AFK CookBook - dokumentacja inżynierii wymagań

Filip Korus, Aleksandra Łapczuk, Krzysztof Wdowczyk

Przedmowa

Celem naszego projektu jest stworzenie ogólnodostępnej książki kucharskiej dla osób, które chcą zarówno czerpać jak i dzielić się z innymi swoimi daniami oraz przepisami. Strona ma ułatwiać również przechowywanie własnych przepisów w prywatnej książce kucharskiej. Naszym celem jest nie tylko dostarczenie praktycznego narzędzia do przechowywania przepisów, ale również stworzenie społeczności, w której użytkownicy mogą dzielić się swoją pasją do gotowania, wymieniać doświadczeniami oraz inspirować się nawzajem.
AFK (Away From Kitchen) Cookbook ma pełnić funkcję nowego, kulinarnego Social Medium.

Słownik pojęć technicznych

- Docker - Platforma umożliwiająca konteneryzację aplikacji, zaprojektowana z myślą o tworzeniu, dostarczaniu i uruchamianiu aplikacji z użyciem technologii konteneryzacji
- Nginx - jeden z najpopularniejszych serwerów HTTP o otwartym kodzie źródłowym. Charakteryzuje się lekką i wydajną architekturą oraz potrafi wytrzymać silne obciążenia przy jednocześnie niskiej zajętości zasobów
- OAuth2 - Protokół uwierzytelniania, umożliwiający użytkownikom logowanie się przy użyciu swojego konta Google. W kontekście Cookbook, OAuth2 jest wykorzystywany do bezpiecznego autoryzowania dostępu do aplikacji przez użytkowników.
- REST API (Representational State Transfer Application Programming Interface) - styl architektoniczny służący do komunikacji między aplikacjami. Opiera się na protokole HTTP i definiuje, w jaki sposób zasoby internetowe mogą być adresowane i manipulowane za pomocą zapytań HTTP, np: GET, POST, PUT, DELETE.
- Punkt końcowy (endpoint) - adres, na który wysyłane są żądania.
- Relacyjna baza danych - rodzaj bazy danych, który pozwala przechowywać powiązane ze sobą elementy danych i zapewnia do nich dostęp. Relacyjne bazy danych są oparte na modelu relacji między tabelami. To prosty i intuicyjny sposób przedstawiania danych. W relacyjnej bazie danych każdy wiersz tabeli jest rekordem z unikatowym identyfikatorem nazywanym kluczem.
- ORM (Object-Relational Mapping) - technika, która służy do łączenia relacyjnych baz danych z obiektowymi językami programowania. Umożliwia ona programistom pracę z bazami danych w sposób obiektowy.

- PHPMyAdmin - panel do zarządzania bazą danych z poziomu przeglądarki internetowej.

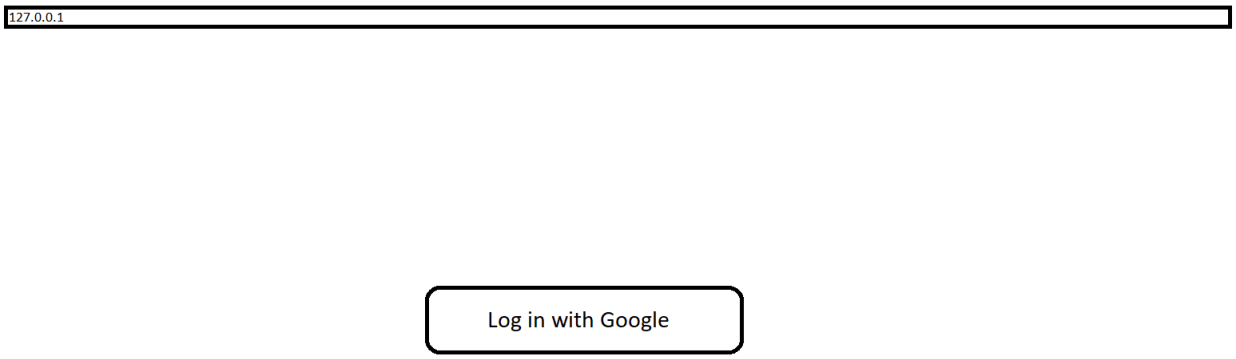
Macierz kompetencji zespołu

Kompetencje	Filip Korus	Aleksandra Łapczuk	Krzysztof Wdowczyk
Znajomość Node.js	Tak	Nie	Nie
Znajomość React.js	Tak	Tak	Nie
Znajomość TypeScript	Tak	Podstawy	Nie
Tworzenie UI/UX	Tak	Tak	Tak
Znajomość HTML, CSS	Tak	Tak	Tak
Bazy danych	Tak	Podstawy	Tak
Docker	Tak	Nie	Nie
Testowanie oprogramowania	Uczę się	Nie	Podstawy

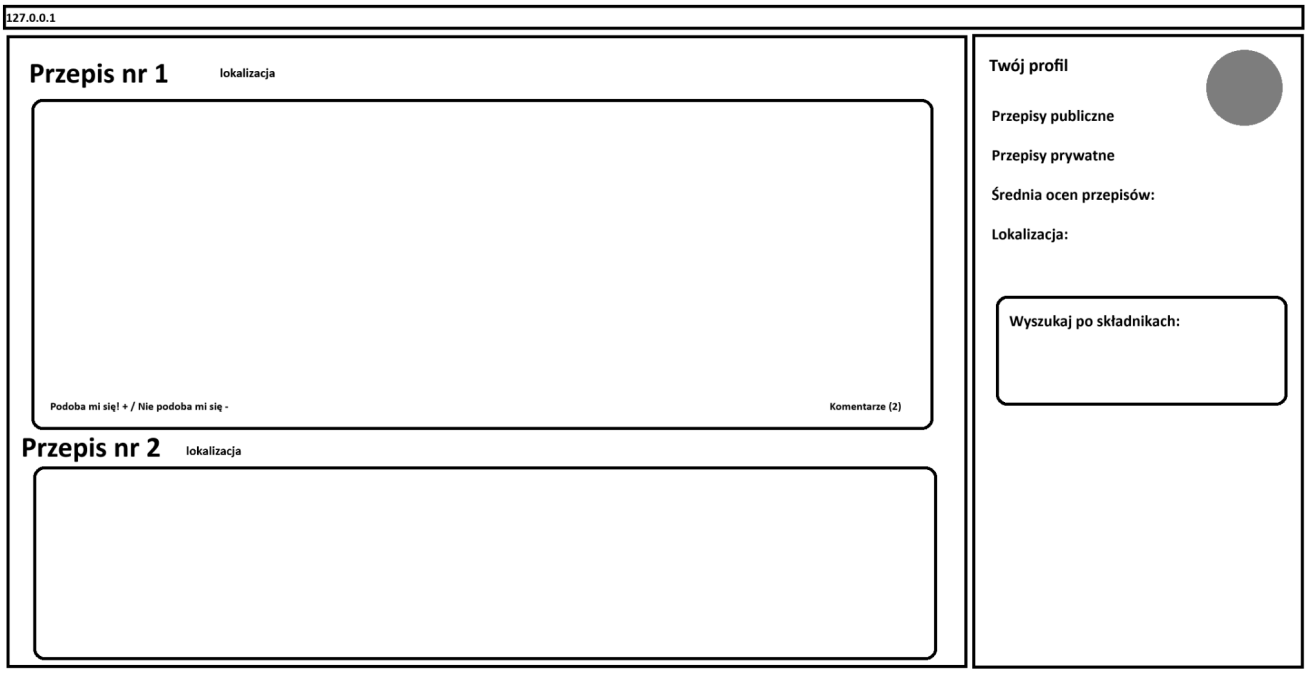
Wymagania użytkownika

Stworzenie książki kucharskiej online, która pozwala na dzielenie się przepisami między użytkownikami. Zalogowani użytkownicy mogą dodawać swoje przepisy i przechowywać je jako publiczne lub prywatne na swoim profilu. Jeśli użytkownik zdecyduje się dodać swój przepis na publiczny “wall”, inni powinni mieć możliwość oceny jego instrukcji. Dodatkowo posty powinny mieć geolokalizację, która pozwoli na zobaczenie, z jakiego regionu świata jest dane danie. Dodatkowo, po wpisaniu w wyszukiwarkę listy dostępnych składników u siebie w lodówce, powinien się pojawić przepis, który można wykonać.

Rysunek 1: ekran logowania



Rysunek 2: poglądowy schemat widoku użytkownika po zalogowaniu



Pytania, które pojawiły się w dyskusji w celu uszczegółowienia projektu

Pytania	Odpowiedzi	Uwagi
Pytanie produktowe nt. dostępności strony	Strona powinna być dostępna 24 godziny 7 dni w tygodniu.	Może być niedostępna z powodu prac wdrożeniowych.
Jak długo trzeba czekać na załadowanie strony?	Strona powinna ładować się w czasie poniżej 5 sekund na połączeniach o przeciętnej prędkości.	Czas ładowania podczas pierwszej wizyty na stronie nie jest brany pod uwagę.
Pytanie organizacyjne nt. Jak wygląda autoryzacja oraz konto użytkownika po zalogowaniu?	Użytkownik dokonuje autoryzacji swoim kontem Google i posługuje się nim przy wstawianiu przepisów oraz komentowaniu.	Sesja jest utrzymywana za pomocą tokenów JWT. Które są przechowywane w ciasteczku typu HTTP-only, aby zapobiec atakom XSS.
Czy udostępnianie lokalizacji będzie obowiązkowe?	Nie	Jeśli ktoś nie chce udostępniać lokalizacji, to nie musi jej podawać lub może wpisać miasto, do którego mu najbliżej.
Czy korzystanie z witryny będzie płatne?	Nie	-
Czy przepisy będzie można eksportować do plików PDF?	Nie	Taka funkcjonalność jest przewidziana w dalszych fazach rozwijania projektu.

Wymagania funkcjonalne:

Użytkownicy mogą tworzyć konto na platformie, powiązując w ten sposób swój profil z przepisami, które chcieliby zapisać. Rejestracja będzie polegała na wykorzystaniu autoryzacji Google OAuth2, nazwą konta będzie więc imie i nazwisko użytkownika. Zalogowani użytkownicy będą mieli możliwość dodawania oraz edycji przepisów, gdzie przepis będzie składał się z takich danych jak sposób przygotowania, lista składników, czas przygotowania, kategorie, lokalizacja, dane autora. System nie pozwoli na wprowadzenie nieprawidłowych danych, takich jak pusta lista składników. Nie pozwoli

też na wprowadzenie zbyt długich danych, aby chronić się przed próbami nadużycia. Zalogowani użytkownicy będą mieli możliwość wyszukiwania przepisów, a także filtrowania wyników po kategoriach, składnikach itd. Każdy zalogowany użytkownik będzie mógł przechowywać w oddzielnej zakładce prywatnie swoje przepisy. Dodatkowo, system będzie świadczył możliwość dodawania lokalizacji przepisów.

Szczegóły implementacyjne

1. Lista punktów końcowych REST API

Metoda	Endpoint	Auth*	Dane wejściowe	Dane wyjściowe***	Kody HTTP	Opis
POST	/auth/login	Nie	{ credential }	{ token, currentUserObject }	200, 201, 400, 401, 403	logowanie, tworzenie konta
POST	/auth/refresh	Tak**	brak	{ token }	200, 401	odświeża access token JWT
GET	/auth/logout	Tak	brak	brak	200, 401	wylogowywanie
GET	/user	Tak	brak	{ user }	200, 401	zwraca dane aktualnie zalogowanego użytkownika
GET	/user/:id	Tak	brak	{ user }	200, 400, 401, 404	zwraca dane użytkownika z danym ID
GET	/recipe/:id	Tak	brak	{ recipe }	200, 400, 401, 404	zwraca przepis, liczbę recenzji przepisu oraz średnią liczbą gwiazdek na podstawie ID przepisu
GET	/recipe $	Tak	query params:	{ page, limit,	200,	zwraca przepisy

			- limit?? (liczba przepisów na stronę), - page?? (numer strony), - excludeMyRecipes? (czy odpowiedź ma zawierać przepisy aktualnie zalogowanego użytkownika)	totalRecipes, totalPages, recipes }	400, 401, 404	(od najnowszych) wraz z ich autorami, kategoriami, składnikami, liczbą recenzji przepisu oraz średnią liczbą gwiazdek
POST	/recipe	Tak	{ title, cookingTimeMinutes, description, isPublic?, location?, ingredients, categories }	{ recipe }	201, 400, 401, 500	tworzy przepis
PUT	/recipe/:id	Tak	{ title, cookingTimeMinutes, description, isPublic?, longitude?, ingredients, categories }	{ recipe }	200, 400, 401, 500	edytuje przepis
DELETE	/recipe/:id	Tak	brak	brak	200, 400, 401, 500	usuwa przepis
GET	/recipe $	Tak	query params: - userId, - limit?? (liczba przepisów na stronę), - page?? (numer strony), - includePublic? (czy odpowiedź ma zawierać publiczne przepisy), - includePrivate? (czy odpowiedź ma zawierać prywatne przepisy)	{ page, limit, totalRecipes, totalPages, recipes }	200, 400, 401, 404	zwraca przepisy użytkownika z danym ID (od najnowszych) wraz z ich autorem, kategoriami, składnikami, liczbą recenzji przepisu oraz średnią liczbą gwiazdek
POST	/recipe/review/:recipeId	Tak	{ stars, comment? }	{ review }	201, 400, 401,	tworzy recenzję przepisu

					500	
GET	/recipe/review/:recipeId $	Tak	query params: - limit?? (liczba przepisów na stronę), - page?? (numer strony)	{ page, limit, totalReviews, totalPages, currentUserReview, reviews }	200, 400, 401, 404	zwraca recenzje danego przepisu (wraz z ich autorami) na podstawie ID przepisu (od najnowszych)
PUT	/recipe/review/:reviewId	Tak	{ stars, comment? }	{ review }	200, 400, 401, 500	edytuje recenzję przepisu
DELETE	/recipe/review/:reviewId	Tak	brak	brak	200, 400, 401, 500	usuwa recencję przepisu
GET	/recipe/review/stars/:recipeId	Tak	brak	{ count, average }	200, 400, 401, 404	zwraca liczbę recenzji przepisu oraz średnią liczbą gwiazdek
GET	/recipe/ingredient/:name $	Tak	query params: - limit?? (liczba przepisów na stronę), - page?? (numer strony), - excludeMyRecipes? (czy odpowiedź ma zawierać przepisy aktualnie zalogowanego użytkownika)	{ page, limit, totalRecipes, totalPages, recipes }	200, 400, 401, 404	zwraca przepisy zawierające składnik o danej nazwie (od najnowszych) wraz z ich autorem, kategoriami, składnikami, liczbą recenzji przepisu oraz średnią liczbą gwiazdek
GET	/recipe/category/:name $	Tak	query params: - limit?? (liczba przepisów na stronę), - page?? (numer strony), - excludeMyRecipes? (czy odpowiedź ma zawierać przepisy	{ page, limit, totalRecipes, totalPages, recipes }	200, 400, 401, 404	zwraca przepisy z danej kategorii (od najnowszych) wraz z ich autorem, kategoriami, składnikami, liczbą recenzji

			aktualnie zalogowanego użytkownika)			przepisu oraz średnią liczbą gwiazdek
GET	/recipe/ingredients/:commaSeparatedNames $	Tak	query params: - limit?? (liczba przepisów na stronę), - page?? (numer strony), - excludeMyRecipes? (czy odpowiedź ma zawierać przepisy aktualnie zalogowanego użytkownika)	{ page, limit, totalRecipes, totalPages, recipes }	200, 400, 401, 404	zwraca przepisy z danymi kategoriami - oddzielonymi przecinkami w URL - (od najnowszych) wraz z ich autorem, kategoriami, składnikami, liczbą recenzji przepisu oraz średnią liczbą gwiazdek
GET	/recipe/categories/:commaSeparatedNames $	Tak	query params: - limit?? (liczba przepisów na stronę), - page?? (numer strony), - excludeMyRecipes? (czy odpowiedź ma zawierać przepisy aktualnie zalogowanego użytkownika)	{ page, limit, totalRecipes, totalPages, recipes }	200, 400, 401, 404	zwraca przepisy z danymi składnikami - oddzielonymi za pomocą przecinków w URL - (od najnowszych) wraz z ich autorem, kategoriami, składnikami, liczbą recenzji przepisu oraz średnią liczbą gwiazdek

* czy punkt końcowy wymaga autoryzacji (access token JWT w nagłówku HTTP)

** autoryzacja poprzez refresh token JWT w ciasteczku typu HTTP-only

*** każdy punkt końcowy zwraca dane w formacie { success, message, data }, gdzie w przypadku powodzenia operacji pole ‘data’ to dane wyjściowe, a w przypadku niepowodzenia (lub gdy odpowiedź nie zwraca danych) pole ‘data’ jest *undefined*

?? parametr (pół)opcjonalny - jeden z kilku parametrów tego typu musi być dostarczony

? parametr opcjonalny

$ endpoint obsługuje paginację

2. Tabela kodów HTTP oraz ich znaczenia w kontekście aplikacji

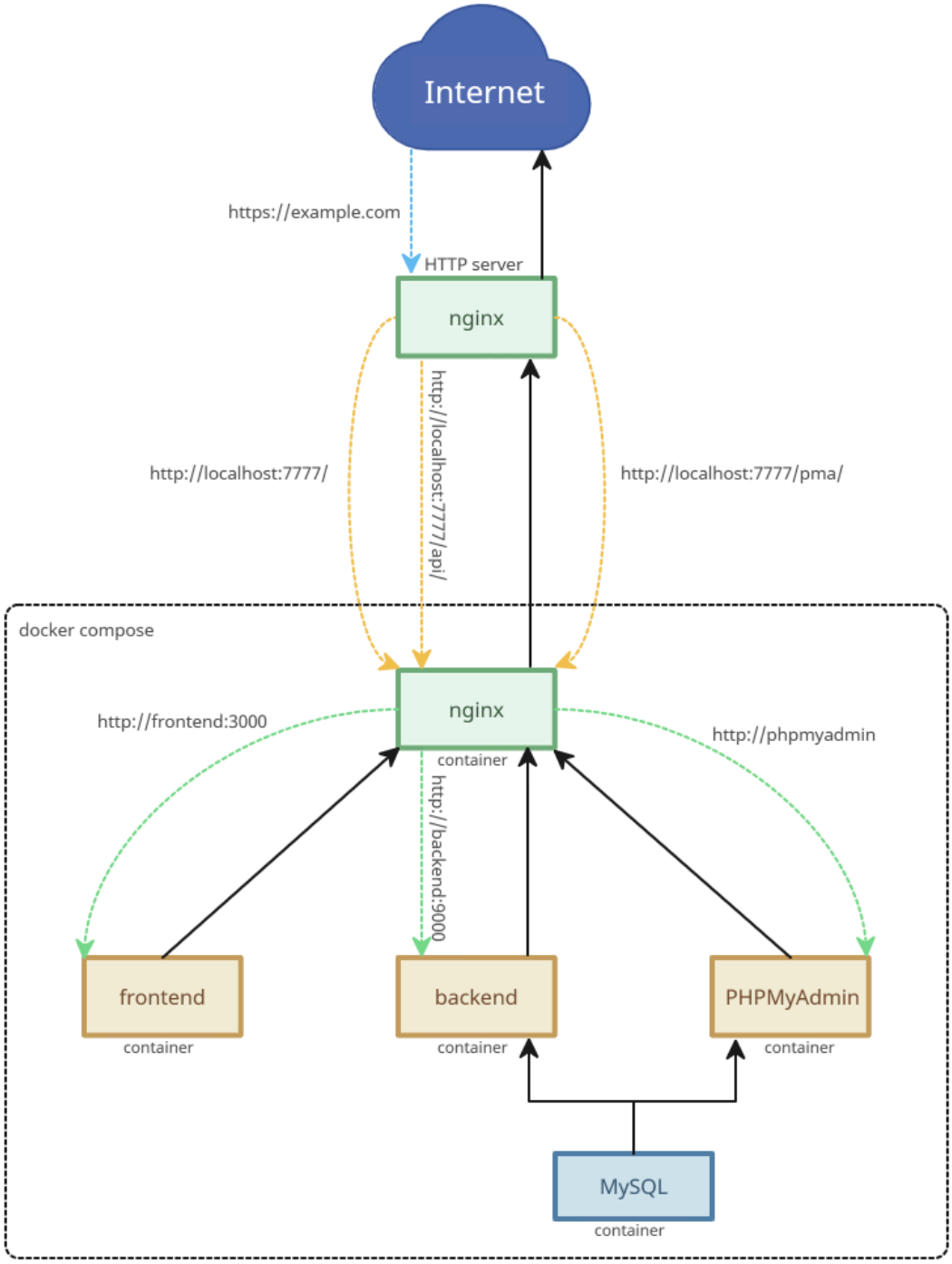
Kod	**Znaczenie**
200	Operacja zakończona powodzeniem.
201	Zasób został utworzony pomyślnie.
400	Niekompletne lub niepoprawne dane wejściowe.
401	Brak uwierzytelnienia.
403	Brak autoryzacji.
404	Zasób nie został znaleziony.
409	Konflikt.
500	Błąd serwera.
502	Błąd serwera proxy.
503	Aplikacja niedostępna.

3. Baza danych

   Jako baza danych zostanie wykorzystana relacyjna baza MySQL działająca w kontenerze. Zarządzanie nią będzie się odbywało za pomocą PHPMyAdmin.
   Jako ORM w Node.js zostanie wykorzystana Prisma.
   Do geolokalizacji zostanie wykorzystane publiczne API https://geocode.maps.co/.

4. Wymagania dotyczące frontendu:
   - Napisany w React.js z użyciem TypeScript.
   - Do budowy UI zostaną użyte gotowe komponenty z biblioteki *Material UI*.
   - Własne komponenty powinny być reużywalne.
   - Zapytania do REST API wykonywane są za pomocą biblioteki *axios.*

5. Projekt architektury w formie UML



MVP (Minimum Viable Product)

Minimum Viable Product AFK CookBook obejmuje działające logowanie się do systemu, posiadanie własnego konta na stronie i możliwość dodawania oraz przeglądania przepisów. Funkcjonalność ta powinna zostać zaimplementowana i dostarczona do klienta maksymalnie do terminu 22.12.2023r. To MVP obejmuje podstawowe funkcje, dzięki któremu będziemy w stanie dostarczyć funkcjonalny produkt w wyznaczonym terminie, pozwalając na dalszy rozwój i dodawanie bardziej zaawansowanych funkcji w kolejnych fazach projektu.

Ewolucja systemu

Ewolucja AFK CookBook obejmuje stopniowy rozwój funkcji oraz usprawnień, zapewniając użytkownikom nowe możliwości i doskonaląc doświadczenie korzystania z aplikacji (UX).

W pierwszej fazie, przez pierwsze dwa miesiące, skupimy się na wdrożeniu podstawowych funkcji, takich jak logowanie przez Google OAuth2, dodawanie, przeglądanie i edytowanie przepisów, oraz ich ocen i komentarzy. Dodatkowo, wprowadzimy funkcję wyszukiwania przepisu po dostępnych składnikach.

W dalszych etapach skoncentrujemy się na rozbudowie społeczności, dodając możliwość eksportu przepisu do formatu PDF, system powiadomień o nowych przepisach i interaktywne rankingi na podstawie ocen.

Chcemy, aby Cookbook stał się miejscem, gdzie pasjonaci kuchni mogą wzajemnie się inspirować i dzielić swoimi doświadczeniami. Skoncentrujemy się na optymalizacji wydajności, przeprowadzając testy bezpieczeństwa i wprowadzając poprawki. Dokumentacja rozwoju będzie stale aktualizowana, aby zapewnić klarowność i zrozumienie dla użytkowników oraz deweloperów.