

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Информационная система предоставления
услуг для автопутешественников**

Студент	_____	Прошина Ольга Александровна
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	Иванчуков А.Г.
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	Такташкин Д.В.
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	Козлов А.Ю.
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	Попова Н.А.
	(подпись)	(фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Прошина Ольга Александровна гр. 19 ВИ1 факультета ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы «Информационная система предоставления услуг для
автопутешественников»

Тема утверждена приказом ПГУ № 321/0 от "___" апреля 20___

3. Руководитель работы Иванчуков А.Г.

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

1) Назначение: информационная система предоставления услуг для автопутешественников
предназначена для предоставления помощи выбора путешествия с оптимальным бюджетом

2) Среда выполнения: ОС Windows 10 и выше

Языки разработки: C#, JS, HTML, CSS

Инструментальные средства: VSCode, Microsoft Visio, SQL Server Express 2019

Технологии разработки: ООП, ASP.NET Core 6.0

3) Основные функции приложения:

а) Сбор информации о бюджете поездки

б) Оценка путешествия

в) Просмотр рейтинга посещаемых мест

г) Яндекс карты

д) Расчет затрат на бензин

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области

2) Анализ функциональных требований

3) Планирование разработки и оценка бюджета

4) Проектирование информационной системы

5) Реализация

6) Контроль качества

7) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	10	12.04.2023 – 19.04.2023	
2	Анализ функциональных требований	5	20.04.2023 – 23.04.2023	
3	Планирование разработки и оценка бюджета	10	24.04.2023 – 01.05.2023	
4	Проектирование информационной системы	10	01.05.2023 – 09.05.2023	
5	Реализация	20	10.05.2023 – 24.05.2023	
6	Контроль качества	5	25.05.2023 – 28.05.2023	
7	Оформление пояснительной записки	5	29.05.2023 – 01.06.2023	

Дата выдачи задания " 22 " _____ декабря _____ 2022 _____

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 85 листов, 21 рисунок, 11 таблиц, 12 использованных источников и 3 приложения.

С#, JAVASCRIPT, CSS, HTML, WEB – ПРИЛОЖЕНИЕ, АНАЛИЗ ДАННЫХ, ASP.NET CORE, VISUAL STUDIO CODE

Объект разработки – web – приложение предоставления услуг для автопутешественников.

Цель работы – разработка web – приложения предоставления услуг для автопутешественников, позволяющего планировать маршруты, находить подходящие места для остановки и предоставлять информацию о затраченном бюджете.

Разработка серверной части приложения осуществлялась на языке программирования С#. Для реализации серверной части использовалась среда разработки Visual Studio Code. В качестве СУБД использовалась SQL Server Express 2019.

Разработка клиентской части приложения осуществлялась на языке программирования JavaScript. Для реализации клиентской части приложения использовалась среда разработки Visual Studio Code. В качестве вспомогательных языковых средств разработки использовались языки HTML и CSS.

					ПГУ 09.03.02 – БР191.18ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Информационная система предоставления услуг для автопутешественников Пояснительная записка	Лит.	Лист	Листов
Разраб.		Прошина О.А.						
Провер.		Иванчуков А.Г.					4	85
Н. Контр.		Такташкин Д.В				группа 19ВИ1		
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области	11
1.1 Анализ предметной области и постановка задачи.....	11
1.2 Анализ доступных приложений для автопутешествий.....	12
1.2.1. Приложение для путешествий «Яндекс.Путешествия».....	12
1.2.2. Приложение для автопутешествий «iOverlander»	12
1.2.3. Приложение для автопутешествий «MultiGo».....	13
1.3 Сравнительный анализ аналогов	13
1.4 Функционально-стоимостный анализ	18
2 Анализ информационных технологий разработки	21
2.1 Инструментальные средства проектирования	21
2.1.1 Язык моделирования UML.....	21
2.1.2 Язык моделирования IDEF1X.....	22
2.2 Инструментальные средства разработки	22
2.2.1 Язык программирования C#.....	23
2.2.2 Платформа разработки ASP.NET	23
2.2.3 Язык разработки JavaScript	24
2.2.4 Выбор СУБД.....	25
3 Анализ требований.....	26
3.1 Функциональные требования.....	26
3.2 Модель предметной области.....	30
4 Проектирование.....	32
4.1 Проектирование серверной части.....	33
4.2 Проектирование клиентской части	36
4.3 Проектирование пользовательского интерфейса.....	37
5 Реализация.....	39

5.1 Реализация серверной части приложения	39
5.2 Реализация клиентской части приложения	39
5.3 Реализация пользовательского интерфейса	43
6. Тестирование информационной системы автопутешествий	47
6.1 Выбор методологии тестирования	47
6.2 Тестирование чёрным ящиком	49
6.3 Расчет числовых метрик программы	51
6.3.1 Метрика Холстеда.....	54
6.3.2 Цикломатическая сложность программы	56
Заключение	59
Список использованных источников	60
Приложение А. Глоссарий	62
Приложение Б. Исходный код серверной части	64
Приложение В. Исходный код клиентской части.....	76

Введение

С развитием интернета открывается все больше возможностей для эффективной работы с удалёнными подразделениями предоставления услуг через онлайн-платформы. Новые технологии и увеличение скорости интернет-соединений существенно расширяют влияние интернета на повседневную жизнь людей. В современном мире особенно важными стали скорость и доступность предоставления услуг, что способствовало развитию интернет-приложений для удовлетворения потребностей пользователей [1].

Автотуризм – это путешествие, осуществляемое с использованием различных транспортных средств. Он предоставляет возможность передвигаться из города в город, затрачивая минимальное количество денежных средств, что делает его доступным для людей различного достатка. Путешествие на автомобиле обладает рядом преимуществ:

- полная свобода выбора маршрута и мест для посещения, так как путешественник самостоятельно определяет свои предпочтения и приоритеты;
- относительно невысокая стоимость всего путешествия по сравнению с другими видами транспорта;
- мобильность и свобода принятия решений во время путешествия;
- относительная экономия времени при перемещении, так как автомобильное движение обычно быстрее, чем поезд [2].

В настоящее время автотуризм становится все более популярным средством отдыха. Однако путешествующие на автомобиле часто сталкиваются с проблемами поиска информации о местах для остановки, подбора оптимального маршрута и расчета примерного бюджета. Все это может вызывать дополнительный стресс и отнимать много времени у путешественников. Разработка информационной системы, предоставляющей услуги для автопутешественников, становится эффективным решением этих проблем.

Создание такой информационной системы является актуальной задачей в современном мире, поскольку все больше людей предпочитают путешествовать на автомобиле, стремясь получить максимальный комфорт и удобство во время поездки. Однако, несмотря на популярность автотуризма, путешествующие на автомобиле сталкиваются с необходимостью самостоятельно собирать и анализировать информацию о маршруте, отелях, достопримечательностях, ресторанах и других объектах, что может быть трудоемким и неэффективным процессом.

Информационная система для автопутешественников, разрабатываемая в рамках данной дипломной работы, позволит решить эти проблемы и обеспечит пользователей надежными и актуальными данными для планирования и осуществления их путешествий. Пользователи смогут легко планировать оптимальные маршруты, находить подходящие места для остановки и получать информацию о различных аспектах путешествия, таких как расходы, пробки на дороге, погода и другие факторы, влияющие на комфортность поездки.

В ходе выполнения данной работы будет проведен процесс разработки и реализации информационной системы, основываясь на современных технологиях и лучших практиках разработки программного обеспечения. Также будут учтены потребности пользователей и предоставлены удобные функциональные возможности системы, которые способствуют планированию и осуществлению автопутешествий.

В результате успешной реализации информационной системы для автопутешественников ожидается повышение удовлетворенности пользователей и облегчение их путешествий, улучшение качества и эффективности планирования маршрутов, а также снижение стресса и экономия времени при подборе мест для остановки и расчете бюджета.

Целью выполнения выпускной квалификационной работы является создание системы, которая осуществляет процесс подбора информации о

автопутешествиях и предоставляет возможность делиться впечатлениями о прошедшем путешествии.

Актуальность данной дипломной работы заключается в необходимости разработки приложений, предоставляющих сопровождающий автопутешественников в поездке функционал. Такое приложение может помочь путешественникам выбрать маршрут, рассчитать расстояние и время в пути. Автопутешественникам, которые едут в дальние поездки на автомобиле, и заранее хотят рассчитать примерный бюджет в дороге, необходимо иметь возможность вести расчет затрат на бензин. Это поможет им спланировать свои расходы на поездку и избежать непредвиденных ситуаций в дороге. Путешествующие на автомобиле часто сталкиваются с проблемами поиска информации о местах для остановки, выбора маршрута и расчета бюджета, что может вызывать дополнительный стресс и затраты времени.

Цель данной работы заключается в создании информационной системы, специализированной для автопутешественников, которая позволяет эффективно планировать маршруты, находить подходящие места для остановки и предоставлять пользователю полезную информацию о затраченном бюджете и других аспектах путешествия. Данная система будет обладать преимуществами в сравнении с существующими решениями, такими как более точное определение оптимальных маршрутов с учетом предпочтений пользователя, предоставление подробной информации о местах остановки, включая отзывы и рекомендации от других путешественников, а также возможность делиться своими впечатлениями и рекомендациями.

Для достижения поставленной цели выпускной квалификационной работы будут выполнены следующие задачи, описанные ниже.

1. Изучение существующих информационных систем, связанных с автопутешествиями. Для этих целей предполагается исследование и анализ существующих решений, чтобы выявить их преимущества и недостатки. Это

позволит определить требования к разрабатываемой системе и учесть лучшие практики в процессе разработки.

2. Проектирование системы. Для этих целей предполагается разработать детальный план системы, включая архитектуру, функциональные модули, базу данных и пользовательский интерфейс. Будут определены основные алгоритмы и методы, необходимые для обеспечения функциональности системы, такие как поиск оптимальных маршрутов, интеграция с внешними сервисами и предоставление информации о различных объектах и услугах для автопутешественников.

3. Разработка и реализация системы. Для этих целей предполагается создание информационной системы с использованием современных технологий и инструментов разработки. Важным аспектом является обеспечение удобного пользовательского интерфейса, интуитивной навигации и надежной работы системы.

4. Тестирование и отладка системы. Для этих целей предполагается проведение тестов функциональности и производительности системы, а также исправление ошибок и устранение неполадок. Это позволит обеспечить высокое качество и надежность работы системы перед ее внедрением.

Таким образом, в ходе данной работы будет рассмотрен процесс разработки и реализации информационной системы предоставления услуг для автопутешественников.

1 Анализ предметной области

1.1 Анализ предметной области и постановка задачи

Предметная область «информационная система предоставления услуг для автопутешественников» включает в себя различные аспекты, связанные с организацией автопутешествий и предоставлением услуг для этой категории туристов.

Автопутешествие – это путешествие на автомобиле с целью ознакомления с новыми местами и культурами, а также для персонального отдыха и развлечений.

Автопутешественник – это человек, который путешествует по миру за рулем автомобиля и наслаждается свободой перемещения.

У автопутешественника могут существовать различные потребности, такие как: проживание в удобном и безопасном месте, перемещение по маршруту, питание, доступ в Интернет, экономия денежных средств и общение с местными жителями для обмена опытом. Также важным аспектом является наличие информации о достопримечательностях, интересных маршрутах, событиях и мероприятиях в регионах, по которым будет проходить путешествие. Для автопутешественника важна свобода и независимость в принятии решений и выборе мест для остановки.

В целом, приложение автопутешествий должно предоставлять пользователям широкий спектр функций для планирования и организации своих путешествий. Пользователи должны иметь возможность создавать и настраивать маршруты, искать информацию о местах, которые можно посетить, делиться своим опытом в новостной ленте, а также функции для расчета времени в пути и расхода топлива.

1.2 Анализ доступных приложений для автопутешествий

Был проведен анализ наиболее популярных приложений для автопутешествий, который основывался на исследовании предметной области. Далее приведен обзор нескольких из них.

1.2.1. Приложение для путешествий «Яндекс.Путешествия»

Яндекс Путешествия – это популярный сервис для путешественников, который предоставляет широкий выбор вариантов размещения в разных городах мира. Здесь можно найти информацию об отелях, гостиницах, апартаментах, хостелах и других вариантах размещения. Пользователи могут забронировать отель, ознакомиться с отзывами других гостей, посмотреть фотографии номеров и многое другое [3].

Особенностью данного приложения является интеграция с другими сервисами Яндекса: Яндекс.Карты, Яндекс.Такси, Яндекс.Расписания транспорта и другими. Это позволяет пользователю получить всю необходимую информацию о поездке, используя только один сервис. Так же в приложении имеется разнообразие предложений. В сервисе представлены не только готовые туры и путёвки, но и возможность самостоятельно подобрать маршрут для поездки.

На главной странице сразу же предлагается возможность выбора транспорта для путешествия, брони отелей, билетов, а также выбора туров.

1.2.2. Приложение для автопутешествий «iOverlander»

iOverlander – это приложение для путешествий на свежем воздухе, специально разработанное для туристов. Сервис включает в себя глобальный каталог кемпингов, стоянок для автофургонов, ресторанов, отелей, заправочных станций и других полезных объектов для путешественников [4].

Особенностью этого приложения является возможность фильтровать места по типу, включая установленные кемпинги, неформальные кемпинги, дикие кемпинги и прочее.

На главной странице доступна возможность ввода пункта назначения, выбора типа места для остановки. Так же в приложении есть возможность указать необходимые удобства.

1.2.3. Приложение для автопутешествий «MultiGo»

MultiGo предоставляет возможность просматривать стоимость топлива, краткое описание конкретной заправочной станции, комментарии и даже фотографии.

Особенностью этого приложения является точность и полнота данных по АЗС, цена бензина, дизельное топливо и газ. АЗС можно просматривать на карте, либо списком. Так же можно настраивать различную фильтрацию: по типу топлива, по бренду АЗС, по типу принимаемых топливных карт и прочего.

На главной странице имеется возможность выбора трассы и поиск ближайших на пути АЗС.

1.3 Сравнительный анализ аналогов

Благодаря анализу существующих аналогов, был выделен перечень основных функций, которые должны присутствовать в приложении для автопутешественников. Так же были определены критерии, которым должно соответствовать приложение.

Каждый из рассмотренных сервисов предлагает уникальные возможности и функции, которые могут быть полезными для разных типов путешествий и индивидуальные внимания путешественников. Яндекс.Путешествие предоставляет информацию о местах, достопримечательностях и отелях, что помогает путешественникам планировать и организовывать свои поездки. iOverlander, с другой стороны, сосредоточен на предоставлении информации о

местах для остановки, кэмпинге и парковке, что делает его ценным ресурсом для автопутешествий. MultiGo занимается оптимизацией автомобильных маршрутов, расчетом затрат на топливо и поиском ближайших заправочных станций, что может быть полезно для путешествий на автомобиле.

В целом, использование таких сервисов может значительно облегчить путешествия, предоставляя удобный доступ к информации и помогая планировать поездки с меньшими трудностями. Однако каждый путешественник должен принять во внимание свои индивидуальные потребности и предпочтения, чтобы выбрать наиболее подходящий для него ресурс или сервис. В связи с этим была проведена сравнительная характеристика аналогов. Ниже приведены критерии:

- пользовательский интерфейс;
- поддержка и обновления;
- функция построения маршрута;
- функция оценки посещенных мест;
- социальные функции;
- наличие функций для расчета затрат на бензин.

Рассмотрим подробно каждый из приведенных выше критериев.

Критерий пользовательский интерфейс заключается в удобстве использования на разных устройствах с различными размерами экранов и разрешениями. Описание и оценка критерия приведены в таблице 1.

Таблица 1 – Критерий пользовательский интерфейс

Оценка	Описание
0	Интерфейс приложения не удобен для использования, интуитивно не понятен
1	Интерфейс приложения удобен для использования, нет затруднения с восприятием логических элементов

Далее рассмотрим критерий поддержки и обновления. Обновления и поддержка являются ключевыми критериями приложения, которые позволяют устранять проблемы и улучшать его производительность. Кроме того, регулярные обновления помогают привлечь новых пользователей и удержать текущих. Поддержка включает в себя предоставление технической помощи пользователям, которые сталкиваются с проблемами при использовании приложения. Обновление предполагает выпуск новых версий приложения, которые содержат исправления ошибок, улучшения и новые функции. Описание и оценка критерия приведены в таблице 2.

Таблица 2 – Критерий поддержка и обновление

Оценка	Описание
0	Обновления не выходят или выходят редко
1	Обновления выходят регулярно, приложение поддерживается разработчиками

Третьим критерием характеристики является функция построения маршрута путешествия. Она позволяет определить оптимальный маршрут для достижения желаемых целей. Описание и оценка критерия приведены в таблице 3.

Таблица 3 – Критерий функция построения маршрута путешествия

Оценка	Описание
0	Построение маршрута в приложении невозможно
1	Построение маршрута в приложении возможно

Затем рассмотрим критерий «функция оценки посещенных мест». Эта функция позволяет пользователям системы оценивать места, которые они посетили, на основе своего опыта и предпочтений. Эта информация полезна для

других путешественников, которые планируют свое путешествие в этот регион. Описание и оценка критерия приведены в таблице 4.

Таблица 4 – Критерий функция оценки посещенных мест

Оценка	Описание
0	Возможность оценить посещенные места не предоставляется
1	Возможность оценить посещенные места предоставляется

Критерий наличия социальных функций, в свою очередь, включает в себя наличие возможности делиться своими маршрутами, а также комментировать места, которые они посещают. Описание и оценка критерия приведены в таблице 5.

Таблица 5 – Критерий наличие социальных функций

Оценка	Описание
0	Возможность делиться впечатлениями о путешествии не предоставляется
1	Возможность делиться впечатлениями о путешествии предоставляется

Критерий наличия функций для расчета затрат на бензин используется для определения ожидаемых затрат на бензин при расчете стоимости передвижения на автомобиле. Функция для расчета затрат на бензин – это инструмент, который позволяет пользователю определить, сколько денег будет потрачено на топливо в зависимости от разных факторов, таких как расстояние, тип автомобиля, расход топлива, стоимость бензина и др. Эта функция полезна в первую очередь для тех, кто часто путешествует на машине, так как позволяет оценить затраты на поездку и более эффективно планировать свой бюджет. Описание и оценка критерия приведены в таблице 6.

Таблица 6 – Критерий наличие функций для расчета затрат на бензин

Оценка	Описание
0	Возможность делиться впечатлениями о путешествии не предоставляется
1	Возможность делиться впечатлениями о путешествии предоставляется

Результаты сравнения аналогов приложения приведены в таблице 7.

Таблица 7 – Сравнительная характеристика продуктов-аналогов

Критерий	Яндекс.Путешествие	iOverlander	MultiGo
Пользовательский интерфейс	1	1	1
Поддержка и обновление	1	1	0
Функция построения маршрута путешествия	1	0	1
Функция оценки посещенных мест	1	0	0
Наличие социальных функций	1	0	0
Наличие функций для расчета затрат на бензин	0	0	0

Проанализировав различные программы-аналоги сервиса автопутешествий, можно сделать вывод, что ни одна из них не обладает всеми необходимыми характеристиками для предоставления комплексного решения запросов пользователей. Некоторые из программ предоставляют только информацию о местах, которые можно посетить во время поездки, в то время как другие позволяют только планировать маршруты, не учитывая специфики пользователя. Более того, большинство программ не имеют полноценной функциональности, поддержки разработчиков. В целом, выбор между

различными программами-аналогами зависит от предпочтений и потребностей каждого пользователя.

1.4 Функционально-стоимостный анализ

Функционально-стоимостный анализ приложения для автопутешественников поможет определить его функциональные возможности и оценить соотношение между затратами на разработку и получаемой от приложения ценностью. Ниже приведены некоторые основные аспекты, которые включены в такой анализ.

Функциональный анализ:

- поиск и отображение информации о различных автопутешествиях и достопримечательностях на карте;
- расчет оптимальных маршрутов и времени в пути с учетом трафика, дорожных условий и предпочтений пользователя;
- предоставление информации о расстоянии, протяженности маршрута, времени в пути, прогнозе затрат на топливо и прочих расходах;
- возможность обмена информацией и впечатлениями с другими автопутешественниками.

Стоимостный анализ:

- оценка затрат на разработку и поддержку приложения, включая затраты на программистов, дизайнеров, тестировщиков, серверное оборудование и прочие расходы;
- определение потенциальных источников дохода, таких как платные подписки, реклама, комиссии за бронирование и т. д.

Функциональность для планирования и организации путешествия:

- возможность создания маршрутных планов с указанием пунктов назначения;
- интеграция с календарем.

Расширенные функциональности:

– предоставление рекомендаций по маршруту и интересным местам, основанных на предпочтениях пользователя, истории его поездок и обратной связи других пользователей;

– возможность добавления и просмотра отзывов и оценок пользователей о различных местах и услугах, связанных с путешествиями.

С помощью программы для работы с электронными таблицами Microsoft Excel был создан план разработки информационной системы для автопутешественников, представленный на рисунке 1 [5].

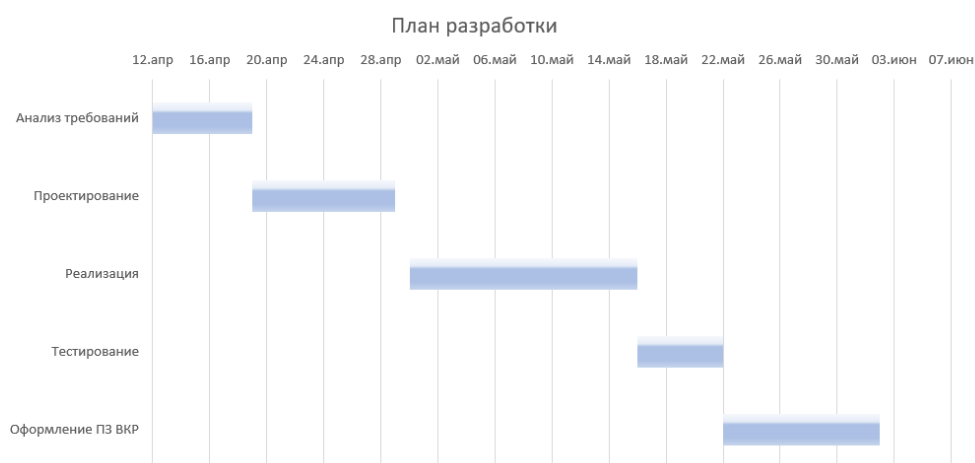


Рисунок 1 – Диаграмма Ганта

После определения задач проекта, на следующем этапе необходимо проанализировать ресурсы и их распределение на каждую задачу.

Таким образом, в ходе выполнения работы по данному этапу были оценены ресурсы проекта. Они включают в себя все необходимое для завершения процесса разработки, например оборудование, материалы, трудовые затраты, а так же оплата разработчикам. Для полноценной оценки стоимости проекта так же было определено количество часов, необходимое для выполнения каждой из задач. Средняя заработная плата разработчиков, аналитиков, тестировщиков по России была взята с сайта <https://gorodrabot.ru>. Ниже приведена таблица 8, в которой поэтапно представлены затраченные ресурсы.

Таблица 8 – Результаты анализа затрат

Название ресурса	Этап	Время выполнения	Затраты
Аналитик	Анализ требований, проектирование	15ч	423р/ч
Программист	Реализация	60ч	554р/ч
Тестировщик	Тестирование	10ч	348р/ч
API Яндекс.Карты	—	—	240р
Итого		43 305 рублей	

В результате проведения функционально-стоимостного анализа разработки приложения была определена стоимость выполнения каждого этапа разработки. Для выполнения были расчётным путём учтены затраты на трудовые ресурсы, использование инфраструктуры, необходимый софт-инструментарий и прочие факторы. Была также определена суммарная стоимость приложения, которая учитывает расходы на все этапы разработки вместе с дополнительными затратами на управление и контроль проекта.

Для выполнения этапа «Анализ требований» и «Проектирование» необходимо 15 часов, что при средней заработной плате аналитика составит 6 345 рублей, для этапа «Реализация» – 33 240 рублей, для этапа «Тестирование» необходимо будет затратить 10 часов, таким образом, стоимость выполнения составит 3 480 рублей. В целом, общая стоимость проекта определяется как сумма стоимостей всех этапов разработки.

Таким образом, в результате функционально-стоимостного анализа, временные затраты для выполнения проекта составят 85 часов, а стоимость его выполнения, с учетом прочих затрат на ресурсы, составит 43 305 рублей. Оценка стоимости является важным элементом планирования проекта и помогает установить бюджет на разработку, а также контролировать затраты в процессе работы.

2 Анализ информационных технологий разработки

2.1 Инструментальные средства проектирования

Для проектирования информационной системы предоставления услуг для автопутешественников использовались языки моделирования UML и IDEF1X. Для разработки диаграмм применялся инструмент Visual Paradigm 17.0.

Рассмотрим подробнее языки моделирования, используемые при проектировании структуры программного обеспечения.

2.1.1 Язык моделирования UML

UML (Unified Modeling Language) – это язык моделирования, созданный для стандартизации процесса проектирования программного обеспечения.

UML представляет собой набор нотаций, которые позволяют инженерам по программному обеспечению создавать диаграммы, описывающие различные аспекты системы: ее структуру, функциональности, процессы, взаимодействие с окружающей средой и т. д.

В состав UML входят различные типы диаграмм, например, диаграммы классов, диаграммы последовательности, диаграммы состояний и т. д. Каждый тип диаграммы описывает конкретный аспект системы.

UML используется для разработки различных типов систем, включая программное обеспечение, аппаратные средства, веб-сервисы и т. д. Он может быть использован для:

- описания требований к системе;
- визуализации структуры и поведения системы;
- анализа и проектирования системы;
- документирования и поддержки системы;
- обмена информацией между разработчиками и пользователем.

UML предоставляет многочисленные преимущества, например:

- улучшение взаимопонимания между разработчиками и пользователями;
- повышение эффективности проектирования и разработки системы;
- сокращение времени разработки и уменьшение затрат на тестирование;
- упрощение документирования системы;
- увеличение качества и надежности системы.

Таким образом, UML является широко используемым языком моделирования, предназначенным для создания стандартизированных диаграмм, описывающих различные аспекты системы. UML предоставляет различные принципы, преимущества и возможности, которые позволяют разработчикам и пользователям создавать высококачественное программное обеспечение [6].

2.1.2 Язык моделирования IDEF1X

Язык моделирования IDEF1X (Integration Definition for Information Modeling) предназначен для разработки информационных моделей, отражающих структуру данных в организации [7].

Методология IDEF1X включает в себя формальный язык моделирования, инструментарий и методы сопровождения проектов в области информационных технологий. Язык моделирования IDEF1X позволяет создавать графические диаграммы, отображающие структуру данных, связи и ограничения на эти данные.

2.2 Инструментальные средства разработки

Для каждого из программных средств выбор средств разработки является вопросом, требующим детального рассмотрения. Необходимо учитывать язык программирования, доступность инструментов, требования к производительности и многие другие факторы. Правильный выбор

инструментов поможет сократить время разработки, повысить качество и надежность программного обеспечения.

Рассмотрим основные инструментальные средства, которые будут использоваться для разработки приложения.

2.2.1 Язык программирования C#

Язык C# является объектно – ориентированным языком программирования, который поддерживает различные конструкции, такие как классы, наследование, полиморфизм и т.д. Он также поддерживает различные типы данных, такие как Int, Float, Double, String и т. д.

Как и большинство языков программирования, программа на C# состоит из классов. Класс определяет структуру объекта, а объект – это экземпляр класса.

Язык C# используется для создания приложений на платформе .NET, включая приложения для Windows, веб – приложения и приложения для мобильных устройств. Его также можно использовать для создания игр, библиотек и других программных компонентов.

C# является одним из наиболее популярных языков программирования и становится все более востребованным в индустрии разработки программного обеспечения.

Таким образом, язык разработки C# предоставляет программистам мощный и гибкий инструмент для написания приложений на платформе .NET. Благодаря своей простоте и продуманной структуре, C# удобен для работы и может использоваться для решения различных задач во многих отраслях разработки программного обеспечения [8].

2.2.2 Платформа разработки ASP.NET

ASP.NET – это бесплатная и открытая кроссплатформенная платформа разработки, которая позволяет создавать веб-приложения и веб-сайты. Она

является частью .NET Framework и поддерживает различные языки программирования, включая C# и Visual Basic [9].

Одним из главных преимуществ ASP.NET является ее интеграция с большинством продуктов и сервисов Microsoft, что делает платформу особенно привлекательной для разработчиков, которые уже используют другие продукты Microsoft. Некоторые другие преимущества ASP.NET включают:

- высокая производительность и масштабируемость;
- широкий выбор инструментов разработки;
- простота разработки и поддержки веб – приложений и веб – сайтов.

Некоторые из основных элементов ASP.NET:

- ASP.NET MVC – фреймворк, который позволяет разработчикам создавать веб – приложения с моделью, представлением и контроллером;
- ASP.NET Web API – фреймворк, который позволяет разработчикам создавать веб-API и веб-службы.

Исходя из вышесказанного, платформа разработки ASP.NET является мощной инструментальной средой для создания веб – приложений и веб – сайтов.

2.2.3 Язык разработки JavaScript

JavaScript – это интерпретируемый язык программирования, который используется для создания динамических и интерактивных веб – страниц и приложений. Язык позволяет создавать интерактивные элементы, обрабатывать события пользователя и изменять содержимое страницы без перезагрузки [10].

Одной из главных особенностей JavaScript является его динамическая типизация. Это означает, что тип переменной может меняться в процессе выполнения программы.

JavaScript также поддерживает объектно-ориентированное программирование, функциональное программирование и асинхронное программирование.

Для разработки информационной системы предоставления услуг для автопутешественников на языке JavaScript необходимо знание других технологий, таких как HTML, CSS, базы данных, а также знание современных технологий веб-разработки, таких как REST API.

В заключение, язык разработки JavaScript является важной технологией для веб – разработки. Он позволяет создавать динамические приложения и обеспечивает легкость разработки и масштабирования.

2.2.4 Выбор СУБД

При разработке информационной системы предоставления услуг для автопутешественников была использована БД SQL Server Express 2019.

При выборе системы управления базами данных (СУБД) были учтены такие факторы, как производительность, масштабируемость, надежность и доступность. SQL Server Express 2019 была выбрана из-за ее характеристик, таких как высокая производительность, масштабируемость, надежность и доступность для различных операционных систем.

Среди ключевых возможностей SQL Server Express 2019 можно выделить:

- поддерживает язык Transact-SQL;
- интеграция с .NET Framework;
- поддержка больших объемов данных.

Таким образом, SQL Server Express 2019 имеет множество возможностей, таких как язык Transact-SQL, интеграцию с .NET Framework и инструменты управления данными.

3 Анализ требований

3.1 Функциональные требования

Функциональные требования к системе представляют собой описание того, какие функции должна предоставлять система, какие условия должны выполняться для ее работы, какие данные и каким образом она должна обрабатывать. Они определяют взаимодействие пользователя с системой и зоны ответственности, а также определяют роль каждого функционального элемента в системе.

Основными требованиями для информационной системы являются:

- регистрация и авторизация пользователей;
- создание маршрута путешествия;
- поиск информации о местах, которые можно посетить в каждом городе;
- пользователи должны иметь возможность делиться своими маршрутами и планами дня, а также комментировать и оценивать места, которые они посещают;
- пользователи должны иметь возможность пользоваться приложением на смартфоне;
- пользователи должны иметь возможность примерного подсчёта денежных затрат на бензин.

Таким образом, информационная система должна обладать удобным интерфейсом, предоставлять широкий спектр услуг и обеспечивать возможность получать данные о маршрутах.

Диаграмма вариантов использования (Use Case diagram) – это графическая модель, которая отображает функциональность системы с точки зрения ее взаимодействия с внешними акторами. Акторами могут быть как конкретные пользователи, так и другие системы. Диаграмма вариантов использования помогает понять, как система будет использоваться, какие

сценарии использования возможны, какие роли выполняют акторы и как взаимодействуют между собой элементы системы.

В данной системе действуют два основных актора: администратор и пользователь.

Администратор должен иметь следующие возможности:

- добавление новых мест;
- добавление и редактирование марки автомобилей;
- добавление типа двигателя автомобиля;
- пройти авторизацию.

Пользователь должен иметь следующие возможности:

- авторизоваться;
- ввод информации о совершённом путешествии;
- чтение отзывов других пользователей;
- оценивать посещенные места;
- просматривать список лучших мест;
- подсчитывать бюджет поездки.

Не зарегистрированному пользователю, в свою очередь, должна предоставляться возможность прохождения регистрации.

Рассмотрим диаграмму вариантов использования в нотации UML, представленную на рисунке 2.

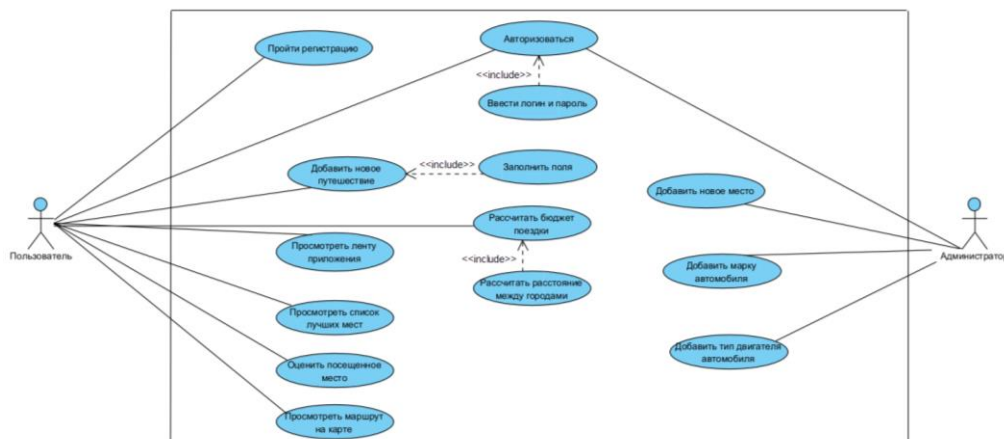


Рисунок 2 – Диаграмма вариантов использования

Вариант использования «Добавить новое путешествие» позволяет пользователю добавлять данные о путешествии. Пользователь может заполнить форму с информацией о путешествии, такой как название места, даты путешествия, бюджет и краткое описание. Также можно указать, кто еще был в поездке, какие достопримечательности посетил и какие впечатления получил. Спецификация представлена на рисунке 3.

Прецедент: добавить новое путешествие
ID: 1
Краткое описание: пользователь добавляет данные о совершенной поездке
Главные актеры: зарегистрированный пользователь
Второстепенные актеры: нет
Предусловия: нет
Основной поток:
1. Пользователь нажимает на кнопку «Оценить место»
2. Пользователь вводит всю необходимую информацию о путешествии в соответствующие поля
3. Пользователь нажимает на кнопку «Отправить»
Постусловия: новое путешествие добавлено

Рисунок 3 – Спецификация варианта использования
«Добавить новое путешествие»

Вариант использования «Оценить посещенное место» позволяет пользователю оценивать места, которые он посетил в период путешествия. Спецификация представлена на рисунке 4.

Прецедент: оценить посещенное место
ID: 2
Краткое описание: пользователь оценивает посещенные места
Главные актеры: зарегистрированный пользователь
Второстепенные актеры: нет
Предусловия: нет
Основной поток:
1. Пользователь нажимает на кнопку «Оценить место»
2. Пользователь вводит название посещенного места
3. Пользователь оценивает место по пятибалльной шкале
4. Пользователь нажимает на кнопку «Отправить»
Постусловия: место оценено

Рисунок 4 – Спецификация варианта использования
«Оценить посещенное место»

Вариант использования «Рассчитать бюджет поездки» позволяет пользователю возможность подсчета примерной суммы, затрачиваемой на

бензин при поездке из одного города в другой. Спецификация представлена на рисунке 5.

Прецедент: рассчитать бюджет поездки
ID: 3
Краткое описание: пользователь подсчитывает примерную сумму, затраченную на бензин
Главные актеры: зарегистрированный пользователь
Второстепенные актеры: нет
Предусловия: нет
Основной поток:
1. Пользователь нажимает на кнопку «Расчет бюджета»
2. Пользователь вводит название пункта отправления
3. Пользователь вводит название пункта назначения
Постусловия: бюджет затрат на бензин в поездке рассчитан

Рисунок 5 – Спецификация варианта использования
«Рассчитать бюджет поездки»

Вариант использования «Авторизоваться» предоставляет пользователю возможность зайти в приложение. После авторизации пользователь может получить доступ к функциям приложения. Спецификация представлена на рисунке 6.

Прецедент: авторизоваться
ID: 4
Краткое описание: пользователь авторизовывается и заходит в приложение
Главные актеры: зарегистрированный пользователь
Второстепенные актеры: нет
Предусловия: нет
Основной поток:
1. Пользователь вводит данные в поле «Логин»
2. Пользователь вводит данные в поле «Пароль»
3. Пользователь нажимает на кнопку «Вход»
Постусловия: вход в приложение выполнен

Рисунок 6 – Спецификация варианта использования «Авторизоваться»

Выделенные варианты использования помогут упростить процесс внедрения приложения и лучше понять, какие основные функции оно должно выполнять. Они дадут ясное представление о том, какая функциональность необходима для того, чтобы решить конкретные задачи и удовлетворить потребности пользователей.

3.2 Модель предметной области

Модель предметной области «Информационная система предоставления услуг для автопутешественников» включает в себя следующие основные компоненты:

- планирование маршрута и подсчёт бюджета;
- рекомендации лучших мест;
- обратная связь о путешествии.

Модуль рекомендаций использует информацию от других модулей для предоставления рекомендаций, а модуль обратной связи позволяет делиться впечатлениями о совершенном автопутешествии.

Концептуальная модель данной предметной области представлена ниже на рисунке 7.

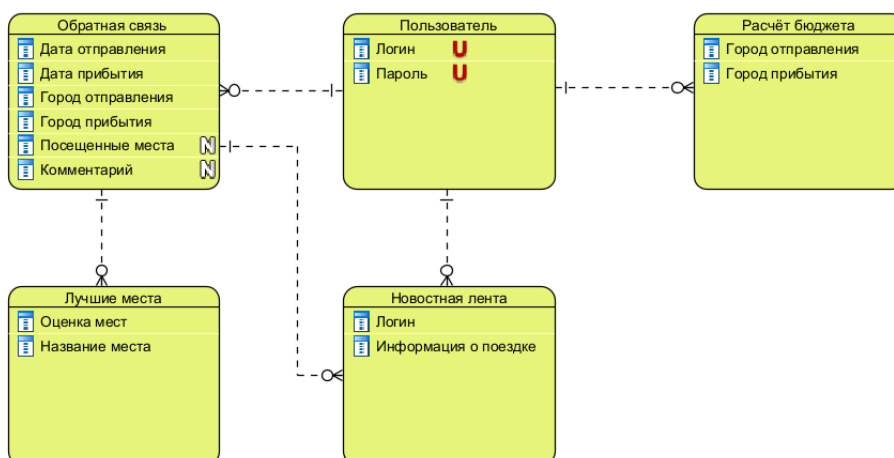


Рисунок 7 – Концептуальная модель предметной области

Для хранения данных о пользователях используется сущность «Пользователь», которая имеет 2 атрибута: логин и пароль.

Сущность «Обратная связь» использует несколько атрибутов, такие как: дата отправления, дата прибытия, город отправления, город прибытия, посещенные места и комментарий. В комментарии пользователь может рассказать о своем путешествии подробнее.

4 Проектирование

На основании анализа предметной области было выполнено проектирование системы. Было принято решение о необходимости использования клиент – серверной архитектуры приложения. Клиент – серверная архитектура позволяет разделить приложение на две части: клиентскую и серверную, которые взаимодействуют друг с другом для обеспечения функциональности приложения.

Клиентская часть приложения выполняется на устройстве пользователя, например, на компьютере или смартфоне. Она представляет собой пользовательский интерфейс приложения, через который пользователь взаимодействует с сервером.

Серверная часть приложения выполняется на удаленном сервере. Она обрабатывает запросы от клиентской части и возвращает результаты обработки. Серверная часть также отвечает за управление базой данных приложения и обеспечение безопасности работы приложения.

Определим основные требования к клиентской и серверной части приложения.

Клиентская часть должна содержать:

- регистрацию/авторизацию в приложении;
- новостную ленту;
- подсчёт бюджета;
- оценку посещенных мест;
- возможность отправки впечатлений о путешествии;
- карту.

Серверная часть системы должна содержать:

- сервис для регистрации/авторизации пользователя;
- сервис для публикаций информации;
- сервис для градации лучших, посещенных пользователями, мест;

- сервис для взаимодействия с картой.

Диаграмма компонентов, описывающая особенности физического представления системы, изображена на рисунке 8.

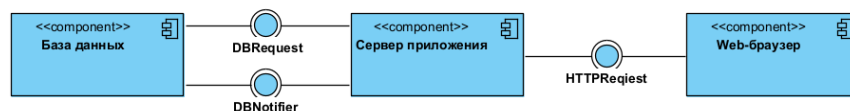


Рисунок 8 – Диаграмма компонентов клиент – серверной архитектуры

Использование клиент – серверной архитектуры позволяет обеспечить эффективную работу приложения.

4.1 Проектирование серверной части

В контексте веб – приложений на C# контроллеры (Controllers) отвечают за обработку запросов от клиента и управление исполнением приложения. Они являются средним звеном между пользовательским интерфейсом и бизнес – логикой приложения.

Контроллеры C# являются классами, которые содержат методы, называемые действиями, и связываются с маршрутами приложения. Действия обычно обрабатывают полученные запросы, собирают данные из моделей, вызывают необходимые методы и передают результаты представлениям.

Контроллеры, используемые в приложении:

- AuthenticationController – отвечает за авторизацию в приложении;
- AccountRegistrationController – отвечает за регистрацию в приложении;
- BudgetController – отвечает за расчет затрат на бензин в течение поездки. Затраты рассчитываются в зависимости от расстояния, марки автомобиля и типа двигателя;

- **FeedController** – отвечает за отображение ленты в приложении. В ленте публикуются отзывы пользователей о поездках;
- **RatingController** – отвечает за отображение списка лучших мест, оцененных пользователями;
- **ReviewController** – отвечает за написание отзыва о поездке. Отзыв может включать в себя: города отправления и прибытия, дату поездки, комментарий, посещенные места и их оценки.

На рисунке 9 представлена диаграмма классов.

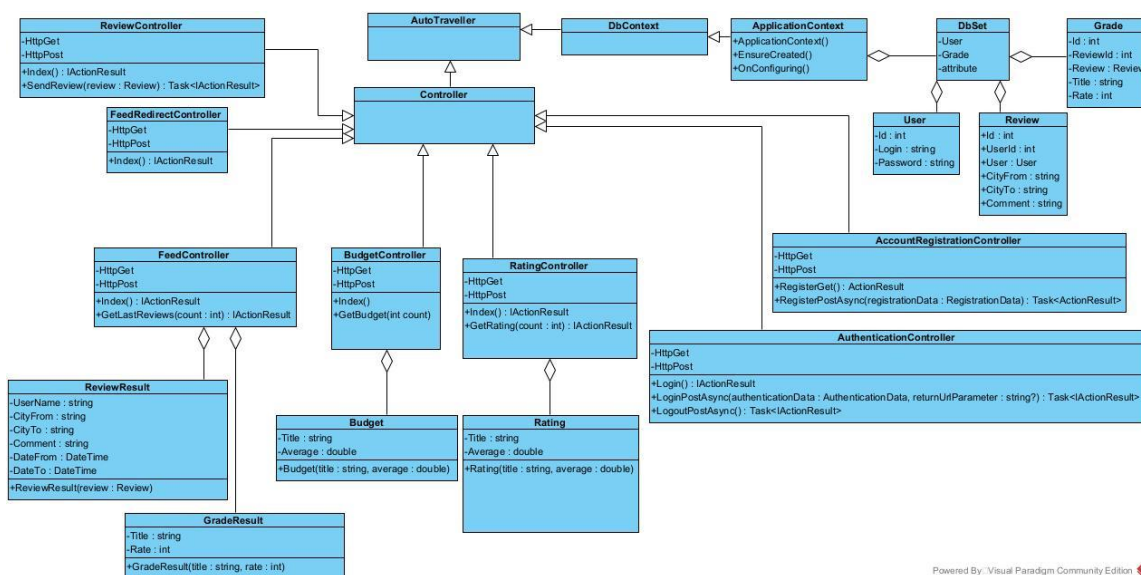


Рисунок 9 – Диаграмма классов

Класс **ReviewResult** имеет публичные свойства для хранения информации о отзыве, которая затем используется для вывода. Конструктор класса принимает объект типа **Review** и извлекает информацию из его свойств, чтобы инициализировать свойства класса **ReviewResult**. Конструктор перебирает список оценок **review.Grades** и создает новые объекты типа **GradeResult** для каждой оценки, которые затем добавляются в список **Grades** класса **ReviewResult**.

GradeResult используется для хранения и обработки результатов оценок, которые состоят из названия (**Title**) и числовой оценки (**Rate**).

Классы `ReviewResult` и `GradeResult` являются запечатанными и не могут быть наследованы или расширены.

Класс `BudgetController` используется для управления бюджетом поездки.

Класс `Budget` является запечатанным и не может быть унаследован другим классом. Он служит для создания объектов, которые содержат название бюджета и среднюю стоимость.

Класс `RatingController` используется для формирования списка лучших мест.

Метод `Index()` возвращает представление `rating.html`.

Метод `GetRating(int count)` осуществляет запрос к базе данных, считывает средний балл для каждого места и возвращает список из наиболее популярных мест с наивысшим средним баллом.

Класс `Rating` является запечатанным и не может быть унаследован, он имеет два свойства: `Title`, которое представляет название рейтинга, и `Average`, которое представляет его средний балл.

`Feedcontroller` отвечает за отображение ленты в приложении. Главной задачей `Feedcontroller` является сбор и обработка данных, которые должны быть отображены в ленте. Для реализации этой задачи `Feedcontroller` соединяется с другими компонентами.

Метод `public IActionResult Index()` возвращает HTML – страницу со структурой ленты новостей.

Метод `public IActionResult GetLastReviews(int count)` получает последние отзывы из базы данных, преобразует их в объекты класса `ReviewResult` и возвращает контент в формате JSON.

Далее рассмотрим взаимодействие с базой данных.

Класс `ApplicationContext` наследуется от `DbContext` – это класс в `Entity Framework Core`, который предоставляет доступ к базе данных.

В этом классе определены три DbSet свойства: Users, Grades и Reviews – это представления таблиц в базе данных. DbSet это коллекция объектов, которые будут использоваться в запросах к базе данных.

Метод EnsureCreated() вызывается в конструкторе ApplicationContext и проверяет, создана ли база данных. Если база данных не создана, то создается новая база данных.

Метод OnConfiguring() переопределяется, чтобы настроить опции, необходимые для взаимодействия с базой данных. В данном случае, используется SQL Server, который запущен на локальной машине (localhost), база данных называется master, и используется подключение Trusted_Connection=True для подключения к базе данных с помощью Windows – аутентификации без указания имени пользователя и пароля.

Таким образом, после создания экземпляра ApplicationContext, можно использовать DbSet и методы, предоставляемые DbContext для доступа к данным в базе данных.

4.2 Проектирование клиентской части

Для реализации клиентской части приложения были выбраны языки: JavaScript, HTML, CSS.

JavaScript – это язык программирования, который позволяет создавать динамические веб-страницы, взаимодействовать с пользователем и обрабатывать данные на стороне клиента. HTML – это язык разметки, который определяет структуру и содержание веб-страницы, тогда как CSS – это язык стилей, который определяет внешний вид и оформление веб-страницы.

JavaScript, HTML и CSS комбинируются вместе, чтобы создать удобный пользовательский интерфейс, который удовлетворяет потребности конечного пользователя.

Одним из основных преимуществ данного подхода является то, что клиентская часть приложения выполняется на стороне пользователя, что

позволяет ускорить его работу и снизить нагрузку на сервер. Кроме того, использование JavaScript, HTML и CSS позволяет разработчикам создавать универсальные интерфейсы, которые показываются на всех типах устройств, без необходимости создания отдельных версий для каждого типа устройства [11].

4.3 Проектирование пользовательского интерфейса

Пользовательский интерфейс – это совокупность элементов, при помощи которых пользователь взаимодействует с программным обеспечением, веб-сайтом или мобильным приложением. Пользовательский интерфейс непосредственно влияет на уровень удобства работы с программным продуктом или услугой, а также на время, необходимое пользователям для выполнения задач.

Пользовательский интерфейс является одним из наиболее важных компонентов любого приложения. Он обеспечивает пользователю доступ к функциональности программы и позволяет работать с ней более эффективно и удобно.

На рисунке 10 представлена модель пользовательского интерфейса главной страницы.

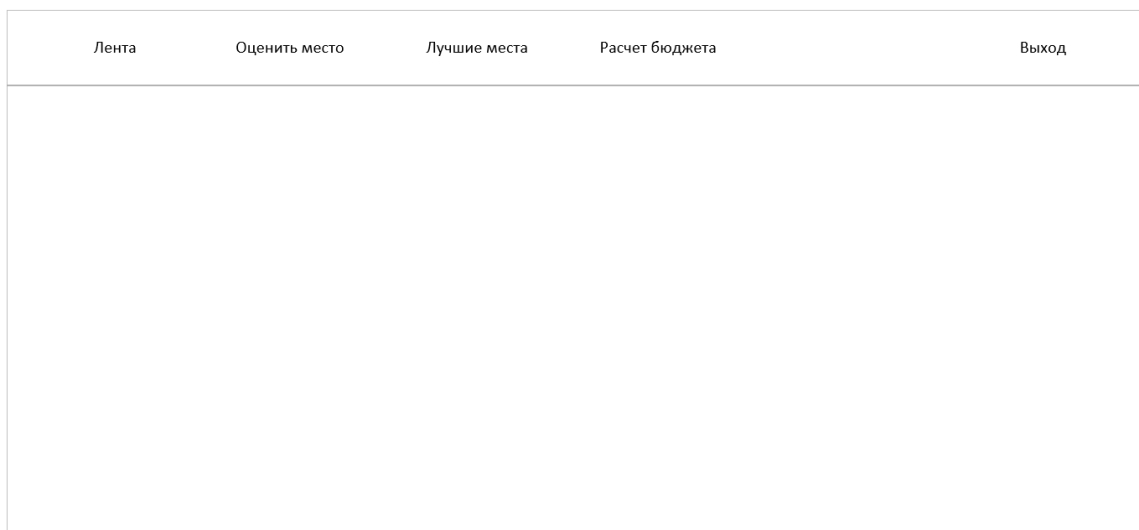


Рисунок 10 – Модель интерфейса главной страницы

Рассмотрим подробнее каждый из разделов:

- раздел «Лента» будет включать в себя записи, опубликованные пользователями;
- раздел «Оценить место» необходим для публикации отзывов пользователей;
- в разделе «Лучшие места» будут показаны посещенные места, которые пользователи оценили выше всего;
- раздел «Расчет бюджета» будет включать в себя отображение маршрута на карте с возможностью подсчета стоимости бензина, затрачиваемого на поездку.

Каждый из разделов будет доступен всем зарегистрированным пользователям. При нажатии кнопки «Выход» будет выполнен переход на страницу входа в приложение. Для регистрации необходимо будет ввести логин и пароль.

5 Реализация

5.1 Реализация серверной части приложения

Информационная система предоставления услуг для автопутешественников включает в себя группу классов, которые хранятся в отдельных файлах с расширением cs. Данные файлы используются для сборки исполняемого.

Таким образом, серверная часть приложения была разбита на 15 модулей, которые приведены на диаграмме компонентов (рисунок 11). Исходный код компонентов серверной части приложения приведен в приложении В.

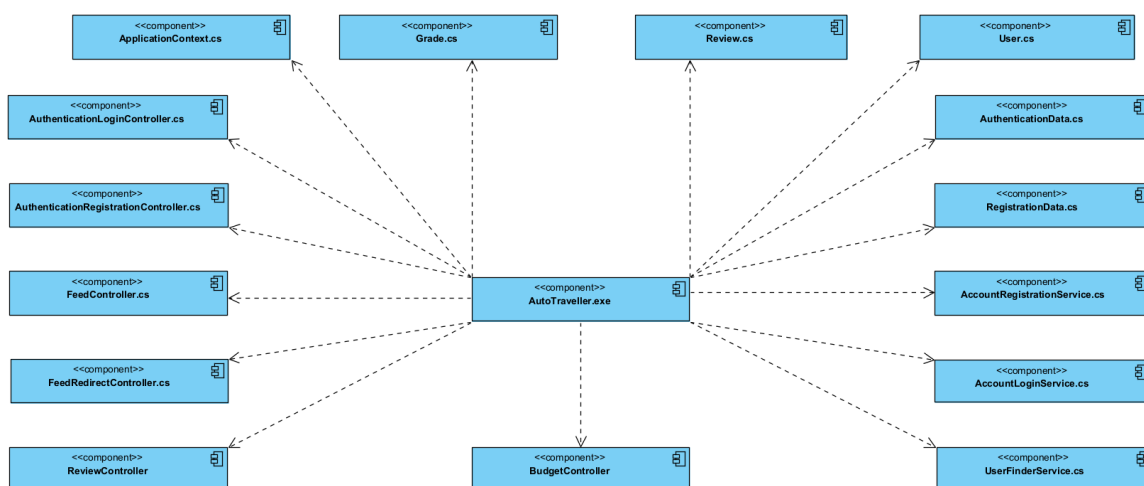


Рисунок 11 – Диаграмма компонентов

В результате сборки проекта получится конечный файл program.cs. Данный файл включает в себя скомпилированные файлы-классы с расширением class. Файл program.cs помещается в контейнер, из-под которого происходит его исполнение. Исходный код серверной части представлен в приложении Б.

5.2 Реализация клиентской части приложения

Клиентская часть приложения реализована на языках JavaScript, HTML, CSS и имеет следующую структуру:

- HTML – страницы, отвечающие за отображение данных на странице;
- CSS – стили, задающие внешний вид страницы;
- JavaScript – функции, отвечающие за обработку событий на странице и взаимодействие с сервером.

Пример кода реализации клиентской части представлен ниже в листинге 1.

Листинг 1 – Код реализации списка отзывов

```
renderReviews();

async function renderReviews()
{
    let reviews = await sendRequest('', 'feed?count=5');
    if(reviews.length === 0)
    {
        const element = document.getElementById('empty-feed-
message');
        element.hidden = false;
        return;
    }
    for(let i = 0; i < reviews.length; i++)
    {
        const review = reviews[i];
        renderReview(review);
    }
}

function renderReview(review)
{
    let divOrderSpace = document.getElementById('reviews-
content');
```

```

    let divStoredOrder = document.createElement('div');

    const userName = review.UserName;
    const cityFrom = review.CityFrom;
    let dateFrom = review.DateFrom.substring(0, 10).replaceAll('-', ' ');
    let cityTo = review.CityTo;
    let dateTo = review.DateTo.substring(0, 10).replaceAll('-', ' ');
    let comment = review.Comment;
    let grades = review.Grades;
    divStoredOrder.className = 'review';

    let travelInformation = `

|                            |
|----------------------------|
| Пользователь: \${userName} |
|----------------------------|



|                            |                   |                           |                   |
|----------------------------|-------------------|---------------------------|-------------------|
| <p>Город отправления: </p> | \${cityFrom}      | <p>Дата отправления: </p> | \${dateFrom}      |
| <p>Город прибытия: </p>    | <p>\${cityTo}</p> | <p>Дата прибытия: </p>    | <p>\${dateTo}</p> |

`

    let visitedPlaces = getVisitedPlaces(grades);
    let commentInfo = `

Комментарий:



${comment}

`;

    divStoredOrder.innerHTML =
` ${travelInformation} ${visitedPlaces} ${commentInfo} `
    divOrderSpace.append(divStoredOrder)
}

function getVisitedPlaces(grades)
{
    let result = '<table class="places"><tr><td>Посещенное
место</td><td><p>Оценка</p></td></tr>'

    if(grades.length === 0) return '';
    for(let i = 0; i < grades.length; i++)
    {
        const grade = grades[i];

```

```

        console.log(grade)

        result +=
`<tr><td>${grade.Title}</td><td><p>${grade.Rate}</p></td></tr>`;

    }

    result += `</table>`

    return result;
}

async function sendRequest(json, uri) {
    const myHeaders = new Headers()
    myHeaders.append('Content-Type', 'application/json')
    const request = new Request(uri, {
        method: 'POST',
        body: json,
        headers:myHeaders
    });

    let search_result = fetch(request)
        .then((response) => {
            return response.json()
        })

    return search_result;
}

async function logout()
{
    await sendRequest('', '/auth/logout');
}

```

Основной функцией является `renderReviews()`, которая получает отзывы из сервера, проверяет, не является ли список пустым, и в случае если список

пуст – отображает сообщение на странице. Если список не пустой, то для каждого отзыва вызывается функция `renderReview()`, которая формирует HTML – элементы для отображения отзыва и добавляет их на страницу.

Функция `renderReview()` формирует отзыв на основе полученной информации и заполняет данными созданный элемент `divStoredOrder`. Элемент выглядит как блок данных, состоящий из информации о пользователе, городах отправления и прибытия, даты выезда и возвращения, списка посещенных мест и комментария о поездке.

Функция `getVisitedPlaces()` формирует HTML – элемент – таблицу с данными о посещенных местах и оценками, и возвращает сформированный элемент в виде строки HTML – кода.

Функция `sendRequest()` реализует запрос к серверу и используется для получения списка отзывов.

Функция `logout()` реализует выход пользователя из системы.

Исходный код клиентской части представлен в приложении В.

5.3 Реализация пользовательского интерфейса

Для обеспечения надлежащего взаимодействия с системой был создан пользовательский интерфейс, соответствующий заданным требованиям в разделе проектирования. Он включает в себя несколько основных вкладок, в том числе: ленту, вкладку оценки путешествия, список лучших мест и вкладку расчета бюджета.

На рисунке 12 представлена форма авторизации пользователя. В ней доступны поля ввода логина и пароля. В случае, если пользователь не был зарегистрирован в системе, следует нажать кнопку «Нет аккаунта? Регистрация».

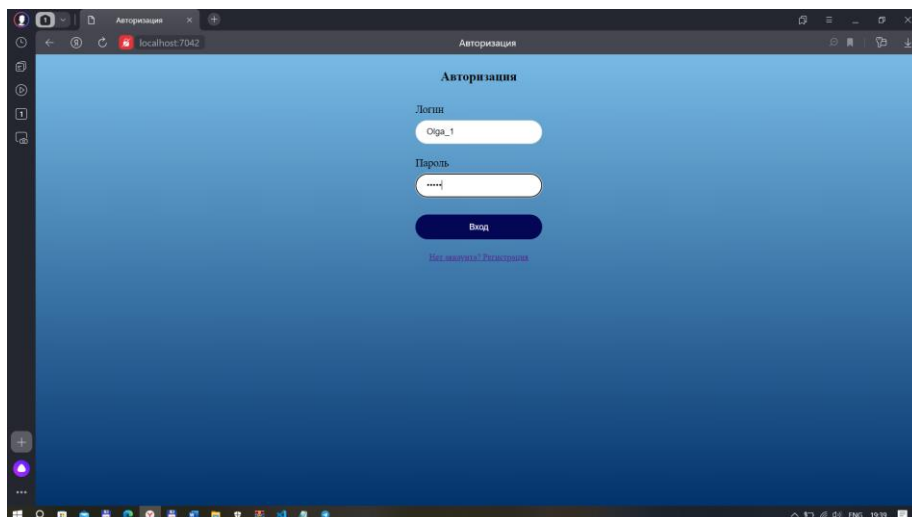


Рисунок 12 – Форма регистрации

На главной странице приложения представлена вкладка «Оценить место» (рисунок 13). В ней доступно заполнение полей, которые позволяют пользователям оставлять оценки и комментарии о местах.

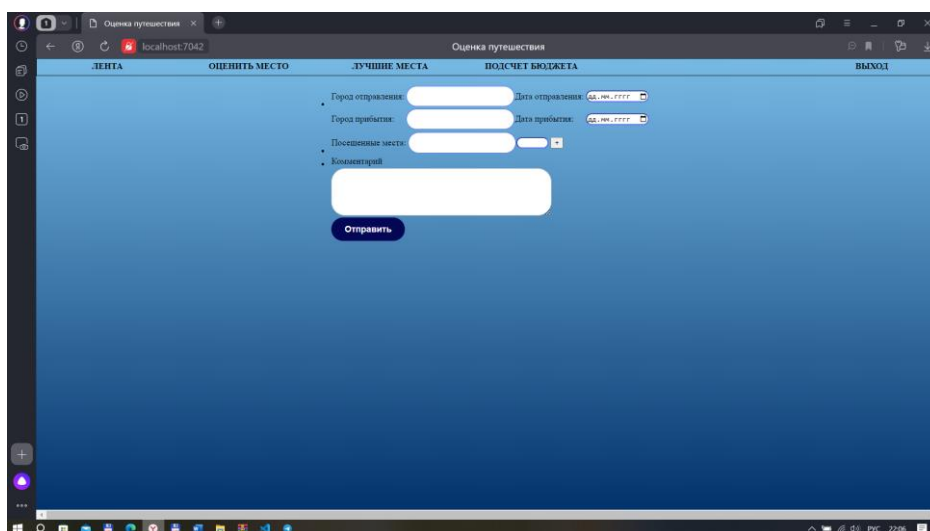


Рисунок 13 – Вкладка «Оценить место»

Далее рассмотрим вкладку «Лента» (рисунок 14). В ней доступен просмотр и чтение оценочных комментариев пользователей. На этой вкладке пользователи могут читать оценки и отзывы о путешествиях, которые оставили другие пользователи.

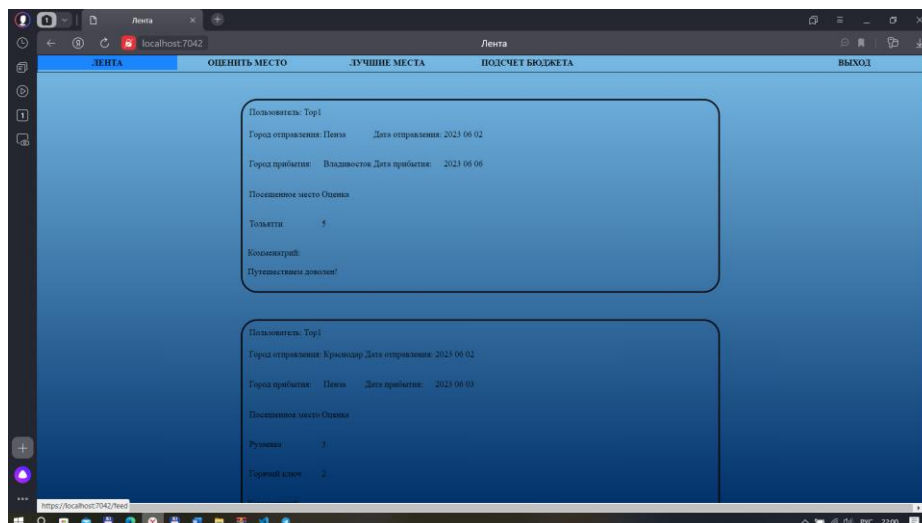


Рисунок 14 – Вкладка «Лента»

Затем рассмотрим вкладку «Лучшие места» (рисунок 15). Рейтинги на этой вкладке могут помочь пользователям выбрать лучшие места для посещения на основе опыта других людей.

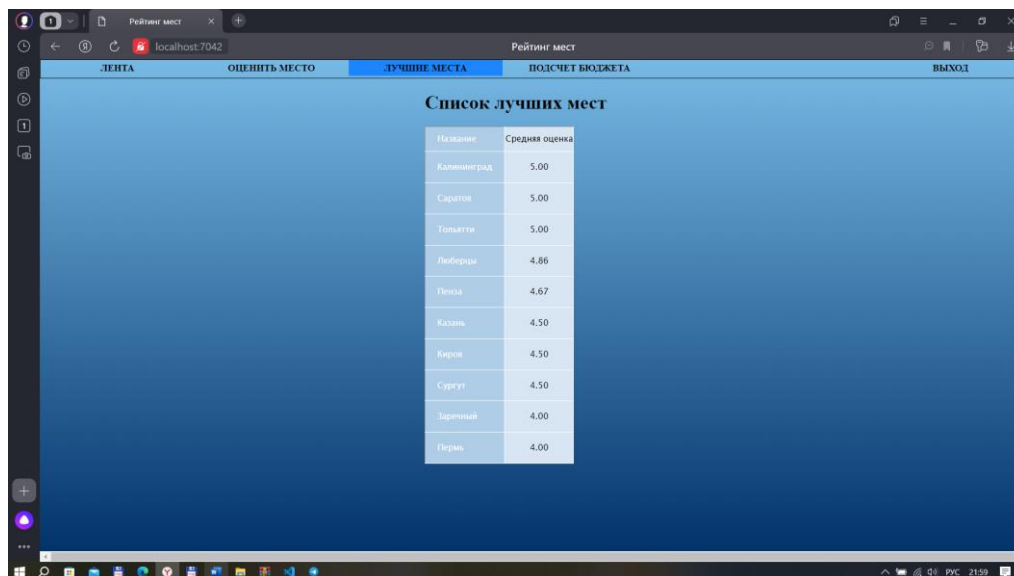


Рисунок 15 – Вкладка «Лучшие места»

Вкладка «Подсчет бюджета» (рисунок 16) необходима для расчета пользователями цены на бензин в зависимости от расстояния.

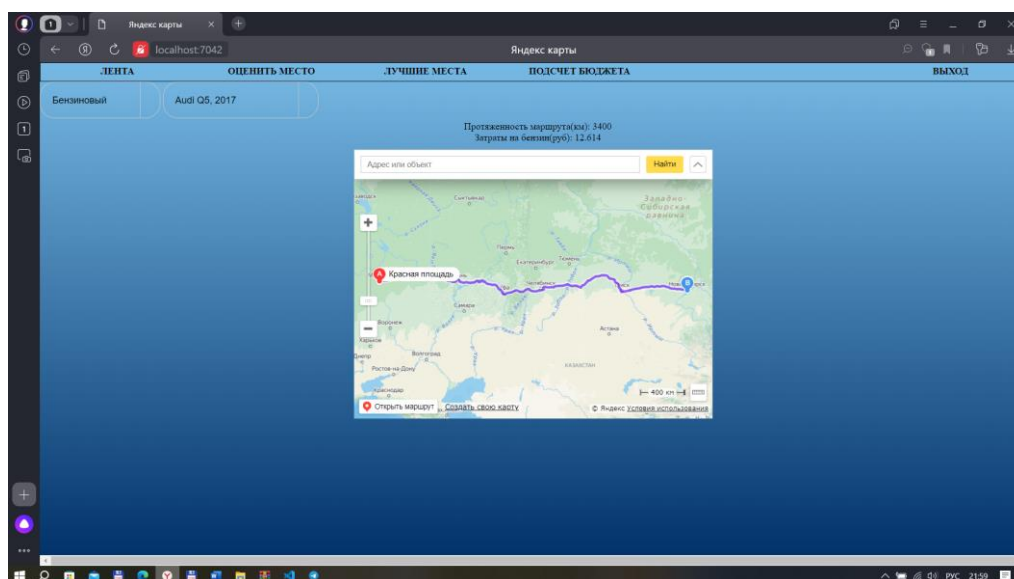


Рисунок 16 – Вкладка «Подсчет бюджета»

При нажатии кнопки «Открыть маршрут» можно просмотреть маршрут более подробно.

6 Тестирование информационной системы автопутешествий

6.1 Выбор методологии тестирования

Для эффективного тестирования программного обеспечения важно выбрать подходящую методологию тестирования. Каждая методология имеет свои преимущества и недостатки. Выбор методологии тестирования зависит от конкретной ситуации, характеристик проекта и требований к продукту.

Режимы тестирования можно разделить на две основные категории:

- функциональное тестирование;
- нефункциональное тестирование.

Основное назначение функционального режима тестирования – проверка функциональных возможностей программного продукта. Это включает в себя тестирование всех функций и операций, которые должны выполняться продуктом в соответствии с его требованиями и спецификацией.

Нефункциональное тестирование фокусируется на проверке нефункциональных аспектов продукта, таких как производительность, безопасность, совместимость и т. д. Нефункциональное тестирование используется для оценки качества продукта по таким показателям, как время отклика, нагрузочная способность, скорость работы и другие.

Для данного проекта была выбрана методология функционального тестирования. Эта методология основана на тестировании функциональности программного обеспечения и проверке его соответствия требованиям. Она наиболее эффективна в случае, когда требования к продукту хорошо структурированы и четко определены.

Функциональное тестирование – это методология тестирования программного обеспечения, которая проверяет соответствие продукта заданным функциональным требованиям.

Существуют различные методы и подходы к функциональному тестированию, такие как:

- классический подход – тестирование каждой функциональности отдельно;
- метод «черного ящика» – тестирование программного продукта без предварительного исследования исходного кода;
- автоматизированное тестирование – использование специальных инструментов для выполнения тестов;
- метод «белого ящика» – тестирование внутренней логики и исходного кода программы.

Для проведения функционального тестирования необходимо определить тест – кейсы, которые покрывают основные функциональные возможности системы. Тест – кейсы должны охватывать как основные, так и редкие сценарии использования системы.

Основными целями функционального тестирования являются:

- проверка работоспособности системы;
- проверка соответствия работающей системы ее описанию.

Преимущества функционального тестирования заключаются в следующем:

- проверка соответствия требованиям и ожиданиям пользователей;
- выявление ошибок в работе продукта;
- повышение качества продукта и уменьшение риска ошибок после выпуска;
- оптимизация процесса тестирования и уменьшение времени на разработку тестовых случаев;
- легкость автоматизации функциональных тестов.

Эти преимущества делают функциональное тестирование наиболее эффективным подходом для тестирования данного проекта.

В данной работе было использовано функциональное тестирование программы чёрным ящиком, потому что оно позволяет своевременно выявить

системные ошибки ПО и, тем самым, избежать множества проблем при работе с ним в дальнейшем [12].

6.2 Тестирование чёрным ящиком

В следующих примерах будет представлено тестирование логина при входе на сайт. Логин должен содержать 4 – 16 символов: цифры, нижнее подчеркивание, латинские буквы. Должен начинаться с буквы.

Для тестирования были выделены правильные и неправильные классы эквивалентности (КЭ), представленные в таблице 8.

Таблица 8 – Классы эквивалентности

	Правильные КЭ	Неправильные КЭ
Длина имени	4-16 (1)	0 – 4 (2) 16 – ∞ (3)
Структура	Начинается с буквы(4)	Начинается не с буквы (5)
Содержание	Буквы Латинские (6) Цифры(7) Нижнее подчеркивание (8)	Остальное (9)

Изображение классов представлено на рисунке 17.

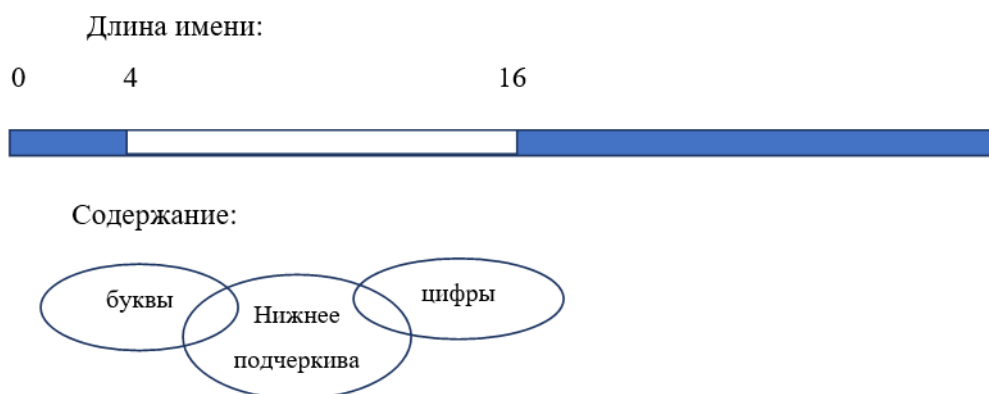


Рисунок 17 – Изображение классов

Список тестов, а также их результатов представлен в таблице 9.

Таблица 9 – Тестирование логина

Тип теста	Описание этапов	Результат
Прохождение успешного сценария по умолчанию	Проверка длины и состава логина	Тест пройден – рисунок 1
Негативное тестирование – неверная длина входных данных	Проверка ввода некорректной длины логина	Тест пройден – рисунок 2
Негативное тестирование – неверные входные данные	Проверка ввода неправильного типа данных	Тест пройден – рисунок 3

Таким образом, для информационной системы были проведены все тесты из вышеперечисленных. Рассмотрим их результаты.

Во – первых, был проведен основной позитивный тест (прохождение успешного сценария по умолчанию). Как видно из рисунка ниже, вход был осуществлён.

Результат прохождения теста представлен на рисунке 18.



Рисунок 18 – Результат прохождения позитивного теста

Во – вторых, было произведено негативное тестирование на неверной длине данных. Как видно из рисунка ниже, при попытке входа выдаётся ошибка о некорректном вводе длины логина.

Результат прохождения теста представлен на рисунке 19.

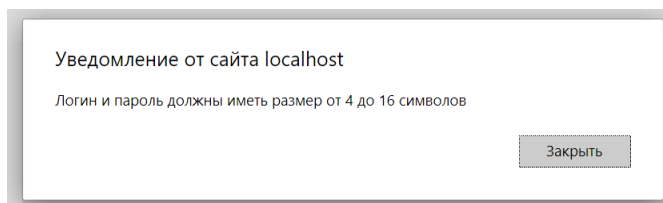


Рисунок 19 – Результат прохождения негативного теста

По окончании было осуществлено негативное тестирование – неверные входные данные:

Результат прохождения теста представлен на рисунке 20.

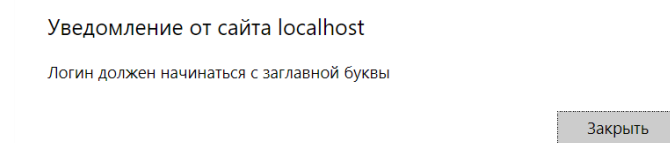


Рисунок 20 – Результат прохождения второго негативного теста

Таким образом, на основе результатов тестирования чёрным ящиком, можно сделать вывод, что приложение корректно обрабатывает входящие данные, отправляет информацию в ответ на корректные входные данные, в случае ошибки приложение корректно её обрабатывает, отсылая клиенту соответствующий текст об ошибке.

6.3 Расчет числовых метрик программы

Для анализа работы программы, производится расчет метрик кода. Существует несколько различных метрик. Для тестирования метода были использованы метрика Холстеда и цикломатическая сложность.

Рассмотрим функцию, на основе которой будет производиться расчет метрик. В качестве примера были выбраны методы `loginvalidation` и

validateForm. Используется для задания ограничений логина и пароля. Код метода представлен в листинге 2.

Листинг 2 – Код метода задания ограничений логина и пароля

```
const urlParams = new URLSearchParams(window.location.search);
const registerParam = urlParams.get('login');
console.log(registerParam)
if(registerParam === 'wrong-data')
{
    document.getElementById("wrong-data-error").hidden = false
}
function validateForm() {
    let loginElement = document.getElementById('log');
    let login = loginElement.value;
    let passwordElement = document.getElementById('pas');
    let password = passwordElement.value;

    // Проверяем наличие нижнего подчеркивания в логине
    if (!login.includes('_')) {
        alert("Логин должен содержать нижнее подчеркивание");
        return false;
    }

    // Проверяем наличие нижнего подчеркивания в пароле
    if (!password.includes('_')) {
        alert("Пароль должен содержать нижнее подчеркивание");
        return false;
    }

    // Проверяем длину логина и пароля
    if ((login.length < 4 || login.length > 16) && (password.length < 4 || password.length > 16)) {
```

```

    alert("Логин и пароль должны иметь размер от 4 до 16 символов");
    return false;
}

// Проверяем длину логина
if (login.length < 4 || login.length > 16) {
    alert("Логин должен иметь размер от 4 до 16 символов");
    return false;
}

// Проверяем, начинается ли логин с заглавной буквы
if (!/^[A-Z]/.test(login)) {
    alert("Логин должен начинаться с заглавной буквы");
    return false;
}

// Проверяем длину пароля
if (password.length < 4 || password.length > 16) {
    alert("Пароль должен иметь размер от 4 до 16 символов");
    return false;
}
}

function sendRequest(json, uri) {
    const myHeaders = new Headers()
    myHeaders.append('Content-Type', 'application/json')
    const request = new Request(uri, {
        method: 'POST',
        body: json,
        headers:myHeaders
    });

    let search_result = fetch(request)

```

```

        .then((response) => {
            return response.json();
        })

    return search_result;
}

```

Таким образом, на основе данного кода представляется возможным расчет указанных выше метрик.

6.3.1 Метрика Холстеда

Метрика Холстеда относится к метрикам, рассчитываемым на основе анализа количества строк и синтаксических элементов исходного кода программы. Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

- число уникальных операторов, включая символы – разделители, имена процедур, функций или методов и знаки операций;
- число уникальных операндов;
- общее число операторов;
- общее число операндов.

Операторы и операнды приведены в таблицах 10, 11 соответственно.

Таблица 10 – Операторы

Операторы	Число вхождений
=	10
===	1
<	4
>	4
If	7

Продолжение таблицы 10

&&	1
return	8
	4
=>	1
;	21
,	5

Таким образом, в методе используется 11 операторов, суммарное число вхождений которых 66.

Таблица 11 – Операнды

Операнды	Число вхождений
alert	8
fetch	1
search_result	2
password	2
urlParams	2
registerParam	3
document.getElementById	3
login	3
loginElement	2
passwordElement	2
login.length	4

Продолжение таблицы 11

password.length	4
-----------------	---

Таким образом, в методе используется 12 операндов, суммарное вхождение которых является 33.

Длина программы (N) вычисляется по следующей формуле:

$$N = N_1 + N_2,$$

где N_1 – общее число вхождений операторов;

N_2 – общее число вхождений операндов.

Для данного метода длина программы равна 99.

Словарь программы (n) рассчитывается по формуле:

$$n = n_1 + n_2,$$

где n_1 – число уникальных операторов;

n_2 – число уникальных операндов.

Для данного метода словарь равен 23.

Объем программы вычисляется по формуле:

$$HV = N \log_2 n.$$

Для данного метода объем программы равен 447,876.

Сложность программы вычисляется по формуле:

$$HD = \left(\frac{n_1}{2}\right) \cdot \left(\frac{N_2}{n_2}\right).$$

Для данного метода сложность равна 15,125, что означает среднюю сложность программы.

6.3.2 Цикломатическая сложность программы

Цикломатическая сложность программы – это число независимых путей исполнения кода в ее структуре управления потоком.

Цикломатическая сложность является важным параметром для оценки сложности программы, а также для планирования тестирования программного обеспечения.

Граф потока управления программы для функции `validateForm` приведен на рисунке 21.

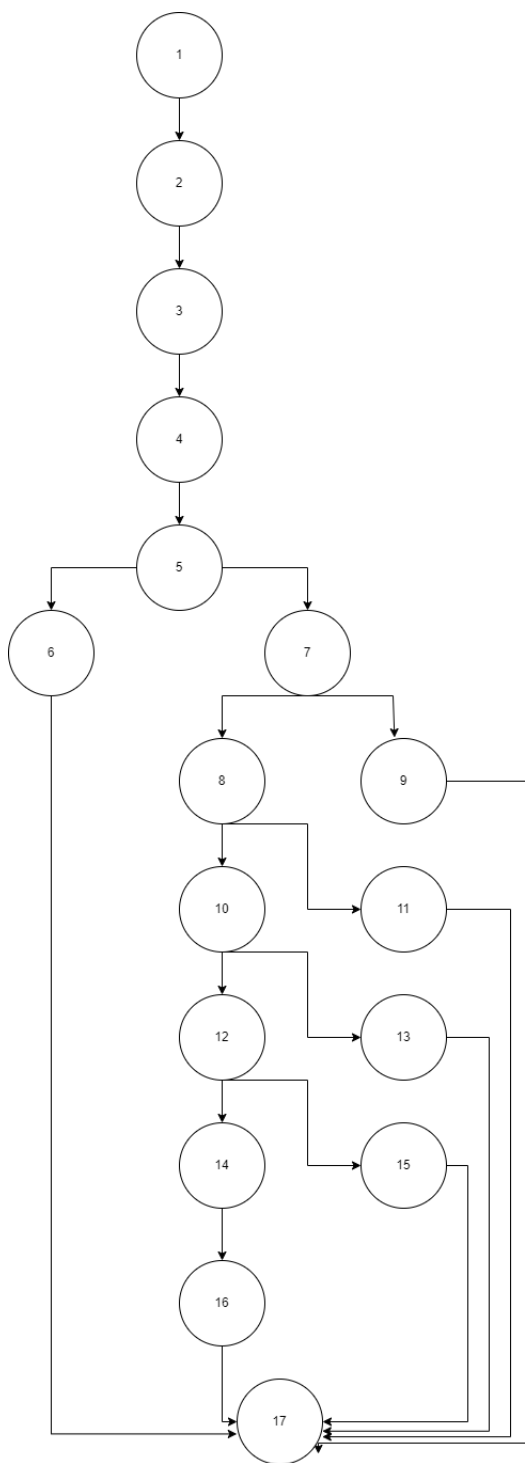


Рисунок 21 – Граф потока управления программы

Для вычисления цикломатической сложности $V(G)$ применяется следующая формула:

$$v(G) = e - n + 2p,$$

где e – количество дуг, n – количество вершин, p – число компонент связности.

Граф содержит 17 вершин, 18 дуг и один компонент связности.

$$v(G) = 17 - 18 + 2.$$

При оценке сложности программы действует правило: если цикломатическое число больше 10, то программа обладает излишней сложностью и её следует разбить на составные части с меньшим значением цикломатического числа.

Таким образом значение цикломатической сложности равно 3, что означает корректную и стабильную работу программы.

Заключение

В результате выполнения выпускной квалификационной работы была разработана информационная система предоставления услуг для автопутешественников. Для ее разработки были использованы современные языки программирования, такие как C#, JS, CSS и HTML.

Была проведена аналитическая работа, которая определила основные требования и функции системы. Также было произведено сравнение продуктов – аналогов определившие необходимые функции, которые должно выполнять приложение.

Были составлены функциональные требования, архитектура приложения и структура данных. Для проектирования использовалась методология UML.

В процессе выполнения работы было проведено тестирование, и расчет метрик кода, показавшее невысокие показатели сложности кода, что упрощает его последующий анализ.

Был проведен расчет затрат разработки, показавший размер и долю затрат при создании приложения.

Таким образом, в результате выполнения выпускной квалификационной работы была разработана информационная система предоставления услуг для автопутешественников в соответствии с техническим заданием. В результате разработки системы были достигнуты поставленные цели и решены задачи проектирования и реализации программного обеспечения.

Список использованных источников

1. Романова В. Влияние современных технологий на нашу жизнь // Urok [Электронный ресурс] URL: <https://urok.1sept.ru/articles/686920> (дата обращения: 18.05.2023).
2. Plusiminusi // Plusiminusi [Электронный ресурс] URL: <https://plusiminusi.ru/plyusy-i-minusy-puteshestviya-na-mashine-avtomobile/> (дата обращения: 20.05.2023).
3. Arrivo, Яндекс Путешествия: что такое, плюсы и минусы // Arrivo [Электронный ресурс] URL: <https://www.arrivo.ru/partners/2023-03/yandeks-puteshestviya-cto-takoe-plyusy-i-minusy-kak-podklyuchit.html> (дата обращения: 25.05.2023).
4. WheelNews, Лучшие приложения для путешествий по России на автомобиле // WheelNews [Электронный ресурс] URL: <https://wheelnews.ru/luchshie-prilozheniya-dlya-puteshestviy-po-rossii-na-avtomobile/> (дата обращения: 25.05.2023).
5. Кухноцев П. Разбираем диаграмму Ганта // Skillbox Media [Электронный ресурс] URL: <https://skillbox.ru/media/management/razbiraem-diagrammu-ganta-instrument-kotoryy-dolzhen-znat-kazhdyy-menedzher/> (дата обращения: 25.05.2023).
6. Нижельская М. Что такое UML // Yandex.practicum [Электронный ресурс] URL: <https://practicum.yandex.ru/blog/uml-diagrammy/> (дата обращения: 25.05.2023).
7. Нижельская М. Что такое ER – диаграмма // Yandex.practicum [Электронный ресурс] URL: <https://practicum.yandex.ru/blog/cto-takoe-er-diagramma/> (дата обращения: 25.05.2023).
8. Metanit, Введение в C# // Metanit [Электронный ресурс] URL: <https://metanit.com/sharp/tutorial/1.1.php> (дата обращения: 26.05.2023).

9. Лок, Э. ASP.NET Core в действии / Э. Лок. –пер. с англ. – М.: ДМК Пресс, 2021. – 906 с.
10. Некрасов С. Главный язык программирования для фронтендера: зачем учить JavaScript // Yandex.practicum [Электронный ресурс] URL: <https://practicum.yandex.ru/blog/что-такое-javascript-zachem-nuzhen/> (дата обращения: 26.05.2023).
11. Сытник Э. Строим свой full-stack на JavaScript: Клиент // Хабр [Электронный ресурс] URL: <https://habr.com/ru/articles/310788/> (дата обращения: 27.05.2023).
12. Савастюк С. Методологии тестирования ПО. Какую выбрать? // ИксБи Софтваре Блог [Электронный ресурс] URL: <https://xbsoftware.ru/blog/metodologii-testirovaniya-po-kakuyu-vybrat/> (дата обращения: 27.05.2023).

ПРИЛОЖЕНИЕ А
ГЛОССАРИЙ
(справочное)

СУБД – система управления базами данных.

JavaScript – мультипарадигменный язык программирования.

C# – современный объектно-ориентированный и типобезопасный язык программирования.

REST API – это прикладной программный интерфейс (API), который использует HTTP-запросы для получения, извлечения, размещения и удаления данных.

ASP.NET – бесплатная интернет-платформа для создания веб-сайтов и веб-приложений с помощью HTML, CSS и JavaScript.

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

Листинг Б.1 – Файл AuthenticationController.cs

```

using AutoTraveller.ActionResults;
using AutoTraveller.Entities;
using AutoTraveller.Services;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;

namespace AutoTraveller.Controllers;

[Route("/auth")]
[ApiController]
public class AuthenticationController : Controller
{
    private const string ReturnUrlParameterQueryKey = "ReturnUrl";
    private readonly Func<string, HtmlResult> _htmlResultFactory;
    private readonly AccountLoginService _accountLoginService;

    public AuthenticationController(
        Func<string, HtmlResult> htmlResultFactory,
        AccountLoginService accountLoginService)
    {
        _htmlResultFactory = htmlResultFactory;
        _accountLoginService = accountLoginService;
    }

    [Route("login")]
    [HttpGet]
    public IActionResult Login()
    {
        return _htmlResultFactory(@"../wwwroot/pagestructure/login.html");
    }

    [HttpPost]
    [Route("login")]

```

```

public async Task<IActionResult> LoginPostAsync(
    [FromForm] AuthenticationData authenticationData,
    [FromQuery(Name=ReturnUrlParameterQueryKey)] string? returnUrlParameter)
{
    var claimsPrincipal = await
_accountLoginService.LoginAsync(authenticationData);
    if (claimsPrincipal == null)
    {
        return Redirect("/auth/login?login=wrong-data");
    }

    await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
claimsPrincipal);
    return Redirect(returnUrlParameter ?? "/");
}

[Route("logout")]
[HttpPost]
public async Task<IActionResult> LogoutPostAsync()
{
    await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
    return Redirect("/auth/login");
}
}

```

Листинг Б.2 – Файл AccountRegistrationController.cs

```

using AutoTraveller.ActionResults;
using AutoTraveller.Entities;
using AutoTraveller.Services;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;

namespace OnlineStore.Controllers.Account;

[Route("/auth/register")]

```

```

[ApiController]
public class AccountRegistrationController : Controller
{
    private readonly AccountRegistrationService _accountRegistrationService;
    private readonly Func<string, HtmlResult> _htmlResultFactory;

    public AccountRegistrationController(
        AccountRegistrationService accountRegistrationService,
        Func<string, HtmlResult> htmlResultFactory)
    {
        _accountRegistrationService = accountRegistrationService;
        _htmlResultFactory = htmlResultFactory;
    }

    [HttpGet]
    public ActionResult RegisterGet()
    {
        return _htmlResultFactory(@"./wwwroot/pagestructure/registration.html");
    }

    [HttpPost]
    public async Task<ActionResult> RegisterPostAsync([FromForm] RegistrationData
registrationData)
    {
        if (await
_accountRegistrationService.IsAccountAlreadyExistsAsync(registrationData.Login))
        {
            return Redirect("/auth/register?register=already-exists");
        }

        var claimsPrincipal = await
_accountRegistrationService.RegisterAccountAsync(registrationData);

        await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
claimsPrincipal);

        return Redirect("/");
    }
}

```

Листинг Б.3 – Файл BudgetController.cs

```

using System.Data.Common;
using AutoTraveller.ActionResults;
using AutoTraveller.Data;
using AutoTraveller.Services;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;

namespace AutoTraveller.Controllers;

[Route("budget")]
[ApiController]
[Authorize]
public class BudgetController : Controller
{
    private readonly ApplicationContext _applicationContext;
    private readonly Func<string, HtmlResult> _htmlResultFactory;
    private readonly UserFinderService _userFinderService;

    public BudgetController(
        ApplicationContext applicationContext,
        Func<string, HtmlResult> htmlResultFactory,
        UserFinderService userFinderService)
    {
        _applicationContext = applicationContext;
        _htmlResultFactory = htmlResultFactory;
        _userFinderService = userFinderService;
    }

    [HttpGet]
    public IActionResult Index()
    {
        return _htmlResultFactory(@"../wwwroot/pagestructure/budget.html");
    }
}

```

```

[HttpPost]
public IActionResult GetBudget(int count)
{
    var result = _applicationContext.Grades.GroupBy(grade =>
grade.Title).Select(group => new Budget(group.Key, group.Average(g => g.Rate)));

    var json =
JsonConvert.SerializeObject(result.ToList().OrderByDescending(rating =>
rating.Average).Take(count));

    return new ContentResult() { Content = json, ContentType = "application/json"
};
}

internal sealed class Budget
{
    public Budget(string title, double average)
    {
        Title = title;
        Average = average;
    }

    public string Title {get; set;}
    public double Average {get; set;}
}

```

Листинг Б.4 – Файл FeedController.cs

```

using AutoTraveller.ActionResults;
using AutoTraveller.Data;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Newtonsoft.Json;

namespace AutoTraveller.Controllers;

```

```

[Route("feed")]
[ApiController]
[Authorize]
public class FeedController : Controller
{
    private readonly ApplicationContext _applicationContext;
    private readonly Func<string, HtmlResult> _htmlResultFactory;

    public FeedController(
        ApplicationContext applicationContext,
        Func<string, HtmlResult> htmlResultFactory)
    {
        _applicationContext = applicationContext;
        _htmlResultFactory = htmlResultFactory;
    }

    [HttpGet]
    public IActionResult Index()
    {
        return _htmlResultFactory(@"./wwwroot/pagestructure/feed.html");
    }

    [HttpPost]
    public IActionResult GetLastReviews(int count)
    {
        var reviews = _applicationContext.Reviews
            .Include(review => review.User)
            .Include(review => review.Grades)
            .OrderByDescending(x => x.Id)
            .Take(count)
            .ToList();

        var reviewResults = reviews
            .Select(review => new ReviewResult(review))
            .ToList();

        var json = JsonConvert.SerializeObject(reviewResults);
    }
}

```

```

        return new ContentResult() { Content = json, ContentType = "application/json"
    };
    }
}

internal sealed class ReviewResult
{
    public ReviewResult(Review review)
    {
        UserName = review.User!.Login;
        CityFrom = review.CityFrom;
        DateFrom = review.DateFrom;
        CityTo = review.CityTo;
        DateTo = review.DateTo;
        Comment = review.Comment;
        Grades = review.Grades
            .Select(grade => new GradeResult(grade.Title, grade.Rate))
            .ToList();
    }

    public string UserName { get; set; }

    public string CityFrom { get; set; }
    public DateTime DateFrom { get; set; }
    public string CityTo { get; set; }
    public DateTime DateTo { get; set; }
    public string Comment { get; set; }

    public List<GradeResult> Grades { get; set; }
}

internal sealed class GradeResult
{
    public GradeResult(string title, int rate)
    {
        Title = title;
    }
}

```

```

        Rate = rate;
    }

    public string Title { get; set; }
    public int Rate { get; set; }
}

```

Листинг Б.5 – Файл FeedRedirectController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace AutoTraveller.Controllers;

[Route("")]
[Authorize]
[ApiController]
public class FeedRedirectController : Controller
{
    private const string FeedRouteTemplate = "/feed";

    [HttpGet]
    public IActionResult Index()
    {
        return Redirect(FeedRouteTemplate);
    }
}

```

Листинг Б.6 – Файл RatingController.cs

```

using System.Data.Common;
using AutoTraveller.ActionResults;
using AutoTraveller.Data;
using AutoTraveller.Services;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;

```

```

namespace AutoTraveller.Controllers;

[Route("rating")]
[ApiController]
[Authorize]
public class RatingController : Controller
{
    private readonly ApplicationContext _applicationContext;
    private readonly Func<string, HtmlResult> _htmlResultFactory;
    private readonly UserFinderService _userFinderService;

    public RatingController(
        ApplicationContext applicationContext,
        Func<string, HtmlResult> htmlResultFactory,
        UserFinderService userFinderService)
    {
        _applicationContext = applicationContext;
        _htmlResultFactory = htmlResultFactory;
        _userFinderService = userFinderService;
    }

    [HttpGet]
    public IActionResult Index()
    {
        return _htmlResultFactory(@"./wwwroot/pagestructure/rating.html");
    }

    [HttpPost]
    public IActionResult GetRating(int count)
    {
        var result = _applicationContext.Grades.GroupBy(grade =>
grade.Title).Select(group => new Rating(group.Key, group.Average(g => g.Rate)));

        var json =
JsonConvert.SerializeObject(result.ToList().OrderByDescending(rating =>
rating.Average).Take(count));

```

```

        return new ContentResult() { Content = json, ContentType = "application/json"
    };
    }
}

internal sealed class Rating
{
    public Rating(string title, double average)
    {
        Title = title;
        Average = average;
    }

    public string Title {get; set;}
    public double Average {get; set;}
}

```

Листинг Б.7 – Файл ReviewController.cs

```

using System.Data.Common;
using AutoTraveller.ActionResults;
using AutoTraveller.Data;
using AutoTraveller.Services;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace AutoTraveller.Controllers;

[Route("review")]
[ApiController]
[Authorize]
public class ReviewController : Controller
{
    private readonly ApplicationContext _applicationContext;
    private readonly Func<string, HtmlResult> _htmlResultFactory;
    private readonly UserFinderService _userFinderService;

```

```

public ReviewController(
    ApplicationContext applicationContext,
    Func<string, HtmlResult> htmlResultFactory,
    UserFinderService userFinderService)
{
    _applicationContext = applicationContext;
    _htmlResultFactory = htmlResultFactory;
    _userFinderService = userFinderService;
}

[HttpGet]
public IActionResult Index()
{
    return _htmlResultFactory(@"./wwwroot/pagestructure/review.html");
}

[HttpPost]
public async Task<IActionResult> SendReview([FromBody] Review review)
{
    _applicationContext.Attach(review);
    var user = await
_userFinderService.GetUserAsync(HttpContext.User.Identity!.Name!);
    review.User = user;

    await _applicationContext.Grades.AddRangeAsync(review.Grades);
    await _applicationContext.Reviews.AddAsync(review);
    await _applicationContext.SaveChangesAsync();

    return Ok();
}
}

```

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД КЛИЕНТСКОЙ ЧАСТИ
(рекомендуемое)

Листинг В.1 – Файл FeedController.js

```
renderReviews();

async function renderReviews()
{
    let reviews = await sendRequest('', 'feed?count=5');
    if(reviews.length === 0)
    {
        const element = document.getElementById('empty-feed-message');
        element.hidden = false;
        return;
    }
    for(let i = 0; i < reviews.length; i++)
    {
        const review = reviews[i];
        renderReview(review);
    }
}

function renderReview(review)
{
    let divOrderSpace = document.getElementById('reviews-content');
    let divStoredOrder = document.createElement('div');

    const userName = review.UserName;
    const cityFrom = review.CityFrom;
    let dateFrom = review.DateFrom.substring(0, 10).replaceAll('-', ' ');
    let cityTo = review.CityTo;
    let dateTo = review.DateTo.substring(0, 10).replaceAll('-', ' ');
    let comment = review.Comment;
    let grades = review.Grades;
    divStoredOrder.className = 'review';

    let travelInformation = `<table><tr><td>Пользователь:
${userName}</td></tr></table><table><tr><td><p>Город отправления:
</p></td><td>${cityFrom}</td><td><p>Дата отправления:
</p></td><td>${dateFrom}</td></tr><tr><td><p>Город прибытия:
```

```

</p></td><td><p>${cityTo}</p></td><td><p>Дата прибытия:
</p></td><td><p>${dateTo}</p></td></tr></table>`

    let visitedPlaces = getVisitedPlaces(grades);

    let commentInfo = `

Комментарий:</p><p>${comment}</p>`;

    divStoredOrder.innerHTML =
`$${travelInformation}${visitedPlaces}${commentInfo}`

    divOrderSpace.append(divStoredOrder)
}

function getVisitedPlaces(grades)
{
    let result = '<table class="places"><tr><td>Посещенное
место</td><td><p>Оценка</p></td></tr>'

    if(grades.length === 0) return '';
    for(let i = 0; i < grades.length; i++)
    {
        const grade = grades[i];
        console.log(grade)

        result +=
`<tr><td>${grade.Title}</td><td><p>${grade.Rate}</p></td></tr>`;
    }
    result += `</table>`

    return result;
}

async function sendRequest(json, uri) {
    const myHeaders = new Headers()
    myHeaders.append('Content-Type', 'application/json')
    const request = new Request(uri, {
        method: 'POST',
        body: json,
        headers:myHeaders
    });

    let search_result = fetch(request)
        .then((response) => {


```

```

        return response.json()
    })

    return search_result;
}

async function logout()
{
    await sendRequest('', '/auth/logout');
}

```

Листинг В.2 – Файл LoginValidation.js

```

const urlParams = new URLSearchParams(window.location.search);
const registerParam = urlParams.get('login');
console.log(registerParam)
if(registerParam === 'wrong-data')
{
    document.getElementById("wrong-data-error").hidden = false
}

function validateForm() {
    let loginElement = document.getElementById('log');
    let login = loginElement.value;
    let passwordElement = document.getElementById('pas');
    let password = passwordElement.value;

    if((login.length < 4 || login.length > 16) && (password.length < 4 ||
password.length > 16)) {
        alert("Логин и пароль должны иметь размер от 4 до 16 символов");
        return false;
    }
    if (login.length < 4 || login.length > 16) {
        alert("Логин должен иметь размер от 4 до 16 символов");
        return false;
    }
    if (password.length < 4 || password.length > 16) {
        alert("Пароль должен иметь размер от 4 до 16 символов");
    }
}

```

```

        return false;
    }
}

function sendRequest(json, uri) {
    const myHeaders = new Headers()
    myHeaders.append('Content-Type', 'application/json')
    const request = new Request(uri, {
        method: 'POST',
        body: json,
        headers:myHeaders
    });

    let search_result = fetch(request)
        .then((response) => {
            return response.json();
        })

    return search_result;
}

```

Листинг В.3 – Файл RatingController.js

```

renderRating();

async function renderRating()
{
    var rating = await sendRequest('', '/rating?count=10');
    renderRate(rating)
}

async function renderRate(rating)
{
    let divOrderSpace = document.getElementById('rating-content');
    let divStoredOrder = document.createElement('div');

    const title = rating.Title;
    const average = rating.Average;

```

```

let tableContent = getRateTableContent(rating)

divStoredOrder.className = 'rating';
if(rating.length === 0)
{
    divStoredOrder.innerHTML = `Пусто`
    divOrderSpace.append(divStoredOrder);
    return;
}

divStoredOrder.innerHTML = `




```

```

        headers:myHeaders
    });

    let search_result = fetch(request)
        .then((response) => {
            return response.json()
        })

    return search_result;
}

```

Листинг В.4 – Файл RegistrationValidation.js

```

const urlParams = new URLSearchParams(window.location.search);
const registerParam = urlParams.get('register');
console.log(registerParam)
if(registerParam === 'already-exists')
{
    document.getElementById("already-exists-error").hidden = false
}

function validateForm() {
    let login = document.forms["registration-form"]["login"].value;
    let password = document.forms["registration-form"]["password"].value;

    if((login.length < 4 || login.length > 16) && (password.length < 4 ||
password.length > 16)) {
        alert("Логин и пароль должны иметь размер от 4 до 16 символов");
        return false;
    }

    if (login.length < 4 || login.length > 16) {
        alert("Логин должен иметь размер от 4 до 16 символов");
        return false;
    }

    if (password.length < 4 || password.length > 16) {
        alert("Пароль должен иметь размер от 4 до 16 символов");
        return false;
    }
}

```

Листинг В.5 – Файл ReviewController.js

```
function addPlace()
{

    var table = document.getElementById("places");
    var row = table.insertRow();
    row.insertCell(0);
    var cell2 = row.insertCell(1);
    var cell3 = row.insertCell(2);

    cell2.innerHTML = '<input type="text" class="profile-info-places-title"
value = "">'

    cell3.innerHTML = `<input type="number" class="profile-info-places-rate"
value = "" maxlength="1" min="1" max="5">`

}

async function sendReview() {
    var cityFrom = document.getElementById('city-from').value;
    var dateFrom = document.getElementById('date-from').value;
    var cityTo = document.getElementById('city-to').value;
    var dateTo = document.getElementById('date-to').value;
    var comment = document.getElementById('comment').value;
    var grades = [];

    var table = document.getElementById("places");
    for (var i = 0, row; row = table.rows[i]; i++) {
        let title = '';
        let grade = -1;

        for (var j = 0, col; col = row.cells[j]; j++) {
            if (col.children.length === 1) {
                let p1 = getByClass(col, 'profile-info-places-title');
                if (p1 !== null) {
                    title = p1.value;
                }

                let p2 = getByClass(col, 'profile-info-places-rate');
                if (p2 !== null) {
```

```

        grade = Number(p2.value);
    }
}

if (title.length < 1) continue;
if (grade < 1 || grade > 5) continue;

let gradeR = {
    Title: title,
    Rate: grade
};

grades.push(gradeR);
}

var review = {
    CityFrom: cityFrom,
    DateFrom: dateFrom,
    CityTo: cityTo,
    DateTo: dateTo,
    Comment: comment,
    Grades: grades
}

if(cityFrom.length === 0 || dateFrom.length === 0 || cityTo.length === 0
|| dateTo.length === 0 || comment.length === 0)
{
    return;
}

await sendRequest(JSON.stringify(review), 'review')
}

function getByClass(parent, className)
{
    if (parent == null) return null;
    const children = parent.children

```

```

    for(let i = 0; i < children.length;i++)
    {
        if (children[i].getAttribute('class') === className)
        {
            return children[i];
        }
    }
    return null;
}

```

```

function sendRequest(json, uri) {
    const myHeaders = new Headers()
    myHeaders.append('Content-Type', 'application/json')
    const request = new Request(uri, {
        method: 'POST',
        body: json,
        headers:myHeaders
    });

    let search_result = fetch(request)
        .then((response) => {
            return response.json()
        })

    return search_result;
}

```

```

async function logout()
{
    await sendRequest('', '/auth/logout');
}

```