СОДЕРЖАНИЕ

[Введение 6](#_Toc58934694)

[1 Актуальность карпулинга в современном мире 8](#_Toc58934695)

[1.1 Важность карпулинга в современном мире 8](#_Toc58934696)

[1.2 Существующие приложения для совместного использования   
частного автомобиля 10](#_Toc58934697)

[1.3 Выводы и постановка задач на дипломное проектирование 14](#_Toc58934698)

[2 Эргономическое проектирование информационной системы 16](#_Toc58934699)

[2.1 Анализ функций и их распределение в проектируемой информационной системе 16](#_Toc58934700)

[2.2 Проектирование алгоритмов работы пользователей   
и эргономических требований к системе 20](#_Toc58934701)

[2.3 Разработка сценария информационного взаимодействия и оценка эргономичности ее пользовательского интерфейса 36](#_Toc58934702)

[3 Разработка веб-приложения для совместного использования   
частного автомобиля 55](#_Toc58934703)

[3.1 Разработка структуры информационной системы 55](#_Toc58934704)

[3.2 Разработка структуры базы данных 66](#_Toc58934705)

[3.3 Тестирование программного средства 72](#_Toc58934706)

[4 Технико-экономическое обоснование разработки веб-приложения для совместного использования частного автомобиля 86](#_Toc58934707)

[4.1 Характеристика программного средства 86](#_Toc58934708)

[4.2 Расчет затрат на разработку программного средства 87](#_Toc58934709)

[4.3 Расчет экономического эффекта у разработчика 91](#_Toc58934710)

[5 Реализация эргономической совместимости работника и технического средства при разработке и эксплуатации веб-приложения 93](#_Toc58934711)

[Заключение 99](#_Toc58934712)

[Список использованных источников 100](#_Toc58934713)

[Приложение А (обязательное) Листинг программы 103](#_Toc58934714)

[Ведомость дипломного проекта 130](#_Toc58934715)

# ВВЕДЕНИЕ

В связи с тем, что из года в год на рынке появляется все больше аналогов одного и того же продукта во всех сферах жизни, одним из способов потенциального привлечения клиентов является не только улучшение качества предлагаемого продукта, но и его клиентоориентированность. Программный продукт не исключение. Самый лучший способ сделать программный продукт клиентоориентированным – это использовать ранее полученный пользовательский опыт, чтобы модифицировать продукт так, чтобы сделать его более адаптированным под пользователя.

К выбранной теме дипломного проекта, а именно разработка веб-приложения для совместного использования частного автомобиля, существует не очень большое количество аналогов, однако, основной их недостаток – они не имеют распространения в Республике Беларусь, а также большинство из их не отвечают одному из важнейших критериев эргономичности – удобство в использовании.

Количество машин на дорогах мира растет с каждым днём. Власти планеты, всерьез озадаченные проблемой автомобильных заторов, всеми силами пытаются найти способ, который помог бы разгрузить дороги.

На данный момент было предпринято несколько мер для разгрузки дорог:

* ограничение числа продаваемых в стране машин;
* создание платных дорог центральных городов;
* организация «Дня без автомобиля».

Действенный и экономичный способ борьбы с пробками придумали в США. Этот способ получил название Carpool и за годы своего существования не только помог решить проблему переполненности американских автодорог, но и заработал нешуточную популярность среди коммуникабельной части населения всего мира.

Приложение, позволяющее с легкостью организовать так называемый Carpool будет сильно востребовано среди пользователей, так поможет попутчику организовать свою поездку в комфорте и по не высокой цене, в места куда не ходит общественный транспорт, а водителям разделить трату на топливо. В данный момент пользователи предпочитают маршрутные транспортные средства для передвижения загородом, однако есть места куда маршрутные транспортные средства не ходят. Данное веб-приложение позволит найти водителя, движущегося в интересующем вам направлении. Таким образом, появляется спрос на простое, удобное и доступное приложение, позволяющее быстро организовать связь между водителем и клиентом по интересующему обоих маршруту.

Целью дипломного проектирования является разработка веб-приложения для совместного использования частного автомобиля и его инженерно-психологиеское обеспечение.

Для достижения цели необходимо решить следующие задачи:

1. Спроектировать информационную систему на основе информации о технологиях, существующих решениях и результатах эргономического проектирования.
2. Разработать информационную систему согласно проекту.

Назначение проектируемого приложения: приложение будет являться сервисом, организующем связь между попутчиком и водителем движущимися в одном направлении.

В пояснительной записке к дипломному проекту будет описана:

1. Информация о технологиях и программах, с помощью которых разработано веб-приложение для совместного использования частного автомобиля.
2. Обзор существующих систем, позволяющих организовать совместное использования частного автомобиля.
3. Анализ функций, которые реализуются в веб-приложении.
4. Разработка алгоритмов действия пользователей: водителя и попутчика.
5. Анализ факторов, определяющих эффективность функционирования системы «человек-машина-среда».
6. Разработка информационной системы.
7. Технико-экономическое обоснование разработки веб-приложения для совместного использования частного автомобиля.
8. Реализация эргономической совместимости работника и технического средства при разработке и эксплуатации веб-приложения.

# 1 АКТУАЛЬНОСТЬ КАРПУЛИНГА В СОВРЕМЕННОМ МИРЕ

## 1.1 Важность карпулинга в современном мире

Человек нашего времени постоянно стремится к оптимизации: времени, денег и пространства. Карпулинг позволяет использовать свободные места частных легковых автомобилей, что снижает суммарное потребление топлива и транспортные расходы участников поездки. Кроме того, регулярное использование совместных поездок гарантированно снижает трафик, помогает разгрузить дороги в часы пик и уменьшает пагубное воздействие автомобильных выбросов на окружающую среду. Пассажиры и водители могут использовать карпулинг как единожды, так и регулярно. Сервис особенно эффективен в районах, слабо охваченных системой [общественного транспорта](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%82%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82).

Карпулинг помогает использовать в поездке свободные места легковых автомобилей. Пассажир и водитель находят друг друга, договариваются о месте встречи, маршруте, об оплате этой услуги (обычно в виде компенсации расходов на бензин) – и вместе путешествуют до места назначения. Это выходит комфортнее, чем на общественном транспорте, дешевле, чем такси, и безопаснее, чем автостопом. Для водителя это тоже выгодно, так как помогает снизить издержки на топливо, ремонт и парковку автомобиля. Кроме того, в дальней поездке попутчик может сменить водителя за рулем или не дать уснуть, отвлекая разговорами. В туристических поездках карпулинг – это еще и способ узнать больше о новом месте и завести полезные знакомства [1].

Выгоднее всего пользоваться карпулингом при поездках на дальние расстояния или регулярных поездках по одному и тому же маршруту – например, на работу со своим коллегой-водителем или другим человеком, которому по пути. Это связано с тем, что иногда, чтобы подобрать пассажира, водителю приходится заезжать за ним, что увеличивает время поездки. Тем не менее, разовые поездки из одного конца города в другой тоже возможны и выгодны [1].

При этом карпулинг приносит ощутимую выгоду и для окружающей среды. В первую очередь, регулярные совместные поездки помогают снизить трафик и сократить количество пробок. Следовательно, уменьшается объем выхлопных газов и негативное влияние на экологию [1].

В Джакарте в 1992 году в рамках борьбы с пробками ввели закон, который требовал, чтобы в часы-пик в машинах, въезжающих на двенадцатиполосную городскую дорогу Джалан Судирман, находилось не менее трех пассажиров. Когда в 2016 году его отменили, на этой и прилегающих улицах число автомобилей сильно увеличилось и привело к росту утренних пробок на 46 %, а вечерних – на 87 %. Авторы исследования отмечали, что машины двигались немногим быстрее пешеходов [1].

Согласно исследованию, проведенному в Калифорнийском университете Беркли в 2010 году, около 20 % респондентов готовы использовать карпулинг, по крайней мере, один раз в неделю. Кроме того, карпулинг пользуется большим спросом у тех, кто ежедневно передвигается в одном и том же направлении (30 %), чем у тех, кто совершают случайные поездки. Основными препятствиями, замедляющими развитие карпулинга, являются короткие поездки и увеличение времени, проводимого в пути.

Использование данного способа передвижения стало возможным благодаря достижениям техники:

1. [GPS-устройства](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B2%D0%B8%D0%B3%D0%B0%D1%82%D0%BE%D1%80_(%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE)) для определения маршрута водителя и организации совместной поездки.
2. [Смартфоны](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD), позволяющие пользователям услуги сделать запрос на поездку, независимо от места нахождения.

[Социальные сети](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8), делающие услугу прозрачной и повышающие уровень доверия между водителем и пассажиром.

Первые проекты, напоминающие карпулинг, появились в 1990-х годах, но сразу же встретили на своем пути такие препятствия, как необходимость развития сообщества пользователей и удобного способа взаимодействия друг с другом. Постепенно телефоны, посредством которых организовывались поездки, были заменены интернетом, электронной почтой и смартфонами, а крупные компании разработали специальные сообщества для пользователей. По состоянию на 2006, такси подобного рода оказывало негативное влияние на автомобильную промышленность [2].

Организовать и поддерживать карпулинг бывает непросто. Власти города способны помочь автовладельцам, организуя специальные места для остановок попутчиков, как это сделано в [Вашингтоне](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D1%88%D0%B8%D0%BD%D0%B3%D1%82%D0%BE%D0%BD_(%D0%BE%D0%BA%D1%80%D1%83%D0%B3_%D0%9A%D0%BE%D0%BB%D1%83%D0%BC%D0%B1%D0%B8%D1%8F)). [Правительство Москвы](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B0%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE_%D0%9C%D0%BE%D1%81%D0%BA%D0%B2%D1%8B) одобрило такие остановки на территории города. В США – вокруг станций метро и крупных автобусных остановок размещены таблички с указаниями направлений для совместной поездки. Также организуются выделенные, для водителей с пассажирами, карпул-полосы движения.

## 1.2 Существующие приложения для совместного использования частного автомобиля

На данный момент на рынке представлено небольшое количество веб-приложений, которые предоставляют возможность поиска попутчиков.

В основном такие веб-приложения направлены на рынки Европы и России. Опишем популярные сервисы, которые могут помочь в осуществлении поиска попутчика.

Самый популярный в России и Европе сервис для поиска попутчиков – BlaBlaCar.

Сервис предоставляет следующие преимущества для пассажиров:

1. Экономия времени: в предложении поездки указано актуальное число свободных мест. Когда пользователь выбирает подходящую поездку, он сразу видит, сколько в машине осталось свободных мест. Больше не нужно звонить десяткам водителей, чтобы уточнить, есть ли у них место – теперь можно просто сразу его забронировать [3].
2. Уверенность: пользователь знает, кто станет его попутчиком. Пользователь может сразу узнать больше о других участниках путешествия. Просматривая каждую поездку, можно ознакомиться не только с полным профилем водителя, но и с профилями людей, которые уже забронировали место на эту поездку [3].
3. Удобство: пользователь может забронировать место в любое время. Благодаря системе онлайн-бронирования, можно мгновенно забронировать место на выбранную поездку. Больше не нужно звонить или писать водителю, теперь это можно сделать за несколько кликов. Запрос будет подтвержден сразу или, если водитель решил подтверждать бронирования вручную, в течение указанного пользователем периода времени [3].
4. Все детали поездки под рукой. После подтверждения бронирования пользователь получает письмо по email со всей необходимой информацией (контактные данные водителя, дата и время отправления, сумма к оплате наличными водителю) [3].
5. Бронирование можно отменить. Если у пользователя изменятся планы и он передумает ехать, он сможет отменить бронирование прямо на сайте [3].

Сервис предоставляет следующие преимущества для водителей:

1. Выбор пассажиров за пользователем. Пользователь может выбрать один из двух способов подтверждении бронирования: Автоматически – пользователю не нужно ничего делать, бронирование каждого пассажира будет подтверждено автоматически сразу после поступления. Вручную: чтобы подтверждать бронирование каждого пассажира самостоятельно. Пользователь может подтвердить или отклонить бронирование через сайт, приложение или ответив на SMS [3].
2. Число свободных мест обновляется автоматически. Когда пассажиры бронируют место онлайн, система автоматически обновляет количество свободных мест в машине для всего маршрута и для каждого участка в отдельности. Больше пользователю не будут постоянно звонить потенциальные пассажиры, чтобы узнать, остались ли места [3].
3. Удобная система оповещения, не обязательно быть онлайн. Пользователь будет получать оповещения по SMS и email о каждом новом запросе на бронирование.  Есть возможность мгновенно посмотреть профили людей, которые хотят ехать с пользователем, чтобы узнать больше о потенциальных пассажирах [3].
4. Пассажиры относятся к поездке более ответственно. Пассажиры бронируют место в машине онлайн. Это значит, что вместо того, чтобы заранее договариваться обо всем по телефону, пользователь сразу может подтвердить их бронирование онлайн. Пассажиры получат официальное подтверждение того, что место сохранено за ними [3].
5. Больше информации о пассажирах, которую легко посмотреть. Список пассажиров, забронировавших места, доступен онлайн. Пользователь может посмотреть их номера телефонов, чтобы иметь возможность по необходимости связаться с ними и обсудить детали путешествия [3].

В 2017 году у BlaBlaCar появился конкурент – BeepCar, основным отличием от BlaBlaCar было то, что разработчик рекомендует специально рассчитанную стоимость поездки. Водитель может завысить её, в пределах нормы, или снизить до нуля [4]. Однако в 2018 году сервис поиска попутчиков BlaBlaCar выкупил сервис BeepCar у холдинга Mail.Ru Group и заключил соглашение о сотрудничестве с группой компаний. По условиям соглашения Mail.Ru Group будет продавать BlaBlaCar рекламу на своих площадках и с осени 2018 года станет перенаправлять пользователей BeepCar на сайт BlaBlaCar [5].

На рисунке 1.1 представлен пользовательский интерфейс приложения BlaBlaCar, в котором можно выделить хорошо проработанный дизайн с приятной цветовой гаммой.

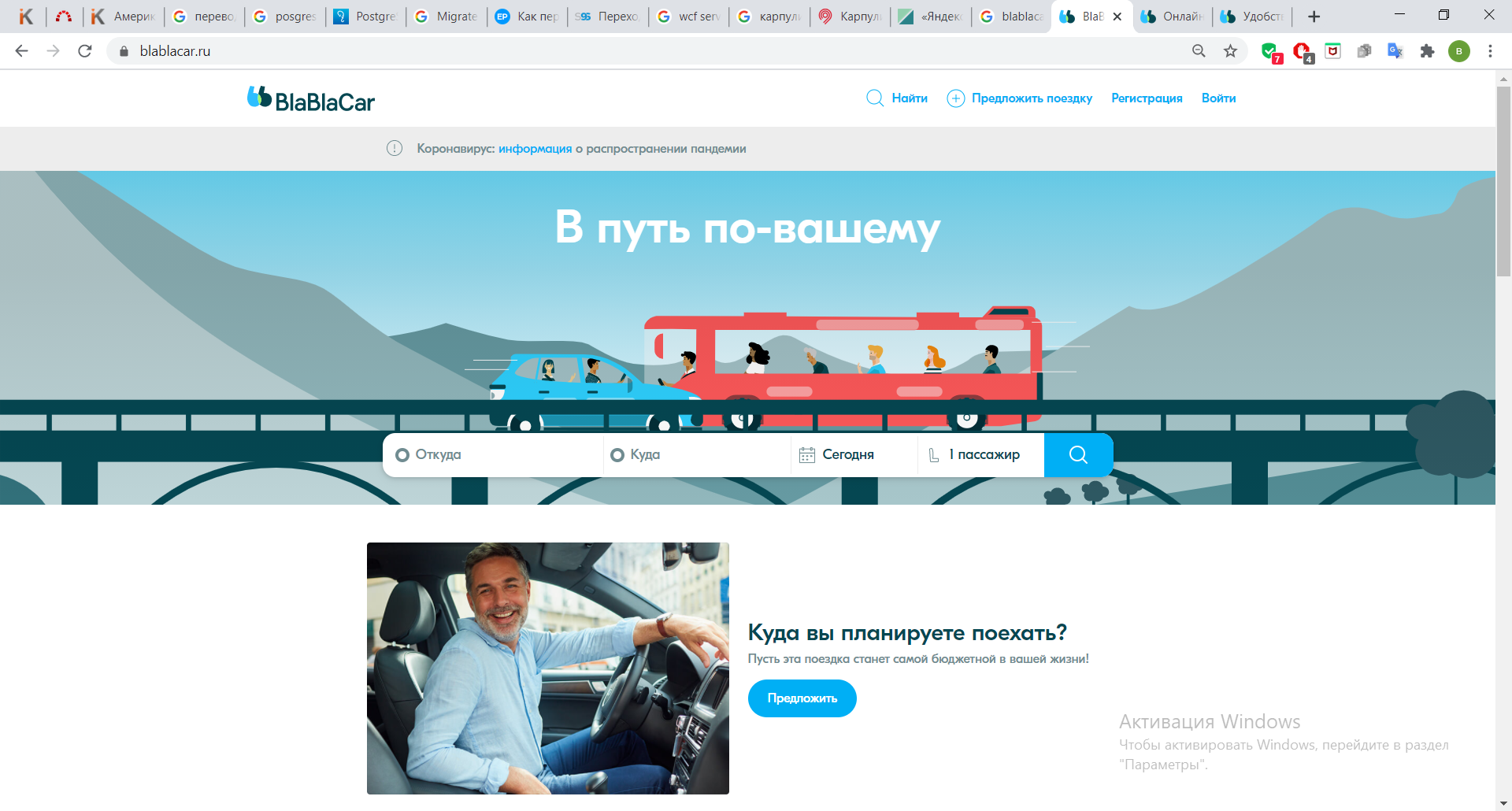


Рисунок 1.1 – Пользовательский интерфейс blablacar.ru [6]

На территории России распространены еще несколько сервисов, два из их: mahnem.ru и dovezu.ru. Пользовательский интерфейс веб-приложений dovezu.ru и mahnem.ru представлен на рисунках 1.2 и 1.3.

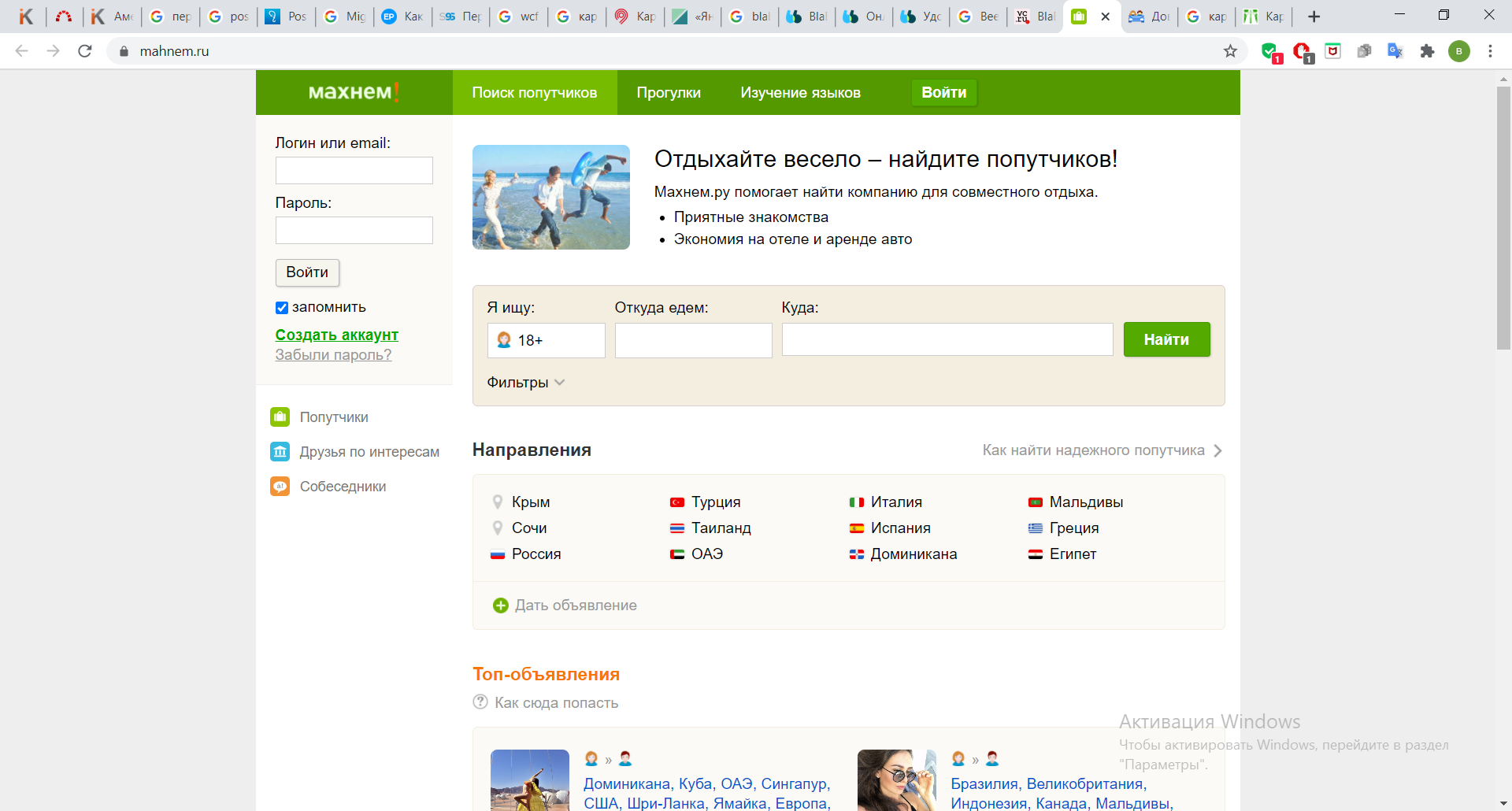


Рисунок 1.2 – Пользовательский интерфейс mahnem.ru [7]

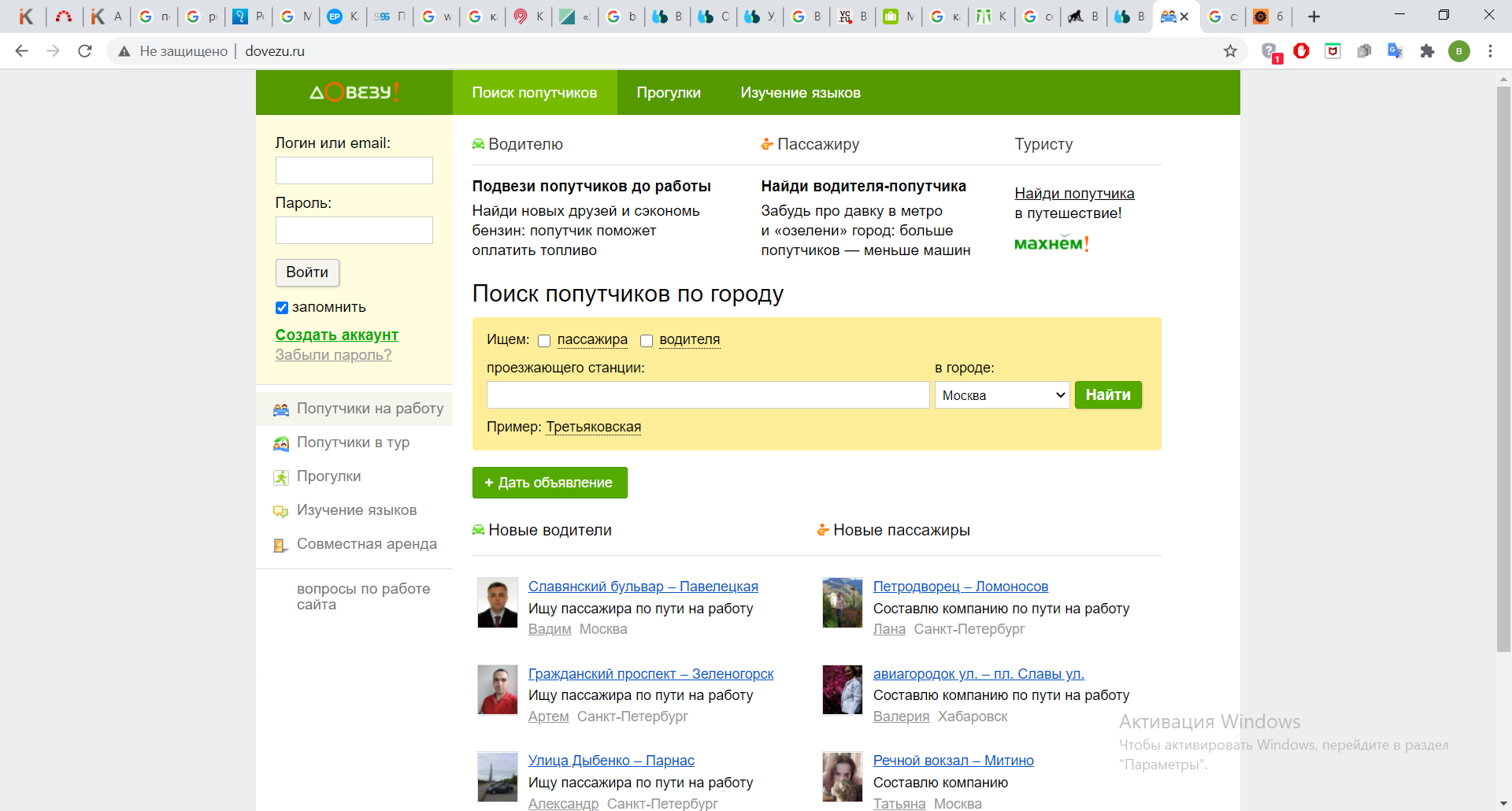


Рисунок 1.3 – Пользовательский интерфейс dovezu.ru [8]

Оба сервиса имеют не проработанный, схожий дизайн. Сервисы не имеют такого функционала и распространения как BlaBlaCar. Пользователи могут лишь оставить заявку на сайте, с указанием маршрута. Затем пользователям отображаются водители или потенциальные попутчики, движущиеся в требуемом направлении, также есть чат, в котором можно обговорить детали поездки. Основная ориентация сервисов – на организацию совместных путешествий. Отличительной чертой сервисов является то, что они позволяют не только организовать совместную поездку, но и прогулку, изучение языков и даже занятие танцами. Данные сервисы будут неудобны для обеспечения регулярных поездок между городами, также немаловажным фактором является то, что администрация не следит за пользователями своего ресурса, что черевато появлением своего рода мошенников.

Анализируя перечисленные выше приложения можно выделить одно заметно превосходящее остальные приложение – Blablacar. Blablacar превосходит осталньые приложения по интерфейсу, функциональности, распространению, однако оно не имеет распространения в Белорусии, на которую будет ориетировано веб-приложение разрабатываемое в рамках дипломого проекта.

## 1.3 Выводы и постановка задач на дипломное проектирование

Информационная система должна быть ориентирована на конечного пользователя, поэтому разрабатываемый интерфейс должен соответствовать следующим принципам:

1. Интерфейс должен быть простой и понятный. Задачи должны решаться минимальным числом действий [9–13].
2. Рекомендуется избегать сложных действий. Действий, которые заставляют пользователей думать лучше избегать [9–13].
3. Убираем очевидное. Не стоит показывать очевидные элементы интерфейса, нужно сосредоточиться только на действительно необходимых вещах [9–13].
4. Соотношение сигнал/шум. В каждом интерфейсе есть важные элементы (сигналы) и маловажные или даже бессмысленные для определенной части системы (шум), нужно концентрироваться на сигналах и избегать шума [9–13].
5. Привычные элементы управления. В любом современном интерфейсе есть много элементов управления, лучше использовать привычные элементы и визуальные образы [9–13].
6. Люди не читают, люди просматривают. Пользователи не любят читать большие массивы текстов, предпочитая им картинки или короткие заметки [9–13].
7. В одном функциональном блоке не может быть больше 5–7 элементов, иначе пользователь не сможет удерживать информацию в кратковременной памяти [9–13].
8. Принцип группировки. Информацию на странице желательно разбивать на логические блоки (группы), так пользователю легче ориентироваться [9–13].
9. Все полезное на виду. Все важные элементы интерфейса должны быть на виду и соответствующим образом выделены [9–13].
10. Принцип трех кликов. Должно быть не более трех кликов для перехода из одного раздела в другой, это же правило применимо к главной странице: любая важная информация должна быть доступна не более чем в три клика [9–13].
11. Однородность. В больших проектах часто встречается однородный функционал в разных частях сайта (например, комментарии), он не должен отличаться. Это же касается и стиля [9–13].
12. Способы решения задачи. Пользователям нужно предлагать способы решения их задач с помощью интерфейса, и эти способы должны быть очевидны [9–13].
13. Принцип единства. Настройки и элементы управления нужно стараться не прятать в отдельные разделы, а дать возможность управлять из одного места, если это уместно [9–13].
14. Тренды. Стоит учитывать современные тенденции, чтобы интерфейс не устарел еще до выхода проекта [9–13].

При проектировании информационной системы необходимо:

* проанализировать функции и их распределение в проектируемой информационной системе;
* разработать алгоритмы работы пользователей информационной системы;
* разработать архитектуру приложения, отвечающую требованиям эффективности, гибкости и масштабируемости системы;
* разработать эргономические требования и сценарии информационного взаимодействия;
* разработать структуру информационной системы;
* разработать структуру базы данных;
* рассчитать затраты на разработку информационной системы;
* провести расчеты по охране труда, то есть реализацию эргономической совместимости работника и технического средства при разработке и эксплуатации веб-приложения.

Так же необходимо разработать следующий графический материал:

* структурную схему приложения;
* структурную схему базы данных;
* блок-схему алгоритма авторизации;
* блок-схему алгоритма создания маршрута;
* эскизы рабочих окон программы.

# 2 ЭРГОНОМИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 2.1 Анализ функций и их распределение в проектируемой информационной системе

Целью дипломного проекта является разработка веб-приложения для совместного использования частного автомобиля и его инженерно-психологиеское обеспечение. Данная система представляет собой приложение, которое позволяет совместно использовать частный автомобиль для поездки. То, что оно является веб-сайтом, позволяет пользователям использовать данную систему в любом месте и с любого устройства, имеющего доступ в сеть Интернет.

Учитывая поставленную цель работы, а также назначение системы, определим функции, которые должны выполнятся:

1. Регистрация учетной записи.
2. Восстановление доступа к учетной записи.
3. Смена пароля учетной записи.
4. Отправление писем на почту посредством почтового сервиса.
5. Добавление данных о автомобилях (для водителя).
6. Добавление данные о правах (для водителя).
7. Добавление данных о пользователе.
8. Выбор точек маршрута.
9. Отображение маршрута на карте.
10. Возможность выбора нескольких точек маршрута (для водителя).
11. Калькуляция стоимости маршрута в зависимости от выбранной точки (для клиента).
12. Просмотр информации о зарегистрированных на маршрут попутчиках (для водителя).
13. Просмотр информации о водителе, осуществляемым поездку (для клиента).
14. Просмотр информации об автомобиле, на котором будет осуществляться поездка.
15. Просмотр дополнительной информации о маршруте (точек маршрута, времени).
16. Оформление нового маршрута (для водителя).
17. Отмена поездки (для водителя).
18. Отмена регистрации на маршрут (для клиента).
19. Поиск маршрутов, удовлетворяющих требованиям (для клиента).
20. Регистрация на маршрут (для клиента).

Распределение функций в СЧМС между человеком и техническими устройствами осуществляется на основе следующих принципов:

– функция передается тому или иному компоненту системы на основе сравнительного анализа человека и техники на предмет возможности и эффективности ее выполнения ими [16];

– за человеком сохраняются функции, которые он должен выполнять в системе обязательно безо всякого дополнительного сравнительного анализа возможностей человека и машины [16].

На человека следует возлагать выполнение функций по:

* распознаванию ситуации в целом по её многим сложно связанным характеристикам, а также при неполной информации о ней [16];
* осуществлению функций индуктивного вывода, т.е. обобщению отдельных фактов в единую систему [16];
* решению задач, в которых отсутствует единый алгоритм или нет четко определённых правил обработки информации [16];
* решению задач, в которых требуется гибкость и приспособляемость к изменяющимся условиям, особенно задач, появление которых заранее трудно предвидеть [16];
* решению задач с высокой ответственностью в случае возникновения ошибки [16].

Машине рекомендуется поручать:

* выполнение всех видов математических расчётов [16];
* выполнение однообразных, постоянно повторяющихся операций, реализуемых по заданному алгоритму [16];
* хранение и динамическое представление больших объёмов однородной информации [16];
* решение задач, требующих дедуктивного вывода, т.е. получения на основе общих правил решений для частных случаев [16];
* выполнение действий, требующих высокой скорости реакции на команду [16].

Учитывая названные принципы, проведем анализ функций системы с целью распределения их между человеком и компьютером. Результаты работы представим в виде таблицы 2.1.

Таблица 2.1 – Распределение функций между человеком и компьютером

|  |  |  |
| --- | --- | --- |
| Название функции | Действия человека | Операции компьютера |
| 1 | 2 | 3 |
| Регистрация учетной записи. | Вводит необходимые данные в поля ввода. | Добавляет идентификационные данные в базу данных. |
| Восстановление доступа к учетной записи. | Вводит необходимые данные в поля ввода. | Генерирует новый пароль, заменяет пароль в базе данных на сгенерированный, отправляет сообщение на почту о восстановлении доступа. |
| Смена пароля учетной записи. | Вводит необходимые данные в поля ввода. | Производит смену пароля, шифрует пароль, добавляет хеш пароля в базу данных, отправляет письмо на почту пользователю, о смене пароля. |
| Отправление писем на почту посредством почтового сервиса. | Регистрирует нового пользователя, восстанавливает доступ к учетной записи или меняет пароль. | Отправляет письмо на почту пользователя о произведенных действиях. |
| Добавление данных об автомобилях (для водителя). | Вводит необходимые данные в поля ввода. | Производит валидацию данных, добавляет данные в базу данных. |
| Добавление данных о правах (для водителя). | Вводит необходимые данные в поля ввода. | Производит валидацию данных, добавляет данные в базу данных. |
| Добавление данных о пользователе. | Вводит необходимые данные в поля ввода. | Производит валидацию данных, добавляет данные в базу данных. |

Продолжение таблицы 2.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Выбор точек маршрута. | Выбирает точки маршрута. | Производит поиск населенных пунктов, улиц, адресов, с помощью апи стороннего сервиса, выводит их в выпадающий список. |
| Отображение маршрута на карте. | Выбирает точки маршрута. | Строит маршрут на карте, с помощью апи стороннего сервиса. |
| Возможность выбора нескольких точек маршрута (для водителя). | Может выбрать несколько точек маршрута. | После ввода точки маршрута, можно добавить панель для добавления последующей точки. |
| Калькуляция стоимости маршрута в зависимости от выбранной точки (для клиента). | Выбирает точки маршрута. | Калькулирует стоимость маршрута, в зависимости от выбранных клиентом точек маршрута. |
| Просмотр информации о зарегистрированных на маршрут попутчиках (для водителя). | Оформляет маршрут, затем просматривает зарегистрированных на его пользователей. | Загружает и отображает данные о зарегистрированных на маршрут пользователях. |
| Просмотр информации о водителе, осуществляемым поездку (для клиента). | Выполняет поиск маршрутов по интересующим точкам. | Загружает и отображает данные о водителе зарегистрированным за маршрутом. |
| Просмотр информации об автомобиле, на котором будет осуществляться поездка. | Выполняет поиск маршрутов по интересующим точкам. | Загружает и отображает данные о автомобиле зарегистрированным за маршрутом. |

Продолжение таблицы 2.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Просмотр дополнительной информации о маршруте (точек маршрута, времени). | Выполняет поиск маршрутов по интересующим точкам. | Загружает и отображает данные о промежуточных точках маршрута, а также времени отбытия и прибытие до каждой точки. |
| Оформление нового маршрута (для водителя). | Вводит необходимые данные в поля ввода, выбирает точки маршрута. | Производит валидацию данных, добавляет данные о маршруте в базу данных. |
| Отмена поездки (для водителя). | Отменяет поездку. | Отмечает маршрут в базе данных, как удаленный. |
| Отмена регистрации на маршрут (для клиента). | Отменяет регистрацию. | Удаляет данные о регистрации пользователя на маршрут из базы данных. |
| Поиск маршрутов, удовлетворяющих требованиям (для клиента). | Выбирает точки маршрута. | Выполняет поиск маршрутов по удовлетворяющим точкам. |
| Регистрация на маршрут (для клиента). | Регистрируется на маршрут. | Добавляет данные в базу данных, о регистрации пользователя на маршрут. |

На основании полученных результатов можно сделать вывод, что проектируемая система будет состоять из двух подсистем, а именно, подсистемы «попутчик – ПК – среда» и подсистемы «водитель – ПК – среда».

## 2.2 Проектирование алгоритмов работы пользователей и эргономических требований к системе

Деятельность пользователя в системе зависит от выполняемых им функций, которые определяются его ролью. Разграничение типа пользователя (водитель, попутчик) происходит на этапе авторизации и аутентификации посредством выбора соответствующей ему роли.

Водитель может заполнить данные о пользователе, о водительском удостоверении, о автомобиле. Сменить учетные данные. Основная задача водителя состоит в создании маршрута.

Попутчик может также заполнить данные о пользователе, сменить учетные данные, однако основной задачей пользователя является –регистрация на маршрут.

Общие алгоритмы, которые могут выполнять как попутчик, так и водитель представлены в таблицах 2.2 – 2.5.

Алгоритмы работы попутчика в системе представлены   
в таблицах 2.6, 2.7.

Алгоритмы работы водителя в системе представлены   
в таблицах 2.8 – 2.11.

Таблица 2.2 – Алгоритм регистрации в системе

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-1 | Автор | Макатров В.Д. |
| Название варианта использования | Вход в систему | Дата | 19.11.2020 |
| Условие | Описание | | |
| 1 | 2  3 | | |
| Описание | Пользователь регистрируется в системе | | |
| Действующее лицо/ Роль | Пользователь: водитель, попутчик | | |
| Предусловия | PRE-1 Рабочий компьютер включен | | |
| Триггер | Пользователь указывает, что хочет зарегистрироваться в системе. | | |

Продолжение таблицы 2.2

|  |  |
| --- | --- |
| 1 | 2  3 |
| Основной сценарий | 1. Пользователь запускает браузер. 2. Переходит на сайт приложения. 3. Пользователь нажимает кнопку «Регистрация». 4. Отображается форма для регистрации. 5. Пользователь вводит почту и пароль в соответствующие поля. 6. Пользователь нажимает кнопку «Регистрация». 7. Система шифрует пароль пользователя. 8. Система вносит данные от аккаунта в базу данных. 9. Система отправляет письмо на почту пользователя, о успешной регистрации в сервисе. 10. Отображается форма для входа в систему. |
| Ожидаемые результаты | POST-1 Отобразилась форма для регистрации.  POST-2 Пароль пользователя зашифрован.  POST-3 Данные от аккаунта занесены в базу данных.  POST-4 На почту пользователя отправлено сообщение о успешной регистрации.  POST-5 Отобразилась форма для входа в систему. |
| Исключения | И1 Не все поля заполнены.   1. Система выдает сообщение «Заполните обязательные поля».   И2 Указанная почта уже зарегистрирована.   1. Система выдает сообщение «Указанная почта уже зарегистрирована».   И3 Введенная почта не соответствует требованиям.   1. Система выдает сообщение «Указанная почта должна иметь …».   И4 Введенный пароль не соответствует требованиям.   1. Система выдает сообщение «Пароль должен содержать …». |

Таблица 2.3 – Алгоритм входа в систему

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-2 | Автор | Макатров В.Д. |
| Название варианта использования | Вход в систему | Дата | 19.11.2020 |
| Условие | Описание | | |
| 1 | 2  3 | | |
| Описание | Пользователь выполняет вход в систему для последующей работы | | |
| Действующее лицо/ Роль | Пользователь: водитель, попутчик | | |
| Предусловия | PRE-1 Рабочий компьютер включен  PRE-2 Пользователь зарегистрирован в системе | | |
| Триггер | Пользователь указывает, что хочет войти в систему | | |
| Основной сценарий | 1. Пользователь запускает браузер. 2. Переходит на сайт приложения. 3. Пользователь вводит логин и пароль в соответствующие поля (И1). 4. Пользователь нажимает кнопку «Войти» (И2). 5. Выполнен вход в систему. 6. Отобразилось окно с маршрутами. | | |
| Ожидаемые результаты | POST-1 Выполнен вход в систему.  POST-2 Отобразились элементы системы в соответствии с ролью пользователя. | | |

Продолжение таблицы 2.3

|  |  |
| --- | --- |
| 1 | 2  3 |
| Исключения | И1 Не все поля заполнены.   1. Система выдает сообщение «Заполните обязательные поля».   И2 Неправильно введены данные для входа.   1. Система выдает сообщение «Введены неверные данные». |

Таблица 2.4 – Алгоритм заполнения данных о пользователе.

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-3 | Автор | Макатров В.Д. |
| Название варианта использования | Заполнение данных о пользователе | Дата | 19.11.2020 |
| Условие | Описание | | |
| 1 | 2  3 | | |
| Описание | Пользователь заполняет личные данные | | |
| Действующее лицо/ Роль | Пользователь: водитель, попутчик | | |
| Предусловия | PRE-1 Выполнен вход в систему.  PRE-2 Пользователь до этого не вводил данные. | | |
| Триггер | Пользователь заходит в сервис. | | |

Продолжение таблицы 2.4

|  |  |
| --- | --- |
| 1 | 2  3 |
| Основной сценарий | 1. Отображается форма для заполнения личных данных. 2. Пользователь заполняет данные. 3. Пользователь нажимает кнопку «Сохранить». 4. Сервис сохраняет данные в базу данных.    1. У попутчика отображается форма с активными маршрутами.    2. У водителя отображается форма для заполнения данных о водительском удостоверении. |
| Ожидаемые результаты | POST-1 Форма для заполнения личных данных отобразилась.  POST-2 Информация сохранена в базе данных.  POST-3 Отобразилась форма в соответствии с ролью пользователя. |

Таблица 2.5 – Алгоритм просмотра маршрутов попутчиком.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-4 | Автор | Макатров В.Д. | |
| Название варианта использования | Просмотр активных маршрутов | Дата | 19.11.2020 | |
| Условие | Описание | | | |
| 1 | 2  3 | | |
| Описание | Пользователь просматривает активные маршруты | | | |
| Действующее лицо/ Роль | Пользователь: водитель, попутчик | | | |
| Предусловия | PRE-1 Выполнен вход в систему.  PRE-2 Пользователь зарегистрирован на маршрут, водитель имеет активный маршрут. | | | |

Продолжение таблицы 2.5

|  |  |
| --- | --- |
| 1 | 2  3 |
| Триггер | Пользователь указывает, что хочет посмотреть информацию о маршруте |
| Основной сценарий | 1. Пользователь переходит в раздел с активными маршрутами.    1. Отобразился экран с информацией о маршрутах на которые зарегистрирован попутчик.    2. Отобразился экран с информацией о маршрутах водителя. |
| Ожидаемые результаты | POST-1 Выполнен вход в систему.  POST-2 Информация отобразилась. |

Таблица 2.6 – Алгоритм регистрации на маршрут

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-5 | Автор | Макатров В.Д. | |
| Название варианта использования | Регистрация на маршрут | Дата | 19.11.2020 | |
| Условие | Описание | | | |
| 1 | 2  3 | | |
| Описание | Пользователь регистрируется на маршрут | | | |
| Действующее лицо/ Роль | Пользователь: попутчик | | | |
| Предусловия | PRE-1 Выполнен вход в учетную запись.  PRE-2 Заполнены личные данные.  PRE-3 В базе данных существует маршрут по указанному направлени. | | | |
| Триггер | Пользователь указывает, что хочет зарегистрироваться на маршрут | | | |

Продолжение таблицы 2.6

|  |  |  |
| --- | --- | --- |
| 1 | 2  3 | |
| Основной сценарий | 1. Пользователь переходит в раздел «Найти маршрут». 2. Пользователь выбирает точку отправления. 3. На карте отображается точка отправления. 4. Пользователь выбирает точку направления. 5. На карте отрисовывается маршрут. 6. Пользователь нажимает «Найти». 7. Отображаются маршруты, соответствующие заданному направлению с подробной информацией о водителе, точках маршрута, автомобиле, стоимости, дате отправления, количестве мест. 8. Пользователь нажимает «Зарегистрироваться». 9. Система регистрирует пользователя на маршрут в базе данных, отображаются активные маршруты. | |
| Ожидаемые результаты | POST-1 На карте отображается точка отправления.  POST-2 На карте отрисовывается маршрут.  POST-3 Отображаются данные о подходящих маршрутах.  POST-4 Система вносит изменения в базу данных.  POST-5 Отображаются активные маршруты. |
| Исключения | И1 время на регистрацию истекло   1. Система выдает сообщение «Регистрация на маршрут окончена».   И2 нету свободных мест   1. Система выдает сообщение «Свободных мест нет». | |

Таблица 2.7 – Алгоритм отмены регистрации на маршрут

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-6 | Автор | Макатров В.Д. |
| Название варианта использования | Отмена регистрации на маршрут | Дата | 19.11.2020 |

Продолжение таблицы 2.7

|  |  |
| --- | --- |
| Условие | Описание |
| 1 | 2  3 |
| Описание | Пользователь отменяет регистрацию на маршрут |
| Действующее лицо/ Роль | Пользователь: попутчик |
| Предусловия | PRE-1 Выполнен вход в учетную запись.  PRE-2 Пользователь зарегистрирован на маршрут. |
| Триггер | Пользователь указывает, что хочет отменить регистрацию на маршрут |
| Основной сценарий | 1. Пользователь переходит в раздел «Активные маршруты». 2. Пользователь выбирает маршрут, на который хочет отменить регистрацию. 3. Пользователь нажимает кнопку «Отменить регистрацию». 4. Система вносит изменения в базе данных. |
| Ожидаемые результаты | POST-1 Отобразились активные маршруты.  POST-2 Изменения сохранены в базе данных.  POST-3 Отобразились обновленные активные маршруты. |

Таблица 2.8 – Алгоритм заполнения данных о водительском удостоверении

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-7 | Автор | Макатров В.Д. |
| Название варианта использования | Добавление данных о водительском удостоверении | Дата | 19.11.2020 |

Продолжение таблицы 2.8

|  |  |  |
| --- | --- | --- |
| Условие | Описание | |
| 1 | 2  3 |
| Описание | Пользователь заполняет данные о водительском удостоверении | |
| Действующее лицо/ Роль | Пользователь: водитель | |
| Предусловия | PRE-1 Выполнен вход в учетную запись под водителем.  PRE-2 Пользователь заполнил личные данные. | |
| Триггер | После того как пользователь заполнил личные данные, его перенаправляют на форму для заполнения данных о водительском удостоверении. | |
| Основной сценарий | 1. Отобразилась форма для заполнения данных о водительском удостоверении. 2. Пользователь заполняет данные о водительском удостоверении. 3. Пользователь нажимает кнопку «Сохранить». 4. Система вносит изменения в базе данных. 5. Система перенаправляет пользователя на форму заполнения данных о автомобиле. | |
| Ожидаемые результаты | POST-1 Отобразилась форма для заполнения данных о водительском удостоверении.  POST-2 Данные о водительском удостоверении сохранены в базе данных.  POST-3 Отобразилась форма заполнения данных о автомобиле. | |

Продолжение таблицы 2.8

|  |  |
| --- | --- |
| 1 | 2  3 |
| Исключения | И1 Дата получения прав позднее сегодняшнего дня   1. Система выдает сообщение «Ошибка! Введены некорректные данные».   И2 Срок действия прав ранее сегодняшнего дня или ранее даты получения прав   1. Система выдает сообщение «Ошибка! Введены некорректные данные». | |

Таблица 2.9 – Алгоритм добавления данных о автомобиле.

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-8 | Автор | Макатров В.Д. |
| Название варианта использования | Добавление данных о автомобиле | Дата | 19.11.2020 |
| Условие | Описание | | |
| 1 | 2  3 | | |
| Описание | Пользователь добавляет данные о автомобиле | | |
| Действующее лицо/ Роль | Пользователь: водитель | | |
| Предусловия | PRE-1 Выполнен вход в учетную запись.  PRE-2 Пользователь заполнил личные данные.  PRE-3 Пользователь заполнил данные о водительском удостоверении. | | |
| Триггер | После того как пользователь заполнил данные о водительском удостоверении, его перенаправляют на форму для заполнения данных о автомобиле. | | |

Продолжение таблицы 2.9

|  |  |
| --- | --- |
| 1 | 2  3 |
| Основной сценарий | 1. Отобразилась форма для заполнения данных о автомобиле. 2. Пользователь заполняет данные о автомобиле. 3. Пользователь нажимает кнопку «Сохранить». 4. Система вносит изменения в базе данных. 5. Система перенаправляет пользователя на форму со списком автомобилей. |
| Ожидаемые результат | POST-1 Отобразилась форма для заполнения данных о автомобиле.  POST-2 Данные о автомобиле сохранены в базе данных.  POST-3 Отобразилась форма со списком автомобилей. |

Таблица 2.10 – Алгоритм добавления маршрута

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-9 | Автор | Макатров В.Д. |
| Название варианта использования | Добавление маршрута водителем | Дата | 19.11.2020 |
| Условие | Описание | | |
| 1 | 2  3 | | |
| Описание | Пользователь регистрирует новый маршрут | | |
| Действующее лицо/ Роль | Пользователь: водитель | | |
| Предусловия | PRE-1 Выполнен вход в систему.  PRE-2 Заполнены личные данные.  PRE-3 Заполнены данные о водительском удостоверении.  PRE-4 Заполнены данные о автомобиле. | | |

Продолжение таблицы 2.10

|  |  |
| --- | --- |
| 1 | 2  3 |
| Триггер | Пользователь указывает, что хочет зарегистрировать новый маршрут |
| Основной сценарий | 1. Пользователь переходит в раздел «Добавление маршрута». 2. Пользователь выбирает точку отправления. 3. На карте отображается точка отправления. 4. Пользователь выбирает конечную точку/точку остановки. 5. На карте отрисовывается маршрут. 6. Пользователь заполняет данные о времени отправления, прибытия и стоимости указанного промежутка. 7. Пользователь нажимает кнопку «Далее». 8. Система отображает форму для заполнения дополнительных данных о маршруте. 9. Пользователь указывает автомобиль, на котором будет осуществляться поездка, дату окончания регистрации на маршрут и количество свободных мест в автомобиле. 10. Пользователь нажимает кнопку «Далее». 11. Система сохраняет данные о маршруте в базе данных. 12. Система перенаправляет пользователя на форму с активными маршрутами, на которой отображается созданный маршрут. |
| Ожидаемые результаты | POST-1 Отобразилась форма для добавления маршрута.  POST-2 На карте отобразилась точка отправления.  POST-3 На карте отрисовался маршрут.  POST-4 Отобразилась форма для заполнения дополнительных данных о маршруте.  POST-5 Данные о маршруте сохранены в базу данных.  POST-6 Отобразилась форма с активными маршрутами. |

Таблица 2.11 – Алгоритм удаления маршрута.

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор варианта использования | ВИ-10 | Автор | Макатров В.Д. |
| Название варианта использования | Удаление маршрута | Дата | 19.11.2020 |
| Условие | Описание | | |
| 1 | 2  3 | | |
| Описание | Пользователь удаляет маршрут | | |
| Действующее лицо/ Роль | Пользователь: водитель | | |
| Предусловия | PRE-1 Выполнен вход в систему.  PRE-2 В базе данных есть хотя бы один маршрут. | | |
| Триггер | Пользователь указывает, что хочет удалить маршрут | | |
| Основной сценарий | 1. Пользователь переходит в раздел «Активные маршруты». 2. Пользователь нажимает «Удалить». 3. Система вносит изменения в базу данных. 4. Система обновляет список активных маршрутов. | | |
| Ожидаемые результаты | POST-1 Отобразилась форма с активными маршрутами.  POST-2 Система внесла изменения в базу данных.  POST-3 Система обновила список активных маршрутов. | | |

Эргономические требования к СЧКС – это требования к системе в целом, ее отдельным подсистемам, оборудованию, рабочей среде, определяемые свойствами человека и устанавливаемые для обеспечения его эффективной и безопасной деятельности. Они формируются на основании экспериментальных исследований и опыта эксплуатации СЧКС, требований эргономических стандартов. Эргономические требования являются общими для систем определенного типа, которые предназначены для эксплуатации в одних и тех же условиях [16].

Соответствие системы каждому отдельному эргономическому требованию определяет единичный эргономический показатель ее качества, соответствие множеству требований той или иной группы определяет групповой эргономический показатель качества системы «человек-компьютер-среда» (гигиенический, антропометрический, физиологический, психофизиологический, психологический, социально-психологический). Групповые эргономические показатели определяют комплексные эргономические показатели, которые в совокупности обуславливают эргономические свойства системы (управляемость, обслуживаемость, усвояемость и обитаемость).

Эргономические требования предъявляют к различным элементам СЧКС: рабочим местам операторов, органам управления, средствам отображения и ввода информации и эксплуатационной документации [16].

Общий подход к разработке эргономических требований и их номенклатура определяется ГОСТ 20.39.108 – 85 Комплексная система общих технических требований. Требования по эргономике, обитаемости и технической эстетике. Номенклатура и порядок выбора [17].

Конкретные общие эргономические требования к системе «человек-компьютер-среда» установлены действующими нормативными документами.

Для точного считывания информации и комфортных условий ее восприятия работа с дисплеями должна проводиться при таких сочетаниях контраста и яркости изображения, внешней освещенности экрана, углового размера знака и угла наблюдения экрана, которые входят в оптимальные или предельно допустимые (при кратковременной работе) диапазоны.

Яркость знаков не должна быть менее 35 кд/ для дисплеев на электронно-лучевой трубке и не менее 20 кд/ для дисплеев с плоскими экранами. Неравномерность яркости рабочего поля экрана и яркости элементов знаков не должна быть более 20 %.

Яркостной контраст изображения, а также внутри знаков и между знаками должен быть не менее 3:1.

Ширина контура знака должна быть в пределах от 0,25 до 0,5 мм, а изменение размеров однотипных знаков на рабочем поле не должно превышать 5 %. Если в документации на дисплей не оговорено проектное расстояние наблюдения, то его принимают равным 50 см для дисплеев с размером экрана по диагонали 14–17`` и 75 см – для экранов 19–21`` [17].

Формы объектов должны соответствовать устойчивым зрительным ассоциациям, т.е. быть похожими на экране на формы реальных объектов. Для графической информации обязательно должны использоваться логические ударения, желательно их использовать также для текстово-графической информации [17].

Последовательность логических ударений должна соответствовать оптимальному порядку восприятия информации. Поля восприятия графической информации должны соответствовать оптимальному порядку изучения информации.

При этом поля восприятия имеют следующие размеры:

– поле точного восприятия: 3⁰ вверх-вниз, 7⁰ вправо-влево или 2,6–2,7 см, 4,8–5,2 см от оси зрения;

– поле опознания расположения: вверх 25⁰, вниз 35⁰, вправо и влево по 32⁰ или 24–28 см, 34–40 см, 31–37 см от оси зрения;

– поле высокозначимой информации: 15⁰ или 14–16 см во все стороны от оси зрения;

– поле главного объекта: 10⁰ или 9–10 см во все стороны от оси зрения.

Должно обеспечиваться соответствие пространственного расположения информации на экране оптимальному порядку изучения.

Степень засоренности поля главного объекта не должна быть большой: превышать 4–6 второстепенных объектов в поле главного объекта [16].

Надписи, обозначающие объекты или органы управления должны быть краткими, однозначно воспринимаемыми и читаться слева направо. Допускается использование только тех слов, которые хорошо известны пользователю. Сокращение слов нежелательно, в крайнем случае, можно использовать только стандартные сокращения.

Параметры предъявляемого на экране текста должны удовлетворять следующим требованиям:

– высота знака – не менее 3 мм;

– отношение ширины буквы, цифры к высоте – в пределах 0,76–0,80;

– толщина линии обводки в прямом контрасте – в пределах 10–15 % от высоты знака;

– в обратном контрасте – в пределах 12–16 % от высоты знака;

– расстояние между знаками – не менее 30 % от высоты знака;

– расстояние между строками – 1,5–2 высоты знака;

– длина строки – 40–80 знакомест [17].

## 2.3 Разработка сценария информационного взаимодействия и оценка эргономичности ее пользовательского интерфейса

В результате эргономического проектирования был разработан пользовательский интерфейс, который удовлетворяет условиям технического задания.

Запуск приложения осуществляется открытием браузера и переходом на сайт приложения. Пользователь видит страницу авторизации.

Следующим шагом является авторизация пользователя в системе. Для этого пользователь вводит логин и пароль. В зависимости от введенных данных пользователи получат доступ к тем или иным страницам системы. Форма авторизации представлена на рисунке 2.1.

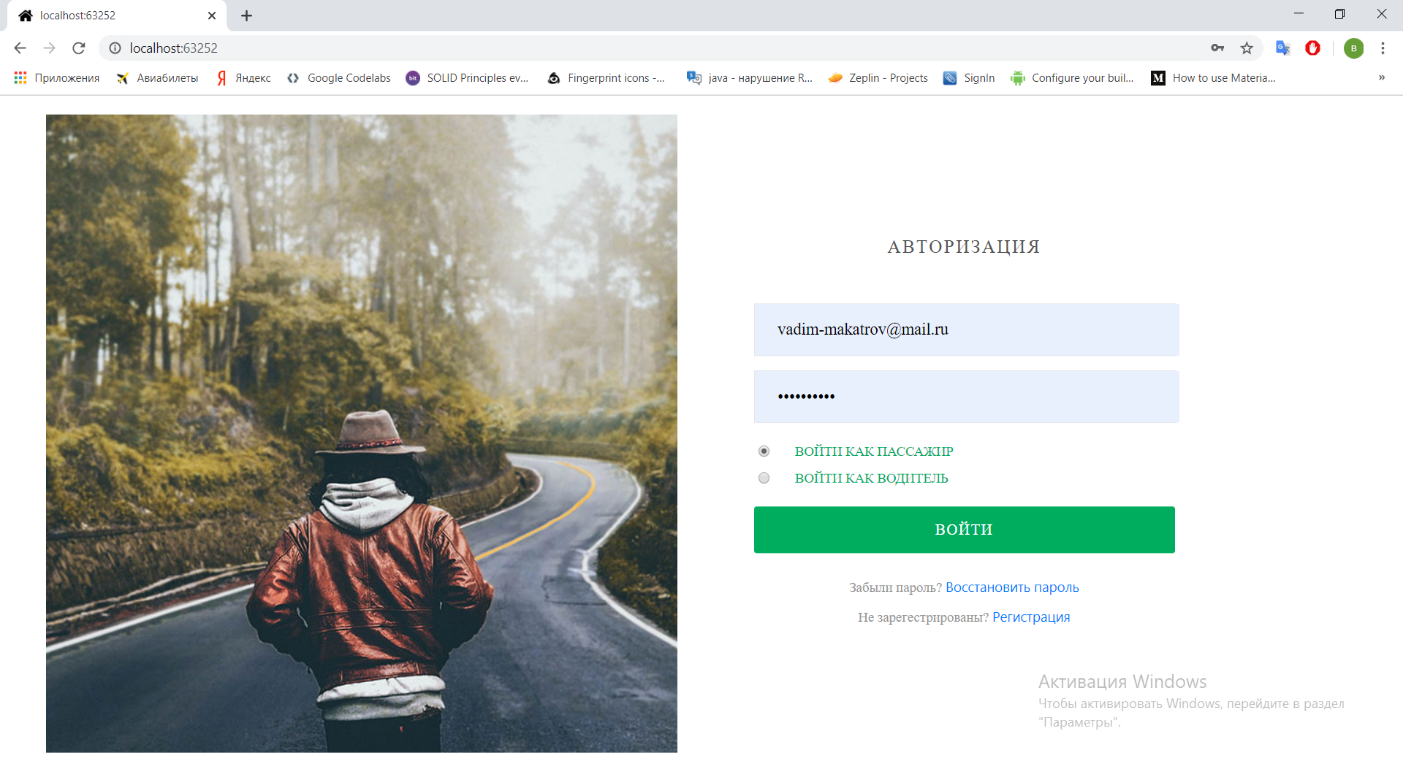


Рисунок 2.1 – Авторизация

При неверно введенных данных под полями появляется сообщение об ошибке.

Далее сценарий работы с системой определяется исходя из роли попутчика или водителя. Ниже представлен алгоритм регистрации на маршрут попутчиком.

После авторизации попутчику отображается экран с маршрутами, на которые он зарегистрирован. Если пользователь зарегистрирован на маршруты, на данной форме будет отображена краткая информация о их. Если пользователь не зарегистрирован ни на один маршрут – отобразится соответствующая фраза. Также на форме находится кнопка для перехода на форму поиска маршрута, страница представлена на рисунке 2.2.

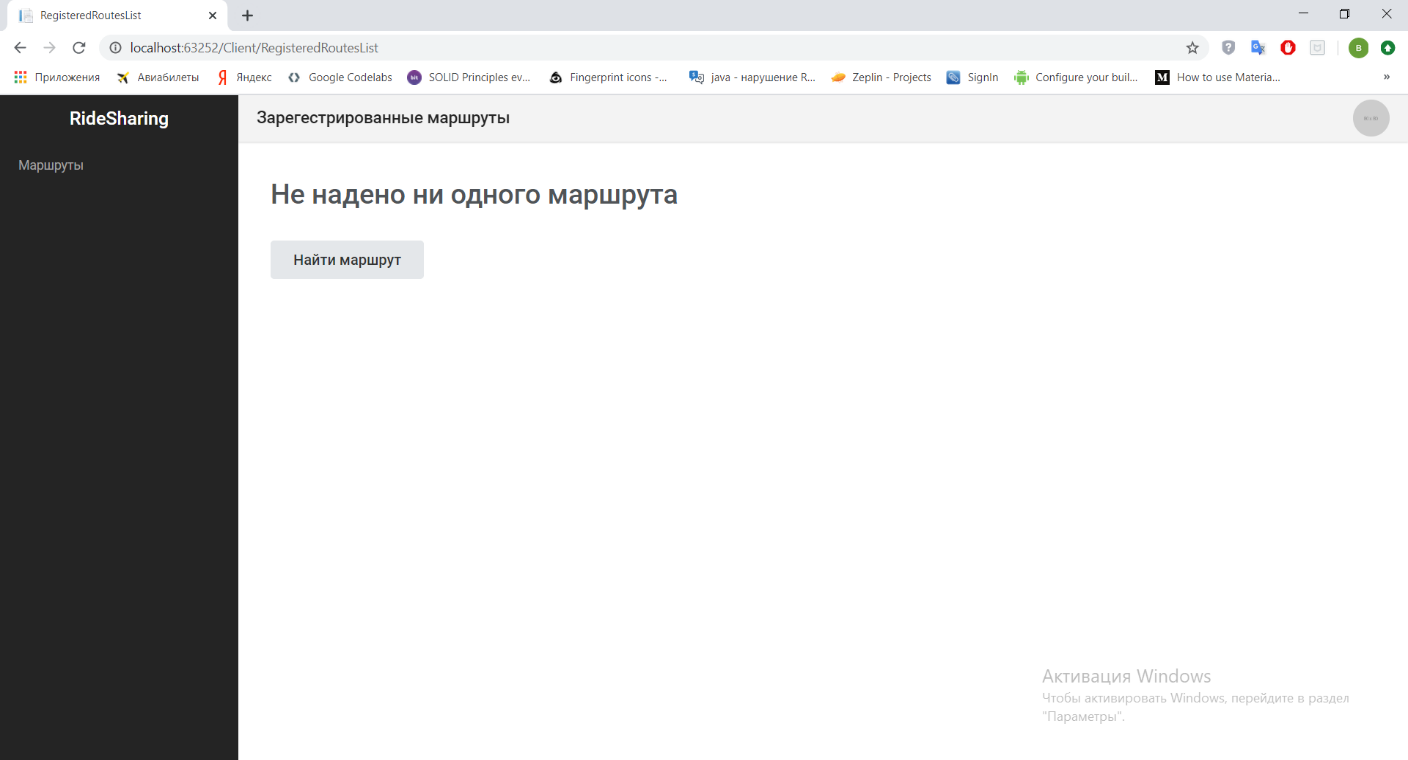


Рисунок 2.2 – Главная страница попутчика

При нажатии кнопки «Найти маршрут», происходит проверка заполнены ли у пользователя личные данные.

Если попутчик ранее не заполнял личные данные, его перенаправляют на экран для заполнения личных данных, после заполнения личных данных отобразится страница поиска маршрута, на которой необходимо выбрать интересующие точки маршрута. Если у пользователя заполнены личные данные ему сразу отобразится экран с маршрутами, на которые он зарегистрирован. Страница представлена на рисунке 2.3.

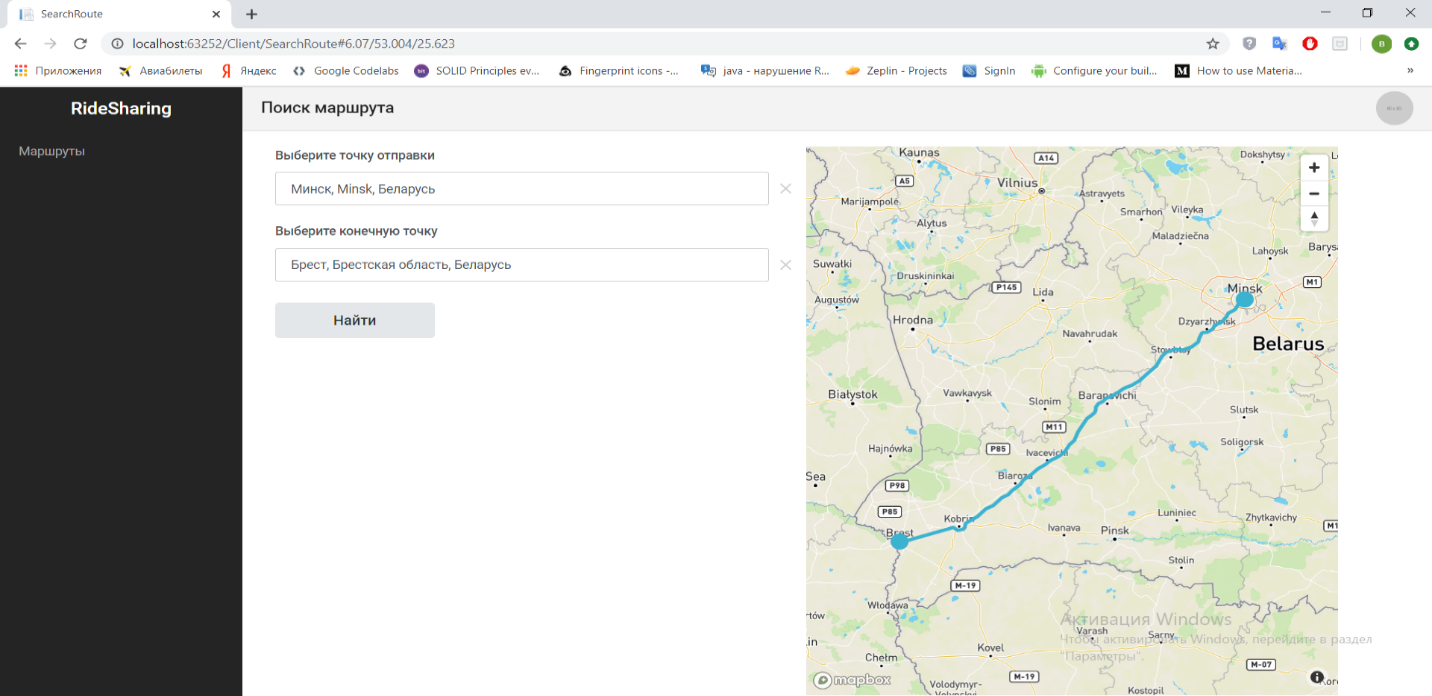


Рисунок 2.3 – Страница поиска маршрута

При нажатии кнопки «Найти», пользователя направляют на страницу с маршрутами, которые удовлетворяют критериям поиска. Страница представлена на рисунке 2.4. На данной странице есть возможность просмотра информации о водителе, автомобиле, точках маршрута, а также можно зарегистрироваться на маршрут. Страница с открытыми о водителе, автомобиле, точках маршрута данными отображена на рисунке 2.5.

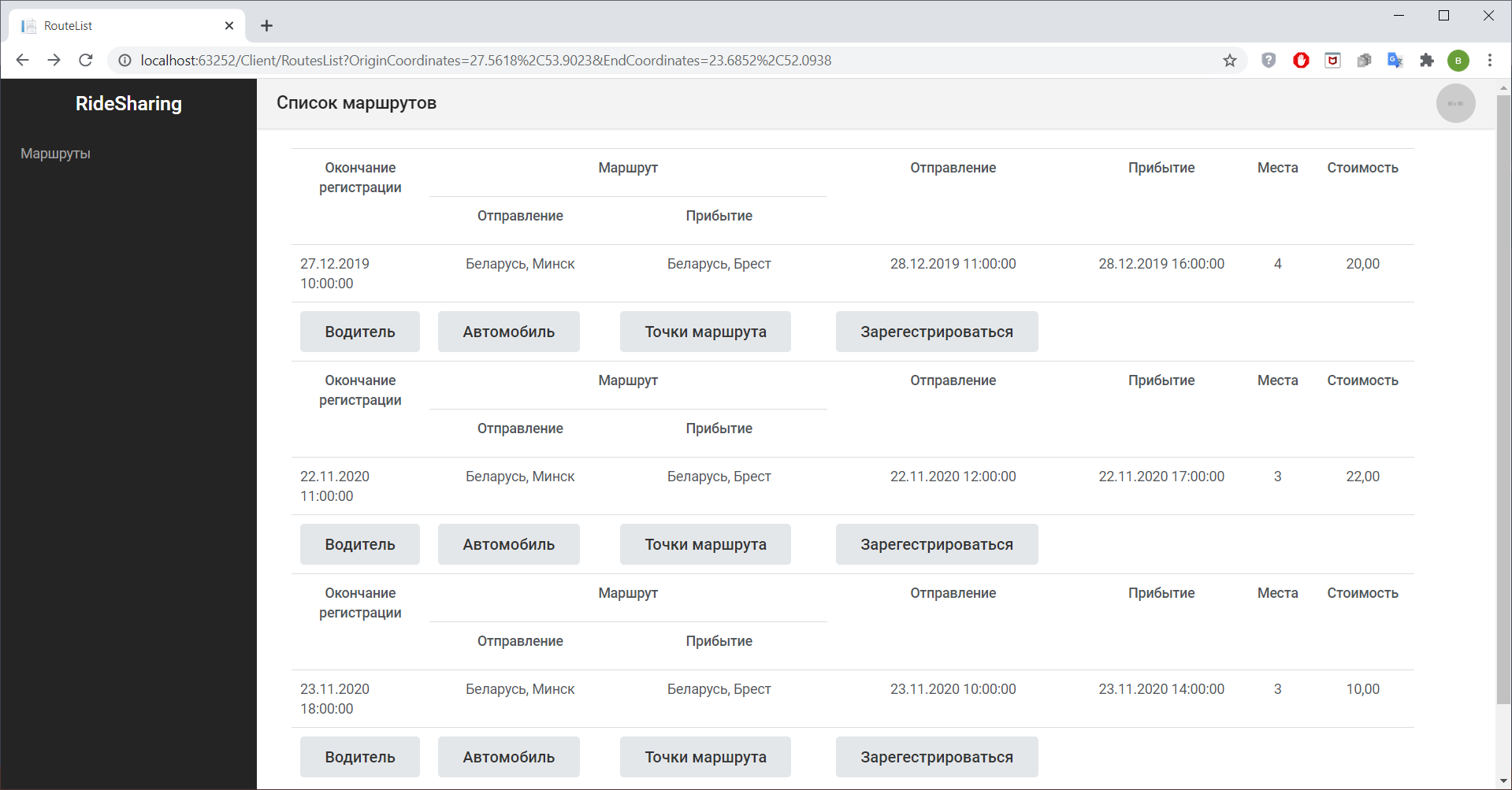


Рисунок 2.4 – Страница выбора маршрута

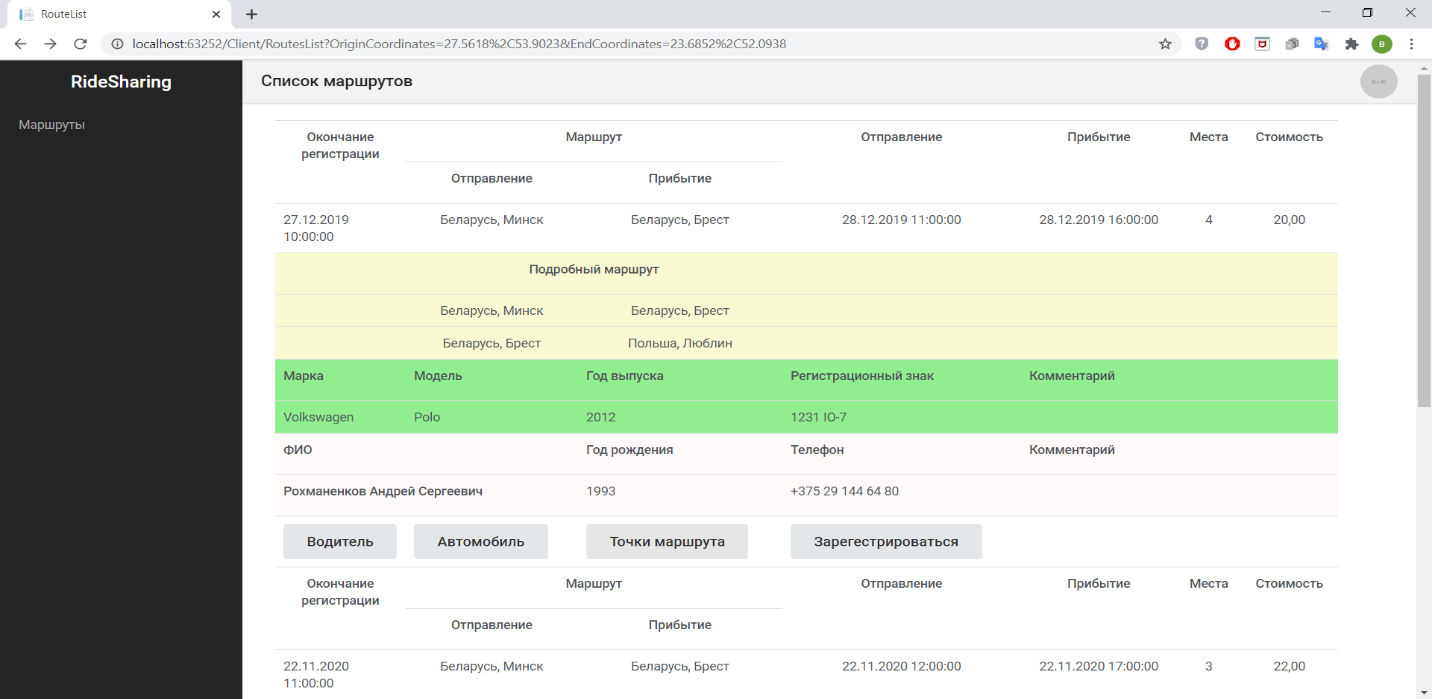


Рисунок 2.5 – Страница выбора маршрута, с развернутыми о водителе, автомобиле, точках маршрута данными

При нажатии кнопки «Зарегистрироваться», пользователя направляют на страницу с активными маршрутами, на котором будет отображен выбранный маршрут, и так же появится кнопка «Отменить регистрацию» при нажатии на которую можно отменить регистрацию на маршрут. Страница представлена на рисунке 2.6.

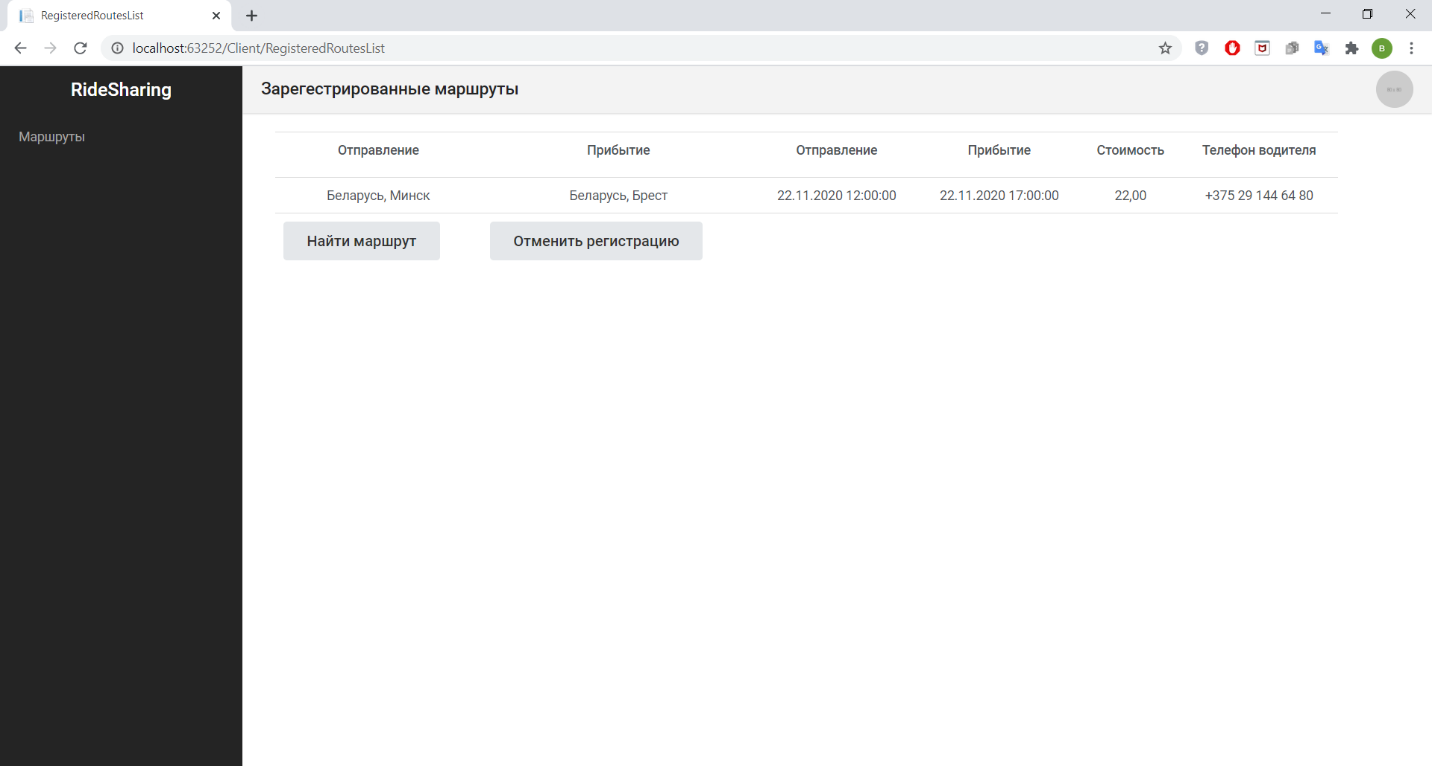


Рисунок 2.6 – Страница с активными маршрутами попутчика

Если пользователь авторизуется как водитель, первым делом система проверит заполнены ли у его личные данные, данные о водительском удостоверении, данные о автомобиле, если перечисленные данные не заполнены, пользователя перенаправят на формы для их заполнения. Если указанные данные заполнены, будет отображен экран с активными маршрутами водителя, на котором пользователь сможет посмотреть информацию о маршруте, посмотреть информацию о зарегистрированных на маршрут попутчиках, удалить маршрут или завершить маршрут. Страница представлена на рисунке 2.7. Также, как и в случае с попутчиком, если у водителя нету активных маршрутов – будет отображена соответствующая фраза.

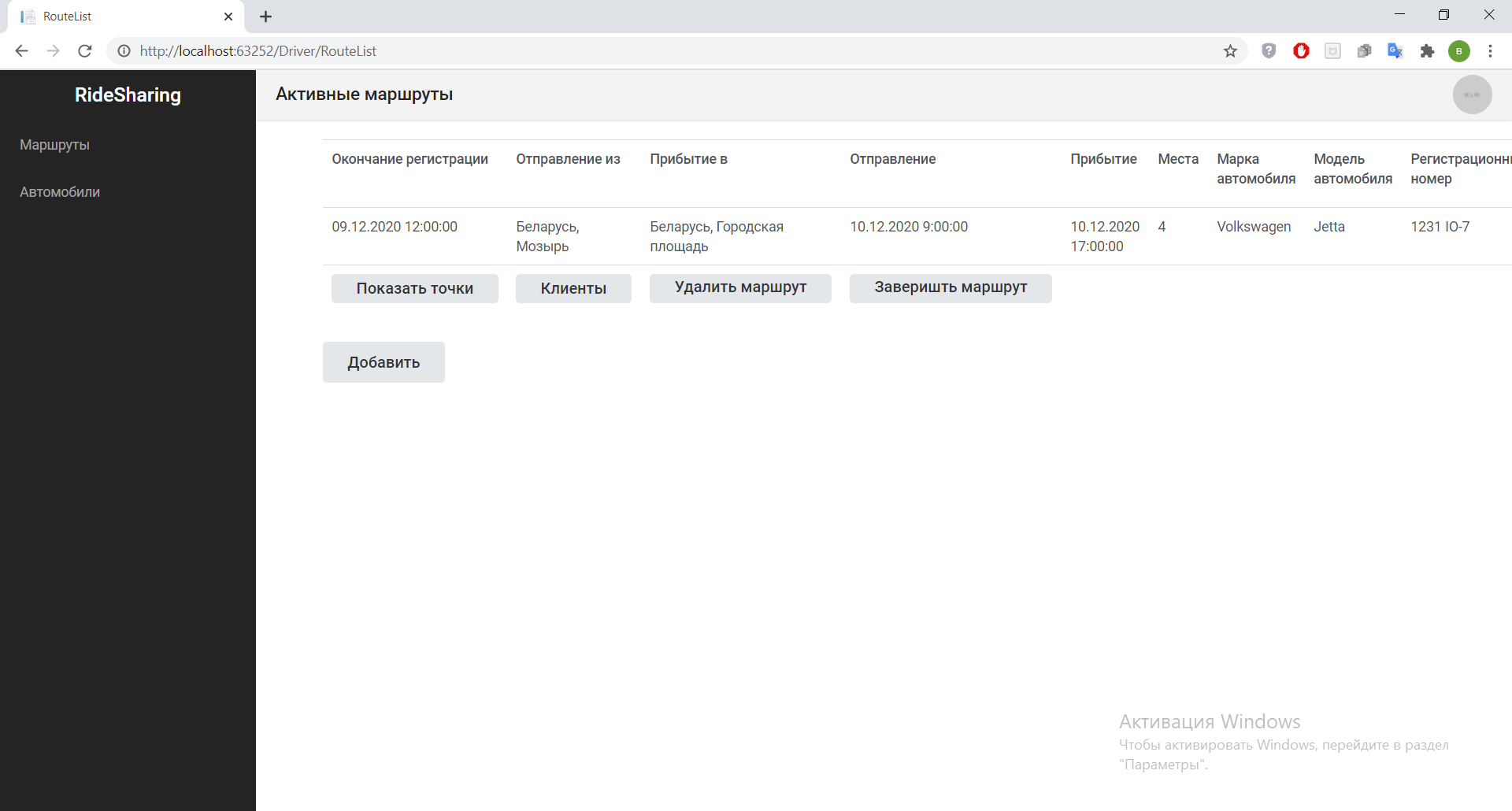


Рисунок 2.7 – Страница с активными маршрутами водителя

При нажатии кнопки «Добавить», пользователя перенаправят на страницу выбора точек маршрута, на которой необходимо выбрать интересующие точки маршрута, а также указать стоимость и дату с временем, для отправления и прибытия. Страница представлена на рисунке 2.8. Затем при нажатии кнопки «Далее» будет отображена форма заполнения дополнительных данных о маршруте, на которой необходимо указать дату и время окончания регистрации на маршрут, выбрать автомобиль, на котором будет осуществляться поездка и указать количество свободных в автомобиле мест. Страница представлена на рисунке 2.9.

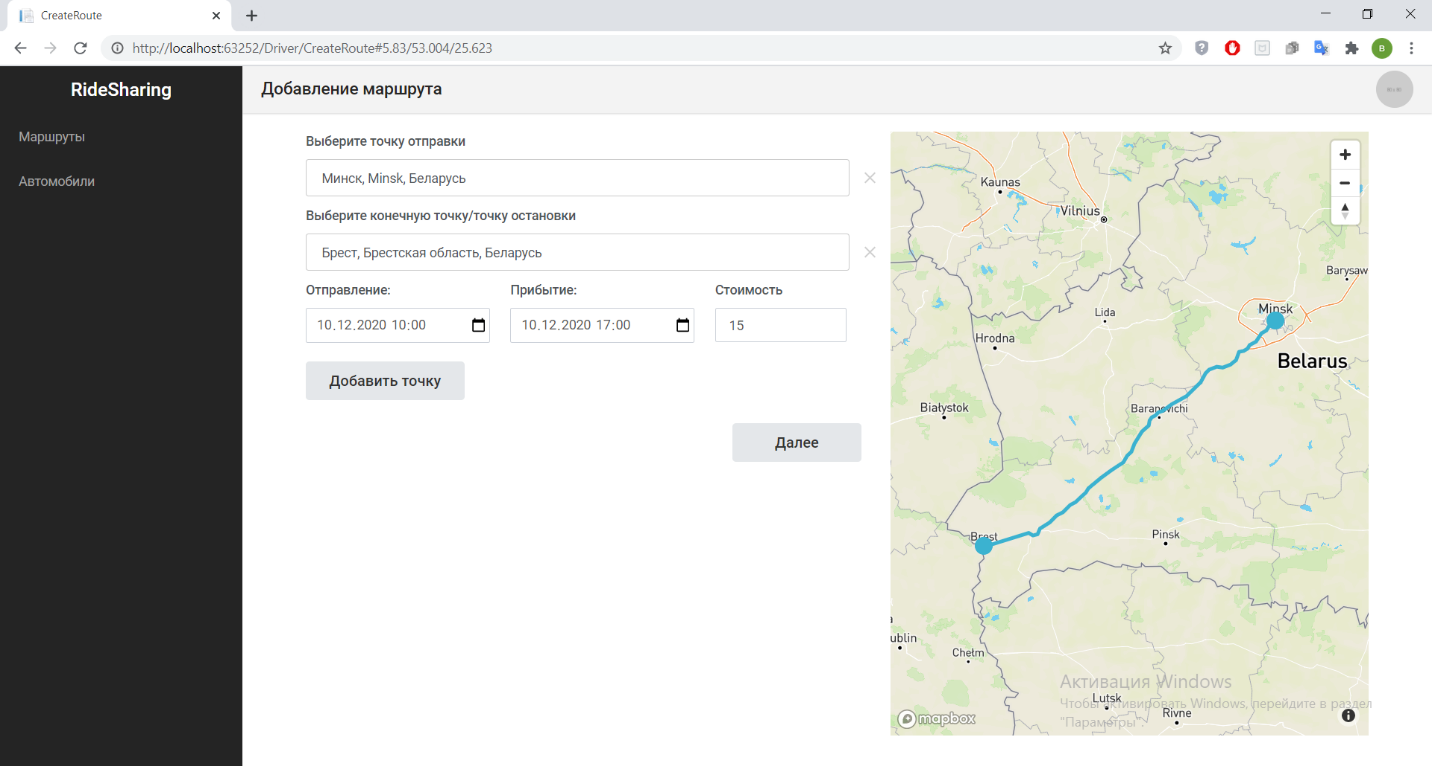


Рисунок 2.8 – Страница выбора точек маршрута водителем

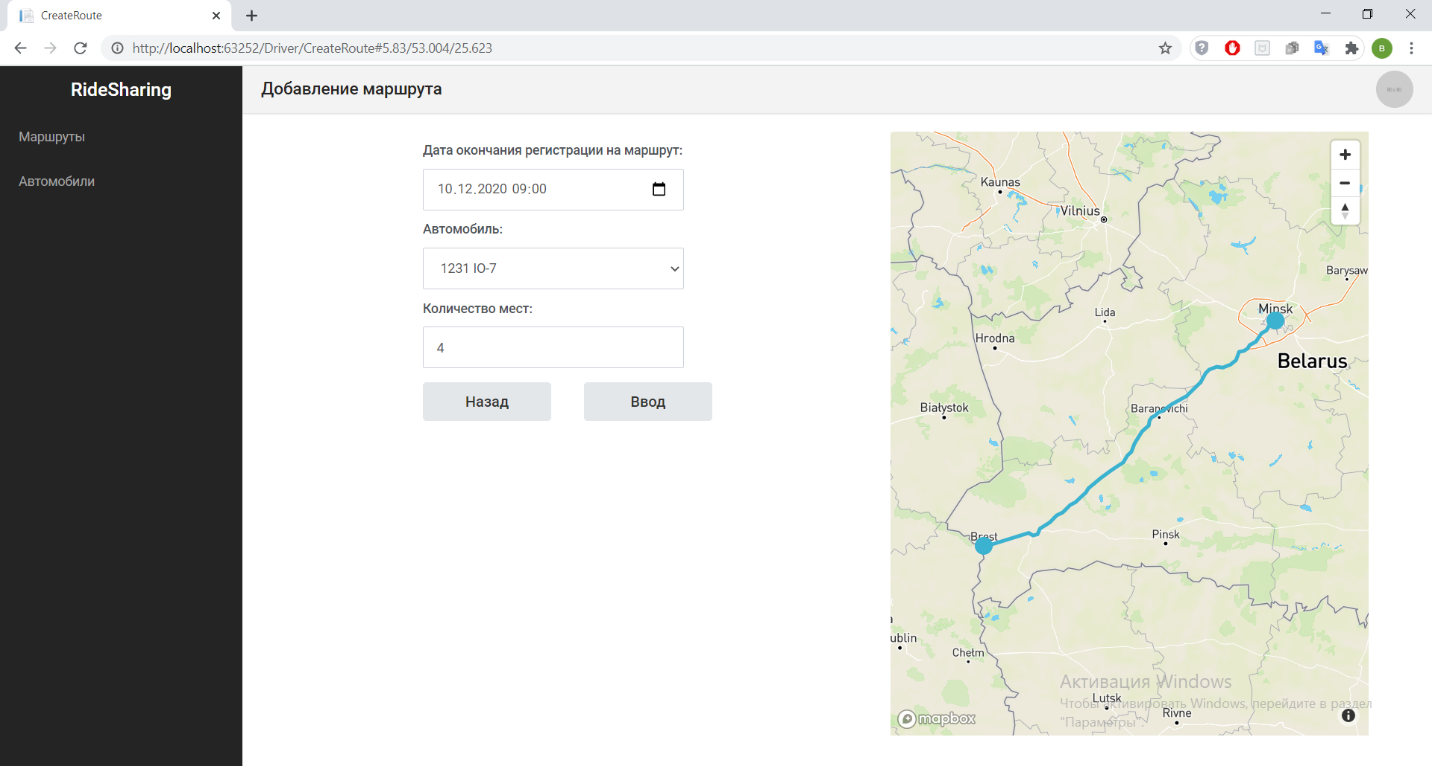


Рисунок 2.9 – Страница заполнения дополнительных данных о маршруте

Затем при нажатии кнопки «Ввод» пользователя направляют на страницу с активными маршрутами, на котором отображаются все активные маршруты. Страница была представлена на рисунке 2.7.

Под эргономическими требованиями понимаются требования к СЧМС (система «человек-машина-среда»), ее отдельным подсистемам и рабочей среде, определяемые свойствами человека и устанавливаемые для обеспечения его комфортной, эффективной и безопасной деятельности [16].

Эргономические требования определяются свойствами человека-оператора и устанавливаются с целью оптимизации его деятельности, они являются базовыми при проектировании СЧМС на основе антропоцентрического подхода. Эргономические требования необходимы для обеспечения:

* рационального распределения функций в СЧМ (система «человек-машина») [16];
* оптимальной организации рабочего места человека-оператора на основе учета в конструкции рабочих характеристик и свойств человека [16];
* соответствия технических средств возможностям человека-оператора по приему и переработке информации и осуществлению управляющих воздействий [16];
* оптимальных для жизнедеятельности и работоспособности человека показателей производственной среды [16].

Эргономические требования для СЧКС (система «человек-компьютер-среда»), разрабатываемой в данном дипломном проекте, представлены в таблице 2.12.

Данный программный продукт является веб-ресурсом, поэтому основными эргономическими требованиями являются требования к информации, предъявляемой на дисплее.

Таблица 2.12 – Эргономические требования к проектируемой СЧКС

|  |  |
| --- | --- |
| Группа эргономических требований | Номенклатура эргономических требований |
| 1 | 2 |
| Психологические | Возможность отмены и корректировки действий.  Предъявление сообщений об ошибочных действиях пользователя. |

Продолжение таблицы 2.12

|  |  |
| --- | --- |
| Психологические | Наличие предупреждений о нежелательных последствиях некоторых действий.  Единый визуальный стиль всех экранов приложения.  Одно и тоже расположение элементов управления на разных окнах.  Использование для названия заголовков меню одного слова (глагола для действий, существительного для объектов).  Наличие кратких и понятных заголовков. |
| Психофизиологические | Соответствие размера надписей и знаков принципам восприятия.  Соответствие контраста знаков и фона особенностям восприятия.  Оптимальный размер знаков на экране.  Соответствие расположения надписей условиям их оптимального считывания.  Соответствие размеров текста в зависимости от его важности. |
| Физиологические | Отсутствие жестов управления, не соответствующим физиологическим возможностям пользователей.  Соответствие компоновки элементов управления принципам экономии рабочих движений.  Отсутствие необходимости устанавливать фокус ввода в открывающихся текстовых полях. |
| Социально-психологические | Разграничение прав пользователей разных типов. |
| Антропометрические | Отсутствие действий управления системой, выходящих за пределы антропометрических возможностей человека. |

Эргономическая оценка инженерных решений − это комплекс научно-технических и организационно-методических мероприятий по оценке выполнения в проектных документах и в образцах СЧМ эргономических требований технического задания, нормативно-технических и руководящих документов, а также разработка рекомендаций для устранения отступлений от этих требований. Указанная оценка проводится при обосновании выполнения каждого этапа опытно-конструкторской разработки: технического предложения, эскизного проекта, рабочего проекта [16].

Для оценки степени соответствия характеристик конкретной СЧМ эргономическим требованиям могут применяться экспериментальные, расчётные и экспертный методы [16]. Эргономическая оценка данной СЧКС будет проведена экспертным методом. В качестве эксперта выступит разработчик этой СЧКС.

Для данной системы проведем оценку свойства управляемость. Рассмотрим следующие группы эргономических требований: физиологические, антропометрические, психологические и психофизиологические. Полученные результаты представлены   
в таблице 2.13.

Таблица 2.13 – Общие эргономические требования к проектируемой СЧКС и соответствующие им единичные эргономические показатели

|  |  |  |
| --- | --- | --- |
| Группа эргономических требований | Эргономическое требование | Единичный эргономический показатель |
| 1 | 2 | 3 |
| Психофизиологические | ПФ-1. Соответствие размеров знаков на экране дисплея оперативному порогу зрения человека | Размеры шрифта текста и знаков |
| ПФ-2. Соответствие контраста знаков и фона оптимальным условиям восприятия | Величина контраста знаков и фона |
| ПФ-3. Соответствие вида контраста знаков и фона уровню освещенности рабочего места | Вид контраста знаков и фона |

Продолжение таблицы 2.13

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Психофизиологические | ПФ-4. Отображение недоступных пунктов меню хорошо различимым блеклым цветом | Цвет недоступных пунктов меню |
| ПФ-5. Соответствие расположения надписей условиям их оптимального считывания | Расположение и ориентация надписей на экране дисплея |
| ПФ-6. Использование пролистываемых и раскрывающихся списков в целях экономии экранного пространства | Наличие и типы пролистываемых и раскрывающихся список |
| Психологические | П-1. Соответствие сложности инструкций, времени, отводимому на их восприятие | Длина инструкции и время ее экспозиции |
| П-2. Один и тот же характер команд на протяжении всего периода работы в системе в схожих ситуациях | Тип ОУ и их обозначение |
| П-3. Наличие указаний на проблемы, возникающие в процессе обслуживания системы | Сообщения об ошибочных действиях пользователей |
| П-4. Наличие подсказок о следующих шагах работы в системе | Сообщения о следующих действиях пользователей |

Продолжение таблицы 2.13

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Психологические | П-5. Наличие предупреждений о нежелательных последствиях некоторых действий | Предупреждения о возможных нежелательных действиях |
| П-6. Соответствие цветов знаков и надписей сформированным стереотипам восприятия цвета | Цвета знаков, кнопок, надписей |
| П-7. Соответствие формы и расположения знаков сформированным стереотипам восприятия | Форма и ориентация знаков |
| П-8. Выделение в текстовых инструкциях смысловых фрагментов | Компоновка текста инструкции (наличие абзацев) или других способов выделения |
| П-9. Отсутствие в текстовых сообщениях аббревиатур, непонятных слов и сокращений | Словарный состав текстовых инструкций |
| П-10. Привлечение внимания пользователей к важным сообщениям | Используемые средства привлечения внимания пользователя (цвет, мигание, звуковые сигналы) |

Продолжение таблицы 2.13

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Психологические | П-11. Наличие индикатора степени выполнения заданий (операций) | Наличие и вид индикатора выполнения |
| П-12. Наличие кратких и понятных заголовков окон | Наличие и вид заголовков окон |
| П-13. Использование для названий пунктов меню одного слова (глагола для действий, существительного для объектов) | Названия пунктов меню |
| П-14. Применение в названиях пунктов меню норм использования заглавных букв, принятых в языке | Названия пунктов меню |
| П-15. Соответствие опций элементов интерфейса установленным, привычным нормам (например, использование клавиши Enter) | Соответствие привычным нормам |
| П-16. Отсутствие у пользователей сложностей в поиске необходимых директив (элементов интерфейса) для управления процессом решения поставленной задачи | Соответствие привычным нормам |

Продолжение таблицы 2.13

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Психологические | П-17. Сообщение об ошибке должно отвечать всего на три вопроса: – в чем заключается проблема? – как исправить эту проблему сейчас? – как сделать так, чтобы проблема не повторилась? | Содержание сообщений об ошибках |
| П-18. Вежливое и понятное пользователю сообщение об ошибках | Содержание сообщений об ошибках |
| П-19. К строкам ввода там, где это возможно, с целью разгрузки памяти целесообразно присоединять выпадающий список допустимых значений | Наличие выпадающих списков допустимых значений в строках ввода |
| П-20. Целесообразно использовать в рамках одного приложения окна, построенные по одному шаблону, в которых одинаковые элементы расположены одинаково | Окна интерфейса в программы |
| П-21. Интерфейсные элементы должны иметь не только согласованные изображения, но и согласованное управление. Например, активизация всех пиктограмм – двойным щелчком мыши. | Средства управления элементами интерфейса |

Продолжение таблицы 2.13

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Психологические | П-22. Следует учитывать при проектировании меню и диалоговых окон логическую последовательность чтения текста справа налево и сверху вниз. В левом верхнем углу следует располагать элемент, с которым пользователь должен работать в первую очередь, а в правом нижнем углу – тот, который используется в конце. | Компоновка опций меню и диалоговых окон |
| Физиологический | Ф-1. Соответствие размеров зон установки курсора физиологическим возможностям движений | Размеры меню, списков, кнопок на экране дисплея |
| Ф-2. Использование в группе радиокнопок не менее одной с режимом по умолчанию | Наличие в группе радиокнопок не менее одной с режимом по умолчанию |
| Ф-4. Использование чекбоксов и радиокнопок для ввода учебного занятия | Наличие чекбоксов и радиокнопок |
| Ф-5. Отсутствие необходимости устанавливать фокус ввода в открывающихся текстовых полях | Наличие фокуса ввода в текстовых полях по умолчанию |
| Ф-6. Соответствие времени экспозиции списков, меню, кнопок скоростным возможностям человека | Длительность экспозиции средств взаимодействия |

Продолжение таблицы 2.13

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Физиологический | Ф-7. Использование крутилок для ввода числовых значений | Наличие крутилок для ввода числовых значений |
| Ф-8. Использование ползунков(слайдеров) для ввода ранжирующихся значений | Наличие слайдеров для ввода ранжирующихся значений |
| Ф-9. Использование значения по умолчанию, где только возможно, чтобы минимизировать процесс ввода информации. | Используемые значения по умолчанию |
| Ф-10. Отсутствие требований к пользователям вводить информацию, которая была предварительно введена или которая может быть автоматически получена из системы | Отсутствие необходимости вводить информацию, которая была ранее введена или которая может быть автоматически получена из системы |
| Гигиенические | Г-1. Соответствие изображения на экране дисплея условиям комфорта зрительной работы пользователей (отсутствие мельканий, слепящих яркостей и т.п.) | Энергетические и временные изображения на экране дисплея |

Продолжение таблицы 2.13

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Социально-психологические | СП-1. Отсутствие условий для возникновения конфликтов интересов или действий пользователей разных типов | Способ разграничения прав пользователей разных типов |

После построения таблицы проведем оценку значений единичных эргономических показателей. Указанные показатели оцениваются по бинарной шкале с числовым диапазоном от 0 до 1. Показатель принимает значение, равное 1, если фактическое его значение соответствует рекомендуемому. Если соответствия нет (т.е. требование фактически не выполняется), то показатель принимается равным 0.

Групповой эргономический показатель ЭПгр рассчитывается как общая оценка по группе единичных показателей:

ЭПгр = ∑ 1 ∕ (∑ 1 + ∑ 0) , (2.1)

где ∑ 1– суммарное число случаев, когда единичный показатель соответствует эргономическому требованию;

∑ 0 – суммарное число случаев, когда соответствия нет [16].

Результаты оценки значений единичных и групповых эргономических показателей приведены в таблице 2.14.

Таблица 2.14 – Значения единичных и групповых эргономических показателей проектируемой системы

|  |  |  |
| --- | --- | --- |
| Группа ЭП | Значение единичных ЭП | Значение групповых ЭП |
| 1 | 2 | 3 |
| Психофизиологические | ПФ-1, ПФ-2, ПФ-3, ПФ-4, ПФ-5, ПФ-6 = 1 | 6 / (6 + 0) = 1 |

Продолжение таблицы 2.14

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Психологические | П-1, П-2, П-3, П-6, П-7, П-9, П-10, П-12, П-13, П-14, П-15, П-16, П-19, П-20, П-21, П-22 = 1  П-4, П-5, П-8, П-11, П-17, П-18=0 | 15 / (15 + 6) =  = 0,71 |
| Физиологические | Ф-1, Ф-2, Ф-3, Ф-4, Ф-5, Ф-6, Ф-7, Ф-9 = 1  Ф-8, Ф-10 = 0 | 8 / (8 + 2) = 0,8 |
| Гигиенические | Г-1 = 1 | 1 / (1 + 0) = 1 |
| Социально-психологические | СП-1=1 | 1 / (1 + 0) = 1 |
| Антропометрические | Не актуальны для данной СЧМ | – |

Проведем оценку эргономических свойств проектируемой СЧКС. Эргономические свойства СЧМ определяются как некоторая совокупность групповых эргономических показателей. Применим аддитивную функцию:

ЭСВ = ∑ ∙ ЭПгрj , (2.2)



где – нормированные весовые коэффициенты, сумма которых должна быть равна 1, т. е**.** ( = 1) [16]**.**

Для оцениваемого эргономического свойства выбираем величины весовых коэффициентов. Результаты распределения весовых коэффициентов представлены в таблице 2.15.

Таблица 2.15 – Значения весовых коэффициентов групп эргономических требований и эргономических свойств данной СЧКС

|  |  |
| --- | --- |
| Группа эргономических требований | Значение весового коэффициента |
| 1 | 2 |
| Психофизиологическая | 0,25 |
| Психологическая | 0,6 |
| Физиологическая | 0,05 |
| Гигиенические | 0,05 |
| Социально-психологические | 0,05 |
| Антропометрическая | – |

На основе таблиц 2.14 и 2.15, определим количественное значение указанных выше эргономических свойств. Получим:

ЭСВуправляемость = 0,25 ⋅ 1 + 0,6 ⋅ 0,71 + 0,05 ⋅ 0,08 + 0,05 ⋅ 1 + 0,05 ⋅ 1 =   
= 0,78

Показатель эргономичности разрабатываемой системы 0,78. Полученное значение группового эргономического показателя оценивается с учетом следующей градации:

* 0,8–1,0 – «отлично» – эргономические характеристики изделия соответствуют базовым значениям;
* 0,5–0,8 – «хорошо» – приближается к базовым, но требуется совершенствование изделия;
* 0,2–0,5 – «удовлетворительно» – далеки от базовых, требуется значительное улучшение изделия;
* 0–0,2 – «неудовлетворительно» – практически не обеспечивается необходимая производительность, удобство и безопасность труда человека – оператора [24].

Согласно проведённой оценке эргономичности системы, уровень эргономичности данной СЧКС оценивается отметкой «хорошо», что означает выполнение базового функционала, однако в системе имеется ряд недостатков, выявленных при проведении экспертизы. Для устранения недостатков системы составлены рекомендации по улучшению эргономичности, проектируемой СЧКС.

Таблица 2.14 – Рекомендации по улучшению эргономичности, проектируемой СЧКС

|  |  |
| --- | --- |
| Невыполненное эргономическое требование | Предложение по улучшению эргономичности |
| 1 | 2 |
| П-4. Наличие подсказок о следующих шагах работы в системе | Добавить подсказки о следующих шагах работы |

Продолжение таблицы 2.14

|  |  |  |
| --- | --- | --- |
| 1 | 2 | |
| П-5. Наличие предупреждений о нежелательных последствиях некоторых действий | Добавить предупреждения о нежелательных последствиях |
| П-8. Выделение в текстовых инструкциях смысловых фрагментов | Выделить смысловые фрагменты в текстовых инструкциях |
| П-11. Наличие индикатора степени выполнения заданий (операций) | Добавить индикаторы оформления маршрута |
| П-17. Сообщение об ошибке должно отвечать всего на три вопроса: – в чем заключается проблема? – как исправить эту проблему сейчас? – как сделать так, чтобы проблема не повторилась? | На данный момент в системе используются общие ошибки, добавить текста ошибок для каждого поля |

# 3 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ СОВМЕСТНОГО ИСПОЛЬЗОВАНИЯ ЧАСТНОГО АВТОМОБИЛЯ

## 3.1 Разработка структуры информационной системы

Для разработки программного продукта был выбран язык C# Asp.Net.MVC фраемворк. На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей [18].

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 8.0, которая вышла в сентябре 2019 года вместе с релизом .NET Core 3 [18].

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче [18].

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей, как, например, лямбды, динамическое связывание, асинхронные методы и т.д [18].

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире [18].

Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

1. Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) – своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках [18].
2. Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Core поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ – Windows, MacOS, Linux, Android, iOS, Tizen [18].
3. Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# – текстовый редактор, чат или сложный веб-сайт – так или иначе мы задействуем библиотеку классов .NET [18].
4. Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET и Entity Framework Core. Для построения графических приложений с богатым насыщенным интерфейсом – технология WPF и UWP, для создания более простых графических приложений – Windows Forms. Для разработки мобильных приложений – Xamarin. Для создания веб-сайтов – ASP.NET и т.д [18].
5. Автоматическая сборка мусора. Это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память [18].

Для дальнейшей масштабируемости проекта была выбрана многоуровневая архитектура, на конечном этапе проект будет иметь 4 слоя работы с данными:

1. Presentation layer (уровень представления): это тот уровень, с которым непосредственно взаимодействует пользователь. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя. Применительно к ASP.NET MVC на данном уровне расположены представления и все те компоненты, который составляют пользовательский интерфейс (стили, статичные страницы html, javascript), а также модели представлений, контроллеры, объекты контекста запроса [20].
2. Business layer (уровень бизнес-логики): содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки [20].
3. Data Access layer (уровень доступа к данным): хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, например, класс контекста данных Entity Framework. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных [20].
4. Common layer (уровень помощников): хранит различные перечисления, списки, классы помощники, которые могут использоваться на каждом из слоев.

Чтобы четко разделять ответственность за различное функционирование в приложении был использован паттерн Model-View-Controller. MVC (Model-View-Controller) является фундаментальным паттерном, который нашел применение во многих технологиях, дал развитие новым технологиям и каждый день облегчает жизнь разработчикам [21].

Впервые паттерн MVC появился в языке SmallTalk. Разработчики должны были придумать архитектурное решение, которое позволяло бы отделить графический интерфейс от бизнес логики, а бизнес логику от данных [21].

Таким образом, в классическом варианте, MVC состоит из трех частей, которые и дали ему название:

* модель;
* представление (View);
* контроллер.

Для облегчения понимания принципов работы и описания, данную информационную систему можно разделить на несколько модулей:

– модуль авторизации;

– модуль водителя;

– модуль попутчика.

Остановимся подробно на каждом из них.

Модуль авторизации обеспечивает контроль учетных записей пользователей, а также содержит в себе методы авторизации, регистрации, верификации, а также метод восстановления доступа в учетную запись.

Рассмотрим работу данного модуля на примере авторизации пользователя.

На рисунке 3.1 приведен код авторизации пользователя. Получив данные из формы приложение обращается к сервису валидации, уровня BLL, на котором проверяется корректность введенных данных, код сервиса валидации приведен на рисунке 3.2. Затем, если все данные введены верно, приложение обращается к сервису, работающему с данными о учетных записях, уровня BLL. В данном сервисе приложение посредством DAL уровня получает данные из базы данных (код метода представлен   
на рисунке 3.3). Затем после некоторых манипуляций с введенными данными (хеширование, добавление статической и динамической солей) приложение сравнивает введенные данные с данными из базы данных. Если данные совпали, возвращает результат, что пользователь авторизован. Если нет – возвращает ошибку. Код данного метода представлен на рисунке 3.4. Если пользователь был успешно авторизован, он будет перенаправлен на главный экран, если метод уровня BLL вернул ошибку, то будет отображена ошибка.

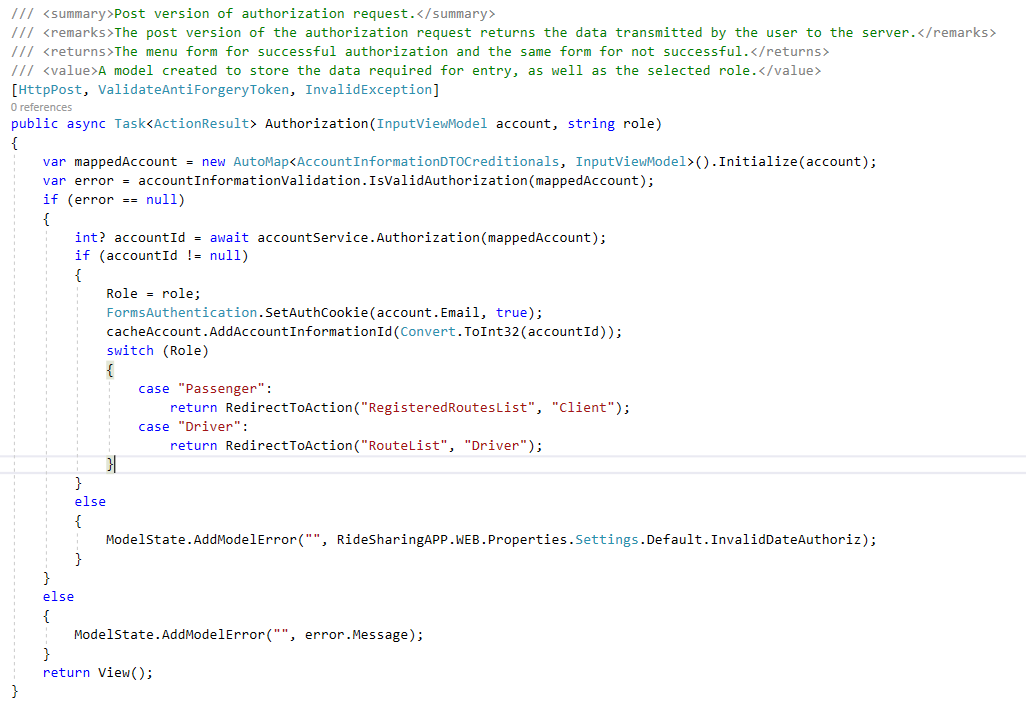


Рисунок 3.1 – Метод авторизации



Рисунок 3.2 – Метод валидации модели

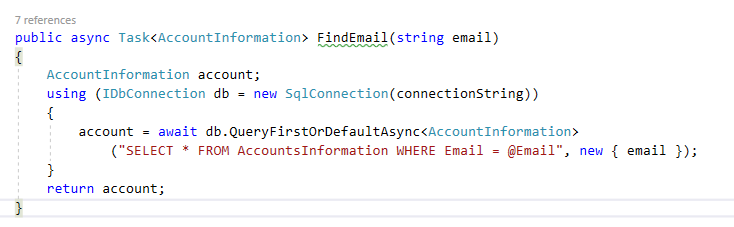


Рисунок 3.3 – Метод получения данных из базы данных

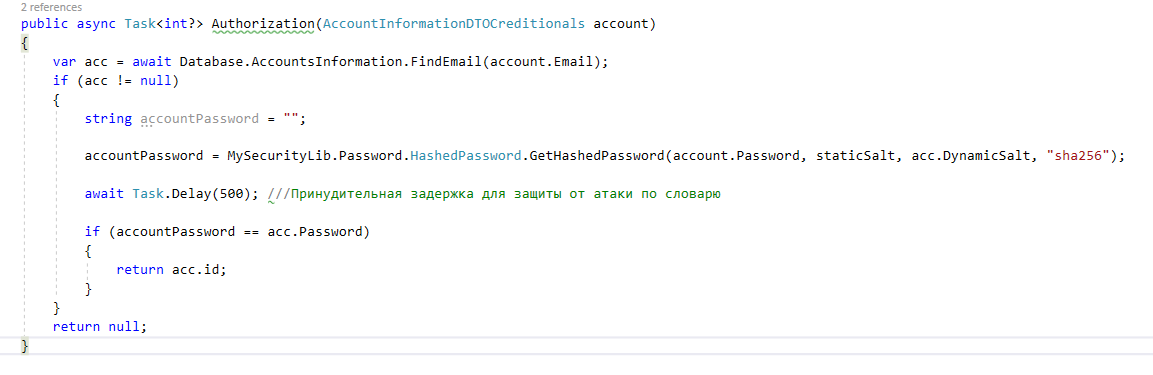


Рисунок 3.4 – Метод сравнения идентификационных данных

В зависимости от роли при успешной авторизации пользователь получит соответствующие права. В данной информационной системе существует 2 роли:

* попутчик (имеет возможность регистрации на маршрут);
* водитель (имеет возможность добавлять маршруты).

Модуль попутчика отвечает за поиск маршрута, регистрацию на маршрут, добавлении личных данных.

Рассмотрим работу данного модуля на примере поиска маршрута и дальнейшей регистрации на его.

Изначально при открытии экрана «Поиск маршрута» срабатывает Get метод который проверяет заполнял ли пользователь личные данные. Если у пользователя добавлены личные данные, приложение отображает страницу «Поиск маршрутов», если данные не заполнены, приложение перенаправляет пользователя на страницу для ввода пользовательских данных.

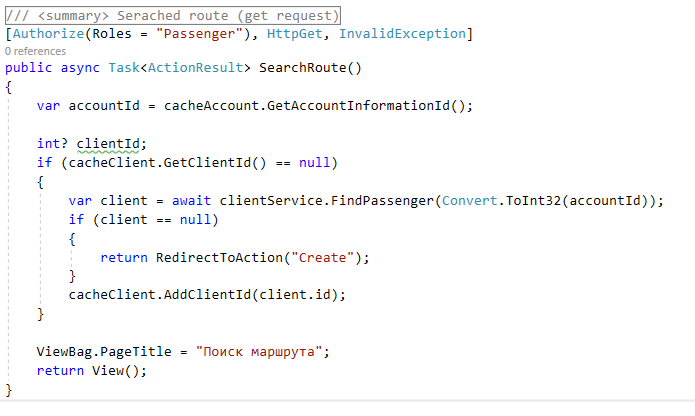


Рисунок 3.5 – Get метод поиска маршрутов

При заполнении поля «Точка отправления» под полем выпадает список с названиями населенных пунктов, адресов, начинающихся с введенного значения. Код функционала позволяющий заполнять выпадающий список приведен на рисунке 3.6.

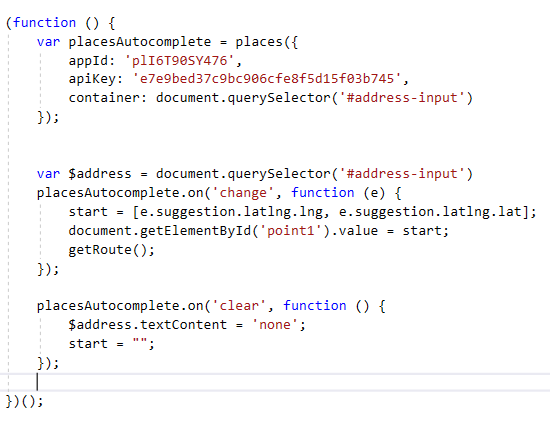


Рисунок 3.6 – Метод заполняющий список с Api запроса

Для заполнения списка значениями используется открытое API, [community.algolia.com/places](https://community.algolia.com/places/).

Когда пользователь выбирает точку отправления, на указанный населенный пункт фокусируется карта. Затем на карте появляется маркер точки отправления. Код функционала, позволяющего сфокусировать карту на населенном пункте приведен на рисунке 3.7. Код функционала, позволяющего нарисовать на карте маркер, приведен на рисунке 3.8.

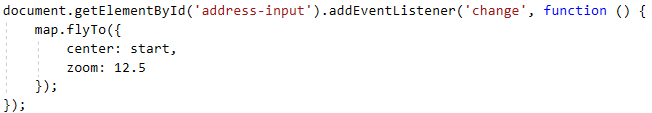


Рисунок 3.7 – Метод фокусировки карты



Рисунок 3.8 – Метод отрисовки маркера на карте

Для отображения карты на форме используется Api сервиса mapbox.com. Во время разработки были испробованы также и другие сервисы: google maps, яндекс карты, однако был выбран mapbox так как он является открытым программным обеспечением. Изначально на карте показываются вручную установленные координаты, в данный момент – координаты города Минск. Код позволяющий отобразить карту, на форме представлен на рисунке 3.9.



Рисунок 3.9 – Метод отображения карты

При выборе точки отправления на карте отрисовывается маршрут, от точки отправления до точки назначения. Построение и отрисовка маршрута происходят также с помощью api сервиса mapbox. Код функционала, позволяющего построить маршрут приведен на рисунке 3.10.



Рисунок 3.10 – Метод построения маршрута

Метод отрисовки маршрута представлен на рисунке 3.11.



Рисунок 3.11 – Метод отрисовки маршрута

После выбора точек маршрута и нажатии на кнопку «Найти» срабатывает Post метод формы поиска маршрута, он проверяет корректность введенных данных и, если точки введены корректно, перенаправляет пользователя на экран со списком маршрутов, если точки введены некорректно показывает соответствующее сообщение. Код данного метода представлен на рисунке 3.12.



Рисунок 3.12 – Post метод поиска маршрута

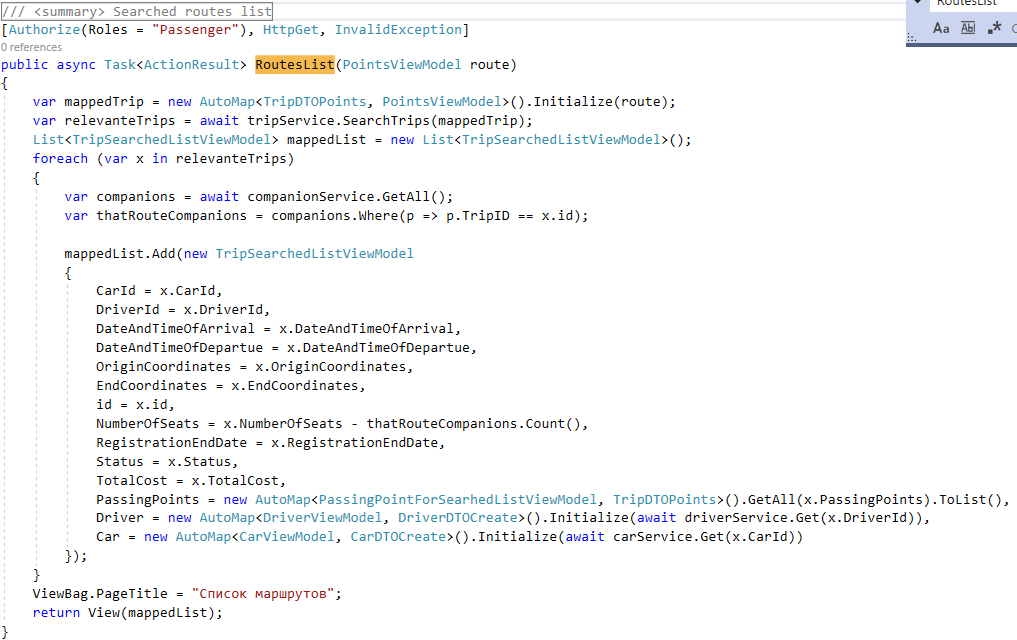


Рисунок 3.13 – Get метод получения списка маршрутов

Данный метод получает из базы данных список маршрутов, которые проходят через указанные ранее точки. Пользователю предлагается зарегистрироваться на какой-нибудь из перечисленных маршрутов. Регистрация на маршрут происходит при нажатии кнопки «Зарегистрироваться» с помощью метода, код которого представлен на рисунке 3.14.



Рисунок 3.14 – Метод регистрации на маршрут

Данный метод проверяет корректность указанных данных и, если данные указаны корректно вносит информацию, что пользователь зарегистрирован на маршрут, в базу данных, затем перенаправляет пользователя на экран с зарегистрированными маршрутами, если данные указаны не корректно, то выводит сообщение о ошибке. Метод, который загружает активные маршруты на форму представлен на рисунке 3.15. Так как на указанную форму можно перейти из меню, сначала метод проверяет вводил ли пользователь личные данные, если не вводил перенаправляет его на форму для ввода личных данных. Затем метод достает из базы данных активные маршруты пользователя. После метод получает номер водителя для каждого активного маршрута.



Рисунок 3.15 – Метод получения активных маршрутов

## 3.2 Разработка структуры базы данных

В качестве системы управления базами данных использовался Microsoft SQL-сервер. Это система управления базами данных, движок которой работает на облачных серверах, а также локальных серверах, причем можно комбинировать типы применяемых серверов   
одновременно [22].

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных [21].

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта [21].

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ [21].

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation") [21].

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения [21].

Одной из уникальных особенностей версии 2016 года является temporal data support (временная поддержка данных), которая позволяет отслеживать изменения данных с течением времени. Последняя версия MS SQL-сервера поддерживает dynamic data masking (динамическую маскировку данных), которая гарантирует, что только авторизованные пользователи будут видеть конфиденциальные данные [22].

Выбор пал на MS SQL-сервер, так как имеет следующие преимущества:

1. Производительность. SQL Server работает очень быстро.
2. Надежность и безопасность. SQL Server предоставляет шифрование данных.
3. Простота. С данной СУБД относительно легко работать и вести администрирование.

К тому же многоуровневая архитектура позволяет без особых сложностей перейти на использование другой базы данных.

Разработана следующая структура базы данных приложения, представлены следующие сущности:

1. Клиенты – таблица которая хранит личные данные о попутчиках (атрибуты сущности представлены в таблице 3.1).
2. Водители – таблица которая содержит личные данные о водителях (атрибуты сущности представлены в таблице 3.2).
3. Водительские удостоверения – таблица которая хранит данные о водительских удостоверениях (атрибуты сущности представлены   
   в таблице 3.3).
4. Автомобили – таблица которая хранит данные о автомобилях (атрибуты сущности представлены в таблице 3.4).
5. Информация о аккаунтах – таблица которая хранит данные о учетных данных пользователей (атрибуты сущности представлены в таблице 3.5).
6. Компаньоны – таблица которая хранит данные о попутчиках, зарегистрированных на маршруты (атрибуты представлены в таблице 3.6).
7. Промежуточные точки – таблица которая хранит данные о точках маршрутов (атрибуты представлены в таблице 3.7).
8. Маршруты – таблица, которая хранит в себе данные о зарегистрированных маршрутов (атрибуты представлены в таблице 3.8).
9. Категории прав – перечисление категорий прав, хранится на стороне приложения (атрибуты сущности представлены в таблице 3.9).
10. Статусы – перечисление статусов маршрута, хранится на стороне приложения (атрибуты представлены в таблице 3.10).

Таблица 3.1 – Атрибуты сущности Clients

| Имя столбца | Тип данных | Ключ | Информация о поле |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код клиента |
| DateOfRegistration | date | – | Дата регистрации |
| Name | varchar(255) | – | Имя |
| Surname | varchar(255) | – | Фамилия |
| Patronymic | varchar(255) | – | Отчество |
| DateOfBirth | date | – | Дата рождения |
| Telephone | varchar(255) | – | Номер телефона |
| Comments | varchar(255) | – | Комментарий |
| Enabled | bit | – | Указывает заблокирована ли учетная запись |
| ReasonForBlocking | varchar(255) | – | Причина блокировки аккаунта |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| AccountInformationID | int | FK | Код информации о аккаунте |

Таблица 3.2 – Атрибуты сущности Drivers

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Ключ | Информация о поле |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код водителя |
| DateOfRegistration | date | – | Дата регистрации |
| Name | varchar(255) | – | Имя |
| Surname | varchar(255) | – | Фамилия |
| Patronymic | varchar(255) | – | Отчество |
| DateOfBirth | date | – | Дата рождения |
| Telephone | varchar(255) | – | Номер телефона |
| Comments | varchar(255) | – | Комментарий |
| Enabled | bit | – | Указывает заблокирована ли учетная запись |
| ReasonForBlocking | varchar(255) | – | Причина блокировки аккаунта |
| AccountInformationID | int | FK | Код информации о аккаунте |

Таблица 3.3 – Атрибуты сущности DriversLicenses

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Ключ | Информация о поле |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код водительского удостоверения |
| DateOfReceiving | date | – | Дата получения прав |
| ExpirationDate | date | – | Дата годности прав |
| DriverLicensesNumber | varchar(255) | – | Номер водительского удостоверения |
| IssuingAuthority | varchar(255) | – | Орган выдавший документ |
| Categories | int | FK | Категории |
| DriverID | int | FK | Код водителя |

Таблица 3.4 – Атрибуты сущности Cars

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Ключ | Информация о поле |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код автомобиля |
| Mark | varchar(255) | – | Марка |
| Model | varchar(255) | – | Модель |
| YearOfIssue | int | – | Год выпуска |
| RegistrationNumber | varchar(255) | – | Регистрационный номер |
| Comments | varchar(255) | – | Комментарий |
| Deleted | bit | – | Удалена ли |
| DriverID | int | FK | Код водителя |

Таблица 3.5 – Атрибуты сущности AccountInformation

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Ключ | Информация о поле |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код информации |
| Email | varchar(255) | – | Почта |
| Password | varchar(255) | – | Пароль |
| DynamicSalt | varchar(255) | – | Динамическая соль |

Таблица 3.6 – Атрибуты сущности Companions

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Ключ | Информация о поле |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код компаньона |
| ClientID | int | FK | Код клиента |
| TripID | int | FK | Код маршрута |
| TotalCost | decimal | – | Окончательная стоимость |
| DateAndTimeOfDepartue | datetime | – | Дата отправления |
| DateAndTimeOfArrival | datetime | – | Дата прибытия |
| OriginCoordinates | varchar(300) | – | Координаты отправления |
| EndCoordinates | varchar(300) | – | Координаты прибытия |

Таблица 3.7 – Атрибуты сущности PassingPoints

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Ключ | Информация о поле |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код точки |

Продолжение таблицы 3.7

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| OriginCoordinates | varchar(255) | – | Координаты отправления |
| EndCoordinates | varchar(255) | – | Координаты прибытия |
| DateAndTimeOfDepartue | datetime | – | Дата отправления |
| DateAndTimeOfArrival | datetime | – | Дата прибытия |
| Cost | decimal | – | Стоимость |
| PointNumberOfThisTrip | int | – | Номер точки данного маршрута |
| TripID | int | FK | Код маршрута |

Таблица 3.8 – Атрибуты сущности Trips

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Ключ | Информация о поле |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код маршрута |
| NumberOfSeats | int | – | Количество мест |
| Status | int | FK | Статус маршрута |
| RegistrationDate | datetime | – | Дата окончания регистрации |
| CarID | int | FK | Код автомобиля |
| DriverID | int | FK | Код водителя |

Таблица 3.9 – Атрибуты сущности Categories

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Ключ | Информация о поле |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код категорий |
| text | string | – | Наименования категорий |

Таблица 3.10 – Атрибуты сущности Statuses

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Ключ | Информация о поле |
| 1 | 2 | 3 | 4 |
| id | int | PK | Код категорий |
| text | string | – | Наименования категорий |

## 3.3 Тестирование программного средства

Тестирование программного средства – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, тестирование – это одна из техник контроля качества, включающая в себя активности по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов [23].

В качестве документации по тестированию могут использоваться следующие виды документации: Acceptance Sheet, Test Survey или тест-кейсы.

Тестовый случай – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Тест-кейсы разделяются по ожидаемому результату на позитивные и негативные. Позитивный тест кейс использует только корректные данные и проверяет, что приложение правильно выполнило вызываемую функцию. Негативный тест кейс оперирует как корректными, так и некорректными данными (минимум 1 некорректный параметр) и ставит целью проверку исключительных ситуаций (срабатывание валидаторов), а также проверяет, что вызываемая приложением функция не выполняется при срабатывании валидатора [24].

Рассмотрим позитивные и негативные тест-кейсы для каждого сценария. В таблицах 3.11–3.18 представлено описание положительных тест-кейсов.

Таблица 3.11 – Позитивный тест-кейс проверки регистрации

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | PTC-1 |
| Описание | Проверка регистрации пользователя в системе |
| Действующее лицо/ роль | Пользователь: водитель, попутчик |

Продолжение таблицы 3.11

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Предусловие | – |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Перейти по пути ./index. | 1. Открывается страница авторизации. |
| 1. Нажать на кнопку «Регистрация». | 1. Открывается страница регистрации. |
| 1. Заполнить поля корректными данными. | 1. В поле почты пользователя появляется введенная почта. 2. В поле пароль, пароль скрывается звездочками. |
| 1. Нажать на кнопку «Регистрация». | 1. Сообщений об ошибках нет. 2. Открывается страница авторизации. 3. На почту пришло сообщение о успешной регистрации. |

Таблица 3.12 – Позитивный тест-кейс восстановления доступа

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | PTC-2 |
| Описание | Проверка восстановления доступа к системе |
| Действующее лицо/ роль | Пользователь: водитель, попутчик |
| Предусловие | 1. Зарегистрировать аккаунт (тест-кейс PTC -1). |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Перейти по пути ./index. | 1. Открывается страница авторизации. |
| 1. Нажать на кнопку «Восстановить пароль». | 1. Открывается страница восстановления пароля. |

Продолжение таблицы 3.12

|  |  |
| --- | --- |
| 3 | 4 |
| 1. Ввести корректную почту. | 1. В поле почты пользователя появляется введенная почта. |
| 1. Нажать кнопку «Восстановить» | 1. Сообщений об ошибках нет. |
| 1. В диалоговом окне нажать «Ок». | 1. На почту пришло сообщение о смене пароля. 2. Открывается страница авторизации. |

Таблица 3.13 – Позитивный тест-кейс авторизации

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | PTC-3 |
| Описание | Проверка авторизации пользователя в системе |
| Действующее лицо/ роль | Пользователь: водитель, попутчик |
| Предусловие | 1. Зарегистрировать аккаунт (тест-кейс PTC-1). 2. Данные должны заполняться в первый раз. |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Перейти по пути ./index. | 1. Открывается страница авторизации. |
| 1. Заполнить поля корректными данными. | 1. В поле почты пользователя появляется введенная почта. 2. В поле пароль, пароль скрывается звездочками. |
| 1. Нажать кнопку «Войти». | 1. Сообщений об ошибках нет. 2. Открывается главная страница. |

Таблица 3.14 – Позитивный тест-кейс заполнения личных данных

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | PTC-4 |
| Описание | Проверка первого заполнения личных данных |
| Действующее лицо/ роль | Пользователь: водитель, попутчик |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Данные должны заполняться в первый раз. |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Выполнить вход в систему (тест-кейс PTC-3). | 1. Открывается страница добавления личных данных. |
| 1. Заполнить поля корректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Сохранить». | 1. Сообщений об ошибках нет. 2. У водителя открывается страница добавления данных о водительском удостоверении, у попутчика открывается страница с активными маршрутами. |

Таблица 3.15 – Позитивный тест-кейс заполнения данных о водительском удостоверении

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | PTC-5 |
| Описание | Проверка первого заполнения данных о водительском удостоверении. |

Продолжение таблицы 3.15

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Действующее лицо/ роль | Пользователь: водитель. |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Данные должны заполняться в первый раз. |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Заполнить личные данные пользователя (тест-кейс PTC-4). | 1. Открывается страница добавления данных о водительском удостоверении. |
| 1. Заполнить поля корректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Сохранить». | 1. Сообщений об ошибках нет. 2. Открывается экран добавления данных о водительском удостоверении. |

Таблица 3.16 – Позитивный тест-кейс заполнения данных о автомобиле

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | PTC-6 |
| Описание | Проверка первого заполнения данных о автомобиле. |
| Действующее лицо/ роль | Пользователь: водитель. |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Данные должны заполняться в первый раз. |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Заполнить данные о водительском удостоверении (тест-кейс PTC-5). | 1. Открывается страница добавления данных о автомобиле. |

Продолжение таблицы 3.16

|  |  |
| --- | --- |
| 3 | 4 |
| 1. Заполнить поля корректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Сохранить». | 1. Сообщений об ошибках нет. 2. Открывается экран с активными маршрутами. |

Таблица 3.17 – Позитивный тест-кейс добавления нового маршрута

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | PTC-7 |
| Описание | Проверка добавления нового маршрута. |
| Действующее лицо/ роль | Пользователь: водитель. |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Заполнить личные данные, данные о водительском удостоверении, данные о автомобиле (тест-кейсы PTC-4, PTC-5, PTC-6). 3. Выполнить вход в систему (тест-кейс PTC-3). |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Нажать кнопку «Добавить маршрут». | 1. Открывается страница добавления маршрута. |
| 1. Выбрать точку отправления. | 1. В поле отображен выбранный населенный пункт. 2. Карта сфокусировалась на пункте отправления. |

Продолжение таблицы 3.17

|  |  |
| --- | --- |
| 3 | 4 |
| 1. Выбрать точку назначения. | 1. В поле отображен выбранный населенный пункт. 2. На карте построился маршрут от пункта отправления до пункта назначения. |
| 1. Заполнить поля корректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Далее». | 1. Открылась форма добавления дополнительной информации о маршруте. |
| 1. Заполнить поля корректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Ввод». | 1. Сообщений об ошибках нет. 2. Открывается экран с активными маршрутами, на котором отображен добавленный маршрут. |

Таблица 3.18 – Позитивный регистрации на маршрут

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | PTC-8 |
| Описание | Проверка регистрации пользователя на маршрут. |
| Действующее лицо/ роль | Пользователь: попутчик. |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Заполнить личные данные (тест-кейс PTC-4). 3. Выполнить вход в систему (тест-кейс PTC-3). |

Продолжение таблицы 3.18

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Нажать кнопку «Найти маршрут». | 1. Открывается страница поиска маршрута. |
| 1. Выбрать точку отправления. | 1. В поле отображен выбранный населенный пункт. 2. Карта сфокусировалась на пункте отправления. |
| 1. Выбрать точку назначения. | 1. В поле ввода отображен выбранный населенный пункт. 2. На карте построился маршрут от пункта отправления до пункта назначения. |
| 1. Нажать кнопку «Найти». | 1. Сообщений об ошибках нет. 2. Открывается экран с маршрутами в данном направлении. 3. Стоимость маршрута рассчитана исходя из маршрута. |
| 1. Выбрать из предложенных маршрутов, нажать кнопку «Зарегистрироваться». | 1. Сообщений об ошибках нет. 2. Открылась страница с активными маршрутами, на котором отображен новый маршрут. |

Вышеизложенные позитивные тест-кейсы пройдены успешно.

Рассмотрим негативные тест-кейсы. В таблицах 3.19–3.26 представлено подробное описание негативных тест-кейсов.

Таблица 3.19 – Негативный тест-кейс проверки регистрации

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | NTC-1 |
| Описание | Проверка регистрации пользователя в системе |

Продолжение таблицы 3.19

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Действующее лицо/ роль | Пользователь: водитель, попутчик |
| Предусловие | – |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Перейти по пути ./index. | 1. Открывается страница авторизации. |
| 1. Нажать на кнопку «Регистрация». | 1. Открывается страница регистрации. |
| 1. Заполнить поля некорректными данными. | 1. В поле почты пользователя появляется введенная почта. 2. В поле пароль, пароль скрывается звездочками. |
| 1. Нажать на кнопку «Регистрация». | 1. Под полями появились сообщения с ошибками. |

Таблица 3.20 – Негативный тест-кейс восстановления доступа

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | NTC-2 |
| Описание | Проверка восстановления доступа к системе |
| Действующее лицо/ роль | Пользователь: водитель, попутчик |
| Предусловие | 1. Зарегистрировать аккаунт (тест-кейс PTC-1). |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Перейти по пути ./index. | 1. Открывается страница авторизации. |

Продолжение таблицы 3.20

|  |  |
| --- | --- |
| 3 | 4 |
| 1. Нажать на кнопку «Восстановить пароль». | 1. Открывается страница восстановления пароля. |
| 1. Ввести некорректную почту. | 1. В поле почты пользователя появляется введенная почта. |
| 1. Нажать кнопку «Восстановить». | 1. Появилось сообщение с ошибкой. |

Таблица 3.21 – Негативный тест-кейс авторизации

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | NTC-3 |
| Описание | Проверка авторизации пользователя в системе |
| Действующее лицо/ роль | Пользователь: водитель, попутчик |
| Предусловие | 1. Зарегистрировать аккаунт (тест-кейс PTC-1). 2. Данные должны заполняться в первый раз. |

|  |  |  |
| --- | --- | --- |
| Выполняемое действие | Ожидаемый результат | |
| 3 | 4 | |
| 1. Перейти по пути ./index. | 1. Открывается страница авторизации. | |
| 1. Заполнить поля некорректными данными. | 1. В поле почты пользователя появляется введенная почта. 2. В поле пароль, пароль скрывается звездочками. | |
| 1. Нажать кнопку «Войти». | | 1. Появились сообщения с ошибками. |

Таблица 3.22 – Негативный тест-кейс заполнения личных данных

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | NTC-4 |
| Описание | Проверка первого заполнения личных данных |
| Действующее лицо/ роль | Пользователь: водитель, попутчик |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Данные должны заполняться в первый раз. |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Выполнить вход в систему (тест-кейс PTC-3). | 1. Открывается страница добавления личных данных. |
| 1. Заполнить поля некорректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Сохранить». | 1. Появились сообщения с ошибками. |

Таблица 3.23 – Негативный тест-кейс заполнения данных о водительском удостоверении

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | NTC-5 |
| Описание | Проверка первого заполнения данных о водительском удостоверении. |
| Действующее лицо/ роль | Пользователь: водитель. |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Данные должны заполняться в первый раз. |

Продолжение таблицы 3.23

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Заполнить личные данные пользователя (тест-кейс PTC-4). | 1. Открывается страница добавления данных о водительском удостоверении. |
| 1. Заполнить поля некорректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Сохранить». | 1. Появились сообщения с ошибками. |

Таблица 3.24 – Негативный тест-кейс заполнения данных о автомобиле

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | NTC-6 |
| Описание | Проверка первого заполнения данных о автомобиле. |
| Действующее лицо/ роль | Пользователь: водитель. |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Данные должны заполняться в первый раз. |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Заполнить данные о водительском удостоверении (тест-кейс PTC-5). | 1. Открывается страница добавления данных о автомобиле. |
| 1. Заполнить поля некорректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Сохранить». | 1. Появились сообщения с ошибками. |

Таблица 3.25 – Негативный тест-кейс добавления нового маршрута

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | NTC-7 |
| Описание | Проверка добавления нового маршрута. |
| Действующее лицо/ роль | Пользователь: водитель. |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Заполнить личные данные, данные о водительском удостоверении, данные о автомобиле (тест-кейсы PTC-4, PTC-5, PTC-6). 3. Выполнить вход в систему (тест-кейс PTC-3). |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Нажать кнопку «Добавить маршрут». | 1. Открывается страница добавления маршрута. |
| 1. Не выбирать точку отправления. | 1. Поле остается пустым. |
| 1. Не выбирать точку назначения. | 1. Поле остается пустым. |
| 1. Заполнить поля некорректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Далее». | 1. Открылась форма добавления дополнительной информации о маршруте. |
| 1. Заполнить поля некорректными данными. | 1. В полях ввода отображены введенные данные. |
| 1. Нажать кнопку «Ввод». | 1. Появились сообщения с ошибками. 2. Открывается форма заполнения точек маршрута. |

Таблица 3.26 – Негативный регистрации на маршрут

|  |  |
| --- | --- |
| Условие | Описание |

|  |  |
| --- | --- |
| 1 | 2 |

|  |  |
| --- | --- |
| Название | NTC-8 |
| Описание | Проверка регистрации пользователя на маршрут. |
| Действующее лицо/ роль | Пользователь: попутчик. |
| Предусловие | 1. Зарегистрировать аккаунт. 2. Заполнить личные данные (тест-кейс PTC-4). 3. Выполнить вход в систему (тест-кейс PTC-3). |

|  |  |
| --- | --- |
| Выполняемое действие | Ожидаемый результат |
| 3 | 4 |
| 1. Нажать кнопку «Найти маршрут». | 1. Открывается страница поиска маршрута. |
| 1. Не выбирать точку отправления. | 1. Поле остается пустым. |
| 1. Не выбирать точку назначения. | 1. Поле остается пустым. |
| 1. Нажать кнопку «Найти». | 1. Появились сообщения с ошибками. |

Вышеизложенные негативные тест-кейсы пройдены успешно.

Описанные тест-кейсы исследуют работоспособность приложения при вводе «правильных» и «неправильных» данных. Подобные исследования являются неотъемлемой частью тестирования программного продукта. Успешное выполнение представленных тест-кейсов демонстрирует корректную работу и соответствие основным требованиям к разработанной системе.

# 4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ СОВМЕСТНОГО ИСПОЛЬЗОВАНИЯ ЧАСТНОГО АВТОМОБИЛЯ

## 4.1 Характеристика программного средства

В рамках дипломного проектирования, сотрудниками   
ООО «Элейтсофтвеа» было разработано веб-приложение RideSharing. Задача приложения состоит в упрощении организации совместного использования частного автомобиля.

Сервис предоставляет следующие преимущества для попутчиков:

1. Есть возможность зарегистрироваться на маршрут с любого устройства с доступом в интернет.
2. Пользователь видит сколько в машине осталось свободных мест, когда выбирает маршрут.
3. Пользователь сразу видит с каким водителем, на каком авто и каким маршрутом они поедут.
4. Благодаря системе онлайн-бронирования, можно в любое время забронировать место на выбранную поездку.
5. Пользователь в любое время может посмотреть дату поездки, ее стоимость, телефон водителя.
6. Можно отменить регистрацию на маршрут, если планы поменялись.

Сервис предоставляет следующие преимущества для водителей:

1. Есть возможность создать маршрут с любого устройства с доступом в интернет.
2. Водитель видит всех пассажиров, зарегистрированных на маршрут, подробную информацию о их.
3. Водитель может указать дополнительные правила в автомобиле в комментариях, например, указать что в автомобиле не курят, или наоборот указать что водитель будет курить в автомобиле.
4. Число свободных мест обновляется автоматически, когда пассажиры бронируют место онлайн.
5. Есть возможность указать несколько точек маршрута, при выборе маршрута попутчиком стоимость будет рассчитана исходя из выбранных точек.

## 4.2 Расчет затрат на разработку программного средства

Основная заработная плата исполнителей проекта определяется по формуле (4.1):

где – коэффициент премий (по данным предприятия);

*n* – количество исполнителей, занятых разработкой программного средства;

– часовая заработная плата *i*-го исполнителя, руб.;

– трудоёмкость работ, выполняемых *i*-ым исполнителем, определяется исходя из сложности разработки программного обеспечения и объёма выполняемых им функций, ч.

В разработке участвует 4 специалиста: бизнес-аналитик, системный архитектор, инженер-программист и специалист по тестированию программного обеспечения. Месячная заработная плата исполнителей (без премии) соответствует размеру, установленному в организации в соответствии со штатным расписанием.

Расчет основной заработной платы исполнителей представлен в виде таблицы 4.1.

Таблица 4.1 – Расчет основной заработной платы исполнителей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Исполнитель | Вид выполняемой работы | Месячная заработная плата, руб. | Часовая заработная плата, руб. | Трудоёмкость  работ, ч. | Итого, руб. |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Бизнес- аналитик | Проектирование и расстановка приоритетов, планирование выполнения задач, контроль, коммуникации | 3059 | 18,21 | 20 | 364,20 |

Продолжение таблицы 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Системный архитектор | Проектирование архитектуры приложения | 7649 | 45,53 | 10 | 455,30 |
| Инженер-программист | Разработка приложения | 2549 | 15,17 | 160 | 2427,20 |
| Специалист по тестированию программного обеспечения | Тестирование работы приложения | 1274 | 7,58 | 40 | 303,20 |
| Основная заработная плата | | | | | 3549,90 |
| Премия (50 %) | | | | | 1774,95 |
| Итого | | | | | 5324,85 |

Дополнительная заработная плата исполнителей проекта определяется по формуле (4.2):

где – норматив дополнительной заработной платы (15 %).

После подстановки значений в формулу (4.2) дополнительная заработная плата составит:

Отчисления на социальные нужды () определяются в соответствии с действующими законодательными актами по формуле (4.3):

где ‒ ставка отчислений в ФСЗН и Белгосстрах (в соответствии с действующим законодательством по состоянию на 01.01.2020 г. ‒ 34,6 %).

После подстановки значений в формулу (4.3) отчисления на социальные нужды составят:

Затраты по статье «Прочие расходы» (), связанные с разработкой программного средства, определяются по формуле (4.4):

где – норматив накладных расходов (20 %).

После подстановки значений в формулу (4.4) накладные расходы составят:

Общая сумма затрат по всем статьям сметы () на разработку ПО рассчитывается по формуле (4.5):

После подстановки значений в формулу (4.5) сумма затрат по всем статьям сметы составит:

Затраты на сопровождение и адаптацию определяются по формуле (4.6):

где – норматив расходов на сопровождение и адаптацию (10 %);

– расходы на сопровождение и адаптацию программного средства в целом по организации (руб.);

– смета расходов в целом по организации без расходов на сопровождение и адаптацию (руб.).

После подстановки значений в формулу (4.6) затраты на сопровождение и адаптацию составят:

Общая сумма расходов на разработку (с затратами на сопровождение и адаптацию) как полная себестоимость программного средства определяется по формуле (4.7):

После подстановки значений в формулу (4.7) общая сумма расходов на разработку (с затратами на сопровождение и адаптацию) составит:

Рентабельность и прибыль по создаваемому программному средству определяется по формуле (4.8):

где – прибыль от реализации программного средства заказчику (руб.);

– уровень рентабельности программного средства (25 %);

– себестоимость программного средства (руб.).

После подстановки значений в формулу (4.8) рентабельность и прибыль составит:

Компания ООО «Элейтсофтвеа» является резидентом Парка Высоких Технологий. По закону резиденты ПВТ освобождаются от всех корпоративных налогов, включая налог на добавленную стоимость, налог на прибыль, а также таможенных пошлин.

Отпускная цена программного средства, которая прогнозируется компанией, определяется по формуле:

После подстановки значений в формулу (4.9) отпускная цена программного средства составит:

## 4.3 Расчет экономического эффекта у разработчика

Так как разрабатываемое программное средство было заказано сторонней организацией, то была определена договорная цена продажи в размере 25 548 рублей.

При индивидуальном заказе программного средства, нам нужно рассчитать экономический эффект для организации-разработчика. Для организации-разработчика экономический эффект – это прибыль.

Прибыль рассчитывается по следующей формуле (4.10):

где П – прибыль от реализации программного средства заказчику (руб.);

Ц – цена реализации программного средства заказчику (руб.);

– сумма расходов на разработку программного средства (руб.).

После подстановки значений в формулу (4.10) прибыль составит:

Рентабельность программного средства рассчитывается   
по формуле (4.11):

После подстановки значений в формулу (4.11) рентабельность составит:

Произведем сравнение планового уровня рентабельности с фактическим по следующей формуле (4.12):

После подстановки значений в формулу (4.12) сравнение планового уровня рентабельности с фактическим составит:

Проведя технико-экономическое обоснование, были получены следующие экономические показатели:

– себестоимость программного средства 10 238,04 руб.;

– отпускная цена программного средства 12 797,55 руб.;

– договорная цена программного средства 25 548 руб.;

– прибыль от реализации программного средства 15 309,96 руб.;

– фактический уровень рентабельности 149,54 %.

Благодаря экономическим показателям было выявлено, что для организации-разработчика ООО «Элейтсофтвеа» данная разработка программного средства является эффективной и экономически целесообразной.

# 5 РЕАЛИЗАЦИЯ ЭРГОНОМИЧЕСКОЙ СОВМЕСТИМОСТИ РАБОТНИКА И ТЕХНИЧЕСКОГО СРЕДСТВА ПРИ РАЗРАБОТКЕ И ЭКСПЛУАТАЦИИ ВЕБ-ПРИЛОЖЕНИЯ

Целью дипломного проекта является разработка веб-приложения приложения для совместного использования частного автомобиля и его инженерно-психологическое обеспечение.

Данная система представляет собой веб-приложение, посредством которого водители смогут найти попутчиков, в дорогу, с которыми можно будет как приятно пообщаться, так и разделить трату на бензин. Попутчикам приложение позволит найти водителя, с автомобилем, который двигается в интересующем направлении, что позволит быстро и с комфортом добраться до интересующего населенного пункта. Приложение можно использовать с любого устройства с доступом в интернет.

Технологический прогресс и широкое внедрение в производство информационных технологий значительно изменяют содержание и условия труда, что является предпосылкой для облегчения труда человека, освобождения его от выполнения однообразных трудоемких ручных операций, и вместе с тем приводит к появлению новых факторов, негативно влияющих на организм работников, среди которых на первое место выходит повышенная напряженность труда, обусловленная высокими требованиями к уровню психической деятельности человека. По этой причине внедрение в производство новейших технологий может быть успешно реализовано и дать положительный эффект лишь при достаточно полном учете характера все усложняющихся связей между человеком и техническим окружением всестороннем учете возможностей человека (человеческого фактора), его физиологических, психологических, антропометрических, эстетических и других свойств [25].

Рациональная совместимость возможностей человека и характеристик технических средств, оптимальное распределение функций между элементами системы «человек-машина» существенно повышают ее надежность, эффективность и обусловливают оптимальное использование человеком технических средств в соответствии с их назначением [25].

Для решения указанных задач эргономика использует данные и методы наук, изучающих свойства и возможности человека – физиологии, психологии, социологии и гигиены труда, антропологии, инженерной психологии и др. На их основании разрабатываются эргономические требования и рекомендации к различным видам технических средств, видам деятельности, организации трудового процесса, рабочим местам и производственной среде [25].

Для того чтобы человеко-машинная система функционировала эффективно и не приносила ущерба здоровью человека, необходимо прежде всего обеспечить совместимость характеристик машины и человека. Совместимость человека с машиной определяется его пространственно-антропометрической, сенсомоторной, энергетической (биомеханической) и психофизиологической совместимостью [25].

Пространственно-антропометрическая совместимостьпредполагает необходимость учета размеров тела человека, его возможности обзора внешнего пространства, определения зоны досягаемости для конечностей и др [25].

Антропометрические характеристики человека подразделяются на статические и динамические. К статическим характеристикам относятся размеры тела и его отдельных частей – рук, ног, кистей, стоп и др., к динамическим – возможные углы поворота отдельных частей тела, зоны досягаемости [25].

Сенсомоторная совместимость предполагает учет скорости двигательных (моторных) операций человека и его сенсорных реакций на различные виды раздражителей (световые, звуковые и др.) при выборе скорости работы машины и подачи сигналов.

Энергетическая (биомеханическая) совместимостьпредусматривает согласование прилагаемых усилий, затрачиваемой мощности, скорости и точности движений ручных и ножных органов управления, с биомеханическими возможностями человека, которые в зависимости от частоты их использования и важности должны располагаться в соответствующих зонах досягаемости. Усилия на органах управления не должны быть слишком маленькими, чтобы можно было контролировать выполненные действия, и не слишком большими, так как большие усилия приводят к быстрой усталости и перенапряжению мышц [26].

Умственный труд связан с приемом и переработкой информации и требует напряжения внимания, памяти, активизации процессов мышления, связан с повышенной эмоциональной нагрузкой. Для умственного труда характерно снижение двигательной активности – гипокинезия. Гипокинезия может являться условием формирования сердечно-сосудистых нарушений у человека [26].

Продолжительная умственная нагрузка оказывает отрицательное влияние на психическую деятельность – ухудшаются внимание, память, функции восприятия окружающей среды. Самочувствие человека и, в конечном счете, его состояние здоровья в значительной мере зависит от правильной организации умственного труда и от параметров окружающей среды, в которой осуществляется умственная деятельность человека [26].

Под контролем зрения совершается до – 80 % – 90 % трудовых операций. Множество тонких и точных операций выполняется в электронной и приборостроительной промышленности. Значительное место среди работ, вызывающих зрительное напряжение, занимает труд операторов, работающих на дисплеях ЭВМ. Такая работа приводит к развитию зрительного утомления. Причиной этого являются:

1. Фиксация близко расположенных, двигающихся объектов.
2. Длительное рассматривание мелких деталей.
3. Постоянный перевод взгляда с одного объекта на другой.
4. Частые и резкие переходы от света к тени и обратно.
5. Пульсации освещенности и другое.

К неблагоприятному воздействию этих факторов добавляется строго фиксированная рабочая поза и гипокинезия. Кроме того, работа за видеотерминалами требует большого нервно-психического напряжения, связанного с необходимостью длительного наблюдения, концентрации памяти и внимания, решения сложных задач [26].

Трудовая деятельность человека должна осуществляться в допустимых условиях производственной среды, речь идет о создании таких условий, при которых негативные воздействия не превышали бы защитных способностей организма [26].

Помещения, где эксплуатируются компьютеры, должны иметь как естественное, так и искусственное освещение. Естественное освещение на рабочих местах с компьютером должно осуществляться через световые проемы, ориентированные преимущественно на север, северо-восток, восток, запад или северо-запад, и обеспечивать коэффициент естественной освещенности не ниже 1,5 процента [26].

Оконные проемы должны быть оборудованы регулируемыми устройствами типа жалюзи, занавесей, внешних козырьков и другого.

Площадь одного рабочего места для пользователей компьютеров должна составлять не менее 6 кв.м. Площадь одного рабочего места может составлять не менее 4,5 кв.м. на каждого пользователя при одновременном наличии двух условий:

1. Отсутствие на рабочем месте периферийных устройств (принтер, сканер и другое).
2. Продолжительность работы должна составлять не более 4 часов в день.

Площадь одного рабочего места может составлять не менее 4,5 кв.м. на каждого пользователя при использовании компьютеров на базе плоских дискретных экранов (жидкокристаллические, плазменные и другие) [14].

Для работы с компьютерами помещения должны быть высотой от пола до потолка не менее 3,0 м.

Требования к рабочему месту, на котором выполняется работа за компьютером, установлены Санитарными нормами № 59 [14].

При размещении рабочих мест с компьютерами расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора) должно быть не менее 2,0 м., а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м [14].

Рабочие места с компьютерами при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рекомендуется изолировать друг от друга перегородками высотой 1,5–2,0 м [14].

Экран видеомонитора должен находиться на расстоянии 600–700 мм от глаз пользователя, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов [14].

Конструкция рабочего стола должна предусматривать оптимальное размещение компьютера и другого оборудования с учетом специфики выполняемой работы и количества оборудования [14].

Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию [14].

Высота рабочей поверхности стола для взрослых пользователей должна регулироваться в пределах 680–800 мм. При отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм [14].

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм [14].

Конструкция рабочего стула (кресла) должна обеспечивать:

1. Ширину и глубину поверхности сиденья не менее 400 мм.
2. Поверхность сиденья с закругленным передним краем.
3. Регулировку высоты поверхности сиденья в пределах 400–550 мм и угла наклона вперед до 15 градусов и назад до 5 градусов.
4. Высоту опорной поверхности спинки 300 +/– 20 мм, ширину не менее 380 мм и радиус кривизны горизонтальной плоскости – 400 мм.
5. Угол наклона спинки в вертикальной плоскости в пределах 0 +/– 30 градусов.
6. Регулировку расстояния спинки от переднего края сиденья в пределах 260–400 мм.
7. Стационарные или съемные подлокотники длиной не менее 250 мм и шириной 50–70 мм.
8. Регулировку подлокотников по высоте над сиденьем   
   в пределах 230 +/– 30 мм и внутреннего расстояния между подлокотниками   
   в пределах 350–500 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии 100–300 мм от края, обращенного ко взрослому пользователю, или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы [14].

Из вышеизложенного следует, что для высокой работоспособности и получения эффективной отдачи от труда работников, с наименьшим ущербом для их состояния здоровья, необходимо:

1. Правильно, в соответствии с научно разработанными нормативами, организовать рабочие места.
2. Поддерживать микроклимат и аэроионный состав воздуха рабочей зоны.
3. Следить за соблюдением режима труда и отдыха.

Соблюдение этих требований позволяет исключить или минимизировать для работников, неблагоприятное воздействие производственной среды и трудового процесса, обеспечить сохранение здоровья.

# ЗАКЛЮЧЕНИЕ

В рамках дипломного проектирования было разработано веб-приложение для упрощения организации совместного использования частного автомобиля RideSharing и его инженерно-психологиеское обеспечение. Разработанная система учитывает недостатки существующих аналогов. Также были проведены следующие расчёты:

* количество машиночасов для разработки информационной системы;
* количество часов работы программиста для разработки информационной системы;
* стоимость часа работы программиста;
* итог заработной платы программиста для разработки информационной системы;
* итог дополнительной заработной платы для разработки информационной системы.

Разработка велась в среде разработки Visual Studio 2019 на операционной системе Windows. В качестве основного языка программирования был выбран C#, для хранения данных использована база данных Microsoft SQL Server.

В процессе проектирования проанализированы существующие аналоги, разработанные как на территории СНГ, так и за его пределами. Проведено технико-экономическое обоснование разработки и реализации на рынке данного программного продукта. Была произведена оценка существующий условий труда, выявлены недостатки и разработаны мероприятия для обеспечения высокой работоспособности программиста при разработке данного приложения.

Были решены следующие задачи:

* на основе информации о технологиях, существующих решениях и результатах эргономического проектирования, спроектирована информационная система;
* согласно проекту, разработана информационная система;
* по составленной тестовой документации проведено тестирование информационной системы.

Результаты работы позволяют сказать, что приложение для упрощения организации совместного использования частного автомобиля выполняет поставленные перед ним задачи.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Transport.mos.ru [Электронный ресурс]. – Режим доступа: [https://transport.mos.ru/mostrans/all\_news/28053/. – Дата доступа: 20.11.20](https://transport.mos.ru/mostrans/all_news/28053/.%20–%20Дата%20доступа:%2020.11.20)20.

[2] Rb.ru [Электронный ресурс]. – Режим доступа: <https://rb.ru/longread/blablacar-russia/>. – Дата доступа: 20.11.2020.

[3] Blog.blablacar.ru [Электронный ресурс]. – Режим доступа: [https://blog.blablacar.ru/blablalife/puteshestvia/sovety-po-servisu/udobstvo-blablacar-dlia-passazhirov/. – Дата доступа: 25.11.20](https://blog.blablacar.ru/blablalife/puteshestvia/sovety-po-servisu/udobstvo-blablacar-dlia-passazhirov/.%20–%20Дата%20доступа:%2025.11.20)20.

[4] Ok.ru [Электронный ресурс]. – Режим доступа: <https://ok.ru/blitssroch/topic/67252438783309/>. – Дата доступа: 26.11.2020.

[5] Vc.ru [Электронный ресурс]. – Режим доступа: <https://vc.ru/flood/43234-blablacar-kupil-konkuriruyushchiy-servis-beepcar-u-mail-ru-group/>. – Дата доступа: 26.11.2020.

[6] Blablacar.ru [Электронный ресурс]. – Режим доступа: <https://www.blablacar.ru/>. – Дата доступа: 26.11.2020.

[7] Mahnem.ru [Электронный ресурс]. – Режим доступа: https://www.mahnem.ru/. – Дата доступа: 22.11.2020.

[8] Dovezu.ru [Электронный ресурс]. – Режим доступа: http://www.dovezu.ru/. – Дата доступа: 23.11.2020.

[9] Ux.pub [Электронный ресурс]. – Режим доступа: <https://ux.pub/15-printsipov-dizayna-interfeysov-kotorye-pozvolyat-sozdavat-luchshie-veb-sayty/>.   
– Дата доступа: 23.11.2020.

[10] Livetyping.com [Электронный ресурс]. – Режим доступа: <https://livetyping.com/ru/blog/printsipi-i-osnovy-ux-dizajna>/. – Дата доступа: 23.11.2020.

[11] Rb.ru [Электронный ресурс]. – Режим доступа: <https://rb.ru/opinion/uxui/>. – Дата доступа: 23.11.2020.

[12] Habr [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/company/SECL_GROUP/blog/182208/>. – Дата доступа: 23.11.2020.

[13] Bokardo.com [Электронный ресурс]. – Режим доступа: <http://bokardo.com/principles-of-user-interface-design/>. – Дата доступа: 23.11.2020.

[14] Bka.brest.by [Электронный ресурс]. – Режим доступа: <https://bka.brest.by/2018/05/31/>. Дата доступа: 25.11.2020.

[15] Habr [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/397261/>. Дата доступа: 26.11.2020.

[16] Шупейко, И. Г. Эргономическое проектирование систем «человек–компьютер–среда». Курсовое проектирование: учеб.-метод. пособие / И. Г. Шупейко. – Минск: БГУИР, 2012. – 97 с.

[17] ГОСТ 20.39.108-85 [Электронный ресурс]. – Режим доступа: <http://docs.cntd.ru/document/gost-20-39-108-85/>. – Дата доступа: 22.11.2020.

[18] Metanit [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/1.1.php/. – Дата доступа: 22.11.2020.

[19] Habr [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/215605/>. – Дата доступа: 26.11.2020.

[20] Metanit [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/mvc5/23.5.php>/. – Дата доступа: 22.11.2020.

[21] Metanit [Электронный ресурс]. – Режим доступа: <https://metanit.com/sql/sqlserver/1.1.php>/. – Дата доступа: 23.11.2020.

[22] Drach.pro [Электронный ресурс]. – Режим доступа: https://drach.pro/blog/hi-tech/item/145-db-comparison/. – Дата доступа: 23.11.2020.

[23] Protesting.Lesson [Электронный ресурс]. – Режим доступа: http://www.protesting.ru/testing/.– Дата доступа: 25.11.2020.

[24] Винокурова, Л.В. Организационная психология. Хрестоматия / Л.В. Винокурова, И.И. Скрипюка. – СПб.: Питер, 2001. – 256 с.

[25] Protesting.Tutorials [Электронный ресурс]. – Режим доступа: http://www.protesting.ru/testing/testcase.html/. – Дата доступа: 25.11.2020.

[26] Михнюк, Т.Ф. Охрана труда: учебник для студ. высш. учеб. завед./ Т.Ф. Михнюк - Мн.: ИВЦ Минфина, 2009. - 345 с.

[27] Девисилов, В.А. Охрана труда: учебник/ В.А. Девисилов - 4-е изд., перераб., и доп. - М.: Форум, 2009. - 496с.

[28] Белов, С. В. Безопасность жизнедеятельности: учебник для вузов / С. В. Белов [и др.]. – М.: Высш. шк. 1999.

[29] Шелмаков, С.В. Ш 444 Экотранспорт: учеб. пособие / Шелмаков С.В. – М.: МАДИ, 2018. – 199 с.

[30] Доманов, А. Т. Стандарт предприятия: Дипломные проекты (работы). Общие требования. СТП 01-2017 / Доманов А. Т., Сорока Н. И. – Минск: БГУИР 2017. – 168 с.

[31] Mapbox.com [Электронный ресурс]. – Режим доступа: <https://docs.mapbox.com/api/overview/>. – Дата доступа: 25.10.2020.

[32] Mapbox.github.io [Электронный ресурс]. – Режим доступа: [https://mapbox.github.io/mapbox-sdk-cs/api/Mapbox.Geocoding.html/. – Дата доступа: 25.10.20](https://mapbox.github.io/mapbox-sdk-cs/api/Mapbox.Geocoding.html/.%20–%20Дата%20доступа:%2025.10.20)20.

[33] Community.algolia.com [Электронный ресурс]. – Режим доступа: <https://community.algolia.com/places/documentation.html>/. – Дата доступа: 25.10.2020.

[34] Habr [Электронный ресурс]. – Режим доступа:  [https://habr.com/ru/sandbox/56203/. – Дата доступа: 25.10.20](%20https://habr.com/ru/sandbox/56203/.%20–%20Дата%20доступа:%2025.10.20)20.

[35] Habr [Электронный ресурс]. – Режим доступа:  [https://habr.com/ru/post/110460/. – Дата доступа: 25.10.20](%20https://habr.com/ru/post/110460/.%20–%20Дата%20доступа:%2025.10.20)20.

# Приложение А (обязательное) Листинг программы

using RideSharingApp.BLL.Interfaces;

using RideSharingApp.BLL.Services;

using RideSharingApp.WEB.Helpers;

using RideSharingAPP.BLL.DTO.AccountInformationDTO;

using RideSharingAPP.BLL.Interfaces.IValidations;

using RideSharingAPP.WEB.Attributes;

using RideSharingAPP.WEB.Attributes.Cache;

using RideSharingAPP.WEB.Models.Authorization;

using System;

using System.Threading.Tasks;

using System.Web.Mvc;

using System.Web.Security;

namespace RideSharingApp.WEB.Controllers

{

/// <summary>The controller in which logging in, registering, restoring access to the account.</summary>

public class AuthorizationController : Controller

{

/// <summary>Variable storing role under which the user is authorized</summary>

public static string Role;

private AccountCache cacheAccount;

IAccountInformationValidation accountInformationValidation;

IAccountInformationService accountService;

/// <summary>Controller constructor.</summary>

/// <value>Accepts the interface of the validation service as well as the service table of accounts.</value>

/// <remarks>Constructor - a function that fires before other constructor methods.</remarks>

public AuthorizationController(IAccountInformationService serv, IAccountInformationValidation val)

{

accountInformationValidation = val;

accountService = serv;

cacheAccount = new AccountCache();

}

/// <summary>Returns an atorization form</summary>

/// <remarks>Form that allows the user to enter the application or go to the registration form, password recovery.</remarks>

/// <returns>View()</returns>

[HttpGet, InvalidException]

public ActionResult Authorization()

{

return View();

}

/// <summary>Post version of authorization request.</summary>

/// <remarks>The post version of the authorization request returns the data transmitted by the user to the server.</remarks>

/// <returns>The menu form for successful authorization and the same form for not successful.</returns>

/// <value>A model created to store the data required for entry, as well as the selected role.</value>

[HttpPost, ValidateAntiForgeryToken, InvalidException]

public async Task<ActionResult> Authorization(InputViewModel account, string role)

{

var mappedAccount = new AutoMap<AccountInformationDTOCreditionals, InputViewModel>().Initialize(account);

var error = accountInformationValidation.IsValidAuthorization(mappedAccount);

if (error == null)

{

int? accountId = await accountService.Authorization(mappedAccount);

if (accountId != null)

{

Role = role;

FormsAuthentication.SetAuthCookie(account.Email, true);

cacheAccount.AddAccountInformationId(Convert.ToInt32(accountId));

switch (Role)

{

case "Passenger":

return RedirectToAction("RegisteredRoutesList", "Client");

case "Driver":

return RedirectToAction("RouteList", "Driver");

}

}

else

{

ModelState.AddModelError("", RideSharingAPP.WEB.Properties.Settings.Default.InvalidDateAuthoriz);

}

}

else

{

ModelState.AddModelError("", error.Message);

}

return View();

}

/// <summary>

/// Exit in account's

/// </summary>

/// <returns>Redirect to LogIn</returns>

[InvalidException]

public ActionResult LogOut()

{

new CarsCache().DeleteCars();

new ClientCache().DeleteClientId();

new DriverCache().DeleteDriverId();

new DriverLicensesCache().DeleteDriverLicensesId();

cacheAccount.DeleteAccountInformationId();

FormsAuthentication.SignOut();

return RedirectToAction("Authorization");

}

/// <summary>Returns an registration form</summary>

/// <remarks>Form allowing you to register in the application.</remarks>

/// <returns>View()</returns>

[HttpGet, InvalidException]

public ActionResult Registration()

{

return View();

}

/// <summary>Post version of registration request.</summary>

/// <remarks>The post version of the registration request returns the data by the user to the server.</remarks>

/// <returns>The menu form for successful authorization and the same form for not successful.</returns>

/// <value>A model created to store the data required for entry, as well as the selected role.</value>

[HttpPost, ValidateAntiForgeryToken, InvalidException]

public async Task<ActionResult> Registration(RegistrationViewModel account, string role)

{

var mappedAccount = new AutoMap<AccountInformationDTOCreditionals, RegistrationViewModel>().Initialize(account);

var error = await accountInformationValidation.IsValidRegistration(mappedAccount);

if (error == null)

{

int? newAccountsId = await accountService.Registration(mappedAccount);

HttpContext.Response.Cookies["AccountInformationId"].Value = newAccountsId.ToString();

await SendEmail.SendAsync(account.Email, "Поздравляем с регистрацией", "Служба поддержки");

return RedirectToAction("Authorization");

}

else

{

ModelState.AddModelError("", error.Message);

}

return View();

}

/// <summary>Returns password recovery form.</summary>

/// <remarks>A form that allows you to restore access to your account.</remarks>

/// <returns>View()</returns>

[HttpGet, InvalidException]

public ActionResult PasswordRecovery()

{

return View();

}

/// <summary>Post version of the request to restore account access.</summary>

/// <remarks>It receives from the client the data required to restore access to the account.</remarks>

/// <returns>Sends a new password to the mail and also redirects to the login form.</returns>

/// <value>Model containing user mail.</value>

[HttpPost, ValidateAntiForgeryToken, InvalidException]

public async Task<ActionResult> PasswordRecovery(PasswordRecoveryView account)

{

var mappedAccount = new AutoMap<AccountInformationDTOGetEmail, PasswordRecoveryView>().Initialize(account);

var error = await accountInformationValidation.IsValidPasswordRecovery(mappedAccount);

if (error == null)

{

string password = await accountService.PasswordRecovery(account.Email);

bool check = await SendEmail.SendAsync(account.Email, "Восстановление пароля", $"Сообщаем Вам, что Ваш пароль на сервере RideSharing был изменен, новый пароль: {password} \nВы можете изменить его в личном кабинете на сайте.");

if (check)

{

return RedirectToAction("Authorization", "Authorization");

}

else

{

ModelState.AddModelError("", RideSharingAPP.WEB.Properties.Settings.Default.SendMessageEror);

}

}

else

{

ModelState.AddModelError("", error.Message);

}

return View();

}

/// <summary>Invokes a method to free database connection memory.</summary>

protected override void Dispose(bool disposing)

{

accountService.Dispose();

base.Dispose(disposing);

}

}

}

using RideSharingApp.BLL.Interfaces;

using RideSharingApp.BLL.Services;

using RideSharingAPP.BLL.DTO.AccountInformationDTO;

using RideSharingAPP.BLL.DTO.CarDTO;

using RideSharingAPP.BLL.DTO.ClientDTO;

using RideSharingAPP.BLL.DTO.CompanionDTO;

using RideSharingAPP.BLL.DTO.DriverDTO;

using RideSharingAPP.BLL.DTO.TripDTO;

using RideSharingAPP.BLL.Interfaces;

using RideSharingAPP.BLL.Interfaces.IValidations;

using RideSharingAPP.WEB.Attributes;

using RideSharingAPP.WEB.Attributes.Cache;

using RideSharingAPP.WEB.Models.Authorization;

using RideSharingAPP.WEB.Models.Car;

using RideSharingAPP.WEB.Models.Client;

using RideSharingAPP.WEB.Models.Companion;

using RideSharingAPP.WEB.Models.Driver;

using RideSharingAPP.WEB.Models.PassingPoint;

using RideSharingAPP.WEB.Models.Route;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Threading.Tasks;

using System.Web.Mvc;

using System.Web.UI;

namespace RideSharingAPP.WEB.Controllers

{

/// <summary>The controller in which all actions will be carried out by the passenger.</summary>

public class ClientController : Controller

{

IAccountInformationValidation accountInformationValidation;

IClientValidation clientValidation;

ITripValidation tripValidation;

ICarValidation carValidation;

IDriverValidation driverValidation;

ICompanionValidation companionValidation;

IAccountInformationService accountInformationService;

ICompanionService companionService;

IDriverService driverService;

ICarService carService;

ITripService tripService;

IClientService clientService;

private ClientCache cacheClient;

private AccountCache cacheAccount;

/// <summary>Controller constructor.</summary>

/// <value>Accepts the interface of the validation service as well as the service table of clients.</value>

/// <remarks>Constructor - a function that fires before other constructor methods.</remarks>

public ClientController(IClientService clientServ, IClientValidation clientValid, ITripValidation tripValid, ITripService tripServ,

IDriverService driverServ, IDriverValidation driverValid, ICarService carServ, ICarValidation carValid, ICompanionService companionServ,

ICompanionValidation companionValid, IAccountInformationService accountServ, IAccountInformationValidation accountVal)

{

carValidation = carValid;

driverValidation = driverValid;

clientValidation = clientValid;

tripValidation = tripValid;

companionValidation = companionValid;

accountInformationValidation = accountVal;

accountInformationService = accountServ;

companionService = companionServ;

driverService = driverServ;

carService = carServ;

tripService = tripServ;

clientService = clientServ;

cacheAccount = new AccountCache();

cacheClient = new ClientCache();

}

/// <summary>Sends data about a new user to the server</summary>

/// <param name="client"> model containing data about a new user.</param>

[Authorize(Roles = "Passenger"), ValidateAntiForgeryToken, HttpPost, InvalidException]

public async Task<ActionResult> Create(ClientViewModel client)

{

var mappedClient = new AutoMap<ClientDTOCreate, ClientViewModel>().Initialize(client);

var error = clientValidation.IsValid(mappedClient);

if (error == null)

{

mappedClient.AccountInformationID = Convert.ToInt32(cacheAccount.GetAccountInformationId());

var newClientsId = await clientService.Create(mappedClient);

cacheClient.AddClientId(Convert.ToInt32(newClientsId));

return RedirectToAction("SearchRoute");

}

else

{

ModelState.AddModelError("", error.Message);

}

return View();

}

/// <summary>

/// Change account's password (get request)

/// </summary>

/// <returns>View</returns>

[Authorize(Roles = "Passenger"), HttpGet, InvalidException]

public ActionResult ChangePassword()

{

return View();

}

/// <summary>

/// Change account's password (post request)

/// </summary>

/// <param name="account"></param>

/// <returns></returns>

[Authorize(Roles = "Passenger"), HttpPost, ValidateAntiForgeryToken, InvalidException]

public async Task<ActionResult> ChangePassword(ChangePasswordViewModel account)

{

var mappedAccount = new AutoMap<AccountInformationDTOChangePassword, ChangePasswordViewModel>().Initialize(account);

var error = accountInformationValidation.IsValidChangePassword(mappedAccount);

if (error == null)

{

mappedAccount.AccountInformationId = Convert.ToInt32(cacheAccount.GetAccountInformationId());

var editAccountId = await accountInformationService.ChangePassword(mappedAccount);

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError("", error.Message);

}

return View(account);

}

/// <summary>

/// Account's details

/// </summary>

/// <returns>View with account's details</returns>

[Authorize(Roles = "Passenger"), HttpGet, InvalidException]

public async Task<ActionResult> Index()

{

var client = await clientService.FindPassenger(Convert.ToInt32(cacheAccount.GetAccountInformationId()));

var mappingClient = new AutoMap<ClientViewModel, ClientDTOCreate>().Initialize(client);

ViewBag.PageTitle = "Данные о пользователе";

return View(mappingClient);

}

/// <summary>Returns the form of creating a new client.</summary>

/// <remarks>Displays a form for filling in data about a new user.</remarks>

/// <returns>View</returns>

[Authorize(Roles = "Passenger"), HttpGet, InvalidException]

public ActionResult Create()

{

return View();

}

/// <summary>

/// Serached route (get request)

/// </summary>

/// <returns>View</returns>

[Authorize(Roles = "Passenger"), HttpGet, InvalidException]

public async Task<ActionResult> SearchRoute()

{

var accountId = cacheAccount.GetAccountInformationId();

int? clientId;

if (cacheClient.GetClientId() == null)

{

var client = await clientService.FindPassenger(Convert.ToInt32(accountId));

if (client == null)

{

return RedirectToAction("Create");

}

cacheClient.AddClientId(client.id);

}

ViewBag.PageTitle = "Поиск маршрута";

return View();

}

/// <summary>

/// Searched route (post request)

/// </summary>

/// <param name="route"></param>

/// <returns></returns>

[Authorize(Roles = "Passenger"), ValidateAntiForgeryToken, HttpPost, InvalidException]

public async Task<ActionResult> SearchRoute(PointsViewModel route)

{

var clientId = cacheClient.GetClientId();

if (clientId == null)

{

return RedirectToAction("Create");

}

var mappingRoute = new AutoMap<TripDTOPoints, PointsViewModel>().Initialize(route);

var error = tripValidation.PointsIsValid(mappingRoute);

if (error == null)

{

return RedirectToAction("RoutesList", route);

}

else

{

ModelState.AddModelError("", error.Message);

}

ViewBag.PageTitle = "Поиск маршрута";

return View();

}

/// <summary>

/// Registered user on route

/// </summary>

/// <param name="tripInfo"></param>

/// <returns></returns>

[Authorize(Roles = "Passenger"), ValidateAntiForgeryToken, HttpPost, SetTempDataModelState, InvalidException]

public async Task<ActionResult> RegisterForRoute(TripSearchedListViewModel tripInfo)

{

var clientId = cacheClient.GetClientId();

if (clientId == null)

{

return RedirectToAction("Create");

}

var error = await companionValidation.IsValid(Convert.ToInt32(clientId), tripInfo.id, Convert.ToInt32(cacheAccount.GetAccountInformationId()));

if (error == null)

{

var companion = new CompanionDTOCreate

{

TripID = tripInfo.id,

ClientID = Convert.ToInt32(clientId),

DateAndTimeOfArrival = tripInfo.DateAndTimeOfArrival,

DateAndTimeOfDepartue = tripInfo.DateAndTimeOfDepartue,

OriginCoordinates = tripInfo.OriginCoordinates,

EndCoordinates = tripInfo.EndCoordinates,

TotalCost = tripInfo.TotalСost

};

var newCompanionId = await companionService.RegistrationForTheRoute(companion);

return RedirectToAction("RegisteredRoutesList");

}

else

{

ModelState.AddModelError("", error.Message);

}

ViewBag.PageTitle = "Поиск маршрута";

return RedirectToAction("RoutesList");

}

/// <summary>

/// Edit user's details (get request)

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[Authorize(Roles = "Passenger"), HttpGet, InvalidException]

public async Task<ActionResult> EditClient(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

ViewBag.PageTitle = "Изменение личных данных";

var client = await clientService.Get(Convert.ToInt32(id));

var mappedClient = new AutoMap<ClientViewModel, ClientDTOCreate>().Initialize(client);

return View(mappedClient);

}

/// <summary>

/// Edit user's details (post request)

/// </summary>

/// <param name="client"></param>

/// <returns></returns>

[Authorize(Roles = "Passenger"), ValidateAntiForgeryToken, HttpPost, InvalidException]

public async Task<ActionResult> EditClient(ClientViewModel client)

{

var clientId = cacheClient.GetClientId();

if (clientId == null)

{

return RedirectToAction("Create");

}

var mappedClient = new AutoMap<ClientDTOCreate, ClientViewModel>().Initialize(client);

mappedClient.AccountInformationID = Convert.ToInt32(cacheAccount.GetAccountInformationId());

var error = clientValidation.IsValid(mappedClient);

if (error == null)

{

var editClientId = await clientService.Update(mappedClient);

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError("", error.Message);

}

return View(clientId);

}

/// <summary>

/// User's routes

/// </summary>

/// <returns></returns>

[Authorize(Roles = "Passenger"), HttpGet, RestoreModelStateFromTempData, InvalidException]

public async Task<ActionResult> RegisteredRoutesList()

{

var accountInformationId = cacheAccount.GetAccountInformationId();

var userId = Convert.ToInt32(accountInformationId);

if (cacheClient.GetClientId() == null)

{

var client = await clientService.FindPassenger(Convert.ToInt32(accountInformationId));

if (client == null)

{

return RedirectToAction("Create");

}

cacheClient.AddClientId(client.id);

}

var clientId = cacheClient.GetClientId();

var activeUserRoutes = await companionService.GetActiveUserRoutes(Convert.ToInt32(clientId));

var mappedListActiveUserRoutes = new AutoMap<CompanionViewModel, CompanionDTOCreate>().GetAll(activeUserRoutes);

foreach (var item in mappedListActiveUserRoutes)

{

item.Telephone = await companionService.GetDriverTelephone(item.TripID);

}

ViewBag.PageTitle = "Зарегистрированные маршруты";

return View(mappedListActiveUserRoutes);

}

/// <summary>

/// Searched routes list

/// </summary>

/// <param name="route"></param>

/// <returns></returns>

[Authorize(Roles = "Passenger"), HttpGet, InvalidException]

public async Task<ActionResult> RoutesList(PointsViewModel route)

{

var mappedTrip = new AutoMap<TripDTOPoints, PointsViewModel>().Initialize(route);

var relevanteTrips = await tripService.SearchTrips(mappedTrip);

List<TripSearchedListViewModel> mappedList = new List<TripSearchedListViewModel>();

foreach (var x in relevanteTrips)

{

var companions = await companionService.GetAll();

var thatRouteCompanions = companions.Where(p => p.TripID == x.id);

mappedList.Add(new TripSearchedListViewModel

{

CarId = x.CarId,

DriverId = x.DriverId,

DateAndTimeOfArrival = x.DateAndTimeOfArrival,

DateAndTimeOfDepartue = x.DateAndTimeOfDepartue,

OriginCoordinates = x.OriginCoordinates,

EndCoordinates = x.EndCoordinates,

id = x.id,

NumberOfSeats = x.NumberOfSeats - thatRouteCompanions.Count(),

RegistrationEndDate = x.RegistrationEndDate,

Status = x.Status,

TotalСost = x.TotalСost,

PassingPoints = new AutoMap<PassingPointForSearhedListViewModel, TripDTOPoints>().GetAll(x.PassingPoints).ToList(),

Driver = new AutoMap<DriverViewModel, DriverDTOCreate>().Initialize(await driverService.Get(x.DriverId)),

Car = new AutoMap<CarViewModel, CarDTOCreate>().Initialize(await carService.Get(x.CarId))

});

}

ViewBag.PageTitle = "Список маршрутов";

return View(mappedList);

}

/// <summary>

/// Unregistration of a route

/// </summary>

/// <param name="companionId"></param>

/// <returns></returns>

[Authorize(Roles = "Passenger"), SetTempDataModelState, InvalidException]

public async Task<ActionResult> Unregister(int? companionId)

{

var clientId = cacheClient.GetClientId();

if (clientId == null)

{

return RedirectToAction("Create");

}

var errorCompanion = await companionValidation.CheckId(companionId);

if (errorCompanion == null)

{

await companionService.Unregister(Convert.ToInt32(companionId));

}

else

{

ModelState.AddModelError("", errorCompanion.Message);

}

return RedirectToAction("RegisteredRoutesList");

}

/// <summary>Invokes a method to free database connection memory.</summary>

protected override void Dispose(bool disposing)

{

accountInformationService.Dispose();

companionService.Dispose();

driverService.Dispose();

carService.Dispose();

tripService.Dispose();

clientService.Dispose();

base.Dispose(disposing);

}

}

}

using RideSharinAPP.COMMON.Enums;

using RideSharingApp.BLL.Interfaces;

using RideSharingApp.BLL.Services;

using RideSharingAPP.BLL.DTO.AccountInformationDTO;

using RideSharingAPP.BLL.DTO.CarDTO;

using RideSharingAPP.BLL.DTO.ClientDTO;

using RideSharingAPP.BLL.DTO.DriverDTO;

using RideSharingAPP.BLL.DTO.DriverLicensesDTO;

using RideSharingAPP.BLL.DTO.PassingPointDTO;

using RideSharingAPP.BLL.DTO.TripDTO;

using RideSharingAPP.BLL.Interfaces;

using RideSharingAPP.BLL.Interfaces.IValidations;

using RideSharingAPP.WEB.Attributes;

using RideSharingAPP.WEB.Attributes.Cache;

using RideSharingAPP.WEB.Models.Authorization;

using RideSharingAPP.WEB.Models.Car;

using RideSharingAPP.WEB.Models.Client;

using RideSharingAPP.WEB.Models.Driver;

using RideSharingAPP.WEB.Models.DriverLicenses;

using RideSharingAPP.WEB.Models.Route;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace RideSharingAPP.WEB.Controllers

{

/// <summary>The controller in which all actions will be carried out by the passenger.</summary>

public class DriverController : Controller

{

IAccountInformationValidation accountInformationValidation;

ITripValidation tripValidation;

IPassingPointValidation passingPointValidation;

IDriverLicensesValidation driverLicensesValidation;

IDriverValidation driverValidation;

ICarValidation carValidation;

ICompanionValidation companionValidation;

IAccountInformationService accountInformationService;

ITripService tripService;

IPassingPointService passingPointService;

IDriverLicensesService driveLicensesService;

IDriverService driverService;

ICarService carService;

ICompanionService companionService;

private AccountCache accountCache;

private DriverCache driverCache;

private CarsCache carsCache;

private DriverLicensesCache driverLicensesCache;

/// <summary>Controller constructor.</summary>

/// <value>Accepts the interface of the validation service as well as the service tables of: drivers, drivers licenses, cars.</value>

/// <remarks>Constructor - a function that fires before other constructor methods.</remarks>

public DriverController(IDriverService driverserv, ICarService carserv, IDriverLicensesService driverLicensesserv, ITripService tripserv, IPassingPointService passPointserv,

ICarValidation carVal, IDriverValidation driverVal, IDriverLicensesValidation driverLicensesVal, ITripValidation tripval, IPassingPointValidation passPointval,

IAccountInformationValidation accInformationVal, IAccountInformationService accInformationServ, ICompanionService companionServ, ICompanionValidation companionVal)

{

driverLicensesValidation = driverLicensesVal;

driverValidation = driverVal;

carValidation = carVal;

tripValidation = tripval;

passingPointValidation = passPointval;

accountInformationValidation = accInformationVal;

companionValidation = companionVal;

companionService = companionServ;

accountInformationService = accInformationServ;

passingPointService = passPointserv;

tripService = tripserv;

driveLicensesService = driverLicensesserv;

driverService = driverserv;

carService = carserv;

accountCache = new AccountCache();

driverCache = new DriverCache();

carsCache = new CarsCache();

driverLicensesCache = new DriverLicensesCache();

}

/// <summary>Checks the availability of data about the driver, his driver’s license and the car, if no data are redirected to the forms to fill.</summary>

/// <returns>View() or Redirect()</returns>

[Authorize(Roles = "Driver"), HttpGet, InvalidException]

public async Task<ActionResult> Index()

{

var driver = await driverService.FindAccount(Convert.ToInt32(accountCache.GetAccountInformationId()));

var mappingDriver = new AutoMap<DriverViewModel, DriverDTOCreate>().Initialize(driver);

ViewBag.PageTitle = "Данные о пользователе";

return View(mappingDriver);

}

/// <summary>Returns the form of creating a new car.</summary>

/// <remarks>Displays a form for filling in data about a new car.</remarks>

/// <returns>View()</returns>

[Authorize(Roles = "Driver"), HttpGet, InvalidException]

public ActionResult CreateCar()

{

ViewBag.PageTitle = "Добавление автомобиля";

return View();

}

/// <summary>Returns the form of creating out driver license information.</summary>

/// <remarks>Displays a form for filling out driver license information.</remarks>

/// <returns>View()</returns>

[Authorize(Roles = "Driver"), HttpGet, InvalidException]

public ActionResult CreateDriverLicenses()

{

var driverLicenses = new DriverLicensesViewModel();

ViewBag.PageTitle = "Водительское удостоверение";

var categoriesEnum = new Categories().GetList();

var categoriesList = categoriesEnum.Select(arg => new { idCategory = arg.idCategory, Categories = arg.Categories }).ToList();

var categoriesSelectedList = new SelectList(categoriesList, "idCategory", "Categories");

driverLicenses.ListCategories = categoriesSelectedList;

return View(driverLicenses);

}

/// <summary>Sends data about a new driver license information to the server.</summary>

/// <param name="driverLicense"> model containing data about a new driver licenses information.</param>

[Authorize(Roles = "Driver"), HttpPost, ValidateAntiForgeryToken, InvalidException]

public async Task<ActionResult> CreateDriverLicenses(DriverLicensesViewModel driverLicense)

{

var validDriverLicenses = new AutoMap<DriverLicensesDTOCreate, DriverLicensesViewModel>().Initialize(driverLicense);

var error = driverLicensesValidation.IsValid(validDriverLicenses);

if (error == null)

{

var mappingDriversLicense = new AutoMap<DriverLicensesDTOCreate, DriverLicensesViewModel>().Initialize(driverLicense);

mappingDriversLicense.DriverID = Convert.ToInt32(driverCache.GetDriverId());

var newDriversLicensesId = await driveLicensesService.Create(mappingDriversLicense);

return RedirectToAction("CreateRoute");

}

else

{

ModelState.AddModelError("", error.Message);

}

var categoriesEnum = new Categories().GetList();

var categoriesList = categoriesEnum.Select(arg => new { idCategory = arg.idCategory, Categories = arg.Categories }).ToList();

var categoriesSelectedList = new SelectList(categoriesList, "idCategory", "Categories");

driverLicense.ListCategories = categoriesSelectedList;

return View(driverLicense);

}

/// <summary>

/// Change user's password (get request)

/// </summary>

/// <returns></returns>

[Authorize(Roles = "Driver"), HttpGet]

public ActionResult ChangePassword()

{

return View();

}

/// <summary>

/// Change user's password (post request)

/// </summary>

/// <param name="account"></param>

/// <returns></returns>

[Authorize(Roles = "Driver"), ValidateAntiForgeryToken, HttpPost, InvalidException]

public async Task<ActionResult>ChangePassword(ChangePasswordViewModel account)

{

var mappedAccount = new AutoMap<AccountInformationDTOChangePassword, ChangePasswordViewModel>().Initialize(account);

var error = accountInformationValidation.IsValidChangePassword(mappedAccount);

if (error == null)

{

mappedAccount.AccountInformationId = Convert.ToInt32(accountCache.GetAccountInformationId());

var editAccountId = await accountInformationService.ChangePassword(mappedAccount);

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError("", error.Message);

}

return View(account);

}

/// <summary>

/// Deleted user's car (get request)

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[Authorize(Roles = "Driver"), SetTempDataModelState, InvalidException]

public async Task<ActionResult> DeleteCar(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

var error = await carValidation.DeleteCarIsValid(Convert.ToInt32(id));

if (error == null)

{

var checkResult = await carService.DeleteCar(Convert.ToInt32(id));

return RedirectToAction("CarList");

}

else

{

ModelState.AddModelError("", error.Message);

return RedirectToAction("CarList");

}

}

/// <summary>

/// Edited driver's details (get request)

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[Authorize(Roles = "Driver"), HttpGet]

public async Task<ActionResult> EditDriver(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

ViewBag.PageTitle = "Изменение личных данных";

var driver = await driverService.Get(Convert.ToInt32(id));

var mappedDriver = new AutoMap<DriverCreateViewModel, DriverDTOCreate>().Initialize(driver);

return View(mappedDriver);

}

/// <summary>

/// Edited driver's details (post request)

/// </summary>

/// <param name="driver"></param>

/// <returns></returns>

[Authorize(Roles = "Driver"), ValidateAntiForgeryToken, HttpPost, InvalidException]

public async Task<ActionResult> EditDriver(DriverCreateViewModel driver)

{

var driverId = driverCache.GetDriverId();

if (driverId == null)

{

return RedirectToAction("Create");

}

var mappedDriver = new AutoMap<DriverDTOCreate, DriverCreateViewModel>().Initialize(driver);

mappedDriver.AccountInformationID = Convert.ToInt32(accountCache.GetAccountInformationId());

var error = driverValidation.IsValid(mappedDriver);

if (error == null)

{

var editDriverId = await driverService.Update(mappedDriver);

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError("", error.Message);

}

return View(driverId);

}

/// <summary>

/// Edited car's comment (get request)

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[Authorize(Roles = "Driver"), HttpGet]

public async Task<ActionResult> EditCar(int? id)

{

var driverId = driverCache.GetDriverId();

if (driverId == null)

{

return RedirectToAction("Create");

}

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

var car = await carService.Get(Convert.ToInt32(id));

var mappedCar = new AutoMap<EditCarComments, CarDTOCreate>().Initialize(car);

if (car == null)

{

return HttpNotFound();

}

return View(mappedCar);

}

/// <summary>

/// Edited car's comment (post request)

/// </summary>

/// <param name="car"></param>

/// <returns></returns>

[Authorize(Roles = "Driver"), ValidateAntiForgeryToken, HttpPost, InvalidException]

public async Task<ActionResult> EditCar(EditCarComments car)

{

var driverId = driverCache.GetDriverId();

if (driverId == null)

{

return RedirectToAction("Create");

}

var driverLicensesId = driverLicensesCache.GetDriverLicensesId();

if (driverLicensesId == null)

{

return RedirectToAction("CreateDriverLicenses");

}

var mappedCar = new AutoMap<CarDTOCreate, EditCarComments>().Initialize(car);

var error = carValidation.EditCarModelIsValid(mappedCar);

if (error == null)

{

mappedCar.DriverId = Convert.ToInt32(driverId);

var editCarId = await carService.EditCar(mappedCar);

return RedirectToAction("CarList");

}

else

{

ModelState.AddModelError("", error.Message);

}

return View(car.id);

}

/// <summary>Sends data about a new car to the server.</summary>

/// <param name="car"> model containing data about a new car.</param>

[Authorize(Roles = "Driver"), HttpPost, ValidateAntiForgeryToken, InvalidException]

public async Task<ActionResult> CreateCar(CarViewModel car)

{

var driverId = driverCache.GetDriverId();

if (driverId == null)

{

return RedirectToAction("Create");

}

var driverLicensesId = driverLicensesCache.GetDriverLicensesId();

if (driverLicensesId == null)

{

return RedirectToAction("CreateDriverLicenses");

}

var mappedCar = new AutoMap<CarDTOCreate, CarViewModel>().Initialize(car);

mappedCar.DriverId = Convert.ToInt32(driverCache.GetDriverId());

var error = carValidation.IsValid(mappedCar);

if (error == null)

{

var newCarsId = await carService.Create(mappedCar);

return RedirectToAction("CarList");

}

else

{

ModelState.AddModelError("", error.Message);

}

ViewBag.PageTitle = "Добавление автомобиля";

return View(car);

}

/// <summary>

/// List user's cars

/// </summary>

/// <returns></returns>

[Authorize(Roles = "Driver"), HttpGet, RestoreModelStateFromTempData, InvalidException]

public async Task<ActionResult> CarList()

{

var driverId = driverCache.GetDriverId();

if (driverId == null)

{

return RedirectToAction("Create");

}

var driverLicensesId = driverLicensesCache.GetDriverLicensesId();

if (driverLicensesId == null)

{

return RedirectToAction("CreateDriverLicenses");

}

var A = ModelState;

ViewBag.PageTitle = "Список автомобилей";

var cars = await carService.FindCars(Convert.ToInt32(driverId));

var mappListCars = cars.Select(p => new AutoMap<CarViewModel, CarDTOCreate>().Initialize(p)).ToList();

return View(mappListCars);

}

/// <summary>Returns the form of creating a new driver.</summary>

/// <remarks>Displays a form for filling out driver license information.</remarks>

/// <returns>View()</returns>

[Authorize(Roles = "Driver"), HttpGet, InvalidException]

public ActionResult Create()

{

ViewBag.PageTitle = "Добавление данных о водителе";

return View();

}

/// <summary>Sends data about a new driver to the server.</summary>

/// <param name="driver"> model containing data about a new driver.</param>

[Authorize(Roles = "Driver"), HttpPost, ValidateAntiForgeryToken, InvalidException]

public async Task<ActionResult> Create(DriverCreateViewModel driver)

{

var validDriver = new AutoMap<DriverDTOCreate, DriverCreateViewModel>().Initialize(driver);

var error = driverValidation.IsValid(validDriver);

if (error == null)

{

var mappedDriver = new AutoMap<DriverDTOCreate, DriverCreateViewModel>().Initialize(driver);

mappedDriver.AccountInformationID = Convert.ToInt32(accountCache.GetAccountInformationId());

var newDriversId = await driverService.Create(mappedDriver);

driverCache.AddDriverId(Convert.ToInt32(newDriversId));

return RedirectToAction("CreateRoute");

}

else

{

ModelState.AddModelError("", error.Message);

}

ViewBag.PageTitle = "Добавление данных о водителе";

return View();

}

/// <summary>

/// Added new route to user (get request)

/// </summary>

/// <returns></returns>

[Authorize(Roles = "Driver"), HttpGet, InvalidException]

public async Task<ActionResult> CreateRoute()

{

var accountInformationId = accountCache.GetAccountInformationId();

DriverDTOCreate driver = new DriverDTOCreate();

if (driverCache.GetDriverId() == null)

{

driver = await driverService.FindAccount(Convert.ToInt32(accountInformationId));

if (driver == null)

{

return RedirectToAction("Create");

}

driverCache.AddDriverId(driver.id);

}

driver.id = Convert.ToInt32(driverCache.GetDriverId());

DriverLicensesDTOCreate driverLicenses = new DriverLicensesDTOCreate();

if (driverLicensesCache.GetDriverLicensesId() == null)

{

driverLicenses = await driveLicensesService.FindDriverLicenses(driver.id);

if (driverLicenses == null)

{

return RedirectToAction("CreateDriverLicenses");

}

driverLicensesCache.AddDriverLicensesId(driverLicenses.id);

}

IEnumerable<CarDTOCreate> cars;

if (carsCache.GetCars() == null)

{

cars = await carService.FindCars(driver.id);

if (Enumerable.Count(cars) == 0)

{

return RedirectToAction("CreateCar");

}

carsCache.AddCars(cars);

}

cars = carsCache.GetCars();

ViewBag.PageTitle = "Добавление маршрута";

var carEnum = await carService.GetAllForDropDown(Convert.ToInt32(driver.id));

var carList = carEnum.Select(arg => new { CarId = arg.id, registrationNumber = arg.RegistrationNumber }).ToList();

var carSelectedList = new SelectList(carList, "CarId", "registrationNumber");

TripViewModel trip = new TripViewModel();

trip.ListCars = carSelectedList;

return View(trip);

}

/// <summary>

/// User's routes

/// </summary>

/// <returns></returns>

[Authorize(Roles = "Driver"), HttpGet, RestoreModelStateFromTempData, InvalidException]

public async Task<ActionResult> RouteList()

{

var accountInformationId = accountCache.GetAccountInformationId();

DriverDTOCreate driver = new DriverDTOCreate();

if (driverCache.GetDriverId() == null)

{

driver = await driverService.FindAccount(Convert.ToInt32(accountInformationId));

if (driver == null)

{

return RedirectToAction("Create");

}

driverCache.AddDriverId(driver.id);

}

driver.id = Convert.ToInt32(driverCache.GetDriverId());

DriverLicensesDTOCreate driverLicenses = new DriverLicensesDTOCreate();

if (driverLicensesCache.GetDriverLicensesId() == null)

{

driverLicenses = await driveLicensesService.FindDriverLicenses(driver.id);

if (driverLicenses == null)

{

return RedirectToAction("CreateDriverLicenses");

}

driverLicensesCache.AddDriverLicensesId(driverLicenses.id);

}

IEnumerable<CarDTOCreate> cars;

if (carsCache.GetCars() == null)

{

cars = await carService.FindCars(driver.id);

if (Enumerable.Count(cars) == 0)

{

return RedirectToAction("CreateCar");

}

carsCache.AddCars(cars);

}

cars = carsCache.GetCars();

var trips = await tripService.GetDriversTrips(Convert.ToInt32(driver.id));

var mappedTrips = trips.Select(p => new AutoMap<TripsListViewModel, TripDTOList>().Initialize(p)).ToList();

foreach (var x in mappedTrips)

{

var points = await passingPointService.GetRoutesPassingPoints(x.id);

var pointsSelectedTrip = points.ToList();

if (pointsSelectedTrip != null)

{

x.Points = pointsSelectedTrip.Select(p => new AutoMap<PassingPointViewModel, PassingPointDTOCreate>().Initialize(p)).ToList();

}

var car = await carService.Get(x.CarId);

var mappedCar = new AutoMap<CarViewModel, CarDTOCreate>().Initialize(car);

x.Car = mappedCar;

var clients = await companionService.GetClients(x.id);

var mappedClients = new AutoMap<ClientWithCompanionDateViewModel, ClientDTOwithCompanionDate>().GetAll(clients).ToList();

x.Clients = mappedClients;

}

ViewBag.PageTitle = "Активные маршруты";

return View(mappedTrips);

}

/// <summary>

/// Added new route to user (post request)

/// </summary>

/// <param name="trip"></param>

/// <returns></returns>

[Authorize(Roles = "Driver"), ValidateAntiForgeryToken, HttpPost, InvalidException]

public async Task<ActionResult> CreateRoute(TripViewModel trip)

{

var accountInformationId = accountCache.GetAccountInformationId();

DriverDTOCreate driver = new DriverDTOCreate();

if (driverCache.GetDriverId() == null)

{

driver = await driverService.FindAccount(Convert.ToInt32(accountInformationId));

if (driver == null)

{

return RedirectToAction("Create");

}

driverCache.AddDriverId(driver.id);

}

driver.id = Convert.ToInt32(driverCache.GetDriverId());

DriverLicensesDTOCreate driverLicenses = new DriverLicensesDTOCreate();

if (driverLicensesCache.GetDriverLicensesId() == null)

{

driverLicenses = await driveLicensesService.FindDriverLicenses(driver.id);

if (driverLicenses == null)

{

return RedirectToAction("CreateDriverLicenses");

}

driverLicensesCache.AddDriverLicensesId(driverLicenses.id);

}

IEnumerable<CarDTOCreate> cars;

if (carsCache.GetCars() == null)

{

cars = await carService.FindCars(driver.id);

if (Enumerable.Count(cars) == 0)

{

return RedirectToAction("CreateCar");

}

carsCache.AddCars(cars);

}

cars = carsCache.GetCars();

ViewBag.PageTitle = "Добавление маршрута";

var mappedPassingPointList = new AutoMap<PassingPointDTOCreate, PassingPointViewModel>().GetAll(trip.Points);

var validRoute = new AutoMap<TripDTOCreate, TripViewModel>().Initialize(trip);

var errorRoute = tripValidation.IsValid(validRoute);

var errorPassingPoint = passingPointValidation.IsValid(mappedPassingPointList);

if (errorPassingPoint == null && errorRoute == null)

{

trip.DriverId = Convert.ToInt32(driver.id);

var mappedTrip = new AutoMap<TripDTOCreate, TripViewModel>().Initialize(trip);

var newTripsId = await tripService.Create(mappedTrip);

int tripId = Convert.ToInt32(newTripsId);

foreach (var a in mappedPassingPointList)

{

a.TripID = tripId;

}

var newPassingPointsId = await passingPointService.Create(mappedPassingPointList, newTripsId);

return RedirectToAction("RouteList");

}

else

{

if (errorPassingPoint != null)

{

ModelState.AddModelError("", errorPassingPoint.Message);

}

if (errorRoute != null)

{

ModelState.AddModelError("", errorRoute.Message);

}

}

var carEnum = await carService.GetAllForDropDown(driver.id);

var carList = carEnum.Select(arg => new { CarId = arg.id, registrationNumber = arg.RegistrationNumber }).ToList();

var carSelectedList = new SelectList(carList, "CarId", "registrationNumber");

trip.ListCars = carSelectedList;

return View(trip);

}

/// <summary>

/// Delete user's trip (get request)

/// </summary>

/// <param name="tripId"></param>

/// <returns></returns>

[Authorize(Roles = "Driver"), SetTempDataModelState, InvalidException]

public async Task<ActionResult> DeleteTrip(int? tripId)

{

var errorRoute = await tripValidation.DeleteTripIsValid(tripId);

if (errorRoute == null)

{

await tripService.SetStatusDeleted(Convert.ToInt32(tripId));

}

else

{

ModelState.AddModelError("", errorRoute.Message);

}

return RedirectToAction("RouteList");

}

/// <summary>

/// Completed user's trip

/// </summary>

/// <param name="tripId"></param>

/// <returns></returns>

[Authorize(Roles = "Driver"), SetTempDataModelState, InvalidException]

public async Task<ActionResult> CompleteTrip(int? tripId)

{

var errorRoute = await tripValidation.CompleteTripIsValid(tripId);

if (errorRoute == null)

{

await tripService.SetStatusCompleted(Convert.ToInt32(tripId));

}

else

{

ModelState.AddModelError("", errorRoute.Message);

}

return RedirectToAction("RouteList");

}

/// <summary>Invokes a method to free database connection memory.</summary>

protected override void Dispose(bool disposing)

{

driveLicensesService.Dispose();

passingPointService.Dispose();

accountInformationService.Dispose();

companionService.Dispose();

driverService.Dispose();

carService.Dispose();

tripService.Dispose();

base.Dispose(disposing);

}

}

}

using System;

using System.Collections.Generic;

using System.Web.Mvc;

using Ninject;

using Ninject.Web.Common;

using RideSharingApp.BLL.Services;

using RideSharingApp.BLL.Interfaces;

using RideSharingAPP.BLL.Interfaces;

using RideSharingAPP.BLL.Services;

using RideSharingAPP.BLL.Interfaces.IValidations;

using RideSharingAPP.BLL.Services.Validations;

using AutoMapper;

using RideSharingAPP.BLL.DTO.AccountInformationDTO;

using RideSharingAPP.WEB.Models.Authorization;

namespace RideSharingAPP.WEB.Util

{

public class NinjectDependencyResolver : IDependencyResolver

{

private IKernel kernel;

public NinjectDependencyResolver()

{

kernel = new StandardKernel();

kernel.Unbind<ModelValidatorProvider>();

AddBindings();

}

public NinjectDependencyResolver(IKernel kernelParam)

{

kernel = kernelParam;

kernel.Unbind<ModelValidatorProvider>();

AddBindings();

}

public object GetService(Type serviceType)

{

return kernel.TryGet(serviceType);

}

public IEnumerable<object> GetServices(Type serviceType)

{

return kernel.GetAll(serviceType);

}

private void AddBindings()

{

kernel.Bind<IAccountInformationService>().To<AccountInformationService>();

kernel.Bind<IDriverService>().To<DriverService>();

kernel.Bind<IClientService>().To<ClientService>();

kernel.Bind<ICarService>().To<CarService>();

kernel.Bind<IDriverLicensesService>().To<DriverLicensesService>();

kernel.Bind<ITripService>().To<TripService>();

kernel.Bind<IPassingPointService>().To<PassingPointService>();

kernel.Bind<ICompanionService>().To<CompanionService>();

kernel.Bind<ICompanionValidation>().To<CompanionValidation>();

kernel.Bind<IPassingPointValidation>().To<PassingPointValidation>();

kernel.Bind<IAccountInformationValidation>().To<AccountInformationValidation>();

kernel.Bind<IDriverValidation>().To<DriverValidation>();

kernel.Bind<IClientValidation>().To<ClientValidation>();

kernel.Bind<ICarValidation>().To<CarValidation>();

kernel.Bind<IDriverLicensesValidation>().To<DriverLicensesValidation>();

kernel.Bind<ITripValidation>().To<TripValidation>();

}

}

}

# Ведомость дипломного проекта