Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **Informatyka techniczna (ITE)**
Specjalność: **Inżynieria systemów informatycznych (INS)**

PRACA DYPLOMOWA INŻYNIERSKA

System oceniania pracy urzędników urzędu miasta

System for evaluating the work of city officials

Bartosz Sulima

Opiekun pracy
Dr. Inż. Paweł Głuchowski

Słowa kluczowe: Java, JavaScript, React, Ocena

WROCŁAW, 2023

Streszczenie

Celem pracy jest zaprojektowanie i implementacja webowej aplikacji bazodanowej na komputery z systemem Windows 10. Aplikacja w wersji dla obywatela ma umożliwić przeglądanie stanu wniosków, oceniać urzędników, z którymi miało się interakcje podczas procedowania wniosków lub poznawanie oceny wybranych urzędników. Aplikacja w wersji dla pracownika ma umożliwić wprowadzanie wniosków obywateli do bazy a także podejmowanie akcji i decyzji w ich sprawie. Część serwerowa aplikacji została napisana w języku Java z użyciem frameworka Spring, z kolei część użytkowa została napisana w języku Java Script z użyciem biblioteki React. W pracy zostanie zawarty opis podobnych rozwiązań, spis wykorzystanych technologii, projekt aplikacji oraz opis implementacji i testów.

Słowa kluczowe: Java, JavaScript, React, Ocena

Abstract

The main goal of this dissertation is the implementation of database application for personal computers running Windows 10 operating system. Application in version for the citizen will allow for browsing motions, evaluating civil servants that took part in proceeding of motions, and browsing evaluations for chosen civil servants. Application in version for worker will allow creating new motions in database and making actions and decisions related to those motions. The server part of the application will be written in Java with Spring framework. The user part of the application will be written in JavaScript with React library. This dissertation will contain description of similar solutions available on market, list of used technologies, application project, description of implementation and performed tests.

Keywords: Java, JavaScript, React, Evaluation

Spis treści

1. Wstęp . . . 10
1.1. Wprowadzenie . . . 10
1.2. Cel i zakres pracy . . . 11
2. Analiza istniejących rozwiązań . . . 12
2.1. Wstęp . . . 12
2.2. Gowork.pl . . . 12
2.3. Empik.com . . . 13
2.4. Znanylekarz.pl . . . 14
2.5. Play.google.com . . . 15
3. Wykorzystane technologie . . . 16
3.1. Wstęp . . . 16
3.2. System Windows 10 . . . 16
3.3. Git . . . 17
3.3.1. GitHub . . . 17
3.4. Java . . . 18
3.4.1. Spring Framework . . . 18
3.5. JavaScript . . . 18
3.5.1. React.js . . . 18
3.6. PostgreSQL . . . 19
3.7. IntelliJ IDEA . . . 19
3.8. Webstorm . . . 20
4. Model Systemu . . . 21
4.1. Wstęp . . . 21
4.2. Wymagania . . . 21
4.2.1. Wymagania funkcjonalne . . . 21
4.2.2. Wymagania niefunkcjonalne . . . 21
4.3. Przypadki użycia dla Aplikacji Pracownika . . . 22
4.3.1. Diagram przypadków użycia dla Aplikacji Pracownika . . . 22
4.3.2. Scenariusze przypadków użycia dla Aplikacji Pracownika . . . 23
4.4. Przypadki użycia dla Aplikacji Obywatela . . . 32
4.4.1. Diagram przypadków użycia dla Aplikacji Obywatela . . . 33
4.4.2. Scenariusze przypadków użycia dla Aplikacji Obywatela . . . 34
4.5. Baza danych . . . 38
4.5.1. Schemat Bazy danych . . . 38
4.6. Diagramy Klas . . . 39
4.7. Schemat architektury systemu . . . 41
4.8. Projekt interfejsu użytkowników . . . 42
5. Implementacja . . . 49
5.1. Wstęp . . . 49

5.2. Wzorce projektowe . . . 49
5.2.1. Repozytorium . . . 49
5.2.2. DTO . . . 50
5.3. Implementacja bazy danych . . . 50
5.4. Implementacja serwera . . . 51
5.4.1. Wstęp . . . 51
5.4.2. Tworzenie klas odpowiadających encjom w bazie danych . . . 51
5.4.3. Tworzenie klas serwisów oraz kontrolerów . . . 52
5.4.4. Autoryzacja przy użyciu JWT . . . 54
5.5. Implementacja aplikacji webowych obywatela i urzędnika . . . 56
5.5.1. Łączność aplikacji webowych z aplikacją serwerową . . . 56
5.5.2. Material UI . . . 57
6. Interfejs Obywatela . . . 59
6.1. Wstęp . . . 59
7. Interfejs Pracownika . . . 63
7.1. Wstęp . . . 63
8. Testy . . . 67
8.1. Testy Lighthouse . . . 67
8.1.1. Wstęp . . . 67
8.1.2. Wyniki . . . 67
8.2. Testy Selenium IDE . . . 68
8.3. Wstęp . . . 68
8.4. Testy integracyjne . . . 69
8.4.1. Wstęp . . . 69
8.4.2. Przykłady testów integracyjnych . . . 69
Literatura . . . 72
A. Instrukcja wdrożeniowa . . . 74
B. Opis załączonej płyty CD/DVD . . . 75

Spis rysunków

1.1. Wyniki sondażu na temat „Satysfakcji z obsługi i dostępu do informacji publicznej w urzędach Jednostek Samorządu Terytorialnego”, Źródło: [22] 10
1.2. Wyniki sondażu na temat satysfakcji publicznej w mieście South Portland, Źródło: [26] 11
2.1. Strona główna portalu gowork.pl Źródło:[11] 12
2.2. Strona główna portalu Empik.com Źródło:[9] 13
2.3. Strona główna portalu znanylekarz.pl Źródło:[13] 14
2.4. Strona główna portalu google play store Źródło:[10] 15
3.1. Udział w rynku poszczególnych wersji systemu Windows. Źródło:[28] 16
3.2. Popularność dodatkowych narzędzi używanych przez profesjonalnych programistów z portalu Stack Overflow w 2021 roku. Źródło:[24] 17
3.3. Ranking najpopularniejszych języków programowania w styczniu 2023. Źródło:[15] 19
4.1. Diagram przypadków użycia dla Aplikacji Pracownika. Źródło: Opracowanie własne. 22
4.2. Diagram przypadków użycia dla Aplikacji Obywatela. Źródło: Opracowanie własne. 33
4.3. Schemat bazy danych. Źródło: Opracowanie własne. 38
4.4. Diagram klas pakietu "model". Źródło: Opracowanie własne. 39
4.5. Diagram klas pakietów "service"i "controller". Źródło: Opracowanie własne. 40
4.6. Schemat architektury systemu. Źródło: Opracowanie własne. 41
4.7. Ekran logowania. Źródło: Opracowanie własne. 42
4.8. Ekran menu głównego. Źródło: Opracowanie własne. 42
4.9. Ekran stanu wszystkich wniosków. Źródło: Opracowanie własne. 43
4.10. Ekran wyboru wniosku do oceny. Źródło: Opracowanie własne. 43
4.11. Ekran wyboru pracownika do oceny. Źródło: Opracowanie własne. 44
4.12. Ekran przeglądania pracowników. Źródło: Opracowanie własne. 44
4.13. Ekran ocen pracownika. Źródło: Opracowanie własne. 45
4.14. Ekran szczegółów konta. Źródło: Opracowanie własne. 45
4.15. Ekran Zmiany hasła. Źródło: Opracowanie własne. 46
4.16. Ekran dodawania nowych kont użytkowników. Źródło: Opracowanie własne. 46
4.17. Ekran wyszukiwania osób. Źródło: Opracowanie własne. 47
4.18. Ekran dodawania nowych wniosków. Źródło: Opracowanie własne. 47
4.19. Ekran zarządzania nierozpatrzonymi wnioskami. Źródło: Opracowanie własne. . . 48
6.1. Ekran logowania. Źródło: Opracowanie własne. 59
6.2. Ekran menu głównego. Źródło: Opracowanie własne. 60
6.3. Ekran stanu wszystkich wniosków. Źródło: Opracowanie własne. 60
6.4. Ekran wyboru wniosku do oceny. Źródło: Opracowanie własne. 61
6.5. Ekran wyboru pracownika do oceny. Źródło: Opracowanie własne. 61
6.6. Ekran przeglądania pracowników. Źródło: Opracowanie własne. 62
6.7. Ekran ocen pracownika. Źródło: Opracowanie własne. 62
7.1. Ekran szczegółów konta. Źródło: Opracowanie własne. 63

7.2. Ekran zmiany hasła. Źródło: Opracowanie własne. 64
7.3. Ekran dodawania nowych kont użytkowników. Źródło: Opracowanie własne. 64
7.4. Ekran wyszukiwania osób. Źródło: Opracowanie własne. 65
7.5. Ekran dodawania nowych typów wniosków. Źródło: Opracowanie własne. 65
7.6. Ekran dodawania nowych wniosków. Źródło: Opracowanie własne. 66
7.7. Ekran zarządzania nierozpatrzonymi wnioskami. Źródło: Opracowanie własne. . . 66
8.1. Wyniki testu Lighthouse aplikacji obywatela. Źródło: Opracowanie własne. 67
8.2. Wyniki testu Lighthouse aplikacji pracownika. Źródło: Opracowanie własne. 67
8.3. Wyniki testu Selenium IDE. Źródło: Opracowanie własne. 68
8.4. Wyniki testów przeprowadzonych w programie Intellij IDEA. Źródło: Opracowanie własne. 69

Spis tabel

4.1. PUp#01 Zalogowanie do systemu. 23
4.2. PUp#02 Zmiana hasła do konta. 24
4.3. PUp#03 Dodawanie nowych danych osobowych do systemu. 25
4.4. PUp#04 Tworzenie nowych kont dla obywateli. 26
4.5. PUp#05 Tworzenie nowych kont dla pracowników. 27
4.6. PUp#06 Edytowanie danych osobowych. 28
4.7. PUp#07 Wprowadzanie wniosków do systemu. 29
4.8. PUp#08 Przeglądanie nierozpatrzonych wniosków istniejących w systemie. 29
4.9. PUp#09 Wprowadzanie akcji, jakie wykonano w związku z procesowaniem danego wniosku do systemu. 30
4.10. PUp#10 Podejmowanie ostatecznych decyzji w sprawie wniosku. 31
4.11. PUp#11 Zmiana hasła do konta innego użytkownika 32
4.12. PUo#01 Zalogowanie do systemu 34
4.13. PUo#02 Zmiana hasła do konta. 35
4.14. PUo#03 Przeglądanie stanu swoich wniosków. 35
4.15. PUo#04 Przeglądanie stanu swoich wniosków. 36
4.16. PUo#05 Ocenianie pracowników, którzy brali udział w procesowaniu ich wniosków po ich rozpatrzeniu. 37

Spis listingów

5.1. Interfejs Repository dla klasy "ActionType". 49
5.2. Kod klasy "WorkerDataDTO". 50
5.3. Kod tworzenia tabel "Personal_Data"oraz "Users"w bazie danych. 50
5.4. Kod dodawania konta administratora do bazy danych. 51
5.5. Właściwości aplikacji określające połączenie z bazą danych. 51
5.6. Kod klasy PersonalData. 51
5.7. Kod serwisu "MotionTypeService". 52
5.8. Fragment kodu kontrolera "UtilityControler". 53
5.9. Kod klasy JwtUtils. 54
5.10. Kod metody logowania.. 56
5.11. Kod sprawdzający stan zalogowania. 56
5.12. Kod Kod komponętu strony do zmiany hasła użytkownika. 57
8.1. Metoda testująca Dodawanie nowych danych osobowych do systemu 69
8.2. Metoda testująca dodawanie nowego konta do systemu 71

Skróty

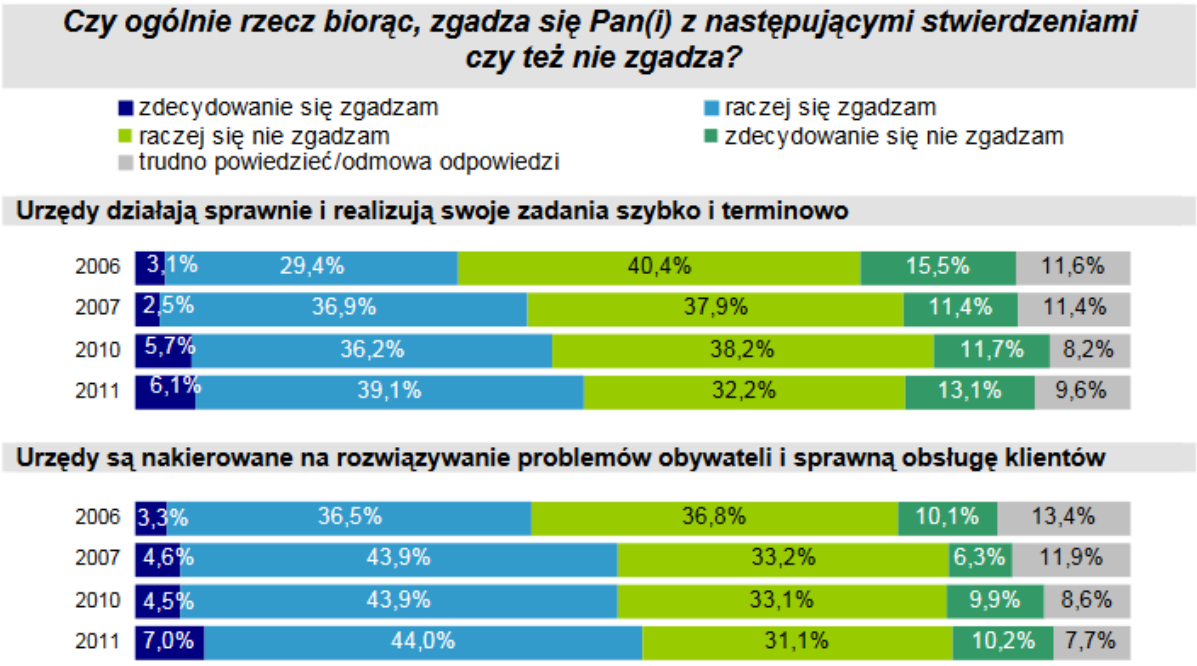
MSWiA (pl. *Ministerstwo Spraw Wewnętrznych i Administracji*)
JPA (ang. *Java Persistence API*)
ORM (ang. *Object-Relational Mapping*)
DOM (ang. *Document Object Model*)
HTML (ang. *HyperText Markup Language*)
JSX (ang. *JavaScript XML*)
PU (pl. *Przypadek użycia*)
HTTP (ang. *Hypertext Transfer Protocol*)
JWT (ang. *JSON Web Tokens*)
DTO (ang. *Data transfer object*)
JSON (ang. *JavaScript Object Notation*)
JS (ang. *JavaScript*)
UI (ang. *User interface*)

Rozdział 1

Wstęp

1.1. Wprowadzenie

Od wielu lat zadowolenie z pracy urzędników państwowych w Polsce było stosunkowo niskie. Według sondażu przeprowadzonego przez MSWiA w 2011 r. na temat „Satysfakcji z obsługi i dostępu do informacji publicznej w urzędach Jednostek Samorządu Terytorialnego” [22] około 45% badanych nie zgadza się ze zdaniem, że „Urzędy działają sprawnie i realizują swoje zadania szybko i terminowo”, 41% badanych nie zgadza się ze zdaniem, że „Urzędy są nakierowane na rozwiązywanie problemów obywateli i sprawną obsługę klientów” i chociaż według tego raportu zadowolenie Polaków z pracy lokalnych urzędników wzrasta w stosunku do poprzednich lat to dalej jest ono znacznie niższe niż zadowolenie w niektórych zachodnich miastach, przykładowo w ankiecie przeprowadzonej przez miasto South Portland w 2009 r. [26] satysfakcja z usług lokalnego urzędu miejskiego deklarowało ponad 91% procent ludzi, którzy z nich korzystali.



Rys. 1.1: Wyniki sondażu na temat „Satysfakcji z obsługi i dostępu do informacji publicznej w urzędach Jednostek Samorządu Terytorialnego”, Źródło: [22]

Question 37
If you have utilized the following city departments, offices, or services during the past 12 months, please rate your level of satisfaction based on your experiences or observations.

Answer Options	Did Not Use	Very Satisfied	Satisfied	Neither Satisfied or Dissatisfied	Dissatisfied	Very Dissatisfied	Response Count
City Hall Services	14.6%	39.6%	38.4%	4.5%	1.9%	1.1%	268

Adjusted for Did Not Use

Very Satisfied	Satisfied	Neither Satisfied or Dissatisfied	Dissatisfied	Very Dissatisfied
46.3%	45.0%	5.2%	2.2%	1.3%

Rys. 1.2: Wyniki sondażu na temat satysfakcji publicznej w mieście South Portland, Źródło: [26]

Jedną z głównych przyczyn niezadowolenia Polaków z pracy urzędników jest brak rozliczania urzędników z ich pracy, co w połączeniu z faktem, że urzędy mimowolnie sprawują monopol na usługi administracyjne, z których obywateli muszą korzystać, co powoduje, że ani urzędnicy, ani władze lokalne odpowiadające za te urzędy nie czują potrzeby, żeby polepszać swoje usługi. Brak sumiennego rozliczania pracy urzędników w dłuższym okresie może powodować nie tylko brak poprawy standardów obsługi obywateli, ale także ich stopniowe pogorszenie.

W związku z tym pojawił się pomysł zrealizowania aplikacji, która dałaby obywatelom szansę wyrażenia swojego zdania na temat pracy urzędników, z którymi mieli oni do czynienia i poprzez upublicznienie tych ocen spowodowałaby oddolną presję w urzędnikach do samo poprawy a także ułatwiła władzom lokalnym podejmowanie decyzji personalnych w celu poprawy pracy urzędu.

1.2. Cel i zakres pracy

Celem pracy jest zaprojektowanie i zaimplementowanie webowej bazodanowej aplikacji użytkownika systemu:

1. W wersji dla pracownika: Aplikacja umożliwia wprowadzanie wniosków obywateli do bazy i/lub podejmowanie akcji i decyzji w ich sprawie.
2. W wersji dla obywatela: Aplikacja umożliwia sprawdzanie stanu swoich spraw i oceniania urzędników, z którymi miało się interakcje lub poznawanie ocen wybranych urzędników.

W ramach celu ogólnego wyznaczone zostały następujące cele cząstkowe:

- Analiza istniejących rozwiązań.
- Opis technologii wykorzystanych podczas implementacji systemu oceniania pracy urzędników urzędu miasta.
- Opracowanie modelu systemu.
- Opis Implementacji.
- Przeprowadzenie testów.

Rozdział 2

Analiza istniejących rozwiązań

2.1. Wstęp

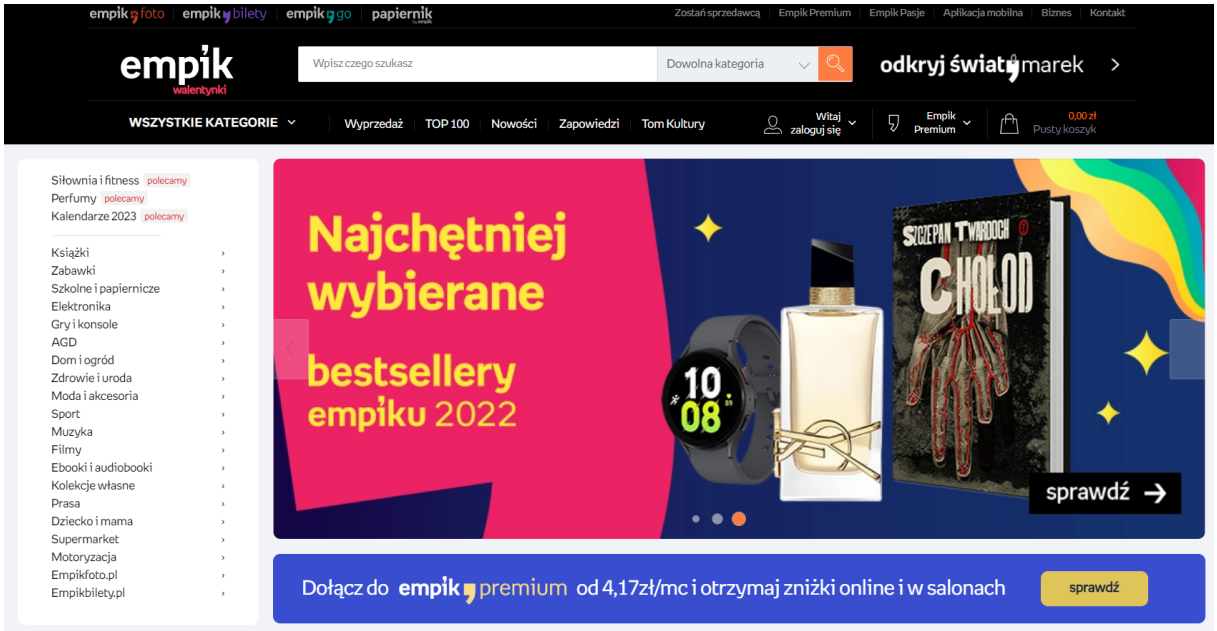
Poniższa analiza została stworzona w celu określenie zalet i wad istniejących na rynku rozwiązań do oceniania osób lub produktów. Do wszystkich wymienionych rozwiązań można uzyskać dostęp poprzez wszystkie popularne przeglądarki internetowe na platformie Windows 10.

2.2. Gowork.pl

Rys. 2.1: Strona główna portalu gowork.pl Źródło:[11]

Gowork.pl jest stroną do przeglądania ofert pracy. Pracodawcy mogą na niej wystawiać oferty pracy a potencjalni pracownicy mogą te oferty przeglądać i składać aplikacje. Strona posiada intuicyjny i przejrzysty interfejs użytkownika w biało-czerwonej kolorystyce. Byli i potencjalni pracownicy mogą także oceniać pracodawców poprzez zostawianie komentarzy pod ich profilami. Według portalu similarweb[1] w grudniu 2022 roku stronę gowork.pl odwiedziło 6,6 miliona użytkowników. Analiza portalu wappalyzer[12] wykazała, że w skład technologii użytych do stworzenia strony wchodzą między innymi: PHP, MySQL, Bootstrap.

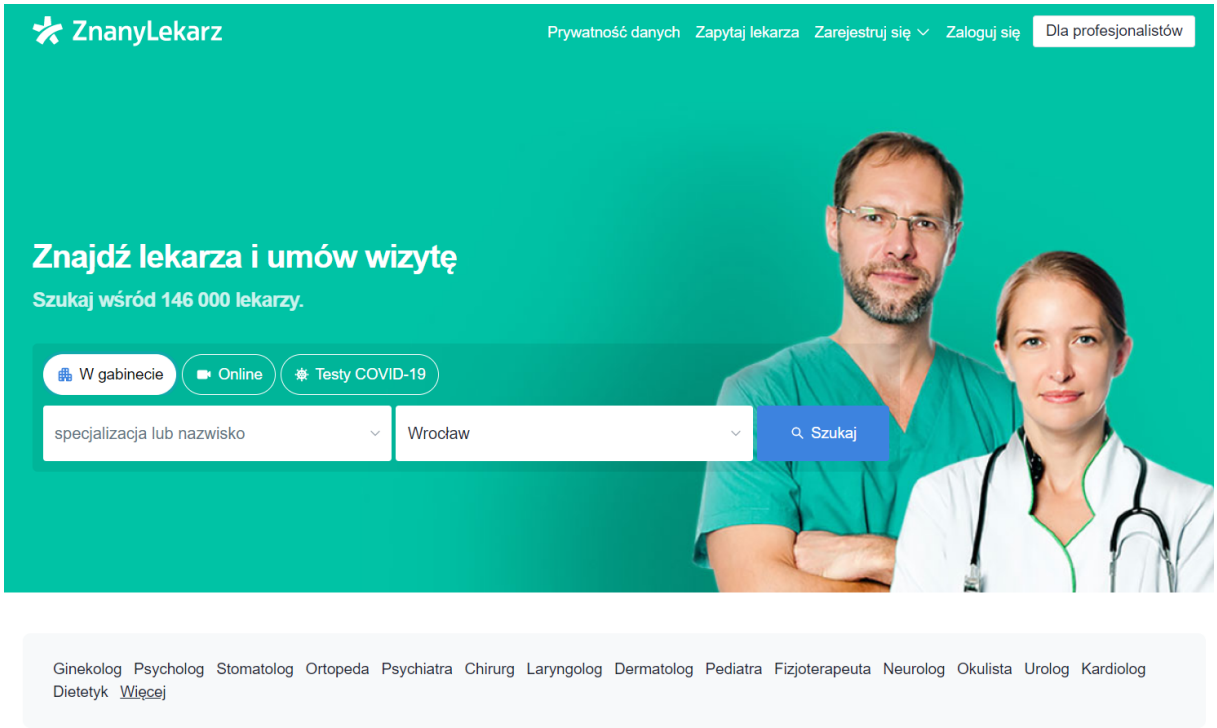
2.3. Empik.com



Rys. 2.2: Strona główna portalu Empik.com Źródło:[9]

Empik.com jest sklepem internetowym, który umożliwia użytkownikom zakup wszelkiego rodzaju produktów od książek i filmów po produkty kosmetyczne i elektronikę. Strona posiada intuicyjny interfejs użytkownika w biało-czarnej kolorystyce, stosunkowo mały rozmiar czcionki może jednak sprawiać problemy z czytelnością. Użytkownicy mogą oceniać produkty po ich zakupie poprzez wystawienie im oceny o 1 do 5 gwiazdek i napisanie komentarza. Według portalu similarweb[1] w grudniu 2022 roku stronę empik.com odwiedziło 22,5 miliona użytkowników. Analiza portalu wappalyzer[12] wykazała, że w skład technologii użytych do stworzenia strony wchodzą między innymi: Java, JavaScript React, AngularJS, Bootstrap.

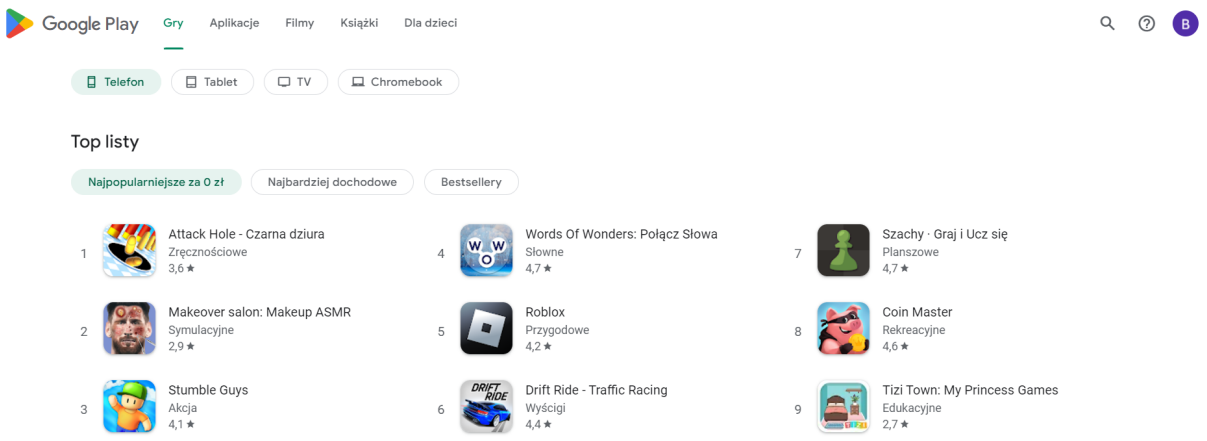
2.4. Znanylekarz.pl



Rys. 2.3: Strona główna portalu znanylekarz.pl Źródło:[13]

Znanylekarz.pl jest stroną, która ułatwia komunikacje między pajetami a lekarzami. Strona posiada przejrzysty i intuicyjny interfejs użytkownika w kolorystyce zielono-białej. Pacjenci mogą zadawać lekarzom pytania a także przeglądać oceny dostępnych lekarzy w danym mieście i umawiać się z nimi na wizyty. Pacjenci mogą także oceniać lekarzy po wizytach poprzez wystawienie im oceny o 1 do 5 gwiazdek i napisanie komentarza. Według portalu similarweb[1] w grudniu 2022 roku stronę znanylekarz.pl odwiedziło 11,3 miliona użytkowników. Analiza portalu wappalyzer[12] wykazała, że w skład technologii użytych do stworzenia strony wchodzą między innymi: JavaScript, Vue.js, Varnish.

2.5. Play.google.com



Rys. 2.4: Strona główna portalu google play store Źródło:[10]

Play.google.com jest stroną, która umożliwia użytkownikom zakup i instalacje gier, aplikacji a także książek i filmów. Strona posiada przejrzysty i intuicyjny interfejs użytkownika w białej kolorystyce. Użytkownicy mogą oceniać produkty po ich zakupie poprzez wystawienie im oceny o 1 do 5 gwiazdek i napisanie komentarza. Według portalu similarweb[1] w grudniu 2022 roku stronę play.google.com odwiedziło 1,2 miliarda użytkowników. Analiza portalu wappalyzer[12] wykazała, że w skład technologii użytych do stworzenia strony wchodzą między innymi: JavaScript, React.

Rozdział 3

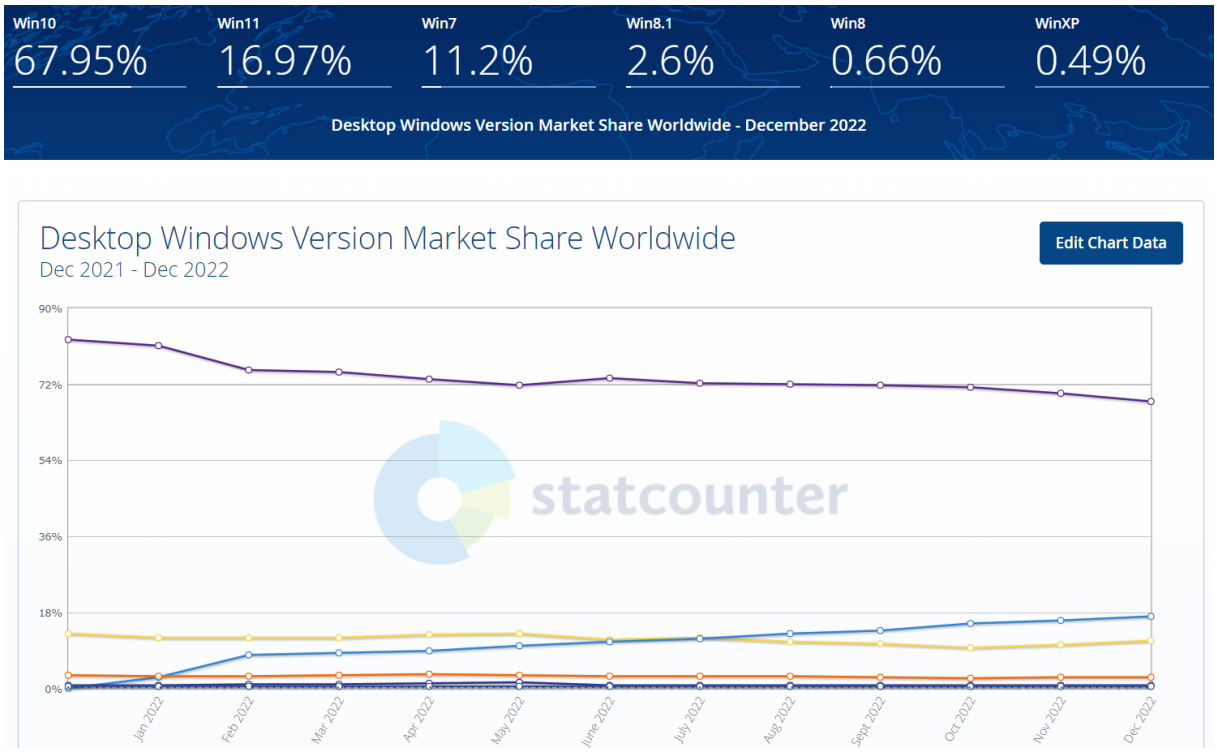
Wykorzystane technologie

3.1. Wstęp

W poniższym rozdziale opisane zostaną technologie wykorzystane podczas projektowania, a następnie implementacji aplikacji obywatela i pracownika urzędu.

3.2. System Windows 10

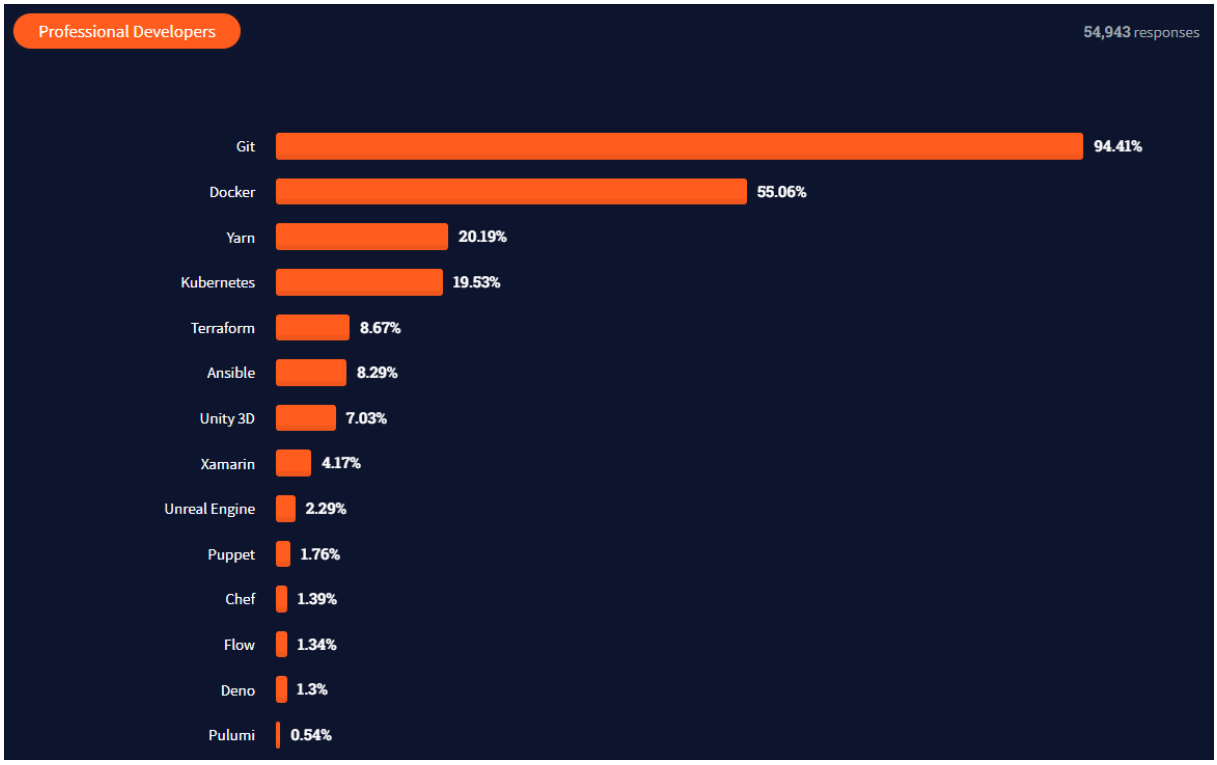
Windows 10 – wersja systemu operacyjnego Microsoft Windows, która została wydana 29 lipca 2015 roku. W grudniu 2022 r. system Windows 10 zajmował 67,95% udziału w rynku systemów operacyjnych marki Windows[28]. Cały proces tworzenia i testowania pracy został przeprowadzony na systemie Windows 10, jest to minimalna wersja systemu Windows, na której należy uruchomić aplikację.



Rys. 3.1: Udział w rynku poszczególnych wersji systemu Windows. Źródło:[28]

3.3. Git

Git – rozproszony system kontroli wersji dostępny na wszystkich popularnych platformach deweloperskich na zasadach darmowej licencji.[27] Git stanowi jedno z najbardziej popularnych narzędzi wykorzystywanym przez programistów z całego świata w 2021 roku korzystało z niego ponad 93% wszystkich programistów i ponad 94% zawodowych programistów[24]. W pracy wykorzystano system kontroli wersji, a dokładnie Git w wersji 2.39.1.



Rys. 3.2: Popularność dodatkowych narzędzi używanych przez profesjonalnych programistów z portalu Stack Overflow w 2021 roku. Źródło:[24]

3.3.1. GitHub

GitHub - serwis internetowy przeznaczony do przechowywania projektów programistycznych, wykorzystujący system kontroli wersji Git. Serwis działa od kwietnia 2008 roku[30]. W styczniu 2023 roku ilość użytkowników serwisu przekroczyła 100 milionów[17]. Serwis GitHub został wykorzystany w celu zabezpieczenia się przed lokalną utratą danych spowodowaną czynnikami losowymi.

3.4. Java

Java – Oparty na klasach nakierowany na programowanie obiektowe język programowania. Język Java został po raz pierwszy wprowadzony na rynek w 1995 roku przez firmę „Sun microsystems” i jest rozwijany do dzisiaj[4]. W styczniu 2023 roku był to drugi najbardziej popularny język programowania[15]. W pracy wykorzystano Język Java w wersji 19, jest to minimalna wersja, na której należy uruchomić aplikację.

3.4.1. Spring Framework

Spring – Framework służący do tworzenia aplikacji w języku Java. Spring oferuje dużą swobodę w tworzeniu rozwiązań, a jednocześnie posiada bogatą dokumentację i zawiera rozwiązania wielu często występujących w programowaniu. W pracy użyto Frameworka Spring w wersji 2.7.5.

Spring Boot JPA

Spring Boot JPA – specyfikacja języka Java do nadzorowania relacyjnych danych w aplikacjach. Specyfikacja ta pozwala na komunikację pomiędzy obiektami w Javie i relacyjną bazą danych. JPA wspiera odwzorowywanie Object-Relation Mapping (ORM), czyli konwersje obiektów występujących w aplikacji, których nie zawsze można zamienić bezpośrednio na obiekty występujące w bazie danych[19].

3.5. JavaScript

JavaScript - język programowania będący podstawą tworzenia współczesnych stron internetowych. W 2022 roku 98% wszystkich stron internetowych na świecie używało języka JavaScript[25]. W styczniu 2023 roku był to trzeci najbardziej popularny język programowania [15]. Wszystkie popularne przeglądarki dostępne na rynku wspierają wykonywanie kodu w JavaScripcie na urządzeniach użytkowników. W pracy użyto JavaScript w wersji ECMAScript 2021.

3.5.1. React.js

React.js – biblioteka języka programowania JavaScript, wykorzystywana do tworzenia graficznych interfejsów aplikacji internetowych. Jedną z cech charakterystycznych biblioteki React.js jest wirtualny DOM (Document Object Model). React przechowuje cały DOM aplikacji w pamięci, po dokonaniu zmian sprawdza różnice między wirtualnym i prawdziwym DOM-em i aktualizuje zmiany. Kolejną z cech szczególnych React jest język JSX. Jest to nakładka na JavaScript, która umożliwia wstawiania kodu HTML (albo komponentów React) bezpośrednio do kodu[18].

Worldwide, Jan 2023 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.93 %	-0.9 %
2		Java	16.78 %	-1.3 %
3		JavaScript	9.63 %	+0.5 %
4	↑	C#	6.99 %	-0.3 %
5	↓	C/C++	6.9 %	-0.5 %
6		PHP	5.29 %	-0.8 %
7		R	4.03 %	-0.2 %
8	↑↑↑	TypeScript	2.79 %	+1.0 %
9		Swift	2.23 %	+0.3 %
10	↓↓	Objective-C	2.2 %	-0.1 %

Rys. 3.3: Ranking najpopularniejszych języków programowania w styczniu 2023. Źródło:[15]

3.6. PostgreSQL

PostgreSQL – System zarządzania relacyjnymi bazami danych używający i rozszerzający język SQL[7]. W ankiecie portalu Stack Overflow z 2021 roku PostgreSQL jest najbardziej pożądanym przez programistów systemem bazodanowym[24]. PostgreSQL został użyty w celu stworzenia i przechowywania bazy danych używanej w trakcie działania aplikacji. W pracy wykorzystano System PostgreSQL w wersji 14, jest to minimalna wersja, na której należy uruchomić aplikację.

pgAdmin4

pgAdmin4 - Program do zarządzania i pracy użytkownika z bazą danych PostgreSQL. program ten jest instalowany automatycznie z PostgreSQL.

3.7. IntelliJ IDEA

IntelliJ IDEA – Zintegrowane środowisko programistyczne dla języka Java posiadające zaawansowany edytor kodu i wbudowane narzędzia pomocne w tworzeniu i testowaniu kodu[5]. IntelliJ IDEA został użyty w celu tworzenia kodu serwera pośredniczącego w wymianie danych pomiędzy bazą danych a aplikacjami obywatela i pracownika. W pracy wykorzystano program IntelliJ IDEA w wersji 2021.3.

3.8. Webstorm

Webstorm - Zintegrowane środowisko programistyczne dla języka JavaScript posiadające zaawansowany edytor kodu i wbudowane narzędzia pomocne w tworzeniu i testowaniu kodu[6]. Webstorm został użyty w celu tworzenia kodu aplikacji webowych obywatela i pracownika. W pracy wykorzystano Webstorm w wersji 2021.3. W pracy wykorzystano program IntelliJ IDEA w wersji 2022.2.3.

Rozdział 4

Model Systemu

4.1. Wstęp

Poniższy rozdział zawiera opis modelu systemu aplikacji obywatela i pracownika urzędu. Składają się na niego: opis wymagań funkcjonalnych i niefunkcjonalnych, przypadki użycia, projekt struktury bazy danych, diagram klas aplikacji oraz projekt interfejsu użytkownika.

4.2. Wymagania

4.2.1. Wymagania funkcjonalne

1. Obywatele mogą się zalogować.
2. Obywatele mogą zmienić hasło do swojego konta.
3. Obywatele mogą przeglądać stan swoich wniosków.
4. Obywatele mogą przeglądać oceny Pracowników.
5. Obywatele mogą Oceniać Pracowników, którzy brali udział w procesowaniu ich wniosków po ich rozpatrzeniu.
6. Pracownicy mogą się zalogować.
7. Pracownicy mogą zmienić hasło do swojego konta.
8. Pracownicy mogą dodawać nowe dane osobowe do systemu.
9. Pracownicy mogą tworzyć nowe konta dla obywateli.
10. Pracownicy mogą tworzyć nowe konta dla pracowników.
11. Pracownicy mogą edytować dane osobowe obywateli.
12. Pracownicy mogą wprowadzać wnioski do systemu.
13. Pracownicy mogą przeglądać nierozpatrzone wnioski istniejące w systemie.
14. Pracownicy mogą wprowadzać akcje, jakie wykonali w związku z procesowaniem danego wniosku do systemu.
15. Pracownicy mogą podejmować ostateczną decyzję w sprawie wniosku.
16. Pracownicy mogą zmieniać hasła do kont innych użytkowników.

4.2.2. Wymagania niefunkcjonalne

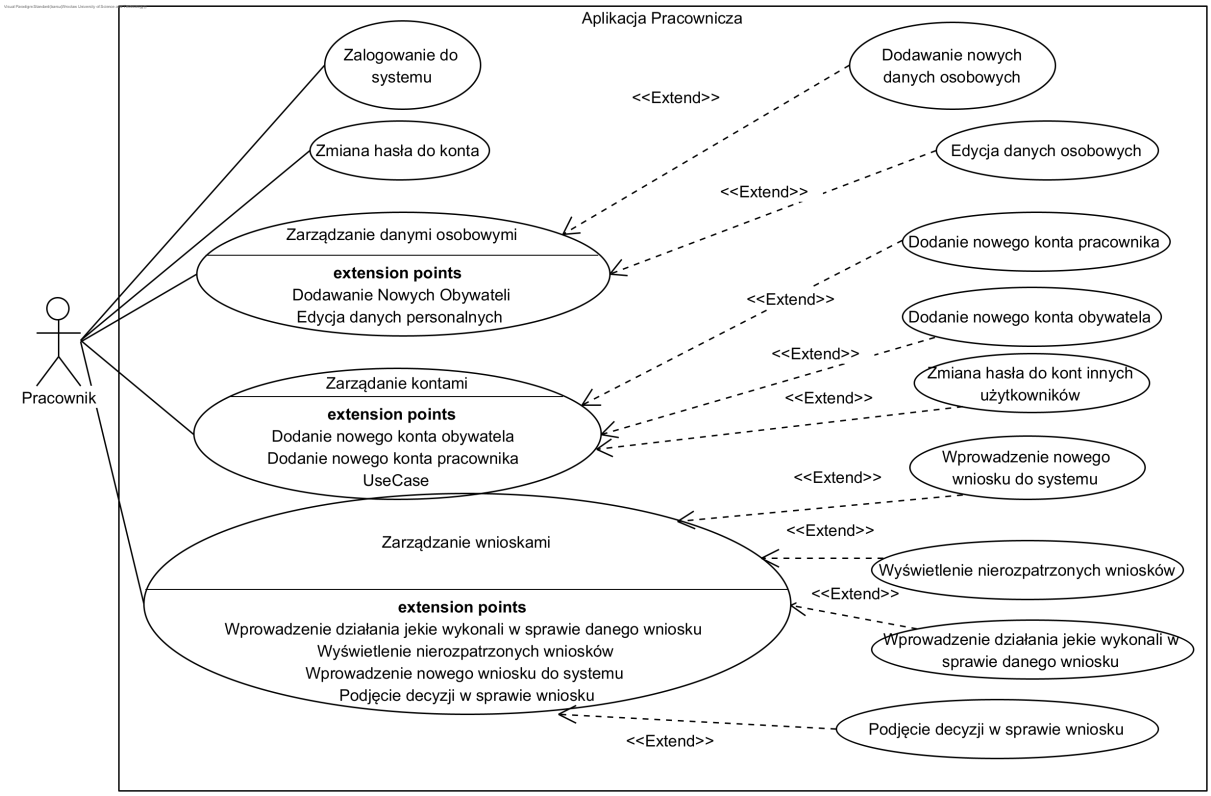
1. Musi działać na systemie Windows 10.
2. Aby korzystać z systemu, pracownicy i obywatele muszą być zalogowani.
3. System powinien rejestrować każde logowanie w celu kontroli bezpieczeństwa.
4. Hasła przechowywane w bazie danych muszą być haszowane.

4.3. Przypadki użycia dla Aplikacji Pracownika

Przypadki użycia przedstawiają działania, jakie będą wykonywane przez użytkowników podczas korzystania z aplikacji[16]. Korzystając z wymagań funkcjonalnych systemu oceniania pracy urzędników urzędu miasta zaprojektowano przypadki użycia aplikacji pracownika o następującym składzie:

PUp#01 Zalogowanie do systemu.
PUp#02 Zmiana hasła do konta.
PUp#03 Dodawanie nowych danych osobowych do systemu.
PUp#04 Tworzenie nowych kont dla obywateli.
PUp#05 Tworzenie nowych kont dla pracowników.
PUp#06 Edytowanie danych osobowych.
PUp#07 Wprowadzanie wniosków do systemu.
PUp#08 Przeglądanie nierozpatrzonych wniosków istniejących w systemie.
PUp#09 Wprowadzanie akcji, jakie wykonano w związku z procesowaniem danego wniosku do systemu.
PUp#10 Podejmowanie ostatecznych decyzji w sprawie wniosku.
PUp#11 Zmiana hasła do konta innego użytkownika.

4.3.1. Diagram przypadków użycia dla Aplikacji Pracownika



Rys. 4.1: Diagram przypadków użycia dla Aplikacji Pracownika. Źródło: Opracowanie własne.

4.3.2. Scenariusze przypadków użycia dla Aplikacji Pracownika

Tab. 4.1: PUp#01 Zalogowanie do systemu.

Aktor	Pracownik.
Opis	Pracownik chce się zalogować do aplikacji.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest niezalogowany. • Widoczny jest ekran logowania.
Przebieg Główny	1. Pracownik wpisuje login i hasło do swojego konta i klika przycisk do zalogowania. 2. Jeśli dane konta są poprawne pracownik zostaje przeniesiony do ekranu menu głównego.
Przebieg Alternatywny	2 a) W przypadku podania złych danych logowania zostaje wyświetlony komunikat o błędzie. Powrót do kroku 1.
Warunki Końcowe	• Pracownik widzi ekran menu głównego.

Tab. 4.2: PUp#02 Zmiana hasła do konta.

Aktor	Pracownik.
Opis	Pracownik chce zmienić hasło do swojego konta.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Szczegóły konta”. 2. Zostaje wyświetlony ekran ze szczegółami konta użytkownika. 3. Pracownik klika przycisk „Zmień hasło”. 4. Zostaje wyświetlony ekran zmiany hasła. 5. Na ekranie zmiany hasła pracownik wpisuje stare i nowe hasło, po czym klika przycisk „Zmień hasło”. 6. Jeżeli podane dane są poprawne to zostaje wyświetlony komunikat o pomyślnej zmianie hasła i pracownik zostaje przeniesiony do ekranu logowania.
Przebieg Alternatywny	6 a) Jeżeli podane dane są niepoprawne to zostaje wyświetlony komunikat o błędzie. Powrót do punktu 5.
Warunki Końcowe	• Zmiana hasła do konta.

Tab. 4.3: PUp#03 Dodawanie nowych danych osobowych do systemu.

Aktor	Pracownik.
Opis	Pracownik chce dodać nowe dane osobowe do systemu.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Dodawanie nowych danych osobowych”. 2. Zostaje wyświetlony ekran dodawania danych osobowych. 3. Pracownik wprowadza w wyznaczone pola dane osobowe, które chce dodać do systemu, po czym klika przycisk „Dodaj osobę”. 4. Jeżeli wszystkie dane osobowe zostały wprowadzone i nie ma już w bazie danych osoby z takim samym peselem to zostaje wyświetlony komunikat o udanym dodaniu osoby do systemu.
Przebieg Alternatywny	4 a) Jeżeli nie wszystkie dane osobowe zostały wypełnione i/lub w bazie danych znajduje się już osoba z takim samym peselem to zostaje wyświetlony komunikat o błędzie. Powrót do punktu 3.
Warunki Końcowe	• Dodanie nowych danych osobowych do systemu.

Tab. 4.4: PUp#04 Tworzenie nowych kont dla obywateli.

Aktor	Pracownik.
Opis	Pracownik chce stworzyć nowe konto dla obywatela.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Dodawanie nowych kont”. 2. Zostaje wyświetlony ekran dodawania nowych kont. 3. Pracownik wprowadza w wyznaczone pola login, hasło, pesel osoby, do której ma to konto należeć a także wybiera typ konta „Obywatel”, po czym klika przycisk „Dodaj”. 4. Jeżeli wszystkie dane zostały wprowadzone, nie ma już w bazie danych konta z takim samym loginem i nie ma już w bazie danych konta z uprawnieniami obywatela dla danego peselu to zostaje wyświetlony komunikat o udanym dodaniu konta do systemu.
Przebieg Alternatywny	4 a) Jeżeli nie wszystkie dane zostały wypełnione i/lub w bazie danych znajduje się już konto z takim samym loginem i/lub jest w bazie danych konto z uprawnieniami obywatela dla danego peselu to zostaje wyświetlony komunikat o błędzie. Powrót do punktu 3.
Warunki Końcowe	• Dodanie nowego konta obywatela do systemu.

Tab. 4.5: PUp#05 Tworzenie nowych kont dla pracowników.

Aktor	Pracownik.
Opis	Pracownik chce stworzyć nowe konto dla pracownika.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Dodawanie nowych kont”. 2. Zostaje wyświetlony ekran dodawania nowych kont. 3. Pracownik wprowadza w wyznaczone pola login, hasło, pesel osoby, do której ma to konto należeć a także wybiera typ konta „Pracownik”, po czym klika przycisk „Dodaj”. 4. Jeżeli wszystkie dane zostały wprowadzone, nie ma już w bazie danych konta z takim samym loginem i nie ma już w bazie danych konta z uprawnieniami pracownika dla danego peselu to zostaje wyświetlony komunikat o udanym dodaniu konta do systemu.
Przebieg Alternatywny	4 a) Jeżeli nie wszystkie dane zostały wypełnione i/lub w bazie danych znajduje się już konto z takim samym loginem i/lub jest w bazie danych konto z uprawnieniami pracownika dla danego peselu to zostaje wyświetlony komunikat o błędzie. Powrót do punktu 3.
Warunki Końcowe	• Dodanie nowego konta pracownika do systemu.

Tab. 4.6: PUp#06 Edytowanie danych osobowych.

Aktor	Pracownik.
Opis	Pracownik chce zmienić wartości danych osobowych dla konkretnej osoby.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Wyszukaj osobę”. 2. Zostaje wyświetlony ekran wyszukiwania osób. 3. Pracownik wprowadza pesel osoby, której dane chce zmienić, po czym klika przycisk z ikoną lupy. 4. Jeżeli osoba z danym peselem znajduje się w bazie danych to zostaje wyświetlony ekran zawierający informacje o koncie. 5. Pracownik klika przycisk „Edytuj dane”. 6. Zostaje wyświetlone okno edycji danych zawierające dane osobowe, które można edytować. 7. Pracownik edytuje dane, po czym kilka przycisk” Zapisz”. 8. Jeżeli zmienione dane są poprawne to zostaje wyświetlony komunikat o udanej modyfikacji danych osobowych.
Przebieg Alternatywny	4 a) Jeżeli osoba z danym peselem nie znajduje się w bazie danych to wracamy do punktu 3. 8 a) Jeżeli zmienione dane są nie poprawne to zostaje wyświetlony komunikat o błędzie. Powrót do punktu 5.
Warunki Końcowe	• Wartości danych osobowych zostały zmienione.

Tab. 4.7: PUp#07 Wprowadzanie wniosków do systemu.

Aktor	Pracownik.
Opis	Pracownik chce wprowadzić nowy wniosek do systemu.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Dodawanie nowego wniosku”. 2. Zostaje wyświetlony ekran dodawania nowego wniosku. 3. Pracownik wprowadza pesel wnioskodawcy a także wybiera wniosek z listy wniosków, po czym klika przycisk „Dodaj”. 4. Jeżeli w bazie danych istnieje osoba z podanym peselem to zostaje wyświetlony komunikat o udanym dodaniu nowego wniosku.
Przebieg Alternatywny	4 a) Jeżeli w bazie danych nie istnieje osoba z podanym peselem to zostaje wyświetlony komunikat o Błędzie. Powrót do punktu 3.
Warunki Końcowe	• Dodanie nowego wniosku do systemu.

Tab. 4.8: PUp#08 Przeglądanie nierozpatrzonych wniosków istniejących w systemie.

Aktor	Pracownik.
Opis	Pracownik chce przeglądać nierozpatrzone wnioski istniejące w systemie.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Zarządzaj nierozpatrzonymi wnioskami”. 2. Zostaje wyświetlony ekran wyszukiwania osób. 3. Pracownik wprowadza pesel osoby, której wnioski chce przejrzeć. 4. Jeżeli osoba z danym peselem znajduje się w bazie danych to zostaje wyświetlony ekran z listą nierozpatrzonych wniosków dla danej osoby.
Przebieg Alternatywny	4 a) Jeżeli osoba z danym peselem nie znajduje się w bazie danych to wracamy do punktu 3.
Warunki Końcowe	• Wyświetlenie listy nierozpatrzonych wniosków dla danej osoby.

Tab. 4.9: PUp#09 Wprowadzanie akcji, jakie wykonano w związku z procesowaniem danego wniosku do systemu.

Aktor	Pracownik.
Opis	Pracownik chce wprowadzić akcje jakie wykonał w związku z procesowaniem danego wniosku do systemu.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Zarządzaj nierozpatrzonymi wnioskami”. 2. Zostaje wyświetlony ekran wyszukiwania osób. 3. Pracownik wprowadza pesel osoby, której wnioski chce przejrzeć. 4. Jeżeli osoba z danym peselem znajduje się w bazie danych to zostaje wyświetlony ekran listę nierozpatrzonych wniosków dla danej osoby. 5. Pracownik klika przycisk „Skonsultuj wnioskodawcę” lub przycisk „Proceduj wniosek”. 6. Zostaje wyświetlony komunikat o pomyślnym wprowadzeniu akcji do systemu.
Przebieg Alternatywny	4 a) Jeżeli osoba z danym peselem nie znajduje się w bazie danych to wracamy do punktu 3.
Warunki Końcowe	• Wprowadzenie akcji wykonanej w związku z procesowaniem wniosku do systemu.

Tab. 4.10: PUp#10 Podejmowanie ostatecznych decyzji w sprawie wniosku.

Aktor	Pracownik.
Opis	Pracownik chce podjąć ostateczną decyzję w sprawie danego wniosku.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Zarządzaj nierozpatrzonymi wnioskami”. 2. Zostaje wyświetlony ekran wyszukiwania osób. 3. Pracownik wprowadza pesel osoby, której wnioski chce przejrzeć. 4. Jeżeli osoba z danym peselem znajduje się w bazie danych to zostaje wyświetlony ekran listę nierozpatrzonych wniosków dla danej osoby. 5. Pracownik klika przycisk „Odrzuć wniosek” lub przycisk „Zatwierdź wniosek”. 6. Zostaje wyświetlony komunikat o pomyślnym zaakceptowaniu lub odrzuceniu wniosku.
Przebieg Alternatywny	4 a) Jeżeli osoba z danym peselem nie znajduje się w bazie danych to wracamy do punktu 3.
Warunki Końcowe	• Podjęcie ostatecznej decyzji w sprawie wniosku.

Tab. 4.11: PUp#11 Zmiana hasła do konta innego użytkownika

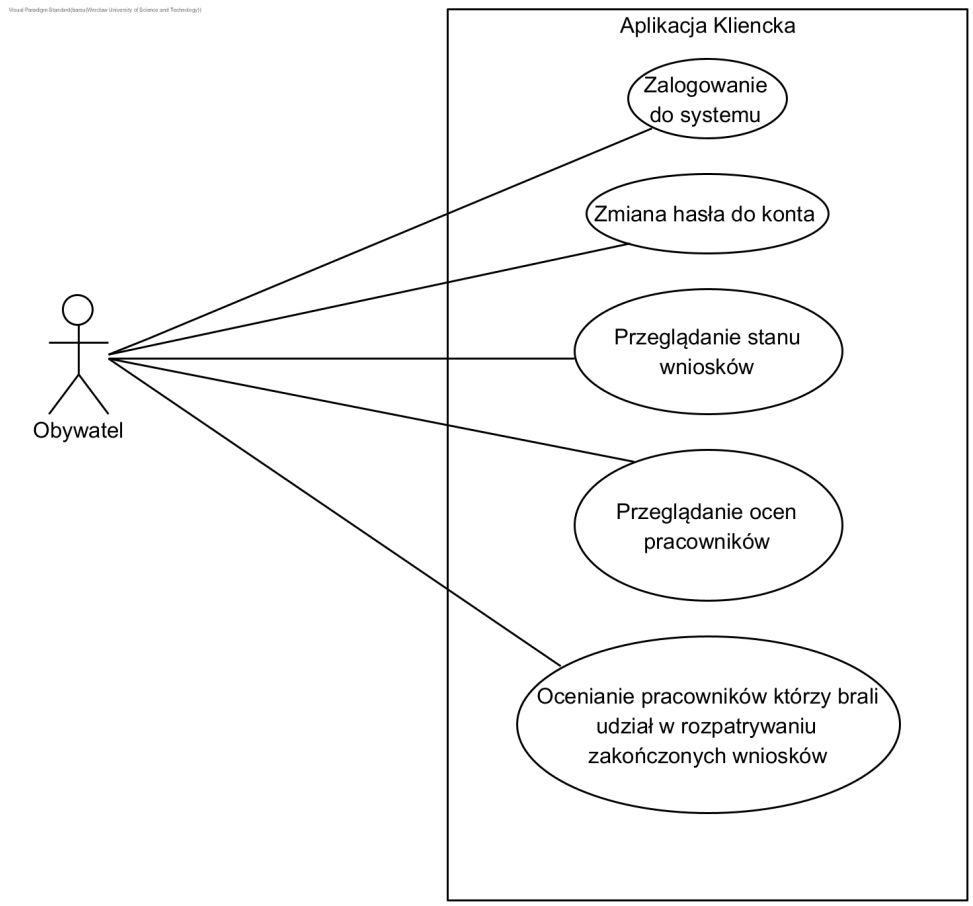
Aktor	Pracownik.
Opis	Pracownik chce zmienić hasło do konta innego użytkownika.
Warunki Początkowe	• Aplikacja Pracownika jest uruchomiona. • Pracownik jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego pracownik klika przycisk „Administracyjna zmiana hasła”. 2. Zostaje wyświetlony ekran zmiany hasła. 3. Na ekranie zmiany hasła pracownik wpisuje login, dla konta, którego hasło chce zmienić i nowe hasło, po czym klika przycisk „Zmień hasło”. 4. Jeżeli konto o podanym loginie istnieje w bazie to zostaje wyświetlony komunikat o pomyślnej zmianie hasła.
Przebieg Alternatywny	4 a) Jeżeli konto o podanym loginie nie istnieje w bazie to zostaje wyświetlony komunikat błędzie. Powrót do punktu 3.
Warunki Końcowe	• Zmiana hasła do konta innego użytkownika.

4.4. Przypadki użycia dla Aplikacji Obywatela

Przypadki użycia przedstawiają działania, jakie będą wykonywane przez użytkowników podczas korzystania z aplikacji[16]. Korzystając z wymagań funkcjonalnych systemu oceniania pracy urzędników urzędu miasta zaprojektowano przypadki użycia aplikacji obywatela o następującym składzie:

PUo#01 Zalogowanie do systemu.
PUo#02 Zmiana hasła do konta.
PUo#03 Przeglądanie stanu swoich wniosków.
PUo#04 Przeglądanie ocen Pracowników.
PUo#05 Ocenianie Pracowników, którzy brali udział w procesowaniu ich wniosków po ich rozpatrzeniu.

4.4.1. Diagram przypadków użycia dla Aplikacji Obywatela



Rys. 4.2: Diagram przypadków użycia dla Aplikacji Obywatela. Źródło: Opracowanie własne.

4.4.2. Scenariusze przypadków użycia dla Aplikacji Obywatela

Tab. 4.12: PUo#01 Zalogowanie do systemu

Aktor	Obywatel.
Opis	Obywatel chce się zalogować do aplikacji.
Warunki Początkowe	• Aplikacja Obywatela jest uruchomiona. • Obywatel jest niezalogowany. • Widoczny jest ekran logowania.
Przebieg Główny	1. Obywatel wpisuje login i hasło do swojego konta i klika przycisk do zalogowania. 2. Jeśli dane konta są poprawne to użytkownik zostaje przeniesiony do ekranu menu głównego.
Przebieg Alternatywny	2 a) W przypadku podania złych danych logowania zostaje wyświetlony komunikat o błędzie. Powrót do kroku 1.
Warunki Końcowe	• Obywatel widzi ekran menu głównego.

Tab. 4.13: PUo#02 Zmiana hasła do konta.

Aktor	Obywatel.
Opis	Obywatel chce zmienić hasło do swojego konta.
Warunki Początkowe	• Aplikacja Obywatela jest uruchomiona. • Obywatel jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego obywatel klika przycisk „Szczegóły konta”. 2. Zostaje wyświetlony ekran ze szczegółami konta użytkownika. 3. Obywatel klika przycisk „Zmień hasło”. 4. Zostaje wyświetlony ekran zmiany hasła. 5. Na ekranie zmiany hasła obywatel wpisuje stare i nowe hasło, po czym klika przycisk „Zmień hasło”. 6. Jeżeli podane dane są poprawne to zostaje wyświetlony komunikat o pomyślnej zmianie hasła i użytkownik zostaje przeniesiony do ekranu logowania.
Przebieg Alternatywny	6 a) Jeżeli podane dane są niepoprawne to zostaje wyświetlony komunikat o błędzie. Powrót do punktu 5.
Warunki Końcowe	• Zmiana hasła do konta.

Tab. 4.14: PUo#03 Przeglądanie stanu swoich wniosków.

Aktor	Obywatel.
Opis	Obywatel chce przejrzeć stan swoich wniosków.
Warunki Początkowe	• Aplikacja Obywatela jest uruchomiona. • Obywatel jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego obywatel klika przycisk „Sprawdź status wniosków”. 2. Zostaje wyświetlony ekran pokazujący stan wszystkich wniosków należących do obywatela.
Warunki Końcowe	• Wyświetlenie listy wniosków należących do obywatela i ich statusów.

Tab. 4.15: PUo#04 Przeglądanie stanu swoich wniosków.

Aktor	Obywatel.
Opis	Obywatel chce przejrzeć stan swoich wniosków.
Warunki Początkowe	• Aplikacja Obywatela jest uruchomiona. • Obywatel jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego obywatel klika przycisk „Sprawdź status wniosków”. 2. Zostaje wyświetlony ekran pokazujący stan wszystkich wniosków należących do obywatela.
Warunki Końcowe	• Wyświetlenie listy wniosków należących do obywatela i ich statusów.

Tab. 4.16: PUo#05 Ocenianie pracowników, którzy brali udział w procesowaniu ich wniosków po ich rozpatrzeniu.

Aktor	Obywatel.
Opis	Obywatel chce ocenić pracownika, który brał udział w procesowaniu jego wniosku.
Warunki Początkowe	• Aplikacja Obywatela jest uruchomiona. • Obywatel jest zalogowany. • Widoczny jest ekran menu głównego.
Przebieg Główny	1. Na ekranie menu głównego obywatel klika przycisk „Oceń pracowników”. 2. Zostaje wyświetlony ekran pokazujący wszystkie zakończone wnioski należące do obywatela. 3. Obywatel klika przycisk „Przejdź do oceny” przy wniosku, dla którego chcę przeprowadzić ocenę pracowników. 4. Zostaje wyświetlony ekran pokazujący listę pracowników, którzy brali udział w rozpatrywaniu wniosku i akcji, jakie każdy z nich przeprowadził w sprawie danego wniosku. 5. Obywatel klika na przycisk „Oceń pracownika” przy pracowniku, dla którego chcę wystawić ocenę. 6. Zostaje wyświetlone okno wprowadzania oceny. 7. Obywatel zaznacza od 1 do 10 gwizdek reprezentujących ocenę (i opcjonalnie wpisuje komentarz do oceny), po czym klika przycisk „Dodaj”. 8. Jeżeli komentarz nie przekroczył maksymalnej dopuszczalnej długości to zostaje wyświetlony komunikat o pomyślnym dodaniu oceny.
Przebieg Alternatywny	8 a) Jeżeli komentarz przekroczył maksymalną dopuszczalną długość to zostaje wyświetlony komunikat o błędzie. Powrót do punktu 5.
Warunki Końcowe	• Dodanie nowej oceny dla pracownika, który brał udział w procesowaniu wniosków obywatela.

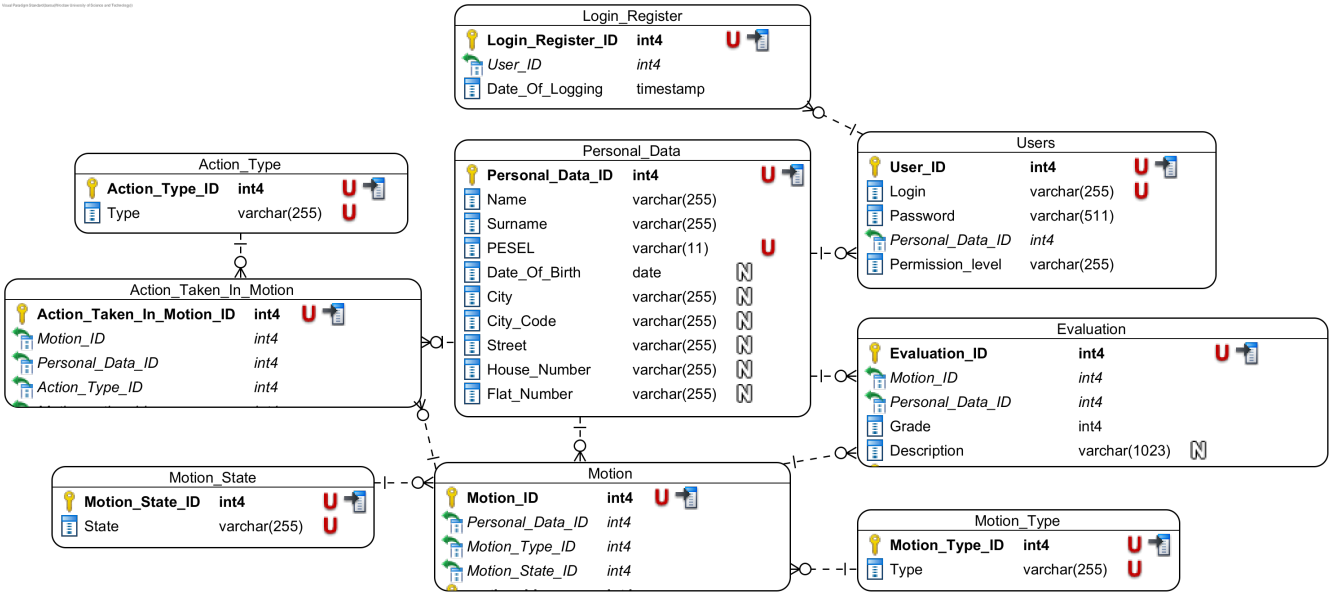
4.5. Baza danych

4.5.1. Schemat Bazy danych

Baza danych będzie się składać z 9 tabel:

- Personal_Datta - Zawierającej dane osobowe.
- Users - Zawierającej dane kont obywateli i pracowników.
- Motion - Zawierającej dane wniosków.
- Motion_Type - Zawierającej typy wniosków.
- Motion_State - Zawierającej stany wniosków.
- Action_Type - Zawierającej typy akcji, jakie pracownicy mogą wykonać.
- Action_Taken_In_Motion - Zawierającej akcje, jakie wykonali pracownicy w związku z procesowaniem danego wniosku.
- Evaluation - Zawierającej Oceny wystawione pracownikom przez obywateli.
- LoginRegister - Zawierającej rejestr logowań kont użytkowników.

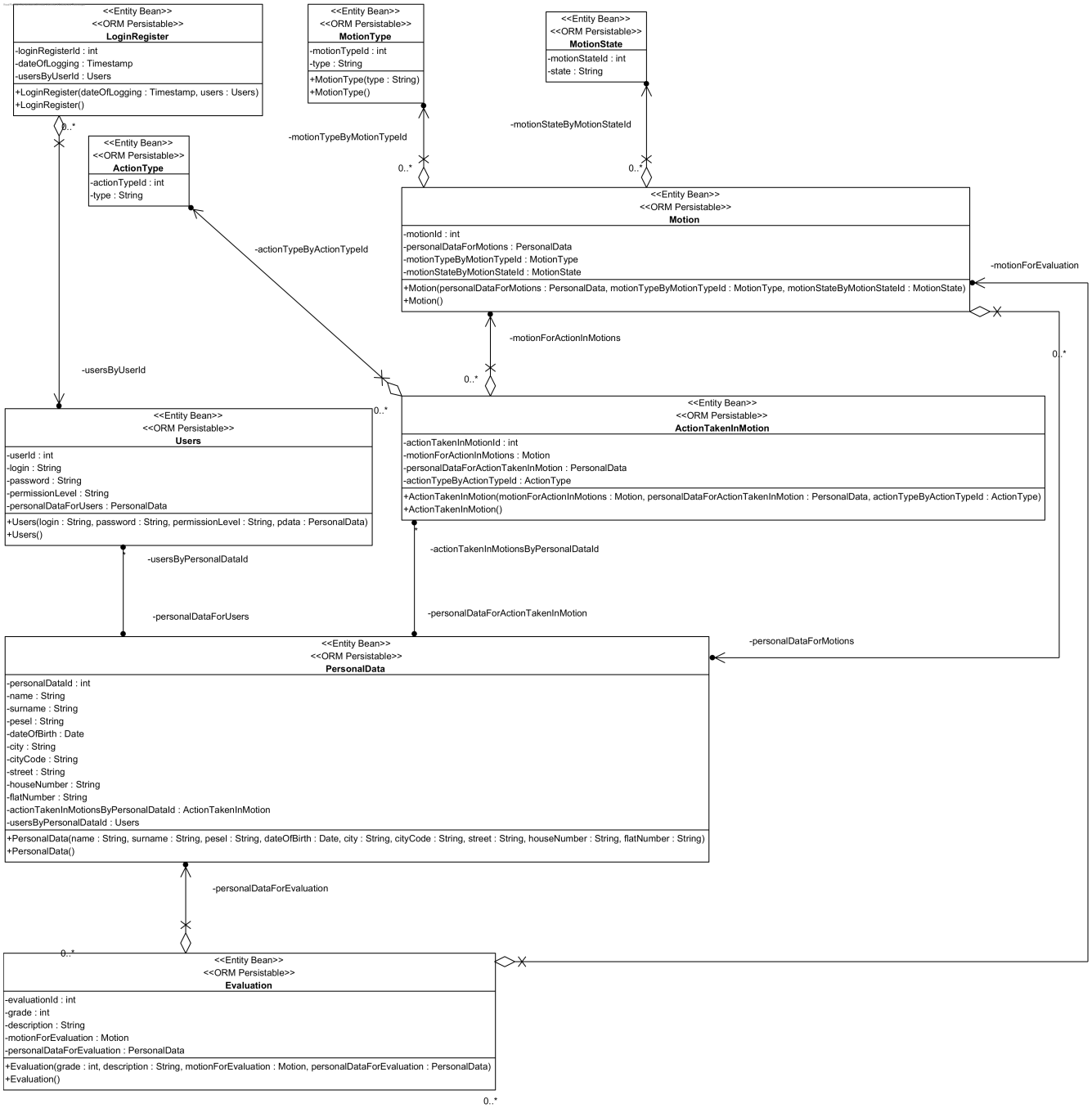
Relacje pomiędzy tabelami będę wyglądać tak jak na diagramie widocznym na rysunku 4.3.



Rys. 4.3: Schemat bazy danych. Źródło: Opracowanie własne.

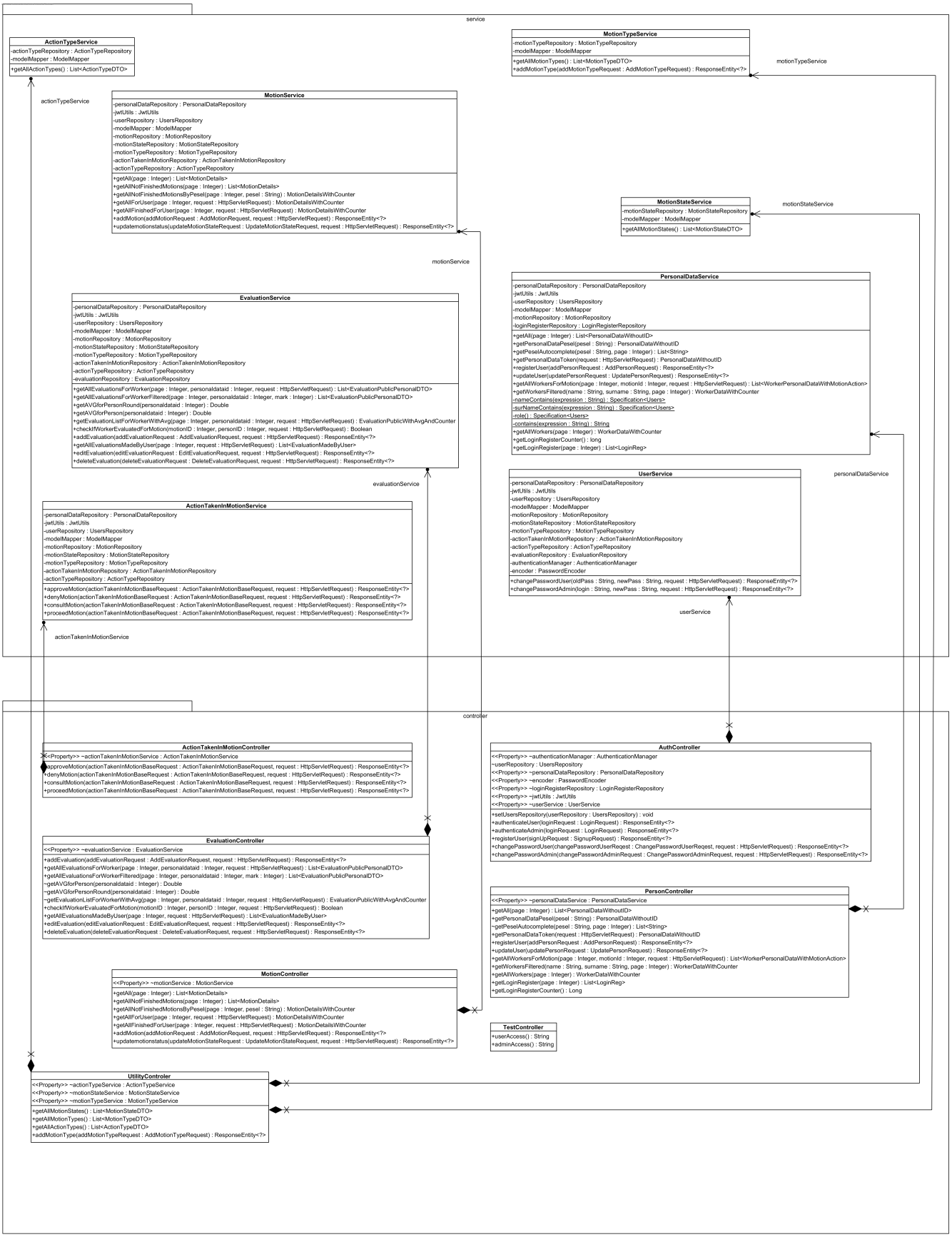
4.6. Diagramy Klas

Diagram klas pakietu model został przedstawiony na rysunku 4.4. Pakiet ten zawiera klasy odpowiadają encjom w bazie danych.



Rys. 4.4: Diagram klas pakietu "model". Źródło: Opracowanie własne.

Diagram klas pakietów: "service"i "controller"został przedstawiony na rysunku 4.5. Pakiet "service"zawiera klasy, które wykonują operacje przy użyciu repozytoriów danych. Pakiet "controller"zawiera klasy, które obsługują endpointy REST Api i zlecają wykonywanie zadań klasom z pakietu "service".



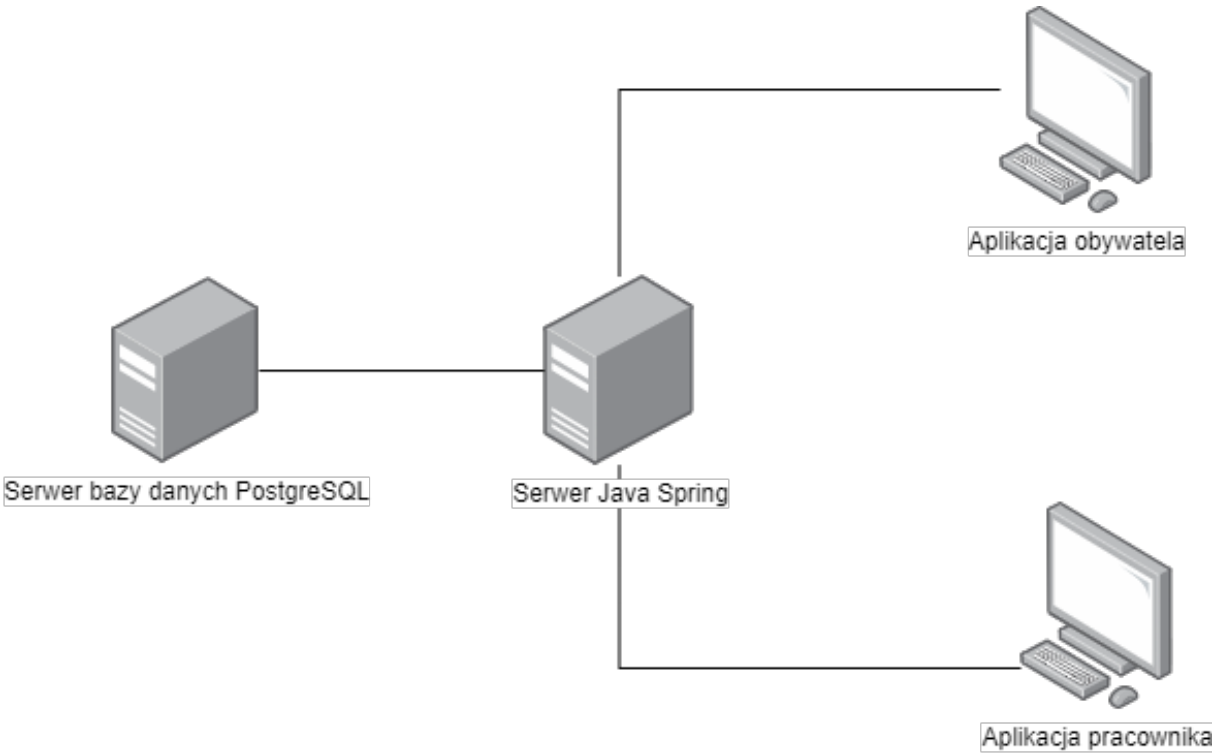
Rys. 4.5: Diagram klas pakietów "service"i "controller". Źródło: Opracowanie własne.

4.7. Schemat architektury systemu

Planowana Architektura systemu będzie składała się z trzech warstw:

- Warstwa danych - Odpowiedzialna za przechowywanie danych potrzebnych do funkcjonowania systemu, składająca się z bazy danych w systemie PostgreSQL.
- Warstwa aplikacji - Odpowiedzialną za logikę biznesową systemu, składającą się z serwera Java z frameworkiem Spring.
- Warstwa prezentacji - Odpowiedzialną za komunikację z użytkownikami, składająca się z dwóch osobnych webowych aplikacji obywatela i pracownika, napisanych w języku JavaScript z biblioteką React.

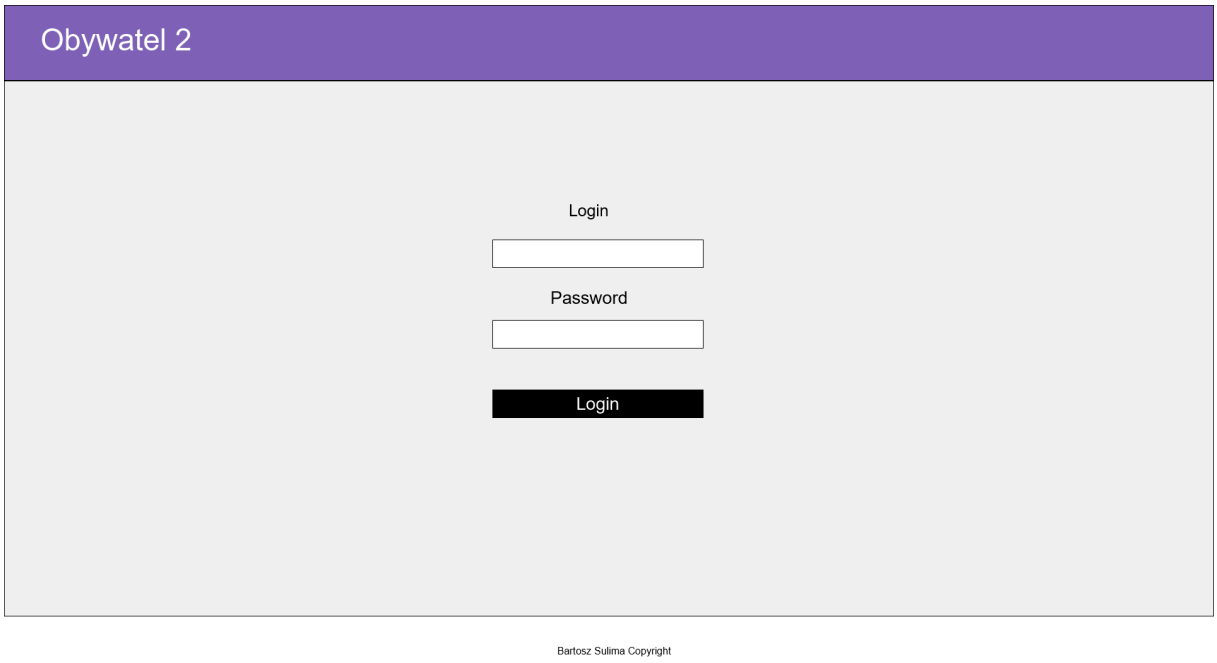
Schemat architektury został przedstawiony na rysunku 4.6.



Rys. 4.6: Schemat architektury systemu. Źródło: Opracowanie własne.

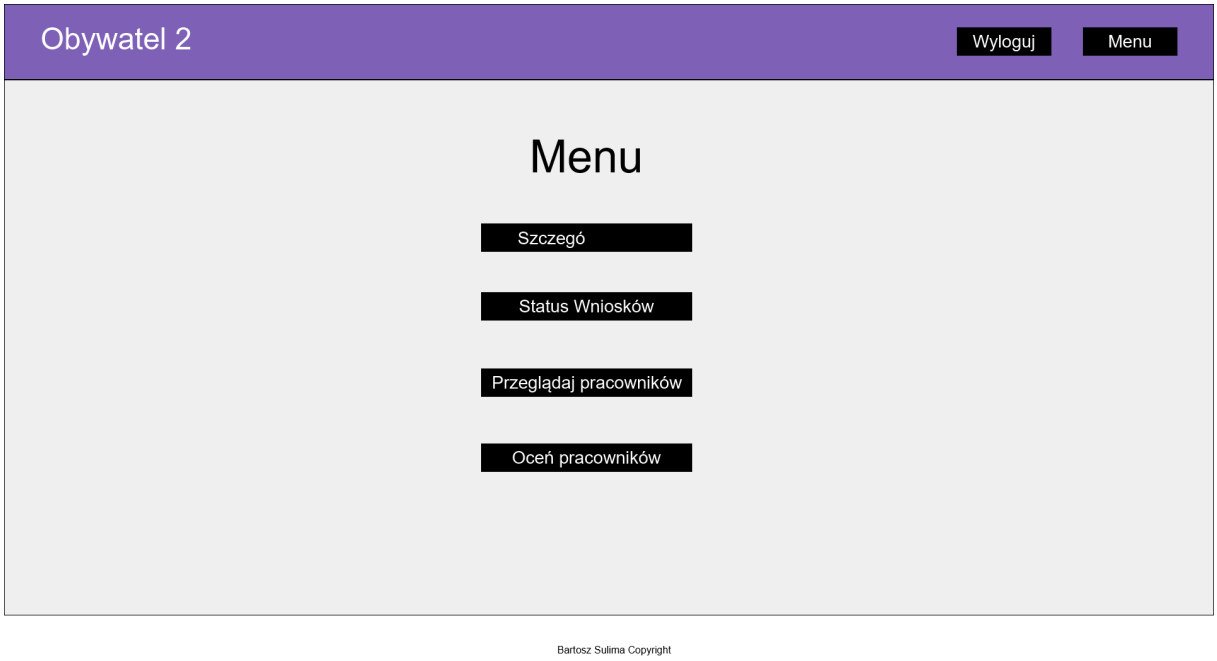
4.8. Projekt interfejsu użytkowników

Projekt ekranu logowania wspólnego dla aplikacji pracowniczej i obywatelskiej został przedstawiony na rysunku 4.7.



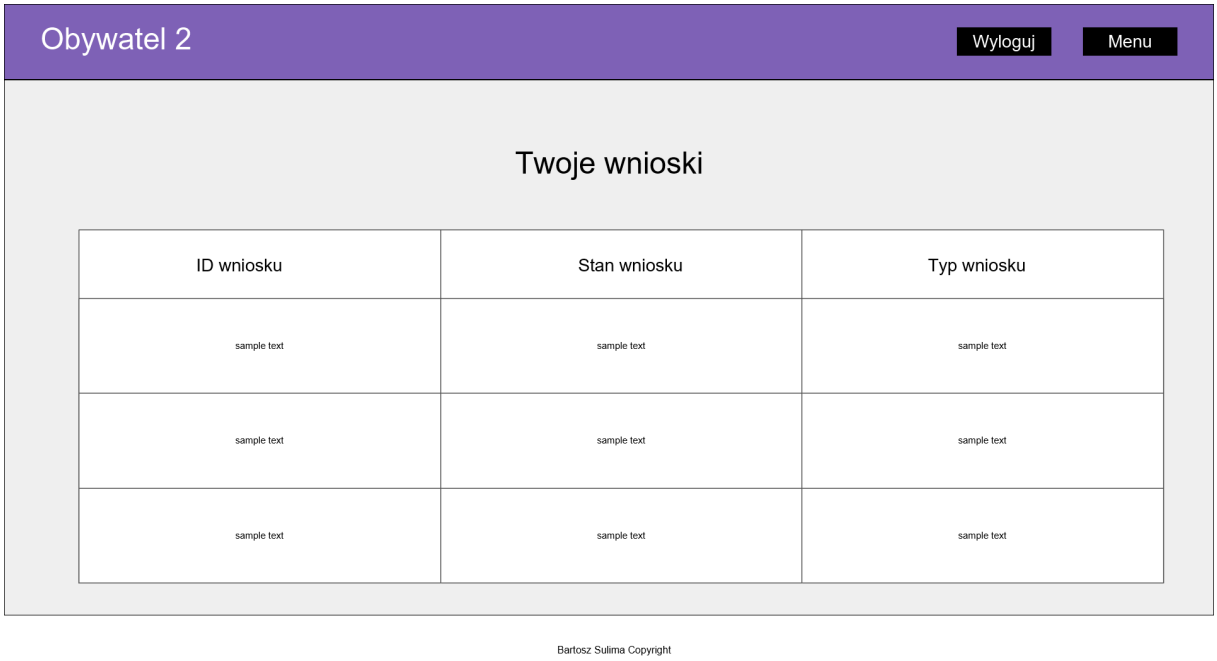
Rys. 4.7: Ekran logowania. Źródło: Opracowanie własne.

Projekt ekranu menu głównego wspólnego dla aplikacji pracowniczej i obywatelskiej został przedstawiony na rysunku 4.8.



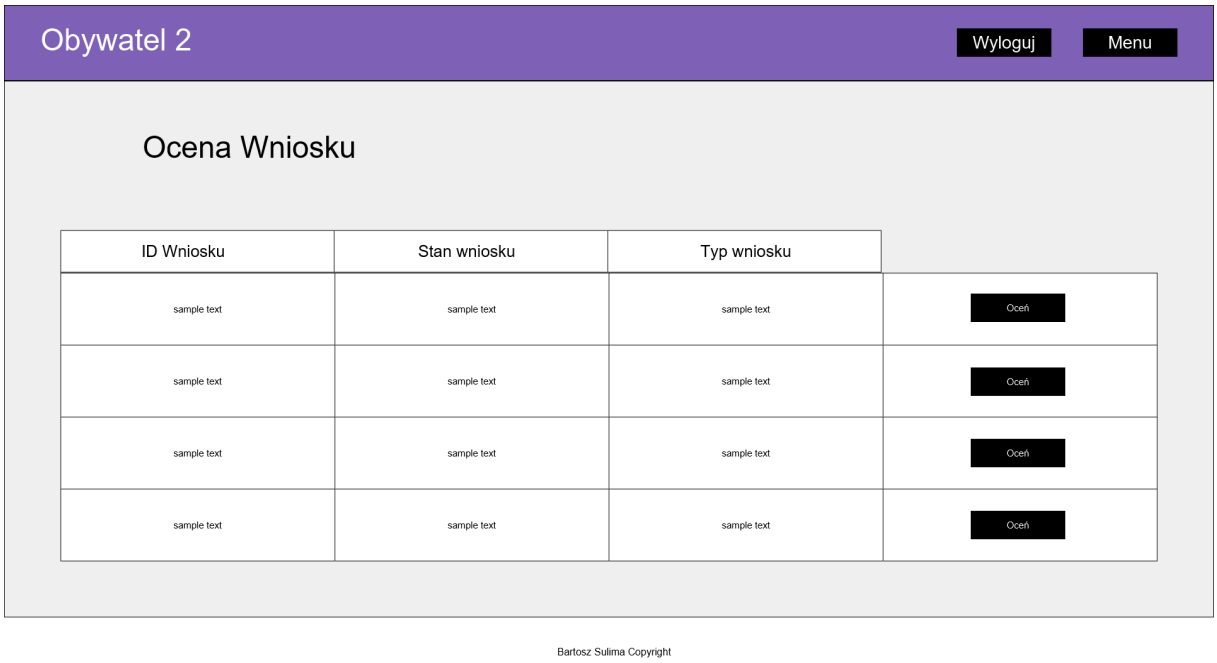
Rys. 4.8: Ekran menu głównego. Źródło: Opracowanie własne.

Projekt ekranu wyświetlającego wszystkie wnioski dla aplikacji obywatelskiej został przedstawiony na rysunku 4.9.



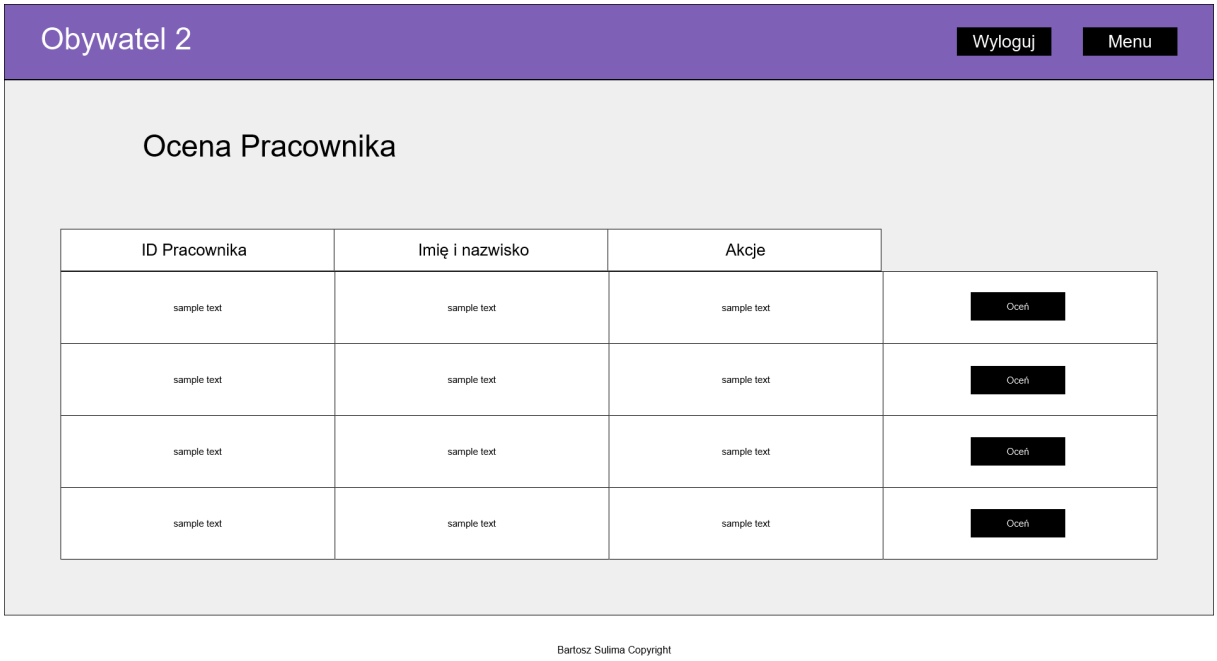
Rys. 4.9: Ekran stanu wszystkich wniosków. Źródło: Opracowanie własne.

Projekt ekranu wyświetlającego wszystkie wnioski gotowe do oceny dla aplikacji obywatelskiej został przedstawiony na rysunku 4.10.



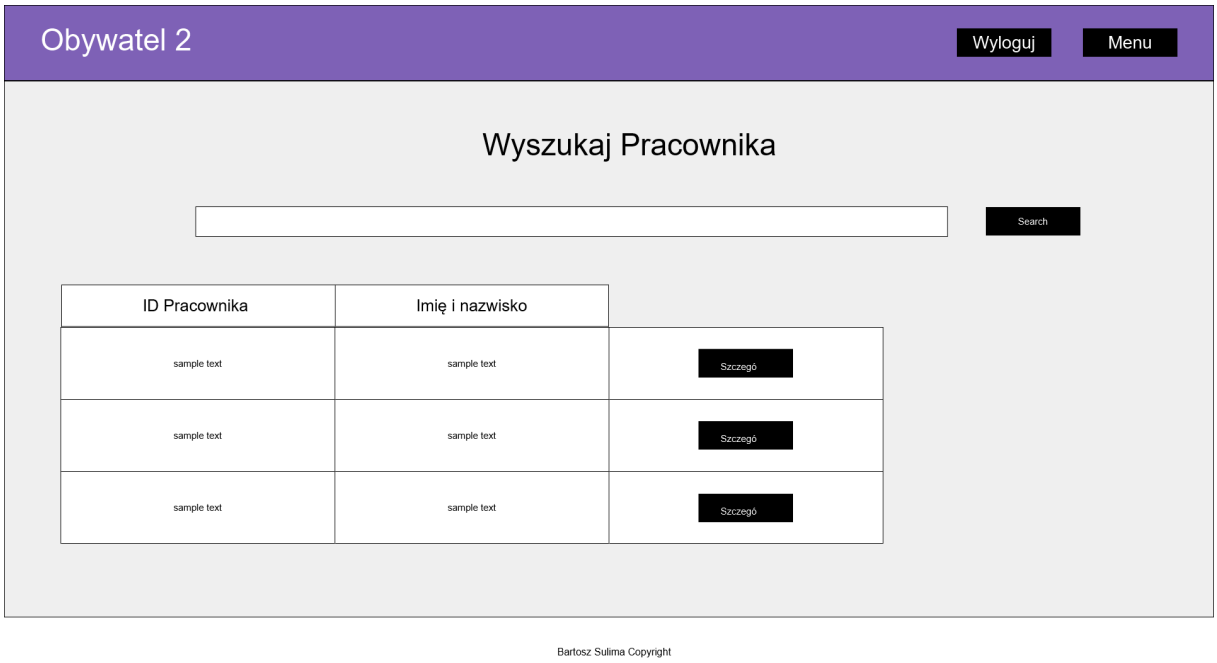
Rys. 4.10: Ekran wyboru wniosku do oceny. Źródło: Opracowanie własne.

Projekt ekranu wyboru pracownika do oceny dla aplikacji obywatelskiej został przedstawiony na rysunku 4.11.



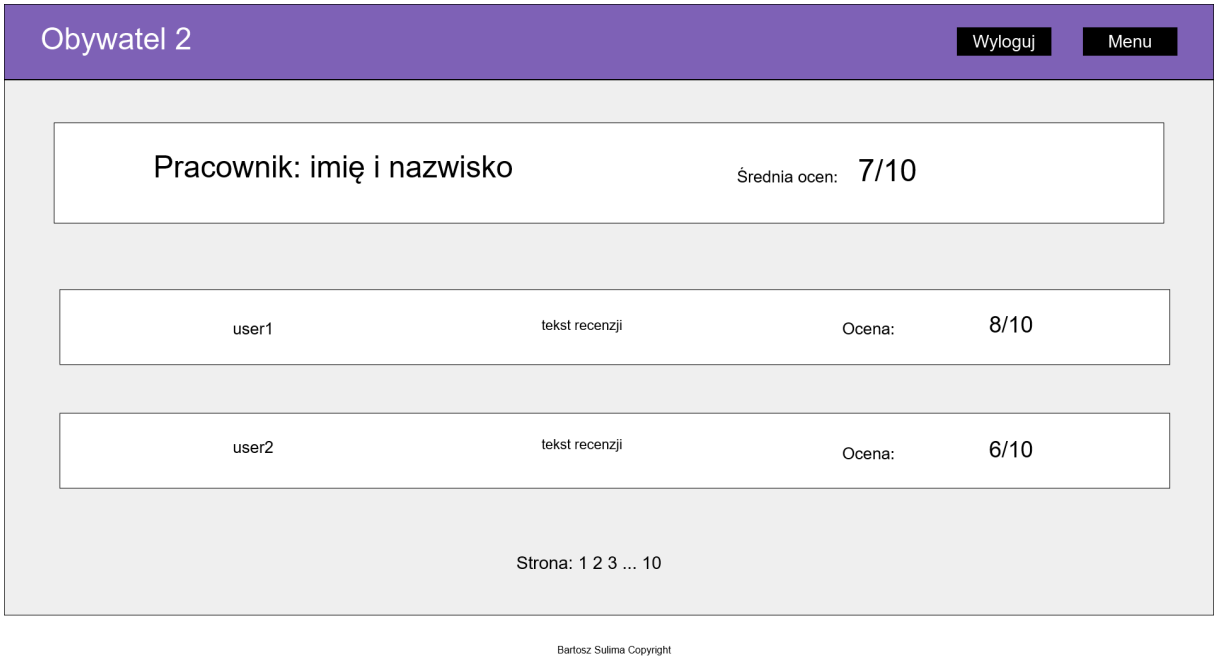
Rys. 4.11: Ekran wyboru pracownika do oceny. Źródło: Opracowanie własne.

Projekt ekranu przeglądania pracowników dla aplikacji obywatelskiej został przedstawiony na rysunku 4.12.



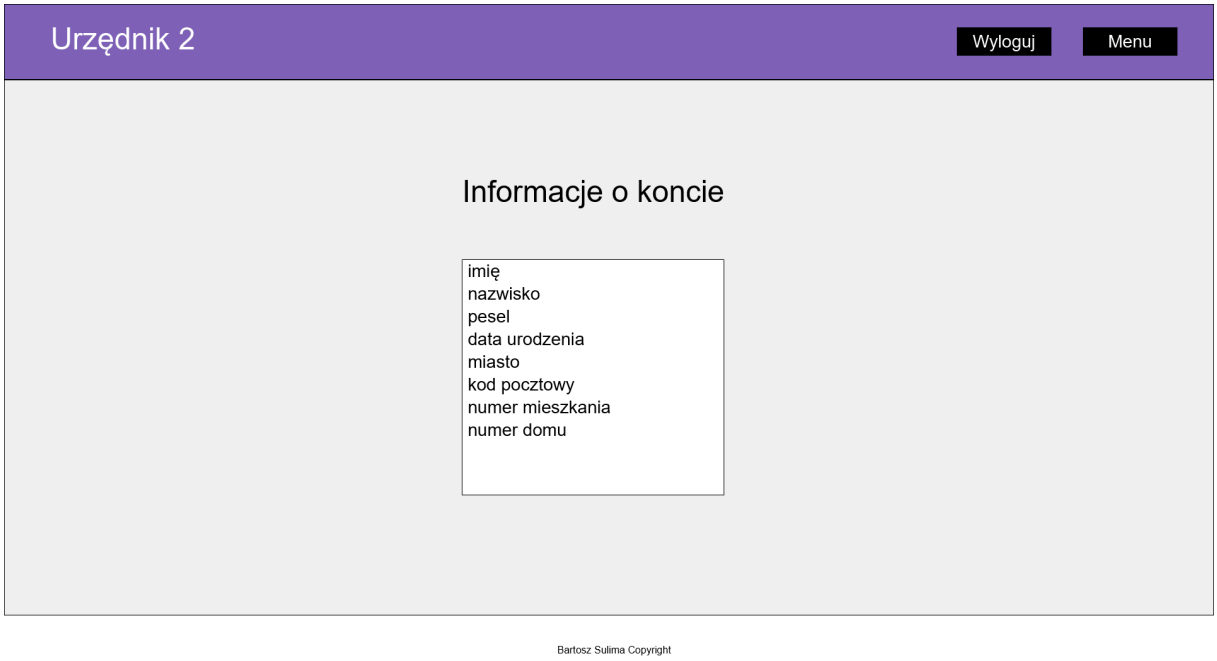
Rys. 4.12: Ekran przeglądania pracowników. Źródło: Opracowanie własne.

Projekt ekranu ocen pracownika dla aplikacji obywatelskiej został przedstawiony na rysunku 4.13.



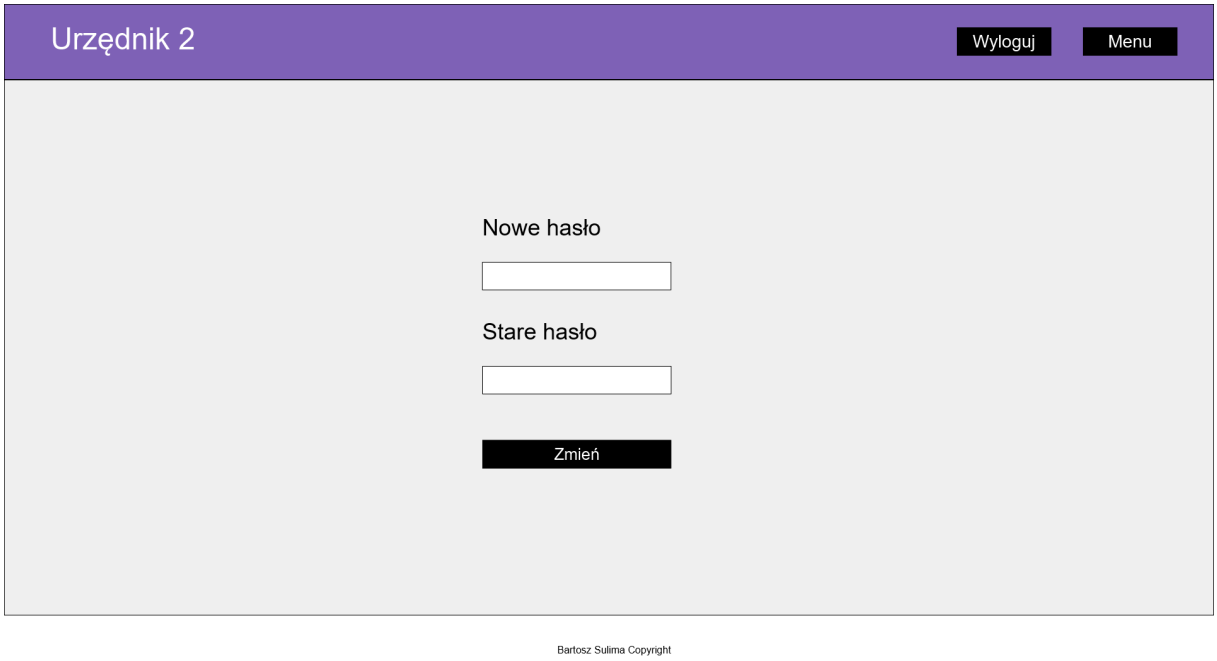
Rys. 4.13: Ekran ocen pracownika. Źródło: Opracowanie własne.

Projekt ekranu szczegółów konta wspólnego dla aplikacji pracowniczej i obywatelskiej został przedstawiony na rysunku 4.14.



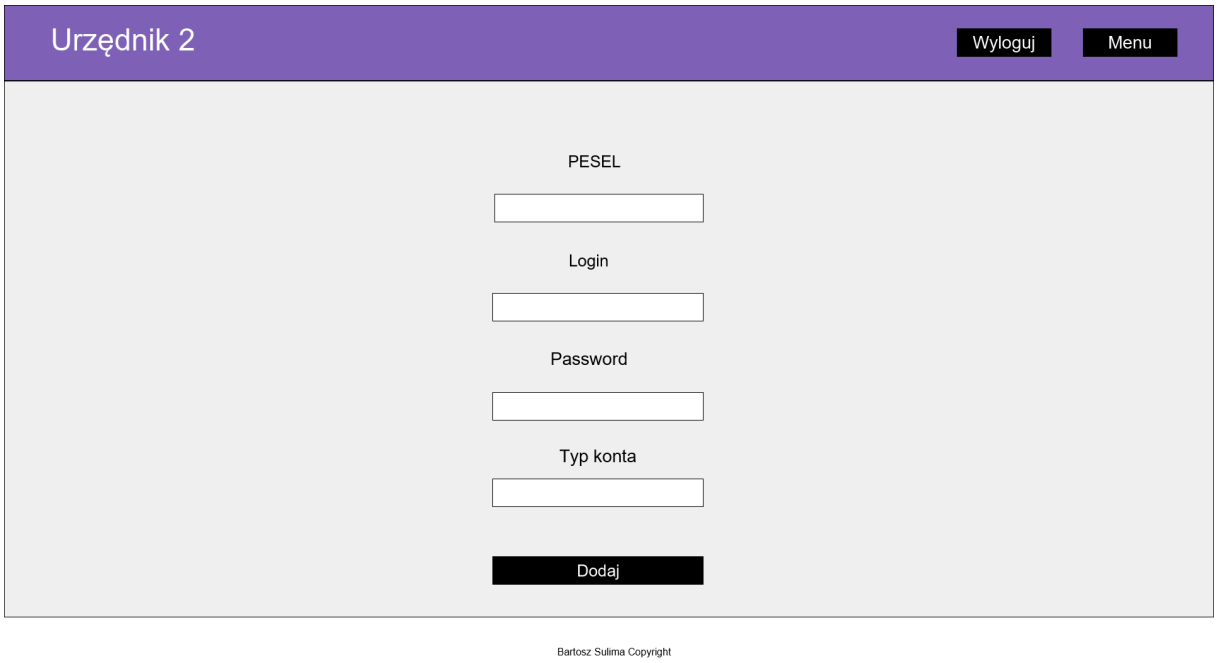
Rys. 4.14: Ekran szczegółów konta. Źródło: Opracowanie własne.

Projekt ekranu zmiany hasła wspólnego dla aplikacji pracowniczej i obywatelskiej został przedstawiony na rysunku 4.15.



Rys. 4.15: Ekran Zmiany hasła. Źródło: Opracowanie własne.

Projekt ekranu dodawania nowych kont użytkowników dla aplikacji pracowniczej został przedstawiony na rysunku 4.16.

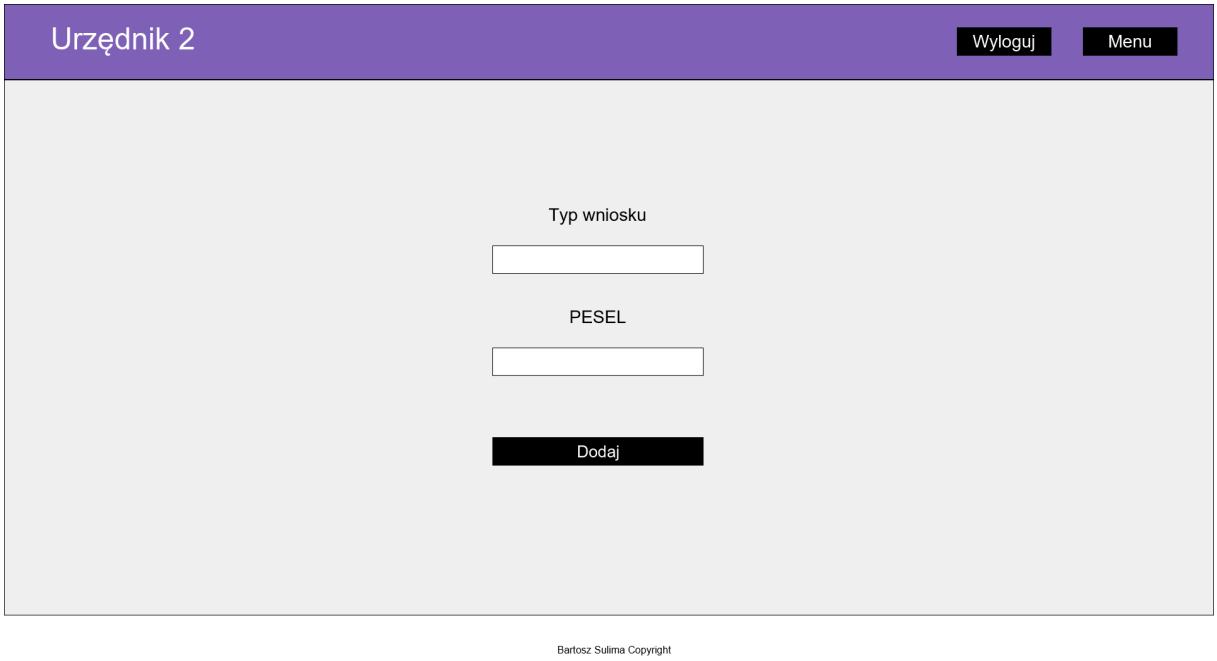


Rys. 4.16: Ekran dodawania nowych kont użytkowników. Źródło: Opracowanie własne.

Projekt ekranu wyszukiwania osób dla aplikacji pracowniczej został przedstawiony na rysunku 4.17.

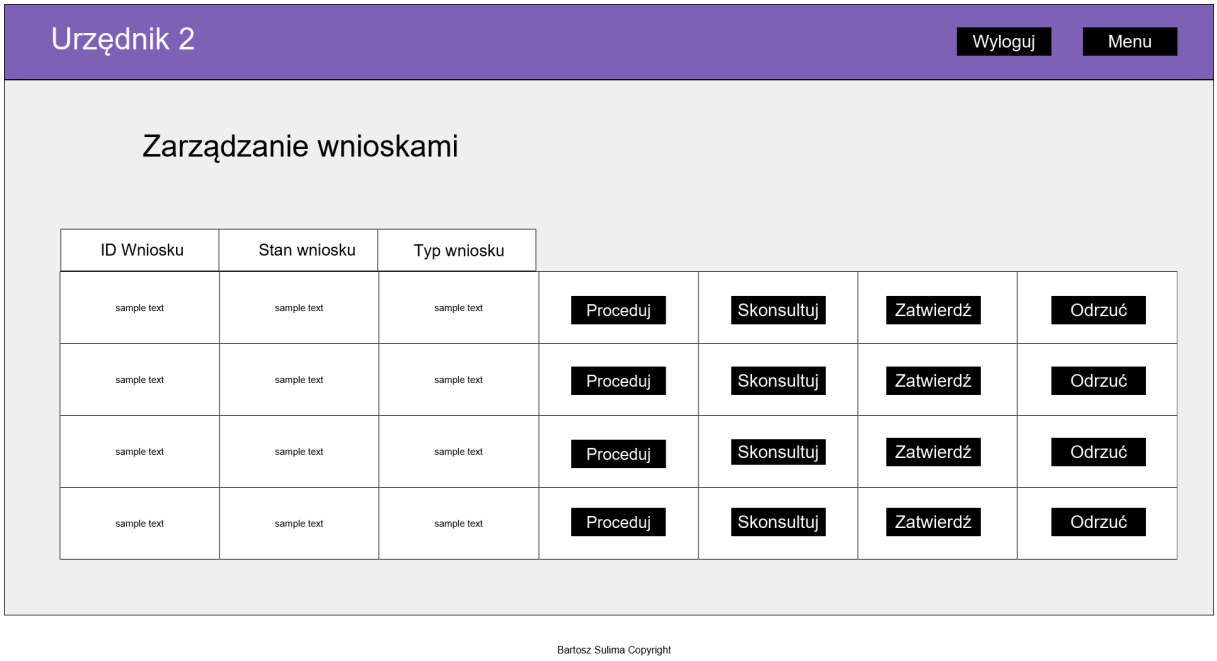
Rys. 4.17: Ekran wyszukiwania osób. Źródło: Opracowanie własne.

Projekt ekranu dodawania nowych wniosków dla aplikacji pracowniczej został przedstawiony na rysunku 4.18.



Rys. 4.18: Ekran dodawania nowych wniosków. Źródło: Opracowanie własne.

Projekt ekranu zarządzania nierozpatrzonymi wnioskami dla aplikacji pracowniczej został przedstawiony na rysunku 4.19.



Rys. 4.19: Ekran zarządzania nierozpatrzonymi wnioskami. Źródło: Opracowanie własne.

Rozdział 5

Implementacja

5.1. Wstęp

W poniższym rozdziale przedstawiony zostanie opis implementacji aplikacji webowych obywatela i urzędnika, wykorzystujących technologie Java, Spring JPA, JavaScript, React i PostgreSQL. Java jest językiem programowania, który zostanie wykorzystany do napisania aplikacji serwerowej pośredniczącej pomiędzy bazą danych a aplikacjami webowymi. Framework Spring umożliwi efektywne zarządzanie bazami danych z wykorzystaniem Java Persistence API (JPA). JavaScript i React będą wykorzystywane do stworzenia webowych aplikacji użytkowników. PostgreSQL będzie przechowywać dane potrzebne do poprawnego działania systemu.

System oceniania pracy urzędników urzędu miasta wykonany w ramach tej pracy jest kierowany na komputery osobiste z systemem operacyjnym Windows 10. Do obsługi systemu wymagana jest instalacja wirtualnej maszyny Java w wersji 19 lub nowszej, a także system zarządzania relacyjnymi bazami danych PostgreSQL w wersji 14 lub nowszej. Do inicjalizacji bazy danych wymagana jest posiadanie programu pgAdmin4, który jest instalowany automatycznie z serwerem bazy danych.

5.2. Wzorce projektowe

5.2.1. Repozytorium

Repozytorium - wzorzec projektowy, umożliwiający oddzielenie logiki biznesowej aplikacji od logiki dostępu do danych. Jest to warstwa pośrednicząca między aplikacją a bazą danych, która odpowiada za przechowywanie i udostępnianie danych. Wzorzec ten pozwala na zaimplementowanie operacji CRUD (Create, Read, Update, Delete) na danych bez konieczności tworzenia bezpośrednich zapytań w SQL. Wzorzec ten jest często wykorzystywany w projektach opartych o architekturę warstwową. Zaletami używania Repozytorium jest mniej skomplikowany kod, co sprawia, że jest go łatwiej utrzymać i modyfikować[20]. Przykład użycia Repozytorium w pracy przedstawiono na listingu 5.1.

Listing 5.1: Interfejs Repository dla klasy "ActionType".

```
@Repository
public interface ActionTypeRepository  extends JpaRepository<ActionType,
    ↪ Integer> {
    Boolean existsByType(String type);
    Optional<ActionType> findByType(String type);
}
```

5.2.2. DTO

DTO (Data Transfer Object) - to wzorzec projektowy służący do transferu danych pomiędzy warstwami aplikacji. Jest to obiekt, który zawiera tylko dane i jest pozbawiony logiki biznesowej. DTO jest często wykorzystywany w aplikacjach udostępniających dane poprzez interfejs REST. Zamiast wysyłać cały obiekt istniejący w logice biznesowej, który może zawierać dane niepotrzebne w warstwie prezentacyjnej lub dane potencjalne wrażliwe, powinno się przeprowadzić konwersję danych z obiektu zawierającego wszystkie dane na DTO posiadające dane niezbędne w danym kontekście[31]. Przykład użycia DTO w pracy przedstawiono na listingu 5.2. Jest to obiekt przechowujący wybrane dane z klasy "PersonalData"przedstawionej na listingu 5.6.

Listing 5.2: Kod klasy "WorkerDataDTO".

```
@Getter
@Setter
public class WorkerDataDTO {
    private int personalDataId;

    private String name;

    private String surname;
}
```

5.3. Implementacja bazy danych

Baza danych została zaimplementowana z użyciem systemu PostgreSQL[7]. Przykładowy kod tworzenia tabel przedstawia listing 5.3. Inicjalizacja bazy danych odbywa się poprzez wywołanie skryptu w "querry tool"programu pgAdmin4. Skrypt ten poza tworzeniem tabel w bazie danych wprowadza do nich podstawowe dane (przykład 5.4) wymagane do korzystania z aplikacji. Struktura bazy danych składa się z 9 tabel, których skład, nazewnictwo i relacje zostały zaimplementowane zgodnie z przygotowanym modelem przedstawionym na diagramie widocznym na rysunku 4.3.

Listing 5.3: Kod tworzenia tabel "Personal_Data"oraz "Users"w bazie danych.

```
CREATE TABLE Personal_Data (
Personal_Data_ID SERIAL NOT NULL,
Name             varchar(255) NOT NULL,
Surname          varchar(255) NOT NULL,
PESEL            varchar(11) NOT NULL UNIQUE,
Date_Of_Birth    date,
City             varchar(255),
City_Code        varchar(255),
Street           varchar(255),
House_Number     varchar(255),
Flat_Number      varchar(255),
PRIMARY KEY (Personal_Data_ID));

CREATE TABLE Users (
User_ID          SERIAL NOT NULL,
Login            varchar(255) NOT NULL UNIQUE,
Password         varchar(511) NOT NULL,
Personal_Data_ID int4 NOT NULL,
Permission_level varchar(255) NOT NULL,
PRIMARY KEY (User_ID));
```

```
ALTER TABLE Users ADD CONSTRAINT FKUsers482500 FOREIGN KEY (
    ↪ Personal_Data_ID) REFERENCES Personal_Data (Personal_Data_ID)
on update restrict
on delete cascade;
```

Listing 5.4: Kod dodawania konta administratora do bazy danych.

```
INSERT INTO personal_data (name,surname,pesel,date_of_birth,city,
    ↪ city_code,street,house_number,flat_number)
VALUES
('Admin Name','Admin Surname','00000000000','2000-01-01','Admin City','
    ↪ Admin City Code','Admin Street','Admin House','Admin Flat');

INSERT INTO users (login, password, personal_data_id, permission_level)
VALUES
('admin','$2a$10$idXjSmTlgUYQK3gvPGAAv.fDswNHOux2SUYdTISdTmONKekcD5v1u'
    ↪ ,1,'ROLE_ADMIN');
```

5.4. Implementacja serwera

5.4.1. Wstęp

Serwer został stworzony w języku Java z wykorzystaniem frameworka Spring i narzędzia do automatycznego budowania Maven. Komunikacja z bazą danych jest przeprowadzona z wykorzystaniem Spring JPA, kod konfiguracyjny połączenia przedstawiono na listingu 5.5.

Listing 5.5: Właściwości aplikacji określające połączenie z bazą danych.

```
spring.datasource.url=jdbc:postgresql://localhost:5432/Town_Database
spring.datasource.username=postgres
spring.datasource.password=DtSBbb5HJv
```

5.4.2. Tworzenie klas odpowiadających encjom w bazie danych

W aplikacji serwerowej stworzono klasy, które odpowiadają encjom w bazie danych, czego przykładem jest klasa "PersonalData"przedstawiona na listingu 5.6. Klasa "PersonalData"jest odpowiednikiem encji "Personal_Data"przedstawionej na listingu 5.3. W klasach użyto biblioteki "Lombok", która automatycznie dostarcza podstawowe metody takie jak "get"i "set".

Listing 5.6: Kod klasy PersonalData.

```
@Getter
@Setter
@Entity
@Table(name = "personal_data", schema = "public", catalog = "Town_Database"
    ↪ )
public class PersonalData {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "personal_data_id", nullable = false)
    private int personalDataId;
    @Basic
    @Column(name = "name", nullable = false, length = 255)
    private String name;
    @Basic
```

```
    @Column(name = "surname", nullable = false, length = 255)
    private String surname;
    @Basic
    @Column(name = "pesel", nullable = false, length = 11, unique = true)
    private String pesel;
    @Basic
    @Column(name = "date_of_birth", nullable = true)
    private Date dateOfBirth;
    @Basic
    @Column(name = "city", nullable = true, length = 255)
    private String city;
    @Basic
    @Column(name = "city_code", nullable = true, length = 255)
    private String cityCode;
    @Basic
    @Column(name = "street", nullable = true, length = 255)
    private String street;
    @Basic
    @Column(name = "house_number", nullable = true, length = 255)
    private String houseNumber;
    @Basic
    @Column(name = "flat_number", nullable = true, length = 255)
    private String flatNumber;
    @OneToMany(mappedBy = "personalDataForActionTakenInMotion", cascade =
        ↪ CascadeType.ALL)
    private Collection<ActionTakenInMotion>
        ↪ actionTakenInMotionsByPersonalDataId;
    @OneToMany(mappedBy = "personalDataForUsers", cascade = CascadeType.ALL
        ↪ )
    private Collection<Users> usersByPersonalDataId;
    public PersonalData(String name, String surname, String pesel, Date
        ↪ dateOfBirth, String city, String cityCode, String street, String
        ↪ houseNumber, String flatNumber) {
        this.name = name;
        this.surname = surname;
        this.pesel = pesel;
        this.dateOfBirth = dateOfBirth;
        this.city = city;
        this.cityCode = cityCode;
        this.street = street;
        this.houseNumber = houseNumber;
        this.flatNumber = flatNumber;
    }
    public PersonalData() {}
}
```

5.4.3. Tworzenie klas serwisów oraz kontrolerów

W aplikacji serwerowej stworzono klasy serwisów, których zadaniem jest wykonywanie operacji przy użyciu repozytoriów danych. Przykład klasy serwisu przedstawiono na listingu 5.7. Kolejnym rodzajem klas stworzonych w aplikacji serwerowej są klasy kontrolerów, które odpowiedzialne są za wystawianie endpointów REST Api, sprawdzanie uprawnień zawartych w tokenie użytkownika tworzonym w trakcie logowania, sprawdzanie czy dane otrzymane w zapytaniu skierowanym na endpoint są właściwego typu, przekazywanie tych danych do odpowiedniej funkcji serwisu i na końcu odsyłanie wyników działania serwisu przez endpoint. Przykład klasy kontrolera przedstawiono na listingu 5.8.

Listing 5.7: Kod serwisu "MotionTypeService".

```
@Service
@RequiredArgsConstructor
public class MotionTypeService {
    private final MotionTypeRepository motionTypeRepository;
    private final ModelMapper modelMapper;
    public List<MotionTypeDTO> getAllMotionTypes() {
        return motionTypeRepository.findAll().stream()
                .map(mapItem -> modelMapper.map(mapItem, MotionTypeDTO.
                    ↪ class))
                .collect(Collectors.toList());
    }


    public ResponseEntity<?> addMotionType( AddMotionTypeRequest
        ↪ addMotionTypeRequest) {

        if (motionTypeRepository.existsByTypeAllIgnoreCase(
            ↪ addMotionTypeRequest.getType())){
            return ResponseEntity
                    .badRequest()
                    .body(new MessageResponse("Błąd: Typ wniosku już w
                        ↪ bazie"));
        }
        MotionType motionType =new MotionType(addMotionTypeRequest.getType
            ↪ ());
        motionTypeRepository.save(motionType);
        return ResponseEntity.ok(new MessageResponse("Typ wniosku dodany"))
            ↪ ;
    }


}
```

Listing 5.8: Fragment kodu kontrolera "UtilityControler".

```
@CrossOrigin(origins = "*", maxAge = 3600)
@RestController
@RequestMapping("/utility")
public class UtilityControler {
    ActionTypeService actionTypeService;
    MotionStateService motionStateService;
    MotionTypeService motionTypeService;

    @Autowired
    public void setActionTypeService(ActionTypeService actionTypeService) {
        this.actionTypeService = actionTypeService;
    }

    @Autowired
    public void setMotionStateService(MotionStateService motionStateService
        ↪ ) {
        this.motionStateService = motionStateService;
    }

    @Autowired
    public void setMotionTypeService(MotionTypeService motionTypeService) {
        this.motionTypeService = motionTypeService;
```

```
    }

    {...}


    @GetMapping("/getAllMotionTypes")

    public List<MotionTypeDTO> getAllMotionTypes() {
        return motionTypeService.getAllMotionTypes();
    }


    @PostMapping("/addMotionType")
    @PreAuthorize("hasRole('ADMIN')")
    public ResponseEntity<?> addMotionType(@Valid @RequestBody
        ↪ AddMotionTypeRequest addMotionTypeRequest) {

        return motionTypeService.addMotionType(addMotionTypeRequest);
    }
}
```

5.4.4. Autoryzacja przy użyciu JWT

Do zabezpieczenia aplikacji serwerowej wykorzystane zostały mechanizmy dostępne w ramach Spring Security oraz JSON Web Token.

JWT (JSON Web Token) - JWT jest to standardowy format uwierzytelniania zwykle używany do przekazywania informacji o użytkowniku pomiędzy systemami lub aplikacjami. Przekazywane informacje są zaszyfrowane w formie tokenu i przesyłane w połączeniach HTTP. W skład tokenu JWT wchdzą trzy części: nagłówek(header), dane(payload) i podpis(signature). Nagłówek opisuje typ używanego tokenu. Dane zawierają informacje o użytkowniku, takie jak jego identyfikator lub uprawnienia. Podpis jest używany do weryfikacji autentyczności tokenu i zapobiega jego modyfikacji[23].

Spring security - Warstwa bezpieczeństwa dal aplikacji w języku Java wykorzystujących framework Spring. Spring security pozwala na zabezpieczenie aplikacji przed dostępem nieautoryzowanych osób i umożliwia implementacje wielu mechanizmów uwierzytelniania i autoryzacji. jednym z takich mechanizmów jest uwierzytelnianie za pomocą tokenów (np. JWT)[29].

Bcrypt - algorytm przeznaczony do hashowania haseł. Jest to jedno z najbardziej bezpiecznych rozwiązań dostępnych obecnie na rynku. Bcrypt automatycznie dodaje sól (losowy ciąg znaków) do hasła przed jego z zahashowaniem, co powoduje, że każde hasło jest unikalne, nawet w przypadku, gdy wielu użytkowników używa tego samego hasła[14].

Kod Klasy odpowiedzialnej za obsługę tokenów JWT został zawarty na listingu 5.9. Zawiera ona metodę odpowiedzialną za generowanie tokenu. W skład tokenu wchodzi rola użytkownika, jego login oraz sekretna wartość liczbowa utrudniająca rozszyfrowanie. Ponadto zawarte zostały również metody pozwalające odpowiednio sprawdzić poprawność tokenu oraz wyciągnąć z niego informacje o użytkowniku.

Listing 5.9: Kod klasy JwtUtils.

```
@Component
public class JwtUtils {
    private static final Logger logger = LoggerFactory.getLogger(JwtUtils.
        ↪ class);

    @Value("${town.app.jwtSecret}")
```

```
    private String jwtSecret;

    @Value("${town.app.jwtExpirationMs}")
    private int jwtExpirationMs;

    public String generateJwtToken(Authentication authentication) {

        UserDetailsImpl userPrincipal = (UserDetailsImpl) authentication.
            ↪ getPrincipal();

        return Jwts.builder()
                .setSubject((userPrincipal.getUsername()))
                .setIssuedAt(new Date())
                .setExpiration(new Date((new Date()).getTime() +
                    ↪ jwtExpirationMs))
                .signWith(SignatureAlgorithm.HS512, jwtSecret)
                .compact();
    }

    public String getUserNameFromJwtToken(String token) {
        return Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(token)
            ↪ .getBody().getSubject();
    }

    public boolean validateJwtToken(String authToken) {
        try {
            Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(authToken
                ↪ );
            return true;
        } catch (SignatureException e) {
            logger.error("Invalid JWT signature: {}", e.getMessage());
        } catch (MalformedJwtException e) {
            logger.error("Invalid JWT token: {}", e.getMessage());
        } catch (ExpiredJwtException e) {
            logger.error("JWT token is expired: {}", e.getMessage());
        } catch (UnsupportedJwtException e) {
            logger.error("JWT token is unsupported: {}", e.getMessage());
        } catch (IllegalArgumentException e) {
            logger.error("JWT claims string is empty: {}", e.getMessage());
        }

        return false;
    }

    public static String parseJwt(HttpServletRequest request) {
        String headerAuth = request.getHeader("Authorization");

        if (StringUtils.hasText(headerAuth) && headerAuth.startsWith("
            ↪ Bearer ")) {
            return headerAuth.substring(7);
        }

        return null;
    }


}
```

5.5. Implementacja aplikacji webowych obywatela i urzędnika

5.5.1. Łączność aplikacji webowych z aplikacją serwerową

Komunikacja z serwerem zrealizowana jest na dwa sposoby - z użyciem nowszego podejścia wykorzystującego funkcję fetch (przykład 5.10) oraz starszego z użyciem funkcji useCallback (przykład 5.11). Podejścia te różnią się głównie tym, iż fetch oferuje mechanizm tzw. obietnic, które realizują zadeklarowany fagment kodu po wykonaniu się całej funkcji. Jest to ważne, ponieważ komunikacja z serwerem odbywa się całkowicie asynchronicznie, przez co konieczna jest reakcja na odpowiedzi serwera w momencie ich otrzymania. Podejście nowsze wykorzystano przy zapytaniach wysyłających lub modyfikujących dane w bazie. Do selekcji danych użyto natomiast starszego podejścia, w połączeniu z hookiem useEffect, który reaguje na zmianę zadeklarowanych w nim obiektów. Pozwala to na pobieranie danych dynamicznie za każdym razem, gdy zawartość obserwowanych obiektów ulega zmianom.

Listing 5.10: Kod metody logowania..

```
const login = async (data) => {
        const url = `${appConfig.server}/auth/loginAdmin`;
        try {
            const response = await fetch(`${url}`, {
                method: "POST",
                headers: {
                    "Content-Type": "application/json",
                },
                body: JSON.stringify({
                    login: data.login,
                    password: data.password,
                }),
            });
            const resp = await response.json();
            if (response.ok) {
                setIsLoggedIn(true)
                setAuthToken(resp.token);
                localStorage.setItem('token', resp.token);
                navigate("/menuWorker");
            } else {
                alert(resp.message);
            }
        } catch (e) {
            alert("Connection lost");
        }
    };
```

Listing 5.11: Kod sprawdzający stan zalogowania.

```
    const checkToken = useCallback(() => {
        const xHttp = new XMLHttpRequest();
        xHttp.onreadystatechange = function () {
            if (this.readyState === 4 && this.status !== 200) {
                logout()
            }
        };

        xHttp.open(
            "GET",
```

```
            `${appConfig.server}/test/admin`,
            true,
            null,
            null
        );
        xHttp.setRequestHeader('Authorization', 'Bearer ' + authToken)
        xHttp.send();

    }, [authToken]);
```

5.5.2. Material UI

Do stworzenia komponentów dla aplikacji webowych użyta została popularna bibloteka Material UI. Zawiera ona gotowe elementy interfejsu użytkownika. Material UI jest dostępne dla aplikacji wykorzystujących bibliotekę React.js i umożliwia łatwe i szybkie tworzenie aplikacji z estetycznym i czytelnym interfejsem użytkownika. Material UI posiada gotowe komponenty, takie jak między innymi: przyciski, alerty, tabele i pola tekstowe, które można łatwo dodać do aplikacji i dostosować do własnych potrzeb. Jednymi z głównych zalet Material UI jest wydajność a także bogata dokumentacja ułatwiająca jej naukę i wykorzystywanie[3]. Przykład użycia komponentów z biblioteki Material UI jest przedstawiony na listingu 5.12.

Listing 5.12: Kod Kod komponętu strony do zmiany hasła użytkownika.

```
<Box>
    <NavBar/>
    <Container className="blackout" component="main" maxWidth="xs">

        <Box
            sx={{
                marginTop: 2,
                display: 'flex',
                flexDirection: 'column',
                alignItems: 'center',
            }}
        >

            <Typography component="h1" variant="h4">
                Zmień Hasło:
            </Typography>
            <Box component="form" onSubmit={handleSubmit} noValidate sx={{
                ↪ mt: 1 }}>
                <TextField
                    margin="normal"
                    required
                    fullWidth
                    type = "password"
                    id="oldpassword"
                    label="Stare Hasło"
                    name="oldpassword"
                    autoComplete="oldpassword"

                />
                <TextField
                    margin="normal"
                    required
                    type = "password"
                    fullWidth
                    name="newpassword"
```

```
                    label="Nowe Hasło"
                    id="newpassword"

                />

                <Button
                    type="submit"
                    fullWidth
                    variant="contained"
                    sx={{ mt: 3, mb: 2 }}

                >
                    Zmień hasło
                </Button>

            </Box>
        </Box>
    </Container>
</Box>
```

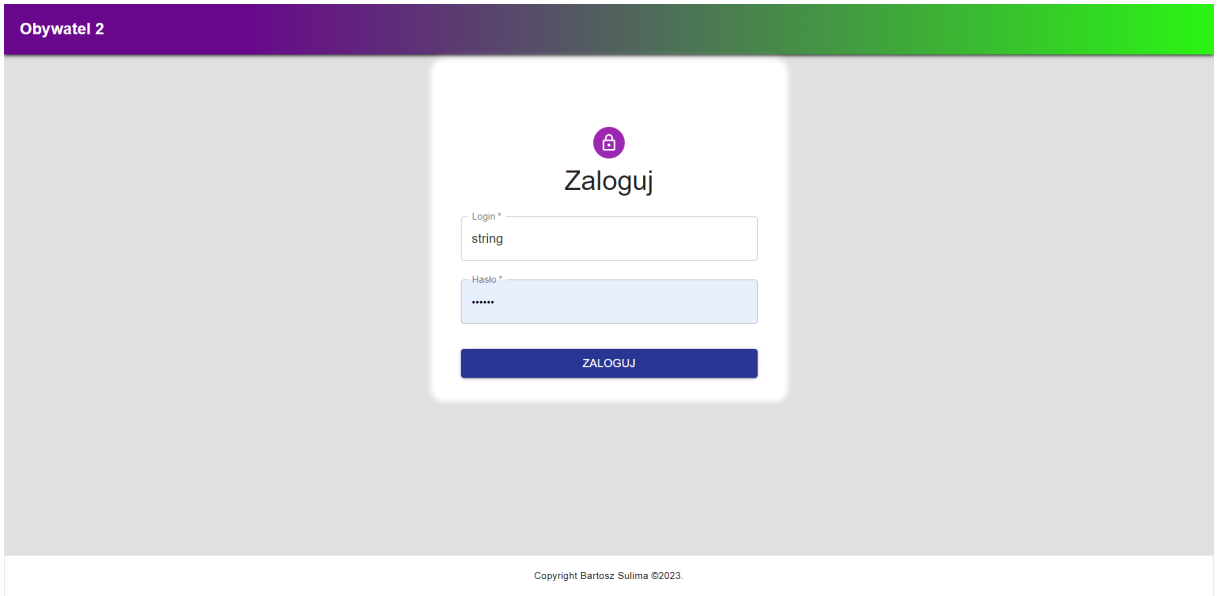
Rozdział 6

Interfejs Obywatela

6.1. Wstęp

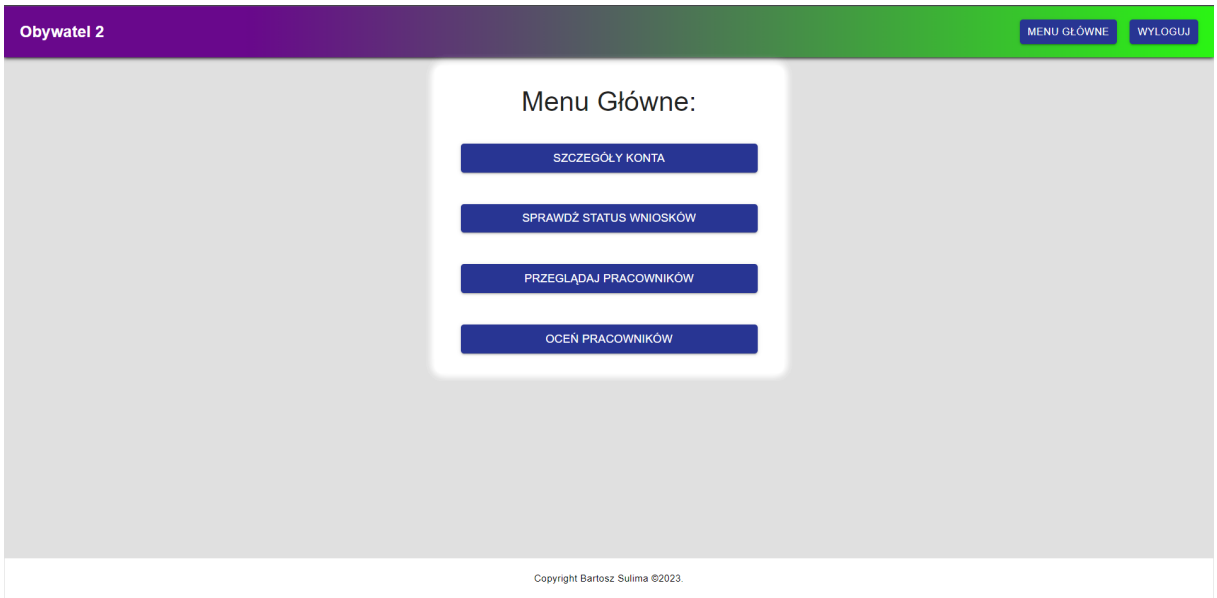
Poniższy rozdział przedstawia końcowy wygląd aplikacji obywatela. W rozdziale tym pominięto część ekranów które były identyczne pod względem wyglądu i funkcji co ekrany w aplikacji pracownika. Odpowiedniki pominiętych ekranów znajdują się w rozdziale opisującym końcowy wygląd aplikacji pracowniczej 7.

Końcowy wygląd ekranu logowania został przedstawiony na rysunku 6.1.



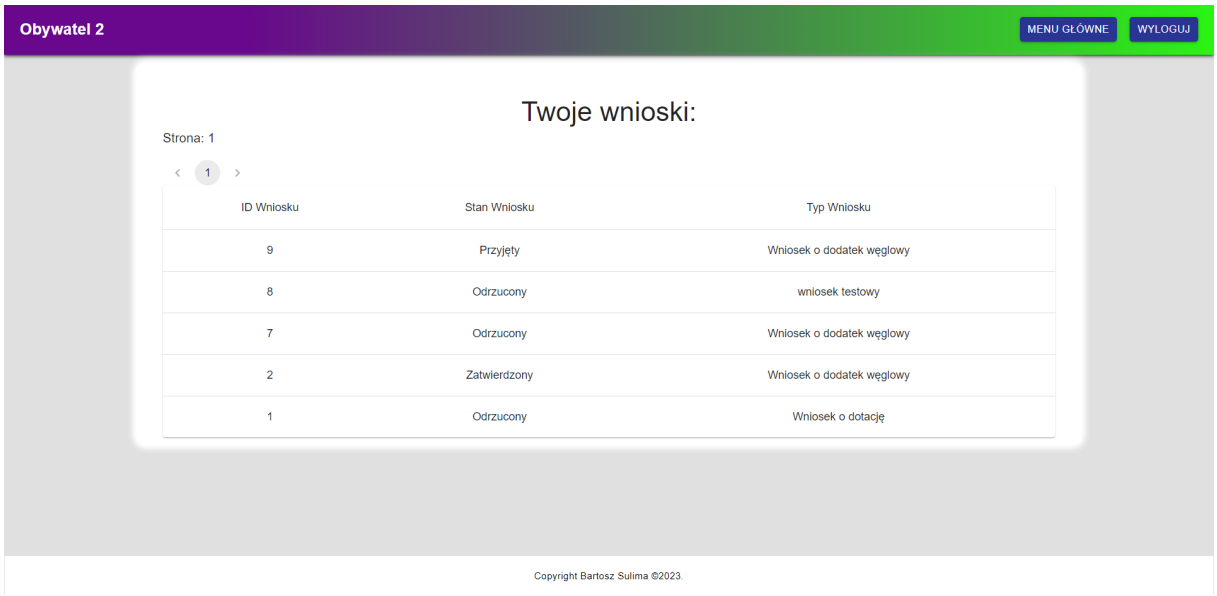
Rys. 6.1: Ekran logowania. Źródło: Opracowanie własne.

Końcowy wygląd ekranu menu głównego został przedstawiony na rysunku 6.2.



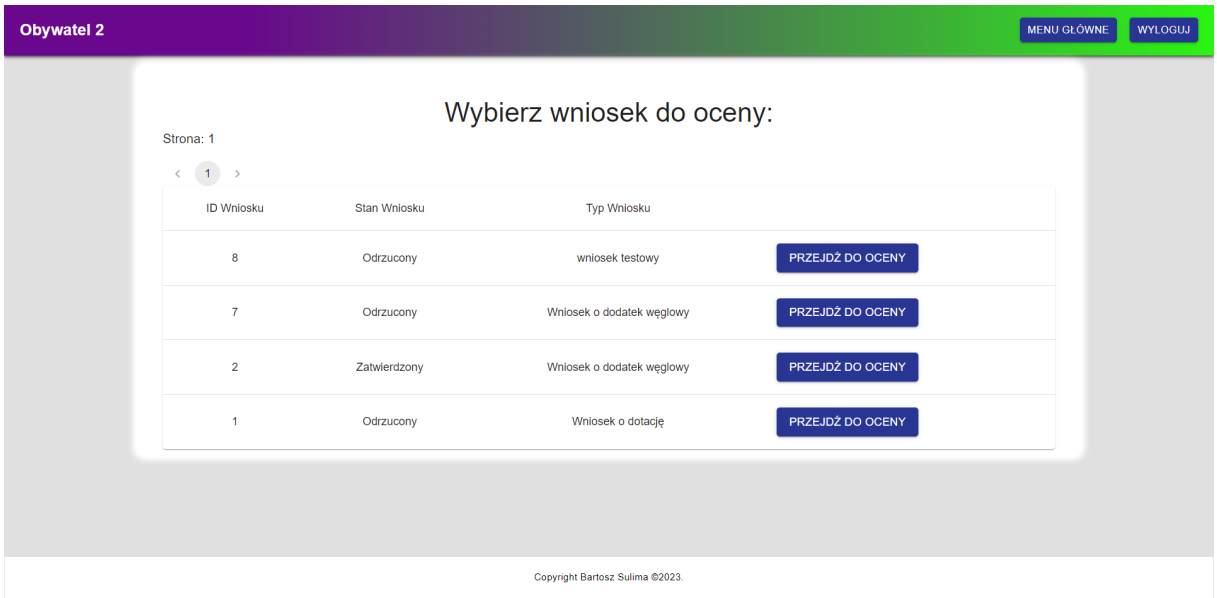
Rys. 6.2: Ekran menu głównego. Źródło: Opracowanie własne.

Końcowy wygląd ekranu stanu wszystkich wniosków został przedstawiony na rysunku 6.3.



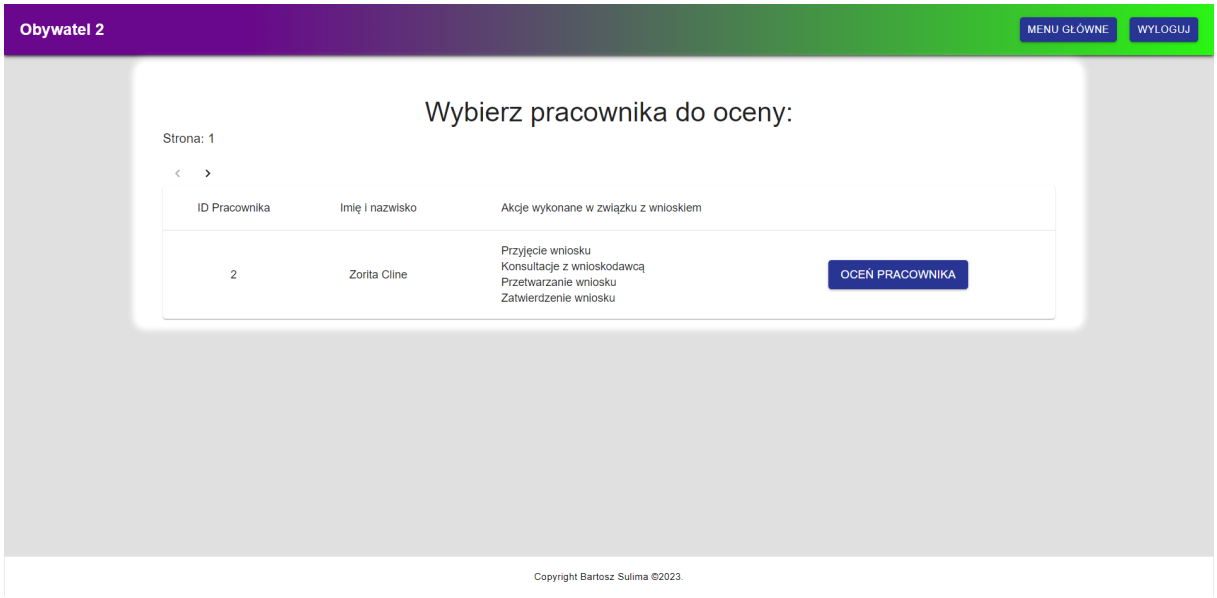
Rys. 6.3: Ekran stanu wszystkich wniosków. Źródło: Opracowanie własne.

Końcowy wygląd ekranu wyboru wniosku do oceny został przedstawiony na rysunku 6.4.



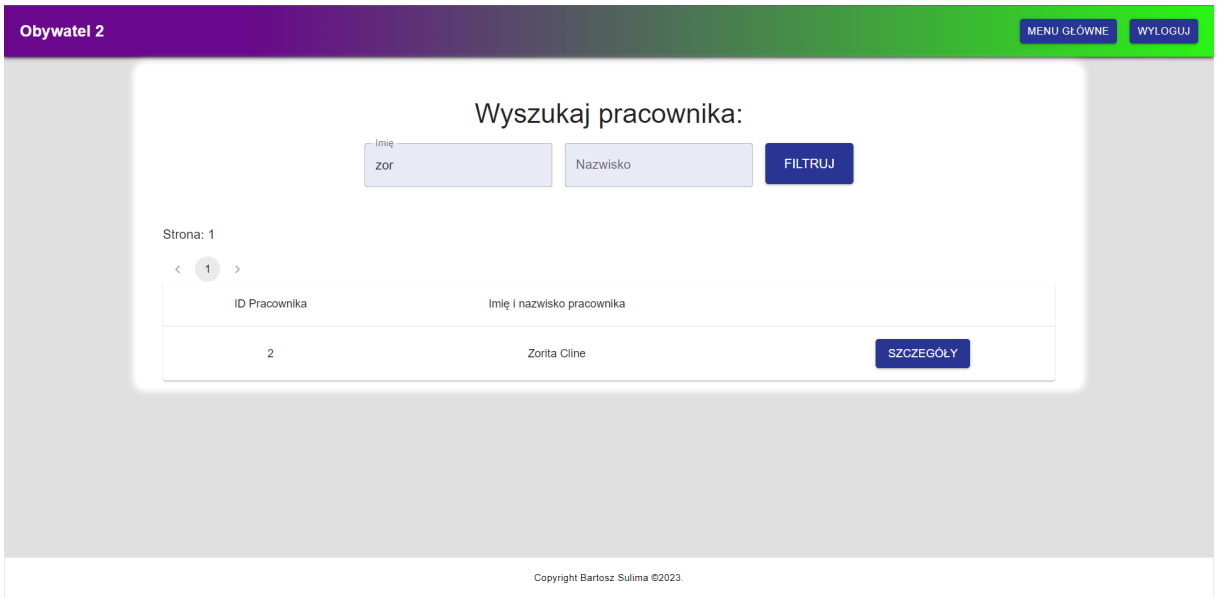
Rys. 6.4: Ekran wyboru wniosku do oceny. Źródło: Opracowanie własne.

Końcowy wygląd ekranu wyboru pracownika do oceny został przedstawiony na rysunku 6.5.



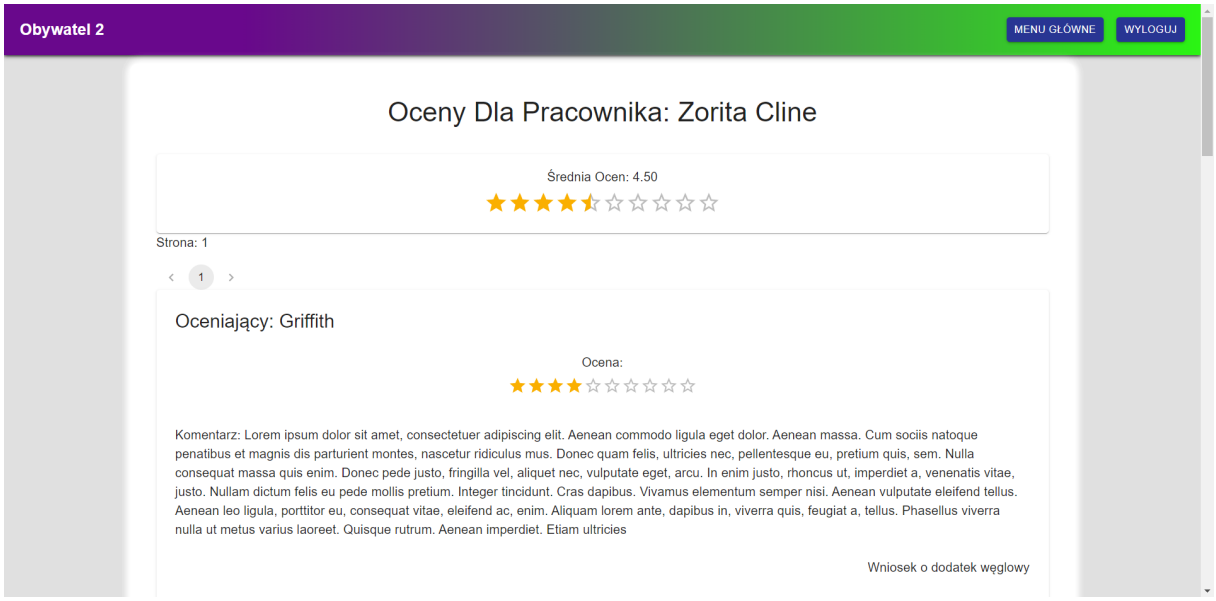
Rys. 6.5: Ekran wyboru pracownika do oceny. Źródło: Opracowanie własne.

Końcowy wygląd ekranu przeglądania pracowników. został przedstawiony na rysunku 6.6.



Rys. 6.6: Ekran przeglądania pracowników. Źródło: Opracowanie własne.

Końcowy wygląd ekranu ocen pracownika został przedstawiony na rysunku 6.7.



Rys. 6.7: Ekran ocen pracownika. Źródło: Opracowanie własne.

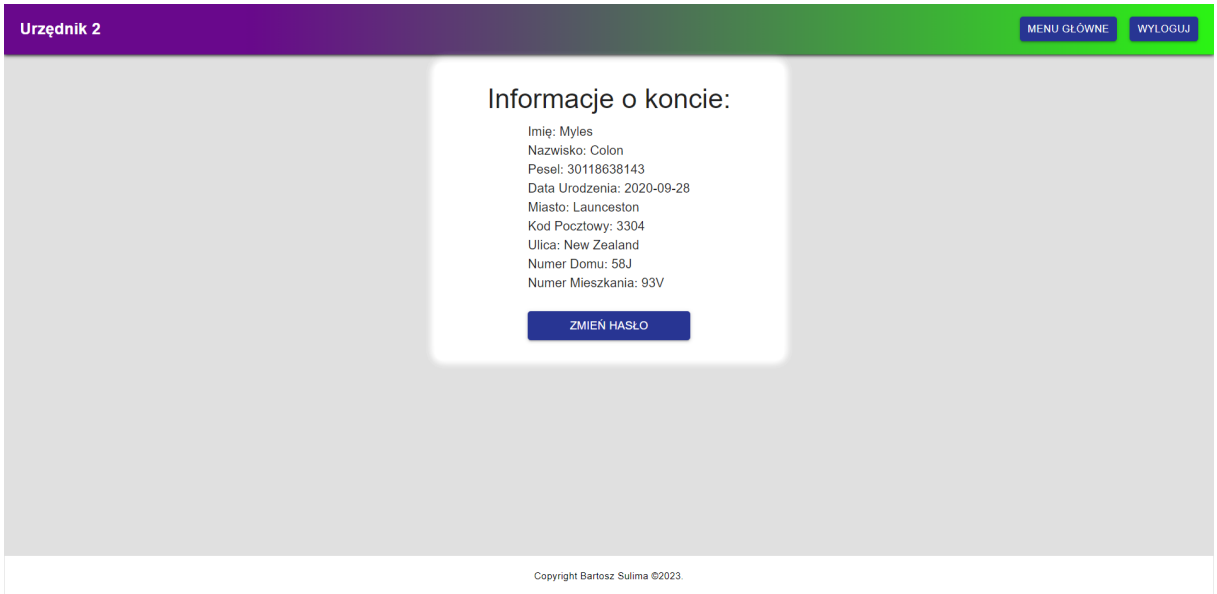
Rozdział 7

Interfejs Pracownika

7.1. Wstęp

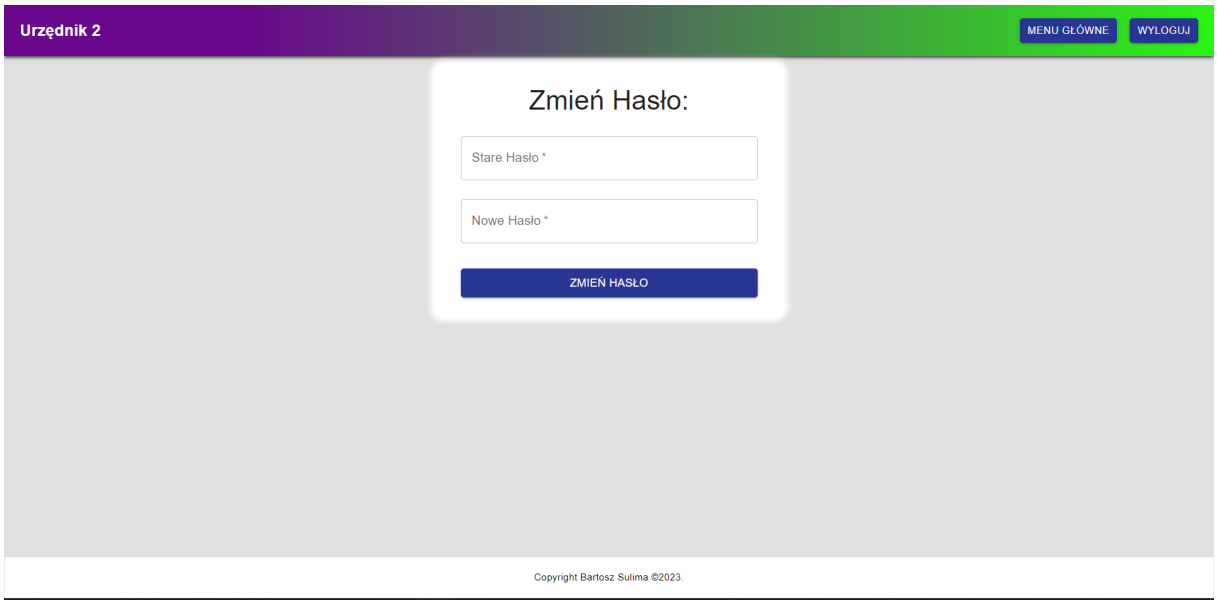
Poniższy rozdział przedstawia końcowy wygląd aplikacji pracownika. W rozdziale tym pominięto część ekranów które były identyczne pod względem wyglądu i funkcji co ekrany w aplikacji obywatela. Odpowiedniki pominiętych ekranów znajdują się w rozdziale opisującym końcowy wygląd aplikacji obywatela 6.

Końcowy wygląd ekranu szczegółów konta został przedstawiony na rysunku 7.1.



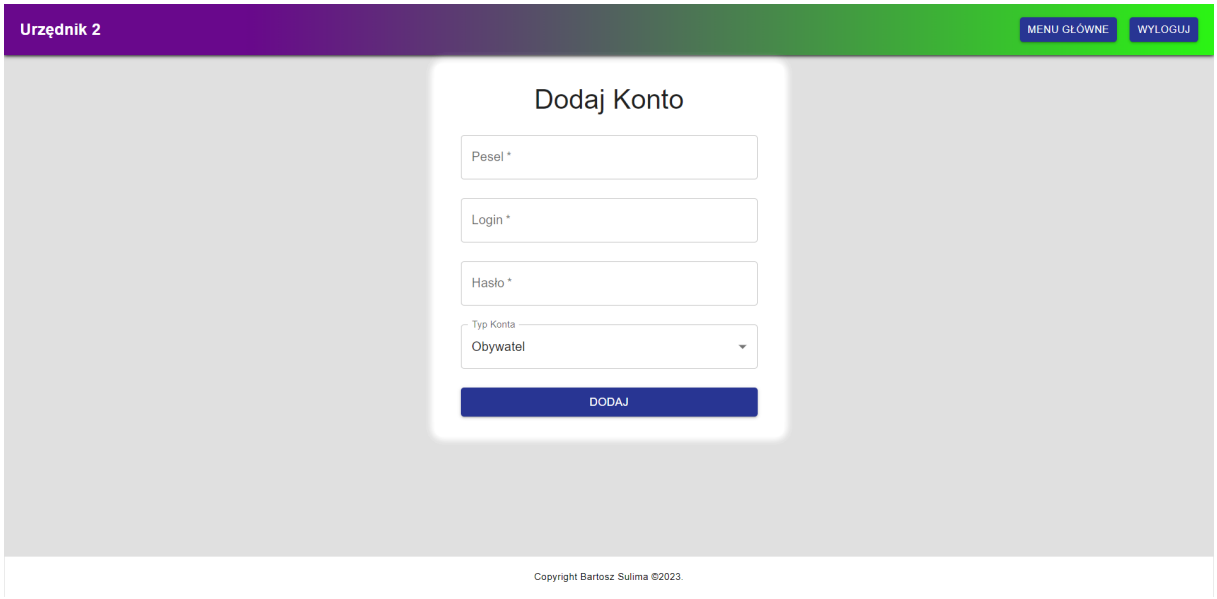
Rys. 7.1: Ekran szczegółów konta. Źródło: Opracowanie własne.

Końcowy wygląd ekranu zmiany hasła został przedstawiony na rysunku 7.2.



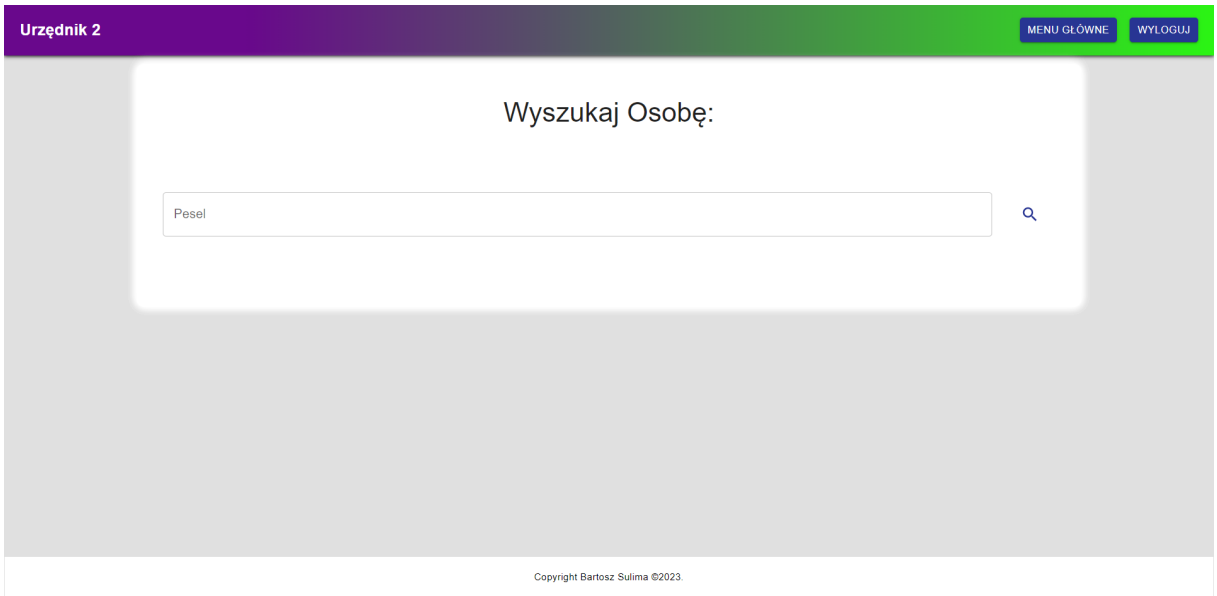
Rys. 7.2: Ekran zmiany hasła. Źródło: Opracowanie własne.

Końcowy wygląd ekranu dodawania nowych kont użytkowników został przedstawiony na rysunku 7.3.



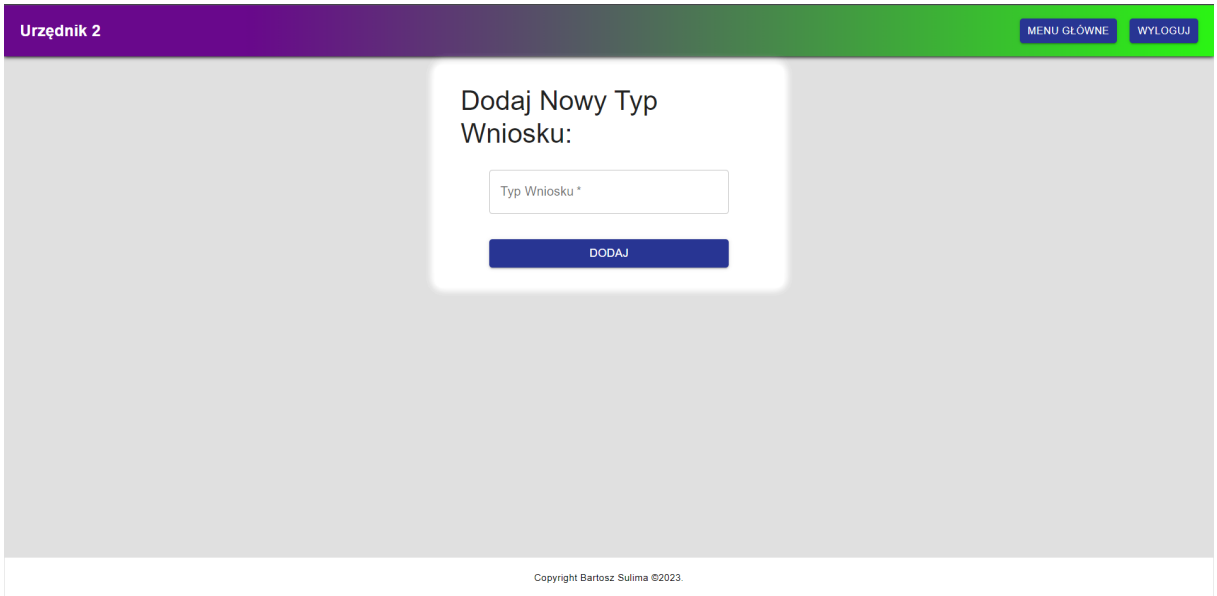
Rys. 7.3: Ekran dodawania nowych kont użytkowników. Źródło: Opracowanie własne.

Końcowy wygląd ekranu wyszukiwania osób został przedstawiony na rysunku 7.4.



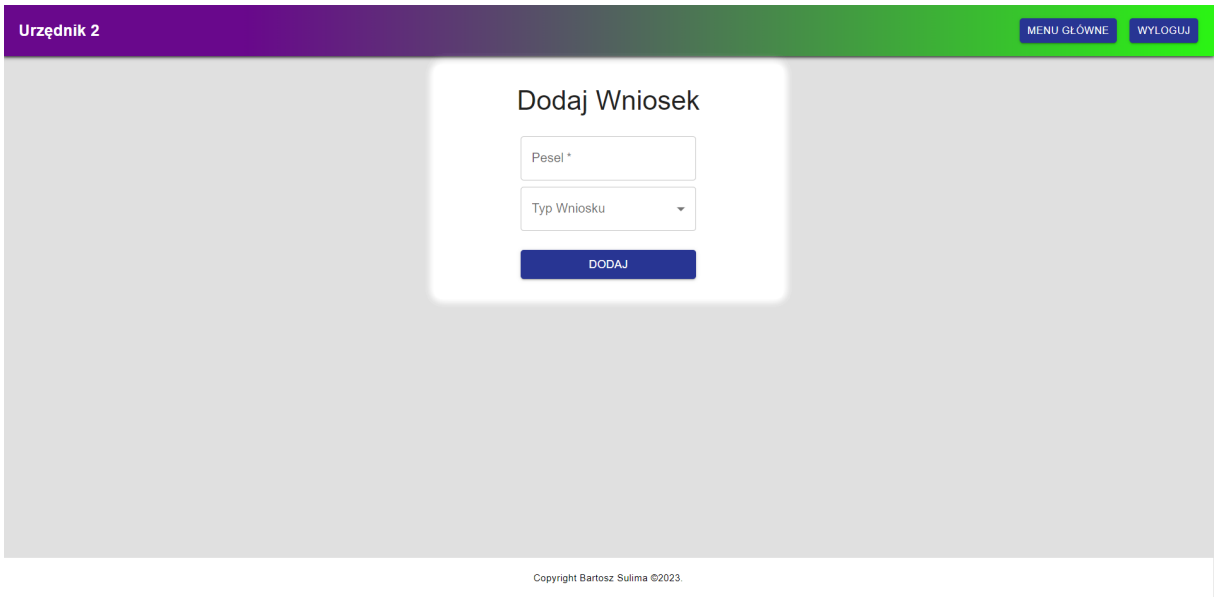
Rys. 7.4: Ekran wyszukiwania osób. Źródło: Opracowanie własne.

Końcowy wygląd ekranu dodawania nowych typów wniosków został przedstawiony na rysunku 7.5.



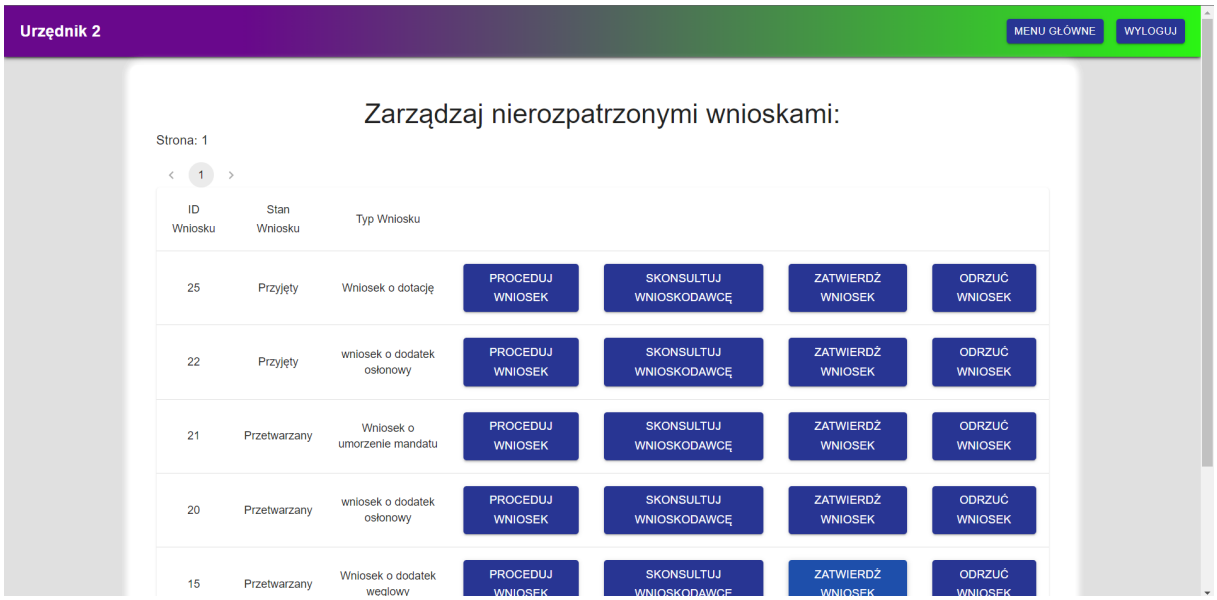
Rys. 7.5: Ekran dodawania nowych typów wniosków. Źródło: Opracowanie własne.

Końcowy wygląd ekranu dodawania nowych wniosków został przedstawiony na rysunku 7.6.



Rys. 7.6: Ekran dodawania nowych wniosków. Źródło: Opracowanie własne.

Końcowy wygląd ekranu zarządzania nierozpatrzonymi wnioskami został przedstawiony na rysunku 7.7.



Rys. 7.7: Ekran zarządzania nierozpatrzonymi wnioskami. Źródło: Opracowanie własne.

Rozdział 8

Testy

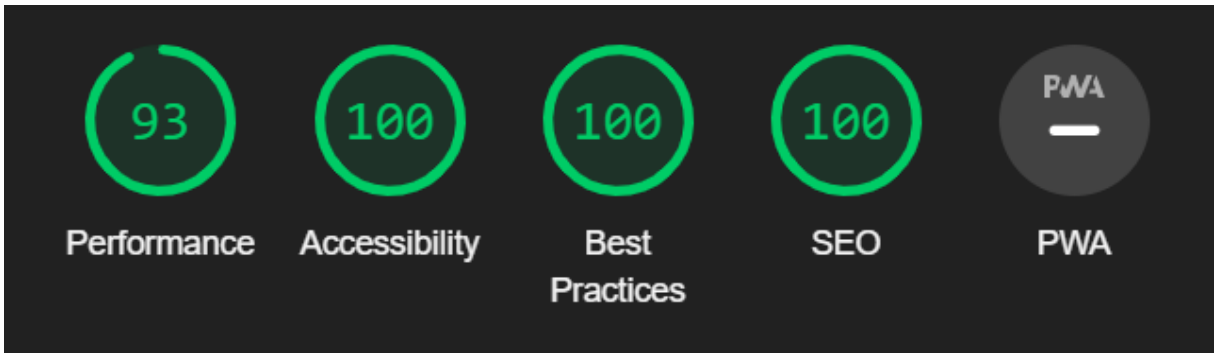
8.1. Testy Lighthouse

8.1.1. Wstęp

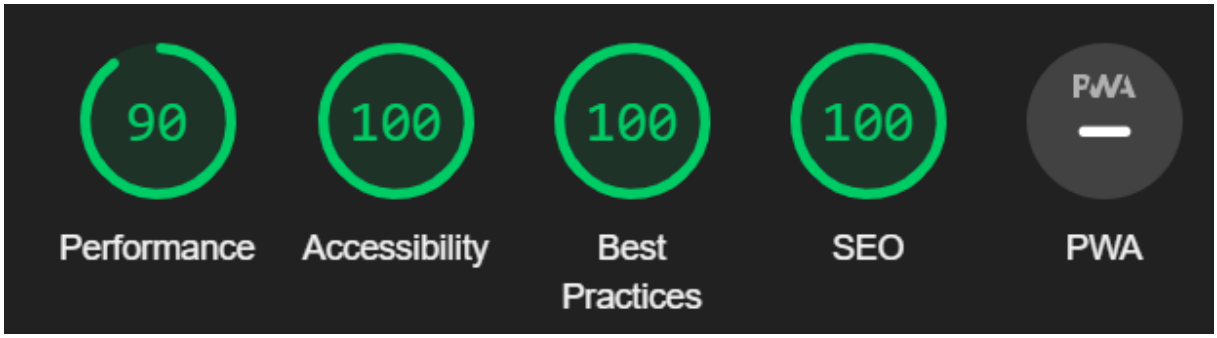
W poniższym rozdziale przedstawiono wyniki testów Lighthouse[2] w aplikacji obywatela i pracownika.

Lighthouse - narzędzie do poprawy jakości stron internetowych. Narzędzie to testuje stronę internetową pod kątem wydajności, dostępności, dobrych praktyk i optymalizacji silnika wyszukiwania.

8.1.2. Wyniki



Rys. 8.1: Wyniki testu Lighthouse aplikacji obywatela. Źródło: Opracowanie własne.



Rys. 8.2: Wyniki testu Lighthouse aplikacji pracownika. Źródło: Opracowanie własne.

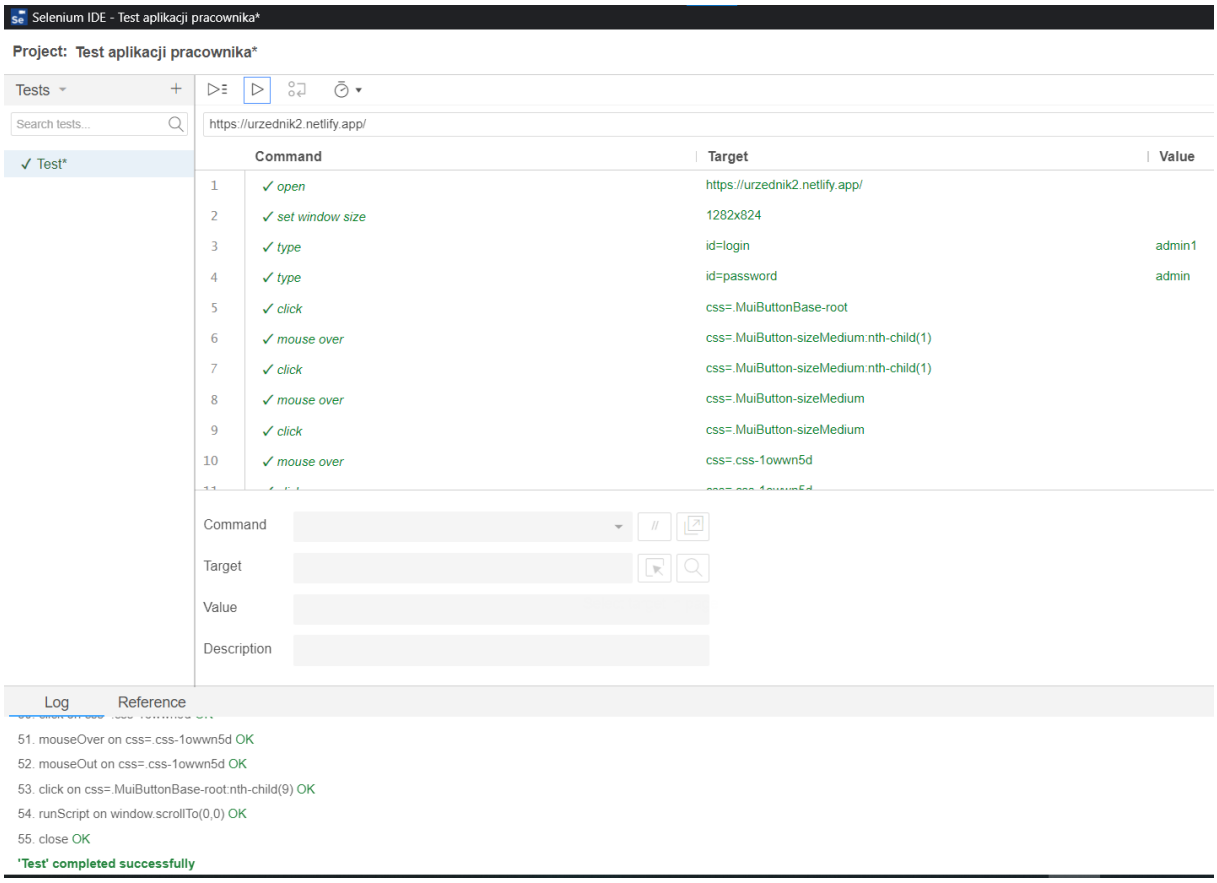
8.2. Testy Selenium IDE

8.3. Wstęp

W poniższym rozdziale przedstawiono wyniki testu Selenium IDE[8].

Selenium IDE - Narzędzie do automatyzacji testów funkcjonalnych dla aplikacji webowych. Pozwala na rejestrowanie i zapisywanie skryptów testów, a także na ich późniejsze uruchamianie i wykonywanie. Selenium IDE posiada łatwy w użyciu interfejs graficzny, który umożliwia szybkie tworzenie skryptów testowych.

Selenium IDE pozwala na testowanie wielu aspektów aplikacji, takich jak przykładowo: funkcjonalność przeglądania, wypełnianie formularzy i elementy interfejsu użytkownika.



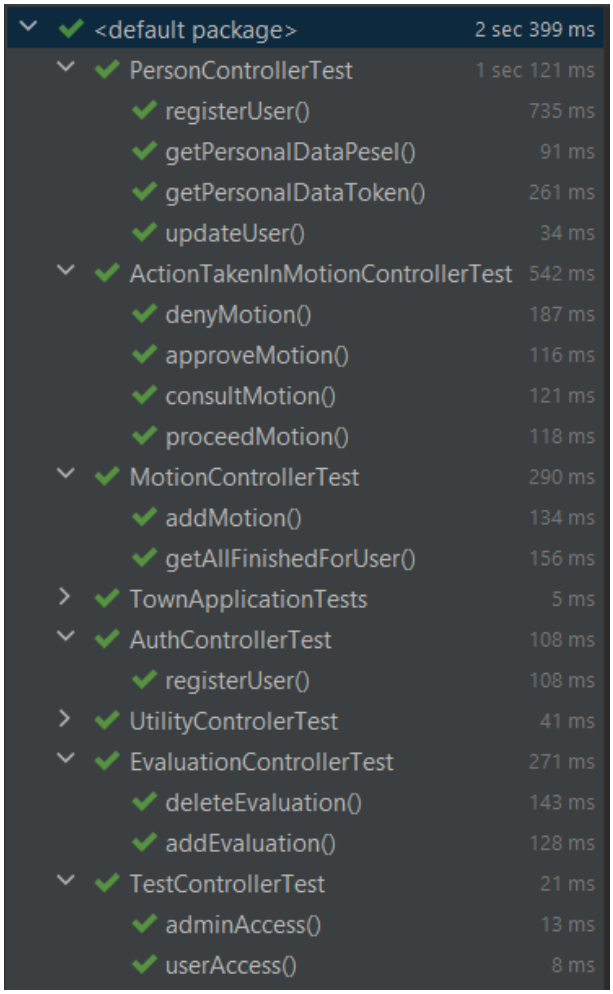
Rys. 8.3: Wyniki testu Selenium IDE. Źródło: Opracowanie własne.

8.4. Testy integracyjne

8.4.1. Wstęp

W poniższym rozdziale przedstawiono listingi wybranych testów integracyjnych, które zostały wykonane podczas testowania serwera.

Testy integracyjne - rodzaj testów, w których sprawdzana jest integracja między elementami i modułami aplikacji. Testy te są wykonywane w celu zapewnienia, że wszystkie elementy aplikacji współpracują ze sobą poprawnie. Mogą one być wykonywane przy użyciu różnych narzędzi i metod, takich jak testy jednostkowe, automatyzowane skrypty, a także ręczne testy[21].



Rys. 8.4: Wyniki testów przeprowadzonych w programie Intellij IDEA. Źródło: Opracowanie własne.

8.4.2. Przykłady testów integracyjnych

Kod metody testowej przedstawionej na listingu 8.1 obejmuje sprawdzenie czy po wysłaniu właściwych danych na ednpoint odpowiedzialny za dodawanie nowych osób do bazy danych, osoba te jest pomyślnie dodana do bazy danych. Metoda ta wymaga uprawnień pracowniczych żeby móc zostać wykonana.

Listing 8.1: Metoda testująca Dodawanie nowych danych osobowych do systemu

```
@Test
@Transactional
@WithMockUser(roles = "ADMIN")
```

```
void registerUser() throws Exception {
    //GIVEN
    //WHEN

    MediaType MEDIA_TYPE_JSON_UTF8 = new MediaType("application", "json",
        ↪ StandardCharsets.UTF_8);


    MvcResult mvcResult = mockMvc.perform(post("/person/add")
                    .accept(MEDIA_TYPE_JSON_UTF8).contentType(
                        ↪ MEDIA_TYPE_JSON_UTF8)
                    .content("""
                            {
                              "date_of_birth": "2023-02-02",
                              "name": "test",
                              "surname": "test2",
                              "pesel": "22222222222",
                              "city": "city",
                              "city_code": "ccode",
                              "street": "rybna",
                              "house_number": "22",
                              "flat_number": "5"
                            }"""))
            .andDo(print())
            .andExpect(status().is(200))
            .andReturn();
    //THEN
    String resp = mvcResult.getResponse().getContentAsString();
    assertThat(resp).isEqualTo("{\"message\":\"Dodano osobę\"}");


    List<PersonalData> personaldatas = personalDataRepository.findAll();


    assertThat(personaldatas.get(0).getName()).isEqualTo("test");
    assertThat(personaldatas.get(0).getStreet()).isEqualTo("rybna");
    assertThat(personaldatas.get(0).getSurname()).isEqualTo("test2");
    assertThat(personaldatas.get(0).getDateOfBirth().toLocalDate()).
        ↪ isEqualTo(Date.valueOf("2023-02-02").toLocalDate());
    assertThat(personaldatas.get(0).getPesel()).isEqualTo("22222222222");
    assertThat(personaldatas.get(0).getCity()).isEqualTo("city");
    assertThat(personaldatas.get(0).getCityCode()).isEqualTo("ccode");
    assertThat(personaldatas.get(0).getHouseNumber()).isEqualTo("22");
    assertThat(personaldatas.get(0).getFlatNumber()).isEqualTo("5");

}
```

Kod metody testowej przedstawionej na listingu 8.2 obejmuje sprawdzenie czy po wysłaniu właściwych danych na ednpoint odpowiedzialny za dodawanie nowych kont użytkowników do bazy danych, konto to jest pomyślnie dodane do bazy danych. Metoda ta wymaga uprawnień pracowniczych żeby móc zostać wykonana.

Listing 8.2: Metoda testująca dodawanie nowego konta do systemu

```
@Test
@Transactional
@WithMockUser(roles = "ADMIN")
void registerUser() throws Exception {

    PersonalData personalData = new PersonalData();
    personalData.setName("Mirosław");
    personalData.setSurname("Łąbądź");
    personalData.setDateOfBirth(Date.valueOf(LocalDate.now()));
    personalData.setPesel("11111111111");
    personalData.setCityCode("58-150");
    personalData.setCity("Strzegom");
    personalData.setStreet("Rybna");
    personalData.setHouseNumber("20");
    personalData.setFlatNumber("2");

    personalDataRepository.save(personalData);

    //Users


    //WHEN
    MediaType MEDIA_TYPE_JSON_UTF8 = new MediaType("application", "json",
        ↪ StandardCharsets.UTF_8);
    MvcResult mvcResult = mockMvc.perform(post("/auth/register")
                    .accept(MEDIA_TYPE_JSON_UTF8).contentType(
                        ↪ MEDIA_TYPE_JSON_UTF8)
                    .content(String.format("""
                            {
                              "pesel": %s,
                              "login": "test",
                              "password": "abba",
                              "role": "ROLE_ADMIN"
                            }""", personalData.getPesel())))
            .andDo(print())
            .andExpect(status().is(200))
            .andReturn();
    //THEN
    String resp = mvcResult.getResponse().getContentAsString();
    assertThat(resp).isEqualTo("{\"message\":\"Użytkownik zarejestrowany\"}
        ↪ ");



    List<Users> users = usersRepository.findAll();
    assertThat(users.get(0).getPermissionLevel()).isEqualTo("ROLE_ADMIN");
    assertThat(users.get(0).getLogin()).isEqualTo("test");
    assertThat(users.get(0).getPersonalDataForUsers().getPesel()).isEqualTo
        ↪ (personalData.getPesel());

}
```

Literatura

[1] Portal Similarweb służący do analizy ruchu na stronach internetowych. https://www.similarweb.com/ [dostęp dnia 20 stycznia 2023].

[2] Dokumentacja narzędzia testowego lighthouse. https://developer.chrome.com/docs/lighthouse/overview/ [dostęp dnia 23 stycznia 2023].

[3] Oficjalna strona biblioteki material ui. https://mui.com/ [dostęp dnia 20 stycznia 2023].

[4] Oficjalna strona języka java. https://www.java.com/pl/ [dostęp dnia 19 stycznia 2023].

[5] Oficjalna strona programu Jetbrains IntelliJ IDEA. https://www.jetbrains.com/idea/ [dostęp dnia 21 stycznia 2023].

[6] Oficjalna strona programu Jetbrains Webstorm. https://www.jetbrains.com/webstorm/ [dostęp dnia 22 stycznia 2023].

[7] Oficjalna strona programu PostgreSQL. https://www.postgresql.org/about/[dostęp dnia 21 stycznia 2023].

[8] Oficjalna strona selenium ide. https://www.selenium.dev/selenium-ide/help [dostęp dnia 25 stycznia 2023].

[9] Portal empic.com. www.empik.com/[dostęp dnia 10 stycznia 2023].

[10] Portal google play store. https://play.google.com/store/games?hl=pl [dostęp dnia 10 stycznia 2023].

[11] Portal gowork.pl. https://www.gowork.pl/[dostęp dnia 10 stycznia 2023].

[12] Portal Wappalyzer służący do analizy technologii wykorzystanych na stronach internetowych. https://www.wappalyzer.com/ [dostęp dnia 20 stycznia 2023].

[13] Portal znanylekarz.pl. www.znanylekarz.pl [dostęp dnia 24 stcznia 2022].

[14] D. Arias. Hashing in Action: Understanding bcrypt. *auth0.com*, 2021. https://auth0.com/blog/hashing-in-action-understanding-bcrypt/ [dostęp dnia 24 stcznia 2022].

[15] P. Carbonnelle. PYPL PopularitY of Programming Language. *pypl.github.io*, 2022. https://pypl.github.io/PYPL.html[dostęp dnia 10 stycznia 2023].

[16] M. Cordova. Understanding Use Cases, Use Case Scenarios, User Stories, Flow Charts. *krasamo.com*, List. 2020. https://www.krasamo.com/understanding-use-cases-use-case-scenarios-user-stories-flow-charts/ [dostęp dnia 21 stycznia 2023].

[17] T. Dohmke. 100 million developers and counting. *github.blog*, Sty. 2023. https://github.blog/2023-01-25-100-million-developers-and-counting/ [dostęp dnia 26 stcznia 2022].

[18] D. Herbert. What is React.js? (Uses, Examples, & More). *blog.hubspot.com*, Czerw. 2022. https://blog.hubspot.com/website/react-js [dostęp dnia 25 stycznia 2023].

[19] Kinga. JPA #2 – ORM, JPA i Hibernate. *clockworkjava.pl*, List. 2020. https://clockworkjava.pl/2020/11/jpa-orm-jpa-i-hibernate/ [dostęp dnia 25 stycznia 2023].

[20] O. G. Mark Pollack, Thomas Risberg. Working with Spring Data Repositories. *docs.spring.io*, 2022. https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html [dostęp dnia 24 stcznia 2022].

[21] T. MIELNIK. Testy integracyjne. *Proceedings of ICALEPCS07*, Czerw. 2022. https://testerzy.pl/baza-wiedzy/artykuly/testy-integracyjne [dostęp dnia 28 stycznia 2023].

[22] MSWiA. Satysfakcja z obslugi i dostepu do informacji publicznej w urzedach jednostek samorzadu terytorialnego. *gov.pl*, 2011. www.gov.pl/web/mswia-instytucja-posredniczaca/satysfakcja-z-obslugi-i-dostepu-do-informacji-publicznej-w-urzedach-jednostek-samorzadu-terytorialnego---raport-z-badania-omnibusowego-zrealizowanego-na-zlecenie-ministerstwa-spraw-wewnetrznych-i-administracji [dostęp dnia 20 stycznia 2023].

[23] Oficjalna strona JWT. Introduction to JSON Web Tokens. *jwt.io*, 2022. https://jwt.io/introduction [dostęp dnia 22 stcznia 2022].

[24] S. Overflow. 2021 Developer survey. *insights.stackoverflow.com*, 2021. https://insights.stackoverflow.com/survey/2021#most-popular-technologies-tools-tech-prof[dostęp dnia 10 stycznia 2023].

[25] N. Raval. Top 10 JavaScript Usage Statistics to Watch Out for in 2022. *radixweb.com*, Wrze. 2022. https://radixweb.com/blog/top-javascript-usage-statistics [dostęp dnia 25 stycznia 2023].

[26] South-Portland. Citizen satisfaction survey. *southportland.org*, 2009. www.southportland.org/our-city/about-south-portland/citizen-satisfaction-survey/ [dostęp dnia 20 stycznia 2023].

[27] D. Spinellis. Git. *IEEE Software*, 2012. https://ieeexplore.ieee.org/abstract/document/6188603?casa_token=7RnxQ6LKT78AAAAA:JBtl9bc37NptlDKugQbUUEu0K9KI3RemmkpDrgCvPK0vAX_d6-zL5MxlU4z_hiYH6r9Zwf3z [dostęp dnia 24 stcznia 2022].

[28] StatCounter. Desktop Windows Version Market Share Worldwide. *gs.statcounter.com*, 2023. https://gs.statcounter.com/windows-version-market-share/desktop/worldwide/[dostęp dnia 10 stycznia 2023].

[29] V. Tanzu. Spring Security. *spring.io*, 2022. https://spring.io/projects/spring-security#overview[dostęp dnia 10 stycznia 2023].

[30] C. Wanstrath. We Launched. *github.blog*, Kwi. 2008. https://github.blog/2008-04-10-we-launched/ [dostęp dnia 24 stcznia 2022].

[31] P. Ćwik. Obiekt domenowy, DTO, DAO. *clockworkjava.pl*, Gru. 2020. https://clockworkjava.pl/2020/12/obiekt-domenowy-dto-dao/ [dostęp dnia 25 stycznia 2023].

Dodatek A

Instrukcja wdrożeniowa

W celu uruchomienia oprogramowania, które zostało stworzone na potrzeby tej pracy należy zainstalować:

- Wirtualną maszynę Java w wersji 19 lub nowszą.
- PostgreSQL w wersji 14 lub nowszej.

Po zainstalowaniu wymaganych programów należy stworzyć bazę danych w PostgreSQL o następujących parametrach:

- Nazwa : Town_Database
- Użytkownik: postgres
- Hasło : DtSBbb5HJv

Po stworzeniu bazy danych należy uruchomić aplikację pgAdmin4 i otworzyć "querry tool"w odpowiedniej bazie danych a następnie skopiować do niego zawartość skryptu inicjalizującego bazę dołączonego do pracy i go wykonać. Następnie należy uruchomić plik .jar zawierający aplikacje serwerową poprzez dwukrotne kliknięcie na plik .bat (Oba pliki znajdują się na płycie i muszą znajdować się w tym samym katalogu, aby działać poprawnie). W Celu wdrożenia aplikacji webowych można wykorzystać serwis netify.com. Zbudowane wersje aplikacji znajdują się na płycie. Po wykonaniu powyższych kroków można się zalogować do aplikacji pracowniczej z następującymi parametrami:

- Login : admin
- Hasło : admin

Po zalogowaniu się należy dodać nowe dane osobowe i stworzyć dla nich nowe konta obywateli i/lub pracowników.

Dodatek B

Opis załączonej płyty CD/DVD

Na płycie zamieszczono dokument `pdf` z tekstem powyższej pracy wraz z załącznikami w postaci:

- Plik .jar potrzebny do uruchomienia aplikacji serwerowej.
- Plik .bat, przy którego pomocy można uruchomić aplikację serwerową (Oba pliki muszą się znajdować w jednym folderze żeby działać poprawnie).
- Folder plików zawierający pliki niezbędne do uruchomienia aplikacji obywatela.
- Folder plików zawierający pliki niezbędne do uruchomienia aplikacji urzędnika.
- Skrypt zawierający kod potrzebny do zainicjalizowania bazy danych w PostgreSQL.