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 КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка клиент-серверного приложения для ведения портфолио по различным профессиям

Студент ИУ7–63Б (группа) _____ (Подпись, дата) Ф.М. Набиев (И.О. Фамилия)

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

2020 г.

РЕФЕРАТ

Отчёт содержит 29 страниц, 7 рисунков, 2 таблиц, 4 источников.

Ключевые слова: базы данных, PostgreSQL, Web-приложение, портфолио, блог.

Целью настоящей курсовой работы является создание клиент-серверного приложения, которое позволяет пользователям регистрироваться, авторизироваться, и создавать портфолио для различных профессий.

В итоге разработан программный продукт, полностью соответствующий поставленному техническому заданию.

ВВЕДЕНИЕ

Конкуренция на рынке труда среди соискателей приводит к необходимости владения претендентами навыком представления своих лучших черт, соответствующих целевой профессии. Очевидно, что данное умение человека испытывается на собеседовании, а так же на предшествующем ему этапе создания резюме.

Проблема написания резюме нетривиальна и обладает значительным количеством факторов, которые критически важно учесть. Одним из таких факторов является лаконичность. Чересчур объёмный и сложный текст будет отталкивать работодателя. Таким образом, резюме представляет собой лишённый деталей набор фактов о потенциальном работнике. Значит ли это, что детали как таковые не нужны для представления претендента перед его возможным нанимателем? Конечно, нет, ведь именно детали дают полную картину человека, выражают степень его компетентности, увлеченности работой и прочих характеристик.

Как было установлено, резюме должно быть лаконичным. Но тогда возникает вопрос: как соискатель может рассказать о себе больше? Люди по-разному решают эту проблему: некоторые ведут посвященный своей специальности блог, кто-то пишет статьи, а иные — создают сайты-визитки. Для достижения максимальной презентации работником его навыков можно объединить все эти способы и резюме в одном месте, а так же предоставить удобные инструменты для управления полученной системы.

Таким образом, **целью** данной курсовой работы создание клиент-серверного web-приложения, которое позволяет пользователям регистрироваться, авторизироваться, и создавать портфолио для различных профессий.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать имеющиеся на рынке решения;
- провести сравнительный анализ существующих моделей БД и СУБД;
- спроектировать базу данных, обеспечивающую структурное хранение данных;

- с использованием выбранной СУБД реализовать спроектированную базу данных;
- реализовать клиент-серверное web-приложение для взаимодействия с имеющейся базой данных;
- проверить работоспособность разработанного приложения.

СОДЕРЖАНИЕ

1 Аналитический раздел

В данном разделе расмматриваются имеющиейся на рынке решения, производится анализ существующих моделей БД и СУБД.

1.1 Постановка задачи

В соответствии с техническим заданием ну курсовую работу требуется разработать клиент-серверное web-приложение, через которое осуществляется доступ чтения, добавления и обновления к находящимся в базе данных данным пользователей, их портфолио, профессиям, записям и комментариям. Это приложение должно обладать следующей функциональностью:

- создание, выборка, изменение и удаление записей с данными пользователя (имя пользователя, пароль, имя, фамилия и т.д.);
- создание, выборка, изменение и удаление записей с данными о професиях;
- создание, выборка, изменение и удаление записей с данными о портфолио пользователй;
- создание, выборка, изменение и удаление записей с данными о записях пользователей;
- создание, выборка, изменение и удаление записей с данными о комментариях пользователей;
- поиск пользователей по их имени пользователя.

1.2 Анализ существующих решений

На сегодняшний день на рынке имеются различные продукты, решающие поставленную цель в разной степени. Для их рассмотрения и анализа введём следующие критерии:

1) возможность создания и публикации резюме/портфолио (далее просто портфолио);
2) возможность ведения блога/написания статей в контексте определённого портфолио;
3) возможность комментирования записей;
4) возможность поиска соискателей;
5) доступность сервиса (характеристика легкости в получении пользователями доступа к функциональности).

1.2.1 LinkedIn

LinkedIn — это социальная сеть для поиска и установления деловых контактов. Она позволяет публиковать профессиональные резюме, осуществлять поиск работы, рекомендовать людей и быть рекомендованным, публиковать вакансии и создавать группы по интересам.

Для доступа к LinkedIn необходимо пройти регистрацию, кроме того данный сервис заблокирован на территории Российской Федерации.

1.2.2 Medium

Medium — это платформа для социальной журналистики. Основными её возможностями является создания и публикация статей на различные темы, комментирование этих статей и различное их продвижение. Просмотр статей на этом ресурсе регистрацией не ограничивается.

1.2.3 Хабр

Хабр — это веб-сайт в системе тематических коллективных блогов. Хабр предоставляет доступ для создания, публикации новостей, аналитических статей, мыслей относительно различных технологиях и инфоповодами, связанными с ними.

Доступ на чтение открыт для всех без исключения, но возможность публикации ограничена не только регистрацией, но и рядом дополнительных требований, служащих гарантом качества публикуемых статей.

1.2.4 Вывод

По итогу анализа представленных решений, имеем следующую таблицу:

Таблица 1.1 – Анализ существующих решений

Номер критерия	1	2	3	4	5
LinkedIn	+	–	–	+	–
Medium	–	+	+	+	+
Хабр	–	+	+	+	+/–

Отсюда можно сделать вывод, что на рынке отсутствует продукт в полной мере достигающий поставленной цели.

1.3 Анализ и выбор модели БД

Модель данных — это совокупность структур данных и операций их обработки. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними[1]. Модель данных представляет сочетание следующих компонентов:

- структурная часть (набор правил, по которым может быть построена база данных);
- управляющая часть (определяет типы дополнительных операций над данными).

Различают следующие основные четыре модели данных[1]:

- иерархическая;

- сетевая;
- объектно-ориентированная;
- реляционная.

Рассмотрим эти модели подробнее.

1.3.1 Иерархическая модель

Иерархические базы данных — это базы данных с древовидной структурой с дугами-связями и узлами-элементами данных. Данная структура предполагает жесткое подчинение между данными в системами. Основной недостаток данной модели — подобная структура не удовлетворяет требованиям многих реальных задач.

1.3.2 Сетевая модель

В сетевых базах данных наряду с вертикальными связями допустимы и горизонтальные связи. Однако в данной модели имеют место многие недостатки иерархической модели и главный из них — необходимость четко определять на физическом уровне связи данных и столь же четко следовать этой структуре связей при запросах к базе.

1.3.3 Объектно-ориентированная модель

Новые области применения средств вычислительной техники, такие автоматизированное проектирование, потребовали от баз данных способности хранить и обрабатывать новые объекты: текст, аудио информацию, видео информацию и прочее. Основные трудности объектно-ориентированного моделирования данных проистекают из того, что такого развитого математического аппарата, на который могла бы опираться общая объектно-ориентированная модель данных, не существует.

1.3.4 Реляционная

Реляционная модель появилась вследствие стремления сделать базу данных максимально гибкой. Данная модель представила простой и эффективный механизм поддержания связей данных. Все данные в модели представляются только в виде таблиц.

1.3.5 Выбор

Реляционная модель данных имеет ряд преимуществ в сравнении с остальными рассмотренными моделями. Например, гибкость и целостность данных. Именно поэтому для реализации данной курсовой работы была выбрана реляционная модель.

1.4 Анализ и выбор СУБД

Система управления базами данных — это комплекс программно-языковых средств, позволяющих создавать базы данных и управлять ими. Среди самых популярных реляционных СУБД можно выделить[2]:

- MySQL;
- MS SQL Server;
- SQLite
- PostgreSQL;

Для выбора СУБД определим следующие критерии:

1. свободное распространение;
2. полная поддержка SQL;
3. наличие ORM;
4. лёгкость развёртывания;

1.4.1 MySQL

MySQL — это СУБД с открытым исходным кодом. Является одной из самой популярной реляционной СУБД. Основными достоинствами данной системы управления базами данных являются:

- простота при работе с ней;
- безопасность;
- масштабируем ость;
- поддержка большей части функционала SQL;
- скорость.

При этом у MySQL есть существенные недостатки:

- фрагментарное использование SQL;
- подверженность DDos-атакам;
- Платная техническая поддержка.

1.4.2 MS SQL Server

MS SQL Server — это СУБД, разрабатываемая компанией Microsoft. Достоинства:

- высокая масштабируемость;
- синхронизация с другими продуктами Microsoft;
- хорошая защита данных.

Недостатки:

- платная схема распространения;
- повышенное потребление ресурсов;

1.4.3 SQLite

SQLite — компактная встраиваемая СУБД. Исходный код библиотеки передан в общественное достояние.

Достоинства:

- файловая структура — вся база данных состоит из одного файла;
- масштабируемость.

Недостатки:

- фрагментарное использование;
- отсутствие системы пользователей;

1.4.4 PostgreSQL

PostgreSQL — самая продвинутая современная СУБД с открытым исходным кодом. Данное ПО обладаем следующими преимуществами:

- возможность использовать множество дополнений помимо мощного встроенного SQL;
- существование большого сообщества пользователей, благодаря которым можно найти ответ на любой вопрос и решить любую проблему;
- поддержка множеством организаций как одного из самых перспективных
- поддержка json, csv.

Недостатки:

- повышенный расход ресурсов;

1.4.5 Выбор

Таблица 1.2 – Анализ СУБД

Номер критерия	1	2	3	4
MySQL	+	+/–	+	+/–
MS SQL Server	–	+	+	+/–
SQLite	+	+/–	+	+
PostgreSQL	+	+	+	+/–

Таким образом, выбор остается сделать из двух СУБД: SQLite и PostgreSQL. Так как в перспективе полная поддержка SQL важнее лёгкости развёртывания, в контексте данной курсовой работы выбор был сделан в пользу PostgreSQL.

1.5 Выводы

В данном разделе была произведена постановка задачи, проведен анализ существующих решений, выбрана в качестве модели данных реляционная модель данных, и PostgreSQL — в качестве СУБД.

2 Конструкторский раздел

В данном разделе рассматривается процесс проектирования структуры приложения.

2.1 Диаграмма прецедентов

На рисунке 2.1 изображена диаграмма прецедентов.

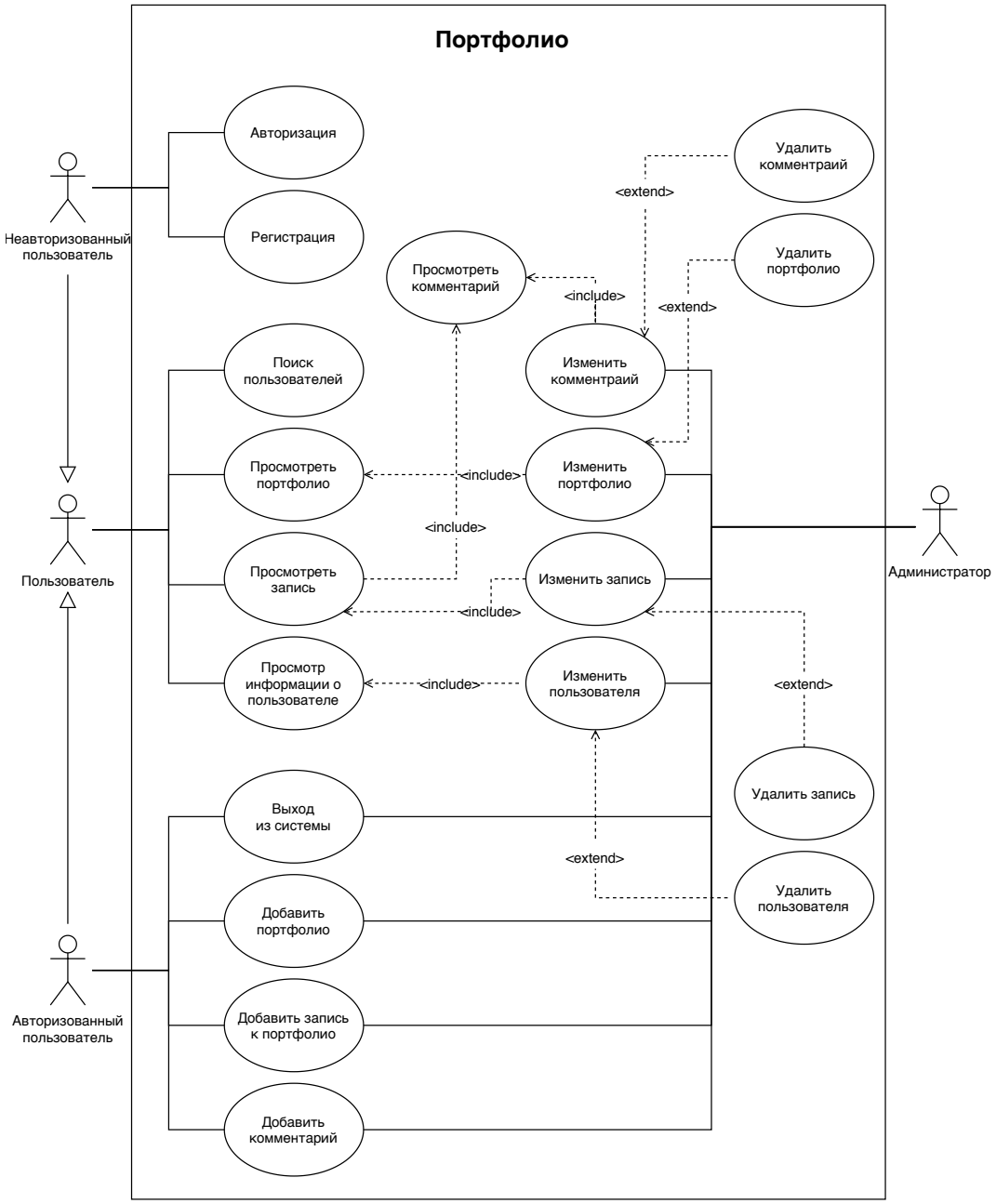

Рис. 2.1 – Диаграмма прецедентов

2.2 Проектирование базы данных

Перед тем как спроектировать базу данных, необходимо выделить сущности и связи между ними. Диаграмма Сущность-Связь изображена на рисунке 2.2.

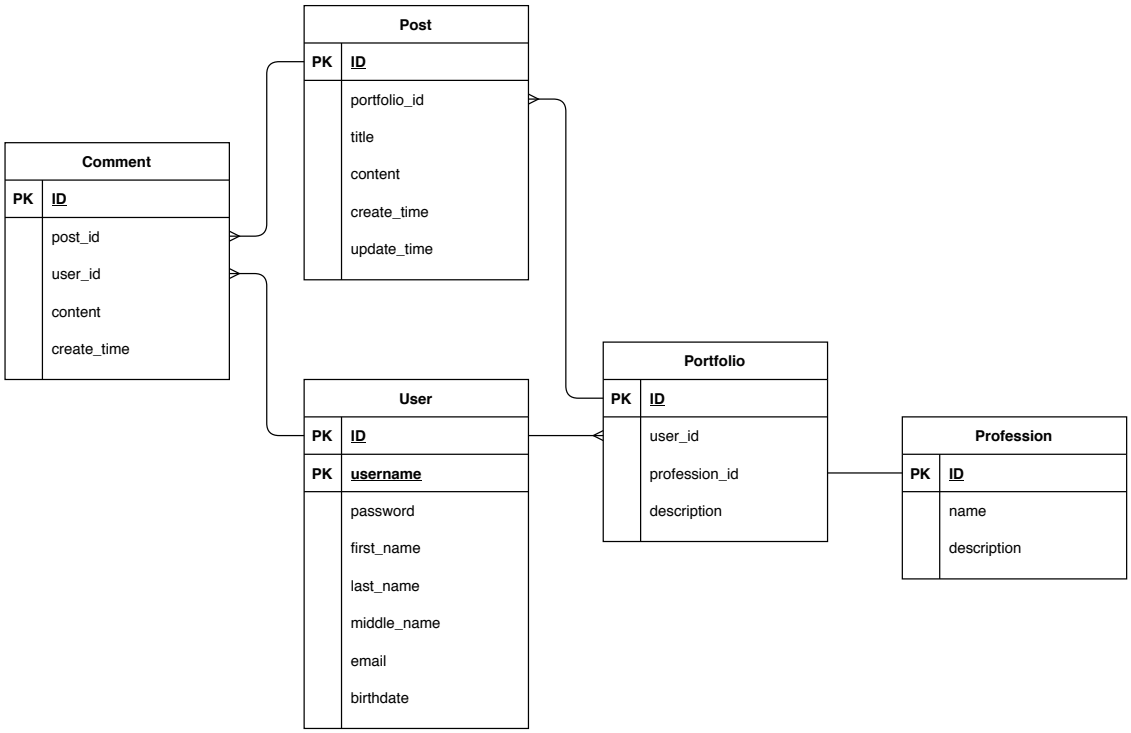

Рис. 2.2 – Диаграмма Сущность-Связь

На основе полученной диаграммы была спроектирована база данных, состоящая из следующих таблиц:

- таблица пользователей User;
- таблица профессий Profession;
- таблица портфолио Portfolio;
- таблица записей Post;
- таблица комментариев Comment.

2.2.1 Таблица User

Данная данная таблица хранит записи данных о пользователях.

- id – целочисленный первичный ключ;
- username — символьное поле, имя пользователя, так же первичный ключ, но более удобный для человеческого восприятия;
- password — зашифрованное символьное поле, пароль;
- first_name — символьное поле, имя;
- last_name — символьное поле, фамилия;
- middle_name — символьное поле, отчество;
- email — символьное поле, электронная почта;
- birthdate — поле даты, дата рождения.

2.2.2 Таблица Profession

Данная таблица содержит записи данных о профессиях.

- id – целочисленный первичный ключ;
- name — символьное поле, название профессии;
- description — символьное поле, описание профессии.

2.2.3 Таблица Portfolio

Данная таблица содержит записи данных о портфолио пользователей.

- id — целочисленный первичный ключ;
- user_id — целочисленный внешний ключ на пользователя-хозяина портфолио;
- profession_id — целочисленный внешний ключ на связанную с данным портфолио профессию;
- description — символьное поле, содержание портфолио.

2.2.4 Таблица Post

Данная таблица содержит записи о данных постов пользователей.

- id — целочисленный первичный ключ;
- portfolio_id — целочисленный внешний ключ на портфолио, в контексте которого данный пост был создан;
- title — символьное поле, заголовок поста;
- content — символьное поле, содержание поста;
- create_time — поле времени и даты, время создания поста;
- update_time — поле времени и даты, время последнего изменения поста.

2.2.5 Таблица Comment

Данная таблица содержит записи о данных комментариев пользователей.

- id — целочисленный первичный ключ;
- post_id — целочисленный внешний ключ на запись, в контексте которой данный комментарий был создан;
- user_id — целочисленный внешний ключ на пользователя-автора комментария;
- content — символьное поле, содержание комментария;
- create_time — поле времени и даты, время создания комментария;

2.3 Выводы

В данном разделе были продемонстрированы диаграмма прецедентов, диаграмма сущность-связь, а так же спроектирована база данных.

3 Технологический раздел

В данном разделе производится выбор средств программной реализации и приводится графический интерфейс пользователя веб-приложения.

3.1 Выбор инструментов разработки

В качестве языка реализации описываемого приложения был выбран Python 3. Такой выбор обусловлен следующими факторами:

- портабельность — Python является интерпретируемым языком программирования, реализация которого представлена на всех актуальных архитектурах и операционных системах;
- мультипарадигмальность и, как следствие, поддержка объектно-ориентированного подхода программирования;
- динамическая типизация;
- удобочитаемость исходного кода;
- богатая стандартная библиотека.

В качестве фреймворка для веб-приложения был выбран Django. Данный фреймворк использует шаблон проектирования MTV («Модель-Шаблон-Представление»), который по сути является интерпретацией патерна MVC («Модель-Представление-Контроллер»)[3]. Для общения с базой данных Django использует собственный ORM, поддерживающий работу с выбранной базой данных. Так же в Django по умолчанию применяется шаблонизатор Jinja.

Для упрощения решения задачи создания интерфейса при помощи html, css и javascript был выбран фреймворк Bootstrap[4].

3.2 Особенности реализации

Рассмотрим некоторые особенности реализации.

3.2.1 Корректировка базы данных

Ещё одним достоинством Django является встроенная система пользователей. Другими словами, данный фреймворк уже реализует методы идентификации, аутентификации, авторизации и т.д., что приводит к тому, что Django самостоятельно создает инициализирующие миграции, описывающие среди прочего таблицу пользователя. Данную таблицу нельзя изменять, и не все её атрибуты соответствуют изображённой на рисунке 2.2 сущности. Для решения этой проблемы было принято решение ввести дополнительную сущность Profile, которая вобрала в себя недостающие атрибуты сущности User. Между этими сущностями установлена связь один к одному. Обновленная диаграмма Сущность-Связь представлена на рисунке 3.1.

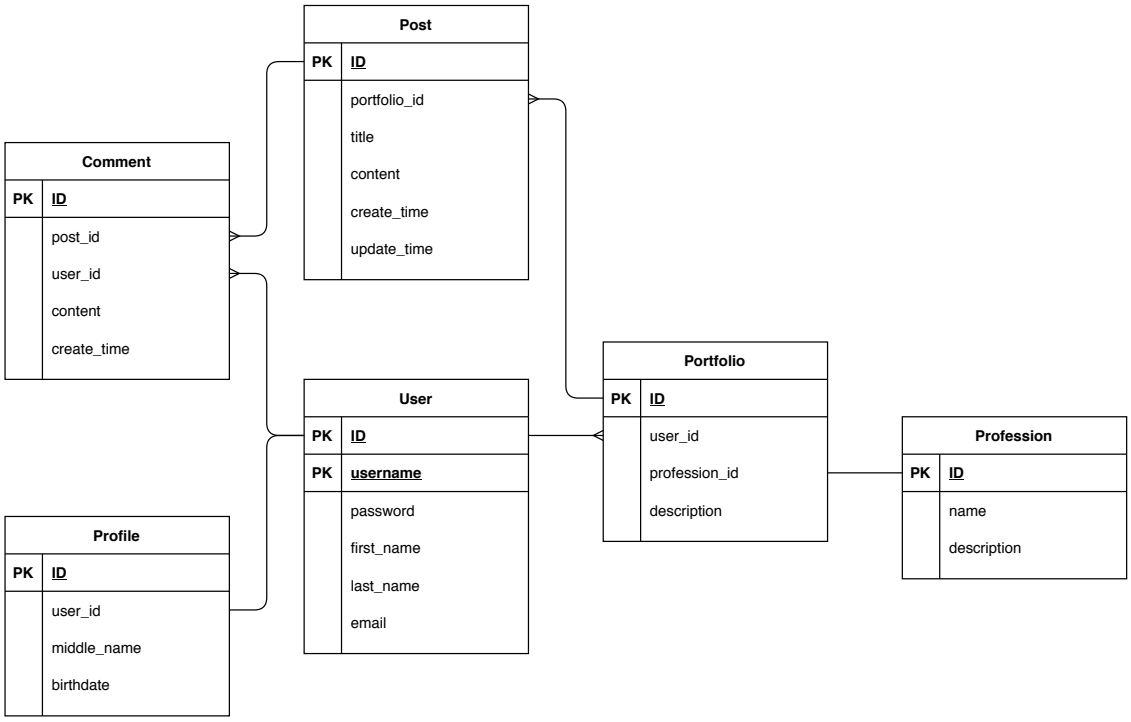

Рис. 3.1 – Обновлённая диаграмма Сущность-Связь

3.2.2 Модели

Для общения с базой данных в Django используются модели. В программе они являются единственным источником хранимых данных и по сути представляют из себя отображение таблиц базы данных (в случае реляционной модели). Каждая модель наследуется от базового класса

Model, который реализует необходимую для взаимодействия с базой данную функциональность. Каждое поле модели представляет соответствующий атрибут таблицы. В листинге 3.1 представлены модели Django данного проекта.

Листинг 3.1 – Модели Django

```
from django.db import models
from django.contrib.auth.models import User

import datetime


class Profile(models.Model):
    user_id = models.ForeignKey(User, on_delete=models.CASCADE)
    middle_name = models.CharField('middle_name', max_length=30, null=True)
    birthdate = models.DateField('birthdate', null=True)

    def __str__(self):
        return 'Profile object (%d) of user (%d)' % (self.id,
                                                     self.user_id.id)


class Profession(models.Model):
    name = models.CharField('name', max_length=50)
    description = models.TextField('description')

    def __str__(self):
        return self.name


class Portfolio(models.Model):
    user_id = models.ForeignKey(User, on_delete=models.CASCADE)
    profession_id = models.ForeignKey(Profession, on_delete=models.CASCADE)
    description = models.TextField('description')

    def __str__(self):
        return '%s %s (%d) — %s' % (self.user_id.first_name,
                                    self.user_id.last_name,
                                    self.user_id.id,
                                    self.profession_id.name)


class Post(models.Model):
    portfolio_id = models.ForeignKey(Portfolio, on_delete=models.CASCADE)
    title = models.CharField('title', max_length=50)
    content = models.TextField('content')
    create_time = models.DateTimeField('create_time')
```

```
42      update_time = models.DateTimeField('update_time', null=True)
43
44      def __str__(self):
45          return self.title
46
47
48  class Comment(models.Model):
49      user_id = models.ForeignKey(User, on_delete=models.CASCADE)
50      post_id = models.ForeignKey(Post, on_delete=models.CASCADE)
51      content = models.TextField('content', max_length=140)
52      create_time = models.DateTimeField('create_time')
```

3.2.3 Менеджмент URL

В листинге 3.2 представлен фрагмент исходного кода приложения, отвечающий за связь представлений и URL.

Листинг 3.2 – Менеджмент URL

```
1  urlpatterns = [
2      path('', views.root, name='root'),
3      path('home/', views.index, name='home'),
4      path('about/', views.about, name='about'),
5      path('search/<str:searching>', views.search, name='search'),
6      path('login/', views.login, name='login'),
7      path('logout/', views.logout, name='logout'),
8      path('join/', views.join, name='join'),
9      path('profile/', views.profile, name='profile'),
10     path('create/', views.create, name='create'),
11     path('user/<str:username>', views.user, name='user'),
12     path('portfolio/<int:portfolio_id>', views.portfolio, '
13          'name='portfolio'),
14     path('post/<int:post_id>', views.post, name='post'),
15 ]
```

3.3 Описание интерфейса программы

Неавторизованному пользователю предоставляется возможность либо авторизоваться, либо пройти регистрацию учётной записи. На рисунке 3.2 изображена форма регистрации, а на рисунке 3.3 — форма авторизации.

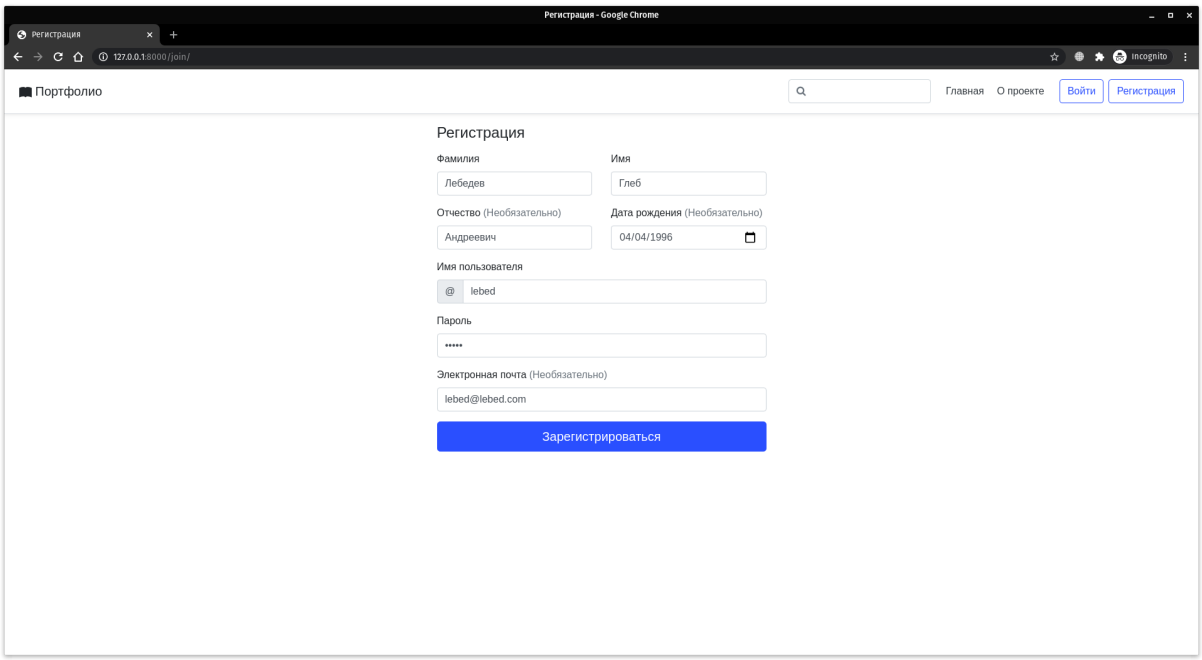

Рис. 3.2 – Форма регистрации

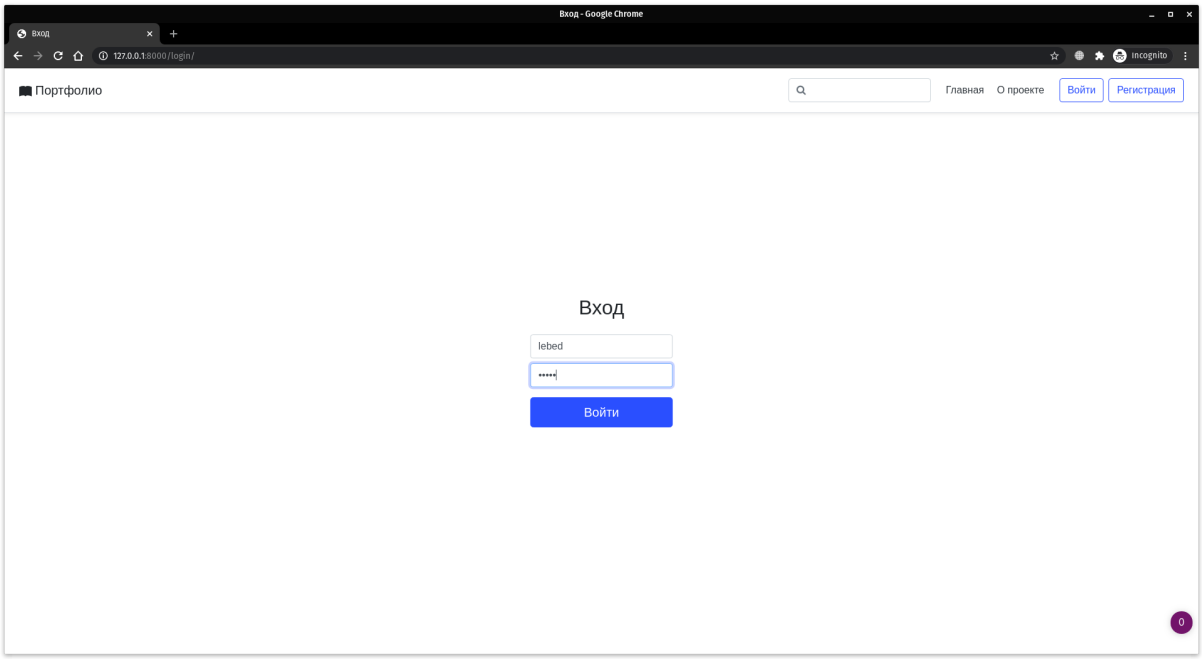

Рис. 3.3 – Форма авторизации

После авторизации пользователя приветствует его персональная страница, на которой отображаются информация о нём и список его портфолио. На рисунке 3.4 изображена описанная страница пользователя. Кроме того, здесь пользователь может перейти к форме создания нового портфолио, нажав на соответствующую кнопку.

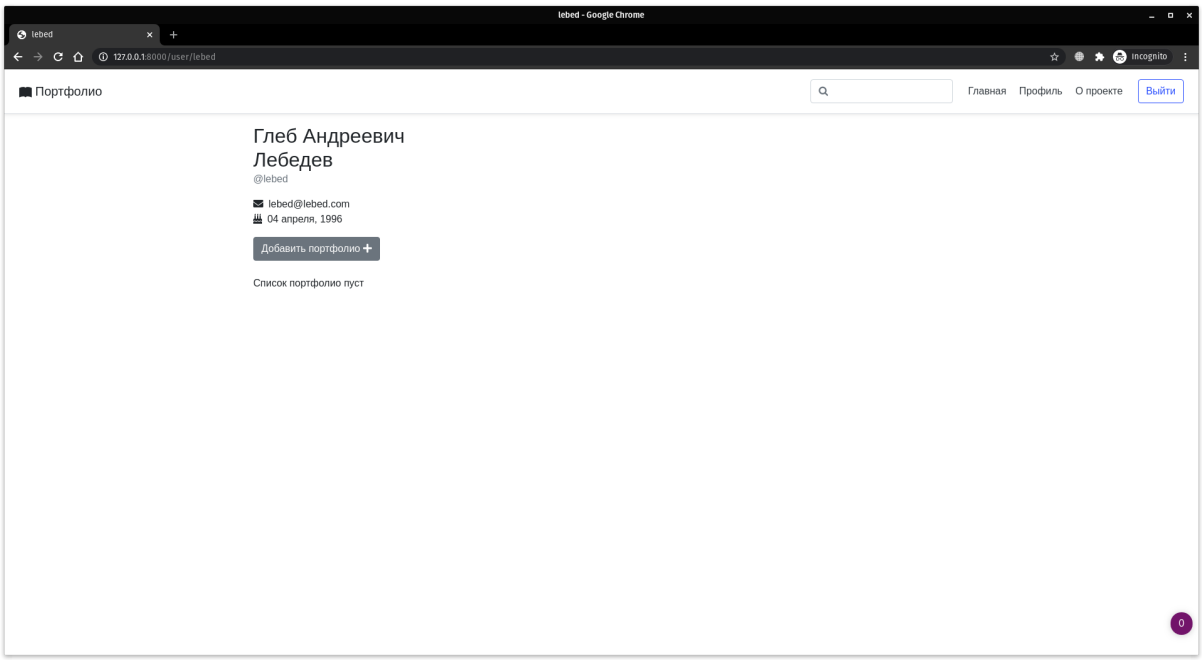

Рис. 3.4 – Персональная страница пользователя

На рисунке 3.5 представлена форма создания нового портфолио. Данная форма доступна только авторизованным пользователям, остальные при попытке доступа к ней будут перенаправлены на форму авторизации.

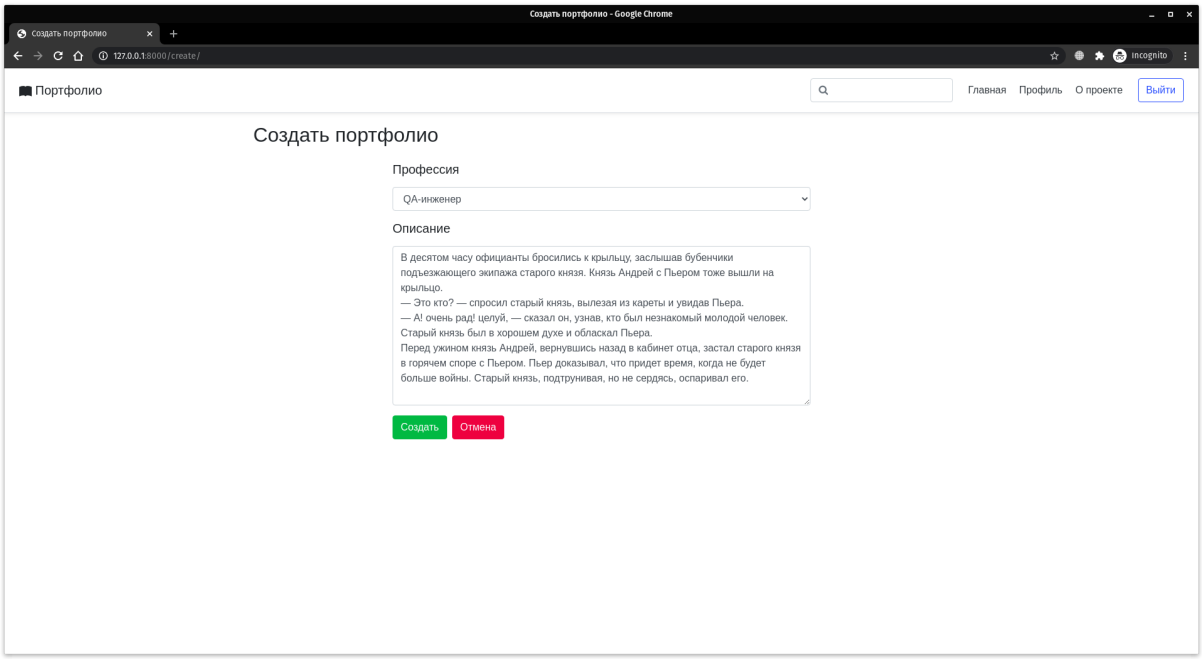

Рис. 3.5 – Форма создания нового портфолио

На рисунке 3.6 изображена страница портфолио. Тут находятся указание пользователя-хозяина портфолио, указание и описание профессии

для которой создано это портфолио, а так же список постов пользователя и форма для создания новых записей.

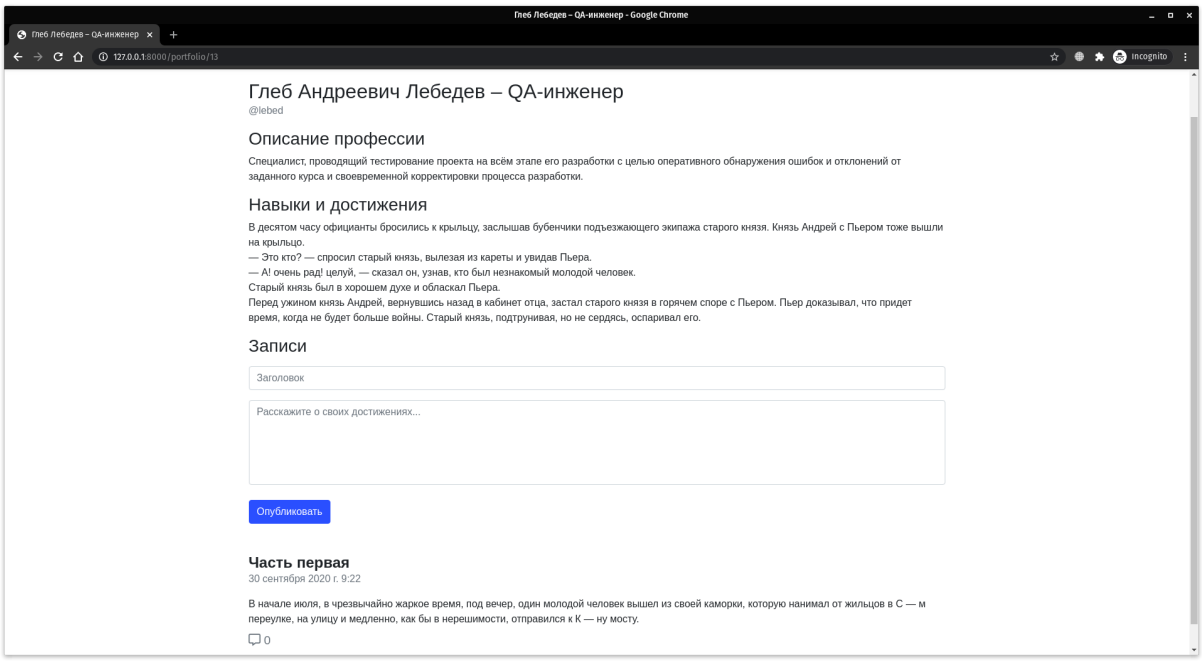

Рис. 3.6 – Страница портфолио

На странице записи (рисунок 3.7) помимо самой записи находится список комментариев пользователей и форма для создания нового комментария.

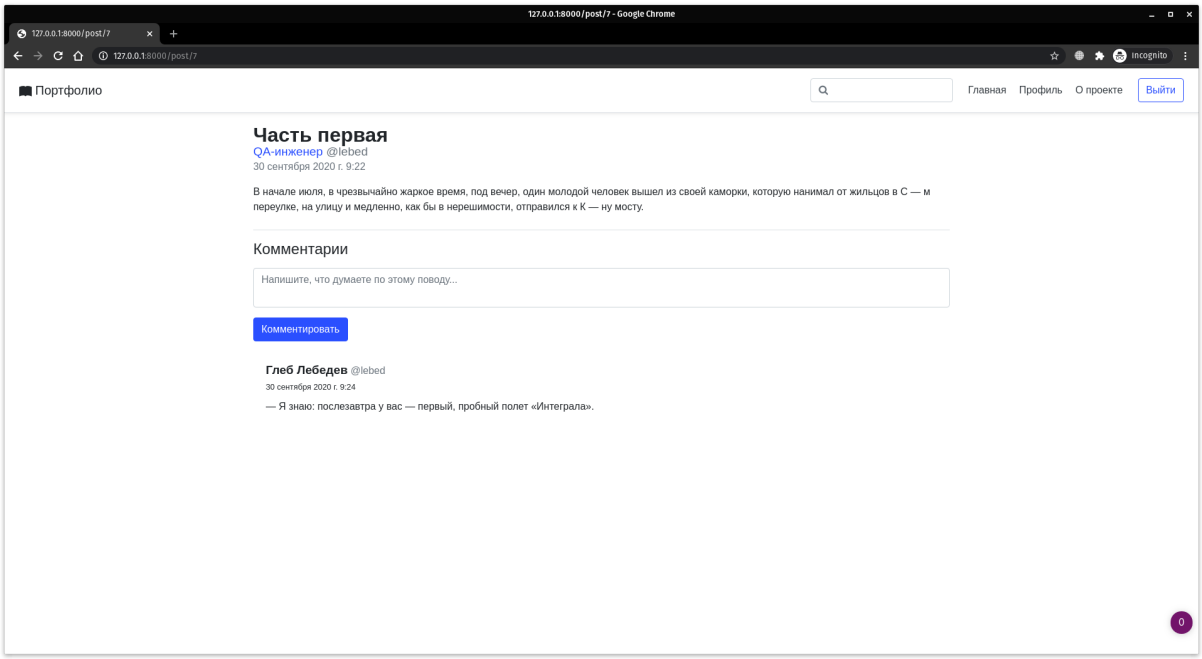

Рис. 3.7 – Страница записи

3.4 Тестирование

Было проведено функциональное тестирование белого ящика, в ходе которого программа отработала правильно. Кроме того было проведено тестирование пользовательского интерфейса: попытка регистрации существующего пользователя, попытка создания повторяющегося портфолио, проведение поиска пользователей по имени пользователя, просмотр персональных страниц, записей и т.д., все элементы интерфейса реагируют корректно. Ожидаемое поведение приложения совпадает с полученными результатами.

Апробация проводилась на ноутбуке, подключённом к сети питания. Конфигурация ноутбука:

- процессор Intel Core i5–8400H 2.5 ГГц;
- 8 гб ОЗУ;
- ОС Ubuntu 20.04.

3.5 Выводы

В данном разделе был произведён выбор инструментов разработки и рассмотрены интерфейс web-приложения и его основные функции.

ЗАКЛЮЧЕНИЕ

Разработан программный продукт в соответствии с поставленным техническим заданием и выполнены следующие задачи:

- проанализированы имеющиеся на рынке решения;
- проведён сравнительный анализ существующих моделей БД и СУБД;
- спроектирована база данных, обеспечивающая структурное хранение данных;
- с использованием выбранной СУБД реализована спроектированная база данных;
- реализовано клиент-серверное web-приложение для взаимодействия с имеющейся базой данных;
- проверена работоспособность разработанного приложения.

Цель курсовой работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Г.А. Петров, С.В. Тихов, В.П. Яковлев «Базы данных, учебное пособие» [Электронный ресурс] — Режим доступа: http://nizrp.narod.ru/metod/kafpriklmatiif/8.pdf.

2. ТОП-10 систем управления базами данных в 2019 году [Электронный ресурс] — Режим доступа: https://proglib.io/p/databases-2019.

3. Django documentation [Электронный ресурс] — Режим доступа: https://docs.djangoproject.com/en/3.0/.

4. Bootstrap documentation [Электронный ресурс] — Режим доступа: https://getbootstrap.com/docs/4.1/getting-started/introduction/.