



POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Kacper Dalgiewicz, s28548

Adam Skorupski, s29441

„Let Me Cook”

Portal społecznościowy z przepisami kulinarnymi

Praca inżynierska napisana pod
kierunkiem:

Dr inż. Mariusza Trzaski,
prof. PJATK

Warszawa, lipiec 2026

Streszczenie

Celem niniejszej pracy inżynierskiej było utworzenie portalu społecznościowego, który umożliwia użytkownikom zamieszczanie własnych przepisów kulinarnych i dzielenie się nimi w Internecie. Serwis posiada wyróżniające go na rynku funkcjonalności, takie jak automatycznie generowana lista składników, rozbudowany edytor tekstu, zaawansowana wyszukiwarka czy inteligentna lista zakupów.

Projekt składa się z responsywnej aplikacji webowej napisanej za pomocą frameworka Next.js oraz biblioteki React, mikroserwisów wykorzystujących framework .NET i bramę API Ocelot, bazy danych Microsoft SQL Server oraz magazynu plików MinIO. Wszystkie elementy są skonteneryzowane i uruchamiane z poziomu platformy Docker.

Słowa kluczowe

Przepis kulinarny, lista zakupów, komentarze, portal społecznościowy, moderacja, administracja, aplikacja webowa, architektura mikroserwisów, API Gateway, Ocelot, C#, TypeScript, .NET 8, Entity Framework Core, MS SQL, REST API, JWT Token, Docker, MinIO, Next.js, React, Bootstrap, Postman, Tiptap.

Spis treści

1.	WSTĘP	5
1.1.	Cel pracy	5
1.2.	Organizacja pracy	5
1.3.	Zakres pracy	5
1.4.	Podział zadań.....	6
2.	ISTNIEJĄCE SERWISY KULINARNE ORAZ BLOGI.....	9
2.1.	Gotujmy.pl.....	9
2.2.	Smaker.pl.....	14
2.3.	Allrecipes.com.....	20
2.4.	Aniagotuje.pl.....	22
2.5.	Podsumowanie.....	25
3.	ANALIZA WYMAGAŃ I KONCEPCJA SYSTEMU	26
3.1.	Słownik pojęć	26
3.2.	Wymagania funkcjonalne.....	27
3.3.	Wymagania niefunkcjonalne	33
3.4.	Aktorzy systemu.....	34
3.5.	Model przypadków użycia	35
3.6.	Analityczny diagram klas	38
3.7.	Diagramy stanów.....	40
3.8.	Diagramy aktywności.....	41
4.	TECHNOLOGIE ORAZ NARZĘDZIA WYKORZYSTANE W PROJEKCIE	43
4.1.	Część serwerowa – backend	43
4.1.1	Środowisko uruchomieniowe .NET	43
4.1.2	C#.....	43
4.1.3	ASP.NET Core	44
4.1.4	Entity Framework Core	45
4.1.5	REST API.....	47
4.1.6	Ocelot.....	48
4.1.7	Serilog.....	48
4.2.	Aplikacja webowa – frontend.....	49
4.2.1	TypeScript.....	49
4.2.2	React	50
4.2.3	Next.js	52
4.2.4	Bootstrap.....	53
4.2.5	Markdown	56
4.2.6	Tiptap.....	57
4.3.	Microsoft SQL Server	57
4.4.	Docker	58
4.4.1	Obrazy Dockera	59
4.4.2	Kontenery.....	59
4.4.3	Docker Compose.....	59
4.5.	Git oraz GitHub	59
4.5.1	System kontroli wersji Git.....	60
4.5.2	Platforma GitHub	60
4.6.	Postman	60
4.7.	Figma.....	61
4.8.	MinIO.....	62
5.	ARCHITEKTURA I IMPLEMENTACJA SYSTEMU	64

5.1.	Implementacja części serwerowej – backend	65
5.1.1	Zastosowane wzorce projektowe	65
5.1.2	Architektura mikroserwisów	69
5.1.3	Komunikacja z bazą danych	75
5.1.4	Mechanizmy uwierzytelniania i autoryzacji	77
5.2.	Implementacja aplikacji webowej – frontend	79
5.2.1	Struktura projektu aplikacji webowej	79
5.2.2	Oprogramowanie pośredniczące	85
5.2.3	Opis użycia bibliotek	87
5.3.	Konteneryzacja i środowisko uruchomieniowe	101
5.3.1	Środowiska uruchomieniowe	101
5.3.2	Zmienne środowiskowe	102
6.	INTERFEJS I FUNKCJONALNOŚCI APLIKACJI	104
6.1.	Nawigacja w systemie	104
6.1.1	Główny panel nawigacyjny	104
6.1.2	Stopka	107
6.2.	Użytkownik niezalogowany	107
6.2.1	Rejestracja i logowanie do systemu	107
6.2.2	Przeglądanie przepisów	109
6.2.3	Przeglądanie profili użytkowników	117
6.3.	Użytkownik zalogowany	119
6.3.1	Zarządzanie swoimi przepisami	119
6.3.2	Zarządzanie profilem	128
6.3.3	Zarządzanie ulubionymi przepisami	130
6.3.4	Zarządzanie listą zakupów	132
6.3.5	Obserwowanie użytkowników	134
6.3.6	Ocenianie przepisów	136
6.3.7	Dodawanie i usuwanie komentarzy	138
6.3.8	Zgłaszanie komentarzy	140
6.3.9	Zmiana hasła	141
6.3.10	Wylogowanie z systemu	142
6.4.	Administracja i moderacja	142
6.4.1	Panel administracyjny	142
6.4.2	Widok profilu użytkownika	161
6.4.3	Widok przepisu	163
7.	PODSUMOWANIE	166
7.1.	Efekty pracy	166
7.2.	Napotkane trudności	166
7.3.	Dalszy rozwój	167
7.4.	Wnioski końcowe	168
8.	BIBLIOGRAFIA	169
9.	SPIS ILUSTRACJI	173
10.	SPIS TABEL	177
11.	SPIS LISTINGÓW	178

1. Wstęp

Powszechny i łatwy dostęp do Internetu sprawił, że wyszukiwanie informacji jest prostsze i szybsze niż kiedykolwiek. Te bogate zasoby pomagają nam w różnych obszarach codziennego życia. Jednym z nich jest gotowanie i przygotowywanie posiłków. Coraz częściej sięgamy po smartfon niż po książkę kucharską w poszukiwaniu przepisów kulinarnych. W sieci, oprócz typowych blogów, na popularności zyskują przepisy umieszczane w mediach społecznościowych, często opatrzone zdjęciami lub krótkim materiałem wideo.

1.1. Cel pracy

Celem niniejszej pracy było stworzenie platformy społecznościowej z przepisami kulinarnymi. Pozwala ona użytkownikom na dodawanie własnych przepisów i udostępnianie ich wszystkim odwiedzającym bez konieczności zakładania własnej strony internetowej. Dzięki uporządkowaniu treści dodanych przez twórców do jednej bazy aplikacja daje możliwość ich zaawansowanego wyszukiwania i filtrowania. Cechy społecznościowe portalu umożliwiają ocenianie oraz zapisywanie przepisów i obserwowanie ich twórców.

Projekt został zrealizowany w ramach pracy inżynierskiej realizowanej na specjalizacji Inżynieria Oprogramowania i Baz Danych w Polsko-Japońskiej Akademii Technik Komputerowych.

1.2. Organizacja pracy

Niniejsza praca opisuje proces tworzenia aplikacji i stanowi jej dokumentację. W rozdziale drugim zaprezentowano istniejące na rynku serwisy i blogi i porównano z proponowanym rozwiązaniem. Rozdział trzeci dokumentuje wymagania aplikacji, aktorów systemu, przypadki użycia oraz diagramy stanów i aktywności.

Następne rozdziały są poświęcone implementacji systemu. Rozdział czwarty opisuje technologie i narzędzia użyte w projekcie. Rozdział piąty opisuje szczegóły implementacyjne i architekturę systemu. W rozdziale szóstym pokazano, jak wygląda interfejs aplikacji i jej funkcjonalności dla poszczególnych aktorów systemu. Rozdział siódmy to podsumowanie całego procesu twórczego i wyciągnięte wnioski.

1.3. Zakres pracy

Praca była tworzona w zespole dwuosobowym, w którym wydzielono dwie główne role:

1. Programista aplikacji serwerowej (*backend*) – odpowiedzialny za architekturę bazy danych, mikroservisów, analizę techniczną i logikę biznesową.
2. Programista aplikacji klienckiej (*frontend*) – odpowiedzialny za projekt i implementację interfejsu użytkownika, integrację systemu oraz logikę prezentacji.

Wspólnym zakresem prac objęto analizę wymagań, dokumentację pracy oraz testowanie rozwiązania.

Prace nad projektem zostały podzielone na poszczególne zadania, które zostały umieszczone w aplikacji Notion. Każde zadanie posiadało nazwę, status, osobę przypisaną, priorytet, obszar oraz zajęcia, na których zespół chce pokazać jego rezultat. Śledzenie postępu prac było możliwe dzięki fazom i statusom:

- Faza planowania:
 - W kolejce,
 - Przyszłe zajęcia,
 - Do zrobienia teraz.
- Faza tworzenia i oceny:
 - W trakcie,
 - Recenzja.
- Faza kończąca:
 - Zrobione,
 - Niezrobione,
 - Pokazane.

Zespół wspólnie planował zadania co zajęcia. Rysunek 1 przedstawia widok zaplanowanych zadań w Notion.



Zadania

Moje

Do zrobienia teraz

Na przyszłe zajęcia

Kolejka

Zrobione

Wszystko

Status

2 rules

Status: Not Pokazane, W ko...

Filter

Status	Nº ID	Aa Name	Assignee	Na zajęcia	Priority	Obszar	
Do zrobienia teraz	COOK-112	Navbar	Adam	PRO2 Zajęcia 3	Medium	frontend	
W trakcie	COOK-108	Lista zakupów	Kacper Adam	PRO2 Zajęcia 3	Medium	backend	
Recenzja	COOK-137	Grupy kategorii	Kacper Adam	PRO2 Zajęcia 3	High	backend frontend	
Recenzja	COOK-127	Dodanie filtra umożliwiającego zwracanie przepisów wyłącznie od obserwowanych użytkowników	Kacper Adam	PRO2 Zajęcia 3	Medium	backend frontend	
Zrobione	COOK-143	Dodanie bio do profilu	Kacper Adam	PRO2 Zajęcia 3	Low		
Zrobione	COOK-139	Zmiana architektury bazy danych na monolit	Kacper	PRO2 Zajęcia 3	High	backend	

+ New task

Rysunek 1 Widok zaplanowanych zadań w Notion. Źródło: Opracowanie własne.

Każda zmiana w repozytorium projektu była dokładnie opisywana oraz recenzowana przez drugiego członka zespołu za pomocą mechanizmu *Pull Requests* w serwisie GitHub. Po akceptacji, nowa funkcjonalność była prezentowana promotorowi na najbliższych zajęciach.

1.4. Podział zadań

W tabeli 1 przedstawiono wykaz szczegółowy zadań i informacje o osobach odpowiedzialnych za każde zadanie z podziałem na fazy.

Tabela 1 Podział i przydział zadań pogrupowany według faz projektu. Źródło: Opracowanie własne.

Faza analizy	
Diagram przypadków użycia	Adam Skorupski
Diagram klas analityczny	Kacper Dalgiewicz
Stan sztuki	Adam Skorupski
Słownik	Kacper Dalgiewicz
Koncepcja części serwerowej i bazodanowej	Kacper Dalgiewicz
Koncepcja części aplikacji webowej	Adam Skorupski
Diagramy aktywności	Adam Skorupski
Konfiguracja projektu w Notion	Adam Skorupski

Faza projektowania	
Diagram klas projektowy	Kacper Dalgiewicz
Diagram architektury aplikacji	Kacper Dalgiewicz Adam Skorupski
Diagramy stanu	Adam Skorupski
Projektowanie ekranów aplikacji webowej	Adam Skorupski
Logo aplikacji	Kacper Dalgiewicz

Faza implementacji	
Konfiguracja serwera uczelnianego	Adam Skorupski
Implementacja backend	Kacper Dalgiewicz
Konfiguracja środowiska Docker	Kacper Dalgiewicz
Implementacja frontend	Adam Skorupski

Faza testów	
Testowanie backend	Adam Skorupski
Testowanie frontend	Kacper Dalgiewicz

Faza dokumentacji	
Rozdział 1.	Adam Skorupski
Rozdział 2.	Adam Skorupski
Rozdział 3.: wstęp, podrozdziały 3.1. – 3.5., 3.7 – 3.8	Adam Skorupski
Rozdział 3.: podrozdział 3.6.	Kacper Dalgiewicz
Rozdział 4.: wstęp, podrozdział 4.1.	Kacper Dalgiewicz

Rozdział 4.: podrozdział 4.2.	Adam Skorupski
Rozdział 4.: podrozdziały 4.3. – 4.8.	Kacper Dalgiewicz
Rozdział 5.: wstęp, podrozdział 5.1.	Kacper Dalgiewicz
Rozdział 5.: podrozdział 5.2.	Adam Skorupski
Rozdział 6.: wstęp, podrozdziały 6.1. – 6.3.	Kacper Dalgiewicz
Rozdział 6.: podrozdział 6.4.	Adam Skorupski
Rozdział 7.	Adam Skorupski

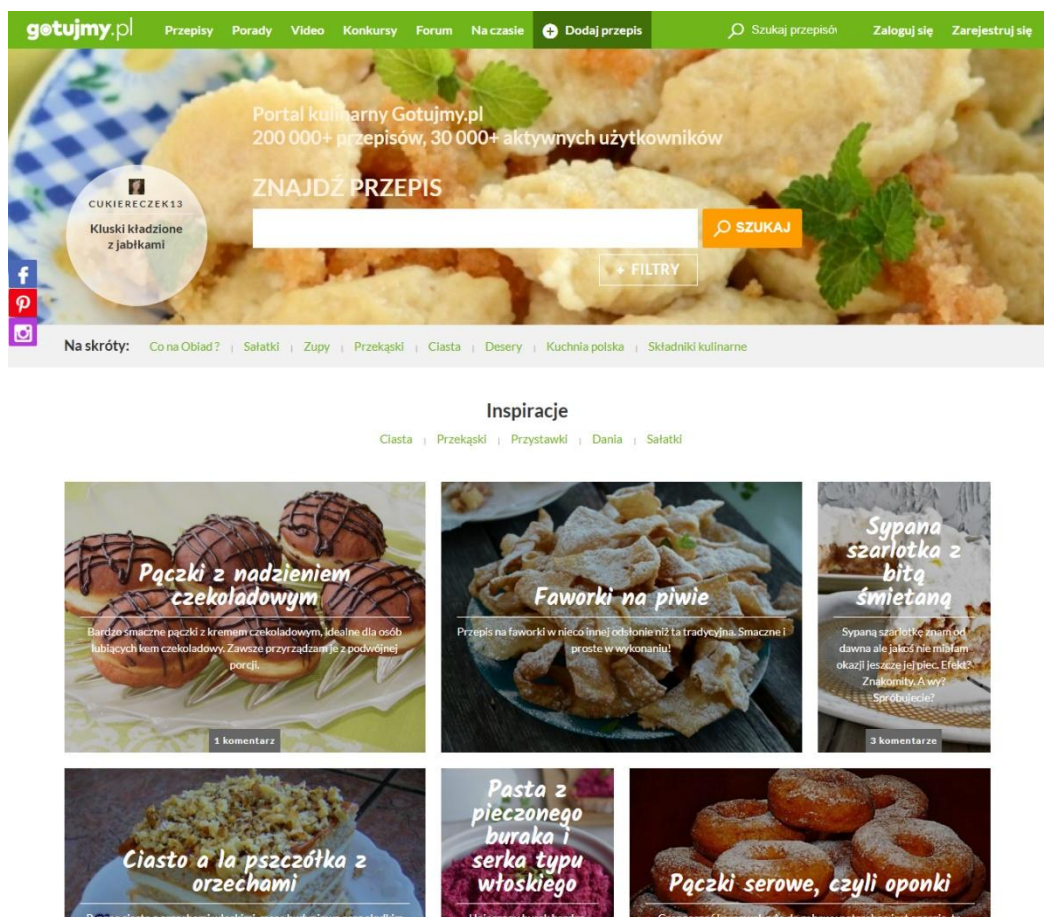
2. Istniejące serwisy kulinarne oraz blogi

W Polsce i na świecie dostępnych jest wiele podobnych platform oraz prywatnych blogów do publikowania własnych przepisów kulinarnych. Przed przystąpieniem do projektowania i implementacji platformy, autorzy przeprowadzili analizę istniejących konkurencyjnych rozwiązań.

Celem analizy było odnalezienie mocnych i słabych stron każdego serwisu. Następnie te rezultaty zostały wykorzystane do sporządzenia wymagań funkcjonalnych docelowego rozwiązania autorów.

2.1. Gotujmy.pl

Gotujmy.pl to portal kulinarny należący do firmy mediowej Burda Media Group. Według [1] serwis posiada ponad 200 tysięcy przepisów oraz ponad 30 tysięcy aktywnych użytkowników. Oprócz bazy przepisów Gotujmy.pl udostępnia artykuły kulinarne oraz forum dla użytkowników serwisu. Rysunek 2 przedstawia stronę główną portalu, a Rysunek 3 forum kulinarne.



Rysunek 2 Strona główna portalu Gotujmy.pl. Źródło: [1]

gotujmy.pl

Przepisy Porady Video Konkursy Forum

Na czasie + Dodaj przepis Szukaj przepisów

1 adamskorupski

Strona główna Forum

Szukaj wątków

Dziś

Przywitaj się

Przepisy i potrawy

Porady kulinarne

Galerie potraw

Imprezy kulinarne i konkursy

Domowe alkohole

Groch z kapustą

Działka i uprawy

Wyposażenie kuchni

Kuchnie świata

Restauracje i inne lokale - opinie

Savoir-Vivre

Media o kulinariach

Forum Gotujących (Super)Rodziców

Przepisy kulinarne Gotujmy.pl - komentarze

Pomoc kuchenna

Katalog forów

Dieta i odżywianie

Wątki: 152 | Posty: 15871 | Ostatni wpis: 30 kwi 2026 12:28

Jeśli nie jesz, a "odżywasz się", tu znajdziesz coś dla siebie.

Catering Firmowy :) (5)

Młody jęczmień (4)

Pyszne jedzonko (3)

aga.anna.kownacka 30 kwi 2026 12:28

aga.anna.kownacka 04 kwi 2026 08:31

aga.anna.kownacka 01 mar 2026 16:39

Zobacz wszystkie

Przywitaj się

Wątki: 465 | Posty: 8670 | Ostatni wpis: 20 lut 2026 13:38

Napisz kim jesteś i co nam dzisiaj ugotujesz. Podziel się z nami swoimi ulubionymi przepisami. Pochwal się swoim popisowym daniem.

Cześć (0)

Dodatkowa praca (0)

Dodatkowa praca (0)

anna_s

elazientek42

elazientek42

Zobacz wszystkie

Przepisy i potrawy

Wątki: 251 | Posty: 3846 | Ostatni wpis: 16 kwi 2026 14:34

Czasami dobrze jest podyskutować, żeby znowu zgłodnieć :)

Przepis na potrawy z warzywami dla dzieci (11)

Cała grządka bazylii... co z nią zrobić? (31)

Co można zrobić z ciecierzycy? (7)

Joanna0 16 kwi 2026 14:34

Joanna0 16 kwi 2026 14:29

Joanna0 16 kwi 2026 14:22

Zobacz wszystkie

Porady kulinarne

Wątki: 267 | Posty: 2961 | Ostatni wpis: 30 kwi 2026 15:43

Robisz coś pierwszy raz? Przerażenie w oczach? Zapytaj, a nie zostanie to bez odpowiedzi.

Zasmażana kapusta (3)

Pampuchy (1)

Suszenie grzybów (5)

Joanna0 30 kwi 2026 15:43

Joanna0 30 kwi 2026 15:41

Joanna0 30 kwi 2026 15:36

PORADY

7 sprawdzonych przepisów na zupę cebulową

★★★★★

7 pomysłów na dania z papryką

★★★★★

Pomysły na obiad - kasza jaglana

★★★★★

PORADY

Pomysły na sycące zapiekanki

★★★★★

Jak zrobić placki po węgiersku?

★★★★★

Pomysły na dania ze szpinakiem

★★★★★

PORADY

Chili con carne - z czym podawać?

★★★★★

Kulinarne pomysły na łososa

★★★★★

Omlet - obiad w 10 minut

★★★★★

Rysunek 3 Forum serwisu Gotujmy.pl. Źródło: [2]

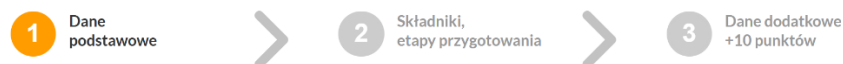
Serwis umożliwia zalogowanym użytkownikom na napisanie i dodanie własnego przepisu. Proces jest podzielony na 3 sekcje:

1. Dane podstawowe,
2. Składniki, etapy przygotowania,
3. Dane dodatkowe.

Rysunek 4 przedstawia wycinek strony kreatora przepisu.

10

Dodaj przepis



Nazwa potrawy: *

0 znaków (12 min, 50 max)

Krótki opis: *

0 znaków (100 min, 250 max)

Sposób przygotowania (Sposób przygotowania możesz przedstawić też w etapach - krok 2):

0 znaków (300 min)

Zdjęcia:

Zdjęcie jest opisane, jeśli jego podpis zawiera minimum 40 znaków wraz ze spacjami



Dodaj zdjęcie

Kategorie: *

Wybierz 1

Wybierz 2

Wybierz 3

Czas przygotowania: *

- ☐ do 30 minut
☐ od 30 do 60 minut
☐ powyżej 60 minut

Poziom trudności: *

- ☐ łatwe
☐ średnie
☐ trudne

Dla ilu porcji:

wartość liczbowa, np: 3

Liczba kalorii:

wartość liczbowa, np: 300

Dalej >

Anuluj

Rysunek 4 Kreator przepisu portalu Gotujmy.pl – sekcja Dane podstawowe. Źródło: [1]

Na stronie przepisu, poza jego przeczytaniem, możemy dany przepis:

- dodać do listy polubionych,
- ocenić,
- przesłać w wiadomości e-mail,
- wydrukować,
- skomentować.

Rysunek 5 przedstawia przykładowy fragment przepisu zamieszczonego na portalu Gotujmy.pl [3].

gotujmy.pl Przepisy Porady Video Więcej + Dodaj przepis Szukaj przepisów 1 adamskorupski

Strona główna > Przepisy > Ciasta > Torcik czekoladowy

Torcik czekoladowy

29 paź 2018 20:15

dorota w

TRUDNOŚĆ: średnie | CZAS: powyżej 60 min | PORCJE: 16

★★★★★
ocena: 5/5 głosów: 1

Torcik bardzo czekoladowy. Prosty w wykonaniu, robi się go dość szybko a potem zostaje tylko delektowanie przy kominku lub pod ciepłym kocem.

Składniki:

Torcik czekoladowy

biskopt

- 6 dużych jajek
- 3/4 szklanki cukru
- 1 płaska łyżka proszku do pieczenia
- 1 szklanka mąki pszennej
- 1/3 szklanki kakao
- 1 łyżka oleju
- 150 ml wody do nasączenia

krem

- 250 g mascarpone
- 200 g czekolady
- 250 ml kremówki
- 2 łyżki cukru

Sposób przygotowania:

Torcik czekoladowy

Białka ubić na sztywną pianę z cukrem i proszkiem do pieczenia. Żółtka krótko ubić z olejem, wlać do piany z białek, wymieszać łyżką. Mąkę wymieszać z kakao, wsypać do jajek, jeszcze raz delikatnie wymieszać łyżką. Wyłożyć do tortownicy o średnicy 21 cm, wyrównać. Wstawić do piekarnika nagrzanego do 170C, piec ok. 35 min. Ostudzić (można upiec dzień wcześniej), lekko ściąć sam czubek ciasta. Pozostały biskopt przekroić wzdłuż na 3 równe części.

150 g czekolady rozpuścić na parze, ostudzić do temperatury pokojowej. Resztę czekolady zetrzeć na tarce o dużych oczkach.

Mascarpone, śmietankę kremówkę i cukier ubić krótko na sztywną masę. Dodać rozpuszczoną i startą czekoladę, jeszcze raz krótko zmiksować. Masę podzielić na 4 części.

Biskoptowy spód nasączyć 1/3 wody (50 ml), nałożyć 1/4 części masy, rozsmarować, powtórzyć z pozostałymi warstwami. Ostatnią część masy rozsmarować na bokach tortu.

Ścięty czubek ciasta pokruszyć na drobno, obsypać boki i część wierzchu tortu. Wstawić do lodówki do schłodzenia.

Zobacz film: Tort czekoladowy

Rysunek 5 Wycinek przepisu zamieszczonego na portalu Gotujmy.pl. Źródło: [3]

Jeżeli serwis wykryje na liście składnik, który istnieje w słowniku kulinarnym portalu, to zostaje on opatrzony hiperłączem do strony opisującej dany produkt. Rysunek 6 przedstawia przykładową stronę w słowniku kulinarnym.

gotujmy.pl
Przepisy
Porady
Video
Więcej
Dodaj przepis
Szukaj przepisów
1
adamskorupski

Strona główna
Słowniki Kulinarne
Jajko

Słownik kulinarny

J

Jajko



Opis: jajka są jednym z najbardziej popularnych produktów kulinarnych. Używane w wypiekach cukierniczych, piekarskich, stosowane do przyrządzenia musów, mas, kremów, klusek. Najpopularniejszym daniem z białka jaja jest beza, natomiast z żółtka kogiel-mogiel. Jajka, często podawane są ugotowane na twardo lub miękko.

Rodzaje jaj:

- Jaja gęsie
- Jaja indyjskie
- Jaja kaczce
- Jaja kurze
- Jaja przepiórcze
- Jaja w proszku
- Jajka perlicze

[← WRÓĆ DO LISTY](#)

Jajko - zobacz przepisy zawierające ten składnik

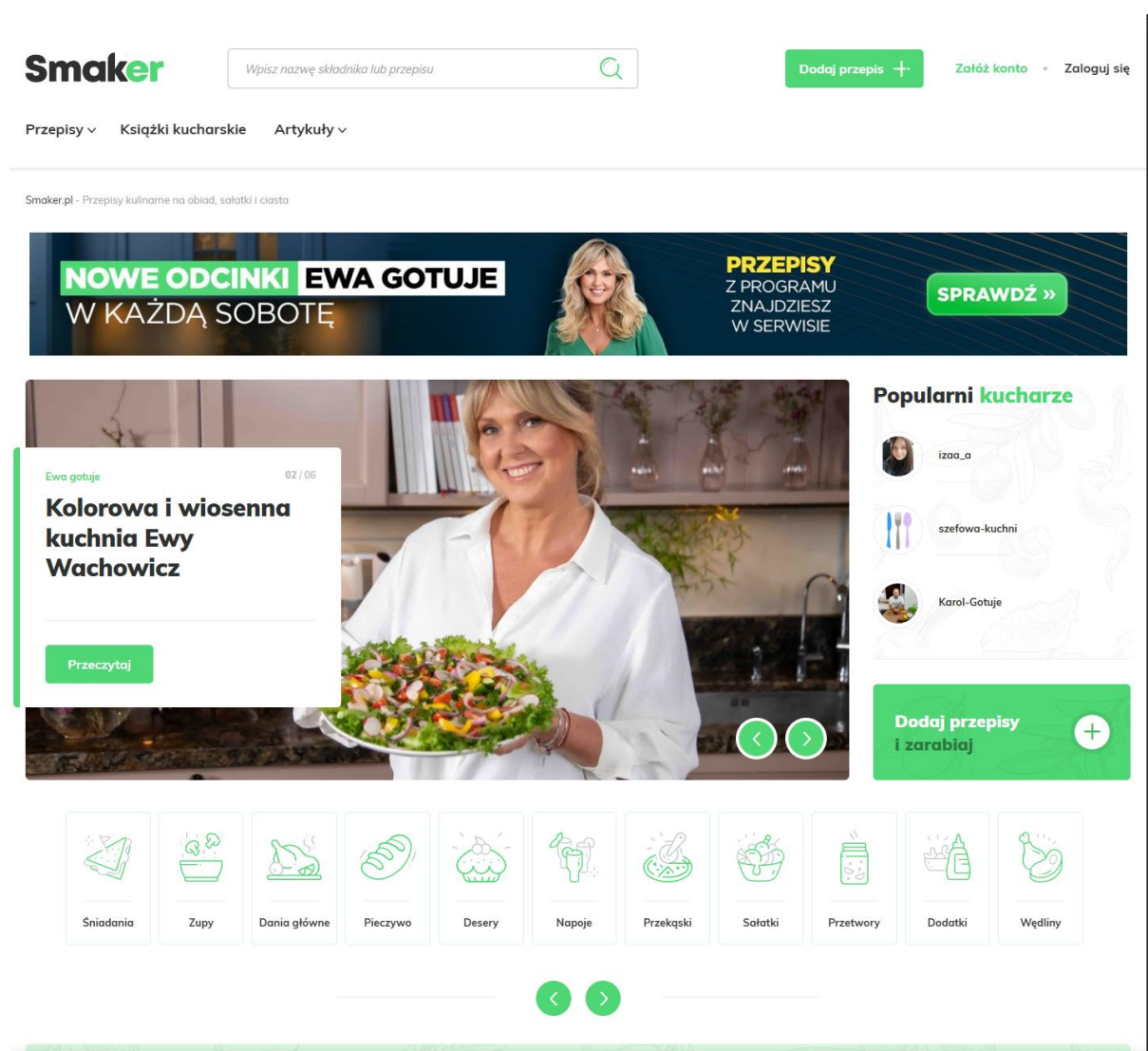




Rysunek 6 Składnik "Jajko" w słowniku kulinarnym Gotujmy.pl. Źródło: [4]

2.2. Smaker.pl

Smaker.pl jest serwisem kulinarnym należącym do Grupy Interia. Platforma posiada 100 tysięcy zarejestrowanych kont, a na jej łamach można przeczytać ponad 100 tysięcy przepisów i 9 tysięcy artykułów [5]. Portal daje możliwość autorom przepisów zarabiania pieniędzy z reklam Google AdSense, co czyni Smaker.pl wysoce atrakcyjnym serwisem dla twórców. Rysunek 7 prezentuje stronę główną witryny.



Rysunek 7 Strona główna serwisu Smaker.pl Źródło: [5]

Kreator przepisu jest podzielony na 4 etapy:

1. Podstawowe informacje,
2. Składniki,
3. Sposób przygotowania,
4. Dodatkowe informacje.

Powyższe etapy są przedstawione na kolejno Rysunek 8, Rysunek 9, Rysunek 10 oraz Rysunek 11.

Dodaj przepis



1. Podstawowe informacje

2. Składniki

3. Sposób przygotowania

4. Dodatkowe informacje

Podgląd miniatury i zdjęcia głównego



adamskorupski

Śniadania

Tytuł

Tytuł

Tytuł

Pozostało znaków: 195

Opis

Opis

Kategoria

Śniadania

Czas przygotowania

do 15 min

Czas gotowania

do 15 min

Danie przygotowane dla

1-2 osoby

Dalej

Rysunek 8 Dodawanie przepisu w Smaker.pl - sekcja Podstawowe informacje. Źródło: [5]

Dodaj przepis



1. Podstawowe informacje

2. Składniki

3. Sposób przygotowania

4. Dodatkowe informacje

Grupa 1 (nieobowiązkowe)

Wpisz nazwę grupy

Możesz utworzyć grupę składników np. na ciasto, na masę lub dodać wszystkie składniki w jednej liście.

Składniki

Wpisz nazwę składnika

Wpisz nazwę składnika

Wpisz nazwę składnika

+ Dodaj nowy składnik

+ Dodaj nową grupę

Wstecz

Dalej

Rysunek 9 Dodawanie przepisu w Smaker.pl - sekcja Składniki. Źródło: [5]

Dodaj przepis

1. Podstawowe informacje
2. Składniki
3. Sposób przygotowania
4. Dodatkowe informacje

Etap 1


Dodaj zdjęcie
JPG, PNG, GIF / maks. 10 MB

Sposób przygotowania

Opisz etap przygotowania

Dodaj kolejny etap

Wstecz

Dalej

Rysunek 10 Dodawanie przepisu w Smaker.pl - sekcja Sposób przygotowania. Źródło: [5]

Dodaj przepis

1. Podstawowe informacje
2. Składniki
3. Sposób przygotowania
4. Dodatkowe informacje

To już ostatni krok!
Dopasowaliśmy Twój przepis zgodnie z podanymi przez Ciebie informacjami.

Możesz je edytować, dodać lub zaakceptować i opublikować przepis.

Jakich przyborów kuchennych potrzeba do sporządzenia tego przepisu?

Prowadzisz własnego bloga lub vloga?

Okazje

☐ Wielkanoc
☐ Grill
☐ Halloween

☐ Boże Narodzenie
☐ Tłusty czwartek
☐ Komunia

☐ Impreza
☐ Walentynki
☐ Do pracy

Kuchnie świata

☐ Amerykańska
☐ Polska
☐ Indyjska
☐ Grecka
☐ Węgierska
☐ Żydowska

☐ Azjatycka
☐ Włoska
☐ Francuska
☐ Bałkańska
☐ Śródziemnomorska

☐ Czeska
☐ Meksykańska
☐ Chińska
☐ Tajska
☐ Ukraińska

Dieta

☐ Bez glutenu
☐ Dla dzieci
☐ Wegańskie

☐ Bez laktozy
☐ Dietetyczne
☐ Dla zdrowia

☐ Bez cukru
☐ Wegetariańskie
☐ Keto

Potrzebne przybory kuchenne

Wpisz przybory kuchenne oddzielając je przecinkiem np. blocha 30x40cm, mikser

Link do bloga

Wklej link

Wstecz

Publikuj

Rysunek 11 Dodawanie przepisu w Smaker.pl - sekcja Dodatkowe informacje. Źródło: [5]

16

Na stronie przepisu użytkownicy mogą skorzystać z wielu akcji, takich jak:

- Ocenienie,
- Dodanie do książki kucharskiej,
- Wysłanie za pomocą wiadomości e-mail,
- Drukowanie,
- Zaplanowanie gotowania na dany dzień,
- Wysłanie, skopiowanie lub wydrukowanie listy składników,
- Wyłączenie wygaszania ekranu,
- Skomentowanie,
- Udostępnienie w mediach społecznościowych,
- Dodanie swojego zdjęcia sporządzonego dania.

Rysunek 12 oraz Rysunek 13 przedstawiają widok przykładowego przepisu w serwisie Smaker [6].


Super
5

Dodaj do książki
4

Wyslij
21

Drukuj
21

Nowość
Zaplanuj
2


Zupy • Przepis na

Odsłon: 9889 • 07.04.2026

Zupa Tom Kha

★★★★★ 5.0 (4 oceny)



do 30 min



> 60 min



3-4 osoby

smaker



Azjatycka

Tajska

Buliony

Warzywa

Owoce

Więcej ▾

Tom Kha to aromatyczna tajska zupa, której bazą jest mleczko kokosowe i bulion. Jest delikatnie kremowa i sycząca. Charakteryzuje się słodko-kwaśno-pikantnym smakiem.

Rysunek 12 Widok przepisu na portalu Smaker - część główna. Źródło: [6]

Składniki

- 400 ml mleczka kokosowego
- 500 ml wody i pół kostki rosołowej lub 500 ml bulionu
- 300 g piersi z kurczaka
- 4 pieczarki
- pół pomidora lub 5 pomidorków koktajlowych
- pół czerwonej papryki
- 1 łyżka pasty Tom Kha
- 2 łyżki oleju rzepakowego
- 2 łyżki sosu rybnego
- 4 liście limonki kaffir (świeże lub suszone)
- 2 łyżki soku z limonki
- 1 ząbek czosnku
- 1 czerwona papryczka chili
- 2 cm korzenia imbiru
- 1 szalotka
- 3 cm łodygi trawy cytrynowej
- 1 łyżeczka brązowego cukru

Wyślij składniki

Etapy przygotowania

☐ Nie wygaszaj ekranu

- 1 Czosnek przepuść przez praskę, a imbir i szalotkę posiekaj drobno. Kurczaka pokrój w kostkę. Pieczarki obierz pokrój w plasterki, a paprykę pozbaw nasion i także pokrój w kostkę.
- 2 Na dnie dużego garnka rozgrzej olej. Wrzuć czosnek, szalotkę i imbir. Smaż przez chwilę, a następnie dodaj mięso i obsmaż je z każdej strony. Dorzuć pieczarki i paprykę. Całość smażyć przez około 6 minut.
- 3 Do garnka wlej mleczko kokosowe oraz wodę z kostką rosołową lub bulion. Dodaj liście limonki kaffir, sok z limonki, cukier, sos rybny, drobno pokrojoną papryczkę chili, trawę cytrynową oraz pastę Tom Kha. Wymieszaj i gotuj na małym ogniu pod przykryciem przez około 20 minut.
- 4 Pomidora sparz, obierz i pokrój w kostkę (pomidorki koktajlowe możesz po prostu przekroić na pół).
- 5 Po 20 minutach dodaj pomidory do zupy, przykryj i gotuj jeszcze około 10 minut.
- 6 Na koniec spróbuj zupy – jeśli będzie za mało kwaśna, dodaj więcej soku z limonki. W razie potrzeby dopraw do smaku solą lub odrobiną sosu sojowego.
- 7 Podawaj posypaną świeżą kolendrą. Zupa świetnie smakuje z ryżem lub makaronem.



Udostępnij



Udostępnij

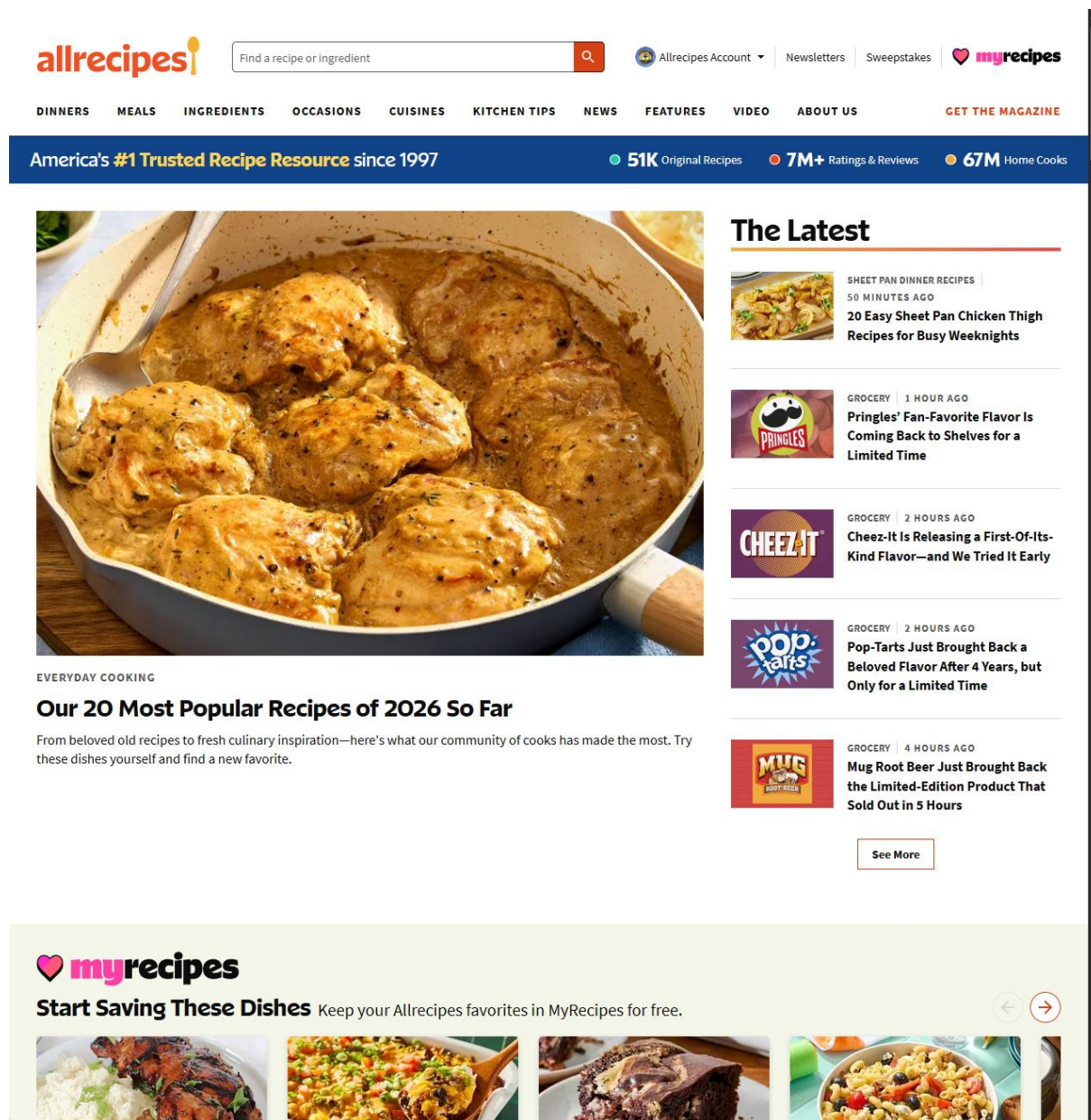


Obserwuj nas
w Google News

Rysunek 13 Widok przepisu na portalu Smaker - składniki i etapy przygotowania. Źródło: [6]

2.3. Allrecipes.com

Jednym z najpopularniejszych zagranicznych serwisów kulinarnych jest Allrecipes.com. Portal powstał w 1997 roku i od tego momentu zebrał ponad 51 tysięcy przepisów i aż 67 milionów użytkowników [7]. Według [8] w ciągu miesiąca serwis notuje nawet 225 milionów wyświetleń. Na Rysunek 14 przedstawiono stronę główną witryny.



Rysunek 14 Strona główna serwisu Allrecipes.com. Źródło: [7]

Na stronie przepisu użytkownik może użyć poniższych funkcjonalności:

- Ocenienie,
- Zapisanie do ulubionych,
- Wydrukowanie,

- Lista makroskładników,
- Udostępnienie w mediach społecznościowych,
- Wyłączenie wygaszania ekranu,
- Pomnożenie lub podzielenie ilości składników,
- Dodanie recenzji,
- Zamieszczenie pytania do przepisu.

Rysunek 15 przedstawia przykładowy przepis zmieszczony w serwisie.

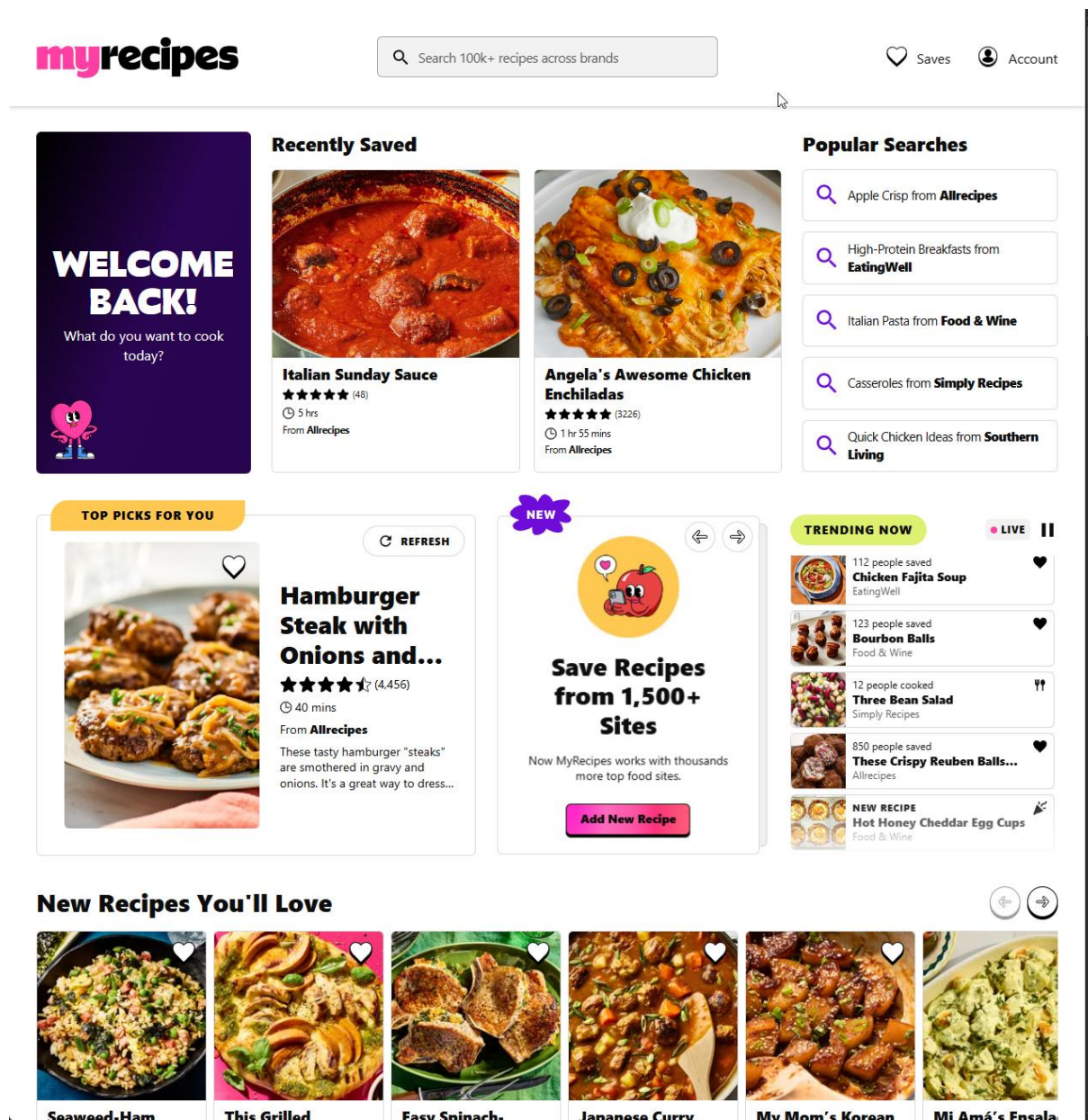
The screenshot displays the Allrecipes website interface for the 'Air Fryer Sesame Chicken Thighs' recipe. At the top, the navigation bar includes the Allrecipes logo, a search bar, and user account options. The breadcrumb trail indicates the recipe is under 'MEAT AND POULTRY > CHICKEN > CHICKEN THIGH RECIPES'. The recipe title is prominently displayed, followed by a 4.2-star rating from 23 reviews and 13 reviews. A brief description states: 'Sesame-flavored chicken thighs are crispy on the outside, yet tender and juicy on the inside.' The recipe is attributed to 'Soup Loving Nicole' and was updated on July 30, 2024. Below the description is a row of interactive buttons: 'SAVE', 'RATE', 'PRINT', and social media sharing icons. A video player shows a person pouring oil into a bowl, with a subtitle: 'Start by mixing together sesame oil, soy sauce, honey,'. Below the video is a row of five small images showing the final dish. A table provides timing and serving details:

Prep Time: 5 mins	Cook Time: 15 mins	Additional Time: 35 mins
Total Time: 55 mins	Servings: 4	Yield: 4 servings

Below the table is a link to 'Jump to Nutrition Facts'. A toggle switch for 'Keep Screen Awake' is visible. The 'Ingredients' section shows a multiplier set to '1X' (with options for 1/2X and 2X) and lists the first two ingredients: '2 tablespoons sesame oil' and '2 tablespoons soy sauce'.

Rysunek 15 Widok przepisu w serwisie Allrecipes. Źródło: [9]

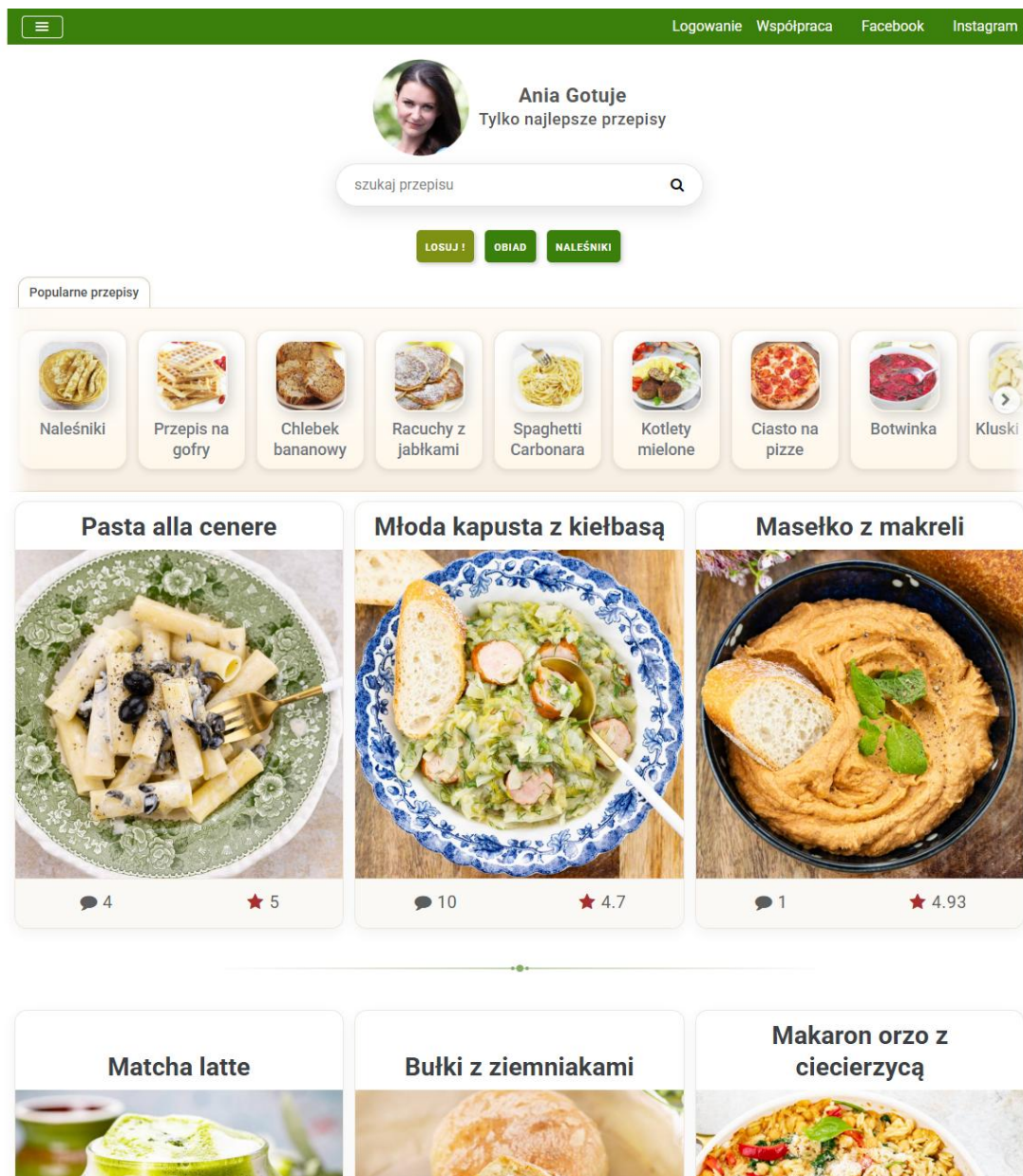
Allrecipes.com należy do wydawnictwa People Inc., które posiada w portfolio również inne kulinarne serwisy. Wydawca łączy je razem w jeden ekosystem, pozwalając użytkownikom na zapisywanie ulubionych przepisów na wspólnej liście korzystając z konta MyRecipes [10]. Rysunek 16 prezentuje widok zalogowanego użytkownika na platformie.



Rysunek 16 Widok zalogowanego użytkownika w aplikacji MyRecipes. Źródło: [10]

2.4. Aniagotuje.pl

Aniagotuje.pl to jeden z najpopularniejszych blogów kulinarnych w Polsce. Według badania [11] w grudniu 2025 roku serwis odwiedziło nawet 7,8 miliona użytkowników, co uplasowało go wśród pierwszej dwudziestki najpopularniejszych stron w Polsce. Rysunek 17 pokazuje stronę główną bloga.



Rysunek 17 Strona główna bloga Aniagotuje.pl. Źródło: [12]

Serwis posiada wiele ciekawych funkcjonalności i małych detali, które są cenione przez użytkowników, takie jak:

- Funkcja niewygaszania ekranu,
- Podane makroskładniki,
- Odznaczanie składników z listy,
- Wydrukowanie przepisu,
- Udostępnienie przepisu lub listy składników,
- Ukrycie zdjęć,
- Zapisanie do ulubionych,

- Sekcja najczęściej zadawanych pytań,
- Informacja o pojemności szklanki.

Na Rysunek 18 znajduje się wycinek jednego z przepisów.

Drukuj
Udostępnij
Komentarze

Przepisy » ciasta i torty » Babka pistacjowa

Babka pistacjowa

Pyszna, puszysta, bardzo aromatyczna, o intensywnym smaku pistacji. Taka jest właśnie ta **babka pistacjowa**, idealna na Wielkanoc i nie tylko. Gorąco polecam mój przepis.

Czas przygotowania: 25 minut

Czas pieczenia: 55 minut

Liczba porcji: forma 23 cm, poj. do 2 L

W 100 g babki:

Wartość energetyczna	462 kcal
Węglowodany	48 g
- W tym cukry	27 g
Białko	9 g
Tłuszcze	26 g

Dieta: wegetariańska

Składniki na babkę [więcej]

☐ mąka pszenna

Używam szklanki o pojemności 250 ml, ale dla dokładności polecam korzystanie z wagi kuchennej. Jajka wyjmij wcześniej z lodówki.

☐ krem pistacjowy
 250 g

☐ duże jajka
 4 sztuki - min. 260 g po rozbiciu

☐ cukier drobny
 150 g

☐ olej roślinny
 130 ml

☐ skórka otarta z limonki
 z minimum 1 sztuki

☐ proszek do pieczenia i soda oczyszczona
 po 1 płaskiej łyżeczce

☐ sól - nie pomijaj
 1/4 płaskiej łyżeczki

Składniki na dekorację

☐ krem pistacjowy - jako polewa
 dowolna ilość

☐ siekane, solone pistacje - do posypania
 dowolna ilość

☐ jadalne, suszone płatki róż - do posypania
 dowolna ilość

Składniki:

Kopiuji

Udostępnij

Nie wygaszaj ekranu

Ukryj zdjęcia

Idź do

Babka pistacjowa

Rysunek 18 Wycinek widoku przepisu na blogu Aniagotuje.pl. Źródło: [13]

24

2.5. Podsumowanie

Wszystkie wyżej wymienione serwisy gromadzą dużą liczbę przepisów kulinarnych, z czego Gotujmy.pl oraz Smaker.pl pozwalają użytkownikom na dodanie własnego przepisu. Każdy z tych serwisów posiada rozwinięte funkcje społecznościowe, takie jak ocenianie, komentowanie, udostępnianie, zapisanie czy wydrukowanie przepisu.

Największą wadą tych portali są ubogie możliwości wyszukiwania i filtrowania przepisów. Sprowadzają się one do podstawowego wyszukiwania tekstowego na podstawie tytułu lub kategorii przepisu. Brakuje ustrukturyzowania składników jako osobnych encji. Uniemożliwia to wdrożenie zaawansowanych funkcji, takich jak filtrowanie według produktów oraz agregowanie składników na liście zakupów użytkownika.

Portale, które opierają się na treściach swoich użytkowników, dysponują ubogimi edytorami tekstu. Ograniczają się one głównie do pisaniu tekstu nieformatowanego i dodawaniu zdjęć, co nie pozwala twórcom na zaprezentowanie swojego przepisu wedle własnego pomysłu i stylu.

Zwrócono również uwagę na to, że większość omówionych serwisów należy do dużych firm mediowych, co skutkuje agresywną monetyzacją. Nadmiar inwazyjnych reklam, banerów i automatycznie odtwarzanych niepowiązanych materiałów wideo utrudnia poruszanie się po serwisie i pogarsza czytelność treści.

Zidentyfikowanie mocnych i słabych stron konkurencyjnych portali pozwoliło autorom na zaprojektowanie oraz dostarczenie rozwiązania, które wyróżnia się dobrze znanymi, ale i nowatorskimi funkcjonalnościami. Proponowana aplikacja pozwala twórcom na umieszczanie własnych przepisów za pomocą edytora z szerokimi możliwościami formatowania tekstu oraz systemem mapowania składników z bazy produktów. W rezultacie użytkownicy zyskują precyzyjne filtrowanie oraz inteligentną listę zakupów.

3. Analiza wymagań i koncepcja systemu

W niniejszym rozdziale przedstawiono proces analizy i założenia projektowe do wdrożenia portalu Let Me Cook. Opisano wymagania funkcjonalne i нефункционалне, aktorów systemu z modelem przypadków użycia oraz diagramy stanów i aktywności.

Celem portalu jest umożliwienie twórcom publikowania treści za pomocą intuicyjnej aplikacji webowej oraz dostarczenie użytkownikom przejrzystego, czytelnego, a zarazem wysoce funkcjonalnego interfejsu do wyszukiwania i przeglądania przepisów. Mając na uwadze znaczący wpływ jakości treści dodanych przez użytkowników na poziom merytoryczny serwisu, aplikacja dostarcza rozbudowane możliwości moderacji przeznaczone dla administratorów.

3.1. Słownik pojęć

Tabela 2 przedstawia słownik pojęć użytych w następnych podrozdziałach.

Tabela 2 Słownik pojęć. Źródło: Opracowanie własne.

Nazwa pojęcia	Definicja
Administrator	Użytkownik serwisu posiadający pełne uprawnienia do zarządzania serwisem oraz kontami innych użytkowników oraz moderatorów, z wyłączeniem SuperAdmina.
Gość	Użytkownik niezalogowany w serwisie.
Lista ulubionych	Niepubliczny zbiór przepisów oznaczonych jako ulubione przez zalogowanego użytkownika.
Lista zakupów	Niepubliczne zestawienie produktów pochodzących z wybranych przepisów lub dodanych ręcznie, umożliwiające użytkownikowi zaplanowanie zakupu artykułów niezbędnych do przygotowania potraw.
Moderacja	Wszyscy moderatorzy, administratorzy oraz SuperAdmini w serwisie.
Moderator	Użytkownik serwisu posiadający częściowe uprawnienia do weryfikowania i zarządzania treściami oraz kontami innych użytkowników, z wyłączeniem kont administratorów.
Produkt	Podstawowy artykuł spożywczy znajdujący się w bazie danych serwisu.
Profil użytkownika	Strona zawierająca informacje o danym użytkowniku, opublikowane przepisy, biografię, czy listę obserwujących/obserwowanych.
Przepis	Wykaz składników i instrukcji opisujących sposób przygotowania potrawy.
Składnik	Produkt przypisany do danego przepisu z określoną ilością i jednostką miary.
SuperAdmin	Administrator posiadający dodatkowe uprawnienia zarządzania innymi administratorami.

3.2. Wymagania funkcjonalne

Ten podrozdział zawiera tabele prezentujące wymagania funkcjonalne aplikacji podzielone według poszczególnych dostępnych ról użytkownika w serwisie:

- Gość,
- Użytkownik zalogowany,
- Moderator,
- Administrator,
- SuperAdmin.

Tabela 3 prezentuje wykaz wymagań funkcjonalnych dostępnych dla Gościa.

Tabela 3 Wymagania funkcjonalne dla Gościa. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1.	Rejestracja konta użytkownika w serwisie	Gość ma możliwość założenia konta w systemie	Gość	
2.	Wyświetlanie opublikowanego przepisu	Gość może czytać przepisy opublikowane w serwisie.	Gość, Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	
3.	Wyszukiwanie przepisów	Gość może wyszukiwać i filtrować przepisy.	Gość, Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	
4.	Wyświetlanie komentarzy pod przepisem	Gość może wyświetlić wszystkie opublikowane komentarze pod przepisem.	Gość, Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	
5.	Zalogowanie się do serwisu	Gość ma możliwość zalogowania się do serwisu.	Gość	Konto istnieje w systemie i nie zostało zablokowane.

6.	Przeglądanie profili użytkowników	Gość może wyświetlić profil użytkownika w serwisie i przeglądać jego przepisy.	Gość, Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Profil istnieje w serwisie.
----	-----------------------------------	--	---	-----------------------------

Tabela 4 zawiera zestawienie wymagań funkcjonalnych dotyczących Użytkowników zalogowanych.

Tabela 4 Wymagania funkcjonalne dla Użytkownika zalogowanego. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1.	Wylogowanie się z serwisu	Użytkownik może się wylogować z serwisu.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	
2.	Dodanie przepisu do serwisu	Użytkownik może stworzyć przepis i przesłać go do akceptacji przez moderację.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	
3.	Przeglądanie własnych przepisów	Użytkownik może przeglądać na profilu swoje zaakceptowane, oczekujące oraz odrzucone przepisy.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Przepis nie jest usunięty.
4.	Zaproponowanie produktu do serwisu	Użytkownik w trakcie tworzenia przepisu może dodać nowy produkt w serwisie, jeśli nie istnieje w bazie danych. Produkt musi zostać zaakceptowany przez moderację. Oczekujący produkt może być użyty w przepisie przez autora produktu.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Zaproponowany produkt musi mieć unikalną nazwę.
5.	Edycja własnego przepisu	Użytkownik może edytować zawartość każdego swojego przepisu. Po edycji przepis przechodzi przez ponowną akceptację moderacji.	Użytkownik zalogowany, Moderator, Administrator,	Edytowany przepis musi być własnością zalogowanego użytkownika.

			SuperAdmin	
6.	Usunięcie własnego przepisu	Użytkownik może usunąć swój przepis.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Usuwany przepis musi być własnością zalogowanego użytkownika.
7.	Ocenienie przepisu	Użytkownik może dać przepisowi ocenę od 1 do 5. Ocenę można edytować oraz usunąć.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Przepis jest zaakceptowany przez moderację.
8.	Dodanie/ usunięcie komentarza do przepisu	Użytkownik może dodać/usunąć komentarz pod przepisem. Komentarz może być odpowiedzią na inny komentarz.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Przepis jest zaakceptowany przez moderację.
9.	Zapisanie/ usunięcie przepisu z listy ulubionych	Użytkownik może dodać/usunąć przepis z własnej, prywatnej listy ulubionych przepisów.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Przepis jest zaakceptowany przez moderację.
10.	Dodanie/ usunięcie/ oznaczenie produktu w liście zakupów	Użytkownik może z poziomu przepisu lub listy zakupów dodać produkt do listy zakupów. Dodany produkt można oznaczyć jako „zakupiony” lub usunąć.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Produkt jest zaakceptowany przez moderację.
11.	Zmiana hasła	Użytkownik może zmienić hasło służące do logowania do serwisu.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Nowe hasło musi być inne niż aktualne.
12.	Zaobserwowanie/ odobserwowanie użytkownika	Użytkownik może zaobserwować/odobserwować innego użytkownika. Przepisy obserwowanych użytkowników wyświetlają się w osobnej sekcji.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Dotyczy użytkownika innego niż zalogowanego.

13.	Personalizacja profilu użytkownika	Użytkownik może dodać/edytować/usunąć swoje zdjęcie profilowe oraz biografię.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Dotyczy własnego profilu.
14.	Zgłoszenie komentarza	Użytkownik może zgłosić komentarz innego użytkownika, jeśli jego zdaniem jest niewłaściwy. W zgłoszeniu należy podać powód. Zgłoszone komentarze są poddawane do rozpatrzenia przez moderację.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Użytkownik nie może zgłosić własnego komentarza.
15.	Przeglądanie przepisów od obserwowanych użytkowników	Użytkownik może przeglądać przepisy wyłącznie od obserwowanych użytkowników na dedykowanej stronie.	Użytkownik zalogowany, Moderator, Administrator, SuperAdmin	Użytkownik obserwuje co najmniej jednego użytkownika.

Tabela 5 przedstawia wymagania funkcjonalne dla roli Moderadora.

Tabela 5 Wymagania funkcjonalne dla Moderadora. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1.	Przeglądanie wszystkich przepisów	Moderator może wyświetlić listę wszystkich przepisy, niezależnie od ich statusu.	Moderator, Administrator, SuperAdmin	
2.	Akceptacja przepisu	Moderator może zaakceptować przepis po jego weryfikacji. Akceptacja upublicznia przepis w serwisie.	Moderator, Administrator, SuperAdmin	Wszystkie składniki przepisu są zaakceptowane. Przepis nie jest zaakceptowany
3.	Odrzucenie przepisu	Moderator może odrzucić przepis po jego weryfikacji. Podczas odrzucenia należy podać powód.	Moderator, Administrator, SuperAdmin	Przepis nie jest odrzucony.
4.	Usunięcie przepisu	Moderator może usunąć przepis.	Moderator, Administrator, SuperAdmin	Przepis nie jest usunięty.

5.	Przeglądanie produktów	Moderator może wyświetlić listę wszystkich produktów, niezależnie od ich statusu.	Moderator, Administrator, SuperAdmin	
6.	Dodanie produktu	Moderator może dodać produkt do bazy danych.	Moderator, Administrator, SuperAdmin	Nazwa produktu jest unikalna.
7.	Akceptacja propozycji produktu	Moderator może zaakceptować propozycję produktu od użytkownika. Akceptacja upublicznia produkt w serwisie.	Moderator, Administrator, SuperAdmin	Nazwa produktu jest unikalna. Produkt nie jest zaakceptowany.
8.	Odrzucenie propozycji produktu	Moderator może odrzucić propozycję produktu od użytkownika. Podczas odrzucenia należy podać powód.	Moderator, Administrator, SuperAdmin	Produkt nie jest odrzucony.
9.	Przeglądanie zgłoszeń komentarzy	Moderator może wyświetlić listę nierozpatrzonych i rozpatrzonych zgłoszeń komentarzy.	Moderator, Administrator, SuperAdmin	
10.	Uwzględnienie zgłoszenia komentarza	Moderator po analizie zgłoszenia może je uwzględnić i tym samym usunąć zgłoszony komentarz. Status zgłoszenia jest zmieniony na rozpatrzony.	Moderator, Administrator, SuperAdmin	Komentarz nie jest usunięty. Komentarz jest zgłoszony. Zgłoszenie jest nierozpatrzone.
11.	Zignorowanie zgłoszenia komentarza	Moderator po analizie zgłoszenia może go nie uwzględnić. Komentarz pozostaje widoczny. Status zgłoszenia jest zmieniony na rozpatrzony.	Moderator, Administrator, SuperAdmin	Komentarz jest zgłoszony. Zgłoszenie jest nierozpatrzone.
12.	Przeglądanie listy kategorii	Moderator może wyświetlić listę wszystkich kategorii w serwisie.	Moderator, Administrator, SuperAdmin	
13.	Dodanie kategorii	Moderator może dodać nową kategorię do serwisu.	Moderator, Administrator, SuperAdmin	Nazwa kategorii jest unikalna.
14.	Edycja/usunięcie kategorii	Moderator może edytować/usunąć kategorię.	Moderator, Administrator, SuperAdmin	Kategoria istnieje na liście.
15.	Ustawienie pozycji kategorii na stronie głównej	Moderator może wyróżnić przepisy z danej kategorii poprzez	Moderator, Administrator,	

		ustawienie pozycji wyświetlania kategorii na stronie głównej.	SuperAdmin	
16.	Przeglądanie listy grup kategorii	Moderator może wyświetlić listę wszystkich grup kategorii w serwisie.	Moderator, Administrator, SuperAdmin	
16.	Dodanie grupy kategorii	Moderator może dodać nową grupę kategorii do serwisu.	Moderator, Administrator, SuperAdmin	Nazwa grupy kategorii jest unikalna.
17.	Edycja/usunięcie grupy kategorii	Moderator może edytować/usunąć grupę kategorii.	Moderator, Administrator, SuperAdmin	Grupa kategorii istnieje na liście.
18.	Przypisanie kategorii do grupy	Moderator może przypisać kategorię do grupy kategorii. Kategoria może należeć do wielu grup.	Moderator, Administrator, SuperAdmin	
19.	Zablokowanie konta użytkownika	Moderator może zablokować innemu użytkownikowi konto. Podczas blokowania należy podać powód.	Moderator, Administrator, SuperAdmin	Użytkownik nie ma roli równej lub wyższej od moderatora. Użytkownik nie ma zablokowanego konta.
20.	Odblokowanie konta użytkownika	Moderator może odblokować konto innego użytkownika.	Moderator, Administrator, SuperAdmin	Użytkownik nie ma roli równej lub wyższej od moderatora. Użytkownik ma zablokowane konto.

Tabela 6 zawiera wymagania funkcjonalne dla roli Administratora.

Tabela 6 Wymagania funkcjonalne dla Administratora. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1.	Przypisanie/odebranie roli Moderatora	Administrator może użytkownikowi i moderatorowi przypisać/odebrać rolę Moderator.	Administrator, SuperAdmin	Dotyczy użytkownika innego niż zalogowanego. Administrator nie może modyfikować ról

				innym administratorom.
2.	Usunięcie produktu	Administrator może usunąć produkt z serwisu.	Administrator, SuperAdmin	Produkt nie jest usunięty.

Tabela 7 przedstawia wymagania funkcjonalne dla roli SuperAdmin.

Tabela 7 Wymagania funkcjonalne dla SuperAdmina. Źródło: Opracowanie własne.

Lp.	Nazwa	Opis	Aktorzy	Warunki początkowe
1.	Przypisanie/odebranie roli Administratora	SuperAdmin może użytkownikowi przypisać/odebrać rolę Administratora.	SuperAdmin	Dotyczy użytkownika innego niż zalogowanego. SuperAdmin nie może modyfikować swoich ról.

3.3. Wymaganie niefunkcjonalne

Ten podrozdział przedstawia wymaganie niefunkcjonalne aplikacji. Definiują one jakościowe aspekty, które serwis musi albo powinien spełniać, by zapewnić prawidłowe działanie. Wymagania niefunkcjonalne zaprezentowano w Tabeli 8.

Tabela 8 Wymagania niefunkcjonalne. Źródło: Opracowanie własne.

Lp.	Opis	Aspekt
1.	Serwis obsługuje co najmniej 1000 użytkowników jednocześnie	Wydajność
2.	Maksymalny czas odpowiedzi nie powinien przekroczyć 5 sekund	
3.	Hasła powinny być szyfrowane	Bezpieczeństwo
4.	System zapewnia bezpieczny sposób uwierzytelniania i trzymania sesji.	
5.	Serwis obsługuje najpopularniejsze przeglądarki internetowe: Chrome, Firefox, Safari, Edge, Opera	Użyteczność
6.	System jest dostępny w języku polskim	
7.	Interfejs użytkownika jest responsywny i wyświetla się poprawnie na komputerze, smartfonie i tablecie.	
8.	Interfejs użytkownika jest czytelny i zrozumiały dla użytkownika.	
9.	System jest zgodny ze standardami dostępności WCAG 2.0	Dostępność
10.	System jest dostępny przez 99.9% czasu z wyjątkiem planowanych przerw technicznych.	Niezawodność

3.4. Aktorzy systemu

Aktorów systemu można podzielić na odbiorców serwisu oraz moderację. Do grupy odbiorców należą:

- Gość,
- Użytkownik zalogowany.

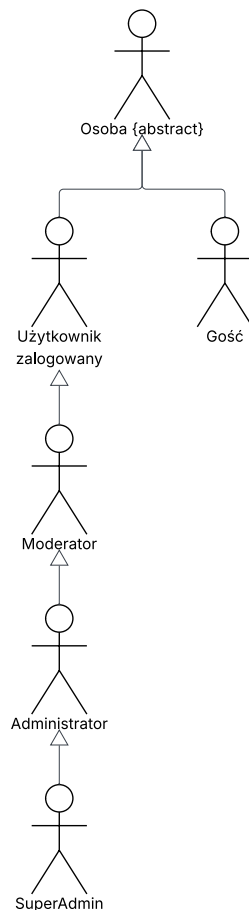
Ta grupa stanowi fundament aktywności aplikacji. Odbiorcy odpowiadają za konsumpcję oraz dostarczanie treści do serwisu.

Do grupy moderacji należą:

- Moderator,
- Administrator,
- SuperAdmin.

Do zadań moderacji należy bieżąca weryfikacja treści, nadzór nad kontami i zachowaniem użytkowników oraz utrzymanie aplikacji.

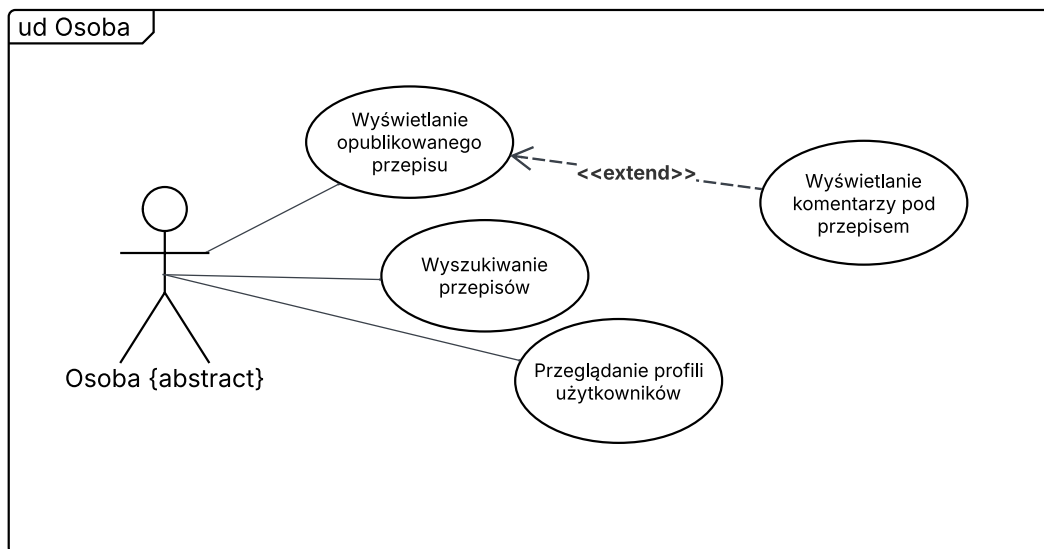
Rysunek 19 przedstawia diagram ilustrujący aktorów serwisu Let Me Cook:



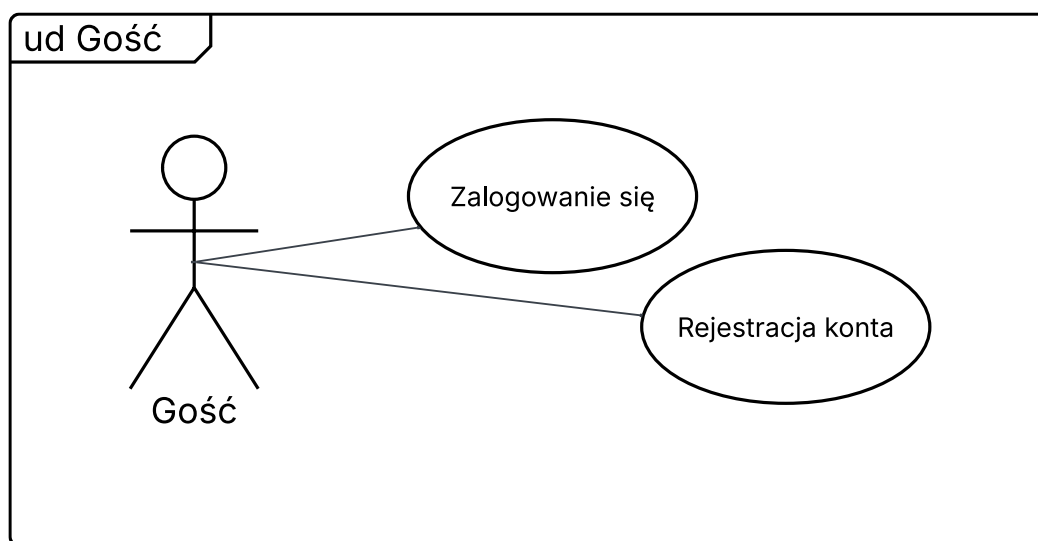
Rysunek 19 Diagram aktorów systemu Let Me Cook. Źródło: Opracowanie własne.

3.5. Model przypadków użycia

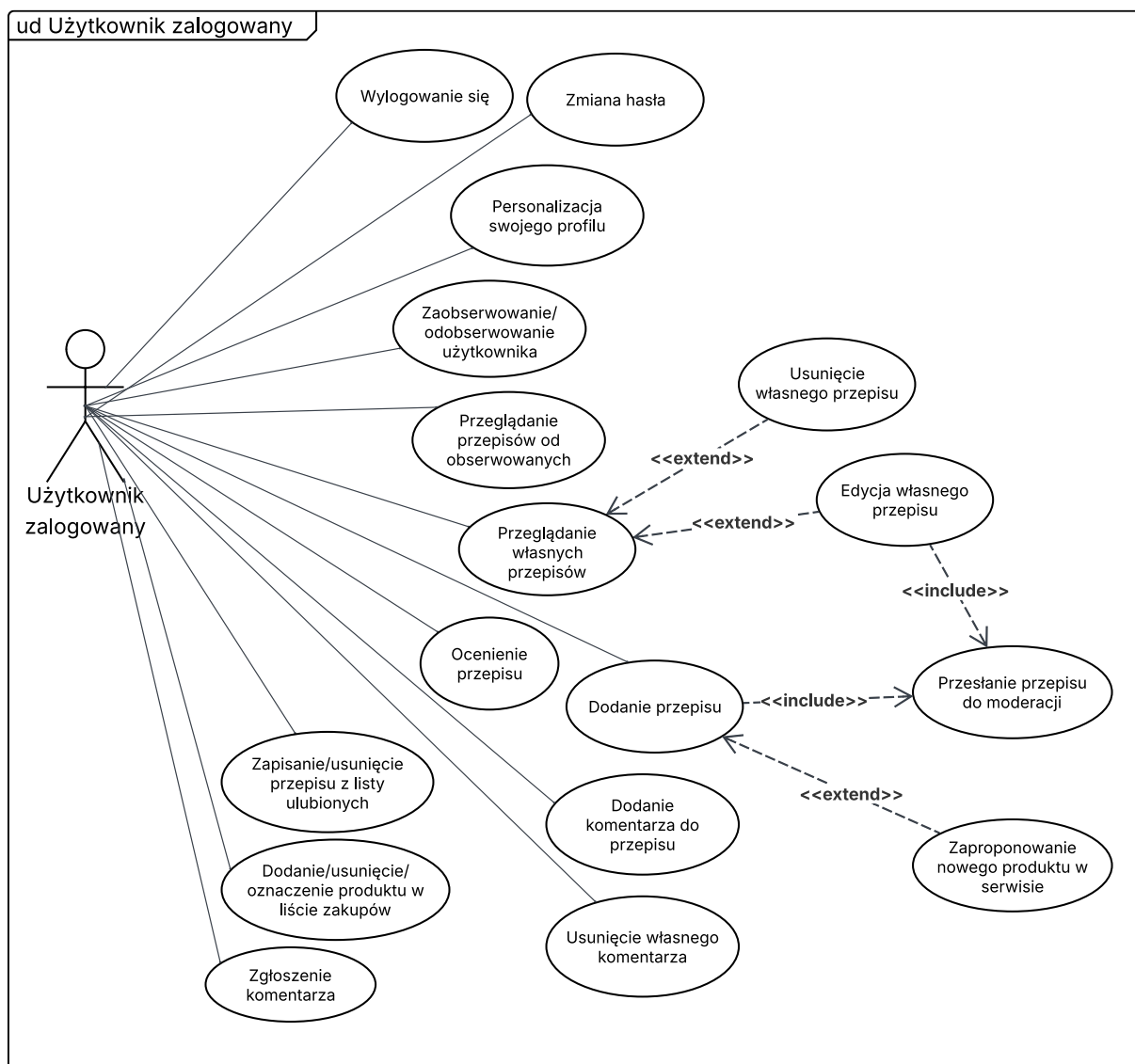
W tym podrozdziale przedstawiono diagramy przypadków użycia systemu *Let Me Cook*. Każdy diagram przedstawia przypadki dla konkretnego, pojedynczego aktora.



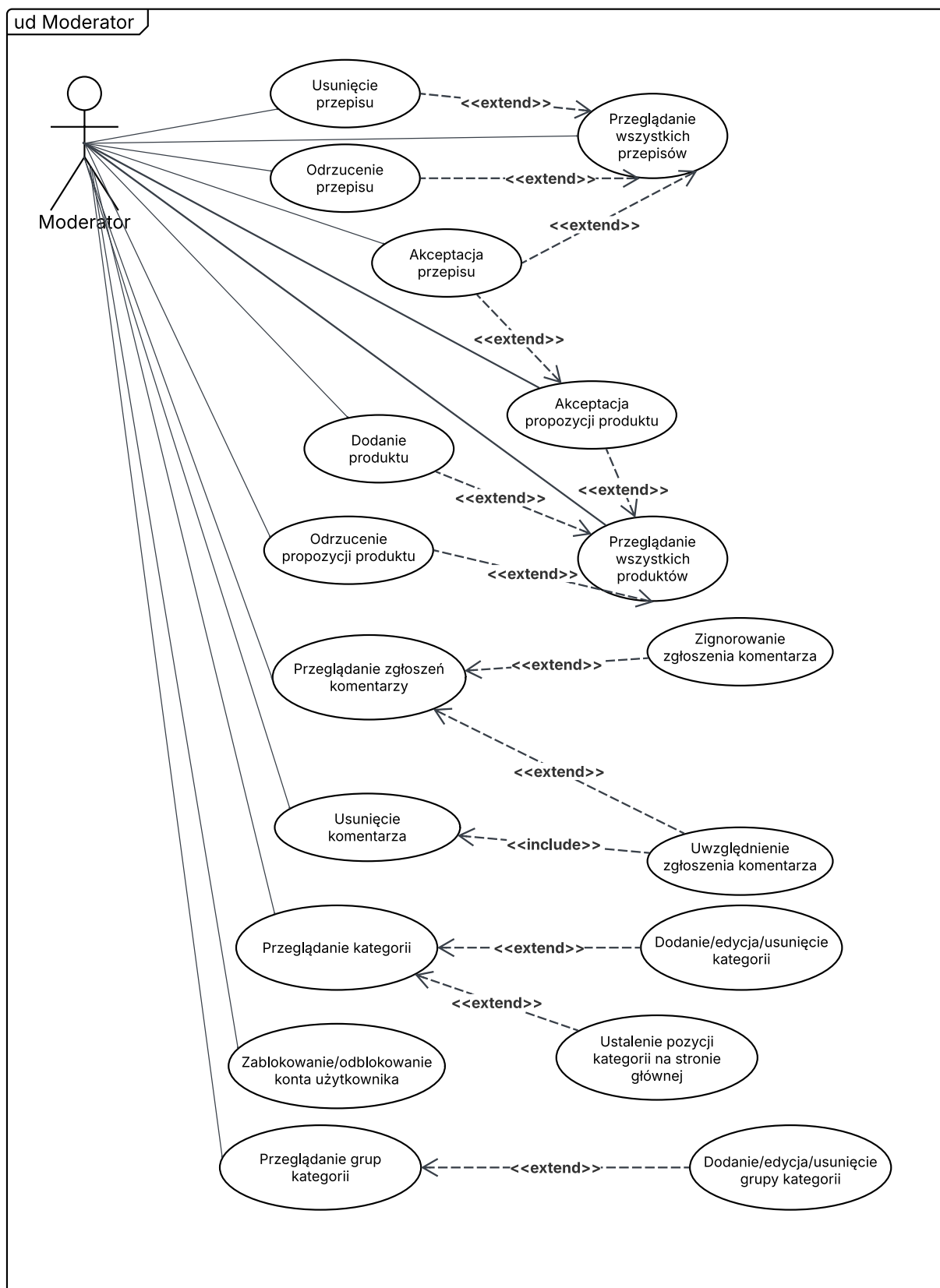
Rysunek 20 Diagram przypadków użycia Osoby. Źródło: Opracowanie własne.



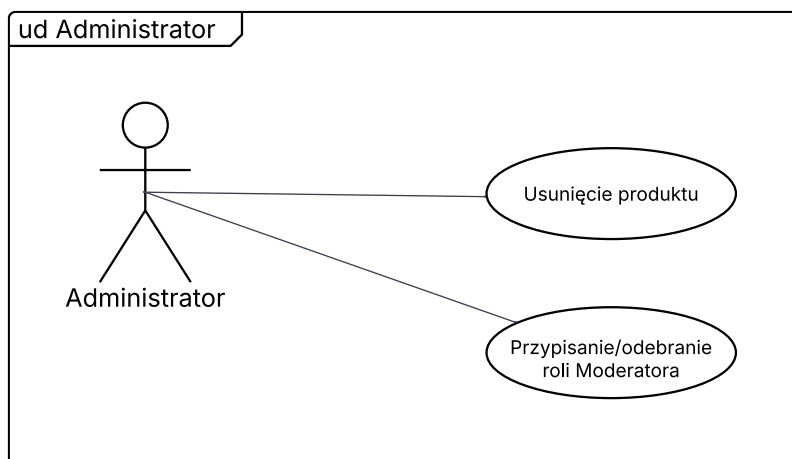
Rysunek 21 Diagram przypadków użycia Gościa. Źródło: Opracowanie własne.



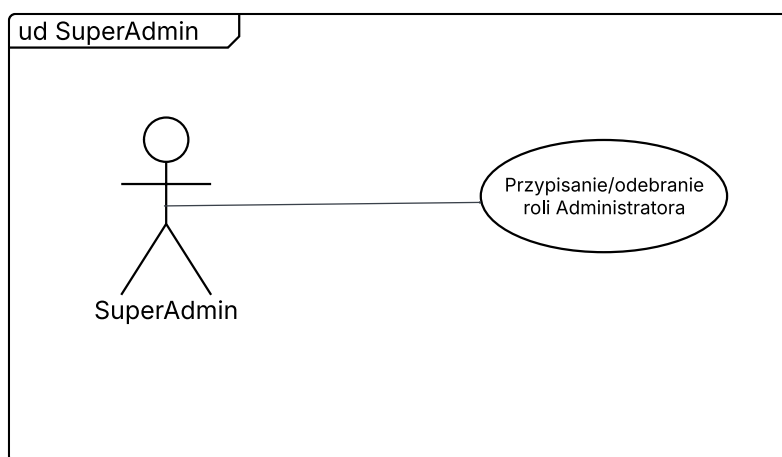
Rysunek 22 Diagram przypadków użycia Użytkownika zalogowanego. Źródło: Opracowanie własne.



Rysunek 23 Diagram przypadków użycia Moderatora. Źródło: Opracowanie własne.



Rysunek 24 Diagram przypadków użycia Administratora. Źródło: Opracowanie własne.

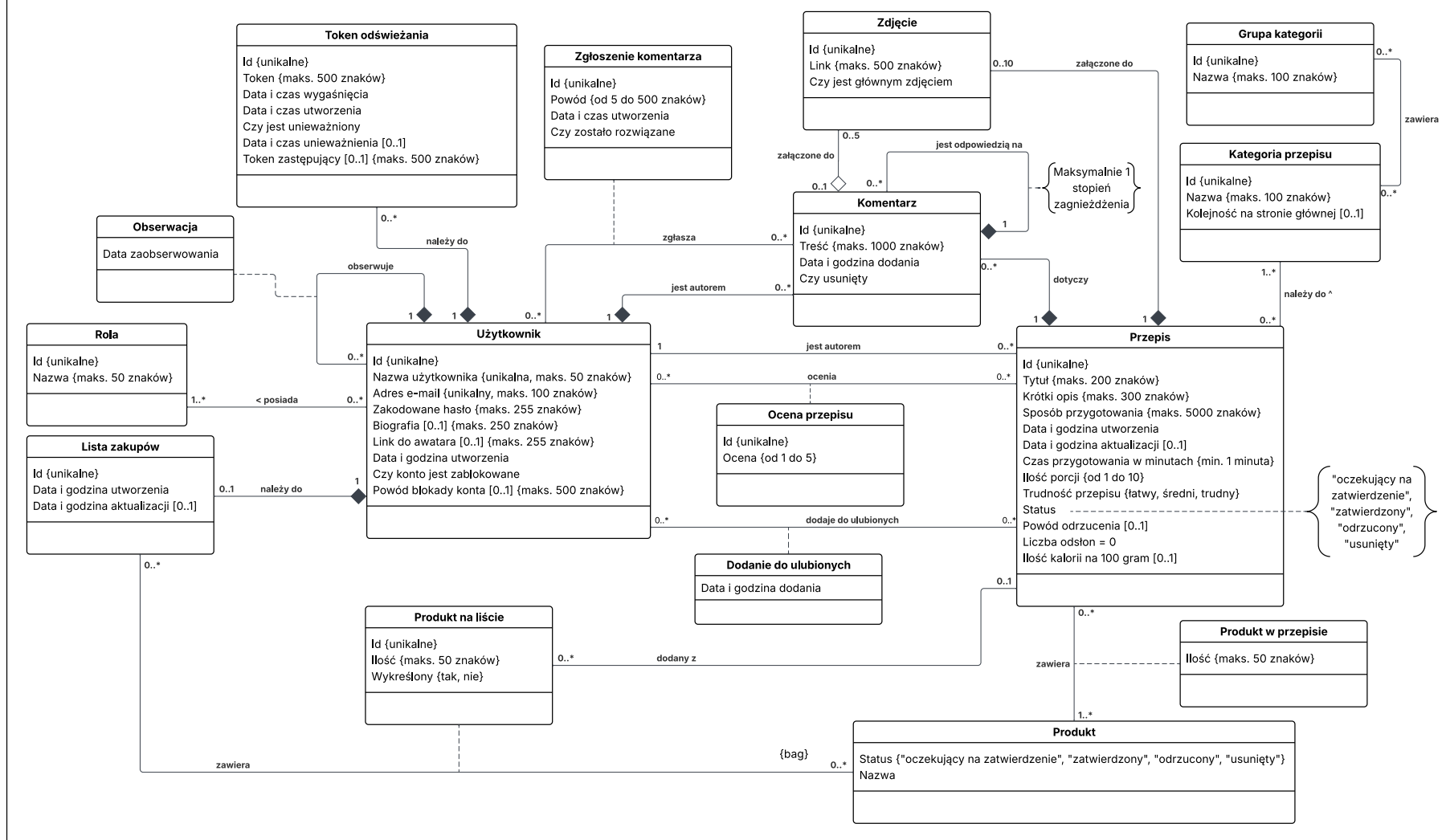


Rysunek 25 Diagram przypadków użycia SuperAdmina. Źródło: Opracowanie własne.

3.6. Analityczny diagram klas

Rysunek 26 przedstawia analityczny diagram klas ilustrujący klasy i asocjacje w opisywanym systemie.

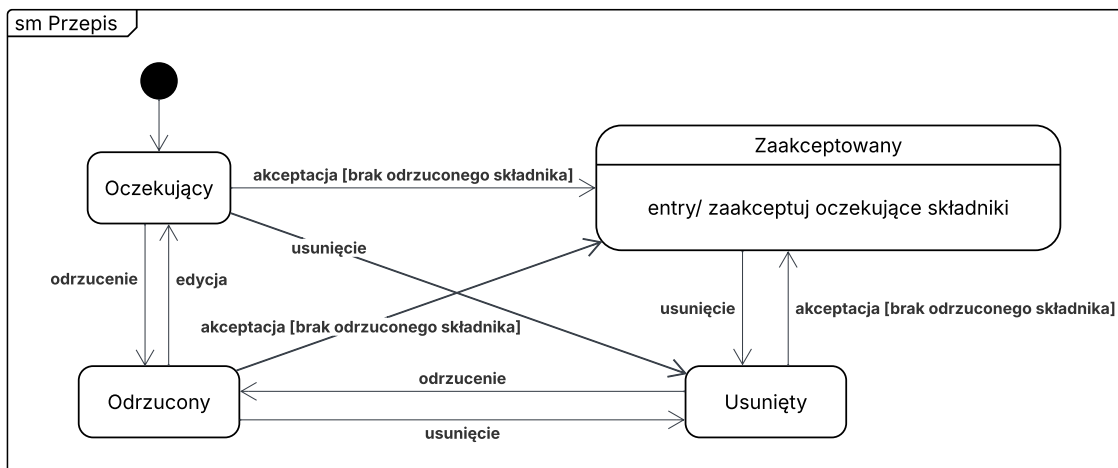
cld "Let Me Cook" Portal społecznościowy z przepisami kulinarnymi



Rysunek 26 Analityczny diagram klas. Źródło: Opracowanie własne.

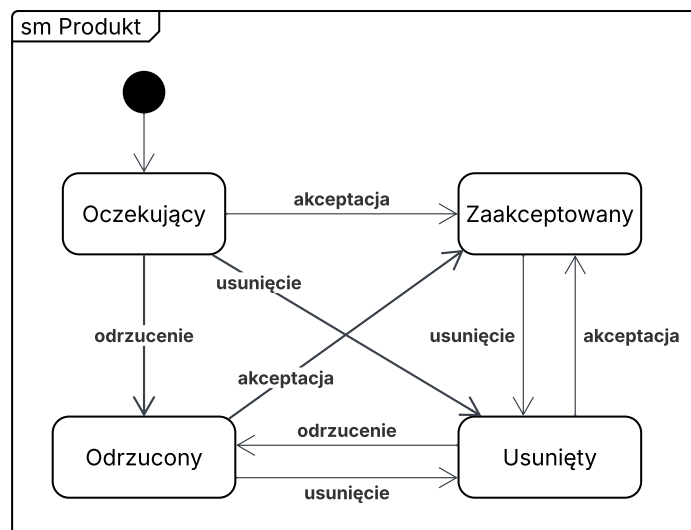
3.7. Diagramy stanów

Rysunek 27 prezentuje diagram stanu przepisu w systemie. Akceptację i odrzucenie przepisu może wykonać jedynie moderacja.



Rysunek 27 Diagram stanu przepisu. Źródło: Opracowanie własne.

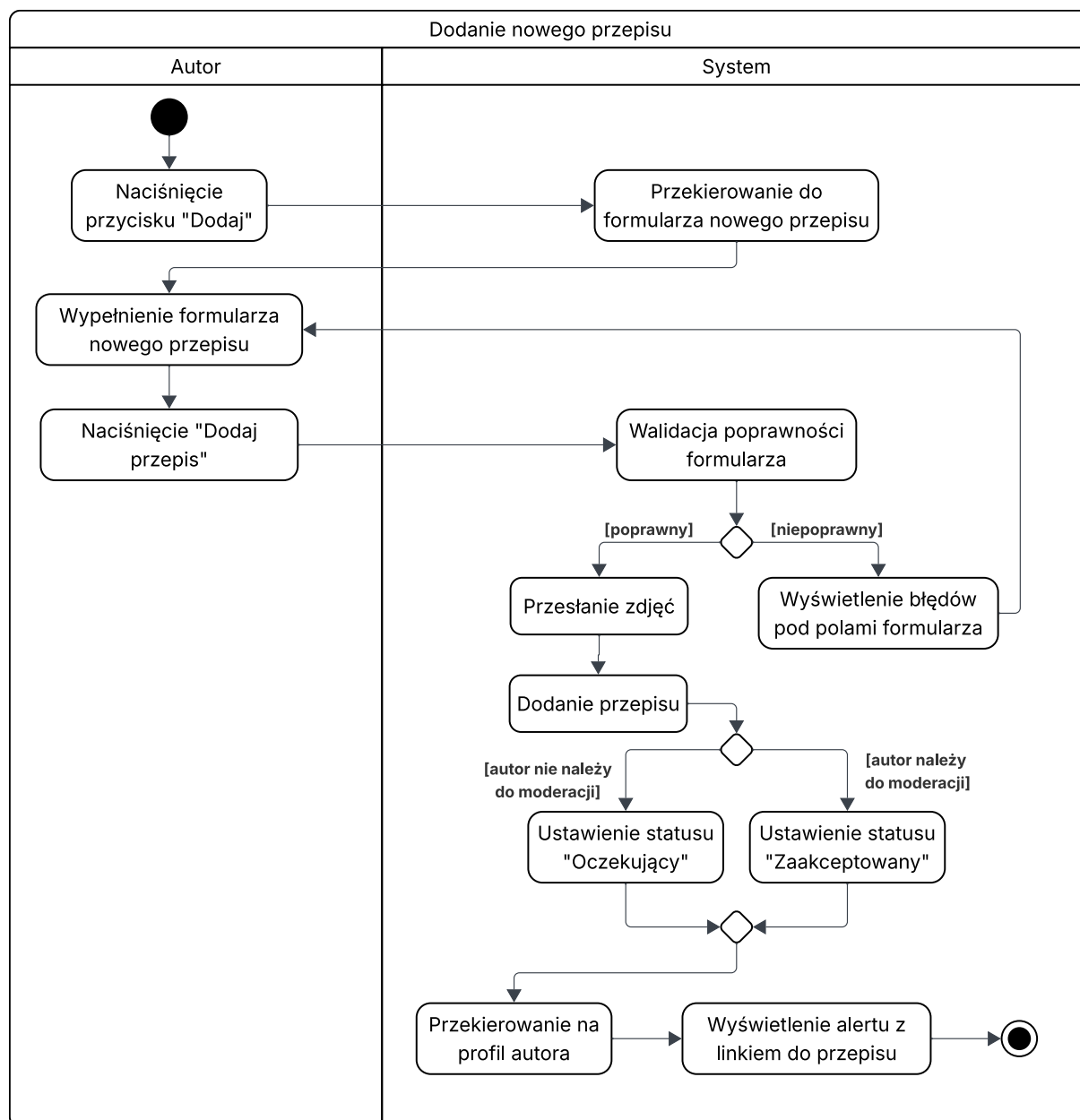
Rysunek 28 przedstawia diagram stanu produktu w systemie. Akceptację i odrzucenie produktu może wykonać jedynie moderacja.



Rysunek 28 Diagram stanu produktu. Źródło: Opracowanie własne.

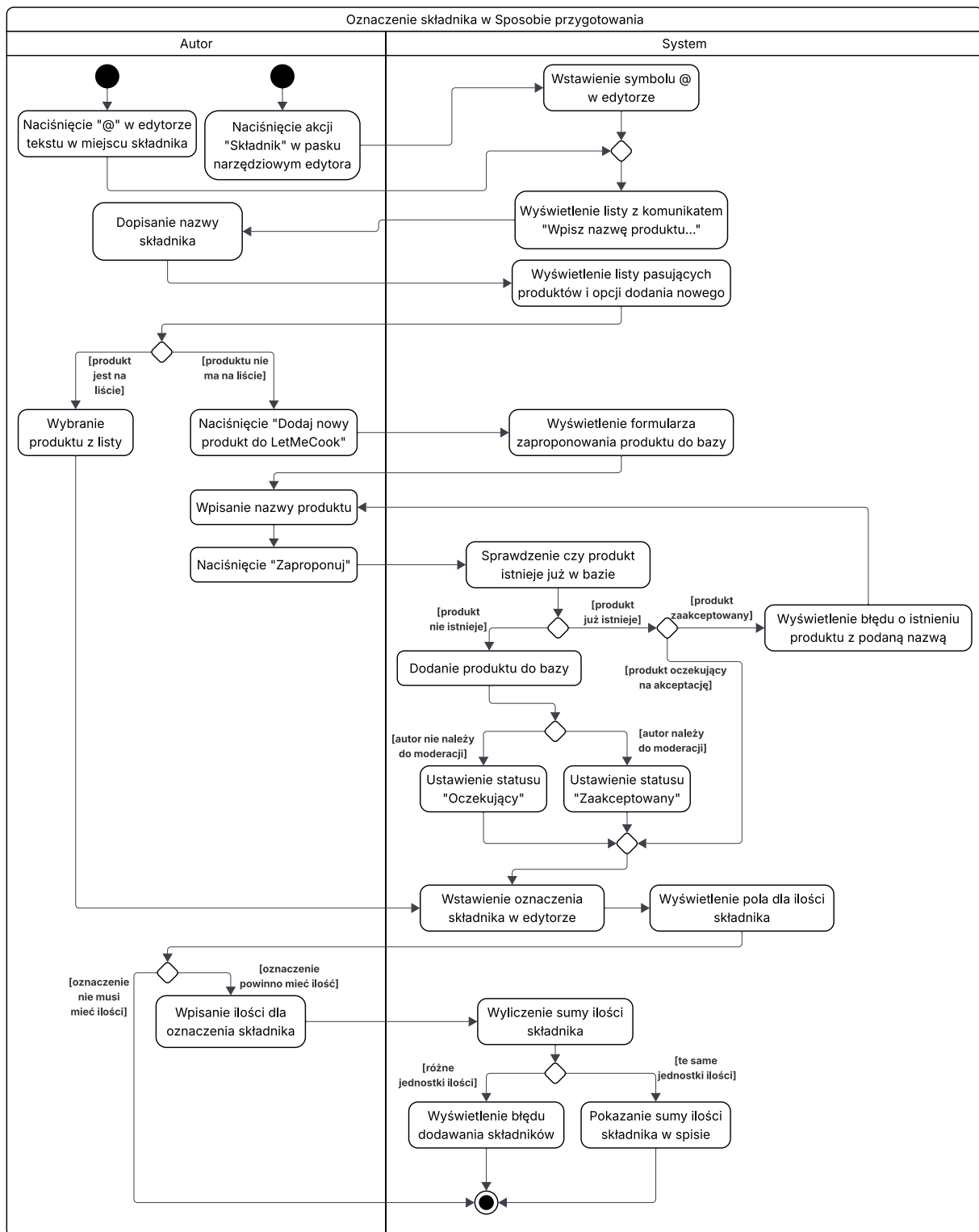
3.8. Diagramy aktywności

Rysunek 29 przedstawia diagram aktywności dla przypadku użycia dodania nowego przepisu w serwisie.



Rysunek 29 Diagram aktywności dla dodania nowego przepisu. Źródło: Opracowanie własne.

Rysunek 30 prezentuje diagram aktywności dla dodawania oznaczenia składnika w polu Sposób przygotowania w formularzu dodawania nowego przepisu.



Rysunek 30 Diagram aktywności dla oznaczenia składnika w polu Sposób przygotowania.

Źródło: Opracowanie własne.

4. Technologie oraz narzędzia wykorzystane w projekcie

Rozdział ten poświęcony jest charakterystyce technologii oraz narzędzi, które posłużyły do zaprojektowania, implementacji i przetestowania aplikacji. Omówiono w nim rozwiązania zastosowane zarówno po stronie serwerowej, jak i klienckiej, a także oprogramowanie umożliwiające konteneryzację, kontrolę wersji oraz organizację pracy w zespole.

4.1. Część serwerowa – backend

Warstwa serwerowa stanowi fundament prawidłowego działania aplikacji. To właśnie w niej zachodzą kluczowe operacje związane z wymianą informacji w standardzie REST API, ruchem sieciowym oraz trwałym zapisem danych. Poniżej przedstawiono technologie, których użyto do zbudowania systemu zintegrowanego, bezpiecznego, odpornego na błędy i gotowego na ewentualną rozbudowę.

4.1.1 Środowisko uruchomieniowe .NET

Pierwsza wersja platformy .NET [14] została wydana przez firmę Microsoft w 2002 roku i funkcjonowała pod nazwą .NET Framework. Początkowo była powiązana wyłącznie z systemami operacyjnymi z rodziny Windows. Przełomowe dla tej technologii było stworzenie od podstaw w 2016 roku środowiska .NET Core, które zapoczątkowało erę wieloplatformowości. W 2020 roku wraz z wydaniem wersji .NET 5 człon „Core” został usunięty z nazwy platformy.

W niniejszym projekcie autorzy wykorzystali platformę .NET w wersji 8. Jest to wersja LTS (ang. *Long Term Support*), która zadebiutowała w listopadzie 2023 roku.

Proces kompilacji kodu jest w .NET podzielony na dwa etapy. Najpierw kod źródłowy zamieniany jest przez dedykowany kompilator na niezależny od sprzętu wspólny język pośredni CIL (ang. *Common Intermediate Language*) [15]. W kolejnym etapie wykorzystywany jest kluczowy element platformy .NET – środowisko uruchomieniowe CLR (ang. *Common Language Runtime*) [16]. Jego wewnętrzny kompilator JIT (ang. *Just In Time*) konwertuje w trakcie działania aplikacji tylko wywoływany kod CIL na kod maszynowy zrozumiały dla docelowego procesora i systemu operacyjnego. Takie działanie kompilatora ma na celu zaoszczędzenie zasobów na konwersję nieużywanego kodu. Warto dodać, że od wersji siódmej platformy .NET możliwa jest zmiana modelu kompilacji JIT na Native AOT (ang. *Ahead Of Time*) [17], który pozwala na kompilację kodu do postaci maszynowej już na etapie budowania aplikacji. Umożliwia to głównie szybszy start i mniejsze zużycie pamięci RAM kosztem maksymalnej przepustowości i elastyczności aplikacji oraz – w trakcie realizacji projektu – ograniczonego działania niektórych narzędzi, jak np. *Entity Framework Core*. Ze względu na wymienione ograniczenia modelu AOT autorzy podczas implementacji projektu postawili na lepiej sprawdzający się dla aplikacji wymagających ciągłego działania i umożliwiający wykorzystanie większej ilości dynamicznych funkcjonalności model JIT.

4.1.2 C#

Pierwsze prace nad językiem C# [18] jako nowym obiektowym językiem programowania – początkowo rozwijanym pod nazwą „Cool” – realizowane przez zespół inżynierów firmy Microsoft, pod przewodnictwem Andersa Hejlsberga, rozpoczęły się na przełomie wieków XX i XXI. Jego pierwsza publiczna wersja pojawiła się w roku 2002 wraz z platformą .NET Framework 1.0.

Z biegiem czasu język ten dynamicznie ewoluował, a wraz z premierą .NET 5 i towarzyszącego mu C# 9.0 w listopadzie 2020 roku, Microsoft ustalił stały harmonogram wydawania nowych wersji:

- W lata nieparzyste (np. 2021, 2023) wydawane są wersje LTS wspierane przez 3 lata,

- W lata parzyste (np. 2022, 2024) wydawane są wersje STS (ang. *Standard Term Support*) wspierane przez 18 miesięcy.

Autorzy w niniejszym projekcie zastosowali język C# w wersji 12 wydany w listopadzie 2023, który jest domyślnym standardem dla środowiska uruchomieniowego .NET 8.

Dzięki ciągłemu rozwojowi C# stał się nowoczesnym i wszechstronnym językiem stanowiącym bezpośrednią konkurencję dla takich języków jak chociażby Java czy C++. Do najważniejszych cech jakie oferuje ten język należą:

- Obiektowy paradygmat z obsługą interfejsów i typów generycznych,
- Zintegrowane zapytania LINQ (ang. *Language Integrated Query*) [19],
- Nowoczesna obsługa asynchroniczności – mechanizm `async/await`,
- Zaawansowana obsługa wyjątków.

Przykładowy program ilustrujący składnię, deklarację zmiennej, podstawowe instrukcje sterujące oraz operację wejścia/wyjścia w języku C# przedstawiony jest w Listing 1.

Listing 1 Przykładowy program w języku C#. Źródło: Opracowanie własne.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        int maxNumber = 10;

        for (int i = 1; i <= maxNumber; i++)
        {
            if (i % 2 == 0)
            {
                Console.WriteLine($"{i} is an even number.");
            }
            else
            {
                Console.WriteLine($"{i} is an odd number.");
            }
        }
    }
}
```

Jedną z największych zalet języka C# jest jego ciągły i przemyślany rozwój. Oferuje on wiele rozwiązań ułatwiających pracę programistom takich jak chociażby *Garbage Collector* [20], czyli mechanizm automatycznego zarządzania pamięcią. Dodatkowo bogaty bazowy zasób typów, wbudowane narzędzia takie jak LINQ oraz natywne wsparcie dla asynchroniczności sprawiają, że C# jest językiem wysoce efektywnym i wszechstronnym.

4.1.3 ASP.NET Core

ASP.NET Core [21] jest otwartoźródłowym frameworkiem stworzonym przez firmę Microsoft przeznaczonym do tworzenia rozwiązań sieciowych na platformie .NET. Narzędzie to zostało wydane w 2016 roku jako następca funkcjonującego do tamtego czasu ASP.NET [22] będącego przez

kilkanaście lat standardem tworzenia aplikacji webowych w ekosystemie Microsoftu. Framework ASP.NET Core pomimo niewielkiej różnicy w nazwie względem poprzednika nie był tylko jego kolejną wersją, a zupełnie nowym, stworzonym od podstaw narzędziem wnoszącym wiele znaczących zmian technologicznych. Największymi atutami ASP.NET Core, których brakowało w ASP.NET są:

- Niezależność od platformy – aplikacje mogą być tworzone i hostowane na systemach Windows, Linux oraz macOS,
- Architektura modułowa – ASP.NET Core podzielony jest na niewielkie pakiety NuGet, co pozwala na dołączenie do projektu wyłącznie potrzebnych modułów i znaczące zmniejszenie rozmiaru aplikacji,
- Znaczna poprawa wydajności – ASP.NET Core jest obecnie uważany za jeden z najszybszych frameworków na rynku wielokrotnie przewyższając pod tym względem swojego poprzednika,
- Wstrzykiwanie zależności (ang. *Dependency Injection*) – w frameworku będącym standardem do 2016 roku programiści musieli polegać na zewnętrznych bibliotekach do zarządzania cyklem życia obiektów, natomiast w nowym narzędziu wzorec ten jest domyślnie dostępny.

Przykładowy podstawowy kontroler w ASP.NET Core przedstawiony jest w Listing 2.

Listing 2 Przykładowy podstawowy kontroler w ASP.NET Core. Źródło: Opracowanie własne.

```
using Microsoft.AspNetCore.Mvc;

namespace WebApi.Controllers;

[ApiController]
[Route("api/[controller]")]
public class InfoController : ControllerBase
{
    [HttpGet]
    public IActionResult GetSystemInfo()
    {
        var info = new
        {
            Name = "Web API",
            Version = "2.0.0"
        };

        return Ok(info);
    }
}
```

Zastosowanie frameworka ASP.NET Core pozwoliło na stworzenie wydajnej, skalowalnej i łatwej w utrzymaniu warstwy sieciowej. Powstałe przy jego użyciu interfejsy API stanowią niezależne i bezpieczne węzły systemu, z którymi aplikacja kliencka może komunikować się pośrednio przez komponent API Gateway.

4.1.4 Entity Framework Core

Entity Framework Core [23] to nowoczesny otwartoźródłowy framework typu ORM (ang. *Object-Relational Mapping*) stworzony i rozwijany przez firmę Microsoft. Jego zadaniem jest

zapewnienie prostej i bezpiecznej komunikacji pomiędzy serwerową częścią aplikacji a bazą danych. Narzędzie to zadebiutowało w 2016 roku zastępując będący od 2008 roku standardem dla programistów .NET ORM Entity Framework [24]. Podobnie jak w przypadku narzędzia ASP.NET Core, nowa generacja frameworka nie była jedynie kolejną aktualizacją, lecz stworzonym całkowicie od podstaw projektem.

Podstawowymi zaletami, które zadecydowały o użyciu Entity Framework Core w niniejszym projekcie oraz które dają mu przewagę nad starszym Entity Frameworkiem są:

- Wieloplatformowość – Entity Framework Core działa nie tylko w systemie Windows, ale również w systemach Linux i macOS,
- Podział frameworka na mniejsze komponenty – ORM został podzielony na małe opcjonalne pakiety NuGet, dzięki czemu programista może dołączyć do aplikacji tylko te części narzędzia, które będą faktycznie używane,
- Skok wydajności – nowa wersja radzi sobie znacznie szybciej ze śledzeniem zmian i przetwarzaniem dużych zbiorów danych dzięki m.in. optymalizacji mechanizmu tworzenia zapytań SQL.

Centralnym punktem narzędzia Entity Framework Core jest klasa dziedzicząca po DbContext [25], która odpowiada za zarządzanie połączeniem z bazą danych, śledzenie zmian w danych oraz tłumaczenie wysokopoziomowego kodu C# na zoptymalizowane zapytania SQL. Przykład takiej klasy zawierającej dwie encje oraz zdefiniowaną relację między nimi ilustruje Listing 3.

Listing 3 Przykładowa klasa dziedzicząca po DbContext. Źródło: Opracowanie własne.

```
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options) : base(options)
    {
    }

    public DbSet<ShoppingList> ShoppingLists { get; set; }
    public DbSet<Product> Products { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<ShoppingList>()
            .HasMany(s => s.Products)
            .WithOne(p => p.ShoppingList)
            .HasForeignKey(p => p.ShoppingListId)
            .OnDelete(DeleteBehavior.Cascade);
    }
}
```

Głównym celem zastosowania Entity Framework Core jest stworzenie warstwy abstrakcji, która umożliwia operacje na danych przy wykorzystaniu jedynie języka C#, zwalniając programistę z obowiązku ręcznego pisania zapytań SQL. Narzędzie zapewnia wysoką wydajność aplikacji dzięki integracji z takimi mechanizmami jak wstrzykiwanie zależności, czy asynchroniczność oraz czyni kod mniej podatnym na błędy i łatwiejszym w testowaniu.

4.1.4.1 Modele danych i podejście Code-First

W prezentowanym systemie wykorzystano Entity Framework Core z podejściem *Code-First*. Oznacza to, że struktura bazy danych zdefiniowana jest bezpośrednio w kodzie źródłowym aplikacji.

W technice Code-First do modelowania danych używa się klas języka C# nazywanych encjami (ang. *Entities*). Każda taka klasa odpowiada pojedynczej tabeli w bazie danych, a jej właściwości poszczególnym kolumnom tej tabeli. Do określenia ograniczeń, relacji i innych zależności w docelowej bazie danych wykorzystuje się następujące mechanizmy:

- Atrybuty mapowania (ang. *Data Annotations*) – atrybuty umieszczane bezpośrednio nad właściwościami w klasach
- Fluent API – konfiguracja wewnątrz głównej klasy kontekstu bazy danych

4.1.4.2 Migracje bazy danych

Aby zachować spójność pomiędzy modelem danych w kodzie C# a faktyczną strukturą bazy, bez konieczności ręcznego pisania skryptów SQL, stosuje się mechanizm migracji bazy danych obecny w Entity Framework Core.

Główną zaletą migracji jest umożliwienie przyrostowego wersjonowania schematu bazy danych. Zmiany w klasach (encjach), zarejestrowanych w klasie dziedziczącej po DbContext, są tłumaczone na generowane po utworzeniu migracji pliki w języku C#, które zawierają instrukcje opisujące, jakie zmiany należy wprowadzić w bazie. Entity Framework Core automatycznie rejestruje historię zaaplikowanych migracji w tabeli `__EFMigrationsHistory` wewnątrz bazy danych. Pozwala to również na łatwe cofnięcie struktury bazy do jej dowolnej historycznej wersji.

Tworzenie migracji odbywa się z poziomu wiersza poleceń – wywołanie komendy `dotnet ef migrations add` generuje nowy plik migracji, natomiast fizyczne zastosowanie zmian w strukturze bazy danych ma miejsce po wykonaniu komendy `dotnet ef database update`.

4.1.5 REST API

Jednym z kluczowych architektonicznych aspektów złożonych systemów informatycznych jest zapewnienie wydajnego i ujednoliconego sposobu komunikacji pomiędzy różnymi modułami systemu. W prezentowanym projekcie zadanie to realizuje REST API [26], czyli rodzaj interfejsu programistycznego, zapewniającego dostęp do określonych zasobów, zaprojektowanego zgodnie z zestawem reguł REST (ang. *Representational State Transfer*).

Zasady architektury REST zostały zdefiniowane przez Roya Fieldinga w 2000 roku. Obecnie ten rodzaj architektury jest najczęściej spotykanym standardem w aplikacjach wielowarstwowych. Aplikacje zaimplementowane zgodnie z praktykami REST określa się też jako systemy RESTful.

Do najważniejszych założeń architektury REST należą:

- Jednolity interfejs – każdy zasób musi być jednoznacznie identyfikowany przez unikalny adres URL (ang. *Uniform Resource Locator*),
- Niezależność aplikacji klienckiej i serwerowej – warstwa prezentacji powinna być wyraźnie oddzielona od logiki biznesowej i przechowywania danych,
- Bezstanowość – serwer nie przechowuje kontekstu pomiędzy zapytaniami, dlatego każde żądanie klienta musi zawierać wszystkie informacje niezbędne do jego obsłużenia.

W systemach RESTful do komunikacji wykorzystywany jest protokół HTTP. Operacje na zasobach realizowane są poprzez określenie parametrów zasobu w adresie URL lub ciele zapytania oraz zastosowanie odpowiednich metod tego protokołu:

- GET – do pobierania informacji o zasobie,

- POST – do tworzenia nowych zasobów,
- PUT lub PATCH – do aktualizacji zasobu,
- DELETE – do usunięcia zasobu.

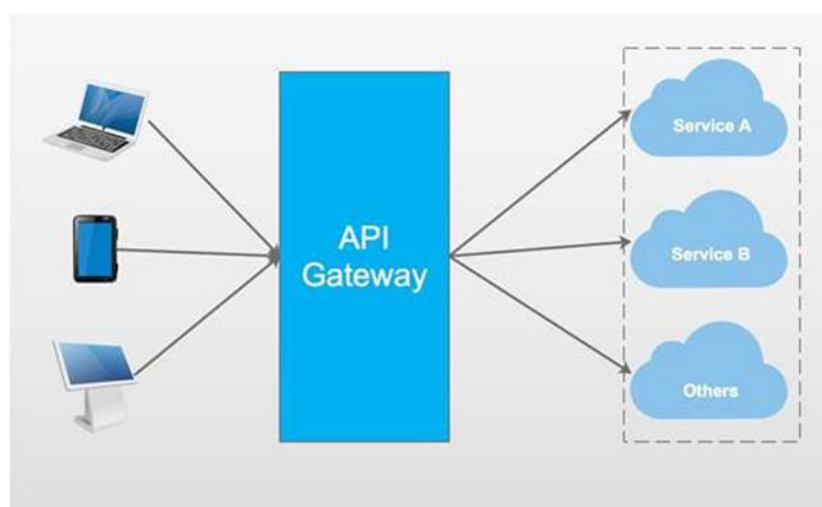
W architekturze REST zarówno dane w żądaniach klienta, jak i w odpowiedziach ze strony serwera przesyłane są najczęściej w formacie JSON. Nerozłącznym elementem odpowiedzi serwera są też ustandaryzowane kody statusu HTTP informujące o tym, czy żądanie zostało obsłużone pomyślnie, czy zakończyło się błędem.

4.1.6 Ocelot

W architekturze mikroserwisów bezpośrednia komunikacja aplikacji klienckiej z wieloma usługami po stronie serwera generuje potencjalne problemy związane z bezpieczeństwem, utrzymaniem oraz skalowalnością systemu. Rozwiązaniem eliminującym te kłopoty jest wprowadzenie bramy dostępowej (ang. *API Gateway*) pośredniczącej w komunikacji klient-serwer. Narzędziem umożliwiającym implementację takiej usługi w .NET jest Ocelot [27].

Ocelot działa jako odwrotne proxy (ang. *reverse proxy*), czyli przechwytuje cały ruch sieciowy skierowany do pojedynczych serwisów oraz z nich wychodzący. Oznacza to, że architektura systemu i adresy IP pojedynczych usług pozostają całkowicie ukryte przed światem zewnętrznym, co znacznie zwiększa ich bezpieczeństwo. Ocelot oferuje też możliwość weryfikacji tokenów autoryzacyjnych zdejmując przy tym z poszczególnych mikroserwisów odpowiedzialność sprawdzania tożsamości użytkownika. Narzędzie zapewnia również wsparcie dla mechanizmów równoważnia obciążenia (ang. *Load Balancing*) umożliwiających rozdzielanie ruchu pomiędzy wiele instancji tej samej usługi.

Rysunek 31 przedstawia schemat komunikacji w systemie wykorzystującym bramę dostępową.



Rysunek 31 Schemat komunikacji w systemie wykorzystującym bramę dostępową. Źródło: [28]

Narzędzie Ocelot w omawianym projekcie stanowi kluczowy element integrujący rozproszoną architekturę backendową w jeden spójny interfejs programistyczny gotowy do komunikacji z docelową aplikacją kliencką.

4.1.7 Serilog

W procesie tworzenia i utrzymania systemów informatycznych niezwykle pomocne w diagnozowaniu i analizie problemów są mechanizmy rejestrowania zdarzeń (ang. *logging*). Pomimo, że ASP.NET Core posiada swój własny system logowania, w prezentowanym projekcie zdecydowano się na zastąpienie go zewnętrzną, otwartoźródłową biblioteką Serilog [29].

Głównym argumentem przemawiającym za wyborem tego narzędzia było jego działanie oparte na tzw. ujściach (ang. *Sinks*). Mechanizm ten umożliwia bardzo proste przekierowanie logów do różnych punktów docelowych jednocześnie. Taka funkcjonalność pozwoliła w omawianym projekcie na zapis zarejestrowanych zdarzeń w bazie danych oraz wypisywanie ich na konsoli w tym samym czasie. Serilog oferuje również możliwość definiowania różnych poziomów logowania dla poszczególnych ujść. W prezentowanym systemie autorzy zdecydowali się wykorzystać tę funkcję biblioteki. Ustalono, że minimalnym poziomem logów trwale zapisywanych w bazie ma być poziom *warning*, aby nie zajmować pamięci mniej istotnymi informacjami diagnostycznymi. Na standardowe wyjście konsoli trafiają natomiast wszystkie bieżące zdarzenia, co umożliwia podgląd działania systemu na żywo.

Wdrożenie narzędzia Serilog pozwoliło przede wszystkim na stosunkowo łatwe zapewnienie trwałości oraz dalszą analizę najważniejszych informacji diagnostycznych w bazie SQL. Jego zastosowanie zapewniło też programistom czytelniejszy podgląd zdarzeń w konsoli dzięki bardziej przejrzystemu formatowaniu logów, niż w przypadku domyślnego mechanizmu logowania w .NET.

4.2. Aplikacja webowa – frontend

Warstwa kliencka odpowiada za prezentację treści i sposób interakcji użytkownika z serwisem. Jej poprawna implementacja ma kluczowe znaczenie dla ogólnego odczucia podczas użytkowania. W tym celu zastosowano zestaw technologii opisanych poniżej, dobranych z uwzględnieniem specyfiki projektu oraz aktualnych standardów.

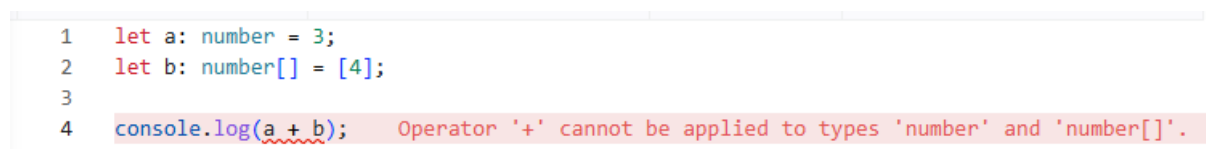
4.2.1 TypeScript

Głównym językiem programowania użytym po stronie aplikacji webowej jest TypeScript. To otwartoźródłowy nadzbiór języka JavaScript utworzony przez Microsoft w 2012 roku [30]. Dzięki dodaniu statycznego typowania, podczas pisania kodu edytor może podpowiadać programiście fragmenty kodu oraz wykrywać błędy przed uruchomieniem aplikacji.

Żeby kod napisany w TypeScript mógł być uruchomiony w przeglądarce lub procesie Node.js, musi zostać najpierw skompilowany do kodu JavaScript przez kompilator (ang. *TypeScript Compiler*) [31] według następujących kroków:

1. Zamiana kodu TypeScript na drzewo składniowe (ang. *AST – Abstract Syntax Tree*),
2. Sprawdzenie drzewa AST pod kątem poprawności typowania,
3. Zamiana drzewa AST na kod JavaScript.

Rysunek 32 przedstawia przykładowy błąd w języku TypeScript ukazany w edytorze Visual Studio Code.



```
1 let a: number = 3;
2 let b: number[] = [4];
3
4 console.log(a + b); Operator '+' cannot be applied to types 'number' and 'number[]'.
```

Rysunek 32 Przykład błędu TypeScript w edytorze Visual Studio Code. Źródło: Opracowanie własne

Listing 4 przedstawia przykładową metodę i jej wywołanie w języku TypeScript z użyciem statycznego typowania

Listing 4 Przykładowy program napisany w TypeScript. Źródło: Opracowanie własne.

```
function checkNumbers(maxNumber: number): void {
  for (let i: number = 1; i <= maxNumber; i++) {
    if (i % 2 === 0) {
      console.log(`${i} is an even number.`);
    } else {
      console.log(`${i} is an odd number.`);
    }
  }
}

checkNumbers(10);
```

4.2.2 React

React to jedna z najpopularniejszych bibliotek JavaScript do tworzenia szybkich i reaktywnych interfejsów użytkownika. Według [32] została ona utworzona w 2011 przez Jordana Walke, inżyniera oprogramowania Facebooka, na potrzeby nowego kanału informacji.

Na konferencji JSConf 2013 React stał się projektem otwartoźródłowym i od tego momentu zyskał sympatię społeczności. Duże firmy, takie jak Uber, Twitter (obecnie X), Airbnb, Netflix przyjęły bibliotekę React do swoich serwisów, co przyczyniło się do rozwoju projektu [32].

Aplikacje w React tworzone są za pomocą pojedynczych, reużywalnych elementów interfejsu, zwanych komponentami [33]. Każdy komponent jest opisany jako pojedyncza funkcja zwracająca znacznik HTML. Umożliwia to JSX (JavaScript XML) – rozszerzenie składni języka JavaScript, które umożliwia pisanie kodu przypominającego HTMLa w JS. Listing 5 przedstawia przykładowy komponent Video w React.

Listing 5 Przykładowy komponent React. Źródło: [33]

```
function Video({ video }) {
  return (
    <div>
      <Thumbnail video={video} />
      <a href={video.url}>
        <h3>{video.title}</h3>
        <p>{video.description}</p>
      </a>
      <LikeButton video={video} />
    </div>
  );
}
```

Oprócz mechanizmu komponentów React oferuje wiele funkcji zwanych *hookami*. Najczęściej używane funkcje i ich użycie to:

- `useState` – zarządzanie stanem komponentu (ang. *state*); każda zmiana stanu powoduje na nowo renderowanie komponentu,
- `useEffect` – synchronizację danych komponentu z zewnętrznymi systemami,
- `useContext` – przekazywanie danych między komponentami,

- `useRef` – trzymanie informacji, które nie są potrzebne do renderowania.

Listing 6 przedstawia przykład komponentu React z użyciem hooków `useState` i `useEffect`. Komponent na początku pobiera z API listę użytkowników, a następnie, w zależności od odpowiedzi z serwera, pokazuje tę listę lub informację o błędzie.

Listing 6 Przykładowy komponent React korzystający z hooków. Źródło: Opracowanie własne.

```
import React, { useState, useEffect } from 'react';

function UserList() {
  const [users, setUsers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await
fetch('https://jsonplaceholder.typicode.com/users');
        if (!response.ok) {
          throw new Error('Błąd sieci!');
        }
        const data = await response.json();
        setUsers(data);
      } catch (err) {
        setError(err.message);
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, []);

  if (loading) return <p>Ładowanie danych...</p>;
  if (error) return <p>Wystąpił błąd: {error}</p>;

  return (
    <div>
      <h3>Lista użytkowników</h3>
      <ul>
        {users.map(user => (
          <li key={user.id}>{user.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default UserList;
```

4.2.3 Next.js

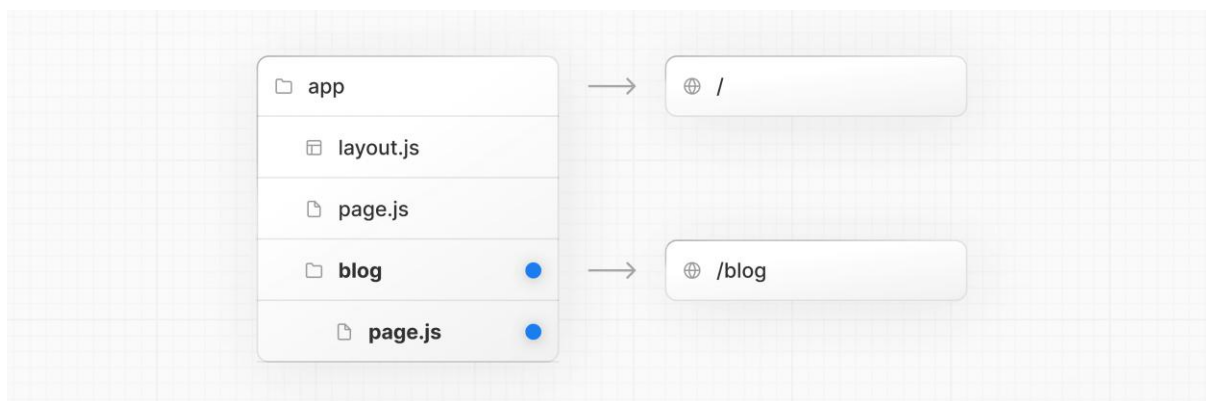
Sama biblioteka React nie jest wystarczająca do pełnowymiarowej obsługi serwisu. Żeby nie budować od podstaw brakującej obsługi routingu, struktury czy optymalizacji wydajności, korzysta się z frameworków, co przyspiesza tworzenie aplikacji.

Jednym z najczęściej używanych i polecanych przez autorów React'a frameworków jest Next.js. Dostarcza on wiele gotowych rozwiązań, takich jak [34]:

- Zaawansowany routing za pomocą struktury plików i folderów,
- Oprogramowanie pośredniczące (ang. *middleware*),
- Wsparcie dla CSS i biblioteki Sass,
- Wbudowane narzędzia do łączenia modułów JS i CSS (bundlery) jak Turbopack i Webpack,
- Renderowanie stron i komponentów po stronie klienta i serwera,
- Optymalizacje obrazów, czcionek i skryptów.

Na potrzeby pracy autorzy zdecydowali się na użycie Next.js w wersji 15.

Według dokumentacji [35] w Next.js stronę główną aplikacji definiujemy w pliku *page.tsx* w katalogu *app*, a elementy współdzielone, takie jak nagłówek, menu czy stopka, umieszczamy w pliku *layout.tsx*. Kolejne strony dodajemy tworząc nowe katalogi z odpowiednią nazwą i umieszczając w nim osobne pliki *page.tsx*. Rysunek 33 przedstawia przykładową strukturę składającą się ze strony głównej / oraz podstrony /blog.



Rysunek 33 Przykładowa struktura strony w Next.js. Źródło: [35]

Listing 7 prezentuje kod podstawowego layoutu aplikacji w Next.js, a Listing 8 przykładową stronę renderowaną po stronie serwera (ang. *Server-Side Rendering*).

Listing 7 Przykładowy plik *layout.tsx* w Next.js. Źródło: [35]

```
app/layout.tsx
export default function DashboardLayout({
  children,
}): {
  children: React.ReactNode
}) {
  return (
    <html lang="en">
```

```

    <body>
      { /* Layout UI */ }
      { /* Place children where you want to render a page or nested
layout */ }
      <main>{children}</main>
    </body>
  </html>
)
}

```

Listing 8 Przykładowa strona w Next.js. Źródło: [35]

```

app/blog/page.tsx
// Dummy imports
import { getPosts } from '@lib/posts'
import { Post } from '@ui/post'

export default async function Page() {
  const posts = await getPosts()

  return (
    <ul>
      {posts.map((post) => (
        <Post key={post.id} post={post} />
      ))}
    </ul>
  )
}

```

4.2.4 Bootstrap

Bootstrap to jeden z najpopularniejszych otwartoźródłowych frameworków opartych na HTML, CSS i JavaScript. Powstał w 2010 roku przez dwóch inżynierów Twittera (obecnie X) i jest rozwijany przez niewielki zespół programistów na GitHub [36].

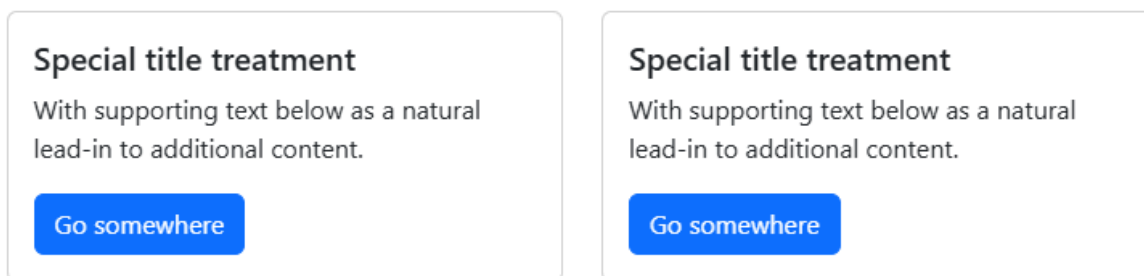
Framework dostarcza gotowy zestaw narzędzi i komponentów UI, które tworzą nowoczesny i przejrzysty język wizualny. Znacząco przyspiesza to proces tworzenia i projektowania aplikacji webowej. Oferowane są między innymi:

- Rozbudowane komponenty: karty, pola formularza, alerty;
- Interaktywne elementy: okna dialogowe (modale), karuzele, zakładki, rozwijane listy, podpowiedzi (ang. *tooltip*);
- System layoutu w siatkę – pozwala na proste dostosowanie układu elementów;
- Klasy narzędziowe (ang. *utility classes*) – umożliwiają szybkie i usystematyzowane stylowanie elementów (np. *rounded* dla zaokrąglenia rogów, *mt-2* dla marginesu górnego, *shadow* dla dodania cienia);
- Wysoka responsywność za pomocą *breakpoints* – ułatwiają kontrolę nad stylem elementów w zależności od wielkości ekranu;

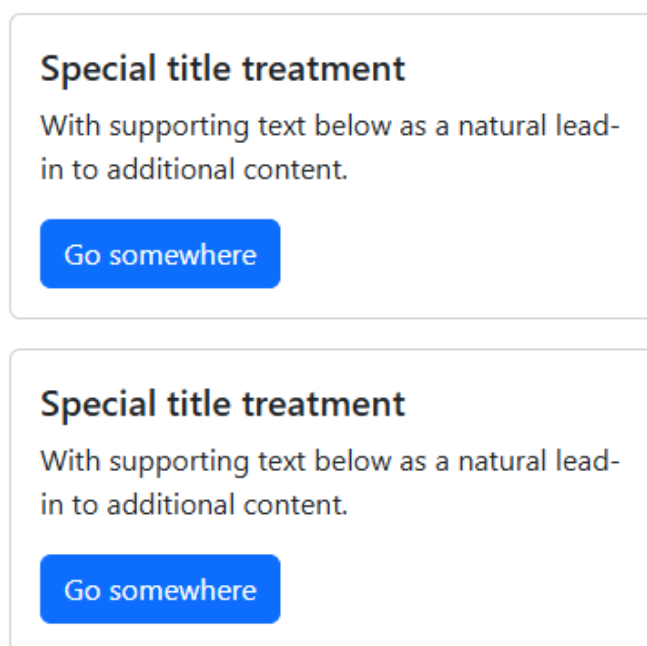
Listing 9 przedstawia przykład ułożenia dwóch kart z nagłówkiem, tekstem oraz przyciskiem. Dzięki zastosowaniu breakpointów oraz systemu siatki, karty są ułożone w poziomie na ekranie komputera, co przedstawia Rysunek 34. Rysunek 35 przedstawia ułożenie kart w pionie na ekranie telefonu.

Listing 9 Przykładowy układ dwóch komponentów Bootstrap. Źródło: [37]

```
<div class="row">
  <div class="col-sm-6 mb-3 mb-sm-0">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Special title treatment</h5>
        <p class="card-text">With supporting text below as a natural
lead-in to additional content.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
    </div>
  </div>
  <div class="col-sm-6">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Special title treatment</h5>
        <p class="card-text">With supporting text below as a natural
lead-in to additional content.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
    </div>
  </div>
</div>
```



Rysunek 34 Przykładowy widok komponentów Bootstrap na ekranie komputera. Źródło: [37]



Rysunek 35 Przykładowy widok komponentów Bootstrap na ekranie telefonu. Źródło: [37]

Autorzy zdecydowali się na użycie biblioteki React Bootstrap w wersji 2.10.10, która zawiera przepisane komponenty Bootstrap’a 5 na język React’a. Upraszcza to korzystanie z Bootstrapa poprzez zmniejszenie ilości rozwlekłego, szablonowego kodu (ang. *boilerplate*) [38]. Listing 10 oraz Listing 11 przedstawiają implementację tego samego komponentu Alert w React Bootstrap i Bootstrap.

Listing 10 Implementacja komponentu Alert w React Bootstrap. Źródło: [38]

```
import * as React from 'react';
import Alert from 'react-bootstrap/Alert';

function Example() {
  return (
    <Alert dismissible variant="danger">
      <Alert.Heading>Oh snap! You got an error!</Alert.Heading>
      <p>Change this and that and try again.</p>
    </Alert>
  );
}
```

Listing 11 Implementacja komponentu Alert w Bootstrap. Źródło: [38]

```
import * as React from 'react';

function Example() {
  return (
    <div class="alert alert-danger alert-dismissible fade show"
    role="alert">
      <strong>Oh snap! You got an error!</strong>
      <p>Change this and that and try again.</p>
      <button
        type="button"
        class="close"
        data-dismiss="alert"
        aria-label="Close"
      >
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
  );
}
```

Na przykładzie powyżej można zauważyć, że w Bootstrap Alert z możliwością schowania wymaga dopisania ręcznie przycisku. W React Bootstrap dzieje się to automatycznie poprzez dodanie atrybutu `dismissable` do komponentu.

4.2.5 Markdown

Markdown to lekki język znaczników (ang. *markup language*) służący do formatowania tekstu. Został utworzony przez Johna Grubera w 2004 roku i obecnie jest to jeden z najpopularniejszych języków znaczników na świecie [39].

Markdown wyróżnia się prostą składnią, która jest czytelna nawet, gdy jest prezentowana jako zwykły, niesformatowany tekst. Wspierane są takie elementy składni, jak:

- Akapity,
- Nagłówki,
- Cytaty,
- Pogrubienia i podkreślenia,
- Listy uporządkowane i nieuporządkowane,
- Linki,
- Grafiki,
- Fragmenty kodu,
- Tabele,
- Poziome linie.

Jego wysoka popularność i adaptacja skłoniła autorów do użycia tego języka w niniejszej pracy do przechowywania sformatowanego tekstu przepisu. Listing 12 przedstawia przykładowy tekst w języku Markdown.

```
# Nagłówek 1

Oto przykładowy tekst zawierający [link do strony głównej
PJATK](https://pja.edu.pl/).

---

## Nagłówek 2
Poniżej przedstawiono listę nieuporządkowaną z wybranymi stylami:
*   **Pogrubienie**
*   *Pochylenie*

### Nagłówek 3
Poniżej przedstawiono listę uporządkowaną:

1. Pierwszy element
2. Drugi element
3. Trzeci element
```

4.2.6 Tiptap

Tiptap to otwartoźródłowy *framework* edytora tekstu sformatowanego, który pozwala na stworzenie własnego edytora według modułowej architektury [40]. Został zbudowany na podstawie innej, niskopoziomowej biblioteki ProseMirror. Tiptap posiada szerokie wsparcie popularnych frameworków JavaScript, takich jak m.in. React, Next.js, Vue, Svelte, Angular, a nawet wspiera czystego JavaScripta.

Najważniejszą cechą frameworka Tiptap jest fakt, że dostarcza on wyłącznie podstawową logikę działania edytora (ang. *headless*). Za warstwę wizualną odpowiada programista, co pozwala na pełną swobodę w projektowaniu interfejsu użytkownika, choć wiąże się z większym nakładem pracy przy implementacji.

Kolejnym istotnym aspektem frameworka jest system rozszerzeń. Pozwalają one na wybranie jedynie tych funkcji edytora, które są potrzebne w projekcie. Dodatkowo Tiptap umożliwia napisanie własnego rozszerzenia, dzięki czemu możliwości implementacyjne niestandardowych funkcjonalności są bardzo rozległe.

Wyżej wymienione właściwości Tiptapa, w połączeniu z obszerną dokumentacją [40], przekonały autorów do skorzystania z tego frameworka w niniejszej pracy. Wdrożenie biblioteki pozwoliło na skomponowanie edytora przepisów z interfejsem zgodnym z frameworkiem Bootstrap oraz zaimplementowanie niestandardowych funkcjonalności za pomocą własnych rozszerzeń.

4.3. Microsoft SQL Server

Elementem odpowiedzialnym za trwałość i bezpieczeństwo przechowywanych informacji w niniejszej aplikacji jest relacyjny system zarządzania bazą danych Microsoft SQL Server (MSSQL) [41]. Decyzja o zastosowaniu modelu relacyjnego wynikała z konieczności zachowania pełnej spójności danych oraz obsługi złożonych relacji pomiędzy encjami w systemie.

Microsoft SQL Server to wydajny i zoptymalizowany pod względem przetwarzania dużych ilości informacji system zarządzania bazą danych. Do bezpośredniej komunikacji z MSSQL wykorzystuje się

język Transact-SQL, nazywany też T-SQL. SQL Server dostarcza zaawansowane rozwiązania administracyjne, takie jak mechanizmy indeksowania minimalizujące czas odczytu, czy opcje modyfikacji konfiguracji transakcji w bazie. Oferuje on także możliwość szeroko pojętego zarządzania polityką bezpieczeństwa poprzez użytkowników, role i uprawnienia.

Listing 13 przedstawia przykładową kwerendę w języku T-SQL realizującą operację łączenia i agregacji danych na encjach przepisów, użytkowników, ocen oraz komentarzy.

Listing 13 Przykładowa kwerenda w języku T-SQL. Źródło: Opracowanie własne.

```
SELECT TOP 5
    r.Title AS RecipeName,
    u.Username AS AuthorName,
    ROUND(AVG(CAST(rat.Rating AS FLOAT)), 2) AS AverageRating,
    COUNT(DISTINCT c.Id) AS CommentsCount
FROM Recipes r
INNER JOIN Users u ON r.AuthorId = u.Id
LEFT JOIN RecipeRatings rat ON r.RecipeId = rat.RecipeId
LEFT JOIN Comments c ON r.RecipeId = c.RecipeId
WHERE r.Status = 'Approved'
GROUP BY r.RecipeId, r.Title, u.Username
HAVING AVG(CAST(rat.Rating AS FLOAT)) >= 4.0
ORDER BY AverageRating DESC;
```

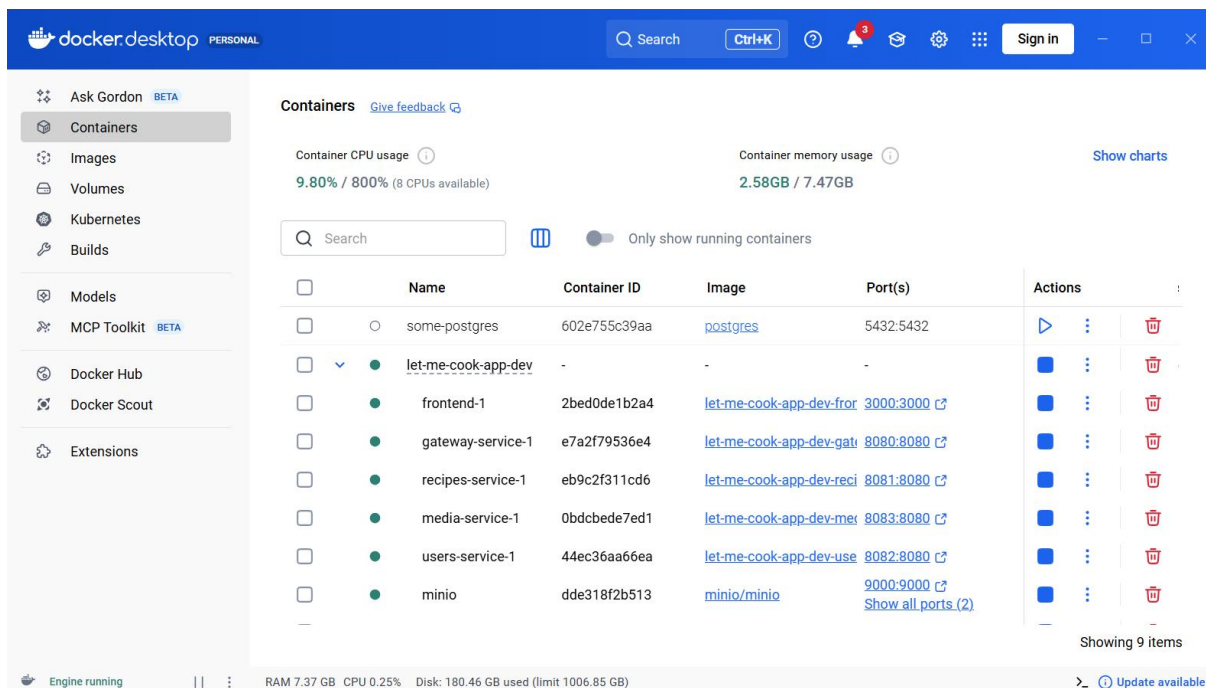
W opisywanej aplikacji wszystkie operacje na danych oraz zarządzanie ich strukturą odbywały się za pośrednictwem opisywanego wcześniej narzędzia Entity Framework Core, dlatego rola Microsoft SQL Server ograniczała się jedynie do egzekwowania zapytań generowanych przez ORM oraz dbania o spójność zapisów. Zgodnie z przyjętymi założeniami architektonicznymi baza danych MSSQL w niniejszym systemie została wydzielona jako niezależny kontener w Dockerze.

4.4. Docker

W trakcie tworzenia i wdrażania aplikacji składających się z wielu modułów jednym z wyzwań jest zapewnienie ich jednakowego, bezawaryjnego działania na różnych maszynach i systemach operacyjnych. W celu wyeliminowania problemów wynikających z różnic w konfiguracji systemowo-sprzętowej w niniejszym projekcie zdecydowano się na użycie platformy Docker [42].

Docker jest otwartoźródłowym narzędziem umożliwiającym uruchamianie aplikacji w odizolowanych środowiskach nazywanych kontenerami. W przeciwieństwie do maszyn wirtualnych nie wymagają one instalacji odrębnego systemu operacyjnego i współdzielą jądro systemu hosta. Dzięki takiemu działaniu jest to rozwiązanie bardzo oszczędne pod kątem zasobów sprzętowych i gotowe do działania w kilka sekund.

Rysunek 36 przedstawia widok interfejsu graficznego narzędzia Docker Desktop z działającym projektem Docker Compose `let-me-cook-app-dev` zawierającym siedem kontenerów.



Rysunek 36 Widok interfejsu graficznego narzędzia Docker Desktop. Źródło: Opracowanie własne.

4.4.1 Obrazy Dockera

Obraz Dockera (ang. *Docker image*) można opisać jako przeznaczona tylko do odczytu paczka oprogramowania zawierająca wszystkie komponenty umożliwiające poprawne uruchomienie i funkcjonowanie danego systemu. Mieści on w sobie takie elementy jak skompilowany kod źródłowy, środowisko uruchomieniowe, zależności systemowe oraz zmienne środowiskowe. Proces generowania obrazu Dockera opisany jest zawsze za pomocą specjalnych instrukcji zapisanych w pliku Dockerfile.

4.4.2 Kontenery

Kontener w Dockerze to instancja danego systemu wygenerowana na podstawie bazowego obrazu. Funkcjonuje on jako odizolowany proces z własnym interfejsem sieciowym oraz dedykowaną strukturą plików. Dzięki takiemu działaniu kontenery są całkowicie niezależne od systemu hosta oraz innych, działających jednocześnie modułów.

4.4.3 Docker Compose

W aplikacjach składających się z wielu modułów ręczne wdrażanie każdego kontenera osobno byłoby czasochłonne i w większym stopniu narażałoby system na błędy. Z tego względu w niniejszym projekcie zdecydowano się na użycie narzędzia Docker Compose, które pozwala na opisanie struktury całej infrastruktury w specjalnym pliku `docker-compose.yml` rozpoznawanym przez Dockera. Dzięki temu zintegrowanie i uruchomienie wszystkich modułów aplikacji jest w pełni zautomatyzowane i ogranicza się do wywołania tylko jednej komendy w wierszu poleceń.

4.5. Git oraz GitHub

W każdym projekcie informatycznym bardzo ważne jest odpowiednie zarządzanie kodem źródłowym. Mając na celu pełną kontrolę nad cyklem życia kodu, jego właściwe wersjonowanie oraz śledzenie wprowadzanych zmian, w niniejszym projekcie, autorzy zdecydowali się zastosować system kontroli wersji Git [43] wraz ze zdalnym repozytorium na platformie GitHub [44].

4.5.1 System kontroli wersji Git

Git jest otwartoźródłowym rozproszonym systemem kontroli wersji umożliwiającym precyzyjne śledzenie zmian w strukturze projektu. Narzędzie to zostało stworzone w 2005 roku przez Linusa Torvaldsa, pierwotnie tylko na potrzeby rozwoju jądra systemu operacyjnego Linux, a obecnie jest standardem na rynku. W odróżnieniu od rozwiązań scentralizowanych Git tworzy kopię całego repozytorium wraz z pełną historią modyfikacji bezpośrednio na komputerze programisty, umożliwiając przy tym pracę bez stałego dostępu do sieci i przyspieszając operacje na plikach.

Kluczową funkcjonalnością wykorzystywaną w trakcie implementacji systemu było tworzenie niezależnych gałęzi rozwoju kodu (ang. *branching*). Pisanie nowego kodu odbywało się na odizolowanych gałęziach roboczych dedykowanych pojedynczym funkcjonalnościom. Dopiero po testach i pomyślnej weryfikacji następowało włączenie nowego rozwiązania do głównej, stabilnej wersji systemu poprzez operację scalania (ang. *merge*), co minimalizowało ryzyko wprowadzenia do niej nowych błędów.

4.5.2 Platforma GitHub

Podczas gdy Git służy do lokalnej kontroli wersji, platforma GitHub dostarcza chmurową infrastrukturę dla zdalnego repozytorium. Zapewnia ona bezpieczną kopię zapasową kodu oraz umożliwia jego synchronizację przy pomocy poleceń `git push` oraz `git pull`.

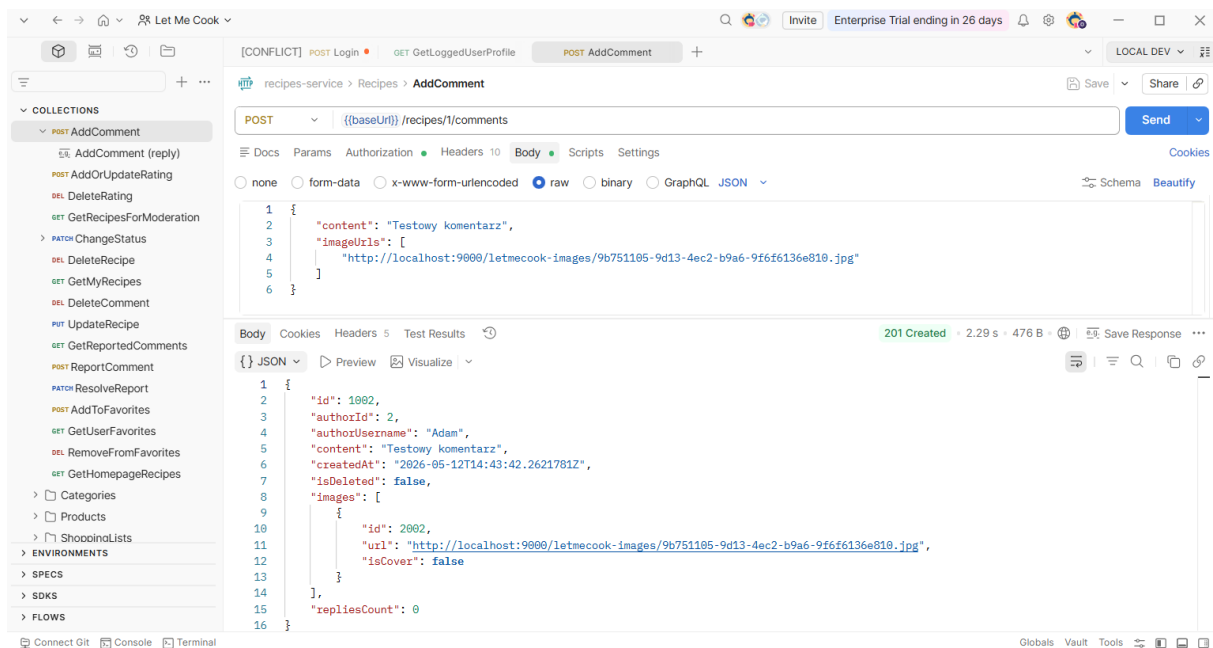
Platforma pozwoliła w zrealizowanym projekcie także na wdrożenie procesu przeglądu kodu przy użyciu mechanizmu *Pull Requests*. Dzięki temu każda nowo wdrażana funkcjonalność wymagała weryfikacji i formalnego zatwierdzenia przez innego programistę, a w razie znalezienia błędów lub innych wątpliwości umożliwiała otwarcie dyskusji, co przełożyło się bezpośrednio na wyższą jakość kodu i lepsze działanie aplikacji.

4.6. Postman

W procesie tworzenia aplikacji webowych istotna jest weryfikacja działania dostarczanych interfejsów sieciowych jeszcze przed ich wykorzystaniem w docelowej aplikacji klienckiej. Aby umożliwić testowanie punktów końcowych (ang. *endpoints*) niezależnie od interfejsu graficznego w niniejszym projekcie zastosowano narzędzie Postman [45].

Postman jest platformą służącą do projektowania i testowania interfejsów API. Jej podstawową funkcjonalnością jest możliwość tworzenia żądań HTTP, które docelowo będą inicjowane przez aplikację frontendową. Postman posiada intuicyjny interfejs graficzny umożliwiający programistom modyfikację wysyłanych pakietów danych, a w szczególności takich ich właściwości jak parametry adresu URL, nagłówki i ciało żądania, czy tokeny autoryzacyjne.

Rysunek 37 ilustruje interfejs graficzny narzędzia Postman podczas testowania punktu końcowego API za pomocą żądania HTTP typu POST.



Rysunek 37 Widok interfejsu graficznego narzędzia Postman. Źródło: Opracowanie własne.

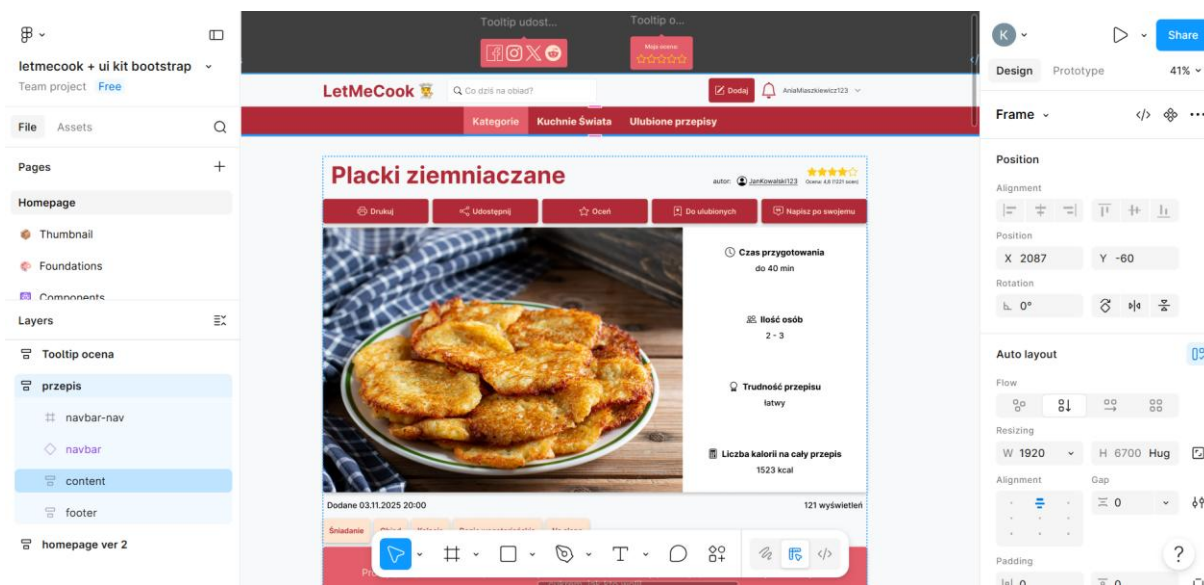
Narzędzie Postman w przedstawianym projekcie odegrało istotną rolę podczas testowania części backendowej oraz późniejszym wykorzystaniu zaimplementowanych interfejsów API w aplikacji klienckiej. Poprzez dokładną weryfikację zwracanych przez serwer odpowiedzi HTTP na konkretne żądania wykazano poprawność działania logiki biznesowej, routingu oraz mechanizmów autoryzacji i autentykacji. Pozwoliło to autorom na wdrażanie sprawdzonych uprzednio rozwiązań backendowych w docelowej aplikacji frontendowej.

4.7. Figma

Przed przystąpieniem do implementacji graficznego interfejsu aplikacji niezbędne było wstępne zaprojektowanie jego układu wizualnego. Aby zminimalizować ryzyko czasochłonnych modyfikacji w kodzie źródłowym, do stworzenia makiet docelowego systemu wykorzystano platformę Figma.

Według [46] Figma to nowoczesne, chmurowe narzędzie do projektowania interfejsu użytkownika (UI) oraz doświadczeń użytkownika (UX), które umożliwia tworzenie prototypów, makiet i pełnych projektów stron internetowych oraz aplikacji mobilnych. Modelowanie widoków w tej aplikacji opiera się na wykorzystywaniu niezależnych komponentów wielokrotnego użytku. Ponieważ do implementacji docelowego interfejsu graficznego wykorzystano technologię React, bazującą na takim samym podejściu do tworzenia warstwy wizualnej, przekształcanie makiet utworzonych w Figmie na rzeczywiste elementy frontendu było bardzo intuicyjne. Ponadto platforma umożliwia współpracę wielu użytkowników jednocześnie, co pozwoliło autorom na dzielenie się swoimi pomysłami wyglądu aplikacji w czasie rzeczywistym.

Rysunek 38 ilustruje makietę wysokiej wierności (ang. *high-fidelity mockup*) prezentującą szczegółowy widok przepisu kulinarnego w narzędziu Figma.



Rysunek 38 Projekt interfejsu graficznego w narzędziu Figma. Źródło: Opracowanie własne.

Wykorzystanie platformy Figma na etapie projektowania opisywanej aplikacji pozwoliło na ustalenie kolorystyki oraz szczegółowe zaplanowanie rozmieszczenia wszystkich elementów interfejsu użytkownika. Przygotowanie dokładnych makiet poszczególnych widoków znacznie ułatwiło dalszą implementację frontendu i zminimalizowało ilość późniejszych modyfikacji w kodzie.

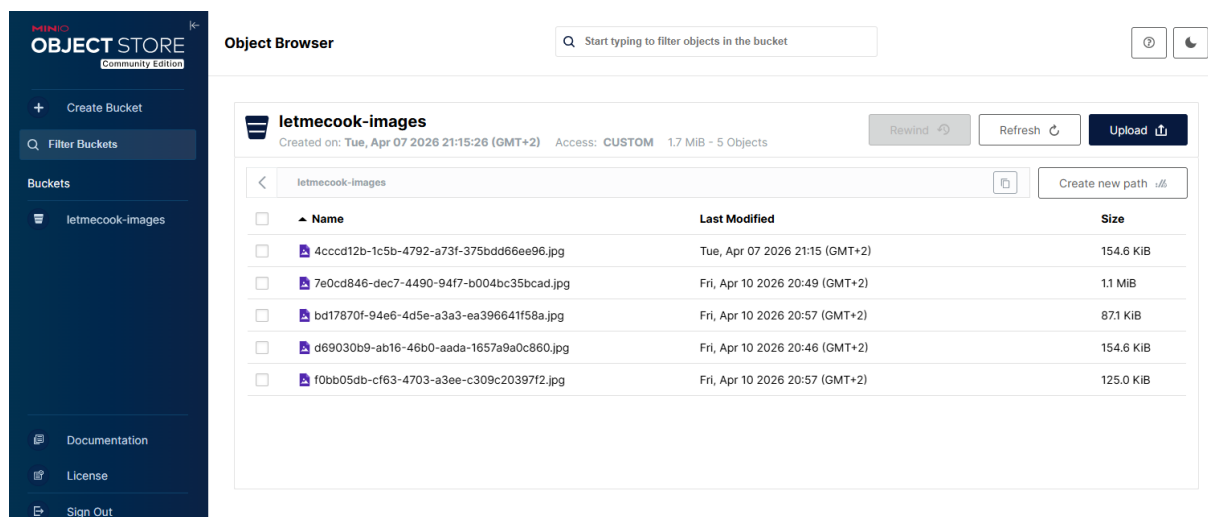
4.8. MinIO

Przechowywanie plików graficznych bezpośrednio w bazie danych znacząco obniża jej wydajność, natomiast zapis w pamięci kontenera będzie prowadził do utraty danych po jego restarcie. Rozwiązaniem tych problemów technicznych w prezentowanym projekcie było wykorzystanie usługi MinIO [47] do przechowywania multimediów.

MinIO to otwartoźródłowy serwer pełniący rolę magazynu obiektów (ang. *object storage*) o wysokiej wydajności. Jedną z głównych właściwości tego rozwiązania jest jego kompatybilność z interfejsem API popularnej usługi Amazon S3 [48]. Taka charakterystyka sprawia, że aplikacja jest przygotowana do migracji do publicznej chmury AWS [49] bez konieczności wprowadzania jakichkolwiek modyfikacji w kodzie źródłowym odpowiedzialnym za obsługę plików.

MinIO przechowuje wgrywane przez użytkowników grafiki jako niezależne obiekty w płaskiej strukturze, pogrupowane w zasobnikach (ang. *buckets*). W zrealizowanym projekcie usługa backendowa media-service waliduje przesyłane zdjęcia i nadaje im odpowiednie metadane, a następnie deleguje ich fizyczny zapis do serwera MinIO. Platforma zwraca później do serwisu jedynie ścieżkę dostępu do określonych plików, dzięki czemu relacyjna baza danych przechowuje jedynie lekki adres URL zamiast dużych danych binarnych.

Rysunek 39 przedstawia panel administracyjny serwera MinIO, w którym widać zasobnik `letmecook-images` wraz z jego zawartością.



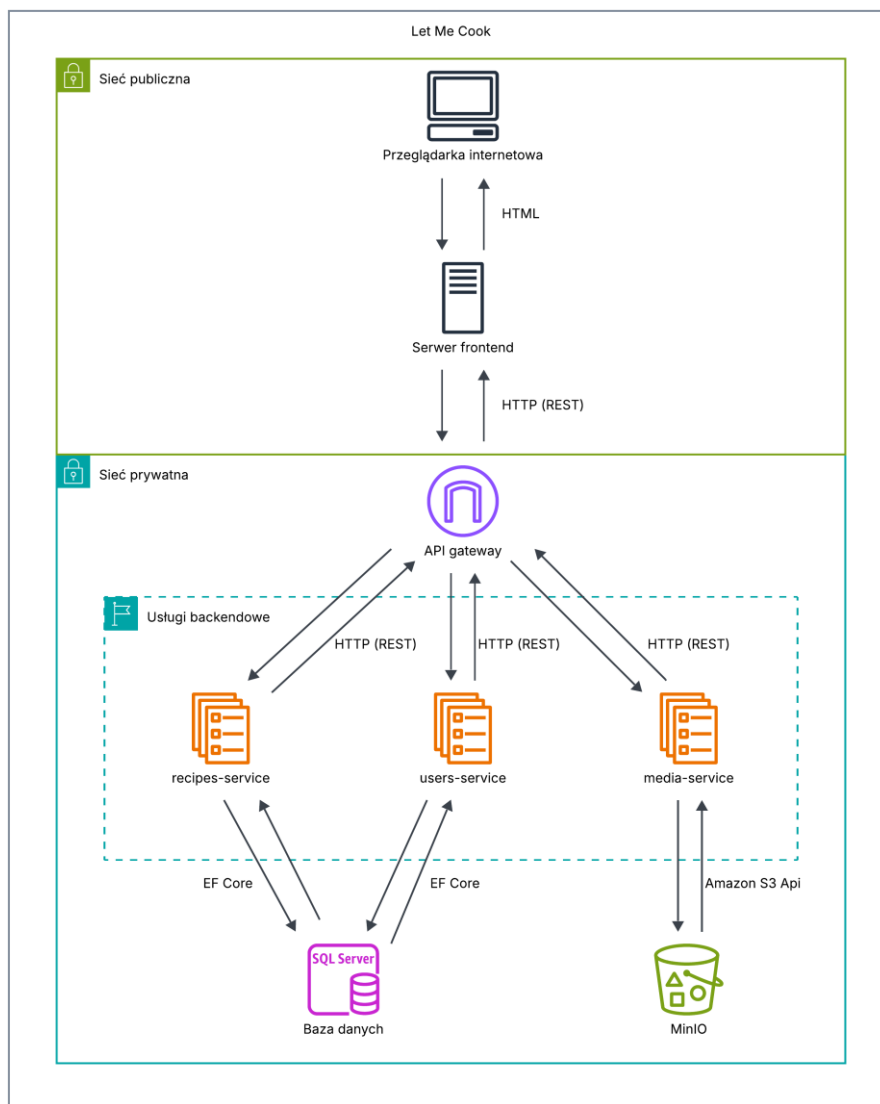
Rysunek 39 Widok panelu administracyjnego serwera MinIO. Źródło: Opracowanie własne.

Oddelegowanie fizycznego zapisu plików graficznych do MinIO pozwoliło zachować bezstanowość pozostałych usług w systemie przy jednoczesnym odciążeniu bazy danych. Przekłada się to na lepszą wydajność całej infrastruktury oraz czyni aplikację gotową na ewentualną migrację do chmury obliczeniowej.

5. Architektura i implementacja systemu

W niniejszym rozdziale szczegółowo przedstawiono architekturę oraz implementację zrealizowanego systemu. Przed rozpoczęciem faktycznego tworzenia aplikacji autorzy ustalili założenia oraz wzorce projektowe, zgodnie z którymi miała ona być budowana. Aby zapewnić jak najwyższą niezawodność oraz bezpieczeństwo infrastruktury, system podzielono na dwa główne, całkowicie odseparowane obszary – warstwę serwerową nazywaną backendem oraz webową aplikację kliencką nazywaną frontendem, komunikujące się ze sobą poprzez interfejs REST API.

Architektura i komunikacja pomiędzy modułami w systemie Let Me Cook została przedstawiona na Rysunek 40.



Rysunek 40 Schemat komunikacji w systemie Let Me Cook. Źródło: Opracowanie własne.

5.1. Implementacja części serwerowej – backend

Część serwerowa zrealizowanego systemu została stworzona w oparciu o architekturę mikroservisów. Przychodzące żądania HTTP w pierwszej kolejności odbierane są przez bramę API Gateway, która dynamicznie przekierowuje je do odpowiednich usług. Pomimo wyraźnego podziału systemu na moduły, kod definiujący model danych wyodrębniony został do projektu współdzielonego przez mikroservisy potrzebujące dostępu do systemu bazodanowego.

W celu zapewnienia jak najwyższej jakości kodu oraz ułatwienia jego ewentualnej rozbudowy, podczas implementacji backendu korzystano ze znanych i sprawdzonych rozwiązań programistycznych. Zastosowanie wzorców projektowych pozwoliło na zminimalizowanie zależności pomiędzy poszczególnymi komponentami, ustandaryzowanie struktury kodu oraz zwiększenie czytelności logiki biznesowej.

Za komunikację aplikacji z bazą danych odpowiada narzędzie Entity Framework Core. Zamiast pisania bezpośrednich zapytań SQL, zastosowano ten ORM w celu zapewnienia czystości kodu oraz aby wytwarzany kod był zgodny z nowoczesnymi standardami wytwarzania oprogramowania.

W opisywanym projekcie zabezpieczenia oparte są na wydawanych tokenach dostępu (ang. *access token*) oraz tokenach odświeżających (ang. *refresh token*) w standardzie JSON Web Token (JWT). Punktem sprawdzającym poprawność i ważność przychodzących tokenów jest brama API Gateway. Takie rozwiązanie zdejmuje odpowiedzialność kontrolowania ich prawidłowości z docelowych usług, pozostawiając im wyłącznie weryfikację, czy dany użytkownik posiada uprawnienia do konkretnych zasobów.

5.1.1 Zastosowane wzorce projektowe

Stosowanie wzorców projektowych jest bardzo istotną kwestią w kontekście tworzenia i utrzymania rozbudowanych systemów. W niniejszym rozdziale opisano kluczowe wzorce, które ukształtowały strukturę mikroservisów w omawianym projekcie.

5.1.1.1 Wstrzykiwanie zależności (Dependency Injection)

Wstrzykiwanie zależności (ang. *Dependency Injection*) to wzorec projektowy, który znacznie ułatwia zarządzanie kodem aplikacji. Wdrożenie tego wzorca ma na celu osiągnięcie inwersji kontroli (ang. *inversion of control*), czyli jednej z dobrych praktyk programistycznych tworzących zbiór zasad SOLID. Dzięki takiemu podejściu zmniejsza się powiązanie (ang. *coupling*) pomiędzy różnymi komponentami systemu, co jest pożądane w inżynierii oprogramowania.

W opisywanej aplikacji za zarządzanie cyklem życia obiektów w ramach wstrzykiwania zależności odpowiada wbudowany mechanizm platformy ASP .NET Core. Usługi, które mają być dostępne do wstrzyknięcia dla innych komponentów w systemie należy najpierw zarejestrować w klasie `Program.cs` danego mikroservis i określić ich cykl życia. Domyślnie w .NET dostępne są trzy warianty czasu życia obiektów:

- Zasięgowy (ang. *scoped*) – obiekt tworzony jest specjalnie dla każdego żądania HTTP i niszczone po jego przetworzeniu. Jest to najlepsze rozwiązanie w przypadku serwisów biznesowych i połączeń z bazą danych, ponieważ gwarantuje zachowanie jednego stanu obiektu w ramach żądania.
- Chwilowy (ang. *transient*) – obiekt tworzony jest za każdym razem, gdy aplikacja zgłosi na niego zapotrzebowanie i od razu potem usuwany. Wykorzystywany w przypadku lekkich usług, które nie muszą przechowywać żadnych danych.
- Jednorazowy (ang. *singleton*) – obiekt tworzony jest przy starcie aplikacji i istnieje przez cały czas jej działania. Sprawdza się najlepiej w przypadku globalnych mechanizmów i konfiguracji.

Listing 14 przedstawia rejestrację zależności w głównym pliku konfiguracyjnym mikroserwisu UsersService. Fragment kodu ukazuje operacje kojarzenia interfejsów z implementującymi je klasami wykonawczymi.

Listing 14 Rejestracja zależności w pliku Program.cs w mikroserwisie UsersService. Źródło: Opracowanie własne.

```
builder.Services
    .AddDbContext<AppDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("Default")
))
    .AddScoped<IAuthService, AuthService>()
    .AddScoped<ITokenService, TokenService>()
    .AddScoped<IUsersService, UsersService.Services.UsersService>();
```

5.1.1.2 Data Transfer Object (DTO)

Komunikacja pomiędzy aplikacją kliencką a mikroserwisami realizowana jest przy użyciu obiektów transferu danych (ang. *Data Transfer Object*) nazywanych DTO [50]. Zastosowanie tego wzorca ma na celu zamaskowanie fizycznej struktury bazy danych i umożliwienie przesyłania informacji wyłącznie niezbędnych w kontekście biznesowym.

W Listing 15 przedstawiono klasę PublicUserDto, która została utworzona w celu bezpiecznego przesyłania informacji o profilu użytkownika. Zwracanie całego rekordu użytkownika z bazy danych skutkowałoby ujawnieniem m.in. zaszyfrowanego hasła, czy tokena odświeżania, co byłoby niedopuszczalne. Zamiast tego system tworzy na podstawie danych pobranych z bazy specjalny obiekt wyjściowy, zawierający atrybuty bezpieczne do upublicznienia i pożądane przez konkretny widok frontendu.

Listing 15 Obiekt transferu danych wykorzystywany do zwracania informacji o profilu użytkownika. Źródło: Opracowanie własne.

```
namespace UsersService.DTOs;

public class PublicUserDto
{
    public int Id { get; set; }
    public required string Username { get; set; }
    public string? Bio { get; set; }
    public string? AvatarUrl { get; set; }
    public DateTime CreatedAt { get; set; }
    public int FollowersCount { get; set; }
    public int FollowingCount { get; set; }
    public bool IsFollowedByCurrentUser { get; set; }
    public bool IsBanned { get; set; }
}
```

5.1.1.3 Wzorzec warstwy serwisów

Architektura mikroservisów opiera się na wzorcu warstwy serwisów (ang. *service layer pattern*). Według [51] warstwa serwisów to architektoniczny wzorzec definiujący zbiór serwisów obsługujących logikę biznesową i pełniących rolę pośrednika pomiędzy różnymi warstwami aplikacji – w szczególności pomiędzy kontrolerem (lub warstwą API) a warstwą dostępu do danych (lub warstwą repozytorium).

W opisywanym projekcie zdecydowano się skorzystać z wzorca warstwy serwisów w celu odseparowania operacji biznesowych od obsługi żądań w kontrolerach. Zamiast tworzyć nowe instance serwisów w kontrolerach, interfejsy serwisów wstrzykiwane są do ich konstruktorów, co zapewnia luźne powiązanie (ang. *loose coupling*) komponentów. Z kolei serwisy komunikujące się z bazą danych w swoich konstruktorach przyjmują referencję do `AppDbContext`, co pozwala im na bezpośrednie operacje na danych z wykorzystaniem języka zapytań LINQ.

Listing 16 przedstawia fragment kontrolera `UserController` do którego wstrzykiwany jest w konstruktorze interfejs `IUsersService`, a następnie w metodzie kontrolera `UnfollowUser()` wywoływana jest metoda serwisu `UnfollowUserAsync()`. Z kolei w Listing 17 znajduje się fragment klasy `UserService` należącej do warstwy serwisów, przyjmującej w swoim konstruktorze klasę kontekstu bazodanowego `AppDbContext` i wykonującej operację usunięcia z bazy danych wybranej obserwacji użytkownika w metodzie `UnfollowUserAsync()`.

Listing 16 Fragment kontrolera `UserController` przedstawiający wykorzystanie warstwy serwisów. Źródło:

Opracowanie własne.

```
// usings
namespace UsersService.Controllers;

[ApiController]
[Route("api/users")]
public class UsersController : ControllerBase
{
    private readonly IUsersService _userService;
    private readonly ILogger<UserController> _logger;

    public UsersController(IUsersService userService,
        ILogger<UserController> logger)
    {
        _userService = userService;
        _logger = logger;
    }

    // other controller methods

    [HttpDelete("{id}/follow")]
    public async Task<IActionResult> UnfollowUser(int id,
        [FromHeader(Name = "X-User-Id")] int currentUserId)
    {
        _logger.LogInformation("Received request: User {FollowerId} wants
        to unfollow User {FollowedId}.", currentUserId, id);

        try
```

```

        {
            await _userService.UnfollowUserAsync(currentUserId, id);

            _logger.LogInformation("Request completed. User {FollowerId}
successfully unfollowed User {FollowedId}.", currentUserId, id);
            return Ok();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An unexpected error occurred while User
{FollowerId} tried to unfollow User {FollowedId}.", currentUserId, id);
        }
        return StatusCode(500, new { Message = "Internal server
error." });
    }
}

```

Listing 17 Fragment klasy UsersService należącej do warstwy serwisów. Źródło: Opracowanie własne.

```

// usings

namespace UsersService.Services;

public class UsersService : IUsersService
{
    private readonly AppDbContext _context;
    private readonly ILogger<UsersService> _logger;

    public UsersService(AppDbContext context, ILogger<UsersService>
logger)
    {
        _context = context;
        _logger = logger;
    }
    public async Task UnfollowUserAsync(int followerId, int followedId)
    {
        _logger.LogDebug("Processing unfollow logic. Follower:
{FollowerId}, Target: {FollowedId}.", followerId, followedId);

        var follow = await _context.UserFollows
            .FirstOrDefaultAsync(uf => uf.FollowerId == followerId &&
uf.FollowedId == followedId);

        if (follow == null)
        {
            _logger.LogDebug("Unfollow skipped. Relation does not exist:
{FollowerId} -> {FollowedId}.", followerId, followedId);
            return;
        }

        _context.UserFollows.Remove(follow);
        await _context.SaveChangesAsync();
    }
}

```

```

        _logger.LogInformation("Follow relation removed from DB: User
{FollowerId} unfollowed User {FollowedId}.", followerId, followedId);
    }
}

```

5.1.2 Architektura mikroservisów

Zastosowanie architektury mikroservisowej w warstwie backendowej opisywanego systemu miało na celu podwyższenie jego skalowalności oraz logiczne rozdzielanie odpowiedzialności na różne komponenty. Dzięki takiemu podejściu przeciążenie pojedynczego modułu lub błąd krytyczny nie paraliżuje działania całej aplikacji. Projektowanie aplikacji w tym modelu niesie jednak ze sobą konieczność wytyczenia jasnych granic pomiędzy modułami, określenia punktów wejścia dla zewnętrznych żądań oraz wypracowania kompromisu w kwestii składowania danych wykorzystywanych przez różne usługi, co zostało szczegółowo opisane w kolejnych podrozdziałach.

Każdy z komponentów, niebędących jedynie biblioteką klas, posiada w systemie swój plik konfiguracyjny `Program.cs`, na podstawie którego jest uruchamiany. W zależności od modułu skonfigurowane są w nim m.in. następujące elementy:

- rejestracja zależności,
- rejestracja kontrolerów API,
- połączenie z bazą danych
- połączenie z usługą Amazon S3,
- logowanie zdarzeń,
- inicjowanie bazy danych,
- działanie Ocelota,
- autoryzacja i uwierzytelnianie,
- polityki CORS.

Listing 18 przedstawia konfigurację mikroservisu `RecipesService` zawartą w pliku `Program.cs`

Listing 18 Zawartość pliku `Program.cs` w mikroservisie `RecipesService`. Źródło: Opracowanie własne.

```

// ...usings

var builder = WebApplication.CreateBuilder(args);

builder.Host.UseSerilog((context, services, configuration) =>
{
    var connectionString =
context.Configuration.GetConnectionString("Default");

    configuration
        .MinimumLevel.Information()
        .MinimumLevel.Override("Microsoft", LogEventLevel.Warning)
        .MinimumLevel.Override("System", LogEventLevel.Warning)
        .Enrich.FromLogContext()
        .Enrich.WithProperty("ServiceName", "RecipesService")

```

```

        .WriteTo.Console();

    if (!string.IsNullOrEmpty(connectionString))
    {
        var sinkOptions = new MSSqlServerSinkOptions
        {
            TableName = "Logs",
            AutoCreateSqlTable = true
        };

        var columnOptions = new ColumnOptions();

        columnOptions.AdditionalColumns = new Collection<SqlColumn>
        {
            new SqlColumn
            {
                ColumnName = "ServiceName",
                DataType = SqlDbType.NVarChar,
                DataLength = 50
            }
        };

        configuration.WriteTo.MSSqlServer(
            connectionString: connectionString,
            sinkOptions: sinkOptions,
            columnOptions: columnOptions,
            restrictedToMinimumLevel: LogEventLevel.Warning
        );
    }
});

builder.Services.AddControllers();

builder.Services
    .AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("Default")
))
    .AddScoped<IRecipesService, RecipesService.Services.RecipesService>()
    .AddScoped<ICategoriesService, CategoriesService>()
    .AddScoped<IProductsService, ProductsService>()
    .AddScoped<IShoppingListsService, ShoppingListsService>();

var app = builder.Build();

app.UseSerilogRequestLogging();
app.MapControllers();
app.Run();

```

5.1.2.1 Podział odpowiedzialności w usługach backendowych

Warstwa serwerowa aplikacji została podzielona na trzy odseparowane od siebie mikroserwisy. Każdy z nich odpowiada za ściśle określoną część logiki biznesowej i realizuje specyficzne dla niej operacje.

Do zdefiniowania punktów końcowych w poniższych mikroserwisach użyto kontrolerów. Klasy te odpowiadają za obsługę parametrów żądań, autoryzację, logowanie zdarzeń, wywoływanie odpowiednich metod z warstwy dostępu do danych oraz zwracanie odpowiednich kodów HTTP. Przykładowy kod kontrolera przedstawiono wcześniej w Listing 2.

5.1.2.1.1. Mikroservis przepisów (RecipeService)

Mikroservis przepisów jest najbardziej rozbudowanym modulem w systemie, odpowiedzialnym za realizację większości logiki biznesowej aplikacji. To właśnie w nim zachodzą wszystkie operacje związane z:

- przepisami,
- produktami,
- kategoriami przepisów,
- listami zakupów.

Aby zapewnić czystość kodu w ramach tego mikroservisu został on podzielony zgodnie z powyższym wypunktowaniem. Podział ten ma swoje odzwierciedlenie w warstwie prezentacji i dostępu do danych – dla każdego zasobu utworzono osobny kontroler oraz wstrzykiwany do kontrolera serwis wykonujący operacje na danych.

Dokładny podział odpowiedzialności w mikroserwisie RecipeService przedstawia się następująco:

- **RecipeController** oraz **IRecipeService** – odpowiadają za wszystkie operacje na przepisach kulinarnych takie jak ich tworzenie, odczyt, edycja i usuwanie, za interakcje użytkowników z przepisami – komentarze, oceny czy dodawanie do ulubionych, a także za operacje dostępne tylko dla moderacji takie jak specjalne zwracanie przepisów i zgłoszeń komentarzy wraz z możliwością ich procesowania dla panelu administracyjnego.
- **ProductsController** oraz **IProductsService** – odpowiadają za bazę produktów w systemie – dodawanie, zwracanie, edycję, usuwanie, moderację a także pilnowanie unikalności produktów.
- **CategoriesController** oraz **ICategoriesService** – odpowiadają za zarządzanie kategoriami przepisów – ich dodawanie, zwracanie, edycję, usuwanie, a także tworzenie, aktualizację i usuwanie grup kategorii.
- **ShoppingListsController** oraz **IShoppingListsService** – odpowiadają za operacje związane z listami zakupów użytkowników – dodawanie, zwracanie i usuwanie produktów z listy oraz zmianę statusu zakupu konkretnych produktów na liście.

5.1.2.1.2. Mikroservis użytkowników (UserService)

Mikroservis użytkowników w opisywanym systemie odpowiedzialny jest za zarządzanie tożsamością, rolami oraz danymi kont użytkowników. W ramach UserService realizowane są operacje związane z:

- procesem rejestracji i logowania,
- zarządzaniem kontami i profilami użytkowników,

- zarządzaniem rolami użytkowników.

Mikroserwis użytkowników został podzielony na dwie główne części dzieląc jego odpowiedzialności na te związane z logowaniem i rejestracją oraz bardziej biznesowe – związane z kontami, obserwacjami i rolami użytkowników. Szczegółowo podział ten przedstawia się następująco:

- **AuthController** oraz **IAuthService** – obsługują bezpieczne logowanie, rejestrację, wylogowanie oraz odświeżanie tokenów. **IAuthService** jest dodatkowo wspierany przez serwis **ITokenService** generujący i walidujący tokeny JWT, niewidoczny z poziomu kontrolera.
- **UsersController** oraz **IUsersService** – odpowiadają za zarządzanie profilami i uprawnieniami użytkowników. Umożliwiają użytkownikom modyfikację biogramów, awatarów i haseł, a także obserwowanie innych użytkowników. Dodatkowo udostępniają administracji możliwość zmiany ról użytkowników oraz blokowanie ich kont w systemie.

5.1.2.1.3. Mikroserwis multimediów (MediaService)

Mikroserwis multimediów jest modulem odpowiedzialnym za zarządzanie zdjęciami. W przeciwieństwie do pozostałych usług nie komunikuje się on z bazą danych, a z obiektywą pamięcią masową MinIO. W jego obrębie realizowane są takie operacje jak:

- walidacja i przesyłanie plików na serwer,
- usuwanie plików z serwera.

Podobnie jak poprzednie mikroserwisy, **MediaService** podzielony jest na warstwę prezentacji oraz dostępu do danych, co pozwala na ewentualną bezinwazyjną zmianę sposobu przechowywania plików w przyszłości. W mikroserwisie multimediów za walidację przesyłanych plików pod kątem rozmiaru oraz ilości odpowiada **MediaController**. Komunikuje się on z serwisem **IStorageService**, który nawiązuje połączenie z serwerem MinIO i transferuje do niego zdjęcia, bądź usuwa istniejące. Serwis odpowiada też za sprawdzanie uprawnień do usuwania konkretnych zdjęć oraz generowanie linków do plików umieszczonych na serwerze.

5.1.2.2 Centralny punkt dostępu – API Gateway

Aby ujednolicić komunikację aplikacji klienckiej z rozproszonym środowiskiem backendowym zdecydowano się na wprowadzenie bramy sieciowej będącej jedynym punktem dostępu do warstwy serwerowej. Rozwiązanie to zostało wdrożone w odrębnym projekcie **GatewayService** przy użyciu biblioteki Ocelot szczegółowo opisanej w rozdziale 4.1.6. Głównym zadaniem bramy API jest przechwytywanie przychodzących żądań HTTP i rutowanie ich do właściwych usług backendowych znajdujących się w sieci prywatnej.

W celu zwiększenia czytelności kodu i ułatwienia utrzymania projektu zdecydowano się skorzystać z mechanizmu łączenia konfiguracji narzędzia Ocelot zamiast trzymać wszystkie reguły w jednym, obszernym pliku. Zasady routingu rozdzielono na mniejsze pliki odpowiadające konkretnym domenom opisywanego systemu (m.in. przepisy, produkty, użytkownicy). Dopiero podczas startu aplikacji wszystkie pliki konfiguracyjne są automatycznie scalane w jeden spójny, zrozumiały dla Ocelota plik.

Listing 19 przedstawia zawartość pliku `ocelot.shoppingLists.json` zawierającego reguły przekierowań i autoryzacji żądań HTTP dla list zakupowych w mikroserwisie przepisów.

Listing 19 Zawartość pliku ocelot definiującego trasy dla list zakupowych. Źródło: Opracowanie własne.

```
{
  "Routes": [
    {
```

```

    "UpstreamPathTemplate": "/shopping-list/{everything}",
    "UpstreamHttpMethod": [ "POST", "DELETE", "PATCH" ],
    "DownstreamPathTemplate": "/api/shopping-list/{everything}",
    "DownstreamScheme": "http",
    "DownstreamHostAndPorts": [ { "Host": "recipes-service", "Port":
8080 } ],
    "AuthenticationOptions": { "AuthenticationProviderKey":
"IdentityServiceKey", "AllowedScopes": [ ] },
    "AddHeadersToRequest": { "X-User-Id": "Claims[sub] > value" },
    "Priority": 5
  },
  {
    "UpstreamPathTemplate": "/shopping-list",
    "UpstreamHttpMethod": [ "GET" ],
    "DownstreamPathTemplate": "/api/shopping-list",
    "DownstreamScheme": "http",
    "DownstreamHostAndPorts": [ { "Host": "recipes-service", "Port":
8080 } ],
    "AuthenticationOptions": { "AuthenticationProviderKey":
"IdentityServiceKey", "AllowedScopes": [ ] },
    "AddHeadersToRequest": { "X-User-Id": "Claims[sub] > value" },
    "Priority": 1
  }
]
}

```

5.1.2.2.1. Weryfikacja tokenów i polityki CORS

Poza kierowaniem ruchem sieciowym wewnątrz warstwy serwerowej systemu, rolą bramy API jest zdjęcie z docelowych mikroserwisów obowiązku walidacji przesyłanych w przychodzących żądaniach tokenów JWT. Takie rozwiązanie gwarantuje też weryfikację kryptograficzną każdego tokena jeszcze przed wejściem żądania do sieci prywatnej. W ramach kontroli tokenów brama sprawdza ich podpis symetryczny, czas życia oraz atrybuty wydawcy (ang. *Issuer*) i odbiorcy (ang. *Audience*). Dodatkowo API Gateway egzekwuje polityki CORS (ang. *Cross-Origin Resource Sharing*), blokując ruch pochodzący z niezaufanych źródeł i przepuszczając do sieci prywatnej wyłącznie żądania pochodzące z dozwolonych adresów aplikacji klienckiej.

5.1.2.2.2. Opcjonalne uwierzytelnianie z `OptionalAuthHandler`

Autentykacja na poziomie bramy sieciowej przebiega na dwa sposoby w zależności od docelowego punktu końcowego żądania. Trasy wymagające bezwzględnego uwierzytelniania obsługiwane są przez natywne mechanizmy Ocelota, które każdorazowo weryfikują obecność tokena. Część punktów końcowych dostępnych dla niezalogowanych użytkowników (wysyłających żądania bez tokena) może jednak zwracać dodatkowe informacje, jeśli żądanie będzie pochodziło od zalogowanego użytkownika (będzie zawierało token) – np. *endpoint* zwracający szczegóły przepisu może zostać wywołany zarówno przez użytkownika anonimowego jak i zalogowanego, jednak w przypadku tego drugiego zostaną dodatkowo zwrócone informacje czy ma on dany przepis dodany do ulubionych oraz jak wcześniej ocenił przepis (jeżeli go ocenił). Taka opcjonalna weryfikacja tokenu jest możliwa dzięki specjalnemu rozwiązaniu w postaci klasy `OptionalAuthHandler`.

Klasa `OptionalAuthHandler` dziedzicząca po abstrakcyjnej klasie `DelegatingHandler` przed przekierowaniem żądania do odpowiedniego mikroserwisu analizuje je pod kątem obecności ważnego

tokena JWT. W przypadku wykrycia poprawnego tokena handler dekoduje z niego identyfikator użytkownika oraz jego rolę, a następnie wstrzykuje wydobyte dane do nagłówków żądania HTTP i przekazuje je dalej do sieci prywatnej. Ponadto, w celu ochrony przed atakami typu *header spoofing* handler zawsze domyślnie nadpisuje nagłówki przychodzących żądań. Implementacja klasy `OptionalAuthHandler` została przedstawiona na Listing 20.

Listing 20 Implementacja klasy `OptionalAuthHandler`. Źródło: Opracowanie własne.

```
using Microsoft.AspNetCore.Authentication;

namespace GatewayService.Handlers;

public class OptionalAuthHandler : DelegatingHandler
{
    private readonly IHttpContextAccessor _httpContextAccessor;

    public OptionalAuthHandler(IHttpContextAccessor httpContextAccessor)
    {
        _httpContextAccessor = httpContextAccessor;
    }

    protected override async Task<HttpResponseMessage>
    SendAsync(HttpRequestMessage request, CancellationToken
    cancellationToken)
    {
        if (request.Headers.Contains("X-User-Id"))
        {
            request.Headers.Remove("X-User-Id");
        }

        if (request.Headers.Contains("X-User-Roles"))
        {
            request.Headers.Remove("X-User-Roles");
        }

        var httpContext = _httpContextAccessor.HttpContext;

        if (httpContext != null)
        {
            var authResult = await
            httpContext.AuthenticateAsync("IdentityServiceKey");

            if (authResult.Succeeded && authResult.Principal != null)
            {
                var user = authResult.Principal;

                var userId = user.FindFirst("sub")?.Value;
                var roles = user.FindAll("role").Select(c => c.Value);
                var rolesString = string.Join(",", roles);

                if (!string.IsNullOrEmpty(userId))
                {

```

```

        request.Headers.Add("X-User-Id", userId);
    }

    if (!string.IsNullOrEmpty(rolesString))
    {
        request.Headers.Add("X-User-Roles", rolesString);
    }
}

return await base.SendAsync(request, cancellationToken);
}
}

```

5.1.2.3 Współdzielenie kodu i zarządzanie bazą danych

Projektując warstwę persystencji danych autorzy świadomie zdecydowali się na jedną bazę dla całego systemu zamiast tworzenia osobnych baz dla każdego mikroserwisu. Powodem takiego rozwiązania było bardzo silne połączenie siecią relacji (ang. *high coupling*) encji z mikroserwisu przepisów oraz mikroserwisu użytkowników. Próba utrzymywania referencji do encji w odrębnych systemach bazodanowych rodziłaby wysokie ryzyko desynchronizacji danych, natomiast wprowadzenie oddzielnego systemu przesyłania komunikatów pomiędzy mikroserwisami nie przyniosłoby zysków współmiernych do kosztów wdrożenia takowego przy obecnej skali projektu.

Aby odpowiednio zrealizować współdzielenie dostępu do bazy danych, a jednocześnie uniknąć duplikacji kodu, mechanizm dostępu do systemu bazodanowego wydzielono do oddzielnej biblioteki `Shared.Database`. Jej struktura dzieli się na następujące obszary:

- `Data` – zawiera klasę `AppDbContext` dziedziczącą po `DbContext`, definiującą relacje pomiędzy encjami i zarządzającą połączeniem z bazą danych,
- `Models` – zbiór klas reprezentujących tabele bazodanowe w postaci obiektowej,
- `Migrations` – generowane przy użyciu narzędzia Entity Framework Core pliki służące do wersjonowania schematu bazy danych,
- `Enums` – zawiera wartości wyliczeniowe określające m.in. dostępne statusy produktów i przepisów, które wykorzystano do nietrwałego usuwania (ang. *soft delete*) rekordów tych encji w systemie.

Zarządzanie współdzieloną przez mikroserwisy bazą danych wymagało ustalenia, która z usług będzie odpowiadała za jej inicjalizację i aktualizację jej schematu, aby uniknąć konfliktów i błędów współbieżności (ang. *race conditions*) w sytuacji równoległego uruchamiania mikroserwisów. Zdecydowano, że odpowiedzialny za to będzie mikroserwis użytkowników. Przy każdym uruchomieniu `UsersService` sprawdza, czy istnienie docelowa baza danych jest już obecna na serwerze i w razie potrzeby ją tworzy. Następnie mikroserwis użytkowników weryfikuje, czy wszystkie migracje (pliki w obszarze `Migrations`) zostały zastosowane. W przypadku weryfikacji negatywnej aplikuje on wszystkie oczekujące migracje, upewniając się tym samym, że schemat bazy danych jest aktualny. Ostatnim etapem procesu rozruchowego `UsersService` jest zasilenie bazy danymi początkowymi (tzw. *seeding*). Odbywa się ono poprzez wywołanie statycznej metody `SeedData()` z dedykowanej klasy `DbInitializer`. Metoda inicjalizuje cztery podstawowe role systemowe oraz tworzy konto z najwyższymi uprawnieniami (`SuperAdmin`), którego poświadczenia odczytywane są ze zmiennych konfiguracyjnych środowiska.

5.1.3 Komunikacja z bazą danych

Wymiana informacji pomiędzy mikroserwisami a serwerem SQL została w całości zrealizowana przy użyciu narzędzia Entity Framework Core. Wprowadza ono dodatkową warstwę abstrakcji pozwalającą na wykonywanie operacji na bazie danych z wykorzystaniem języka C# oraz technologii LINQ. Narzędzie Entity Framework Core zostało szczegółowo opisane w rozdziale 4.1.4.

Istotnym aspektem komunikacji komponentów usługowych z bazą danych jest jej całkowita asynchroniczność. Dzięki takiemu podejściu wątki serwera obsługujące żądania HTTP nie są blokowane w oczekiwaniu na odpowiedź z bazy, lecz mogą w tym czasie przetwarzać inne żądania, co znacznie zwiększa efektywność całego systemu.

Aby jeszcze bardziej zwiększyć wydajność systemu, w miejscach gdzie dane pobierane są z bazy wyłącznie w celu odczytania ich, wykorzystywano metodę `AsNoTracking()`. Wyłącza ona mechanizm *Change Tracker* śledzący zmiany w zwracanych danych, co sprawia że takie zapytania są znacznie szybsze i zużywają mniej pamięci. Listing 21 przedstawia operację asynchronicznego pobrania z bazy danych listy produktów z użyciem metody `AsNoTracking()` bez możliwości modyfikacji zwróconych danych w ramach transakcji.

Listing 21 Operacja asynchronicznego pobrania z bazy danych listy produktów do odczytu. Źródło:

Opracowanie własne.

```
var products = await _context.Products
    .AsNoTracking()
    .Where(p => p.Status == ProductStatus.Approved)
    .Select(p => new ProductDto
    {
        Id = p.Id,
        Name = p.Name,
        Status = p.Status.ToString(),
        CreatedBy = p.AuthorId,
        AuthorUsername = p.Author.Username,
        CreatedAt = p.CreatedAt,
        RejectionReason = p.RejectionReason,
    })
    .ToListAsync();
```

W przypadku operacji modyfikujących Entity Framework Core automatycznie łączy zmiany w obiektach i wysyła je do bazy w ramach jednej transakcji podczas wywołania `SaveChangesAsync()`. W Listing 22 przedstawiono operację pobrania konkretnego wystąpienia produktu na liście zakupów, następnie operację usunięcia tego wystąpienia z listy oraz towarzyszącą temu zdarzeniu operację zmiany czasu ostatniej aktualizacji listy zakupów w ramach jednej transakcji z wykorzystaniem metody `SaveChangesAsync()`.

Listing 22 Operacja asynchronicznego usunięcia wystąpienia produktu na liście zakupów. Źródło: Opracowanie własne.

```
var item = await _context.ShoppingListItems
    .Include(i => i.ShoppingList)
    .FirstOrDefaultAsync(i => i.Id == itemId);

// validations

_context.ShoppingListItems.Remove(item);
```

```
item.ShoppingList.UpdatedAt = DateTime.UtcNow;  
  
await _context.SaveChangesAsync();
```

5.1.4 Mechanizmy uwierzytelniania i autoryzacji

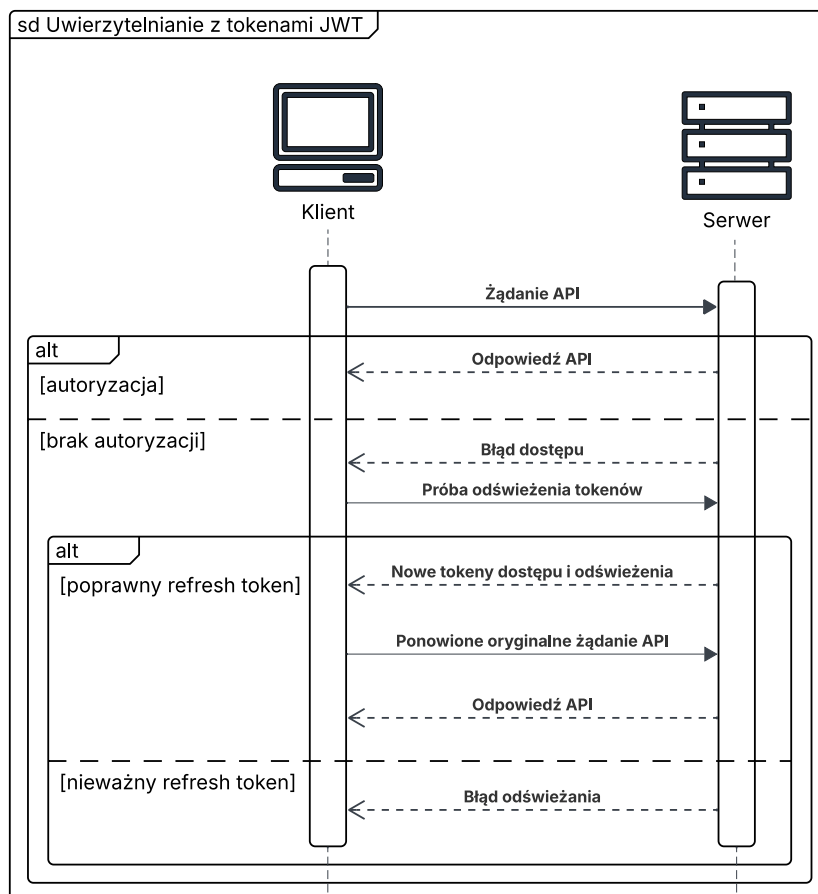
Bezpieczeństwo aplikacji opiera się na weryfikacji tożsamości użytkownika oraz określeniu i skutecznym egzekwowaniu jego uprawnień w systemie. Poniższe podrozdziały przybliżają techniczną realizację tych założeń, omawiając zarządzanie sesją przy użyciu tokenów JWT, a także weryfikację i mechanizm przyznawania uprawnień w hierarchicznej strukturze ról.

5.1.4.1 Uwierzytelnianie użytkowników z wykorzystaniem tokenów JWT

Weryfikacja tożsamości użytkowników w opisywanym systemie realizowana jest w oparciu o bezstanowych standard JSON Web Token (JWT). Aby zapewnić stały poziom bezpieczeństwa przy jednoczesnym zachowaniu płynności działania aplikacji klienckiej, zdecydowano się na implementację w mikrosерwisie użytkowników mechanizmu uwierzytelniania opartego na dwóch tokenach:

- Token dostępowy (ang. *Access Token*) – jest głównym kluczem dostępowym zabezpieczonym algorytmem HMAC-SHA256, ważnym tylko przez 15 minut. Zawiera w sobie zakodowany identyfikator oraz adres e-mail użytkownika, a także jego rolę.
- Token odświeżania (ang. *Refresh Token*) – jest to ważny przez siedem dni token pomocniczy służący do odnawiania tokenów dostępowych w postaci generowanego losowo ciągu znaków w standardzie Base64 zapisywanego w bazie danych od razu po utworzeniu. W przypadku wygaśnięcia tokena dostępowego jest on asynchronicznie przesyłany w specjalnym żądaniu odświeżenia *access tokena* inicjowanym przez aplikację kliencką i porównywany z tym zapisanym w bazie. Aktualny *refresh token* jest oznaczony jako nieważny i podmieniany na nowy wraz z odświeżeniem tokenu dostępowego. Dzięki tokenowi odświeżenia możliwe jest też natychmiastowe unieważnienie sesji użytkownika poprzez usunięcie jego aktualnego *refresh tokena* z bazy danych, co wykorzystano w przypadku operacji blokowania konta użytkownika.

Za walidację przychodzących tokenów i zwracanie nowych odpowiada *AuthService*. Komunikuje się on z *TokenService* odpowiedzialnym wyłącznie za generowanie tokenów. W metodzie *RefreshTokenAsync()* odpowiedzialnej za rotację tokenów został również uwzględniony przypadek, gdy aplikacja kliencka wysła wiele próśb o ich odświeżenie jednocześnie. Aby zapobiec nagłemu wylogowaniu użytkownika w takiej sytuacji, tylko pierwsze z przesłanych żądań odświeża tokeny, a kolejnym żądaniom, które zostaną odebrane w ciągu 30 sekund od wygenerowania ostatniego *refresh tokena*, token ten również zostanie zwrócony. Dodatkowo, podczas każdej próby rotacji tokenów weryfikowany jest status konta, aby w przypadku żądania pochodzącego z konta zablokowanego uniemożliwić wydanie nowych. Rysunek 41 przedstawia schemat podstawowego uwierzytelniania opartego na tokenie dostępowym i tokenie odświeżania.



Rysunek 41 Diagram sekwencji przedstawiający uwierzytelnianie oparte na tokenie dostępowym i tokenie odświeżania. Źródło: Opracowanie własne.

5.1.4.2 Zarządzanie uprawnieniami i hierarchia ról

Autoryzacja w systemie realizowana jest w oparciu o hierarchiczny model ról. Standardowemu użytkownikowi w momencie rejestracji automatycznie przypisywana jest rola User. Wyższe uprawnienia mogą zostać nadane wyłącznie przez użytkowników z rolami Admin oraz SuperAdmin. Hierarchia ról w opisywanym systemie uporządkowana od roli z najniższymi uprawnieniami do roli z wszystkimi możliwymi uprawnieniami przedstawia się następująco:

- User,
- Moderator,
- Admin,
- SuperAdmin.

Uprawnienia poszczególnych ról szczegółowo opisano w rozdziale 3.2.

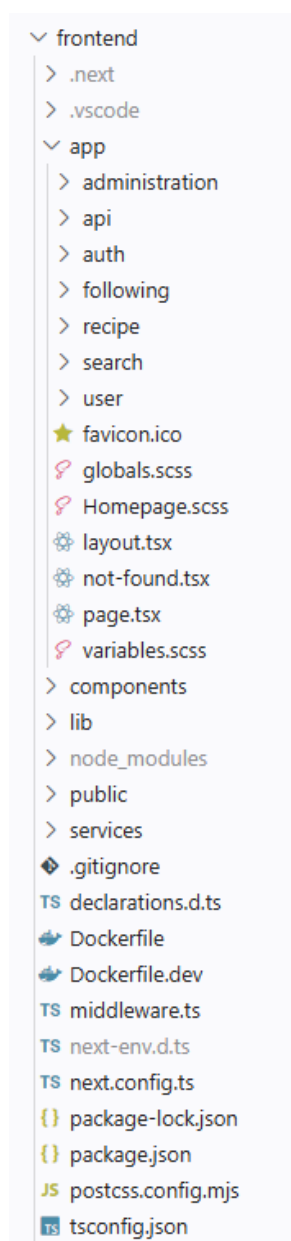
W celu zabezpieczenia aplikacji przed nieuprawnioną eskalacją przywilejów, wdrożono rygorystyczne reguły weryfikacji podczas nadawania i odbierania ról użytkownikom. Użytkownik wykonujący operację zmiany ról może modyfikować wyłącznie konta o niższej randze niż jego własna, ale nie może nadawać im ról równej lub wyższej niż ta, którą sam posiada. Wyjątkiem jest rola SuperAdmin, która jest zarezerwowana wyłącznie dla jednego, głównego konta w systemie. System odrzuca wszelkie próby nadania lub zabrania tej roli, co uniemożliwia całkowite przejęcie nad nim kontroli w przypadku wycieku poświadczeń zwykłego administratora.

5.2. Implementacja aplikacji webowej – frontend

Aplikacja webowa została stworzona za pomocą języka TypeScript z wykorzystaniem biblioteki React do budowy interfejsu graficznego oraz Next.js do obsługi routingu stron. Wygląd interfejsu oparto o bibliotekę React Bootstrap z modyfikacjami. Kod został napisany za pomocą edytora Visual Studio Code.

5.2.1 Struktura projektu aplikacji webowej

Kod aplikacji webowej znajduje się w katalogu frontend w monorepozytorium. Rysunek 42 prezentuje szkielet aplikacji w edytorze Visual Studio Code.



Rysunek 42 Struktura aplikacji webowej Let Me Cook w Visual Studio Code. Źródło: Opracowanie własne.

Bezpośrednio w folderze `frontend` mieszczą się pliki konfiguracyjne. Najważniejsze z nich to:

- `Dockerfile` oraz `Dockerfile.dev` – do tworzenia obrazów projektu w Docker,
- `next.config.ts` – plik konfiguracyjny Next.js,
- `package.json` oraz `package-lock.json` – pliki definiujące projekt w Node.js,
- `tsconfig.json` – plik konfiguracyjny kompilatora TypeScript,
- `middleware.ts` – część oprogramowania pośredniczącego Next.js, opisane szczegółowo w rozdziale 5.2.2.1.

Poniżej znajduje się opis poszczególnych podkatalogów aplikacji oraz najistotniejszych plików:

- `/app` – *App router* – zawiera strukturę stron, niektóre komponenty w formacie `.tsx` oraz style Sass:
 - `variables.scss` – wykaz nadpisanych zmiennych styli z biblioteki Bootstrap,
 - `globals.scss` – własne, globalne style Sass,
 - `page.tsx` – strona główna aplikacji,
 - `layout.tsx` – dodaje globalne funkcjonalności i elementy nawigacji do każdej strony,
 - `/api` – *Route Handlers*, część oprogramowania pośredniczącego Next.js, opisane szczegółowo w rozdziale 5.2.2.2,
 - `/administration` – strony panelu administracyjnego,
 - `/auth` – strony logowania i rejestracji użytkownika,
 - `/following` – strona wyświetlająca przepisy tylko od obserwowanych użytkowników,
 - `/recipe` – strony przedstawiające przepis oraz edytor przepisu,
 - `/search` – strona wyszukiwarki przepisów,
 - `/user` – strony przedstawiające profil użytkownika, ulubione przepisy, listy zakupów oraz ustawienia konta;
- `/components` – zawiera komponenty UI pogrupowane według rodzaju komponentu,
- `/lib` – kod i własne biblioteki TypeScript obsługujące dane, klasy i połączenie z częścią serwisową,
 - `auth.ts`, `change-password.ts` – biblioteki obsługujące uwierzytelnianie użytkownika,
 - `client-axios.ts` – instancja biblioteki Axios dla stron renderowanych po stronie użytkownika,
 - `gateway-axios.ts` – instancja biblioteki Axios dla stron renderowanych po stronie serwera,
 - `enums.ts` – typy wyliczeniowe,
 - `types.ts` – typy,
 - `html-markdown-converter.ts` – konfiguracja bibliotek Unified, Remark i Rehype to konwertowania kodu Markdown do HTML i odwrotnie,

- `ingredient-amount.ts` – biblioteka obsługująca operacje na ilościach składników,
- `utils.ts`, `recipe-utils.ts` – funkcje pomocnicze;
- `/public` – pliki statyczne,
- `/services` – zawiera zbiory wywołań API do części serwerowej projektu.

Aby zapewnić szybkość aplikacji, autorzy zdecydowali się podzielić komponenty na serwerowe i klienckie. Komponenty serwerowe zawierają metadane, ładują dane i renderują częściowo interfejs użytkownika podczas budowania aplikacji na serwerze. Ten schemat ogranicza ilość kodu JavaScript uruchamianego po stronie przeglądarki i zmniejsza czas oczekiwania użytkownika na stronę [52]. Komponenty klienckie są potrzebne do reaktywnej obsługi elementów na stronie z interakcjami użytkownika po stronie przeglądarki.

Listing 23 prezentuje fragment strony panelu administracyjnego z komponentem serwerowym. Zawiera on metadane i strukturę komponentów strony. Jednym z komponentów klienckich na stronie jest `AddCategoryButton`, obsługujący dodawanie kategorii za pomocą formularza w modalu. Kod tego komponentu jest pokazany w Listing 24.

Listing 23 Fragment komponentu serwerowego tworzącego panel administracyjny. Źródło: Opracowanie własne.

```
app/user/administration/dashboard/page.tsx
// ...imports

export async function generateMetadata() {
  return {
    title: "Panel administracyjny",
  };
}

export default async function AdminDashboardPage() {
  const user = await getCurrentJwtUser();
  if (!user) return;

  const userRoles = user.role.split(",");
  const rolesBadgeMap: { [key: string]: string } = {
    SuperAdmin: "primary",
    Admin: "secondary",
    Moderator: "tertiary",
    User: "success",
  };

  return (
    <>
      <CardText className="text-muted d-flex flex-row align-items-center gap-2">
        <span>#{user.id}</span>
        {userRoles.map((role) => (
          <Badge key={role} bg={rolesBadgeMap[role] || "secondary">

```

```

        {role}
      </Badge>
    )))
  </CardText>
  <h2 className="mb-5">Witaj {user.username}!</h2>
  <div className="text-muted mb-2">Szybkie akcje</div>
  <Stack direction="horizontal" gap={2}>
    <AddProductButton />
    <AddCategoryButton />
  </Stack>
  <div className="text-muted mb-2">Wymaga akceptacji</div>
  <AdminSummaryAcceptanceTable />
  <div className="text-muted mb-2">Zgłoszenia</div>
  <AdminSummaryReportsTable />
</>
);
}

```

Listing 24 Fragment komponentu klienckiego AddCategoryButton. Źródło: Opracowanie własne.

```

components/buttons/crud/category/AddCategoryButton.tsx
"use client";

// ...imports

const APPROVED_TAB_URL = "/administration/categories?tab=approved";

export default function AddCategoryButton() {
  const categoryService = getCategoryService(clientAxios);

  const {
    register,
    handleSubmit,
    formState: { errors },
    reset,
    setValue,
  } = useForm();
  const toast = useToast();

  async function onSubmit(data: any, closeModal?: () => void) {
    // ...
  }

  return (
    <ModalButton
      triggerProps={{ variant: "success", className: "mb-3" }}
      shouldCloseOnSubmit={false}
      onSubmit={(closeModal) =>
        handleSubmit((data) => onSubmit(data, closeModal))()
      }
      modalTitle="Dodaj nową kategorię"
    />
  );
}

```

```

modalFormBody={
  <>
    <Alert variant="info">
      {/* ... */}
    </Alert>
    <Form.Group className="mb-3" controlId="name">
      <Form.Label>Nazwa kategorii:</Form.Label>
      <Form.Control
        autoFocus
        isValid={!errors.name}
        placeholder="Podaj nazwę kategorii..."
        {...register("name", {
          required: "Podanie nazwy jest wymagane",
          maxLength: {
            value: 100,
            message: "Nazwa nie może przekraczać 100 znaków.",
          },
          minLength: {
            value: 3,
            message: "Nazwa musi mieć co najmniej 3 znaki.",
          },
          onChange: (e) => setValue("name",
firstLetterUppercase(e.target.value)),
        })}
      />
      <Form.Control.Feedback type="invalid">
        {errors.name?.message as string}
      </Form.Control.Feedback>
    </Form.Group>
    <Form.Group className="mb-3" controlId="homepageOrder">
      <Form.Label>Pozycja na stronie głównej:</Form.Label>
      <Form.Select
        id="homepageOrder"
        {...register("homepageOrder", {
          valueAsNumber: true,
        })}
      >
        <option value="0">Brak</option>
        {Array.from({ length: 20 }, (_, i) => (
          <option key={i + 1} value={i + 1}>
            {i + 1}
          </option>
        ))}
      </Form.Select>
    </Form.Group>
  </>
}
>
  <i className="bi bi-plus-lg"></i>Dodaj kategorię
</ModalButton>
);
}

```

W celu umożliwienia użytkownikom dostępu do przepisów, zastosowano wbudowany mechanizm dynamicznych segmentów trasy (ang. *Dynamic Route Segments*). Za jego pomocą można odczytywać parametry z ścieżki URL i na ich podstawie pokazywać odpowiednią zawartość strony [53]. Segment należy utworzyć poprzez zawarcie nazwy parametru w nawiasy kwadratowe, np. `[id]`.

W Listing 25 przedstawiono fragment strony profilu użytkownika, która za pomocą parametru `id` i mechanizmowi *Dynamic Route Segments* wczytuje profil użytkownika o podanym identyfikatorze w URLu. Przykładowo dla użytkownika o identyfikatorze 1002 należy wejść na stronę pod adresem `http://localhost/user/profile/1002`.

Listing 25 Fragment strony profilu użytkownika korzystającej z *Dynamic Route Segments*.

Źródło: Opracowanie własne.

```
/app/user/profile/[id]/page.tsx
// ...imports

export async function generateMetadata({
  params,
}): {
  params: Promise<{ id: string }>;
}): Promise<Metadata> {
  const id = Number((await params).id);
  const userService = getUserService(gatewayAxios);
  const userProfile = await userService.getUserById({ id });

  return {
    title: `${userProfile.username} - Profil użytkownika`,
    description: `Profil użytkownika ${userProfile.username} w serwisie
LetMeCook`,
  };
}

export default async function UserProfilePage({
  params,
}): {
  params: Promise<{ id: string }>;
}) {
  const id = Number((await params).id);

  const loggedUser = await getCurrentJwtUser();
  const isOwnProfile = loggedUser?.id === String(id);

  const userService = getUserService(gatewayAxios);
  const userProfile = await userService.getUserById({ id });

  return (
    <>
      {userProfile.isBanned && (
        <Alert variant="danger" className="mb-2">
          <i className="bi bi-exclamation-circle"></i>

```

```

        Konto tego użytkownika zostało zablokowane.
    </Alert>
  })
  <SectionPanelBody>
    <SectionSidePanel>
      <ProfileCards userId={userProfile.id} />
    </SectionSidePanel>
    <SectionMainPanel>
      <Card>
        <CardBody>
          <h3 className="mb-3 fw-bold text-center text-lg-start">
            Przepisy
          </h3>
          <ProfileRecipeList
            isOwnProfile={isOwnProfile}
            userProfile={userProfile}
          />
        </CardBody>
      </Card>
    </SectionMainPanel>
  </SectionPanelBody>
</>
);
}

```

5.2.2 Oprogramowanie pośredniczące

W aplikacji webowej zastosowano oprogramowanie pośredniczące (ang. *middleware*), stanowiące warstwę pośredniczącą między przeglądarką lub serwerem Next.js a częścią serwerową. Zapewnia ono zapisywanie tokenów uwierzytelniających JWT przychodzących z API jako ciastka (ang. *cookies*), odświeżania sesji użytkownika przy zapytaniach do serwera kiedy to potrzebne i kontrolę dostępu do stron wymagających zalogowania.

Oprogramowanie pośredniczące podzielone jest na:

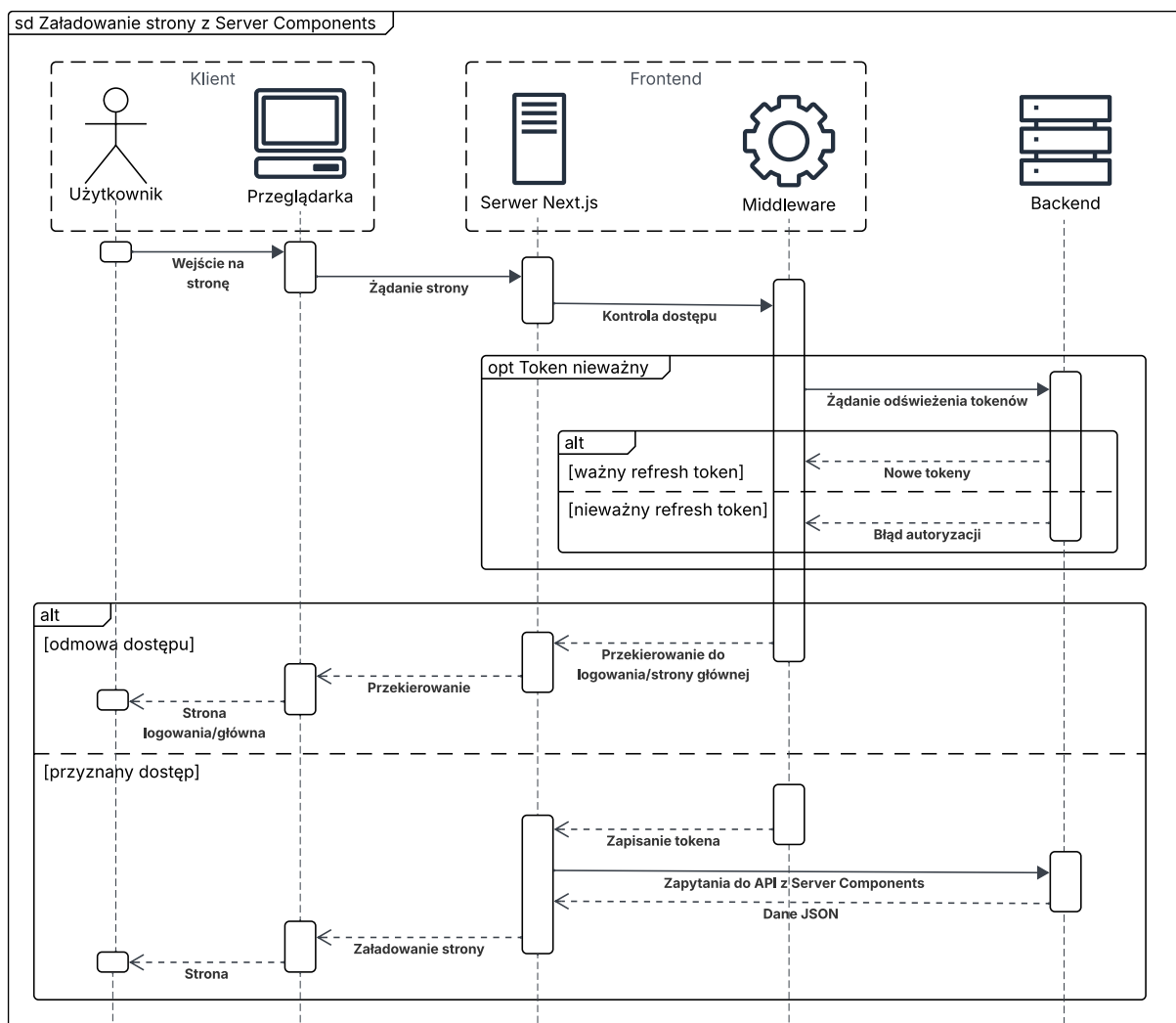
- `middleware.ts` – na poziomie routingu,
- `/app/api/[...path]/route.ts` – proxy dla zapytań z przeglądarki,

5.2.2.1 Middleware.ts

Oprogramowanie pośredniczące `middleware.ts` uruchamiane jest zawsze, gdy użytkownik przynosi się między stronami aplikacji [54]. Na początku serwer frontendu sprawdza ważność tokenu uwierzytelniającego – jeśli jest nieważny, wysyłane jest żądanie odświeżenia tokenów do backendu. Dalej następuje podstawowa kontrola dostępu – oprogramowanie sprawdza, czy użytkownik jest zalogowany, a jeżeli tak to jakie posiada role. W przypadku, gdy na podstawie tych informacji i ścieżki do żądanej strony stwierdzono odmowę dostępu, następuje przekierowanie na stronę logowania lub stronę główną.

W momencie przyznania dostępu do strony `middleware` zapisuje nowe tokeny dostępowe jako pliki cookie (gdy były odświeżane) i przetwarzanie po stronie `middleware` się kończy. Jeżeli strona zawiera komponenty działające po stronie serwera, które pobierają dane z backendu, to zapytania są kierowane prosto do bramy pośredniczącej na backendzie. Na końcu następuje wygenerowanie strony po stronie serwera i przesłanie jej do przeglądarki.

Rysunek 43 ilustruje mechanizm działania oprogramowania pośredniczącego pliku `middleware.ts` w Next.js za pomocą diagramu sekwencji.



Rysunek 43 Diagram sekwencji załadowania strony z oprogramowaniem pośredniczącym Next.js w aplikacji Let Me Cook. Źródło: Opracowanie własne.

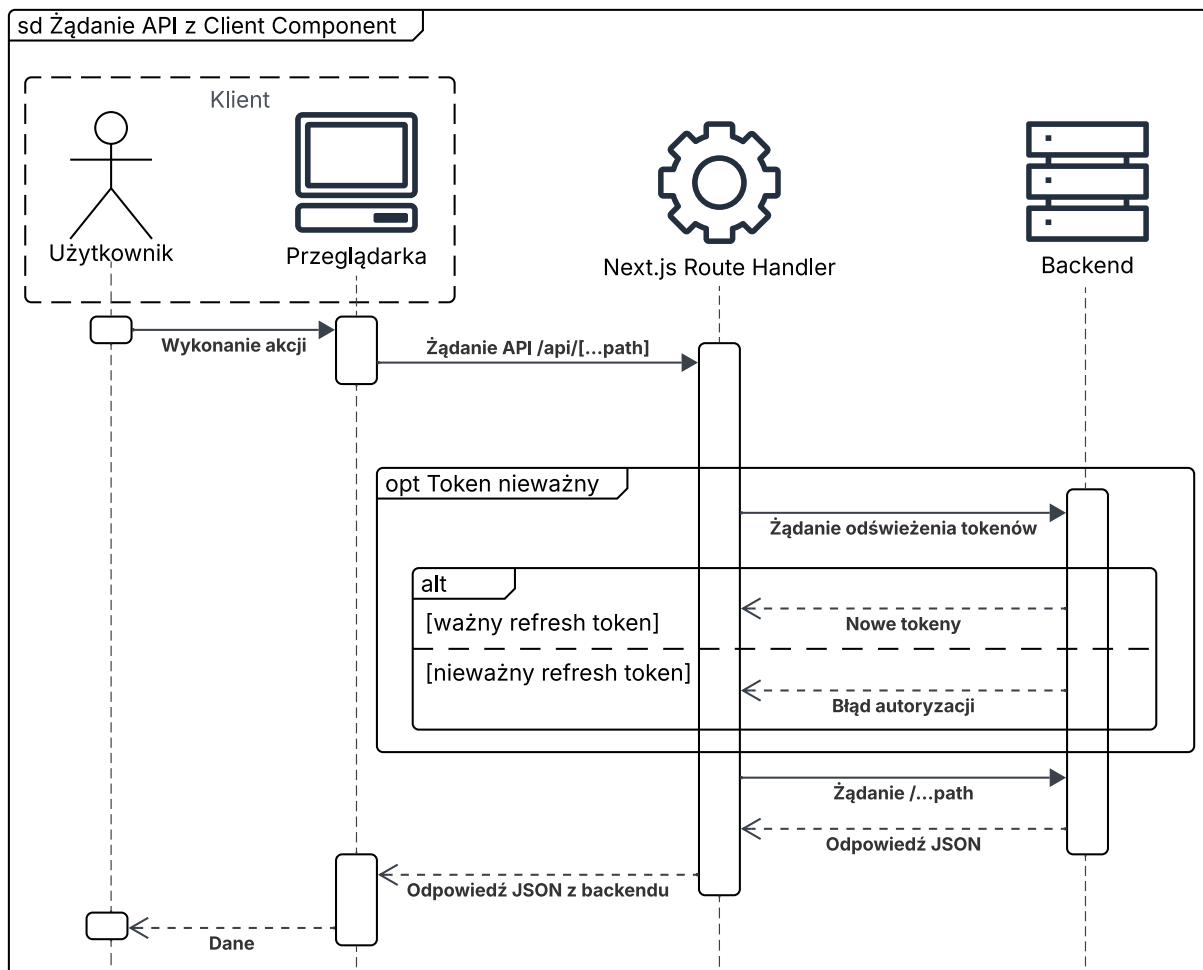
5.2.2.2 Proxy /api/...

Do poprawnego działania aplikacji, oprócz odświeżania tokenów podczas zmieniania stron, należy upewnić się, że zapytania do backendu również posiadają ważne tokeny. W tym celu zastosowano mechanizm Route Handlers wbudowany w Next.js. Pozwala on na niestandardowe obsługiwane żądań API dla zdefiniowanej ścieżki. Na potrzeby projektu utworzono 3 takie funkcje obsługujące:

1. `/api/auth/login/route.ts` – wysyła żądanie zalogowania użytkownika do backendu i zapisuje otrzymane tokeny jako pliki cookie,
2. `/api/auth/logout/route.ts` – wysyła żądanie wylogowania użytkownika i usuwa pliki cookie unieważnionymi tokenami,

3. `/api/[...path]/route.ts` – przechwytuje żądania z przeglądarki użytkownika i dołącza token autoryzacyjny w nagłówku, w razie potrzeby odświeża token.

Ostatnia funkcja stanowi swego rodzaju proxy poprzez użycie *Dynamic Route Segments* – każde zapytanie idące z przeglądarki klienta (*Client Component*) przechodzi przez tę funkcję. Rysunek 44 prezentuje mechanizm działania Route Handler `/api/[...path]/route.ts` poprzez diagram sekwencji.



Rysunek 44 Diagram sekwencji działania proxy z Route Handlers w aplikacji *Let Me Cook*.

Źródło: Opracowanie własne.

5.2.3 Opis użycia bibliotek

W tym podrozdziale opisano sposób użycia bibliotek użytych po stronie frontendu. Odpowiednia konfiguracja bibliotek, zastosowanie mechanizmu dziedziczenia czy napisanie własnych bibliotek skróciło czas pisania kodu, zwiększyło jego czytelność oraz łatwość późniejszej konserwacji.

5.2.3.1 Biblioteka Bootstrap

W celu dostosowania domyślnego stylu i palety kolorów Bootstrapa, użyto pliku `variables.scss`, który nadpisuje niektóre zmienne biblioteki, oraz plik `globals.scss` który importuje bibliotekę i dodaje własne style. Następnie plik `variables.scss` uwzględniono w

konfiguracji Next.js, aby został dołączony do każdego komponentu React, a plik `globals.scss` załączono do pliku `layout.tsx`, który odpowiada za bazowy rozkład komponentów i styli.

Listing 26 przedstawia fragment pliku `variables.scss`, a Listing 27 użycie tego pliku w konfiguracji Next.js.

Listing 26 Fragment pliku `variables.scss` nadpisujący zmienne Bootstrap. Źródło: Opracowanie własne.

```
@import "bootstrap/scss/functions";
@import "bootstrap/scss/variables";

$enable-negative-margins: true;

$primary: $red-600;
$secondary: $red-400;
$tertiary: $orange-100;
$background: $gray-100;
$foreground: #171717;

$font-family-sans-serif: 'Inter', system-ui, -apple-system, "Segoe UI",
Roboto, "Helvetica Neue", "Noto Sans", "Liberation Sans", Arial, sans-
serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto
Color Emoji";

$form-check-input-checked-bg-color: $primary;
$form-check-input-checked-border-color: $primary;
$form-check-input-focus-border: $tertiary;
$form-check-input-focus-box-shadow: 0 0 0 .25rem rgba($orange-100, .25);

$btn-link-color: $secondary;
$btn-link-hover-color: darken($secondary, 10%);

$form-switch-focus-color: $primary;
$form-switch-focus-bg-image: url("data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='-4 -4 8 8'><circle r='3'
fill='#{$form-switch-focus-color}'/></svg>");

$dropdown-link-active-bg: $secondary;

// ...reszta zmian w Link, Tooltip, Form.Range, ListGroup, Accordion,
Nav, Pagination, Carousel

$custom-colors: (
  "primary": $primary,
  "secondary": $secondary,
  "tertiary": $tertiary,
  "background": $background,
  "foreground": $foreground
);
$theme-colors: map-merge($theme-colors, $custom-colors);
```

Listing 27 Fragment pliku konfiguracyjnego Next.js next.config.ts. Źródło: Opracowanie własne.

```
const nextConfig: NextConfig = {
  /* config options here */
  sassOptions: {
    additionalData: `@import "/app/variables.scss";`,
    quietDeps: true,
  },
  // ...
}
```

Oprócz konfiguracji stylu, Bootstrap został użyty do utworzenia własnych komponentów składających się z poszczególnych komponentów z biblioteki. Listing 28 pokazuje fragment kodu komponentu ModalButton, który korzysta z komponentów Bootstrapa takich jak OverlayTrigger, Tooltip, Modal i Button.

Listing 28 Fragment kodu komponentu ModalButton. Źródło: Opracowanie własne.

```
"use client";
// ...imports, ModalButtonProps

export default function ModalButton({
  children,
  tooltipText,
  onModalOpen,
  onCancel,
  onSubmit,
  modalTitle = "Potwierdzenie",
  modalFormBody,
  triggerProps,
  submitProps,
  shouldOpenModal = true,
  shouldCloseOnSubmit = true,
}: ModalButtonProps) {
  const [showModal, setShowModal] = useState(false);
  const [isLoading, setIsLoading] = useState(false);

  const handleConfirm = async (e: any) => { /* ... */ };
  const renderTrigger = () => { /* ... */ };

  return (
    <>
      {tooltipText ? (
        <OverlayTrigger
          overlay={
            <Tooltip style={{ position: "fixed"
}}>{tooltipText}</Tooltip>
          }
        >
          {renderTrigger()}
        </OverlayTrigger>
      )}
```

```

) : (renderTrigger()))}
<Modal
  show={showModal}
  onHide={() => {
    setShowModal(false);
    onCancel?.();
  }}
  onShow={onModalOpen}
>
  <Modal.Header closeButton>
    <Modal.Title>{modalTitle}</Modal.Title>
  </Modal.Header>

  <Modal.Body>
    <Form onSubmit={handleConfirm}>
      {modalFormBody ?? <p>Czy na pewno chcesz wykonać tę
operację?</p>}
    </Form>
  </Modal.Body>

  <Modal.Footer>
    <Button
      variant="outline-dark"
      onClick={() => {
        setShowModal(false);
        onCancel?.();
      }}
      disabled={isLoading}
    >
      Anuluj
    </Button>
    <Button
      variant="success"
      type={submitProps?.type || "button"}
      {...submitProps}
      onClick={handleConfirm}
      disabled={isLoading || submitProps?.disabled}
    >
      {isLoading ? <Spinner /> : submitProps?.children ||
"Zatwierdź"}
    </Button>
  </Modal.Footer>
</Modal>
</>
);
}

```

5.2.3.2 Biblioteka TipTap

Biblioteka TipTap posłużyła w pracy do utworzenia edytora sformatowanego tekstu w sposobie przygotowania przepisu. Edytor wspiera podstawowe funkcjonalności, jak na przykład ustawienie

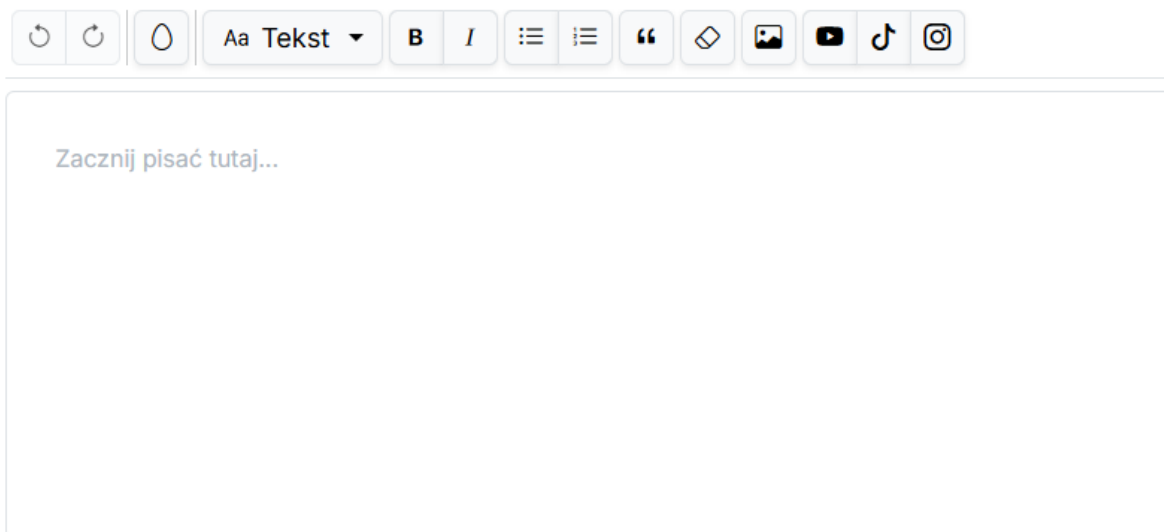
nagłówek, pogrubienie, pochylenie czy listy, ale też umieszczenie obrazu lub umieszczenie ramki do filmu w serwisie YouTube.

Za pomocą TipTap utworzono komponent `MarkdownRecipeEditor`, zawierający instancję edytora z konfiguracją rozszerzeń oraz interfejsem graficznym. Listing 30 przedstawia fragment kodu tego komponentu odpowiedzialnego za inicjowanie biblioteki TipTap, a Rysunek 45 jego interfejs graficzny.

Listing 29 Inicjalizacja instancji edytora TipTap w komponencie MarkdownRecipeEditor.

Źródło: Opracowanie własne.

```
const editor = useEditor({
  editorProps: editorProps,
  content: value,
  extensions: [
    StarterKit.configure({
      underline: false, // not supported in markdown
      code: false,
      codeBlock: false,
      heading: { levels: [1, 2, 3] },
    }),
    BubbleMenuExtension,
    FileHandler.configure({
      allowedMimeTypes: ["image/png", "image/jpeg", "image/gif",
"image/webp",
    ],
      onDrop: async (currentEditor, files, pos) => { /* ... */},
      onPaste: (currentEditor, files, htmlContent) => { /* ... */},
    }),
    Image.configure({ HTMLAttributes: { class: "img-fluid img-thumbnail
mx-auto d-block" }}),
    Youtube.configure({ HTMLAttributes: { class: "embed-only-horizontal"
}}),
    Tiktok, Instagram, IngredientMention,
    Extension.create({
      name: "mentionSuggestion",
      addProseMirrorPlugins() {return [Suggestion({ /* ... */})];},
    }),
    Placeholder.configure({ placeholder: "Zacznij pisać tutaj..." }),
  ],
  // Don't render immediately on the server to avoid SSR issues
  immediatelyRender: false,
  onUpdate: ({ editor }) => {
    onUpdate?.(editor);
    calculateIngredientAmounts(editor);
  },
  onCreate: ({ editor }) => {
    onCreate?.(editor);
    mentionedIngredientAmountsRef.current =
getMentionedIngredientMentionAmounts(editor);
  },
});
```



Rysunek 45 Interfejs graficzny komponentu MarkdownRecipeEditor. Źródło: Opracowanie własne.

Dodatkowo, w ramach niniejszej pracy, opracowano własne rozszerzenia do biblioteki TipTap:

- IngredientMention – do mechanizmu oznaczania składników w tekście,
- MentionSuggestion – do obsługi podpowiedzi nazw składników do oznaczenia po naciśnięciu znaku @ na klawiaturze,
- Instagram – do umieszczania ramek z zdjęciem/wideo w serwisie Instagram; bliźniacze do istniejącego rozszerzenia YouTube,
- TikTok - do umieszczania ramek z wideo w serwisie TikTok; bliźniacze do istniejącego rozszerzenia YouTube.

Tworzenie własnego rozszerzenia polega na dziedziczeniu jednego z trzech bazowych typów oraz implementacji wybranych metod [55]. Typy, które można rozszerzać, to:

- *Extension* – dodają nowe funkcjonalności lub zmieniają zachowania edytora, ale nie są nowymi elementami w tekście (użyte w MentionSuggestion),
- *Mark* – dodają formatowanie tekstu (użyte w IngredientMention),
- *Node* – są to bloki budujące strukturę dokumentu (użyte w Instagram, TikTok).

Listing 30 przedstawia fragment kodu rozszerzenia IngredientMention, które wykorzystuje typ Mark. Kod definiuje w jaki sposób TipTap powinien zapisywać to niestandardowe formatowanie do postaci kodu HTML i odczytywać je z powrotem. Dodatkowo skorzystano z metody onTransaction, która waliduje formatowanie przy każdej zmianie tekstu w edytorze, żeby zapobiec powstawaniu pustych oznaczeń składnika.

Listing 30 Fragment kodu rozszerzenia IngredientMention dla edytora TipTap. Źródło: Opracowanie własne.

```
import { Mark, mergeAttributes } from '@tiptap/core'

export const IngredientMention = Mark.create({
  name: 'ingredientMention',
  excludes: '_',
  keepOnSplit: false,
  exitable: true,
  inclusive: true,

  addAttributes() {
    // kod definiujący atrybuty dla tego mark'a
    return {
      ingredientId: {
        default: null,
        parseHTML: element => element.getAttribute('data-ingredient-id'),
        renderHTML: attributes => {
          if (!attributes.ingredientId) { return {} }
          return { 'data-ingredient-id': attributes.ingredientId }
        },
      },
      label: { default: null },
      amount: {
        default: '',
        parseHTML: element => element.getAttribute('data-ingredient-amount'),
        renderHTML: attributes => {
          if (!attributes.amount) return {}

          return { 'data-ingredient-amount': attributes.amount }
        },
      },
    }
  },

  renderHTML({ HTMLAttributes }) {
    // kod generujący HTML dla tego mark'a
    return ['span', mergeAttributes(HTMLAttributes, {
      class: 'ingredient-item bg-tertiary rounded-1 fw-bold'
    }), 0]
  },

  parseHTML() { return [{ tag: 'span[data-ingredient-id]' }] },

  onTransaction({ transaction }) {
    // kod wykonujący się po każdej zmianie w edytorze
    // ...
  },
})
```

5.2.3.3 Biblioteka ingredient-amount

W ramach niniejszej pracy utworzono autorską bibliotekę *ingredient-amount*. Zadaniem biblioteki jest sumowanie ilości składników z obsługą różnych jednostek, ich przedrostków oraz deklinacji w języku polskim. Aplikacja wykorzystuje to rozwiązanie:

- w edytorze tekstu przepisu – do sumowania miar oznaczeń produktów na potrzeby automatycznie generowanej listy składników,
- na liście zakupów – w sytuacjach, gdy ten sam produkt zostaje dodany kilka razy.

Na podstawie wpisanej wartości przez użytkownika (np. „1/2 kg”) biblioteka tworzy obiekt klasy *Amount*, w której osobno przechowywane są wielkość liczbowa oraz jednostka. Przy próbie dodawania ze sobą dwóch takich obiektów aplikacja odnajduje ich jednostki na liście obsługiwanych typów i dokonuje ich porównania. Jeżeli obie jednostki są identyczne lub różnią się wyłącznie przedrostkiem (np. „kilogram” i „dekagram”), biblioteka sumuje ich wartości i sprowadza je do bazowej jednostki, poprawnie odmienionej w języku polskim. W przypadku, gdy bazowe jednostki są różne (np. „litr” i „sztuka”), rzucany jest wyjątek niezgodności typów. Gdy miara składnika nie posiada określonej jednostki, domyślnie przyjmowana jest „sztuka”.

W niniejszej pracy obsługiwane są następujące jednostki:

- gramy (wraz z kilogramami, dekagramami),
- litry (wraz z mililitrami i centrylitrami),
- łyżeczki,
- łyżki,
- sztuki,
- szczypty,
- szklanki,
- plastry,
- ząbki.

Dodatkowo możliwe jest sumowanie jednostek niewystępujących na powyższej liście pod warunkiem, że zostały zapisane w tej samej formie gramatycznej (np. „1 miara” i „1 miara”) – wtedy wynik będzie sformatowany jako „2 x miara”.

Listing 31 przedstawia fragment listy obsługiwanych jednostek wraz z przypisanymi do nich jednostkami bazowymi oraz opcjonalną odmianą przez przypadki. Z kolei Listing 32 prezentuje funkcję odpowiedzialną za analizę składniową (ang. *parsing*) zapisu miary składnika wprowadzonego przez użytkownika i tworzenie na jego podstawie obiektu *Amount*.

Listing 31 Fragment listy obsługiwanych jednostek przez bibliotekę ingredient-amount.

Źródło: Opracowanie własne.

```
export const UNITS = {
  g: {
    label: "g",
    base: "g",
    factor: 1,
    aliases: ["g", "gram", "gramy", "gramow", "grama"],
```

```

},
kg: {
  label: "kg",
  base: "g",
  factor: 1000,
  aliases: ["kg", "kilogram", "kilogramy", "kilogramow", "kilograma"],
},
dag: {
  label: "dag",
  base: "g",
  factor: 10,
  aliases: ["dag", "dkg", "deko", "dekagram", "dekagramy",
"dekagramow", "dekagrama"],
},
// ...
lyzeczka: {
  label: "łyżeczka",
  labelPolish: {
    one: "łyżeczka",
    few: "łyżeczki",
    many: "łyżeczek",
    part: "łyżeczki",
  },
  base: "lyzeczka",
  factor: 1,
  aliases: ["lyzeczka", "lyzeczka", "lyzeczki", "lyzeczek"],
},
// ...
}

```

Listing 32 Funkcja biblioteki ingredient-amount odpowiedzialna za parsowanie ilości.

Źródło: Opracowanie własne.

```

export function getAmount(amountWithUnit: string): Amount {
  const trimmed = amountWithUnit.trim();

  if (trimmed.length == 0) return new Amount({ amount: 0, unit: "" });

  const unitMatch = trimmed.match(/[a-zA-ZąćęłńóśźżĄĆĘŁŃÓŚŻŻst]+$/);

  let unit: string;
  let amountPart: string;

  if (unitMatch) {
    unit = unitMatch[0];
    amountPart = trimmed.slice(0, trimmed.length - unit.length).trim();
  } else {
    unit = "sztuka";
    amountPart = trimmed;
  }
}

```

```

let parsedAmount: number;

// Sprawdź, czy to ułamek (np. "1/2", "3/4")
const fractionMatch = amountPart.match(
  /^(\d+(?:[. ,]\d+)?)\s*\s*(\d+(?:[. ,]\d+)?)$/ ,
);
if (fractionMatch) {
  const numerator = parseFloat(fractionMatch[1].replaceAll(",", "."));
  const denominator = parseFloat(fractionMatch[2].replaceAll(",",
    "."));
  parsedAmount = numerator / denominator;
} else {
  parsedAmount = parseFloat(amountPart.replaceAll(",", "."));
}

const amount = isNaN(parsedAmount) ? 1 : parsedAmount;

return new Amount({ amount, unit });
}

```

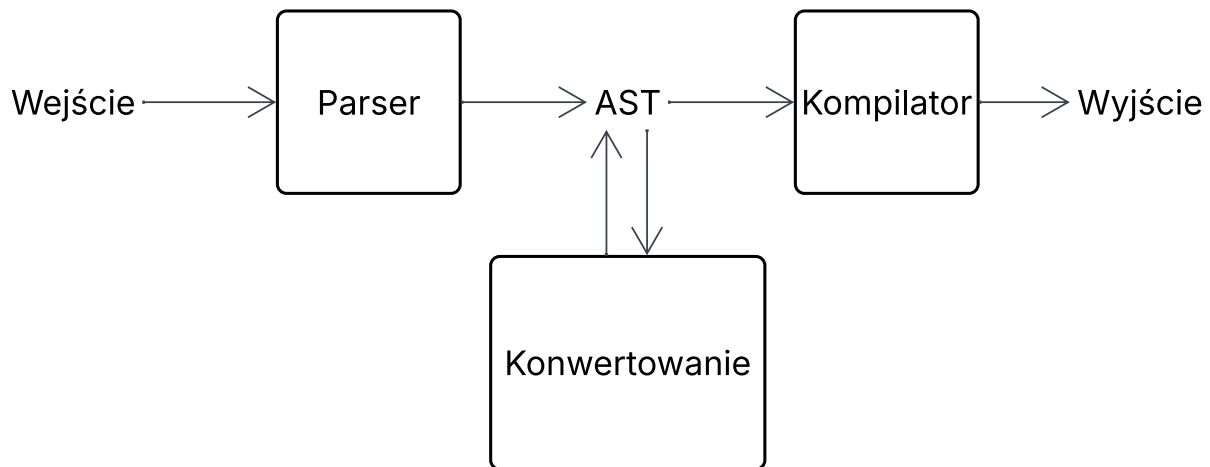
5.2.3.4 Biblioteka Unified

W celu uniezależnienia formatu treści przepisów od zastosowanego edytora, zdecydowano o utrzymaniu tekstu sformatowanego w sposobie przygotowania w formacie Markdown. Edytor TipTap jednak nie obsługuje natywnie tego formatu – dostępne opcje to HTML, JSON lub zwykły tekst. Aby móc wyeksportować zawartość do Markdown, zastosowano bibliotekę Unified.

Według [56] Unified to interfejs służący do przetwarzania i procesowania kodu za pomocą struktury drzew składniowych (ang. *Syntax Tree*). Biblioteka pozwala na proste, ale i zaawansowane manipulowanie drzewem i konwertowanie kodu za pomocą wtyczek.

W aplikacji webowej utworzono wewnętrzną bibliotekę `html-markdown-converter.ts`. Oprócz wspierania typowych dla HTML i Markdown znaczników, obsługuje ona własne, niestandardowe elementy, jak oznaczenia składników czy ramki z odtwarzaczami wideo do sieci społecznościowych.

Rysunek 46 prezentuje schemat konwertowania za pomocą biblioteki Unified.



Rysunek 46 Schemat konwertowania kodu za pomocą biblioteki Unified. Źródło: [56]

5.2.3.5 Biblioteka TanStack Table

Według [57] biblioteka TanStack Table pozwala na łatwe przekształcanie danych w rozbudowane tabele. Zapewnia przy tym wbudowaną obsługę paginacji, filtrów, sortowania czy wyszukiwania. W projekcie wykorzystano ją w panelu administracyjnym do prezentacji i zarządzania dodanymi obiektami w serwisie. Interfejs graficzny tabeli z dodatkową obsługą dostosowaną do projektu tworzą razem autorski komponent `ResultTable`.

Listing 33 przedstawia implementację tabeli `CategoriesTable` z listą kategorii za pomocą komponentu `ResultTable` korzystającego z TanStack Table.

Listing 33 Komponent `CategoriesTable` prezentujący listę kategorii za pomocą komponentu `ResultTable` z TanStack Table. Źródło: Opracowanie własne.

```
// ...imports

const columnHelper = createColumnHelper<Category>();

export default function CategoriesTable() {
  const categoryService = getCategoryService(clientAxios);

  const fetchData = useCallback(
    async (params: FetchDataParams) => {
      return await categoryService.getCategories({
        searchPhrase: params.searchPhrase,
        pageNumber: params.pageIndex + 1,
        pageSize: params.pageSize,
        sortBy: params.sortBy,
        sortDirection: params.sortDirection,
      });
    },
    []
  );
}
```

```

// table columns
const columns = useMemo(
  () => [
    columnHelper.accessor("id", {
      header: "ID",
      enableSorting: false,
      cell: (info) => info.getValue(),
    }),
    columnHelper.accessor("name", {
      header: "Nazwa",
      cell: (info) => info.getValue(),
    }),
    columnHelper.accessor("homepageOrder", {
      header: "Strona główna",
      enableSorting: false,
      cell: (info) => info.getValue(),
    }),
    columnHelper.display({
      id: "actions",
      header: "Akcje",
      enableSorting: false,
      cell: (info) => (
        <AdminManageCategoryButtonGroup
          category={info.row.original}
        />
      ),
    }),
  ],
  []
);

return (
  <ResultTable<Category>
    columns={columns}
    fetchData={fetchData}
    initialSorting={[{ id: "name", desc: false }]}
    paginationMode="server"
  />
);
}

```

5.2.3.6 Biblioteka React Hook Form

Biblioteka React Hook Form [58] została użyta do komponowania i walidacji formularzy na stronach. Dzięki niej walidacje i komunikaty są zdefiniowane bezpośrednio przy polach, unikamy dużej ilości state'ów, a obsługa wysyłki i błędów jest wbudowana i spójna w całym formularzu.

Tworzenie formularza odbywa się według etapów poniżej:

1. Utworzenie nowej instancji biblioteki i pomocniczych zmiennych,
2. Zdefiniowanie funkcji wywoływanej po pomyślnym wypełnieniu formularza,
3. Kod formularza w HTML razem z nazwami i walidacjami pól.

Listing 34 przedstawia przykładową inicjalizację nowego formularza w React Hook Form, a Listing 35 fragment formularza rejestracji nowego konta w *Let Me Cook* z użyciem tej biblioteki.

Listing 34 Przykład inicjalizacji nowego formularza w React Hook Form. Źródło: Opracowanie własne.

```
const {
  register,
  handleSubmit,
  watch,
  setError,
  formState: { errors, isSubmitting },
} = useForm();
const password = watch("password");
const onSubmit = async (data: any) => { /* ... */ }
```

Listing 35 Fragment formularza rejestracji z użyciem React Hook Form. Źródło: Opracowanie własne.

```
<Form
  noValidate
  className="d-flex flex-column gap-3"
  onSubmit={handleSubmit(onSubmit)}
  method="POST"
  action="#"
>
  <FormGroup>
    <FormLabel htmlFor="username">Nazwa użytkownika</FormLabel>
    <FormControl
      type="text"
      id="username"
      isValid={!errors.username}
      {...register("username", {
        required: "Nazwa użytkownika nie może być pusta",
        minLength: {
          value: 3,
          message: "Nazwa użytkownika musi mieć min. 3 znaki",
        },
        maxLength: {
          value: 50,
          message: "Nazwa użytkownika może mieć max. 50 znaków",
        },
      })}
    />
    <FormControl.Feedback type="invalid">
      {errors.username?.message as string}
    </FormControl.Feedback>
  </FormGroup>

  <FormGroup>
    <FormLabel htmlFor="email">Adres e-mail</FormLabel>
    <FormControl
      type="email"
      id="email"
```

```

        isValid={!!errors.email}
        {...register("email", {
            required: "Adres e-mail nie może być pusty",
            pattern: {
                value: /^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i,
                message: "Proszę podać poprawny adres e-mail",
            },
        })}
    />
    <FormControl.Feedback type="invalid">
        {errors.email?.message as string}
    </FormControl.Feedback>
</FormGroup>

<FormGroup>
    <FormLabel htmlFor="password">Hasło</FormLabel>
    <FormControl
        type="password"
        id="password"
        isValid={!!errors.password}
        {...register("password", {
            required: "Hasło jest wymagane",
            minLength: {
                value: 8,
                message: "Hasło musi mieć min. 8 znaków",
            },
            maxLength: {
                value: 100,
                message: "Hasło może mieć max. 100 znaków",
            },
        })}
    />
    <FormControl.Feedback type="invalid">
        {errors.password?.message as string}
    </FormControl.Feedback>
</FormGroup>

<FormGroup>
    <FormLabel htmlFor="confirmPassword">
        Potwierdź hasło
    </FormLabel>
    <FormControl
        type="password"
        id="confirmPassword"
        isValid={!!errors.confirmPassword}
        {...register("confirmPassword", {
            required: "Potwierdzenie hasła jest wymagane",
            validate: (value) =>
                value === password || "Hasła nie pasują",
        })}
    />
    <FormControl.Feedback type="invalid">

```

```
        {errors.confirmPassword?.message as string}
    </FormControl.Feedback>
</FormGroup>
<Button type="submit" disabled={isSubmitting}>
    {isSubmitting ? <Spinner /> : "Zarejestruj się"}
</Button>
</Form>
```

5.3. Konteneryzacja i środowisko uruchomieniowe

Oprócz kodu źródłowego aplikacji webowej oraz aplikacji serwerowej, struktura repozytorium projektu zawiera pełną konfigurację środowiskową opartą na platformie Docker. Wykorzystanie konteneryzacji pozwoliło na uniezależnienie procesu uruchamiania serwisu od konfiguracji maszyny gospodarza.

Dzięki przygotowanym plikom konfiguracyjnym `docker-compose.yml` oraz plikom zmiennych środowiskowych (`.env`), proces uruchamiania środowiska został znacząco uproszczony. Wystarczy do tego instalacja narzędzia Docker i wykonanie pojedynczego polecenia `docker compose up`.

Niniejszy podrozdział szczegółowo opisuje architekturę utworzonego środowiska deweloperskiego oraz produkcyjnego. Przedstawiono kompozycję plików konfiguracyjnych, rolę zmiennych środowiskowych i przeanalizowano komunikację sieciową między kontenerami.

5.3.1 Środowiska uruchomieniowe

Środowiska uruchomieniowe aplikacji stanowią kluczowy element procesu wytwarzania, testowania i utrzymania oprogramowania. Określają one zestaw usług i narzędzi, które są potrzebne do poprawnego działania serwisu.

W niniejszej pracy wyróżniono dwa środowiska:

- Środowisko produkcyjne – finałowa, publiczna wersja aplikacji, z której korzystają użytkownicy docelowi, zawierająca prawdziwe dane i tworząca wartość biznesową,
- Środowisko deweloperskie – bezpieczna i eksperymentalna przestrzeń przeznaczona do tworzenia i testowania nowych funkcjonalności, w odizolowaniu od środowiska produkcyjnego, z dodatkowymi narzędziami do debugowania.

Uruchamianie środowiska następuje poprzez platformę Docker. Na konfigurację składają się charakterystyczne dla tej platformy pliki:

- `docker-compose.yml` – opisuje kolejność uruchamiania i konfigurację poszczególnych serwisów, a także porty i wolumeny danych, do których serwisy mają mieć dostęp,
- `Dockerfile` – zawiera instrukcję dla Dockera jak utworzyć obraz poszczególnego mikroservisu lub narzędzia i jak ono ma działać.

Powyższe pliki uruchamiają środowisko produkcyjne. Do uruchomienia środowiska deweloperskiego służą ich odpowiedniki z rozszerzeniem `.dev`: `docker-compose.dev.yml` i `Dockerfile.dev`.

Komponenty systemu są uruchamiane według następujących zależności:

1. **Infrastruktura:** Microsoft SQL Server i MinIO startują niezależnie,
2. **Mikroservisy backendowe:**

- a. Mikroserwis multimediiów (MediaService) startuje po MinIO,
- b. Mikroserwis użytkowników (UserService) startuje po Microsoft SQL Server,
- c. Mikroserwis przepisów (RecipesService) startuje po Microsoft SQL Server i mikroserwisie użytkowników,
- d. API Gateway startuje po Microsoft SQL Server i mikroserwisie przepisów,

3. **Frontend** startuje po API Gateway.

Aby uruchomić aplikację, należy najpierw upewnić się, że usługa Dockera jest aktywna. Następnie z poziomu wiersza poleceń trzeba przejść do katalogu projektu i wpisać komendę:

- Dla środowiska produkcyjnego:
docker-compose --env-file db.env up -d --build
- Dla środowiska deweloperskiego:
docker-compose --env-file db.env -f docker-compose.dev.yml up -d --build

Po uruchomieniu środowiska otwierane są dwa rodzaje portów: zewnętrzne (na maszynie hosta) oraz wewnętrzne (w kontenerze), które zostają ze sobą zmapowane. Tabela 9 przedstawia zestawienie otwartych portów poszczególnych komponentów na zewnątrz i wewnątrz kontenerów według rodzaju środowiska.

Tabela 9 Zestawienie otwartych portów aplikacji według środowiska oraz ich alokacji (host/kontener).

Źródło: Opracowanie własne.

Komponent	Środowisko produkcyjne		Środowisko deweloperskie	
	Host	Kontener	Host	Kontener
Aplikacja webowa	80	80	3000	3000
API Gateway	8443	8080	8080	8080
Baza danych MS SQL	1433	1433	1433	1433
MinIO	9000	9000	9000,9090	9000,9001
RecipesService	8081	8080	8081	8080
UserService	8082	8080	8082	8080
MediaService	8083	8080	8083	8080

5.3.2 Zmienne środowiskowe

Do utworzenia instancji bazy danych i uruchomienia środowiska wymagane jest utworzenie czterech plików .env zawierających wartości zmiennych środowiskowych dla poszczególnych komponentów: admin.env, aws.env, db.env oraz jwt.env. Pliki te zawierają poufne dane logowania i klucze dostępu. Wyizolowanie ich do plików .env służy uniknięciu przypadkowego przesłania ich do repozytorium Git poprzez użycie mechanizmu .gitignore. Dodatkowo, zmienne środowiskowe zostały rozdzielone do różnych plików ze względów bezpieczeństwa. Dzięki takiemu podziałowi poszczególne kontenery otrzymują dostęp wyłącznie do wartości tych zmiennych, które są im faktycznie potrzebne, co jest zgodne z zasadą najmniejszego uprzywilejowania.

W pliku db.env zdefiniowane jest hasło administratora bazy danych MS SQL pod loginem „sa”. To hasło wykorzystywane jest najpierw przez serwer bazy danych do zabezpieczenia domyślnego konta, a następnie przez mikroserwisy RecipesService i UserService do połączenia z bazą.

Listing 36 Przykładowa konfiguracja pliku db.env (dane zanonimizowane). Źródło: Opracowanie własne.

```
SA_PASSWORD=<UKRYTE>
```

Plik `admin.env` definiuje inicjalny adres e-mail i hasło SuperAdmina w serwisie *Let Me Cook* poprzez zmienne `ADMIN_EMAIL` i `ADMIN_PASSWORD`. Poświadczenia są odczytywane przez mikroserwis użytkowników przy inicjalizacji bazy danych.

Listing 37 Przykładowa konfiguracja pliku admin.env (dane zanonimizowane). Źródło: Opracowanie własne.

```
ADMIN_EMAIL=admin@letmecook.pl  
ADMIN_PASSWORD=<UKRYTE>
```

Do konfiguracji usługi MinIO oraz uwierzytelnienia połączenia z poziomu mikroserwisu `MediaService` wykorzystywany jest plik `aws.env`. Określa on dane logowania do panelu administracyjnego MinIO poprzez zmienne `MINIO_ROOT_USER` i `MINIO_ROOT_PASSWORD` oraz identyfikator i klucz dostępu dla mikroserwisów do autoryzacji żądań (`AWS_ACCESS_KEY_ID` oraz `AWS_SECRET_ACCESS_KEY`).

Listing 38 Przykładowa konfiguracja pliku aws.env (dane zanonimizowane). Źródło: Opracowanie własne.

```
MINIO_ROOT_USER=admin  
MINIO_ROOT_PASSWORD=<UKRYTE>  
  
AWS_ACCESS_KEY_ID=admin  
AWS_SECRET_ACCESS_KEY=<UKRYTE>
```

Za konfigurację uwierzytelniania opartego na tokenach JWT odpowiada plik `jwt.env`. Parametr `Jwt__Key` określa tajny ciąg znaków służący do tworzenia kryptograficznego podpisu tokena. Wystawcę tokenu definiuje parametr `Jwt__Issuer`, a jego odbiorcę – `Jwt__Audience`. Zmienne są wykorzystywane przez mikroserwis użytkowników (do wystawiania tokenów) oraz API Gateway (do weryfikowania tokenów).

Listing 39 Przykładowa konfiguracja pliku jwt.env (dane zanonimizowane). Źródło: Opracowanie własne.

```
Jwt__Key=<UKRYTE>  
Jwt__Issuer=http://users-service  
Jwt__Audience=api-gateway
```

6. Interfejs i funkcjonalności aplikacji

W niniejszym rozdziale szczegółowo omówiono interfejs graficzny oraz zestaw funkcjonalności platformy „Let Me Cook”. Wszystkie wymagania funkcjonalne określone w rozdziale 3.2 zostały odwzorowane na konkretne widoki i akcje w warstwie frontendowej. Aplikacja jest w pełni responsywna, a więc wygląda i działa prawidłowo niezależnie od urządzenia, z którego korzysta użytkownik.

Cała warstwa wizualna systemu została zaprojektowana tak, aby jak najlepiej wspierać użytkownika w osiąganiu jego celów. Została ona zrealizowana zgodnie z wytycznymi UX (ang. *User Experience*) oraz UI (ang. *User Interface*), co przekłada się również na przejrzystość i łatwość obsługi aplikacji. Zastosowanie jednej, określonej palety barw, w której jasne tło kontrastuje z czerwonymi elementami miało na celu skierowanie uwagi użytkownika na najważniejsze komponenty interfejsu oraz ujednolicenie go we wszystkich widokach.

6.1. Nawigacja w systemie

Układ nawigacyjny w serwisie „Let Me Cook” opracowano tak, aby był on jak najbardziej przejrzysty i zapewniał stały dostęp do najważniejszych funkcji systemu. Podstawowymi elementami odpowiedzialnymi za poruszanie się po systemie są główny panel nawigacyjny oraz stopka obecne w każdym widoku systemowym.

6.1.1 Główny panel nawigacyjny

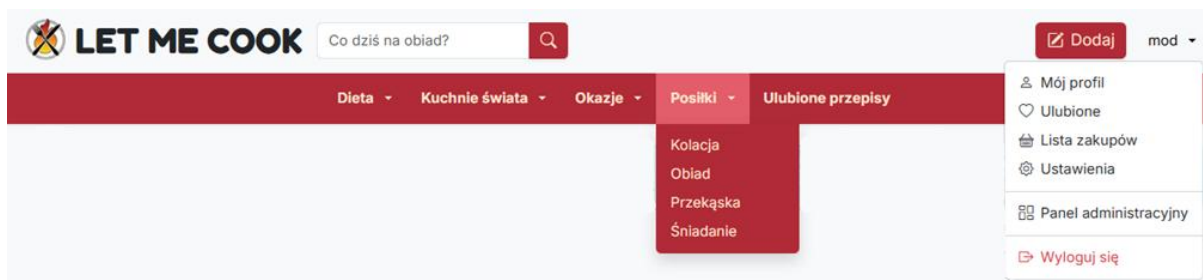
Główny panel nawigacyjny serwisu umiejscowiony jest zawsze na samej górze ekranu. Ilość elementów, a także ich działanie nie różni się w zależności od rozdzielczości ekranu urządzenia. Są jednak pewne różnice w rozmieszczeniu komponentów. Wynikają one z potrzeby dopasowania wyglądu serwisu do urządzenia końcowego użytkownika, aby korzystanie z platformy było możliwie najwygodniejsze i najbardziej intuicyjne.

6.1.1.1 Wersja desktopowa

W wersji desktopowej panel nawigacyjny składa się z dwóch sekcji. Jego pierwsza część, z jasnym tłem, zawiera następujące elementy:

- Logo serwisu,
- Moduł wyszukiwania,
- Przycisk „Dodaj”,
- Przycisk „Zaloguj się” lub nazwę zalogowanego użytkownika.

Rysunek 47 przedstawia główny panel nawigacyjny na ekranie o wysokiej rozdzielczości z rozwiniętymi podrzędnymi menu kategorii oraz opcji dla użytkownika z rolą moderatora.



Rysunek 47 Główny panel nawigacyjny w wersji desktopowej. Źródło: Opracowanie własne.

Elementami wyrównanymi do lewej strony, na wyższym poziomie panelu są logo oraz wyszukiwarka. Logo pełni rolę odsyłacza do strony głównej z dowolnego miejsca w aplikacji. Nieco bardziej po prawej stronie znajduje się pasek wyszukiwania składający się z pola tekstowego oraz przycisku z ikoną lupy. Po wpisaniu frazy związanej z poszukiwanymi przepisami i kliknięciu przycisku użytkownik przekierowywany jest do wyszukiwarki, opisaną szczegółowo w rozdziale 6.2.2.3.

Pozostałe elementy górnej części panelu nawigacyjnego wyrównane są do jego prawej strony. Przycisk „Dodaj” przekierowuje zalogowanego użytkownika do ekranu dodawania przepisu, omówionego dokładnie w rozdziale 6.3.1.1, natomiast gościa do widoku logowania. Jego stała obecność w głównym panelu nawigacyjnym ma zachęcać odwiedzających do dzielenia się własnymi pomysłami kulinarnymi na platformie. Najbardziej na prawo umieszczono przycisk zmieniający swoją zawartość oraz funkcję w zależności od tego, czy użytkownik jest zalogowany. Jeśli odwiedzający nie ma aktywnej sesji logowania wyświetlany jest tam odsyłacz z tekstem „Zaloguj się”, przekierowujący do formularza logowania opisanego w rozdziale 6.2.1. W przeciwnym wypadku – jeżeli użytkownik jest zalogowany – w tym miejscu widnieje nazwa jego konta z ikoną strzałki. Po kliknięciu tego przycisku przez uwierzytelnionego użytkownika, wyświetlany jest panel z przekierowującymi na odpowiednie podstrony opcjami:

- Mój profil,
- Ulubione,
- Lista zakupów,
- Ustawienia,
- Panel administracyjny (tylko w przypadku użytkownika z odpowiednimi uprawnieniami),
- Wyloguj się.

Uzupełnieniem panelu nawigacyjnego jest ciemnoczerwony pasek z grupami kategorii oraz odnośnikiem do ulubionych przepisów, umieszczony bezpośrednio pod pierwszą, wyżej opisaną częścią. Najechnie kursorem na jedną z grup rozwija podrzędne menu, zawierające przyporządkowane do niej tematycznie kategorie przepisów. Kliknięcie na jedną z kategorii skutkuje przekierowaniem użytkownika do widoku wyszukiwarki z przepisami do niej należącymi.

6.1.1.2 Wersja na urządzenia mobilne

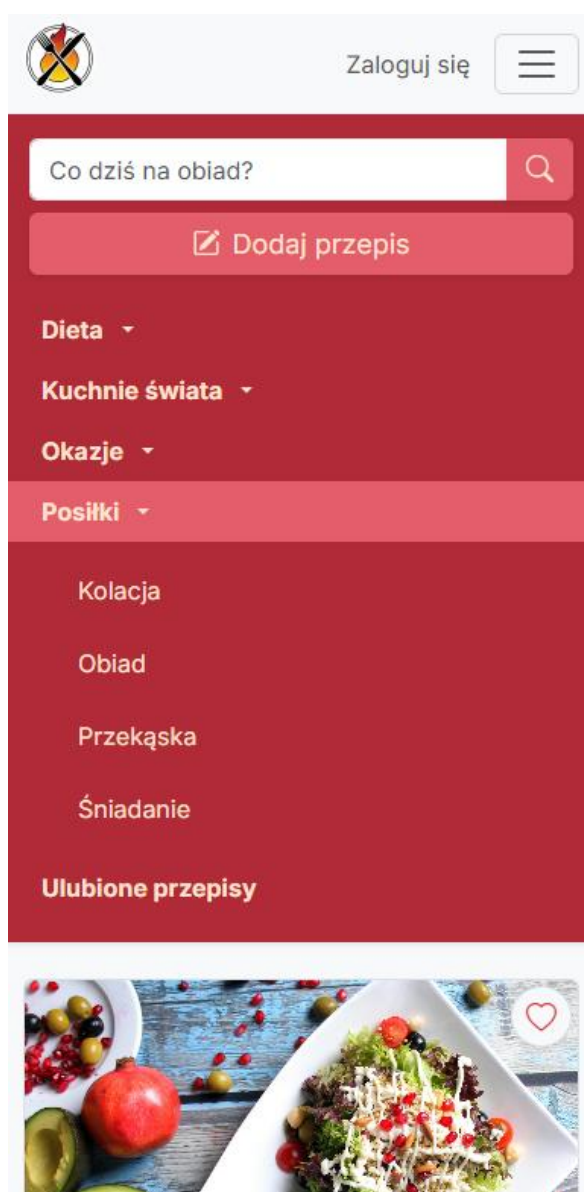
W przypadku wersji serwisu na urządzenia mobilne domyślnie na górze ekranu obecna jest tylko pierwsza część panelu nawigacyjnego, z jasnym tłem. Umieszczono w niej następujące elementy:

- Logo serwisu,
- Przycisk „Zaloguj się” lub przycisk rozwijający opcje dla zalogowanych użytkowników,
- Przycisk rozwijający menu z innymi komponentami.

Największą zmianą względem wersji na urządzenia z ekranami o wysokiej rozdzielczości jest tutaj przycisk służący do rozwijania dodatkowego menu. Pełni ono rolę dolnego, ciemnoczerwonego paska z wersji desktopowej z dwoma komponentami przeniesionymi z górnej belki. Menu pozostaje widoczne aż do ponownego wciśnięcia przycisku w prawym górnym rogu. W jego skład wchodzi następujące elementy:

- Moduł wyszukiwania (przeniesiony z głównej części panelu nawigacyjnego),
- Przycisk „Dodaj przepis” (przeniesiony z głównej części panelu nawigacyjnego),
- Lista grup kategorii,
- Odnosnik do ulubionych przepisów.

Rysunek 48 przedstawia panel nawigacyjny z rozwiniętym menu w wersji na urządzenia mobilne.



Rysunek 48 Główny panel nawigacyjny w wersji na urządzenia mobilne. Źródło: Opracowanie własne.

6.1.2 Stopka

Stopka jest elementem interfejsu umiejscowionym na samym dole każdego ekranu w aplikacji. Dzieli się ona na dwie części. Pierwsza z nich zawiera napis „Mamy przepis na wszystko” oraz sugerowane kategorie, będące odnośnikami do widoku wyszukiwarki zawierającego przepisy należące do wybranej kategorii. Kategorie ułożone są w kolumnach, których ilość zależy od rozdzielczości ekranu użytkownika. Sekcja ta posiada charakterystyczne tło w kratę, przypominające obrus.

Dolny segment stopki wyróżnia się jednolitym, czerwonym tłem i zawiera podstawowe informacje o platformie. W górnej części tej sekcji znajduje się logo serwisu i adnotacja o prawach autorskich. Na samym dole umieszczono linki do następujących podstron informacyjnych:

- Kontakt,
- Regulamin,
- O nas.

W wersji desktopowej wymienione odnośniki rozłożone są poziomo, natomiast na urządzeniach mobilnych ich układ jest pionowy. Należy zaznaczyć, że powyższe podstrony nie istnieją w bieżącej wersji systemu, a odnośniki pełnią wyłącznie pogładową funkcję. Rysunek 49 przedstawia wygląd stopki na ekranach o wysokiej rozdzielczości.



Rysunek 49 Stopka w wersji desktopowej. Źródło: Opracowanie własne.

6.2. Użytkownik niezalogowany

W opisywanym systemie użytkownik niezalogowany (nazywany też gościem) – poza możliwością rejestracji i zalogowania – dysponuje wyłącznie uprawnieniami do odczytu ogólnodostępnych zasobów. Pozwala to nowym odwiedzającym na bezproblemowe przeglądanie w serwisie takich treści jak opublikowane przepisy i komentarze pod nimi, a także profile użytkowników. Goście mogą również korzystać z wyszukiwarki przepisów z zaawansowanymi filtrami. Z kolei operacje powodujące zmiany w bazie danych są zablokowane dla użytkowników niezalogowanych. Każda próba wykonania akcji wymagającej uwierzytelnienia jest płynnie obsługiwana przez frontend, który przekierowuje gościa do widoku logowania.

6.2.1 Rejestracja i logowanie do systemu

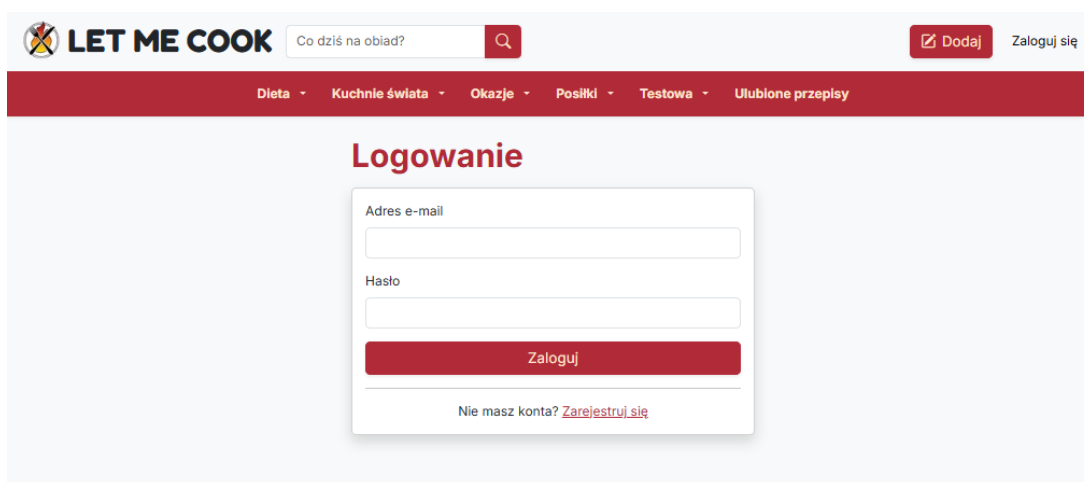
Warunkiem korzystania z rozszerzonych możliwości aplikacji jest autentykacja użytkownika. Przycisk przekierowujący do widoku logowania umieszczono w głównym menu nawigacyjnym, dzięki czemu jest on dostępny z każdego miejsca w systemie. Od strony wizualnej formularze logowania i rejestracji charakteryzują się centralnym układem elementów i minimalistycznym designem, co pozwala

użytkownikowi skupić się wyłącznie na procesie uwierzytelniania w aplikacji. W dolnej części obu formularzy znajduje się przycisk służący do wykonania akcji zarejestrowania bądź zalogowania, po wciśnięciu którego dane są wstępnie walidowane, a następnie odpowiednie żądanie przesyłane jest do serwerowej warstwy systemu.

Formularz logowania opiera się na standardowym uwierzytelnianiu za pomocą adresu e-mail oraz hasła. Aby pomyślnie zalogować się do systemu należy spełnić następujące warunki:

- Podać poprawny adres e-mail przypisany do konta,
- Podać poprawne hasło,
- Konto powiązane z podanym adresem e-mail nie może być zablokowane.

Na samym dole formularza logowania znajduje się odsyłacz do ekranu rejestracji w postaci wyróżnionego tekstu „Zarejestruj się”. Widok logowania do omawianego systemu zaprezentowano na Rysunek 50.



Rysunek 50 Widok formularza logowania. Źródło: Opracowanie własne.

W przypadku tworzenia nowego konta konieczne jest podanie nazwy użytkownika, adresu e-mail oraz dwukrotne wprowadzenie hasła, w celu zminimalizowania ryzyka pomyłki. Żeby pomyślnie przejść proces rejestracji, użytkownik musi:

- Podać nazwę użytkownika mającą pomiędzy 3 a 50 znaków,
- Podać poprawny adres e-mail,
- Podać nazwę użytkownika oraz e-mail nieprzypisane do żadnego istniejącego w systemie konta,
- Podać hasło mające od 8 do 100 znaków,
- Wpisać takie same wartości w pole hasła oraz potwierdzenia hasła.

Pod formularzem rejestracji umieszczono tekst „Zaloguj się”, będący odsyłaczem do ekranu logowania. Na Rysunek 51 przedstawiony został widok rejestracji użytkownika w serwisie „Let Me Cook”.

LET ME COOK Co dziś na obiad?

Dieta - Kuchnie świata - Okazje - Posiłki - Testowa - Ulubione przepisy

Załącz konto

Nazwa użytkownika

Adres e-mail

Hasło

Potwierdź hasło

Masz już konto? [Zaloguj się](#)

Rysunek 51 Widok formularza rejestracji użytkownika. Źródło: Opracowanie własne.

6.2.2 Przeglądanie przepisów

W tym rozdziale omówiono funkcjonalności i elementy interfejsu umożliwiające eksplorację bazy kulinarnej w serwisie „Let Me Cook”. Poniżej szczegółowo przedstawiono układ strony głównej, ekran z detalami konkretnego przepisu, a także wyszukiwarkę przepisów w opisywanym systemie.

6.2.2.1 Strona główna serwisu

Strona główna jest pierwszym widokiem w serwisie, z jakim użytkownik ma styczność. Na samej górze ekranu umiejscowiona jest duża karuzela z proponowanymi przepisami. Dla każdego wyróżnionego w tym komponencie dania wyświetlane jest jego zdjęcie oraz podstawowe informacje. Proponowane przepisy można przewijać klikając strzałki znajdujące się po lewej oraz po prawej stronie karuzeli.

W dalszej części ekranu zastosowano układ oparty na poziomych listach, które segregują przepisy według kategorii. Każdy taki element listujący dania zawiera nazwę kategorii, z której pochodzą oraz przycisk „Zobacz więcej”, który przekierowuje do widoku zawierającego wszystkie przepisy z danej kategorii. Dania z wybranej kategorii można przewijać strzałkami, analogicznie jak w przypadku karuzeli na górze strony. Pojedyncze przepisy wyświetlane są w formie przejrzystych kart, na których widnieją ich zdjęcia oraz podstawowe informacje, takie jak:

- Tytuł,
- Średnia ocen,
- Krótki opis,
- Autor.

Na karcie każdego przepisu, w jej prawym górnym rogu, znajduje się też przycisk z ikonką serca, po kliknięciu którego danie zostanie dodane do ulubionych w przypadku użytkownika zalogowanego. Przycisk ten widoczny jest również w przypadku gościa, jednak po jego kliknięciu użytkownik niezalogowany zostanie przekierowany do widoku logowania.

6.2.2.2 Widok szczegółów przepisu

Ekran szczegółów przepisu został zaprojektowany tak, aby dostarczał użytkownikowi wszystkich niezbędnych informacji o konkretnym daniu. Jego układ rozplanowano tak, aby na górze znajdowały się najbardziej ogólne informacje o przepisie, natomiast w dalszej części były detale dotyczące przygotowania dania.

6.2.2.2.1. Wersja desktopowa

W widoku dla komputerów osobistych, w najwyższej części ekranu znajduje się duży nagłówek będący tytułem przepisu. Z jego prawej strony umieszczono zdjęcie oraz nazwę autora potrawy. Najbardziej na prawo na tym poziomie osadzono średnią ocenę w postaci pięciostopniowej skali z dokładnością do połowy gwiazdki. Bezpośrednio pod tytułem zlokalizowano tagi kategorii, do których danie zostało przypisane. Po kliknięciu któregoś z tagów, użytkownik jest przekierowywany do widoku wyszukiwarki zawierającego przepisy należące do wybranej kategorii. Na tej samej wysokości, ale po prawej stronie umieszczono cztery przyciski akcji, pozwalające kolejno na:

- Wydrukowanie przepisu,
- Udostępnienie przepisu,
- Ocenienie przepisu (tylko dla zalogowanych użytkowników),
- Dodanie przepisu do ulubionych (tylko dla zalogowanych użytkowników).

Najwięcej miejsca w górnej partii ekranu zajmuje panel zawierający zdjęcia oraz podstawowe informacje o przepisie. Większą część panelu wypełnia segment z fotografiami dania. Użytkownik może przewijać je na trzy sposoby:

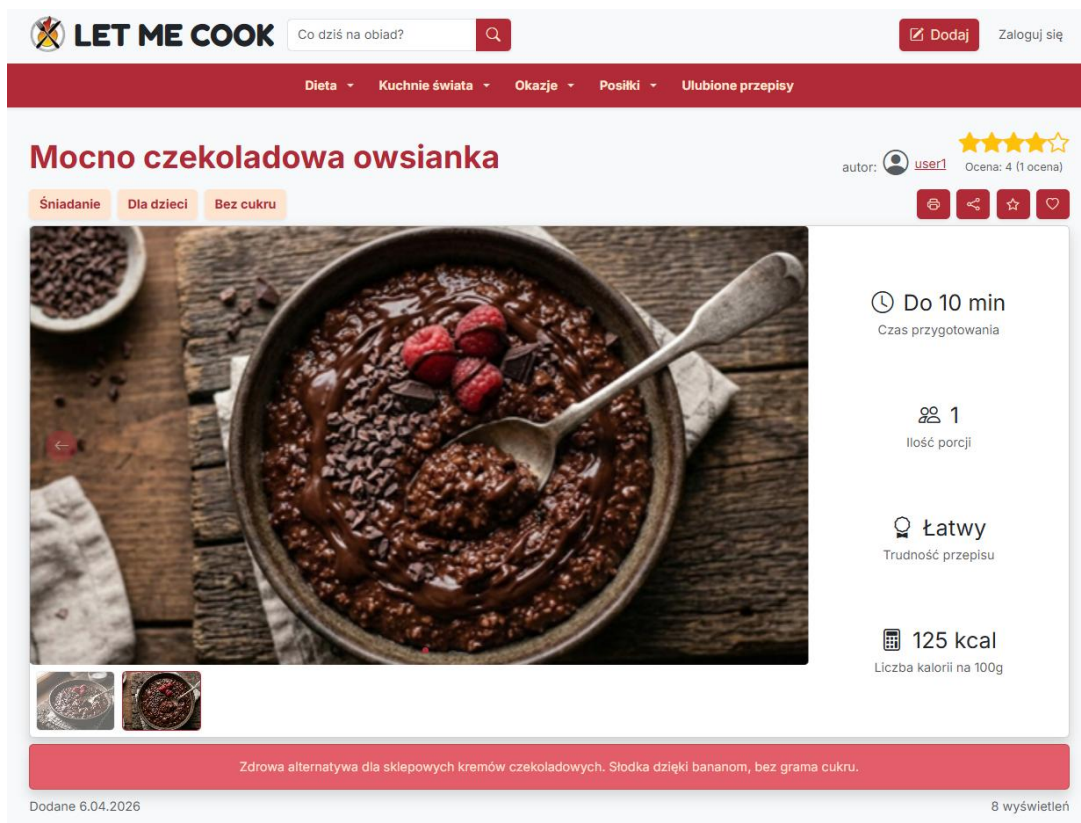
- Klikając przyciski ze strzałkami z lewej oraz prawej strony zdjęcia,
- Klikając kropki w dolnej części zdjęcia,
- Klikając miniaturowe zdjęcia w dolnej części panelu.

Przestrzeń po prawej stronie panelu zagospodarowano na najważniejsze parametry potrawy, takie jak:

- Czas przygotowania,
- Ilość porcji,
- Trudność przepisu,
- Liczba kalorii na 100 gramów.

Pod panelem ze zdjęciem i podstawowymi informacjami znajduje się komponent z jasnoczerwonym tłem zawierający krótki opis dania. Bezpośrednio pod tym elementem znajduje się wyrównana do lewej strony data publikacji oraz liczba wyświetleń przepisu po przeciwnej stronie.

Górną część widoku szczegółów przepisu na urządzeniach z ekranami o wysokiej rozdzielczości przedstawiono na Rysunek 52.



Rysunek 52 Górna część widoku szczegółów przepisu w wersji desktopowej. Źródło: Opracowanie własne

W dalszej części ekranu umiejscowiono komponenty zawierające znacznie bardziej szczegółowe informacje dotyczące przygotowania przepisu. Pierwszym elementem po wyżej opisanych jest panel ze składnikami potrzebnymi do przyrządzenia dania. Patrząc od góry widać nazwę segmentu – „Składniki” – wyrównaną do lewej oraz wyrównane do prawej strony przyciski umożliwiające kolejno:

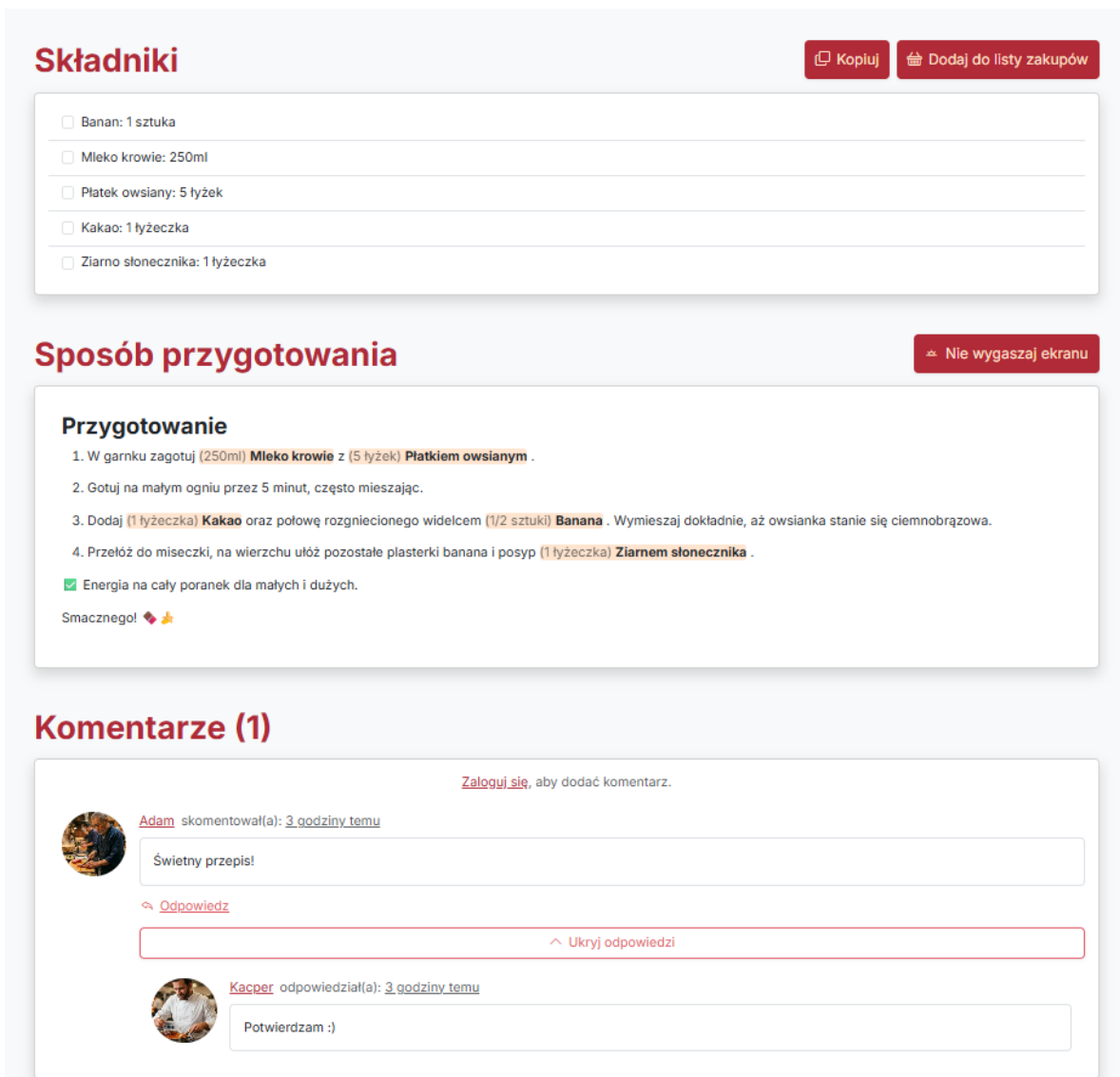
- Skopiowanie listy składników do schowka,
- Dodanie składników do listy zakupów (tylko dla zalogowanych użytkowników).

Zestawienie potrzebnych do przygotowania dania produktów zostało zaimplementowane jako lista z polami wyboru (ang. *checkbox*), aby umożliwić użytkownikom odznaczanie składników, które udało mu się zgromadzić.

W dalszej części ekranu znajduje się komponent zatytułowany „Sposób przygotowania”. Nad właściwą recepturą znajduje się tu dodatkowo przycisk, który uniemożliwia blokowanie się urządzenia podczas gotowania. Sposób opisanie procedury przygotowania dania w tym segmencie zależny jest całkowicie od autora przepisu. W celu zwiększenia czytelności składniki i ich ilości są zawsze wyróżnione w tekście.

Na samym dole ekranu ze szczegółami przepisu umieszczono komponent z opiniami użytkowników zatytułowany „Komentarze” z łączną liczbą komentarzy zawartą przy nazwie sekcji. Przy każdym wpisie w tym segmencie widnieje nazwa użytkownika, który go opublikował oraz ile czasu temu został dodany. Jeżeli są odpowiedzi do komentarza, można je rozwinąć lub ukryć klikając podłużny pasek znajdujący się pod nim.

Rysunek 53 przedstawia dolną część widoku szczegółów przepisu w wersji desktopowej.



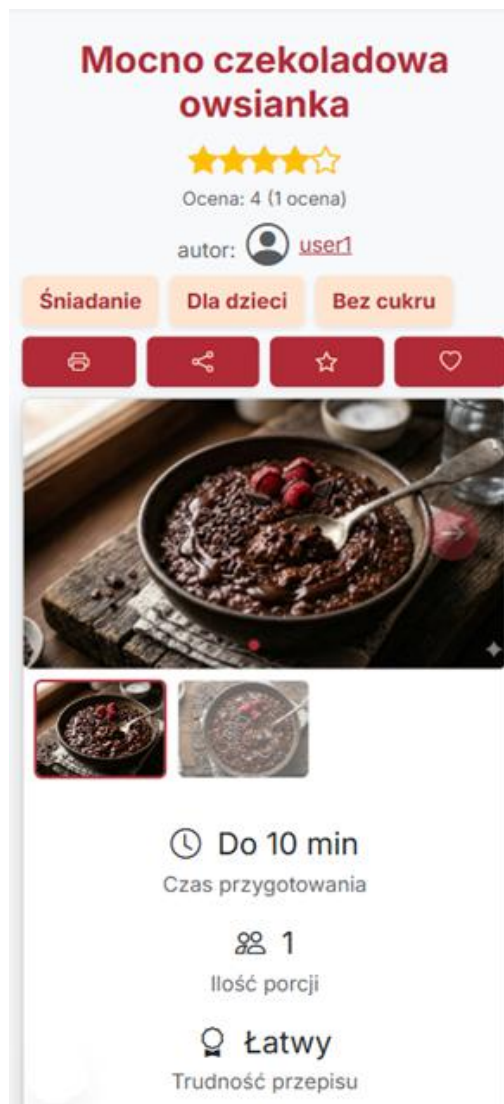
Rysunek 53 Dolna część widoku szczegółów przepisu w wersji desktopowej. Źródło: Opracowanie własne.

6.2.2.2.2. Wersja na urządzenia mobilne

W wersji na urządzenia mobilne układ w widoku szczegółów przepisu został znacznie bardziej skoncentrowany, aby pozostawić wszystkie funkcjonalności i jednocześnie zapewnić czytelność interfejsu. W górnej części ekranu tytuł, ocena oraz autor przepisu są wyśrodkowane. Tagi kategorii oraz przyciski akcji na ekranie smartfona zawijają się i zajmują dostępną szerokość w kilku wierszach.

Układ segmentu ze zdjęciami oraz podstawowymi parametrami przepisu na urządzeniach mobilnych również znacznie różni się względem wersji desktopowej. Część informacyjna została tu przeniesiona pod fotografie potrawy.

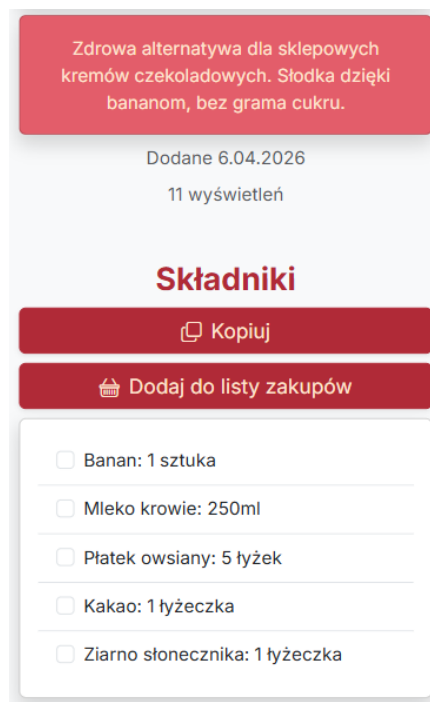
Na Rysunek 54 przedstawiona została górna część widoku szczegółów przepisu na urządzeniach mobilnych.



Rysunek 54 Górna część widoku szczegółów przepisu w wersji na urządzenia mobilne. Źródło: Opracowanie własne.

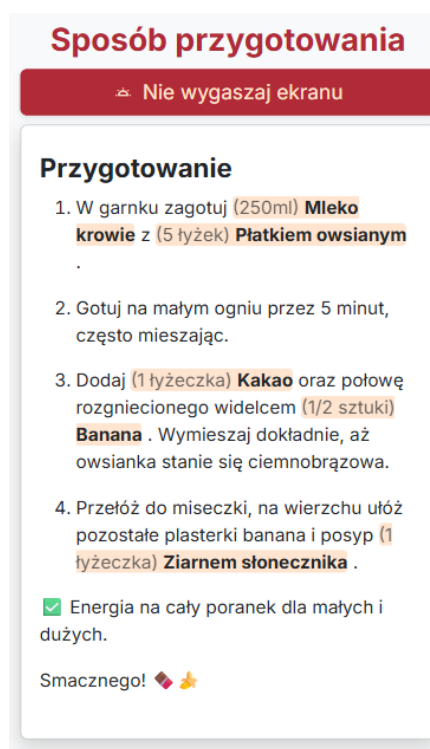
W dalszej części ekranu kolejne sekcje ułożone są w tej samej kolejności jak w desktopowej wersji aplikacji, jednak wszystkie komponenty zostały wycelowane i dopasowane do szerokości wyświetlacza. Przyciski „Kopiuj”, „Dodaj do listy zakupów” i „Nie wygaszaj ekranu” na ekranie smartfona zajmują całą jego szerokość, tak samo jak wszystkie sekcje w tym widoku.

Rysunek 55 przedstawia część z krótkim opisem i składnikami, Rysunek 56 segment ze sposobem przygotowania, a Rysunek 57 sekcję komentarzy w widoku szczegółów przepisu w wersji na urządzenia mobilne.



Rysunek 55 Część widoku szczegółów przepisu z krótkim opisem i składnikami w wersji na urządzenia mobilne.

Źródło: Opracowanie własne.



Rysunek 56 Sekcja „Sposób przygotowania” w widoku szczegółów przepisu w wersji na urządzenia mobilne.

Źródło: Opracowanie własne.



Rysunek 57 Sekcja „Komentarze” w widoku szczegółów przepisu w wersji na urządzenia mobilne. Źródło:

Opracowanie własne.

6.2.2.3 Wyszukiwanie przepisów

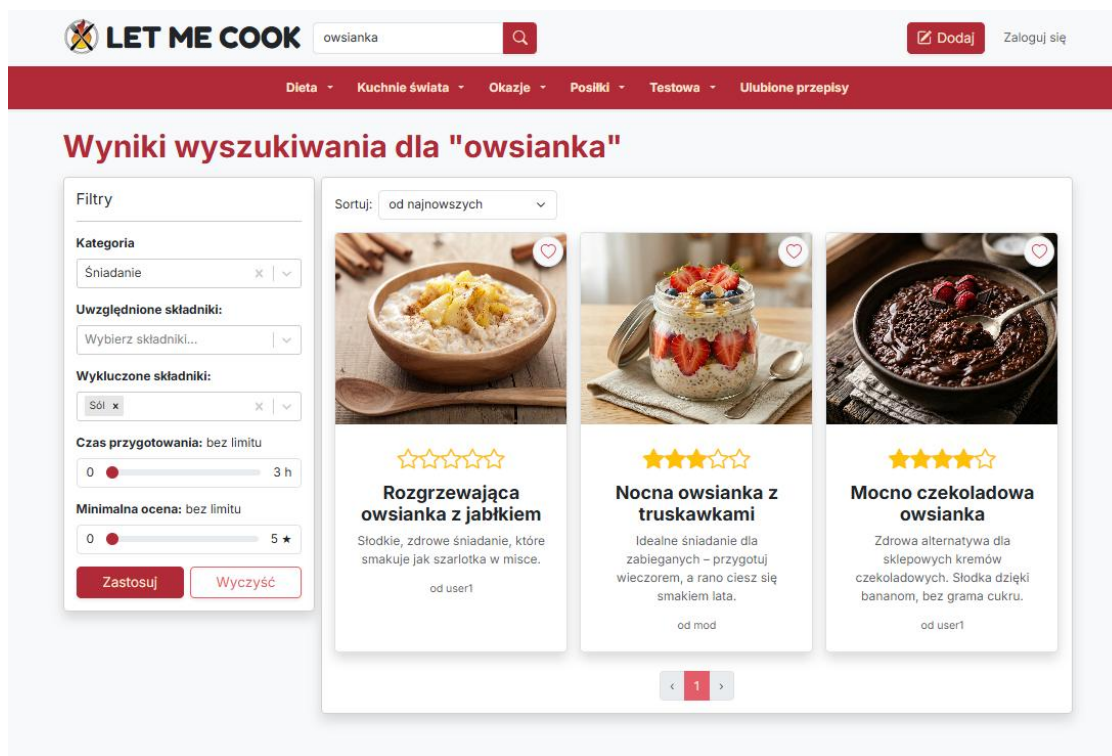
Aby ułatwić użytkownikom odnajdywanie konkretnych przepisów, system wyposażono w zintegrowany mechanizm wyszukiwania. Przejście do wyszukiwarki odbywa się poprzez wpisanie frazy związanej z poszukiwanym przepisem w pole wyszukiwania znajdujące się na górze, w głównym menu nawigacyjnym, a następnie naciśnięcie przycisku z lupą. Po wykonaniu takiej akcji użytkownik zostanie przeniesiony do widoku wyszukiwania, w którym wylistowane będą wyniki wyszukiwania na podstawie podanego tekstu. Na tym ekranie znajduje się również panel boczny, pozwalający na dalsze zawężanie listy przepisów, poprzez określenie wartości następujących filtrów:

- Kategoria,
- Uwzględnione składniki,
- Wykluczone składniki,
- Czas przygotowania,
- Minimalna ocena,
- Tylko od obserwowanych (wyłącznie dla zalogowanych użytkowników).

Wylistowane przepisy można również posortować wybierając jedną z wartości w polu „Sortuj” znajdującym się nad wynikami wyszukiwania. Dostępne są następujące sposoby sortowania wyników:

- Od najnowszych (domyślnie),
- Od najpopularniejszych,
- Od najwyżżej ocenianych,
- Od najszybszych.

W wyszukiwarce wyświetlanych jest dwanaście przepisów na stronę. Użytkownik ma możliwość przełączania stron z wynikami przy użyciu panelu z dostępnymi stronami, znajdującego się pod wylistowanymi przepisami. Rysunek 58 przedstawia widok wyszukiwania na ekranach o wysokiej rozdzielczości, natomiast Rysunek 59 na urządzeniach mobilnych.



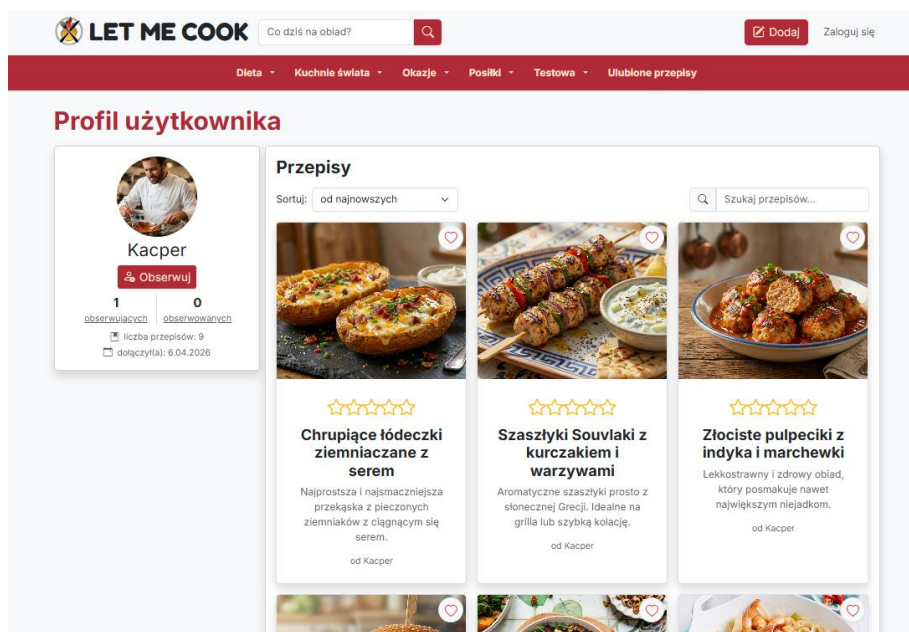
Rysunek 58 Widok wyszukiwania przepisów w wersji desktopowej. Źródło: Opracowanie własne.



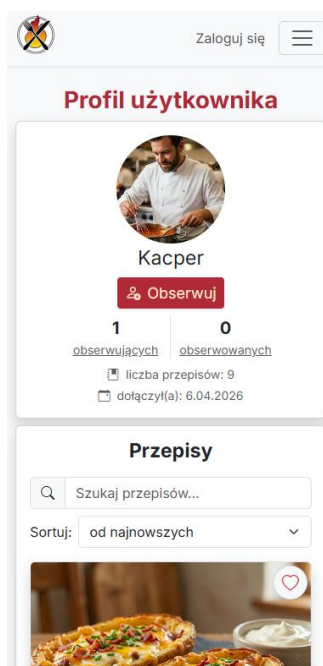
Rysunek 59 Widok wyszukiwania przepisów na urządzeniach mobilnych. Źródło: Opracowanie własne.

6.2.3 Przeglądanie profili użytkowników

System pozwala wszystkim odwiedzającym na przeglądanie profili użytkowników posiadających w nim konta. Przejście na widok profilu danego użytkownika odbywa się po kliknięciu w jego nazwę lub zdjęcie w dowolnym miejscu w aplikacji. Widok profilu, przedstawiony na Rysunek 60 w wersji desktopowej oraz na Rysunek 61 w wersji mobilnej dzieli się na dwie główne sekcje.



Rysunek 60 Widok profilu użytkownika w wersji desktopowej. Źródło: Opracowanie własne.



Rysunek 61 Widok profilu użytkownika na urządzeniach mobilnych. Źródło: Opracowanie własne.

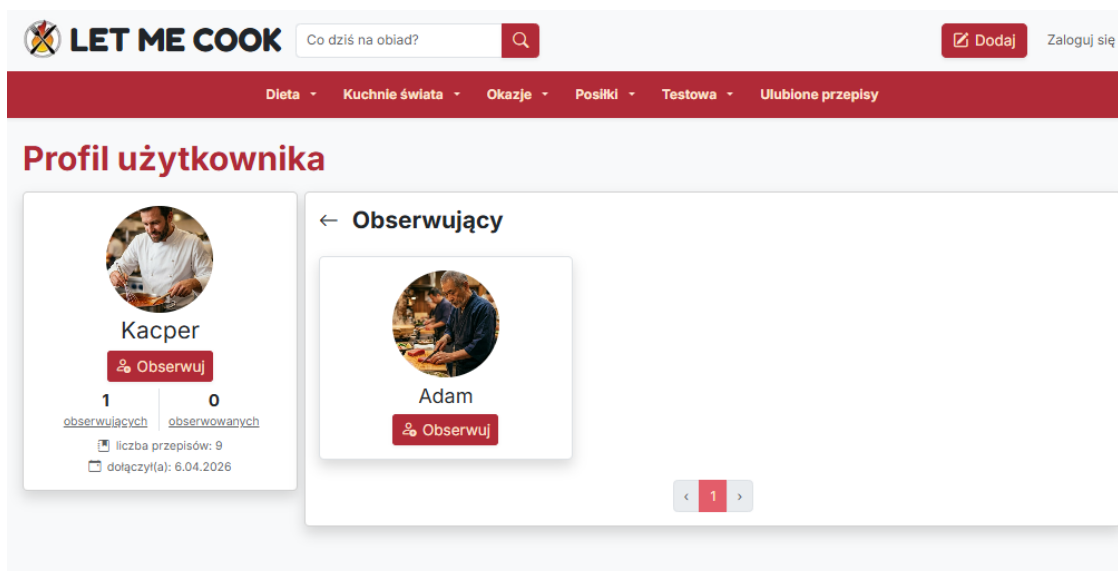
Panel znajdujący się po prawej stronie w wersji desktopowej na widoku profilu zawiera przepisy dodane przez tego użytkownika. Nad listą przepisów znajduje się pole z opcjami sortowania oraz pole umożliwiające wyszukiwanie opublikowanych przez użytkownika przepisów. W polu sortowania dostępne są następujące opcje:

- Od najnowszych (domyślnie),
- Od najpopularniejszych,
- Od najwyżżej ocenianych,
- Od najszybszych.

W panelu po lewej stronie widoku profilu w wersji desktopowej wyświetlane są informacje dotyczące konta użytkownika, takie jak:

- Awatar użytkownika
- Nazwa użytkownika,
- Liczba obserwujących,
- Liczba obserwowanych,
- Liczba dodanych przepisów,
- Data utworzenia konta.

Oprócz powyższych, w tej sekcji znajduje się również przycisk „Obserwuj”, po wciśnięciu którego dane konto zostanie zaobserwowane przez aktualnie zalogowanego użytkownika, natomiast w przypadku gościa, system przekieruje odwiedzającego do strony logowania. W panelu z informacjami o użytkowniku elementami interaktywnymi są też komponenty wyświetlające liczbę obserwujących i obserwowanych, po kliknięciu których system przekieruje użytkownika do widoku listującego odpowiednio konta obserwujące, bądź obserwowane przez danego użytkownika. Widok kont obserwujących użytkownika przedstawiono na Rysunek 62.



Rysunek 62 Widok kont obserwujących użytkownika. Źródło: Opracowanie własne.

6.3. Użytkownik zalogowany

W przeciwieństwie do gościa, użytkownik zalogowany ma możliwość dodawania i modyfikacji określonych zasobów w kulinarnej bazie serwisu. Pomyślne przejście procesu uwierzytelnienia odblokowuje odwiedzającemu dostęp do wielu społecznościowych funkcji platformy. Będąc zalogowanym może on przede wszystkim publikować, modyfikować i usuwać własne przepisy kulinarne. Oprócz tego zyskuje też dostęp do innych form interakcji ze społecznością serwisu, takich jak:

- Obserwowanie oraz zbieranie obserwacji od użytkowników,
- Zarządzanie swoim publicznym profilem,
- Ocenianie przepisów,
- Dodawanie komentarzy oraz odpowiadanie na nie,
- Zgłaszanie nieodpowiednich komentarzy moderacji.

Zalogowanie w systemie odblokowuje również możliwość korzystania z prywatnych funkcjonalności, przydatnych dla samego użytkownika jak:

- Niepubliczna lista zakupów,
- Dodawanie i przeglądanie ulubionych przepisów.

System umożliwia uwierzytelnionemu użytkownikowi także zmianę hasła do konta oraz bezpieczne wylogowanie.

6.3.1 Zarządzanie swoimi przepisami

Jednym z najważniejszych uprawnień użytkownika zalogowanego w omawianym serwisie jest możliwość aktywnego współtworzenia bazy kulinarnej portalu. W niniejszym podrozdziale szczegółowo przedstawiono zarządzanie przepisem od strony jego autora.

6.3.1.1 Dodawanie przepisów

Aby przejść do widoku dodawania przepisu należy wcisnąć przycisk „Dodaj” umieszczony w głównym panelu nawigacyjnym. Widok ten został podzielony na trzy obszary. Pierwszym z nich jest sekcja „Podstawowe informacje”. Znajdują się w niej następujące pola:

- Tytuł – długość do dwustu znaków,
- Krótki opis – długość do trzystu znaków,
- Kategorie – minimum jedna wybrana,
- Zdjęcia – maksymalnie dziesięć, do pięciu MB każde.

Rysunek 63 przedstawia sekcję „Podstawowe informacje” w widoku „Nowy przepis”.

Mei

Dieta
Kuchnie świata
Okazje
Posiłki
Ulubione przepisy

Nowy przepis

Podstawowe informacje

Tytuł
43 / 200 znaków

Krótki opis

Klasyczna, niezwykle prosta włoska kolacja, która idealnie sprawdzi się również jako lekka przekąska podczas wieczornego aperitivo. Sekret tego dania tkwi w prostocie – dobra oliwa i dojrzewające w słońcu pomidory przywołują na myśl ciepłe wieczory spędzane w miasteczkach nad jeziorem Garda.

292 / 300 znaków

Kategorie

Zdjęcia
2 / 10 zdjęć, maks. 5 MB

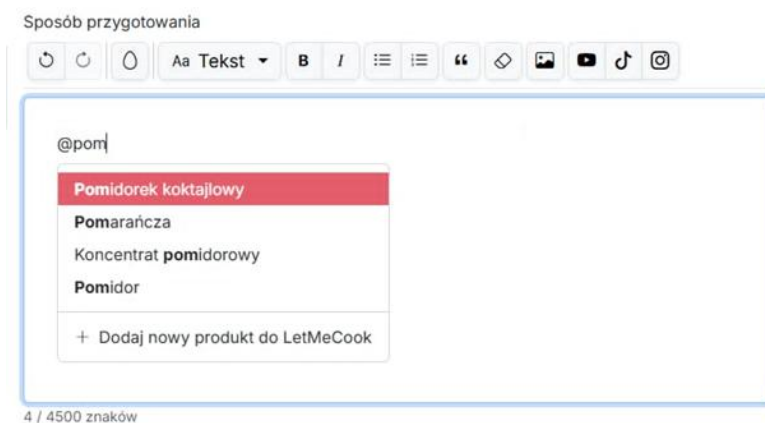
Rysunek 63 Sekcja „Podstawowe informacje” w widoku „Nowy przepis”. Źródło: Opracowanie własne.

Najbardziej złożonym elementem interfejsu w widoku tworzenia przepisu jest sekcja „Wykonanie”. Podzielono ją na dwie części. Pierwsza z nich zatytułowana „Składniki” zawiera produkty oraz ich ilości automatycznie pozyskiwane z drugiej części, czyli „Sposobu przygotowania”.

W „Sposobie przygotowania” znajduje się zaawansowany edytor tekstu. Bezpośrednio nad polem do wpisywania tekstu znajdują się przyciski umożliwiające wykonanie kolejno akcji:

- Cofania zmian,
- Ponawiania zmian,
- Wstawiania składnika,
- Ustawiania stylu tekstu,
- Przełączania pogrubienia tekstu,
- Przełączania kursywy,
- Wstawiania listy nieuporządkowanej,
- Wstawiania listy numerowanej,
- Wstawiania cytatu,
- Czyszczenia formatowania tekstu,
- Dodania zdjęcia,
- Dodania wideo z serwisu YouTube,
- Dodania wideo z serwisu TikTok,
- Dodania wideo z serwisu Instagram.

Edytor posiada unikalną funkcjonalność dodawania składników. Po wciśnięciu przycisku z ikoną jajka lub wpisaniu znaku „@” wyświetlana jest podręczna wyszukiwarka dynamicznie podpowiadająca produkty z bazy danych, która jest widoczna na Rysunek 64. Jeśli poszukiwanego składnika brakuje w bazie systemu, użytkownik może skorzystać z opcji „Dodaj nowy produkt do LetMeCook” znajdującej się pod wynikami wyszukiwania. Po jej wybraniu wyświetlane jest okno dialogowe „Zaproponuj nowy produkt” przedstawione na Rysunek 65.



Rysunek 64 Wyszukiwanie produktów w sekcji „Sposób przygotowania” w widoku „Nowy przepis”. Źródło:

Opracowanie własne.

Zaproponuj nowy produkt

? Nie możesz znaleźć składnika potrzebnego do Twojego przepisu w naszej bazie produktów? Możesz go zaproponować i dodać do przepisu za pomocą formularza poniżej.

Nazwa produktu:

Ciabatta

Klikając "Zaproponuj", akceptujesz zasady dodawania produktów:

! Nazwy produktów muszą:

- **zaczynać się z wielkiej litery,**
- **występować w liczbie pojedynczej,**
- **być w formie zdrobniałej tylko, gdy jest to zwyczajowe.**

Dopuszcza się wielką literę w środku nazwy tylko w przypadku nazw własnych, ale staraj się ich unikać.

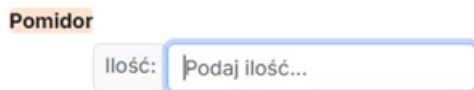
Przykłady: *Mąka pszenna, Przyprawa Maggi, Jajko, Pomidorek koktajlowy itp.*

Anuluj

Zaproponuj

Rysunek 65 Okno dialogowe „Zaproponuj nowy produkt”. Źródło: Opracowanie własne.

W oknie „Zaproponuj nowy produkt” należy podać nazwę proponowanego składnika w odpowiedniej formie, a następnie zatwierdzić klikając przycisk „Zaproponuj”. Zaproponowany składnik jest od razu potem gotowy do użycia w edytorze. Wybranie produktu z listy skutkuje jego dodaniem w sekcji ze składnikami oraz wyświetleniem pola służącego do wskazania jego ilości widocznego na Rysunek 66. Wprowadzona w tym polu wartość pojawia się w tekście w nawiasie obok składnika (z możliwością jej późniejszej edycji po kliknięciu) oraz dodana przy jego wystąpieniu w sekcji „Składniki”. Co istotne, system automatycznie sumuje w tej części ilości zadeklarowane przy różnych wystąpieniach tego samego produktu w tekście.



Rysunek 66 Pole ilości składnika w sekcji „Sposób przygotowania” w widoku „Nowy przepis”. Źródło: Opracowanie własne.

Za pomocą dedykowanych przycisków autor może osadzić w treści sposobu przygotowania zdjęcia, a także wideo z serwisów YouTube, TikTok, czy Instagram po podaniu linku do materiału.

Na Rysunek 67 przedstawiono sekcję „Wykonanie” w widoku dodawania przepisu.

Wykonanie

Składniki

Pomidor malinowy	2 sztuki
Oliwa z oliwek	1 łyżka
Bazylia	1 garść
Sól	szczypta
Pieprz czarny	szczypta
Ciabatta	1 sztuka
Czosnek	1 ząbek

Sposób przygotowania

Chrupiąca bruschetta



- Przygotowanie pomidorów**

Umyj i pokrój w drobną kostkę [2 sztuki] **Pomidor malinowy**. Przetóń je do miski, dodaj [1 łyżka] **Oliwa z oliwek**, posiekaną [1 garść] **Bazylię** oraz [szczypta] **Sól** i [szczypta] **Pieprz czarny**. Wymieszaj dokładnie i odstaw na 10 minut.
- Przygotowanie pieczywa**

Pokrój pieczywo [1 sztuka] **Ciabatta** na ukośne kromki o grubości około 1,5 cm. Ułóż je na blasze do pieczenia i skrop delikatnie **Oliwą z oliwek**. Piecz w piekarniku rozgrzanym do 200 stopni przez około 5 minut, aż będą złociste i chrupiące.
- Składanie i podanie**

Ciepłe grzanki od razu natrzyj z jednej strony przekrojonym [1 ząbek] **Czosnkiem**. Nałóż na każdą kromkę przygotowaną wcześniej mieszankę pomidorową wraz z powstałym sosem.



657 / 4500 znaków

Rysunek 67 Sekcja „Wykonanie” w widoku „Nowy przepis”. Źródło: Opracowanie własne.

Ostatnim etapem przed publikacją przepisu jest uzupełnienie sekcji „Dodatkowe informacje”. Umieszczono w niej następujące pola:

- Trudność przepisu (łatwy, normalny, trudny),
- Ilość porcji,
- Czas przygotowania,
- Liczba kalorii w 100g (opcjonalne).

Po uzupełnieniu wszystkich parametrów dodawanego przepisu użytkownik zatwierdza wprowadzone informacje poprzez kliknięcie przycisku „Dodaj przepis” umieszczonego na samym dole formularza. Sekcję „Dodatkowe informacje” oraz przycisk zatwierdzający tworzenie przepisu przedstawiono na Rysunek 68.

Dodatkowe informacje

Trudność przepisu: łatwy

Ilość porcji: 2 porcje 1 10

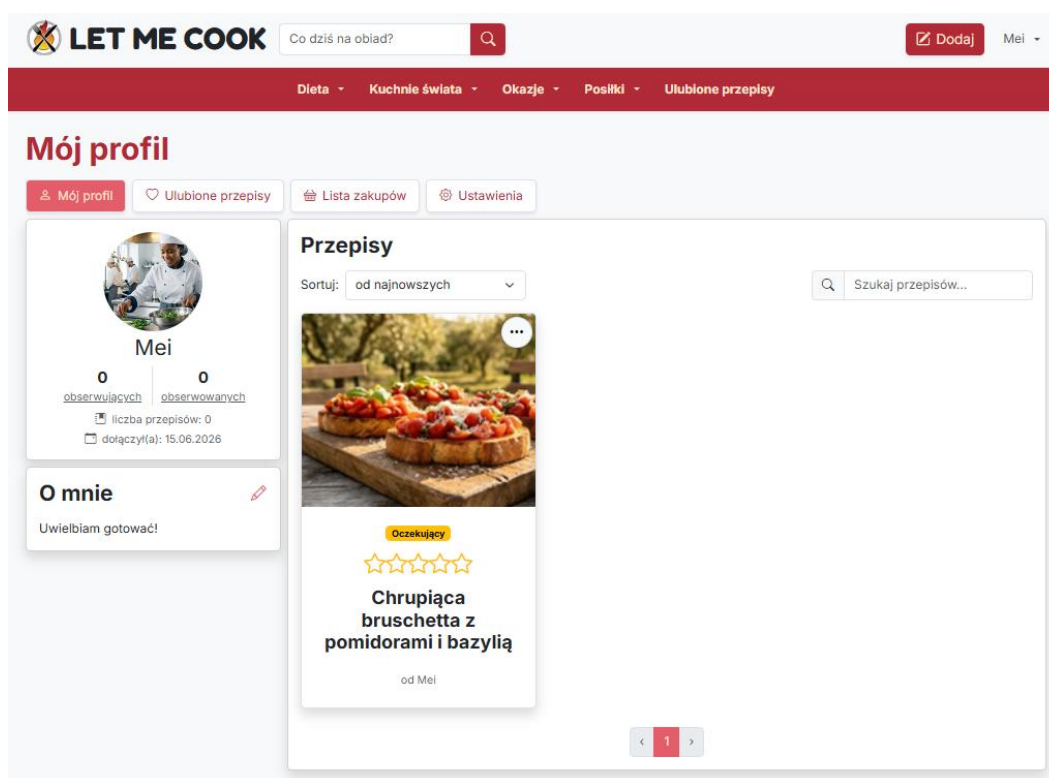
Czas przygotowania: 0 godzin 15 minut

Liczba kalorii w 100g (opcjonalnie): 180 kcal

+ Dodaj przepis

Rysunek 68 Sekcja „Dodatkowe informacje” w widoku „Nowy przepis”. Źródło: Opracowanie własne.

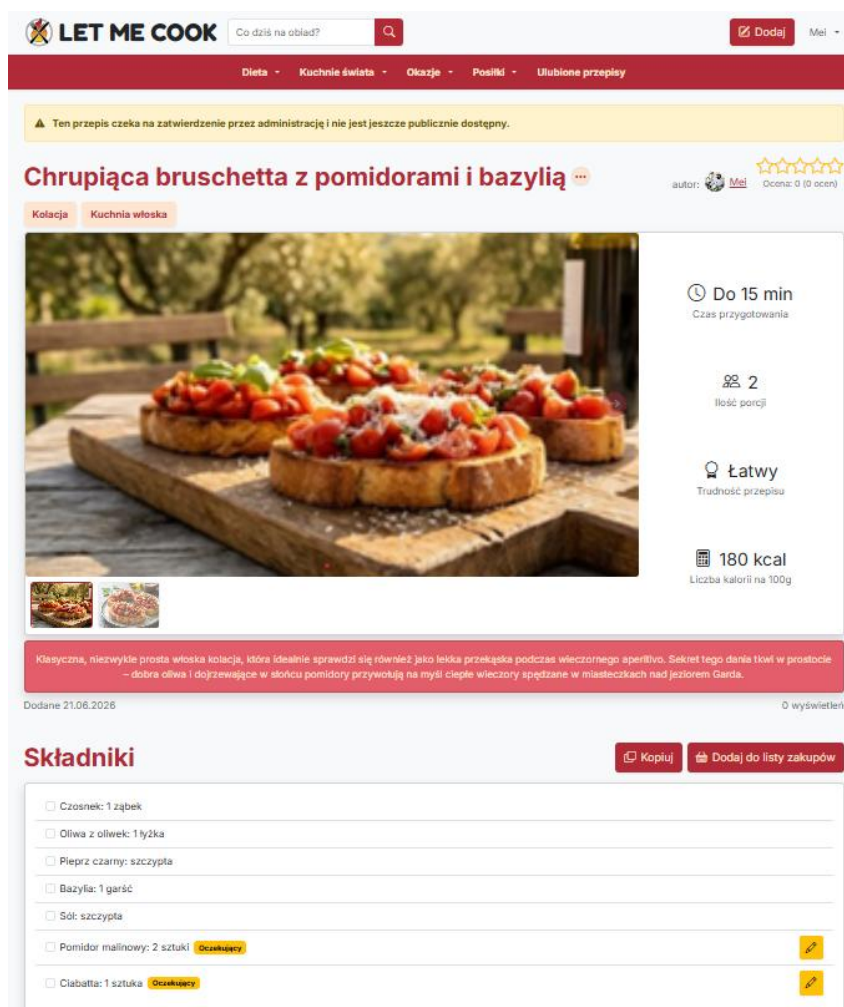
Przepis dodany przez użytkownika nieposiadającego roli moderatora ani administratora oczekuje na zatwierdzenie przez moderację. Dopiero po jego sprawdzeniu i zaakceptowaniu przez uprawnionego użytkownika staje się on widoczny dla wszystkich. Niezatwierdzony przepis oznaczony jest dla autora jako oczekujący w widoku jego profilu, co pokazano na Rysunek 69.



Rysunek 69 Oczekujący na zatwierdzenie przepis w widoku „Mój profil”. Źródło: Opracowanie własne.

Twórca może przejść do widoku szczegółów oczekującego na zatwierdzenie przepisu. Po przejściu do tego ekranu, w jego górnej części widnieje informacja, że przepis czeka na zatwierdzenie przez administrację i nie jest jeszcze publicznie dostępny. Jeżeli przy tworzeniu przepisu użytkownik skorzystał z produktów nieistniejących w bazie serwisu, również są one oznaczone w sekcji „Składniki”

jako oczekujące i możliwa jest jeszcze ich edycja. Widok szczegółów oczekującego na zatwierdzenie przepisu przedstawiono na Rysunek 70.



Rysunek 70 Widok szczegółów oczekującego na zatwierdzenie przepisu. Źródło: Opracowanie własne.

6.3.1.2 Edycja przepisów

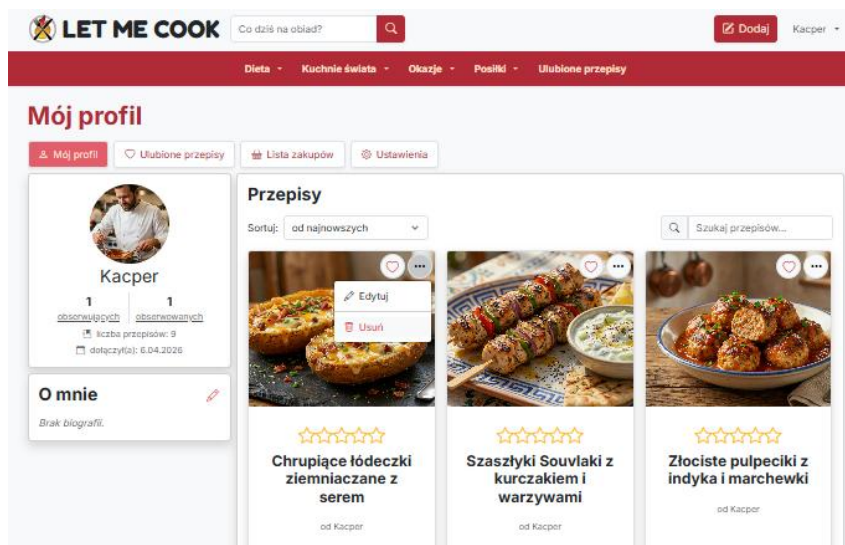
System pozwala twórcom na edycję udostępnionych wcześniej przepisów. Użytkownik może przejść do widoku edycji na dwa sposoby. Pierwszy z nich to kliknięcie ikony trzykropka w prawym górnym rogu miniatury danego przepisu w widoku profilu użytkownika, a następnie wybranie opcji „Edytuj”. Drugim sposobem jest wciśnięcie ikony trzech kropek przy tytule, a następnie opcji „Edytuj” z poziomu widoku szczegółów przepisu.

Interfejs edycji przepisu opiera się na ponownym wykorzystaniu formularza opisanego szczegółowo w podrozdziale Dodawanie przepisów. Główna różnica polega na zasileniu widoku dotychczasowymi informacjami o daniu. Elementem dodanym w ekranie edycji jest umieszczony na górze alert informujący o konieczności ponownego sprawdzenia przepisu przez moderację po zapisaniu zmian. Widok edycji istniejącego przepisu został przedstawiony na Rysunek 71.

Rysunek 71 Widok „Edycja przepisu”. Źródło: Opracowanie własne.

6.3.1.3 Usuwanie przepisów

W celu zapewniania użytkownikom pełnej kontroli nad zbiorem ich publikacji, system umożliwia usuwanie ich z publicznej bazy przepisów. Twórca może wywołać opcję „Usuń” w tych samych miejscach, co edycję przepisu – z poziomu widoku profilu, klikając ikonę trzykropka w prawym górnym rogu przepisu oraz z widoku szczegółów przepisu klikając analogiczną ikonę przy tytule przepisu. Wybranie tej opcji powoduje bezpieczne wycofanie wpisu z publicznej bazy poprzez zmianę jego statusu, co chroni strukturę bazy przed utratą spójności relacyjnej. Możliwość późniejszego przywrócenia przepisu przysługuje wyłącznie moderacji. Na Rysunek 72 została przedstawiona opcja usuwania przepisu z poziomu widoku profilu użytkownika, natomiast na Rysunek 73 z poziomu widoków szczegółów przepisu.

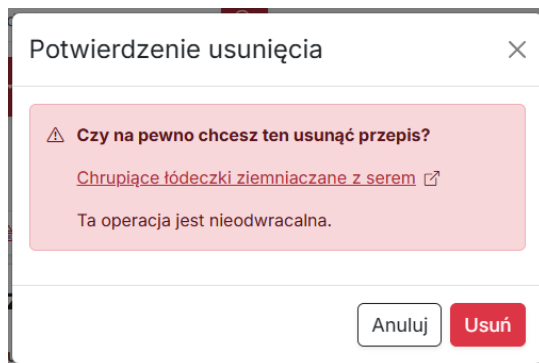


Rysunek 72 Opcja usuwania przepisu z widoku „Mój profil”. Źródło: Opracowanie własne.

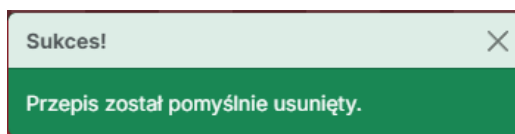


Rysunek 73 Opcja usuwania przepisu z widoku szczegółów przepisu. Źródło: Opracowanie własne.

Po wybraniu opcji „Usuń”, użytkownikowi wyświetlane jest menu dialogowe „Potwierdzenie usunięcia” wymagające potwierdzenia wykonania akcji usunięcia przepisu, które przedstawiono na Rysunek 74. Pomyślne usunięcie przepisu komunikowane jest wyświetleniem odpowiedniego powiadomienia przedstawionego na Rysunek 75.



Rysunek 74 Okno dialogowe „Potwierdzenie usunięcia”. Źródło: Opracowanie własne.

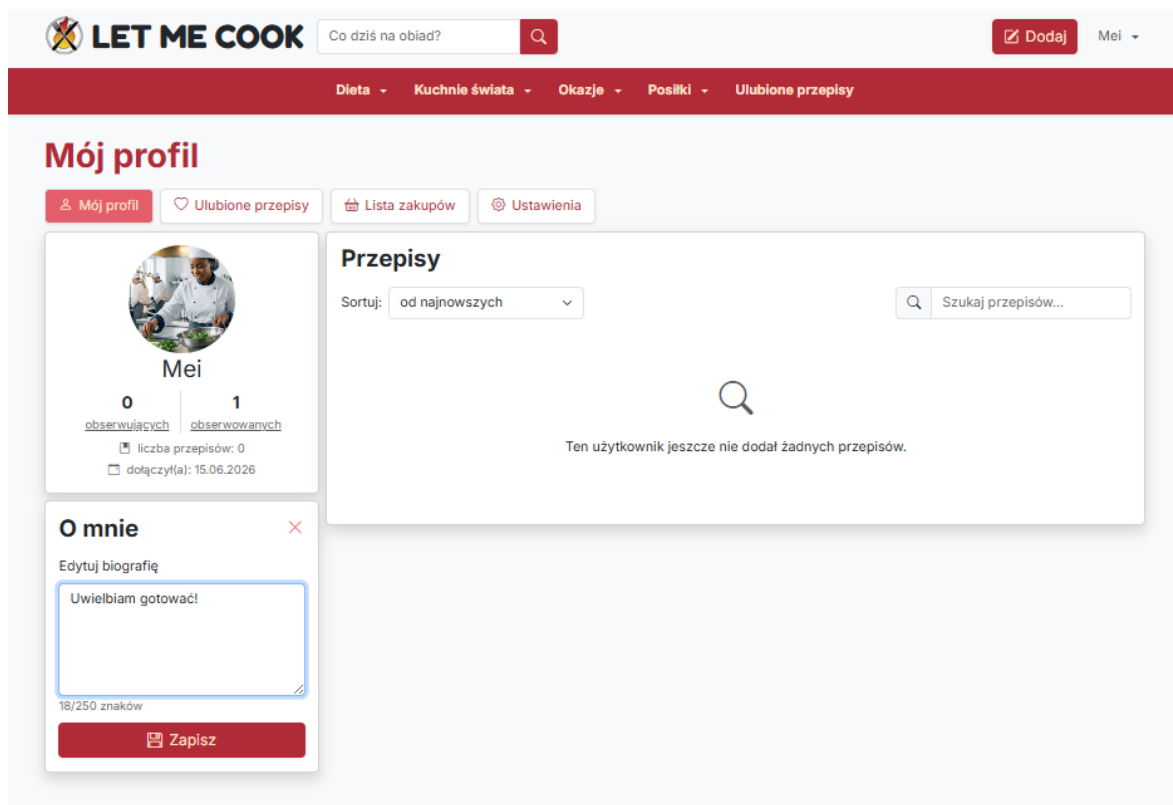


Rysunek 75 Powiadomienie o pomyślnym usunięciu przepisu. Źródło: Opracowanie własne.

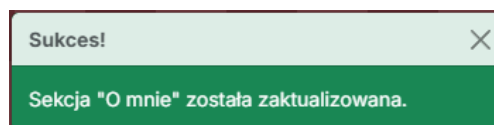
6.3.2 Zarządzanie profilem

Użytkownik zalogowany ma możliwość edycji danych prezentowanych na jego profilu. Personalizacja obejmuje modyfikację zawartości sekcji „O mnie” oraz zdjęcia profilowego.

Edycję biografii inicjuje się poprzez kliknięcie ikony ołówka w sekcji „O mnie”. Po wykonaniu tej akcji pojawia się pole umożliwiające modyfikację treści sekcji oraz przycisk „Zapisz” bezpośrednio pod nim. Użytkownik może anulować edycję informacji o sobie, wybierając przycisk „X” w prawym górnym rogu sekcji. Rysunek 76 przedstawia widok profilu użytkownika z sekcją „O mnie” w trakcie modyfikacji, natomiast na Rysunek 77 widnieje powiadomienie o pomyślnym wykonaniu akcji zapisu biografii.



Rysunek 76 Widok „Mój profil” z sekcją „O mnie” w trakcie modyfikacji. Źródło: Opracowanie własne.

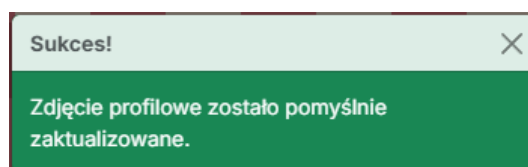


Rysunek 77 Powiadomienie o pomyślnej modyfikacji sekcji „O mnie” w widoku „Mój profil”. Źródło: Opracowanie własne.

Niemniej istotną funkcjonalnością w kontekście personalizacji profilu jest możliwość zmiany zdjęcia profilowego. Po najechaniu kursorem na obecny awatar, użytkownikowi wyświetli się ikona aparatu na półprzezroczystym tle. Jej kliknięcie otwiera okno dialogowe „Edytuj zdjęcie profilowe”. W tym miejscu użytkownik może wgrać nową grafikę za pomocą mechanizmu przeciągnij i upuść lub tradycyjnego wyboru pliku z dysku. Ma on także możliwość całkowitego usunięcia dotychczasowego awatara bez zmiany na inny korzystając z czerwonego przycisku „Usuń”. Użytkownik może zrezygnować z aktualizacji swojego zdjęcia wybierając przycisk „Anuluj” lub ikonę „X” w prawym górnym rogu okna. Na Rysunek 78 przedstawiono okno dialogowe „Edytuj zdjęcie profilowe”, natomiast na Rysunek 79 powiadomienie wyskakujące po pomyślnej zmianie awatara.



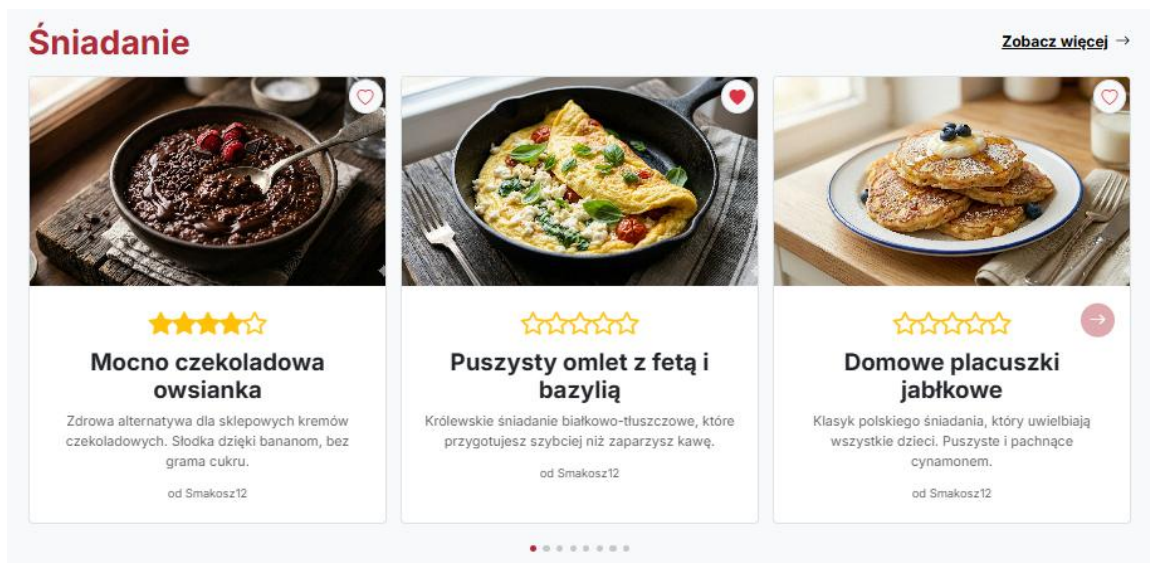
Rysunek 78 Okno dialogowe „Edytuj zdjęcie profilowe”. Źródło: Opracowanie własne.



Rysunek 79 Powiadomienie o pomyślnej zmianie zdjęcia profilowego w widoku „Mój profil”. Źródło: Opracowanie własne.

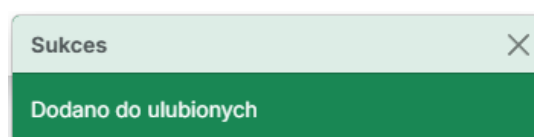
6.3.3 Zarządzanie ulubionymi przepisami

Zalogowany użytkownik ma możliwość zarządzania prywatną kolekcją ulubionych przepisów. W widokach zawierających siatki przepisów z podstawowymi informacjami, jak strona główna, wyniki wyszukiwania, czy profil twórcy użytkownik ma możliwość dodania potrawy do ulubionych poprzez kliknięcie w ikonę serduszka umieszczoną w prawym górnym rogu na miniaturowym zdjęciu dania. W widoku szczegółów przepisu dedykowany przycisk z sercem umieszczono w górnej części ekranu, tuż pod średnią ocen. Przyciski te działają na zasadzie przełącznika – ponowne kliknięcie aktywnej ikony w dowolnym z wymienionych miejsc skutkuje usunięciem go z ulubionych. Rysunek 80 przedstawia jedną z karuzel na stronie głównej, w której środkowy przepis został dodany do ulubionych.

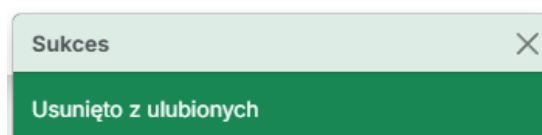


Rysunek 80 Karuzela z przepisem dodanym do ulubionych. Źródło: Opracowanie własne.

Po wykonaniu akcji dodania przepisu do ulubionych, w prawym dolnym ekranu wyświetlane jest powiadomienie o sukcesie przedstawione na Rysunek 81. Na Rysunek 82 przedstawiono natomiast powiadomienie pokazujące się po udanym usunięciu przepisu z ulubionych.



Rysunek 81 Powiadomienie o pomyślnym dodaniu przepisu do ulubionych. Źródło: Opracowanie własne.

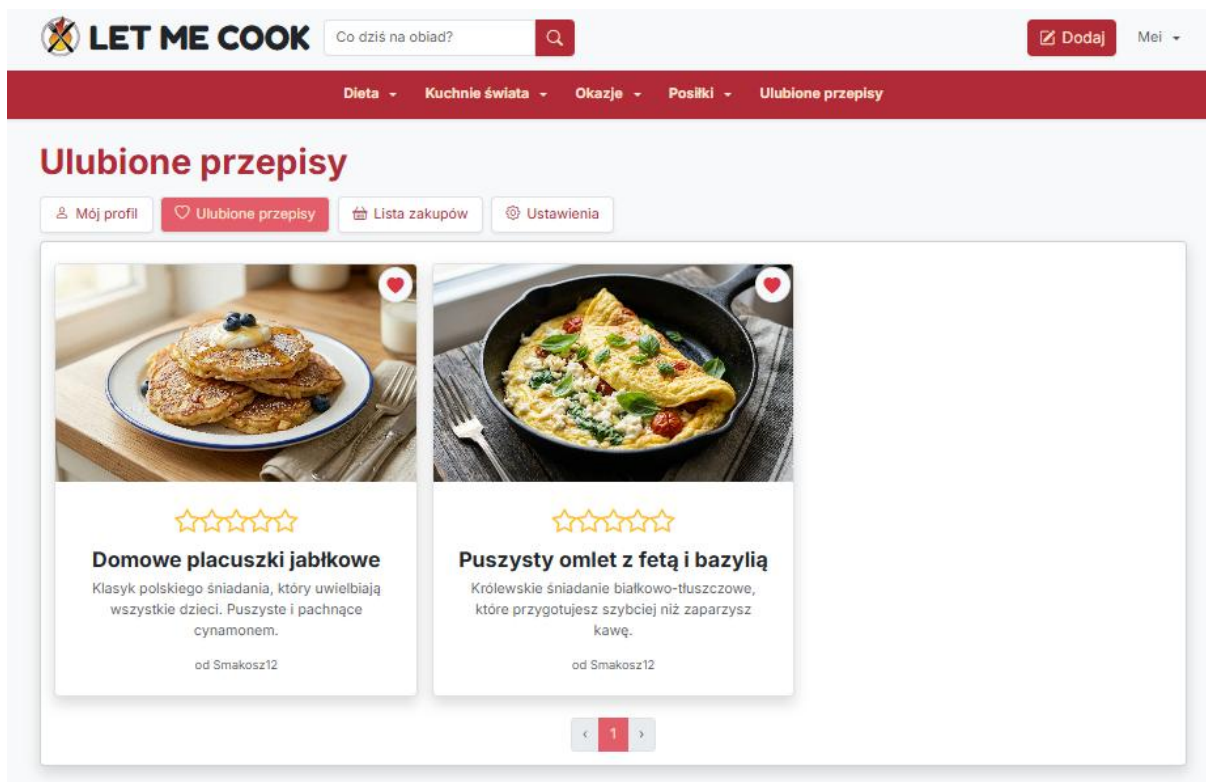


Rysunek 82 Powiadomienie o pomyślnym usunięciu przepisu z ulubionych. Źródło: Opracowanie własne.

Przepisy oznaczone przez użytkownika sercem wyświetlane są w dedykowanym widoku „Ulubione przepisy”. Użytkownik może przejść do tego widoku na trzy sposoby:

- Po wybraniu opcji „Ulubione” z menu rozwijanego po kliknięciu na nazwę użytkownika w panelu nawigacyjnym,
- Po kliknięciu odsyłacza „Ulubione przepisy” na pasku z grupami kategorii,
- Po wybraniu zakładki „Ulubione przepisy” będąc w widoku swojego profilu, listy zakupów lub ustawień.

Widok z przepisami dodanymi przez użytkownika do ulubionych przedstawiono na Rysunek 83.



Rysunek 83 Widok „Ulubione przepisy”. Źródło: Opracowanie własne.

6.3.4 Zarządzanie listą zakupów

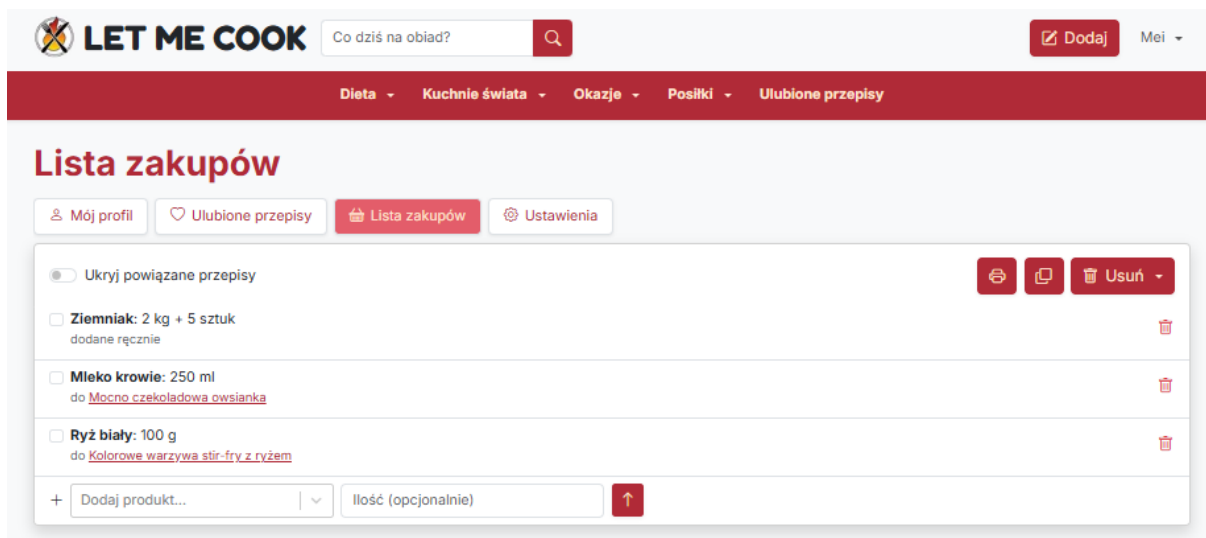
Lista zakupów jest narzędziem ułatwiającym użytkownikowi planowanie posiłków. Przejść do tego modułu można z poziomu menu dostępnego po kliknięciu nazwy użytkownika w głównym panelu nawigacyjnym i wybraniu opcji „Lista zakupów”. Aby wyświetlić widok listy zakupów można również wcisnąć zakładkę „Lista zakupów” będąc na jednej z prywatnych podstron użytkownika, takich jak „Mój profil”, „Ulubione przepisy” oraz „Ustawienia”.

W widoku listy zakupów można manualnie wprowadzać nowe pozycje. Służy do tego zlokalizowany na dole panel dodawania produktów. Składa się on z pola wyboru składników, pola określającego ilość oraz przycisku zatwierdzającego. Po zatwierdzeniu, produkt pojawia się w głównej części widoku jako nowa pozycja na liście.

Przy każdym produkcie na liście znajduje się interaktywny element wyboru, który pozwala na oznaczenie go jako zakupiony, a także przycisk z ikoną kosza do jego usunięcia. System wyposażono w mechanizm agregacji składników na liście. Jeśli algorytm rozpozna kompatybilne jednostki miary, automatycznie sumuje ilości tego samego produktu. W przypadku braku możliwości złączenia jednostek, aplikacja wyświetla ilości połączone znakiem plusa. Interfejs listy zakupów wyposażono również w następujące komponenty:

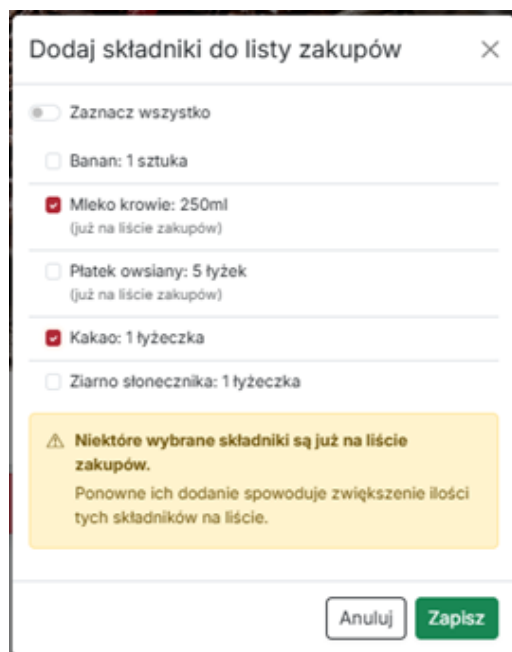
- Przełącznik „Ukryj powiązane przepisy” pozwalający na zwinięcie informacji o tym, skąd produkt został dodany do listy,
- Przycisk umożliwiający wydruk listy zakupów,
- Przycisk umożliwiający skopiowanie listy do schowka,
- Przycisk umożliwiający usunięcie oznaczonych lub wszystkich produktów z listy zakupów.

Widok listy zakupów przedstawiono na Rysunek 84.

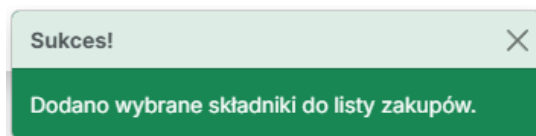


Rysunek 84 Widok „Lista zakupów”. Źródło: Opracowanie własne.

Dodawanie produktów do listy można nie tylko ręcznie, z poziomu jej widoku ale także z ekranu szczegółów przepisu. Można to zrobić klikając przycisk „Dodaj do listy zakupów” umieszczony w sekcji „Składniki”. Po jego naciśnięciu wyświetlane jest okno dialogowe „Dodaj składniki do listy zakupów”. Prezentuje ono wszystkie produkty zawarte w przepisie wraz z możliwością ich selektywnego zaznaczenia. Aby zapobiec nieświadomemu dodaniu składników znajdujących się już na liście, w oknie dodawania produktów wyświetlany jest alert ostrzegający, że niektóre z nich znajdują się już na liście, a ich dodanie spowoduje zwiększenie ilości ich istniejących wystąpień. Rysunek 85 przedstawia okno dialogowe „Dodaj składniki do listy zakupów” otwierane z widoku szczegółów przepisu. Po pomyślnym dodaniu produktów wyświetlane jest powiadomienie potwierdzające wykonanie akcji pokazane na Rysunek 86.



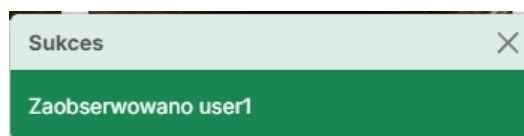
Rysunek 85 Okno dialogowe „Dodaj składniki do listy zakupów”. Źródło: Opracowanie własne.



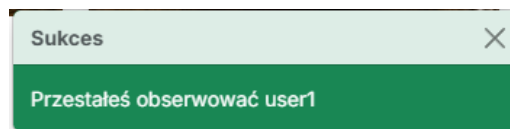
Rysunek 86 Powiadomienie o pomyślnym dodaniu składników do listy zakupów. Źródło: Opracowanie własne.

6.3.5 Obserwowanie użytkowników

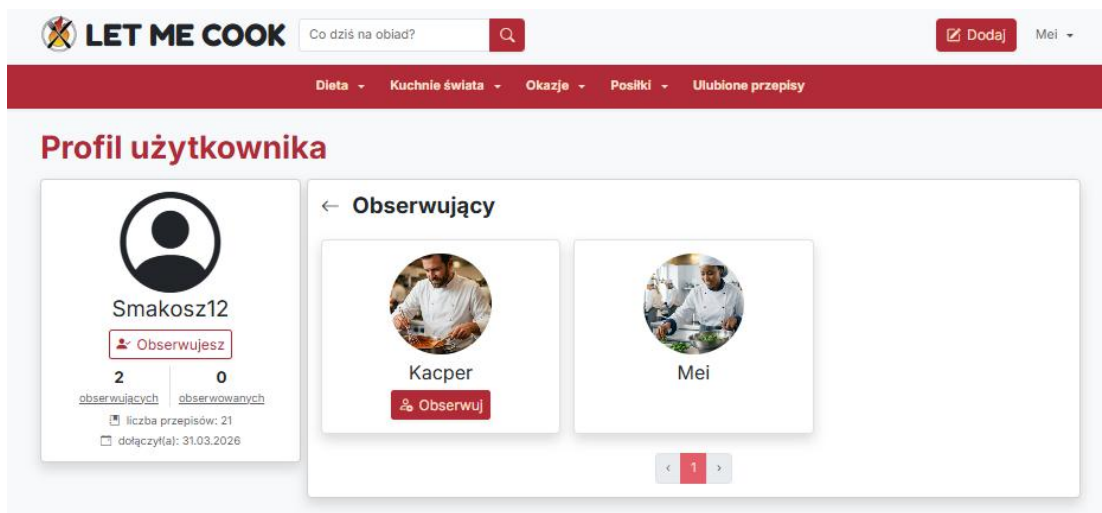
W celu ułatwienia śledzenia aktywności ulubionych autorów, w systemie wdrożono mechanizm obserwacji użytkowników. Dane konto można zaobserwować klikając przycisk „Obserwuj” znajdujący się w widoku jego profilu. Przycisk ten jest także dostępny pod nazwami kont w widokach z listą obserwowanych i obserwujących danego użytkownika. Po wciśnięciu zmienia się jego wygląd oraz tekst w nim zawarty z „Obserwuj” na „Obserwujesz”, a także wyskakuje powiadomienie z informacją o pomyślnym wykonaniu akcji przedstawione na x. Po ponownym jego kliknięciu obserwacja zostaje usunięta, a na ekranie pojawia się powiadomienie o skutecznym usunięciu obserwacji danego użytkownika widoczne na y. Rysunek 89 Widok kont obserwujących użytkownika w przypadku użytkownika zalogowanego. Źródło: Opracowanie własne. Rysunek 89 przedstawia widok z listą kont obserwujących użytkownika, na którym widoczne są przyciski „Obserwujesz” i „Obserwuj”.



Rysunek 87 Powiadomienie o pomyślnym zaobserwowaniu użytkownika. Źródło: Opracowanie własne.



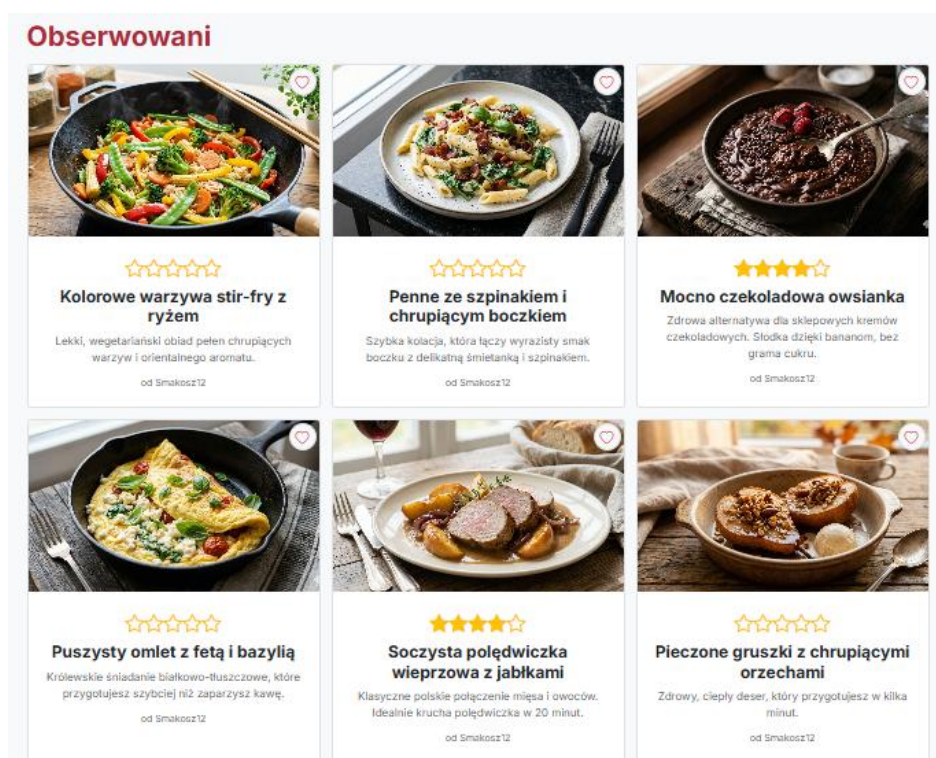
Rysunek 88 Powiadomienie o pomyślnym usunięciu obserwacji użytkownika. Źródło: Opracowanie własne.



Rysunek 89 Widok kont obserwujących użytkownika w przypadku użytkownika zalogowanego. Źródło:

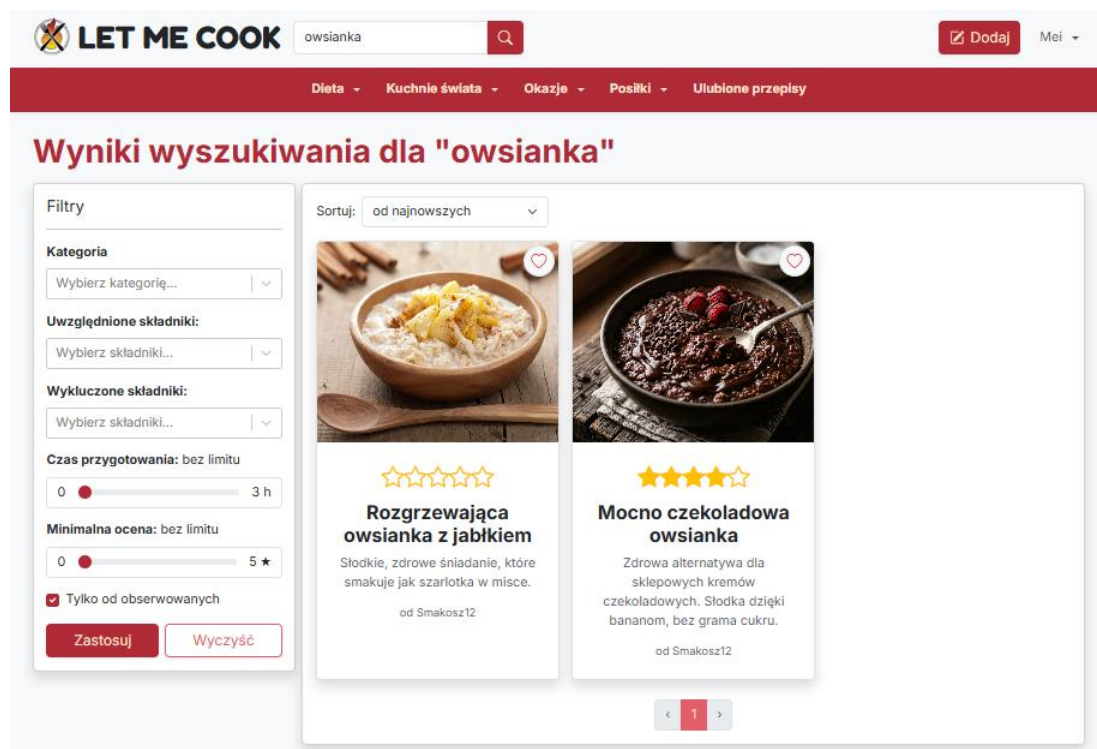
Opracowanie własne.

Główną korzyścią z obserwowania innych twórców jest częściowa personalizacja wyświetlanych treści. Przepisy opublikowane przez konta obserwowane przez użytkownika wylistowane są w kolejności od najnowszych w sekcji „Obserwowani” w dolnej części strony głównej, przedstawionej na Rysunek 90.



Rysunek 90 Sekcja „Obserwowani” na stronie głównej. Źródło: Opracowanie własne.

Mechanizm obserwowania został również uwzględniony w wyszukiwarce, gdzie wyniki wyszukiwania można zawęzić tylko do potraw dodanych przez obserwowanych użytkowników przy użyciu filtra „Tylko od obserwowanych”. Widok wyników wyszukiwania po zastosowaniu tego filtra przedstawiono na Rysunek 91.



Rysunek 91 Widok wyników wyszukiwania po zastosowaniu filtra „Tylko od obserwowanych”. Źródło:

Opracowanie własne.

6.3.6 Ocenianie przepisów

Zalogowani użytkownicy mają możliwość wystawiania przepisom ocen w skali od jednego do pięciu. Ocenienie przepisu możliwe jest wyłącznie z widoku szczegółów konkretnego dania. Proces oceny potrawy inicjuje się poprzez wciśnięcie przycisku z symbolem gwiazdki, znajdującego się pod aktualną średnią oceną przepisu. Kliknięcie przycisku wywołuje dymek z pięcioma gwiazdkami do wyboru. Wybór ilości gwiazdek powoduje natychmiastowe przeliczenie i odświeżenie średniej oceny przepisu, bez konieczności przeładowywania strony. Rysunek 92 przedstawia widok szczegółów przepisu z rozwiniętym dymkiem oceny.



Rysunek 92 Widok szczegółów przepisu z rozwiniętym dymkiem oceny. Źródło: Opracowanie własne.

Aplikacja w czytelny sposób sygnalizuje, czy przepis został już oceniony przez użytkownika. Ikonka gwiazdki na przycisku dodawania oceny staje się w pełni zamalowana po zrecenzowaniu potrawy. Naciśnięcie przycisku z zapełnioną gwiazdką wyświetla ten sam dymek, który jest pokazywany w przypadku braku wcześniejszej oceny, jednak w tym wypadku w otwartym komponencie widnieje wystawiona wcześniej ilość gwiazdek. Użytkownik ma możliwość odznaczenia zapisanej wartości, co skutkuje usunięciem wystawionej oceny i ponownym przeliczeniem średniej dla tego dania. Na Rysunek 93 przedstawiono widok szczegółów przepisu z rozwiniętym dymkiem oceny w przypadku ocenienia przepisu przez użytkownika.



Rysunek 93 Widok szczegółów przepisu z rozwiniętym dymkiem oceny w przypadku ocenienia przepisu przez użytkownika. Źródło: Opracowanie własne.

6.3.7 Dodawanie i usuwanie komentarzy

Dodawanie komentarzy możliwe jest wyłącznie z widoku szczegółów konkretnego przepisu. Aby utworzyć nowy wpis trzeba wprowadzić jego treść w dużym polu tekstowym na samej górze sekcji dyskusyjnej, a następnie zatwierdzić akcję przyciskiem „Dodaj komentarz”. Aplikacja uniemożliwia dodanie komentarza bez podania jego treści. System pozwala na dołączenie do pięciu zdjęć do jednej publikacji. Dodawać fotografie można poprzez kliknięcie przycisku z ikonką aparatu, znajdującego się pod polem z treścią komentarza. Po wciśnięciu tego przycisku wyświetlony zostanie systemowy eksplorator plików, w którym użytkownik może wybrać odpowiednie grafiki. Tworzenie nowego komentarza ze zdjęciem przedstawiono na Rysunek 94

Rysunek 94 Formularz tworzenia komentarza. Źródło: Opracowanie własne.

System umożliwia również prowadzenie dyskusji pomiędzy użytkownikami poprzez odpowiadanie na komentarze. Aby odpowiedzieć na dany wpis należy kliknąć znajdujący się pod nim odnośnik „Odpowiedz”. Po wykonaniu takiej akcji wyświetlany jest formularz służący do przesłania odpowiedzi, analogiczny do formularza tworzenia komentarza pierwszego stopnia. Aby zatwierdzić treść odpowiedzi należy wcisnąć przycisk „Wyślij” znajdujący się niżej. Jeżeli użytkownik zrezygnuje z pisanja odpowiedzi, może z powrotem zwinąć formularz jej tworzenia klikając przycisk „Anuluj” umieszczony obok przycisku do jej przesłania. Rysunek 95 przedstawia formularz tworzenia odpowiedzi do komentarza.

The screenshot shows a web interface titled "Komentarze (1)". At the top is a text input field labeled "Napisz komentarz...". Below it is a camera icon and a red button labeled "Dodaj komentarz". A user profile picture is shown next to the text "Mej skomentował(a): 5 minut temu". Below this is a comment box containing the text "Wyszło znakomicie!" and an image of a dish. A red link "Odpowiedz" is visible. Below the link is another text input field labeled "Napisz odpowiedź..." containing the text "Nie potwierdzam, moje wyszło bardzo słabe...". Below this is another image of a dish with a red 'X' icon. At the bottom right are two buttons: "Anuluj" (with a red 'X' icon) and "Wyślij" (with a speech bubble icon).

Rysunek 95 Formularz tworzenia odpowiedzi do komentarza. Źródło: Opracowanie własne.

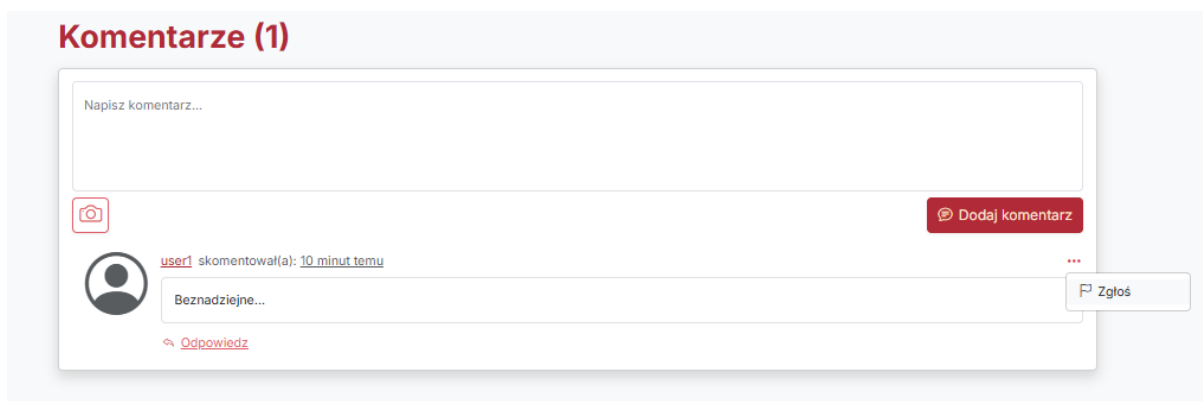
Użytkownik ma możliwość usunięcia swojego wpisu, klikając w znajdującą się przy nim ikonkę trzykropka, a następnie przycisk „Usuń”. Po wybraniu tej opcji wyświetlone zostanie okno dialogowe „Usuń komentarz” wymagające potwierdzenia wykonania akcji, lub jej anulowania, które zostało przedstawione na Rysunek 96.

The screenshot shows a dialog box titled "Usuń komentarz" with a close button (X) in the top right corner. The main text asks "Czy na pewno chcesz usunąć ten komentarz?". Below this, it shows the comment text: "Kacper: 'Nie potwierdzam, moje wyszło bardzo słabe...'". At the bottom right are two buttons: "Anuluj" (white) and "Usuń" (red).

Rysunek 96 Okno dialogowe „Usuń komentarz”. Źródło: Opracowanie własne.

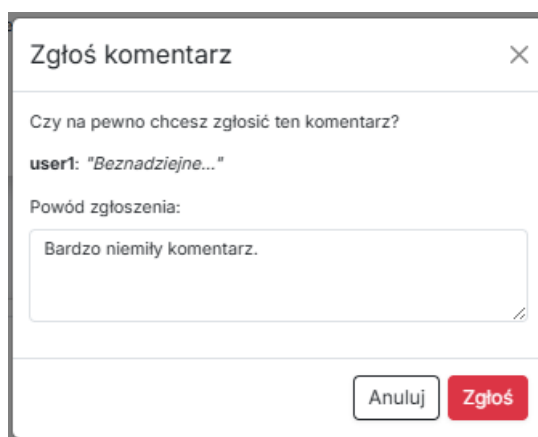
6.3.8 Zgłaszanie komentarzy

Aby utrzymać odpowiedni poziom dyskusji, system umożliwia zalogowanym użytkownikom zgłaszanie komentarzy naruszających zasady platformy. Proces ten inicjowany jest poprzez kliknięcie ikonki wielokropka przy wybranym wpisie, a następnie wybranie opcji „Zgłoś”, przedstawionej na Rysunek 97.



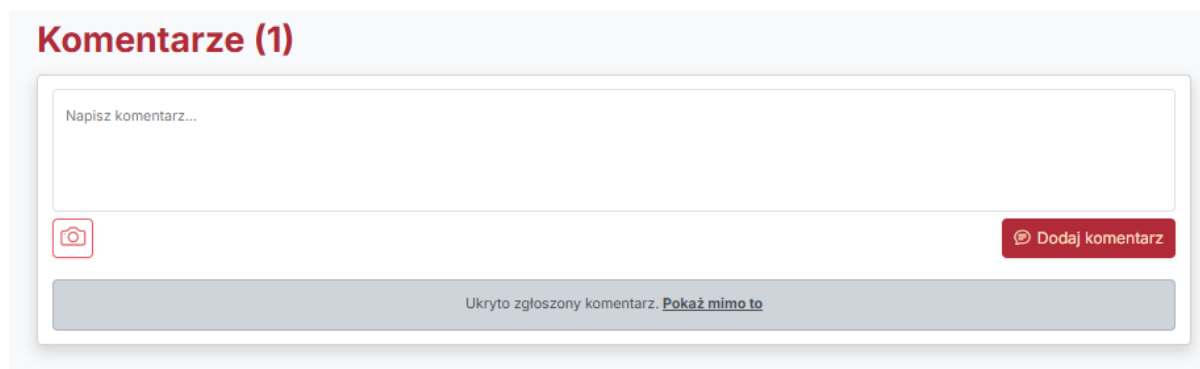
Rysunek 97 Zainicjowanie procesu zgłoszenia komentarza. Źródło: Opracowanie własne.

W kolejnym etapie użytkownik musi podać powód zgłoszenia w wyświetlonym oknie dialogowym „Zgłoś komentarz”. Po wpisaniu odpowiedniego uzasadnienia, zgłaszający musi zatwierdzić operację poprzez kliknięcie przycisku „Zgłoś” z czerwonym tłem, znajdującego się w dolnej części okna dialogowego. Jeżeli użytkownik zrezygnuje z tworzenia zgłoszenia, może zamknąć okno dialogowe klikając ikonkę „X” w jego prawym górnym rogu, bądź przycisk „Anuluj” na dole. Dalsze procesowanie zgłoszenia przez moderację zostało omówione w rozdziale 6.4.1.3. Rysunek 98 przedstawia okno dialogowe „Zgłoś komentarz” z uzupełnionym powodem zgłoszenia.



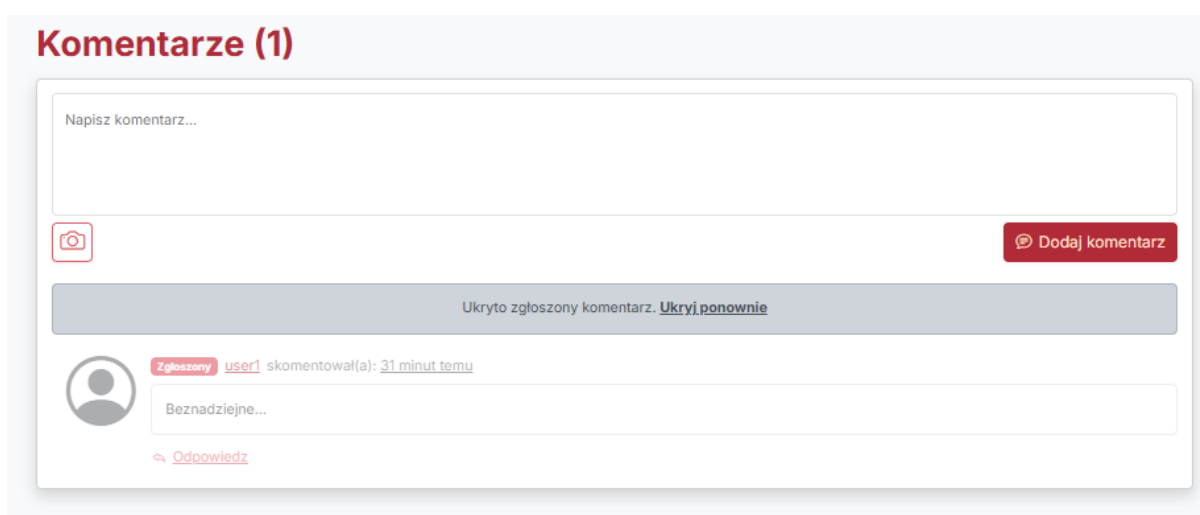
Rysunek 98 Okno dialogowe „Zgłoś komentarz”. Źródło: Opracowanie własne.

Skutkiem pomyślnego przesłania zgłoszenia, widocznym w interfejsie użytkownika, jest ukrycie zgłoszonego wpisu. W jego miejscu wyświetla się szary pasek z tekstem „Ukryto zgłoszony komentarz” oraz interaktywną opcją „Pokaż mimo to” przedstawiony na Rysunek 99.



Rysunek 99 Komentarz ukryty po zgłoszeniu. Źródło: Opracowanie własne.

Pomimo zgłoszenia, system pozwala na podgląd ukrytego komentarza na żądanie użytkownika. Wybranie opcji „Pokaż mimo to” odsłania zgłoszony wpis, który zostaje dodatkowo oznaczony tagiem „Zgłoszony”, widocznym tylko dla zgłaszającego. Odkryty komentarz można wówczas z powrotem schować poprzez kliknięcie elementu „Ukryj ponownie” na szarym pasku. Rysunek 100 przedstawia zgłoszony wpis, który został odsłonięty przez zalogowanego użytkownika.



Rysunek 100 Komentarz odkryty po zgłoszeniu. Źródło: Opracowanie własne.

6.3.9 Zmiana hasła

Każdy użytkownik posiadający konto w systemie ma możliwość zmiany swojego hasła dostępowego. Jest to możliwe z poziomu widoku ustawień. Do ekranu tego można przejść z menu rozwijanego po kliknięciu w nazwę użytkownika w głównym panelu nawigacyjnym. Alternatywnie do tego widoku można dostać się będąc w widoku profilu, ulubionych przepisów lub listy zakupów i wybierając zakładkę ustawień. Widok ustawień przedstawiono na Rysunek 101.

Rysunek 101 Widok „Ustawienia”. Źródło: Opracowanie własne.

Zmiana hasła realizowana jest poprzez formularz składający się z następujących pól:

- Aktualne hasło,
- Nowe hasło,
- Potwierdź nowe hasło.

System wymaga autoryzacji operacji poprzez podanie aktualnego hasła w pierwszym polu. W dwóch kolejnych polach konieczne jest wprowadzenie nowego hasła, co stanowi zabezpieczenie przed przypadkowymi błędami przy wpisywaniu. Po poprawnym wypełnieniu i zatwierdzeniu formularza, system aktualizuje poświadczenia użytkownika w bazie i nowe hasło natychmiast staje się aktywne.

6.3.10 Wylogowanie z systemu

Wylogowanie z serwisu możliwe jest wyłącznie z poziomu menu wyświetlanego po kliknięciu w nazwę zalogowanego użytkownika, w prawym górnym rogu panelu nawigacyjnego. Po wybraniu z listy opcji „Wyloguj się” natychmiast kończona jest aktywna sesja konta w systemie, a użytkownik przekierowywany jest każdorazowo do strony głównej ze statusem gościa.

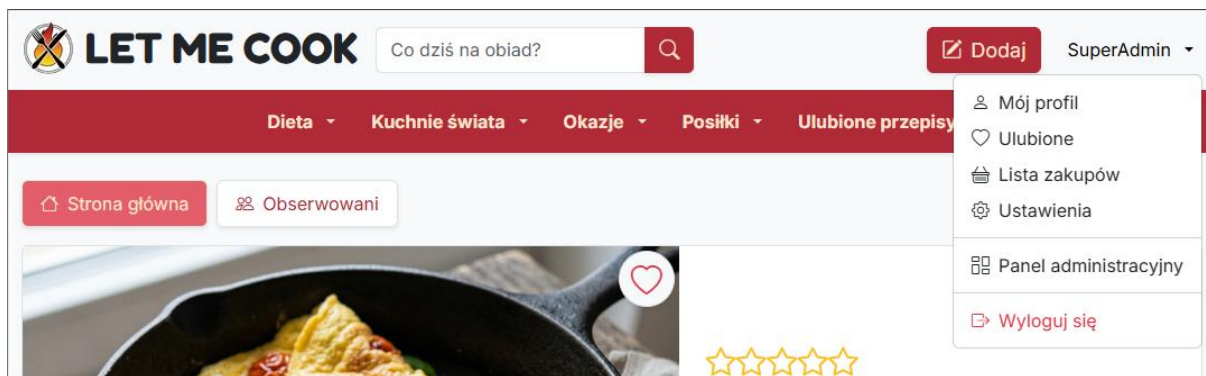
6.4. Administracja i moderacja

Ten podrozdział poświęcony jest dodatkowym widokom i funkcjonalnościom aplikacji dostępnym wyłącznie dla moderatorów i administratorów serwisu. Pozwalają one w prosty i efektywny sposób zarządzać zasobami serwisu, takimi jak przepisy, produkty, kategorie i ich grupy, a także zgłoszenia komentarzy i konta użytkowników.

6.4.1 Panel administracyjny

Panel administracyjny stanowi centralne miejsce zarządzania zasobami serwisu. Skupia ono zarówno narzędzia moderacyjne, jak i konfiguracyjne.

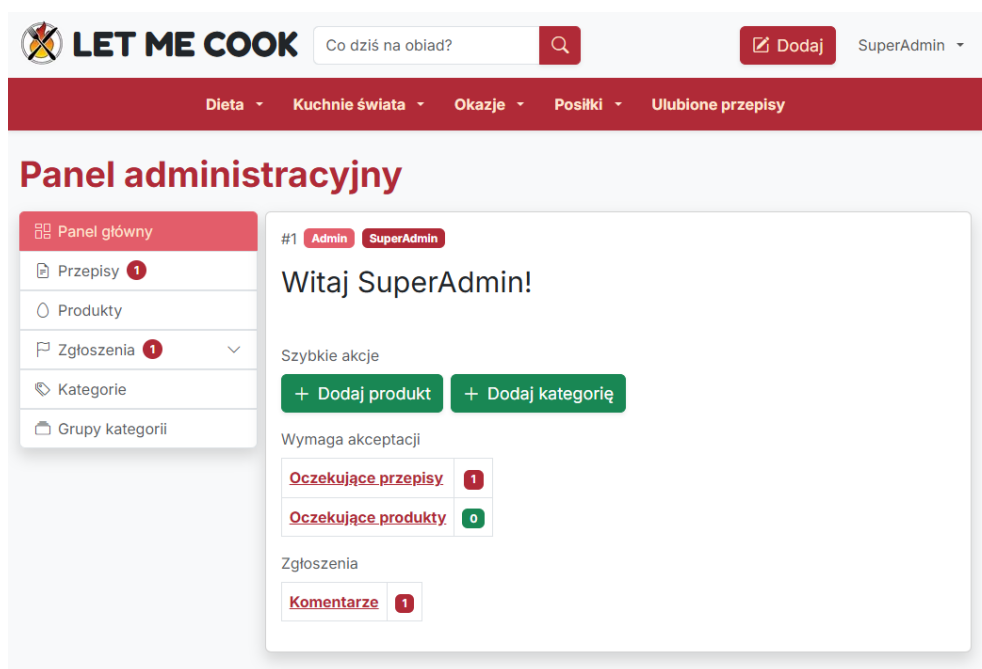
Dostęp do panelu administracyjnego jest przyznany wyłącznie dla ról Moderator, Administrator i SuperAdmin. Rysunek 61 prezentuje rozwinięte menu użytkownika w pasku nawigacyjnym z widocznym linkiem kierującym do panelu administracyjnego.



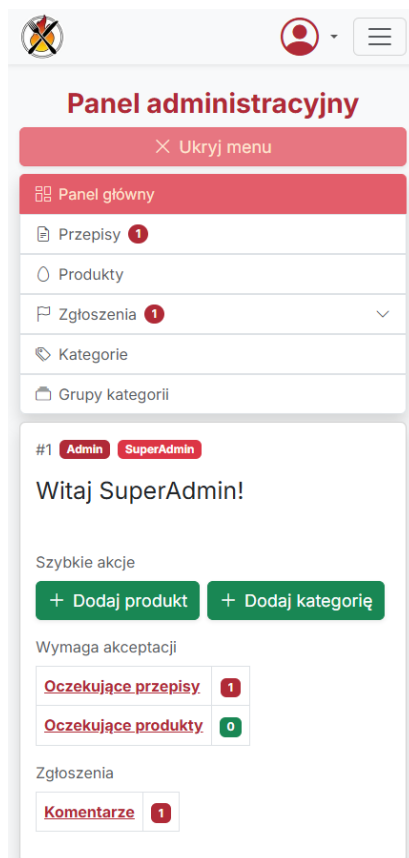
Rysunek 102 Menu użytkownika z widocznym linkiem do panelu administracyjnego. Źródło: Opracowanie własne

Po naciśnięciu linku ukazuje się panel administracyjny z domyślnie pokazanym ekranem głównym. Po lewej stronie (lub w górnej części ekranu w widoku mobilnym) widoczne jest menu z sekcjami: „Panel główny”, „Przepisy”, „Produkty”, „Zgłoszenia”, „Kategorie” i „Grupy kategorii”. Obok nazw sekcji „Przepisy”, „Produkty” i „Zgłoszenia” może pojawić się plakietka z liczbą obiektów oczekujących na weryfikację przez moderację. Pośrodku umieszczono identyfikator, przypisane role oraz przywitanie użytkownika, a pod nimi skróty do najprzydatniejszych akcji oraz liczniki obiektów do zweryfikowania.

Rysunek 103 prezentuje ekran główny panelu administracyjnego na ekranie komputera, a Rysunek 104 – na ekranie urządzenia mobilnego z widocznym rozwiniętym menu.



Rysunek 103 Ekran główny panelu administracyjnego - widok desktopowy. Źródło: Opracowanie własne



Rysunek 104 Ekran główny panelu administracyjnego - widok mobilny. Źródło: Opracowanie własne

6.4.1.1 Zarządzanie istniejącymi przepisami

Pierwszą sekcją panelu administracyjnego jest sekcja zarządzania przepisami. Podzielona jest ona według statusów przepisów na cztery zakładki:

- Oczekujące,
- Zaakceptowane,
- Odrzucone,
- Usunięte.

Każda zakładka zawiera wyszukiwarkę, elementy paginacji oraz tabelę z wylistowanymi przepisami o określonym statusie. Dla każdej pozycji widoczne są najważniejsze informacje, takie jak: identyfikator, tytuł z hiperłączem do strony przepisu, nazwa autora z hiperłączem do jego profilu, data utworzenia, status i dostępne akcje. Jeżeli przepis jest odrzucony, widoczny jest również powód odrzucenia po najechaniu kursorem na ikonkę obok statusu.

Rysunek 105, Rysunek 106, Rysunek 107 oraz Rysunek 108 przedstawiają listy przepisów w panelu administracyjnym podzielone według statusu.

Panel administracyjny

Panel główny

Przepisy 2

Produkty

Zgłoszenia 1

Kategorie

Grupy kategorii

Oczekujące Zaakceptowane Odrzucone Usunięte

Szukaj...

ID	Tytuł	Autor	Utworzono	Status	Akcje
1039	Kolorowe warzywa stir-fry z ryżem	user1	6.04.2026, 13:30:53	Oczekujący	✓ ✗ 🗑️
1012	Klasyczny burger wołowy z warzywami	Kacper	6.04.2026, 13:23:47	Oczekujący	✓ ✗ 🗑️

1

Strona 1 z 1

Pokaż: 10 Łącznie: 2

Rysunek 105 Lista oczekujących przepisów w panelu administracyjnym. Źródło: Opracowanie własne.

Panel administracyjny

Panel główny

Przepisy 2

Produkty

Zgłoszenia 1

Kategorie

Grupy kategorii

Oczekujące Zaakceptowane Odrzucone Usunięte

Szukaj...

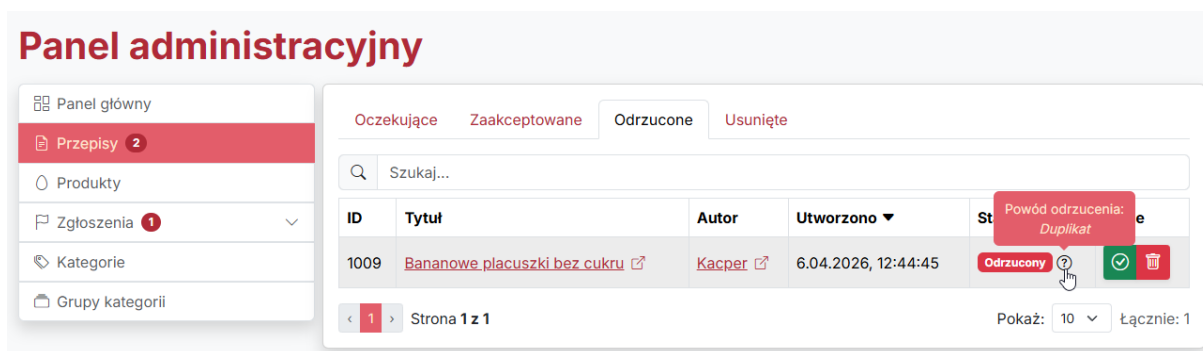
ID	Tytuł	Autor	Utworzono	Status	Akcje
1038	Penne ze szpinakiem i chrupiącym boczkiem	user1	6.04.2026, 13:30:50	Zatwierdzony	✗ 🗑️
1037	Mocno czekoladowa owsianka	user1	6.04.2026, 13:30:43	Zatwierdzony	✗ 🗑️
1036	Puszysty omlet z fetą i bazylią	user1	6.04.2026, 13:30:39	Zatwierdzony	✗ 🗑️
1035	Soczysta połówiczka wieprzowa z jabłkami	user1	6.04.2026, 13:30:36	Zatwierdzony	✗ 🗑️
1034	Pieczone gruszki z chrupiącymi orzechami	user1	6.04.2026, 13:29:53	Zatwierdzony	✗ 🗑️
1033	Szybka wołowina z brokułami	user1	6.04.2026, 13:29:50	Zatwierdzony	✗ 🗑️
1032	Zupa krem z marchewki i imbiru	user1	6.04.2026, 13:29:47	Zatwierdzony	✗ 🗑️
1031	Salatka z prażonymi pestkami i fetą	user1	6.04.2026, 13:29:42	Zatwierdzony	✗ 🗑️
1030	Kurczak duszony w białym winie	user1	6.04.2026, 13:29:38	Zatwierdzony	✗ 🗑️
1029	Złoty filet z dorsza	user1	6.04.2026, 13:28:51	Zatwierdzony	✗ 🗑️

1 2 3 4

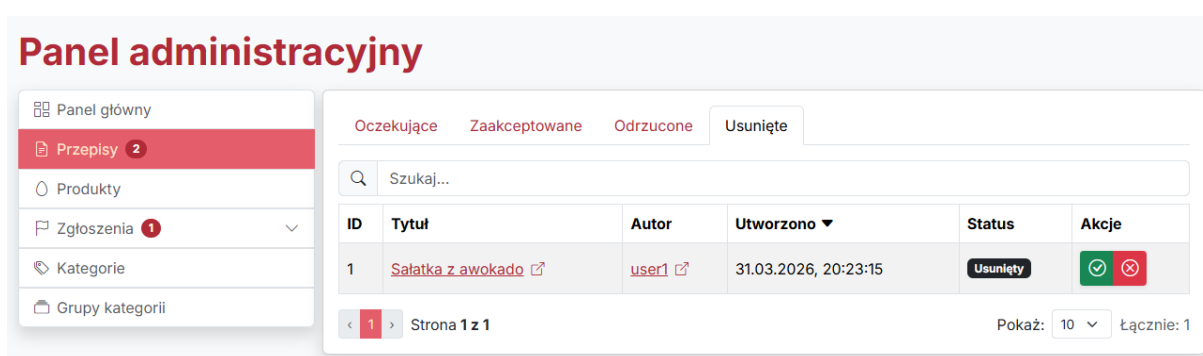
Strona 1 z 4

Pokaż: 10 Łącznie: 37

Rysunek 106 Lista zaakceptowanych przepisów w panelu administracyjnym. Źródło: Opracowanie własne.



Rysunek 107 Lista odrzuconych przepisów w panelu administracyjnym. Źródło: Opracowanie własne.



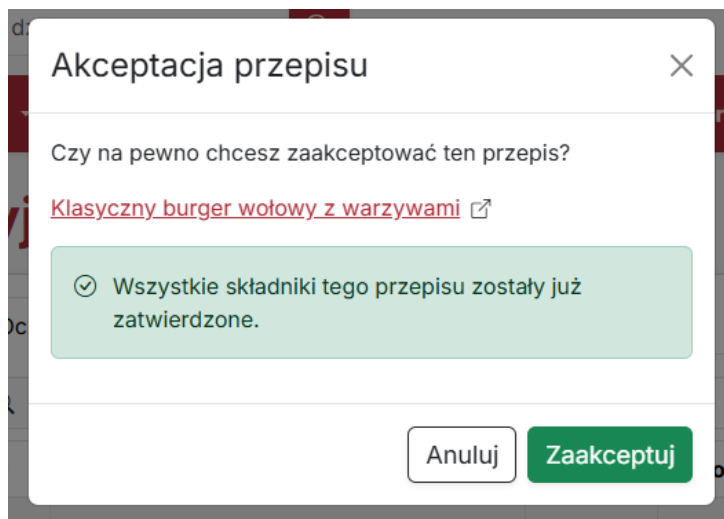
Rysunek 108 Lista usuniętych przepisów w panelu administracyjnym. Źródło: Opracowanie własne.

6.4.1.1.1. Zatwierdzanie przepisów

Przepisy, które nie są jeszcze zaakceptowane w serwisie, można zaakceptować klikając na zielony przycisk „Zaakceptuj” obok przepisu. Po naciśnięciu pojawia się okno dialogowe potwierdzające akceptację. W oknie umieszczone jest pytanie potwierdzające, link do przepisu, alert z informacją o stanie składników oraz przyciski „Anuluj” i „Zaakceptuj”.

W zależności od stanu produktów dołączonych do przepisu, wybierany jest jeden z trzech dostępnych wariantów okna dialogowego:

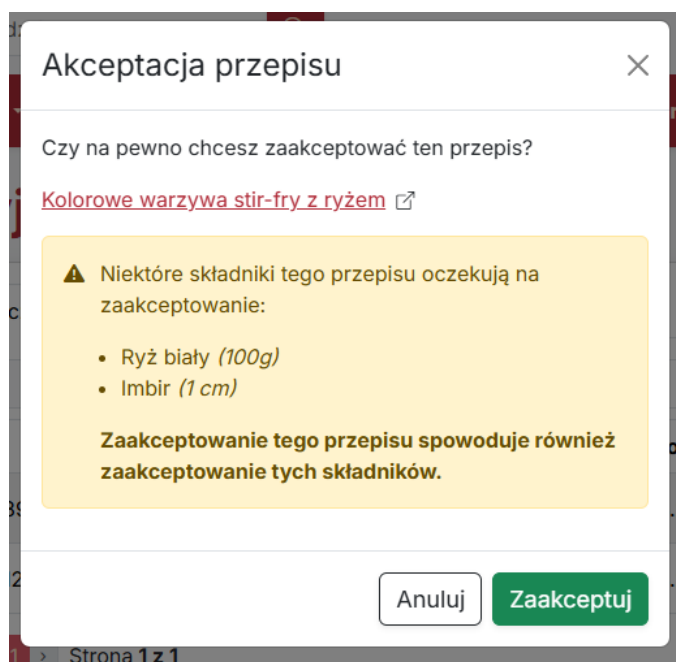
1. Z alertem potwierdzającym, że wszystkie składniki przepisu są zatwierdzone – akceptacja przepisu jedynie zmienia jego status (Rysunek 109),



Rysunek 109 Okno dialogowe zatwierdzania przepisu - wariant z wszystkimi zaakceptowanymi składnikami.

Źródło: Opracowanie własne.

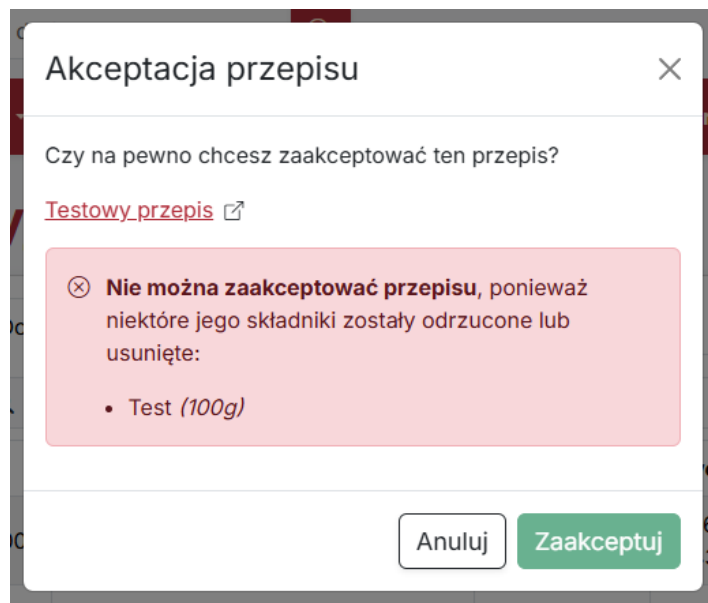
2. Z alertem informującym o oczekujących produktach, które są składnikami przepisu – akceptacja przepisu zatwierdza również te produkty (Rysunek 110),



Rysunek 110 Okno dialogowe zatwierdzania przepisu - wariant z oczekującymi składnikami.

Źródło: Opracowanie własne.

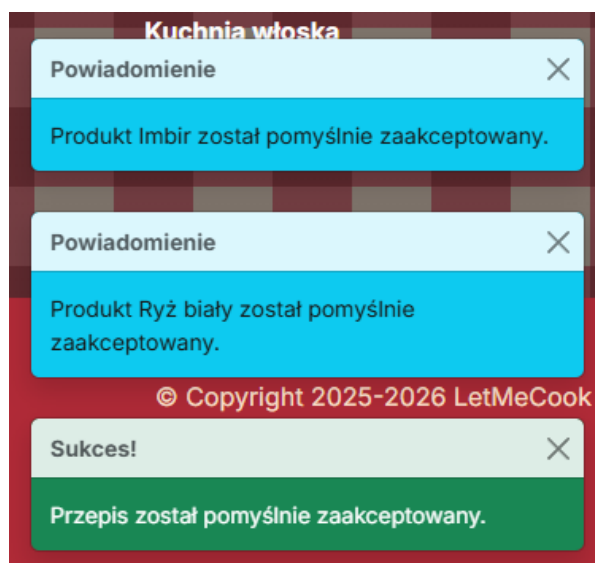
3. Z alertem sygnalizującym odrzucone lub usunięte produkty użyte w przepisie – akceptacja przepisu nie jest wówczas możliwa.



Rysunek 111 Okno dialogowe zatwierdzania przepisu - wariant z odrzuconym składnikiem.

Źródło: Opracowanie własne.

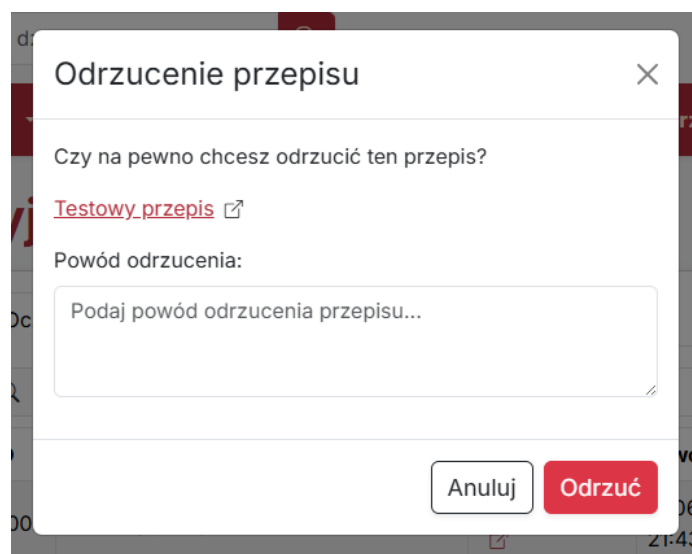
Po pomyślnym zaakceptowaniu przepisu strona się odświeża, a w prawym dolnym rogu pokazują się powiadomienia o akceptacji przepisu oraz – jeżeli przepis zawierał oczekujące produkty – o akceptacji produktów. Efekt ten został przedstawiony na Rysunek 112.



Rysunek 112 Powiadomienia o akceptacji przepisu i produktów w przepisie. Źródło: Opracowanie własne.

6.4.1.1.2. Odrzucanie przepisów

Każdy przepis, który nie posiada jeszcze statusu „Odrzucony”, może zostać odrzucony przez moderację poprzez naciśnięcie czerwonego przycisku „Odrzuć” z krzyżykiem. Po naciśnięciu pokazuje się okno dialogowe (Rysunek 113), do którego należy wpisać powód takiej decyzji. Ta informacja zwrotna jest widoczna zarówno dla moderacji, jak i dla autora przepisu.



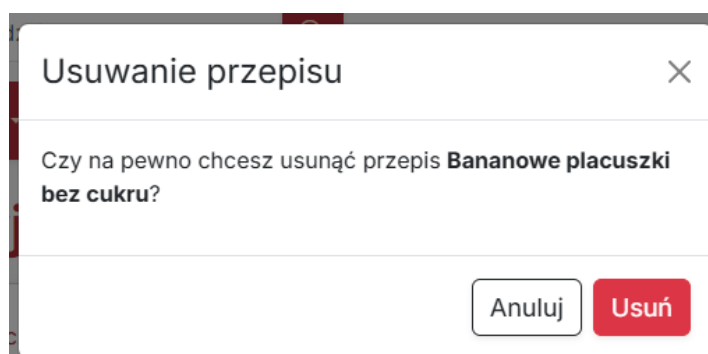
Rysunek 113 Okno dialogowe odrzucania przepisu. Źródło: Opracowanie własne.

Odrzucenie przepisu nie oznacza jego usunięcia – autor nadal będzie widział odrzuconą pozycję na swoim profilu wraz z powodem. Przepis może zostać zmodyfikowany przez autora, co zmieni jego status z powrotem na „Oczekujący”.

6.4.1.1.3. Usuwanie przepisów

Moderacja może również z poziomu panelu administracyjnego usunąć przepis z serwisu. Następuje wówczas usunięcie nietrwale (ang. *soft delete*) – przepis dalej istnieje w serwisie, ale staje się on niewidoczny dla wszystkich użytkowników nienależących do grona moderatorów lub administratorów.

Do usunięcia przepisu służy przycisk z ikoną kosza. Po naciśnięciu ukazuje się okno dialogowe potwierdzające chęć usunięcia przepisu, które zostało przedstawione na Rysunek 114.



Rysunek 114 Okno dialogowe usunięcia przepisu w panelu administracyjnym. Źródło: Opracowanie własne.

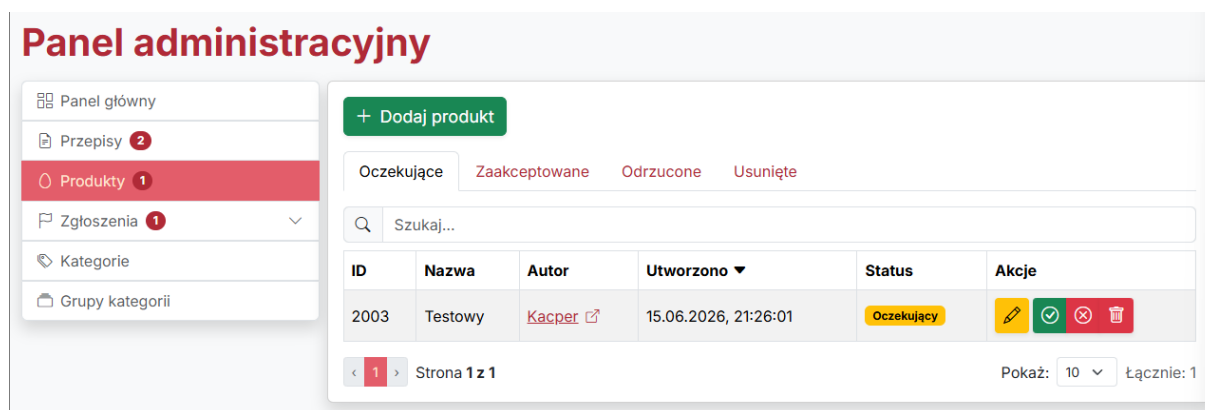
6.4.1.2 Zarządzanie produktami

Następną sekcją panelu administracyjnego jest sekcja zarządzania bazą produktów w serwisie. Z poziomu tego widoku można zarządzać istniejącymi produktami, a także dodawać nowe. Podobnie jak sekcja zarządzania przepisów została ona podzielona według statusów na cztery zakładki:

- Oczekujące,
- Zaakceptowane,
- Odrzucone,
- Usunięte.

Każda zakładka zawiera wyszukiwarkę, elementy paginacji oraz tabelę z wylistowanymi produktami o określonym statusie. Dla każdej pozycji widoczne są najważniejsze informacje, takie jak: identyfikator, nazwa produktu, nazwa autora z hiperłączem do jego profilu, data utworzenia, status i dostępne akcje. Jeżeli produkt jest odrzucony, widoczny jest również powód odrzucenia po najechaniu kursorem na ikonkę obok statusu.

Rysunek 115, Rysunek 116, Rysunek 117 i Rysunek 118 prezentują listy produktów w panelu administracyjnym podzielone według statusu.



Rysunek 115 Lista oczekujących produktów w panelu administracyjnym. Źródło: Opracowanie własne.

Panel administracyjny

- Panel główny
- Przepisy 2
- Produkty 1**
- Zgłoszenia 1
- Kategorie
- Grupy kategorii

+ Dodaj produkt

Oczekujące Zaakceptowane **Odrzucone** Usunięte

Szukaj...

ID	Nazwa	Autor	Utworzono ▼	Status	Akcje
1004	Ogórek konserwowy	mod	1.04.2026, 23:42:50	Zatwierdzony	 
1003	Ketchup	mod	1.04.2026, 23:42:28	Zatwierdzony	 
1002	Bułka	mod	1.04.2026, 23:42:16	Zatwierdzony	 
105	Sok z cytryny	user1	31.03.2026, 20:21:36	Zatwierdzony	 
104	Sól	user1	31.03.2026, 20:21:17	Zatwierdzony	 
103	Salata	user1	31.03.2026, 20:20:37	Zatwierdzony	 
102	Rukola	user1	31.03.2026, 20:20:31	Zatwierdzony	 
101	Pomidor	user1	31.03.2026, 20:20:02	Zatwierdzony	 
100	Bulion warzywny	SuperAdmin	31.03.2026, 19:58:31	Zatwierdzony	 
99	Koncentrat pomidorowy	SuperAdmin	31.03.2026, 19:58:31	Zatwierdzony	 

< 1 2 3 4 5 ... 11 > Strona 1 z 11

Pokaż: 10 ▼ łącznie: 108

Rysunek 116 Lista zatwierdzonych produktów w panelu administracyjnym. Źródło: Opracowanie własne.

Panel administracyjny

- Panel główny
- Przepisy 2
- Produkty 1**
- Zgłoszenia 1
- Kategorie
- Grupy kategorii

+ Dodaj produkt

Oczekujące Zaakceptowane **Odrzucone** Usunięte

Szukaj...

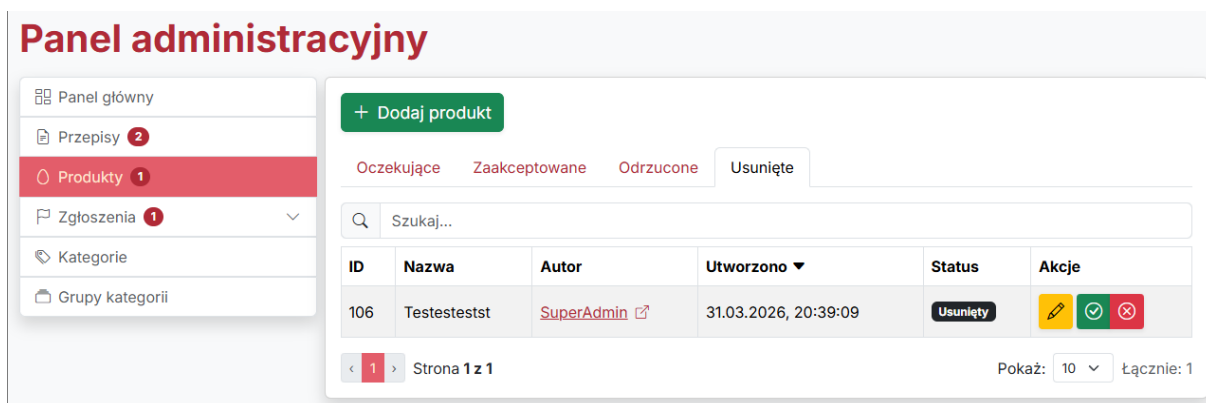
Powód odrzucenia:
nazwa nie odnosi się do
faktycznego produktu

ID	Nazwa	Autor	Utworzono ▼	Status	Akcje
2002	Test	SuperAdmin	15.06.2026, 21:25:33	Odrzucony	 

< 1 > Strona 1 z 1

Pokaż: 10 ▼ łącznie: 1

Rysunek 117 Lista odrzuconych produktów w panelu administracyjnym. Źródło: Opracowanie własne.



Rysunek 118 Lista usuniętych produktów w panelu administracyjnym. Źródło: Opracowanie własne.

6.4.1.2.1. Dodawanie produktów

Administrator lub moderator może dodać produkt poprzez naciśnięcie przycisku „Dodaj produkt”. Kliknięcie pokazuje okno dialogowe z formularzem. Aplikacja pozwala na dodanie produktów pojedynczo (Rysunek 119) lub wielu na raz poprzez import pliku CSV (Rysunek 120).

Rysunek 119 Okno dialogowe dodawania produktu pojedynczo. Źródło: Opracowanie własne.

Dodaj nowy produkt

Nazwy produktów muszą:

- zaczynać się z wielkiej litery,
- występować w liczbie pojedynczej,
- być w formie zdrobniałej tylko, gdy jest to zwyczajowe.

Dopuszcza się wielką literę w środku nazwy tylko w przypadku nazw własnych, ale staraj się ich unikać.

Przykłady: *Mąka pszenna, Przyprawa Maggi, Jajko, Pomidorek koktajlowy itp.*

☒ Dodaj wiele produktów (CSV)

Lista produktów (CSV):

Wybierz plik lista.csv

Plik powinien zawierać kolumnę z nazwami produktów, gdzie każdy produkt jest w osobnym wierszu.

Podgląd danych z CSV

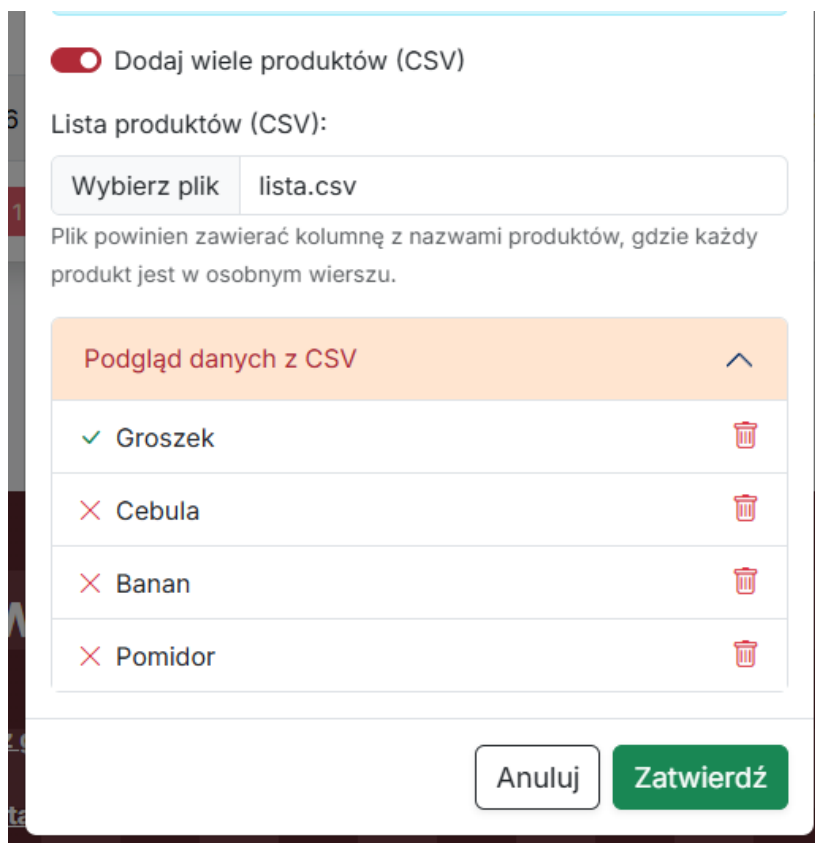
Groszek	
Cebula	
Banan	
Pomidor	

Anuluj

Zatwierdź

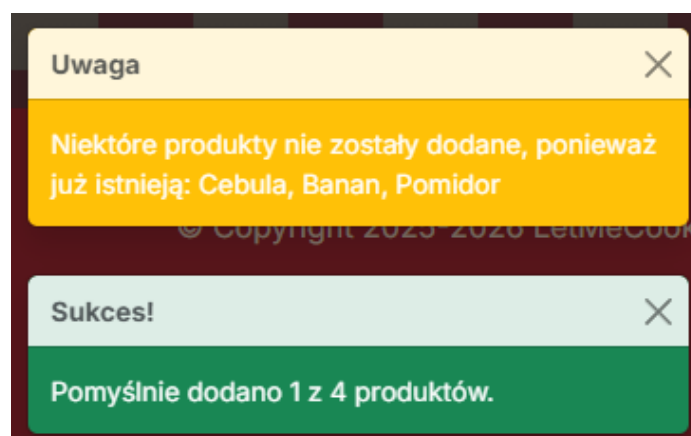
Rysunek 120 Okno dialogowe dodawania wielu produktów z pliku CSV. Źródło: Opracowanie własne.

Po naciśnięciu guzika „Zatwierdź” system dodaje do bazy produkty. Jeżeli dana pozycja istnieje już w bazie danych, wyświetlany jest odpowiedni błąd. Rysunek 121 ilustruje wynik niepełnego importu produktów z pliku. Pozycje zaimportowane pomyślnie oznaczone są symbolem wyboru, a niezaimportowane – ikoną X. Rysunek 122 prezentuje powiadomienia pokazane na skutek niepełnego importu.



Rysunek 121 Okno dialogowe dodawania wielu produktów z pliku CSV z wynikiem dodawania.

Źródło: Opracowanie własne.



Rysunek 122 Powiadomienia o dodaniu niewszystkich produktów z listy. Źródło: Opracowanie własne.

6.4.1.2.2. Odrzucanie produktów

Produkt, który nie jest w statusie „Odrzucony”, można odrzucić poprzez naciśnięcie obok niego czerwonego przycisku „Odrzuć” z ikoną X. Po kliknięciu pojawia się okno dialogowe z formularzem, do którego należy wpisać powód odrzucenia. Powód jest widoczny dla autora na stronie przepisu oraz w sekcji odrzuconych pozycji dla innych moderatorów i administratorów.

Odrzucenie produktu sprawia, że nowe i oczekujące przepisy zawierające ten produkt w składnikach nie mogą zostać zaakceptowane. Nie wpływa ono jednak na już zaakceptowane przepisy – pozostają one w niezmienionym stanie. Użytkownicy nie mogą też użyć ani zaproponować ponownie tego produktu w bazie.

Rysunek 123 prezentuje okno dialogowe odrzucania produktu.

Rysunek 123 Okno dialogowe odrzucania produktu. Źródło: Opracowanie własne.

6.4.1.2.3. Edycja produktów

Moderacja i administracja mają prawo do zmiany nazwy produktu w bazie danych poprzez panel administracyjny. Służy do tego żółty przycisk „Edytuj” z ikoną ołówka. Po naciśnięciu ukazuje się okno dialogowe z polem wprowadzania nowej nazwy.

Jeżeli produkt nie jest w statusie „Zaakceptowany”, jego edycja powoduje jednocześnie akceptację – o czym informuje odpowiedni komunikat w formularzu. Co ważne, modyfikacja nazwy składnika dotyczy jedynie jego wystąpień na liście składników w przepisach, natomiast w sekcji opisującej sposób przygotowania nazwa pozostaje niezmienną.

Rysunek 124 przedstawia okno dialogowe edycji produktu ze statusem „Oczekujący”.

Rysunek 124 Okno dialogowe edycji produktu - wariant ze zmianą statusu. Źródło: Opracowanie własne.

6.4.1.2.4. Usuwanie produktów

Panel administracyjny daje możliwość usunięcia produktu z bazy. Przeznaczony jest do tego czerwony przycisk „Usuń” z ikoną kosza. Po potwierdzeniu w oknie dialogowym produkt zostaje skasowany nieatrwal – dalej istnieje w bazie, ale ze statusem „Usunięty”. Akcja nie jest dostępna dla moderatorów – wyłącznie administratorzy i superadmini mają prawo do usuwania.

Serwis traktuje usunięte składniki analogicznie jak odrzucone – użytkownicy nie mogą ich użyć i zaproponować ponownie, jednak ich dotychczasowe wystąpienia w zaakceptowanych przepisach pozostają nienaruszone.

Rysunek 125 ilustruje okno dialogowe usuwania produktu pojawiające się po kliknięciu „Usuń”.

Rysunek 125 Okno dialogowe usuwania produktu. Źródło: Opracowanie własne.

6.4.1.3 Rozpatrywanie zgłoszeń komentarzy

Kolejną pozycją w menu panelu administracyjnego jest sekcja zgłoszeń komentarzy. Dzieli się one na nierozpatrzone (nieprzejrane dotychczas przez moderację) i rozpatrzone (zweryfikowane i odpowiednio usunięte lub pozostawione w serwisie).

Obie zakładki zawierają wyszukiwarkę, elementy paginacji oraz tabelę z podstawowymi informacjami. Dla pozycji nierozpatrzonych dostępne są dwie akcje: „Usuń komentarz i rozpatrz zgłoszenie” oraz „Zignoruj”. W tabeli zgłoszeń rozpatrzonych znajduje się natomiast dedykowana kolumna informująca o tym, czy dany komentarz został usunięty z serwisu.

Każdy wiersz w tabeli można rozwinąć, aby wyświetlić pełne informacje zarówno o samym zgłoszeniu, jak i o komentarzu. Szczegóły te są również widoczne w oknach dialogowych potwierdzających akcję.

Rysunek 126 prezentuje widok listy nierozpatrzonych zgłoszeń komentarzy wraz z rozwiniętymi szczegółami wybranego zgłoszenia, natomiast na Rysunek 127 przedstawiono zgłoszenia rozpatrzone.

Panel administracyjny

Panel główny
Przepisy 2
Produkty 1
Zgłoszenia 1
Komentarze 1
Kategorie
Grupy kategorii

Nierozpatrzone Rozpatrzone

Szukaj...

	ID	Autor komentarza	Komentarz	Powód zgłoszenia	Akcje
▼	1002	Kacper	testowy komentarz	komentarz nic nie wnosi	

Komentarz #1002

Autor komentarza	Kacper
Treść	testowy komentarz
Dodano	16.06.2026, 21:13:24
Pod przepisem	Link do przepisu

Zgłoszenie #1002

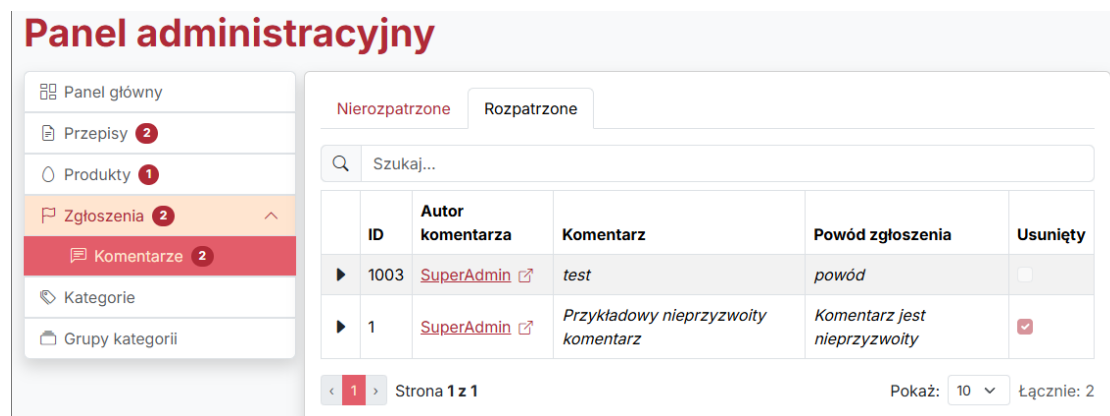
Zgłaszający	user1
Powód	komentarz nic nie wnosi
Data	16.06.2026, 21:14:01

► 2 [SuperAdmin](#) [Fajny przepis](#) [Brak powodu](#)

< 1 > Strona 1 z 1 Pokaż: 10 Łącznie: 2

Rysunek 126 Lista nierozpatrzonych zgłoszeń komentarzy w panelu administracyjnym.

Źródło: Opracowanie własne.



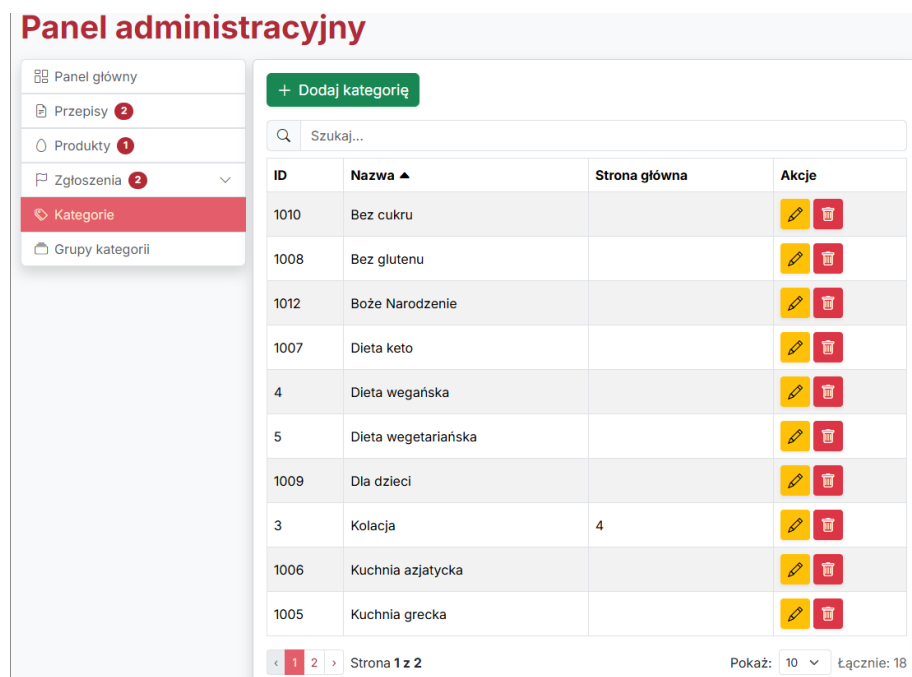
Rysunek 127 Lista rozpatrzonych zgłoszeń komentarzy w panelu administracyjnym. Źródło: Opracowanie własne.

6.4.1.4 Zarządzanie kategoriami

Następna sekcja panelu administracyjnego pozwala na zarządzanie kategoriami przepisów. Z jej poziomu administratorzy i moderatorzy mogą dodawać, edytować oraz usuwać poszczególne kategorie, a także umieszczać je na stronie głównej.

Dodanie nowej kategorii odbywa się poprzez naciśnięcie przycisku „Dodaj kategorię”. W oknie dialogowym należy podać unikalną nazwę i opcjonalnie pozycję kategorii na stronie głównej. Po pomyślnej walidacji, lista kategorii zostaje odświeżona.

Rysunek 128 ilustruje sekcję kategorii w panelu administracyjnym, a Rysunek 129 przedstawia okno dialogowe z formularzem dodania nowej kategorii.



Rysunek 128 Lista kategorii w panelu administracyjnym. Źródło: Opracowanie własne.

Dodaj nową kategorię

① Nazwy kategorii muszą:

- **zaczynać się z wielkiej litery,**
- **występować w liczbie pojedynczej.**

Przykłady: *Śniadanie, Obiad, Danie wegetariańskie itp.*

Nazwa kategorii:

Podaj nazwę kategorii...

Pozycja na stronie głównej:

Brak

Anuluj Zatwierdź

Rysunek 129 Okno dialogowe dodawania nowej kategorii. Źródło: Opracowanie własne.

6.4.1.5 Zarządzanie grupami kategorii

Ostatnia sekcja w panelu administracyjnym pozwala na zarządzanie grupami kategorii. Służą one do budowy dwupoziomowego paska nawigacyjnego złożonego z kategorii powiązanych ze sobą tematycznie. Widok ten pozwala dodać, edytować oraz usunąć grupę, a także przypisać do niej poszczególne elementy.

Aby utworzyć nową grupę, należy nacisnąć przycisk „Dodaj grupę kategorii” i wpisać unikalną nazwę w oknie dialogowym. Następnie do nowo utworzonej struktury przypisuje się odpowiednie kategorie za pomocą formularza otwieranego przyciskiem „Przypisz kategorie”. Przypisane kategorie do danej grupy są widoczne po rozwinięciu pozycji w tabeli.

Na Rysunek 130 zaprezentowano sekcję grup kategorii w panelu administracyjnym wraz z rozwiniętymi szczegółami danej pozycji. Rysunek 131 prezentuje formularz przypisywania kategorii do grupy.

Dieta
Kuchnie świata
Okazje
Posiłki
Ulubione przepisy

Panel administracyjny

Panel główny

Przepisy 2

Produkty 1

Zgłoszenia 2

Kategorie

Grupy kategorii

+ Dodaj grupę kategorii

Szukaj...

ID	Nazwa	Akcje
1004	Dieta	  
1003	Kuchnie świata	  

#1003 Kuchnie świata

Przypisz kategorię

ID	Nazwa	Akcje
1006	Kuchnia azjatycka	
1005	Kuchnia grecka	
1003	Kuchnia polska	
1004	Kuchnia włoska	

1005	Okazje	  
1	Posiłki	  

1

Strona 1 z 1

Pokaż: 10 łącznie: 4

Rysunek 130 Lista grup kategorii w panelu administracyjnym. Źródło: Opracowanie własne.

Dodaj kategorię do grupy

Wybierz kategorie:

Bez cukru x

Bez glutenu x

x

v

Anuluj

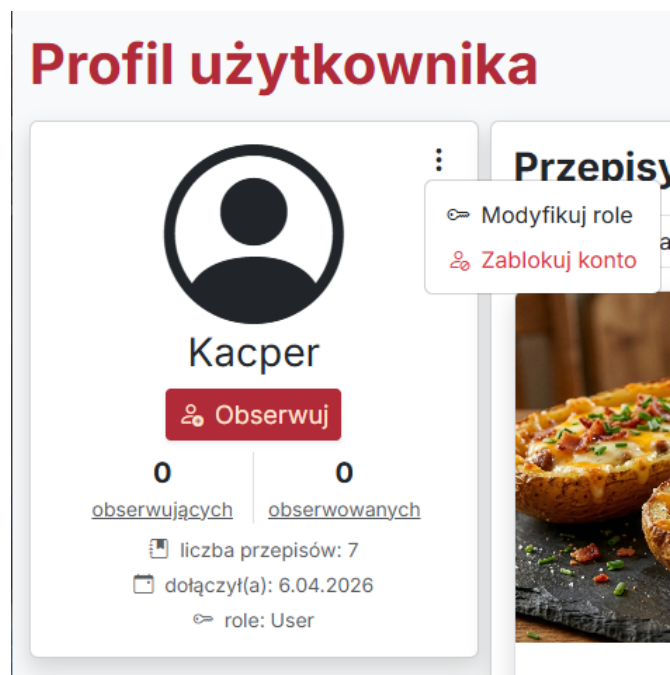
Zatwierdź

Rysunek 131 Okno dialogowe przypisywania kategorii do grupy. Źródło: Opracowanie własne.

160

6.4.2 Widok profilu użytkownika

Z poziomu profilu użytkownika moderator ma wgląd w przypisane mu role, a także dostęp do funkcji blokowania oraz odblokowywania konta. Administratorzy dysponują dodatkowo możliwością modyfikacji tych ról. Akcje te są zablokowane w przypadku wyświetlania własnego profilu zalogowanego administratora. Rysunek 132 prezentuje widok profilu użytkownika z rozwiniętym menu zarządzania kontem.



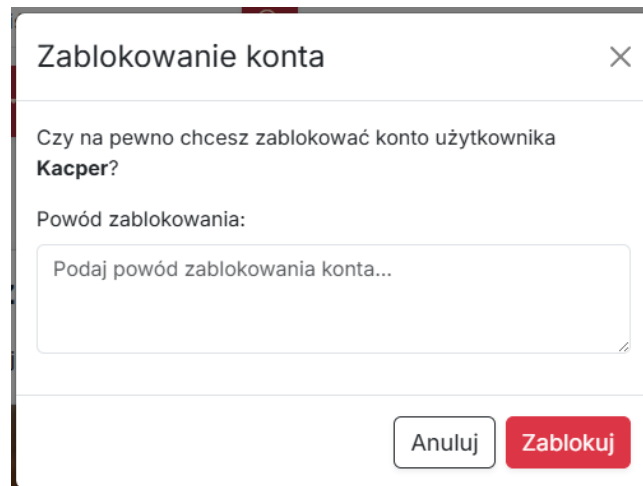
Rysunek 132 Widok profilu użytkownika dla administrator. Źródło: Opracowanie własne.

W niniejszej pracy zastosowano hierarchiczny model ról – zalogowana osoba nie może modyfikować blokady ani uprawnień konta, które posiada uprawnienia wyższe lub równe jej własnym. Zgodnie z tą strukturą:

- Moderator – może zarządzać wyłącznie blokadą kont zwykłych użytkowników;
- Administrator – ma prawo do zarządzania blokadami oraz rolami zarówno użytkowników, jak i moderatorów;
- SuperAdmin – posiada pełne uprawnienia, w tym możliwość modyfikacji blokad i ról użytkowników, moderatorów oraz administratorów

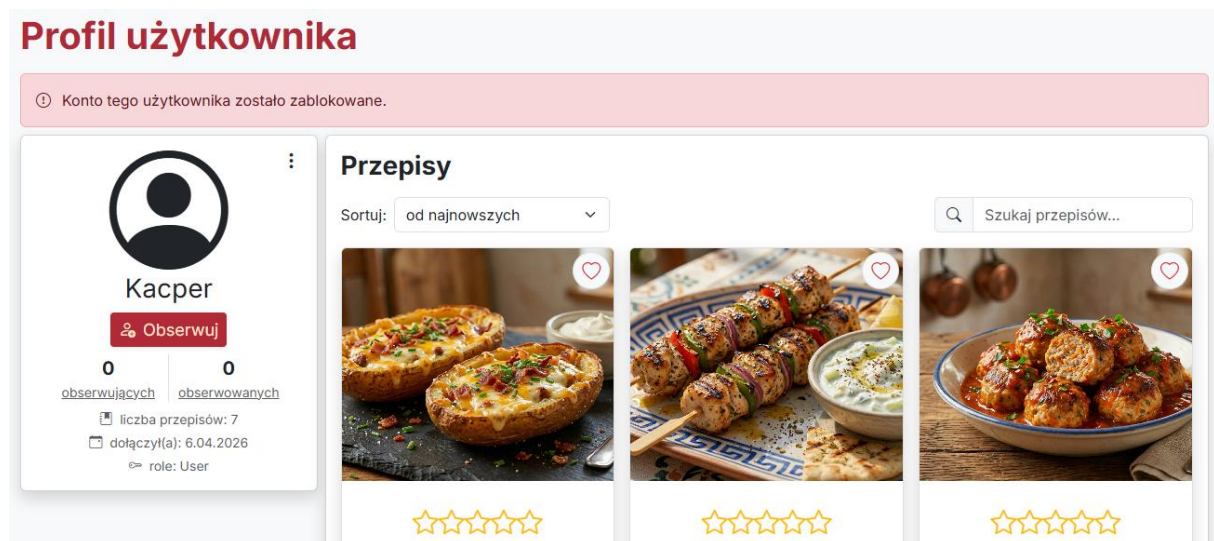
6.4.2.1 Blokowanie kont użytkowników

Proces blokowania konta rozpoczyna się po naciśnięciu przycisku „Zablokuj konto” w menu zarządzania użytkownikiem na jego profilu. W wyświetlonym formularzu należy zamieścić powód takiej operacji, który będzie widoczny dla użytkownika przy próbie zalogowania się do serwisu. Rysunek 133 przedstawia okno dialogowe blokowania konta.



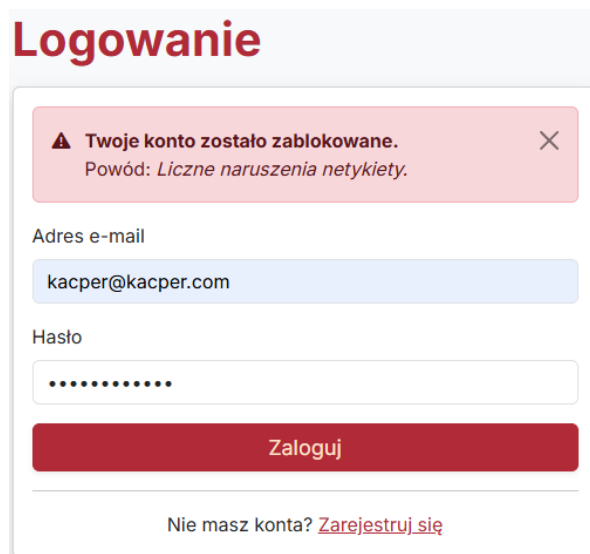
Rysunek 133 Okno dialogowe blokowania konta użytkownika. Źródło: Opracowanie własne.

Po zablokowaniu konta na profilu użytkownika informacje takie jak opublikowane przepisy, nazwa czy awatar pozostają nienaruszone, jednak nad całym profilem wyświetla się odpowiedni alert informacyjny. W menu zarządzania kontem pojawia się akcja odblokowania konta. Rysunek 134 przedstawia widok profilu zablokowanego użytkownika.



Rysunek 134 Widok profilu użytkownika z zablokowanym kontem. Źródło: Opracowanie własne.

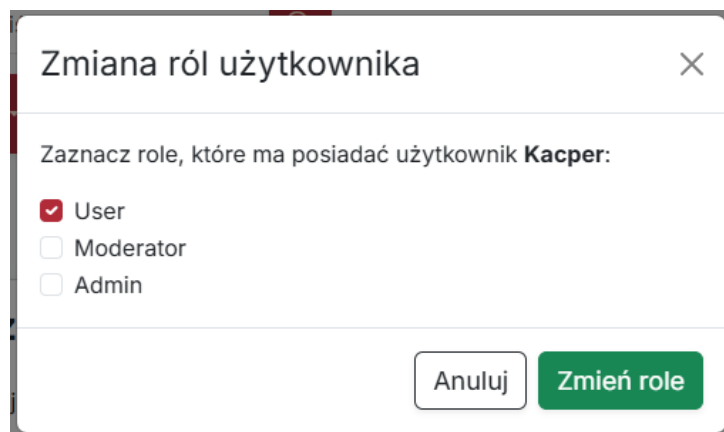
Konsekwencją założenia blokady na konto użytkownika jest uniemożliwienie zalogowania się na do serwisu. Taka próba kończy się wyświetleniem komunikatu na stronie logowania z podanym powodem, który został zaprezentowany na Rysunek 135.



Rysunek 135 Alert o zablokowaniu konta użytkownika na stronie logowania. Źródło: Opracowanie własne.

6.4.2.2 Zarządzanie rolami użytkowników

Proces zmiany uprawnień rozpoczyna się po naciśnięciu przycisku „Modyfikuj rolę” w menu na profilu użytkownika. W oknie dialogowym, przedstawionym na Rysunek 136, wyświetlane są role, którymi zalogowany administrator może zarządzać. Zaznaczając i odznaczając odpowiednie pola wyboru, administrator nadaje lub odbiera uprawnienia, a następnie zatwierdza całą operację przyciskiem „Zmień rolę”.



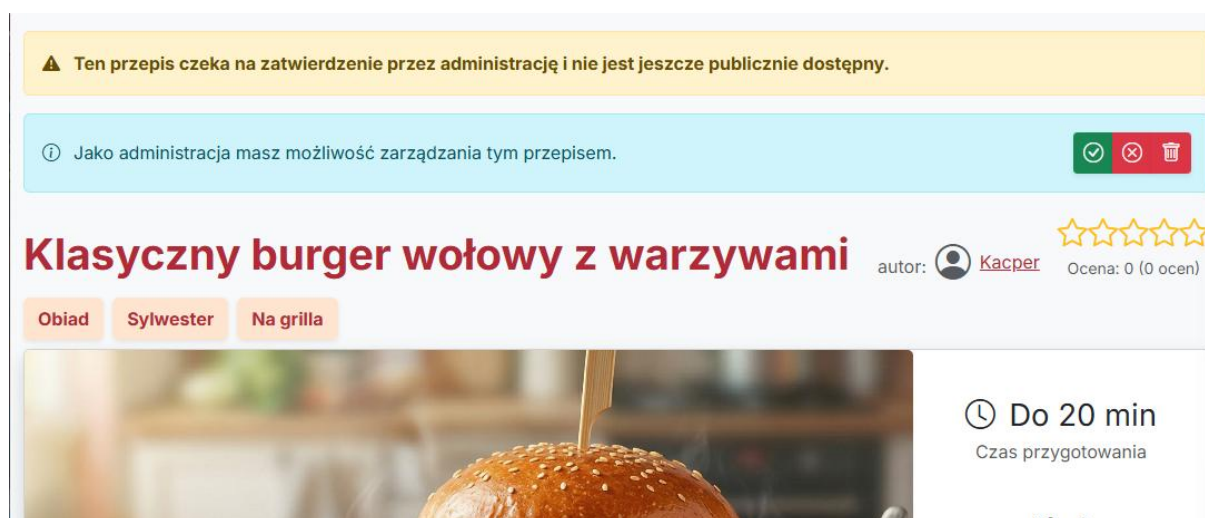
Rysunek 136 Okno dialogowe zarządzania rolami użytkownika. Źródło: Opracowanie własne.

6.4.3 Widok przepisu

Moderatorzy oraz administratorzy mogą zarządzać konkretnym przepisem, jego składnikami i komentarzami bezpośrednio z poziomu strony danego dania. Pozwala to na pominięcie wyszukiwania konkretnych pozycji w panelu administracyjnym. Poniżej omówiono akcje dostępne w tym widoku dla personelu zarządzającego.

6.4.3.1 Zmiana statusu przepisu

Niezależnie od statusu, na górze strony przepisu widoczny jest dla administracji alert informujący o możliwości zarządzania przepisem. Z jego poziomu dostępne są przyciski takich akcji jak: „Zaakceptuj”, „Odrzuć” czy „Usuń”. Ich działanie, wraz z oknami dialogowymi, jest tożsame z funkcjonalnościami w panelu administracyjnym, które zostały szczegółowo omówione w rozdziale 6.4.1.1. Na Rysunek 137 zaprezentowano widok przepisu ze statusem „Oczekujący” wraz z funkcjami widocznymi dla administratora.



Rysunek 137 Widok oczekującego przepisu z widocznymi akcjami dla administracji. Źródło: Opracowanie własne.

6.4.3.2 Zmiana statusu produktu

Dla składników, które nie są w statusie „Zaakceptowany”, widoczne są w sekcji „Składniki” przepisu ich statusy oraz przyciski akcji. Administrator może z tego poziomu dany produkt zaakceptować, odrzucić, usunąć oraz edytować. Działania te odbywają się analogicznie jak w panelu administracyjnym, co zostało opisane w rozdziale 6.4.1.2. Rysunek 138 ilustruje sekcję składników na stronie przepisu z widocznymi akcjami i statusami.

Składniki

KopiujiDodaj do listy zakupów

☐ Ogórek kiszony: 1 sztuka
☐ Cebula: 1/4 sztuki
☐ Wołowina: 300g
☐ Pieprz czarny Oczekujący



☐ Musztarda: 2 łyżeczki
☐ Pomidor: 1 sztuka
☐ Sałata: 2 liście
☐ Sól
☐ Bułka: 2 sztuki
☐ Ketchup:
☐ Test: test Odrzucony



Powód odrzucenia:
nazwa nie odnosi się do
faktycznego produktu

Rysunek 138 Widok składników przepisu z przyciskami akcji dla administratora. Źródło: Opracowanie własne.

6.4.3.3 Usuwanie komentarzy

Osoby moderujące mogą usuwać komentarze innych użytkowników bezpośrednio z poziomu sekcji komentarzy pod przepisem. Odbywa się to w taki sam sposób, jak opisano w rozdziale 6.3.7. Dodatkowo dla usuniętych komentarzy widoczne są dane autora oraz pierwotna treść, co pozwala na weryfikację i moderację kont niezależnie od obecnego stanu wpisu. Widok sekcji komentarzy dla moderatora został zaprezentowany na Rysunek 139.



adam skomentował(a): 34 sekundy temu

Komentarz numer 2

Odpowiedz

^ Ukryj odpowiedzi



Usunięty adam odpowiedział(a): 12 sekund temu

test



user1 skomentował(a): 19 minut temu

Komentarz user1

Zgłoś

Usuń

Odpowiedz

v Pokaż odpowiedzi (1)



Usunięty SuperAdmin skomentował(a): 17 godzin temu

test

Rysunek 139 Widok dla moderacji sekcji komentarzy pod przepisem. Źródło: Opracowanie własne.

165

7. Podsumowanie

Utworzenie od podstaw aplikacji zarówno webowej, jak i serwerowej, było wymagającym i czasochłonnym zadaniem. Pomimo napotkanych trudności oraz napiętych grafików w życiu prywatnym i zawodowym obu członków zespołu, udało się w pół roku zaimplementować zaawansowaną platformę „Let Me Cook”.

Na początku zespół dysponował jedynie ogólnym pomysłem na aplikację oraz listą kilku głównych funkcjonalności. Obaj członkowie mieli komercyjne, lecz podstawowe doświadczenie w programowaniu rozwiązań frontendowych i backendowych. W czasie prac nad projektem autorzy zdobyli doświadczenie w stawianiu aplikacji webowej i serwerowej od podstaw, a także w konteneryzacji środowiska. Dzięki sprawnej komunikacji i wypracowanym wewnątrz procesom zespół sukcesywnie planował, analizował, implementował i prezentował przyrosty w postępie prac.

Działania zespołu były regularnie oceniane zarówno w ramach wewnętrznych przeglądów kodu i testów, jak i co dwa tygodnie przez promotora oraz innych studentów podczas seminariów dyplomowych na uczelni. Otrzymywane opinie i wskazówki wpłynęły pozytywnie na ostateczną jakość i rozwój aplikacji.

W następnych podrozdziałach opisano szczegółowo efekty pracy, napotkane trudności, a także pomysły na dalszy rozwój projektu.

7.1. Efekty pracy

Głównym założeniem projektu było utworzenie nowoczesnej i prostej w obsłudze platformy do dzielenia się przepisami kulinarnymi. Autorzy treści mogą w serwisie dzielić się swoimi daniami z innymi użytkownikami bez potrzeby zakładania własnej strony internetowej. Portal pozwala na dodawanie zdjęć, umieszczenie sformatowanego tekstu, a także promowanie swoich mediów społecznościowych poprzez dołączanie materiałów wideo z tych zewnętrznych portali.

Użytkownicy szukający kulinarnych inspiracji zyskują w ten sposób serwis agregujący w jednym miejscu przepisy od wielu autorów. Dzięki mechanizmowi mapowania składników z bazy danych serwis pozwala na zaawansowane wyszukiwanie treści poprzez wykluczanie i uwzględnianie wybranych produktów, a także umożliwia tworzenie wspólnej, inteligentnej listy zakupów. Do dyspozycji użytkowników oddano również dodatkowe funkcjonalności społecznościowe, takie jak ocenianie, komentowanie oraz dodawanie przepisów do ulubionych.

Ponieważ *Let Me Cook* opiera się na treściach dodawanych przez użytkowników, zaprojektowano i wdrożono zestaw narzędzi oraz mechanizmów dla moderatorów serwisu. Każdy nowo dodany przepis i produkt musi najpierw zostać zweryfikowany ręcznie. Aplikacja w ramach panelu administracyjnego umożliwia kontrolę nad zasobami w przyjazny i łatwy sposób.

Powstała w ramach pracy aplikacja posiada responsywny układ strony, wskutek czego można z niej komfortowo korzystać na komputerze, telefonie i tablecie, bez potrzeby instalowania dodatkowego oprogramowania. Zwieńczeniem prac projektowych jest w pełni skonteneryzowane i gotowe do wdrożenia środowisko aplikacyjne.

7.2. Napotkane trudności

W trakcie realizacji projektu zespół natrafił na wiele trudności, które opóźniały i zakłócały planowy przebieg prac. Niektóre z nich wynikały z początkowego braku doświadczenia autorów, inne natomiast były spowodowane czynnikami zewnętrznymi.

Jednym z pierwszych wyzwań, jakie stanęło przed autorami, było poprawne skonfigurowanie środowiska deweloperskiego za pomocą narzędzia Docker. Architektura projektu oraz konieczność odseparowania od siebie konfiguracji lokalnej i produkcyjnej wymusiły utworzenie dedykowanych plików *Dockerfile*, *docker-compose* oraz *.env*. Ponieważ zespół nie posiadał wcześniej zaawansowanej wiedzy w tym zakresie, proces ten wymagał dodatkowego czasu na naukę. Szczególnie problematyczne okazało się właściwe rozplanowanie kolejności uruchamiania poszczególnych komponentów systemu.

Kolejną trudnością okazała się implementacja procesu autentyfikacji użytkownika. Po stronie backendu kłopotliwe było opcjonalne uwierzytelnianie żądań w sytuacjach, gdy dana końcówka API miała zwracać dodatkowe informacje tylko w przypadku, gdy użytkownik jest zalogowany. Z kolei po stronie frontendu wymagająca była obsługa mechanizmu automatycznego dołączania oraz warunkowego odświeżania tokena uwierzytelniającego przed wysłaniem zapytania do serwera. W rezultacie wdrożono autorskie mechanizmy pośredniczące, co wymagało od zespołu dodatkowego nakładu pracy i czasu na testy.

Niespodziewanym i znaczącym problemem stała się zmiana cenników i planów narzędzia Postman. Według [59] od 1 marca 2026 r. darmowy plan, z którego korzystał zespół, został pozbawiony funkcjonalności współdzielenia kolekcji. Stanowiły one podstawę testów API po stronie backendu i wspomagały implementację zapytań we frontendowej warstwie aplikacji. Aby zminimalizować negatywny wpływ tej zmiany na harmonogram prac, autorzy skorzystali z bezpłatnych wersji próbnych płatnych wariantów Postmana.

Ostatnią opisywaną trudnością zewnętrzną jest fakt, że otwartoźródłowa wersja serwera MinIO przestała być rozwijana przez twórców, a jej repozytorium w serwisie GitHub zostało zarchiwizowane 25 kwietnia 2026 roku [60]. O ile ta sytuacja nie wpływa bezpośrednio na obecne działanie aplikacji, o tyle brak wsparcia niesie ryzyko bezpieczeństwa związane z niełatwymi podatnościami. Wymusza to na autorach w przyszłości migrację magazynu plików na darmowe, lecz komercyjne rozwiązanie MinIO AIStor (będące następcą projektu *open-source*) lub znalezienie alternatywnego serwera obiektowego.

7.3. Dalszy rozwój

Ze względu na ograniczony czas oraz środki niektóre funkcjonalności, niebędące kluczowymi dla podstawowego działania systemu, nie zostały zrealizowane. Poniżej wymieniono dodatkowe pomysły na rozwój platformy, które pozwoliłyby użytkownikom na bardziej komfortowe korzystanie z serwisu lub podniosłyby stabilność całej aplikacji:

1. Wdrożenie certyfikatu SSL i szyfrowania transmisji – obecnie aplikacja obsługuje wyłącznie ruch nieszyfrowany (HTTP), co niesie ze sobą ryzyko przechwycenia poświadczeń lub tokenów autoryzujących użytkowników,
2. Migracja repozytorium plików (MinIO) – ze względu na zakończenie wsparcia dla wersji *open-source*, konieczne jest wdrożenie alternatywnego rozwiązania hostowanego samodzielnie (np. Garage), lub skorzystanie z komercyjnych usług jak MinIO AIStor czy Amazon S3,
3. Integracja z zewnętrznymi dostawcami tożsamości (OAuth2) – implementacja logowania za pomocą kont społecznościowych (np. Google, Facebook) ułatwiłaby proces rejestracji, co mogłoby przełożyć się pozytywnie na ilość użytkowników platformy,
4. System powiadomień oraz obsługa e-mail – automatyczne notyfikacje (wewnątrz aplikacji oraz mailowe) pozwoliłyby na bieżąco informować o interakcjach pod przepisami i komentarzami czy też o aktywnościach obserwowanych profili. Dodatkowo integracja z serwerem SMTP umożliwiłaby bezpieczny reset hasła oraz weryfikację adresów e-mail podczas rejestracji,

5. Internacjonalizacja (tzw. i18n) – dodanie obsługi innych języków niż polski (choćby angielskiego) pozwoliłoby na otwarcie serwisu na globalne grono odbiorców i ułatwiło skalowanie projektu.

7.4. Wnioski końcowe

W ciągu dwóch semestrów autorom udało się zrealizować wszystkie najważniejsze wymagania funkcjonalne i нефункционалне dla projektu Let Me Cook. Efektem prac jest przyjazna w obsłudze aplikacja skupiająca przepisy kulinarne, która posiada unikalne na rynku funkcjonalności, takie jak szerokie wsparcie dla filmów z zewnętrznych mediów społecznościowych, mechanizm automatycznego tworzenia listy składników czy inteligentna lista zakupów. Wytworzony produkt posiada solidne podstawy działania i jest gotowy do dalszego rozwoju oraz wdrożenia w środowisku produkcyjnym.

Autorzy w ramach niniejszej pracy opanowali tworzenie aplikacji z warstwą frontendową i serwerową, a także konteneryzację i tworzenie odizolowanych środowisk za pomocą Dockera. Rozwinęli również umiejętności współpracy, prezentacji postępu oraz pracy pod presją czasu. W rezultacie uzyskane doświadczenie przygotowało autorów do pracy jako inżynierowie oprogramowania.

Z perspektywy całego cyklu wytwórczego oprogramowania, kluczowym wnioskiem jest potwierdzenie elastyczności wybranego stosu technologicznego. Podział serwisu na niezależne aplikacje i mikroserwisy razem z zastosowaniem konteneryzacji ułatwił równoległe prowadzenie prac programistycznych i zminimalizował liczbę konfliktów podczas integracji kodu.

Projekt Let Me Cook udowodnił, że wykorzystanie nowoczesnych wzorców projektowych i sprawdzonych narzędzi pozwala niewielkiemu zespołowi na sprawne dostarczenie kompletnego produktu o wysokim stopniu innowacyjności. Powstała platforma odpowiada na realne potrzeby współczesnych użytkowników i amatorów gotowania, dając im przestrzeń do dzielenia się własnymi przepisami oraz ułatwiając ich wyszukiwanie i gromadzenie.

8. Bibliografia

- [1] „Strona główna Gotujmy.pl,” [Online]. Available: <https://gotujmy.pl/>. [Data uzyskania dostępu: 26 kwietnia 2026].
- [2] „Forum serwisu Gotujmy.pl,” [Online]. Available: <https://gotujmy.pl/forum/>. [Data uzyskania dostępu: 30 kwietnia 2026].
- [3] „Przepis "Torcik czekoladowy" w serwisie Gotujmy.pl,” [Online]. Available: <https://gotujmy.pl/torcik-czekoladowy,przepisy-ciasta-przepis,276618.html>. [Data uzyskania dostępu: 30 kwietnia 2026].
- [4] „Składnik "Jajko" w słowniku kulinarnym Gotujmy.pl,” [Online]. Available: <https://gotujmy.pl/jajko,skladnik-kulinarny,1339.html>. [Data uzyskania dostępu: 30 kwietnia 2026].
- [5] „Strona główna Smaker.pl,” [Online]. Available: <https://smaker.pl/>. [Data uzyskania dostępu: 26 kwietnia 2026].
- [6] „Przepis "Zupa Tom Kha" na portalu Smaker,” [Online]. Available: <https://smaker.pl/przepisy-zupy/przepis-zupa-tom-kha,2072251,smaker.html>. [Data uzyskania dostępu: 30 kwietnia 2026].
- [7] „Strona główna serwisu Allrecipes.com,” [Online]. Available: <https://www.allrecipes.com/>. [Data uzyskania dostępu: 27 kwietnia 2026].
- [8] „People Inc. - opis serwisu Allrecipes.com,” [Online]. Available: <https://www.people.inc/brands/food-drink/allrecipes>. [Data uzyskania dostępu: 27 kwietnia 2026].
- [9] „Przepis "Air Fryer Sesame Chicken Thighs" w serwisie Allrecipes,” [Online]. Available: <https://www.allrecipes.com/recipe/272399/air-fryer-sesame-chicken-thighs/>. [Data uzyskania dostępu: 30 kwietnia 2026].
- [10] „Strona główna aplikacji MyRecipes,” [Online]. Available: <https://www.myrecipes.com/>. [Data uzyskania dostępu: 27 kwietnia 2026].
- [11] „Wyniki badania Mediapanel za grudzień 2025 - internet,” [Online]. Available: <https://media-panel.pl/pl/aktualnosci/wyniki-badania-mediapanel-za-grudzien-2025-internet/>. [Data uzyskania dostępu: 27 kwietnia 2026].
- [12] „Strona główna bloga Aniagotuje.pl,” [Online]. Available: <https://aniagotuje.pl/>. [Data uzyskania dostępu: 27 kwietnia 2026].
- [13] „Przepis na babkę pistacjową na blogu Aniagotuje.pl,” [Online]. Available: <https://aniagotuje.pl/przepis/babka-pistacjowa>. [Data uzyskania dostępu: 27 kwietnia 2026].
- [14] Microsoft, „Wprowadzenie do platformy .NET - Dokumentacja .NET,” [Online]. Available: <https://learn.microsoft.com/pl-pl/dotnet/core/introduction>. [Data uzyskania dostępu: 30 kwietnia 2026].
- [15] Microsoft, „Proces zarządzanego wykonania - Dokumentacja .NET,” [Online]. Available: <https://learn.microsoft.com/pl-pl/dotnet/standard/managed-execution-process>. [Data uzyskania dostępu: 18 kwietnia 2026].
- [16] Microsoft, „Omówienie środowiska uruchomieniowego języka wspólnego (CLR) - Dokumentacja .NET,” [Online]. Available: <https://learn.microsoft.com/pl-pl/dotnet/standard/clr>. [Data uzyskania dostępu: 18 kwietnia 2026].

- [17] S. Z. A. Jafri, „Just-In-Time (JIT) vs. Ahead-Of-Time (AOT) Compilation in .NET Core,” Medium, 16 lipca 2025. [Online]. Available: https://medium.com/@syed.zeeshan.ali.jafri_99339/just-in-time-jit-vs-ahead-of-time-aot-compilation-in-net-core-c7650f1a366a. [Data uzyskania dostępu: 18 kwietnia 2026].
- [18] A. Joseph, C# 12 w pigułce. Kompendium programisty - ISBN: 978-83-289-1483-4, Helion, 2024.
- [19] Microsoft, „Wprowadzenie do zapytań LINQ w języku C# - Dokumentacja .NET,” [Online]. Available: <https://learn.microsoft.com/pl-pl/dotnet/csharp/linq/get-started/introduction-to-linq-queries>. [Data uzyskania dostępu: 2026 kwietnia 30].
- [20] Microsoft, „Podstawy odzyskiwania pamięci - Dokumentacja .NET,” [Online]. Available: <https://learn.microsoft.com/pl-pl/dotnet/standard/garbage-collection/fundamentals>. [Data uzyskania dostępu: 2026 kwietnia 30].
- [21] Microsoft, „Dokumentacja - ASP .NET Core,” [Online]. Available: <https://learn.microsoft.com/pl-pl/aspnet/core/overview>. [Data uzyskania dostępu: 18 kwietnia 2026].
- [22] Z. Shah, „ASP.NET Vs. ASP.NET Core: Key Difference Between Them,” Medium, 28 maja 2024. [Online]. Available: <https://devshi-bambhaniya.medium.com/asp-net-vs-asp-net-core-key-difference-between-them-2676d9de328e>. [Data uzyskania dostępu: 18 kwietnia 2026].
- [23] Microsoft, „Dokumentacja - Entity Framework Core,” [Online]. Available: <https://learn.microsoft.com/pl-pl/ef/core/>. [Data uzyskania dostępu: 21 kwietnia 2026].
- [24] M. K. Absy, „Differences Between Entity Framework (EF) and Entity Framework Core (EF Core),” Medium, 23 lipca 2024. [Online]. Available: <https://medium.com/@mohamad.absi.g/differences-between-entity-framework-ef-and-entity-framework-core-ef-core-8a249f4174fc>. [Data uzyskania dostępu: 21 kwietnia 2026].
- [25] R. Nwonah, „Understanding DbContext and DbSet in EF Core: A Creative Dive,” Medium, 21 kwietnia 2024. [Online]. Available: <https://medium.com/@nwonahr/understanding-dbcontext-and-dbset-in-ef-core-a-creative-dive-f0c5d6b92471>. [Data uzyskania dostępu: 22 kwietnia 2026].
- [26] P. Powell i I. Smalley, „What is REST API?,” IBM, 24 kwietnia 2025. [Online]. Available: <https://www.ibm.com/think/topics/rest-apis>. [Data uzyskania dostępu: 2026 kwietnia 25].
- [27] F. GÜRDAL, „What is Ocelot Gateway? (ASP.NET Core Set Up and Pilot Project),” Medium, 24 kwietnia 2024. [Online]. Available: <https://medium.com/@fatihgurdal/what-is-ocelot-gateway-asp-net-core-set-up-and-pilot-project-0130dc564b9f>. [Data uzyskania dostępu: 27 kwietnia 2026].
- [28] C. Wong, „Building API Gateway Using Ocelot In ASP.NET Core,” C# Corner, 2018. [Online]. Available: <https://www.c-sharpcorner.com/article/building-api-gateway-using-ocelot-in-asp-net-core/>. [Data uzyskania dostępu: 26 kwietnia 2026].
- [29] J. Patil, „Logging in .NET Core with Serilog,” Medium, 22 grudnia 2023. [Online]. Available: <https://medium.com/@jaydeepvpatil225/logging-in-net-core-with-serilog-3d4ee2ad48cd>. [Data uzyskania dostępu: 27 kwietnia 2026].
- [30] „Temat "TypeScript" w GitHub,” [Online]. Available: <https://github.com/topics/typescript>. [Data uzyskania dostępu: 4 maja 2026].

- [31] B. Cherny, Programming TypeScript, Sebastopol: O'Reilly Media Inc., 2019.
- [32] A. B. & E. Porcello, Learning React, Sebastopol: O'Reilly Media, 2020.
- [33] „Dokumentacja React,” [Online]. Available: <https://react.dev/>. [Data uzyskania dostępu: 6 maja 2026].
- [34] „Strona główna Next.js,” [Online]. Available: <https://nextjs.org/>. [Data uzyskania dostępu: 6 maja 2026].
- [35] „Dokumentacja Next.js,” [Online]. Available: <https://nextjs.org/docs>. [Data uzyskania dostępu: 6 maja 2026].
- [36] „Historia Bootstrap,” [Online]. Available: <https://getbootstrap.com/docs/5.3/about/overview>. [Data uzyskania dostępu: 7 maja 2026].
- [37] „Dokumentacja komponentu Card w Bootstrap,” [Online]. Available: <https://getbootstrap.com/docs/5.3/components/card>. [Data uzyskania dostępu: 7 maja 2026].
- [38] „Dlaczego React Bootstrap? - dokumentacja React Bootstrap,” [Online]. Available: <https://react-bootstrap.netlify.app/docs/getting-started/why-react-bootstrap>. [Data uzyskania dostępu: 7 maja 2026].
- [39] „Getting Started - Markdown Guide,” [Online]. Available: <https://www.markdownguide.org/getting-started/>. [Data uzyskania dostępu: 7 maja 2026].
- [40] „Dokumentacja frameworka TipTap,” [Online]. Available: <https://tiptap.dev/docs/editor/getting-started/overview>. [Data uzyskania dostępu: 11 maja 2026].
- [41] Microsoft, „Dokumentacja - SQL Server,” [Online]. Available: <https://learn.microsoft.com/pl-pl/sql/sql-server/what-is-sql-server?view=sql-server-ver17>. [Data uzyskania dostępu: 13 maja 2026].
- [42] I. Smalley i S. Susnjara, „What is Docker?,” IBM, 6 czerwca 2024. [Online]. Available: <https://www.ibm.com/think/topics/docker>. [Data uzyskania dostępu: 5 maja 2026].
- [43] P. Ramu, „What is Git?,” Medium, 6 listopada 2022. [Online]. Available: <https://medium.com/@prakashram1327/what-is-git-ad9e9b27cc26>. [Data uzyskania dostępu: 12 maja 2026].
- [44] Gen, „What is GitHub?,” Medium, 10 czerwca 2020. [Online]. Available: <https://fengyuan-yap.medium.com/what-is-github-260412a0dcd7>. [Data uzyskania dostępu: 12 maja 2026].
- [45] A. Caraffa, „Postman: what it is and how it works,” Openapi, 19 września 2025. [Online]. Available: <https://openapi.com/blog/postman-what-it-is-and-how-it-works>. [Data uzyskania dostępu: 12 maja 2026].
- [46] D. Bigosiński, „Co to jest narzędzie Figma, do czego służy i na jakich etapach projektowych jest użyteczna?,” Semcore, 14 czerwca 2024. [Online]. Available: <https://semcore.pl/co-to-jest-narzedzie-figma-do-czego-sluzi-i-na-jakich-etapach-projektowych-jest-uzyteczna/>. [Data uzyskania dostępu: 13 maja 2026].
- [47] Agami Technologies, „Working with MinIO,” Medium, 15 lutego 2022. [Online]. Available: <https://medium.com/@agamitechnologies/working-with-minio-39d08a0ad36>. [Data uzyskania dostępu: 12 maja 2026].

- [48] Amazon, „Amazon S3: 20 years of building what’s next,” [Online]. Available: <https://aws.amazon.com/s3/>. [Data uzyskania dostępu: 12 maja 2026].
- [49] Amazon, „Why AWS?,” [Online]. Available: <https://aws.amazon.com/what-is-aws/>. [Data uzyskania dostępu: 12 maja 2026].
- [50] A. Klimenko, „The Complete Guide to Data Transfer Objects (DTOs): From Basics to Enterprise Patterns,” Medium, 13 sierpnia 2025. [Online]. Available: <https://medium.com/@alxkm/the-complete-guide-to-data-transfer-objects-dtos-from-basics-to-enterprise-patterns-fcddd3a6bc9a>. [Data uzyskania dostępu: 1 czerwca 2026].
- [51] BrownKing, „Understanding the Service Layer in Software Architecture,” Medium, 27 grudnia 2024. [Online]. Available: <https://medium.com/@navroops38/understanding-the-service-layer-in-software-architecture-df9b676b3a16>. [Data uzyskania dostępu: 3 czerwca 2026].
- [52] „Server and Client Components - Dokumentacja Next.js,” [Online]. Available: <https://nextjs.org/docs/app/getting-started/server-and-client-components>. [Data uzyskania dostępu: 24 maja 2026].
- [53] „Dynamic Route Segments - Dokumentacja Next.js,” [Online]. Available: <https://nextjs.org/docs/app/api-reference/file-conventions/dynamic-routes>. [Data uzyskania dostępu: 24 maja 2026].
- [54] „Opis middleware.js - dokumentacja Next.js 15,” [Online]. Available: <https://nextjs.org/docs/15/app/api-reference/file-conventions/middleware>. [Data uzyskania dostępu: 30 maja 2026].
- [55] „Create a new extension - dokumentacja TipTap,” [Online]. Available: <https://tiptap.dev/docs/editor/extensions/custom-extensions/create-new>. [Data uzyskania dostępu: 25 maja 2026].
- [56] „Kod źródłowy biblioteki Unified,” [Online]. Available: <https://github.com/unifiedjs/unified>. [Data uzyskania dostępu: 27 maja 2026].
- [57] „Wprowadzenie do TanStack Table,” [Online]. Available: <https://tanstack.com/table/latest/docs/introduction>. [Data uzyskania dostępu: 31 maja 2026].
- [58] „Strona główna React Hook Form,” [Online]. Available: <https://www.react-hook-form.com/>. [Data uzyskania dostępu: 31 maja 2026].
- [59] „Coming to Postman in March: AI-native capabilities, a new API Catalog, and updated plans and pricing,” Postman, [Online]. Available: <https://blog.postman.com/new-capabilities-march-2026/>. [Data uzyskania dostępu: 17 czerwca 2026].
- [60] „Repozytorium MinIO na GitHub,” [Online]. Available: <https://github.com/minio/minio>. [Data uzyskania dostępu: 17 czerwca 2026].

9. Spis ilustracji

Rysunek 1 Widok zaplanowanych zadań w Notion. Źródło: Opracowanie własne.....	6
Rysunek 2 Strona główna portalu Gotujmy.pl. Źródło: [1].....	9
Rysunek 3 Forum serwisu Gotujmy.pl. Źródło: [2].....	10
Rysunek 4 Kreator przepisu portalu Gotujmy.pl – sekcja Dane podstawowe. Źródło: [1].....	11
Rysunek 5 Wycinek przepisu zamieszczonego na portalu Gotujmy.pl. Źródło: [3].....	12
Rysunek 6 Składnik "Jajko" w słowniku kulinarnym Gotujmy.pl. Źródło: [4].....	13
Rysunek 7 Strona główna serwisu Smaker.pl Źródło: [5].....	14
Rysunek 8 Dodawanie przepisu w Smaker.pl - sekcja Podstawowe informacje. Źródło: [5].....	15
Rysunek 9 Dodawanie przepisu w Smaker.pl - sekcja Składniki. Źródło: [5].....	15
Rysunek 10 Dodawanie przepisu w Smaker.pl - sekcja Sposób przygotowania. Źródło: [5].....	16
Rysunek 11 Dodawanie przepisu w Smaker.pl - sekcja Dodatkowe informacje. Źródło: [5].....	16
Rysunek 12 Widok przepisu na portalu Smaker - część główna. Źródło: [6].....	18
Rysunek 13 Widok przepisu na portalu Smaker - składniki i etapy przygotowania. Źródło: [6].....	19
Rysunek 14 Strona główna serwisu Allrecipes.com. Źródło: [7].....	20
Rysunek 15 Widok przepisu w serwisie Allrecipes. Źródło: [9].....	21
Rysunek 16 Widok zalogowanego użytkownika w aplikacji MyRecipes. Źródło: [10].....	22
Rysunek 17 Strona główna bloga Aniagotuje.pl. Źródło: [12].....	23
Rysunek 18 Wycinek widoku przepisu na blogu Aniagotuje.pl. Źródło: [13].....	24
Rysunek 19 Diagram aktorów systemu Let Me Cook. Źródło: Opracowanie własne.	34
Rysunek 20 Diagram przypadków użycia Osoby. Źródło: Opracowanie własne.	35
Rysunek 21 Diagram przypadków użycia Gościa. Źródło: Opracowanie własne.	35
Rysunek 22 Diagram przypadków użycia Użytkownika zalogowanego. Źródło: Opracowanie własne.	36
Rysunek 23 Diagram przypadków użycia Moderатора. Źródło: Opracowanie własne.	37
Rysunek 24 Diagram przypadków użycia Administratora. Źródło: Opracowanie własne.	38
Rysunek 25 Diagram przypadków użycia SuperAdmina. Źródło: Opracowanie własne.	38
Rysunek 26 Analityczny diagram klas. Źródło: Opracowanie własne.	39
Rysunek 27 Diagram stanu przepisu. Źródło: Opracowanie własne.	40
Rysunek 28 Diagram stanu produktu. Źródło: Opracowanie własne.	40
Rysunek 29 Diagram aktywności dla dodania nowego przepisu. Źródło: Opracowanie własne.	41
Rysunek 30 Diagram aktywności dla oznaczenia składnika w polu Sposób przygotowania. Źródło: Opracowanie własne.....	42
Rysunek 31 Schemat komunikacji w systemie wykorzystującym bramę dostępową. Źródło: [28].....	48
Rysunek 32 Przykład błędu TypeScript w edytorze Visual Studio Code. Źródło: Opracowanie własne.....	49
Rysunek 33 Przykładowa struktura strony w Next.js. Źródło: [35].....	52
Rysunek 34 Przykładowy widok komponentów Bootstrap na ekranie komputera. Źródło: [37].....	55
Rysunek 35 Przykładowy widok komponentów Bootstrap na ekranie telefonu. Źródło: [37].....	55
Rysunek 36 Widok interfejsu graficznego narzędzia Docker Desktop. Źródło: Opracowanie własne.	59
Rysunek 37 Widok interfejsu graficznego narzędzia Postman. Źródło: Opracowanie własne.	61
Rysunek 38 Projekt interfejsu graficznego w narzędziu Figma. Źródło: Opracowanie własne.	62
Rysunek 39 Widok panelu administracyjnego serwera MinIO. Źródło: Opracowanie własne.	63
Rysunek 40 Schemat komunikacji w systemie Let Me Cook. Źródło: Opracowanie własne.	64
Rysunek 41 Diagram sekwencji przedstawiający uwierzytelnianie oparte na tokenie dostępowym i tokenie odświeżania. Źródło: Opracowanie własne.	78
Rysunek 42 Struktura aplikacji webowej Let Me Cook w Visual Studio Code. Źródło: Opracowanie własne. .	79
Rysunek 43 Diagram sekwencji załadowania strony z oprogramowaniem pośredniczącym Next.js w aplikacji Let Me Cook. Źródło: Opracowanie własne.	86
Rysunek 44 Diagram sekwencji działania proxy z Route Handlers w aplikacji Let Me Cook. Źródło: Opracowanie własne.....	87
Rysunek 45 Interfejs graficzny komponentu MarkdownRecipeEditor. Źródło: Opracowanie własne.	92
Rysunek 46 Schemat konwertowania kodu za pomocą biblioteki Unified. Źródło: [56].....	97

Rysunek 47 Główny panel nawigacyjny w wersji desktopowej. Źródło: Opracowanie własne.....	105
Rysunek 48 Główny panel nawigacyjny w wersji na urządzenia mobilne. Źródło: Opracowanie własne.	106
Rysunek 49 Stopka w wersji desktopowej. Źródło: Opracowanie własne.	107
Rysunek 50 Widok formularza logowania. Źródło: Opracowanie własne.	108
Rysunek 51 Widok formularza rejestracji użytkownika. Źródło: Opracowanie własne.	109
Rysunek 52 Górna część widoku szczegółów przepisu w wersji desktopowej. Źródło: Opracowanie własne	111
Rysunek 53 Dolna część widoku szczegółów przepisu w wersji desktopowej. Źródło: Opracowanie własne.	112
Rysunek 54 Górna część widoku szczegółów przepisu w wersji na urządzenia mobilne. Źródło: Opracowanie własne.	113
Rysunek 55 Część widoku szczegółów przepisu z krótkim opisem i składnikami w wersji na urządzenia mobilne. Źródło: Opracowanie własne.....	114
Rysunek 56 Sekcja „Sposób przygotowania” w widoku szczegółów przepisu w wersji na urządzenia mobilne. Źródło: Opracowanie własne.	114
Rysunek 57 Sekcja „Komentarze” w widoku szczegółów przepisu w wersji na urządzenia mobilne. Źródło: Opracowanie własne.....	115
Rysunek 58 Widok wyszukiwania przepisów w wersji desktopowej. Źródło: Opracowanie własne.....	116
Rysunek 59 Widok wyszukiwania przepisów na urządzeniach mobilnych. Źródło: Opracowanie własne.	116
Rysunek 60 Widok profilu użytkownika w wersji desktopowej. Źródło: Opracowanie własne.	117
Rysunek 61 Widok profilu użytkownika na urządzeniach mobilnych. Źródło: Opracowanie własne.	117
Rysunek 62 Widok kont obserwujących użytkownika. Źródło: Opracowanie własne.	118
Rysunek 63 Sekcja „Podstawowe informacje” w widoku „Nowy przepis”. Źródło: Opracowanie własne.	120
Rysunek 64 Wyszukiwanie produktów w sekcji „Sposób przygotowania” w widoku „Nowy przepis”. Źródło: Opracowanie własne.....	121
Rysunek 65 Okno dialogowe „Zaproponuj nowy produkt”. Źródło: Opracowanie własne.	121
Rysunek 66 Pole ilości składnika w sekcji „Sposób przygotowania” w widoku „Nowy przepis”. Źródło: Opracowanie własne.....	122
Rysunek 67 Sekcja „Wykonanie” w widoku „Nowy przepis”. Źródło: Opracowanie własne.	123
Rysunek 68 Sekcja „Dodatkowe informacje” w widoku „Nowy przepis”. Źródło: Opracowanie własne.	124
Rysunek 69 Oczekujący na zatwierdzenie przepis w widoku „Mój profil”. Źródło: Opracowanie własne.	124
Rysunek 70 Widok szczegółów oczekującego na zatwierdzenie przepisu. Źródło: Opracowanie własne.	125
Rysunek 71 Widok „Edycja przepisu”. Źródło: Opracowanie własne.	126
Rysunek 72 Opcja usuwania przepisu z widoku „Mój profil”. Źródło: Opracowanie własne.	127
Rysunek 73 Opcja usuwania przepisu z widoku szczegółów przepisu. Źródło: Opracowanie własne.	127
Rysunek 74 Okno dialogowe „Potwierdzenie usunięcia”. Źródło: Opracowanie własne.....	128
Rysunek 75 Powiadomienie o pomyślnym usunięciu przepisu. Źródło: Opracowanie własne.	128
Rysunek 76 Widok „Mój profil” z sekcją „O mnie” w trakcie modyfikacji. Źródło: Opracowanie własne.	129
Rysunek 77 Powiadomienie o pomyślnej modyfikacji sekcji „O mnie” w widoku „Mój profil”. Źródło: Opracowanie własne.....	129
Rysunek 78 Okno dialogowe „Edytuj zdjęcie profilowe”. Źródło: Opracowanie własne.	130
Rysunek 79 Powiadomienie o pomyślnej zmianie zdjęcia profilowego w widoku „Mój profil”. Źródło: Opracowanie własne.....	130
Rysunek 80 Karuzela z przepisem dodanym do ulubionych. Źródło: Opracowanie własne.	131
Rysunek 81 Powiadomienie o pomyślnym dodaniu przepisu do ulubionych. Źródło: Opracowanie własne.	131
Rysunek 82 Powiadomienie o pomyślnym usunięciu przepisu z ulubionych. Źródło: Opracowanie własne..	131
Rysunek 83 Widok „Ulubione przepisy”. Źródło: Opracowanie własne.	132
Rysunek 84 Widok „Lista zakupów”. Źródło: Opracowanie własne.	133
Rysunek 85 Okno dialogowe „Dodaj składniki do listy zakupów”. Źródło: Opracowanie własne.....	133
Rysunek 86 Powiadomienie o pomyślnym dodaniu składników do listy zakupów. Źródło: Opracowanie własne.	134
Rysunek 87 Powiadomienie o pomyślnym zaobserwowaniu użytkownika. Źródło: Opracowanie własne.	134
Rysunek 88 Powiadomienie o pomyślnym usunięciu obserwacji użytkownika. Źródło: Opracowanie własne.	134
Rysunek 89 Widok kont obserwujących użytkownika w przypadku użytkownika zalogowanego. Źródło: Opracowanie własne.....	135
Rysunek 90 Sekcja „Obserwowani” na stronie głównej. Źródło: Opracowanie własne.	135

Rysunek 91 Widok wyników wyszukiwania po zastosowaniu filtra „Tylko od obserwowanych”. Źródło: Opracowanie własne.....	136
Rysunek 92 Widok szczegółów przepisu z rozwiniętym dymkiem oceny. Źródło: Opracowanie własne.	137
Rysunek 93 Widok szczegółów przepisu z rozwiniętym dymkiem oceny w przypadku ocenienia przepisu przez użytkownika. Źródło: Opracowanie własne.....	138
Rysunek 94 Formularz tworzenia komentarza. Źródło: Opracowanie własne.	138
Rysunek 95 Formularz tworzenia odpowiedzi do komentarza. Źródło: Opracowanie własne.	139
Rysunek 96 Okno dialogowe „Usuń komentarz”. Źródło: Opracowanie własne.	139
Rysunek 97 Zainicjowanie procesu zgłoszenia komentarza. Źródło: Opracowanie własne.	140
Rysunek 98 Okno dialogowe „Zgłoś komentarz”. Źródło: Opracowanie własne.	140
Rysunek 99 Komentarz ukryty po zgłoszeniu. Źródło: Opracowanie własne.	141
Rysunek 100 Komentarz odkryty po zgłoszeniu. Źródło: Opracowanie własne.	141
Rysunek 101 Widok „Ustawienia”. Źródło: Opracowanie własne.	142
Rysunek 102 Menu użytkownika z widocznym linkiem do panelu administracyjnego. Źródło: Opracowanie własne	143
Rysunek 103 Ekran główny panelu administracyjnego - widok desktopowy. Źródło: Opracowanie własne .	143
Rysunek 104 Ekran główny panelu administracyjnego - widok mobilny. Źródło: Opracowanie własne	144
Rysunek 105 Lista oczekujących przepisów w panelu administracyjnym. Źródło: Opracowanie własne.	145
Rysunek 106 Lista zaakceptowanych przepisów w panelu administracyjnym. Źródło: Opracowanie własne.	145
Rysunek 107 Lista odrzuconych przepisów w panelu administracyjnym. Źródło: Opracowanie własne.	146
Rysunek 108 Lista usuniętych przepisów w panelu administracyjnym. Źródło: Opracowanie własne.	146
Rysunek 109 Okno dialogowe zatwierdzania przepisu - wariant z wszystkimi zaakceptowanymi składnikami. Źródło: Opracowanie własne.	147
Rysunek 110 Okno dialogowe zatwierdzania przepisu - wariant z oczekującymi składnikami. Źródło: Opracowanie własne.....	147
Rysunek 111 Okno dialogowe zatwierdzania przepisu - wariant z odrzuconym składnikiem. Źródło: Opracowanie własne.....	148
Rysunek 112 Powiadomienia o akceptacji przepisu i produktów w przepisie. Źródło: Opracowanie własne.	148
Rysunek 113 Okno dialogowe odrzucania przepisu. Źródło: Opracowanie własne.	149
Rysunek 114 Okno dialogowe usunięcia przepisu w panelu administracyjnym. Źródło: Opracowanie własne.	149
Rysunek 115 Lista oczekujących produktów w panelu administracyjnym. Źródło: Opracowanie własne.	150
Rysunek 116 Lista zatwierdzonych produktów w panelu administracyjnym. Źródło: Opracowanie własne.	151
Rysunek 117 Lista odrzuconych produktów w panelu administracyjnym. Źródło: Opracowanie własne.	151
Rysunek 118 Lista usuniętych produktów w panelu administracyjnym. Źródło: Opracowanie własne.	152
Rysunek 119 Okno dialogowe dodawania produktu pojedynczo. Źródło: Opracowanie własne.....	152
Rysunek 120 Okno dialogowe dodawania wielu produktów z pliku CSV. Źródło: Opracowanie własne.	153
Rysunek 121 Okno dialogowe dodawania wielu produktów z pliku CSV z wynikiem dodawania. Źródło: Opracowanie własne.....	154
Rysunek 122 Powiadomienia o dodaniu niewszystkich produktów z listy. Źródło: Opracowanie własne.	154
Rysunek 123 Okno dialogowe odrzucania produktu. Źródło: Opracowanie własne.....	155
Rysunek 124 Okno dialogowe edycji produktu - wariant ze zmianą statusu. Źródło: Opracowanie własne..	156
Rysunek 125 Okno dialogowe usuwania produktu. Źródło: Opracowanie własne.	156
Rysunek 126 Lista nierozpatrzonych zgłoszeń komentarzy w panelu administracyjnym. Źródło: Opracowanie własne.	157
Rysunek 127 Lista rozpatrzonych zgłoszeń komentarzy w panelu administracyjnym. Źródło: Opracowanie własne.	158
Rysunek 128 Lista kategorii w panelu administracyjnym. Źródło: Opracowanie własne.....	158
Rysunek 129 Okno dialogowe dodawania nowej kategorii. Źródło: Opracowanie własne.	159
Rysunek 130 Lista grup kategorii w panelu administracyjnym. Źródło: Opracowanie własne.	160
Rysunek 131 Okno dialogowe przypisywania kategorii do grupy. Źródło: Opracowanie własne.	160
Rysunek 132 Widok profilu użytkownika dla administrator. Źródło: Opracowanie własne.	161
Rysunek 133 Okno dialogowe blokowania konta użytkownika. Źródło: Opracowanie własne.	162
Rysunek 134 Widok profilu użytkownika z zablokowanym kontem. Źródło: Opracowanie własne.	162

Rysunek 135 Alert o zablokowaniu konta użytkownika na stronie logowania. Źródło: Opracowanie własne.	163
Rysunek 136 Okno dialogowe zarządzania rolami użytkownika. Źródło: Opracowanie własne.	163
Rysunek 137 Widok oczekującego przepisu z widocznymi akcjami dla administracji. Źródło: Opracowanie własne.	164
Rysunek 138 Widok składników przepisu z przyciskami akcji dla administratora. Źródło: Opracowanie własne.	165
Rysunek 139 Widok dla moderacji sekcji komentarzy pod przepisem. Źródło: Opracowanie własne.	165

10. Spis tabel

Tabela 1 Podział i przydział zadań pogrupowany według faz projektu. Źródło: Opracowanie własne.	7
Tabela 2 Słownik pojęć. Źródło: Opracowanie własne.....	26
Tabela 3 Wymagania funkcjonalne dla Gościa. Źródło: Opracowanie własne.....	27
Tabela 4 Wymagania funkcjonalne dla Użytkownika zalogowanego. Źródło: Opracowanie własne.	28
Tabela 5 Wymagania funkcjonalne dla Moderadora. Źródło: Opracowanie własne.	30
Tabela 6 Wymagania funkcjonalne dla Administratora. Źródło: Opracowanie własne.	32
Tabela 7 Wymagania funkcjonalne dla SuperAdmina. Źródło. Opracowanie własne.....	33
Tabela 8 Wymagania нефункционалне. Źródło: Opracowanie własne.....	33
Tabela 9 Zestawienie otwartych portów aplikacji według środowiska oraz ich alokacji (host/kontener). Źródło: Opracowanie własne.	102

11. Spis listingów

Listing 1 Przykładowy program w języku C#. Źródło: Opracowanie własne.	44
Listing 2 Przykładowy podstawowy kontroler w ASP.NET Core. Źródło: Opracowanie własne.	45
Listing 3 Przykładowa klasa dziedzicząca po DbContext. Źródło: Opracowanie własne.	46
Listing 4 Przykładowy program napisany w TypeScript. Źródło: Opracowanie własne.	50
Listing 5 Przykładowy komponent React. Źródło: [33].....	50
Listing 6 Przykładowy komponent React korzystający z hooków. Źródło: Opracowanie własne.	51
Listing 7 Przykładowy plik layout.tsx w Next.js. Źródło: [35]	52
Listing 8 Przykładowa strona w Next.js. Źródło: [35]	53
Listing 9 Przykładowy układ dwóch komponentów Bootstrap. Źródło: [37]	54
Listing 10 Implementacja komponentu Alert w React Bootstrap. Źródło: [38]	55
Listing 11 Implementacja komponentu Alert w Bootstrap. Źródło: [38]	56
Listing 12 Przykładowy tekst w Markdown. Źródło: Opracowanie własne.	57
Listing 13 Przykładowa kwerenda w języku T-SQL. Źródło: Opracowanie własne.	58
Listing 14 Rejestracja zależności w pliku Program.cs w mikroserwisie UserService. Źródło: Opracowanie własne.	66
Listing 15 Obiekt transferu danych wykorzystywany do zwracania informacji o profilu użytkownika. Źródło: Opracowanie własne.....	66
Listing 16 Fragment kontrolera UsersController przedstawiający wykorzystanie warstwy serwisów. Źródło: Opracowanie własne.....	67
Listing 17 Fragment klasy UserService należącej do warstwy serwisów. Źródło: Opracowanie własne.....	68
Listing 18 Zawartość pliku Program.cs w mikroserwisie RecipesService. Źródło: Opracowanie własne.	69
Listing 19 Zawartość pliku ocelot definiującego trasy dla list zakupowych. Źródło: Opracowanie własne.	72
Listing 20 Implementacja klasy OptionalAuthHandler. Źródło: Opracowanie własne.	74
Listing 21 Operacja asynchronicznego pobrania z bazy danych listy produktów do odczytu. Źródło: Opracowanie własne.....	76
Listing 22 Operacja asynchronicznego usunięcia wystąpienia produktu na liście zakupów. Źródło: Opracowanie własne.....	76
Listing 23 Fragment komponentu serwerowego tworzącego panel administracyjny. Źródło: Opracowanie własne.	81
Listing 24 Fragment komponentu klienckiego AddCategoryButton. Źródło: Opracowanie własne.	82
Listing 25 Fragment strony profilu użytkownika korzystającej z Dynamic Route Segments. Źródło: Opracowanie własne.....	84
Listing 26 Fragment pliku variables.scss nadpisujący zmienne Bootstrap. Źródło: Opracowanie własne.	88
Listing 27 Fragment pliku konfiguracyjnego Next.js next.config.ts. Źródło: Opracowanie własne.	89
Listing 28 Fragment kodu komponentu ModalButton. Źródło: Opracowanie własne.	89
Listing 29 Inicjalizacja instancji edytora TipTap w komponencie MarkdownRecipeEditor. Źródło: Opracowanie własne.....	91
Listing 30 Fragment kodu rozszerzenia IngredientMention dla edytora TipTap. Źródło: Opracowanie własne.	93
Listing 31 Fragment listy obsługiwanych jednostek przez bibliotekę ingredient-amount. Źródło: Opracowanie własne.	94
Listing 32 Funkcja biblioteki ingredient-amount odpowiedzialna za parsowanie ilości. Źródło: Opracowanie własne.	95
Listing 33 Komponent CategoriesTable prezentujący listę kategorii za pomocą komponentu ResultTable z TanStack Table. Źródło: Opracowanie własne.	97
Listing 34 Przykład inicjalizacji nowego formularza w React Hook Form. Źródło: Opracowanie własne.....	99
Listing 35 Fragment formularza rejestracji z użyciem React Hook Form. Źródło: Opracowanie własne.	99
Listing 36 Przykładowa konfiguracja pliku db.env (dane zanonimizowane). Źródło: Opracowanie własne. ..	103
Listing 37 Przykładowa konfiguracja pliku admin.env (dane zanonimizowane). Źródło: Opracowanie własne.	103

Listing 38 Przykładowa konfiguracja pliku `aws.env` (dane zanonimizowane). Źródło: Opracowanie własne. 103

Listing 39 Przykładowa konfiguracja pliku `jwt.env` (dane zanonimizowane). Źródło: Opracowanie własne.. 103