AGH Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Fizyki i Informatyki Stosowanej

Praca magisterska

Justyna Gacek

kierunek studiów: **Informatyka Stosowana**

specjalność: **Grafika komputerowa i przetwarzanie obrazów**

System do przeprowadzania głosowań i sondaży oparty na aplikacjach mobilnych i internetowych

Opiekun: **dr hab. inż. Maciej Wołoszyn**

Kraków, listopad 2020

Oświadczenie studenta

Uprzedzony(-a) o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2018 r. poz. 1191 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony(-a) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Jednocześnie Uczelnia informuje, że zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych Uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli Uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony, autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto Uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. — Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem systemu antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych.

.....
(czytelny podpis)

Kraków, listopad 2020

Tematyka pracy magisterskiej i praktyki dyplomowej Justyny Gacek, studentki drugiego roku studiów drugiego stopnia na kierunku informatyka stosowana, specjalności Grafika komputerowa i przetwarzanie obrazów

Temat pracy magisterskiej: **System do przeprowadzania głosowań i sondaży oparty na aplikacjach mobilnych i internetowych**

Opiekun pracy:	dr hab. inż. Maciej Wołoszyn
Recenzent pracy:	
Miejsce praktyki dyplomowej:	WFiIS AGH, Kraków

Program pracy magisterskiej i praktyki dyplomowej

1. Omówienie realizacji pracy magisterskiej z opiekunem.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Praktyka dyplomowa:
   - zdefiniowanie celu pracy oraz zaplanowanie funkcjonalności aplikacji,
   - wytypowanie narzędzi programistycznych oraz określenie możliwości ich użycia,
   - przegląd konkurencyjnych rozwiązań,
   - opracowanie architektury poszczególnych aplikacji w projekcie,
   - sporządzenie sprawozdania z praktyki.
4. Rozpoczęcie tworzenia aplikacji.
5. Rozwijanie aplikacji oraz sprawdzanie ich poprawnego działania na każdym z etapów.
6. Prezentacja gotowych aplikacji oraz zatwierdzenie ich przez opiekuna pracy.
7. Opracowanie redakcyjne pracy.

Termin oddania w dziekanacie:

....
(podpis kierownika katedry) (podpis opiekuna)

Merytoryczna ocena pracy przez opiekuna:

Merytoryczna ocena pracy przez recenzenta:

Za inspiracje, wyrozumiałość oraz pomoc przy realizacji pracy
pragnę serdecznie podziękować mojemu promotorowi
dr hab. inż. Maciejowi Wołoszynowi.

Spis treści

1 Wprowadzenie

W dzisiejszych czasach ważne jest badanie opinii publicznej na wiele tematów. Na podstawie wyników ankiet, sondaży oraz głosowań podejmowane są decyzje, które mogą mieć duży wpływ na życie każdego człowieka. Wiele instytucji, takich jak np. szkoły, banki, zakłady produkcyjne, opierają swoje decyzje na podstawie opinii pracowników. W przypadku konieczności podjęcia decyzji, której konsekwencje mogą być odczuwalne przez wszystkich członków danej wspólnoty, ważne jest, aby podjęta ona została zgodnie z wolą większości. Biorąc pod uwagę tempo życia przeciętnego człowieka, bardzo trudno jest znaleźć czas, aby brać udział w wielu ankietach czy też głosowaniach. Dzięki aplikacjom internetowym oraz mobilnym znacznie łatwiej jest angażować się w tego typu aktywności. Aplikacja, dzięki której w szybki, niewymagający wysiłku sposób można wyrazić swoją opinię, jest pożądanym narzędziem. Nierozłączną częścią życia człowieka stał się dostęp do Internetu, który w znacznej większości realizowany jest z wykorzystaniem urządzeń mobilnych. Na rynku znajdują się rozwiązania pozwalające na przeprowadzanie ankiet oraz sondaży, ale brak kompleksowości wykonania, oraz niedopracowanie istotnych funkcji sprawiają, że są to rozwiązania, które mogą zostać udoskonalone. Ważnym czynnikiem wpływającym na odczucia użytkownika jest estetyka wykonania danej aplikacji oraz ergonomiczne rozmieszczenie poszczególnych elementów na ekranie, które mogą sprawić, że dany projekt będzie chętnie wykorzystywany przez odbiorców. Aby osiągnąć sukces i stworzyć aplikację przyjazną użytkownikowi należy uwzględnić te kwestie już na etapie projektowania. Całościowe podejście do etapu tworzenia ma na celu zapewnienie kompatybilności pomiędzy poszczególnymi modułami aplikacji oraz zbliżony wygląd interfejsu niezależnie od rodzaju wykorzystanego urządzenia. Założeniem niniejszej pracy jest stworzenie w pełni funkcjonalnej aplikacji służącej do tworzenia ankiet, sondaży oraz głosowań. Istotną cechą wykonanej aplikacji jest systemowe podejście do modułu tworzenia, udostępniania oraz analizy wyników tworzonych formularzy. Aplikacja została stworzona w oparciu o wiele nowoczesnych technologii, aby możliwe było przygotowanie współdziałających ze sobą modułów w wersji przeglądarkowej oraz mobilnej. Do stworzenia znacznie bardziej rozbudowanej aplikacji internetowej wykorzystany został framework Angular, natomiast do modułu mobilnego język programowania Kotlin. Aplikacja backendowa została napisana przy użyciu frameworka Django. Kluczowym aspektem tego projektu było wykorzystanie nowoczesnych, stabilnych i wystarczająco rozwiniętych rozwiązań, dzięki którym istnieje możliwość wykonania projektu w sposób profesjonalny.

2 Cel pracy

Celem pracy jest przygotowanie możliwie uniwersalnego oraz łatwego w dalszej rozbudowie systemu informatycznego pozwalającego na wygodne zbieranie oraz przetwarzanie informacji dotyczących głosowań, sondaży, badań opinii oraz testów. Projekt będzie oparty na aplikacji internetowej, za pomocą której możliwe będzie tworzenia formularzy, udostępnianie ich potencjalnym respondentom, a także analiza zebranych wyników. Kolejnym rezultatem projektu będzie umożliwienie wypełniania ankiet z wykorzystaniem wcześniej wspomnianej aplikacji internetowej oraz specjalnie w tym celu przygotowanej aplikacji mobilnej. Oba rozwiązania będą posiadały zabezpieczenia chroniące anonimowość respondentów.

3 Użyte technologie

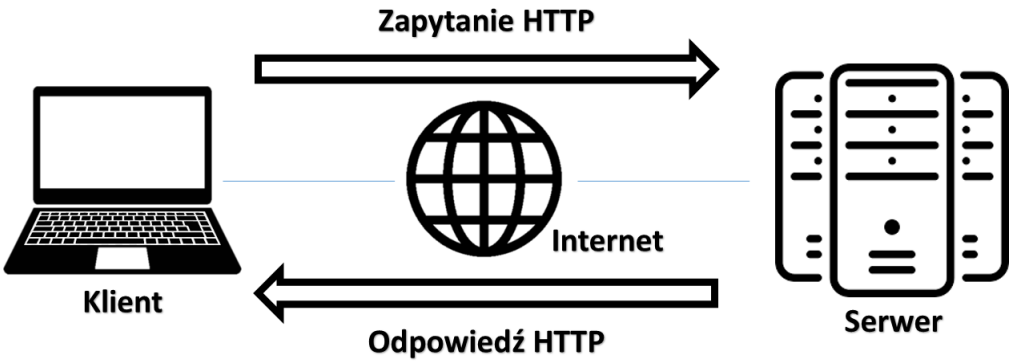
Do wykonania aplikacji, wytypowano technologie wyróżniające się popularnością oraz dostępnością rozwiązań. Taki wybór został podyktowany chęcią stworzenia nowoczesnych, intuicyjnych, dobrze wyglądających oraz szybkich aplikacji, które w pełni zadowolą potencjalnego klienta.

3.1 Technologie backendowe

W celu wykonania projektu niezbędne jest zastosowanie gotowych rozwiązań w postaci języków programowania, bibliotek, frameworków czy też baz danych. Poniższy rozdział zawiera opis rozwiązań wykorzystanych do stworzenia aplikacji backendowej.

3.1.1 Protokół HTTP

Do komunikacji pomiędzy aplikacją klienta, czyli przykładowo aplikacją internetową lub mobilną, a aplikacją serwerową, służą usługi internetowe (ang. Web Services). Proces wymiany danych odbywa się z użyciem sieci internetowej. Usługa może być stworzona w dowolnym języku programowania, a jej działanie jest niezależne od systemu operacyjnego. Stanowi zbiór standardów umożliwiających przesyłanie danych między aplikacjami w Internecie. Do komunikacji pomiędzy klientem i serwerem używany jest protokół sieciowy HTTP (ang. Hypertext Transfer Protocol). Komunikacja odbywa się poprzez przesyłanie wiadomości. Klient otwiera połączenie TCP z serwerem oraz wysyła żądanie (ang. request). Serwer generuje odpowiedź (ang. response), a następnie zwraca ją do klienta oraz zamyka połączenie. Protokół jest bezstanowy, ponieważ żadna ze stron biorących nim udział, nie zapamiętuje informacji o zaistniałym połączeniu. Standardowo protokół używa portu 90 do komunikacji. Na rysunku 1 przedstawiony został proces wymiany informacji pomiędzy klientem a serwerem.



Rysunek 1: Realizacja połączenia pomiędzy klientem a serwerem w protokole HTTP.

Format żądania oraz odpowiedzi mają podobną budowę. Składają się z linii rozpoczynającej,

nagłówka oraz opcjonalnego ciała. Początkowy wiersz dla żądania jest zbudowany z nazwy metody, ścieżki do lokalizacji żądanego zasobu, oraz używanej wersji protokołu HTTP. Poniżej przedstawiono przykładową linię rozpoczynającą dla żądania klienta:

GET /path/to/file/index.html HTTP/1.0

Zawarta w linii początkowej nazwa funkcji definiuje, jaką operację klient chce wykonać. Do najczęściej stosowanych metod protokołu HTTP należą:

- GET – pobranie istniejącego zasobu
- POST – dodanie nowego zasobu
- PUT – aktualizacja istniejącego zasobu
- DELETE – usunięcie istniejącego zasobu

Linia rozpoczynająca odpowiedzi serwera jest zbudowana z wersji protokołu HTTP, kodu zawierającego status oraz frazy w języku angielskim opisującej słownie ten kod. Wartość ta jest trzycyfrową liczbą, której pierwsza cyfra identyfikuje ogólną kategorię odpowiedzi. Poniżej przedstawione zostały dwa przykładowe wiersze rozpoczynające wiadomość zwrotną serwera:

HTTP/1.0 200 OK
HTTP/1.0 404 Not Found

Nagłówek zawiera metadane, czyli informacje o żądaniu, odpowiedzi, lub przesłanym ciele. Linie nagłówka definiowane są w formacie „nazwa: wartość". Przykładowo mogą informować o formacie przesyłanego dokumentu, kodowaniu, języku, w którym użytkownik chce czytać zawartość strony, rodzaju uwierzytelniania, dacie oraz wielu innych parametrach potrzebnych do prawidłowej komunikacji z serwerem.

Niektóre zapytania lub odpowiedzi HTTP mogą przesyłać ciało (ang. body). W przypadku odpowiedzi może to być zasób zwracany do klienta, a w żądaniu dane, które są przesyłane na serwer. Udostępniane informacje mogą przyjmować różne typy. Najbardziej popularnym formatem danych przesyłanych w wiadomościach HTTP jest JSON (ang. JavaScript Object Notation). [1]

3.1.2 REST API

API (ang. Application Programming Interface) to interfejs umożliwiający komunikację. Jest zbiorem odpowiednio przygotowanych metod, do których można odwołać się za pomocą adresów URL, zwanych punktami końcowymi (ang. Endpoints). Metody te zwykle komunikują się z bazą danych w celu pobrania lub modyfikacji zasobów oraz zwracają odpowiedź do klienta.

Aby ustandaryzować zasady tworzenia API został stworzony REST (ang. Representational

State Transfer). Jest to styl architektury używany do projektowania usług sieciowych, wprowadzający zbiór dobrych praktyk. Do wykonywania połączeń używany jest protokół HTTP. REST definiuje sześć zasad, które powinny być przestrzegane, aby stworzona usługa była zgodna z jego założeniami:

- Głównym założeniem jest całkowita niezależność aplikacji klienta oraz serwera, co sprawia, że każda z nich może być rozwijana niezależnie od drugiej, a także aplikacja serwerowa może spełniać wymagania wielu klientów.

- Kolejna zasada to bezstanowość. Zakłada ona, że każde żądanie przesyłane od klienta do serwera powinno zawierać wszystkie niezbędne informacje potrzebne do jego obsługi. Dzięki temu na serwerze nie powinny być zapisywane żadne informacje dotyczące połączenia z klientem.

- Odpowiedzi od serwera mogą być buforowane po stronie klienta. Powinien on jawnie wysyłać informacje czy zezwala na wykonanie takiego działania. Proces ten pozwala na eliminacje niektórych interakcji między klientem a serwerem. Takie podejście znacznie zmniejsza obciążenie serwera, szczególnie w przypadkach, gdy dane aktualizowane są rzadko, a co za tym idzie, poprawia wydajność oraz skalowalność.

- Kolejna z zasad dotyczy warstwowości systemu. Gdy pomiędzy klientem, a serwerem istnieją pośrednicy jak np. serwer proxy, nie będzie to miało żadnego wpływu zarówno na aplikacje klienta jak i serwera.

- Kolejna reguła stanowi o unikalności interfejsu. Można ją opisać czterema zasadami. W żądaniach, zasoby identyfikuje się przy użyciu unikalnych adresów URI, a sposób ich prezentacji jest niezależny. Klient nie może bezpośrednio wchodzić w interakcję z zasobami na serwerze, przykładowo wykonując zapytania na bazie danych. Po otrzymaniu zasobu od serwera klient może prosić o jego usunięcie bądź modyfikację, ponieważ posiada wszystkie niezbędne do tego dane. Do wykonania tej operacji nie musi on posiadać informacji, w jaki sposób odbywa się to po stronie serwera. Każde żądanie oraz odpowiedź powinny zawierać niezbędne informacje jak np. typ otrzymanych danych, w celu zrozumienia ich przez odbiorcę. Klient, który raz skorzystał z API, powinien mieć możliwość dynamicznego korzystania z łączy, które są udostępniane przez serwer, aby mógł odnaleźć wszystkie dostępne zasoby.

- Ostatnia zasada dotyczy przesyłania kodu do klienta, aby tymczasowo rozszerzyć jego funkcjonalność. Jest ona opcjonalna.

Spełnienie założeń REST zapewnia, że stworzony system jest wydajny, niezawodny, skalowalny, przenośny oraz prosty. Jeśli zaprojektowane API spełnia zasady REST, nazywane jest REST API, a usługa internetowa przestrzegająca tych reguł nazywana jest RESTful Web Service. [2]

3.1.3 Python

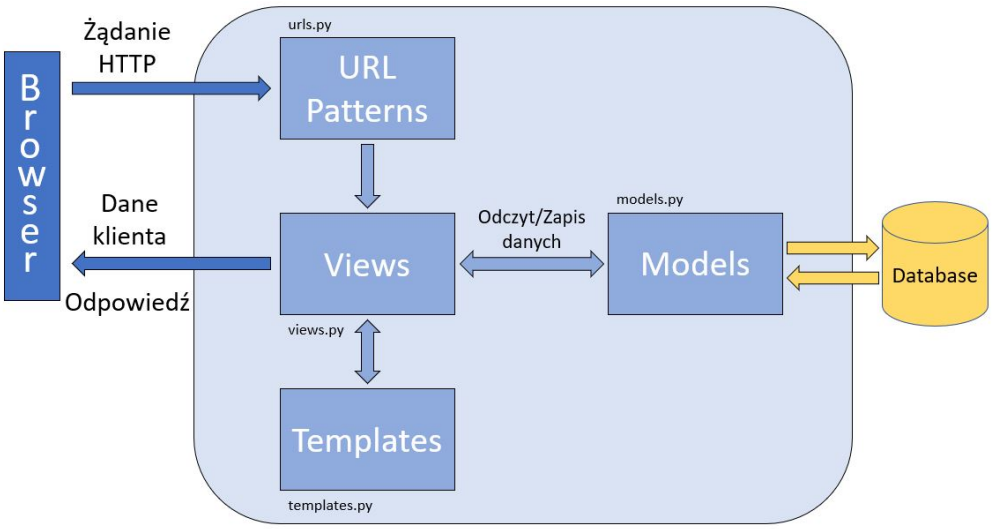
Wykorzystany w pracy język programowania Python jest językiem wysokiego poziomu, który zaprojektowany został do ogólnych zastosowań. Szczególnie ceniony jest w projektach, które skupiają się na analizie danych, uczeniu maszynowym, tworzeniu stron internetowych zarówno po stronie frontendu, jak i backendu oraz szeroko pojętym pisaniu skryptów. Obecnie Python znajduje się w czołówce najbardziej popularnych języków. Jest on ciągle rozwijany oraz dopracowywany. Cechami, które charakteryzują język Python, są łatwość obsługi, czytelność kodu oraz produktywność. Dlatego w szczególności naddaje się dla osób początkujących. Duża ilość bibliotek oraz frameworków sprawia, że tworzenie nowych projektów jest znacznie prostsze. Język jest dynamicznie typowany, co oznacza, że nie ma potrzeby ustawiania typów podczas deklarowania zmiennych. Cechuje go automatyczne zarządzanie pamięcią, za które odpowiada *garbage collector*. Język zapewnia różne rodzaje programowania obiektowe, strukturalne oraz funkcyjne. Python jest językiem interpretowanym, a jego interpreter jest dostępny dla wielu systemów operacyjnych. [3]

3.1.4 Django

Framework Django jest darmowym rozwiązaniem przeznaczonym do tworzenia aplikacji internetowych. Jest napisany z wykorzystaniem popularnego języka Python. Zaletą Django jest automatycznie generowany i kompletny panel administracyjny. Idea działania polega na oddzieleniu logiki aplikacji (widok), logiki biznesowej (model), wyglądu (szablon) oraz baz danych. Ważną cechą aplikacji napisanych w Django jest skalowalność oraz wydajne działanie w warunkach obciążenia. Dużą zaletą jest mechanizm mapowania obiektowo-relacyjnego, który umożliwia działanie na bazach danych, nie wykorzystując zapytań SQL. Zapewnia wsparcie dla szerokiego grona baz danych, które mogą okazać się kluczowe w przypadku niektórych projektów.

Gdy Django otrzyma zapytanie od klienta, *URL dispatcher* przegląda wzorce adresów URL z pliku urls.py w celu dopasowania do odpowiedniej ścieżki. Gdy pasujący wzorzec adresu zostanie znaleziony, wywoływany jest widok przypisany do niego. Widoki posiadają funkcje odpowiadające za logikę biznesową, przetwarzają dane oraz przekazują je do szablonów. Korzystają z modeli Django, aby skomunikować się z bazą danych w celu uzyskania rekordów, bądź ich zmodyfikowania. Model reprezentuje obiekt z polami definiującymi jego atrybuty. Z reguły odwzorowuje on na jedną tabelę w bazie danych. Dzięki mapowaniu obiektowo-relacyjnemu pozwala na bezpieczne operowanie zasobami z bazy danych, bez użycia zapytań SQL. Po każdym zmodyfikowaniu modelu należy zaktualizować bazę danych poprzez wykonanie migracji. Operacja ta odwzorowuje elementy modeli na tabele w relacyjnej bazie danych. W Django szablony odpowiadają za generowanie stron HTML aplikacji internetowej oraz wyświetlanie danych otrzymanych z widoku. Wygenerowany szablon przekazywany jest do widoku, który zwraca kompletną zawartość strony do przeglądarki klienta. Na rysunku 2 przedstawiony zo-

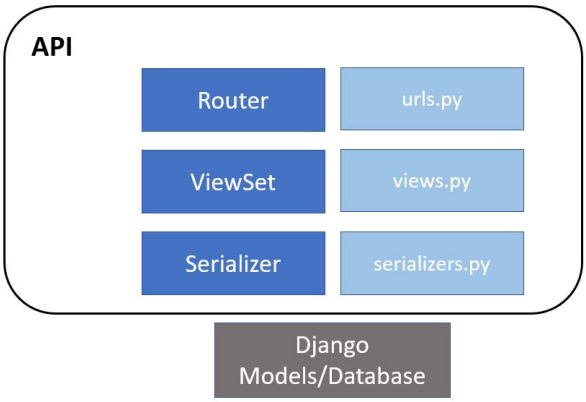
stał schemat działania aplikacji stworzonej za pomocą Django. [4]



Rysunek 2: Schemat działania aplikacji internetowej stworzonej z użyciem frameworku Django.

3.1.5 Django REST Framework

W związku z tym, że typowe podejście Django nie definiuje REST API, z którego mogą korzystać zewnętrzne aplikacje klienckie, zdecydowano o zastosowaniu nieco innego podejścia. Pakiet Django REST Framework współpracuje z typowymi modelami Django w celu stworzenia wydajnego i elastycznego REST API. Na rysunku 3 można zaobserwować podział interfejsu na trzy warstwy: Serializer, ViewSet i Router.



Rysunek 3: Interfejs Django REST Framework.

Modele Django reprezentują obiekty przechowywane w bazie danych, jednak interfejs API będzie musiał przesyłać informacje w mniej złożonej strukturze. Modele muszą zostać przetłumaczone na taki format, który może być przekazywany przez interfejs API. Serializator wykonuje to tłumaczenie w dwie strony. Przetwarza model na słownik w języku Python, a ten następnie jest konwertowany do formatu wspieranego przez API, jak np. JSON czy XML. Zadaniem serializatorów jest również wykonywanie deserializacji, czyli przekształcanie otrzymanych

w zapytaniu HTTP danych na typy złożone – modele, po wcześniejszym sprawdzeniu ich poprawności. Po tej operacji rekordy mogą zostać dodane do bazy danych. Jeśli jednak dane nie przejdą pomyślnie procesu walidacji, zostanie zwrócony wyjątek informujący o błędzie.

W zbiorach widoków (ViewSet) definiowane są dostępne operacje. Widok odpowiedzialny jest za zrozumienie żądania oraz wygenerowanie poprawnego wyjścia, czyli przykładowo zwrócenie listy elementów z bazy. ViewSet łączy logikę wielu widoków powiązanych ze sobą w jednej klasie. Najpopularniejszym jego podtypem jest ModelViewSet, który ma wbudowane operacje takie jak *create, retrieve, update, destroy* i *list*. Funkcje te mogą być nadpisane poprzez inne działanie, jakie zostanie zdefiniowane przez programistę.

Ostatnią warstwą jest Router, który określa wzorce adresów URL definiujące ścieżki do zasobów. Zapewniają one spójny sposób na połączenie logiki widoku z adresem URL. Routery automatycznie mapują przychodzące od klienta żądanie na odpowiednią akcję w zestawie widoków. [5]

3.1.6 Baza danych PostgreSql

Oprogramowanie PostgreSQL jest jednym z najpopularniejszych otwartych systemów służących do zarządzania relacyjnymi bazami danych. Pozostałymi systemami, z którymi konkuruje, są przykładowo MySQL oraz SQLite. Relacyjna baza danych składa się z wielu tabel, które posiadają rekordy. Każdy z rekordów identyfikowany jest za pomocą unikalnego identyfikatora zwanego kluczem głównym. Tabele połączone są ze sobą relacjami. Sposób ten pozwala na uniknięcie redundancji oraz przeprowadzanie analizy z wykorzystanie wielu tabel. Można wyróżnić trzy rodzaje relacji pomiędzy tabelami: jeden do jeden, jeden do wielu oraz wiele do wielu. [6]

3.1.7 Biblioteka Hashids

Hashids jest to biblioteka przystosowana do wielu języków programowania. Jej zadaniem jest konwersja liczb całkowitych na ciągi znaków po to, aby zwiększyć bezpieczeństwo przesyłanych informacji. W celu wykonania czynności polegającej na zastąpieniu liczby ciągiem znaków należy wprowadzić trzy kluczowe parametry:

- wartość soli - indywidualnie ustawiany ciąg znaków
- oczekiwana długość wygenerowanego ciągu
- alfabet - zbiór znaków, z których może zostać ciąg stworzony

Parametry te można dowolnie ustawiać. Zmiana któregokolwiek z nich spowoduje zmianę wygenerowanego łańcucha znaków. Alfabet musi mieć co najmniej 16 unikalnych znaków. Jest on przetasowywany na bazie soli. Ciąg soli nie może być dłuższy niż rozmiar podanego alfabetu. Twórcy zadbali o fakt, aby generowane ciągi znaków były przyjazne, szczególnie ma to znaczenie, gdy są umieszczane w adresach URL. Aby zapobiec wystąpieniu niecenzuralnych słów,

określone zostały reguły uniemożliwiające wystąpienie pewnych liter obok siebie. W Django można stworzyć pole typu Hashids będące kluczem głównym tabeli. Dzięki temu istnieje możliwość wyszukiwania elementów zarówno za pomocą identyfikatora id, jak i jego zakodowanej wersji. Istnieje możliwość kodowania oraz dekodowania ciągu znaków przy wykorzystaniu funkcji oferowanych przez bibliotekę. [7]

3.1.8 PyCharm

PyCharm to zintegrowane środowisko programistyczne stworzone w celu wsparcia procesu programowania z wykorzystaniem języka Python, stworzone przez firmę JetBrains. Obecnie stało się najbardziej rozbudowanym i popularnym rozwiązaniem do programowania z wykorzystaniem tego języka oraz bibliotek i frameworków w nim napisanych. Środowisko wspiera pisanie aplikacji internetowych w Django, dlatego zostało wybrane do stworzenia projektu. Zaletami tego rozwiązania są prosta edycja i analiza kodu, system podpowiadania składni oraz inspekcji kodu, wbudowany debugger służący do weryfikacji poprawności wprowadzonego kodu oraz zintegrowany systemu kontroli wersji. Dodatkowo oferuje rozbudowane narzędzia do nawigacji w kodzie projektu.

3.2 Technologie internetowe

Poniższy podrozdział zawiera opisy technologii wykorzystanych do stworzenia aplikacji internetowej.

3.2.1 TypeScript

TypeScript jest językiem programowania, stworzonym przez programistów Microsoft, którego składnia jest identyczna jak w JavaScript oraz stanowi rozwinięcie tego języka o dodatkowe funkcje, jednocześnie poprawiając jego słabe strony. JavaScript jest skryptowym językiem programowania, który najczęściej znajduje zastosowanie w aplikacjach internetowych. Odpowiada za zmianę treści w zależności od sytuacji, dzięki temu strony są interaktywne, a nie tylko wyświetlają statyczne informacje. Pozwala na dynamiczne kreowanie treści na stronie internetowej. Język TypeScript jest transpilowany do kodu JavaScriptu. Główną cechą rozszerzającą jest wprowadzenie typowania statycznego, oznacza to, że zmienne oraz argumenty funkcji mają zdefiniowane typy danych. Mogą to być typy proste, ale również złożone klasy definiowane przez programistę. Podejście to zmniejsza prawdopodobieństwo popełnienia błędu związanego z otrzymaniem zmiennej innego typu niż przewidywano. Tego typu błędy pojawią się już na etapie uruchamiania projektu. TypeScript zmienia również podejście do obiektowości, pozwalając na definiowanie klas, interfejsów, typów generycznych, importowanie modułów oraz wiele innych funkcji związanych z ideą obiektowości. [8]

3.2.2 HTML

HTML jest językiem znaczników przeznaczonym do tworzenia stron internetowych. Opisuje semantycznie strukturę elementów definiujących zawartość witryny. Struktura pliku opiera się na znacznikach oraz ich atrybutach. Znaczniki mogą definiować takie elementy jak obrazy, przyciski, interaktywne formularze oraz wiele innych obiektów, które można znaleźć na każdej stronie internetowej. Zasięg elementu jest definiowany przez parę tagów: początkowy oraz końcowy, które są zapisane w nawiasach kątowych. Zawartość, czyli np. tekst, umieszczana jest pomiędzy tymi znacznikami. Tego typu komponenty mogą być zagnieżdżone względem siebie, czyli element nadrzędny może posiadać dzieci. Istnieje również drugi rodzaj elementów, które nie wymagają znacznika zamykającego, jak np. <img>. W ten sposób definiują, że nie zezwalają na umieszczanie wewnątrz tekstu ani żadnego innego znacznika. W tagu początkowym mogą znajdować się atrybuty, czyli informacje opisujące go, jak np. unikalny identyfikator *id* elementu, czy atrybuty *style* oraz *class* używane do przypisania właściwości prezentacyjnych. Występują one w postaci par klucz-wartość, oddzielone operatorem „=", ale również mogą występować jako pojedyncze wartości. Przeglądarka otrzymuje dokument HTML od serwera, a następnie renderuje go do multimedialnej strony internetowej. HTML jest ciągle rozwijany. Obecnie najnowszą wersją jest HTML5. W tej wersji wprowadzone zostały nowe funkcjonalności jak np. nowe tagi, nowe typy pola *input* oraz dodatkowe atrybuty elementów formularzy. [9]

3.2.3 CSS, SASS, SCSS

CSS (ang. Cascading Style Sheets) [10], czyli kaskadowe arkusze stylu stosowane są do definiowania opisu prezentacji stron internetowych. Za pomocą odpowiednich dyrektyw definiowane są style dla poszczególnych elementów dokumentu HTML. Przykładowo można w ten sposób zdefiniować wielkość, kolor, marginesy, odstępy oraz pozycję elementów na stronie, a także wiele innych cech odpowiadających za prezentację. Reguły takie definiowane są za pomocą selektora określającego element, którego ma dotyczyć oraz deklaracji, w której zawarte są nazwy właściwości oraz przyjęte dla nich wartości. Każda reguła wygląda w następujący sposób:

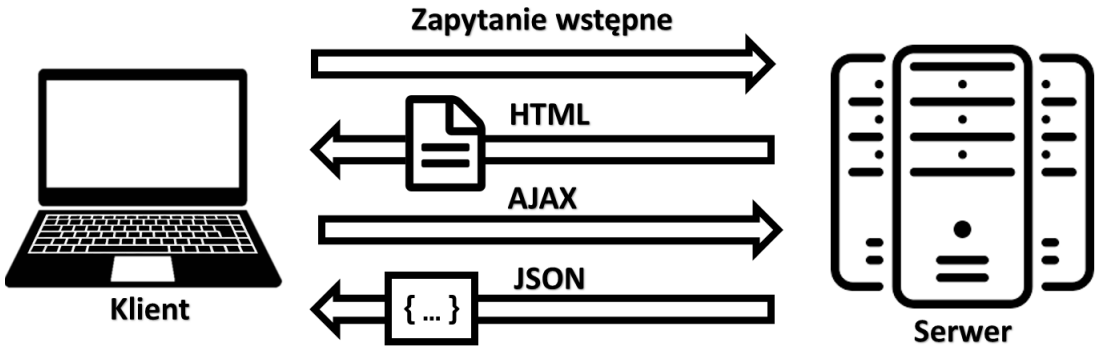
selektor {właściwość: wartość}

Ich hierarchia ustalana jest w sposób kaskadowy, gdy styl dla danego elementu zostanie zdefiniowany w wielu miejscach np. w głównym pliku aplikacji ze stylami, zewnętrznych arkuszach oraz w atrybucie stylu konkretnego elementu. Głównym zamysłem stworzenia arkuszy stylów było wydzielenie struktur odpowiedzialnych za wygląd aplikacji od samej jej struktury. Takie podejście sprawia, że kod jest dużo czytelniejszy oraz ułatwia wprowadzanie zmian wyglądu na wielu stronach jednocześnie, nie ingerując w kod poszczególnych dokumentów HTML, a modyfikując jedynie plik definiujący style. SASS [11] jest preprocesorowym językiem skryptowym, który jest kompilowany do zwykłego CSS, jednak oferuje wiele ulepszeń. Może on wykorzystywać

dwa typy składni. Pierwszy typowy dla SASS używający wcięć oraz drugi zwany SCSS, który używa identycznego formatowania, jak zwykły CSS, bazującego na nawiasach klamrowych oraz średnikach do oddzielania właściwości w blokach.

3.2.4 Angular 8

Angular jest frameworkiem rozwijanym przez zespół Google, przeznaczonym do tworzenia aplikacji internetowych typu „Single Page Application" używając kodu HTML oraz TypeScript. Termin jednostronicowa aplikacja oznacza, że strona w całości jest ładowana podczas jej pierwszego uruchomienia, czyli cały kod HTML i CSS jest pobierany z serwera. Kolejne akcje wykonywane na stronie, czyli interakcja z użytkownikiem, wyświetlanie nowych danych pobranych z serwera, odbywają się poprzez dynamiczne przepisywanie pojedynczych obiektów, zamiast przeładowywania całych stron. Za renderowanie poszczególnych elementów odpowiada przeglądarka. W efekcie interakcja z użytkownikiem oraz wyświetlanie jej efektów odbywa się znacznie szybciej, co poprawia wrażenia użytkownika podczas jej używania. Angular obecnie jest stabilną, dojrzałą oraz ciągle rozwijającą się technologią. Powstało wiele bibliotek oferujących gotowe rozwiązania, np. komponenty interfejsu użytkownika. Na rysunku 4 przedstawiony został schemat działania jednostronicowej aplikacji.



Rysunek 4: Schemat ładowania aplikacji typu „Single Page Application".

Aplikacja Angulara zbudowana jest z wielu składników zwanych komponentami, które tworzą strukturę drzewa z komponentami nadrzędnymi zwanymi rodzicami oraz podrzędnymi, które nazywane są dziećmi. Mogą się one ze sobą komunikować, przesyłając dane. Komponent jest najbardziej podstawowym elementem tworzącym interfejs użytkownika aplikacji. Aplikacja musi posiadać co najmniej jeden główny komponent, którym jest App. Składniki te zbudowane są z dwóch głównych części. Pierwszą jest klasa, napisana z użyciem języka TypeScript. Odpowiada ona za logikę komponentu aplikacji oraz przechowywanie danych. Bezpośrednio z nią powiązany jest drugi plik nazywany szablonem. Szablon odpowiada za definiowanie widoku wyświetlanego w przeglądarce. Dodatkowo komponent może zawierać plik przechowujący arkusze stylu, które są odpowiedzialne za jego wygląd. Składnia szablonów łączy kod HTML z rozszerzeniami Angulara jak np. zastosowanie dyrektyw strukturalnych. Każdy komponent

posiada swój cykl życia zawierający między innymi inicjalizację komponentu, zmiany w nim zachodzące, czy jego zniknięcie. Programista ma możliwość reagowania na zachodzące zdarzenia, dzięki metodom, które udostępnia Angular.

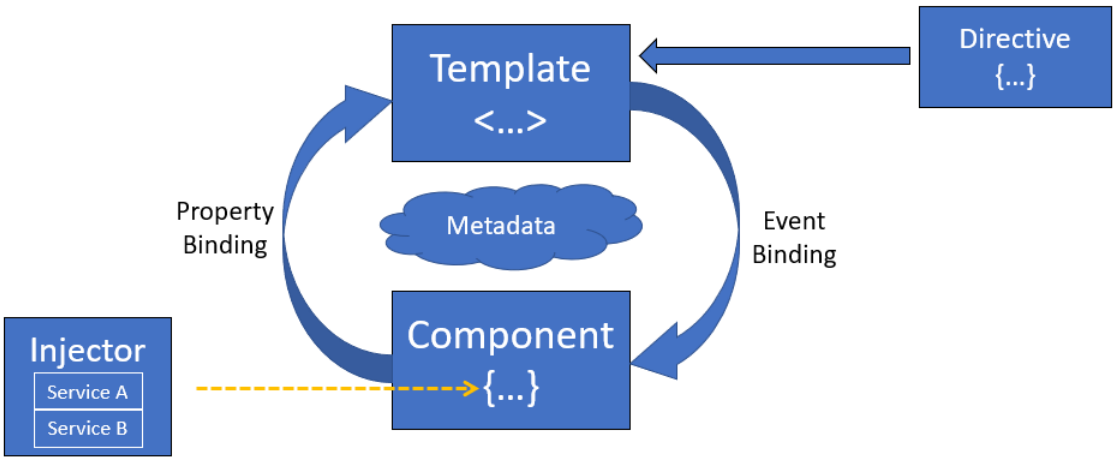
Komunikacja pomiędzy klasą komponentu a szablonem odbywa się poprzez bindowanie. Jest to proces, który pozwala aplikacjom wyświetlać wartości danych i reagować na działania użytkownika. W Angularze można wyróżnić dwa typy bindowania: bindowanie właściwości oraz bindowanie zdarzeń. Pierwszy typ umożliwia powiązanie pewnej zmiennej z komponentu z elementem w szablonie, co pozwala na automatyczne pobranie jej wartości podczas renderowania widoku. Drugi z wymienionych typów bindowania umożliwia nasłuchiwanie określonych zdarzeń, takich jak np. naciśnięcie przycisku oraz przypisuje takiemu zdarzeniu za pomocą funkcji pewną akcję w komponencie.

Dyrektywy służą do dodawania niestandardowego zachowania do szablonu. Można je podzielić na strukturalne i atrybutowe. Dyrektywy strukturalne zmieniają architekturę widoku, dodając logikę aplikacji do sposobu renderowania strony. Najpopularniejszymi dyrektywami są *ngFor* i *ngIf*. Pozwalają one na użycie pętli oraz wyrażenia warunkowego w pliku HTML. Dyrektywy atrybutów zmieniają wygląd lub zachowanie istniejącego elementu. Przykładem jest Dyrektywa *ngModel*, która implementuje dwukierunkowe wiązanie danych, co oznacza, że zmiany użytkownika są odzwierciedlane automatycznie w aplikacji.

Klasa komponentu posiada metadane, które informują o konfiguracji. Są to przykładowo: definicja selektora za pomocą którego można używać komponentu w plikach szablonów, powiązanie z plikiem szablonu, ścieżka do pliku zawierającego arkusze stylu, lista wstrzykiwanych zależności oraz inne.

Komponenty korzystają z serwisów, które zapewniają określoną funkcjonalność niezwiązaną bezpośrednio z widokami. Jest to klasa o dobrze określonym celu. Zwykle ich zadaniem jest komunikacja z serwerem w celu pobrania danych, ale mogą też mieć inne zadania jak np. zapisywanie logów do pliku, czy na konsolę. Serwisy są wstrzykiwane do komponentów jako zależności. Proces ten polega na przekazywaniu gotowych, utworzonych instancji obiektów udostępniających swoje metody i właściwości obiektom, które z nich korzystają. Takie podejście sprawia, że kod jest modułowy, wydajny oraz może być używany w wielu komponentach.

Na rysunku 5 przedstawiony został diagram prezentujący, w jaki sposób powiązane są podstawowe elementy składowe aplikacji napisanej z użyciem frameworku Angular.



Rysunek 5: Schemat działania aplikacji napisanej przy użyciu frameworku Angular.

Aplikacje Angulara są modułowe. Każda musi posiadać przynajmniej jeden moduł, zwany modułem głównym AppModule, który umożliwia ładowanie początkowe aplikacji, definiując kontekst kompilacji. Zadaniem tego bytu jest zbieranie wszystkich składowych aplikacji, czyli komponentów, serwisów oraz innych modułów w jednym miejscu. Aby komponent lub serwis mógł istnieć, musi zostać dodany w pliku modułu. W module importowane są wszystkie niezbędne do działania aplikacji składowe. Moduł może również eksportować własne funkcje na zewnątrz, aby mogły być używane w innych modułach.

Komponenty, serwisy oraz moduły są typowymi klasami, które używają dekoratorów. Dekorator określa typ oraz przypisuje specyficzne metadane, dzięki którym Angular wie, w jaki sposób ich używać. Przykładowo za pomocą metadanych serwisu przekazywana jest informacja do komponentu, że serwis ma być używany poprzez wstrzykiwanie zależności.

Ważną składową aplikacji Angulara są modele, które są zwykłymi klasami pozwalającymi na definiowanie własnych typów danych. Stosowane są do mapowania danych otrzymanych z serwera w postaci JSON na obiekty.

Kolejną ważną funkcją jest routing, umożliwiający tworzenie ścieżek służących do nawigacji pomiędzy widokami w aplikacji. Do tworzenia routingu używany jest moduł RoutingModule, który tworzy serwis odpowiadający za sterowanie nawigacją. Ścieżki definiowane są w tablicy tras, reprezentowanej przez typ Routes, jako obiekty posiadające atrybuty. Każdy taki obiekt jest zbudowany z adresu oraz komponentu, który ma zostać wyświetlony po przekierowaniu aplikacji na daną ścieżkę. Istnieje również możliwość definiowania parametrów w ścieżce, co jest niezbędne przy tworzeniu aplikacji internetowych. Obiekty tras mogą być rozbudowane o dodatkowe pola. Przykładowo można przesłać statyczne dane do komponentu za pomocą atrybutu data lub przekierować pod inny adres. Nawigowanie w aplikacji może odbywać się na różne sposoby, zarówno w klasach komponentu, jak i szablonach. W szablonach do przemieszczania używana jest dyrektywa routerLink. Do nawigacji w komponentach stosuje się klasę Router, która definiuje metody służące do nawigowania w aplikacji. Routing może być definiowany w

pliku modułu, ale dobrą praktyką, szczególnie podczas tworzenia dużych aplikacji, jest wydzielenie osobnego pliku odpowiedzialnego za nawigację. [12]

3.2.5 Angular CLI

Angular CLI to narzędzie wiersza poleceń, które pozwala w prosty sposób zarządzać projektem aplikacji. Narzędzie oferuje zbiór komend, które usprawniają proces tworzenia nowej aplikacji, uruchamiania jej czy budowania. Za pomocą jednej komendy można stworzyć gotowy do uruchomienia szablon projektu Angulara, zawierający wszystkie niezbędne pliki, pakiety czy zależności. Narzędzie dostarcza polecenia pozwalające na rozbudowę projektu, czyli utworzenie elementów aplikacji takich jak komponenty, serwisy, klasy, trasy oraz dodaje je automatycznie do pliku modułu. Konsola pozwala na uruchamianie aplikacji lokalnie podczas programowania, przebudowując automatycznie pewną część projektu po każdej zmianie w kodzie, jak również zbudowanie projektu gotowego do wdrożenia na serwer produkcyjny. Narzędzie jest również pomocne podczas uruchamiania testów aplikacji. [13]

3.2.6 Angular Material oraz MDB Angular

Poprawnie wykonany oraz ergonomiczne zaprojektowany interfejs użytkownika ma bardzo duże znacznie z punktu widzenia osoby, która będzie korzystać z aplikacji. Ważnym aspektem jest również wygląd wykonanej aplikacji, który powinien zachęcać do korzystania z niej. W przypadku tworzenia systemu proces projektowania i wykonywania interfejsu jest czasochłonny, a uzyskiwane efekty nie zawsze spełniają oczekiwania programistów i klientów. W tym celu powstało wiele zewnętrznych bibliotek oferujących gotowe komponenty interfejsu, które cechują się wysoką wydajnością, estetycznym oraz nowoczesnym wyglądem elementów, jakością wykonania oraz łatwością użycia. W projekcie zostały wykorzystane dwie tego typu biblioteki Angular Material [14] oraz MDB Angular [15]. Zapewniają one między innymi elementy umożliwiające nawigację, ustawienia położenia, obsługę formularzy, listę ikon, różne typy wyskakujących okien, zaawansowane tabele oraz wiele innych komponentów interfejsu użytkownika, które można spotkać na stronach internetowych. Ważną ich cechą jest wsparcie tworzenia aplikacji przeznaczonych na różnego rozmiaru urządzenia. Angular Material jest najbardziej popularną biblioteką, tworzoną przez zespół Google. Druga z bibliotek, czyli MDB oferuje wsparcie dla czterech najważniejszych frameworków do tworzenia aplikacji internetowych. Obie biblioteki posiadają dobrą dokumentację techniczną, co znacznie poprawia komfort ich użycia.

3.2.7 Ngx-charts

W dzisiejszych czasach wizualizacja danych jest niezwykle istotnym aspektem. Duża ilość informacji, które docierają do każdego człowieka, sprawią, że należy je prezentować w sposób odpowiedni. Dzięki wykorzystaniu biblioteki ngx-charts można w prosty sposób zaprezentować

duża ilość danych liczbowych, za pomocą interaktywnych wykresów. Baza ta oferuje dostęp do wielu różnego rodzaju typów wykresów oraz pozwala je dowolnie modyfikować. Dzięki gotowej bazie starannie przygotowanych wykresów można w szybki sposób uatrakcyjnić wykonywany przez siebie projekt oraz zaprezentować dane tak, aby były one łatwe w analizie. Dodatkowo produkt ten posiada dobrą dokumentacje, która znacząco ułatwia implementacje wykresów. [16]

3.2.8 Ngx-cookie-service

Cookies to niewielkie pakiety danych tymczasowo zapisywane na urządzeniu klienta podczas odwiedzania serwisów internetowych. Często pliki te są bardzo małe i nie przekraczają one kilku kilobajtów. Składają się z ciągu par zawierających nazwę oraz wartości rekordu. Są one oddzielane średnikiem. Za ich pomocą możliwe jest przechowywanie informacji przykładowo dotyczących konta lub uwierzytelniania. Dodatkowo z wykorzystaniem tych plików możliwe jest zapamiętanie preferencji użytkownika lub personalizowanie sposobu wyświetlania stron internetowych. Pakiet ngx-cookie-service udostępnia serwis pozwalający na pracę z plikami cookies. Zapewnia metody służące do zapisywania, pobierania, jak również usuwania danych zawartych w tych plikach. [17]

3.2.9 IntelliJ

Oprogramowanie IntelliJ zostało wykorzystane jako zintegrowane środowisko programistyczne w celu zwiększenia ergonomii oraz efektywności pracy. Jest to produkt firmy JetBrains, który został stworzony w celu ułatwiania pracy programistom. Posiada wsparcie dla wielu języków programowania oraz frameworków. Zapewnia on również obsługę dla projektów tworzonych w Angularze, dlatego jest przydatnym narzędziem wspierającym twórcę na każdym etapie programowania. Aplikacja ułatwia tworzenie projektu poprzez czytelny sposób wyświetlania stworzonego kodu, podpowiadanie składni, łatwy dostęp do części składowych projektu, możliwość tworzenia szablonów elementów aplikacji oraz wbudowany asystent wykrywający błędy składniowe. Ważnym aspektem jest również wbudowany moduł do obsługi narzędzia Git.

3.3 Technologie mobilne

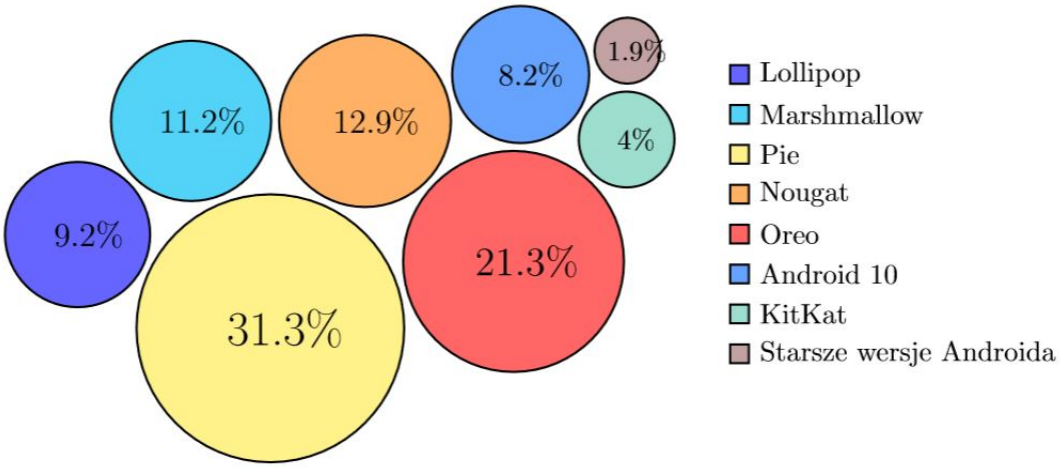
W poniższym podrozdziale opisano technologie, które posłużyły do stworzenia aplikacji mobilnej.

3.3.1 System operacyjny Android

Android jest najpopularniejszym systemem operacyjnym stosowanym w urządzeniach mobilnych. Ogromna popularność i obecność tego systemu na wielu urządzeniach sprawia, że aplikacje tworzone właśnie na tę platformę w łatwy sposób mogą zyskać wielu użytkowników. Oczywiście

popularność systemu sprawia, że konkurencja w przypadku aplikacji jest ogromna. Z danych statystycznych wynika, że w 2020 roku użytkownicy systemu android zdecydowali się pobrać wybraną przez siebie aplikację ponad 28 mld razy. Mieli oni wybór spośród 2,7 mln dostępnych pozycji. Ogromna część z nich oferuje darmowy dostęp do części projektu, a wymaga płatności jedynie za dodatkowe funkcje lub możliwości. Dzięki temu każdy z użytkowników może przetestować wiele aplikacji i wybrać tą, która najbardziej przypadnie mu do gustu. Otwiera to szerokie możliwości dla programistów, ponieważ ten typ dystrybucji sprawia, że każda dobrze zaplanowania i stworzona w sposób odpowiedni aplikacja może odnaleźć klientów.

System Android publikowany jest wersjach, które z biegiem lat były zastępowane kolejnymi, coraz bardziej dopracowanymi i dostosowanymi do nowoczesnych urządzeń systemami operacyjnymi. Pierwsza wersja opublikowana została 23 września 2008 roku i została nazwana Apple Pie. Kolejnymi wersjami były Banana Bread, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo, Pie. 22 sierpnia 2019 podjęta została decyzja o zmianie systematyki nazewnictwa systemów. Wcześniej używane nazwy słodyczy zastąpione zostały numerami wychodzących systemów. Po tej zmianie wypuszczone zostały Android 10 oraz 11. Na rysunku 6 przestawiono procentowy udział każdej z wersji systemu na urządzeniach mobilnych w 2020 roku.



Rysunek 6: Udział wersji systemu Android w kwietniu 2020.
Obrazek własny wykonany w LaTex; źródło danych: [18].

3.3.2 Kotlin

Do stworzenia aplikacji mobilnej użyty został język Kotlin. Jest to stosunkowo nowy język programowania działający na wirtualnej maszynie Javy. Stanowi alternatywę do Javy, ale jednocześnie dobrze współpracują razem w jednej aplikacji. W związku z tym programy napisane w Kotlinie dobrze działają z bibliotekami stworzonymi dla Javy. Kotlin uznawany jest za język ogólnego przeznaczenia, dlatego znajduje bardzo szerokie zastosowanie, szczególnie przy tworzeniu aplikacji mobilnych dla systemu Android i aplikacji serwerowych, ale również tworzeniu

aplikacji internetowych, czy też desktopowych, oraz wiele innych. Od 2019 roku, Google uznało język Kotlin za preferowany język to tworzenia aplikacji mobilnych na Androida. Obecne wiele aplikacji napisanych w Javie, przepisywanych jest na kod Kotlina. Język pozwala na tworzenie aplikacji zorientowanych na programowanie obiektowe, ale także na programowanie funkcyjne.

Kotlin jest językiem, który stosuje statyczne typowanie danych, czyli w momencie kompilowania kodu każda zmienna posiada jasno określony typ. Zaletą takiego podejścia jest bezpieczeństwo, ponieważ zwiększa liczbę błędów wykrytych na etapie kompilacji, a także poprawia optymalizację kodu. W Kotlinie nie ma obowiązku jawnie określać typu zmiennej, dlatego że jest on określany przez kompilator automatycznie na podstawie przypisanej wartości. Taka operacja nosi nazwę domniemania typu. Jej efektem jest zmniejszenie ilości niepotrzebnego kodu. Kotlin jest uznawany za zwięzły, w przeciwieństwie do Javy. Potrzebuje znacznie mniejszej ilości kodu, do wykonania tych samych operacji. W rezultacie zwiększa tempo produkcji oraz zmniejsza ilość błędów popełnianych podczas pisania aplikacji.

Twórcy zadbali o rozwiązanie problemów, które bardzo często pojawiały się podczas pisania aplikacji mobilnych w Javie. Głównym problemem, który pojawiał się, było nieoczekiwane zamknięcie aplikacji. Twórcy Kotlina postanowili wyeliminować zagrożenia powiązane z dostępem do referencji null. Język zmniejsza ryzyko pojawiania się wyjątku NullPointerException podczas działania aplikacji, ponieważ sytuacje takie wykrywa już na etapie kompilacji kodu. Stworzony został w tym celu system, który kontroluje czy zmiennej może zostać przypisana wartość null, czy nie. Utworzona w normalny sposób zmienna nie może mieć nigdy przypisanej wartości null. Każda taka próba spowoduje błąd na etapie kompilacji. Aby zmienna mogła mieć przypisaną wartość null, programista musi zadeklarować ją w jawny sposób, używając do tego operatora bezpiecznego wywołania „?”. [19]

3.3.3 Retrofit 2 oraz GsonConverter

Retrofit [20] to biblioteka zaprojektowana jako efektywny oraz bezpieczny REST klient, stworzona przez zespół Square. Zapewnia wsparcie podczas uwierzytelniania oraz obsługi żądań i odpowiedzi na etapie komunikacji klient-serwer. Komunikacja odbywa się z zastosowaniem REST API, które udostępniane jest przez serwer. Aby wykonywać zapytania, należy zdefiniować konwerter. Opisuje on sposób, w jaki powinna odbywać się serializacja oraz deserializacja przesyłanych danych. W tym celu zastosowano bibliotekę GsonConverter, która pozwala konwertować obiekty Javy na reprezentacje w postaci JSON oraz w przeciwnym kierunku. Biblioteka Retrofit zapewnia wsparcie dla aplikacji tworzonych na system Android w językach Java oraz Kotlin. Jej użycie znacznie zmniejsza ilość kodu potrzebną do zrealizowania komunikacji.

3.3.4 Gradle

Gradle jest elastycznym narzędziem, które służy do automatyzowania procesu budowania projektów napisanych w różnych językach. Znajduje szczególnie zastosowanie w dużych pro-

gramach, gdzie kontrolowanie tego procesu jest skomplikowane, ale w małych projektach jest również bardzo pomocny. Do głównych zadań należą kompilowanie projektu, zarządzanie zależnościami, importowanie zewnętrznych bibliotek, uruchamianie testów, budowanie gotowego projektu, wdrożenie aplikacji na serwer produkcyjny oraz wiele innych użytecznych funkcji. Konfiguracja zawarta jest w pliku build.gradle, który powstaje automatycznie podczas tworzenia projektu aplikacji w Android Studio. Jest to skrypt zwykle napisany w języku Groovy, zawierający listę komend, które są wykonywane. Większość środowisk programistycznych wspiera użycie Gradle, ale też istnieje możliwość zarządzania tym narzędziem za pomocą konsoli. Główną zaletą jest jego szybkość, ponieważ stosuje tak zwane kompilowanie przyrostowe, polegające na śledzeniu, które pliki uległy zmianie od poprzedniej kompilacji. Rekompilowane są tylko zmienione pliki oraz inne zależne od nich. W przypadku dużych projektów takie podejście może znacznie zmniejszyć czas potrzebny na kompilacje. [21]

3.3.5 Android Studio Code

Android Studio to zintegrowane środowisko programistyczne (ang. Integrated Development Environment), oparte na IntelliJ IDEA oraz stworzone przez JetBrains. Jest oficjalnym narzędziem do tworzenia aplikacji mobilnych na Androida. Platforma jest bardzo rozbudowana i pozwala na sprawne zarządzanie projektem. Oferuje duży zestaw narzędzi potrzebnych do tworzenia tego typu aplikacji oraz wsparcie dla języka Kotlin, po zainstalowaniu wtyczki przy pomocy menadżera wtyczek. Środowisko oferuje narzędzie do konwertowania kodu Javy na kod Kotlina, co jest szczególnie przydatne dla programistów posiadających umiejętności programowania w Javie. Konwerter jest także pomocnym narzędziem podczas migracji aplikacji z kodu Javy. Do automatyzacji budowania projektów używane jest narzędzie Gradle. Środowisko oferuje zaawansowany edytor układów, za pomocą którego programista może dodawać komponenty interfejsu użytkownika do widoków, przeciągając i upuszczając, edytować parametry oraz podejrzeć wygenerowany kod w formacie XML. Funkcja ta pozwala na szybki podgląd zbudowanego szkieletu komponentów bez potrzeby uruchamiania aplikacji.

Android Studio pozwala tworzyć wirtualne urządzenia, na którym uruchamia zbudowaną aplikację zamiast podłączania fizycznego produktu za pomocą kabla USB. Dodatkowo menedżer pozwala na utworzenie wielu urządzeń o różnych parametrach takich jak np. rozdzielczość, pojemność dysku, konkretna wersja systemu operacyjnego Android. Takie podejście pozwala na przetestowanie aplikacji na szerokiej gamie urządzeń, o zróżnicowanych parametrach bez konieczności użycia wielu urządzeń fizycznych.

Narzędzie jest zintegrowane z systemem kontroli wersji, co pozwala na śledzenie zmian w projekcie. Środowisko oferuje programistom wiele gotowych szablonów aktywności, co znacznie przyśpiesza proces ich tworzenia. Posiada również narzędzie Android Lint, którego zadaniem jest analizowanie kodu źródłowego i wykrywanie potencjalnych błędów w kodzie oraz informowanie programisty poprzez wyświetlane ostrzeżenia. Android Studio pozwala również na monitoro-

wanie w czasie rzeczywistym obciążenia procesora, pamięci oraz wydajności sieci urządzenia, na którym aplikacja jest uruchamiana. Android Studio to narzędzie, które znacznie upraszcza pracę programiście, oferując szeroką gamę użytecznych funkcjonalności. [22]

3.4 Kontrola wersji kodu - GIT

Git jest systemem kontroli wersji kodu. Zapewnia efektywną pracę nad dużymi projektami, rozwijanymi przez zespoły deweloperów. Oferuje możliwość tworzenia gałęzi (ang. branch), co pozwala na pracę przez wielu programistów równocześnie. Git oferuje algorytmy pozwalające na łączenie (ang. merge) zmian z różnych gałęzi oraz rozwiązywanie potencjalnych konfliktów. Każdy uczestnik biorący udział w tworzeniu aplikacji posiada swoją lokalną wersję repozytorium. W dowolnym momencie programista ma możliwość zapisania swojej obecnej wersji projektu. Główna gałąź w projekcie nazywana jest *master* i to właśnie na niej powinna znajdować się w pełni działająca, najnowsza wersja projektu. Git pozwala również na zarządzanie historią kodu, dlatego w każdym momencie możliwy jest powrót do wybranej poprzedniej wersji. Mimo iż projekt wykonywany w ramach niniejszej pracy był jednoosobowy, zdecydowano się na użycie narzędzia Git. Jest to dobra praktyka nawet w przypadku małych projektów, ponieważ pozwala na śledzenie zmian oraz postępów w wykonywaniu projektu, poprzez zapisywanie kolejnych wersji. Kod aplikacji został umieszczony w serwisie internetowym GitHub. [23]

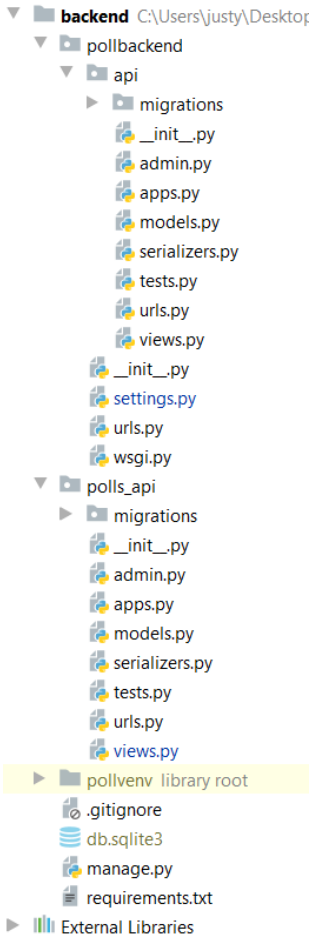
4 Architektura projektu

W niniejszym rozdziale przedstawiona została architektura poszczególnych aplikacji, które wykonano w ramach pracy. Kod źródłowy projektu znajduje się w publicznym repozytorium na GitHubie: `https://github.com/mwoloszyn/aghPoll`. W kolejnych podrozdziałach opisane zostały struktury aplikacji backendowej, internetowej oraz mobilnej.

4.1 Architektura backendu oraz bazy danych

Niniejszy projekt został wykonany zgodnie z modelem komunikacji klient-serwer. Aplikacja backendowa posłużyła do stworzenia REST API, które umożliwia korzystanie z zasobów znajdujących się na serwerze. W związku z tym, że system składa się z dwóch aplikacji klienckich, zastosowanie tego typu modelu okazało się kluczowe.

Do stworzenia aplikacji backendowej użyto pakietu Django REST Framework. Struktura utworzonego projektu składającego się z dwóch aplikacji została przedstawiona na rysunku 7. Zadaniem pierwszej jest obsługa rejestracji oraz autoryzacji użytkownika. Druga odpowiada za wszystkie operacje związane z formularzami.



Rysunek 7: Struktura projektu aplikacji backendowej.

W przypadku aplikacji stworzonych z wykorzystaniem Django kluczową rolę odrywa plik settings.py. Zawiera on całą konfigurację projektu. Znajdują się w nim między innymi informacje o wszystkich zainstalowanych w projekcie bibliotekach, konfiguracja bazy danych, lista hostów, z których możliwy jest dostęp do aplikacji, globalne ustawienia dotyczące dostępu do API, ustawienia CORS oraz wiele innych informacji niezbędnych do poprawnego działania projektu.

Django generuje gotowy moduł do zarządzania użytkownikami. Pierwszy element aplikacji to modele, które odwzorowywane są na tabele w bazie danych. Jako model użytkownika posłużyła klasa User udostępniana w pakiecie *contrib.auth*. Posiada ona wszystkie niezbędne pola do przechowywania informacji o użytkowniku.

Następnie w pliku views.py utworzono klasę UserSerializer, dziedziczącą po ModelSerializer. Klasy serializatorów są ściśle odwzorowywane na definicje modeli w Django. Domyślnie generują one zestaw atrybutów na podstawie modelu, jednak tworząc tę klasę, można zadeklarować tylko te, które mają być użyte do procesu serializacji obiektu. W projekcie użyte zostały pola *username, email* oraz *password* do przechowywania podstawowych informacji o użytkowniku. Użycie dziedziczenia sprawia, że klasa zawiera proste, domyślne implementacje metod *create()* oraz *update()*, które można dowolnie nadpisać własnym działaniem.

Widoki odpowiadają za logikę aplikacji. W pliku views.py utworzony został widok UserViewSet dziedziczący po klasie ModelViewSet. Podejście to sprawia, że automatycznie definiowane są funkcje *list(), retrieve(), create(), update(), partial_update()* oraz *destroy()*. Kolejna klasa w opisywanym pliku to CustomAuthToken dziedzicząca po ObtainAuthToken. Zdefiniowana w niej metoda POST służy do autoryzacji użytkownika, w celu sprawdzenia, czy jest on uprawniony do korzystania z API. Po otrzymaniu żądania zawierającego nazwę użytkownika oraz hasło, metoda sprawdza, czy taki użytkownik istnieje. W momencie poprawnej weryfikacji w odpowiedzi zwraca jego identyfikator, nazwę użytkownika oraz wygenerowany token. Wykorzystanie oferowanego przez Django modelu umożliwia korzystanie z funkcji takich jak generowanie tokenów, hashowanie haseł, dodawanie nowych użytkowników oraz wiele innych funkcjonalności niezbędnych do rejestracji oraz autoryzacji.

W pliku url.py zdefiniowane zostały wzorce adresów URL, pod które należy wysyłać żądania, aby skorzystać z API oferującego dostęp do zasobów w bazie danych. Poniżej przedstawione są stworzone w ramach tej części aplikacji punkty końcowe (ang. Endpoints):

- GET /api/users/ - zwraca listę wszystkich użytkowników.
- POST /api/users/ - dodaje nowego użytkownika.
- POST /api/auth/ - sprawdza czy istnieje użytkownik o podanej nazwie oraz haśle. W przypadku powodzenia generuje token oraz zwraca go w odpowiedzi wraz z nazwą użytkownika oraz jego identyfikatorem.

W celu poprawnego działania aplikacji nazwanej polls_api utworzono siedem modeli. Są to klasy, które mają zdefiniowane pola wraz z typami, jakie reprezentują. Istnieje również

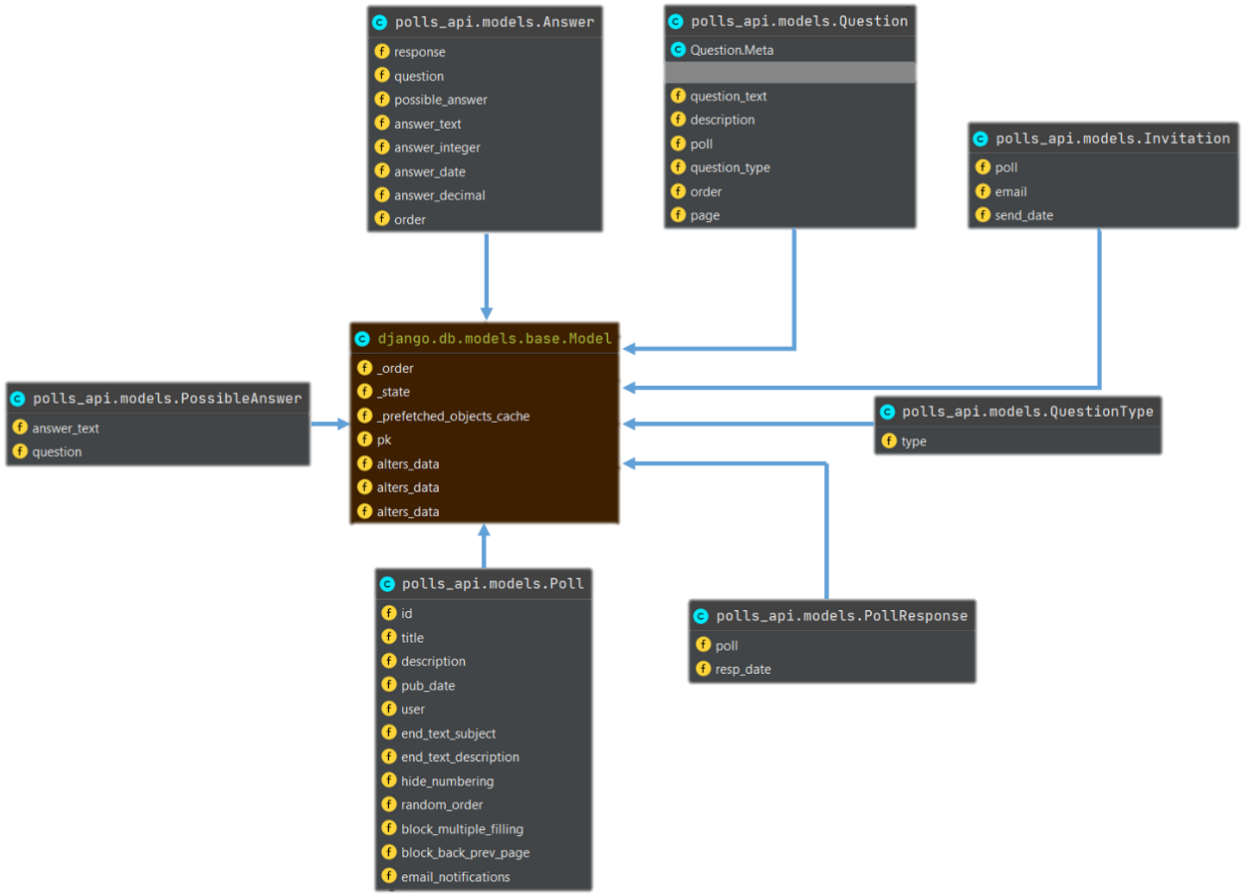
możliwość definiowania kluczy obcych powalających na odwołania do innych modeli. Skrypt 1 prezentuje fragment kodu definiujący model Answer.

```
class Answer(models.Model):
    response = models.ForeignKey('PollResponse', on_delete=models.CASCADE, related_name='
    answers')
    question = models.ForeignKey('Question', on_delete=models.CASCADE)
    possible_answer = models.ForeignKey('PossibleAnswer', on_delete=models.CASCADE, null=True)
    answer_text = models.TextField(null=True)
    answer_integer = models.IntegerField(null=True)
    answer_date = models.DateTimeField(null=True)
    answer_decimal = models.FloatField(null=True)
    order = models.IntegerField(null=True)

    class Meta:
        ordering = ['order']
```

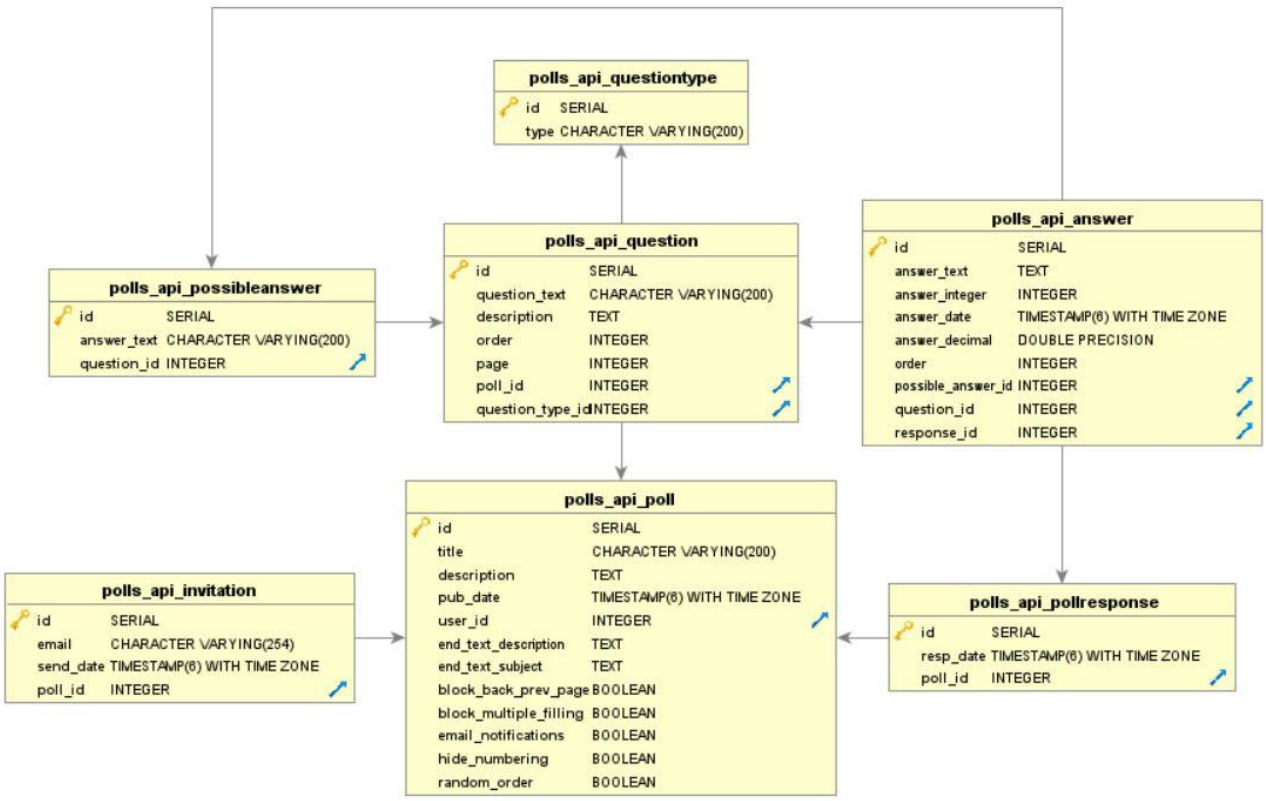
Skrypt 1: Przykładowy model Answer.

W celu zobrazowania wszystkich modeli utworzonych w ramach aplikacji na rysunku 8 przedstawiony został diagram klas. Wszystkie utworzone modele dziedziczą po klasie Model.



Rysunek 8: Diagram klas modeli.

Każda klasa jest odwzorowywana w postaci jednej tabeli w bazie danych. Po utworzeniu modeli oraz każdej zmianie konieczna jest ponowna migracja bazy danych, w celu jej zaktualizowania. Przykładem modyfikacji są dodanie lub usunięcie atrybutów, powstanie nowych modeli oraz usunięcie istniejących. Django oferuje gotowe komendy, które uruchamiają migracje oraz automatycznie zarządzają schematem bazy danych. W projekcie tworzony jest katalog *migrations*, w którym można sprawdzić, jakie zmiany nastąpiły w ramach poszczególnych migracji. Po wykonaniu wyżej opisanego procesu, w bazie danych PostgreSql utworzone zostały wszystkie tabele zdefiniowane przez modele Django w pliku models.py. Na rysunku 9 przedstawiony został diagram związków encji (ERD, ang. Entity-Relationship Diagram). Model ten służy do zobrazowania obiektów oraz związków pomiędzy nimi.



Rysunek 9: Diagram ERD bazy danych.

Wygenerowana baza danych zbudowana jest z następujących tabel. Każda z nich zawiera klucz główny, który generowany jest automatycznie:

- Poll - przechowuje informacje opisujące ankietę, takie jak tytuł (title), opis (description), data utworzenia (pub_date), identyfikator użytkownika (user), który ją utworzył, komunikat końcowy (end_text_subject), opis końcowy (end_text_description), a także ustawienia dotyczące ukrywania numeracji (hide_numbering), losowej kolejności pytań (random_order), blokowania wielokrotnego wypełniana (block_multiple_filling), blokowania powrotu do poprzedniej strony (block_back_prev_page) oraz powiadomień e-mail

(email_notifications).

- QuestionType - przechowuje informacje na temat wszystkich dostępnych typów pytań. Zawiera kolumnę tekstową definiującą typ (type).
- Question - przechowuje informacje o pytaniu takie jak tekst pytania (question_text), jego opis (description), numer porządkowy w ankiecie (order), stronę formularza, do której jest przypisany (page), klucz obcy informujący do jakiej ankiety przynależy pytanie (poll_id), oraz klucz obcy opisujący typ pytania (question_type).
- PossibleAnswer - reprezentuje odpowiedzi na pytania, które posiadają warianty do wyboru lub uszeregowania. Przechowuje tekst zawarty w odpowiedzi (answer_text) oraz klucz obcy do tabeli Question informujący o przynależności do konkretnego pytania.
- PollResponse - reprezentuje odpowiedź dotyczącą całej ankiety. Zawiera kolumny opisujące datę odpowiedzi (resp_date) oraz klucz obcy do tabeli Poll, definiujący przynależność odpowiedzi do ankiety (poll_id).
- Answer - reprezentuje odpowiedź na konkretne pytanie. Zawiera kolumny opisujące informacje zwrotne w formie tekstu(answer_text), liczby całkowitej (answer_integer), daty (answer_date), liczby zmiennoprzecinkowej (answer_decimal), liczby całkowej definiującej kolejność wybranej odpowiedzi (order), klucz obcy do tabeli PossibleAnswer (possible_answer_id), klucz obcy do tabeli Question (question_id) oraz klucz obcy do Response (response_id).
- Invitation - reprezentuje informacje o wysłanych zaproszeniach. Zawiera kolumny definiujące adres email (email), datę wysłania (send_date) oraz klucz obcy do ankiety, do której wypełnienia zaproszenie zostało wysłane (poll_id).

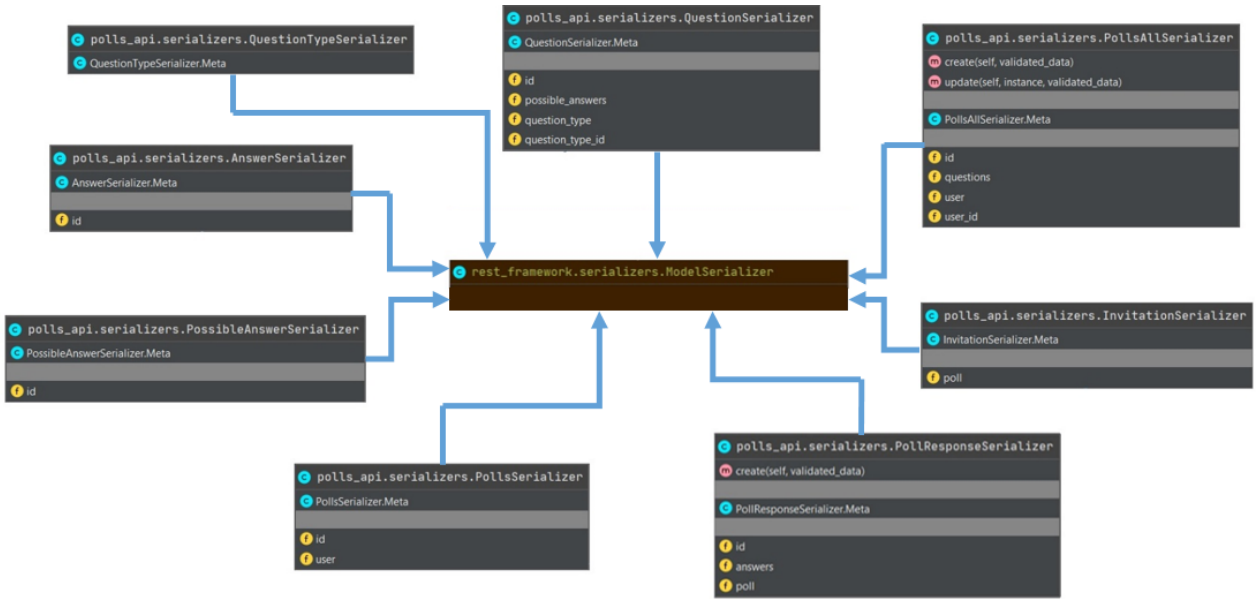
Kolejnym niezbędnym elementem do utworzenia API jest zdefiniowanie serializatorów dla poszczególnych modeli. Znajdują się one w pliku serializer.py i są podobnie zbudowane. Odpowiadają one za serializację oraz deserializację przesyłanych danych. Skrypt 2 prezentuje przykładowy kod tworzący klasę QuestionSerializer. Wewnątrz klasy Meta zdefiniowany jest model oraz pola, które mają być zwracane. Aby możliwe było generowanie pól będących kluczami obcymi, należy stworzyć obiekty ich serializatorów, a następnie dodać je jako parametry w polu *fields*. Podobna sytuacja ma miejsce podczas definiowania parametru identyfikatora, aby mógł zostać zwracany należy stworzyć zmienną typu Integer, a następnie dodać ją do pola *fields*.

```
1 class QuestionSerializer(serializers.ModelSerializer):
2     id = serializers.IntegerField(required=False)
3     possible_answers = PossibleAnswerSerializer(many=True)
4     question_type = QuestionTypeSerializer(many=False, read_only=True)
5     question_type_id = serializers.IntegerField(write_only=True)
```

```
6       class Meta:
7           model = Question
8           fields = ('id', 'question_text', 'description', 'possible_answers',
9           'question_type', 'question_type_id', 'order', 'page')
```

Skrypt 2: Przykładowa klasa QuestionSerializer.

Na rysunku 10 przedstawiony został diagram wygenerowany dla klas serializatorów. Każda klasa modelu posiada odpowiadającą jej klasę serializatora. Istnieje możliwość nadpisania działania metod *create()* oraz *update()*, które pochodzą z klasy bazowej ModelSerializer. Podejście to zostało zastosowane w przypadku klas PollsAllSerializer oraz PollResponseSerializer, aby umożliwić dodawanie oraz edycję danych zagnieżdżonych. Klasyczne metody *create()* oraz *update()* tworzone są dla prostych modeli, nieposiadających kluczy obcych.



Rysunek 10: Model klas serializatorów.

Kolejnym etapem jest utworzenie klas typu ViewSet w pliku views.py. Dziedziczą one po klasie ModelViewSet, dlatego oferują standardowe metody potrzebne do obsługi takie jak *list(), retrieve(), create(), update(), partial_update()* oraz *destroy()*. Niektóre z nich zostały nadpisane w celu zmiany ich standardowego zachowania. Skrypt 3 przedstawia przykładową implementacje klasy typu ViewSet. Warto zwrócić uwagę na metodę *create()*, która służy do dodawania zaproszenia do bazy danych. Oczekiwanym zachowaniem w tej sytuacji jest nie tylko dodanie obiektu, ale również wysłanie wiadomości email do użytkowników określonych w ciele żądania. Do tego celu posłużyła funkcja *send_email* zawarta w pakiecie *django.core.mail*. Poczta jest realizowana poprzez użycie hosta SMTP. W tym celu założono konto pocztowe w serwisie gmail, z którego wysyłane są wiadomości. Wszystkie ustawienia, takie jak np. EMAIL_HOST, EMAIL_PORT, EMAIL_USER oraz EMAIL_HOST_PASSWORD zostały zawarte w pliku settings.py.

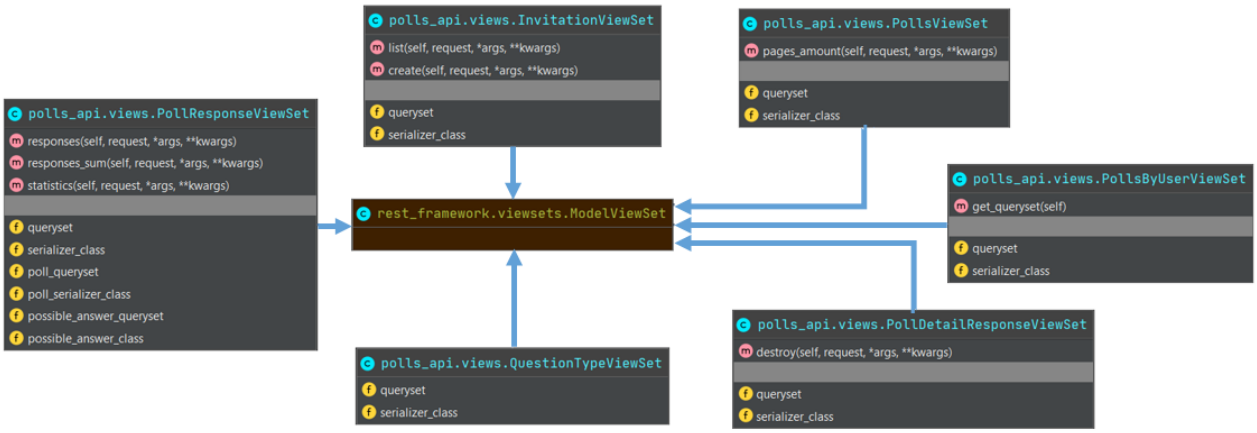
```python
class InvitationViewSet(viewsets.ModelViewSet):
    queryset = Invitation.objects.all()
    serializer_class = InvitationSerializer

    def list(self, request, *args, **kwargs):
        queryset = self.get_queryset().filter(poll_id=kwargs['pk'])
        serializer = InvitationSerializer(queryset, many=True)
        return Response(serializer.data)

    def create(self, request, *args, **kwargs):
        responses = []
        recipient_list = request.data['recipient_list']
        if len(recipient_list):
            for recipient in recipient_list:
                invitation = Invitation.objects.create(
                    poll_id=request.data['poll'],
                    email=recipient['email'])
                responses.append(model_to_dict(invitation))
        recipients = [recipient['email'] for recipient in recipient_list]
        send_email(subject=request.data['subject'],
                   content=request.data['content'],
                   recipient_list=recipients,
                   html_message=request.data['content'])
        return Response(responses)
```

Skrypt 3: Przykładowa klasa InvitationViewSet.

Na rysunku 11 przedstawiony został schemat obrazujący wszystkie powstałe w pliku views.py klasy typu ViewSet.



Rysunek 11: Model reprezentujący klasy typy ViewSet.

Poniżej przedstawione zostały wszystkie punkty końcowe aplikacji wraz metodami oraz opisem działania:

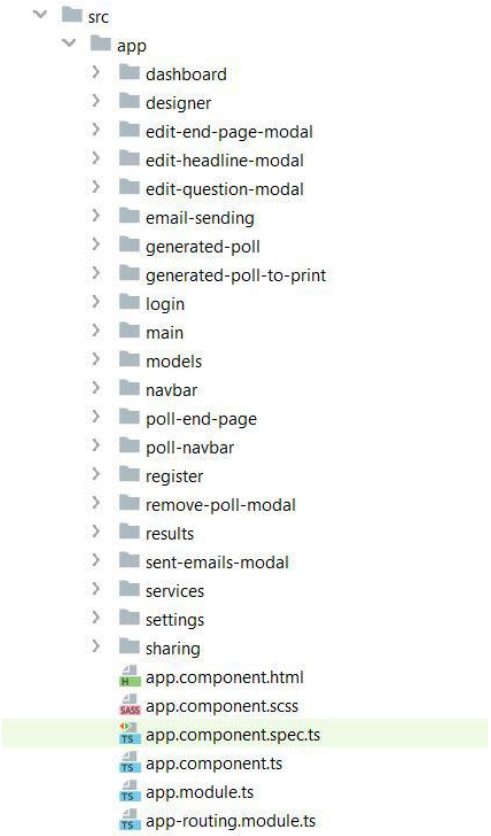
- GET /polls_api/polls/ - Zwraca listę wszystkich ankiet utworzonych w bazie danych.
- POST /polls_api/polls/ - Dodaje nową ankietę do bazy danych.

- GET /polls_api/polls?user_id=<int:user_id> - Zwraca listę ankiet utworzonych przez użytkownika o podanym jako parametr numerze user_id.
- GET /polls_api/polls/<str:pk>/ - Zwraca obiekt ankiety o podanym jako parametr pk identyfikatorze. Parametr pk może być w postaci liczbowej lub zakodowanej.
- DELETE /polls_api/polls/<str:pk>/ - Usuwa ankietę o podanym jako parametr pk identyfikatorze. Parametr pk może być w postaci liczbowej lub zakodowanej.
- PUT /polls_api/polls/<str:pk>/ - Modyfikuje ankietę o podanym jako parametr pk identyfikatorze. Parametr pk może być w postaci liczbowej lub zakodowanej.
- GET /polls_api/questions/types/ - Zwraca listę wszystkich typów pytań.
- GET /polls_api/polls/<str:pk>/pages_amount/ - Zwraca liczbę stron dla podanej jako parametr pk ankiety. Parametr pk może być w postaci liczbowej lub zakodowanej.
- GET /polls_api/polls/<str:pk>/responses/ - Zwraca listę wszystkich zebranych odpowiedzi dla ankiety o numerze id podanej jako parametr pk. Parametr pk może być identyfikatorem w postaci liczbowej lub zakodowanej.
- POST /polls_api/polls/<str:pk>/responses/ - Dodaje nową odpowiedź dla ankiety o numerze identyfikacyjnym podanym jako parametr pk. Parametr pk może być identyfikatorem w postaci liczbowej lub zakodowanej.
- GET /polls_api/polls/<str:pk>/responses/<int:resp_pk>/ - Zwraca odpowiedź (o id :resp_pk) dla ankiety o identyfikatorze pk. Parametr pk może być identyfikatorem w postaci liczbowej lub zakodowanej.
- DELETE /polls_api/polls/<str:pk>/responses/<int:resp_pk>/ - Usuwa odpowiedź (o id :resp_pk) dla ankiety o identyfikatorze pk. Parametr pk może być identyfikatorem w postaci liczbowej lub zakodowanej.
- GET /polls_api/polls/<int:pk>/statistics/ - Zwraca statyki odpowiedzi dla ankiety o identyfikatorze podanym jako parametr pk.
- GET /polls_api/polls/<int:pk>/responses/sum/ - Zwraca sumę odpowiedzi dla ankiedy o podanym jako paramtr pk identyfikatorze.
- GET /polls_api/polls/<int:pk>/invitations/ - Zwraca listę wysłanych zaproszeń poprzez wiadomości email dla ankiety o numerze identyfikacyjnym podanym jako parametr pk.
- POST /polls_api/polls/<int:pk>/invitations/ - Dodaje zaproszenie dla ankiety o numerze identyfikacyjnym podanym jako parametr pk.

4.2 Architektura frontendu – aplikacja internetowa

Do utworzenia aplikacji internetowej posłużył framework Angular. Stanowi ona niezależną aplikację korzystającą z REST API, które udostępniane jest przez aplikację serwerową. Projekt składa się z dwudziestu komponentów, z których każdy odpowiada za pewną zdefiniowaną funkcjonalność. Założeniem tego rozwiązania było stworzenie intuicyjnego oraz prostego w obsłudze interfejsu użytkownika.

W niniejszym podrozdziale opisana została implementacja poszczególnych funkcji, które dostarcza aplikacja. Na rysunku 12 przedstawiona została struktura katalogów utworzonych w ramach projektu.



Rysunek 12: Struktura aplikacji internetowej.

Każdy komponent zdefiniowany jest przy użyciu szablonu, klasy definiującej komponent oraz pliku zawierającego arkusze stylów. Szablon odpowiada za wyświetlanie interfejsu użytkownika. Zadaniem klasy komponentu jest definiowanie logiki, w tym również pobieranie danych z serwera. Poniżej przedstawiono listę stworzonych komponentów wraz z ich opisem:

- AppComponent – główny komponent aplikacji.
- NavbarComponent – wyświetlanie górnego paska nawigacji, na którym znajduje się logo projektu oraz w zależności od tego czy użytkownik jest zalogowany, wyświetlanie przycisków „Zaloguj”, „Wyloguj” oraz „Lista ankiet”.

- MainComponent – wyświetla pierwszy widok pojawiający się po uruchomieniu aplikacji oraz zawiera komponent Register.
- RegisterComponent – wyświetlanie formularza do rejestracji użytkownika, walidacja wprowadzonych do niego danych oraz wysłanie żądania o dodanie użytkownika do bazy danych. Po prawidłowym wypełnieniu wszystkich pól oraz kliknięciu przycisku „Zarejestruj", następuje przejście do komponentu Login.
- LoginComponent – wyświetlanie widoku zawierającego formularz logowania oraz walidacja wprowadzanych danych. Po prawidłowym wypełnieniu formularza oraz kliknięciu przycisku „Zaloguj", jego zawartość przesyłana jest do aplikacji serwerowej. Zwracany token autoryzujący użytkownika, zapisywany jest w LocalStorage. Kolejno następuje przekierowanie do strony zawierającej komponent Dashboard.
- DashboardComponent – pobieranie listy ankiet z serwera oraz wyświetlanie grafik prezentujących wszystkie dostępne formularze zalogowanego użytkownika. Wyświetlanie formularza służącego do dodawania nowej ankiety po kliknięciu przycisku „Dodaj ankietę". Po poprawnym wypełnieniu pola z nazwą następuje wysłanie żądania do serwera w celu utworzenia nowego formularza oraz przekierowanie do komponentu Designer. Z wykorzystaniem komponentu Dashboard możliwe jest przejście do edycji (komponent Designer), wyników (komponent Results), podglądu (komponent GeneratedPoll) oraz usunięcia formularza. Kolejną funkcjonalnością jest wyświetlanie okna dialogowego (komponent RemovePollModalComponent) potwierdzającego chęć usunięcia formularza. Po potwierdzeniu do serwera zastaje wysłane żądanie o usunięcie rekordu dotyczącego ankiety z bazy danych.
- RemovePollModalComponent – generuje okno dialogowe służące do potwierdzenia decyzji o usunięciu formularza. Zwraca do komponentu Dashboard informacje o wyborze.
- PollNavbarComponent – odpowiada za wyświetlanie paska, na którym znajdują się zakładki „pytania", „ustawienia", „edycja" oraz „wyniki". Umożliwia przełączanie się pomiędzy nimi, przekierowując do odpowiedniego komponentu.
- DesignerComponent – pobieranie danych z serwera dotyczących ankiety, jej liczby stron oraz dostępnych typów pytań. Wyświetlanie przycisków z typami pytań oraz dodanie logiki dotyczącej umieszczania poszczególnych typów pytań w formularzu. Utworzenie ich listy oraz zaimplementowanie możliwości zmiany kolejności. Po każdej zmianie obiektu ankiety, wysyłanie żądania do serwera o aktualizacje rekordów. Utworzenie przycisków służących do nawigacji pomiędzy kolejnymi stronami, tworzenia nowych oraz dodania im logiki. Wyświetlanie okien dialogowych w celu edycji pytania (komponent EditQuestionModal), nagłówka (komponent EditHeadlineModal) oraz strony końcowej (komponent EditEndPageModal), a także przekazywanie im jako parametr obiektu do edycji.

- EditQuestionModalComponent – wyświetlanie odpowiedniego okna dialogowego służącego do edycji pytania. W zależności od otrzymanego jako parametr typu pytania, generowana jest inna zawartość okna. Zaprojektowanie edycji, na które składa się możliwość zmiany zawartości istniejących pól oraz dodania nowych. W zależności od tego, czy zmiany mają być zapisane czy anulowane, zwracany jest zmodyfikowany lub niezmodyfikowany obiekt do komponentu rodzica.

- EditHeadlineModalComponent / EditEndPageModalComponent – odpowiada za edycję nagłówka ankiety. Z punktu widzenia wymiany informacji zostało to rozwiązane tak samo jak w przypadku komponentu EditQuestionModal, z tą różnicą, że zamiast edycji pytania następuje zmiana jego nagłówka oraz dla każdego formularza wyświetlane jest identyczne okno dialogowe. Analogiczne zachowanie ma miejsce w przypadku wyświetlania komponentu EditEndPageModal, jednak odnosi się on do edycji strony końcowej ankiety.

- SettingsComponent – pobiera ankietę wysyłając żądanie do serwera oraz odpowiada za wyświetlanie widoku zawierającego funkcje pozwalające na zmianę jej ustawień. Zmieniając dowolną wartość pojedynczego ustawienia za pomocą dwupołożeniowego przycisku, następuje wysłanie żądania do bazy w celu aktualizacji formularza.

- SharingComponent – wyświetla widok zawierający informacje o możliwych metodach udostępniania. Odpowiada za generowanie unikalnych kodów służących do wyszukiwania wybranej ankiety w aplikacji mobilnej oraz unikalnego linku, za pomocą którego możliwe jest bezpośrednie przekierowanie do ankiety. Kody generowane są z wykorzystaniem biblioteki Hashids. Po kliknięciu przycisku „Wyślij zaproszenia" otwiera komponent EmailSendingComponent oraz przekazuje do niego identyfikator ankiety jako parametr. Udostępnia możliwość zapisania formularza w formacie PDF.

- EmailSendingComponent – odpowiada za generowanie okna dialogowego zawierającego pola niezbędne do stworzenia wiadomości. Po poprawnym uzupełnieniu wszystkich pól formularza i kliknięciu przycisku „wyślij", wysyłane jest żądanie do serwera o dodanie nowego rekordu do tabeli zawierającej zaproszenia. Serwer automatycznie wysyła wiadomości z zaproszeniami do odbiorców. Wybierając opcję „Wysłane zaproszenia" otwiera okno dialogowe zawierające komponent SentEmailModalComponent, przekazując mu identyfikator ankiety jako parametr.

- SentEmailModalComponent – wysyła żądanie do serwera w celu pobrania listy wysłanych zaproszeń dotyczących danej ankiety. Wyświetla otrzymane wyniki w postaci tabeli.

- ResultsComponent – wysyła żądania do serwera, w celu otrzymania listy odpowiedzi, ich liczby oraz statystyk. Dane te wyświetlane są w zakładce „Indywidualne odpowiedzi" w postaci tabeli, z możliwością podziału na strony, sortowania rekordów po dowolnej kolumnie oraz usuwania wybranych elementów poprzez wysłanie żądania do serwera. Dla

każdego pytania w zakładce „Analiza wyników” z użyciem bibliotek generowane są odpowiednie wykresy oraz tabele, dostosowane do typu pytania. Dodatkową funkcjonalnością jest możliwość ukrycia lub pokazania wyników dla poszczególnych pytań.

- GeneratedPollComponent – wysyła żądania do serwera w celu pobrania formularza oraz liczby stron. Następnie wyświetla go w sposób wybrany przez użytkownika podczas tworzenia ankiety. Wygląd każdego pytania w ankiecie został dostosowany do zawartości, a jego prezentacja utrzymana jest w jednolicie opracowanej stylistyce. Po prawidłowym wypełnieniu formularza następuje wygenerowanie tablicy zawierającej odpowiedzi w formacie obsługiwanym przez serwer, wysłanie żądania o dodanie rekordów oraz przekierowanie do strony końcowej (komponent PollEndPageComponent). W komponencie zastosowana została walidacja wprowadzanych wartości. W przypadku wykrycia błędu niemożliwe jest przejście do kolejnej strony oraz wysłanie żądania do serwera. Komponent zawiera wszystkie informacje dotyczące wyglądu oraz ustawień, które zostały wybrane na etapie tworzenia.

- PollEndPageComponent – odbiera dane dotyczące ankiety od komponentu GeneratedPollComponent oraz odpowiada za wyświetlenie ekranu podziękowania.

- GeneratedPollToPrintComponent - wysyła żądanie do serwera w celu pobrania formularza. Następnie wyświetla go w sposób możliwy do wypełnienia w wersji papierowej, czyli nie uwzględniając podziału na strony oraz dostosowując pytania do możliwości wypełnienia ich w sposób tradycyjny.

Kolejnym niezbędnym elementem aplikacji napisanej z użyciem frameworka Angular są serwisy, które zostały zdefiniowane w katalogu *services*. W projekcie zostały utworzone trzy klasy serwisów:

- UserService – zawiera funkcje wykorzystywane w celu wysyłania żądań HTTP do serwera, dzięki którym możliwe jest zarządzanie kontem użytkownika.

- PollService – zawiera funkcje, które wykorzystywane są przez komponenty w celu wysłania żądań HTTP do serwera o dane dotyczące ankiet.

- HashIdService – zawiera parametry niezbędne do kodowania liczb z użyciem biblioteki HashIds.

Do wysyłania żądań służy interfejs klienta definiowany poprzez klasę HttpClient. Skrypt 4 prezentuje przykładową funkcję służącą do wysyłania żądania do serwera. Do klasy serwisu, w konstruktorze wstrzykiwany jest jako zależność obiekt typu HttpClient, na którym wywoływana jest jedna z metod HTTP. Odbywa się ono w sposób asynchroniczny, a zwracany obiekt jest typu Observable. Jako parametry wywołania podane są adres URL zasobu oraz nagłówki żądania.

```
  private headers = {
    headers: new HttpHeaders({
      'Content-Type': 'application/json',
      Authorization: 'Bearer ' + localStorage.getItem('token')
    })
  };
  constructor(private http: HttpClient) {}

  getPollWithId(pollId): Observable<any> {
    return this.http.get(this.url + 'polls/' + pollId + '/', this.headers);
  }
```

Skrypt 4: Przykładowa funkcja z serwisu służąca do pobierania formularza.

Katalog *models* zawiera klasy: Answer, Invitation, Poll, PossibleAnswer, Question oraz Response. Definiują one typy dla obiektów używanych w aplikacji. Otrzymane od serwera dane w postaci JSON są odwzorowywane na obiekty tych klas.

Do utworzenia ścieżek służących do nawigacji pomiędzy widokami w aplikacji posłużyła stworzona w pliku app-routing.module.ts tablica routingu. Skrypt 5 prezentuje fragment kodu definiujący wszystkie możliwe ścieżki aplikacji wraz z komponentami, które są otwierane podczas przekierowania:

```
const routes: Routes = [
  {path: '', redirectTo: 'rejestracja', pathMatch: 'full'},
  {path: 'logowanie', component: LoginComponent},
  {path: 'rejestracja', component: MainComponent},
  {path: 'tablica', component: DashboardComponent},
  {path: 'ankieta/:id/kreator', component: DesignerComponent},
  {path: 'ankieta/:id/ustawienia', component: SettingsComponent},
  {path: 'ankieta/:id/udostepnianie', component: SharingComponent},
  {path: 'ankieta/:id/wyniki', component: ResultsComponent},
  {path: 'ankieta/:id', component: GeneratedPollComponent},
  {path: 'ankieta/:id/druk', component: GeneratedPollToPrintComponent},
  {path: 'ankieta/:id/koniec', component: PollEndPageComponent},
  {path: '**', redirectTo: 'rejestracja'}
];
```

Skrypt 5: Kod definiujący tablicę routingu.

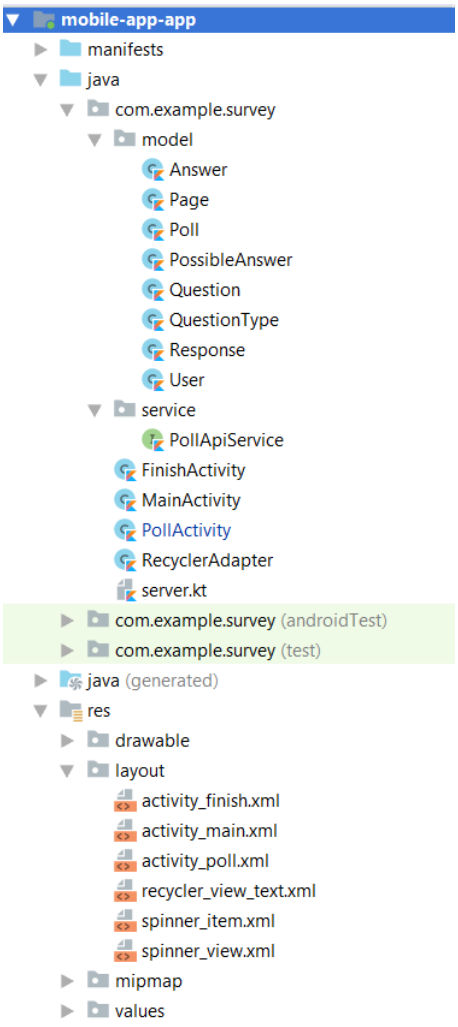
4.3 Architektura frontendu – aplikacja mobilna

Aplikacja mobilna służąca do wypełniania ankiet została zaimplementowana w języku Kotlin. Tego typu aplikacje zbudowane są z aktywności (ang. Activity), które są kluczową częścią projektów tworzonych dla systemu operacyjnego Android. Klasy te odpowiadają za tworzenie okna aplikacji, wypełnienie go wieloma elementami programu oraz interakcję z użytkownikiem. Z reguły reprezentują one pojedyncze okno aplikacji. Każda aktywność ma zdefiniowaną metodę *onCreate()*, która wywoływana jest podczas jej pierwszego tworzenia. Metoda ta odpowiada

między innymi za ładowanie pliku zawierającego *layout*. Jej działanie może zostać dowolnie nadpisane.

Każda aplikacja mobilna musi posiadać plik o nazwie AndroidManifest.xml w katalogu głównym projektu. Zawiera on wszystkie podstawowe informacje o aplikacji. Używany jest podczas kompilacji. Przykładowo znajdują się w nim informacje o wersji SDK, ustawieniach dostępu, definicji wszystkich aktywności, usług oraz intencji.

Nieodłącznym elementem aplikacji mobilnej są również pliki w formacie XML znajdujące się w folderze *layouts*, które definiują poszczególne widoki. Można je generować z użyciem przeznaczonego do tego edytora wizualnego w Android Studio Code lub wypełniając pliki źródłowe. W widoku zdefiniowany jest *layout*, który określa sposób umiejscowienia elementów na ekranie aplikacji. Przykładowo użycie klasy LinearLayout sprawi, że elementy wewnętrzne będą usytuowane jeden pod drugim. Wewnątrz mogą być definiowane przyciski, pola tekstowe, obrazy oraz wiele innych elementów, które można spotkać używając aplikacji mobilnych. Każda aktywność ma przypisany widok, który ładowany jest na ekranie aplikacji podczas jej tworzenia. Na rysunku 13 przedstawiony został schemat wykonanej aplikacji mobilnej.



Rysunek 13: Struktura aplikacji mobilnej.

W ramach projektu utworzone zostały trzy aktywności, z których każda odpowiada za definicje innego ekranu aplikacji.

MainActivity odpowiada za tworzenie ekranu głównego aplikacji. W powiązanym z nią pliku activity_main.xml zdefiniowane zostało logo, pole tekstowe służące do wprowadzenia identyfikatora ankiety oraz przycisk służący do wyszukania formularza. W klasie aktywności utworzona została metoda służąca do wysłania żądania do serwera w celu otrzymania obiektu zawierającego ankietę z pytaniami. Jeżeli proces ten zakończy się sukcesem następuje przejście do aktywności PollActivity. Przekierowanie odbywa się z zastosowaniem intencji (ang. Intent). Intencje są jednym z podstawowych komponentów aplikacji mobilnej. Ich mechanizm odpowiada za uruchamianie odpowiednich aktywności. Pozwala również na przesyłanie pomiędzy nimi obiektów. Skrypt 6 prezentuje fragment kodu służący do uruchomienia aktywności PollActivity, przesyłając do niej pobrany z serwera obiekt ankiety.

```
1 var pollActivity: Intent = Intent(applicationContext, PollActivity::class.java)
2 pollActivity.putExtra("poll", poll)
3 startActivity(pollActivity)
```

Skrypt 6: Definicja intencji.

Kolejną aktywnością utworzoną w projekcie jest PollActivity, która jest odpowiedzialna za wyświetlanie ekranu zawierającego wygenerowaną ankietę. W metodzie *onCreate()* odbierane są dane przekazywane z użyciem intencji. W związku z tym, że zawartość ankiety nie jest stała, wszystkie pola nie mogą być definiowane bezpośrednio w pliku XML. Tworzone są one w sposób dynamiczny w kodzie aktywności. Liczba stron formularza również jest zmienna, dlatego podczas zmiany strony lista pytań jest filtrowana po jej numerze oraz generowana jest nowa zawartość okna, a poprzednia czyszczona. W celu utworzenia widoku zawierającego listę pytań wykorzystana została metoda *createQuestions()*, której zadaniem jest iterowanie po wszystkich pytaniach, które mają zostać wyświetlone na danej stronie oraz dynamiczne ich tworzenie na podstawie typu pytania. W aktywności zaimplementowano metodę, która zlicza niewypełnione, jak również błędnie wypełnione pytania. Na tej podstawie uniemożliwia przejście do kolejnej strony lub wysłanie formularza. W takiej sytuacji pod każdym błędnie wypełnionym pytaniem generowane jest czerwone pole informujące o błędzie. Skrypt 7 prezentuje fragment kodu służący do wygenerowania pola tekstowego w sposób dynamiczny:

```
1 val questionDescription = TextView(this)
2 questionDescription.textSize = 20f
3 questionDescription.text = question.description
4 poll_list_layout.addView(questionDescription)
```

Skrypt 7: Tworzenie pola tekstowego w sposób dynamiczny.

Po poprawnym wypełnieniu wszystkich pytań możliwe jest wysłanie odpowiedzi. W tym celu w kodzie wypełniana jest tablica, w taki sposób, aby jej zawartość była zgodna z formatem,

jakiego oczekuje REST API. Następnie zostaje wysłane do serwera żądanie dodania danych. Za pomocą intencji uruchamiana jest trzecia aktywność zdefiniowana w projekcie.

FinishActivity utworzona została w celu wyświetlania ekranu końcowego zawierającego podziękowania za wypełnienie formularza. Poprzez wykorzystanie intencji przekazano obiekt ankiety w postaci parametru, aby możliwe było wyświetlenie tekstu pożegnalnego. Z tą aktywnością ściśle powiązany jest plik layoutu activity_finish.xml, w którym zdefiniowane zostały pola tekstowe, obraz zawierający logo oraz przycisk umożliwiający powrót do strony głównej aplikacji. W tym celu również został wykorzystany obiekt intencji.

W katalogu *model* stworzone zostały klasy reprezentujące poszczególne modele danych. Dzięki temu istnieje możliwość konwertowania otrzymywanych danych z formatu JSON na obiekty modeli. W katalogu *service* zdefiniowany został plik o nazwie PollApiService. W pliku tym określana jest struktura zapytań, które dostosowane są do formatu REST API. Do każdego z nich przypisany jest nagłówek oraz metoda HTTP wraz z adresem zasobu. Skrypt 8 przedstawia przykładową metodę, która służy do pobrania ankiety z serwera.

```
interface PollApiService {
    @Headers("Accept: application/json")
    @GET("polls/{id}")
    fun getPollById(@Path("id") id:String): Call<Poll>
}
```

Skrypt 8: Metoda definiująca strukturę zapytania.

Aby istniała możliwość komunikacji z serwerem stworzony został obiekt klienta, który posłużył do wykonania połączenia. Następnie z użyciem tego obiektu oraz wykonując metodę *getPollById* z pliku PollApiService zwracany jest obiekt typu Call. Skrypt 9 prezentuje metodę, w której tworzony jest obiekt klienta oraz wywołanie służące do wysłania żądania.

```
 private fun fetchPollWithId(id: String): Call<Poll> {
    val retrofit = Retrofit.Builder()
        .addConverterFactory(GsonConverterFactory.create())
        .baseUrl(address)
        .build()
    val pollApiService = retrofit.create(PollApiService::class.java)
    val myCall:Call<Poll> = pollApiService.getPollById(id)
    return myCall
 }
```

Skrypt 9: Funkcja tworząca klienta Retrofit.

Kolejnym krokiem jest wysłanie asynchronicznego żądania do serwera używając do tego metody *enqueue*. Jako argument przyjmuje ona wywołanie zwrotne. Wewnątrz tego wywołania zaimplementowane zostały dwie metody, które sprawdzają czy operacja przebiegła pomyślnie i w zależności od rezultatu wykonują odmienne akcje. Skrypt 10 prezentuje wywołanie metody *enqueue* na obiekcie typu Call.

```
fetchPollWithId(pollId).enqueue(object: Callback<Poll>{
    override fun onFailure(call: Call<Poll>, t: Throwable) {
        Log.e("ERROR", t.message.toString())
    }
    override fun onResponse(call: Call<Poll>, response: Response<Poll>) {
        val poll: Poll = response.body()!!
        val pollActivity: Intent = Intent(applicationContext, PollActivity::class.java)
        pollActivity.putExtra("poll", poll)
        startActivity(pollActivity)
    }
})
```

Skrypt 10: Wysłanie żądania za pomocą funkcji *enqueue*.

4.4 Testy aplikacji

W przypadku aplikacji backendowej nie zostały stworzone testy, które w sposób automatyczny weryfikowałyby poprawność wprowadzonego kodu. Spowodowane było to dostępnością narzędzi oferowanych do wysyłania żądań. Do testów wykorzystana została aplikacja Postman oraz narzędzie, które oferuje Django Rest Famework działające w przeglądarce. Posłużyły one do sprawdzenia, czy uzyskana wiadomość zwrotna z serwera jest zgodna z oczekiwaniami twórcy. W aplikacjach tworzone były zapytania z wykorzystaniem funkcji GET, które służyły do weryfikacji poprawności otrzymywanych wyników w formacie JSON oraz funkcji POST w celu sprawdzenia, czy przesłane obiekty dodają się w sposób poprawny do bazy danych. Projektowane były prawidłowe zapytania w celu weryfikacji prawidłowości działania REST API oraz zapytania z błędnym ciałem, których zadaniem było sprawdzenie zachowania aplikacji w momencie przesłania błędnych wartości.

Frontendowa część aplikacji testowana była w sposób manualny. Wszystkie możliwe konfiguracje, akcje czy też działania zostały sprawdzone. Testy te podzielone zostały na dwa etapy ze względu na różnice funkcjonalności w zależności od tego czy użytkownik jest zalogowany czy nie. W przypadku funkcjonalności dla użytkownika zalogowanego wykonano szereg testów dla funkcji związanych z tworzeniem formularzy, dodawaniem pytań, ich edycją, udostępnianiem stworzonych formularzy oraz analizą otrzymywanych wyników. Dla funkcjonalności przypisanych do użytkownika niezalogowanego, sprawdzano poprawność wypełnienia ankiet, wszelkie zabezpieczenia przed przesłaniem błędnie wypełnionego formularza oraz stronę wizualną prezentacji. Ze względu na szczegółowe sprawdzenie poprawności działania aplikacji nie było konieczności tworzenia testów automatycznych.

5 Prezentacja funkcjonalności aplikacji

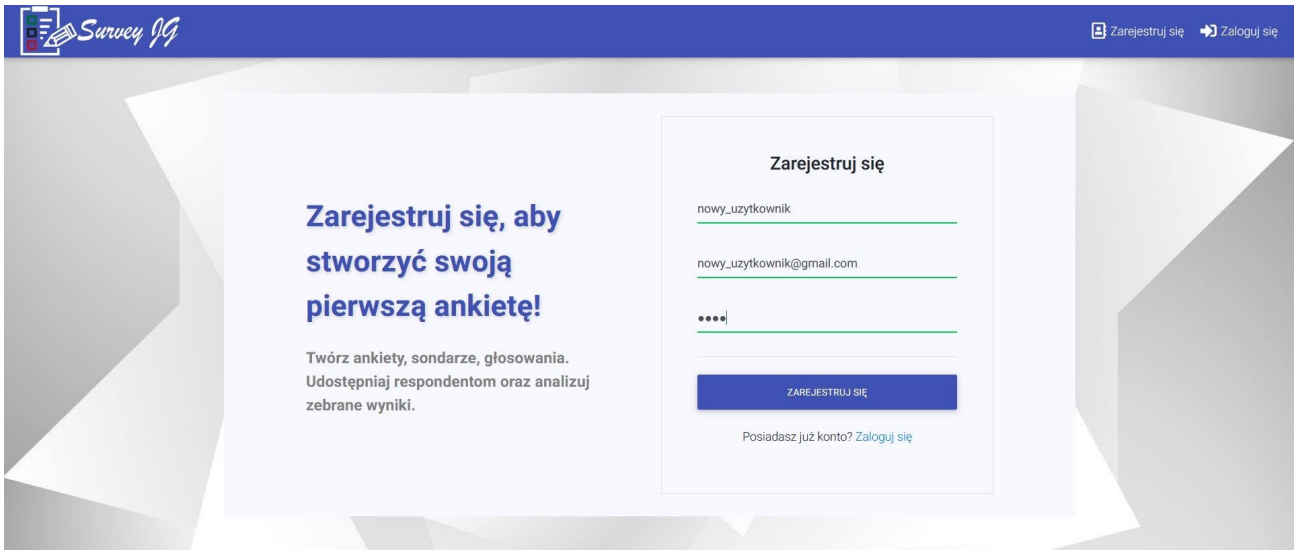
Założeniem całego projektu było stworzenie aplikacji w dwóch wersjach. Pierwsza z nich to aplikacja internetowa, do której wykorzystania niezbędna jest przeglądarka internetowa. Druga natomiast jest aplikacją mobilną. W zależności od wybranej opcji, funkcjonalności znacznie się różnią. Istotnie bardziej rozbudowana jest wersja internetowa, ponieważ jej zadaniem jest umożliwienie użytkownikowi tworzenia ankiet oraz zarządzania nimi. Wersja mobilna ma służyć głównie do wypełniania wcześniej przygotowanych formularzy.

5.1 Funkcjonalności aplikacji internetowej wymagające logowania

W niniejszym podrozdziale przedstawione zostały wszystkie funkcje, które oferuje system po zalogowaniu.

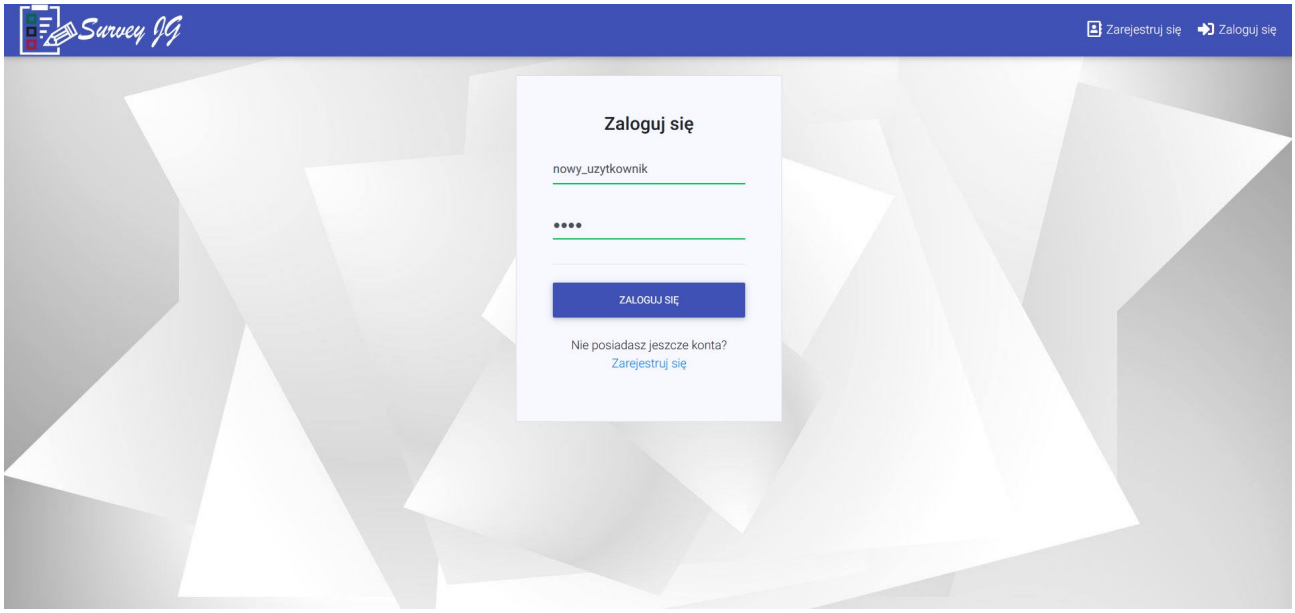
5.1.1 Rejestracja i logowanie

Pierwszym ekranem ukazującym się po uruchomieniu aplikacji jest widok z możliwością zarejestrowania się dla nowych użytkowników. Na stronie powitalnej widoczna jest informacja, która ma zachęcić użytkownika do skorzystania z aplikacji. W lewym górnym rogu znajduje się logo, które zostało dodane w obu wersjach aplikacji internetowej oraz mobilnej. Po prawej widoczne są dwa przyciski, które ułatwiają nawigację pomiędzy stronami, na których istnieje możliwość logowania i rejestracji użytkowników. Po prawidłowym wypełnieniu formularza rejestracji, na ekranie pojawi się komunikat, informujący o poprawnym dodaniu użytkownika oraz nastąpi przekierowanie do strony logowania. W sytuacji jego błędnego wypełnienia, źle zdefiniowane pola zostaną podkreślone kolorem czerwonym. Zdjęcie ekranu widoczne jest na rysunku 14.



Rysunek 14: Ekran główny z oknem służącym do rejestracji nowych użytkowników.

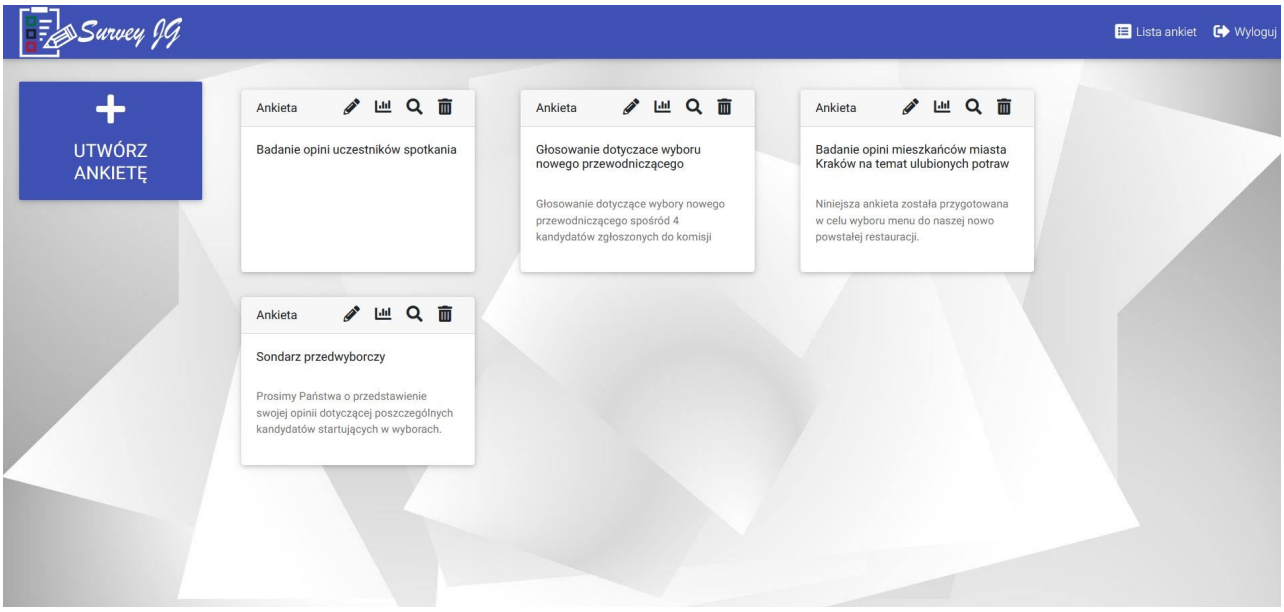
W przypadku zarejestrowanych wcześniej użytkowników istnieje możliwość przejścia do strony logowania. Można tego dokonać używając zlokalizowanego w prawej górnej części ekranu przycisku lub wykorzystując odnośnik znajdujący się w dolnej części okna służącego do rejestracji nowych użytkowników. W przypadku tego formularza również wykonywana jest walidacja danych, która sprawdza, czy wszystkie pola wypełnione są w sposób prawidłowy. Funkcja logowania jest przydatna w przypadku tego typu aplikacji, ponieważ umożliwia wgląd do listy wszystkich stworzonych przez siebie ankiet oraz pozwala na zarządzanie nimi. Widok ekranu logowania zaprezentowany jest na rysunku 15.



Rysunek 15: Widok ekranu z formularzem do logowania.

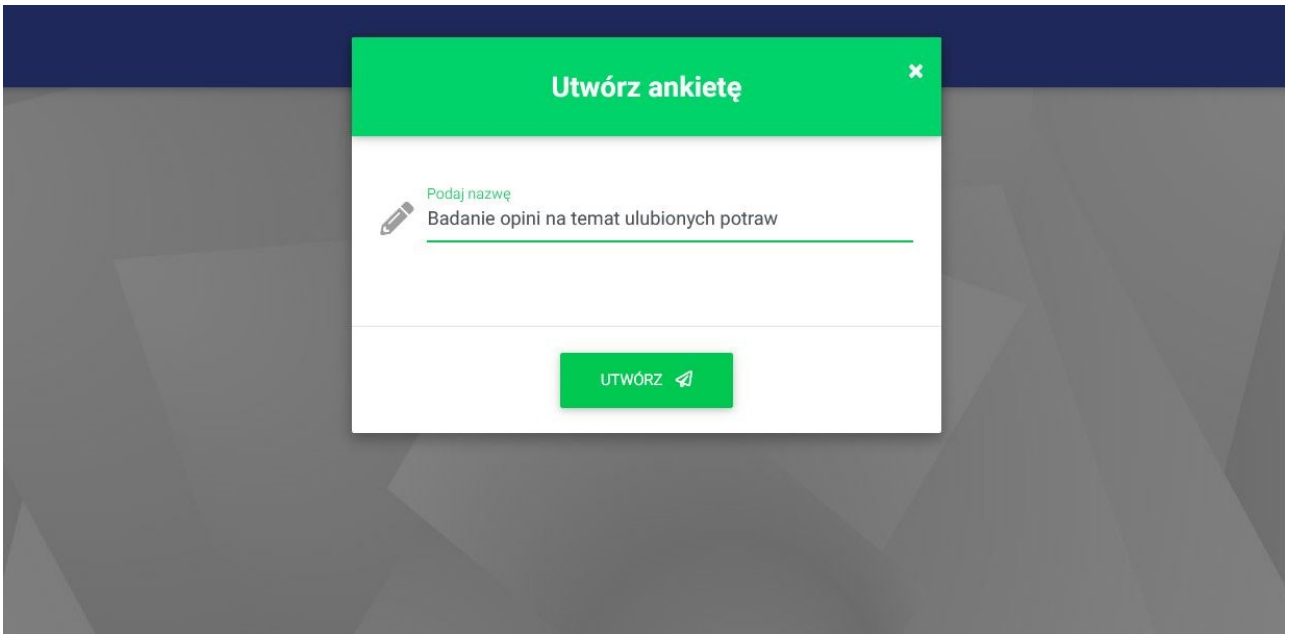
5.1.2 Tablica służąca do podglądu i zarządzania formularzami

Po zalogowaniu użytkownikowi ukazuje się główny ekran aplikacji, na którym widoczne są wszystkie wcześniej utworzone formularze. Po lewej stronie usytuowany został duży przycisk „utwórz ankietę", służący to tworzenia nowego formularza. W przypadku każdej widocznej ankiety, która prezentowana jest w postaci kart, widoczne są cztery ikony, które ułatwiają użytkownikowi szybką nawigację pomiędzy zbiorem funkcji. Poszczególne ikony odpowiadają kolejno za edycje formularza, wyświetlenie uzyskanych rezultatów, przejście do podglądu wygenerowanego dokumentu oraz możliwość jego usunięcia. Dodatkowo w celu łatwiejszego odnalezienia wyszukiwanej ankiety poniżej ikon zawarty został tytuł oraz krótki opis, jeśli został wprowadzony przez użytkownika. Widok strony głównej aplikacji jest umieszczony na rysunku 16.



Rysunek 16: Widok strony z podglądem wszystkich ankiet zalogowanego użytkownika.

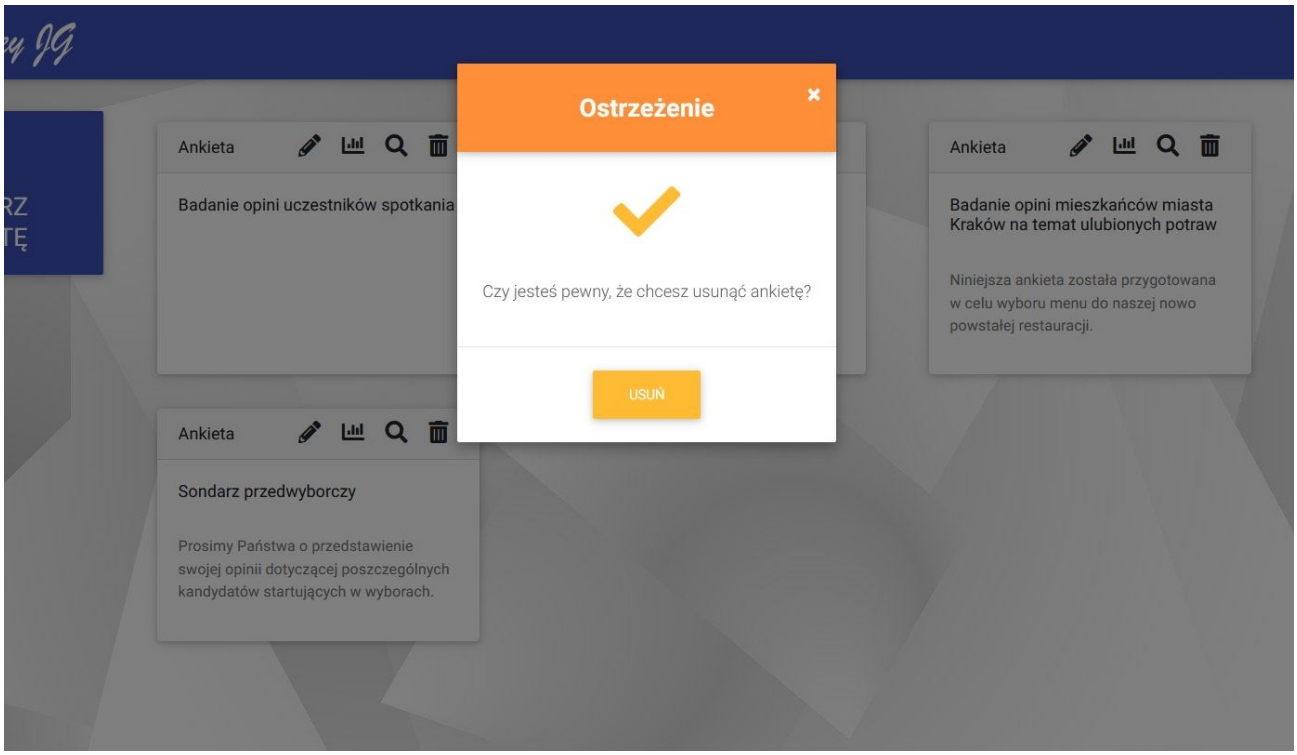
Po kliknięciu przycisku, służącego do tworzenia ankiety, pojawia się okno dialogowe, w którym użytkownik powinien wpisać nazwę nowego formularza oraz zatwierdzić jego utworzenie. Po akceptacji nastąpi automatyczne przekierowanie do edycji nowo powstałego projektu. Na rysunku 17 przedstawiono widok okna służącego do tworzenia nowej ankiety.



Rysunek 17: Okno do tworzenia nowej ankiety.

Z wykorzystaniem ikony przedstawiającej kosz, użytkownik może usunąć wybrany formularz w dowolnym momencie. Aby wyeliminować możliwość przypadkowego usunięcia ankiety, stworzone zostało specjalne okno dialogowe, za pomocą którego konieczne jest potwierdzenie tej operacji. Po wykonaniu tej czynności okno znika, a z listy ankiet użytkownika usuwana jest

wybrana pozycja. Na rysunku 18 zaprezentowany został widok okna dialogowego służącego do potwierdzenia chęci usunięcia ankiety.



Rysunek 18: Widok okna dialogowego służącego do potwierdzenia chęci usunięcia ankiety.

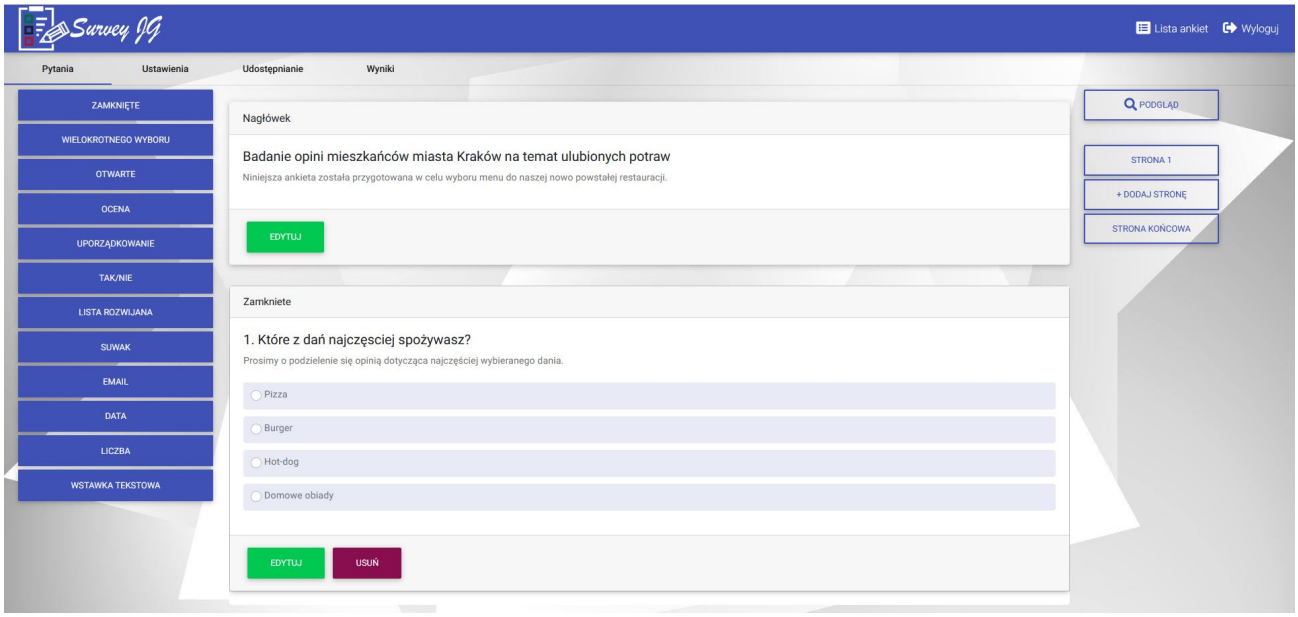
5.1.3 Generowanie formularza

Po utworzeniu nowej ankiety lub wybraniu opcji edycji użytkownikowi ukazuje się ekran widoczny na rysunku 19. Tryb widoku ankiety podzielony został na cztery zakładki „pytania", „ustawienia", „udostępnianie" oraz „wyniki". Zakładka zatytułowana „pytania" umożliwia dodawanie nowych pytań do formularza, ich edycję, zmianę kolejności oraz usuwanie.

Aplikacja oferuje możliwość wykorzystania różnych typów pytań. Ze względu na ich dużą różnorodność, przygotowana została lista z wykorzystaniem której użytkownik może wybrać typ pytania, który najbardziej odpowiada jego oczekiwaniom. Ma on możliwość selekcji pomiędzy pytaniami zamkniętymi z możliwością wyboru jednej pozycji, pytaniami zamkniętymi z możliwością wielokrotnego wyboru, pytaniami otwartymi, możliwością oceny danego zagadnienia, uporządkowaniem w kolejności wypisanych zagadnień, pytaniem, w którym ankietowany może odpowiedzieć twierdząco lub przecząco, pytaniem, w którym osoba wypełniająca ankietę może wybrać odpowiedź z rozwijanej listy. Użytkownik może również wybrać opcję, w której ankietowany za pomocą suwaka wybiera, w jakim stopniu zgadza się z tezą postawioną w pytaniu. Kolejnym rodzajem pytania jest „email", gdzie ankietowany może podać swój adres e-mail, w przypadku którego aplikacja sprawdza poprawność wprowadzonych wartości. Podobnie jest z pytaniem o datę, lecz w celu zminimalizowania błędów, jako podpowiedź dla osoby ankietowanej na ekranie ukazuje się wirtualny kalendarz. W pytaniu nazwanym „liczba" ankietowany

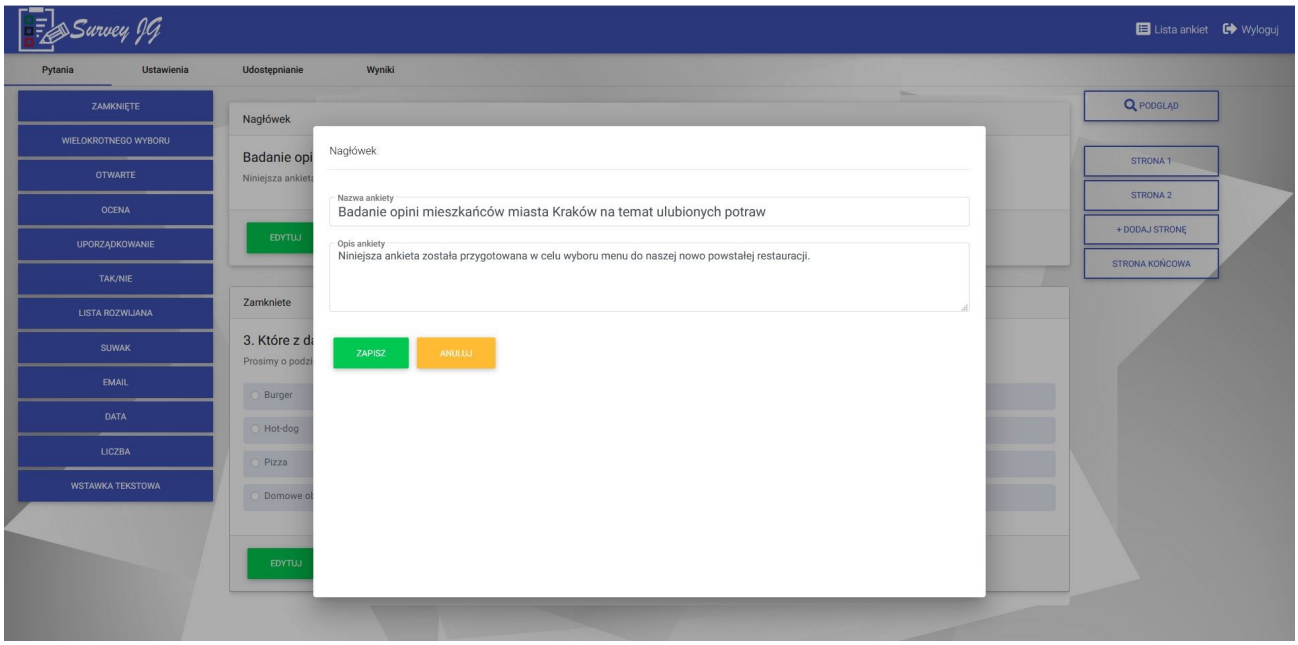
musi podać dowolną liczbę całkowitą lub zmiennoprzecinkową. Unikalną cechą tego pytania jest możliwość wprowadzenia znaków będących jedynie cyframi. Jako ostatnia wersja pytania stworzona została „wstawka tekstowa", która może zostać wykorzystana przez użytkownika jako komentarz pomiędzy pytaniami. Może również służyć jako podział ankiety na rozdziały i podrozdziały. Po prawej stronie widoczny jest przycisk pozwalający na przejście do podglądu wygenerowanej ankiety.

Podczas tworzenia formularza możliwe jest odseparowanie poszczególnych grup pytań na osobnych stronach. W prawej części ekranu aplikacji znajdują się przyciski pozwalające na nawigację pomiędzy istniejącymi stronami oraz dodawanie nowych.



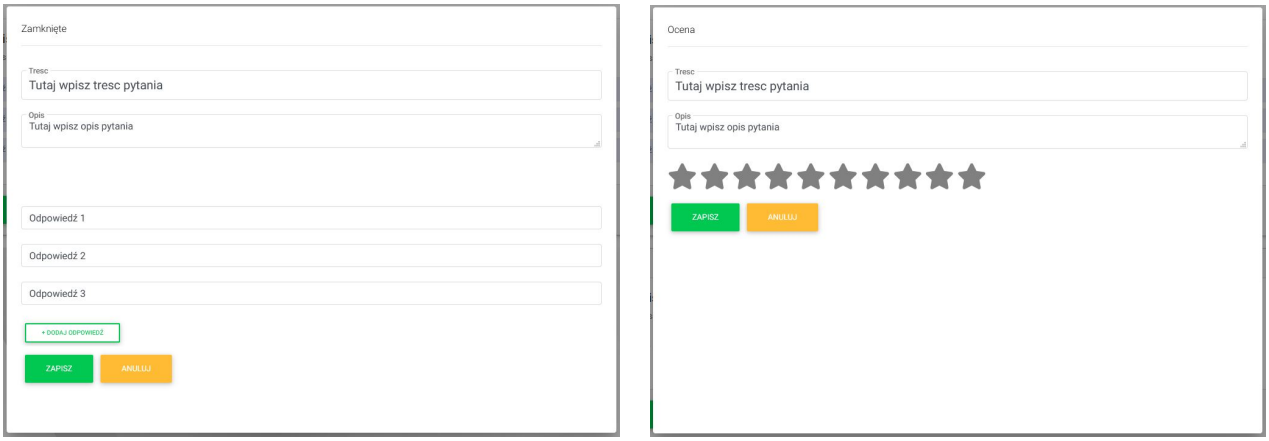
Rysunek 19: Widok umożliwiający edycje utworzonej ankiety.

Funkcja nagłówka, która pojawia się na początku pierwszej strony edycji ankiety, została stworzona w celu ułatwienia nadania nazwy ankiecie oraz umożliwienia dodania komentarza lub krótkiego opisu odnoszącego się do całego dokumentu. Okienko z widoczną edycją nagłówka zostało zaprezentowane na rysunku 20.



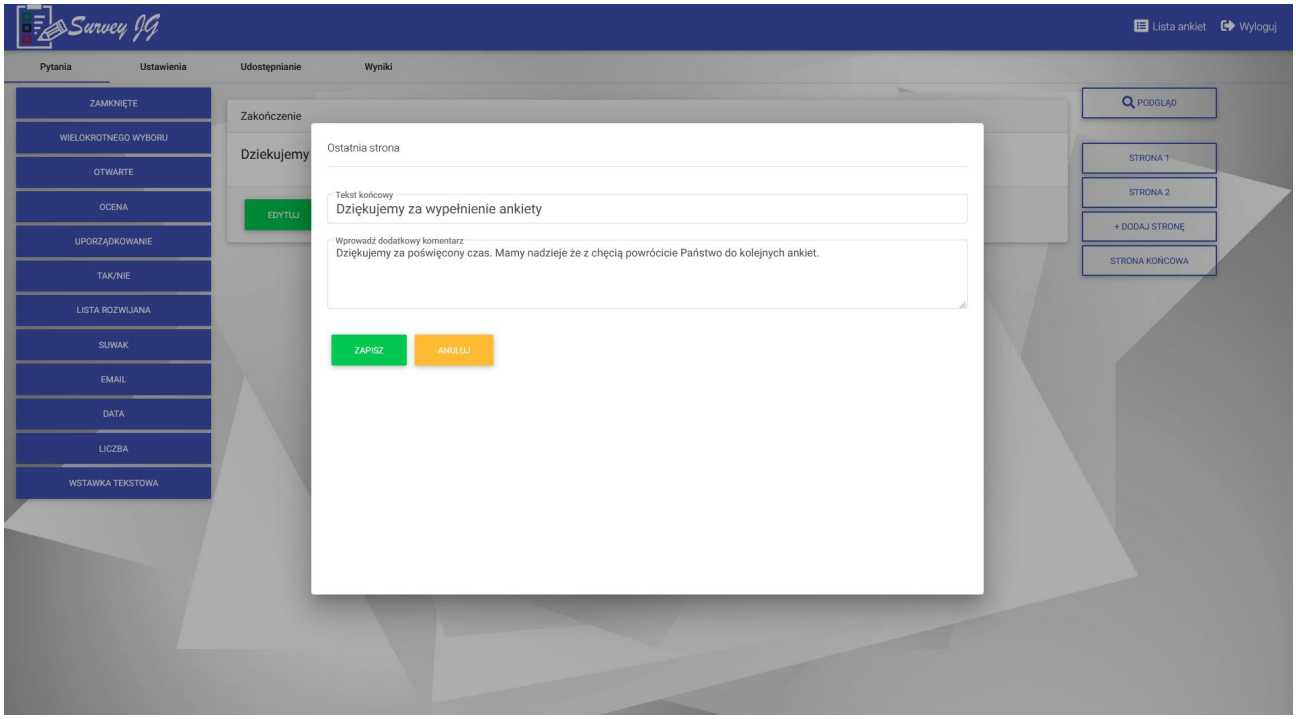
Rysunek 20: Okno pozwalające na edycję nagłówka formularza.

Użytkownik ma możliwość modyfikacji pytań w dowolnym momencie. Na rysunku 21 zaprezentowane zostały przykładowe okna edycyjne. W zależności od rodzaju pytania pole to może wyglądać w zróżnicowany sposób. Dla każdego z wariantów pytań istnieje możliwość wprowadzenia treści oraz dodatkowego komentarza. W przypadku pytań, w których respondent ma możliwość wyboru jednej z wielu odpowiedzi, możliwa jest modyfikacja ich ilości oraz wprowadzanie treści dla każdej z nich. Widok tych okien został dopasowany do rodzaju pytania. Przykładowo edycja pytania, w którym ankietowany może wybrać w jakim stopniu zgadza się z tezą zawartą w treści zagadnienia, znacznie różni się wyglądem od wcześniej opisywanego. W dolnej części okna znajdują się dwa przyciski. Jeden z nich służy do zapisywania wprowadzonych zmian. Rolą drugiego jest możliwość ich porzucenia oraz powrót do poprzedniego widoku.



Rysunek 21: Przykładowe okna służące do edycji pytań.

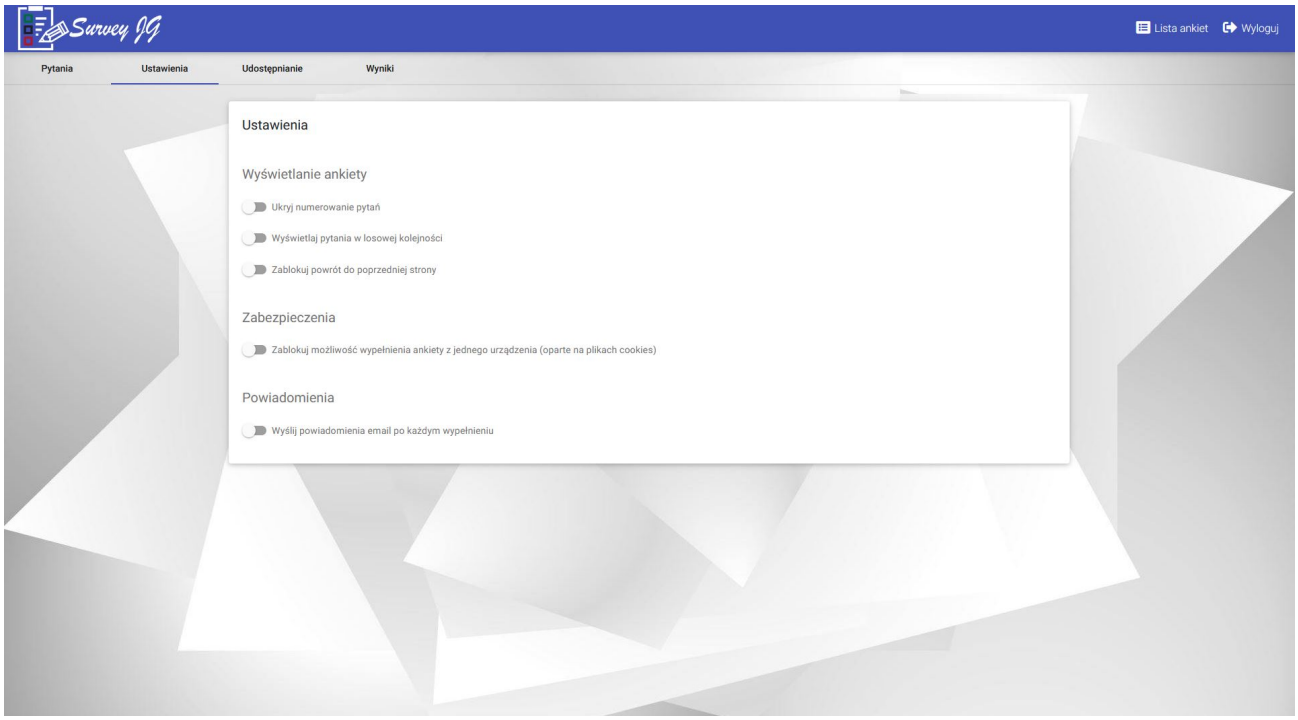
W aplikacji istnieje również możliwość dodania podziękowania za wypełnienie ankiety, które pojawi się na ostatniej stronie po udzieleniu odpowiedzi na wszystkie pytania. W tym celu należy przejść do strony końcowej oraz użyć przycisku do edycji. Podobnie jak w przypadku wprowadzania zmian dotyczących pytań, istnieje możliwość edycji tekstu końcowego oraz dodania opcjonalnego komentarza, korzystając z okna dialogowego. Widok okna jest zaprezentowany na rysunku 22.



Rysunek 22: Okno dialogowe edycji tekstu końcowego.

5.1.4 Ustawienia formularzy

W aplikacji użytkownik ma możliwość dokonywania wyboru bazowych ustawień dotyczących formularza. Opcja ta zawarta jest w zakładce ustawienia oraz przedstawiona została na rysunku 23. Widoczne funkcje podzielone zostały na trzy kategorie dotyczące wyświetlania ankiety, zabezpieczenia oraz powiadomień. W kategorii pierwszej użytkownik może wybrać opcje ukrywania numerów pytań, wyświetlania pytań na poszczególnych stronach w losowej kolejności oraz zablokowania możliwości powrotu do poprzedniej strony. W zabezpieczeniach użytkownik ma możliwość ochrony formularza przed wielokrotnym wypełnianiem przez tego samego respondenta. Funkcja ta oparta jest o informacje zapisywane w plikach cookies. W przypadku powiadomień użytkownik może zadeklarować chęć otrzymywania wiadomości e-mail po każdorazowym wypełnieniu ankiety przez respondenta. Wybór ustawień dokonywany jest przy użyciu przycisków przełączania.

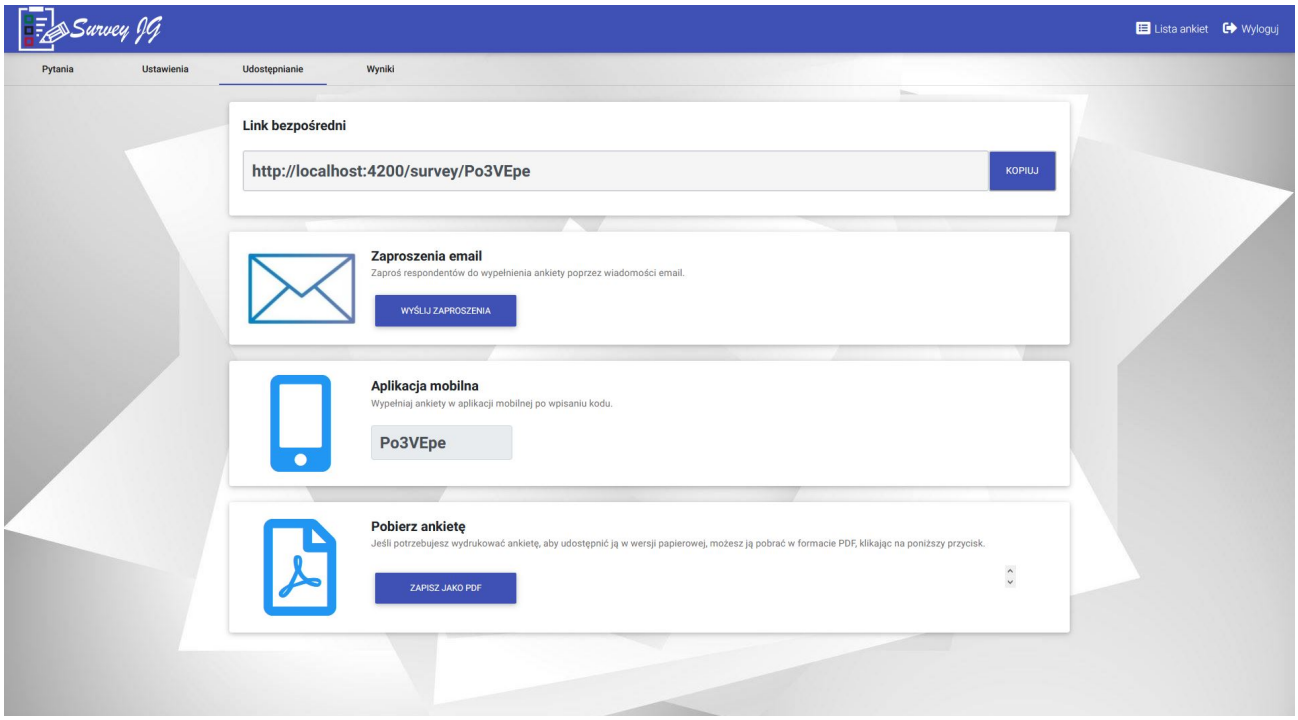


Rysunek 23: Widok ustawień formularza.

5.1.5 Metody udostępniania formularzy

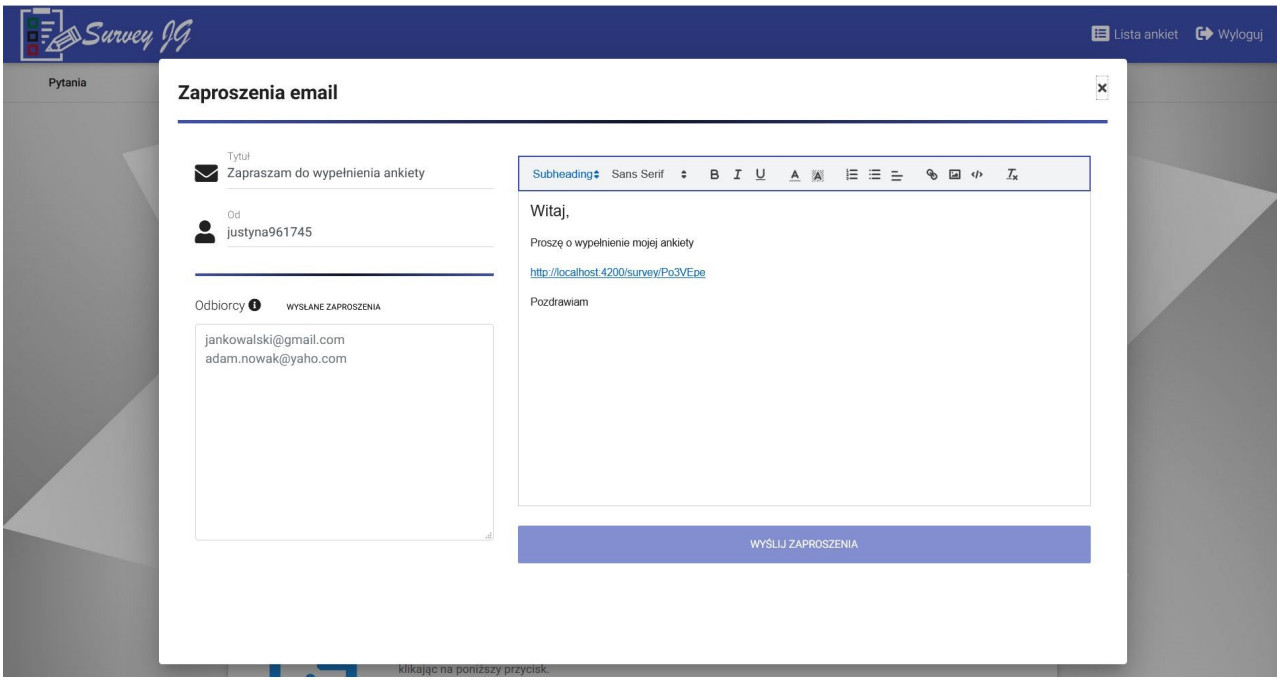
Kolejną zakładką, która zaprezentowana została na rysunku 24 jest „udostępnianie". Użytkownik ma możliwość wyboru kilku opcji udostępnienia stworzonej przez siebie ankiety. Numer identyfikacyjny formularza został zakodowany i jego siedmioznakowa reprezentacja może posłużyć do weryfikacji ankiety. Najprostszą z metod udostępniania jest unikalny link generowany dla każdej z ankiet na podstawie zakodowanego identyfikatora. Przesłanie linku do respondentów sprawi, że zostaną oni bezpośrednio przekierowani do utworzonego formularza. Kolejną możliwością jest wybranie opcji zaproszeń za pomocą wiadomości e-mail.

Jako następna metoda udostępniania ankiet, stworzona została aplikacja mobilna. Aby możliwe było szybkie odnalezienie danej ankiety z zastosowaniem tego narzędzia, używany jest unikalny, identyfikujący ciąg znaków. Wprowadzenie go w aplikacji mobilnej przekieruje ankietowanego bezpośrednio do wybranego formularza. Z myślą o użytkownikach, którzy wolą przeprowadzać ankiety w sposób tradycyjny, stworzona została możliwość zapisania w formacie PDF do późniejszego wydrukowania. W tym celu, niektóre pytania zostały przetworzone w taki sposób, aby odpowiedź na nie była możliwa w wersji papierowej. Dzięki temu podejściu można w szybki sposób przygotować, wydrukować i przeprowadzić sondaż lub głosowanie w sposób tradycyjny.



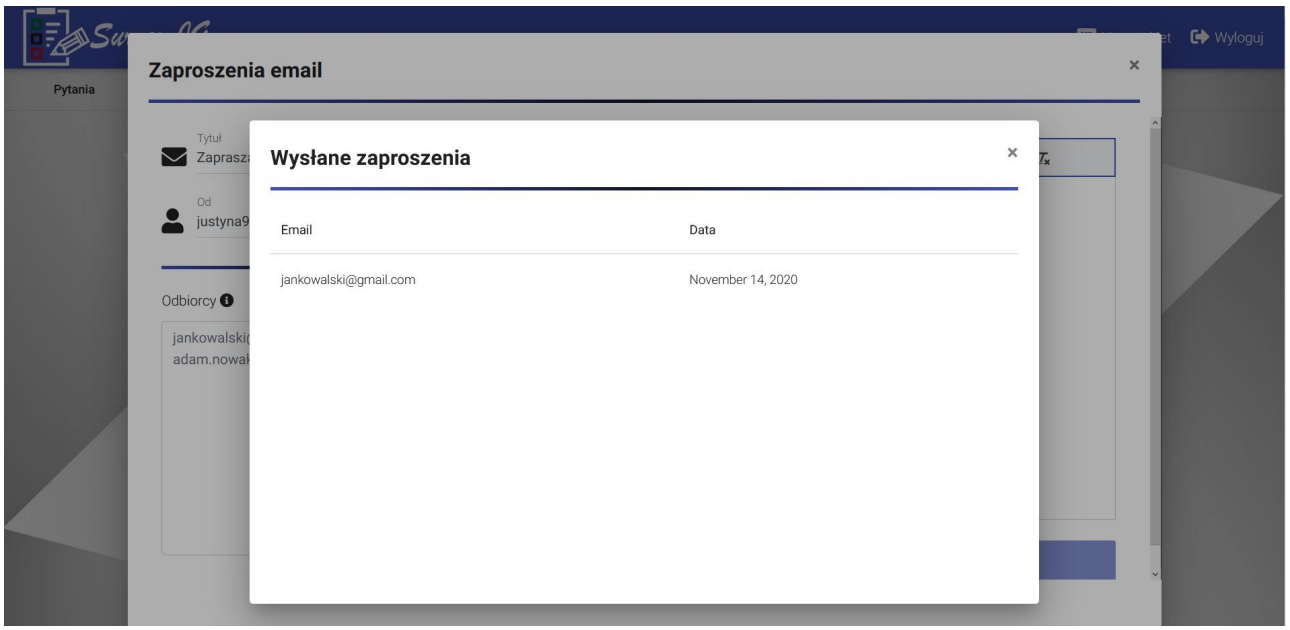
Rysunek 24: Widok możliwości udostępniania.

Jedną z metod udostępniania formularzy jest rozsyłanie zaproszeń za pomocą wiadomość e-mail. Na rysunku 25 przedstawiono okno służące do tworzenia zaproszeń. Aby przesłać wiadomość, należy wprowadzić tytuł oraz nazwę użytkownika. Kolejne pole przeznaczone jest do wprowadzenia listy adresów e-mail, do których zaproszenia mają zostać przekazane. Do każdego z nich zostanie wysłana jednocześnie wiadomość o treści, która zostanie wprowadzona przez użytkownika w polu znajdującym się po prawej stronie okna. Widoczna wiadomość generuje się automatycznie, ale użytkownik może dowolnie edytować jej treść. Dodatkowo stworzona została ikona z literą „i", która wyświetla podpowiedź jak definiować odbiorców.



Rysunek 25: Okno do tworzenia zaproszeń e-mail.

Klikając na odnośnik „Wysłane zaproszenia", użytkownik ma możliwość sprawdzenia listy wysłanych wcześniej wiadomości. Na rysunku 26 zaprezentowane zostało okno, w którym widoczne są wszystkie wysłane zaproszenia. Jest to pomocne w przypadku, gdy ankieta wysyłana jest do dużej liczby odbiorców lub w momencie, gdy proces wysyłania ankiet podzielony jest na kilka etapów, dzięki czemu użytkownik może w łatwy sposób sprawdzić, do kogo dana ankieta została już wysłana.



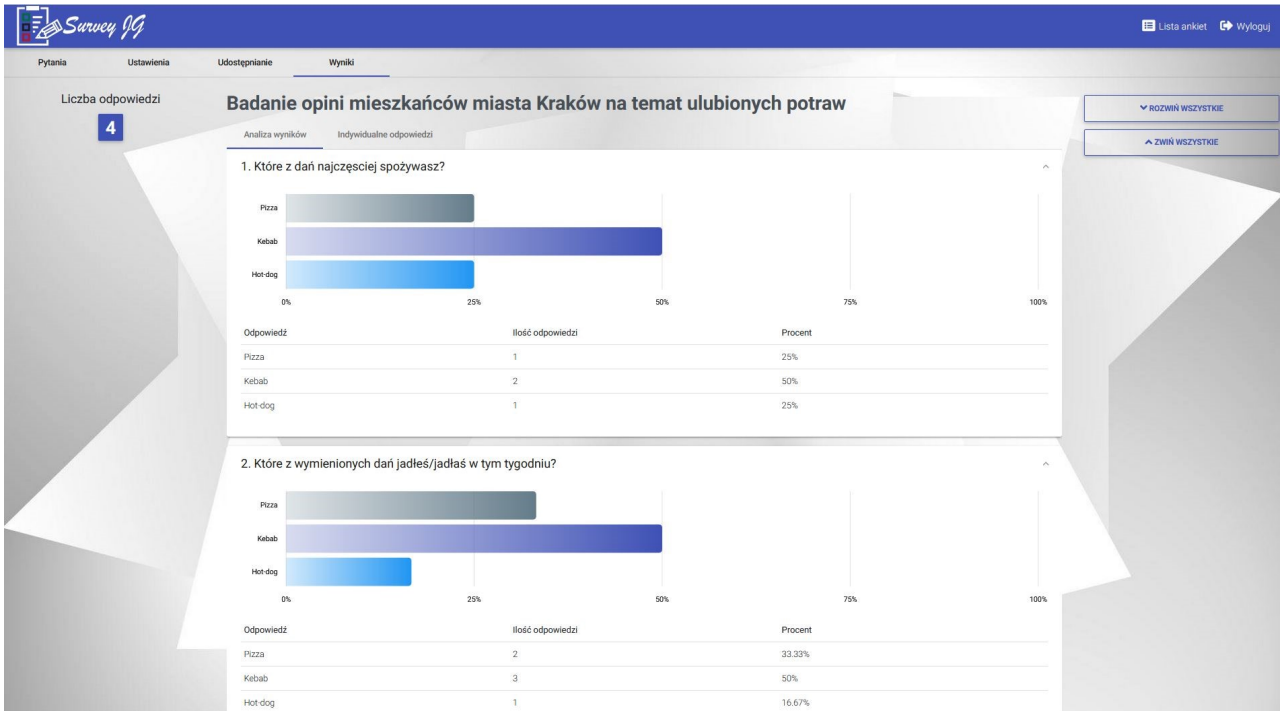
Rysunek 26: Okno prezentujące listę wysłanych zaproszeń.

5.1.6 Analiza wyników

Ważnym aspektem aplikacji, który został zaprezentowany w postaci kolejnej zakładki, są „wyniki". Jest to moduł, w którym użytkownik może w szybki i prosty sposób przeanalizować wyniki, które generują się automatycznie, po każdym uzupełnieniu ankiety. Zakładka podzielona została na dwa obszary „Analiza wyników" oraz „Indywidualne odpowiedzi". Pierwszy z nich prezentuje indywidualne odpowiedzi poszczególnych respondentów w postaci tabeli, natomiast w drugim znajduje się zbiorcza analiza wyników przedstawiona w postaci graficznej.

W celu znacznie łatwiejszej analizy poszczególnych pytań do każdego wybranego rodzaju stworzony został osobny sposób prezentacji wyników. Każde pytanie ma możliwość rozwinięcia graficznej analizy rezultatów. Ze względu na fakt, że zbyt duża liczba wykresów znajdujących się jeden nad drugim może utrudniać analizę wyników, istnieje możliwość ukrycia lub pokazania wykresów dotyczących dowolnie wybranych pytań.

W przypadku pytań zamkniętych z możliwością pojedynczego i wielokrotnego wyboru, pytania z listą wybieranych odpowiedzi oraz pytania z możliwością wyboru tak lub nie, wyniki prezentowane są w postaci poziomego histogramu. Na osi pionowej widoczne są kolejne możliwości wyboru, natomiast na osi poziomej można zaobserwować udział procentowy poszczególnych odpowiedzi. Poniżej graficznej prezentacji znajduje się tabela z dokładnymi wartościami procentowymi dla każdej wybranej odpowiedzi. Na rysunku 27 przedstawiono graficzną prezentację pytań w widoku „analiza wyników" z wygenerowanymi przykładowymi histogramami.



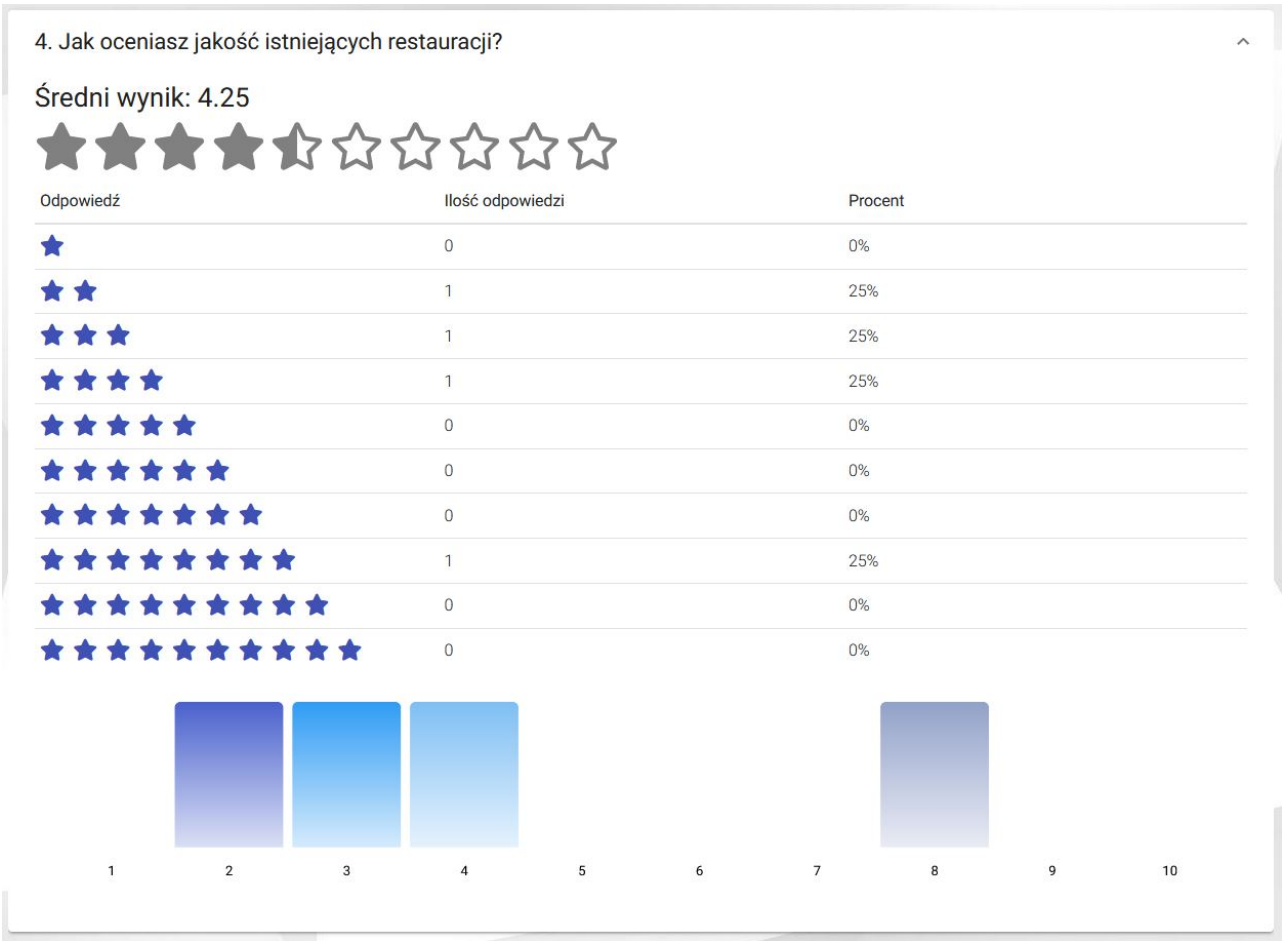
Rysunek 27: Graficzna prezentacja wyników z wygenerowanymi przykładowymi histogramami.

Kolejnym rodzajem są pytania otwarte. Prezentacja wyników ukazywana jest w postaci tabeli, gdzie w jednej kolumnie widoczne są odpowiedzi, natomiast w drugiej prezentowany jest

unikalny numer respondenta. Widok okna z wynikami dla pytania otwartego widoczny jest na rysunku 28.

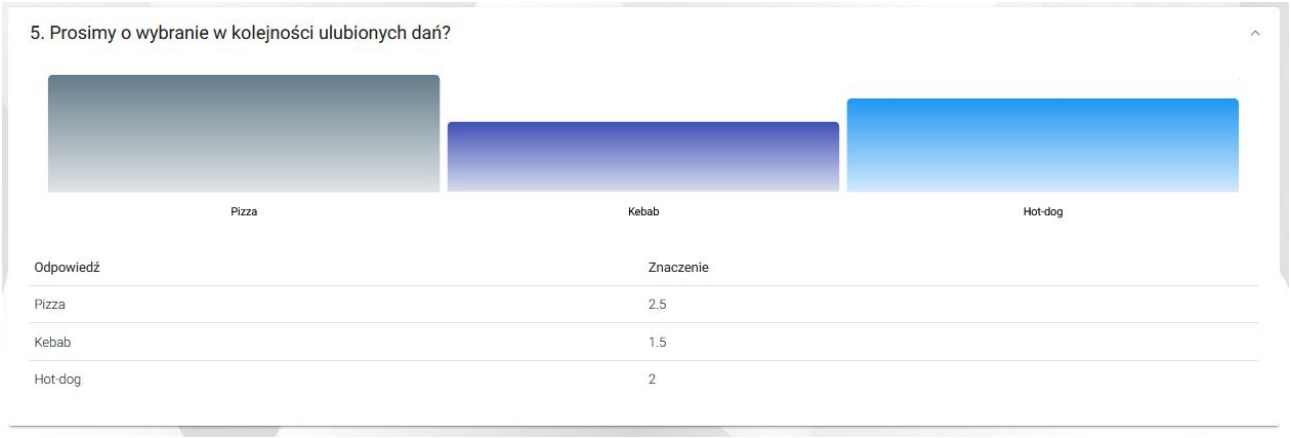
Rysunek 28: Wyniki dla pytania otwartego.

Dla pytań, w których ankietowany może wskazać w jakim stopniu zgadza się z tezą zawartą w pytaniu, wyliczana jest średnia arytmetyczna, która jest jednocześnie prezentowana w postaci graficznej oraz liczbowej. Dodatkowo użytkownik może na schemacie oraz pionowym histogramie sprawdzić procentowy rozkład dla poszczególnych odpowiedzi. Statystyka dla tego typu pytania widoczna jest na rysunku 29.



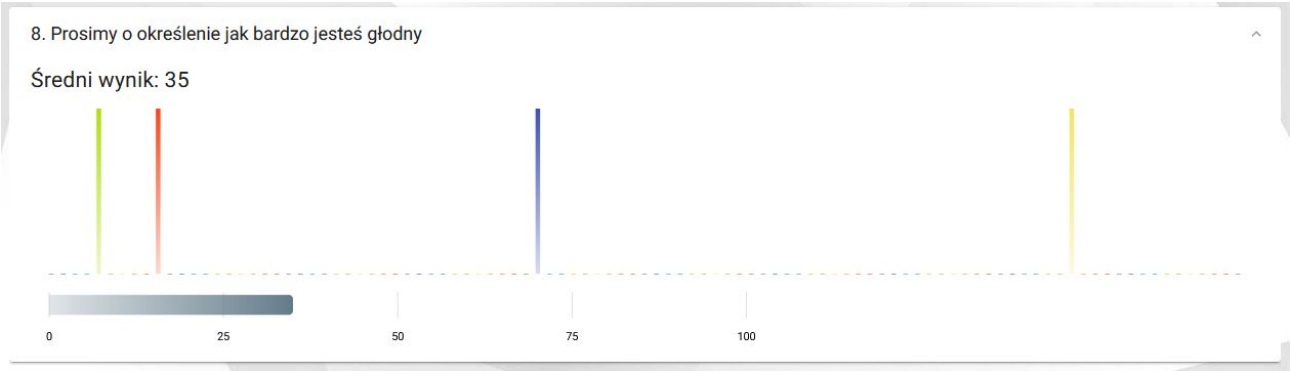
Rysunek 29: Graficzna reprezentacja wyników dla pytania dotyczącego oceny.

W przypadku pytania, w którym wymagane jest ustawianie odpowiedzi w preferowanej kolejności, w celu wizualizacji danych wyliczany jest współczynnik znaczenia, który prezentowany jest w postaci histogramu oraz tabeli z wartościami. Algorytm obliczeniowy opracowany został w celu określenia znaczenia poszczególnych odpowiedzi w oparciu o kolejność ich wyboru przez ankietowanych. Wyliczenia zostały zaprojektowane w taki sposób, że im niższa wartość współczynnika jest uzyskiwana, tym większe znaczenie ma dana odpowiedź. Zakres otrzymywanego współczynnika mieści się w przedziale od 1 do liczby możliwych do wybrania odpowiedzi. Wartość 1 oznacza, że dana odpowiedź zawsze znajdowała się na pierwszym miejscu, natomiast wartość równa liczbie możliwych do wyboru odpowiedzi oznacza usytuowanie jej zawsze na ostatnim miejscu. Współczynnik ten obliczany jest poprzez nadanie wartości kolejnym odpowiedziom wybranym przez respondenta. Odpowiedź umieszczona na miejscu pierwszym otrzymuje wartość 1, natomiast każda kolejna wartość o jeden większą. Wyliczenie współczynnika w oparciu o określoną liczbę pytań odbywa się poprzez zsumowanie wartości, które uzyskała dana odpowiedź oraz podzielenie jej przez liczbę odpowiedzi w pytaniu. Na rysunku 30 przedstawiony został histogram oraz tabela prezentujące statystyki dla opisanego powyżej typu pytania.



Rysunek 30: Graficzna reprezentacja wyników dla pytania dotyczącego ustawienia kolejności.

Dla pytania, w którym ankietowani wybierają liczbę z zakresu 0-100 za pomocą suwaka, prezentacja wyników oparta jest o wykres. Można na nim zaobserwować rozkład liczby odpowiedzi zawierających daną wartość. Dodatkowo w postaci liczbowej oraz poziomego histogramu prezentowana jest średnia wybranych wartości. Widok okna z wynikami dla tego typu pytania widoczny jest na rysunku 31.



Rysunek 31: Graficzna reprezentacja wyników dla pytania typu suwak.

Analizując pytanie, w którym ankietowany może wpisać adres e-mail, otrzymywane wyniki prezentowane są w postaci tabeli. Po lewej stronie widoczny jest adres e-mail, natomiast po prawej znajduje się unikalny numer ankietowanego. Widok tabeli z wynikami dla tego typu pytania widoczny jest na rysunku 32.

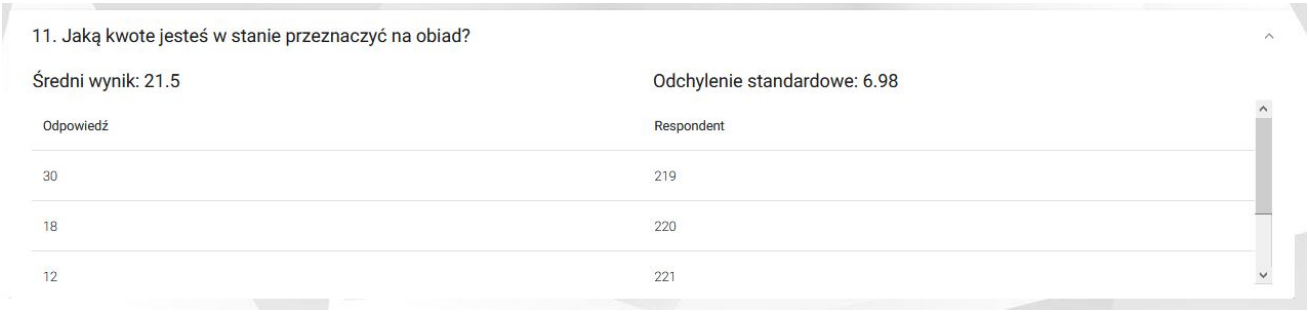
Rysunek 32: Wyniki dla pytania, w którym ankietowany musi wprowadzić adres e-mail.

Biorąc pod uwagę pytanie umożliwiające wprowadzenie daty z wirtualnego kalendarza, wyniki prezentowane są w postaci tabeli, w której pierwsza kolumna zawiera zaznaczoną datę. W drugiej znajduje się liczba określająca ilość wystąpień danej wartości. Ostatnia zawiera wymienione po przecinku unikalne numery respondentów, którzy wybrali daną datę. Widok okna zawierającego wyniki dla tego typu pytania znajduje się na rysunku 33.

Rysunek 33: Tabela prezentująca wyniki dotyczące pytania z możliwością wyboru daty.

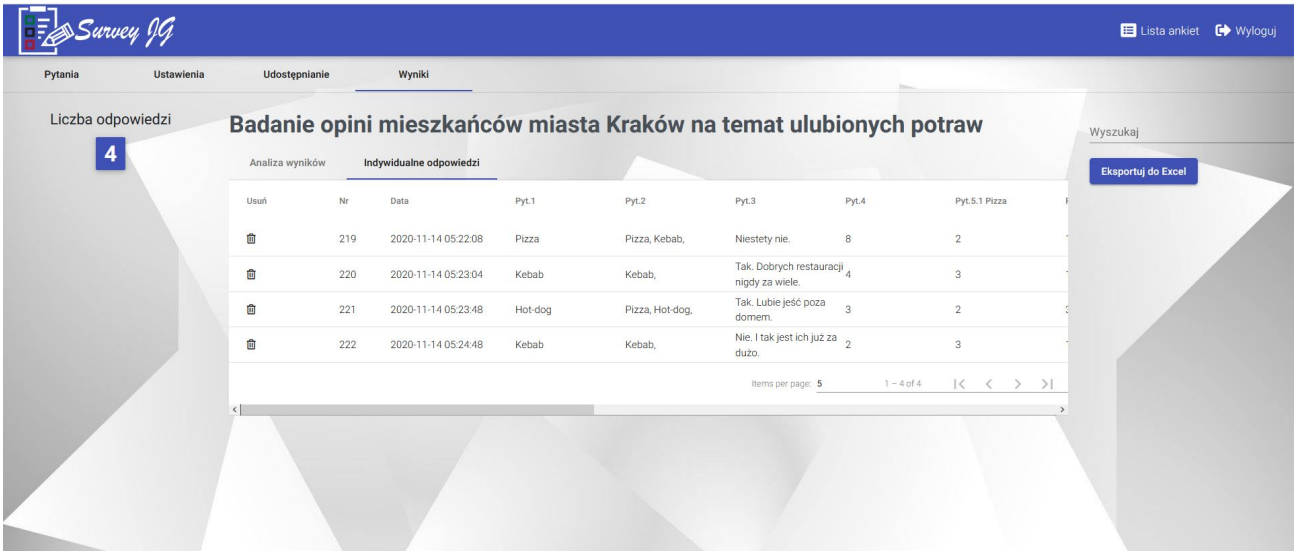
Dla typu pytania, w którym ankietowany ma możliwość podania wyłącznie liczby, wyniki prezentowane są w postaci tabelarycznej z wprowadzoną wartością liczbową oraz unikalnym

numerem respondenta dla każdego wpisu. Dodatkowo obliczana jest średnia arytmetyczna oraz odchylenie standardowe dla wszystkich wprowadzonych w ankiecie wartości. Prezentacja przykładowych wyników dla tego typu pytania znajduje się na rysunku 34.



Rysunek 34: Prezentacja wyników dla pytania o liczbę.

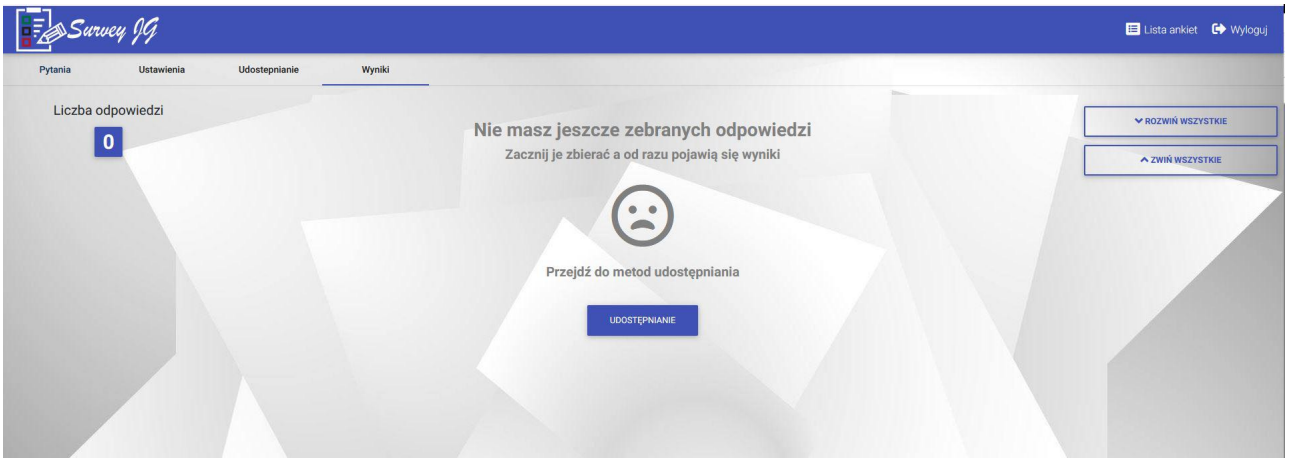
Istnieje również możliwość przeanalizowania uzyskanych wyników w oparciu o zbiór wszystkich odpowiedzi dokonanych przez każdego z respondentów. Są one prezentowane w postaci tabelarycznej. Użytkownik ma możliwość sortowania tabeli z wykorzystaniem dowolnej kolumny. W kolejnych kolumnach prezentowane są odpowiedzi na poszczególne pytania udzielone przez ankietowanych. Istnieje również możliwość wyszukiwania dowolnej wartości w tabeli. Ten sposób prezentacji wyników jest w szczególności przydatny w momencie, gdy użytkownik chce przeanalizować relacje pomiędzy odpowiedziami na kolejne pytania udzielone przez jednego lub kilku ankietowanych. Użytkownik ma możliwość usunięcia odpowiedzi wybranych respondentów. Widok tabeli został zaprezentowany na rysunku 35.



Rysunek 35: Widok tabeli z zestawieniem wyników dla poszczególnych respondentów.

Aby użytkownik miał dostęp do analizy wyników ankiety, niezbędna jest przynajmniej jedna odpowiedź w postaci wypełnionego formularza. Po lewej stronie okna widocznego na rysunku 36 znajduje się licznik prezentujący liczbę respondentów, którzy w sposób prawidłowy wypełnili

ankietę. W sytuacji, gdy formularz nie został jeszcze wypełniony przez żadnego ankietowanego wyświetlany jest ekran informujący użytkownika o konieczności udostępnienia ankiety w celu zebrania wyników. Aby ułatwić użytkownikowi lokalizację metod udostępniania, stworzony został przycisk widoczny na środku strony, który przekierowuje twórcę bezpośrednio do zakładki umożliwiającej szybkie rozpowszechnienie formularza.

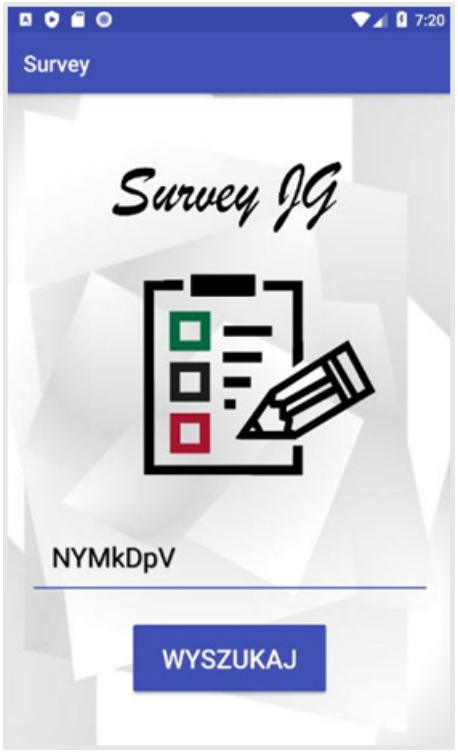


Rysunek 36: Widok informujący o braku wyników dla danej ankiety.

5.2 Funkcjonalności aplikacji internetowej oraz mobilnej niewymagające logowania

W przypadku aplikacji służących do wypełniania ankiet ważne jest, aby były one proste w obsłudze. Łatwość obsługi jest kluczowa, ponieważ od niej może zależeć popularność aplikacji wśród użytkowników, która bezpośrednio przekłada się na ilość zbieranych wyników. Brak konieczności logowania w celu wypełnienia ankiety sprawia, że znacznie większa liczba respondentów uzupełni przekazany im formularz.

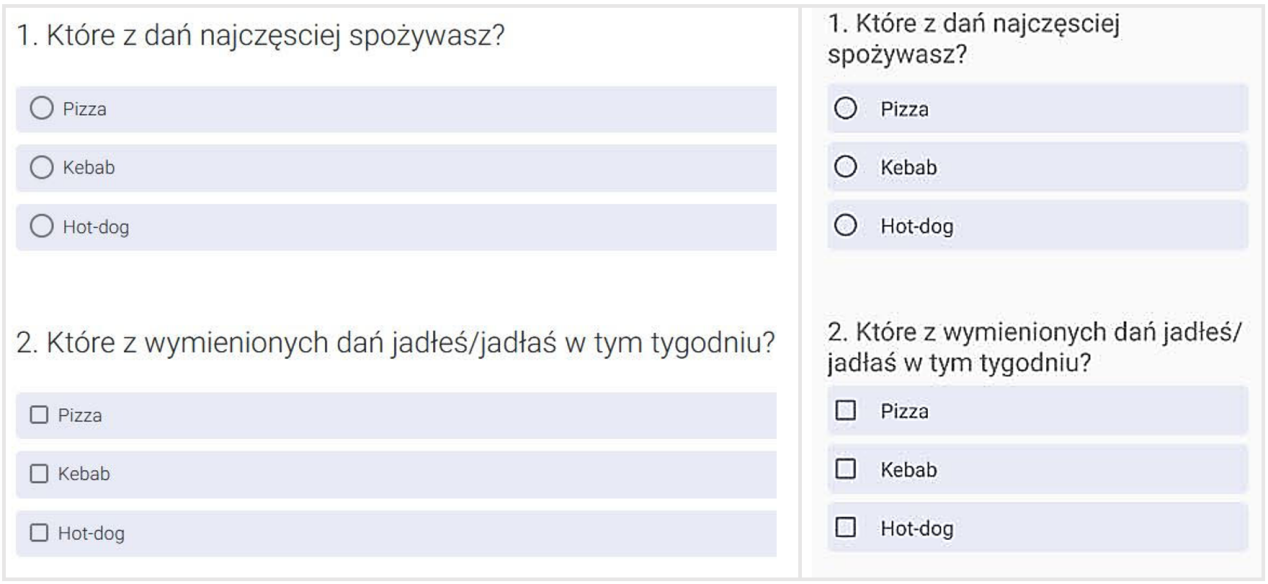
Aplikacja mobilna została przygotowana specjalnie w celu dotarcia do szerszego grona odbiorców. Większość użytkowników korzystających z urządzeń mobilnych znacznie chętniej poświęca czas na wypełnienie ankiety właśnie z wykorzystaniem tego typu urządzeń. Na rysunku 37 zaprezentowany został ekran powitalny aplikacji mobilnej, na którym widoczne jest logo oraz nazwa aplikacji. Dodatkowo znajduje się również pole tekstowe, w które należy wprowadzić kod formularza w celu jego wypełnienia. Ten kod może zostać odnaleziony przez twórcę ankiety w zakładce udostępnianie w aplikacji internetowej oraz przesłany do potencjalnych respondentów.



Rysunek 37: Widok ekranu głównego aplikacji mobilnej.

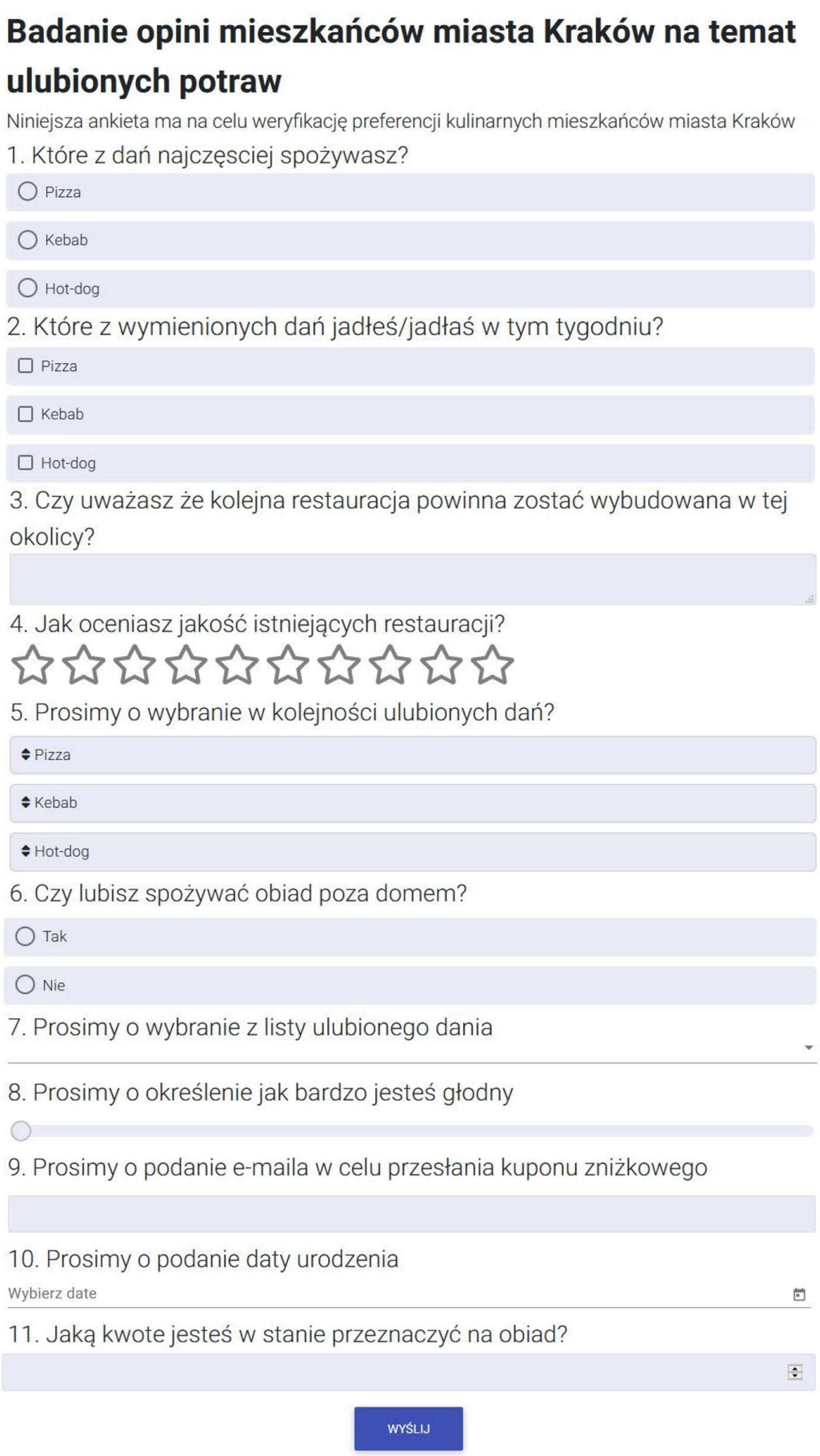
W przypadku chęci wypełnienia ankiety z wykorzystaniem aplikacji internetowej respondent może wykorzystać wygenerowany link lub zostać zaproszony do wzięcia udziału w ankiecie za pośrednictwem wiadomości e-mail. Podobnie jak w przypadku aplikacji mobilnej nie ma konieczności logowania się w celu wypełniania ankiety.

Na rysunku 38 zaprezentowana została ta sama ankieta widoczna w aplikacji internetowej oraz mobilnej. Ze względu na różnice pomiędzy urządzeniami wygląd nieznacznie się różni. Pomimo tego sposób wypełniania ankiet jest podobny. Największą różnicą wynikającą z wykorzystanych urządzeń jest sposób zaznaczania oraz wprowadzania odpowiedzi. W przypadku aplikacji internetowej odpowiedzi zaznaczane są z wykorzystaniem kursora, natomiast tekst wprowadzany jest z wykorzystaniem klawiatury. Jeśli chodzi o urządzenia mobilne, do zaznaczania odpowiedzi używany jest ekran dotykowy. Wprowadzanie tekstu realizowane jest poprzez wirtualną klawiaturę urządzania.



Rysunek 38: Widok ankiety w aplikacji internetowej (po lewej) oraz w aplikacji mobilnej (po prawej).

Na rysunku 39 zaprezentowana została przykładowa ankieta, w której wykorzystano wszystkie typy pytań. W celu ułatwienia wypełniania ankiety różne typy pytań posiadają różne ikony znajdujące się przy odpowiedziach. W przypadku pytań zamkniętych jednokrotnego wyboru zastosowana została ikona okręgu, natomiast dla pytań wielokrotnego wyboru są to kwadraty. Zastosowanie różnych ikon w tego typu pytaniach ma za zadanie znacząco przyspieszyć proces wypełniania formularza. Osoba mająca pewne doświadczenie z wykorzystaniem tej aplikacji bez zastanowienia będzie wiedziała, czy w danym pytaniu może zaznaczyć jedną odpowiedź, czy kilka. Dla pytań wymagających wpisania odpowiedzi przygotowane zostało pole tekstowe wyróżnione kolorem odmiennym od tła. W przypadku pytania, w którym ankietowany może wybrać, w jakim stopniu zgadza się z tezą zawartą w pytaniu, wykorzystane zostały ikony gwiazdek. Respondent może zaznaczyć różną liczbę gwiazdek, w zależności od tego, w jakim stopniu zgadza się z tezą pytania. Całość ankiety została zachowana w jednolitej kolorystyce i stylistyce.



Rysunek 39: Widok przykładowej ankiety z wszystkimi typami pytań.

Podczas wypełniania formularza respondent zobowiązany jest do wypełnienia wszystkich pytań w sposób prawidłowy zgodnie z założeniami twórcy ankiety. Pominięcie któregoś z pytań lub wypełnienie niepoprawnie skutkuje brakiem możliwości przejścia do kolejnej strony formularza, a także uniemożliwieniem przesłania ankiety. W przypadku niedostosowania się do powyższych wytycznych, na ekranie aplikacji pojawi się alert informujący o nieprawidłowym wypełnieniu. Dodatkowo w celu ułatwienia wyeliminowania błędnych odpowiedzi, pod każdym pytaniem, które uniemożliwia przejście do kolejnego etapu, ukaże się informacja dotycząca błędu. Założeniem systemu nie jest skomplikowanie procesu wypełniania ankiety, lecz unifikacja danych wykorzystywanych to tworzenia analiz wyników. Na rysunku 40 przedstawiony został widok formularza z niepoprawnie wypełnionymi odpowiedziami. W identyczny sposób zostają wyświetlane komunikaty w aplikacji mobilnej.

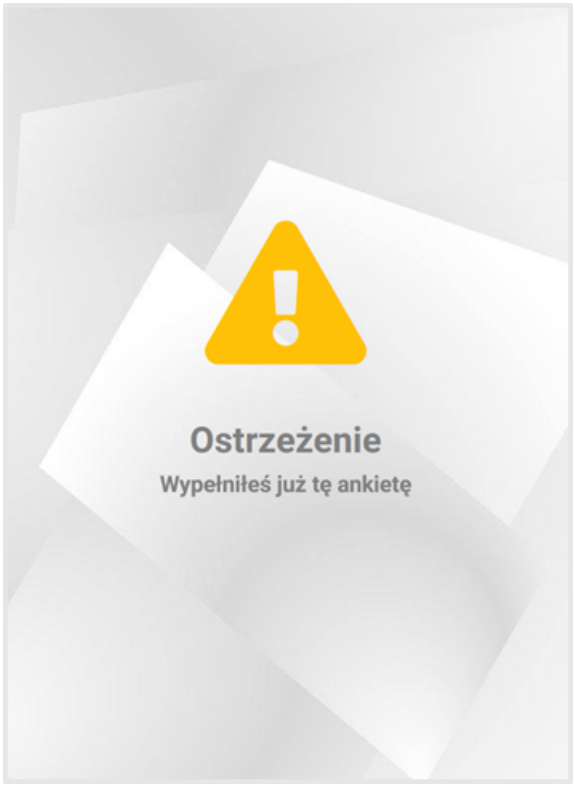
Rysunek 40: Formularz z niepoprawnie wypełnionymi odpowiedziami.

Po wypełnieniu ankiety respondentowi ukazuje się ekran widoczny na rysunku 41, na którym znajduje się podziękowanie za wypełnienie ankiety oraz logo aplikacji. Klikając na grafikę, użytkownik zostanie przekierowany do ekranu głównego aplikacji, gdzie będzie miał możliwość

rejestracji w celu utworzenia konta. Podobnie jest w przypadku aplikacji mobilnej, w której dodatkowo znajduje się przycisk, za pomocą którego ankietowany może w szybki sposób powrócić do ekranu głównego.

Rysunek 41: Ekran podziękowania w aplikacji internetowej (po lewej) oraz mobilnej (po prawej).

Aplikacja w oparciu o pliki cookies zabezpiecza ankiety przed wielokrotnym wypełnianiem przez tego samego respondenta. W momencie, gdy ankietowany próbuje kolejny raz wypełnić tą samą ankietę, na jego ekranie ukazuje się komunikat widoczny na rysunku 42.



Rysunek 42: Komunikat o braku możliwości ponownego wypełnienia ankiety.

6 Możliwości rozwoju

Każdy projekt, nawet w momencie jego ukończenia posiada obszary, które mogą zostać usprawnione lub funkcje, które mogą zostać dodane w celu zwiększenia satysfakcji potencjalnego klienta. Typowe projekty informatyczne tworzone są przez zespoły posiadające architektów, deweloperów oraz testerów oprogramowania. Dzięki tak rozbudowanej kadrze możliwe jest stworzenie kompleksowego narzędzia, posiadającego funkcje, których poziom zaawansowania jest tak duży, że stają się one praktycznie niemożliwe do stworzenia przez jedną osobę. W ramach niniejszej pracy stworzony został jednoosobowy projekt, którego celem było przejście przez wszystkie etapy projektowania systemu oraz stworzenie możliwie jak najbardziej zaawansowanego produktu. Poniżej przedstawione zostały pomysły na dalszy rozwój aplikacji wykonywanej w ramach niniejszej pracy.

Element aplikacji, który mógłby zostać usprawniony to zabezpieczenie przed wielokrotnym wypełnianiem ankiety. Obecne rozwiązanie oparte jest na plikach cookies. Jest to metoda wymagająca uczciwości użytkowników, ponieważ każdy z nich może wyczyścić pliki cookies zapisane w przeglądarce, przez co system uzna go za nowego użytkownika chcącego wypełnić ankietę. Rozwiązaniem pozwalającym wykluczyć konieczność polegania na uczciwości ankietowanego jest generowanie unikalnego kodu dla pojedynczego respondenta oraz stworzenie systemu, który weryfikowałby poziom wypełnienia danej ankiety dla pojedynczego kodu.

Kolejna opcja, która możliwa jest do stworzenia w przyszłości to dowolność modyfikacji końcowego wyglądu przygotowywanej ankiety, sondażu lub glosowania. Jest to istotne z punktu widzenia użytkowania, ponieważ możliwość personalizacji wyglądu oraz umieszczania zewnętrznych grafik, umożliwiałaby wykorzystanie tej aplikacji przez klientów biznesowych.

Wartościową funkcją, która mogłaby zostać wprowadzona, jest statystyka dotycząca liczby odwiedzin, skonfrontowana z liczbą w pełni wypełnionych ankiet. Dodatkowo możliwa do zastosowania jest funkcja oceny danej ankiety wyświetlająca się na ekranie podziękowania za jej wypełnienie. Takie dane pozwoliłyby użytkownikowi wyciągnąć wnioski dotyczące projektowania przyszłych formularzy oraz umożliwiłyby dostosowanie pytań do preferencji respondentów.

7 Podsumowanie

Celem niniejszej pracy magisterskiej było zbudowanie systemu aplikacji pozwalających na tworzenie ankiet, sondaży oraz głosowań, jak również zbieranie odpowiedzi z użyciem dwóch rodzajów aplikacji internetowej oraz mobilnej. Nieodłączną częścią systemu miała być analiza wyników w postaci graficznej oraz tabelarycznej. Ważnym aspektem oprogramowania miał być system, pozwalający wykorzystać kilka różnych metod w celu udostępnienie formularzy potencjalnym respondentom. W efekcie powstały aplikacje spełniające większość wymagań postawionych na etapie projektowania. Stworzone moduły stanowią w pełni funkcjonalny system do tworzenia formularzy oraz zbierania opinii na dowolne tematy.

Podczas etapu projektowania aplikacji zostały wybrane opisane w rozdziale trzecim technologie. Analizując proces implementacji systemu oraz testując gotowy produkt można stwierdzić, że ich dobór był właściwy. Na żadnym z etapów tworzenia projektu nie napotkano problemów, których rozwiązanie byłoby niemożliwe z wykorzystaniem opisanych technologii oraz narzędzi. Dzięki umiejętnemu zastosowaniu języków, frameworków oraz bibliotek gotowy projekt jest wykonany w sposób estetyczny i przemyślany. Biorąc pod uwagę cały proces tworzenia aplikacji, można stwierdzić, że wykorzystane technologie w pełni spełniły stawiane im wymagania, a proces tworzenia okazał się wartościowy oraz satysfakcjonujący. Pozwolił on na zdobycie oraz utrwalenie wiedzy z zakresu projektowania oraz tworzenia systemu opartego o architekturę klient-serwer. Zdobyta podczas projektowania oraz implementacji systemu wiedza dotycząca tworzenia aplikacji po stronie serwera, jak również aplikacji internetowej oraz mobilnej, z pewnością zostaną wykorzystane w karierze zawodowej autorki pracy.

8 Bibliografia

[1] *HTTP* [dostęp: 15.11.2020] https://tools.ietf.org/html/rfc2616

[2] K. Lange: *The Little Book on REST Services*, Copenhagen 2016

[3] D. Kuhlman: *A Python Book: Beginning Python, Advanced Python, and Python Exercises*, 2013

[4] A. Holovaty, J. K. Moss *The Definitive Guide to Django: Web Development Done Right*, 2019

[5] *Django Rest Framework* [dostęp: 15.11.2020] https://www.django-rest-framework.org/

[6] *PostgreSql* [dostęp: 15.11.2020] https://www.oracle.com/pl/database/what-is-a-relational-database/

[7] *Hashids* [dostęp: 15.11.2020] https://hashids.org/

[8] *TypeScript* [dostęp: 15.11.2020] https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

[9] *HTML* [dostęp: 15.11.2020] https://devdocs.io/html/

[10] *CSS* [dostęp: 15.11.2020] https://devdocs.io/css/

[11] *SASS* [dostęp: 15.11.2020] https://sass-lang.com/guide

[12] *Angular* [dostęp: 15.11.2020] https://angular.io/guide/architecture

[13] *Angular CLI* [dostęp: 15.11.2020] https://angular.io/cli

[14] *Angular Material* [dostęp: 15.11.2020] https://material.angular.io/

[15] *MDB Angular* [dostęp: 15.11.2020] https://mdbootstrap.com/docs/angular/

[16] *Ngx-charts* [dostęp: 15.11.2020] https://swimlane.gitbook.io/ngx-charts/

[17] *Ngx-cookie-service* [dostęp: 15.11.2020] https://github.com/stevermeister/ngx-cookie-service/blob/master/README.md

[18] *Android* [dostęp: 15.11.2020] https://mobirank.pl/2020/04/14/udzial-wersji-systemu-android-w-kwietniu-2020-r/

[19] P. Späth: Learn Kotlin for Android Development. The Next Generation Language for Modern Android Apps Programming, 2019

[20] *Retrofit* [dostęp: 15.11.2020] https://square.github.io/retrofit/

[21] *Gradle* [dostęp: 15.11.2020] https://www.samouczekprogramisty.pl/wstep-do-gradle/

[22] *Android Studio* [dostęp: 15.11.2020] https://developer.android.com/studio

[23] *GIT* [dostęp: 15.11.2020] https://www.atlassian.com/git/tutorials/comparing-workflows