Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина Программирование сетевых приложений

К защите допустить

М.Н. Салапура

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УЧЕТА РАБОТЫ АВТОШКОЛЫ

БГУИР КП 1-40 01 02-02 028 ПЗ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | |  | | М.А. Хоруженко  гр. 573601 |
| Руководитель | |  | | М.Н. Салапура, ассистент кафедры  экономической информатики |
|  | |  | |  |
|  | |  | |  |
|  | |  | |  |
|  | |  | |  |
|  |  | |
|  |  | |

Минск 2017

**Содержание**

[Введение 5](#_Toc501424167)

[1 Описание деятельности автошколы 7](#_Toc501424168)

[2 Постановка задачи и обзор методов её решения 10](#_Toc501424169)

[3 Функциональное моделирование на основе стандарта IDEF0 12](#_Toc501424170)

[4 Информационная модель системы и её описание 14](#_Toc501424171)

[5 Описание алгоритмов реализующих бизнес-логику серверной части проектируемой системы 17](#_Toc501424172)

[6 Руководство пользователя 20](#_Toc501424173)

[7 Результаты тестирования разработаной системы 28](#_Toc501424174)

[Заключение 31](#_Toc501424175)

[Список использованных источников 33](#_Toc501424176)

[Приложение А (обязательное) Диаграмма вариантов использования 34](#_Toc501424177)

[Приложение Б (обязательное) Диаграмма последовательности 35](#_Toc501424178)

[Приложение В (обязательное) Диаграмма развертывания 36](#_Toc501424179)

[Приложение Г (обязательное) Функциональная модель IDEF0 37](#_Toc501424180)

[Приложение Д (обязательное) Листинг SQL-скрипта 39](#_Toc501424181)

[Приложение Е (обязательное) Блок-схемы алгоритмов 53](#_Toc501424182)

[Приложение Ж (обязательное) Диаграмма состояний 56](#_Toc501424183)

[Приложение З (обязательное) Диаграмма компонентов 57](#_Toc501424184)

[Приложение И (обязательное) Диаграмма классов 58](#_Toc501424185)

[Приложение К (обязательное) Листинг кода 62](#_Toc501424186)

# ВВЕДЕНИЕ

К настоящему времени информационные технологии плотно закрепились в жизни общества. Они используются практически во всех сферах деятельности человека, способствуют повышению производительности многих организаций и предприятий, а также облегчают выполнение многих задач, требующих точности и внимательности.

Быстрое развитие компьютерной индустрии, включая разработку программного обеспечения, привело к появлению множества автоматизированных систем различных направлений деятельности. С их помощью намного проще и удобнее производить учет и обработку данных, поэтому они очень востребованы и в процессе учета работы автошколы.

В связи с улучшением качества жизни и благосостояния людей в данный момент практически у каждого есть собственный автомобиль. Следовательно, для получения прав огромному количеству людей необходимо пройти обучение в автошколе. Поэтому учет данных о студентах автошколы процесс трудоемкий и является потенциальным источником большого количества человеческих ошибок.

Автоматизированная система учета работы автошколы позволяет хранить информацию о всех преподавателях и группах автошколы, а также о студентах и их текущей успеваемости. С ее помощью можно легко и быстро просмотреть данные о любом пользователе, что очень полезно и удобно при работе с большим потоком данных.

Целью данного курсового проекта является улучшение качества предоставляемого сервиса любой автошколы.

В процессе достижения поставленной цели будет разработано клиент-серверное приложение. Для его создания потребуется решение следующих задач:

* тщательно проанализировать процесс учетной деятельности автошколы;
* разработать алгоритмы работы;
* разработать функциональные модели по стандарту IDEF0;
* разработать диаграммы состояний для наиболее сложных процессов системы;
* создать базу данных, содержащую информацию о студентах, преподавателях и группах, придерживаясь стандартов;
* реализовать базовый функционал приложения, а именно, предоставить возможность регистрации и авторизации пользователей, просмотра, удаления и редактирования данных;
* предусмотреть разделение доступа управления программой, необходимо наличие трех ролей: администраторы, преподаватели, студенты;
* разработать необходимый функционал для каждого типа пользователей;
* протестировать программные модули, в случае необходимости их доработать;

Объект исследования данного курсового проекта – функциональная система автошколы, содержащая информацию о студентах и их успеваемости, позволяющая автоматизировать процесс формирования групп и назначение им преподавателей, а также процесс выставления зачетов по тестам студентам.

1. **ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

Предметной областью является автошкола, предоставляющая возможность получения водительских прав различных категорий и услуги по обучению вождению.

Автошкола – учебное заведение специального образования, предназначенное для подготовки водителей. В автошколе ведется подготовка водителей к сдаче экзаменов на право управления транспортными средствами. Экзамены на право управления транспортными средствами каждой категории сдаются отдельно [1].

Национальное водительское удостоверение должно соответствовать требованиям Конвенции о дорожном движении, согласно которым удостоверение печатается на языке или языках, установленных органом власти, который выдает или которому поручено выдавать это удостоверение. Национальное водительское удостоверение признается почти во всех странах Европы, так как здесь подписана Конвенция о дорожном движении. В странах же, не подписавших данных документ, для получения права управления транспортным средством необходима повторная сдача экзаменов.

Первыми в мире водительскими правами стала «справка о способности управлять транспортным средством с механическим мотором», вручённая 14 августа 1893 года префектом полиции Парижа Луи Лепином ныне неизвестному французскому автомобилисту. Тогда подобные справки выдавались без каких-либо экзаменов, для их получения кандидат был должен лишь продемонстрировать умение управлять собственным автомобилем [2].

В базе хранится информация о каждом студенте, а именно, его фамилия, имя, отчество и номер телефона, данные об учителях и группах. Администратор автошколы следит за наполняемостью групп, распределяет новых студентов по группам, назначает группам преподавателей. Группы в свою очередь различаются категория обучения, таких категорий существует 4:

* «А» – обучение вождению на мотоциклах, начинать обучение разрешено с шестнадцати лет.
* «В» – обучение вождению на легковых автомобилях разрешённой максимальной массой до 3500 тонн, начинать обучение можно с 18 лет.
* «С» – Автомобили свыше 3500 тонн, обучение можно начинать с 18 лет.
* «D» – Автомобили предназначенные для перевозки пассажиров, обучение можно начинать при наличии стажа вождения не менее трех лет [2].

Рассмотрим основные сущности, присутствующие в системе.

Группа – сущность, описывающая существующие в автошколе группы, в которые происходит набор студентов. Основные атрибуты данной сущности:

− уникальный номер группы;

− категория обучения;

− количество студентов, которых можно зачислить в группу;

− дата начала занятий;

* дата окончания занятий, высчитывается автоматически, обучение длится три месяца.
* преподаватель, проводящий занятия и тестирование студентов группы.

Преподаватель – сущность, хранящая информацию о преподавателях автошколы. Содержит следующие атрибуты:

− уникальный идентификатор;

− ФИО преподавателя;

− телефон преподавателя, также уникальный;

− опыт работы;

Студент – сущность, содержащая информацию о студентах, проходящих обучение в автошколе. Для ее описания используются следующие атрибуты:

− уникальный идентификатор;

− ФИО студента;

− номер телефона студента;

− номер группы, в которую студент был зачислен для прохождения обучения.

Тест – сущность, содержащая информацию о тестах, которые должен пройти каждый студент для допуска к экзамену. Характеризуется двумя атрибутами:

− номер главы, по которой проводиться тест;

− тема теста.

Процесс зачисления новых студентов в группы происходит следующим образом. Человек, желающий пройти обучение в автошколе связывается с администратором автошколы, который в свою очередь регистрирует нового студента, зачисляет его в определенную группу при наличии в ней свободных мест, в соответствии с требованиями клиента. Если подходящей группы не найдено, то её необходимо добавить и назначить ей преподавателя из базы. Преподаватели в системе регистрируются самостоятельно, имеют возможность просматривать информацию о своих группах и о студентах каждой группы. В течении обучения выставляют студентам зачеты по прохождению тестов.

Помимо лекционной части обучения в автошколах существует также практическая часть. Теоретические занятия проводятся в составе группы, занятия по вождению проводятся индивидуально с каждым студентом сначала на закрытой площадке для учебной езды, затем по учебным маршрутам, утвержденным местными органами ГАИ. Допуск к сдаче экзаменов на получение водительского удостоверения в ГАИ производится только при наличии зачетов по всем тестам, а также после прохождения сорока часов практических занятий. Также перед сдачей экзамена в ГАИ необходимо сдать внутренний экзамен в автошколе.

Внутренний экзамен состоит из нескольких этапов: теоретическая часть, вождение на учебной площадке с выполнением всех обязательных элементов, вождения в условиях городского движения. Допуск к прохождению следующего этапа учащийся получается только при успешном прохождении предыдущего.

Для более эффективного и быстрого распределения учащихся по группам, а также для обеспечения возможности легко и удобно просматривать уровень их текущей успеваемости целесообразно создать программное приложение, с помощью которого будет обеспечена работа с базой данных, хранящей всю полезную информацию о учащихся, преподавателях и группах автошколы.

При внедрении автоматизированной системы автошколы получат множество преимуществ, таких как:

− уменьшение количества человеческих ошибок;

− экономия времени при поиске подходящей группы;

− оперативное зачисление в группы;

− возможность просмотра статистики успеваемости учащихся.

1. **ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ**

Целью курсового проекта является повышение эффективности работы автошкол.

Для достижения данной цели в ходе выполнения курсового проекта будет разработано клиент-серверное десктопное приложение, с помощью которого администраторы смогут легко и оперативно регистрировать новых учащихся, преподавателям будет предоставлен удобный интерфейс для работы с каждой и группой, а также с каждым отдельно взятым учащимся. У учащихся же будет возможность просмотра своей текущей успеваемости и других полезных данных.

Разрабатываемая программа будет включать в себя два модуля: клиентское приложение и серверное приложение.

Программное приложение будет реализовано на языке программирования Java. Java – объектно-ориентированный язык, является достаточно сильно типизированным. Приложения языка при выполнении транслируются Java-машиной в специальный байт-код, что позволяет таким приложения работать на любой компьютерной архитектуре.

Серверное приложение будет реализовано в виде консольного приложения. В консоли будет отображаться вся информация о текущей работе сервера, параметры текущего соединения, данные о авторизовавшихся пользователях и операциях, которые они производят.

Пользоваться разрабатываемой системой потенциально будет большое число пользователей, значит, логично предположить, что в какой-либо момент возможна ситуация попытки подключения к серверу более одного клиента. В следствие этого было принято решение разработки многопоточного серверного приложения. Оно будет в состоянии одновременно обрабатывать совершенно разные запросы от администраторов, преподавателей и учащихся, такие как:

* создание новой учетной записи;
* авторизация в системе;
* добавление новой группы в базу данных;
* редактирование различной информации;
* выставление зачетов учащимся;
* просмотр диаграммы успеваемости.

Доступ к базе данных будет производиться на серверной части на основе стандарта JDBC (стандарт взаимодействия приложения с различными СУБД). Данный стандарт основан на концепции драйверов, дающих возможность получить доступ к базе данных через специальный url. Для разработки самой базы данных будет использоваться СУБД MySQL. Это свободная реляционная система управления базами данных, данная СУБД отлично подходит для небольших и средних программных приложений. MySQL является одной из самых популярных систем управления базами данных, благодаря своей высокой производительности, надежности и удобному интерфейсу. Содержит множество различных типов данных и всевозможных видов таблиц.

Клиентская часть приложения будет разработана в виде GUI-приложения. Разрабатываемый интерфейс будет прост и понятен для любого пользователя, будет предусмотрено наличие окон-подсказок при выполнении различных операций, в случае ошибок или некорректно введенных данных также будут появляться предупреждающие окна. Через оконный интерфейс пользователям системы будет предоставлен полный функционал работы с приложением в зависимости от их роли.

Подробнее рассмотрим задачи, которые может выполнять администратор (рисунок А.1, Приложение А). Администратор имеет доступ ко всем данным системы и может производить функции добавления, удаления и редактирования этих данных.

Функционал учителя включает следующие возможности: просмотр, редактирование данных о себе и о своих группах и студентах этих групп.

В ходе написания оконного приложения будут использованы стандартные библиотеки пользовательского интерфейса SWING и AWT. AWT – самая ранняя библиотека пользовательского интерфейса, в следствие чего в ней много недоработок и устаревших деталей. По сути, AWT – это просто прослойка, написанная на языке Java, которая вызывает нативные методы из библиотек других языков. Из-за этого нельзя с точностью сказать, как будет выглядеть интерфейс программы при запуске её на другой платформе. Кроме того, в AWT достаточно скудное разнообразие компонентов и функционала, нет таблиц, в кнопках не поддерживается отображение иконок. Использованные ресурсы AWT старается освобождать автоматически. Это немного усложняет архитектуру и влияет на производительность.

SWING же более новый набор графических компонентов, построенных на основании базовых компонентов AWT. SWING превосходит AWT по разнообразию и функциональности, для отрисовки компонент используется 2D. Предоставлена большая иерархия классов, с помощью которых можно легко и быстро создавать нужные компоненты.

1. **ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0**

Функциональная модель была разработана для процесса регистрации в системе нового учащегося. Для создания модели данный процесс сначала был тщательно изучен и проанализирован. Использовался стандарт IDEF0

Для рассмотрения был взят процесс управления новым рейсом администратором. Соответственно, необходимо смоделировать функциональную модель управления рейсами компании, на основе стандарта IDEF0. IDEF0 – нотация описания бизнес-процессов. Основана на методологии SADT.

Рассмотрим входные и выходные параметры, а также механизм выполнения бизнес-процесса и управляющие элементы по отношению к процессу регистрации в системе нового пользователя.

Для того, чтобы зарегистрировать кого-то в системе, кто-то должен обраться к администратору с такой просьбой. То есть на вход мы получаем некоторого человека, который желает пройти обучение в автошколе.

К потокам управления можем отнести:

* нормативные акты, регламентирующие деятельность по подготовке водителей транспортных средств;
* базу действующих групп;
* закон о лицензировании образовательных учреждений.

Заниматься выполнением операции регистрации учащихся будет администратор автошколы. Значит, механизм бизнес-процесса – администратор.

На выходе получим нового учащегося автошколы, зарегистрированного в системе.

Далее осуществим декомпозицию блока «Регистрация нового пользователя». Данный процесс можно представить в виде совокупности четырех подпроцессов (рисунок Г.2, Приложение Г):

* запросить необходимую информацию о группах от сервера, исходя из требований клиента;
* получить информацию от сервера;
* проанализировать полученные данные;
* добавить нового учащегося в базу данных.

Каждый из подпроцессов также имеет свои входные, управляющие, выходные, механизмы выполнения.

В свою очередь, процесс добавления нового учащегося можно также декомпозировать. Он состоит из следующих процессов (рисунок Г.3, Приложение Г):

* согласовать условия добавления с клиентом;
* сформировать запрос с информацией о потенциальном учащемся;
* отправить запрос на добавление серверу;
* получить подтверждение добавления от сервера.

Как и в предыдущем случае, каждый из перечисленных процессов имеет входные, выходные потоки, а также потоки управления и механизмы выполнения.

Формирование запроса с информацией о потенциальном учащемся включает в себя следующие этапы (рисунок Г.4, Приложение Г):

* внести в форму личные данные клиента;
* указать номер группы для зачисления из списка возможных;
* проверить корректность внесенных данных.

Здесь на входе мы получаем согласованные данные для добавления, а на выходе полностью сформированные данные о клиенте.

Функциональная модель помогает досконально изучить процесс регистрации нового пользователя и все его подпроцессы, вплоть до элементарных. С помощью такой диаграммы можно без труда разобраться во всех нюансах данного процесса и затем поэтапно запрограммировать его.

1. **ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ**

База данных была разработана с помощью сервера MySQL 5.5. Для визуального проектирования использовался инструмент MySQL Workbench версии 6.3 редакции Community Edition, которая распространяется под свободной лицензией. На рисунке ниже приведена ER диаграмма разработанной базы данных. На ней расположено 6 таблиц, указаны связи между ними.

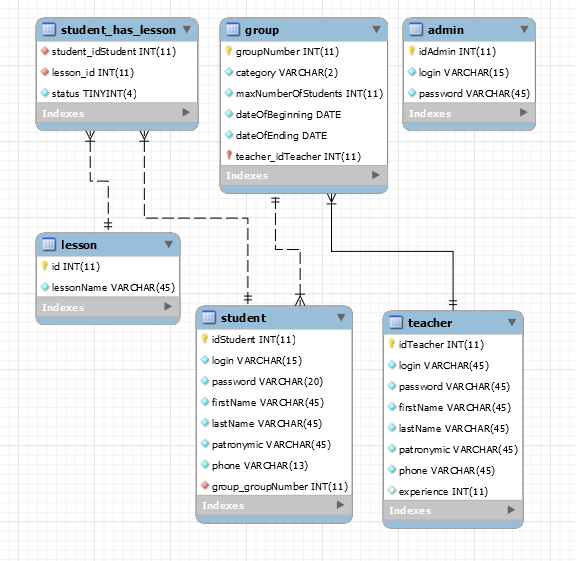


Рисунок 4.1 – ER диаграмма базы данных “drivingschool”

Рассмотрим более детально таблицы, представленные на диаграмме.

Таблица admin предназначена для хранения данных о всех администраторах, зарегистрированных в системе. Содержит следующие поля:

* idAdmin – уникальный идентификатор;
* login – логин администратора, должен иметь уникальное значение;
* password – пароль для подтверждения личности при авторизации.

Каждое поле данной таблицы должно обязательно содержать какое-либо значение, иначе запись просто не будет добавлена. Это требование реализуется при помощи выставления каждому полю специального идентификатора NOT NULL.

В таблице student будут храниться данные обо всех учащихся, которых зарегистрирует администратор. Учащимся при регистрации также присваивается определенный уникальный логин и пароль, с использование которых учащиеся смогут авторизоваться в системе. Помимо этого, в таблице присутствуют следующие свойства:

* idStudent – уникальный идентификатор студента, для быстрого поиска в базе данных;
* lastName – фамилия учащегося;
* firstName – имя учащегося;
* patronymic – отчество учащегося;
* phone – номер телефона учащегося, полю выставлен атрибут UQ, то есть оно должно быть уникальным;
* groupNumber – номер группы, в которую зачислен студент.

Таблица student связана с таблицей group, которая будет описана ниже. Связь 1:M (в одной группе может обучаться много студентов). Свойство groupNumber таблицы student служит является внешним ключом, служит для связи таблиц.

Таблица teacher создана для хранения информации о преподавателях нашей системы. Преподаватели могу как самостоятельно регистрироваться в системе, так и быть добавленными в систему администратором. В любом случае они имеют уникальный логин, а также секретный пароль для авторизации. Также в таблице присутствуют поля:

* idTeacher – уникальный идентификатор учителя, для быстрого поиска в базе данных;
* lastName – фамилия преподавателя;
* firstName – имя преподавателя;
* patronymic – отчество преподавателя;
* phone – номер телефона преподавателя, полю выставлен атрибут UQ, то есть оно должно быть уникальным;
* experience – опыт работы преподавателя.

Данные обо всех действующих группах содержаться в таблице group. В ней присутствуют следующие свойства:

* groupNumber – уникальный номер группы;
* category – категория водительских прав, которые будут получены после успешного прохождения курсов вождения (A,B,C или D);
* maxNumberOfStudents – максимальное число студентов, которые могут быть зачислены в данную группу;
* dateOfBeginning –дата начала занятий группы;
* dateOfEnding – дата окончания занятий в группе, в программе будет реализован автоматический подсчет данного свойства.

Таблица group связана с таблицами student и teacher. Связь с таблицей teacher – 1:М (1 преподаватель может вести занятия у нескольких групп). Связь осуществляется посредством поля idTeacher. Связь с таблицей student была описана выше.

Таблица lesson хранит данные о тестах, которые необходимо пройти каждому студенту для допуска к внутреннему экзамену в атвтошколе. Таблица содержит следующие свойства:

* id – номер теста, уникальный;
* lessonName – наименование теста, также уникальное.

Таблица lesson связана с помощью связи M:M с таблицей student (каждому студенту соответствует несколько тестов, каждый тест должен быть выполнен всеми студентами). Связь М:М реализуется с помощью вспомогательной таблицы student\_has\_lesson. Эта таблица хранит id студента, выполнившего тест, id теста, который был выполнен, а также статус результата тестирования (зачет, незачет).

Разработанная база данных находиться в первой нормальной форме, так как все атрибуты атомарны, то есть ни один из них нельзя разделить на более простые атрибуты, которые соответствуют каким-то другим свойствам описываемой сущности (например, атрибут ФИО не является атомарным, так как может быть разбит на более мелкие атрибуты. Однако, в некоторых системах такой атрибут может быть уместен). База данных находится во второй нормальной форме, так как она находится в первой нормальной форме (это было доказано выше), а также каждый неключевой атрибут, то есть любой столбец, который не является ключом, в том числе и внешним, зависит от первичного ключа [3]. База данных находится в третьей нормальной форме, так как она находится во второй нормальной форме и в таблицах нет неключевых полей, зависящих от значения других неключевых полей [4].

Листинг скрипта генерации базы данных приведен в Приложении Д.

1. **ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ**

Подробно рассмотрим алгоритм удаления преподавателя из базы данных. Блок-схема данного алгоритма представлена в Приложении Е. Код реализующий данный метод:

dao = new TeacherDAO(connection);  
  
String loginTeacher = myMessage.getTeacher().getLogin();  
dao = new TeacherDAO(connection);  
Teacher teacher = (Teacher)dao.findEntityByLogin(loginTeacher);  
if(teacher!=null) {  
 int idTeacher = teacher.getIdTeacher();  
  
 dao = new GroupDAO(connection);  
 if (dao.findByIdTeacher(idTeacher) != null) {  
 myMessage.setMessageType(MessageType.*TEACHER\_HAS\_GROUPS*);  
 return MessageType.*TEACHER\_HAS\_GROUPS*;  
 }  
}  
else{  
 dao = new TeacherDAO(connection);  
 dao.delete(myMessage.getTeacher().getLogin());  
 myMessage.setMessageType(MessageType.*SUCCESSFUL*);  
 return MessageType.*SUCCESSFUL*;  
}  
return MessageType.*DEFAULT*;

Изначально из присланного от администратора сообщения извлекается логин преподавателя, которого требуется удалить из базы данных. Затем производится проверка существования преподавателя с таким логином в базе данных, если такого преподавателя не существует, удалять из базы данных нечего, поэтому просто возвращаем администратору сообщение по умолчанию. Если же все хорошо и преподаватель с таки логином найдет, производим следующую проверку. Извлекаем из найденной по логину в базе данных записи о преподавателе id этого преподавателя. Изначально id нам было неизвестно, можем его извлечь только из базы данных. Затем проверяем в базе групп, соответствует ли данный id преподавателя записи о какой-либо группе. Если да, значит удаление производить нельзя, пока не передадим группы данного преподавателя кому-либо другому. Возвращаем администратору сообщение о том, что у данного преподавателя есть группы. Если же групп у преподавателя нет, может спокойной вызывать запрос на удаление преподавателя из базы данных преподавателей по логину, возвращаем администратору сообщение об успешном выполнении операции.

Рассмотрим детально алгоритм редактирования данных о преподавателе, диаграмма состояний для данного процесса представлена в Приложении Ж. Ниже приведен фрагмент кода программы, отвечающий за редактирование данных о преподавателе:

Teacher teacher;  
teacher = myMessage.getTeacher();  
String ourphone = teacher.getPhone();  
myMessage = new MyMessage();  
  
dao = new TeacherDAO(connection);  
if (!dao.checkPhoneAndId(teacher.getId(), teacher.getPhone())) {  
 teacher = (Teacher) dao.update(teacher);  
 if (teacher != null) {  
 myMessage.setMessageType(MessageType.*REDACT\_SUCCESS*);  
 return myMessage;  
 }  
}  
  
else {  
 dao = new StudentDAO(connection);  
 if (dao.checkPhone(ourphone)) {  
 dao = new TeacherDAO(connection);  
 if(dao.checkPhone(ourphone)) {  
  
 teacher = (Teacher) dao.update(teacher);  
 if (teacher != null) {  
 myMessage.setMessageType(MessageType.*REDACT\_SUCCESS*);  
 return myMessage;  
 }  
 }  
 }  
}  
myMessage.setMessageType(MessageType.*SAME\_PHONE\_NUMBERS*);  
return myMessage;

Так как телефоны всех пользователей должны быть уникальны, значит изначально мы должны проверить, удовлетворяют ли внесенные данные этому условию. Извлекаем из полученного сообщения номер телефона для редактирования и id преподавателя, данные о котором редактируются. Сначала делаем запрос к базе данных учителей и проверяем, не является ли номер телефона номер редактируемого преподавателя, так как вполне возможно, что преподаватель изменил какие-либо другие данные о себе, а номер телефона оставил прежним. Значит, если телефон принадлежит текущему преподавателю, все хорошо и можно осуществлять редактирование базы данных, вернув при этом пользователю сообщению об успешном редактировании. Если же номер телефона не принадлежит текущему преподавателю, производим проверку уникальности телефона в базах преподавателей и студентов. В случае неуникальности телефона, возвращаем сообщение об этом и изменение в базе данных не производим. В обратном случае изменяем данные в базе данных и возвращаем сообщение об успешном выполнении операции.

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Для запуска приложения сначала нужно сгенерировать с помощью SQL скрипта необходимую базу данных, если она не была создана до этого. Если требуется изменить настройки подключения к базе данных в конфигурационном файле. Затем запустить сначала серверную часть, затем клиентскую.

В результате появляется окно главного меню с тремя кнопками: вход в систему, регистрация в системе, выход. Результат запуска программы представлен на рисунке 6.1.

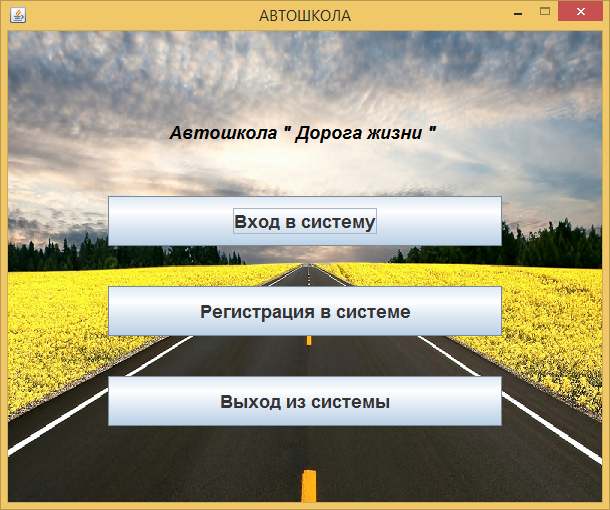


Рисунок 6.1 – Главное меню

При нажатии на кнопку «Регистрация в системе» появится диалоговое окно с формой для заполнения (рисунок 6.2). Туда необходимо внести требуемую информацию. На окне есть две кнопки: зарегистрировать и отмена. При нажатии кнопки «Зарегистрировать» снова появится окно главного меню, сверху будет показан результат регистрации.

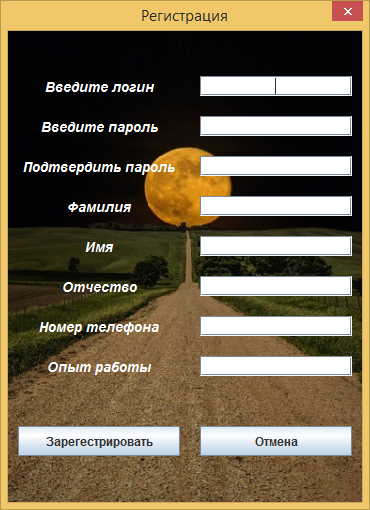


Рисунок 6.2 – Окно регистрации

Теперь рассмотрим пункт главного меню «Вход в систему». Так как в разработанном приложении существует три роли: администратор, преподаватель, учащийся, то в зависимости от введеного логина и пароля высветится одно из следующих окон: меню администратора, менюпреподавателя или меню учащегося. Сначала зайдем под администратором и изучим его функционал. Администратор может работать со всеми данными автошколы, выбирая соответствующую кнопку меню. Меню администратора приведено на рисунке 6.3.

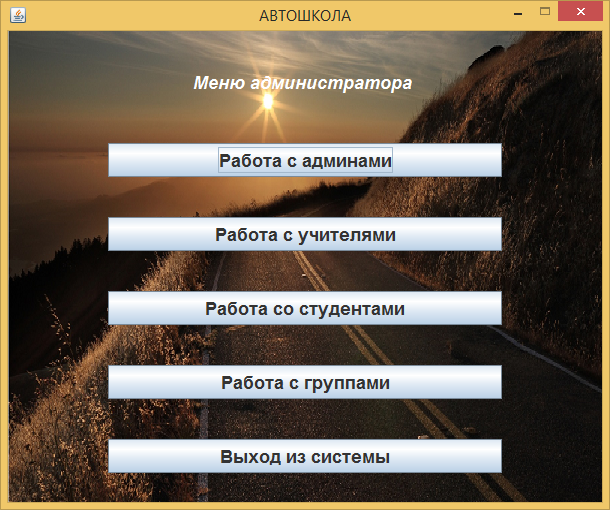


Рисунок 6.3 – Меню администратора

При нажатии на кнопку «Работа с админами» появляется окошко с информацией обо всех администраторах системы (рисунок 6.4). Информация представляется в табличной форме. Внизу окна расположены четыре кнопки: «Добавить», «Удалить», «Редактировать», «Назад». Перед нажатием кнопок «Редактировать» и «Удалить» сначала необходимо выделить строку таблицы, данные которой будут изменяться. Аналогичные окна появятся и при нажатии остальных кнопок главного меню администратора, отличие будет лишь в том, что в окнах будет выводиться информация об преподавателях, студентах или группах.

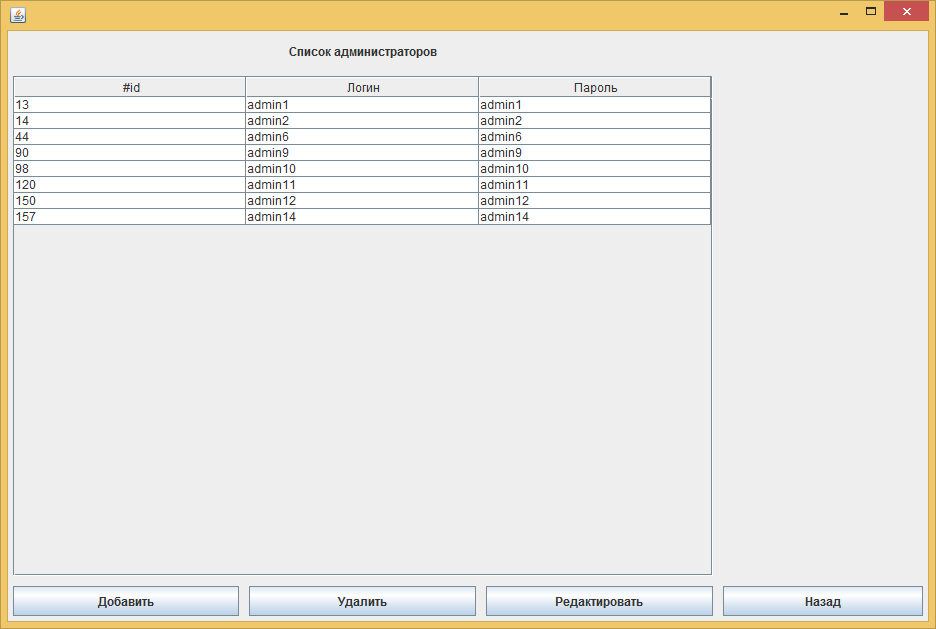


Рисунок 6.4 – Окно работы с администраторами

Теперь рассмотрим меню преподавателя, которое откроется при вводе логина и пароля преподавателя. Данное меню представлено на рисунке 6.5.

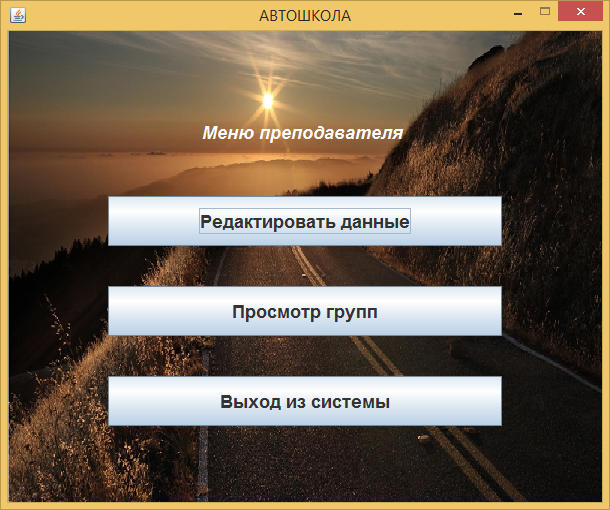


Рисунок 6.5 – Меню преподавателя

При нажатии на кнопку «Просмотр групп» высветится окно с информацией обо всех группах преподавателя (рисунок 6.6).

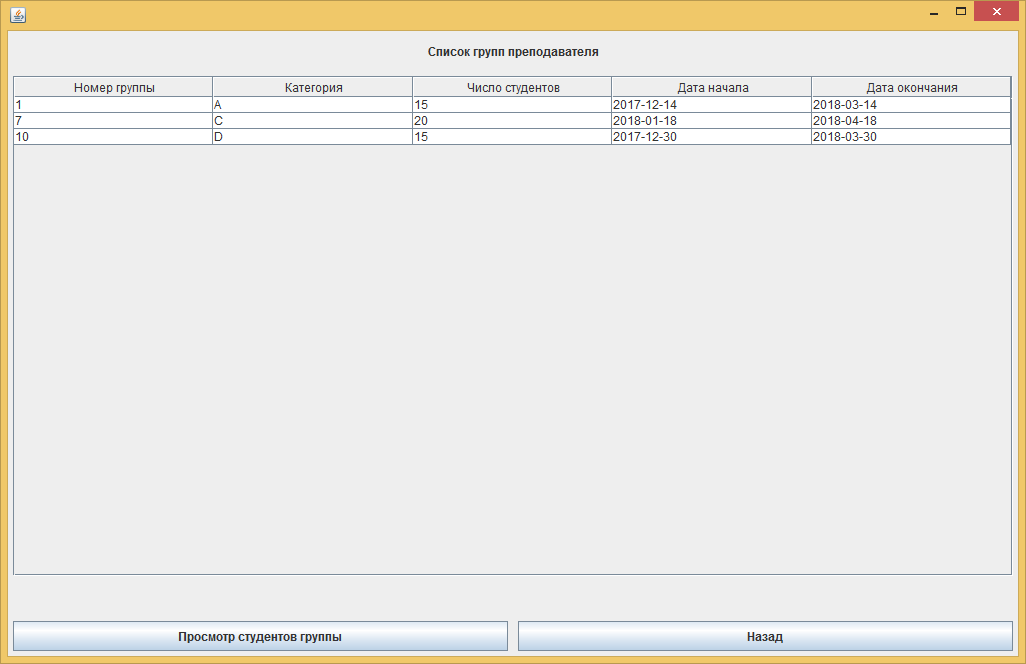


Рисунок 6.6 – Окно списка групп преподавателя

По выборе определенной строкb и нажатии на кнопку «Просмотр студентов грппы» высветиться окно с информацией о студентах выбранной группы (рисунок 6.7).

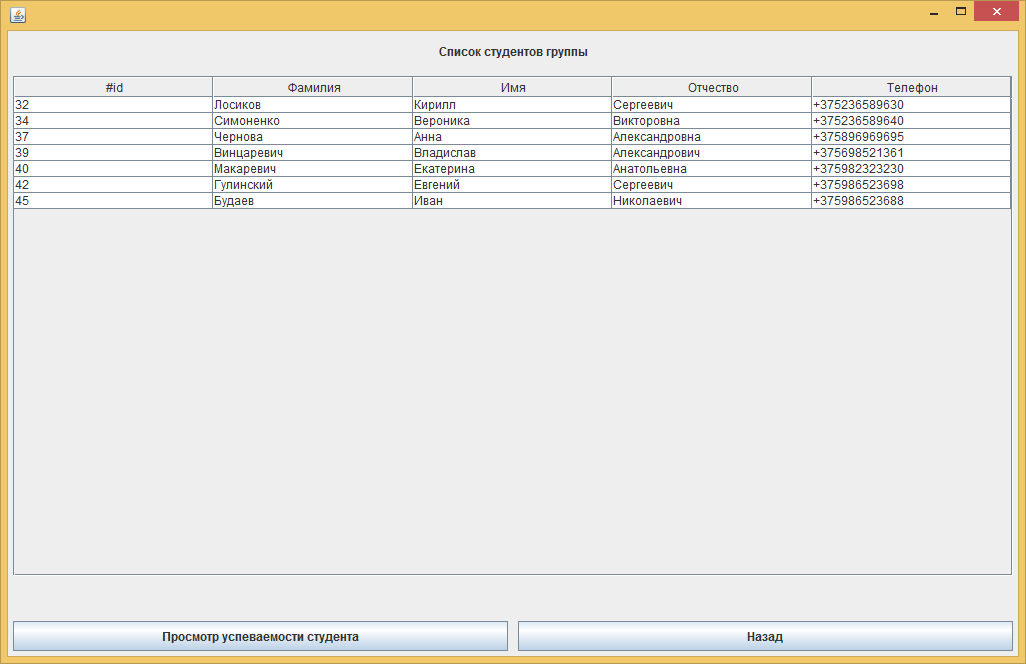


Рисунок 6.7 – Окно со списком студентов определенной группы

При нажатии на кнопку «Просмотр успеваемости студента» перейдем в окно, где будут указаны все зачеты и незачеты текущего студента по всем тестам. Это окно представлено на рисунке 6.8. На окне есть кнопка «Поставить зачет», с помощью которой можно изменить статус теста на зачет, обратное выполнить нельзя. Также есть кнопка «Просмотр диаграммы успеваемости», при нажатии на которую появится диаграмма, на которой отражено общее количество студентов автошколы, сдавших зачет по определенному предмету к текущему моменту. Диаграмма представлена на рисунке 6.9.

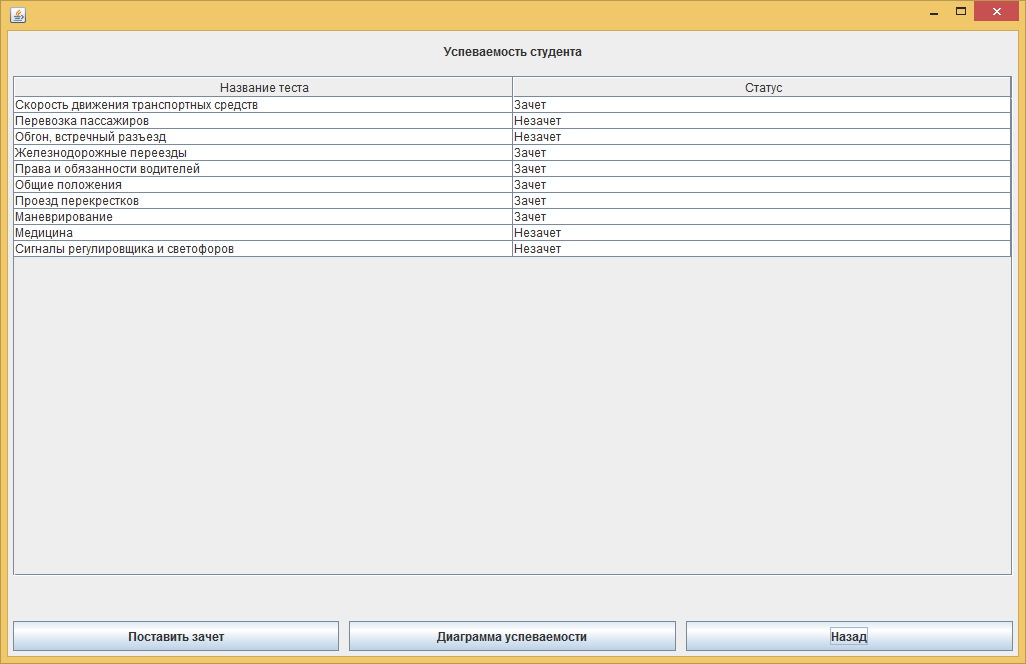


Рисунок 6.8 – Окно успеваемости студента

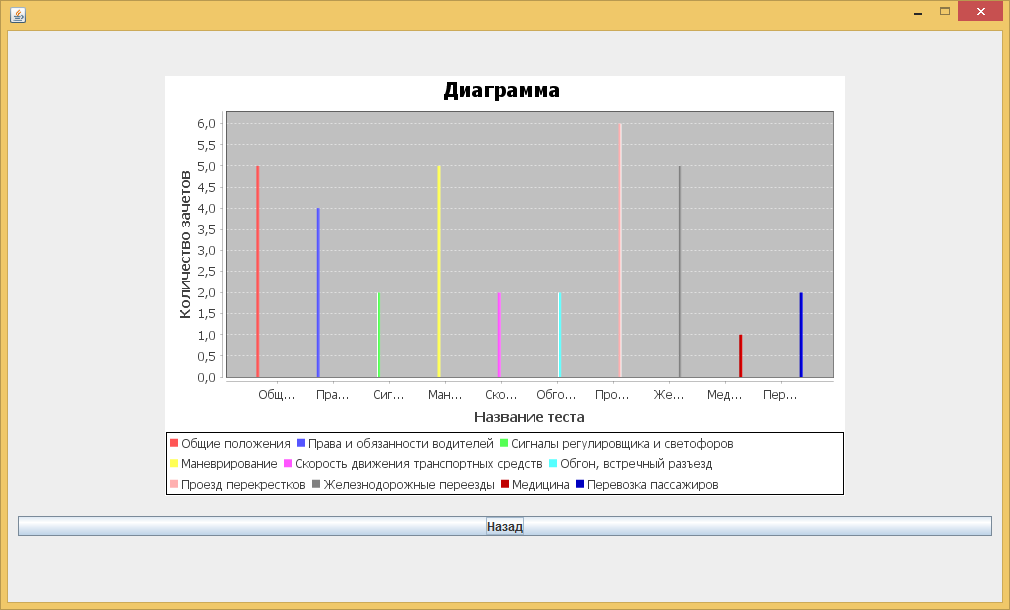


Рисунок 6.9 – Диаграмма успеваемости

При входе под учетной записью учащегося появится меню студента (рисунок 6.10)

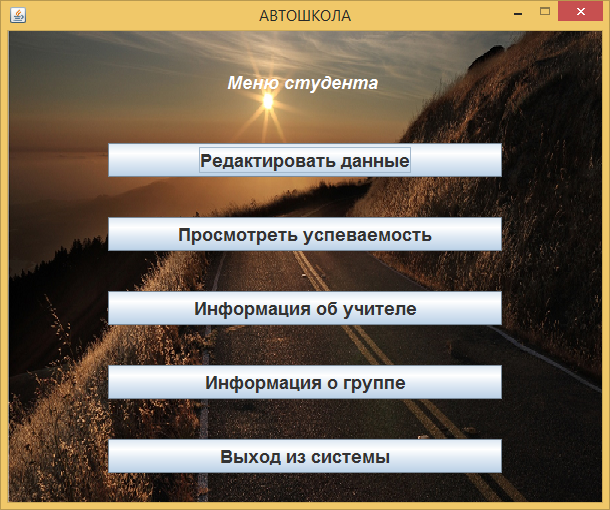


Рисунок 6.10 – Меню студента

При нажатии кнопки «Редактирование данных» появится диалоговое окно с уже заполненной старыми данными о текущем учащемся формой. Учащийся может изменить любое поле и нажать кнопку «Редактировать». На окне появится сообщение о результате редактирования. Результат нажатия на кнопку «Редактирование данных» представлен на рисунке 6.11.

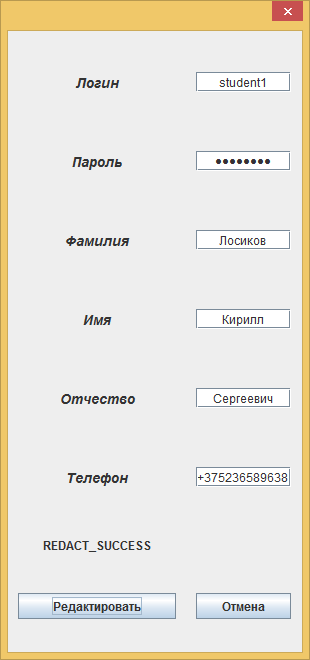


Рисунок 6.11 – Диалоговое окно редактирования данных

Остальные кнопки меню студента аналогичны описаным выше.

1. **РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАНОЙ СИСТЕМЫ**

Разработанное приложение было тщательно протестировано не только при благоприятных условиях и вводе корректных данных, но и в случаях исключительных ситуаций.

Например, если при регистрации или редактировании данных пользователем будут введены некорректные данные, он будет предупрежден об этом. Поле с неправильно введенными данными подсветится красным цветом. На рисунке 7.1 можно увидеть, что при попытке ввода имени с маленькой буквы, у поля имя изменится цвет фона.

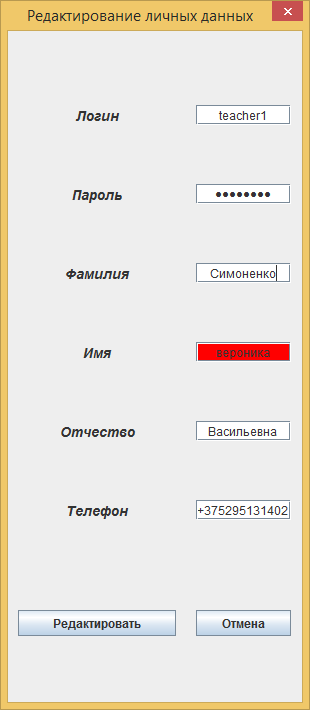


Рисунок 7.1 – Диалоговое окно, при вводе некорректных данных

При попытке ввода при редактировании или добавлении нового пользователя уже существующего номера телефона редактирование произведено не будет, а выведется предупреждающее сообщение. Данная исключительная ситуация отражена на рисунке 7.2.

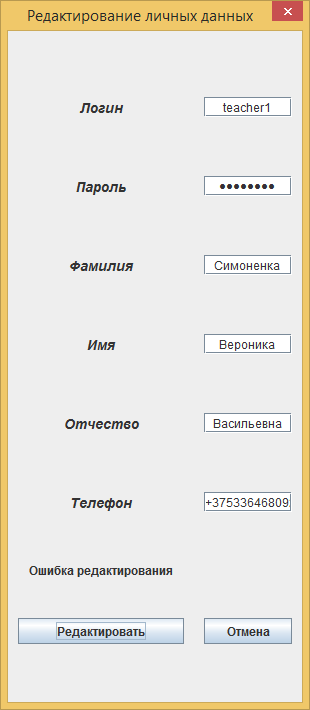


Рисунок 7.2 – Окно редактирования, с неуникальным телефоном

Если при попытке авторизации будут введены неправильный логин либо пароль, система уведомит пользователя о том, что в базе данных нет зарегистрированного аккаунта с таким логином и паролем. Результат попытки входа с вводом несуществующего логина показан на рисунке 7.3.



Рисунок 7.3 – Ошибка авторизации

В разработанном приложении предусмотрена обработка самых разнообразных исключительных ситуаций. В результате, при случайном возникновении ошибок, пользователю будет на понятном языке разъяснена причина ошибки. Также исключена ситуация потери или искажения полезных данных из-за неосознанной работы пользователя.

# ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта был изучен процесс учёта деятельности автошколы, который потребовал решение различных проблем.

В процессе изучения проблемы работы автошкол, связанной с огромным потоком требующей обработки информации, был предложен вариант создания программного средства для быстрого и структурированного доступа к этой информации.

В проекте была разработана структура данных, которая обеспечивает комфортную работу в программе и помогает быстро находить необходимую информацию. На её основе была разработана база данных в третьей нормальной форме, содержащая всю необходимую информацию о сущностях присутствующих в деятельности автошколы.

В программе реализован удобный пользовательский интерфейс, с помощью которого легко выполнять требуемые операции, интерфейс понятен в использовании, содержит множество всплывающих окон подсказок. Правильно разработанный пользовательский интерфейс имеет прямое влияние на эффективность работы пользователей с приложением. Разработанное программное средство нетрудно внедрить в работу автошколы, так как любой пользователь, независимо от возрастной категории и уровня знаний компьютерной техники, быстро освоиться с предлагаемым интерфейсом.

Система работает в многопоточном режиме с архитектурой клиент-сервер. К серверу может подключаться неограниченное число клиентов.

В ходе выполнения курсового проекта, были решены следующие задачи:

* исследованы и изучены все аспекты деятельности автошколы;
* разработан алгоритм работы отдельных функций и всей системы в целом;
* создана база данных;
* разработаны функции приложения;
* написан программный код в соответствии со всеми требованиями.

Данный проект соответствует поставленным требованиям. Также были предусмотрены все исключительны ситуации.

Подводя итоги, можно сказать, что было разработано приложение, которое имеет удобный клиентский интерфейс, различные функции работы с данными, а также удобную систему выставления зачетов студентам.

В заключении хочется отметить, что разработанное программное средство поможет оптимизировать работу автошкол и повысить качество обслуживания клиентов, цель курсового проекта можно считать достигнутой.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Автошколы [Электронный ресурс]. – Режим доступа: https://dic.academic.ru/dic.nsf/ruwiki/Автошкола
2. Конструирование базы данных [Электронный ресурс]. – Режим

доступа: https://revolution.allbest.ru/programming/00557685\_0.html

1. Карвин Б. – Программирование баз данных SQL. Типичные ошибки и их устранение (Профессиональные компьютерные книги) – 2012 : 204с.
2. MySQL / Ларри Ульман Пер. с англ. Слинкина А. А. – М. : ДМК «Пресс» СПб, Питер – 2004. – 352 с.: ил.
3. Гамма, Э. Приемы объектно-ориентированного программирования. Паттерны проектирования. Справочное пособие/ Р. Хелм, Р.Джонсон. – М. : Питер, 2010. — С. 368.
4. Аткинсон, Л. MySQL. Библиотека профессионала. Справочное пособие – : Издательский дом «Вильямс», 2014 – 624с.
5. Java Code Conventions
6. Шилдт, Герберт, Холмс, Джеймс – Искусство программирования на Java. : Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 336 с.: ил. – Парал. тит. англ.

# ПРИЛОЖЕНИЕ А

**(Обязательное)**

**Диаграмма вариантов использования**



Рисунок А.1 – Диаграмма вариантов использования

# ПРИЛОЖЕНИЕ Б

**(Обязательное)**

**Диаграмма последовательности выставления зачета**



Рисунок Б.1 – Диаграмма последовательности выставления зачета

# ПРИЛОЖЕНИЕ В

**(Обязательное)**

**Диаграмма развёртывания**



Рисунок В.1 – Диаграмма развёртывания

# ПРИЛОЖЕНИЕ Г

**(Обязательное)**

**Процесс регистрации нового учащегося**

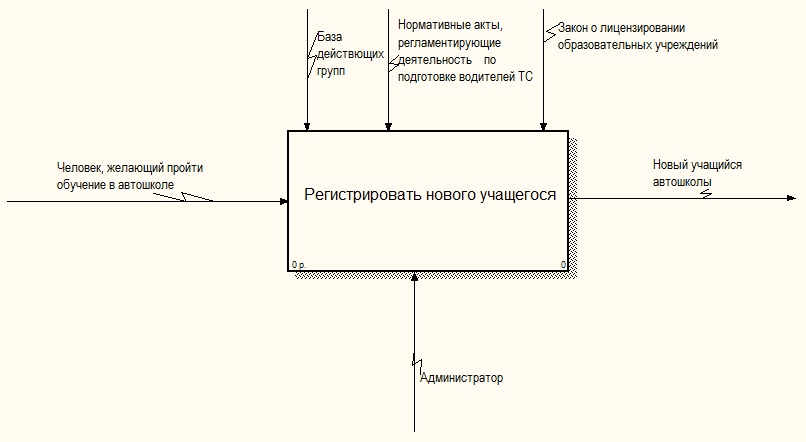


Рисунок Г.1 – Контекстная диаграмма регистрации нового учащегося

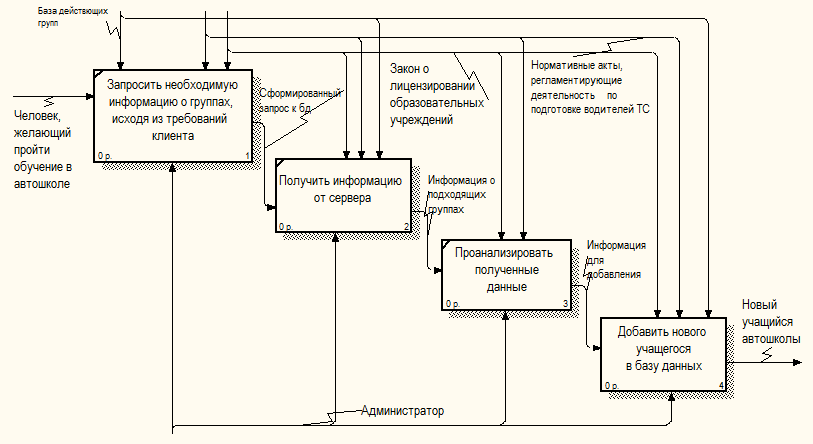


Рисунок Г.2 – Декомпозиция контекстной модели

Продолжение приложения Г

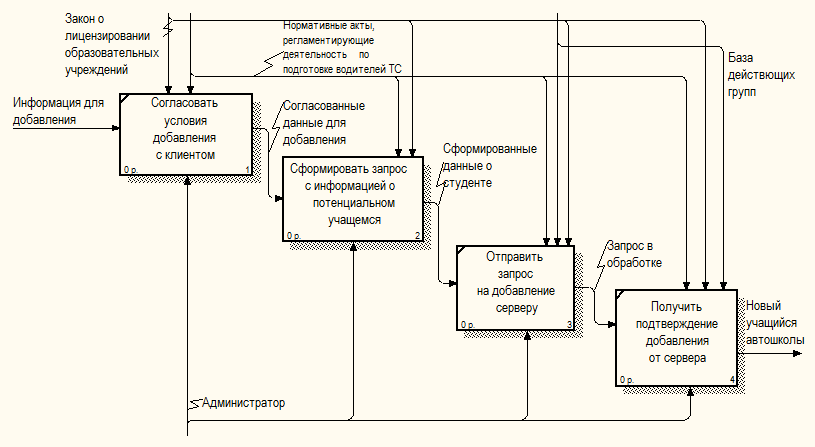


Рисунок Г.3 – Декомпозиция блока «Добавить нового учащегося в базу данных»

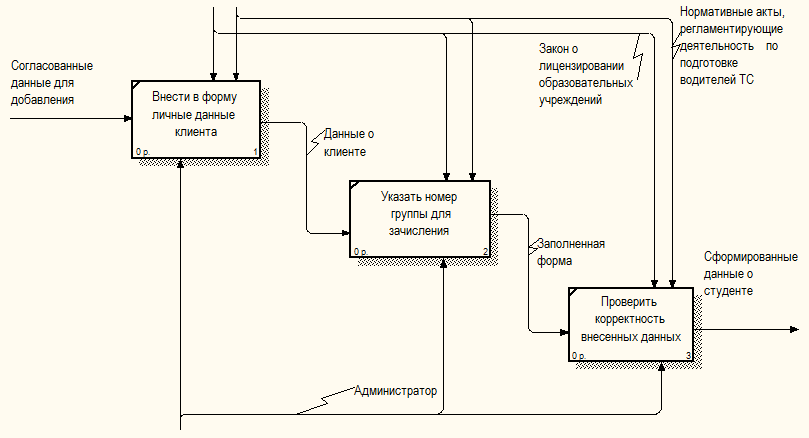


Рисунок Г.4 – Декомпозиция блока «Сформировать запрос с информацией о потенциальном учащемся»

# ПРИЛОЖЕНИЕ Е

**(Обязательное)**

**Блок-схемы алгоритмов, реализующих бизнес-логику**

**серверной части приложения**

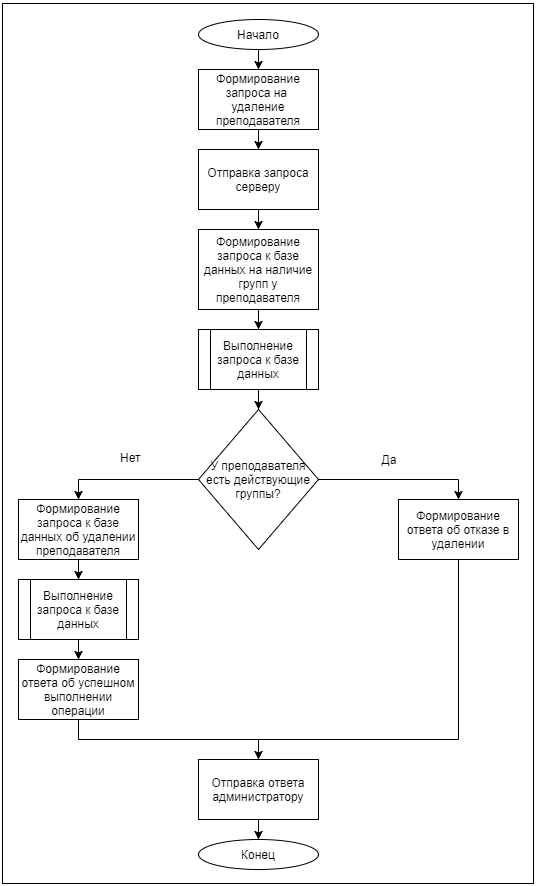
****

Рисунок Е.1 – Блок-схема алгоритма удаления преподавателя

Продолжение приложения Е

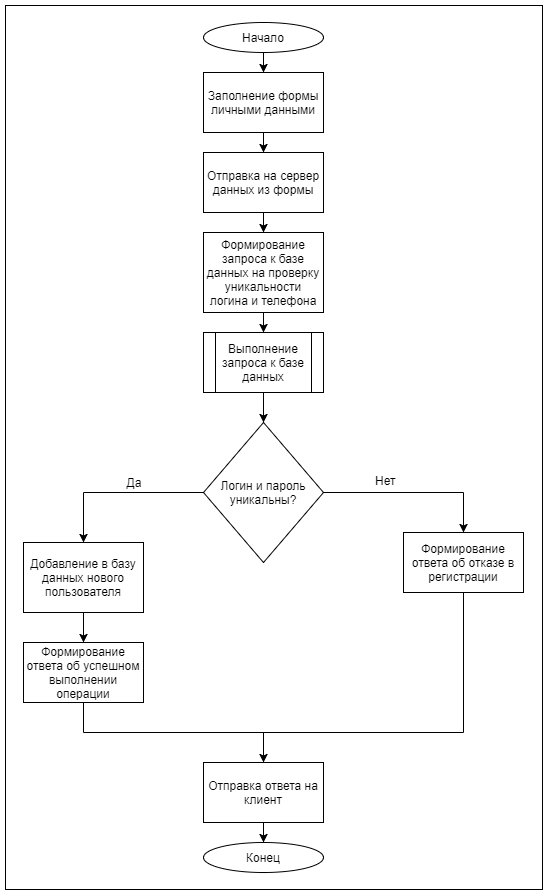


Рисунок Е.2 – Блок-схема алгоритма регистрации нового пользователя

Продолжение приложения Е

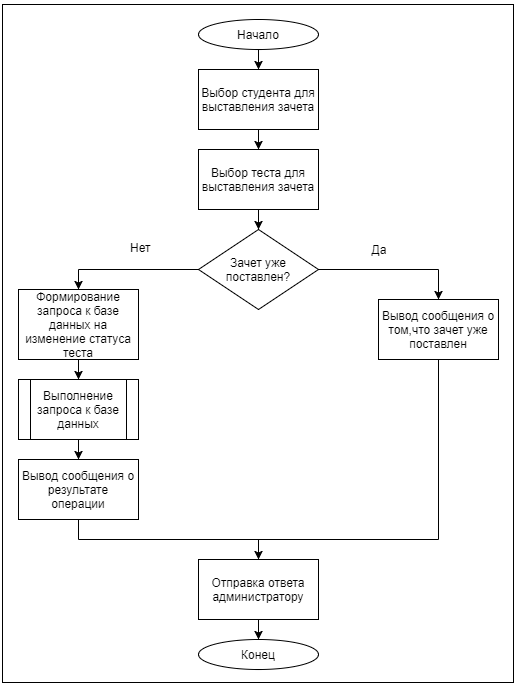


Рисунок Е.3 – Блок-схема выставления зачета учащемуся

# ПРИЛОЖЕНИЕ Ж

**(Обязательное)**

**Диаграмма состояний процесса редактирования данных**



Рисунок Ж.1 – Диаграмма состояний процесса редактирования данных

# ПРИЛОЖЕНИЕ З

**(Обязательное)**

**Диаграмма компонентов**



Рисунок З.1 – Диаграмма компонентов

# ПРИЛОЖЕНИЕ И

**(Обязательное)**

**Диаграммы классов**

****

Рисунок И.1 – Диаграмма классов, предназначенных для хранения данных из базы данных



Рисунок И.2 – Диаграмма классов для оконного представления



Рисунок И.3 – Диаграмма классов, описывающих логику работы сервера



Рисунок И.4 – Диаграмма классов, реализующих паттерн DaoFactory