**ПРИЛОЖЕНИЯ**

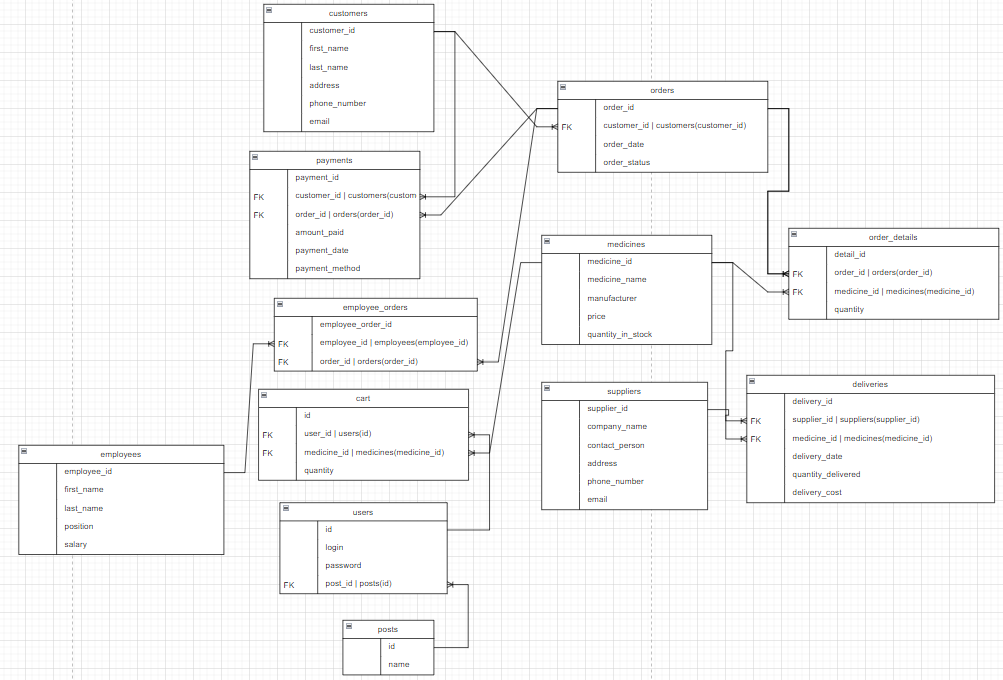
Приложения А. ERD-диаграмма

Приложения Б. Код базы данных

Приложения В. Код приложения

Приложения Г. Скриншоты работы приложения

**Приложение А**

****

**Приложение Б**

CREATE TABLE IF NOT EXISTS posts (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name VARCHAR(100));

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

login VARCHAR(100),

password VARCHAR(100),

post\_id INTEGER,

FOREIGN KEY (post\_id) REFERENCES posts(id));

CREATE TABLE IF NOT EXISTS suppliers (

supplier\_id INTEGER PRIMARY KEY,

company\_name VARCHAR(255),

contact\_person VARCHAR(255),

address VARCHAR(255),

phone\_number VARCHAR(15),

email VARCHAR(255));

CREATE TABLE IF NOT EXISTS medicines (

medicine\_id INTEGER PRIMARY KEY,

medicine\_name VARCHAR(255),

manufacturer VARCHAR(255),

price INTEGER,

quantity\_in\_stock INTEGER);

CREATE TABLE IF NOT EXISTS employees (

employee\_id INTEGER PRIMARY KEY,

first\_name VARCHAR(255),

last\_name VARCHAR(255),

position VARCHAR(100),

salary INTEGER);

CREATE TABLE IF NOT EXISTS customers (

customer\_id INTEGER PRIMARY KEY,

first\_name VARCHAR(255),

last\_name VARCHAR(255),

address VARCHAR(255),

phone\_number VARCHAR(15),

email VARCHAR(255));

CREATE TABLE IF NOT EXISTS orders (

order\_id INTEGER PRIMARY KEY,

customer\_id INTEGER,

order\_date DATE,

order\_status VARCHAR(50),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id));

CREATE TABLE IF NOT EXISTS order\_details (

detail\_id INTEGER PRIMARY KEY,

order\_id INTEGER,

medicine\_id INTEGER,

quantity INTEGER,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (medicine\_id) REFERENCES medicines(medicine\_id));

CREATE TABLE IF NOT EXISTS employee\_orders (

employee\_order\_id INTEGER PRIMARY KEY,

employee\_id INTEGER,

order\_id INTEGER,

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id),

FOREIGN KEY (order\_id) REFERENCES orders(order\_id));

CREATE TABLE IF NOT EXISTS deliveries (

delivery\_id INTEGER PRIMARY KEY,

supplier\_id INTEGER,

medicine\_id INTEGER,

delivery\_date DATE,

quantity\_delivered INTEGER,

delivery\_cost INTEGER,

FOREIGN KEY (supplier\_id) REFERENCES suppliers(supplier\_id),

FOREIGN KEY (medicine\_id) REFERENCES medicines(medicine\_id));

CREATE TABLE IF NOT EXISTS payments (

payment\_id INTEGER PRIMARY KEY,

customer\_id INTEGER,

order\_id INTEGER,

amount\_paid INTEGER,

payment\_date DATE,

payment\_method VARCHAR(50),

FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id));

CREATE TABLE IF NOT EXISTS cart (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

medicine\_id INTEGER,

quantity INTEGER,

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (medicine\_id) REFERENCES medicines(medicine\_id));

INSERT INTO posts (name) VALUES

('Администратор'), ('Пользователь'), ('Сотрудник');

INSERT INTO users (login, password, post\_id) VALUES

('admin', 'admin', 1),

('user1', 'user', 2), ('user2', 'user', 2), ('user3', 'user', 2), ('user4', 'user', 2), ('user5', 'user', 2),

('employee', 'employee', 3);

INSERT INTO suppliers (supplier\_id, company\_name, contact\_person, address, phone\_number, email) VALUES

(1, 'Фармацевтика АО', 'Иван Иванов', 'ул. Примерная, 1, Город, Страна', '1234567890', 'ivan@example.com'),

(2, 'Медикал Трейд', 'Мария Смирнова', 'пр. Центральный, 2, Город, Страна', '0987654321', 'maria@example.com'),

(3, 'Здоровье для всех', 'Алексей Кузнецов', 'пр-т Солнечный, 3, Город, Страна', '1112223333', 'alex@example.com'),

(4, 'Медснаб', 'Елена Попова', 'ул. Московская, 4, Город, Страна', '5556667777', 'elena@example.com'),

(5, 'Фармкомплект', 'Дмитрий Иванов', 'пр-т Ленина, 5, Город, Страна', '9998887777', 'dmitry@example.com'),

(6, 'Аптека РУ', 'Анна Соколова', 'ул. Зеленая, 6, Город, Страна', '4445556666', 'anna@example.com'),

(7, 'Медсервис', 'Артем Васильев', 'пер. Луговой, 7, Город, Страна', '7778889999', 'artem@example.com'),

(8, 'Здравник', 'Ольга Новикова', 'ул. Цветочная, 8, Город, Страна', '6665554444', 'olga@example.com'),

(9, 'Фарма онлайн', 'Владимир Петров', 'пр-т Гагарина, 9, Город, Страна', '2223334444', 'vladimir@example.com'),

(10, 'Фарммаркет', 'Екатерина Семенова', 'ул. Парковая, 10, Город, Страна', '3334445555', 'ekaterina@example.com');

INSERT INTO medicines (medicine\_id, medicine\_name, manufacturer, price, quantity\_in\_stock) VALUES

(1, 'Парацетамол', 'ФармаПром', 50, 100),

(2, 'Ибупрофен', 'МедФарм', 10, 200),

(3, 'Аспирин', 'ФармКорп', 15, 150),

(4, 'Цитрамон', 'ЗдравМед', 5, 300),

(5, 'Нурофен', 'Медика', 80, 50),

(6, 'Лоратадин', 'АллергФарм', 20, 100),

(7, 'Амоксициллин', 'АнтибиотикПлюс', 100, 75),

(8, 'Азитромицин', 'АнтиФарм', 150, 40),

(9, 'Цефтриаксон', 'Фармакор', 120, 30),

(10, 'Метформин', 'ДиабетПром', 200, 25);

INSERT INTO customers (customer\_id, first\_name, last\_name, address, phone\_number, email) VALUES

(1, 'Иван', 'Петров', 'ул. Парковая, 1, Город, Страна', '1112223333', 'ivan@example.com'),

(2, 'Марина', 'Сидорова', 'пр-т Лесной, 2, Город, Страна', '4445556666', 'marina@example.com'),

(3, 'Александр', 'Иванов', 'ул. Центральная, 3, Город, Страна', '7778889999', 'alexander@example.com'),

(4, 'Ольга', 'Кузнецова', 'пр. Солнечный, 4, Город, Страна', '2223334444', 'olga@example.com'),

(5, 'Виктор', 'Смирнов', 'ул. Гагарина, 5, Город, Страна', '5556667777', 'victor@example.com'),

(6, 'Татьяна', 'Морозова', 'пер. Луговой, 6, Город, Страна', '8889990000', 'tatiana@example.com'),

(7, 'Игорь', 'Волков', 'ул. Зеленая, 7, Город, Страна', '3334445555', 'igor@example.com'),

(8, 'Елена', 'Козлова', 'пр-т Цветочный, 8, Город, Страна', '6667778888', 'elena@example.com'),

(9, 'Николай', 'Макаров', 'ул. Московская, 9, Город, Страна', '9990001111', 'nikolay@example.com'),

(10,'Анна', 'Лебедева', 'пр-т Ленина, 10, Город, Страна', '1234567890', 'anna@example.com');

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_status) VALUES

(1, 1, '2024-03-21', 'В обработке'),

(2, 2, '2024-03-20', 'Выполнен'),

(3, 3, '2024-03-19', 'В обработке'),

(4, 4, '2024-03-18', 'Выполнен'),

(5, 5, '2024-03-17', 'В обработке'),

(6, 6, '2024-03-16', 'Выполнен'),

(7, 7, '2024-03-15', 'В обработке'),

(8, 8, '2024-03-14', 'Выполнен'),

(9, 9, '2024-03-13', 'В обработке'),

(10, 10, '2024-03-12', 'Выполнен');

INSERT INTO order\_details (detail\_id, order\_id, medicine\_id, quantity) VALUES

(1, 1, 1, 2),

(2, 1, 2, 1),

(3, 2, 3, 4),

(4, 2, 4, 2),

(5, 3, 5, 1),

(6, 3, 6, 3),

(7, 4, 7, 2),

(8, 4, 8, 1),

(9, 5, 9, 3),

(10, 5, 10, 2);

INSERT INTO payments (payment\_id, order\_id, customer\_id, amount\_paid, payment\_date, payment\_method) VALUES

(1, 1, 1, 150, '2024-03-21', 'Наличные'),

(2, 2, 2, 130, '2024-03-20', 'Карта'),

(3, 3, 3, 115, '2024-03-19', 'Наличные'),

(4, 4, 4, 220, '2024-03-18', 'Карта'),

(5, 5, 5, 190, '2024-03-17', 'Наличные'),

(6, 6, 6, 250, '2024-03-16', 'Карта'),

(7, 7, 7, 90, '2024-03-15', 'Наличные'),

(8, 8, 8, 180, '2024-03-14', 'Карта'),

(9, 9, 9, 120, '2024-03-13', 'Наличные'),

(10, 10, 10, 200, '2024-03-12', 'Карта');

INSERT INTO employees (employee\_id, first\_name, last\_name, position, salary) VALUES

(1, 'Алексей', 'Иванов', 'Менеджер', 50000),

(2, 'Елена', 'Петрова', 'Ассистент менеджера', 35000),

(3, 'Иван', 'Смирнов', 'Супервайзер склада', 40000),

(4, 'Татьяна', 'Кузнецова', 'Кладовщик', 30000),

(5, 'Андрей', 'Морозов', 'Бухгалтер', 45000),

(6, 'Елена', 'Новикова', 'Водитель', 32000),

(7, 'Михаил', 'Васильев', 'Менеджер по обслуживанию клиентов', 35000),

(8, 'Анастасия', 'Тимофеева', 'Менеджер по продажам', 48000),

(9, 'Сергей', 'Козлов', 'Агент по закупкам', 42000),

(10, 'Марина', 'Павлова', 'Маркетолог', 38000);

INSERT INTO deliveries (delivery\_id, supplier\_id, medicine\_id, delivery\_date, quantity\_delivered, delivery\_cost) VALUES

(1, 1, 3, '2024-03-21', 50, 1000),

(2, 2, 1, '2024-03-22', 30, 2000),

(3, 3, 5, '2024-03-23', 20, 3000),

(4, 4, 2, '2024-03-24', 40, 4000),

(5, 5, 4, '2024-03-25', 25, 5000),

(6, 6, 3, '2024-03-26', 35, 6000),

(7, 7, 1, '2024-03-27', 45, 7000),

(8, 8, 5, '2024-03-28', 15, 8000),

(9, 9, 2, '2024-03-29', 55, 9000),

(10, 10, 4, '2024-03-30', 10, 10000);

INSERT INTO employee\_orders (employee\_order\_id, employee\_id, order\_id) VALUES

(1, 1, 1),

(2, 2, 2),

(3, 3, 3),

(4, 4, 4),

(5, 5, 5),

(6, 6, 6),

(7, 7, 7),

(8, 8, 8),

(9, 9, 9),

(10, 10, 10);

**Приложение В**

Файл create\_db.py

def create\_tables():  
 # Определяем путь к базе данных  
 db\_path = os.path.join(os.path.dirname(\_\_file\_\_), '../pharmacy.db')  
  
 # Проверяем существование каталога, если нет, создаем его  
 if not os.path.exists(os.path.dirname(db\_path)):  
 os.makedirs(os.path.dirname(db\_path))  
  
 connection = sqlite3.connect(db\_path)  
 cursor = connection.cursor()  
  
 cursor.executescript('''  
 CREATE TABLE IF NOT EXISTS posts (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 name VARCHAR(100)  
 );  
  
 CREATE TABLE IF NOT EXISTS users (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 login VARCHAR(100),  
 password VARCHAR(100),  
 post\_id INTEGER,  
 FOREIGN KEY (post\_id) REFERENCES posts(id)  
 );  
  
 CREATE TABLE IF NOT EXISTS suppliers (  
 supplier\_id INTEGER PRIMARY KEY,  
 company\_name VARCHAR(255),  
 contact\_person VARCHAR(255),  
 address VARCHAR(255),  
 phone\_number VARCHAR(15),  
 email VARCHAR(255)  
 );  
  
 CREATE TABLE IF NOT EXISTS medicines (  
 medicine\_id INTEGER PRIMARY KEY,  
 medicine\_name VARCHAR(255),  
 manufacturer VARCHAR(255),  
 price INTEGER,  
 quantity\_in\_stock INTEGER  
 );  
  
 CREATE TABLE IF NOT EXISTS employees (  
 employee\_id INTEGER PRIMARY KEY,  
 first\_name VARCHAR(255),  
 last\_name VARCHAR(255),  
 position VARCHAR(100),  
 salary INTEGER  
 );  
  
 CREATE TABLE IF NOT EXISTS customers (  
 customer\_id INTEGER PRIMARY KEY,  
 first\_name VARCHAR(255),  
 last\_name VARCHAR(255),  
 address VARCHAR(255),  
 phone\_number VARCHAR(15),  
 email VARCHAR(255)  
 );  
  
 CREATE TABLE IF NOT EXISTS orders (  
 order\_id INTEGER PRIMARY KEY,  
 customer\_id INTEGER,  
 order\_date DATE,  
 order\_status VARCHAR(50),  
 FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)  
 );  
  
 CREATE TABLE IF NOT EXISTS order\_details (  
 detail\_id INTEGER PRIMARY KEY,  
 order\_id INTEGER,  
 medicine\_id INTEGER,  
 quantity INTEGER,  
 FOREIGN KEY (order\_id) REFERENCES orders(order\_id),  
 FOREIGN KEY (medicine\_id) REFERENCES medicines(medicine\_id)  
 );  
  
 CREATE TABLE IF NOT EXISTS employee\_orders (  
 employee\_order\_id INTEGER PRIMARY KEY,  
 employee\_id INTEGER,  
 order\_id INTEGER,  
 FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id),  
 FOREIGN KEY (order\_id) REFERENCES orders(order\_id)  
 );  
  
 CREATE TABLE IF NOT EXISTS deliveries (  
 delivery\_id INTEGER PRIMARY KEY,  
 supplier\_id INTEGER,  
 medicine\_id INTEGER,  
 delivery\_date DATE,  
 quantity\_delivered INTEGER,  
 delivery\_cost INTEGER,  
 FOREIGN KEY (supplier\_id) REFERENCES suppliers(supplier\_id),  
 FOREIGN KEY (medicine\_id) REFERENCES medicines(medicine\_id)  
 );  
  
 CREATE TABLE IF NOT EXISTS payments (  
 payment\_id INTEGER PRIMARY KEY,  
 customer\_id INTEGER,  
 order\_id INTEGER,  
 amount\_paid INTEGER,  
 payment\_date DATE,  
 payment\_method VARCHAR(50),  
 FOREIGN KEY (order\_id) REFERENCES orders(order\_id),  
 FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)  
 );  
  
 CREATE TABLE IF NOT EXISTS cart (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 user\_id INTEGER,  
 medicine\_id INTEGER,  
 quantity INTEGER,  
 FOREIGN KEY (user\_id) REFERENCES users(id),  
 FOREIGN KEY (medicine\_id) REFERENCES medicines(medicine\_id)  
 );  
 ''')  
  
 cursor.executescript('''  
 INSERT INTO posts (name) VALUES  
 ('Администратор'), ('Пользователь'), ('Сотрудник');  
  
 INSERT INTO users (login, password, post\_id) VALUES  
 ('admin', 'admin', 1),   
 ('user1', 'user', 2), ('user2', 'user', 2), ('user3', 'user', 2), ('user4', 'user', 2), ('user5', 'user', 2),  
 ('employee', 'employee', 3);  
  
 INSERT INTO suppliers (supplier\_id, company\_name, contact\_person, address, phone\_number, email) VALUES  
 (1, 'Фармацевтика АО', 'Иван Иванов', 'ул. Примерная, 1, Город, Страна', '1234567890', 'ivan@example.com'),  
 (2, 'Медикал Трейд', 'Мария Смирнова', 'пр. Центральный, 2, Город, Страна', '0987654321', 'maria@example.com'),  
 (3, 'Здоровье для всех', 'Алексей Кузнецов', 'пр-т Солнечный, 3, Город, Страна', '1112223333', 'alex@example.com'),  
 (4, 'Медснаб', 'Елена Попова', 'ул. Московская, 4, Город, Страна', '5556667777', 'elena@example.com'),  
 (5, 'Фармкомплект', 'Дмитрий Иванов', 'пр-т Ленина, 5, Город, Страна', '9998887777', 'dmitry@example.com'),  
 (6, 'Аптека РУ', 'Анна Соколова', 'ул. Зеленая, 6, Город, Страна', '4445556666', 'anna@example.com'),  
 (7, 'Медсервис', 'Артем Васильев', 'пер. Луговой, 7, Город, Страна', '7778889999', 'artem@example.com'),  
 (8, 'Здравник', 'Ольга Новикова', 'ул. Цветочная, 8, Город, Страна', '6665554444', 'olga@example.com'),  
 (9, 'Фарма онлайн', 'Владимир Петров', 'пр-т Гагарина, 9, Город, Страна', '2223334444', 'vladimir@example.com'),  
 (10, 'Фарммаркет', 'Екатерина Семенова', 'ул. Парковая, 10, Город, Страна', '3334445555', 'ekaterina@example.com');  
  
 INSERT INTO medicines (medicine\_id, medicine\_name, manufacturer, price, quantity\_in\_stock) VALUES  
 (1, 'Парацетамол', 'ФармаПром', 50, 100),  
 (2, 'Ибупрофен', 'МедФарм', 10, 200),  
 (3, 'Аспирин', 'ФармКорп', 15, 150),  
 (4, 'Цитрамон', 'ЗдравМед', 5, 300),  
 (5, 'Нурофен', 'Медика', 80, 50),  
 (6, 'Лоратадин', 'АллергФарм', 20, 100),  
 (7, 'Амоксициллин', 'АнтибиотикПлюс', 100, 75),  
 (8, 'Азитромицин', 'АнтиФарм', 150, 40),  
 (9, 'Цефтриаксон', 'Фармакор', 120, 30),  
 (10, 'Метформин', 'ДиабетПром', 200, 25);  
  
 INSERT INTO customers (customer\_id, first\_name, last\_name, address, phone\_number, email) VALUES  
 (1, 'Иван', 'Петров', 'ул. Парковая, 1, Город, Страна', '1112223333', 'ivan@example.com'),  
 (2, 'Марина', 'Сидорова', 'пр-т Лесной, 2, Город, Страна', '4445556666', 'marina@example.com'),  
 (3, 'Александр', 'Иванов', 'ул. Центральная, 3, Город, Страна', '7778889999', 'alexander@example.com'),  
 (4, 'Ольга', 'Кузнецова', 'пр. Солнечный, 4, Город, Страна', '2223334444', 'olga@example.com'),  
 (5, 'Виктор', 'Смирнов', 'ул. Гагарина, 5, Город, Страна', '5556667777', 'victor@example.com'),  
 (6, 'Татьяна', 'Морозова', 'пер. Луговой, 6, Город, Страна', '8889990000', 'tatiana@example.com'),  
 (7, 'Игорь', 'Волков', 'ул. Зеленая, 7, Город, Страна', '3334445555', 'igor@example.com'),  
 (8, 'Елена', 'Козлова', 'пр-т Цветочный, 8, Город, Страна', '6667778888', 'elena@example.com'),  
 (9, 'Николай', 'Макаров', 'ул. Московская, 9, Город, Страна', '9990001111', 'nikolay@example.com'),  
 (10,'Анна', 'Лебедева', 'пр-т Ленина, 10, Город, Страна', '1234567890', 'anna@example.com');  
  
 INSERT INTO orders (order\_id, customer\_id, order\_date, order\_status) VALUES  
 (1, 1, '2024-03-21', 'В обработке'),  
 (2, 2, '2024-03-20', 'Выполнен'),  
 (3, 3, '2024-03-19', 'В обработке'),  
 (4, 4, '2024-03-18', 'Выполнен'),  
 (5, 5, '2024-03-17', 'В обработке'),  
 (6, 6, '2024-03-16', 'Выполнен'),  
 (7, 7, '2024-03-15', 'В обработке'),  
 (8, 8, '2024-03-14', 'Выполнен'),  
 (9, 9, '2024-03-13', 'В обработке'),  
 (10, 10, '2024-03-12', 'Выполнен');  
  
 INSERT INTO order\_details (detail\_id, order\_id, medicine\_id, quantity) VALUES  
 (1, 1, 1, 2),  
 (2, 1, 2, 1),  
 (3, 2, 3, 4),  
 (4, 2, 4, 2),  
 (5, 3, 5, 1),  
 (6, 3, 6, 3),  
 (7, 4, 7, 2),  
 (8, 4, 8, 1),  
 (9, 5, 9, 3),  
 (10, 5, 10, 2);  
  
 INSERT INTO payments (payment\_id, order\_id, customer\_id, amount\_paid, payment\_date, payment\_method) VALUES  
 (1, 1, 1, 150, '2024-03-21', 'Наличные'),  
 (2, 2, 2, 130, '2024-03-20', 'Карта'),  
 (3, 3, 3, 115, '2024-03-19', 'Наличные'),  
 (4, 4, 4, 220, '2024-03-18', 'Карта'),  
 (5, 5, 5, 190, '2024-03-17', 'Наличные'),  
 (6, 6, 6, 250, '2024-03-16', 'Карта'),  
 (7, 7, 7, 90, '2024-03-15', 'Наличные'),  
 (8, 8, 8, 180, '2024-03-14', 'Карта'),  
 (9, 9, 9, 120, '2024-03-13', 'Наличные'),  
 (10, 10, 10, 200, '2024-03-12', 'Карта');  
  
 INSERT INTO employees (employee\_id, first\_name, last\_name, position, salary) VALUES  
 (1, 'Алексей', 'Иванов', 'Менеджер', 50000),  
 (2, 'Елена', 'Петрова', 'Ассистент менеджера', 35000),  
 (3, 'Иван', 'Смирнов', 'Супервайзер склада', 40000),  
 (4, 'Татьяна', 'Кузнецова', 'Кладовщик', 30000),  
 (5, 'Андрей', 'Морозов', 'Бухгалтер', 45000),  
 (6, 'Елена', 'Новикова', 'Водитель', 32000),  
 (7, 'Михаил', 'Васильев', 'Менеджер по обслуживанию клиентов', 35000),  
 (8, 'Анастасия', 'Тимофеева', 'Менеджер по продажам', 48000),  
 (9, 'Сергей', 'Козлов', 'Агент по закупкам', 42000),  
 (10, 'Марина', 'Павлова', 'Маркетолог', 38000);  
  
 INSERT INTO deliveries (delivery\_id, supplier\_id, medicine\_id, delivery\_date, quantity\_delivered, delivery\_cost) VALUES  
 (1, 1, 3, '2024-03-21', 50, 1000),  
 (2, 2, 1, '2024-03-22', 30, 2000),  
 (3, 3, 5, '2024-03-23', 20, 3000),  
 (4, 4, 2, '2024-03-24', 40, 4000),  
 (5, 5, 4, '2024-03-25', 25, 5000),  
 (6, 6, 3, '2024-03-26', 35, 6000),  
 (7, 7, 1, '2024-03-27', 45, 7000),  
 (8, 8, 5, '2024-03-28', 15, 8000),  
 (9, 9, 2, '2024-03-29', 55, 9000),  
 (10, 10, 4, '2024-03-30', 10, 10000);  
  
 INSERT INTO employee\_orders (employee\_order\_id, employee\_id, order\_id) VALUES  
 (1, 1, 1),  
 (2, 2, 2),  
 (3, 3, 3),  
 (4, 4, 4),  
 (5, 5, 5),  
 (6, 6, 6),  
 (7, 7, 7),  
 (8, 8, 8),  
 (9, 9, 9),  
 (10, 10, 10);  
 ''')  
  
 connection.commit()  
  
 # Проверка данных в таблице users  
 cursor.execute('SELECT \* FROM users')  
 users = cursor.fetchall()  
 print(f"DEBUG: Users in the database: {users}")  
  
 connection.close()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 create\_tables()

Файл admin\_window.py

from PyQt6.QtWidgets import QMainWindow, QVBoxLayout, QLabel, QPushButton, QWidget, QTableWidget, QTableWidgetItem, QMessageBox

from PyQt6.QtGui import QFont

from PyQt6.QtCore import Qt

from utils.db\_utils import fetch\_all, execute\_query

class AdminWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle('Администратор')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.central\_widget.setStyleSheet("background-color: #E6E6FA;")

layout = QVBoxLayout()

self.label\_title = QLabel('Панель администратора')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

layout.addWidget(self.label\_title)

self.button\_manage\_users = self.create\_button('Управление пользователями')

self.button\_manage\_users.clicked.connect(self.manage\_users)

layout.addWidget(self.button\_manage\_users)

self.button\_view\_reports = self.create\_button('Просмотр отчетов')

self.button\_view\_reports.clicked.connect(self.view\_reports)

layout.addWidget(self.button\_view\_reports)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def manage\_users(self):

from gui.admin\_window import UsersWindow # Импорт здесь, чтобы избежать циклического импорта

self.users\_window = UsersWindow()

self.users\_window.show()

def view\_reports(self):

reports = fetch\_all("SELECT \* FROM reports")

from gui.admin\_window import ReportsWindow # Импорт здесь, чтобы избежать циклического импорта

self.reports\_window = ReportsWindow(reports)

self.reports\_window.show()

def back(self):

self.close()

class UsersWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle('Управление пользователями')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.central\_widget.setStyleSheet("background-color: #E6E6FA;")

layout = QVBoxLayout()

self.label\_title = QLabel('Пользователи')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

layout.addWidget(self.label\_title)

self.table = QTableWidget()

layout.addWidget(self.table)

self.button\_add = self.create\_button('Добавить пользователя')

self.button\_add.clicked.connect(self.add\_user)

layout.addWidget(self.button\_add)

self.button\_delete = self.create\_button('Удалить пользователя')

self.button\_delete.clicked.connect(self.delete\_user)

layout.addWidget(self.button\_delete)

self.button\_update = self.create\_button('Изменить пользователя')

self.button\_update.clicked.connect(self.update\_user)

layout.addWidget(self.button\_update)

self.button\_load = self.create\_button('Загрузить пользователей')

self.button\_load.clicked.connect(self.load\_users)

layout.addWidget(self.button\_load)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

self.load\_users()

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def load\_users(self):

users = fetch\_all("SELECT \* FROM users")

self.table.setRowCount(len(users))

self.table.setColumnCount(4)

self.table.setHorizontalHeaderLabels(['ID', 'Логин', 'Пароль', 'Роль'])

for row\_index, row\_data in enumerate(users):

for col\_index, col\_data in enumerate(row\_data):

self.table.setItem(row\_index, col\_index, QTableWidgetItem(str(col\_data)))

def add\_user(self):

# Логика добавления пользователя

pass

def delete\_user(self):

row = self.table.currentRow()

if row == -1:

QMessageBox.warning(self, 'Ошибка', 'Выберите строку для удаления')

return

user\_id = self.table.item(row, 0).text()

execute\_query("DELETE FROM users WHERE id = ?", (user\_id,))

QMessageBox.information(self, 'Успех', 'Пользователь удален')

self.load\_users()

def update\_user(self):

row = self.table.currentRow()

if row == -1:

QMessageBox.warning(self, 'Ошибка', 'Выберите строку для изменения')

return

user\_id = self.table.item(row, 0).text()

# Логика изменения данных пользователя

pass

def back(self):

self.close()

class ReportsWindow(QMainWindow):

def \_\_init\_\_(self, reports):

super().\_\_init\_\_()

self.setWindowTitle('Просмотр отчетов')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.central\_widget.setStyleSheet("background-color: #E6E6FA;")

layout = QVBoxLayout()

self.label\_title = QLabel('Отчеты')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

layout.addWidget(self.label\_title)

self.table = QTableWidget()

self.table.setColumnCount(3)

self.table.setHorizontalHeaderLabels(['ID', 'Название', 'Дата'])

self.table.setRowCount(len(reports))

for row\_index, row\_data in enumerate(reports):

for col\_index, col\_data in enumerate(row\_data):

self.table.setItem(row\_index, col\_index, QTableWidgetItem(str(col\_data)))

layout.addWidget(self.table)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def back(self):

self.close()

Файл employee\_window.py

from PyQt6.QtGui import QFont, QPixmap, QPalette, QBrush

from PyQt6.QtWidgets import QMainWindow, QVBoxLayout, QLabel, QPushButton, QWidget, QTableWidget, QTableWidgetItem, \

QLineEdit, QMessageBox, QInputDialog

from PyQt6.QtCore import Qt

from utils.db\_utils import execute\_query, fetch\_all

class EmployeeWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle('Сотрудник')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.label\_title = QLabel('Онлайн аптека')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

layout.addWidget(self.label\_title)

self.button\_manage\_medicines = self.create\_button('Управление товарами')

self.button\_manage\_medicines.clicked.connect(self.manage\_medicines)

layout.addWidget(self.button\_manage\_medicines)

self.button\_view\_order\_quantity = self.create\_button('Просмотр количества заказов')

self.button\_view\_order\_quantity.clicked.connect(self.view\_order\_quantity)

layout.addWidget(self.button\_view\_order\_quantity)

self.button\_view\_order\_info = self.create\_button('Просмотр информации заказов')

self.button\_view\_order\_info.clicked.connect(self.view\_order\_info)

layout.addWidget(self.button\_view\_order\_info)

self.button\_view\_medicines\_quantity = self.create\_button('Просмотр количества товаров')

self.button\_view\_medicines\_quantity.clicked.connect(self.view\_medicines\_quantity)

layout.addWidget(self.button\_view\_medicines\_quantity)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def manage\_medicines(self):

self.medicines\_window = ManageMedicinesWindow()

self.medicines\_window.show()

def view\_order\_quantity(self):

orders = fetch\_all("SELECT COUNT(\*) FROM orders")

QMessageBox.information(self, 'Количество заказов', f"Количество заказов: {orders[0][0]}")

def view\_order\_info(self):

orders = fetch\_all("SELECT \* FROM orders")

self.order\_info\_window = OrderInfoWindow(orders)

self.order\_info\_window.show()

def view\_medicines\_quantity(self):

medicines\_quantity = fetch\_all("SELECT medicine\_name, quantity\_in\_stock FROM medicines")

self.medicines\_quantity\_window = MedicinesQuantityWindow(medicines\_quantity)

self.medicines\_quantity\_window.show()

def back(self):

self.close()

class ManageMedicinesWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle('Управление товарами')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.label\_title = QLabel('Управление товарами')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

layout.addWidget(self.label\_title)

self.table = QTableWidget()

layout.addWidget(self.table)

self.button\_add = self.create\_button('Добавить товар')

self.button\_add.clicked.connect(self.add\_medicine)

layout.addWidget(self.button\_add)

self.button\_delete = self.create\_button('Удалить товар')

self.button\_delete.clicked.connect(self.delete\_medicine)

layout.addWidget(self.button\_delete)

self.button\_update = self.create\_button('Изменить товар')

self.button\_update.clicked.connect(self.update\_medicine)

layout.addWidget(self.button\_update)

self.button\_load = self.create\_button('Загрузить товары')

self.button\_load.clicked.connect(self.load\_medicines)

layout.addWidget(self.button\_load)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

self.load\_medicines()

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def load\_medicines(self):

medicines = fetch\_all("SELECT \* FROM medicines")

self.table.setRowCount(len(medicines))

self.table.setColumnCount(5)

self.table.setHorizontalHeaderLabels(

['ID', 'Название', 'Производитель', 'Цена', 'Количество на складе'])

for row\_index, row\_data in enumerate(medicines):

for col\_index, col\_data in enumerate(row\_data):

self.table.setItem(row\_index, col\_index, QTableWidgetItem(str(col\_data)))

def add\_medicine(self):

try:

medicine\_name, ok = QInputDialog.getText(self, 'Название товара', 'Введите название:')

if not ok or not medicine\_name:

return

manufacturer, ok = QInputDialog.getText(self, 'Производитель', 'Введите производителя:')

if not ok or not manufacturer:

return

price, ok = QInputDialog.getInt(self, 'Цена', 'Введите цену:')

if not ok or not price:

return

quantity\_in\_stock, ok = QInputDialog.getInt(self, 'Количество на складе', 'Введите количество на складе:')

if not ok or not quantity\_in\_stock:

return

execute\_query(

"INSERT INTO medicines (medicine\_name, manufacturer, price, quantity\_in\_stock) VALUES (?, ?, ?, ?)",

(medicine\_name, manufacturer, price, quantity\_in\_stock))

QMessageBox.information(self, 'Успех', 'Товар добавлен')

self.load\_medicines()

except Exception as e:

QMessageBox.critical(self, 'Ошибка', f'Ошибка добавления товара: {str(e)}')

def delete\_medicine(self):

row = self.table.currentRow()

if row == -1:

QMessageBox.warning(self, 'Ошибка', 'Выберите строку для удаления')

return

try:

medicine\_id = self.table.item(row, 0).text()

execute\_query("DELETE FROM medicines WHERE medicine\_id = ?", (medicine\_id,))

QMessageBox.information(self, 'Успех', 'Товар удален')

self.load\_medicines()

except Exception as e:

QMessageBox.critical(self, 'Ошибка', f'Ошибка удаления товара: {str(e)}')

def update\_medicine(self):

row = self.table.currentRow()

if row == -1:

QMessageBox.warning(self, 'Ошибка', 'Выберите строку для изменения')

return

try:

medicine\_id = self.table.item(row, 0).text()

medicine\_name, ok = QInputDialog.getText(self, 'Название товара', 'Введите название:',

text=self.table.item(row, 1).text())

if not ok or not medicine\_name:

return

manufacturer, ok = QInputDialog.getText(self, 'Производитель', 'Введите производителя:',

text=self.table.item(row, 2).text())

if not ok or not manufacturer:

return

price, ok = QInputDialog.getInt(self, 'Цена', 'Введите цену:', value=int(self.table.item(row, 3).text()))

if not ok or not price:

return

quantity\_in\_stock, ok = QInputDialog.getInt(self, 'Количество на складе', 'Введите количество на складе:',

value=int(self.table.item(row, 4).text()))

if not ok or not quantity\_in\_stock:

return

execute\_query(

"UPDATE medicines SET medicine\_name = ?, manufacturer = ?, price = ?, quantity\_in\_stock = ? WHERE medicine\_id = ?",

(medicine\_name, manufacturer, price, quantity\_in\_stock, medicine\_id))

QMessageBox.information(self, 'Успех', 'Товар изменен')

self.load\_medicines()

except Exception as e:

QMessageBox.critical(self, 'Ошибка', f'Ошибка изменения товара: {str(e)}')

def back(self):

self.close()

class OrderInfoWindow(QMainWindow):

def \_\_init\_\_(self, orders):

super().\_\_init\_\_()

self.setWindowTitle('Информация о заказах')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.label\_title = QLabel('Заказы')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

layout.addWidget(self.label\_title)

self.table = QTableWidget()

self.table.setColumnCount(5)

self.table.setHorizontalHeaderLabels(['ID', 'Customer ID', 'Order Date', 'Status', 'Total Price'])

self.table.setRowCount(len(orders))

for row\_index, row\_data in enumerate(orders):

for col\_index, col\_data in enumerate(row\_data):

self.table.setItem(row\_index, col\_index, QTableWidgetItem(str(col\_data)))

layout.addWidget(self.table)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def back(self):

self.close()

class MedicinesQuantityWindow(QMainWindow):

def \_\_init\_\_(self, medicines\_quantity):

super().\_\_init\_\_()

self.setWindowTitle('Количество товаров')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.label\_title = QLabel('Количество товаров на складе')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

layout.addWidget(self.label\_title)

self.table = QTableWidget()

self.table.setColumnCount(2)

self.table.setHorizontalHeaderLabels(['Название', 'Количество'])

self.table.setRowCount(len(medicines\_quantity))

for row\_index, row\_data in enumerate(medicines\_quantity):

for col\_index, col\_data in enumerate(row\_data):

self.table.setItem(row\_index, col\_index, QTableWidgetItem(str(col\_data)))

layout.addWidget(self.table)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def back(self):

self.close()

Файл login\_window.py

import sys  
from PyQt6.QtWidgets import (  
 QApplication, QMainWindow, QVBoxLayout, QLabel, QLineEdit, QPushButton, QWidget, QMessageBox  
)  
from PyQt6.QtGui import QFont, QPixmap, QPalette, QBrush  
from PyQt6.QtCore import Qt  
from utils.db\_utils import fetch\_all, fetch\_one  
import os  
  
  
class LoginWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 # Load background image  
 self.pixmap = QPixmap("img/pharmacy\_background.jpg")  
  
 # Set fixed size based on background image  
 self.setFixedSize(self.pixmap.size())  
  
 self.setWindowTitle('Авторизация - Онлайн аптека')  
  
 self.central\_widget = QWidget()  
 self.setCentralWidget(self.central\_widget)  
  
 self.set\_background()  
  
 layout = QVBoxLayout()  
 layout.setContentsMargins(30, 30, 30, 30)  
 layout.setSpacing(20)  
  
 self.label\_company = QLabel('Онлайн аптека')  
 self.label\_company.setFont(QFont('Times New Romans', 24, QFont.Weight.Bold))  
 self.label\_company.setAlignment(Qt.AlignmentFlag.AlignCenter)  
 self.label\_company.setStyleSheet("color: red;")  
 layout.addWidget(self.label\_company)  
  
 self.label\_title = QLabel('Вход в систему')  
 self.label\_title.setFont(QFont('Times New Romans', 20))  
 self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)  
 self.label\_title.setStyleSheet("color: red;")  
 layout.addWidget(self.label\_title)  
  
 self.input\_login = QLineEdit()  
 self.input\_login.setPlaceholderText('Логин')  
 self.input\_login.setFont(QFont('Arial', 14))  
 self.input\_login.setStyleSheet(  
 "background-color: #9370DB; color: white; padding: 10px; border-radius: 15px;"  
 )  
 layout.addWidget(self.input\_login)  
  
 self.input\_password = QLineEdit()  
 self.input\_password.setPlaceholderText('Пароль')  
 self.input\_password.setFont(QFont('Arial', 14))  
 self.input\_password.setEchoMode(QLineEdit.EchoMode.Password)  
 self.input\_password.setStyleSheet(  
 "background-color: #9370DB; color: white; padding: 10px; border-radius: 15px;"  
 )  
 layout.addWidget(self.input\_password)  
  
 self.button\_login = QPushButton('Войти')  
 self.button\_login.setFont(QFont('Arial', 16))  
 self.button\_login.setStyleSheet(  
 "background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"  
 )  
 self.button\_login.clicked.connect(self.check\_credentials)  
 layout.addWidget(self.button\_login)  
  
 self.central\_widget.setLayout(layout)  
  
 def set\_background(self):  
 palette = QPalette()  
 palette.setBrush(QPalette.ColorRole.Window, QBrush(self.pixmap))  
 self.setPalette(palette)  
  
 def check\_credentials(self):  
 login = self.input\_login.text().strip()  
 password = self.input\_password.text().strip()  
  
 print(f"DEBUG: Input login: {login}, Input password: {password}") # Отладочная информация  
  
 try:  
 # Дополнительная отладочная информация  
 print("DEBUG: Connecting to database...")  
  
 # Проверка всех записей в таблице users  
 all\_users = fetch\_all("SELECT id, login, password, post\_id FROM users")  
 print(f"DEBUG: All users: {all\_users}") # Отладочная информация  
  
 # Выполнение запроса для конкретного пользователя  
 results = fetch\_all("SELECT id, post\_id, password FROM users WHERE login=?", (login,))  
  
 print("DEBUG: SQL query executed.")  
 print(f"DEBUG: Results of fetch\_all: {results}") # Отладочная информация  
 except Exception as e:  
 QMessageBox.critical(self, 'Ошибка', f'Ошибка подключения к базе данных: {str(e)}')  
 print(f"DEBUG: Database connection error: {e}") # Отладочная информация  
 return  
  
 if results:  
 # Выберите первую запись из результатов  
 result = results[0]  
 user\_id, post\_id, stored\_password = result  
 print(  
 f"DEBUG: Retrieved user\_id={user\_id}, post\_id={post\_id}, stored\_password={stored\_password}") # Отладочная информация  
 if password == stored\_password: # Простое сравнение строк для незашифрованных паролей  
 if post\_id == 1:  
 from gui.admin\_window import AdminWindow # Импорт здесь, чтобы избежать циклического импорта  
 self.admin\_window = AdminWindow()  
 self.admin\_window.show()  
 elif post\_id == 2:  
 from gui.user\_window import UserWindow # Импорт здесь, чтобы избежать циклического импорта  
 self.user\_window = UserWindow(user\_id)  
 self.user\_window.show()  
 elif post\_id == 3:  
 from gui.employee\_window import EmployeeWindow # Импорт здесь, чтобы избежать циклического импорта  
 self.employee\_window = EmployeeWindow()  
 self.employee\_window.show()  
 self.hide()  
 else:  
 print("DEBUG: Password mismatch") # Отладочная информация  
 QMessageBox.warning(self, 'Ошибка', 'Неверный логин или пароль')  
 else:  
 print("DEBUG: No user found") # Отладочная информация  
 QMessageBox.warning(self, 'Ошибка', 'Неверный логин или пароль')  
  
 def back\_to\_login(self):  
 self.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 login\_window = LoginWindow()  
 login\_window.show()  
 sys.exit(app.exec())

Файл user\_window.py

from PyQt6.QtWidgets import (

QMainWindow, QVBoxLayout, QLabel, QPushButton, QWidget, QTableWidget, QTableWidgetItem,

QLineEdit, QMessageBox, QInputDialog, QHBoxLayout, QGridLayout

)

from PyQt6.QtGui import QFont, QPalette, QPixmap, QBrush

from PyQt6.QtCore import Qt

from utils.db\_utils import execute\_query, fetch\_all, fetch\_one

class UserWindow(QMainWindow):

def \_\_init\_\_(self, user\_id):

super().\_\_init\_\_()

self.user\_id = user\_id

self.setWindowTitle('Пользователь')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QGridLayout()

self.label\_company = QLabel('Онлайн аптека')

self.label\_company.setFont(QFont('Arial', 24, QFont.Weight.Bold))

self.label\_company.setAlignment(Qt.AlignmentFlag.AlignCenter)

self.label\_company.setStyleSheet("color: brown;")

layout.addWidget(self.label\_company, 0, 0, 1, 2)

self.button\_view\_medicines = self.create\_button('Каталог')

self.button\_view\_medicines.clicked.connect(self.view\_medicines)

layout.addWidget(self.button\_view\_medicines, 1, 0)

self.button\_view\_cart = self.create\_button('Корзина')

self.button\_view\_cart.clicked.connect(self.view\_cart)

layout.addWidget(self.button\_view\_cart, 1, 1)

self.button\_view\_orders = self.create\_button('Заказы')

self.button\_view\_orders.clicked.connect(self.view\_orders)

layout.addWidget(self.button\_view\_orders, 2, 0)

self.button\_profile = self.create\_button('Профиль')

self.button\_profile.clicked.connect(self.open\_profile)

layout.addWidget(self.button\_profile, 2, 1)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back, 3, 0, 1, 2)

self.central\_widget.setLayout(layout)

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def view\_medicines(self):

try:

print("DEBUG: Fetching medicines...") # Отладочная информация

medicines = fetch\_all("SELECT \* FROM medicines")

print(f"DEBUG: Fetched medicines: {medicines}") # Отладочная информация

self.medicines\_window = MedicinesWindow(medicines, self.user\_id)

self.medicines\_window.show()

print("DEBUG: MedicinesWindow shown") # Отладочная информация

except Exception as e:

print(f"DEBUG: Error fetching medicines: {e}") # Отладочная информация

QMessageBox.critical(self, 'Ошибка', f'Ошибка получения данных о товарах: {str(e)}')

def view\_cart(self):

self.cart\_window = CartWindow(self.user\_id)

self.cart\_window.show()

def view\_orders(self):

try:

orders = fetch\_all("SELECT \* FROM orders WHERE customer\_id = ?", (self.user\_id,))

self.orders\_window = OrdersWindow(orders)

self.orders\_window.show()

except Exception as e:

print(f"DEBUG: Error fetching orders: {e}") # Отладочная информация

QMessageBox.critical(self, 'Ошибка', f'Ошибка получения данных о заказах: {str(e)}')

def open\_profile(self):

self.profile\_window = ProfileWindow(self.user\_id)

self.profile\_window.show()

def back(self):

from gui.login\_window import LoginWindow

self.login\_window = LoginWindow()

self.login\_window.show()

self.close()

class MedicinesWindow(QMainWindow):

def \_\_init\_\_(self, medicines, user\_id):

super().\_\_init\_\_()

self.user\_id = user\_id

self.setWindowTitle('Просмотр товаров')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.label\_title = QLabel('Товары')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

self.label\_title.setStyleSheet("color: brown;")

layout.addWidget(self.label\_title)

self.table = QTableWidget()

self.table.setColumnCount(6)

self.table.setHorizontalHeaderLabels(

['ID', 'Название', 'Производитель', 'Цена', 'Количество на складе', ''])

self.table.setRowCount(len(medicines))

self.table.verticalHeader().setVisible(False)

self.table.setEditTriggers(QTableWidget.EditTrigger.NoEditTriggers)

self.table.setStyleSheet("background: transparent;")

for row\_index, row\_data in enumerate(medicines):

for col\_index, col\_data in enumerate(row\_data):

self.table.setItem(row\_index, col\_index, QTableWidgetItem(str(col\_data)))

add\_button = QPushButton('Добавить в корзину')

add\_button.clicked.connect(lambda \_, r=row\_index: self.add\_to\_cart(r))

self.table.setCellWidget(row\_index, 5, add\_button)

layout.addWidget(self.table)

buttons\_layout = QHBoxLayout()

self.button\_view\_cart = self.create\_button('Перейти в корзину')

self.button\_view\_cart.clicked.connect(self.view\_cart)

buttons\_layout.addWidget(self.button\_view\_cart)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

buttons\_layout.addWidget(self.button\_back)

layout.addLayout(buttons\_layout)

self.central\_widget.setLayout(layout)

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def add\_to\_cart(self, row):

medicine\_id = self.table.item(row, 0).text()

medicine\_name = self.table.item(row, 1).text()

stock\_quantity = int(self.table.item(row, 4).text())

quantity, ok = QInputDialog.getInt(self, 'Количество', f'Введите количество для "{medicine\_name}":', 1, 1, stock\_quantity)

if not ok or quantity <= 0:

return

try:

execute\_query("INSERT INTO cart (user\_id, medicine\_id, quantity) VALUES (?, ?, ?)", (self.user\_id, medicine\_id, quantity))

QMessageBox.information(self, 'Успех', 'Товар добавлен в корзину')

except Exception as e:

print(f"DEBUG: Error adding to cart: {e}") # Отладочная информация

QMessageBox.critical(self, 'Ошибка', f'Ошибка добавления товара в корзину: {str(e)}')

def view\_cart(self):

self.cart\_window = CartWindow(self.user\_id)

self.cart\_window.show()

def back(self):

self.close()

class CartWindow(QMainWindow):

def \_\_init\_\_(self, user\_id):

super().\_\_init\_\_()

self.user\_id = user\_id

self.setWindowTitle('Корзина')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.label\_title = QLabel('Корзина')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

self.label\_title.setStyleSheet("color: brown;")

layout.addWidget(self.label\_title)

self.table = QTableWidget()

self.table.setColumnCount(4)

self.table.setHorizontalHeaderLabels(['ID', 'Название', 'Количество', 'Цена'])

layout.addWidget(self.table)

self.table.verticalHeader().setVisible(False)

self.table.setEditTriggers(QTableWidget.EditTrigger.NoEditTriggers)

self.table.setStyleSheet("background: transparent;")

self.label\_total = QLabel('Общая сумма: 0')

self.label\_total.setFont(QFont('Arial', 14))

self.label\_total.setStyleSheet("color: brown;")

layout.addWidget(self.label\_total)

self.button\_remove\_from\_cart = self.create\_button('Удалить из корзины')

self.button\_remove\_from\_cart.clicked.connect(self.remove\_from\_cart)

layout.addWidget(self.button\_remove\_from\_cart)

self.button\_create\_order = self.create\_button('Сделать заказ')

self.button\_create\_order.clicked.connect(self.create\_order)

layout.addWidget(self.button\_create\_order)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

self.load\_cart()

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def load\_cart(self):

cart\_items = fetch\_all("""

SELECT cart.id, medicines.medicine\_name, cart.quantity, medicines.price

FROM cart

JOIN medicines ON cart.medicine\_id = medicines.medicine\_id

WHERE cart.user\_id = ?

""", (self.user\_id,))

self.table.setRowCount(len(cart\_items))

total = 0

for row\_index, row\_data in enumerate(cart\_items):

for col\_index, col\_data in enumerate(row\_data):

self.table.setItem(row\_index, col\_index, QTableWidgetItem(str(col\_data)))

if col\_index == 3:

total += row\_data[2] \* row\_data[3]

self.label\_total.setText(f'Общая сумма: {total}')

def remove\_from\_cart(self):

row = self.table.currentRow()

if row == -1:

QMessageBox.warning(self, 'Ошибка', 'Выберите строку для удаления из корзины')

return

cart\_id = self.table.item(row, 0).text()

execute\_query("DELETE FROM cart WHERE id = ?", (cart\_id,))

QMessageBox.information(self, 'Успех', 'Товар удален из корзины')

self.load\_cart()

def create\_order(self):

cart\_items = fetch\_all("SELECT medicine\_id, quantity FROM cart WHERE user\_id = ?", (self.user\_id,))

if not cart\_items:

QMessageBox.warning(self, 'Ошибка', 'Корзина пуста')

return

execute\_query(

"INSERT INTO orders (customer\_id, order\_date, order\_status) VALUES (?, date('now'), 'В обработке')",

(self.user\_id,))

order\_id = fetch\_one("SELECT last\_insert\_rowid()")[0]

for medicine\_id, quantity in cart\_items:

execute\_query("INSERT INTO order\_details (order\_id, medicine\_id, quantity) VALUES (?, ?, ?)",

(order\_id, medicine\_id, quantity))

execute\_query("DELETE FROM cart WHERE user\_id = ?", (self.user\_id,))

QMessageBox.information(self, 'Успех', 'Заказ создан')

self.close()

def back(self):

self.close()

class OrdersWindow(QMainWindow):

def \_\_init\_\_(self, orders):

super().\_\_init\_\_()

self.setWindowTitle('Просмотр заказов')

self.setGeometry(100, 100, 800, 600)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.label\_title = QLabel('Заказы')

self.label\_title.setFont(QFont('Arial', 20))

self.label\_title.setAlignment(Qt.AlignmentFlag.AlignCenter)

self.label\_title.setStyleSheet("color: brown;")

layout.addWidget(self.label\_title)

self.table = QTableWidget()

self.table.setColumnCount(4)

self.table.setHorizontalHeaderLabels(['ID', 'Customer ID', 'Order Date', 'Status'])

self.table.setRowCount(len(orders))

self.table.verticalHeader().setVisible(False)

self.table.setEditTriggers(QTableWidget.EditTrigger.NoEditTriggers)

self.table.setStyleSheet("background: transparent;")

for row\_index, row\_data in enumerate(orders):

for col\_index, col\_data in enumerate(row\_data):

self.table.setItem(row\_index, col\_index, QTableWidgetItem(str(col\_data)))

layout.addWidget(self.table)

self.button\_back = QPushButton('Назад')

self.button\_back.setFont(QFont('Arial', 14))

self.button\_back.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def back(self):

self.close()

class ProfileWindow(QMainWindow):

def \_\_init\_\_(self, user\_id):

super().\_\_init\_\_()

self.user\_id = user\_id

self.setWindowTitle('Профиль')

self.setGeometry(100, 100, 400, 300)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.button\_change\_credentials = self.create\_button('Смена логина или пароля')

self.button\_change\_credentials.clicked.connect(self.change\_credentials)

layout.addWidget(self.button\_change\_credentials)

self.button\_change\_personal\_info = self.create\_button('Смена личных данных')

self.button\_change\_personal\_info.clicked.connect(self.change\_personal\_info)

layout.addWidget(self.button\_change\_personal\_info)

self.button\_view\_orders = self.create\_button('Заказы')

self.button\_view\_orders.clicked.connect(self.view\_orders)

layout.addWidget(self.button\_view\_orders)

self.button\_back = self.create\_button('Назад')

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def create\_button(self, text):

button = QPushButton(text)

button.setFont(QFont('Arial', 14))

button.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

return button

def change\_credentials(self):

self.update\_user\_window = UpdateUserWindow(self.user\_id)

self.update\_user\_window.show()

def change\_personal\_info(self):

self.update\_personal\_info\_window = UpdatePersonalInfoWindow(self.user\_id)

self.update\_personal\_info\_window.show()

def view\_orders(self):

orders = fetch\_all("SELECT \* FROM orders WHERE customer\_id = ?", (self.user\_id,))

self.orders\_window = OrdersWindow(orders)

self.orders\_window.show()

def back(self):

self.close()

class UpdateUserWindow(QMainWindow):

def \_\_init\_\_(self, user\_id):

super().\_\_init\_\_()

self.user\_id = user\_id

self.setWindowTitle('Смена логина или пароля')

self.setGeometry(100, 100, 400, 300)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.label\_login = QLabel('Новый логин:')

layout.addWidget(self.label\_login)

self.input\_login = QLineEdit()

layout.addWidget(self.input\_login)

self.label\_password = QLabel('Новый пароль:')

layout.addWidget(self.label\_password)

self.input\_password = QLineEdit()

layout.addWidget(self.input\_password)

self.button\_update = QPushButton('Обновить')

self.button\_update.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

self.button\_update.clicked.connect(self.update\_user\_in\_db)

layout.addWidget(self.button\_update)

self.button\_back = QPushButton('Назад')

self.button\_back.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def update\_user\_in\_db(self):

login = self.input\_login.text()

password = self.input\_password.text()

try:

execute\_query("UPDATE users SET login=?, password=? WHERE id=?", (login, password, self.user\_id))

QMessageBox.information(self, 'Успех', 'Данные пользователя обновлены')

self.close()

except Exception as e:

print(f"DEBUG: Error updating user: {e}") # Отладочная информация

QMessageBox.critical(self, 'Ошибка', f'Ошибка обновления данных пользователя: {str(e)}')

def back(self):

self.close()

class UpdatePersonalInfoWindow(QMainWindow):

def \_\_init\_\_(self, user\_id):

super().\_\_init\_\_()

self.user\_id = user\_id

self.setWindowTitle('Смена личных данных')

self.setGeometry(100, 100, 400, 400)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.set\_background()

layout = QVBoxLayout()

self.label\_first\_name = QLabel('Имя:')

layout.addWidget(self.label\_first\_name)

self.input\_first\_name = QLineEdit()

layout.addWidget(self.input\_first\_name)

self.label\_last\_name = QLabel('Фамилия:')

layout.addWidget(self.label\_last\_name)

self.input\_last\_name = QLineEdit()

layout.addWidget(self.input\_last\_name)

self.label\_address = QLabel('Адрес:')

layout.addWidget(self.label\_address)

self.input\_address = QLineEdit()

layout.addWidget(self.input\_address)

self.label\_phone = QLabel('Телефон:')

layout.addWidget(self.label\_phone)

self.input\_phone = QLineEdit()

layout.addWidget(self.input\_phone)

self.label\_email = QLabel('Email:')

layout.addWidget(self.label\_email)

self.input\_email = QLineEdit()

layout.addWidget(self.input\_email)

self.button\_update = QPushButton('Обновить')

self.button\_update.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

self.button\_update.clicked.connect(self.update\_personal\_info\_in\_db)

layout.addWidget(self.button\_update)

self.button\_back = QPushButton('Назад')

self.button\_back.setStyleSheet(

"background-color: #9370DB; color: white; border-radius: 15px; padding: 10px;"

)

self.button\_back.clicked.connect(self.back)

layout.addWidget(self.button\_back)

self.central\_widget.setLayout(layout)

self.load\_user\_data()

def set\_background(self):

palette = QPalette()

pixmap = QPixmap("img/pharmacy\_background.jpg")

palette.setBrush(QPalette.ColorRole.Window, QBrush(pixmap))

self.setPalette(palette)

def load\_user\_data(self):

user\_data = fetch\_one("SELECT first\_name, last\_name, address, phone\_number, email FROM customers WHERE customer\_id = ?", (self.user\_id,))

if user\_data:

self.input\_first\_name.setText(user\_data[0])

self.input\_last\_name.setText(user\_data[1])

self.input\_address.setText(user\_data[2])

self.input\_phone.setText(user\_data[3])

self.input\_email.setText(user\_data[4])

def update\_personal\_info\_in\_db(self):

first\_name = self.input\_first\_name.text()

last\_name = self.input\_last\_name.text()

address = self.input\_address.text()

phone = self.input\_phone.text()

email = self.input\_email.text()

try:

execute\_query("UPDATE customers SET first\_name=?, last\_name=?, address=?, phone\_number=?, email=? WHERE customer\_id=?",

(first\_name, last\_name, address, phone, email, self.user\_id))

QMessageBox.information(self, 'Успех', 'Личные данные обновлены')

self.close()

except Exception as e:

print(f"DEBUG: Error updating personal info: {e}") # Отладочная информация

QMessageBox.critical(self, 'Ошибка', f'Ошибка обновления личных данных: {str(e)}')

def back(self):

self.close()

Файл db\_utils.py

import sqlite3  
import os  
  
def get\_db\_path():  
 db\_path = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../pharmacy.db'))  
 print(f"Используемый путь к базе данных: {db\_path}")  
 return db\_path  
  
def fetch\_one(query, params=()):  
 try:  
 db\_path = get\_db\_path()  
 if not os.path.exists(db\_path):  
 raise Exception(f"База данных не найдена по пути: {db\_path}")  
 connection = sqlite3.connect(db\_path)  
 cursor = connection.cursor()  
 cursor.execute(query, params)  
 result = cursor.fetchone()  
 except sqlite3.Error as e:  
 print(f"Ошибка при выполнении запроса: {e}")  
 result = None  
 finally:  
 connection.close()  
 return result  
  
def fetch\_all(query, params=()):  
 try:  
 db\_path = get\_db\_path()  
 if not os.path.exists(db\_path):  
 raise Exception(f"База данных не найдена по пути: {db\_path}")  
 connection = sqlite3.connect(db\_path)  
 cursor = connection.cursor()  
 cursor.execute(query, params)  
 result = cursor.fetchall()  
 except sqlite3.Error as e:  
 print(f"Ошибка при выполнении запроса: {e}")  
 result = None  
 finally:  
 connection.close()  
 return result  
  
def execute\_query(query, params=()):  
 try:  
 db\_path = get\_db\_path()  
 if not os.path.exists(db\_path):  
 raise Exception(f"База данных не найдена по пути: {db\_path}")  
 connection = sqlite3.connect(db\_path)  
 cursor = connection.cursor()  
 cursor.execute(query, params)  
 connection.commit()  
 except sqlite3.Error as e:  
 print(f"Ошибка при выполнении запроса: {e}")  
 finally:  
 connection.close()

Файл main.py

import sys  
from PyQt6.QtWidgets import QApplication  
from gui.login\_window import LoginWindow  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 login\_window = LoginWindow()  
 login\_window.show()  
 sys.exit(app.exec())

**Приложение Г**

Интерфейс программы для пользователя.

