МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Корпоративные информационные системы
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильной версии райдшерингового сервиса»

Студент А.А. Скобов
(И.О. Фамилия) (личная подпись)

Руководитель А.П. Тонких
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Тема: «Разработка мобильной версии райдшерингового сервиса».

Объектом исследования является процесс совершения совместных поездок студентов учебного заведения. В исследовании рассматриваются задачи разработки алгоритма указанного процесса и разработка программной реализации разрабатываемого алгоритма на node.js, а также react.js.

Структура ВКР представлена введением, тремя главами, заключением, списком литературы.

В введении описывается актуальность проводимого исследования, а также рынок, для которого реализовывается данная система. В первой главе проводится описание и бизнес—анализ реализуемого сервиса, связанного с транспортно-логистическими системами, а также реализация интерфейса пользователей с постановкой конкретных задач.

Во второй главе описывается реализация программного обеспечения по разработанному алгоритму и взаимодействию с внешними системами с использованием особенностей node.js и react.js.

В третьей главе приводятся результаты альфа и бета тестирования, визуализируются результаты работы реализованного программного решения.

Данная бакалаврская работа состоит из 13 рисунков, 10 таблиц, списка из 7 источников на иностранном языке, 15 источников на русском языке и 4 приложений.

Abstract

The graduation work is devoted to development of a mobile version of a ride-sharing service.

The object of the research is the process of making joint trips of students of an educational institution. The study considers the tasks of developing an algorithm for the specified process and the development of a software implementation of the developed algorithm in node.js, as well as react.js.

The work consists of an introduction, three chapters, a conclusion, and a list of references.

The introduction describes the relevance of the research being conducted, as well as the market for which the system is being implemented. The first chapter provides a description and business analysis of the implemented service associated with transport and logistics systems, as well as the implementation of the user interface with the formulation of specific tasks.

The second chapter describes the implementation of software according to the developed algorithm and interaction with external systems using the features of node.js and react.js.

The third chapter presents the results of alpha and beta testing, visualizes the results of the implemented software solution.

The work also contains 13 figures, 10 tables, a list of 7 references in a foreign language and 15 references in Russian, and 4 appendices.

Содержание

Введение

Актуальность выбранной темы подтверждается тем, что направление исследований связано с развитием услуг и систем на основе интеллектуальных платформ, сетей и инфраструктуры в логистике людей, и вещей, относящихся к рынку НТИ «Автонет». Это дает широкие возможности коммерциализации результатов исследований, благодаря научно-техническому подходу, используемый для реализации данного проекта.

Основанием для выполнения бакалаврской работы является договор (соглашение) с ФГБУ «Фонд содействия развитию малых форм предприятий в научно-технической сфере» от 31.07.2019 г. № 14742ГУ/2019 о предоставлении гранта на выполнение научно–исследовательских работ и оценку перспектив коммерческого использования результатов в рамках реализации инновационного проекта.

Объектом исследования бакалаврской работы является мобильная райдшеринговая система для потенциальных пассажиров и автовладельцев — web–приложение, содержимым которого является функциональная система поиска, бронирования и создания поездок.

Предметом исследования является экспертная система для автоматического бронирования и создания поездок, для реализации функциональной райдшеринговой системы. Поиск поездок осуществляется в системе следующим образом: пользователь указывает точку и время на карте, после чего система в определенном квадрате находит все точки, которые попадают в квадрат вокруг точки размером километр на километр, а также удовлетворяют требованиям временного промежутка. Создание поездки начинается с добавления всей необходимой информации о водителе, автомобиле и самой поездке.

Целью бакалаврской работы является создание программного обеспечения, предназначенного для осуществления совместных поездок в

городской среде с применением современных технологий проектирования и создания web–ресурсов. Программное обеспечение представляет из себя сервис для совместных поездок, который позволит снизить общую загруженность на дороге из-за меньшего числа машин, а также позволит скоординировать водителей и пассажиров между собой.

Задачи:

1) Изучить известные сервисы для совершения поездок в городской среде;

2) Проанализировать научные работы в области транспортно-логистических экспертных систем;

3) Разработка алгоритма построения маршрутов;

4) Создание архитектуры приложения на основе методов бизнес-анализа;

5) Реализовать программное обеспечение по разработанной архитектуре.

Методом исследования при разработке web-приложения являлись теоретические основы web-дизайна и web-программирования, а также исследование возможных вариантов реализации экспертной системы для автоматического бронирования поездок. Математический анализ алгоритмов для построения и хранения оптимальных маршрутов со временем их старта и окончания, также с учетом времени во всех промежуточных точках.

В первой главе проведен анализ информационной системы, составлено описание и требования.

Во второй главе описывается процесс построения архитектуры и реализации программного обеспечения для автоматизации процесса поиска и создания поездок.

В третьей главе проведено альфа и бета тестирование, реализованного программного обеспечения, в результате которых были устранены все найденные ошибки сервиса.

1 Анализ проблемы перемещения студентов до образовательных учереждений

1.1 Анализ определения актуальности проблемы

Решаемой задачей является разработка web–приложения с наличием клиентской части, панелью администратора, подключенной базой данных для хранения информации и возможностью регистрации пользователей путем создания аккаунта с использованием электронной почты. Начнем с реализации панели администратора.

Функциями администратора являются:

— модерирование пользователей т.е. возможное редактирование, либо удаление информации профилей, а также блокирование пользователей;

— модерирование поездок;

— модерирование отзывов;

— модерирование шаблонов поездок;

— модерирование точек поездок;

— отслеживание процессов в web-приложении благодаря логированию данных.

Функциями пользователей являются:

— создание учетных записей пользователей;

— создание поездок;

— бронирование поездок;

— поиск поездок;

— оставление отзывов;

В работе web–приложения используются следующие ресурсы:

— OpenStreetMap (OSM) – web-картографический открытый ресурс, позволяющий получать информацию из базы данных карт городов,

созданных за счет персональных GPS-трекеров, аэрофотографии, видеозаписи, спутниковых снимков;

— база данных, в которой хранятся учетные записи пользователей, данные о поездках, данные о шаблонах поездок, данные логирования;

— сервер, обрабатывающий запросы, поступающие из клиентской и администраторской части приложения.

1.2 Разработка и анализ модели бизнес-процесса «КАК ЕСТЬ»

Для описания модели информационной системы «как есть», используем нотацию UML. Диаграмма вариантов использования описывает зависимости и взаимоотношения между группами вариантов использования и действующих лиц, участвующими в процессе.

Вариант использования описывается, с точки зрения действующего лица, группу действий в системе, которые приводят к конкретному результату. Действующее лицо является внешним источником, взаимодействующий с системой через вариант использования.

Действующие лица могут быть как реальными людьми, так и другими компьютерными системами или внешними событиями для тестирования системы [3].

Диаграмма вариантов использования «как есть» представлена на рисунке 1.

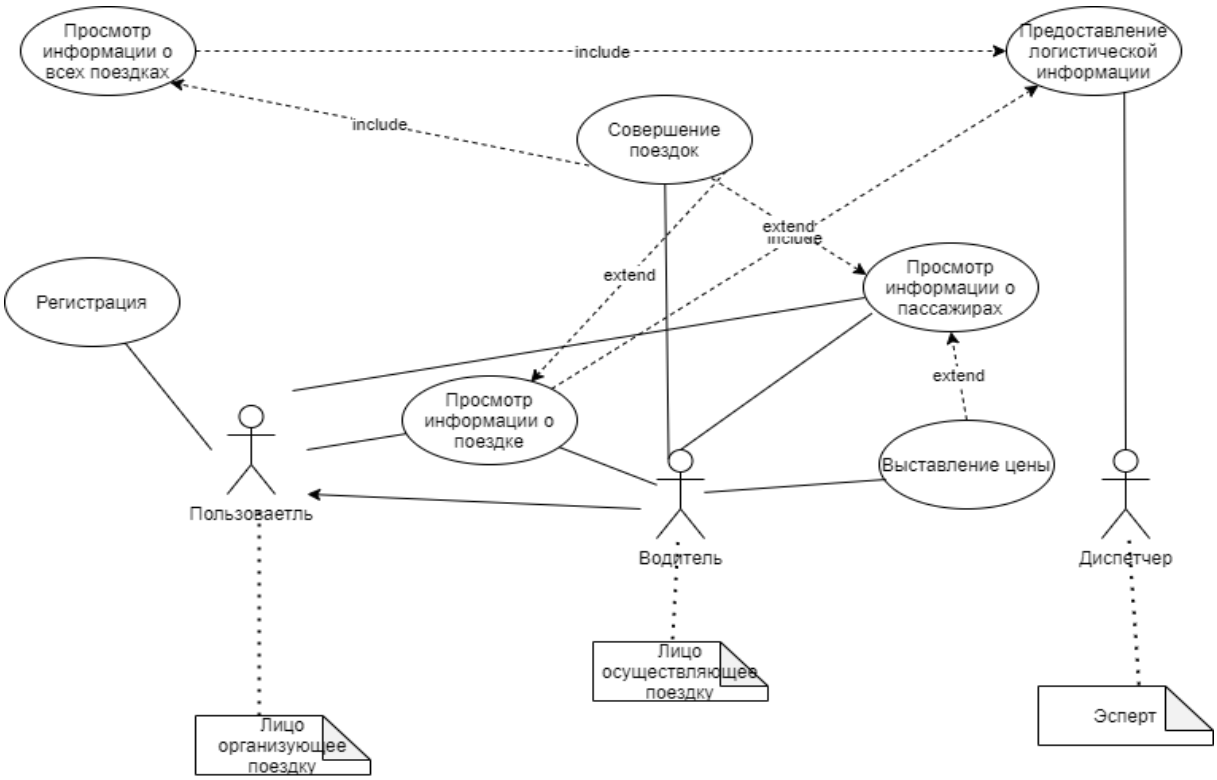

Рисунок 1 – Диаграмма вариантов использования «как есть»

«Open Street Map внешний источник» передает картографическую информацию в экспертную систему сервиса посредством API открытого источника и реализации отдельного сервера с применением Docker контейнера для возможности дальнейшего развертывания на любом сервере, при возникновении необходимости увеличения памяти, потоков и мощностей в связи с увеличением числа пользователей [1].

Экспертная система «Диспетчер», специалист отдела, предоставляющего услугу взаимного поиска водителей и пассажиров, взаимодействуют с информационной модулем системы сервиса через реализованное web–приложение. Специалист «Диспетчер», осуществляющий взаимодействие пассажира и водителя.

Пассажир в информационном модуле сервиса – это пользователь, удачно прошедший регистрацию и аутентификацию.

Пассажир взаимодействует с информационной системой через

вариант использования: «Поиск водителя» включающий вариант использования «Выдача вариантов водителей».

«Договоренность о совместной поездке» включающий вариант использования «Выдача уведомления о совместной поездке», возможность включение варианта использования «Просмотр отзывов о водителе».

Цель и сценарии включенного варианта использования должны иметь смысл независимо друг от друга, для дальнейшего использования.

Смысл включения варианта использования «Выдача информации о водителях» в «Просмотр информации о поездках», что в интерфейсной части при просмотре информации о водителях невозможно без выдачи информации о поездках.

Очевидно, что и «Совершение поездки» невозможно без «Выдача информации о водителе» т.к. смысл действия совершения райдшеринговой операции невозможно без конкретной логистической информации, но водителю не обязательно знать полную информацию о пассажире для совершения поездки.

Благодаря гибкому подходу в анализе модели «как есть» и анализ предметной области в целом были выявлены недостатки текущей работы в транспортно-логистической системе (Таблица 1).

К обработке информации в экспертной системе предъявляются следующие стандартные требования, предъявляемые в предметной области:

— достоверность информации для реализации совместных поездок;

— время задержки ответа серверной части сервиса;

— соответствие внутрикорпоративным регламентам малого инновационного предприятия;

— обеспечение полноты информации;

— реализованные затраты при обработке информации не должны превышать получаемый экономический эффект – проект должен быть рентабельным;

— способность изменятся в соответствии с изменяющимся информационным полем, где работает система.

В основном, в агрегаторах такси нет возможности совместных поездок в черте города, отсутствует функциональная возможность. В последнее время, в связи с ростом доли рынка онлайн сервисов такси, растущие запросы к надёжности, удобству, уменьшению стоимости услуги, из вышеописанного анализа, как следствие – повысились требования к скорости обмена данными, возникла потребность в модернизации существующих информационных системах и добавление нового функционала.

В модели информационной системы «как есть» составные части задачи автоматизации решены частично или не решены совсем (Таблица 1).

Таблица 1 – Задача автоматизации

Порядковый номер	Объект автоматизации	Состояние автоматизации «как есть»
1	обеспечение картографической информации	решена
2	анализ маршрутов	решена
3	процесс совершения совместных поездок	не решен
4	процесс риск - менеджмента совместных поездок	не решен

Определение путей устранения недостатков существующей технологии. Наиболее полно преимущества от использования информационных систем проявляются при получении доступа большего числа участников к таким системам благодаря собственным мощностям для разработки или покупки у компаний разработчика.

При этом, широкое распространение информационно–экспертных систем позволяет участникам движения решить большой комплекс задач:

— получить неограниченное количество попутчиков;

— расширить возможность перемещения в городской среде пассажирам;

— обеспечить людей с ограниченными возможностями дешевле перемещаться по городу;

— расширить возможности по функциям онлайн-сервисов такси для совершения совместных поездок и использование дополнительных инструментов;

— снизить издержки при перемещении на личном транспорте;

— свести к нулю потери по обслуживанию ТС благодаря совместному потреблению;

— реализовать возможность взаимной помощи на дорогах.

Наблюдаемый активный рост числа райдшеринговых систем на мировом рынке онлайн агрегаторов обусловливает начало активной конкурентной борьбы в сфере создания информационных и экспертных систем.

Условиями, влияющими на выбор профессиональными участниками той или иной системы, служат не только технические детали реализации системы, но и функциональные возможности, прозрачность алгоритмов работы, качество и надежность информационной системы в целом, данных систем нет на территории России [20].

1.3 Требования к функциональности информационной системы

Требования к программному обеспечению – совокупность утверждений относительно атрибутов, свойств или качества программной системы, подлежащей реализации [12].

Основные цели автоматизации (нефункциональные требования):

— надежность (связь; колокация; тестирование; прозрачность; ведение логов);

— устойчивость (кэширование; резервирование, бэкап);

— масштабируемость (параллельность выполнения процессов; дополнительные ресурсы доступа к логистической информации благодаря реализации картографии при помощи Open Street Map);

— гибкость (модульная система);

— возможность рефакторинга;

— доступность (сервис разработан с возможностью использования на любом устройстве и браузерах);

— скорость (стремящая к минимуму обработка карты и взаимного поиска;

Автоматизация технологических процессов информационно-экспертной системы призвана обеспечить следующие возможности (функциональные требования) [12]:

— контроль рисков (рыночный риск; риск концентрации на отдельной функции; риск ликвидности; операционный риск; юридический риск);

— райдшеринг маршрутов следования;

— умный подбор подходящего водителя;

— работа со API различных систем;

— аналитический модуль (возможность проведения анализа) при помощи Google Analytics и Яндекс.Метрики;

— анализ пользовательского опыта;

— процесс совершения поездки;

— процесс риск – менеджмента.

В таблице 2 иллюстрированы пути перехода к желаемому состоянию системы по основным направлениям.

Таблица 2 – Таблица перехода

Объект автоматизации	Недостаток	Решение
Обмен информацией	разрозненность возможностей оплаты поездки	использовать системы схожие с «Яндекс.Кассой» для удобства оплаты поездки
	ограничения в использовании кредитных карт для оплаты	создание системы оплаты схожей с системой агрегатора «Ситимобил»
Анализ личных данных пользователей	запрещено использовать возраст, пол, отзывы о пользователях и прочие личные данные для передачи третьим лицам для анализа	масштабировать принятие стратегических решений в режиме реального времени
	отсутствие нативного инструментария анализа	создание механизма сбора пользовательского опыта и обработки её в Экспертной системе
Процесс риск менеджмента	низкая скорость принятия функциональных решений	создать систему учета рисков (СУР)
Процесс совершения поездок со стороны водителя	низкая диверсификация направлений поездок	обеспечить возможность создания достаточного числа картографической информации
Процесс совершения поездки со стороны пассажира	юридическая безопасность	создание документов с прописанными условиями использования и политики конфиденциальности

Предлагаемая таблица не только описывает рамки комплекса задач, но является наглядным показателем формулировки требований к настоящему проекту, показателем будущего видения решений, которые описывают специфичные требования к информационной системе.

1.4 Изучение возможности использования уже существующих райдшеринговых систем

На данный момент не выявлено сервисов подобной направленности на территории РФ и СНГ. Существуют системы по поиску попутчиков для поездок между городами, к примеру, BlaBlaCar, Довезу.ру, Попутчик.ру.

Данные ресурсы не предоставляют функциональные возможности по построению, поиску и бронированию поездок внутри города, в отличии от данной экспертной системы.

В качестве примеров для выявления плюсов и минусов можно рассмотреть системы BlaBlaCar, Довезу.ру и разрабатываемую систему для сравнения (таблица 3).

Таблица 3 – Сравнение систем

Критерии	Системы		
	BlaBlaCar	довезу.ру	разрабатываемая система
Возможность создания совместных поездок внутри города	нет	нет	есть
Возможность поиска поездок внутри города	нет	нет	есть
Актуальность информации	есть	есть	есть
Целевая аудитория	население стран, в которых запущен сервис	население стран, в которых запущен сервис	население РФ

На основе проведенного анализа в таблице 1 можно сделать вывод, что разрабатываемая система обладает всеми необходимыми свойствами и не имеет аналогов на территории РФ.

1.4.1 Экономическая сущность комплекса задач

Объектом рассмотрения является процесс автоматизации оказания райдшеринговых услуг. Выберем входящую в данный объект задачу, создания встраиваемых модулей для автоматизации взаимного поиска пассажиров и водителей, другими словами, создание среды для развертывания экспертной системы.

Данная задача относится к классу задач «Автоматизация работы диспетчеров такси» и необходима для поддержания работы райдшерингового онлайн сервиса и тенденции выбранного рынка «Автонет». Результаты решения данной задачи являются основой для требований, согласованных с фондом для реализации.

Поэтому задача «Создание встраиваемых модулей» является важной и неотъемлемой частью процесса автоматизации поиска и создания поездок для совместного перемещения в черте города. Информацию для решения задачи получают на этапах проектирования в виде технического задания.

Результаты решения задачи могут служить идеальной площадкой для тестирования и реализации бизнес-стратегий ресурса [10].

Главная задача сервиса отвечать требованиям, согласованные с заказчиком (фондом), иначе соблюдать договорные обязательства малого инновационного предприятия перед клиентом.

Поэтому, экономические показатели, направленные на повышение рентабельности бизнеса, свойственные типичным коммерческим организациям, в нашем случае, имеют первостепенное значение.

На показатели рентабельности экономической деятельности в первую очередь влияет оборот компании, как и количество самих клиентов, дает весомую часть получаемой прибыли. Как и любые другие показатели, доход этот можно увеличить, путем географического и

функционального расширения приложения [3].

1.5 Выбор методологии проектирования и архитектуры ЭС

В настоящее время, в области разработки и реализации экспертных систем сложилось следующее положение: с одной стороны, квалификация коллективов разработчиков здесь, как правило, достаточно высока, с другой стороны, одна из сложнейших проблем, препятствующих широкому внедрению ЭС, является недостаточное знание системными аналитиками и программистами предметных областей, в рамках которых готовятся проекты [5].

Создание и внедрение систем общения с базами данных и особенно информационно-экспертных систем, и их широкое распространение выдвинуло проблему совершенствования методологии создания информационных систем на передний план [5].

На текущий момент выделяют две основных методологии проектирования информационных систем: структурные методологии и объектно-ориентированные методологии проектирования.

Многие структурные методологии моделирования бизнес-процессов состоят из методологий: SADT – метод структурного анализа и проектирования, семейство стандартов IDEF и алгоритмические языки [11]. Основные типы методологий моделирования и анализа бизнес–процессов:

— SADT методология структурного анализа и проектирования, интегрирующая процесс моделирования, управления конфигурацией проекта, использования дополнительных языковых средств и руководства проектом;

— IDEF0 модели процессное моделирование в общей нотации IDEF. Изучаемая система или процесс представляются в виде

взаимосвязанных функций;

— IDEF1x модели так же являются информационным моделированием, основанным на концепции «Сущность связь». Как правило, используется для моделирования реляционных структур баз данных;

— IDEF3 модели (англ. ProcessDescriptionCapture), описание и документирование процессов информационной системы, используется для моделирования альтернатив и причинно–следственных связей;

— Нотация DFD является удобным средством для формирования контекстной диаграммы, то есть диаграммы, показывающей разрабатываемую ИС в коммуникации с внешней средой [15].

В основе вышеперечисленных типов методологий лежит метод декомпозиции. Выбранная методология для использования в настоящем проекте должна отвечать требованиям [15]:

— Простота моделирования предметной области;

— Моделирования сложных бизнес-процессов;

— Быть наглядной моделью;

— Выявлять возможность повторного использования и рефакторинга кода.

Проведем сравнительный анализ и последующий выбор из существующих на сегодняшний день объектно–ориентированным методологий:

— SA/SD содержит несколько вариантов обозначений для формальной спецификации программных систем. В методологии SA/SD организован этап структурного анализа. После структурного анализа начинается этап структурного конструирования, в процессе которого разрабатываются и уточняются более тонкие детали проектируемой системы.

— JSD (англ. JacksonStructuredDevelopment) объединяет этап анализа

требований и этап разработки системы в один общий этап разработки спецификаций проектируемой системы. Методология JSD может быть названа объектно-ориентированной с большой натяжкой: в ней почти не рассматривается структура объектов, мало внимания уделяется их атрибутам.

— OSA (англ. Object-OrientedSystemAnalysis) обеспечивает объектно–ориентированный анализ программных систем и не содержит возможностей, связанных с поддержкой этапа разработки.

— UML является стандартной нотацией визуального анализа и моделирования программных систем.

При выборе методологии проектирования используем критерии, указанные в таблице 4.

Таблица 4 – Критерии выбора методологии проектирования

Критерий	Весовой коэффициент
Возможность обнаружения ошибок на начал-х этапах ЖЦ	3
Генерация кода	3
Наглядность	3
Наличие программного продукта (ПП)	2
Опыт работы	2
Генерация кода	3
Наглядность	3
Наличие программного продукта (ПП)	2
Опыт работы	2

Оценку методологий по критериям проведем согласно бальной системе: 1 – нет реализации, 2 – частично реализован критерий, 3 – реализован (Таблица 5).

Таблица 5 — Выбор методологии проектирования

Критерий	k (вес критерия)	SA/SD	JSD	OSA	UML
Возможность обнаружения ошибок на начальных этапах ЖЦ	3	2	1	1	3
Генерация кода	3	2	2	2	3
Наглядность	1	2	3	2	3
Наличие программного продукта (ПП)	2	1	2	1	3
Опыт работы	3	2	2	3	3
Итого (с учетом k – значимости критерия)		20	22	20	37

Однозначно можно сказать, что по вышеперечисленным критериям, отвечающим современным требованиям, автор проекта отдает предпочтение объектно–ориентированным методологиям. Для анализа проектирования ИС будет применяться объектно-ориентированная нотация UML.

В данном работе применяется объектно–ориентированный подход к проектированию ИС, так как сервис реализован языке программирования JavaScript. Разработка информационной системы ресурсоемкая, т.е. требует несколько сотен человек-часов, в процессе реализации были использованы разные методологии в зависимости от стадии проекта.

Распространенные гибкие технологии: eXtremeProgramming (XP). Технология XP, (экстремальное программирование) является наиболее гибкой технологией. Основой проектной документации считается тщательно прокомментированный код. Большое внимание в технологии уделяется тестированию.

FeatureDrivenDevelopment (FDD). Используемое в FDD понятие функции или свойства системы достаточно близко к понятию прецедента

использования, используемому в RUP. Существенное отличие: "каждая функция должна допускать реализацию не более, чем за две недели".

FDD включает пять процессов: составление общей модели, список функций системы, план работы по функциям, проектирование функции, разработка функции [13].

RationalUnifiedProcess (RUP). Основа RUP это:

— Ранняя идентификация и непрерывное устранение основных рисков;

— Концентрация на выполнении требований заказчиков к исполняемой программе (модель прецедентов);

— Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки;

— Компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта;

— Постоянное обеспечение качества на всех этапах разработки проекта;

— Работа над проектом в команде, ключевая роль в которой принадлежит архитекторам.

Принципы: итерационный подход; планирование и управление проектом на основе функциональных требований к системе (варианты использования); построение системы на базе архитектуры программного обеспечения [10].

При выборе технологии проектирования используем критерии с коэффициентами (Таблица 6).

Таблица 6 — Критерии выбора с коэффициентами

Критерий	Весовой коэффициент
Возможность выбора степени формализации разработки	2

Продолжение таблицы 6

Критерий	Весовой коэффициент
Итерационный подход	2
Наглядность	2
Производительность	3
Простота использования	2
Наличие программного продукта (ПП)	3

Оценку технологии по критериям проведем согласно бальной системе: 1 – нет реализации, 2 – частично реализован критерий, 3 – реализован (Таблица 7).

Таблица 7 – Оценка технологий

Критерий	k (вес критерия)	XP	FDD	RUP
Возможность выбора степени формализации разработки	2	1	1	2
Итерационный подход;	2	3	3	2
Наглядность	2	3	2	2
Производительность	3	3	3	3
Простота использования	2	2	2	2
Наличие ПП	3	3	1	2
Итого (с учетом k – значимости критерия)		34	27	31

Вывод: при проектировании ИС будет применяться гибкая технология проектирования XP.

Будем понимать под архитектурой ПО внутреннюю структуру продукта (компоненты и их связи), MVP пользовательского интерфейса проекта, а также знания и решения, являющиеся инструментом разработки

и управления проектом.

Уровень развития информационных технологий на сегодняшний день при построении информационных систем позволяет выбрать архитектуру из достаточно широкого спектра. При создании экспертной системы подразделения рассматривались различные архитектуры построения информационных систем: файл-серверная архитектура и архитектура «клиент–сервер».

Файл-серверная архитектура нацелена на использование небольших объемов передаваемой информации и работу небольшого числа пользователей. Системы, построенные на основе указанной архитектуры, применяются в небольших компаниях для решения небольших изолированных задач [12].

Файл-серверная архитектура не подходит для реализации ЭС, призванных работать с большим объемом передаваемой информации при высоких нагрузках на сервер, что не позволяет разрабатывать проектируемую ЭС в соответствии с данной архитектурой.

В архитектуре «клиент–сервер» логика приложения выполняется либо на клиентской стороне (“толстый” клиент), либо на серверной стороне, обычно, в виде хранимых процедур базы данных (“тонкий” клиент). Основной, помимо сложности масштабирования, проблемой архитектуры клиент-сервер, является необходимость установления соединения между базой данных и каждым пользователем [12].

Выбор был сделан в сторону «клиент–сервер», тип двухзвенная, т.е. интерфейсная часть и база данных.

1.6 Разработка модели бизнес-процесса «КАК БУДЕТ»

Ранее были выявлены недостатки текущей модели «как есть», т.е. работы в существующих условиях. Для их устранения необходимо составить модель «как должно быть», в которой учитывались и

устранялись все недостатки текущих процессов.

В рамках технологии XP, необходимо помнить о повторяемости итерационного подхода и содержании самих периодов. Стадии XP: начальная стадия, уточнение, конструирование, внедрение [15]. На примере создание модели «как должно быть» проходим стадию начало:

— При анализе предметной области, были сформулированы границы ИС, было проведено бизнес-моделирование и создана модель «как есть»;

— Создано экономическое обоснование;

— В таблице перехода, а при повторении итерации в стадии уточнение, разбили на блоки. Были определены основные требования, ограничения и ключевая функциональность сервиса;

— Создаём базовую версию модели прецедентов.

При соблюдении требований технологии XP, повторим итерации в стадии уточнения, благодаря чему у нас появится несколько моделей «как должно быть», с конечной моделью.

Для того чтобы более точно понять, как должна работать информационная система, все чаще используется описание функциональности системы через варианты использования. Варианты использования предназначены в первую очередь для определения функциональных требований к системе и управляют всем процессом разработки.

Все основные виды деятельности, такие как анализ, проектирование, тестирование выполняются на основе вариантов использования. Во время анализа и проектирования варианты использования позволяют понять, как результаты, которые хочет получить пользователь влияют на архитектуру системы и как должны себя вести компоненты системы, для того чтобы реализовать нужную функциональность [15].

В процессе тестирования, описанные ранее варианты использования позволяют проще оценить точность реализации требований, а также

провести пошаговую проверку этих требований.

Стратегия использования прецедентов при определении требований определяет необходимость brainstorm для выявления основных функций ресурса [13]. Такой подход позволяет выявить функции необходимые большему числу пользователей и исключит лишние.

Диаграмма вариантов использования состоит из актеров, для которых система производит действие и собственно действия Use Case, которое описывает то, что актер хочет получить от системы. Актер обозначается значком человечка, а Use Case – овалом. Дополнительно в диаграммы могут быть добавлены комментарии.

Вид диаграммы вариантов использования «как должно быть» представлен на рисунке 2.

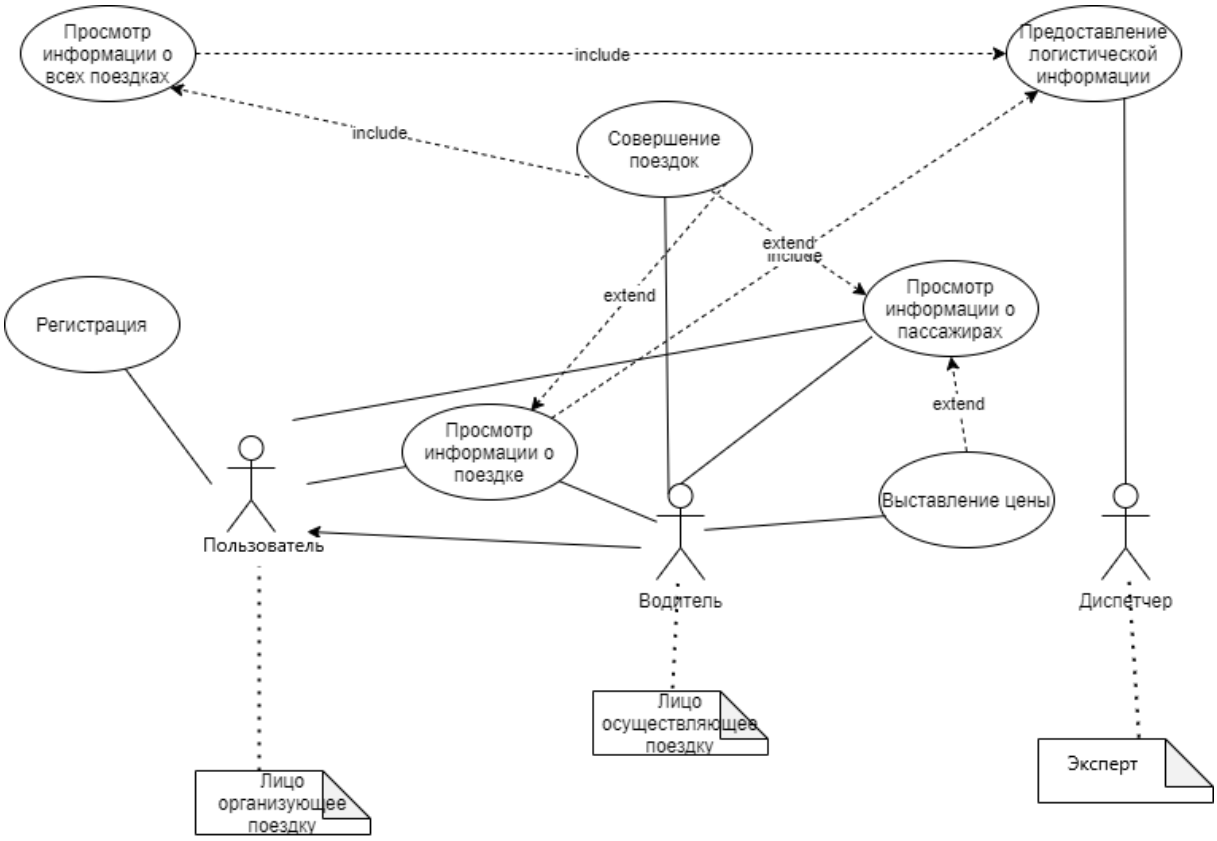

Рисунок 2 – Диаграмма вариантов использования «как должно быть», 1 вариант

Для реализации требуемого функционала добавлены необходимые прецеденты.

Администратор – сотрудник сервиса, может благодаря панели администратора: просматривать всю информацию на ресурсе, а также скачивать и предоставлять информацию.

Модератор – специалист, осуществляющий отслеживание состояние модулей сервиса и в случае возникновения экстренных ситуаций сообщать руководству.

Унифицированный процесс – это процесс, управляемый прецедентами, которые отражают сценарии взаимодействия специалистов. Фактически, это взгляд пользователей на программную систему снаружи. Таким образом, одним из важнейших этапов разработки, согласно XP, будет этап определения требований, который заключается в сборе всех возможных пожеланий к работе системы, которые только могут прийти в голову потенциальным пользователям и аналитикам [13].

Простота диаграммы прецедентов позволяет легко общаться с заказчиками в процессе определения требований, выявлять ограничения, налагаемые на систему и на выполнение отдельных требований, например, время реакции системы, которые в настоящей работе попали в раздел нефункциональных требований.

Также диаграмма прецедентов может использоваться для создания сценариев тестирования, поскольку все взаимодействие пользователей и системы уже определены. Сценарий представляет собой последовательность шагов, описывающих взаимодействие между пользователем и системой.

Большой комплекс задач не позволяет полностью разобрать предложенную диаграмму (Рисунок 2), в связи с этим разберем диаграмму вариантов использования решающую часть функций информационно-экспертной системы, представленную на рисунке 3.

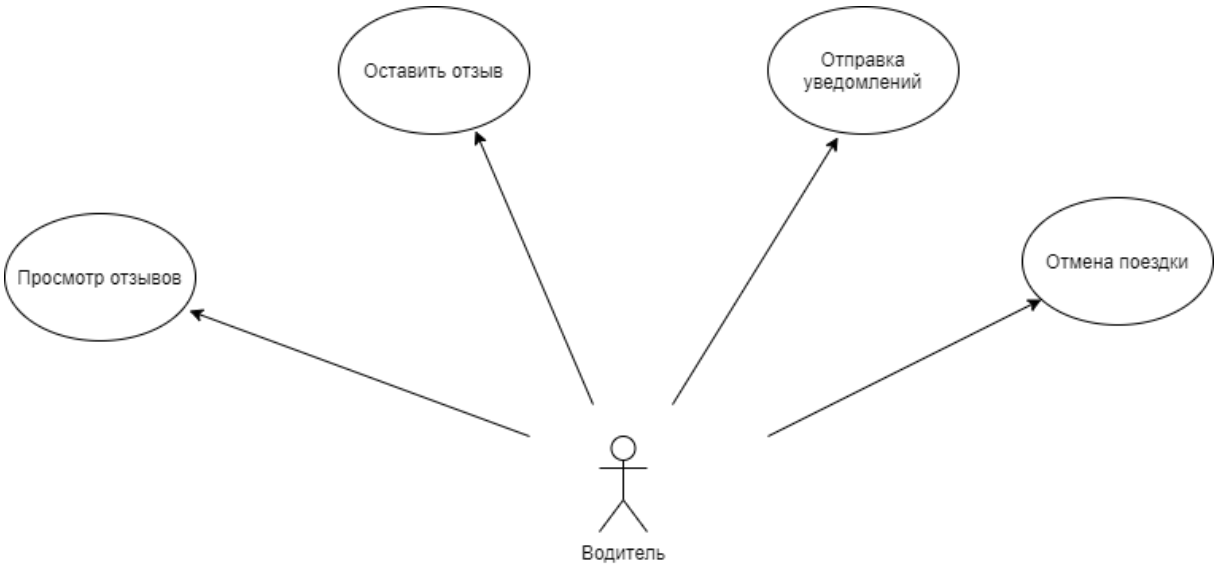

Рисунок 3 – Диаграмма вариантов использования «как должно быть», вариант 2

Выделение классов и объектов – одна из самых сложных задач объектно–ориентированного проектирования, которая осуществляется в процессе декомпозиции ключевых абстракций программной системы.

Декомпозиция занимает центральное место в объектно–ориентированном анализе и проектировании программного обеспечения. Под объектно-ориентированной декомпозицией понимается процесс разбиения системы на части, соответствующие объектам предметной области [5].

Практическое применение объектно–ориентированного проектирования приводит к объектно–ориентированной декомпозиции, при которой мы рассматриваем мир, как совокупность объектов, согласованно действующих для обеспечения требуемого поведения [5].

Прецеденты были разбиты с использованием добавления дополнительных вариантов использования, более подробно на рисунке 4.

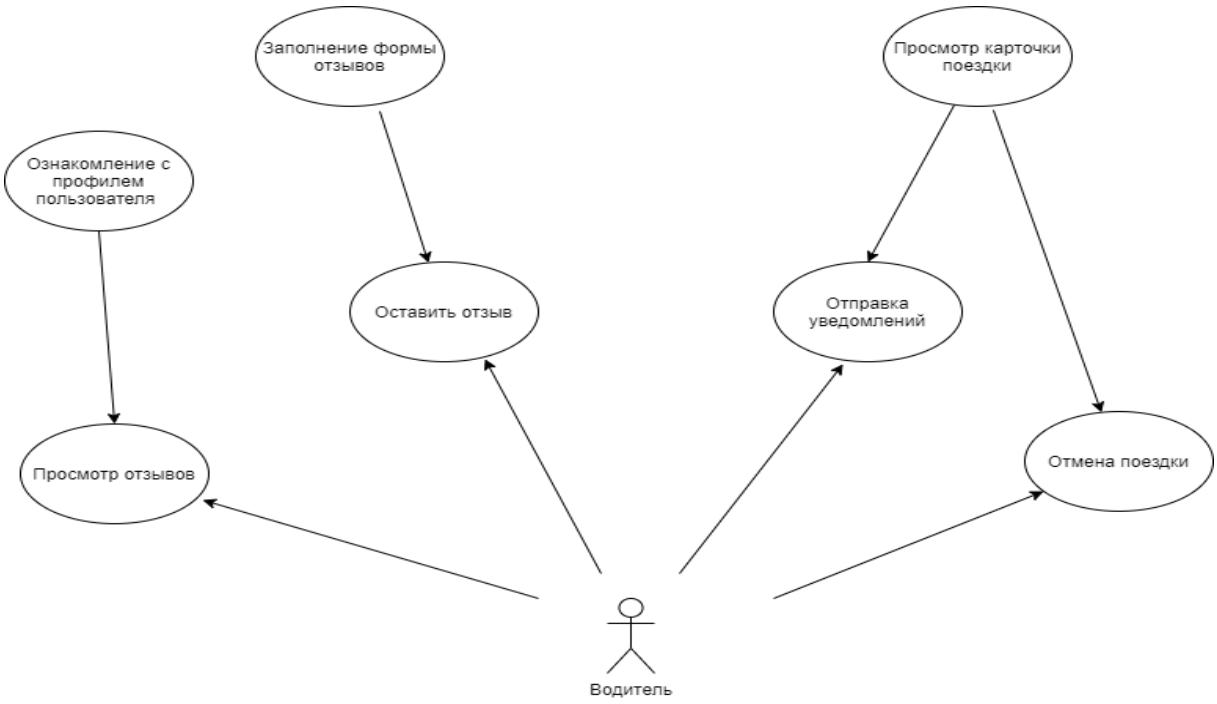

Рисунок 4 – Диаграмма вариантов использования «как должно быть», вариант 3

При проектировании и разработке комплекса задач учитывалась необходимость присутствия свойств открытой системы. Свойства открытой системы, такие как масштабируемость, модульность структуры, использование локальных сетей.

Выполним постановку целей автоматизации путем выделения следующих решений-блоков:

— поиск поездок;

— создание поездок;

— построение маршрутов;

— модуль подключения сервера к картографическому сервису Open Street Map через API;

— модуль динамического экспорта информации для учетных систем;

— подключения базы данных автомобилей.

В результате планирования информационной системы было принято решение реализовать проект на платформе Node.js по типу клиент-

серверной архитектуры и модульному принципу.

Центральное звено, под названием «сервер», к которому подключаются внешние модули. Модули представляют собой функциональные блоки с различным назначением. Одни модули осуществляют работу с картами, которые отображаются благодаря клиентской часть приложения. Другие модули эти карты генерируют. Третьи модули управляют записью информации. Четвертые модули экспортируют информацию [1].

Блок поиска поездки. Позволяет пассажиру найти поездку по необходимым параметрам даты, времени и маршруту. Модуль создания поездки позволяет водителю создать поездку, введя информацию о времени, маршруте, промежуточных точках, а также добавить комментарии к поездке.

Блок построение маршрутов. Данный блок реализован благодаря функциям дробления пути и созданию промежуточных точек маршрута. Картографический клиент. Этот модуль осуществляет подключение к отдельному серверу, в котором находятся фреймы карт, благодаря которым и осуществляется основной пользовательский сценарий.

Модуль динамического экспорта информации для учетных систем позволяет хранить все данные, генерируемые на сервисе. Благодаря логированию и дальнейшем отображении в панели администратора вся информация хранится в формате JSON. Без этого недопустима работа приложения с юридической точки зрения.

Предусмотрена ручное управление процессами в системе в случаях отказа, взлома или восстановление целостности сервиса. Данный модуль необходим для того, чтобы администратор в случае возникновения непредвиденных обстоятельств смог наладить работу сервиса самостоятельно.

Модуль, связанный с добавлением автомобилей при регистрации водителей. Данная проблема была решена благодаря подключению API

стороннего сервиса с базами данных о всех существующих автомобилях с возможностью выбора цвета.

1.7 Разработка прототипа интерфейса пользователя

При разработке экспертной системы был создан интерфейс с помощью собранных метрический данных с потенциальных пользователей. Учитывая, что на интерфейсных формах не должно быть ничего лишнего.

Именно с помощью интерфейса происходит коммуникация конечных пользователей с информационной системой. Для выявления прецедентов бизнес–логики разрабатываемой системы и составления диаграммы вариантов использования были созданы прототипы пользовательского интерфейса.

Это помогло выявить скрытые прецеденты взаимодействия пользователя и системы, определить структуру системы, понять какие сущности нужно отобразить в модель предметной области.

Прототипы интерфейсов пользователя требуются для распределения компонентов управления по экранным формам оптимальным образом. Они позволяют проектировать логику приложения отталкиваясь от того, как будет устроен пользовательский интерфейс [6].

При реализации интерфейсной части было решено использовать стратегию mobile first, предполагающая создание мобильной версии в первую очередь. На основе полученных данных была построена диаграмма деятельности UML, которая изображена на рисунке 5.

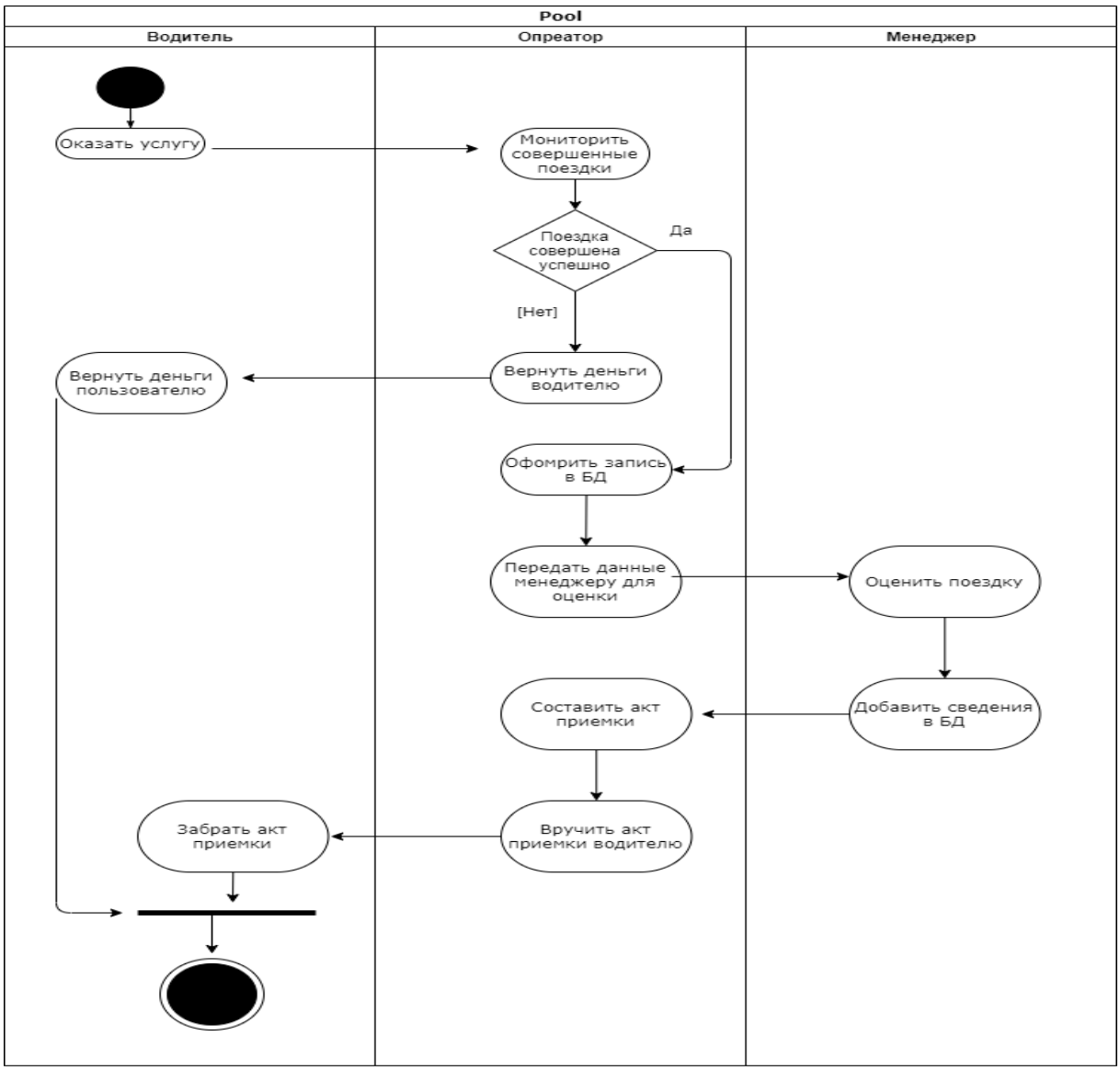

Рисунок 5 – Бизнес-процесс в системе BPMN

Для начала работы пользователь должен зарегистрироваться и авторизоваться в сервисе. Исходя из этого в меню появится дополнительное графическое изображение, свидетельствующее об успешном входе в систему.

При нажатии кнопки «Зарегистрироваться» начинается процесс регистрации. Если пользователь входит в ИС, то он переходит на основную экранную форму. Здесь располагается меню, документы, вопросы и ответы по проекту, а также информация о самом сервисе. Прототип страницы авторизации пользователя изображен на рисунке 6.

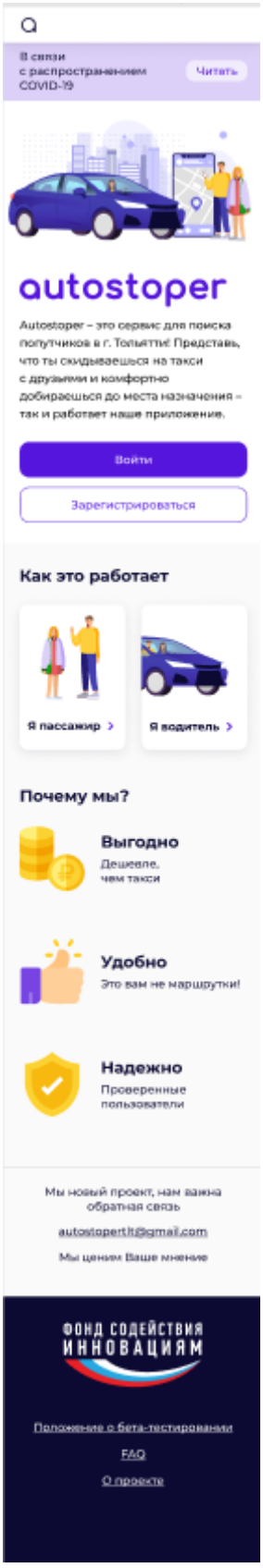

Рисунок 6 – Экранная форма «Авторизация»

При нажатии «О проекте» появится landing page в котором проект описан в стиле Storytelling. (рисунок 7).

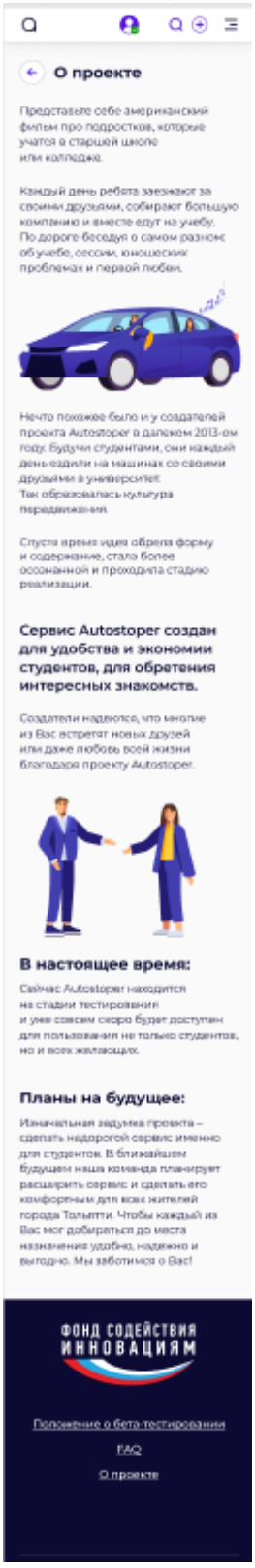

Рисунок 7 – Выпадающий список таблиц

На рисунке 8 представлена иллюстрация фильтра таблицы.

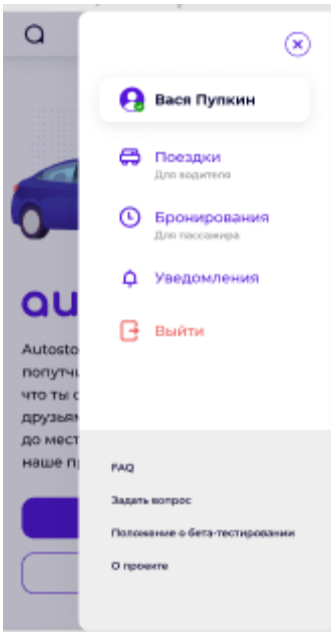

Рисунок 8 – Фильтр таблицы

А также представлены перечень документов, вопросы и ответы, информация для обратной связи, предупреждение о распространении covid–19, краткая информация о самом сервисе.

1.8 Постановка задачи на разработку экспертной системы

В процессе проектирования были сформированы следующие данные, необходимые для реализации ЭС:

— ЭС создается для автоматизации поиска и создании поездки, формирование маршрутов, расчета ориентировочного времени в пути;

— ЭС должна быть реализована с использованием двухуровневой архитектуры, отвечающей всем поставленным задачам, описанными в ТЗ проекта;

— Серверная часть ЭС должна быть реализована с использованием технологий, позволяющих развернуть проект на различных платформах;

— Клиентская часть ЭС должна быть реализована с использованием технологий, позволяющих выполнять приложение на всех устройствах;

— Клиентская и серверная части должны взаимодействовать с использованием JSON файлов;

— Сервис должен быть удобный и интуитивно понятным пользователям;

— Приложение должно поддерживать соответствие баз данных.

В результате проведенных в первой главе настоящего проекта исследований и работ:

— Выполнено описание предметной области;

— Наглядно показана актуальность, новизна, практическая ценность предлагаемой системы;

— Проведен сравнительный анализ технологий, методологий и архитектуры проектирования моделей информационных систем в области райдшеринговой логистики;

— Выполнен анализ существующей технологий, определение путей устранения недостатков, характеристика существующих процессов;

— Выполнен переход от состояния «как есть» к состоянию «как должно быть», представлен комплекс задач и возможная таблица перехода;

— Выполнена постановка цели и подзадач.

2 Реализация программного обеспечения для решения задачи определения степени вовлеченности студентов в учебный процесс

2.1 Выбор платформы разработки

Клиентская часть web – приложения реализована в соответствии со всеми современными подходами создания и поддержки web – ресурсов, а именно благодаря использованию компонентно-ориентированного программирования.

Данная технология является оптимальной для решения задач, связанных с масштабированием и эксплуатацией информационного ресурса, для написания клиентской части известны следующие технологии: ASP.NET, Django, Laravel, Drupal, jQuery, Express, Spring, React.js, Angular.js, Vue.js, Flask, Ruby on Rails.

На рисунке 9 изображены результаты опроса StackOverflow на котором показано, какие web–фреймворки наиболее часто используются при разработке web–приложений:

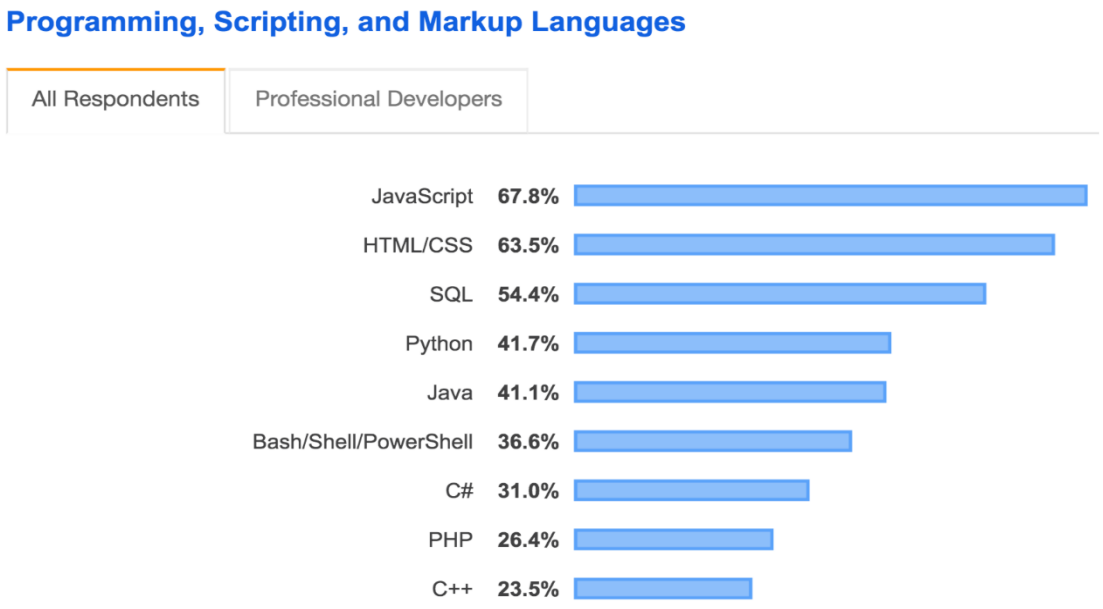

Рисунок 9 – Результаты опроса StackOverflow

Из данного рейтинга можно выделить 3 технологии, а именно jQuery, React, Angular. При разработке web-приложения была использована технология React - JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов, которая активно используется в современных IT-проектах, с потенциалом дальнейшего развития и интеграции [7].

Для написания серверной логики web–приложения, наиболее популярные следующие технологии:

— Express;

— Spring;

— Java EE;

— Ruby on rails;

— Django.

Среди всех технологий на основании опроса StackOverflow можно выделить явных фаворитов: Express, Spring, Django, все три фреймворка имеют достаточную документацию, широко распространены, а также реализуются на высокоуровневых языках программирования, все это является необходимыми условиями для полноценной реализации, поддержки и эксплуатации web-приложения [7]. Для реализации ресурса была выбрана технология Express.

Также для создания полноценного web-приложения необходима база данных, наиболее популярными на данный момент являются:

— PostgreSQL;

— MongoDB;

— MySQL.

Для использования в проекте была выбрана СУБД PostgreSQL, т.к. она обладает следующим рядом достоинств:

Сильными сторонами PostgreSQL считаются [7]:

— Высокопроизводительные и надёжные механизмы транзакций и репликации;

— Расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl; дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки модулей расширения на языке C;

— наследование;

— возможность индексирования геометрических объектов и наличие базирующегося на ней расширения PostGIS;

— встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;

— расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных).

2.2 Реализация серверной части web-приложения

Для написания бэкенд части web-приложения были использованы следующие библиотеки:

— Apidoc – библиотека для составления API документации на основе комментариев;

— Body–parser – библиотека для работы телом запроса на сервер;

— Express–validator – библиотека для предобработки запроса, позволяет в случае ошибки остановить запрос и не обрабатывать невалидные данные;

— Pg – библиотека для отправки SQL запросов на сервер;

— Express–session – библиотека для создания сессий;

— Nodemailer – библиотека для создания email рассылок.

Сам проект с серверной логикой web–приложения имеет следующую структуру:

— Build;

— Config;

— Controllers;

— Libs;

— Public;

— Repositories;

— Routes;

— Services;

— Views;

— Index.js.

Файл index.js является основным файлом на сервере, он содержит в себе разворачивание сервера и его настройку, в котором происходит подключение большого количества библиотек и промежуточных обработчиков.

Папка build должна содержать в себе билд приложения React, которое будет использоваться как клиентская часть web-приложения.

Папка config содержит в себе конфигурационные данные, внутри нее есть файл config.js с основными константами сервера, файл models.sql содержит SQL скрипт для создания БД, прочие файлы необходимы для создания стартовых моделей, например, в проекте необходимы список моделей транспортных средств, именно в тех файлах будут скрипты для записей моделей в БД.

Папка controllers содержит обработчики сервера, каждый контроллер – это набор логики одного определенного роута.

Папка libs содержит в себе часто используемые или объемные функции, которые были написаны для проекта.

Папка public содержит в себе публичные файлы, которые будут раздаваться сервером, как статичные.

Папка repositories содержит функции, которые отправляют SQL запросы в БД. Они не имеют никакой предварительной валидации и не

выполняют сложную бизнес-логику сервера, они необходимы для обращения к БД.

Папка routes хранит в себе все обработчики запросов, в данной папке описаны все роуты, на которые необходимо отвечать серверу. Для каждой модели создается отдельная папка роутов, а в ней по 3 файла, каждый из этих файлов отвечает за свой уровень доступа:

— публичный – приписка файла public;

— для пользователей – приписка verification;

— для администраторов – приписка admin.

Таким образом, если необходимо описать роуты для сущности пользователя, под названием user, то файлы будут находится в папке users, а внутри будет 3 файла:

— users_public.js;

— users_verification.js;

— user_admin.js.

Папка services содержит в себе набор функций, которые потенциально могут быть переиспользованы в проекте, в отличии от папки libs данная папка будет содержать функции, которые производят действия непосредственно с самими сущностями БД.

Папка views содержит в себе набор файлов с расширением ejs, которое будет использоваться для написания панели администратора [10].

2.3 Описание и анализ технологии разработки

Используется гибкий подход к созданию программного обеспечения. Целью такого подхода является минимизация рисков, путём сведения разработки к серии коротких циклов, называемых итерациями, которые обычно длятся одну-две недели, данный период называется спрингом [1].

Каждая итерация сама по себе выглядит как программный проект в миниатюре, и включает все необходимые задачи. В качестве основных

подходов к моделированию, реализации и тестированию используются Model Driven Architecture и разработка через тестирование TDD.

Сутью MDA является управление кодом через модель и генерацию. Это означает, что с помощью соответствующих программных средств, строится метамодель всей информационной системы, описывающая статические и динамические аспекты по правилам, устанавливаемыми конкретной реализацией продукта, позволяющего генерировать код. Далее, средства генерации анализирует метамодель и генерирует соответствующий код.

Разработка через тестирование предполагает создание модульных тестов до написания кода. Созданный тест ограничивает минимальный функционал метода или класса, и сборка проекта не произойдет до тех пор, пока удачно не выполнится весь набор тестов [1].

Таким образом гарантируется, что собранный проект будет удовлетворять условиям установленных бизнес-правил. Кроме того, прохождение всего набора тестов при добавлении новых классов или методов класса реализует регрессионное тестирование. При передаче заказчику очередной версии системы проводятся интеграционные тесты, проверяющие взаимодействие всех модулей в виде единого целого.

Гибкие методологии предполагают итеративную разработку, каждая итерация разработки требует определенной последовательности шагов, направленных на реализацию функционала разрабатываемой информационной системы. Типичная последовательность действий в итерации выглядит следующим образом [1]:

— Анализ предметной области с учетом функционала, добавленного на предыдущей итерации;

— Выявление наиболее приоритетных задач и добавление их в предстоящую итерацию;

— Оценка примерной трудоемкости каждой из запланированных задач;

— Проектирование архитектуры для каждой запланированной задачи и анализ общей архитектуры разрабатываемой системы на предмет возможного рефакторинга;

— Реализация функционала с параллельным юнит – тестированием;

— Интеграция;

— Интеграционное тестирование;

— Выпуск.

Структура модели предметной области в гибких проектах быстро меняется. Для того чтобы учесть все вносимые изменения в начале итерации проводится анализ предметной области. Эта операция входит в этап планирования.

Выявление наиболее приоритетных задач так же является частью планирования для того, чтобы выполнять в планируемой итерации только те задачи, решение которых является наиболее перспективным для развития или задач, требующих решения.

Оценка трудоемкости требуется для составления графиков производительности команды разработчиков. На этом этапе каждый участник команды называет примерное время, за которое он бы смог справиться с задачей. После чего берется либо усредненное время, либо наименьшее время.

Необходимо отметить, что время, в которое оценена та или иная задача, является примерной оценкой и не всегда задача выполняется в установленный срок. Сумма оценок выполненных задач за итерацию учитывается при планировании следующей итерации, являясь, своего рода, обратной связью по перебору или недобору задач.

Интеграция предполагает сборку всех частей, как уже реализованных ранее, так и тех, что были созданы на данной итерации, воедино. Сборка происходит при помощи автоматизированных средств сборки, преимущественно на специализированном удаленном сервере.

После сборки составляется план тестирования и выполняется

интеграционное тестирование, целью которого является выявление возможных ошибок в собранном проекте.

В случае обнаружения ошибок составляется отчет об ошибках, которые подлежат устранению. После устранения всех ошибок очередная версия программы сдается заказчику.

Создание крупных проектов требует применения определенного набора инструментов разработки. Ниже описаны средства разработки, применяемые при работе над настоящим проектом.

Для облегчения труда разработчиков предназначены интегрированные средства разработки, позволяющие автоматизировать рутинные операции, такие как форматирование кода, автоматическое создание стандартных методов класса.

Обычно, среда разработки включает в себя текстовый редактор, компилятор и/или интерпретатор, средства автоматизации сборки и отладчик.

Иногда также содержит средства для интеграции с системами управления версиями GIT и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя.

Среда разработки для создания информационной системы является VisualStudio - коммерческая IDE компании Microsoft. Первая версия VS появилась в 1997 году и быстро приобрела популярность, как IDE с широким набором интегрированных инструментов, которые позволяли быстро реорганизовывать исходные тексты программ и логично соотносится с итерациями используя выбранную технологии проектирования.

Дизайн среды ориентирован на продуктивность работы, позволяя сконцентрироваться на разработке функциональности, в то время как Visual Studio берет на себя выполнение рутинных операций.

2.4 Проектирование базы данных информационной системы МИП

Для проектирования БД была использована методология IDEF1X, так как она крайне подробно показывает концептуальную модель БД. Для создания схемы использовался онлайн сервис draw.io. Схема БД для данного web-приложения представлена на рисунке 10:

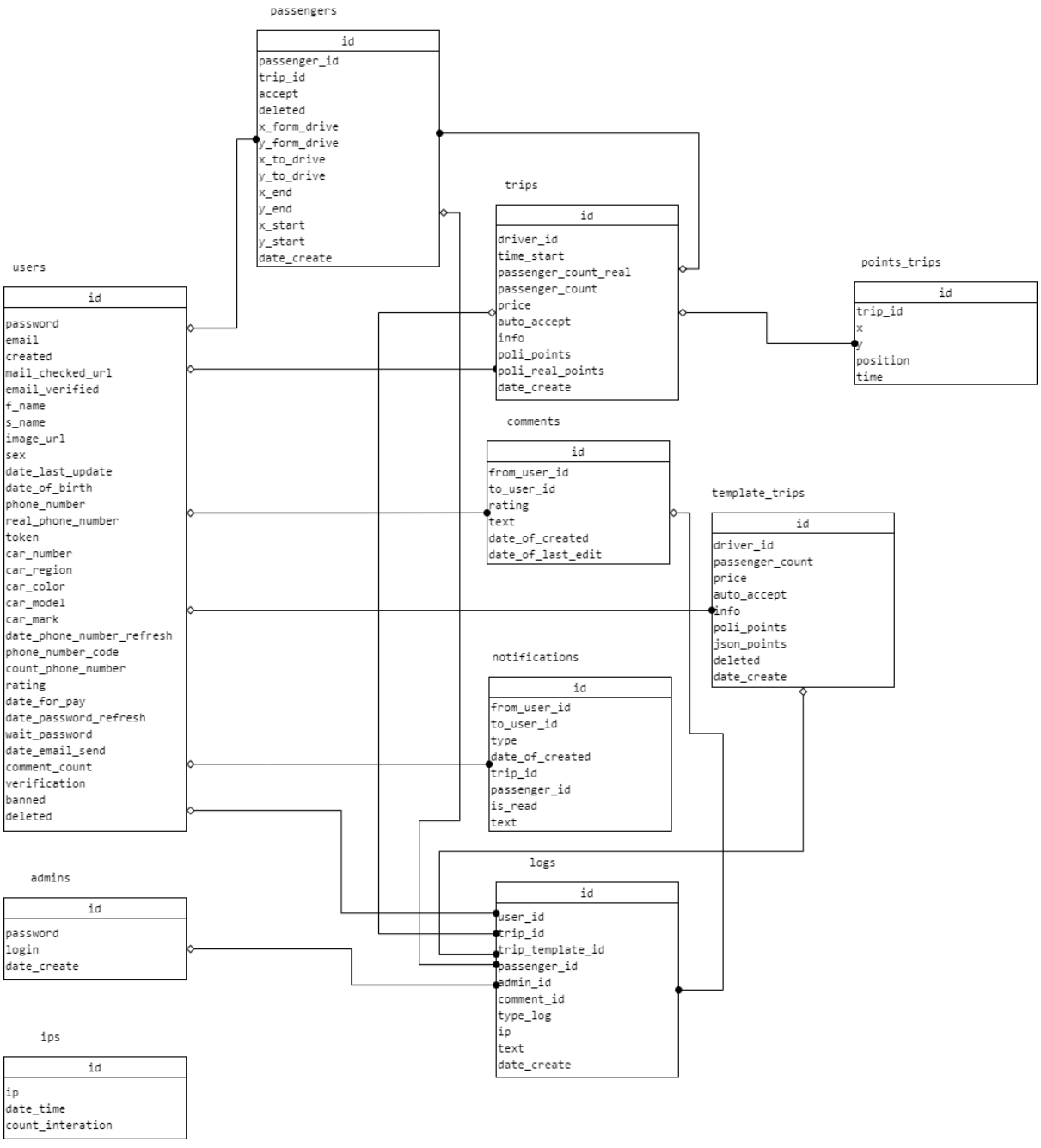

Рисунок 10 – Схема базы данных web-приложения

Данная схема БД предусматривает абсолютно все варианты дальнейшего развития приложения, каждая сущность в БД является самостоятельной, а удаление одной из них не повлечет за собой удаление остальных связи, так как они независимы. Необходимо разобрать основные моменты данной БД. Наиболее важными таблицами являются таблица users и таблица logs. Данные таблицы полностью описывают весь подход реализации web–приложения.

Таблица logs содержит в себе ссылки на элементы таблицы, к которым относится лог, таким образом, данный подход позволяет реализовать интуитивно понятную панель администратора, в которой сможет освоиться даже пользователь без навыков программирования, так как лог будет иметь свои ссылки на элементы и подробное текстовое описание.

Также, поскольку в проекте есть необходимость сохранения данных, даже после изменения, то для таких целей логии подходят максимально верно. При изменении данных будет создавать лог со старыми данными сущности, что запретит пользователю бесследно изменять информацию, например, о свое профиле.

За счет типа логов можно понять масштаб проблемы того или иного лога. Например, если лог имеет тип «info», то это говорит о том, что данный лог говорит об обычной информации, а если тип «catch», то это уже исключение, на которое стоит обратить внимание администрации. Каждый из логов также имеет ip-адрес, с которого пришел запрос, это необходимо для дальнейшего роста, а, следовательно, блокировка по ip-адресу, если такая необходимость возникнет.

Таблица user содержит полное описание пользователя. Данная таблица позволяет выполнять все базовые потребности пользователя, например, такие как обновление информации об аккаунте. Ее поля были рассчитаны на большое число функций.

Например, поля rating и comment_count, по факту, можно было получать отдельными SQL запросами, но такой подход будет сильно грузить систему, так как в случае, когда полей нет, в запросе на получение пользователя необходимо обратиться к таблице комментариев, из-за чего сложность данного запроса увеличится прямо пропорционально количеству комментариев в таблице.

Особое внимание также стоит уделить таблице template_trips, так как в проекте было принято решение об ограниченном количестве шаблонов у одного пользователя, было принято решение хранить шаблон поездки в формате JSON-объекта, так как это позволяет значительно увеличить скорость выборки одного шаблона, схема оптимизации такая же, как и для полей rating и comments_count у таблицы users.

2.5 Реализация работы с маршрутами

Для работы с маршрутами было принято решение использовать уже готовые сервисы при помощи API. Для работы с картами был развернут сервер Open Street Map, который отображает карту, развернутую на сервере. Для этого использовалась технология Docker в которой хранятся все карты сервиса.

Помимо самих карт было реализовано нестандартной подход к хранению путей. Все маршруты хранятся в полилинии, так как для отображения маршрута возможно использовать только массив промежуточных точек, ибо при получении списка маршрутов необходима информация о пути следования.

Для создания маршрута была реализована функция дробления пути на стороне сервера, так как пользователям необходимо найти на маршруте точки в соответствии со временем, поэтому был разработан алгоритм для создания промежуточных точек. Для этого были создать условные

единицы для расчёта, момента времени нахождения водителя в конкретной точке.

Хранить каждую точку с интервалом в 1 миллисекунду ненадежно и нерационально по объему занимаемого места. Поэтому была добавлена условная средняя скорость по городу. Таким образом, мы можем разделить маршрут на необходимы интервалы, и получить время маршрута в минутах.

Такой способ является приблизительным, так как рассчитывать все значения с абсолютной точностью затратно по времени и по оперативной памяти сервера. Для реализации данной задачи была создана следующая рекурсивная функция (Приложение А):

Рекурсивная функция drop получает на вход массив с промежуточными точками, а на выходе отдает тот же маршрут, но только уже разбитый на необходимые отрезки. Данная функция проходит по массиву и считает расстояние между точками, для подсчета расстояния используется теорема Пифагора.

Если расстояние больше указанной в конфигурационном файле, тогда отрезок делится на 2 равные части и рекурсивно запускает обработку этого отрезка еще раз и так до тех пор, пока не будет отрезков, больше указанных в конфигураторе. Тем самым, за очень короткое время и с минимальными затратами памяти, был получен массив, который имеет в среднем отрезки меньше тех, которые указаны в конфигураторе.

2.6 Реализация личного кабинета пользователя web–приложения

Для реализации личного кабинета пользователя на стороне сервера использовалась технология JSON Web Token (JWT) – открытый стандарт (RFC 7519) для создания токенов доступа, основанный на JSON формате, для аутентификации и авторизации пользователя.

Аутентификация пользователя проходит в несколько шагов. При получении запроса на аутентификацию контроллер обрабатывает запрос. Он проверяет, была ли произведена с данного ip-адреса попытка аутентифицироваться до этого. Если пользователь делает запрос в первый раз, то в таблице с попытками создается новая запись для данного ip, которая записывает время последнего обращения и сам ip-адрес.

Если ip был найден в таблице, то идет проверка того, насколько давно был сделан данный запрос, если минимальное время интервала не прошло, то пользователю выводится сообщение об ошибке.

Если пользователь не превысил лимит интервалов, то контроллер проверяет наличие email-адреса в БД, если email отсутствует, то контроллер выводит сообщение об ошибке и сообщает, что пользователь не найден, в противном случае пользователь переходит к следующему.

Далее контроллер производит проверку пароля между паролем в БД и паролем, который пришел от пользователя.

Если пароли не совпали, то пользователю выводится сообщение об ошибке и обработка прекращается, если все прошло успешно, то пользователь идет дальше.

При авторизации токен записывается в БД и закрепляется за пользователем, после чего подписывается, где получает срок жизни и отдается пользователю. Такой подход делает полностью бесполезным украденный токен, так как у него есть ограниченный срок жизни, а в случае необходимости токен можно будет удалить из БД [6].

Для проведения авторизации был создан промежуточный обработчик. В случае, если контроллер рассчитан на аутентифицированного пользователя, то перед вызовом контроллера запускается данная промежуточная функция.

Она обрабатывает запрос, получает из заголовка токен пользователя и расшифровывает его, если токен расшифровывается без ошибок и находится в БД у пользователя, тогда в запрос добавляется информация о

пользователе, и он идет далее в обработку контроллера при помощи вызова функции next.

2.7 Реализация личного кабинета администратора web–приложения

Для реализации личного кабинета администратора было принято решение использовать фреймворк Bootstrap–4, он позволяет без излишних затрат по времени реализовать доступный и интуитивно понятный интерфейс.

Также, для потенциального масштабирования было принято решение использовать шаблонизатор ejs, который позволяет при написании малого количества кода собирать страницы так, как будет удобно пользователю. На рисунке 11 показана реализация кабинета администратора.

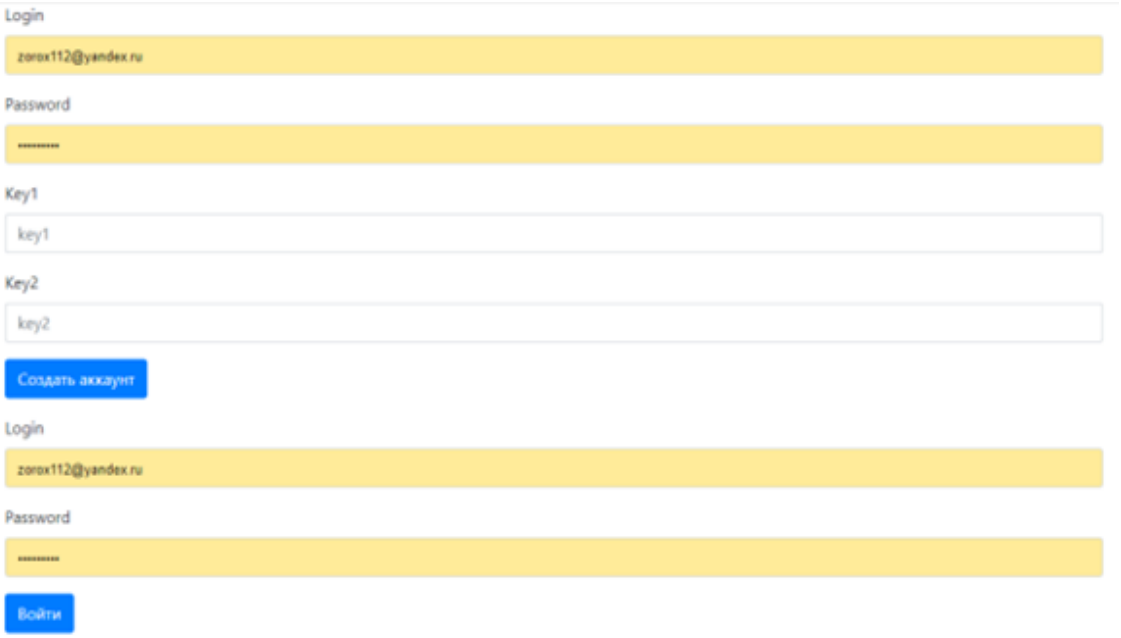

Рисунок 11 – Личный кабинет администратора

В первую очередь были созданы:

— формы регистрации/логина;

— роуты обработчика логина;

— роуты обработчика регистрации;

— ejs шаблоны.

В форме регистрации есть 2 поля с ключами. Ключи выставляются в конфигурационном файле, и без их наличия нельзя создать нового администратора.

Обработчики логина и регистрации администратора вначале проверку, как и роут авторизации на интервал запроса, он берется из конфига, на данный момент там указано значение в 5 минут, это значит, что попытки по созданию или логину администратора можно осуществлять только 1 раз в 5 минут.

После проверки на то, что пользователь отправляет запросы не чаще, чем 1 раз в 5 минут идет проверка на то, доступна ли функция создания администратора вообще. В конфигурационном файле можно выключить возможность создания администраторов, чтобы обезопасить проект от взлома.

Таким образом, все аккаунты администраторов возможно создать сразу и запретить данный функционал в дальнейшем. Далее идет проверка ключей, если оба ключа совпали, то пользователю создается администратор, если нет, то пользователь получает сообщении об ошибке. Все действия также логируются, чтобы администратор имел возможность просмотра попыток взлома панели администратора.

Аутентификация и авторизация администратора проводится за счет сессии, так как панель администратора всегда находится на том же домене, что и сервер, то сессия позволит обезопасить администратора, от нежеланного воздействия со стороны, а не позволит даже случайно

передать токен доступа, так как из JavaScript можно будет запретить доступ к сессии [15].

Сессия хранится на стороне клиента, ее нельзя подделать за счет подписи, изменить или получить значения сессии возможно, если сервер это разрешит, но подделать подпись – нет, из-за чего изменения данных в сессии становится бесполезным.

В роуте для аутентификации идет сначала поиск администратора по его логину, если такой логин не был найден в БД, то пользователю выдается сообщение об ошибке, иначе идет проверка пароля, который пришел и пароля из БД. Если пароли не совпали, то выводится сообщение об ошибке.

Если пароли совпали, то для пользователя создается сессия, и его перенаправляет на следующую страницу. Все сессии в проекте хранятся в оперативной памяти, так как количество администраторов будет минимальным.

Все действия при аутентификации логируются, а также, сообщения об ошибках не несут в себе большого смысла, это было сделано из соображений защиты, чтобы пользователь не мог понять, что именно он делает не так, либо он ошибся с паролем, либо с почтой.

Из сессии пользователя доставался логин администратора, если этот логин был в БД, то пользователь попадает на панель администратора, если нет, то его переносит на страницу входа в панель администратора. Данная промежуточная функция запускается на всех роутах, доступ к которым должен иметь только администратор.

Для отображения интерфейса были написаны ejs шаблоны. За счет данных шаблонов есть возможность создавать изменяемые навигационные панели, а также создавать пагинацию в любое время.

Роут делает запрос для конкретной страницы, а также передает параметры поиска в саму функцию. После чего роут собирает ejs шаблон со своими данными и отдает его пользователю, как статичную HTML

страницу (Приложение Б). На рисунке 12 изображен конечный интерфейс администратора.

Рисунок 12 – Конечный интерфейс администратора

Для редактирования пользователя были написаны такие же обработчики, только было сделано приведение типов, для булевых типов данных (Приложение В).

Конечный интерфейс редактирования пользователя изображен на рисунке 13:

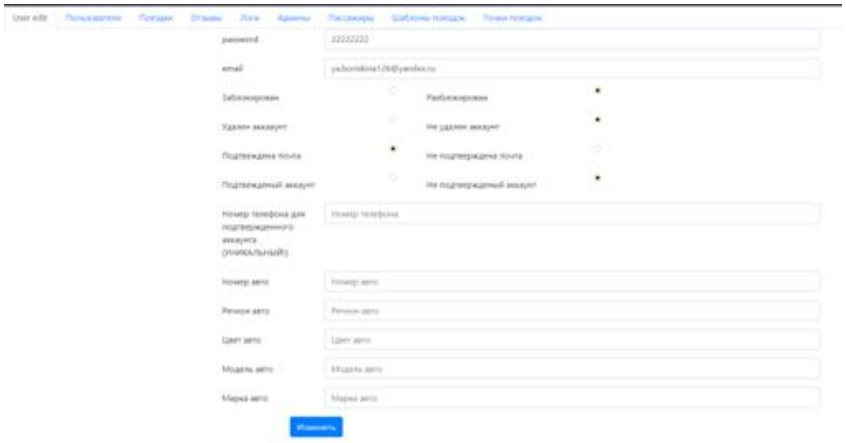

Рисунок 13 – Интерфейс редактирования пользователей

Данный интерфейс позволяет изменять всю доступную информацию о пользователе.

В главе 2 была рассмотрена реализация райдшеринговой системы, опираясь на поставленные в главе 1 задачи и требования.

В качестве средств реализации были выбраны язык программирования JavaScript и фреймворк React. Архитектура системы

состоит из 4х частей: пользовательский интерфейс, личный кабинет администратора и сервер, на котором обрабатываются запросы и хранится база данных.

Для хранения записей была выбрана СУБД PostrgreSQL. Для работы с картами был использован сервис OSM, поднят сервер через Docker и разработан алгоритм нахождения поездок.

3 Тестирование разработанной информационной торговой системы

3.1 Описание процесса тестирования

Тестирование программного обеспечения – процесс исследования программного обеспечения с целью получения информации о качестве продукта. Существует множество подходов к решению задачи тестирования и верификации кода [7].

Качество программных средств можно определить, как совокупную характеристику исследуемого ПО с учётом следующих составляющих: надёжность, сопровождаемость, практичность, эффективность, мобильность, функциональность, доступность.

На сегодняшний день выделяют ряд тестов для проверки разработанной системы, ниже представлены некоторые из них: Модульное тестирование – тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция.

Часто модульное тестирование осуществляется разработчиками. Регрессионное тестирование – после внесения изменений в очередную версию программы, регрессионные тесты подтверждают, что сделанные изменения не повлияли на работоспособность остальной функциональности приложения. Регрессионное тестирование может

выполняться как вручную, так и средствами автоматизации тестирования.

Интеграционное тестирование – тестируются интерфейсы между компонентами, подсистемами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.

Системное тестирование – тестируется интегрированная система на её соответствие требованиям.

Альфа-тестирование – имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования.

Бета-тестирование – в некоторых случаях выполняется распространение версии с ограничениями (по функциональности или времени работы) для некоторой группы лиц, с тем чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Предполагается использование метода альфа и бета тестирования разработки в качестве метода тестирования программы. Этот метод тестирования предполагает сбор команды QA специалистов с заранее готовыми BPMN-диаграммами для всех адаптивных состояний сервиса, а также с подробно описанными пользовательскими сценариями. Данный метод тестирования является наиболее эффективным для проверки всех модулей и системы в целом.

Таким образом, гарантируется, что собранный проект будет удовлетворять условиям установленных бизнес-правил. Кроме того, прохождение альфа и бета тестов поможет сгенерировать новые функции для следующей итерации проекта при помощи обратной связи участников

тестирования. При передаче фонду очередной версии системы проводятся интеграционные тесты, проверяющие взаимодействие всех модулей ИС в виде единого целого.

Каждая ошибка в системе вносилась и подробно описывалась в системе управления проектами Trello благодаря которой велся процесс тестирования. Все ошибки были устранены и исправлены.

3.2 Оценка качества пользовательского интерфейса

В ходе проделанной работы была создана информационная система для осуществления совместных поездок. На очередном этапе необходимо дать оценку эффективности данной разработки. Был закрыт первый этап проекта перед фондом, реализованы все необходимые функции сервиса.

Эксперты фонда после предъявленного научно-технического отчета сочли все обязательства выполненными и был подписан договор на реализацию второго этапа проекта.

3.3 Требования к программно-аппаратным средствам ИС

На данном этапе требуется описать требования к платформе, на которой предполагается развертывание созданной информационной системы.

Исходя из выбранных технологий реализации ИС для функционирования системы потребуются следующие программные средства на стороне сервера:

— Docker;

— PostgreSQL Server;

— Операционная система, поддерживающая работу с протоколом SSH;

— Фильтрация трафика.

Техническое обеспечение (ТО) представляет собой комплекс технических средств, предназначенных для обработки данных в рамках информационной системы. В состав входят 2 сервера, осуществляющий обработку информации, средства сбора и регистрации информации, средства накопления, хранения данных и выдачи результатной информации.

Анализ использования аппаратных ресурсов сервера с выполняющейся на нем ИС при альфа-тестировании продукта показал, что для внедрения данного проекта потребуется несколько смартфонов (минимум 3шт.) со следующими характеристиками (Таблица 10).

Таблица 10 - Технические требования

Наименование	Характеристика (рекомендуется использовать современные решения)
Центральный процессор	ЦП с частотой не менее 2GHz
Оперативная память	4GB
Периферия (устройства ввода/вывода)	стандарт
Экран	различных моделей телефонов от 4.7 до 9 дюймов

В результате проведенных в третьей главе настоящего проекта исследований и работ:

— Иллюстрирована технология тестирования и пример проведения тестов (Приложение Г).

— Сформулированы требования к программно-аппаратным средствам ИС.

Заключение

Для создания web–приложения было проведено исследование решений по созданию экспертной системы автоматического бронирования и создания поездок, которые привели к ряду изменений в конечной системе и ее дизайне. Например, в ходе разработки была переработана схема БД, так как ее изначальный вариант не удовлетворял потребностям в оптимальных запросах.

Для создания маршрутов было исследованы оптимальные алгоритмы по поиску совместных маршрутов. Для реализации алгоритма были разработаны собственные способы хранения маршрутов, для увеличения оптимальности реализации сервиса в целом.

Для отображения маршрутов и их создания был разработан ряд функций на основе Open Street Map со следующим функционалом: поиск по построение оптимального пути с помощью алгоритма поиска на графах. Данный набор функций позволил значительно уменьшить время, которое было необходимо серверу на создание ответа на запрос.

Разработка web–приложения включала в себя проектирование и создание архитектуры сложного клиент-серверного продукта. Для создания этого приложения было разработано более 5 сложных функций, которые привели к оптимизации процессов в системе.

Были приняты решения по оптимизации БД, которые позволили сократить скорость обработки запросов и количество сущностей. Для работы с аутентификацией пользователя были применены 2 разных метода, которые полностью удовлетворяли необходимые потребности задачи.

В процессе выполнения бакалаврской работы были закрыты два этапа программы УМНИК в рамках которого были выделены денежные средства для реализации данного проекта, на условиях указанных в

соглашениях для выполнения НИР. Итогом сотрудничества стал научно-технический результат, заявленный в процессе подписания соглашения.

Список используемой литературы

1. Амблер С. Гибкие технологии: экстремальное программирование и унифицированный процесс разработки. Библиотека программиста [Текст]: - Спб.: Питер, 2005. - 412 с.
2. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS. 2-е изд. / Пер. с англ.; Под общей редакцией проф. С. Орлова — СПб.: Питер, 2006. — 736 с.
3. Варзунов А. В., Торосян Е. К., Сажнева Л. П., Анализ и управление бизнес-процессами // Учебное пособие. – СПб: Университет ИТМО, 2016. –112 с.
4. Введение в реальный ITSM / Роб Ингланд; Пер. с англ. – М.: Лайвбук, 2010. – 132 с.
5. Ларман К. Применение UML 2.0 и шаблонов проектирования. Практическое руководство. 3-е издание. [Текст]: Пер с англ. - М.: ООО «И.Д. Вильямс», 2009. - 736 с.: ил. - Парал. тит. англ.
6. Леоненков А. В. Самоучитель UML 2. - БХВ-Петербург, 2007 — 576 стр. - 3000 экз. - ISBN 978-5-94157-878-8.
7. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения.: Пер. с англ. — СПб: Питер, 2019 — 410 стр. — ISBN: 978-5-4461-0772-8.
8. Мартин Р. Чистый код: создание, анализ и рефакторинг. : Пер. с англ. — СПб: Питер, 2018 — 464 стр. — ISBN: 978-5-496-00487-9.
9. Моделирование бизнес-процессов: учебник и практикум для академического бакалавриата / О. И. Долганова, Е. В. Виноградова, А. М. Лобанова ; под ред. О. И. Долгановой. — М. : Издательство Юрайт, 2016. — 289 с. — Серия : Бакалавр.

10. Орлов С. А. Технологии разработки программного обеспечения: Учебник. — СПб: Питер, 2002. — 464 с. — ISBN 5-94723-145-X.

11. Орлов, С. А. Теория и практика языков программирования. — СПб: Питер, 2017 — 688 стр. — ISBN: 978-5-4461-0491-8.

12. Проектирование информационных систем [Электронный ресурс] 2009. — Режим доступа: http://www.intuit.ru/department/se/devis/6/, свободный. — Загл. с экрана. — Яз. Рус.

13. Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализа прецедентов [Текст]: Пер. с англ. - М.: ДМК Пресс, 2002. - 160 с.: ил. (Серия «Объектно-ориентированные технологии программирования»)

14. Современный экономический словарь. / Под редакцией Райзберг Б.А., Лозовский Л.Ш., Стародубцева Е.Б.– М.: Инфра-М., 2006.

15. Унифицированный язык моделирования [Электронный ресурс] .2008. — Режим доступа: http://ru.wikipedia.org/wiki/UML, свободный. — Загл. с экрана. — Яз. Рус.

16. Фаулер М., Скотт К. UML. Основы, 3-е издание. — СПб: Питер, 2005 — 192 с. — 2000 экз. — ISBN: 5-93286-060-X.

17. Цуканова О. А. Методология и инструментарий моделирования бизнес-процессов: учебное пособие – СПб.: Университет ИТМО, 2015. – 100 с.

18. Открытые системы: Журнал/Издательство «Открытые системы» - Москва.: №06 2003.

19. Бизнес-процесс [Электронный ресурс]. 2011. — Режим доступа: http://ru.wikipedia.org/wiki/Бизнес-процесс, свободный. — Загл. С экрана. — Яз. Рус.

20. Ride chaining [Электронный ресурс]. 2019 — Режим доступа: https://patents.justia.com/patent/11004343, свободный.

21. Overhead view image generation [Электронный ресурс]. 2019 - Режим доступа: https://patents.justia.com/patent/11004343 , свободный.

22. APPROACHES FOR ENCODING ENVIRONMENTAL INFORMATION [Электронный ресурс]. 2019 — Режим доступа: https://patents.justia.com/patent/20210124350 , свободный.

23. Utilizing artificial neural networks to evaluate routes based on generated route tiles [Электронный ресурс]. 2019 — Режим доступа: https://patents.justia.com/patent/10989544 , свободный.

24. Display screen or portion thereof with graphical user interface [Электронный ресурс]. 2019 — Режим доступа: https://patents.justia.com/patent/D916896 , свободный.

Приложение А

Рекурсивная функция drop

```
const {accuracy} = require("../../config/config");
function drop ({ array }){
let arr  = [], newX, newY;
for(let i = 0; I < array.length – 1; i++){
let delX = array[i][0] – array[i+1][0];
let delY = array[i][1] – array[i+1][1];
let pow2katX = delX*delX;
let pow2katY = delY*delY;
// расстояние в пол квартала;
If(pow2katX + pow2katY > accuracy){
newX = array[i][0] – ((array[i][0] – array[i+1][0]/2);
newY = array[i][1] – ((array[i][1] – array[i+1][1]/2);
drop({
        array: [array[i] – ((array[i][0] – array[i+1][0])/2];
        check false,
}}.map((item: I)=>arr.push(item));
} else {
        arr.push(array[i+1]);
}
}
return arr;
}
module.exports = function f{{array}} {
        let q = array[0];
        let arr = drop ({array: array});
arr.unshift(q);
return arr;}
```

Приложение Б

Ejs шаблоны

```
routrer.get("/coments", async (req,res) => {
const page = req.query.page?req.query.page: 0;
const comments = await findCommentsForAdmin({
page,
params: req.query,
});
res.status(200).render('pages/comments', {
nav,
current_id 4,
comments,
page,
type: "comments",
params: req.query,
})
});
```

Приложение В

Ejs шаблоны для булевых типов данных

```
Object.keys(req.query).map(item =>) {
if(req.query[item] === "false") {
req.query[item] = false;
}
if(req.query[item] === "true") {
req.query[item] = true;
}
});
```

Приложение Г

Технология тестирования

Было проведено альфа и бета тестирования в процессе которых были выявлены ошибки при первой итерации разработки. Все проблемы были занесены в систему управление проектами Trello для дальнейшего устранения (рисунок Г.1).

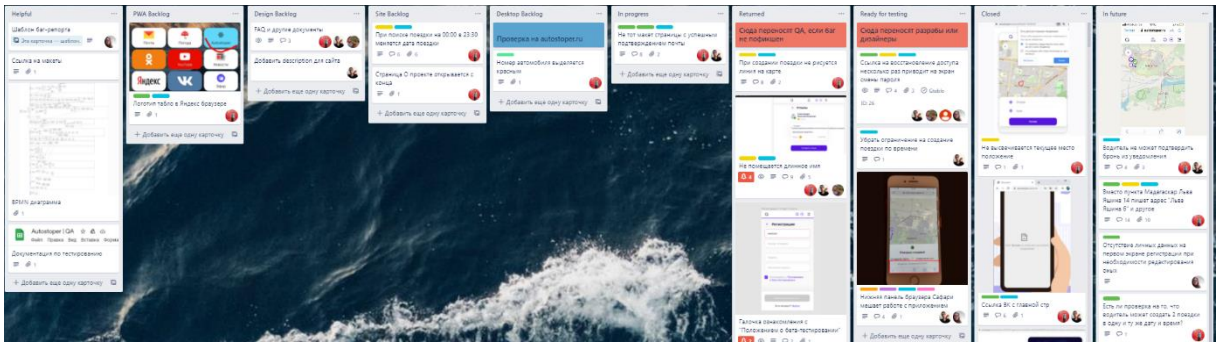

Рисунок Г.1 – Система управления проектами в Trello