**Оглавление**

[Аннотация](#_Toc26196310)

[Введение](#_Toc26196311) 4

[1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ](#_Toc26196312) 4

[1.1 Основное функциональное назначение программного средства](#_Toc26196313) 5

[1.2 Полное наименование программного средства](#_Toc26196314) 5

[1.3 Условное обозначение программного средства](#_Toc26196315) 5

[1.4 Разработчики программного средства](#_Toc26196316) 5

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ](#_Toc26196317) 6

[2.1 Основание для разработки](#_Toc26196318) 6

[2.2 Назначение разработки](#_Toc26196319) 6

[2.3 Требование к программному средству](#_Toc26196320) 6

[2.4 Требования к программной документации](#_Toc26196321) 7

[2.5 Требования к эргономике и технической эстетике](#_Toc26196322) 7

[2.6 Стадии и этапы разработки](#_Toc26196323) 8

[2.7 Порядок контроля и приемки](#_Toc26196324) 9

[3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ 11](#_Toc26196325)

[3.1 Декомпозиция поставленной задачи 1](#_Toc26196326)0

[3.2 Общая архитектура программного средства 1](#_Toc26196327)1

[3.3 Разработка алгоритма решения задачи 1](#_Toc26196328)4

[3.4 Реализация функционального назначения программного средства 1](#_Toc26196329)5

[3.5 Структурная организация данных 1](#_Toc26196330)5

[3.6 Разработка интерфейса ПС](#_Toc26196331) 26

[3.7 Описание структуры выходной информации 3](#_Toc26196332)1

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 26](#_Toc26196333)

[4.1 Назначение программного средства 26](#_Toc26196334)

[4.2 Условия выполнения программного средства 26](#_Toc26196335)

[4.3 Эксплуатация программного средства 26](#_Toc26196336)

[4.4 Сообщения пользователю 32](#_Toc26196337)

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА 34](#_Toc26196338)

[5.1 Обоснование необходимого количества тестов 34](#_Toc26196339)

[5.2 Описание тестовых пакетов 35](#_Toc26196340)

[6. ЗАКЛЮЧЕНИЕ 42](#_Toc26196341)

[6.1 Оценка качества программного средства с помощью метрик 42](#_Toc26196342)

[7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45](#_Toc26196343)

[Приложение А листинг программного модуля 47](#_Toc26196344)

[Приложение B диск с исполняемым кодом программного модуля 65](#_Toc26196345)

**Аннотация**

На курсовой проект студента Ларионова А.C. на тему: Разработка интегрированного программного модуля «Автошкола». Курсовой проект представлен пояснительной запиской на 65 листов машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве Автошкола. В главе 2 приведено техническое задание на проектирование БД согласно ГОСТ 19.201. Глава 3 содержит пояснительную записку к программному продукту согласно ГОСТ 19.404-79. В главе 4 содержится руководство пользователя согласно ГОСТ 19.504-79. Глава 5 содержит обоснование количества необходимых тестов и тестовые пакеты для оценки работоспособности программного продукта. В главе 6 приведены результаты ручного и автоматического тестирования программного продукта и выполнена оценка качества с помощью метрик.

В работе содержится общее описание программного средства, предназначенного для объекта проектирования – интегрированного программного модуля «Автошкола», изложены требования к программному средству, описаны его архитектура, реализация функций, интерфейс, приведено руководство пользователя, обоснованы тестовые пакеты и выполнена оценка качества программного продукта с помощью метрик.

**Введение**

В курсовом проекте была поставлена задача разработки интегрированного программного модуля «Автошкола», который позволяет автоматизировать учет студентов.

Для достижения поставленной цели был составлен основной алгоритм работы программы, а также разработаны алгоритмы вспомогательных процедур.

Для реализации алгоритмов был выбран язык программирования Java. Выбор обусловлен тем, что возможностей этого языка достаточно для достижения поставленной цели моей курсового проекта.

При разработке интерфейса программы, я учитывал наиболее нужные и важные функциональные возможности, которые должна выполнять данная программа, а также тип пользователей, которые с ней будут работать.

**1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена для автоматизации учета студентов. Программа не требует владения современными информационными технологиями для эффективного использования данного программного средства, т.е. подходит для любого пользователя.

**1.2 Полное наименование программного средства**

Разработка интегрированного программного модуля «Автошкола».

**1.3 Условное обозначение программного средства**

«autoschool-app»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 215 Ларионов А.С.

**2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Основание для разработки**

Работа ведётся на основании задания на курсовой проект по специальности 09.02.07 — «Информационные системы и программирование».

**2.2 Назначение разработки**

Программа «autoschool-app» кросс-платформенная. Программа предназначена для автоматизации деятельности администратора по учету студентов автошколы. Программа позволяет вводить, изменять и удалять данные студентов, инструкторов , автомобилей, категорий групп а также просматривать статусы,

**2.3 Требование к программному средству**

«autoschool-app» устанавливается на персональный компьютер, имеющий представленные минимальные системные требования:

Оперативная память: 4 Гб;

Процессор: Intel Core i / Phenom II X4 / AMD Athlon64 X2 4600 + 2.4 ГГц;

Видео карта: nVidia GeForce 8600 / ATI Radeon HD 2400;

Звуковая карта: Совместимая с DirectX;

Программный продукт «autoschool-app» является кроссплатформенным продуктом, что позволяет не ограничиваться в работе с определенной версией ОС.

«autoschool-app» написана на объектно-ориентированном языке программировании Java в интегрированной среде разработки IntelliJ IDEA. В качестве СУБД используется PostgresQL. Программа не может подлежать сторонней оптимизации и редактированию программного кода.

**2.4 Требования к программной документации**

Состав программной документации должен включать в себя:

1. техническое задание;

2. сроки выполнения работ;

3. программу и методику испытаний;

4. эксплуатационные инструкции пользователю;

**2.5 Требования к эргономике и технической эстетике**

Система обеспечивает удобный для пользователей системный интерфейс, отвечающий следующим требованиям:

- единый стиль оформления для пользовательских интерфейсов;

- удобная, интуитивно понятная навигация в интерфейсе пользователя;

- взаимодействие пользователя с системой осуществляется на двух языках по выбору (русский и английский), для наибольшего удобства

Пользовательские интерфейсы системы спроектированы и разработаны с применением единых принципов графического представления информации и организации доступа к функциональным возможностям и сервисам. Разработан графический дизайн пользовательских интерфейсов, цветовые, шрифтовые и композиционные решения для отображения текстов, изображений, таблиц, гиперссылок, управляющих и навигационных элементов (меню, кнопок, форм и т.п.).

Экранные формы спроектированы с учетом требований унификации:

- все экранные формы пользовательского интерфейса выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

- для обозначения сходных операций использованы сходные графические значки, кнопки и другие управляющие (навигационные) элементы. Термины, используемые для обозначения типовых операций (добавление информационной сущности, редактирование поля данных), а также последовательности действий пользователя при их выполнении унифицированы;

- внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки) реализованы одинаково для однотипных элементов.

**2.6 Стадии и этапы разработки**

Разработка должна быть проведена в три стадии:

1. разработка технического задания;

2. рабочее проектирование;

3. внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;

2. разработка программной документации;

3. испытания программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;

2. определение и уточнение требований к техническим средствам;

3. определение требований к программе;

4. определение стадий, этапов и сроков разработки программы и документации на нее;

5. выбор языков программирования;

6. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с пунктом Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1. разработка, согласование и утверждение порядка и методики испытаний;

2. проведение испытаний;

3. корректировка программы и программной документации по результатам испытаний.

**2.7 Порядок контроля и приемки**

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой и календарным планом. Сдача-приемка осуществляется комиссией, в состав которой входят представители ГАПОУ и работодатель. По результатам приемки подписывается акт приемочной комиссии.

В указанном подразделе, согласно таблице «Сроки выполнения работ» технического задания, будет описаны перечень участвующих организаций, место и сроки проведения работ, согласно п. 2.8 ГОСТ 34.602-89.

Порядок согласования и утверждения приемочной документации должен регламентироваться организационно-распорядительной документацией организации, принимающей участие в создании программного продукта. Согласно разделу «Приемка результатов разработки» ГОСТ 15.001-88 для согласования и утверждения приемочной документации создается приемочная комиссия (приказом).

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

**3. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ**

**3.1 Декомпозиция поставленной задачи**

Для достижения цели работы была поставлена следующая задача:

Создать модуль “CodeDatabase” отвечающий за сохранение БД. Результат работы модуля - сохранение БД.

Поставленные задачи наглядно отображаются на диаграмме деятельности.

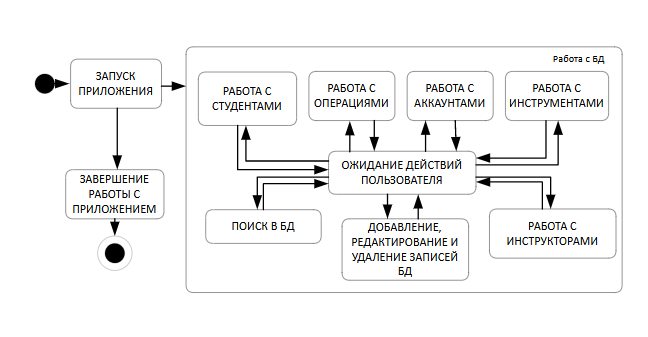


Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

Объектом моделирования является Автошкола.

Рассматриваемые процессы: Добавление и удаление информации о мероприятиях.

Объекты моделирования представлены на диаграмме классов, рис.2.

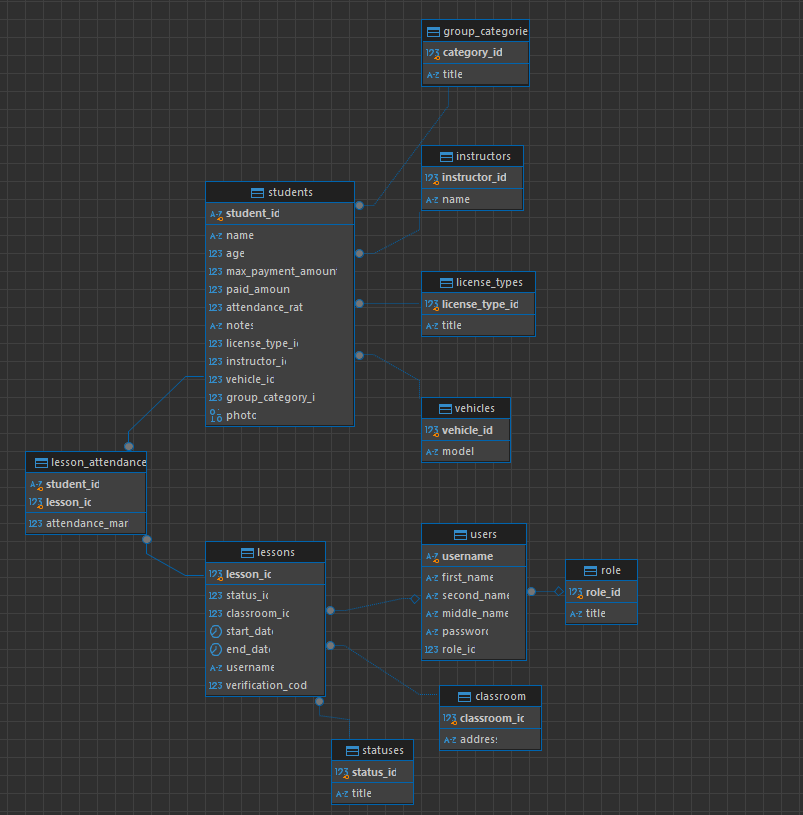


Рис.2 Диаграмма классов

По полученной декомпозиции задач была спроектирована следующая архитектура программного средства.

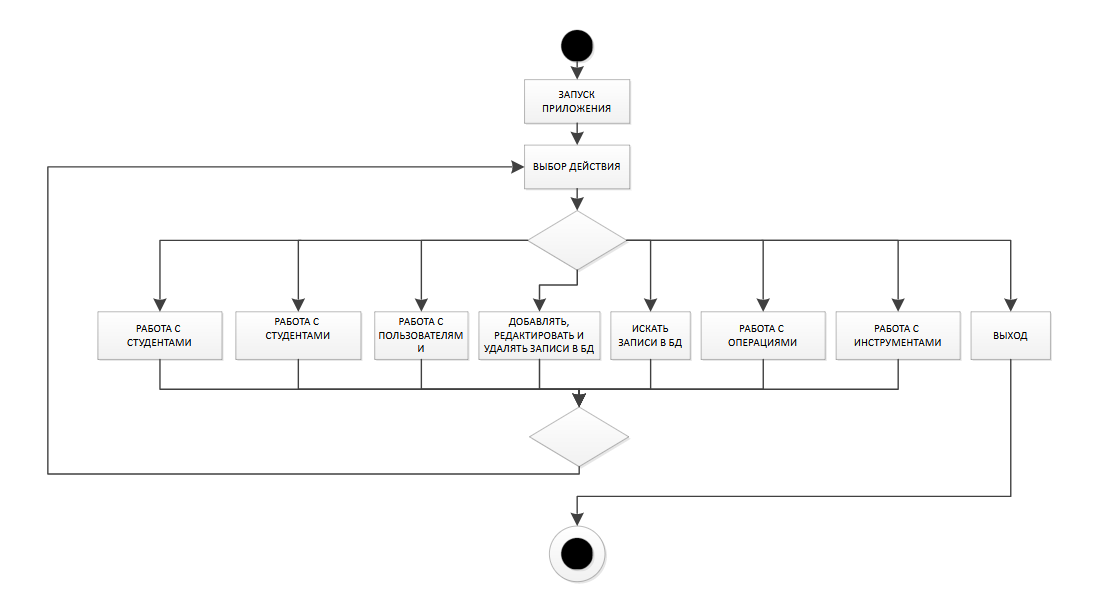


Рис.3 Схема событийно-ориентированной архитектуры

В «autoschool-app» используется событийно-ориентированная архитектура.(Рис.3) Роль студента (источник событий) в программе выполняют: пользователи программы, за роль стоков (потребители событий) отвечают таблицы, входящие в базу данных. Например, когда пользователь выбирает определенное действие: добавить, удалить, редактировать и т.д. система осуществит выбранные действия, и база отреагирует соответствующим образом: запись добавлена, удалена, отредактирована, либо отчет выведен в  PDF. Системная архитектура пользователя рассматривает это изменение состояния как событие, создаваемое, публикуемое, определяемое и потребляемое различными приложениями в составе архитектуры.

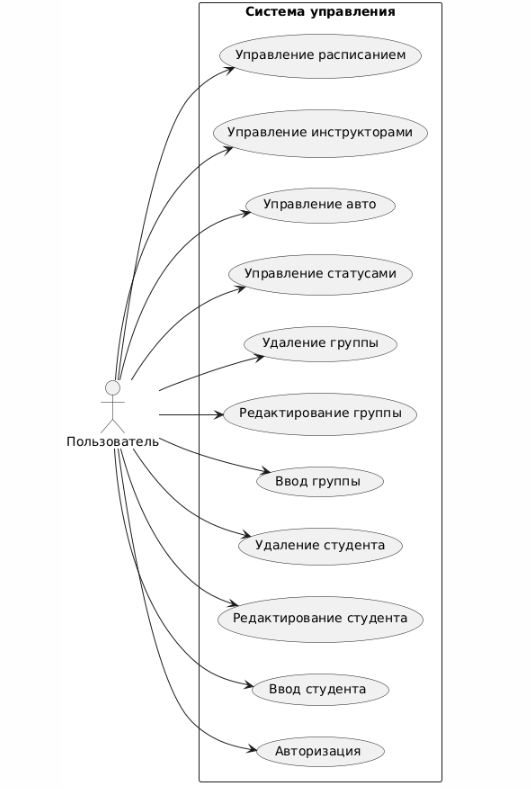


Рис 4. Диаграмма вариантов использования

**3.3 Разработка алгоритма решения задачи**

**Общий алгоритм**

 Рис 5. Алгоритм программы

**3.4 Реализация функционального назначения программного средства**

Программа «autoschool-app» имеет следующий набор входных данных, такие как: cтуденты и информация о них.

Данные вводятся пользователем в соответствующие поля ввода, снабженные всплывающими подсказками.

Выходными данными являются: отображение списка студентов в виде плиточного интерфейса.

Данные вводятся пользователем в соответствующие поля вывода, снабженные всплывающими подсказками.

Выходные данные редактировать вручную невозможно, т.к. они служат в качестве отчетной информации для пользователя.

**3.5 Структурная организация данных**

Для создания БД необходимо определиться с данными, которые необходимы для полноценного функционирования системы. Все эти данные указаны в реляционной модели «БД Автошкола» представленная на рисунке. Любая реляционная база данных и называется реляционной, что характеризуется отношениями (relation) между таблицами. На рисунке изображены таблицы моей базы данных. При этом одна таблица является родительской (главной), а вторая – дочерней (подчиненной). Главной таблицей является «Студенты». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кодда. (рис.6)

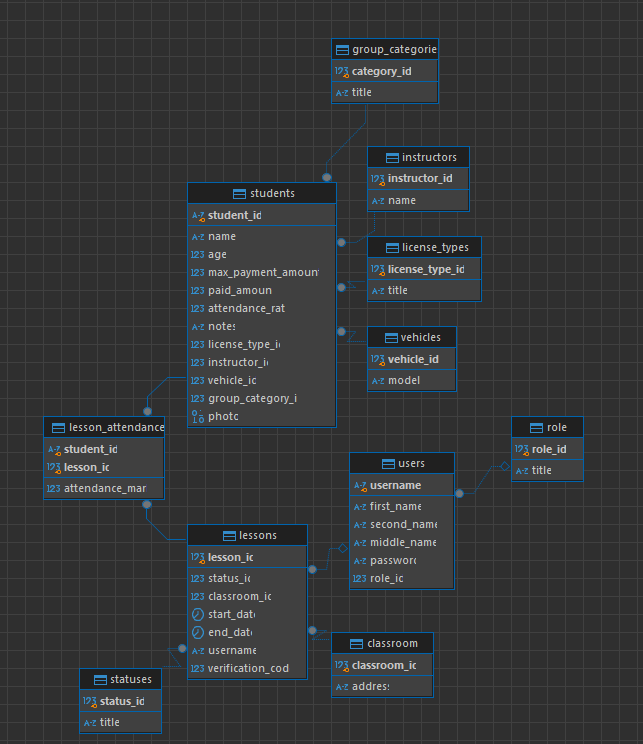


Рис.6 Реляционная модель

**Первичный ключ в базе уникален, используется для организации отношений между таблицами, который не может иметь пустых и повторяющихся значений.**  
Первичными ключами в базе являются поля: classroom\_id (таблица classroom), category\_id (таблица group\_categories), instructor\_id (таблица instructors), license\_type\_id (таблица license\_types), role\_id (таблица role), status\_id (таблица statuses), vehicle\_id (таблица vehicles), student\_id (таблица students), lesson\_id (таблица lessons). Остальные ID-ключи являются внешними ключами.

**Для организации более эффективной обработки данных применяется нормализация. Таблицы моей БД находятся в 3НФ:**

* **БД находится в форме 1НФ потому, что:**
  + Таблица не имеет повторяющихся записей;
  + Каждый атрибут отношения хранит одно-единственное значение и не является списком, ни множеством значений;
  + Таблица не имеет повторяющихся групп полей.
* **Вторая нормальная форма (2НФ):**
  + Устранены атрибуты, зависящие только от части уникального (первичного) идентификатора, т.е. ID.
* **Третья нормальная форма (3НФ):**
  + Отсутствуют атрибуты, зависящие от атрибутов, не входящих в уникальный (первичный) ключ.

**На основе реляционной модели базы данных мною была построена следующая база данных.**  
Связь таблиц организована следующим образом:

* Таблица students связана с таблицей group\_categories по полю group\_category\_id (тип связи один ко многим, 1:М).
* Таблица students связана с таблицей instructors по полю instructor\_id (тип связи 1:М).
* Таблица students связана с таблицей vehicles по полю vehicle\_id (тип связи 1:М).
* Таблица students связана с таблицей license\_types по полю license\_type\_id (тип связи 1:М).
* Таблица lessons связана с таблицей classroom по полю classroom\_id (тип связи 1:М).
* Таблица lessons связана с таблицей statuses по полю status\_id (тип связи 1:М).
* Таблица lessons связана с таблицей users по полю username (тип связи 1:М).
* Таблица lesson\_attendance связана с таблицами students и lessons через внешние ключи student\_id и lesson\_id соответственно (тип связи многие ко многим, M:М).

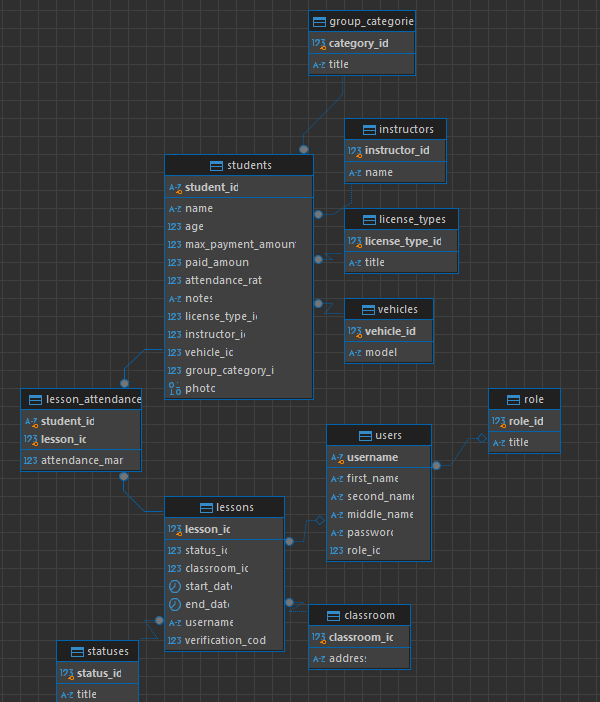


Рис.7 Схема базы данных

Таблица “group\_categories”(Категория групп)

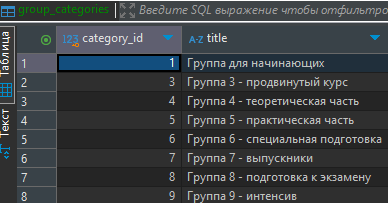


Рис.8 Таблица group\_categories

Таблица “instructors”(Инструктора)

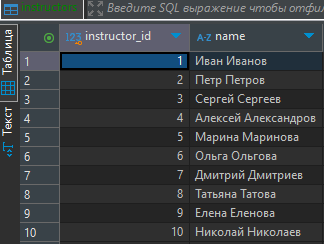


Рис.9 Таблица instructors

Таблица “license\_types”(Тип лицензии)

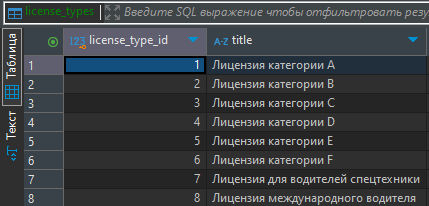


Рис.10 Таблица license\_types

Таблица “vehicles”(Транспорт)

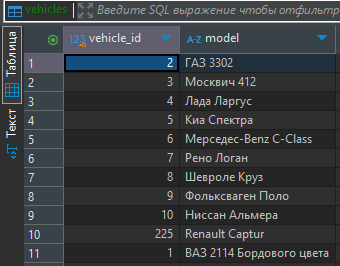


Рис.11 Таблица vehicles

Таблица “students”(Студенты)

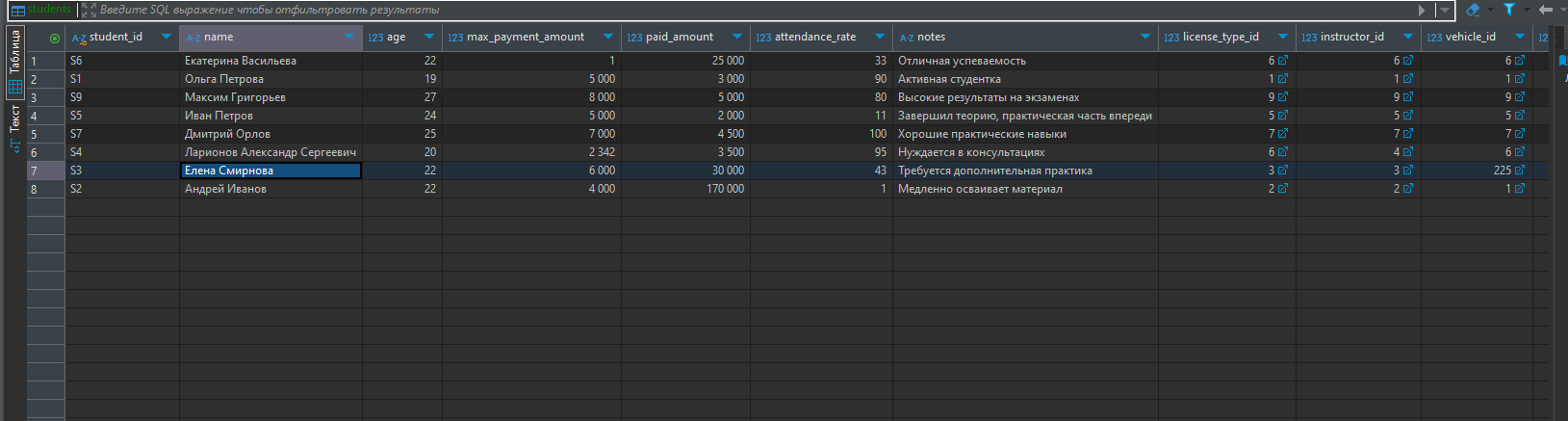


Рис.12 Таблица students

Таблица “lesson\_attendance”(Оценки за урок)

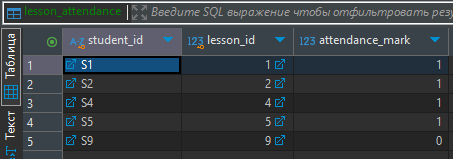


Рис.13 Таблица lesson\_attendance

Таблица “lessons”(Уроки)

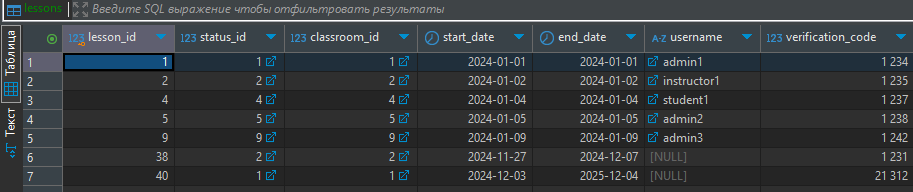


Рис.14 Таблица lessons

Таблица “users”(Пользователи)

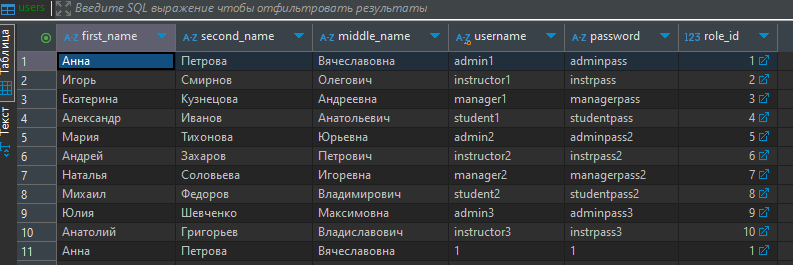


Рис.15 Таблица users

Таблица “role”(Роли)

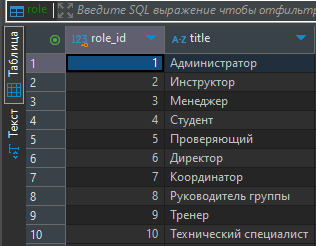


Рис.16 Таблица role

Таблица “classroom”(Учебный класс)

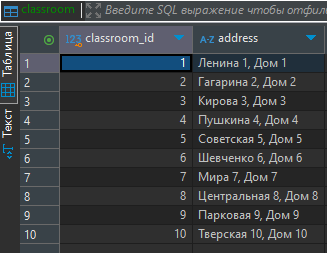


Рис.17 Таблица classroom

Таблица “statuses”(Статусы)

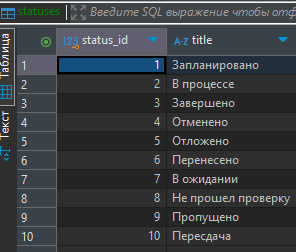
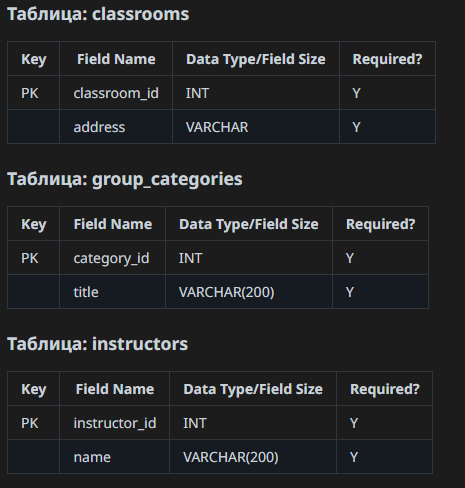
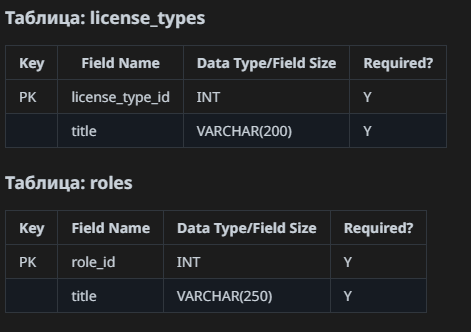
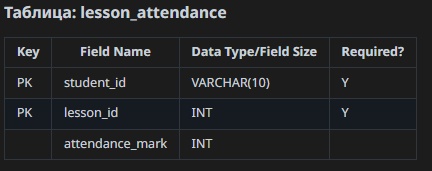


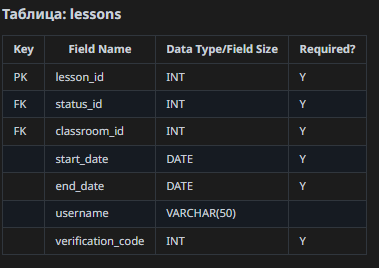
Рис.18 Таблица statuses

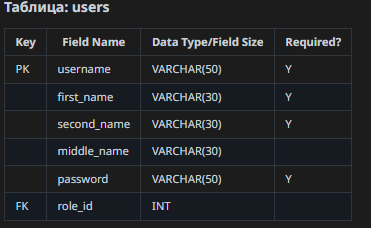
Словарь данных

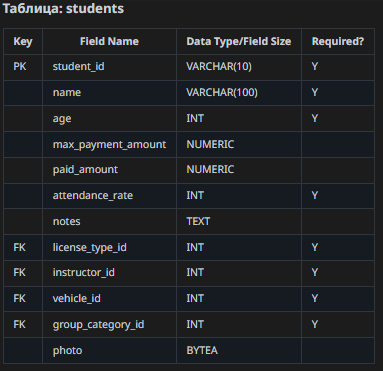


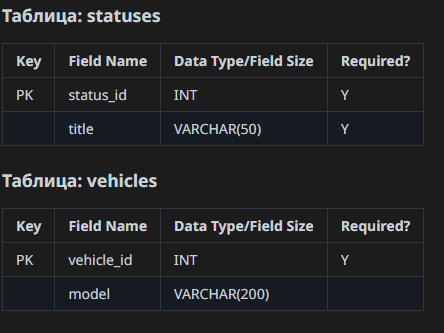




****

****

****

****

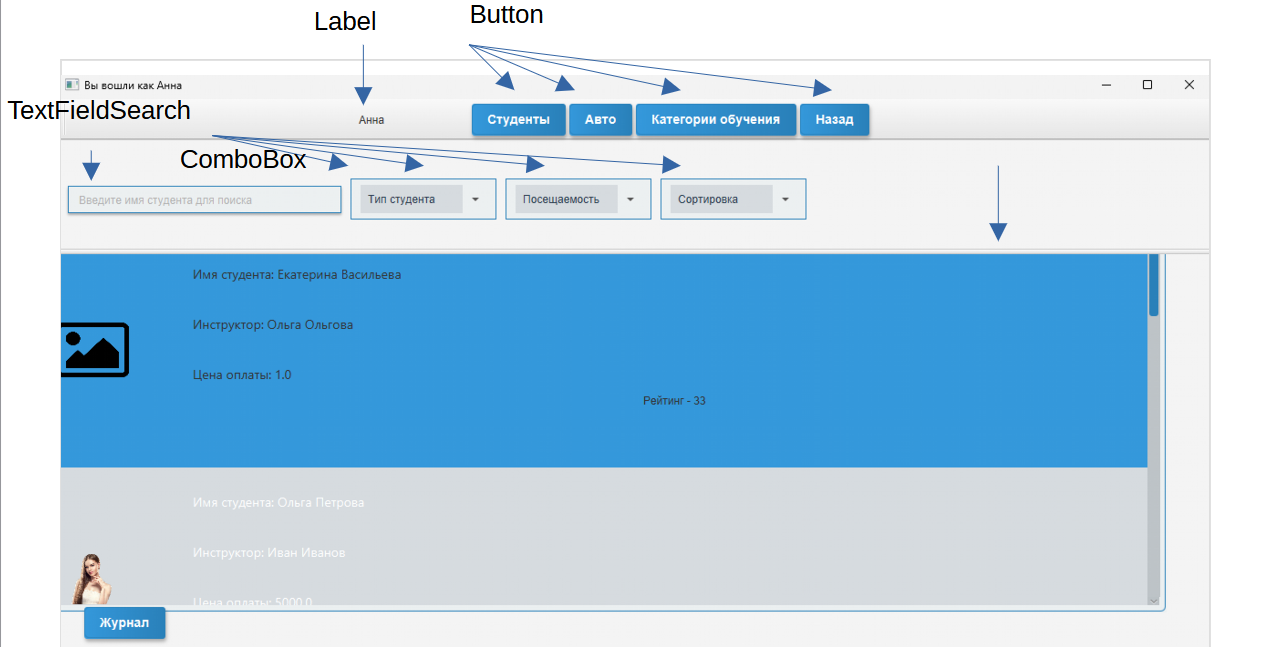
**3.6 Разработка интерфейса ПС**

В приложении предусмотрена авторизация, так как оно многопользовательское. Для запуска приложения достаточно щелкнуть дважды левой кнопкой мыши по ярлыку.

После старта приложения пользователь увидит форму авторизации. Успешная авторизация предоставит доступ к главному окну программы, где будут отображаться все события из базы данных. Неуспешная авторизация вернет пользователя на форму авторизации с соответствующим сообщением об ошибке. «autoschool-app» обладает удобным интерфейсом, так что даже неопытному в обращение с пользовательскими компьютерными программами пользователю не составит труда быстро привыкнуть к работе с данным программным средством.

На рабочем экране расположены несколько кнопок для комфортной работы с приложением.

**Главная форма.** Основная форма приложения, в которой в виде строк отображаются данные по студентам. Компоненты формы: ListView – для отображения данных студентов по автошколы, Combobox – выпадающие списки, TextBox, Label . (Рис.21)

Рис.19 Главная

**Страница Cтуденты**. Компоненты Страницы: TableView – для отображения списка студентов, Button для перехода на дополнительные страницы, TextBox и Label. (Рис.20)

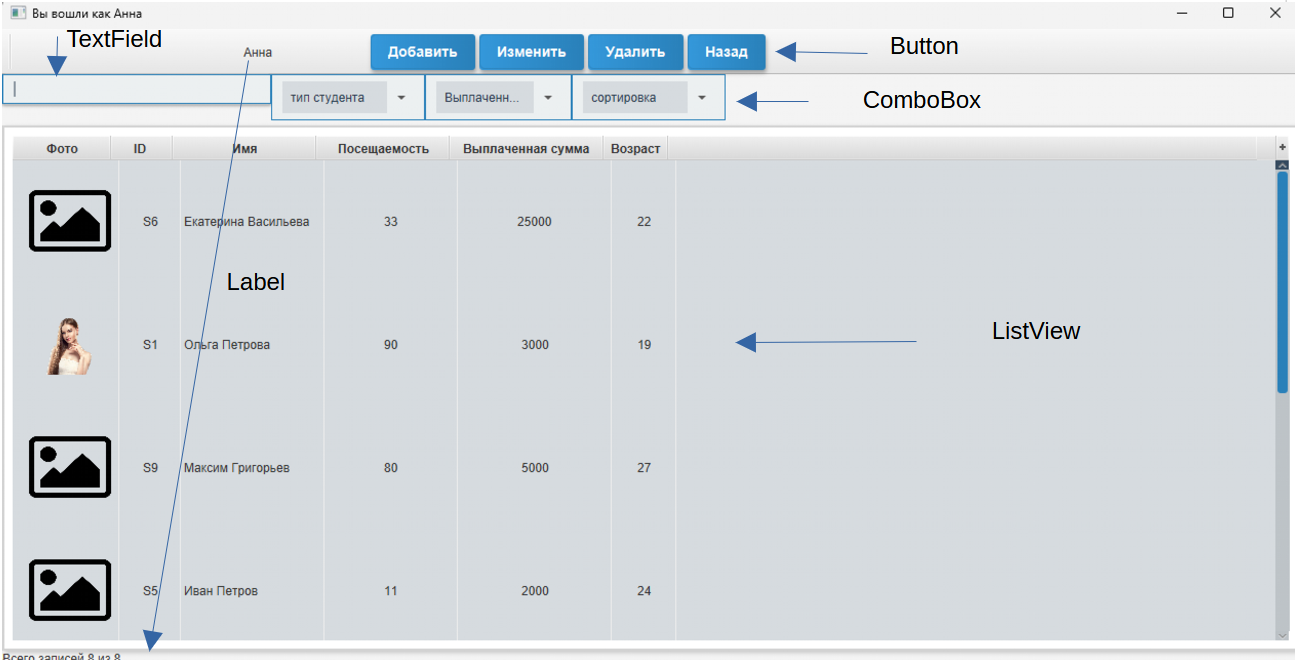


Рис.20 Cтуденты

**Страница Добавление и изменение**. Интерфейс данной формы состоит из компонентов: TextBox – поля ввода, СomboBox – выбор лицензии и инструктора и Button. (Рис.21)

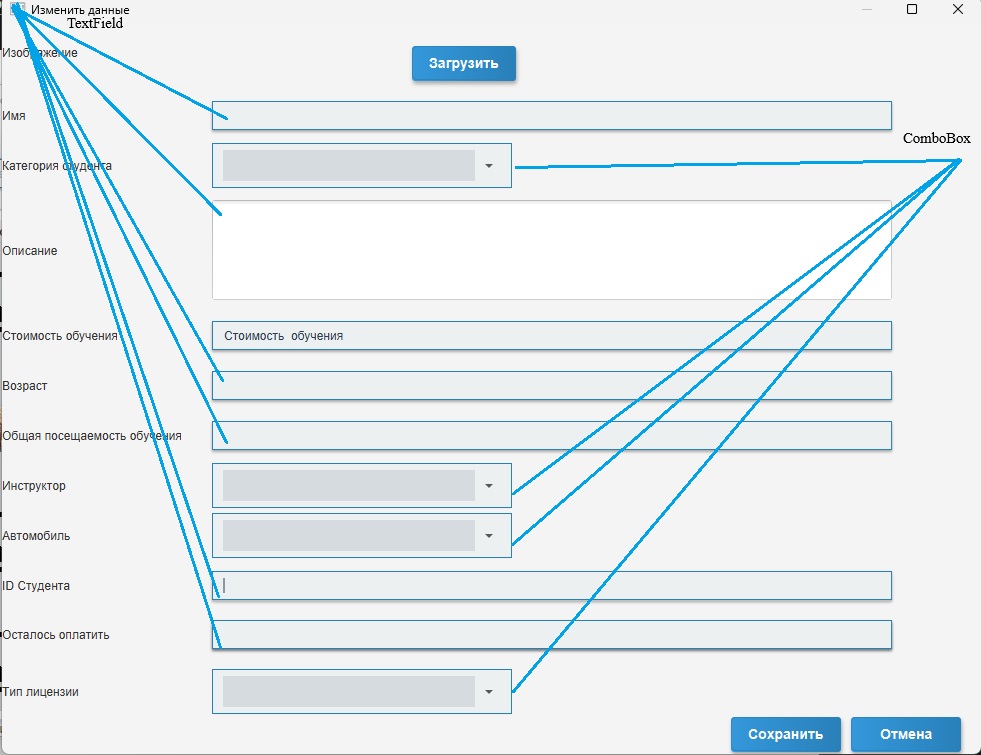
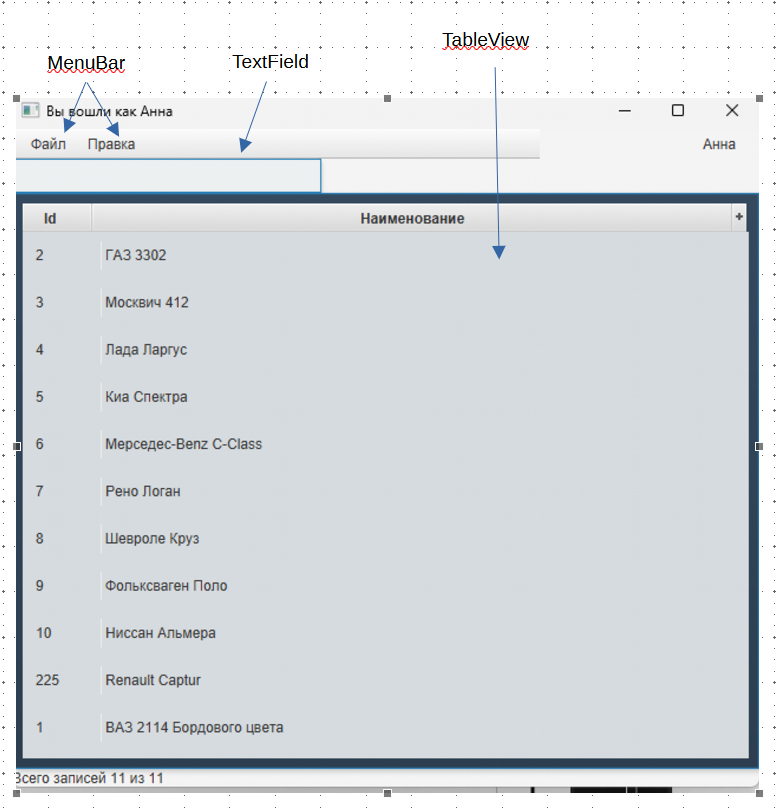
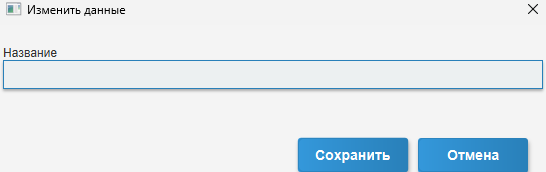


Рис.21 Страница Добавление и изменение

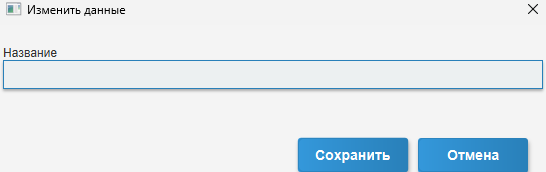
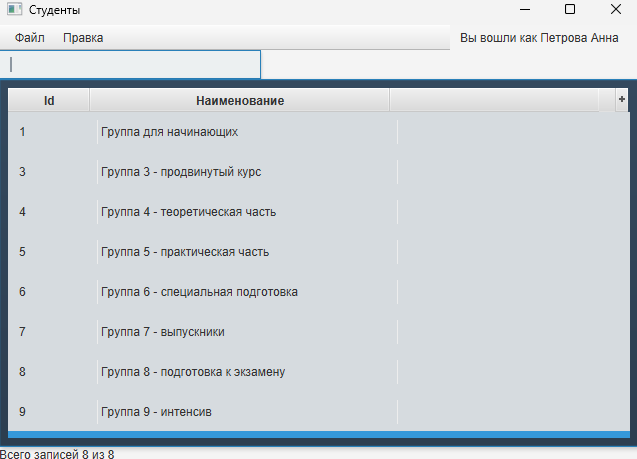
**Страница авто.** Компоненты Страницы: TableView – для отображения списка авто, MenuBar для перехода на дополнительные страницы, TextBox и Label. (Рис.22)

Рис.22 Операции

**Страница Добавление и изменение**. Интерфейс данной формы состоит из компонентов TextField для ввода автомобиля а также Button. (Рис.23)

Рис.23 Страница Добавление и изменение

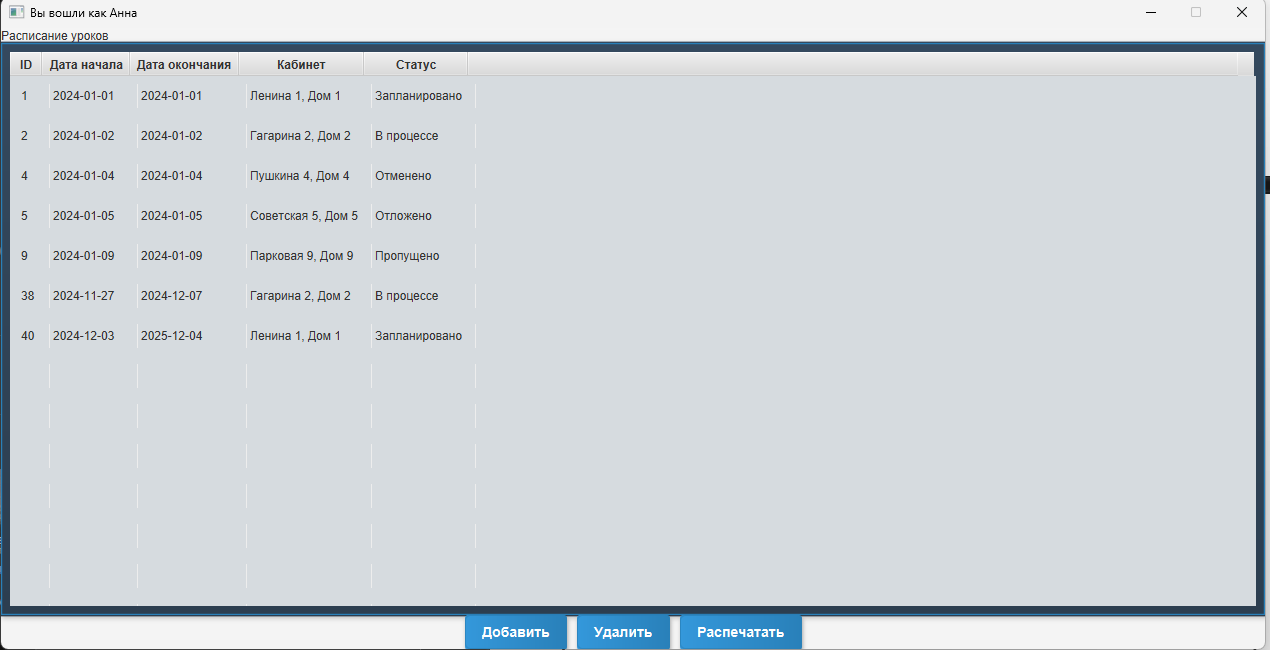
**Страница Категория обучения**. Компоненты Страницы: TableView – для отображения списка групп, MenuBar для перехода на дополнительные страницы, TextBox и Label. (Рис.30)

Рис.24 Страница Категория обучения

**Страница Добавление и изменение**. Интерфейс данной формы состоит из компонентов TextField для ввода автомобиля а также Button. (Рис.25)

Рис.25 Страница Добавление и изменение

**Страница Расписание уроков**. Компоненты Страницы: TableView – для отображения списка групп, MenuBar для перехода на дополнительные страницы, TextBox и Label. (Рис.26)

Рис.26 Страница расписание уроков

**Страница Добавление и изменение**. Интерфейс данной формы состоит из компонентов: TextBox – поля ввода DatePicker – выбор даты, Combobox – выпадающие списки и Button. (Рис.27)

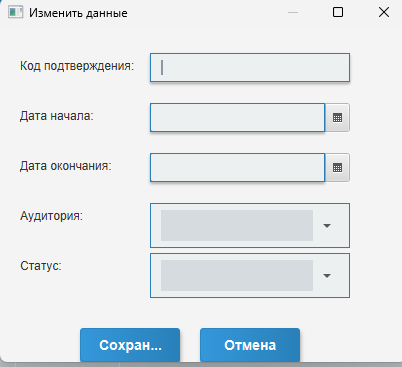
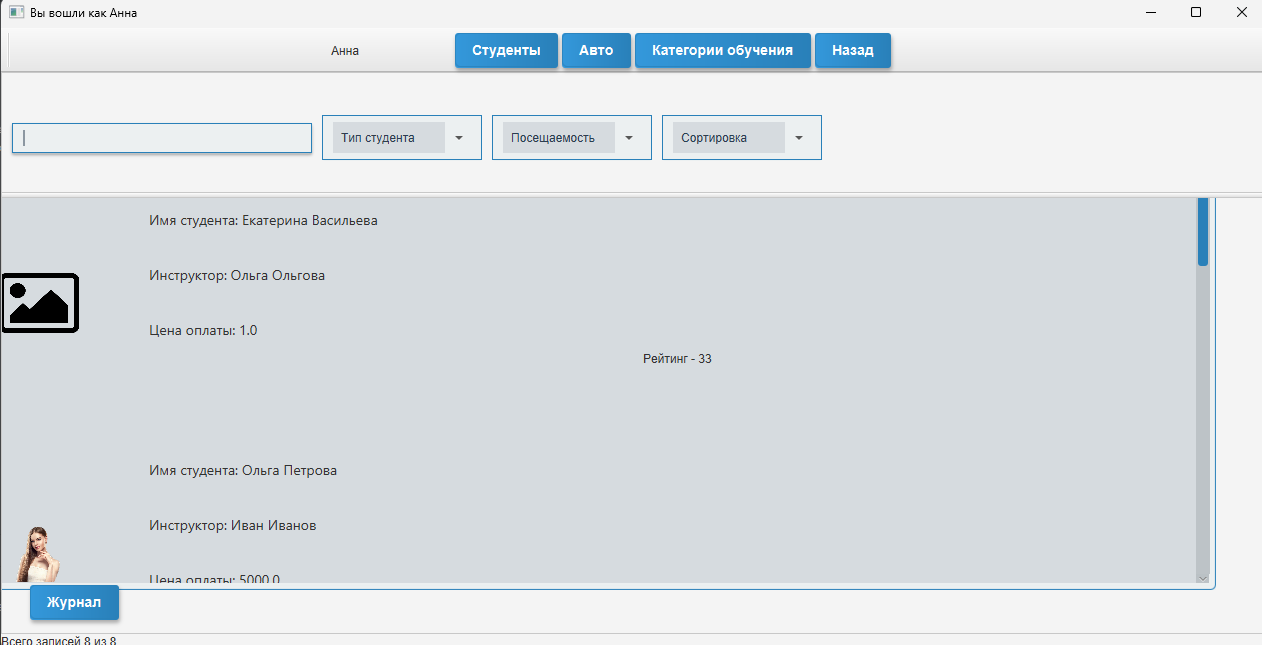


Рис.27 Страница Добавление и изменение

**3.7 Описание структуры выходной информации**

Выходными данными являются списки студентов и его данных в автошколе отфильтрованные по разным критериям.

Рис.28 Списки студентов

**4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Назначение программного средства**

Назначением программного средства является автоматизация учета студентов и автошколы .

**4.2 Условия выполнения программного средства**

Условия, при соблюдении которых возможно выполнение работы с программным продуктом:

-Компьютер с установленной на нём «autoschool-app».

Подготовительные действия:

-На иконке «autoschool-app» рабочего стола произвести двойной щелчок левой кнопкой мыши.

**4.3 Эксплуатация программного средства**

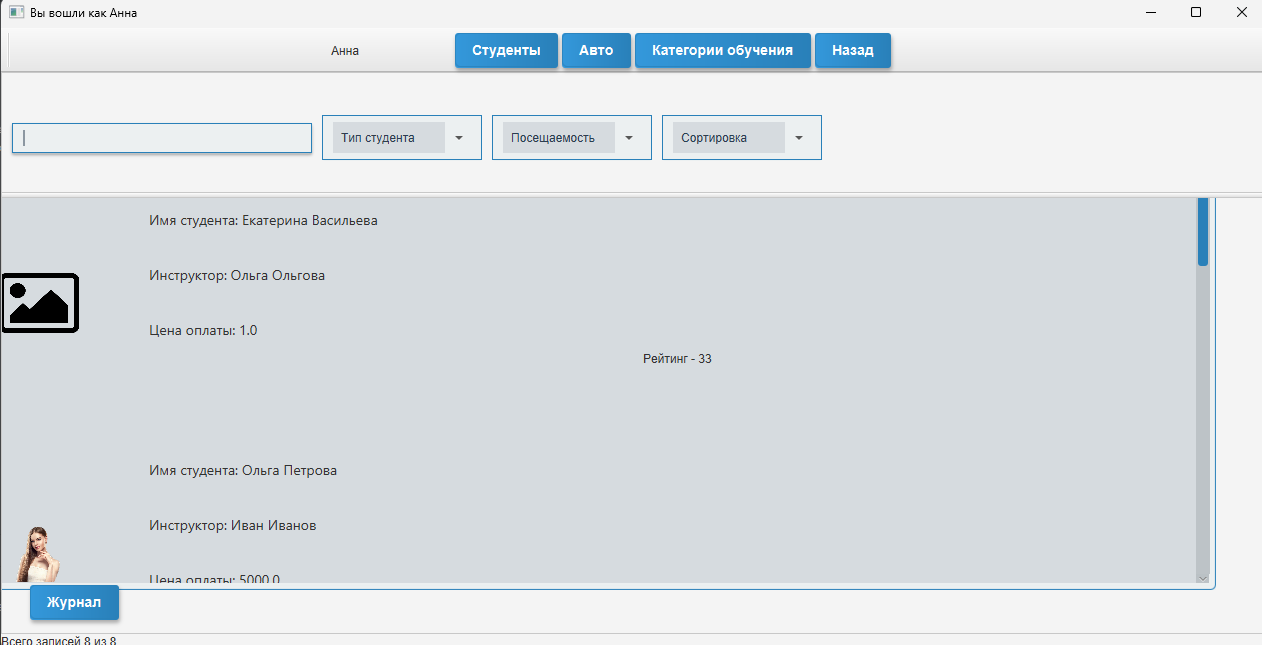
Этот раздел содержит обзор приложений «autoschool-app» и ее основных параметров.

***Активация продукта***

Активировать копию программы autoschool-app можно сразу после её установки, когда появится окно подтверждения подлинности Вашей версии. Ключ активации расположен внутри упаковки программного продукта. Для успешного выполнения активации ключ активации необходимо вводить именно в том виде, в котором он предоставлен.

Вводные сведения об интерфейсе пользователя

После старта приложения откроется главное окно приложения – Список cтудентов данных по автошколе в строчном формате. На форме есть кнопка для перехода на другие страницы. Есть поля для поиска студентов по названию и выпадающие списки для фильтрации. (Рис.29)

Рис.29 Окно данные cтудентов в автошколе

Кликнув по кнопке «Выход» программа потребует подтверждение.(Рис.30)

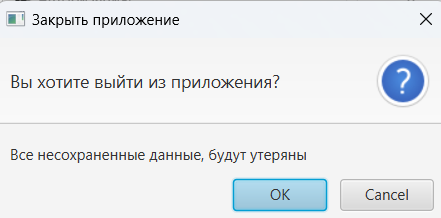


Рис.30 Сообщение программы

Ниже описаны пункты меню управления и информации.

«Страница Студенты»: эта форма предназначена для отображения списка студентов. (Рис. 31)

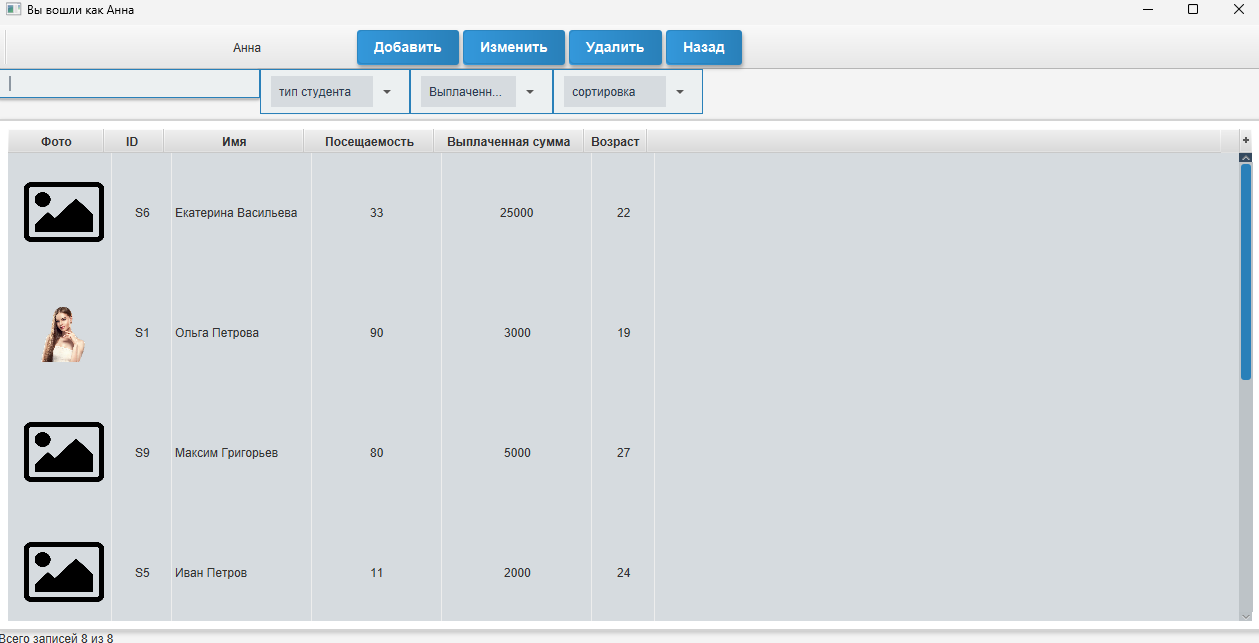
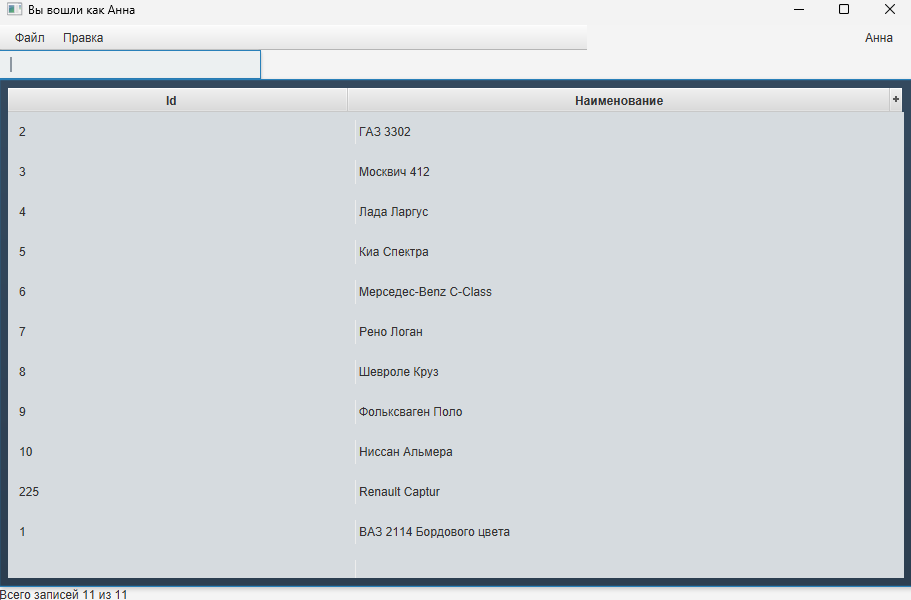


Рис.31 Страница Студенты

«Страница Авто»: эта форма предназначена для отображения списка автомобилей. (Рис. 32)

Рис.32 Страница Студенты

«Страница Категории обучения»: эта форма предназначена для отображения списка категорий обучений. (Рис. 33)

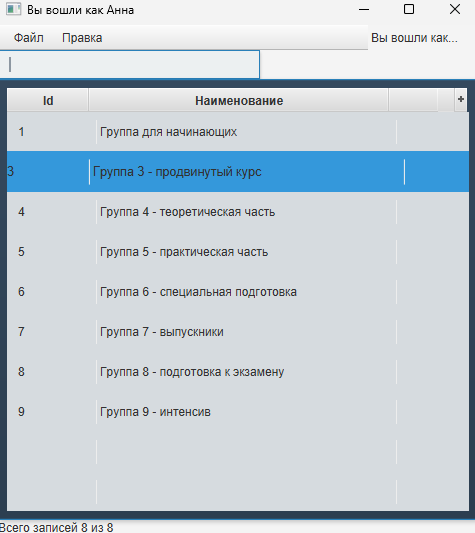


Рис.33 Страница категорий обучений

Работа с autoschool-app

При успешной авторизации в программу откроется главное окно. Для работы с данными используйте формы «Cтуденты», «Авто», «Категории обучения». В каждой форме вы можете:

•Добавить новую запись: Используйте кнопку «Добавить».

•Редактировать существующую запись: Используйте кнопку «Изменить».

•Удалить запись: Используйте кнопку «Удалить». Будьте осторожны, эта операция необратима.

Кнопки управления («Добавить», «Удалить», «Изменить») расположены на каждой форме.

Чтобы закрыть программу, закройте окно приложения.

Действия этих кнопок приведены в таблице 1:

Таблица-1 Действие кнопок

|  |  |
| --- | --- |
| Добавить запись | Прежде чем добавить запись в таблицу необходимо нажать на «Правка» → «Добавить». Поля формы будут очищены, пользователю требуется ввести необходимые данные. Если пользователь нажмет кнопку «Сохранить», то введенная информация добавится в БД. Если нажать кнопку «Отменить», то введенная информация пользователем удалится. |
| Удалить запись | Кнопка «Удалить» позволяет удалить конкретную запись. |
| Изменить данные | Кнопка «Изменить» открывает форму в режиме редактирования информации |
|  | «Поиск» позволяет найти нужную информацию. |

**4.4 Сообщения пользователю**

Для уверенной и комфортной работы пользователю важно понимать поведение программы, а программе, в свою очередь, поведение пользователя. Для информирования и уточнения намерений пользователей в программах используются сообщения.

При попытке удалить запись, пользователю выводится сообщение с запросом на подтверждение или отмену действия. (Рис.34)

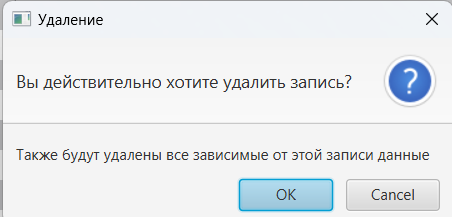


Рис.34 Сообщение Подтверждение удаления

После добавления новой записи, пользователю выводится сообщение. (Рис.35)

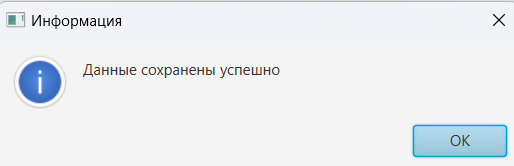


Рис.35 Сообщение Запись добавлена

После обновления записи, пользователю выводится сообщение. (Рис.36)

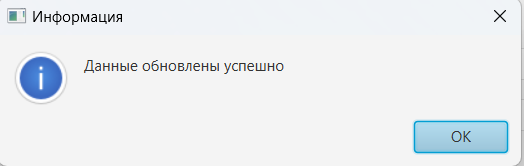


Рис.36 Сообщение Запись обновлена

При попытке неправильно ввода , программа выводит пользователю сообщение о невозможности действия. (Рис.37)

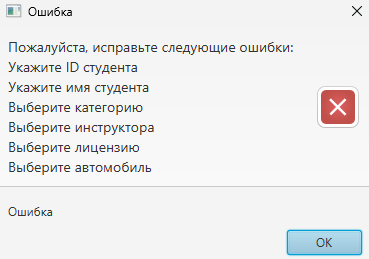


Рис.37 Сообщение Ошибка валидации

**5. РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Обоснование необходимого количества тестов**

Для разработанного алгоритма был построен граф.



Рис.27 Потоковый граф

Необходимое количество тестов определяется по формуле V(G) = E-N+2, где E – это число ребер, а N–это число вершин данного графа. Согласно представленному графу E=35, N=25. Далее по формуле рассчитаем необходимое количество тестовых сценариев (тест-кейсов).

V(G)=35-25+2=12.

Для данного программного средства необходимо разработать не менее 12 тестовых сценариев.

**5.2 Описание тестовых пакетов**

Таблица 2 – Тестирование корректного добавления информации в Студенты

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Студенты | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Студенты | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  student\_id = "1234567890"  name = "Иван Иванов"  age = 25  max\_payment\_amount = 50000 paid\_amount = 25000 attendance\_rate = 90 notes ="Отличный студент" license\_type\_id = 1  instructor\_id = 2  vehicle\_id = 3  group\_category\_id = 4  photo = [данные изображения] Статус = Завершилось  Цвет = Красный | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Студенты открыта | | Отображается форма Студенты |  |

Таблица 3 - Тестирование некорректного добавления информации в таблицу Студенты

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Студенты | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Студенты | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  student\_id = "1234567890"  name = "Иван Иванов"  age =  max\_payment\_amount = 50000 paid\_amount = 25000 attendance\_rate = 90 notes ="Отличный студент" license\_type\_id = 1  instructor\_id = 2  vehicle\_id = 3  group\_category\_id = 4  photo = [данные изображения] Статус =  Цвет = Красный | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, так как поле age пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Студенты открыта | | Отображается форма Студенты |  |

Таблица 4 – Тестирование корректного редактирования информации в таблице Cтуденты

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблицу Студенты | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Студенты | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  student\_id = "1234567890"  name = "Николай Николаевич"  age = 25  max\_payment\_amount = 50000 paid\_amount = 25000 attendance\_rate = 90 notes ="Отличный студент" license\_type\_id = 1  instructor\_id = 2  vehicle\_id = 3  group\_category\_id = 4  photo = [данные изображения] Статус =  Цвет = Красный | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Студенты открыта | | Отображается форма Студенты |  |

Таблица 5 – Тестирование некорректного редактирования информации в таблице Студенты

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблицу Студенты | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Студенты | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  student\_id = "1234567890"  name = "Николай Николаевич"  age =  max\_payment\_amount = 50000 paid\_amount = 25000 attendance\_rate = 90 notes ="Отличный студент" license\_type\_id =  instructor\_id = 2  vehicle\_id = 3  group\_category\_id = 4  photo = [данные изображения] Статус =  Цвет = Зеленый | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поле license\_type\_id пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Студенты открыта | | Отображается форма Студенты |  |

Таблица 6 – Тестирование корректного добавления информации в таблицу Авто

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Авто | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Авто | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  vehicle\_id = '1'  title = 'РеноЛоган'  Статус = Не началось | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Авто открыта | | Отображается форма Авто |  |

Таблица 7 – Тестирование некорректного добавления информации в таблицу Авто

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Авто | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Авто | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  vehicle\_id =  title =  Статус = Не началось | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Авто открыта | | Отображается форма Авто |  |

Таблица 8 – Тестирование корректного редактирования информации в таблице Авто

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблицу Авто | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Авто | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  vehicle\_id = ‘1’  title = ‘СИБИРЬ’  Статус = Не началось | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Авто открыта | | Отображается форма Авто |  |

Таблица 9 – Тестирование некорректного редактирования информации в таблице Авто

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблицу Авто | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Авто | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  vehicle\_id =  title =  Статус = Не началось | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Авто открыта | | Отображается форма Авто |  |

Таблица 10 – Тестирование корректного добавления информации в таблице Тип лицензии

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Тип лицензии | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Тип лицензии | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  license\_type\_id = ‘1’  title = ‘Лицензия категории A’  Статус = Не началось | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Тип лицензии открыта | | Отображается форма Тип лицензии |  |

Таблица 11 – Тестирование некорректного добавления информации в таблице Тип лицензии

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Тип лицензии | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Тип лицензии | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  license\_type\_id =  title = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, так как поле пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Тип лицензии открыта | | Отображается форма Тип лицензии |  |

Таблица 12 – Тестирование корректного редактирования информации в таблице Тип лицензии

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблицу Тип лицензии | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Тип лицензии | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  license\_type\_id = ’1’  title = ‘Лицензия категории A’ | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Тип лицензии открыта | | Отображается форма Тип лицензии |  |

Таблица 13 – Тестирование некорректного редактирования информации в таблице Тип лицензии

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблицу Тип лицензии | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Тип лицензии | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить Заполните форму  license\_type\_id =  title = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Тип лицензии открыта | | Отображается форма Тип лицензии |  |

**6. ЗАКЛЮЧЕНИЕ**

**6.1 Оценка качества программного средства с помощью метрик**

Оценка качества программного средства проводится исходя из стандарта оценки качества ISО 9126.

На компьютере следующей конфигурации:

Установленная операционная система – Windows 7 Ultimate 32бит.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 2 Gb

Таблица 14 – Требования к количественным характеристикам качества программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Надежность** |  | | |
| *Завершенность:*  *-* наработка на отказ при отсутствии рестарта; | Часы | 10 | 10 |
| *Устойчивость:*  - относительные ресурсы на обеспечение надежности и рестарта. | % | 0,5 | 0,5 |
| *Восстанавливаемость:*  -длительность восстановления. | Минуты | 2 | 2 |
| *Доступность-готовность:*  -относительное время работоспособного функционирования. | Вероятность | 0,998 | 0,998 |
| **Эффективность** |  | | |
| *Временная эффективность:*  *-*время отклика - получение результатов на типовое задание;  - пропускная способность число типовых заданий, исполняемых в единицу времени. | Секунды  Число в минуту | 3  60 | 3  60 |
| *Используемость ресурсов:*  *-* относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства. |  | 0,001 | 0,001 |

Таблица 15 – Требования к качественным характеристикам программного средства.

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Практичность** |  | | |
| *Простота использования:*  **-**среднее время ввода заданий;  -среднее время отклика на задание. | Секунды  Секунды | 10  3 | 10  3 |
| *Изучаемость :*  - трудоемкость изучения применения ПС;  -продолжительность изучения;  -объем эксплуатационной документации; | Чел.-часы  Часы  Страницы | 5  2  15 | 5  2  15 |
| **Сопровождаемость** |  | | |
| *Изменяемость:*  -трудоемкость подготовки изменений;  -длительность подготовки изменений. | Чел.-часы  Часы | 5  5 | 5  5 |
| *Тестируемость:*  *-трудоемкость тестирования изменений;*  *- длительность тестирования изменений.* | Чел.-часы  Часы | 3  3 | 3  3 |
| **Мобильность** |  | | |
| *Адаптируемость:*  - трудоемкость адаптации;  -длительность адаптации. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Простота установки:*  -трудоемкость инсталляции;  -длительность инсталляции. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Замещаемость:*  -трудоемкость замены компонентов;  - длительность замены компонентов. | Чел.-часы  Часы | 0,2  0,2 | 0,2  0,2 |

Таблица 16 – Сложность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип объекта | Количество | Вес | | | Итого |
|  |  | Простой | Средний | Сложный |  |
| Экран | 9 | x1 | x2 | **x3** | =27 |
| Отчет | 2 | x2 | **x5** | x8 | =10 |

Таблица 17 – Сложность экрана

|  |  |  |  |
| --- | --- | --- | --- |
| Экраны | Количество Категорииских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | **Всего <8** | Всего >8 |
|  | (<2 срв, <3 клт) | **(2-3 срв, 3-5 клт)** | (>3 срв, >5 клт) |
| <3 | Простой | Простой | Средний |
| 3-7 | Простой | Средний | Сложный |
| **>8** | Средний | **Сложный** | Сложный |

Таблица 18 – Сложность отчета

|  |  |  |  |
| --- | --- | --- | --- |
| Отчеты | Количество Категорииских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | Всего >8 |
|  | (<2 срв, <3 клт) | **(2-3 срв, 3-5 клт)** | (>3 срв, >5 клт) |
| 0 или 1 | Простой | Простой | Средний |
| **2 или 3** | Простой | **Средний** | Сложный |
| >4 | Средний | Сложный | Сложный |

Производительность = 4 тысяч строк кода/2 месяца= 2 тысяч строк кода/месяц

Качество = 2 ошибки /4 тысяч строк кода=0,5 ошибок/тысяч строк кода

**7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. К. Г. Хорстманн, Г. Корнелл - "Java. Полное руководство" (4-е издание, 2019) - 960 страниц

2. Брюс Эккель - "Философия Java" (3-е издание, 2007) - 864 страницы

3. С. Л. Мэтьюс - "Java. Руководство для начинающих" (2015) - 344 страницы

4. Кэтти Сьерра, Берти Бейтс - "Понимание Java. Как стать выдающимся программистом" (2006) - 600 страниц

5. Д. Фланаган - "Java в примерах" (2014) - 660 страниц

6. С. Б. Блейк - "Java: Библиотека программиста" (2017) - 500 страниц

7. Е. Б. Летунов - "Java. Разработка на платформе Java SE 8" (2016) - 480 страниц

8. М. Фаулер - "Java. Модернизируемый код" (2018) - 432 страницы

9. С. Оукс - "Java. Эффективная работа с памятью" (2013) - 480 страниц

10. Г. Л. Герберт - "Java. Практика программирования" (2012) - 400 страниц

11. ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению

12. ГОСТ 19.102-77.ЕСПД. Стадии разработки

13. ГОСТ 19.505-79.ЕСПД. Руководство оператора. Требования к содержанию и оформлению

14. ГОСТ 19.002-80.ЕСПД. Схемы алгоритмов и программ. Правила выполнения

15. ГОСТ 19.404-79.ЕСПД. Пояснительная записка. Требования к содержанию и оформлению

**Приложение А**

Разработка интегрированного программного модуля «Инвестиционная компания»

Текст кода программного средства

460.ЗМК.00010-01 12 01

**Текст кода программного средства**

**base-styles.css**

/\* Основные стили \*/

.root {

-fx-main-bg-color: #2C3E50; /\* Тёмно-синий фон \*/

-fx-font-family: "Arial", sans-serif;

}

/\* Кнопки \*/

.button {

-fx-border-style: solid;

-fx-border-width: 0.8px;

-fx-border-radius: 5px;

-fx-border-color: #2980B9; /\* Голубой \*/

-fx-background-color: #3498DB; /\* Светло-голубой \*/

-fx-padding: 8px 16px;

-fx-text-fill: #FFFFFF; /\* Белый текст \*/

-fx-font-size: 12px;

-fx-font-weight: bold;

-fx-cursor: hand;

-fx-effect: dropshadow(three-pass-box, rgba(0, 0, 0, 0.2), 4, 0, 0, 3);

-fx-transition: all 0.3s ease;

}

.button:hover {

-fx-background-color: #2980B9; /\* Голубой \*/

-fx-border-color: #3498DB;

-fx-text-fill: #ECF0F1; /\* Светлый текст \*/

-fx-scale-x: 1.1;

-fx-scale-y: 1.1;

}

.button:pressed {

-fx-background-color: #1F618D; /\* Тёмно-синий \*/

-fx-text-fill: #FFFFFF; /\* Белый текст \*/

}

/\* Поля ввода \*/

.text-field {

-fx-border-style: solid;

-fx-border-width: 1px;

-fx-border-color: #2980B9; /\* Голубой \*/

-fx-background-color: #ECF0F1; /\* Светлый фон \*/

-fx-padding: 6px 10px;

-fx-font-size: 12px;

-fx-text-fill: #2C3E50; /\* Тёмно-синий текст \*/

-fx-focus-color: #3498DB; /\* Светло-голубой при фокусе \*/

}

/\* Заголовки \*/

.header-label {

-fx-font-size: 18px;

-fx-font-weight: bold;

-fx-background-color: #2980B9; /\* Голубой \*/

-fx-text-fill: #FFFFFF; /\* Белый текст \*/

-fx-padding: 10px;

-fx-alignment: center;

}

/\* Списки \*/

.list-view {

-fx-background-color: #ECF0F1; /\* Светлый фон \*/

-fx-border-color: #2980B9; /\* Голубой \*/

-fx-border-width: 1px;

-fx-padding: 6px;

-fx-border-radius: 5px;

}

.list-cell {

-fx-font-size: 12px;

-fx-text-fill: #2C3E50; /\* Тёмно-синий текст \*/

-fx-background-color: #D6DBDF; /\* Светло-серый \*/

-fx-padding: 8px;

}

.list-cell:hover {

-fx-background-color: #3498DB; /\* Светло-голубой при наведении \*/

-fx-text-fill: #FFFFFF; /\* Белый текст \*/

}

/\* Панель прокрутки \*/

.scroll-bar {

-fx-background-color: #34495E; /\* Серо-синий \*/

}

.scroll-bar .track {

-fx-background-color: #BDC3C7; /\* Светло-серый \*/

}

.scroll-bar .thumb {

-fx-background-color: #2980B9; /\* Голубой \*/

}

/\* Прогресс-бары \*/

.progress-bar {

-fx-accent: #3498DB; /\* Светло-голубой \*/

-fx-background-color: #ECF0F1; /\* Светлый фон \*/

}

/\* Фильтры \*/

.combo-box {

-fx-background-color: #ECF0F1; /\* Светлый фон \*/

-fx-border-color: #2980B9; /\* Голубой \*/

-fx-font-size: 12px;

-fx-padding: 6px 10px;

-fx-text-fill: #2C3E50; /\* Тёмно-синий текст \*/

}

/\* Всплывающие подсказки \*/

.tooltip {

-fx-background-color: #2980B9; /\* Голубой \*/

-fx-text-fill: #FFFFFF; /\* Белый текст \*/

-fx-font-size: 12px;

-fx-padding: 6px;

}

/\* Таблица \*/

.table-view {

-fx-background-color: #34495E; /\* Серо-синий фон \*/

-fx-border-color: #2980B9; /\* Голубой \*/

-fx-border-width: 1.2px;

-fx-padding: 8px;

-fx-effect: dropshadow(three-pass-box, rgba(0, 0, 0, 0.2), 6, 0, 2, 2);

}

.table-row-cell {

-fx-background-color: #D6DBDF; /\* Светло-серый \*/

-fx-text-fill: #2C3E50; /\* Тёмно-синий текст \*/

-fx-font-size: 12px;

-fx-padding: 8px;

}

.table-row-cell:selected {

-fx-background-color: #2980B9; /\* Голубой \*/

-fx-text-fill: #FFFFFF; /\* Белый текст при выделении \*/

}

.table-row-cell:hover {

-fx-background-color: #3498DB; /\* Светло-голубой при наведении \*/

-fx-text-fill: #ECF0F1; /\* Светлый текст \*/

-fx-scale-x: 1.03;

-fx-scale-y: 1.03;

}

/\* Заголовки колонок \*/

.table-column-header {

-fx-background-color: #2980B9; /\* Голубой \*/

-fx-text-fill: #FFFFFF; /\* Белый текст \*/

-fx-font-size: 14px;

-fx-font-weight: bold;

-fx-alignment: center;

-fx-padding: 10px;

}

.table-column-header:hover {

-fx-background-color: #1F618D; /\* Тёмно-синий \*/

}

.table-column-header .label {

-fx-text-fill: #FFFFFF; /\* Белый текст \*/

}

/\* Тень для поля ввода \*/

.text-field {

-fx-effect: dropshadow(gaussian, rgba(0, 0, 0, 0.3), 4, 0, 0, 2);

}

/\* Тень для кнопок \*/

.button {

-fx-effect: dropshadow(gaussian, rgba(0, 0, 0, 0.3), 6, 0, 1, 3);

}

/\* Анимация при наведении на строки таблицы \*/

.table-row-cell {

-fx-transition: background-color 0.3s ease, transform 0.2s ease;

}

.table-row-cell:hover {

-fx-background-color: #3498DB; /\* Светло-голубой при наведении \*/

-fx-text-fill: #ECF0F1; /\* Светлый текст \*/

-fx-scale-x: 1.05;

-fx-scale-y: 1.05;

}

/\* Иконки на кнопках \*/

.button .graphic {

-fx-padding: 0 8px 0 0; /\* Отступы между иконкой и текстом \*/

-fx-alignment: center-left; /\* Выравнивание \*/

}

.button .text {

-fx-font-size: 14px;

}

/\* Градиентный фон для кнопок \*/

.button {

-fx-background-color: linear-gradient(to right, #3498DB, #2980B9);

}

/\* Градиент для фона таблицы \*/

.table-view {

-fx-background-color: linear-gradient(to bottom, #34495E, #2C3E50);

}

/\* Уведомления \*/

.notification {

-fx-background-color: #E74C3C; /\* Красный для ошибок \*/

-fx-text-fill: #FFFFFF; /\* Белый текст \*/

-fx-padding: 10px;

-fx-font-size: 14px;

}

/\* Успешное сообщение \*/

.success-message {

-fx-background-color: #2ECC71; /\* Зеленый для успеха \*/

-fx-text-fill: #FFFFFF; /\* Белый текст \*/

-fx-padding: 10px;

-fx-font-size: 14px;

}

/\* Градиент для полосы прокрутки \*/

.scroll-bar {

-fx-background-color: linear-gradient(to top, #BDC3C7, #34495E);

}

**Класс Manager**

package org.main.autoschoolapp.util;

import javafx.application.Platform;

import javafx.fxml.FXMLLoader;

import javafx.geometry.Rectangle2D;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

import javafx.stage.Modality;

import javafx.stage.Screen;

import javafx.stage.Stage;

import javafx.stage.Window;

import org.main.autoschoolapp.AutoschoolApp;

import org.main.autoschoolapp.model.GroupCategory;

import org.main.autoschoolapp.model.Student;

import org.main.autoschoolapp.model.User;

import org.main.autoschoolapp.model.Vehicle;

import java.io.IOException;

import java.util.Optional;

public class Manager {

public static User currentUser = null;

public static Stage mainStage;

public static Stage secondStage;

public static Stage currentStage;

public static Student currentStudent;

public static Vehicle currentVehicle;

public static GroupCategory currentGroupCategory;

public static Window primaryStage;

public static void ShowEditWindow(String fxmlFileName) {

Stage newWindow = new Stage();

FXMLLoader fxmlLoader = new FXMLLoader(AutoschoolApp.class.getResource(fxmlFileName));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

} catch (IOException e) {

throw new RuntimeException(e);

}

newWindow.setTitle("Изменить данные");

newWindow.initOwner(Manager.secondStage);

newWindow.setResizable(false);

newWindow.initModality(Modality.WINDOW\_MODAL);

newWindow.setScene(scene);

Manager.currentStage = newWindow;

newWindow.showAndWait();

Manager.currentStage = null;

}

public static void LoadSecondStageScene(String fxmlFileName, String title) {

FXMLLoader fxmlLoader = new FXMLLoader(AutoschoolApp.class.getResource(fxmlFileName));

Scene scene = null;

try {

// Устанавливаем фиксированные размеры для сцены

scene = new Scene(fxmlLoader.load(), 1280, 660); // Ширина 1280px, высота 660px

scene.getStylesheets().add("base-styles.css");

// Устанавливаем сцену для второго окна

Manager.secondStage.setScene(scene);

// Устанавливаем окно с заданным размером

Manager.secondStage.setWidth(1280); // Ширина 1280px

Manager.secondStage.setHeight(660); // Высота 660px

// Убираем максимизацию, если она была

Manager.secondStage.setMaximized(false);

Manager.secondStage.setTitle(title);

} catch (IOException e) {

throw new RuntimeException(e);

}

}

public static void ShowPopup() {

Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

alert.setTitle("Закрыть приложение");

alert.setHeaderText("Вы хотите выйти из приложения?");

alert.setContentText("Все несохраненные данные, будут утеряны");

Optional<ButtonType> result = alert.showAndWait();

if (result.get() == ButtonType.OK) {

Platform.exit();

}

}

public static void ShowErrorMessageBox(String message) {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Ошибка");

alert.setHeaderText(message);

alert.showAndWait();

}

public static void MessageBox(String title, String header, String message, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(header);

alert.setContentText(message);

alert.showAndWait();

}

public static Optional<ButtonType> ShowConfirmPopup() {

Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

alert.setTitle("Удаление");

alert.setHeaderText("Вы действительно хотите удалить запись?");

alert.setContentText("Также будут удалены все зависимые от этой записи данные");

Optional<ButtonType> result = alert.showAndWait();

return result;

}

}

**Класс MakeCaptcha**

package org.main.autoschoolapp.util;

import javafx.embed.swing.SwingFXUtils;

import javafx.scene.image.Image;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.io.IOException;

import java.util.Random;

public class MakeCaptcha {

private static String captchaCode;

public static Image CreateImage(int width, int height, int symbolCount) throws IOException {

Random rnd = new Random();

//Создадим изображение

BufferedImage result = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

Graphics2D g2d = result.createGraphics();

g2d.setColor(Color.lightGray);

g2d.fillRect(0, 0, width, height);

//draw a string

//Сгенерируем текст

char symbol;

// нбор допустимых символов

String alphabet = "23456789QWERTYUPASDFGHJKLZXCVBNM";

// текст капчи

String captcha = "";

// размер поля для одного символа

int h = width / symbolCount;

for (int i = 0; i < symbolCount; ++i) {

// генерируем размер буквы

int size = rnd.nextInt(20) + 20;

g2d.setFont(new Font("Arial", Font.BOLD, size));

// выбираем любой символ из допустимого набора

symbol = alphabet.charAt(rnd.nextInt(alphabet.length()));

g2d.setPaint(getRandomColor());

// текст капчи

captcha += symbol;

// генерируем позиции рисования символа

int Xpos = rnd.nextInt(h / 2) + h \* i;

int Ypos = rnd.nextInt(height / 2) + 20;

g2d.drawString(String.valueOf(symbol), Xpos, Ypos);

}

// add lines

for (int i = 0; i < 10; ++i) {

g2d.setPaint(getRandomColor());

int x1 = rnd.nextInt(width / 10);

int y1 = rnd.nextInt(height);

int x2 = rnd.nextInt(width);

int y2 = rnd.nextInt(height);

g2d.drawLine(x1, y1, x2, y2);

}

for (int i = 0; i < 100; ++i) {

g2d.setPaint(getRandomColor());

int x1 = rnd.nextInt(width);

int y1 = rnd.nextInt(height);

g2d.draw(new Rectangle(x1, y1, 1, 1));

}

captchaCode = captcha;

//disposes of this graphics context and releases any system resources that it is using

g2d.dispose();

return SwingFXUtils.toFXImage(result, null);

}

public static String captchaCode() {

return captchaCode;

}

public static Color getRandomColor()

{

Random rnd = new Random();

int red = rnd.nextInt(256);

int green = rnd.nextInt(256);

int blue = rnd.nextInt(256);

return new Color(red, green, blue);

}

}

**Класс HibernateSessionFactoryUtil**

package org.main.autoschoolapp.util;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateSessionFactoryUtil {

private static SessionFactory sessionFactory;

static {

try {

sessionFactory = new Configuration().configure().buildSessionFactory();

} catch (Exception e) {

e.printStackTrace();

throw new ExceptionInInitializerError("Initial SessionFactory creation failed: " + e);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

**Файл hibernate.cfg.xml**

<?xml version = "1.0" encoding = "utf-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- Set URL -->

<property name = "hibernate.connection.url">jdbc:postgresql://localhost:5433/postgres</property>

<!-- Set User Name -->

<property name = "hibernate.connection.username">postgres</property>

<!-- Set Password -->

<property name = "hibernate.connection.password">111</property>

<!-- Set Driver Name -->

<property name = "hibernate.connection.driver\_class">org.postgresql.Driver</property>

<property name = "hibernate.show\_sql">true</property>

<property name="hibernate.show\_sql">true</property>

<property name="hibernate.format\_sql">true</property>

<property name="hibernate.use\_sql\_comments">true</property>

<property name="hibernate.logger.org.hibernate.SQL">DEBUG</property>

<property name="hibernate.logger.org.hibernate.type.descriptor.sql.BasicBinder">TRACE</property>

<property name="hibernate.bytecode.use\_reflection\_optimizer">false</property>

<!-- Optional: Auto-generate schema -->

<!-- <property name = "hibernate.hbm2ddl.auto">create</property> -->

<mapping class="org.main.autoschoolapp.model.ClassRoom" />

<mapping class="org.main.autoschoolapp.model.GroupCategory" />

<mapping class="org.main.autoschoolapp.model.Instructor" />

<mapping class="org.main.autoschoolapp.model.Lesson" />

<mapping class="org.main.autoschoolapp.model.LessonAttendance" />

<mapping class="org.main.autoschoolapp.model.LicenseType" />

<mapping class="org.main.autoschoolapp.model.Role" />

<mapping class="org.main.autoschoolapp.model.Status" />

<mapping class="org.main.autoschoolapp.model.Student" />

<mapping class="org.main.autoschoolapp.model.User" />

<mapping class="org.main.autoschoolapp.model.Vehicle" />

</session-factory>

</hibernate-configuration>

**Файл интерфейса страницы авто**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.geometry.Insets?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.Menu?>  
<?import javafx.scene.control.MenuBar?>  
<?import javafx.scene.control.MenuItem?>  
<?import javafx.scene.control.TableColumn?>  
<?import javafx.scene.control.TableView?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.BorderPane?>  
<?import javafx.scene.layout.FlowPane?>  
<?import javafx.scene.layout.HBox?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="400" minWidth="650" prefHeight="500.0" prefWidth="650.0" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.main.autoschoolapp.controllers.VehicleTableViewController">  
 <children>  
 <BorderPane prefHeight="200.0" prefWidth="200.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">  
 <top>  
 <HBox BorderPane.alignment="CENTER">  
 <children>  
 <MenuBar prefHeight="25.0" prefWidth="381.0" HBox.hgrow="ALWAYS">  
 <menus>  
 <Menu mnemonicParsing="false" text="Файл">  
 <items>  
 <MenuItem fx:id="MenuItemBack" mnemonicParsing="false" onAction="#MenuItemBackAction" text="Назад" />  
 </items>  
 </Menu>  
 <Menu mnemonicParsing="false" text="Правка">  
 <items>  
 <MenuItem fx:id="MenuItemAdd" mnemonicParsing="false" onAction="#MenuItemAddAction" text="Добавить" />  
 <MenuItem fx:id="MenuItemUpdate" mnemonicParsing="false" onAction="#MenuItemUpdateAction" text="Изменить" />  
 <MenuItem fx:id="MenuItemDelete" mnemonicParsing="false" onAction="#MenuItemDeleteAction" text="Удалить" />  
 </items>  
 </Menu>  
  
 </menus>  
 </MenuBar>  
 <Label fx:id="LabelUser" alignment="CENTER\_RIGHT" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" prefWidth="120.0" styleClass="user-label" text="Label" HBox.hgrow="ALWAYS">  
 <padding>  
 <Insets right="20.0" />  
 </padding>  
 </Label>  
 </children>  
 </HBox>  
 </top>  
 <center>  
 <BorderPane BorderPane.alignment="CENTER">  
 <top>  
 <FlowPane minHeight="-Infinity" nodeOrientation="LEFT\_TO\_RIGHT" prefWidth="200.0" rowValignment="TOP" BorderPane.alignment="CENTER">  
 <children>  
 <TextField fx:id="TextFieldSearch" onAction="#TextFieldSearchAction" prefHeight="25.0" prefWidth="262.0" promptText="Введите название для поиска" />  
 </children>  
 </FlowPane>  
 </top>  
 <center>  
 <TableView fx:id="TableViewVehicles" prefHeight="200.0" prefWidth="200.0" tableMenuButtonVisible="true" BorderPane.alignment="CENTER">  
 <columns>  
 <TableColumn id="TableColumnVehicleId" fx:id="TableColumnVehicleId" minWidth="0.0" prefWidth="93.0" text="Id" />  
 <TableColumn id="TableColumnName" fx:id="TableColumnModel" maxWidth="1.7976931348623157E308" prefWidth="556.0" resizable="false" text="Наименование" />  
 </columns>  
 </TableView>  
 </center>  
 </BorderPane>  
 </center>  
 <bottom>  
 <Label fx:id="LabelInfo" text="Label" BorderPane.alignment="CENTER\_LEFT" />  
 </bottom>  
 </BorderPane>  
 </children>  
</AnchorPane>

**Файл интерфейса страницы редактирования авто**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.geometry.Insets?>  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.ButtonBar?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.BorderPane?>  
<?import javafx.scene.layout.GridPane?>  
<?import javafx.scene.layout.VBox?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="150.0" prefWidth="550.0" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.main.autoschoolapp.controllers.VehicleEditViewController">  
 <children>  
 <BorderPane layoutX="170.0" layoutY="47.0" prefHeight="200.0" prefWidth="200.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">  
 <bottom>  
 <ButtonBar prefHeight="40.0" prefWidth="200.0" BorderPane.alignment="CENTER">  
 <buttons>  
 <Button fx:id="BtnSave" defaultButton="true" mnemonicParsing="false" onAction="#BtnSaveAction" text="Сохранить" />  
 <Button fx:id="BtnCancel" cancelButton="true" mnemonicParsing="false" onAction="#BtnCancelAction" text="Отмена" />  
 </buttons>  
 <padding>  
 <Insets right="20.0" />  
 </padding>  
 </ButtonBar>  
 </bottom>  
 <center>  
 <VBox BorderPane.alignment="CENTER">  
 <Label maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" text="Название" GridPane.rowIndex="0" />  
 <TextField fx:id="TextFieldModel" GridPane.columnIndex="1" GridPane.rowIndex="0" />  
 <BorderPane.margin>  
 <Insets left="5.0" right="5.0" top="20.0" />  
 </BorderPane.margin>  
  
 </VBox>  
 </center>  
 </BorderPane>  
 </children>  
</AnchorPane>

**Файл интерфейса страницы студента**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.ComboBox?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.Separator?>  
<?import javafx.scene.control.TableColumn?>  
<?import javafx.scene.control.TableView?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.control.ToolBar?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.BorderPane?>  
<?import javafx.scene.layout.FlowPane?>  
  
<?import java.net.URL?>  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.main.autoschoolapp.controllers.StudentTableViewController">  
 <children>  
 <BorderPane prefHeight="200.0" prefWidth="200.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">  
 <top>  
 <ToolBar prefHeight="40.0" prefWidth="200.0" BorderPane.alignment="CENTER\_RIGHT">  
 <items>  
 <Separator halignment="LEFT" maxWidth="1.7976931348623157E308" prefHeight="30.0" prefWidth="225.0" />  
 <Label fx:id="LabelUser" prefWidth="120.0" text="Label" />  
 <Button fx:id="BtnAdd" mnemonicParsing="false" onAction="#BtnAddAction" text="Добавить" />  
 <Button fx:id="BtnUpdate" mnemonicParsing="false" onAction="#BtnUpdateAction" text="Изменить" />  
 <Button fx:id="BtnDelete" mnemonicParsing="false" onAction="#BtnDeleteAction" text="Удалить" />  
 <Button fx:id="BtnBack" alignment="CENTER\_RIGHT" mnemonicParsing="false" onAction="#BtnBackAction" text="Назад" textAlignment="RIGHT" />  
 </items>  
 </ToolBar>  
 </top>  
 <center>  
 <BorderPane BorderPane.alignment="CENTER">  
 <top>  
 <FlowPane minHeight="-Infinity" nodeOrientation="LEFT\_TO\_RIGHT" prefHeight="50.0" prefWidth="200.0" rowValignment="TOP" BorderPane.alignment="CENTER">  
 <children>  
 <TextField fx:id="TextFieldSearch" onAction="#TextFieldSearchAction" onInputMethodTextChanged="#TextFieldTextChanged" prefHeight="25.0" prefWidth="262.0" promptText="Введите название для поиска" />  
 <ComboBox fx:id="ComboBoxStudentType" onAction="#ComboBoxStudentTypeAction" prefWidth="150.0" promptText="тип студента" />  
 <ComboBox fx:id="ComboBoxPaidAmount" onAction="#ComboBoxPaidAmount" prefHeight="26.0" prefWidth="143.0" promptText="Выплаченная сумма" />  
 <ComboBox fx:id="ComboBoxSort" onAction="#ComboBoxSortAction" prefWidth="150.0" promptText="сортировка" />  
 </children>  
 </FlowPane>  
 </top>  
 <center>  
 <TableView fx:id="TableViewStudents"  
 fixedCellSize="120.0"  
 prefHeight="400.0"  
 prefWidth="600.0"  
 tableMenuButtonVisible="true"  
 BorderPane.alignment="CENTER"  
 style="-fx-border-color: lightgray; -fx-border-width: 2px; -fx-background-color: white;">  
 <columns>  
 <TableColumn id="TableColumnPhoto"  
 fx:id="TableColumnPhoto"  
 prefWidth="96.0"  
 resizable="false"  
 text="Фото"  
 style="-fx-alignment: CENTER;">  
 </TableColumn>  
 <TableColumn id="TableColumnStudentId"  
 fx:id="TableColumnStudentId"  
 minWidth="0.0"  
 prefWidth="60.0"  
 text="ID "  
 style="-fx-alignment: CENTER;">  
 </TableColumn>  
 <TableColumn id="TableColumnName"  
 fx:id="TableColumnName"  
 maxWidth="1.7976931348623157E308"  
 prefWidth="140.0"  
 resizable="false"  
 text="Имя"  
 style="-fx-alignment: CENTER-LEFT;">  
 </TableColumn>  
 <TableColumn id="TableColumnAttendanceRate"  
 fx:id="TableColumnAttendanceRate"  
 prefWidth="130.0"  
 text="Посещаемость"  
 style="-fx-alignment: CENTER;">  
 </TableColumn>  
 <TableColumn id="TableColumnPaidAmount"  
 fx:id="TableColumnPaidAmount"  
 prefWidth="150.0"  
 text="Выплаченная сумма"  
 style="-fx-alignment: CENTER;">  
 </TableColumn>  
 <TableColumn id="TableColumnAge"  
 fx:id="TableColumnAge"  
 prefWidth="80.0"  
 text="Возраст"  
 style="-fx-alignment: CENTER;">  
 </TableColumn>  
 </columns>  
 </TableView>  
  
 </center>  
 </BorderPane>  
 </center>  
 <bottom>  
 <Label fx:id="LabelInfo" text="Label" BorderPane.alignment="CENTER\_LEFT" />  
 </bottom>  
 </BorderPane>  
 </children>  
</AnchorPane>

**Файл страницы редактирования студента**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.geometry.Insets?>  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.ButtonBar?>  
<?import javafx.scene.control.ComboBox?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TextArea?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.image.ImageView?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.BorderPane?>  
<?import javafx.scene.layout.ColumnConstraints?>  
<?import javafx.scene.layout.GridPane?>  
<?import javafx.scene.layout.RowConstraints?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="758.0" prefWidth="967.0" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.main.autoschoolapp.controllers.StudentEditViewController">  
 <children>  
 <BorderPane layoutX="170.0" layoutY="47.0" prefHeight="200.0" prefWidth="200.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">  
 <bottom>  
 <ButtonBar prefHeight="40.0" prefWidth="200.0" BorderPane.alignment="CENTER">  
 <buttons>  
 <Button fx:id="BtnSave" defaultButton="true" mnemonicParsing="false" onAction="#BtnSaveAction" text="Сохранить" />  
 <Button fx:id="BtnCancel" cancelButton="true" mnemonicParsing="false" onAction="#BtnCancelAction" text="Отмена" />  
 </buttons>  
 <padding>  
 <Insets right="20.0" />  
 </padding>  
 </ButtonBar>  
 </bottom>  
 <center>  
 <GridPane hgap="20" vgap="20" BorderPane.alignment="CENTER">  
 <columnConstraints>  
 <ColumnConstraints hgrow="SOMETIMES" maxWidth="190.0" minWidth="10.0" prefWidth="190.0" />  
 <ColumnConstraints hgrow="SOMETIMES" maxWidth="680.0" minWidth="10.0" prefWidth="610.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="ALWAYS" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" />  
 <RowConstraints minHeight="10.0" prefHeight="100.0" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" />  
 <RowConstraints />  
 <RowConstraints />  
 </rowConstraints>  
 <children>  
 <Label maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" text="Имя" GridPane.rowIndex="1" />  
 <TextField fx:id="TextFieldName" onAction="#TextFieldName" GridPane.columnIndex="1" GridPane.rowIndex="1" />  
  
 <Label text="Описание" GridPane.rowIndex="3" />  
 <TextArea fx:id="TextAreaNote" onDragDetected="#TextAreaDescription" prefHeight="200.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="3" />  
  
 <Label text="Стоимость обучения" GridPane.rowIndex="4" />  
 <TextField fx:id="TextFieldMaxPaymentAmount" onAction="#TextFieldMaxPayment" text="Стоимость обучения" GridPane.columnIndex="1" GridPane.rowIndex="4" />  
  
 <Label text="Категория студента" GridPane.rowIndex="2" />  
 <ComboBox fx:id="ComboBoxGroupCategory" onAction="#ComboBoxCategory" prefHeight="25.0" prefWidth="300.0" GridPane.columnIndex="1" GridPane.rowIndex="2" />  
  
 <Label text="Общая посещаемость обучения" GridPane.rowIndex="6" />  
  
 <Label text="Инструктор" GridPane.rowIndex="7" />  
 <ComboBox fx:id="ComboBoxInstructor" onAction="#ComboBoxInstructor" prefHeight="25.0" prefWidth="300.0" GridPane.columnIndex="1" GridPane.rowIndex="7" />  
  
 <Label text="Автомобиль" GridPane.rowIndex="8" />  
 <ComboBox fx:id="ComboBoxVehicle" onAction="#ComboBoxVehicle" prefHeight="25.0" prefWidth="300.0" GridPane.columnIndex="1" GridPane.rowIndex="8" />  
 <TextField fx:id="TextFieldAge" onAction="#TextFieldAge" GridPane.columnIndex="1" GridPane.rowIndex="5" />  
 <Label text="Возраст" GridPane.rowIndex="5" />  
 <TextField fx:id="TextFieldAttendanceRate" onAction="#TextFieldAttendanceRate" GridPane.columnIndex="1" GridPane.rowIndex="6" />  
  
 <Label text="Тип лицензии" GridPane.rowIndex="11" />  
 <ComboBox fx:id="ComboBoxLicenseType" onAction="#ComboBoxLicenseType" prefHeight="25.0" prefWidth="300.0" GridPane.columnIndex="1" GridPane.rowIndex="11" />  
  
 <Label text="Изображение" />  
 <ImageView fx:id="ImageViewPhoto" fitHeight="75.0" fitWidth="124.0" pickOnBounds="true" preserveRatio="true" GridPane.columnIndex="1" />  
  
 <Button fx:id="BtnLoadImage" mnemonicParsing="false" onAction="#BtnLoadImageAction" text="Загрузить" GridPane.columnIndex="1" GridPane.valignment="BOTTOM">  
 <GridPane.margin>  
 <Insets left="200.0" />  
 </GridPane.margin>  
 </Button>  
  
 <Label text="ID Студента" GridPane.rowIndex="9" />  
 <TextField fx:id="TextFieldIDStudent" onAction="#TextFieldIDStudent" GridPane.columnIndex="1" GridPane.rowIndex="9" />  
  
 <TextField fx:id="TextFieldPaidAmount" onAction="#TextFieldPaidAmount" text="Осталось оплатить" GridPane.columnIndex="1" GridPane.rowIndex="10" />  
 <TextField fx:id="TextFieldDiscountAmountMax" onAction="#TextFieldDiscountAmountMax" GridPane.columnIndex="1" GridPane.rowIndex="10" />  
 <Label text="Осталось оплатить" GridPane.rowIndex="10" />  
 </children>  
 </GridPane>  
 </center>  
 </BorderPane>  
 </children>  
</AnchorPane>

**Файл интерфейса страницы отображения студентов**   
<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.geometry.Insets?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.image.ImageView?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.ColumnConstraints?>  
<?import javafx.scene.layout.GridPane?>  
<?import javafx.scene.layout.RowConstraints?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="1.7976931348623157E308" minHeight="-Infinity" minWidth="-Infinity" prefHeight="234.0" prefWidth="756.0" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.main.autoschoolapp.controllers.ListCellController">  
 <GridPane hgap="20" vgap="15" AnchorPane.bottomAnchor="10" AnchorPane.leftAnchor="10" AnchorPane.rightAnchor="10" AnchorPane.topAnchor="10">  
 <columnConstraints>  
 <ColumnConstraints minWidth="100.0" prefWidth="120.0" />  
 <ColumnConstraints hgrow="ALWAYS" maxWidth="474.0" minWidth="10.0" prefWidth="374.0" />  
 <ColumnConstraints maxWidth="202.0" minWidth="120.0" prefWidth="202.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints minHeight="40.0" prefHeight="50.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="40.0" prefHeight="50.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="40.0" prefHeight="50.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="40.0" prefHeight="50.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="40.0" prefHeight="50.0" vgrow="SOMETIMES" />  
 </rowConstraints>  
  
 <!-- ImageView for student photo -->  
 <ImageView fx:id="ImageViewPhoto" fitHeight="120.0" fitWidth="80.0" pickOnBounds="true" preserveRatio="true" GridPane.columnIndex="0" GridPane.rowSpan="4" />  
  
 <!-- Attendance rate label -->  
 <Label fx:id="LabelAttendanceRate" alignment="CENTER" prefHeight="80.0" prefWidth="89.0" text="Оценка" textAlignment="CENTER" GridPane.columnIndex="2" GridPane.rowIndex="1" GridPane.rowSpan="4">  
 <font>  
 <Font size="12.0" />  
 </font>  
 </Label>  
  
 <!-- Student name label -->  
 <Label fx:id="LabelName" alignment="CENTER\_LEFT" text="Имя студента:" GridPane.columnIndex="1" GridPane.rowIndex="0">  
 <font>  
 <Font size="14.0" />  
 </font>  
 <padding>  
 <Insets left="10.0" />  
 </padding>  
 </Label>  
  
 <!-- Instructor name label -->  
 <Label fx:id="LabelInstructor" alignment="CENTER\_LEFT" text="Имя инструктора" GridPane.columnIndex="1" GridPane.rowIndex="1">  
 <font>  
 <Font size="14.0" />  
 </font>  
 <padding>  
 <Insets left="10.0" />  
 </padding>  
 </Label>  
  
 <!-- Max payment amount label -->  
 <Label fx:id="LabelMaxPaymentAmount" alignment="CENTER\_LEFT" text="Цена:" GridPane.columnIndex="1" GridPane.rowIndex="2">  
 <font>  
 <Font size="14.0" />  
 </font>  
 <padding>  
 <Insets left="10.0" />  
 </padding>  
 </Label>  
 </GridPane>  
</AnchorPane>

Файл интерфейса расписания

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<?import javafx.scene.text.\*?>

<BorderPane xmlns="http://javafx.com/javafx" xmlns:fx="http://javafx.com/fxml"

fx:controller="org.main.autoschoolapp.controllers.ScheduleViewController">

<top>

<Label text="Расписание уроков" styleClass="header"/>

</top>

<center>

<TableView fx:id="lessonTable">

<columns>

<TableColumn fx:id="lessonIdColumn" text="ID"/>

<TableColumn fx:id="startDateColumn" text="Дата начала"/>

<TableColumn fx:id="endDateColumn" text="Дата окончания"/>

<TableColumn fx:id="classRoomColumn" text="Кабинет"/>

<TableColumn fx:id="statusColumn" text="Статус"/>

</columns>

</TableView>

</center>

<bottom>

<HBox spacing="10" alignment="CENTER">

<Button text="Добавить" onAction="#handleAdd"/>

<Button text="Удалить" onAction="#handleDelete"/>

<Button text="Распечатать" onAction="#handlePrint"/>

</HBox>

</bottom>

</BorderPane>

**Файл интерфейса главного окна**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.ComboBox?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.ListView?>  
<?import javafx.scene.control.Separator?>  
<?import javafx.scene.control.SplitPane?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.control.ToolBar?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.BorderPane?>  
<?import javafx.scene.layout.FlowPane?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="586.0" prefWidth="917.0" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.main.autoschoolapp.controllers.MainWindowController">  
  
 <children>  
 <!-- BorderPane as the main container -->  
 <BorderPane prefHeight="625.0" prefWidth="951.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">  
  
 <!-- SplitPane inside BorderPane center -->  
 <center>  
 <SplitPane dividerPositions="0.2" orientation="VERTICAL" prefHeight="400.0" prefWidth="800.0" BorderPane.alignment="CENTER">  
  
 <items>  
 <!-- First AnchorPane with FlowPane for search and combo boxes -->  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="150.0" prefWidth="800.0">  
  
 <children>  
 <FlowPane alignment="CENTER\_LEFT" hgap="10" layoutX="10.0" layoutY="10.0" nodeOrientation="LEFT\_TO\_RIGHT" prefHeight="150.0" prefWidth="780.0" vgap="10" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="10.0" AnchorPane.rightAnchor="10.0" AnchorPane.topAnchor="10.0">  
  
 <children>  
 <!-- Search TextField -->  
 <TextField fx:id="TextFieldSearch" onAction="#TextFieldSearchAction" prefHeight="30.0" prefWidth="300.0" promptText="Введите имя студента для поиска" />  
  
 <!-- Student Type ComboBox -->  
 <ComboBox fx:id="CmboBoxStudentType" onAction="#CmboBoxStudentTypeAction" prefWidth="160.0" promptText="Тип студента" />  
 <ComboBox fx:id="ComboBoxDiscount" onAction="#filterData" prefWidth="160.0" promptText="Посещаемость" />  
  
 <!-- Sort ComboBox -->  
 <ComboBox fx:id="ComboboxSort" onAction="#ComboboxSortAction" prefWidth="160.0" promptText="Сортировка" />  
 </children>  
 </FlowPane>  
 </children>  
 </AnchorPane>  
  
 <!-- Second AnchorPane with ListView -->  
 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="250.0" prefWidth="800.0">  
  
 <children>  
 <ListView fx:id="ListViewStudent" layoutX="-28.0" layoutY="-23.0" prefHeight="378.0" prefWidth="849.0" AnchorPane.bottomAnchor="43.0" AnchorPane.leftAnchor="-28.0" AnchorPane.rightAnchor="48.0" AnchorPane.topAnchor="-23.0" />  
 <Button fx:id="BtnJournal" layoutX="28.0" layoutY="387.0" mnemonicParsing="false" onAction="#BtnJournalAction" text="Журнал" />  
 </children>  
 </AnchorPane>  
 </items>  
 </SplitPane>  
 </center>  
  
 <!-- ToolBar at the top of the BorderPane -->  
 <top>  
 <ToolBar prefHeight="40.0" prefWidth="800.0" BorderPane.alignment="CENTER\_RIGHT">  
  
 <items>  
 <Separator halignment="LEFT" maxWidth="1.7976931348623157E308" prefHeight="30.0" prefWidth="320.0" />  
  
 <Label fx:id="LabelUser" prefWidth="120.0" />  
  
 <Button fx:id="BtnStudents" mnemonicParsing="false" onAction="#BtnStudentsAction" text="Студенты" />  
 <Button fx:id="BtnVehicle" mnemonicParsing="false" onAction="#BtnGoVehicleAction" text="Авто" />  
 <Button fx:id="BtnGroupCategory" mnemonicParsing="false" onAction="#BtnGroupCategoryAction" text="Категории обучения" />  
  
 <Button fx:id="BtnBack" alignment="CENTER\_RIGHT" mnemonicParsing="false" onAction="#BtnBackAction" text="Назад" textAlignment="RIGHT" />  
 </items>  
 </ToolBar>  
 </top>  
  
 <!-- Bottom Label in BorderPane -->  
 <bottom>  
 <Label fx:id="LabelInfo" text="Label" BorderPane.alignment="CENTER\_LEFT" />  
 </bottom>  
 </BorderPane>  
 </children>  
</AnchorPane>

**Файл интерфейса авторизации**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.geometry.Insets?>  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.PasswordField?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.image.Image?>  
<?import javafx.scene.image.ImageView?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.ColumnConstraints?>  
<?import javafx.scene.layout.GridPane?>  
<?import javafx.scene.layout.HBox?>  
<?import javafx.scene.layout.RowConstraints?>  
  
<AnchorPane maxHeight="200.0" maxWidth="350.0" minHeight="200.0" minWidth="350.0" prefHeight="200.0" prefWidth="350.0" xmlns="http://javafx.com/javafx/22" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.main.autoschoolapp.controllers.LoginController">  
 <children>  
 <GridPane layoutX="25.0" layoutY="34.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">  
 <columnConstraints>  
 <ColumnConstraints hgrow="SOMETIMES" maxWidth="100.0" minWidth="100.0" prefWidth="100.0" />  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints maxHeight="40.0" minHeight="40.0" prefHeight="40.0" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="30.0" minHeight="30.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="30.0" minHeight="30.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 <RowConstraints fx:id="ThirdRow" maxHeight="-Infinity" minHeight="0.0" prefHeight="50.0" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="1.7976931348623157E308" minHeight="50.0" prefHeight="50.0" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <children>  
 <HBox alignment="CENTER" fillHeight="false" nodeOrientation="LEFT\_TO\_RIGHT" prefHeight="100.0" prefWidth="200.0" spacing="10.0" GridPane.columnSpan="2" GridPane.hgrow="ALWAYS" GridPane.rowIndex="4" GridPane.vgrow="SOMETIMES">  
 <children>  
 <Button fx:id="BtnOk" alignment="CENTER" defaultButton="true" maxHeight="-Infinity" maxWidth="-Infinity" mnemonicParsing="false" onAction="#BtnOkActon" prefHeight="34.0" prefWidth="100.0" text="OK" textAlignment="CENTER" HBox.hgrow="ALWAYS" />  
 <Button fx:id="BtnCancel" alignment="CENTER" cancelButton="true" maxHeight="-Infinity" maxWidth="-Infinity" mnemonicParsing="false" onAction="#BtnCancelAction" prefHeight="34.0" prefWidth="100.0" text="Cancel" textAlignment="CENTER" HBox.hgrow="ALWAYS" />  
 </children>  
 <opaqueInsets>  
 <Insets bottom="5.0" left="5.0" right="5.0" top="5.0" />  
 </opaqueInsets>  
 <padding>  
 <Insets bottom="5.0" left="5.0" right="5.0" top="5.0" />  
 </padding>  
 <GridPane.margin>  
 <Insets />  
 </GridPane.margin>  
 </HBox>  
 <Label alignment="CENTER" prefHeight="40.0" prefWidth="374.0" styleClass="header-label" text="ООО АВТОШКОЛА" textAlignment="CENTER" GridPane.columnSpan="2" GridPane.vgrow="SOMETIMES" />  
 <PasswordField fx:id="PasswordField" promptText="Пароль" GridPane.columnSpan="2" GridPane.rowIndex="2" GridPane.vgrow="ALWAYS">  
 <GridPane.margin>  
 <Insets left="30.0" right="30.0" />  
 </GridPane.margin></PasswordField>  
 <TextField fx:id="TextFieldUsername" promptText="Логин" GridPane.columnSpan="2" GridPane.rowIndex="1" GridPane.vgrow="ALWAYS">  
 <GridPane.margin>  
 <Insets left="30.0" right="30.0" />  
 </GridPane.margin></TextField>  
 <ImageView fitHeight="33.0" fitWidth="85.0" pickOnBounds="true" preserveRatio="true" GridPane.halignment="CENTER" GridPane.valignment="CENTER">  
 <image>  
 <Image url="@car.png" />  
 </image>  
 </ImageView>  
 <ImageView fx:id="ImageViewCaptcha" fitHeight="40.0" fitWidth="150.0" pickOnBounds="true" preserveRatio="true" GridPane.columnSpan="2" GridPane.rowIndex="3">  
 <GridPane.margin>  
 <Insets bottom="5.0" left="60.0" top="5.0" />  
 </GridPane.margin></ImageView>  
 <TextField fx:id="TextFieldCaptcha" maxHeight="-Infinity" maxWidth="-Infinity" prefHeight="26.0" prefWidth="125.0" promptText="Введите капчу" GridPane.columnIndex="1" GridPane.rowIndex="3">  
 <GridPane.margin>  
 <Insets left="130.0" right="5.0" />  
 </GridPane.margin>  
 </TextField>  
 <Button fx:id="BtnRenewCaptcha" alignment="CENTER" contentDisplay="CENTER" mnemonicParsing="false" onAction="#BtnRenewCaptchaAction" prefHeight="41.0" prefWidth="21.0" text="O" textAlignment="CENTER" GridPane.rowIndex="3">  
 <GridPane.margin>  
 <Insets bottom="5.0" left="5.0" top="5.0" />  
 </GridPane.margin></Button>  
 </children>  
 </GridPane>  
 </children>  
</AnchorPane>  
  
**Файл интерфейса страницы категории групп**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.Menu?>

<?import javafx.scene.control.MenuBar?>

<?import javafx.scene.control.MenuItem?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.FlowPane?>

<?import javafx.scene.layout.HBox?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/22" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.main.autoschoolapp.controllers.GroupCategoryTableViewController">

<children>

<BorderPane prefHeight="200.0" prefWidth="200.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">

<center>

<BorderPane BorderPane.alignment="CENTER">

<top>

<FlowPane nodeOrientation="LEFT\_TO\_RIGHT" rowValignment="TOP" BorderPane.alignment="CENTER">

<children>

<TextField fx:id="TextFieldSearch" onAction="#TextFieldSearchAction" prefHeight="25.0" prefWidth="262.0" promptText="Введите название для поиска" />

</children>

</FlowPane>

</top>

<center>

<TableView fx:id="TableViewCategories" prefHeight="200.0" prefWidth="200.0" tableMenuButtonVisible="true" BorderPane.alignment="CENTER">

<columns>

<TableColumn id="TableColumnCountInStock" fx:id="TableColumnId" prefWidth="82.0" text="Id" />

<TableColumn id="TableColumnTitle" fx:id="TableColumnTitle" editable="false" maxWidth="1.7976931348623157E308" prefWidth="300.0" text="Наименование" />

</columns>

</TableView>

</center>

</BorderPane>

</center>

<bottom>

<Label fx:id="LabelInfo" text="Label" BorderPane.alignment="CENTER\_LEFT" />

</bottom>

<top>

<HBox BorderPane.alignment="CENTER">

<children>

<MenuBar prefHeight="25.0" prefWidth="381.0" HBox.hgrow="ALWAYS">

<menus>

<Menu mnemonicParsing="false" text="Файл">

<items>

<MenuItem fx:id="MenuItemBack" mnemonicParsing="false" onAction="#MenuItemBackAction" text="Назад" />

</items>

</Menu>

<Menu mnemonicParsing="false" text="Правка">

<items>

<MenuItem fx:id="MenuItemAdd" mnemonicParsing="false" onAction="#MenuItemAddAction" text="Добавить" />

<MenuItem fx:id="MenuItemUpdate" mnemonicParsing="false" onAction="#MenuItemUpdateAction" text="Изменить" />

<MenuItem fx:id="MenuItemDelete" mnemonicParsing="false" onAction="#MenuItemDeleteAction" text="Удалить" />

</items>

</Menu>

</menus>

</MenuBar>

<Label fx:id="LabelUser" alignment="CENTER\_RIGHT" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" prefWidth="120.0" styleClass="user-label" text="Label" HBox.hgrow="ALWAYS">

<padding>

<Insets right="20.0" />

</padding>

</Label>

</children>

</HBox>

</top>

</BorderPane>

</children>

</AnchorPane>

**Файл интерфейса страницы редактирования категорий групп**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.\*?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="120.0"

prefWidth="550.0" xmlns="http://javafx.com/javafx/17.0.2-ea" xmlns:fx="http://javafx.com/fxml/1"

fx:controller="org.main.autoschoolapp.controllers.GroupCategoryEditViewController">

<children>

<BorderPane layoutX="170.0" layoutY="47.0" prefHeight="200.0" prefWidth="200.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">

<bottom>

<ButtonBar prefHeight="40.0" prefWidth="200.0" BorderPane.alignment="CENTER">

<buttons>

<Button fx:id="BtnSave" defaultButton="true" mnemonicParsing="false" onAction="#BtnSaveAction" text="Сохранить" />

<Button fx:id="BtnCancel" cancelButton="true" mnemonicParsing="false" onAction="#BtnCancelAction" text="Отмена" />

</buttons>

<padding>

<Insets right="20.0" />

</padding>

</ButtonBar>

</bottom>

<center>

<VBox BorderPane.alignment="CENTER">

<Label maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" text="Название" GridPane.rowIndex="0" />

<TextField fx:id="TextFieldTitle" GridPane.columnIndex="1" GridPane.rowIndex="0" />

<BorderPane.margin>

<Insets left="5.0" right="5.0" top="20.0" />

</BorderPane.margin>

</VBox>

</center>

</BorderPane>

</children>

</AnchorPane>

**Файл module-info**

module org.main.autoschool {  
 requires javafx.controls;  
 requires javafx.fxml;  
 requires jakarta.persistence;  
 requires org.hibernate.orm.core;  
 requires java.naming;  
 requires java.desktop;  
 requires javafx.swing;  
 requires org.hibernate.validator;  
 requires org.postgresql.jdbc;  
 requires jakarta.validation;  
 requires itextpdf;  
  
 opens org.main.autoschoolapp to javafx.fxml;  
 opens org.main.autoschoolapp.model to org.hibernate.orm.core;  
 exports org.main.autoschoolapp;  
 exports org.main.autoschoolapp.controllers;  
 exports org.main.autoschoolapp.model;  
 opens org.main.autoschoolapp.controllers to javafx.fxml;  
 opens org.main.autoschoolapp.util to org.hibernate.orm.core;  
  
}

**Файл AutoschoolApp**

package org.main.autoschoolapp;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.scene.image.Image;  
import javafx.stage.Stage;  
import org.main.autoschoolapp.model.User;  
import org.main.autoschoolapp.util.Manager;  
  
import java.io.IOException;  
  
  
public class AutoschoolApp extends Application {  
  
 public User currentUser;  
  
 public static void main(String[] args) {  
 launch();  
  
  
 }  
  
 @Override  
 public void start(Stage stage) throws IOException {  
  
 stage.getIcons().add(new Image(AutoschoolApp.class.getResourceAsStream("car.png")));  
 stage.setScene(getNewScene());  
 stage.setTitle("Авторизация!");  
 stage.setResizable(false);  
 stage.setOnCloseRequest(event -> {  
 Manager.ShowPopup();  
 });  
 stage.setOnShown(windowEvent -> {  
 try {  
 stage.setScene(getNewScene());  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 });  
  
 Manager.mainStage = stage;  
 stage.show();  
 }  
  
 Scene getNewScene() throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(AutoschoolApp.class.getResource("login-view.fxml"));  
 Scene scene = new Scene(fxmlLoader.load());  
 scene.getStylesheets().add("base-styles.css");  
 return scene;  
 }  
}

**Класс ClassRoomService**

package org.main.autoschoolapp.service;

import org.main.autoschoolapp.model.ClassRoom;

import org.main.autoschoolapp.repository.ClassRoomDao;

import java.util.List;

public class ClassRoomService {

private ClassRoomDao classRoomDao = new ClassRoomDao();

public ClassRoomService() {

}

public List<ClassRoom> findAll() {

return classRoomDao.findAll();

}

public ClassRoom findOne(final long id) {

return classRoomDao.findOne(id);

}

public void save(final ClassRoom entity) {

if (entity == null)

return;

classRoomDao.save(entity);

}

public void update(final ClassRoom entity) {

if (entity == null)

return;

classRoomDao.update(entity);

}

public void delete(final ClassRoom entity) {

if (entity == null)

return;

classRoomDao.delete(entity);

}

public void deleteById(final Long id) {

if (id == null)

return;

classRoomDao.deleteById(id);

}

}

**Класс GroupCategoryService**

package org.main.autoschoolapp.service;

import java.util.List;

import org.main.autoschoolapp.model.GroupCategory;

import org.main.autoschoolapp.repository.ClassRoomDao;

import org.main.autoschoolapp.repository.GroupCategoryDao;

public class GroupCategoryService {

private GroupCategoryDao groupCategoryDao = new GroupCategoryDao();

public GroupCategoryService() {

}

public List<GroupCategory> findAll() {

return groupCategoryDao.findAll();

}

public GroupCategory findOne(final long id) {

return groupCategoryDao.findOne(id);

}

public void save(final GroupCategory entity)

{

if (entity == null)

return;

groupCategoryDao.save(entity);

}

public void update(final GroupCategory entity)

{

if (entity == null)

return;

groupCategoryDao.update(entity);

}

public void delete(final GroupCategory entity)

{

if (entity == null)

return;

groupCategoryDao.delete(entity);

}

public void deleteById(final Long id)

{

if (id == null)

return;

groupCategoryDao.deleteById(id);

}

}

**Класс InstructorService**

package org.main.autoschoolapp.service;

import org.main.autoschoolapp.model.Instructor;

import org.main.autoschoolapp.repository.InstructorDao;

import java.util.List;

public class InstructorService {

private InstructorDao instructorDao = new InstructorDao();

public InstructorService() {

}

public List<Instructor> findAll() {

return instructorDao.findAll();

}

public Instructor findOne(final long id) {

return instructorDao.findOne(id);

}

public void save(final Instructor entity) {

if (entity == null)

return;

instructorDao.save(entity);

}

public void update(final Instructor entity) {

if (entity == null)

return;

instructorDao.update(entity);

}

public void delete(final Instructor entity) {

if (entity == null)

return;

instructorDao.delete(entity);

}

public void deleteById(final Long id) {

if (id == null)

return;

instructorDao.deleteById(id);

}

}

**Класс LessonAttendanceService**

package org.main.autoschoolapp.service;

import org.main.autoschoolapp.model.LessonAttendance;

import org.main.autoschoolapp.repository.LessonAttendanceDao;

import java.util.List;

public class LessonAttendanceService {

private LessonAttendanceDao lessonAttendanceDao = new LessonAttendanceDao();

public LessonAttendanceService() {

}

public List<LessonAttendance> findAll() {

return lessonAttendanceDao.findAll();

}

public LessonAttendance findOne(final long id) {

return lessonAttendanceDao.findOne(id);

}

public void save(final LessonAttendance entity) {

if (entity == null)

return;

lessonAttendanceDao.save(entity);

}

public void update(final LessonAttendance entity) {

if (entity == null)

return;

lessonAttendanceDao.update(entity);

}

public void delete(final LessonAttendance entity) {

if (entity == null)

return;

lessonAttendanceDao.delete(entity);

}

public void deleteById(final Long id) {

if (id == null)

return;

lessonAttendanceDao.deleteById(id);

}

}

**Класс LessonService**

package org.main.autoschoolapp.service;

import org.main.autoschoolapp.model.Lesson;

import org.main.autoschoolapp.repository.LessonDao;

import java.util.List;

public class LessonService {

private LessonDao lessonDao = new LessonDao();

public LessonService() {

}

public List<Lesson> findAll() {

return lessonDao.findAll();

}

public Lesson findOne(final long id) {

return lessonDao.findOne(id);

}

public void save(final Lesson entity) {

if (entity == null)

return;

lessonDao.save(entity);

}

public void update(final Lesson entity) {

if (entity == null)

return;

lessonDao.update(entity);

}

public void delete(final Lesson entity) {

if (entity == null)

return;

lessonDao.delete(entity);

}

public void deleteById(final Long id) {

if (id == null)

return;

lessonDao.deleteById(id);

}

}

**Класс LicenseTypeService**

package org.main.autoschoolapp.service;

import org.main.autoschoolapp.model.LicenseType;

import org.main.autoschoolapp.repository.LicenseTypeDao;

import java.util.List;

public class LicenseTypeService {

private LicenseTypeDao licenseTypeDao = new LicenseTypeDao();

public LicenseTypeService() {

}

public List<LicenseType> findAll() {

return licenseTypeDao.findAll();

}

public LicenseType findOne(final long id) {

return licenseTypeDao.findOne(id);

}

public void save(final LicenseType entity) {

if (entity == null)

return;

licenseTypeDao.save(entity);

}

public void update(final LicenseType entity) {

if (entity == null)

return;

licenseTypeDao.update(entity);

}

public void delete(final LicenseType entity) {

if (entity == null)

return;

licenseTypeDao.delete(entity);

}

public void deleteById(final Long id) {

if (id == null)

return;

licenseTypeDao.deleteById(id);

}

}

**Класс StatusService**

package org.main.autoschoolapp.service;

import org.main.autoschoolapp.model.Status;

import org.main.autoschoolapp.repository.StatusDao;

import java.util.List;

public class StatusService {

private StatusDao statusDao = new StatusDao();

public StatusService() {

}

public List<Status> findAll() {

return statusDao.findAll();

}

public Status findOne(final long id) {

return statusDao.findOne(id);

}

public void save(final Status entity) {

if (entity == null)

return;

statusDao.save(entity);

}

public void update(final Status entity) {

if (entity == null)

return;

statusDao.update(entity);

}

public void delete(final Status entity) {

if (entity == null)

return;

statusDao.delete(entity);

}

public void deleteById(final Long id) {

if (id == null)

return;

statusDao.deleteById(id);

}

}

**Класс StudentService**

package org.main.autoschoolapp.service;

import org.main.autoschoolapp.model.Student;

import org.main.autoschoolapp.repository.StudentDao;

import java.util.List;

public class StudentService {

private StudentDao studentDao = new StudentDao();

public StudentService() {

}

public List<Student> findAll() {

return studentDao.findAll();

}

public Student findOne(final long id) {

return studentDao.findOne(id);

}

public void save(final Student entity)

{

if (entity == null)

return;

studentDao.save(entity);

}

public void update(final Student entity)

{

if (entity == null)

return;

studentDao.update(entity);

}

public void delete(final Student entity)

{

if (entity == null)

return;

studentDao.delete(entity);

}

public void deleteById(final Long id)

{

if (id == null)

return;

studentDao.deleteById(id);

}

}

**Класс UserService**

package org.main.autoschoolapp.service;

import org.main.autoschoolapp.model.User;

import org.main.autoschoolapp.repository.UserDao;

import java.util.List;

public class UserService {

private UserDao userDao = new UserDao();

public UserService() {

}

public List<User> findAll() {

return userDao.findAll();

}

public User findOne(final long id) {

return userDao.findOne(id);

}

public void save(final User entity) {

if (entity == null)

return;

userDao.save(entity);

}

public void update(final User entity) {

if (entity == null)

return;

userDao.update(entity);

}

public void delete(final User entity) {

if (entity == null)

return;

userDao.delete(entity);

}

public void deleteById(final Long id) {

if (id == null)

return;

userDao.deleteById(id);

}

}

**Класс VehicleService**

package org.main.autoschoolapp.service;

import org.main.autoschoolapp.model.Vehicle;

import org.main.autoschoolapp.repository.VehicleDao;

import java.util.List;

public class VehicleService {

private VehicleDao vehicleDao = new VehicleDao();

public VehicleService() {

}

public List<Vehicle> findAll() {

return vehicleDao.findAll();

}

public Vehicle findOne(final long id) {

return vehicleDao.findOne(id);

}

public void save(final Vehicle entity) {

if (entity == null)

return;

vehicleDao.save(entity);

}

public void update(final Vehicle entity) {

if (entity == null)

return;

vehicleDao.update(entity);

}

public void delete(final Vehicle entity) {

if (entity == null)

return;

vehicleDao.delete(entity);

}

public void deleteById(final Long id) {

if (id == null)

return;

vehicleDao.deleteById(id);

}

}

**Класс BaseDao**

package org.main.autoschoolapp.repository;

import org.hibernate.Session;

import org.hibernate.Transaction;

import org.main.autoschoolapp.util.HibernateSessionFactoryUtil;

import java.util.List;

public class BaseDao<T> {

private Class<T> clazz;

public BaseDao(Class<T> clazz) {

this.clazz = clazz;

}

protected Session getCurrentSession() {

return HibernateSessionFactoryUtil.getSessionFactory().openSession();

}

public void save(final T entity) {

Session session = getCurrentSession();

Transaction tx1 = session.beginTransaction();

session.save(entity);

tx1.commit();

session.close();

}

public void update(final T entity) {

Session session = getCurrentSession();

Transaction tx1 = session.beginTransaction();

// Используем merge вместо update для предотвращения ошибок сессий

session.merge(entity);

tx1.commit();

session.close();

}

public void delete(final T entity) {

Session session = getCurrentSession();

Transaction tx1 = session.beginTransaction();

session.delete(entity);

tx1.commit();

session.close();

}

public void deleteById(final long entityId) {

final T entity = findOne(entityId);

delete(entity);

}

public T findOne(final long id) {

Session session = getCurrentSession();

T entity = session.get(clazz, id);

session.close();

return entity;

}

public List<T> findAll() {

Session session = getCurrentSession();

List<T> items = session.createQuery("from " + clazz.getName(), clazz).list();

session.close();

return items;

}

}

**Класс ClassRoomDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.ClassRoom;

public class ClassRoomDao extends BaseDao<ClassRoom> {

public ClassRoomDao() {

super(ClassRoom.class);

}

}

**Класс GroupCategoryDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.GroupCategory;

import org.main.autoschoolapp.model.Instructor;

public class GroupCategoryDao extends BaseDao<GroupCategory> {

public GroupCategoryDao() {

super(GroupCategory.class);

}

}

**Класс InstructorDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.Instructor;

public class InstructorDao extends BaseDao<Instructor> {

public InstructorDao() {

super(Instructor.class);

}

}

**Класс LessonAttendanceDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.LessonAttendance;

public class LessonAttendanceDao extends BaseDao<LessonAttendance> {

public LessonAttendanceDao() {

super(LessonAttendance.class);

}

}

**Класс LessonDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.Lesson;

public class LessonDao extends BaseDao<Lesson>{

public LessonDao(){

super(Lesson.class);

}

}

**Класс LicenseTypeDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.LicenseType;

public class LicenseTypeDao extends BaseDao<LicenseType> {

public LicenseTypeDao() {

super(LicenseType.class);

}

}

**Класс LicenseTypeDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.LicenseType;

public class LicenseTypeDao extends BaseDao<LicenseType> {

public LicenseTypeDao() {

super(LicenseType.class);

}

}

**Класс StatusDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.Status;

public class StatusDao extends BaseDao<Status> {

public StatusDao(){

super(Status.class);

}

}

**Класс StudentDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.Student;

public class StudentDao extends BaseDao<Student> {

public StudentDao() {

super(Student.class);

}

}

**Класс UserDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.User;

public class UserDao extends BaseDao<User> {

public UserDao() {

super(User.class);

}

}

**Класс VehicleDao**

package org.main.autoschoolapp.repository;

import org.main.autoschoolapp.model.Vehicle;

public class VehicleDao extends BaseDao<Vehicle> {

public VehicleDao() {

super(Vehicle.class);

}

}

**Класс ClassRoom**

package org.main.autoschoolapp.model;

import jakarta.persistence.\*;

import java.util.HashSet;

import java.util.Objects;

import java.util.Set;

@Entity

@Table(name = "classroom", schema = "autoschool")

public class ClassRoom {

@Id

@Column(name = "classroom\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long classRoomId;

@Column(name = "address", nullable = false)

private String address;

public ClassRoom() {

}

public ClassRoom(Long classRoomId, String address) {

this.classRoomId = classRoomId;

this.address = address;

}

@OneToMany(fetch = FetchType.EAGER, mappedBy = "classRoom")

private Set<Lesson> lessons = new HashSet<>();

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof ClassRoom that)) return false;

return Objects.equals(classRoomId, that.classRoomId) && Objects.equals(address, that.address);

}

@Override

public int hashCode() {

final int hashCode = 17 \* classRoomId.hashCode() + 31 \* address.hashCode();

return hashCode;

}

@Override

public String toString() {

return address;

}

public Long getClassRoomId() {

return classRoomId;

}

public void setClassRoomId(Long classRoomId) {

this.classRoomId = classRoomId;

}

public String getTitle() {

return address;

}

public void setTitle(String address) {

this.address = address;

}

public Set<Lesson> getLessons() {

return lessons;

}

public void setLessons(Set<Lesson> lessons) {

this.lessons = lessons;

}

}

**Класс GroupCategory**

package org.main.autoschoolapp.model;

import jakarta.persistence.\*;

import java.util.HashSet;

import java.util.Objects;

import java.util.Set;

@Entity

@Table(name = "group\_categories", schema = "autoschool")

public class GroupCategory {

@Id

@Column(name = "category\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long categoryId;

@Column(name = "title", nullable = false, length = 200)

private String title;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "category")

private Set<Student> students = new HashSet<Student>();

public GroupCategory() {

}

public GroupCategory(Long categoryId, String title) {

this.categoryId = categoryId;

this.title = title;

}

public Set<Student> getStudents() {

return students;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof GroupCategory groupCategory)) return false;

return Objects.equals(categoryId, groupCategory.categoryId) && Objects.equals(title, groupCategory.title);

}

@Override

public int hashCode() {

final int hashCode = 17 \* categoryId.hashCode() + 31 \* title.hashCode();

return hashCode;

}

public Long getGroupCategoryId() {

return categoryId;

}

public void setGroupCategoryId(Long categoryId) {

this.categoryId = categoryId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

@Override

public String toString() {

return title;

}

}

**Класс Instructor**

package org.main.autoschoolapp.model;

import jakarta.persistence.\*;

import java.util.HashSet;

import java.util.Objects;

import java.util.Set;

@Entity

@Table(name = "instructors", schema = "autoschool")

public class Instructor {

@Id

@Column(name = "instructor\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long instructorId;

@Column(name = "name", nullable = false, length = 200)

private String name;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "instructor")

private Set<Student> students = new HashSet<Student>();

public Instructor() {

}

public Instructor(Long instructorId, String title) {

this.instructorId = instructorId;

this.name = title;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Instructor that)) return false;

return Objects.equals(instructorId, that.instructorId) && Objects.equals(name, that.name);

}

@Override

public int hashCode() {

final int hashCode = 17 \* instructorId.hashCode() + 31 \* name.hashCode();

return hashCode;

}

public Long getInstructorId() {

return instructorId;

}

public void setInstructorId(Long instructorId) {

this.instructorId = instructorId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return name;

}

public Set<Student> getStudents() {

return students;

}

public void setStudents(Set<Student> students) {

this.students = students;

}

}

**Класс Lesson**

package org.main.autoschoolapp.model;

import jakarta.persistence.\*;

import java.time.LocalDate;

import java.util.\*;

import static java.lang.Math.abs;

@Entity

@Table(name = "lessons", schema = "autoschool")

public class Lesson {

@Id

@Column(name = "lesson\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY) // Используйте нужную стратегию для генерации ID

private Long lessonId;

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Lesson order)) return false;

return Objects.equals(lessonId, order.lessonId) && Objects.equals(status, order.status) && Objects.equals(classRoom, order.classRoom) && Objects.equals(startDate, order.startDate) && Objects.equals(endDate, order.endDate) && Objects.equals(user, order.user) && Objects.equals(verificationCode, order.verificationCode) && Objects.equals(lessonAttendances, order.lessonAttendances);

}

@Override

public int hashCode() {

return Objects.hash(lessonId, status,

classRoom, startDate, endDate,

user, verificationCode, lessonAttendances);

}

@ManyToOne

@JoinColumn(name = "status\_id", nullable = false)

private Status status;

@ManyToOne

@JoinColumn(name = "classroom\_id", nullable = false)

private ClassRoom classRoom;

@Column(name = "start\_date", nullable = false)

private LocalDate startDate;

@Column(name = "end\_date", nullable = false)

private LocalDate endDate;

@ManyToOne

@JoinColumn(name = "username", nullable = true)

private User user;

@Column(name = "verification\_code", nullable = false)

private Integer verificationCode;

@OneToMany(mappedBy = "lesson", cascade = {CascadeType.PERSIST, CascadeType.REMOVE}, fetch = FetchType.EAGER)

private List<LessonAttendance> lessonAttendances = new ArrayList<>();

public Lesson() {

}

public List<LessonAttendance> getLessonAttendances() {

return lessonAttendances;

}

public void setLessonAttendances(List<LessonAttendance> lessonAttendances) {

this.lessonAttendances = lessonAttendances;

}

public Long getLessonId() {

return lessonId;

}

public void setLessonId(Long lessonId) {

this.lessonId = lessonId;

}

public Status getStatus() {

return status;

}

public void setStatus(Status status) {

this.status = status;

}

public ClassRoom getClassRoom() {

return classRoom;

}

public void setClassRoom(ClassRoom classRoom) {

this.classRoom = classRoom;

}

public LocalDate getStartDate() {

return startDate;

}

public void setStartDate(LocalDate startDate) {

this.startDate = startDate;

}

public LocalDate getEndDate() {

return endDate;

}

public void setEndDate(LocalDate endDate) {

this.endDate = endDate;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public Integer getVerificationCode() {

return verificationCode;

}

public void setVerificationCode(Integer verificationCode) {

this.verificationCode = verificationCode;

}

}

**Класс LessonAttendance**

package org.main.autoschoolapp.model;  
  
import jakarta.persistence.\*;  
  
import java.util.Objects;  
  
@Entity  
@IdClass(LessonAttendanceId.class)  
@Table(name = "lesson\_attendance", schema = "autoschool")  
public class LessonAttendance {  
  
 @Id  
 @ManyToOne  
 @JoinColumn(name = "lesson\_id", nullable = false)  
 private Lesson lesson;  
  
 @Id  
 @ManyToOne  
 @JoinColumn(name = "student\_id", nullable = false)  
 private Student student;  
  
 // Removed the direct 'age' column mapping, it will be accessed via the 'Student' entity  
  
 public LessonAttendance(Lesson lesson, Student student) {  
 this.lesson = lesson;  
 this.student = student;  
 }  
  
 public LessonAttendance(Student student) {  
 this.student = student;  
 }  
  
 public LessonAttendance() {  
 // Default constructor  
 }  
  
 public Lesson getLesson() {  
 return lesson;  
 }  
  
 public void setLesson(Lesson lesson) {  
 this.lesson = lesson;  
 }  
  
 public Student getStudent() {  
 return student;  
 }  
  
 public void setStudent(Student student) {  
 this.student = student;  
 }  
  
 public Integer getAge() {  
 return student != null ? student.getAge() : null; // Access 'age' from 'Student' entity  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof LessonAttendance that)) return false;  
 return Objects.equals(lesson, that.lesson) && Objects.equals(student, that.student);  
 }  
  
 @Override  
 public int hashCode() {  
 return 17 \* lesson.getLessonId().hashCode() +  
 31 \* student.getStudentId().hashCode();  
 }  
}

**Класс LessonAttendanceId**

package org.main.autoschoolapp.model;

import java.io.Serializable;

import java.util.Objects;

public class LessonAttendanceId implements Serializable {

private Lesson lesson;

private Student student;

public LessonAttendanceId() {

}

public LessonAttendanceId(Lesson lesson, Student student) {

this.lesson = lesson;

this.student = student;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof LessonAttendanceId that)) return false;

return Objects.equals(lesson, that.lesson) && Objects.equals(student, that.student);

}

@Override

public int hashCode() {

return Objects.hash(lesson, student);

}

// equals() and hashCode()

}

**Класс LicenseType**

package org.main.autoschoolapp.model;

import jakarta.persistence.\*;

import java.util.HashSet;

import java.util.Objects;

import java.util.Set;

@Entity

@Table(name = "license\_types", schema = "autoschool")

public class LicenseType {

@Id

@Column(name = "license\_type\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long licenseTypeId;

@Column(name = "title", nullable = false, length = 200)

private String title;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "licenseType")

private Set<Student> students = new HashSet<Student>();

public LicenseType() {

}

public LicenseType(Long licenseTypeId, String title) {

this.licenseTypeId = licenseTypeId;

this.title = title;

}

public Long getLicenseTypeId() {

return licenseTypeId;

}

public void setLicenseTypeId(Long licenseTypeId) {

this.licenseTypeId = licenseTypeId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

@Override

public String toString() {

return title;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof LicenseType licenseType)) return false;

return Objects.equals(licenseTypeId, licenseType.licenseTypeId) && Objects.equals(title, licenseType.title);

}

@Override

public int hashCode() {

final int hashCode = 17 \* licenseTypeId.hashCode() + 31 \* title.hashCode();

return hashCode;

}

public Set<Student> getStudents() {

return students;

}

public void setStudents(Set<Student> students) {

this.students = students;

}

}

**Класс Role**

package org.main.autoschoolapp.model;

import jakarta.persistence.\*;

import java.util.HashSet;

import java.util.Objects;

import java.util.Set;

@Entity

@Table(name = "role", schema = "autoschool")

public class Role {

@Id

@Column(name = "role\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long roleId;

@Column(name = "title", nullable = false, length = 200)

private String title;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "role")

private Set<User> users = new HashSet<User>();

public Role() {

}

public Role(Long categoryId, String title) {

this.roleId = categoryId;

this.title = title;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Role role)) return false;

return Objects.equals(roleId, role.roleId) && Objects.equals(title, role.title);

}

@Override

public int hashCode() {

final int hashCode = 17 \* roleId.hashCode() + 31 \* title.hashCode();

return hashCode;

}

public Long getRoleId() {

return roleId;

}

public void setRoleId(Long categoryId) {

this.roleId = categoryId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Set<User> getUsers() {

return users;

}

public void setUsers(Set<User> users) {

this.users = users;

}

@Override

public String toString() {

return title ;

}

}

**Класс Status**

package org.main.autoschoolapp.model;

import jakarta.persistence.\*;

import java.util.HashSet;

import java.util.Objects;

import java.util.Set;

@Entity

@Table(name = "statuses", schema = "autoschool")

public class Status {

@Id

@Column(name = "status\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long statusId;

@Column(name = "title", nullable = false, length = 200)

private String title;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "status")

private Set<Lesson> lessons = new HashSet<Lesson>();

public Status() {

}

public Status(Long statusId, String title) {

this.statusId = statusId;

this.title = title;

}

@Override

public String toString() {

return title;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Status status)) return false;

return Objects.equals(statusId, status.statusId) && Objects.equals(title, status.title);

}

@Override

public int hashCode() {

final int hashCode = 17 \* statusId.hashCode() + 31 \* title.hashCode();

return hashCode;

}

public Long getStatusId() {

return statusId;

}

public void setStatusId(Long statusId) {

this.statusId = statusId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Set<Lesson> getLessons() {

return lessons;

}

public void setLessons(Set<Lesson> lessons) {

this.lessons = lessons;

}

}

**Класс Student**

package org.main.autoschoolapp.model;

import jakarta.persistence.\*;

import javafx.beans.property.SimpleStringProperty;

import javafx.beans.property.StringProperty;

import javafx.embed.swing.SwingFXUtils;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import org.main.autoschoolapp.AutoschoolApp;

import javax.imageio.ImageIO;

import java.awt.image.BufferedImage;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.util.\*;

@Entity

@Table(name = "students", schema = "autoschool")

public class Student {

@Id

@Column(name = "student\_id", nullable = false, length = 100)

private String studentId;

@Column(name = "name", nullable = false, length = 100)

private String name;

@Column(name = "notes")

private String notes;

@Column(name = "max\_payment\_amount", nullable = false)

private Double maxPaymentAmount;

@Column(name = "age")

private Integer age;

@Column(name = "paid\_amount")

private Integer paidAmount;

@Column(name = "attendance\_rate", nullable = false)

private Integer attendanceRate;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "student")

private List<LessonAttendance> lessonAttendances = new ArrayList<>();

@ManyToOne

@JoinColumn(name = "vehicle\_id", nullable = false)

private Vehicle vehicle;

@ManyToOne

@JoinColumn(name = "instructor\_id", nullable = false)

private Instructor instructor;

@ManyToOne

@JoinColumn(name = "license\_type\_id", nullable = false)

private LicenseType licenseType;

@ManyToOne

@JoinColumn(name = "group\_category\_id", nullable = false)

private GroupCategory category;

@Column(name = "photo")

private byte[] photo;

public Student() {

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Student product)) return false;

return Objects.equals(studentId, product.studentId)

&& Objects.equals(name, product.name)

&& Objects.equals(notes, product.notes)

&& Objects.equals(age, product.age)

&& Objects.equals(maxPaymentAmount, product.maxPaymentAmount) && Objects.equals(paidAmount, product.paidAmount) && Objects.equals(attendanceRate, product.attendanceRate) && Objects.equals(vehicle, product.vehicle) && Objects.equals(instructor, product.instructor) && Objects.equals(licenseType, product.licenseType) && Objects.equals(category, product.category);

}

@Override

public int hashCode() {

return Objects.hash(studentId,

name, notes, age,

maxPaymentAmount, paidAmount, attendanceRate, vehicle, instructor, licenseType, category);

}

public StringProperty getPropertyTitle() {

return new SimpleStringProperty(this.name);

}

public List<LessonAttendance> getLessonAttendance() {

return lessonAttendances;

}

public String getStudentId() {

return studentId;

}

public void setStudentId(String studentId) {

this.studentId = studentId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getNotes() {

return notes;

}

public void setNotes(String notes) {

this.notes = notes;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

public Double getMaxPaymentAmount() {

return maxPaymentAmount;

}

public void setMaxPaymentAmount(Double maxPaymentAmount) {

this.maxPaymentAmount = maxPaymentAmount;

}

public Integer getPaidAmount() {

return paidAmount;

}

public void setPaidAmount(Integer paidAmount) {

this.paidAmount = paidAmount;

}

public Integer getAttendanceRate() {

return attendanceRate;

}

public void setAttendanceRate(Integer attendanceRate) {

this.attendanceRate = attendanceRate;

if (attendanceRate < 0)

this.attendanceRate = 0;

}

public Vehicle getVehicle() {

return vehicle;

}

public void setVehicle(Vehicle vehicle) {

this.vehicle = vehicle;

}

public Instructor getInstructor() {

return instructor;

}

public void setInstructor(Instructor instructor) {

this.instructor = instructor;

}

public LicenseType getLicenseType() {

return licenseType;

}

public void setLicenseType(LicenseType licenseType) {

this.licenseType = licenseType;

}

public GroupCategory getGroupCategory() {

return category;

}

public void setGroupCategory(GroupCategory category) {

this.category = category;

}

public boolean isHasPhoto() {

return photo != null;

}

public Image getPhoto() throws IOException {

if (photo == null)

return new Image(AutoschoolApp.class.getResourceAsStream("picture.png"));

BufferedImage capture = ImageIO.read(new ByteArrayInputStream(photo));

return SwingFXUtils.toFXImage(capture, null);

}

public void setPhoto(Image img) throws IOException {

BufferedImage buf = SwingFXUtils.fromFXImage(img, null);

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ImageIO.write(buf, "jpg", baos);

byte[] bytes = baos.toByteArray();

this.photo = bytes;

}

public ImageView getImage() throws IOException {

ImageView image = new ImageView();

image.setImage(getPhoto());

image.setFitHeight(60);

image.setPreserveRatio(true);

return image;

}

public Double getPriceWithDiscount() {

return age \* (1 - paidAmount / 100.0);

}

}

**Класс User**

package org.main.autoschoolapp.model;

// Java Program to Illustrate Creation of Simple POJO Class

// Importing required classes

import jakarta.persistence.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

@Entity

@Table(name = "users", schema = "autoschool")

public class User {

@Id

@Column(name = "username")

private String username;

@Column(name = "first\_name")

private String firstName;

@Column(name = "second\_name")

private String secondName;

@Column(name = "middle\_name")

private String middleName;

@Column(name = "password")

private String password;

@ManyToOne

@JoinColumn(name = "role\_id", nullable = false)

private Role role;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "user", cascade = CascadeType.PERSIST)

private List<Lesson> lessons = new ArrayList<>();

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getSecondName() {

return secondName;

}

public void setSecondName(String secondName) {

this.secondName = secondName;

}

public String getMiddleName() {

return middleName;

}

public void setMiddleName(String middleName) {

this.middleName = middleName;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public Role getRole() {

return role;

}

public void setRole(Role role) {

this.role = role;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof User user)) return false;

return Objects.equals(username, user.username) && Objects.equals(firstName, user.firstName) && Objects.equals(secondName, user.secondName) && Objects.equals(middleName, user.middleName) && Objects.equals(password, user.password);

}

@Override

public int hashCode() {

final int hashCode = 17 \* username.hashCode() + 31 \* firstName.hashCode() + 17 \* secondName.hashCode() + 31 \* middleName.hashCode() + 17 \* password.hashCode();

return hashCode;

}

public List<Lesson> getLessons() {

return lessons;

}

public void setLessons(List<Lesson> lessons) {

this.lessons = lessons;

}

}

**Класс Vehicle**

package org.main.autoschoolapp.model;

import jakarta.persistence.\*;

import java.util.HashSet;

import java.util.Objects;

import java.util.Set;

@Entity

@Table(name = "vehicles", schema = "autoschool")

public class Vehicle {

@Id

@Column(name = "vehicle\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long vehicleId;

@Column(name = "model", nullable = false, length = 200)

private String model;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "vehicle")

private Set<Student> students = new HashSet<Student>();

public Vehicle() {

}

public Vehicle(Long vehicleId, String title) {

this.vehicleId = vehicleId;

this.model = title;

}

public Long getVehicleId() {

return vehicleId;

}

public void setVehicleId(Long categoryId) {

this.vehicleId = vehicleId;

}

public String getModel() {

return model;

}

public void setModel(String model) {

this.model = model;

}

@Override

public String toString() {

return model;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Vehicle vehicle)) return false;

return Objects.equals(vehicleId, vehicle.vehicleId) && Objects.equals(model, vehicle.model);

}

@Override

public int hashCode() {

final int hashCode = 17 \* vehicleId.hashCode() + 31 \* model.hashCode();

return hashCode;

}

public Set<Student> getStudents() {

return students;

}

public void setStudents(Set<Student> students) {

this.students = students;

}

}

**Класс GroupCategoryEditViewController**

package org.main.autoschoolapp.controllers;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import javafx.stage.Stage;

import org.main.autoschoolapp.service.GroupCategoryService;

import org.main.autoschoolapp.service.VehicleService;

import org.main.autoschoolapp.util.Manager;

import java.io.IOException;

import static org.main.autoschoolapp.util.Manager.MessageBox;

public class GroupCategoryEditViewController {

private GroupCategoryService groupCategoryService = new GroupCategoryService();

@FXML

private Button BtnCancel;

@FXML

private Button BtnSave;

@FXML

private TextField TextFieldTitle;

@FXML

void BtnCancelAction(ActionEvent event) {

Stage stage = (Stage) BtnCancel.getScene().getWindow();

// do what you have to do

stage.close();

}

@FXML

void BtnSaveAction(ActionEvent event) throws IOException {

String error = checkFields().toString();

if (!error.isEmpty()) {

MessageBox("Ошибка", "Заполните поля", error, Alert.AlertType.ERROR);

return;

}

Manager.currentGroupCategory.setTitle(TextFieldTitle.getText());

if (Manager.currentGroupCategory.getGroupCategoryId() == null) {

Manager.currentGroupCategory.setTitle(TextFieldTitle.getText());

groupCategoryService.save(Manager.currentGroupCategory);

MessageBox("Информация", "", "Данные сохранены успешно", Alert.AlertType.INFORMATION);

} else {

groupCategoryService.update(Manager.currentGroupCategory);

MessageBox("Информация", "", "Данные обновлены успешно", Alert.AlertType.INFORMATION);

}

}

StringBuilder checkFields() {

StringBuilder error = new StringBuilder();

if (TextFieldTitle.getText().isEmpty()) {

error.append("Укажите название категории\n");

}

return error;

}

}

**Класс GroupCategoryTableViewController**

package org.main.autoschoolapp.controllers;

import javafx.beans.property.SimpleStringProperty;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import org.main.autoschoolapp.AutoschoolApp;

import org.main.autoschoolapp.model.GroupCategory;

import org.main.autoschoolapp.service.GroupCategoryService;

import org.main.autoschoolapp.util.Manager;

import java.io.IOException;

import java.net.URL;

import java.util.List;

import java.util.Optional;

import java.util.ResourceBundle;

import java.util.stream.Collectors;

import static org.main.autoschoolapp.util.Manager.\*;

public class GroupCategoryTableViewController implements Initializable {

@FXML

private Label LabelInfo;

@FXML

private Label LabelUser;

@FXML

private MenuItem MenuItemAdd;

@FXML

private MenuItem MenuItemBack;

@FXML

private MenuItem MenuItemDelete;

@FXML

private MenuItem MenuItemUpdate;

@FXML

private TableColumn<GroupCategory, String> TableColumnId;

@FXML

private TableColumn<GroupCategory, String> TableColumnTitle;

@FXML

private TableView<GroupCategory> TableViewCategories;

@FXML

private TextField TextFieldSearch;

private int itemsCount;

private GroupCategoryService groupCategoryService = new GroupCategoryService();

void filterData() {

List<GroupCategory> categories = groupCategoryService.findAll();

itemsCount = categories.size();

String searchText = TextFieldSearch.getText();

if (!searchText.isEmpty()) {

categories = categories.stream().filter(product -> product.getTitle().toLowerCase().contains(searchText.toLowerCase())).collect(Collectors.toList());

}

TableViewCategories.getItems().clear();

for (GroupCategory category : categories) {

TableViewCategories.getItems().add(category);

}

int filteredItemsCount = categories.size();

LabelInfo.setText("Всего записей " + filteredItemsCount + " из " + itemsCount);

}

@FXML

void MenuItemAddAction(ActionEvent event) {

Manager.currentStudent = null;

//ShowEditProductWindow();

filterData();

}

public void initController() {

LabelUser.setText("Вы вошли как " + currentUser.getSecondName()+ " "+ currentUser.getFirstName());

setCellValueFactories();

filterData();

}

private void setCellValueFactories() {

TableColumnId.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getGroupCategoryId().toString()));

TableColumnTitle.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getTitle()));

}

@FXML

void MenuItemBackAction(ActionEvent event) {

FXMLLoader fxmlLoader = new FXMLLoader(AutoschoolApp.class.getResource("main-view.fxml"));

Scene scene = null;

try {

// Задаем фиксированные размеры окна

scene = new Scene(fxmlLoader.load(), 1280, 660);

scene.getStylesheets().add("base-styles.css");

// Убираем автоматическое разворачивание окна

Manager.secondStage.setMaximized(false);

Manager.secondStage.setResizable(false);

// Устанавливаем сцену

Manager.secondStage.setScene(scene);

} catch (IOException e) {

throw new RuntimeException(e);

}

}

@FXML

void MenuItemDeleteAction(ActionEvent event) {

GroupCategory category = TableViewCategories.getSelectionModel().getSelectedItem();

if (!category.getStudents().isEmpty()) {

ShowErrorMessageBox("Ошибка целостности данных, у данного курса есть студенты");

return;

}

Optional<ButtonType> result = ShowConfirmPopup();

if (result.get() == ButtonType.OK) {

groupCategoryService.delete(category);

filterData();

}

}

@FXML

void MenuItemUpdateAction(ActionEvent event) {

GroupCategory category = TableViewCategories.getSelectionModel().getSelectedItem();

// Manager.currentProduct = category;

// ShowEditCategoryWindow();

filterData();

}

@FXML

void TextFieldSearchAction(ActionEvent event) {

filterData();

}

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

initController();

}

}

**Класс ListCellController**

package org.main.autoschoolapp.controllers;

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import javafx.scene.image.ImageView;

import org.main.autoschoolapp.model.Student;

import java.io.IOException;

public class ListCellController {

// Ссылка на ImageView для фотографии студента

@FXML

private ImageView ImageViewPhoto;

// Ссылка на Label для отображения оценки посещаемости

@FXML

private Label LabelAttendanceRate;

// Ссылка на Label для отображения имени студента

@FXML

private Label LabelName;

// Ссылка на Label для отображения имени инструктора

@FXML

private Label LabelInstructor;

// Ссылка на Label для отображения цены

@FXML

private Label LabelMaxPaymentAmount;

// Метод для инициализации значений, которые будут отображаться на экране

public void initialize() {

}

public void setStudent(Student student) throws IOException {

ImageViewPhoto.setImage(student.getPhoto());

LabelAttendanceRate.setText("Рейтинг - "+ student.getAttendanceRate().toString());

LabelName.setText("Имя студента: " + student.getName());

LabelInstructor.setText("Инструктор: " + student.getInstructor().getName());

LabelMaxPaymentAmount.setText("Цена оплаты: " + String.valueOf(student.getMaxPaymentAmount()));

}

}

**Класс LoginController**

package org.main.autoschoolapp.controllers;  
  
import jakarta.persistence.Query;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.fxml.Initializable;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.PasswordField;  
import javafx.scene.control.TextField;  
import javafx.scene.image.ImageView;  
import javafx.scene.layout.RowConstraints;  
import javafx.stage.Stage;  
import org.hibernate.Session;  
import org.main.autoschoolapp.AutoschoolApp;  
import org.main.autoschoolapp.model.User;  
import org.main.autoschoolapp.util.HibernateSessionFactoryUtil;  
import org.main.autoschoolapp.util.MakeCaptcha;  
import org.main.autoschoolapp.util.Manager;  
import org.main.autoschoolapp.service.UserService;  
  
  
import java.io.IOException;  
import java.net.URL;  
import java.util.\*;  
  
import static org.main.autoschoolapp.util.Manager.ShowErrorMessageBox;  
  
public class LoginController implements Initializable {  
  
 boolean isWrongCaptha;  
 boolean isShowCaptha;  
 UserService userService = new UserService();  
 String captchaCode;  
 int secondsLeft;  
 @FXML  
 private Button BtnCancel;  
 @FXML  
 RowConstraints ThirdRow;  
 @FXML  
 private Button BtnOk;  
 @FXML  
 private PasswordField PasswordField;  
 @FXML  
 private TextField TextFieldUsername;  
 @FXML  
 private TextField TextFieldCaptcha;  
 @FXML  
 private ImageView ImageViewCaptcha;  
 @FXML Button BtnRenewCaptcha;  
  
  
 @Override  
 public void initialize(URL url, ResourceBundle resourceBundle) {  
 initController();  
 }  
  
 @FXML  
 void BtnRenewCaptchaAction(ActionEvent event) {  
 generateCaptcha();  
 }  
 @FXML  
 void BtnCancelAction(ActionEvent event) {  
 Manager.ShowPopup();  
 }  
  
 @FXML  
 void BtnOkActon(ActionEvent event) {  
  
 List<User> users = userService.findAll();  
 Optional<User> person = users.stream().filter(user -> user.getUsername().equals(TextFieldUsername.getText()) &&  
 user.getPassword().equals(PasswordField.getText())).findFirst();  
  
 if (person.isEmpty() && isShowCaptha && !TextFieldCaptcha.getText().equals(captchaCode)) {  
 System.out.println("Bad error");  
 ShowErrorMessageBox("Не верный логин, пароль или текст капчи");  
 blockButtons();  
 return;  
 }  
 if (person.isEmpty() && (!isShowCaptha)) {  
 System.out.println("Bad error");  
 generateCaptcha();  
 isShowCaptha = true;  
 ThirdRow.setPrefHeight(50);  
 ImageViewCaptcha.setVisible(true);  
 TextFieldCaptcha.setVisible(true);  
 BtnRenewCaptcha.setVisible(true);  
 ShowErrorMessageBox("Не верный логин или пароль");  
 return;  
 }  
 if (person.isPresent() && isShowCaptha && !TextFieldCaptcha.getText().equals(captchaCode)) {  
 blockButtons();  
 ShowErrorMessageBox("Не верный логин, пароль или текст капчи");  
 return;  
 }  
  
 if (person.isPresent() && isShowCaptha && TextFieldCaptcha.getText().equals(captchaCode)) {  
 showMainWindow(person.get());  
 return;  
 }  
  
 if (person.isPresent() && !isShowCaptha) {  
 showMainWindow(person.get());  
 }  
 }  
  
 public void generateCaptcha() {  
 try {  
 ImageViewCaptcha.setImage(MakeCaptcha.CreateImage(150,40, 4));  
 captchaCode = MakeCaptcha.captchaCode();  
 System.out.println(captchaCode);  
 } catch (IOException e) {  
 System.out.println(e.getMessage());  
 }  
 }  
  
  
 public void showMainWindow(User person) {  
 Manager.currentUser = person;  
 System.out.println(Manager.currentUser);  
 Manager.mainStage.hide();  
 Stage newWindow = new Stage();  
 FXMLLoader fxmlLoader = new FXMLLoader(AutoschoolApp.class.getResource("main-view.fxml"));  
  
 Scene scene = null;  
 try {  
 scene = new Scene(fxmlLoader.load());  
 scene.getStylesheets().add("base-styles.css");  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 newWindow.setTitle("Вы вошли как " + Manager.currentUser.getFirstName());  
 newWindow.setScene(scene);  
 newWindow.setOnCloseRequest(e -> {  
 Manager.mainStage.show();  
 });  
 Manager.secondStage = newWindow;  
 // Устанавливаем размер окна  
 newWindow.setWidth(1280); // Ширина 1280px  
 newWindow.setHeight(660); // Высота 1080px  
  
 newWindow.show();  
 }  
  
 public void initTimer() {  
 TimerTask task = new TimerTask() {  
 public void run() {  
 System.out.println("Task performed on: " + new Date() + "n" +  
 "Thread's name: " + Thread.currentThread().getName());  
 secondsLeft--;  
 if (secondsLeft == 0) ;  
 {  
 BtnOk.setDisable(false);  
 BtnCancel.setDisable(false);  
 this.cancel();  
 }  
 }  
 };  
 Timer timer = new Timer("Timer");  
  
 long delay = 10000L;  
 timer.schedule(task, delay);  
 }  
  
 public void blockButtons()  
 {  
 initTimer();  
 secondsLeft = 10;  
 BtnOk.setDisable(true);  
 BtnCancel.setDisable(true);  
 }  
 public void initController()  
 {  
 this.isWrongCaptha = false;  
 this.isShowCaptha = false;  
 ThirdRow.setPrefHeight(0);  
 TextFieldCaptcha.setVisible(false);  
 BtnRenewCaptcha.setVisible(false);  
 ImageViewCaptcha.setVisible(false);  
 isWrongCaptha = false;  
 isShowCaptha = false;  
 captchaCode = "";  
 secondsLeft = 0;  
 }  
  
  
}

**Класс MainWindowController**

package org.main.autoschoolapp.controllers;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.AnchorPane;

import org.main.autoschoolapp.model.GroupCategory;

import org.main.autoschoolapp.model.Student;

import org.main.autoschoolapp.service.GroupCategoryService;

import org.main.autoschoolapp.service.StudentService;

import org.main.autoschoolapp.util.Manager;

import java.io.IOException;

import java.net.URL;

import java.util.Comparator;

import java.util.List;

import java.util.ResourceBundle;

import java.util.stream.Collectors;

public class MainWindowController implements Initializable {

@FXML

private TextField TextFieldSearch;

@FXML

private ComboBox<GroupCategory> CmboBoxStudentType;

@FXML

private ComboBox<String> ComboboxSort;

@FXML

private ListView<Student> ListViewStudent;

@FXML

private Label LabelUser;

@FXML

private Label LabelInfo;

@FXML

ComboBox<String> ComboBoxDiscount;

private final GroupCategoryService groupCategoryService = new GroupCategoryService();

private final StudentService studentService = new StudentService();

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

LabelUser.setText(Manager.currentUser.getFirstName());

List<GroupCategory> categoryList = groupCategoryService.findAll();

categoryList.add(0, new GroupCategory(0L, "Все"));

ObservableList<GroupCategory> categories = FXCollections.observableArrayList(categoryList);

CmboBoxStudentType.setItems(categories);

ObservableList<String> discounts = FXCollections.observableArrayList("По рейтингу", "0-35", "35-65", "от 65");

ComboBoxDiscount.setItems(discounts);

ObservableList<String> orders = FXCollections.observableArrayList("по возрастанию цены", "по убыванию цены");

ComboboxSort.setItems(orders);

filterData();

}

public void TextFieldSearchAction(ActionEvent actionEvent) {

filterData();

}

public void CmboBoxStudentTypeAction(ActionEvent actionEvent) {filterData();

}

// Method for sort combo box action

public void ComboboxSortAction(ActionEvent actionEvent) {

filterData();

}

@FXML

void filterData() {

// Получаем всех студентов

List<Student> students = studentService.findAll();

int ageStudent = students.size();

// Фильтрация по типу студента

if (!CmboBoxStudentType.getSelectionModel().isEmpty()) {

GroupCategory category = CmboBoxStudentType.getValue();

if (category.getGroupCategoryId() != 0) {

students = students.stream()

.filter(student -> student.getGroupCategory().getGroupCategoryId().equals(category.getGroupCategoryId()))

.collect(Collectors.toList());

}

}

// Фильтрация по рейтингу посещаемости

if (!ComboBoxDiscount.getSelectionModel().isEmpty()) {

String attendanceRate = ComboBoxDiscount.getValue();

switch (attendanceRate) {

case "0-35":

students = students.stream()

.filter(student -> student.getAttendanceRate() >= 0 && student.getAttendanceRate() <= 35)

.collect(Collectors.toList());

break;

case "35-65":

students = students.stream()

.filter(student -> student.getAttendanceRate() > 35 && student.getAttendanceRate() <= 65)

.collect(Collectors.toList());

break;

case "от 65":

students = students.stream()

.filter(student -> student.getAttendanceRate() > 65)

.collect(Collectors.toList());

break;

case "По рейтингу":

// Если выбрано "По рейтингу", оставляем всех студентов без изменений

break;

}

}

// Сортировка

if (!ComboboxSort.getSelectionModel().isEmpty()) {

String order = ComboboxSort.getValue();

Comparator<Student> comparator = Comparator.comparing(Student::getPaidAmount);

if (order.equals("по убыванию цены")) {

comparator = comparator.reversed();

}

students = students.stream()

.sorted(comparator)

.collect(Collectors.toList());

}

// Поиск

String searchText = TextFieldSearch.getText();

if (!searchText.isEmpty()) {

students = students.stream()

.filter(student -> student.getName().toLowerCase().contains(searchText.toLowerCase()))

.collect(Collectors.toList());

}

// Обновление списка

ListViewStudent.getItems().clear();

ListViewStudent.getItems().addAll(students);

ListViewStudent.setCellFactory(lv -> new StudentCell());

// Обновление информации

LabelInfo.setText("Всего записей " + students.size() + " из " + ageStudent);

}

public void BtnStudentsAction() {

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/org/main/autoschoolapp/students-table-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

Manager.secondStage.setScene(scene);

// Если необходимо, можно вызвать метод для отображения главного окна: Manager.mainStage.show();

} catch (IOException e) {

throw new RuntimeException("Ошибка загрузки окна студентов: " + e.getMessage(), e);

}

}

public void BtnBackAction(ActionEvent actionEvent) {

}

public void loadStudents(GroupCategory groupCategory) {

ListViewStudent.getItems().clear();

List<Student> students = studentService.findAll();

if (groupCategory != null) {

students = students.stream()

.filter(student -> student.getGroupCategory().getGroupCategoryId().equals(groupCategory.getGroupCategoryId()))

.collect(Collectors.toList());

}

for (Student student : students) {

ListViewStudent.getItems().add(student); // добавляем объект Student, а не его строковое представление

}

// Указываем, что для отображения данных в ListView будет использоваться StudentCell

ListViewStudent.setCellFactory(lv -> new StudentCell());

}

public void BtnGoVehicleAction(ActionEvent actionEvent) {

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/org/main/autoschoolapp/vehicle-table-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

Manager.secondStage.setScene(scene);

// Если необходимо, можно вызвать метод для отображения главного окна: Manager.mainStage.show();

} catch (IOException e) {

throw new RuntimeException("Ошибка загрузки окна студентов: " + e.getMessage(), e);

}

}

public void BtnGroupCategoryAction(ActionEvent actionEvent) {

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/org/main/autoschoolapp/group-category-table-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

Manager.secondStage.setScene(scene);

// Если необходимо, можно вызвать метод для отображения главного окна: Manager.mainStage.show();

} catch (IOException e) {

throw new RuntimeException("Ошибка загрузки окна категорий: " + e.getMessage(), e);

}

}

public void BtnJournalAction(ActionEvent actionEvent) {

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/org/main/autoschoolapp/schedule-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

Manager.secondStage.setScene(scene);

// Если необходимо, можно вызвать метод для отображения главного окна: Manager.mainStage.show();

} catch (IOException e) {

throw new RuntimeException("Ошибка загрузки окна категорий: " + e.getMessage(), e);

}

}

}

**Класс ScheduleViewController**

package org.main.autoschoolapp.controllers;

import com.itextpdf.text.\*;

import com.itextpdf.text.pdf.BaseFont;

import com.itextpdf.text.pdf.PdfWriter;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.stage.FileChooser;

import javafx.stage.Modality;

import javafx.stage.Stage;

import org.main.autoschoolapp.AutoschoolApp;

import org.main.autoschoolapp.model.ClassRoom;

import org.main.autoschoolapp.model.Lesson;

import org.main.autoschoolapp.model.Status;

import org.main.autoschoolapp.service.ClassRoomService;

import org.main.autoschoolapp.service.LessonService;

import org.main.autoschoolapp.service.StatusService;

import org.main.autoschoolapp.util.Manager;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.Optional;

public class ScheduleViewController {

private static final String FONT = "src/main/resources/fonts/arial.ttf";

private final LessonService lessonService = new LessonService();

private final ClassRoomService classRoomService = new ClassRoomService();

private final StatusService statusService = new StatusService();

@FXML

private TableView<Lesson> lessonTable;

@FXML

private TableColumn<Lesson, Long> lessonIdColumn;

@FXML

private TableColumn<Lesson, String> startDateColumn;

@FXML

private TableColumn<Lesson, String> endDateColumn;

@FXML

private TableColumn<Lesson, String> classRoomColumn;

@FXML

private TableColumn<Lesson, String> statusColumn;

@FXML

public void initialize() {

lessonIdColumn.setCellValueFactory(new PropertyValueFactory<>("lessonId"));

startDateColumn.setCellValueFactory(new PropertyValueFactory<>("startDate"));

endDateColumn.setCellValueFactory(new PropertyValueFactory<>("endDate"));

classRoomColumn.setCellValueFactory(new PropertyValueFactory<>("classRoom"));

statusColumn.setCellValueFactory(new PropertyValueFactory<>("status"));

lessonTable.setItems(FXCollections.observableArrayList(lessonService.findAll()));

}

@FXML

public void handleAdd() {

Stage newWindow = new Stage();

FXMLLoader fxmlLoader = new FXMLLoader(AutoschoolApp.class.getResource("editLesson.fxml"));

try {

Scene scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

newWindow.setTitle("Изменить данные");

newWindow.initOwner(Manager.secondStage);

newWindow.initModality(Modality.WINDOW\_MODAL);

newWindow.setScene(scene);

Manager.currentStage = newWindow;

newWindow.showAndWait();

} catch (IOException e) {

throw new RuntimeException(e);

} finally {

Manager.currentStage = null;

}

}

@FXML

public void handleDelete() {

Lesson selectedLesson = lessonTable.getSelectionModel().getSelectedItem();

if (selectedLesson != null) {

Alert alert = new Alert(Alert.AlertType.CONFIRMATION, "Вы уверены, что хотите удалить урок?");

Optional<ButtonType> result = alert.showAndWait();

if (result.isPresent() && result.get() == ButtonType.OK) {

lessonService.delete(selectedLesson);

lessonTable.getItems().remove(selectedLesson);

}

} else {

showAlert("Ошибка", "Выберите урок для удаления");

}

}

@FXML

public void handlePrint() {

try {

printScheduleToPDF();

} catch (DocumentException | IOException e) {

showAlert("Ошибка", "Не удалось создать PDF файл: " + e.getMessage());

}

}

private void printScheduleToPDF() throws DocumentException, IOException {

FileChooser fileChooser = new FileChooser();

fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("PDF files (\*.PDF)", "\*.pdf"));

File file = fileChooser.showSaveDialog(new Stage());

if (file != null) {

Document document = new Document(PageSize.A4);

PdfWriter.getInstance(document, new FileOutputStream(file));

Font titleFont = FontFactory.getFont(FONT, "cp1251", BaseFont.EMBEDDED, 18);

Font contentFont = FontFactory.getFont(FONT, "cp1251", BaseFont.EMBEDDED, 12);

document.open();

// Заголовок

Paragraph title = new Paragraph("РАСПИСАНИЕ ЗАНЯТИЙ", titleFont);

title.setAlignment(Element.ALIGN\_CENTER);

title.setSpacingAfter(20);

document.add(title);

// Цикл для уроков

document.add(new Paragraph("Уроки:", contentFont));

for (Lesson lesson : lessonService.findAll()) {

String lessonDetails = String.format("Дата начала: %s, Дата окончания: %s, Кабинет: %s",

lesson.getStartDate().format(DateTimeFormatter.ofPattern("dd.MM.yyyy")),

lesson.getEndDate().format(DateTimeFormatter.ofPattern("dd.MM.yyyy")),

lesson.getClassRoom() != null ? lesson.getClassRoom().getTitle() : "N/A");

document.add(new Paragraph(lessonDetails, contentFont));

}

document.close();

}

}

private void showAlert(String title, String message) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

}

**Класс StudentCell**

package org.main.autoschoolapp.controllers;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.control.ListCell;

import org.main.autoschoolapp.AutoschoolApp;

import org.main.autoschoolapp.model.Student;

import java.io.IOException;

public class StudentCell extends ListCell<Student> {

private final Parent root ;

private ListCellController controller ;

public StudentCell() {

try {

FXMLLoader loader = new FXMLLoader(AutoschoolApp.class.getResource("studentcell-view.fxml"));

root = loader.load();

controller = loader.getController() ;

} catch (IOException exc) {

throw new RuntimeException(exc);

}

}

@Override

protected void updateItem(Student student, boolean empty) {

super.updateItem(student, empty);

if (empty || student==null) {

setGraphic(null);

} else {

try {

controller.setStudent(student);

} catch (IOException e) {

throw new RuntimeException(e);

}

setGraphic(root);

}

}

}

**Класс StudentEditViewController**

package org.main.autoschoolapp.controllers;

import javafx.collections.FXCollections;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.\*;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.input.MouseEvent;

import javafx.stage.FileChooser;

import javafx.stage.Stage;

import org.main.autoschoolapp.model.\*;

import org.main.autoschoolapp.service.\*;

import org.main.autoschoolapp.util.Manager;

import java.io.File;

import java.io.IOException;

import java.net.MalformedURLException;

import java.net.URL;

import java.util.ResourceBundle;

import static org.main.autoschoolapp.util.Manager.MessageBox;

public class StudentEditViewController implements Initializable {

public TextField TextFieldDiscountAmountMax;

public TextField TextFieldMaxPaymentAmount;

boolean imageLoaded = false;

@FXML

private Button BtnCancel;

@FXML

private Button BtnLoadImage;

@FXML

private Button BtnSave;

private final GroupCategoryService groupCategoryService = new GroupCategoryService();

private final InstructorService instructorService = new InstructorService();

private final LicenseTypeService licenseTypeService = new LicenseTypeService();

private final VehicleService vehicleService = new VehicleService();

private StudentService studentService = new StudentService();

@FXML

private ComboBox<GroupCategory> ComboBoxGroupCategory;

@FXML

private ComboBox<Instructor> ComboBoxInstructor;

@FXML

private ComboBox<LicenseType> ComboBoxLicenseType;

@FXML

private ComboBox<Vehicle> ComboBoxVehicle;

@FXML

private ImageView ImageViewPhoto;

@FXML

private TextArea TextAreaNote;

@FXML

private TextField TextFieldIDStudent;

@FXML

private TextField TextFieldAge;

@FXML

private TextField TextFieldAttendanceRate;

@FXML

private TextField TextFieldPaidAmount;

@FXML

private TextField TextFieldName;

public void BtnLoadImageAction(ActionEvent actionEvent) throws MalformedURLException {

FileChooser fileChooser = new FileChooser();

fileChooser.getExtensionFilters().addAll(

new FileChooser.ExtensionFilter("JPG", "\*.jpg")

);

Stage stage = (Stage) BtnLoadImage.getScene().getWindow();

File file = fileChooser.showOpenDialog(stage);

if (file != null) {

String imageUrl = file.toURI().toURL().toExternalForm();

ImageViewPhoto.setImage(new Image(imageUrl));

imageLoaded = true;

}

}

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

ComboBoxGroupCategory.setItems(FXCollections.observableArrayList(groupCategoryService.findAll()));

ComboBoxLicenseType.setItems(FXCollections.observableArrayList(licenseTypeService.findAll()));

ComboBoxInstructor.setItems(FXCollections.observableArrayList(instructorService.findAll()));

ComboBoxVehicle.setItems(FXCollections.observableArrayList(vehicleService.findAll()));

if (Manager.currentStudent != null) {

TextFieldIDStudent.setEditable(false);

TextFieldIDStudent.setText(Manager.currentStudent.getStudentId());

TextFieldName.setText(Manager.currentStudent.getName());

TextAreaNote.setText(Manager.currentStudent.getNotes());

TextFieldMaxPaymentAmount.setText(String.format("%.2f", (double) Manager.currentStudent.getAge()));

// Здесь выводим посещаемость в TextField

TextFieldAge.setText(Manager.currentStudent.getAge().toString());

TextFieldAttendanceRate.setText(Manager.currentStudent.getAttendanceRate().toString());

TextFieldPaidAmount.setText(Manager.currentStudent.getPaidAmount().toString());

TextFieldMaxPaymentAmount.setText(Manager.currentStudent.getMaxPaymentAmount().toString());

TextFieldDiscountAmountMax.setText(Manager.currentStudent.getPaidAmount().toString());

try {

ImageViewPhoto.setImage(Manager.currentStudent.getPhoto());

} catch (IOException e) {

throw new RuntimeException(e);

}

ComboBoxGroupCategory.setValue(Manager.currentStudent.getGroupCategory());

ComboBoxLicenseType.setValue(Manager.currentStudent.getLicenseType());

ComboBoxInstructor.setValue(Manager.currentStudent.getInstructor());

ComboBoxVehicle.setValue(Manager.currentStudent.getVehicle());

} else {

Manager.currentStudent = new Student();

}

}

@FXML

void BtnSaveAction(ActionEvent event) throws IOException {

// Проверка обязательных полей

StringBuilder validationResult = checkFields();

if (validationResult.length() > 0) {

MessageBox("Ошибка", "Пожалуйста, исправьте следующие ошибки:\n" + validationResult.toString(), "Ошибка", Alert.AlertType.ERROR);

return;

}

// Устанавливаем значения для студента

Manager.currentStudent.setName(TextFieldName.getText());

Manager.currentStudent.setGroupCategory(ComboBoxGroupCategory.getValue());

Manager.currentStudent.setLicenseType(ComboBoxLicenseType.getValue());

Manager.currentStudent.setVehicle(ComboBoxVehicle.getValue());

Manager.currentStudent.setInstructor(ComboBoxInstructor.getValue());

Manager.currentStudent.setNotes(TextAreaNote.getText());

// Преобразуем и устанавливаем возраст

try {

int age = Integer.parseInt(TextFieldAge.getText());

Manager.currentStudent.setAge(age);

} catch (NumberFormatException e) {

MessageBox("Ошибка", "Неверно указан возраст", "Ошибка", Alert.AlertType.ERROR);

return;

}

// Преобразуем и устанавливаем числовые значения для PaidAmount и MaxPaymentAmount

try {

int paidAmount = Integer.parseInt(TextFieldDiscountAmountMax.getText());

Manager.currentStudent.setPaidAmount(paidAmount);

} catch (NumberFormatException e) {

MessageBox("Ошибка", "Неверно указана сумма оплаты", "Ошибка", Alert.AlertType.ERROR);

return;

}

try {

double maxPaymentAmount = Double.parseDouble(TextFieldMaxPaymentAmount.getText());

Manager.currentStudent.setMaxPaymentAmount(maxPaymentAmount); // Corrected method

} catch (NumberFormatException e) {

MessageBox("Ошибка", "Неверно указана максимальная сумма оплаты", "Ошибка", Alert.AlertType.ERROR);

return;

}

try {

int attendanceRate = Integer.parseInt(TextFieldAttendanceRate.getText());

Manager.currentStudent.setAttendanceRate(attendanceRate);

} catch (NumberFormatException e) {

MessageBox("Ошибка", "Неверно указана посещаемость", "Ошибка", Alert.AlertType.ERROR);

return;

}

// Устанавливаем изображение, если оно было загружено

if (imageLoaded) {

Manager.currentStudent.setPhoto(ImageViewPhoto.getImage());

}

// Сохраняем или обновляем данные студента

if (Manager.currentStudent.getStudentId() == null || Manager.currentStudent.getStudentId().isEmpty()) {

Manager.currentStudent.setStudentId(TextFieldIDStudent.getText());

studentService.save(Manager.currentStudent);

MessageBox("Информация", "", "Данные сохранены успешно", Alert.AlertType.INFORMATION);

} else {

studentService.update(Manager.currentStudent);

MessageBox("Информация", "", "Данные обновлены успешно", Alert.AlertType.INFORMATION);

}

}

StringBuilder checkFields() {

StringBuilder error = new StringBuilder();

if (TextFieldIDStudent.getText().isEmpty()) {

error.append("Укажите ID cтудента\n");

}

if (TextFieldName.getText().isEmpty()) {

error.append("Укажите имя студента\n");

}

if (TextFieldAge.getText().isEmpty()) {

error.append("Укажите возраст\n");

}

if (ComboBoxGroupCategory.getValue() == null) {

error.append("Выберите категорию\n");

}

if (ComboBoxInstructor.getValue() == null) {

error.append("Выберите инструктора\n");

}

if (ComboBoxLicenseType.getValue() == null) {

error.append("Выберите лицензию\n");

}

if (ComboBoxVehicle.getValue() == null) {

error.append("Выберите автомобиль\n");

}

return error;

}

public void BtnCancelAction(ActionEvent actionEvent) {

Stage stage = (Stage) BtnCancel.getScene().getWindow();

stage.close();

}

public void TextFieldName(ActionEvent actionEvent) {

}

public void TextAreaDescription(MouseEvent mouseEvent) {

}

public void TextFieldMaxPayment(ActionEvent actionEvent) {

}

public void ComboBoxCategory(ActionEvent actionEvent) {

}

public void ComboBoxAttendanceRate(ActionEvent actionEvent) {

}

public void ComboBoxInstructor(ActionEvent actionEvent) {

}

public void ComboBoxVehicle(ActionEvent actionEvent) {

}

public void TextFieldAge(ActionEvent actionEvent) {

}

public void TextFieldDiscountAmountMax(ActionEvent actionEvent) {

}

public void TextFieldPaidAmount(ActionEvent actionEvent) {

}

public void ComboBoxLicenseType(ActionEvent actionEvent) {

}

public void setStudentService(StudentService studentService) {

this.studentService = studentService;

}

public void TextFieldIDStudent(ActionEvent actionEvent) {

}

public void TextFieldAttendanceRate(ActionEvent actionEvent) {

}

}

**Класс StudentTableViewController**

package org.main.autoschoolapp.controllers;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.beans.property.SimpleObjectProperty;

import javafx.beans.property.SimpleStringProperty;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.image.ImageView;

import javafx.scene.input.InputMethodEvent;

import javafx.stage.Modality;

import javafx.stage.Stage;

import org.main.autoschoolapp.AutoschoolApp;

import org.main.autoschoolapp.model.GroupCategory;

import org.main.autoschoolapp.model.Student;

import org.main.autoschoolapp.service.GroupCategoryService;

import org.main.autoschoolapp.service.StudentService;

import org.main.autoschoolapp.util.Manager;

import java.io.IOException;

import java.net.URL;

import java.util.Comparator;

import java.util.List;

import java.util.Optional;

import java.util.ResourceBundle;

import java.util.stream.Collectors;

import static org.main.autoschoolapp.util.Manager.ShowConfirmPopup;

import static org.main.autoschoolapp.util.Manager.ShowErrorMessageBox;

public class StudentTableViewController implements Initializable {

private final GroupCategoryService categoryService = new GroupCategoryService();

private final StudentService studentService = new StudentService();

@FXML

private ComboBox<String> ComboBoxPaidAmount;

@FXML

private ComboBox<GroupCategory> ComboBoxStudentType;

@FXML

private ComboBox<String> ComboBoxSort;

@FXML

private Button BtnBack;

@FXML

private TableColumn<Student, ImageView> TableColumnPhoto;

@FXML

private TableColumn<Student, Integer> TableColumnAttendanceRate;

@FXML

private TableColumn<Student, Integer> TableColumnPaidAmount;

@FXML

private TableColumn<Student, String> TableColumnAge;

@FXML

private TableColumn<Student, String> TableColumnStudentId;

@FXML

private Button BtnAdd;

@FXML

private Button BtnDelete;

@FXML

private Button BtnUpdate;

@FXML

private TableColumn<Student, String> TableColumnName;

@FXML

private Label LabelInfo;

@FXML

private Label LabelUser;

@FXML

private TextField TextFieldSearch;

@FXML

private TableView<Student> TableViewStudents;

void filterData() {

List<Student> students = studentService.findAll();

int itemsCount = students.size();

// Фильтрация по типу группы

if (!ComboBoxStudentType.getSelectionModel().isEmpty()) {

GroupCategory category = ComboBoxStudentType.getValue();

if (category.getGroupCategoryId() != 0) {

students = students.stream()

.filter(student -> student.getGroupCategory().getGroupCategoryId().equals(category.getGroupCategoryId()))

.collect(Collectors.toList());

}

}

// Фильтрация по сумме оплаты

if (!ComboBoxPaidAmount.getSelectionModel().isEmpty()) {

String discounts = ComboBoxPaidAmount.getValue();

if (discounts.equals("0 - 5 000")) {

students = students.stream().filter(student -> student.getPaidAmount() < 5000).collect(Collectors.toList());

} else if (discounts.equals("5 000 - 15 000")) {

students = students.stream().filter(student -> student.getPaidAmount() >= 5000 && student.getPaidAmount() < 15000).collect(Collectors.toList());

} else if (discounts.equals("15 000 и более")) {

students = students.stream().filter(student -> student.getPaidAmount() >= 15000).collect(Collectors.toList());

}

}

// Сортировка по цене

if (!ComboBoxSort.getSelectionModel().isEmpty()) {

String order = ComboBoxSort.getValue();

if (order.equals("по возрастанию цены")) {

students = students.stream().sorted(Comparator.comparing(Student::getPaidAmount)).collect(Collectors.toList());

} else if (order.equals("по убыванию цены")) {

students = students.stream().sorted(Comparator.comparing(Student::getPaidAmount).reversed()).collect(Collectors.toList());

}

}

// Поиск по имени студента

String searchText = TextFieldSearch.getText();

if (!searchText.isEmpty()) {

students = students.stream().filter(student -> student.getName().toLowerCase().contains(searchText.toLowerCase())).collect(Collectors.toList());

}

// Обновление таблицы

TableViewStudents.getItems().clear();

TableViewStudents.getItems().addAll(students);

// Обновление информации о количестве записей

int filteredItemsCount = students.size();

LabelInfo.setText("Всего записей " + filteredItemsCount + " из " + itemsCount);

}

public void BtnAddAction(ActionEvent actionEvent) {

Student student = TableViewStudents.getSelectionModel().getSelectedItem();

Manager.currentStudent = null;

ShowEditProductWindow();

filterData();

}

void ShowEditProductWindow() {

Stage newWindow = new Stage();

FXMLLoader fxmlLoader = new FXMLLoader(AutoschoolApp.class.getResource("student-edit-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

} catch (IOException e) {

throw new RuntimeException(e);

}

newWindow.setTitle("Изменить данные");

newWindow.initOwner(Manager.secondStage);

newWindow.initModality(Modality.WINDOW\_MODAL);

newWindow.setScene(scene);

Manager.currentStage = newWindow;

newWindow.showAndWait();

Manager.currentStage = null;

filterData();

}

public void BtnUpdateAction(ActionEvent actionEvent) {

Student student = TableViewStudents.getSelectionModel().getSelectedItem();

Manager.currentStudent = student;

ShowEditProductWindow();

filterData();

}

@FXML

void BtnDeleteAction(ActionEvent event) {

Student student = TableViewStudents.getSelectionModel().getSelectedItem();

if (!student.getLessonAttendance().isEmpty())

{

ShowErrorMessageBox("Ошибка целостности данных, у данного клиента есть оценки");

return;

}

Optional<ButtonType> result = ShowConfirmPopup();

if (result.get() == ButtonType.OK) {

studentService.delete(student);

filterData();

}

}

public void BtnBackAction(ActionEvent actionEvent) {

FXMLLoader fxmlLoader = new FXMLLoader(AutoschoolApp.class.getResource("main-view.fxml"));

Scene scene = null;

try {

scene = new Scene(fxmlLoader.load());

scene.getStylesheets().add("base-styles.css");

Manager.secondStage.setScene(scene);

//Manager.mainStage.show();

} catch (IOException e) {

throw new RuntimeException(e);

}

}

public void TextFieldSearchAction(ActionEvent actionEvent) {

filterData();

}

public void TextFieldTextChanged(InputMethodEvent inputMethodEvent) {

}

public void ComboBoxStudentTypeAction(ActionEvent actionEvent) {

filterData();

}

public void ComboBoxPaidAmount(ActionEvent actionEvent) {

filterData();

}

public void ComboBoxSortAction(ActionEvent actionEvent) {

filterData();

}

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

initController();

}

public void initController() {

LabelUser.setText(Manager.currentUser.getFirstName());

List<GroupCategory> categoryList = categoryService.findAll();

categoryList.add(0, new GroupCategory(0L, "Все"));

ObservableList<GroupCategory> categories = FXCollections.observableArrayList(categoryList);

ComboBoxStudentType.setItems(categories);

ObservableList<String> discounts = FXCollections.observableArrayList("Все студенты", "0 - 5 000", "5 000 - 15 000", "15 000 и более");

ComboBoxPaidAmount.setItems(discounts);

ObservableList<String> orders = FXCollections.observableArrayList("по возрастанию цены", "по убыванию цены");

ComboBoxSort.setItems(orders);

setCellValueFactories();

filterData();

}

private void setCellValueFactories() {

TableColumnPhoto.setCellValueFactory(cellData -> {

try {

return new SimpleObjectProperty<ImageView>(cellData.getValue().getImage());

} catch (IOException e) {

throw new RuntimeException(e);

}

});

TableColumnStudentId.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getStudentId()));

TableColumnName.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getName()));

TableColumnAge.setCellValueFactory(cellData -> new SimpleIntegerProperty(cellData.getValue().getAge()).asObject().asString());

TableColumnAttendanceRate.setCellValueFactory(cellData ->

new SimpleIntegerProperty((int) cellData.getValue().getAttendanceRate()).asObject());

TableColumnPaidAmount.setCellValueFactory(cellData -> new SimpleIntegerProperty(cellData.getValue().getPaidAmount()).asObject());

}

}

**Класс VehicleEditViewController**

package org.main.autoschoolapp.controllers;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import javafx.stage.Stage;

import org.main.autoschoolapp.model.Vehicle;

import org.main.autoschoolapp.service.VehicleService;

import org.main.autoschoolapp.util.Manager;

import org.main.autoschoolapp.util.Manager.\*;

import java.net.URL;

import java.util.ResourceBundle;

import static org.main.autoschoolapp.util.Manager.MessageBox;

public class VehicleEditViewController implements Initializable {

private VehicleService vehicleService = new VehicleService();

@FXML

private Button BtnCancel;

@FXML

private Button BtnSave;

@FXML

private TextField TextFieldModel;

@FXML

void BtnCancelAction(ActionEvent event) {

Stage stage = (Stage) BtnCancel.getScene().getWindow();

stage.close();

}

@FXML

void BtnSaveAction(ActionEvent event) {

String error = checkFields().toString();

if (!error.isEmpty()) {

MessageBox("Ошибка", "Заполните поля", error, Alert.AlertType.ERROR);

return;

}

Manager.currentVehicle.setModel(TextFieldModel.getText());

if (Manager.currentVehicle.getVehicleId() == null) {

Manager.currentVehicle.setModel(TextFieldModel.getText());

vehicleService.save(Manager.currentVehicle);

MessageBox("Информация", "", "Данные сохранены успешно", Alert.AlertType.INFORMATION);

} else {

vehicleService.update(Manager.currentVehicle);

MessageBox("Информация", "", "Данные обновлены успешно", Alert.AlertType.INFORMATION);

}

}

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

if (Manager.currentVehicle != null) {

TextFieldModel.setText(Manager.currentVehicle.getModel());

} else {

Manager.currentVehicle = new Vehicle();

}

}

StringBuilder checkFields() {

StringBuilder error = new StringBuilder();

if (TextFieldModel.getText().isEmpty()) {

error.append("Укажите модель\n");

}

return error;

}

}

**Класс VehicleTableViewController**

package org.main.autoschoolapp.controllers;

import javafx.beans.property.SimpleStringProperty;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.\*;

import org.main.autoschoolapp.model.Instructor;

import org.main.autoschoolapp.model.Vehicle;

import org.main.autoschoolapp.service.VehicleService;

import org.main.autoschoolapp.util.Manager;

import java.net.URL;

import java.util.List;

import java.util.Optional;

import java.util.ResourceBundle;

import java.util.stream.Collectors;

import static org.main.autoschoolapp.util.Manager.\*;

public class VehicleTableViewController implements Initializable {

private int itemsCount;

private VehicleService vehicleService = new VehicleService();

@FXML

private TableColumn<Vehicle, String> TableColumnVehicleId;

@FXML

private TableColumn<Vehicle, String> TableColumnModel;

@FXML

private Label LabelInfo;

@FXML

private Label LabelUser;

@FXML

private MenuItem MenuItemAdd;

@FXML

private MenuItem MenuItemBack;

@FXML

private MenuItem MenuItemDelete;

@FXML

private MenuItem MenuItemUpdate;

@FXML

private TableView<Vehicle> TableViewVehicles;

@FXML

private TextField TextFieldSearch;

@FXML

void MenuItemAddAction(ActionEvent event) {

currentVehicle = null;

ShowEditWindow("/org/main/autoschoolapp/vehicle-edit-table-view.fxml");

filterData();

}

@FXML

void MenuItemBackAction(ActionEvent event) {

Manager.LoadSecondStageScene("/org/main/autoschoolapp/main-view.fxml", "Cтуденты");

}

@FXML

void MenuItemDeleteAction(ActionEvent event) {

Vehicle unittype = TableViewVehicles.getSelectionModel().getSelectedItem();

if (!unittype.getStudents().isEmpty()) {

ShowErrorMessageBox("Ошибка целостности данного автомобиля есть зависимости.");

return;

}

Optional<ButtonType> result = ShowConfirmPopup();

if (result.get() == ButtonType.OK) {

vehicleService.delete(unittype);

filterData();

}

}

@FXML

void MenuItemUpdateAction(ActionEvent event) {

Vehicle vehicle = TableViewVehicles.getSelectionModel().getSelectedItem();

currentVehicle = vehicle;

ShowEditWindow("/org/main/autoschoolapp/vehicle-edit-table-view.fxml");

filterData();

}

@FXML

void TextFieldSearchAction(ActionEvent event) {

filterData();

}

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

initController();

}

public void initController() {

LabelUser.setText(Manager.currentUser.getFirstName());

List<Vehicle> unittypeList = vehicleService.findAll();

unittypeList.add(0, new Vehicle(0L, "Все"));

ObservableList<Vehicle> unittypes = FXCollections.observableArrayList(unittypeList);

TableViewVehicles.setItems(unittypes);

setCellValueFactories();

filterData();

}

void filterData() {

List<Vehicle> vehicle = vehicleService.findAll();

itemsCount = vehicle.size();

String searchText = TextFieldSearch.getText();

if (!searchText.isEmpty()) {

vehicle = vehicle.stream().filter(unittypes -> unittypes.getModel().toLowerCase().contains(searchText.toLowerCase())).collect(Collectors.toList());

}

TableViewVehicles.getItems().clear();

for (Vehicle vehicles : vehicle) {

TableViewVehicles.getItems().add(vehicles);

}

int filteredItemsCount = vehicle.size();

LabelInfo.setText("Всего записей " + filteredItemsCount + " из " + itemsCount);

TableViewVehicles.setColumnResizePolicy(TableView.CONSTRAINED\_RESIZE\_POLICY\_FLEX\_LAST\_COLUMN);

}

private void setCellValueFactories() {

TableColumnVehicleId.setCellValueFactory(cellData ->

new SimpleStringProperty(String.valueOf(cellData.getValue().getVehicleId())));

TableColumnModel.setCellValueFactory(cellData -> new SimpleStringProperty(cellData.getValue().getModel()));

}

}

**Приложение B**

**Диск с исполняемым кодом программного модуля**