Министерство образования и науки Российской Федерации
ФГБОУ ВО РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
(РГГМУ)

Институт Информационных систем и геотехнологий
КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

БАКАЛАВРСКАЯ РАБОТА

На тему: Проектирование и разработка информационной системы “Путеводитель по кофейням” с элементами ГИС технологий

Исполнитель _____Шевченко Ярослав Денисович_____
(фамилия, имя, отчество)
Руководитель _____к.т.н. доцент кафедры ПИ_____
(ученая степень, ученое звание)
_____Яготинцева Наталья Владимировна_____
(фамилия, имя, отчество)

«К защите допускаю»
Заведующий кафедрой _____
(подпись)
_____
(ученая степень, ученое звание)
_____
(фамилия, имя, отчество)
«___»_____20__г.

Санкт–Петербург
2022

Содержание

Введение

В своих истоках человечество добывало пропитание с помощью охоты и собирательства. Это было необходимо в рамках выживания. Цивилизация со временем развивалась, а также упрощались процессы добычи еды. Люди начали питаться не для того, чтобы выжить, а в том числе для получения удовольствия от пищи. Появлялись новые виды блюд и напитков. В настоящем времени одним из самых популярных напитков является кофе. Востребованность этого напитка очень сложно переоценить. Он стоит на втором месте по популярности напитков во всем мире.

В современном мире существует большое количество разновидностей сортов кофе и способов его приготовления. Вкус данного продукта может меняться от места, где он был приготовлен, от человека, который его приготовил, а также в зависимости от зерен, на которых он был приготовлен. У всех людей свои вкусы и предпочтения.

В настоящее время, один из самых главных вопросов состоит в том, как и где найти кофе, который тебе понравится.

Решением данного вопроса является не обход всех заведений в своем городе или стране, а возможность воспользоваться приложением, в котором будет указанно расположение всех ближайших мест где можно выпить кофе, а также информации о них, которая состоит из отзывов посетителей, которые там бывали, диаграммы, которая представляют из себя статистику популярных напитков, все это позволит человеку определиться со своим выбором.

Исходя из всего вышеперечисленного основной целью работы является проектирование и разработка информационной системы с ГИС технологиями по нахождению пути до кофейни.

Приведем остальные аспекты, связанные с выпускной квалификационной работой:

1. Задачами выпускной квалификационной работы являются:

    1) Анализ отрасли темы разработки

2) Разработка схем, диаграмм и сценариев, формирующих проект для последующей реализации;
3) Создание готового прототипа исходя из произведенной проектной работы;
4) Обоснование надежности системы.

2. Используемые методы, технологии и стандарты:
1) Сравнительный анализ имеющихся на рынке информационных систем;
2) Анализ и моделирование бизнес-процессов посредством BPMN;
3) Моделирование системы при помощи UML диаграмм
4) Шаблон проектирования Model-View-ViewModel
5) Технологии проектирования баз данных и информационных систем;
6) Серверные технологии.

3. Используемые инструментарные средства:
1) Программы пакета Microsoft Office;
2) Язык программирования C#
3) Фреймворк разработки Xamarin
4) Интегрированная среда разработки Visual Studio 2022
5) Виртуальный сервер Ubuntu
6) Инструмент администрирования баз данных HeidiSQL

В качестве результата работы будет представлено:

1. Модели информационной системы
2. Результат анализа надежности системы
3. UML диаграммы проектируемой системы
4. Интерфейс и пример работы прототипа
5. Анализ произведенной работы

Приведенный далее процесс разработки является всего лишь вариантом реализации проекта, существует множество других решений при возможных подходах к созданию информационной системы.

1. Анализ картографических сервисов

1.1 Проблемы картографических сервисов

Проблема навигации всегда была важна и актуальна. С давних пор люди пользовались различными навигационными комплексами. Поначалу, человечество применяло навигационные знаки на дорогах, которые могли с течением времени стать не читаемыми под длительным воздействием природных условий, затем оно прибегло к компасу и карте, но карты могли стареть или становиться не достоверными, так же приходилось носить с собой множество карт для каждой конкретной местности и, при этом, не все люди умели ориентироваться по ним, после всего этого, на смену этим вещам пришёл новый аналог в виде спутниковой навигации, которой пользуются множество картографических сервисов сегодня.

В наше время, картографические сервисы не просто стали использовать спутниковую навигацию, а добавили информацию в картографические объекты, то есть, метка на карте стала способна нести не только данные о местоположении объекта, а еще и данные о владельце, номер контактного телефона, описание, отзывы и другие всевозможные поля.

Современные картографические сервисы такие как: «Google Maps», «Yandex.Maps» и «2GIS» предоставляют точную и достоверную информацию пользователям, но из-за большого количества поступающих данных, они обобщают сведения обо всех картографических точках, тем самым упуская более детальное описание объектов. Конечно, в эти сервисы можно добавить загружаемый модуль, который добавит более детальные сведения к какой-то категории объектов или можно выпустить глобальное обновление для всего приложения. Но существует несколько причин, которые исключают эти варианты:

Во-первых, не каждый пользователь поймет, что для более детальной информации о категории ему требуется скачать отдельный, загружаемый модуль.

Во-вторых, глобальное обновление будет неоправданно дорогим для компании, которая выпускает данный программный продукт, так как неизвестно, насколько это обновление будет использоваться другой частью пользователей, которым не особо нужен новый функционал.

Остается последний вариант, создание специализированного программного обеспечения, которое будет нацелено всего на одну категорию. Только так картографический сервис способен предоставлять самую детальную и проработанную информацию о конкретном объекте.

Имея более детальные сведения, пользователь, с большей вероятностью, сможет сделать выбор, какое место ему стоит посетить, в конечном итоге, от которого он будет доволен.

Для данной дипломной работы была выбрана категория заведений, в которых подают кофе. Выбор был основан на популярности данного напитка и количестве точек, которые его предоставляют.

Благодаря картографической системе, которая специализирована на категории кофе, пользователю будет доступна обширная информация, например, не только о часах работы заведения, а еще и сведения о том, какой кофе подается в конкретном месте, какой их продукт на вкус, запах, основываясь на статистических данных пользователей поставивших оценку напитку, какой десерт к определенному напитку стоит взять по рекомендациям других пользователей.

1.2 Сравнение с программными аналогами

Необходимо провести сравнительный анализ других схожих информационных систем, для выявления их основных недостатков и достоинств. Это поможет выявить в каких доработках нуждается та или иная система, что поспособствует более грамотному решению в каких блоках нуждается информационная система, которая проектируется и разрабатывается в данной дипломной работе.

Таблица 1.1. Сравнение достоинств и недостатков схожих информационных систем

Программный продукт	**Критерий оценки**	**Признаки**
Google Maps	Достоинства	Наличие рейтинговой системы Возможность комментирования Огромная база данных заведений
	Недостатки	Приложение перегружено лишним функционалом Данные обобщены
OnePriceCoffee	Достоинства	Акционные предложения Программа лояльности
	Недостатки	Приложение завязано на кофейне одной компании Устаревший дизайн Большой расход оперативной памяти
CoffeLike	Достоинства	Навигация на карте Программа лояльности
	Недостатки	Приложение завязано на кофейне одной компании

Благодаря проведенному сравнительному анализу достоинств и недостатков схожих информационных систем были выявлены общие функции, приведённые в таблице 1.2.

Таблица 1.2. Анализ функций схожих информационных систем

Функционал	Google Maps	OnePriceCoffee	CoffeLike
Наличие карты	+	-	+
Возможность навигации	+	-	+
Возможность сделать заказ	-	+	-
Выбор места на интерактивной карте	+	-	-
Поиск по ключевым словам	+	-	-

Функционал	Google Maps	OnePriceCoffee	CoffeLike
Предоставление контактных данных	+	+	-
Система оценивания и комментирования заведений	+	-	-
Push-уведомления	+	-	+
Получение акционных предложений	-	+	-

Таким образом, можно выделить следующий функционал для нашей информационной системы:

Таблица 1.3. Возможные функции информационной системы

№	**Необходимые функции**	**Желательные функции**	**Возможные функции**	**Отсутствующие функции**
1.	Возможность нахождения оптимального пути	Отправление отчета об ошибке администратору	Система рекомендаций для пользователя	Система кэшбека и скидок для пользователей
2.	Система оценивания заведений	Система авторизации и регистрации пользователей	Система поощрения пользователей	Наличие платной подписки на определённые кофейни
3.	Нахождение кофейни, по ключевым словам	Изменение информации о кофейни администратором	Система рекламы и продвижения для владельцев заведений	Поддержка PC
4.	Поддержка Android	Система индивидуализации профиля	Поддержка IOS	Поддержка WEB клиента

1.3 Анализ объекта исследования

Системный анализ[1] — научный метод, который отличается междисциплинарным подходом к решению сложных проблем. Объектом системного анализа выступают практические проблемы, которые связанны с созданием новых и модернизацией существующих систем.

В рамках выпускной квалификационной работы были использованы следующие методологии анализа:

- SWOT
- VCM
- BRP
- ISA

Для понимания процессов, протекающих внутри разрабатываемой информационной системы, проведем анализ. Сначала проведем анализ согласно методологии ISA[2].

Таблица 1.4. ISA анализ

	A	B	C	D	E	F
1	Интерфейс Сервер Приложение	Взаимодействие с заведениями	Сервер Приложение	Пользователь	В момент использования	Получение данных
2	Количество посещений Количество продаж Процент покупок	Проверка системы на функционирование	Аренда сервера	Покупка продвижения	После покупки	Получение прибыли, идея
3	Приложение Бизнес логика	Просмотр карты заведений	Смартфон	Администратор Владелец Пользователь	Программирование, отладка, тестирование, сопровождение	Создание проекта
4	Юр. Документы	Оформление документов	Сервер	Администратор, владелец	Документ оказания услуги	Юр. Грамотно вывести на рынок
5	Количество продаж Количество посещений	Покупка продвижения	Приложение ИС	Администратор	При покупке продвижения	Получение клиентов

Благодаря данному анализу можно выяснить какие роли, составляющие информационной системы и процессы, должны происходить.

Далее проведем VCM анализ для оценивания возможности получения выгоды от разрабатываемой информационной системы.

1. Емкость – Контактная, навигационная информация о кофейнях
2. Роль в отраслевой структуре – социальная помощь в навигации
3. Конкурент:
    a. Отечественный разработчик
    b. Бесплатное распространение
    c. Подробность информации
4. Новое направление – распространение рекламы посредством поднятия заведений в поиске, подписки
5. Выгода – Дальнейшее введение рекламы заведений при большом охвате пользователей

Благодаря данному анализу мы определили конкурентоспособность и конкурентные преимущества.

Теперь приступим к SWOT анализу. SWOT[3] – это комплекс маркетинговых и других исследований сильных и слабых сторон предприятия или конкретного объекта. Основная задача заключается в разработке бизнес-стратегии развития предприятия или объекта, удостоверившись в том, что были учтены все главные факторы – движущие силы для успешного роста. А также рассмотрены возможности внутри компании и внешние факторы.

Таблица 1.5. SWOT анализ

	Положительное влияние	Отрицательное влияние
Внутренняя среда	Сильные стороны • Актуальность и востребованность решения • Уникальность • Постоянное увеличение сетей кофеен	Слабые стороны • Короткий срок использования • Не именитый разработчик

	Положительное влияние	Отрицательное влияние
Внешняя среда	Возможности •Возможность войти на рынок •Рост числа клиентов •Заработок на коллаборации с владельцами	Угрозы •Вполне возможен сценарий, где более сильный и богатый разработчик выиграет рынок •Перенасыщение рекламой

Проведем BPR[4] анализ функции навигации и добавление комментария пользователя.

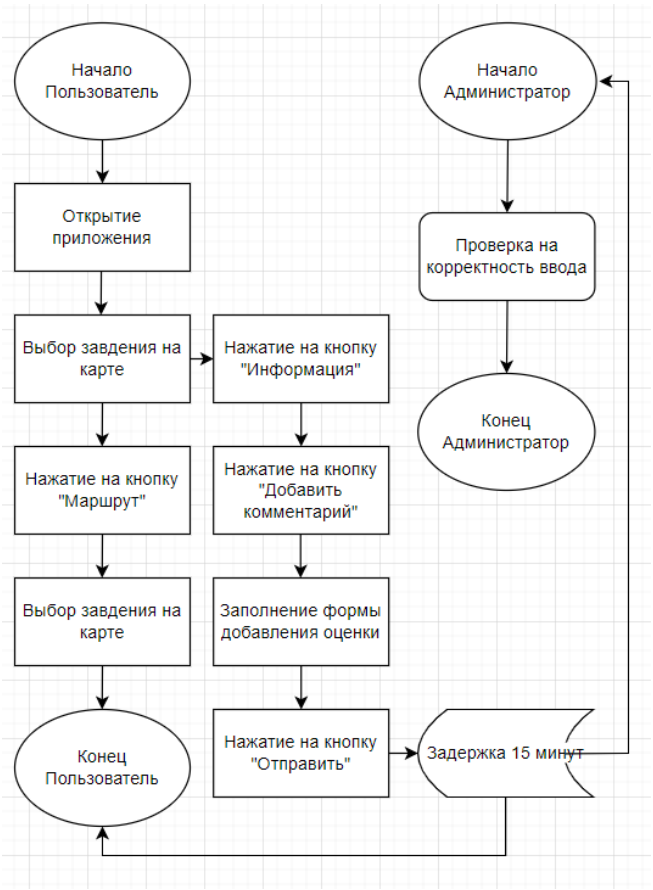

Рисунок 1.1 - BPR диаграмма

После проведения анализа можем составить диаграмму потоков, протекающих внутри информационной системы.

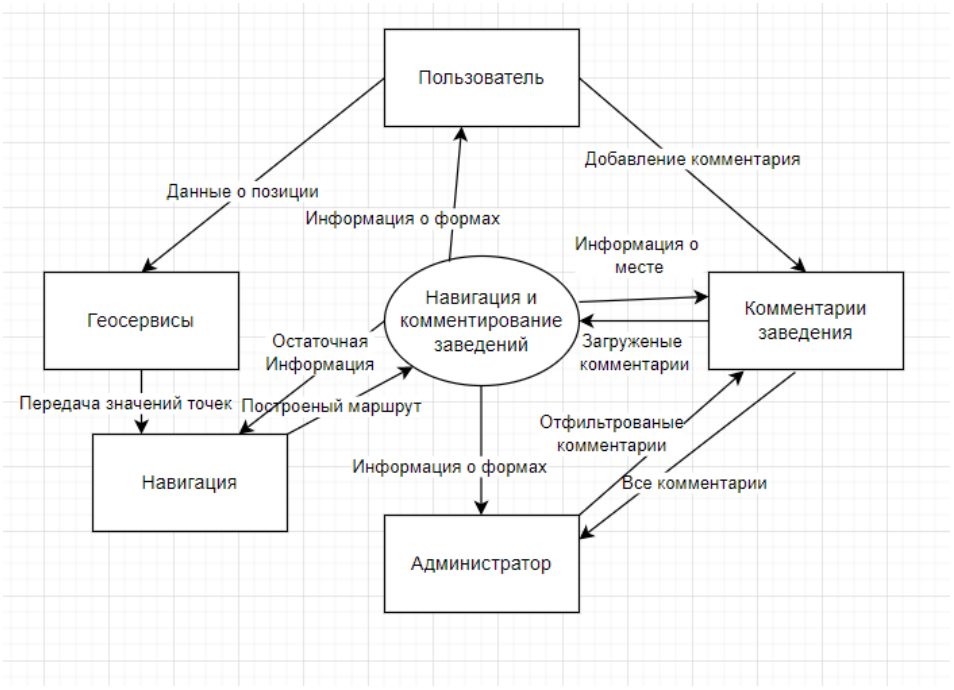

Рисунок 1.2 - Потоки в разрабатываемой информационной системе

По данной диаграмме (Рисунок 1.2) можно визуально определить основные потоки в действиях навигации и комментировании заведений.

2. Проектирование информационной системы

2.1 UML моделирование

UML - язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML расшифровывается как унифицированный язык моделирования, что дает понимание диаграммы не только создателю, но еще и другим людям, которые знают язык моделирования UML

В данной работе применяются следующие UML диаграммы[5]:

- Диаграмма вариантов использования
- Диаграмма последовательностей
- Диаграмма классов
- Диаграмма компонентов
- Диаграмма развертывания

Начнем с построения диаграммы вариантов использования[6]. Это диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент[7] — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

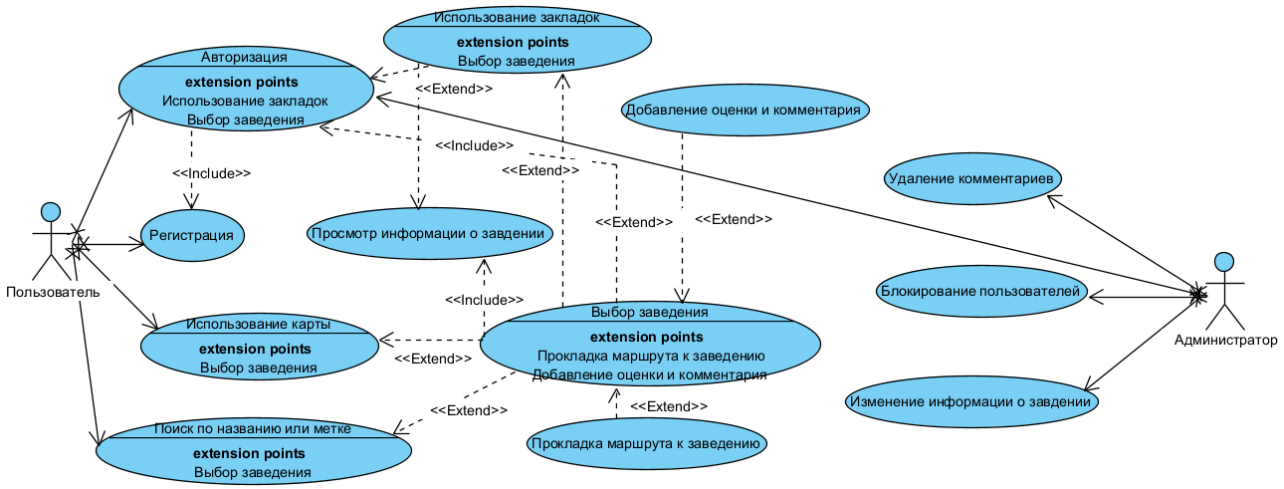

Рисунок 2.1 - Диаграмма вариантов использования

Продолжим моделирование структур диаграммой классов. Она позволяет описать общую структуру классов информационной системы, отобразить их иерархию, поля, свойства, методы, интерфейсы, их взаимодействие и взаимосвязь.

Следующая диаграмма отображает структуру классов[8], разбитых по пакетам, серверной части информационной системы.

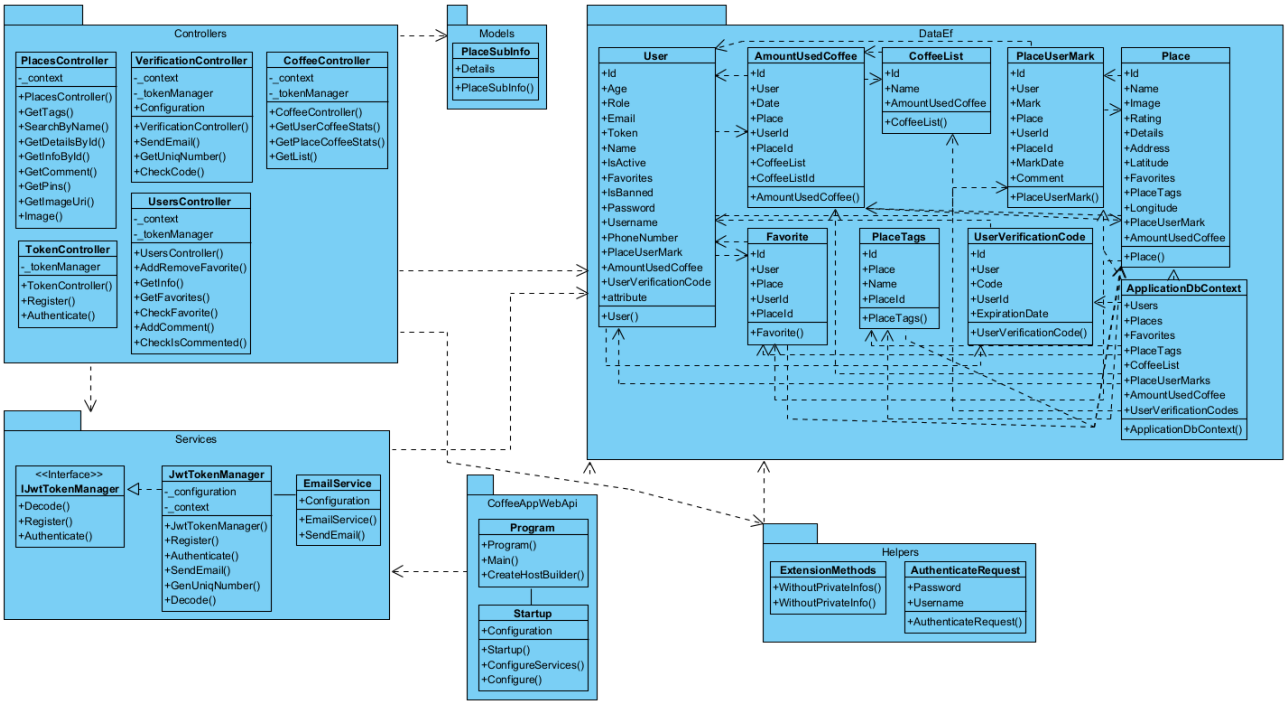

Рисунок 2.2 - Диаграмма классов серверной части

Так как серверная часть взаимодействует с клиентской частью, приведем еще одну классовую диаграмму. Она описана в Приложении А.1.

Диаграмма последовательности[9] показывает жизненный цикл объектов и их взаимодействие между собой в течении этого жизненного цикла в рамках заданного условия.

Диаграмма имеет вид объектов расположенных горизонтально друг за другом, а их жизненный цикл имеет вертикальное отображение. Объекты передают некие данные между собой или актёром при помощи сообщений.

Сообщения могут быть разных типов, а также они могут отображать действия, процессы, свойства начального или конечного объектов.

Диаграмма способна отобразить очень подробно процессы, которые возникают при определенных взаимодействиях актера и системы. Диаграммы последовательности следует приводить, только для клиентской части, так как пользователь может взаимодействовать напрямую только с ней.

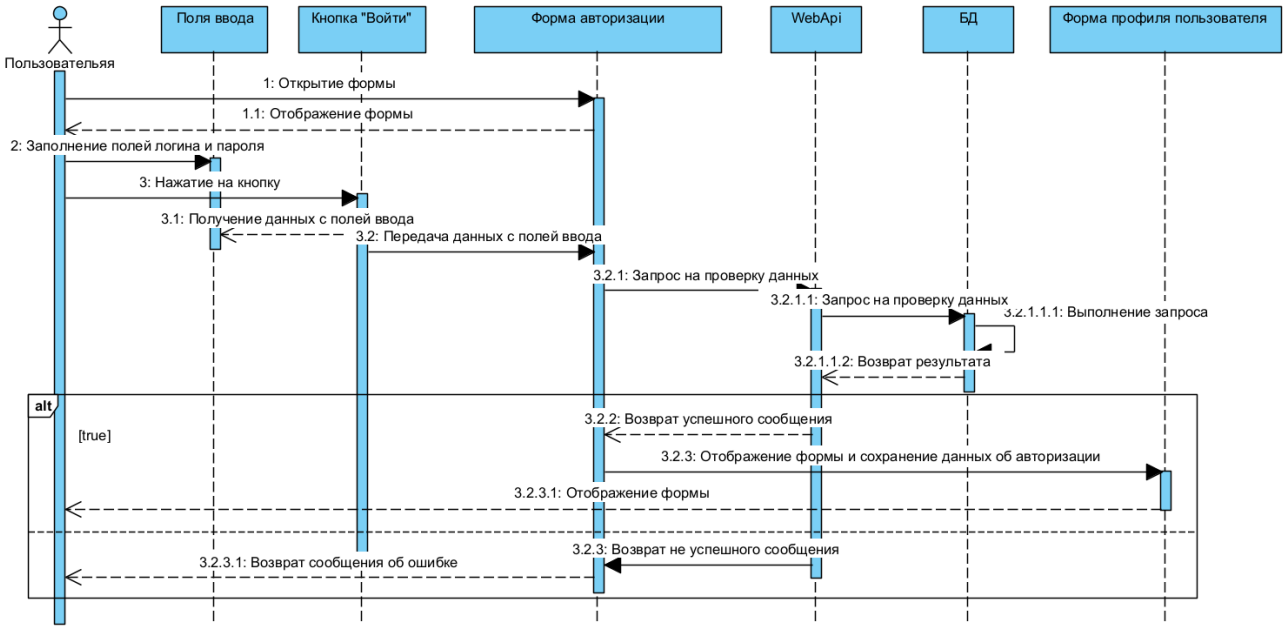

Рисунок 2.3 - Диаграмма последовательности. Авторизация

Из диаграммы на рисунке 2.3 можно наблюдать, что система способна отреагировать на неправильные данные ввода отображением сообщения об ошибке пользователю. Для обеспечения простоты ввода пользователя, следует требовать данные только логина и пароля, другие данные для аутентификации и авторизации не требуются.

Процесс регистрации частично отличается от стандартной регистрации поэтому диаграмму последовательности следует привести. Она описана в

Приложении Б.1. По началу действия, регистрация почти не отличается от авторизации, отличие только в количестве полей для ввода пользователем. Но, после передачи данных на серверную часть происходит верификация данных, генерируется уникальный токен для аутентификации аккаунта и отправляется электронное письмо, со сгенерированным, случайным, шестизначным, уникальным кодом подтверждения, на адрес введенный пользователем. При корректном вводе проверочного кода, пользователя авторизует, передавая клиентскому приложению сгенерированный токен, и перенаправляет на форму профиля. При не корректном адресе электронной почты или неправильным подтверждающим кодом, регистрация пользователя отменяется и данные, которые были добавлены в базу данных, удаляются.

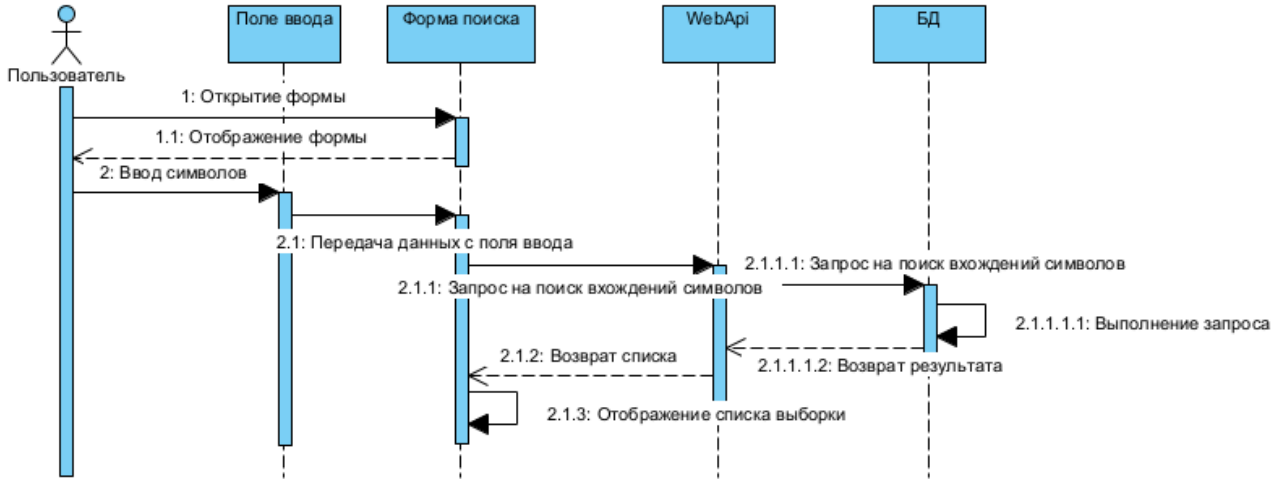

Рисунок 2.4 - Диаграмма последовательности. Поиск

Поиск заведений осуществляется на серверной части приложения обращаясь к индексу наименования в таблице заведений. Обращение происходит через полнотекстовый поиск. Полнотекстовый поиск работает как по всему содержимому столбца или столбцов, так и по части текста содержащего строку.

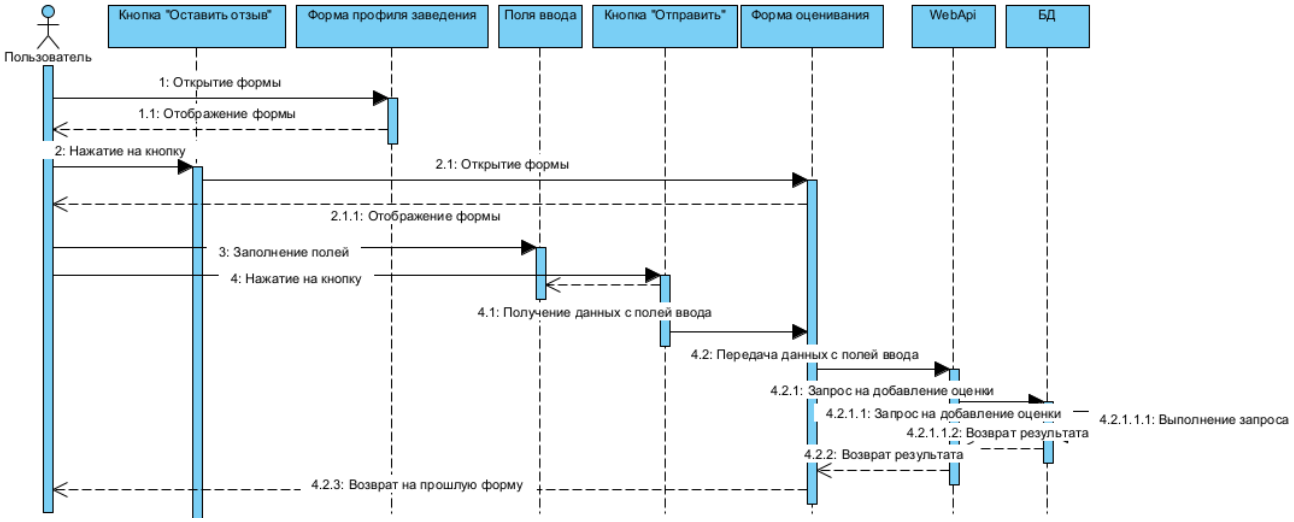

Рисунок 2.5 - Диаграмма последовательности. Оценивание

Для оценивания заведений пользователь должен быть авторизован, но для просмотра комментариев авторизация не требуется. На форме профиля, если пользователь ранее не оценивал выбранное заведение, будет доступна кнопка для добавления отзыва. После нажатия произойдет переход на форму оценивания, на которой можно выбрать оценку и оставить текстовый отзыв. После отправления отзыва, в течении пятнадцати минут, оценка заведения изменится с учетом новой рецензии.

При большом количестве текстовых отзывов скрипт написанный на языке структурированных запросов найдет наиболее используемые слова во всех рецензиях и введет эти слова в таблицу тэгов.

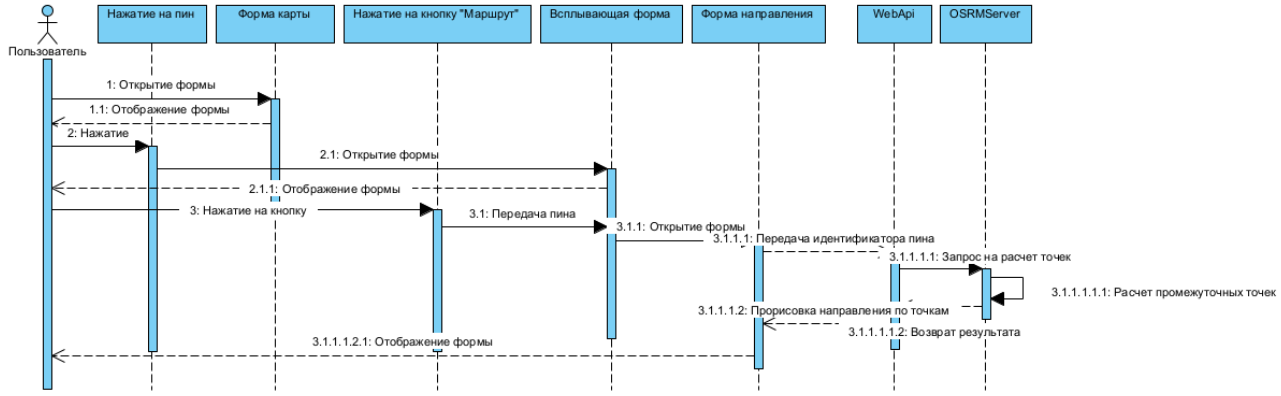

Рисунок 2.6 - Диаграмма последовательности. Нахождение маршрута

Для нахождения пути происходит обращение к серверу, которому передаются данные о текущем местонахождении пользователя и координаты заведения, выбранного пользователем. Серверное приложение, в свою

очередь, обрабатывает данный запрос и передает две позиции “OSRM” серверу, который установлен на отдельном высокопроизводительном сервере, для подсчета промежуточных координат.

После обработки запроса на подсчет промежуточных координат серверному приложению возвращается лист со всеми точками. При получении списка клиентской частью, приложении расставляет невидимые позиции на карте и соединяет их сплошной линией. Таким образом, достигается эффект маршрута состоящего всего из двух точек, начальной и конечной.

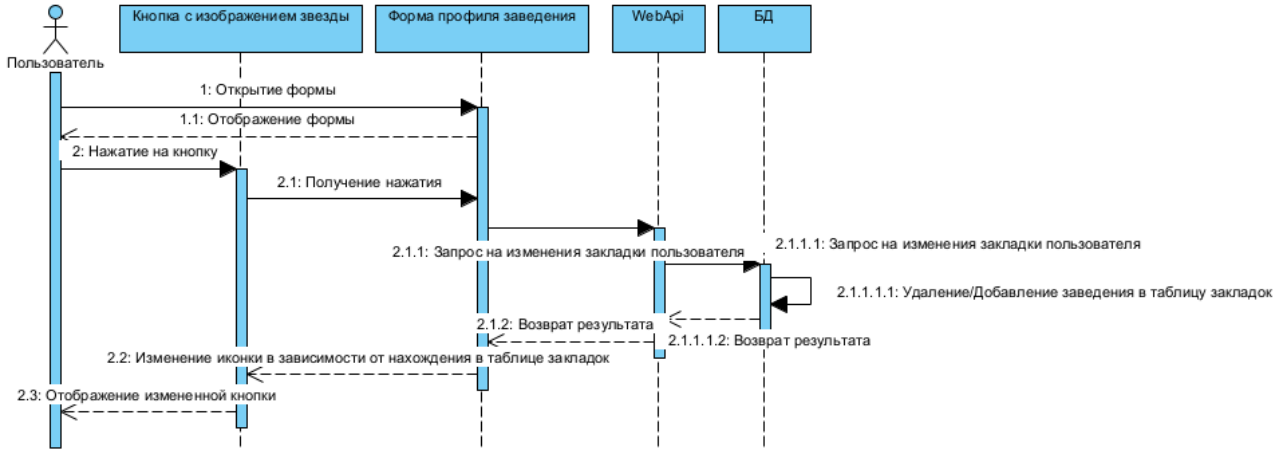

Рисунок 2.7 - Диаграмма последовательности. Добавление/Удаление закладки

Добавление и удаление закладок доступно только для авторизованных пользователей, так как данные о сохранении заведения пользователем добавляются в таблицу базы данных. Визуально пользователей может определить, находится ли место в закладках, на форме закладок, место отображается в списке, или на форме заведения, иконка добавления закладки закрашивается серым цветом.

Следующей диаграммой будет рассмотрена диаграмма компонентов[10]. Данная диаграмма следует цели отобразить структурную связь компонентов между собой.

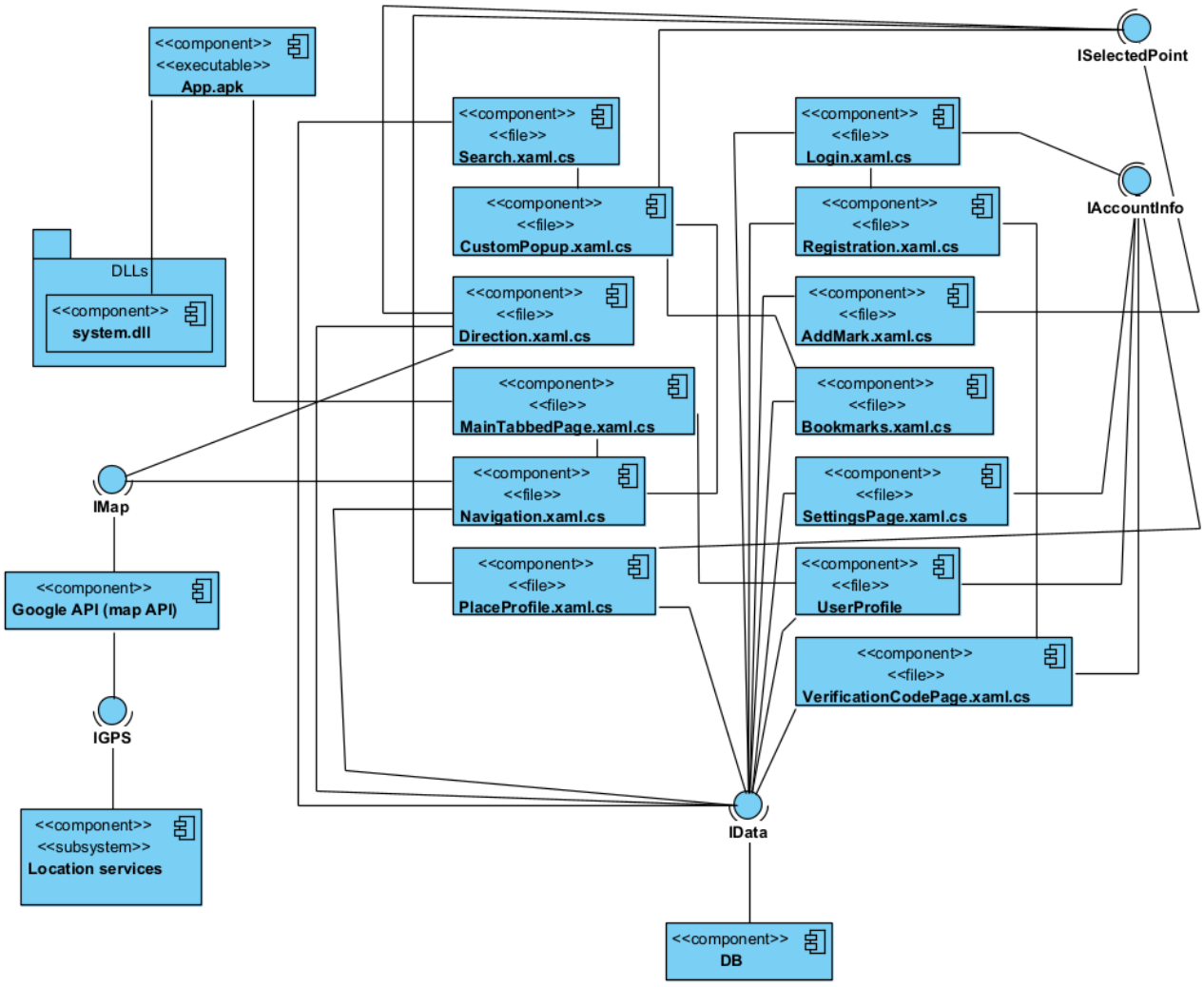

Рисунок 2.8 - Диаграмма компонентов

На диаграмме были выделены все визуальные классы форм, с помощью которых и происходит взаимодействие всех частей системы, а также выделены сторонние компоненты позиционирования в пространстве, встроенные в смартфон, взаимодействие с которыми происходит посредством прикладного программного интерфейса. Компонентом запуска и установки всей клиентской системы является “App.apk”.

Последней UML диаграммой будет рассмотрена диаграмма развёртывания[11][12]. Она моделирует физическое развёртывание информационной системы.

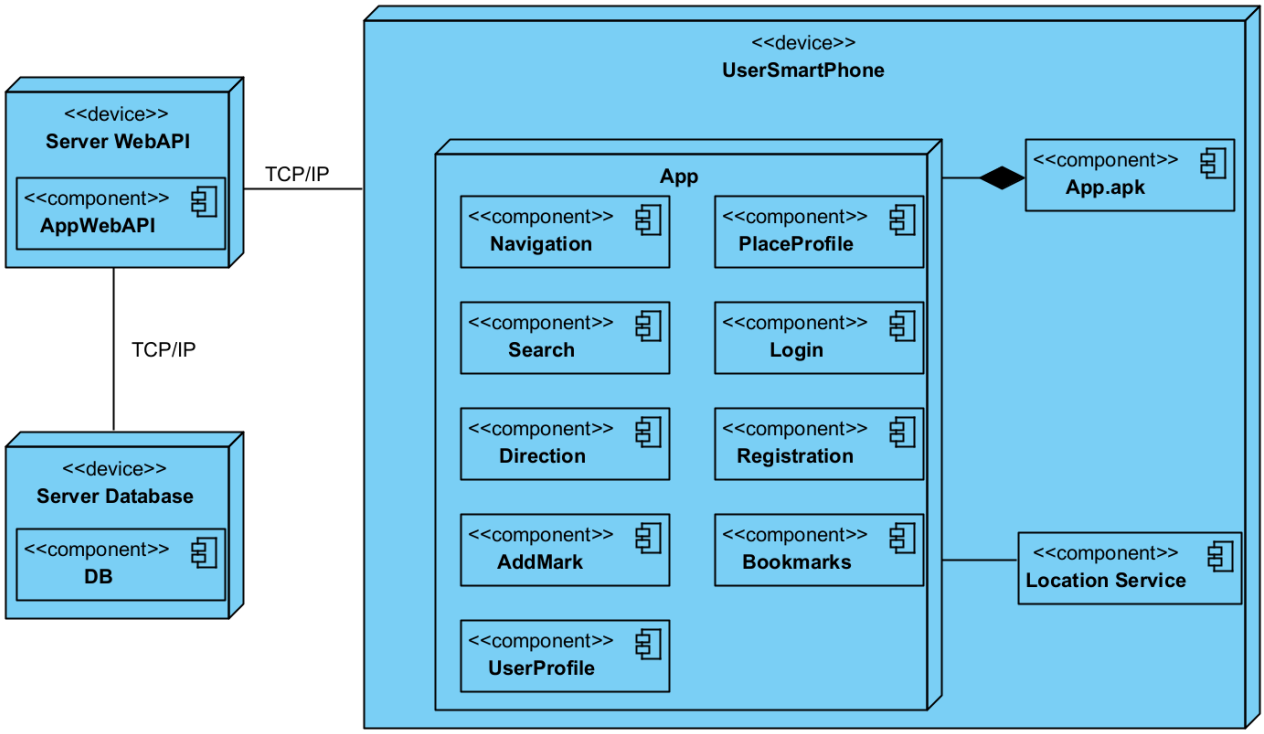

Рисунок 2.9 - Диаграмма развёртывания

Из диаграммы можно вывести, что существует три устройства: сервер базы данных, устройства серверной части[13], клиентское устройство. В клиентском приложении находится большая часть компонентов для взаимодействия между пользователя с другими частями системы. Все внешние связи осуществляются на основе протокола TCP/IP[14].

2.2 Моделирование бизнес-процессов в нотации BPMN

Нотация BPMN[15][16] используется для описания бизнес-процессов нижнего уровня при помощи блок-схем. Данная нотация очень проста для понимания, как человеку, который работает над проектом, так и сторонним людям. При помощи этого моделирования можно описать любой процесс.

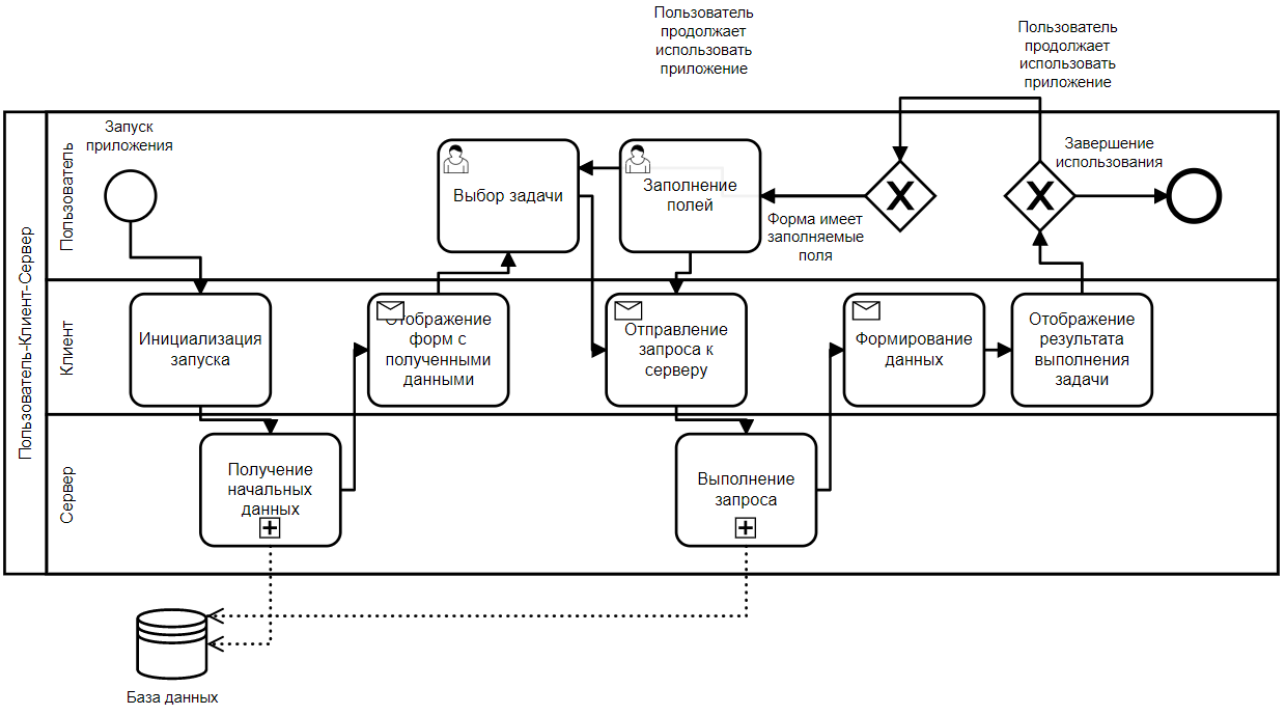

Рисунок 2.10 - BPMN диаграмма общих действий

На диаграмме общих действий (Рисунок 2.10) можно наблюдать, что она разделена на три уровня: пользователь, клиент и сервер. На каждом из уровней происходит выполнение от лица этого уровня. Диаграмма описывает общее использование приложения от запуска до завершения пользователем.

После запуска приложение происходит инициализация запуска на клиентской части, на которой происходит переход на серверную часть и вызывается получение начальных данных.

После этого, коллекция данных передается на клиентский уровень. Пользователь выбирает, какое действие ему требуется совершить. Данные с клиента передаются на сервер и обработанные данные обратно. Если пользователь не завершает выполнение приложения, то пользователь может продолжать пользоваться информационной системой.

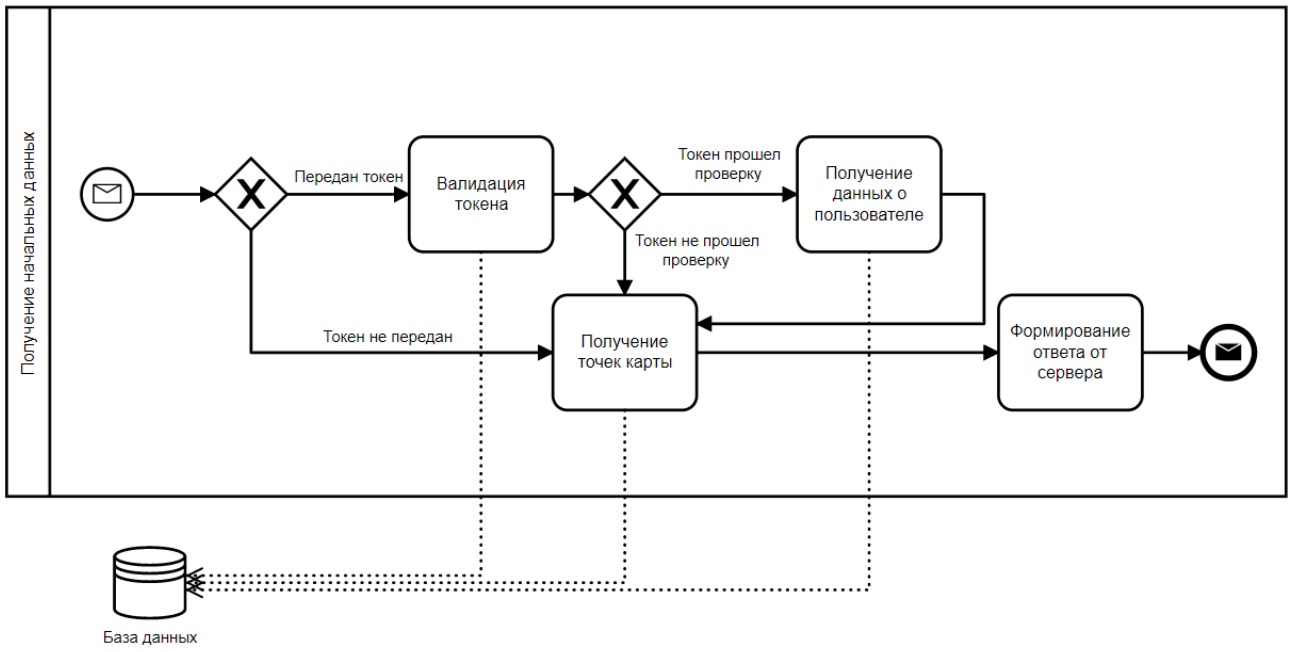

Рисунок 2.11 - BPMN диаграмма получения начальных данных

Диаграмма на рисунке 2.11 описывает получение начальных данных информационной системы. Происходит получение точек карты от базы данных. Дополнительно, если токен был передан, происходит валидация информации и получение данных о пользователе.

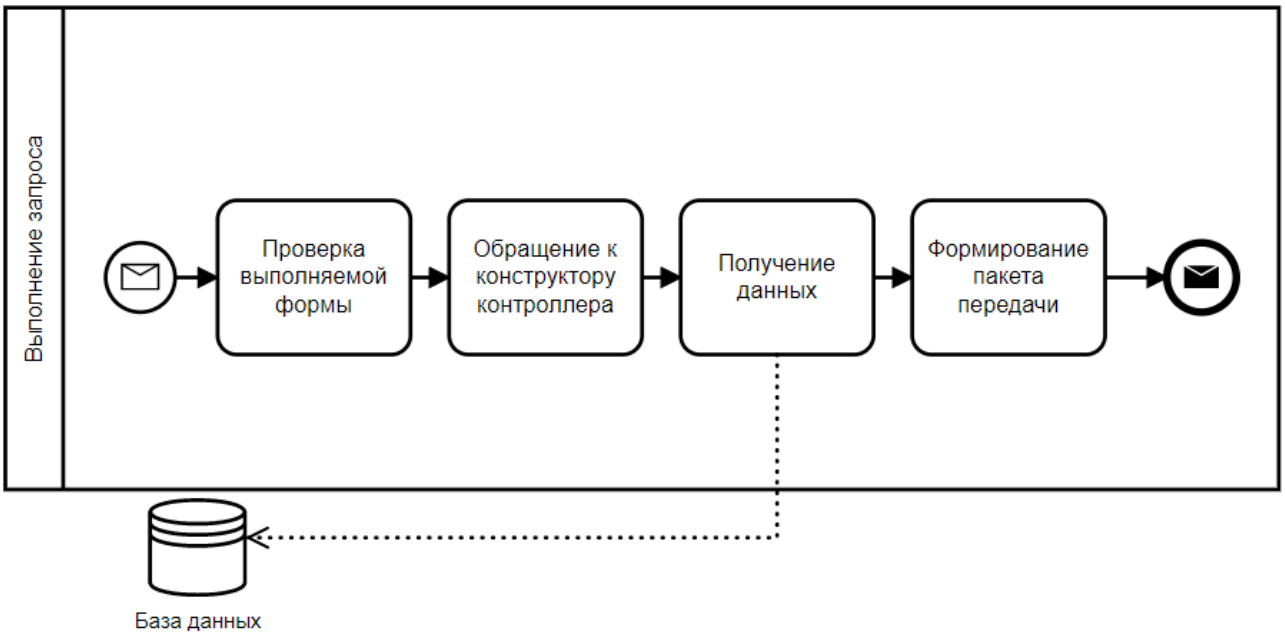

Рисунок 2.12 - BPMN диаграмма выполнения запроса

На диаграмме рисунка 2.12 описано выполнение запроса посылаемого от сервера и выполняется формирование пакета передачи клиентской части.

2.3 Схема базы данных

Схема базы данных построена при помощи ER-модели[17]. Эта модель используется при высокоуровневом проектирование схем баз данных.

Данный стандарт позволяет обозначить сущности и их свойства, с указанием типов данных, а также обеспечивает возможность выделения связей между сущностями. Таблицы описаны на рисунке 2.13

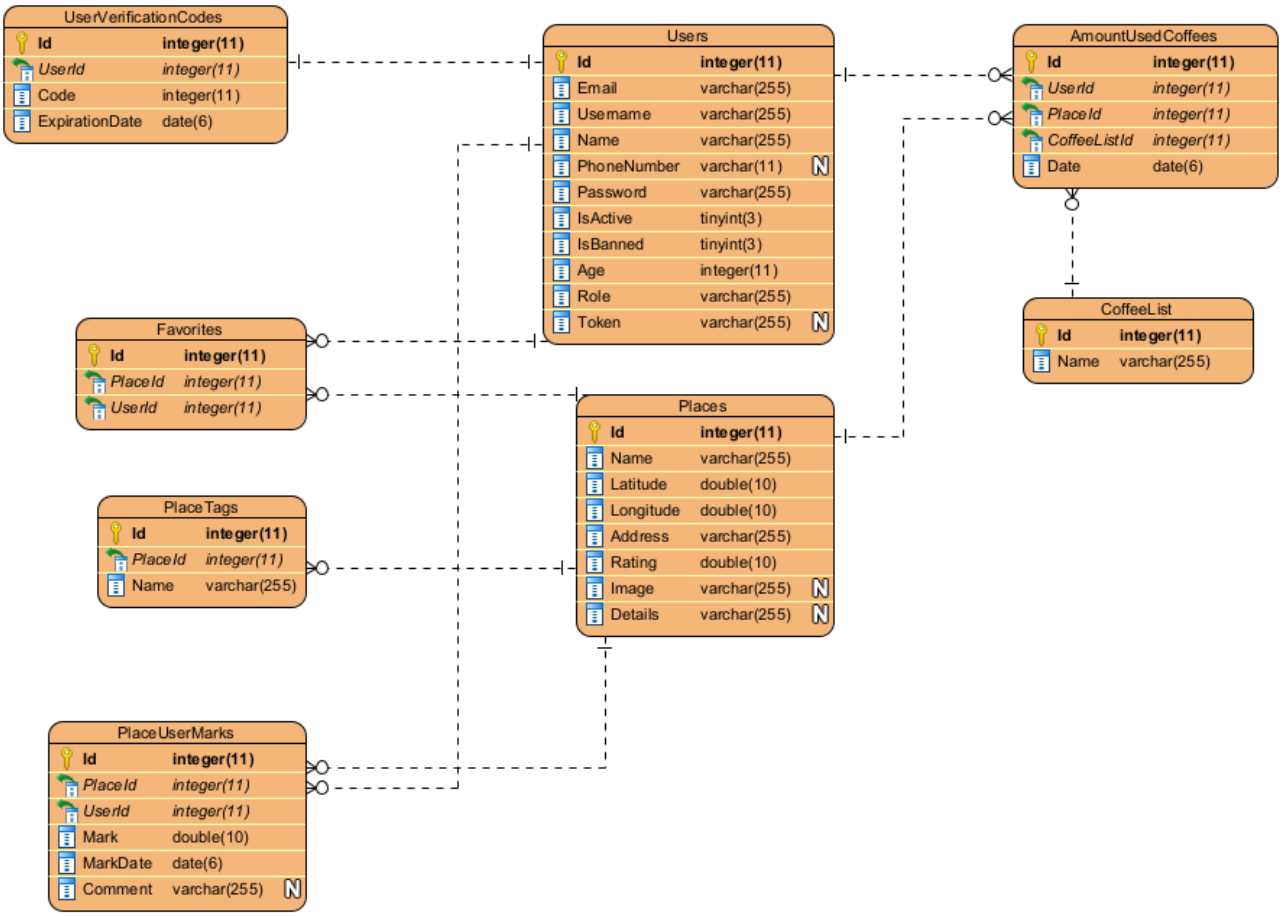

Рисунок 2.13 - Схема базы данных

Таблицы соответствуют первой[18] и второй[19] нормальной форме. А именно устранены повторяющиеся группы в таблицах, отдельные таблицы для наборов данных, относящихся к нескольким записям, таблицы имеют первичный ключ, все таблицы связаны между собой.

Более подробно рассмотрим за что отвечает каждая из структур.

- Таблица “Users” отвечает за хранение данных пользователей;
- Таблица “Places” хранит в себе, как геоинформацию, так и информационно-справочные данные о месте;

- Таблица “Favorites” является связующей таблицей между “Users” и “Places”. Хранит данные о сохраненных заведениях в закладках пользователей;
- Таблица “UserVerificationCodes” содержит в себе временные, сгенерированные коды подтверждения аккаунтов пользователей;
- Таблица “PlaceTags” имеет данные о тэгах принадлежащих заведениям таблицы “Places”;
- Таблица “PlaceUserMarks” имеет связи с таблицами “Users” и “Places” и содержит в себе комментарии и оценки пользователей определенных заведений;
- Таблица “CoffeeList” хранит данные о кофее;
- Таблица “AmountUsedCoffees” содержит ссылки на таблицы “Users”, “Places” и “CoffeeList”. Таблица несет в себе цель подсчета статистики выпитого кофее, как для пользователями, так и для заведения.

2.4 Схема восстановления и сохранения информации

Восстановление информации[20] – это процесс замены повреждённых данных ранее сохраненной информацией (копией).

Восстановление информации является одной из функций систем управления базами данных, которая в случаях физических и системных сбоях воспроизведет базу данных в актуальном состоянии на определенный отрезок времени. Схема восстановления информации описана на рисунке 2.14.

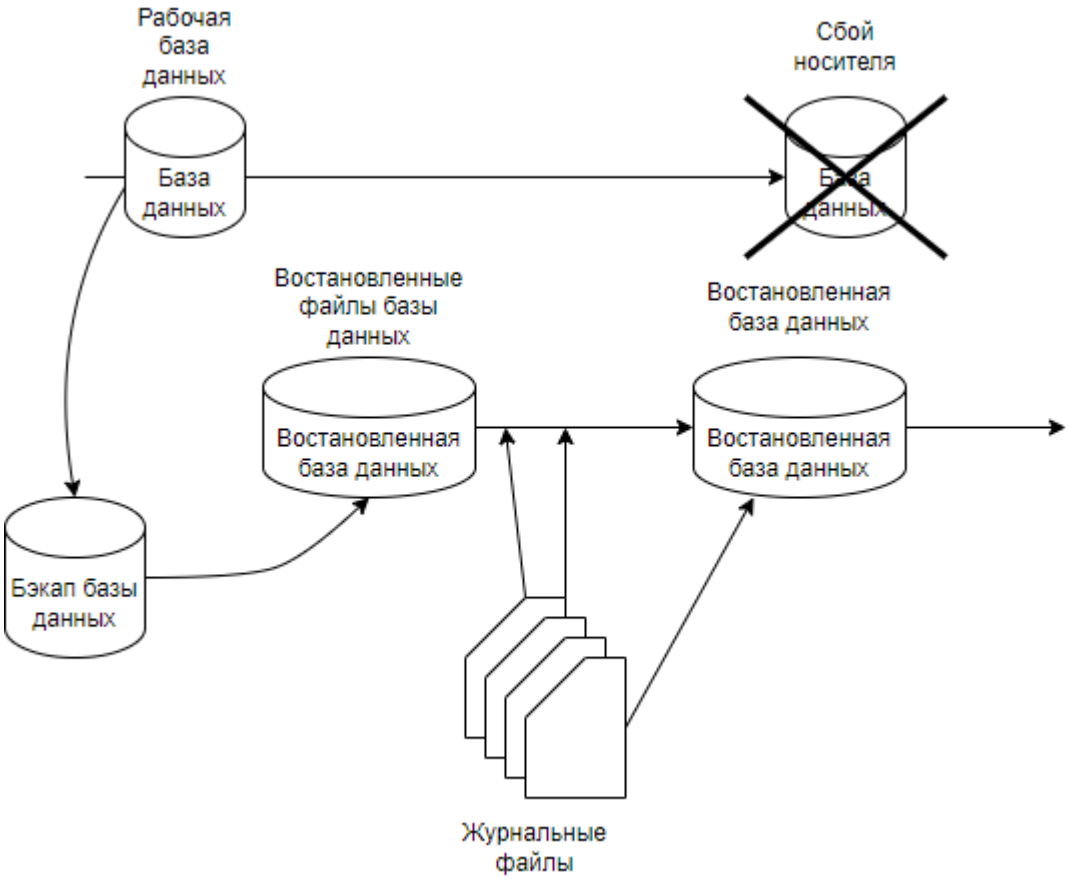

Рисунок 2.14 - Схема восстановления информации

Для корректировки восстановления используются журнальные файлы, которые записываются в системную таблицу серверной части.

3. Разработка информационной системы

3.1. Серверная сборка

Так как клиентская сборка обращается к удаленному программному обеспечению, то разработку следует начать с серверной сборки.

Произведем загрузку в проект следующих библиотек[21] для функционирования частей системы (Таблица 3.1).

Таблица 3.2. Библиотеки серверной части

Название пакета	Описание
Microsoft.AspNetCore.Authentication.JwtBearer	Включает в себя поддержку аутентификации посредством токенов[22]
Microsoft.EntityFrameworkCore	Сопоставляет модели приложения с таблицами баз данных[23]
Microsoft.EntityFrameworkCore.Design	Добавляет консольные команды для взаимодействия с удаленной базой данных
Microsoft.EntityFrameworkCore.Relational	Добавляет функционал связей для моделей[24]
Microsoft.EntityFrameworkCore.SqlServer	Подключение удаленной базы данных
Microsoft.EntityFrameworkCore.Tools	Упрощение консольных команд взаимодействия между моделями
Pomelo.EntityFrameworkCore.MySql	Словарь соответствия типов данных базы данных и типов параметров моделей

Одним из важных моментов разработки интерфейса прикладного программирования для веб-сервера является настройка изначального класса “Startup”[25], который содержит в себе обязательный метод “Configure” и необязательный “ConfigureServices”. При запуске серверного приложения,

среда выполнения ASP.Net[26] вызовет конструктор данного класса, затем метод “ConfigureServices” и в самом конце метод “Configure”.

В методе “ConfigureServices” производится настройка и регистрация сервисов, которые будут использоваться в приложении (Рисунок 3.1).

```
public void ConfigureServices(IServiceCollection services)
{
    //Указание удаленной базы данных для ОРМ системы
    var connectionString = Configuration.GetConnectionString("DefaultConnection");
    var serverVersion = ServerVersion.AutoDetect(connectionString);
    services.AddDbContext<ApplicationDbContext>(options =>
    {
        options.UseMySql(connectionString, serverVersion);
    });

    //Настройка использования контроллеров
    services.AddControllers()
        .AddJsonOptions(x => x.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.Preserve);
    //Настройка схем аутентификации и подключение системы взаимозаменяемых токенов для проверки доступа
    services.AddAuthentication(options =>
    {
        options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
        options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    })
        .AddJwtBearer(options =>
        {
            var key = Configuration.GetValue<string>("JwtConfig:Key");
            var keyBytes = Encoding.ASCII.GetBytes(key);
            options.SaveToken = true;
            options.TokenValidationParameters = new TokenValidationParameters
            {
                IssuerSigningKey = new SymmetricSecurityKey(keyBytes),
                ValidateLifetime = true,
                ValidateAudience = false,
                ValidateIssuer = false,
            };
        });
    //Подключение инъекции зависимостей для интерфейса
    services.AddScoped<IJwtTokenManager, JwtTokenManager>();
}
```

Рисунок 3.1 - Метод ConfigureServices

В методе Configure указываются какие сервисы и части системы будут использоваться в ходе работы серверной сборки.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseHttpsRedirection(); //Перенаправление на защищенное соединение

    app.UseRouting(); //Указание использование промежуточных сервисов

    app.UseAuthentication(); //Использование аутентификации
    app.UseAuthorization(); //Использование авторизации

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers(); //Взаимодействие с контроллерами
    });
}
```

Рисунок 3.2 - Метод Configure

По схеме базы данных создаем классы моделей. Структура создания моделей идентична. Приведем пример одной из классовых моделей.

```
[Index(nameof(Latitude), nameof(Longitude), IsUnique = true, Name = "Location")]

public class Place
{

    public int Id { get; set; }
    [Required]

    public string Name { get; set; }
    [Required]

    public string Address { get; set; }

    public double Rating { get; set; }
    [Required]

    public double Latitude { get; set; }
    [Required]

    public double Longitude { get; set; }

    public string Image { get; set; }

    [JsonIgnore]

    public string Details { get; set; }
    [JsonIgnore]

    public List<Favorite> Favorites { get; set; } = new List<Favorite>();
    [JsonIgnore]

    public List<AmountUsedCoffee> AmountUsedCoffee { get; set; } = new List<AmountUsedCoffee>();

    public List<PlaceTags> PlaceTags { get; set; } = new List<PlaceTags>();
    0 references | 0 changes | 0 authors, 0 changes
    public List<PlaceUserMark> PlaceUserMark { get; set; } = new List<PlaceUserMark>();
}
```

Рисунок 3.3 - Пример модели

Библиотека EntityFramework автоматически проставляет для свойств с именем Id настройку autoincrement[27] и первичный ключ.

Для свойств можно указать различные атрибуты например: атрибут “Required” запрещает содержание типа “null” в столбце таблицы, а атрибут [JsonIgnore] позволяет игнорировать свойство при передаче посредством REST[28] архитектуры. Можно указать атрибуты на уровне самого класса. На этом рисунке (Рисунок 3.3) атрибут добавит новый индекс с двумя полями, при этом индекс будет содержать только уникальные значения.

Так же EntityFramework позволяет указывать связи таблицы для моделей. По рисунку 3.3 можно увидеть, что таблица “Place” будет иметь связь один ко многим с таблицами “Favorites”, “AmountUsedCoffee”, “PlaceTags” и “PlaceUserMark”, так как содержит листы этих моделей.

После создания всех моделей следует указать их как таблицы в классе “ApplicationDbContext”, ссылку на которую записали в методе “ConfigureServices”.

Для добавления или обновления таблиц в базе данных используются две команды: “add-migration *Имя миграции*” для создания класса миграции и “update-database” для загрузки миграции на сервер базы данных. После проведенной работы будет создана готовая база данных с нужными таблицами и полями этих таблиц, без написания кода на языке запросов.

Далее следует перейти к добавлению функционала аутентификации пользователей для их идентификации и разграничения прав. Для этого создается сервис “JwtTokenManager”[29].

```
public string Authenticate(string userName, string password)
{
    var user = _context.Users.SingleOrDefault(x => x.Username.Equals(userName) && x.Password.Equals(password));
    if (user == null)
    {
        return null;
    }
    var key = _configuration.GetValue<string>("JwtConfig:Key");
    var keyBytes = Encoding.ASCII.GetBytes(key);

    var tokenHandler = new JwtSecurityTokenHandler();

    var tokenDescriptor = new SecurityTokenDescriptor()
    {
        Subject = new ClaimsIdentity(new Claim[]
        {
            new Claim(ClaimTypes.NameIdentifier, userName),
            new Claim(ClaimTypes.Role, user.Role)
        }),
        Expires = DateTime.UtcNow.AddYears(1),
        SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(keyBytes), SecurityAlgorithms.HmacSha256Signature)
    };

    var token = tokenHandler.CreateToken(tokenDescriptor);
    user.Token = tokenHandler.WriteToken(token);
    _context.Update(user);
    _context.SaveChanges();
    if (user.IsActive == false)
    {
        if (SendEmail(user))
        {
            return $"activateAccount {tokenHandler.WriteToken(token)}";
        }
        return null;
    }
    return tokenHandler.WriteToken(token);
}
```

Рисунок 3.4 - JwtTokenManager метод Authenticate

Приведенный код (Рисунок 3.4) позволяет проверить существует ли уже пользователь с заданными параметрами. После этого сгенерировать уникальный токен[30] с секретным ключом[31], который автоматически генерируется. Записать в этот токен данные, которые он будет содержать, например: логин, роль и дату истечения действия токена.

Сгенерированный токен передается только одному пользователю, что и обеспечивает идентификацию.

Теперь выделим основные контроллеры, которые будут использоваться для взаимодействия серверной части с клиентской частью.

Таблица 3.3. Описание контроллеров

Название	Описание
CoffeeController	Взаимодействие с объектами кофе и статистикой по этим объектам
PlacesController	Взаимодействие с объектами кофеен и их атрибутами
TokenController	Методы авторизации и регистрации пользователей

Название	Описание
UsersController	Взаимодействие с объектами пользователя и их атрибутами
VerificationController	Методы подтверждения аккаунта посредством email

Рассмотрим один из них. Один из самых основных это “PlacesController”. Посредством обращения к этому контроллеру при помощи REST запросов можно получить все тэги заведения (Рисунок 3.5), получить ссылку на изображение (Рисунок 3.6) или получить список всех заведений для отображения на карте (Рисунок 3.7).

```
[AllowAnonymous]
[HttpGet("gettags/{id}")]
0 references | 0 changes | 0 authors, 0 changes
public IActionResult GetTags(int id)
{
    return Ok(_context.PlaceTags.Where(x => x.PlaceId == id));
}
```

Рисунок 3.5 - Метод GetTags

```
[AllowAnonymous]
[HttpGet("getimageuri/{id}")]
0 references | Yaroslav Shevchenko, 11 days ago | 1 author, 1 change
public IActionResult GetImageUri(int id)
{
    var query = _context.Places.FirstOrDefault(x => x.Id == id);
    if (query == null)
        return NotFound();
    return Ok(query.Image);
}
```

Рисунок 3.6 - Метод GetImageUri

```
[AllowAnonymous]
[HttpGet("getpins")]
0 references | Yaroslav Shevchenko, 11 days ago | 1 author, 1 change
public IActionResult GetPins()
{
    return Ok(_context.Places.Include(x => x.PlaceTags));
}
```

Рисунок 3.7 - Метод GetPins

А также можем найти заведение по имени, по его идентификатору, получить изображение на прямую или получить все комментарии.

Для более удобного развёртывания серверной части добавим проект в хранилище докер[32]. Докер позволит провести автоматическое развёртывание при помощи нескольких команд на сервере. Контейнер в себе содержит операционную систему, данные приложения и метаданные для взаимодействия с ним.

Для того, чтобы загрузить докер образ в хранилище, следует в Visual Studio нажать кнопку “Publish” (Рисунок 3.8). При первом запуске следует ввести свои учетные данные от аккаунта “Docker hub”, в дальнейшем это не требуется.

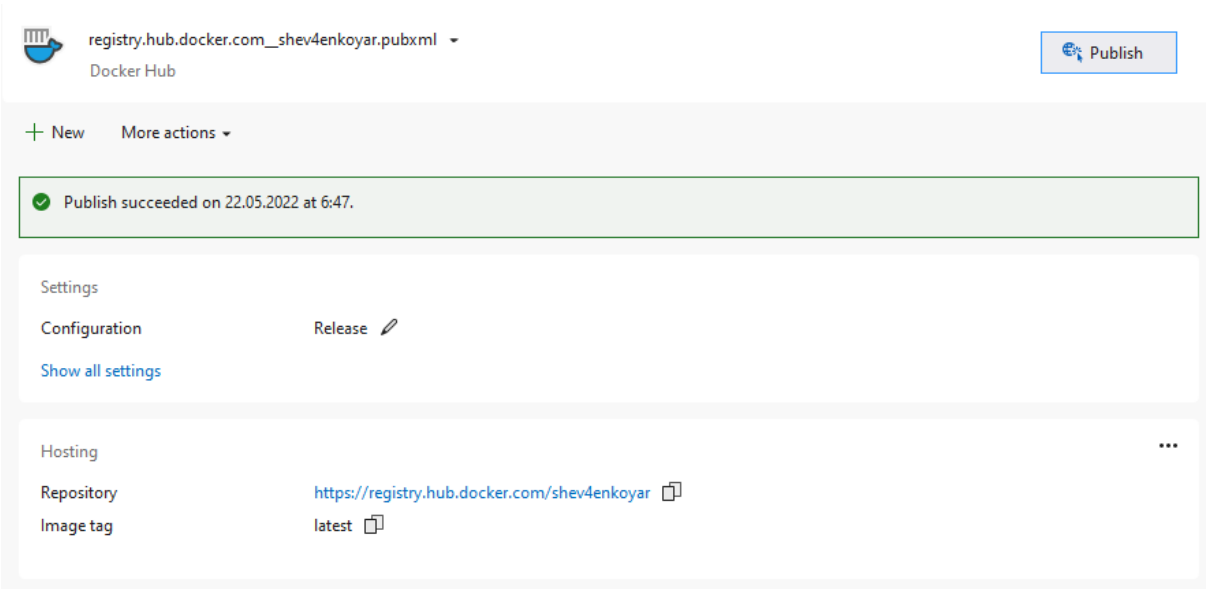

Рисунок 3.8 - Загрузка в докер хранилище

После успешной загрузки заходим в консоль сервера и прописываем команду “docker pull *ссылка на свой репозиторий:тэг*”, что позволит загрузить на сервер образ приложения.

Следующей командой прописываем “docker run -it -v “$(pwd)”:/app/Images -d -p 80:80/tcp *название репозитория*”, это запустит докер файл серверного приложения, по открытому порту, а также добавит внешнюю папку сервера для хранения с изображениями, по текущему пути.

После проведенных действий серверное приложение доступно в открытом доступе. Для проверки укажем в адресной строке любого браузера ссылку к контроллеру с параметрами.

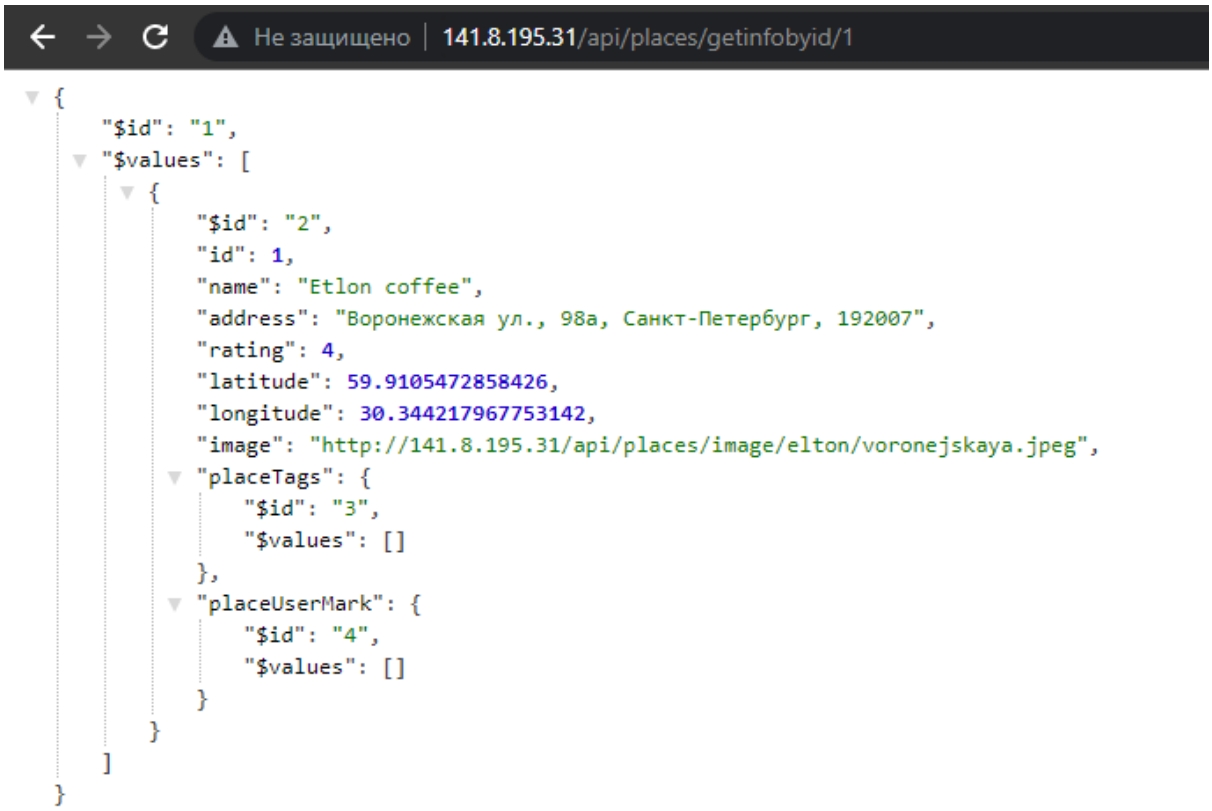

Рисунок 3.9 - Пример получения данных от сервера

Как можем наблюдать, сервер по ссылке возвращает данные в формате JSON[33] (Рисунок 3.9). Сервер работает исправно.

3.2. Клиентская сборка

В клиентской сборке будут использованы следующие библиотеки:

Таблица 3.4. Библиотеки клиентской части

Название пакета	Описание
Xamarin.Forms.Maps	Функционал взаимодействия с картой
Xamarin.Essentials	Интерфейс взаимодействия с API мобильных устройств и обновление их возобновляемых интерфейсов

Название пакета	Описание
Xamarin.CommunityToolkit	Увеличение функциональных возможностей стандартных страниц
Newtonsoft.Json	Функционал работы с JSON типами
NETStandard.Library	Пакет взаимодействия языка и фреймворков
OSRMLib	Класс REST запросов к серверу маршрутов
Refractored.MvvmHelpers	Упрощение использования событий для архитектуры MVVM
Rg.Plugins.Popup	Добавление всплывающих форм
Xamarin.Forms.PancakeView	Увеличение функционала дизайна страниц
Microcharts.Forms	Добавление диаграмм

В Xamarin.Forms[34] используется архитектура построения приложений MVVM[35], Model-View-ViewModel. Именно поэтому следует скопировать все модели с серверной части в клиентское приложение. В данной архитектуре визуальные элементы хранятся во View, в Model хранятся поля данных, а во ViewModel происходит взаимодействие всех частей приложения.

Решение клиентского приложения состоит из трех проектов.

- Обобщенный проект
- Проект Android
- Проект iOS

Удобство технологии Xamarin заключается в том, что код, который пишется в обобщенном проекте является кроссплатформенным.

Все View могут быть написаны либо на языке C#, либо на языке XAML[36]. В качестве примера приведем, часть кода написанного на языке XAML для формы поиска в приложении (Рисунок 3.10).

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    x:Class="CoffeeApp.Views.Search"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:vm="clr-namespace:CoffeeApp.ViewModels">
    <ContentPage.BindingContext>
        <vm:SearchViewModel />
    </ContentPage.BindingContext>
    <ContentPage.Background>
        <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
            <GradientStop Offset="0.1" Color="#0c1015" />
            <GradientStop Offset="1" Color="#0c1015" />
        </LinearGradientBrush>
    </ContentPage.Background>
    <ContentPage.Content>
        <Grid>
            <Frame
                Margin="0,20,0,30"
                BackgroundColor="#141921"
                CornerRadius="10"
                HorizontalOptions="Center"
                VerticalOptions="Start"
                WidthRequest="300">
                <StackLayout>
                    <Label
                        x:Name="SearchLabel"
                        FontSize="30"
                        HorizontalTextAlignment="Start"
                        TextColor="#FFFFFF" />
                    <SearchBar
                        x:Name="searchEntry"
                        PlaceholderColor="#999999"
                        TextChanged="Handle_TextChanged"
                        TextColor="#B8C6CD" />
                    <ListView
                        x:Name="EmployeeListView"
                        HasUnevenRows="true"
                        IsPullToRefreshEnabled="True"
                        ItemTapped="Handle_ItemTapped"
                        ItemsSource="{Binding MyListCollector}"
                        SeparatorVisibility="Default">
                        <ListView.ItemTemplate>
                            <DataTemplate>
                                <ViewCell>
                                    <StackLayout Orientation="Horizontal">
                                        <Image
                                            HeightRequest="50"
                                            Source="{Binding Image}"
                                            WidthRequest="50" />
                                        <StackLayout HorizontalOptions="StartAndExpand">
```

Рисунок 3.10 - Пример кода XAML

Синтаксис этого языка очень похож на синтаксис XML. Существуют открывающие и закрывающие тэги. Тэги могут быть одиночными, так и вложенными. Данный код создает следующее отображение (Рисунок 3.11)

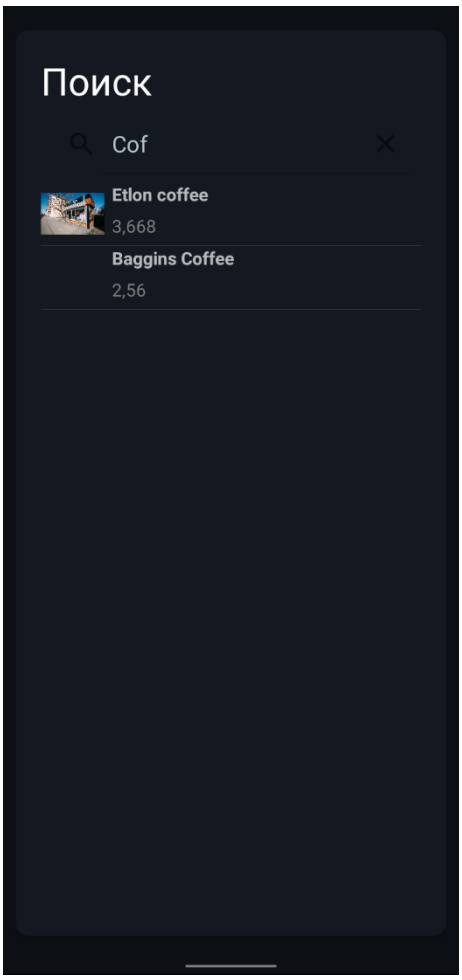

Рисунок 3.11 - Форма поиска

Форма получает данные от ViewModel, который задается в тэге “ContentPage.BindingContext”

Для указания переменной, которую следует получать из контекста используется ключевое слово “Binding”. Программа будет переходить к ViewModel для получения значения.

Для данной используется следующий ViewModel (Рисунок 3.12).

```
class SearchViewModel
{
    4 references
    public ObservableCollection<Place> MyListCollector { get; set; }
    0 references
    public SearchViewModel()
    {
        GetPlaces();
    }

    1 reference
    private void GetPlaces()
    {
        using (var client = new HttpClient())
        {
            try
            {
                var res = client.GetAsync("http://141.8.195.31/api/places/getpins").Result;
                if (res.StatusCode == System.Net.HttpStatusCode.OK)
                {
                    var content = res.Content.ReadAsStringAsync().Result;
                    List<Place> places = JsonConvert.DeserializeObject<List<Place>>(content);
                    MyListCollector = new ObservableCollection<Place>();
                    foreach (Place p in places)
                    {
                        MyListCollector.Add(p);
                    }
                }

            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }
    }

}
```

Рисунок 3.12 - ViewModel поиска

Так как ранее, был подключен пакет упрощающий взаимодействие с архитектурой MVVM, для указания переменных контекста используется всего одна строка. В методе “GetPlaces” на рисунке 3.12, происходит взаимодействие клиентской сборки с серверной. Производится обращение к методу “getpins” контроллера “Places”. В течении выполнения этого метода будет заполнена коллекция “MyListCollector”, которая, в свою очередь, и является ранее прикреплённой переменной контекста во View.

Для других View структура отображения идентична. Создается отображение, происходит выполнение ViewModel для получения данных моделей.

Следует привести также пример формы навигации, так как она является одной из самых главных и связующих форм в приложении.

Во ViewModel навигации находятся метод получения и отображения текущей позиции пользователя (Рисунок 3.13), а также происходит вызов метода получения всех точек карты и дальнейшее их добавление в коллекцию.

```
public async void DisplayCurLocation(CustomPin customPin = null)
{
    var request = new GeolocationRequest(GeolocationAccuracy.High);
    Location location = await Geolocation.GetLocationAsync(request);
    if (location != null)
    {
        if (customPin == null)
            _currPosition = new Position(location.Latitude, location.Longitude);
        else
            _currPosition = customPin.Position;
        MapSpan mapSpan = MapSpan.FromCenterAndRadius(_currPosition, Distance.FromKilometers(0.5));
        customMap.MoveToRegion(mapSpan);
    }

    customMap.CustomPins = GetAllPins();
    foreach (var item in customMap.CustomPins)
    {
        item.MarkerClicked += Item_MarkerClicked;
        customMap.Pins.Add(item);
    }
    if (customPin != null)
    {
        var pop = new CustomPopup(customPin);
        await App.Current.MainPage.Navigation.PushPopupAsync(pop, true);
    }
}
```

Рисунок 3.13 - Метод DisplayCurLocation

Каждая точка подписывается на событие “MarkerClicked”, при выполнении этого события будет произведен вызов метода Item_MarkerClicked (Рисунок 3.14). В этом методе происходит перенаправление на всплывающую форму информации о заведении.

```
private void Item_MarkerClicked(object sender, PinClickedEventArgs e)
{
    var pop = new CustomPopup(sender);
    App.Current.MainPage.Navigation.PushPopupAsync(pop, true);
}
```

Рисунок 3.14 - Метод Item_MarkerClicked

В этой всплывающей форме может быть нажата кнопка “Навигации”, которая открывает форму с построенным маршрутом до точки. Эта форма, для получения промежуточных точек обращается к сервису “OSRMService”[37] (Рисунок 3.15).

В метод “GetRoute” передаются стартовая позиция и конечная позиция. Стартовой позицией будет являться позиция пользователя, а конечной позиция заведения. Этими данными заполняется свойство “Coordinates”.

После чего, происходит вызов метода “Call”, который подсчитывает промежуточные точки.

Далее полученные данные следует привести к виду, который используется в приложении и вернуть их для проведения линии пути между ними[38].

```
public async Task<List<Position>> GetRoute(Location startPos, Location endPos)
{
    routeService.Coordinates = new List<Location> { startPos, endPos };

    var response = await routeService.Call();

    var points = response.Routes[0].Geometry;

    List<Position> positions = new List<Position>();

    foreach (var point in points)
    {
        positions.Add(new Position(point.Latitude, point.Longitude));
        Console.WriteLine($"Point: {point.Latitude}; {point.Longitude}");
    }
    return positions;
}
```

Рисунок 3.15 - Метод GetRoute

Таким образом происходи получение пути и построение маршрута к нужной точке.

3.3. Интерфейс взаимодействия с информационной системой

Когда любой пользователь открывает приложение, первым делом он видит не функционал, не возможности приложения, а его интерфейс[39]. Именно поэтому взаимодействие пользователя с приложением посредством интерфейса очень важно.

При запуске приложения пользователь попадает на главное окно. Главное окно должно отображать либо главный функционал системы, либо меню из которого можно быстро добраться до главных возможностей приложения.

В данной работе, пользователь попадает после запуска на форму карты (Рисунок 3.16), на которой также находится меню, позволяющее переместится в другие участки приложения.

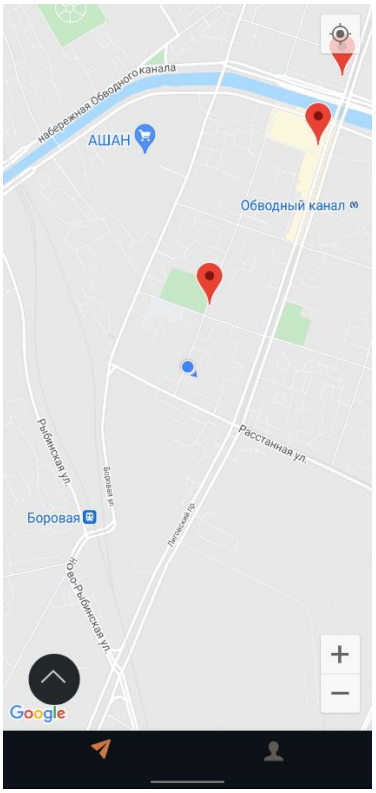

Рисунок 3.16 - Форма карты

С данной формы пользователь может попасть на форму входа в приложение (Рисунок 3.17).

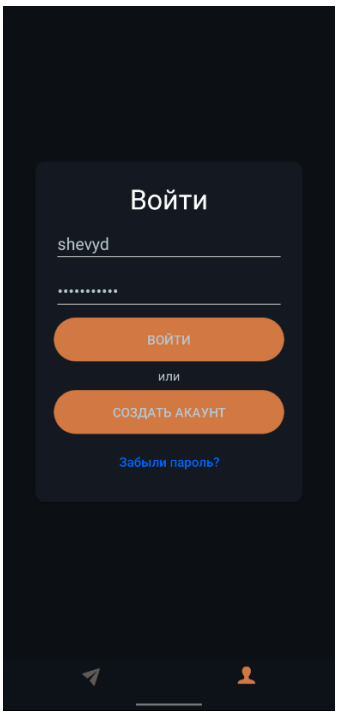

Рисунок 3.17 - Форма входа и регистрации

После ввода личных данных его перенаправит на форму аккаунта (Рисунок 3.18), где содержится его статистика и информация.

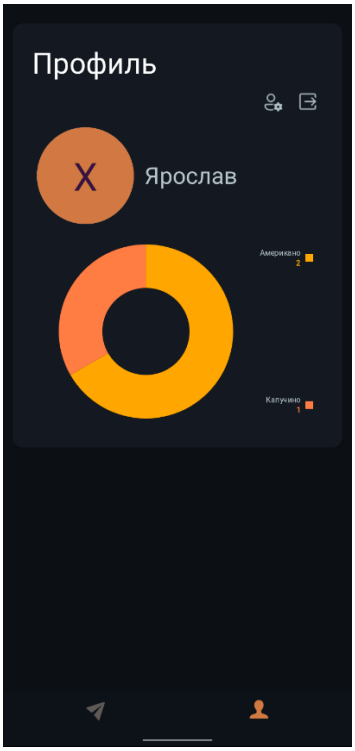

Рисунок 3.18 - Профиль пользователя

На форме навигации существует возможность открыть дополнительное меню (Рисунок 3.19) в котором находятся кнопки перехода на форму закладок

(Рисунок 3.20) и форму поиска (Рисунок 3.21), с которых можно перейти к всплывающему меню заведения.

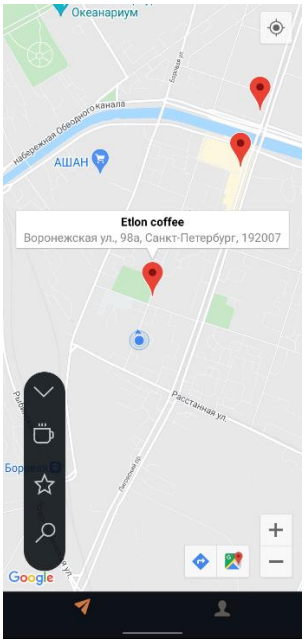

Рисунок 3.19 - Дополнительное меню

В дополнительном меню находятся кнопки переводящие на другие сегменты системы.

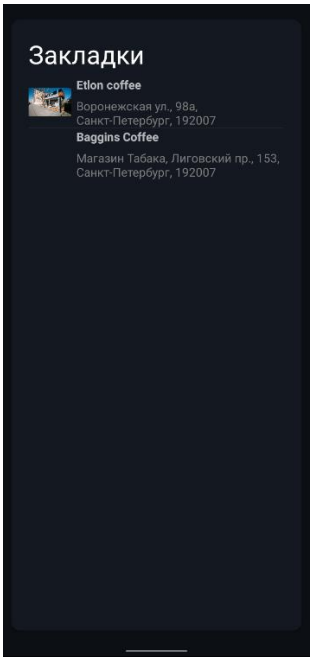

Рисунок 3.20 - Форма закладок

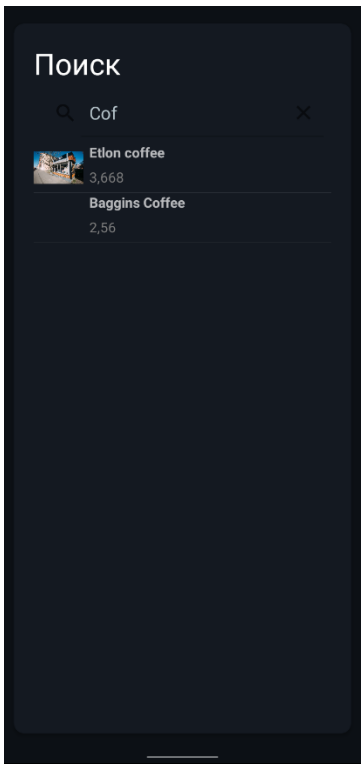

Рисунок 3.213 - Форма поиска

Если нажать на иконку заведения на форме карты, то откроется всплывающее меню заведения (Рисунок 3.22).

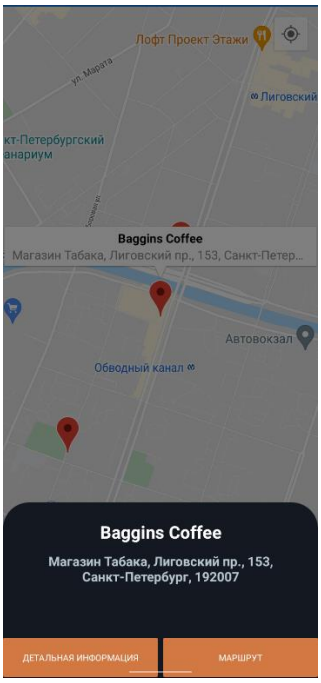

Рисунок 3.224 - Всплывающая форма

На данной форме можно открыть подробную информацию (Рисунок 3.23) или сразу построить маршрут (Рисунок 3.25).

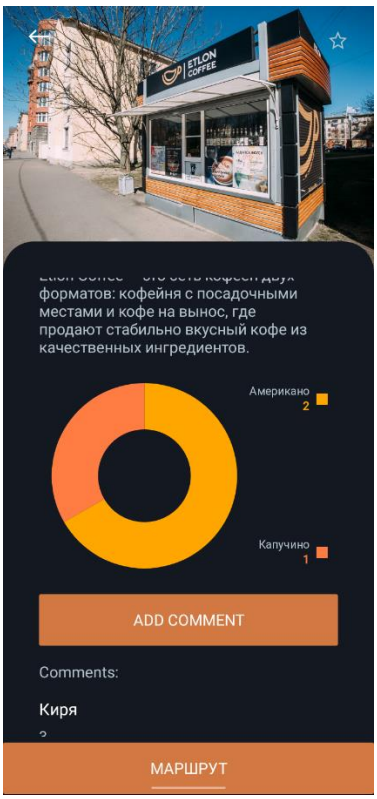

Рисунок 3.23 - Профиль заведения

В профиле заведения авторизированным пользователям предоставляется возможность добавить место в закладки по клику на иконку звезды справа сверху, а также оставить отзыв с оценкой о заведении. По клику на кнопку оставления комментария открывается форма отзыва. После добавления отзыва на форме заведения появится новый комментарий с оценкой пользователя (Рисунок 3.24).

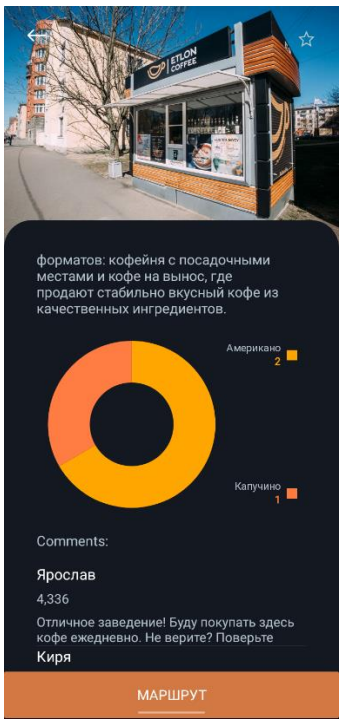

Рисунок 3.24 - Новый комментарий на форме заведения

По нажатию на кнопку маршрута открывается форма маршрута (Рисунок 3.25) с проложенным путем к заведению от текущего положения пользователя.

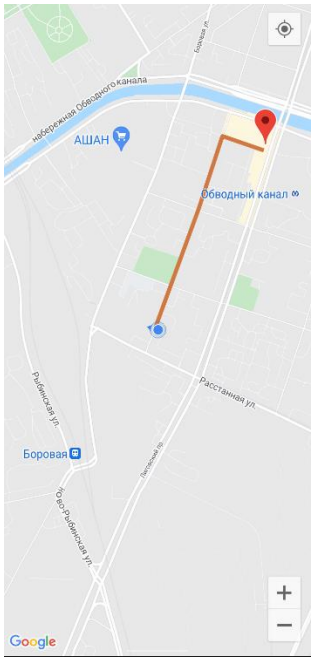

Рисунок 3.25 - Форма маршрута

Форма маршрута отображает пользователя, конечную точку и путь до точки на карте.

3.4. Надежность программного обеспечения

Начнем с того, что такое надежность программного обеспечения. Надежность ПО[40] – это вероятность его работы без отказов в течении определенного времени.

Существуют основные классы ошибок в программах:

- Ошибки вычислений – ошибки данного класса содержаться в закодированных математических выражениях или получаемых с их помощью результатах. Примерами таких ошибок является неверное преобразование типов переменных, неверный знак операции, ошибка в выражении индекса, переполнение или потеря значимости при вычислениях.
- Логические ошибки – являются причиной искажения алгоритма решения задачи. Такого рода ошибки возникают в связи с неверной передачей управления, неверном задании диапазона изменения параметров цикла, неверных условий и т.д.
- Ошибки ввода-вывода – связаны с такими действиями, как управление вводом-выводом, формирование выходных записей и определение размеров записей.
- Ошибки манипулирования данными – примерами таких ошибок являются неверно определенное число элементов данных, неверные начальные значения, присвоенные данным, неверно указанная длина операнда, имя переменной и т.д.

Протестируем клиентское приложение с помощью метода оценки надежности, основанном на модели Шумана.

В модели предполагается, что тестирование проводится в несколько этапов, каждый из которых представляет собой выполнение программы по набору тестовых данных. Выявленные в течение этапа тестирования ошибки регистрируются, но не исправляются. По завершении этапа исправляются все обнаруженные ошибки, корректируются тестовые наборы и проводится новый этап тестирования. Составим таблицу этапов тестирования (Таблица 3.5).

Таблица 3.5. Этапы тестирования клиентского приложения

Этапы	t, ч	m
1	30	3
2	20	2
3	30	1
4	20	1

Определим первоначальное количество ошибок N в программе из зависимости (Формула 3.1).

$$\sum_{i=1}^{k} m_i * \frac{\sum_{i=1}^{k} t_i}{\sum_{i=1}^{k} \frac{m_i}{N - n_{i-1}}} = \sum_{i=1}^{k} (N - n_{i-1}). \quad (3.1)$$

Благодаря данному выражению определили, что N равно 10.

Надежность можно найти по формуле (Формула 3.2)

$$R(t) = e^{-\lambda * t}. \quad (3.2)$$

Где интенсивность отказов $(-\lambda)$ равна следующему выражению (Формула 3.3)

$$\lambda = (N - n) * \frac{\sum_{i=1}^{k} \frac{m_i}{N - n_{i-1}}}{\sum_{i=1}^{k} t_i}. \quad (3.3)$$

Подставив значения получим интенсивность отказов равную 0,010357143.

Теперь подсчитаем, какая будет надежность у клиентского приложения по модели Шумана (Формула 3.4).

$$R(10) = e^{-0,010357143 * 10} = 0,9016116454. \quad (3.4)$$

Итоговой надёжностью будет значение равное 0,9016116454

Далее подсчитаем надежность для сервера авторизации.

Таблица 3.6. Этапы тестирования приложения сервера авторизации

Этапы	t, ч	m
1	30	5
2	20	2
3	30	1
4	20	1

- Первоначальное количество ошибок, по заданным параметрам равно 10
- Интенсивность отказов равна 0,01734
- Итоговая надежность за 10 часов равна 0,8408012271

Подсчитаем надежность для сервера базы данных

Таблица 3.7. Этапы тестирования приложения сервера базы данных

Этапы	t, ч	m
1	30	2
2	20	1
3	30	1
4	20	1

- Первоначальное количество ошибок, по заданным параметрам равно 11
- Интенсивность отказов равна 0,00560786
- Итоговая надежность за 10 часов равна 0,9454648194

Подведем таблицу надежностей всех систем

Таблица 3.8. Надежность систем по методу Шумана

Система	Надежность
Клиентское приложение	0,9016116454
Приложение сервера авторизации	0,8408012271
Система	Надежность
Приложение сервера базы данных	0,9454648194

Протестируем клиентское приложение при помощи метода оценки надежности, основанном на модели Нельсона-Коркорэна. Это обобщенная модель модели Нельсона и модели Коркорэна.

Приложение запустим сто раз и проверим, сколько будет запусков с отказом.

После запуска приложения на одном устройстве, количество запусков с отказом равно 2. Тогда, можем подсчитать надежность приложения по формуле (Формула 3.5).

$$R(n) = 1 - \frac{n^{-}}{n}. \quad (3.5)$$

Где в числителе находится количество запусков с отказом, а в знаменателе общее количество запусков. Рассчитаем вероятность (Формула 3.6).

$$R(100) = 1 - \frac{2}{100} = 0{,}98. \quad (3.6)$$

Из этого следует, что надежность клиентского приложения равна 0,98.

С аналогичными параметрами, подсчитаем надежность для JWT сервера и сервера базы данных.

Для JWT сервера ошибочных запусков было 1 (Формула 3.7).

$$R(100) = 1 - \frac{1}{100} = 0{,}99. \quad (3.7)$$

Надежность сервера авторизации равна 0,99.

Для сервера базы данных, количество проблемных запусков равна 2 (Формула 3.8).

$$R(100) = 1 - \frac{2}{100} = 0{,}98. \quad (8)$$

Надежность сервера базы данных равна 0,98.

Таким образом, можем составить итоговую таблицу надежности для данных систем.

Таблица 3.9. Надежность систем по методу Нельсона-Коркорэна

Система	Надежность
Клиентское приложение	0,98
Приложение сервера авторизации	0,99
Приложение сервера базы данных	0,98

Вся система состоит из трех элементов: клиент, сервер базы данных и сервер авторизации.

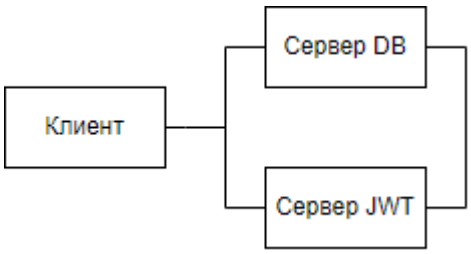

Рисунок 5 - Схема взаимодействия частей системы

Сервера подключены через параллельное соединение, а клиент подключен через последовательное к ним.

Вероятность безотказной работы всей системы по модели Шумана равна (Формула 3.9).

$$P(t) = 0{,}9016116454 * (1 - ((1 - 0{,}8408012271) * (1 - 0{,}9454648194))) = 0{,}8937839128. \quad (3.9)$$

Вероятность безотказной работы всего программного комплекса по модели Нельсона-Коркорэна равна (Формула 3.10).

$$P(t) = 0{,}98 * ((1 - (1 - 0{,}99) * (1 - 0{,}98))) = 0{,}979804. \quad (3.10)$$

Следуя из полученных вероятностей безотказной работы, можем получить среднюю вероятность (Формула 3.11)

$$P = \frac{(0{,}8937839128 + 0{,}979804)}{2} = 0{,}9367939564. \quad (3.11)$$

Из проделанного расчета следует, что вся информационная система имеет надежность равную 0,9367939564. Этой надежности достаточно для длительного пользования системой безошибочно.

Заключение

Говоря о кофе, мы понимаем важность этого напитка, ценим его вкус и аромат. Но остается одно, для того, чтобы сполна ощутить данный напиток, нужно найти место в котором подают лучший продукт. Эту задачу способны решить две системы в совокупности: система навигации и маршрутизации, а также система оценивания и комментирования.

Из анализа информационной системы вывели, что с большой долей вероятности приложение будет пользоваться спросом, что существуют некоторые способы монетизации информационной системы, а также, в совокупности факторов, прикладное обеспечение может являться жизнеспособным продуктом. Во время анализа и проектирования были разработаны диаграммы, схемы и сценарии, которые упростили дальнейшую разработку.

Перейдя к разработке, было логично остановится на паттерне разработки “MVC”, из-за того, что в этой архитектуре существует строгое разграничение данных, визуальных форм и управляющих классов. Благодаря этой структуре внешнее использование базы данных сильно упростилось. Некоторая часть времени была затрачена на улучшение и упрощение пользовательского интерфейса. Так как, пользователя недолжно ничего отвлекать от запланированного им действия по использованию программы.

Обобщив все выше сказанное был получен готовый прототип, который протестировали на надежность информационной системы. Текущая версия программного комплекса способна находить кофейни, прокладывать маршруты, авторизовывать и регистрировать аккаунты, добавлять комментарии к заведениям, для дальнейшего просмотра и оценивания другими пользователями.

Список использованных источников и литературы

1. Системный анализ [Электронный ресурс] — Электрон. ст. — Системный анализ — URL: http://systems-analysis.ru/systems_analysis.html, свободный. — Яз. рус. — (Дата обращ. 03.05.2021).
2. Руководство по применению ISA при аудите малых и средних организаций: учеб. пособие / Международная Федерация Бухгалтеров - IFAC, 2014. - 234 с
3. SWOT-анализ [Электронный ресурс] / Кирилл Осипов — Электрон. ст. — Calltouch Blog — URL: https://blog.calltouch.ru/glossary/swot-analiz/, свободный. — Яз. рус. — (Дата обращ. 04.05.2021).
4. Business process reengineering (BPR) [Электронный ресурс] / Linda Tucci — Электрон. ст. — TechTarget — URL: https://www.techtarget.com/searchcio/definition/business-process-reengineering, свободный. — Яз. анг. — (Дата обращ. 04.05.2021).
5. Давайте знакомится с программированием [Электронный ресурс]. — Режим доступа: https://prog-cpp.ru/uml-classes/
6. Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс] / Purojok — Электрон. ст. — Хабр — URL: https://habr.com/ru/post/566218/, свободный. — Яз. рус. — (Дата обращ. 04.05.2021).
7. Studbooks. Студенческая библиотека онлайн [Электронный ресурс]. — Режим доступа: https://studbooks.net/2029825/informatika/opisanie_pretsedentov
8. Creately. Крейтли Блог [Электронный ресурс]. — Режим доступа: https://creately.com/blog/ru/uncategorized-ru/%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5-%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5-%D0%BF%D0%BE-%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%

BC%D0%B5-
%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2/

9. Документация по платформе Flexberry [Электронный ресурс]. — Режим доступа: https://flexberry.github.io/ru/fd_sequence-diagram.html
10. Creately. Крейтли Блог [Электронный ресурс]. — Режим доступа: https://creately.com/blog/ru/uncategorized-ru/%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5-%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5-%D0%BF%D0%BE-%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D0%B9-%D0%B4%D0%B8%D0%B0%D0%B3/
11. СтудИзба [Электронный ресурс]. — Режим доступа: https://studizba.com/lectures/informatika-i-programmirovanie/sovremennye-tehnologii-programmirovaniya/4992-diagrammy-razvertyvaniya.html
12. Microsoft Visio 2010 [Электронный ресурс]. — Режим доступа: https://support.microsoft.com/ru-ru/office/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D1%81%D1%85%D0%B5%D0%BC-%D1%80%D0%B0%D0%B7%D0%B2%D0%B5%D1%80%D1%82%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F-uml-60811688-d811-4387-9715-c1f02f30b125
13. MDN Web Docs [Электронный ресурс]. — Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Introduction
14. Руководство по TCP/IP для начинающих [Электронный ресурс] / T-Rex — Электрон. ст. — Selecteld — URL: https://selectel.ru/blog/tcp-ip-for-beginners/, свободный. — Яз. рус. — (Дата обращ. 06.05.2021).

15. Comindware: Workflow software [Электронный ресурс]. — Режим доступа: https://www.comindware.com/ru/blog-%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F-bpmn-2-0-%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B-%D0%B8-%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5
16. Business Studio [Электронный ресурс]. — Режим доступа: https://www.businessstudio.ru/wiki/docs/current/doku.php/ru/csdesign/bpmodeling/bpmn_notation
17. Lucidchart. Интеллектуальное построение диаграмм [Электронный ресурс]. — Режим доступа: https://www.lucidchart.com/pages/ru/erd-%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0
18. Нормализация [Электронный ресурс] / admin — Электрон. ст. — Metanit: Сайт о программировании — URL: https://metanit.com/sql/tutorial/2.1.php#:~:text=%D0%9F%D0%B5%D1%80%D0%B2%D0%B0%D1%8F%20%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F%20%D1%84%D0%BE%D1%80%D0%BC%D0%B0%20(1NF)%20%D0%BF%D1%80%D0%B5%D0%B4%D0%BF%D0%BE%D0%BB%D0%B0%D0%B3%D0%B0%D0%B5%D1%82,%D0%B4%D0%BE%D0%BB%D0%B6%D0%B5%D0%BD%20%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B5%D1%82%D1%8C%20%D0%BE%D1%82%20%D0%BF%D0%B5%D1%80%D0%B2%D0%B8%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BA%D0%BB%D1%8E%D1%87%D0%B0. — Яз. рус. — (Дата обращ. 08.05.2021).
19. Заметки IT специалиста: Блог о компьютерах и программировании для начинающих [Электронный ресурс]. — Режим доступа: https://info-comp.ru/second-normal-form

20. Учебно-методический комплекс [Электронный ресурс]. — Режим доступа: http://bseu.by/it/tohod/sdo4.htm
21. Microsoft Docs .Net [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/standard/class-library-overview
22. Авторизация с помощью JWT-токенов [Электронный ресурс] / admin — Электрон. ст. — Metanit: Сайт о программировании — URL: https://metanit.com/sharp/aspnet5/23.7.php — Яз. рус. — (Дата обращ. 10.05.2021).
23. Microsoft Docs .Net [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/ru-ru/ef/core/
24. Создание моделей в Entity Framework Core [Электронный ресурс] / admin — Электрон. ст. — Metanit: Сайт о программировании — URL: https://metanit.com/sharp/entityframeworkcore/2.3.php — Яз. рус. — (Дата обращ. 10.05.2021).
25. Что из себя представляет класс Startup и Program.cs в ASP.NET Core [Электронный ресурс] / MaxRokatansky — Электрон. ст. — Хабр — URL: https://habr.com/ru/company/otus/blog/542494/, свободный. — Яз. рус. — (Дата обращ. 10.05.2021).
26. Microsoft Docs .Net [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0
27. Microsoft Docs .Net [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/ru-ru/ef/core/modeling/generated-properties?tabs=data-annotations
28. REST, что же ты такое? [Электронный ресурс] / Андрей Бураков — Электрон. ст. — System.Education — URL: https://systems.education/what-is-rest, свободный. — Яз. рус. — (Дата обращ. 11.05.2021).
29. LexikJWTAuthenticationBundle [Электронный ресурс] / chalasr — Электрон. ст. — System.Education — URL:

https://github.com/lexik/LexikJWTAuthenticationBundle/blob/2.x/Resources/doc/index.rst#id11, свободный. — Яз. анг. — (Дата обращ. 11.05.2021).

30. Пять простых шагов для понимания JSON Web Tokens (JWT) [Электронный ресурс] / Borisyuzhakov — Электрон. ст. — Хабр — URL: https://habr.com/ru/post/340146/, свободный. — Яз. рус. — (Дата обращ. 12.05.2021).
31. Cyber Polygon [Электронный ресурс]. — Режим доступа: https://cyberpolygon.com/ru/materials/security-of-json-web-tokens-jwt/
32. Что такое Docker: для чего он нужен и где используется [Электронный ресурс] / T-Rex — Электрон. ст. — Selecteld — URL: https://selectel.ru/blog/what-is-docker/, свободный. — Яз. рус. — (Дата обращ. 13.05.2021).
33. JSON [Электронный ресурс]. — Режим доступа: https://www.json.org/json-en.html
34. Подробно о Xamarin [Электронный ресурс] / Andrey Shelehim — Электрон. ст. — Хабр — URL: https://habr.com/ru/post/188130/, свободный. — Яз. рус. — (Дата обращ. 13.05.2021).
35. MVVM: полное понимание (+WPF) Часть 1 [Электронный ресурс] / oggr — Электрон. ст. — Хабр — URL: https://habr.com/ru/post/338518/, свободный. — Яз. рус. — (Дата обращ. 13.05.2021).
36. XAML [Электронный ресурс] / admin — Электрон. ст. — Metanit: Сайт о программировании — URL: https://metanit.com/sharp/wpf/2.php — Яз. рус. — (Дата обращ. 13.05.2021).
37. OSRM API Documentation [Электронный ресурс]. — Режим доступа: http://project-osrm.org/docs/v5.5.1/api/#general-options
38. Делаем маршрутизацию (роутинг) на OpenStreetMap. Введение [Электронный ресурс] / N-Cube — Электрон. ст. — Хабр — URL: https://habr.com/ru/post/511144/, свободный. — Яз. рус. — (Дата обращ. 14.05.2021).

39. 16 BEST TECHNIQUES FOR CREATING A USER-FRIENDLY INTERFACE [Электронный ресурс] / Sofia V. — Электрон. ст. — Geniusee — URL: https://geniusee.com/single-blog/16-techniques-for-creating-a-user-friendly-interface#:~:text=interface%20for%20usability).-,What%20is%20a%20user%2Dfriendly%20interface%3F,provide%20a%20perfect%20user%20experience., свободный. — Яз. рус. — (Дата обращ. 15.05.2021).
40. Функциональная устойчивость информационных систем [Электронный ресурс] — Электрон. ст. — Системный анализ — URL: http://systems-analysis.ru/systems_analysis.html, свободный. — Яз. рус. — (Дата обращ. 15.05.2021).

Приложение А

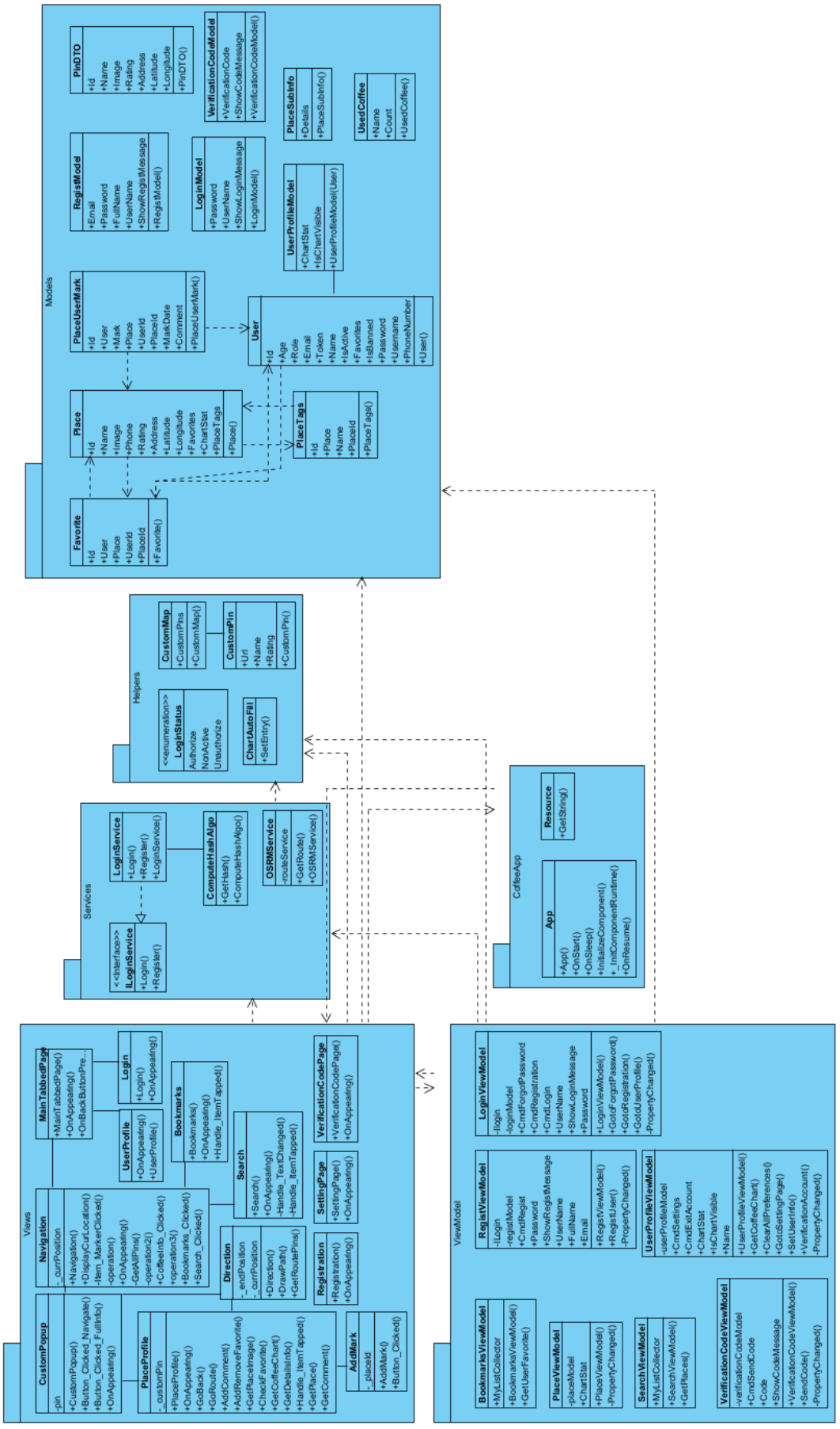

Рисунок А.1 – Классовая диаграмма клиентской части

Приложение Б

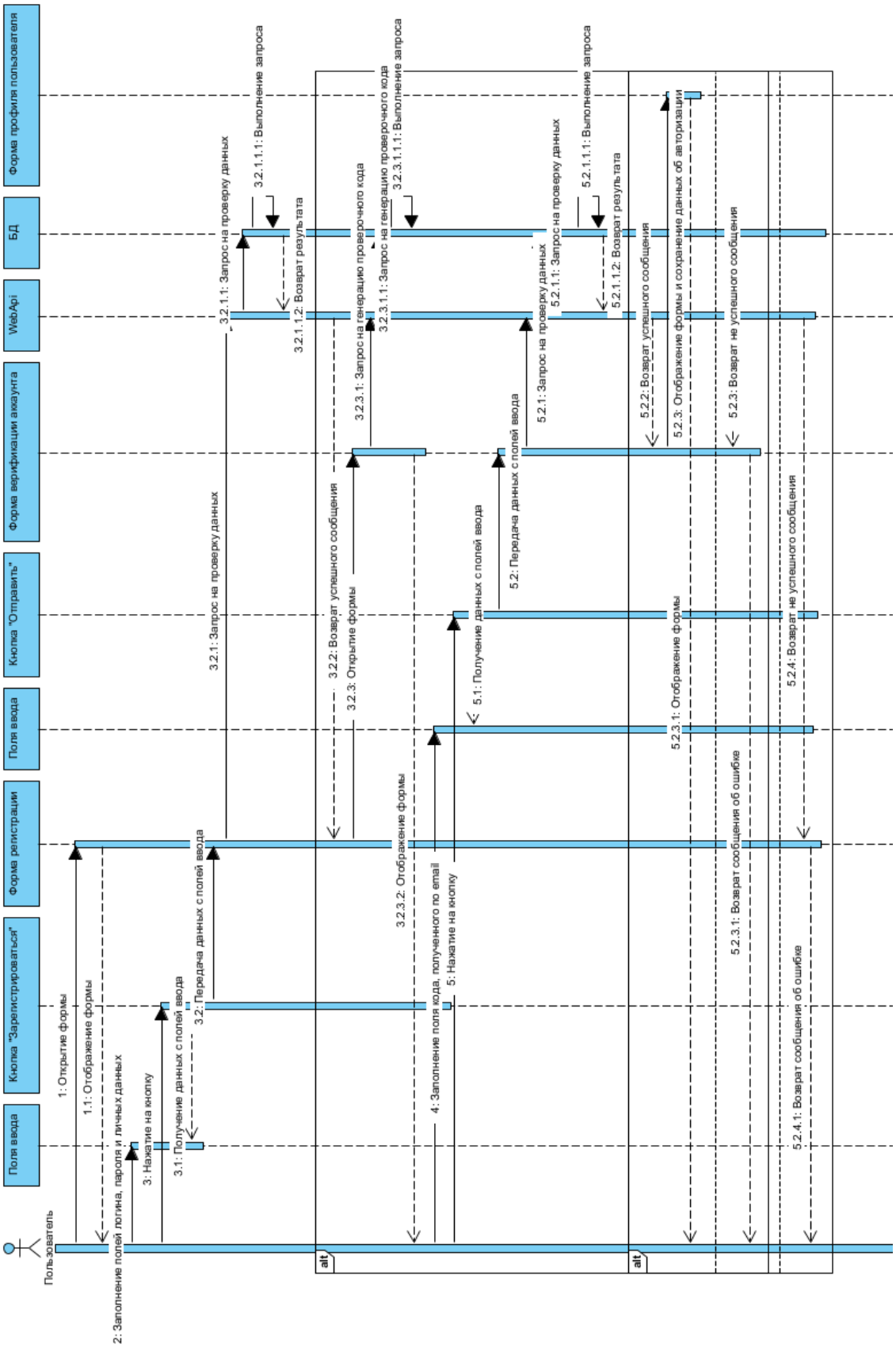

Рисунок Б.1 - Диаграмма последовательности. Регистрация