Spis Treści

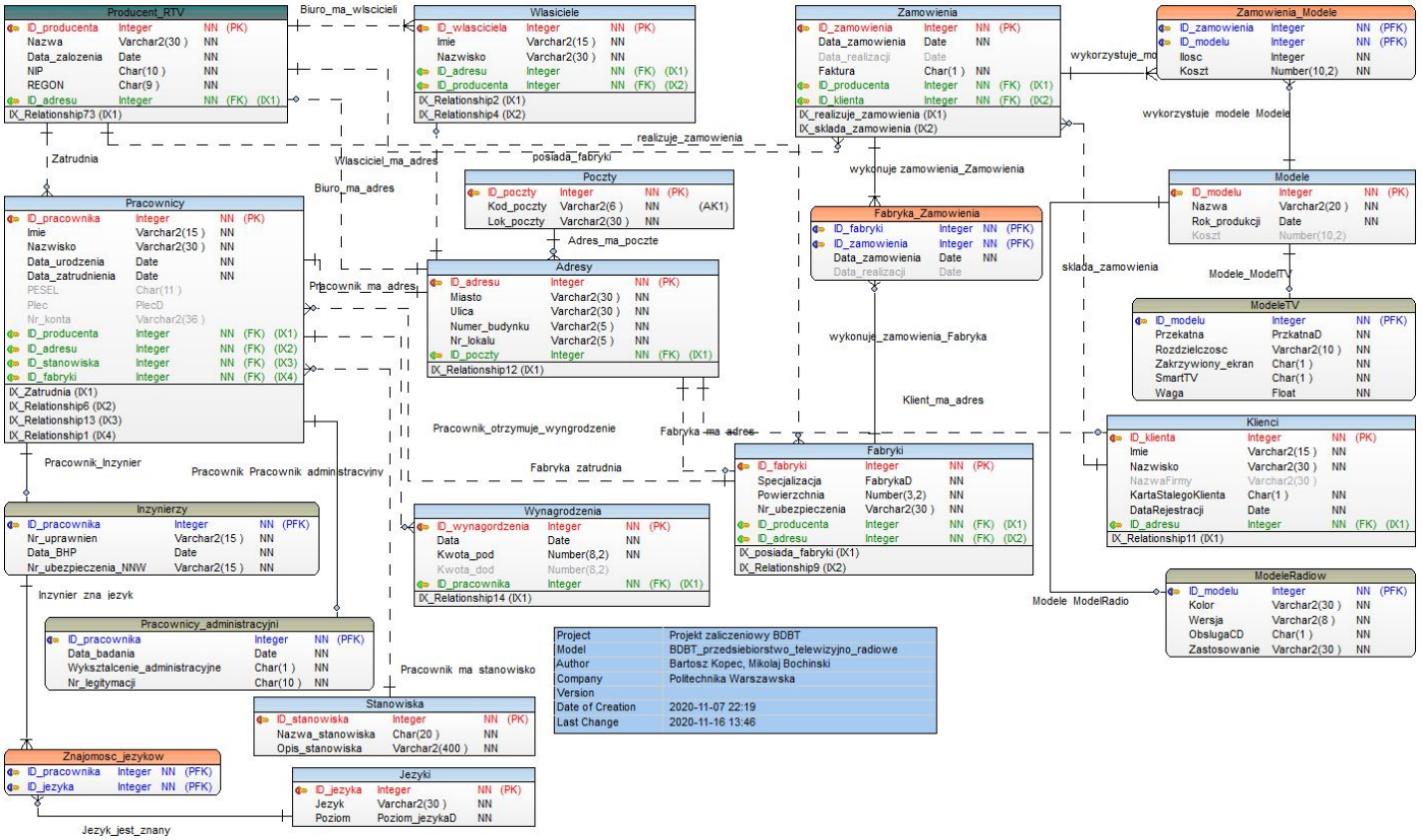
1. Zakres i cel projektu

1.1. Cel

Celem drugiej części projektu z przedmiotu BDBT jest przygotowanie aplikacji służącej do obsługi uproszczonej bazy danych przygotowanej w pierwszej części projektu. Aplikacja przygotowana w architekturze wielowarstwowej (strona www - aplikacja klienta - serwer bazy danych).

1.2. Założenia i zakres

Aplikacja obsługuje cztery spośród dziewiętnastu relacji dostępnych w bazie danych "Przedsiębiorstwa Telewizyjno-radiowego". Są to tabele: Pracownicy (ang. Employees), Adresy (ang. Addresses), Fabryki (ang. Factories) oraz Stanowiska (ang. Positions). Zdecydowano się na nie ze względu możliwość pokazania podstawowych funkcjonalności aplikacji - CRUD - tj.:
(C)reate - dodawanie rekordów do bazy,
(R)ead - odczytywanie danych z bazy,
(U)pdate - aktualizacja krotek,
(D)elete - usuwanie danych.
Aplikacja będzie oferowała dwie odmienne perspektywy, które będą korzystały z różnych funkcjonalności CRUD. Pierwsza będzie perspektywą administratora, z pełnym dostępem do tabel oraz funkcjonalności systemu.



Rys. 1 - model relacyjny bazy

Reprezentuje to użytkownika, który w razie potrzeby "ręcznie" mógłby nanieść jakieś poprawki do bazy, w przypadku błędów innych użytkowników systemu (np. błąd w danych pracownika podczas rejestracji). Drugą perspektywą jest widok Księgowości. Ma ona wgląd do danych wszystkich użytkowników oraz fabryk, przy czym nie ma uprawnień do ich zmiany, dodawania, czy usuwania. Wyjątkiem jest możliwość zmiany numeru konta pracownika. Reprezentuje to użytkownika, który pracuje w dziale księgowości i ma wgląd do danych pracowników.
Obsługa perspektyw jest zapewniona poprzez prosty system logowania (wykorzystujący moduł Spring Security), który po zapytaniu o login i hasło wyświetla odpowiedni interfejs GUI.

2. Wykorzystane technologie

2.1. Java

Aplikacja została napisana przy wykorzystaniu środowiska Java oraz edytora Eclipse. W ramach rozszerzenia funkcjonalności zastosowano rozszerzenie OJDBC oraz frameworki opisane poniżej.
W programie zostały utworzone odpowiednie klasy, pozwalające na obsługę bazy. Każda wczytywana relacja jest reprezentowana przez odpowiednie dwie klasy - klasę macierzystą oraz klasę Data Access Object (DAO). Klasa macierzysta jest wzorcem poszczególnych krotek wczytanych z bazy i posiada zmienne odzwierciedlające pola z tabeli. Są w niej również zaimplementowane odpowiednie metody GET/SET w celu modyfikacji i podglądu danych. Klasa DAO jest jednolitym interfejsem pozwalającym na komunikację z bazą danych i pobranie odpowiednich danych.

```
public class Employees {

        private int id_pracownika;
        private String imie;
        private String nazwisko;
        private String data_urodzenia;
        private String data_zatrudnienia;
        private String PESEL;
        private String plec;
        private String nr_konta;
        private int id_producenta;
        private int id_adresu;
        private int id_stanowiska;
        private int id_fabryki;
(...)
public int getId_pracownika() {
        return id_pracownika;
}

public void setId_pracownika(int id_pracownika) {
        this.id_pracownika = id_pracownika;
}
(...)
```

Wycinek z kodu nr. 1 - klasa Employees

Ze względu na wykorzystane frameworki, kod aplikacji zawiera charakterystyczne elementy składniowe oraz pewne specyficzne klasy wspomagające obsługę programu. Specyfika

projektu oraz jego założenia implikują wykorzystanie poza językiem Java również fragmentów języka HTML oraz XML.

2.2. Spring

Spring jest platformą złożoną z wielu projektów, która dedykowana jest do tworzenia aplikacji w języku Java. Jego kluczowym elementem jest kontener wstrzykiwania zależności, który daje możliwość niejawnego tworzenia obiektów oraz automatycznego dodawania funkcjonalności do tworzonego projektu. Zapewnia to wygodę, ponieważ w wypadku modyfikacji funkcjonalności aplikacji, zmiany dokonuje się w jednym miejscu, a następnie Spring zajmuje się resztą zmian. W naszym projekcie zostały wykorzystane w głównej mierze trzy moduły - Spring Boot, Spring Data oraz Spring Security.
Pierwszy z nich jest podstawowym projektem Springa, który pozwala na szybką budowę aplikacji w oparciu o przygotowaną kolekcję szablonów startowych. Dla konkretnego template'u wszystkie wymagane zależności zostaną automatycznie dociągnięte i podpięte pod projekt.
Drugi moduł jest projektem upraszczającym dostęp do baz danych. Główną ideą Spring Data jest zminimalizowanie ilości powtarzalnego kodu, czyli przykładowo jeśli nasza aplikacja wykorzystuje JPA, potrzebujemy stworzyć repozytorium udostępniające podstawowe metody CRUD, to korzystając ze Spring Data całość sprowadza się do stworzenia jednego prostego interfejsu.
Trzeci moduł - Spring Security - jest wykorzystywany do zabezpieczania projektu aplikacji oraz zapewnia podstawowe interfejsy m. in. do logowania i kontroli sesji użytkownika.

2.3. Maven

Maven jest narzędziem automatyzującym budowę oprogramowania przeznaczonego na platformę Java. Jest on pewnym uproszczeniem i uzupełnieniem, bardzo silnie związanym z innymi frameworkami - m.in. Java Spring. Plik określający sposób budowy aplikacji nosi nazwę POM (Project Object Model). Jest to plik XML'a kompleksowo opisujący projekt. Zawiera informacje o wersjach wykorzystanego oprogramowania, zależnościach zachodzących w projekcie, podpiętych framework'ach i innych charakterystykach projektu.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
         <modelVersion>4.0.0</modelVersion>

         <!-- Project details -->
         <groupId>bdbt_proj</groupId>
         <artifactId>Producer</artifactId>
         <version>0.0.1-SNAPSHOT</version>

         <!-- Java details -->
```

```
	<properties>
		<java.version>11</java.version>
	</properties>

	<!-- SpringBoot initialization -->
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-parent</artifactId>
		<version>2.4.0</version>
		<relativePath>Producer/pom.xml</relativePath>
	</parent>
(...)
```

Wycinek z kodu nr. 2 - plik POM.xml

2.4. Thymeleaf

Thymeleaf to silnik szablonów umożliwiający budowę elementów interfejsu użytkownika po stronie serwera (backendu). Co ciekawe znajduje też zastosowanie w budowie aplikacji, które nie są webowe. Pozwala definiować własne atrybuty szablonów lub znaczników, jednak domyślnie zawiera standardowe dialekty (Standard i SpringStandard), które definiują bogaty zestaw funkcjonalności. Standardowe dialekty są łatwe do rozpoznania, gdyż zawierają atrybuty zaczynające się od przedrostka **th**.

3. Prezentacja GUI oraz działania aplikacji

3.1. Strona startowa

Na początku, po wejściu na stronę aplikacji, wyświetla się strona startowa. Przycisk “SIGN IN” przenosi do strony logowania.

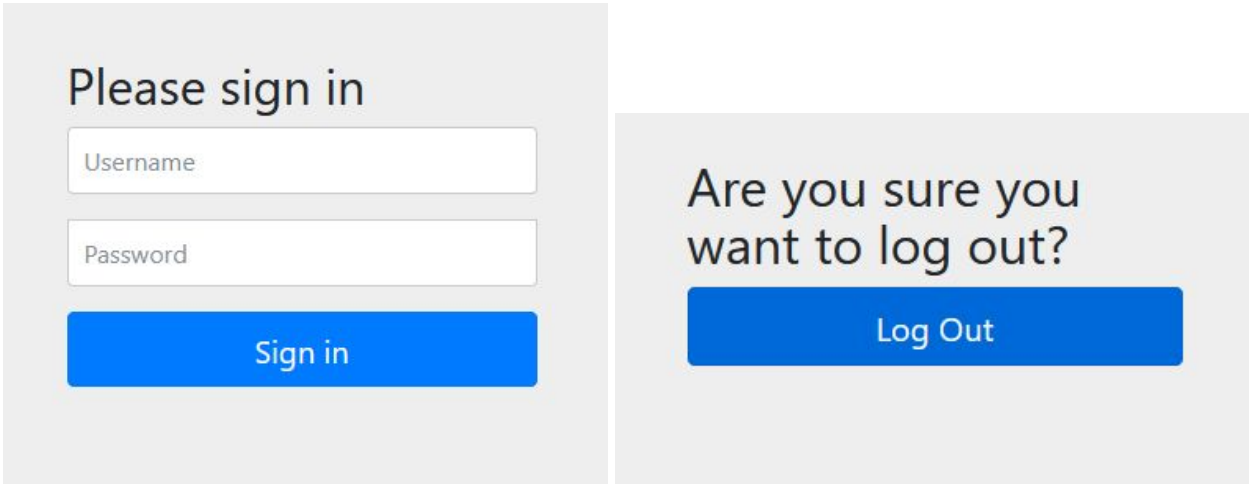
Rys. 2 - strona główna

3.2. System logowania

Wymagane jest zalogowanie się na jedno z kont dostępnych w systemie. Na potrzeby projektu zostały utworzone następujące konta:

- *admin* - konto z perspektywą szeroką
- *user* - konto z perspektywą wąską

Szata graficzna odbiega od reszty aplikacji ze względu na zastosowanie domyślnego szablonu strony logowania. Pozostała część systemu posiada ujednoliconą szatę graficzną, stworzoną przy pomocy środowiska *Cascade Style Sheet*.

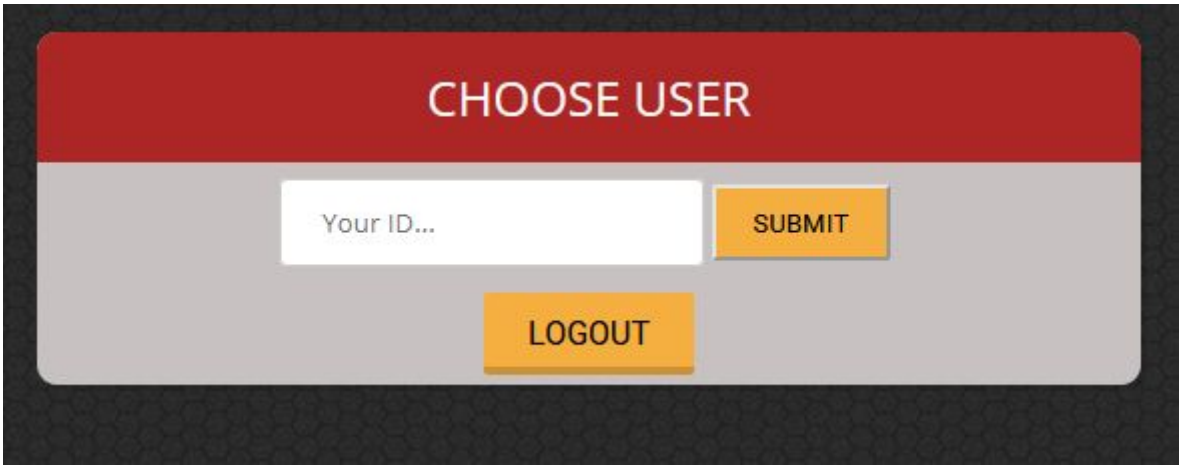


Rys. 3 - interfejs logowania

3.3. Prezentacja perspektyw aplikacji

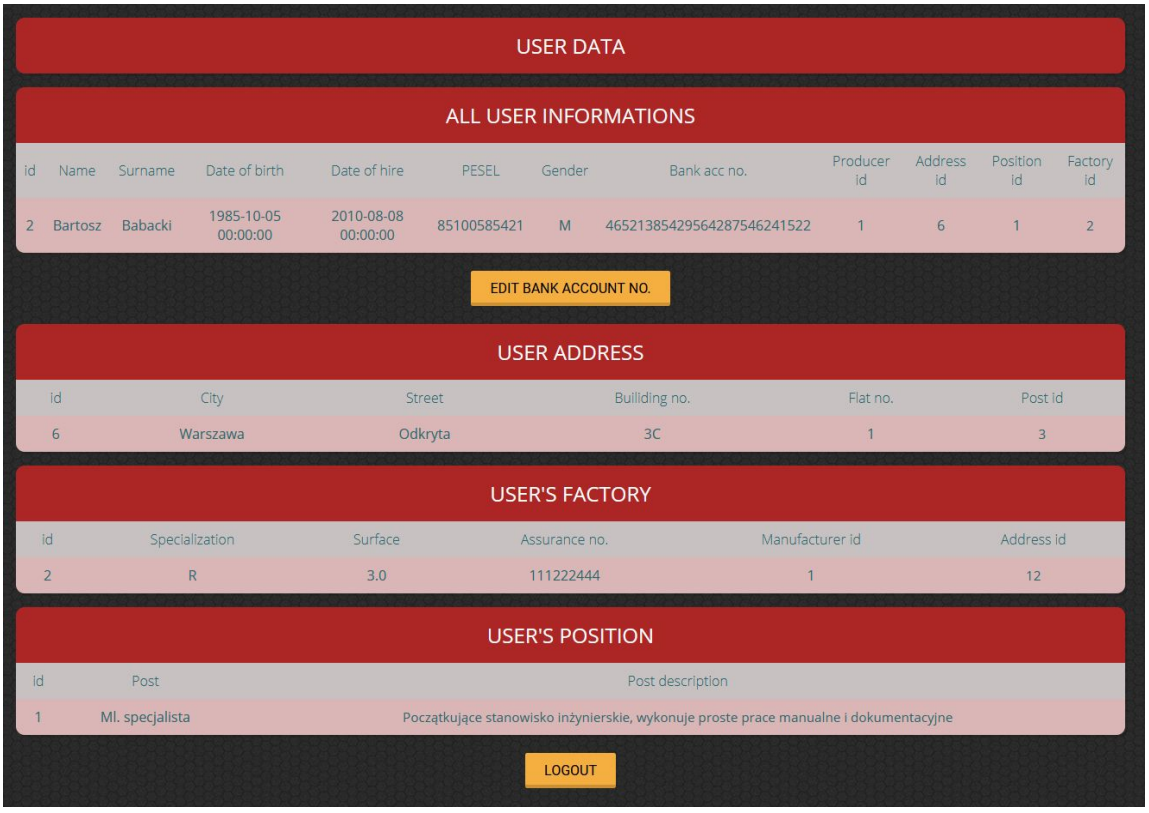
3.3.1. Perspektywa wąska

Po zalogowaniu się jako użytkownik *user*, aplikacja przechodzi do okna wyboru pracownika. W tym miejscu mamy możliwość wpisania identyfikatora pracownika i przejścia do strony z jego danymi.



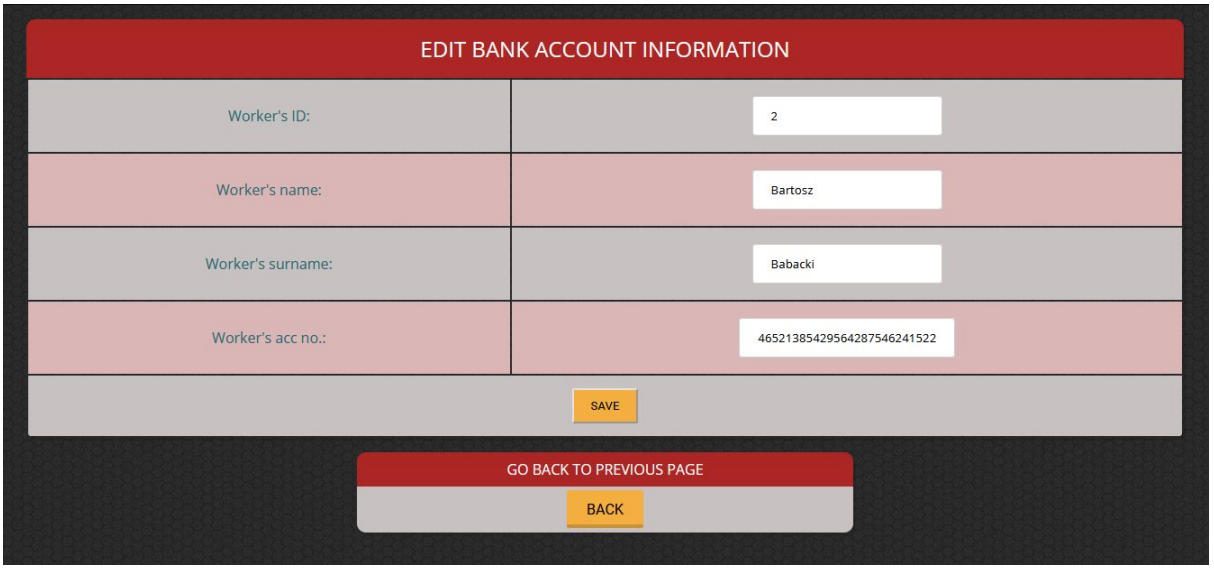
Rys. 4 - okno wyboru pracownika

Poniżej zaprezentowano przykładowy widok danych pracownika oraz fabryki i pozycji jaką zajmuje w przedsiębiorstwie. Naciśnięcie przycisku “EDIT BANK ACCOUNT NO.” przenosi do strony dedykowanej do edycji numeru konta pracownika. Natomiast naciśnięcie przycisku “LOGOUT” przekieruje użytkownika do interfejsu logowania z prośbą o potwierdzenie chęci wylogowania (Rys.2)



Rys. 5 - okno danych pracownika

Okno edycji numeru konta oferuje podgląd identyfikatora, imienia i nazwiska pracownika, ale jedynie numer konta może zostać zmieniony. Można cofnąć się do poprzedniej strony, bez wprowadzania zmian, poprzez skorzystanie z przycisku “BACK”.

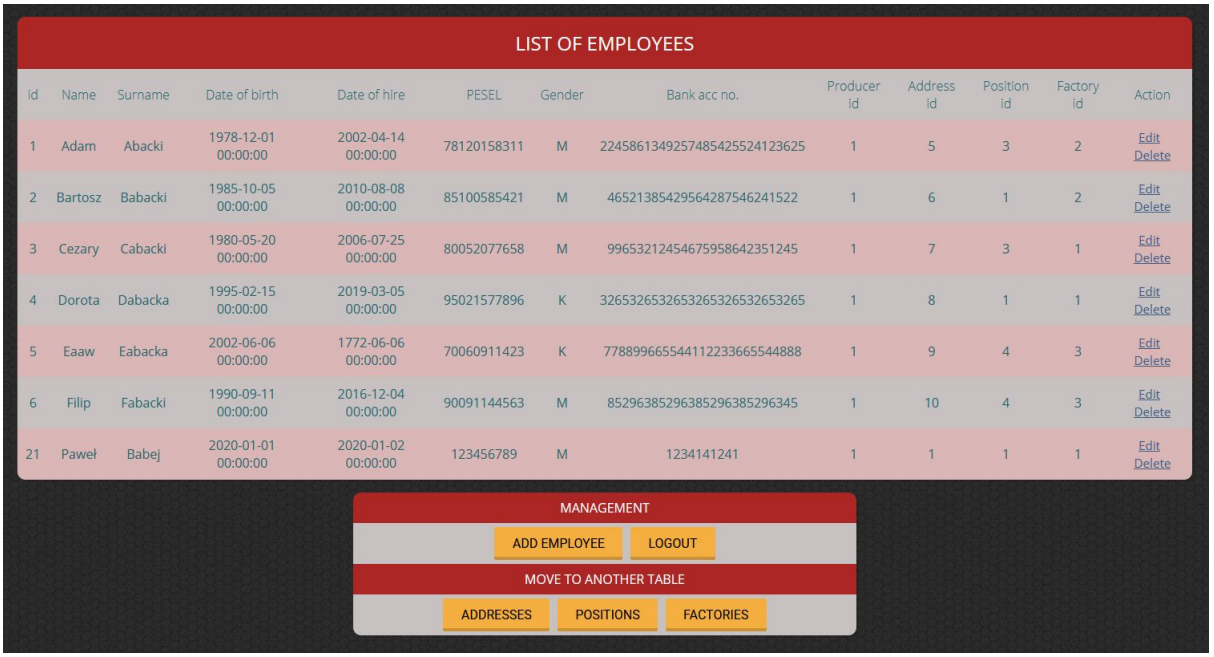


Rys. 6 - modyfikacja numeru konta w perspektywie wąskiej

3.3.2. Perspektywa szeroka

Poniżej zaprezentowano przykładowy widok strony po zalogowaniu na konto administratora. Oczom użytkownika ukazuje się tabela zawierająca dane każdego z pracowników firmy. Strona umożliwia edycję danych oraz dodawanie nowych pracowników, a także przejście do

kolejnych stron z: adresami- przycisk ADDRESSES; stanowiskami- przycisk POSITIONS; fabrykami- przycisk FACTORIES.



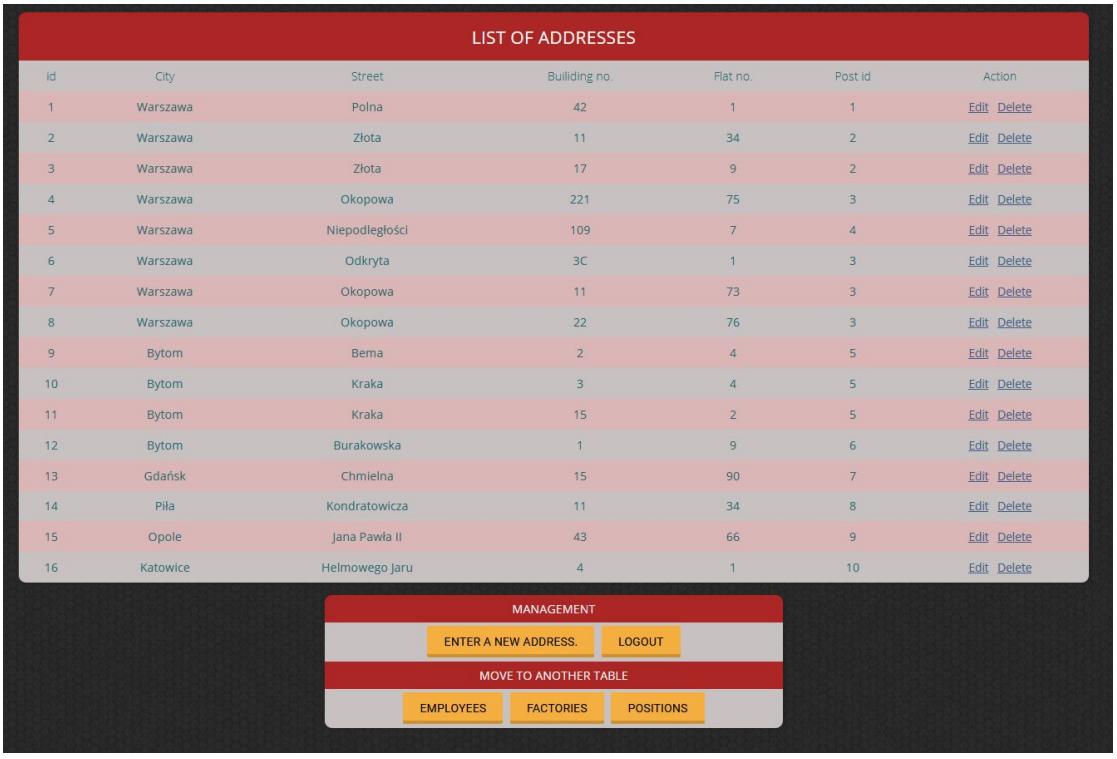
Rys. 7 - tabela z danymi pracowników firmy

Rys. 8 - Panel z przyciskami, które umożliwiają poruszanie się między stronami

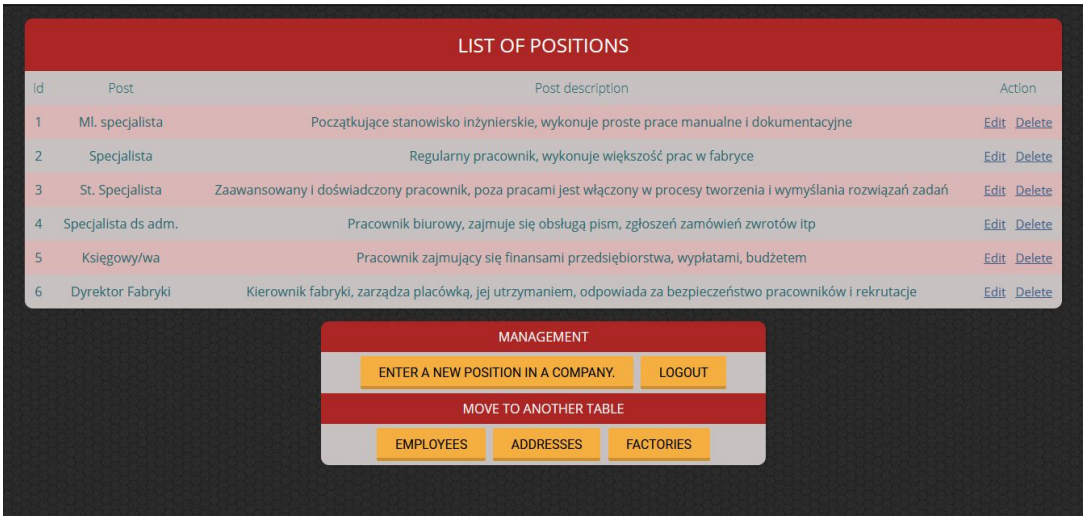
3.4. Operacje CRUD

3.4.1. Wyświetlanie danych (Read)

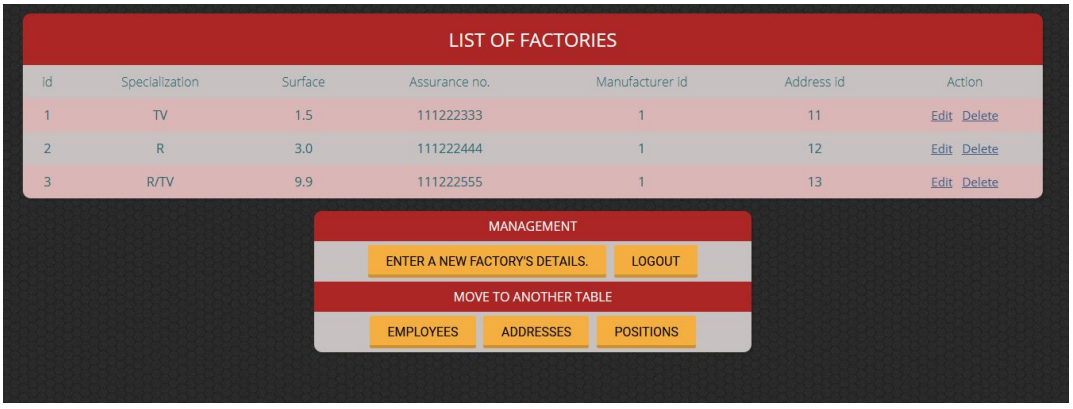
Funkcjonalność ta została zapewniona odrębnie dla obu perspektyw. W poprzednich punktach (3.2.1 oraz 3.2.2) funkcjonalność wczytywania danych z bazy została odpowiednio zaprezentowana. Poniżej w ramach komplementarności sprawozdania umieszczamy zrzuty ekranu prezentujące tablice Adresy, Stanowiska oraz Fabryki.



Rys. 9 - podgląd adresów zarejestrowanych w bazie



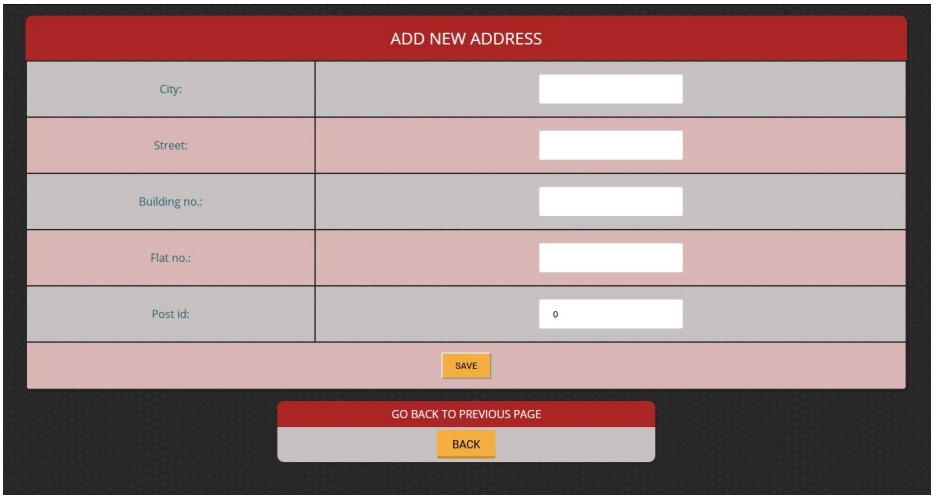
Rys. 10 - podgląd danych w tablicy Stanowiska



Rys. 11 - podgląd danych w tablicy Fabryki

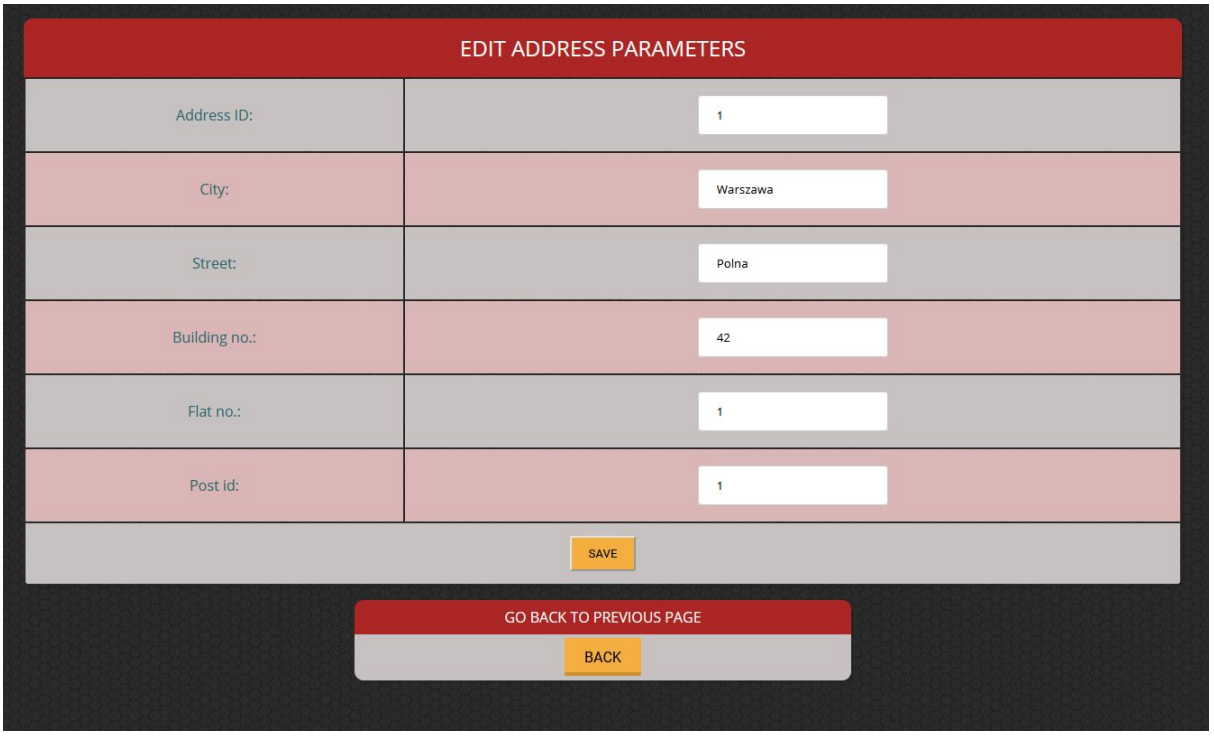
3.4.2. Dodawanie oraz modyfikacja danych (Create & Update)

Użytkownik będący zalogowany do konta z perspektywą szeroką ma możliwość wprowadzania nowych danych do bazy oraz modyfikacji już istniejących. W sekcji "MANAGEMENT" każdej tablicy został umieszczony przycisk z odnośnikiem do strony pozwalającej na dodawanie nowych danych.



Rys. 12 - okno dodawania nowego adresu

Przy każdym wierszu w tabeli jest widoczny przycisk "EDIT", który przenosi do strony edycji danych. Dla wygody użytkownika systemu, dane są automatycznie wczytywane do pól. Nie można edytować identyfikatora danego wiersza. Pozostałe tabele posiadają analogiczne strony do dodawania oraz edycji danych.



Rys. 13 - okno edycji adresu

3.4.3. Usuwanie danych (Delete)

Pod przyciskiem “EDIT” znajduje się drugi - “DELETE”, odpowiedzialny za usuwanie danych z bazy.

4. Bibliografia i źródła

- spotkania projektowe z przedmiotu BDBT, czwartki, mgr inż. Tomasz Mrozek
- slajdy wykładowe z przedmiotu BDBT, dr hab inż. Marcin Kowalczyk
- https://stackoverflow.com/, wiele różnych stron
- https://www.w3schools.com/, kurs SQL
- https://www.oracletutorial.com/oracle-date-functions/oracle-to_date/
- Dokumentacje wykorzystanych framework’ów (Spring, Maven, Thymeleaf)

5. Uwagi do przedmiotu

W naszej opinii przedmiot był ciekawy i rozwijający. Ogólny program jest dobrze przemyślany i zachęca do kontynuacji tematu. Zasadniczym minusem jednakże była stosunkowo mała ilość praktycznego zastosowania języka SQL. Uważamy, że w pierwszej części projektu można by postawić większy nacisk na pisanie kwerend (prostych i złożonych). Ciekawym pomysłem była idea “Złotego Strzału” i zachęcała ona do poprawienia błędów - warto rozważyć zastosowanie jej również w drugiej części projektu.