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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:
_«Разработка веб-сайта для просмотра сведений о различных университетах»

Студент ___ИУ7-63Б___ _____ С.А. Свердлов
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ А.П. Ковтушенко
(Подпись, дата) (И.О.Фамилия)

2022 г.

Реферат

Данная расчетно-пояснительная записка к курсовому проекту на тему «Разработка веб-сайта для просмотра сведений о различных университетах» выполнена на 41 странице, содержит 15 рисунков, 3 таблицы, 16 листингов, список использованных источников из пяти наименований и 1 приложение.

Ключевые слова: проектирование приложения, база данных, университет, разработка веб-сайта, программа, модель данных, пользователь, язык программирования.

Целью данной работы является разработка базы данных с информацией о различных университетах и веб-приложение для взаимодействия с базой данных.

Для достижения обозначенной цели в ходе работы будут достигнуты следующие задачи:

1. провести анализ предметной области и существующих СУБД, сформулировать ограничения для текущего исследования;
2. спроектировать базу данных, содержащую информацию о различных университетах;
3. реализовать веб-приложение для взаимодействия с этой базой данных;
4. исследовать индексы для базовых таблиц, произвести замеры времени.

Работа включает в себя 4 смысловые части: аналитическую, конструкторскую, технологическую и исследовательскую. В ходе работы постепенно достигаются все поставленные задачи. Каждая глава содержит краткие выводы о достигнутых результатах исследования. Также, в работе изучаются возможности языка Python и библиотеки Flask, различных баз данных и принципы работы с PostgreSQL и PGAdmin4. Результатом данного курсового проекта является разработанные приложение и веб-сайт, содержащие сведения о различных университетах.

Содержание

Введение

В современном мире существует множество университетов с различными направлениями. Каждый университет имеет свою уникальность, историю, различную стоимость обучения и проходные баллы. ВУЗ может входить в международные рейтинги. Качество образования напрямую зависит от трудоустройства на работу, поэтому к выбору вуза нужно подходить очень внимательно.

Цель данной работы – разработать базу данных с информацией о различных университетах и веб-приложение для взаимодействия с базой данных.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

- Проанализировать предметную область, сформулировать ограничения предметной области.
- Провести анализ существующих СУБД.
- Спроектировать базу данных, содержащую информацию о различных университетах.
- Реализовать веб-приложение для взаимодействия с этой базой данных.
- Исследовать индексы для базовых таблиц, произвести замеры времени.

1 Аналитическая часть

В данном разделе будет проанализирована поставленная задача, формализованы данные, необходимые для ее решения, а также приведены теоретические сведения, на основании которых будет сделан выбор способа представления данных и управления ими.

В рамках поставленной задачи необходимо разработать базу данных с полной информацией о каждом университете и веб-приложение для взаимосвязи с базой данных.

Требования к веб-сайту:

1. Пользователь должен заходить и сразу видеть список возможных университетов, иметь возможность добавить ВУЗ в избранное и просмотреть список всех выбранных вузов.
2. У пользователя должна быть возможность зарегистрироваться на портале, выполнить запросы с имеющейся базой данных, просмотреть всю необходимую информацию, выйти из аккаунта.
3. При клике на иконку университета, должен быть переход на страницу с полной информацией об этом вузе.
4. Пользователь должен иметь возможность просмотреть контакты организаторов сайта и при необходимости с ними связаться.
5. На стартовой странице пользователя должна кнопка с обратной связью, возможность просмотреть все запросы и просмотреть краткую информацию о каждом университете.
6. При нажатии на конкретный ВУЗ должна быть ярко выделена кнопка «Детальнее».
7. В личном кабинете клиент может просмотреть личную информацию о себе и выйти из аккаунта.

Как университеты должны попадать на сайт?

1. Университе́т — высшее учебное заведение, где готовятся специалисты по фундаментальным и многим прикладным наукам.

2. Администратор может добавить определенный университет на сайт, если университет находится в списке высших учебных заведений Российской Федерации.
3. Администратор несет ответственность за добавление заведомо ложной информации о данном университете.
4. Администратор имеет право удалить университет с сайта без объяснения причины.

1.1 Формализация данных

В базе данных должны храниться сведения о:

- университетах;
- пользователях;
- запросах;
- удаленных университетах из базы данных;
- количестве юношей и девушек в различном университете;
- рейтинге университета на различных сайтах.

В таблице 1.1 показаны категории и сведения о данных.

Таблица 1.1 – Категории и сведения о данных

Категория	**Сведения**
Университет	Название, год постройки, год реконструкции, количество студентов, минимальная стоимость обучения, краткая информация, полная информация
Пользователь	Логин, пароль в хешированном виде, электронная почта, мобильный телефон, список избранных вузов, тип пользователя
Запросы	Описание запроса, запрос в форме SQL
Пол	ID университета, количество парней, количество девушек
Рейтинг университета	Номер университета, позиция в рейтинге THE, позиция в рейтинге QS, позиция в рейтинге RAEX

Информация об удаленном университете	ID удаленного университете, информация, когда был удален университет.
Название специальности	ID специальности, ID университета, Номер специальности, название специальности
Рейтинг специальности	ID специальности, рейтинг этой специальности

1.2 Типы пользователей

Из задачи ясно, что для просмотра всей информации и выполнении запроса нужна авторизация пользователей. Это делит пользователей на авторизованных и неавторизованных.

Для управления веб-сервисом университетов принято решение ввести роль администратора. Варианты пользователей и их возможности указаны в таблице 1.2.

Таблица 1.2 – Типы пользователей и их функционал

Тип пользователя	**Функционал**
Неавторизован-ный	Регистрация, авторизация, просмотр краткой информации об университете, просмотр всех запросов, просмотр контактной информации
Авторизованный	Просмотр полной информации об университете, выполнение всех запросов, добавление университетов в избранное, просмотр личного кабинета
Администратор	Изменение информации об университете, добавление и удаление запросов, редактирование любой информации, добавление университетов в избранное, полный доступ к БД.

Диаграмма прецедентов представлена на рисунке 1.1:

Рисунок 1.1 – Диаграмма прецедентов

Модель предметной области университетов в классической нотации Питера Чена представлена на рисунке 1.2.

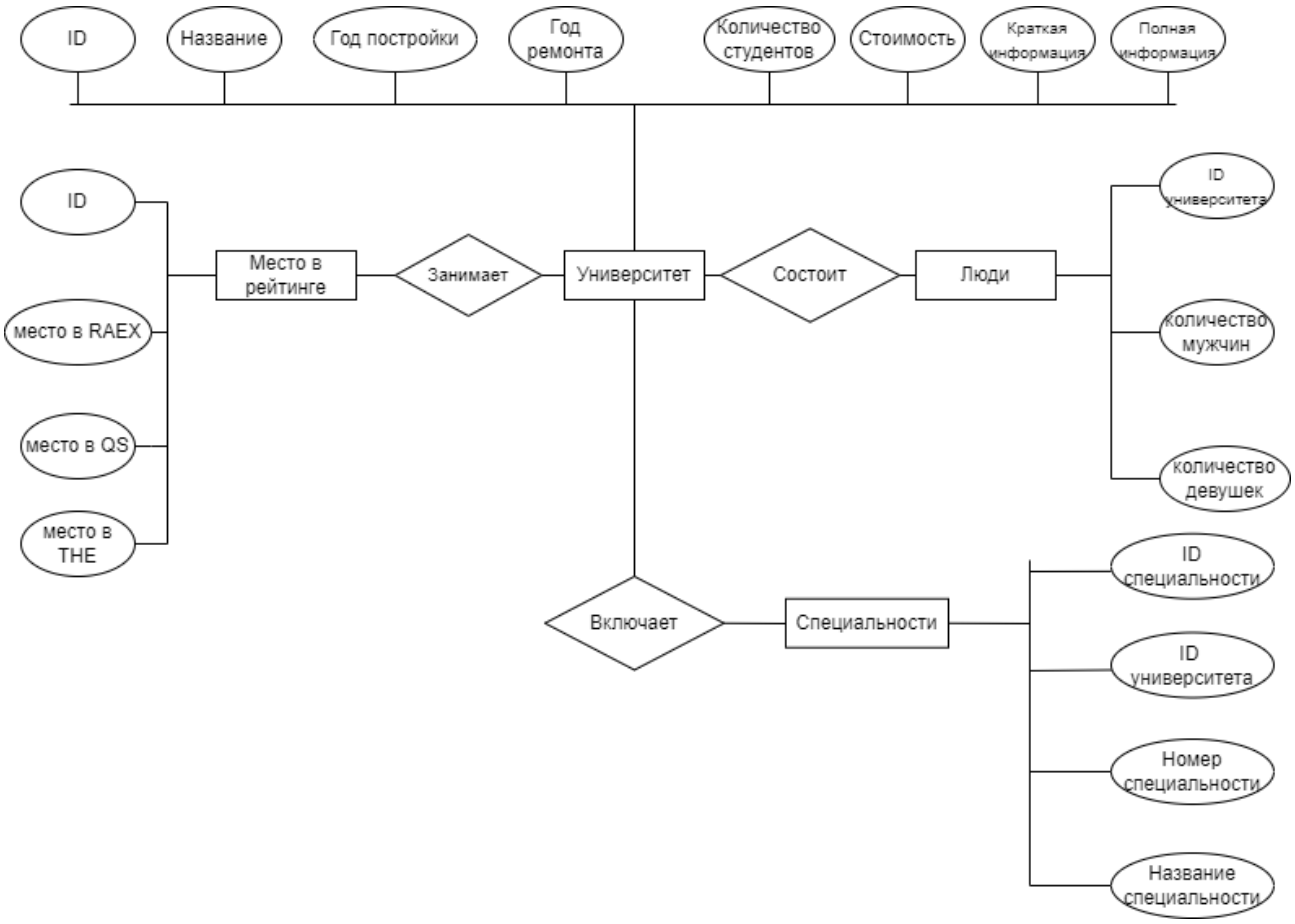


Рисунок 1.2 – Модель предметной области в нотации Питера Чена

1.3 Описание существующих СУБД

Система управления базами данных СУБД — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [8].

Главные задачи СУБД – это создание базы данных и манипуляция данными. Такая система позволяет обеспечить надежность хранения данных, их целостность и неизбыточность.

Основными функциями СУБД являются:

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

1.3.1 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. [3]

Существует 3 основных типа моделей организации данных:

– дореляционная;

– реляционная;

– постреляционная.

К реализации дореляционной модели относятся инвертированные списки(файлы), иерархические и сетевые модели данных.

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 [2]. Концептуальная схема иерархической модели данных показана на рисунке 1.3.

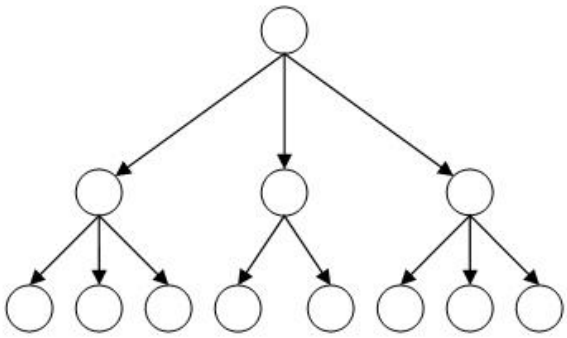


Рисунок 1.3 – Концептуальная схема иерархической модели данных

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 [3]. 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. Концептуальная схема сетевой модели данных показана на рисунке 1.4.

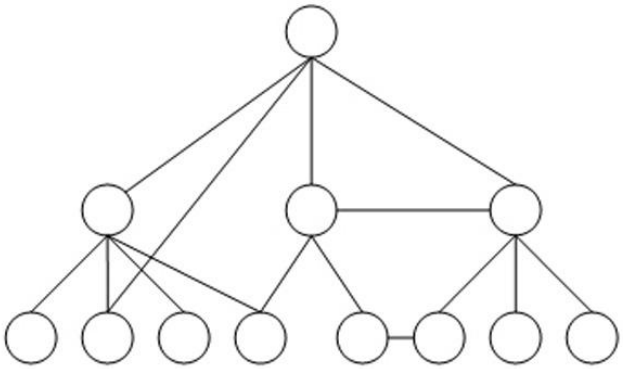


Рисунок 1.4 – Концептуальная схема сетевой модели данных

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. Концептуальная схема реляционной модели данных показана на рисунке 1.5.

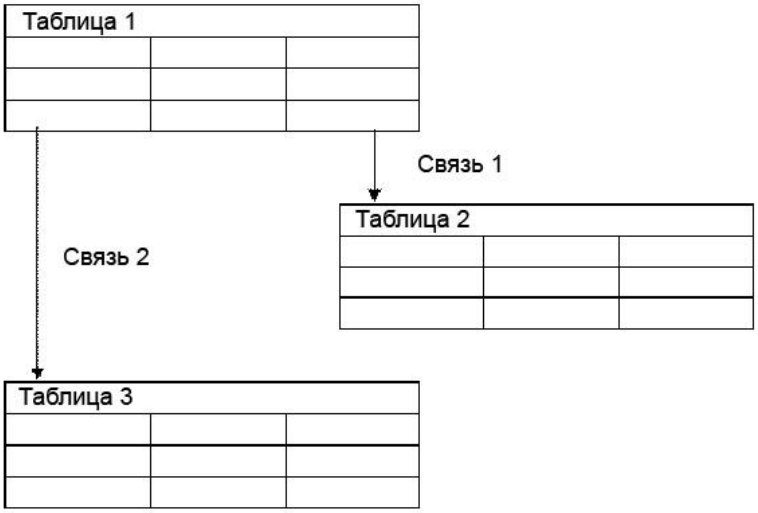


Рисунок 1.5 – Концептуальная схема реляционной модели данных

Наиболее популярными реляционными СУБД являются Oracle, Microsoft SQL Server и PostgreSQL [1].

Постреляционная модель данных является расширенной реляционной моделью, которая снимает ограничение неделимости хранящихся в записях таблиц данных [7]. При использовании постреляционной модели данных допускается создание – полей, значения которых допускают подзначения.

Главное достоинство постреляционных СУБД состоит в возможности представления совокупности связанных реляционных таблиц в виде одной постреляционной таблицы, что делает их очень удобными для сложных объектов данных, таких как мультимедийные данные, данные для географических информационных систем и др. Недостатком является сложность обеспечения целостности данных. Постреляционными СУБД являются такие системы, как uniVerse, Pick, Bubba и др. Концептуальная схема постреляционной модели данных показана на рисунке 1.6.

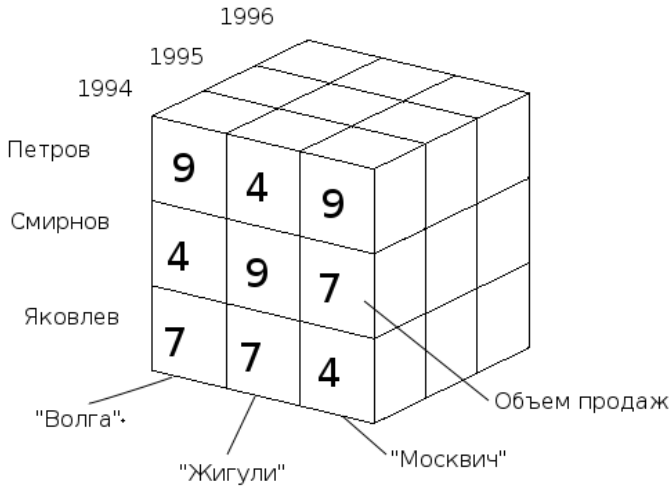


Рисунок 1.6 – Концептуальная схема постреляционной модели данных

1.4 Выводы из аналитического раздела

В данном разделе была проанализирована предметная область и сформулированы ограничения предметной области. Также были рассмотрены основные теоретические сведения о СУБД, их функциях и типах. Так как задача предполагает использование разнообразных запросов, включающее в себя выборки элементов строк и столбцов различной сложности, приоритетным свойством модели данных является гибкость, простота использования и независимость от приложения, поэтому в качестве модели организации данных была выбрана реляционная модель.

2 Конструкторская часть

В данном разделе будут рассмотрены основные структурные элементы базы данных, определена модель предметной области, сформулированы требования к программе, выделены основные компоненты веб-приложения и их функции, а также приведены схемы некоторых методов.

База данных должна хранить рассмотренные в таблице 1.1 данные. В соответствии с этой таблицей можно выделить следующие таблицы:

- таблица университетов University;
- таблица полов male_female;
- таблица пользователей Users;
- таблица запросов Query;
- таблица архивных университетов university_log;
- таблица рейтинга университета rank_university;
- таблица названия специальностей name_speciality;
- таблица рейтинга специальностей rank_speciality.

Таблица **University** должна хранить информацию об университете:

- id – уникальный идентификатор университета, PK;
- name – имя университета, varchar;
- year_construction– год постройки университета, integer;
- year_repair – год ремонта университета, integer;
- count_student – количество студентов, integer;
- min_cost – минимальная стоимость обучения, integer;
- intro – краткая информация об университете, varchar;
- about – дополнительное описание университета, varchar;

Таблица **male_female** должна хранить информацию о категориях:

- id_university - идентификатор университета, integer, FK;
- count_male – количество парней в университете, integer
- count_female – количество девушек в университете, integer

Таблица **User** должна хранить информацию о пользователях:

– id - уникальный идентификатор пользователя, integer, PK;
– username – логин пользователя, используется для авторизации, должен быть уникальным, varchar;
– password – пароль пользователя в хешированном состоянии, используется для авторизации, varchar;
– email – электронная почта пользователя, varchar
– mobile – номер мобильного телефона, varchar
– university_selected – список избранных вузов, integer[]
– type_user – тип пользователя, varchar

Таблица **university_log** должна хранить информацию об удаленных университетов:

– university_id - идентификатор удаленного университета, FK;
– description – детальная информация, varchar.

Таблица **rank_university** должна хранить информацию о рейтинге университета:

– id_university - идентификатор университета, FK;
– position_raex – позиция университета в рейтинге RAEX, integer
– position_qs – позиция университета в рейтинге QS, integer
– position_the – позиция университета в рейтинге THE, integer

Таблица **query** должна хранить информацию о запросах:

– id – идентификатор запроса, integer
– intro – описание запроса, varchar
– full_text – сам запрос в формате SQL

Таблица **name_speciality** должна хранить информацию о специальностях:

– id_speciality – идентификатор специальности, integer, PK
– id_university – идентификатор университета, FK
– number_speciality – номер специальности, varchar
– name_speciality – название специальности, varchar

Таблица **rank_speciality** должна хранить информацию о рейтинге каждой специальности

– id_speciality – идентификатор специальности, integer, FK
– rank_speciality – рейтинг специальности, integer

На рисунке 2.1 представлена диаграмма «сущность-связь» для базы данных

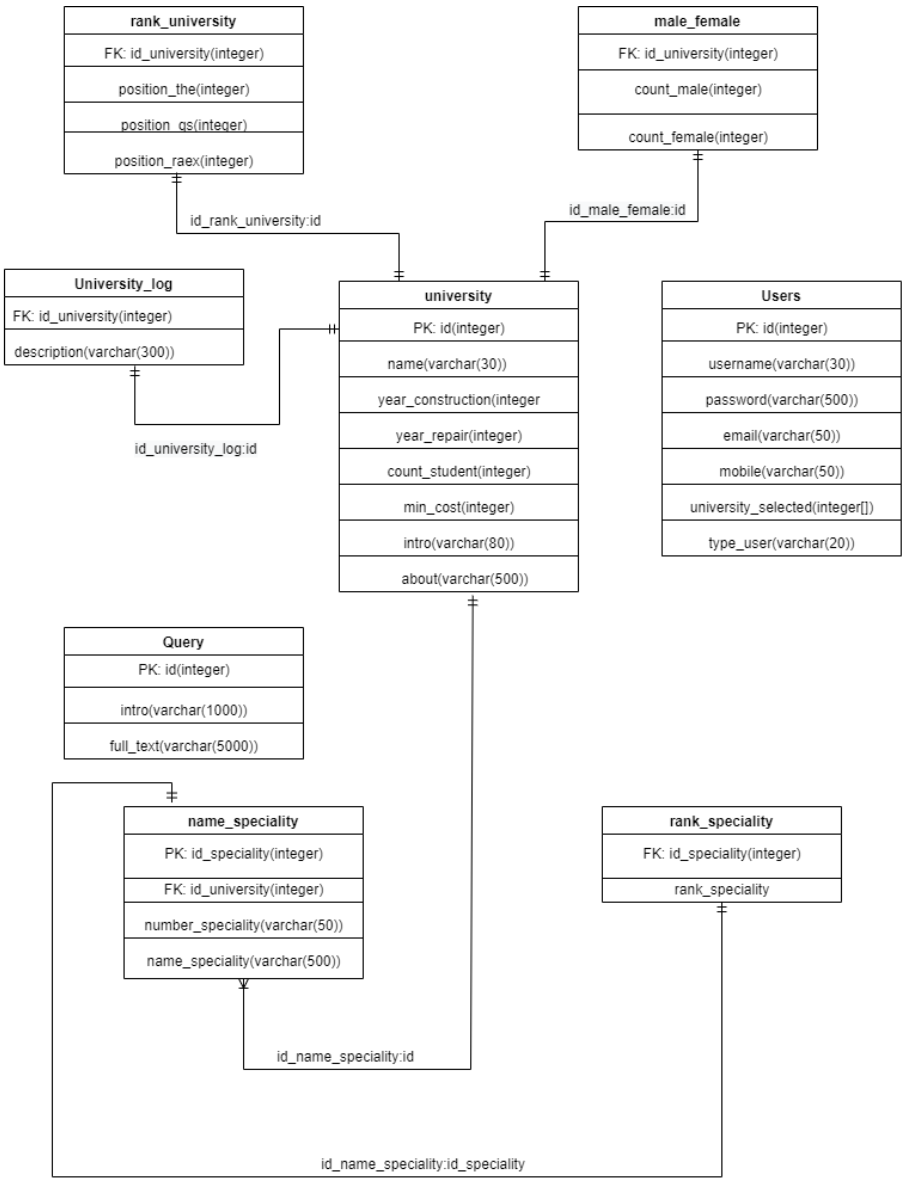


Рисунок 2.1 – Диаграмма «сущность-связь» базы данных

2.1 Требования к программе

Для успешной реализации поставленной задачи программа должна предоставлять следующие возможности:

– авторизация;
– регистрация;
– вывод списка всех университетов;
– добавление университета;
– удаление университета;
– редактирование информации об университете;
– добавление запроса;
– выполнение запроса;
– удаление запроса;
– изменение информации запроса;
– добавление университета в избранное.

Возможности незарегистрированного пользователя:

– просмотр краткой информации об университете;
– просмотр запросов.

Возможности зарегистрированного пользователя:

– просмотр полной информации об университете;
– выполнение запросов;
– добавление университета в избранное.

Возможности администратора:

– редактирование запросов;
– редактирование информации об университете;
– добавление и удаление университета;
– добавление и удаление запроса.

Для регистрации на сайте пользователь должен обязательно заполнить следующие поля: имя, email, логин, пароль с подтверждением и мобильный номер телефона. Если какое-то поле не заполнено или заполнено некорректно, то приложение предупреждает об ошибке и регистрация не происходит. При авторизации человек должен указать свой никнейм и пароль. Все пароли

хранятся в базе данных в хешированном виде. На рисунке 2.2 представлена блок-схема функции регистрации.

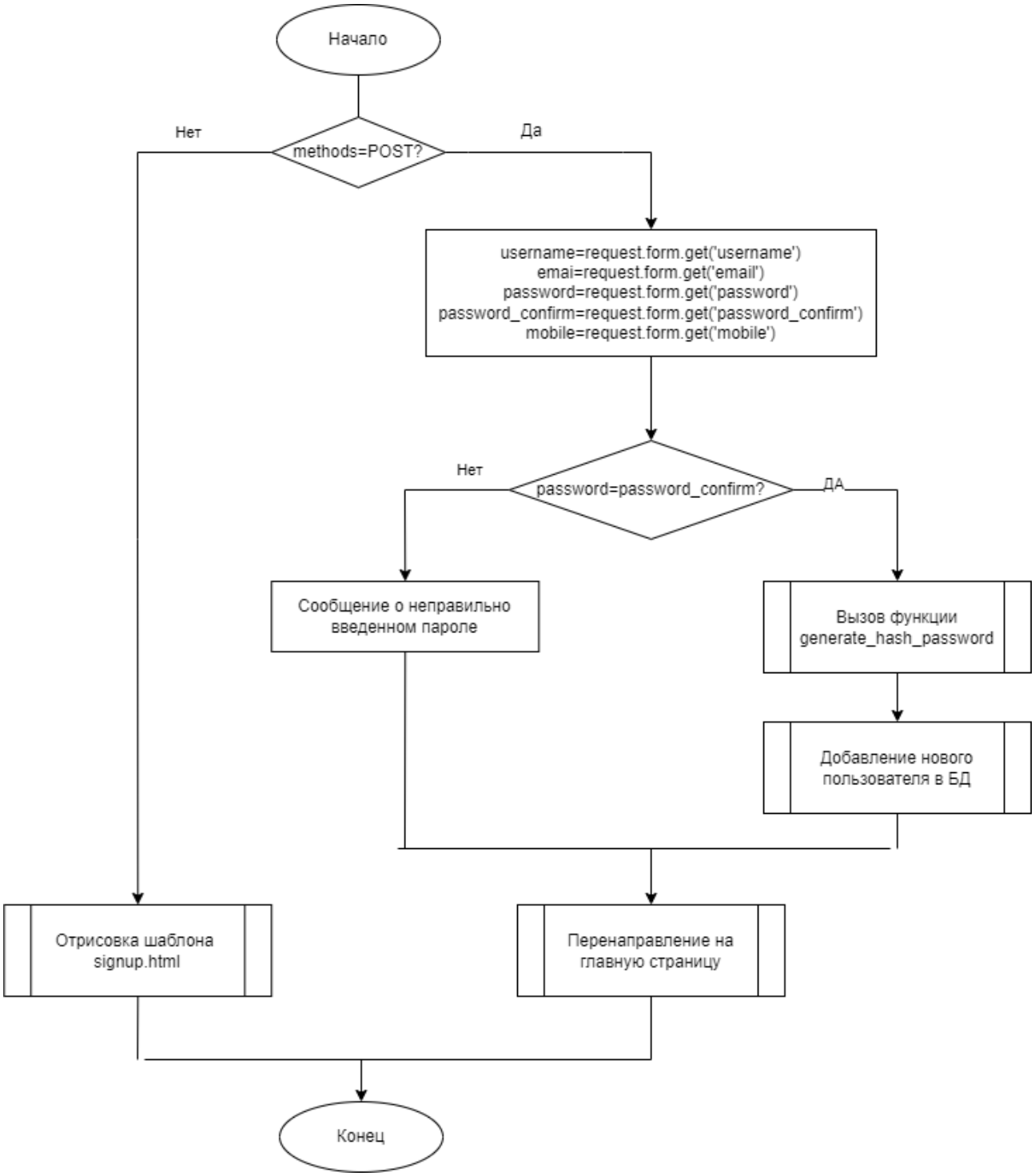


Рисунок 2.2 – Блок-схема функции регистрации

2.2 Ролевая модель

В данном проекте реализована ролевая модель на уровне базы данных с разными пользователями для безопасности и целостности базы данных. Некоторые роли дают доступ только к определенным столбцам таблицы. Ролевая модель включает в себя нескольких пользователей:

1) admin – администратор сайта, который имеет доступ ко всем таблицам;
2) registered – зарегистрированный пользователь сайта;
3) unregistered – незарегистрированный пользователь сайта.

Возможности администратора(admin):

1) select, insert, update, delete для таблицы university;
2) select, insert, update, delete для таблицы query;
3) select, insert, update, delete для таблицы male_female;
4) select, insert, update, delete для таблицы name_speciality;
5) select, insert, update, delete для таблицы rank_speciality;
6) select, insert, update, delete для таблицы rank_university;
7) select, insert, update, delete для таблицы users;
8) создание новых ролей;
9) создание базы данных.

Возможности зарегистрированного пользователя(registered):

1) select для таблицы university;
2) select для таблицы query;
3) select для таблицы male_female;
4) select для таблицы name_speciality;
5) select для таблицы rank_speciality;
6) select для таблицы rank_university.

Возможности незарегистрированного пользователя (unregistered):

1) select (id, name, intro) для таблицы university;
2) select (id, intro) для таблицы query;

Пример создания ролевой модели приведен в листинге 2.1:

```
create user unregistered with password ‘1';
create user admin with password ‘12345’ createrole
createdb;
create user registered with password ‘123';

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO
«admin»;
GRANT SELECT (id, intro) ON query TO «unregistered»;
GRANT SELECT (id, name, intro) ON university TO
"unregistered";
GRANT SELECT ON users TO «registered";
GRANT SELECT ON university TO «registered";
GRANT SELECT ON query TO «registered";
GRANT SELECT ON rank_university TO «registered";
GRANT SELECT ON male_female TO «registered»;
GRANT SELECT ON name_speciality TO «registered»;
GRANT SELECT ON rank_speciality TO «registered»;
```

Листинг 2.1 – создание ролевой модели на уровне базы данных

2.3 Выводы из конструкторского раздела

В данном разделе были рассмотрены основные структурные элементы базы данных, определена модель предметной области, сформулированы требования к программе, выделены основные компоненты веб-приложения и их функции, а также приведены схемы некоторых методов. Был приведен пример создания ролевой модели на уровне базы данных для обеспечения безопасности данных. Указаны все возможности каждого пользователя, его полномочия. Приведена блок-схема функции регистрации. Показаны все таблицы в базе данных и связи между ними.

3 Технологическая часть

В технологической части будут выбраны язык программирования, среда разработки, СУБД и вспомогательные технологии с указанием причины выбора, приведены структура классов программы и листинги кода, а также продемонстрированы возможности веб-интерфейса.

В качестве языка программирования был выбран Python [4] вместе с библиотекой Flask [5] т. к.:

- этот язык подходит для написания веб-приложений;
- удобная система поиска ошибки, отладки;
- данная библиотека имеет большую популярность, много информации в интернете, качественная документация;
- есть возможность использовать наследование, классы и т. д.;

В качестве среды разработки была выбрана «PyCharm» по следующим причинам:

- она бесплатна в пользовании студентами;
- она имеет множество удобств, которые облегчают процесс написания и отладки кода;
- PyCharm-кросс-платформенная среда разработки, что позволяет использовать ее на Mac OS.
- Интерактивная консоль для Python

В качестве СУБД была выбрана PostgreSQL [1], так как она основана на реляционной модели данных и имеет следующие достоинства:

1. поддержка массивов как типов данных;
2. возможность создания нового типа данных и поддержка различных типов данных;
3. создание триггеров и функций;
4. поддержка операционных систем семейства Unix;

5. нет ограничений на максимальный размер базы данных

3.1 Проектирование приложения

Программу можно разделить на 2 основные части:

- front-end – приложение, с которым взаимодействует пользователь, отображение данных, клиентская сторона;
- back-end – взаимодействие с базой данных: получение, удаление и изменение данных, внутреннее функционирование

Чтобы следовать принципу разделения интерфейса, каждую из частей стоит выделить в отдельный проект. Также стоит отдельно добавить проект для данных т. к. они разделяются между back-end-ом и front-end-ом. Получим следующую структуру решения:

- AccessToDB – подключение к БД, отправка запросов, получение информации из БД;
- DataStructures – классы данных, дублирующие структуру таблиц БД, нужны для ослабления связанности кода и облегченного взаимодействия между БД и приложением;
- UI – пользовательский интерфейс, через UI осуществляется взаимодействие пользователя с программой.

Проект DataStructures должен состоять из следующих компонентов:

- University – класс университета;
- Query– класс запроса;
- User – класс пользователя;
- Male_Female – класс с информацией о количестве парней и девушек в каждом университете;
- Rank_university – класс с рейтингом университетов;
- Name_speciality – класс с названиями специальностей;
- Rank_speciality – класс с рейтингом специальности.

Хранящиеся в этих классах поля дублируют соответствующие поля из БД. За исключением пароля пользователя. Пользовательский пароль не хранится в программе, он хранится в БД в хешированном состоянии.

Проект AccessToDB должен состоять следующих компонентов:

1. Connector - класс коннектора, который напрямую должен осуществлять подключение к базе данных, отправку различных запросов. Все взаимодействие с БД проходит через коннектор.
2. Функции работы с данными, а именно:
   - получение;
   - изменение;
   - удаление;
   - проверка данных.

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

- Unauthorized – вызов функции, если нет доступа у пользователя к данной странице;
- Login_required – проверка зарегистрирован пользователь или нет;
- Login – функция для авторизации пользователей;
- Logout – функция для выхода пользователя;
- Load_user – загрузка пользователя;
- University_detail – получение полной информации об университете;
- University_update – изменение данных об университете;
- Query_run – выполнение запроса;
- Query_update – изменение данных запроса;
- Register – функция для регистрации.

3.2 Структура и состав классов

В этом разделе будут рассмотрена структура и состав классов.

University – класс университета с основной информацией.

Users – класс пользователя.

Query - класс запроса.

Male_female – класс, содержащий информацию о количестве парней и девушек в каждом университете.

Rank_university – класс, содержащий информацию о позиции каждого вуза в рейтинге.

Rank_speciality – класс, содержащий информацию о рейтинге каждой специальности.

Name_speciality – класс, содержащий название каждой специальности.

Структура каждого класса соответствует данным таблиц базы данных, описанным в разделе 2.1, и представлена в листингах А.1–А.7 (см. приложение А).

3.3 Взаимодействие с базой данных

Класс SQLAlchemy позволяет работать с движком базы данных PostgreSQL, проинициализировав объект этого класса и задав определенные параметры, например, URI базы данных, отвечающий за доступ к ней, в конфигурации веб-приложения.

Для запросов к базе данных и любых других операций над ней используются классы Session и Query, предоставляемые SQLAlchemy, а также модуль Python - psycopg2 [5], имеющий курсоры.

Пример использования SQLAlchemy и psycopg2 приведен в листингах А.8–А.11 (см. приложение А).

3.4 Триггер

Для отслеживания удаленных университетов создана специальная таблица university_log, в которой хранится информация об архивных университетах. Триггер delete_university и триггерная функция After_delete представлена
в листингах 3.1 и 3.2:

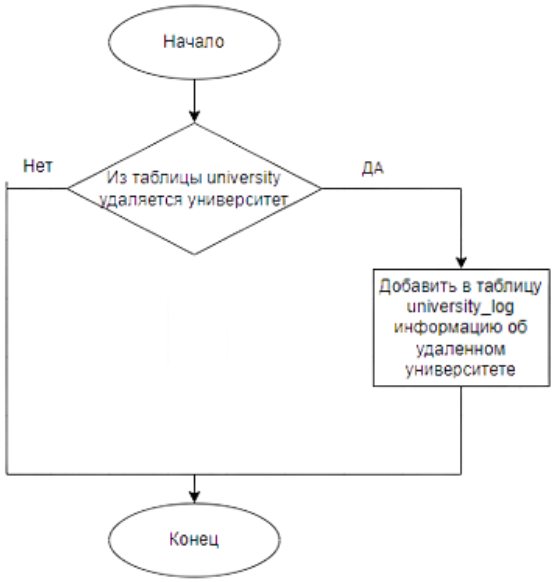
```
CREATE TRIGGER after_delete
AFTER DELETE ON university
FOR EACH ROW
EXECUTE PROCEDURE after_delete();
```

Листинг 3.1 - создание триггера

```
CREATE OR REPLACE FUNCTION after_delete() RETURNS TRIGGER
AS $$
BEGIN
INSERT into university_log Values(OLD.ID, CONCAT('Удален
университет ', OLD.NAME, 'Краткое описание: ', OLD.INRO,
'-> Удален', NOW()));
END;
$$ LANGUAGE plpgsql;
```

Листинг 3.2 - триггерная функция

Блок-схема работы функции триггера представлена в листинге 3.3:



Листинг 3.3 – Пример создания и удаления индексов

Программа состоит из 4 директорий, созданных по умолчанию FLASK:

– static – хранение стилей CSS;

– templates – хранение HTML форм для пользовательского интерфейса;

– venv – директория, созданная по умолчанию и хранящая конфигурацию для создания и работоспособности сервера.

Внутри проектов расположены модули, которые отвечают за работоспособность проекта.

App.py – основной модуль, содержащий Backend.

3.5 Графический интерфейс пользователя

Пользовательский интерфейс реализуется клиентской частью – браузером - посредством языка разметки HTML и формального языка описания внешнего вида веб-страницы CSS и представляет собой совокупность веб-страниц.

Для размещения и реализации различных элементов веб-интерфейса, включая иконки, стилизованное оформление кнопок и ссылок, был использован свободный набор инструментов для создания веб-приложений – Bootstrap [6].

На рисунке 3.1 представлена главная страница сайта администратора.

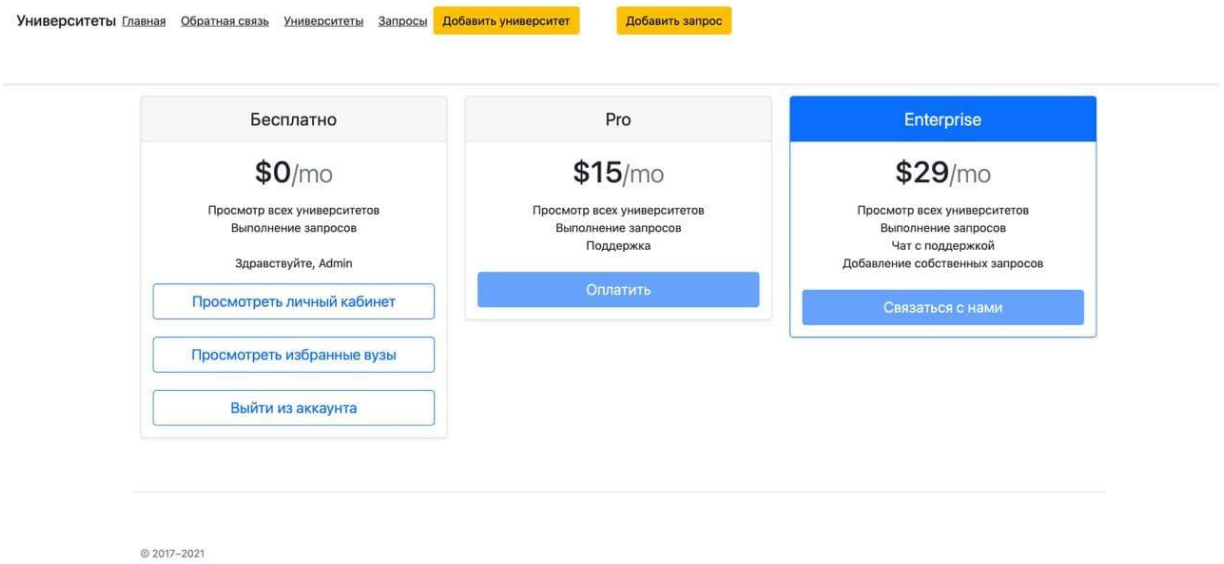


Рисунок 3.1 – Главная страница администратора

На рисунке 3.2 представлена страница авторизации

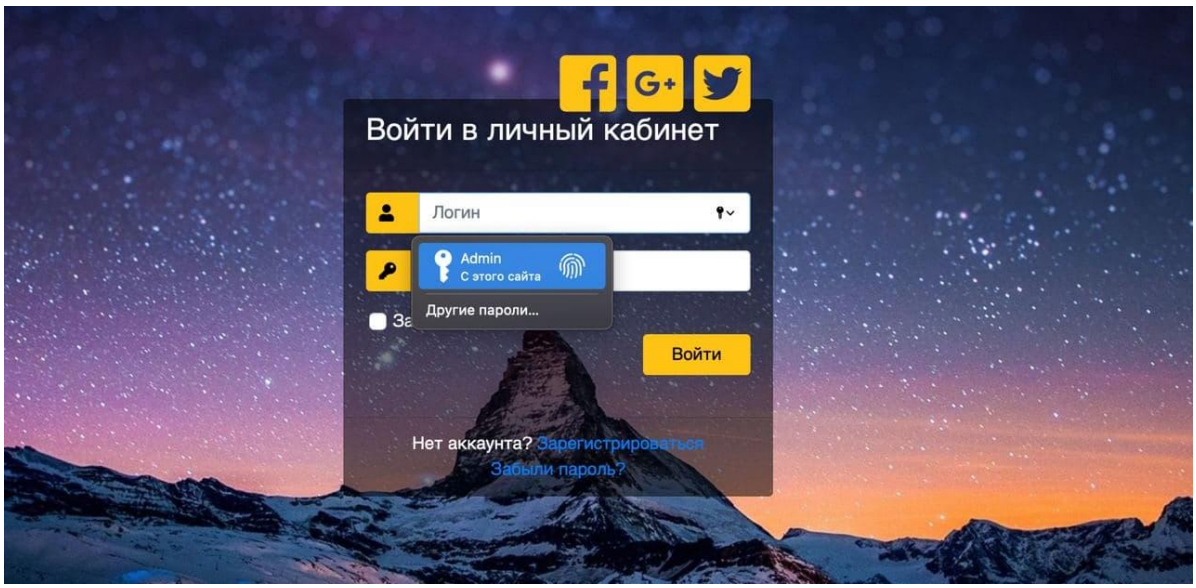


Рисунок 3.2 – Авторизация пользователя

На рисунке 3.3 показана страница для регистрации новых пользователей, которые хотят получить доступ ко всем возможностям сайта

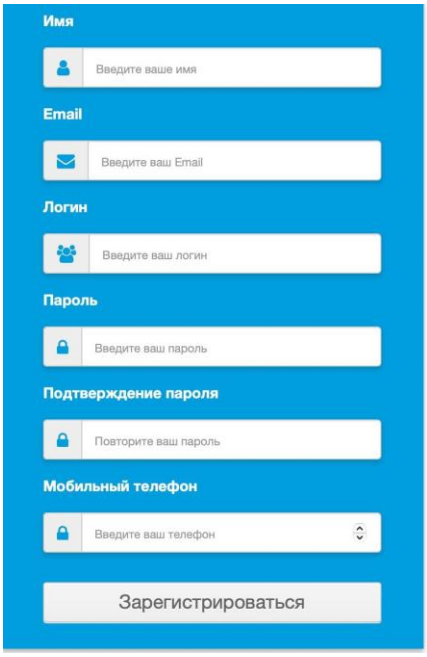


Рисунок 3.3 – Страница с регистрацией новых пользователей

На рисунке 3.4 представлена контактная информация о сайте, местоположению офиса и формой обратной связи.

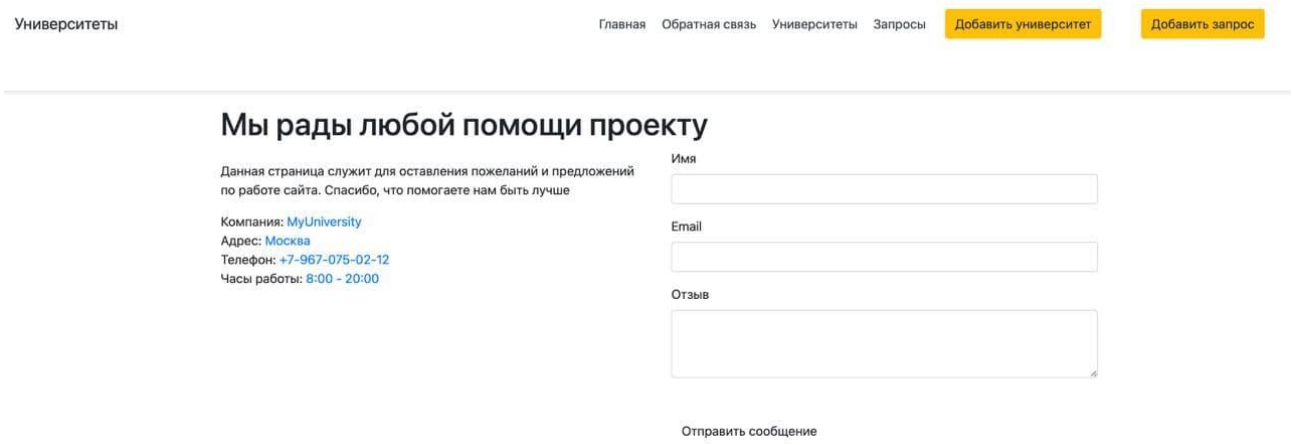


Рисунок 3.4 – Форма обратной связи

На рисунке 3.5 показана форма для добавления университета (доступна только администратору)

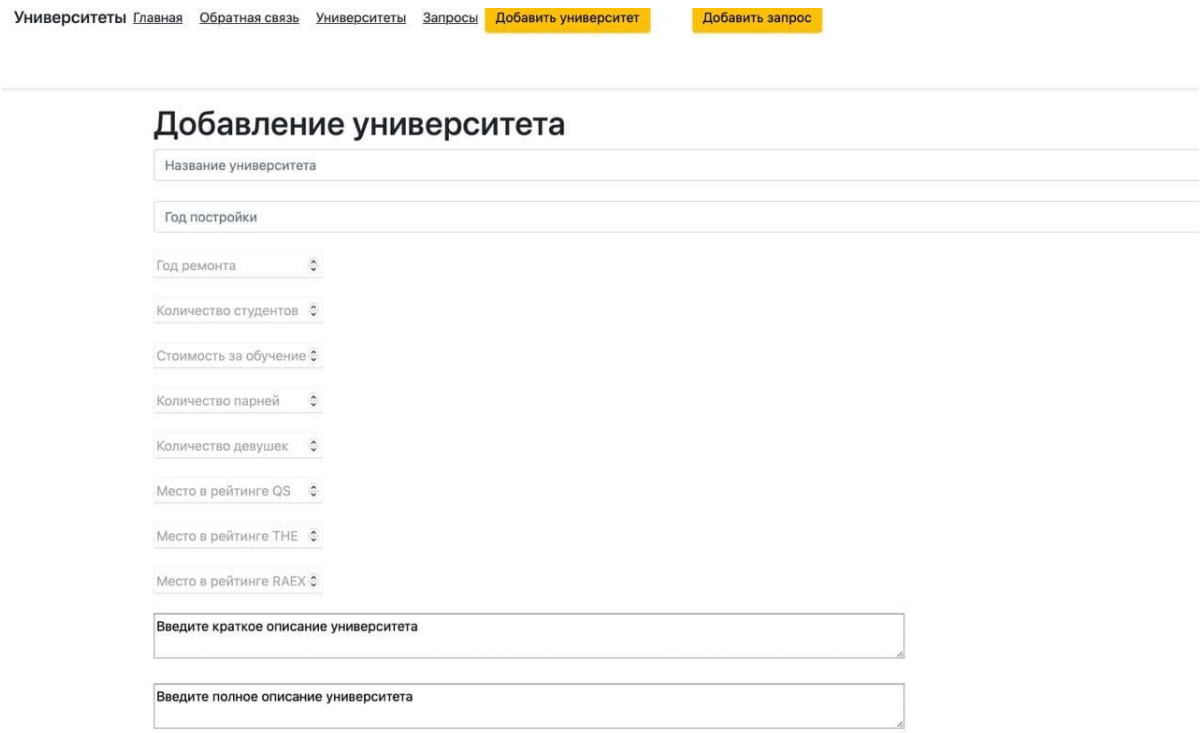


Рисунок 3.5 – Форма добавления университета

На рисунке 3.6 показан список всех университетов, содержащихся в базе данных.

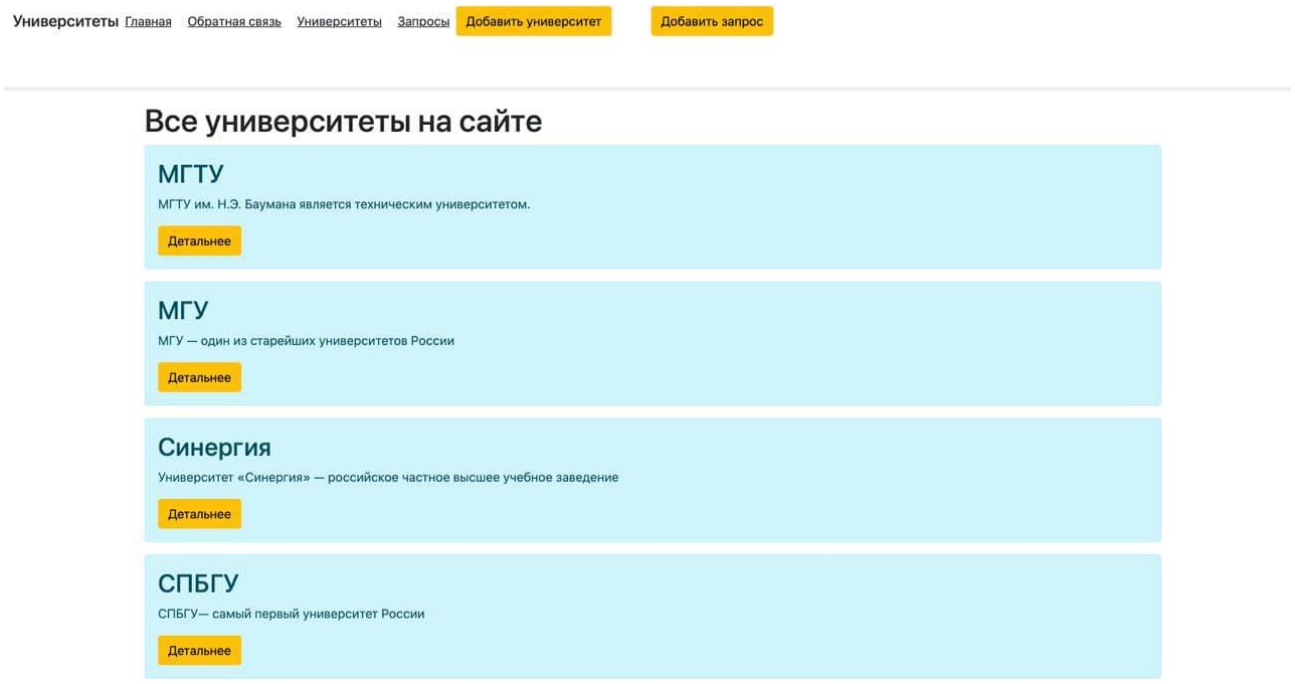


Рисунок 3.6 – Список всех университетов на сайте

На рисунке 3.7 показаны все запросы с возможностью удаления и редактирования (доступно только администратору).

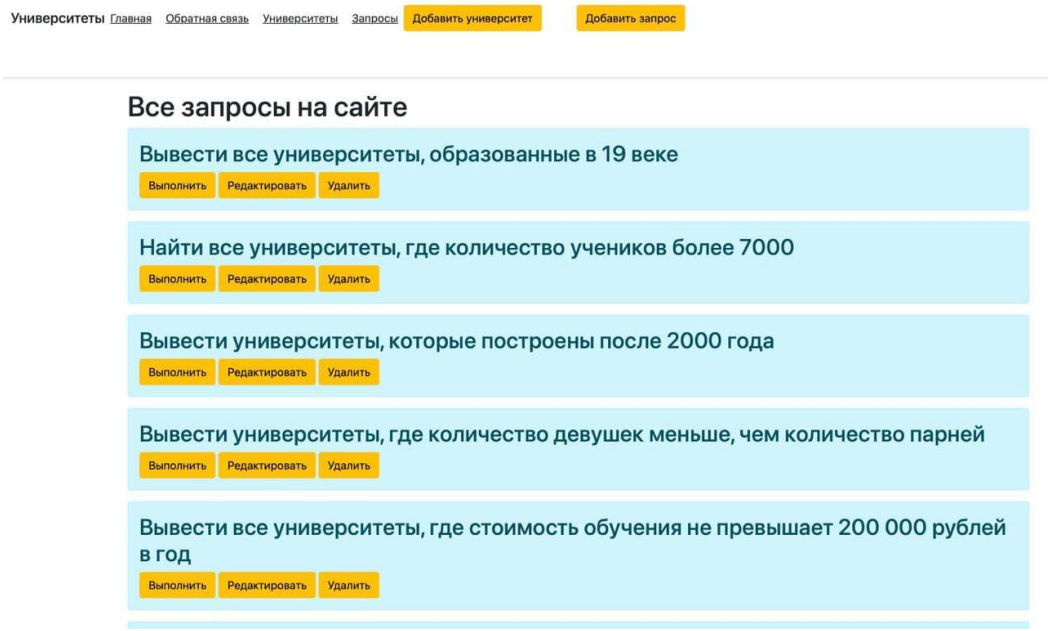


Рисунок 3.7 – Список запросов

На рисунке 3.8 показана полная информация об университете, с возможностью добавить в избранное (доступно только зарегистрированным

пользователям) редактированием и удалением информации (доступ только у администратора).

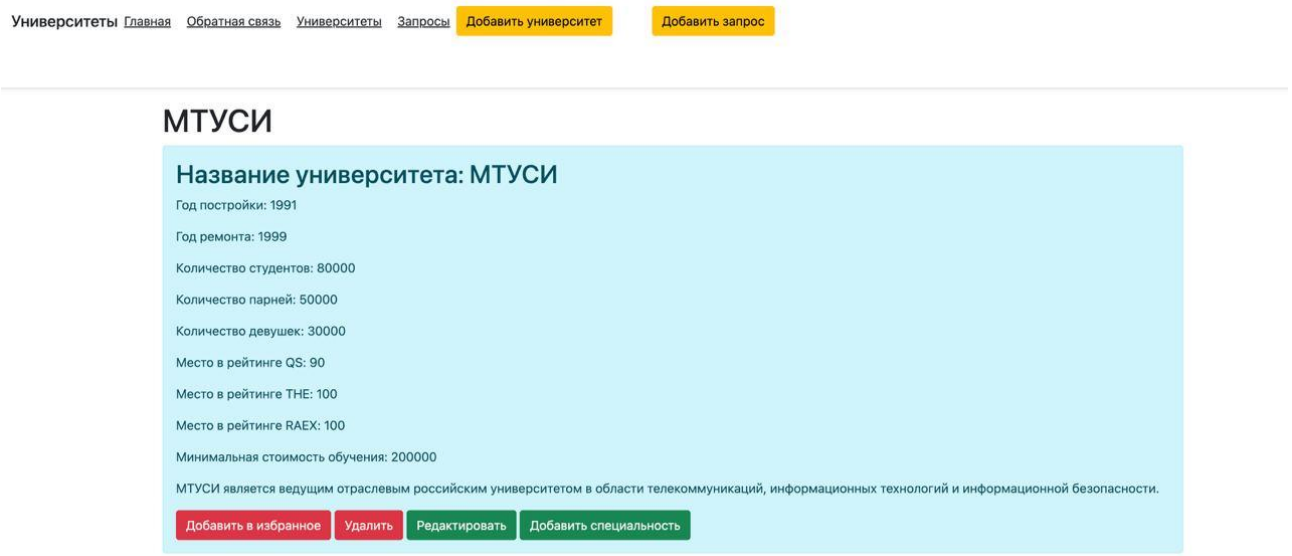


Рисунок 3.8 – Полная информация об университете

3.6 Выводы из технологического раздела

В данном разделе была выбран язык программирования Python и среда разработки PyCharm, так же выбрана СУБД – PostgreSQL. Для создания пользовательского интерфейса использовалась библиотека Flask и Bootstrap5. Были рассмотрены структура и состав реализованных классов, предоставлены сведения о модулях программы и об ее интерфейсе. Приведены примеры веб-интерфейса, реализация различных возможностей веб-сайта.

4 Исследовательская часть

В данном разделе будет исследована скорость доступа к базе данных с использованием индексов для таблицы. Так же будет проведено сравнение оптимизированной таблицы, где используются индексы с таблицей, которая создается по умолчанию.

Индекс – это упорядоченный указатель на записи в таблице [2]. Указатель означает, что индекс содержит значения одного или нескольких полей в таблице и адреса данных, на которых располагаются эти значения.

Основная функция индексов - обеспечивать быстрый поиск записи в таблице [2]. Индекс не является частью таблицы — это отдельный объект, связанный с таблицей и другими объектами базы данных.

4.1 Создание и удаление индексов

Для создания индексов в таблице используется команда CREATE INDEX.

Необходимо указать имя индекса и список атрибутов, по которым он будет строиться, по желанию есть возможность указать тип индекса.

Примеры создания и удаления индексов для таблицы продемонстрирован в листинге 4.1:

```
CREATE INDEX  my_index on test_index(age)
DROP INDEX  my_index
CREATE       INDEX       index_year_construction       on
university(year_construction)
```

Листинг 4.1 – Пример создания и удаления индексов

С помощью индексов можно ускорить работу, а по мере увеличения количества записей в таблице преимущества индексов становятся очевиднее. Необходимо обратить внимание, что индексам необходимо место для хранения и время на обновление, поэтому индексы лучше использовать в базах данных, где происходят плановые обновления, а не постоянные.

Для замеров времени использовался компьютер со следующими параметрами:

1. ОЗУ: 16Гб;
2. чип: Apple M1 PRO;
3. общее количество ядер:10 (8 производительности и 2 эффективности).

4.2 Результаты замеров времени

Для тестирования возможностей индексов был произведен замер времени с использованием библиотеки Time для Python. Результаты эксперимента представлены в таблице 4.1. В таблице находится 1027 элементов. Каждый поиск был произведен 10 000 раз. Так же был произведен замер времени для вставки 20 000 новых элементов.

Поиск: “Select * from university where year_construction = 1020;”

Таблица 4.1 – Результаты замеров времени

	Результат времени, с.	Результат времени, с.
	Поиск	Вставка 20 000 новых элементов
С индексами	0.83 с.	2.27 с.
Без индексов	1.17 с.	2.06 с.

Исходя из полученных данных, можно сделать вывод, что поиск по таблице с индексами примерно быстрее на 40%. Однако при добавлении данных в таблицу тратится время на обновление индексов, тем самым, добавление занимает больше времени.

4.3 Выводы из экспериментальной части

В данном разделе была исследована работа с индексами. Так же были продемонстрированы команды для создания и удаления индексов из базы данных, произведены замеры времени для поиска информации в таблицах с индексами и без индексов. В таблице с правильно выбранными индексами поиск идет быстрее примерно на 40%, но при добавлении данных в таблицу компьютеру необходимо время на обновление таблицы индексов.

Заключение

Во время выполнения данного проекта были рассмотрены существующие виды СУБД, описана структура базы данных и приложения. Была разработана база данных университетов и веб-приложение для взаимодействия с этой базой данных.

Программа реализована таким образом, что пользователь может получать полную информацию об университетах, регистрироваться в системе, авторизовываться, выполнять запросы, просматривать личный кабинет, добавлять университеты в избранное. Администратор может редактировать информацию, добавлять и удалять университеты, добавлять и удалять запросы. Незарегистрированный пользователь может смотреть контакты создателей сайта и краткую информацию о каждом университете.

В ходе выполнения поставленной задачи были изучены возможности языка Python и библиотекой Flask. Получен опыт работы с PostgreSQL и PGAdmin4 получены знания в области баз данных.

Реализованы следующие задачи:

1) проанализирована предметная область, сформулированы ограничения предметной области;
2) проведен анализ существующих СУБД;
3) спроектирована база данных, содержащая информацию о различных университетах;
4) реализовано веб-приложение для взаимодействия с этой базой данных;
5) исследованы индексы для базовых таблиц, произведены замеры времени.

Список использованных источников

1. Postgresql [Электронный ресурс] Режим доступа: https://www.postgresql.org
2. Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: «Вильямс», 2006.
3. Бородина А. И. Реляционная модель данных [Электронный ресурс]
4. Лучано Рамальо Python. К вершинам мастерства. – М.: ДМК Пресс, 2016. – 768 с.
5. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. – 2-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2019. – 161 с.
6. Bootstrap [Электронный ресурс] Режим доступа: https://getbootstrap.com
7. Бородина А. И. Постреляционная, многомерная и объектно-ориентированная модели данных [Электронный ресурс]
8. Кузнецов С. Д. Система управления базами данных [Электронный ресурс]

Приложение А

Листинг А.1– Класс пользователя

```
class Users (db.Model, UserMixin):
    id = db.Column(db.INTEGER, primary_key=True)
    username = db.Column(db.String(128), unique=True)
    password = db.Column(db.String(500))
    email = db.Column(db.String(255), unique=True)
    mobile = db.Column(db.String(255), unique=True)
    university_selected = db.Column(db.String(70))
    type_user = db.Column(db.String(20))
    def __repr__(self):
        return '<User %r' % self.id
```

Листинг А.2 – Класс рейтинг университетов

```
class rank_university(db.Model):
    id_university=db.Column(db.INTEGER, primary_key=True)
    position_raex = db.Column(db.INTEGER)
    position_qs = db.Column(db.INTEGER)
    position_the = db.Column(db.INTEGER)
    def __repr__(self):
        return '<Rank_university %r' % self.id
```

Листинг А.3 – Класс для количества парней и девушек

```
class MaleFemale(db.Model):
    id_university=db.Column(db.INTEGER, primary_key=True)
    count_male = db.Column(db.INTEGER)
    count_female = db.Column(db.INTEGER)
    def __repr__(self):
        return '<Male_Female %r' % self.id
```

Листинг А.4 – Класс университета

```
class University (db.Model):
    id = db.Column(db.INTEGER, primary_key=True)
    name = db.Column(db.String(30))
    year_construction = db.Column(db.INTEGER)
    year_repair = db.Column(db.INTEGER)
    count_student = db.Column(db.INTEGER)
    min_cost = db.Column(db.INTEGER)
    intro = db.Column(db.String(180))
    about = db.Column(db.String(500))
    def __repr__(self):
        return '<University %r' % self.id
```

Листинг А.5 – Класс запроса

```
class Query (db.Model, UserMixin):
    id = db.Column(db.INTEGER, primary_key=True)
    intro = db.Column(db.String(400))
    full_text = db.Column(db.String(400))
    def __repr__(self):
        return '<Admin %r' % self.id
```

Листинг А.6 – Класс названия специальности

```
    class Name_speciality(db.Model, UserMixin):
        id_speciality         =         db.Column(db.INTEGER,
primary_key=True)
        id_university = db.Column(db.INTEGER)
        number_speciality = db.Column(db.String(1000))
        name_speciality = db.Column(db.String(1000))
        def __repr__(self):
            return '<Admin %r' % self.id_speciality
```

Листинг А.7 – Класс с рейтингами специальностей

```
    class Rank_speciality(db.Model, UserMixin):
    id_speciality = db.Column(db.INTEGER, primary_key=True)
    rank_speciality = db.Column(db.INTEGER)
        def __repr__(self):
            return '<Admin %r' % self.id_speciality
```

Листинг А.8 – Вызывается при регистрации

```
    @app.route('/register', methods=['GET', 'POST'])
    def register():
        if request.method == "POST":
            name = request.form.get('name')
            username = request.form.get('username')
            email = request.form.get('email')
            password = request.form.get('password')
            password_confirm = request.form.get('confirm')
            mobile = request.form.get('mobile')
            if password_confirm != password or not email or
not name or not mobile:
                return render_template("Error")
            hash_password = generate_password_hash(password)
                new_user=
         Users(id=Users.query.order_by(Users.id).all().__len_
         _()+1,   username=username,   password=hash_password,
         email=email, mobile=mobile, type_user = "user")
            db.session.add(new_user)
            db.session.commit()
            return redirect('/login')
        else:
            return render_template("signup.html")
```

Листинг А.9 – Вызывается при попытке авторизоваться

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == "POST":
        username = request.form.get('username')
        password = request.form.get('password')
        try:
            if login and password:

user=Users.query.filter_by(username=username).first()
               if user and password:
         check_password_hash(user.password, password):
                    login_user(user)
                return redirect('/')
        except:
            return render_template("Login.html")
    else:
        return render_template("Login.html")
```

Листинг А.10 – Выполнение запроса

```
@app.route('/query/<int:id>/run')
@login_required
def query_run(id):
    query = Query.query.get(id)
    text = query.full_text
    cursor.execute(text)
    records = cursor.fetchall()
    need_universities = []
    for record in records:
        id = record[0]
        university = University.query.get(id)
        need_universities.append(university)
    return                    render_template("posts.html",
universities=need_universities,   my_title="Университеты,
удовлетворяющие запросу" + "\n '" + query.intro + "'")
```

Листинг А.11 – Соединение с БД

```
connection = psycopg2.connect(user="serega",
                              password="1234",
                              host="localhost",
                              port="5432",
                              database="postgres")
cursor = connection.cursor()
```