Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:
“Агрегатор информации по горнолыжным курортам”

Студент ИУ7-63Б _____ И. Д. Челядинов
(Группа) (Подпись,дата) (И.О. Фамилия)

Руководитель курсового проекта _____ Д.Е. Бекасов
(Подпись, дата) (И.О.Фамилия)

Москва, 2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
_____ И.В.Рудаков
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

ЗАДАНИЕ
на выполнение курсового проекта

по дисциплине Базы данных

Студент группы ИУ7-63Б

Челядинов Илья Дмитриевич
(Фамилия, имя, отчество)

Тема курсового проекта Агрегатор информации по горнолыжным курортам.
Направленность КП (учебный, исследовательский, практический, производственный, др.)

учебный

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание*:*
Разработать приложение, позволяющее пользователю: осуществлять регистрацию, вход, осуществлять поиск горнолыжных курортов в базе данных, оставлять комментарии к найденным курортам, а также просматривать информации по найденным курортам, такую как: открытые подъемники, открытые трассы. Предусмотреть роли модератора, администратора, и пользователя приложения. Обеспечить для модератора возможность корректировки существующей информации: удаление пользователей, комментариев. Обеспечить возможность для администратора вносить изменения в базу данных: удалять пользователей, выдавать пользователям права администратора, удалять комментарии.

Оформление *курсового проекта:* Расчетно-пояснительная записка на 25-30 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
На защиту проекта должна быть предоставлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, интерфейс, результаты проведенных исследований.

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсового проекта _____ Бекасов Д. Е.
(Подпись, дата) (И.О.Фамилия)

Студент _____ Челядинов И. Д.
(Подпись, дата) (И.О.Фамилия)

Москва, 2021 г.

Оглавление

Введение

В современном мире из – за стремительного роста количества информации в сети пользователю становится труднее добыть достоверную информацию. Для того, чтобы получить полную информацию об интересном пользователю объекте, необходимо посетить несколько ресурсов.

Данная проблема может быть решена с помощью современной технологии – “Агрегатора”. Агрегатор – это веб приложение, которое автоматически собирает информацию с различных информационных ресурсов и предоставляет ее пользователю в удобном, доступном виде.

Цель данной работы – создание агрегатора, который поможет пользователям получать достоверную информацию о горнолыжных курортах.

Чтобы достичь поставленной цели, необходимо решить следущие задачи:

1) сформировать задание и необходимый функционал;
2) сформировать требования к базе данных, определить круг задач, которые будут решаться с использованием созданной базы данных;
3) спроектировать модель базы данных;
4) спроектировать приложение для доступа к базе данных;
5) осуществить выбор СУБД и технических средств на основе сформированных требований;
6) создать базу данных и необходимые сущности;
7) разработать программный интерфейс для работы с базой данных;
8) разработать программное обеспечение для взаимодействия пользователя и базы данных.

1 Аналитическая часть

В данном разделе будет проанализирована поставленная задача и рассмотрены способы ее реализации.

1.1 Постановка задачи

Разработать веб-сервис “агрегатор” для отображения сети о горнолыжных курортах. В данном веб-сервисе предусмотреть возможность просматривания информации о горнолыжных курортах “Роза Хутор”, “Горнолыжный курорт “Лаура””, “Красная поляна”, такой как: существующие курорты, открытые подъемники и трассы. Также реализовать возможность создания аккаунта и добавления комментариев. Предусмотреть роли пользователя, модератора, администратора. Для роли модератора реализовать возможность добавления, удаления комментариев, а также удаления пользователей с ролью “пользователь”. Для роли администратора реализовать возможность добавления и удаления комментариев, удаления пользователей с ролью “пользователь” и “модератор”, изменения роли пользователей на роль “модератор”. Вся информация о курортах, пользователях, комментариях должна содержаться в базе данных. Также все действия пользователей должны автоматически сохраняться в аудит-журнале.

1.2 Формализация данных

База данных должна хранить информацию о:

- горнолыжных курортах;
- трассах и подъемниках, в том числе открытых и закрытых;
- пользовательских аккаунтов;
- комментариев, оставленных пользователями;
- действиях, совершаемых пользователями – журнал аудит.

Таблица 1.1 – категории и сведения о данных

Категория	Сведения
Курорт	Название курорта, город.
Дороги	Тип дороги (подъемник или трасса), название дороги, статус работы дороги, длина дороги, ширина дороги, время работы дороги.
Пользователь	Имя аккаунта, почта пользователя, пароль, роль пользователя, дата регистрации.
Комментарии	Идентификатор пользователя, идентификатор курорта, содержимое комментария.
Аудит	Идентификатор пользователя, описание действия пользователя, дата и время действия.

Связи между сущностями отображены на ER – диаграмме:

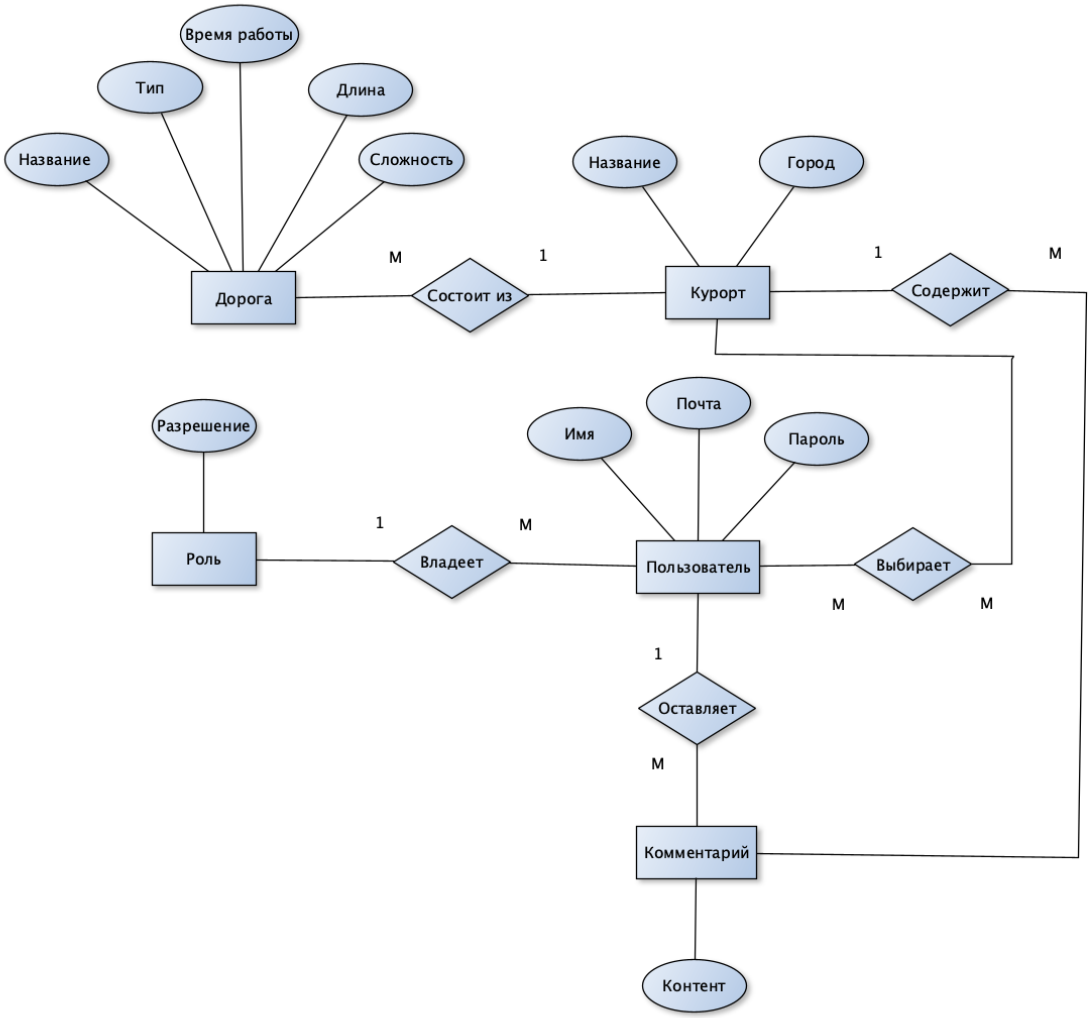

Рис. 1.1 – диаграмма сущностей

1.3 Типы пользователей

Для того, чтобы реализовать некоторые функции веб-сервиса, необходимо предусмотреть ролевую модель.

Для управления действиями пользователей принято решение ввести три роли, кроме базовой роли гостя: пользователь, модератор, администратор.

Таблица 1.2 – типы пользователей и их функционал

Тип пользователя	Функционал
Гость	Просмотр главной страницы, поиск дорог, вход, регистрация.

Авторизированный пользователь	Просмотр главной страницы, просмотр страницы пользователя, поиск дорог, просмотр доступных курортов и информации о них, просмотр информации о каждом курорте, добавление комментариев о курортах.
Модератор	Просмотр главной страницы, просмотр страницы пользователя, просмотр страницы администратора, поиск дорог, просмотр доступных курортов и информации о них, просмотр информации о каждом курорте, добавление комментариев о курортах, удаление комментариев о курортах, удаление аккаунтов пользователей.
Администратор	Просмотр главной страницы, просмотр страницы пользователя, просмотр страницы администратора, поиск дорог, просмотр доступных курортов и информации о них, просмотр информации о каждом курорте, добавление комментариев о курортах, удаление комментариев о курортах, удаление аккаунтов пользователей, изменение роли пользователя на роль модератора.

1.4 Описание существующих систем управления базами данных

Система управления базами данных, сокр. СУБД — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

1.4.1 Основные функции СУБД

Основными функциями СУБД являются:

- управление данными во внешней памяти;
- управление данными в оперативной памяти;
- журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
- поддержка языков БД;

1.4.2 Классификация СУБД по модели данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных. [2]

Существует 3 основных типа моделей организации данных:

- иерархическая;
- сетевая;
- реляционная.

В иерархической модели данных используется представление базы данных в виде древовидной структуры, состоящей из объектов различных уровней. Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка к потомку, при этом возможна ситуация, когда объект-предок имеет несколько потомков, тогда как у объекта-потомка обязателен только один предок.

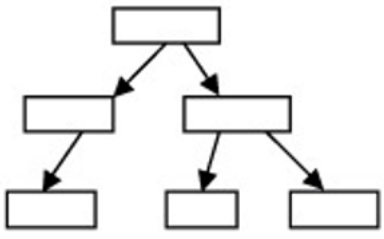

Рис 1.2 – структура иерархической модели данных

В сетевой модели данных, в отличии от иерархической, у потомка может иметься любое число предков. Сетевая БД состоит из набора экземпляров определенного типа записи и набора экземпляров определенного типа связей между этими записями.

Главным недостатком сетевой модели данных являются жесткость и высокая сложность схемы базы данных, построенной на основе этой модели. Так как логика процедуры выбора данных зависит от физической организации этих данных, то эта модель не является полностью независимой от приложения. Иначе говоря, если будет необходимо изменить структуру данных, то нужно будет изменять и приложение.

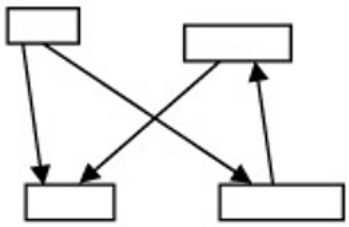

Рис 1.3 – структура сетевой модели данных

Реляционная модель данных является совокупностью данных и состоит из набора двумерных таблиц. При табличной организации отсутствует иерархия элементов. Таблицы состоят из строк – записей и столбцов – полей. На пересечении строк и столбцов находятся конкретные значения. Для каждого поля определяется множество его значений. За счет возможности просмотра строк и столбцов в любом порядке достигается гибкость выбора подмножества элементов.

Реляционная модель является удобной и наиболее широко используемой формой представления данных.

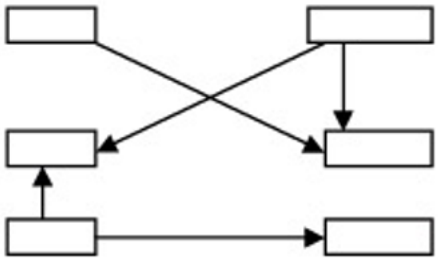

Рис 1.4 – структура реляционной модели данных

1.5 Выбор СУБД

В настоящее время существует множество различных СУБД, наиболее популярные из которых: MongoDB, Oracle, MS SQL Server, PostgreSQL, Tarantool. В связи с выбором реляционной модели данных, необходимо провести сравнительный анализ между наиболее популярными реляционными СУБД, такими как PostgreSQL, MS SQL Server и Oracle:

Таблица 1.3 – сравнительный анализ реляционных СУБД

	PostgreSQL	MS SQL Server	Oracle
Свободное распространение	+	+	-
Год выпуска	1989	1989	1980
Последний релиз	13.4, август 2021	SQL Server 2019, ноябрь 2019	19c, февраль 2019
Поддержка триггеров	+	+	+
Поддержка json	+	+	+
Возможность параллельного выполнения	+	+	+

Как видно из таблицы, каждая СУБД имеет свои плюсы, поэтому для реализации агрегатора была выбрана СУБД Postgresql исключительно из – за опыта работы с данной СУБД.

1.6 Выводы из аналитического раздела

В данном разделе была проанализирована поставленная задача и способы ее реализации, рассмотрены основные типы баз данных. В качестве используемой в данной работе была выбрана реляционная СУБД PostgreSQL.

2 Конструкторская часть

2.1 Проектирование базы данных

Спроектированная база данных будет хранить рассмотренные в таблице 1.1 данные. В соответствии с данной таблицей можно выделить следующие сущности:

- таблица курортов Courorts;
- таблица дорог Roads;
- таблица пользователей Users;
- таблица комментариев Comment;
- Таблица аудит Audit.

1) Таблица Courorts должна хранить информацию о горнолыжных курортах:

- id_courort – уникальный идентификатор курорта, integer, PK;
- name_courort – название курорта, varchar, unique not null;
- city – город, в котором располагается данный курорт, varchar, not null;
- visability – видимость курорта, поле которое позволит не отображать курорты при расширении проекта, integer, not null.

2) Таблица Roads должна хранить информацию о трассах и подъемниках:

- id_road – уникальный идентификатор дороги, integer, PK;
- id_courort – идентификатор курорта, в котором находится данная дорога, integer, not null, FK;
- type_road – тип дороги (подъемник/трасса), varchar not null;
- name_road – название дороги, varchar, not null;
- work_status – статус работы дороги в данный момент, integer, not null;
- complexity – сложность трассы, опциональное значение, varchar;
- length – длина трассы, опциональное значение, varchar;
- width – ширина трассы, опциональное значение, varchar;
- worktime – время работы дороги, опциональное значение, varchar.

3) Таблица пользователей Users должна хранить информацию о пользователях:

- id_user – уникальный идентификатор пользователя, integer, PK;
- name – имя аккаунта пользователя, varchar, not null;
- email – почта пользователя, varchar, not null;
- password – пароль пользователя, varchar, not null;
- user_role – роль пользователя, varchar, not null;
- dt_registration – дата регистрации, date.

4) Таблица Comment должна хранить информацию о комментариях:

- id_comment – уникальный идентификатор комментария, integer, PK;
- id_user – уникальный идентификатор пользователя, которому принадлежит комментарий, integer, FK;
- id_courort – уникальный идентификатор курорта, к которому оставлен комментарий, integer, not null;
- content – содержимое комментария, varchar, not null;
- likes – количество пользователей, которым понравился комментарий, integer;
- visability – видимость комментария, integer;
- datetime – дата и время добавления комментария, timestamp;

5) Таблица Audit должна хранить информацию о действиях пользователей:

- id_interaction – уникальный идентификатор взаимодействия, integer, PK;
- id_user – уникальный идентификатор пользователя, integer, FK;
- action – описание действия пользователя, varchar, not null;
- dt_interaction – дата и время действия пользователя, timestamp;

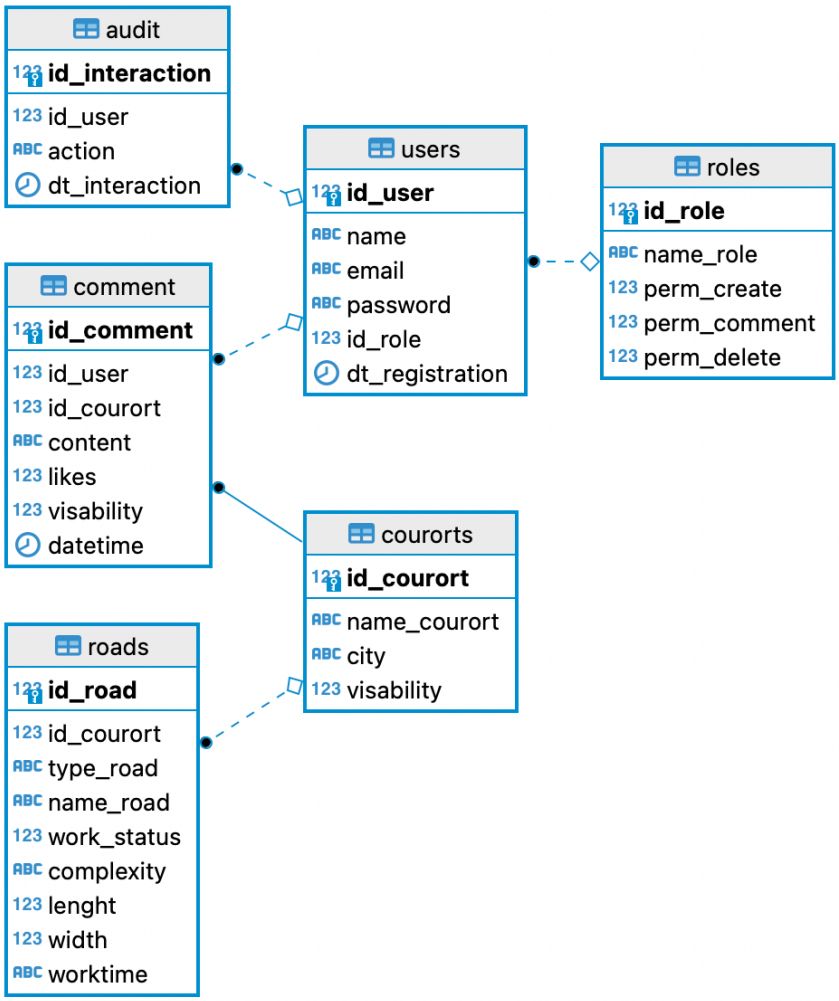

Рис 2.1 – Схема БД.

2.2 Требования к программе

Для реализации поставленной задачи программа должна предоставлять следующие возможности:

- авторизация;
- регистрация;
- вывод списка курортов;
- вывод списка дорог для каждого курорта;
- возможность поиска дорог;
- вывод списка комментариев;
- возможность добавления комментариев.

Возможности модератора:

- удаление комментариев;
- удаление пользователей;
- вывод списка пользователей.

Возможности администратора:

- удаление комментариев;
- удаление пользователей и модераторов;
- смена роли пользователя;
- вывод списка пользователей.

2.3 Проектирование приложения

Программа будет состоять из двух составных частей:

1) front-end – приложение, с которым непосредственно взаимодействует пользователь;
2) back-end – приложение, с которым взаимодействует front-end и база данных: получение, изменение, удаление данных

2.3.1 Проектирование back-end части приложения

Для корректной работы проекта back-end часть приложения будет состоять из следующих составных частей: API, database, parser. API проектировалось по принципам REST.

API (Application Programming Interface) – серверная часть приложения, которая должна принимать, обрабатывает запросы от front-enda взаимодействует с промежуточным слоем между API и database.

Database – часть приложения, которая должна содержать модели для работы с базой данных, а также клиент, который принимает запросы от API и отправляет их непосредственно базе данных.

Parser – часть приложения, которая должна получать, обрабатывать необходимые данные для работы агрегатора.

Часть Database должна состоять из следующих компонентов:

- DbConnection – класс коннектор, который должен напрямую осуществлять подключение к базе данных.
- DbInteraction – класс, через который должны происходить все взаимодействия между базой данных и API частью;
- функции для работы с данными.

Для удовлетворения требованиям к программе, описанным в разделе 2.2, необходимы следующие функции доступа к данным:

- add_user_info – функция добавления нового пользователя;
- get_user – функция получения данных пользователя, если он существует. Необходима для входа в аккаунт;
- delete_user_by_email – функция удаления пользователя;
- make_mod_by_email – функция смены роли пользователя;
- add_comment – функция добавления комментария;
- delete_comment – функция удаления комментария;
- show_comments – функция получения комментариев;
- update_rosa – функция обновления данных о курорте “Роза хутор”;
- update_laura – функция обновления данных о курорте “Лаура”;
- update_polyana – функция обновления данных о курорте “Красная поляна”;
- update_roads – функция, с которой взаимодействует parser: очищает таблицу с неактуальными данными и обновляет их путем вызова функции update_rosa, update_laura, update_polyana;
- get_all_users – функция получения списка пользователей;
- check_user – функция, которая проверяет существование пользователя;
- get_courorts – функция, которая выдает список курортов;
- get_roads_and_courorts – функция получения списка дорог и курортов;
- get_rosa – функция получения дорог на курорте “Роза хутор”;
- get_gorod – функция получения дорог на курорте “Красная поляна”;
- get_laura – функция получения дорог на курорте “Лаура”;

Также для работы с таблицей аудит необходимо реализовать триггеры:

- user_insert_trigger – триггер, который добавляет запись в таблицу “audit” при добавлении данных в таблицу “users”;
- comments_insert_trigger – триггер, который добавляет запись в таблицу “audit” при добавлении данных в таблицу “comments”;
- user_delete_trigger – триггер, который добавляет запись в таблицу “audit” при удалении данных в таблице “users”.

2.4 Вывод из конструкторского раздела

В данном разделе были спроектирована база данных и приложение, были формализованы требования к программе.

3 Технологическая часть

3.1 Выбор и обоснование языка программирования и среды разработки

В качестве языка разработки back-end части был выбран Python по следующим причинам:

- данный язык прост в освоении, что существенно повышает скорость разработки;
- поддерживает большое количество библиотек для работы с данными, что также облегчает разработку.

В качестве среды разработки front-end части был выбран Javascript, фреймворк Vue.

В качестве среды разработки для разработки back-end части была выбрана “Pycharm” по следующим причинам:

- бесплатна в пользовании студентам;
- имеет удобные короткие команды, что значительно ускоряет разработку;
- имеет возможность отладки кода.

Редактор “Visual Studio Code” был выбран для front-end части по следующим причинам:

- бесплатное ПО;
- имеет множество плагинов для удобной работы с веб-приложениями.

3.2 Структура и состав классов

В данном разделе будет рассмотрена структура и состав классов.

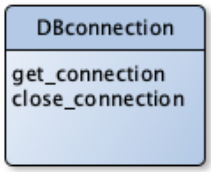

Рис 3.2.1 – структура класса DBconnection

DBconnection – класс коннектора, который осуществляет подключение к базе данных напрямую, выполняет различные запросы. Все взаимодействие с БД происходит через коннектор.

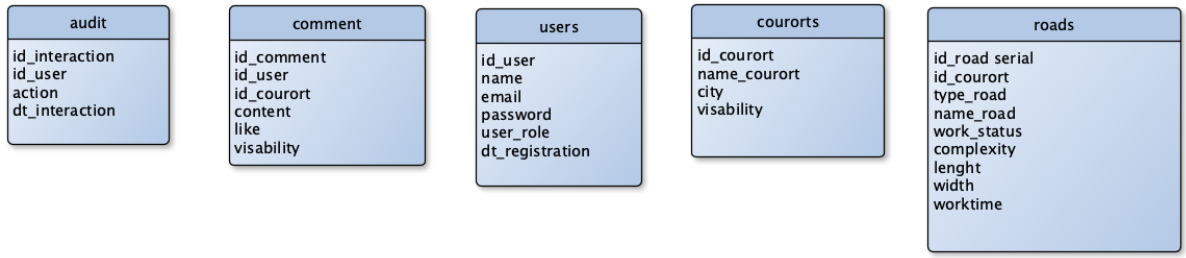

Рис. 3.2.2 – структура моделей ORM.

Audit – класс аудит;

comment – класс комментариев;

users – класс пользователя

courorts – класс курортов

roads – класс дорог курортов

В листинге 3.2.1. представлена модель таблицы Users, через которую происходит взаимодействие с таблицей в базе данных.

Листинг 3.2.1 – модель таблицы Users

```
class Users(DbModel):
    id_user = AutoField(column_name="id_user")
    name = TextField(unique=True)
    password = TextField(column_name="password")
    email=TextField(column_name="email")
    user_role = TextField(column_name="user_role")
    dt_registration = DateField(column_name="dt_registration", default=datetime.now())

    class Meta:
        table_name = "users"
```

Взаимодействие с базой данных происходит с помощью класса DBconnection с помощью класса DbInteraction – листинги 3.2.2, 3.2.3.

Листинг 3.2.2 – класс DBconnection

```python
class DBconnection:
    def __init__(self, host, port, user, db_name, password, rebuild_db=False):
        self.host = host
        self.port = port
        self.user = user
        self.db_name = db_name
        self.password = password
        self.rebuild_db = rebuild_db
        self.pg_conn = 0

    def get_connection(self):
        self.pg_conn = PostgresqlDatabase(
            database=self.db_name,
            user=self.user,
            password=self.password,
            host=self.host,
            port=self.port,
        )
        return self.pg_conn

    def close_connection(self):
        self.pg_conn.close()
```

Листинг 3.2.3 – конструктор класса DbInteraction

```python
class DbInteraction:
    def __init__(self, host, port, user, db_name, password, rebuild_db=False):
        self.postgres_connection = DBconnection(host, port, user, db_name, password, rebuild_db)

        if rebuild_db:
            self.rebuild()
```

В листингах 3.2.4, 3.2.5 представлены примеры запросов к базе данных.

Листинг 3.2.4 – пример запроса к базе данных через модель Users

```python
def get_user_info(self, username):
    query = Users.select().where(Users.name==username)
    curs = self.postgres_connection.get_connection().cursor()
    curs.execute(str(query))
    user = curs.fetchone()
    curs.close()

    if user:
        return {'username':user[1], 'password':user[2], 'email':user[3]}
    else:
        raise UserNotFoundException('UserNotFound')
```

Листинг 3.2.5 – пример запроса к базе данных через DbInteraction и DBconnection

```
def make_mod_by_email(self, email):
    query_comm = "update users set user_role = 'mod' where email = '" +str(email) + "    ';"
    cur = self.postgres_connection.get_connection().cursor()
    cur.execute(query_comm)
    self.postgres_connection.pg_conn.commit()
```

За коммуникацию между пользовательским интерфейсом и базой данных отвечает класс Server.

Листинг 3.2.6 – конструктор класса Server

```
class Server:
    def __init__(self, host, port, db_host, db_port, user, db_name, password, rebuild_db=False):
        self.host = host
        self.port = port

        self.db_interaction = DbInteraction(
            host=db_host,
            port=db_port,
            user=user,
            password=password,
            db_name=db_name,
            rebuild_db=True
        )

        self.app = Flask(__name__)
        self.app.config['SECRET_KEY'] = 'super-secret'
        self.cors = CORS(self.app)
        self.jwt = JWTManager(self.app)
        self.app.config.from_object(__name__)
        self.app.add_url_rule("/shutdown", view_func=self.shutdown)
        self.app.add_url_rule("/ping", view_func=self.get_home)
        self.app.add_url_rule("/home", view_func=self.get_home)
        self.app.add_url_rule("/add_user", view_func=self.add_user_info, methods=['POST'])
        self.app.add_url_rule("/login", view_func=self.login, methods=['POST'])
        self.app.add_url_rule("/get_courorts", view_func=self.get_courorts)
        self.app.add_url_rule("/get_rosa", view_func=self.get_rosa)
        self.app.add_url_rule("/get_gorod", view_func=self.get_gorod)
        self.app.add_url_rule("/get_laura", view_func=self.get_laura)
        self.app.add_url_rule("/get_roads_and_courorts", view_func=self.get_roads_and_courorts)
        self.app.add_url_rule("/get_all_users", view_func=self.get_all_users)
        self.app.add_url_rule("/delete_user", view_func=self.delete_user, methods=['POST'])
        self.app.add_url_rule("/add_comment", view_func=self.add_comment, methods=['POST'])
        self.app.add_url_rule("/get_comments", view_func=self.get_comments, methods=['POST'])
        self.app.add_url_rule("/delete_comment", view_func=self.delete_comment, methods=['POST'])
        self.app.add_url_rule("/make_mod", view_func=self.make_mod, methods=['POST'])
```

В листинге 3.2.7 представлен пример триггера базы данных.

Листинг 3.2.7 – пример триггера базы данных.

```
CREATE TRIGGER comment_insert_trigger
  AFTER INSERT
  ON "comment"
  FOR EACH ROW
  EXECUTE PROCEDURE comments_insert_trigger_fnc();

CREATE OR REPLACE FUNCTION users_delete_trigger_fnc()
  RETURNS trigger AS
$$
BEGIN
    INSERT INTO "audit" ("id_user", "action", "dt_interaction")
    VALUES(NEW."id_user",'user deleted' ,current_date);
RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

CREATE TRIGGER user_delete_trigger
  AFTER delete
  ON "users"
  FOR EACH ROW
  EXECUTE PROCEDURE users_delete_trigger_fnc();
```

3.3 Интерфейс приложения

Интерфейс программы выполнен в виде веб-приложения. На рисунках 3.3.1, 3.3.2 представлена страница поиска дорог.

Рис 3.3.1 – страница поиска дорог горнолыжных курортов

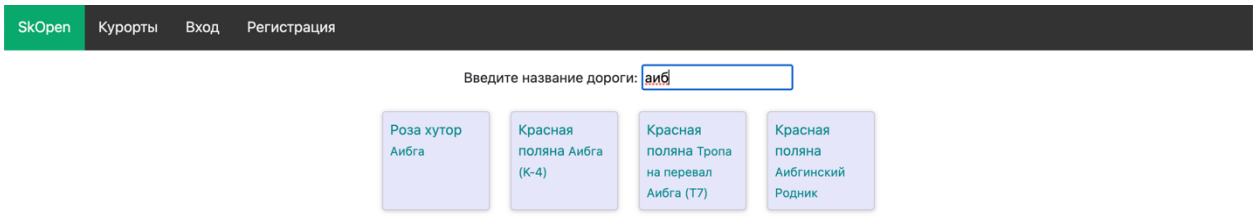

Рис 3.3.2 – страница поиска дорог горнолыжных курортов

На рисунка 3.3.3, 3.3.4 представлены страница курортов и страница с дорогами горнолыжного курорта “Лаура”.

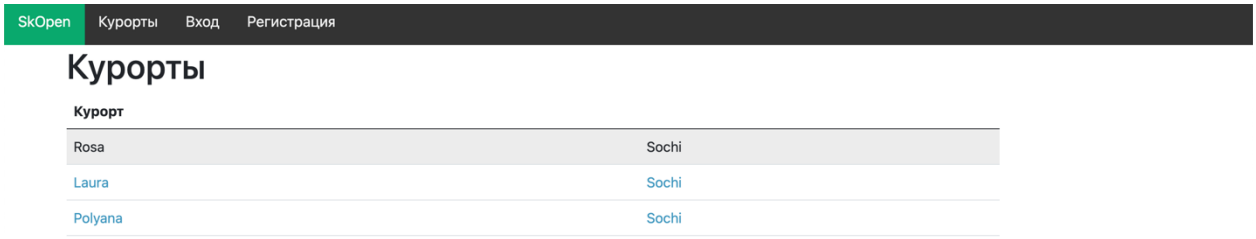

Рис 3.3.3 – страница со списком горнолыжных курортов

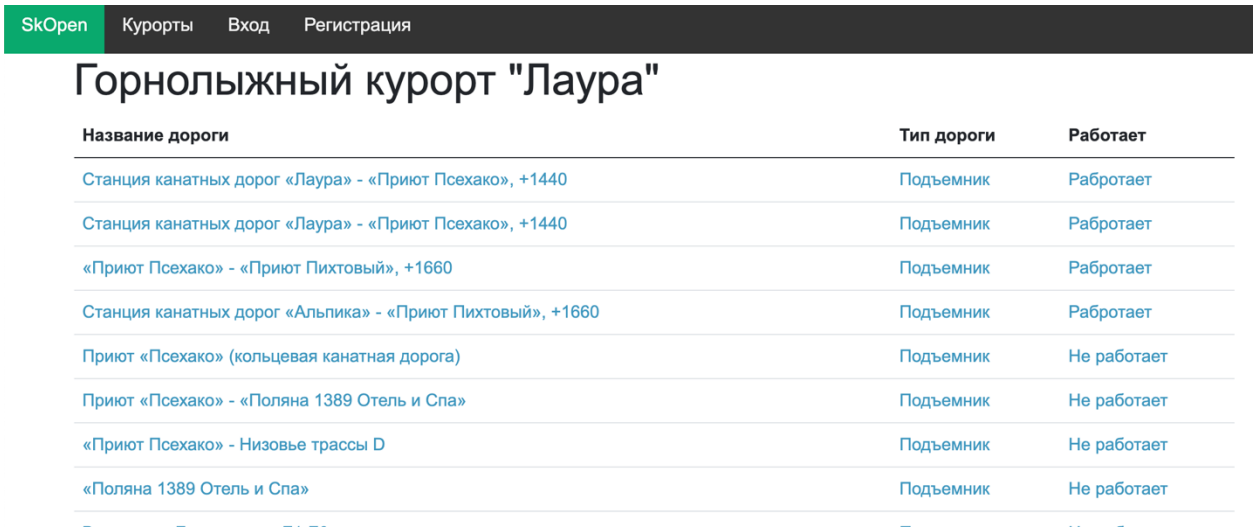

Рис 3.3.4 – горнолыжный курорт “Лаура”

На данной странице пользователь может посмотреть интересующую информацию о горнолыжном курорте “Лаура”, а также оставить комментарий.

На рисунках 3.3.5, 3.3.6, 3.3.7, 3.3.8 представлены страницы входа, регистрации, личная страница пользователя и страница администратора

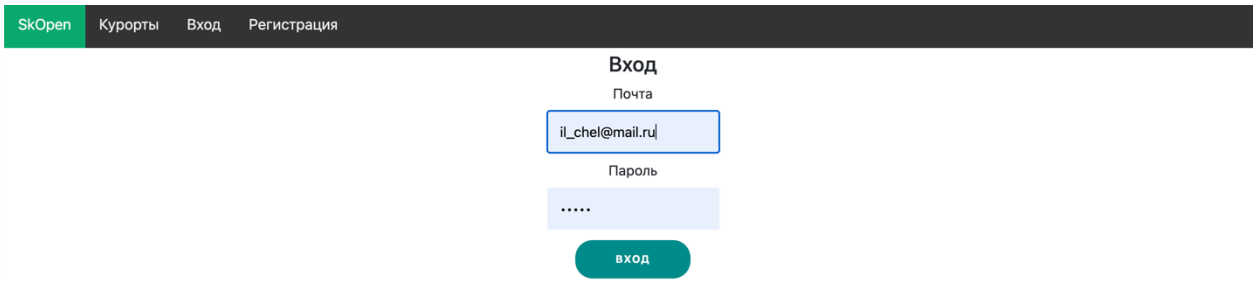

Рис 3.3.5 – страница авторизации

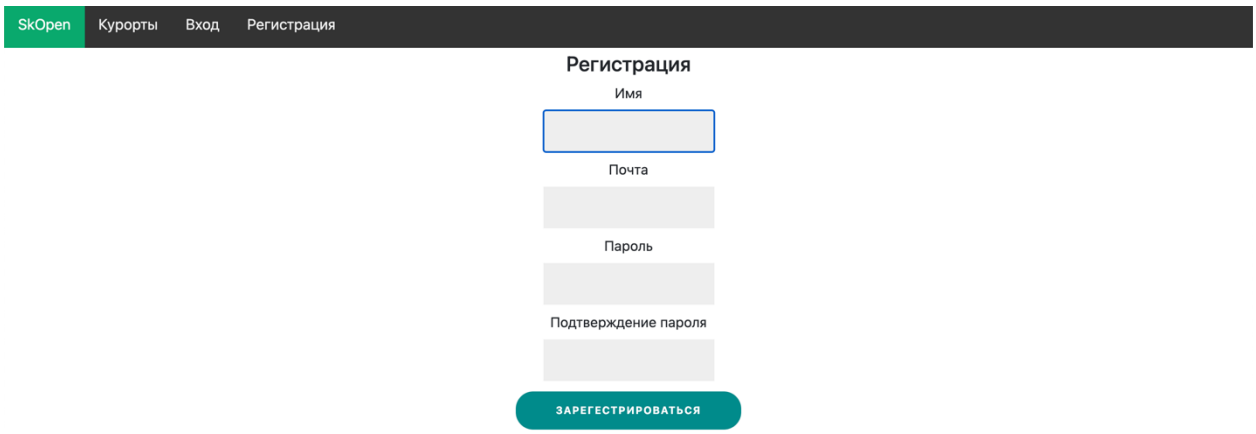

Рис 3.3.6 – страница регистрации

На личной странице пользователя пользователь с ролью “модератор” или “администратор” доступна ссылка на страницу администратора.

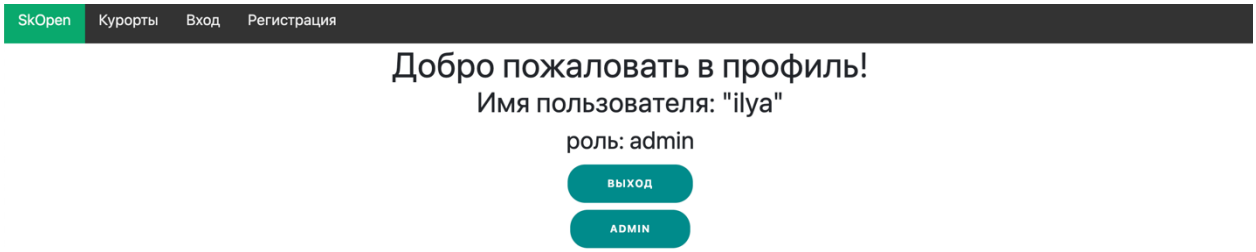

Рис 3.3.7 – личная страница пользователя

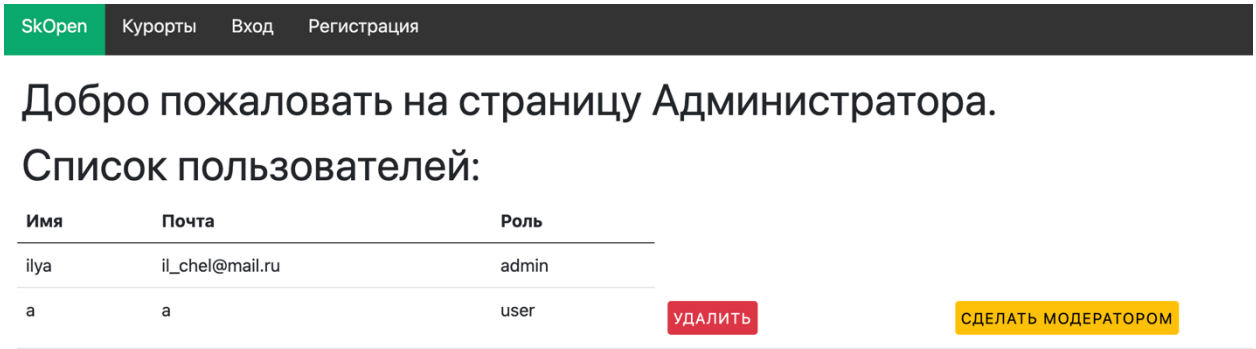

Рис 3.3.8 – страница администратора

На странице администратора пользователь с ролью “модератор” может удалить другого пользователя. Пользователь с ролью “администратор” может удалить другого пользователя или сделать модератором.

На рисунке 3.3.9 представлена секция комментариев горнолыжного курорта “Роза хутор”. В данном разделе пользователь может оставить комментарий к одному из курортов: Роза хутор, Лаура, Красная поляна. У каждого курорта своя секция комментариев. Пользователь с ролью “модератор” или “администратор” может удалить комментарий

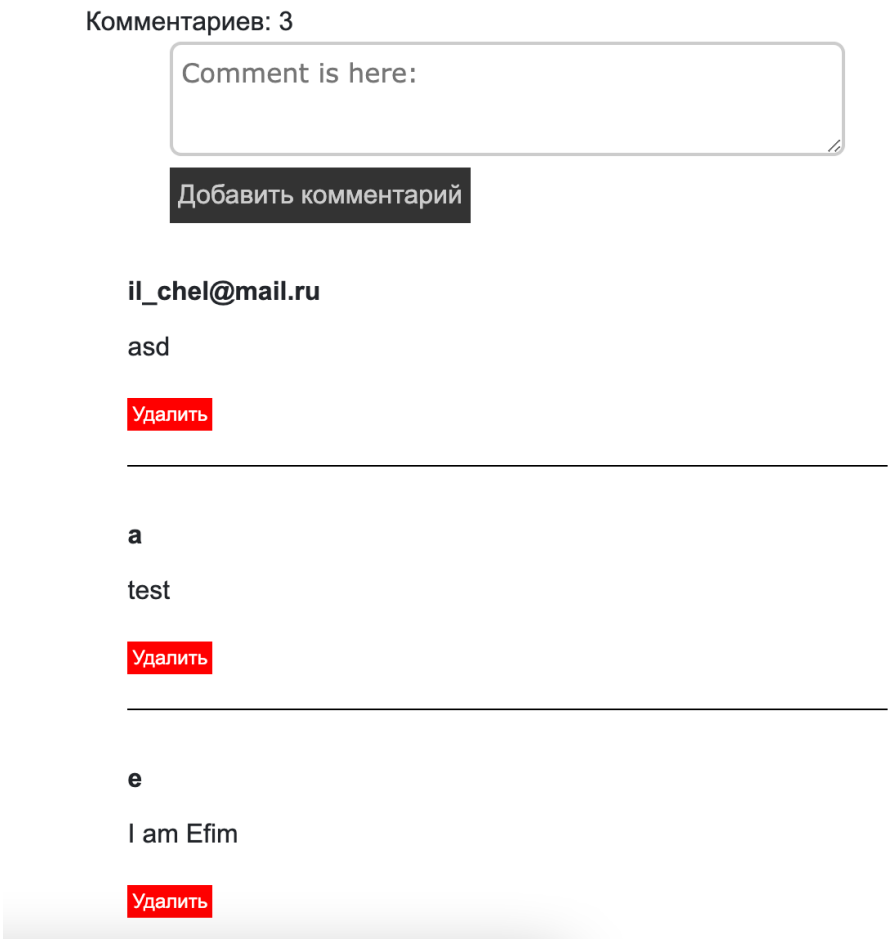

Рис 3.3.9 – секция комментариев

3.4 Выводы из технологического раздела.

В данном разделе были выбран языки программирования Python и Javascript, и среды разработки “Pycharm” и “Visual studio code”. Были рассмотрены структура и состав реализованных классов, представлены сведения об интерфейсе.

Заключение

Во время выполнения курсового проекта были рассмотрены существующие виды баз данных и СУБД. Было разработано программное обеспечение, которое предоставляет интерфейс к базе данных.

Программа реализована таким образом, чтобы давать возможность пользователям получать актуальную и правдоподобную информацию напрямую с веб-сайтов интересующих услуг. Также пользователь может регистрироваться, осуществлять поиск интересной трассы или подъемника по названию, оставлять комментарии для других пользователей. Модератор имеет возможность модерировать действия пользователей, путем удаления комментариев или пользовательских аккаунтов. Администратор имеет возможность удалять пользовательские аккаунты, комментарии, а также изменять роль пользователя.

В ходе выполнения поставленной задачи были изучены возможности СУБД postgresql, возможности разработки серверной части приложений на Python и язык Javascript. Также были приобретены глубокие знания в области баз данных.

Список использованной литературы

1. ISO/IEC TR 10032:2003 Information technology — Reference model of data management

2. Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: «Вильямс», 2006.

3. Архипенков, С. Хранилища данных. От концепции до внедрения / С. Архипенков, Д. Голубев, О. Максименко. - М.: Диалог-Мифи, 2002. - 528 с.

4. Мюллер, Р.Дж. Базы данных и UML. Проектирование / Р.Дж. Мюллер. - М.: ЛОРИ, 2002. - 420 с.

5. Мерсер Drupal 6. Создание надежных и полнофункциональных веб-сайтов, блогов, форумов, порталов и сайтов-сообществ / Мерсер, Дэвид. - М.: Вильямс, 2009. - 272 с.