

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального назначения»
Специализация: «Эксплуатация вычислительных машин,
комплексов, систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

Сервис совместного поиска попутчиков

Студент

Антон Сергеевич Кривошеин

(подпись, дата)

(ФИО полностью)

Руководитель

Р.Н. Федюнин

(подпись, дата)

(фамилия, инициалы)

Нормоконтролёр

А.В. Кучин

(подпись, дата)

(фамилия, инициалы)

Рецензент

К.Т.Н. (должность, место работы)

02.07.2019

(подпись, дата)

А.И. Мартышкин
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 31.05.19 г. № 38-09-14)

Заведующий кафедрой

(подпись)

М.А. Митрохин

(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 05.07.2019 № 7)

Секретарь ГЭК

(подпись)

Е.И. Гурин

Пенза 2019

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

Факультет

вычислительной техники

Кафедра

«Вычислительная
техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

«25» апреля 20 19 г.

ЗАДАНИЕ

НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Кривошеин А.С. гр. 14ВО2 факультета ВТ

Кривошеин Антон Сергеевич

2. Тема работы: Разработка системы, помогающей пользователям находить себе попутчиков для проезда в пределах города или путешествия на дальние расстояния.

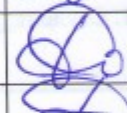
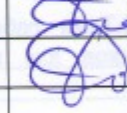
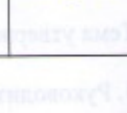
Тема утверждена приказом ПГУ № 53410 от "29" 04 20 19 г.

3. Руководитель работы Федюнин Роман Николаевич

4. Задание на работу (назначение разработки, исходные данные и т.п.) В данной работе поставлена задача разработать сервис взаимного поиска попутного транспорта и пассажиров. Данная система должна предоставлять пользователям возможность регистрации в системе, создания маршрутов поездок, управления своими маршрутами, составления заявок на поездки, управления своими заявками, поиска необходимой информации по сайту, а также взаимодействовать с другими пользователями системы.

5. Перечень подлежащих разработке вопросов Разработка базы данных онлайн сервиса,
разработка логической модели базы данных, разработка физической модели базы данных,
разработка программного обеспечения, разработка и анализ пользовательского интерфейса,
разработка программного обеспечения клиентской части, разработка программного обеспечения
серверной части.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Выбор темы и разработка индивидуального плана проведения работ	10	11.02.19-12.02.19	
2	Подбор и изучение материала по теме работы	464	12.02.19-17.04.19	
3	Получение и анализ результатов	300	17.04.19-25.05.19	
4	Оформление отчёта	90	02.06.19	
	Общий объём часов	864		

Дата выдачи задания " 11 " 02 20 19 г.

Руководитель работы  (подпись, дата)

Задание к исполнению принял студент  (подпись, дата)

Работу к защите допустить
 Декан факультета  05.07.2019
 (подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 211 листов, 54 рисунка, 40 таблиц, 9 источников, 4 приложения.

ОНЛАЙН СИСТЕМА, БАЗА ДАННЫХ, YANDEXAPI, APIВКОНТАКТЕ.

Объектом разработки является онлайн сервис совместного поиска попутчиков.

Цель работы – проведение предпроектного анализа существующих онлайн сервисов совместного поиска попутчиков и разработка онлайн системы с учетом исправления недостатков в имеющихся системах.

В результате проведенной работы создан проект, удовлетворяющей поставленной задаче.

В пояснительной записке описано: технология разработки системы, описано проведение предпроектного анализа, выявлены недостатки имеющихся систем, проведена разработка и отладка онлайн сервиса.

В проектной части описана разработка системы.

Описана технология разработки и отладки серверной части и интерфейса системы, а также описан протокол взаимодействия серверной и интерфейсной частей.

В экономической части дается технико-экономическое обоснование целесообразности разработки системы.

В разделе безопасности жизнедеятельности описаны психофизиологические факторы, воздействующие на оператора ЭВМ, а так же рассмотрено воздействие работы с ПК на зрение человека.

Графическая часть содержит схемы всей системы, данных, структуры базы данных, алгоритма работы программных модулей.

Содержание

Введение.....	9
1. Анализ предметной области и постановка задачи на разработку.....	13
1.1 Обоснование разработки.....	13
1.2 Анализ бизнес-процессов предметной области.....	16
1.3 Сравнительный анализ существующих аналогов.....	22
1.4 Постановка задачи на разработку.....	31
1.5 Функциональные возможности онлайн сервиса.....	37
1.6 Требования к архитектуре.....	40
1.7 Требования к пользовательскому интерфейсу.....	42
1.7.1 Критерии качества пользовательского интерфейса.....	44
1.7.2 Методика оценки эффективности пользовательского интерфейса на основе модели GOMS.....	46
1.7.3 Требования к аппаратно-программному обеспечению.....	50
1.8 Выбор технологий и средств разработки.....	51
1.8.1 Выбор средств реализации серверной части.....	51
1.8.2 Выбор средств реализации клиентской части.....	56
1.8.3 Выбор среды разработки.....	58
2. Реализация онлайн сервиса.....	59
2.1 Разработка базы данных онлайн сервиса.....	59
2.2 Разработка логической модели базы данных.....	61
2.3 Разработка физической модели базы данных.....	77
2.4 Разработка программного обеспечения.....	87
2.4.1 Разработка и анализ пользовательского интерфейса.....	87

2.4.2 Разработка программного обеспечения клиентской части.....	102
2.4.3 Разработка программного обеспечения серверной части.....	110
3. Технология разработки и отладки web-приложений в среде NetBeans...	117
3.1 Создание проекта в NetBeans.....	117
3.2 Отладка кода с помощью XDebug.....	122
3.2.1 Как работает отладка.....	124
4. Техничко-экономическое обоснование разработки сайта «сервис совместного поиска попутчиков».....	127
4.1 Расчет себестоимости программного продукта.....	127
4.2 Эффективность разработки сайта.....	136
5. Безопасность жизнедеятельности. Организация рабочего места программиста.....	139
5.1 Характеристика условий труда рабочего места программиста.....	139
5.2 Требование к рабочему помещению программиста.....	140
5.2.1 Окраска помещений.....	140
5.2.2 Освещение рабочего места.....	141
5.2.3 Параметры микроклимата на рабочем месте.....	143
5.2.4 Требования к шуму и вибрации в рабочих помещениях.....	145
5.2.5 Режим труда программиста.....	147
5.3 Расчет освещенности рабочего места программиста.....	149
Заключение.....	154
Список использованных источников.....	155
ЛИСТИНГ ПРИЛОЖЕНИЯ.....	156
Приложение А.....	156
ВЕДОМОСТЬ ДИПЛОМНОГО ПРОЕКТА.....	204
Приложение Б.....	204
ГРАФИЧЕСКАЯ ЧАСТЬ.....	205

Приложение В.....	205
ПРЕЗЕНТАЦИЯ.....	206
Приложение Г.....	206

Введение

С каждым годом компьютеры и прочие «умные» устройства все сильнее проникают в нашу жизнь. В 2018 году в России насчитывалось приблизительно 74,4 миллиона компьютеров [1], что на 20% больше, чем в предыдущем. Кроме того, все большее распространение получают различные переносные электронные устройства: ноутбуки, смартфоны, планшеты и другие – которые по возможностям сравнимы с настольными компьютерами.

Вместе с развитием компьютерной техники все большее количество устройств подключаются к глобальной сети Интернет. В 2017 году 49% жителей России уже имели доступ к сети Интернет. Для сравнения, в 2016 году лишь 43,6% были подключены к сети Интернет. По прогнозам Минкомсвязи РФ к 2020 году уже 90 миллионов жителей нашей страны будут пользоваться ресурсами Интернет [1].

От развития информационных технологий перейдем к одной из острейших проблем крупных городов – рост количества личного транспорта. Несмотря на все удобства личного транспорта, легковые автомобили – это самый расточительный транспорт по сравнению с другими. Основная доля (63 %) экологического ущерба планете связана с автотранспортом. Величина ежегодного экологического ущерба от функционирования транспортного комплекса РФ составляет 3,4 млрд долларов США, или примерно 1,5 % валового национального продукта. Выбросы загрязняющих веществ в атмосферу от автотранспортных средств составили 12 190,7 тыс. тонн [2].

Проблема любого крупного города – транспортные пробки. Для борьбы с данной проблемой существует множество методов и каждый из них имеет свои преимущества и недостатки. Один из таких методов – carpool – совместное использование автомобиля для поездок в определенные пункты назначения или с

определенных пунктов отправки[3].

Совместное использование автомобилей началось зарождаться еще в 70-х годах XX века. С появлением интернета и средств мобильной связи, данное направление использования автомобилей все шире начинает применяться во многих странах мира. Власти некоторых городов оказывают содействие автовладельцам прибегающим к системе совместного использования автомобилей. В некоторых странах для водителей, использующих систему carpool, иногда предусматривается даже выделение отдельных полос движения на дорогах города.

Выигрыш от использования системы совместного использования автомобилей очевиден. Кроме экономии самих автомобилистов, пользующихся данной системой, на топливе, дорожных и парковочных сборах, эксплуатационных расходах,- данный метод позволяет существенно разгрузить некоторые участки дорожного движения, а соответственно и вероятность возникновения дорожных пробок, уменьшить выхлоп вредных газов в атмосферу городов, а также снизить психологическую нагрузку на участников дорожного движения.

Целью дипломного проектирования является разработка системы, помогающей пользователям находить себе попутчиков для проезда в пределах города или путешествия на дальние расстояния. Предполагается, что подобная система поможет снизить нагрузку на транспортную систему города Пенза.

В ходе дипломного проектирования был проведен анализ предметной области и бизнес-процессов, происходящих в ней, поставлена задача на разработку, проведен анализ требований на разработку, выбраны технологии разработки и выполнена разработка и тестирование программного обеспечения.

При выполнении дипломного проекта активно использовалась средство визуального моделирования объектно-ориентированных информационных систем StartUML.

В результате выполненной работы получено программное обеспечение, протестированное и полностью удовлетворяющее поставленным требованиям, а также пояснительная записка к дипломному проекту.

В настоящее время разработанное программное обеспечение в виде сервиса находится в опытной эксплуатации. Акт о внедрении сервиса приведен в приложении Б.

1. Анализ предметной области и постановка задачи на разработку

1.1 Обоснование разработки

Размеры современного города таковы, что добраться из одной точки в другую практически невозможно без использования транспортных средств. Наиболее удобный вид транспорта – личный автомобиль. Именно поэтому, начиная с 1970-го года, в России начинается рост количества личных автомобилей. Если в 70м году в среднем по России на тысячу человек приходилось около 10 автомобилей, то уже в начале 21го века данная цифра составляла 150. Ожидается, что к 2025 году на одну тысячу будет приходиться более 400 автомобилей. Динамика роста количества автомобилей приведена на рисунке 1.1

Рисунок 1.1 Количество автомобилей на 1000 человек
(в среднем по России)

Стоит отметить, что транспортная сеть большинства городов была сформирована в середине 20го века и ее пропускная способность изначально не была рассчитана под такой транспортный поток. Именно поэтому сейчас большинство крупных городов столкнулись с такой проблемой как пробки (рисунок 1.2).



Рисунок 1.1 – Транспортная инфраструктура РФ

Даже в Пензе, далеко не самом большом городе России, в час пик на многих участках транспортной сети возникают заторы, которые значительно снижают среднюю скорость транспортного потока.

Кроме проблем с транспортными пробками, автотранспорт таит в себе еще одну угрозу – экологическую. Каждый автомобиль выбрасывает в окружающую среду большое количество вредных веществ. Среди них оксид азота, канцерогены, свинец, бром, хром и их соединения. Данные вещества могут стать причиной различных заболеваний, в том числе рака легких и нарушений сердечнососудистой системы. На рисунке 1.3 представлена сравнительная диаграмма долей выбросов загрязняющих веществ автотранспортом в наиболее крупных городах мира.

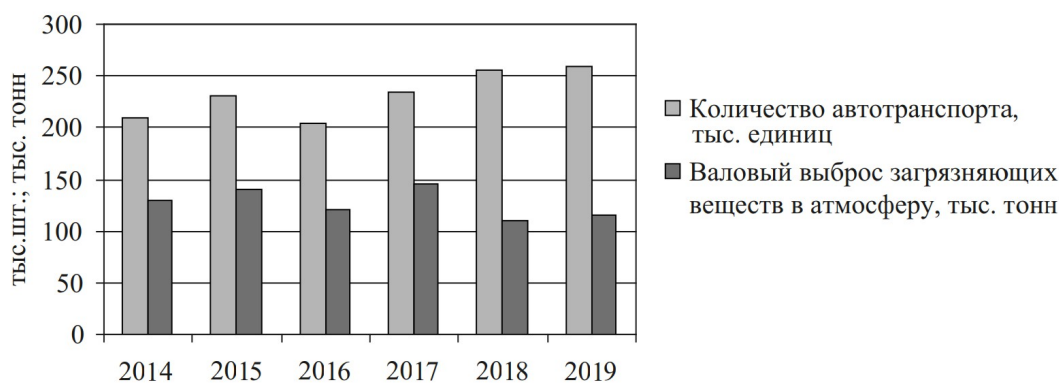


Рисунок 1.2 – Доля выбросов загрязняющих веществ от автотранспорта

Многие крупные города мира, столкнувшись с перечисленными проблемами, начали активно бороться с личным транспортом. Существует несколько способов уменьшить количество личных автомобилей на дорогах города.

Одним из таких способов является Carpool – совместное использование автотранспорта. Суть метода заключается в том, что автотранспорт начинает использоваться рациональнее. Нередко на дорогах города можно заметить автомобили с одним пассажиром - водителем. Очевидно, что в таком случае автомобиль используется нерационально. К тому же, наверняка найдется множество людей, маршрут которых совпадает с данным автомобилем. Но им приходится пользоваться либо собственным личным транспортом, либо общественным. Не секрет, что в городах России ситуация с общественным транспортом оставляет желать лучшего. В часы пик общественный транспорт часто не справляется с потоком пассажиров, что доставляет неудобство всем участникам движения.

В качестве наглядного примера существования проблемы в городе Пенза можно привести карту пробок, предоставляемую картографическим сервисом Яндекс.Карты. На карте изображено среднее состояние дорог в 16:30 по пятницам.

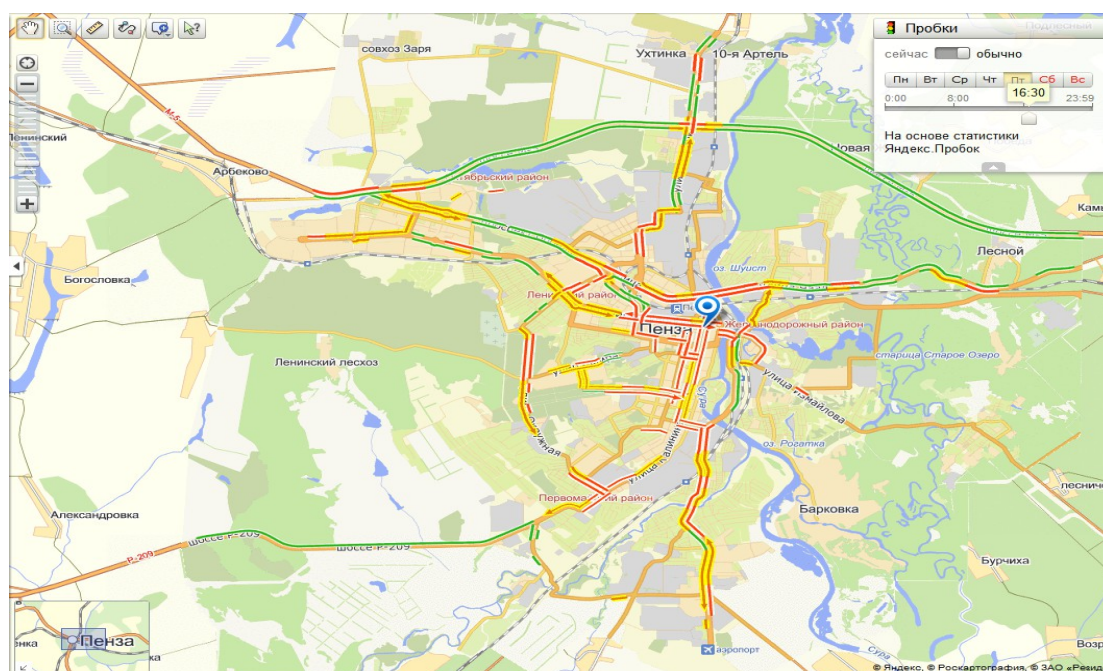


Рисунок 1.4 – Карта пробок в городе Пенза по данным сервиса Яндекс.Карты

На приведенном изображении видно, что на большей части главных дорог города движение транспортного потока затруднено. Это говорит о наличии проблемы транспортных пробок в нашем городе.

Логичным выходом из данной ситуации является оптимизация использования личного транспорта. Необходимо свести вместе водителя, который каждый день ездит на автомобиле один, и пассажиров, маршрут которых совпадает с маршрутом водителя. При этом решение должно быть удобным для обеих сторон, так как только это позволит сделать совместное использование транспорта популярным.

Выигрыш от использования системы совместного использования автомобилей очевиден. Кроме экономии самих автомобилистов, пользующихся данной системой, на топливе, дорожных и парковочных сборах, эксплуатационных расходах, - данный метод позволяет существенно разгрузить некоторые участки дорожного движения, а соответственно и вероятность возникновения дорожных пробок, уменьшить выхлоп вредных газов в атмосферу городов, а также снизить психологическую нагрузку на участников дорожного движения.

Хотя в России данный метод использования автомобилей имеет крайне низкие показатели, однако, уже и в нашей стране появляются некоторые компании предоставляющие услуги по системе carpool. Кроме того, возникают интернет ресурсы, позволяющие производить поиск гражданами попутчиков для совместного проезда в транспортном средстве к месту работы, учебы, службы и т.д.[3].

1.2 Анализ бизнес-процессов предметной области

В данном дипломном проекте требуется разработать онлайнсервис поиска попутного транспорта и пассажиров. Сервис должен обеспечивать пользователям возможность осуществлять регистрацию собственных маршрутов (в случае если пользователь является водителем личного автомобиля) и поиск подходящего маршрута (в случае если пользователь выступает в роли пассажира).

При этом сервис должен иметь развитую систему поиска маршрутов и

подбора альтернативных маршрутов. При поиске маршрута должны учитываться различные варианты сочетания искомого маршрута и уже существующих. Например, искомый маршрут может являться участком существующего маршрута. Или искомый маршрут отличается от одного из существующих только расположением конечных точек, но разница может быть не критична для пользователя. При реализации проекта должны активно использоваться различные картографические сервисы, предоставляющие открытый API. Это поможет сделать конечный продукт максимально удобным для конечных пользователей.

На рисунке 1.5 представлена диаграмма деятельности бизнес-процесса поиска маршрута и участия в нем. Данный бизнес-процесс включает в себя авторизацию пользователя в системе и сам процесс поиска маршрута и участия в нем. В том случае, если пользователь еще не зарегистрирован в системе, он может пройти регистрацию. После регистрации он автоматически будет авторизован и перенаправлен на страницу поиска маршрута.

Процесс поиска маршрута отличается для пользователя, который выступает как пассажир, и для пользователя, выступающего как водитель транспортного средства. В первом случае, пользователю достаточно найти маршрут и подать заявку на участие в нем. В случае, если пользователь выступает водителем автомобиля, ему необходимо указать данные о своем транспортном средстве. В остальном процесс поиска маршрута идентичен для всех пользователей: необходимо выбрать желаемый маршрут и отправить заявку на участие в нем. После того, как заявка отправлена, владелец маршрута, т.е. тот пользователь, который создал маршрут, должен рассмотреть заявку и вынести решение: принять заявку или отклонить. Отправитель заявки будет уведомлен о решении владельца маршрута специальным уведомлением. Если заявка одобрена, то ее отправитель включается в список участников маршрута, после чего ему становится доступным список других участников.

На рисунке 1.5 приведена диаграмма деятельности бизнес-процесса добавления нового маршрута в систему. Как и в бизнес-процессе поиска

маршрута, данный бизнес-процесс включает в себя процесс авторизации. Также бизнес-процесс имеет различия в зависимости от роли пользователя. Для пользователя, который выступает как водитель транспортного средства, необходимо выбрать указать данные об автомобиле, на котором будет совершена поездка по новому маршруту. Для удобства использования, информация о транспортном средстве сохраняется в системе, и процесс добавления сведений об автомобиле достаточно пройти один раз для каждого автомобиля. После этого достаточно просто выбрать нужный автомобиль из списка доступных.

Для пользователя, выступающего в роли пассажира, есть возможность указать требования к автомобилю, его водителю и остальным пассажирам. Данная информация не является обязательной, а служит только для уточнения последующего поиска подходящего маршрута.

На рисунке 1.6 представлена диаграмма деятельности бизнес-процесса авторизации пользователя в системе. Данный процесс включает в себя две взаимоисключающих ветви: авторизация зарегистрированного пользователя и регистрация нового. Для авторизации зарегистрированного пользователя достаточно ввести имя пользователя и пароль, указанные при регистрации в системе. После проверки введенных данных произойдет либо перенаправление на главную страницу системы с измененными правами авторизованного пользователя, либо вывод сообщения об ошибке, в котором содержится справочная информация по возможным путям исправления ошибки. Если пользователь еще не зарегистрирован в системе, ему необходимо пройти процедуру регистрации.

Регистрация состоит из двух последовательных шагов. На первом шаге пользователь указывает личные данные, необходимые для дальнейшей работы в системе. После этого данные отправляются на сервер, где проходят проверку на корректность информации.

Если проверка прошла успешно, пользователю на указанный им адрес электронной почты отправляется письмо, содержащее ссылку, переход по которой подтвердит право владения пользователем указанным почтовым ящиком.

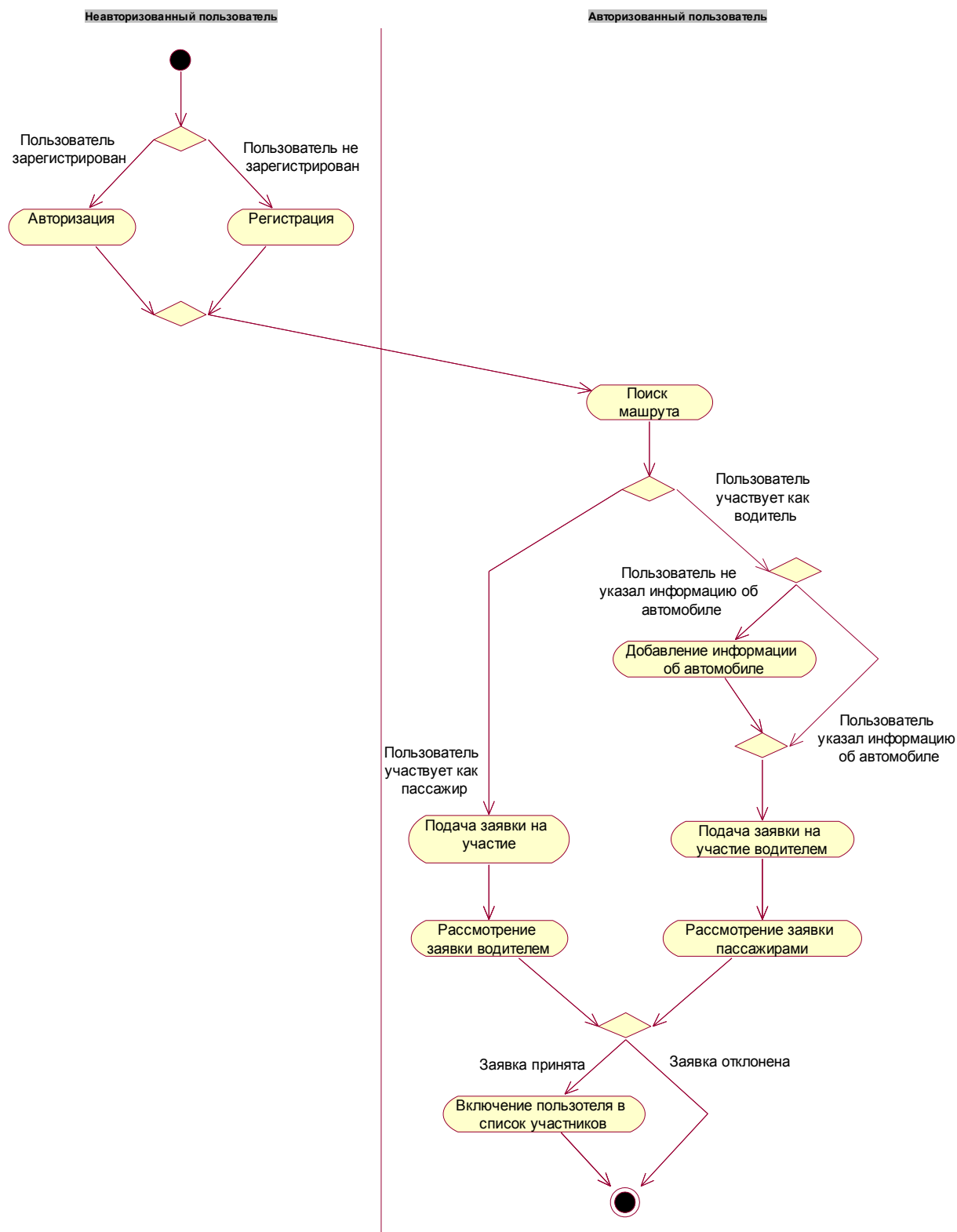


Рисунок 1.5 – Диаграмма деятельности бизнес-процесса поиска маршрута и участия в нем.

После того, как пользователь перейдет по указанной ссылке, он автоматически будет перенаправлен на главную страницу сервиса с измененными правами авторизованного пользователя.

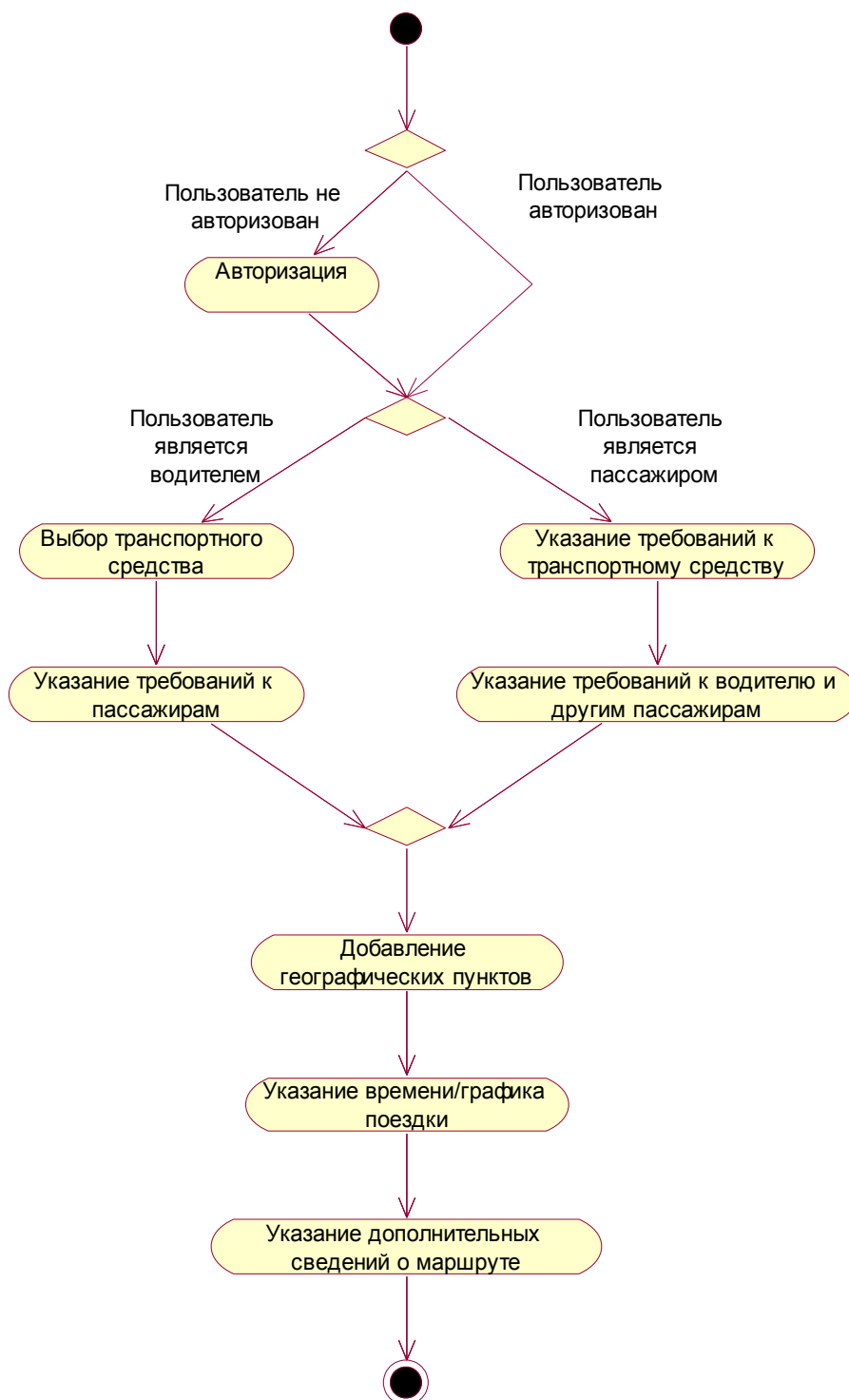


Рисунок 1.6 - Диаграмма деятельности бизнес-процесса добавления маршрута



Рисунок 1.7 - Диаграмма деятельности авторизации пользователя в системе

1.3 Сравнительный анализ существующих аналогов

Прежде чем сформировать подробные требования к разрабатываемой системе, был произведен анализ уже существующих систем. Данный анализ необходим для того, чтобы при разработке системы учесть преимущества и недостатки существующих систем. Так как практика carpool появилась довольно давно, то и в Интернете на данный момент есть большое количество сайтов, которые позволяют найти пассажиру и водителю друг друга. Но не все сайты одинаково популярны среди пользователей.

Для того, чтобы выделить наиболее популярные сайты, был использован показатель посещаемости – количество уникальных посетителей за одни сутки. Также при сравнении учитывались следующие показатели:

- посещаемость ресурса;
- отсутствие рекламы;
- доступность сервиса;
- качество интерфейса страниц ресурса (дизайн, качество изображений и др.) – оценка по пятибалльной шкале: 1 – плохо, 5 - отлично;
- наличие и качество поиска по сайту;
- наполнение информацией (количество существующих маршрутов, предложений, заявок и пр.);
- направленность на конкретный географический регион;
- наличие функций общения между пользователями системы;
- сложность и необходимость регистрации в системе.

В таблице таблица 1.1 приведены существующие решения, которые были взяты для сравнительного анализа.

Таблица 1.1 Существующие решения сервисов поиска попутчика

Название	Производитель	Адрес ресурса	Посещаемость (уник.польз. / сут.)
«Доедем вместе! Сервис поиска попутчиков и попутных машин»	ООО «ПопутчикСервис»	http://www.doedemvmeste.ru	445
BlaBlaCar Сервис поиска попутчиков и попутных машин»	Фредерик Маззелла Николас Брюссон и Франсис Наппез Франция	https://www.blablacar.ru	5327
«Махнем - сервис поиска попутчиков»	«MIT-Group - Интернет решения»	http://mahnem.ru	241
«Попутчик – поиск попутков и попутчиков»	ООО «Poputchik»	http://www.poputchik.ru	520

Сервис поиска попутного транспорта и пассажиров «Доедем вместе!».

Сервис имеет наибольшую популярность среди пользователей сети Интернет. Однако качество исполнения сервиса оставляет желать лучшего. На рисунке 1.8 приведена главная страница сервиса.

Посещаемость: 445 уникальных посетителей в сутки.

Реклама: три рекламных блока, слева – вертикальный блок, в центре и снизу – горизонтальные блоки.

Доступность сервиса: полностью бесплатный.

Качество интерфейса: 3.

Наличие и качество поиска по сайту: глобального поиска нет. Есть разделение на «Поиск попутных пассажиров» и «Поиск попутной машины», которые работают как фильтры для общего списка.

Наполнение информацией: большое количество зарегистрированных маршрутов и заявок на поиск попутных пассажиров и автомобилей.

Направленность на конкретный географический регион: нет.

Наличие функций обмена сообщениями: для зарегистрированных пользователей доступен обмен текстовыми сообщениями.

Сложность и необходимость регистрации: регистрация необходима для добавления заявок на поиск попутных пассажиров или автомобилей. Поиск среди существующих заявок возможен без регистрации. Регистрация проходит в один шаг, после заполнения формы становятся доступны функции зарегистрированного пользователя.

DOED E M V M E S T E . R U

Найти попутных пассажиров

Найти попутную машину

EMail:

Пароль:

[Забыли пароль?](#) [Войти](#)

[Регистрация](#)

Найти:

[Попутных пассажиров](#)

[Попутную машину](#)

Разместить заявку о:

[Возможности взять попутчиков](#)

[Желания найти попутку](#)

[Главная \(о проекте\)](#)

[Часто задаваемые вопросы](#)

[Техника безопасности](#)

[Обратная связь](#)

Дерево участия

[Ссылка на регистрацию ИП и ООО в Москве](#)

Нам 8 лет! Уже более 40 000 человек в России (и не только) стали пользователями нашего сайта!

Здесь Вы можете найти [попутную машину \(попутку\)](#) или [попутных пассажиров](#).

Идея сервиса довольно проста. Задумайтесь, сколько автомобилистов ежедневно бороздит просторы нашей страны и даже выезжают за ее пределы на собственном железном коне? При этом в большинстве случаев автомобиль не обременен большим количеством пассажиров и почти всегда в нем есть свободные места. А теперь задумайтесь о том, какое количество людей также нуждаются в том, чтобы добраться «из пункта А в пункт Б»? Очень часто такое путешествие сопряжено с судорожным поиском уже давно раскупленных билетов, нервотрепой, и, в итоге, зачастую, оборачивается поездкой в неудобное время, день и по завышенной цене. Многие люди в целях экономии вообще ограничивают себя в поездках и остаются дома вместо того, чтобы повеселить родных, друзей, да и просто «проветриться» в другой город!

А теперь подумайте, а что если помочь этим потенциальным [пассажирам](#) и [автомобилистам](#), планирующим поездку в [одном и том же направлении](#), [найти друг друга](#)? Что если обеспечить им [дополнительные факторы безопасности](#) (см. в разделе [техника безопасности](#)) и тем самым успокоить опасения относительно поездки с незнакомым водителем/пассажиром? Это было бы весьма взаимовыгодное сотрудничество для обоих!

Во-первых, скорее всего, [стоимость такой поездки для пассажира будет заметно ниже](#), чем на других видах транспорта.

Во-вторых, это возможность [для водителя частично окупить затраты на поездку](#).

В-третьих, для обоих сторон это возможность [сэкономить поездку общением и провести время веселее!](#)

Совместные поездки приобретают все большую популярность во всем мире. К примеру, в Германии около **20 000 человек ежедневно** передвигаются таким способом!

Конечно же, многим из нас не все равно с кем отправиться в путь, к примеру, кто-то не выносит курящих попутчиков, для кого-то имеет значение возраст и пол. Поэтому важно обеспечить [максимальный учет пожеланий обеих сторон](#), и на сервисе "[Доедем вместе!](#)" мы это делаем!

Желаем Вам счастливого пути! ☺

Поделиться: [f](#) [t](#) [v](#) [p](#)

Последние 30 заявок посетителей сервиса "[Доедем вместе!](#)":

Кто	Ищет	Откуда	Куда	Дата, время выезда	Цена	Примечания
Андрей	Попутных пассажиров	буденновск <small>(Российская Федерация)</small>	Киров <small>(Российская Федерация)</small>	15.05.19 09:00	2500	

Рисунок 1.8- Главная страница ресурса «Доедем вместе»

Сервис поиска попутного транспорта и пассажиров «BlaBlaCar».

Главной особенностью данного сервиса является отсутствие направленности на какой-либо конкретный масштаб маршрутов: есть возможность создавать маршруты как в пределах одного города, так и по всему миру. На рисунке 1.9 приведена главная страница сервиса.

URL-ссылка: <https://www.blablacar.ru>

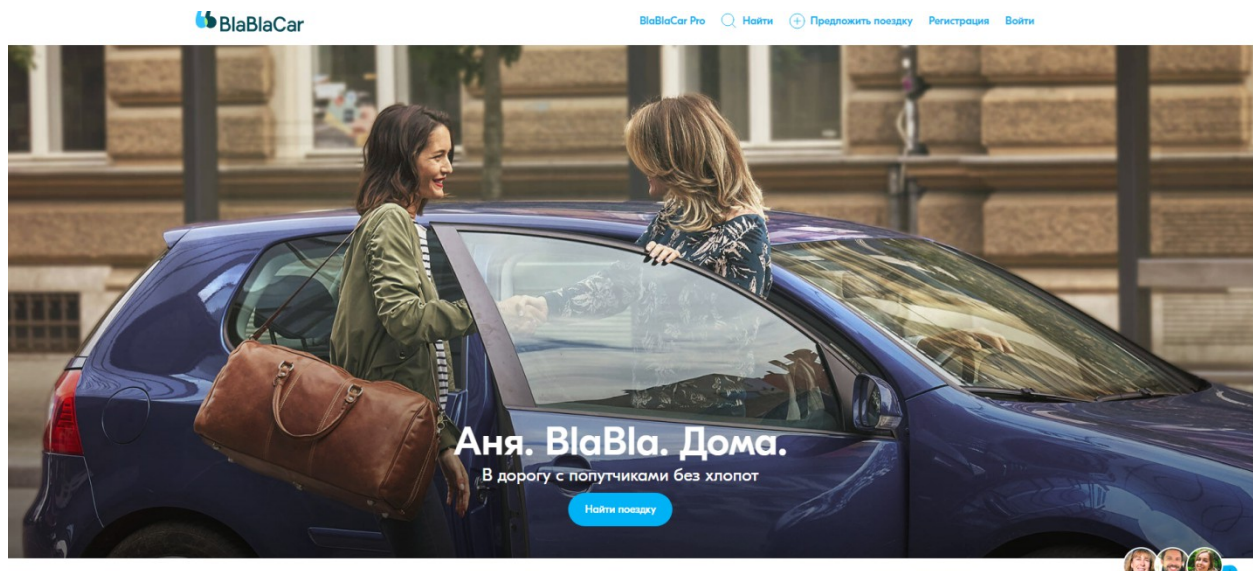


Рисунок 1.9 – Главная страница ресурса «BlaBlaCar»

Посещаемость: 5327 уникальных посетителей в сутки.

Реклама: присутствует.

Доступность сервиса: платный с 2019 года.

Качество интерфейса: 5.

Наличие и качество поиска по сайту: поиск полноценный.

Наполнение информацией: большое количество зарегистрированных маршрутов и заявок на поиск попутных пассажиров и автомобилей.

Направленность на конкретный географический регион: нет.

Наличие функций обмена сообщениями: для зарегистрированных пользователей доступен обмен текстовыми сообщениями, графическим и видеоконтентом.

Сложность и необходимость регистрации: регистрация необходима для добавления заявок на поиск попутных пассажиров или автомобилей. Регистрация проходит в три шага, после заполнения формы становятся доступны функции зарегистрированного пользователя. Зарегистрированные пользователи могут добавлять друг друга в друзья, создавать заметки, создавать фотоальбомы и загружать в них фотографии, обмениваться сообщениями.

Сервис поиска попутного транспорта и пассажиров «Махнем». Данный сервис является наиболее качественным с эстетической точки зрения, а также в плане интеграции с социальными сетями.

На рисунке 1.10 приведена главная страница сервиса.

URL-ссылка: <http://mahnem.ru>

махнем! Поиск попутчиков Прогулки Изучение языков

Логин или email:

Пароль:

Войти

☐ запомнить

[Создать аккаунт](#)
[Забыли пароль?](#)

Попутчики
Друзья для прогулок
Собеседники

Махнем.ру объединяет путешественников для поиска попутчиков и попутчиц на отдых. Чтобы попутчик нашелся быстрее, [добавьте](#) объявление.

Узнайте больше о том, [как найти хорошего попутчика](#).

Путешествуй весело – найди попутчиков!

Найди новых друзей
выбери направление и интересы

Вместе – дешевле
раздели затраты на отель и аренду авто

Я ищу: 18+ Когда: – Путешествие:

☐ уже отдыхаю ?

Откуда едем: ± 50 km Куда:

Найти

Дать объявление

[Вьетнам](#) [Испания](#) [Мальдивы](#) [Таиланд](#) [Море](#)
[Греция](#) [Италия](#) [ОАЭ](#) [Турция](#) [Выходные](#)
[Доминикана](#) [Кипр](#) [Россия](#) [Франция](#) [Европа](#)

Топ-объявления

Как сюда попасть

[Куда угодно](#)
Мужчину , 30-40 для знакомства , общения , и возможно поездки к...

[Испания, Италия, Россия, Франция](#)
Двое состоятельных мужчин (32 и 36 лет) в поисках двух...

[Кипр, Тунис, Турция](#)
Нужна попутчица на Кипр или иные моря, в июле. Веселая и мудрая....

[Доминикана, Камбоджа, Мексика, Монако, ОАЭ...](#)
Хочу на солнышко! ☀️

Рисунок 1.10 – Главная страница ресурса «Махнем»

Посещаемость: 241 уникальный посетитель в сутки.

Реклама: отсутствует.

Доступность сервиса: полностью бесплатный.

Качество интерфейса: 5.

Наличие и качество поиска по сайту: доступны два варианта поиска: простой – пункты отправки и назначения и расширенный – время поездки, фильтр по рейтингу и полу водителя и стоимости проезда.

Наполнение информацией: небольшое количество зарегистрированных пользователей, среднее количество маршрутов и заявок.

Направленность на конкретный географический регион: нет.

Наличие функций обмена сообщениями: для зарегистрированных пользователей доступен обмен текстовыми сообщениями.

Сложность и необходимость регистрации: доступны два варианта регистрации: стандартная в 2 шага, и через социальную сеть «ВКонтакте».

Сервис поиска попутного транспорта и пассажиров «Попутчик». Сервис является полностью бесплатным, о чем говорится на странице «О проекте». Также сервис обладает наименее нагруженным интерфейсом и простой регистрацией. На рисунке 1.11 представлена главная страница сервиса.

URL-ссылка: <http://www.poputchik.ru>

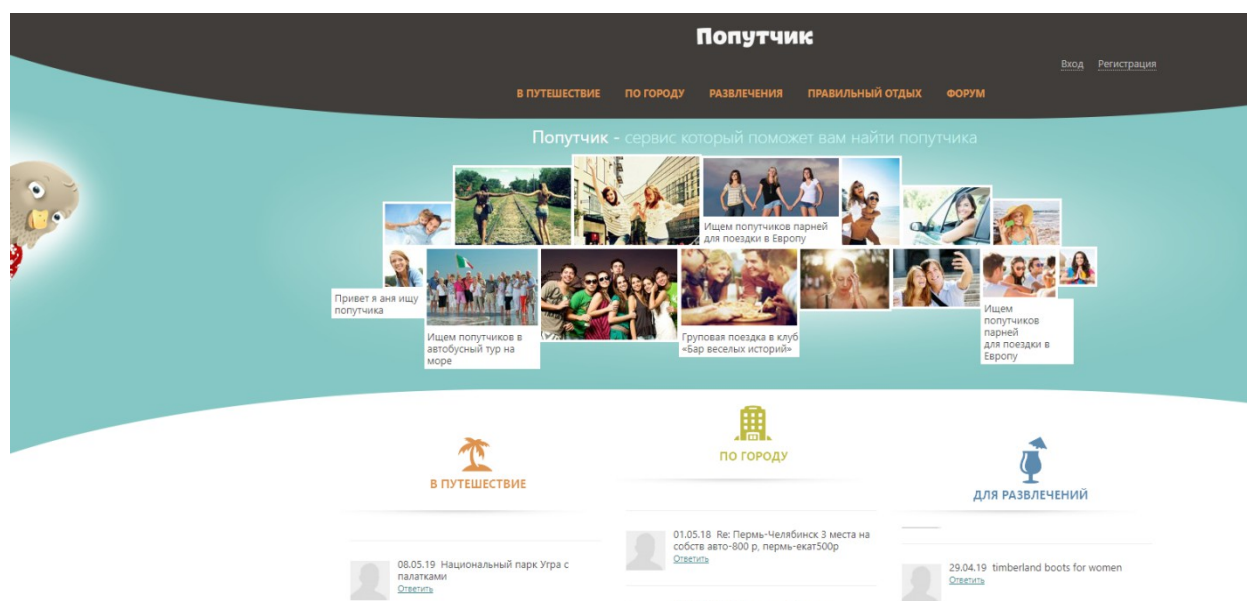


Рисунок 1.31 – Главная страница ресурса «Давай со мной»

Посещаемость: 520 уникальных посетителей в сутки.

Реклама: отсутствует.

Доступность сервиса: полностью бесплатный.

Качество интерфейса: 4.

Наличие и качество поиска по сайту: поиск разделен на две категории – «Ищу водителей» и «Ищу пассажиров». Поиск возможен только по точке отправления и прибытия. Для города Москва возможно уточнение по станции метро. Также возможно дополнительно указать дату поездки.

Наполнение информацией: небольшое количество зарегистрированных пользователей, среднее количество маршрутов и заявок по Москве и Санкт-Петербургу.

Направленность на конкретный географический регион: нет.

Наличие функций обмена сообщениями: доступна отправка текстовых сообщений без регистрации (с указанием email) владельцам маршрутов или заявок.

Сложность и необходимость регистрации: регистрация происходит в процессе создания маршрута или заявки. Дополнительных функций для зарегистрированных пользователей нет.

Результаты сравнительного анализа существующих аналогов. По результатам проведенного анализа можно составить список функций, реализованных в каждом проекте удачно или нет. Каждый элемент полученного списка можно охарактеризовать как удачное решение или нет. Исходя из этой характеристики при реализации дипломного проекта, удачные решения могут быть включены в проект. В таблице 1.2 приведены результаты сравнительного анализа (анализ производился с помощью интернет сервиса <https://builtwith.com>).

К удачным решениям можно отнести следующее:

- быстрая регистрация и возможность авторизации через социальные сети (сторонние сервисы) – в настоящее время все большую популярность получает способ авторизации через сторонние сервисы, в частности – через социальные сети. Не секрет, что большая часть пользователей сети Интернет зарегистрированы в различных социальных сетях, поэтому для них крайне удобно не проходить каждый раз регистрацию, а использовать свой профиль в какой-либо социальной сети. Данная функция обязательно должна быть реализована в любом сервисе, ориентированном на рядового пользователя;

- интеллектуальная система поиска – грамотно реализованная система поиска позволяет пользователю значительно сократить затраты времени на поиск нужной информации на сайте. Кроме того, релевантные результаты поиска повышают степень доверия пользователя к системе в целом;

- развитая система коммуникации пользователей – все из проанализированных систем имеют систему обмена сообщениями. Очевидно, что это необходимая часть любого сервиса, ориентированного на взаимодействие пользователей между собой.

Кроме обмена сообщениями, также полезной функцией является возможность создавать списки друзей – с помощью этой функции пользователи могут выделять из общего числа пользователей системы тех, с кем они контактируют наиболее часто.

При анализе существующих решений были выявлены некоторые недостатки, которые могут существенно повлиять на популярность сервиса среди пользователей.

Отсутствие возможности работы с картой – в настоящее время в русскоязычной части сети Интернет появилось несколько развитых картографических сервисов, которые предоставляют бесплатный доступ к API, поэтому использование в проекте инструментов работы с картой не составляет больших трудностей. Однако, положительный эффект от использования данных инструментов налицо – карта наиболее удобна для указания маршрута следования, а также для поиска различных объектов. Поэтому инструменты

разрабатываемого проекта должны сочетать обычные способы внесения информации с картографическими способами.

Отсутствие интеграции с социальными сетями – в настоящий момент социальные сети получают все большее распространение, практически каждый пользователь сети Интернет зарегистрирован в той или иной социальной сети. С точки зрения пользователя системы, удобно реализовать импорт всей необходимой информации из профиля пользователя в социальной сети. Это, во-первых, экономит время пользователя на повторное внесение информации при регистрации, и, во-вторых, гарантирует истинность указанных сведений. Кроме импорта личной информации пользователя, также возможен и поиск друзей из социальной сети среди пользователей системы. Данная функция также экономит время пользователя, которое он потратит на ручной поиск своих друзей.

Отсутствие мобильной версии сайта – в настоящий момент все большее распространение получают мобильные устройства – смартфоны и планшеты, которые могут обеспечить полноценный доступ к Интернет-ресурсам.

Но в силу особенностей данных устройств, для них необходима отдельная версия сайта, которая должна быть специально адаптирована для использования именно на мобильных устройствах. Наличие мобильной версии сайта является обязательным для любого сервиса, ориентированного на рядовых пользователей.

1.4 Постановка задачи на разработку

В данной работе поставлена задача разработать сервис взаимного поиска попутного транспорта и пассажиров. Данная система должна предоставлять пользователям возможность регистрации в системе, создания маршрутов поездок, управления своими маршрутами, составления заявок на поездки, управления своими заявками, поиска необходимой информации по сайту, а также взаимодействовать с другими пользователями системы.

Система поиска попутчика и попутного транспорта должна быть построена по трехуровневой архитектуре и представлять собой веб-приложение. Данное веб-приложение должно быть спроектировано с учетом возможного дальнейшего расширения функциональности. Приложение должно легко масштабироваться, так как нагрузка будет возрастать пропорционально росту количества пользователей.

Разрабатываемая система должна иметь интеграцию с наиболее популярными социальными сетями - ВКонтакте, Facebook и Twitter. Интеграция заключается в возможности регистрации и авторизации через профиль в социальной сети, импорта списка друзей и возможности публикации создаваемых маршрутов или заявок в социальную сеть.

Также в ходе разработки системы должна быть спроектирована и реализована мощная система поиска по сайту, которая должна позволять пользователю находить любую интересующую его информацию: маршруты, заявки, пользователей и другую информацию. Система поиска также должна включать в себя систему фильтрации результатов.

В инструментах работы с маршрутами и заявками должны активно использоваться картографические сервисы, предоставляющие открытый интерфейс взаимодействия: Яндекс. Карты, GoogleMaps и аналогичные.

На рисунке Рисунок 1. представлена модель предметной области интернет-сервиса поиска попутного транспорта и пассажиров.

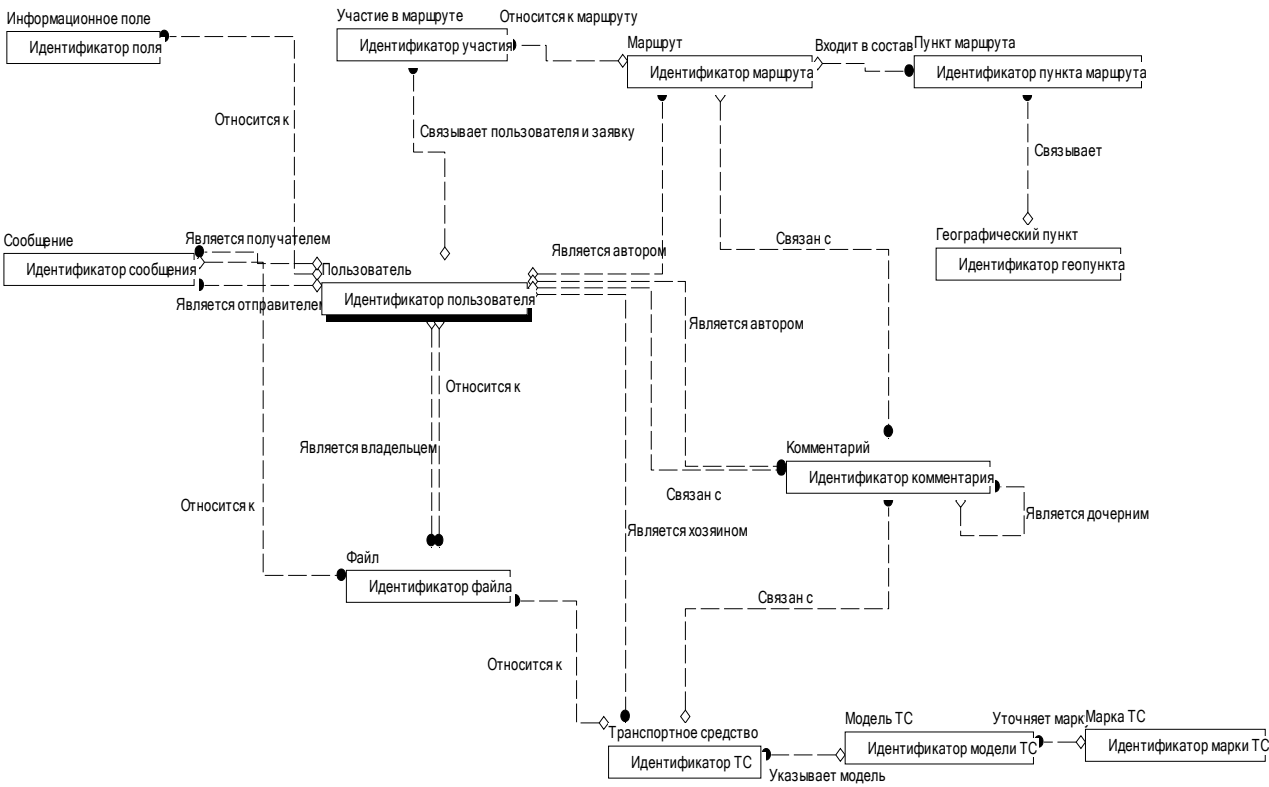


Рисунок 1.12 – Модель предметной области сервиса

В таблице 1.3 приводятся количественные характеристики сущностей, а также наличие и тип связей с другими сущностями предметной области.

Таблица 1.3 Модель предметной области сервиса

Название сущности	Кол-во экземпляров сущностей	Ограничение доступа	Тип связи с другими сущностями
1	2	3	4

Пользователь	10000	Полный доступ	1:M (Сообщение) 1:M (Файл) 1:M (Комментарий) 1:M (Транспортное средство) 1:M (Информационное поле) 1:M (Участие в маршруте) 1:M (Маршрут)
Сообщение	100000	Добавление, чтение	M:1 (Пользователь) 1:M (Файл)
Информационное поле	50000	Полный доступ	M:1 (Пользователь)
Файл	100000	Полный доступ	M:1 (Сообщение) M:1 (Пользователь) M:1 (Транспортное средство)
Участие в маршруте	100000	Полный доступ	M:1 (Пользователь) M:1 (Маршрут)
Маршрут	50000	Добавление, изменение, чтение	1:M (Участие в маршруте) M:1 (Пользователь) 1:M (Комментарий) 1:M (Пункт маршрута)
Пункт маршрута	200000	Полный доступ	M:1 (Маршрут) M:1 (Географический пункт)
Географический пункт	150000	Добавление, чтение, изменение	1:M (Пункт маршрута)

В таблице 1.4 приведено описание атрибутов сущности «Пользователь».

Таблица 1.2 – Атрибуты сущности «Пользователь»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Фамилия	Символьный	А-я	
Имя	Символьный	А-я	
Отчество	Символьный	А-я	
Дата рождения	Дата		
Логин	Символьный	Любой символ, длина не менее 6 знаков	АК
Пароль	Символьный	Любой символ, длина не менее 6 знаков	АК

Роль	Целый	1-3	
------	-------	-----	--

В таблице 1.5 приведено описание атрибутов сущности «Сообщение».

Таблица 1.3 – Атрибуты сущности «Сообщение»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Текст	Символьный		
Статус	Целый	1-3	
Дата отправки	Дата		АК
Отправитель	Целый	Положительное целое	АК
Получатель	Целый	Положительное целое	АК

В таблице 1.6 приведено описание атрибутов сущности «Информационное поле».

Таблица 1.4 – Атрибуты сущности «Информационное поле»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Название	Символьный	А-я	АК
Тип	Целый	1-10	
Значение	Символьный		
Пользователь	Целый	Положительное целое	АК

В таблице 1.7 приведено описание атрибутов сущности «Файл».

Таблица 1.5 – Атрибуты сущности «Файл»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Название	Символьный		АК
Дата создания	Дата		АК
Тип файла	Целый	1-4	АК
Путь к файлу	Символьный	a-z, /, \, -, ., ,	АК

Автор	Целый	Положительное целое	
Сообщение	Целый	Положительное целое	
Транспортное средство	Целый	Положительное целое	
Пользователь	Целый	Положительное целое	

В таблице 1.8 приведено описание атрибутов сущности «Участие в маршруте».

Таблица 1.6 – Атрибуты сущности «Участие в маршруте»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Дата создания	Дата		
Статус	Целый	1-3	
Тип	Целый	1-3	
Маршрут	Целый	Положительное целое	АК
Пользователь	Целый	Положительное целое	АК

В таблице 1.9 приведено описание атрибутов сущности «Маршрут».

Таблица 1.7 – Атрибуты сущности «Маршрут»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Название	Символьный		АК
Описание	Символьный		
Дата создания	Дата		АК
Дата отправления	Дата	Не ранее даты создания	
Дата прибытия	Дата	Не ранее даты отправления	
Автор	Целый	Положительное целое	АК

В таблице 1.10 приведено описание атрибутов сущности «Пункт маршрута».

Таблица 1.8 – Атрибуты сущности «Пункт маршрута»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Название	Символьный		
Тип пункта	Целый	1-4	
Порядковый номер	Целый	Положительное целое	
Дата прибытия	Дата	Не ранее даты отправления	
Географический пункт	Целый	Положительное целое	АК
Маршрут	Целый	Положительное целое	АК

В таблице 1.11 приведено описание атрибутов сущности «Географический пункт».

Таблица 1.9 – Атрибуты сущности «Географический пункт»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Название	Символьный		
Координаты	Символьный		АК

В таблице 1.12 приведено описание атрибутов сущности «Комментарий».

Таблица 1.10 – Атрибуты сущности «Комментарий»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Заголовок	Символьный		
Текст	Символьный		
Дата создания	Дата		АК
Родительский комментарий	Целый	Положительное целое	

Автор	Целый	Положительное целое	АК
Транспортное средство	Целый	Положительное целое	АК
Маршрут	Целый	Положительное целое	АК
Пользователь	Целый	Положительное целое	АК

Таблица 1.113 – Атрибуты сущности «Модель транспортного средства»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Название	Символьный		АК
Количество мест	Целый	Положительное целое	
Марка транспортного средства	Целый	Положительное целое	АК

В таблице 1.1Таблица 1.124 приведено описание атрибутов сущности «Марка транспортного средства».

Таблица 1.124 – Атрибуты сущности «Марка транспортного средства»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Название	Символьный		АК

1.5 Функциональные возможности онлайн сервиса

Все пользователи разрабатываемого сервиса будут делиться на три группы: гости, авторизованные пользователи и администраторы системы. Отличие между указанными группами заключается в наличии или отсутствии прав на доступ к инструментам сервиса. Пользователи группы «Гость» обладает самыми минимальными правами – лишь просмотр списка маршрутов и информации о выбранном маршруте. Права пользователей

группы «Авторизованный пользователь» предполагают доступ к страницам управления маршрутами, друзьями, сообщениями, а также личным профилем. Пользователи группы «Администратор системы» обладают полными правами на доступ к любому разделу сервиса.

В соответствии с данным делением для каждой группы пользователей были разработаны диаграммы вариантов использования. На рисунке 1.13 приведена диаграмма вариантов использования для группы пользователей «Гость».

В таблице 1.15 приведено описание вариантов использования для пользователей группы «Гость».

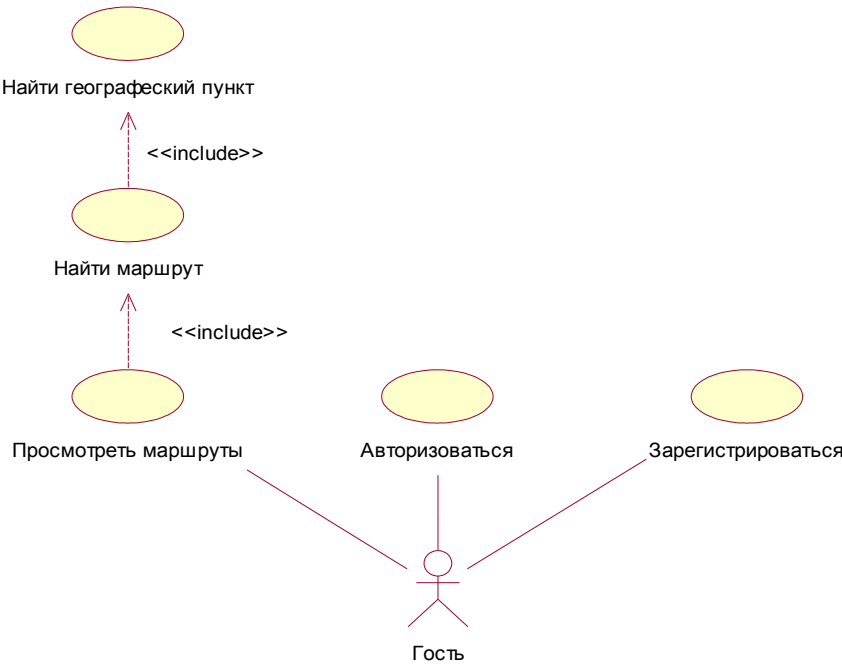


Рисунок 1.13 - Диаграмма вариантов использования для роли «Гость»

Таблица 1.15 – Описание вариантов использования для роли «Гость»

Название	Описание
Просмотреть маршруты	Просмотреть список всех маршрутов, зарегистрированных в системе и актуальных на момент просмотра

Найти маршрут	С помощью системы поиска выбрать из общего списка тот маршрут, который наиболее подходит пользователю
Найти географический пункт	С помощью системы поиска выбрать географический пункт. Выбор происходит по первым буквам названия пункта, либо по географическим признакам.
Авторизоваться	Успешно пройти процедуру авторизации на сайте.
Зарегистрироваться	Успешно пройти процедуру регистрации на сайте.

На рисунке 1.14 приведена диаграмма вариантов использования для группы пользователей «Авторизованный пользователь».

В таблице 1.16 приведено описание вариантов использования для пользователей группы «Авторизованный пользователь».

Таблица 1.16 – Описание вариантов использования для роли «Авторизованный пользователь»

Название	Описание
1	2
Просмотреть маршруты	Просмотреть список всех маршрутов, зарегистрированных в системе и актуальных на момент просмотра
Просмотреть свои маршруты	Просмотреть список маршрутов, автором которых является текущий авторизованный пользователь
Найти маршрут	С помощью системы поиска выбрать из общего списка тот маршрут, который наиболее подходит пользователю
Найти географический пункт	С помощью системы поиска выбрать географический пункт. Выбор происходит по первым буквам названия пункта, либо по географическим признакам.
Просмотреть маршрут	Просмотреть подробные сведения о выбранном маршруте.
Отправить заявку на участие	Создать и отправить заявку на участие в выбранном маршруте
Принять заявку на участие	Просмотреть список всех заявок на участие в выбранном маршруте и принять решение о принятии или отклонении каждой
Добавить маршрут	Разместить и опубликовать информацию о маршруте

Просмотреть сообщения	Просмотреть список сообщений, в которых текущий авторизованный пользователь является отправителем или получателем
Написать сообщение	Отправить сообщению другому зарегистрированному пользователю сервиса
Найти пользователя	С помощью системы поиска найти другого зарегистрированного пользователя сервиса
Просмотреть профиль пользователя	Просмотреть информацию зарегистрированного пользователя сервиса
Просмотреть друзей	Просмотреть список зарегистрированных пользователей сервиса, с которыми текущий авторизованный пользователь состоит в «дружбе».
Добавить в друзья	Добавить зарегистрированного пользователя сервиса в «друзья».
Просмотреть личный профиль	Просмотреть профиль текущего авторизованного пользователя.
Редактировать личный профиль	Редактировать личные данные текущего авторизованного пользователя.

1.6 Требования к архитектуре

Разрабатываемая система должна быть построена по трехуровневой архитектуре: клиент – сервер приложений – сервер базы данных и представлять собой веб-приложение [4].

Клиент — это интерфейсный (обычно графический) компонент, который представляет первый уровень, собственно приложение для конечного пользователя. Первый уровень не должен иметь прямых связей с базой данных (по требованиям безопасности), быть нагруженным основной бизнес-логикой (по требованиям масштабируемости) и хранить состояние приложения (по требованиям надежности). На первый уровень может быть вынесена и обычно выносится простейшая бизнес-логика: интерфейс авторизации, алгоритмы шифрования, проверка вводимых значений на допустимость и соответствие формату, несложные операции (сортировка, группировка, подсчет значений) с данными, уже загруженными на терминал.

Сервер приложений располагается на втором уровне. На втором уровне сосредоточена большая часть бизнес-логики. Вне его остаются фрагменты, экспортируемые на терминалы, а также погруженные в третий уровень хранимые процедуры и триггеры.

Сервер базы данных обеспечивает хранение данных и выносятся на третий уровень. Обычно это стандартная реляционная или объектно-ориентированная СУБД.

Если третий уровень представляет собой базу данных вместе с хранимыми процедурами, триггерами и схемой, описывающей приложение в терминах реляционной модели, то второй уровень строится как программный интерфейс, связывающий клиентские компоненты с прикладной логикой базы данных.

В простейшей конфигурации физически сервер приложений может быть совмещён с сервером базы данных на одном компьютере, к которому по сети подключается один или несколько терминалов.

В «правильной» (с точки зрения безопасности, надёжности, масштабирования) конфигурации сервер базы данных находится на выделенном компьютере (или кластере), к которому по сети подключены один или несколько серверов приложений, к которым, в свою очередь, по сети подключаются терминалы [5].

Рисунок 1.15 – Диаграмма развертывания приложения

1.7 Требования к пользовательскому интерфейсу

При разработке проекта пользовательского интерфейса учитывается мнение художника, человеческий фактор и интуиция потенциального пользователя. Некоторые основные принципы построения оконных интерфейсов проясняются после длительной работы с ними [6]. Вот несколько самых важных принципов:

- необходимо понять, в чем состоит работа пользователя. Как правило, проектировщики интерфейсов проводят анализ заданий для понимания сути работы пользователя;

- нужно сделать так, чтобы клиент чувствовал, что он контролирует взаимодействие. Интерфейс пользователя должен обеспечивать возможность отмены действий;

- желательно предоставить клиенту несколько вариантов завершения каждой операции, связанной с интерфейсом (наподобие закрытия окна или файла), и не обращать особого внимания на его ошибки;

- внимание пользователя привлекает верхний левый угол экрана. Поэтому самую важную информацию необходимо размещать именно там;

- нужно учитывать пространственное расположение элементов. Связанные друг с другом компоненты экрана следует размещать рядом, например в одной рамке;

- необходимо обращать особое внимание на удобочитаемость и ясность элементов интерфейса. Для передачи идей и понятий необходимо использовать активный залог;

- нужно ограничивать количество используемых цветов. Из всего многообразия необходимо остановиться на нескольких. Чрезмерное множество цветов будет отвлекать пользователя от выполняемых им задач;

- как и в случае с цветами, необходимо ограничивать число шрифтов. Желательно избегать курсивов и витиеватых шрифтов;

- необходимо стараться создавать компоненты пользовательского интерфейса одинакового размера. При использовании компонентов различных размеров, разнообразии цветов и шрифтов создается мозаика, которую специалисты в области GUI называют дизайном в стиле «клоунских штанов»;

- компоненты и поля данных необходимо выравнивать по левому краю. Это уменьшает нагрузку на глаза при просмотре экрана;

- если пользователь после прочтения и обработки определенного блока информации должен щелкнуть на кнопках, то такие кнопки лучше разместить справа от блока информации или же под этим блоком и, опять-таки, справа. Это соответствует естественной тенденции (присущей нашей культуре) читать слева направо. Если одна из кнопок является кнопкой по умолчанию, то ее нужно выделить и сделать первой кнопкой в наборе.

1.7.1 Критерии качества пользовательского интерфейса

Существует четыре основных критерия качества любого интерфейса[7], а именно:

- эффективность работы пользователей;
- количество человеческих ошибок;
- скорость обучения;
- субъективное удовлетворение пользователей (подразумевается, что соответствие интерфейса задачам пользователя является неотъемлемым свойством интерфейса).

Эффективность выполнения работы. Длительность выполнения того или иного действия пользователем состоит из следующих составных частей:

- длительности восприятия исходной информации;
- длительности интеллектуальной работы (пользователь думает, что он должен сделать);
- длительности физических действий пользователя;
- длительности реакции системы.

Так как скорость мышления пользователя не зависит от программного обеспечения, то ускорить скорость выполнения работы можно только уменьшая длительность физических действий пользователя и время отклика системы. Для решения данных задач существует ряд методов.

Непосредственное манипулирование. Смысл этого метода очень прост. Пользователь не отдает команды системе, а манипулирует объектами. Очевидно, что даже такое простое действие, как стирание файла, на самом деле состоит из многих малых, уже не делимых, действий (жестов). При этом для ускорения мыслительной работы пользователя необходимо не только сокращать количество этих жестов, но и делать эти жесты более простыми.

Потеря фокуса внимания. При работе с системой на пользователя постоянно воздействуют внешние раздражители, которые заставляют пользователя время от времени отвлекаться от производимой работы. Таким

образом, необходимо максимально облегчать возвращение пользователей к работе и проектировать интерфейс так, чтобы пользователи возможно меньше о нем думали.

Быстрый или точный. Любое физическое действие, совершаемое с помощью мускулатуры, может быть или точным, или быстрым. Таким образом, чтобы физическое действие пользователя было быстрым, оно не должно быть точным. Пользователь, как правило, управляет компьютером двумя способами, а именно мышью и клавиатурой. Клавиатура не требует особой точности движений – неважно, быстро нажали клавишу или медленно, равно как сильно или слабо. Мышь, напротив, инерционна, поэтому оптимизация использования мыши в системе может существенно повысить общую скорость работы.

Длительность реакции системы. Данный показатель напрямую связан со скоростью работы пользователя. Большое время реакции системы заставляет пользователя отвлекаться от работы, а затем дополнительно тратить время на восстановление фокуса. При этом нельзя исключить случаи, когда долгое время ожидания является оправданным (передача большого количества информации, выполнение сложной операции и т.д.). В этих случаях система должна предоставить пользователю точные данные о времени завершения длительной операции. Тогда пользователь сможет переключиться на другую операцию, не отвлекаясь на периодическую проверку завершения текущей.

Человеческие ошибки. При работе с любой системой возникают ошибки, которые можно разделить на несколько групп.

Ошибки, вызванные недостаточным знанием предметной области. Ошибки данной группы могут быть исправлены обучением пользователя работе с системой. Для этой цели система должна содержать справочную информацию.

Ошибки, вызванные не считыванием показаний системы, которые одинаково часто производят как опытные, так и неопытные пользователи.

Первые не считывают показаний системы потому, что у них уже сложилось мнение о текущем состоянии, и они считают излишним его проверять, вторые – потому что они либо забывают считывать показания, либо не знают, что это нужно делать и как.

Моторные ошибки, количество которых фактически пренебрежимо мало, но не так мало, чтобы вовсе их не учитывать. Сущностью этих ошибок являются ситуации, когда пользователь знает, что он должен сделать, знает, как этого добиться, но не может выполнить действие нормально из-за того, что физические действия, которые нужно выполнить, выполнить трудно.

Важной функцией любой системы является возможность отменить выполненное действие. Данная возможность зачастую помогает пользователю не только исправить сделанную ошибку, но свободно изучать систему методом проб и ошибок, будучи при этом уверенным в возможности отменить сделанную ошибку.

1.7.2 Методика оценки эффективности пользовательского интерфейса на основе модели GOMS

Количественные методы помогают свести спорные вопросы в оценке качества интерфейса к простым вычислениям.

Одним из лучших подходов к количественному анализу моделей интерфейсов является классическая модель GOMS (the model of goals, objects, methods and selection rules) – модель целей, объектов, методов и выбор правил [9].

Эта модель основана на оценке скорости ввода. Время, требуемое для выполнения какой-то задачи системой пользователь-компьютер, является суммой всех временных интервалов, которые потребовались системе на выполнение элементарных действий, составляющих данную задачу.

В таблице 1.17 приведены стандартные средние интервалы для некоторых действий, выполняемых различными пользователями.

Для того чтобы определить, в какие моменты будут проходить ментальные операции, необходимо следовать некоторым правилам.

Правило 0. Начальная расстановка операторов М. Оператор М устанавливается перед всеми операторами К (нажатие клавиши) и Р, предназначенными для выбора команд, если Р указывает на аргументы этих команд оператор М не ставится.

Таблица 1.17 – Интервалы некоторых жестов, выполняемых пользователями персональных компьютеров

Название жеста	Условное обозначение	Среднее время, с
Нажатие клавиши	К	0,2
Указание (на какую-либо позицию на экране монитора)	Р	1,1
Перемещение руки с клавиатуры на мышь или с мыши на клавиатуру	Н	0,4
Ментальная задержка	М	1,35
Время задержки ответа компьютера	Р	-

Правило 1. Удаление ожидаемых операторов М. Если оператор, следующий за оператором М ожидаемый с точки зрения оператора, предшествующего М, то этот оператор М может быть удален и последовательность РМК превращается в РК.

Правило 2. Удаление операторов М внутри когнитивных единиц. Если строка типа «МКМКМК...» принадлежит когнитивной единице, то следует удалить все операторы М, кроме первого. Когнитивной единицей является непрерывная последовательность вводимых символов, которые могут образовывать название команды или аргумент.

Правило 3. Удаление М перед последовательными разделителями. Если оператор К означает лишний разделитель, стоящий в конце когнитивной единицы (например разделитель команды, следующий сразу за

разделителем аргумента этой команды) то следует удалить оператор М, стоящий перед ним.

Правило 4. Удаление операторов М, которые являются прерывателями команд. Если оператор К является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор М, стоящий перед ним. Но если оператор К является разделителем для строки аргументов или любой другой изменяемой строки, то М следует сохранить перед ним.

Правило 5. Удаление перекрывающихся операторов М. Любую часть операторов М, которая перекрывает оператор R, означающий задержку, связанную с ожиданием ответа компьютера, учитывать не следует.

Если имеется модель интерфейса, то с помощью GOMS можно определить время, необходимое пользователю для выполнения любой, четко сформулированной задачи, для которой данный интерфейс предусмотрен. Однако модели анализа не могут дать ответ на вопрос о том, насколько быстро должен работать интерфейс. Чтобы ответить на него, необходимо воспользоваться мерой, применяемой в теории информации.

Чтобы сделать правильную оценку времени, необходимого для выполнения задачи с помощью самого быстрого интерфейса, прежде всего, следует определить минимально количество информации, которое пользователь должен ввести, чтобы выполнить задачу. Это минимальное количество не зависит от модели интерфейса. Если методы работы, используемые в предполагаемом интерфейсе, требуют введения такого количества информации, которое превышает минимальное, это означает, что пользователь делает лишнюю работу и поэтому интерфейс следует усовершенствовать. С другой стороны, если от пользователя требуется ввести именно то количество информации, которое необходимо для

выполнения задачи, то для этой задачи интерфейс нельзя сделать более производительным путем изменения количества информации. В этом случае пути улучшения интерфейса (а также много путей для ухудшения) все же остаются, но, по крайней мере, данная цель повышения производительности будет уже достигнута.

Информационно-теоретическая производительность определяется также, как понятие производительности определяется в термодинамике – отношением мощности на выходе к мощности на входе процесса. Если в течение какого-то периода времени электрогенератор, работающий от двигателя мощностью в 1000 ватт, производить 820 ватт, то он имеет производительность $820/1000=0,82$. Производительность также часто обозначается через проценты.

Информационная производительность интерфейса E определяется как отношение минимального количества информации, необходимого для выполнения задачи, к количеству информации, которое должен ввести пользователь. Так же как и в отношении физической производительности, параметр E может изменяться в пределах от 0 до 1. Если никакой работы для выполнения задачи не требуется или работа просто не производится, то производительность составляет 1 (чтобы избежать деления на 0).

Производительность E может равняться и 0 в случаях, когда пользователь должен ввести информацию, которая бесполезна для выполнения задачи.

В параметре E учитывается только информация, необходимая для задачи и информация, вводимая пользователем.

Так для выбора одного варианта из четырех возможных достаточно двух бит информации, из восьми – трех, из 16 – 4-х и т.д.

Суммарное количество передаваемой информации определяется, как
Количество информации для каждого варианта определяется как

Если вероятности для каждого варианта или элемента не являются равными и i -й вариант имеет вероятность $P(i)$, то информация, передаваемая этим вариантом, определяется как .

Количество информации по всем вариантам является их суммой по всем вариантам выражения, которое при равновероятных вариантах сводится к первому выражению.

1.7.3 Требования к аппаратно-программному обеспечению

В таблице 1.18 указаны минимальная и рекомендуемая конфигурации аппаратного и программного обеспечения, необходимые и достаточные для корректного функционирования онлайнсервиса.

Таблица 1.138 – Минимальная и рекомендуемая конфигурации аппаратного и программного обеспечения

Параметр оборудования	Минимальная конфигурация	Рекомендуемая конфигурация
Процессор	Intel Celeron	Intel Core i7
Объем оперативной памяти	4 Гб	16 Гб
Объем постоянной памяти	500 Гб	1 Тб
Скорость сетевого интерфейса	100 Мбит/сек	1 Гбит/сек
Операционная система	Операционная система, использующая ядро Linux и основанная на Debian	

Обоснование выбора операционной системы. В современном компьютерном мире выбор операционной системы заключается в выборе из двух вариантов: MicrosoftWindowsили Linux[10]. Главным недостатком первой ОС является высокая цена на серверные варианты дистрибутивов. Соответственно платным является и все программное обеспечение, необходимое для поддержания работы сервиса. В отличие от MicrosoftWindowsоперационные системы семейства Linuxлишены этого

недостатка – ядро Linux изначально распространяется свободно. Основная часть дистрибутивов, основанных на ядре Linux, распространяются свободно. Исключение составляют особые дистрибутивы, распространяемые за деньги, но плата, как правило, взимается за улучшенную техническую поддержку, предоставляемую в случае приобретения продукта. Так как при постановке задачи на разработку интернет-сервиса была поставлена задача использовать только свободно-распространяемое ПО, то выбор, очевидно, падает на операционные системы, основанные на ядре Linux[11, 12].

В качестве конкретного дистрибутива был выбран продукт – Ubuntu. Выбор обусловлен высокой популярностью данного дистрибутива, хорошей поддержкой свободным сообществом и частым выпуском обновлений.

1.8 Выбор технологий и средств разработки

1.8.1 Выбор средств реализации серверной части

Язык программирования. В настоящее время существует большое количества языков программирования, позволяющие разрабатывать программное обеспечение для сети Интернет. Наиболее популярным на сегодняшний день является PHP (Personal Home Page Tools) - скриптовый язык программирования общего назначения[13]. Основным преимуществом данного языка является низкий порог вхождения для начала разработки. Помимо этого язык имеет и другие преимущества перед остальными, но, несмотря на это, у языка есть большой недостаток – программы на PHP интерпретируются, но не компилируются. Как следствие, программы на PHP уступают по производительности программам, написанным на других языках. Также недостатком данного языка является отсутствие полной реализации возможности объектно-ориентированного программирования [14].

В языке программирования Java отсутствуют указанные недостатки. Java—это объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems.

Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) – программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор [15].

Достоинство подобного способа выполнения программ – в полной независимости байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание [16, 17].

Для разработки программного обеспечения серверной части интернет-сервиса был выбран язык Java, как наиболее оптимальный для выполнения поставленной задачи.

Сервер приложений. В таблице 1.19 приведены существующие серверы приложений.

Таблица 1.19 – Серверы приложений

Поставщик/ Продукт	Редакция/ Версия	Дата вы- пуска	JDK	EJB	JSP	JMS	J2EE лиц.	J2EE серт.	Цена	Совместимость
Apache/ Tomcat	v4.1	Dec-01	1.3 1.4.1		1.2				Free OSS	JDK 1.1.8 JVM
BEA/ WebLogic	Express 8.1 Basic	Feb-03	1.4.1	2.0	1.2	1.0.2	+	1.3	\$495	Win 2K, Solaris 8,9 Redhat Linux 2.1
	Express 8.1 Premium								\$3,000	
	Workgroup 8.1								\$4,000	
	Server 8.1 Advantage								\$10,000	
	Server 8.1 Premium								\$17,000	
IBM/ WebSphere	Express 5.0	Oct-02	IBM 1.3.1	2.0	1.2	1.0.2 +XA	+	1.3	\$1778 /CPU	NT, Win2K, Solaris, AIX, OS/400, HP-UX, Linux (Red Hat, SuSE, TurboLinux), Linux/390, NetWare, OS/390
	Enterprise 5.0	Mar-03							\$25K /CPU	
	WebSphere v5	Nov-02							\$8K /CPU	
	WebSphere Network De- ployment v5	Nov-02							\$12K /CPU	
JBoss/ JBoss	v3.2	Apr-03	1.3.1 1.4.1	2.0	1.2	1.0.2			Free OSS (LGPL)	NT, Win2k, OS/390, Solaris, ...
Oracle/ Oracle Applica- tion Server 10g (includes Toplink)	Java Edition v9.0.4	Dec-03	1.4.1	2.0	1.2	1.0.2b	+	1.3	\$5K /CPU	Solaris, HP-UX, Redhat Linux, United Linux, NT, Win2K
	Standard Edition v9.0.4		1.4.2 (толь- ко для Linux)						\$10K /CPU	
	Enterprise Edition v9.0.4								\$20K /CPU	
Sun/ Sun Java System Application Server	Platform Edition 7.0	Oct-02	1.4	2.0	1.2	1.0.2	+	1.3	Free	Solaris 8, 9; WinXP, Win2K; Linux Red Hat 7.2, Sun Linux 5.0
	Standard Edition 7.0	Sept-03							\$2000 /CPU	
	Enterprise Edition 7.0								\$10,000 /CPU	

Так как при постановке задачи было поставлено условие использования только свободного программного обеспечения, то из приведенного списка серверов приложений остаются только OracleGlassFishв бесплатной версии, JBossApplicationsServerи ApacheTomcat. Из приведенного списка можно исключить ApacheTomcat, так как сервер приложений GlassFishвключает в себя EmbeddedTomcat, таким образом предоставляя все возможности, имеющиеся у ApacheTomcat. Кроме того GlassFishпредоставляет более широкий спектр технологий, таких как кластеризация, EJB-контейнер и JCA-контейнер, а также набор служб: JNDI, JMS, mail и другие.

Исходя из приведенных фактов, выбор сервера приложений был остановлен на OracleGlassFishв бесплатной версии [18].

Фреймворк для разработки программного обеспечения. Для разработки программного обеспечения серверной части был использован PlayFramework.

PlayFramework – это фреймворк с открытым исходным кодом для разработки веб-приложений, который основан на шаблоне проектирования «Модель-Представление-Контроллер». Главной целью данного фреймворка является повышение производительности разработчика путем снижением

количества настроек, быстрому применению изменений в коде и отображением необходимо информации об ошибках в браузере.

PlayFrameworkимеет большое количество преимуществ перед остальными фреймворками:

- фреймворк полностью удовлетворяет требованиям RESTful – сервер приложений не хранит состояние каждого из пользователей;
- минимальная настройка среды окружения для начала работы фреймворка;
- быстрое тестирование внесенных изменений – изменения применяются без перезагрузки приложения, необходимо только лишь обновить страницу браузера;
- фреймворк имеет встроенные средства для работы с модулями тестирования JUnit Selenium [19].

Выбор системы управления базами данных

Система управления базами данных (СУБД) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Существует большое количество СУБД, ориентированных на выполнения различных задач. Для хранения данных веб-приложений наиболее популярными являются MySQL, PostgreSQL, OracleDatabaseи MSSQLServer. Так как СУБД Oracle Database и MSSQLServer являются платными, то они заранее не подходят для использования в проекте, так как одной из целей проекта является использование бесплатного программного обеспечения.

В таблице 1.20 приведено сравнение двух СУБД – MySQLи PostgreSQL [20, 21, 22].

Таблица 1.20 – Сравнение СУБД MySQLи PostgreSQL

Особенность	MySQL	PostgreSQL
Производительность	Высокая скорость работы – один из приоритетов разработки продукта	Основная идея продукта – максимальное число настроек и соответствие стандарту
Многопоточность	Плохая поддержка	Хорошая поддержка
Ключи	Не поддерживает check constraint; вторичные ключи реализованы частично.	Полная реализация; поддержка ON DELETE CASCADE и ON UPDATE CASCADE.
Транзакции	Толькодля InnoDB. Поддержка SAVEPOINT, ROLLBACK TO SAVEPOINT. Уровни блокировки: table level (MyISAM).	Поддерживается; read committed и уровни изоляции. Поддержка ROLLBACK, ROLLBACK TO SAVEPOINT. Уровни блокировки: row level, table level.
Вложенные запросы	Есть, но могут работать непроизводительно	Есть
Автоматическое восстановление после сбоев	Частичное для InnoDB – нужно вручную сделать backup.	Write Ahead Logging (WAL).

Подводя итоги, можно сказать следующее: MySQL и PostgreSQL – две наиболее популярные open-source СУБД в мире. Каждая СУБД имеет свои особенности и отличия. MySQL - быстрое хранилище для простых запросов с минимальной настройкой [20].PostgreSQL - надежное хранилище для большого объема данных с возможностью расширения, репликации, полностью соответствующее современным стандартам языка SQL [22]. Так как в разрабатываемом проекте будут храниться личные данные пользователей, а в будущем планируется внедрить систему онлайн продажи билетов, то на первое место выходит гарантированная сохранность данных.

Поэтому в качестве СУБД была выбрана PostgreSQL как наиболее удовлетворяющая требованиям проекта.

1.8.2 Выбор средств реализации клиентской части

Язык программирования. В данной области существует только один язык программирования – JavaScript – объектно-ориентированный скриптовый язык программирования. Является диалектом языка ECMAScript.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам [23].

JavaScript обладает рядом свойств объектно-ориентированного языка, но реализованное в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными объектно-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам – функции как объекты первого класса, объекты как списки, анонимные функции, замыкания – что придаёт языку дополнительную гибкость [24].

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

- объекты, с возможностью интроспекции;
- функции как объекты первого класса;
- автоматическое приведение типов;
- автоматическая сборка мусора;
- анонимные функции.

В языке отсутствуют такие полезные вещи, как:

- модульная система: JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
- стандартная библиотека: в частности, отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода/вывода, базовых типов для бинарных данных;
- стандартные интерфейсы к веб-серверам и базам данных;
- система управления пакетами, которая бы отслеживала зависимости и автоматически устанавливала их [25].

Язык гипертекстовой разметки. HTML – стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа, в удобной для человека форме[26].

Язык HTML был разработан британским учёным Тимом Бернерсом-Ли приблизительно в 1989-1991 годах в стенах Европейского совета по ядерным исследованиям в Женеве (Швейцария). HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки. HTML успешно справлялся с проблемой сложности SGML путём определения небольшого набора структурных и семантических элементов – дескрипторов. Дескрипторы также часто называют «тегами». С помощью HTML можно легко создать относительно простой, но красиво оформленный документ. Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. Мультимедийные возможности были добавлены позже.

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. С течением времени, основная идея платформонезависимости языка HTML была отдана в своеобразную жертву современным потребностям в мультимедийном и графическом оформлении [27].

На данный момент актуальной версией HTML является версия 5.

1.8.3 Выбор среды разработки

Для разработки программного обеспечения сервиса была использована интегрированная среда разработки ПО IBMNetBeans, разрабатываемая компанией IBM. Данная среда разработки, появившись в 2001 году, быстро приобрела популярность, как первая Java IDE с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя им сконцентрироваться на разработке функциональности, в то время как IBMNetBeans берет на себя выполнение рутинных операций. Среди прочих возможностей, IBMNetBeans хорошо совместима со многими популярными open source инструментами разработчиков, такими как CVS, Subversion, Apache Ant, Maven и JUnit [28].

2. Реализация онлайн сервиса

2.1 Разработка базы данных онлайн сервиса

Модель сущность-связь (ER-модель) – модель данных, позволяющая описывать концептуальные схемы предметной области.

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма сущность-связь.

IDEF1X (IDEF1 Extended) –DataModeling– методология построения реляционных структур (баз данных), относится к типу методологий «Сущность-взаимосвязь» (ER–Entity-Relationship) и, как правило, используется для моделирования реляционных баз данных, имеющих отношение к рассматриваемой системе;

IDEF1X использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Концептуальной схемой называется универсальное представление структуры данных в рамках

коммерческого предприятия, независимое от конечной реализации базы данных и аппаратной платформы. Будучи статическим методом разработки, IDEF1X изначально не предназначен для динамического анализа по принципу "AS IS", тем не менее, он иногда применяется в этом качестве, как альтернатива методу IDEF1. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы (скажем с помощью метода IDEF1) и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято.

Сущности в IDEF1X и их атрибуты. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но однозначно отличаемых друг от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира. Связи между сущностями в IDEF1X представляют собой ссылки, соединения и ассоциации между сущностями.

Преимущества IDEF1X. Основным преимуществом IDEF1X, по сравнению с другими многочисленными методами разработки реляционных баз данных, такими как ER и ENALIM является жесткая и строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построенной модели, которая, несомненно, является значительным недостатком ER.

StartUML – CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать

базы данных, хранилища и витрины данных. Модели данных помогают визуализировать структуру данных, обеспечивая эффективный процесс организации, управления и администрирования таких аспектов деятельности предприятия, как уровень сложности данных, технологий баз данных и среды развертывания.

StartUML предназначен для всех компаний, разрабатывающих и использующих базы данных, для администраторов баз данных, системных аналитиков, проектировщиков баз данных, разработчиков, руководителей проектов. StartUML позволяет управлять данными в процессе корпоративных изменений, а также в условиях стремительно изменяющихся технологий.

StartUML позволяет наглядно отображать сложные структуры данных. Удобная в использовании графическая среда приложения упрощает разработку базы данных и автоматизирует множество трудоемких задач, уменьшая сроки создания высококачественных и высокопроизводительных транзакционных баз данных и хранилищ данных. Данное решение улучшает коммуникацию в вашей организации, обеспечивая совместную работу администраторов и разработчиков баз данных, многократное использование модели, а также наглядное представление комплексных активов данных в удобном для понимания и обслуживания формате [29].

2.2 Разработка логической модели базы данных

На основании разработанной концептуальной модели данных, приведенной на рисунке 1.12, была разработана логическая модель базы данных (рисунок 2.1).

Нормализация базы данных. Нормальная форма – свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение[30].

Процесс преобразования отношений базы данных (БД) к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации. Как отмечает К. Дейт, общее назначение процесса нормализации заключается в следующем:

- исключение некоторых типов избыточности;
- устранение некоторых аномалий обновления;
- разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;
- упрощение процедуры применения необходимых ограничений целостности.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

При том, что идеи нормализации весьма полезны для проектирования баз данных, они отнюдь не являются универсальным или исчерпывающим средством повышения качества проекта БД. Это связано с тем, что существует слишком большое разнообразие возможных ошибок и недостатков в структуре БД, которые нормализацией не устраняются. Несмотря на эти рассуждения, теория нормализации является очень ценным достижением реляционной теории и практики, поскольку она даёт научно строгие и обоснованные критерии качества проекта БД и формальные методы для усовершенствования этого качества. Этим теория нормализации резко выделяется на фоне чисто эмпирических подходов к проектированию, которые предлагаются в других моделях данных. Более того, можно утверждать, что во всей сфере информационных технологий практически отсутствуют методы оценки и улучшения проектных решений, сопоставимые с теорией нормализации реляционных баз данных по уровню формальной строгости [31].

Нормализацию иногда упрекают на том основании, что «это просто здравый смысл», а любой компетентный профессионал и сам «естественным образом» спроектирует полностью нормализованную БД без необходимости применять теорию зависимостей. Однако, как указывает К. Дейт, нормализация в точности и является теми принципами здравого смысла, которыми руководствуется в своём сознании зрелый проектировщик, то есть принципы нормализации – это формализованный здравый смысл. Между тем, идентифицировать и формализовать принципы здравого смысла – весьма трудная задача, и успех в её решении является существенным достижением.

Нормальные формы. В создании и развитии теории нормализации принимали участие многие учёные. Однако первые три нормальные формы и концепцию функциональной зависимости предложил Э. Кодд.

Первая нормальная форма (1NF). Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом

допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов. В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение. Что же касается различных таблиц, то они могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1НФ.

Вторая нормальная форма (2NF). Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от ее потенциального ключа.

Третья нормальная форма (3NF). Переменная отношения находится в 3NF тогда и только тогда, когда она находится во второй нормальной форме и отсутствуют транзитивные зависимости неключевых атрибутов от ключевых.

Нормальная форма Бойса – Кодда (BCNF). Переменная отношения находится в BCNF тогда и только тогда, когда каждая ее нетривиальная и неприводимая слева функциональная зависимость имеет в качестве своего детерминанта некоторый потенциальный ключ.

Четвёртая нормальная форма (4NF). Переменная отношения находится в 4NF, если она находится в BCNF и не содержит нетривиальных многозначных зависимостей.

Пятая нормальная форма (5NF). Переменная отношения находится в пятой нормальной форме (иначе – в проекционно-соединительной нормальной форме) тогда и только тогда, когда каждая нетривиальная зависимость соединения в ней определяется потенциальным ключом (ключами) этого отношения [32].

Анализ уровня нормализации. Для того чтобы произвести анализ уровня нормализации базы данных необходимо выделить все функциональные зависимости, присутствующие в БД.

Функциональные зависимости отношения «Информационное поле»:

Идентификатор поля → название, значение, идентификатор пользователя, идентификатор типа поля.

Функциональные зависимости отношения «Сообщение»:

Идентификатор сообщения → текст, дата отправки, идентификатор отправителя, идентификатор получателя, идентификатор группы файлов, идентификатора статуса.

Функциональные зависимости отношения «Статус сообщения»:

Идентификатор статуса → Название.

Функциональные зависимости отношения «Статус участия»:

Идентификатор статуса → Название.

Функциональные зависимости отношения «Тип участия»:

Идентификатор типа → Название.

Функциональные зависимости отношения «Тип поля»:

Идентификатор типа поля → Название.

Функциональные зависимости отношения «Участие в маршруте»:

Идентификатор участия → дата создания, идентификатор статуса, идентификатор типа, идентификатор пользователя, идентификатор маршрута.

Функциональные зависимости отношения «Пользователь»:

Идентификатор пользователя → фамилия, имя, отчество, дата рождения, дата регистрации, логин, пароль, идентификатор роли.

Функциональные зависимости отношения «Роль»:

Идентификатор роли → Название.

Функциональные зависимости отношения «Тип файла»:

Идентификатор типа → название, максимальный размер файла, допустимые расширения.

Функциональные зависимости отношения «Файл»:

Идентификатор файла → название, дата создания, идентификатор типа файла, путь к файлу, идентификатор автора, идентификатор сообщения, идентификатор транспортного средства, идентификатор пользователя, идентификатор типа файла.

Функциональные зависимости отношения «Маршрут»:

Идентификатор маршрута → название, описание, дата создания, дата отправления, дата прибытия, идентификатор автора.

Функциональные зависимости отношения «Комментарий»:

Идентификатор комментария → заголовок, текст, дата создания, идентификатор автора, идентификатор родительского комментария, идентификатор пользователя, идентификатор маршрута, идентификатор транспортного средства.

Функциональные зависимости отношения «Транспортное средство»:

Идентификатор транспортного средства → название, государственный номер, год выпуска, идентификатор цвета, идентификатор типа кузова, дата создания, идентификатор владельца, идентификатор модели.

Функциональные зависимости отношения «Тип кузова»:

Идентификатор типа кузова → Название.

Функциональные зависимости отношения «Цвет»:

Идентификатор цвета → Название, код.

Функциональные зависимости отношения «Модель транспортного средства»:

Идентификатор модели ТС → название, количество мест, идентификатор марки ТС.

Функциональные зависимости отношения «Марка транспортного средства»:

Идентификатор марки ТС → Название.

Функциональные зависимости отношения «Пункт маршрута»:

Идентификатор пункта маршрута → название, идентификатор типа пункта, порядковый номер, дата прибытия, идентификатор географического пункта, идентификатор маршрута.

Функциональные зависимости отношения «Тип пункта маршрута»

Идентификатор типа название

Функциональные зависимости отношения «Географический пункт»

Идентификатор географического пункта название координаты

Каждая из приведенных функциональных зависимостей отношений соответствует условиям третьей нормальной формы.

Поэтому вся модель базы данных находится в третьей нормальной форме.

В таблице 2.1

Таблица приведено описание атрибутов сущности «Пользователь»

Таблица 2.1 – Атрибуты сущности «Пользователь»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор пользователя	Целый	Положительное целое	РК

Фамилия	Символьный	А-я	
Имя	Символьный	А-я	
Отчество	Символьный	А-я	
Дата рождения	Дата		
Логин	Символьный	Любой символ, длина не менее 6 знаков	
Пароль	Символьный	Любой символ, длина не менее 6 знаков	
Идентификатор роли	Целый	Положительное целое	FK

В таблице 2.2 приведено описание атрибутов сущности «Роль».

Таблица 2.2 – Атрибуты сущности «Роль»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор роли	Целый	Положительное целое	PK
Название	Символьный	А-я	

В таблице 2.3 приведено описание атрибутов сущности «Сообщение».

Таблица 2.3 – Атрибуты сущности «Сообщение»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор сообщения	Целый	Положительное целое	PK
Текст	Символьный		
Идентификатор статуса	Целый	Положительное целое	FK
Дата отправки	Дата		
Идентификатор отправителя	Целый	Положительное целое	FK
Идентификатор получателя	Целый	Положительное целое	FK

В таблице 2.4 приведено описание атрибутов сущности «Статус сообщения».

Таблица 2.4 – Атрибуты сущности «Статус сообщения»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор статуса	Целый	Положительное целое	РК
Название	Символьный	А-я	

В таблице 2.5 приведено описание атрибутов сущности «Информационное поле».

Таблица 2.5 – Атрибуты сущности «Информационное поле»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор информационного поля	Целый	Положительное целое	РК
Название	Символьный	А-я	
Идентификатор типа поля	Целый	Положительное целое	FK
Значение	Символьный		
Идентификатор пользователя	Целый	Положительное целое	FK

В таблице 2.6 приведено описание атрибутов сущности «Тип поля».

Таблица 2.6 – Атрибуты сущности «Тип поля»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор типа	Целый	Положительное целое	РК

Название	Символьный	А-я	
----------	------------	-----	--

В таблице 2.7 приведено описание атрибутов сущности «Файл».

Таблица 2.7 – Атрибуты сущности «Файл»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор файла	Целый	Положительное целое	РК
Название	Символьный		
Дата создания	Дата		
Идентификатор типа файла	Целый	Положительное целое	FK
Путь к файлу	Символьный	a-z, /, _, -, ., ,	
Идентификатор автора	Целый	Положительное целое	FK
Идентификатор сообщения	Целый	Положительное целое	FK
Идентификатор транспортного средства	Целый	Положительное целое	FK
Идентификатор пользователя	Целый	Положительное целое	FK

В таблице 2.8 приведено описание атрибутов сущности «Тип файла».

Таблица 2.8 – Атрибуты сущности «Тип файла»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор типа	Целый	Положительное целое	РК
Название	Символьный	А-я	
Максимальный размер файла	Целый	Положительное целое	
Допустимые расширения	Символьный		

В таблице 2.9 приведено описание атрибутов сущности «Участие в маршруте».

Таблица 2.9 – Атрибуты сущности «Участие в маршруте»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор участия в маршруте	Целый	Положительное целое	PK
Дата создания	Дата		
Идентификатор статуса	Целый	Положительное целое	FK
Идентификатор типа	Целый	Положительное целое	FK
Идентификатор маршрута	Целый	Положительное целое	FK
Идентификатор пользователя	Целый	Положительное целое	FK

В таблице 2.10 приведено описание атрибутов сущности «Статус участия».

Таблица 2.10 – Атрибуты сущности «Статус участия»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор статуса	Целый	Положительное целое	PK
Название	Символьный	А-я	

В таблице 2.11 приведено описание атрибутов сущности «Тип участия».

Таблица 2.11 – Атрибуты сущности «Тип участия»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
--------------	---------------	-----------------------------	---------------

		значений	
Идентификатор типа	Целый	Положительное целое	РК
Название	Символьный	А-я	

В таблице 2.12 приведено описание атрибутов сущности «Маршрут».

Таблица 2.12 – Атрибуты сущности «Маршрут»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор маршрута	Целый	Положительное целое	РК
Название	Символьный		
Описание	Символьный		
Дата создания	Дата		
Дата отправления	Дата	Не ранее даты создания	
Дата прибытия	Дата	Не ранее даты отправления	
Идентификатор автора	Целый	Положительное целое	FK

В таблице 2.13 приведено описание атрибутов сущности «Пункт маршрута».

Таблица 2.13 – Атрибуты сущности «Пункт маршрута»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор пункта маршрута	Целый	Положительное целое	РК
Название	Символьный		
Идентификатор типа пункта	Целый	Положительное целое	FK
Порядковый номер	Целый	Положительное целое	
Дата прибытия	Дата	Не ранее даты отправления	

Идентификатор географического пункта	Целый	Положительное целое	FK
Идентификатор маршрута	Целый	Положительное целое	FK

В таблице 2.14 приведено описание атрибутов сущности «Тип пункта».

Таблица 2.14 – Атрибуты сущности «Тип пункта маршрута»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор типа	Целый	Положительное целое	PK
Название пункта	Символьный	255	

В таблице 2.15 приведено описание атрибутов сущности «Географический пункт».

Таблица 2.15 – Атрибуты сущности «Географический пункт»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор географического пункта	Целый	Положительное целое	PK
Название	Символьный		
Координаты	Символьный		

В таблице 2.16 приведено описание атрибутов сущности «Комментарий».

Таблица 2.16 – Атрибуты сущности «Комментарий»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
--------------	---------------	-----------------------------	---------------

Идентификатор комментария	Целый	Положительное целое	PK
Заголовок	Символьный		
Текст	Символьный		
Дата создания	Дата		
Идентификатор родительского комментария	Целый	Положительное целое	FK
Идентификатор автора	Целый	Положительное целое	FK
Идентификатор транспортного средства	Целый	Положительное целое	FK
Идентификатор маршрута	Целый	Положительное целое	FK
Идентификатор пользователя	Целый	Положительное целое	FK

Ошибка: источник перекрестной ссылки не найден приведено описание атрибутов сущности «Транспортное средство».

Таблица 2.17 – Атрибуты сущности «Транспортное средство»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор транспортного средства	Целый	Положительное целое	PK
Название	Символьный		
Государственный регистрационный номер	Символьный	а-я, 0-9	
Идентификатор типа кузова	Целый	Положительное целое	FK
Год выпуска	Целый	Не менее 1950	
Идентификатор цвета	Целый	Положительное целое	FK
Дата создания	Дата		
Идентификатор владельца	Целый	Положительное целое	FK
Идентификатор	Целый	Положительное	FK

модели		целое	
--------	--	-------	--

В таблице 2.18 приведено описание атрибутов сущности «Тип кузова».

Таблица 2.18 – Атрибуты сущности «Тип кузова»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор типа	Целый	Положительное целое	РК
Название	Символьный	А-я	

В таблице 2.19 приведено описание атрибутов сущности «Цвет».

Таблица 2.19 – Атрибуты сущности «Цвет»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор цвета	Целый	Положительное целое	РК
Название	Символьный	А-я	
Код	Символьный	0-9, А-F, #	

В таблице 2.20 приведено описание атрибутов сущности «Модель транспортного средства».

Таблица 2.20 – Атрибуты сущности «Модель транспортного средства»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор модели транспортного средства	Целый	Положительное целое	РК
Название	Символьный		
Количество мест	Целый	Положительное целое	
Идентификатор марки транспортного средства	Целый	Положительное целое	FK

В таблице 2.21 приведено описание атрибутов сущности «Марка транспортного средства».

Таблица 2.21 – Атрибуты сущности «Марка транспортного средства»

Имя атрибута	Формат данных	Область допустимых значений	Роль атрибута
Идентификатор марки транспортного средства	Целый	Положительное целое	РК
Название	Символьный		

2.3 Разработка физической модели базы данных

В процессе моделирования физической модели базы данных происходит переход от абстрактно-логических сущностей к физическим таблицам в базе данных. То, как логические модели будут выражаться в терминах физической базы данных, включая выбор СУБД, типов данных по умолчанию и эффективных систем индексирования, принято называть физическим уровнем модели Erwin. Перевод логической модели в физическую производится согласно основным правилам такого перевода:

- объекты становятся таблицами в физической базе данных;
- атрибуты становятся колонками в физической базе данных. Для каждой колонки нужно выбрать подходящий тип данных;
- уникальные идентификаторы становятся колонками, не допускающими значение NULL. В физической базе данных они называются первичными ключами (primary keys);
- отношения моделируются в виде внешних ключей (foreign keys).

Физическая организация данных оказывает основное влияние на эксплуатационные характеристики БД [32].

В таблице 2.22 приведено описание атрибутов таблицы «User» («Пользователь»).

Таблица 2.22 – Атрибуты таблицы «User» («Пользователь»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
id	Идентификатор пользователя	Int	11	id>0	Нет	PK
surname	Фамилия	Varchar	512		Да	
name	Имя	Varchar	128		Нет	
father_name	Отчество	Varchar	128		Да	
birthday	Дата рождения	Datetime		01.01.1950	Да	
registration_dt	Дата регистрации	Datetime		01.01.2010	Нет	
login	Логин	Varchar	64	a-z, -, @, #, &, (,), [,], ?	Нет	
password	Пароль	Varchar	64	a-z, -, @, #, &, (,), [,], ?	Нет	
role_id	Идентификатор роли	Int	11	Role_id>0	Нет	FK

В таблице 2.23 приведено описание атрибутов таблицы «Role» («Роль»).

Таблица 2.23 – Атрибуты таблицы «Role» («Роль»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
Id	Идентификатор роли	Int	11	id>0	Нет	PK
title	Название	Varchar	512		Да	

Таблица 2.24 приведено описание атрибутов таблицы «Info_field» («Информационное поле»).

Таблица 2.24 – Атрибуты таблицы атрибутов таблицы «Info_field» («Информационное поле»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
Id	Идентификатор поля	Int	11	id>0	Нет	PK
title	Название	Varchar	512		Да	
value	Значение	Varchar	512		Да	
user_id	Идентификатор пользователя	Int	11	User_id>0	Нет	FK
Type_id	Идентификатор типа поля	Int	11	Type_id>0	Нет	FK

В таблице 2.25 приведено описание атрибутов таблицы «Info_field_type» («Тип информационного поля»).

Таблица 2.25 – Атрибуты таблицы атрибутов таблицы «Info_field_type» («Тип информационного поля»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
id	Идентификатор типа поля	Int	11	id>0	Нет	PK
title	Название	Varchar	512		Да	

В таблице 2.26 приведено описание атрибутов таблицы «Message» («Сообщение»).

Таблица 2.26 – Атрибуты таблицы атрибутов таблицы «Message» («Сообщение»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
-------------	-------------------------	------------	-------------	-----------------------------	---------------------------	------

Id	Идентификатор сообщения	Int	11	id>0	Нет	PK
Send_date	Дата отправления	DATETIME	Нет			
Sender_user_id	Идентификатор пользователя-отправителя	Int	11	Send_user_id>0	Нет	FK
Text	Текст сообщения	Varchar	2048		Нет	
Receiver_user_id	Идентификатор пользователя-получателя	Int	11	Receiver_user_id>0	Нет	FK
Status_id	Идентификатор статуса	Int	11	Status_id>0	Нет	

В таблице 2.27 приведено описание атрибутов таблицы «Message_status» («Статус сообщения»).

Таблица 2.27 – Атрибуты таблицы атрибутов таблицы «Message_status» («Статус сообщения»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
Id	Идентификатор статуса сообщения	Int	11	id>0	Нет	PK
Title	Название статуса	Varchar	512		Нет	

В таблице 2.28 приведено описание атрибутов таблицы «File» («Файл»).

Таблица 2.28 – Атрибуты таблицы атрибутов таблицы «File» («Файл»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
id	Идентификатор файла	Int	11	id>0	Нет	PK
Title	Название файла	Varchar	512		Нет	

Create_dt	Дата создания	DATETIME			Нет	
author_id	Идентификатор пользователя-автора	Int	11	author_id >0	Нет	FK
message_id	Идентификатор сообщения	Int	11	message_id >0	Да	FK
vehicle_id	Идентификатор транспортного средства	Int	11	vehicle_id >0	Да	FK
user_id	Идентификатор пользователя	Int	11	user_id >0	Да	FK
type_id	Идентификатор типа файла	Int	11	type_id >0	Да	FK

В таблице 2.29 приведено описание атрибутов таблицы «File_type» («Тип файла»).

Таблица 2.29 – Атрибуты таблицы атрибутов таблицы «File_type» («Тип файла»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
id	Идентификатор типа файла	Int	11	id>0	Нет	PK
Title	Название типа файла	Varchar	512		Нет	

Таблица 2.30 – Атрибуты таблицы атрибутов таблицы «Vehicle» («Транспортное средство»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
Id	Идентификатор транспортного средства	Int	11	id>0	Нет	PK
Title	Название транспортного средства	Varchar	512	Нет	Нет	
Reg_number	Государственный регистрационный номер	Varchar	32	Нет	Нет	
Year	Год выпуска	Int	11	Year>1940	Да	

Create_dt	Дата создания	DATETIME			Нет	
owner_id	Идентификатор владельца ТС	Int	11	Owner__id >0	Нет	FK
model_vehicle_id	Идентификатор модели транспортного средства	Int	11	model_vehicle_id >0		
color_id	Идентификатор цвета	Int	11	color_id >0		
type_id	Идентификатор типа ТС	Int	11	type_id >0	Нет	FK

В таблице 2.31 приведено описание атрибутов таблицы «Vehicle_type» («Тип транспортного средства»).

Таблица 2.31 – Атрибуты таблицы атрибутов таблицы «Vehicle_type» («Тип транспортного средства»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
id	Идентификатор типа ТС	Int	11	id>0	Нет	PK
Title	Название типа ТС	Varchar	512		Нет	

В таблице 2.32 приведено описание атрибутов таблицы «Color» («Цвет»).

Таблица 2.31 – Атрибуты цвета автомобиля

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
Id	Идентификатор цвета	Int	11	id>0	Нет	PK
Title	Название цвета	Varchar	512		Нет	
Rgb_code	Код цвета	Varchar	16		Да	

В таблице 2.32 приведено описание атрибутов таблицы «Model_vehicle» («Модель транспортного средства»).

Таблица 2.32 – Атрибуты таблицы атрибутов таблицы «Model_vehicle» («Модель транспортного средства»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
id	Идентификатор модели	Int	11	id>0	Нет	
Title	Название модели	Varchar	512		Нет	
Count_places	Количество мест	Int	11		Нет	
Mark_vehicle_id	Идентификатор марки ТС	Int	11	Mark_vehicle_id>0	Нет	FK

В таблице 2.33 приведено описание атрибутов таблицы «Mark_vehicle» («Марка ТС»).

Таблица 2.33 – Атрибуты таблицы атрибутов таблицы «Mark_vehicle» («Марка ТС»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
Id	Идентификатор марки ТС	Int	11	id>0	Нет	PK
Title	Название марки ТС	Varchar	512		Нет	

Ошибка: источник перекрестной ссылки не найден приведено описание атрибутов таблицы «Comment» («Комментарий»).

Таблица 2.34 – Атрибуты таблицы атрибутов таблицы «Comment» («Комментарий»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения	Роль
-------------	-------------------------	------------	-------------	-----------------------------	----------------------	------

					Null	
Id	Идентификатор комментария	Int	11	id>0	Нет	PK
Text	Текст сообщения	Varchar	512		Нет	
Create_dt	Дата создания	DATETIME			Нет	
Parent_comment_id	Идентификатор родительского комментария	Int	11	Parent_comment_id >0	Да	FK
Author_user_id	Идентификатор автора	Int	11	Author_user_id >0	Нет	FK
vehicle_id	Идентификатор транспортного средства	Int	11	vehicle_id >0	Да	FK
user_id	Идентификатор пользователя	Int	11	user_id >0	Да	FK
route_id	Идентификатор маршрута	Int	11	route_id >0	Да	FK

В таблице 2.35 приведено описание атрибутов таблицы «Route_point» («Пункт маршрута»).

Таблица 2.35 – Атрибуты таблицы атрибутов таблицы «Route_point» («Пункт маршрута»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
Id	Идентификатор пункта маршрута	Int	11	id>0	Нет	PK
Title	Название пункта	Varchar	512		Нет	
Position	Порядковый номер	Int	11	Position>0	Нет	
Arrival_dt	Дата прибытия	Datetime			Да	
Geopoint_id	Идентификатор географического пункта	Int	11	Geopoint_id >0	Нет	FK
route_id	Идентификатор маршрута	Int	11	route_id >0	Нет	FK
type_id	Идентификатор типа пункта	Int	11	type_id >0	Нет	FK

В таблице 2.36 приведено описание атрибутов таблицы «Route_point_type» («Тип пункта маршрута»).

Таблица 2.36 – Атрибуты таблицы атрибутов таблицы «Route_point_type» («Тип пункта маршрута»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
Id	Идентификатор типа пункта маршрута	Int	11	id>0	Нет	PK
Title	Название типа пункта маршрута	Varchar	512		Нет	

В таблице 2.37 приведено описание атрибутов таблицы «Route» («Маршрут»).

Таблица 2.37 – Атрибуты таблицы атрибутов таблицы «Route» («Маршрут»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль
id	Идентификатор маршрута	Int	11	id>0	Нет	PK
Title	Название маршрута	Varchar	512		Нет	
Description	Описание маршрута	Varchar	2048		Да	
Create_dt	Дата создания	Datetime			нет	
Departure_dt	Дата отправления	Datetime			Нет	
Arrival_dt	Дата прибытия	Datetime				
uthor_user_id	Идентификатор автора	Int	1	Author_user_id >0	Нет	К
Vehicle_id	Идентификатор транспортного средства	Int	11	Vehicle_id >0	Нет	FK

В таблице 2.38 приведено описание атрибутов таблицы «Participate_route» («Участник маршрута»).

Таблица 2.38 – Атрибуты таблицы атрибутов таблицы «Participate_route» («Участник маршрута»)

Имя	Содержательное	Тип	Разм	Область	Возможно	Рол
-----	----------------	-----	------	---------	----------	-----

столбца	описание	данных	ерно сть	допустимых значений	сть значения Null	ь
Id	Идентификатор участника маршрута	Int	11	id>0	Нет	PK
Create_dt	Дата создания	Datetime			нет	
User_id	Идентификатор пользователя	Int	11	User_id>0	Нет	FK
Route_id	Идентификатор маршрута	Int	11	Route_id>0	Нет	FK
Status_id	Идентификатор статуса участника	Int	11	Status_id	Нет	FK
Type_id	Идентификатор типа участника	Int	11		Нет	

В таблице 2.39 приведено описание атрибутов таблицы «Participate_status» («Статус участника»).

Таблица 2.39 – Атрибуты таблицы атрибутов таблицы «Participate_status» («Статус участника»)

Имя столбца	Содержательное описание	Тип данных	Разм ерно сть	Область допустимых значений	Возможно сть значения Null	Рол ь
Id	Идентификатор статуса участника	Int	11	id>0	Нет	PK
Title	Название статуса участника	Varchar	512		Нет	

В таблице 2.40 приведено описание атрибутов таблицы «Participate_type» («Тип участника»).

Таблица 2.40 – Атрибуты таблицы атрибутов таблицы «Participate_type» («Тип участника»)

Имя столбца	Содержательное описание	Тип данных	Разм ерно сть	Область допустимых значений	Возможно сть значения Null	Рол ь
id	Идентификатор типа участника	Int	11	id>0	Нет	PK
Title	Название типа участника	Varchar	512		Нет	

2.4 Разработка программного обеспечения

2.4.1 Разработка и анализ пользовательского интерфейса

Пользовательский интерфейс сервиса должен позволять пользователю легко и быстро перемещаться по страницам сервиса. На рисунке 2.3 представлена карта навигации по страницам сервиса.

Каждая страница сервиса включает в себя панель навигации, которая предоставляет ссылки для быстрого перехода на главную страницу и страницы первого уровня: «Личный профиль», «Мои друзья», «Мои сообщения» и «Мои маршруты». Также панель навигации содержит поле быстрого поиска, которое позволяет осуществлять быстрый поиск по страницам сервиса.

Таблица 2.41 приведено описание страниц сервиса.

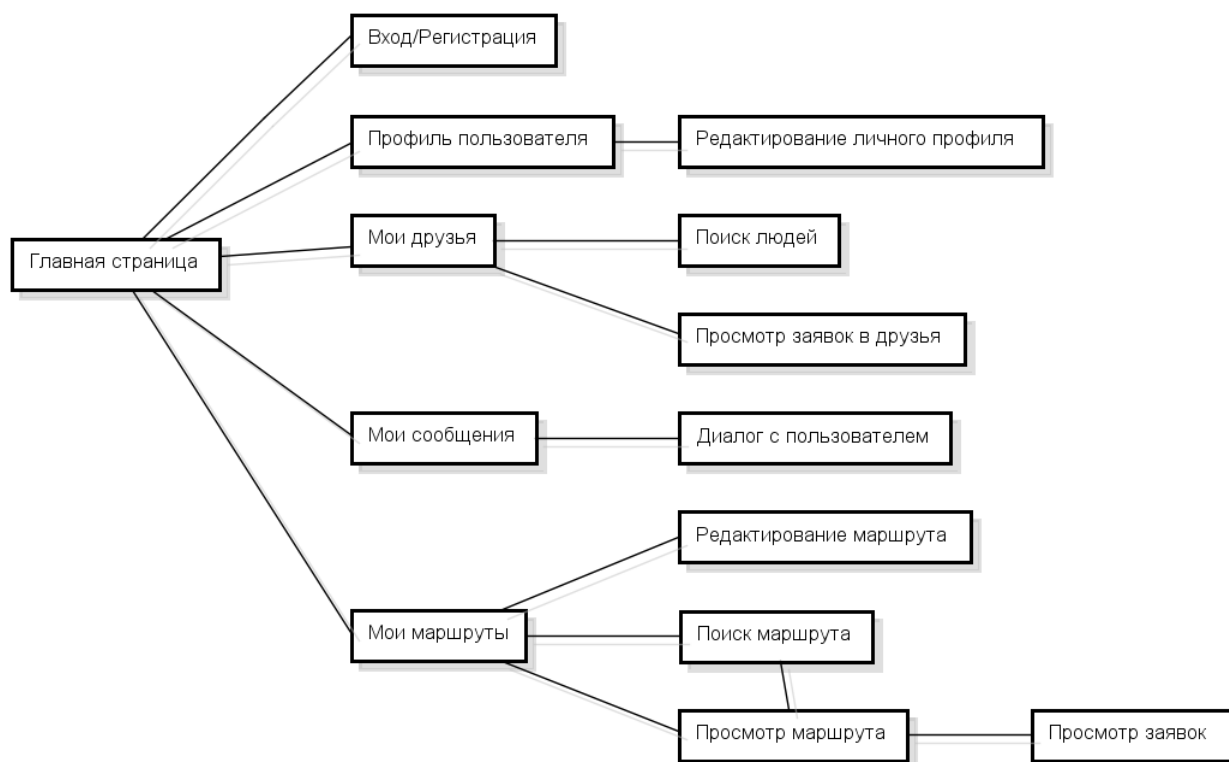


Рисунок 2.3 – Карта навигации по страницам сервиса

Таблица 2.41 – Описание страниц сервиса

Название страницы	Описание
1	2
Главная страница	Корневая страница сервиса. Основное назначение страницы – привлечение новых пользователей, которые посетили ресурс первый раз. На странице приводится описание основных принципов работы сервиса, а также преимуществ, которые получит пользователь, используя сервис.
Вход/регистрация	Страница, содержащая средства для авторизации в системе уже зарегистрированного пользователя и для регистрации нового пользователя.
Профиль пользователя	Страница просмотра данных выбранного зарегистрированного пользователя сервиса. Также служит для просмотра и редактирования собственных данных текущего авторизованного пользователя.

Мои друзья	Страница просмотра списка пользователей, состоящих в отношении «друг» с текущим авторизованным пользователем.
Поиск людей	Страница поиска зарегистрированных пользователей сервиса по различным личным данным.
Просмотр заявок в друзья	Страница просмотра поступивших заявок на добавление в друзья, отправленных другими зарегистрированными пользователями сервиса.
Мои сообщения	Страница просмотра списка диалогов, начатых текущим авторизованным пользователем с другими зарегистрированными пользователями сервиса.
Диалог с пользователем	Страница переписки текущего авторизованного пользователя с другим зарегистрированным пользователем сервиса.
Мои маршруты	Страница просмотра списка собственных маршрутов текущего авторизованного пользователя сервиса, а также маршрутов, в которых он участвует.
Редактирование маршрута	Страница редактирования выбранного собственного маршрута текущего авторизованного пользователя системы.
Поиск маршрута	Страница поиска маршрута, содержащая панель расширенного поиска с использованием географической карты одного из картографических сервисов.
Просмотр маршрута	Страница просмотра выбранного маршрута.
Просмотр заявок на поездку	Страница просмотра поступивших заявок на участие в выбранном маршруте текущего авторизованного пользователя.

Создание интерфейса начинается с построения макетов каждой страницы. Для выполнения данной задачи был использован бесплатный интернет-сервис pinjamock.com. На рисунках 2.4 – 2.9 представлены макеты страниц сервиса.

На рисунке 2.4 представлен макет главной страницы сервиса. Главная цель данной страницы – привлечение новых пользователей, которые впервые посетили данный сервис. На главной странице приводится короткий

видеоролик и текст, описывающий основные преимущества, которые получает пользователь, пользуясь данным сервисом.

Каждая страница сервиса включает в себя единый блок навигации, расположенный в верхней части страницы и предназначенный для навигации по сервису. Данный блок включает в себя ссылку на главную страницу сервиса, а также на наиболее часто используемые страницы: «Мои маршруты», «Мои друзья» и «Мои сообщения».

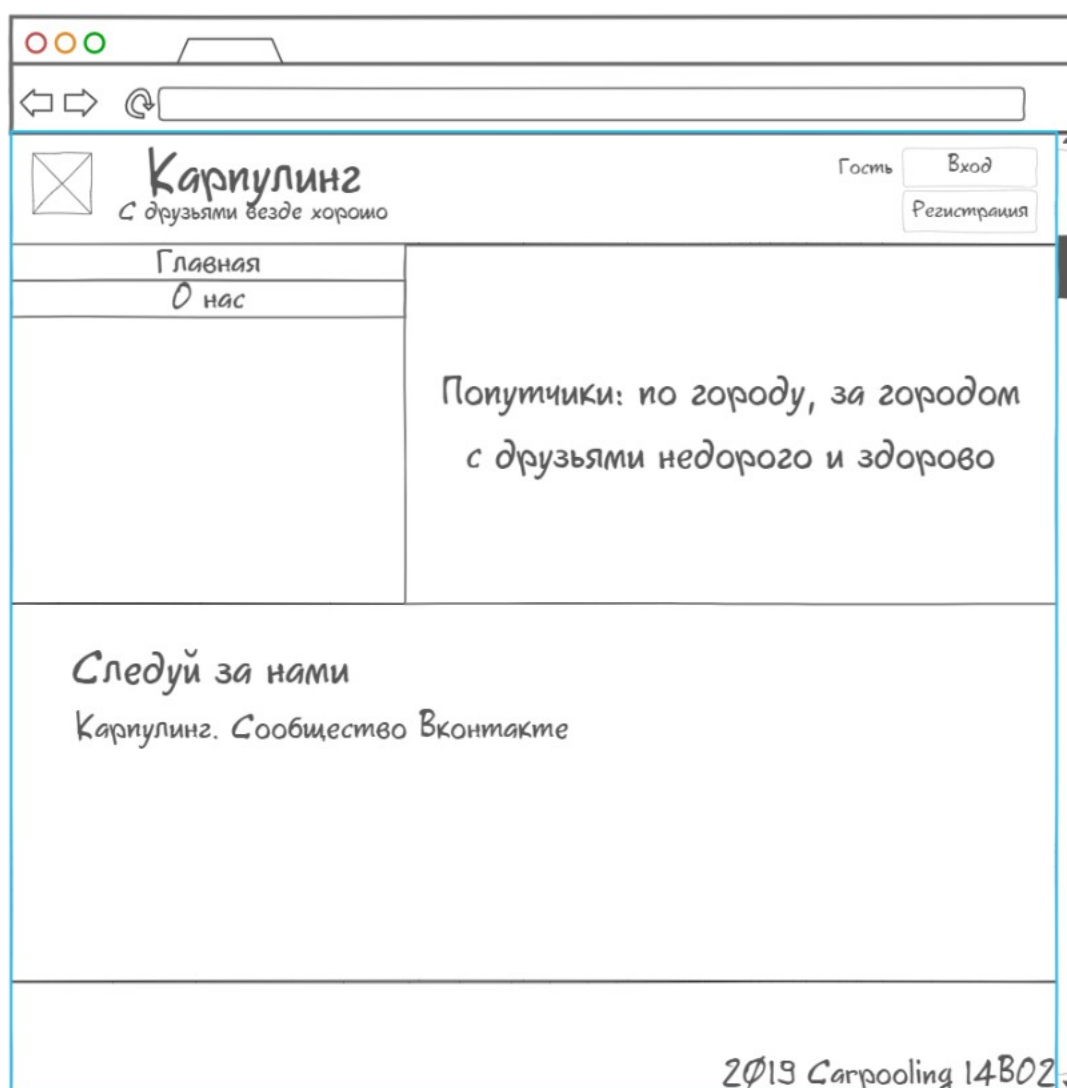


Рисунок 2.4 – Макет главной страницы сервиса

Помимо ссылок, блок навигации содержит панель поиска и кнопку авторизации. Назначение кнопки авторизации различается для

авторизованного пользователя и гостя. Для авторизованного пользователя при нажатии на кнопку авторизации открывается меню, позволяющее перейти на страницу профиля пользователя или выйти из системы. Для незарегистрированного пользователя при нажатии на кнопку открывается страница, содержащая средства авторизации и регистрации.

На рисунке 2.5 приведен макет страницы просмотра списка маршрутов. В списке маршрутов выводится основная информация, необходимая для выбора маршрута. При клике по строке таблицы происходит переход на страницу просмотра выбранного маршрута. Также данная страница содержит кнопку «Создать маршрут», при нажатии на которую происходит переход на страницу редактирования нового маршрута. При создании маршрута происходит автоматическое сохранение нового маршрута с параметрами по умолчанию. После этого происходит лишь редактирование созданного маршрута.

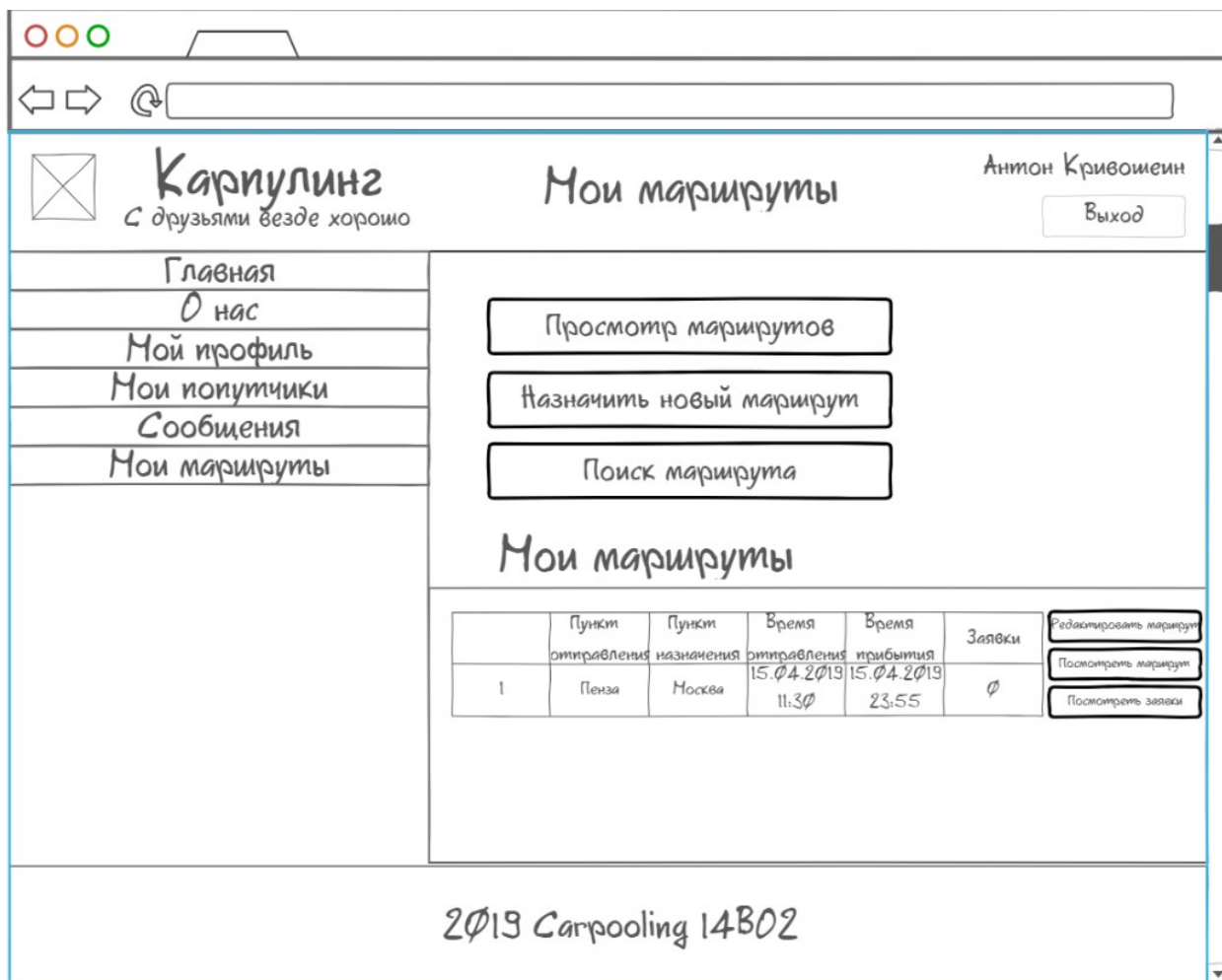


Рисунок 2.5 – Макет страницы просмотра списка маршрутов

На рисунке 2.7 приведен макет страницы поиска маршрута. Данная страница содержит инструменты, позволяющие из общей массы всех маршрутов, зарегистрированных в системе, произвести поиск необходимых, указав критерии поиска.

Основными критериями для выбора маршрута являются пункты отправления и назначения, а также время отправления и прибытия.

Карпулинг

С друзьями везде хорошо

Поиск маршрута

Антон Кривошеин

Выход

Главная

О нас

Мой профиль

Мои попутчики

Сообщения

Мои маршруты

Пункт отправления

Пункт назначения

Время отправления

Время прибытия

Пенза

11.04.2019

Искать

Результаты поиска

Найден 1 маршрут

Пункт отправления

Пункт назначения

Время отправления

Время прибытия

Автор

Время создания

Автомобиль

Заявок

Пенза

Н. Ломов

11.04.2019 9:00

11.04.2019 12:30

Александр Пушкин

02.04.2019 15:00

Renault Logan

0

Подать заявку

2019 Carpooling 14802

Рисунок 2.6 – Макет страницы поиска маршрута

При этом на странице расположена интерактивная карта, которая позволяет указывать пункт отправления или назначения непосредственно на карте. Список найденных маршрутов обновляется динамически при изменении какого-либо критерия поиска. При этом чем больше будет указано критериев, тем более точным будет поиск. По аналогии со странице просмотра списка маршрутов, при клике по строке маршрута происходит переход на страницу просмотра выбранного маршрута.

Для оценки эффективности интерфейса приведем расчеты для нескольких основных сценариев пользовательских действий, которые наиболее часто будут применяться на практике.

Сценарий «Создать маршрут». Данный сценарий выполняет заполнение основных полей данных о маршруте и его сохранение. Сценарий состоит из следующих примитивных пользовательских жестов:

- переместить курсор в поле названия маршрута (P);
- ввести название маршрута «Мой первый маршрут» ($H + 20 * K$);
- переместить курсор в поле ввода даты отправления ($H + P$);
- ввести дату отправления «11:30 11.03.2019» ($H + 16 * K$);
- переместить курсор в поле ввода даты прибытия ($H + P$);
- ввести дату прибытия «12:30 15.03.2019» ($H + 16 * K$);
- нажать на кнопку «Добавить пункт» ($H + K$);
- переместить курсор в поле ввода названия пункта ($H + P$);
- ввести название пункта «Пенза» ($H + 5 * K$);
- переместить курсор в поле ввода даты прибытия в пункт ($H + P$);
- ввести дату прибытия в пункт «12:30 15.03.2019» ($H + 16 * K$);
- переместить курсор в поле ввода количества свободных мест ($H + P$);
- ввести количество свободных мест ($H + K$);
- переместить курсор в поле ввода выбора автомобиля ($H + P$);
- выбрать автомобиль из списка ($H + K + P + K$);
- нажать кнопку сохранения маршрута ($H + K$).

Исходя из приведенного списка получим следующую последовательность условных символов:

$$\begin{aligned}
 &P + H + 20 * K + H + P + H + 16 * K + H + K + H + P + H + 5 * K + H + P + H \\
 &\quad + \\
 &\quad + 16 * K + H + P + H + K + P + K + H + K.
 \end{aligned}$$

Применив правило 0, получим последовательность:

$$\begin{aligned}
 &M + P + H + 20 * (M + K) + H + M + P + H + 16 * (M + K) + H + M + K + H \\
 &\quad + \\
 &\quad + M + P + H + 5 * (M + K) + H + M + P + H + 16 * (M + K) + H + M + P + \\
 &\quad + H + M + K + M + P + M + K + H + M + K.
 \end{aligned}$$

Применив правило 1, получим последовательность:

$$\begin{aligned}
 &M + P + H + 20 * (M + K) + H + M + P + H + 16 * (M + K) + H + M + K + H \\
 &\quad + \\
 &\quad + M + P + H + 5 * (M + K) + H + M + P + H + 16 * (M + K) + H + M + P + \\
 &\quad + H + M + K + M + P + K + H + M + K.
 \end{aligned}$$

Применив правило 2, получим последовательность:

$$\begin{aligned}
 &M + P + H + M + 20 * K + H + M + P + H + M + 16 * K + H + M + K + H + \\
 &\quad + M + P + H + M + 5 * K + H + M + P + H + M + 16 * K + H + M + P + \\
 &\quad + H + M + K + M + P + K + H + M + K.
 \end{aligned}$$

После применения правил 3, 4 и 5 последовательность останется неизменной. Таким образом, время выполнения указанного сценария можно рассчитать по полученной последовательности:

$$M + P + H + M + 20 * K + H + M + P + H + M + 16 * K + H + M + K + H +$$

$$\begin{aligned}
& + M + P + H + M + 5 * K + H + M + P + H + M + 16 * K + H + M + P + \\
& + H + M + K + M + P + K + H + M + K = \\
& = 13 * M + 6 * P + 11 * H + 61 * K = \\
& = 13 * 1,35 + 6 * 1,1 + 11 * 0,4 + 61 * 0,2 = 15,012 \text{ с.}
\end{aligned}$$

Из приведенных расчетов следует, что на выполнение сценария создания маршрута пользователь потратит, в среднем, 15 секунд. Данный результат является допустимым, поэтому макет страницы редактирования маршрута остается в виде, приведенном на рисунке .

На рисунке Ошибка: источник перёкрестной ссылки не найден представлен макет страницы просмотра маршрута. Данная страница предназначена для отображения подробной информации о маршруте в текстовом и графическом виде. Страницу можно условно разделить на две части. В одной части информация представлена в текстовом виде (название маршрута, время отправления и прибытия, пункты маршрута, периодичность поездок и количество мест). В другой части информация отображается графически на интерактивной географической карте, которая предоставляется картографическим сервисом. Подобное представление информации в двух видах позволяет пользователю выбрать тот способ представления информации, который ему наиболее удобен.

На рисунке 2.9 представлен макет страницы просмотра списка друзей. Информация на странице представлена в виде списка, где каждый элемент содержит информацию о зарегистрированном пользователе сервиса, который состоит в дружеских отношениях с текущим авторизованным пользователем. Для каждого пользователя отображается его фотография, фамилия и имя, текущий статус присутствия в сети, а также быстрые ссылки для доступа к маршрутам пользователя, его друзьям и автомобилям. Также для каждого пользователя в списке доступны кнопки быстрых действий: написать сообщения и удалить из друзей.

На рисунке 2.10 представлен макет страницы просмотра предложений дружбы. Данные заявки поступают от других зарегистрированных пользователей сервиса. Текущий авторизованный пользователь может либо принять заявку, либо отклонить, для чего у каждой заявки в списке расположены соответствующие кнопки.

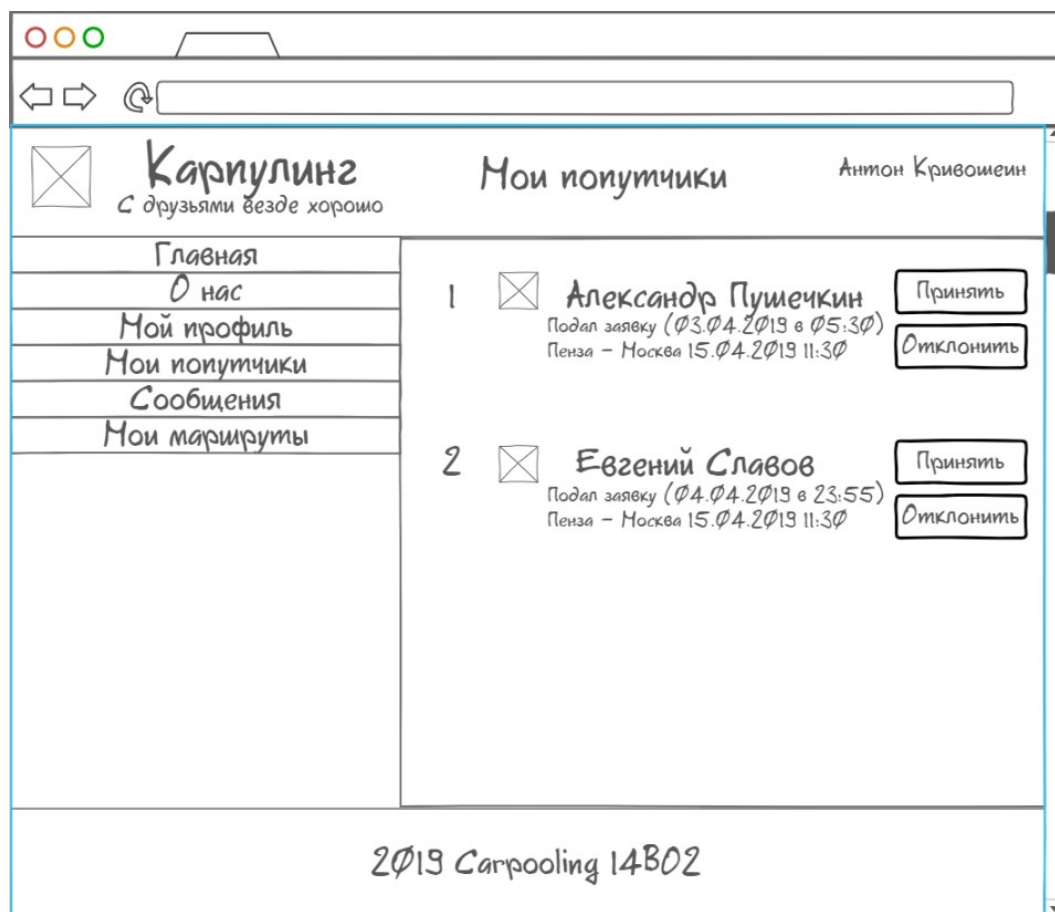


Рисунок 2.8 – Макет страницы просмотра списка попутчиков

На рисунке 2.11 представлен макет страницы поиска людей. Данная функция полезна для пользователей в случае, когда необходимо найти других пользователей, зарегистрированных на сервисе. Поиск может осуществляться по нескольким ключевым словам. Также возможно уточнение параметров поиска.

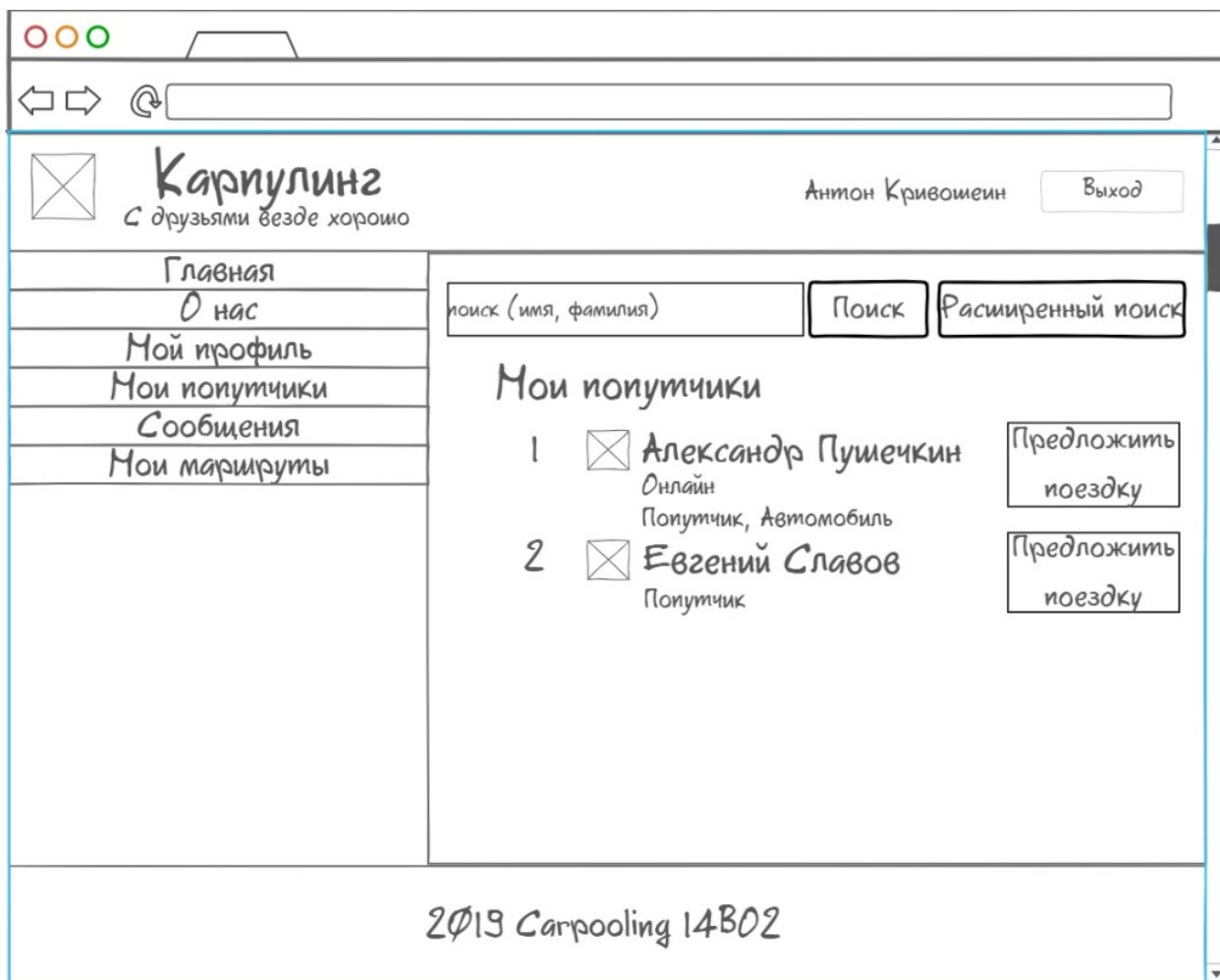


Рисунок 2.9 – Макет страницы поиска людей

На рисунке 2.12 представлен макет страницы просмотра списка диалогов. Диалог – это переписка между текущим авторизованным и другим зарегистрированным пользователями системы. Для каждого диалога доступна функция удаления всего диалога.

Новые сообщения добавляются в нижнюю часть диалога, сдвигая более старые вверх. При этом пользователю доступна полоса прокрутки, которая позволяет обратиться к любому сообщению из переписки. Для снижения объема передаваемых данных между браузером и сервером подгрузка истории переписки осуществляется фиксированным количеством сообщений.

Таким образом, по мере прокручивания списка сообщений, пользователь может получить доступ ко всем сообщениям.

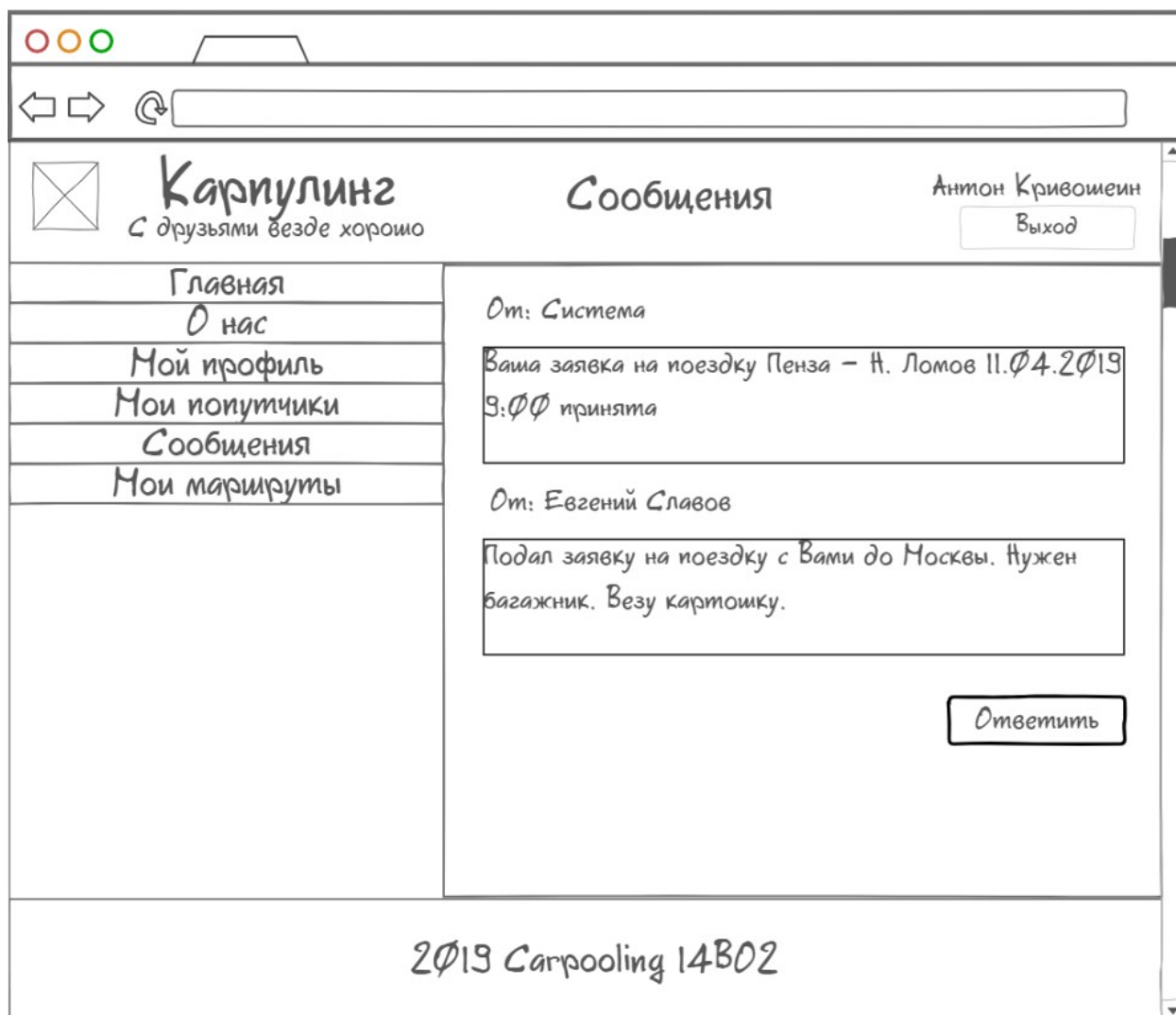


Рисунок 2.10 – Макет страницы просмотра списка диалогов

На рисунке 2.11 представлен макет страницы просмотра профиля пользователя. На данной странице присутствует фотография пользователя, показатель его рейтинга, ссылка быстрого перехода к диалогу с пользователем, основная информация о пользователе и его автомобилях, а также географическая карта, на которой указывается место, откуда пользователь последний раз заходил в систему или где находится в данный момент.



Рисунок 2.11 – Макет страницы просмотра профиля пользователя

2.4.2 Разработка программного обеспечения клиентской части

Так как клиентская часть будет выполняться в браузере на компьютере пользователя, необходимо обеспечить корректную работу приложения во всех популярных версиях браузеров. Для решения данной задачи наиболее оптимально использовать специальный фреймворк, который обеспечивает совместимость с большим количеством браузеров и может гарантировать работу приложения в каждом из них. Поэтому разработка программного обеспечения клиентской части велась с использованием CSS-фреймворка Bootstrap.

Twitter Bootstrap – это набор инструментов от Twitter (относится к классу инструментов: CSS-фреймворк), созданный для облегчения разработки Web приложений и сайтов. Он включает CSS и HTML для типографики, форм, кнопок, таблиц, сеток, навигации, и много другого.

Основные инструменты Bootstrap:

- сетки – заранее заданные размеры колонок, которые можно сразу же использовать;
- шаблоны – фиксированный или резиновый шаблон документа;
- типографика – описания шрифтов, определение некоторых классов для шрифтов таких как код, цитаты и т.п.;
- медиа – представляет некоторое управление изображениями и видео;
- таблицы – средства оформления таблиц, вплоть до добавления функциональности сортировки;
- формы – классы для оформления не только форм, но и некоторых событий, происходящих с ними;

- навигация — классы оформления для вкладок, переключателей страниц, меню и панели управления;
- алерты — оформление диалоговых окон, подсказок и всплывающих окон.

На основе фреймворка были спроектированы страницы сервиса, представленные на рисунках.

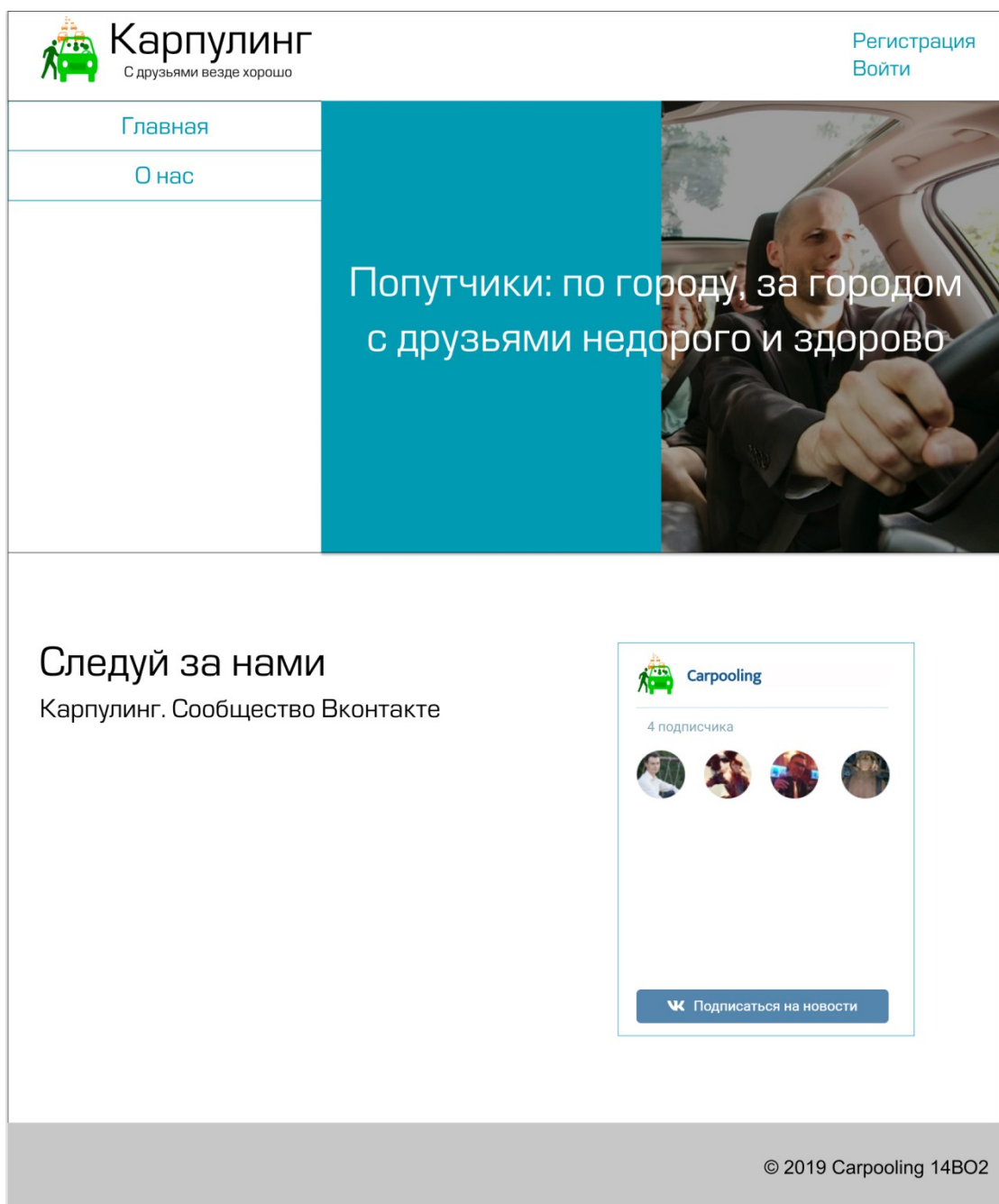


Рисунок 2.12 – Главная страница сервиса

На рисунке 2.12 приведено изображение главной страницы сервиса. В соответствии с макетом на главной странице отображается короткое описание сервиса, а также примеры, в которых использование данного сервиса способно принести пользователю выгоду. Главная страница носит рекламный характер и служит для привлечения новых пользователей.

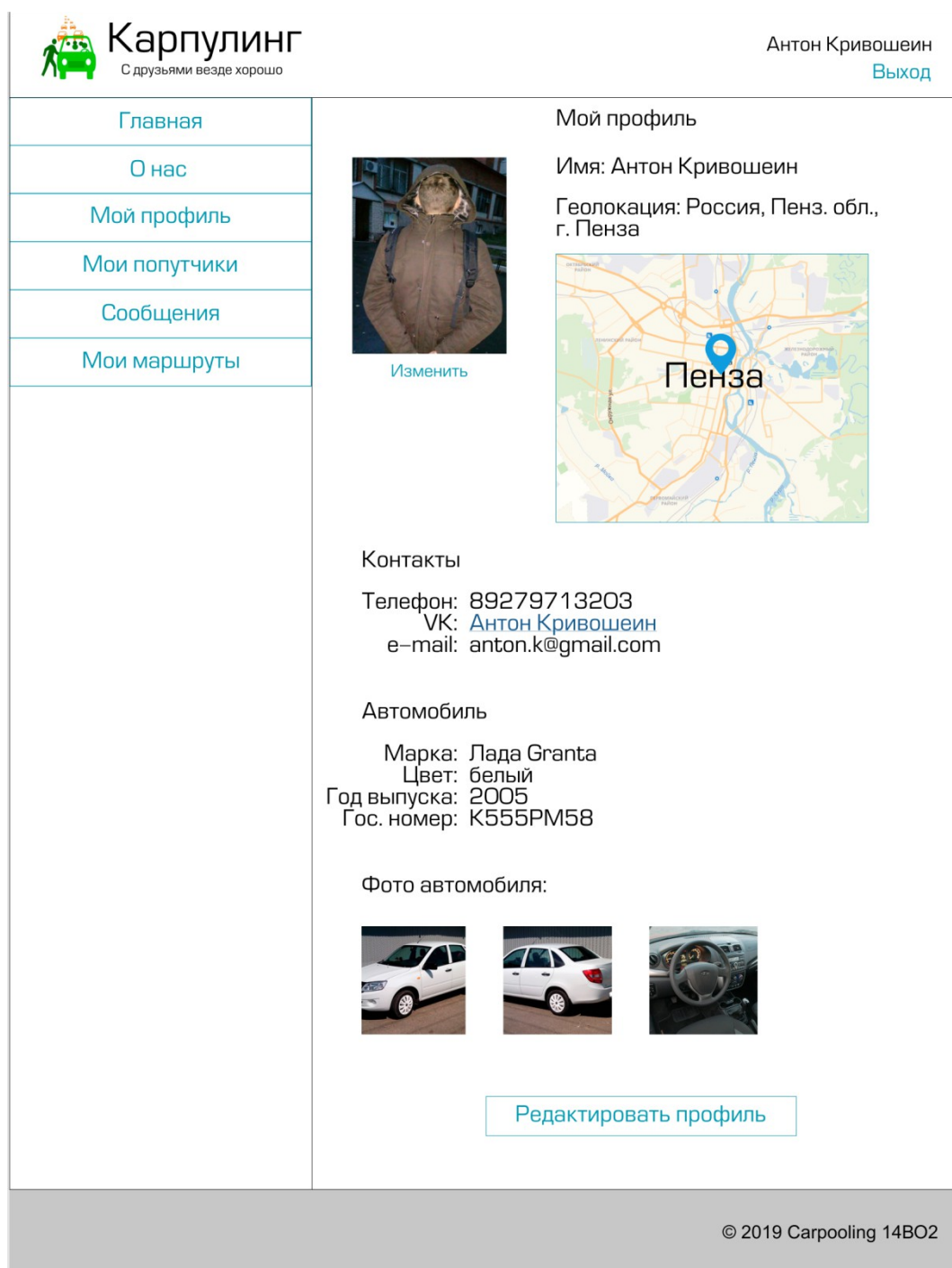


Рисунок 2.13 – Страница просмотра личного профиля пользователя

На рисунке 2.13 приведено изображение страницы просмотра личного профиля пользователя. На странице, в соответствии с разработанным макетом, отображается фотография пользователя, личная информация, а также карта, на которой указано место последнего входа пользователя в систему. В случае, если пользователь выступает в роли водителя, на странице отображаются сведения об его автомобилях.

На рисунке 2.14 приведено изображение страницы просмотра списка друзей. Для каждого пользователя в списке отображается фотография, фамилия и имя, контактные данные, а также кнопки перехода к диалогу с пользователем и удаления выбранного пользователя из друзей.

На рисунке 2.15 приведено изображение страницы поиска людей. Данная страница содержит панель поиска, содержащую элементы для ввода параметров поиска, и список пользователей, найденных по указанным параметрам.

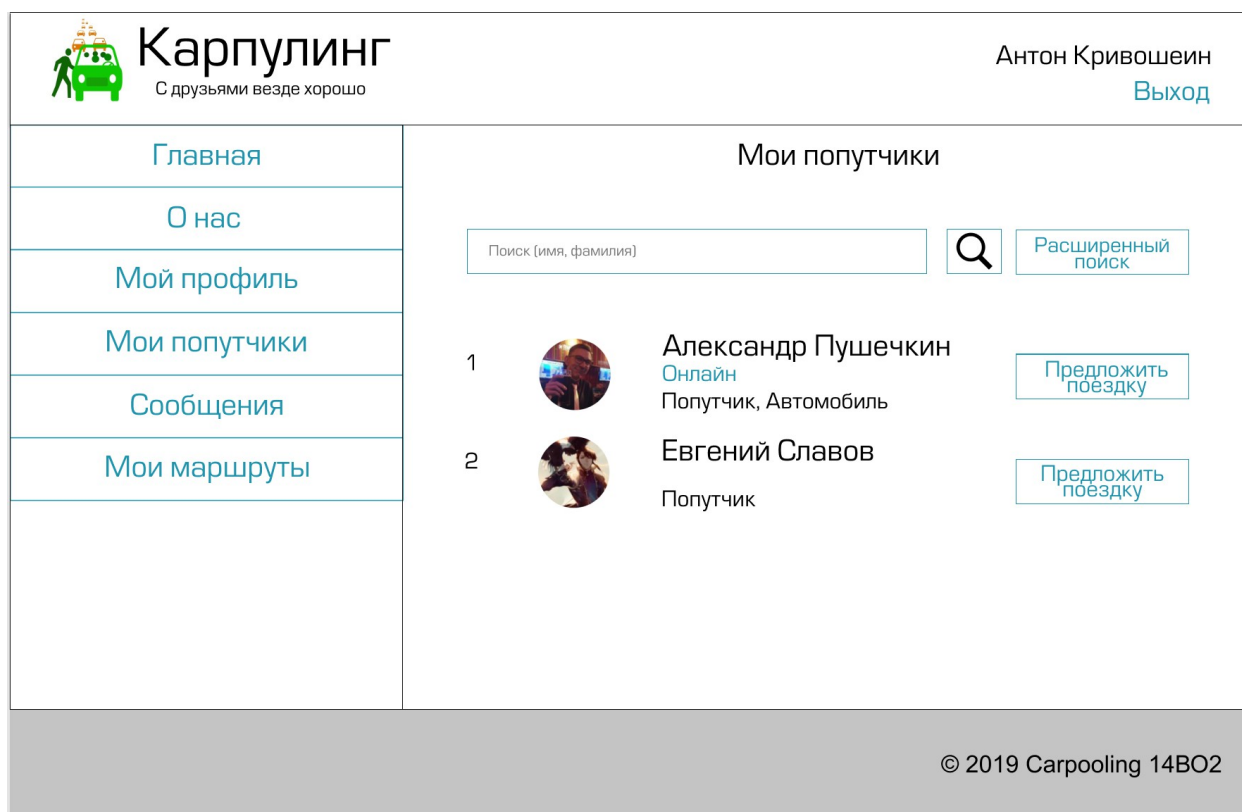


Рисунок 2.14 – Страница просмотра списка друзей



Карпулинг
С друзьями везде хорошо

Антон Кривошеин
[Выход](#)

[Главная](#)
[О нас](#)
[Мой профиль](#)
[Мои попутчики](#)
[Сообщения](#)
[Мои маршруты](#)

Поиск попутчиков

Пол

Мужской ▼

Год рождения

1996

Город

Пенза ▼

Пермь

Петрозаводск

Печора

Автомобиль

☐ Есть

1 Результат поиска



Евгений Славов
Попутчик, Автомобиль

[Предложить поездку](#)

© 2019 Carpooling 14BO2

Рисунок 2.15 – Страница поиска людей

На рисунке 2.16 приведено изображение страницы просмотра списка диалогов. Данная страница содержит список последних диалогов текущего авторизованного пользователя с другими зарегистрированными пользователями. В каждом диалоге выводится фотография собеседника, его имя и фамилия, а также дата и содержимое последнего сообщения. Также для каждого диалога доступна кнопка быстрого удаления всех сообщений в выбранном диалоге.

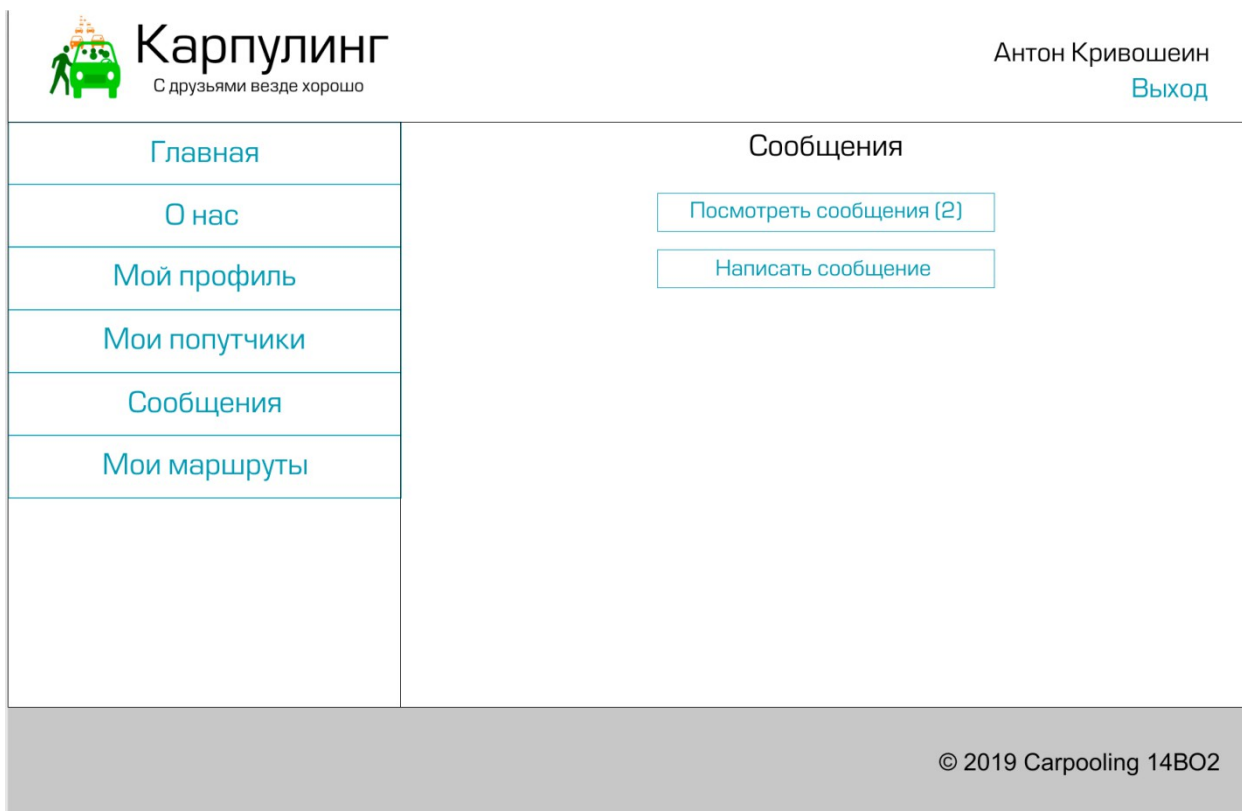


Рисунок 4 – Страница просмотра списка диалогов

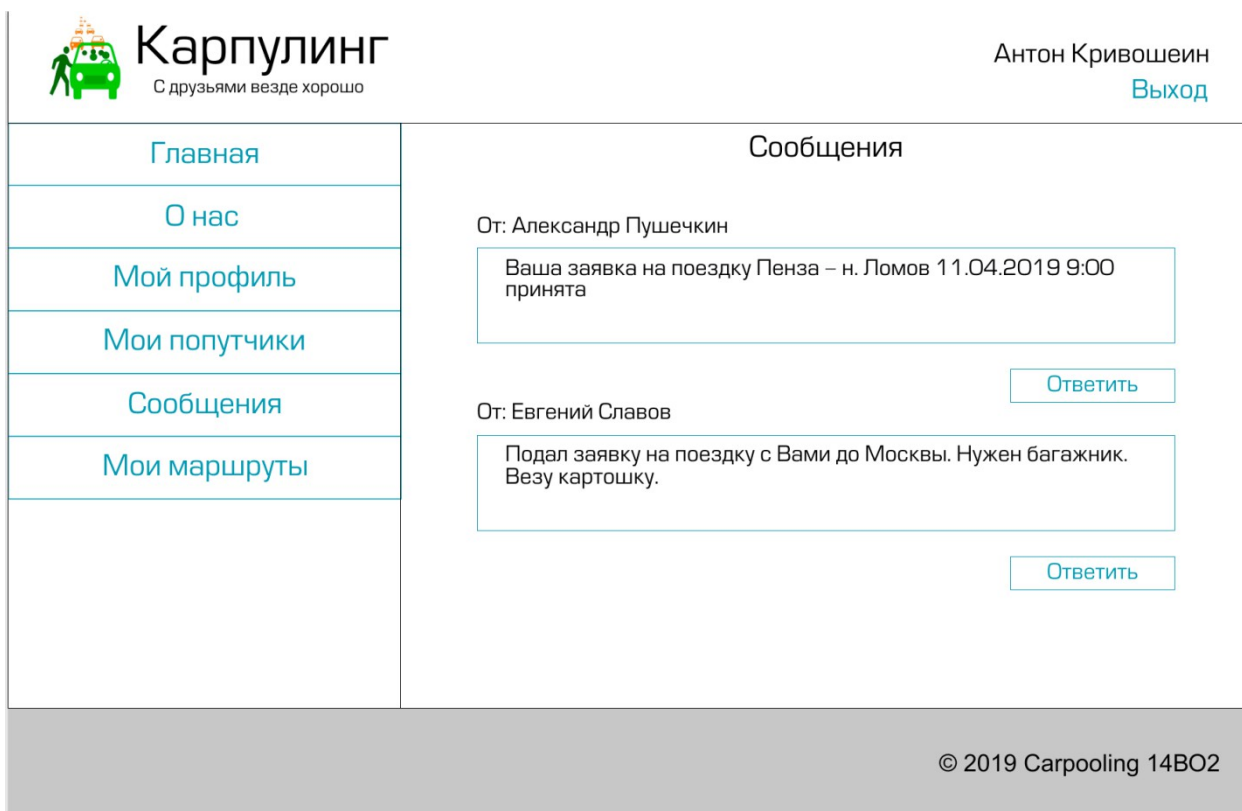


Рисунок 2.16 – Страница диалога между двумя пользователями

На рисунке 2.16 приведено изображение страницы диалога между пользователями. Диалог представляет собой список сообщений, которыми обмениваются собеседники. Сообщения выводятся в хронологическом порядке. Для каждого сообщения отображается отправитель, краткая информация о пользователе, а также текст самого сообщения.

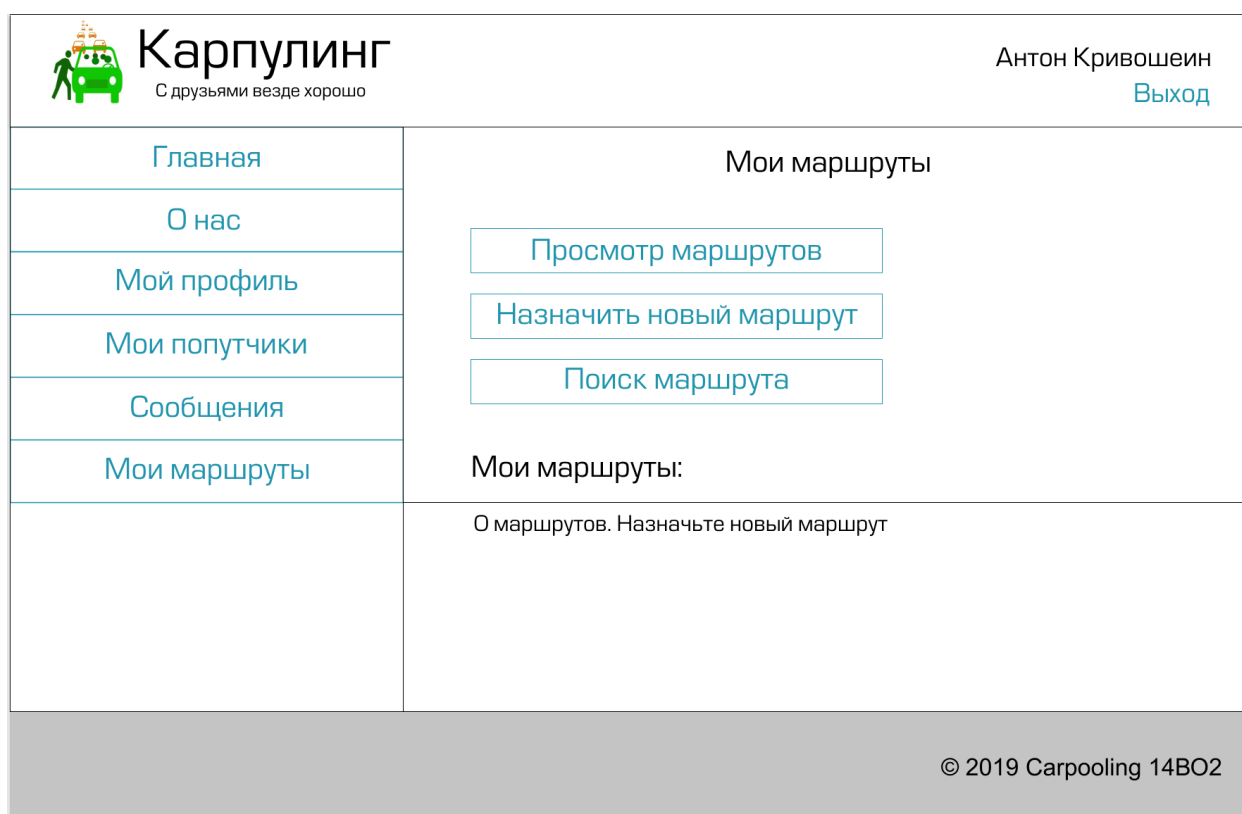


Рисунок 2.17 – Страница просмотра списка маршрутов пользователя

На рисунке 2.17 приведено изображение страницы списка маршрутов пользователя. Список представляет собой таблицу, в которой каждая строка содержит данные о маршруте. Для каждого маршрута выводится его название, место и время отправления, место и время прибытия в конечный пункт, список транзитных пунктов, сведения об авторе маршрута, его автомобиле (если автор – водитель), а также дата и время создания маршрута.

На рисунке 2.18 приведено изображение страницы просмотра выбранного маршрута. Страницу можно условно разделить на две части: левая часть представляет данные в текстовом виде, а правая – в графическом. Информация о маршруте представлена в блочном виде. Сначала отображается список пунктов, из которых состоит маршрут. Для каждого пункта отображается время прибытия в пункт маршрута. Далее следует график поездок, где указывается, в какие дни недели будет совершаться поездка по маршруту. Далее следуют блоки информации о наличии свободных мест и водителе. В правой части страницы расположен блок вывода карты картографического сервиса. На карте отображается маршрут в виде линии синего цвета.



Карпулинг

С друзьями везде хорошо

Антон Кривошеин

Выход

Главная

О нас

Мой профиль

Мои попутчики

Сообщения

Мои маршруты

Назначить новый маршрут

Пункт отправления

Пенза

Пункт назначения

Москва

Время отправления

15.04.2019 11:30

Время прибытия

15.04.2019 23:55

Применить

Детали маршрута

Путь следования

Пункт	Дата прибытия
Мокшан	15.04.2019 12:50
н. Ломов	15.04.2019 15:30
Спасск	15.04.2019 17:30
Зубова Поляна	15.04.2019 18:30
Шацк	15.04.2019 19:00
Рязань	15.04.2019 21:25
Коломна	15.04.2019 22:45
Москва	15.04.2019 23:55

Автомобиль: Автоваз Лада Granta

Количество мест:

Всего

4

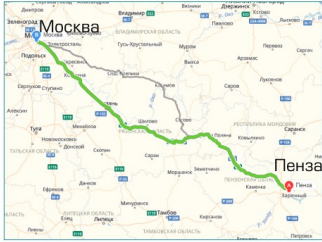
Занято

0

Осталось

4

Сохранить маршрут



© 2019 Carpooling 14BO2

Рисунок 2.18 – Страница просмотра выбранного маршрута

На рисунке 2.19 представлено изображение страницы редактирования выбранного маршрута. Данная страница аналогична странице просмотра, но она содержит элементы редактирования информации о маршруте.

 Карпулинг С друзьями везде хорошо		Антон Кривошеин Выход
Главная	Мои маршруты	
О нас	Просмотр маршрутов	
Мой профиль	Назначить новый маршрут	
Мои попутчики	Поиск маршрута	
Сообщения	Мои маршруты:	
Мои маршруты	О маршрутов. Назначьте новый маршрут	
© 2019 Carpooling 14BO2		

Рисунок 2.19 – Страница редактирования выбранного маршрута

В приложении А приведен листинг клиентской части приложения.

2.4.3 Разработка программного обеспечения серверной части

Разработка программного обеспечения серверной части велась на основе фреймворка PlayFramework.

На рисунке 2.20 представлена диаграмма классов моделей данных приложения. Данные модели являются отображением таблиц из базы данных в классы приложения. Данный подход позволяет оперировать данными в базе

данных как объектами классов, таким образом, реализуя объектно-ориентированный подход к проектированию приложения. Для отображения данных в объекты классов используется фреймворк Hibernate¹, который позволяет переложить работу по синхронизации данных в приложении и базе данных с программиста на автоматизированные модули. Программист использует лишь абстрактные методы для управления данными.

Приведем спецификации некоторых классов приложения.

Класс Application является контроллером, который отображает главную страницу приложения.

```
public class Application extends Controller
```

ClassReference

Класс отображения главной страницы приложения

PublicMemberFunctions

- public static void **index()**

Метод контроллера, выполняющий отображение главной страницы сервиса

Класс Security – предназначен для осуществления контроля доступа пользователей к страницам сервиса.

```
public class Security extends Controller
```

Class Reference

Класс, обеспечивающий безопасность закрытых страниц сервиса. Выполняет проверку прав текущего пользователя. В случае отсутствия прав на просмотр страницы перенаправляет на страницу авторизации.

Public Member Functions

- public static void **checkAndPut()**

Метод проверки прав текущего пользователя

- public static void **setCurrentUser**(User user)

Метод установки текущего пользователя

- public static User **getCurrentUser()**

Метод получения текущего пользователя

Рисунок 2.21 – Диаграмма классов контроллеров (пакет controllers)

Класс **Friends**— предназначен для отображения страниц списков пользователей.

```
public class Friends extends Controller
```

Class Reference

```
public static void all(String tab)
```

Метод, обеспечивающий отображение выбранной страницы просмотра списка пользователей

```
public static void sendRequest(long toUser)
```

Метод отправки заявки на добавление в список друзей

```
public static void acceptRequest(long id)
```

Метод принятия заявки на добавление в друзья

```
public static void removeRequest(long id)
```

Метод отклонения заявки на добавление в друзья

```
public static void removeFriend(long id)
```

Метод удаления выбранного пользователя из списка друзей

Класс Login – предназначен для отображения страницы авторизации/регистрации.

```
public class Login extends Controller
```

Class Reference

```
public static void registrate(String login, String password, String confirm,  
String surname, String name, String email)
```

Метод, выполняющий регистрацию нового пользователя и его автоматическую авторизацию на сервисе.

```
public static void authenticate(String username, String password, boolean  
remember) throws Throwable
```

Метод, выполняющий авторизацию пользователя на сервисе.

```
public static void login
```

Метод, обеспечивающий отображение страницы авторизации/регистрации.

```
public static void logout()
```

Метод, производящий очистку данных о текущем пользователе («выход из системы»).

Класс Messages – предназначен для отображения страницы списка диалогов и страницы переписки двух пользователей.

```
public class Messages extends Controller
```

Class Reference

```
publicstaticvoidall() {
```

Метод, обеспечивающий отображение страницы просмотра списка диалогов пользователя.

```
public static void dialog(long to)
```

Метод, обеспечивающий отображение страницы переписки между двумя пользователями.

```
public static void sendMessage(long to, String text)
```

Метод, выполняющий отправку сообщения.

```
publicstaticvoiddeleteMessage(longid)
```

Метод, выполняющий удаление сообщения.

```
public static void deleteDialog(long user)
```

Метод, выполняющий удаление диалога.

Класс Profiles— предназначен для отображения страниц просмотра и редактирования личного профиля пользователя.

```
public class Profiles extends Controller
```

Class Reference

```
public static void show(long id)
```

Метод, обеспечивающий отображение страницы просмотра личного профиля пользователя.

```
public static void edit(long id)
```

Метод, обеспечивающий отображение страницы редактирования личного профиля пользователя.

```
public static void save(User user)
```

Метод, обеспечивающий сохранение информации о пользователе.

```
public static void addVehicle()
```

Метод, производящий добавление нового автомобиля для текущего пользователя.

```
public static void saveVehicle(Vehicle vehicle)
```

Метод, производящий сохранение данных об автомобиле.

```
public static void uploadFile(File file)
```

Метод, производящий загрузку на сервер файла, указанного пользователем.

```
public static void removeFoto(long id)
```

Метод, выполняющий удаление файла, ранее загруженного на сервер.

Класс `Routes` — предназначен для отображения страниц списка маршрутов, просмотра и редактирования выбранного маршрута.

```
public class Routes extends Controller
```

Class Reference

```
public static void all()
```

Метод, обеспечивающий отображение страницы просмотра списка маршрутов.

```
public static void show(long id)
```

Метод, обеспечивающий отображение страницы просмотра выбранного маршрута.

```
public static void add()
```

Метод, производящий создание и сохранение нового маршрута.

```
public static void edit(long id)
```

Метод, обеспечивающий отображение страницы редактирования выбранного маршрута.

```
public static void addRoutePoint(long route, String title, long type, String arrivalDt)
```

Метод, производящий создание и сохранение нового пункта маршрута.

```
public static void sendRequest(long routeId)
```

Метод, производящий отправку заявки на участие в маршруте.

```
public static void removeRequest(long id)
```

Метод, производящий удаление заявки на участие в маршруте.

```
public static void acceptRequest(long id)
```

Метод, производящий принятие заявки на участие в маршруте.

В приложении А приведен исходный код серверной части приложения.

3. Технология разработки и отладки web-приложений в среде NetBeans

Для реализации данного дипломного проекта была выбрана среда разработки NetBeans.

Эта среда— бесплатная, быстрая, кросс-платформенная, хорошо понимает языки Java и PHP. В ней приятно писать код, потому что присутствует все необходимое для этого инструменты — подсветка, completion из всех подключенных файлов, браузер классов PHP, всплывающая документация по стандартным функциям и так далее. Вместе с тем, это не просто текстовый редактор. NetBeans умеет скачивать и закачивать файлы на удаленные сервера по FTP и SFTP, имеет развитый SQL-терминал и полностью автоматизирует работу с SVN или CVS.

Рассмотрим технологию создания проекта для реализации фронтэнд приложения «Карпулинг».

3.1 Создание проекта в NetBeans

Ниже описана технология создания проекта в среде Eclipse.

Первый шаг, открытие помощника создания проекта - **File > New > Project > PHP > PHP Application**.

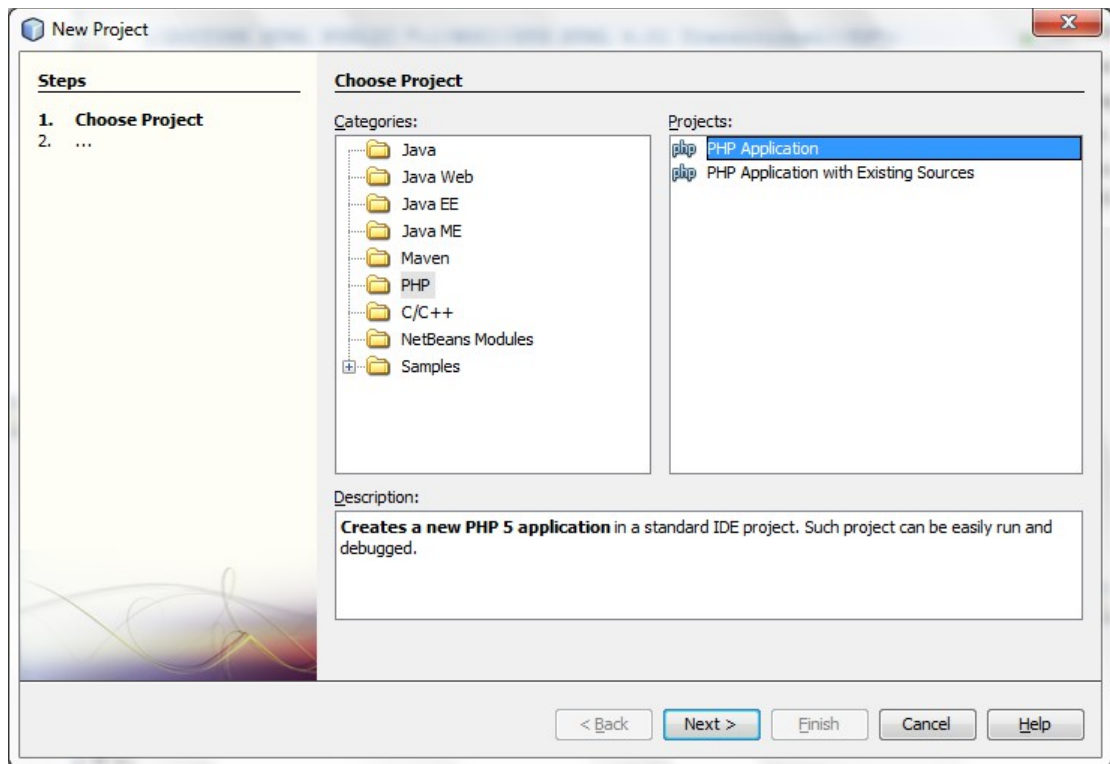


Рисунок 3.1 – Создание нового проекта

Нажмите кнопку **Next** и перейдите в диалог настройки пространства имен проекта (рисунок 3.2).

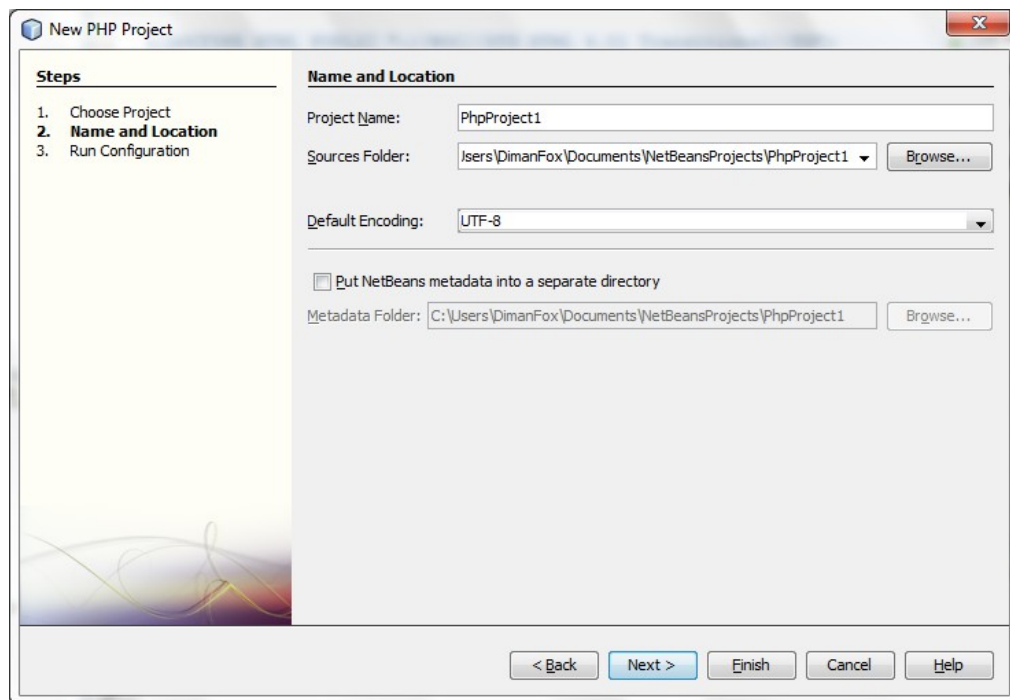


Рисунок 3.2 – Настройка пространства имен проекта

Нажмите кнопку **Next** и перейдите в диалог конфигурации проекта (рисунок 3.3).

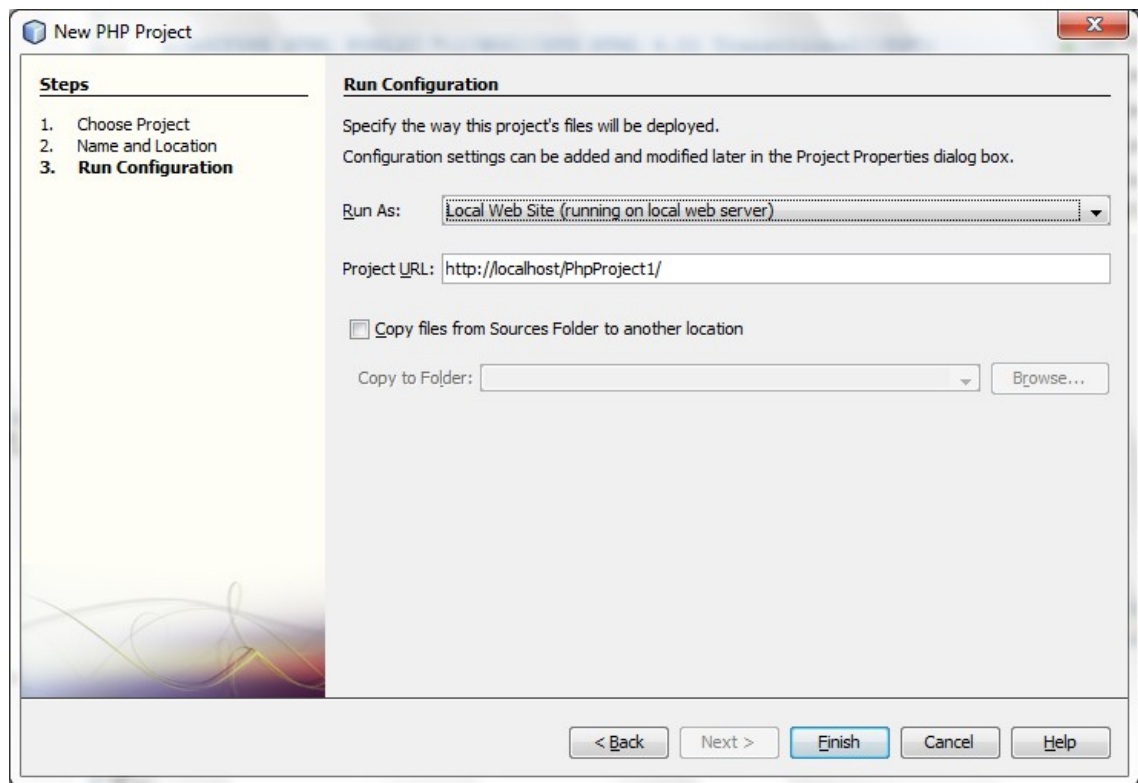


Рисунок 3.3 – Настройка конфигурации проекта

В результате вышеперечисленных операций появляется окно нового проекта, в котором присутствует структура каталога проекта и сгенерированные автоматически файлы настройки (рис. 3.4).

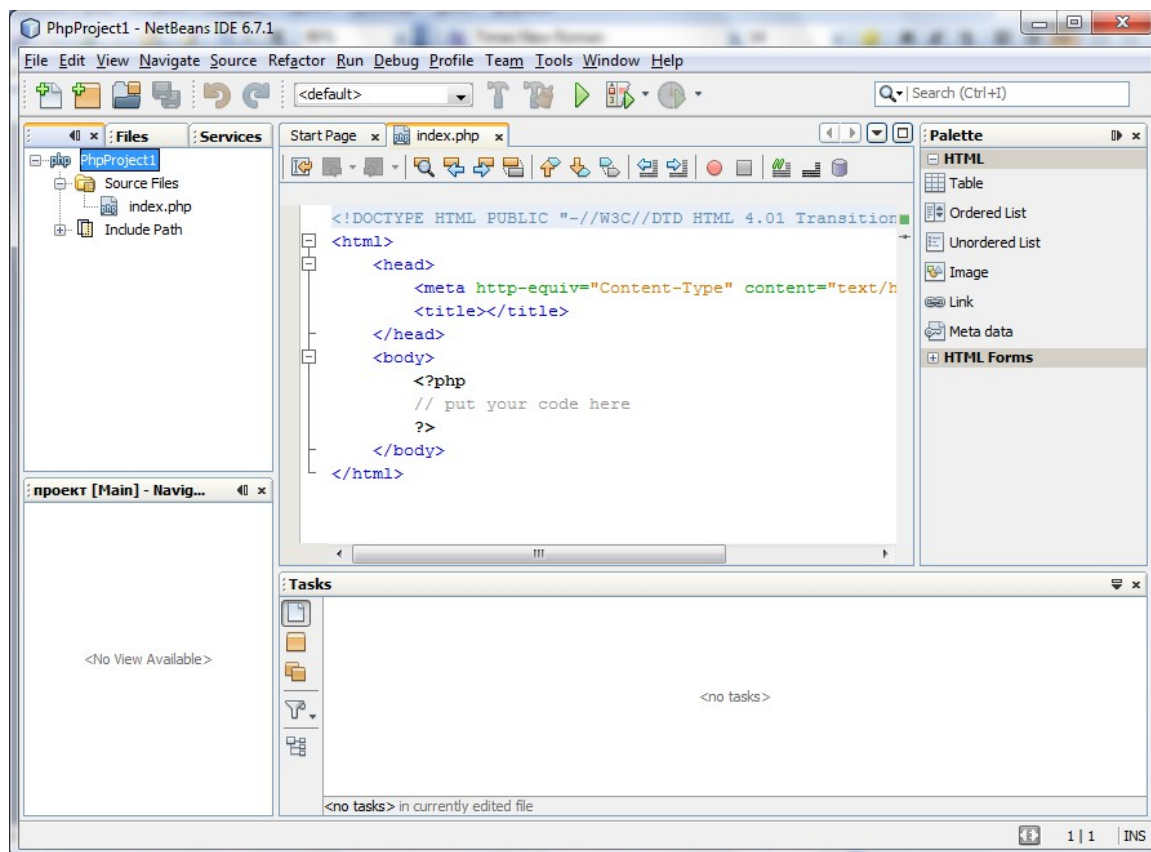


Рисунок 3.4 – Окно нового проекта

По умолчанию в проект добавлен файл `index.php`, который является главным файлом проекта, и к которому будут подключаться остальные файлы.

Для добавления новых файлов в проект нужно воспользоваться пунктом меню - **File > New > NewFile**.

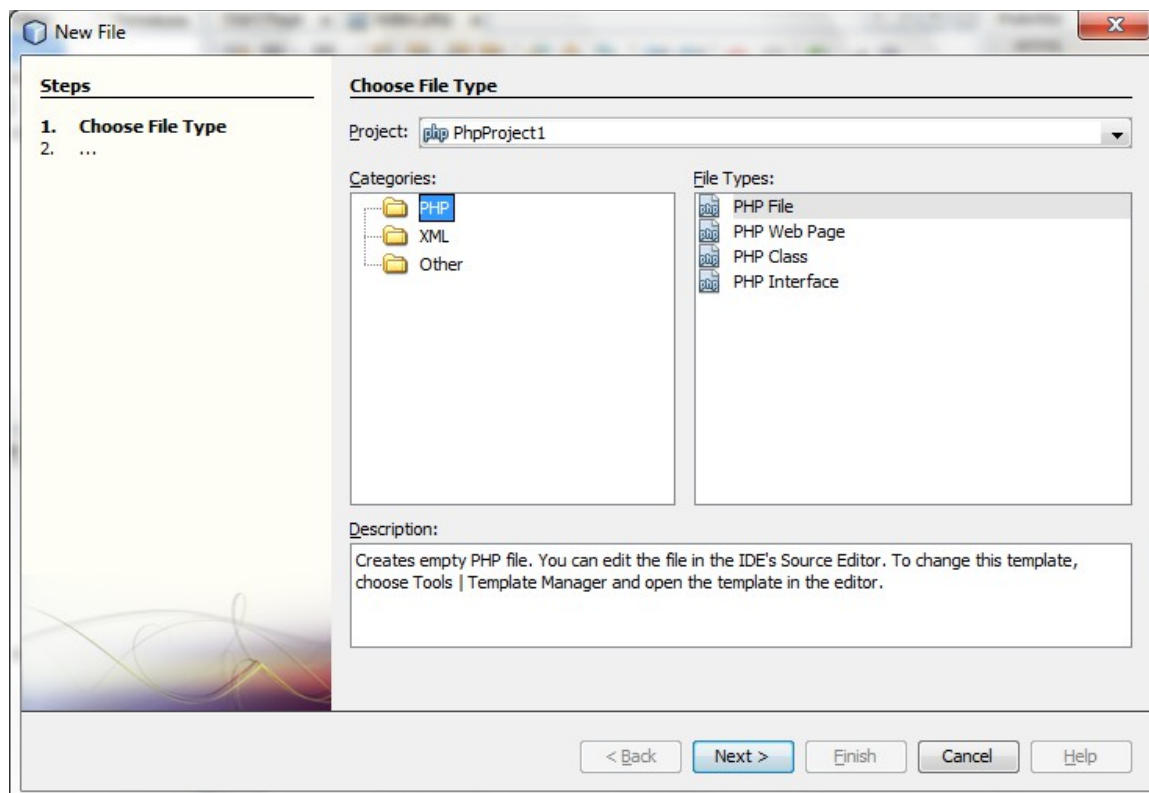


Рисунок 3.5 – Добавление нового файла в проект

После того как выбрали тип файла, который нужно добавить в проект, следует нажать кнопку «Next» для того чтобы задать имя файлу.

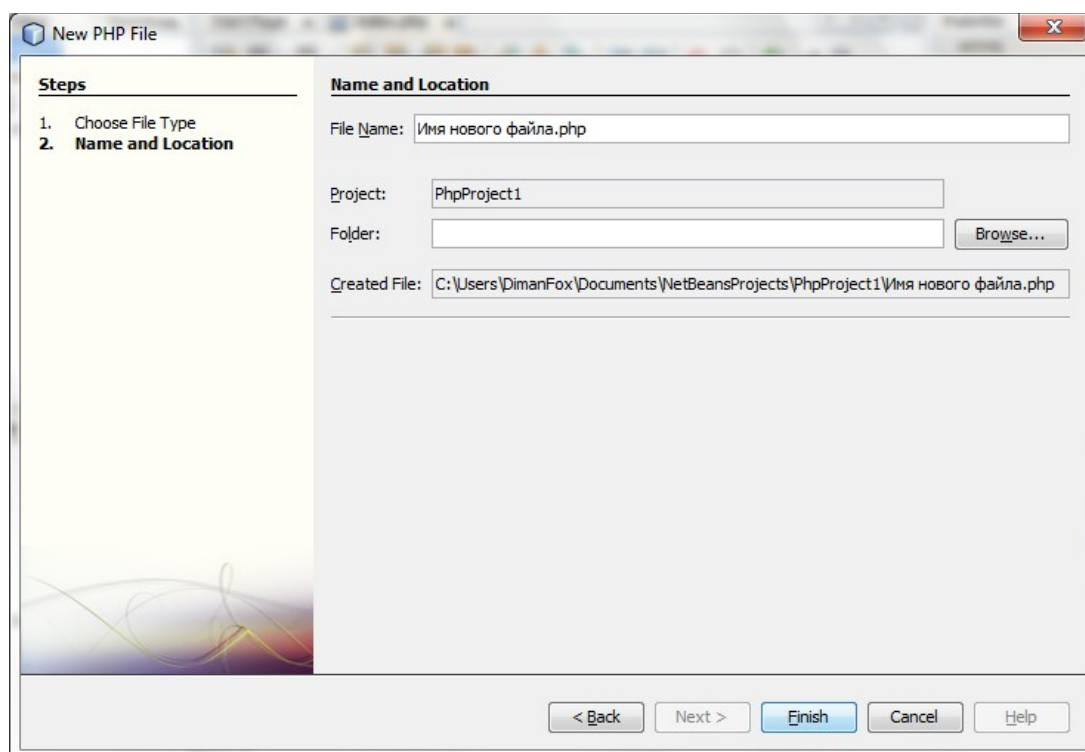


Рисунок 3.6 – Добавление нового файла в проект

После того как задали имя новому файлу, можно нажать кнопку «Finish» и новый файл добавиться в проект с минимум кода-шаблоном.

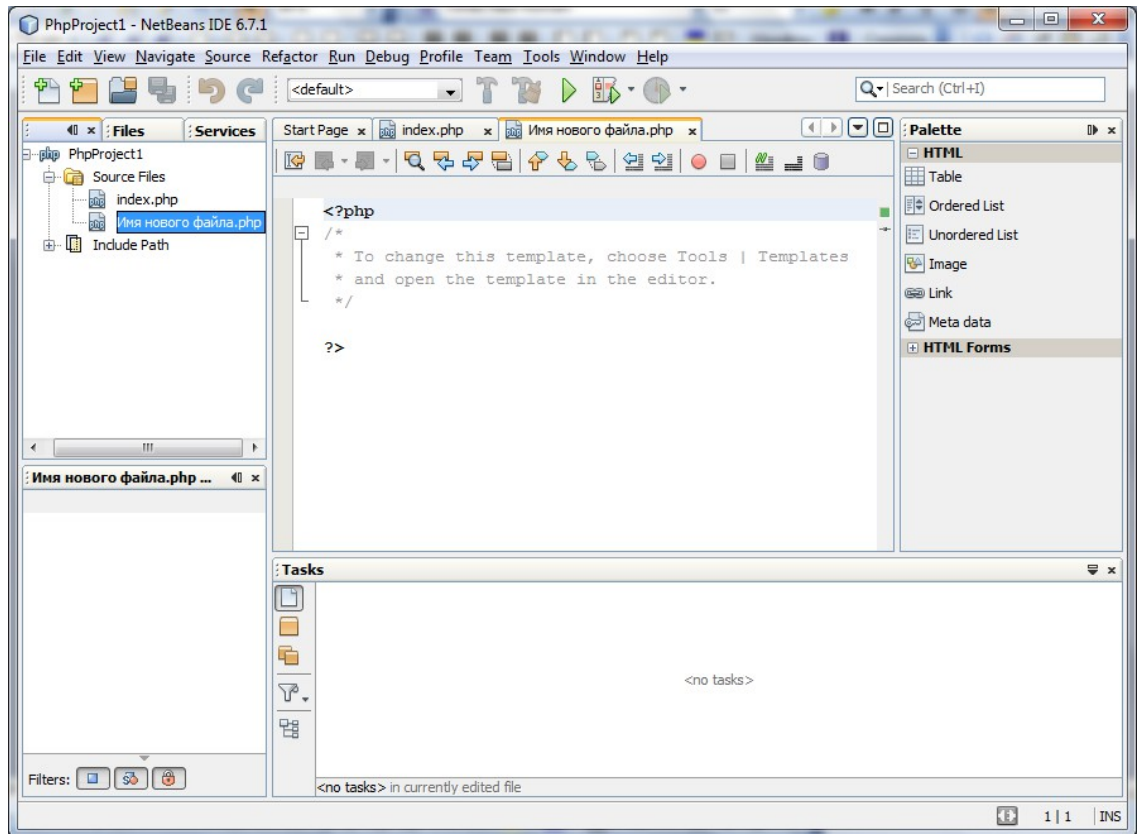


Рисунок 3.7 – Окно проекта

3.2 Отладка кода с помощью XDebug

Отладка программного обеспечения самая ненавистная работа для разработчика. Большинство используют для отладки связку `echo(print_r,var_dump)` и `exit(die)`, переходя от одной строке к другой. Однако, если ошибка появляется вновь в этом файле, требуется заново прописывать отладочные команды. Можно конечно отладку обернуть внутри конструкции `if`, которая будет работать только, когда определена какая-либо константа,

например DEBUG. Но каждый такой if будет слегка замедлять производительность, да и смотреться будет в коде очень ужасно.

Xdebug позволяет создавать логи трассировки, достаточно хороший выход для данной ситуации, и не надо будет изменять программу. Однако лог трассировки, особенно когда он создан для части программы, предоставляет нам много информации, не имеющей никакого отношения к отладке, таким образом, использование отладчика это оптимальное решение. Отладчик позволит вам остановить запуск программы на некоторое время, проверить или модифицировать текущее значение переменной, а затем продолжить выполнение программы. Запуская программу шаг за шагом, вы можете близко взглянуть как выполняется ваш код, что поможет найти вам места ошибок.

В среде разработки NetBeans IDE для отладки PHP кода есть очень удобное решение – Xdebug. Для его настройки необходимо:

В соответствии с версией php, [выбрать и скачать](#) библиотеку xdebug .

Скопировать файл php_xdebug.dll в папку /php/ext/.

В файле php.ini добавить секцию XDebug и прописать в ней следующее:

```
[XDebug]
```

```
extension=php_xdebug-2.0.5-5.2.dll
```

```
zend_extension_ts="C:\server\php\ext\php_xdebug-2.0.5-5.2.dll"
```

```
xdebug.remote_enable=on
```

```
xdebug.remote_handler=dbgp
```

```
xdebug.remote_host=localhost
```

xdebug.remote_port=9000

Перезапускаем Apache

В NetBeans IDE перейтив Tools-Options-PHP. Навкладке General всекции Command Line указать путь до php.exe. Всекции Debugging прописать port 9000 и Session ID netbeans-xdebug.

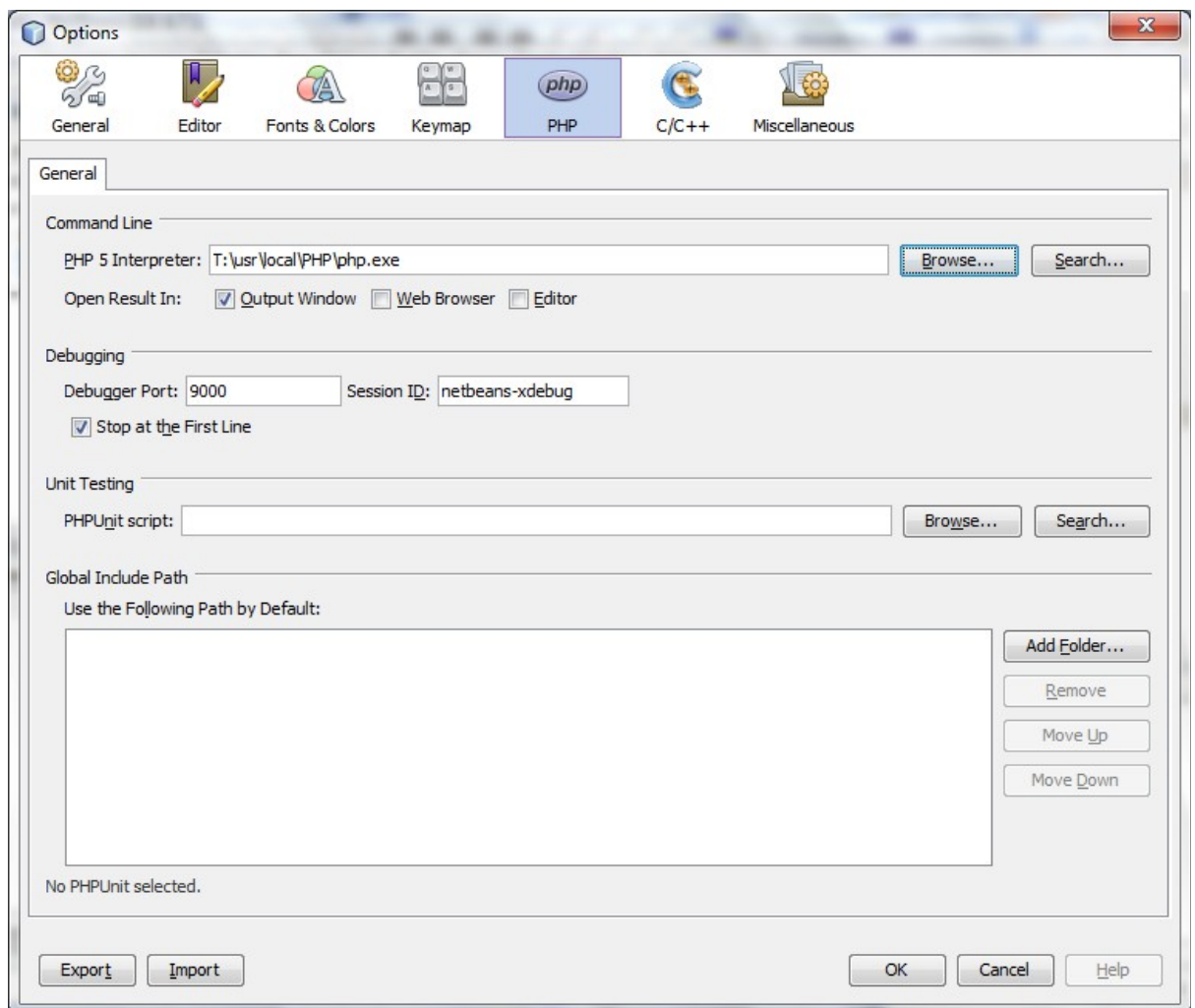


Рисунок 3.8 – Настройка отладчика

3.2.1 Как работает отладка

Когда отладка включена в php.ini, xdebug начинает контролировать выполнение программы, это значит, что xdebug может останавливать выполнение программы в любой момент, а потом запускать ее снова.

Когда выполнение программы остановлено, xdebug может получить информацию о текущем состоянии программы, например, прочитать текущие значения переменных. Также xdebug представляет возможность изменять значения переменных во время выполнения скрипта.

Расширение xdebug – это сервер, ожидающий соединений с клиентами на определенном порту, задаваемом в конфигурации. Существует два протокола, которые могут быть использованы для коммуникации между xdebug-клиентом и xdebug-сервером (GDB и DBGp).

GDB – это старый протокол, который был заменен на DBGp. Посылая команды серверу-xdebug, xdebug-клиент удаленно контролирует выполнение, сообщает PHP о приостановке запуска, выполнении следующей строчки кода или продолжении выполнения программы. Клиент обычно встроен в редактор или IDE (в нашем случае в NetBeans), поэтому вы не будете иметь дело с протоколом общения напрямую.

Веб-сервер с xdebug может быть запущен на другой операционной системе, нежели xdebug-клиент. Вот почему xdebug называется удаленным отладчиком(remote debugger). Для упрощения, мы настроим сервер и клиент на одном и том же компьютере.

Существует два режима начала сессии отладки. Эти режимы контролируются в php.ini настройкой xdebug.remote_mode. Значение по умолчанию req, xdebug-клиент будет соединяться с сервером всякий раз когда начинается выполнение скрипта. Если вы хотите, чтобы xdebug соединялся с сервером только тогда, когда установлен breakpoint или когда в скрипте возникла ошибка, то можно установить xdebug.remote_mode в jit.

Для того чтобы начать отладку, нужно послать параметр XDEBUG_SESSION_START в скрипт как часть GET, POST или COOKIE. Значение этого параметра это название сессии отладки, которое должно быть уникальным, по имени сессии xdebug может различать различные сессии,

запущенные параллельно. Для завершения сессии отладки необходимо послать скрипту команду `XDEBUG_SESSION_STOP`.

Вместо того чтобы вручную начинать и заканчивать отладочные сессии, следует установить специальный `firefox plugin`, который поможет легко начинать и заканчивать сессии одним нажатием мышки.

4. Техничко-экономическое обоснование разработки сайта «сервис совместного поиска попутчиков»

4.1 Расчет себестоимости программного продукта

Процесс разработки Интернет-сайта включает в себя следующие основные этапы:

1. анализ требований к сайту;
2. проектирование, составление алгоритма;
3. написание самой программы;
4. тестирование и отладка;

При этом первые два этапа представляют собой интеллектуальный труд разработчика, который не может быть автоматизирован, а время, затрачиваемое на выполнение этих этапов, зависит только от накопленных знаний и квалификации программиста. Следующие этапы представляет собой работу при переводе разрабатываемой программы на язык программирования и устранения всех ошибок.

При определении затрат времени на разработку Интернет-сайта был использован метод экспертной оценки. В качестве экспертов выступают разработчик программы и руководитель проекта.

На основании экспертных оценок затрат времени определяется средняя оценка, которая рассчитывается по формуле:

$$tcp = (3tp + 2tc) / 5, (4.1)$$

где tp – оценка, данная руководителем проекта;

tc – оценка, данная автором проекта или другим специалистом;

t_{cp} – средняя оценка.

Экспертные оценки времени делаются для трех ситуаций:

1. наименее возможная величина затрат, a_i ;
2. наиболее вероятная величина затрат, m_i ;
3. наиболее возможная величина затрат, b_i .

Результаты расчета средней оценки затрат времени приведем в таблице 4.1:

Таблица 4.1 - Экспертные и средние оценки затрат времени в днях

Этапы разработки программы	Наименее возможная величина затрат, A_i			Наиболее вероятная величина затрат, M_i			Наиболее возможная величина затрат, b_i		
	t_p	t_c	T_{cp}	p	c	T_{tcp}	t_p	t_c	t_{cp}
Анализ требований	2,0	2,0	2,0	3,0	4,0	3,4	4,0	4,5	4,2
Проектирование, составление алгоритма	3,0	3,5	2,6	4,0	4,5	4,2	5,0	6,0	5,4
Написание программы	6,5	7,0	6,7	7,0	8,0	7,4	9,0	10,0	9,4
Тестирование, отладка	6,0	7,5	6,6	7,0	8,0	7,4	12	12	12

Далее определяется математически ожидаемая величина затрат времени и стандартное отклонение для каждого этапа:

$$MO_i = (a_i + 4m_i + b_i)/6, (4.2)$$

$$G_i = (b_i - a_i)/6, (4.3)$$

где a_i , b_i , m_i – средние оценки.

Общая величина математического ожидания и общая величина стандартного отклонения рассчитываются по формулам:

$$MO = \sum MO_i, (4.4)$$

$$(4.5)$$

Коэффициент вариации:

$$Vi = G/MO * 100\%, (4.6)$$

где i – меняется от 1 до n (количество этапов разработки).

Таблица 4.2 - Математически ожидаемая величина затрат времени и стандартное отклонение

Этапы разработки программы	Математически ожидаемая величина затрат времени, MO_i	Стандартное отклонение G_i	Коэффициент вариации, V_i
Анализ требований, t_1	3,3	0,4	12,1
Проектирование, t_2	4,2	0,5	11,9
Написание программы, t_3	7,7	0,5	6,5
Тестирование, t_4	9,8	0,9	9,2

Себестоимость разработки программного продукта включает в себя затраты на зарплату разработчика, а также отчисления во внебюджетные фонды и затраты, связанные с использованием машинного времени. Тогда формула для расчета себестоимости выглядит следующим образом:

$$C = (3/m) * k * (t_1 + t_2) * (1 + k_n) + 8 * t_3 * C_m + 8 * t_4 * C_1 \quad (4.7)$$

где Z – среднемесячная зарплата программиста (примем $Z=20000$ руб.);

m – количество рабочих дней в месяце (примем $m=22$ дня);

t_1 – время, затраченное на анализ требований к программному продукту, чел.дн.

t_2 – время, затраченное на отладку программы, чел.дн.

t_3 – время, затраченное на разработку программы с

использованием машинного времени;

k – коэффициент, учитывающий начисления во внебюджетные фонды,
 $k=1,302$;

k_n – коэффициент, учитывающий накладные расходы, $k_n = 2$.

C_m – стоимость 1 часа машинного времени;

C_1 – стоимость 1 часа работы в сети интернет.

Произведем расчет стоимости одного часа работы машины, для этого воспользуемся формулой:

$$(4.8)$$

где $Z_{гПК}$ – годовые текущие затраты на эксплуатацию персонального компьютера

$T_{гПК}$ – годовой фонд времени полезной работы персонального компьютера.

Годовой фонд времени полезной работы ПК рассчитывается по формуле:

$$T_{гэПК} = n - (\pi + P)(4.9)$$

где n - количество рабочих часов в год. ($n = 1987$ часов).

π - профилактика ($5\text{ч/месс} \cdot 12\text{мес} = 60$ часов);

P – ремонты в год ($6\text{дн} \cdot 8\text{ч/дн} = 48$ часов);

$$T_{гэПК} = 1879\text{ч.}$$

В свою очередь, затраты на эксплуатацию компьютера, вычисляются по формуле:

$$З_{гэПК} = З_{гЗП} + З_{гЭ/Э} + З_{гпр}(4.10)$$

где $З_{гЗП}$ – годовые затраты на заработную плату обслуживающего персонала;

$З_{гЭ/Э}$ – годовые затраты на электроэнергию;

$З_{гпр}$ – прочие затраты в год (включают ремонт, расходные материалы и т.п.).

Годовые затраты на заработную плату обслуживающего персонала можно определить по формуле:

$$З_{гЗП} = З_{ОП} \cdot k_p \cdot k_{ст} \cdot 12(4.11)$$

где ЗОП – заработная плата обслуживающего персонала в месяц. ЗОП = 1000 руб;

k_p – районный коэффициент (1,2);

$k_{ст}$ – страховые взносы (1,302).

$$ЗгЗП = 1000 \cdot 1,2 \cdot 1,302 \cdot 12 = 18748.8 \text{ руб.}$$

Годовые затраты электроэнергии можно определить по формуле:

$$ЗгЭ/Э = M_{уПК} \cdot T_{гПК} \cdot ЦЭ/Э (4.12)$$

где $M_{уПК}$ - установочная мощность ПК (0,4 кВт/час);

$ЦЭ/Э$ - цена электроэнергии в руб./кВт-ч (1,76 руб./кВт-ч)

$$ЗгЭ/Э = 0,4 \cdot 1879 \cdot 1,76 = 1322.8 \text{ руб.}$$

Прочие затраты можно принять равными 5% от стоимости компьютера:

$$Згпр = 20000 \cdot 0,05 = 1000 \text{ руб.}$$

Используя формулу (4.8) можно получить стоимость одного часа работы с компьютером:

$$C_m = 21071.6 / 1879 = 11,21 \text{ руб.}$$

Зная все составляющие формулы, рассчитаем затраты на разработку программного продукта:

$$C=(22000/22)*1,302*(3.3+4.2)(1+2)+8*7.7*15+8*9.8*0.98=30295.83 \text{ руб.}$$

Сравним полученную стоимость со стоимостью доработки программного обеспечения организацией разработчиком либо силами сторонних разработчиков.

Исходя из среднерыночной стоимости часа разработки программного обеспечения на рынке 100 у.е. за час, исходя из минимальных 14 часов необходимых на разработку, получаем, что минимальная стоимость может составить 1400 у.е., при средних расчетных сроках разработки сумма составит 2170 у.е. Таким образом, можно сделать вывод, что более выгодным для организации является разработка операции собственными силами, без привлечения сторонних организаций.

Отсюда программный продукт может быть теоретически продан, а значит для оценки:

$$\Pi = C*(1+(P/100)), (4.13)$$

где С – себестоимость данного программного продукта;

Р – рентабельность (примем Р = 30%);

$$\Pi = 30295.83*(1+(30/100))=39384.58 \text{ руб.}$$

Цена с налогом на добавленную стоимость рассчитывается по формуле:

$$\Pi_{\text{с ндс}} = \Pi * K_{\text{ндс}}, (4.14)$$

где Кндс – коэффициент НДС, равный 1,1; Ц - цена без НДС;

$C_{с\ ндс} = 39384,58 * 1,20 = 46473,81$ руб.

4.2 Эффективность разработки сайта

К основным задачам, которые могут быть решены с помощью Интернет-сайта, можно отнести:

1. предоставление информации о себе, о своих товарах и услугах;
2. выход на новые рынки;
3. формирование постоянной целевой аудитории;
4. получение отклика от клиентов.

Имея свой сайт в сети Интернет, появляется возможность существенно сэкономить на вложении средств в другие виды рекламы. Указав ссылку на телеэкране в ходе видеоролика или добавив ее в макет графического блока в журнале или газете, исчезнет необходимость детального освещения сфер деятельности, размещения исчерпывающей информации о предлагаемых товарах и услугах.

Интернет сайт доступен 24 часа в сутки, а значит, в любое время сможет предоставлять необходимую информацию.

Правильно спроектированный сайт обладает возможностью безграничного расширения информационной структуры – разделы, подразделы, страницы и т.д. Если сайт легче управляется, значит и развивается быстрее, следовательно – быстрее окупается его разработка.

На сайте реализуется:

1. предоставление информации о себе, об услугах;
2. формирование постоянной целевой аудитории;
3. получение отклика от пользователей;
4. оповещение о важных событиях организации;

5. демонстрация результатов работы организации.

В данный момент продукцию организации рекламируют в журналах, на радио, других сайтах. Годовые затраты на рекламу приведены в таблице 4.3.

Таблица 4.3 – Годовые затраты на рекламу

Виды рекламы	До появления сайта, тыс. руб.	После появления сайта, тыс. руб.
Журналы	180	-
Радио	280	280
Другие сайты	200	80
Итого	560	360

Затраты на рекламу были предоставлены организацией. После создания сайта для организации появится возможность отказаться от какого-то вида рекламы или сократить количество рекламодателей, как в случае с другими сайтами, и получить экономию, которая пойдет на окупаемость сайта.

Рассчитаем годовую экономию по рекламе по формуле

$$\Delta \text{Э}_{\text{год}} = A - B, (4.15)$$

где $\Delta \text{Э}_{\text{год}}$ – годовая экономия по рекламе; А - затраты на рекламу до появления сайта; В - затраты на рекламу после появления сайта;

$$\Delta \text{Э}_{\text{год}} = 660 - 360 = 300 \text{ тыс. руб.}$$

Период окупаемости затрат на приобретение веб-сайта компанией вычисляется по формуле:

$$Po = C / \Delta \text{Эгод} \quad (4.16)$$

где C – себестоимость разработанного программного продукта;

$\Delta \text{Эгод}$ – годовая экономия по рекламе.

$$Po = 39384,58 / 300000 = 0,10 \text{ года}$$

Таким образом, после внедрения данного программного продукта он окупится через 0,10 года.

5. Безопасность жизнедеятельности. Организация рабочего места программиста

5.1 Характеристика условий труда рабочего места программиста

Научно-технический прогресс внес серьезные изменения в условия производственной деятельности работников умственного труда. Их труд стал более интенсивным, напряженным, требующим значительных затрат умственной, эмоциональной и физической энергии. Это потребовало комплексного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

В настоящее время компьютерная техника широко применяется во всех областях деятельности человека. При работе с компьютером, программист подвергается воздействию вредных электромагнитных полей (диапазон радиочастот ВЧ, УВЧ и СВЧ), инфракрасного и ионизирующего излучений, статического электричества и т.д.

Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой, высокой напряженностью зрительной работы и достаточно большой нагрузкой на мышцы рук при работе с клавиатурой. Большое значение имеет рациональная конструкция и расположение элементов рабочего места, что важно для поддержания оптимальной рабочей позы программиста.

В процессе работы с компьютером необходимо соблюдать правильный режим труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на

головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

Данный раздел дипломного проекта посвящен рассмотрению следующих вопросов по организации рабочего места программиста:

1. определение оптимальных условий работы программиста;
2. расчет освещенности рабочего места программиста;
3. расчет уровня шума на рабочем месте программиста.

5.2 Требование к рабочему помещению программиста

5.2.1 Окраска помещений

Окраска помещений и мебели должна способствовать созданию благоприятных условий для зрительного восприятия, хорошего настроения.

Источники света, такие как светильники и окна, которые дают отражение от поверхности экрана, значительно ухудшают точность знаков и влекут за собой помехи физиологического характера, которые могут выразиться в значительном напряжении, особенно при продолжительной работе. Отражение, включая отражения от вторичных источников света, должно быть, сведено к минимуму. Для защиты от избыточной яркости окон могут быть применены шторы и экраны.

В зависимости от ориентации окон рекомендуется следующая окраска стен и пола:

окна ориентированы на юг: - стены зеленовато-голубого или светло-голубого цвета; пол - зеленый;

окна ориентированы на север: - стены светло-оранжевого или оранжево-желтого цвета; пол - красновато-оранжевый;

окна ориентированы на восток: - стены желто-зеленого цвета; пол зеленый или красновато-оранжевый;

окна ориентированы на запад: - стены желто-зеленого или голубовато-зеленого цвета; пол зеленый или красновато-оранжевый.

В помещениях, где находится компьютер, необходимо обеспечить следующие величины коэффициента отражения: для потолка: 60...70%, для стен: 40...50%, для пола: около 30%. Для других поверхностей и рабочей мебели: 30...40%.

5.2.2 Освещение рабочего места

Правильно спроектированное и выполненное производственное освещение улучшает условия зрительной работы, снижает утомляемость, способствует повышению производительности труда, благотворно влияет на производственную среду, оказывая положительное психологическое воздействие на программиста, повышает безопасность труда и снижает травматизм.

Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение и резь в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям, поэтому столь важен правильный расчет освещенности.

Существует три вида освещения - естественное, искусственное и совмещенное (естественное и искусственное вместе).

Естественное освещение - освещение помещений дневным светом, проникающим через световые проемы в наружных ограждающих конструкциях помещений. Естественное освещение характеризуется тем, что меняется в широких пределах в зависимости от времени дня, времени года, характера области и ряда других факторов.

Искусственное освещение применяется при работе в темное время суток и днем, когда не удастся обеспечить нормированные значения коэффициента естественного освещения (пасмурная погода, короткий световой день). Освещение, при котором недостаточное по нормам естественное освещение дополняется искусственным, называется совмещенным освещением.

Искусственное освещение подразделяется на рабочее, аварийное, эвакуационное, охранное. Рабочее освещение, в свою очередь, может быть общим или комбинированным. Общее - освещение, при котором светильники размещаются в верхней зоне помещения равномерно или применительно к расположению оборудования. Комбинированное - освещение, при котором к общему добавляется местное освещение.

Согласно СНиП 23-05-2010 в помещениях вычислительных центров необходимо применить систему комбинированного освещения.

При выполнении работ категории высокой зрительной точности (наименьший размер объекта различения 0,3...0,5 мм) величина коэффициента естественного освещения (КЕО) должна быть не ниже 1,5%, а при зрительной работе средней точности (наименьший размер объекта различения 0,5...1,0 мм) КЕО должен быть не ниже 1,0%. В качестве источников искусственного освещения обычно используются люминесцентные лампы типа ЛБ или ДРЛ, которые попарно объединяются в светильники, которые должны располагаться над рабочими поверхностями равномерно.

Требования к освещенности в помещениях, где установлены компьютеры, следующие: при выполнении зрительных работ высокой точности общая освещенность должна составлять 300 лк, а комбинированная - 750 лк; аналогичные требования при выполнении работ средней точности - 200 и 300 лк соответственно.

Кроме того, все поле зрения должно быть освещено достаточно равномерно – это основное гигиеническое требование. Иными словами, степень освещения помещения и яркость экрана компьютера должны быть примерно одинаковыми, т.к. яркий свет в районе периферийного зрения значительно увеличивает напряженность глаз и, как следствие, приводит к их быстрой утомляемости.

5.2.3 Параметры микроклимата на рабочем месте

Параметры микроклимата могут меняться в широких пределах, в то время как необходимым условием жизнедеятельности человека является поддержание постоянства температуры тела благодаря терморегуляции, т.е. способности организма регулировать отдачу тепла в окружающую среду. Принцип нормирования микроклимата – создание оптимальных условий для теплообмена тела человека с окружающей средой.

Вычислительная техника является источником существенных тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении. В помещениях, где установлены компьютеры, должны соблюдаться определенные параметры микроклимата. В СНиП 41-01-2003 установлены величины параметров микроклимата, создающие комфортные условия. Эти нормы устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения (см. табл. 5.1).

Объем помещений, в которых размещены работники вычислительных центров, не должен быть меньше $19,5 \text{ м}^3/\text{человека}$ с учетом максимального числа одновременно работающих всмену. Нормы подачи свежего воздуха в помещения, где расположены компьютеры, приведены в табл. 5.2.

Для обеспечения комфортных условий используются как организационные методы (рациональная организация проведения работ в зависимости от времени года и суток, чередование труда и отдыха), так и

технические средства (вентиляция, кондиционирование воздуха, отопительная система).

Таблица 5.1 Параметры микроклимата для помещений, где установлены компьютеры.

Период года	Параметр микроклимата	Величина
Холодный	Температура воздуха в помещении	22...24 °С
	Относительная влажность	40...60 %
	Скорость движения воздуха	до 0,1 м/с
Теплый	Температура воздуха в помещении	23...25 °С
	Относительная влажность	40...60 %
	Скорость движения воздуха	0,1...0,2 м/с

Таблица 5.2 Нормы подачи свежего воздуха в помещения, где расположены компьютеры

Характеристика помещения	Объемный расход подаваемого в помещение свежего воздуха, м ³ /на одного человека в час
Объем до 20 м ³ на человека	Не менее 30
20...40 м ³ на человека	Не менее 20
Более 40 м ³ на человека	Естественная вентиляция

5.2.4 Требования к шуму и вибрации в рабочих помещениях

Шум ухудшает условия труда оказывая вредное действие на организм человека. Работающие в условиях длительного шумового воздействия испытывают раздражительность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т. д. Такие нарушения в работе ряда органов и систем организма человека могут вызвать негативные изменения в эмоциональном состоянии человека вплоть до стрессовых. Под воздействием шума снижается концентрация внимания, нарушаются физиологические функции, появляется усталость в связи с повышенными энергетическими затратами и нервно-психическим напряжением

ухудшается речевая коммутация. Все это снижает работоспособность человека и его производительность, качество и безопасность труда. Длительное воздействие интенсивного шума [выше 80 дБ (А)] на слух человека приводит к его частичной или полной потере. В табл. 5.3 указаны предельные уровни звука в зависимости от категории тяжести и напряженности труда, являющиеся безопасными в отношении сохранения здоровья и работоспособности.

Таблица 5.3 Предельные уровни звука, дБ, на рабочих местах.

Категория напряженности труда		Категория тяжести труда	
I.		Легкая	
II.		Средняя	
III.		Тяжелая	
IV.		Очень тяжелая	
I. Мало напряженный			
	80		
	80		
	75		
	75		
II. Умеренно напряженный			
	70		
	70		
	65		
	65		
III. Напряженный			
	60		
	60		
	-		
	-		

IV. Очень напряженный

50

50

-

-

Уровень шума на рабочем месте программиста не должен превышать 65 дБА. Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, могут быть облицованы звукопоглощающими материалами. Уровень вибрации в помещениях вычислительных центров может быть снижен путем установки оборудования на специальные виброизоляторы.

5.2.5 Режим труда программиста

Как уже было неоднократно отмечено, при работе с компьютером очень важную роль играет соблюдение правильного режима труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

В табл. 5.4 представлены сведения о регламентированных перерывах, которые необходимо делать при работе на компьютере, в зависимости от продолжительности рабочей смены, видов и категорий трудовой деятельности с ВДТ (видеодисплейный терминал) и ПЭВМ (в соответствии с СанПиН 2.2.4 3359-16 «Санитарно-эпидемиологические требования к физическим факторам на рабочих местах»).

Таблица 5.4 Время регламентированных перерывов

Категория работ	Уровень нагрузки за рабочую смену при видах работы с ВДТ	Суммарное время регламентированных
-----------------	--	------------------------------------

ВДТ или ПЭВМ	Группа А, количество знаков	Группа Б, количество знаков	Группа В, часов	При 8-часовой смене	При 12-часовой смене
I	до 20 000	до 15 000	до 2,0	30	70
II	до 40 000	до 30 000	до 4,0	50	90
III	до 60 000	до 40 000	до 6,0	70	120

Примечание. Время перерывов дано при соблюдении указанных Санитарных правил и норм. При несоответствии фактических условий труда требованиям Санитарных правил и норм время регламентированных перерывов следует увеличить на 30%.

В соответствии со СанПиН 2.2.4 3359-16 все виды трудовой деятельности, связанные с использованием компьютера, разделяются на три группы:

группа А: работа по считыванию информации с экрана ВДТ или ПЭВМ с предварительным запросом;

группа Б: работа по вводу информации;

группа В: творческая работа в режиме диалога с ЭВМ.

Эффективность перерывов повышается при сочетании с производственной гимнастикой или организации специального помещения для отдыха персонала с удобной мягкой мебелью, аквариумом, зеленой зоной и т.п.

5.3 Расчет освещенности рабочего места программиста

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения.

Обычно искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

- по спектральному составу света они близки к дневному, естественному свету;
- обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);
- обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);
- более длительный срок службы.

Расчет освещения производится для комнаты площадью 15 м^2 , ширина которой 5 м, высота - 3 м. Воспользуемся методом светового потока.

Для определения количества светильников определим световой поток, падающий на поверхность по формуле:

, где

F - рассчитываемый световой поток, Лм;

E - нормированная минимальная освещенность, Лк (определяется по таблице).

Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет

$$E = 300 \text{ Лк};$$

S - площадь освещаемого помещения (в нашем случае $S = 15 \text{ м}^2$);

Z - отношение средней освещенности к минимальной (обычно принимается равным 1,1...1,2, принимаем $Z = 1,1$);

K - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение зависит от типа помещения и характера проводимых в нем работ и в нашем случае $K = 1,5$);

n - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемых коэффициентами отражения от стен (P_c) и потолка (P_n)), значение коэффициентов P_c и P_n были указаны выше: $P_c = 40\%$, $P_n = 60\%$. Значение n определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

, где

S - площадь помещения, $S = 15 \text{ м}^2$;

h - расчетная высота подвеса, $h = 2.92 \text{ м}$;

A - ширина помещения, $A = 3 \text{ м}$;

B - длина помещения, $B = 5 \text{ м}$.

Подставив значения получим:

Зная индекс помещения I , по таблице находим $n = 0,22$

Подставим все значения в формулу для определения светового потока F :

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световой поток которых $F = 4320$ Лк.

Рассчитаем необходимое количество ламп по формуле:

, где

N - определяемое число ламп;

F - световой поток, $F = 33750$ Лм;

F_{λ} - световой поток лампы, $F_{\lambda} = 4320$ Лм.

При выборе осветительных приборов используем светильники типа ОД. Каждый светильник комплектуется двумя лампами.

5.4 Расчет уровня шума на рабочем месте

Одним из неблагоприятных факторов производственной среды в ИВЦ является высокий уровень шума, создаваемый печатными устройствами, оборудованием для кондиционирования воздуха, вентиляторами систем охлаждения в самих ЭВМ.

Для решения вопросов о необходимости и целесообразности снижения шума необходимо знать уровни шума на рабочем месте программиста.

Уровень шума, возникающий от нескольких некогерентных источников, работающих одновременно, подсчитывается на основании принципа энергетического суммирования излучений отдельных источников:

где

L_i – уровень звукового давления i -го источника шума;

n – количество источников шума.

Полученные результаты расчета сравниваются с допустимым значением уровня шума для данного рабочего места. Если результаты расчета выше допустимого значения уровня шума, то необходимы специальные меры по снижению шума. К ним относятся: облицовка стен и потолка зала звукопоглощающими материалами, снижение шума в источнике, правильная планировка оборудования и рациональная организация рабочего места программиста.

Уровни звукового давления источников шума, действующих на рабочем месте, представлены в табл. 5.5. Обычно рабочее место программиста оснащено следующим оборудованием: винчестер в системном блоке, вентилятор(ы) систем охлаждения ПК, монитор, клавиатура, принтер, сканер.

Таблица 5.5 Уровни звукового давления различных источников.

Источник шума	Уровень шума, дБ
Жесткий диск	40
Вентилятор	45
Монитор	17
Клавиатура	10
Принтер	45
Сканер	42

Подставив значения уровня звукового давления для каждого вида оборудования в формулу, получим:

$$L_{\Sigma} = 10 \cdot \lg(10^4 + 10^{4,5} + 10^{1,7} + 10^1 + 10^{4,5} + 10^{4,2}) = 49,5 \text{ дБ}$$

Полученное значение не превышает допустимый уровень шума для рабочего места оператора, равный 65 дБ (ГОСТ 12.1.003-83). И если учесть, что вряд ли такие периферийные устройства как сканер и принтер будут использоваться одновременно, то эта цифра будет еще ниже. Кроме того, при работе принтера непосредственное присутствие необязательно, т.к. принтер снабжен механизмом автоподачи листов.

Заключение

В результате выполнения дипломного проекта был разработан интернет-сервис взаимного поиска попутного транспорта и пассажиров.

Серверная часть сервиса реализована средствами языка Java, сервера приложений GlassFish и СУБД PostgreSQL. Клиентская часть сервиса – на HTML5, JavaScript и jQuery. При разработке сервиса большое внимание было уделено интеграции с социальными сетями и картографическими сервисами.

Разработанный сервис предназначен для решения проблемы пробок в современных крупных городах, а также для снижения уровня загрязнения от транспорта путем оптимизации распределения пассажиров между автомобилями. Так как пользователем сервиса может любой человек, имеющий выход в сеть Интернет, нельзя предсказать его уровень компьютерной грамотности. В связи с этим обстоятельством каждый инструмент сервиса имеет интуитивно понятный интерфейс.

При разработке программных средств были учтены все требования, изложенные в техническом задании. Тем не менее, планируется дальнейшее развитие сервиса в направлении интеграции системы с кассами автовокзалов страны для обеспечения пользователям сервиса возможности покупки билета на междугородние автобусы «не выходя из дома».

Было произведено комплексное тестирование разработанных программных средств, в результате которого было установлено полное соответствие разработки и технического задания.

В настоящее время сервис находится в тестовой эксплуатации.

Список использованных источников

1. Биллиг В. А. Основы программирования на JAVA. — М.: Изд-во «Интернет-университет информационных технологий — ИНТУИТ.ру», 2018. — 488 с.
 2. Брукс Ф. Мифический человеко-месяц, или как создаются программные комплексы. — М: «Символ-Плюс», 2010 г. — 304 с.
 3. Ватсон К. PHP. Библия программиста. — М.: Лори, 2018. — 880 с.
 4. Вирт Н. Алгоритмы и структуры данных. — СПб: Невский диалект, 2019 г. — 352 с.
 5. Гиббонз П. Платформа Плаудля Java-программистов. — СПб.: Питер, 2017. — 336 с.
 6. Голуб А. И. JavaиHibernate. Правила программирования. — М: БИНОМ, 2006 г. — 272 с.
 7. Гуннерсон Э. Введение в модели GOMS. Библиотека программиста. — СПб.: Питер, 2016. — 304 с.
 8. Кораблев В. Самоучитель SQL. — СПб.: Питер; Киев: Издательская группа BHV, 2016. — 528 с.
 9. Либерти Д. Программирование на jQuery. — СПб.: Символ-Плюс, 2017. — 688 с.
- Майо Д. JAVA. Искусство программирования. Энциклопедия программиста. — Киев: «ДиаСофт», 2017. — 656 с.

ЛИСТИНГ ПРИЛОЖЕНИЯ

Приложение А

(обязательное)

/controllers/Applications.java

```
package controllers;

import models.User;
import play.mvc.Controller;
import play.mvc.With;

@With(Security.class)
public class Application extends Controller {

    public static void index() {
        User u = Security.getCurrentUser();
        if (u != null) {
            Profiles.show(u.id);
        }
        render();
    }
}
```

/controllers/Friends.java

```
package controllers;

import models.Friendship;
import models.User;
import play.mvc.Controller;
import play.mvc.With;

import java.util.List;

/**
 */
@With(Security.class)
public class Friends extends Controller {
    public static void all(String tab) {
        if (tab == null || tab.isEmpty()) {
            tab = "all";
        }
        User user = Security.getCurrentUser();
        notFoundIfNull(user);
    }
}
```



```

        @Required String confirm,
        String surname,
        String name,
        @Required @Email String email) {

    validation.equals(password, confirm);

    if (validation.hasErrors()) {
        params.flash();
        validation.keep();
        login();
    }

    User user = new User();
    user.email = email;
    user.login = login;
    user.name = name;
    user.surname = surname;
    user.password = password;
    user.role = Role.AUTHORISED;
    user.save();

    Security.setCurrentUser(user);

    Profiles.show(user.id);
}

    public static void authenticate(String username, String password,
    boolean remember) throws Throwable {

        User user = User.find("login = ? AND password = ?", username,
    password).first();
        if (user != null) {
            Security.setCurrentUser(user);
            if (request.isAjax()) {
                renderJSON("{\"result\":\"success\"}");
            } else {
                Application.index();
            }
        } else {
            if (request.isAjax()) {
                renderJSON("{\"result\":\"failed\"}");
            } else {
                params.flash();
                validation.keep();
                login();
            }
        }
    }

    public static void login() {
        User u = Security.getCurrentUser();
        if (u != null) {
            Profiles.show(u.id);
        }
        render();
    }
}

```

```

        public static void logout() {
            Security.setCurrentUser(null);
            if (request.isAjax()) {
                renderJSON("{\"result\":\"success\"}");
            } else {
                Application.index();
            }
        }
    }
}

```

/controllers/Messages.java

```
package controllers;
```

```

import models.Message;
import models.User;
import play.mvc.Controller;
import play.mvc.With;

```

```

import java.util.Collections;
import java.util.Date;
import java.util.List;

```

```
/**
```

```
@With(Security.class)
```

```
public class Messages extends Controller {
```

```
    public static void all() {
```

```
        User user = Security.getCurrentUser();
```

```
        List<Message> messageList = Message.find("sender = ?1 OR  
receiver = ?1", user).fetch();
```

```
        render(messageList);
```

```
    }
```

```
    public static void dialog(long to) {
```

```
        User user = Security.getCurrentUser();
```

```
        notFoundIfNull(user);
```

```
        User toUser = User.findById(to);
```

```
        notFoundIfNull(toUser);
```

```
        List<Message> messageList = Message.find("(sender = ?1 AND  
receiver = ?2) OR (sender = ?2 AND receiver = ?1)", user,  
toUser).fetch();
```

```
        Collections.sort(messageList);
```

```
        render(toUser, messageList);
```

```
    }
```

```
    public static void sendMessage(long to, String text) {
```

```
        User sender = Security.getCurrentUser();
```

```
        notFoundIfNull(sender);
```

```
        User receiver = User.findById(to);
```

```
        notFoundIfNull(receiver);
```

```
        Message msg = new Message();
```

```
        msg.receiver = receiver;
```

```
        msg.sender = sender;
```

```
        msg.sendDt = new Date();
```

```

        msg.status = Message.MESSAGE_DELIVERED;
        msg.text = text;
        msg.save();
        ok();
    }
}

```

140

/controllers/Profiles.java

```

package controllers;

import models.User;
import models.Vehicle;
import play.mvc.Controller;
import play.mvc.With;

import java.io.File;

/**
 */
@With(Security.class)
public class Profiles extends Controller {

    public static void show(long id) {
        User user;
        if (id == 0) {
            user = Security.getCurrentUser();
        } else {
            user = User.findById(id);
        }
        notFoundIfNull(user);
        render(user);
    }

    public static void edit(long id) {
        User user = User.findById(id);
        notFoundIfNull(user);
        render(user);
    }

    public static void save(User user) {
        user.save();
        edit(user.id);
    }
}

```

/controlles/Routes.java

```

package controllers;

import models.*;
import play.data.validation.Required;
import play.mvc.Controller;
import play.mvc.With;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

```

```

import java.util.List;

/**
 */
@With(Security.class)
public class Routes extends Controller {
    public static void all() throws Exception {
        User user = Security.getCurrentUser();
        notFoundIfNull(user);

        List<Route> routeList = Route.find("SELECT pr.route FROM
ParticipateRoute pr WHERE pr.user = ?1 AND pr.type = ?2", user,
ParticipateRoute.TYPE_OWNER).fetch();
        render(routeList);
    }

    public static void show(long id) {
        Route route = Route.findById(id);
        if (route == null) {
            notFound();
        }
        render(route);
    }

    public static void add() {
        Route route = new Route();
        route.author = Security.getCurrentUser();
        route.createDt = new Date();
        Calendar cal = Calendar.getInstance();
        route.departureDt = cal.getTime();
        cal.add(Calendar.DAY_OF_YEAR, 1);
        route.arrivalDt = cal.getTime();
        route.title = "Безымянный маршрут";
        route.save();
        edit(route.id);
    }

    public static void edit(long id) {
        Route route = Route.findById(id);
        notFoundIfNull(route);

        render(route);
    }

    public static void addRoutePoint(@Required long route, @Required
String title, @Required long type, @Required String arrivalDt) {
        if (validation.hasErrors()) {
            error();
        }

        Route r = Route.findById(route);
        if (r == null) {
            error();
        }
    }
}

```

```
Date arrivalDate = null;

SimpleDateFormat formatter = new SimpleDateFormat("HH:mm
dd.MM.yyyy");

try {

    arrivalDate = formatter.parse(arrivalDt);

} catch (ParseException pe) {

    error(pe.getLocalizedMessage());

}


RoutePoint routePoint = new RoutePoint();

routePoint.arrivalDt = arrivalDate;


List<GeoPoint> geoPoints = GeoPoint.find("title = ?1", title).fetch();

if (geoPoints != null && !geoPoints.isEmpty()) {

    routePoint.geoPoint = geoPoints.get(0);

} else {

    GeoPoint gp = new GeoPoint();

    gp.title = title;

    gp.save();

    routePoint.geoPoint = gp;

}
```

```

        routePoint.position = 0;

        routePoint.route = r;

        routePoint.title = title;

        routePoint.type = type;

        routePoint.save();

        ok();

    }

    public static void sendRequest(long routeId) {
        User user = Security.getCurrentUser();
        notFoundIfNull(user);

        Route route = Route.findById(routeId);
        notFoundIfNull(route);

        ParticipateRoute pr = new ParticipateRoute();
        pr.route = route;
        pr.status = ParticipateRoute.STATUS_PENDING;
        pr.type = ParticipateRoute.TYPE_MEMBER;
        pr.user = user;
        pr.save();

        ok();
    }
}

```

/controllers/Security.java

```

package controllers;

import models.Session;
import models.User;
import play.cache.Cache;
import play.mvc.Before;
import play.mvc.Controller;
import play.mvc.Http;
import play.mvc.Util;

/**
 */
public class Security extends Controller {

    @Before
    public static void checkAndPut() {
        renderArgs.put("currentUser", getCurrentUser());
    }

    @Util
    public static void setCurrentUser(User user) {

```

```

        if (user != null) {
            String key = session.getId();
            response.setCookie("key", key, "30d");
            User u = (User) Session.find("SELECT s.user FROM Session s
WHERE s.key = ?1", key).first();
            if (u == null) {
                Session s = new Session();
                s.key = key;
                s.user = user;
                s.remoteAddress = request.remoteAddress;
                s.save();
            }
        } else {
            response.removeCookie("key");
        }
    }

    @Util
    public static User getCurrentUser() {
        if (request.cookies.containsKey("key")) {
            Http.Cookie cookie = request.cookies.get("key");
            return (User) Session.find("SELECT s.user FROM Session s
WHERE s.key = ?1", cookie.value).first();
        }
        return null;
    }

    public static void onDisconnect() {
        Cache.delete("currentUser" + session.getId());
        Application.index();
    }
}

```

/models/types/FileType.java

```

package models.types;

import models.BaseModel;

import javax.persistence.Column;
import javax.persistence.Entity;

/**
 */
@Entity
public class FileType extends BaseModel {
    @Column(nullable = false)
    public String title;
    public int maxSize;
    public String extensions;
}

```

/models/types/FriendshipStatus.java

```

package models.types;

import models.BaseModel;

import javax.persistence.Column;
import javax.persistence.Entity;

```

```

/**
 */
@Entity
public class FriendshipStatus extends BaseModel {
    @Column(nullable = false)
    public String title;
}

```

/models/types/InfoFieldType.java

```

package models.types;

import models.BaseModel;

import javax.persistence.Column;
import javax.persistence.Entity;

```

```

/**
 */
@Entity
public class InfoFieldType extends BaseModel {
    @Column(nullable = false)
    public String title;
}

```

/models/types/MessageStatus.java

```

package models.types;

import models.BaseModel;

import javax.persistence.Column;
import javax.persistence.Entity;

```

```

/**
 */
@Entity
public class MessageStatus extends BaseModel {
    @Column(nullable = false)
    public String title;
}

```

/models/types/ParticipateStatus.java

```

package models.types;

import models.BaseModel;

import javax.persistence.Column;
import javax.persistence.Entity;
/**
 */
@Entity
public class ParticipateStatus extends BaseModel {
    @Column(nullable = false)
    public String title;
}

```

/models/types/ParticipateRoute.java

```

package models.types;

```

```

import models.BaseModel;

import javax.persistence.Column;
import javax.persistence.Entity;

/**
 */
@Entity
public class ParticipateType extends BaseModel {
    @Column(nullable = false)
    public String title;
}

```

/models/types/RoutePointType.java

```

package models.types;

import models.BaseModel;

import javax.persistence.Column;
import javax.persistence.Entity;

/**
    @Entity

    public class RoutePointType extends BaseModel {

        @Column(nullable = false)

        public String title;

    }

```

/models/types/VehicleType.java

```

package models.types;

import models.BaseModel;

import javax.persistence.Column;

import javax.persistence.Entity;

```

`@Entity`

```
public class VehicleType extends BaseModel {
```

```
    @Column(nullable = false)
```

```
    public String title;
```

```
}
```

`/models/BaseModel.java`

```
package models;
```

```
import play.db.jpa.GenericModel;
```

```
import javax.persistence.*;
```

```
/**
```

```
*/
```

```
@MappedSuperclass
```

```
public class BaseModel extends GenericModel {
```

```
    @Id
```

```
    @Column(nullable = false)
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    public long id;
```

```
}
```

`/models/Color.java`

```
package models;
```

```
import javax.persistence.Column;
```

```
import javax.persistence.Entity;
```

```
/**
```

```
*/
```

```
@Entity
```

```
public class Color extends BaseModel {
```

```
    @Column(nullable = false)
```

```
    public String title;
```

```
    @Column(nullable = false)
```

```
    public String hex;
```

```
}
```

`/models/Comment.java`

```
package models;
```

```

import javax.persistence.*;
import java.util.Date;
import java.util.List;

/**
 */
@Entity
public class Comment extends BaseModel {
    public String title;
    @Column(nullable = false)
    public String text;
    @Temporal(value = TemporalType.TIMESTAMP)
    public Date createdt;
    @ManyToOne(fetch = FetchType.LAZY)
    public Comment parentComment;
    @OneToMany(mappedBy = "parentComment")
    public List<Comment> childCommentList;
    @ManyToOne(fetch = FetchType.LAZY)
    public User author;
    @ManyToOne(fetch = FetchType.LAZY)
    public Vehicle vehicle;
    @ManyToOne(fetch = FetchType.LAZY)
    public User user;
    @ManyToOne(fetch = FetchType.LAZY)
    public Route route;
}

```

/models/File.java

```

package models;

import models.types.FileType;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.ManyToOne;

/**
 */
@Entity
public class File extends BaseModel {
    @Column(nullable = false)
    public String title;
    @ManyToOne(optional = false)
    public FileType type;
    @Column(nullable = false)
    public String path;
    @ManyToOne
    public User author;
    @ManyToOne(fetch = FetchType.LAZY)
    public Message message;
    @ManyToOne(fetch = FetchType.LAZY)
    public Vehicle vehicle;
}

```

/models/Friendship.java

```

package models;

```

```

import javax.persistence.Entity;
import javax.persistence.ManyToOne;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import java.util.Date;

/**
 */
@Entity
public class Friendship extends BaseModel {

    public final static long STATUS_PENDING = 1;

    public final static long STATUS_ACCEPTED = 2;

    public final static long STATUS_REJECTED = 3;


    @ManyToOne(optional = false)

    public User owner;

    @ManyToOne(optional = false)

    public User friend;

    public long status = STATUS_PENDING;

    public String message;

    @Temporal(value = TemporalType.TIMESTAMP)

    public Date createDt = new Date();

}

```

/models/GeoPoint.java

```

package models;

```

```
import javax.persistence.Column;

import javax.persistence.Entity;

/**

@Entity

public class GeoPoint extends BaseModel {

    @Column(nullable = false)

    public String title;

    public String coordinates;

}
```

/models/InfoField.java

```
package models;

import models.types.InfoFieldType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.ManyToOne;

@Entity
```

```

public class InfoField extends BaseModel {

    @Column(nullable = false)

    public String title;

    @ManyToOne(optional = false)

    public InfoFieldType type;

    @Column(nullable = false)

    public String value;

    @ManyToOne(optional = false)

    public User user;

}

```

/models/MarkVehicle.java

```

package models;

import javax.persistence.Column;
import javax.persistence.Entity;

@Entity
public class MarkVehicle extends BaseModel {
    @Column(nullable = false)
    public String title;
}

```

/models/Message.java

```

package models;

import play.data.validation.Required;

import javax.persistence.*;
import java.util.Date;
import java.util.List;

/**
 */
@Entity
public class Message extends BaseModel implements Comparable<Message>
{
    public final static int MESSAGE_DELIVERED = 1;
    public final static int MESSAGE_READ = 2;
    public final static int MESSAGE_DELETED = 3;
    @Column(nullable = false)

```

```

        public String text;
        @Column(nullable = false)
        public int status = MESSAGE_DELIVERED;
        @Temporal(value = TemporalType.TIMESTAMP)
        public Date sendDt;
        @Required
        @ManyToOne(optional = false)
        public User sender;
        @Required
        @ManyToOne(optional = false)
        public User receiver;
        @OneToMany(mappedBy = "message", fetch = FetchType.LAZY)
        public List<File> fileList;

        @Override
        public int compareTo(Message msg) {
            return this.sendDt.compareTo(msg.sendDt);
        }
    }
}

```

```

/models/ModelVehicle.java
package models;

```

```

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.ManyToOne;

```

```

/**
 */

```

```

@Entity

```

```

public class ModelVehicle extends BaseModel {
    @Column(nullable = false)
    public String title;
    public int countPlaces;
    @ManyToOne(optional = false)
    public MarkVehicle markVehicle;
}

```

```

/model/ParticipateRoute.java
package models;

```

```

import javax.persistence.*;

```

```

* Time: 23:27

```

```

*/

```

```

@Entity

```

```

public class ParticipateRoute extends BaseModel {
    public final static long TYPE_OWNER = 1;
    public final static long TYPE_MEMBER = 2;
    public final static long STATUS_PENDING = 1;
    public final static long STATUS_ACCEPTED = 2;
    public final static long STATUS_REJECTED = 3;
    @Temporal(value = TemporalType.TIMESTAMP)
    public Date createDt = new Date();
    @Column(nullable = false)
    public long status = STATUS_PENDING;
    @Column(nullable = false)
    public long type = TYPE_MEMBER;
    @ManyToOne(optional = false)

```

```

        public User user;
        @ManyToOne(optional = false)
        public Route route;
    }

```

/model/Role.java

```

package models;

import javax.persistence.Column;
import javax.persistence.Entity;

/**
 */
@Entity
public class Role extends BaseModel {
    public final static long GUEST = 1;
    public final static long AUTHORISED = 2;
    public final static long ADMIN = 3;
    @Column(nullable = false)
    public String title;
}

```

/models/Route.java

```

package models;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 */
@Entity
public class Route extends BaseModel {
    @Column
    public String title;
    @Column
    public String description;
    @Temporal(value = TemporalType.TIMESTAMP)
    public Date createDt;
    @Column(nullable = false)
    @Temporal(value = TemporalType.TIMESTAMP)
    public Date departureDt;
    @Column(nullable = false)
    @Temporal(value = TemporalType.TIMESTAMP)
    public Date arrivalDt;
    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    public User author;
    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    public Vehicle vehicle;
    @OneToMany(mappedBy = "route")
    public List<RoutePoint> routePointList = new
    ArrayList<RoutePoint>();

    public RoutePoint getDeparturePoint() {
        if (routePointList != null && !routePointList.isEmpty()) {
            for (RoutePoint rp : routePointList) {
                if (rp.isDeparturePoint()) {

```

```

        return rp;
    }
}

return null;
}

public RoutePoint getArrivalPoint() {
    if (routePointList != null && !routePointList.isEmpty()) {
        for (RoutePoint rp : routePointList) {
            if (rp.isArrivalPoint()) {
                return rp;
            }
        }
    }
    return null;
}

public List<RoutePoint> getTransitPoints() {
    List<RoutePoint> transitPoints = new ArrayList<>();
    if (routePointList != null && !routePointList.isEmpty()) {
        for (RoutePoint rp : routePointList) {
            if (rp.isTransitPoint()) {
                transitPoints.add(rp);
            }
        }
    }
    return transitPoints;
}
}

```

/models/RoutePoint.java

```

package models;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.ManyToOne;
import java.util.Date;

/**
 */
@Entity
public class RoutePoint extends BaseModel {
    public final static int ROUTE_TYPE_DEPARTURE = 1;
    public final static int ROUTE_TYPE_TRANSIT = 2;
    public final static int ROUTE_TYPE_ARRIVAL = 3;
    @Column(nullable = false)
    public String title;
    @Column(nullable = false)
    public long type;
    @Column(nullable = false)
    public int position;
    @Column(nullable = false)
    public Date arrivalDt;
    @ManyToOne
    public GeoPoint geoPoint;
    @ManyToOne(optional = false)
    public Route route;
}

```

```

    public boolean isDeparturePoint() {
        return type == ROUTE_TYPE_DEPARTURE;
    }

    public boolean isArrivalPoint() {
        return type == ROUTE_TYPE_ARRIVAL;
    }

    public boolean isTransitPoint() {
        return type == ROUTE_TYPE_TRANSIT;
    }
}

```

/models/Session.java

```

package models;

import javax.persistence.*;
import java.util.Date;

/**
 */
@Entity
public class Session extends BaseModel {
    @Column(nullable = false)
    public String key;
    @Temporal(value = TemporalType.TIMESTAMP)
    public Date createDt;
    @Column
    public String remoteAddress;
    @ManyToOne(optional = false)
    public User user;
}

```

/models/User.java

```

package models;

import play.data.validation.Email;
import play.data.validation.Max;
import play.data.validation.Min;
import play.data.validation.Required;

import javax.persistence.*;
import java.util.Date;
import java.util.List;

/**
 */
@Entity
@Table(name = "uuser")
public class User extends BaseModel {
    @Column
    public String surname;
    @Column(nullable = false)
    public String name;
    @Column
    public String fatherName;
    @Email

```

```

@Column(nullable = false)
public String email;
@Column(name = "roleuid", nullable = false)
public long role = Role.AUTHORISED;
@Column
@Required
@Max(value = 64)
@Min(value = 2)
public String login;
@Column
@Required
@Max(value = 128)
@Min(value = 2)
public String password;
@Column
@Temporal(value = TemporalType.DATE)
public Date birthDate;
@OneToMany(mappedBy = "user")
public List<Session> sessionList;

public String getSurnameWithName() {
    String result = surname != null ? surname + " " : "";
    result += name != null ? name : "";
    return result;
}

public Session getSessionByKey(String key) {
    if (sessionList != null && !sessionList.isEmpty() && key !=
null && !key.isEmpty()) {
        for (Session s : sessionList) {
            if (s.key.equals(key)) {
                return s;
            }
        }
    }
    return null;
}
}

```

/models/Vehicle.java

```

package models;

import models.types.VehicleType;

import javax.persistence.*;
import java.util.Date;
import java.util.List;

/**
 */
@Entity
public class Vehicle extends BaseModel {
    public String title;
    @Temporal(value = TemporalType.TIMESTAMP)
    public Date createDt;
    @Column(nullable = false)
    public String regNumber;
    @Column(nullable = false)

```

```

    public int year;
    @ManyToOne(optional = false)
    public Color color;
    @ManyToOne(optional = false)
    public VehicleType vehicleType;
    @ManyToOne(optional = false)
    public User owner;
    @ManyToOne(optional = false)
    public ModelVehicle model;
    @OneToMany(mappedBy = "vehicle", fetch = FetchType.LAZY)
    public List<File> fileList;
}

```

/views/main.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```

    <title>#{get 'title' /} :: Едем вместе</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

```

```
    <meta name="description" content="">
```

```
    <meta name="author" content="">
```

```
    <meta charset="{$_response_encoding}">
```

```

    <link rel="stylesheet" media="screen"
href="@{'/public/css/bootstrap.min.css'}"/>

```

```

    <link rel="stylesheet" media="screen"
href="@{'/public/css/style.css'}"/>

```

```

    <link rel="stylesheet" media="screen"
href="http://yandex.st/jquery-
ui/1.8.20/themes/flick/jquery.ui.all.min.css"/>

```

```
    <!--[if lt IE 9]>
```

```
    <script
```

```
src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
```

```
    <![endif]-->
```

```
#{get 'moreStyles' /}
```

```

    <script src="http://yandex.st/jquery/1.7.2/jquery.min.js"
type="text/javascript"

```

```
        charset="{$_response_encoding}"></script>
```

```

    <script src="http://yandex.st/jquery-ui/1.8.20/jquery-ui.min.js"
type="text/javascript"

```

```
        charset="{$_response_encoding}"></script>
```

```

    <script src="@{'/public/js/bootstrap.min.js'}"
type="text/javascript" charset="{$_response_encoding}"></script>

```

```

    <script src="@{'/public/js/common.js'}" type="text/javascript"
charset="{$_response_encoding}"></script>

```

```
#{get 'moreScripts' /}
```

```
</head>
```

```
<body>
```

```
<header>
```

```
    <div class="navbar navbar-fixed-top">
```

```
        <div class="navbar-inner">
```

```
            <div class="container">
```

```

                <a class="btn btn-navbar" data-toggle="collapse"
data-target=".nav-collapse">

```

```
                    <span class="icon-bar"></span>
```

```

        <span class="icon-bar"></span>
        </a>
        <a class="brand" href="/">ЕдемВместе</a>

        <div class="btn-group pull-right">
        #{if currentUser != null}
            <a class="btn dropdown-toggle" data-
toggle="dropdown" href="#">
                <i class="icon-user"></i>${
currentUser.surname} ${currentUser.name}
                <span class="caret"></span>
            </a>
            <ul class="dropdown-menu">
                <li><a
href="@{Profiles.show(currentUser.id)}">Профиль</a></li>
                <li class="divider"></li>
                <li><a
href="@{Login.logout()}">Выход</a></li>
            </ul>
        #{/if}
        #{if !currentUser}
            <a data-toggle="modal" class="btn"
href="#dlgLogin"><i
                class="icon-user"></i>Войти</a>
        #{/if}
        </div>
        <div class="nav-collapse">
            <ul class="nav">
                <li #{if request.controller ==
'Application'}class="active"#{/if}><a href="/">Главная</a></li>
                <li #{if request.controller ==
'Routes'}class="active"#{/if}><a href="/routes">Маршруты</a></li>
                <li #{if request.controller ==
'Friends'}class="active"#{/if}><a href="/friends">Друзья</a></li>
                <li #{if request.controller ==
'Messages'}class="active"#{/if}><a href="/im">Сообщения</a></li>
            </ul>
            <form class="navbar-search pull-right">
                <input type="text" class="search-
query" placeholder="Поиск">
            </form>
        </div>

    </div>
</div>
<div class="container">
    #{get 'moreNav'/}
</div>
</header>
<div class="container">
    <div class="row-fluid">
        #{doLayout /}
    </div>
    <footer>
        <div class="form-actions">

```

```

        </div>
    </footer>
</div>
#{if !currentUser}
<div class="modal hide fade" id="dlgLogin">
    <div class="modal-header">
        <button class="close" data-dismiss="modal">×</button>
        <h3>Авторизация</h3>
    </div>
    <div class="modal-body" style="padding-bottom: 0;">
        <form action="{@Login.authenticate()}" method="post"
class="form-horizontal"
        onsubmit="EV.login(this);return false;">
            <div class="control-group">
                <label class="control-label">Имя
пользователя:</label>

                <div class="controls">
                    <input type="text" name="username"
id="username" value="{flash.login}" required="required"/>
                </div>
            </div>
            <div class="control-group">
                <label class="control-label">Пароль:</label>

                <div class="controls">
                    <input type="password" name="password"
id="password" value=""/>
                </div>
            </div>
            <div class="control-group">
                <div class="controls">
                    <label for="remember" class="checkbox">
                        Запомнить меня
                        <input type="checkbox" id="remember"
name="remember"
                                value="true" $
{flash.remember ? 'checked="true"' : ''} />
                    </label>

                    <pclass="help-block">Включите, если хотите,
чтобы система узнавала Вас при
                        каждом входе</p>
                </div>
            </div>
            <div class="form-actions">
                <input type="submit" value="Войти" class="btn
btn-primary"/>
                <input type="button" value="Напомнить пароль"
class="btn"/>
            </div>
        </form>
    </div>
</div>
#{/if}
</body>
</html>

```

/views/Application/index.html

```
{extends 'main.html' /}
```

```
{set title:'Главная страница' /}
```

```
<div class="hero-unit">
```

```
  <h1>Едем вместе</h1>
```

```
  <p>Надоело толкаться в общественном транспорте в час пик? Надоело
  ездить в пустой машине и платить за бензин?</p>
```

```
  <p>Этот сервис поможет Вам решить Ваши проблемы!</p>
```

```
  <p><a class="btn btn-primary btn-large"
  href="{Login.login()}">Зарегистрироваться &raquo;</a></p>
</div>
```

```
<div class="row-fluid">
```

```
  <div class="span3">
```

```
    <h2>Поиск пассажиров</h2>
```

```
    <p>У Вас есть собственное авто? Часто приходится ездить в
    одиночестве? Надоело платить за бензин? На этом сайте
```

```
      Вы найдете себе попутчиков, с которым будет приятно
    путешествовать или добираться до работы. А также Вы
    сможете сократить свои расходы на топливо!
```

```
    </p>
```

```
  </div>
```

```
  <div class="span3">
```

```
    <h2>Поиск попутных автомобилей</h2>
```

```
    <p>
```

```
      Нет автомобиля, но хочется комфорта? Надоело толкаться
    в общественном транспорте? Работа далеко от дома?
```

```
      На этом сайте Вы сможете найти себе попутное авто,
    добираться на котором до работы станет просто и
    комфортно!
```

```
    </p>
```

```
  </div>
```

```
  <div class="span3">
```

```
    <h2>Общение с друзьями</h2>
```

```
    <p>
```

```
      Нашли постоянного попутчика? Понравился водитель
    автомобиля? Хочется продолжить общение? С нашим сервисом
```

```
      это просто - добавляйтесь в друзья и общайтесь! Будте
    в курсе новостей своих друзей!
```

```
    </p>
```

```
  </div>
```

```
  <div class="span3">
```

```
    <h2>Обсуждения</h2>
```

```
    <p>
```

```
      Понравилась поездка? Классный автомобиль? Оставьте
    свой комментарий - пусть другие узнают об этом! С нашим
```

```
      сайтом это просто - здесь можно комментировать всё и
    всех! Транслируйте свои комментарии в профиль
```

```
      социальной сети! Приглашайте к обсуждению друзей!
```

```
    </p>
```

```
  </div>
```

/views/Friends/all.html

```
{% extends 'main.html' %}
{% set title: 'Друзья' %}
```

```
<div class="row-fluid">
<ul class="nav nav-tabs" id="tabs">
  <li {% if tab?.equals('all') %}class="active"{% /if %}><a
href="@{Friends.all('all')}">Все друзья</a></li>
  <li {% if tab?.equals('online') %}class="active"{% /if %}><a
href="@{Friends.all('online')}">Друзья онлайн</a></li>
  <li {% if tab?.equals('requests') %}class="active"{% /if %}><a
href="@{Friends.all('requests')}">Предложения
    дружбы</a>
  </li>
  <li {% if tab?.equals('search') %}class="active"{% /if %}><a
href="@{Friends.all('search')}">Поиск людей</a></li>
</ul>

<div class="tab-content">
  <div class="tab-pane {% if tab?.equals('all') %}active{% /if %}"
id="all">
    <table class="table table-striped table-condensed">
      <thead>
        <tr>
          <th>Фото</th>
          <th>Фамилия и имя</th>
          <th>Контактные данные</th>
          <th>Действия</th>
        </tr>
      </thead>
      {% list items:friendshipList, as:'friendship' %}
      <tr>
        <td>
          
        </td>
        <td>
          <a href="@{Profiles.show(friendship.friend.id)}"
title="Просмотреть профиль">
            {% friendship.friend.surnameWithName %}
          </a>
        </td>
        <td>
          <a href="mailto:{% friendship.friend.email %}">
            <i class="icon-envelope"></i>
            {% friendship.friend.email %}
          </a>
        </td>
        <td>
          <button class="btn btn-primary">
            <i class="icon-envelope icon-
white"></i>
            Написать сообщение
          </button>
          <button class="btn">
            <i class="icon-remove"></i>
            Удалить из друзей
          </button>
        </td>
      </tr>
    </table>
  </div>
</div>
```

```

        </button>
      </td>
    </tr>
  </list>
</table>
</div>
<div class="tab-pane #{if tab?.equals('online')}active#{/if}"
id="online">
  <table class="table table-striped table-condensed">
    <thead>
      <tr>
        <th>Фото</th>
        <th>Фамилия и имя</th>
        <th>Контактные данные</th>
        <th>Действия</th>
      </tr>
    </thead>
    <#{list items:requests, as:'friendship'}>
      <tr>
        <td>
          
        </td>
        <td>
          <a
href="@{Profiles.show(friendship.friend.id)}" title="Просмотреть
профиль">
            <#{friendship.friend.surnameWithName}>
          </a>
        </td>
        <td>
          <a href="mailto:#{friendship.friend.email}">
            <i class="icon-envelope"></i>
            <#{friendship.friend.email}>
          </a>
        </td>
        <td>
          <button class="btn btn-primary">
            <i class="icon-envelope icon-
white"></i>
            Написать сообщение
          </button>
          <button class="btn">
            <i class="icon-remove"></i>
            Удалить из друзей
          </button>
        </td>
      </tr>
    </list>
  </table>
</div>
<div class="tab-pane #{if tab?.equals('requests')}active#{/if}"
id="requests">
  <table class="table table-striped table-condensed">
    <thead>
      <tr>
        <th>Фото</th>
        <th>Фамилия и имя</th>

```

```

        <th>Контактные данные</th>
        <th>Действия</th>
    </tr>
</thead>
#{list items:requests, as:'friendship'}
<tr>
    <td>
        
    </td>
    <td>
        <a
href="@{Profiles.show(friendship.friend.id)}" title="Просмотреть
профиль">
            ${friendship.friend.surnameWithName}
        </a>
    </td>
    <td>
        <a href="mailto:${friendship.friend.email}">
            <i class="icon-envelope"></i>
            ${friendship.friend.email}
        </a>
    </td>
    <td>
        <button class="btn btn-primary">
            <i class="icon-ok icon-white"></i>
            Принять заявку
        </button>
        <button class="btn btn-danger">
            <i class="icon-ban-circle icon-
white"></i>
            Отклонить заявку
        </button>
    </td>
</tr>
</table>
#{/list}
</div>
<div class="tab-pane #{if tab?.equals('search')}active#{/if}"
id="search">
    <form action="#" class="well form-search">
        <div>
            <input type="text" placeholder="Введите слово для
поиска" class="input-xlarge" style="width: 90%">
            <button class="btn btn-primary">
                <i class="icon-search icon-white"></i>
                Найти
            </button>
        </div>

        <div style="margin-top: 10px">
            Пол:
            <select class="input-medium">
                <option>Мужской</option>
            </select>
            &nbsp;
            Год рождения:
            <select class="input-medium">

```

```

        <option>12.09.1989</option>
    </select>
    &nbsp;
    Страна:
    <select class="input-medium">
        <option>Россия</option>
    </select>
    &nbsp;
    Город:
    <select class="input-medium">
        <option>Пенза</option>
    </select>
</div>
</form>
<table class="table table-striped table-condensed">
    <thead>
    <tr>
        <th>Фото</th>
        <th>Фамилия и имя</th>
        <th>Контактные данные</th>
        <th>Действия</th>
    </tr>
    </thead>
    #{list items:requests, as:'friendship'}
    <tr>
        <td>
            
        </td>
        <td>
            <a href="@{Profiles.show(friendship.friend.id)}"
title="Просмотреть профиль">
                ${friendship.friend.surnameWithName}
            </a>
        </td>
        <td>
            <div>
                <a href="mailto:$
{friendship.friend.email}">
                    <i class="icon-envelope"></i>
                    ${friendship.friend.email}
                </a>
            </div>
        </td>
        <td>
            <button class="btn btn-primary">
                <i class="icon-plus-sign icon-
white"></i>
                Добавить в друзья
            </button>
            <button class="btn btn-danger">
                <i class="icon-envelope icon-
white"></i>
                Написать сообщение
            </button>
        </td>
    </tr>

```

```

        #}/{list}
      </table>
    </div>
  </div>
</div>

```

/views/Login/login.html

```

#{extends 'main.html' /}
#{set 'title'}Авторизация#{/set}
<div class="page-header">
  <h1>Авторизация</h1>
</div>

```

```

#{if flash.error}
<div class="alert alert-error">
  &{flash.error}
</div>
#{/if}
#{if flash.success}
<div class="alert alert-success">
  &{flash.success}
</div>
#{/if}
<div class="row-fluid">
  <div class="span6">
    <h2>Вход</h2>

```

```

    <form action="@{Login.authenticate()}" method="post"
class="form-horizontal well">

```

```

      <div class="control-group">
        <label class="control-
label">&{'secure.username'}</label>

```

```

          <div class="controls">
            <input type="text" name="login" value="$
{flash.login}" required="required"/>
          </div>
        </div>

```

```

      <div class="control-group">
        <label class="control-
label">&{'secure.password'}</label>

```

```

          <div class="controls">
            <input type="password" name="password"
value=""/>
          </div>
        </div>

```

```

      <div class="control-group">
        <div class="controls">
          <label for="remember" class="checkbox">
            &{'secure.remember'}

```

```

            <input type="checkbox" id="remember"
name="remember"

```

```

                                value="true" $
{flash.remember ? 'checked="true"' : ''} />
                                </label>

                                <pclass="help-block">Включите, если хотите,
чтобы система узнавала Вас при каждом
                                входе</p>
                                </div>
                                </div>
                                <div class="form-actions">
                                    <input type="submit" value="{secure.signin}"
class="btn btn-primary"/>
                                    <input type="button" value="Напомнить пароль"
class="btn"/>
                                </div>
                                </form>
                                </div>
                                <div class="span6">
                                    <h2>Регистрация</h2>

                                    <form action="@{Login.registrate()}" method="post"
class="form-horizontal well">
                                        <div class="control-group">
                                            <label class="control-label">Ваше имя:</label>

                                            <div class="controls">
                                                <input type="text" name="name" value="$
{flash.name}" required="required"/>
                                            </div>
                                        </div>
                                        <div class="control-group">
                                            <label class="control-label">Ваша
фамилия:</label>

                                            <div class="controls">
                                                <input type="text" name="name" value="$
{flash.surname}" required="required"/>
                                            </div>
                                        </div>
                                        <div class="control-group">
                                            <label class="control-label">Логин:</label>

                                            <div class="controls">
                                                <input type="text" name="login" value="$
{flash.login}" required="required"/>
                                            </div>
                                        </div>
                                        <p>Допустимые символы: буквы латинского
алфавита и цифры</p>
                                        </div>
                                    </div>
                                    <div class="control-group">
                                        <label class="control-label">Пароль:</label>

                                        <div class="controls">
                                            <input type="password" name="password"
value="" required="required"/>
                                        </div>
                                    </div>

```

```

        <div class="control-group">
            <label class="control-label">Подтверждение:</label>

            <div class="controls">
                <input type="password" name="confirm"
value="" required="required"/>
            </div>
        </div>
        <div class="control-group">
            <label class="control-label">Email:</label>

            <div class="controls">
                <input type="email" name="email" value="{flash.email}"/>
            </div>
        </div>

        <div class="form-actions">
            <input type="submit" value="Зарегистрироваться"
class="btn btn-primary"/>
        </div>
    </form>
</div>
</div>

```

/views/Messages/all.html

```

#{extends 'main.html' /}
#{set title:'Сообщения' /}

```

```

<div class="row-fluid">
    <ul class="nav nav-tabs" id="tabs">
        <li class="active"><a href="{Messages.all()}">Все
диалоги</a></li>
        <li class="active"><a href="#">Диалог с Иван Иванов</a></li>
    </ul>

    <div class="tab-content">
        <div class="tab-pane active">
            <table class="table table-striped table-condensed">
                #{list items:messageList, as:'msg'}
                <tr>
                    <td style="width: 64px">
                        
                    </td>
                    <td>
                        <div>
                            <a href="{Profiles.show(msg.sender.id)}" title="Просмотреть
диалог">
                                {msg.sender.surnameWithName}
                            </a>
                        </div>
                        <div style="opacity: .5; font-size:
80%">

```

```

dd.MM.yyyy") }
                                ${msg.sendDt?.format("HH:mm
                                </div>
                                </td>
                                <td>
                                    <a href="#" title="Просмотреть
диалог">
                                        ${msg.text}
                                        </a>
                                </td>
                                <td style="text-align: right">
                                    <button class="btn">
                                        <i class="icon-remove"></i>
                                        Удалить диалог
                                    </button>
                                </td>
                                </tr>
                                #{/list}
                                </table>
                                </div>

                                </div>
                                </div>

```

/views/Messages/dialog.html

```

#{extends 'main.html' /}
#{set title:"Диалог с " + toUser.surnameWithName /}

<div class="row-fluid">
    <ul class="nav nav-tabs" id="tabs">
        <li class="active"><a href="@{Messages.all()}">Все
диалоги</a></li>
        <li class="active"><a
href="@{Messages.dialog(toUser.id)}">Диалог с $
{toUser.surnameWithName}</a></li>
    </ul>

    <div class="tab-content">
        <div class="tab-pane active">
            <table class="table table-striped table-condensed">
                <thead>
                    <tr>
                        <th>
                            <div style="margin: 0 100px;text-align: left">
                                <div style="display: inline-block; width: 100px; float: left;">
                                    <div>
                                        <ahref="@{Profiles.show(msg.sender.id)}" title="Просмотреть
диалог">

```

```

{msg.sender.surnameWithName}
$
</a>
</div>
<div style="opacity: .
5; font-size: 80%">
$
{msg.sendDt?.format("HH:mm dd.MM.yyyy")}
</div>
</div>
<div>
${msg.text}
</div>
</div>
</td>
</tr>
#{/if}
#{else }
<tr>
<td>

<div style="margin: 0 100px;text-
align: right">
<div style="display:
inline-block;width: 100px;float: right;">
<div>
<a href="@{Profiles.show(msg.sender.id)}" title="Просмотреть
диалог">
$
{msg.sender.surnameWithName}
</a>
</div>
<div style="opacity: .
5; font-size: 80%">
$
{msg.sendDt?.format("HH:mm dd.MM.yyyy")}
</div>
</div>
<div>
${msg.text}
</div>
</div>
</td>
</tr>
#{/else}
#{/list}
<tr>

```

```
<td style="text-align: center;overflow:
hidden;">

<textarea rows="4" style="width:
98%;resize: none;">

</textarea>

<button class="btn btn-primary"
style="float: right;">

<i class="icon-envelope icon-
white"></i>

Отправить

</button>

</td>

</tr>

</table>

</div>

</div>

</div>
```

/views/Profiles/show.html

`{extends 'main.html' /}`

`{set title:user.surnameWithName /}`

```

    #{set 'moreScripts'}

    <script src="http://api-maps.yandex.ru/2.0/?
load=package.full&mode=debug&lang=ru-RU" type="text/javascript"></script>

    <script type="text/javascript">

        var myMap;

        ymaps.ready(function () {

            myMap = new ymaps.Map("YMapsID", {

                // Центр карты

                center:[55.76, 37.64],

                // Коэффициент масштабирования
                zoom:10,
                // Тип карты
                type:"yandex#map"
            }
        );

    });

    function sendRequest(obj) {
        $.post('@{Friends.add(user.id)}',
            function (data, obj) {
                $(obj).html(data);
            })
    }
</script>
#{/set}

#{set 'moreStyles'}
<style type="text/css">
    .rating {
        padding: 5px;
        text-align: center;
    }

    .send-msg {
        padding: 5px;
        text-align: center;
    }

    .last-place {

```

```

        padding-left: 30px;
    }

    .contact {
        padding-left: 30px;
    }

    .auto {
        padding-left: 30px;
    }

    .auto img {
        float: left;
        height: 64px;
        margin-right: 10px;
    }
</style>
#{/set}

<ul class="breadcrumb">
#{if !currentUser.equals(user)}
    <li style="float: right;">
        <a href="#" onclick="sendRequest(this);return
false;">Добавить в друзья</a>
    </li>
#{/if}
    <li><a href="/">Главная</a><span
class="divider">&raquo;</span></li>
    <li class="active">${user.surnameWithName}</li>
</ul>

<div class="row-fluid">
    <div class="span4">
        

        <div class="rating">
            <span class="title">Рейтинг</span>
            
            
            
            
            
        </div>
        #{if !currentUser.equals(user)}
            <div class="send-msg">
                <a href="@{Messages.dialog(user.id)}">Написать
сообщение</a>
            </div>
        #{/if}
    </div>
    <div class="span4">
        <div class="surname-with-name">
            <h2>${user.surnameWithName}</h2>
        </div>
        <div class="birthday">
            ${user.birthDate?.format("dd.MM.yyyy")}
        </div>
        <h3>Последнее местоположение</h3>
    </div>
</div>

```

```
<div class="last-place">  
    <a href="#">Россия, Пензенская область, с.Саполга</a>  
</div>  
<br/>  
  
<h3>Контакты</h3>  
  
<div class="contact">  
    Моб. тел. +7 927 649 85 01  
</div>  
<div class="contact">  
    E-mail <a href="#">romeo007.06@gmail.com</a>  
</div>  
<div class="contact">  
    Вконтакте<a href="#">КривошеинАнтон</a>  
</div>  
  
<h3>Автомобили</h3>  
  
<div class="auto">  
      
  
    <div>AUDI 100</div>  
    <div>Цвет белый</div>  
    <div>Гос. номер а001аа58</div>  
</div>  
</div>  
<div class="span4">  
    <div id="YMapsID" style="width:370px;height:370px"></div>  
</div>  
</div>
```

/view/Routes/all.html

```
{extends 'main.html' /}  
{set title:'Маршруты' /}
```

{set 'moreNav'}

```
<ul class="breadcrumb">  
    <li style="float: right;">  
        <a href="@{Routes.add()}">Новый маршрут</a>  
    </li>  
    <li>  
        <a href="/">Главная</a><span class="divider">&raquo;</span>  
    </li>  
    <li class="active">  
        Маршруты  
    </li>  
</ul>
```

{/set}

```
<div class="row-fluid">  
    <div class="accordion" id="accordion2">  
        <div class="accordion-group" data-toggle="collapse">  
            <div class="accordion-heading">  
                <a class="accordion-toggle" data-  
toggle="collapse" data-parent="#accordion2" href="#collapseOne">  
                    <h3>Поиск маршрута</h3>
```

```

        </a>
    </div>
    <div id="collapseOne" class="accordion-body collapse"
style="height: 0px; ">
        <div class="accordion-inner">
            <form action="" method="post">

                </form>

            </div>
        </div>
    </div>

</div>

<table class="table table-striped table-bordered table-
condensed">
    <thead>
    <tr>
        <th>#</th>
        <th>Маршрут</th>
        <th>Место и время отправления</th>
        <th>Место и время прибытия</th>
        <th>Транзитные пункты</th>
        <th>Автор маршрута</th>
        <th>Автомобиль</th>
        <th>Дата регистрации маршрута</th>
    </tr>
    </thead>

    #{list items:routeList, as:'route'}
    <tr>
        <td>#{route?.id}</td>
        <td>
            <div class="title">
                <a href="@{Routes.show(route.id)}"
title="Просмотреть">#{route?.title}</a>
            </div>
            <div class="description">
                #{route?.description}
            </div>
        </td>
        <td>
            <div class="point-title">
                #{if route?.departurePoint}
                #{route?.departurePoint?.title}
                #{/if}
                #{else }
                Пункт не указан
                #{/else}
            </div>
            <div class="date">
                #{route?.departureDt.format("HH:mm dd.MM.yyyy")}
            </div>
        </td>
        <td>
            <div class="point-title">
                #{if route?.arrivalPoint}
                #{route?.arrivalPoint?.title}

```

```

        #{{/if}}
        #{{else }}
            Пункт не указан
        #{{/else}}
    </div>
    <div class="date">
        {{route?.arrivalDt.format("HH:mm dd.MM.yyyy")}}
    </div>
</td>
<td>
    {{list items:route.transitPoints, as:'point'}}
    <div class="point-title">
        {{point.title}}
    </div>
    #{{/list}}
    #{{else}}
        Нет транзитных пунктов
    #{{/else}}
</td>
<td>
    <div class="author">
        {{route?.author?.surnameWithName}}
    </div>
    <div class="birthdate">
        {{route?.author?.birthDate?.format("dd.MM.yyyy")}}
    </div>
</td>
<td>
    {{if route.vehicle != null}}
        {{route.vehicle?.title}} :: $
        {{route.vehicle.model.markVehicle.title}} {{route.vehicle.model.title}}
    {{/if}}
    #{{else}}
        Не указан
    #{{/else}}
</td>
<td>{{route?.createDt.format("HH:mm dd.MM.yyyy")}}</td>
</tr>
#{{/list}}
</table>
</div>

```

/views/Routes/edit.html

```

{{extends 'main.html' /}}
{{set title:'Маршруты' /}}
{{set 'moreScripts'}}
<script src="http://api-maps.yandex.ru/2.0/?
load=package.full&mode=debug&lang=ru-RU"
type="text/javascript"></script>
<script type="text/javascript">
    var myMap;

    initMap = function () {
        myMap = new ymaps.Map("YMapsID", {
            center:[55.76, 37.64],
            zoom:10,
            type:"yandex#map",

```

```

        behaviors:["default", "scrollZoom"]
    }
    );
    myMap.controls.add("mapTools")
        .add("zoomControl")
        .add("typeSelector");

    //Установка текущего города в поле Город
    document.getElementById('inputCity').value =
ymaps.geolocation.city;
    };

    ymaps.ready(function () {
        initMap()
    });

    setCenter = function (title) {
        var myGeocoder = new ymaps.geocode(title);

        myGeocoder.then(
            function (res) {

myMap.panTo(res.geoObjects.get(0).geometry.getCoordinates());
            }
        );
    };

    makeRoute = function () {
        new ymaps.route([
            'Пенза',
            'Саполга',
            'Москва'
        ], {
            mapStateAutoApply:true
        }).then(function (route) {
            // можно выставить настройки графики маршруту
            route.options.set({ strokeColor:'0000ffff',
opacity:0.9 });
            // добавляем маршрут на карту
            myMap.geoObjects.add(route);
        });
    };

    var myCollection;

    searchCity = function (event, ui) {
        /*alert('event = ' + event + '; ui = ' + ui);*/
    };

    $(document).ready(function () {
        $("#inputCity").autocomplete({
            minLength:2,
            source:['Пенза', 'Саполга'],
            search:function (event, ui) {
                searchCity(event, ui);
            }
        });
    });

```

```

        $('#inputCity').bind('keyup', function () {

            ymaps.geocode($('#inputCity').val(), {results:100,
kind:'locality'}).then(function (res) {
                myCollection = new ymaps.GeoObjectCollection();
                myCollection = res.geoObjects;
                myMap.geoObjects.add(myCollection);
                myMap.setCenter(myCollection[0]);
            });
            return false;
        });
    });

    function showPanelAddRoutePoint() {
        $('.js-btn-show-panel').hide();
        $('.js-add-item').show();
    }

    function addRoutePoint() {
        $('.js-btn-show-panel').show();
        $('.js-add-item').hide();
    }

</script>
#{/set}

#{set 'moreStyles'}
<style type="text/css">
    a.context.edit {
        margin: 8px 0;
    }

    table.schedule {
        width: 490px;
    }

    table.schedule td,
    table.schedule th {
        width: 70px;
        text-align: center;
    }

    .table.schedule tbody tr:hover td,
    .table.schedule tbody tr:hover th {
        background: inherit;
    }
</style>
#{/set}

#{set 'moreNav'}
<ul class="breadcrumb">
    <li class=""><a href="/">Главная</a><span
class="divider">&raquo;</span></li>
    <li class=""><a href="@{Routes.all()}">Маршруты</a><span
class="divider">&raquo;</span></li>
    <li class="active"><a href="@{Routes.show(route.id) }">${
route.title}</a></li>
</ul>

```

```

#{/set}

<div class="row-fluid">
<div class="span6">
<label>Название маршрута</label>
<input type="text" class="input-large span12" value="{route.title}"
required="true"/>

<div class="well">
    <h2>Маршрут следования</h2>

    <div>
        Я хочу составить маршрут

        <div class="btn-group" data-toggle="buttons-radio"
            style="display: inline-block; vertical-align:
middle;">
            <button id="btnRouteOnCity" class="btn active"
onclick="EV.Router.changeType('city');">По городу
            </button>
            <button id="btnRouteOnCountry" class="btn"
onclick="EV.Router.changeType('country');">По стране
            </button>
            <button id="btnRouteOnWorld" class="btn"
onclick="EV.Router.changeType('world');">По миру</button>
        </div>
    </div>
    <br/>

    <div id="tbRouteOnCity">
        <form class="form-horizontal" id="fmSelectCity"
onsubmit="return false;">
            <div class="control-group">
                <label class="control-label">Выберите
город:</label>

                <div class="controls">
                    <input id="inputCity" type="text"
style="vertical-align: middle;" value=""
placeholder="Начните набирать
название"/>
                </div>
            </div>
        </form>
        <table class="table table-striped table-bordered table-
condensed">
            <thead>
                <tr>
                    <th>#</th>
                    <th>Название пункта</th>
                    <th>Время прибытия</th>
                    <th></th>
                </tr>
            </thead>
            <tr>
                <th>1</th>
                <th><a onclick="setCenter('Пенза');return false;"
href="#" class="context"

```

```

                                title="На карте">Пенза</a>
                            </th>
                            <th>13:15</th>
                            <td>
                                <a class="btn btn-mini" href="#"
title="Переместить">
                                    <i class="icon-move"></i>
                                </a>
                                <a class="btn btn-mini" href="#"
title="Редактировать">
                                    <i class="icon-pencil"></i>
                                </a>
                                <a class="btn btn-mini btn-warning" href="#"
title="Удалить пункт">
                                    <i class="icon-minus icon-white"></i>
                                </a>
                            </td>
                        </tr>
                        <tr>
                            <td>2</td>
                            <td><a onclick="setCenter('Саполга');return
false;" href="#" class="context"
                                title="На карте">Саполга</a></td>
                            <td>13:45</td>
                            <td>
                                <a class="btn btn-mini" href="#"
title="Переместить">
                                    <i class="icon-move"></i>
                                </a>
                                <a class="btn btn-mini" href="#"
title="Редактировать">
                                    <i class="icon-pencil"></i>
                                </a>
                                <a class="btn btn-mini btn-warning" href="#"
title="Удалить пункт">
                                    <i class="icon-minus icon-white"></i>
                                </a>
                            </td>
                        </tr>
                        <tr>
                            <td>3</td>
                            <td><a onclick="setCenter('Москва');return
false;" href="#" class="context"
                                title="На карте">Москва</a>
                            </td>
                            <td>14:15</td>
                            <td>
                                <a class="btn btn-mini" href="#"
title="Переместить">
                                    <i class="icon-move"></i>
                                </a>
                                <a class="btn btn-mini" href="#"
title="Редактировать">
                                    <i class="icon-pencil"></i>
                                </a>
                                <a class="btn btn-mini btn-warning" href="#"
title="Удалить пункт">
                                    <i class="icon-minus icon-white"></i>

```

```

        </a>
    </td>
</tr>
</table>
<div style="text-align: right">
    <button class="btn btn-success js-btn-show-panel"
href="#" title="Добавить пункт"
        onclick="showPanelAddRoutePoint();return
false;">
        <i class="icon-plus icon-white"></i>
        Добавить пункт
    </button>
</div>
<form class="js-add-item well well-small" style="display:
none;">
    <div class="span6">
        <label>Название пункта</label>
        <input class="js-new-item-title input-large">
        <label>Тип пункта</label>
        <select>
            <option>Начальный пункт</option>
            <option>Транзитный пункт</option>
            <option>Конечный пункт</option>
        </select>
    </div>
    <div class="span6">
        <label>Дата прибытия</label>
        <input class="js-new-item-date input-large">
    </div>
    <div style="clear: both;"></div>
    <div class="form-actions">
        <button class="btn btn-primary" href="#"
            onclick="addRoutePoint();return
false;">
            <i class="icon-plus"></i>
            Добавить
        </button>
    </div>
</form>
</div>
<div id="tbRouteOnCountry" style="display: none;">
    <table class="table table-striped table-bordered table-
condensed">
        <thead>
            <tr>
                <th>#</th>
                <th>Название пункта</th>
                <th>Время прибытия</th>
                <th></th>
            </tr>
        </thead>
        <tr>
            <th>1</th>
            <th><a onclick="setCenter('Пенза');return false;"
href="#" class="context"
                title="На карте">Пенза</a>
            </th>
            <th>13:15</th>

```

```

<td>
    <a class="btn btn-mini" href="#"
title="Переместить">
        <i class="icon-move"></i>
    </a>
    <a class="btn btn-mini" href="#"
title="Редактировать">
        <i class="icon-pencil"></i>
    </a>
    <a class="btn btn-mini btn-warning" href="#"
title="Удалить пункт">
        <i class="icon-minus icon-white"></i>
    </a>
</td>
</tr>
<tr>
    <td>2</td>
    <td><a onclick="setCenter('Саполга');return
false;" href="#" class="context"
        title="На карте">Саполга</a></td>
    <td>13:45</td>
    <td>
        <a class="btn btn-mini" href="#"
title="Переместить">
            <i class="icon-move"></i>
        </a>
        <a class="btn btn-mini" href="#"
title="Редактировать">
            <i class="icon-pencil"></i>
        </a>
        <a class="btn btn-mini btn-warning" href="#"
title="Удалить пункт">
            <i class="icon-minus icon-white"></i>
        </a>
    </td>
</tr>
<tr>
    <td>3</td>
    <td><a onclick="setCenter('Москва');return
false;" href="#" class="context"
        title="На карте">Москва</a>
    </td>
    <td>14:15</td>
    <td>
        <a class="btn btn-mini" href="#"
title="Переместить">
            <i class="icon-move"></i>
        </a>
        <a class="btn btn-mini" href="#"
title="Редактировать">
            <i class="icon-pencil"></i>
        </a>
        <a class="btn btn-mini btn-warning" href="#"
title="Удалить пункт">
            <i class="icon-minus icon-white"></i>
        </a>
    </td>
</tr>
</tr>

```

```

        </table>
        <div style="text-align: right">
            <a class="btn btn-success" href="#" title="Добавить
пункт">
                <i class="icon-plus icon-white"></i>
                Добавить пункт
            </a>
        </div>
    </div>
    <div id="tbRouteOnWorld" style="display: none;">

    </div>

</div>

<div class="well">
    <a style="float: right;" href="/" title="Редактировать"
class="context edit">Редактировать</a>

    <h2>График поездок</h2>

    <table class="schedule table table-striped table-bordered table-
condensed">
        <tr>
            <th>Пн</th>
            <th>Вт</th>
            <th>Ср</th>
            <th>Чт</th>
            <th>Пт</th>
            <th>Сб</th>
            <th>Вс</th>
        </tr>
        <tr>
            <td>14:00</td>
            <td>--</td>
            <td>7:00</td>
            <td>9:00</td>
            <td>--</td>
            <td>9:00</td>
            <td>4:00</td>
        </tr>
    </table>
</div>
<div class="well">
    <a style="float: right;" href="/" title="Редактировать"
class="context edit">Редактировать</a>

    <h2>Свободные места</h2>

    <table class="table table-striped table-bordered table-
condensed">
        <tr>
            <td>Всего мест</td>
            <td>4</td>
            <td>Осталось свободных</td>
            <td>2</td>
        </tr>
    </table>

```

```

<div style="text-align: right">
    <a class="btn btn-success" href="#" title="Занять место">
        <i class="icon-ok icon-white"></i>
        Занять место
    </a>
</div>
</div>
<div class="well">
    <a style="float: right;" href="/" title="Редактировать"
class="context edit">Редактировать</a>

    <h2>Информация о водителе и автомобиле</h2>

    <div class="span5">
        <h3>Водитель</h3>
        <table class="table table-striped table-bordered table-
condensed">
            <tr>
                <td>ФИО</td>
                <td>КривошеинАнтон</td>
            </tr>
            <tr>
                <td>Год рождения</td>
                <td>1989</td>
            </tr>
            <tr>
                <td>Телефон</td>
                <td>+7 (927) 649-85-01</td>
            </tr>
        </table>
    </div>
    <div class="span5">
        <h3>Автомобиль</h3>
        <table class="table table-striped table-bordered table-
condensed">
            <tr>
                <td>Марка и модель</td>
                <td>AUDI 100</td>
            </tr>
            <tr>
                <td>Гос. номер</td>
                <td>a777иа58</td>
            </tr>
            <tr>
                <td>Цвет</td>
                <td>Белый</td>
            </tr>
        </table>
    </div>
</div>
</div>
<div class="span6" style="position: relative; ">
    <div id="YMapsID" style="width:570px;height:400px;position:
fixed;"></div>
</div>
</div>

```

ВЕДОМОСТЬ ДИПЛОМНОГО ПРОЕКТА

Приложение Б

(обязательное)

№ строки	Формат	Обозначение	Наименование	Кол-во листов	Примечание
1					
2			<u>Документация текстовая</u>		
3					
4	A4		Пояснительная записка	189	
5					
6			<u>Документация графическая</u>		
7					
8	A1		Бизнес-процесс поиска маршрута и	1	
9			участия в нем. Диаграмма деятельности		
10					
11	A1		Бизнес-процесс добавления маршрута.	1	
12			Диаграмма деятельности		
13					
14	A1		Бизнес-процесс авторизации	1	
15			пользователя в системе.		
16			Диаграмма деятельности		
17					
18	A1		Взаимодействие неавторизованного	1	
19			пользователя с системой. Диаграмма		
20			вариантов использования		
21					
22	A1		Взаимодействие авторизованного	1	
23			пользователя с системой. Диаграмма		
24			вариантов использования		
25					
26	A1		Архитектура сервиса. Диаграмма	1	
27			Развертывания		
28					
29	A1		Классы приложения для доступа	1	
30			к базе данных. Диаграмма классов		

ГРАФИЧЕСКАЯ ЧАСТЬ

Приложение В

(обязательное)

ПРЕЗЕНТАЦИЯ

Приложение Г
(обязательное)



Рисунок 4.1 Титульный лист

Анализ предметной области

ЦЕЛЬ:

проведение предпроектного анализа существующих онлайн сервисов совместного поиска попутчиков и разработка онлайн системы с учетом исправления недостатков в имеющихся системах

ЗАДАЧИ:

разработка базы данных онлайн сервиса
разработка логической модели базы данных
разработка физической модели базы данных
разработка программного обеспечения
разработка и анализ пользовательского интерфейса
разработка программного обеспечения клиентской части
разработка программного обеспечения серверной части

Рисунок 4.2 Анализ предметной области

Технические характеристики

ФУНКЦИИ СИСТЕМЫ:

- регистрация в системе
- создание маршрутов поездок
- управление своими маршрутами
- составление заявок на поездки
- управление своими заявками
- поиск необходимой информации по сайту
- взаимодействие с другими пользователями системы

Рисунок 4.3 Технические характеристики

Диаграмма развёртывания

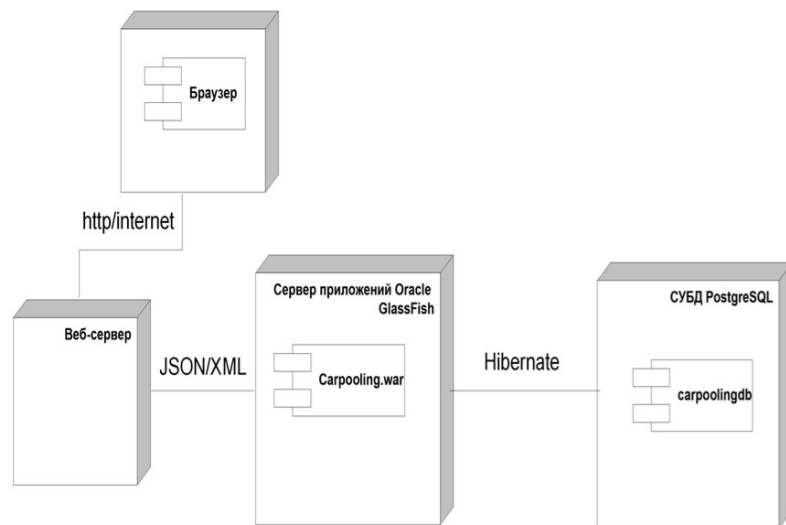


Рисунок 4.4 Диаграмма развёртывания

Диаграмма использования

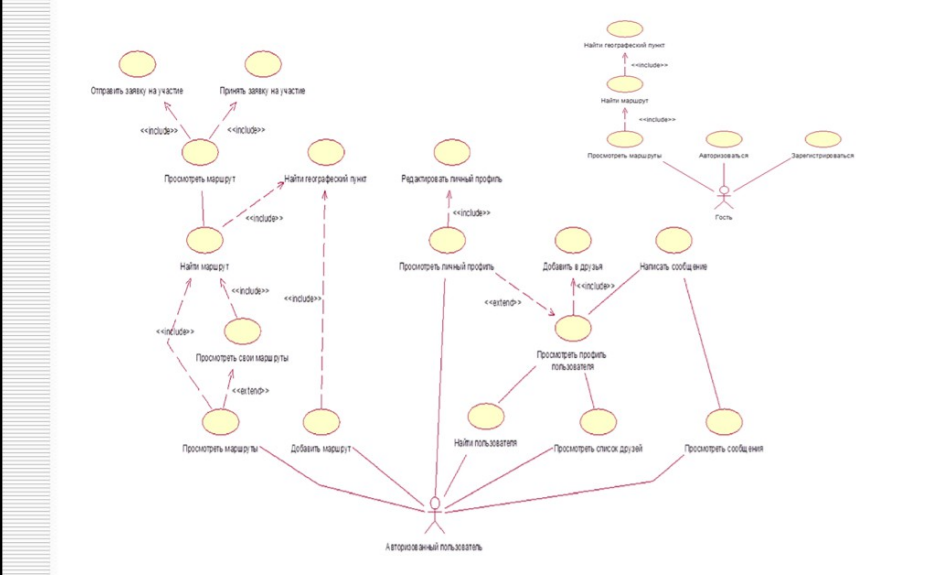


Рисунок 4.5 Диаграмма использования

Логическая модель базы данных

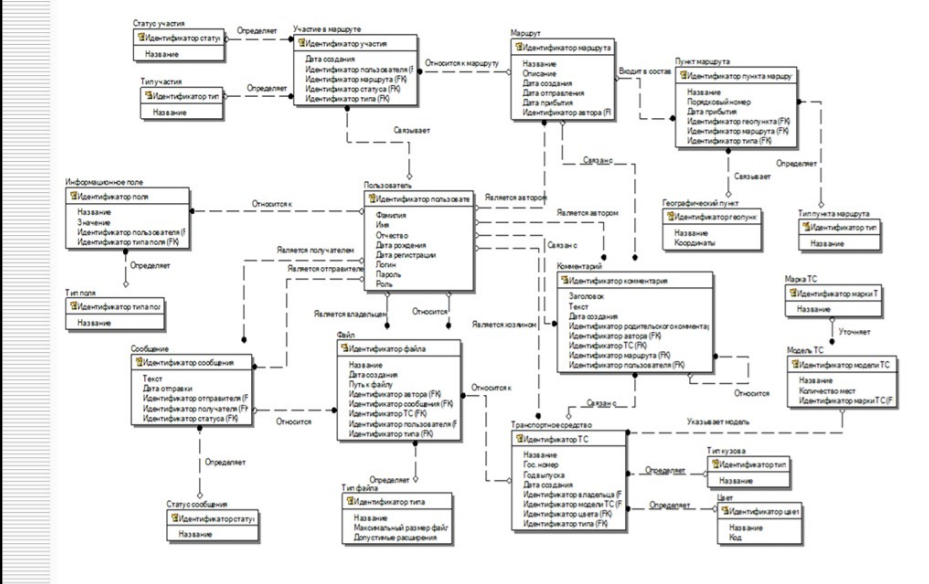


Рисунок 4.6 Логическая модель базы данных

Физическая модель базы данных

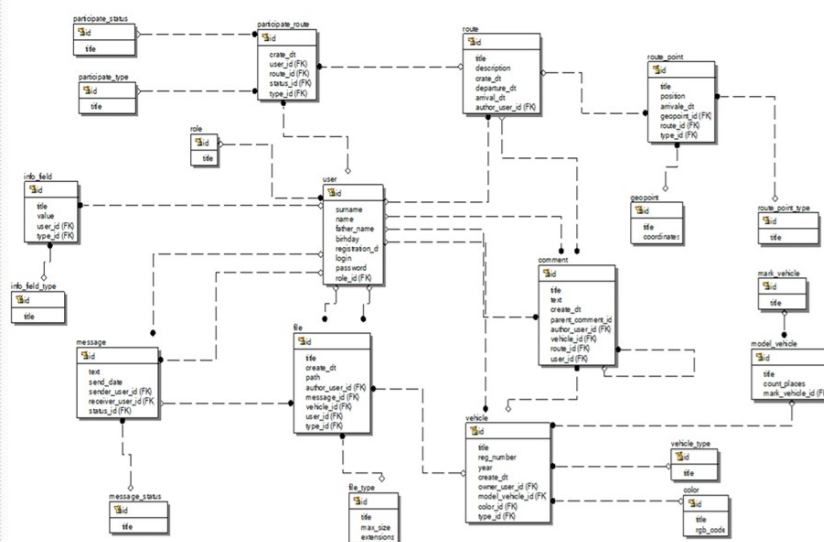


Рисунок 4.7 Физическая модель базы данных

Диаграмма деятельности авторизации пользователя в системе

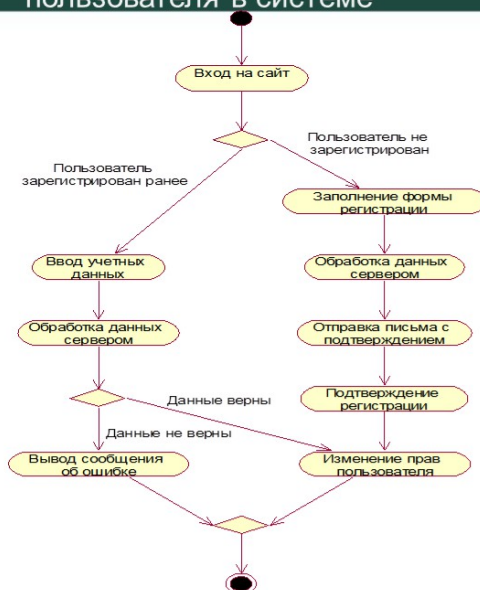


Рисунок 4.8 Диаграмма деятельности авторизации пользователя в системе

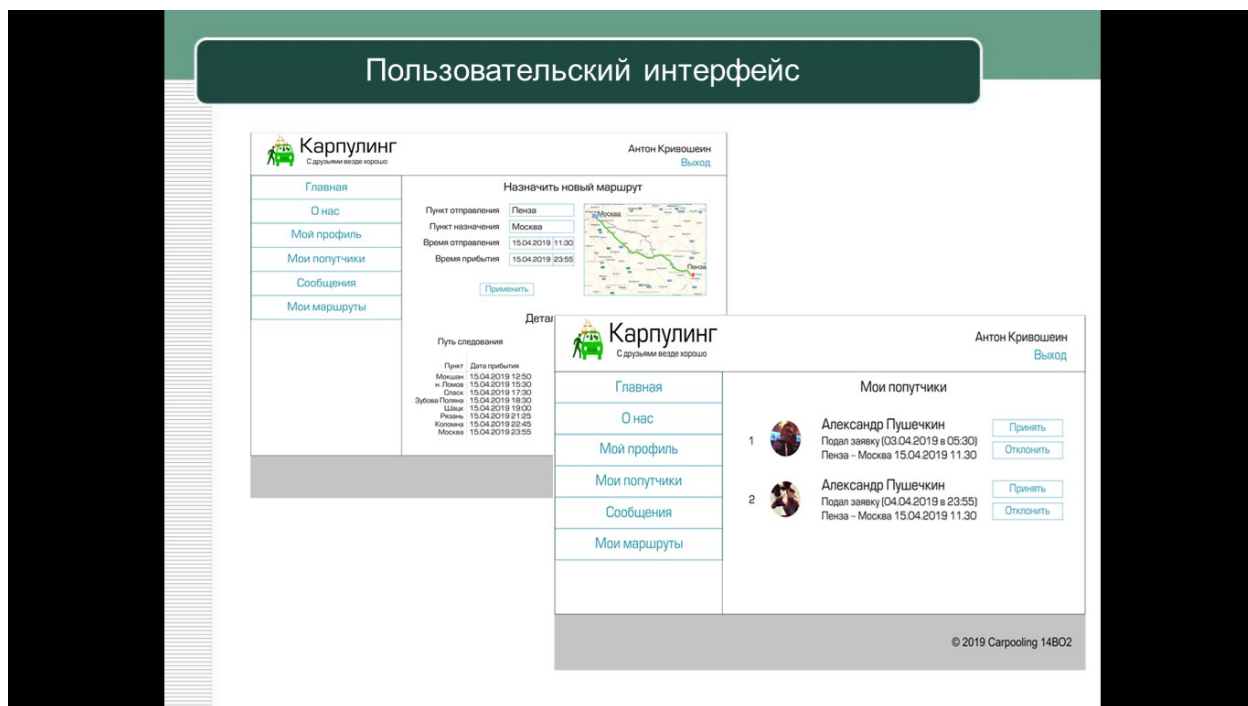


Рисунок 4.11 Пользовательский интерфейс

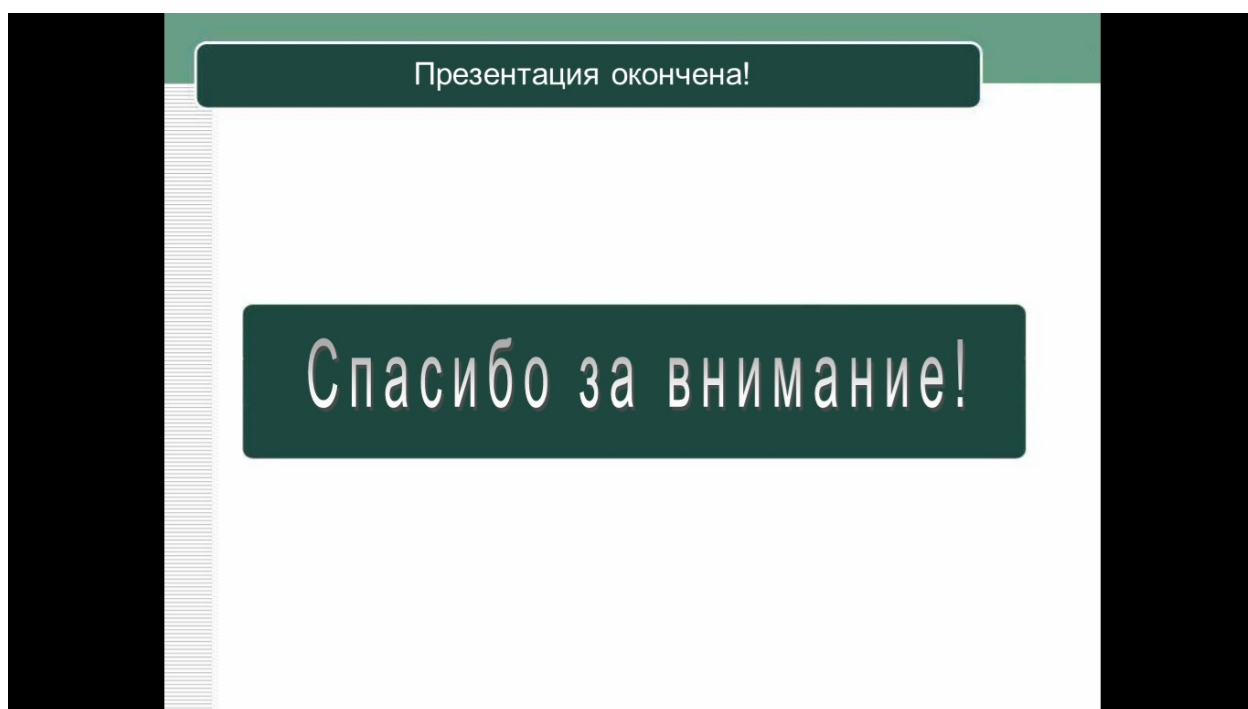


Рисунок 4.12 Завершающий слайд