

1
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

СТАРТАП-ПРОЕКТ. ВЕБ-ПРИЛОЖЕНИЕ

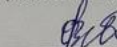
«ФУД-КАРТА»

Студент


17.06.2024
(подпись, дата)

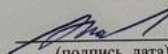
Кодиров Искандарходжа
Нусратиллоходжаевич
(ФИО полностью)

Руководитель


17.06.2024
(подпись, дата)

Юрова О.В.
(фамилия, инициалы)

Нормоконтролёр


17.06.2024
(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от
20.06.2024, № 32)

Секретарь ГЭК


(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«19» 04 2024 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Кодиров Искандарходжа Нусратиллоходжаевич гр. 20ВВП1 факультета вычислительной техники

2. Тема работы Стартап-проект. Веб-приложение «Фуд-карта»

Тема утверждена приказом ПГУ №_405/о_____ от "19" _____ апреля _____ 2024 __ г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Юрова О.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

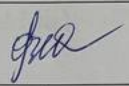
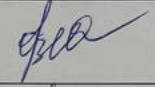
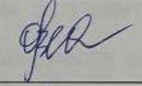
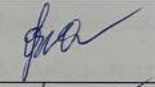
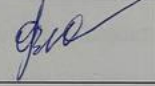
Разработать веб – приложение «Фуд-карта», позволяющее создавать личные библиотеки меток в картах и работать с ними. Составить бизнес-модель и финансовый план. Язык программирования Java, среда разработки IntelliJ IDEA. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. Spring Framework
2. ORM
3. PostgreSQL Database

5. Перечень подлежащих разработке вопросов

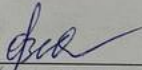
Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки мобильного приложения. Выбрать язык программирования, среду разработки и базу данных.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	45	19.04.2024 – 25.04.2024	
2	Анализ требований	26	26.04.2024 – 30.04.2024	
3	Выбор технологий, применяемых при проектировании и разработке	55	01.05.2024 – 06.05.2024	
4	Создание пользовательского интерфейса	65	06.05.2024 – 14.05.2024	
5	Разработка серверной части	84	14.05.2024 – 24.05.2024	
6	Оформление пояснительной записки	40	25.05.2024 – 31.05.2024	

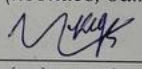
Дата выдачи задания "19" апреля 2024 г.

Руководитель бакалаврской работы _____


(подпись, дата)

Юрова О.В.

Задание к исполнению принял студент _____


(подпись, дата)

Кодиров И.Н.

Работу к защите допустить
Декан факультета _____

 19.06.2024
(подпись, дата)

Фионова Л.Р.

Реферат

Бакалаврская работа содержит 86 листов, 17 рисунков, 4 таблицы, 8 источников, 3 приложения

ВЕБ – ПРИЛОЖЕНИЕ, INTELLIJ IDEA, JAVA, SQL, SPRING FRAMEWORK, POSTGRESQL, HTML, CSS, JS

Цель работы – разработать веб-приложение, позволяющее создавать маршруты для гастрономического туризма и составления личных списков избранных ресторанов и точек общественного питания.

Разработанное приложение должно позволять пользователям создавать собственные списки мест и заведений, а также предоставлять возможность их редактирования, например, изменение описания метки или удаление метки. Помимо этого, приложение будет позволять строить маршруты до выбранных мест и предоставлять удобный интерфейс для взаимодействия с картой. Списки создаются и сохраняются конкретным пользователем поэтому в приложении необходимо авторизация и регистрация.

Результатом данной работы является веб – приложение, которое соответствует поставленным задачам. А именно разработанное, протестированное и работающее приложение для введения личных меток местоположений в городах, позволяющее собирать интересные места в свои библиотеки. Серверная часть приложения была реализована на языке Java, с использованием Spring Framework, для работы с базой данных использовался язык SQL. Для хранения данных была выбрана СУБД PostgreSQL. Для реализации пользовательского интерфейса были использован язык разметки HTML, язык каскадных таблиц стилей CSS, и язык программирования JavaScript. Среда разработки IntelliJ IDEA.

					ПГУ1.090301.10.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.	Кодиров И.Н.			18.06.24	Лит.	Лист	Листов
Провер.	Юрова О.В.			18.06.24		2	86
Н. Контр	Бычков А.С.			18.06.24	ФВТ, каф. ВТ гр. 20ВВП1		
Утверд.	Митрахин М.А.			18.06.24			
					Стартап-проект. Веб-приложение «Фуд-карта» Пояснительная записка		

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Стартап-проект. Веб – приложение «Фуд карта».....	7
1.1 Анализ темы	7
1.2 Анализ аналогов.....	7
1.2.1 Google Maps.....	8
1.2.2 Яндекс Карты	9
1.2.3 Yelp.....	11
1.2.4 Вывод	13
1.3 Постановка задачи	13
2 Выбор инструментов разработки	14
2.1 Выбор языка программирования.....	14
2.1.1 Язык программирования «Java».....	14
2.1.2 Язык программирования «JavaScript»	15
2.1.3 Язык программирования «SQL»	15
2.1.4 Вывод	15
2.2 Выбор среды разработки.....	15
2.3 Выбор базы данных	16
3 Архитектура веб – приложения	17
3.1 Паттерн «MVC».....	17
4 Реализация приложения	18
4.1 Описание функционала	18
4.2 Описание работы приложения.....	23
4.3 Основная реализация приложения	24
4.3.1 Configuration	25
4.3.2 Controller	26
4.3.3 Service.....	28
4.3.4 Repository	30

4.3.5 Entity	31
5 Тестирование	35
5.1 Модульное тестирование	35
5.2 Вывод.....	40
6 Безопасность жизнедеятельности.....	41
6.1 Требования безопасности при работе с ПК и рабочему месту.	41
6.2 Требования к освещению.	42
7 Стартап проект	44
7.1 Описание продукта	44
7.2 Ценностное предложение.....	44
7.3 Потоки доходов	45
7.4 Ключевые ресурсы.....	45
7.5 Ключевые виды деятельности	45
7.6 Оценка затрат	46
7.7 Оценка доходов	46
7.8 Расчет срока окупаемости	47
Заключение	48
Список используемых источников.....	49
Приложение А – UML-диаграммы.....	50
Приложение Б – Листинг программы	55
Приложение В – Презентация.....	82

Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения.

БД – база данных

ОС – операционная система

ПО – программное обеспечение

ПК – персональный компьютер

СУБД – система управления базами данных

IDE – (от англ. Integrated Development Environment) – интегрированная среда разработки

JS – (от англ. JavaScript) – язык программирования

UI – (от англ. User Interface) – пользовательский интерфейс

UML – (от англ. Unified Modeling Language) – унифицированный язык моделирования

					ПГУ1.090301.10.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В современном мире развитие информационных технологий позволило людям создавать интерактивные карты местностей и взаимодействовать с ними. Благодаря такому развитию появляется вопрос информированности о разнообразных заведениях в ресторанной индустрии, занимающей значительную роль в жизни каждого из нас. Существование таких технологий создаёт новые способы взаимодействия с ними. В данном контексте, создание карты, позволяющей создавать собственные списки заведений с собственным описанием и особенностями приглянувшимися вам и другим людям, становится удобным инструментом для планирования собственного досуга, и полезным не только для конечного потребителя, но и для самих владельцев заведений.

Результатом данной работы является проект, соответствующий поставленной задаче, а именно: разработанное и протестированное приложение для работы с картой, позволяющее создавать библиотеки маршрутов по городу, метки и полное их содержание создаются пользователем и сохраняются на его аккаунте. Данная работа является попыткой внести вклад в развитие информационных технологий в сфере общественного питания и туризма.

					ПГУ1.090301.10.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1 Стартап-проект. Веб – приложение «Фуд карта»

1.1 Анализ темы

В наше время люди сталкиваются с проблемой переизбытка существующих ресторанов и кафе среди, которых сложно выбрать подходящее для вас место. А множество существующих приложений предлагают формальную информацию о заведениях, но не предоставляют возможности создания персонализированных меток и ведения собственных записей.

Приложения позволяющее вести подобные записи, может быть очень полезно не только для путешествий или какого-либо туризма, но и для собственного удобства. Оно даёт возможность быстро и удобно, искать и выбирать нужные места из списка собственных избранных мест или любых других мест на карте, а также строить до них маршруты.

Следует отметить, что подобные приложения имеют свои недостатки, такие как сложность внедрения и высокие затраты на разработку и поддержку. Но учитывая растущую потребность пользователей в быстром и удобном доступе к информации, их использование становится очень актуальным в наше время.

1.2 Анализ аналогов

Поскольку тема работы с картами всегда являлась актуальной и важной, существует много качественных реализаций, в которых от части реализованы данные функции. В данном пункте рассмотрены такие популярные сервисы как:

- Google Карты;
- Яндекс Карты;
- Yelp.

Проведен анализ и выявлены достоинства и недостатки каждого из них.

1.2.1 Google Maps

Удобный сервис для работы с картами, представляет из себя набор приложений, карту и спутниковые снимки нашей планеты, интегрирован в экосистему Google и имеет большое количество возможностей.

Достоинства:

- большая и актуальная база данных. Хранят информацию почти о любой точке земного шара, как объекте инфраструктуры, так и коммерческих помещений зданий и постоянно обновляют информацию;
- множество функций. Пользователи, имеющие Google аккаунт, могут использовать совместно другие сервисы от Google, такие как календарь, почту, отзывы и рейтинги;
- точность. Точная информация о дорожных ситуациях построение маршрутов для водителей, пешеходов и пользователей общественного транспорта.

Недостатки:

- возможная ограниченная доступность в нашем регионе;
- наличие большого количества рекламы и баннеров.

Интерфейс этого сервиса представлен на рисунке 1

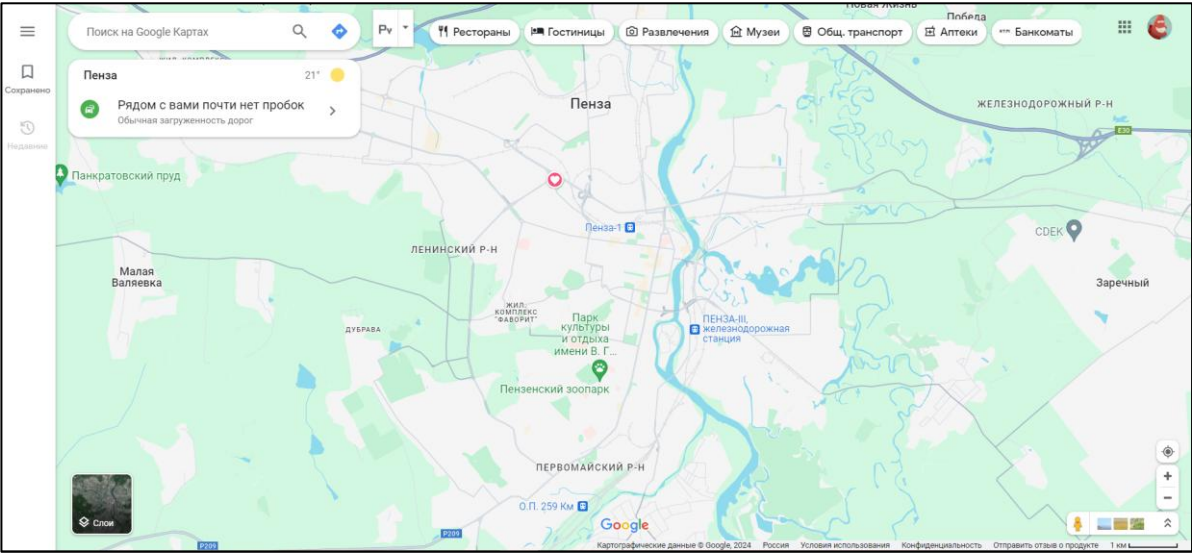


Рисунок 1 - Главная страница с картой

Так же в этих картах реализовано боковое меню изображено на рисунке 2

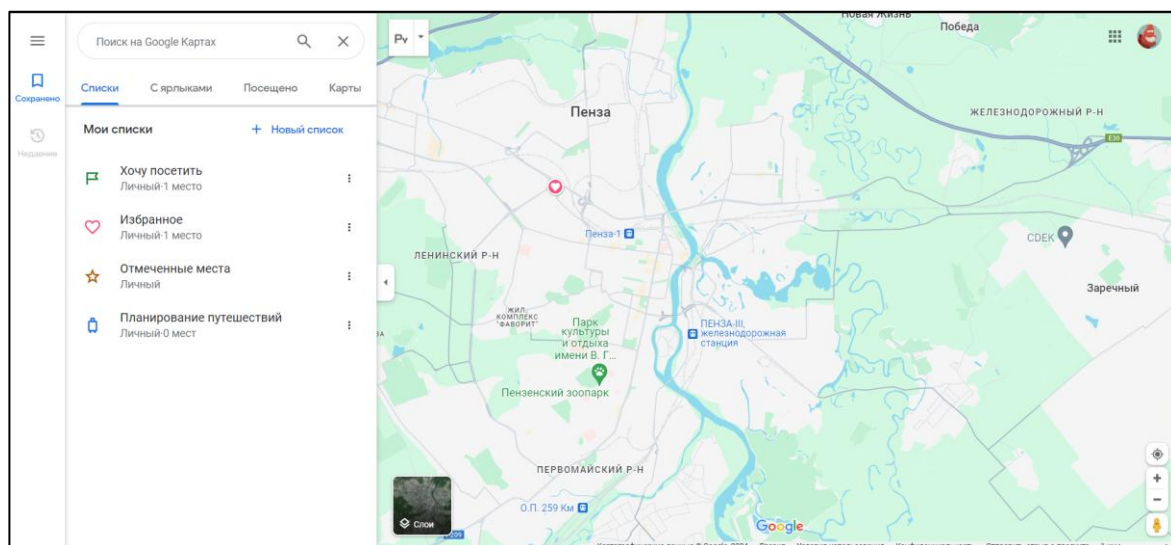


Рисунок 2 – Боковое меню с метками

1.2.2 Яндекс Карты

Популярный картографический сервис, содержащий обширные и актуальные карты с возможностью поиска различных объектов инфраструктуры от АЗС до ресторанов и банков. Могут планировать маршруты с учётом расписаний общественных средств передвижения, таких как метро, автобусы, трамваи. А интеграции с другими сервисами от Яндекса делают их ещё удобнее. Позволяют сохранять метки в избранное и предлагают подборки мест по городам.

Достоинства:

- большая и актуальная база данных организаций и объектов по всей РФ. Включая информацию об этих местах, и отзывы от людей с описанием и изображениями;
- подробные карты. Отображают трафик и движения маршрутных ТС и других удобных функций для планирования маршрутов;
- удобный интерфейс. Интуитивно понятный интерфейс с поиском и фильтрацией объектов, что упрощает использование конечного пользователю;

– интеграция с другими сервисами Яндекса. Возможность совмещать работу карт с такими сервисами как Яндекс.Еда или Яндекс.Такси для удобства перемещения.

Недостатки:

– неточность в данных об организациях, что может привести к проблемам в поиске;

– наличие большого количества рекламы, что может отвлекать пользователя;

– перегруженное приложение: большое количество интегрированных сервисов, слишком много различных функций, в которых нужно разбираться.

Основная страница с картой и кнопками Яндекса позволяющие менять режимы отображения самой карты, показывая пробки или маршруты движения, сбоку видна иконка пользователя и кнопки для увеличения уменьшения масштаба карты рисунок 3.

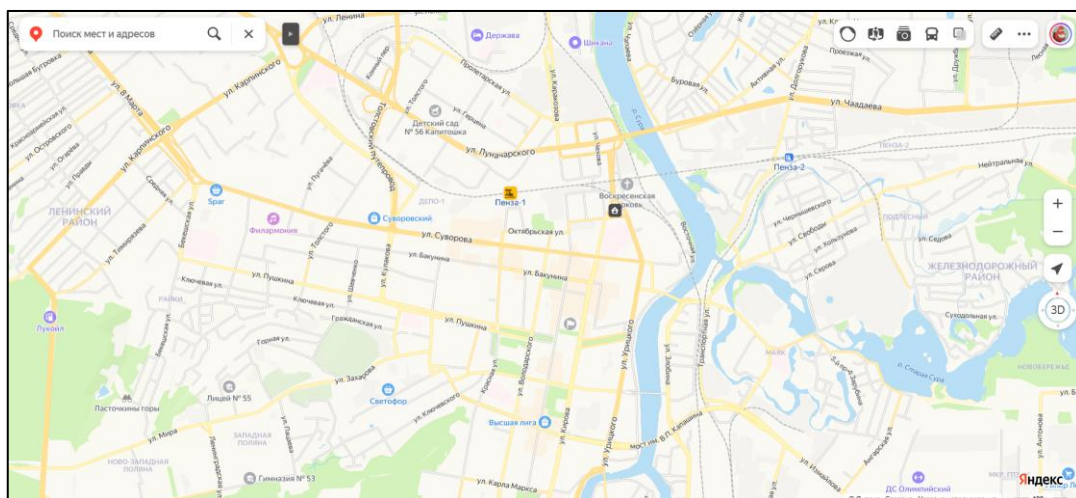


Рисунок 3 – Главная страница с картой

Так же в приложении имеются подборки мест по городам, созданные и отобранные по рекомендациям пользователей на рисунке 4.

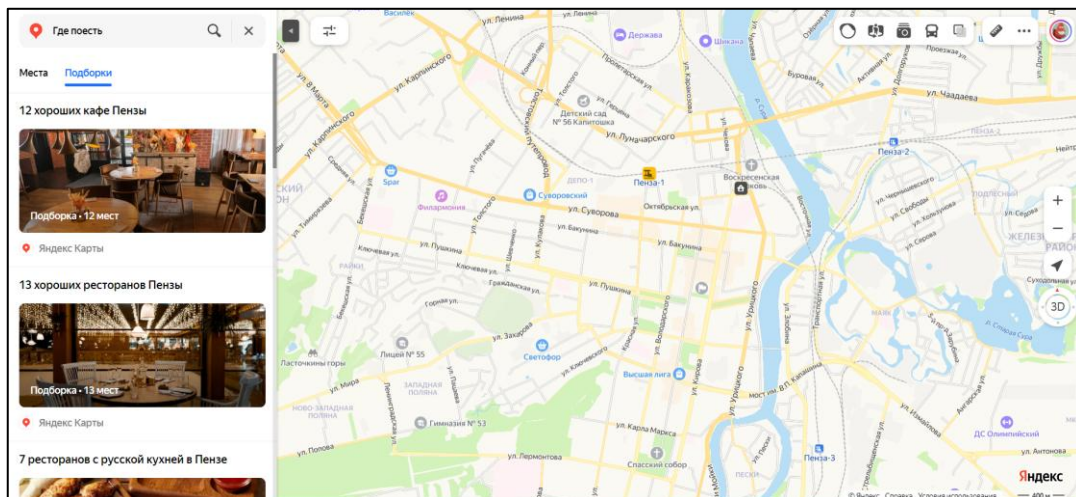


Рисунок 4 - Боковое меню с метками

1.2.3 Yelp

Это веб – сайт, в котором можно искать рестораны и другие места на местном рынке услуг, смотреть рейтинги и обзоры, заведения имеют собственные страницы, с которыми можно взаимодействовать.

Достоинства:

- большая база отзывов. Данная площадка имеет огромное количество пользователей и почти на каждое заведение имеется большая база отзывов, благодаря которым пользователи могут принимать обоснованные решения;
- подробная информация о заведениях. Каждая страница содержит подробную информацию о заведении начиная от часов работы и меню до фотографий ссылок на их собственные интернет – ресурсы;
- аккаунты для предпринимателей. Пользователи владеющие такими аккаунтами могут управлять своими заведениями в данной площадке обновляя информацию и запуская рекламные компании;
- социальная сеть. Есть возможность добавлять друзей следить за их отзывами и рекомендациями, а также делиться собственными;

– интеграция со сторонними сервисами. Наличие таких возможностей как бронирование заказа, столика в ресторане, оформления доставок делает его универсальным и удобным для пользователей.

Недостатки:

- только английская локализация;
- недоступен в РФ.

Интерфейс приложения представлен на рисунке 5.

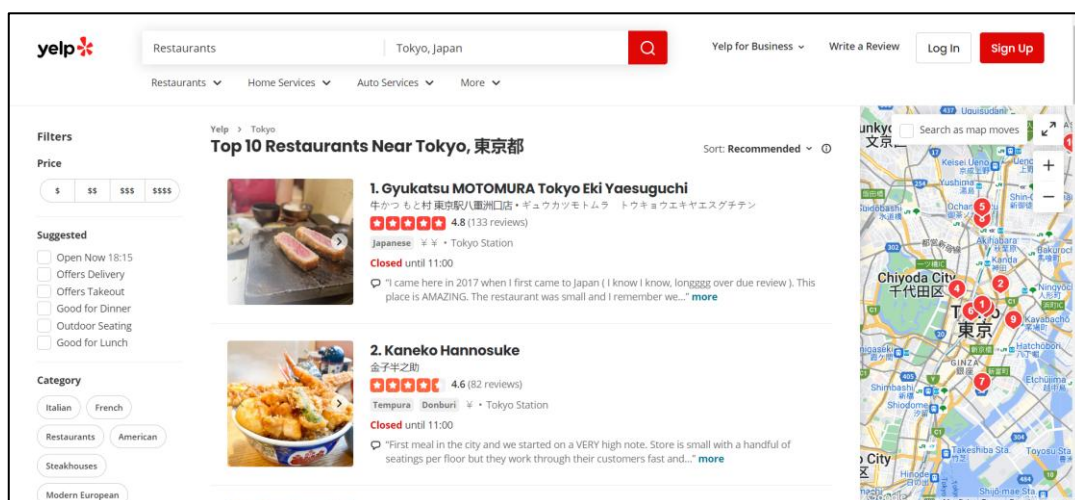


Рисунок 5 - Главная страница со списком привычек

1.2.4 Вывод

Результаты анализа существующих решений помогают понять, что базовый функционал карт у аналогов почти что не меняется, а меняется только то, как преподносят его. Также важно понимать, что веб-сайт — это только удобный способ получить информацию об интересующем вас месте. Он должен обладать простым, гибким и удобным интерфейсом для комфортного использования. Можно заметить, что ни один из сервисов не предлагает возможности создавать собственные библиотеки и делиться местоположениями, что конечно для обычных пользователей добавляет больше возможностей.

1.3 Постановка задачи

Необходимо разработать веб-приложение «Фуд карта», которое должно обладать следующим функционалом.

В серверной части веб-приложения должно быть реализовано:

- вход и регистрация пользователя;
- проверку корректности вводимых данных;
- обеспечение безопасности, позволяя добавлять метки в базу, только авторизованным пользователям;
- добавление меток их редактирование и удаление в базу данных.

Со стороны клиента должно быть реализовано:

- удобный и интуитивно понятный интерфейс;
- картографический сервис отображающий все метки;
- просмотр списка меток;
- построение маршрутов к конечным меткам.

Минимальные системные требования:

- операционная система Windows (8, 10, 11);
- веб-браузер: Yandex Browser, Google Chrome, Mozilla Firefox;
- наличие интернета.

2 Выбор инструментов разработки

Для разработки и оптимизации веб-приложения «Фуд карта» необходимо использовать соответствующие технологии, языки программирования, среды разработки, базы данных всё будет рассмотрено в этом разделе.

2.1 Выбор языка программирования

На ОС Windows можно писать на огромном количестве языков, и у каждого из них свои области применения. В данной работе необходимо реализовать веб –приложение и поэтому важно выбрать наиболее эффективные и простые инструменты для этого. В этой работе для реализации серверной и клиентской составляющей были выбраны следующие языки:

- Java;
- JavaScript;
- SQL.

Каждый из них отвечает за свою часть в реализации приложения, некоторые обладают дополнительными библиотеками и фреймворками расширяющими возможности в разработке.

2.1.1 Язык программирования «Java»

Один из самых популярных языков для программирования веб-приложений, известен своей надежностью, безопасностью и масштабируемостью. Это многоплатформенный, объектно-ориентированный язык программирования, который сам может использоваться как платформа. Подходит для написания высокопроизводительных программ, способных обрабатывать большое количество данных, и имеет большое количество библиотек и фреймворков. Один из лучших его фреймворков для разработки в вэбе это Spring Framework. Он предоставляет кучу возможностей, упрощающих написание, тестирование и повторное использование кода. Модуль Spring MVC

обеспечивает удобные инструменты для создания веб-приложений, а модуль Spring Boot ускоряет настройку и развертывание приложений.

Этот фреймворк также включает такие дополнения как Spring Security для реализации безопасности, Spring Data JPA/JDBC упрощающий работу с реляционными и нереляционными базами данных. А активное сообщество и постепенные обновления делают его лучшим выбором для программиста.

2.1.2 Язык программирования «JavaScript»

JavaScript – интерпретируемый язык программирования, является неотъемлемой частью при разработке веб-приложений. Обеспечивает динамичность, интерактивность, отзывчивость веб-страниц. Имеет множество библиотек, которые упрощают разработку и обеспечивают высокую производительность. Исходя из выше перечисленного, является незаменимым при разработке веб-приложений.

2.1.3 Язык программирования «SQL»

SQL – язык разработанный специально для работы с реляционными базами данных. Является основным языком для работы с базами данных. Использует декларативный подход, что значит для получения результата необходимо написать не способ его получения, а сам желаемый результат. Позволяет проводить различные манипуляции с данными в БД, создавать и управлять их структурами данных. Поддерживает транзакции.

2.1.4 Вывод

Каждый из этих языков является, если не основным, то одним из лидирующих в своей сфере разработки. Они предлагают всё необходимое для реализации не только этого, но и большинства других веб-разработок, имеют достаточную гибкость и большой вес в разработке.

2.2 Выбор среды разработки

Рассмотрим самую популярную и поддерживаемую IDE для разработки на языке Java – IntelliJ IDEA. IntelliJ IDEA — это интегрированная среда разработки

(IDE) от JetBrains, популярная среди разработчиков благодаря поддержке множества языков, включая Java, JavaScript, SQL и другие. Ключевые особенности включают умное автодополнение кода, мощные инструменты рефакторинга, навигации и поиска, а также интеграцию с системами контроля версий.

IDE поддерживает различные фреймворки для тестирования, системы сборки, и веб-разработку. Огромное количество плагинов позволяет расширять функциональность IDE. Преимущества IntelliJ IDEA включают интуитивный интерфейс, высокую производительность, функции анализа и рефакторинга кода, сильную интеграцию с экосистемой JetBrains и сообщество с хорошей поддержкой. IntelliJ IDEA является мощным инструментом для профессиональной разработки программного обеспечения, предлагая богатый набор функций для повышения продуктивности и качества кода.

2.3 Выбор базы данных

В качестве базы данных была выбрана PostgreSQL. Это мощная, открытая и объектно-реляционная система управления базами данных, известная своей надежностью, производительностью. Она поддерживает множество стандартных функций SQL, обеспечивает надёжное управление над данными с поддержкой ACID – транзакций. Её возможности включают аутентификацию, шифрование данных и управление доступом на уровне строк. Имеет активное сообщество разработчиков и пользователей и предоставляет обширную документацию. Эта гибкая и настраиваемая СУБД подходит для широкого спектра приложений, от небольшого проекта до крупной корпоративной системы.

					ПГУ1.090301.10.001 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

3 Архитектура веб – приложения

3.1 Паттерн «MVC»

В данном разделе будет описана архитектура веб – приложения и взаимодействие выбранных технологий внутри программы. Шаблон программирования используемый в это приложения – MVC.

MVC – это популярный подход при разработке сетевых приложений, он разделяет приложение на три составляющие: модель (Model), отображение (View) и контроллер (Controller).

Модель – представляет из себя набор всех классов, отвечающих за бизнес-логику, она инкапсулирует в себе данные приложения и состоит из бинов.

Отображение – набор классов, который отвечает за отображение данных из модели, по сути создавая пользовательский интерфейс, как правило генерирует HTML – код, результат которого мы можем видеть в браузере.

Контроллер – набор классов, которые обрабатывают запросы пользователя. Они являются мостом между моделью и отображением, классы получают соответствующий запрос создают модель и далее передают её для отображения.

В данной работе помимо работы с этим паттерном, необходимо для более правильного написания кода, соблюдать и другие паттерны проектирования. Для работы с базами данных соблюдать принципы ACID. При написании самой программы необходимо соблюдать принципы SOLID.

4 Реализация приложения

Перед началом разработки программного обеспечения следует разобраться в архитектуре и функционале разрабатываемого приложения. В этой главе будут рассмотрены основные компоненты, функционал приложения, сущности в БД.

4.1 Описание функционала

Данным веб-приложением можно пользоваться, как и авторизованным, так и неавторизованным пользователям. Но для испытания всего функционала данного веб-приложение необходима авторизация, либо регистрация. После неё пользователю открываются все возможности для взаимодействия с картой:

- добавление меток;
- редактирование меток;
- удаление меток;
- поиск по меткам;
- переход по метке
- построение маршрутов.

Данные варианты использования представлены на диаграмме см. рисунок

8.

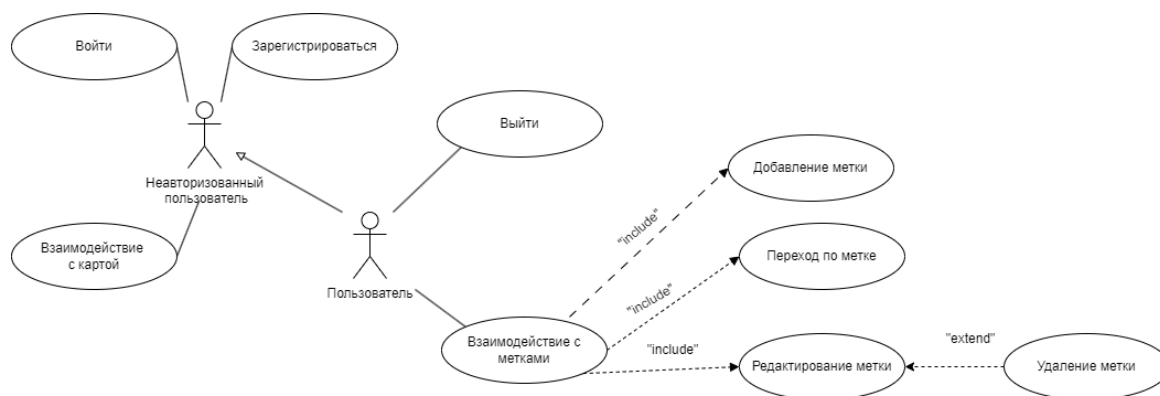


Рисунок 6 – UML-диаграмма вариантов использования для пользователей

Функции данного веб-приложения можно разделить на две группы:

- для авторизованного пользователя;
- для неавторизованного пользователя.

Неавторизованный пользователь обладает меньшим количеством функций, он имеет возможность пользоваться картой, посмотреть интерфейс веб-приложения и авторизоваться.

Для авторизованного пользователя уже открывается весь набор функций, которые позволяют ему помимо просмотра и базового взаимодействия с картой создавать собственные метки, загружать в них изображения, добавлять описание и ставить оценки, помимо этого пользователь имеет возможность переходить по каждой метки, получать их местоположение на карте и строить до них маршруты.

Пользователь также может выйти из аккаунта или в любой момент времени просто закрыть вкладку.

					ПГУ1.090301.10.001 ПЗ	Лист
						19
Изм.	Лист	№ докум.	Подпись	Дата		

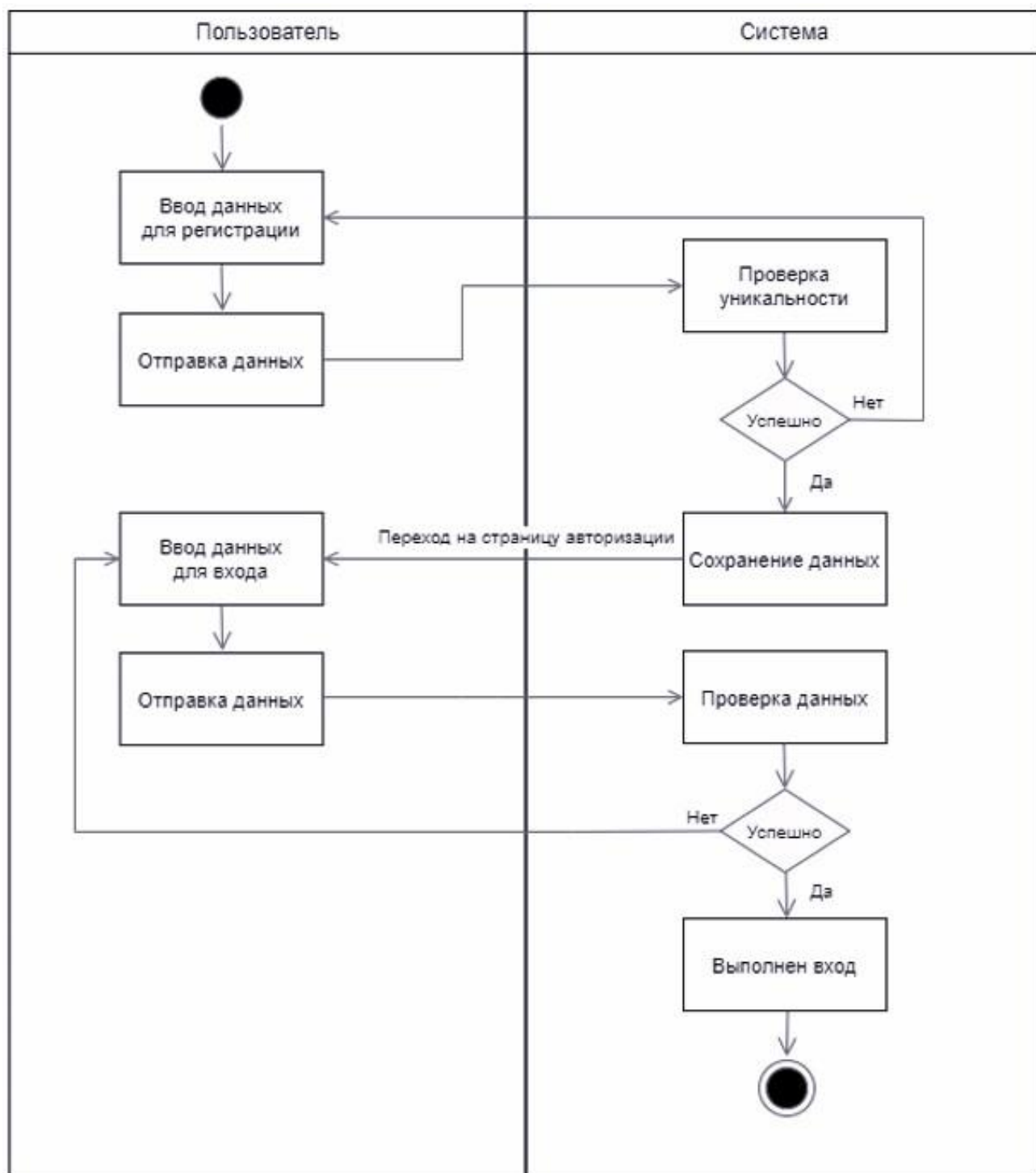


Рисунок 7 – UML-диаграмма деятельности для авторизации/регистрации

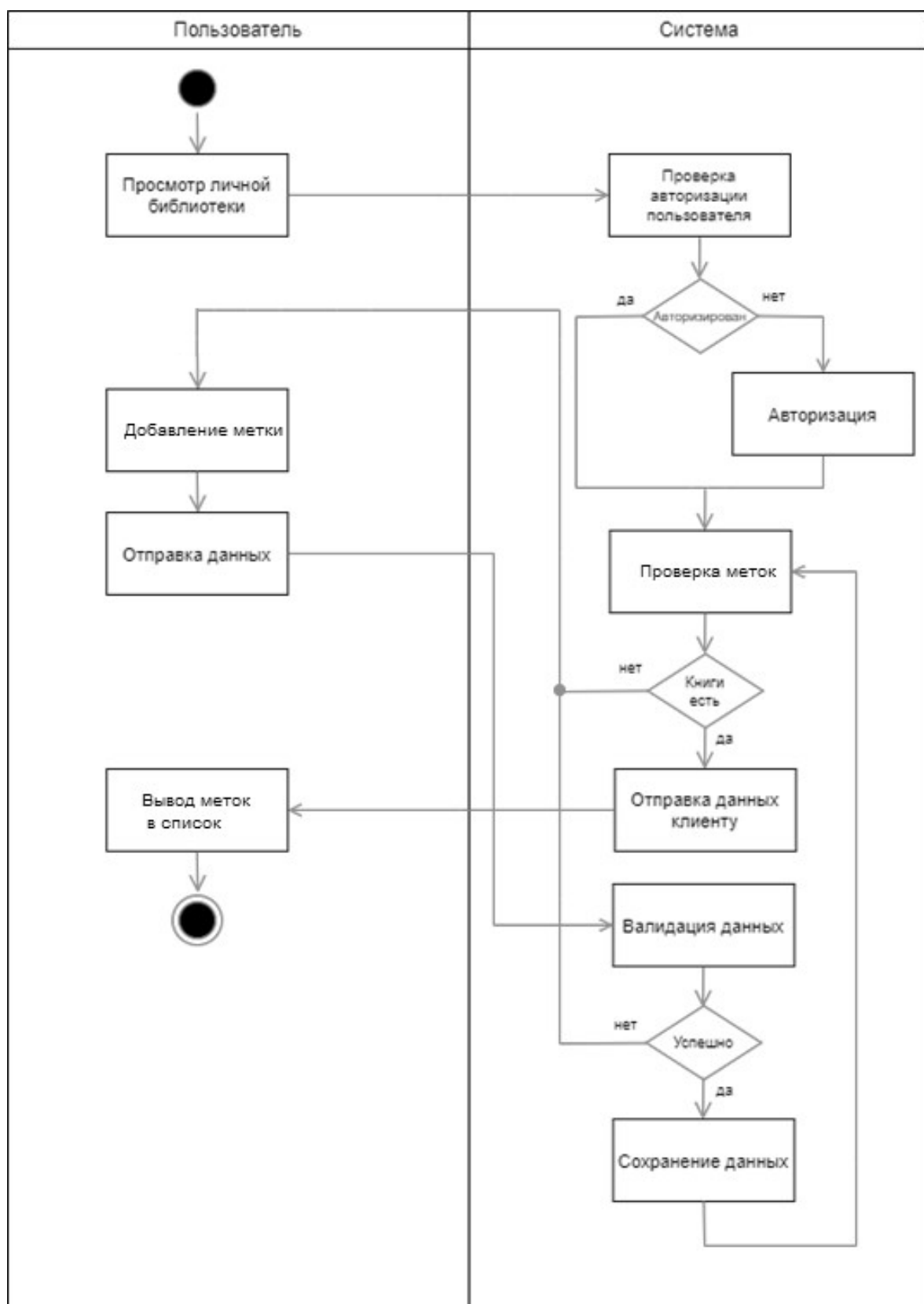


Рисунок 8 – UML диаграмма деятельности для просмотра списка меток и добавления метки

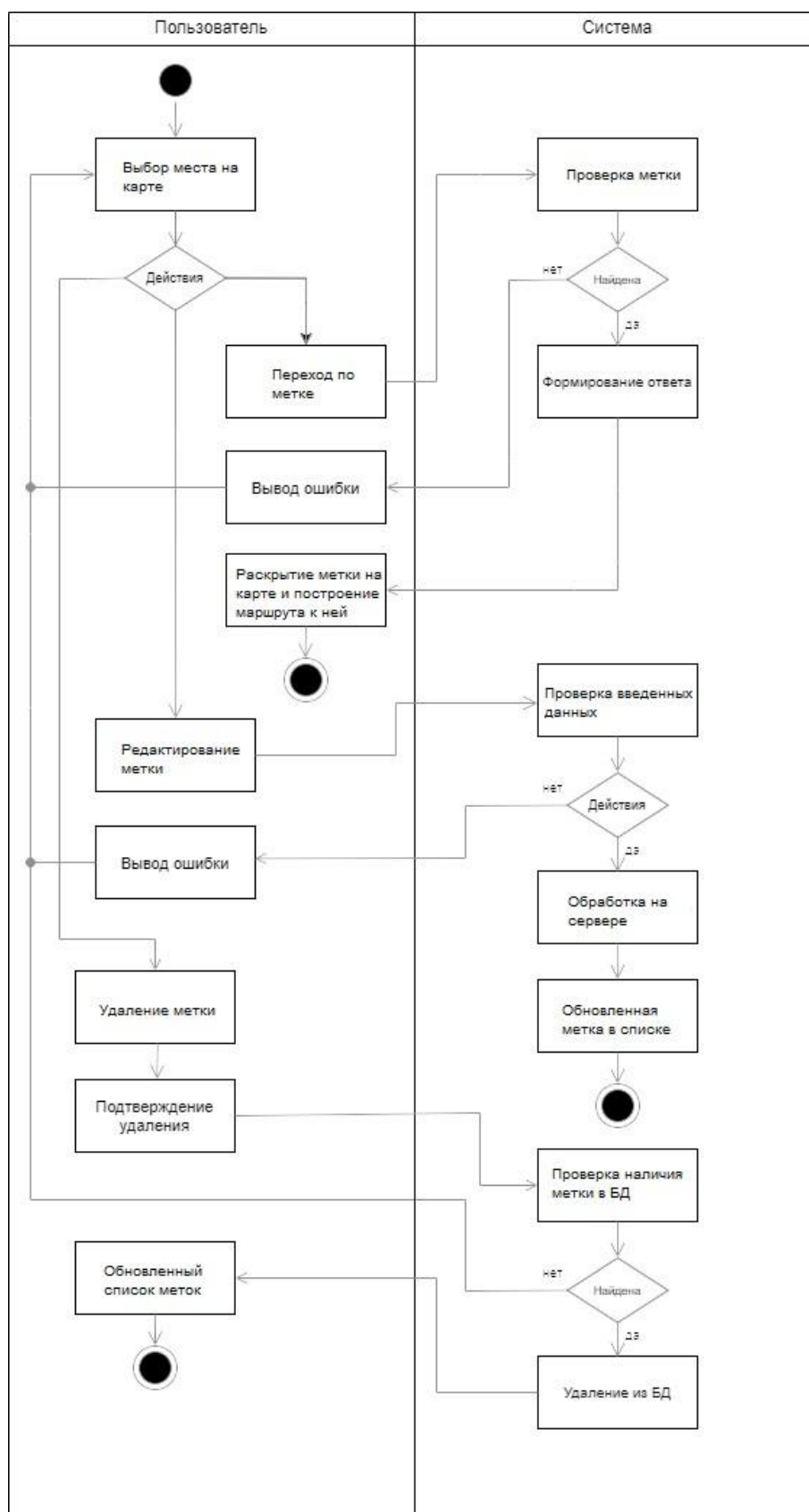


Рисунок 9 – UML диаграмма деятельности взаимодействия с одной меткой

4.2 Описание работы приложения

Пользователи совершают действия только на клиентской части приложения, написанной на HTML/CSS и используя JavaScript для реализации работы с картой. По ходу использования пользователь будет отправлять на сервер запросы на выполнение задач. Сервер работает с нашей БД, может обращаться к ней для создания, удаления, изменения содержимого в них. При успешном запросе к БД сервер выполнит операцию и вернёт ответ в нужном нам формате. При возникновении ошибок, сервер прерывает операцию и выводит информацию об ошибке.

Все выполняемые действия как пользователя с сервером изображены на диаграмме последовательности на рисунке 12.

Физическое расположение каждого из компонентов данной системы представлено на рисунке 13.

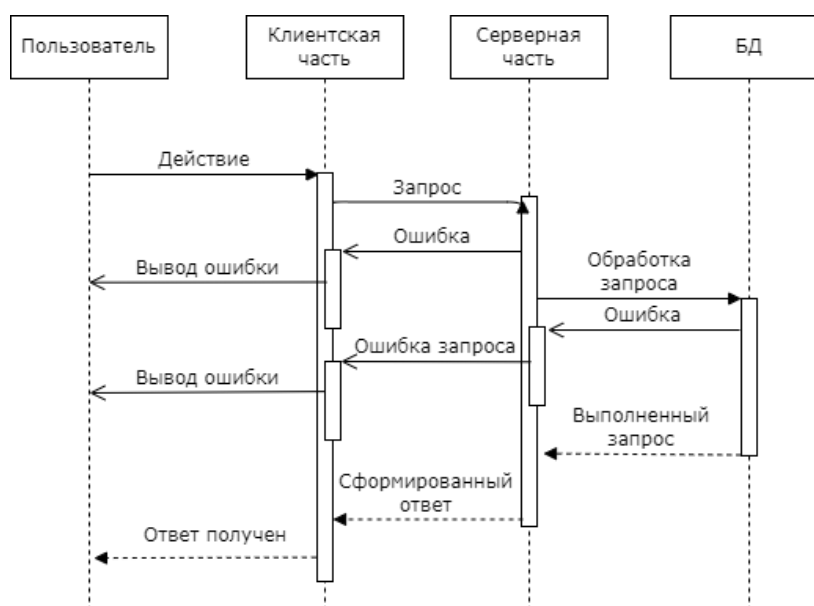


Рисунок 10 – UML диаграмма последовательности приложения

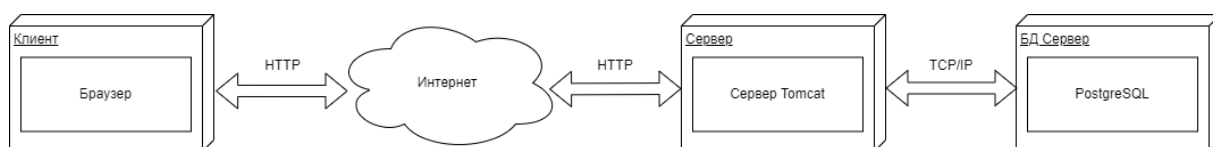


Рисунок 11 – UML диаграмма развертывания

4.3 Основная реализация приложения

Так как мы работаем с фреймворком Spring он сам выполняет большую часть работы по конфигурации приложения, всё что нам остаётся это правильно его использовать. Условное его можно поделить на несколько пакетов:

- configuration;
- controller;
- service;
- entity;
- repository;

Configuration – набор классов которые содержат конфигурацию нашего веб-слоя. С помощью них мы настраиваем компоненты и бины, настраиваем наши контроллеры. Помимо этого, в них мы указываем пути к статическим ресурсам, таким как CSS, JavaScript и изображения.

Controller – являются центральным звеном, которое обеспечивает связь между различными компонентами веб – приложения. Классы данного компонента связывают запросы клиента с бизнес-логикой и представлениями.

Service – набор классов, которые содержат в себе всю бизнес – логику тем самым отделяя её от контроллеров и репозиторий, помимо этого они управляют транзакциями к БД обращаясь для этого к репозиториям.

Entity – набор классов который играет ключевую роль при работе с БД, представляют собой структуру данных и соответствует таблицам в БД.

Repository – набор интерфейсов, которые определяют правила взаимодействия с БД предоставляя уровень абстракции между бизнес-логикой и данными. Управляют операциями создания, чтения, обновления и удаления в БД.

Посмотрим на конкретный пример реализации авторизации пользователя.

4.3.1 Configuration

В данном уровне содержится класс `WebSecurityConfig`, с помощью которого мы настраиваем безопасность нашего веб-приложения при помощи аннотаций и бинов для управления аутентификацией и авторизацией. Аннотация `@Configuration` указывает, что данный класс является конфигурационным и содержит определения бинов, аннотация `@EnableWebSecurity` включает поддержку безопасности Spring Security, а аннотация `EnableGlobalMethodSecurity (prePostEnabled = true)` включает поддержку аннотация безопасности на уровне методов. Сам класс `WebSecurityConfig` используется для настройки безопасности, он определяет как обрабатывать аутентификацию пользователей, какие страницы доступны без аутентификации, настройку формы входа и выхода, конфигурацию для шифрования паролей и интеграцию с сервисами для управления пользователями.

Листинг 1. Класс `WebSecurityConfig`:

```
@Autowired
protected UserService userService;

@Autowired
private PasswordEncoder passwordEncoder;

@Bean
public static PasswordEncoder getPasswordEncoder(){
    return new BCryptPasswordEncoder(8);
}

@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity
http) throws Exception {
    http
        .authorizeHttpRequests((requests) -> requests
            .requestMatchers("/", "/registration",
"/static/**", "/activate/*").permitAll()
            .anyRequest().authenticated()
        )
        .formLogin((form) -> form
            .loginPage("/login")
            .permitAll()
        )
    }
```

```

        .logout((logout) -> logout.permitAll());
        http.csrf().disable();

        return http.build();
    }

    @Autowired
    protected void configure(AuthenticationManagerBuilder auth)
    throws Exception {
        auth.userDetailsService(userService)
            .passwordEncoder(passwordEncoder);
    }

```

4.3.2 Controller

На уровне контроллеров за авторизацию и регистрацию отвечает класс `RegistrationController`. Контроллер обрабатывает все запросы, которые отправляются к серверу. В нем используются аннотации `@Controller` – указывающий что класс является контроллером Spring MVC, аннотация `@Autowired` которая внедряет зависимость `UserService` для управления логикой, аннотации `@GetMapping` и `@PostMapping` для обработки GET и POST запросов соответственно, аннотация `@Valid` для того чтобы провалидировать конкретный объект и аннотация `@PathVariable` для привязки переменной пути к параметру метода.

Метод `registration` обрабатывает GET-запрос, в котором возвращает имя страницы регистрации для отображения на экране.

Листинг 2. Метод `registration()`

```

@GetMapping("/registration")
public String registration()
{
    return "registration";
}

```

Метод `addUser` обрабатывает POST-запросы для добавления нового пользователя, проверяет совпадают ли пароли, наличие ошибок валидации и существует ли уже такой пользователь, если все проверки проходят успешно создаётся новый пользователь и нас перенаправляет на страницу логина.

Листинг 3. Метод `adduser()`:

					ПГУ1.090301.10.001 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @PostMapping("/registration")
        public String addUser(@Valid User user, BindingResult
bindingResult, Model model){
            if (user.getPassword() != null &&
!user.getPassword().equals(user.getPassword2()))
            {
                model.addAttribute("passwordError", "Password not
equals");
            }

            if (bindingResult.hasErrors()) {
                Map<String, String> errors =
ControllerUtils.getErrors(bindingResult);
                model.mergeAttributes(errors);
                return "registration";
            }

            if (!userService.addUser(user)) {
                model.addAttribute("usernameError", "User
exists!");
                return "registration";
            }

            return "redirect:/login";
        }

```

Метод activate обрабатывает GET-запрос для активации пользователя, используя UserService для активации по коду, добавляет сообщение с результатом в модель. После срабатывания перебрасывает нас на страницу входа login.

Листинг 4. Метод activate():

```

@GetMapping("/activate/{code}")
public String activate(Model model, @PathVariable String
code){
    boolean isActivated = userService.activateUser(code);

    if (isActivated) {
        model.addAttribute("message", "User activated
succesfully");
    }
    else{
        model.addAttribute("message", "Activation code is
not found");
    }

    return "login";
}

```

4.3.3 Service

Сервисные классы содержат всю бизнес-логику приложения и являются посредниками между контроллерами и репозиториями. Аннотация `@Service` – указывает, что этот класс является сервисом, аннотация `@Autowired` – автоматически внедряет зависимости в этот класс. В данном слое будет рассмотрен класс `UserService` и все методы участвующие в процессе авторизации и регистрации.

Метод `loadUserByUsername` является реализацией метода из интерфейса `UserDetailsService` отвечает за аутентификацию пользователя, он загружает пользователя по его имени.

Листинг 5. `loadUserByUsername()`:

```
@Override
public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
    return userRepo.findByUsername(username);
}
```

Метод `addUser` принимает объект `User`, добавляет нового пользователя в систему. Проверяет существует ли уже такой пользователь в БД, устанавливает параметры, сохраняет пользователя и отправляет письмо с кодом активации нового пользователя.

Листинг 6. `addUser()`:

```
public boolean addUser(User user) {
    User userfromDB = userRepo.findByUsername(user.getUsername());
    if (userfromDB != null) {
        return false;
    }
    user.setActive(true);
    user.setRoles(Collections.singleton(Role.USER));
    user.setActivationCode(UUID.randomUUID().toString());

    user.setPassword(passwordEncoder.encode(user.getPassword()));
    userRepo.save(user);
    sendMessage(user);
    return true;
}
```

Метод `sendMessage` формирует и отправляет письмо с активационным кодом, на указанную пользователем почту. Используя метод интерфейса `mailSender` мы отправляем письмо.

Листинг 7. `sendMessage()`:

```
private void sendMessage(User user) {
    if (!StringUtils.isEmpty(user.getEmail())) {
        String message = String.format(
            "Hello, %s! \n" +
            "Welcome to Food Map. Please, visit
next link: http://localhost:8080/activate/%s",
            user.getUsername(),
            user.getActivationCode()
        );
        mailSender.send(user.getEmail(), "Activation code",
message);
    }
}
```

Метод `activateUser` ищет пользователя по коду активации, если находит то код удаляется, а пользователь становится активированный.

Листинг 8. `activateUser()`:

```
public boolean activateUser(String code) {
    User user = userRepo.findByActivationCode(code);
    if (user == null) {
        return false;
    }
    user.setActivationCode(null);
    userRepo.save(user);
    return true;
}
```

Метод `findAll` возвращает список всех пользователей

Листинг 9. `findAll()`:

```
public List<User> findAll() {
    return userRepo.findAll();
}
```

Метод `saveUser` принимает объект пользователя, имя пользователя и ключи соответствующие ролям пользователя. Сам метод устанавливает новые имя пользователя и роли в соответствии с принятыми значениями.

Листинг 10. `saveUser()`:

					ПГУ1.090301.10.001 ПЗ	Лист
						29
Изм.	Лист	№ докум.	Подпись	Дата		

```

    public void saveUser(User user, String username, Map<String,
String> form) {
        user.setUsername(username);
        Set<String> roles = Arrays.stream(Role.values())
            .map(Role::name)
            .collect(Collectors.toSet());
        user.getRoles().clear();
        for (String key : form.keySet()) {
            if (roles.contains(key)) {
                user.getRoles().add(Role.valueOf(key));
            }
        }
        userRepo.save(user);
    }
}

```

Метод updateProfile нужен для обновление профиля пользователя смотрит обновлена ли почта пользователя, введен ли новый пароль, если да, отправляет новый код активации и сохраняет обновленного пользователя в БД.

Листинг 11. updateProfile():

```

    public void updateProfile(User user, String password, String
email) {
        String userEmail = user.getEmail();
        boolean isEmailChanged = (email != null &&
!email.equals(userEmail)) ||
            (userEmail != null && !userEmail.equals(email));
        if (isEmailChanged) {
            user.setEmail(email);
            if (!StringUtils.isEmpty(email)) {
                user.setActivationCode(UUID.randomUUID().toString());
            }
        }
        if (!StringUtils.isEmpty(password)) {
            user.setPassword(passwordEncoder.encode(password));
        }
        userRepo.save(user);
        if (isEmailChanged) {
            sendMessage(user);
        }
    }
}

```

4.3.4 Repository

Слой репозитория управляет доступом к данным пользователей и управляет ими в БД. Рассмотрим класс UserRepo, он является репозиторием Spring расширяя интерфейс JpaRepository, поэтому Spring может управлять его созданием и внедрением. Наследуясь от JpaRepository мы получаем набор

стандартных операций для работы с БД, помимо этого содержит два метода. Первый `findByUsername()` находит пользователя по его имени, Spring сам реализует этот метод, второй `findByActivationCode()` находит пользователя по его коду активации, этот метод также автоматически реализуется Spring.

Листинг 12. Интерфейс UserRepo:

```
public interface UserRepo extends JpaRepository<User, Long> {
    User findByUsername(String username);
    User findByActivationCode(String code);
}
```

4.3.5 Entity

Этот слой представляет собой сущности в БД. Рассмотрим класс `User` которая представляет собой сущность пользователя. Аннотация `@Entity` указывает, что класс является сущностью, аннотация `@Table( name= "usr")` указывает имя таблицы к которой будет привязана это сущность и имя этой таблицы, аннотация `@Id` обозначает что поле `id` будет первичным ключом таблицы, аннотация `@GeneratedValue` указывает как будет меняться значение `id` в нашем случае это автоинкремент, аннотация `@NotBlank` указывает что данное поле не может быть пустым, аннотация `@Transient` указывает что данное поле не должно сохраняться в БД, аннотация `@Email` говорит что поле должно содержать корректный адрес почты, аннотация `@ElementCollection` указывает что данное поле является коллекцией элементов, которые хранятся в отдельной таблице, но принадлежат одной сущности, аннотация `@CollectionTable` указывает имя таблицы в которой хранятся элементы коллекции и имя строки для связи с текущей таблицей, аннотация `@Enumerated(EnumType.String)` указывает, что значения должны храниться как строки в БД.

Листинг 13. Класс User:

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;
@NotBlank(message = "username cannot be empty")
```

```

private String username;

public String getPassword2() {
    return password2;
}
public void setPassword2(String password2) {
    this.password2 = password2;
}
@NotBlank(message = "password cannot be empty")
private String password;
@Transient
@NotBlank(message = "password confirm cannot be empty")
private String password2;
private boolean active;
>Email(message = "email is not correct")
@NotBlank(message = "username cannot be empty")
private String email;
private String activationCode;
@ElementCollection(targetClass = Role.class, fetch =
FetchType.EAGER)
@CollectionTable(name = "user_role", joinColumns =
@JoinColumn(name = "user_id"))
@Enumerated(EnumType.STRING)
private Set<Role> roles;
public String getEmail() {
    return email;
}
public boolean isActive() {
    return active;
}
public void setActive(boolean active) {
    this.active = active;
}
@Override
public boolean isAccountNonExpired() {
    return true;
}
@Override
public boolean isAccountNonLocked() {
    return true;
}
@Override
public boolean isCredentialsNonExpired() {
    return true;
}
@Override
public boolean isEnabled() {
    return isActive();
}
@Override

```

```

public Collection<? extends GrantedAuthority>
getAuthorities() {
    return getRoles();
}

```

Чтобы совсем не запутаться в классах и их методах посмотрим на рисунок 14, на котором представлена диаграмма классов.

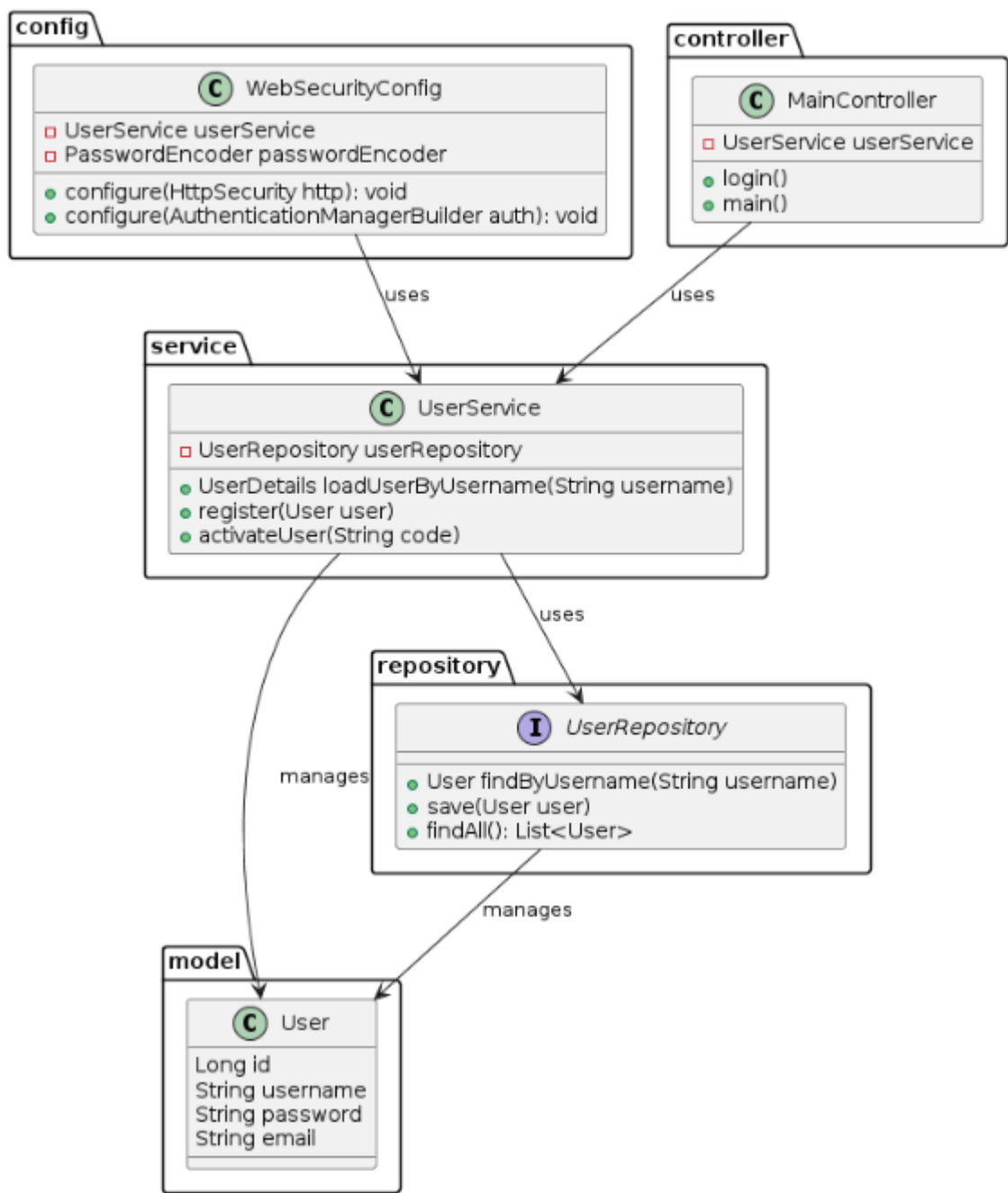


Рисунок 12 - Диаграмма классов для авторизации

Теперь благодаря такой реализации приложение можно легко тестировать и расширять.

5 Тестирование

Для проверки работоспособности и выявления сбоев в программе необходимо провести тестирование веб-приложения «Фуд-карта».

В данном пункте будет проведено юнит-тестирование программы. Юнит-тесты предполагают тестирование каждого модуля программы, каждый отдельный кусок кода будет проверяться на правильность работы.

Для этого к каждому модулю программы будут написаны конкретные тесты на языке Java. Это нужно для проверки их работоспособности и при сбое программы в какой-то ситуации мы будем видеть на каком из тестов она отвалилась, что в будущем при её доработке может помочь нам.

5.1 Модульное тестирование

В предыдущем разделе были описаны методы и классы, участвующие при авторизации или регистрации пользователя. В данном пункте будут написаны тест-кейсы для каждого из этих методов.

Таблица 1 – Модульное тестирование сервиса

UserService		
Методы	Ожидаемый результат	Фактический результат
testaddUser()	Если пользователь добавился возвращает – True, если нет – False	Совпадает с ожидаемым
testFindAll()	Возвращает размер списка пользователей	Совпадает с ожидаемым
testSaveUser()	Сохраняет пользователя с новыми ролями и именем	Совпадает с ожидаемым

UserService		
testUpdateUser()	Сохраняет пользователя с обновленной почтой и никнеймом и отправляет уведомление об этом	Совпадает с ожидаемым
testActivateUser()	Активирует аккаунт нового созданного пользователя	Совпадает с ожидаемым
testLoadByUsername()	Ищет пользователя по никнейму	Совпадет с ожидаемым

Запускаем тесты в IDE и смотрим результат их выполнения. Как видно на рисунке 15 тесты выполнены успешно.

```

✓ UserServiceTest (com.example.testdiplom.se 1 sec 238 ms
  ✓ testLoadUserByUsername() 1 sec 179 ms
  ✓ testAddUser() 20 ms
  ✓ testFindAll() 5 ms
  ✓ testUpdateUser() 22 ms
  ✓ testActivateUser() 6 ms
  ✓ testSaveuUser() 6 ms

```

Рисунок 13 – Запущенные тесты UserServiceTest

Сервис был протестирован теперь можно протестировать следующие модули программы такие как контроллеры, конфигурации, репозитории и сущности. В сервисе хранится бизнес логика, перейдем к контроллерам, обрабатывающим запросы

Таблица 2 – Модульное тестирование контроллера

RegistrationController		
Методы	Ожидаемый результат	Фактический

RegistrationController		
		результат
registration()	Запрос будет удачный и вернет представление registration	Совпадает с ожидаемым
addUser()	Добавляет пользователя, проверяет существует ли уже такой пользователь	Совпадает с ожидаемым
activate()	Проверяет активацию успешную/неуспешную и перебрасывает на представление login	Совпадает с ожидаемым

Запускаем тесты в IDE и смотрим результат их выполнения. Как видно на рисунке 16 тесты контроллера выполнились успешно.

```

✓ RegistrationControllerTest example.testdiplom.se 1 sec 203 ms
  ✓ testAddUser() 1 sec 185 ms
  ✓ testActivate() 7 ms
  ✓ testRegistration() 11 ms

```

Рисунок 146 – Запущенные тесты TestRegistrationController

Теперь напишем модульные тесты для конфигураций безопасности. Отвечающих за шифрование пароля, доступ неавторизованных пользователей к отображениям.

Таблица 3 – Модульное тестирование конфигурации

WebSecurityConfig		
Методы	Ожидаемый результат	Фактический

		результат
getPasswordEncoder()	Проверяет работает ли шифрователь пароля корректно	Совпадает с ожидаемым
SecurityFilterChain()	Проверяет доступ для пользователей с аутентификацией и без аутентификации	Совпадает с ожидаемым
LoginTest()	Проверяет доступно ли представление login	Совпадает с ожидаемым

Запускаем тесты в IDE и смотрим результат их выполнения. Как видно на рисунке 17 тесты выполнились успешно.

```

✓ WebSecurityConfigTest (com.example.testdiplom.c 751 ms
✓ passwordEncoderBean() 399 ms
✓ givenCustomLoginPage_whenGetLogin_thenOk 232 ms
✓ givenPublicUrl_whenGetRequest_thenOk() 120 ms

```

Рисунок 157 – Запущенные тесты WebSecurityConfigTest

Протестируем сущность пользователя.

Таблица 4 – Модульное тестирование сущности

User		
Методы	Ожидаемый результат	Фактический результат
Getters	Проверяет геттеры	Совпадает с ожидаемым
Setters	Проверяет сеттеры	Совпадает с ожидаемым

User		
testUserDetails	Проверяет методы интерфейса userdetails	Совпадает с ожидаемым

✓ UserTest (com.example.testdiplom.domain)	131 ms
✓ testGettersAndSetters()	17 ms
✓ testUserDetailsMethods()	1 ms
✓ testValidUser()	104 ms
✓ testValidationConstraints()	9 ms

Рисунок 168 – Запущенные тесты UserTest

Теперь протестируем репозитории и их методы.

Таблица 4 – Модульное тестирование репозитория

UserRepo		
Методы	Ожидаемый результат	Фактический результат
findByUsername	Проверяет геттеры	Совпадает с ожидаемым
findByActivationCode	Проверяет сеттеры	Совпадает с ожидаемым
testUserDetails	Проверяет методы интерфейса userdetails	Совпадает с ожидаемым

✓ UserRepoTest (com.example.testdiplom.ser	1 sec 710 ms
✓ testNotFound()	1 sec 694 ms
✓ testFindByUsername()	6 ms
✓ testfindByActivationCode()	10 ms

Рисунок 177 – Запущенные тесты UserTest

5.2 Вывод

Благодаря юнит-тестированию программы были выявлены части программы, работающие некорректно. После все найденные на данном этапе проблемы и задачи были исправлены и обработаны, это позволило сделать более надежное и отказоустойчивое приложение.

					ПГУ1.090301.10.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

6 Безопасность жизнедеятельности

Разработчики и люди чью профессия тесно с работой на компьютере проводят много времени перед ним, поэтому безопасность при работе за компьютером очень важна. Несоблюдение мер предосторожности может привести к множеству проблем со здоровьем, включая напряжение глаз, плохую осанку, повышенный риск синдрома запястного канала и других заболеваний.

В этом разделе будут рассмотрены основные аспекты безопасности жизнедеятельности и рекомендации пользователей ПК.

6.1 Требования безопасности при работе с ПК и рабочему месту

Требования безопасности жизнедеятельности при работе с ПК охватывают различные аспекты, при взаимодействии с техникой человек может подвергаться разным вредным воздействиям. И для их предотвращение необходима правильная организация рабочего места. При организации рабочего места оно должно быть хорошо освещено. Помещения, предусмотренные для работы с ПК, обязательно должны иметь искусственное и естественное освещение, но не слишком яркое.

Также важна мебель, на которой человек проводит большую часть времени, рабочее кресло должно быть регулируемым по высоте, с поддержкой поясницы и возможностью регулировки наклона спинки. Стол должен обеспечивать достаточно места для комфортного расположения оборудования и рук, а самое рабочее место должно быть организовано с учетом эргономических принципов, чтобы минимизировать напряжение мышц и суставов человека. Но постоянно работать тоже изводит организм, поэтому необходимо делать перерывы. В перерыве хорошо делать разминку для глаз, а так же простые физические упражнения. Перед работой с компьютером руки должны быть

чистыми, поэтому всегда следите за чистотой периферии, пыль и грязь необходимо вытереть с клавиатуры, мышки и монитора.

Правила работы за ПК включают правильную осанку, настройку монитора под ваш глаз и использование защитных средств. Пользователь должен сидеть с прямой спиной, с расслабленными плечами и ногами, яркость нашего монитора должна быть настроена под комфортный уровень, чтобы не вызвать перенапряжения глаз, а при длительной работе за компьютером рекомендуется использовать очки с теплым фильтром, нейтрализующим синий спектр, для более комфортной работы за ПК.

При работе также рекомендуется чередовать различные виды задач, чтобы избежать перенапряжения и стресса. Лучше планировать задачи, ставить им приоритеты и заниматься тайм-менеджментом.

В целях безопасности следует не пользоваться ПК при наличии внешних повреждений. Необходимо следить на своем рабочем месте за исправным состоянием техники и не оставлять ее включенной на долго без присмотра людей. Состояние кабелей и соединений следует регулярно проверять. Не используйте поврежденные провода. А оборудование в свою очередь должно быть правильно подключено к электросети, через сертифицированные устройства с защитой от перепадов напряжения.

Таким образом, следуя данным мерам безопасности работа за ПК будет максимально комфортной и безопасной для как для обычного пользователя, так и для разработчиков.

6.2 Требования к освещению

Недостаточное или неправильное освещение может привести к различным проблемам со здоровьем, поэтому важно соблюдать нормы освещения. Неправильное освещение может быть причиной повышенной утомляемости от перенапряжения глаз, что приводит к снижению продуктивности. Однако

					ПГУ1.090301.10.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

усталость глаз и головные боли могут быть вызваны также слишком чрезмерным освещением.

Важно понимать, что от освещения в комнате зависит общая яркость и равномерность распределения света в помещении. Возникают проблемы, если свет распределяется неправильно. Чтобы предотвратить эти проблемы, следует использовать искусственное и естественное освещение.

Искусственное освещение помещений для работы ПК должно обеспечиваться системой общего равномерного освещения, которая может включать люстры и потолочные светильники. При необходимости можно установить лампы местного освещения. В качестве источников света можно использовать например лампы накаливания. Лампочки синего цвета не подходят так как от них быстрее устают глаза. Так же в одном помещении нельзя использовать люминесцентные лампы и лампы накаливания. Попадание ламп в глаза и появление бликов на поверхности экранов мониторов можно предотвратить с помощью экранирующих решеток или рассеивателей.

Рекомендации по освещению при работе с ПК:

- освещенность составляла 300 люкс;
- яркость рабочего стола не превышала 500 люкс;
- на рабочем месте следует использовать естественное освещение.

Соблюдение всех этих мер необходимо для комфортной работы за персональным компьютером.

7 Стартап проект

Данный продукт предназначен для туристов, семей, групп друзей планирующих экскурсии или отдых в собственном или другом городе. Ведение собственных записей мест, которые заинтересовали вас могут однозначно заинтересовать и других людей, не только рестораны и кафе, это могут быть даже самые неоднозначные места, лавочка рядом с ларьком открывающая невероятный вид, это не оставит равнодушным любого туриста или человека, что в ежедневной суете не замечал красот собственного города.

7.1 Описание продукта

Целевая аудитория данного приложения, это туристы, путешественники, семьи, группы друзей, жители, интересующиеся собственным городом, рестораторы и владельцы как небольших предприятий, так и крупного бизнеса.

Основные функции приложения, это создание персонализированных меток и получение библиотек из меток интересных конкретному пользователю, он может редактировать метку как душе угодно, загружать изображения, давать собственные наименования, ставить оценку, опираясь на собственные ощущения.

7.2 Ценностное предложение

Для обычных пользователей данное приложения имеет много достоинств. Удобство в планировании маршрутов с учетом собственных предпочтений, полная свобода в выборе маршрута, доступ к качественной информации от картографического сервиса, возможность построения маршрутов по собственным меткам. Для предпринимателей это возможность зарекомендовать себя клиентам, получить рекламу, следственно дополнительный трафик и прибыль.

					ПГУ1.090301.10.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

7.3 Потоки доходов

Существуют разные модели доходов, которые могут приносить доход нашему приложению:

- реклама. Местные предприниматели могут делиться с нами информацией о своих заведениях и акциях которые можно было бы создавать и вписывать на их метках;
- партнерство. Платные партнерские программы с большими сетями ресторанов, другими крупными компаниями;
- связь с рестораторами. Для возможности мгновенного заказа в нашем приложении;
- платные подписки на сервис. Возможность ограничивая количества меток до определенного количества без подписки.

7.4 Ключевые ресурсы

Ключевые ресурсы необходимые при разработке этого стартап – проекта:

- людские ресурсы;
- поиск средств;
- поддержка продукта;
- партнерства с рестораторами и предпринимателями.

Для того чтобы создать приложение, соответствующее всем требованиям и поддерживать его функциональность необходима подходящая команда, состоящая из разработчиков, маркетологов, бухгалтеров и менеджеров.

7.5 Ключевые виды деятельности

Виды деятельности необходимые для функционирования продукта:

- разработка и поддержка приложения: Анализ пользовательского опыта и постоянное улучшение функционала для его улучшения, исправление ошибок и проблем в процессе;

					ПГУ1.090301.10.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

- Создание контента: Производство и обновление фото, видео, текстовых и аудиоматериалов;
- Маркетинг: Запуск рекламных кампаний, работа в социальных сетях, организация мероприятий для привлечения пользователей и инвесторов;

7.6 Оценка затрат

Оценка всех затрат необходимых для создания и полного запуска приложения:

- разработка и поддержка. Затраты на дизайн, разработку - 150000р., затраты на поддержку работоспособности -50000р./мес., затраты на аренду оборудования – 20000р./мес., затраты на оборудования для разработчиков – 150000р., затраты на аренду помещения – 30000 р./мес.;
- маркетинг. Рекламные кампании и продвижение в социальных сетях - 50 000р./мес., организация мероприятий для аудитории – 30000р./мес.;
- создание контента: Разработка маршрутов, создание фото/видео сопровождения, написание текстов - 50 000р. Поддержка обновления и добавления актуального контента - 15 000р./мес..

Необходимые первоначальные затраты: 545 000р.

Необходимые ежемесячные затраты: 195 000р.

7.7 Оценка доходов

Модели монетизации:

- реклама. Средний доход с рекламы составляет 25р. с одного активного пользователя в месяц;
- партнерства. Доход от партнерских программ с ресторанами и другими заведениями составляет около 50 000р. в месяц. Из расчета наличия 5 партнеров, стоимость размещения рекламы каждого партнера - 10000р./мес.;
- сотрудничество с ресторанами. Комиссия за заказы – 25 000р. в месяц;

– платные подписки на сервис увеличивающие возможности пользователей 150р./мес. с пользователя.

Оценка доходов при 10000 активных пользователях:

– 10% пользователей покупают премиум подписку:
1000 пользователей * 150 = 150 000р. в месяц;

– Реклама: 10,000 пользователей * 25 = 25 000р. в месяц;

– Сотрудничество с предприятиями: 50 000р. в месяц;

– Коммиссия с заказов: 25 000 в месяц.

Итого ежемесячный доход: 250 000р.

7.8 Расчет срока окупаемости

Для расчёта срока окупаемости необходимо рассчитать сколько средств необходимо на первоначальные взносы, ежемесячные расходы, ежемесячные доходы и рассчитать прибыль за месяц.

Прибыль равна разности между общим доходом и общим расходом.

Ежемесячная прибыль: 250 000 – 195 000 = 55000р.

Для расчёта срока нам необходимо посчитать как быстро мы погасим первоначальную сумму исходя из нашей ежемесячной прибыли.

Срок окупаемости приложения: 545000 / 55000 = 9.9 месяцев

Таким образом, при указанных условиях, приложение окупится менее чем через 10 месяцев.

Заключение

Результатом данной выпускной квалификационной работы можно считать успешно разработанное веб – приложение «Фуд карта». Были выполнены следующие задачи: анализ актуальности темы и аналогов, выбор инструментов разработки, анализ архитектуры веб-приложения, разработка и тестирование приложения, составление бизнес-плана проекта.

Проведенные тестирование и оценка показали, что веб – приложение «Фуд карта» работает и соответствует всем поставленным целям. В ходе разработки были улучшены навыки работы с языком запросов SQL и приобретены навыки работы с языком JavaScript.

В дальнейшем приложение можно было бы усовершенствовать улучшив визуально интерфейс и добавив многопользовательские функции.

					ПГУ1.090301.10.001 ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. ru.wikipedia.org: [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/Java> (дата обращения: 25.04.2024)
2. habr.com: [Электронный ресурс]. URL: <https://habr.com/ru/articles/336816/> (дата обращения: 29.04.2024)
3. yandex.ru/dev/: [Электронный ресурс]. URL: <https://yandex.ru/dev/jsapi30/doc/ru/> (дата обращения: 1.05.2024)
4. habr.com: [Электронный ресурс]. URL: <https://habr.com/ru/articles/689766/> (дата обращения: 5.05.2024)
5. spring.io: [Электронный ресурс]. URL: <https://spring.io/guides/gs/rest-service> (дата обращения: 6.05.2024)
6. w3school.com: [Электронный ресурс]. URL: https://www.w3schools.com/sql/sql_distinct.asp (дата обращения: 10.05.2024)
7. spring.io/guides: [Электронный ресурс]. URL: <https://spring.io/guides/gs/testing-web> (дата обращения: 13.05.2024)
8. Санитарно-гигиенические рекомендации для организации рабочего места пользователя персональным компьютером. [Электронный ресурс]. URL: <https://sakhgig.ru/main/info/664-sanitarno-gigienicheskie-rekomendacii-dlya-organizacii-rabochego-mesta-polzovatelya-personalnym-kompyuterom.html> (дата обращения 20.05.2024)

Приложение А

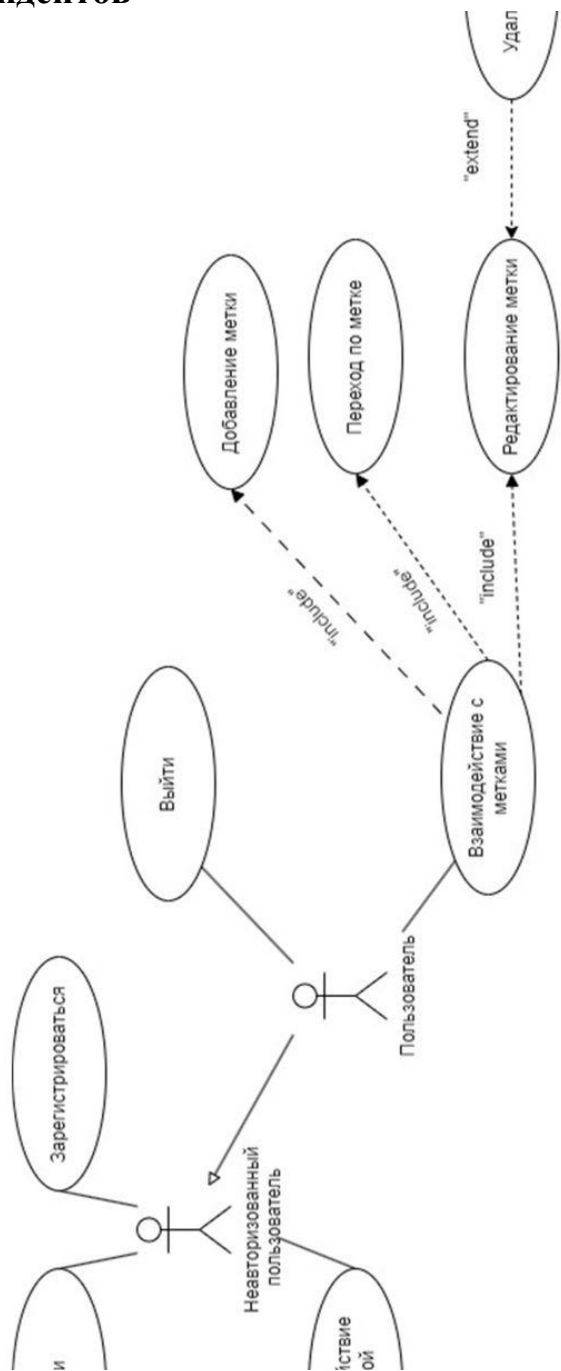
(Обязательное)

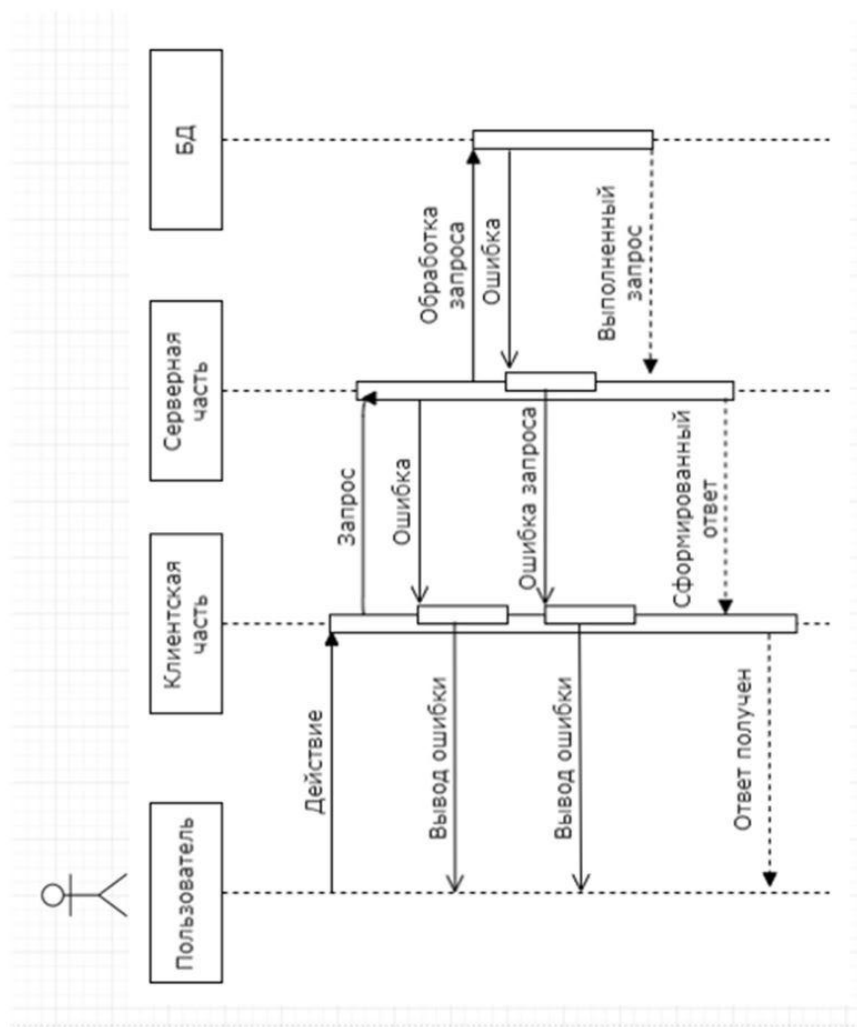
Стартап-проект. Веб-приложение «Фуд карта»

UML-диаграммы

					ПГУ1.090301.10.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1-Диаграмма прецендентов





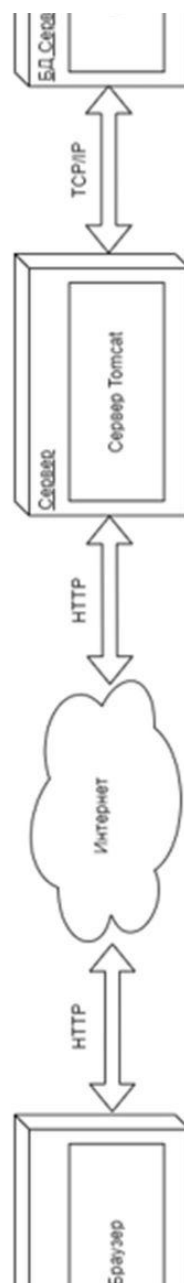
Приложение А.4-Диаграмма развертывания

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.10.001 ПЗ

Лист

53



Приложение А.5-Диаграмма классов

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.10.001 ПЗ

Лист

54


```

    public WebSecurityConfig(CustomUserDetailsService
userDetailsService){
        this.userDetailsService = userDetailsService;
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .antMatchers("/", "/signup", "/index/**",
"/find").permitAll()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .loginPage("/login")
            .usernameParameter("email")
            .passwordParameter("password")
            .failureUrl("/login?error=true")
            .permitAll()
            .and()
            .logout()
            .logoutUrl("/logout")
            .logoutSuccessUrl("/")
            .invalidateHttpSession(true)
            .deleteCookies("JSESSIONID")
            .permitAll();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
        auth
            .userDetailsService(userDetailsService)
            .passwordEncoder(passwordEncoder());
    }

    @Bean
    public PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder(8);
    }
}

```

MvcConfig.java

```

@Configuration
public class MvcConfig implements WebMvcConfigurer {

    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/login").setViewName("login");
    }
}

```

Role.java

					ПГУ1.090301.10.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

```

public enum Role implements GrantedAuthority {
    USER;

    @Override
    public String getAuthority() {
        return name();
    }
}

```

UserRepository.java

```

public interface UserRepository extends JpaRepository<User, Long> {
    User findUserByEmail(String email);
}

```

MarkerRepository.java

```

public interface BookRepository extends JpaRepository<Book, Long> {
    List<Marker> findByMarkerName(String bookName);
    List<Marker> findByUser(User user);
}

```

UserService.java

```

@Service
public class UserService {

    @Autowired
    private final UserRepository userRepository;
    @Autowired
    private final PasswordEncoder passwordEncoder;

    public UserService(UserRepository userRepository,
                       PasswordEncoder passwordEncoder) {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
    }

    public User findUser(Long id) {
        return userRepository.findById(id).orElse(null);
    }

    public User findUserByEmail(String email) {
        return userRepository.findUserByEmail(email);
    }

    public boolean addUser(User user) {
        String email = user.getEmail();

        if (userRepository.findUserByEmail(email) != null) {
            return false;
        }
        user.setRoles(Collections.singleton(Role.USER));
        user.setPassword(passwordEncoder.encode(user.getPassword()));
    }
}

```

```

        userRepository.save(user);

        return true;
    }

    public void updateUser(Principal principal, User user){
        User oldUser = getUserByPrincipal(principal);

        if(user.getFirstName() != null ||
!user.getFirstName().equals(oldUser.getFirstName())){
            oldUser.setFirstName(user.getFirstName());
        }

        if(user.getLastName() != null ||
!user.getLastName().equals(oldUser.getLastName())){
            oldUser.setLastName(user.getLastName());
        }

        if(user.getPassword() != null){

oldUser.setPassword(passwordEncoder.encode(user.getPassword()));
        }

        userRepository.save(oldUser);
    }

    public void deleteUser(Long id){
        userRepository.deleteById(id);
    }

    public User getUserByPrincipal(Principal principal){
        if(principal == null) return new User();
        return userRepository.findUserByEmail(principal.getName());
    }
}

```

ControllerUtils.java

```

public class ControllerUtils {

    static    Map<String,    String>    getErrors(BindingResult
bindingResult){

        Collector<FieldError, ?, Map<String, String>> collector
= Collectors.toMap(

            fieldError -> fieldError.getField() + "Error",
            FieldError::getDefaultMessage

        );

        return
bindingResult.getFieldErrors().stream().collect(collector);
    }
}

```

```
}
```

MainController.java

```
@Controller
public class MainController {
    private final MarkerRepo markerRepo;

    public MainController(MarkerRepo markerRepo) {
        this.markerRepo = markerRepo;
    }

    @Value("${upload.path}")
    private String uploadPath;
    @GetMapping("/")
    public String greeting() {
        return "hello";
    }

    @GetMapping("/main")
    public String newmain(Map<String, Object> model)
    {
        Iterable<Message> messages = markerRepo.findAll();

        model.put("messages", messages);

        return "main";
    }
    @PostMapping("/main")
    public String add(
        @AuthenticationPrincipal User user,
        @Valid Message message,
        BindingResult bindingResult,
        Model model,
        @RequestParam("file")MultipartFile file
    ) throws IOException {
        message.setAuthor(user);
        if(bindingResult.hasErrors()){
            Map<String, String> errorsMap =
ControllerUtils.getErrors(bindingResult);
            model.mergeAttributes(errorsMap);
            model.addAttribute("message", message);
        }else {
            if (file != null &&
!file.getOriginalFilename().isEmpty()) {
                File uploadDir = new File(uploadPath);
                if (!uploadDir.exists()) {
                    uploadDir.mkdir();
                }
            }
        }
    }
}
```

Изм.	Лист	№ докум.	Подпись	Дата

```

        }
        String uuidFile = UUID.randomUUID().toString();

        String resultFilename = uuidFile + "." +
file.getOriginalFilename();

        file.transferTo(new File(uploadPath + "/" +
resultFilename));

        message.setFilename(resultFilename);

    }
    model.addAttribute("message", null);
    markerRepo.save(message);
}

Iterable<Message> messages = markerRepo.findAll();

model.addAttribute("messages", messages);

return "main";
}

@PostMapping("filter")
public String filter(@RequestParam String filter,
Map<String, Object> model)
{
    Iterable<Message> messages;
    if(filter != null && !filter.isEmpty())
    {
        messages = markerRepo.findByName(filter);
    }
    else{
        messages = markerRepo.findAll();
    }
    model.put("messages", messages);
    return "main";
}
}

```

Файл «RegistrationController.java»

```

@Controller
public class RegistrationController {
    @Autowired

```

					ПГУ1.090301.10.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

```

private UserService userService;

@GetMapping("/registration")
public String registration()
{
    return "registration";
}

@PostMapping("/registration")
public String addUser(@Valid User user, BindingResult
bindingResult, Model model){
    if(user.getPassword() != null &&
!user.getPassword().equals(user.getPassword2()))
    {
        model.addAttribute("passwordError", "Password not
equals");
    }

    if(bindingResult.hasErrors()){
        Map<String, String> errors =
ControllerUtils.getErrors(bindingResult);
        model.mergeAttributes(errors);
        return "registration";
    }

    if(!userService.addUser(user)){
        model.addAttribute("usernameError", "User exists!");
        return "registration";
    }

    return "redirect:/login";
}

@GetMapping("/activate/{code}")
public String activate(Model model, @PathVariable String
code){
    boolean isActivated = userService.activateUser(code);

    if(isActivated){
        model.addAttribute("message", "User activated
succesfully");
    }
    else{
        model.addAttribute("message", "Activation code is
not found");
    }
}

```

```

        return "login";
    }

}

```

Файл «UserController.java»

```

@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    private UserService userService;
    @PreAuthorize("hasAuthority('ADMIN')")
    @GetMapping
    public String userList(Model model){
        model.addAttribute("users", userService.findAll());
        return "userList";
    }
    @PreAuthorize("hasAuthority('ADMIN')")
    @GetMapping("/{user}")
    public String userEditForm(@PathVariable User user, Model
model){
        model.addAttribute("user", user);
        model.addAttribute("roles", Role.values());
        return "userEdit";
    }
    @PreAuthorize("hasAuthority('ADMIN')")
    @PostMapping
    public String userSave(
        @RequestParam String username,
        @RequestParam Map<String, String> form,
        @RequestParam("userId") User user
    ) {
        userService.saveUser(user, username, form);

        return "redirect:/user";
    }
    @GetMapping("profile")
    public String getProfile(Model model,
@AuthenticationPrincipal User user) {
        model.addAttribute("username", user.getUsername());
        model.addAttribute("email", user.getEmail());

        return "profile";
    }

    @PostMapping("profile")
    public String updateProfile(

```

					ПГУ1.090301.10.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @AuthenticationPrincipal User user,
        @RequestParam String password,
        @RequestParam String email
    ) {
        userService.updateProfile(user, password, email);

        return "redirect:/user/profile";
    }
}

```

Файл «Marker.java»

```

@Entity
public class Marker {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    @NotBlank(message = "Please fill the message")
    private String name;
    @Length(max = 255, message = "Email too long")
    private String email;
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "user_id")
    private User author;
    private String filename;

    public String getFilename() {
        return filename;
    }

    public void setFilename(String filename) {
        this.filename = filename;
    }

    public String getAuthorName(){
        return author != null ? author.getUsername():"<none>";
    }
    public Marker() {
    }
    public Marker(String name, String email, User user) {
        this.author = user;
        this.name = name;
        this.email = email;
    }
    public User getAuthor() {
        return author;
    }
}

```

```

    public void setAuthor(User author) {
        this.author = author;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}

```

Файл «User.java»

```

@Entity
@Table(name = "usr")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @NotBlank(message = "username cannot be empty")
    private String username;

    public String getPassword2() {
        return password2;
    }

    public void setPassword2(String password2) {

```

```

        this.password2 = password2;
    }

    @NotBlank(message = "password cannot be empty")
    private String password;
    @Transient
    @NotBlank(message = "password confirm cannot be empty")
    private String password2;
    private boolean active;
    @Email(message = "email is not correct")
    @NotBlank(message = "username cannot be empty")
    private String email;
    private String activationCode;

    @ElementCollection(targetClass = Role.class, fetch =
FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns =
@JoinColumn(name = "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getActivationCode() {
        return activationCode;
    }

    public void setActivationCode(String activationCode) {
        this.activationCode = activationCode;
    }
    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    public Long getId() {
        return id;
    }

```

```

public void setId(Long id) {
    this.id = id;
}

public String getUsername() {
    return username;
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return isActive();
}

public void setUsername(String username) {
    this.username = username;
}

@Override
public Collection<? extends GrantedAuthority>
getAuthorities() {
    return getRoles();
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

```

```

    public Set<Role> getRoles() {
        return roles;
    }

    public void setRoles(Set<Role> roles) {
        this.roles = roles;
    }
}

```

Файл «MarkerRepo.java»

```

public interface MarkerRepo extends CrudRepository<Marker,
Integer> {
    List<Marker> findByName(String name);
}

```

Файл «UserRepo.java»

```

public interface UserRepo extends JpaRepository<User, Long> {
    User findByUsername(String username);

    User findByActivationCode(String code);
}

```

Файл «MailSende.java»

```

@Service
public class MailSender {
    @Autowired
    private JavaMailSender mailSender;
    @Value("${spring.mail.username}")
    private String username;
    public void send(String emailto, String subject, String
message) {
        SimpleMailMessage mailMessage = new SimpleMailMessage();
        mailMessage.setFrom(username);
        mailMessage.setTo(emailto);
        mailMessage.setSubject(subject);
        mailMessage.setText(message);

        mailSender.send(mailMessage);
    }
}

```

```
}  
}
```

Файл «user.html»

```
<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
<title>MyBooks</title>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>  
<link rel="stylesheet"  
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.  
min.css">  
<link rel="stylesheet" type="text/css" href="/images/style.css">  
</head>  
<body>  
  
<header th:insert="blocks/header :: headerLogin(${user})"></header>  
  
<main class="main-container container px-3 mt-5 d-flex justify-  
content-center" style="width: 1200px">  
<div style="width: 400px">  
<h1 class="h3 mb-3 mt-5 fw-normal">Редактирование</h1>  
  
<div class="row mt-2 mb-5 d-flex align-items-center justify-content-  
center">  
<form method="post" th:action="@{/user/edit}">  
<input type="hidden" name="id" th:field="${user.id}">  
  
<div class="form-floating mt-2">  
<input type="text" class="form-control" name="firstName"  
id="firstName" th:value="${user.firstName}">  
<label for="firstName">Имя</label>  
</div>  
  
<div class="form-floating mt-2">  
<input type="text" class="form-control" name="lastName" id="lastName"  
th:value="${user.lastName}">  
<label for="lastName">Фамилия</label>  
</div>  
  
<div class="form-floating mt-2">  
<input type="email" class="form-control" name="email" id="email"  
th:value="${user.email}">  
<label for="email">Логин</label>  
</div>  
<div class="form-floating mt-2">  
<input type="password" class="form-control" name="password"  
id="password">  
<label for="password">Новый пароль</label>  
</div>  
  
<button class="w-100 btn btn-lg btn-outline-primary mt-2"  
type="submit">Изменить</button>  
</form>
```

					ПГУ1.090301.10.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

```

<form class="col" method="post" th:action="@{/user/delete}"
onsubmit="return confirm('Вы уверены, что хотите удалить аккаунт?')">
<button type="submit" class="w-100 btn btn-lg btn-outline-danger mt-
2">Удалить аккаунт</button>
</form>
</div>

</div>

</main>

<footer th:insert="blocks/footer :: footer"></footer>
</body>
</html>

```

Файл «singup.html»

```

<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>MyBooks</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ"
crossorigin="anonymous">
</head>
<body class="text-center">

<main class="mx-auto" style="width: 350px">
<form method="post" th:action="@{/signup}">

<h1 class="h3 mb-3 fw-normal">Регистрация</h1>

<div class="form-floating mt-2">
<input type="text" class="form-control" name="firstName"
id="firstName" placeholder="Имя">
<label for="firstName">Имя</label>
</div>

<div class="form-floating mt-2">
<input type="text" class="form-control" name="lastName" id="lastName"
placeholder="Фамилия">
<label for="lastName">Фамилия</label>
</div>

<div class="form-floating mt-2">
<input type="email" class="form-control" name="email" id="email"
placeholder="name@example.com">
<label for="email">Логин</label>
</div>

```

```

<div class="form-floating mt-2">
<input type="password" class="form-control" name="password"
id="password" placeholder="Пароль">
<label for="password">Пароль</label>
</div>

<button class="w-100 btn btn-lg btn-primary mt-2"
type="submit">Регистрация</button>
</form>

<div class="container" th:if="${error}">
<div class="alert alert-info mt-2" th:text="${error}"></div>
</div>
</main>

</body>
</html>

```

Файл «login.html»

```

<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>MyBooks</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.
min.css">
</head>
<body class="text-center">

<main class="mx-auto mt-3" style="width: 350px">
<form th:action="@{/login}" method="post" class="mt-5">

<h1 class="h3 mb-3 fw-normal">Вход</h1>

<div class="form-floating mt-2">
<input type="text" class="form-control" name="email" id="email"
placeholder="name@example.com">
<label for="email">Логин</label>
</div>
<div class="form-floating mt-2">
<input type="password" class="form-control" name="password"
id="password" placeholder="Пароль">
<label for="password">Пароль</label>
</div>

<div th:if="${errorMessage}" class="alert alert-danger mt-2"
role="alert" th:text="${errorMessage}"></div>

<button class="w-100 btn btn-lg btn-primary mt-2"
type="submit">Вход</button>
</form>

```

```

<form th:action="@{/singup}" method="get">
<button class="w-100 btn btn-lg btn-primary mt-2"
type="submit">Регистрация</button>
</form>

<p class="mt-5 mb-3 text-body-secondary">© 2017-2023</p>

</main>

</body>
</html>

```

Файл «index.html»

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Yandex Maps with Bootstrap Sidebar</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap
.min.css" rel="stylesheet">
    <style>
        body {
            display: flex;
            flex-direction: column;
            height: 100vh;
            margin: 0;
        }
        #map {flex-grow: 1;}
        #sidebar {
            width: 300px;
            height: 100%;
            overflow-y: auto;
            background-color: #f8f9fa;
            position: fixed;
            left: -300px;
            top: 0;

```

					ПГУ1.090301.10.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

```

        transition: left 0.3s ease;
        z-index: 2;
    }
    #sidebar.open {left: 0; }
    #map.shifted {margin-left: 300px;}
    .placemark-item img {
        width: 100%;
        height: auto;
        object-fit: cover;}
    .rating {color: #f39c12; }
    .placemark-content {
        display: flex;
        flex-direction: column;}
    .navbar-brand img {
        height: 30px;}
    #toggleButton {
        position: relative;
        left: 0;
        top: 0px;
        z-index: 3;
        transition: left 0.3s ease;}
    #sidebar.open ~ #toggleButton {
        left: 300; }
</style>
</head>
<body>
    <div id="container">
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <div class="mr-1">
                <button id="toggleButton" class="btn btn-
warning">Меню</button>
            </div>
            <a class="navbar-brand" href="#">
                                
    </a>

    <form class="form-inline mr-auto" id="searchForm">

        <input class="form-control mr-sm-2" type="search"
placeholder="Поиск меток" aria-label="Поиск" id="searchInput">

        <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Поиск</button>

    </form>

    <div class="ml-2">

        <button class="btn btn-outline-primary">Войти</button>

        <button class="btn btn-outline-
secondary">Регистрация</button>

    </div>

</nav>
</div>

<div id="sidebar" class="p-3">

    <br><br>

    <h2 class="h4">Добавить/Редактировать метку</h2>

    <form id="placemark-form">

        <div class="form-group">

            <label for="placeName">Название места</label>

            <input type="text" class="form-control" id="placeName"
placeholder="Название места" required>

        </div>

        <div class="form-group">

            <label for="description">Описание</label>

            <input type="text" class="form-control"
id="description" placeholder="Описание" required>

        </div>

        <div class="form-group">

            <label for="imageFiles">Изображения</label>

            <input type="file" class="form-control-file"
id="imageFiles" accept="image/*" multiple>

        </div>

        <div class="form-group">

            <label for="rating">Оценка (от 1 до 5)</label>

```

```

        <input type="number" class="form-control" id="rating"
placeholder="Оценка (от 1 до 5)" min="1" max="5" required>

    </div>

    <button type="submit" class="btn btn-warning">Сохранить
метку</button>

</form>

<h2 class="h4 mt-4">Метки</h2>

<ul id="placemark-list" class="list-group">
    <!-- Здесь будут появляться метки -->
</ul>

</div>

<div id="map"></div>

<script src="https://api-maps.yandex.ru/2.1/?apikey=785109ed-0dac-
45db-afaa-d893b184c454&lang=ru_RU" type="text/javascript"></script>

<script src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>

<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper
.min.js"></script>

<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.m
in.js"></script>

<script>
    ymaps.ready(init);

    let myMap;
    let placemarks = [];
    let placemarkList = document.getElementById('placemark-list');
    let sidebar = document.getElementById('sidebar');
    let mapContainer = document.getElementById('map');
    let toggleButton = document.getElementById('toggleButton');
    let placemarkForm = document.getElementById('placemark-form');
    let searchForm = document.getElementById('searchForm');
    let searchInput = document.getElementById('searchInput');
    let selectedCoords = null;
    let editingIndex = null;
    toggleButton.addEventListener('click', function() {

```

```

        sidebar.classList.toggle('open');
        mapContainer.classList.toggle('shifted');
    });
    function init() {
        myMap = new ymaps.Map("map", {
            center: [55.76, 37.64], // Координаты центра карты
            zoom: 10 // Zoom
        });
        // Обработка клика на карте
        myMap.events.add('click', function (e) {
            selectedCoords = e.get('coords');
            editingIndex = null;
            sidebar.classList.add('open');
            mapContainer.classList.add('shifted');
        });
    }
    placemarkForm.addEventListener('submit', function(event) {
        event.preventDefault();
        if (selectedCoords || editingIndex !== null) {
            const placeName =
document.getElementById('placeName').value;
            const description =
document.getElementById('description').value;
            const rating =
document.getElementById('rating').value;
            const imageFiles =
document.getElementById('imageFiles').files;
            const imageUrls = [];
            const processPlacemark = () => {
                if (editingIndex === null) {
                    ymaps.geocode(selectedCoords).then((res) => {
                        const address =
res.geoObjects.get(0).getAddressLine();
                        addPlacemark(selectedCoords, address,
placeName, description, imageUrls, rating);
                    });
                }
            };
        }
    });

```

```

        } else {
            editPlacemark(editingIndex, placeName,
description, imageUrls, rating);
        }
        placemarkForm.reset();
        sidebar.classList.remove('open');
        mapContainer.classList.remove('shifted');
    };
    if (imageFiles.length > 0) {
        let fileReaders = [];
        for (let i = 0; i < imageFiles.length; i++) {
            let reader = new FileReader();
            reader.onload = function(e) {
                imageUrls.push(e.target.result);
                if (imageUrls.length ===
imageFiles.length) {
                    processPlacemark();
                }
            };
            reader.readAsDataURL(imageFiles[i]);
        }
    } else {
        processPlacemark();
    }
}

});

function addPlacemark(coords, address, placeName, description,
imageUrls, rating) {
    var balloonContent = '<strong>Адрес:</strong> ' + address
+ '<br>' +
                                '<strong>Название места:</strong> ' +
placeName + '<br>' +
                                '<strong>Описание:</strong> ' +
description + '<br>' +
                                '<strong>Оценка:</strong> ' + rating
+'<br>';

    if (imageUrls.length > 0) {

```

					ПГУ1.090301.10.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

```

        balloonContent += '<div>';

        imageUrls.forEach(function(url) {
            balloonContent += '';
        });

        balloonContent += '</div>';
    }

    var placemark = new ymaps.Placemark(coords, {
        hintContent: address,
        balloonContent: balloonContent
    }, {
        iconLayout: 'default#image',
        iconImageHref:
'C:\Users\Iskandar\Desktop\testmap\static\img\logo.jpg',
        iconImageSize: [30, 42],
        iconImageOffset: [-5, -38]
    });

    placemarks.push(placemark);
    myMap.geoObjects.add(placemark);
    updatePlacemarkList();
}

function editPlacemark(index, placeName, description,
imageUrls, rating) {
    const placemark = placemarks[index];
    const coords = placemark.geometry.getCoordinates();

    var balloonContent = '<strong>Адрес:</strong> ' +
placemark.properties.get('hintContent') + '<br>' +

                                '<strong>Название места:</strong> ' +
placeName + '<br>' +

                                '<strong>Описание:</strong> ' +
description + '<br>' +

                                '<strong>Оценка:</strong> ' + rating
+ '<br>';

    if (imageUrls.length > 0) {
        balloonContent += '<div>';

        imageUrls.forEach(function(url) {

```

```

        balloonContent += '';

        });

        balloonContent += '</div>';

    }

    placemark.properties.set('balloonContent',
balloonContent);

    updatePlacemarkList();

}

function deletePlacemark(index) {

    myMap.geoObjects.remove(placemarks[index]);

    placemarks.splice(index, 1);

    updatePlacemarkList();

}

function updatePlacemarkList() {

    placemarkList.innerHTML = '';

    placemarks.forEach((placemark, index) => {

        var listItem = document.createElement('li');

        listItem.className = 'placemark-item list-group-item';

        var balloonData =
placemark.properties.get('balloonContent');

        var firstImage = balloonData.match(/Название
места:<\ /strong> ([^<]+)<br>/)[1];

        var description =
balloonData.match(/<strong>Описание:<\ /strong> ([^<]+)<br>/)[1];

        var rating =
balloonData.match(/<strong>Оценка:<\ /strong> (\d+)/)[1];

        var contentDiv = document.createElement('div');

        contentDiv.className = 'placemark-content';

        if (firstImage) {

            var img = document.createElement('img');

            img.src = firstImage[1];

            img.alt = 'Image';

            img.className = 'img-thumbnail mb-2';

```

```

        contentDiv.appendChild(img);
    }
    var placeNameSpan = document.createElement('span');
    placeNameSpan.textContent = placeName;
    contentDiv.appendChild(placeNameSpan);
    var descSpan = document.createElement('span');
    descSpan.textContent = description;
    contentDiv.appendChild(descSpan);
    var ratingSpan = document.createElement('span');
    ratingSpan.className = 'rating';
    ratingSpan.textContent = '★'.repeat(rating);
    contentDiv.appendChild(ratingSpan);
    var btnGroup = document.createElement('div');
    btnGroup.className = 'btn-group mt-2';
    var editBtn = document.createElement('button');
    editBtn.className = 'btn btn-sm btn-primary edit-
button';

    editBtn.textContent = 'Редактировать';
    var deleteBtn = document.createElement('button');
    deleteBtn.className = 'btn btn-sm btn-danger delete-
button';

    deleteBtn.textContent = 'Удалить';
    btnGroup.appendChild(editBtn);
    btnGroup.appendChild(deleteBtn);
    listItem.appendChild(contentDiv);
    listItem.appendChild(btnGroup);
    listItem.setAttribute('data-index', index);
    listItem.querySelector('.delete-
button').addEventListener('click', function (e) {
        e.stopPropagation();
        deletePlacemark(index);
    });
    listItem.querySelector('.edit-
button').addEventListener('click', function (e) {
        e.stopPropagation();

```

```

        editPlacemarkForm(index);
    });
    listItem.addEventListener('click', function () {
myMap.setCenter(placemarks[index].geometry.getCoordinates(), 14, {
        duration: 300
    });
    placemarks[index].balloon.open();
    });
    placemarkList.appendChild(listItem);
    });
}

function editPlacemarkForm(index) {
    const placemark = placemarks[index];
    const balloonData =
placemark.properties.get('balloonContent');
    const placeName = balloonData.match(/<strong>Название
места:<\strong> ([^<]+)<br>/)[1];
    const description =
balloonData.match(/<strong>Описание:<\strong> ([^<]+)<br>/)[1];
    const rating =
balloonData.match(/<strong>Оценка:<\strong> (\d+)/)[1];
    const imageMatches = [...balloonData.matchAll(/ {

```

```

        const placeName =
        placemark.properties.get('balloonContent').match(/<strong>Название
        места:<\/strong> ([^<]+)<br>\/)[1].toLowerCase();

        if (placeName.includes(searchTerm)) {

        myMap.setCenter(placemarks[index].geometry.getCoordinates(), 14, {
            duration: 300

            });

            placemarks[index].balloon.open();

        }

    });

});

</script>
</body>
</html>

```

					ПГУ1.090301.10.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательно)

Стартап-проект. Веб-приложение «Фуд карта»

Презентация

					ПГУ1.090301.10.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

Стартап-проект. Веб-приложение «Фуд – карта»

Автор работы: студент группы 20ВВП1 Кодиров И.Н.
Научный руководитель: к.т.н., доцент кафедры ВТ Юрова О.В,

Пенза 2024

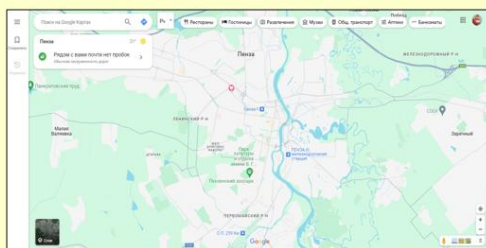
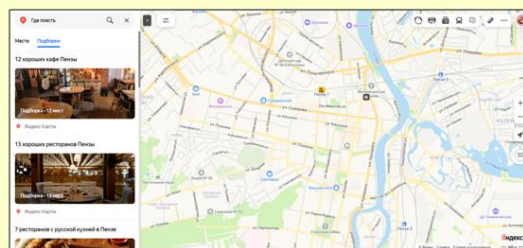
Актуальность проекта

В современном мире развитие информационных технологий позволило людям создавать интерактивные карты местностей и взаимодействовать с ними. Благодаря такому развитию появляется вопрос информированности о разнообразных заведениях в ресторанной индустрии, занимающей значительную роль в жизни каждого из нас. Существование таких технологий создаёт новые способы взаимодействия с ними. В данном контексте, создание карты, позволяющей создавать собственные списки заведений с собственным описанием и особенностями приглянувшимися вам и другим людям, становится удобным инструментом для планирования собственного досуга, и полезным не только для конечного потребителя, но и для самих владельцев заведений.

Обзор аналогов

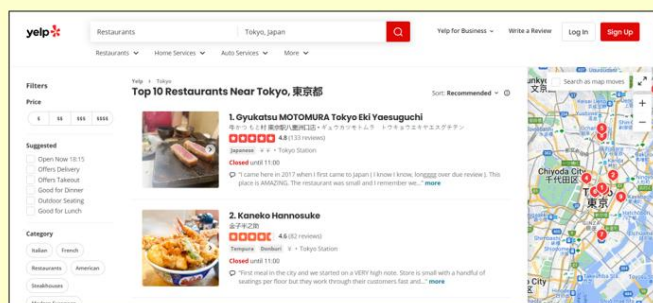
Google Карты

Яндекс Карты



Обзор аналогов

Yelp!



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.10.001 ПЗ

Лист

84

Используемый стек технологий



Диаграмма прецедентов

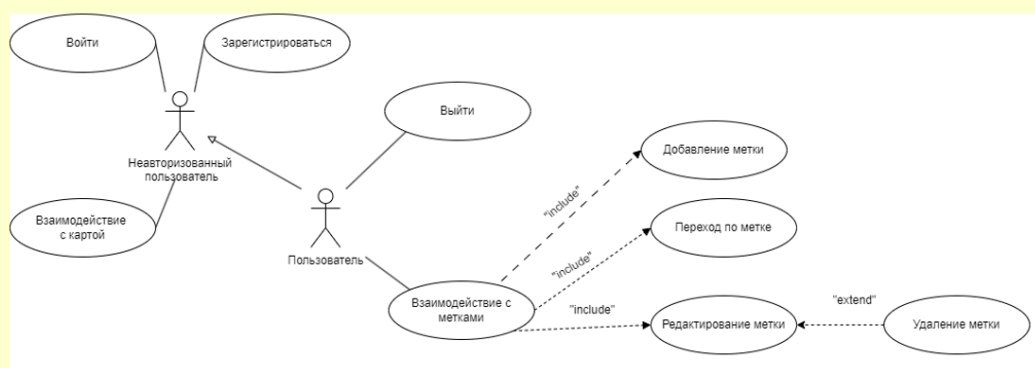
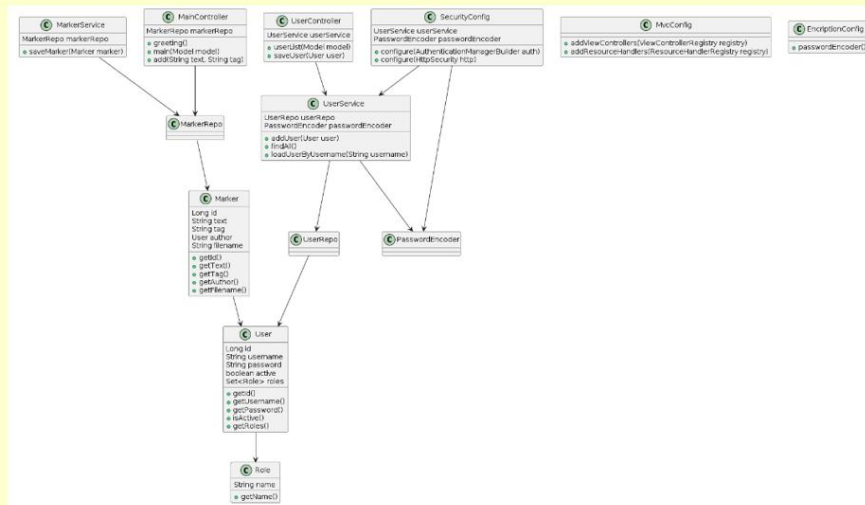
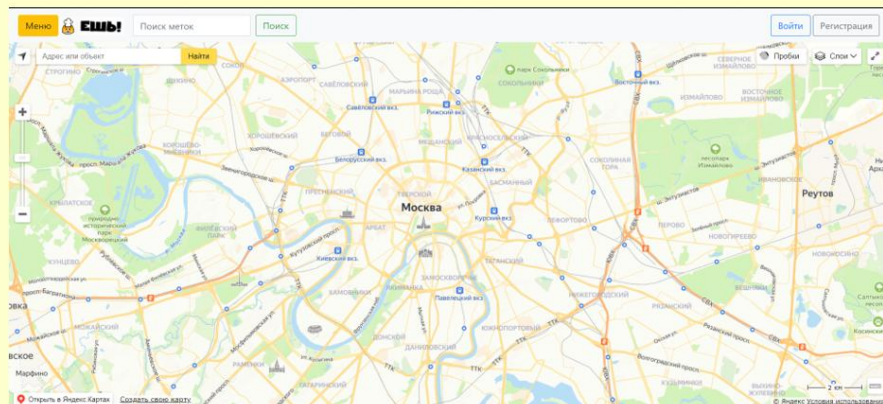


Диаграмма классов



Веб-приложение «Фуд-карта»



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.10.001 ПЗ

Лист

86

Меню

**Добавить/
Редактировать метку**

Название места

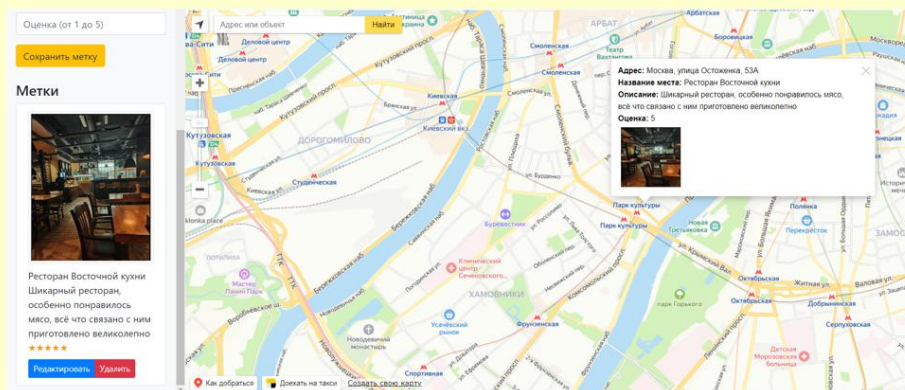
Описание

Изображения

Оценка (от 1 до 5)

Сохранить метку

Метки



Спасибо за внимание



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.10.001 ПЗ

Лист

87