|  |  |
| --- | --- |
| ­­­  **К Г Э У** | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»**  (ФГБОУ ВО «КГЭУ») |

**Документация к АРМ администратора автосалона**

**Дисциплина: Проектный практикум**

|  |  |
| --- | --- |
| **Выполнил:** | Гайфутдинов И.И. |
| **Группа:** | ТРП-1-20 |
| **Проверил:** | Алексеев И.П. |
|  |  |
|  |  |

Казань – 2023-2024

**Оглавление**

[**1. Введение** 3](#_Toc161242573)

[**2. Зависимости** 3](#_Toc161242574)

[**3. Структура проекта** 3](#_Toc161242575)

[**4. База данных** 4](#_Toc161242576)

[**5. main.py** 4](#_Toc161242577)

[**6. db.py** 5](#_Toc161242578)

[**7. login\_window.py** 7](#_Toc161242579)

[**8. main\_window.py** 9](#_Toc161242580)

[**9. tab\_manager** 10](#_Toc161242581)

[**10. Основные функциональные возможности** 10](#_Toc161242582)

[**11. Инструкция по установке и запуску** 11](#_Toc161242583)

[**12. Инструкция по использованию** 12](#_Toc161242584)

[**13. Заключение** 13](#_Toc161242585)

**Документация**

# **1. Введение**

Проект "АРМ администратора автосалона" представляет собой программу для управления информацией об автомобилях, клиентах и заявках на просмотр. Программа разработана на языке программирования Python с использованием библиотеки Tkinter для создания графического интерфейса пользователя (GUI). В качестве базы данных используется SQLite.

# **2. Зависимости**

- Python 3.x

- Tkinter

- tkcalendar

- SQLite

- Pillow

# **3. Структура проекта**

Проект состоит из следующих основных компонентов:

- `main.py`: Основной файл, содержащий точку входа в программу и инициализацию основных компонентов.

- `db.py`: Модуль для работы с базой данных SQLite, включающий класс `Database`.

- `login\_window.py`: Модуль для создания окна авторизации с использованием библиотеки Tkinter.

- `main\_window.py`: Модуль для создания главного окна программы, предоставляющего функциональность по управлению автомобилями, клиентами и заявками.

# **4. База данных**

База данных включает в себя основные таблицы:

- `users`: Хранит информацию о пользователях для авторизации.

- `cars`: Содержит данные об автомобилях, такие как марка, цвет, год выпуска и другие.

- `clients`: Содержит информацию о клиентах, включая ФИО, год рождения, пол и дату регистрации.

- `applications`: Хранит заявки на просмотр автомобилей, включая связи с конкретными автомобилями и клиентами.

# **5. main.py**

a) Импорт модулей:

- tkinter as tk: Импортируется для создания графического интерфейса пользователя (GUI).

- Database: Импортируется для взаимодействия с базой данных SQLite.

- LoginWindow: Импортируется для создания окна авторизации.

- MainWindow: Импортируется для создания главного окна приложения.

b) Функция main():

db = Database(): Создается объект для взаимодействия с базой данных.

root = tk.Tk(): Создается главное окно приложения.

on\_successful\_login(user): Функция, вызываемая при успешной авторизации пользователя.

current\_user = db.get\_username\_by\_id(user): Получает имя пользователя по его ID.

main\_window = MainWindow(tk.Toplevel(root), db, current\_user): Создает главное окно приложения и передает ему соединение с базой данных и имя пользователя.

root.withdraw(): Скрывает окно авторизации.

login\_window = LoginWindow(root, db, on\_successful\_login): Создает окно авторизации.

root.bind("<Escape>", lambda q: exit()): Привязывает нажатие клавиши Esc к выходу из приложения.

root.mainloop(): Запускает главный цикл обработки событий GUI.

db.close\_connection(): Закрывает соединение с базой данных после завершения работы приложения.

c) Запуск программы:

if \_\_name\_\_ == "\_\_main\_\_":: Обеспечивает запуск функции main() только при непосредственном запуске скрипта, а не при импортировании модуля.

Резюме:

Код main.py задает базовую структуру приложения, включая создание главного окна, подключение к базе данных, запуск окна авторизации и переход к главному окну при успешной авторизации.

# **6. db.py**

a) Класс Database:

Конструктор:

Устанавливает соединение с базой данных автосалон.db.

Создает курсор для выполнения SQL-запросов.

При необходимости создает таблицы пользователей, автомобилей, клиентов и заявок на просмотр.

Методы для работы с пользователями:

create\_users\_table(): Создает таблицу пользователей, если она не существует.

check\_user\_credentials(username, password): Проверяет наличие пользователя с указанными логином и паролем.

Методы для работы с автомобилями:

create\_tables(): Создает таблицы автомобилей, клиентов и заявок, если они не существуют.

add\_car(...): Добавляет новый автомобиль в базу данных.

get\_cars(): Возвращает список всех автомобилей.

delete\_car(car\_id): Удаляет автомобиль по его ID.

edit\_car(...): Редактирует информацию об автомобиле.

get\_car(car\_id): Возвращает информацию об автомобиле по его ID.

get\_car\_name\_by\_id(car\_id): Возвращает название автомобиля по его ID.

Методы для работы с клиентами:

add\_client(...): Добавляет нового клиента в базу данных.

get\_clients(): Возвращает список всех клиентов.

delete\_client(client\_id): Удаляет клиента по его ID.

edit\_client(...): Редактирует информацию о клиенте.

get\_client(client\_id): Возвращает информацию о клиенте по его ID.

get\_client\_name\_by\_id(client\_id): Возвращает полное имя клиента по его ID.

Методы для работы с заявками:

add\_application(...): Добавляет новую заявку на просмотр автомобиля.

get\_applications(): Возвращает список всех заявок.

delete\_application(application\_id): Удаляет заявку по ее ID.

edit\_application(...): Редактирует информацию о заявке.

get\_application(application\_id): Возвращает информацию о заявке по ее ID.

Закрытие соединения:

close\_connection(): Закрывает соединение с базой данных.

Резюме:

Код db.py обеспечивает взаимодействие между приложением и базой данных SQLite. Он содержит класс Database, который позволяет создавать таблицы, добавлять, удалять и редактировать данные, а также выполнять различные запросы для получения информации из базы данных.

# **7. login\_window.py**

a) Класс LoginWindow:

Конструктор:

Принимает объект root главного окна, объект database для взаимодействия с базой данных и функцию on\_successful\_login для вызова при успешной авторизации.

Устанавливает заголовок окна "Логин".

Применяет базовый белый цвет фона.

Создает виджеты окна.

Метод create\_widgets():

Настраивает стили для кнопок и полей ввода.

Создает метки для логина и пароля.

Создает поля ввода для логина и пароля (пароль отображается звездочками).

Создает кнопку "Войти", привязанную к методу check\_login().

Метод check\_login():

Получает введенные логин и пароль из полей ввода.

Проверяет их корректность, используя метод check\_user\_credentials() объекта database.

Если логин и пароль верны, выводит сообщение об успешном входе и вызывает функцию on\_successful\_login(), передавая ей ID пользователя.

Если логин или пароль неверны, выводит сообщение об ошибке.

Резюме:

Код login\_window.py отвечает за создание и функционирование окна авторизации. Он создает простую форму для ввода логина и пароля, а также обеспечивает проверку введенных данных и переход к главному окну при успешной авторизации.

# **8. main\_window.py**

a) Основной класс:

MainWindow: управляет главным окном приложения и взаимодействием с базой данных.

b) Вкладки:

Автомобили: создает таблицу для отображения автомобилей, включая кнопки добавления, удаления и редактирования.

Клиенты: создает таблицу для отображения клиентов, включая кнопки добавления, удаления и редактирования.

Заявки на просмотр: создает таблицу для отображения заявок на просмотр автомобилей, включая кнопки добавления, удаления, редактирования и отметки как показанной.

c) База данных:

Класс MainWindow принимает объект базы данных в конструкторе, позволяя взаимодействовать с ней.

d) Основные функции:

Добавление элементов: для автомобилей, клиентов и заявок есть отдельные окна для добавления новых данных.

Удаление элементов: позволяет удалять выбранные элементы из таблиц.

Редактирование элементов: открывает окна для редактирования существующих данных.

Обновление таблиц: после изменений в данных обновляет содержимое таблиц.

Отметка заявки как показанной: помечает заявку на просмотр как просмотренную.

e) Работа с изображениями автомобилей:

При добавлении или редактировании автомобиля можно загрузить его изображение.

Изображение хранится в базе данных и отображается в таблице.

# **9. tab\_manager**

Модуль является шаблоном/заготовкой для написания интерфейсов для удобной масштабируемости приложения

**tab\_manager.py** – отвечает за менеджера вкладок

**entity.py** – отвечает за шаблон и поведение создаваемой новой сущности

**tab.py** – отвечает за интерфейс представления новой сущности

Дополнительная информация по настройке и доработке указано в коде в виде комментариев

# **10. Основные функциональные возможности**

***10.1. Авторизация***

Пользователь входит в систему с использованием логина и пароля. После успешной авторизации открывается главное окно программы в виде табличного представления.

***10.2. Управление Автомобилями***

Администратор может просматривать, добавлять, редактировать и удалять записи об автомобилях. Данные об автомобилях включают марку, цвет, год выпуска, объем двигателя, лошадиные силы и тип коробки передач.

***10.3. Управление Клиентами***

Программа предоставляет функциональность по управлению информацией о клиентах. Администратор может добавлять, редактировать и удалять записи о клиентах, включая ФИО, год рождения, пол и дату регистрации.

***10.4. Управление Заявками на Просмотр***

Система позволяет администратору просматривать, добавлять, редактировать и удалять заявки на просмотр автомобилей. Заявки содержат информацию об автомобиле, клиенте и дате просмотра.

***10.5. Управление Пользователями***

Система позволяет администратору системы просматривать, добавлять, редактировать и удалять пользователей в приложении. Также имеется возможность назначения прав администратора системы.

# **11. Инструкция по установке и запуску**

1. Убедитесь, что у вас установлен Python 3.x.

2. Установите необходимые библиотеки (есть файл requirements.txt).

3. Загрузите все файлы проекта в одну директорию.

4. Запустите `main.py` с использованием Python.

# **12. Инструкция по использованию**

В главном окне программы вы увидите три вкладки:

Автомобили: В этой вкладке вы можете добавлять, редактировать и удалять автомобили.

Клиенты: В этой вкладке вы можете добавлять, редактировать и удалять клиентов.

Заявки на просмотр: В этой вкладке вы можете добавлять, редактировать и удалять заявки на просмотр.

a) Добавление нового объекта:

Для добавления нового объекта нажмите кнопку "Добавить" в соответствующей вкладке.

Например, чтобы добавить новый автомобиль, нажмите кнопку "Добавить" на вкладке "Автомобили". В открывшемся окне введите данные нового автомобиля и нажмите кнопку "Сохранить".

b) Редактирование существующего объекта:

Для редактирования существующего объекта выберите его в списке и нажмите кнопку "Редактировать".

В открывшемся окне внесите необходимые изменения и нажмите кнопку "Сохранить".

c) Удаление существующего объекта:

Для удаления существующего объекта выберите его в списке и нажмите кнопку "Удалить".

d) Отметка заявки на просмотр как показанной:

Чтобы отметить заявку на просмотр как показанную, выполните следующие действия:

Выберите заявку в списке заявок.

Нажмите кнопку "Отметить как показанную".

с) Показать все заявки:

Чтобы показать все заявки на просмотр:

Нажмите кнопку "Все".

# **13. Заключение**

Проект предоставляет удобное рабочее место для администратора автосалона, обеспечивая функциональность по управлению информацией об автомобилях, клиентах и заявках на просмотр.