



Grupa C, zespół 10, nr projektu 31

Kowalska Marcelina
Mianowski Michał

ZOO

Dokumentacja drugiej części projektu z przedmiotu Bazy Danych i Big Data (BDBT)
Semestr 21Z

Spis treści

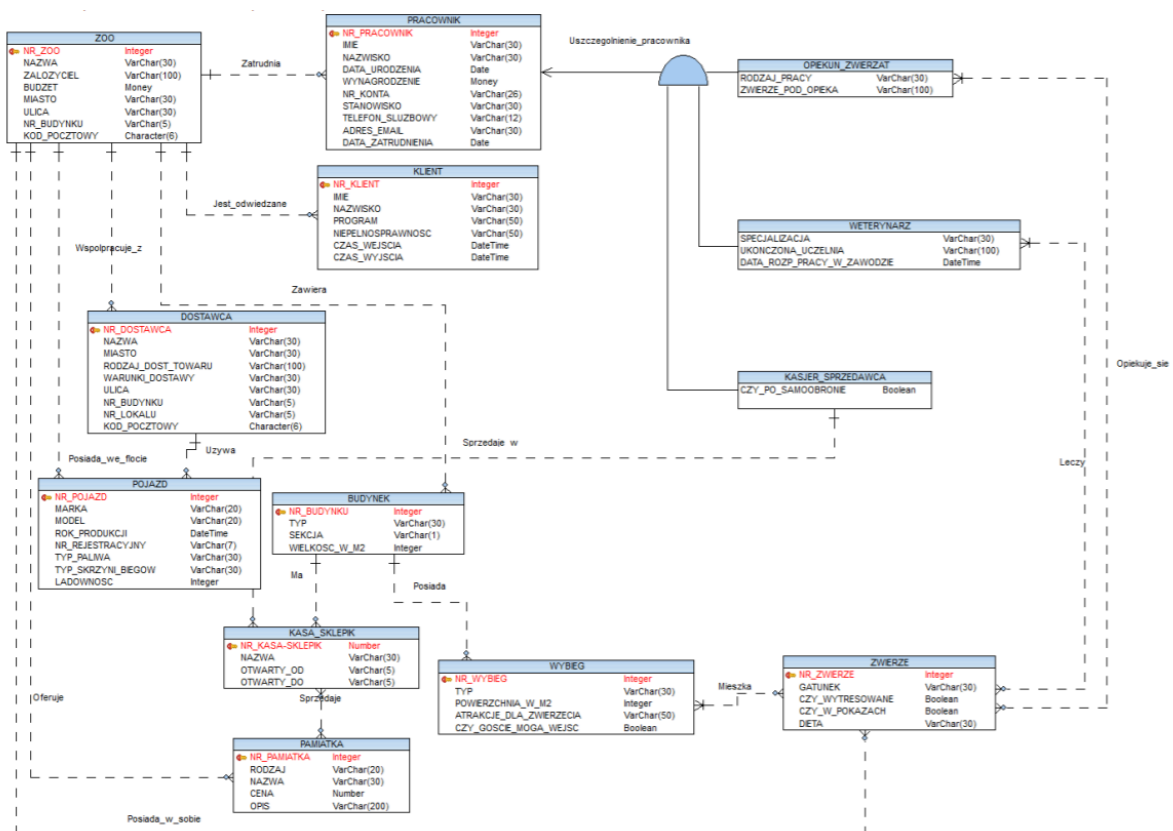
1. Zakres i cel projektu	2
2. Technologie zastosowane w projekcie.	3
2.1 Język Java.	3
2.2 Język HTML.....	4
2.3 Framework Java Spring i Spring Boot.....	4
2.4 Silnik szablonów Thymeleaf.	5
2.5 Maven.	5
3. Prezentacja działania.	6
3.1 Wąska perspektywa - widok gościa w Zoo.....	7
3.2 Szeroka perspektywa – widok dyrektora Zoo.....	9
4. Bibliografia.	12

1. Zakres i cel projektu.

Celem II części projektu z przedmiotu BDBT było przygotowanie przez zespół projektowy aplikacji współpracującej z wcześniej zaprojektowaną bazą danych. Rozwiązanie stworzono w wielowarstwowej architekturze (cienki klient w postaci przeglądarki internetowej, serwerem jest baza danych).

Z dostępnych w bazie danych relacji (rys.1) wybrano najistotniejsze z nich i na ich podstawie stworzono aplikację. Były to następujące relacje: KLIENCI (ang. *Clients*), ZWIERZETA (ang. *Animals*), WYBIEGI (ang. *Enclosures*) oraz PRACOWNICY (ang. *Employees*). Z relacji KLIENCI skorzystano tylko w celu dodania do niej obecnego klienta (po wyborze widoku gościa w oknie startowym) oraz modyfikacji jego danych. Ze względu na wielkość oraz stopień rozbudowania bazy zdecydowano się na zaprezentowanie tylko najważniejszych funkcji – dodawania, modyfikacji, przeglądania oraz usuwania. Oczywiście, używanym silnikiem bazy relacyjnej jest Oracle.

Zamysłem aplikacji było stworzenie prostego symulatora zarządzania Zoo, który mógłby posłużyć do nauki obsługi takiego systemu dla studentów kierunków związanych z zarządzaniem. Oprócz tego, może stanowić również realizację prototypu, konceptu gry przeglądarkowej opierającej się na zarządzaniu posiadanymi zasobami.



Rys. 1 Schemat ER na poziomie konceptualnym.

2. Technologie zastosowane w projekcie.

Aplikacja została napisana w języku Java. IntelliJ IDEA została wybrana jako zintegrowane środowisko programistyczne. Dodatkowo wykorzystano framework Java Spring oraz silnik szablonów Thymeleaf. W rozwiązaniu zastosowano szablon Model-Widok-Kontroler (MVC), który rozdziela aplikację na poszczególne części odpowiedzialne za inne aspekty. Nad logiką aplikacji czuwa model. Widok, jak sama nazwa może wskazywać, opisuje graficzny interfejs użytkownika (GUI). Kontroler służy za obsługę danych wejściowych, odpowiednie reakcje na nie i aktualizacje modelu. Poszczególne technologie zostały opisane w poniższych podpunktach.

2.1 Język Java.

Nie przypadkowo akurat ten język programowania został wybrany do implementacji aplikacji. Najpopularniejsze RDBMS (systemy zarządzania relacyjną bazą danych, ang. *Relational Database Management System*) nie oferują wystarczających rozwiązań jeśli chodzi o graficzne interfejsy dostępu do baz danych oraz zaawansowanych możliwości przetwarzania danych z bazy. W celu nadrobienia tej niedogodności stworzono liczne programistyczne interfejsy (API) dla różnych języków programowania. Niestety i to ma swoje wady: są to biblioteki dynamiczne oraz nieraz skomplikowane dla danych platform systemowych. Java na tle ich wszystkich wyróżnia się swoim API. Jego najważniejszą cechą jest niezależność – od RDBMS, platformy sprzętowej oraz systemu operacyjnego. Uniwersalność tego rozwiązania oraz łatwość użycia powoduje, że to właśnie ten język jest używany do szybkiego i prostego tworzenia aplikacji bazodanowych. Utworzone klasy w tym języku reprezentują pojedyncze rekordy z danych relacji z obsługiwanej bazy danych. Składowe tych klas odzwierciedlają atrybuty poszczególnych relacji. Między innymi dzięki nim jest możliwe realizowanie kolejnych podstawowych operacji na bazie danych. Dla przykładu, obiekty danych klas przechowują dane aktualizowanego rekordu. Każdej takiej klasie towarzyszy inna istotna klasa oferująca wykonywanie określonych poleceń języka SQL - Data Access Class Object. To właśnie one są głównie odpowiedzialne za pobieranie oraz przekazywanie danych z i do bazy danych.

```
// KLASA WYBIEG
public class Enclosure {

    private int id;
    private String typeEnclosure;
    private int surface;
    private String attractions;
    private String areVisitorsAllowed;
    private int nrBuilding;

    public Enclosure(){

    }

    public Enclosure(int id, String type, int surface, String attractions, String guestAllowed, int nrBuild) {
        this.id = id;
        this.typeEnclosure = type;
        this.surface = surface;
        this.attractions = attractions;
        this.areVisitorsAllowed = guestAllowed;
        this.nrBuilding = nrBuild;
    }
}
```

Rys. 2 Widok z IDE na implementację prostej klasy napisanej w języku Java – reprezentacja rekordu z relacji WYBIEGI.

2.2 Język HTML.

Z tym językiem przeciętni użytkownicy Internetu spotykają się nieustannie. Wszystkie strony w Internecie są tworzone za pomocą tego języka. Jest językiem znacznikowym pozwalającym opisać strukturę informacji zawartych wewnątrz strony internetowej, nadając im odpowiednie znaczenie semantyczne danym tekstom. Łatwa edycja wyglądu dokumentu HTML w przeglądarce oraz pełna gama możliwości związanych z kaskadowymi arkuszami stylów czyni go bardzo atrakcyjnym narzędziem do wszelkiego formatowania tekstów oraz ciekawej reprezentacji treści w Internecie. Zarówno w tym projekcie skorzystano z tego i za pomocą arkuszy stylów nadano aplikacji nowoczesny, odpowiadający najnowszym trendom minimalistyczny wygląd.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>ZOO Manager Application</title>
  <link rel="stylesheet" th:href="@{/css/stylesheets.css}"/>
</head>
<body>
  <div align="center">
    <!-- Page header -->
    <h1>Enclosures</h1>

    <!-- Entry to a new page -->
    <a href="/new/enclosure">Enter new enclosure</a>
```

Rys. 3 Fragment jednego z plików z rozszerzeniem html.

2.3 Framework Java Spring i Spring Boot.

Java Spring jest szkieletem tworzenia aplikacji w języku Java. Cechy tego framework'a przyczyniają się do prostego sposobu tworzenia rozwiniętych systemów. Jego najważniejszym elementem jest kontener wstrzykiwania zależności. Umożliwia niejawnie tworzenia obiektów i automatycznego dodawania danej funkcjonalności do projektu. Cechuje się to wygodą, z tego względu, że w sytuacji modyfikacji funkcjonalności aplikacji, zmiany dokonuje się w jednym miejscu. Resztą zmian zajmuje się Spring. Wśród jego zalet wymienia się również wypromowanie oraz rozpowszechnienie różnych funkcji, które nie były kiedyś w wystarczającym stopniu doceniane. Łatwa przyswajalność oraz różnorodność dostępnych modułów i wszelkiego rodzaju rozszerzeń czyni ten framework niezwykle skutecznym rozwiązaniem. Z tego powodu wynika jego niezwykła popularność, a nauka go na pewno przyniesie same korzyści uczącym się go w przyszłości. Wspomniany w tytule podpunkt Spring Boot jeszcze bardziej ułatwia pracę ze Spring'em.

2.4 Silnik szablonów Thymeleaf.

Thymeleaf jest silnikiem szablonów. Dzięki niemu możliwa jest budowa określonych komponentów graficznego interfejsu użytkownika po stronie serwera. Oferuje również definiowanie atrybutów własnych znaczników oraz szablonów. Standardowa wersja ma domyślne dialekty, które i tak zawierają szeroki wachlarz funkcjonalności. Można je łatwo rozpoznać w pliku po charakterystycznym przedrostku *th* przy atrybucie.

```
<!--/*@thymesVar id="enclosure" type="zoo_webapp.zoomanager.Enclosure.Enclosure"*/-->
<!--/*@thymesVar id="list" type="java.util.List<zoo_webapp.zoomanager.Enclosure.Enclosure>"*/-->
<tr th:each = "enclosure : ${list}">
  <!--/*@thymesVar id="enclosure" type="zoo_webapp.zoomanager.Enclosure.Enclosure"*/-->
  <!--/*@thymesVar id="id" type="java.lang.Number"*/-->
  <td th:text = "${enclosure.id}">ID</td>
  <!--/*@thymesVar id="type" type="java.lang.String"*/-->
  <td th:text = "${enclosure.typeEnclosure}">Type</td>
  <!--/*@thymesVar id="surface" type="java.lang.Number"*/-->
  <td th:text = "${enclosure.surface}">Surface</td>
  <!--/*@thymesVar id="attractions" type="java.lang.String"*/-->
  <td th:text = "${enclosure.attractions}">Attractions for Animals</td>
```

Rys. 4 Zastosowanie silnika Thymeleaf w projekcie.

2.5 Maven.

Narzędzie Apache Maven automatyzuje budowę oprogramowania napisanego w języku Java. Projekt jest opisany w specjalnym pliku noszącym nazwę *pom.xml*. Ten dokument XML-owy zawiera szczegółowe informacje dotyczące projektu. Tymi informacjami są m.in. występujące w projekcie zależności, wersje wykorzystanego oprogramowania, wszystkie podpięte framework'i oraz inne charakterystyczne cechy projektu.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.4.1</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>zoo_webapp</groupId>
<artifactId>zoomanager</artifactId>
<version>0.0.1-SNAPSHOT</version>

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

<dependencies>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
  </dependency>
```

Rys. 5 Fragment zastosowanego w projekcie pliku *pom.xml*.

3. Prezentacja działania.

Przed rozpoczęciem działania aplikacji jest nawiązywane połączenie z bazą danych znajdującą się na serwerze wydziału. Za tą funkcję odpowiada klasa DBConnector. Udostępnia ona statyczną metodę *connect*, która właśnie nawiązuje połączenie. Utworzone połączenie jest przechowywane jako zmienna statyczna *connectionDB* w tej klasie.

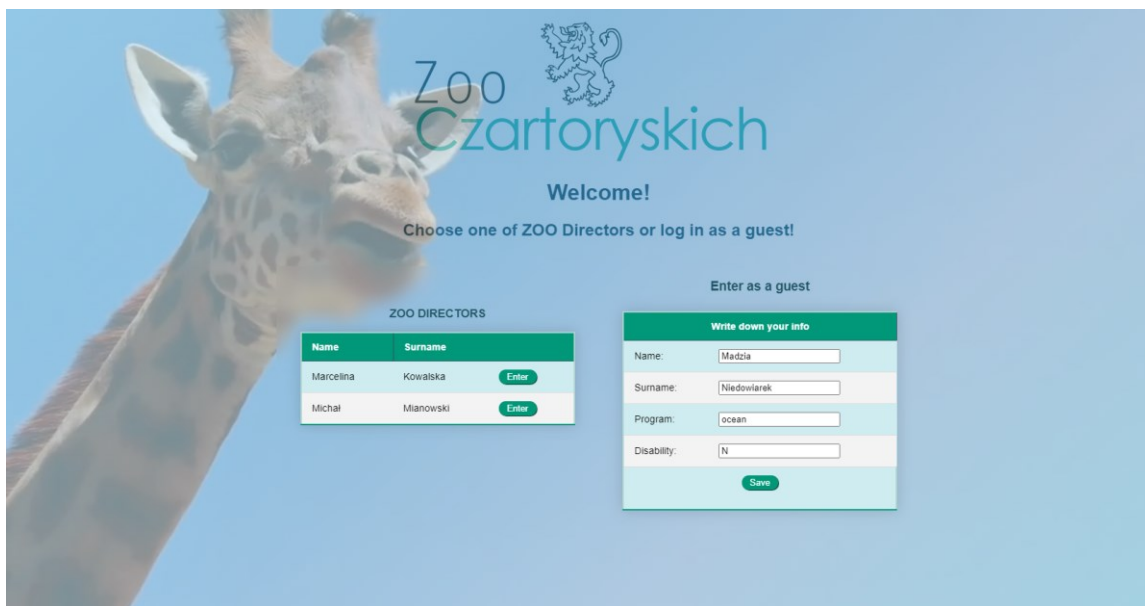
```
public static Connection connectionDB;
```

Rys. 6 Zmienna przechowująca połączenie do bazy danych (serwera).

```
public static void connect(){
    Connection connection = null;
    try {
        connection = DriverManager.getConnection(url,user,password);
        System.out.println("Connected!");
    } catch (SQLException e) {
        e.printStackTrace();
    }
    connectionDB = connection;
}
```

Rys. 7 Funkcja odpowiedzialna za nawiązywanie połączenia.

W dalszych etapach aplikacja komunikuje się z bazą danych poprzez łączy JDBC. JDBC to specjalny interfejs programowania, który umożliwia aplikacjom napisanym w języku Java porozumiewać się z bazami danych za pomocą zapytań języka SQL. Dodano odpowiednie zależności (ang. *dependency*) w pliku *pom.xml*, aby możliwa była interakcja ze zdalną bazą danych. Zgodnie z założeniami nałożonymi przez prowadzącego, umożliwiono wybór dwóch perspektyw już na samym początku działania aplikacji w oknie głównym. Przejście do szerokiej perspektywy można uzyskać za pomocą wybrania opcji *Enter* (rys. 8) znajdującej się przy danych dyrektorach ogrodu zoologicznego. Wąska perspektywa jest dostępna po wprowadzeniu danych klienta, który zamierza uzyskać dostępne dla niego informacje o Zoo.



Rys. 8 Widok okna startowego aplikacji.

3.1 Wąska perspektywa - widok gościa w Zoo.

Na wąską perspektywę został wybrany widok odwiedzającego Zoo. Po wybraniu odpowiedniej opcji na samym początku w bazie jest dodawany nowy klient. Przy polu *Disability* należy wpisać jedną z dwóch wartości: T (tak) lub N (nie).

Enter as a guest

Write down your info

Name:

Surname:

Program:

Disability:

Save

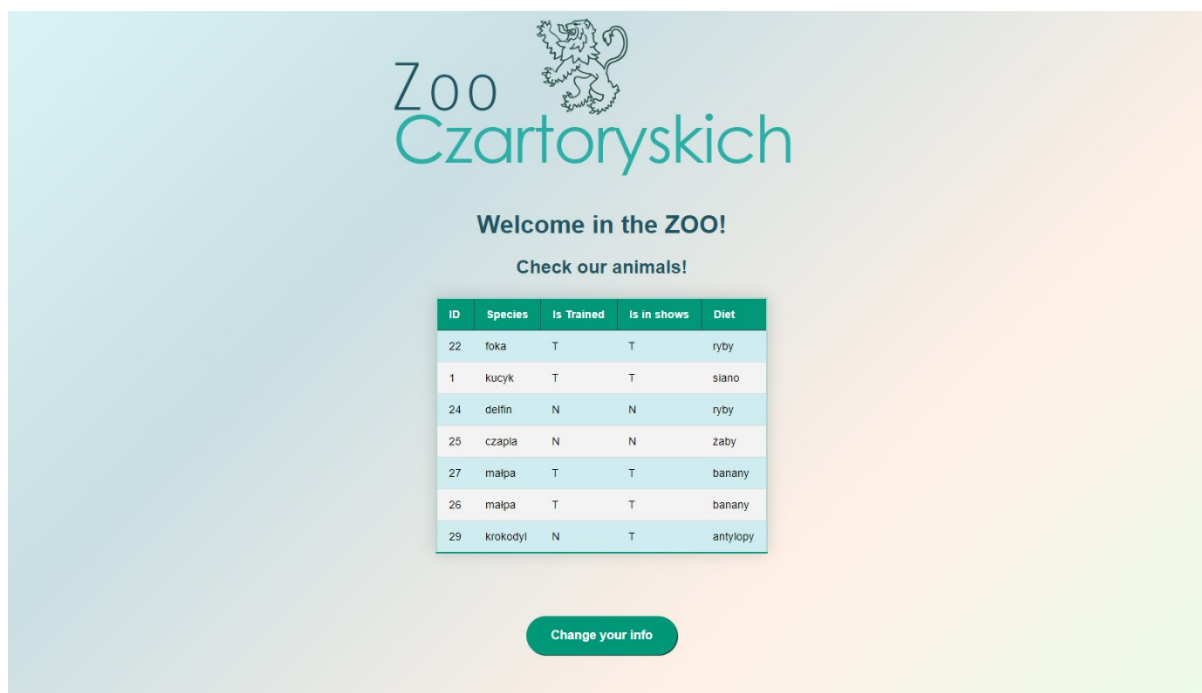
Rys. 9 Przykładowo uzupełniony formularz nowego klienta.

54	110	Wlodek	Marchewka	dżdżownice	N	21/01/27	21/01/27	1
55	114	Madzia	Niedowiarek	ocean	N	21/01/28	21/01/28	1

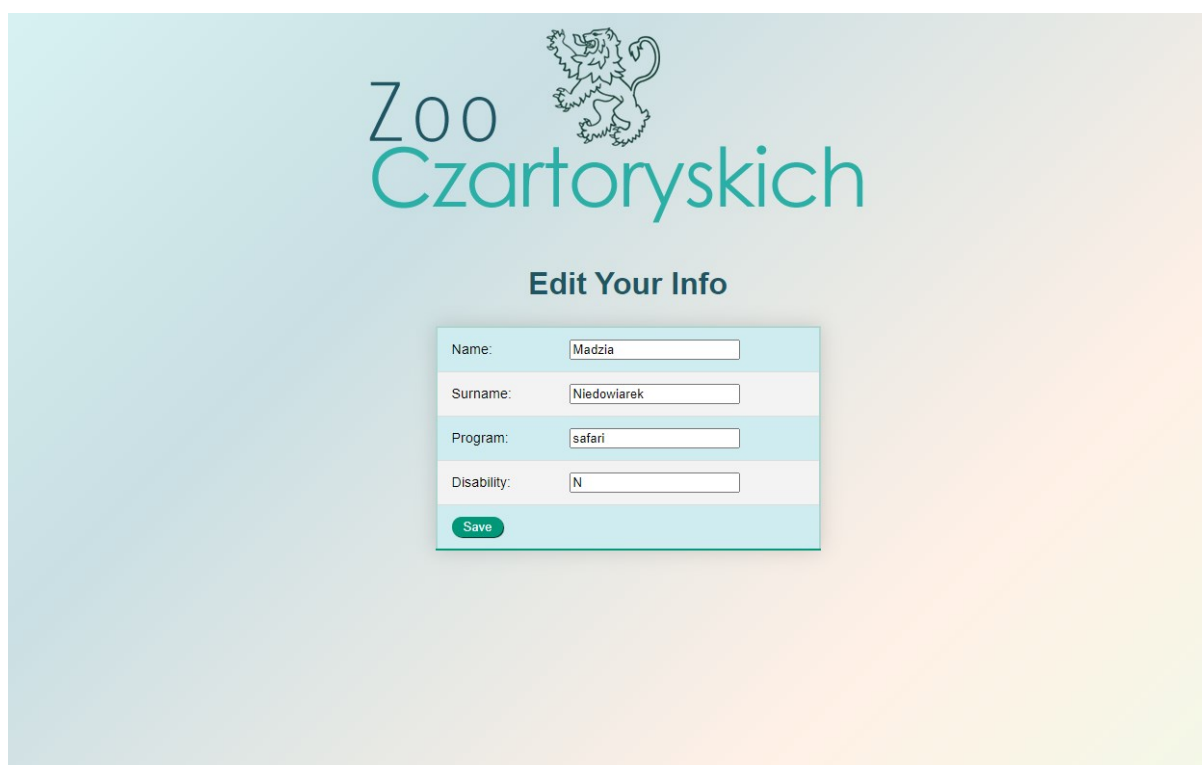
Rys. 10 Nowy klient w bazie danych – zrzut ekranu z SQL Developera.

Może on jedynie oglądać listę zwierząt, które są w Zoo oraz edytować swoje dane w specjalnym formularzu. Jego zmiany zostały ograniczone do imienia, nazwiska,

niepełnosprawności (w przypadku błędnego wprowadzenia) oraz programu wizyty ogrodu zoologicznego – inne atrybuty klienta, ze względu na swoją naturę, nie mogą być edytowane.



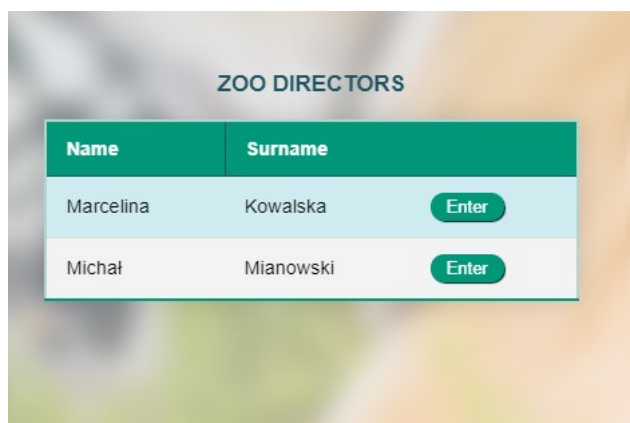
Rys. 11 Widok gościa.



Rys. 12 Formularz zmiany danych.

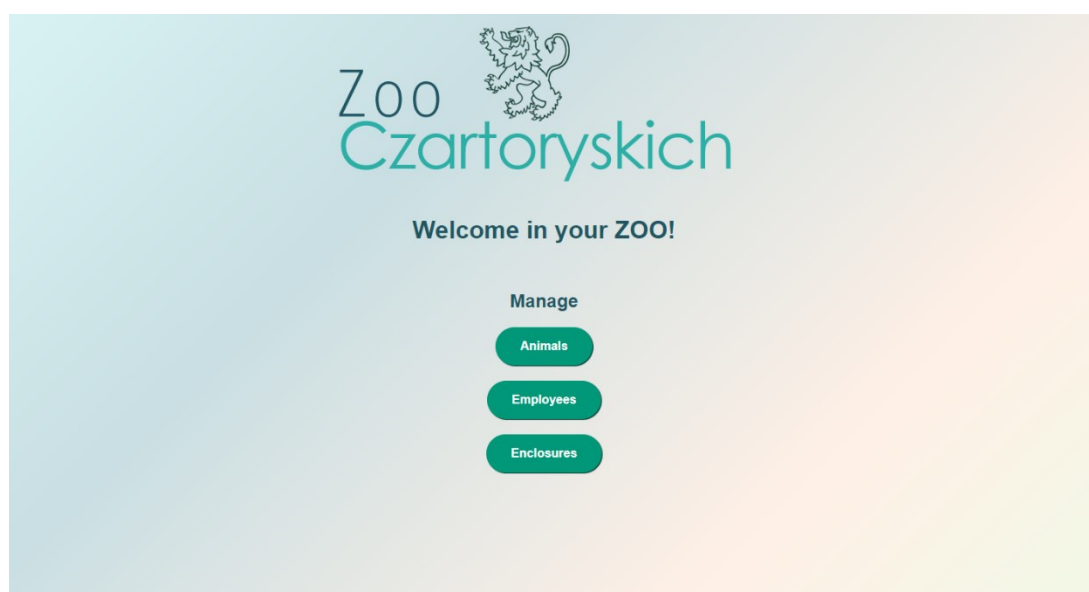
3.2 Szeroka perspektywa – widok dyrektora Zoo.

Na samym początku, w oknie startowym możemy wybrać jako który spośród dostępnych dyrektorów chcemy zarządzać ogrodem zoologicznym (patrz rys.13).




Rys. 13 Wybór dyrektora ZOO w oknie startowym.

Może on zarządzać trzema dostępnymi w swoim widoku relacjami – zwierzętami, pracownikami i wybiegami. Do każdej z nich może dodać nowy obiekt (rys. 15), zmodyfikować konkretny (rys. 16) lub usunąć istniejący (rys. 17 i 18). Z każdego widoku zarządzania można powrócić do głównego widoku dyrektora. Zastąpiono niektóre klucze obce występujące w danych relacjach właściwymi atrybutami z innych relacji, które są „słownikami” poszczególnych atrybutów. Dla przykładu, zamiast identyfikatora gatunku pobierana jest jego nazwa. W przypadku utworzenia nowego zwierzęcia konieczne jest dodanie do relacji GATUNKI nowego gatunku, jeśli on w niej nie istnieje. Tworząc zwierzę możemy określić czy jest wytresowane oraz czy bierze udział w pokazach (T- tak, N- nie). Podobnie jest w wybiegach – za pomocą wartości T(tak) lub N(nie) określamy czy wybieg jest otwarty dla zwiedzających oraz czy ma na swoim terenie atrakcje dla zwierząt (np. basen, małpi gaj itp.). W przypadku dodawania nowego wybiegu wartość numeru budynku nie może być mniejsza ani równa 0.




Rys. 14 Widok dyrektora.


Enter New Animal

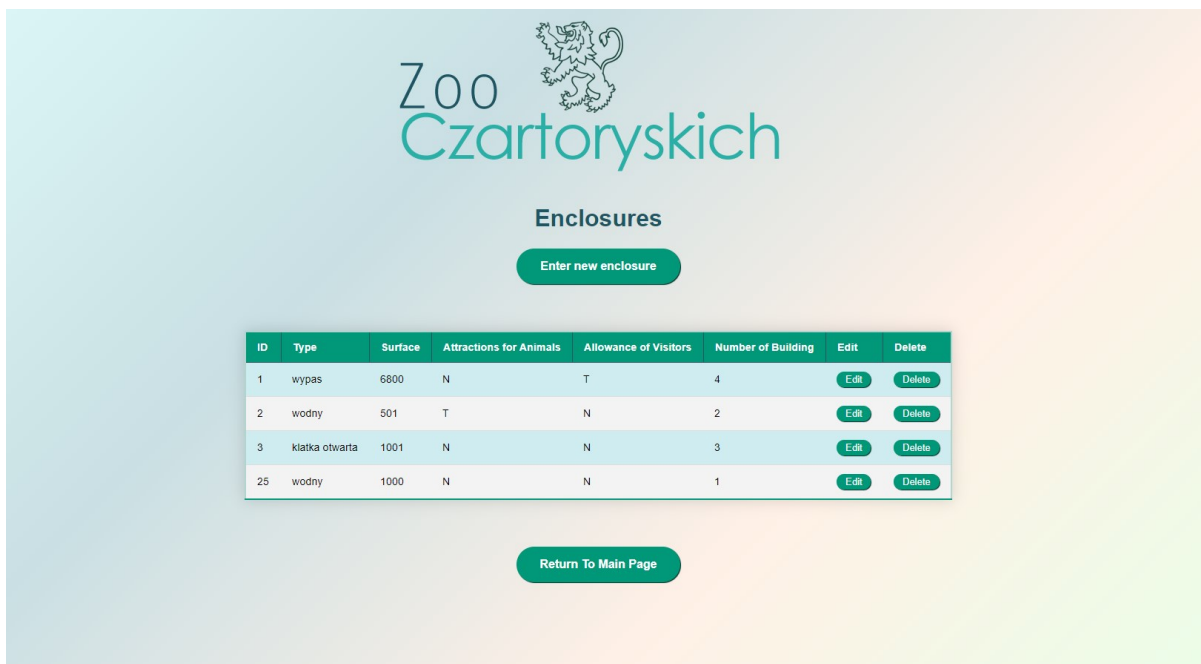
Species:	<input type="text" value="małpa"/>
Diet:	<input type="text" value="banany"/>
Is Trained:	<input type="text" value="T"/>
Is In Shows:	<input type="text" value="T"/>
<input type="button" value="Save"/>	

Rys. 15 Wprowadzanie danych dla nowego zwierzęcia.

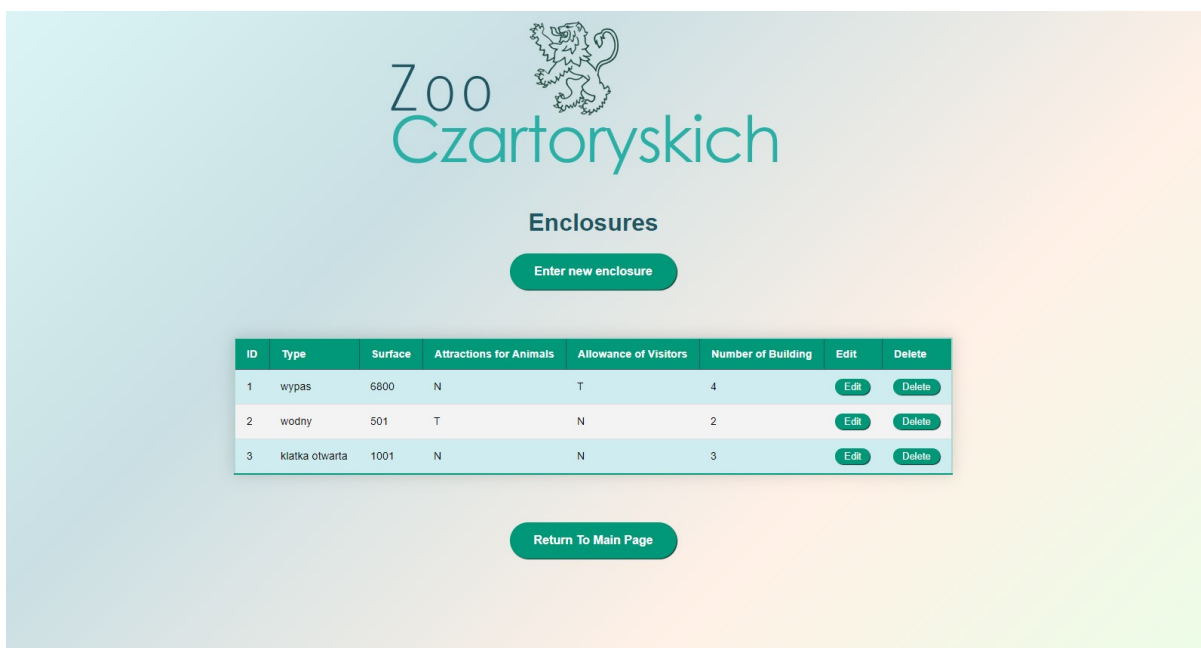

Edit Employee

ID:	1
Name:	<input type="text" value="Natalia"/>
Surname:	<input type="text" value="Nikulska"/>
Birth Date:	<input type="text" value="08.01.1993"/> <input type="button" value="📅"/>
Salary:	<input type="text" value="9010"/>
Bank Account Nr:	<input type="text" value="231466875321"/>
Phone Number:	<input type="text" value="321654987"/>
Mail:	<input type="text" value="nat@alia.co"/>
Employment Date:	<input type="text" value="27.01.2021"/> <input type="button" value="📅"/>
<input type="button" value="Save"/>	

Rys. 16 Edycja danych dotyczących danego pracownika.



Rys. 17 Widok tabeli przed usunięciem wybranego wybiegu



Rys. 18 Widok tabeli po usunięciu wybranego wybiegu.

Udało się wygenerować odpowiedni plik typu jar (plik `zoomanager-0.0.1-SNAPSHOT.jar`). Po wykonaniu napisanej w wierszu poleceń komendy `java -jar zoomanager-0.0.1-SNAPSHOT.jar` uruchamia się serwer TomCat. Po zainicjalizowaniu serwera aplikacja zostaje uruchomiona i gotowa do pracy z użytkownikiem. Dostęp do przeglądarkowej aplikacji jest możliwy poprzez wpisanie w przeglądarce internetowej `localhost:8080`.

4. Bibliografia.

1. Wykłady udostępnione w ramach przedmiotu BDBT.
2. <https://edu.pjwstk.edu.pl/wyklady/mpr/scb/W7/W7.htm>
3. <https://bykowski.pl/8-powodow-dla-ktorych-warto-uczyc-sie-spring-framework/>
4. https://pl.wikipedia.org/wiki/Apache_Maven