

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(МОСКОВСКИЙ ПОЛИТЕХ)

ДОПУЩЕНА К ЗАЩИТЕ

Заведующий кафедрой

\_\_\_\_\_ / Е.А. Пухова/

Руководитель образовательной  
программы

\_\_\_\_\_ / М.В.  
Даньшина /

**«СОЗДАНИЕ ПРИЛОЖЕНИЯ ДЛЯ СИСТЕМЫ  
АВТОСТРАХОВАНИЯ»**

Курсовая работа

по направлению 09.03.01 Информатика и вычислительная техника

Образовательная программа «Информационные технологии управления  
бизнесом»

Студент: \_\_\_\_\_ / Дукага Кобонго Грег Давид 221-365 /  
*подпись* *ФИО, группа*

Руководитель ВКР: \_\_\_\_\_ / Армаш М.Н., ст.  
преподаватель /

*подпись*

*ФИО, уч. звание и степень*

Москва, 2023

# Оглавление

ВВЕДЕНИЕ .....	4
1. ОСНОВНАЯ ЧАСТЬ .....	6
1. Предметная область и технологии .....	6
1.1 Актуальность .....	6
1.2 Инструменты для создания .....	6
2. Обзор аналогов .....	6
3. Вывод.....	8
2. ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ .....	9
2.1 Структура базы данных .....	9
2.2 Описание функций программы.....	10
3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	13
3.1 Регистрация нового пользователя.....	13
3.2 Изменение данных .....	13

3.3	Просмотр данных .....	14
4.	РУКОВОДСТВО ПРОГРАММИСТА .....	16
	ЗАКЛЮЧЕНИЕ .....	17
	Список источников информации .....	18

## **ВВЕДЕНИЕ**

В данном проекте будет создано приложение для хранения информации по автострахованию. Эта программа - это клиент-серверное приложение для управления базой данных автомобильной страховой компании. Она позволяет зарегистрировать нового пользователя, добавить машину и полис, а также удалить запросы, машины, полисы и клиентов.

Каждому водителю настоятельно рекомендуется оформить для своей машины страховой полис, чтобы в случае аварии за ущерб платил не он сам, а его страховая компания. Приложение, которое будет создано в данном проекте, подразумевает систему, в которой клиенты могут регистрироваться и вносить информацию о своих автомобилях, а Администратор может регулировать весь поток данных.

Данный проект поможет обеспечить хранение данных по автомобилям, их страховке (КАСКО, ОСАГО), и самое главное — конфиденциальность информации. Система устроена таким образом, что каждый клиент страховой компании может видеть только свои данные и данные по своему автомобилю.

Однако сотрудник компании, имеющий роль Администратора в приложении, имеет право распоряжаться всей базой данных: добавление, изменение а также удаление данных.

В целом, программист отвечает за создание программного обеспечения, которое помогает автоматизировать процессы в организации, повышает эффективность и улучшает взаимодействие с базой данных и пользовательским интерфейсом.

Программист должен иметь хорошее понимание процессов базы данных и быть в состоянии создавать запросы SQL для получения, изменения, добавления и удаления данных. Кроме того, он должен иметь навыки создания графического интерфейса для пользователей, а также обработки ошибок и проверки правильности введенных данных.

В целом, данная программа представляет собой небольшую систему управления базой данных с графическим интерфейсом пользователя. Пользователь может взаимодействовать с базой данных через форму ввода имени пользователя и пароля и выпадающее меню с именами таблиц в базе данных. Функции позволяют отображать данные из выбранной таблицы, искать данные в таблице по запросу пользователя, выводить информацию о конкретной строке таблицы и сохранять изменения в базе данных.

# **1. ОСНОВНАЯ ЧАСТЬ**

## **1. Предметная область и технологии**

### **1.1 Актуальность**

Несчастные случаи, а также инциденты, связанные с автомобилями, увеличиваются изо дня в день. На сегодняшний день как никогда важно развивать систему страхования. С автомобилем может случиться все, что угодно: от маленькой царапины до автокатастрофы. Система страхования устроена таким образом, что при наличии автостраховки компании страхования возместят пострадавшему весь ущерб.

### **1.2 Инструменты для создания**

Для создания этого проекта используется SQL и Python. SQL используется для создания базы данных, язык программирования Python используется для создания кода программы и регулировки внешнего вида окон.

Данная программа представляет собой графический интерфейс пользователя для взаимодействия с базой данных "System". Программа написана на языке Python с использованием PyQt5 и PostgreSQL.

## **2. Обзор аналогов**

### **АльфаСтрахование**

#### **1. Навигация**

Навигация сайта АльфаСтрахования простая и понятная. Есть вертикальная навигация по темам. На горизонтальной панели вверху есть кнопки «Войти» и «Помощь». Это означает, что пользователи не будут теряться на сайте.

#### **2. Дизайн**

Дизайн сайта АльфаСтрахование выполнен в стиле АльфаБанка — красный цвет. Но большая часть это белый цвет. Поэтому это всё вместе выглядит хорошо. Панели и навигация в красном цвете и текст с основной информацией на белом фоне.

### 3. Сильные и слабые стороны проекта

Сильные стороны проекта АльфаСтрахование это понятная навигация и хороший дизайн. Плохие стороны этого проекта это большое количество дополнительных кнопок на экране. Например, «Сообщение» или «Оцените нас»

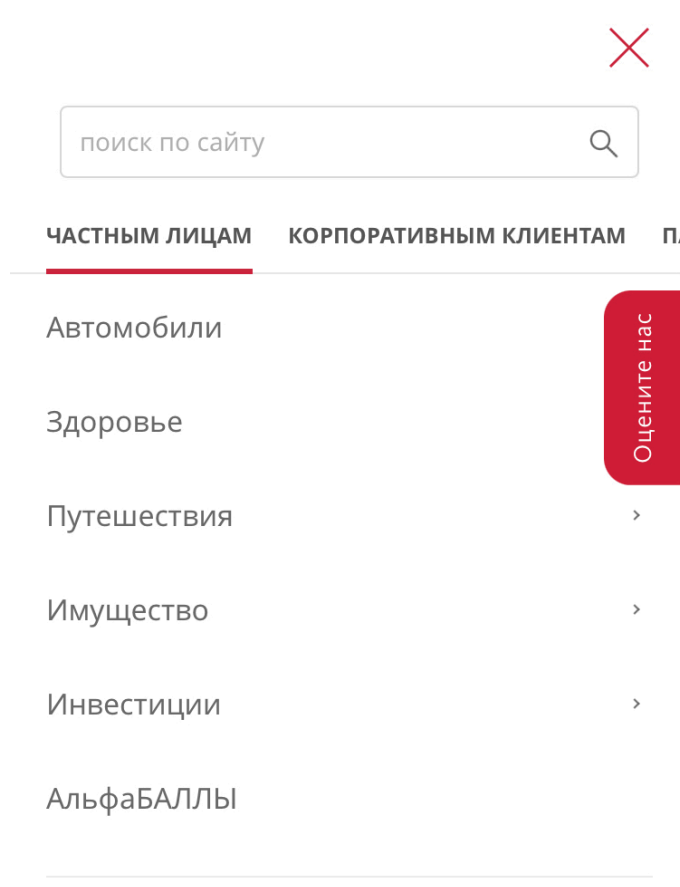


Рисунок 1 - Навигация  
АльфаСтрахование

### 4. Вывод

АльфаСтрахование это хороший сайт и можно взять за основу дизайн, но нужно добавлять меньше кнопок на экране.

### 3. Вывод

Для создания приложения для автострахования нужно внимательно работать над Python код, ведь важная часть проекта это конфиденциальность данных. Каждый пользователь может видеть только свою информацию, не должен видеть информацию других людей. Также сделать систему для связи между пользователем и администратором приложения.



Рисунок 2 - Дизайн  
АльфаСтрахование



## 2. ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ

### 2.1 Структура базы данных

Ниже представлен ER-diagram базы данных для проекта.

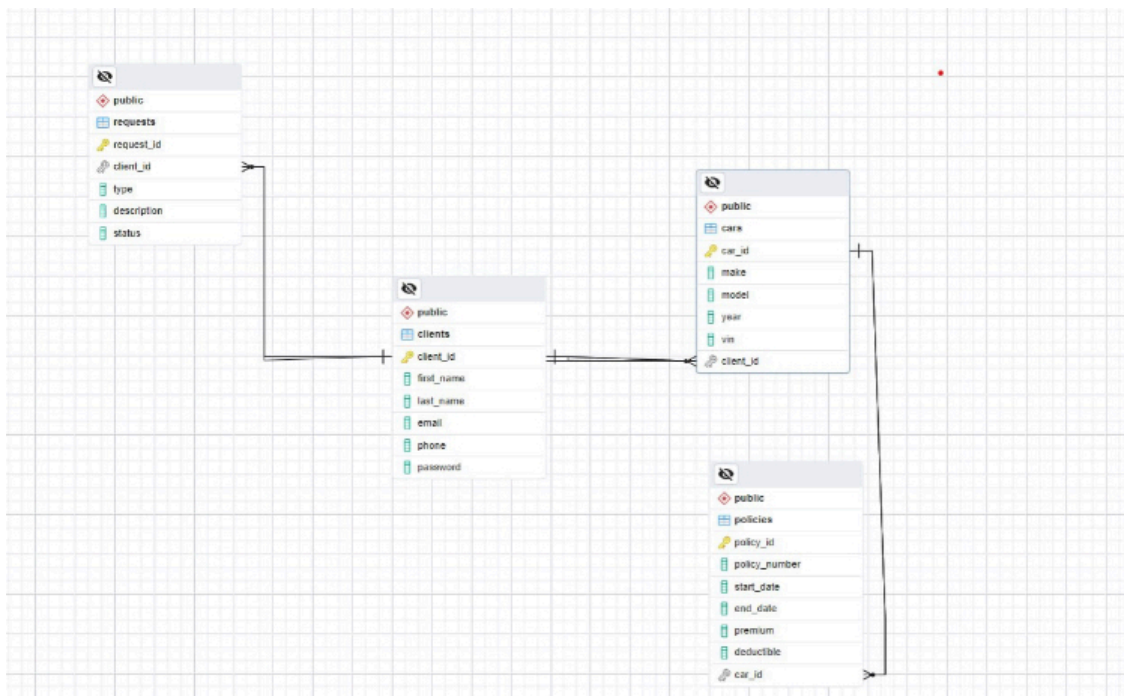


Рисунок 3 - ER-diagram

Эти три таблицы и одна связующая (requests) образуют схему базы данных для этой системы автомобильной страховой компании.

Clients - это таблица, которая содержит информацию о каждом клиенте, включая уникальный идентификатор `client_id`, имя, фамилию, электронную почту, номер телефона и хэш пароля.

Cars - это таблица, которая содержит информацию об автомобилях, зарегистрированных у клиентов. Она содержит уникальный идентификатор `car_id`, марку, модель, год выпуска, VIN-код и `client_id`, который связывает машину с определенным клиентом из таблицы Clients.

Policies - это таблица, которая содержит информацию о полисах, приобретенных клиентами. Она содержит уникальный идентификатор `policy_id`, номер полиса, дату начала и окончания страхования, премию (стоимость) полиса, сумму вычета и `car_id`, который связывает полис с определенной машиной из таблицы Cars.

requests - это таблица, которая содержит информацию о запросах клиентов, относящихся к различным аспектам работы страховой компании, таким как запросы на изменения полисов, запросы на ремонт автомобиля и т.д. Эта таблица содержит уникальный идентификатор `request_id`, идентификатор клиента `client_id`, тип запроса `type`, описание запроса `description` и его статус `status`.

## 2.2 Описание функций программы

Каждая функция использует библиотеку `psycopg2` для подключения к PostgreSQL серверу и выполнения запросов. Все изменения в базе данных сохраняются после выполнения транзакции командой `conn.commit()`. Если произошла ошибка при работе с базой данных, в списке виджетов выводится сообщение об ошибке.

**Функция `login()`** выполняет первоначальное подключение к базе данных, проверяет наличие введенного пользователем имени и пароля в базе данных "clients" или является ли он администратором. Если имя и пароль найдены в базе данных "clients", то отображаются соответствующие сообщения и выпадающее меню, содержащее имена таблиц в базе данных. Если имя и пароль равны "admin" и "12345", то выпадающее меню также будет содержать таблицу "requests". Далее происходит вызов функции `display_table_data()`, которая выводит данные из выбранной таблицы в виджет таблицы.

**Функция `display_table_data(selected_table)`** получает имя выбранной таблицы и выполняет запрос на извлечение атрибутов и значений из этой таблицы. Запрос формируется в зависимости от того, выполняет ли вход администратор или обычный пользователь. Если выбранная таблица - "clients", то отображаются только данные пользователя, который выполнил вход. Если выбрана таблица "cars" или "requests", отображаются данные конкретного пользователя. Если выбрана таблица "policies", отображаются данные всех машин пользователя.

**Функция `search()`** ищет совпадения в таблице по введенному пользователем запросу и подсвечивает ячейки, в которых найдено совпадение.

```
def register_user():
    global conn
    Form3, Windows3 = uic.loadUiType('connexion.ui')
    windows3 = Windows3()
    form3 = Form3()
    form3.setupUi(windows3)
```

Рисунок 4 - Функция `register_user()`

**Функция `display_selected_row_data(selected_table, selected_id)`** выводит данные о выбранной строке таблицы в виджет таблицы.

**Функция `save_table_data()`** сохраняет изменения в базе данных. Она получает имя выбранной таблицы, имя и пароль пользователя, а затем проходит по всем измененным ячейкам и обновляет значения записей в базе данных. Если выбрана таблица "policies", то обновление выполняется по полю `policy_id`, а для других таблиц - по полю `client_id`.

**Функция register\_user()** отвечает за регистрацию нового пользователя. Она использует модуль `ui` для загрузки интерфейса пользователя из файла `.ui`. После того как пользователь заполнил все необходимые поля, данные отправляются на сервер, где они проверяются на дубликаты (электронная почта или номер телефона) в базе данных. Если данные уникальны, они добавляются в таблицу `clients`. Затем создается новая запись в таблице `requests`, связанная с этим клиентом.

```
def search():
    search_query = form.lineEdit.text()

    pattern = re.compile(search_query, re.IGNORECASE)

    for i in range(form.tableWidget.rowCount()):
        for j in range(form.tableWidget.columnCount()):

            cell_text = form.tableWidget.item(i, j).text()

            match = re.search(pattern, cell_text)

            if match:
                item = form.tableWidget.item(i, j)
                item.setBackground(QtGui.QColor('yellow'))
            else:
                item = form.tableWidget.item(i, j)
                item.setBackground(QtGui.QColor('white'))
```

Рисунок 5 - Функция search()

**Функция add()** добавляет в базу данных информацию об автомобиле и полисе. Сначала она проверяет код доступа, введенный пользователем. Если код верный, загружается интерфейс пользователя и заполняются соответствующие поля. Затем данные отправляются на сервер и добавляются в таблицы `cars` и `policies`.

**Функция delete()** удаляет записи из базы данных. Как и в функции `add()`, сначала проверяется код доступа. Затем пользователь может выбрать, какую запись он хочет удалить - запросы, машины, полисы или клиентов. Затем выбранная запись удаляется из соответствующей таблицы.

### 3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

#### 3.1 Регистрация нового пользователя

Для регистрации нажмите на кнопку «Регистрация нового клиента», после чего введите свои данные во всплывающем окне:

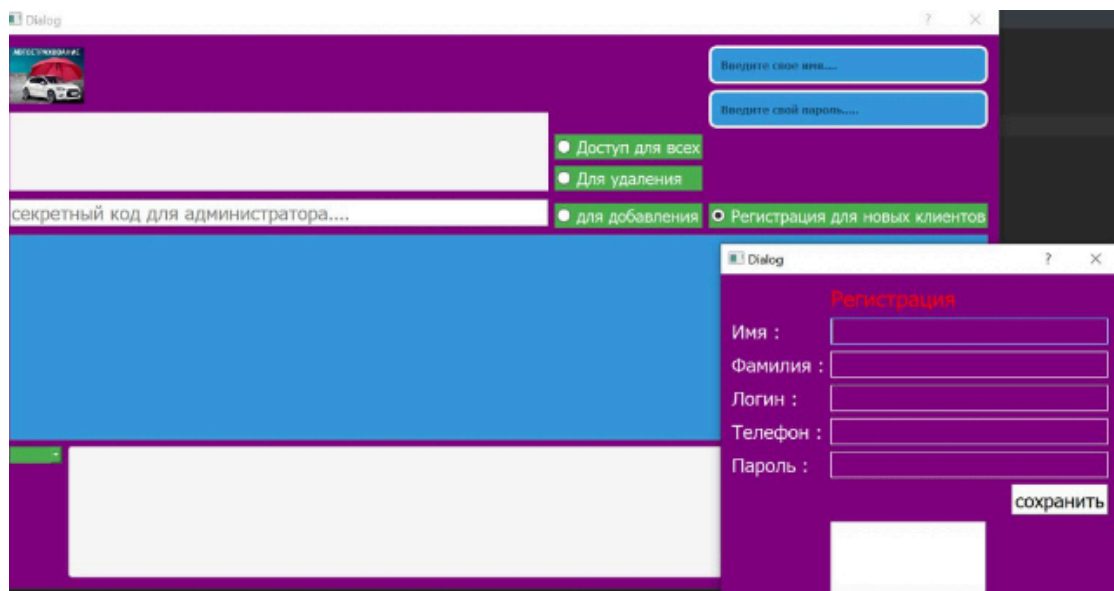


Рисунок 6 - Регистрация

#### 3.2 Изменение данных

Для изменения данных необходимо обратиться к Администратору приложения через вкладку requests. Тогда администратор увидит созданный пользователем запрос:

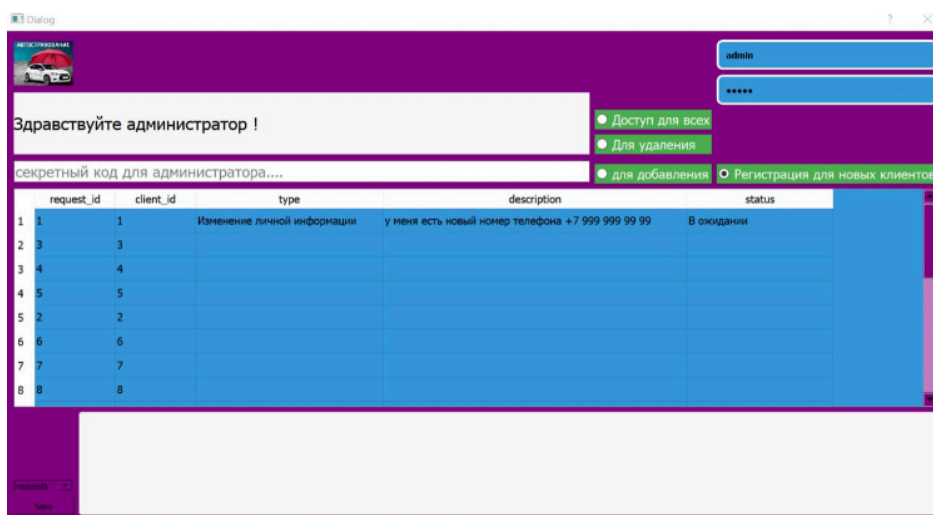
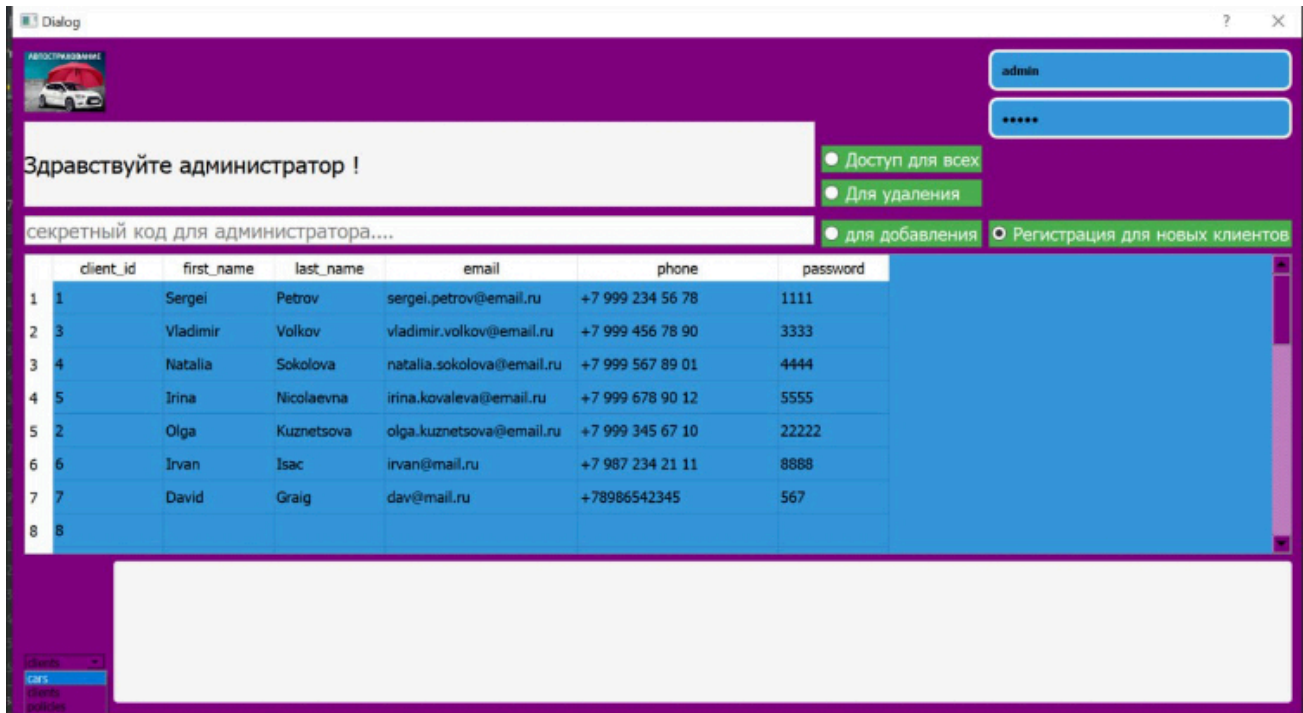


Рисунок 7 - Запрос пользователя на изменение данных

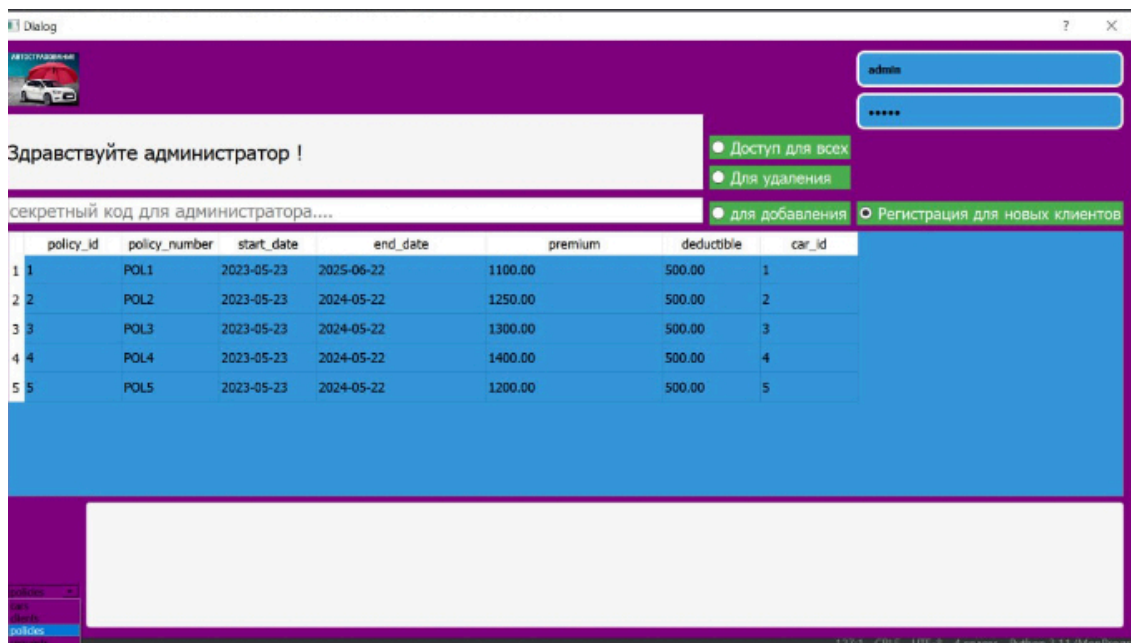
### 3.3 Просмотр данных

Администратор имеет возможность просматривать все данные по клиентам, а также их автомобилям и страховым полисам:



	client_id	first_name	last_name	email	phone	password
1	1	Sergei	Petrov	sergei.petrov@email.ru	+7 999 234 56 78	1111
2	3	Vladimir	Volkov	vladimir.volkov@email.ru	+7 999 456 78 90	3333
3	4	Natalia	Sokolova	natalia.sokolova@email.ru	+7 999 567 89 01	4444
4	5	Irina	Nicolaevna	irina.kovaleva@email.ru	+7 999 678 90 12	5555
5	2	Olga	Kuznetsova	olga.kuznetsova@email.ru	+7 999 345 67 10	22222
6	6	Irvan	Isac	irvan@mail.ru	+7 987 234 21 11	8888
7	7	David	Graig	dav@mail.ru	+78986542345	567
8	8					

Рисунок 8 - Данные по клиентам



	policy_id	policy_number	start_date	end_date	premium	deductible	car_id
1	1	POL1	2023-05-23	2025-06-22	1100.00	500.00	1
2	2	POL2	2023-05-23	2024-05-22	1250.00	500.00	2
3	3	POL3	2023-05-23	2024-05-22	1300.00	500.00	3
4	4	POL4	2023-05-23	2024-05-22	1400.00	500.00	4
5	5	POL5	2023-05-23	2024-05-22	1200.00	500.00	5

Рисунок 9 - Данные по страховым полисам

Dialog

АВТОСТРАХОВАНИЕ

Здравствуйтесь администратор !

секретный код для администратора....

admin

\*\*\*\*\*

☐ Доступ для всех  
☒ Для удаления  
☐ для добавления

☒ Регистрация для новых клиентов

	car_id	make	model	year	vin	client_id
1	1	Nissan	Sentra	2021	3N1AB8CV5M...	1
2	2	Hyundai	Elantra	2020	5NPD84LF6LH...	2
3	3	Honda	Civic	2022	19XFC2F5XNE...	3
4	4	Ford	Focus	2019	1FADP3K29JL...	4
5	5	Volkswagen	Jetta	2022	3VWC57BU3M...	5
6	6	Toyota	civic	2004	5GJKE8NHJKLR	6

cars

Save

Рисунок 10 - Данные по автомобилям

Администратор может добавлять новых клиентов в таблицу Clients, добавлять новые машины и полисы для существующих клиентов, а также просматривать информацию по запросам клиентов в таблице requests. Клиенты могут создавать запросы в таблице requests, запрашивая изменения своих полисов или ремонт своих автомобилей. Таким образом, эти таблицы обеспечивают функциональность системы страховой компании.

## 4. РУКОВОДСТВО ПРОГРАММИСТА

Глобальные переменные `conn`, `db` и `save_button` инициализируются со значением `None` в самом начале программы. Переменная `conn` будет использоваться для установления соединения с базой данных, а переменная `db` хранит имя базы данных. Переменная `save_button` хранит ссылку на объект кнопки сохранения, которая создается при первом нажатии на ячейку таблицы.



## **ЗАКЛЮЧЕНИЕ**

В контексте данной системы страховой компании, программист отвечает за создание программного обеспечения, которое предоставляет удобный и надежный интерфейс для работы с базой данных. Эта программа может быть написана на одном из многих языков программирования, таких как Python, Java или C++, и должна быть способна связываться с PostgreSQL сервером, чтобы получать доступ к базе данных.

## **Список источников информации**

[alfastrah.ru](http://alfastrah.ru)