opracowanie własne.

```
@Transactional
public HoleResponse assignUsersToHole(Long projectId, Long holeId,
AssignUsersRequest request) {
```

```

User currentUser = securityContextService.getCurrentUser();
accessControlService.requireAdminOrDocumenter(currentUser);

Project project = findProjectById(projectId);
accessControlService.requireProjectAccessByCompany(currentUser, project);

Hole hole = findHoleById(holeId);
validateHoleBelongsToProject(hole, projectId);

log.info("Assigning {} users to hole ID: {} in project ID: {} by user: {}",
        request.getUserIds().size(), holeId, projectId,
        currentUser.getEmail());

List<User> usersToAssign = userRepository.findAllById(request.getUserIds());

if (usersToAssign.size() != request.getUserIds().size()) {
    throw new IllegalArgumentException("Nie znaleziono jednego lub więcej
użytkowników");
}

boolean wasEmpty = hole.getAssignedGeologists().isEmpty();
List<User> newlyAssignedUsers = new ArrayList<>();

for (User user : usersToAssign) {
    accessControlService.validateGeologistForProject(user, project);

    if (!hole.getAssignedGeologists().contains(user)) {
        hole.getAssignedGeologists().add(user);
        newlyAssignedUsers.add(user);
        log.debug("Added user {} to hole {}", user.getEmail(), holeId);
    } else {
        log.debug("User {} already assigned to hole {}", user.getEmail(),
holeId);
    }
}

if (wasEmpty && !hole.getAssignedGeologists().isEmpty()) {
    hole.markAsAssigned();
    if (hole.getStatus() == Hole.HoleStatus.UNASSIGNED) {
        hole.setStatus(Hole.HoleStatus.ASSIGNED);
    }
}

Hole updatedHole = holeRepository.save(hole);
log.info("Users assigned successfully to hole: {}", holeId);

if (!newlyAssignedUsers.isEmpty()) {
    notificationService.notifyHoleAssigned(newlyAssignedUsers, updatedHole,
currentUser);
}

return mapToHoleResponse(updatedHole);
}

```

5.5.3. Przypisywanie geologów do projektów i otworów

Przypisywanie pracowników odbywa się na dwóch płaszczyznach: projektowej, decydującej o dostępie użytkownika do zasobów danego projektu oraz wewnątrz projektowej, oznaczającej zgodę na prowadzenie badań w konkretnym otworze. Za obie odpowiadają serwisy scharakteryzowane w podrozdziałach 5.5.1 i 5.5.2.

W `ProjectService` kluczową rolę pełnią metody `assignUsersToProject` oraz `removeUsersFromProject`. Ich logika jest analogiczna, przy czym metoda przypisująca zawiera dodatkową weryfikację kandydatów – jej przebieg przedstawia Listing 25.

Listing 25. Metoda `assignUsersToProject` z `ProjectService`. Źródło: Opracowanie własne.

```
@Transactional
public ProjectResponse assignUsersToProject(Long projectId, AssignUsersRequest
request) {
    User currentUser = securityContextService.getCurrentUser();
    accessControlService.requireAdminOrDocumenter(currentUser);

    Project project = findProjectById(projectId);
    accessControlService.requireProjectAccessByCompany(currentUser, project);

    log.info("Assigning {} users to project ID: {} by user: {}",
        request.getUserIds().size(), projectId, currentUser.getEmail());

    List<User> usersToAssign = getUsersForProject(request.getUserIds(),
currentUser.getCompany());

    List<User> newlyAssignedUsers = new ArrayList<>();

    for (User user : usersToAssign) {
        if (!project.getUsers().contains(user)) {
            project.getUsers().add(user);
            newlyAssignedUsers.add(user);
            log.debug("Added user {} to project {}", user.getEmail(), projectId);
        } else {
            log.debug("User {} already assigned to project {}", user.getEmail(),
projectId);
        }
    }

    Project updatedProject = projectRepository.save(project);
    log.info("Users assigned successfully to project: {}", projectId);

    if (!newlyAssignedUsers.isEmpty()) {
        notificationService.notifyProjectAssigned(newlyAssignedUsers, project,
currentUser);
    }

    return mapToProjectResponse(updatedProject);
}
```

Przypisania na poziomie otworu obejmują dwa typy użytkowników: geologów, czyli wykonawców badań (może być ich wielu), oraz dokumentatora odpowiedzialnego za weryfikację wyników (co najwyżej jeden). Przypisywaniem geologów zajmują się metody `assignUsersToHole` (Listing 24) i `removeUsersFromHole`; w odróżnieniu od wariantu projektowego metoda przypisująca dodatkowo weryfikuje przynależność każdego kandydata do projektu (`validateGeologistForProject`) oraz aktualizuje status otworu przy pierwszym przypisaniu.

Dokumentator przydzielany jest automatycznie w chwili utworzenia otworu przez prywatną metodę `assignDocumenterRoundRobin`. Administrator może jednak zmienić ten przydział ręcznie – służy do tego metoda `manuallyAssignDocumenter` (Listing 26), wystawiona w `HoleController` pod punktem końcowym `/projects/{projectId}/holes/{holeId}/assign-documenter`.

Listing 26. Metoda `manuallyAssignDocumenter` odpowiedzialna za przypisywanie dokumentatorów do otworu w celu ich rewizji. Źródło: Opracowanie własne.

```
@Transactional
```

```

public HoleResponse manuallyAssignDocumenter(Long projectId, Long holeId, Long
documenterId) {
    User currentUser = securityContextService.getCurrentUser();
    accessControlService.requireAdmin(currentUser);

    Project project = findProjectById(projectId);
    accessControlService.requireProjectAccessByCompany(currentUser, project);

    Hole hole = findHoleById(holeId);
    validateHoleBelongsToProject(hole, projectId);

    User documenter = userRepository.findById(documenterId)
        .orElseThrow(() -> new UserNotFoundException("Nie znaleziono
dokumentatora o ID: " + documenterId));

    if (!documenter.isEnabled()) {
        throw new IllegalArgumentException("Konto dokumenta jest wyłączone");
    }
    if (documenter.getRole() != User.UserRole.DOCUMENTER) {
        throw new IllegalArgumentException("Użytkownik nie ma roli DOCUMENTER");
    }
    if (!documenter.getCompany().getId().equals(project.getCompany().getId())) {
        throw new IllegalArgumentException("Dokumentator nie należy do firmy
projektu");
    }

    log.info("Admin {} manually assigning documenter {} to hole {}",
currentUser.getEmail(), documenterId, holeId);

    hole.setAssignedDocumenter(documenter);
    Hole updatedHole = holeRepository.save(hole);

    notificationService.notifyHoleAssigned(List.of(documenter), updatedHole,
currentUser);

    return mapToHoleResponse(updatedHole);
}

```

5.5.4. Import otworów

Seryjne wprowadzanie danych do systemu umożliwiają moduły importu. Funkcjonalność obejmuje dwa scenariusze: import listy encji (ang. *batch import*) oraz import danych pojedynczego badania do wskazanego otworu.

Import listy otworów obsługuje punkt końcowy `/import/{projectId}/holes` w `ImportController`, a logikę importu realizuje metoda `importHoles` serwisu `ImportService` (Listing 27). Importem wielu wierć i sondowań zajmują się metody `importBoreholes` i `importSoundings` tego samego serwisu. Każda z nich korzysta z metod udostępnionych przez `HoleService`, `BoreholeService` oraz `SoundingService`. Przykładem metody tworzącej wiele encji jednocześnie jest `createOrReplaceMultipleBoreholes`, wykorzystywana podczas importu listy wierć (Listing 28).

Przygotowanie danych odbywa się w aplikacji internetowej, która umożliwia wybór separatora i kodowania znaków, prezentuje wymaganą strukturę plików oraz sprawdza wysyłane dane (Rysunek 38). Proces importu przebiega w trzech etapach:

- sprawdzenia poprawności pliku, mapowania kolumn i wyboru separatora w aplikacji internetowej;
- wstępnego przetworzenia danych wraz ze wskazaniem i usunięciem błędnych rekordów;
- właściwego importu, wykonywanego w ramach transakcji bazodanowej.

Drugi scenariusz, czyli import danych pojedynczego badania, służy uzupełnieniu istniejącego już otworu o wyniki wiercenia lub sondowania. Obsługują go dwa punkty końcowe w ImportController:

- `/import/{projectId}/holes/{holeId}/borehole`
- `/import/{projectId}/holes/{holeId}/sounding`

Przykładem metody tworzącej pojedyncze wiercenie jest `createOrReplaceBorehole` (Listing 29).

Listing 27. Metoda `importHoles` (wraz z `validateHoleRequests`) odpowiedzialna za import otworów do systemu. Źródło: Opracowanie własne.

```
@Transactional
public List<HoleResponse> importHoles(Long projectId, List<CreateHoleRequest>
requests) {
    if (requests == null || requests.isEmpty()) {
        throw new ValidationException(List.of("Brak danych do importu"));
    }

    validateHoleRequests(requests);

    log.info("Importing batch of {} holes for project ID: {}", requests.size(),
projectId);
    return holeService.createHoles(projectId, requests);
}
private void validateHoleRequests(List<CreateHoleRequest> requests) {
    List<String> errors = new ArrayList<>();
    Set<String> namesInBatch = new HashSet<>();

    for (int i = 0; i < requests.size(); i++) {
        CreateHoleRequest request = requests.get(i);
        String prefix = "Wiersz " + (i + 1) + ": ";

        if (request.getName() == null || request.getName().trim().isEmpty()) {
            errors.add(prefix + "brak nazwy otworu");
        } else if (!namesInBatch.add(request.getName().trim().toLowerCase())) {
            errors.add(prefix + "zduplikowana nazwa otworu w pliku: '" +
request.getName().trim() + "'");
        }

        Double latitude = request.getLatitude();
        if (latitude == null || latitude.isNaN()) {
            errors.add(prefix + "brak lub nieprawidłowa szerokość geograficzna");
        } else if (latitude < -90 || latitude > 90) {
            errors.add(prefix + "szerokość geograficzna musi być w zakresie -90 do
90");
        }

        Double longitude = request.getLongitude();
        if (longitude == null || longitude.isNaN()) {
            errors.add(prefix + "brak lub nieprawidłowa długość geograficzna");
        } else if (longitude < -180 || longitude > 180) {
            errors.add(prefix + "długość geograficzna musi być w zakresie -180 do
180");
        }
    }

    if (!errors.isEmpty()) {
        throw new ValidationException(errors);
    }
}
```

Listing 28. Metoda createOrReplaceMultipleBoreholes odpowiedzialna za seryjne tworzenie lub zastępowanie encji Borehole. Źródło: Opracowanie własne.

```
@Transactional
public List<BoreholeResponse> createOrReplaceMultipleBoreholes(Long projectId,
List<CreateBoreholeRequest> requests) {
    User currentUser = securityContextService.getCurrentUser();
    List<BoreholeResponse> responses = new ArrayList<>();

    for (CreateBoreholeRequest request : requests) {
        Hole hole = findHoleByNameAndProjectId(request.getName(), projectId);
        validateProjectAccess(currentUser, hole.getProject(), projectId);
    }

    for (CreateBoreholeRequest request : requests) {
        try {
            Hole hole = findHoleByNameAndProjectId(request.getName(), projectId);
            BoreholeResponse response = createOrReplaceBorehole(projectId,
hole.getId(), request);
            responses.add(response);
        } catch (Exception e) {
            log.error("Failed to create/replace borehole for hole: {}, error: {}",
request.getName(), e.getMessage());
            throw e;
        }
    }
    return responses;
}
```

Listing 29. Metoda createOrReplaceBorehole odpowiedzialna za tworzenie jednej encji wiercenia. Źródło: Opracowanie własne.

```
@Transactional
public BoreholeResponse createOrReplaceBorehole(Long projectId, Long holeId,
CreateBoreholeRequest request) {
    User currentUser = securityContextService.getCurrentUser();
    Hole hole = findHoleById(holeId);
    Project project = hole.getProject();

    validateProjectAccess(currentUser, project, projectId);

    if (currentUser.getRole() == User.UserRole.GEOLOGIST) {
        validateGeologistAssignedToHole(currentUser, hole);
    }

    if (request.getLayers() != null && !request.getLayers().isEmpty()) {
        validateLayerDepthRanges(request.getLayers());
    }

    List<Borehole> existingBoreholes =
boreholeRepository.findByHoleIdWithLayers(holeId);

    Borehole borehole;
    if (!existingBoreholes.isEmpty()) {
        borehole = existingBoreholes.get(0);
        log.info("Updating existing borehole ID: {} for hole ID: {} by user: {}",
borehole.getId(), holeId, currentUser.getEmail());

        deleteAllLayerImages(borehole.getLayers());
        boreholeLayerRepository.deleteAll(borehole.getLayers());
        borehole.getLayers().clear();

        borehole.setDepth(request.getDepth());
    }
```

```

        borehole.setDiameter(request.getDiameter());
        borehole.setNotes(request.getNotes());
    } else {
        log.info("Creating borehole '{}' for hole ID: {} by user: {}",
request.getName(), holeId, currentUser.getEmail());

        borehole =
Borehole.builder().depth(request.getDepth()).diameter(request.getDiameter()).notes(
request.getNotes()).hole(hole).build();

        holeService.startHoleWork(hole);
    }

    Borehole savedBorehole = boreholeRepository.save(borehole);

    if (request.getLayers() != null && !request.getLayers().isEmpty()) {
        List<BoreholeLayer> layers = request.getLayers().stream().map(layerRequest
-> createBoreholeLayer(savedBorehole, layerRequest)).collect(Collectors.toList());
        List<BoreholeLayer> savedLayers = boreholeLayerRepository.saveAll(layers);
        savedBorehole.getLayers().addAll(savedLayers);
    }

    log.info("Borehole saved successfully with ID: {}", savedBorehole.getId());
    return mapToBoreholeResponse(savedBorehole);
}

```

← Powrót do projektu

Otwory Wiercenia Sondowania

Import otworów

Wybierz separator CSV (wymagany) Windows-1250 (Excel PL)

Wybierz plik CSV z danymi otworów

Wybierz separator CSV przed wgraniem pliku

Struktura pliku CSV:

Name;Latitude;Longitude;Voivodeship;County;Commune

Opis kolumn:

Name - nazwa otworu (wymagane)
Latitude - szerokość geograficzna (wymagane)
Longitude - długość geograficzna (wymagane)
Voivodeship - województwo (opcjonalne)
County - powiat (opcjonalne)
Commune - gmina (opcjonalne)

Przykładowe dane:

Name;Latitude;Longitude;Voivodeship;County;Commune
Otwór1;50.12345;19.12345;małopolskie;krakowski;Kraków
Otwór2;50.54321;19.54321;śląskie;katowicki;Katowice

Separator: przecinek (,), średnik (;) lub tabulator (tab)

Pola województwo, powiat i gmina są opcjonalne i mogą być puste. Współrzędne geograficzne mogą używać przecinka lub kropki jako separatora dziesiętnego.

Wiersze zawierające błędy (np. brak wymaganych pól) nie zostaną zaimportowane. Sprawdź komunikaty błędów i popraw format danych.

Rysunek 38. Ekran modułu importu listy encji do systemu KRET. Źródło: Opracowanie własne.

5.6. Wizualizacja mapowa i nawigacja

Komponent mapowy stanowi jeden z głównych elementów nawigacyjnych systemu i występuje zarówno w aplikacji internetowej, jak i mobilnej (Rysunek 57 i Rysunek 60). Mapa prezentuje rozmieszczenie punktów badawczych projektu, umożliwia otwarcie panelu wybranego otworu, jego edycję oraz uruchomienie nawigacji do punktu z wykorzystaniem usługi Google Maps. Użytkownik może przełączać podkłady mapowe co ułatwia lokalizację punktów badawczych w zróżnicowanym terenie.

Implementację po stronie aplikacji internetowej tworzą:

- komponent `ProjectMap`, okalający cały moduł zarządza stanem interfejsu oraz udostępnia akcje nawigacji, edycji otworu i przejścia do jego modułu;
- komponent `MapContent` (Listing 30), odpowiedzialny za właściwą wizualizację. Zbudowany z użyciem biblioteki `react-leaflet`, wyświetla mapę i markery oraz asynchronicznie pobiera dane. Jest przy tym ładowany dynamicznie z pominięciem renderowania po stronie serwera (leniwe ładowanie, ang. *lazy loading*), co zmniejsza początkowy rozmiar strony;
- konfiguracja warstw mapowych pochodzących od różnych dostawców, m.in. `OpenStreetMap` i `OpenTopoMap`.

Listing 30. Komponent `MapContent` odpowiedzialny za wizualizację mapową. Źródło: Opracowanie własne.

```
function MapContent({
  holes,
  center,
  providerConfig,
  selectedHole,
  onHoleClick,
}: MapContentProps) {
  return (
    <MapContainer
      center={[center.lat, center.lng]}
      zoom={13}
      className="w-full h-full"
      style={{ minHeight: '400px' }}
    >
      <TileLayer
        url={providerConfig.urlTemplate}
        attribution={providerConfig.attribution}
        maxZoom={providerConfig.maxZoom}
      />
      <MapUpdater center={center} holes={holes} />

      {holes.map((hole) => (
        <Marker
          key={hole.id}
          position={[hole.latitude, hole.longitude]}
          icon={selectedHole?.id === hole.id ? selectedIcon : defaultIcon}
          eventHandlers={{
            click: () => onHoleClick(hole),
          }}
        />
      ))}
    </MapContainer>
  );
}
```

Komponent mapowy wspiera również tworzenie otworów. Współrzędne nowego punktu można podać w polach formularza lub wskazać bezpośrednio na mapie (Rysunek 58), a położenie istniejącego

markera skorygować przeciągnięciem. Wybór lokalizacji na mapie uzupełnia współrzędne geograficzne oraz uruchamia serwis `openStreetMapService`, który na ich podstawie pobiera dane administracyjne punktu: województwo, powiat i gminę.

5.7. Formularze badań geologicznych

Formularze badań geologicznych stanowią centralny element aplikacji klienckich systemu KRET. Z perspektywy użytkownika końcowego są to dwa odrębne typy formularzy odpowiadające dwóm rodzajom badań prowadzonych przez geologów terenowych: wierceniu oraz sondowaniu dynamicznemu. Z perspektywy implementacji oba formularze wykorzystują wspólny zestaw mechanizmów: sprawdzanie danych pod stronie klienta na podstawie reguł zdefiniowanych w bibliotece współdzielonej, automatyczny zapis wprowadzanych zmian, dynamiczne pola niestandardowe, mechanizm wzorców formularzy oraz dokumentację fotograficzną.

5.7.1. Formularz wiercenia

Formularz wiercenia (Rysunek 40) służy do dokumentowania profilu litologicznego rozpoznawanego za pomocą wiertnicy. Główną strukturą danych jest uporządkowana lista warstw geologicznych, z których każda opisana jest głębokością początkową i końcową, typem litologicznym według wyliczenia `LithologyType`, opcjonalnym opisem słownym, wilgotnością, plastycznością, miąższością oraz znacznikiem zwierciadła wody gruntowej. Edycję warstwy przedstawia Rysunek 39.

The screenshot shows a mobile application interface for editing a geological layer. The form is titled "Edytuj warstwę" and includes the following fields and controls:

- Od (m) ***: Input field with value "0".
- Do (m) ***: Input field with value "1".
- Opis litologiczny**: Text area containing "Bardzo pylący".
- Typ litologiczny**: Dropdown menu showing "Pn Piasek pylasty".
- Miąższość (m)**: Input field with value "12".
- Wilgotność**: Input field with value "1".
- Plastyczność**: Input field with value "1".
- Zwierciadło wody**: Checkable option, currently checked.
- Buttons**: "Anuluj" (Cancel) and "Zapisz" (Save).

Rysunek 39. Ekran edycji warstwy w formularzu wiercenia w aplikacji mobilnej. Źródło: Opracowanie własne.

14:46

...



 Odwiert



Regional Geological Survey 2025 - Hole 002

 Auto-zapis



Podstawowe informacje

Nazwa badania

Regional Geological Survey 2025 - Hole 002

Głębokość (m)

25

Średnica (m)

8

Warstwy geologiczne

1

 Tryb zdjęć 360°

☐

0.00 - 1.00 m

Piasek pylasty



1 zdjęcie



Aparat

Galeria

+ 1.00 m

Notatki

1

Piach

Projekty

Badanie

Mapa

Sprzęt

Profil

Rysunek 40. Ekran formularza wiercenia w aplikacji mobilnej z listą warstw geologicznych. Źródło: Opracowanie własne.

Implementacja formularza po stronie aplikacji mobilnej oparta jest na `useBoreholeForm` z biblioteki współdzielonej. Hak ten zarządza stanem formularza, inicjalizuje formularz w przypadku jego edycji, zwraca uporządkowaną listę warstw oraz zarządza mechanizmem automatycznego zapisu.

5.7.2. Formularz sondowania

Formularz sondowania (Rysunek 41) służy do dokumentowania badania sondą dynamiczną, podczas którego rejestrowana jest liczba uderzeń sondy N potrzebna do zagłębienia jej o ustalony interwał. Główną strukturą danych jest lista pomiarów uporządkowanych według głębokości, gdzie każdy pomiar opisany jest głębokością oraz liczbą uderzeń.

W systemie KRET zaimplementowano cztery typy sond dynamicznych zdefiniowane w wyliczeniu `SoundingKind`: DPL (*Dynamic Probe Light*), DPM (*Dynamic Probe Medium*), DPH (*Dynamic Probe Heavy*) oraz DPSH (*Dynamic Probe Super Heavy*). Każdy typ posiada własne wartości domyślne ciężaru bijaka, wysokości spadku oraz powierzchni stożka.

14:41

Sondowanie

Regional Geological Survey 2025 - Hole 002

Auto-zapis

Podstawowe informacje

Nazwa sondowania

Regional Geological Survey 2025 - Hole 002

Typ sondowania

DPL DPM DPH DPSH

Miot 10kg · Spadek 0.5m · Stożek 10cm² · Interwał 10cm

Głębokość (m)

0.8

Woda (m)

0.6

Pomiary

2

GŁĘBOKOŚĆ	UDERZENIA
0.10	3
0.50	8

+ Dodaj pomiar (0.60 m)

Zakończenie

Powód zakończenia

Osiągnięto docelową głębokość

Notatki

Grunt piaszczysty, dobre warunki do sondowania

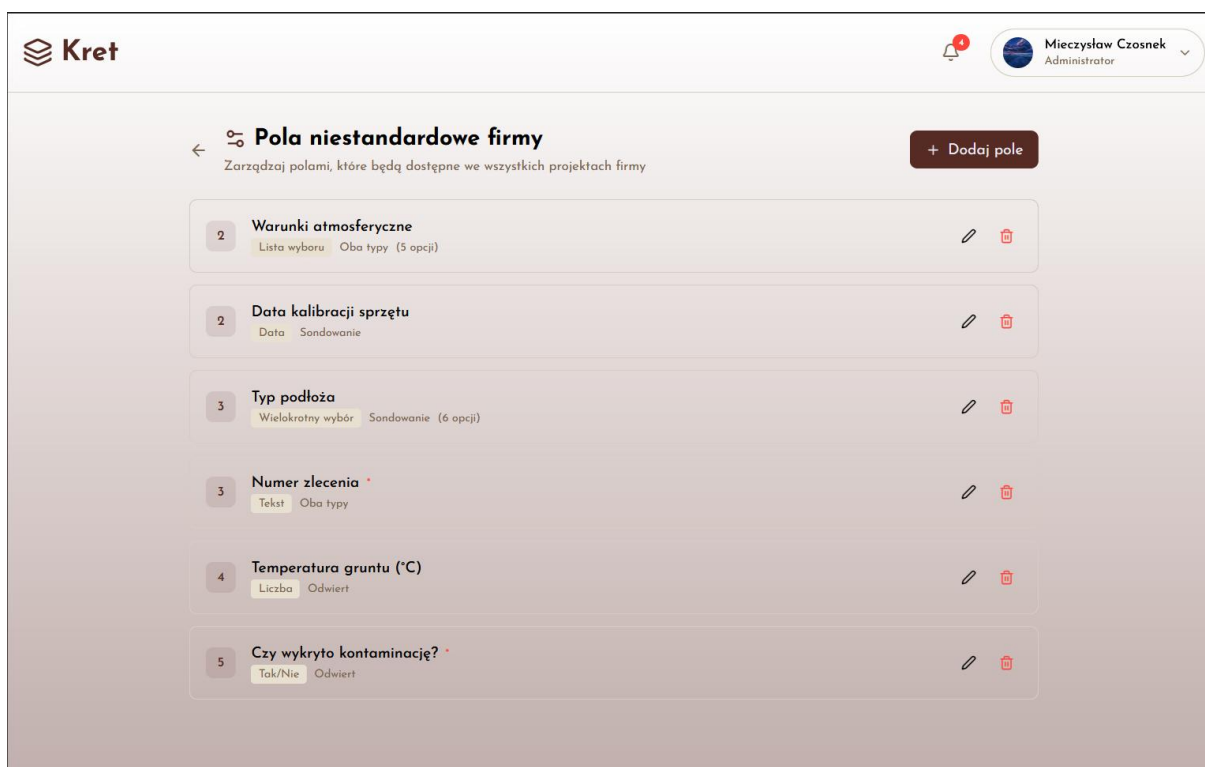
Projekty Badanie Mapa Sprzęt Profil

Rysunek 41. Ekran formularza sondowania w aplikacji mobilnej. Źródło: Opracowanie własne.

5.7.3. Pola niestandardowe

Mechanizm pól niestandardowych pozwala administratorom oraz dokumentatorom dostosować zakres gromadzonych danych do specyfiki konkretnej firmy lub projektu, bez konieczności modyfikacji kodu aplikacji. Pola niestandardowe definiowane są w aplikacji internetowej (Rysunek 42) i mogą być następnie aktywowane lub dezaktywowane na poziomie konkretnych projektów. Technicznie jest to realizowane za pomocą tabeli pośredniczącej `project_custom_fields_settings`.

Każde pole niestandardowe posiada typ zdefiniowany w wyliczeniu `CustomFieldType`: tekst, liczba, data, wartość logiczna, lista wyboru pojedynczego, lista wielokrotnego wyboru lub plik. W zależności od typu pole posiada zestaw atrybutów konfiguracyjnych (minimalna i maksymalna wartość liczbowa, minimalna i maksymalna długość tekstu, opcje do wyboru, wartość domyślna, oznaczenie wymagania wypełnienia). Definicja pola określa również, w którym typie badania jest ono używane, jest możliwość wyboru sondowania, wiercenia lub obu opcji naraz. Wartości pól niestandardowych przechowywane są w polimorficznej tabeli `custom_field_values`, gdzie konkretna kolumna wartości wypełniana jest w zależności od typu pola.



Rysunek 42. Panel definiowania pól niestandardowych w aplikacji internetowej. Źródło: Opracowanie własne.

5.7.4. Wzorce formularzy

Mechanizm wzorców formularzy służy do przyspieszania powtarzalnego wprowadzania danych w terenie. Geolog pracujący na konkretnym projekcie często wykonuje serię badań o podobnym profilu. Bez mechanizmu wzorców każde takie badanie trzeba wypełniać od zera. Wzorec to predefiniowany zestaw wartości pól, który może zostać zastosowany do nowego formularza jednym kliknięciem. Okno wyboru wzorca prezentuje Rysunek 43.

Wybierz wzorzec

Moj wzorzec
Projekt
8 pól

Wzorzec dla projektu
Projekt
9 pól

POLA STANDARDOWE (3)

Głębokość = 25
Średnica = 8
Uwagi = Piach

POLA NIESTANDARDOWE (6)

Warunki atmosferyczne = Deszczowo
Numer zlecenia = 12345678
Temperatura gruntu (°C) = 50

Czy wykryto zanieczyszczenie? = Nie
Numer próbki laboratoryjnej = 1234hhht
Głębokość poboru próbki (m) = 25

Krzysztof Krawczyk

Zastosuj wzorzec

Rysunek 43. Okno wyboru wzorca formularza w aplikacji mobilnej. Źródło: Opracowanie własne.

Wzorce mogą funkcjonować na dwóch poziomach: firmowym oraz projektowym ograniczonym do jednego projektu. Każdy wzorzec posiada również poziom widoczności: prywatny lub publiczny (w obrębie firmy). Treść wzorca przechowywana jest w kolumnie JSON `fields_json`, co pozwala na elastyczną reprezentację stanu formularza bezkompromisowo uwzględniając dynamiczne wartości pól niestandardowych.

5.7.5. Automatyczny zapis formularzy

Praca terenowa odbywa się w warunkach, w których utrata danych byłaby szczególnie kosztowna. W najgorszym wypadku wymóg ponownego wiercenia stanowiłby znaczny nakład czasu i wysiłku logistycznego na potrzeby pojedynczego otworu. Z tego powodu w formularzach badań zaimplementowano mechanizm automatycznego zapisywania zapisu z opóźnieniem (ang. *debounce*). Zapis aktywowany jest po krótkiej przerwie od ostatniej modyfikacji formularza, co stanowi kompromis między częstotliwością synchronizacji z serwerem a obciążeniem łącza i baterii urządzenia. Listing 31 przedstawia kluczowy fragment implementacji.

Listing 31. Mechanizm automatycznego zapisu w haku `useBoreholeForm`. Źródło: Opracowanie własne.

```
const performAutoSave = useCallback(async () => {
  const currentFormData = formDataRef.current;
  const currentLayers = layersRef.current;

  if (!currentFormData.name.trim()) return;

  if (hasLayersOverlap(currentLayers)) {
    onOverlapWarningRef.current?.();
    return;
  }
})
```

```

const payload = {
  name: currentFormData.name.trim(),
  depth: currentFormData.depth
  ? parseNumber(currentFormData.depth) : undefined,
  diameter: currentFormData.diameter
  ? parseNumber(currentFormData.diameter) : undefined,
  notes: currentFormData.notes.trim() || undefined,
  layers: mapFormLayersToPayload(currentLayers),
};

setIsAutoSaving(true);
setLastSaveStatus(null);

const result = await onSaveRef.current(payload);

setIsAutoSaving(false);
setLastSaveStatus(result.success ? 'saved' : 'error');

if (!result.success && result.error) {
  onErrorRef.current?.(result.error);
}

setTimeout(() => setLastSaveStatus(null), 2000);
}, []);

```

Przed wykonaniem zapisu mechanizm sprawdza, czy warstwy litologiczne nie nakładają się na siebie, co w razie konfliktu wstrzymuje zapis automatyczny, po czym użytkownik dostaje ostrzeżenie. Zapobiega to utrwalaniu niespójnych danych, Stan zapisu udostępniany jest w interfejsie użytkownika, dzięki czemu pracujący nad formularzem ma stałą informację zwrotną czy jego praca została zachowana.

Mechanizm automatycznego zapisu działa dwutorowo. Pola standardowe wraz z listą warstw zapisywane są przez haki `useBoreholeForm` oraz `useSoundingForm`, natomiast wartości pól niestandardowych obsługiwane są przez odrębny hak `useCustomFields`, który analogicznym mechanizmem zapisu wysyła zmiany do końcówki `custom-field-values`.

5.7.6. Dokumentacja fotograficzna i zdjęcia 360

Każda warstwa geologiczna w formularzu wiercenia może zostać udokumentowana fotograficznie. Zdjęcia powiązane są z konkretną warstwą i przechowywane w pamięci obiektowej `MinIO` za pośrednictwem mechanizmu opisanego w podrozdziale 5.4.

System rozróżnia dwa tryby dokumentacji. Tryb standardowy pozwala dołączyć do warstwy dowolną liczbę pojedynczych zdjęć z aparatu lub galerii. Tryb panoramiczny 360° służy do dokumentacji rdzenia wiertniczego. Zestaw panoramiczny wymaga minimum czterech zdjęć (stała `MIN_360_PHOTOS`), a usunięcie obejmuje zawsze cały zestaw – nie można skasować pojedynczego zdjęcia, co zapobiega doprowadzeniu panoramy do stanu niekompletnego. Obie reguły zaimplementowano jako funkcje sprawdzające w bibliotece współdzielonej, co przedstawia Listing 32.

Listing 32. Sprawdzenie zestawu zdjęć panoramicznych 360° w bibliotece współdzielonej. Źródło: Opracowanie własne.

```

export const canUpload360Images = (
  existing360Count: number,
  newFilesCount: number
): { allowed: boolean; error?: string } => {
  const hasExistingSet = existing360Count >= MIN_360_PHOTOS;
  if (hasExistingSet) {
    return { allowed: true };
  }
  if (newFilesCount < MIN_360_PHOTOS) {
    return {
      allowed: false,

```

```

        error: `Zdjęcia 360° wymagają minimum ${MIN_360_PHOTOS} `
            + `zdjęć. Wybrano: ${newFilesCount}`,
    };
}
return { allowed: true };
};

export const canDelete360Image = (
    current360Count: number
): { allowed: boolean; error?: string } => {
    const remainingAfterDelete = current360Count - 1;
    if (remainingAfterDelete > 0 && remainingAfterDelete < MIN_360_PHOTOS) {
        return {
            allowed: false,
            error: `Nie można usunąć pojedynczego zdjęcia 360°. `
                + `Usuń wszystkie zdjęcia 360° z tej warstwy.`,
        };
    }
    return { allowed: true };
};

```

Sprawdzenie obu reguł (`canUpload360Images` oraz `canDelete360Image`) umieszczona została w bibliotece współdzielonej, dzięki czemu obowiązują identycznie niezależnie od miejsca wywołania. Przesyłanie odbywa się asynchronicznie ze wskaźnikiem postępu, a w trybie pracy bez łączności z siecią operacje na zdjęciach kolejgowane są do późniejszej synchronizacji.

5.7.7. Automatyczne sprawdzanie i oddawanie badań do sprawdzenia

Po zakończeniu prac w terenie geolog oddaje badanie do sprawdzenia, co rozpoczyna jego dalszą weryfikację w biurze. Operacja ta nie sprowadza się do zwykłej zmiany statusu, a poprzedza ją automatyczne sprawdzenie kompletności i poprawności danych, wykonywane po stronie serwera w metodzie `submitHole` klasy `HoleService`.

Przed przyjęciem badania serwer weryfikuje kilka warunków. Oddać badanie może wyłącznie geolog przypisany do otworu, a sam otwór musi znajdować się w statusie roboczym (`IN_PROGRESS`) oraz zawierać dane wiercenia lub sondowania. Następnie uruchamiane jest szczegółowe sprawdzenie merytoryczne w metodzie `validateHoleForSubmission`, która sprawdza poprawność danych odpowiednio dla wiercenia lub sondowania, gromadząc wszystkie wykryte uchybienia na liście błędów. Jeżeli lista nie jest pusta, operacja zostaje przerwana wyjątkiem `ValidationException` zawierającym komplet komunikatów, dzięki czemu użytkownik otrzymuje od razu pełną listę rzeczy do poprawienia, zamiast korygować błędy pojedynczo. Listing 33 przedstawia fragment tego sprawdzenia dla wiercenia.

Listing 33. Fragment sprawdzenia danych wiercenia przed oddaniem badania. Źródło: Opracowanie własne.

```

private void validateBoreholeForSubmission(
    Borehole borehole, Hole hole, List<String> errors) {
    if (borehole.getDepth() == null) {
        errors.add("Odwiert: brak głębokości");
    }
    if (borehole.getDiameter() == null) {
        errors.add("Odwiert: brak średnicy");
    }

    List<BoreholeLayer> layers = borehole.getLayers();
    for (int i = 0; i < layers.size(); i++) {
        BoreholeLayer layer = layers.get(i);
        String prefix = "Warstwa " + (i + 1) + ": ";
        if (layer.getDepthFrom() == null) {
            errors.add(prefix + "brak głębokości od");
        }
    }
}

```

```

    }
    if (layer.getDepthTo() == null) {
        errors.add(prefix + "brak głębokości do");
    }
    if (layer.getLithologyType() == null) {
        errors.add(prefix + "brak litologii");
    }
    if (layer.getDepthFrom() != null && layer.getDepthTo() != null
        && layer.getDepthFrom().compareTo(layer.getDepthTo()) > 0) {
        errors.add(prefix + "głębokość od > do");
    }
}

validateBoreholeLayerOverlaps(layers, errors);
}

```

Po pomyślnej serii sprawdzeń otwór zmienia status na oddany do sprawdzenia (SUBMITTED), a system automatycznie przydziela dokumentatora odpowiedzialnego za weryfikację. Mechanizm tego przydziału omówiono w podrozdziale 5.8. Zmiana statusu wyzwała powiadomienie informujące przypisanego dokumentatora o przekazaniu badania do weryfikacji. Otwór oznaczony jako oddany do sprawdzenia nie może już być edytowany przez geologa, co zabezpiecza dane przed modyfikacją w trakcie weryfikacji.

5.7.8. Usuwanie badań

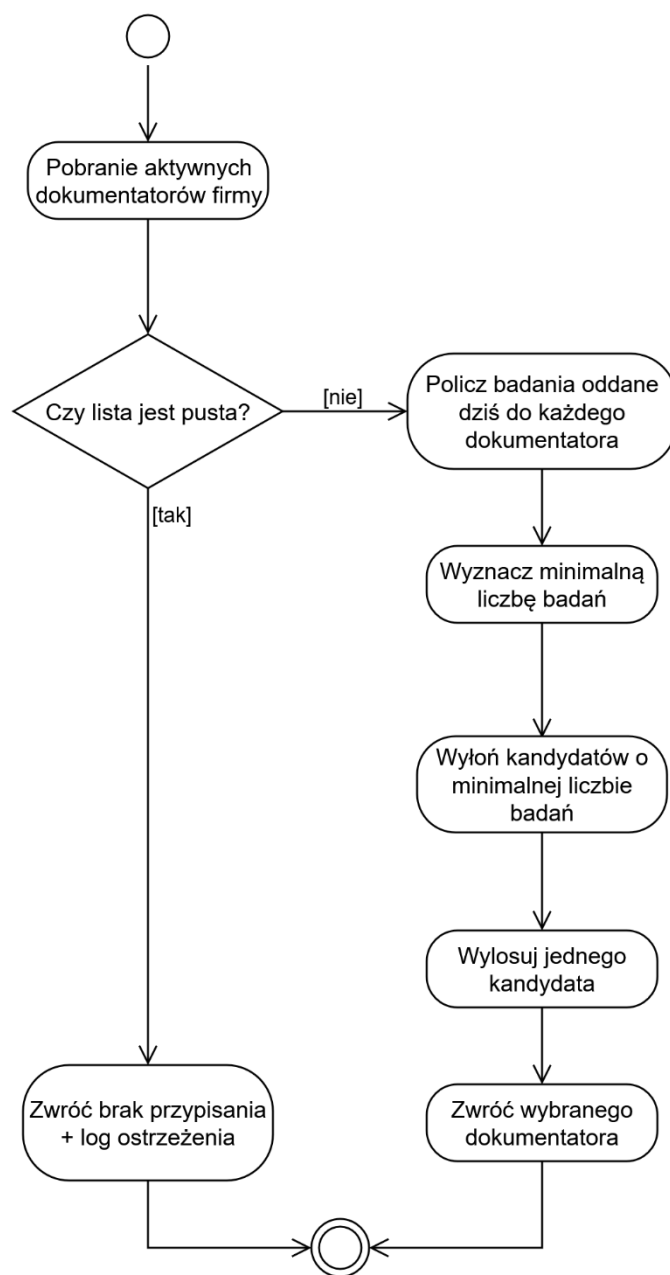
Usunięcie danych badania jest operacją nieodwracalną, dlatego obwarowano ją zestawem reguł chroniących integralność danych projektu. Przed wykonaniem operacji serwer weryfikuje uprawnienia użytkownika do projektu, a w przypadku geologa dodatkowo sprawdza czy jest on przypisany do danego otworu – geolog nie może usuwać badań z otworów, do których nie został przydzielony. Kluczowym zabezpieczeniem jest blokada usuwania badań o statusie oddanym do sprawdzenia (SUBMITTED) lub zamkniętym (CLOSED). Próba taka kończy się wyjątkiem `ValidationException`, ponieważ dane przekazane do weryfikacji lub już zatwierdzone nie powinny zniknąć z projektu.

Usunięcie badania pociąga za sobą skasowanie powiązanej z nim dokumentacji fotograficznej. Przed usunięciem samego wiercenia metoda usuwa wszystkie zdjęcia przypisane do jego warstw, zarówno z bazy metadanych, jak i z pamięci obiektowej `MinIO`, co zapobiega pozostawieniu osieroconych obiektów zajmujących miejsce w magazynie. Po usunięciu danych badania sam otwór pozostaje w systemie. Usunięciu podlega jedynie powiązane z nim wiercenie lub sondowanie, dzięki czemu otwór może zostać udokumentowany ponownie.

5.8. Automatyczne przypisywanie dokumentatorów

W typowej firmie geologicznej liczba prowadzonych równolegle badań terenowych przewyższa liczbę dokumentatorów zdolnych do ich weryfikacji. Bez wsparcia narzędzi informatycznych podział pracy weryfikacyjnej odbywa się w razie potrzeby i nie jest rozwiązywany systemowo, tylko przez przykładowo kierownika projektu, który rozdziela pracę, czego rezultatem może być nierównomierne rozłożenie pracy.

W systemie KRET zaimplementowano algorytm automatycznego przypisywania dokumentatorów oparty na zmodyfikowanym schemacie „każdy z każdym” (ang. *round-robin*) z dynamicznym wyważeniem obciążenia, co przedstawia Rysunek 44. Algorytm uruchamiany jest w metodzie `submitHole` klasy `HoleService` w momencie, gdy geolog terenowy przekazuje badanie do weryfikacji.



Rysunek 44. Diagram działania algorytmu automatycznego przypisywania dokumentatorów. Źródło: Opracowanie własne.

Implementację algorytmu przedstawia Listing 34.

Listing 34. Algorytm automatycznego przypisywania dokumentatorów. Źródło: Opracowanie własne.

```

private User assignDocumenterRoundRobin(Hole hole) {
    Long companyId = hole.getProject().getCompany().getId();
    List<User> documenters = userRepository
        .findByCompanyIdAndRole(companyId, User.UserRole.DOCUMENTER)
        .stream()
        .filter(User::isEnabled)
        .toList();
}

```

```

if (documenters.isEmpty()) {
    log.warn("No active documenters found for company {}", " +
        "skipping documenter assignment", companyId);
    return null;
}

Instant startOfDay = LocalDate.now(ZoneOffset.UTC)
    .atStartOfDay(ZoneOffset.UTC).toInstant();
Instant startOfNextDay = startOfDay.plus(1, ChronoUnit.DAYS);

Map<Long, Long> countMap = documenters.stream().collect(
    Collectors.toMap(
        User::getId,
        d -> holeRepository.countByAssignedDocumenterIdToday(
            d.getId(), startOfDay, startOfNextDay)));

long minCount = countMap.values().stream()
    .mapToLong(Long::longValue).min().orElse(0);

List<User> candidates = documenters.stream()
    .filter(d -> countMap.get(d.getId()) == minCount)
    .toList();

return candidates.get(new Random().nextInt(candidates.size()));
}

```

Algorytm składa się z czterech kroków. Pierwszy krok to pobranie wszystkich aktywnych dokumentatorów firmy. Drugi krok to obliczenie liczby badań przekazanych dziś do każdego z tych dokumentatorów. Trzeci krok to wyłonienie kandydatów o najmniejszej ilości przypisanych badań. Wyważanie obciążenia w cyklu dziennym, a nie globalnym, sprawia, że krótkotrwała niedostępność jednego z dokumentatorów nie powoduje, że po jego powrocie zostanie on przeciążony badaniami do sprawdzenia. Czwarty krok to losowy wybór z kandydatów o najmniejszym obciążeniu.

Algorytm ten stanowi rozwiązanie pośrednie między klasycznym „każdy z każdym”, gdzie kolejność determinuje wybór, a algorytmem najmniejszej liczby zadań. Wybór ten podyktowany był specyfiką pracy dokumentatora geologii – weryfikacja jednego badania zajmuje od kilku do kilkudziesięciu minut, a obciążenie zmienia się skokowo w ciągu dnia, zależnie od natężenia prac geologów terenowych.

5.9. System dyskusji

System dyskusji zapewnia komunikację między dokumentatorem a geologami terenowymi w kontekście sprawdzanych badań. Wątek dyskusyjny można założyć dla dowolnego pola formularza wiercenia lub sondowania (Rysunek 75 i Rysunek 76), co pozwala precyzyjnie wskazać miejsce wymagające wyjaśnienia. Dyskusje stanowią integralną część procesu weryfikacji danych. Ich celem jest wyjaśnianie niejasności i wsparcie geologa terenowego w poprawie dokumentacji.

Logikę modułu realizuje `DiscussionService`, udostępniający operacje CRUD na wątkach i wiadomościach. Serwis ściśle współpracuje z `NotificationService`. Utworzenie dyskusji skutkuje powiadomieniem zainteresowanych użytkowników. Przebieg tworzenia wątku, wraz z powiązaniem go z właściwym badaniem na podstawie przekazanego typu, przedstawia metoda `createDiscussion` (Listing 35).

Listing 35. Metoda `createDiscussion` odpowiedzialna za tworzenie dyskusji pod określonym polem formularza badania. Źródło: Opracowanie własne.

```
@Transactional
```

```

public DiscussionResponse createDiscussion(Long projectId, Long holeId, String
examinationType,
                                CreateDiscussionRequest request) {
    User currentUser = securityContextService.getCurrentUser();
    Hole hole = findHoleById(holeId);
    validateProjectAccess(currentUser, hole.getProject(), projectId);

    Discussion.DiscussionBuilder builder = Discussion.builder()
        .fieldPath(request.getFieldPath())
        .fieldLabel(request.getFieldLabel())
        .createdBy(currentUser);

    if ("borehole".equalsIgnoreCase(examinationType)) {
        Borehole borehole =
boreholeRepository.findByHoleId(holeId).stream().findFirst()
        .orElseThrow(() -> new ResourceNotFoundException("Nie
znaleziono odwiertu dla otworu o ID: " + holeId));
        builder.borehole(borehole);
    } else if ("sounding".equalsIgnoreCase(examinationType)) {
        Sounding sounding = soundingRepository.findByHoleId(holeId)
        .orElseThrow(() -> new ResourceNotFoundException("Nie
znaleziono sondowania dla otworu o ID: " + holeId));
        builder.sounding(sounding);
    } else {
        throw new ValidationException("Nieprawidłowy typ badania: " +
examinationType + ". Dozwolone wartości: 'borehole' lub 'sounding'");
    }

    Discussion discussion = builder.build();

    DiscussionMessage initialMessage = DiscussionMessage.builder()
        .discussion(discussion)
        .author(currentUser)
        .content(request.getInitialMessage())
        .build();
    discussion.getMessages().add(initialMessage);

    Discussion saved = discussionRepository.save(discussion);
    log.info("Discussion created with ID: {} on {} field '{}'' by user: {}",
        saved.getId(), examinationType, request.getFieldPath(),
currentUser.getEmail());

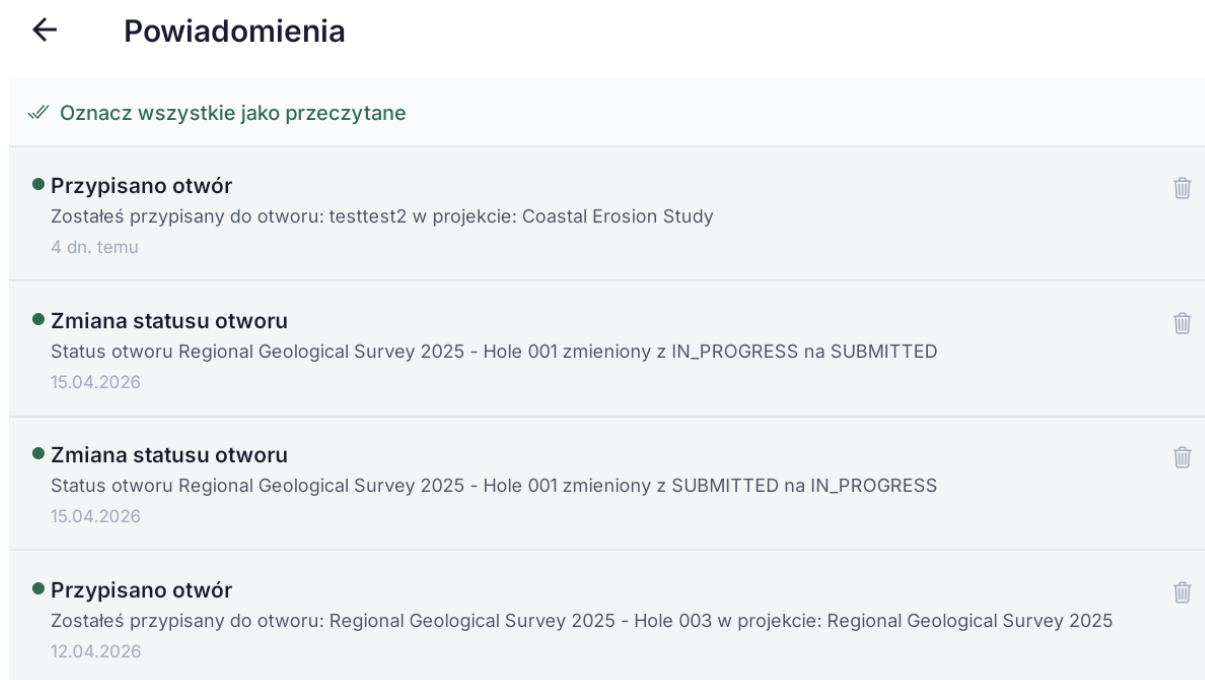
    notificationService.notifyDiscussionCreated(saved, hole, currentUser);

    return mapToResponse(saved);
}

```

5.10. Powiadomienia

System powiadomień informuje użytkowników o zdarzeniach wymagających ich uwagi. Powiadomienia trafiają do użytkownika dwoma kanałami: jako wpis w wewnętrznej skrzynce dostępnej z poziomu aplikacji internetowej i mobilnej (Rysunek 45) oraz jako powiadomienie wypychane (ang. *push notification*) wysyłane na urządzenie mobilne.



Rysunek 45. Ekran listy powiadomień w aplikacji mobilnej. Źródło: Opracowanie własne.

Typy powiadomień zdefiniowane są w wyliczeniu `NotificationType` i obejmują sześć kategorii zdarzeń przedstawionych w Tabeli 23.

Tabela 23. Typy powiadomień w systemie KRET. Źródło: Opracowanie własne.

Typ	Wyzwalacz	Odbiorca
PROJECT_ASSIGNED	Dodanie użytkownika do projektu	Dodany użytkownik
HOLE_ASSIGNED	Przypisanie geologa do otworu	Przypisany geolog
HOLE_STATUS_CHANGED	Zmiana statusu otworu	Przypisani geolodzy i dokumentator
DISCUSSION_NEW_MESSAGE	Nowa wiadomość w dyskusji	Uczestnicy dyskusji i dokumentator otworu
EQUIPMENT_RETURNED	Zwrot zarezerwowanego sprzętu	Właściciel rezerwacji
EQUIPMENT_RESERVATION_CANCELLED	Anulowanie rezerwacji sprzętu	Właściciel rezerwacji

Każde powiadomienie zawiera tytuł, treść, znacznik czasu utworzenia, flagę przeczytania oraz metadane w formacie JSON pozwalające aplikacjom klienckim na nawigację do kontekstu zdarzenia. Listing 36 przedstawia fragment klasy `NotificationService` odpowiedzialny za utworzenie

powiadomienia o zmianie statusu otworu oraz jego utrwalenie wraz z próbą wysyłki notyfikacji wypychanej.

Listing 36. Tworzenie i wysyłka powiadomienia o zmianie statusu otworu. Źródło: Opracowanie własne.

```
@Transactional
public void notifyHoleStatusChanged(Hole hole,
                                   String oldStatus, String newStatus) {
    User currentUser = securityContextService.getCurrentUser();
    Set<User> recipients = new HashSet<>(hole.getAssignedGeologists());
    if (hole.getAssignedDocumenter() != null) {
        recipients.add(hole.getAssignedDocumenter());
    }
    recipients.remove(currentUser);

    Map<String, Object> metadata = Map.of(
        "projectId", hole.getProject().getId(),
        "holeId", hole.getId());
    String message = "Status otworu " + hole.getName() +
        " zmieniony z " + oldStatus + " na " + newStatus;

    for (User recipient : recipients) {
        createAndSendNotification(recipient,
            NotificationType.HOLE_STATUS_CHANGED,
            "Zmiana statusu otworu", message, metadata);
    }
}

private void createAndSendNotification(User user, NotificationType type,
    String title, String message, Map<String, Object> metadata) {
    Notification notification = Notification.builder()
        .user(user).type(type).title(title)
        .message(message).metadata(metadata).build();
    Notification saved = notificationRepository.save(notification);

    try {
        expoPushService.sendPushNotification(saved);
    } catch (Exception e) {
        log.warn("Failed to send push notification for user {}: {}",
            user.getEmail(), e.getMessage());
    }
}
```

Powiadomienie najpierw zapisywane jest w bazie danych, a dopiero później następuje próba wysłania powiadomienia na urządzenie mobilne. Wysyłka powiadomienia wypychanego otoczona jest blokiem przechwytyjącym wyjątki, dzięki czemu ewentualne błędy nie przerywają aktualnego potoku. Osoba wywołująca zmianę zawsze jest pomijana jako odbiorca notyfikacji.

Wysyłkę powiadomienia wypychanego realizuje `expoPushService`, korzystający z publicznego interfejsu Expo Push API. Powiadomienie wysyłane jest na wszystkie urządzenia danego użytkownika. Jest to możliwe dzięki mechanizmowi tokenów urządzeń. Aplikacja mobilna rejestruje swój unikalny token urządzenia, a serwer zapisuje go w tabeli `device_tokens`, dzięki czemu użytkownicy mogą otrzymywać powiadomienia na wiele urządzeń. Powiadomienie wypychane również posiada typ zdarzenia i jego metadane, co pozwala aplikacji mobilnej na kierowanie użytkownika do kontekstu powiadomienia.

5.11. System eksportu danych

Moduł eksportu udostępnia wyniki pracy w projekcie w formatach CSV, XLSX i PDF, z możliwością wyboru zakresu danych (Rysunek 93). Dla dwóch pierwszych formatów użytkownik

decyduje dodatkowo o kodowaniu znaków, separatorze oraz formacie zapisu dat, litologii i współrzędnych (Rysunek 94), a powtarzalne konfiguracje może zapisywać jako szablony eksportu. Zdjęcia warstw nie wchodzą w skład plików wynikowych – ich eksport realizowany jest osobno. Cała funkcjonalność dostępna jest wyłącznie w aplikacji internetowej.

Moduł tworzą dwa elementy:

- komponent `ExportProjectPage`, w którym użytkownik definiuje zakres, format i opcje zaawansowane eksportu;
- serwis `ExportService` po stronie serwera, udostępniony przez `ExportController`.

Proces rozpoczyna się od weryfikacji uprawnień użytkownika w metodzie `exportProject` (Listing 37). Następnie metoda `collectExportData` (Listing 38) agreguje encje objęte wybranym zakresem: otwory, wiercenia z warstwami geologicznymi, sondowania z pomiarami, pola niestandardowe oraz zdjęcia warstw. Po skompletowaniu danych sterowanie przekazywane jest do metody `exportToCsv` lub `exportToXlsx`, odpowiedzialnej za utworzenie pliku wynikowego.

Listing 37. Metoda `exportProject` odpowiedzialna za eksport plików z systemu. Źródło: Opracowanie własne.

```
@Transactional(readOnly = true)
public ExportResult exportProject(Long projectId, ExportProjectRequest request)
{
    User currentUser = securityContextService.getCurrentUser();
    accessControlService.requireAdminOrDocumenter(currentUser);

    Project project = projectRepository.findById(projectId)
        .orElseThrow(() -> new ResourceNotFoundException("Nie znaleziono
projektu o ID: " + projectId));
    accessControlService.requireProjectAccessByCompany(currentUser, project);

    if (!request.getFormat().isImplemented()) {
        throw new ValidationException("Format " + request.getFormat() + " nie
jest jeszcze zaimplementowany");
    }

    ExportOptions options = request.getOptions() != null ?
request.getOptions() : ExportOptions.builder().build();
    ExportScope scope = request.getScope() != null ? request.getScope() :
ExportScope.builder().build();
    List<ExportFieldMapping> mappings = request.getCustomMappings();

    ExportData data = collectExportData(project, scope, options);

    return switch (request.getFormat()) {
        case CSV -> exportToCsv(project, data, options, scope, mappings);
        case XLSX -> exportToXlsx(project, data, options, scope, mappings);
        default -> throw new ValidationException("Format " +
request.getFormat() + " nie jest jeszcze zaimplementowany");
    };
}
```

Listing 38. Fragment metody `collectExportData` odpowiedzialnej za agregację encji objętych eksportem. Źródło: Opracowanie własne.

```
private ExportData collectExportData(Project project, ExportScope scope,
ExportOptions options) {
    List<Hole> holes = scope.getIncludeHoles()
        ? holeRepository.findById(project.getId()) : List.of();

    Map<Long, Borehole> boreholesByHoleId = new HashMap<>();
    Map<Long, List<BoreholeLayer>> layersByBoreholeId = new HashMap<>();
}
```

```

Map<Long, List<BoreholeLayerImage>> imagesByLayerId = new HashMap<>();

for (Hole hole : holes) {
    if (scope.getIncludeBoreholes() && hole.getBorehole() != null) {
        Borehole borehole = hole.getBorehole();
        boreholesByHoleId.put(hole.getId(), borehole);

        if (scope.getIncludeLayers()) {
            List<BoreholeLayer> layers = boreholeLayerRepository
                .findByBoreholeIdOrderByDepthFromAsc(borehole.getId());
            layersByBoreholeId.put(borehole.getId(), layers);

            if (scope.getIncludeImages()) {
                for (BoreholeLayer layer : layers) {
                    List<BoreholeLayerImage> images =
boreholeLayerImageRepository
                        .findByBoreholeLayerId(layer.getId());
                    if (!images.isEmpty()) {
                        imagesByLayerId.put(layer.getId(), images);
                    }
                }
            }
        }
    }

    // analogiczna gałąź gromadzi sondowania wraz z pomiarami – pominięto
}

// gromadzenie wartości pól niestandardowych wierceń i sondowań – pominięto

return ExportData.builder()
    .project(project)
    .holes(holes)
    .boreholesByHoleId(boreholesByHoleId)
    // pozostałe zagregowane kolekcje – pominięto
    .build();
}

```

Pliki wynikowe przyjmują następującą postać:

- CSV – każda sekcja danych zapisywana jest jako osobny plik, a całość pakowana do archiwum ZIP;
- XLSX – plik generowany przy użyciu biblioteki Apache POI, w którym każdej sekcji odpowiada oddzielny arkusz;
- PDF – graficzna karta otworu dla wybranego wiercenia, generowana przez odrębny moduł.

Eksport karty otworu polega na wygenerowaniu dokumentu w formacie A4 z profilem litologicznym wybranego wiercenia (Rysunek 96). Konfigurację udostępnia edytor w aplikacji internetowej (Rysunek 95), pozwalający na tymczasową modyfikację danych otworu bez ingerencji w same dane wiercenia. Wprowadzone korekty służą wyłącznie wydrukowi i nie są utrwalane w systemie. Profil litologiczny rysowany jest za pomocą HTML i CSS, z możliwością zmiany pionowej skali warstw oraz ograniczenia liczby stron dokumentu. Ze względu na ograniczone możliwości odwzorowania tak przygotowanego układu w bibliotece Apache PDFBox plik PDF generowany jest mechanizmem przeglądarek window.print.

5.12. Baza wiedzy

Moduł bazy wiedzy (w kodzie: KnowledgeBase) pełni w systemie podwójną rolę: platformy szkoleniowej dla pracowników oraz firmowej tablicy ogłoszeń. Gromadzi artykuły wraz z załącznikami, porządkowane przy użyciu kategorii i tagów (Rysunek 81 i Rysunek 82).

Artykuły dzielą się na firmowe, widoczne wyłącznie w obrębie danej firmy oraz globalne, dostępne dla wszystkich firm korzystających z systemu KRET. Użytkownicy mogą tworzyć jedynie artykuły firmowe. Wpisy globalne pełnią rolę ogłoszeń zespołu KRET.

Logikę modułu tworzą trzy serwisy:

- KnowledgeBaseArticleService – odpowiada za dodawanie, modyfikowanie i usuwanie artykułów, obsługę załączników oraz filtrowanie i wyszukiwanie wpisów;
- KnowledgeBaseCategoryService i KnowledgeBaseTagService – zarządzają kategoriami i tagami w zakresie standardowych operacji CRUD.

Utworzenie artykułu metodą `createArticle` (Listing 39) nie oznacza natychmiastowej publikacji. Wpis otrzymuje status szkicu, który można opublikować w dowolnym momencie metodą `publishArticle` (Listing 40). Formularz tworzenia artykułu przedstawia Rysunek 83.

Listing 39. Metoda `createArticle` odpowiedzialna za tworzenie artykułu w bazie wiedzy. Źródło: Opracowanie własne.

```
@Transactional
public KnowledgeBaseArticleResponse
createArticle(CreateKnowledgeBaseArticleRequest request) {
    User currentUser = securityContextService.getCurrentUser();
    accessControlService.requireCompanyMembership(currentUser, "tworzyć
artykuły bazy wiedzy");
    accessControlService.requireAdminOrDocumenter(currentUser);

    KnowledgeBaseArticle article = KnowledgeBaseArticle.builder()
        .title(request.getTitle())
        .content(request.getContent())
        .slug(generateSlug(request.getTitle()))
        .status(KnowledgeBaseArticleStatus.DRAFT)
        .company(currentUser.getCompany())
        .createdBy(currentUser)
        .updatedBy(currentUser)
        .build();

    if (request.getCategoryId() != null) {
        KnowledgeBaseCategory category =
categoryRepository.findById(request.getCategoryId())
            .orElseThrow(() -> new ResourceNotFoundException("Nie
znaleziono kategorii o ID: " + request.getCategoryId()));
        validateResourceBelongsToCompanyOrGlobal(category.getCompany(),
currentUser);
        article.setCategory(category);
    }

    if (request.getTagIds() != null && !request.getTagIds().isEmpty()) {
        Set<KnowledgeBaseTag> tags =
resolveAndValidateTags(request.getTagIds(), currentUser);
        article.setTags(tags);
    }

    article = articleRepository.save(article);
    log.info("Knowledge base article created with ID: {} by user: {}",
article.getId(), currentUser.getEmail());
    return mapToResponse(article);
}
```

```
}
```

Listing 40. Metoda publishArticle odpowiedzialna za publikacje uprzednio utworzonego artykułu. Źródło: Opracowanie własne.

```
@Transactional
public KnowledgeBaseArticleResponse publishArticle(Long articleId) {
    User currentUser = securityContextService.getCurrentUser();
    accessControlService.requireCompanyMembership(currentUser, "publikować
artykuły bazy wiedzy");
    accessControlService.requireAdminOrDocumenter(currentUser);

    KnowledgeBaseArticle article = findArticleByIdWithDetails(articleId);
    validateCompanyOwnership(article, currentUser);

    article.setStatus(KnowledgeBaseArticleStatus.PUBLISHED);
    if (article.getPublishedAt() == null) {
        article.setPublishedAt(Instant.now());
    }

    article = articleRepository.save(article);
    log.info("Knowledge base article published with ID: {} by user: {}",
articleId, currentUser.getEmail());
    return mapToResponse(article);
}
```

5.13. Statystyki i wizualizacje

Moduł statystyk dostarcza uprawnionym użytkownikom wgląd w postęp prac i stan projektów. Obejmuje przegląd najważniejszych metryk firmy, ranking geologów oraz szczegółowe statystyki projektowe, z możliwością filtrowania w przedziałach czasowych (7, 30 i 90 dni lub zakres własny). Uzupełnieniem modułu są kalendarz prac, prezentujący terminarz projektów, otworów i rezerwacji sprzętu wraz z nawigacją do powiązanych zasobów, oraz profil użytkownika umożliwiający edycję danych i awatara.

5.13.1. Statystyki projektowe

Interfejs statystyk stanowi komponent `ReportsPage` aplikacji internetowej, pełniący funkcję panelu wyboru okresu i prezentacji wyników (Rysunek 77). Dane dostarcza serwis `StatisticsService` poprzez dwie metody: `getOverview` (Listing 41), zwracającą zbiorczy raport statystyczny dla wybranego okresu, oraz `getGeologistRanking`, budującą ranking geologów na podstawie średniego czasu pracy, metrażu i liczby wykonanych otworów.

Raport zwracany przez metodę `getOverview` obejmuje:

- liczbę otworów ukończonych, realizowanych oraz oczekujących na weryfikację;
- liczbę aktywnych projektów i aktywnych geologów terenowych;
- sumę metrów bieżących wykonanych wierceń i sondowań;
- liczbę udokumentowanych warstw geologicznych i zarejestrowanych pomiarów sondowań;
- średni czas pracy i weryfikacji oraz średni metraż przypadający na otwór.

Ranking geologów zestawia dla każdego użytkownika liczbę wykonanych otworów, metraż wierceń oraz średni czas pracy. Na tej podstawie użytkownicy są szeregowani, a wynik prezentowany w aplikacji internetowej.

Listing 41. Fragment metody getOverview odpowiedzialnej za wyznaczenie statystyk firmy. Źródło: Opracowanie własne.

```
@Transactional(readOnly = true)
public StatisticsOverviewResponse getOverview(int periodDays) {
    User currentUser = securityContextService.getCurrentUser();
    accessControlService.requireAdminOrDocumenter(currentUser);

    Company company = currentUser.getCompany();
    LocalDate endDate = LocalDate.now();
    LocalDate startDate = endDate.minusDays(periodDays);
    Instant periodStart =
startDate.atStartOfDay(ZoneId.systemDefault()).toInstant();

    List<Project> projects = projectRepository.findByCompanyId(company.getId());
    List<Long> projectIds = projects.stream()
        .map(Project::getId).collect(Collectors.toList());

    List<Hole> allHoles = new ArrayList<>();
    for (Long projectId : projectIds) {
        allHoles.addAll(holeRepository.findByProjectId(projectId));
    }

    List<Hole> holesClosedInPeriod = allHoles.stream()
        .filter(h -> h.getClosedAt() != null
            && h.getClosedAt().isAfter(periodStart))
        .collect(Collectors.toList());

    Set<Long> activeGeologistIds = allHoles.stream()
        .filter(h -> h.getStatus() == Hole.HoleStatus.IN_PROGRESS
            || h.getStatus() == Hole.HoleStatus.ASSIGNED)
        .flatMap(h -> h.getAssignedGeologists().stream())
        .map(User::getId)
        .collect(Collectors.toSet());

    // analogiczne zliczenia aktywnych projektów oraz otworów w trakcie
    // i oczekujących na weryfikację - pominięto

    double totalBoreholeMeters = 0;
    int layersDocumented = 0;

    for (Hole hole : holesClosedInPeriod) {
        List<Borehole> boreholes =
boreholeRepository.findByHoleIdWithLayers(hole.getId());
        for (Borehole borehole : boreholes) {
            if (borehole.getDepth() != null) {
                totalBoreholeMeters += borehole.getDepth();
            }
            layersDocumented += borehole.getLayers().size();
        }
    }

    // analogiczna pętla agreguje metraż sondowań i liczbę pomiarów - pominięto

    Double avgWorkTimeHours = calculateAverageWorkTime(holesClosedInPeriod);
    Double avgReviewTimeHours = calculateAverageReviewTime(holesClosedInPeriod);
    Double avgTotalTimeHours = calculateAverageTotalTime(holesClosedInPeriod);

    return StatisticsOverviewResponse.builder()
        // budowa zagnieżdżonych sekcji odpowiedzi
        // (okres, podsumowanie, średnie) – pominięto
        .build();
}
```

5.13.2. Profil geologa

Moduł „mój profil” umożliwia użytkownikowi przegląd i modyfikację danych osobowych: imienia, nazwiska, numeru telefonu oraz zdjęcia profilowego. Administratorzy mają dodatkowo możliwość ustawienia logotypu firmy. Logika realizowana jest po stronie serwera przez `UserService`, którego najważniejsze metody w tym zakresie to:

- `getCurrentUserProfile` – pobranie profilu zalogowanego użytkownika;
- `updateCurrentUserProfile` – aktualizacja danych profilu;
- `uploadAvatar` – wgranie lub podmiana awatara;
- `uploadCompanyLogo` – zmiana logotypu firmy, dostępna wyłącznie dla administratorów.

Obsługę plików graficznych (awatarów i logotypów) zapewnia serwis `StorageService` opisany w podrozdziale 5.4. Serwer zapisuje i usuwa pliki strumieniowo oraz generuje adresy URL do ich pobierania.

5.13.3. Wizualizacje na kalendarzu

Komponent `CalendarWithEvents` prezentuje na kalendarzu zdarzenia związane z realizacją przedsięwzięć: terminy projektów, zaplanowane badania terenowe oraz okresy rezerwacji sprzętu (Rysunek 78). Poza wizualizacją kalendarz umożliwia nawigację do powiązanych zasobów.

Moduł zrealizowano w aplikacji internetowej z wykorzystaniem biblioteki `react-day-picker`, odpowiedzialnej za graficzną reprezentację kalendarza. Dane o terminach pochodzą z warstwy serwerowej i pobierane są za pośrednictwem serwisów `projectsService`, `holesService` oraz `reservationsService`.

Zdarzenia prezentowane są w dwóch formach: zdarzenia jednodniowe oznaczane są markerami, których liczba w danym dniu odpowiada liczbie zdarzeń, natomiast zdarzenia wielodniowe zakresami obejmującymi cały czas ich trwania. Wybranie dnia wyświetla pod kalendarzem listę przypisanych do niego zdarzeń.

5.14. System zarządzania sprzętem

System zarządzania sprzętem pozwala zdefiniować listę zasobów firmy – wiertnic, sond czy innego wyposażenia terenowego – bez ograniczeń co do ich rodzaju (Rysunek 87). Moduł umożliwia przegląd zasobów, ich rezerwowanie oraz śledzenie historii zmian statusów. Jego celem jest zapewnienie firmie kontroli nad posiadanym wyposażeniem. Funkcjonalność podzielono na dwie części: stronę sprzętu oraz stronę rezerwacji.

Po stronie aplikacji internetowej moduł tworzą dwa komponenty:

- `EquipmentPage` – odpowiada za prezentację zasobów firmy, ich dodawanie, edycję i usuwanie oraz podgląd historii statusów;
- `ReservationsPage` – odpowiada za zarządzanie rezerwacjami: tworzenie, przedłużanie, edycję i usuwanie, z możliwością wyświetlenia rezerwacji własnych lub całej firmy (Rysunek 90).

W warstwie serwerowej wykorzystywane są dwa serwisy:

- `EquipmentService` – realizuje operacje CRUD na sprzęcie oraz prowadzi historię statusów;
- `EquipmentReservationService` – realizuje operacje CRUD na rezerwacjach: przebieg tworzenia rezerwacji, obejmujący sprawdzanie poprawności dat, dostępności sprzętu i kolizji terminów, przedstawia metoda `createReservation` (Listing 42).

Listing 42. Metoda createReservation odpowiedzialna za tworzenie rezerwacji sprzętu. Źródło: Opracowanie własne.

```
@Transactional
public ReservationResponse createReservation(CreateReservationRequest request)
{
    User currentUser = securityContextService.getCurrentUser();
    accessControlService.requireCompanyMembership(currentUser, "tworzyć
rezerwacje");

    Equipment equipment = findEquipmentById(request.getEquipmentId());
    validateEquipmentCompanyAccess(currentUser, equipment);

    // Validate dates
    if (request.getStartDate().isAfter(request.getEndDate())) {
        throw new ValidationException("Data rozpoczęcia musi być przed data
zakończenia");
    }

    if (request.getStartDate().isBefore(LocalDate.now())) {
        throw new ValidationException("Data rozpoczęcia nie może być w
przeszłości");
    }

    // Check equipment availability - only RETIRED and OUT_OF_ORDER cannot be
reserved
    if (equipment.getStatus() == EquipmentStatus.RETIRED) {
        throw new ConflictException("Sprzęt został wycofany z eksploatacji");
    }
    if (equipment.getStatus() == EquipmentStatus.OUT_OF_ORDER) {
        throw new ConflictException("Sprzęt jest niesprawny i nie może być
zarezerwowany");
    }

    // Check for conflicting reservations
    if (reservationRepository.existsConflictingReservation(
        equipment.getId(), request.getStartDate(), request.getEndDate())) {
        throw new ConflictException("Sprzęt jest już zarezerwowany w tym
terminie");
    }

    log.info("Creating reservation for equipment {} from {} to {} by user: {}",
        equipment.getId(), request.getStartDate(), request.getEndDate(),
        currentUser.getEmail());

    // Get project if provided
    Project project = null;
    if (request.getProjectId() != null) {
        project = projectRepository.findById(request.getProjectId())
            .orElseThrow(() -> new ResourceNotFoundException("Nie
znaleziono projektu o ID: " + request.getProjectId()));
    }

    // Validate project access
    if
(!project.getCompany().getId().equals(currentUser.getCompany().getId())) {
        throw new AccessDeniedException("Nie masz dostępu do tego
projektu");
    }

    EquipmentReservation reservation = EquipmentReservation.builder()
        .equipment(equipment)
        .user(currentUser)
        .project(project)
        .startDate(request.getStartDate())
        .endDate(request.getEndDate())
```

```

        .status(ReservationStatus.ACTIVE)
        .notes(request.getNotes())
        .build();

    EquipmentReservation savedReservation =
reservationRepository.save(reservation);

    // Update equipment status if reservation starts today
    if (request.getStartDate().equals(LocalDate.now()) ||
request.getStartDate().isBefore(LocalDate.now())) {
        updateEquipmentStatusForReservation(equipment, EquipmentStatus.IN_USE,
currentUser,
            "Reserved by " + currentUser.getFirstName() + " " +
currentUser.getLastName());
    }

    log.info("Reservation created successfully with ID: {}",
savedReservation.getId());
    return mapToReservationResponse(savedReservation);
}

```

5.15. Architektura zorientowana na pracę offline i synchronizacja

Badania geologiczne prowadzone są w terenie, często w lokalizacjach pozbawionych zasięgu sieci komórkowej. Aplikacja mobilna KRET zaprojektowana została zgodnie z podejściem zorientowanym na pracę *offline*, w którym brak połączenia z siecią nie stanowi przeszkody w pracy i nie jest to sytuacja wyjątkowa. Mechanizm opiera się na trzech elementach: monitorowaniu stanu sieci, lokalnej pamięci podręcznej danych oraz kolejce operacji oczekujących na synchronizację.

Stan połączenia monitorowany jest przez kontekst `NetworkContext` wykorzystujący bibliotekę `netinfo`. Po wykryciu przywrócenia połączenia synchronizacja kolejki uruchamiana jest z opóźnieniem pół sekundy, co zapobiega przedwczesnemu rozpoczęciu wysyłki przy niestabilnym, migającym połączeniu.

Dane niezbędne do pracy w terenie (struktura projektów, otworów, pól niestandardowych, wzorców, wypełnianych formularzy) przechowywane są w lokalnej pamięci podręcznej zaimplementowanej w serwisie `offlineCache`, opartej na magazynie danych `AsyncStorage`. Każdy wpis pamięci podręcznej opatrzony jest znacznikiem ważności. Po upływie czasu 24 godzin dane uznawane są za nieaktualne i odświeżane przy najbliższym dostępie do sieci. Zdjęcia wykonane w trybie *offline* zapisywane są w lokalnym systemie plików za pośrednictwem mechanizmu `expo-file-system` i wysyłane na serwer po przywróceniu łączności.

Wszystkie modyfikacje danych wykonane w trybie *offline* trafiają do kolejki operacji w serwisie `offlineSync`. Każda operacja opisana jest typem oraz identyfikatorami kontekstu, co wykorzystywane jest do unikania duplikacji operacji w kolejce: jeśli ta sama jednostka danych jest wielokrotnie modyfikowana w trybie *offline*, w kolejce pozostaje wyłącznie jej najnowszy stan. Mechanizm ten przedstawia Listing 43. Po przywróceniu połączenia kolejka przetwarzana jest sekwencyjnie. W przypadku niepowodzenia pojedynczej operacji jej licznik prób jest zwiększany, a operacja ponawiana przy kolejnych synchronizacjach do osiągnięcia limitu trzech prób, po którym zostaje oznaczona jako nieudana i zgłoszona użytkownikowi co przedstawia Rysunek 46. Postęp synchronizacji oraz liczba operacji oczekujących prezentowane są na dedykowanym ekranie.

Listing 43. Dodawanie operacji do kolejki offline z deduplikacją. Źródło: Opracowanie własne.

```

async addToQueue(
    type: OfflineOperationType,
    projectId: number,

```

```

holeId: number,
payload: any,
layerId?: number
): Promise<OfflineOperation> {
  const queue = await this.getQueue();
  const now = new Date().toISOString();

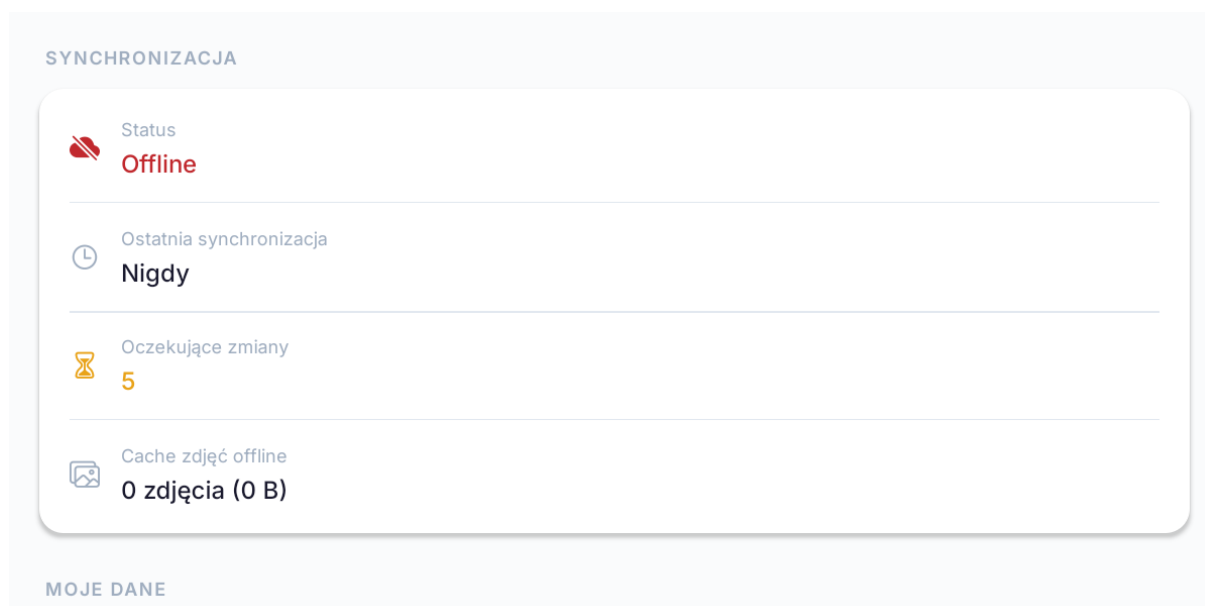
  const existingIndex = queue.findIndex(
    (op) =>
      op.type === type &&
      op.projectId === projectId &&
      op.holeId === holeId &&
      op.layerId === layerId &&
      op.status === 'PENDING'
  );

  if (existingIndex !== -1 && type !== 'UPLOAD_LAYER_IMAGE') {
    queue[existingIndex].payload = payload;
    queue[existingIndex].updatedAt = now;
    await this.saveQueue(queue);
    return queue[existingIndex];
  }

  const operation: OfflineOperation = {
    id: generateId(),
    type, projectId, holeId, layerId, payload,
    status: 'PENDING',
    retryCount: 0,
    createdAt: now,
    updatedAt: now,
  };

  queue.push(operation);
  await this.saveQueue(queue);
  return operation;
}

```



Rysunek 46. Ekran synchronizacji danych w aplikacji mobilnej. Źródło: Opracowanie własne.

5.16. Współdzielona biblioteka modułów

Projekt wieloplatformowy, w którym aplikacja internetowa i mobilna odwzorowują te same struktury danych, reguły walidacji oraz kontrakty komunikacji z serwerem, niesie ryzyko dryfu implementacyjnego. Ta sama reguła zapisana niezależnie w dwóch miejscach z czasem zaczyna się różnić. Aby temu zapobiec wspólną logikę aplikacji klienckich wydzielono do biblioteki @geokret/shared napisanej w języku TypeScript. Biblioteka obejmuje sześć modułów: definicję typów, stałe, reguły sprawdzania danych, klienta API wraz z obsługą uwierzytelniania, współdzielone haki React oraz funkcje pomocnicze. Reguły sprawdzania danych warstw litologicznych, konfiguracja typów litologicznych, sprawdzanie zdjęć panoramicznych oraz logika automatycznego zapisu omówione we wcześniejszych rozdziałach pochodzą właśnie z tej biblioteki.

5.16.1. Adapter trwałego przechowywania

Klient API biblioteki musi przechowywać tokeny uwierzytelniające między uruchomieniami aplikacji, jednak mechanizm trwałego przechowywania różni się między platformami. Aplikacja internetowa korzysta z magazynu przeglądarki, a mobilna aplikacja z magazynu AsyncStorage. Aby biblioteka pozostała niezależna od platformy, różnicę tę ukryto za interfejsem StorageAdapter, którego konkretną implementację dostarcza aplikacja kliencka.

Listing 44 przedstawia ten interfejs wraz z typowanym opakowaniem TypedStorage.

Listing 44. Interfejs StorageAdapter i opakowanie TypedStorage. Źródło: Opracowanie własne.

```
export interface StorageAdapter {
  setItem(key: string, value: string): Promise<void>;
  getItem(key: string): Promise<string | null>;
  removeItem(key: string): Promise<void>;
  clear(): Promise<void>;
  multiGet?(keys: string[]): Promise<Array<[string, string | null]>>;
}

export class TypedStorage {
  constructor(private adapter: StorageAdapter) {}

  async storeData<T>(key: string, value: T): Promise<boolean> {
    try {
      const jsonValue = JSON.stringify(value);
      await this.adapter.setItem(key, jsonValue);
      return true;
    } catch (error) {
      console.error(`Error storing data for key ${key}:`, error);
      return false;
    }
  }

  async getData<T>(key: string): Promise<T | null> {
    try {
      const jsonValue = await this.adapter.getItem(key);
      return jsonValue != null ? JSON.parse(jsonValue) : null;
    } catch (error) {
      console.error(`Error retrieving data for key ${key}:`, error);
      return null;
    }
  }
}
```

Klasa TypedStorage opakowuje surowy adapter, dodając serializację formatu JSON oraz typowanie generyczne, dzięki czemu warstwy wyższe operują na obiektach typowanych. Takie rozwiązanie pozwala umieścić całą logikę klienta API w bibliotece współdzielonej, pozostawiając

aplikacjom klienckim jedynie dostarczenie kilkuliniowej implementacji adaptera właściwej dla danej platformy.

5.16.2. Przechwytywanie odświeżania tokenu

Wygaśnięcie tokenu dostępowego po 15 minutach (podrozdział 5.3) wymaga przezroczystego dla użytkownika odświeżania sesji. Realizuje to mechanizm przechwytyjący odpowiedzi biblioteki Axios, reagujący na kod HTTP 401. Istotnym problemem jest sytuacja, w której wiele żądań wykonywanych równolegle otrzymuje jednocześnie kod błędu HTTP 401. Naiwna implementacja podjęłaby wówczas kilka równoczesnych prób odświeżenia, z których wszystkie poza pierwszą zakończyłyby się błędem. Rozwiązaniem jest współdzielenie jednej obietnicy odświeżenia między wszystkie oczekujące żądania, co przedstawia Listing 45.

Listing 45. Współdzielona obietnica odświeżenia tokenu. Źródło: Opracowanie własne.

```
let isRefreshing = false;
let refreshPromise: Promise<boolean> | null = null;

apiClient.axios.interceptors.response.use(
  (response) => response,
  async (error) => {
    const originalRequest = error.config;
    if (
      error.response?.status === 401 &&
      !originalRequest._retry &&
      !isAuthEndpoint(originalRequest.url)
    ) {
      originalRequest._retry = true;

      if (!isRefreshing) {
        isRefreshing = true;
        refreshPromise = doRefresh().finally(() => {
          isRefreshing = false;
        });
      }

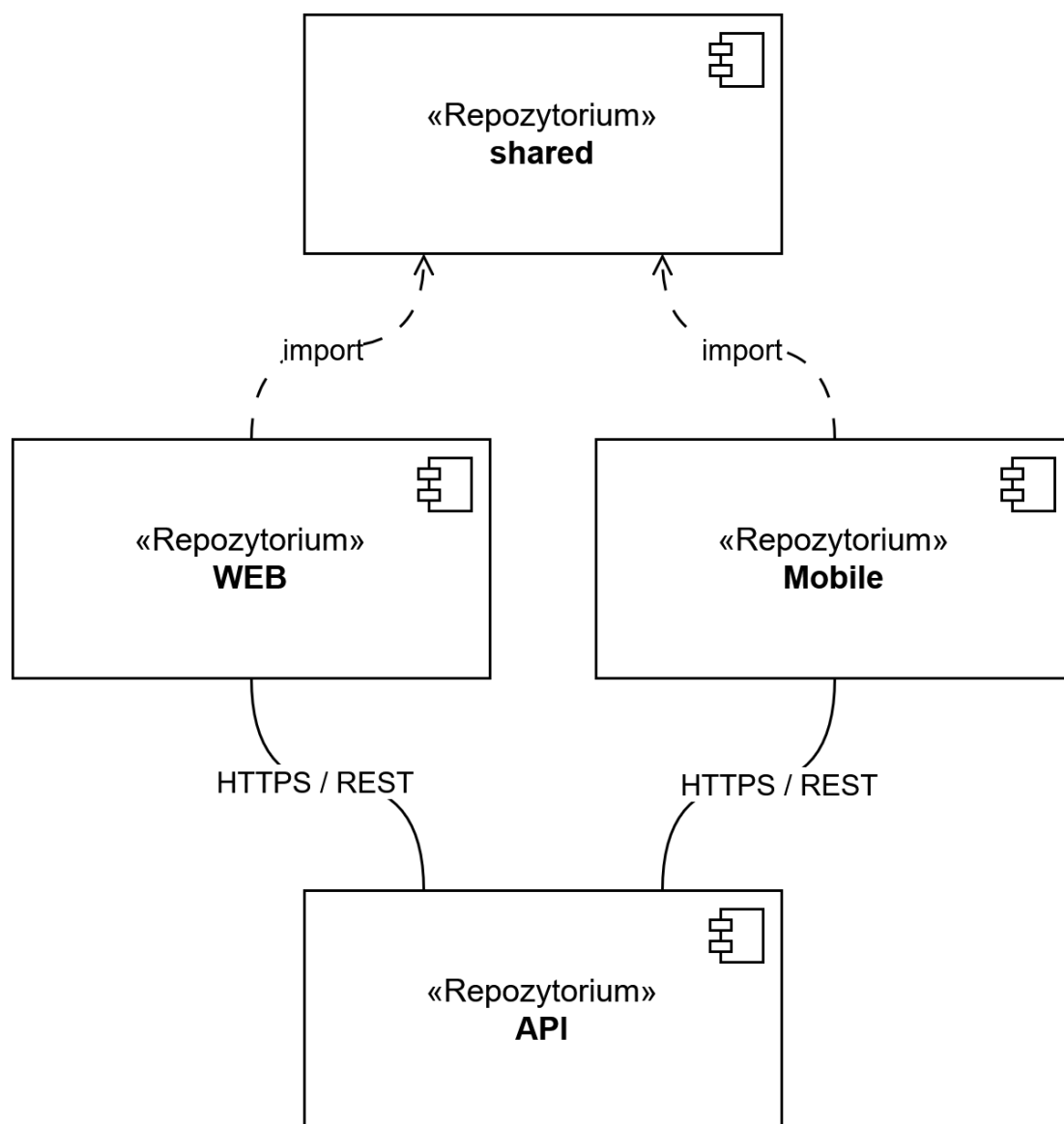
      const refreshed = await refreshPromise;
      if (refreshed) {
        return apiClient.axios(originalRequest);
      }
    }
    return Promise.reject(error);
  }
);
```

Pierwsze żądanie, które napotka kod błędu HTTP 401, rozpoczyna odświeżenie i zapisuje jego obietnicę w zmiennej `refreshPromise`. Każde kolejne żądanie, zamiast rozpoczynać własne odświeżenie oczekuje na tą samą obietnicę. Po jej spełnieniu wszystkie żądania ponawiane są z nowym tokenem dostępowym. Warunek pomijający końcówki uwierzytelniania zapobiega zapętleniu, w którym nieudane odświeżenie samo wyzwałoby próbę odświeżenia.

5.16.3. Dystrybucja i wersjonowanie

Biblioteka współdzielona nie jest publikowana w publicznym rejestrze npm, lecz instalowana bezpośrednio z repozytorium Git, ze wskazaniem konkretnego znacznika wersji. W pliku zależności aplikacji klienckiej referencja przyjmuje postać wskazującą adres repozytorium oraz znacznik (przykładowo `git+https://github.com/geokret/shared.git#v1.43.0`). Powiązanie wersji ze znacznikiem Git gwarantuje, że dana wersja aplikacji klienckiej zawsze korzysta z dokładnie tej samej, niezmienniej wersji biblioteki, eliminując ryzyko niekontrolowanej zmiany wspólnej logiki. Aktualizacja biblioteki w aplikacji klienckiej wymaga świadomej zmiany znacznika i ponownej

instalacji zależności, co czyni proces przewidywalnym. Relacje między czterema repozytoriami systemu przedstawia Rysunek 47.



Rysunek 47. Zależności między repozytoriami systemu KRET. Źródło: Opracowanie własne.

6. Prezentacja systemu

Niniejszy rozdział prezentuje gotowy system KRET z perspektywy użytkownika końcowego. Celem obecnego rozdziału jest zilustrowanie zrealizowanych funkcjonalności w działaniu.

System udostępniany jest za pośrednictwem dwóch aplikacji klienckich o rozłącznym przeznaczeniu. Aplikacja internetowa kierowana jest do administratorów oraz dokumentatorów, a aplikacja mobilna skierowana do geologów terenowych. Przy każdym rzucie ekranu wskazano, której aplikacji dotyczy.

6.1. System zaproszeń i logowanie

System KRET przeznaczony jest dla pracowników konkretnych firm geologicznych, dlatego zrezygnowano z otwartej rejestracji na rzecz mechanizmu zaproszeń opisanego w podrozdziale 5.3.5. Osoba niezalogowana, określana jako gość, po wejściu na stronę internetową widzi ekran startowy który przedstawia Rysunek 48. Ekran ten pełni funkcję informacyjną.



Rysunek 48. Ekran startowy aplikacji internetowej dla niezalogowanego użytkownika. Źródło: Opracowanie własne.

Zaproszenie jest wysyłane przez administratora lub dokumentatora z poziomu aplikacji internetowej. Po jego wysłaniu zapraszana osoba otrzymuje wiadomość email zawierającą odnośnik aktywacyjny o ograniczonym terminie ważności. Kliknięcie odnośnika przenosi Gościa do formularza rejestracji, który przedstawia Rysunek 49. Formularz wymaga podania danych osobowych i ustawienia hasła. Rola w systemie zostaje przypisana automatycznie na podstawie zaproszenia.

Dokończ rejestrację

Zaproszenie od Mieczysław Czosnek dla
s15154+dokumentator1@pjwstk.edu.pl

Bartosz Kowalski
s15154+dokumentator1@pjwstk.edu.pl **Dokumentator**

USTAW DANE LOGOWANIA

Imię: Nazwisko:

Hasło: Potwierdź hasło:

OPCJONALNIE

Telefon:

Utwórz konto →

Tworząc konto, akceptujesz regulamin i politykę prywatności.

Dołącz do zespołu badawczego

Twój dostęp do profesjonalnych narzędzi geologicznych jest o krok stąd.

KRET

Rysunek 49. Formularz rejestracji konta otwierany po kliknięciu odnośnika z zaproszenia. Źródło: Opracowanie własne.

Po pomyślnym założeniu konta użytkownik przechodzi do ekranu logowania, który przedstawia Rysunek 50.

Login

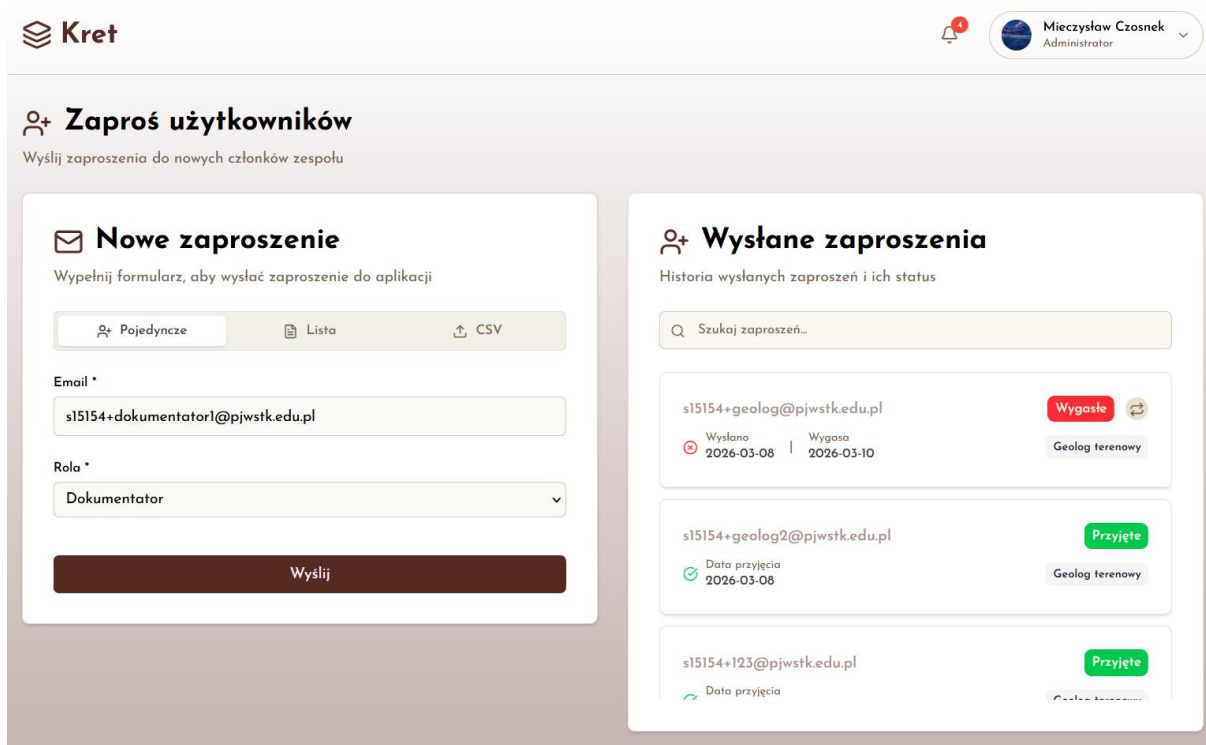
Witaj ponownie!

Login / Email

Hasło

Zaloguj się

Rysunek 50. Ekran logowania do aplikacji internetowej. Źródło: Opracowanie własne.



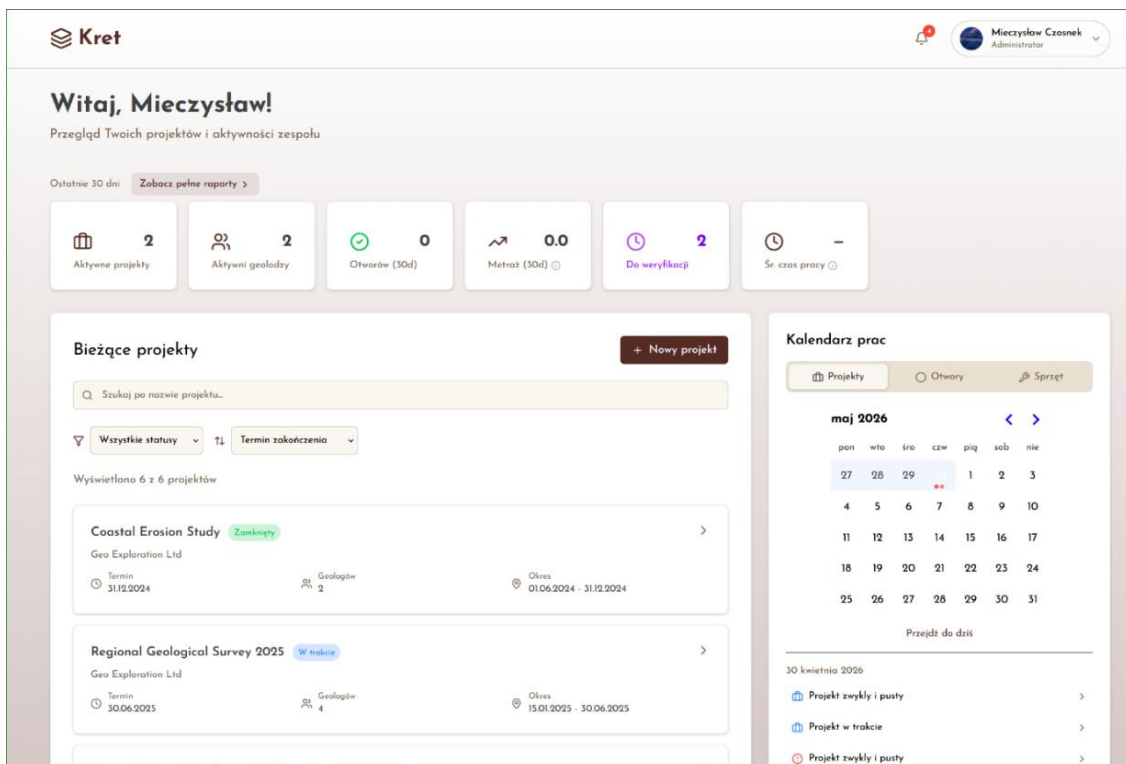
Rysunek 51. Panel zarządzania zaproszeniami w aplikacji internetowej. Źródło: Opracowanie własne.

Administrator oraz dokumentator dysponują panelem zarządzania zaproszeniami, który przedstawia Rysunek 51. Panel udostępnia listę zaproszeń wraz z aktualnym stanem.

6.2. Zarządzanie projektami

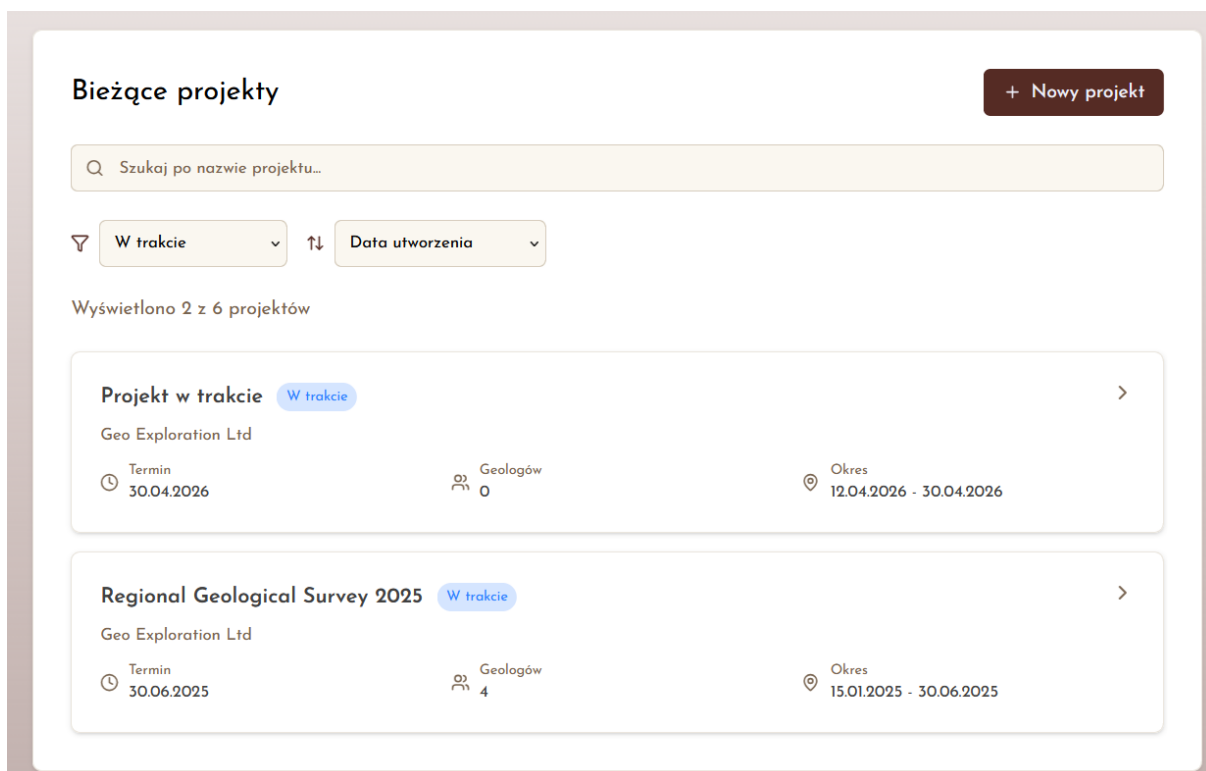
Po zalogowaniu administrator lub dokumentator trafia do panelu głównego stanowiącego punkt wyjścia do pozostałych funkcji systemu. Niniejszy podrozdział przedstawia panel główny oraz operacje związane z cyklem życia projektu.

Panel główny, który przedstawia Rysunek 52, gromadzi w jednym miejscu skrócone statystyki i listę prowadzonych projektów. Dzięki temu użytkownik uzyskuje przegląd bieżącego stanu prac firmy bezpośrednio po zalogowaniu.



Rysunek 52. Panel główny aplikacji internetowej. Źródło: Opracowanie własne.

Pełną listę projektów wraz z mechanizmem filtrowania przedstawia Rysunek 53. Filtrowanie umożliwia ograniczenie widoku projektów.



Rysunek 53. Lista projektów z mechanizmem filtrowania. Źródło: Opracowanie własne.

Tworzenie nowego projektu odbywa się za pośrednictwem formularza, który przedstawia Rysunek 54. Po określeniu podstawowych właściwości projektu i zapisaniu go, projekt staje się dostępny na liście i może być dalej konfigurowany w widoku szczegółowym.

Kret

← Powrót

Nowy projekt

Nazwa projektu *

Depozyt minerałów

Opis projektu

Na północ od Warszawy w powiecie Serockim wykryto depozyt minerałów.

Status

Przygotowanie

Data rozpoczęcia

01.06.2026

Data zakończenia

30.06.2026

Przypisz geologów

☐ Paweł Wronek (pawel.wronek@geo-exploration.com)

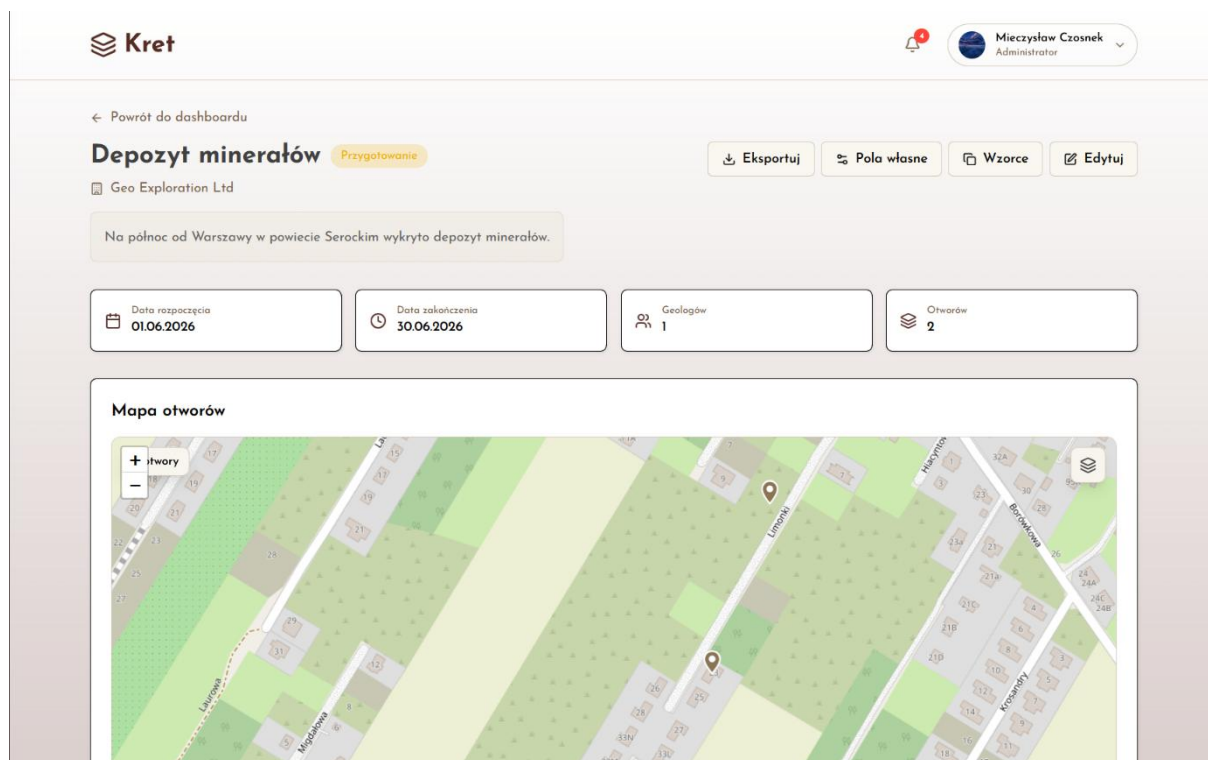
☒ Krzysztof Krawczyk (krzysztof.krawczyk@geo-exploration.com)

Utwórz projekt Anuluj

Rysunek 54. Formularz tworzenia nowego projektu. Źródło: Opracowanie własne.

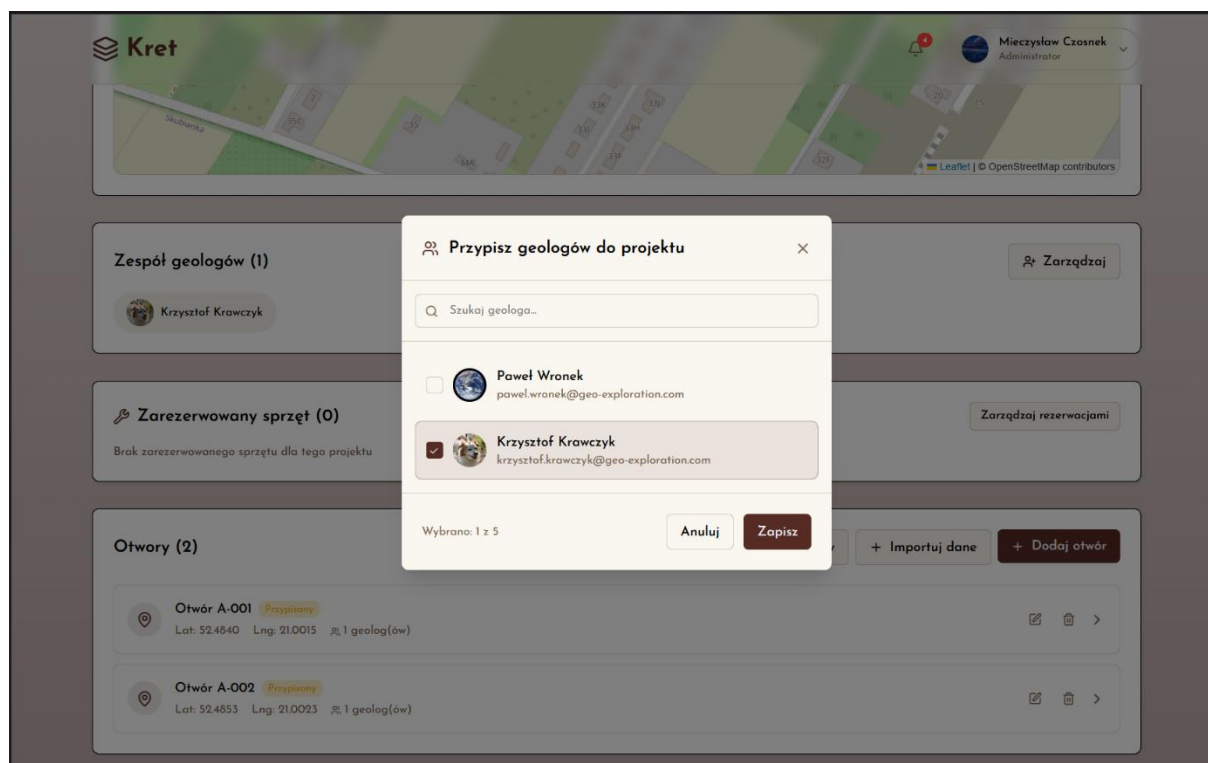
Widok szczegółowy, który przedstawia Rysunek 55, stanowi centralne miejsce zarządzania zleceniem. Z jego poziomu dostępne są operacje:

- edycja właściwości projektu oraz jego ustawień;
- zarządzanie polami niestandardowymi oraz wzorcami formularzy obowiązującymi w projekcie;
- przypisywanie geologów oraz rezerwacja sprzętu na potrzeby prac;
- zarządzanie otworami badawczymi, w tym ich tworzenie, import oraz wydruk kart;
- eksport danych projektu w wybranym formacie.



Rysunek 55. Widok szczegółowy projektu w aplikacji internetowej. Źródło: Opracowanie własne.

Przypisywanie geologów do projektu przedstawia Rysunek 56. Przypisanie to stanowi warunek wstępny do przypisania otworu do danego geologa.

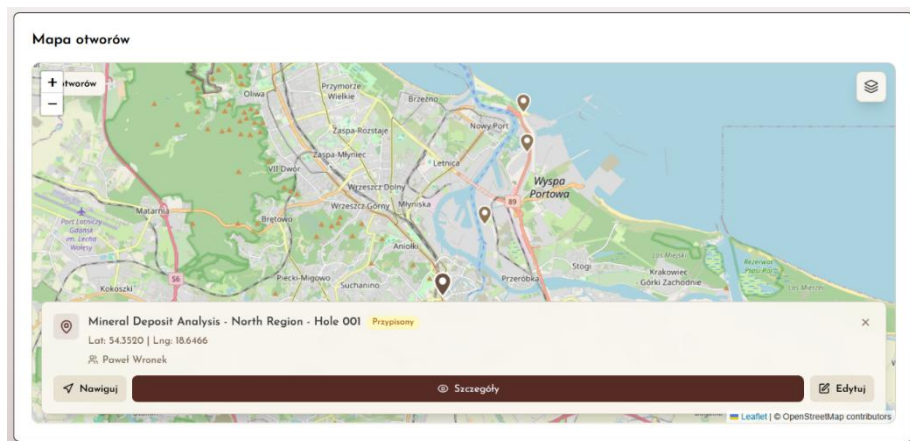


Rysunek 56. Przypisywanie geologów do projektu. Źródło: Opracowanie własne.

6.3. Mapa otworów i zlecenia terenowe

Centralnym punktem prezentacji zleceń terenowych w systemie KRET jest mapa. Niniejszy podrozdział przedstawia wizualizację otworów na mapie, proces tworzenia nowego zlecenia oraz odpowiadające im widoki w aplikacji mobilnej.

Mapę projektu z naniesionymi otworami badawczymi przedstawia Rysunek 57.



Rysunek 57. Mapa projektu z naniesionymi otworami badawczymi w aplikacji internetowej. Źródło: Opracowanie własne.

Tworzenie nowego otworu przedstawia Rysunek 58. W formularzu uzupełnia się szczegóły otworu oraz wybiera pozycję otworu na osadzonej mini-mapie lub podając jego współrzędne. Dane administracyjne są automatycznie uzupełniane na podstawie współrzędnych.

Rysunek 58. Formularz tworzenia nowego otworu z mini-mapą do wskazania lokalizacji. Źródło: Opracowanie własne.

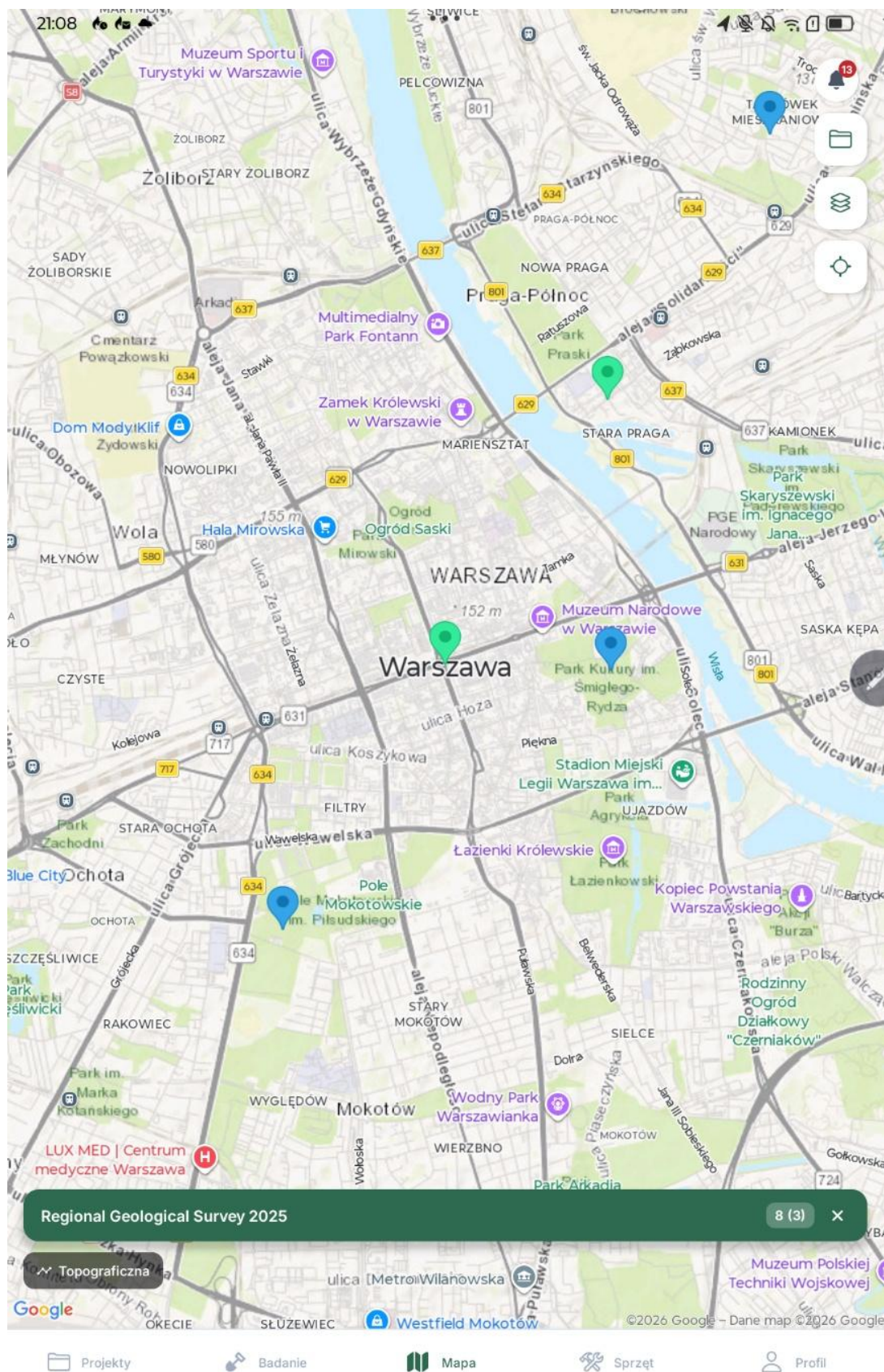
Widok szczegółowy otworu w aplikacji internetowej przedstawia Rysunek 59. Z poziomu tego widoku dokumentator zarządza przydziałami oraz przechodzi do weryfikacji wprowadzonych danych.

The screenshot displays the 'Coastal Erosion Study - Hole 001' page in the Kret application. The page is managed by 'Mieczysław Czołnek, Administrator'. It features several sections:

- Project Overview:** Includes a link to 'Powrót do projektu: Coastal Erosion Study', the project name 'Coastal Erosion Study - Hole 001', a 'Zamknięty' (Closed) status, and an 'Edytuj' (Edit) button.
- Location:** Displays coordinates '53.428500, 14.552800' and a link to 'Otwórz w Google Maps'.
- Closed Date:** Shows 'Zamknięto 09.01.2026'.
- Assigned Geologists:** Lists 'Krzysztof Krawczyk' with a 'Zarządzaj' (Manage) button.
- Assigned Documenter:** Lists 'Beata Kozidrak' with a 'Zmień' (Change) button.
- Borehole Details:**
 - Odwiert (Borehole):** Includes 'Edytuj odwiert' (Edit borehole) and 'Usuń odwiert' (Delete borehole) buttons.
 - Depth:** 'Głębokość: 12.5 m'.
 - Diameter:** 'Średnica: 150 m'.
 - Layers:** 'Warstwy: 3'.
 - Geological Layers:** A list showing '0.00 - 0.30 m' with 'Humus' and 'Humus, ciemnobrzązowy'.

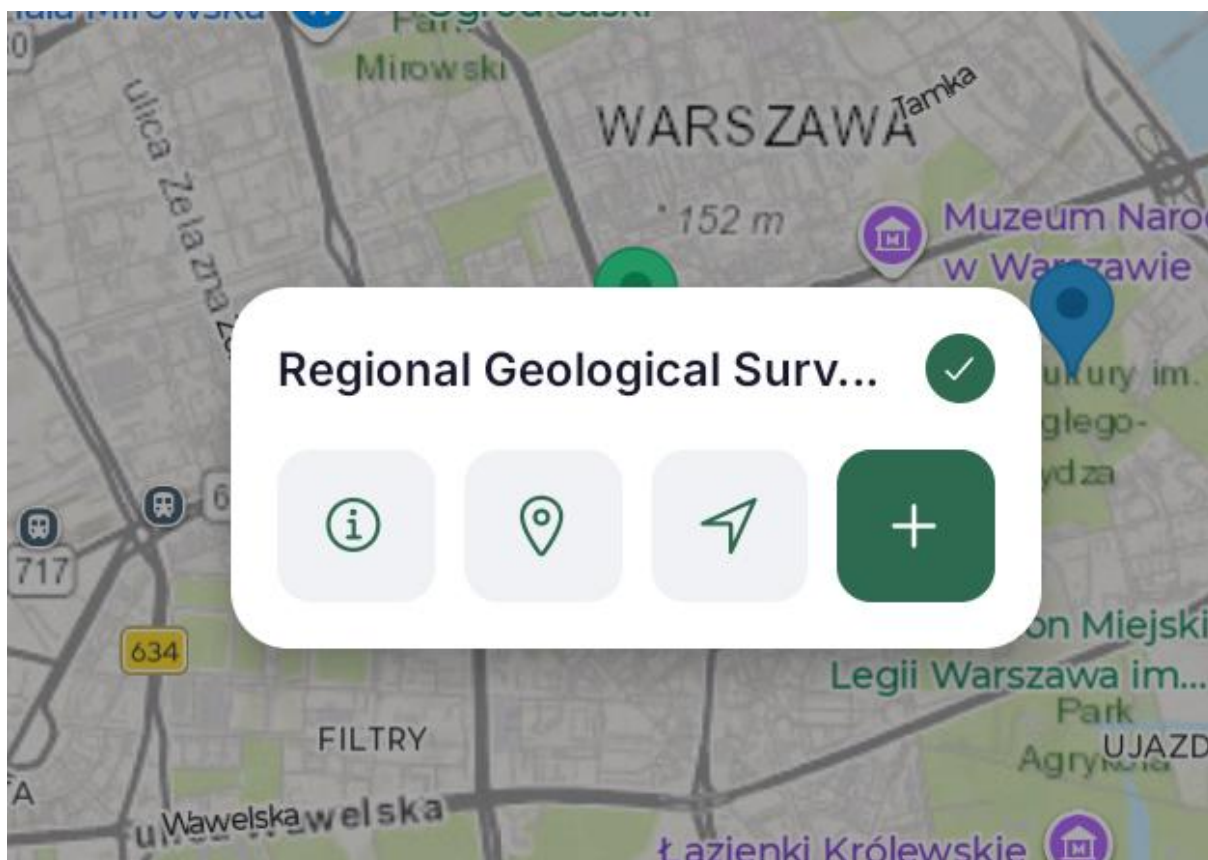
Rysunek 59. Widok szczegółowy otworu w aplikacji internetowej. Źródło: Opracowanie własne.

Geolog terenowy korzysta z mapy w aplikacji mobilnej, którą przedstawia Rysunek 60. Po wybraniu projektu mapa prezentuje wszystkie jego otwory. Mapa udostępnia pięć typów map do wyboru: mapę OpenStreetMap, mapę topograficzną OpenTopoMap, zdjęcia satelitarne Esri, mapę topograficzną Esri oraz mapę w ciemnym wariantcie.



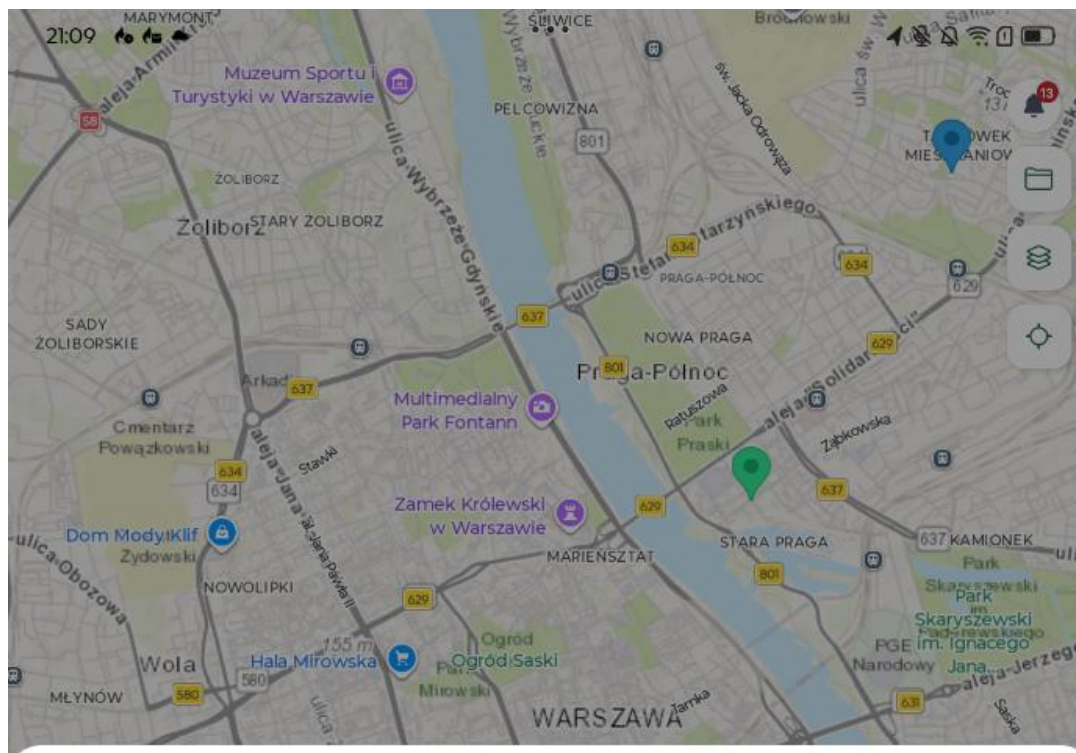
Rysunek 60. Mapa otworów w projekcie w aplikacji mobilnej. Źródło: Opracowanie własne.

Wybór znacznika otworu otwiera podręczne menu szybkich akcji, które przedstawia Rysunek 61. Menu udostępnia cztery operacje: wyświetlanie szczegółów otworu, pokazanie lokalizacji na systemowej mapie, uruchomienie nawigacji w systemowej mapie oraz przejście do badania.



Rysunek 61. Podręczne menu szybkich akcji otworu w aplikacji mobilnej. Źródło: Opracowanie własne.

Wybór opcji szczegółów otwiera pełny widok otworu, który przedstawia Rysunek 62. Prezentuje on dane otworu oraz udostępnia trzy akcje – pokazanie lokalizacji otworu w mapie systemowej, uruchomienie nawigacji w systemowej mapie oraz przeprowadzenie badania.



Szczegóły otworu

Regional Geological Survey 2025 - Hole 001

DO SPRAWDZENIA

Przypisany do Ciebie

Lokalizacja

52.229700, 21.012200

Projekt

Regional Geological Survey 2025

Geologowie

Paweł Wronek

Krzysztof Krawczyk



Na mapie



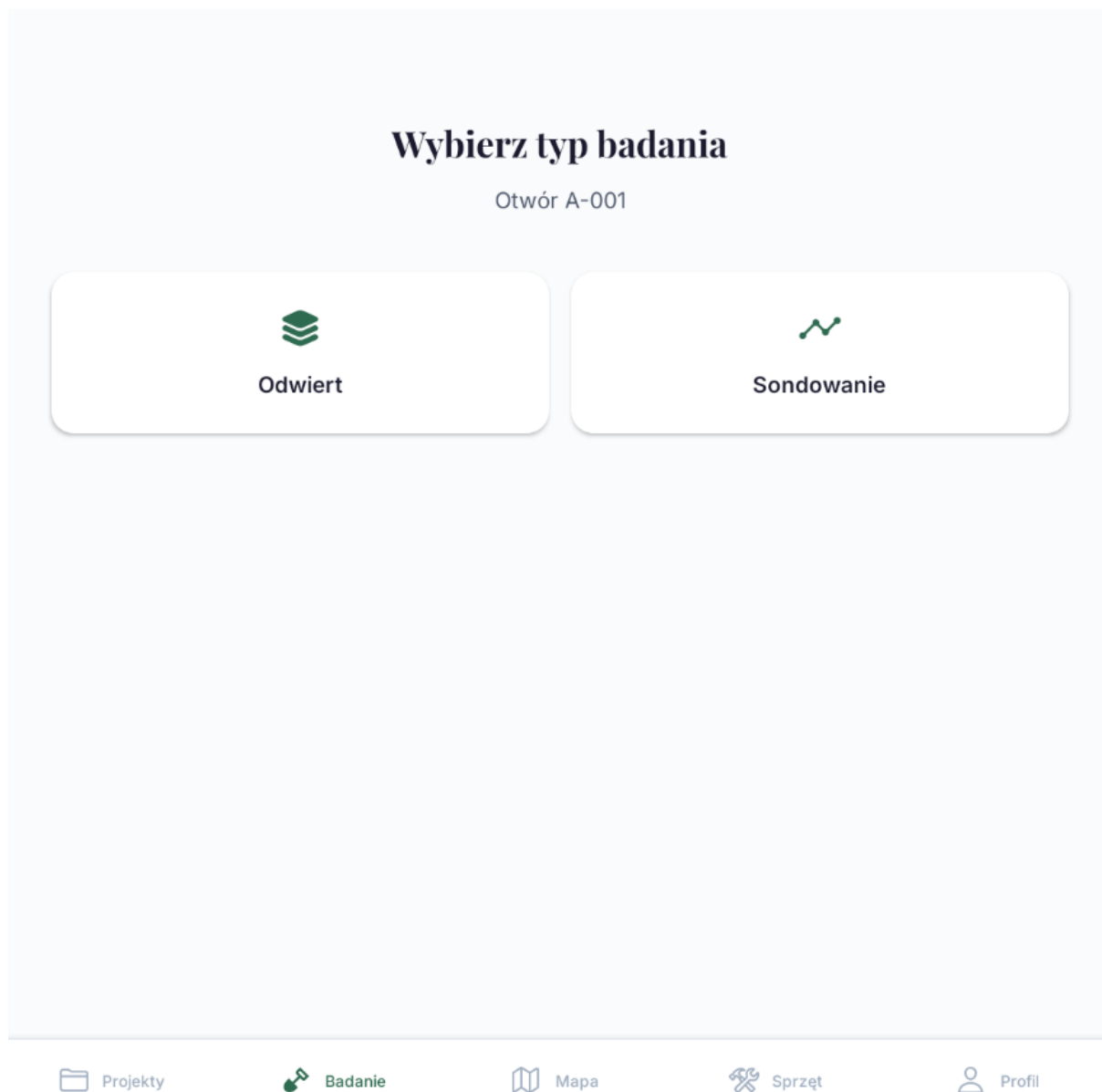
Nawiguj



Przeprowadź badanie

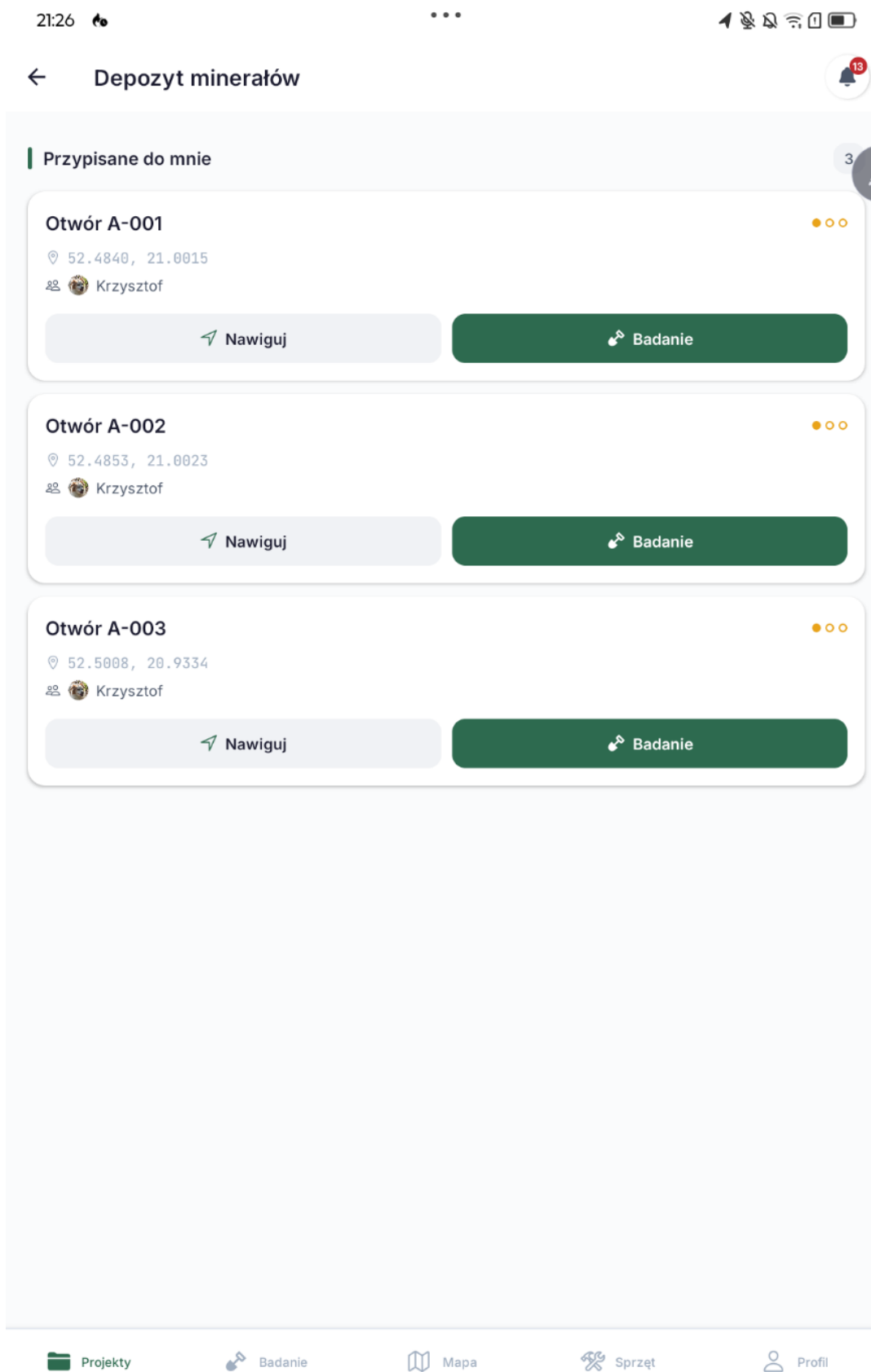
Rysunek 62. Widok szczegółów otworu w aplikacji mobilnej. Źródło: Opracowanie własne.

Wybór akcji przeprowadzenia badania przenosi geologa do wyboru typu badania, który przedstawia Rysunek 63. Ponieważ pojedynczy otwór może być dokumentowany jako wiercenie (odwiert) albo jako sondowanie dynamiczne, na tym etapie geolog wskazuje rodzaj badania, co determinuje dalszy formularz omówiony w podrozdziale 6.4.



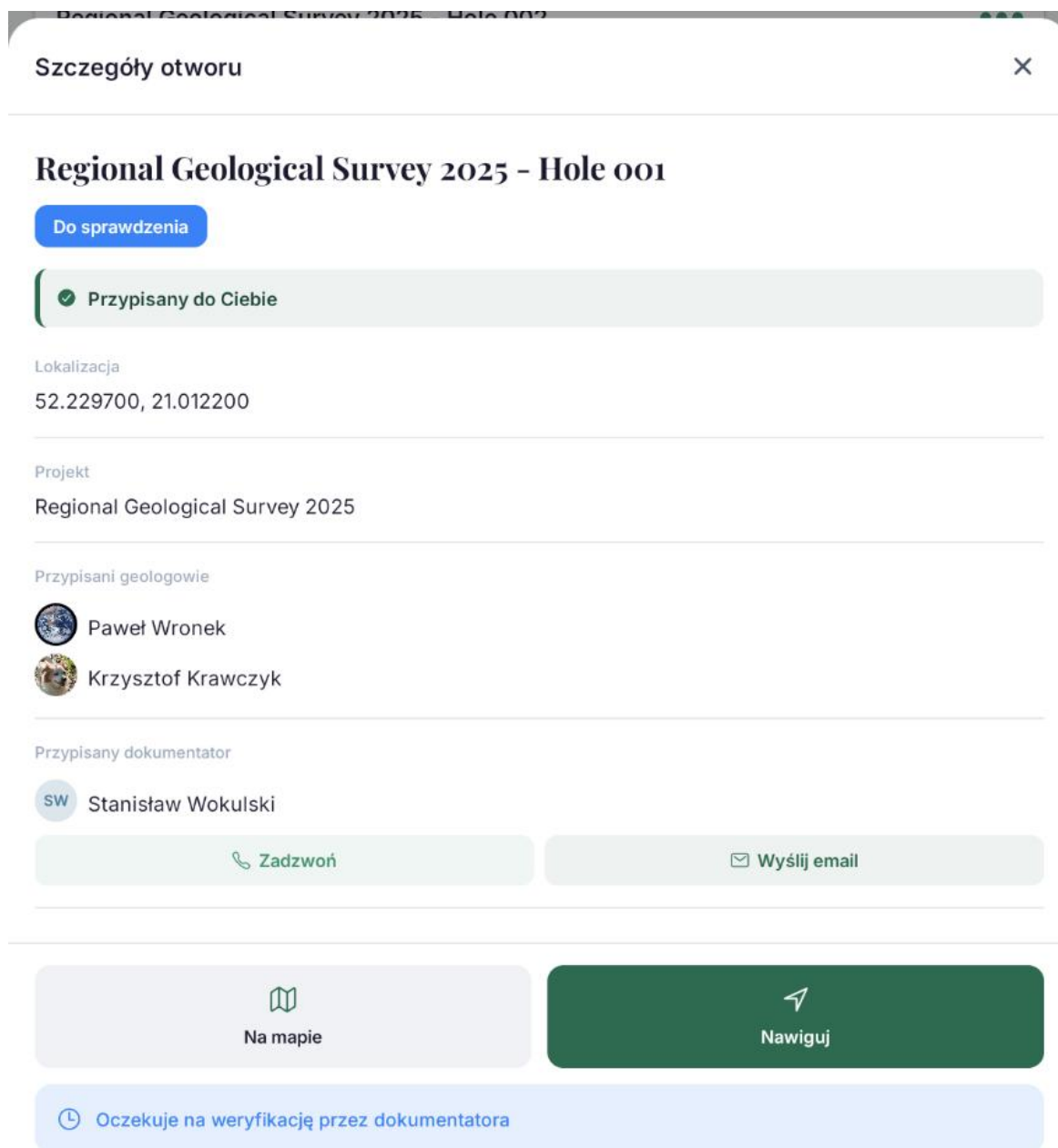
Rysunek 63. Wybór typu badania w aplikacji mobilnej. Źródło: Opracowanie własne.

Alternatywną drogą dostępu do otworów, niezależną od mapy, jest lista otworów projektu, którą przedstawia Rysunek 64. Geolog wybiera projekt z listy przypisanych projektów, po czym otrzymuje wykaz jego otworów podzielony na dwie grupy: otwory przypisane do niego oraz pozostałe.



Rysunek 64. Lista otworów projektu w aplikacji mobilnej. Źródło: Opracowanie własne.

Wybór otworu z listy otwiera jego szczegóły, które przedstawia Rysunek 65. Widok ten, oprócz danych prezentowanych na mapie, zawiera również dane przypisanego dokumentatora wraz z akcjami umożliwiającymi bezpośredni kontakt telefoniczny lub mailowy, co usprawnia komunikację geologa z osobą weryfikującą jego pracę.



Szczegóły otworu

Regional Geological Survey 2025 - Hole 001

Do sprawdzenia

✓ Przypisany do Ciebie

Lokalizacja
52.229700, 21.012200

Projekt
Regional Geological Survey 2025

Przypisani geolodzy

 Paweł Wronek

 Krzysztof Krawczyk

Przypisany dokumentator

 Stanisław Wokulski

 Zadzwoń

 Wyślij email

 Na mapie

 Nawiguj

 Oczekuje na weryfikację przez dokumentatora

Rysunek 65. Szczegóły otworu z danymi dokumentatora w aplikacji mobilnej. Źródło: Opracowanie własne.

6.4. Formularze badań geologicznych

Formularze badań stanowią najważniejszy element pracy terenowej i centralną funkcję aplikacji mobilnej. W systemie KRET prowadzone są dwa rodzaje badań: wiercenie oraz sondowanie

dynamiczne. Oba formularze mogą zostać uzupełnione o pola niestandardowe. Niezależnie od typu badania aplikacja mobilna automatycznie zapisuje wprowadzane dane, chroniąc je przed utratą oraz umożliwia pełną pracę w trybie *offline*.

6.4.1. Formularz wiercenia

Formularz wiercenia służy do dokumentowania profilu litologicznego otworu rozpoznawanego za pomocą wiertnicy. Jego pełny widok przedstawia Rysunek 66. Formularz podzielony jest na sekcję podstawowych informacji o badaniu oraz sekcję warstw geologicznych. W górnej części ekranu prezentowany jest wskaźnik stanu zapisu, informujący o trwającym zapisie, jego powodzeniu lub wystąpieniu błędu.

22:48

← Odwiert

Regional Geological Survey 2025 - Hole 002

Auto-zapis

Podstawowe informacje

Nazwa badania

Regional Geological Survey 2025 - Hole 002

Głębokość (m)

25

Średnica (m)

8

Warstwy geologiczne

Tryb zdjęć 360°

PR 0.00 - 1.00 m

Piasek pylasty

1 zdjęcie

Aparat

Galeria

1.00 m

Notatki

Piach

Projekty Badanie Mapa Sprzęt Profil

Rysunek 66. Formularz wiercenia w aplikacji mobilnej. Źródło: Opracowanie własne.

Wypełnianie formularza rozpoczyna się zwykle od zastosowania wzorca formularza, którego okno wyboru przedstawia Rysunek 67. Wzorzec wczytuje predefiniowany zestaw wartości pól formularza, przyspieszając dokumentację serii badań o zbliżonym profilu.

Wybierz wzorzec ×

Moj wzorzec Projekt 8 pól ○ ▼

Wzorzec dla projektu Projekt 9 pól ○ ▲

POLA STANDARDOWE (3)

Głębokość = 25 Średnica = 8 Uwagi = Piach

POLA NIESTANDARDOWE (6)

Warunki atmosferyczne = Deszczowo Numer zlecenia = 12345678 Temperatura gruntu (°C) = 50

Czy wykryto kontaminację? = Nie Numer próbki laboratoryjnej = 1234hhht Głębokość poboru próbki (m) = 25

Krzysztof Krawczyk

Zastosuj wzorzec

Rysunek 67. Okno wyboru wzorca formularza w aplikacji mobilnej. Źródło: Opracowanie własne.

Zasadniczą częścią formularza jest uporządkowana lista warstw geologicznych. Geolog dodaje kolejne warstwy, określając dla każdej z nich głębokość początkową i końcową, typ litologiczny oraz pozostałe parametry gruntu. Okno edycji pojedynczej warstwy przedstawia Rysunek 68.

Edytuj warstwę

×

Od (m) *

0

Do (m) *

1

Opis litologiczny

Bardzo pylący

Typ litologiczny

Ptt Piasek pylasty

▼

Mięszość (m)

12

Wilgotność

1

Plastyczność

1

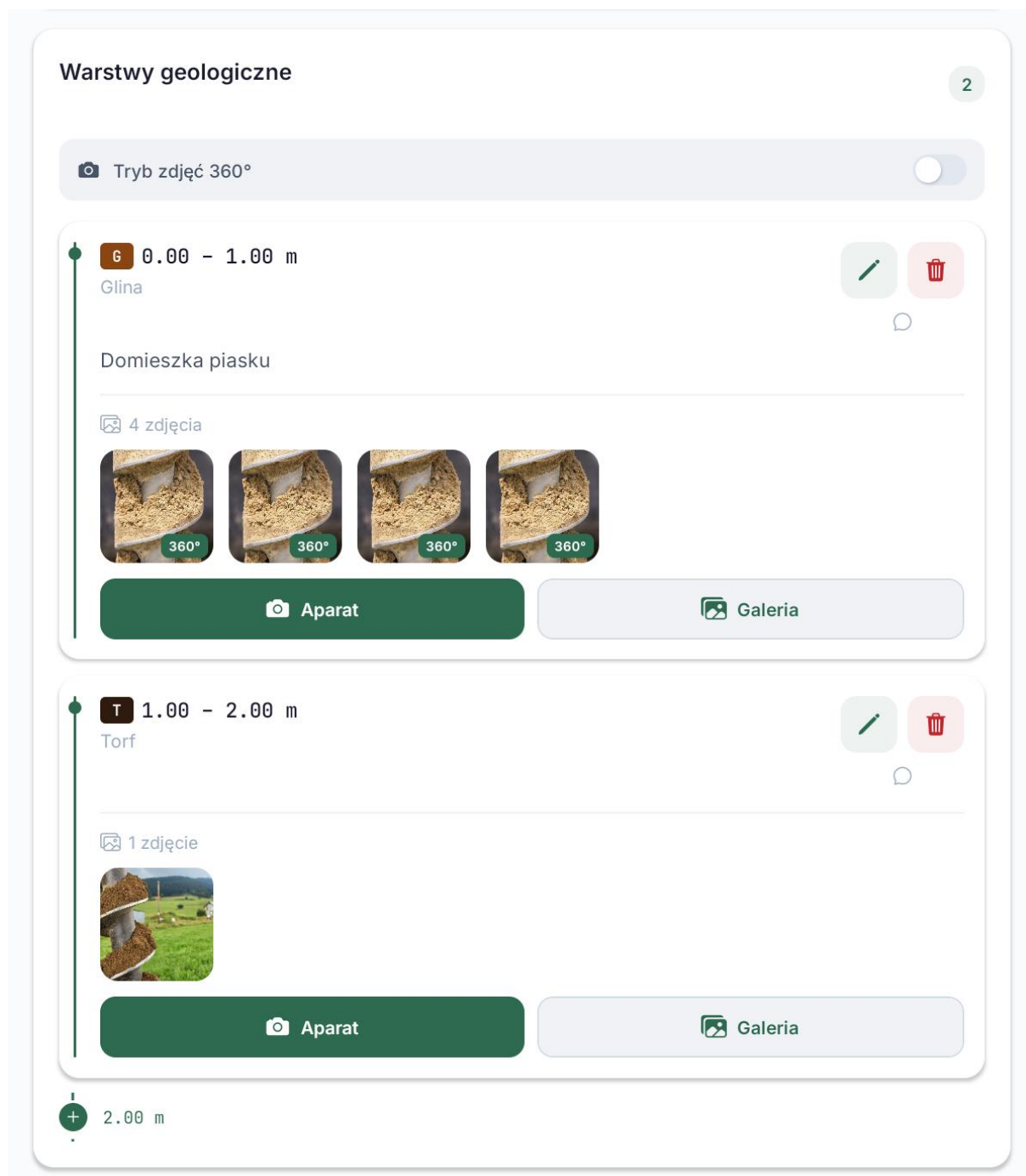
☒  Zwierciadło wody

Anuluj

Zapisz

Rysunek 68. Okno dodawania i edycji warstwy litologicznej w aplikacji mobilnej. Źródło: Opracowanie własne.

Dokumentację fotograficzną warstwy przedstawia Rysunek 69. Geolog wykonuje zdjęcia aparatem urządzenia lub wybiera je z galerii, a system wiąże je z konkretną warstwą litologiczną. Oprócz pojedynczych zdjęć dostępny jest tryb zdjęć 360 stopni, w którym geolog wykonuje serię co najmniej czterech fotografii dokumentujących otoczenie punktu badawczego; zdjęcia takie oznaczane są w galerii znacznikiem „360°”.



Rysunek 69. Dokumentacja fotograficzna warstwy litologicznej w aplikacji mobilnej. Źródło: Opracowanie własne.

6.4.2. Formularz sondowania

Formularz sondowania służy do dokumentowania badania sondą dynamiczną i przedstawia go Rysunek 70. Zasadniczą strukturą danych jest uporządkowana lista pomiarów, w której każdy wpis opisuje głębokość oraz liczbę uderzeń sondy na odcinku penetracji.

←

Sondowanie

Otwór A-001

Auto-zapis

Podstawowe informacje

Nazwa sondowania

Otwór A-001

Typ sondowania

DPL ✓

DPM

DPH

DPSH

Młot 10kg · Spadek 0.5m · Stożek 10cm² · Interwał 10cm

Głębokość (m)

1

Woda (m)

1

Pomiary

4

GŁĘBOKOŚĆ	UDERZENIA	
0.00	5	
0.70	20	
0.90	25	
1.00	5	
Dodaj pomiar (1.10 m)		

Zakończenie

Powód zakończenia

Woda gruntowa

Notatki

Projekty

Badanie

Mapa

Sprzęt

Profil

Rysunek 70. Formularz sondowania w aplikacji mobilnej. Źródło: Opracowanie własne.

6.4.3. Pola niestandardowe

Oba formularze badań mogą zostać rozszerzone o pola niestandardowe – dodatkowe pola zdefiniowane przez administratora lub dokumentatora na poziomie firmy lub projektu, dostosowujące zakres gromadzonych danych do specyfiki konkretnej firmy. Sekcję pól niestandardowych w formularzu badania przedstawia Rysunek 71.

The screenshot displays the 'Sondowanie' (Survey) form in a mobile application. At the top, there is a back arrow, the title 'Sondowanie', and a notification bell icon with '13' alerts. Below the title is the identifier 'Otwór A-001' and an 'Auto-zapis' (Auto-save) status. A toolbar contains icons for chat, delete, refresh, document, and bookmark. The main section is titled 'Pola niestandardowe' (Custom Fields) and contains several input fields:

- Warunki atmosferyczne** (Atmospheric conditions): A dropdown menu currently showing 'Słonecznie' (Sunny).
- Data kalibracji sprzętu** (Equipment calibration date): A date picker showing '2026-06-02'.
- Numer zlecenia** (Order number): A text input field containing '12345678'.
- Typ podłoża** (Substrate type): A multiple-choice selection with options: 'Piasek' (Sand), 'Żwir' (Gravel), 'Gлина' (Clay - selected), 'Ił' (Silt), 'Torf' (Peat), and 'Skala' (Rock - selected).

The bottom navigation bar includes icons and labels for 'Projekty' (Projects), 'Badanie' (Survey), 'Mapa' (Map), 'Sprzęt' (Equipment), and 'Profil' (Profile).

Rysunek 71. Sekcja pól niestandardowych w formularzu sondowania w aplikacji mobilnej. Źródło: Opracowanie własne.

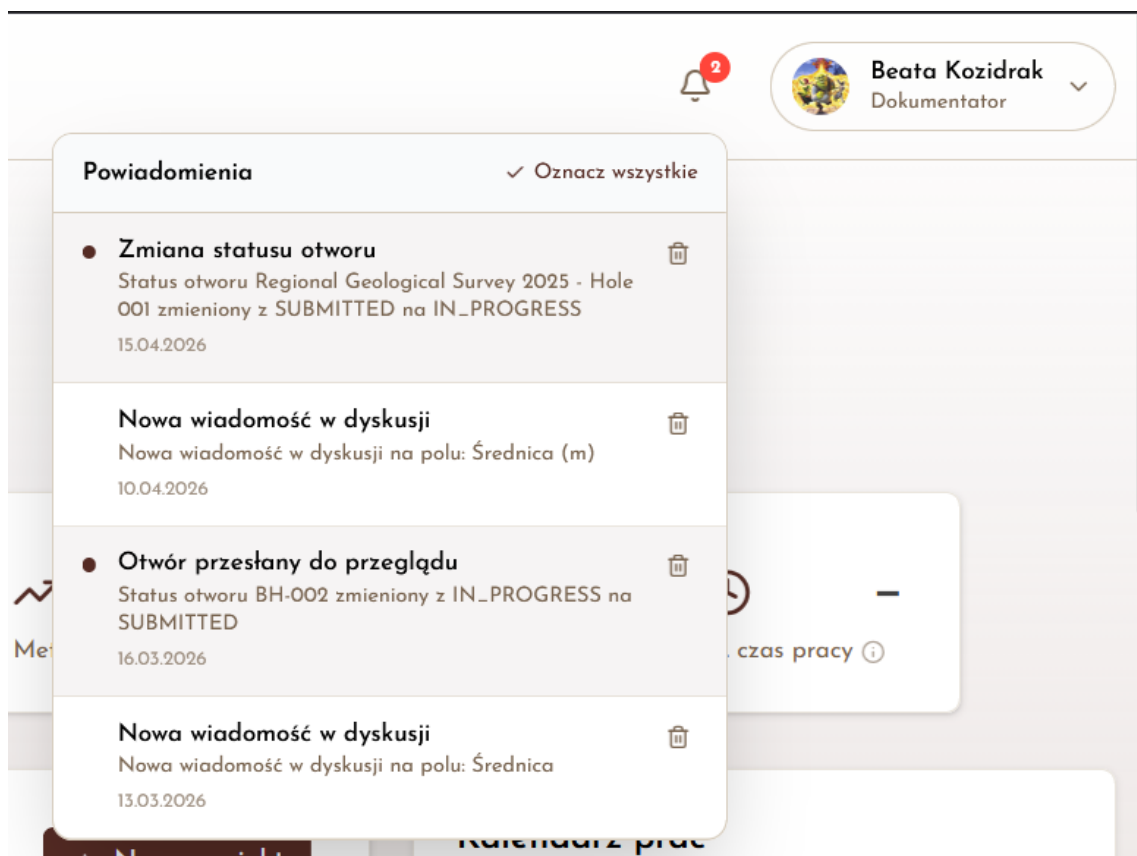
6.4.4. Firmowe wzorce formularzy

Wzorce formularzy, których zastosowanie zostało zaprezentowane w podrozdziale 6.4.1, definiowane są w aplikacji internetowej na poziomie firmy przez administratora lub dokumentatora. Formularz tworzenia wzorca firmowego przedstawia Rysunek 72.

Rysunek 72. Formularz tworzenia wzorca formularza w aplikacji internetowej. Źródło: Opracowanie własne.

6.5. Weryfikacja dokumentacji i dyskusje

Po przekazaniu badania przez geologa terenowego rozpoczyna się etap weryfikacji wykonywanej przez dokumentatora. Dokumentator informowany jest o przypisaniu do weryfikacji za pośrednictwem powiadomienia. Skrzynkę powiadomień, dostępną zarówno w aplikacji internetowej jak i mobilnej, przedstawia Rysunek 73. Wybór powiadomienia przenosi użytkownika bezpośrednio do kontekstu zdarzenia.



Rysunek 73. Skrzynka powiadomień w aplikacji internetowej. Źródło: Opracowanie własne.

Widok badania od strony dokumentatora przedstawia Rysunek 74. Prezentuje on kompletne dane wprowadzone przez geologa.

[← Powrót do projektu: Regional Geological Survey 2025](#)

Regional Geological Survey 2025 - Hole 001

[Do sprawdzenia](#)
[Zatwierdzi](#)
[Odrzuć](#)

[Edytuj](#)

Lokalizacja
52.229700, 21.012200
Otwórz w Google Maps

Zaktualizowano
15.04.2026

Przypisani geolodzy
2

Przypisani geolodzy (2)

[Zarządzaj](#)

Przypisany dokumentator

Odwierci

[Edytuj odwierci](#)
[Usuń odwierci](#)

Głębokość: **12 m**

Średnica: **12 m**

Warstw: **2**

Warstwy geologiczne

0.00 – 0.10 m
Zwir
6 zdjęć

0.20 – 2.50 m
Piasek gruby
7 zdjęć

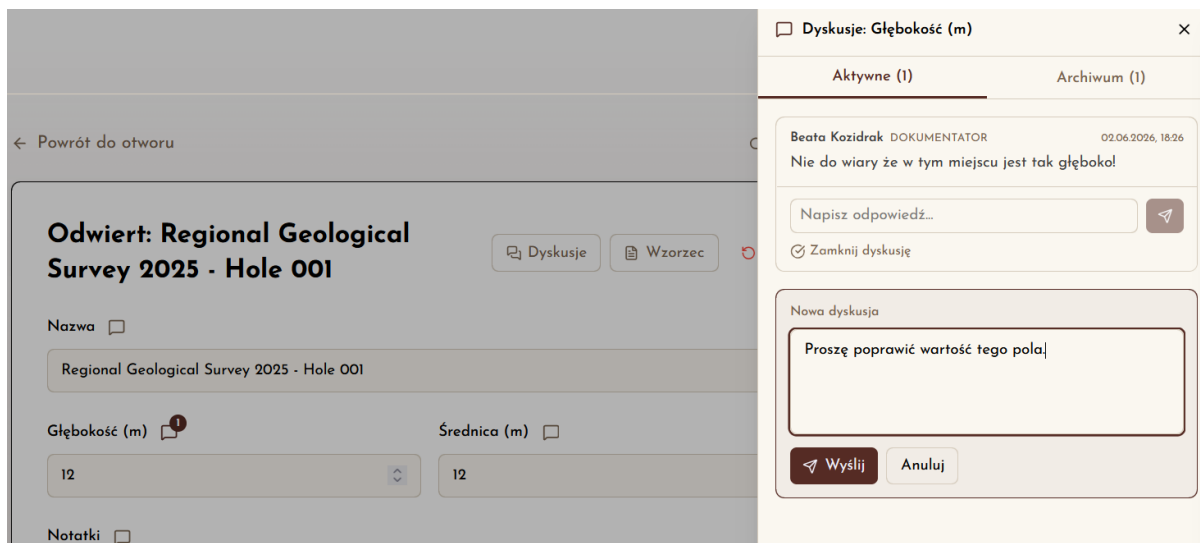
Sondowanie dynamiczne

Brak sondowania

Rysunek 74. Widok badania w trybie weryfikacji przez dokumentatora w aplikacji internetowej. Źródło: Opracowanie własne.

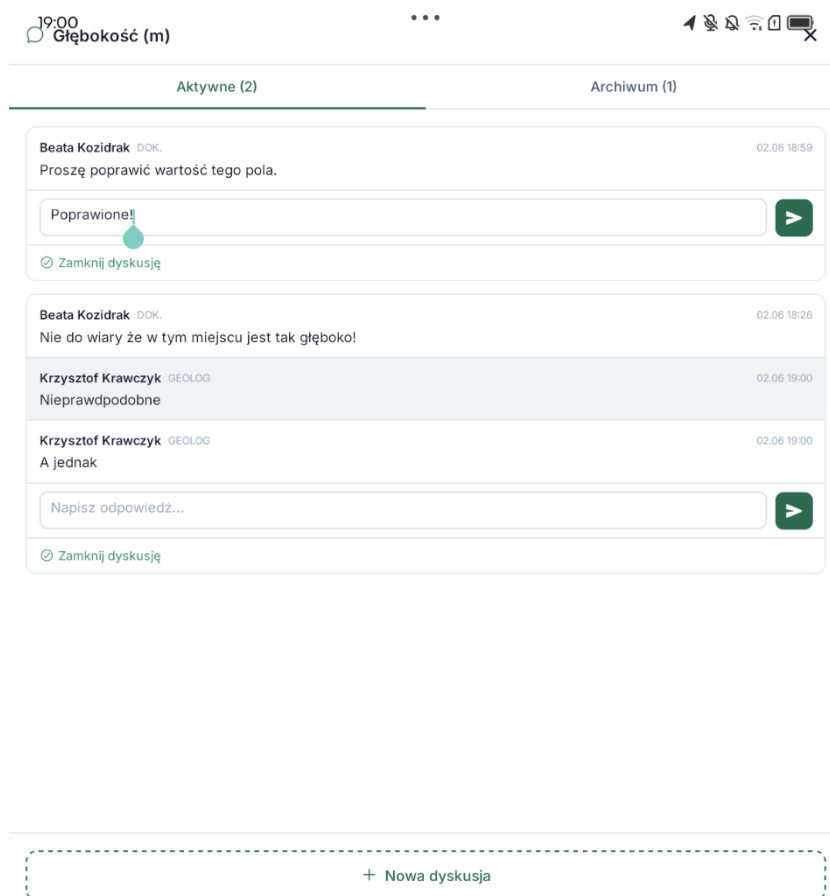
W przypadku wątpliwości co do prawidłowości danych dokumentator dodaje dyskusję co przedstawia Rysunek 75. Powiązanie dyskusji z pojedynczym polem precyzyjnie wskazuje miejsce wymagające poprawy. Istnienie otwartej dyskusji jest zaznaczone w formularzu w postaci licznika z ikonką dyskusji. Dodanie dyskusji skutkuje wysłaniem powiadomienia systemowego oraz wypychanego do przypisanych geologów.

149



Rysunek 75. Dyskusja powiązana z polem formularza w aplikacji internetowej. Źródło: Opracowanie własne.

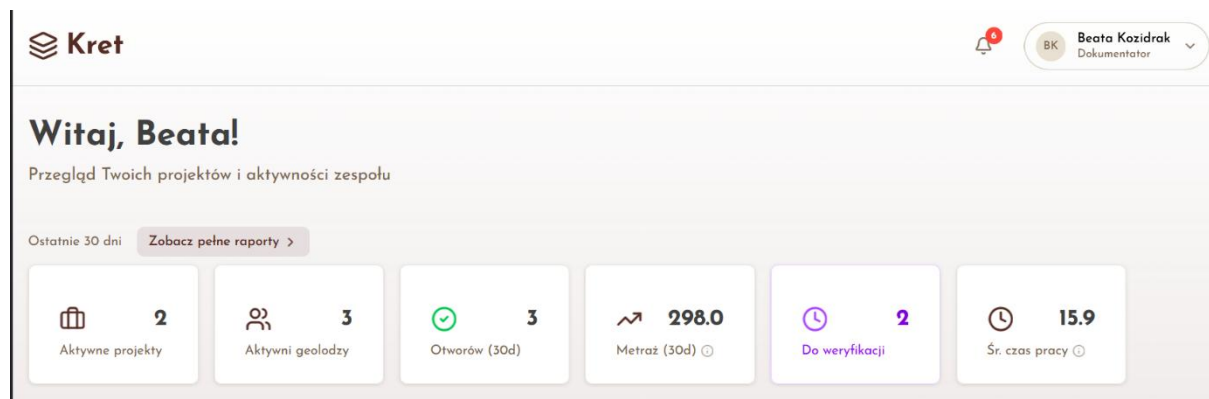
Geolog terenowy odczytuje zgłoszoną uwagę, nanosi poprawkę oraz odpowiada w wątku dyskusji, co przedstawia Rysunek 76. Po usunięciu wszystkich zastrzeżeń dokumentator zatwierdza badanie, co kończy proces weryfikacji i zamyka cykl życia otworu.



Rysunek 76. Odpowiedź geologa terenowego w wątku dyskusji w aplikacji mobilnej. Źródło: Opracowanie własne.

6.6. Statystyki i kalendarz

System KRET udostępnia narzędzia analityczne wspierające monitorowanie postępu zleceń. Po zalogowaniu administrator oraz dokumentator trafiają na pulpit, którego górną część przedstawia Rysunek 77. Prezentuje on zestaw kafelków ze zagregowanymi wskaźnikami opisującymi między innymi liczbę aktywnych projektów, projekty w trakcie realizacji, czy liczbę aktywnych geologów oraz liczbę badań oczekujących na weryfikację.



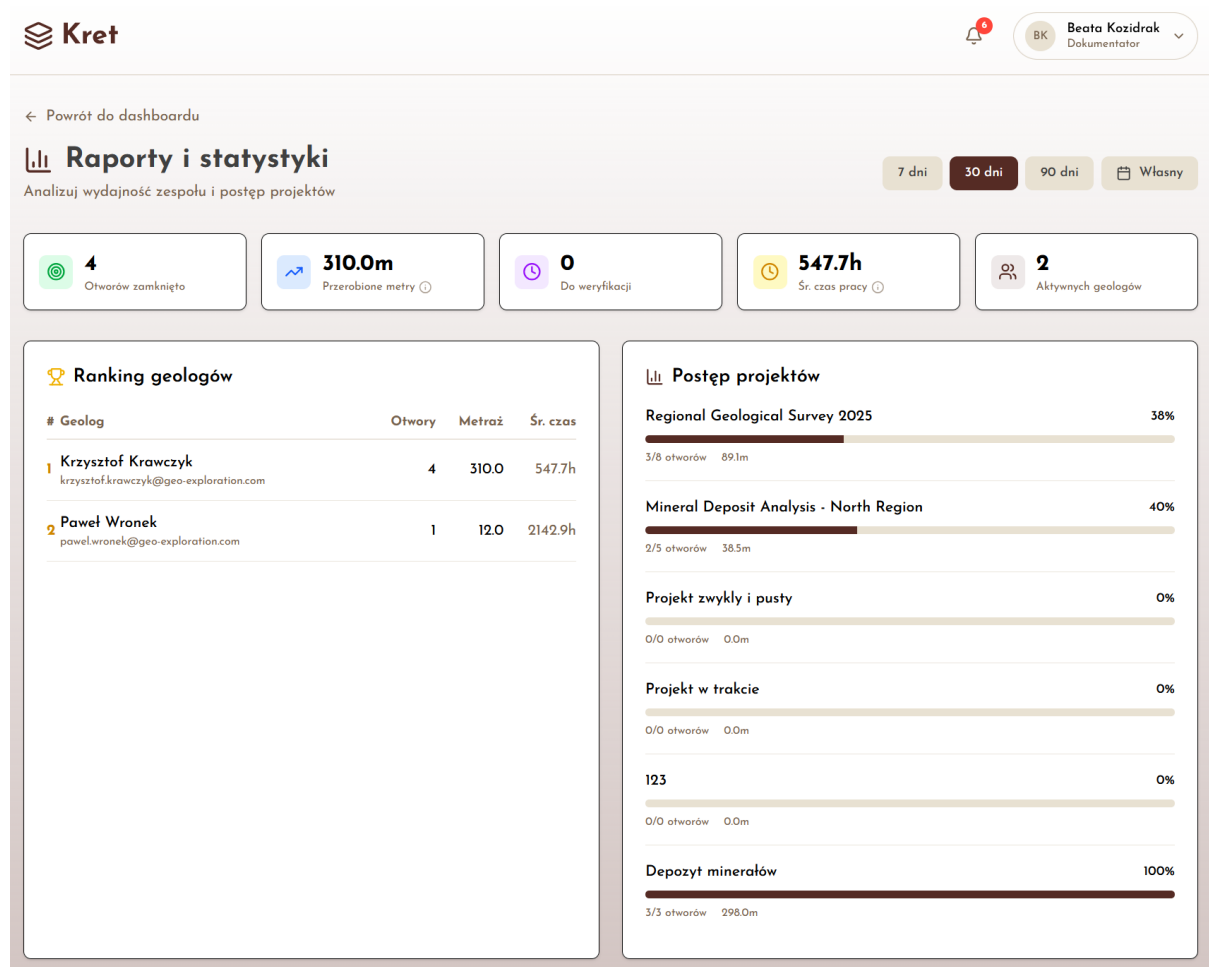
Rysunek 77. Kafelki wskaźników na pulpicie aplikacji internetowej. Źródło: Opracowanie własne.

Integralną częścią panelu głównego jest kalendarz prac, który przedstawia Rysunek 78. Kalendarz gromadzi trzy rodzaje wydarzeń: terminy projektów, terminy związane z otworami oraz rezerwacje sprzętu. Towarzyszy mu zestawienie zadań na bieżący tydzień, które wypisuje otwory oczekujące na weryfikację oraz zbliżające się terminy planowanego zamknięcia projektów.



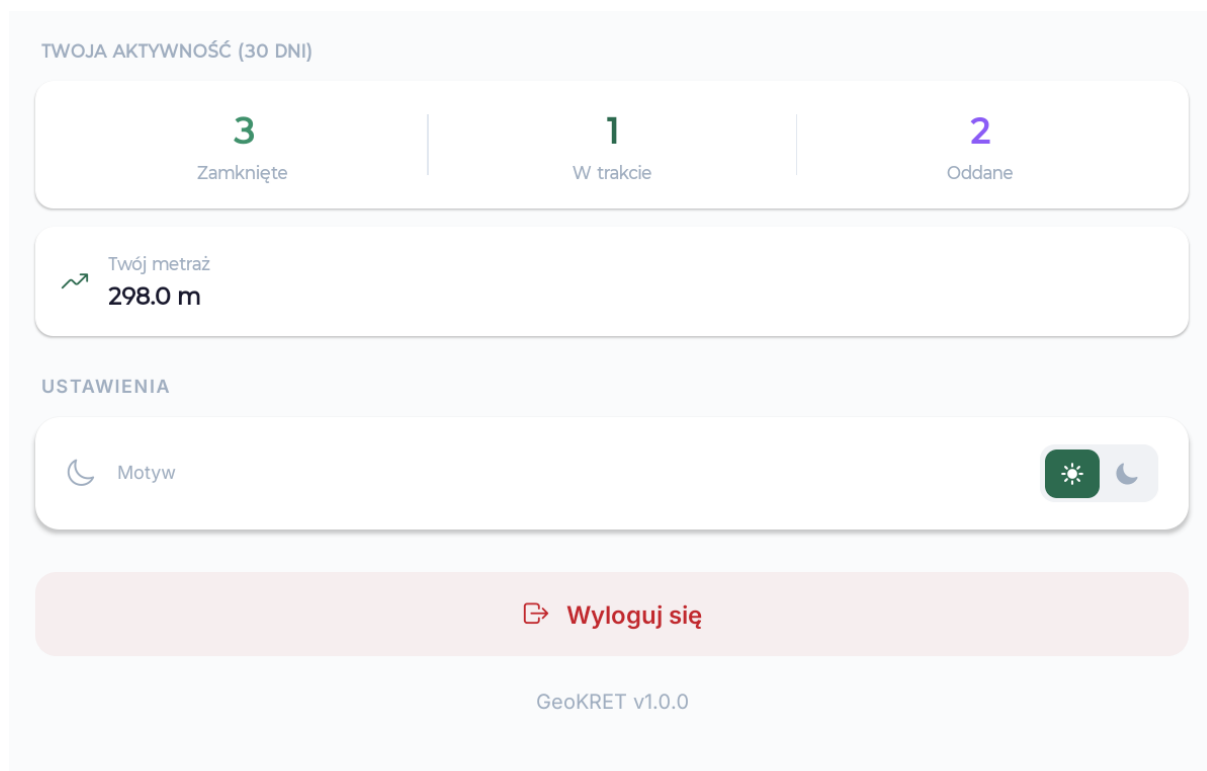
Rysunek 78. Kalendarz prac na pulpicie aplikacji internetowej. Źródło: Opracowanie własne.

Bardziej szczegółowe zestawienie udostępnia ekran raportów, który przedstawia Rysunek 79. Prezentuje on postępy poszczególnych projektów, ranking geologów oraz wskaźniki zbiorcze, takie jak liczba badań, aktywnych geologów, liczba i metraż udokumentowanych otworów, średni czas realizacji.



Rysunek 79. Ekran raportów w aplikacji internetowej. Źródło: Opracowanie własne.

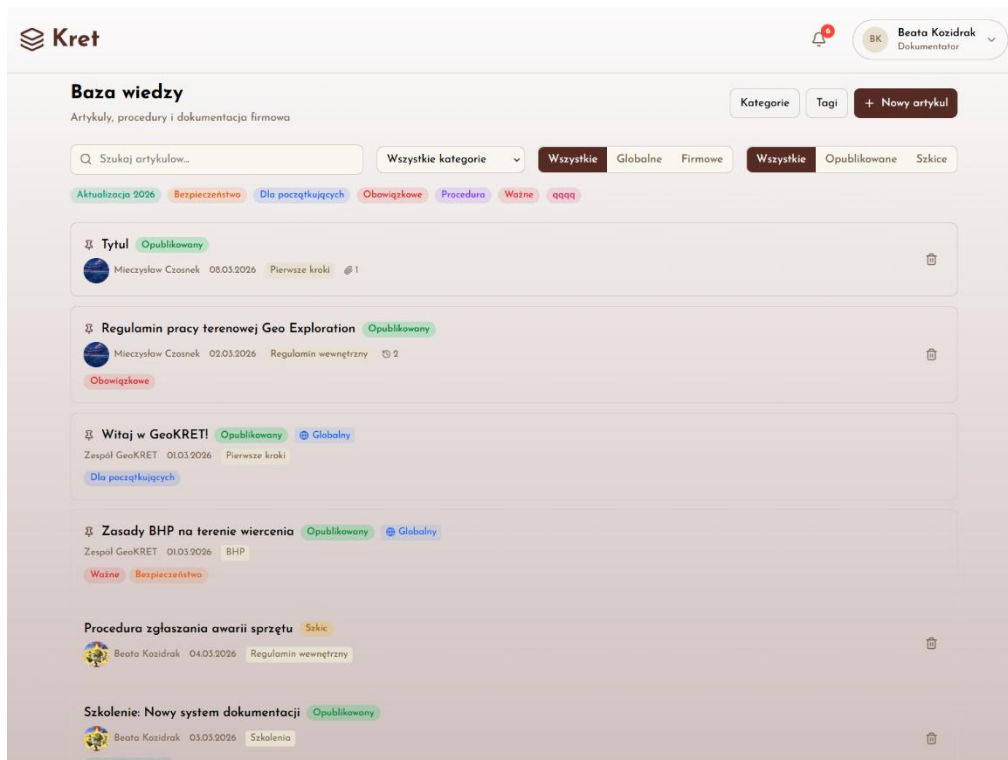
Statystyki własnej pracy są również dostępne dla geologa terenowego w jego profilu w aplikacji mobilnej, co przedstawia Rysunek 80. Zestawienie obejmuje aktywność z ostatnich 30 dni, dzięki temu geolog dysponuje bieżącym podsumowaniem własnych dokonań.



Rysunek 80. Statystyki aktywności w profilu geologa w aplikacji mobilnej. Źródło: Opracowanie własne.

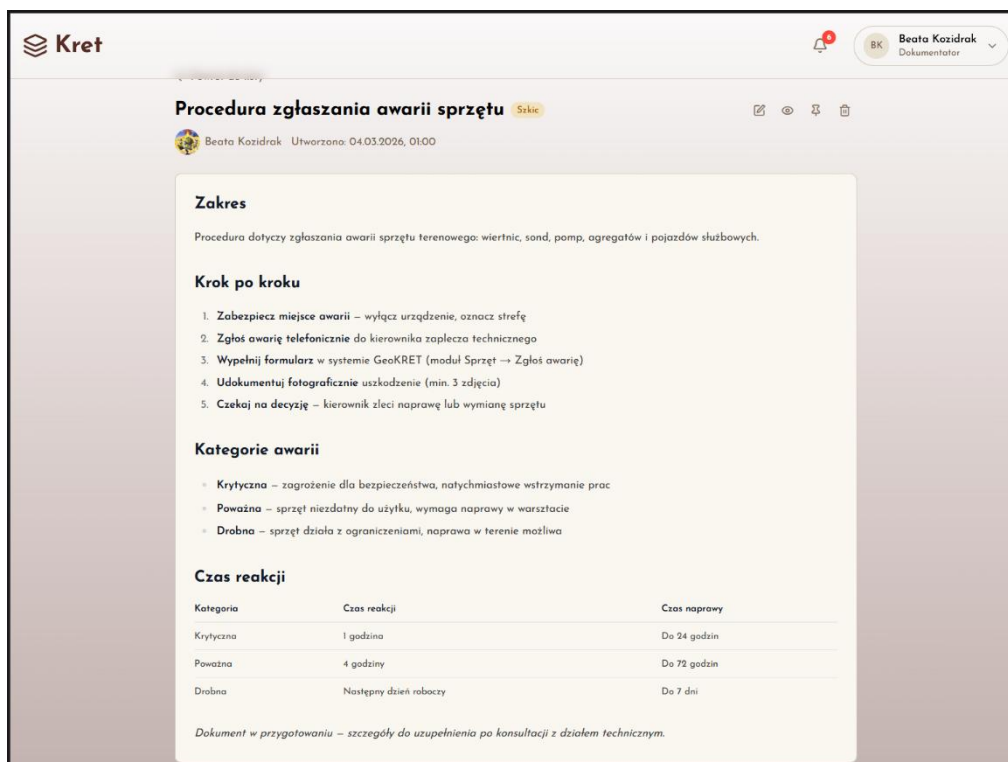
6.7. Baza wiedzy

Baza wiedzy gromadzi materiały pomocnicze dla pracowników firmy: poradniki, opisy procedur, dobre praktyki. Listę artykułów bazy wiedzy przedstawia Rysunek 81. Artykuły opatrzone są tagami, kategoriami oraz oznaczeniem statusu (wersja robocza lub opublikowana). Panel filtrów wraz z wyszukiwarką pozwala na szybkie znalezienie artykułu. Artykuły szczególnie istotne mogą zostać przypięte na początku listy. Z bazy wiedzy korzystają wszystkie role, lecz redagowanie treści jest zarezerwowane tylko dla administratora i dokumentatora operujących z poziomu aplikacji internetowej.



Rysunek 81. Lista artykułów bazy wiedzy z filtrami i tagami w aplikacji internetowej. Źródło: Opracowanie własne.

Widok pojedynczego artykułu przedstawia Rysunek 82.



Rysunek 82. Widok pojedynczego artykułu bazy wiedzy w aplikacji internetowej. Źródło: Opracowanie własne.

Formularz tworzenia i edycji artykułu przedstawia Rysunek 83.

Kret Beata Kozdrak Dokumentator

← Powrot do artykułu

Edytuj artykuł

Tytuł
Procedura zgłaszania awarii sprzętu

Treść (Markdown) Podgląd

Zakres

Procedura dotyczy zgłaszania awarii sprzętu terenowego: wiertnic, sond, pomp, agregatów i pojazdów służbowych.

Krok po kroku

1. ****Zabezpiecz miejsce awarii**** – wyłącz urządzenie, oznacz strefę
2. ****Zgłoś awarię telefonicznie**** do kierownika zaplecza technicznego
3. ****Wypełnij formularz**** w systemie GeoKRET (moduł Sprzęt → Zgłoś awarię)
4. ****Udokumentuj fotograficznie**** uszkodzenie (min. 3 zdjęcia)
5. ****Czekaj na decyzję**** – kierownik zleci naprawę lub wymianę sprzętu

Kategorie awarii

Kategoria
Regulamin wewnętrzny

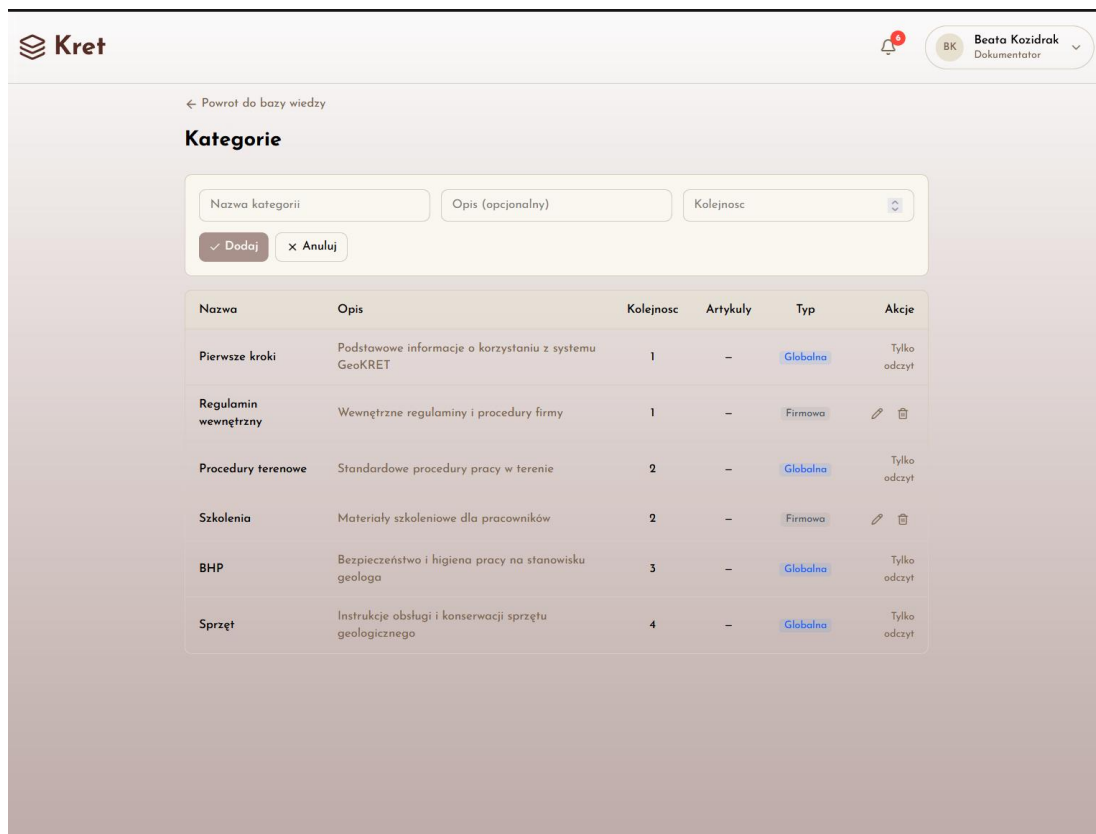
Tagi
Aktualizacja 2026 Bezpieczeństwo Dla początkujących Obowiązkowe Procedura Ważne qqqq

Opis zmiany
Opisz co zostało zmienione...

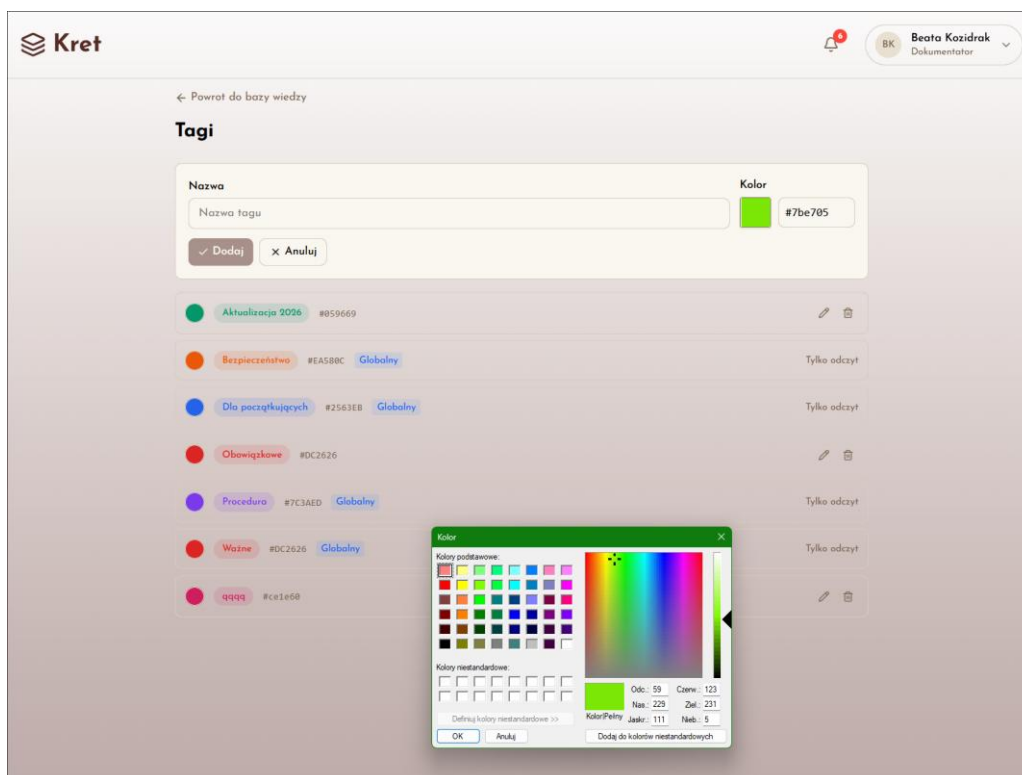
Zapisz zmiany **Anuluj**

Rysunek 83. Formularz tworzenia i edycji artykułu bazy wiedzy w aplikacji internetowej. Źródło: Opracowanie własne.

Strukturę bazy wiedzy porządkują kategorie oraz tagi, którymi zarządza się na osobnym ekranie przedstawionym na Rysunek 84 i Rysunek 85.

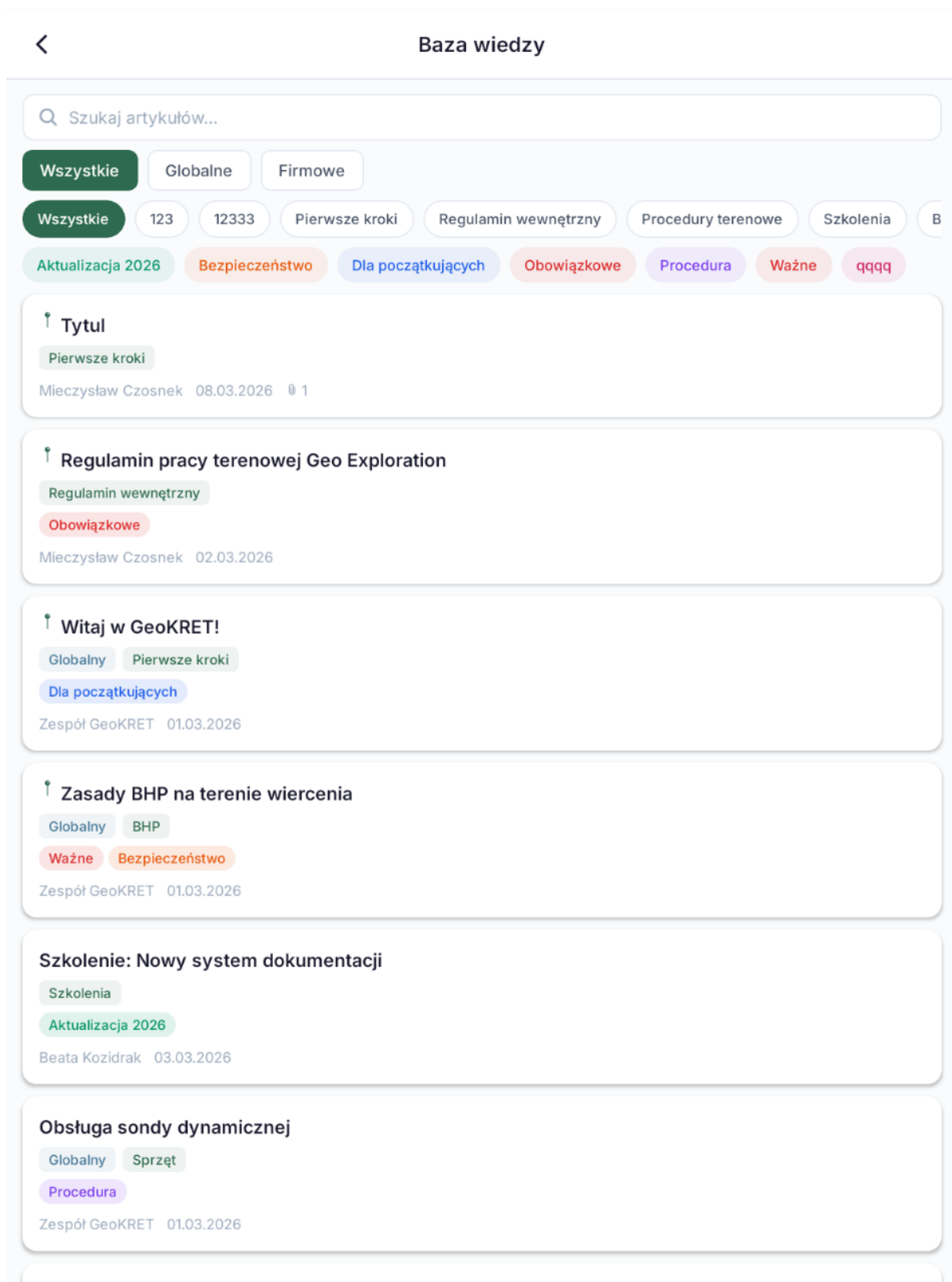


Rysunek 84. Zarządzanie kategoriami bazy wiedzy w aplikacji internetowej. Źródło: Opracowanie własne.



Rysunek 85. Zarządzanie tagami bazy wiedzy w aplikacji internetowej. Źródło: Opracowanie własne.

Dostęp do bazy wiedzy jest również dostępny z aplikacji mobilnej, a mobilną listę artykułów przedstawia Rysunek 86. Mobilna wersja również pozwala na zaawansowane przeszukiwanie artykułów poprzez wykorzystanie tagów i kategorii.

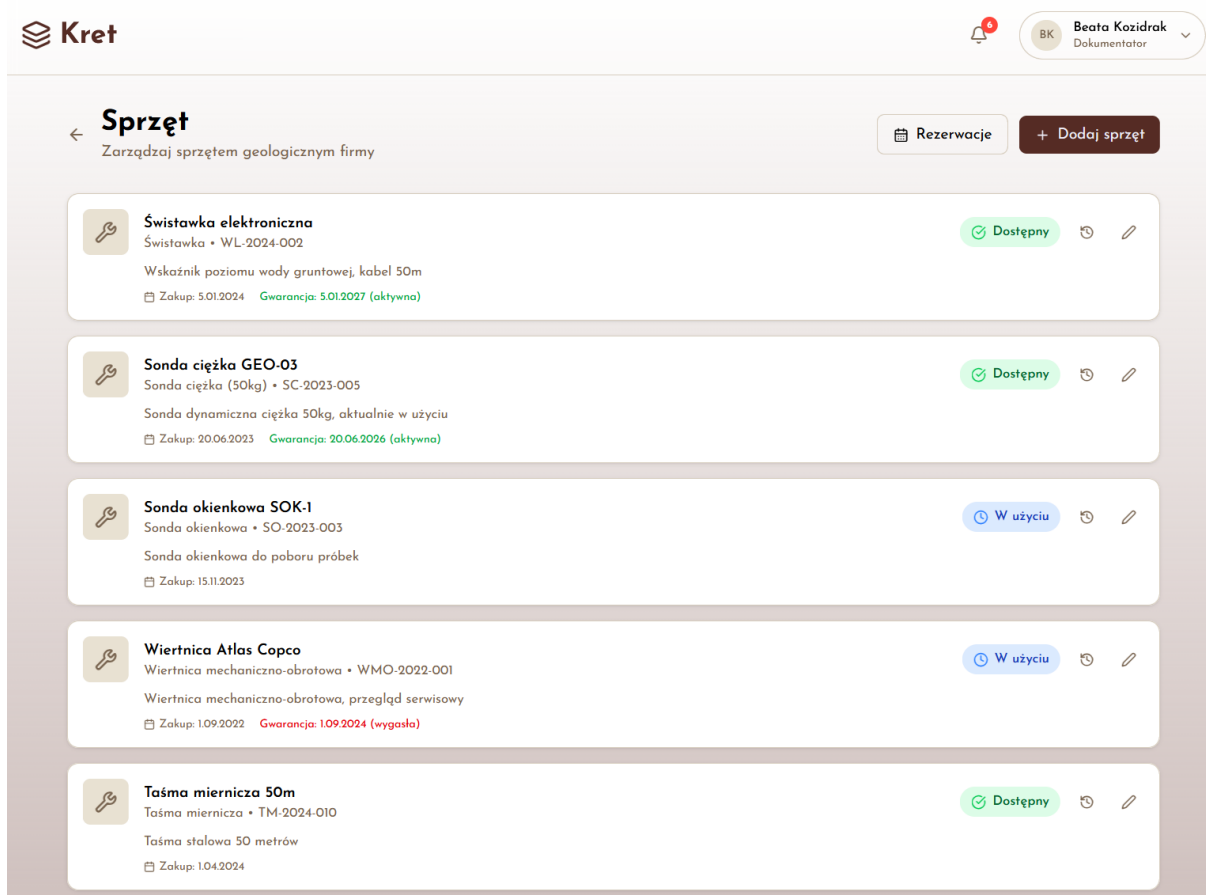


Rysunek 86. Lista artykułów bazy wiedzy w aplikacji mobilnej. Źródło: Opracowanie własne.

6.8. Zarządzanie sprzętem

Moduł zarządzania sprzętem prowadzi ewidencję wyposażenia firmy oraz obsługuje jego rezerwacje na potrzeby projektów.

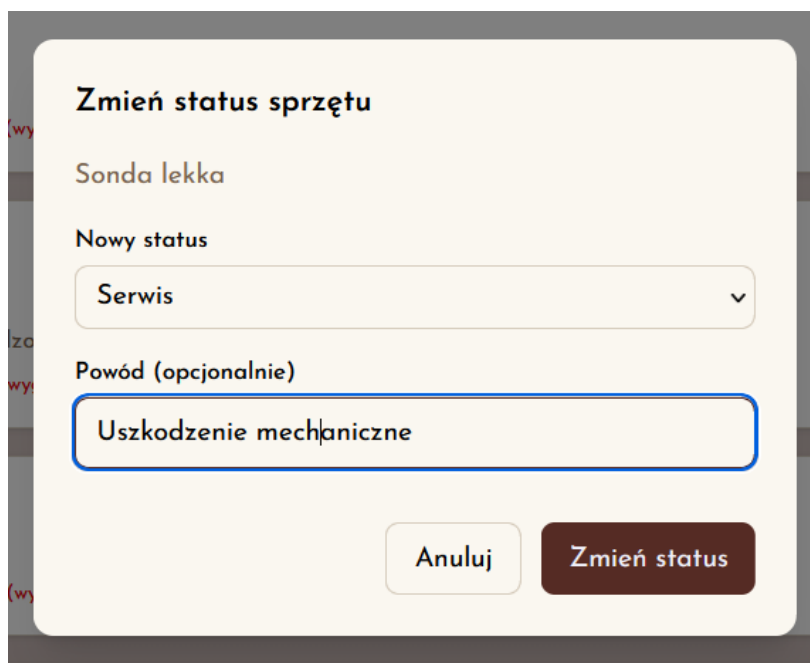
Listę sprzętu firmy wraz z bieżącymi statusami przedstawia Rysunek 87. Każdy egzemplarz opisany jest typem, numerem seryjnym oraz datami zakupu i zakończenia gwarancji. Rozróżniane jest pięć statusów: dostępny, w użyciu, w konserwacji, uszkodzony oraz wycofany.



Sprzęt Zarządzaj sprzętem geologicznym firmy		
	Świstawka elektroniczna Świstawka • WL-2024-002 Wskaźnik poziomu wody gruntowej, kabel 50m Zakup: 5.01.2024 Gwarancja: 5.01.2027 (aktywna)	
	Sonda ciężka GEO-03 Sonda ciężka (50kg) • SC-2023-005 Sonda dynamiczna ciężka 50kg, aktualnie w użyciu Zakup: 20.06.2023 Gwarancja: 20.06.2026 (aktywna)	
	Sonda okienkowa SOK-1 Sonda okienkowa • SO-2023-003 Sonda okienkowa do poboru próbek Zakup: 15.11.2023	
	Wiertnica Atlas Copco Wiertnica mechaniczno-obrotowa • WMO-2022-001 Wiertnica mechaniczno-obrotowa, przegląd serwisowy Zakup: 1.09.2022 Gwarancja: 1.09.2024 (wygasła)	
	Taśma miernicza 50m Taśma miernicza • TM-2024-010 Taśma stalowa 50 metrów Zakup: 1.04.2024	

Rysunek 87. Lista sprzętu firmy z bieżącymi statusami w aplikacji internetowej. Źródło: Opracowanie własne.

Ręczną zmianę statusu sprzętu prezentuje Rysunek 88.



Zmień status sprzętu

Sonda lekka

Nowy status

Serwis

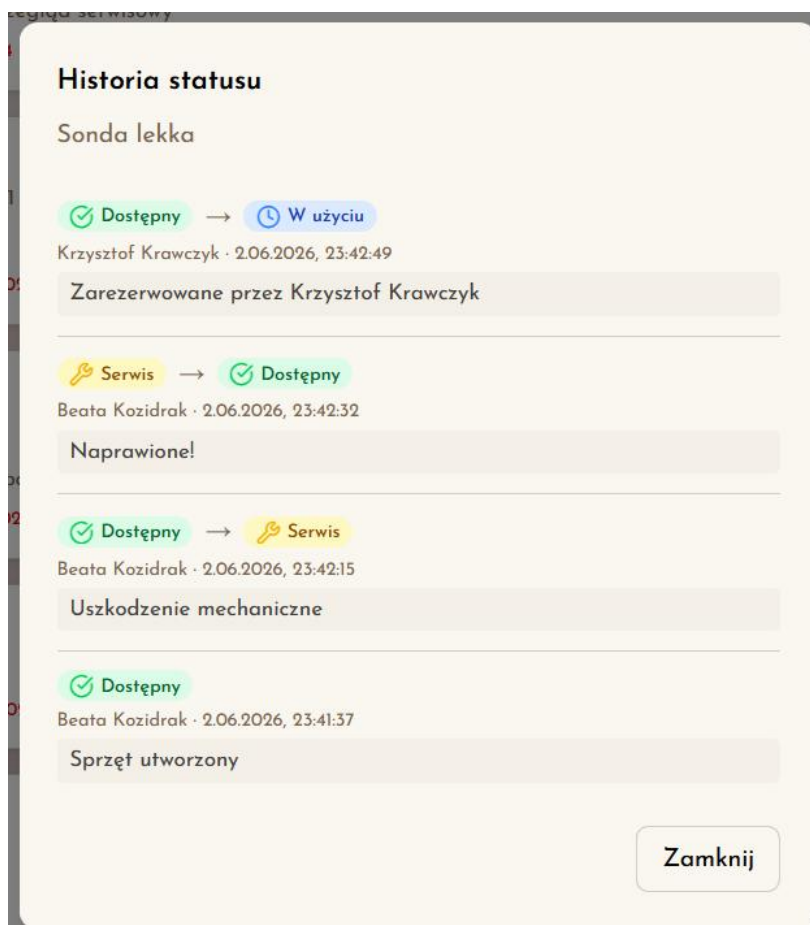
Powód (opcjonalnie)

Uszkodzenie mechaniczne

Anuluj Zmień status

Rysunek 88. Zmiana statusu sprzętu w aplikacji internetowej. Źródło: Opracowanie własne.

Historię zmian statusu przedstawia Rysunek 89.



Historia statusu

Sonda lekka

✓ Dostępny → ⌚ W użyciu

Krzysztof Krawczyk · 2.06.2026, 23:42:49

Zarezerwowane przez Krzysztof Krawczyk

🔧 Serwis → ✓ Dostępny

Beata Kozidrak · 2.06.2026, 23:42:32

Naprawione!

✓ Dostępny → 🔧 Serwis

Beata Kozidrak · 2.06.2026, 23:42:15

Uszkodzenie mechaniczne

✓ Dostępny

Beata Kozidrak · 2.06.2026, 23:41:37

Sprzęt utworzony

Zamknij

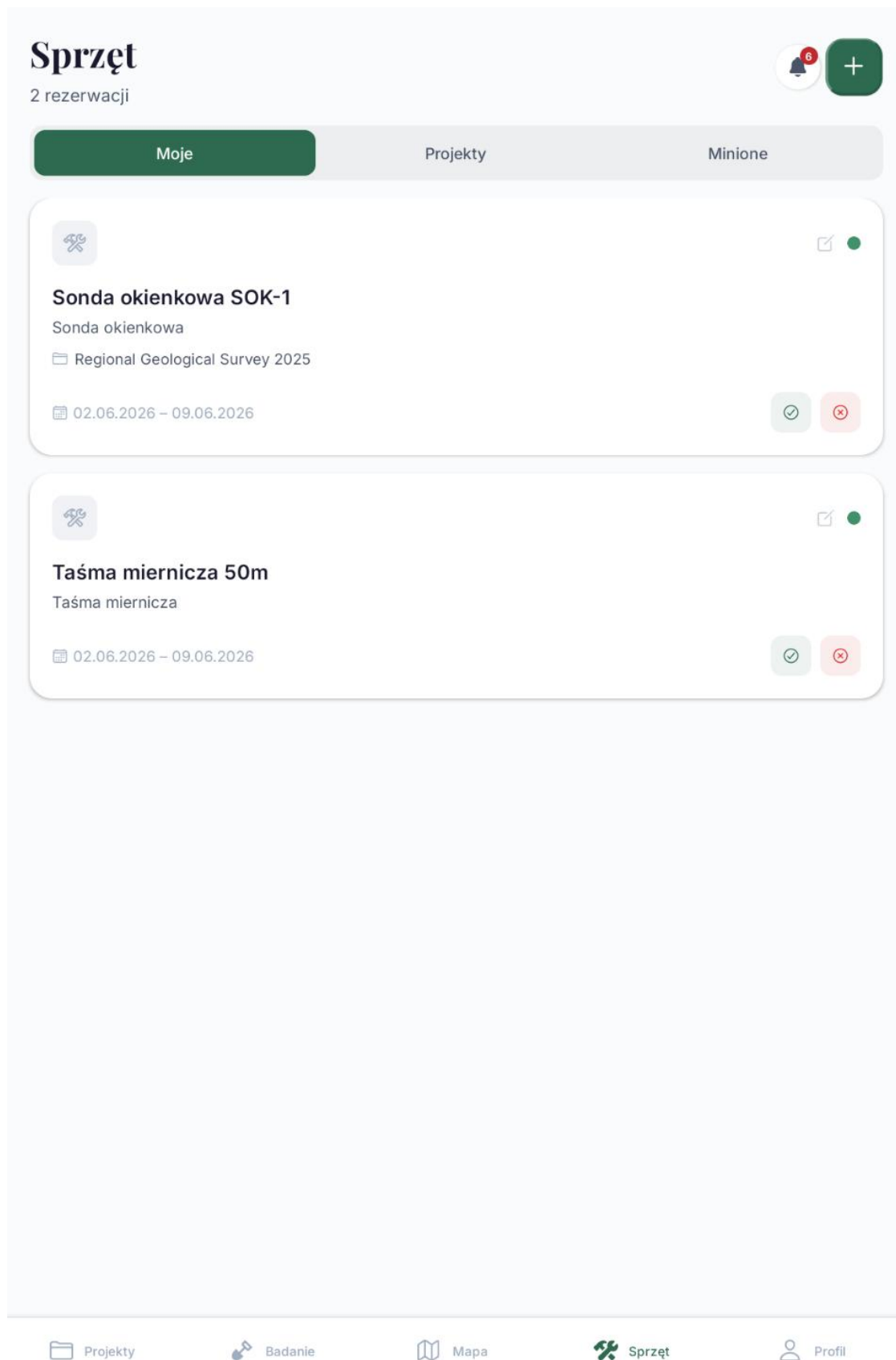
Rysunek 89. Historia statusu sprzętu w aplikacji internetowej. Źródło: Opracowanie własne.

Rezerwację sprzętu przedstawia Rysunek 90. Użytkownik wskazuje egzemplarz oraz okres rezerwacji, a opcjonalnie wiąże rezerwację z projektem.

The screenshot displays the 'Kret' application interface for equipment reservations. The main header shows the 'Kret' logo and a user profile for 'Beata Kozdrak, Dokumentator'. The page title is 'Rezerwacje sprzętu' with a subtitle 'Przeglądaj i zarządzaj rezerwacjami'. A '+ Nowa rezerwacja' button is in the top right. Below the title are two tabs: 'Moje rezerwacje' and 'Wszystkie rezerwacje'. A list of reservations is shown, including 'Sonda lekka GEO-01', 'Sonda ciężka GEO-03', and another 'Sonda lekka GEO-01'. A modal window titled 'Nowa rezerwacja' is open in the center. It contains the following fields: 'Sprzęt *' with a dropdown menu showing 'Świstawka elektroniczna (Świstawka)'; 'Projekt (opcjonalnie)' with a dropdown menu showing 'Mineral Deposit Analysis - North Region'; 'Od *' and 'Do *' date pickers showing '02.06.2026' and '09.06.2026' respectively; and a 'Notatki' section with a text input field 'Opcjonalne informacje...'. At the bottom of the modal are 'Anuluj' and 'Zarezerwuj' buttons. The background reservations list shows dates, status (e.g., 'Zakończona', 'Aktywna'), and action icons.

Rysunek 90. Rezerwacja sprzętu w aplikacji internetowej. Źródło: Opracowanie własne.

Rezerwacje sprzętu są również dostępne dla geologa z poziomu aplikacji mobilnej, co przedstawia Rysunek 91 oraz Rysunek 92.



Rysunek 91. Lista złożonych rezerwacji sprzętu w aplikacji mobilnej. Źródło: Opracowanie własne.

X

Nowa rezerwacja

Sprzęt

Sonda ciężka GEO-03 (Sonda ciężka (50ka))

Projekt (opcjonalnie)

Mineral Deposit Analysis - North Region

Termin

Od

02.06.2026

Do

09.06.2026

Notatki (opcjonalnie)

Dodatkowe informacje...

✓ Zarezerwuj sprzęt

Rysunek 92. Rezerwacje sprzętu w aplikacji mobilnej. Źródło: Opracowanie własne.

6.9. Eksport danych

System KRET umożliwia wyeksportowanie zgromadzonych danych do zewnętrznego oprogramowania branżowego oraz wygenerowanie końcowej dokumentacji w formacie PDF.

Okno eksportu danych projektu przedstawia Rysunek 93. Eksport przebiega dwuetapowo: użytkownik najpierw określa zakres danych poprzez wybranie grup informacji do uwzględnienia, a następnie wybiera format wynikowy. W bieżącej wersji systemu zaimplementowano eksport do formatów CSV i XLSX. W systemie przewidziano kolejne znaczące, ale mniej używane na polskim rynku formaty JSON, GeoJSON, XML i AGS 4., ich wdrożenie zaplanowano jako rozszerzenie systemu.





BK
Beata Kozdrak
Dokumentator

← Powrót do projektu


Eksportuj projekt
Coastal Erosion Study

1. Zakres danych

☒ Otwory

☒ Odwierty

☒ Warstwy geologiczne

☒ Sondowania

☒ Pomiar sondowań

☒ Pola własne

☒ Zdjęcia warstw

2. Format eksportu

 CSV
CSV (ZIP z plikami)

 XLSX
Excel (.xlsx)

 JSON
JSON (wkrótce)

 GEOJSON
GeoJSON (dla GIS) (wkrótce)

 AGS4
AGS 4 (standard UK) (wkrótce)

 XML
XML (Eurocode) (wkrótce)

3. Szablony eksportu

 Zapisz jako szablon

Rysunek 93. Okno eksportu danych projektu w aplikacji internetowej. Źródło: Opracowanie własne.

Dla eksportu danych do plików tekstowych, wyeksportowane końcowe klucze danych można zmienić na dowolne wartości, zmienić właściwości końcowego pliku takie jak separator danych w pliku, a następnie tę konfigurację można zapisać jako szablon na przyszłość. Konfiguracje eksportu danych prezentuje Rysunek 94.

163

3. Szablony eksportu

Zapisz jako szablon

Domyślne ustawienia

Domyslny

XLSX • Prywatny

4. Mapowanie kolumn

Wybierz pola i zmień nazwy kolumn

☒

Otwór

Status otworu

→

Status otworu

☒

Otwór

Szerokość geograficzna

→

Szerokość geograficzn

☒

Otwór

Długość geograficzna

→

Długość geograficzna

☒

Otwór

Przypisani geolodzy

→

Przypisani geolodzy

☒

Odwierť

Nazwa odwiertu

→

Nazwa odwiertu

☒

Odwierť

Głębokość odwiertu [m]

→

Głębokość odwiertu [m]

5. Opcje zaawansowane

Kodowanie

UTF-8

Separator (CSV)

Średnik (;)

Format daty

DD.MM.YYYY

Separator dziesiętny

Przecinek (.) - polski

Format litologii

Symbol + nazwa (Pg - Piasek gruby)

Format współrzędnych

Dziesiętne (52.4067)

Anuluj

Eksportuj projekt

Rysunek 94. Konfiguracja eksportu danych XLSX w aplikacji internetowej. Źródło: Opracowanie własne.

Końcowym dokumentem wynikowym pojedynczego otworu jest karta otworu. Konfigurację karty przedstawia Rysunek 95. Użytkownik określa tytuł główny, podtytuł, numer załącznika, preferowaną skalę oraz maksymalną liczbę stron.

164

Gmina: Szczecin Powiat: Szczecin Województwo: województwo zachodniopomorskie	KARTA OTWORU BADAWCZEGO		Zał. Nr: 3.1	
	Profil numer Coastal Erosion Study - Hole 001		Współrzędne: X: — Y: —	
	Inwestor: Geo Exploration Ltd Zleceniodawca: Coastal Erosion Study Operator: —		Głębokość: 12.50 m	Data: 6.01.2026
			Skala: 1:100	

1	2	3	4	5	6	7	8	9	10
GLEBOKOŚĆ/CIŚCIE [MPa]	SKALA	PROFIL	PRZELOT [m]	OPIS PN-EN ISO 14688-2018	SYMBOL GRUNTU PN-EN ISO	SYMBOL GRUNTU WG PN-B	STAN GRUNTU	WILGOTNOŚĆ	WARSTWA GEOTECHNICZNA
	0.0 0.3	Ps	0.30	Humus, ciemnobrązowy Piasek średni, żółty	H	—	—	0.7	1
		Ps	3.50		Ps	—	—	0.4	2
▼ 3.5	3.3	Po	12.50	Piasek gruby ze żwirem, szary	Po	—	—	0.9	3

Rysunek wykonano programem **GeoKret** zgodnie z normą PN-B-02480:1986

Rysunek 96. Wygenerowana karta otworu w systemie KRET. Źródło: Opracowanie własne.

6.10. Import otworów

Oprócz ręcznego wprowadzania danych system KRET umożliwia ich import z plików w formacie CSV. Import obejmuje trzy rodzaje danych: import otworów, wierceń oraz sondowań.

Okno importu wraz z ustawieniami przedstawia Rysunek 97. Przed wskazaniem pliku użytkownik wskazuje parametry odczytu, czyli separator pól oraz kodowanie znaków. Ekran udostępnia opis oczekiwanych pól, co ułatwia przygotowanie pliku.

← Powrót do projektu

Otwory Wiercenia Sondowania

Import sondowań dynamicznych

Średnik (;) Windows-1250 (Excel PL)

Plik: sounding.csv

Nazwa otworu	Typ Sondy	Głębokość (m)	Ilość uderzeń (N)	Głębokość wody
tabc	DPL	0.1	3	
tabc	DPL	0.2	4	
tabc	DPL	0.3	12	
tabc	DPH	0.1	15	

Importuj Wróć do wyboru pliku

Wymagana struktura pliku CSV:

```
HoleName;Kind;Depth;BlowCount;GroundwaterDepth;TerminationReason;Remarks;Notes
```

Opis kolumn:

- HoleName - nazwa otworu, musi zgadzać się z otworem w projekcie (wymagane)
- Kind - typ sondy: DPL, DPM, DPH, DPSH, SPT (wymagane)
- Depth - głębokość wykonania pomiaru w metrach np. 0.1, 0.2 (wymagane)
- BlowCount - liczba uderzeń N dla danej głębokości (wymagane)
- GroundwaterDepth - głębokość wody gruntowej (opcjonalne)
- TerminationReason - powód zakończenia z listy poniżej (opcjonalne)
- Remarks - uwagi do konkretnego pomiaru/wiersza (opcjonalne)
- Notes - ogólne uwagi do całego sondowania (opcjonalne)

Powody zakończenia sondowania (TerminationReason):

```
REFUSAL, TARGET_DEPTH_REACHED, BEARING_LAYER_REACHED, GROUNDWATER, EQUIPMENT_FAILURE, OTHER
```

Przykładowe dane w CSV:

```
HoleName;Kind;Depth;BlowCount;GroundwaterDepth;TerminationReason;Remarks;Notes
Otwór1;DPL;0.1;3;1.5;TARGET_DEPTH_REACHED;Luźny piasek;Słonecznie
Otwór1;DPL;0.2;4;;;;
Otwór1;DPL;0.3;12;;;;
```

Każdy wiersz reprezentuje JEDEN punkt pomiarowy. Wiele wierszy o tej samej nazwie otworu (HoleName) zostanie zgrupowanych w jedno sondowanie z listą pomiarów. Parametry ogólne (Kind, Notes, itp.) są pobierane z pierwszego wiersza dla danego otworu.

Rysunek 97. Okno importu danych z ustawieniami odczytu pliku w aplikacji internetowej. Źródło: Opracowanie własne.

Po wskazaniu pliku system wyświetla jego zawartość w formie tabeli oraz oznacza wykryte błędy ze wskazaniem numeru wiersza i przyczyny co przedstawia Rysunek 98.

Import sondowań dynamicznych

Średnik (;)

UTF-8

Plik: sounding.csv

Nazwa otworu	Typ Sondy	Głębokość (m)	Ilość uderzeń (N)	Głębokość wod.
Brak	DPL	0.1	3	
1abc	DPL	0.2	4	
1abc	DPL	0.3	12	
1abc	DPH	0.1	15	

Importuj

Wróć do wyboru pliku

Rysunek 98. Podgląd zawartości importowanego pliku z oznaczeniem błędów. Źródło: Opracowanie własne.

7. Testy i ewaluacja

Weryfikacja poprawności działania systemu KRET stanowiła integralną część procesu wytwórczego. Prowadzenie testów przez zespół było wykonywane równoległe z rozwojem kolejnych modułów. Każda funkcjonalność przed scaleniem z gałęzią główną podlegała ocenie kodu przez drugiego członka zespołu oraz weryfikacji działania.

Przyjęto dwa komplementarne podejścia do weryfikacji poprawności. Zautomatyzowane testy zabezpieczają logikę aplikacji serwerowej przed regresją i uruchamiane są przy każdej zmianie w kodzie wypchniętej do repozytorium kodu. Testy manualne sprawdzają kompletne ścieżki użytkownika według scenariuszy opisanych w przypadkach użycia zdefiniowanych w podrozdziale 3.7. Uzupełnieniem weryfikacji technicznej były nieformalne konsultacje z praktykami branży geologicznej, pozwalające na bieżącą ocenę przydatności wdrażanych rozwiązań, omówione w podrozdziale 7.3.

Wykryte błędy korygowano na bieżąco. Wadliwego kodu nie wprowadzano do gałęzi głównej, a napotkane usterki naprawiano doraźnie. Ryzyko trafienia błędu do gałęzi głównej ograniczały ocena kodu w ramach żądania scalenia oraz sygnalizowanie niepowodzenia testów w ramach ciągłej integracji, uruchamianej dla każdego żądania scalenia.

7.1. Testy automatyczne

Zautomatyzowane testy aplikacji serwerowej w języku Java tworzone przy pomocy biblioteki JUnit w wersji 5, dostarczanej wraz z zależnościami biblioteki Mockito oraz AssertJ przez moduł `spring-boot-starter-test`. JUnit udostępnia środowisko uruchomieniowe testów oraz adnotacje oznaczające metody testowe. Mockito pozwala na tworzenie atrap zależności, natomiast AssertJ zapewnia czytelny zapis testowych twierdzeń porównujących wynik oczekiwany z rzeczywistym. W projekcie wyróżniono dwa rodzaje testów zautomatyzowanych: testy jednostkowe weryfikujące izolowane fragmenty logiki oraz testy integracyjne sprawdzające współdziałanie warstw aplikacji.

Testy jednostkowe obejmują logikę biznesową skupioną w warstwie serwisów oraz logikę encji. Są to elementy o charakterze obliczeniowym, dające się weryfikować w izolacji. Zależności bazodanowe testowanego serwisu zastępowane są atrapami repozytoriów oznaczonymi jako `@Mock` i wstrzykiwane do badanego obiektu, dzięki czemu test sprawdza wyłącznie logikę serwisu niezależnie od stanu bazy danych. Szczególnie istotnym przedmiotem testów jednostkowych są reguły poprawności struktury warstw geologicznych. Reguła dopuszcza warstwy stykające się granicą, lecz odrzuca warstwy nakładające się. Listing 46 przedstawia test weryfikujący tę regułę.

Listing 46. Test jednostkowy reguły walidacji nakładania się warstw geologicznych. Źródło: Opracowanie własne.

```
@ExtendWith(MockitoExtension.class)
class BoreholeServiceTest {

    @Mock
    private BoreholeRepository boreholeRepository;
    @Mock
    private HoleRepository holeRepository;
    @Mock
    private SecurityContextService securityContextService;
    // pozostałe atrapy zależności oraz przygotowanie danych
    // testowych w metodzie @BeforeEach pominięto

    @InjectMocks
    private BoreholeService boreholeService;
```

```

@Test
void createBorehole_WithOverlappingLayers_ThrowsValidationException() {
    CreateBoreholeLayerRequest layer1 = new CreateBoreholeLayerRequest();
    layer1.setDepthFrom(0.0);
    layer1.setDepthTo(6.0);

    CreateBoreholeLayerRequest layer2 = new CreateBoreholeLayerRequest();
    layer2.setDepthFrom(5.0); // Nakłada się na layer1
    layer2.setDepthTo(10.0);

    CreateBoreholeRequest request = new CreateBoreholeRequest();
    request.setName("Borehole with Overlapping Layers");
    request.setDepth(15.0);
    request.setDiameter(0.6);
    request.setLayers(List.of(layer1, layer2));

    when(securityContextService.getCurrentUser()).thenReturn(adminUser);

    when(holeRepository.findById(testHole.getId())).thenReturn(Optional.of(testHole));

    assertThatThrownBy(() -> boreholeService.createOrReplaceBorehole(
        testProject.getId(), testHole.getId(), request))
        .isInstanceOf(ValidationException.class)
        .hasMessageContaining("nakładają się");

    verify(boreholeRepository, never()).save(any(Borehole.class));
}
}

```

Atrapy zastępują współpracujące z serwisem repozytoria i serwisy pomocnicze, a ich zachowanie na potrzeby scenariusza określają wywołania `when(...).thenReturn(...)`. Test przygotowuje dwie warstwy o zakresach 0-6m oraz 5-10m, które nakładają się na odcinku od 5 do 6 metra. Po wywołaniu metody serwisu twierdzenie testowe `assertThatThrownBy` sprawdza zgłoszenie wyjątku `ValidationException` z komunikatem zawierającym frazę „nakładają się”, natomiast wywołanie `verify(..., never())` potwierdza, że w wyniku nieudanego sprawdzenia reguły żadne wiercenie nie zostało zapisane w repozytorium. Komplementarny przypadek, pominięty dla zwięzłości, potwierdza, że wiercenie w warstwami stykającymi się wspólną granicą (0-5m oraz 5-10m) są przyjmowane jako poprawne.

Testy integracyjne obejmują warstwę kontrolerów i weryfikują pełny przepływ żądania. Uruchamiane są z pełnym kontekstem aplikacji (dzięki adnotacji `@SpringBootTest`). Rolę bazy danych pełni baza danych H2 operująca w pamięci. Żądania HTTP wykonywane są za pomocą obiektu `MockMvc`, a każdy przypadek opatrzony jest adnotacją `@Transactional`, dzięki czemu zmiany wprowadzone w bazie są wycofywane po jego zakończeniu, co zapewnia scenariuszom wzajemną niezależność. Listing 47 przedstawia test integracyjny końcówki logowania weryfikujący reakcję systemu na nieprawidłowe hasło.

Listing 47. Test integracyjny końcówki logowania dla nieprawidłowych danych uwierzytelniających.
Źródło: Opracowanie własne.

```

@Test
@DisplayName("Should return 401 with invalid password")
void testLoginWithInvalidPassword() throws Exception {
    LoginRequest loginRequest = new LoginRequest("test@example.com",
        "wrongpassword");

    mockMvc.perform(post("/auth/login")
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(loginRequest)))
        .andExpect(status().isUnauthorized());
}

```

```

        .andExpect(jsonPath("$.message",
            containsString("Nieprawidłowy adres e-mail lub hasło")));
    }

```

Test wysyła żądanie logowania z poprawnym adresem e-mail i nieprawidłowym hasłem, a następnie sprawdza, że serwer zwraca kod odpowiedzi HTTP 401 oraz jednolity komunikat o nieprawidłowych danych. Analogiczny test dla nieistniejącego konta potwierdza, że odpowiedź jest identyczna.

Zestaw testów zautomatyzowanych aplikacji serwerowej obejmuje 47 klas testowych zawierających łącznie ponad 600 testów: 387 testów jednostkowych serwisów, 56 testów jednostkowych encji oraz 176 testów integracyjnych. Testy uruchamiane są lokalnie poleceniem `./mvnw test`, a ich wyniki publikowane są w formacie raportu narzędzia Surefire. Kluczowym elementem zapewniania jakości jest jednak ich automatyczne uruchamianie w ramach ciągłej integracji w repozytorium kodu. Przy wypchnięciu zmian do gałęzi głównej oraz przy każdym utworzeniu żądania scalenia platforma GitHub Actions uruchamia komplet testów i przedstawia raport z ich wykonania, co jest widoczne na Rysunek 99.

Make equipment status entries polish

Artifacts 2

Re-run all jobs

CI on: push

GitHub Actions / Test Results (ubuntu-latest, Java 21) succeeded yesterday in 0s

662 passed, 0 failed and 0 skipped

tests 662 passed

Report	Passed	Failed	Skipped	Time
target/surefire-reports/TEST-com.geokret.api.ApiApplicationTests.xml	1 ✓			14s
target/surefire-reports/TEST-com.geokret.controller.AuthControllerIntegrationTest.xml	11 ✓			6s
target/surefire-reports/TEST-com.geokret.controller.BoreholeControllerIntegrationTest.xml	19 ✓			4s
target/surefire-reports/TEST-com.geokret.controller.DiscussionControllerIntegrationTest.xml	13 ✓			2s
target/surefire-reports/TEST-com.geokret.controller.ExportControllerIntegrationTest.xml	13 ✓			3s
target/surefire-reports/TEST-com.geokret.controller.FormTemplateControllerIntegrationTest.xml	20 ✓			5s
target/surefire-reports/TEST-com.geokret.controller.HoleControllerIntegrationTest.xml	15 ✓			4s
target/surefire-reports/TEST-com.geokret.controller.KnowledgeBaseControllerIntegrationTest.xml	24 ✓			6s
target/surefire-reports/TEST-com.geokret.controller.NotificationControllerIntegrationTest.xml	14 ✓			2s
target/surefire-reports/TEST-com.geokret.controller.ProjectControllerIntegrationTest.xml	22 ✓			5s
target/surefire-reports/TEST-com.geokret.controller.SoundingControllerIntegrationTest.xml	6 ✓			1s
target/surefire-reports/TEST-com.geokret.controller.UserControllerIntegrationTest.xml	19 ✓			2s

Rysunek 99. Raport testów zautomatyzowanych po wypchnięciu zmian do gałęzi głównej na platformie GitHub. Źródło: Opracowanie własne.

7.2. Testy manualne

Testy manualne przeprowadzono metodą czarnej skrzynki, w której weryfikacja koncentruje się na zgodności zaobserwowanego zachowania systemu z oczekiwaniami, bez odwoływania się do struktury kodu. Scenariusze testowe pochodzą z przypadków użycia opisanych w podrozdziale 3.6.

Testy prowadzono na wdrożonych aplikacjach klienckich wykonując scenariusze w taki sposób jak realizowałyby je użytkownik końcowy. Działanie aplikacji serwerowej weryfikowano pośrednio,

poprzez obserwację odpowiedzi interfejsu użytkownika oraz zmian stanu danych wywołanych poszczególnymi operacjami.

Tabela 24, Tabela 25, Tabela 26 oraz Tabela 27 przedstawiają wybrane scenariusze testów manualnych obejmujące kluczowe funkcjonalności systemu.

Tabela 24. Scenariusz testowy: logowanie z nieprawidłowym hasłem. Źródło: Opracowanie własne.

Nazwa	Sprawdzenie reakcji systemu na próbę logowania z błędnymi danymi uwierzytelniającymi	
Warunek początkowy	W systemie istnieje konto użytkownika o znanym adresie e-mail	
Krok	Czynność	Oczekiwany rezultat
1	Wprowadzenie poprawnego adresu e-mail oraz nieprawidłowego hasła i zatwierdzenie logowania	System odrzuca logowanie i zwraca odpowiedź HTTP 401
2	Odczytanie komunikatu błędu	Wyświetlony zostaje jednolity komunikat „Nieprawidłowy adres e-mail lub hasło”, bez wskazania, czy błędny jest adres e-mail, czy hasło

Tabela 25. Scenariusz testowy: sprawdzenie nakładających się warstw w formularzu wiercenia. Źródło: Opracowanie własne.

Nazwa	Sprawdzenie reguły wykrywania nakładania się warstw przy zachowaniu poprawności warstw o wspólnej granicy	
Warunek początkowy	Otwarty formularz wiercenia w trybie edycji	
Krok	Czynność	Oczekiwany rezultat
1	Dodanie dwóch warstw o granicach 0–5 m oraz 5–10 m	Obie warstwy zostają zaakceptowane jako poprawne (wspólna granica jest dozwolona)
2	Zmiana drugiej warstwy na zakres 4–10 m, tak aby nakładała się na pierwszą	System zgłasza błąd nakładania się warstw i nie pozwala na oddanie badania do sprawdzenia

Tabela 26. Scenariusz testowy: oddanie niekompletnego badania do sprawdzenia. Źródło: Opracowanie własne.

Nazwa	Oddanie niekompletnego badania do sprawdzenia	
Warunek początkowy	Badanie w trakcie wypełniania, zawierające braki (np. niekompletne dane warstw)	
Krok	Czynność	Oczekiwany rezultat
1	Wybranie akcji „Oddaj do sprawdzenia” dla niekompletnego badania	System uruchamia pełne sprawdzenie formularza i odrzuca przekazanie badania

2	Odczytanie odpowiedzi systemu	Wyświetlony zostaje dialog z listą wszystkich wykrytych braków, status badania pozostaje niezmieniony
---	-------------------------------	---

Tabela 27. Scenariusz testowy: eksport karty otworu. Źródło: Opracowanie własne.

Nazwa	Eksport karty otworu	
Warunek początkowy	W projekcie istnieje otwór z kompletnym, zatwierdzonym wierceniem	
Krok	Czynność	Oczekiwany rezultat
1	Otwarcie modułu eksportu i wskazanie parametrów karty otworu	System przyjmuje parametry i przygotowuje dokument
2	Zatwierdzenie eksportu	Wygenerowany zostaje dokument karty otworu zawierający dane wiercenia zgodne z zawartością bazy

Przedstawione scenariusze stanowią wybór reprezentatywny. Analogicznym testom manualnym poddano pozostałe moduły systemu, w tym rejestracja użytkownika przez system zaproszeń, import otworów z pliku CSV, zarządzanie projektami i sprzętem, system dyskusji oraz pracę w trybie *offline* z następującą synchronizacją.

7.3. Wyniki testów

Przeprowadzone testy potwierdziły, że prototyp systemu KRET realizuje założone wymagania funkcjonalne. Zestaw zautomatyzowanych testów aplikacji serwerowej uruchamiany jest przy każdej zmianie w kodzie i sygnalizuje naruszenie istniejącej funkcjonalności zabezpieczając logikę biznesową przed regresją. Testy manualne potwierdziły poprawność kompletnych ścieżek użytkownika.

Oprócz weryfikacji technicznej przydatność rozwiązań oceniano w sposób nieformalny, w oparciu o informacje zwrotne od praktyków branży geologicznej. Jak opisano w podrozdziale 1.3, jeden z autorów, współpracujący na co dzień z osobami z branży geologicznej, na bieżąco konsultował proponowane i wdrażane funkcjonalności. Konsultacje te nie przyjęły formy kontrolowanych badań użyteczności, nie zakładano kont testowych ani nie prowadzono sesji z udziałem użytkowników operujących systemem, dostarczały jednak jakościowych informacji zwrotnych od osób znających rzeczywiste warunki pracy terenowej, które wpływały na zakres gromadzonych danych oraz kształt interfejsu.

Przeprowadzone testy nie objęły jednak wszystkich możliwych aspektów weryfikacji. Testy zautomatyzowane ograniczono jedynie do aplikacji serwerowej. Testy manualne, nie gwarantują wykrycia wszystkich przypadków brzegowych. Prototyp mobilny weryfikowano wyłącznie na platformie Android, zgodnie z zakresem prac określonym w podrozdziale 1.2.

8. Podsumowanie i wnioski

Niniejszy rozdział podsumowuje wyniki pracy nad systemem KRET. Podrozdział 8.1 zestawia osiągnięcia projektu w odniesieniu do celu określonego w sekcji 1.1, treść 8.2 omawia trudności napotkane w trakcie wytwarzania prototypu, podrozdział 8.3 przedstawia ograniczenia obecnej wersji, a sekcja 8.4 proponowane kierunki dalszego rozwoju.

8.1. Osiągnięcia projektu

Cel pracy określony w podrozdziale 1.1 został zrealizowany. Zaprojektowano i zaimplementowano działający prototyp systemu KRET, zastępujący papierową dokumentację terenowych badań geologicznych. System powstał w czterech komponentach: aplikacji serwerowej, aplikacji internetowej, aplikacji mobilnej oraz współdzielonej bibliotece modułów języka TypeScript.

W ramach prototypu zaimplementowano komplet modułów funkcjonalnych przewidzianych w zakresie pracy. Zrealizowano zarządzanie projektami, otworami i użytkownikami oraz system zleceń terenowych z wizualizacją na mapie. Geolog terenowy wypełnia formularze przypisanych badań wspierany przez pola niestandardowe, dynamiczne szablony i dokumentację fotograficzną. Proces sprawdzania dokumentacji wsparto mechanizmem automatycznego przypisywania dokumentatorów oraz systemem dyskusji umożliwiającym komunikację przy weryfikacji badania. Uzupełnieniem wspierającym pracę firm jest baza wiedzy, rozszerzalny system eksportu danych, system zaproszeń, statystyki i raporty oraz architektura aplikacji mobilnej zorientowana na pracę bez dostępu do Internetu.

Najistotniejszym efektem jest odpowiedź na lukę rynkową wskazaną w podrozdziale 1.1. Zamiast koncentrować się na pojedynczym aspekcie pracy geologa, system łączy w jednym narzędziu zbieranie danych w terenie, koordynację ich sprawdzania oraz przygotowanie do dalszej analizy. Spójność reguł sprawdzania danych pomiędzy aplikacją internetową a mobilną stanowi biblioteka współdzielona stanowiąca dla obu aplikacji klienckich źródło prawdy na temat modułów służących sprawdzaniu danych. Poprawność warstwy serwerowej potwierdza ponadto obszerny zestaw testów zautomatyzowanych omówiony w rozdziale 7.

8.2. Napotkane trudności

Realizacja systemu przebiegała w większości sprawnie. Najbardziej przekrojowym wzywaniem okazała się architektura zorientowana na pracę bez dostępu do sieci. Nie stanowiła ona pojedynczego, trudnego do rozwiązania problemu, lecz stałe założenie projektowe, które należało uwzględniać przy implementacji praktycznie każdej funkcjonalności. Wymagało to konsekwentnego pamiętania o tym wymaganiu na każdym etapie prac, zarówno po stronie przechowywania danych, jak i synchronizacji.

Drugą istotną trudnością było generowanie karty otworu w formacie PDF. W pierwotnym podejściu dokument przygotowywano po stronie przeglądarki z wykorzystaniem mechanizmu druku. Rozwiązanie to okazało się niespójne pomiędzy przeglądarkami. Różnice w sposobie wydruku prowadziły między innymi do ścinania krawędzi tabeli, co wymagało stosowania obejść, takich jak dodatkowy margines zabezpieczający oraz wielokrotnego wykonania testu druku na różnych przeglądarkach. Wnioskiem z tego problemu jest konieczność przeniesienia generowania dokumentów PDF na stronę serwera, co zapewniłoby powtarzalny proces niezależny od przeglądarki użytkownika.

8.3. Ograniczenia

Obecna wersja systemu stanowi prototyp i nie została przystosowana do pracy na skalę komercyjną. Aplikację mobilną zrealizowano wyłącznie na platformę Android i nie opublikowano w sklepie Play. Integracja z oprogramowaniem GeoStar ograniczona jest do eksportu danych w formacie CSV.

Część zaplanowanych funkcjonalności zaimplementowano w zakresie podstawowym. Rozszerzalny system eksportu danych obsługuje obecnie formaty CSV oraz XLSX, natomiast pozostałe przewidziane formaty pozostają zadeklarowane, lecz nie zaimplementowane. Generowanie karty otworu w formacie PDF odbywa się po stronie przeglądarki, co wiąże się niespójnościami pomiędzy przeglądarkami.

Zakres weryfikacji automatycznej ograniczono do aplikacji serwerowej. Aplikacje klienckie zostały objęte jedynie testami manualnymi, nie utworzono zautomatyzowanej ciągłej integracji testującej aplikacje klienckiej automatycznymi testami jednostkowymi, integracyjnym czy testami *end-to-end*.

Ograniczeniem o charakterze architektonicznym dotyczą skalowalności systemu. Wartości pól niestandardowych przechowywane są jako odrębne wiersze we wspólnej tabeli powiązane z definicją pola oraz badaniem. Przy obecnej skali rozwiązanie jest wystarczające, jednak wraz ze wzrostem liczby badań oraz zdefiniowanych pól, liczba wierszy rośnie multiplikatywnie. Niezależnie od tego system nie wprowadza ograniczeń ilościowych na zasoby zakładane przez użytkowników. W rezultacie ilość gromadzonych danych oraz obciążenie systemu mogą rosnąć w sposób nieograniczony. Wdrożenie produkcyjne wymagałoby wczesnego wdrożenia limitów i mechanizmów ochrony, w celu przyzwyczajenia użytkowników do obowiązujących ograniczeń od początku korzystania z usługi.

8.4. Propozycje dalszego rozwoju

Głównym kierunkiem dalszych prac jest komercjalizacja systemu poprzez wyjście z fazy prototypu i udostępnienie rozwiązania zainteresowanym przedsiębiorstwom z branży geologicznej. Z perspektywy inżynierskiej wymaga to przede wszystkim szeregu prac technicznych. Należą do nich:

- dostosowanie zasobów infrastruktury do skali komercyjnej wraz z wprowadzeniem limitów zasobów;
- ograniczenie częstotliwości żądań do serwera;
- przeniesienie generowania kart otworów do warstwy serwerowej;
- udostępnienie aplikacji mobilnej także na platformie iOS;
- dokończenie zaplanowanych eksportów danych.

Konieczne staje się przy tym wdrożenie monitoringu infrastruktury oraz centralnego gromadzenia i analizy logów aplikacyjnych. Pozwoliłoby to reagować na alarmy, przypadki nadmiernego zużycia zasobów oraz diagnozować błędy. Zwiększeniu niezawodności rozwiązania w środowisku produkcyjnym sprzyjałoby rozszerzeniu testów zautomatyzowanych o aplikacje klienckie, a w dalszej perspektywie uwagi wymaga również skalowalność przechowywania pól niestandardowych.

Poza obszarem inżynierskim przygotowania wymagają kwestie związane z udostępnieniem usługi, czyli oferta, polityka cenowa oraz działania marketingowe. Istotnym krokiem byłaby przy tym wczesna współpraca z użytkownikami z przedsiębiorstw branżowych, która pozwoliłaby na ustrukturyzowaną weryfikację rozwiązania w warunkach rzeczywistej pracy.

Zrealizowany prototyp stanowi solidną podstawę do dalszego rozwoju w kierunku gotowego produktu, odpowiadającego na potrzeby przedsiębiorstw prowadzących terenowe badania geologiczne.

9. Bibliografia

- [1] Norma PN-EN ISO 14688-1:2018-05 – Rozpoznanie i badania geotechniczne. Oznaczanie i klasyfikowanie gruntów. Część 1: Oznaczanie i opis. Polski Komitet Normalizacyjny.
- [2] Norma PN-EN ISO 14689:2018-05 – Rozpoznanie i badania geotechniczne. Oznaczanie, opis i klasyfikacja skał. Polski Komitet Normalizacyjny.
- [3] Norma PN-B-02480:1986 – Grunty budowlane. Określenia, symbole, podział i opis gruntów. Polski Komitet Normalizacyjny.
- [4] Ustawa z dnia 9 czerwca 2011 r. – Prawo geologiczne i górnicze. Tekst jednolity: Dz.U. 2024 poz. 1290.
- [5] Strona internetowa – Soft-Projekt, Pakiet GeoStar – Informacje – Dostęp: 03.05.2026. http://soft-projekt.com.pl/pl/teksty/geostar_1/informacje_o_pakiecie
- [6] Strona internetowa – Soft-Projekt, GeoStar – Oferta i ceny – Dostęp: 03.05.2026. <http://softproj.pnet.pl/pl/teksty/geostar>
- [7] Strona internetowa – Soft-Projekt, Strona główna – Dostęp: 03.05.2026. <http://soft-projekt.com.pl/>
- [8] Strona internetowa – Uniwersytet Warszawski, Informator ECTS, przedmiot 1300-WGSOGI „GeoStar w opracowaniach geologiczno-inżynierskich” – Dostęp: 03.05.2026. https://informatorects.uw.edu.pl/pl/courses/view?prz_kod=1300-WGSOGI
- [9] Majer E., Ryżyński G. (red.), 2022. Atlasy geologiczno-inżynierskie w skali 1:10 000 lub mniejszej. Instrukcja wykonywania. Państwowy Instytut Geologiczny – Państwowy Instytut Badawczy, Warszawa – Dostęp: 10.06.2026. <https://bazadata.pgi.gov.pl/data/agi/agi-instrukcja-2022.pdf>.
- [10] Strona internetowa – Geokrak Sp. z o.o., Pakiet programów GeoStar – Dostęp: 03.05.2026. <https://www.geokrak.pl/pl/geologia-inzynierska/prace-kameralne/10-pl/nasze-uslugi/geologia-inzynierska/171-pakiet-programow-geostar.html>
- [11] Strona internetowa – Geosoft Sp. z o.o., Oprogramowanie geotechniczne (GEO DB, CPT-pro, CPT-CAD, SEISMIC-pro) – Dostęp: 03.05.2026. <https://www.geosoft.com.pl/pl/category/oprogramowanie-geotechniczne>
- [12] Strona internetowa – Fine Software, GEO5 Geotechnical Software (programy, edycje, Fine Cloud License, Data Collector) – Dostęp: 03.05.2026. <https://www.finesoftware.eu/geotechnical-software/>
- [13] Cennik – Fine Software, GEO5 – cennik PLN obowiązujący od 1 listopada 2025 r. – Dostęp: 03.05.2026. <https://www.finesoftware.pl/data/price-lists/fine-price-list-pl-pln-geo5.pdf>
- [14] Strona internetowa – GIMSGEO Łukasz Zawadzki, Geological Information Management System – Dostęp: 03.05.2026. <https://gimsgeo.pl/>
- [15] Strona internetowa – GeoAid Sp. z o.o., Aplikacja mobilna i platforma internetowa dla geologów – Dostęp: 03.05.2026. <https://geoaid.pl/>
- [16] Strona internetowa – OPENGIS.ch, QField for QGIS – Dostęp: 03.05.2026. <https://qfield.org/>
- [17] Strona internetowa – Lutra Consulting, Mergin Maps – Open source geological mapping with British Geological Survey – Dostęp: 03.05.2026. <https://merginmaps.com/case-studies/open-source-geological-mapping-with-british-geological-survey>
- [18] Strona internetowa – British Geological Survey, BGS·SIGMA digital mapping toolkit – Dostęp: 03.05.2026. <https://www.bgs.ac.uk/technologies/software/sigma/>
- [19] Strona internetowa – Salesforce, Field Service Management Software – Dostęp: 03.05.2026. <https://www.salesforce.com/service/field-service-management/>
- [20] Strona internetowa – ServiceTitan, Field Service Dispatch Software – Dostęp: 03.05.2026. <https://www.servicetitan.com/features/dispatch-software>
- [21] Jones M., Bradley J., Sakimura N. – JSON Web Token (JWT). RFC 7519, Internet Engineering Task Force (IETF), maj 2015. – Dostęp: 25.05.2026. <https://datatracker.ietf.org/doc/html/rfc7519>

- [22] OpenAPI Initiative – OpenAPI Specification, wersja 3.0.3. The Linux Foundation, luty 2021.
– Dostęp: 25.05.2026. <https://spec.openapis.org/oas/v3.0.3>
- [23] Dokumentacja techniczna – Oracle Corporation, Java SE 21 LTS. – Dostęp: 25.05.2026.
<https://docs.oracle.com/en/java/javase/21/>
- [24] Dokumentacja techniczna – Broadcom / VMware, Spring Boot 3.4 Reference Documentation.
– Dostęp: 25.05.2026. <https://docs.spring.io/spring-boot/reference/index.html>
- [25] Dokumentacja techniczna – The Apache Software Foundation, Apache Maven. – Dostęp: 25.05.2026. <https://maven.apache.org/guides/>
- [26] Dokumentacja techniczna – The PostgreSQL Global Development Group, PostgreSQL 15 Documentation. – Dostęp: 25.05.2026. <https://www.postgresql.org/docs/15/>
- [27] Dokumentacja techniczna – PostgreSQL Project (OSGeo), PostgreSQL 3.5 Manual. – Dostęp: 25.05.2026. <https://postgis.net/docs/manual-3.5/>
- [28] Dokumentacja techniczna – Red Hat, Hibernate ORM – User Guide: Spatial. – Dostęp: 25.05.2026.
https://docs.jboss.org/hibernate/orm/current/userguide/html_single/Hibernate_User_Guide.html#spatial
- [29] Dokumentacja techniczna – Meta Platforms, React – The library for web and native user interfaces. – Dostęp: 25.05.2026. <https://react.dev/>
- [30] Dokumentacja techniczna – Vercel Inc., Next.js Documentation. – Dostęp: 25.05.2026.
<https://nextjs.org/docs>
- [31] Dokumentacja techniczna – Microsoft Corporation, TypeScript Handbook. – Dostęp: 25.05.2026. <https://www.typescriptlang.org/docs/>
- [32] Dokumentacja techniczna – Tailwind Labs, Tailwind CSS Documentation. – Dostęp: 25.05.2026. <https://tailwindcss.com/docs>
- [33] Dokumentacja techniczna – Meta Platforms, React Native – Learn once, write anywhere. – Dostęp: 25.05.2026. <https://reactnative.dev/docs/getting-started>
- [34] Dokumentacja techniczna – Expo, Expo SDK 54 Documentation. – Dostęp: 25.05.2026.
<https://docs.expo.dev/>
- [35] Agafonkin V. – Leaflet – an open-source JavaScript library for interactive maps. – Dostęp: 25.05.2026. <https://leafletjs.com/>
- [36] Strona internetowa – React Native Maps Community, react-native-maps. – Dostęp: 25.05.2026. <https://github.com/react-native-maps/react-native-maps>
- [37] Strona internetowa – OpenStreetMap Foundation, OpenStreetMap. – Dostęp: 25.05.2026.
<https://www.openstreetmap.org/about>
- [38] Dokumentacja techniczna – MinIO Inc., MinIO Object Storage for Linux – Dostęp: 10.06.2026. <https://github.com/minio/minio>
- [39] Dokumentacja techniczna – Liquibase Inc., Liquibase Documentation. – Dostęp: 25.05.2026.
<https://docs.liquibase.com/>
- [40] Zwitterloot R., Spilker R. – Project Lombok. – Dostęp: 25.05.2026.
<https://projectlombok.org/>
- [41] Dokumentacja techniczna – Broadcom / VMware, Spring Security Reference. – Dostęp: 25.05.2026. <https://docs.spring.io/spring-security/reference/>
- [42] Provos N., Mazières D. – A Future-Adaptable Password Scheme. W: Proceedings of the 1999 USENIX Annual Technical Conference, Monterey, USA, czerwiec 1999, s. 81–91. Dostęp: 10.06.2026. <https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf>
- [43] Dokumentacja techniczna – The Apache Software Foundation, Apache POI – the Java API for Microsoft Documents. – Dostęp: 25.05.2026. <https://poi.apache.org/>
- [44] Dokumentacja techniczna – Docker Inc., Docker Documentation. – Dostęp: 25.05.2026.
<https://docs.docker.com/>
- [45] Dokumentacja techniczna – F5 Inc., NGINX Documentation. – Dostęp: 25.05.2026.
<https://nginx.org/en/docs/>
- [46] Strona internetowa – Internet Security Research Group (ISRG), Let's Encrypt – Free SSL/TLS Certificates. – Dostęp: 25.05.2026. <https://letsencrypt.org/docs/>

- [47] Dokumentacja techniczna – Oracle Corporation, Oracle Cloud Infrastructure Documentation. – Dostęp: 25.05.2026. <https://docs.oracle.com/en-us/iaas/Content/home.htm>
- [48] Dokumentacja techniczna – JetBrains s.r.o., IntelliJ IDEA Documentation. – Dostęp: 25.05.2026. <https://www.jetbrains.com/help/idea/>
- [49] Strona internetowa – JGraph Ltd., diagrams.net (draw.io). – Dostęp: 25.05.2026. <https://www.drawio.com/>
- [50] Dokumentacja techniczna – GitHub Inc., GitHub Docs (GitHub Actions, GitHub Projects). – Dostęp: 25.05.2026. <https://docs.github.com/>
- [51] Dokumentacja techniczna – SmartBear Software, Swagger UI Documentation. – Dostęp: 25.05.2026. <https://swagger.io/docs/open-source-tools/swagger-ui/usage/installation/>

10. Spis rysunków

Rysunek 1. Karta otworu badawczego wygenerowana przez program GeoStar (dane identyfikujące zostały zanonimizowane). Źródło: Opracowanie własne na podstawie programu GeoStar [5].	11
Rysunek 2. Główny widok pakietu GeoStar 8i Ultimate (wersja 8.5, wydanie 2023) w konfiguracji geologiczno-inżynierskiej. Źródło: Opracowanie własne na podstawie programu GeoStar [5].	12
Rysunek 3. Moduł generowania przekrojów geologicznych w pakiecie GeoStar. Źródło: Opracowanie własne na podstawie programu GeoStar [5].	13
Rysunek 4. Główny widok programu Geosoft GEO DB. Źródło: [11].	14
Rysunek 5. Karta otworu wygenerowana przez Geosoft GEO DB z równoczesną klasyfikacją według norm PN-EN ISO 14688-1 i PN-B-02480. Źródło: [11].	15
Rysunek 6. Plik IFC wygenerowany w Geosoft GEO DB i otwarty w przeglądarce BIMvision – ilustracja natywnego wsparcia standardu BIM. Źródło: [11].	16
Rysunek 7. Aplikacja mobilna GEO5 Data Collector – edycja warstwy litologicznej w terenie. Źródło: [12].	17
Rysunek 8. Program GEO5 Stratigraphy – trójwymiarowy model podłoża gruntowego wygenerowany z danych otworów badawczych. Źródło: [12].	18
Rysunek 9. Główny widok aplikacji internetowej GIMSGEO z listą dostępnych modułów. Źródło: Opracowanie własne na podstawie programu GIMSGEO [14].	19
Rysunek 10. Moduł "Edytor mobilny" GIMSGEO w widoku przeglądarki mobilnej. Źródło: Opracowanie własne na podstawie programu GIMSGEO [14].	20
Rysunek 11. Panel GeoAid – widok statystyki i listy zrealizowanych badań. Źródło: [15].	21
Rysunek 12. Karta otworu geotechnicznego wygenerowana przez platformę GeoAid. Źródło: [15].	22
Rysunek 13. Aplikacja mobilna QField wykorzystywana do mapowania geologicznego – formularz wprowadzania obserwacji geologa strukturalnego. Źródło: [16].	23
Rysunek 14. Aplikacja mobilna Mergin Maps – edycja atrybutów obiektu terenowego oraz widok mapy z lokalizacją. Źródło: [17].	24
Rysunek 15. Interfejs systemu BGS·SIGMA – zasobniki BGS·SIGMA Mobile i BGS·SIGMA Desktop w środowisku ESRI ArcMap z otwartym oknem pomiarów strukturalnych. Źródło: [18].	25
Rysunek 16. Diagram przypadków użycia dla Gościa. Źródło: Opracowanie własne.	43
Rysunek 17. Diagram przypadków użycia dla geologa terenowego. Źródło: Opracowanie własne.	44
Rysunek 18. Diagram przypadków użycia dla Dokumentatora. Źródło: Opracowanie własne.	45
Rysunek 19. Diagram przypadków użycia dla Administratora. Źródło: Opracowanie własne.	46
Rysunek 20. Analityczny diagram klas systemu KRET. Źródło: Opracowanie własne.	53
Rysunek 21. Diagram stanów klasy Otwór. Źródło: Opracowanie własne.	54
Rysunek 22. Diagram stanów klasy Sprzęt. Źródło: Opracowanie własne.	55
Rysunek 23. Diagram aktywności – rejestracja użytkownika. Źródło: Opracowanie własne.	56
Rysunek 24. Diagram aktywności – dodawanie nowego otworu. Źródło: Opracowanie własne.	57
Rysunek 25. Diagram aktywności – tworzenie wiercenia. Źródło: Opracowanie własne.	58

Rysunek 26. Diagram aktywności – weryfikacja i zatwierdzenie wiercenia. Źródło: Opracowanie własne.	59
Rysunek 27. Diagram aktywności – eksport karty otworu. Źródło: Opracowanie własne.	60
Rysunek 28. Edytor diagramów Draw.io z otwartym diagramem klas systemu KRET. Źródło: opracowanie własne.	73
Rysunek 29. Historia uruchomień GitHub Actions w repozytorium API systemu KRET. Źródło: opracowanie własne.	74
Rysunek 30. Szczegóły przebiegu GitHub Actions w repozytorium API systemu KRET. Źródło: Opracowanie własne.	74
Rysunek 31. Dokumentacja interfejsu API systemu KRET wygenerowana w Swagger UI. Źródło: opracowanie własne.	75
Rysunek 32. Diagram architektury wdrożeniowej systemu KRET. Źródło: Opracowanie własne.	76
Rysunek 33. Diagram potoku ciągłej integracji w repozytorium API. Źródło: Opracowanie własne.	80
Rysunek 34. Diagram sekwencji uwierzytelniania żądań. Źródło: Opracowanie własne.	83
Rysunek 35. Ekran profilu użytkownika w aplikacji mobilnej. Źródło: Opracowanie własne.	88
Rysunek 36. Ekran wysyłania zaproszenia w aplikacji internetowej. Źródło: Opracowanie własne.	89
Rysunek 37. Diagram sekwencji wgrywania i udostępniania pliku. Źródło: Opracowanie własne.	93
Rysunek 38. Ekran modułu importu listy encji do systemu KRET. Źródło: Opracowanie własne.	100
Rysunek 39. Ekran edycji warstwy w formularzu wiercenia w aplikacji mobilnej. Źródło: Opracowanie własne.	102
Rysunek 40. Ekran formularza wiercenia w aplikacji mobilnej z listą warstw geologicznych. Źródło: Opracowanie własne.	103
Rysunek 41. Ekran formularza sondowania w aplikacji mobilnej. Źródło: Opracowanie własne.	104
Rysunek 42. Panel definiowania pól niestandardowych w aplikacji internetowej. Źródło: Opracowanie własne.	105
Rysunek 43. Okno wyboru wzorca formularza w aplikacji mobilnej. Źródło: Opracowanie własne.	106
Rysunek 44. Diagram działania algorytmu automatycznego przypisywania dokumentatorów. Źródło: Opracowanie własne.	110
Rysunek 45. Ekran listy powiadomień w aplikacji mobilnej. Źródło: Opracowanie własne.	113
Rysunek 46. Ekran synchronizacji danych w aplikacji mobilnej. Źródło: Opracowanie własne.	123
Rysunek 47. Zależności między repozytoriami systemu KRET. Źródło: Opracowanie własne.	126
Rysunek 48. Ekran startowy aplikacji internetowej dla niezalogowanego użytkownika. Źródło: Opracowanie własne.	127
Rysunek 49. Formularz rejestracji konta otwierany po kliknięciu odnośnika z zaproszenia. Źródło: Opracowanie własne.	128
Rysunek 50. Ekran logowania do aplikacji internetowej. Źródło: Opracowanie własne.	128
Rysunek 51. Panel zarządzania zaproszeniami w aplikacji internetowej. Źródło: Opracowanie własne.	129

Rysunek 52. Panel główny aplikacji internetowej. Źródło: Opracowanie własne.....	130
Rysunek 53. Lista projektów z mechanizmem filtrowania. Źródło: Opracowanie własne....	130
Rysunek 54. Formularz tworzenia nowego projektu. Źródło: Opracowanie własne.....	131
Rysunek 55. Widok szczegółowy projektu w aplikacji internetowej. Źródło: Opracowanie własne.....	132
Rysunek 56. Przypisywanie geologów do projektu. Źródło: Opracowanie własne.....	132
Rysunek 57. Mapa projektu z naniesionymi otworami badawczymi w aplikacji internetowej. Źródło: Opracowanie własne.	133
Rysunek 58. Formularz tworzenia nowego otworu z mini-mapą do wskazania lokalizacji. Źródło: Opracowanie własne.	133
Rysunek 59. Widok szczegółowy otworu w aplikacji internetowej. Źródło: Opracowanie własne.....	134
Rysunek 60. Mapa otworów w projekcie w aplikacji mobilnej. Źródło: Opracowanie własne.	135
Rysunek 61. Podręczne menu szybkich akcji otworu w aplikacji mobilnej. Źródło: Opracowanie własne.	136
Rysunek 62. Widok szczegółów otworu w aplikacji mobilnej. Źródło: Opracowanie własne.	137
Rysunek 63. Wybór typu badania w aplikacji mobilnej. Źródło: Opracowanie własne.....	138
Rysunek 64. Lista otworów projektu w aplikacji mobilnej. Źródło: Opracowanie własne...	139
Rysunek 65. Szczegóły otworu z danymi dokumentatora w aplikacji mobilnej. Źródło: Opracowanie własne.	140
Rysunek 66. Formularz wiercenia w aplikacji mobilnej. Źródło: Opracowanie własne.	141
Rysunek 67. Okno wyboru wzorca formularza w aplikacji mobilnej. Źródło: Opracowanie własne.....	142
Rysunek 68. Okno dodawania i edycji warstwy litologicznej w aplikacji mobilnej. Źródło: Opracowanie własne.	143
Rysunek 69. Dokumentacja fotograficzna warstwy litologicznej w aplikacji mobilnej. Źródło: Opracowanie własne.	144
Rysunek 70. Formularz sondowania w aplikacji mobilnej. Źródło: Opracowanie własne...	145
Rysunek 71. Sekcja pól niestandardowych w formularzu sondowania w aplikacji mobilnej. Źródło: Opracowanie własne.	146
Rysunek 72. Formularz tworzenia wzorca formularza w aplikacji internetowej. Źródło: Opracowanie własne.	147
Rysunek 73. Skrzynka powiadomień w aplikacji internetowej. Źródło: Opracowanie własne.	148
Rysunek 74. Widok badania w trybie weryfikacji przez dokumentatora w aplikacji internetowej. Źródło: Opracowanie własne.	149
Rysunek 75. Dyskusja powiązana z polem formularza w aplikacji internetowej. Źródło: Opracowanie własne.	150
Rysunek 76. Odpowiedź geologa terenowego w wątku dyskusji w aplikacji mobilnej. Źródło: Opracowanie własne.	150
Rysunek 77. Kafelki wskaźników na pulpicie aplikacji internetowej. Źródło: Opracowanie własne.....	151
Rysunek 78. Kalendarz prac na pulpicie aplikacji internetowej. Źródło: Opracowanie własne.	151
Rysunek 79. Ekran raportów w aplikacji internetowej. Źródło: Opracowanie własne.....	152

Rysunek 80. Statystyki aktywności w profilu geologa w aplikacji mobilnej. Źródło: Opracowanie własne.	153
Rysunek 81. Lista artykułów bazy wiedzy z filtrami i tagami w aplikacji internetowej. Źródło: Opracowanie własne.	154
Rysunek 82. Widok pojedynczego artykułu bazy wiedzy w aplikacji internetowej. Źródło: Opracowanie własne.	154
Rysunek 83. Formularz tworzenia i edycji artykułu bazy wiedzy w aplikacji internetowej. Źródło: Opracowanie własne.	155
Rysunek 84. Zarządzanie kategoriami bazy wiedzy w aplikacji internetowej. Źródło: Opracowanie własne.	156
Rysunek 85. Zarządzanie tagami bazy wiedzy w aplikacji internetowej. Źródło: Opracowanie własne.	156
Rysunek 86. Lista artykułów bazy wiedzy w aplikacji mobilnej. Źródło: Opracowanie własne.	157
Rysunek 87. Lista sprzętu firmy z bieżącymi statusami w aplikacji internetowej. Źródło: Opracowanie własne.	158
Rysunek 88. Zmiana statusu sprzętu w aplikacji internetowej. Źródło: Opracowanie własne.	159
Rysunek 89. Historia statusu sprzętu w aplikacji internetowej. Źródło: Opracowanie własne.	159
Rysunek 90. Rezerwacja sprzętu w aplikacji internetowej. Źródło: Opracowanie własne....	160
Rysunek 91. Lista złożonych rezerwacji sprzętu w aplikacji mobilnej. Źródło: Opracowanie własne.	161
Rysunek 92. Rezerwacje sprzętu w aplikacji mobilnej. Źródło: Opracowanie własne.	162
Rysunek 93. Okno eksportu danych projektu w aplikacji internetowej. Źródło: Opracowanie własne.	163
Rysunek 94. Konfiguracja eksportu danych XLSX w aplikacji internetowej. Źródło: Opracowanie własne.	164
Rysunek 95. Konfiguracja karty otworu w aplikacji internetowej. Źródło: Opracowanie własne.	165
Rysunek 96. Wygenerowana karta otworu w systemie KRET. Źródło: Opracowanie własne.	166
Rysunek 97. Okno importu danych z ustawieniami odczytu pliku w aplikacji internetowej. Źródło: Opracowanie własne.	167
Rysunek 98. Podgląd zawartości importowanego pliku z oznaczeniem błędów. Źródło: Opracowanie własne.	168
Rysunek 99. Raport testów zautomatyzowanych po wypchnięciu zmian do gałęzi głównej na platformie GitHub. Źródło: Opracowanie własne.	171

11. Spis tabel

Tabela 1. Podział prac pomiędzy autorów. Źródło: opracowanie własne.....	8
Tabela 2. Porównanie systemów pod względem cech dokumentacyjnych. Źródło: Opracowanie własne.	26
Tabela 3. Porównanie systemów pod względem cech operacyjnego zarządzania pracami terenowymi. Źródło: Opracowanie własne.	27
Tabela 4. Wymagania funkcjonalne modułu panelu użytkownika. Źródło: Opracowanie własne.	32
Tabela 5. Wymagania funkcjonalne modułu panelu projektu. Źródło: Opracowanie własne.	33
Tabela 6. Wymagania funkcjonalne modułu panelu otworu. Źródło: Opracowanie własne. ...	34
Tabela 7. Wymagania funkcjonalne modułu panelu odwiertu. Źródło: Opracowanie własne.	35
Tabela 8. Wymagania funkcjonalne modułu panelu sondowania. Źródło: Opracowanie własne.	36
Tabela 9. Wymagania funkcjonalne modułu pól niestandardowych. Źródło: Opracowanie własne.	37
Tabela 10. Wymagania funkcjonalne modułu wzorców formularzy. Źródło: Opracowanie własne.	37
Tabela 11. Wymagania funkcjonalne modułu sprzętu. Źródło: Opracowanie własne.	38
Tabela 12. Wymagania funkcjonalne modułu bazy wiedzy. Źródło: Opracowanie własne. ...	39
Tabela 13. Wymagania funkcjonalne modułu panelu zaproszeń. Źródło: Opracowanie własne.	40
Tabela 14. Wymagania niefunkcjonalne. Źródło: Opracowanie własne.	41
Tabela 15. Scenariusz przepływu zdarzeń przypadku „Rejestracja użytkownika”. Źródło: Opracowanie własne.	47
Tabela 16. Scenariusz przepływu zdarzeń przypadku „Tworzenie nowego otworu”. Źródło: Opracowanie własne.	48
Tabela 17. Scenariusz przepływu zdarzeń przypadku „Wypełnianie formularza wiercenia”. Źródło: Opracowanie własne.	48
Tabela 18. Scenariusz przepływu zdarzeń przypadku „Importowanie wiercenia z pliku”. Źródło: Opracowanie własne.	49
Tabela 19. Scenariusz przepływu zdarzeń przypadku „Praca w trybie offline”. Źródło: Opracowanie własne.	50
Tabela 20. Scenariusz przepływu zdarzeń przypadku „Weryfikacja i zatwierdzenie badania”. Źródło: Opracowanie własne.	51
Tabela 21. Scenariusz przepływu zdarzeń przypadku „Eksport karty otworu”. Źródło: Opracowanie własne.	52
Tabela 22. Tabela uprawnień ról użytkowników w systemie KRET. Źródło: Opracowanie własne.	87
Tabela 23. Typy powiadomień w systemie KRET. Źródło: Opracowanie własne.	113
Tabela 24. Scenariusz testowy: logowanie z nieprawidłowym hasłem. Źródło: Opracowanie własne.	172
Tabela 25. Scenariusz testowy: sprawdzenie nakładających się warstw w formularzu wiercenia. Źródło: Opracowanie własne.	172
Tabela 26. Scenariusz testowy: oddanie niekompletnego badania do sprawdzenia. Źródło: Opracowanie własne.	172
Tabela 27. Scenariusz testowy: eksport karty otworu. Źródło: Opracowanie własne.	173

12. Spis listingów

Listing 1. Przykładowa definicja kontrolera REST w Spring Boot. Źródło: [24]	62
Listing 2. Mapowanie pola geometrycznego w encji JPA z wykorzystaniem Hibernate Spatial. Źródło: [28]	63
Listing 3. Przykładowy komponent funkcyjny w React z typami TypeScript. Źródło: [29]...	64
Listing 4. Przykładowe definicje typów w TypeScript: typ unii oraz interfejs. Źródło: [31]..	65
Listing 5. Przykładowy komponent React Native z wykorzystaniem natywnych elementów interfejsu. Źródło: [33]	66
Listing 6. Przykładowa interaktywna mapa z wykorzystaniem React Leaflet i OpenStreetMap. Źródło: [35].	67
Listing 7. Konfiguracja klienta MinIO w aplikacji Spring Boot. Źródło: [38].	68
Listing 8. Przykładowy changeset Liquibase dodający kolumnę do tabeli. Źródło: [39].	69
Listing 9. Przykładowa konfiguracja łańcucha filtrów w Spring Security. Źródło: [41].	70
Listing 10. Tworzenie arkusza Excel z wykorzystaniem biblioteki Apache POI. Źródło: [43].	71
Listing 11. Przykładowy wieloetapowy plik Dockerfile dla aplikacji Spring Boot. Źródło: [44].	72
Listing 12. Fragment konfiguracji Docker Compose. Źródło: Opracowanie własne.	77
Listing 13. Plik konfiguracyjny dla ciągłej integracji GitHub Actions. Źródło: Opracowanie własne.	79
Listing 14. Fragment migracji Liquibase tworzącej tabelę otworów badawczych. Źródło: Opracowanie własne.	81
Listing 15. Fragment encji Hole z mapowaniem lokalizacji przestrzennej. Źródło: Opracowanie własne.	82
Listing 16. Fragment metody login w klasie AuthService. Źródło: Opracowanie własne.	84
Listing 17. Konfiguracja łańcucha filtrów Spring Security. Źródło: Opracowanie własne.	84
Listing 18. Filtr JwtTokenFilter sprawdzający tokeny dostępowe. Źródło: Opracowanie własne.	85
Listing 19. Operacje na tokenach odświeżania w klasie RefreshTokenService. Źródło: Opracowanie własne.	86
Listing 20. Tworzenie i wysyłka zaproszenia. Źródło: Opracowanie własne.	89
Listing 21. Konfiguracja dwóch klientów MinIO. Źródło: Opracowanie własne.	91
Listing 22. Metoda wgrywania pliku do MinIO. Źródło: Opracowanie własne.	91
Listing 23. Generowanie wstępnie podpisanego adresu URL. Źródło: Opracowanie własne.	92
Listing 24. Metoda assignUsersToHole serwisu HoleService odpowiedzialna za przypisywanie geologów do otworu. Źródło: Opracowanie własne.	94
Listing 25. Metoda assignUsersToProject z ProjectService. Źródło: Opracowanie własne.	96
Listing 26. Metoda manuallyAssignDocumenter odpowiedzialna za przypisywanie dokumentatorów do otworu w celu ich rewizji. Źródło: Opracowanie własne.	96
Listing 27. Metoda importHoles (wraz z validateHoleRequests) odpowiedzialna za import otworów do systemu. Źródło: Opracowanie własne.	98
Listing 28. Metoda createOrReplaceMultipleBoreholes odpowiedzialna za seryjne tworzenie lub zastępowanie encji Borehole. Źródło: Opracowanie własne.	99
Listing 29. Metoda createOrReplaceBorehole odpowiedzialna za tworzenie jednej encji wiercenia. Źródło: Opracowanie własne.	99
Listing 30. Komponent MapContent odpowiedzialny za wizualizację mapową. Źródło: Opracowanie własne.	101

Listing 31. Mechanizm automatycznego zapisu w haku useBoreholeForm. Źródło: Opracowanie własne.	106
Listing 32. Sprawdzenie zestawu zdjęć panoramicznych 360° w bibliotece współdzielonej. Źródło: Opracowanie własne.	107
Listing 33. Fragment sprawdzenia danych wiercenia przed oddaniem badania. Źródło: Opracowanie własne.	108
Listing 34. Algorytm automatycznego przypisywania dokumentatorów. Źródło: Opracowanie własne.	110
Listing 35. Metoda createDiscussion odpowiedzialna za tworzenie dyskusji pod określonym polem formularza badania. Źródło: Opracowanie własne.	111
Listing 36. Tworzenie i wysyłka powiadomienia o zmianie statusu otworu. Źródło: Opracowanie własne.	114
Listing 37. Metoda exportProject odpowiedzialna za eksport plików z systemu. Źródło: Opracowanie własne.	115
Listing 38. Fragment metody collectExportData odpowiedzialnej za agregację encji objętych eksportem. Źródło: Opracowanie własne.	115
Listing 39. Metoda createArticle odpowiedzialna za tworzenie artykułu w bazie wiedzy. Źródło: Opracowanie własne.	117
Listing 40. Metoda publishArticle odpowiedzialna za publikację uprzednio utworzonego artykułu. Źródło: Opracowanie własne.	118
Listing 41. Fragment metody getOverview odpowiedzialnej za wyznaczenie statystyk firmy. Źródło: Opracowanie własne.	119
Listing 42. Metoda createReservation odpowiedzialna za tworzenie rezerwacji sprzętu. Źródło: Opracowanie własne.	121
Listing 43. Dodawanie operacji do kolejki offline z deduplikacją. Źródło: Opracowanie własne.	122
Listing 44. Interfejs StorageAdapter i opakowanie TypedStorage. Źródło: Opracowanie własne.	124
Listing 45. Współdzielona obietnica odświeżenia tokenu. Źródło: Opracowanie własne.	125
Listing 46. Test jednostkowy reguły walidacji nakładania się warstw geologicznych. Źródło: Opracowanie własne.	169
Listing 47. Test integracyjny końcówki logowania dla nieprawidłowych danych uwierzytelniających. Źródło: Opracowanie własne.	170