

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

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

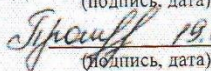
Информационная система автошколы

Студент


(подпись, дата)

Лялин Никита Сергеевич
(ФИО полностью)

Руководитель


(подпись, дата)

Прошкина Е.Н.
(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.
(фамилия, инициалы)

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

Заведующий кафедрой

(подпись)

Макарычев П.П.
(фамилия, инициалы)

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

Секретарь ГЭК

(подпись)

Попова Н.А.
(фамилия, инициалы)

Пенза, 2020

«Утверждаю»
Заведующий кафедрой
_____ П.П. Макарычев «__» _____ 2020 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

1. Студент Лялин Никита Срегеевич гр. 16ВП1 факультета ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы Информационная система автошколы

Тема утверждена приказом ПГУ № 1488/о от " 30 " _____ декабря _____ 2019

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Прошкина Е.Н.

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

1) *Разработать информационную системы автошколы в виде web-приложения*

2) *Среда выполнения:*

Языковые средства разработки: PHP5.6, HTML, CSS, JS

Инструментальные средства разработки: PhpStorm 2020.1, фреймворк Yii2

Технологии разработки: RUP, ООП

3) *Основные функции информационной системы:*

а) регистрация и авторизация пользователей;

б) предоставление возможности изучения теории правил дорожного движения;

в) предоставление возможность записи на практический урок вождения;

г) предоставление возможность добавления водителей и данных о них;

д) предоставление возможность просмотра статистики пройденных тестов по экзаменационным билетам правил дорожного движения

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	15.04.2020 – 20.04.2020	<i>Громов</i>
2	Проектирование	25%	20.04.2020 – 30.04.2020	<i>Громов</i>
3	Реализация	30%	30.04.2020 – 10.05.2020	<i>Громов</i>
4	Контроль качества программного обеспечения	5%	19.05.2020 – 22.05.2020	<i>Громов</i>
5	Оформление пояснительной записки	10%	22.05.2020 – 26.06.2020	<i>Громов</i>

Дата выдачи задания "30" декабря 2019

Руководитель бакалаврской работы
(подпись, дата)

Громов 19.06.2020

Задание к исполнению принял студент
(подпись, дата)

Громов 19.06.2020

Работу к защите допустить
Декан факультета
(подпись, дата)

Реферат

Пояснительная записка содержит 81 лист, 29 рисунков, 10 таблиц, 16 использованный источник, 5 приложений.

ИНФОРМАЦИОННАЯ СИСТЕМА, АВТОШКОЛА, ПДД, БАЗА ДАННЫХ, MVC, ФРЕЙМВОРК Yii2, PHP5.6, UML 2.0, PHPSTORM.

Объектом разработки является программные средства автоматизации процесса деятельности автошколы.

Цель работы – разработка информационной системы автошколы.

В результате проведенной работы создано информационная система автошколы в виде web-приложения, автоматизирующее основные виды деятельности автошколы.

Разработка проводилась на языке программирования PHP5.6, в качестве сред разработки использовались среда программирования PhpStorm 2020.1 и фреймворк Yii2.

					ПГУ 09.03.04 – 08БР161.19 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Информационная система автошколы. Пояснительная записка	Лит.	Лист	Листов
Разраб.	Лялин Н.С.			19.06.2020			4	81
Пров.	Прошкина Е.Н.			19.06.2020				
Реценз.								
Н. контр.	Такташкин Д.В.							
Утв.								
						Гр. 16ВП1		

Содержание

Введение	7
1 Анализ предметной области и требований	9
1.1 Анализ предметной области и постановка задач	9
1.2 Анализ аналогичных программных продуктов	11
1.3 Анализ требований	15
1.3.1 Функциональные требования	16
1.3.2 Нефункциональные требования	18
2 Проектирование информационной системы автошколы	19
2.1 Базовая архитектура системы	19
2.2 Выбор паттерна проектирования	20
2.3 Логическая модель базы данных	22
2.4 Пользовательский интерфейс информационной системы	24
2.5 Проектирование программных средств	28
2.6 Планирование проекта и оценка бюджета	30
3 Реализация информационной системы автошколы	42
3.1 Физическая модель базы данных	42
3.2 Модель реализации клиентского приложения	43
4 Контроль качества программного обеспечения	46
4.1 Функциональное тестирование	46
4.2 Метрики кода	48
Заключение	53
Приложение А. Глоссарий	56
Приложение Б. Листинг программы	58
Приложение В. Графические модели паттернов управления	66

Приложение Г. Адаптивность устройства под мобильные устройства	69
Приложение Д. Результаты функционального тестирования	73

Введение

В настоящее время, область информационных технологий является одной из наиболее развивающихся во всем современном мире. В России, как и во многих других странах, проводятся множество различных программ по развитию данной области. Так в 2017 году был издан указ президентом Российской Федерации «О Стратегии развития информационного общества в Российской Федерации на 2017–2030 годы», который пришел на смену такому же указу от 2008 года. Указ предусматривает широкомасштабное применение сервисов сети Интернет, мобильных устройств и приложений [1]. Также в 2017 году была принята программа «Цифровая экономика Российской Федерации». На основе этой программы в 2019 году была сформирована национальная программа «Цифровая экономика Российской Федерации» [2]. Основой этой программы является развитие цифровой экономики для повышения конкурентоспособности страны.

Благодаря непрерывному увеличению числа владельцев смартфонов, планшетов, персональных компьютеров и развитию интернет-инфраструктуры, во всем мире начали развиваться всевозможные интернет-сервисы и приложения. Каждый день эти сервисы помогают человеку в различных сферах его жизни для обмена сообщениями, для работы с документами, для хранения файлов на облачных сервисах и постоянного доступа к ним, для совместной работы, для работы с графикой, аудио и видео, для ведения дел и хранения заметок и многих других дел.

До появления первого компьютера во всех развитых странах, включая Россию, начало развиваться автомобилестроение. Сегодня трудно представить жизнь человека без применения транспорта. Каждый день люди отправляются по своим делам на личном или общественном транспорте. С конца XX-века преобладает тенденция роста числа личного транспорта по сравнению с общественным. Увеличение количества личного транспорта привело к росту потребности в автошколах.

С развитием информационных технологий стало проще и удобнее осуществлять сбор, хранение и обработку информации в цифровом виде. Для этого начали создавать различные информационные системы, направленные на автоматизацию или упрощение обработки данных. Это привело к тому, что бумажной работы стало намного меньше.

Целью данной квалификационной работы является разработка информационной системы автошколы, задачей которой является автоматизация деятельности автошколы, в том числе предоставление информации об автошколе, предоставление личного кабинета для курсантов, в котором те могли бы изучать теоретическую часть правил дорожного движения, проходить тестирование по экзаменационным билетам, записываться на практические занятия. Помимо этого, система должна предоставлять возможность просмотра статистики прохождения тестирования по билетам.

Данная работа является актуальной так как в настоящее время растет потребность в личном автотранспорте, это в свою очередь приводит к тому, люди идут учиться в автошколы, чтобы выучить теорию правил дорожного движения, научиться управлять автомобилем соблюдая все правила дорожного движения. Так как численность желающих получить водительское удостоверение не уменьшается, на российском рынке автошкол открывается все больше автошкол с различными предложениями для потребителей. Разрабатываемая в рамках выпускной квалификационной работы информационная система должно позволить ускорить этот процесс изучения теоретической базы и отследить статистику изучения и решения экзаменационных билетов.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задач

Согласно официальным данным на сегодняшний день, число автошкол в России превысило отметку в 14000 автошкол. На рисунке 1 приведено сравнение количества автошкол в регионах страны. Исходя из этих данных, в тройку лидеров по количеству автошкол входят Свердловская область, Краснодарский край и Московская область. Так же на рисунке показано соотношение частных автошкол к общему количеству в регионе. Можно заметить, что на сегодня частные автошколы с большим перевесом преобладают над государственными.

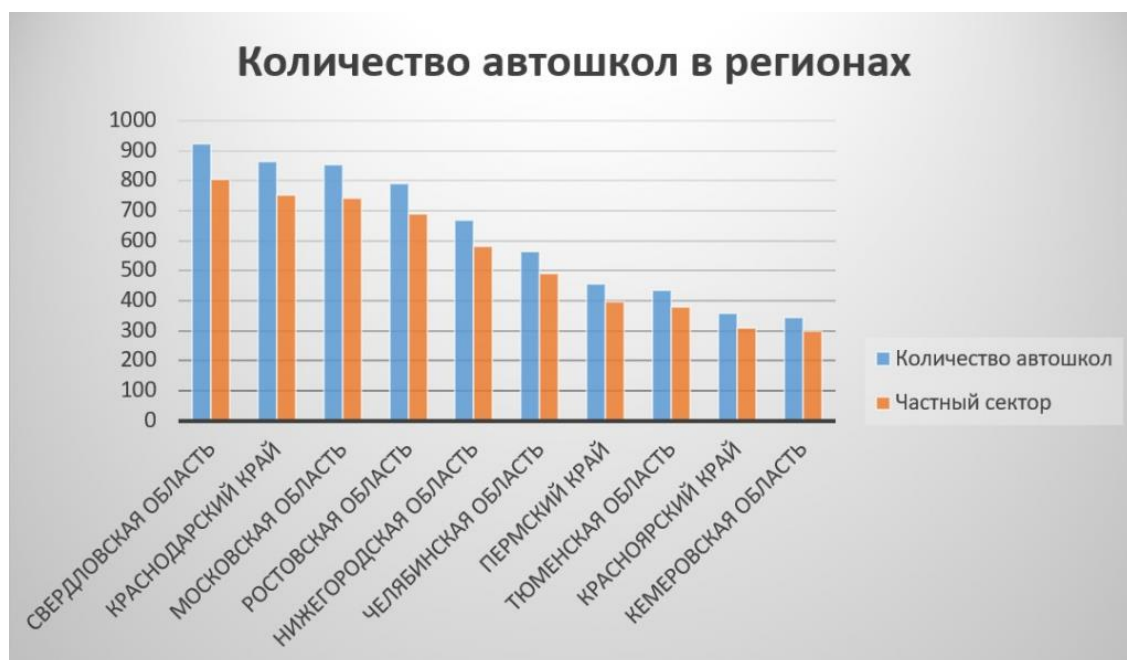


Рисунок 1 – Количество автошкол в регионах России

На рисунке 2 изображена диаграмма сравнения количества автошкол в городах, с наибольшим количеством автошкол. К этим городам относится Москва, Санкт-Петербург, Екатеринбург, Нижний Новгород, Ростов-на-Дону, Краснодар, Челябинск, Новосибирск, Ульяновск.

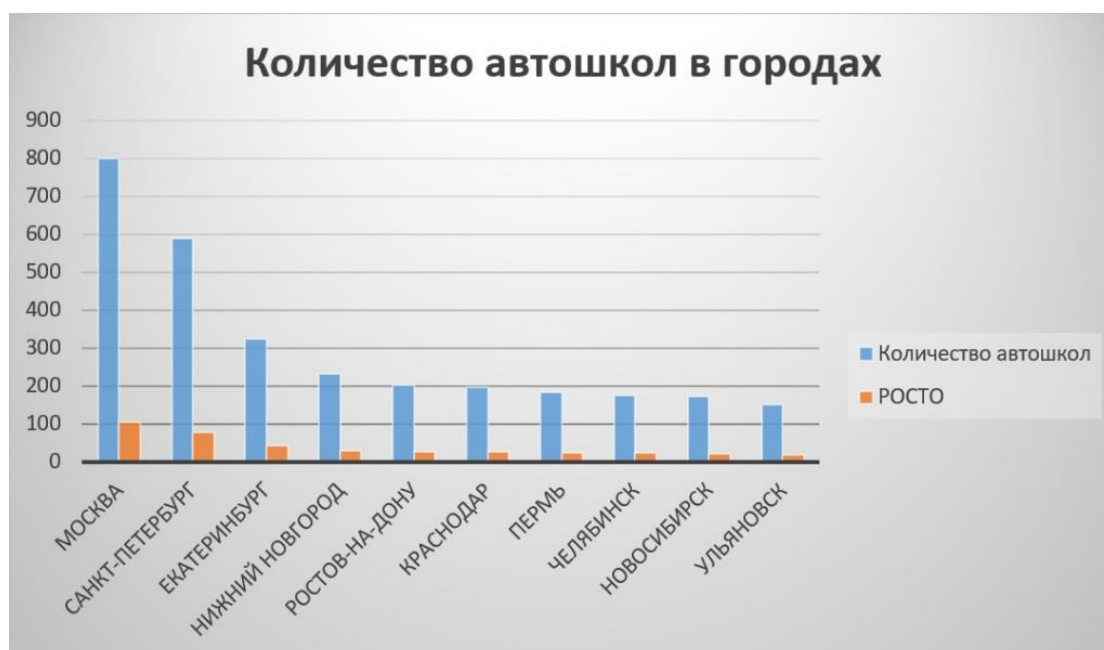


Рисунок 2 – Количество автошкол в городах России

Частный сегмент такого бизнеса наращивает потенциал и не всегда отличное качество услуг, наличие необходимой материальной базы приводит к тому, что человек пройдя недели обучения в таком учреждении, сразу сдаст все экзамены и пополнит ряды автолюбителей.

На данный момент, выбор автошколы начинается с поиска данных об автошколе в сети Интернет, прочтения отзывов о ней. На выбор автошколы влияет наличие веб-страницы автошколы, её внешний вид и информация, приведённая на ней. Так же в мире, где информационные технологии очень развиты, будущим курсантам автошкол, важно наличие web-приложения, в котором будет все необходимое для успешного прохождения обучения и сдачи экзаменов в автошколе. Актуальность тематики заключается в том, что из-за большого количества желающих обучаться вести учет всех обучающихся, следить за их успеваемостью становится тяжелее, а разрабатываемая информационная система должна помочь в решении данной задачи.

Разрабатываемое приложение должно состоять из двух частей – клиентской и администраторской. Приложение должно предоставлять пользователю следующие основные функции:

- просмотр информации об автошколе;

- авторизация пользователя;
- прохождение тестирования по экзаменационным билетам;
- просмотр статистики пройденных тестов;
- просмотр теоретического материала по темам
- запись на практическое занятие.

Администраторская часть приложения должна позволять следующие основные функции:

- редактирование информации об автошколе;
- добавление и удаление курсантов;
- редактирование экзаменационных билетов для тестирования;
- редактирование теоретического материала;
- просмотр статистики прохождения тестов по экзаменационным билетам;
- просмотр статистики пройденных тестов;
- просмотр лекций по темам
- редактирование информации об инструкторах, включая их расписание.

Главным условием является то, что пользователь не должен иметь доступ к функциям администратора.

1.2 Анализ аналогичных программных продуктов

На данный момент существует несколько похожих приложений. Наиболее популярны из них: «Автошкола-контроль», «Автошкола Онлайн», «ИСО Профтех». Они являются наиболее распространенным сервисом для автоматизации работы автошкол.

«Автошкола контроль» – это облачный сервис для автоматизации автошколы. Авторизоваться в личном кабинете автошколы можно как на сайте, так и в специально разработанном мобильном приложении. Сервис

сотрудничает с разными автошколами России и Республики Беларусь. Главный экран приложения изображен на рисунке 3.

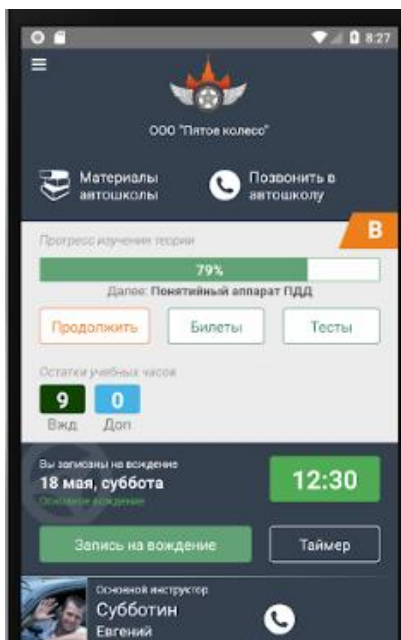


Рисунок 3 – Главный экран личного кабинета

Сервис был запущен в 2016 году в Республике Татарстан, первыми автошколами, которые использовали этот сервис, стали автошколы Федерации Автошкол Республики Татарстан.

Сервис обладает простым и интуитивно понятным пользовательским интерфейсом.

Сервис имеет как достоинства, так и недостатки. Это объясняется тем, что сервис относительно недавно на рынке и является развивающимся продуктом.

К достоинствам можно отнести наличие учета в специально созданной CRM-системе, наличие личного кабинета для курсанта, онлайн-запись на вождение, учет часов вождения.

Из недостатков следует отметить то, что нет возможности просмотреть статистику прохождения экзаменационных билетов, платная СМС-рассылка, а также стоимость внедрения данного сервиса.

Перейдем к рассмотрению следующей аналогичной системы «Автошкола Онлайн».

«Автошкола Онлайн» – это портал дистанционного обучения правилам дорожного движения. Главный экран представлен на рисунке 4.

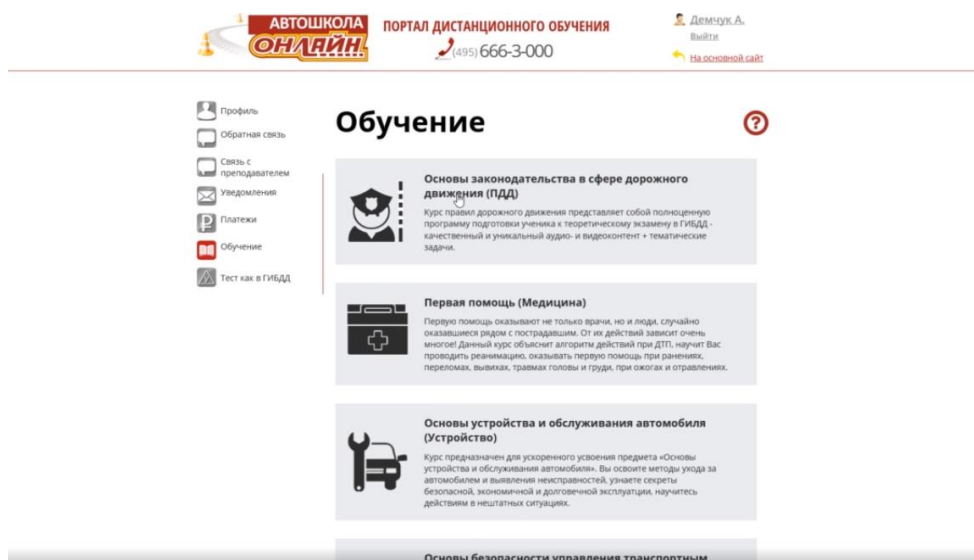


Рисунок 4 – Главный личного кабинета «Автошкола Онлайн»

Портал предоставляет возможность изучения теории правил дорожного движения в формате онлайн. Портал имеет минимально простой пользовательский интерфейс, что является плюсом при прохождении тестирования.

К достоинствам можно отнести следующее:

- 1) наличие личного кабинета;
- 2) возможность изучать теоретический материал;
- 3) проходить тестирование по билетам в форме экзамена;
- 4) наличие обратной связи с автошколой.

Из недостатков выделяется следующее:

- 1) отсутствие возможности записаться на занятие по вождению;
- 2) просмотр статистики прохождения билетов;
- 3) отсутствие возможности прохождения билетов по тематическим блокам.

Портал работает только с автошколами Москвы и Московской области.

Рассмотрим следующую наиболее популярную аналогичную систему «Автошкола-контроль»

Компания «Профтехнология» начинала свой рост на рынке учебных пособий и оборудования для автошкол. С сентября 2016 года начала работать интерактивная система обучения «ИСО Профтех», что позволило избавиться от затрат на покупку пособий и оборудования, поместив все необходимое для изучения теории и сдачи экзаменов в одной информационной системе.

Из достоинств системы можно выделить наличие теоретической части правил дорожного движения и прохождение тестирования по билетам и тематическим блокам, просмотр статистики, наличие мобильного приложения.

Из недостатков, отсутствие обратной связи с автошколой, отсутствие возможности записаться на практическое занятие.

Интерактивная система обучения используется многими автошколами в России.

Общее сравнение аналогов и разрабатываемой информационной системы автошколы приведено в таблице 1.

Таблица 1 – Анализ характеристик аналогов и разрабатываемой системы

Название параметра	«Автошкола-контроль»	«Автошкола Онлайн»	«ИСО Профтех»	Информационная система автошколы
Личный кабинет	Да	Да	Да	Да
Прохождение тестов	Да	Да	Да	Да
Запись на практические занятия	Да	Нет	Нет	Да

Продолжение таблицы 1

Просмотр статистики пройденных тестов	Нет	Нет	Да	Да
Изучение теоретического материала	Нет	Да	Нет	Да
Прохождение тестов по тематическим блокам	Нет	Нет	Да	Да

Исходя из анализа подобных программных продуктов, принято решение в ходе данной работы реализовать собственную информационную систему автошколы, которая не будет иметь вышеперечисленных недостатков, а именно: отсутствие возможности просмотра теоретического материала, просмотр статистики прохождения тестов. Разрабатываемое программное обеспечение информационной системы актуально для частных автошкол, которые хотят получить необходимый функционал, который может поспособствовать повышению спроса данной автошколы с минимальными затратами.

1.3 Анализ требований

Анализ требований является частью процесса разработки программного обеспечения, которая включает в себя сбор требований, выявление взаимосвязей, их систематизацию и документирование [3].

Полнота и качество требований играют важную роль в успехе реализации проекта. Требования к ПО должны быть выполнимыми, тестируемыми и детализованными. Выделяют функциональные и нефункциональные требования. Функциональные требования – список задач, которые должно решать само программное обеспечение. Они определяют действия системы и какова реакция системы на различные входные данные. Нефункциональные требования определяют свойства и характеристики системы, не относящиеся к ее поведению. Они включают производительность, удобство сопровождения, надежность, безопасность и другие критерии [3].

1.3.1 Функциональные требования

В результате анализа предметной области и продуктов-аналогов были выделены следующие основные функциональные требования к системе:

- а) для пользователя-администратора:
 - 1) регистрация пользователя;
 - 2) добавление и изменение теоретической базы ПДД;
 - 3) просмотр статистики пройденных тестов любого пользователя;
 - 4) изменение расписания водителей-инструкторов.
- б) для обычного пользователя:
 - 1) авторизация пользователя;
 - 2) запись на практическое занятие;
 - 3) изучение теоретической базы ПДД;
 - 4) прохождение тестов по вопросам билетов;
 - 5) просмотр статистики пройденных тестов.

Для описания функциональных требований к системе была выбрана диаграмма вариантов использования.

Диаграмма вариантов использования или диаграмма прецедентов служит для описания функциональности и поведения проектируемой системы.

Она отражает отношения между актерами и прецедентами или вариантами использования.

Актер обозначает набор ролей пользователя, взаимодействующего с некоторой сущностью, например, системой, подсистемой или классом. В качестве актера может выступать человек, внешняя сущность, класс или другая система. Актеры не могут быть связаны друг с другом.

Прецедент обозначает выполняемые системой действия, приводящие к наблюдаемым актерами результатам. Имя прецедента связано с непрерываемым сценарием, который является конкретной последовательностью действий, иллюстрирующей поведение.

Включение (include) показывает определенную связь между основным вариантом использования и его специальными возможностями. Отношение включения определяется между двумя вариантами использования и указывает, что поведение, заданное для одного варианта использования, включается как составной фрагмент в последовательность поведения другого варианта использования.

Расширение (extend) определяет связь основного варианта использования с другим, функциональное поведение которого задействуется основным не всегда, а только при выполнении специальных условий.

Несколько актеров могут взаимодействовать с одним и тем же множеством вариантов использования. Эта общность свойств и поведения представляется в виде отношения обобщения с другим, актером, который моделирует соответствующую общность ролей.

Отношение обобщения графически обозначается сплошной линией со стрелкой в форме незакрашенного треугольника, стрелка ведет от дочернего к родительскому варианту использования.

Диаграмма вариантов использования представлена на рисунке 5.

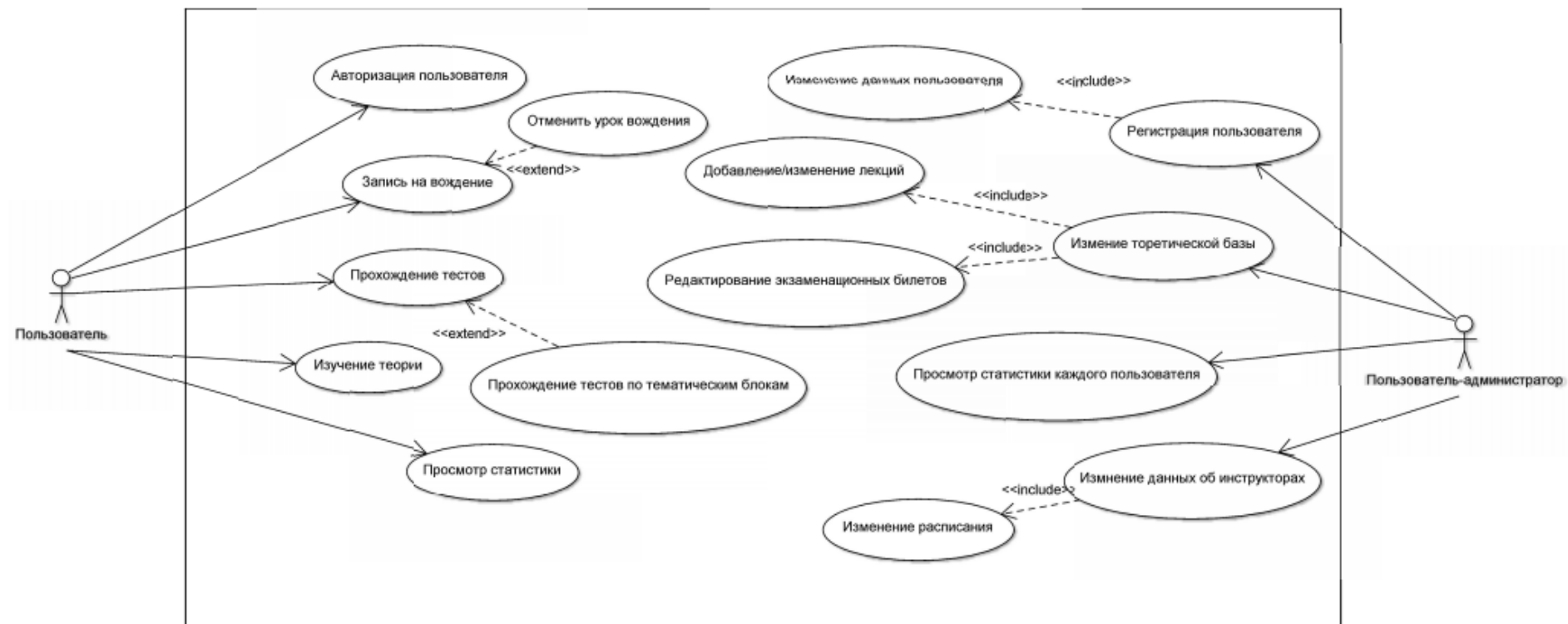


Рисунок 5 – Диаграмма вариантов использования

Диаграмма вариантов использования отображает функциональные требования к системе. На диаграмме видно, какие функции выполняет каждый актер.

Рассмотрим подробнее некоторые прецеденты, для этого приведем их подробную спецификацию. На рисунке 6 изображена спецификация прецедента «Регистрация пользователя».

Прецедент: Регистрация пользователя
ID: 1
Краткое описание: Пользователь-администратор регистрирует нового пользователя
Главные актеры: Пользователь-администратор
Второстепенные актеры: Нет
Предусловия: Пользователь-администратор открыл систему
Основной поток: <ol style="list-style-type: none">1. Пользователь-администратор вводит данные о пользователе2. Пользователь-администратор нажимает кнопку «Зарегистрировать»3. Система проверяет не существует ли такой пользователь в системе4. ЕСЛИ пользователь не существует<ol style="list-style-type: none">4.1. Система генерирует пароль для нового пользователя занося зашифрованный пароль в базу данных4.2. Система отправляет на указанную электронную почту пользователя сообщение о регистрации пользователя с данными для авторизации5. ИНАЧЕ<ol style="list-style-type: none">5.1. Сервер возвращает сообщение об ошибке
Постусловия: Нет
Альтернативные потоки: Нет

Рисунок 6 – Спецификация прецедента «Регистрация пользователя»

Данная спецификация описана для варианта использования «Регистрация пользователя», который предназначен для регистрации нового пользователя в системе и генерации данных для доступа в систему.

На рисунке 7 изображена спецификация прецедента «Авторизация пользователя».

Прецедент: Авторизация пользователя
ID: 2
Краткое описание: Пользователь авторизуется в системе
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: Пользователь был зарегистрирован пользователем-администратором в системе
Основной поток: <ol style="list-style-type: none"> 1. Пользователь вводит данные для доступа 2. Система проверяет наличие пользователя в системе 3. ЕСЛИ пользователь найден <ol style="list-style-type: none"> 3.1. Система сравнивает зашифрованный после введения пароль с зашифрованным паролем в базе данных 3.2. ЕСЛИ пароли совпадают <ol style="list-style-type: none"> 3.2.1. Система перенаправляет пользователя в личный кабинет 3.3. ИНАЧЕ <ol style="list-style-type: none"> 3.3.1. Система вводит сообщение о неправильно введенном пароле 4. ИНАЧЕ <ol style="list-style-type: none"> 4.1. Система выдает сообщение о том, что пользователь не найден в системе
Постусловия: Нет
Альтернативные потоки: нет

Рисунок 7 – Прецедент «Авторизация пользователя»

Данная спецификация описана для варианта использования «Авторизация пользователя», который предназначен для авторизации пользователя в системе.

На рисунке 8 изображена спецификация прецедента «Авторизация пользователя».

Прецедент: Записаться на вождение
ID: 3
Краткое описание: Пользователь записывается на практическое занятие к инструктору
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: Пользователь авторизовался в системе
Основной поток: <ol style="list-style-type: none"> 1. Пользователь переходит на страницу где показан список всех доступных инструкторов 2. Пользователь выбирает инструктора 3. Пользователь выбирает дату и время занятия 4. Система проверяет свободно ли выбранное время для занятия 5. ЕСЛИ занятие свободно <ol style="list-style-type: none"> 5.1. Система заносит пользователя в список занятий на выбранное время 6. ИНАЧЕ <ol style="list-style-type: none"> 6.1. Система возвращает сообщение о том, что занятие на выбранное время уже занято другим пользователем
Постусловия: Нет
Альтернативные потоки: нет

Рисунок 8 – Прецедент «Записаться на вождение»

Данная спецификация описана для варианта использования «Записаться на вождение», который предназначен для добавления авторизованного пользователя в список занятий.

Для варианта использования «Авторизация пользователя» составим диаграмму деятельности (рисунок 9).

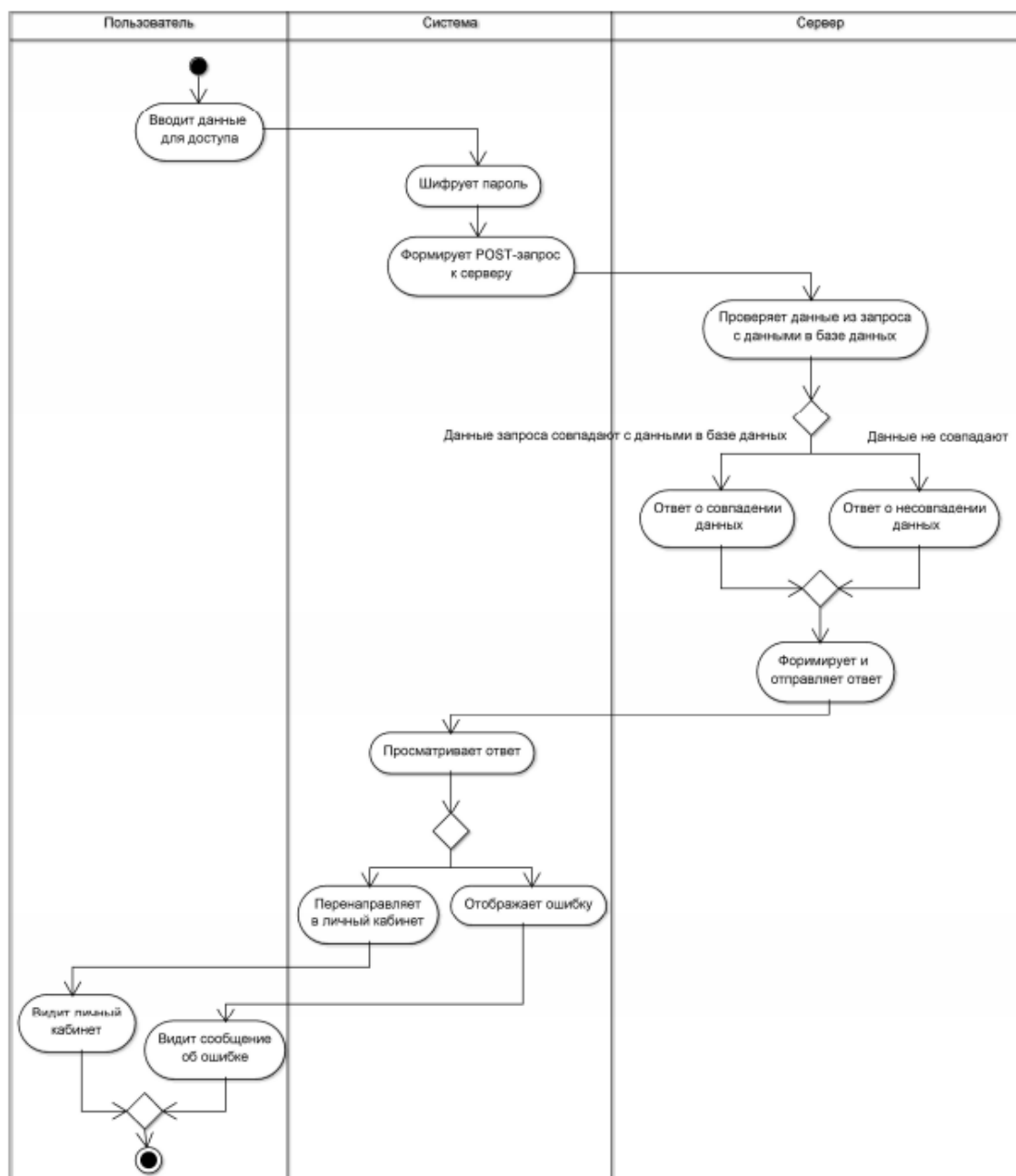


Рисунок 9 – Диаграмма деятельности для прецедента «Авторизация пользователя»

После открытия стартовой страницы системы, пользователь должен ввести свои учетные данные в соответствующие поля ввода. Далее

приложение шифрует пароль, что обеспечивает надежность сохранения и передачи данных, формирует из полученных данных HTTP-запрос и отправляет его на сервер методом POST. Сервер, получив запрос от клиента, сверяет данные полученные из запроса с данными из базы данных. Если соответствие было не найдено, то сервер возвращает клиенту сообщение об ошибке, в противном случае он пользователь переходит в личный кабинет системы.

1.3.2 Нефункциональные требования

Разработку можно разделить на 2 этапа: разработка серверной части и разработка клиентской части. Составим требования для каждой из них.

Требования к клиентской части:

- клиентская часть будет является набором веб-страниц;
- система должна обладать интерфейсом, удобным и понятным для пользователя;
- в системе не должно возникать исключительных ситуаций, приводящих к аварийному завершению работы;
- разработка клиентской части должна осуществляться с использованием языка гипертекстовой разметки HTML5, CSS-стилей и языка программирования JS;
- веб-страницы должны быть адаптированы под мобильные устройства.

Выделим требования к серверной части приложения:

- серверная часть должна быть реализована на языке PHP5.6 в среде PHPStorm с использованием фреймворка Yii2;
- сервер Apache 2.4;
- в качестве базы данных должна использоваться СУБД MySQL.

В результате анализа были составлены нефункциональные требования к клиентской части и серверной части.

2 Проектирование информационной системы автошколы

2.1 Базовая архитектура системы

В настоящее время программные приложения и информационные системы вышли на такой уровень развития, что термин «архитектура» в применении к ним уже давно не удивляет. Правильную, эффективную и надежно функционирующую информационную систему построить не так просто, как может показаться на первый взгляд.

Под архитектурой систем положено понимать совокупность решений относительно:

- организации программной системы;
- выбора структурных элементов, составляющих систему и их интерфейсов;
- объединения этих элементов в подсистемы;
- поведения этих элементов во взаимодействии с другими элементами данной системы;
- архитектурного стиля, который определяет логическую и физическую организацию системы.

Охват архитектуры программной системы также не заканчивается структурными построениями, она также охватывает и правила ее использования и интеграции с другими системами, функциональность, производительность, гибкость, надежность, полноту, технологические ограничения и вопрос пользовательского интерфейса.

Принято выделять следующую классификацию программных систем по их архитектуре:

- централизованная архитектура;
- архитектура «файл-сервер»;
- двухзвенная архитектура «клиент-сервер»;
- многозвенная архитектура «клиент-сервер»;
- архитектура распределенных систем.

Для правильной реализации данного проекта в рамках выпускной квалификационной работы было принято использовать архитектуру «клиент-сервер», так как данная архитектура используется для реализации web-приложений. В таких приложениях как правило серверная часть имеет различные архитектурные решения и специальные возможности, как например:

- а) отсутствует необходимость устанавливать и использовать дополнительное ПО на стороне клиента;
- б) используя единое место хранения данных обеспечиваются минимальные требования для поддержания целостности данных;
- с) работоспособность сервера и каналов связи определяют доступность и недоступность;
- д) архитектура web-систем не имеет существенных ограничений и другие.

Всё это дает возможность довольно быстро и просто изменять приложение.

2.2 Выбор паттерна проектирования

Рассмотрим паттерны, предназначенные для проектирования Web-приложения. К ним относятся:

- модель-представление-контроллер (Model View Controller);
- контроллер страниц (Page Controller);
- преобразователь (Transform View);
- представление по шаблону (Template View);
- контроллер запросов (Front Controller);
- контроллер приложения (Application Controller);
- двухшаговая шаблонизация (Two Step View).

Описание основных паттернов проектирования web-приложений приводятся в таблице 2.

Таблица 2 – Web-паттерны представления данных

№ п/п	Название	Описание
1.	Модель-представление-контроллер (Model View Controller)	Представление отображает содержимое модели средствами графического интерфейса. Контроллер получает входные данные от пользователя, выполняет операции над моделью и указывает представлению на необходимость соответствующего обновления. В этом плане графический интерфейс можно рассматривать как совокупность представления и контроллера (рисунок В.1).
2	Контроллер страниц (Page Controller)	В основе контроллера страниц лежит идея создания компонентов, которые будут выполнять роль контроллеров для каждой страницы Веб-сайта. На практике количество контроллеров не всегда в точности соответствует количеству страниц, поскольку иногда при щелчке на ссылке открываются страницы с разным динамическим содержимым (рисунок В.2).
3	Контроллер запросов (Front Controller)	Контроллер запросов обрабатывает все запросы, поступающие к Веб-сайту, состоит из двух частей: Веб-обработчика и иерархии команд. Веб-обработчик – это объект, выполняющий фактическое получение POST- или GET-запросов, поступивших на Веб-сервер. Веб-обработчик обычно реализуется в виде класса, поскольку он не генерирует никаких откликов. Команды также являются классами; более того, им не нужно знать о наличии Веб-окружения, несмотря на то, что им часто передается информация из HTTP-запросов. В большинстве случаев Веб-обработчик – это довольно простая программа, функции которой заключаются в выборе нужной команды (рисунок В.3).
4	Представление по шаблону (Template View)	Основная идея, лежащая в основе типового решения представление по шаблону, – вставка маркеров в текст готовой статической HTML-страницы. При вызове страницы для обслуживания запроса эти маркеры будут заменены результатами некоторых вычислений (например, результатами выполнения запросов к базе данных). Подобная схема позволяет создавать статическую часть страницы с помощью обычных средств, например, текстовых редакторов, работающих по принципу WYSIWYG, и не требует знания языков программирования. Для получения динамической информации маркеры обращаются к отдельным программам (рисунок Г.4).
5	Преобразователь (Transform View)	Преобразует записи в HTML по одной. Когда выполняются запросы к БД, вы получаете данные, но этого недостаточно, чтобы отобразить нормальную web-страницу. Использование Transform View подразумевает преобразование, когда на входе есть модель, а на выходе HTML.

Для реализации проекта в рамках выпускной квалификационной работы был выбран MVC паттерн.

MVC(Model-View-Controller) – паттерн, разделяющий логику приложения на три части:

- model – часть отвечающая за работу с базой данных и различные вычисления;
- view – набор web-страниц(интерфейс), отображающих полученные данные от модели;
- controller – обработчик запросов от пользователя, полученные данные передает в модель.

Таким образом, при использовании данного шаблона можно писать независимые друг от друга блоки кода, при изменении которых не будет нарушена работа остальной части приложения.

При таком разделении компонентов приложения, соблюдается концепция разделения ответственности. При соблюдении такой концепции намного проще проводить работу над отдельными компонентами.

Выбор этого паттерна объясняется тем, что паттерн интуитивно понятен, прост в реализации, создан для создания гибких и масштабируемых приложений, которые можно будет в дальнейшем легко сопровождать и дополнять функционал приложения.

2.3 Логическая модель базы данных

Логическая модель показывает логическую структуры предметной области конкретной задачи. Она строится на основе концептуальной модели с учетом выбранной модели данных, при этом не учитывая особенности целевой СУБД.

Для устранения избыточности модели производится нормализация – процесс преобразования базы данных в вид, отвечающий нормальным формам. Таким образом происходит структурирование отношений так, чтобы

в каждом отношении хранились только первичные факты. Итогом нормализации будет являться модель базы данных с уменьшенным потенциальным несоответствием хранимой в базе информации.

Для нормализации базы данных необходимо и достаточно соблюдения трех нормальных форм:

- 1) первая нормальная форма (когда все атрибуты являются простыми);
- 2) вторая нормальная форма (когда модель соответствует первой нормальной форме и при этом все неключевые атрибуты неприводимо зависят от первичного ключа);
- 3) третья нормальная форма (когда она соответствует второй нормальной форме и при этом все неключевые поля, содержимое которых может относиться к разным таблицам, вынесены в отдельные таблицы).

Построим логическую модель базы данных в нотации IDEF1X при помощи case-средства ERwin Data Modeler. Логическая модель представлена на рисунке 10.

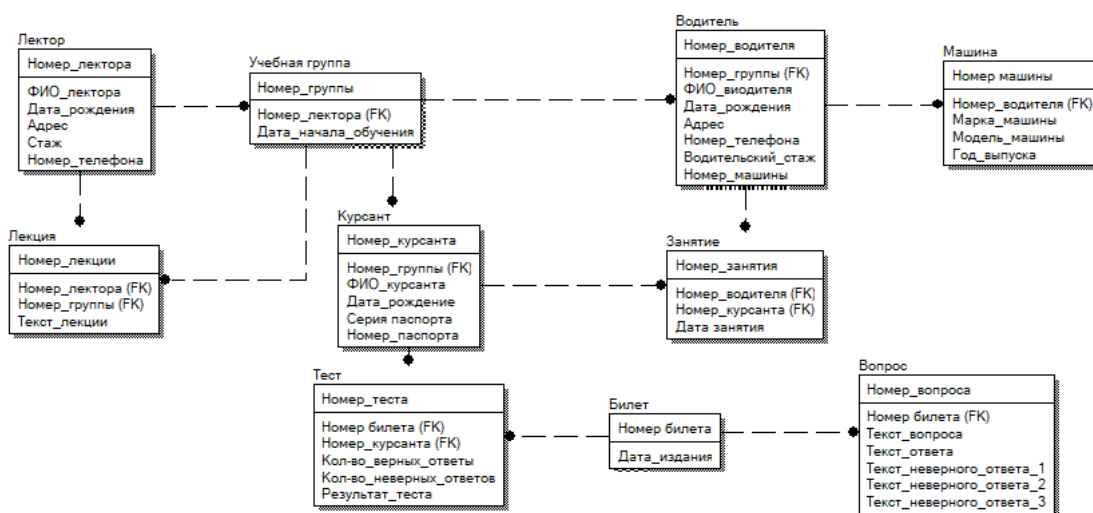


Рисунок 10 – Логическая модель базы данных

Полученная логическая модель состоит из 10 сущностей: «Лектор», «Лекция», «Учебная группа», «Курсанты», «Водитель», «Машина», «Занятие», «Тест», «Билет», «Вопрос». Каждая сущность связана с другой неидентифицирующими связями. Сущность «Лектор» имеет поля содержащие

ФИО лектора, дату рождения, адрес проживания, стаж работы, номер телефона. Сущность «Лекция» содержит идентификатор лектора, идентификатор группы, которой предназначена лекция и сам текст лекции. Сущность «Учебная группа» содержит поле идентификатора лектора, преподающего теоретические знания ПДД и дату начала обучения. Сущность «Машина», содержит поле идентификатора водителя, который преподаёт на этой машине, марку машины, её модель и год выпуска. Сущность «Водитель» содержит информацию о водителе, его ФИО, адрес, дату рождения, номер телефона, водительский стаж, номер машины, номер группы. Сущность «Курсант» состоит из полей, содержащих идентификатор группы, в которой состоит курсант, его ФИО, дату рождения и паспортные данные. Сущность «Билет» содержит номер билета и дату издания билета. Сущность «Вопрос» содержит поле с номером билета, текст вопроса, текст ответа и 3 неверных ответа. Сущность «Тест» содержит поле с номером билета, поле с номером курсанта, который проходил тест, количество верных ответов, количество неверных ответов и результат теста.

Разработанная логическая модель базы данных не содержит транзитивных зависимостей. Таким образом можно сделать вывод, что модель соответствует третьей нормальной форме.

2.4 Пользовательский интерфейс информационной системы

На основе диаграммы вариантов использования и этапа анализа требований был спроектирован простой и интуитивно понятный графический пользовательский интерфейс на языке гипертекстовой разметки HTML с использованием таблиц стилей CSS. На рисунке 11 представлен графический интерфейс формы авторизации доступный пользователям.

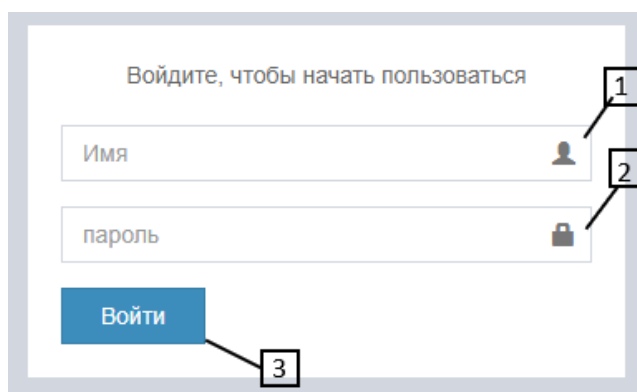


Рисунок 11 – Форма авторизации

Цифрами на рисунке 11 обозначены:

- 1 – поле ввода логина;
- 2 – поле ввода пароля;
- 3 – кнопка отправки формы.

При заполнении формы пользователь нажимает кнопку «Войти», после чего проверяется наличие данного пользователя в базе данных. Если пользователь не найден, будет выведена соответствующая ошибка, иначе, пользователь будет перенаправлен на основную страницу приложения, которое представлено на рисунке 12.

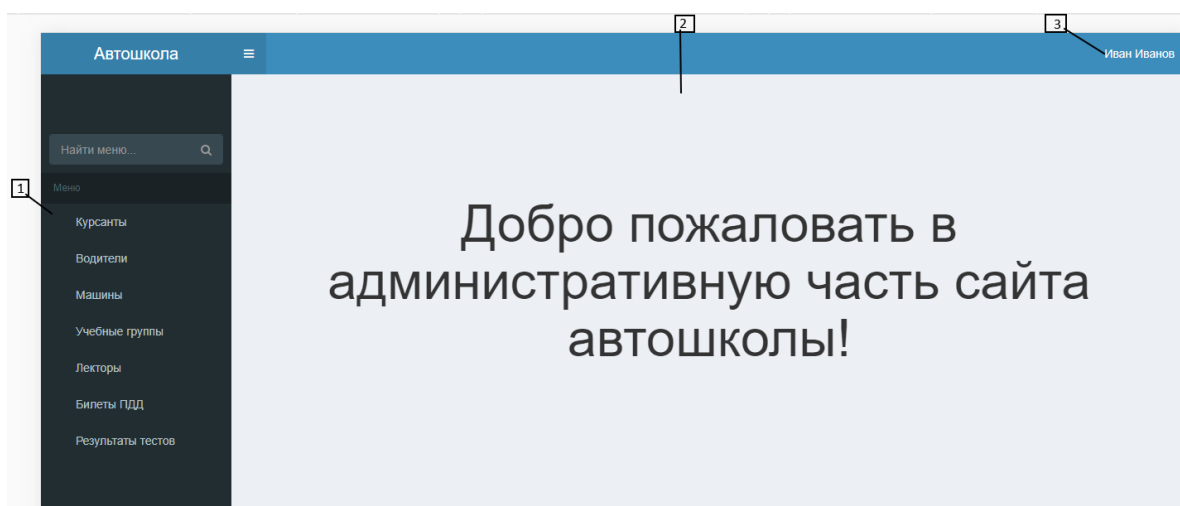


Рисунок 12 – Основная страница приложения

Цифрами на рисунке 12 обозначены:

- 1 – навигационная панель;

2 – основная область отображения форм;

3 – имя авторизованного пользователя.

Рассмотрим типовые страницы приложения на рисунке 13 изображена страница отображения списка зарегистрированных курсантов.

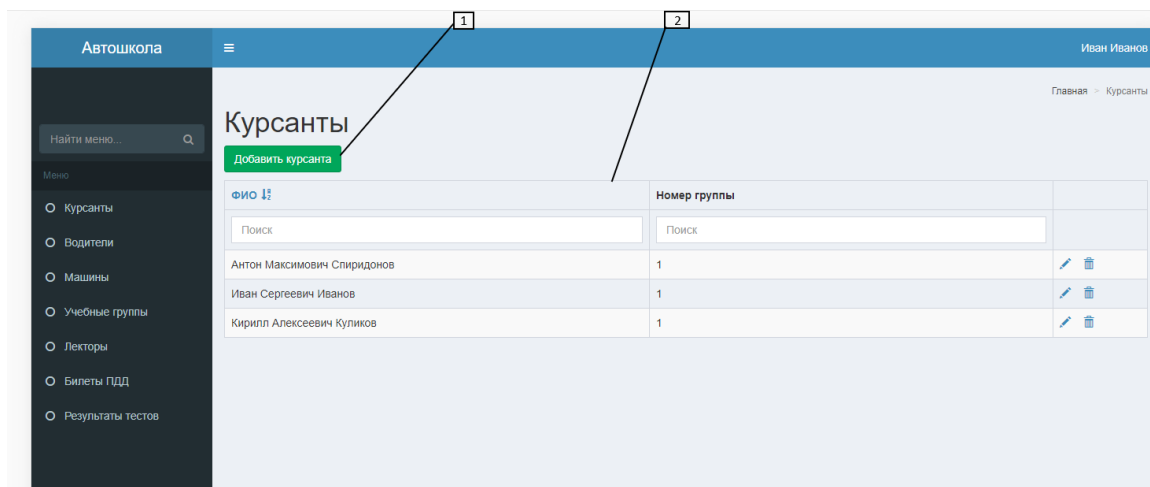


Рисунок 13 – Страница отображения списка курсантов

Цифрами на рисунке 13 обозначены:

1 – кнопка добавления нового курсанта;

2 – таблица отображения списка курсантов с фильтрами, для быстрого поиска и кнопками для изменения данных и удаления записи.

На рисунке 14 изображена страница добавления нового курсанта.

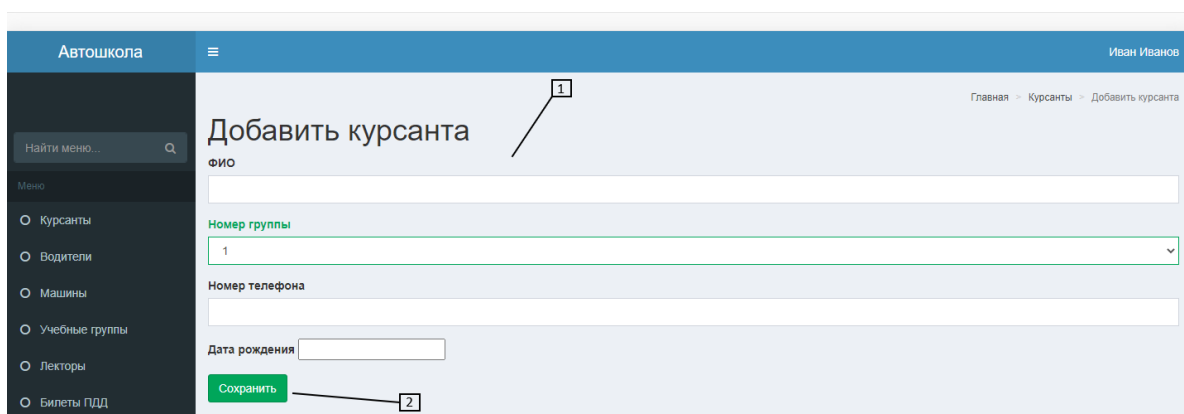


Рисунок 14 – Страница добавления нового курсанта

На рисунке 14 цифрами обозначены:

1 – форма для заполнения данных о новом курсанте;

2 – кнопка сохранения данных о новом курсанте.

На рисунке 15 изображена страница редактирования данных о существующем курсанте автошколы.

The screenshot shows a web application interface for an auto school. The top header is blue with the text 'Автошкола' and a user name 'Иван Иванов'. A sidebar menu on the left contains links like 'Курсанты', 'Водители', 'Машины', etc. The main content area is titled 'Редактировать курсанта: Антон Максимович Спиридонов'. It contains a form with the following fields: 'ФИО' (Anton Maximovich Spiridonov), 'Номер группы' (1), 'Номер телефона' (89542224545), and 'Дата рождения' (2020-01-05). A blue 'Сохранить' button is at the bottom of the form. Callout box 1 points to the 'ФИО' field, and callout box 2 points to the 'Сохранить' button.

Рисунок 15 – Страница редактирование данных курсанта

На рисунке 15 цифрами обозначены:

1 – форма редактирования данных о курсанте;

2 – кнопка сохранения изменённых данных о существующем курсанте.

Интерфейс страницы редактирования сделан похожим на интерфейс создания нового куранта, для более простого понимания пользователем.

На рисунке 16 изображена страница просмотра статистики прохождения тестирования по экзаменационным билетам конкретного курсанта. Цифрами на рисунке 16 обозначены:

1 – ФИО просматриваемого курсанта;

2 – панель отображения круговой диаграммы с данными о прохождении тестирования по билету;

3 – круговая диаграмма с данными результатов прохождении тестов по экзаменационным билетам.

Для отображения статистики выбрана именно круговая диаграмма, так как она позволяет наиболее наглядно показать результаты пройденных тестов. Цветами выделяются в процентном соотношении участки диаграммы с разными результатами тестирования.

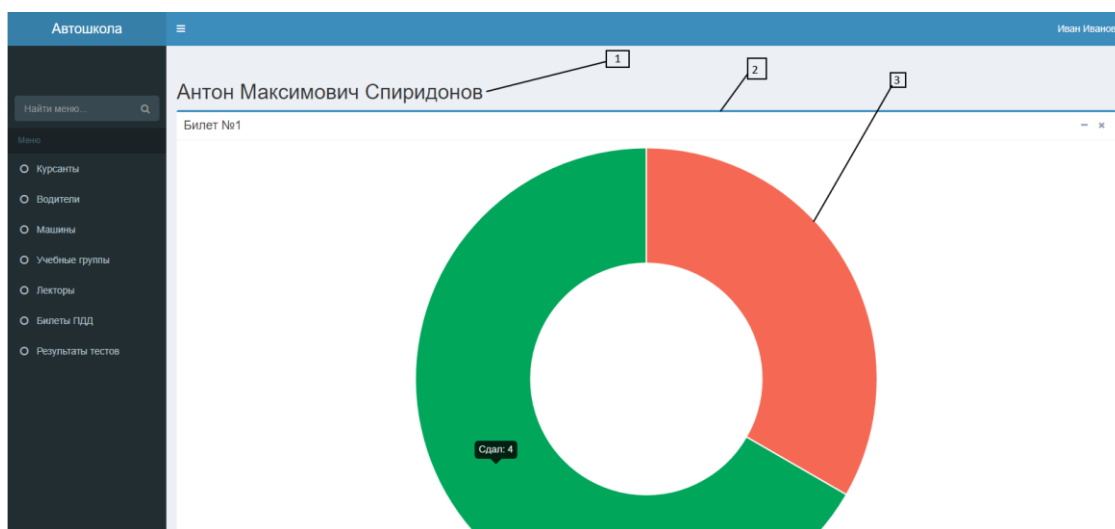


Рисунок 16 – Страница отображение результатов прохождения тестирования

В наши дни важным пунктом для web-приложения является адаптивность интерфейса под мобильные устройства. В ходе проектирования пользовательского интерфейса было учтено использование приложения на мобильных устройствах. Рисунки интерфейса представлены в приложении Г.

2.5 Проектирование программных средств

В соответствии с моделью предметной области были спроектированы следующие классы, представленные на диаграмме классов (рисунок 17).

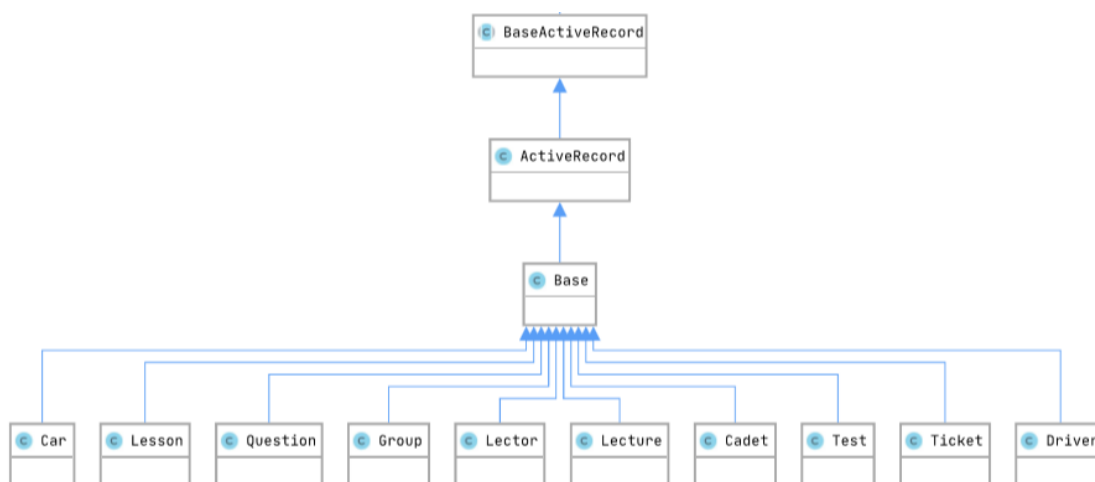


Рисунок 17 – Диаграмма классов

Краткое описание классов приведено ниже.

ActiveRecord – базовый класс обеспечивает объектно-ориентированный интерфейс для доступа и манипулирования данными, хранящимися в базах данных.

Классы-сущности:

- Base – базовый класс, реализующий основные методы каждого зависимого класса;
- Car – класс модели предоставляющий данные из сущности «car» в базе данных;
- Lesson – класс модели предоставляющий данные из сущности «lesson» в базе данных;
- Question – класс модели предоставляющий данные из сущности «question» в базе данных;
- Group – класс модели предоставляющий данные из сущности «group» в базе данных;
- Lecture – класс модели предоставляющий данные из сущности «lecture» в базе данных;
- Lector – класс модели предоставляющий данные из сущности «lector» в базе данных;
- Cadet – класс модели предоставляющий данные из сущности «cadet» в базе данных;
- Test – класс модели предоставляющий данные из сущности «test» в базе данных;
- Ticket – класс модели предоставляющий данные из сущности «ticket» в базе данных;
- Driver – класс модели предоставляющий данные из сущности «driver» в базе данных.

Все классы будут являться моделями при использовании паттерна web-проектирования MVC.

2.6 Планирование проекта и оценка бюджета

В реализации проекта немаловажно оценка времени и стоимости реализации. Рассмотрим перечень задач, представляющих из себя этапы создания проекта (таблица 3).

Таблица 3 – Перечень задач

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований для проекта разработки информационной системы автошколы	-	-
3	Обзор приложений-аналогов	3 дней	1
4	Составление списка функциональных возможностей приложения	5 дня	1
5	Выбор программного средства для реализации приложения	3 дня	3;4
6	Анализ требований завершен	-	5
7	Проектирование для проекта разработки информационной системы автошколы	-	-
8	Проектирование интерфейса приложения	5 дня	6
9	Проектирование структур данных	5 дней	6

Продолжение таблицы 3

№	Название задачи	Длительность	Предшественники
10	Выбор архитектуры программного обеспечения	2 дня	9
11	Проектирование завершено	-	8;10
12	Реализация	-	-
13	Программирование классов приложения	10 дней	11
14	Создание базы данных	6 дней	11
15	Создание интерфейса приложения	4 дня	13;14
16	Реализация завершена	-	15
17	Отладка	-	-
18	Тестирование работы приложения	8 дней	16
19	Исправление ошибок	7 дней	18
20	Отладка завершена	-	19
21	Конец проекта	-	20

Из таблицы 3 видно, что для реализации проекта необходимо осуществить двадцать одну задачу. Определим список затрачиваемых ресурсов проекта (таблица 4).

Таблица 4 – Ресурсы проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист1	Трудовой	110 Р\ч	200 Р\ч
Программист2	Трудовой	95 Р\ч	190 Р\ч

Продолжение таблицы 4

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Интернет	Затраты		
Компьютер	Материальный	1000 Р	
Сервер	Материальный	500 Р	

Распределим ресурсы на все задачи необходимые для реализации проекта (таблица 5).

Таблица 5 – Распределение ресурсов проекта

№	Название задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований для проекта разработки информационной системы автошколы		
3	Обзор приложений-аналогов	Программист1 Интернет Компьютер	100 % 1 шт.
4	Составление списка функциональных возможностей приложения	Программист1 Программист2 Компьютер	50 % 50 % 1 шт.
5	Выбор программного средства для реализации приложения	Программист2 Интернет Компьютер	100 % 1 шт.

Продолжение таблицы 5

№	Название задачи	Ресурс	Единицы
6	Анализ требований завершен		
7	Проектирование для проекта разработки web-приложения для информационной системы автошколы		
8	Проектирование интерфейса приложения	Программист1 Интернет Компьютер	100 % 1 шт.
9	Проектирование структур данных	Программист2 Интернет Компьютер	100 % 1 шт.
10	Выбор архитектуры программного обеспечения	Программист2 Интернет Компьютер	100 % 1 шт.
11	Проектирование завершено		
12	Реализация		
13	Программирование классов приложения	Программист1 Интернет Сервер Компьютер	100 % 1 шт. 1 шт.
14	Создание базы данных	Программист2 Интернет Компьютер Сервер	100 % 1 шт. 1шт.

Окончание таблицы 5

№	Название задачи	Ресурс	Единицы
15	Интеграция клиентской и серверной частей приложения	Программист1 Программист2 Интернет Компьютер Сервер	100 % 50 % 2 шт. 1 шт.
16	Реализация завершена		
17	Отладка		
18	Тестирование работы приложения	Программист1 Программист2 Интернет Компьютер Сервер	100 % 100 % 2 шт. 1 шт.
19	Исправление ошибок	Программист1 Программист2 Интернет Компьютер Сервер	100 % 100 % 2 шт. 1 шт.
20	Отладка завершена		
21	Конец проекта		

Самостоятельно рассчитаем время, которое необходимо для реализации проекта и вычислим его критический путь. Чтобы вычислить критический путь необходимо для каждой задачи определить раннее и позднее время начала работ. Вручную рассчитаем время и занесем все вычисления раннего времени начала работ приведено в таблице 6. В результате должно получиться минимальное количество дней для реализации проекта.

Таблица 6 – Расчет раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	1	0	0	0+0	0
4	1	0	0	0+0	0
5	3	0	3	Max(0+3;0+5)	5
	4	0	5		
6	5	5	3	5+3	8
8	6	8	0	8+0	8
9	6	8	0	8+0	8
10	9	8	5	8+5	13
11	8	8	5	Max(8+5;13+2)	15
	10	13	2		
13	11	15	0	15+0	15
14	11	15	0	15+0	15
15	13	15	10	Max(15+10;15+6)	25
	14	15	6		
16	15	25	4	25+4	29
18	16	29	0	30+0	29
19	18	29	6	29+6	35
20	19	35	4	35+4	39
21	20	39	0	39+0	39

Из результатов таблицы видно, что выполнение проекта должно занять 39 рабочих дней.

Вычисления позднего времени начал работ приведено в таблице 7.

Таблица 7 – Расчет позднего времени начала работ

Работа	Последователи		Длительность	Вычисление	Позднее время начала
	№	Позднее время начала			
21	-	-	-	-	39
20	21	39	0	39-0	39
19	20	39	4	39-4	35
18	19	35	6	35-6	29
16	18	29	0	29-0	29
15	16	29	4	29-4	25
14	15	25	6	25-6	19
13	15	25	10	25-10	15
11	13	15	0	Min(15- 0;19-0)	15
	14	19	0		
10	11	15	2	15-2	13
9	10	13	5	13-5	8
8	11	13	5	13-5	8
6	8	8	0	Min(8-0;8-0)	8
	9	8			
5	6	8	3	8-3	5
4	5	5	5	5-5	0
3	5	5	3	5-3	2
1	3	2	0	Min(0-0;2-0)	0
	4	0	0		

Полученные результаты раннего и позднего времени начала работ позволяют вычислить критический путь для нашего проекта. Для этого

занесем полученные данные в таблицу 8. Работы, у которых резерв времени будет равен нулю, будут составлять критический путь.

Таблица 8 – Нахождение критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	2	2
4	0	0	0
5	5	5	0
6	8	8	0
8	8	8	0
9	8	8	0
10	13	13	0
11	15	15	0
13	15	15	0
14	15	19	4
15	25	25	0
16	29	29	0
18	29	29	0
19	35	35	0
20	39	39	0
21	39	39	0

Из таблицы 8 видно, что критический путь составляют работы с номерами 1, 4, 5, 6, 8, 10, 11, 13, 15, 16, 18, 19, 20, 21.

При помощи системы управления проектами Microsoft Project выполним планирование задач проекта (рисунок 18).

№	Название задачи	ительнос	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало проекта	0 дней	Ср 01.04.20	Ср 01.04.20		
2	Анализ требований для проекта	8 дней	Ср 01.04.20	Пт 10.04.20		
3	Обзор приложений-аналогов	3 дней	Ср 01.04.20	Пт 03.04.20	1	Программист 1;Интернет;Компьютер[1]
4	Составление списка функционал	5 дней	Ср 01.04.20	Вт 07.04.20	1	Программист 1;Программист 2[50%]
5	Выбор программного средства д	3 дней	Ср 08.04.20	Пт 10.04.20	3;4	Программист 2;Компьютер[1];Интернет
6	Анализ требований завершен	0 дней	Пт 10.04.20	Пт 10.04.20	5	
7	Проектирование для проекта раз	7 дней	Пн 13.04.20	Вт 21.04.20		
8	Проектирование интерфейса при	5 дней	Пн 13.04.20	Пт 17.04.20	6	Программист 1;Программист 2[25%];Интернет;Компьютер[1]
9	Проектирование структур данны	5 дней	Пн 13.04.20	Пт 17.04.20	6	Программист 2;Интернет;Компьютер[1]
10	Выбор архитектуры программног	1 день	Пн 20.04.20	Вт 21.04.20	9	Программист 2;Интернет;Компьютер[1]
11	Проектирование завершено	0 дней	Вт 21.04.20	Вт 21.04.20	8;10	
12	Реализация	14 дней	Ср 22.04.20	Пн 11.05.20		
13	Программирование классов прил	10 дней	Ср 22.04.20	Вт 05.05.20	11	Программист 1;Интернет;Сервер[1];Компьютер[1]
14	Создание базы данных	6 дней	Ср 22.04.20	Ср 29.04.20	11	Программист 2;Интернет;Компьютер[1];Сервер[1]
15	Создание интерфейса приложени	4 дней	Ср 06.05.20	Пн 11.05.20	13;14	Программист 1;Программист 2[50%];Интернет;Компьютер[1];Сервер[1]
16	Реализация завершена	0 дней	Пн 11.05.20	Пн 11.05.20	15	
17	Отладка	10 дней	Вт 12.05.20	Пн 25.05.20		
18	Тестирование работы приложени	6 дней	Вт 12.05.20	Вт 19.05.20	16	Программист 1;Программист 2[50%];Компьютер[1];Интернет;Сервер[1]
19	Исправление ошибок	4 дней	Ср 20.05.20	Пн 25.05.20	18	Программист 2;Программист 1[50%];Компьютер[1];Интернет;Сервер[1]
20	Отладка завершена	0 дней	Пн 25.05.20	Пн 25.05.20	19	
21	Конец проекта	0 дней	Пн 25.05.20	Пн 25.05.20	20	

Рисунок 18 – Планирование задач в Microsoft Project

Microsoft Project по задачам, введенным в системе автоматически выстроил диаграмму Ганта (рисунок 19, 20).

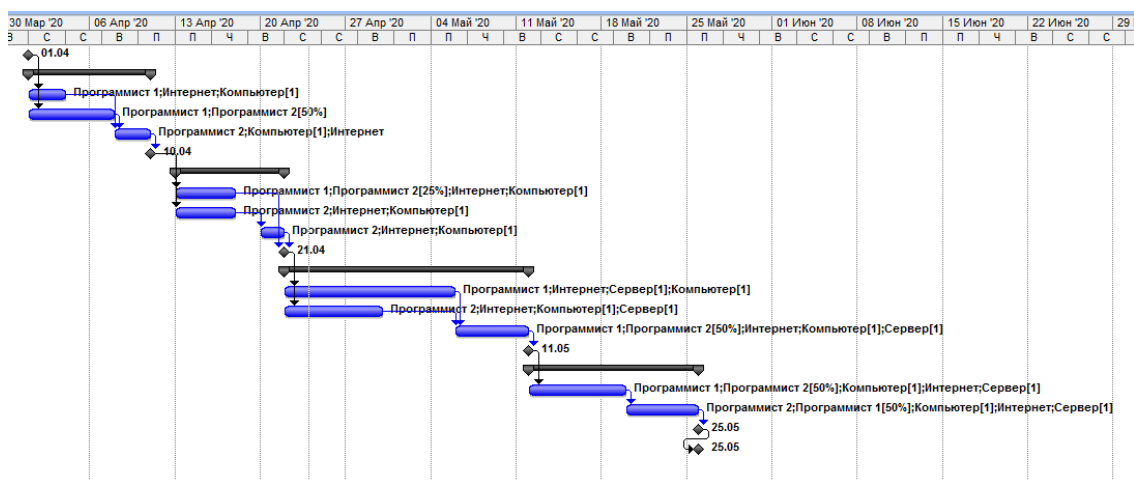


Рисунок 19 – Диаграмма Ганта для реализуемого проекта

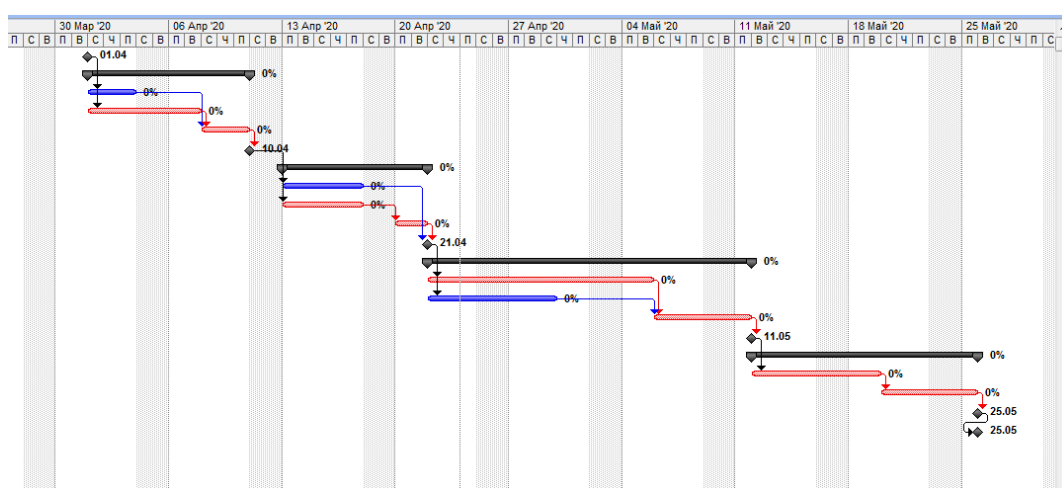


Рисунок 20 – Диаграмма Ганта проекта с отмеченным критическим путем

На рисунке 20 изображен сетевой график построенный программой Microsoft Project Professional. Красным цветом на графике выделен критический путь, содержащий критические работы номер 1, 4, 5, 6, 8, 10, 11, 13, 15, 16, 18, 19, 20 и 21. Если сравнить этот путь с критическим путем, вычисленным вручную, можно сделать вывод, что данные пути совпадают.

На рисунке 21 изображен общий отчет по реализуемому проекту, на котором отмечены автоматически подсчитанные длительность реализации проекта равная 39 дней, что совпадает с вычисленной ранее вручную, и предполагаемые затраты – 62 190 руб.

<u>Даты</u>			
Начало:	Ср 01.04.20	Окончание:	Пн 25.05.20
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
<u>Длительность</u>			
Текущий план:	39 дней	Оставшаяся:	39 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	39 дней	Процент завершения:	0%
<u>Трудозатраты</u>			
Текущий план:	482 ч	Оставшиеся:	482 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	482 ч	Процент завершения:	0%
<u>Затраты</u>			
Текущий план:	62 190,00 Р	Оставшиеся:	62 190,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	62 190,00 Р		
<u>Состояние задачи</u>		<u>Состояние ресурса</u>	
Еще не начатые задачи:	21	Трудовые ресурсы:	1
Выполняющиеся задачи:	0	Трудовые ресурсы с прев. дос:	2
Завершенные задачи:	0	Материальные ресурсы:	2
Итого задач:	21	Итого ресурсов:	5

Рисунок 21 – Обзор проекта

На рисунке 22 и рисунке 23 представлены графики распределения для ресурса «Компьютер» и ресурса «Сервер» соответственно. Из графиков видно, что использование ресурсов не превышают норму.

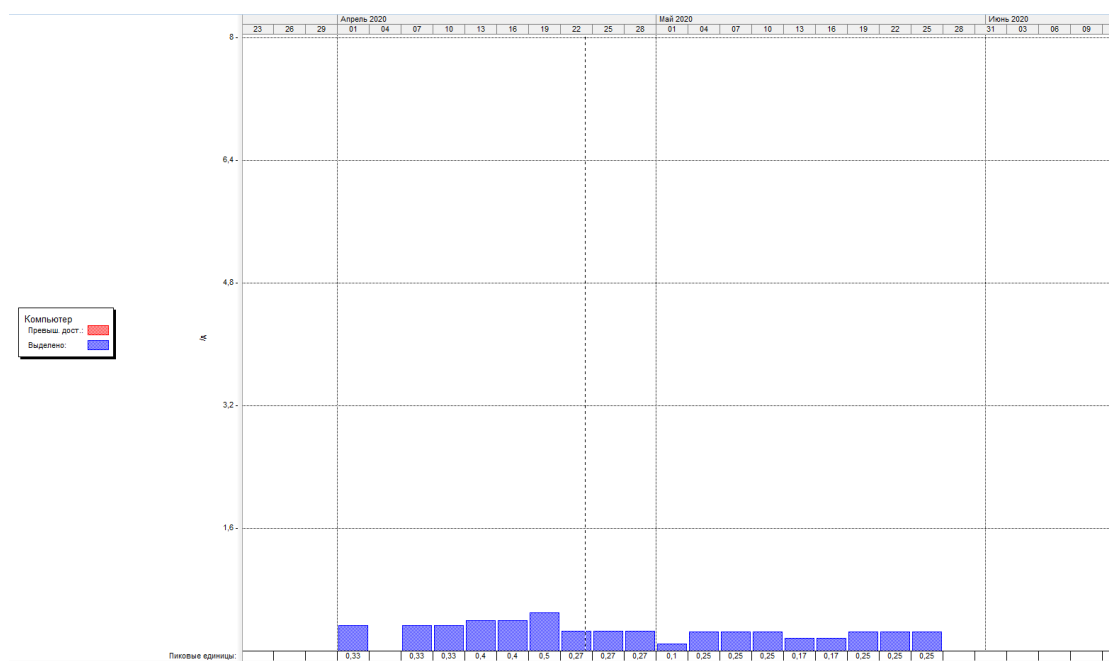


Рисунок 22 – График ресурсов для ресурса «Компьютер»

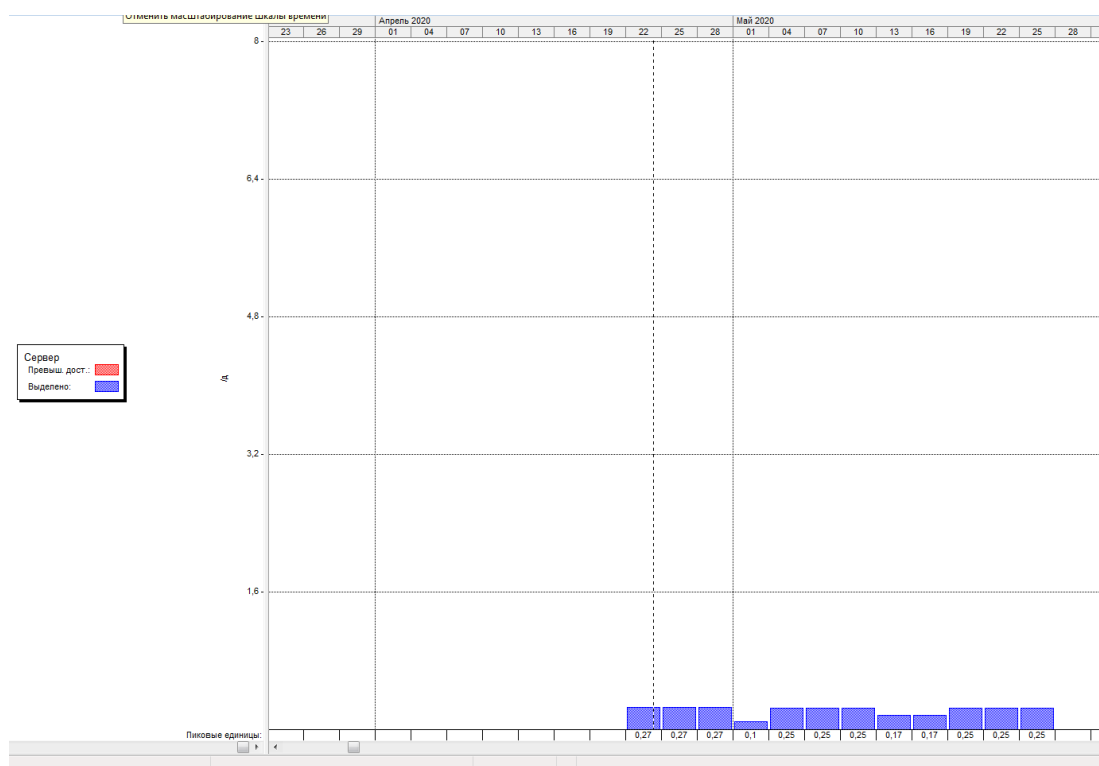


Рисунок 23 – График ресурсов для ресурса «Сервер»

На рисунке 24 и рисунке 25 представлены графики распределения для ресурса «Программист 1» и ресурса «Программист 2» соответственно.

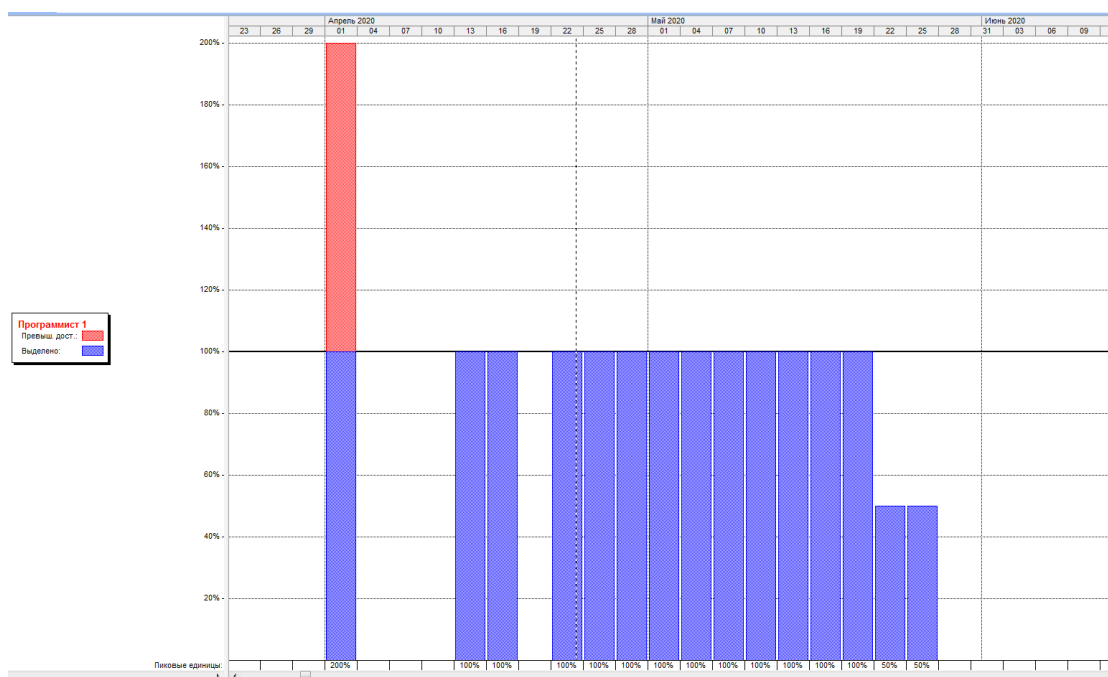


Рисунок 24 – График ресурсов для ресурса «Программист1»

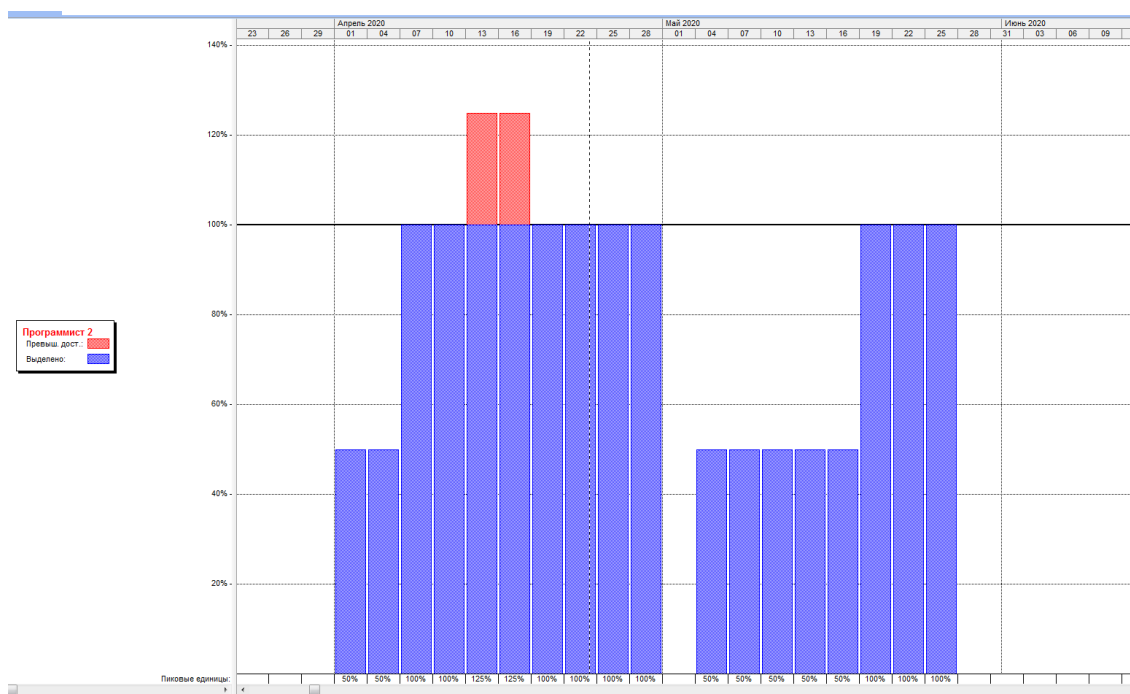


Рисунок 25 – График ресурсов для ресурса «Программист2»

Анализируя графики распределения ресурсов Программист 1 и Программист2, можно сделать вывод что загруженность ресурсов неравномерная.

3 Реализация информационной системы автошколы

3.1 Физическая модель базы данных

Физическая модель строится на основе логической модели данных и учитывает особенности целевой СУБД. В качестве СУБД был выбран продукт MySQL.

При помощи case-средства ERWin Data Modeler была разработана физическая модель, представленная на рисунке 26.

Физическая модель базы данных содержит все поля, описанные в логической модели, все связи сохранены и определены типы полей.

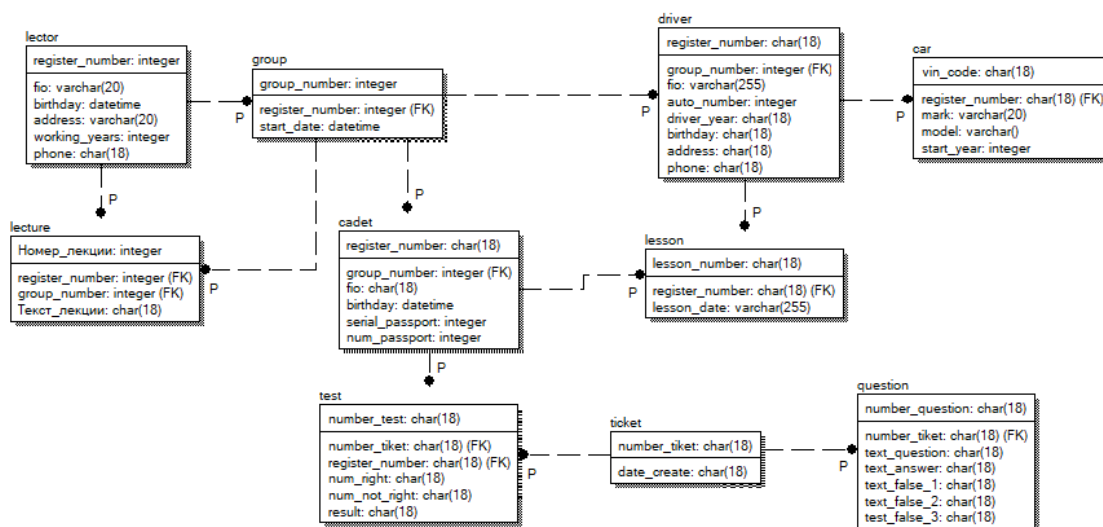


Рисунок 26 – Физическая модель базы данных

Сгенерированный скрипт DDL в ERWin Data Modeler был выполнен в СУБД MySQL. В результате была получена схема базы данных, представленная на рисунке 27.

В полученной базе данных присутствуют все поля, определены первичные и вторичные ключи, сохранены все связи, определены типы данных всех полей.

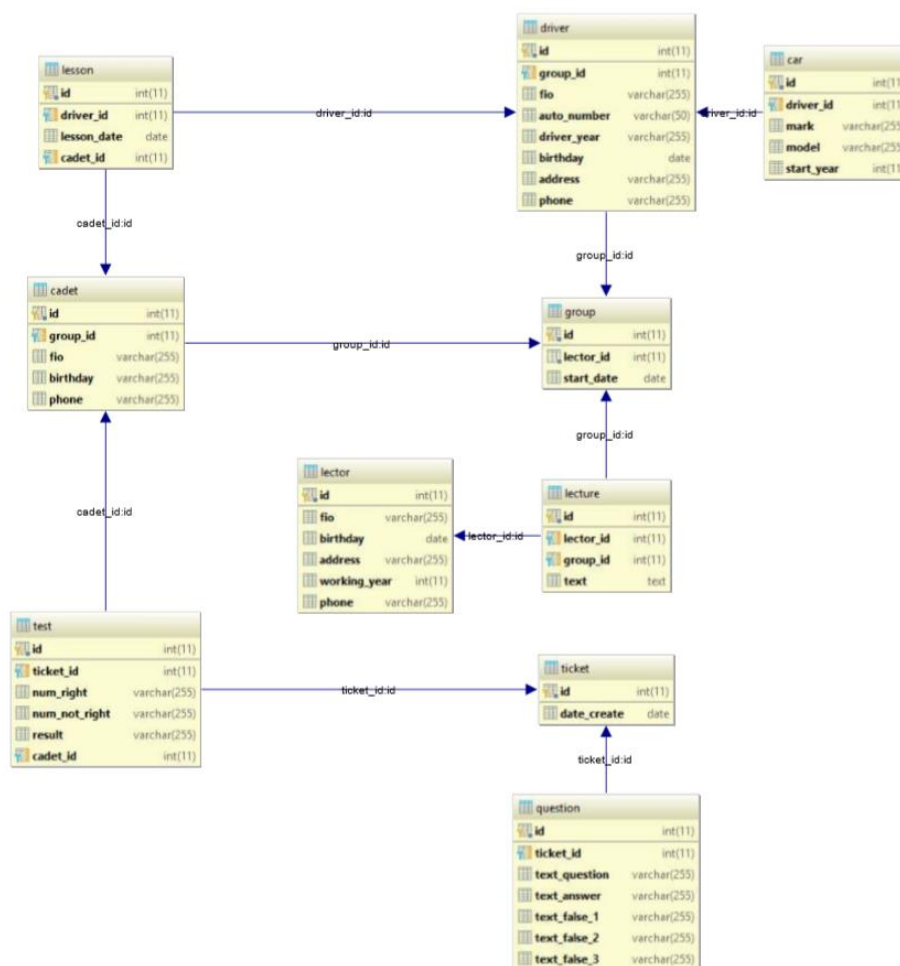


Рисунок 27 – Схема базы данных

Схема базы данных, полученная в СУБД MySQL и физическая модель базы данных полностью совпадают, следовательно, можно сделать вывод, что база данных реализована корректно.

3.2 Модель реализации клиентского приложения

Модель реализации описывает то, как реализуются элементы модели проектирования в виде компонентов. Для отображения этой реализации используется одна из UML-диаграмм, которая называется диаграммой компонентов (рисунок 28).

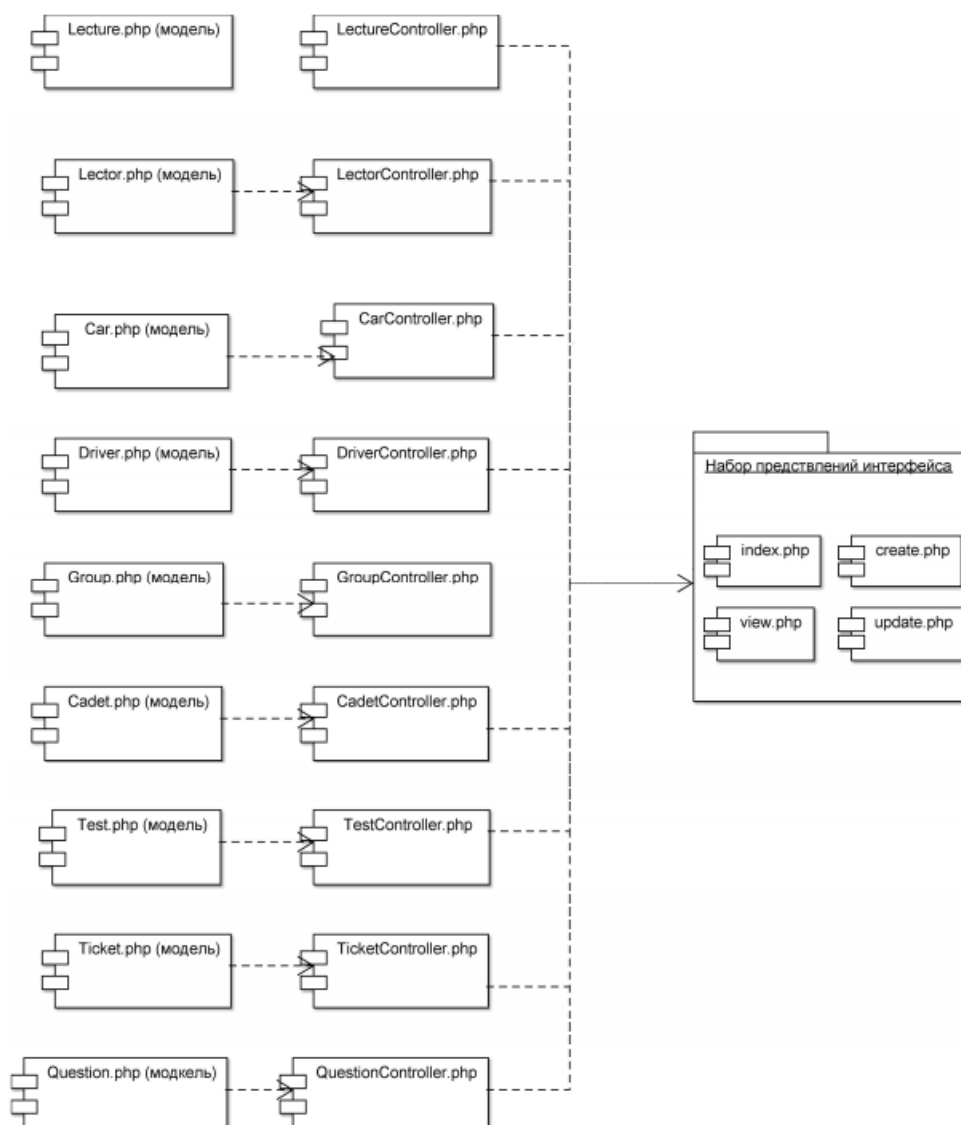


Рисунок 28 – Диаграмма компонентов

Исходный код информационной системы на языке PHP частично приведен в приложении Б.

Одним из способов моделирования статического вида системы с точки зрения развертывания является диаграмма развертывания (рисунок 25).

Диаграмма развертывания показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения – маршруты передачи информации между аппаратными узлами.

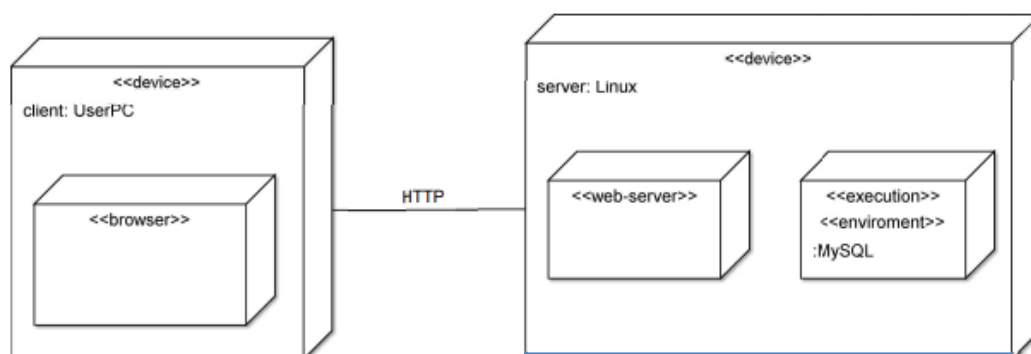


Рисунок 29 – Диаграмма развертывания

Из диаграммы видно, что для развертывания приложения необходим сервер для хранения web-сервера и базы данных, со стороны клиента, необходимо, чтобы на рабочей станции был установлен браузер. Передача данных идет по средствам HTTP-запросов.

4 Контроль качества программного обеспечения

Контроль качества программного обеспечения – неотъемлемый этап в разработке приложений. Одним из важнейших способов контроля качества является тестирование программного обеспечения.

Тестирование – способ обеспечения качества программного продукта, которое представляет собой ряд мероприятий, связанных с различного рода испытаниями тестируемого объекта для установления соответствия или несоответствия техническим требованиям и направленных на выявление дефектов [17].

Под дефектами понимаются ошибки в программном обеспечении, которые ведут к нарушению работы приложения. Необнаруженные на стадии разработки дефекты способны принести заказчику материальные убытки, а также исправление таких ошибок при эксплуатации программного обеспечения может быть затруднено и экономически невыгодно. Поэтому тестирование является неотъемлемой частью разработки приложений.

Для выпускной квалификационной работы было выбрано тестирование методом черного ящика, которое подразумевает по собой осуществление функционального тестирования.

4.1 Функциональное тестирование

Функциональное тестирование – способ проверки программного обеспечения путем составления плана тестирования, который строится на основе прецедентов, описанных в диаграмме вариантов использования. Следовательно, план тестирования включает в себя не только описание самих тестов, но и их предполагаемые результаты выполнения.

Функциональное тестирование является одним из ключевых видов тестирования, задача которого установить соответствие разработанного программного обеспечения исходным функциональным требованиям заказчика [14].

План функционального тестирования приведен в таблице 9, а его результаты в приложении Д.

Таблица 9 – План функционального тестирования

№	Вариант использования	Тест	Результаты
1	Авторизация	Авторизация с неверными данными	Тест выполнен. Система не нашла пользователя в базе и выдала ошибку (рисунок Д.1).
2		Авторизация с верными данными	Тест выполнен. Система перенаправила пользователя на главную страницу (рисунок Д.2).
3	Добавить курсанта	Открыть форму для добавления курсанта. Не заполняя полей, нажать на кнопку «Сохранить».	Тест выполнен. Система вывела сообщения об ошибке (рисунок Д.3).
4		Открыть форму для добавления курсанта. Заполнить поля и нажать на кнопку «Сохранить».	Тест выполнен. Новый пользователь добавлен в систему. Система вывела сообщение об успешном добавлении новой записи (рисунок Д.4).
5		Открыть форму редактирования данных курсанта. Изменить ФИО. Нажать кнопку сохранить	Тест выполнен. Новый пользователь добавлен в систему. Система вывела сообщение об успешном изменении записи (рисунок Д.5).
6	Изменение теоретической базы	Редактирование лекции	Тест выполнен. Новый пользователь добавлен в систему. Система вывела сообщение об успешном изменении записи (рисунок Д.6).
7		Редактирование билета	Тест выполнен. Новый пользователь добавлен в систему. Система вывела сообщение об успешном изменении записи (рисунок Д.7).
8	Просмотр статистики каждого пользователя	Просмотр статистики определенного курсанта пользователя	Тест выполнен. Новый пользователь добавлен в систему. Система вывела сообщение об успешном изменении записи (рисунок Д.8).

Продолжение таблицы 9

№	Вариант использования	Тест	Результаты
9	Изменение данных об инструкторе	Открыть форму редактирования данных водителя. Изменить ФИО. Нажать кнопку сохранить	Тест выполнен. Новый пользователь добавлен в систему. Система вывела сообщение об успешном изменении записи (рисунок Д.9).
10	Записаться на вождение	Открыть форму для записи на урок. Выбрать время и нажать кнопку «Записаться».	Тест выполнен. Система добавила новую запись в таблицу и выдала сообщение (рисунок Д.10).
11	Прохождение тестов	Открыть форму прохождения тестов, ответить на все вопросы и нажать кнопку «Завершить тестирование»	Тест выполнен. Система занесла результаты в базу данных и выдала окно с результатами прохождения (рисунок Д.11).

Таким образом, проанализировав результаты функционального тестирования можно сделать вывод о том, что разработанное программное обеспечение подсистемы отчетности автоматизированной охранной системы работает корректно и соответствует исходным функциональным требованиям заказчика.

4.2 Метрики кода

Метрики кода представляют собой набор оценок программного обеспечения, которые дают разработчикам более глубокое представление о разрабатываемом коде и служат рекомендательными характеристиками. Они позволяют руководителям проектов и предприятий изучить сложность разработанного или разрабатываемого проекта, оценить объем работ, стилистику разрабатываемой программы и усилия, потраченные каждым разработчиком для реализации решения [16].

Примером часто используемой метрики является комплексный показатель качества кода – Maintainability Index. Рассчитывается метрика по формуле:

$$MI = \max\left(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * \frac{100}{171}\right)$$

где HV – Halstead Volume, вычислительная сложность: чем больше операторов, тем больше значение этой метрики;

CC – Cyclomatic Complexity, показатель структурной сложности кода, т.е. количество различных ветвей в коде, чем больше этот показатель, тем больше тестов должно быть написано, для полного покрытия кода;

LoC – количество строк кода.

Комплексный показатель качества кода может принимать значения от 0 до 100 и показывает относительную сложность поддержки кода. Чем больше значение этой метрики, тем легче поддерживать код.

Для определения вычислительной сложности программы сначала следует рассчитать словарь (1) и длину программы (2), а затем вычислить на их основе вычислительную сложность (3).

Словарь программы:

$$\eta = \eta_1 + \eta_2, \quad (1)$$

где η_1 – число простых (отдельных) операторов и операций в программе;

η_2 – число простых (отдельных) операндов (констант и переменных) в программе.

Длина программы:

$$N = N_1 + N_2, \quad (2)$$

где N_1 – общее число простых (отдельных) операторов и операций в программе;

N_2 – общее число простых (отдельных) операндов в программе.

Вычислительная сложность:

$$HV = N \log_2 \eta. \quad (3)$$

Структурную сложность программы можно рассчитать таким образом:

$$CC = KOP + KOЦ + 1,$$

где KOP – количество операторов разветвления;

KOЦ – количество операторов цикла.

Если полученное значение CC больше 10, то это свидетельствует о структурной сложности программы.

Проведем анализ метрик кода для функции, которая проверяет, была ли найдена запись в архиве сообщений (листинг 1).

Листинг 1 – Функция нахождения свободного времени для записи на вождение на языке PHP

```
public function getTime($date = null)
{
    $curTimes = [];
    if (empty($date)) {
        $date = date('Y-m-d');
    }
    if (($date >= date('Y-m-d'))) {
        $curLessonTimes = Lesson::find()->where(['=' , date',
$date])->select(['date'])
        ->orderBy(['date' => SORT_ASC])->column();
        $ curLessonTimes =
array_count_values($curLessonTimes);
        foreach ($curLessonTimes as $time => $count) {
            $realTime = LessonCount::findOne(['name' =>
$time]);
            if ((empty($realTime)) || ($count >= $realTime-
>count)) {
                $curTimes[] = $time;
            }
        }
        foreach (OrderDeliveryCount::find()->all() as $time)
        {
            $r_time = trim(explode('-', $time->name)[1]);
            if (strtotime(date('Y-m-d H:i',
+strtotime('+24 hours')) >=
strtotime($date . ' ' . $r_time) || $time->count == 0) {
                $curTimes[] = $time->name;
            }
        }
    } else {
        $curTimes = LessonCount::find()->select('name')-
>orderBy(['name' => SORT_ASC])->column();
    }
    return $curTimes;
}
```

Произведем расчет комплексного показателя качества кода (MI) для функции (смотри листинг 1).

Данные, необходимые для расчета формул 1 и 2 приведены ниже в таблице 10.

Таблица 10 – Данные для расчета словаря и длины программы

Операторы			Операнды		
№	Оператор	Количество	№	Операнд	Количество
1	if	4	1	curTimes	4
2	foreach	1	2	date	1
3	return	2	3	curLessonTimes	2
4	=	8	4	realTime	4
5	>=	1			
6	&	3			
7		2			
8	==	1			
9	;	8			
10	->	9			
Итого	$\eta_1=10$	$N_1=39$		$\eta_2=4$	$N_2=11$

Словарь программы:

$$\eta = \eta_1 + \eta_2 = 10 + 4 = 14.$$

Длина программы:

$$N = N_1 + N_2 = 39 + 11 = 50.$$

Вычислительная сложность:

$$HV = 50 \times 3,807 = 190.35.$$

Количество строк кода:

$$LoC = 24.$$

Структурная сложность:

$$CC = 3 + 1 + 1 = 5,$$

$$MI = \max \left(0, (171 - 5.2 * \ln(190.35) - 0.23 * 5 - 16.2 * \ln(24)) * \frac{100}{171} \right) = 112, 4.$$

Комплексный показатель качества равен приблизительно 112.

На основе результатов расчёта можно сделать вывод о том, что структурная сложность данной функции невысока, следовательно, программное обеспечение не является структурно сложным. Комплексный показатель качества кода находится чуть выше среднего уровня, что говорит о том, что исходный код программного обеспечения сравнительно просто поддерживать.

Таким образом, программное обеспечение информационной системы автошколы в целом имеет высокий уровень качества.

Заключение

В ходе выполнения выпускной квалификационной работы была разработана информационная система автошколы.

После рассмотрения предметной области и постановки задач был проведен сравнительный анализ разработанной информационной системы и аналогичных продуктов, выявлены их преимущества и недостатки. На его основе были составлены требования к разрабатываемой системе.

На этапе проектирования системы было осуществлено планирование разработки и оценка бюджета, которое показало, что длительность реализации проекта составляет 39 дней и на его реализацию следует затратить 62 190 руб. Также на этом этапе были созданы концепции окон пользовательского интерфейса приложения, разработана структура классов, реализующая его.

Разработка программного средства была разделена на две составляющие: разработку клиентской части и разработку серверной части. Клиентская часть должна быть представлена набором веб-страниц на языке гипертекстовой разметки HTML5 с использованием CSS-стилей и языка программирования JS. Серверная часть должна быть выполнена на языке PHP5.6 с использованием фреймворка Yii2 и использовать СУБД MySQL. Веб-страницы должны быть адаптированы для мобильных устройств.

Тестирование клиентской части приложения проводилось методами функционального. Также были просчитаны метрики кода для оценки его качества.

Далее приложение будет дорабатываться, будет реализовано расширение функциональных возможностей администраторской части для работы с документами. Также будут расширены функциональные возможности пользователей и добавлен чат с инструкторами, лекторами и администраторами.

Как видно из всего вышесказанного, поставленная задача по реализации информационной системы автошколы была выполнена.

Список использованных источников

1. Указ президента Российской Федерации «О стратегии развития информационного общества в Российской Федерации на 20017–2030 годы». [Электронный ресурс]. – URL: <http://pravo.gov.ru/proxy/ips/?docbody=&nd=102431687> (дата обращения: 19.05.2020).
2. «Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации. Цифровая экономика РФ». [Электронный ресурс]. – URL: <https://digital.gov.ru/ru/activity/directions/858/> (дата обращения: 19.05.2020).
3. Анализ требований // Wikipedia [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Анализ_требований (дата обращения: 19.05.2020)
4. RUP. Общие сведения // Informicus [Электронный ресурс]. - URL: <http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=7> (дата обращения: 23.05.2020)
5. Технология RUP // Narod [Электронный ресурс]. - URL: <http://mei06.narod.ru/sem9/pis/shpora/48.htm> (дата обращения: 23.05.2020)
6. Асинхронный веб // TProger [Электронный ресурс]. - URL: <https://tproger.ru/translations/what-are-web-sockets> (дата обращения: 23.05.2020)
7. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2е издание / Д. Арлоу, И. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.
8. Методология. Стандарт IDEF1 // Iteranet IT-журнал [Электронный ресурс]. – URL: <http://iteranet.ru/it-novosti/2013/10/16/metodologiya-standart-idef1/> (дата обращения: 22.05.2020).
9. Хаф Л. Методология разработки программного обеспечения // Компьютер Пресс [Электронный ресурс]. – URL: <https://compress.ru/article.aspx?id=9633> (дата обращения: 22.05.2020).

10. Богданов, В.В. Управление проектами в Microsoft Project 2007. Учебный курс (+CD) / В.В. Богданов – СПб.: Питер, 2008. – 592 с.
11. Конноли Т., Бегг К., Страчан А. Базы данных: Проектирование, реализация и сопровождение. Теория и практика. Второе издание. – М.: Вильямс, 2000. – 1139 с.
12. WPF 4 : статья // ProfessorWeb : сайт. 2011. URL: http://professorweb.ru/my/WPF/base_WPF/level1/1_8.php (Дата обращения 25.05.2020).
13. Новиков Д.А. Управление проектами: Организационные механизмы. – М.: ПМСОФТ, 2007. – 140 с.
14. Котляров В.П., Коликова Т.В. Основы тестирования программного обеспечения. – М.: Бином. Лаборатория знаний, 2006. – 285 с.
15. Канер С., Фолк Д, Нгуен Е.К. Тестирование программного обеспечения. – К.: ДиаСофт, 2001. – 544 с.
16. Майерс Г. Искусство тестирования программ. – М.: Финансы и статистика. – 2004. – 267 с

Приложение А
ГЛОССАРИЙ
(справочное)

ИС – информационные системы.

АП – активное правило.

БД – база данных.

ПрО – предметная область (ПрО).

НБД – наблюдаемая база данных.

СУБД – система управления базами данных.

DDL (Data Definition Language) – язык описания данных.

URL (Uniform Resource Locator) – единый указатель ресурса.

MVC (Template View) – паттерн «Модель-представление-контроллер».

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, основанный на технологии «клиент-сервер».

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

СУВ – системы управления версиями.

ПО – программное обеспечение.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Приложение Б

ЛИСТИНГ ПРОГРАММЫ

(рекомендуемое)

main.php

```

var express = require('express');
var http = require('http');
var path = require('path');
var config = require('config');
var log = require('lib/log')(module);
var mongoose = require('lib/mongoose');
var HttpError = require('error').HttpError;

var app = express();

app.engine('ejs', require('ejs-locals'));
app.set('views', path.join(__dirname, '/templates'));
app.set('view engine', 'ejs');

app.use(express.favicon());

if (app.get('env') === 'development') {
  app.use(express.logger('dev'));
} else {
  app.use(express.logger('default'));
}

app.use(express.bodyParser());
app.use(express.cookieParser());
var MongoStore = require('connect-mongo')(express);
app.use(express.session({
  secret: config.get('session:secret'),
  key: config.get('session:key'),
  cookie: config.get('session:cookie')
  // store: new MongoStore({mongoose_connection: mongoose.connection})
}));

app.use(require('middleware/sendHttpError'));
app.use(require('middleware/loadUser'));
app.use(app.router);
require('routes')(app);
app.use(express.static(path.join(__dirname, 'public')));

var livereload = require('livereload');
var lrserver = livereload.createServer();
lrserver.watch([__dirname + "/public", __dirname + "/templates/*.ejs",
__dirname + "/layout", __dirname + "/partials"]);

app.use(function(err, req, res, next) {
  if (typeof err === 'number') { // next(404);
    err = new HttpError(err);
  }

  if (err instanceof HttpError) {
    res.sendHttpError(err);
  } else {
    if (app.get('env') === 'development') {
      express.errorHandler()(err, req, res, next);
    } else {
      log.error(err);
      err = new HttpError(500);
      res.sendHttpError(err);
    }
  }
});

http.createServer(app).listen(config.get('port'), function() {
  log.info('Express server listening on port ' + config.get('port'));
});

```

index.php

```

var checkAuth = require('middleware/checkAuth');
var HttpError = require('error').HttpError;

module.exports = function(app) {

    app.get('/', require('./startpage').get);

    app.get('/login', require('./login').get);
    app.post('/login', require('./login').post);

    app.post('/logout', require('./logout').post);

    app.get('/construct', checkAuth, require('./construct').get);
    app.get('/auditor', checkAuth, require('./auditor').get);
    app.get('/settings', checkAuth, require('./settings').get);

    app.get('/analyze', function(req, res, next) {
        return next(new HttpError(404, 'Страница не найдена, либо находится в
        разработке'));
    });
};

```

login.php

```

var User = require('models/user').User;
var HttpError = require('error').HttpError;
var AuthError = require('models/user').AuthError;
var async = require('async');

exports.get = function(req, res) {
    res.render('login');
};

exports.post = function(req, res, next) {
    var username = req.body.username;
    var password = req.body.password;
    User.authorize(username, password, function(err, user) {
        if (err) {
            if (err instanceof AuthError) {
                return next(new HttpError(403, err.message));
            } else {
                return next(err);
            }
        }
        req.session.user = user._id;
        res.send({});
    });
};

```

Logout.php

```

exports.post = function(req, res) {
    req.session.destroy();
    res.redirect('/');
};

```

view.php

```

<!DOCTYPE html>
<html>
<head>
    <title><%=blocks.title%></title>

    <link rel="stylesheet"
href="/vendor/bower_components/bootstrap/dist/css/bootstrap.css"/>
    <link rel="stylesheet" href="/css/app.css"/>
    <script type="javascript" src="/js/main.js"></script>

```

```

        <script src="/vendor/bower_components/jquery/dist/jquery.js"></script>
        <script
src="/vendor/bower_components/bootstrap/dist/js/bootstrap.js"></script>

</head>
<body>
<header>
    <%-blocks.header%>
    <%-partial('../partials/topNav')%>
</header>
<section class="content-wrap">
    <div class="col-md-2 nav-left">
        <h4>Active Rules Manager</h4>
        <ul class="col-md-12 nav-btns-wrap">
            <li id="constructor" class="col-md-12 btn"><a
href="/construct">Конструктор правил</a></li>
            <li id="analyze" class="col-md-12 btn"><a
href="/analyze">Анализ</a></li>
            <li id="audit" class="col-md-12 btn"><a href="/auditor">Аудит
Исполнения</a></li>
            <li id="settings" class="col-md-12 btn active"><a
href="/settings">Настройки</a></li>
        </ul>
    </div>
    <div class="col-lg-10 container">
        <h1><%=blocks.title%></h1>
        <%-body -%>
    </div>
</section>
<footer>
    <%-partial('../partials/botNav')%>
    <%-blocks.footer%>
</footer>
</body>
</html>

```

login.php

```

<% layout('../layout/page') -%>
<% block('title', "Регистрация и вход") -%>

<p>Введите имя пользователя и пароль. Если такого пользователя нет - он будет
создан.</p>

<form class="form-horizontal login-form" name="login-form">
    <div class="form-group">
        <label for="input-username" class="col-lg-2 control-
label">Имя</label>

        <div class="col-lg-10">
            <input name="username" value="name" type="text" class="form-
control" id="input-username" placeholder="Имя">
        </div>
    </div>
    <div class="form-group">
        <label for="input-password" class="col-lg-2 control-
label">Пароль</label>

        <div class="col-lg-10">
            <input name="password" value="pass" type="password" class="form-
control" id="input-password" placeholder="Пароль">
        </div>
    </div>
</form>

```

```

<div class="form-group">
  <div class="col-lg-offset-2 col-lg-10">
    <button type="submit" class="btn btn-primary" data-loading-
text="Отправляю...">Войти</button>
    <span class="help-block error"></span>
  </div>
</div>
</form>
<script>
  $(document.forms['login-form']).on('submit', function() {
    var form = $(this);
    $('.error', form).html('');
    $(":submit", form).button("loading");
    $.ajax({
      url: "/login",
      method: "POST",
      data: form.serialize(),
      complete: function() {
        $(":submit", form).button("reset");
      },
      statusCode: {
        200: function() {
          form.html("Вы вошли в сайт").addClass('alert-success');
          window.location.href = "/construct";
        },
        403: function(jqXHR) {
          var error = JSON.parse(jqXHR.responseText);
          $('.error', form).html(error.message);
        }
      }
    });
    return false; });
</script>

```

construct.php

```

<% layout('layout/page') -%>
<% block('title', 'Конструктор АП'); -%>

<div class="col-md-12">
  <form class="rule-creator-form">
    <div class="tabs">
      <input id="tab1" type="radio" name="tabs">
      <label for="tab1" title="Основные параметры">Основные
параметры</label>

      <input id="tab2" type="radio" name="tabs">
      <label for="tab2" title="Вкладка 2">Событие вызова</label>

      <input id="tab3" type="radio" name="tabs" checked>
      <label for="tab3" title="Вкладка 3">Условие срабатывания</label>

      <input id="tab4" type="radio" name="tabs">
      <label for="tab4" title="Вкладка 4">Вызываемые действия</label>

    <section id="content-tab1">
      <p>Ввод/модификация основных характеристик АП</p>
      <h4>Название</h4>
      <input id="ruleName" name="ruleName" placeholder=" ">
      <h4>Комментарий</h4>
      <textarea id="ruleDesc" name="ruleDesc" placeholder="
"></textarea>

      <h4>Наблюдаемая база данных</h4>
      <input id="dbName" name="dbName" placeholder=" ">
      <h4>Текущее состояние правила</h4>
      <span id="ruleState" style="color: orange">on Create</span>
    </section>
  </form>

```

```

</section>
<section id="content-tab2">
  <p>
    Выберите тип события данного правила:
  </p>
  <input id="ruleEvent" type="radio" name="ruleEvent">
  <label for="ruleEvent" title="Событие1">Событие1</label>
</section>
<section id="content-tab3">
  <p>
    Выбор/создание условия создания активного правила
  </p>
  <table class="table conditions">

<tr><th>Выбор</th><th>№</th><th>Название</th><th>Содержание</th>
  </tr>
  <% conditions.foreach( function() {
    Document.write("<tr><td><input type='checkbox' name='ruleCond'
value='condition.id' ></td><td>" + condition.id + "</td><td>" +
condition.name + "</td><td>" + condition.content + "</td><td><a class='edit-
link' href='\" + condition.id + \"'>Редатировать</a></td></tr>")
  }%>

  </table>
  <form class="cond-form">
    <h4>Название условия</h4>
    <input id="ruleCondName" type="text" name="ruleCondName">
    <h4>Содержание условия</h4>
    <textarea id="ruleCondDesc" type="text"
name="ruleCondDesc"></textarea>

    <p><input type="button" class="btn" value="Добавить"></p>
  </form>
</section>
<section id="content-tab4">
  <p>
    Выбор/создание действий вызываемых правилом:
  </p>
  <table class="table actions">
    <tr>
      <th>Выбор</th>
      <th>№</th>
      <th>Название</th>
      <th>Содержание</th>
    </tr>
    <% actions.foreach( function() {
      Document.write("<tr><td><input type='checkbox' name='ruleCond'
value='action.id' ></td><td>" + action.id + "</td><td>" + action.name +
"</td><td>" + action.content + "</td><td><a class='edit-link' href='\" +
action.id + \"'>Редатировать</a></td></tr>") }%>
    </table>
    <form class="action-form">
      <h4>Название действия</h4>
      <input id="ruleActionName" type="text"
name="ruleActionName">
      <h4>Содержание действия</h4>
      <textarea id="ruleActionDesc" type="text"
name="ruleActionDesc"></textarea>
      <p><input type="button" class="btn" value="Добавить"></p>
    </form>
  </section>
</div>
<input type="button" class="btn" value="Сохранить">

```

```

    </form>
</div>

```

create.php (представление аудитора)

```

<% layout('layout/page') -%>
<% block('title', 'Аудит Исполнения АП'); -%>
<div class="col-xs-6 table-wrap">
    <h2>Аудит Исполнения</h2>
    <table class="rule table">
        <tr>
            <th>№</th>
            <th>Название</th>
            <th>Пинг</th>
            <th>Статус</th>
        </tr>
    </table>
</div>
<div>
    <div>
        <h2>Журнал</h2>
        <h2 class="col-xs-6"><button class="btn btn-success" onclick="loadLog
() ">Обновить</button></h2>
    </div>
    <table class="table auditor-log col-xs-12">
        <tr>
            <th>Время</th>
            <th>Содержание</th>
        </tr>
    </table>
</div>

<script>
$(document).ready(function () {
    loadTables ();
});
function loadTables (type){
    loadLog ();
    loadRule ();
    return false;
}
function loadLog () {
    $.ajax({
        url: "/auditor",
        method: "POST",
        error: function(jqXHR) {
            var error = JSON.parse(jqXHR.responseText);
            $('.error', form).html(error.message);
        },
        complete: function(journalRows) {
            var table = $('.auditor-log');
            $('.auditor-log td').remove();
            $.each(journalRows.responseJSON, function(index) {
var journalContent = journalRows.responseJSON[index]['journalContent'];
var journalTime = journalRows.responseJSON[index]['journalTime'];

            table.append('<tr id=' + index + '><td>' + journalTime + '</td><td>' +
journalContent + '</td></tr>');
            });
        }
    });
}

function loadRule () {

```

```

$.ajax({
  url: "/rule",
  method: "GET",
  error: function(jqXHR) {
    var error = JSON.parse(jqXHR.responseText);
    $('#.error', form).html(error.message);
  },
  complete: function(rules) {
    console.log(rules.responseJSON);
    var table = $('#.rule');
    $('#.rule td').remove();
    $.each(rules.responseJSON, function(index) {
      var name = rules.responseJSON[index]['name'];
      var ping = rules.responseJSON[index]['ping'];
      var status = rules.responseJSON[index]['status'];

      table.append('<tr id=' + index + '><td>' + (index+1) +
        '</td><td>' + name + '</td><td>' + ping + ' mc</td><td></td></tr>');
    });
  }
});
</script>

```

Приложение В

ГРАФИЧЕСКИЕ МОДЕЛИ

ПАТТЕРНОВ УПРАВЛЕНИЯ

(рекомендуемое)

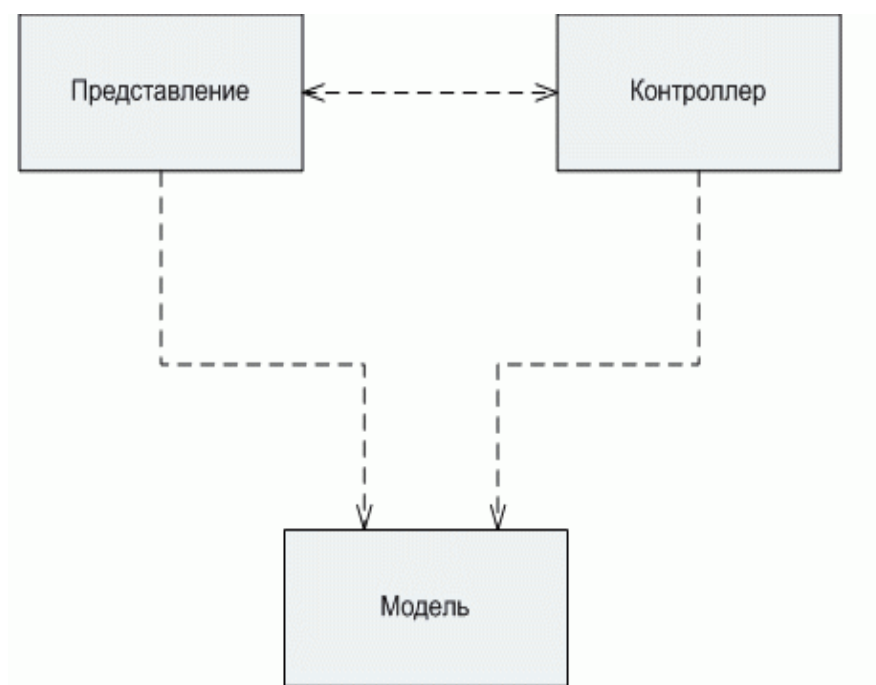


Рисунок В.1 – MVC-паттерн

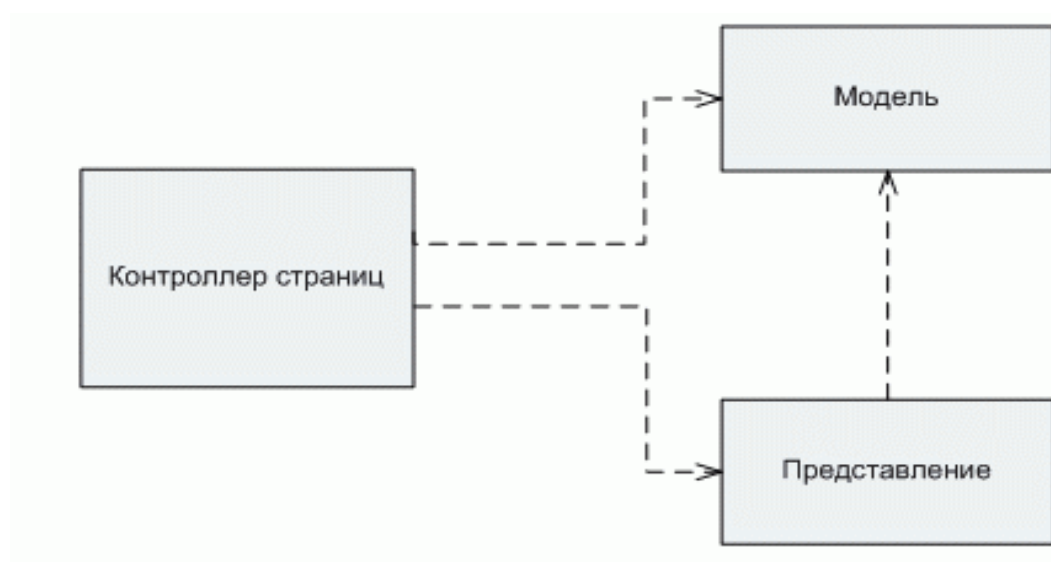


Рисунок В.2 – PS-паттерн

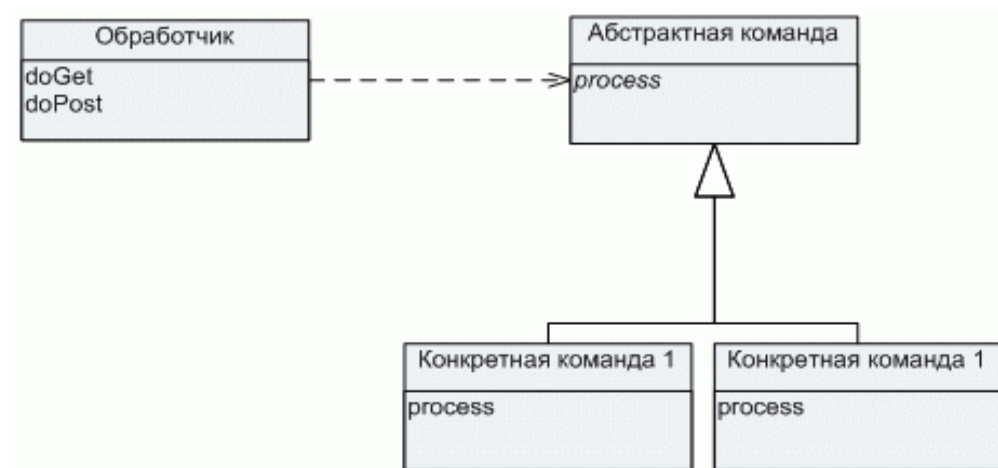


Рисунок В.3 – FC-паттерн

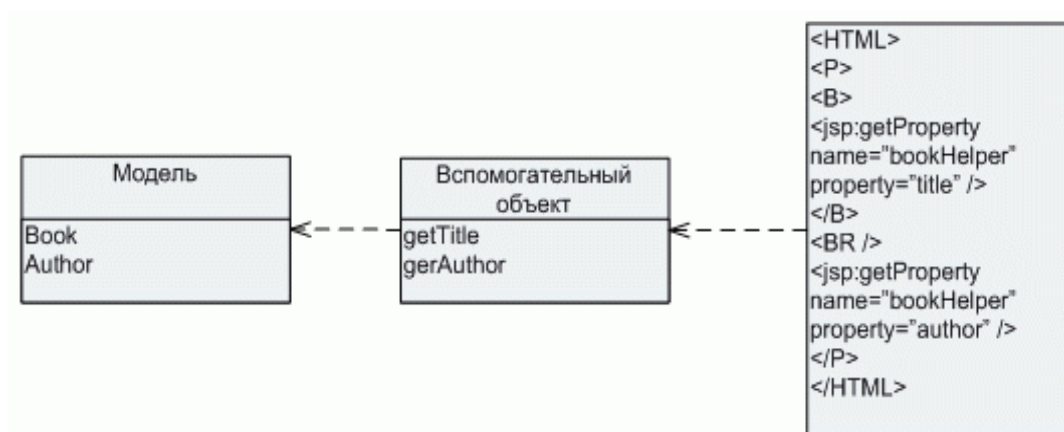


Рисунок В.4 – TV-паттерн

Приложение Г

**АДАПТИВНОСТЬ ПРИЛОЖЕНИЯ ПОД МОБИЛЬНЫЕ
УСТРОЙСТВА**

(рекомендуемое)

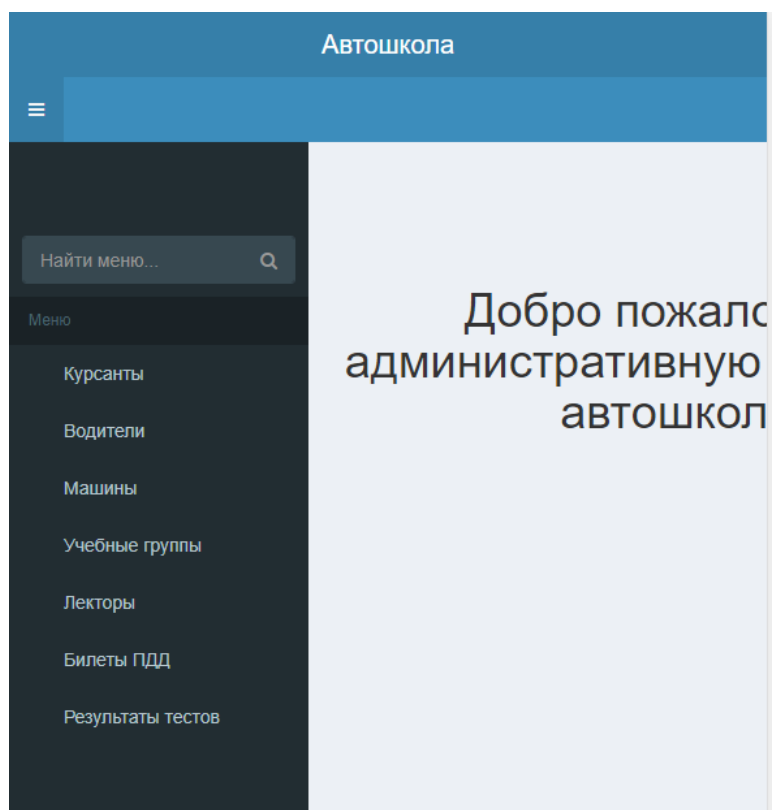


Рисунок Г.1 – Основная страница приложения

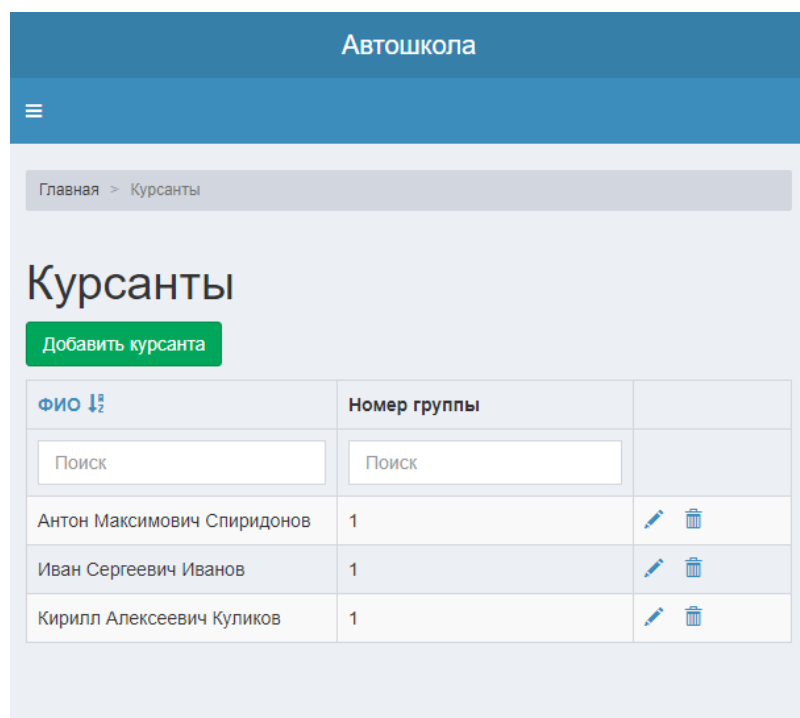


Рисунок Г.2 – Страница отображения списка курсантов

The screenshot shows the 'Автошкола' (Driving School) application interface. At the top is a blue header with the title 'Автошкола' and a hamburger menu icon. Below the header is a breadcrumb trail: 'Главная > Курсанты > Добавить курсанта'. The main heading is 'Добавить курсанта'. The form contains four input fields: 'ФИО' (empty), 'Номер группы' (dropdown menu with '1' selected), 'Номер телефона' (empty), and 'Дата рождения' (empty). A green 'Сохранить' (Save) button is at the bottom.

Рисунок Г.3 – Страница добавления курсанта

The screenshot shows the 'Автошкола' (Driving School) application interface for editing a student. At the top is a blue header with the title 'Автошкола' and a hamburger menu icon. Below the header is a breadcrumb trail: 'Главная > Курсанты > Антон Максимович Спиридонов > Редактировать'. The main heading is 'Редактировать курсанта: Антон Максимович Спиридонов'. The form contains four input fields: 'ФИО' (filled with 'Антон Максимович Спиридонов'), 'Номер группы' (dropdown menu with '1' selected), 'Номер телефона' (filled with '89542224848'), and 'Дата рождения' (filled with '2020-01-05'). A blue 'Сохранить' (Save) button is at the bottom.

Рисунок Г.4 – Страница изменение данных о курсанте

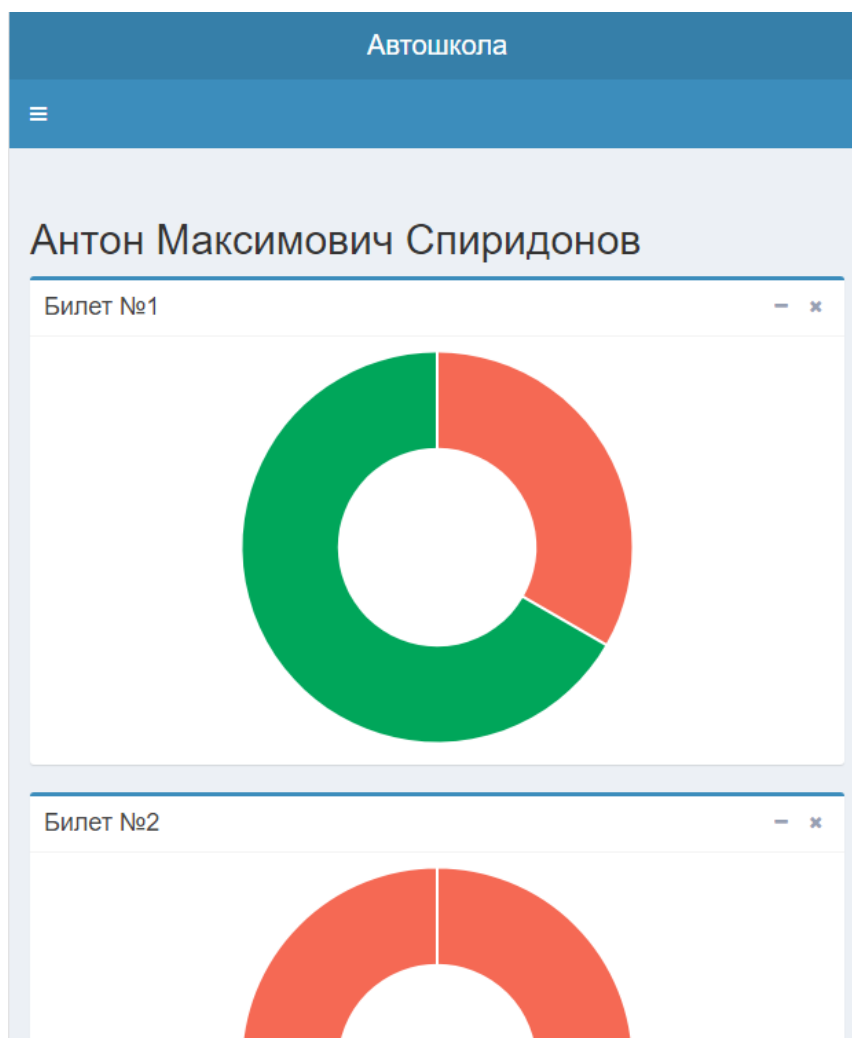


Рисунок Г.5 – Страница просмотра статистики прохождения тестов

Приложение Д

РЕЗУЛЬТАТЫ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ

(рекомендуемое)

Автошкола

Водите что бы начать пользоваться

8954888524

.....

Неверный логин или пароль

Войти

Рисунок Д.1 – Результат функционального теста №1

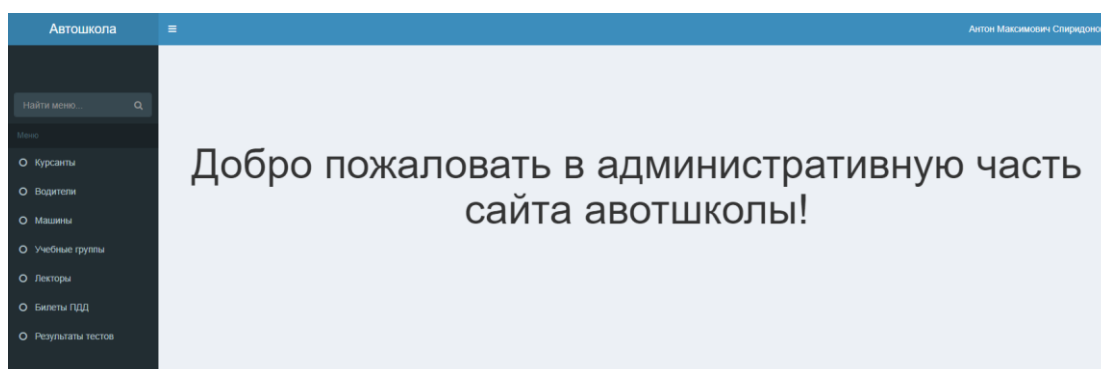


Рисунок Д.2 – Результат функционального теста №2

Главная > Курсы > Добавить курсанта

Добавить курсанта

ФИО

Необходимо заполнить «ФИО».

Номер группы

Выберите группу

Необходимо заполнить «Номер группы».

Номер телефона

Необходимо заполнить «Номер телефона».

Дата рождения

Необходимо заполнить «Дата рождения».

Сохранить

Рисунок Д.3 – Результат функционального теста №3

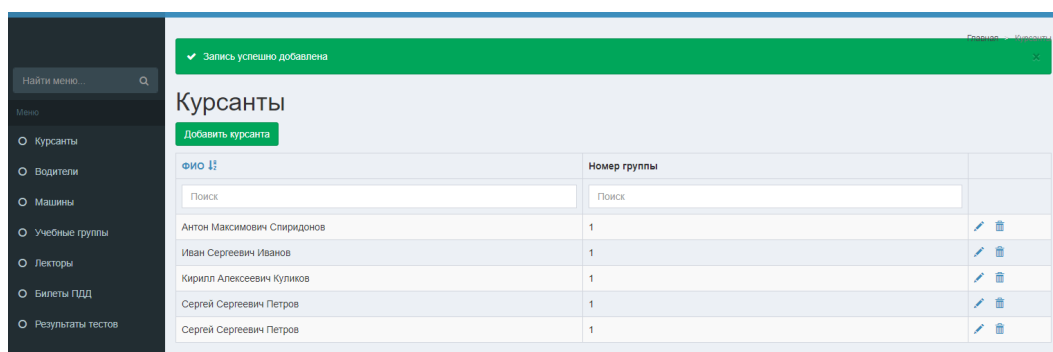


Рисунок Д.4 – Результат функционального теста №4

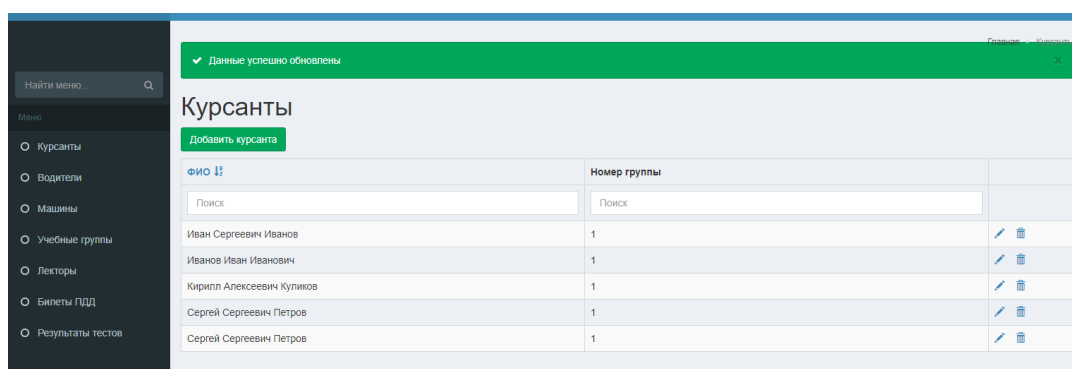


Рисунок Д.5 – Результат функционального теста №5

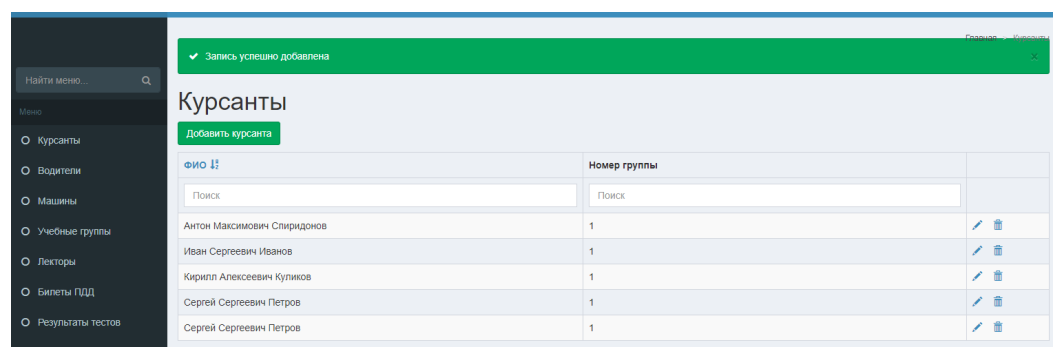


Рисунок Д.6 – Результат функционального теста №6

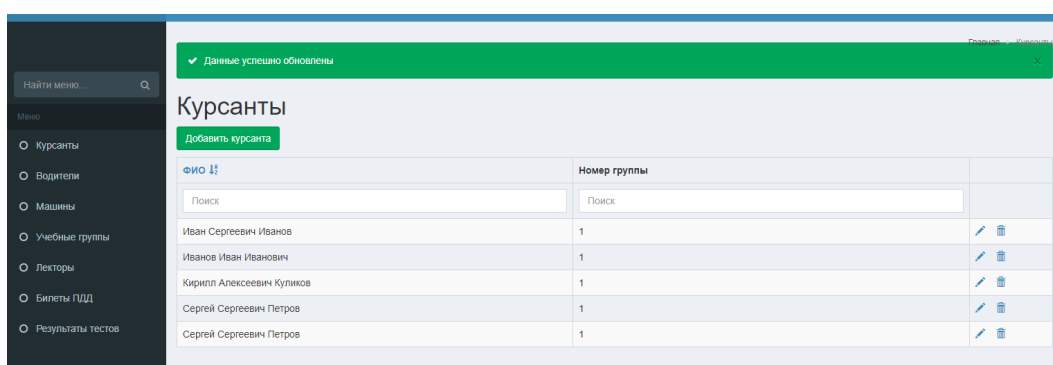


Рисунок Д.7 – Результат функционального теста №7

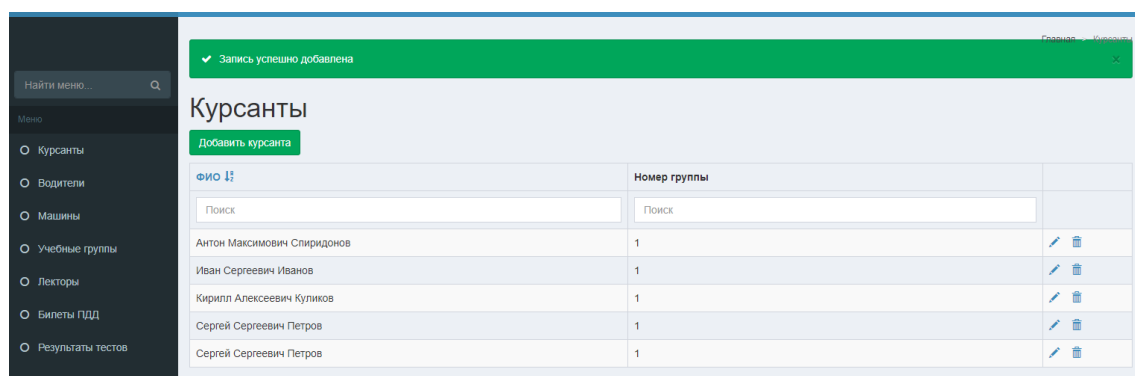


Рисунок Д.8 – Результат функционального теста №8

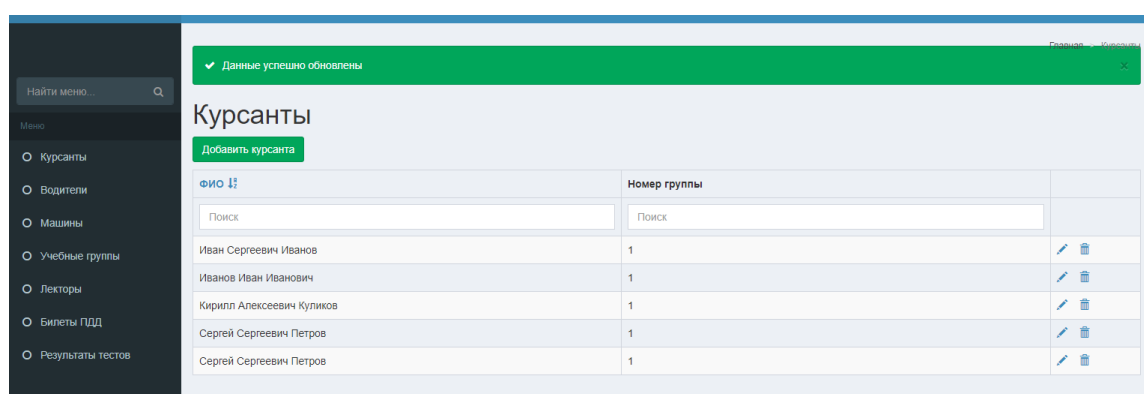


Рисунок Д.9 – Результат функционального теста №9

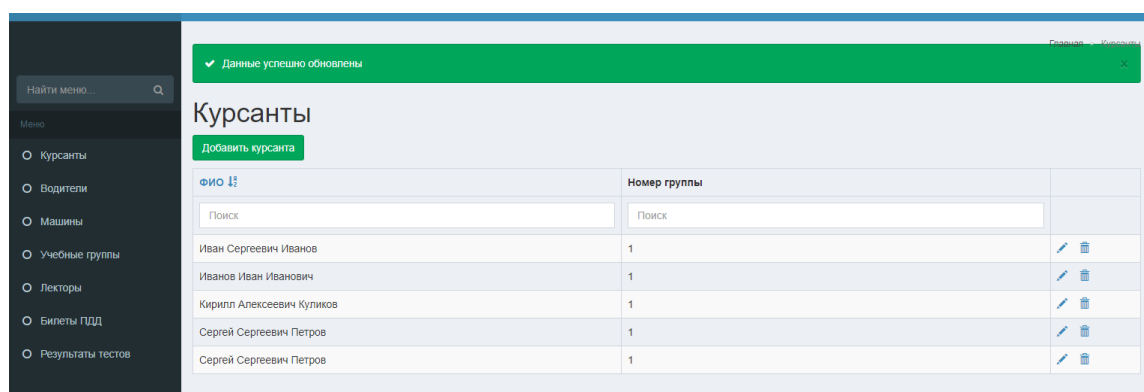


Рисунок Д.10 – Результат функционального теста №10