POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Michał Ryśkiewicz
Nr albumu s24014

System zarządzania ankietami w podejściu bezserwerowym

Praca magisterska

Dr inż. Mariusz Trzaska

Warszawa, luty, 2023

Streszczenie

Niniejsza praca dotyczy tematyki tworzenia i zarządzania ankietami. Bazuje ona na obecnie popularnych rozwiązaniach. Podzielone są one odpowiednio pod kątem specyficznych architektur i zastosowanych rozwiązań. Na bazie tej analizy powstał prototyp aplikacji uzupełniający lukę systemów powstałą na rynku.

W dalszej części omówione są wykorzystane w prototypie rozwiązania w tym m.in. wybrane technologie, oraz przyjęte opcje architektoniczne. Dodatkowo pod względem analizy rozwiązań przytoczone zostały strategie wykorzystywane przez różnych dostawców chmurowych.

Na zakończenie pracy przedstawione zostały możliwe kierunki rozwoju prototypu. Rozwiązania te garściami czerpią z obecnych nowoczesnych zastosowań, a także przedstawione są przykłady potencjalnego ich zastosowania.

Spis Treści

1. Wstęp

Dane są paliwem XXI wieku brzmią w takim lub podobnym tonie nagłówki wielu artykułów, wystąpień i dyskusji dotyczących m.in. szeroko rozumianego biznesu jak i zagadnień związanych z przetwarzaniem danych i uczenia maszynowego [1], [2], [3]. W obecnym coraz bardziej zdigitalizowanym świecie widzimy większą potrzebę zbierania danych od użytkowników. Informacje użytkownicy mogą przekazywać w sposób świadomy np. w postaci ankiet lub w sposób nieświadomy zbierając np. godziny wejścia na stronę, lokalizacje, czy ilość czasu spędzoną na danej stronie. Niektóre z firm, np. Meta opiera znaczną część swojego modelu biznesowego w oparciu właśnie o zbieranie i przetwarzanie danych [4], [5]. Meta wykorzystuje pozyskiwanie informacji, np. poprzez analizę polubień użytkowników, wrzucane przez nich zdjęcia, czy nagrania, jak również zbierając metadane, np. ciasteczka z innych stron znajdujących się w przeglądarce użytkownika.

Pomijając technologicznych gigantów takich jak w.w. Meta, mamy dość dużą ilość firm, organizacji czy osób znanych, którzy nie mogą sobie pozwolić na utrzymanie dedykowanych działów analizy danych. Jednym z najprostszych sposobów otrzymania informacji dla niniejszych przedsięwzięć o oczekiwaniach wśród swojej grupy odbiorczej jest stworzenie dedykowanej ankiety w celu prześledzenia danych trendów.

W związku z powyższym za cel swojej pracy dyplomowej postawiłem sobie analizę dostępnych na rynku narzędzi do tworzenia i zarządzania ankietami. Dodatkowo w ramach niniejszego porównania stworzyłem prototyp bezsewerowego systemu ankietowego, który mógłby uzupełnić pewną lukę powstałą na tejże arenie. Aplikacja w głównej mierze nakierowana jest na wykorzystanie nowoczesnych technologii tak, aby mogła ona spełnić oczekiwania dzisiejszego rynku. Strona internetowa została oparta na architekturze REST API, a także wykorzystuje nowoczesne rozszerzenia języka JavaScript/TypeScript takie jak biblioteka React.js
i środowisko uruchomieniowe Node.js (w wersji bezserwerowych funkcji). Do przechowywania danych posłużyłem się nierelacyjną bazą danych (NoSQL) Firestore.

Podział niniejszej pracy jest następujący. W rozdziale drugim przedstawione są narzędzia i gotowe systemy, które wykorzystywane są przy zarządzaniu ankietami. Rozdział trzeci traktuje o poszczególnych narzędziach wykorzystanych do stworzenia prototypu bezserwerowego systemu zarządzania ankietami. W rozdziale czwartym znajdują się informacje o inspiracji, możliwościach, oraz wadach i zaletach stworzonego na rzecz pracy systemu. W przedostatnim rozdziale umieszczone zostały sprecyzowane informacje dotyczące architektury sys-

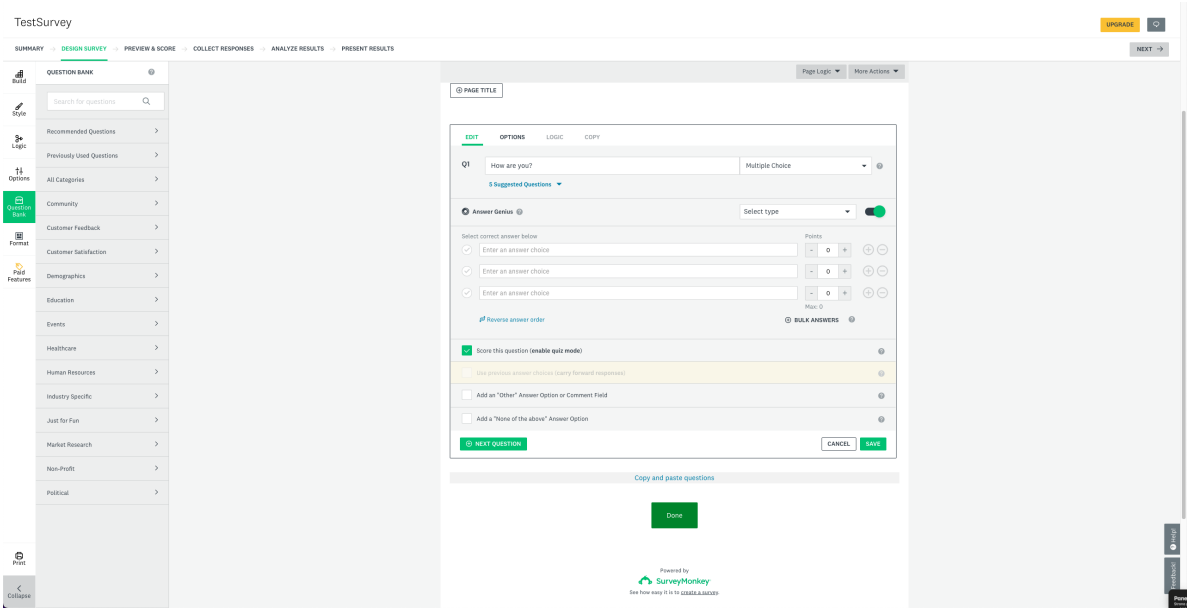
temu, z jakich technologii korzysta i jakie są ich zalety i wady, a także zostały opisane potencjalne możliwości rozwoju i usprawniania aplikacji. W ostatnim rozdziale umieszczone zostało podsumowanie pracy i ogólna ocena narzędzi. Na ich podstawie sporządzone zostały wnioski.

2. Różnorodność systemów oraz ich możliwości

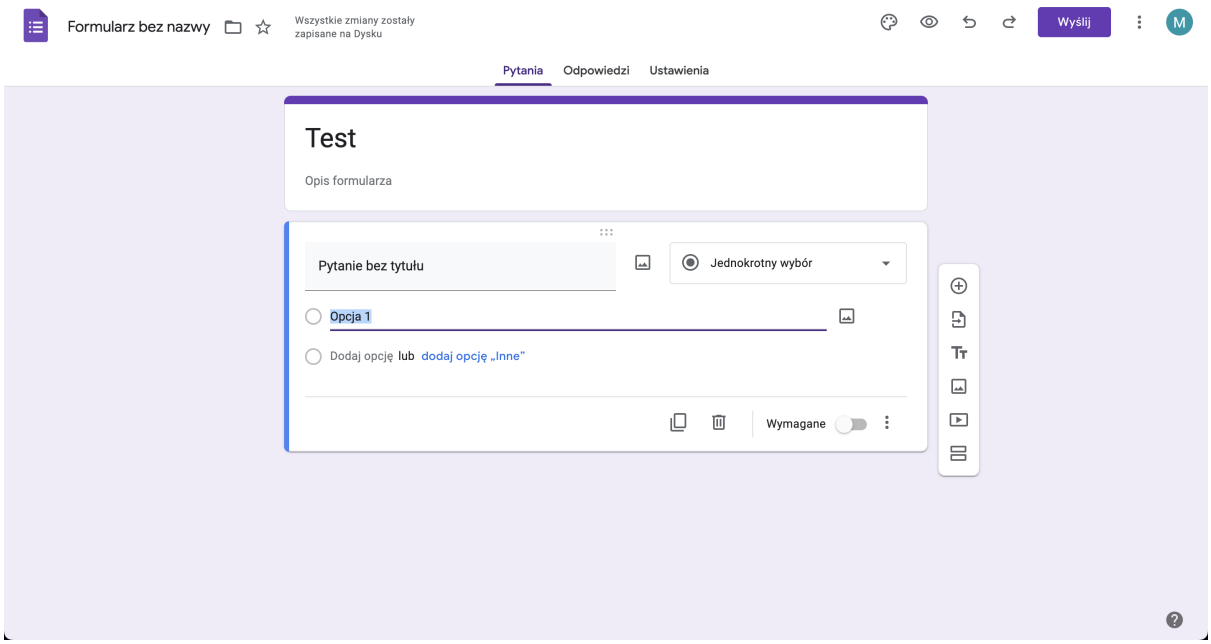
Obecnie analiza danych jest podstawą wielu biznesów. Firmy zbierają różnorodne zbiory informacji, dostosowują je do własnych potrzeb, a następnie analizują tak by czerpać z nich jak największe korzyści przy doborze odpowiedniej strategii biznesowej [6]. Ze względu na różnorodność zbieranych danych, tak samo odmienne są systemy wykorzystywane przez konkretne firmy w celu zebrania informacji. Niektóre z nich korzystają z zaawansowanych dedykowanych aplikacji, których architektura jest skomplikowana, a analiza danych [7] przeprowadzana jest przez zespoły pracowników. Użytkownicy, którzy nie posiadają takich wymagań skierować się mogą w popularne i szeroko dostępne narzędzia lub gotowe systemy, które zostały opisane w dalszej części rozdziału.

2.1. Systemy ankietowe typu platformy jako usługi

Na rynku istnieją już platformy do tworzenia i zarządzania ankietami działających na zasadach platformy jako usługi. Systemy bardzo często wykorzystują subskrypcję udostępnionych dla użytkowników wybranych zasobów. Wachlarz udostępnionych funkcjonalności jest zróżnicowany i różni się w zależności od wybranej aplikacji. Jedną z bardziej popularnych aplikacji [8] jest Surveymonkey, w której tworzenie przykładowej ankiety zostało uwiecznione na rysunku 2.1. Strona ta w porównaniu do konkurencji [9], którym jest Google Forms, udostępnia wiele zaawansowanych możliwości m.in. zaznaczenia prawidłowych odpowiedzi w ankietach, więcej opcji stylowania czy stworzenie opcji płatności w ankiecie. Różnice pomiędzy tworzeniem przykładowej ankiety można zobaczyć porównując rysunek 2.1. z rysunkiem 2.2. w którym został stworzony przykładowy formularz w aplikacji Google Forms. Widzimy na nim prostszy wizualnie system, ale również uboższy w kwestii dostosowania dla bardziej wymagających użytkowników.

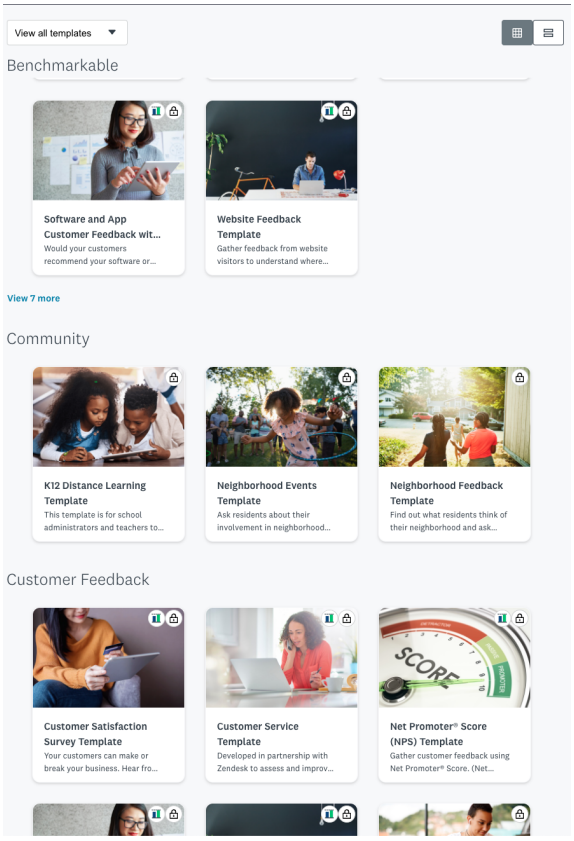


Rysunek 2.1. Formularz tworzenia ankiety w programie Surveymonkey. Źródło: [10]



Rysunek 2.2. Formularz tworzenia ankiety w aplikacji Google Forms. Źródło: [11]

Dodatkowo aplikacja udostępnia gotowe szablony ankiet, przedstawione na rysunku 2.3. Dzięki przygotowanym wcześniej wzorom ankiet użytkownik może wybrać jeden z wielu gotowych formularzy, aby zacząć zbierać potrzebne dane.



Rysunek 2.3. Przykładowe szablony ankiet w aplikacji Surveymonkey. Źródło: [12]

Bardzo istotną częścią platform typu usługi są też opcje mające za zadanie zwiększyć zasięg danych ankiet, np. poprzez możliwość wstrzyknięcia ankiety do naszego już istniejącego systemu lub poprzez wstrzyknięcie ankiet do aplikacji mobilnych. Nierzadko platformy takie udostępniają również narzędzia do analizy zebranych wyników, np. poprzez graficzne przedstawienie trendów, eksportu danych do formatów .csv czy .spss.

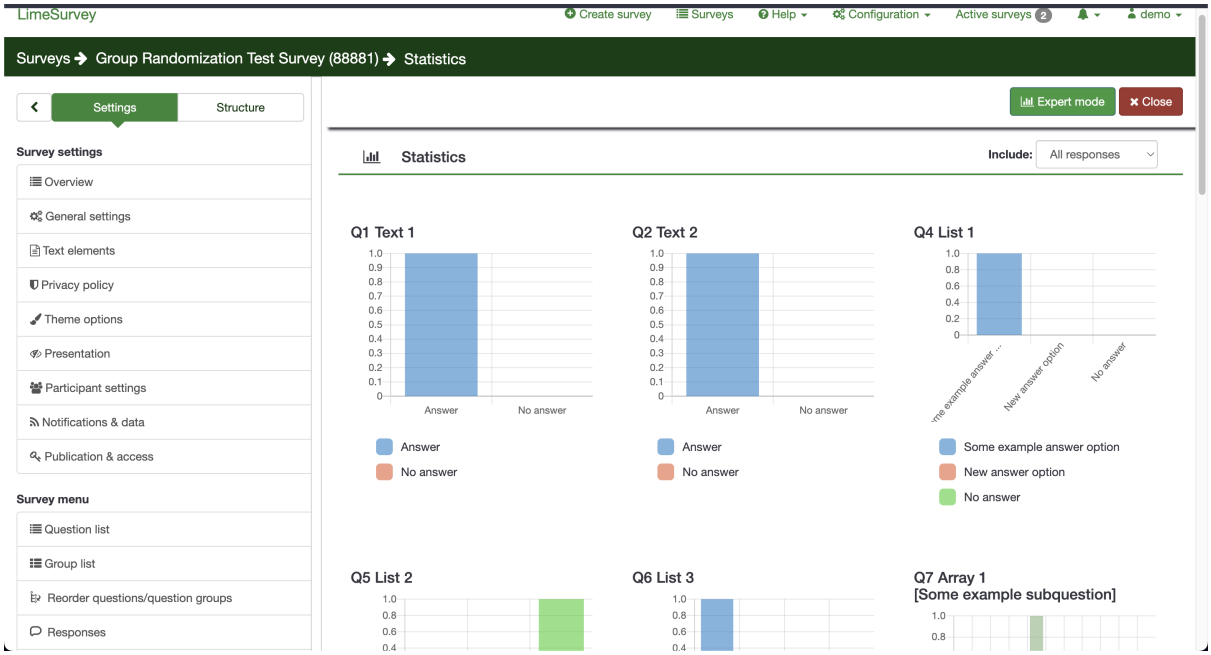
Pod względem kwestii technicznych dodatkową zaletą jest również zniesienie odpowiedzialności z użytkownika za konieczność samodzielnego utrzymywania i zarządzania danym systemem. W takich systemach użytkownik nie musi zajmować się administracją serwerów, tworzenia backupów, czy zabezpieczenia w odpowiedni sposób baz danych oraz newralgicznych punktów w systemie. Aplikacje typu platform jako usługi w głównej mierze tym faktem zdobyły swoją popularność. Zwiększenie popularności osiągnęły również niewielkimi wymaganiami wiedzy technicznej, aby zacząć korzystać z danej platformy. Dodatkową zaletą tych systemów okazuje się również możliwość ręcznego (lub automatycznego) skalowania systemów, np. pod kątem zwiększenia liczby tworzonych ankiet w zależności od potrzeb użytkownika.

2.2. Otwartoźródłowe aplikacje ankietowe w podejściu serwerowym

Innym podejściem do tworzenia i zarządzania ankietami przejawia się możliwość oparta o otwartoźródłowe (ang. *opensource*) aplikacje, którym należy zapewnić odpowiednie zasoby, np. przestrzeń dyskową na dedykowanym serwerze, odpowiednią bazę danych czy przekazać odpowiednio przygotowane dane wejściowe do serwerów mailowych. Aplikacje te w przeciwieństwie do aplikacji typu usługi najczęściej wymagają od użytkownika specjalnej wiedzy technicznej. Umiejętności, które powinna posiadać osoba konfigurująca taką platformę bardzo często różnią się od siebie ze względu na wykorzystywane przez aplikację technologie. Powoduje to, że proces wdrożenia platformy do wykorzystania na produkcyjnym środowisku potrafi być skomplikowany i czasochłonny.

Pomimo potrzeby samodzielnego konfigurowania systemu, własnoręczna konfiguracja okazać się może niezwykle potrzebna przy wyspecjalizowanych systemach. Główną zaletą jest możliwość podłączenia takiej aplikacji pod już istniejącą infrastrukturę, co otwiera wachlarz możliwości. Dzięki takiemu podejściu możemy, np. mieć natychmiastowy dostęp do wcześniej przygotowanego zbioru danych (np. różnego rodzaju metadanych), mamy pełną kontrolę nad danym systemem - możemy go niezależnie dostosowywać do naszych potrzeb i nie być uzależnionym od zmian wprowadzanych przez usługodawcę. Dodatkową zaletą okazać się może udostępnienie kodu źródłowego do wglądu publicznego. Dzięki takiemu zabiegowi społeczność danego narzędzia może szybciej wprowadzać zmiany i naprawiać błędy powstałe podczas tworzenia oprogramowania.

Jedną z przykładowych otwartoźródłowych aplikacji ankietowych jest Limesurvey [13]. System ten stworzony został przez grupę ludzi [14] i udostępniony na zasadach otwartoźródłowej licencji GPL 2.0 [15]. Aplikacja udostępnia podstawowe narzędzia do pracy z ankietami w tym m.in. tworzenie i edycja ankiet, możliwość udostępniania i dostosowywania ankiet, a także możliwość eksportu i wyświetlenia danych zebranych z ankiet, co prezentuje rysunek 2.4.



Rysunek 2.4. Przykładowe statystyki z ankiet w programie Limesurvey. Źródło: [16]

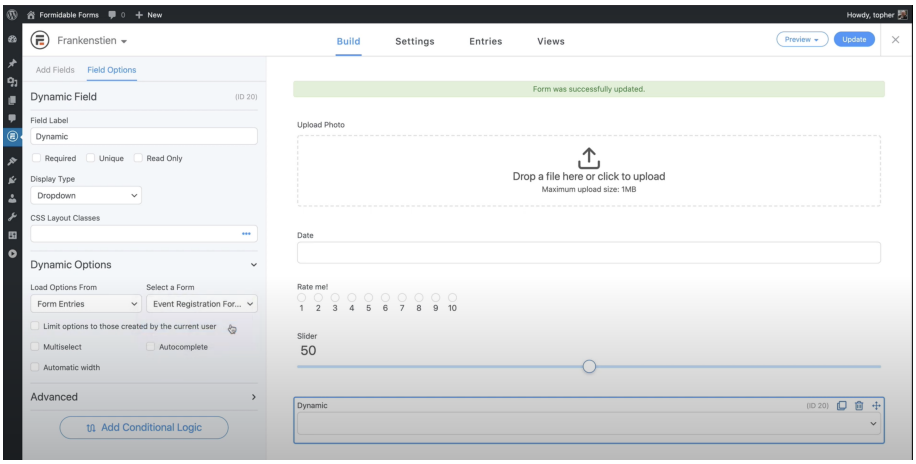
Warto wspomnieć również o wadach takiego rozwiązania jak przedstawione wyżej Limesurvey. Pomimo, że aplikacja cały czas jest utrzymywana i rozwijana, należy pamiętać o kwestii cyberbezpieczeństwa. W przypadku udostępniania kodu źródłowego osoby niepożądane mają ułatwiony dostęp do informacji o potencjalnych niezgodnościach w kodzie źródłowym i mogłyby chcieć wykorzystać tę lukę w ramach tzw. *"day-zero-exploit"* [17]. W związku z tym dodatkową kwestią o jaką należy zadbać przy wykorzystywaniu otwartoźródłowych systemów jest monitorowanie, aktualizowanie a także zabezpieczenia wykorzystywanego oprogramowania.

2.3. Aplikacje ankietowe w podejściu serwerowym

W poprzednim rozdziale zostały omówione programy z otwartoźródłową licencją, które to swój kod udostępniają publicznie, a ich utrzymanie jest w pełni zależne od użytkownika. W tym rozdziale zostały omówione programy które również wymagają od użytkownika samodzielnej infrastruktury jednak ich kod nie jest udostępniony w internecie.

Według danych udostępnionych przez *W3Techs* [18], aż 43% stron internetowych w internecie stworzonych zostało przy wykorzystaniu narzędzia *Wordpress*. Z racji swojej popularności do niniejszego oprogramowania powstało wiele rozszerzeń w tym również te dotyczące tworzenia i zarządzania ankietami.

Rysunek 2.5. przedstawia wtyczkę "Formidable Forms". Pozwala ona m.in. na tworzenie ankiet i formularzy, oraz posiada możliwość wizualizacji otrzymanych wyników. Ze względu na szerokie spektrum zastosowania narzędzie wyposażone jest w wiele podobnych rozwiązań [19] udostępnionych np. w programie Surveymonkey. Zaliczyć możemy do nich m.in. możliwość skorzystania z gotowych szablonów, wizualizacji wyników czy importowania danych z plików o rozszerzeniu .csv.



Rysunek 2.5. Tworzenie ankiety z wykorzystaniem wtyczki "Formidable Forms". Źródło: [20]

Takie rozwiązanie w porównaniu do otwartoźródłowych systemów ankietowych częściowo zwalnia użytkownika z monitorowania zmian kodu wprowadzanych do repozytorium. Pamiętać jednak należy, że konieczność aktualizowania oprogramowania w celu zachowania bezpieczeństwa aplikacji, ciągle pozostaje po stronie administratora systemu.

2.4. Rozwiązania chmurowe i hybrydowe

Niniejszy rozdział warto rozpocząć od kwestii nawiązującej do tytułu pracy. Mianowicie czym jest bezserwerowość. Zgodnie z informacjami zawartymi na stronie firmy Oracle [21] jest to sposób uruchomienia aplikacji w taki sposób by zwolnić użytkownika z konieczności zarządzania fizycznymi serwerami. Ze względu na możliwą szerszą interpretację "przetwarzania bezserwerowego", w dalszej części pracy termin ten odnosi się do wyżej wymienionej definicji.

Warto zaznaczyć, iż przetwarzanie bezserwerowe jest ściśle skorelowane z rozwiązaniami chmurowymi. Bazując na artykule *"Why Is Cloud Computing Growing in Popularity"* [22], zyskują one coraz to większą popularność. Spowodowane jest to wieloma czynnikami,

a kluczowe w tej kwestii są m.in.

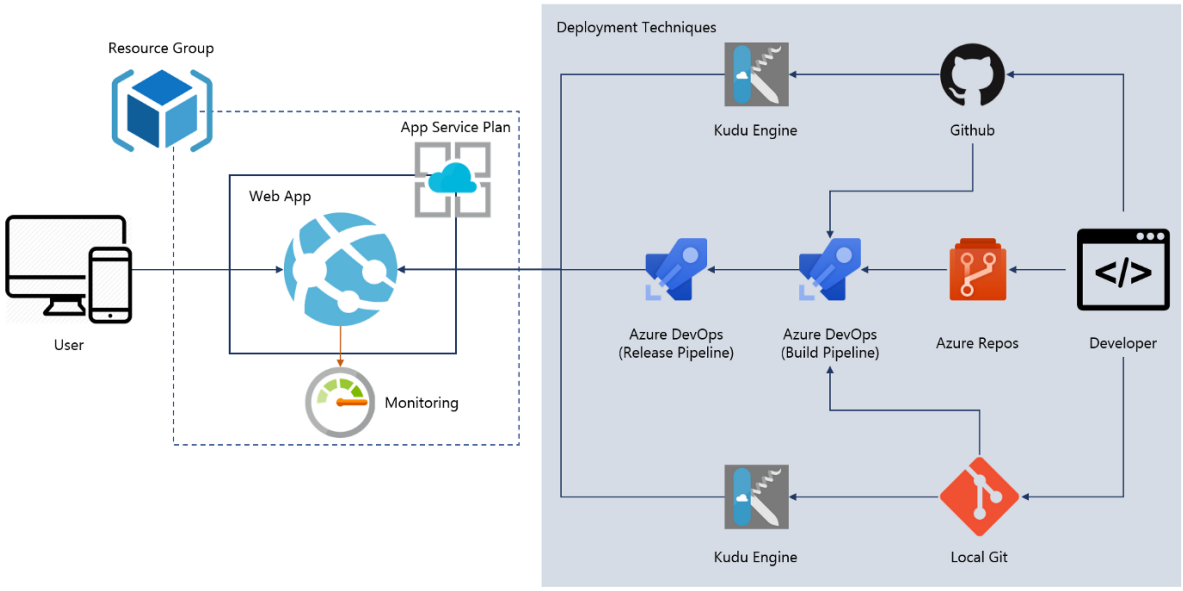
- udostępnianie uniwersalnych narzędzi dla różnorodnych biznesów, np. uwierzytelnianie użytkowników, *hosting* [23] stron i systemów, udostępnianie magazynów danych na przechowywanie plików typu BLOB (ang. *binary large objects*) [24],
- prostota i możliwości łatwej skalowalności usług w zależności od np. liczby użytkowników aktualnie korzystających z systemu,
- przeniesienie odpowiedzialności za utrzymanie cyberbezpieczeństwa i administracji systemami na jak najlepszym poziomie na zewnętrznych usługodawców,
- zwiększenie prędkości wprowadzania produktu, aktualizacji lub poprawek dla użytkowników końcowych,
- zwiększenie mobilności dla użytkowników korzystających z danych usług,
- możliwości zmniejszenia kosztów produkcji i utrzymania dzięki doborze odpowiednich parametrów przez użytkownika

W związku z opisanymi wyżej korzyściami płynącymi z dyskontowania rozwiązań chmurowych wiele użytkowników postanawia częściowo lub w całości oprzeć architekturę danych systemów właśnie o mniejsze systemy udostępniane przez dostawców chmurowych (ang. *cloud providers*). Taka w pełni oparta o rozwiązania chmurowe architektura, może przypominać tą z rysunku 2.6.

Na podstawie danych przedstawionych w artykule *"Top 10 cloud service providers"* [25], trzema największymi dostawcami rozwiązań chmurowych są odpowiednio: Amazon Web Services (AWS), Microsoft Azure oraz Google Cloud Platform (GCP). Główne różnice pomiędzy wyżej wymienionymi dostawcami opierają się przede wszystkim o udostępnione przez nich funkcjonalności jak i regionami na których można wykorzystać udostępnione użytkownikom zasoby.

W kwestii użytkowników opracowujących aplikację leży decyzja, którą z chmur wykorzystać. Istnieją również systemy mieszające różne "chmury", np. uwierzytelnianie użytkowników może być zawarte na platformie Microsoft Azure, bazy danych tego systemu utrzymywane są na Amazon Web Service, a serwer odpowiedzialny za wysyłanie maili utrzymywany jest na lokalnych serwerach firmy.

Oczywiście wyżej wymienione rozwiązanie hybrydowe komplikuje architekturę systemu. Wprowadza dodatkowe utrudnienia i zagrożenia, np. w przypadku niedostępności któregoś z elementu systemu. W związku z powyższym takie rozwiązania powinny być brane pod uwagę na etapie projektu architektury aplikacji i odpowiednio przeanalizowane, np. korzystając z analizy "SWOT" [26].



Rysunek 2.6. Przykładowa architektura aplikacji opartej o rozwiązania chmurowe wraz z zaznaczonymi usługami platformy Microsoft Azure. Źródło: [27]

Warto również zwrócić uwagę na minusy jakie niesie ze sobą korzystanie z architektury opartej o rozwiązania chmurowe są to m.in.

- brak pełnej kontroli nad systemem czy usługą, a także ograniczona elastyczność pod względem dostosowania do potrzeb konkretnego użytkownika,
- skomplikowany proces obliczania płatności za wykorzystywane zasoby,
- awarie serwerów i związane z tym opóźnienia i braki w dostępności do wybranych usług.

3. Opis narzędzi zastosowanych w pracy

Rozdział ten poświęcony został przedstawieniu narzędzi, które zostały wykorzystane podczas tworzenia prototypu aplikacji.

3.1. Firebase

Firebase jest to platforma typu *Backend as a Service* [28] udostępniająca usługi hostingowe dla wielu rodzajów aplikacji. Wydana w 2011 r. i od tego czasu sukcesywnie rozwijana oddaje w ręce klientów wiele serwisów, w tym m.in.

- Uwierzytelnianie użytkowników przy pomocy "Firebase Authnetication". Udostępnia ona zasoby serwerowe, łatwy do użycia zestaw narzędzi dla programistów, oraz gotowy graficzny interfejs użytkownika do zarządzania zarejestrowanymi osobami.
- Hosting nierelacyjnej bazy danych "Firebase Firestore". Platforma oddaje w ręce użytkowników zasoby na przechowywanie danych oraz odpowiednie akcesoria do manipulowania informacjami.
- Utrzymanie przestrzeni dyskowej "Cloud Storage". Narzędzie to odpowiada za manipulację danych typu BLOB.
- Udostępnianie i zarządzanie własnoręcznie napisanymi skryptami w wybranych językach programowania przy pomocy "Cloud Functions". Usługa ta udostępnia infrastrukturę, narzędzia i interfejs graficzny do tworzenia i zarządzania własnymi programami często będących kluczową częścią logiki biznesowej.
- Hosting programów użytkowników przy pomocy "Firebase Hosting". Umożliwia użytkownikom końcowym serwowanie wcześniej przygotowanych stron internetowych.
- Analizę danych dostępną za pomocą "Google Analytics for Firebase". Serwis ten umożliwia użytkownikom podłączenie swoich aplikacji pod specjalnie przygotowany serwer. Ma za zadanie zbierać i przetwarzać różnego rodzaju dane zebrane podczas korzystania przez użytkowników końcowych z danej aplikacji.

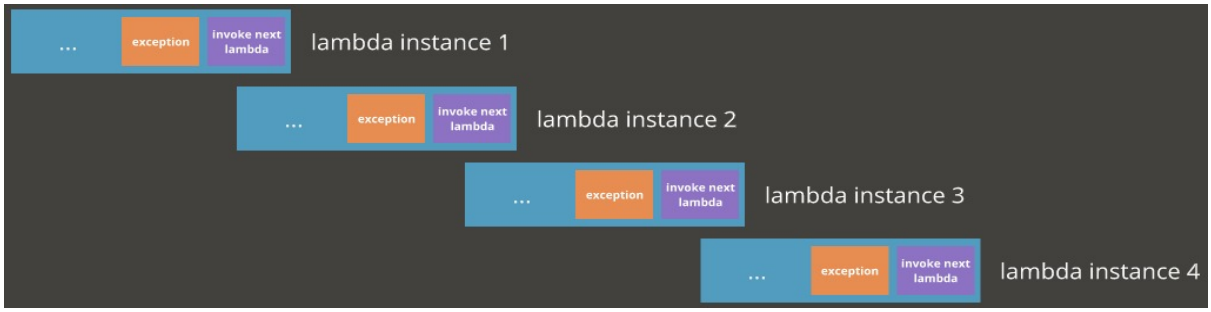
W tym miejscu warto wyjaśnić czym są funkcje przechowywane w chmurze oraz w jaki sposób platforma firebase różni się od innych dostępnych na rynku rozwiązań.

3.1.1. *Functions as a Service*

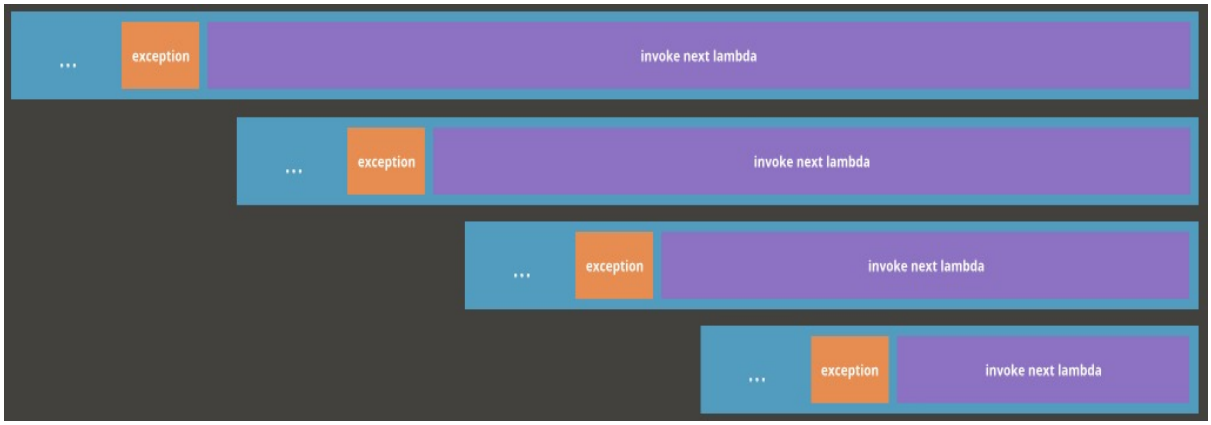
Termin *FaaS* (ang. *Functions as a Service*) jest ściśle związany z przetwarzaniem bezserwerowym. Jego głównym założeniem jest udostępnienie odpowiednich zasobów serwera programistom, tak by Ci mogli pisać kawałki programów w dowolnym języku programowania. Następnie napisany kod jest utrzymywany na danej platformie, a odpowiednia funkcja wywoływana jest przez aplikację kliencką.

Takie podejście pozwala mieszać ze sobą różnego rodzaju technologie, a także w założeniu pozwala nam na zmniejszenie kosztów utrzymania aplikacji. Opłata jaka jest naliczana użytkownikowi najczęściej odpowiada wartości jak długa dana funkcja była przetwarzana, a także ile fizycznie zasobów. np. pamięci podręcznej wykorzystała.

W związku z taką postawą znane są również przypadki tworzenia tzw. "nieskończonych pętli" [29], [30]. Uruchomienie funkcji bez jasnego warunku wyjścia powodować może gwałtowny wzrost wykorzystywanych zasobów, a co za tym idzie znaczne zwiększenie wydatków w krótkim czasie. Proces wywołania funkcji z prawidłowym warunkiem wyjścia, oraz fragmentu kodu z "nieskończoną pętlą" przedstawiony został odpowiednio na rysunku 3.1 oraz 3.2.

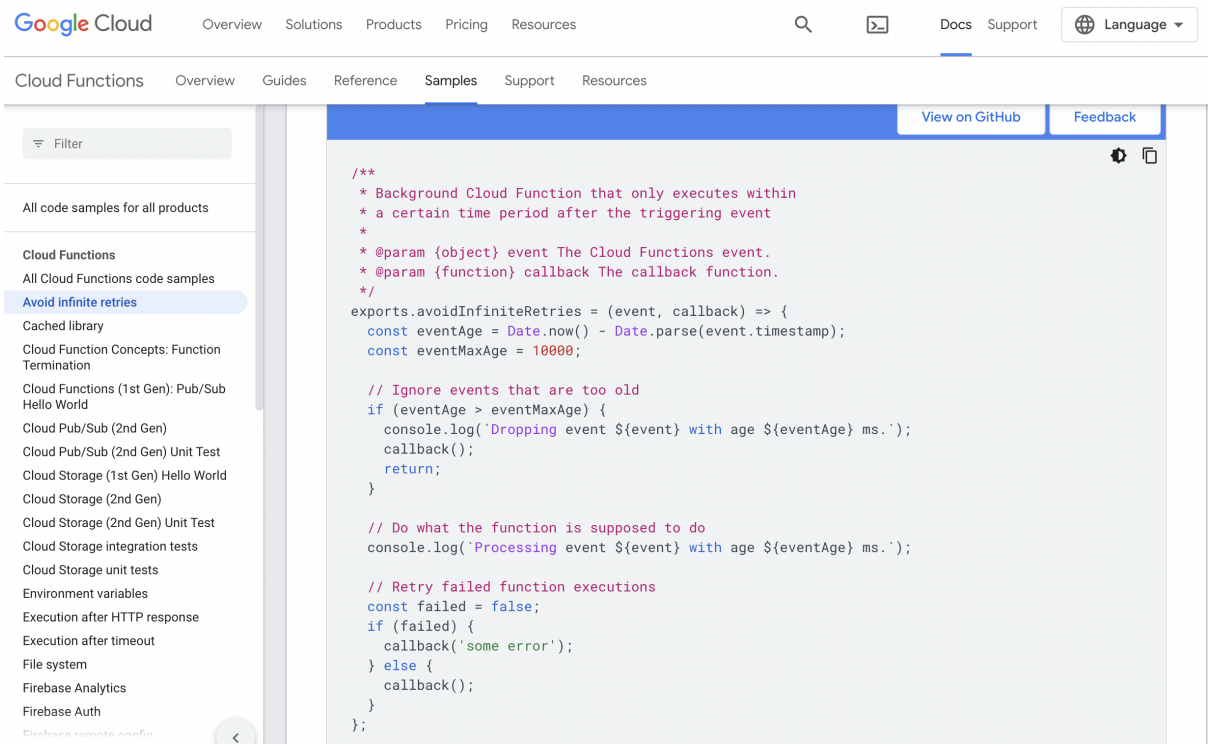


Rysunek 3.1. Prawidłowy schemat uruchomienia i zakończenia funkcji. Źródło: [31]



Rysunek 3.2. Schemat uruchomienia "nieskończonych" funkcji. Źródło: [31]

Warto zaznaczyć, że usługodawcy rozwiązań chmurowych zdają sobie sprawę z takich precedensów. W związku z tym informacje na ten temat można również znaleźć w dokumentacji danej usługi chmurowej. Przykład takiej informacji znajduje się na rysunku 3.3. Zdarza się, że pomimo naliczonych dużych opłat, firmy takie umarzają daną płatność, jednakże takie sytuacje rozpatrywane są indywidualnie po wcześniejszym kontakcie z odpowiednim działem obsługi technicznej.



Rysunek 3.3. Fragment dokumentacji "Cloud Functions" wraz z listingiem kodu odpornym na "nieskończoną pętlę". Źródło: [32].

3.1.2. Porównanie z alternatywnymi usługami chmurowymi

Pomimo wielu korzyści które oferuje Firebase nie jest to niezastąpione narzędzie na rynku. Na podstawie danych znajdujących się w artykule *"26 Cloud Computing Statistics"* [33], firma Google odpowiedzialna za wsparcie i rozwój platformy Firebase odpowiada za 7% rynku platform chmurowych.

Wpływ na taki stan rzeczy ma m.in. fakt, iż funkcjonalności udostępniane przez system nie są unikatowe. Dla przykładu najpopularniejszą platformą chmurową na chwilę obecną jest Amazon Web Service. Serwis dostarcza wiele rozwiązań dla szerokiego grona odbiorców, jednak to co istotne wśród wszystkich produktów, są również dostępne wszystkie zamienniki oferowane przez Firebase. Wspólne systemy dostępne na obu platformach zostały przedstawione na rysunku 3.4.

Rysunek 3.4. Przedstawienie części wspólnych systemów platformy Firebase i Amazon Web Service. Źródło: [34].

Dodatkową zaletą wyboru platformy AWS jest fakt, iż użytkownik ma dowolność wyboru wykorzystywanej bazy danych. Jest to o tyle istotne, że dzięki temu systemy utrzymywane na tej stronie mogą być dużo prostsze do przeniesienia na inne serwery lub chmurę, natomiast w przypadku skorzystania z baz danych od google'a, migracja danych może być skomplikowana.

Z drugiej jednak strony warto wziąć pod rozwagę, że Firebase udostępnia intuicyjny i prostszy w obsłudze panel do zarządzania tymi samymi systemami co Amazon Web Services. Dodatkową zaletą jest również darmowy plan dostępny na platformie od Google, który umożliwia hostowanie mało wymagających aplikacji.

3.2. React.js

React.js jest to biblioteka języka JavaScript. Dostosowana jest ona również do obsługi języka TypeScript. Została stworzona przez firmę Facebook (aktualnie Meta) w 2013 r. Biblioteka zyskała swoją sympatię wielu programistów na świecie, o czym świadczyć mogą statystyki zebrane przez *"The State of JS 2021"* [35]. Wśród ankietowanych 100% z nich zadeklarowało, że jest zaznajomiona z niniejszą biblioteką i aż 70% ponownie wybierze bibliotekę React.js przy tworzeniu następnego projektu.

Swoją popularność zawdzięcza m.in. niewielkimi umiejętnościami technicznymi jakie powinien mieć programista, szybkim procesem tworzenia aplikacji i wprowadzania na nich zmian dzięki zastosowaniu tzw. Obiektowego Modelu Dokumentu (ang. Document Object Model, DOM) [36]. Przełomowym okazało się również wprowadzenie języka JSX. Jest to nakładka na standardowy język JavaScript, jednakże struktura tego języka jest łudząco podobna do składni prezentowanej w HTML5.

Dzięki temu zabiegowi wydzielanie pomniejszych części kodu do osobnych komponentów [36], stało się dużo prostsze i bardziej intuicyjne, a to w połączeniu z wirtualnym obiektowym modelem dokumentu pozwoliło na znaczne przyspieszenie i ułatwienie pracy autorom stron internetowych.

3.3. Next.js

Next.js jest to otwartoźródłowy *framework* [37] do tworzenia nowoczesnych aplikacji internetowych. Zapewnia on wiele rozmaitych dodatków, które przyspieszają i ułatwiają pisanie stron internetowych. Główne korzyści, którymi zagarnął sobie popularność programistów zajmujących się pisaniem stron w bibliotece React.js to m.in.

- Możliwość generowania stron typu *Server Side Rendering* (SSR) [38] i statycznych stron internetowych. Możliwość tworzenia aplikacji po stronie serwerowej stała się niezwykle pożądana w dzisiejszych serwisach sieciowych. Jest to szczególnie istotne w przypadku konieczności dostarczenia jak najszybciej gotowej strony końcowemu użytkownikowi.

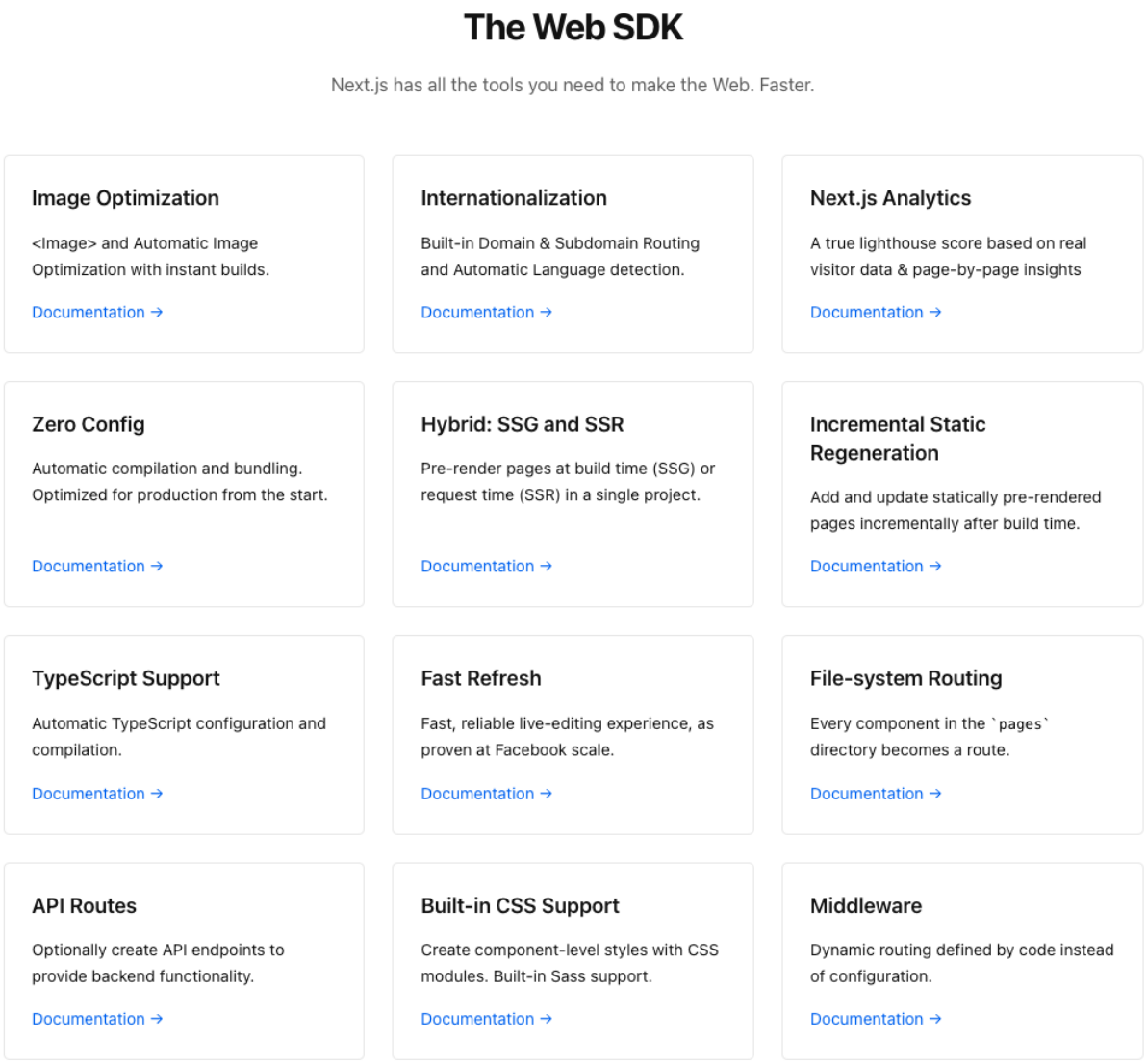
  Dodatkowo szybkość przesłania danych ma również wpływ na tzw. pozycjonowanie stron internetowych (ang. *Search Engine Optimalisation*) (SEO) [39]. Zgodnie z artykułem *How To Improve SEO Rankings in 2022* [40], jednym z kluczowych czynników mających wpływ na wyszukiwanie naszej strony przez przeglądarki internetowe ma właśnie

wpływ prędkości dostarczania strony. Dzięki takiemu zabiegowi zwiększa się prawdopodobieństwo, iż więcej osób zobaczy taką stronę.

- Optymalizacja obrazków i czcionek z możliwością załadowania poprzez mechanizm *lazy loading* [41]. Technika ta pozwala na ładowanie zawartości strony o większych rozmiarach dopiero w momencie wyświetlenia strony końcowemu użytkownikowi.
  Ma to również wpływ na zmniejszenie czasu dostarczenia zawartości strony.

- Uproszczenie konfiguracji, tworzenia aplikacji serwerowej i połączeń z bazą danych w jednym projekcie. Framework umożliwia również za sprawą specjalnie nazwanych plików na zastosowanie m.in. *proxy* [42] oraz *middleware* [43].

Dodatkowe udogodnienia oferowane przez Next.js zostały przedstawione na rysunku 3.5. Warto zaznaczyć, że osoby korzystające z frameworka nie są zmuszone do zastosowania wszystkich udostępnionych funkcjonalności. Jednakże część z nich, np. odpowiednie nazywanie plików w celu stworzenia odpowiedniej podstrony są obowiązkowe. W związku z powyższym w roli programisty pozostaje odpowiednie wykorzystanie funkcji i dostosowania struktury projektu pod dane narzędzie.



Rysunek 3.5. Narzędzia oferowane przez framework Next.js. Źródło: [44].

4. Podejście bezserwerowe do systemów ankietowych

Przedstawione w rozdziale drugim niniejszej pracy istniejące rozwiązania dotyczące programów do tworzenia i zarządzania ankietami zajmują znaczną część rynku. Lukę między w pełni zarządzanymi aplikacjami a utrzymywanymi samodzielnie mogłyby zapełnić otwartoźródłowe aplikacje w podejściu bezserwerowym.

Koncepcja ta opiera się m.in. o udostępnienie kodu źródłowego takiego programu wszystkim użytkownikom. Zawartość takiego projektu powinna brać szczególną uwagę na możliwość indywidualnej rozbudowy. W kwestii osoby chcącej skorzystać z tego systemu leżałoby jedynie przygotowanie odpowiedniej konfiguracji projektu, np. bazując na Firebase. Następnie przy pomocy prostego instalatora czy użyciu odpowiedniej dokumentacji, użytkownik umieszczałby kod na własnych zasobach chmury. Zarządzanie ankietami przez użytkownika końcowego odbywałoby się przez specjalny portal znajdujący się w kodzie źródłowym.

4.1. Modyfikacje kodu

W toku tego rozumowania bardzo istotne jest zachowanie przede wszystkim prostoty rozbudowy aplikacji, a co za tym idzie również kompatybilność języków programowania pomiędzy różnymi częściami systemu. Ważne jest, by ewentualne modyfikacje były możliwe do wprowadzenia przez jak najmniejszą liczbę osób w jak najkrótszym czasie.

W celu zachowania prostoty autor postawił na wykorzystanie języka TypeScript z biblioteką React.js i frameworkiem Next.js. Wpływ na to miał m.in. fakt licznych bibliotek udostępnionych przez użytkowników poprzez repozytorium "npm" (ang. *Node Package Manager*) [45]. Rozszerzenia te umożliwiają modyfikację projektu zarówno pod kątem wizualnym, przetwarzaniem logiki programu jak i dostosowania części serwerowej.

Dzięki nim możliwe jest odpowiednie dostosowanie warstwy prezentacyjnej, np. poprzez zamianę biblioteki komponentów dla biblioteki React.js lub zastosowanie starszych bibliotek np. jquery [46]. Dodatkowo dzięki zastosowaniu Next.js, osoba modyfikująca projekt może również skorzystać z bibliotek przeznaczonych do manipulacji danymi po stronie serwera zamiast tworzenia nowej osobnej aplikacji.

Dodatkową zaletą związaną z wyborem języka TypeScript jest m.in. sprawdzanie gotowych już typów umieszczonych w projekcie, tak aby nowe fragmenty kodu były kompaty-

bilne z już istniejącymi. Język ten jest również relatywnie prosty do nauki przede wszystkim dla początkujących użytkowników. Według informacji zawartych w artykule *"The Most In-Demand Programming Languages for 2022"* [47] język JavaScript, czyli podzbiór języka TypeScript, jest najbardziej popularnym językiem programowania. Dzięki temu w sieci dostępnych jest wiele kursów, artykułów czy dokumentacji, która ułatwia proces dostosowywania aplikacji pod własne wymagania.

4.2. Dostosowanie do infrastruktury

Ze względu na różnice oferowane przez usługodawców chmurowych istotną sprawą jest również udostępnienie użytkownikom możliwości podmiany kodu z podejścia bazującego obecnie na *Cloud Functions* na korzystanie z własnej aplikacji serwerowej lub wykorzystanie innych funkcji, np. *Azure Functions* [48].

W stworzonym projekcie całość infrastruktury została oparta o platformę Firebase i oferowane przez nią funkcjonalności. Dodatkowo właściciel takiego projektu otrzymuje możliwość odpowiedniego skalowania zasobów w miarę zmieniania się ruchu na stronie.

W przypadku chęci wykorzystania innej platformy użytkownik zobligowany byłby do indywidualnej podmiany poszczególnych części systemu. Wiązałoby się to również z odpowiednim dostosowaniem kodu projektu do obsługi innej platformy chmurowej.

Problemem w stworzeniu uniwersalnego dostosowania aplikacji do infrastruktury jest mnogość obecnych rozwiązań. Z tego względu odpowiedzialność za dostosowanie aplikacji do innych platform została przeniesiona w całości na użytkownika końcowego projektu.

Ewentualną próbą rozwiązania tego problemu mogłoby być przygotowanie odpowiednich scenariuszy dla wybranych platform chmurowych, np. AWS, czy Azure. Rozwiązanie to mogłoby zadziałać dla części przypadków. Jednakże zgodnie z opisywanymi w rozdziale drugim rozwiązaniami hybrydowymi, liczba kombinacji możliwych konfiguracji jest zbyt duża. W związku z tym każdorazowe tworzenie osobnego "uniwersalnego" scenariusza nie miałoby większego sensu.

4.3. Adaptacja bazy danych

Stworzony prototyp aplikacji przechowuje dane w nierelacyjnej bazie danych Firestore. Główną zaletą wykorzystania takiej bazy jest m.in. prosta struktura przechowywanych danych, a także nieskomplikowane wprowadzanie zmian w istniejących już modelach danych.

Pierwszą z zalet płynącą z tego rozwiązania jest m.in. prostota wprowadzania zmian w istniejących schematach. Każdy z użytkowników korzystający z takiego rozwiązania może indywidualnie dostosować model pod własne możliwości bez konieczności tworzenia specjalnych migracji. Dodatkowym ułatwieniem jest w tym wypadku również podobieństwo przechowywanych danych do obiektów tworzonych w języku JavaScript.

Dodatkową zaletą korzystania z nierelacyjnej bazy danych autorstwa firmy Google, jest również brak konieczności definiowania struktury przechowywanych danych. W tym przypadku po części serwerowej należy przekazać odpowiedni obiekt do zapisania w bazie danych, a Firestore automatycznie dobierze odpowiednie typy danych.

W przypadku chęci podmiany systemu do przechowywania danych przez użytkownika również istnieje kilka zalet skorzystania z Firestore. W momencie zmiany na inną nierelacyjną bazę, np. MongoDB [49], konieczne jest zdefiniowanie odpowiednich struktur na przechowanie danych i podmianę odpowiednich zapytań w kodzie na takie, które umożliwiają komunikację z wybraną bazą danych. W tym przypadku struktura bazy danych nie zmienia się.

W razie chęci wykorzystania relacyjnej bazy danych, sytuacja będzie w dużej mierze zależała od celu jaki ma zostać osiągnięty. Mając wystarczającą liczbę danych można by odpowiednio je sformatować do widoku tabel w relacyjnych bazach danych. Wymagałoby to jednak przeprojektowaniu schematów przechowywanych danych w obecnej bazie i dostosowania je pod poszczególne wartości udostępnione przez bazę danych.

Najprostszy przypadek migracji, np. do bazy PostrgreSQL mógłby wiązać się z zapisaniem pojedynczego obiektu do tabeli jako typ JSON [50], [51] lub odpowiednio sformatowany tekst. W tym przypadku również należałoby przeanalizować wszystkie wady i zalety i dostosować rozwiązanie pod dany problem. Odpowiedzialność w tej kwestii również została przerzucona na użytkownika końcowego.

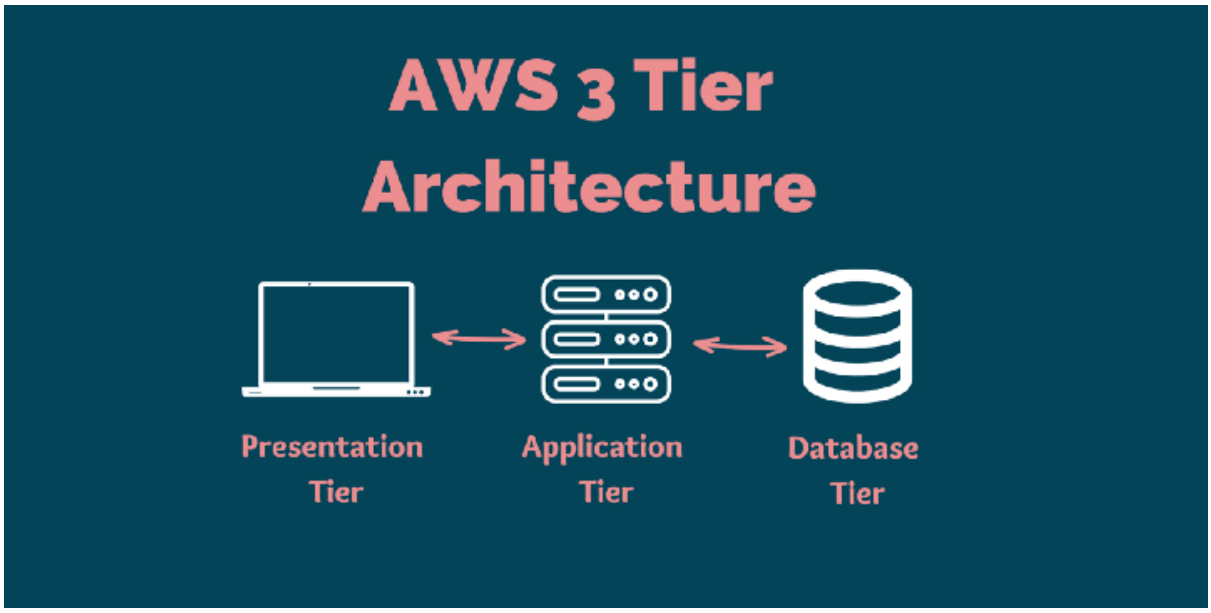
5. Prototyp i możliwości bezserwerowej aplikacji ankietowej

Stworzony prototyp aplikacji został przeznaczony dla dowolnych grup użytkowników końcowych. Otrzymują oni możliwość wgrania gotowego systemu na samodzielnie stworzony projekt Firebase. Platforma udostępnia tworzenie, udostępnianie oraz kolekcjonowanie wyników stworzonych ankiet. System zezwala na tworzenie formularzy do wypełnienia przez anonimowych użytkowników jak i zarejestrowanych. Dodatkowo wspiera również możliwość dostosowania tła udostępnianej ankiety i opcję jej załączenia poprzez specjalny tag HTML na dowolną inną stronę internetową.

Istotną kwestią jest również zbieranie odpowiedzi z uzupełnionych ankiet. Odpowiedzi udzielone przez użytkownika są zbierane w sposób jawny. Analogicznie informacje takie jak przybliżona lokalizacja z której został uzupełniony formularz oraz czas jej uzupełnienia nie są zbierane od użytkowników w pełni świadomie. Aplikacja następnie pozwala na eksport tych danych do pliku z rozszerzeniem .csv. Także na podejrzenie uzupełnionej ankiety przez każdego użytkownika.

5.1. Architektura i struktura projektu

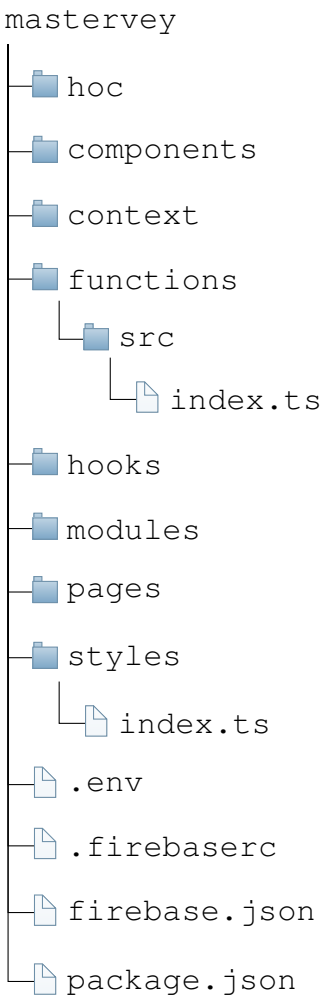
Projekt wykorzystuje architekturę trójwartstwową, której schemat został zaprezentowany na rysunku 5.1.



Rysunek 5.1. Schemat systemu opartego o architekturę trójwarstwową. Źródło: [52].

Aplikacja w takiej formie jest relatywnie prosta w utrzymaniu, a także umożliwia jej rozbudowę np. o dodatkowe serwisy. Należy pamiętać również, że schemat ten jest jedynie abstrakcyjnym pojęciem i ma za zadanie wyróżnić poszczególne części systemu. Może, ale nie musi ono oznaczać, że dana aplikacja składa się z co najmniej dwóch aplikacji i pojedynczego serwera bazy danych.

Na tak opartej architekturze aplikacji została stworzona struktura projektu przedstawiona na rysunku 5.2.



Rysunek 5.2 Struktura wybranych plików i katalogów aplikacji. Źródło: Opracowanie własne.

Przedstawiony na powyższym rysunku układ ma swoje odzwierciedlenie w logice aplikacji. Każdy z wyróżnionych elementów odpowiada za następujące kwestie.

- "hoc" - w tym katalogu przechowywane są wszystkie komponenty mające za zadanie przekazać pewne dane lub określić stan komponentów niższego rzędu. Koncepcja ta została szczegółowo opisana w dokumentacji biblioteki React.js w artykule *"Higher-Order Components"* [53].

- "components" - folder zawiera komponenty przeznaczone do ponownego użycia w dowolnych częściach programu. Wyodrębnianie części wspólnych dla wielu partii programu związane jest z tzw. regułą *DRY* (ang. Don't Repeat Yourself) [54]. Dzięki zastosowaniu niniejszej podpowiedzi aplikacja składa się z kilku prostych do modyfikacji fragmentów co pozwala na szybką i nieskomplikowaną rozbudowę czy edycje kluczowych partii systemu.

- "context" - w tym miejscu przetrzymywane są pliki korzystające z mechanizmu biblioteki React.js zwanej "context". Ten fragment szczegółowo opisuje dokumentacja [55]. Jednakże w skrócie proces służyć ma przekazywaniu danych w głąb drzewa DOM do odpowiednich komponentów. Dzięki temu w przypadku zmiany stanu kontekstu ponownie renderowane zostaną tylko wybrane elementy programu.

- "functions" - folder ten został wygenerowany przez *CLI* [56] platformy Firebase. W katalogu tym stworzony został osobny program na bazie środowiska Node.js. Centralnym punktem katalogu jest plik "index.ts" znajdujący się w podkatalogu "src". Właśnie z tego miejsca muszą być wyeksportowane wszystkie funkcje przeznaczone do uruchomienia na platformie Firebase.

- "hooks" - w katalogu zostały umieszczone pliki, które wykorzystują następny wbudowany mechanizm z biblioteki React.js [57]. Upraszczając "hook" jest to funkcja, której głównym zadaniem jest manipulacja stanem danego komponentu. Mechanizmy takie mogą tworzyć programiści, ale również część z nich jest już przygotowana przez twórców biblioteki.

- "modules" - w tym miejscu przechowywane są katalogi, które odpowiadają konkretnym elementom na stronie. Przykładowo znajdują się tu pliki z komponentami, które znalazły zastosowanie w jednym fragmencie kodu projektu.

- "pages" - w folderze tym znajdują się pliki odpowiedzialne za generowanie poszczególnych stron aplikacji. Katalog został ustawiony domyślnie przez framework Next.js. Uzasadnieniem takiego podejścia jest fakt, iż podczas budowy projektu narzędzie wstępnie tworzy stronę przy pomocy tagów HTML [58]. Głównym celem tego rozwiązania jest chęć przyspieszenia czasu dostarczenia strony do użytkownika końcowego.

- "styles" - przechowuje jedynie plik "index.ts". W tym miejscu skonfigurowany jest wygląd większości komponentów wykorzystywanych w projekcie. Dzięki takiemu podejściu zmiany szat graficznych są dużo prostsze do wdrożenia. Dodatkowo programista w razie konieczności stworzenia własnego komponentu ma pewność, że korzystając ze zmiennych z niniejszego pliku, wygląd nowego elementu będzie kompatybilny z resztą projektu.

- ".env" - plik ten przechowuje zmienne środowiskowe [59] wykorzystywane w reszcie programu.

- ".firebaserc" - w tym miejscu przechowywane są informacje dotyczące projektu Firebase. Dane w nim zawarte są zapisane w postaci JSON.

- "firebase.json" - w dokumencie tym w formacie JSON zawarte są informacje dotyczące hostingu projektu na platformie Firebase. Dodatkowo znajdują się tu również części odpowiedzialne za lokalne dostosowanie emulatorów[60], np. uwierzytelniania użytkowników czy bazy danych.

- "package.json" - w pliku tym zawarte są informacje potrzebne do uruchomienia projektu przez framework Next.js. Znaleźć tu można informacje dotyczące m.in. autora projektu, modułów potrzebnych do prawidłowego działania projektu, a także jakie komendy może uruchomić użytkownik z poziomu CLI. Dostępnych jest wiele więcej opcji do konfiguracji, których lista znajduje się w dokumentacji npm [61].

5.2. Część prezentacyjna

W niniejszym podrozdziale zostały opisane wybrane fragmenty kodu odpowiedzialne za warstwę kliencką. Pierwszym ciekawym przykładem kodu jest punkt startowy programu. Plik "_app.ts", którego zawartość prezentuje listing 5.2.1 oraz 5.2.2. Dokument znajduje się w folderze *"pages"*.

```
function MyApp({ Component, pageProps }: AppPropsWithLayout) {
  const getLayout = Component.getLayout ?? ((page) => page);
  return (
    <AuthProvider>
      <ThemeProvider theme={theme}>
        <QueryClientProvider client={queryClient}>
          <Hydrate state={pageProps.dehydratedState}>
            <Head>
              <title>Mastersurvey - your survey manager.</title>
              <meta name="description"
                content="Create, manage and share your own survey!"
              />
              <link rel="icon" href="/favicon.ico" />
            </Head>
            <CssBaseline />
            {Component.requireAuth ? (
              <AuthGuard>{getLayout(<Component {...pageProps} />)}
              </AuthGuard>
            ) : (
              getLayout(<Component {...pageProps} />)
            )}
          </Hydrate>
        </QueryClientProvider>
      </ThemeProvider>
    </AuthProvider>
  );
}
```

Listing 5.1. Podstawowy komponent warstwy prezentacyjnej

Analizując listing 5.1 warto zwrócić uwagę na część z mechanik zastosowanych w pierwszych dwóch liniach. Pierwszą z nich jest tzw. destrukturyzcja [62]. Dzięki temu zabiegowi można bezpośrednio odwołać się do się do pól obiektu JavaScript zamiast każdora-

zowo odwoływać się poprzez wpisywanie nazwy tego obiektu. W linii 2 natomiast wykorzystany jest specjalny operator `??` [63]. Z jego pomocą w TypeScripcie można zweryfikować czy wartość z lewej strony jest typu *nullish* [63]. W sytuacji kiedy jest to wartość *nullish* zostaje wybrana prawa strona równania. W tym wypadku wykorzystywane jest to w celu odpowiedniego renderowania wyglądu danej strony.

Od linii 4 do 7 odpowiednio wywoływane są komponenty które powinny być ogólne dostępne w aplikacji. Pierwszym z nich jest `AuthProvider`, czyli odpowiedni komponent który odpowiedzialny jest za uwierzytelnianie. W jego implementacji znajdują się adekwatne metody które modyfikują pola tego obiektu, a te z kolei wykorzystywane są przez pozostałe komponenty niższego rzędu.

Następnie wywoływany jest `ThemeProvider`, czyli komponent który steruje stylowaniem aplikacji. Przekazany jest mu jeden argument `theme`, który to jest importowany z pliku "index.ts" znajdującym się w folderze *"styles"*. Poniższe dwa komponenty wykorzystane są z otwartoźródłowej biblioteki "react-query" [64]. Narzędzie to ułatwia wysyłanie zapytań HTTP i jednocześnie dostarcza użyteczne zmienne odpowiednim komponentom. Dostosowane jest do korzystania po stronie przeglądarek użytkownika (komponent `QueryClientProvider`) jak i zapytań tworzonych po stronie serwera (komponent `Hydrate`).

W liniach 8-14 przy pomocy specjalnego komponentu `Head` pochodzącym z frameworka Next.js konfigurowany jest nagłówek strony HTML. W tym przypadku ustawiany jest odpowiedni tytuł strony (komponent `title`), opis (komponent `meta`), a także ikona widoczna w oknie przeglądarki. Zabiegi te mają na celu lepsze pozycjonowanie strony przez wyszukiwarki internetowe.

Linia 15 odpowiada jedynie za ustawienie domyślnego stylu dla każdej ze stron. Ciekawszą częścią są fragmenty kodu znajdujące się w liniach 16-21. Na tym poziomie sprawdzane jest czy dana strona wymaga uwierzytelnienia od użytkownika czy jest dostępna publicznie. Jeśli strona dostępna jest dla zalogowanych osób, wtedy wywoływany jest odpowiedni *hoc* `AuthGuard`. Komponent ten sprawdza, czy w pamięci podręcznej przeglądarki są ustawione odpowiednie pola. Jeśli tak, wartości te są dekodowane i odpowiednio sprawdzane, aby upewnić się że konsument ma prawo do zobaczenia danej zawartości.

Innym intrygującym elementem programu jest zawarty w tym samym pliku proces konfiguracji połączenia z częścią bezserwerową aplikacji, której kod przedstawiony został

na listingu 5.2.

```
1 axios.defaults.baseURL = process.env.NEXT_PUBLIC_API;
2
3 axios.interceptors.request.use((config: AxiosRequestConfig) => {
4   const token = nookies.get()?.token;
5   config.headers.Authorization = token ? `Bearer ${token}` : "";
6   return config;
7 });
8
9 const queryClient = new QueryClient();
```

Listing 5.2. Konfiguracja obiektów do komunikacji z funkcjami firebase

Począwszy od pierwszej linii kodu, ustawiany jest URL nakierowany na funkcje Firebase. Przy pomocy specjalnego obiektu `axios` konfigurowane jest odpowiednie pole, tak aby domyślnie wszystkie zapytania HTTP wykonywane były na właściwy adres.

Następnie w liniach 3-7 ustawiany jest nagłówek `Authorization` wszystkich metod REST [65]. Dane potrzebne do przesłania, wyciągane są z pamięci podręcznej przeglądarki przy pomocy odpowiedniego obiektu `nookies`. Informacja ta jest niezwykle istotna ponieważ na jej podstawie część bezserwerowa aplikacji może stwierdzić czy użytkownik jest zalogowany czy nie. W zależności od implementacji danego punktu końcowego, funkcja zwraca odpowiednią odpowiedź lub informację dotyczącą braku wymaganych uprawnień.

W ostatniej linii listingu 5.2. tworzony jest odpowiedni obiekt wymagany przez bibliotekę "react-query". Utworzona zmienna przekazywana jest następnie w linii nr 6 listingu 5.1.

Ostatnim ciekawym przykładem kodu jest ten przedstawiony na listingu 5.3.

```
const validationSchema = yup.object().shape({
  isPublic: yup.boolean(),
});
const Generate = () => {
  const { handleSubmit, control } = useForm({
    resolver: yupResolver(validationSchema),
  });
  const { mutateAsync, isLoading } = useMutation(
    "saveSurvey",
    async (input: { create: Question[]; isPublic: string }) => {[...]}
  );

  const { fields, remove, update } = useFieldArray({
    control,
    name: "create",
  });
  return (
        <form id="generate-survey" onSubmit={handleSubmit(mutateAsync)}>
            {fields?.map((field: Question, index: number) => {
              return (
                  <CreateQuestionCardNew
                    remove={remove}
                    control={control}
                    update={update}
                    index={index}
                    question={field}
                  />
              );
            })}
        </form>
```

Listing 5.3. Formularz tworzenia ankiety

Fragment odpowiedzialny za tworzenie ankiet warto przeanalizować od pierwszych linii kodu. W liniach 1-3 generowany jest specjalny obiekt `yup`. Jego głównym zadaniem jest walidacja danych po stronie klienta i zwrócenie odpowiednich informacji w przypadku znalezienia błędu lub poprawnie rozpoznanego obiektu.

Obiekt weryfikujący przekazywany jest następnie w linii 6 do odpowiedniej metody `yupResolver`, która to w całości przekazywana jest do hook'a `useForm`. Istotna jest tu informacja, iż metoda `yupResolved` formatuje odpowiednio otrzymane informacje z obiektu `yup` i dostosowuje je do prawidłowego formatu wykorzystywanego przez hook `useForm` pochodzącego z biblioteki "react-hook-form".

Następnie w linii 5 destrukturyzowane są odpowiednie pola ze zwróconego obiektu. Wykorzystywane są one m.in. w następnej funkcji `useFieldArray`, a także bezpośrednio przez komponent `form` w linii 18. Zaczynając od linii 13, hook `useFieldArray` służy do generowanie dynamicznego pola w formularzu. Udostępnia on funkcje i pola dzięki którym w prosty sposób można manipulować danymi zawartymi w tablicy kwestionariusza. Po odpowiednim mapowaniu pól w linii 19, wszystkie wyłuskane dane z funkcji przekazywane są do komponentu `CreateQuestionNewCard`. Następnie na podstawie przekazanych informacji, obiekt JSX wyświetla odpowiednio przygotowane pole formularza.

W momencie wypełnienia przez użytkownika wybranych pól, wykorzystana jest druga z metod zwracanych przez hook `useForm`. Metoda `handleSubmit` ma za zadanie odpowiednio zaktualizować informacje dla reszty obiektu zwracanego przez `useForm`. Jako jedyny argument funkcja przyjmuje metodę `mutateAsync`, która jest metodą obiektu zwracanego przez hook `useMutation` z biblioteki "react-query".

Kiedy użytkownik próbuje przesłać dane na część bezserwerową aplikacji, uruchamiana jest funkcja znajdująca się w linii 10 listingu 5.3. W ciele tej funkcji wywoływane jest zapytanie HTTP przy pomocy obiektu `axios` a następnie zwracane są informacje otrzymany przez niniejszy obiekt. Ze względu na swoją prostotę ta część została pominięta na listingu.

Po otrzymaniu stosownej odpowiedzi z funkcji Firebase odpowiednio aktualizowane są pola i metody zwracane przez hooki `useMutation` i `useForm` (za pośrednictwem metody `handleSubmit`). Dodatkowo w momencie otrzymania informacji zwrotnej zostaje wygenerowana stosowna informacja dla użytkownika o sukcesie lub niepomyślnym przetworzeniu zapytania.

5.3. Warstwa bezserwerowa

Sekcja ta została poświęcona przedstawieniu wybranych fragmentów części bezserwerowej aplikacji. Zgodnie z rysunkiem 5.2 informacje zawarte w dalszych fragmentach podrozdziału są skupione wokół kodu znajdującego się w folderze *"functions"*.

Pierwszym przykładowym kodem, któremu warto się przyjrzeć jest ten umieszczony na listingu 5.4.

```
1  import * as admin from "firebase-admin";
2
3  [...]
4
5  admin.initializeApp();
6
7  admin.firestore().settings({ ignoreUndefinedProperties: true });
8
9  export {
10   signup,
11   createSurvey,
12   [...]
13 }
```

Listing 5.4. Plik początkowy funkcji firebase "index.ts"

Plik ten rozpoczyna się od importowania wszystkich funkcji i obiektów zawartych w bibliotece `firebase-admin` udostępnionej przez autorów platformy. Następnie w linii 3 importowane są wszystkie własnoręcznie napisane funkcje, a te eksportowane są dalej w liniach 9-13. Zabieg ten jest konieczny, ponieważ domyślnie "Firebase Functions" importuje jedynie metody z pliku "index.ts". Jednakże w celu zwiększenia czytelności kodu, został on odpowiednio sformatowany i podzielony na pliki.

Następnie w linii 5 tworzona jest nowa instancja aplikacji Firebase. Jej głównym zadaniem jest połączenie się z istniejącym już projektem Firebase przy pomocy odpowiednich plików konfiguracyjnych. Z kolei w linii 7 ustawiane są odpowiednie ustawienia dla bazy danych "Firestore". Po uruchomieniu wyżej opisanych metod, zmiany te są propagowane po odpowiednich polach i metodach obiektu `admin` i nadaje się on do dalszej pracy z poszczególnymi funkcjami.

Przykładową funkcją która korzysta z wyżej opisanej konfiguracji jest metoda `signup`

przedstawiona na listingu 5.5.

```
1  import * as functions from "firebase-functions";
2  import * as admin from "firebase-admin";
3
4  export const signup = functions.https.onRequest((request, response) => {
5      const { email, password } = request.body;
6
7      const { uid } = await admin.auth().createUser({
8        email,
9        password,
10     });
11
12     response.send(uid);
13 });
```

Listing 5.5. Funkcja rejestrująca użytkowników

W pierwszych dwóch liniach importowane są zależności wymagane w dalszej części listingu. Następnie w 4 linii wykorzystywany jest obiekt `functions`, którego implementacja metody `onRequest` będzie nasłuchiwała na zapytania z części klienckiej. Warto zaznaczyć, że w przypadku funkcji Firebase nie jest konieczne bezpośrednie przypisanie jakiej metody HTTP należy nasłuchiwać.

Następnie w linii 5 destrukturyzowane są odpowiednie dane z obiektu `request.body`. Informacje te przekazywane są do metody `createUser` pochodzącej z obiektu `admin`. Paczka `firebase-admin` odpowiada w tym przypadku za komunikację z serwisami platformy Firebase. Następnie w celu odwołania się do konkretnej usługi należy wykorzystać odpowiednio udostępnioną funkcję. W niniejszym przypadku jest to metoda `auth()`. Dzięki jej zastosowaniu, nowo utworzony użytkownik nie jest zapisywany bezpośrednio w bazie danych, a przechowywany jest w specjalnie przygotowanym do tego miejscu.

Jeśli użytkownik został pomyślnie dodany do serwisu uwierzytelniającego, zwracany zostaje obiekt z polem `uid`. Następnie obiekt ten zwracany jest do warstwy klienckiej w linii 12 przy pomocy metody `send`.

Ostatnim wartym omówienia przykładem kodu jest fragment znajdujący się na listingu 5.6.

```
1  export const getSurveys = functions.https.onRequest((req, res) => {
2      const tokenId = req.get("Authorization")?.split("Bearer ")[1];
3
4      if (!tokenId || typeof tokenId === "undefined") {
5        return res.status(401).send("Unauthorized");
6      }
7
8      const { uid } = await admin.auth().verifyIdToken(tokenId as string);
9
10      if (!uid) return res.status(401).send("Unauthorized");
11
12     const surveysRef = admin.firestore().collection("surveys");
13     const snapshot = await surveysRef.where("createdBy", "==", uid).get();
14
15     if (snapshot.empty) return res.status(200).send([]);
16
17     const data = snapshot.docs.map((doc) => doc.data());
18
19     return res.status(200).send(data);
20 });
```

Listing 5.6. Funkcja pobierająca ankiety

Analizę listingu 5.6. warto rozpocząć od 2 linii. W tym miejscu wyłuskiwany jest token, który to powinien znaleźć się w wysłanym zapytaniu (jest on ustawiany w listingu 5.2). Następnie w linii 4 następuje sprawdzenie, czy została wysłana odpowiednia informacja. Jeśli nie zwracany jest obiekt `res` z odpowiednim statusem i wiadomością dla warstwy klienckiej.

W linii 8 korzystając z metody `auth()`, sprawdzamy ważność przekazanego tokena. Weryfikacja przeprowadzana jest dzięki specjalnie udostępnionej funkcji `verifyIdToken`. Następnie w przypadku pomyślnej weryfikacji, zwracany jest odpowiedni obiekt z polem `uid`. W przypadku jeśli token stracił swoją ważność nie zostanie zwrócony ten sam element i przypadek ten zostanie wyłapany w linii 10.

W linii 12 i 13 następuje odwołanie się do bazy danych Firestore. W pierwszej z nich, wywoływana jest odpowiednia metoda `firestore` (odpowiedzialna za komunikację z bazą),

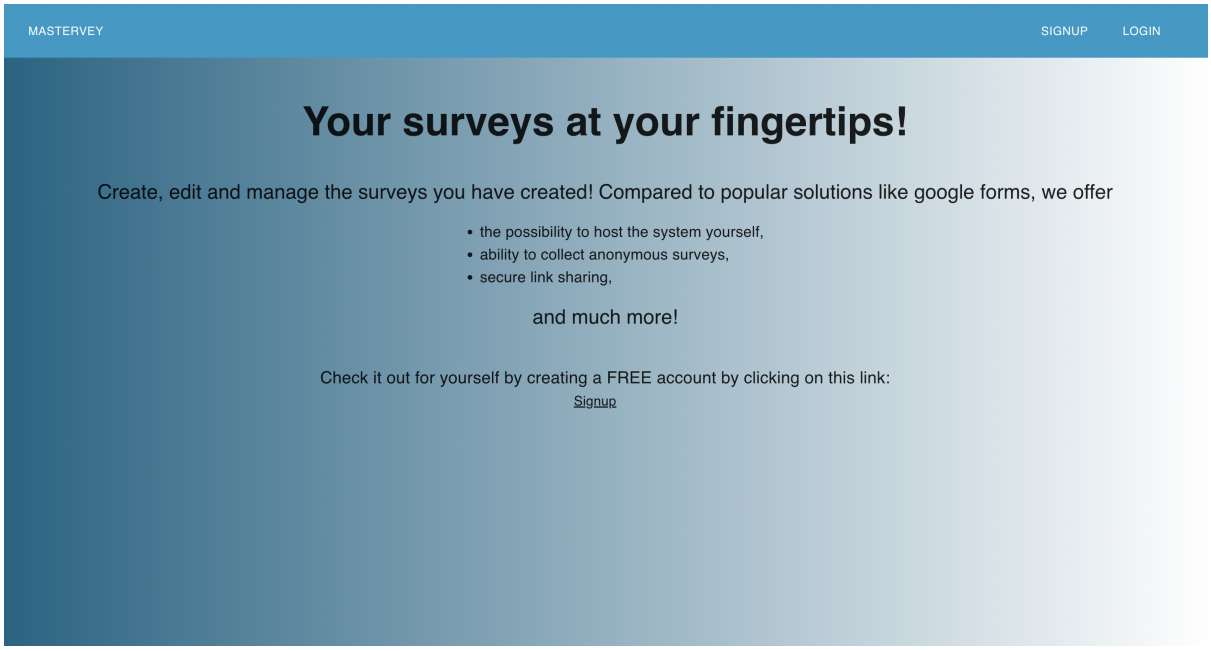
a następnie wskazana jest kolekcja do której należy się odwołać. Ta informacja przypisywana jest do zmiennej `surveysRef` z której to w linii 13 można bezpośrednio wyodrębnić dane. W niniejszym przypadku wykorzystana została metoda `where`, która jako kolejne argumenty przyjmuje nazwę pola przechowywanych obiektów, odpowiedni operator [66] i jako ostatni wartość przekazaną do funkcji. Na koniec linii 13 wywoływana jest metoda `get` zaznaczającą, że Firestore ma pobrać wszystkie dane spełniający warunek znajdujący się w metodzie `where`. Zwrócone w ten sposób dane przypisywane zostają do zmiennej `snapshot`.

Następnie w linii 15 następuje sprawdzenie czy zwrócony z bazy danych obiekt nie jest pusty. Jeśli jest, zostaje zwrócona odpowiednia wartość do warstwy prezentacyjnej.
Z kolei jeśli `snapshot` nie jest pusty w linii 17 następuje jego odpowiednie mapowanie, tak by zwrócić gotowe wyniki zapytania. Tak przygotowane informacje przekazywane są do odesłania do użytkownika w linii 19.

Warto zwrócić uwagę na fakt, iż w porównaniu do listingu 5.2, w obecnym przypadku z całej funkcji zwracany jest obiekt `res`. Zwrócenie drugiego parametru funkcji w przypadku metod Firebase nie wpływa na działanie programu. Dzieje się to dlatego, że domyślnie wszystkie funkcje zwracają obiekt typu *response*.

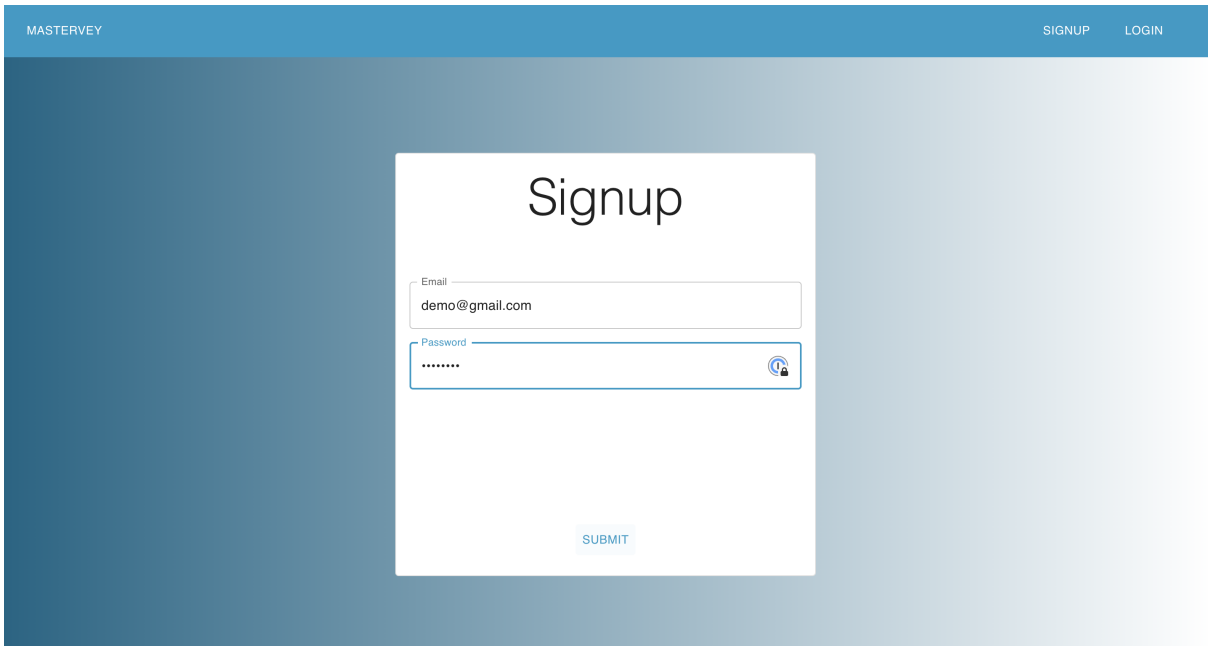
5.4. Interfejs użytkownika

Sekcja ta została poświęcona przedstawieniu interfejsu użytkownika. W niniejszym fragmencie zostały ukazane wybrane funkcjonalności oferowane przez prototyp bezserwerowego systemu zarządzania ankietami. Po wejściu na stronę główną prototypu aplikacji ukazuje się widok przedstawiony na rysunku 5.3.



Rysunek 5.3. Strona główna prototypu aplikacji. Źródło: Opracowanie własne.

Na stronie głównej użytkownik może dowiedzieć się o możliwościach oferowanych przez system. Z tego miejsca można przejść do strony rejestracji i logowania. W celu założenia nowego konta, należy wybrać zakładkę signup widoczną w górnym prawym rogu ekranu. Po jej wybraniu następuje przekierowanie na widok przedstawiony na rysunku 5.4.

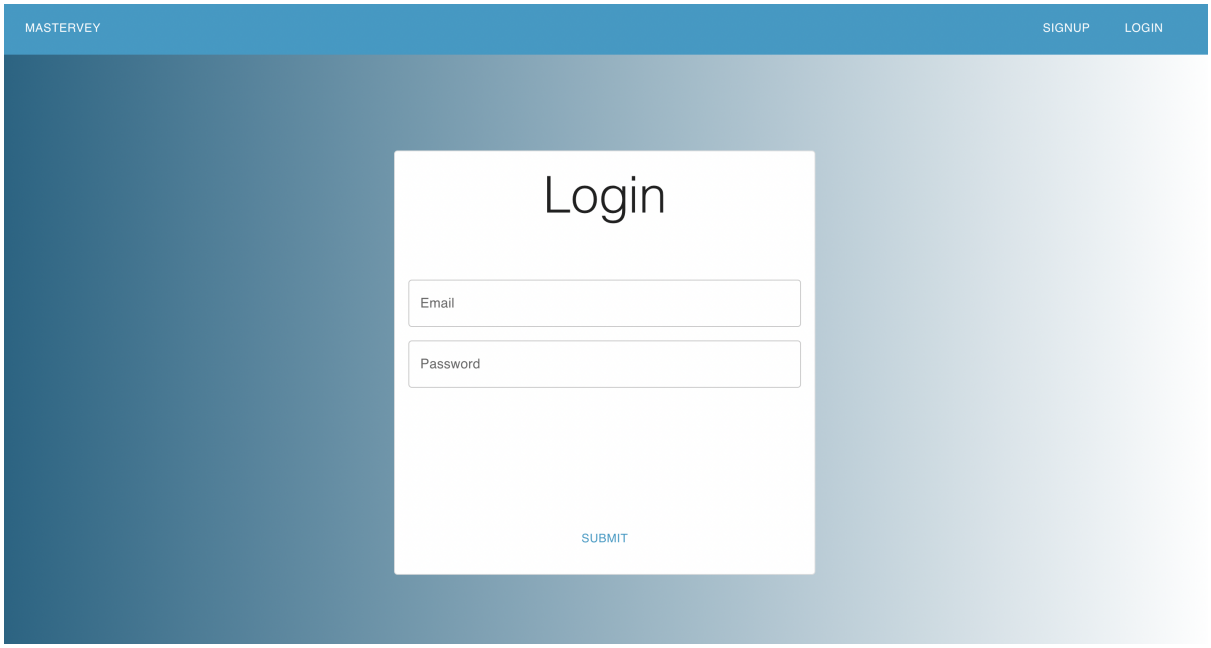


Rysunek 5.4. Strona rejestracji użytkownika. Źródło: Opracowanie własne.

W tym miejscu wymagane jest podanie prawidłowego adresu email oraz hasła o minimal-

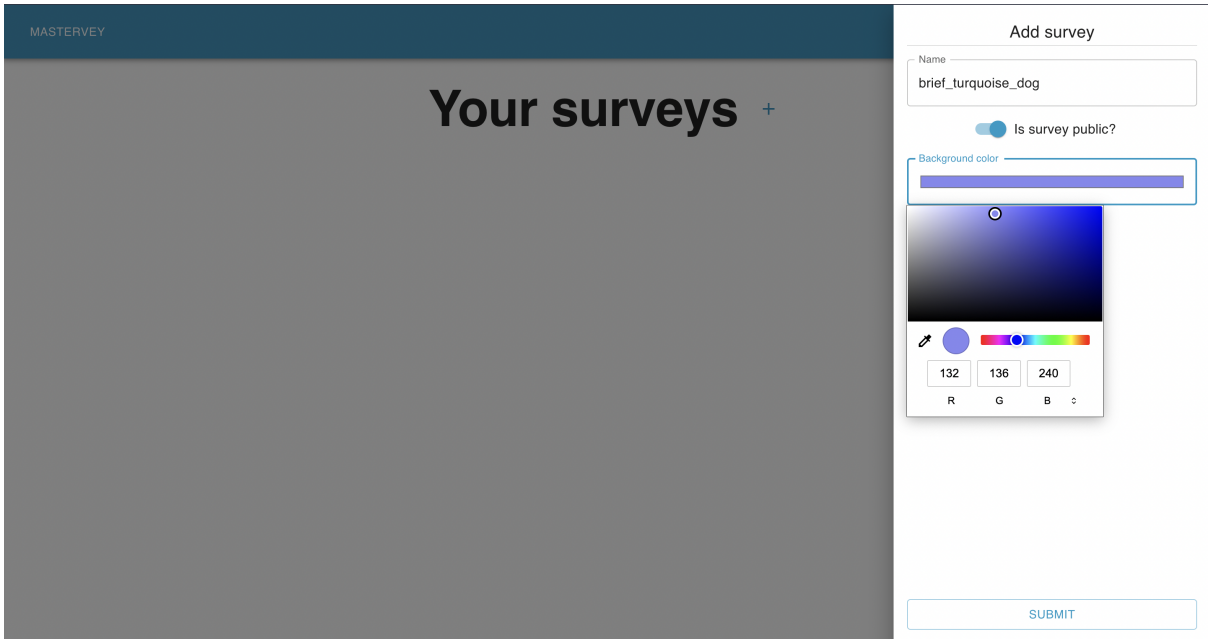
nej długości ośmiu znaków w którego składu wchodzi co najmniej jedna wielka litera i liczba. W przypadku przekazania nieprawidłowych danych lub informacji już istniejących w systemie, zostanie wyświetlony komunikat w przerwie pomiędzy polem *password* i przyciskiem oznaczonym *submit*.

W przypadku kiedy użytkownik posiada zarejestrowane konto w systemie, może się on zalogować klikając w przycisk *login* w prawym górnym ekranie. Następnie zostaje on przekierowany na stronę logowania widoczna na rysunku 5.5.



Rysunek 5.5. Ekran logowania do systemu. Źródło: Opracowanie własne.

W tym wypadku należy podać adres email oraz hasło użytkonika już istniejącego w systemie. Po pomyślnej weryfikacji wprowadzonych danych, użytkownik zostaje przekierowany na pulipt nawigacyjny gdzie widoczne są jego wszystkie ankiety. Z tego miejsca można stworzyć ankietę, naciskając znak "+" położony na prawo od napisu *"your surveys"*. Po wybraniu tej opcji zostanie otworzony panel kreowania nowej ankiety, który został przedstawiony na rysuku 5.6.

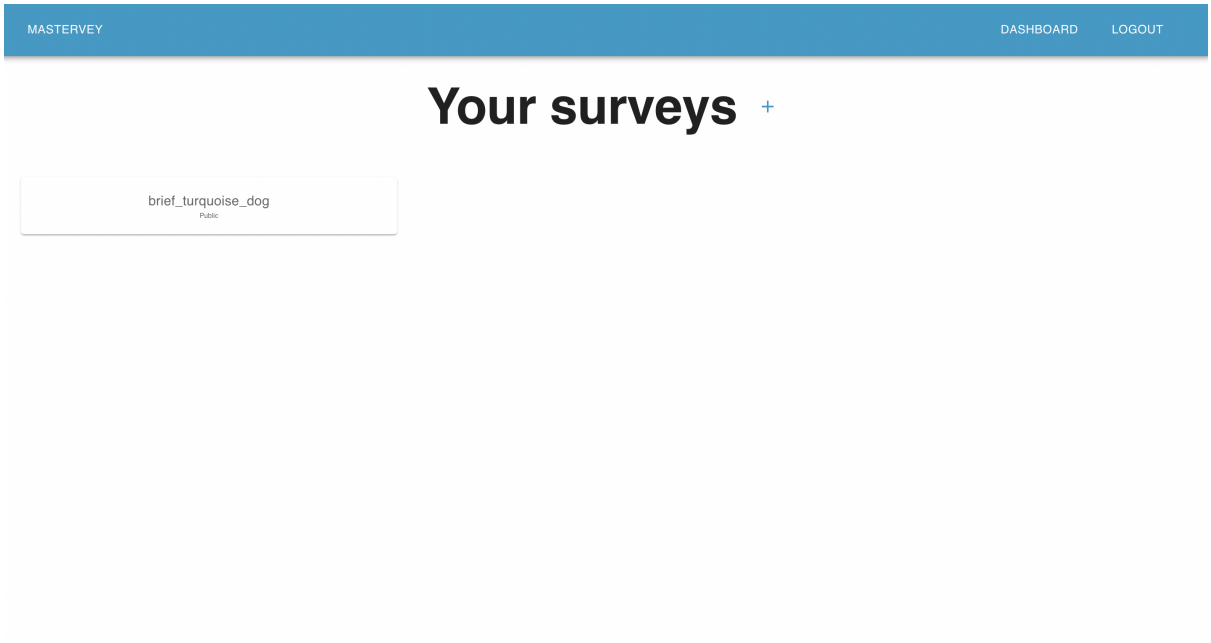


Rysunek 5.6. Panel tworzenia nowej ankiety. Źródło: Opracowanie własne.

Użytkownik w ramach tworzenia ankiety wprowadzić może m.in.

- Unikatową nazwę ankiety. W przypadku wprowadzenia istniejącej już nazwy zostaje wyświetlony odpowiedni komunikat z prośbą o zmianę tytułu. Jest to jedyna wartość w formularzu, która jest obowiązkowa do wprowadzenia.
- Informację czy odpowiedzi mogą udzielać osoby niezarejestrowane w systemie.
- Kolor tła, który wyświetlany jest przy udostępnianiu ankiety.

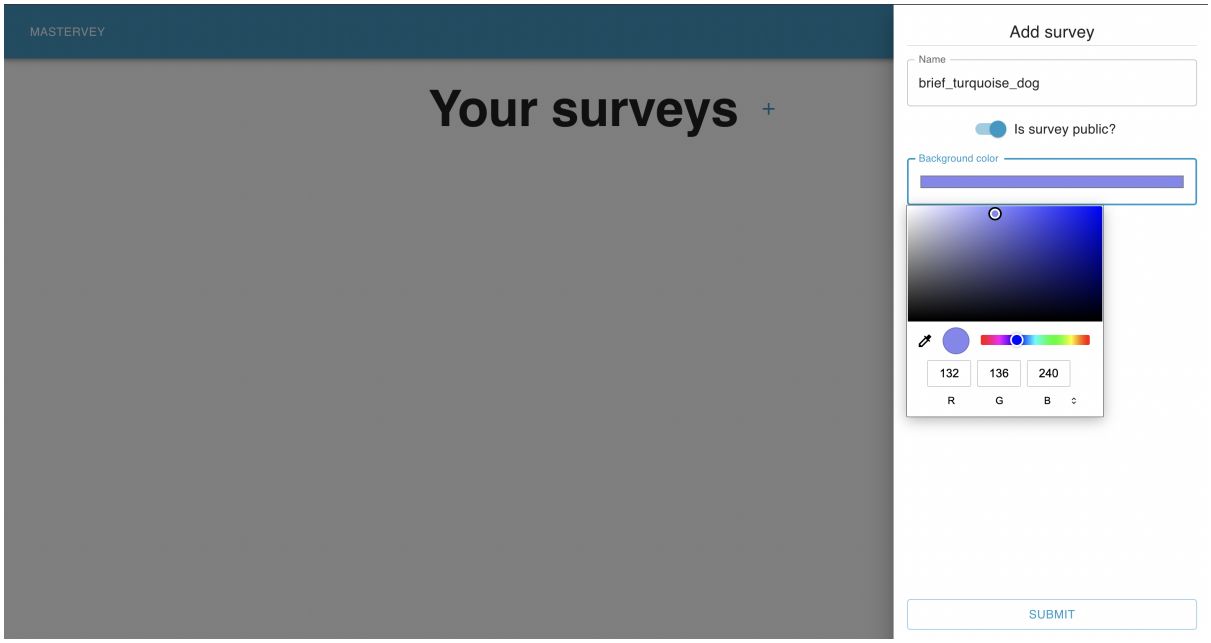
Po uzupełnieniu formularza i wysłaniu go klikając przyciskiem *submit*, nowo stworzona ankieta pojawia się na stronie głównej co zostało przedstawione na rysunku 5.7.



Rysunek 5.7. Pulpit nawigacyjny ze stworzoną ankietą. Źródło: Opracowanie własne.

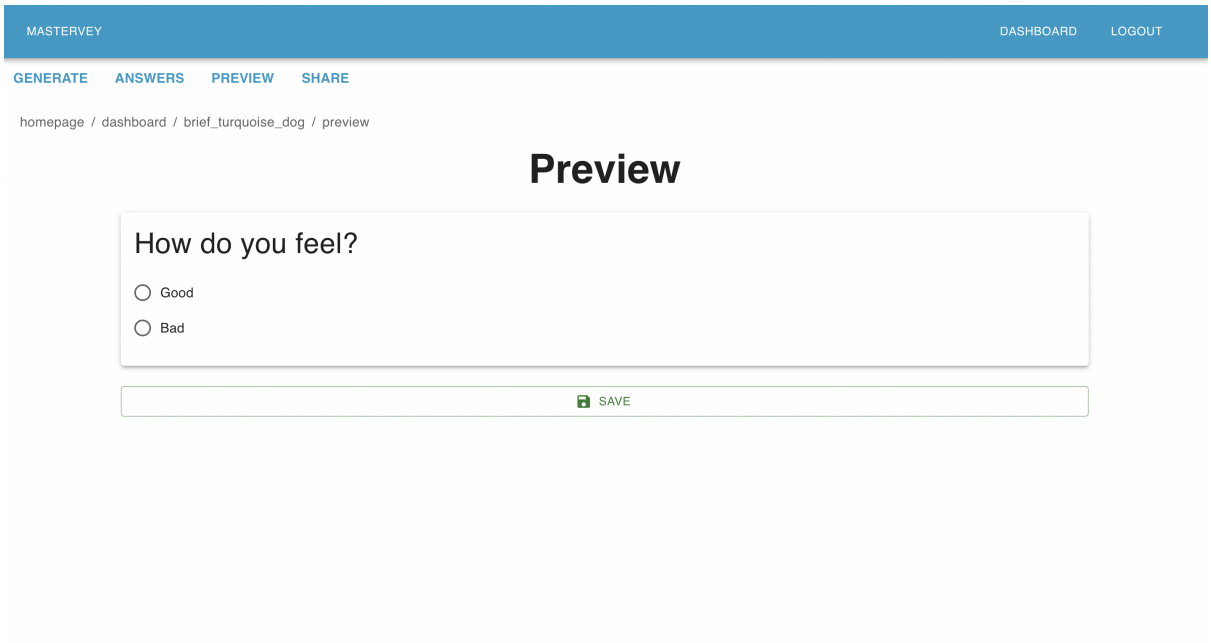
Tak wyświetlona zakładka jest jednocześnie odnośnikiem do panelu zarządzania daną ankietą. Aby wejść w opcję edycji, należy najechać na wybrany element kursorem myszy i kliknąć lewym przyciskiem myszy. Następnie użytkownik zostaje przekierowany do strony tworzenia i edycji.

W tym miejscu konfigurowane są odpowiednie pola ankiety w tym m.in. użytkownik wprowadza pytanie, wybiera formę akceptowanej odpowiedzi, oraz dla wyselekcjonowanych opcji przygotowuje możliwe reakcje dla swoich odbiorców. Tak przykładowe pytanie będące jednocześnie całą ankieta, zostało przedstawione na rysunku 5.8.



Rysunek 5.8. Formularz edycji ankiety. Źródło: Opracowanie własne.

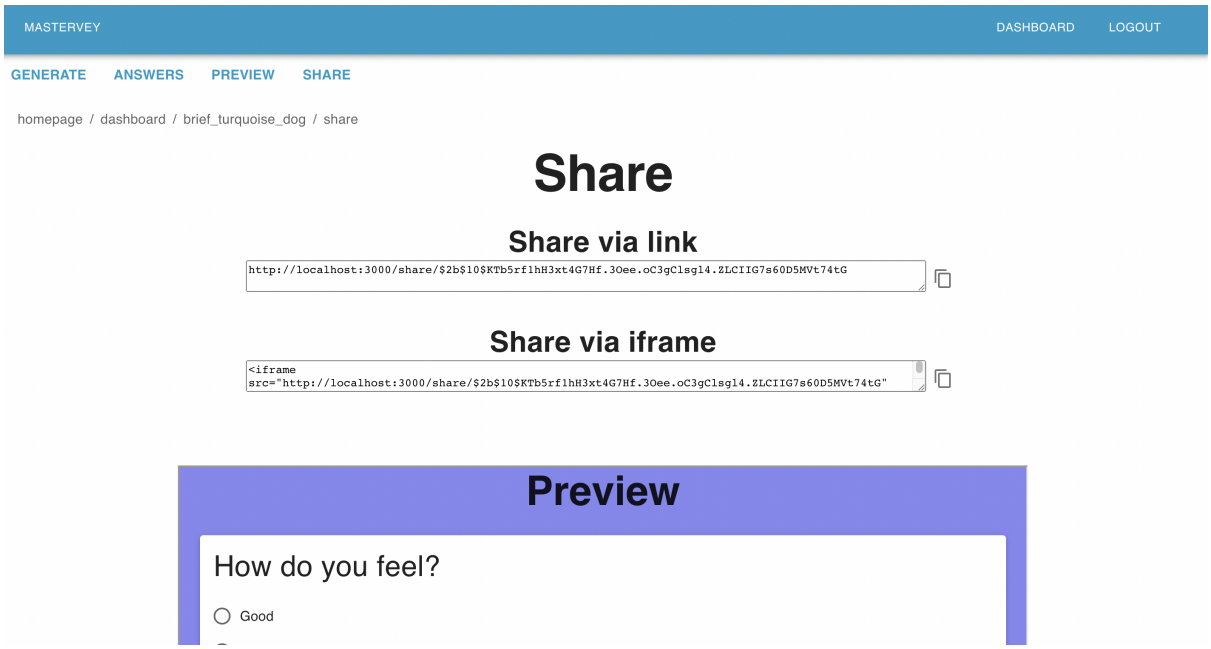
Po skonstruowaniu całości formularza należy kliknąć przycisk *save*, aby zatwierdzić wprowadzone zmiany. W celu zobaczenia jak prezentuje się nowo utworzona ankieta należy przejść do opcji *preview* widocznej pod jasno niebieskim paskiem nawigacyjnym strony. Po jej wybraniu następuje przekierowanie na stronę widoczną na rysunku 5.9.



Rysunek 5.9. Podgląd ankiety. Źródło: Opracowanie własne.

Dodatkową opcją na przetestowanie wyglądu aktualnej ankiety i jednoczesne udostępnienie

jej wybranej grupie odbiorców jest wybranie opcji *share*. Po wybraniu tej opcji następuje przekierowanie na stronę widoczną na rysunku 5.10.

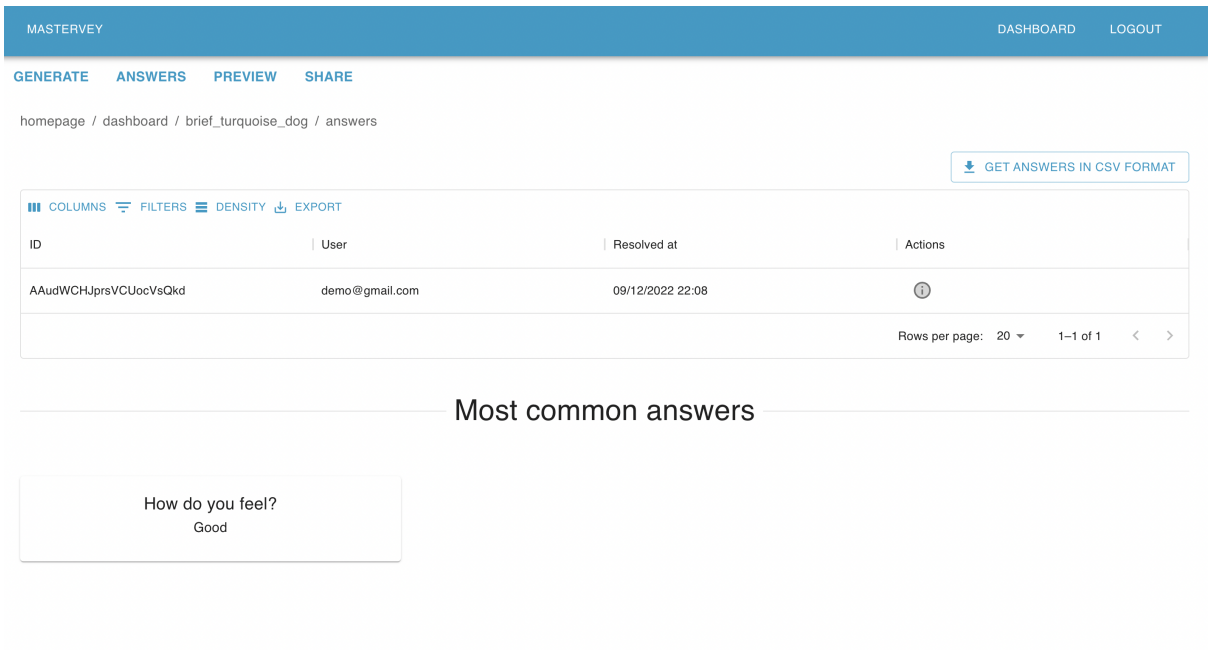


Rysunek 5.10. Panel udostępniania ankiety. Źródło: Opracowanie własne.

Użytkownik na chwilę obecną ma dwie opcje udostępnienia wybranej ankiety. Pierwsza z nich polega na przesłaniu tzw. "zahaszowanego" [67] linku bezpośrednio do odbiorcy. W tym wypadku haszowanie odbywa się przy pomocy algorytmu "bcrypt" [68]. Drugą z opcji jest umieszczenie specjalnie przygotowanego tagu "iframe" do umieszczenia go w wybranym źródle cyfrowym. Niezależnie od tego w przypadku udostępnienia formularza zastosowane są wszystkie opcje wybrane przy tworzeniu ankiety. Przykładowa strona, którą otrzymałby odbiorca poprzez link została przedstawiona na rysunku 5.11.

Rysunek 5.11. Udostępniona ankieta z zaznaczonymi opcjami. Źródło: Opracowanie własne.

Po zebraniu wyników, właściciel ankiety może przejść do zakładki *answers* w celu przejrzenia odpowiedzi. Widok tej strony został przedstawiony na rysunku 5.12.

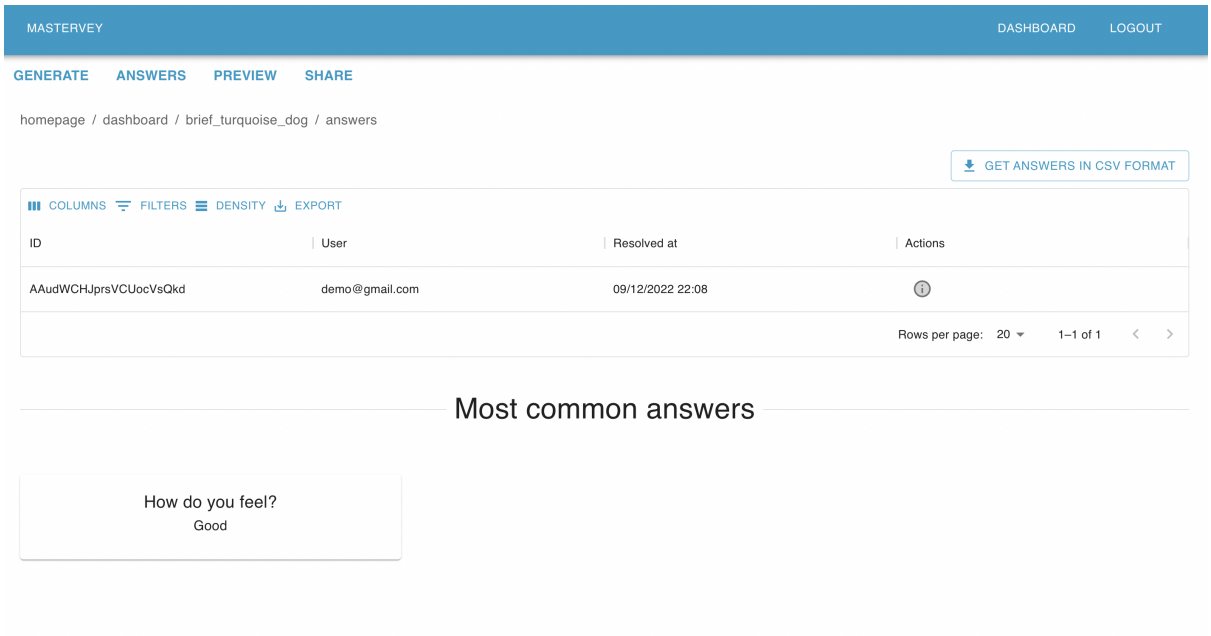


Rysunek 5.12. Strona z zebranymi wynikami. Źródło: Opracowanie własne.

Z tego widoku użytkownik może m.in.

- Wyeksportować zebrane wyniki do pliku z rozszerzeniem ".csv".

- Przejrzeć kto i o której rozwiązał daną ankietę. Dodatkowo po kliknięciu w czwartą kolumnę od lewej strony istnieje możliwość szczegółowego podglądu udzielonych odpowiedzi. Przykład takiego widoku został przedstawiony na rysnku 5.13.
- Zobaczyć najczęśniej udzielane odpowiedzi wyświetlone na dole strony.



Rysunek 5.13. Widok poszczególnie wybranych odpowiedzi. Źródło: Opracowanie własne.

W momencie ukończenia pracy z daną ankietą użytkownik może wrócić do pulpitu nawigacyjnego wybierając opcję *dashboard* widoczną w górnym prawym rogu ekranu. Zostaje również możliwość kliknięcia *logout* po której następuje wylogowanie użytkownika i przekierowanie na stronę główną.

5.5. Wady i zalety stworzonego rozwiązania

Stworzony program ma swoje blaski i cienie. Należy pamiętać, że jest to jedynie narzędzie mające swoje zastosowanie w konkretnych przypadkach. W związku z tym podrozdział ten traktuje o różnych plusach i minusach niniejszego programu.

Pierwszym z widocznych zalet niniejszego programu jest możliwość jego indywidualnego zarządzania. Projekt został stworzony z myślą o samodzielnym utrzymaniu na własnym koncie Firebase. Dzięki temu użytkownik otrzymuje pełen dostęp do zebranych przez siebie danych. Ma możliwość ich eksportu, odpowiedniej obróbki, a także ma pełną kontrolę nad

użytkownikami mającymi dostęp do ankiet i platformy. Rozwiązanie to mogłoby być bardzo ciekawą propozycją np. dla niewielkich organizacji w której każdy z pracowników tworzyłby własne ankiety na potrzeby zbadania poszczególnych części rynku.

Druga zaleta jest związana z tematem cyberbezpieczeństwa. Niejednokrotnie zdarza się, że w przypadku niedużych aplikacji kwestie bezpieczeństwa nie są tak istotne jak w np. zaawansowanych systemach bankowych. W związku z tym nierzadko temat ten bywa pomijany przez programistów aplikacji. W zaproponowanym rozwiązaniu, dzięki oparciu o rozwiązania chmurowe można zwiększyć odporność aplikacji na wybrane zagrożenia, np. wyciek haseł użytkowników. W niniejszym przypadku administrator nie ma dostępu do haseł, a te przetrzymywane są przez platformę. Oczywiście nie oznacza to, że wyciek wrażliwych informacji nie może zdarzyć się takiemu serwisowi jak Firebase. Jednakże możliwości i już wdrożone rozwiązania [69] przez firmę są często nieporównywalnie większe w stosunku do mniejszych organizacji.

Trzecią zaletą jest szybka i prosta możliwość dostosowania projektu pod własne potrzeby. Dzięki rozbudowanemu i bogatemu repozytorium npm, każdy z użytkowników bez problemu może dostosować projekt pod indywidualne potrzeby. Dodatkowo projekt nie korzysta z zaawansowanych menadżerów stanów jakim jest np. Redux [70]. Sprawia to, że próg wejścia nawet dla osób niezaznajomionych z językiem TypeScript jest dużo niższy niż w przypadku innych alternatywnych projektów opartych o te same technologie.

Kolejną z zalet jest możliwość tworzenia prywatnych i publicznych ankiet. Ze względu na tę funkcjonalność użytkownik może według własnych preferencji wybierać grupy docelowe którym chce udostępnić przygotowany formularz. Dodatkowo dzięki kontroli zarejestrowanych osób w prosty sposób może zablokować dostęp nieupoważnionym.

Ostatnią z zalet jest możliwość dołączania ankiet do własnych stron internetowych. Dzięki przygotowaniu w aplikacji specjalnej sekcji użytkownik otrzymuje unikatowy fragment kodu. Ten po wklejeniu w odpowiednim miejscu w docelowej stronie internetowej, wyświetli stworzoną wcześniej ankietę.

Z drugiej strony w projekcie możemy też wytypować wady. Głównym niezbyt dobrym rozwiązaniem jest brak zadbania o użytkowników nie technicznych. Projekt właściwie od początku wymaga od użytkownika znajomości technicznych m.in. do stworzenia i konfiguracji projektu.

Drugim z problemów jest brak zaimplementowanych ustawień pozwalającym użytkown-

ikom indywidualnie dostosowywać aplikację pod własne potrzeby. Ze względu na dużą różnorodność oczekiwań użytkowników, udostępnienie opcji dostosowania np. wyglądu aplikacji lub możliwości zbierania meta danych mogłoby okazać się przydatnym rozwiązaniem.

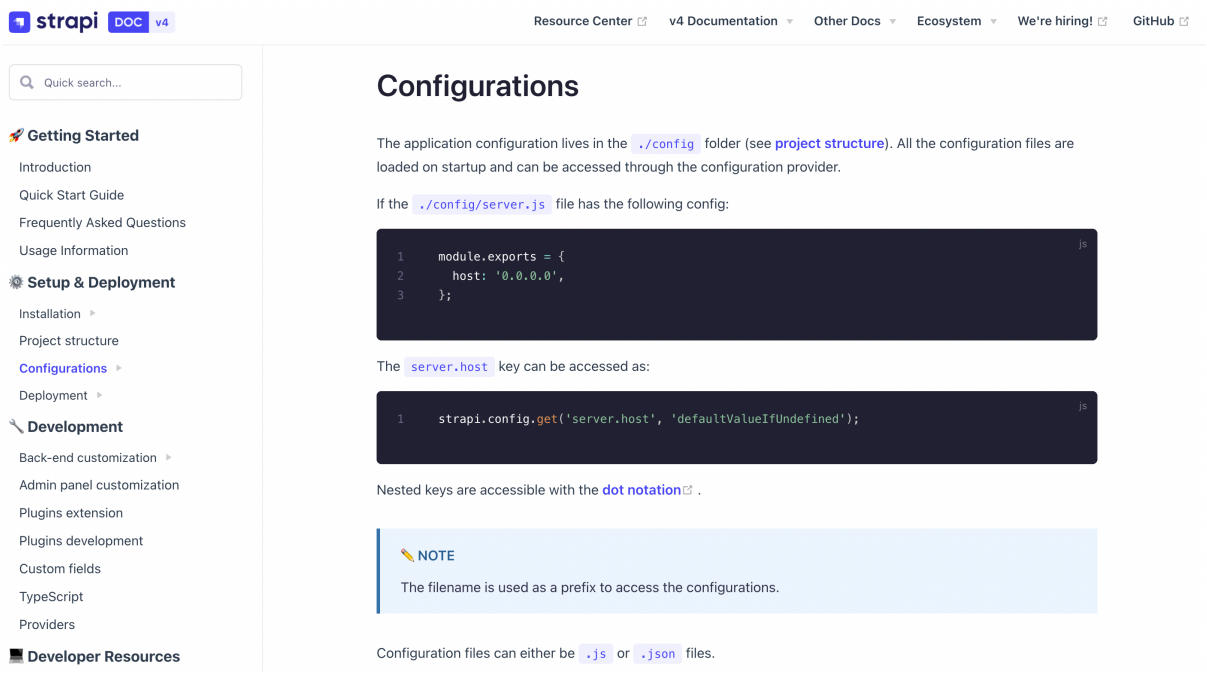
Ostatnią z wad jest paradoksalnie brak kontroli nad wszystkimi aspektami aplikacji. W przypadku jeśli użytkownik końcowy chciałby bazować na stworzonym projekcie jest on w pełni uzależniony od platformy Firebase. W momencie kiedy potrzebowałby skorzystać z rozwiązania niedostępnego na niniejszej platformie, byłby on zmuszony do znacznego skomplikowania architektury swojego systemu. To w konsekwencji mogłoby powodować zwiększone koszty utrzymania aplikacji lub zwiększenie podatności na niektóre zagrożenia cybernetyczne (np. na ataki *man in the middle* [71]).

5.6. Możliwości rozwoju i usprawnienia

W książce *"Atomowe nawyki. Drobne zmiany, niezwykłe efekty"* autorstwa Clear James'a [72], przytoczony zostaje przykład historii usprawnienia brytyjskiej załogi kolarskiej. W 2003 roku osiągała ona fatalne wyniki sportowe. Po wprowadzeniu w życie niewielkich modyfikacji wprowadzanych każdego dnia drużyna ta stała się jedną z najlepszych na świecie. Zgodnie z tym nurtem podrozdział ten dedykowany został puencie dotyczącej możliwych ulepszeń i wprowadzania coraz to nowszych zmian do bezserwerowego systemu zarządzania ankietami.

Pierwszą rzeczą od której należałoby rozpocząć ulepszanie aplikacji jest m.in. pozbycie się tzw. długu technologicznego [73]. W uproszczeniu są to zaległości techniczne które zostały poczynione w trakcie tworzenia projektu. W tym przypadku taką sytuacją objęta jest dokumentacja techniczna dla użytkowników. Na chwilę obecną nie została ona wdrożona w stopniu w którym znacząco przyspieszyłoby to wykorzystanie niniejszego projektu w celach użytkowych. Związane jest to niejako z pierwszą z wad tej inicjatywy opisanej w rozdziale 5.4.

Na rysunku 5.14 został przedstawiony fragment źródła technicznego aplikacji CMS "Strapi" [74]. Na bazie tego pierwowzoru można by oprzeć dokumentację do niniejszego systemu.



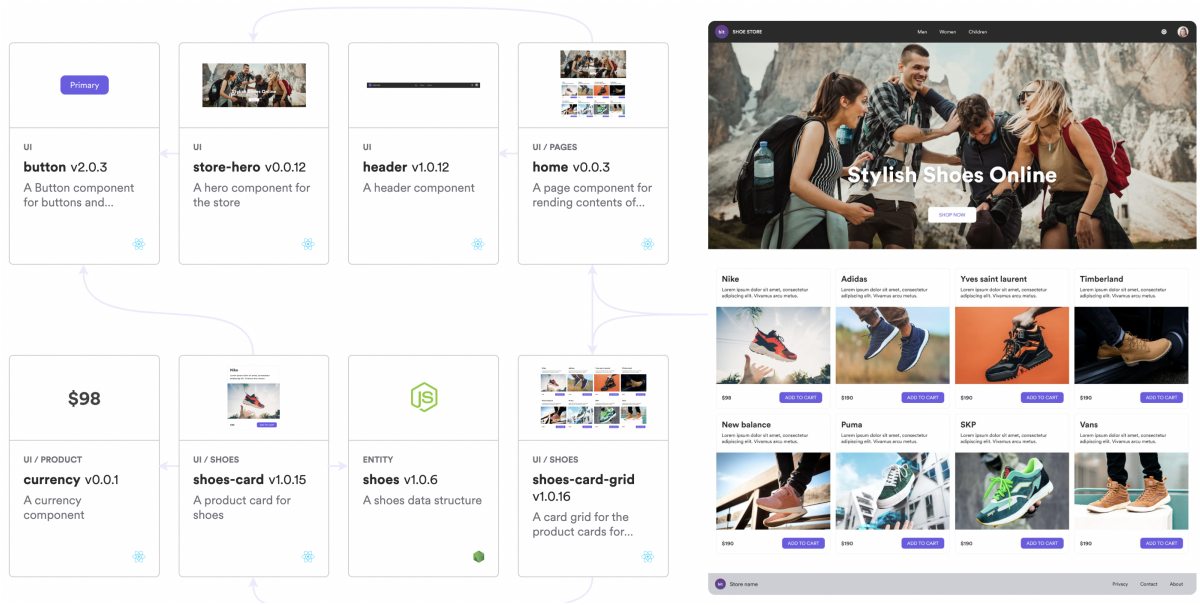
Rysunek 5.14. Fragment dokumentacji systemu CMS - Strapi. Źródło: [75].

Jest to o tyle reprezentatywny przykład, iż dokumentacja ta jest bogata w informacje w tym m.in.

- Zawiera opis wdrożenia na kilka wybranych platform chmurowych jak i informacje o sposobie samodzielnego hostingu.
- Przekazuje informacje o wybranych możliwościach konfiguracji platformy.
- Bezpośrednio ze strony odwołuje do kodu źródłowego skąd można go pobrać.
- Zawiera odpowiednie samouczki dla osób nietechnicznych chcących skorzystać z programu.

Tak szeroka dokumentacja prowadzi nas do drugiego usprawnienia, jakim jest udostępnienie aplikacji na różne platformy chmurowe. Szczegółowo problem ten został przedstawiony w rozdziale 5.4. Dzięki rozszerzeniu opcji hostingu systemu na wielu platformach istnieje szansa na przyspieszenie popularności aplikacji. Tym samym rośnie również prawdopodobieństwo wzrostu liczby osób zainteresowanych projektem mogących wziąć czynny udział w usprawnianiu otwartoźródłowej aplikacji. Dodatkową korzyścią byłoby również wsparcie większej liczby osób, które bezserwerowy system do zarządzania ankiet chciałyby dołączyć do swoich już istniejących aplikacji.

Trzecim pomysłem na rozbudowę aplikacji mogłoby być stworzenie własnej biblioteki komponentów potrzebnej do budowy części elementów systemu. Pomysł ten oparty jest o platformę "bit" [76]. W dużym skrócie, strona ta pozwala na udostępnianie i dzielenie się różnymi komponentami stworzonymi przez społeczność aplikacji internetowej. Przykład takich elementów przedstawiony został na rysunku 5.15. Dzięki takiemu zastosowaniu niniejszy system byłby jeszcze bardziej dopasowany pod potrzeby indywidualnych użytkowników, a tym samym zwiększałby swoją konkurencyjność.



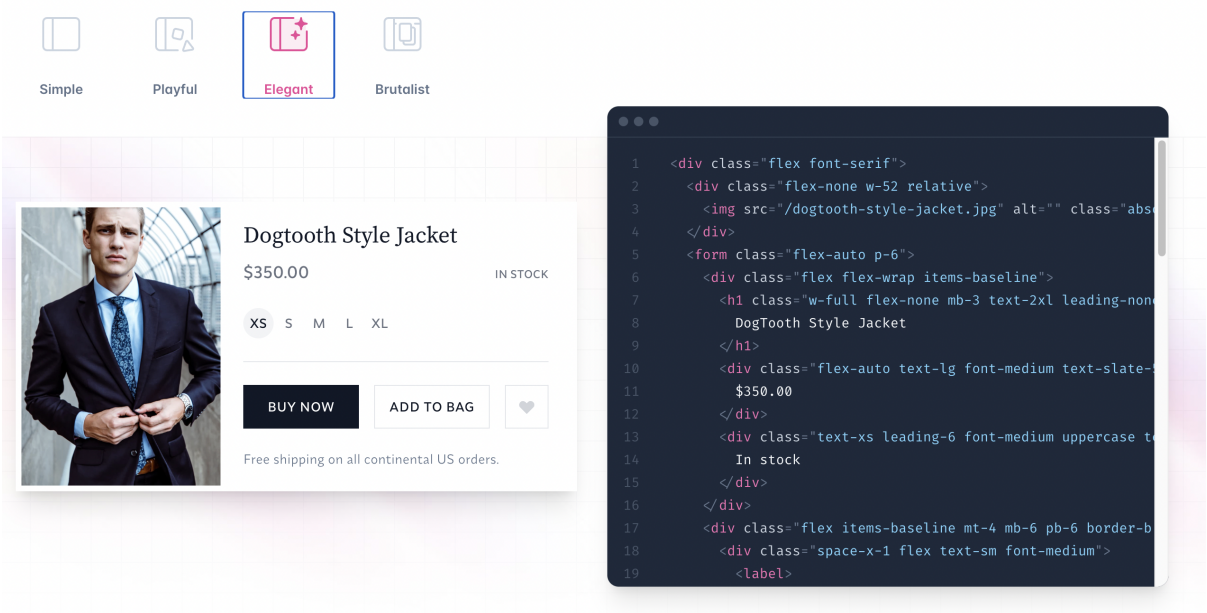
Rysunek 5.15. Strona główna aplikacji bit z przykładowymi komponentami. Źródło: [76].

Następnym z pomysłów na rozbudowę byłaby możliwość adaptacji aplikacji do kilku wybranych nowoczesnych bibliotek internetowych jak ma to miejsce, np. w modułach "tanstack" [77]. Paczka dostępna w repozytorium npm początkowo udostępniała jedynie wybrane funkcjonalności bibliotece React.js. Obecnie ze względu na coraz to nowsze rozwiązania internetowe (o czym świadczyć mogą statystyki dostępne na stronie "State of JS" [35]) twórcy aplikacji zdecydowali się na poszerzenie już gotowych funkcji na inne platformy. Dzięki temu wzrasta popularność niniejszego rozwiązania, a tym samym zysk dla autora modułów.

Kolejnym z pomysłów jest możliwość stworzenia sklepu z dodatkami dla użytkowników programu. Pomysł ten również czerpie z platformy strapi. W tym przypadku grono fanów, stworzyło różnego rodzaju wtyczki w tym m.in. umożliwiające wykorzystanie internetowego tłumacza do translacji wpisów na inne języki. Taki aspekt zdecydowanie byłby przydatny w przypadku tworzenia ankiet mających za zadanie zebrać dane z grup ludzi różnych

narodowości.

Należy również pamiętać o możliwości usprawniania już istniejących funkcjonalności. Pierwszą z opcji na ulepszenie już istniejącego elementu systemu jest m.in. ulepszenie opcji stylowania ankiet. Pomimo wyboru opcji tła ankiety, warto by dodać dodatkowe opcje zwiększające atrakcyjność udostępnianych formularzy. Dobrym pomysłem byłoby wykonanie kilku przykładowych motywów, które użytkownik mógłby zastosować. Pomysł ten czerpie z biblioteki "tailwindcss" [78], której propozycje motywów pojedynczego komponentu zostały przedstawione na rysunku 5.16.

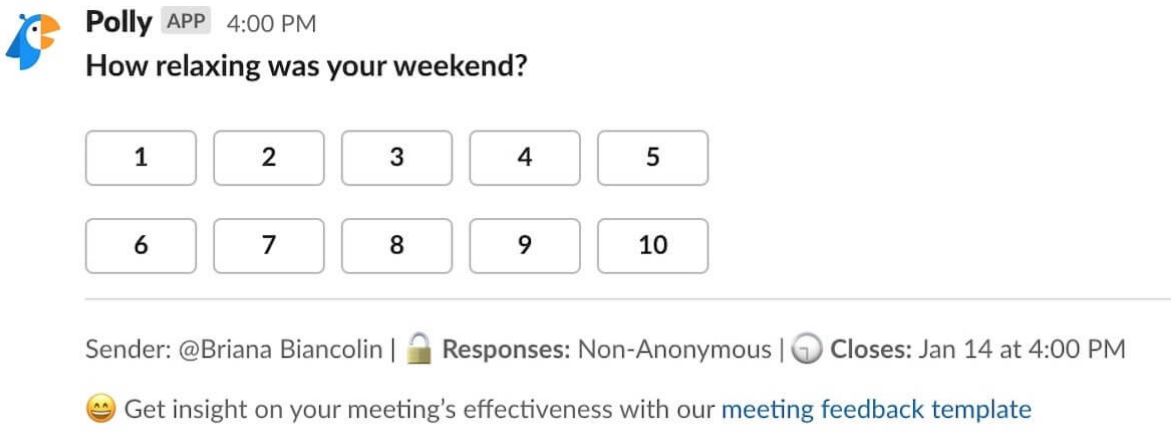


Rysunek 5.16. Komponent Tailwindcss wykorzystujący jeden z dostępnych motywów. Źródło: [78].

Następną z możliwości jest ulepszenie panelu odpowiedzialnego za statystyki zebranych ankiet. Przykładem na bazie którego mogłaby powstać ulepszona wersja jest, np. widok znajdujący się na rysunku 2.4. Przedstawienie danych w odpowiedniej formie wizualnej uprościłoby korzystanie z serwisu zwłaszcza użytkownikom niewymagających skomplikowanej analizy ankiet.

Ostatnim sposobem usprawnienia niniejszego projektu jest zwiększenie możliwości udostępniania stworzonych ankiet. Na chwilę obecną system umożliwia dołączenie wygenerowanej ankiety do własnej strony internetowej, a także udostępnienie jej przy pomocy specjalnego linku. Dobrym pomysłem byłaby możliwość wygenerowania formularzy w formie dostosowanej do druku. Poza fizyczną możliwością udostępniania na szczególną uwagę zasługuje

opcja wtyczek lub dodatków do popularnych komunikatorów do których można by podłączyć obecną aplikację. Przykładem takiego rozszerzenia może być program "Polly" [79] dostępny dla komunikatora "Slack" [80]. Przykładowa ankieta została przedstawiona na rysunku 5.17.



Rysunek 5.17. Ankieta Polly dostosowana do obsługi ankiet w komunikatorze Slack. Źródło: [79].

Należy pamiętać również o ograniczeniach wynikających z wykorzystywanych popularnych komunikatorów. Jedną z takich niedogodności mogą być, np. względy bezpieczeństwa z którym borykają się aplikacje typu "messenger" autorstwa firmy Meta. Zgodnie z informacjami zawartymi w artykule *"Why You Should Stop Using Facebook Messenger"*[81], komunikator boryka się z niedostatecznym poziomem zabezpieczeń. W momencie tworzenia wtyczki na każdą z takich platform ważna byłaby również analiza bezpieczeństwa danego systemu.

6. Podsumowanie

Dzięki przeprowadzonej analizie, sprawdzeniu dostępnych narzędzi i ich porównaniu możliwe było stworzenie prototypu bezserwerowego systemu do zarządzania ankiet. Aplikacja dzięki wykorzystanym technikom i dobraniu odpowiednich narzędzi cechuje się uniwersalnym podejściem w temacie przetwarzania i zbierania danych z formularzy.

Najważniejszą konkluzją płynącą ze stworzonej pracy jest przede wszystkim koncept na którym oparty został prototyp aplikacji. Pomimo części wad, pomysł zawiera wiele interesujących rozwiązań. Po odpowiednim dopracowaniu projektu, a także zaimplementowaniu wybranych ulepszeń platforma mogłaby znaleźć wiele praktycznych zastosowań. Dzięki niej użytkownicy końcowi mogliby znacząco poszerzyć możliwości zbierania informacji od swoich grup odbiorczych. Dodatkowo w zależności od potrzeby mieliby oni odpowiednie narzędzia do wyciągania właściwych wniosków z otrzymanych wyników.

Uważam, że cel pracy został osiągnięty, a przytoczone przykłady wraz z opisem wyczerpująco przedstawiły temat jakim jest bezserwerowe podejście do systemu zarządzania ankietami. Ze względu na wykorzystanie nowych rozwiązań ze świata biznesu i technologii zakładam, że rozwiązanie po odpowiednim dopracowaniu niosłoby ze sobą wiele korzyści. Projekt w porównaniu do konkurencji cechuje znaczne obniżenie kosztów obsługi, Zapewnia także użytkownikom swobodę w przechowywanych i przetwarzanych danych. Przy możliwości zachowania odpowiedniego skalowania, aplikacja wyróżnia się również większą odpornością na wybrane cyberataki. Dzięki temu koncepcja ta zdaje się wpisywać w dzisiejsze potrzeby i trendy rynku, związane z migracją systemów do platform chmurowych.

Literatura

[1] Wiesław Pawłowicz. *Dane to najważniejsze paliwo dla współczesnego biznesu (z dnia 12.11.2022)*. URL: https://www.computerworld.pl/wywiad/Dane-to-najwazniejsze-paliwo-dla-wspolczesnego-biznesu,413902.html.

[2] Kiran Das. *Fuel Of Future: Why Data Is Fuel In The Modern Economy (z dnia 12.11.2022)*. URL: https://www.linkedin.com/pulse/fuel-future-why-data-modern-economy-kiran-das/.

[3] InterSystemsCorp. *Data is the Fuel (z dnia 12.11.2022)*. URL: https://www.youtube.com/watch?v=gabcqVCPfQM.

[4] Nikita Sheth. *How does Meta (Facebook) make money? (z dnia 12.11.2022)*. URL: https://finty.com/us/business-models/meta/.

[5] Avantika Monnappa. *How Facebook uses Big Data: The Good, the Bad, and the Ugly (z dnia 12.11.2022)*. URL: https://www.simplilearn.com/how-facebook-is-using-big-data-article.

[6] Qualtarics. *How to use statistical analysis methods and tests for surveys (z dnia 18.11.2022)*. URL: https://www.qualtrics.com/experience-management/research/survey-analysis-types/.

[7] Karin Kelley. *What is Data Analysis? Methods, Process and Types Explained (z dnia 18.11.2022)*. URL: https://www.simplilearn.com/data-analysis-methods-process-types-article.

[8] Mark Pickavance Brian Turner. *Best survey tools of 2022 (z dnia 14.11.2022)*. URL: https://www.techradar.com/best/best-survey-tools.

[9] Kristina Lauren. *SurveyMonkey vs. Google Forms: Which should you use? [2022] (z dnia 14.11.2022)*. URL: https://zapier.com/blog/google-forms-vs-surveymonkey/.

[10] *(z dnia 14.11.2022)*. URL: https://www.surveymonkey.com/.

[11] *(z dnia 14.11.2022)*. URL: https://docs.google.com/forms/u/0/.

[12] *(z dnia 14.11.2022)*. URL: https://www.surveymonkey.com/.

[13] Lime Survey. *(z dnia 17.11.2022)*. URL: https://github.com/LimeSurvey/LimeSurvey.

[14] Lime Survey. *(z dnia 17.11.2022)*. URL: https://github.com/LimeSurvey/LimeSurvey/graphs/contributors.

[15] FOSSA Editorial Team. *Open Source Software Licenses 101: GPL v2 (z dnia 17.11.2022)*. URL: https://fossa.com/blog/open-source-software-licenses-101-gpl-v2/.

[16] *(z dnia 17.11.2022)*. URL: https://demo.limesurvey.org/index.php?r=admin.

[17] *Czym są Exploity Zero-Day? (z dnia 17.11.2022)*. URL: https://trybawaryjny.pl/czym-sa-exploity-zero-day/.

[18] Anna Fitzgerald. *20 WordPress Statistics You Should Know in 2022 (z dnia 17.11.2022)*. URL: https://blog.hubspot.com/website/wordpress-stats.

[19] Formidable Forms. *(z dnia 17.11.2022)*. URL: https://formidableforms.com/wordpress-survey-plugin/.

[20] OSTraining. *(z dnia 18.11.2022)*. URL: https://www.youtube.com/watch?v=NjWg6sCQWnU.

[21] *(z dnia 07.12.2022)*. URL: https://www.oracle.com/pl/cloud/cloud-native/functions/what-is-serverless/.

[22] *Why Is Cloud Computing Growing in Popularity (z dnia 18.11.2022)*. URL: https://www.amazingsupport.co.uk/why-is-cloud-computing-growing-in-popularity/.

[23] Wiktor Szura. *Co to jest hosting WWW - Definicja (z dnia 18.11.2022)*. URL: https://ks.pl/slownik/co-to-jest-hosting.

[24] Anni Burchfiel. *What is a BLOB (Binary Large Object)? Can it be Tokenized? (z dnia 18.11.2022)*. URL: https://www.tokenex.com/blog/ab-what-is-a-blob-binary-large-object-can-it-be-tokenized/.

[25] Mary Zhang. *Top 10 Cloud Service Providers (z dnia 18.11.2022)*. URL: https://dgtlinfra.com/top-10-cloud-service-providers-2022/.

[26] Centrum Doradczo Szkoleniowe MALON GROUP Sp. z o. o. *Technika Analityczna SWOT (z dnia 18.11.2022)*. URL: https://www.iso.org.pl/uslugi-zarzadzania/wdrazanie-systemow/zarzadzanie-strategiczne/analiza-swot/.

[27] Vishal Patel. *Multiple ways of deploying a Node.js application into Azure App Service (z dnia 22.11.2022)*. URL: https://medium.com/@vishal1909/multiple-ways-of-deploying-a-node-js-application-into-azure-app-service-51c0173f6731.

[28] Jakub Gutkowski. *Dwa typy Serverless (z dnia 22.11.2022)*. URL: https://blog.gutek.pl/2017/04/20/dwa-typy-serverless/.

[29] TechTarget Contributor. *Infinite loop (endless loop)(z dnia 22.11.2022)*. URL: https://www.techtarget.com/whatis/definition/infinite-loop-endless-loop.

[30] Tyson Cadenhead. *I Wrote a Recursive Lambda Function That Cost My Company Hundreds of Dollars(z dnia 22.11.2022)*. URL: https://thenable.io/how-a-recursive-lambda-function-cost-hundreds-of-dollars.

[31] Tyson Cadenhead. *I Wrote a Recursive Lambda Function That Cost My Company Hundreds of Dollars(z dnia 22.11.2022)*. URL: https://thenable.io/how-a-recursive-lambda-function-cost-hundreds-of-dollars.

[32] *Avoid infinite retries (z dnia 22.11.2022)*. URL: https://cloud.google.com/functions/docs/samples/functions-tips-infinite-retries#functions_tips_infinite_retries-nodejs.

[33] Vladimir Sumina. *Top 26 Cloud Computing Statistics (z dnia 22.11.2022)*. URL: https://www.cloudwards.net/cloud-computing-statistics/.

[34] Saurabh Barot. *Firebase Vs AWS: Which One to Choose in 2022? (z dnia 22.11.2022)*. URL: https://aglowiditsolutions.com/blog/firebase-vs-aws/.

[35] Brian Holt. *Front-end Frameworks (z dnia 22.11.2022)*. URL: https://2021.stateofjs.com/en-US/libraries/front-end-frameworks/.

[36] Adwaith KS. *React.js Basics – The DOM, Components, and Declarative Views Explained (z dnia 22.11.2022)*. URL: https://www.freecodecamp.org/news/reactjs-basics-dom-components-declarative-views/.

[37] Codecademy Team. *What Is a Framework? (z dnia 23.11.2022)*. URL: https://www.codecademy.com/resources/blog/what-is-a-framework/.

[38] *What is Server-Side Rendering? (z dnia 23.11.2022)*. URL: https://www.heavy.ai/technical-glossary/server-side-rendering.

[39] *Co to jest pozycjonowanie? Informacje podstawowe o SEO (z dnia 23.11.2022)*. URL: https://pomoc.home.pl/baza-wiedzy/pozycjonowanie-informacje-podstawowe.

[40] *How To Improve SEO Rankings in 2022 (z dnia 23.11.2022)*. URL: https://www.quicksprout.com/ways-to-improve-seo-ranking/.

[41] ishaniaggarwal. *What is Lazy Loading? (z dnia 23.11.2022)*. URL: https://www.geeksforgeeks.org/what-is-lazy-loading/.

[42] *Proxy - co to jest i do czego służy? (z dnia 23.11.2022)*. URL: https://www.morele.net/wiadomosc/proxy-co-to-jest-i-do-czego-sluzy/18056/.

[43] *What is Middleware? Technology's Go-to Middleman (z dnia 23.11.2022)*. URL: https://www.talend.com/resources/what-is-middleware/.

[44] *(z dnia 23.11.2022)*. URL: https://nextjs.org/.

[45] *What is npm? (z dnia 23.11.2022)*. URL: https://www.w3schools.com/whatis/whatis_npm.asp.

[46] Paweł Mansfeld. *Query – co to jest, zalety i wady – czy warto używać? (z dnia 23.11.2022)*. URL: https://mansfeld.pl/programowanie/jquery-co-to-jest-czy-warto-uzywac/.

[47] *The Most In-Demand Programming Languages for 2022 (z dnia 23.11.2022)*. URL: https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/.

[48] Paweł Mansfeld. *Introduction to Azure Functions (z dnia 23.11.2022)*. URL: https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview.

[49] Kamil Nahotko. *MongoDB, czyli łatwoskalowalna baza danych NoSQL (z dnia 23.11.2022)*. URL: https://boringowl.io/tag/mongodb.

[50] *JSON - Introduction (z dnia 23.11.2022)*. URL: https://www.w3schools.com/js/js_json_intro.asp.

[51] *9.15. JSON Functions and Operators (z dnia 23.11.2022)*. URL: https://www.postgresql.org/docs/9.3/functions-json.html.

[52] Darcy Wood. *Together We Build: an AWS 3-Tier Architecture (z dnia 23.11.2022)*. URL: https://towardsaws.com/together-we-build-an-aws-3-tier-architecture-62db9bba4f3a.

[53] *Higher-Order Components (z dnia 26.11.2022)*. URL: https://reactjs.org/docs/higher-order-components.html.

[54] Jenny Palomino Leah Wasser. *Lesson 4. DRY Code and Modularity (z dnia 26.11.2022)*. URL: https://www.earthdatascience.org/courses/intro-to-earth-data-science/write-efficient-python-code/intro-to-clean-code/dry-modular-code/.

[55] *Context (z dnia 26.11.2022)*. URL: https://reactjs.org/docs/context.html.

[56] Tomasz Kozon. *Czym jest CLI — kiedy i dlaczego warto sięgnąć po wiersz poleceń? (z dnia 26.11.2022)*. URL: https://boringowl.io/blog/cli.

[57] *Introducting Hooks (z dnia 26.11.2022)*. URL: https://reactjs.org/docs/hooks-intro.html.

[58] *Pages (z dnia 26.11.2022)*. URL: https://nextjs.org/docs/basic-features/pages.

[59] Bartosz Dąbek. *Zmienne środowiskowe – Czyli Jak Poprawić Produktywność! (z dnia 26.11.2022)*. URL: https://www.bdabek.pl/zmienne-srodowiskowe-czyli-jak-poprawic-produktywnosc-%5C%F0%5C%9F%5C%98%5C%B2/.

[60] Dark Archon. *Jak działa emulator? (z dnia 26.11.2022)*. URL: https://arhn.eu/2018/07/jak-dziala-emulator/.

[61] *package.json (z dnia 26.11.2022)*. URL: https://docs.npmjs.com/cli/v9/configuring-npm/package-json.

[62] Marcin Czarkowski. *Destrukturyzacja – Powtórka przed ReactJS 3 (z dnia 26.11.2022)*. URL: http://www.algosmart.pl/destrukturyzacja-powtorka-reactjs-3/.

[63] Marius Schulz. *Nullish Coalescing: The ?? Operator in TypeScript (z dnia 26.11.2022)*. URL: https://mariusschulz.com/blog/nullish-coalescing-the-operator-in-typescript.

[64] *(z dnia 26.11.2022)*. URL: https://react-query-v3.tanstack.com/.

[65] Dominik Szczepaniak. *Wstęp do REST API (z dnia 27.11.2022)*. URL: https://devszczepaniak.pl/wstep-do-rest-api/.

[66] *Wykonuj proste i złożone zapytania w Cloud Firestore (z dnia 27.11.2022)*. URL: https://firebase.google.com/docs/firestore/query-data/queries#node.js.

[67] *Na czym polega Hashowanie? (z dnia 06.12.2022)*. URL: https://academy.binance.com/pl/articles/what-is-hashing.

[68] Dan Arias. *Hashing in Action: Understanding bcrypt (z dnia 06.12.2022)*. URL: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/.

[69] *Privacy and Security in Firebase (z dnia 27.11.2022)*. URL: https://firebase.google.com/support/privacy.

[70] *(z dnia 27.11.2022)*. URL: https://redux.js.org/.

[71] Przemysław Szmaj. *Atak Man in the middle. Na czym polega i jak się przed nim bronić? (z dnia 27.11.2022)*. URL: https://geek.justjoin.it/atak-man-in-the-middle-na-czym-polega-i-jak-sie-przed-nim-bronic/.

[72] Clear James. *Atomowe nawyki. Drobne zmiany, niezwykłe efekty*. Wydawnictwo Galaktyka, 2019.

[73] Piotr Rawski. *Dług technologiczny – czym jest i jak nim zarządzać? (z dnia 28.11.2022)*. URL: https://goodpoint.blog/dlug-technologiczny-czym-nim-zarzadzac/.

[74] Przemysław Durał. *Co to jest CMS - Definicja (z dnia 28.11.2022)*. URL: https://ks.pl/slownik/co-to-jest-cms.

[75] *Configurations (z dnia 28.11.2022)*. URL: https://docs.strapi.io/developer-docs/latest/setup-deployment-guides/configurations.html.

[76] *(z dnia 28.11.2022)*. URL: https://bit.dev/.

[77] *(z dnia 28.11.2022)*. URL: https://tanstack.com/.

[78] *(z dnia 28.11.2022)*. URL: https://tailwindcss.com/.

[79] Briana Biancolin. *Want to Create a Slack Poll? Follow These 4 Steps (z dnia 28.11.2022)*. URL: https://www.polly.ai/blog/slack-poll.

[80] Michał Serwiński. *Komunikator Slack – pięć rad, które usprawnią pracę twojego zespołu (z dnia 28.11.2022)*. URL: https://publicystyka.ngo.pl/slack-piec-funkcji-ktore-usprawnia-prace-twojego-zespolu.

[81] Zak Doffman. *Why You Should Stop Using Facebook Messenger (z dnia 28.11.2022)*. URL: https://www.forbes.com/sites/zakdoffman/2020/07/25/why-you-should-stop-using-facebook-messenger-encryption-whatsapp-update-twitter-hack/?sh=4e342d7969ad.

Wykaz rysunków

Wykaz kodu