Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Paweł Giżka
Nr albumu: 385522

Szymon Gajda
Nr albumu: 385477

Kamil Ćwintal
Nr albumu: 385369

Tomasz Kanas
Nr albumu: 385674

Aplikacja mobilna "Podziel się"

Praca licencjacka
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dra Janusza Jabłonowskiego
Instytut Informatyki

Czerwiec 2019

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpisy autorów pracy

Streszczenie

"Podziel się" to aplikacja mobilna przeznaczona do wzajemnego dzielenia się żywnością dla użytkowników systemów operacyjnych Android oraz iOS. Celem projektu było stworzenie platformy, która umożliwi łatwe i bezpłatne dzielenie się jedzeniem z innymi mieszkańcami w okolicy: osoby posiadające nadwyżkę artykułów żywnościowych lub gotowych potraw mogą przekazać je użytkownikom, którzy zadeklarują chęć ich odbioru. W pracy przedstawiono szczegółowy opis aplikacji oraz całościowej architektury projektu, a także przybliżono proces jego powstawania.

Słowa kluczowe

informatyka, aplikacja mobilna, żywność, foodsharing, React Native

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

Software and its engineering
Applied computing → Law, social and behavioral sciences → Sociology

Spis treści

Wprowadzenie 5

1. Podstawowe definicje 9

2. Przegląd konkurencyjnych rozwiązań 11
2.1. *Olio* 11
2.2. *Too Good To Go* 11
2.3. Grupa "*Freeganism, dumpster diving, food sharing — Warszawa*" 12

3. Implementacja 13
3.1. Architektura systemu 13
3.1.1. Aplikacja mobilna 13
3.1.2. Część serwerowa 15
3.1.3. Komunikacja klient - serwer 15
3.2. PostGIS 15
3.3. Przechowywanie zdjęć 16
3.4. Powiadomienia 16
3.5. Logowanie za pomocą Facebooka 17
3.6. Działanie aplikacji mobilnej na dwóch systemach operacyjnych 17

4. Infrastruktura serwera 23
4.1. Kryteria 23
4.2. Amazon Lightsail 23
4.3. Oprogramowanie serwera 23
4.4. Dostęp do serwera 24
4.5. Dalszy rozwój 24

5. Przypadki użycia 25
5.1. Rejestracja 25
5.1.1. Krótki opis 25
5.1.2. Cele 25
5.1.3. Warunki wstępne 25
5.1.4. Czynności 25
5.2. Dodawanie nowego ogłoszenia 28
5.2.1. Krótki opis 28
5.2.2. Cele 28
5.2.3. Warunki wstępne 28
5.2.4. Czynności 28
5.2.5. Możliwe rozwinięcia 29

5.3. Konwersacja (czat) z innym użytkownikiem . . . 30
5.3.1. Krótki opis . . . 30
5.3.2. Cele . . . 30
5.3.3. Warunki wstępne . . . 30
5.3.4. Czynności . . . 31
5.3.5. Możliwe rozwinięcia . . . 31
5.4. Ocena innego użytkownika . . . 32
5.4.1. Krótki opis . . . 32
5.4.2. Cele . . . 32
5.4.3. Warunki wstępne . . . 32
5.4.4. Czynności . . . 33
5.4.5. Możliwe rozwinięcia . . . 33
5.5. Odbiór produktów oferowanych na liście ogłoszeń . . . 34
5.5.1. Krótki opis . . . 34
5.5.2. Cele . . . 34
5.5.3. Warunki wstępne . . . 34
5.5.4. Czynności . . . 35

6. Zarządzanie projektem . . . 37
6.1. Wykorzystane narzędzia . . . 37
6.2. Współpraca z zamawiającą . . . 37
6.3. Organizacja pracy . . . 37
6.4. Podział obowiązków . . . 38
6.4.1. Kamil Ćwintal . . . 38
6.4.2. Szymon Gajda . . . 38
6.4.3. Paweł Giżka . . . 38
6.4.4. Tomasz Kanas . . . 38

7. Zawartość płyty . . . 39

8. Podsumowanie . . . 41

Bibliografia . . . 43

Wprowadzenie

Kontekst społeczny

Według współczesnych szacunków ([FAO]), około 1/3 wytworzonej na świecie żywności nigdy nie zostanie zjedzona. Skala problemu jest szczególnie widoczna w danych Eurostatu ([Eurostat]), z których wynika, że w Polsce rocznie marnuje się około 9 milionów ton żywności. Jej wartość sięga około 50 złotych miesięcznie w przeliczeniu na jednego mieszkańca Polski. Badanie [Kantar Millward Brown] wykazało, że do wyrzucania jedzenia przyznaje się 42% Polaków, podczas gdy produkty te mogłyby zaspokoić podstawowe potrzeby żywieniowe niemal 2 mln osób żyjących w tzw. skrajnym ubóstwie ([Główny Urząd Statystyczny]). Nasz kraj jest piątym wśród państw Unii Europejskiej, w których marnuje się najwięcej żywności.

W krajach rozwiniętych około 50% wyrzucanego jedzenia pochodzi z gospodarstw domowych, w których na skutek nierozważnego planowania zakupów jedzenie ulega zepsuciu i trafia do śmietnika ([Polityka]). Wierzymy, że możliwe jest przeciwdziałanie temu zjawisku na poziomie lokalnym.

Istnieją ruchy społeczne (dumpster diving, freeganizm, zero waste) promujące możliwie racjonalny obrót żywnością oraz uwrażliwiające społeczeństwo na problem masowego wyrzucania jedzenia.

Problem do rozwiązania

Obecnie w Polsce istnieją liczne grupy entuzjastów wymiany żywności funkcjonujące na portalach społecznościowych (przeważnie na Facebooku), jednak duża liczebność tych grup uniemożliwia skuteczną organizację wymiany jedzenia w satysfakcjonujący sposób. Nasza aplikacja *"Podziel się"* powstała, aby wspierać dalsze funkcjonowanie tej inicjatywy oraz zapewnić zwolennikom idei food sharingu platformę do efektywnej wymiany żywności, także w znacznie większej skali.

Zamawiający

Aplikacja *"Podziel się"* powstała na zamówienie pani Marii Skołożyńskiej, działaczki społecznej oraz pomysłodawczyni akcji *Podziel się Posiłkiem z Bezdomnymi* ([podzielmysie.pl]), podczas której mieszkańcy całej Polski mogą przekazać pozostałe po świętach (Bożego Narodzenia, Wielkanocy) posiłki do wolontariuszy, którzy zajmują się transportem jedzenia do osób potrzebujących. Pani Maria jest aktywistką działającą na rzecz przeciwdziałania marnowaniu jedzenia oraz główną koordynatorką akcji dystrybucji żywności. Działalność pani Marii Skołożyńskiej ma zasięg ogólnopolski; akcje gromadzenia żywności dla bezdomnych i ubogich działają obecnie w kilkudziesięciu miastach, m. in. w Warszawie, Krakowie, Wrocławiu, Trójmieście, Szczecinie, Toruniu, Bydgoszczy, Katowicach, Łodzi. W 2018 r. akcja *Podziel się Po-*

siłkiem z Bezdomnymi, w uznaniu dla skuteczności działania oraz promocji pozytywnych postaw społecznych, otrzymała Europejską Nagrodę Reduce Food Waste ([reducefoodwaste.eu]).

Osoby zaangażowane w projekt

Aplikacja *"Podziel się"* została opracowana oraz wdrożona przez zespół studentów informatyki Uniwersytetu Warszawskiego: Kamila Ćwintala, Szymona Gajdę, Pawła Giżkę oraz Tomasza Kanasa. Opiekę nad projektem sprawował dr Janusz Jabłonowski. Wymagania funkcjonalne zostały dostarczone zespołowi przez pomysłodawczynie projektu: p. Marię Skołożyńską oraz p. Martynę Dakowską.

Grupa docelowa

Aplikacja *"Podziel się"* jest przeznaczona dla użytkowników smartfonów, którym nie jest obojętny problem marnowania żywności. *"Podziel się"* może stanowić interesującą propozycję dla osób, które nie mają czasu na przygotowanie posiłku albo gotują zbyt dużo (i jednocześnie nie chcą wyrzucać nadwyżki jedzenia). Aplikacja *"Podziel się"* będzie docelowo zrzeszać społeczność osób zainteresowanych tematyką food sharingu i zero waste, które dbają o środowisko naturalne oraz racjonalną gospodarkę żywnością.

Opis funkcjonalności

Aplikacja *"Podziel się"* umożliwia:

- bezpłatną rejestrację nowego konta,
- uzupełnienie własnego profilu o podstawowe dane personalne,
- dodawanie nowego ogłoszenia (opisu produktu wraz z preferowaną lokalizacją miejsca odbioru),
- zamieszczenie w ogłoszeniu zdjęć produktu, opakowania, etykiet itp.,
- przeglądanie dostępnych do odebrania posiłków w okolicy aktualnej lokalizacji użytkownika,
- sortowanie dostępnych ogłoszeń w oparciu o ustalone kryteria (odległość, data wystawienia produktu, data przydatności itp.),
- funkcję czatu między odbiorcą a darczyńcą w celu ustalenia szczegółów odbioru, zadawania dodatkowych pytań,
- przechowywanie historii konwersacji z innymi użytkownikami,
- wskazanie użytkownika, który otrzyma wystawiony produkt/artykuł żywnościowy,
- pośrednictwo w procesie przekazania,
- system recenzji/ocen użytkowników,
- oznaczanie wybranych użytkowników jako "obserwowanych", dzięki czemu ich nowe ogłoszenia będą dodatkowo eksponowane na liście dostępnych ogłoszeń,
- zgłaszanie ogłoszeń o nieodpowiedniej zawartości, co skutkuje powiadomieniem moderatora o konieczności weryfikacji zamieszczonej treści.

Aplikację wyróżnia nowoczesny interfejs graficzny, zaprojektowany specjalnie na potrzeby projektu.

Zawartość pracy

Rozdział 1 podaje definicje niektórych specjalistycznych terminów użytych w pracy licencjackiej. Rozdział 2 stanowi przegląd oraz analizę konkurencyjnych rozwiązań. W rozdziale 3 przedstawiono całościową architekturę systemu oraz wyzwania i problemy napotkane podczas jego implementacji. Infrastruktura serwera została szczegółowo opisana w rozdziale 4. Rozdział 5 przedstawia najważniejsze przypadki użycia aplikacji. Rozdział 6 poświęcony jest metodyce tworzenia projektu oraz podziałowi pracy w zespole. Końcowe rozdziały zawierają: spis zawartości płyty dołączonej do pracy oraz krótkie podsumowanie projektu.

Rozdział 1

Podstawowe definicje

Poniżej znajdują się objaśnienia podstawowych pojęć i definicji używanych w dalszej części pracy licencjackiej.

- **Framework** — z Wikipedii: "(...) *szkielet do budowy aplikacji. Definiuje on strukturę aplikacji oraz ogólny mechanizm jej działania, a także dostarcza zestaw komponentów i bibliotek ogólnego przeznaczenia do wykonywania określonych zadań*" (patrz [Framework]). W polskiej terminologii informatycznej spotyka się również określenie *zrąb*, jednak w dalszej części pracy będziemy konsekwentnie stosować określenie *framework* jako słowo bardziej rozpoznawalne i rozpowszechnione w codziennej praktyce programistycznej.
- **Amazon S3** — usługa firmy Amazon umożliwiająca przechowywanie danych (pamięć masowa w chmurze).
- **React Native** — framework stworzony przez firmę Facebook, pozwalający na tworzenie aplikacji jednocześnie na systemy Android oraz iOS.
- **Django** — framework do tworzenia aplikacji internetowych.
- **Firebase** — platforma stworzona przez firmę Google, oferuje narzędzia wspierające tworzenie aplikacji mobilnych.
- **Firebase Cloud Messaging (FCM)** — narzędzie platformy Firebase pozwalające na wysyłanie powiadomień do telefonu użytkownika.
- **Apple Push Notification (APN)** — narzędzie firmy Apple umożliwiające wysyłanie natychmiastowych powiadomień do telefonu użytkownika.
- **Gunicorn** — serwer pozwalający na uruchomienie aplikacji napisanej z wykorzystaniem frameworka Django.
- **nginx** — serwer HTTP, wykorzystywany również jako serwer proxy.

Rozdział 2

Przegląd konkurencyjnych rozwiązań

Na przestrzeni ostatnich kilku lat można zaobserwować systematyczny wzrost zainteresowania tematyką marnowania żywności. Przyczyn tego zjawiska należy upatrywać w coraz wyższym poziomie świadomości ekologicznej polskiego społeczeństwa, bardziej intensywnych działaniach władz i rozmaitych instytucji (w tym Komisji Europejskiej - patrz [ec.europa.eu]), a także w licznych publikacjach naukowych na ten temat. Trend "zero waste" przejawia się w bogatej ofercie aplikacji mobilnych pomagających wymieniać się jedzeniem, które inaczej by się zmarnowało. Co ciekawe, w różnych państwach popularność zyskały różne aplikacje. Na przykład w Wielkiej Brytanii najpopularniejsze jest *Olio*, w Europie Zachodniej - *Too Good To Go*, a w Stanach Zjednoczonych - *Unsung*. W Polsce nie ma jeszcze żadnej rozpoznawalnej aplikacji, chociaż na portalach społecznościowych istnieją grupy spełniające podobną funkcję. W dalszej części tego rozdziału zostaną dokładniej omówione niektóre konkurencyjne rozwiązania.

2.1. *Olio*

Spośród wymienionych w tym rozdziale aplikacji *Olio* jest koncepcyjnie najbardziej podobne do *Podziel się*. Aplikacja posiada bardzo prosty interfejs, pozwalający już w kilka chwil po zainstalowaniu dodać swoje pierwsze ogłoszenie, czy też znaleźć na liście ogłoszeń (która może być sortowana po odległości lub czasie dodania ogłoszenia) produkt, na który mamy ochotę. *Olio*, tak samo jak *Podziel się*, jest zintegrowane z Mapami Google oraz wykorzystuje lokalizację telefonu (choć można tę opcję wyłączyć). *Olio* udostępnia też proste forum, które uznaliśmy za zbędne w takiej aplikacji, nie daje natomiast możliwości podziału ogłoszeń na kategorie (np. wegańskie, bezglutenowe itp.) - w *Olio* specyfikuje się tylko, czy dany produkt jest żywnością, czy nie.

Olio jest bardzo popularną aplikacją. Obecnie ma ponad milion użytkowników i ponad półtora miliona ogłoszeń ([olioex.com]). Największą popularność zdobyła w Wielkiej Brytanii, w Polsce praktycznie nie funkcjonuje.

2.2. *Too Good To Go*

Ta aplikacja reprezentuje zupełnie inne podejście niż *Podziel się*. Umożliwia restauracjom i jadłodajniom zbyt niesprzedanych posiłków po znacznie niższych cenach niedługo przed zamknięciem. Oznacza to, że oferty są zwykle płatne - w przeciwieństwie do *Podziel się*, gdzie wszystkie oferowane produkty są darmowe. *Too Good To Go* ma nieco bardziej skomplikowany interfejs niż *Olio* czy *Podziel się*, jest zintegrowana z Mapami Google, wykorzystuje

lokalizację telefonu, umożliwia też przeglądanie mapy z zaznaczonymi wszystkimi dostępnymi jadłodajniami oraz posiada opcję dodawania jadłodajni do listy swoich ulubionych.

Too Good To Go jest popularna w praktycznie całej Europie Zachodniej, posiadając obecnie prawie 25 tysięcy współpracujących jadłodajni ([toogoodtogo.com]). W Warszawie aplikacja jest wykorzystywana przez zaledwie 20 jadłodajni.

2.3. Grupa "*Freeganism, dumpster diving, food sharing — Warszawa*"

Największą konkurencją dla aplikacji mobilnych są grupy funkcjonujące na portalu społecznościowym Facebook, które w założeniu mają służyć do tego samego celu. Tego rodzaju lokalnych grup jest w Polsce bardzo wiele, ta została wybrana jako przykładowa głównie ze względu na jej największą popularność - w chwili obecnej liczy prawie 9 tysięcy członków, a publikowanych jest na niej około 700 postów miesięcznie. Główny problem takich grup wynika z niedostosowania platformy — portalu Facebook. Nie posiada on narzędzi umożliwiających wygodne przeglądanie listy aktualnych ofert, nie pozwala filtrować ani sortować ogłoszeń po odległości, ani nawet łatwo odrzucić już wygasłe ogłoszenia. Bardzo ogranicza to rozrost takich grup. Warto wspomnieć o ciekawym rozwiązaniu, z którego korzysta przedstawiona grupa - jest to koncept jadłodzielni. Są to zazwyczaj ogólnodostępne lodówki i szafki na żywność, w których można umieszczać produkty przeznaczone do oddania. Udostępniane są zwykle przez budynki użyteczności publicznej, takie jak urząd miasta, uniwersytet czy centrum kultury. Jest to szczególnie wygodne w odbiorze, ponieważ nie trzeba nawet wiedzieć, kto dokładnie ma żywność do oddania - wystarczy będąc w pobliżu sprawdzić, czy nie ma w niej akurat czegoś, na co ma się ochotę.

Rozdział 3

Implementacja

Niniejszy rozdział zawiera opis architektury projektu w ujęciu ogólnym, a także szczegóły realizacji poszczególnych komponentów systemu. Przedstawione zostały problemy i wyzwania, które napotkaliśmy w trakcie implementacji kluczowych funkcjonalności. Prezentujemy także sposoby ich rozwiązania.

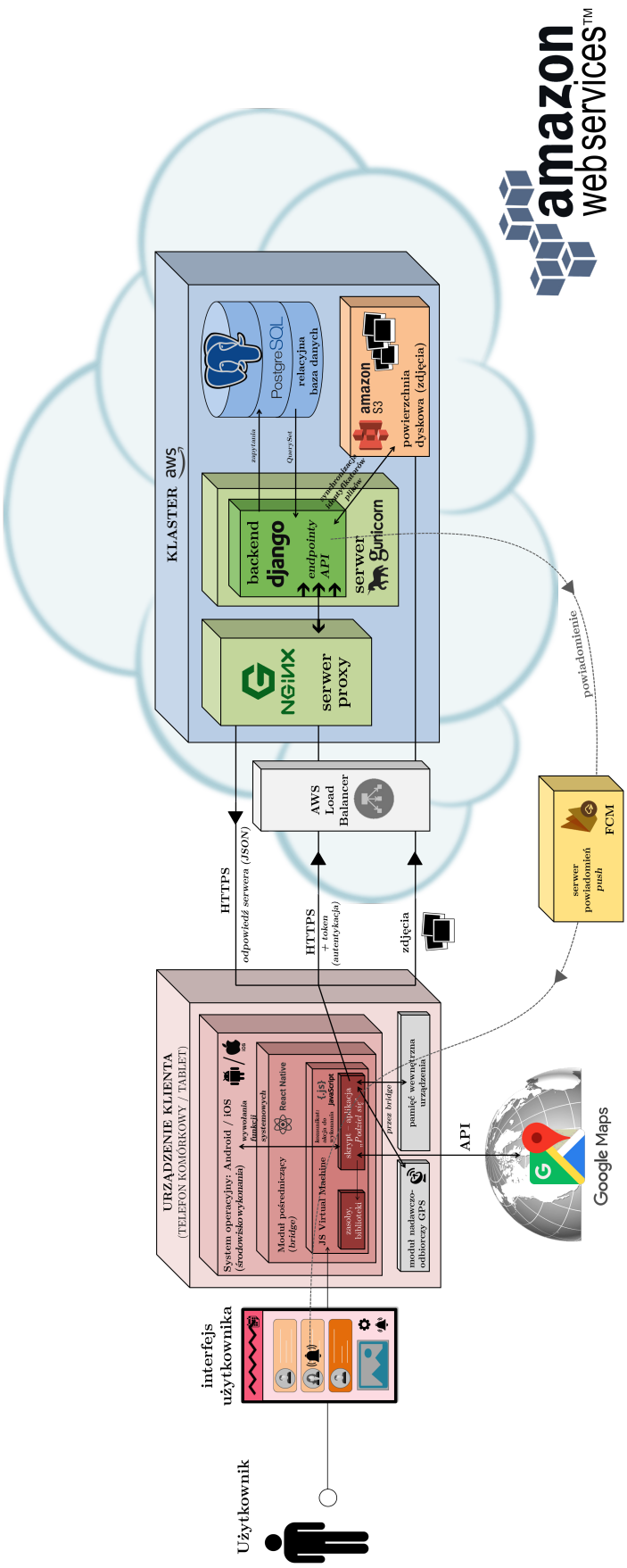
3.1. Architektura systemu

Architektura aplikacji *"Podziel się"* składa się z dwóch głównych komponentów: aplikacji mobilnej oraz części serwerowej. Wykorzystywane są również zewnętrzne usługi, takie jak: Firebase, Facebook, Mapy Google, Amazon S3.

3.1.1. Aplikacja mobilna

Aplikacja *"Podziel się"* przeznaczona jest dla użytkowników dwóch najpopularniejszych mobilnych systemów operacyjnych: Androida oraz iOS. Interfejs aplikacji powstał z wykorzystaniem nowoczesnego frameworka React Native, pozwalającego opisać graficzne rozmieszczenie elementów na ekranie w składni zbliżonej do arkuszy stylów CSS, zaś logikę działania aplikacji w kodzie JavaScriptu. W React Native, przy użyciu specjalnego języka znaczników JSX, zostały zdefiniowane tzw. komponenty (lista ogłoszeń, przycisk wysyłania wiadomości, menu szufladkowe, okienko z komunikatem itp.), na bazie których React Native buduje komponenty natywne dla Androida oraz iOS. Pozwala to na ujednolicenie wyglądu interfejsu graficznego dla użytkowników różnych systemów i architektur — aplikacja nie odbiega wyglądem od rozwiązań natywnych, projektowanych specjalnie z myślą o konkretnej platformie. Filozofia React Native polega na stosowaniu schematów tworzenia aplikacji webowych przy projektowaniu aplikacji działających w środowisku mobilnym, w szczególności możliwe jest korzystanie z rozmaitych pomocniczych bibliotek JavaScriptu.

Ważnym elementem React Native jest moduł pośredniczący (ang. *bridge*), który tłumaczy wywołania pochodzące z wątków JavaScriptu na polecenia skierowane do systemu operacyjnego. Komunikaty opisujące akcje do wykonania są przesyłane w formacie JSON do modułu pośredniczącego, a po ich otrzymaniu *bridge* wywołuje odpowiednie funkcje systemowe. Wymiana komunikatów jest asynchroniczna oraz nieblokująca, zatem React Native nie ingeruje nadmiernie w pracę systemu operacyjnego oraz umożliwia sprawne renderowanie widoków na ekranie. Co więcej, aplikacja ma możliwość swobodnej komunikacji ze środowiskiem wykonania; w razie potrzeby może uzyskać dostęp do modułu GPS, aparatu, pamięci wewnętrznej urządzenia itp.



Rysunek 3.1: Schemat architektury systemu (opracowanie własne)

3.1.2. Część serwerowa

Część serwerowa została stworzona w języku Python, w oparciu o framework Django. Pozwala on na szybkie tworzenie sieciowych interfejsów programistycznych aplikacji (po angielsku Web API) oraz łatwą pracę z relacyjnymi bazami danych dzięki rozbudowanemu mechanizmowi mapowania obiektowo-relacyjnego (ORM). Wspomniany mechanizm polega na odwzorowaniu logicznych modeli encji, reprezentowanych jako klasy w języku Python, na rzeczywistą treść skryptów tworzących tabele i więzy spójności w bazie danych. Podobnie, wyniki zapytań są konwertowane do obiektu typu `QuerySet`, który może podlegać dalszemu przetwarzaniu przez serwer.

Jednym z głównych wymogów zamawiającej było stworzenie panelu administracyjnego. Wybrany przez nas framework umożliwia automatyczne generowanie panelu administratora. Tak powstały panel pozwala na wykonywanie kompletu operacji CRUD: dodawania nowych danych, odczytania istniejących informacji, modyfikowania oraz usuwania wpisów znajdujących się w bazie.

Do przechowywania danych została wykorzystana relacyjna baza danych PostgreSQL z wtyczką PostGIS, która pozwala na wydajne wykonywanie zapytań przestrzennych. Zdjęcia przechowywane są w zewnętrznej usłudze Amazon S3.

3.1.3. Komunikacja klient - serwer

Wymiana danych pomiędzy aplikacją mobilną a serwerem odbywa się za pomocą bezstanowego protokołu HTTP. Dane przesyłane są w formacie JSON, jest on formatem tekstowym i bazuje na podzbiorze języka JavaScript. Każde zapytanie do serwera uwierzytelniane jest unikatowym tokenem przyznawanym w momencie logowania bądź rejestracji, w związku z tym hasło użytkownika nie jest przechowywane w pamięci telefonu. Przy każdym ponownym logowaniu generowany jest nowy token.

Dane z serwera do urządzenia mobilnego (powiadomienia) wysyłane są natomiast za pośrednictwem usługi Firebase.

3.2. PostGIS

Jedną z funkcjonalności naszej aplikacji jest prezentowanie użytkownikowi ofert z adresem odbioru położonym w pobliżu jego aktualnej lokalizacji. Aby to osiągnąć, musielibyśmy efektywnie wyznaczać po stronie serwera wszystkie ogłoszenia z bazy danych, które są oddalone od zadanej lokalizacji o nie więcej niż pewną wartość otrzymaną w zapytaniu użytkownika. Pierwszym podejściem do rozwiązania tego problemu było wyznaczanie dla każdego zapytania wszystkich odległości pomiędzy adresem odbioru ogłoszeń a otrzymaną lokalizacją, a następnie wybranie tych, które są położone bliżej niż maksymalna odległość wskazana przez użytkownika. Byłoby to jednak rozwiązanie skrajnie nieefektywne, ponieważ dla każdego zapytania wymaga przejrzenia wszystkich ofert z bazy, co przy większej liczbie użytkowników i ofert spowodowałoby poważne problemy z wydajnością całego systemu.

Rozwiązaniem, na które ostatecznie się zdecydowaliśmy, jest skorzystanie z rozszerzenia bazy danych PostgreSQL o nazwie PostGIS. Pozwala ono na wprowadzanie danych geograficznych oraz ich odpowiednią obsługę. Dla każdego ogłoszenia w bazie danych zapisujemy współrzędne geograficzne miejsca odbioru, dzięki czemu odpowiednia funkcja rozszerzenia PostGIS pozwala na znalezienie wszystkich ofert w ustalonej odległości od lokalizacji użytkownika. Operacja ta jest efektywna dlatego, że wyżej wspomniane rozszerzenie zmienia sposób przechowywania danych geograficznych w porównaniu do standardowej bazy danych.

Zamiast B-drzewa wykorzystuje R-drzewo, które jest strukturą danych wspomagającą wyszukiwanie obiektów w przestrzeni wielowymiarowej. Dla rzeczywistych danych to rozwiązanie jest dużo szybsze od podejścia naiwnego przedstawionego wcześniej.

3.3. Przechowywanie zdjęć

W trakcie tworzenia systemu musieliśmy wybrać sposób przechowywania zdjęć. Pierwszym narzucającym się pomysłem było zapisywanie zdjęć na lokalnym dysku serwera. Szybko jednak zdaliśmy sobie sprawę, że takie rozwiązanie posiada szereg wad. Pierwszą z nich jest brak elastyczności — w przypadku wyczerpania miejsca bylibyśmy zmuszeni wykupić droższy serwer, nawet jeżeli nie potrzebowalibyśmy szybszego procesora czy większej pamięci RAM. Kolejną wadą jest duże obciążenie serwera, który musiałby obsłużyć zapytania HTTP związane z transferem tych zdjęć na urządzenia mobilne.

Ostatecznie zdecydowaliśmy się skorzystać z serwisu Amazon Simple Storage Service (Amazon S3). Usługa pozwala przechowywać dane w sposób trwały i bezpieczny. Zaletą Amazon S3 jest nieograniczona pojemność dyskowa oraz brak narzuconych limitów na transfer wychodzący. Co więcej, koszt usługi jest proporcjonalny do sumarycznego rozmiaru przechowywanych danych, mamy więc pełną elastyczność jeśli chodzi o koszty. Transfer plików jest szyfrowany protokołem SSL, ponadto same dane podlegają 256-bitowemu szyfrowaniu po stronie serwera.

Obsługa zdjęć zrealizowana jest w następujący sposób: część mobilna kompresuje fotografie przed wysłaniem, następnie przesyła je w jednym zapytaniu wraz z informacjami o ogłoszeniu do serwera aplikacji *"Podziel się"*. Serwer zapisuje dane ogłoszenia i metadane zdjęć w bazie PostgreSQL, a następnie wysyła obrazy do usługi Amazon S3. Przy pobieraniu informacji o dostępnych ogłoszeniach lub szczegółach danego ogłoszenia serwer zwraca odnośniki do zdjęć, następnie urządzenie mobilne pozyskuje dane bezpośrednio z serwisu Amazon S3. Dzięki temu w znacznym stopniu redukowane jest obciążenie serwera aplikacji *"Podziel się"*.

W dalszej perspektywie, w miarę rozwoju projektu, Amazon S3 oferuje rozszerzone plany wykorzystania oraz opcję archiwizacji danych z możliwością łatwego odtworzenia w przypadku nadpisania, usunięcia lub utracenia plików.

3.4. Powiadomienia

Aplikacja *"Podziel się"* wspiera użytkowników w zarządzaniu licznymi ogłoszeniami wykorzystując mechanizm powiadomień. Wspomniana funkcjonalność pozwala na wyświetlanie użytkownikom krótkich komunikatów o zajściu zdarzeń wymagających uwagi. Powiadomienia wyświetlane są w przypadku zajścia jednego z następujących zdarzeń:

- otrzymania wiadomości od innego użytkownika,
- pojawienia się nowego ogłoszenia w okolicy,
- dodania nowego ogłoszenia przez obserwowaną osobę,
- polubienia naszego ogłoszenia.

Najprostszym sposobem implementacji tej funkcjonalności jest regularne odpytywanie serwera, czy są dostępne nowe powiadomienia. Jednakże ze względu na ograniczenia systemów Android oraz iOS obsługa procesów w tle (to znaczy wtedy, gdy użytkownik nie korzysta z aplikacji) jest bardzo mocno ograniczona. Procesy w tle nie mogą być uruchamiane częściej niż co kilkadziesiąt minut — związane jest to z oszczędzaniem energii.

Z tego powodu zdecydowaliśmy się na użycie zewnętrznych usług, takich jak: Firebase Cloud Messaging (FCM) dla systemu Android oraz dedykowanego Apple Push Notification (APN) w przypadku iOS. Umożliwiają one niemalże natychmiastowe dostarczanie wiadomości z serwera do telefonu użytkownika.

Schemat działania systemu powiadomień wygląda następująco: urządzenie mobilne przy pierwszym uruchomieniu pobiera unikatowy identyfikator (token) z serwisu FCM bądź APN, dalej jest on wysyłany na serwer aplikacji *"Podziel się"* w trakcie logowania lub rejestracji, a następnie zapisywany w bazie danych. Jeżeli zajdzie potrzeba dostarczenia powiadomienia na dane urządzenie, serwer aplikacji *"Podziel się"* wyśle do serwera FCM wiadomość zawierającą token danego telefonu oraz zawartość powiadomienia. Usługa FCM przekieruje notyfikację do konkretnego użytkownika na podstawie danego identyfikatora urządzenia. Zachowanie telefonu lub tabletu po odebraniu nowego powiadomienia zależy od preferencji użytkownika, może to być okienko z wiadomością, sygnał dźwiękowy lub ikonka na ekranie.

Udało nam się zaimplementować w pełni funkcjonalne rozwiązanie na platformie Android. Jeśli chodzi o system iOS, to wymogiem usługi APN było posiadanie konta w programie *Apple Developer Program*. Niestety, ze względów czasowych i formalnych nie byliśmy w stanie przystąpić do tego programu. Jednakże zarówno aplikacja mobilna, jak i część serwerowa są już przystosowane do obsługi APN — wystarczy jedynie odpowiednio skonfigurować platformę Firebase (wgrać odpowiednie certyfikaty wygenerowane na koncie dewelopera Apple), wtedy powiadomienia do użytkowników systemu firmy Apple będą przekierowywane do APN.

3.5. Logowanie za pomocą Facebooka

Oprócz klasycznej możliwości logowania poprzez podanie adresu e-mail i hasła, istnieje także opcja zalogowania się przez powiązanie swojego konta w serwisie Facebook z kontem w aplikacji *Podziel się*. Cały proces z punktu widzenia użytkownika polega na naciśnięciu przycisku logowania z Facebookiem na ekranie powitalnym, udzieleniu odpowiednich zezwoleń oraz zalogowaniu się na swoje konto w serwisie Facebook. Po wykonaniu tych czynności użytkownik może już korzystać z aplikacji, co więcej, jego dane i zdjęcie profilowe zostaną automatycznie pobrane. Taka możliwość pozwala na jeszcze szybsze skorzystanie z aplikacji *Podziel się* niż klasyczne logowanie.

Funkcjonalność ta jest możliwa dzięki modułowi integrującemu konto w serwisie Facebook z aplikacją napisaną z użyciem React Native. Po udzieleniu potrzebnych zezwoleń i zalogowaniu użytkownika na konto Facebook, aplikacja *Podziel się* pobiera klucz pozwalający na identyfikację użytkownika. Następnie uzyskany klucz jest przesyłany do serwera, gdzie jest weryfikowany i na jego podstawie (a także osobnego klucza specyficznego dla całej aplikacji) pobierane są dane użytkownika, takie jak: imię, nazwisko, zdjęcie profilowe, identyfikator. Kolejnym krokiem jest sprawdzenie, czy ten użytkownik posiada już konto w naszej aplikacji; jeżeli nie, to jego dane są dodawane do bazy, a następnie do urządzenia mobilnego wysyłany jest token uwierzytelniający — analogicznie jak przy klasycznym logowaniu.

3.6. Działanie aplikacji mobilnej na dwóch systemach operacyjnych

System operacyjny Android jest niezwykle popularny w naszym kraju. Według IDC 91,5% sprzedanych smartfonów w 2017 roku stanowiły urządzenia działające pod kontrolą systemu firmy Google (patrz [IDC]).

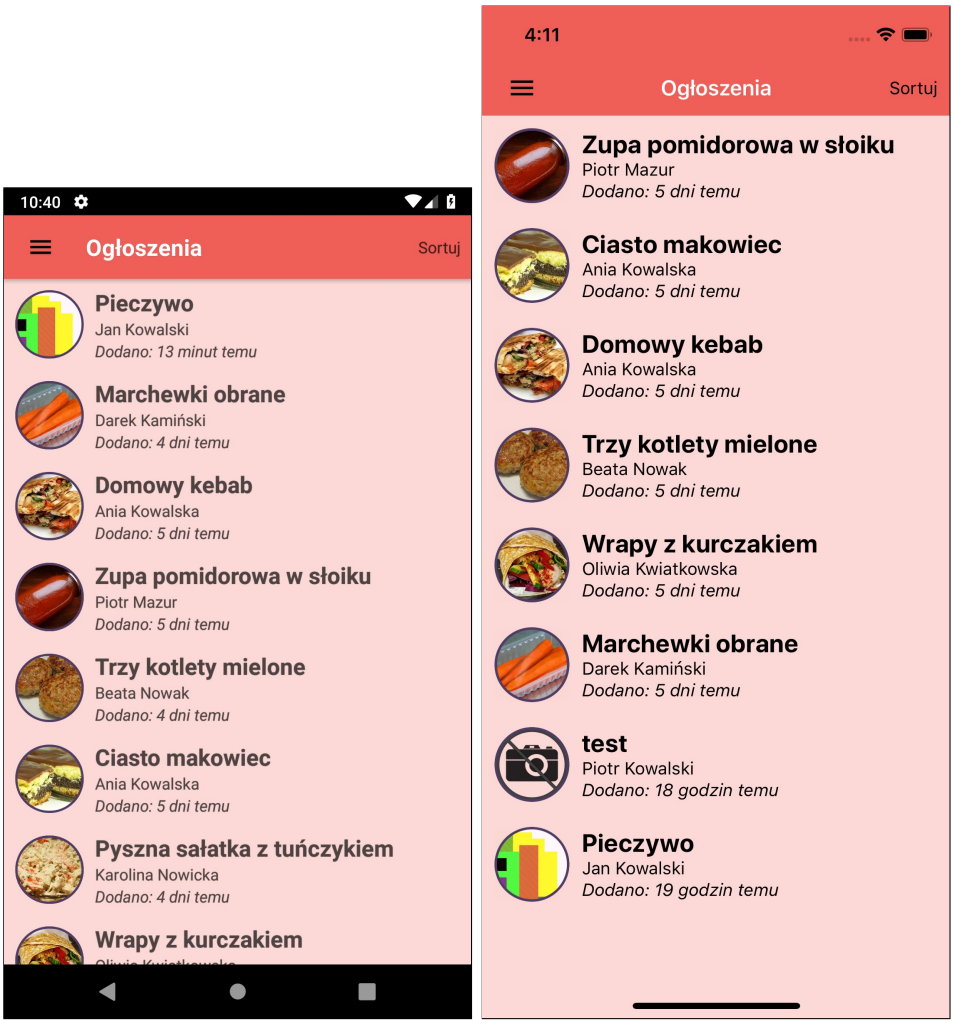
Z tego powodu postanowiliśmy, że w pierwszej kolejności stworzymy aplikację na platformę Android, a dopiero później zajmiemy się systemem iOS. W trakcie pracy nad wersją androidową używaliśmy tylko i wyłącznie bibliotek, których twórcy wyraźnie deklarowali zgodność z obydwoma systemami. Pozwoliło nam to dosyć bezboleśnie uruchomić aplikację na symulatorze iPhone.

Nie udało nam się jedynie zaimplementować dwóch funkcjonalności:

- Pierwszą z nich jest logowanie z Facebookiem. Pomimo zapewnień twórców biblioteki o bezproblemowym działaniu na platformie iOS nie udało nam się odpowiednio jej skonfigurować do działania z naszą aplikacją.

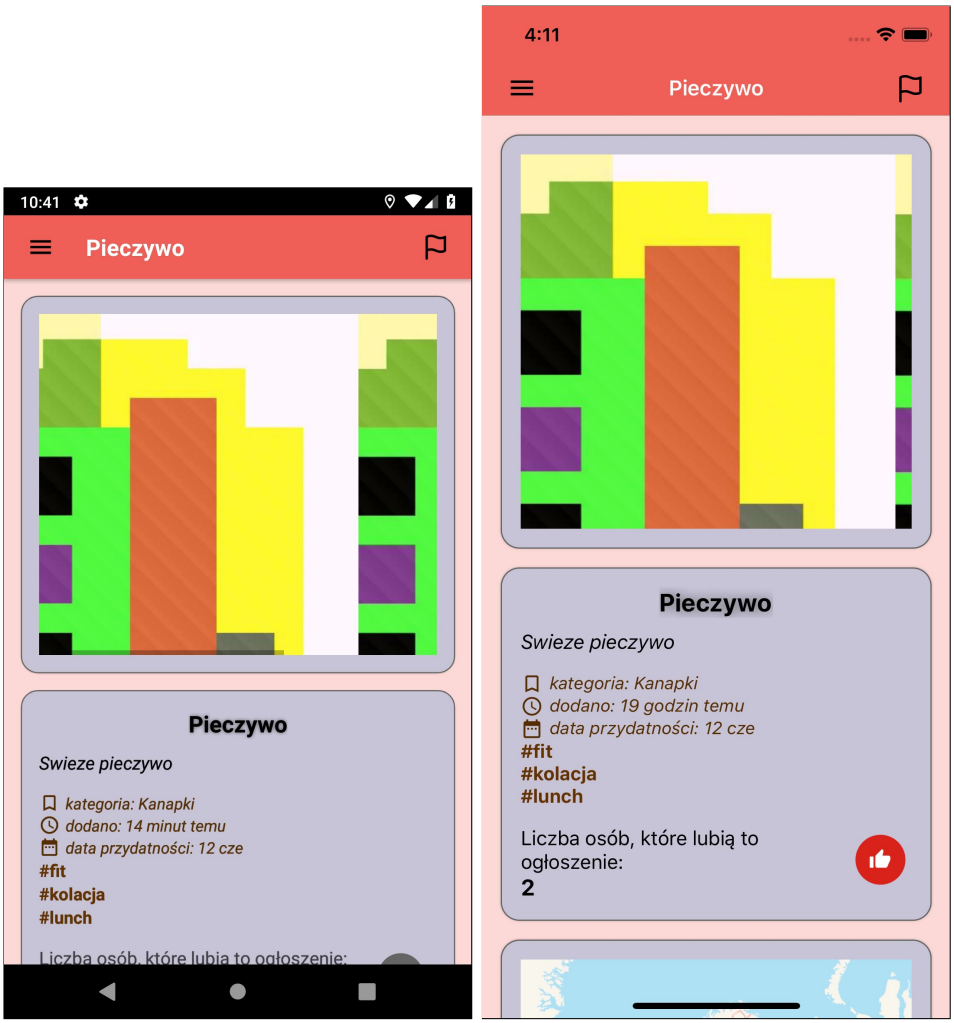
- Drugą są powiadomienia. Tak jak wspomnieliśmy we wcześniejszym podrozdziale, wymogiem było posiadanie konta w programie *Apple Developer Program*.

Wszystkie pozostałe funkcjonalności działają dokładnie tak, jak na systemie firmy Google. W szczególności obie wersje mają identyczny interfejs użytkownika. Różnice są widoczne jedynie w szczegółach charakterystycznych dla danej platformy, na przykład komponentach wyboru daty lub zdjęć z galerii. Na rysunkach: 3.2, 3.3 oraz 3.4 prezentujemy zrzuty tych samych ekranów aplikacji, wykonanych odpowiednio na emulatorze Android i symulatorze iPhone. Różnią się między sobą jedynie wielkością i proporcjami ekranów urządzeń mobilnych oraz wyświetlaną godziną systemową.



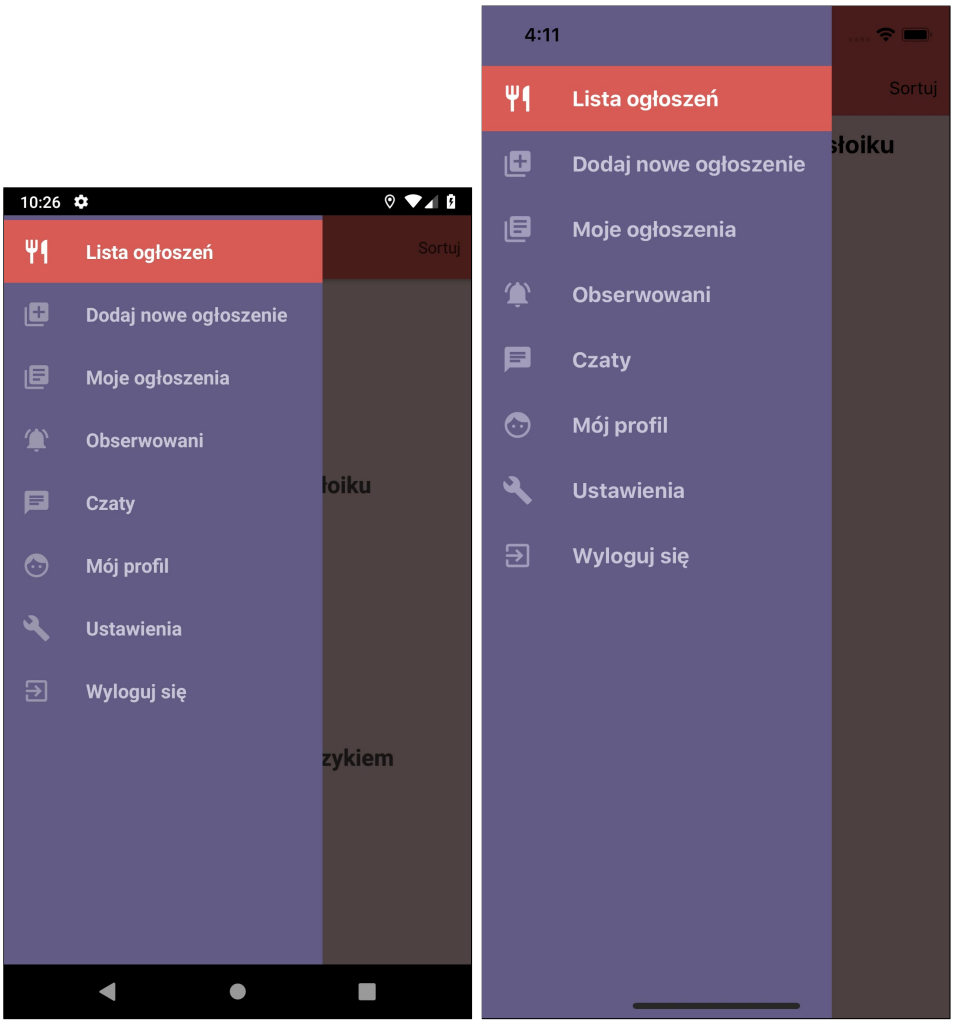
(a) Android. (b) iPhone.

Rysunek 3.2: Lista ogłoszeń.
Różnice w kolejności prezentowanych ogłoszeń wynikają z odmiennych kryteriów sortowania.



(a) Android. (b) iPhone.

Rysunek 3.3: Szczegóły ogłoszenia.



(a) Android. (b) iPhone.

Rysunek 3.4: Menu główne.

Rozdział 4

Infrastruktura serwera

Potrzeba udostępnienia pierwszej wersji aplikacji *Podziel się* zamawiającej postawiła nas przed koniecznością uruchomienia stałego serwera aplikacyjnego. Musieliśmy podjąć ważną decyzję dotyczącą wyboru platformy, na której będzie uruchomiony nasz serwer. Z uwagi na bardzo dużą liczbę firm oferujących usługi typu wirtualny prywatny serwer, nie był to łatwy wybór.

4.1. Kryteria

Przy wyborze usługodawcy kierowaliśmy się następującymi kryteriami:

- **Cena** — w fazie rozwojowej systemu z aplikacji korzystało tylko kilka osób: nasz zespół oraz zamawiająca, nie potrzebowaliśmy więc drogiego i bardzo wydajnego serwera,
- **Skalowalność** — chcieliśmy mieć możliwość łatwego zwiększenia dostępnych zasobów, jeśli wzrośnie liczba osób korzystających z systemu,
- **System operacyjny Linux** — wybrany serwer powinien działać w oparciu o jedną z dystrybucji systemu Linux,
- **Łatwa administracja** — nikt z nas nie miał wcześniej do czynienia z administrowaniem serwerem, wybrany serwis musiał więc być dostatecznie prosty w obsłudze.

4.2. Amazon Lightsail

Nasz wybór padł na usługę *Lightsail* firmy Amazon. Decyzja o wybraniu produktu firmy Amazon została podjęta z uwagi na jej wydajność, skalowalność, stosunkowo niską awaryjność, akceptowalną cenę ($3,50 to koszt najtańszego miesięcznego pakietu) oraz obecność rozbudowanych mechanizmów zapewniających bezpieczeństwo danych. *Lightsail* udostępnia maszynę wirtualną działającą pod kontrolą systemu Ubuntu ze statycznym adresem IP.

Administracja jest możliwa za pomocą witryny internetowej. Możliwe jest między innymi: zresetowanie maszyny, reinstalacja systemu czy wykonanie i przywrócenie kopii zapasowej.

4.3. Oprogramowanie serwera

Część serwerowa aplikacji działa pod kontrolą serwera Gunicorn. Gunicorn odpowiada za wykonanie odpowiedniego kodu języka Python oraz współpracę z serwerem pośredniczącym nginx. nginx odbiera połączenia od klientów oraz obsługuje statyczne zapytania HTTP, zaś

dynamiczne przekierowuje do Gunicorna i Django z wykorzystaniem standardu WSGI (Web Server Gateway Interface). Po otrzymaniu odpowiedzi nginx odsyła klientowi wynik zapytania. Wprowadzenie serwera proxy pozwala na łatwiejszą regulację obciążenia oraz zapobieganie opóźnieniom wynikającym z konieczności obsługi klientów dysponującym powolnym połączeniem internetowym.

Baza danych PostgreSQL wraz z wtyczką PostGIS uruchomiona jest w kontenerze Docker.

4.4. Dostęp do serwera

Połączenie z serwerem odbywa się poprzez protokół SSH. Możliwy jest również dostęp za pośrednictwem konsoli dostępnej na stronie internetowej serwisu *Lightsail*.

4.5. Dalszy rozwój

W miarę dalszego rozwoju projektu, narzędzie firmy Amazon oferuje elastyczne plany zapewniające większą liczbę rdzeni procesora, pamięci RAM czy przestrzeni dyskowej. Istnieje również możliwość łatwej integracji z innymi produktami z rodziny Amazon Web Services (AWS), takimi jak dedykowane serwery do równoważenia obciążenia, narzędzia do zarządzania domenami i certyfikatami SSL.

Rozdział 5

Przypadki użycia

Niniejszy rozdział prezentuje kluczowe przypadki użycia aplikacji, ze szczególnym uwzględnieniem interakcji pomiędzy użytkownikiem a komponentami systemu. Przedstawione zostały również ścieżki alternatywne, a także dodatkowe funkcjonalności, o które można wzbogacić aplikację w dalszej perspektywie czasowej.

5.1. Rejestracja

5.1.1. Krótki opis

Utworzenie nowego konta użytkownika w aplikacji.

5.1.2. Cele

- Umożliwienie aplikacji identyfikacji użytkowników w celu dostarczenia im określonych funkcjonalności.
- Uzyskanie informacji koniecznych do poprawnego działania aplikacji.

5.1.3. Warunki wstępne

- Aplikacja jest zainstalowana na urządzeniu użytkownika.
- Użytkownik dysponuje stałym połączeniem internetowym.

5.1.4. Czynności

Czynności podstawowe

1. Użytkownik naciska przycisk “Załóż nowe konto”.
2. Aplikacja wyświetla użytkownikowi formularz rejestracyjny.
3. Użytkownik wprowadza swoje imię, nazwisko, adres e-mail, hasło.
4. W polu “Powtórz hasło” użytkownik wpisuje drugi raz to samo hasło.
5. Użytkownik naciska przycisk “Utwórz konto”.
6. Aplikacja potwierdza poprawną rejestrację.

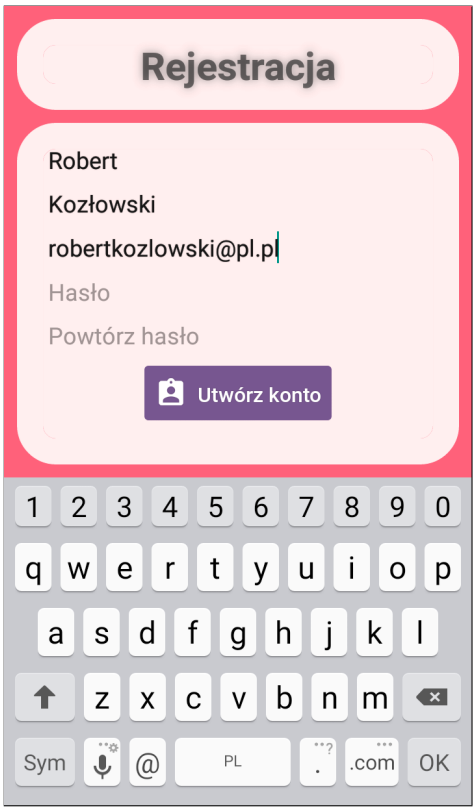
7. Następuje przejście do ekranu dodawania zdjęcia profilowego. Początkowo widoczny jest domyślny awatar.
   - Jeżeli użytkownik nie chce zamieścić własnego zdjęcia, naciska przycisk “Dalej”.
   - W przypadku wskazania opcji “Zmień zdjęcie profilowe” aplikacja wyświetla okienko z prośbą o wskazanie źródła zdjęcia, do wyboru: załączenie zdjęcia z pamięci urządzenia (galerii) lub utworzenie aparatem nowego zdjęcia.
     - Jeżeli użytkownik wyrazi chęć dodania istniejącego zdjęcia, aplikacja uruchamia klasyczny interfejs przeglądania galerii systemu operacyjnego.
     - Wybór opcji zrobienia nowego zdjęcia przekazuje sterowanie wbudowanej w system operacyjny aplikacji obsługującej aparat.
8. Aplikacja pyta użytkownika, czy chce korzystać z usługi geolokalizacji do sortowania listy ogłoszeń względem aktualnego położenia.
9. Użytkownik wyraża zgodę na korzystanie z geolokalizacji.
10. Proces rejestracji został zakończony — użytkownik jest domyślnie zalogowany do aplikacji na danym urządzeniu.

Czynności alternatywne

1. Jeśli użytkownik wypełnił formularz w sposób niepoprawny:
   - za drugim razem wprowadził inne hasło,
   - nie wypełnił któregoś z pól,
   - wprowadził niepoprawny składniowo adres e-mail,

   aplikacja powiadamia użytkownika o popełnionym błędzie i odsyła go z powrotem do edycji formularza.
2. Jeżeli użytkownik nie wyraził zgody na korzystanie z geolokalizacji, aplikacja pyta użytkownika, względem jakiego miejsca ma sortować ogłoszenia i wyświetla interaktywną mapę. Aby kontynuować, użytkownik musi wskazać konkretne miejsce na mapie.

(a) Ekran powitalny.



(b) Formularz rejestracyjny.

(c) Wybór zdjęcia profilowego.

Rysunek 5.1: Proces rejestracji nowego konta.

5.2. Dodawanie nowego ogłoszenia

5.2.1. Krótki opis

Wprowadzenie przez użytkownika informacji potrzebnych do stworzenia nowego ogłoszenia oraz opublikowanie go na liście aktualnie dostępnych ogłoszeń.

5.2.2. Cele

Zapewnienie możliwości przeglądania nowo utworzonego ogłoszenia innym użytkownikom aplikacji.

5.2.3. Warunki wstępne

- Aplikacja jest zainstalowana na urządzeniu użytkownika.
- Użytkownik ma poprawnie założone konto.
- Użytkownik jest zalogowany w aplikacji.
- Użytkownik dysponuje stabilnym połączeniem internetowym.

5.2.4. Czynności

Czynności podstawowe

1. Użytkownik naciska oznaczony przycisk, który rozwija szufladkowe menu.
2. Użytkownik wskazuje w menu pozycję "Dodaj nowe ogłoszenie".
3. Na ekranie pojawia się formularz dodawania ogłoszenia z podpisanymi polami.
4. Użytkownik wprowadza (w dowolnej kolejności) wymagane informacje:
    - Tytuł ogłoszenia, który będzie widoczny na liście ogłoszeń
    - Właściwa treść ogłoszenia
    - Sugerowana data przydatności do spożycia
        - Użytkownik otwiera okienko kalendarza z graficznym rozkładem dni bieżącego miesiąca.
        - W razie potrzeby użytkownik przechodzi za pomocą przycisków-strzałek do dalszych miesięcy.
        - Użytkownik wskazuje konkretny dzień w kalendarzu.
    - Preferowany adres odbioru (z rozwijanej listy uprzednio zapisanych w aplikacji miejsc odbioru)
    - Kategoria ogłoszenia (np. pieczywo, mięso, warzywa, dania pasteryzowane...)
        - Użytkownik rozwija listę dostępnych kategorii.
        - Użytkownik wskazuje kategorię, która wydaje się najbardziej odpowiednia do zawartości ogłoszenia.
    - Hasztagi przypisane do ogłoszenia (np. `#bezglutenu`, `#mięsne`, `#słodkie`, `#wegetariańskie`...)
        - Użytkownik rozwija listę dostępnych hasztagów.

  - Użytkownik wybiera (być może wiele) hasztagów pasujących do ogłoszenia.
  - Dostępna jest także wyszukiwarka hasztagów.

5. Użytkownik przesuwa widok ekranu na koniec formularza i naciska przycisk "Dodaj ogłoszenie".

6. Aplikacja komunikuje się z serwerem, wysyłając prośbę o dodanie wprowadzonego ogłoszenia do bazy danych.

7. Aplikacja otrzymuje odpowiedź od serwera — ogłoszenie zostało dodane pomyślnie.

8. Użytkownik dostaje informację zwrotną: aplikacja wyświetla na ekranie listę ogłoszeń użytkownika z nowo dodanym ogłoszeniem na szczycie listy.

Czynności alternatywne

Użytkownik może opcjonalnie dodać zdjęcia artykułu żywnościowego, opakowania, etykiet.

1. Użytkownik wskazuje przycisk "Dodaj nowe zdjęcie".

2. Aplikacja wyświetla okienko z prośbą o wskazanie źródła zdjęcia, do wyboru: załączenie zdjęcia z pamięci urządzenia (galerii) lub utworzenie aparatem nowego zdjęcia.

   - Jeżeli użytkownik wyrazi chęć dodania istniejącego zdjęcia, aplikacja uruchamia klasyczny interfejs przeglądania galerii systemu operacyjnego.
   - Wybór opcji zrobienia nowego zdjęcia przekazuje sterowanie wbudowanej w system operacyjny aplikacji obsługującej aparat.

3. Jeżeli zdjęcie ma zbyt duży rozmiar, następuje jego kompresja (po stronie aplikacji).

4. Na ekranie formularza pojawia się miniaturka dodanego zdjęcia.

Czynności niepoprawne

We wszystkich wymienionych poniżej przypadkach na ekranie wyświetla się okienko ze stosownym komunikatem. Użytkownik zostaje przekierowany z powrotem do ekranu edycji formularza, aby mógł wprowadzić zmiany.

1. Użytkownik nie wprowadził kompletu wymaganych informacji.

2. Wartości niektórych pól formularza nie spełniają kryteriów akceptacji wskazanych przez zamawiającą (np. zbyt mała liczba wybranych hasztagów)

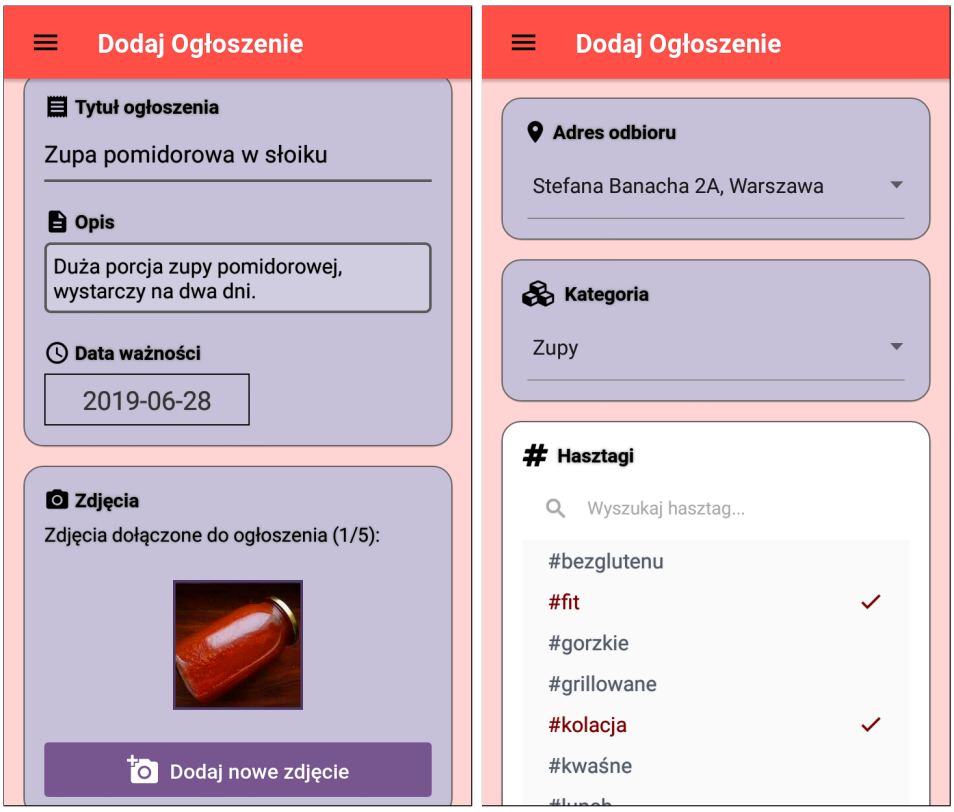
3. Nastąpił błąd komunikacji z serwerem — prośba o dodanie ogłoszenia nie została rozpatrzona.

4. Serwer zasygnalizował błąd podczas operacji dodawania ogłoszenia do bazy danych.

5.2.5. Możliwe rozwinięcia

Automatyczna klasyfikacja ogłoszeń

Wyświetlenie podpowiedzi co do wyboru odpowiedniej kategorii na podstawie analizy słownictwa użytego w treści ogłoszenia.



Rysunek 5.2: Formularz dodawania nowego ogłoszenia.

5.3. Konwersacja (czat) z innym użytkownikiem

5.3.1. Krótki opis

Wykorzystanie funkcji czatu do komunikacji z innym użytkownikiem aplikacji.

5.3.2. Cele

- Możliwość zadawania wystawcy ogłoszenia dodatkowych pytań.
- Udostępnienie prywatnego kanału do ustalania szczegółów odbioru produktu.
- Zachęcenie użytkowników aplikacji do współtworzenia aktywnej społeczności.

5.3.3. Warunki wstępne

- Aplikacja jest zainstalowana na urządzeniu użytkownika.
- Użytkownik ma poprawnie założone konto.
- Użytkownik jest zalogowany w aplikacji.
- Użytkownik dysponuje stabilnym połączeniem internetowym.

5.3.4. Czynności

Czynności podstawowe

1. Użytkownik A otwiera ekran czatu z użytkownikiem B. Może to zrobić na kilka sposobów:
   - Naciskając przycisk "Otwórz czat z wystawiającym" widocznego na ekranie ogłoszenia wystawionego przez B.
   - Wybierając opcję "Nowy czat" w sekcji *Czaty*.
   - Wskazując awatar użytkownika B w sekcji *Czaty* (jeżeli użytkownicy A oraz B rozmawiali ze sobą wcześniej).
2. W dolnej części ekranu pojawia się pole tekstowe do wpisania wiadomości oraz przycisk służący do jej wysłania. Jeżeli użytkownicy A i B rozmawiali ze sobą wcześniej, na ekranie będzie także widoczna przewijana lista poprzednich wiadomości, uporządkowana chronologicznie.
3. Użytkownik A wpisuje treść wiadomości i wysyła ją.
4. Nowa wiadomość zostaje dopisana do prowadzonej historii konwersacji.
5. Użytkownik A oczekuje na odpowiedź użytkownika B:
   - Jeżeli osoba B udzieli odpowiedzi od razu, jej treść pojawi się na ekranie użytkownika A w czasie rzeczywistym.
   - W przypadku, gdy użytkownik A nie korzysta w danej chwili z aplikacji "Podziel się" w systemie Android, otrzyma powiadomienie (notyfikację) systemu operacyjnego o nowej wiadomości od użytkownika B.

Czynności niepoprawne

1. Nastąpił błąd komunikacji z serwerem — nie udało się wysłać wiadomości.
2. Serwer zasygnalizował błąd podczas dopisywania wiadomości do historii konwersacji.

5.3.5. Możliwe rozwinięcia

Przesyłanie obrazów

Umożliwienie przesyłania wiadomości w formie zdjęć. W tym celu użytkownik wysyłający wiadomość mógłby wybrać piktogram "prześlij obraz". Dalszy scenariusz byłby analogiczny do opisanego w *Czynnościach alternatywnych* z punktu 5.2.4.

Rysunek 5.3: Ekran czatu.

5.4. Ocena innego użytkownika

5.4.1. Krótki opis

Wystawienie innemu użytkownikowi oceny w skali od 1 do 6 oraz możliwość napisania krótkiego komentarza dotyczącego współpracy z nim.

5.4.2. Cele

Informowanie innych użytkowników o jakości współpracy z daną osobą, w szczególności ostrzeganie innych przed nierzetelnymi użytkownikami.

5.4.3. Warunki wstępne

- Aplikacja jest zainstalowana na urządzeniu użytkownika.
- Użytkownik ma poprawnie założone konto.
- Użytkownik jest zalogowany w aplikacji.
- Użytkownik dysponuje stabilnym połączeniem internetowym.

5.4.4. Czynności

Czynności podstawowe

1. Użytkownik A otwiera publiczny profil użytkownika B (zwykle poprzez naciśnięcie zdjęcia profilowego na ekranie ogłoszenia, które użytkownik B wystawił).
2. Wyświetla się ekran z profilem użytkownika B, a na nim następujące elementy:
   - Imię oraz nazwisko użytkownika.
   - Zdjęcie profilowe.
   - Licznik osób obserwujących.
   - Przycisk "Obserwuj"/ "Przestań obserwować".
   - Zbiorcza ocena innych użytkowników w skali od 1 do 6.
   - Sześciogwiazdkowa skala własnej oceny.
   - Pole tekstowe przeznaczone na wpisanie komentarza.
   - Przycisk "Wyślij opinię".
   - Lista opinii i komentarzy wystawionych przez społeczność użytkowników.
3. Użytkownik A ocenia użytkownika B w skali od 1 do 6.
4. Użytkownik A wpisuje krótki komentarz odnośnie użytkownika B.
5. Użytkownik A naciska przycisk "Wyślij opinię".

Czynności alternatywne

Użytkownik A może co najwyżej raz ocenić użytkownika B. Jeśli miało już to miejsce, użytkownik A może edytować swoją ocenę na sześciostopniowej skali. Istnieje jednak możliwość wielokrotnego wystawiania komentarzy (np. dla każdego ogłoszenia B oddzielnie).

Czynności niepoprawne

1. Nastąpił błąd komunikacji z serwerem — prośba o dodanie/edycję opinii nie została rozpatrzona.
2. Serwer zasygnalizował błąd podczas operacji dodawania/edycji opinii w bazie danych.

5.4.5. Możliwe rozwinięcia

Blokada możliwości oceny danego użytkownika do momentu, w którym odbierzemy od niego (lub on od nas) produkt z przynajmniej jednego ogłoszenia.

Rysunek 5.4: Profil użytkownika.

5.5. Odbiór produktów oferowanych na liście ogłoszeń

5.5.1. Krótki opis

Umożliwienie użytkownikowi, który jest zainteresowany odbiorem przedmiotu z ogłoszenia (A), oraz użytkownikowi, który wystawił ogłoszenie (B), wspólnego ustalenia szczegółów odbioru.

5.5.2. Cele

- Umożliwienie użytkownikom ustalenia szczegółów odbioru produktu (takich jak termin odbioru czy miejsce odbioru).
- Łatwiejsze zarządzanie statusem ogłoszeń.

5.5.3. Warunki wstępne

- Aplikacja jest zainstalowana na urządzeniach użytkowników A i B.
- Użytkownicy mają poprawnie założone konta.
- Użytkownicy są zalogowani w aplikacji.
- Użytkownik B dodał ogłoszenie.
- Użytkownicy dysponują stabilnym połączeniem internetowym.

5.5.4. Czynności

Czynności podstawowe

Użytkownik A

1. Użytkownik otwiera listę dostępnych ogłoszeń i wskazuje jedno z nich.
2. Na ekranie ze szczegółami ogłoszenia użytkownik naciska przycisk *"Otwórz czat z wystawiającym"*.
3. Użytkownikowi otwiera się ekran czatu z autorem ogłoszenia.
4. Użytkownik wysyła wiadomość, w której wyraża chęć odbioru produktu z ogłoszenia.
5. Jeżeli użytkownik otrzyma odpowiedź pozytywną, użytkownicy ustalają między sobą szczegóły odbioru.

Użytkownik B

1. Użytkownik otrzymuje nową wiadomość na czacie z chęcią odbioru oferowanego produktu.
2. Użytkownik decyduje, czy chce oddać oferowany produkt użytkownikowi A:
   - Jeżeli B nie chce oddać produktu użytkownikowi A, wysyła wiadomość odmowną i interakcja się kończy.
   - Jeżeli B chce oddać produkt, wysyła wiadomość pozytywną, interakcja trwa dalej.
3. Użytkownicy ustalają między sobą szczegóły odbioru.
4. Po oddaniu oferowanego produktu, użytkownik naciska przycisk *"Oznacz jako odebrane"* na ekranie szczegółów swojego ogłoszenia, dzięki czemu produkt zniknie z listy dostępnych ogłoszeń. Podczas tej operacji użytkownik musi wskazać, dla kogo została dokonana rezerwacja.

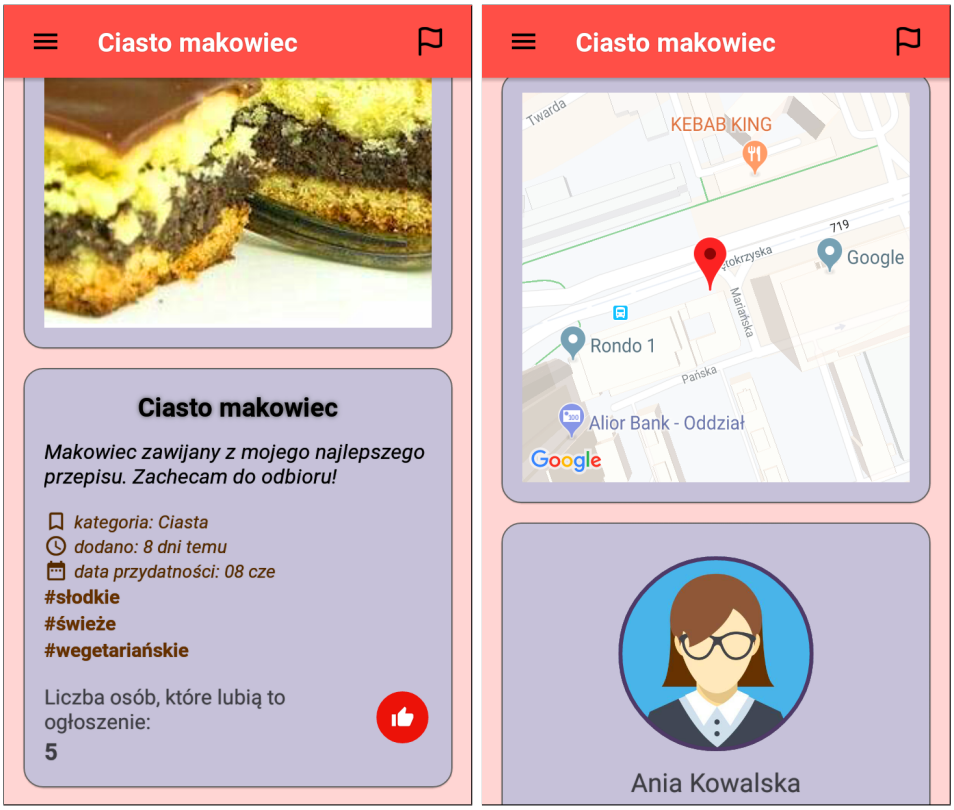
Czynności alternatywne

Jeżeli użytkownik chciałby wycofać ogłoszenie z listy, może nacisnąć przycisk *"Usuń ogłoszenie"*.

Czynności niepoprawne

1. Czynności niepoprawne z podpunktu 5.3.4 dotyczące funkcjonalności czatu.
2. Nastąpił błąd komunikacji z serwerem — prośba o oznaczenie ogłoszenia jako oddane nie została rozpatrzona.
3. Serwer zasygnalizował błąd podczas operacji związanej z oznaczeniem ogłoszenia jako oddane.

Rysunek 5.5: Lista dostępnych ogłoszeń.



Rysunek 5.6: Szczegóły ogłoszenia.

Rozdział 6

Zarządzanie projektem

6.1. Wykorzystane narzędzia

W trakcie prac nad projektem głównym kanałem komunikacji był portal Slack. Wykorzystywaliśmy go zarówno do komunikacji między członkami zespołu, jak również między zespołem a zamawiającą. Dzięki niemu mogliśmy w łatwy sposób dzielić się plikami i dokumentami. Niezwykle pomocna okazała się również możliwość oznaczania postów z ważnymi informacjami, na przykład instrukcją dostępu do serwera.

Kod źródłowy programu oraz tekst pracy licencjackiej przechowywane były w serwisie GitHub, który pozwalał także w prosty sposób spisywać i aktualizować listę zadań do zrobienia.

6.2. Współpraca z zamawiającą

Współpraca z zamawiającą była istotną częścią tego projektu. Pani Maria nie ma doświadczenia w wytwarzaniu oprogramowania, nie wiedziała więc, jakich informacji potrzebujemy. Aby wzajemnie dobrze się porozumieć, prowadziliśmy częste rozmowy, zwłaszcza na początku procesu tworzenia aplikacji. Pani Maria na co dzień mieszka w Londynie, co znacząco utrudniło spotkania "twarzą w twarz". Udało nam się zorganizować tylko jedno takie spotkanie (na początku współpracy), ale zaowocowało ono dość dokładnym spisem oczekiwanych funkcjonalności, dzięki któremu dalsza praca przebiegała bez żadnych nieporozumień. Przez cały czas staraliśmy się na bieżąco informować zamawiającą o postępach dokonywanych w projekcie, w szczególności udostępnialiśmy aktualne wersje aplikacji, aby upewnić się, że spełnia oczekiwania pani Marii. Niestety, w końcowej fazie projektu komunikacja nie była już tak częsta: pomimo wielu próśb, przestaliśmy regularnie otrzymywać informację zwrotną dotyczącą działania aplikacji. Jednakże pomimo trudności związanych z komunikacją, udało nam się stworzyć aplikację zgodną z oczekiwaniami zamawiającej.

6.3. Organizacja pracy

Nie korzystaliśmy w bezpośredni sposób z żadnej ze znanych metodyk zarządzania projektami programistycznymi, jednakże trzymaliśmy się kilku istotnych zasad i dobrych praktyk. Przede wszystkim na bieżąco spisywaliśmy listę rzeczy do zrobienia. Staraliśmy się również, aby zadania były jak najmniejsze. Lista była aktualizowana co tydzień na zajęciach z przedmiotu Zespołowy Projekt Programistyczny. Każdy z nas samodzielnie decydował, które zadanie

zechce wykonać jako następne. Wspólnie zaprojektowaliśmy architekturę aplikacji oraz podejmowaliśmy najważniejsze decyzje dotyczące projektu. Na każdych zajęciach staraliśmy się na bieżąco omawiać problemy, które napotkaliśmy.

Co ważne, nie dzieliliśmy zadań na te z części mobilnej i serwerowej. Pojedyncze zadanie dotyczyło najczęściej zaimplementowania jakiejś funkcjonalności, na przykład dodawania zdjęć do ogłoszenia. Osoba odpowiedzialna za zadanie implementowała zarówno część dotyczącą aplikacji mobilnej, jak i serwerowej. W ten sposób mogliśmy znacznie zrównoleglić prace nad projektem (przykładowo, osoba rozwijająca aplikację mobilną nie musiała czekać, aż ktoś inny zrealizuje niezbędną pracę po stronie serwera). Ponadto uniknęliśmy w ten sposób nieporozumień związanych z działaniem nowo napisanego kodu. Co więcej — każdy z nas, w równym stopniu, nauczył się zarówno tworzenia aplikacji mobilnej we frameworku React Native, jak i realizacji części serwerowej z wykorzystaniem frameworka Django.

6.4. Podział obowiązków

6.4.1. Kamil Ćwintal

- widok listy ogłoszeń,
- zdjęcia ogłoszeń,
- zdjęcia profilowe użytkowników,
- kategorie i hasztagi w ogłoszeniach,
- dopracowanie interfejsu graficznego aplikacji.

6.4.2. Szymon Gajda

- dodawanie ogłoszeń,
- dodawanie lokalizacji do ogłoszeń,
- wyświetlanie mapy w ogłoszeniach,
- filtrowanie i sortowanie ogłoszeń po odległości między użytkownikiem a ogłoszeniem,
- logowanie z Facebookiem,
- zgłaszanie nieodpowiednich ogłoszeń.

6.4.3. Paweł Giżka

- logowanie i rejestracja,
- czat między użytkownikami,
- powiadomienia,
- sortowanie ogłoszeń,
- możliwość obserwacji wybranych użytkowników,
- konfiguracja serwera aplikacji.

6.4.4. Tomasz Kanas

- widok szczegółów ogłoszenia,
- widok własnych ogłoszeń,
- widok profilu innych użytkowników,
- ocenianie użytkowników,
- widok i możliwość edycji własnego profilu.

Rozdział 7

Zawartość płyty

`/PodzielSieBackend/`	Kod źródłowy części serwerowej.
`/PodzielSieBackend/README.md`	Instrukcja uruchamiania części serwerowej.
`/PodzielSieMobile/`	Kod źródłowy części mobilnej.
`/PodzielSieMobile/README.md`	Instrukcja uruchamiania części mobilnej.
`/PracaLicencjacka.pdf`	Praca licencjacka.
`/Prezentacja.pdf`	Końcowa prezentacja projektu.

Rozdział 8

Podsumowanie

W trakcie ostatnich siedmiu miesięcy powstała w pełni funkcjonalna aplikacja mobilna, która w istotny sposób może przyczynić się do zmniejszenia ilości marnowanej żywności. Udało nam się zrealizować zdecydowaną większość wymagań zamawiającej, w szczególności wszystkie niezbędne funkcjonalności zostały zaimplementowane.

W trakcie prac nad projektem nauczyliśmy się wielu zagadnień związanych z inżynierią oprogramowania, tworzenia aplikacji mobilnych z wykorzystaniem React Native, implementacji części serwerowej we frameworku Django, integracji ze zewnętrznymi usługami oraz administrowania serwerem aplikacyjnym. Była to dla nas również okazja do rozwinięcia tak zwanych umiejętności miękkich — pracy w zespole, utrzymywania kontaktu z zamawiającym.

Obecnie czekamy na informacje od zamawiającej dotyczące wdrożenia aplikacji do użytku. Niezależnie od tego, czy aplikacja zostanie wdrożona, jesteśmy z niej bardzo zadowoleni. Realizacja tego projektu dała nam wiele satysfakcji.

Bibliografia

[FAO] raport Organizacji Narodów Zjednoczonych do spraw Wyżywienia i Rolnictwa *"Global food losses and food waste"*, `fao.org/3/mb060e/mb060e.pdf`

[Eurostat] raport techniczny Komisji Europejskiej z 2010 roku *"Preparatory Study on Food Waste Across EU 27"*, `ec.europa.eu/environment/eussd/pdf/bio_foodwaste_report.pdf`

[Kantar Millward Brown] raport Federacji Polskich Banków Żywności *"Nie marnuję jedzenia 2018"* z wynikami badań zrealizowanych przez Kantar Millward Brown, `bzsos.pl/2018/raportu-nie-marnuj-jedzenia-2018`

[Główny Urząd Statystyczny] badanie Głównego Urzędu Statystycznego *"Ubóstwo w Polsce w latach 2015 i 2016"* `stat.gov.pl/obszary-tematyczne/warunki-zycia/ubostwo-pomoc-spoleczna`

[Polityka] opracowanie *"Zgubione kalorie. Jak skutecznie walczyć z marnotrawieniem żywności"* dla magazynu Polityka Insight, `politykainsight.pl/gospodarka/ryzykaitrendy/_resource/multimedium/20145149`

[podzielmysie.pl] oficjalna strona akcji społecznej *"Podziel się Posiłkiem z Bezdomnymi"*, `podzielmysie.pl`

[reducefoodwaste.eu] oficjalna strona internetowa projektu Reduce Food Waste, `reducefoodwaste.eu`

[Framework] `pl.wikipedia.org/wiki/Framework`

[ec.europa.eu] Działania KE ws. marnowania żywności `ec.europa.eu/food/safety/food_waste_en`

[olioex.com] Statystyki dotyczące aplikacji *Olio*, `olioex.com`

[toogoodtogo.com] Oficjalna strona aplikacji *Too Good To Go*, `toogoodtogo.com`

[IDC] Popularność systemu Android w Polsce `gsmonline.pl/artykuly/sprzedaz-smartfonow-w-polsce-2017`

[React Native] `facebook.github.io/react-native`

[Mapy Google] `developers.google.com/maps/documentation`

[Firebase] `firebase.google.com/docs/cloud-messaging`

[Apple Push Notifications] `developer.apple.com/notifications`

[Amazon Web Services] aws.amazon.com

[Django] oficjalny samouczek Django dostępny na stronie djangoproject.com

[Gunicorn] gunicorn.org

[nginx] nginx.com

[PostgreSQL] postgresql.org