Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

|  |  |
| --- | --- |
|  | Номер зачетной книжки 6510115 |
|  | Производственная практика зачтена с оценкой  \_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)  (цифрой) (прописью) |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись руководителя практики от БГУИР)  \_\_\_\_.\_\_\_\_\_.2019 |

**ОТЧЕТ**

**по производственной практике**

Место прохождения практики: ООО «Софтарекс Текнолоджиес»

Сроки прохождения практики: с 08.06.2019 по 02.07.2019

|  |  |  |
| --- | --- | --- |
| Руководитель практики от предприятия:  \_\_\_\_\_\_\_\_\_\_\_\_ Н.И.Мельник  (подпись руководителя)  М.П. |  | Студент группы 651005 В.А.Орол  (подпись студента)  Руководитель практики от БГУИР  С.В.Болтак − ассистент кафедры ПОИТ |

Минск 2019

Содержание

[**ВВЕДЕНИЕ** 3](#_Toc12836665)

[**1 ОЗНАКОМЛЕНИЕ С ПРЕДПРИЯТИЕМ** 4](#_Toc12836666)

[**1.1** **Здравоохранение** 4](#_Toc12836667)

[**1.2** **Финансовые технологии** 5](#_Toc12836668)

[**1.3** **Энергетика и коммунальное хозяйство** 5](#_Toc12836669)

[**1.4** **Ресторанная и гостиничная бизнес, индустрия гостеприимства** 5](#_Toc12836670)

[**1.5** **Наука о данных и компьютерное зрение** 6](#_Toc12836671)

[**1.6** **Машинное обучение** 6](#_Toc12836672)

[**2 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ** 8](#_Toc12836673)

[**2.1 Техническое задание** 8](#_Toc12836674)

[**2.2 Распознавание образов с помощью искусственных нейронных сетей** 8](#_Toc12836675)

[2.2.1 Области применения систем распознавания 11](#_Toc12836676)

[**2.3 Разработка задания к лабораторной работе** 13](#_Toc12836677)

[**2.4 Ход выполнения лабораторной работы** 14](#_Toc12836678)

[**2.5 Разработка методических указаний к лабораторной работе** 18](#_Toc12836679)

[2.5.1 Методические указания к выполнению лабораторной работы по теме «Распознавание образов с помощью персептрона» 18](#_Toc12836680)

[**ЗАКЛЮЧЕНИЕ** 24](#_Toc12836681)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 25](#_Toc12836682)

[**ПРИЛОЖЕНИЕ** 26](#_Toc12836683)

# **ВВЕДЕНИЕ**

Производственная практика является частью учебного процесса, связанная с опытом реальной работы на том или ином профильном предприятии.

Цель производственной практики − овладение формами и методами работы специалиста, приобретение навыков и профессиональных знаний, необходимых для работы по специальности.

В ходе распределения местом прохождения практики было выбрано ООО «Софтарекс Текнолоджиес».

По прибытию на практику были сформулированы следующие задачи:

* получение общих сведений о предприятии;
* изучение нормативной документации;
* изучение функциональных обязанностей по должности прохождения практики;
* изучение структуры организации;
* исследование видов программных продуктов, используемых на предприятии;
* получение и выполнение индивидуального задания, при использовании выбранной технологии.

# **1 ОЗНАКОМЛЕНИЕ С ПРЕДПРИЯТИЕМ**

ООО «Софтарекс Текнолоджиес», являющаяся центром разработки программного обеспечения компании Softarex Technologies Inc. (США), была официально зарегистрирована в 2016 году. Головной офис находится в Александрии, штат Вирджиния. На сегодняшний день в минском офисе компании усердно работают над увлекательными проектами более сорока сотрудников.

Авторитетное в сфере разработки программного обеспечения и информационных технологий издание Software Magazine в 2016 включило «Софтарекс Технолоджиес» в Software 500 − рейтинг крупнейших мировых провайдеров услуг в сфере информационных технологий и разработки программного обеспечения.

Богатый опыт сотрудников компании дает уникальную возможность для разработки пользовательского программного обеспечения. Высокоэффективная методология работы приводит к минимальным недочетам на всех этапах разработки и внедрения системы, что, в свою очередь, положительно влияет на степень удовлетворенности клиентов. «Софтарекс Текнолоджиес» разрабатывает программное обеспечение для множества отраслей.

Компания является резидентом Парка высоких технологий. Парк высоких технологий является одним из крупнейших ИТ-кластеров в Центральной и Восточной Европе. Уникальность ПВТ заключается в удачном сочетании качественного технического образования, высокого уровня профессионализма ИТ-специалистов и государственной поддержки ИТ-отрасли.

## **Здравоохранение**

В «Софтарекс Текнолоджиес» работают сотрудники с сертификатами HIPAA, которые имеют решающее значение для работы в этой сфере. HIPAA или Health Insurance Portability and Accountability Act − акт о мобильности и подотчётности медицинского страхования, который был принят 21 августа 1996 прежде всего, чтобы модернизировать поток медицинской информации, предсказать, как личная информация, хранящаяся в медицинских учреждениях и медицинских страховых отраслях, должна быть защищена от мошенничества и краж, а также обращаться к ограничениям на медицинское страхование.

«Софтарекс Технолоджиес» разрабатывает и интегрирует различные EMR, EHR, HIS, RIS и другие информационные системы, наравне с решениями, соответствующими медицинскому стандарту HL7. Седьмой уровень поддерживает выполнение таких задач как:

* структурирование передаваемых данных;
* возможности проектирования систем;
* достижение согласованности передач;
* безопасность;
* идентификация участников;
* доступность.

Для развития в данной области предприятие также проводит совместные научные исследования с крупнейшими вузами, в том числе крупнейшим калифорнийским университетом

## **Финансовые технологии**

Fintech является новой финансовой отраслью, которая применяет технологии для улучшения финансовой деятельности». Ирэн Олдридж и Стив Кравцив отмечают несколько областей распространения финтеха, например, автоматизацию страхования, торговли и управления рисками.

На данный момент в «Софтарекс Текнолоджиес» присутствует множество специалистов, имеющих более чем пятнадцатилетний опыт работы в сфере финансов, благодаря чему были успешно разработаны несколько собственных продуктов в данной области, в том числе финансовый чатбот, помогающий клиентам принимать грамотные решения в области личных финансов.

## **Энергетика и коммунальное хозяйство**

Энергетическая промышленность представляет собой совокупность всех отраслей, участвующих в производстве и продаже энергии, в том числе топлива добычи, производства, переработки и распределения.

«Софтарекс Текнолоджиес» активно занимается разработкой различных систем для энергетической промышленности и коммунальных услуг. Среди них более ста датчиков, отвечающих за мониторинг параметров воды, отопления, газа, электричества и т. д. Системы, разработанные «Софтарекс Текнолоджиес», экономят до 25% электроэнергии, до 25% увеличивают срок службы электроламп и на 20% сокращают время на регулировку искусственного освещения помещений.

## **Ресторанная и гостиничная бизнес, индустрия гостеприимства**

Индустрия гостеприимства – это развитые технологии быстрой обработки и подачи заказа, предоставление слаженных технических моментов в обслуживании клиента (доступ в Интернет, качественная телефонная связь, доступные междугородние и международные тарифы), предупреждение желаний клиента.

В «Софтарекс Текнолоджиес» было создано несколько систем, упрощающих и модернизирующих ресторанную и гостиничную индустрии. Среди них онлайн-платформа, регулирующая поток клиентов и помогающая в бронировании столов в общественных местах, многокомпонентная система для одной из крупнейших сетей ресторанов быстрого питания в США, вклющая в себя систему распознавания изображений, серверную часть и приложение для iPad.

## **Наука о данных и компьютерное зрение**

Наука о данных − раздел информатики, изучающий проблемы анализа, обработки и представления данных в цифровой форме. Объединяет методы по обработке данных в условиях больших объёмов и высокого уровня параллелизма, статистические методы, методы интеллектуального анализа данных и приложения искусственного интеллекта для работы с данными, а также методы проектирования и разработки баз данных. Компьютерное зрение (иначе техническое зрение) − теория и технология создания машин, которые могут производить обнаружение, отслеживание и классификацию объектов.

За спинами сотрудников «Софтарекс Текнолоджиес» имеется более сотни опубликованных научных статей в области компьютерного зрения. На протяжении многих лет накапливался опыт в реализации алгоритмов компьютерного зрения в реальных приложениях. Разработанные компанией решения нашли применение в сетях ресторанов, спортивной аналитике, здравоохранении, промышленности и многих других областях.

## **Машинное обучение**

Машинное обучение − это научное исследование алгоритмов и статистических моделей, которые используются компьютерными системами для эффективного выполнения конкретной задачи без использования явных инструкций, вместо этого опираясь на шаблоны и умозаключения. Рассматривается как подмножество искусственного интеллекта.

В «Софтарекс Текнолоджиес» было разработано множество решений для прогнозирования состояния пациентов и их потребностей в лечении и госпитализации в будущем. Были разработаны алгоритмы на основе искусственного интеллекта, предназначенные для адаптивных изменений в сценариях обработки данных.

Одним из интереснейших достижений компании в области машинного обучения является разработанное специально для бейсбольных скаутов десктопное Windows-приложение, позволяющее автоматизировать процесс анализа игры сборной США по бейсболу.

# **2 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**2.1 Техническое задание**

2.1.1 Требования технологического стека

1. Java 8+
2. Maven or Gradle
3. Spring boot + Spring mvc
4. JPA, Spring data
5. DB: PostgreSQL
6. Vanilla JS or Jquery (optional ReactJS)
7. Twitter Bootstrap
8. WebSockets
9. Source code to public git repo (github.com)
10. IDE: IDEA

2.1.2 Технические детали

Разработать приложение, предназначенное для сбора данных от пользователей. Форма для сбора данных должна иметь общее количество полей.

Поля являются записями из базы данных. Поле имеет следующие поля:

1. Label – строка.

2. Type - перечисление (однострочный текст, многострочный текст, переключатель, флажок, поле со списком, дата).

3. Required - boolean (ответ на это поле будет обязательным).

4. Is Active - boolean (пользователь делает поле неактивным по любой причине, оно удалит поле из формы).

2.1.3 Оформление

Приложение должно состоять из 9 страниц:

1. Страница входа.

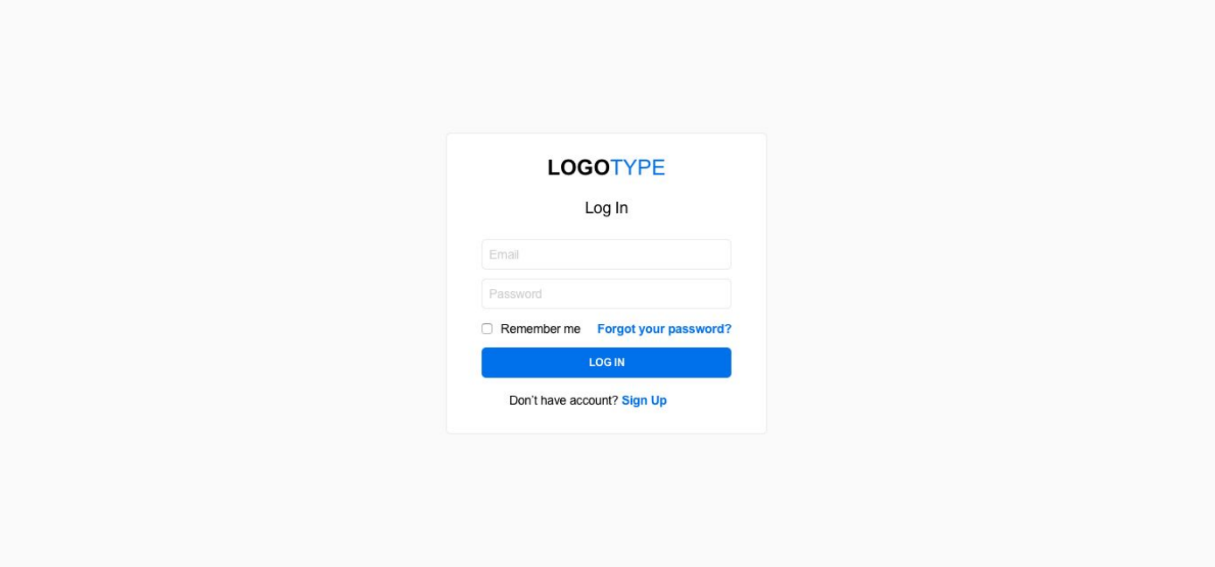


Рисунок 2.1 – Пример страницы № 1

2. Страница регистрации (уведомление по электронной почте после регистрации требуется обязательно).

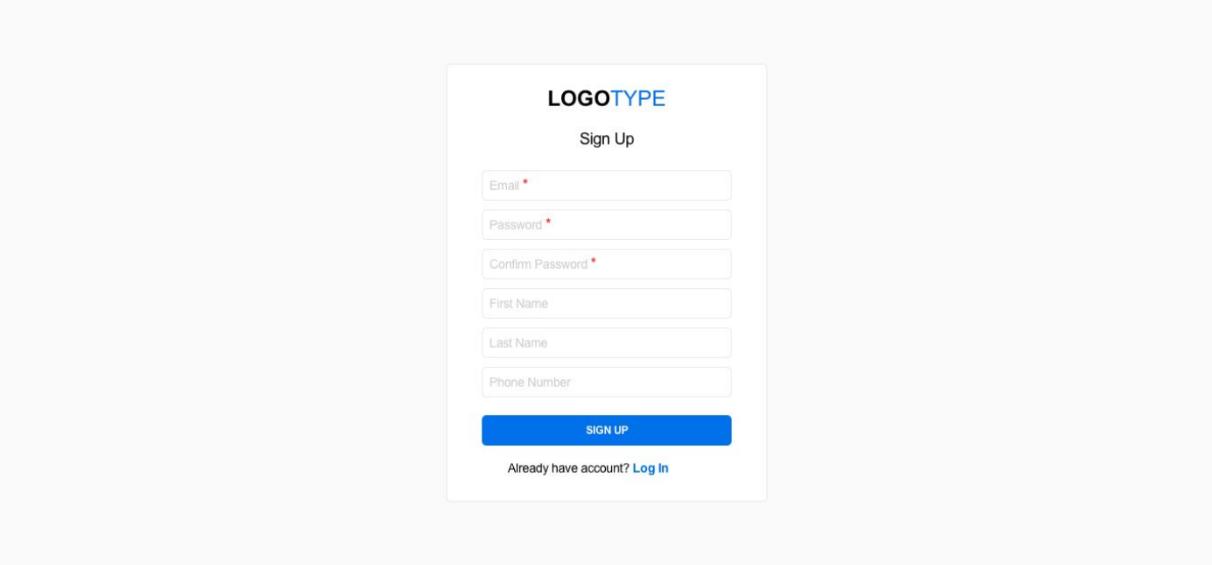


Рисунок 2.2 – Пример страницы № 2

3. Редактировать страницу профиля.

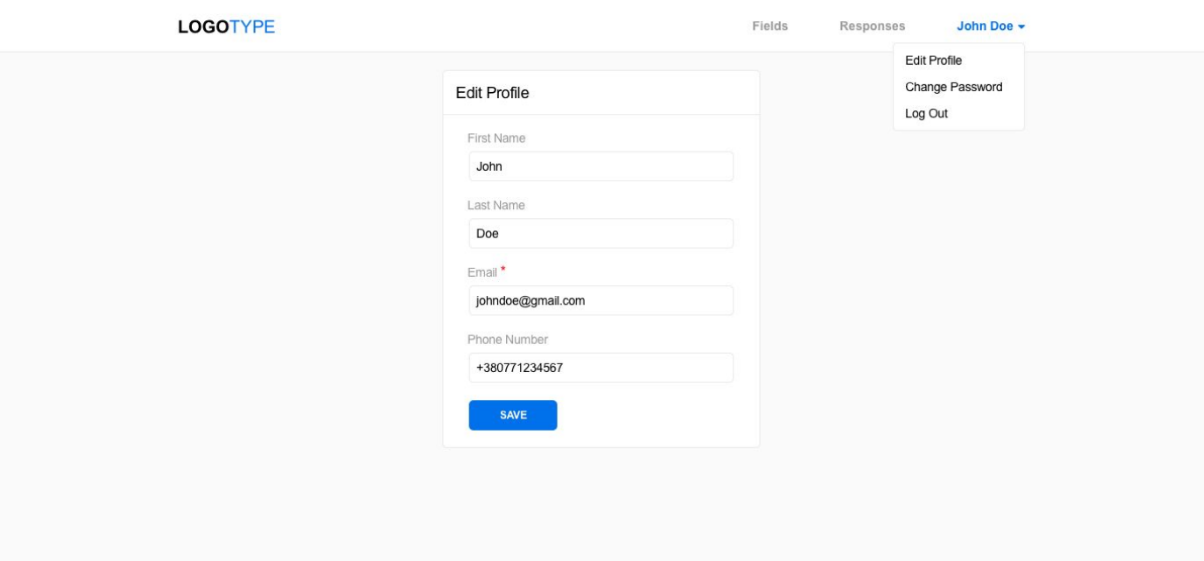


Рисунок 2.3 – Пример страницы № 3

4. Смена пароля на странице (требуется уведомление по электронной почте после изменения).

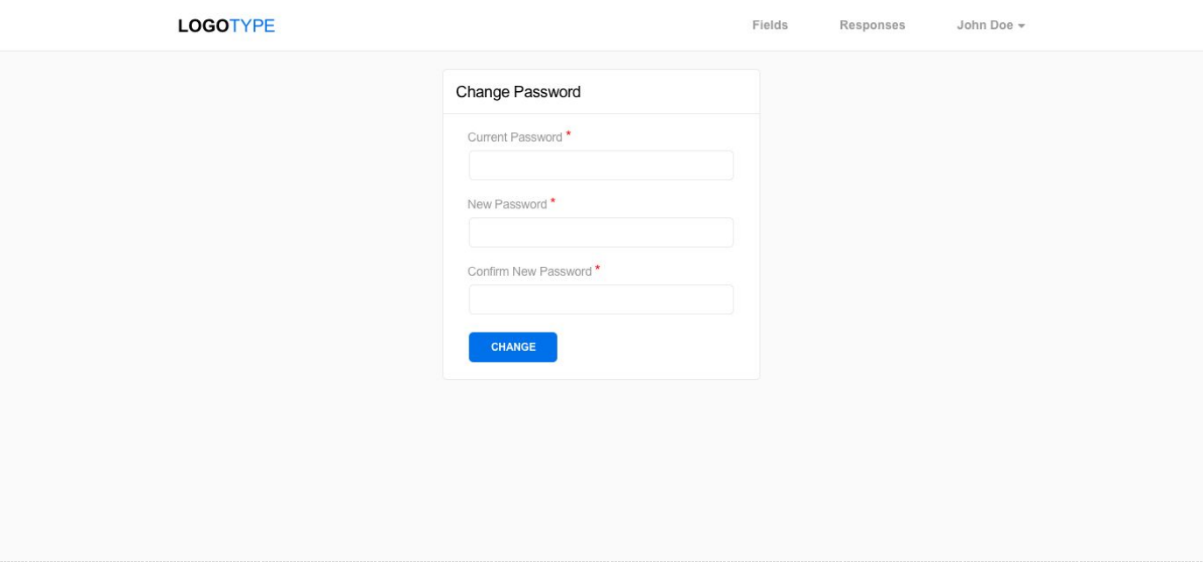


Рисунок 2.4 – Пример страницы № 4

5. Страница списка полей. На данной странице будут перечислены все поля, хранящиеся в базе данных. Пользователь сможет создавать, редактировать, удалять поля. Страница должна быть доступным по пути domainname.com/fields.

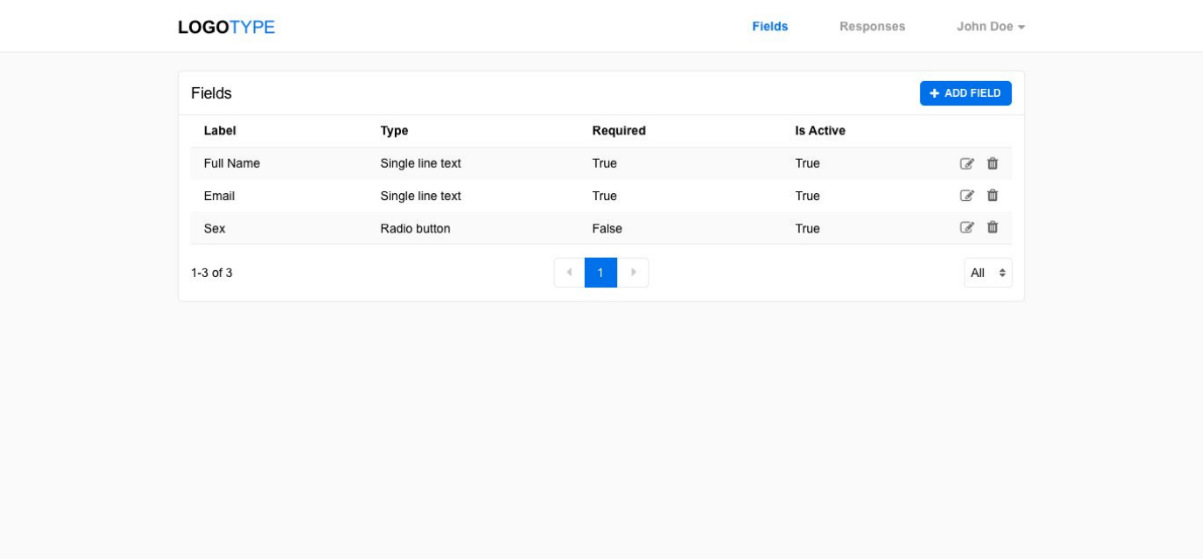


Рисунок 2.5 – Пример страницы № 5

6. Диалог добавления или редактирования поля. На данной странице пользователь сможет создавать новые поля с определенным количеством свойств. Все свойства обязательны.

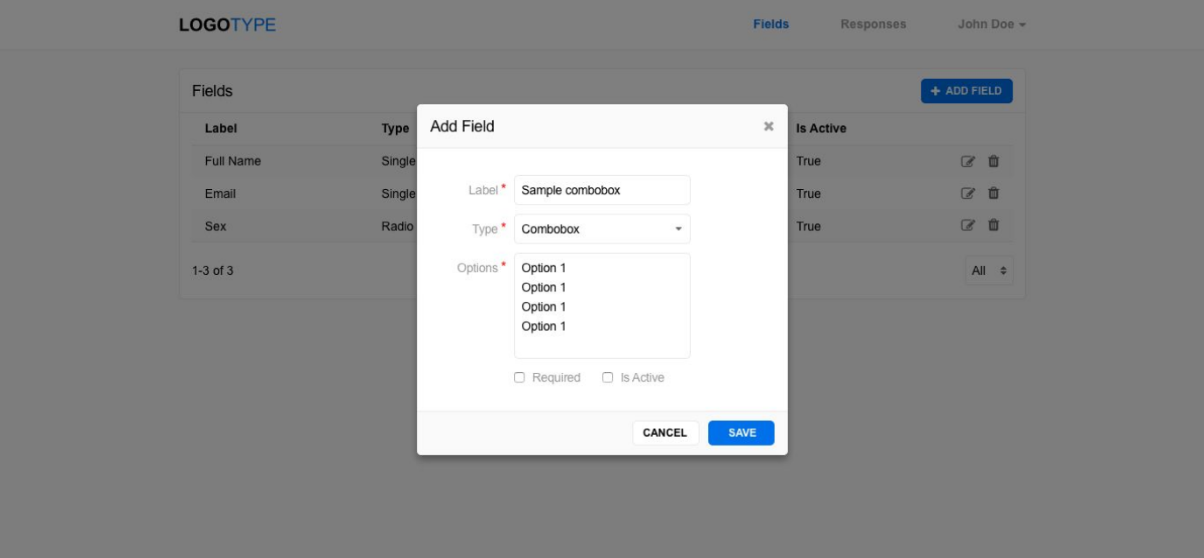


Рисунок 2.6 – Пример страницы № 6

7. Страница анкеты. Она даст возможность заполнить все активные поля, созданные на странице № 5. Не обязательные поля можно оставить пустыми. Форма сброса очистит все введенные данные (кнопка сброса отсутствует на макете, но должна быть реализована). После отправки пользователь будет перенаправлен на страницу поздравления (№ 8).

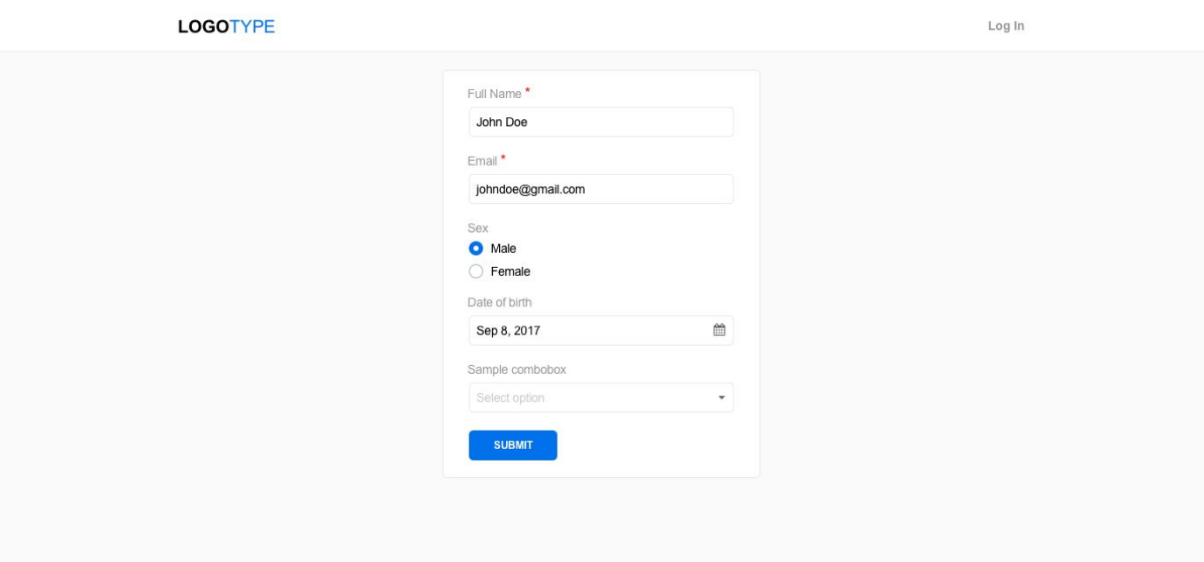


Рисунок 2.7 – Пример страницы № 7

8. Успешная подача ответа.

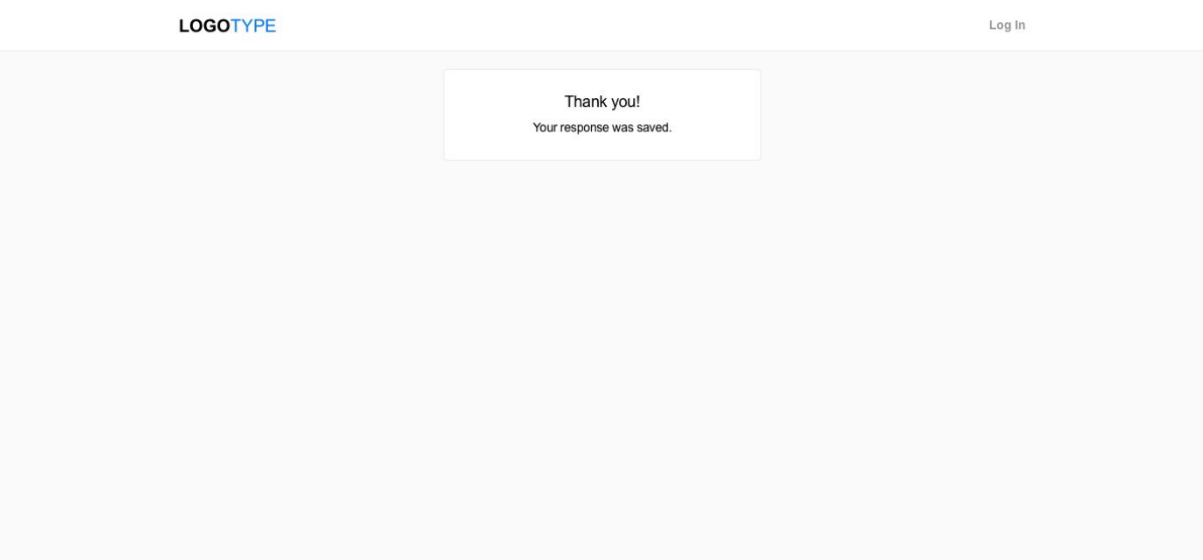


Рисунок 2.8 – Пример страницы № 8

9. Страница списка ответов. На ней можно получить все ответы, заполненные пользователями. Они должны попадать в этот список автоматически после отправки. Для этого будут использоваться веб-сокеты: количество ответов в меню должно обновляться каждый раз, когда кто-то представляет данные. Столбцы для этого соответствуют каждому полю, созданному на странице № 5. Количество столбцов не меняется динамически, пока пользователь находится на странице, и меняется, когда пользователь обновляет страницу. Страница должна быть доступна по пути domainname.com/responses

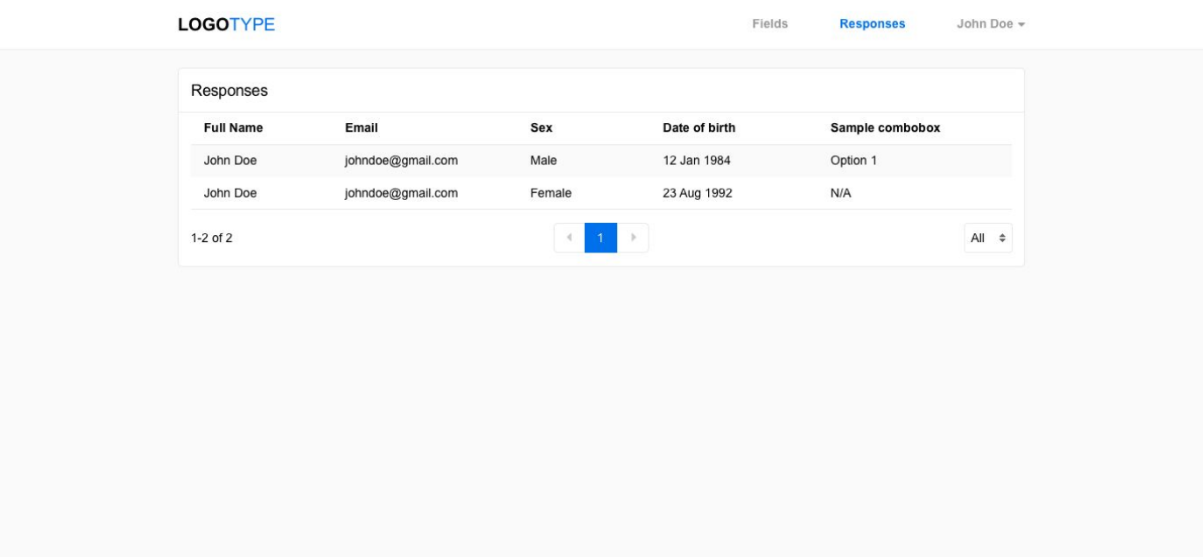


Рисунок 2.9 – Пример страницы № 9

**2.2 Анализ ресурсов и разработка приложения**

2.2.1 Gradle Build Tool

Gradle – система автоматической сборки, построенная на принципах Apache Ant и Apache Maven, но предоставляющая DSL на языках Groovy и Kotlin вместо традиционной XML-образной формы представления конфигурации проекта. В отличие от Apache Maven, основанного на концепции жизненного цикла проекта, и Apache Ant, в котором порядок выполнения задач (targets) определяется отношениями зависимости (depends-on), Gradle использует направленный ациклический граф для определения порядка выполнения задач. Gradle был разработан для расширяемых многопроектных сборок, и поддерживает инкрементальные сборки, определяя, какие компоненты дерева сборки не изменились и какие задачи, зависимые от этих частей, не требуют перезапуска.

В моём проекте используется стандартная структура каталогов Maven для исходных кодов и ресурсов. Такая структура включает в себя следующие каталоги:

* src/main/java,
* src/main/resources,
* src/test/java,
* src/test/resources.

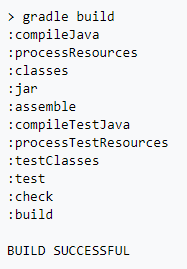


Рисунок 2.10 – Результат выполнения команды «gradle build»

2.2.2 Spring Boot

Spring Framework — универсальный фреймворк с открытым исходным кодом для Java-платформы. Spring Boot — это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию код.

Spring Boot обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов. Чтобы ускорить процесс управления зависимостями, Spring Boot неявно упаковывает необходимые сторонние зависимости для каждого типа приложения на основе Spring и предоставляет их разработчику посредством так называемых starter-пакетов (spring-boot-starter-web, spring-boot-starter-data-jpa и т.д.)

Starter-пакеты представляют собой набор удобных дескрипторов зависимостей, которые можно включить в свое приложение. Это позволяет получить универсальное решение для всех, связанных со Spring технологий, избавляя разработчика от лишнего поиска примеров кода и загрузки из них требуемых дескрипторов зависимостей.

Каждое Spring Boot web-приложение включает встроенный web-сервер.

Для создания веб-приложения при помози Spring Boot я изначально настроил его в файле pom.xml. Так как все приложения Spring Boot конфигурируются от spring-boot-starter-parent, перед дальнейшим определением зависимостей необходимо добавить starter-parent.

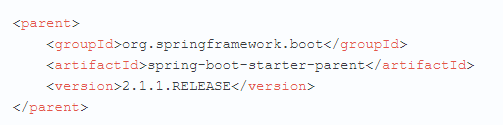


Рисунок 2.11 – Добавление starter-parent



Рисунок 2.12 – Добавление Spring Boot плагина

2.2.3 PostgreSQL

PostgreSQL – свободная объектно-реляционная система управления базами данных. Согласно результатам автоматизированного исследования различного ПО на предмет ошибок, проведённом в 2005 году, в исходном коде PostgreSQL было найдено 20 проблемных мест на 775 000 строк исходного кода (в среднем, одна ошибка на 39 000 строк кода). Для сравнения: MySQL – 97 проблем, одна ошибка на 8 000 строк кода; FreeBSD (целиком) – 306 проблем, одна ошибка на 2 500 строк кода; Linux (только ядро) – 950 проблем, одна ошибка на 800 строк кода.

PostgreSQL не просто реляционная, а объектно-реляционная СУБД. Это даёт ему некоторые преимущества над другими SQL базами данных с открытым исходным кодом, такими как MySQL, MariaDB и Firebird. Фундаментальная характеристика объектно-реляционной базы данных – это поддержка пользовательских объектов и их поведения, включая типы данных, функции, операции, домены и индексы. Это делает Постгрес невероятно гибким и надежным. Среди прочего, он умеет создавать, хранить и извлекать сложные структуры данных.

Поддержка JSON в PostgreSQL позволяет вам перейти к хранению schema-less данных в SQL базе данных. Это может быть полезно, когда структура данных требует определённой гибкости: например, если в процессе разработки структура всё ещё меняется или неизвестно, какие поля будет содержать объект данных.

# **ЗАКЛЮЧЕНИЕ**

В результате проделанной во время производственной практики работы была изучена структура предприятия и его документооборот, углублены и закреплены знания, полученные при изучении специальных дисциплин, а также разработано программное средство, отвечающее поставленным требованиям. Были дополнены и закреплены навыки в разработке веб-приложений на языке Java, углублены знания в разработке и тестировании программного обеспечения.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

[1] Сайт БГУИР [Электронный ресурс]. Режим доступа: <https://www.bsuir.by/ru/kaf-poit>

[2] Сайт Парк высоких технологий [Электронный ресурс]. Режим доступа: <http://www.park.by/>

[3] Habrahabr [Электронный ресурс]. Режим доступа: <https://habrahabr.ru/post/144881>

[4] Википедия [Электронный ресурс]. Режим доступа: <https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD>

**ПРИЛОЖЕНИЕ**

Исходный код программы

package com.epam.esm.giftcertificates.entity;

import javax.persistence.Column;

import javax.persistence.Embeddable;

import java.io.Serializable;

import java.util.Objects;

/\*\*

\* This class stores values of the certificate and purchase ids.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@Embeddable

public class CompositeId implements Serializable {

@Column(name = "gift\_certificate\_id", nullable = false)

private Long certificateId;

@Column(name = "purchase\_id", nullable = false)

private Long purchaseId;

/\*\*

\* {@inheritDoc}

\*/

@Override

public String toString() {

return "CompositeId{" +

"certificateId=" + certificateId +

", purchaseId=" + purchaseId +

'}';

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public boolean equals(final Object object) {

boolean result;

CompositeId compositeId;

if (object != null) {

compositeId = (CompositeId) object;

if (this == object) {

result = true;

} else if (getClass() != object.getClass()) {

result = false;

} else {

result = Objects.equals(certificateId, compositeId.certificateId) &&

Objects.equals(purchaseId, compositeId.purchaseId);

}

} else {

result = false;

}

return result;

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public int hashCode() {

return Objects.hash(certificateId, purchaseId);

}

public Long getCertificateId() {

return certificateId;

}

public void setCertificateId(final Long certificateId) {

this.certificateId = certificateId;

}

public Long getPurchaseId() {

return purchaseId;

}

public void setPurchaseId(final Long purchaseId) {

this.purchaseId = purchaseId;

}

}

package com.epam.esm.giftcertificates.entity;

import javax.persistence.\*;

import java.math.BigDecimal;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

/\*\*

\* This entity class stores info about gift certificate.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@Entity

@Table(name = "gift\_certificate")

public class GiftCertificate {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name", length = 100, nullable = false)

private String name;

@Column(name = "description", nullable = false)

private String description;

@Column(name = "price", nullable = false)

private BigDecimal price;

@Column(name = "date\_of\_creation", nullable = false)

private LocalDateTime dateOfCreation;

@Column(name = "date\_of\_modification", nullable = false)

private LocalDateTime dateOfModification;

@Column(name = "duration\_in\_days", nullable = false)

private Integer durationInDays;

@Column(name = "active", nullable = false)

private Boolean active;

@ManyToMany(fetch = FetchType.LAZY)

@JoinTable(name = "gift\_certificate\_m2m\_tag",

joinColumns = @JoinColumn(name = "gift\_certificate\_id", nullable = false),

inverseJoinColumns = @JoinColumn(name = "tag\_id", nullable = false))

private List<Tag> tags;

public GiftCertificate() {

tags = new ArrayList<>();

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public String toString() {

return "GiftCertificate{" +

"id=" + id +

", name='" + name + '\'' +

", description='" + description + '\'' +

", price=" + price +

", dateOfCreation=" + dateOfCreation +

", dateOfModification=" + dateOfModification +

", durationInDays=" + durationInDays +

", active=" + active +

'}';

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public boolean equals(final Object object) {

boolean result;

GiftCertificate giftCertificate;

if (object != null) {

giftCertificate = (GiftCertificate) object;

if (this == object) {

result = true;

} else if (getClass() != object.getClass()) {

result = false;

} else {

result = Objects.equals(id, giftCertificate.id) &&

Objects.equals(name, giftCertificate.name) &&

Objects.equals(description, giftCertificate.description) &&

Objects.equals(price, giftCertificate.price) &&

Objects.equals(dateOfCreation, giftCertificate.dateOfCreation) &&

Objects.equals(dateOfModification, giftCertificate.dateOfModification) &&

Objects.equals(durationInDays, giftCertificate.durationInDays) &&

Objects.equals(active, giftCertificate.active);

}

} else {

result = false;

}

return result;

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public int hashCode() {

return Objects.hash(id, name, description, price, dateOfCreation,

dateOfModification, durationInDays, active);

}

public Long getId() {

return id;

}

public void setId(final Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(final String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(final String description) {

this.description = description;

}

public BigDecimal getPrice() {

return price;

}

public void setPrice(final BigDecimal price) {

this.price = price;

}

public LocalDateTime getDateOfCreation() {

return dateOfCreation;

}

public void setDateOfCreation(final LocalDateTime dateOfCreation) {

this.dateOfCreation = dateOfCreation;

}

public LocalDateTime getDateOfModification() {

return dateOfModification;

}

public void setDateOfModification(final LocalDateTime dateOfModification) {

this.dateOfModification = dateOfModification;

}

public Integer getDurationInDays() {

return durationInDays;

}

public void setDurationInDays(final Integer durationInDays) {

this.durationInDays = durationInDays;

}

public Boolean isActive() {

return active;

}

public void setActive(final Boolean active) {

this.active = active;

}

public List<Tag> getTags() {

return tags;

}

public void setTags(final List<Tag> tags) {

this.tags = tags;

}

}

package com.epam.esm.giftcertificates.entity;

import javax.persistence.\*;

import java.math.BigDecimal;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

/\*\*

\* This entity class stores info about purchase.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@Entity

@Table(name = "purchase")

public class Purchase {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "cost", nullable = false)

private BigDecimal cost;

@Column(name = "timestamp", nullable = false)

private LocalDateTime timestamp;

@OneToMany(mappedBy = "purchase", fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private List<PurchaseCertificate> purchaseCertificates;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "certificate\_user\_id", nullable = false)

private User user;

public Purchase() {

purchaseCertificates = new ArrayList<>();

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public String toString() {

return "Purchase{" +

"id=" + id +

", cost=" + cost +

", timestamp=" + timestamp +

", user=" + user +

'}';

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public boolean equals(final Object object) {

boolean result;

Purchase purchase;

if (object != null) {

purchase = (Purchase) object;

if (this == object) {

result = true;

} else if (getClass() != object.getClass()) {

result = false;

} else {

result = Objects.equals(id, purchase.id) &&

Objects.equals(cost, purchase.cost) &&

Objects.equals(timestamp, purchase.timestamp) &&

Objects.equals(user, purchase.user);

}

} else {

result = false;

}

return result;

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public int hashCode() {

return Objects.hash(id, cost, timestamp, user);

}

public Long getId() {

return id;

}

public void setId(final Long id) {

this.id = id;

}

public BigDecimal getCost() {

return cost;

}

public void setCost(final BigDecimal cost) {

this.cost = cost;

}

public LocalDateTime getTimestamp() {

return timestamp;

}

public void setTimestamp(final LocalDateTime timestamp) {

this.timestamp = timestamp;

}

public List<PurchaseCertificate> getPurchaseCertificates() {

return purchaseCertificates;

}

public void setPurchaseCertificates(final List<PurchaseCertificate> purchaseCertificates) {

this.purchaseCertificates = purchaseCertificates;

}

public User getUser() {

return user;

}

public void setUser(final User user) {

this.user = user;

}

}

package com.epam.esm.giftcertificates.entity;

import javax.persistence.\*;

import java.math.BigDecimal;

import java.util.Objects;

/\*\*

\* Entity class for many to many table.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@Entity

@Table(name = "purchase\_m2m\_gift\_certificate")

public class PurchaseCertificate {

@EmbeddedId

private CompositeId compositeId;

@Column(name = "purchase\_price", nullable = false)

private BigDecimal purchasePrice;

@Column(name = "count", nullable = false)

private Long count;

@ManyToOne

@MapsId("purchase\_id")

private Purchase purchase;

@ManyToOne

@MapsId("gift\_certificate\_id")

private GiftCertificate giftCertificate;

/\*\*

\* {@inheritDoc}

\*/

@Override

public String toString() {

return "PurchaseCertificate{" +

"compositeId=" + compositeId +

", purchasePrice=" + purchasePrice +

", count=" + count +

'}';

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public boolean equals(final Object object) {

boolean result;

PurchaseCertificate purchaseCertificate;

if (object != null) {

purchaseCertificate = (PurchaseCertificate) object;

if (this == object) {

result = true;

} else if (getClass() != object.getClass()) {

result = false;

} else {

result = Objects.equals(compositeId, purchaseCertificate.compositeId) &&

Objects.equals(purchasePrice, purchaseCertificate.purchasePrice) &&

Objects.equals(count, purchaseCertificate.count);

}

} else {

result = false;

}

return result;

}

@Override

public int hashCode() {

return Objects.hash(compositeId, purchasePrice, count);

}

public CompositeId getCompositeId() {

return compositeId;

}

public void setCompositeId(final CompositeId compositeId) {

this.compositeId = compositeId;

}

public BigDecimal getPurchasePrice() {

return purchasePrice;

}

public void setPurchasePrice(final BigDecimal purchasePrice) {

this.purchasePrice = purchasePrice;

}

public Long getCount() {

return count;

}

public void setCount(final Long count) {

this.count = count;

}

public Purchase getPurchase() {

return purchase;

}

public void setPurchase(final Purchase purchase) {

this.purchase = purchase;

}

public GiftCertificate getGiftCertificate() {

return giftCertificate;

}

public void setGiftCertificate(final GiftCertificate giftCertificate) {

this.giftCertificate = giftCertificate;

}

}

package com.epam.esm.giftcertificates.entity;

import javax.persistence.\*;

import java.util.Objects;

/\*\*

\* This entity class stores tag name of gift certificate.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@Entity

@Table(name = "tag")

public class Tag {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name", nullable = false, length = 35, unique = true)

private String name;

/\*\*

\* {@inheritDoc}

\*/

@Override

public String toString() {

return "Tag{" + "id=" + id + ", name='" + name + '\'' + '}';

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public boolean equals(final Object object) {

boolean result;

Tag tag;

if (object != null) {

tag = (Tag) object;

if (this == object) {

result = true;

} else if (getClass() != object.getClass()) {

result = false;

} else {

result = Objects.equals(id, tag.id) && Objects.equals(name, tag.name);

}

} else {

result = false;

}

return result;

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public int hashCode() {

return Objects.hash(id, name);

}

public Long getId() {

return id;

}

public void setId(final Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(final String name) {

this.name = name;

}

}

package com.epam.esm.giftcertificates.entity;

import com.epam.esm.giftcertificates.entity.enumeration.UserRole;

import javax.persistence.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

/\*\*

\* This entity class stores user info.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@Entity

@Table(name = "certificate\_user")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "username", nullable = false, unique = true, length = 35)

private String username;

@Column(name = "password", nullable = false, length = 60)

private String password;

@Column(name = "user\_role", nullable = false, length = 5)

@Enumerated(value = EnumType.STRING)

private UserRole userRole;

@OneToMany(mappedBy = "user", fetch = FetchType.LAZY)

private List<Purchase> purchases;

public User() {

purchases = new ArrayList<>();

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public String toString() {

return "User{" +

"id=" + id +

", username='" + username + '\'' +

", password='" + password + '\'' +

", userRole=" + userRole +

'}';

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public boolean equals(final Object object) {

boolean result;

User user;

if (object != null) {

user = (User) object;

if (this == object) {

result = true;

} else if (getClass() != object.getClass()) {

result = false;

} else {

result = Objects.equals(id, user.id) &&

Objects.equals(username, user.username) &&

Objects.equals(password, user.password) &&

userRole == user.userRole;

}

} else {

result = false;

}

return result;

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public int hashCode() {

return Objects.hash(id, username, password, userRole);

}

public Long getId() {

return id;

}

public void setId(final Long id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(final String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(final String password) {

this.password = password;

}

public UserRole getUserRole() {

return userRole;

}

public void setUserRole(final UserRole userRole) {

this.userRole = userRole;

}

public List<Purchase> getPurchases() {

return purchases;

}

public void setPurchases(final List<Purchase> purchases) {

this.purchases = purchases;

}

}

package com.epam.esm.giftcertificates.config;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.oauth2.config.annotation.configurers.ClientDetailsServiceConfigurer;

import org.springframework.security.oauth2.config.annotation.web.configuration.AuthorizationServerConfigurerAdapter;

import org.springframework.security.oauth2.config.annotation.web.configuration.EnableAuthorizationServer;

import org.springframework.security.oauth2.config.annotation.web.configurers.AuthorizationServerEndpointsConfigurer;

import org.springframework.security.oauth2.config.annotation.web.configurers.AuthorizationServerSecurityConfigurer;

import org.springframework.security.oauth2.provider.token.TokenStore;

import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;

import org.springframework.security.oauth2.provider.token.store.JwtTokenStore;

/\*\*

\* Configuration class for authorization server role.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@Configuration

@EnableAuthorizationServer

public class AuthorizationServerConfig extends AuthorizationServerConfigurerAdapter {

private static final String CLIENT\_ID = "gift-certificates";

private static final String CLIENT\_SECRET = "$2a$04$bSMnqKXCGcMp22es4R6bsOjnX7is79Mp6Zo3N//4wlaPaRquxd1Ma";

private static final String GRANT\_TYPE\_PASSWORD = "password";

private static final String REFRESH\_TOKEN = "refresh\_token";

private static final String SCOPE\_READ = "read";

private static final String SCOPE\_WRITE = "write";

private static final String TRUST = "trust";

private static final int ACCESS\_TOKEN\_VALIDITY\_SECONDS = 3\_600;

private static final int REFRESH\_TOKEN\_VALIDITY\_SECONDS = 21\_600;

private AuthenticationManager authenticationManager;

@Autowired

public AuthorizationServerConfig(final AuthenticationManager authenticationManager) {

this.authenticationManager = authenticationManager;

}

/\*\*

\* Bean for object {@link JwtAccessTokenConverter}.

\*

\* @return value of the object {@link JwtAccessTokenConverter}

\*/

@Bean

public JwtAccessTokenConverter accessTokenConverter() {

JwtAccessTokenConverter converter = new JwtAccessTokenConverter();

converter.setSigningKey("as466gf");

return converter;

}

/\*\*

\* Bean for object {@link JwtTokenStore}.

\*

\* @return value of the object {@link JwtTokenStore}

\*/

@Bean

public TokenStore tokenStore() {

return new JwtTokenStore(accessTokenConverter());

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public void configure(final ClientDetailsServiceConfigurer clients) throws Exception {

clients

.inMemory()

.withClient(CLIENT\_ID)

.secret(CLIENT\_SECRET)

.authorizedGrantTypes(GRANT\_TYPE\_PASSWORD, REFRESH\_TOKEN)

.scopes(SCOPE\_READ, SCOPE\_WRITE, TRUST)

.accessTokenValiditySeconds(ACCESS\_TOKEN\_VALIDITY\_SECONDS)

.refreshTokenValiditySeconds(REFRESH\_TOKEN\_VALIDITY\_SECONDS);

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public void configure(final AuthorizationServerEndpointsConfigurer endpoints) {

endpoints.tokenStore(tokenStore())

.authenticationManager(authenticationManager)

.accessTokenConverter(accessTokenConverter());

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public void configure(final AuthorizationServerSecurityConfigurer security) {

security

.tokenKeyAccess("permitAll()")

.checkTokenAccess("isAuthenticated()");

}

}

package com.epam.esm.giftcertificates.config;

import org.modelmapper.ModelMapper;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.support.ResourceBundleMessageSource;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

/\*\*

\* Runner class for SpringBoot application.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@SpringBootApplication

@ComponentScan("com.epam.esm")

@EntityScan("com.epam.esm.giftcertificates.entity")

public class GiftCertificatesApp {

private static final String VALIDATION\_MESSAGES = "ValidationMessages";

public static void main(final String[] args) {

SpringApplication.run(GiftCertificatesApp.class, args);

}

/\*\*

\* Bean for object {@link ModelMapper}.

\*

\* @return value of the object {@link ModelMapper}

\*/

@Bean

public ModelMapper getModelMapper() {

return new ModelMapper();

}

/\*\*

\* Bean for object {@link ResourceBundleMessageSource}.

\*

\* @return value of the object {@link ResourceBundleMessageSource}

\*/

@Bean

public ResourceBundleMessageSource getMessageSource() {

final ResourceBundleMessageSource source = new ResourceBundleMessageSource();

source.setBasenames(VALIDATION\_MESSAGES);

return source;

}

/\*\*

\* Bean for object {@link BCryptPasswordEncoder}.

\*

\* @return value of the object {@link BCryptPasswordEncoder}

\*/

@Bean

public BCryptPasswordEncoder encoder() {

return new BCryptPasswordEncoder(4);

}

}

package com.epam.esm.giftcertificates.config;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Primary;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.oauth2.config.annotation.web.configuration.EnableResourceServer;

import org.springframework.security.oauth2.config.annotation.web.configuration.ResourceServerConfigurerAdapter;

import org.springframework.security.oauth2.config.annotation.web.configurers.ResourceServerSecurityConfigurer;

import org.springframework.security.oauth2.provider.token.DefaultTokenServices;

import org.springframework.security.oauth2.provider.token.TokenStore;

/\*\*

\* Configuration class for resource server role.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@Configuration

@EnableResourceServer

public class ResourceServerConfig extends ResourceServerConfigurerAdapter {

private static final String RESOURCE\_ID = "resource\_id";

private TokenStore tokenStore;

@Autowired

public ResourceServerConfig(final TokenStore tokenStore) {

this.tokenStore = tokenStore;

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public void configure(final ResourceServerSecurityConfigurer resources) {

resources.resourceId(RESOURCE\_ID).stateless(false);

}

/\*\*

\* {@inheritDoc}

\*/

@Override

public void configure(final HttpSecurity http) throws Exception {

http

.authorizeRequests()

.anyRequest()

.permitAll();

}

/\*\*

\* Bean for object {@link DefaultTokenServices}.

\*

\* @return value of the object {@link DefaultTokenServices}

\*/

@Bean

@Primary

public DefaultTokenServices tokenServices() {

DefaultTokenServices defaultTokenServices = new DefaultTokenServices();

defaultTokenServices.setTokenStore(tokenStore);

return defaultTokenServices;

}

}

package com.epam.esm.giftcertificates.config;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

/\*\*

\* Configuration class for security.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

public class SecurityConfig extends WebSecurityConfigurerAdapter {

private UserDetailsService userService;

@Autowired

public SecurityConfig(@Qualifier(value = "userService") final UserDetailsService userService) {

this.userService = userService;

}

/\*\*

\* Bean for object {@link AuthenticationManager}.

\*

\* @return value of the object {@link AuthenticationManager}

\*/

@Override

@Bean

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

@Autowired

public void globalUserDetails(final AuthenticationManagerBuilder auth,

final BCryptPasswordEncoder encoder) throws Exception {

auth.userDetailsService(userService)

.passwordEncoder(encoder);

}

/\*\*

\* {@inheritDoc}

\*/

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.csrf()

.disable()

.anonymous()

.disable()

.authorizeRequests()

.antMatchers("/")

.permitAll()

.antMatchers("/oauth/token")

.permitAll()

.anyRequest()

.authenticated()

.and()

.formLogin()

.permitAll();

}

}

package com.epam.esm.giftcertificates.controller;

import com.epam.esm.giftcertificates.dto.\*;

import com.epam.esm.giftcertificates.service.CertificateService;

import com.epam.esm.giftcertificates.service.TagService;

import com.epam.esm.giftcertificates.service.UserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.web.bind.annotation.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

/\*\*

\* Controller for work with dto class {@link UserDto}.

\*

\* @author Vadim\_Orol

\* @version 1.0

\*/

@RestController

@RequestMapping("/users")

public class UserController {

private UserService userService;

private CertificateService certificateService;

private TagService tagService;

@Autowired

public UserController(final UserService userService, final CertificateService certificateService,

final TagService tagService) {

this.userService = userService;

this.certificateService = certificateService;

this.tagService = tagService;

}

/\*\*

\* Get method for select all objects {@link UserDto}.

\*

\* @param username value of the username

\* @param offset value of the pagination offset

\* @param limit value of the pagination data max limit

\* @return list of objects {@link UserDto}

\*/

@GetMapping

@PreAuthorize("hasRole('ADMIN')")

public List<UserDto> findAll(@RequestParam final Optional<String> username,

@RequestParam(defaultValue = "0") final Integer offset,

@RequestParam(defaultValue = "10") final Integer limit) {

String name;

List<UserDto> users;

if (!username.isPresent()) {

users = userService.findAll(new PaginationDto(offset, limit));

} else {

name = username.get();

users = new ArrayList<UserDto>() {{

add(userService.findByUsername(name));

}};

}

return users;

}

/\*\*

\* Get mapping for select object {@link GiftCertificateDto} for user.

\*

\* @param username value of the username

\* @param offset value of the pagination offset position

\* @param limit value of the pagination data max limit

\* @return value of the object {@link ResponseEntity}

\*/

@CrossOrigin

@GetMapping("/{username}/certificates")

@PreAuthorize("hasRole('USER') or hasRole('ADMIN')")

public ResponseEntity<CertificatePageDto> findUserCertificates(@RequestParam(defaultValue = "0") final Integer offset,

@RequestParam(defaultValue = "10") final Integer limit,

@PathVariable final String username) {

CertificatePageDto certificatePage = certificateService.findByUsername(username,

new PaginationDto(offset, limit));

return new ResponseEntity<>(certificatePage, HttpStatus.OK);

}

/\*\*

\* Get mapping for select object {@link TagDto} by user id.

\*

\* @param id value of the user id

\* @return value of the object {@link ResponseEntity}

\*/

@GetMapping("/{id}/tags")

@PreAuthorize("hasRole('USER') or hasRole('ADMIN')")

public ResponseEntity<TagDto> findPopularTag(@PathVariable final Long id) {

return new ResponseEntity<>(tagService.findByUserId(id), HttpStatus.OK);

}

/\*\*

\* Get method for select object {@link UserDto} by id.

\*

\* @param id value of the object id

\* @return value of the object {@link ResponseEntity}

\*/

@GetMapping("/{id}")

@PreAuthorize("hasRole('ADMIN')")

public ResponseEntity<UserDto> findById(@PathVariable final Long id) {

return new ResponseEntity<>(userService.findById(id), HttpStatus.OK);

}

/\*\*

\* Delete method for remove object {@link UserDto}.

\*

\* @param id value of the object id

\* @return value of the object {@link ResponseEntity}

\*/

@DeleteMapping("/{id}")

@PreAuthorize("hasRole('ADMIN')")

public ResponseEntity<HttpStatus> delete(@PathVariable final Long id) {

userService.remove(id);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

/\*\*

\* Post method for create object {@link UserDto}.

\*

\* @param user value of the object {@link UserDto}

\* @return value of the object {@link ResponseEntity}

\*/

@PostMapping

@PreAuthorize("!hasRole('USER') and !hasRole('ADMIN')")

public ResponseEntity<UserDto> create(@RequestBody final UserDto user) {

return new ResponseEntity<>(userService.add(user), HttpStatus.OK);

}

}