Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №5

по курсу «Базы данных»

Выполнил студент группы ИВТ-31\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Бикметов И.Р/

Проверил доцент кафедры ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Клюкин В.Л./

Киров 2025

1. Задание

- разработать приложение на python для работы с БД

- приложение должно выполнять следующие действия: отображение БД, добавление записи, удаление записи, сортировка БД

2. Диаграмма классов для работы с БД

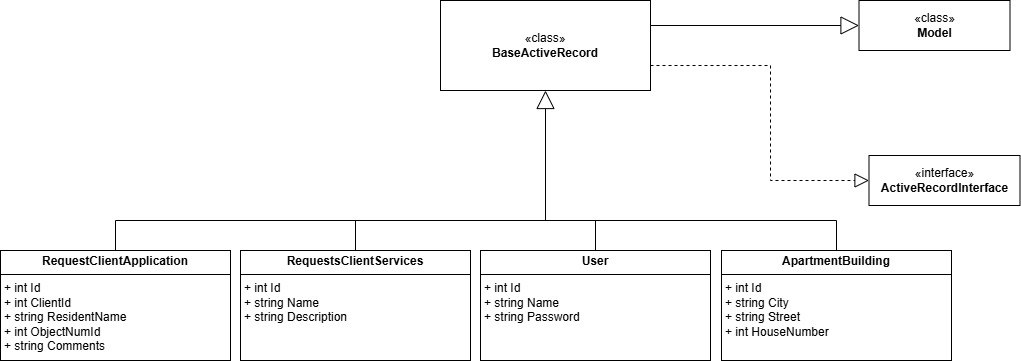


Рисунок 1 – диаграмма классов

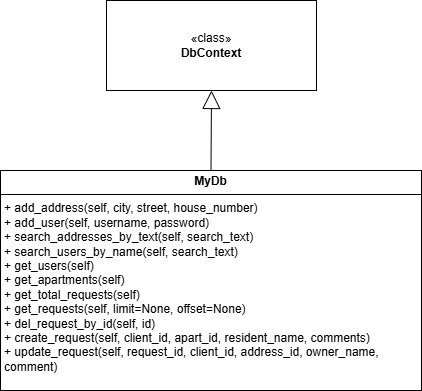


Рисунок 2 – диаграмма классов

3. Листинг кода

from sqlalchemy import create\_engine

from sqlalchemy.orm import sessionmaker

from sqlalchemy.exc import SQLAlchemyError

from sqlalchemy import or\_, cast, func

from User import \*

from ClientApplication import \*

from ApartmentBuilding import \*

class MyDb():

    def \_\_init\_\_(self):

        self.engine = create\_engine("postgresql://postgres:13579@localhost/mydb")

        self.Session = sessionmaker(autoflush=False, bind=self.engine)

    def get\_address\_by\_details(self, city, street, house\_number):

        try:

            with self.Session(autoflush=False, bind=self.engine) as db:

                # Ищем адрес по городу, улице и номеру дома

                address = db.query(ApartmentBuilding).filter(

                    ApartmentBuilding.city == city,

                    ApartmentBuilding.street == street,

                    ApartmentBuilding.house\_number == house\_number

                ).first()

                return address

        except SQLAlchemyError as e:

            print(f"Ошибка при поиске адреса: {e}")

            return None

    def add\_address(self, city, street, house\_number):

        try:

            with self.Session(autoflush=False, bind=self.engine) as db:

                # Создаем новый объект ApartmentBuilding

                new\_address = ApartmentBuilding(

                    city=city,

                    street=street,

                    house\_number=house\_number

                )

                db.add(new\_address)

                db.commit()

                print(f"Адрес успешно добавлен: {city}, {street}, {house\_number}")

                return new\_address

        except SQLAlchemyError as e:

            db.rollback()

            print(f"Ошибка при добавлении адреса: {e}")

            return None

    def get\_user\_by\_name\_and\_password(self, name, password):

        try:

            with self.Session(autoflush=False, bind=self.engine) as db:

                # Ищем пользователя по имени и паролю

                user = db.query(User).filter(

                    User.name == name,

                    User.password == password

                ).first()

                return user

        except SQLAlchemyError as e:

            print(f"Ошибка при поиске пользователя: {e}")

            return None

    def add\_user(self, username, password):

        try:

            with self.Session(autoflush=False, bind=self.engine) as db:

                # Создаем нового пользователя

                new\_user = User(name=username, password=password)

                db.add(new\_user)

                db.commit()

                print(f"Пользователь {username} успешно добавлен!")

                return new\_user

        except SQLAlchemyError as e:

            db.rollback()

            print(f"Ошибка при добавлении пользователя: {e}")

            return None

    def search\_addresses\_by\_text(self, search\_text):

        with self.Session(autoflush=False, bind=self.engine) as db:

            # Если search\_text пустой, возвращаем все адреса

            if not search\_text:

                addresses = db.query(ApartmentBuilding).all()

                return addresses

            # Преобразуем house\_number в текст для поиска

            house\_number\_as\_text = cast(ApartmentBuilding.house\_number, String)

            # Выполняем поиск адресов по подстроке

            addresses = db.query(ApartmentBuilding).filter(

                or\_(

                    ApartmentBuilding.city.ilike(f"%{search\_text}%"),  # Поиск по городу

                    ApartmentBuilding.street.ilike(f"%{search\_text}%"),  # Поиск по улице

                    house\_number\_as\_text.ilike(f"%{search\_text}%")  # Поиск по номеру дома (как текст)

                )

            ).all()

            return addresses

    def search\_users\_by\_name(self, search\_text):

        with self.Session(autoflush=False, bind=self.engine) as db:

            # Выполняем поиск пользователей по подстроке

            users = db.query(User).filter(

                or\_(

                    User.name.ilike(f"%{search\_text}%"),  # Поиск по имени (регистронезависимый)

                )

            ).all()

            return users

    def get\_users(self):

        with self.Session(autoflush=False, bind=self.engine) as db:

            users = db.query(User).order\_by(User.id.desc()).all()

            return users

    def get\_apartments(self):

        with self.Session(autoflush=False, bind=self.engine) as db:

            apartments = db.query(ApartmentBuilding).order\_by(ApartmentBuilding.id.desc()).all()

            return apartments

    def get\_total\_requests(self):

        with self.Session(autoflush=False, bind=self.engine) as db:

            # Выполняем запрос для получения общего количества заявок

            total\_requests = db.query(func.count(ClientApplication.id)).scalar()

            return total\_requests

    def get\_requests(self, limit=None, offset=None):

        with self.Session(autoflush=False, bind=self.engine) as db:

            # Выполняем запрос с пагинацией

            query = db.query(ClientApplication, ApartmentBuilding, User)\

                .join(ApartmentBuilding)\

                .join(User)\

                .order\_by(ClientApplication.id.desc())

            if limit is not None:

                query = query.limit(limit)

            if offset is not None:

                query = query.offset(offset)

            requests = query.all()

            return requests

    def get\_request\_by\_id(self, id):

        with self.Session(autoflush=False, bind=self.engine) as db:

            request = (

                db.query(ClientApplication)

                .filter(ClientApplication.id == id)

                .first()

            )

            return request

    def del\_request\_by\_id(self, id):

        try:

            with self.Session(autoflush=False, bind=self.engine) as db:

                # Находим запись по id

                request = db.query(ClientApplication).filter(ClientApplication.id == id).first()

                if request:

                    # Удаляем запись

                    db.delete(request)

                    db.commit()

                    print(f"Запись с id={id} успешно удалена.")

                else:

                    print(f"Запись с id={id} не найдена.")

        except SQLAlchemyError as e:

            db.rollback()

            print(f"Ошибка при удалении записи: {e}")

    def get\_request\_by\_data(self, client\_id, apart\_id, resident\_name, comments):

        with self.Session(autoflush=False, bind=self.engine) as db:

            # Находим запись по id

            request = db.query(ClientApplication).filter(ClientApplication.client\_id == client\_id).filter(ClientApplication.object\_num\_id == apart\_id).filter(ClientApplication.resident\_name == resident\_name).filter(ClientApplication.comments == comments).first()

            return request

    def create\_request(self, client\_id, apart\_id, resident\_name, comments):

        try:

            with self.Session(autoflush=False, bind=self.engine) as db:

                # Создаем новый объект ClientApplication

                new\_request = ClientApplication(

                    client\_id=client\_id,

                    object\_num\_id=apart\_id,

                    resident\_name=resident\_name,

                    comments=comments,

                    object\_type\_id = 1

                )

                # Добавляем объект в сессию

                db.add(new\_request)

                # Фиксируем изменения в базе данных

                db.commit()

                print("Заявка успешно создана!")

                return new\_request  # Возвращаем созданный объект (опционально)

        except SQLAlchemyError as e:

            # В случае ошибки откатываем изменения

            db.rollback()

            print(f"Ошибка при создании заявки: {e}")

            return None

    def update\_request(self, request\_id, client\_id, address\_id, owner\_name, comment):

        try:

            with self.Session(autoflush=False, bind=self.engine) as db:

                # Находим заявку по ID

                request = db.query(ClientApplication).filter(ClientApplication.id == request\_id).first()

                if request:

                    # Обновляем поля заявки

                    request.client\_id = client\_id

                    request.object\_num\_id = address\_id

                    request.resident\_name = owner\_name

                    request.comments = comment

                    # Фиксируем изменения в базе данных

                    db.commit()

                    print("Заявка успешно обновлена!")

                    return request  # Возвращаем обновленный объект (опционально)

                else:

                    print("Заявка с указанным ID не найдена.")

                    return None

        except SQLAlchemyError as e:

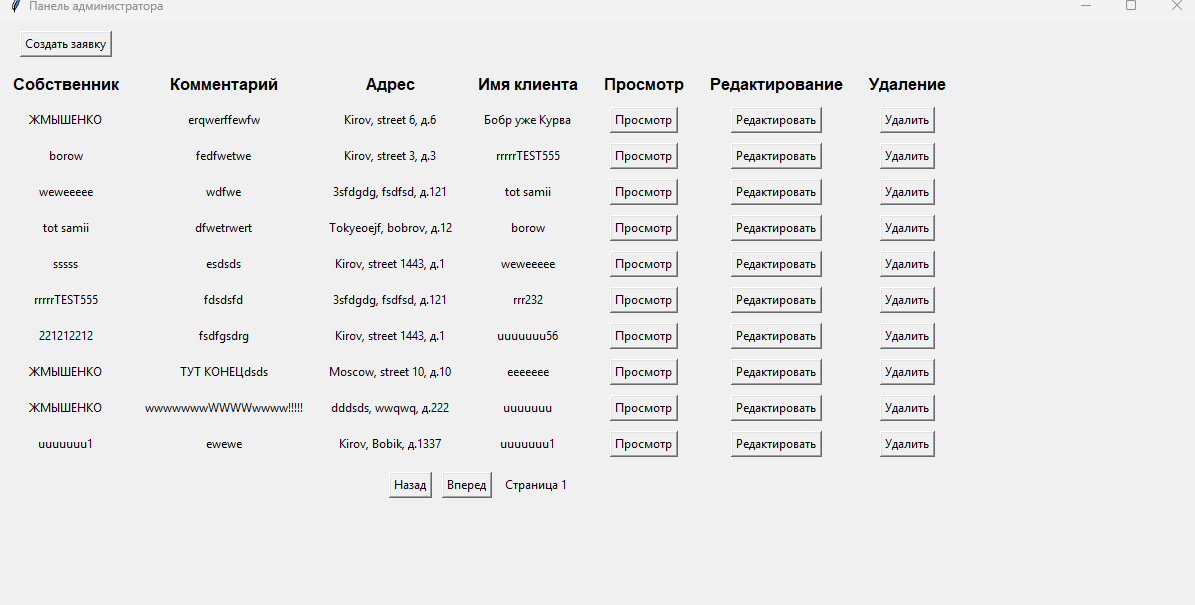
            # В случае ошибки откатываем изменения

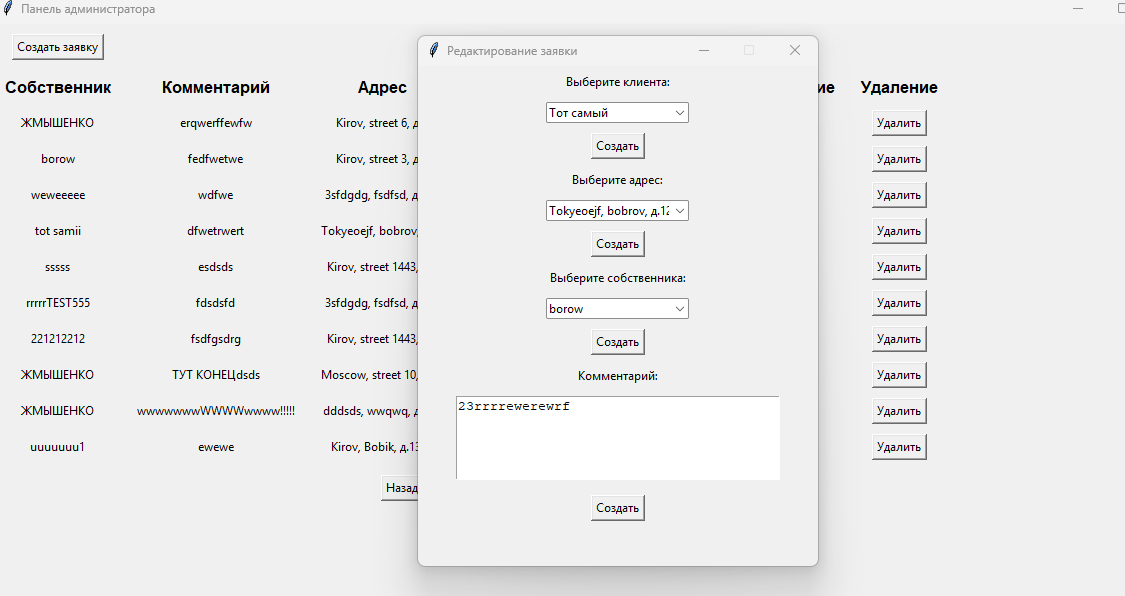
            db.rollback()

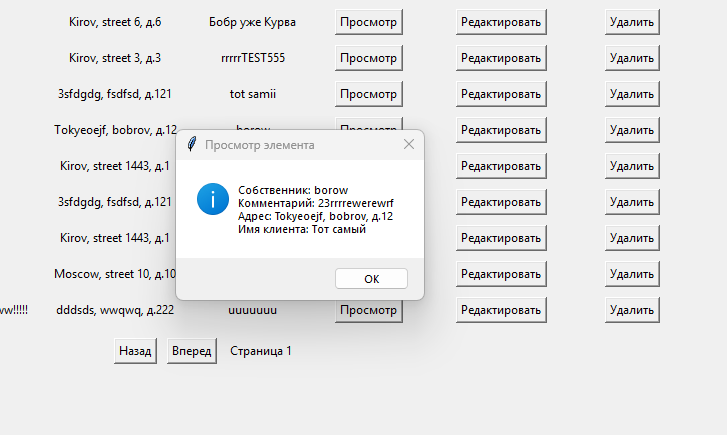
            print(f"Ошибка при обновлении заявки: {e}")

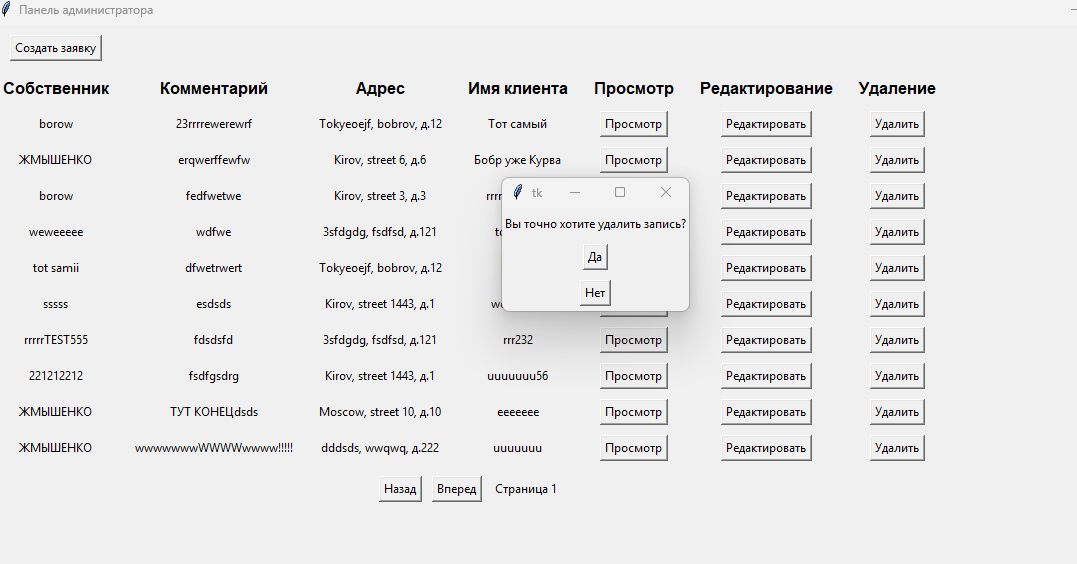
            return None

4. Экранные формы









5. Вывод

В ходе выполнения лабораторной работы были получены навыки разработки приложения для работы с бд.