МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И
ТЕХНОЛОГИЧЕСКОГО ОБРАЗОВАНИЯ**

Кафедра информационных технологий и электронного обучения

Основная профессиональная образовательная программа

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) «Технологии разработки программного обеспечения»

форма обучения – очная

Курсовая работа

«Анализ инструментов и технологий для разработки электронного портфолио»

Обучающегося 4 курса

Собинина Егора Яковлевича

Научный руководитель:

Кандидат физико-математических наук,

доцент кафедры ИТиЭО

Жуков Николай Николаевич

Санкт-Петербург

2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время электронное портфолио становится все более популярным инструментом для демонстрации профессиональных навыков и достижений. Оно позволяет специалистам представить свои проекты, опыт работы и компетенции в электронном формате, что упрощает процесс поиска работы и повышает конкурентоспособность на рынке труда. Кроме того электронное портфолио может использоваться студентом, для демонстрации своих работ, выполненных в период обучения

Если же говорить об IT-специальностях, то электронное портфолио может быть таким же проектом для развития компетенций. В нем можно попрактиковаться в использовании изученных ранее технологий и применить какие-либо новые знания на практике.

В данной курсовой работе мы проведем анализ инструментов и технологий для создания электронного портфолио на примере портфолио IT-специалиста, а также реализуем портфолио на оптимальной технологии.

Предмет исследования – способы организации электронного портфолио.

Цели работы:

1. Рассмотреть различные подходы к созданию портфолио.
2. Проанализировать преимущества и недостатки различных инструментов и технологий.
3. Реализовать электронное портфолио используя оптимальные инструменты.

Инструменты для создания портфолио

Tilda и Google Sites

В первую очередь можно вспомнить про такие решения как Google Sites и Tilda. Они предлагают достаточно обширный набор инструментов для создания простых статичных веб-сайтов.

Как Tilda, так и Google Sites имеют довольно низкий порог входа, так что кто любой человек может за вечер разобраться и сделать себе портфолио. Это хороший вариант для людей, чьи технические познания не очень высоки. Например представители профессий, не связанных с IT-технологиями, могут воспользоваться этими инструментами, потому как для них не важно как и где размещено портфолио, важно чтобы любой мог с ним ознакомиться.

Однако если мы будем говорить о портфолио IT-специалиста, то это не лучший вариант, так как использование таких простых инструментов косвенно намекает на нашу низкую квалификацию.

Преимущества

- Самая быстрая и простая разработка;
- Бесплатный хостинг.

Недостатки

- Малые возможности по доработке функционала;
- Не подходит IT-специалистам.

Организация страницы на GitHub и Replit

Этот вариант уже не такой уж и плохой, там можно быстро и удобно организовать портфолио IT-специалиста. Из-за особенностей данного сервиса нам можно не думать о том, как хранить наши проекты и как предоставлять к ним доступ, потому как это их изначальная и первостепенная задача.

Ко всему прочему многие работодатели просят сразу ссылку на GitHub, а значит это может быть вполне неплохим решением.

Однако данная реализация не подходит людям, которые не связаны с IT-отраслью, потому как данные сервисы в подавляющем большинстве случаев используются именно в среде информационных технологий. Если, в частности, говорить о системе Replit, то она вообще заточена только для IT-специалистов.

Так же мы имеем довольно скудные возможности по изменению интерфейса, что с одной стороны ограничивает наши возможности, а с другой стороны не позволяет наделать большое количество ошибок.

Преимущества

- Быстрая и простая разработка;
- Понятные и востребованные сервисы среди работодателей в сфере информационных технологий.

Недостатки

- Малые возможности по доработке функционала;
- Совсем не используется в сферах отличных от IT.

Разработка портфолио с помощью веб-фреймворков

С первого взгляда может показаться, что этот вариант подходит только IT-специалистам, однако я считаю, что это не так. Разработка портфолио с помощью программных решений для специалистов других профессий ограниченно доступна ввиду большого количества готовых решений, которые довольно легко правятся под свои нужды, а также существует огромный рынок аутсорс решений.

Такой подход предоставляет самые широкие возможности по реализации всех идей и позволяет закрыть любые возникающие требования к электронному портфолио.

Так же использование своего собственного решения позволяет в меньшей степени зависеть от сторонних сервисов и не создает рамок для дальнейшего развития.

Преимущества

- Самое гибкое решение по реализации;

- Меньшая зависимость от сторонних сервисов.

Недостатки

- Высокий порог входа;
- Возможная необходимость денежных затрат.

Какой веб-фреймворк выбрать для backend части портфолио

Так как в большей степени мы изучаем разработку на Python, то и выбор веб-фреймворка у нас ограничивается этим языком, хотя никто в действительности и не ограничивает выбор языка. Среди самых популярных решений мы можем увидеть Flask[4], FastAPI[3] и Django[2].

Flask и FasAPI это более легковесные решения, которые не имеют почти ничего из коробки. Нам придется отдельно ставить ORM (например SQLAlchemy), писать самому или найти готовую Admin-панель, отдельно искать решения для реализации front-end'a.

Несмотря на кажущуюся простоту и легковесность, FastAPI и Flask - не лучший выбор для начинающего разработчика, потому как они не предлагают никаких готовых решений по организации архитектуры нашего приложения. Кроме того, Django все еще остается одним из самых популярных решений для backend разработки на Python.

А значит изучение данного инструмента позволит нам в дальнейшем значительно продвинуться по карьерной лестнице.

Кроме того, в интернете большая часть материалов написана именно по Django, а также большее количество вакансий требует именно эту технологию.

Таблица 1 – Сравнение Python фреймворков

	Django	FastAPI	Flask
Преимущества	• Наиболее популярный; • Много обучающего материала; • Очень большое сообщество; • Самая подробная и качественная документация; • Наличие архитектуры проекта «из коробки» (иногда может быть минусом); • Наличие панели администратора «из коробки».	• Самый современный; • Быстрый; • Легковесный; • Часть документации на русском языке; • Возможность реализовать любой шаблон проектирования .	• Легковесный; • Много обучающего материала; • Хорошая документация; • Наличие «быстрого старта»; • Возможность реализовать любой шаблон проектирования .
Недостатки	• Наиболее требовательный к ресурсам; • Привязанность к шаблону проектирования MTV (разновидность MVC).	• Не так много обучающего материала; • Отсутствие архитектуры проекта «из коробки» (иногда может быть плюсом); • Отсутствие панели администратора «из коробки».	• Теряет актуальность после появления FastAPI; • Отсутствие архитектуры проекта «из коробки» (иногда может быть плюсом); • Отсутствие панели администратора «из коробки».

Небольшая справка по другим технологиям

Также хочу заметить, что такое портфолио возможно использование и других инструментов, например Java Script в связке с Node.js, Ruby в связке с Ruby on Rails, написание веб-сервера на Golang и реализация фронтенда с помощью Java Script фреймворков, однако они имеют некоторые существенные недостатки (более высокая сложность разработки у Node.js и Golang, более низкая поддержка сообществом и более низкая производительность у Ruby on Rails), в связи с которыми предпочтение отдается языку Python с его веб-фреймворками.

Так же мы не будем использовать статичные HTML страницы, размещаемые на веб-сервере, потому как такое портфолио будет крайне тяжело поддерживать и/или вносить какие-либо правки. Кроме того, такое портфолио не будет соответствовать современным стандартам разработки.

Исходя из всех перечисленных тезисов, оптимальным вариантом реализации электронного портфолио был выбран стек из Python и фреймворка Django.

Организация интерфейса пользователя и наполнение портфолио

Портфолио в первую очередь является своеобразной «витриной работ». Исходя из этого можно сделать вывод о реализации UI части и наполнении.

Наполнение

В первую очередь портфолио должно хорошо справляться со своей основной задачей, а значит не должно быть перегружено ни с точки зрения дизайна, ни с точки зрения наполнения[8,с.32-33]. Следует сделать интуитивно-понятный интерфейс, потому как часто с портфолио могут просматривать сотрудники HR-отделов.

Портфолио не должно быть переполненным и следует его наполнить только наиболее комплексными работами, которые отражают именно навыки специалиста, а значит работы, не связанные со специальностью, включать не стоит

Если же портфолио собирается студентом для учебных работ, то оно должно быть удобно и понятно для преподавателя, потому как он в таком случае является основным пользователем нашего портфолио. В таком случае портфолио должно включать в себя все работы, с которыми преподавателю необходимо ознакомиться.

Организация пользовательского интерфейса

Главная страница

Главная страница портфолио должна содержать карточки с кратким описанием используемых технологий и картинкой. В нижней части карточки имеет смысл сделать ссылку или кнопку, которая будет переводить на страницу с деталями[8,с.86-87]. Таким образом можно обеспечить быстрый доступ к интересующим работам. Также стоит сделать вкладку с информацией о владельце данного сайта, это будет дополнительным подтверждением авторства работ, выложенных в портфолио.

Страница проекта

Переходя на страницу проекта в первую очередь, должно появляться более подробное описание все работы. Может быть, туда стоит поместить условие задачи,

по которому был реализован проект или краткое описание принципа работы или идеи, если таковое имеется.

Необходимо добавить и ссылки на источники, где будут лежать файлы выполненных работ или ссылки на уже работающие проекты/приложения. Делается это для того, чтобы не нагружать наш сервер тяжеловесными файлами, а также распределить нагрузку на наш сервер. Кроме того, если мы говорим о портфолио разработчика, то в индустрии есть негласный стандарт размещения своих работ (например, на платформе GitHub или Bitbucket), потому как они предоставляют более широкий функционал по возможности получения доступа к коду. Портфолио же не должно обладать таким функционалом и достаточно будет просто разместить ссылки на подобные ресурсы.

Как и на главной странице необходимо разместить короткий список используемых технологий, а также разместить картинку, для того чтобы разбавить текстовую информацию.

Реализация программного продукта

Описание работы

Эта часть портфолио будет организована средствами фреймворка для веб-приложений Django. Он оптимально подходит для данного проекта ввиду изложенных преимуществ и недостатков (Таблица 1).

Наш проект структурно будет состоять из 2 приложений:

- Стандартное приложение config (тут были внесены некоторые изменения, потому как согласно технической документации Django, это приложение должно называться также, как и сам проект), который будет содержать в себе настройки Django приложения;
- Приложение projects, содержащее в себе ORM модель наших проектов, базовый и расширяющие шаблоны html страниц.

Кроме того, стоит сразу создать директорию static и uploads, на одном уровне с нашими приложениями. В директории static будут размещены статические файлы, а в директории uploads будут размещены все загружаемые к проектам картинки. Файлы manage.py и файл базы данных были созданы самим фреймворком автоматически при старте проекта.

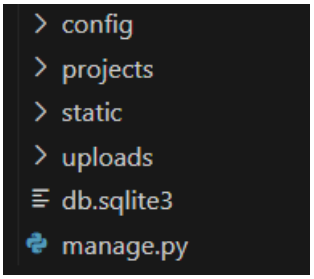

Рисунок 1 – Основная структура проекта

Редактирование конфигурации

В первую очередь после создания приложения projects в CLI, необходимо зарегистрировать его в файле конфигурации всего Django проекта (config/settings.py).

```
INSTALLED_APPS = [
    'projects.apps.ProjectsConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Рисунок 2 – Регистрация приложений

Так же необходимо зарегистрировать ранее созданные директории static и uploads в том же файле конфигурации Django проекта.

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR,  'static')
]

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

MEDIA_ROOT = BASE_DIR / "uploads/"
MEDIA_URL = "media/"
```

Рисунок 3 – Регистрация директорий

Далее нам необходимо сконфигурировать urls.py, который лежит рядом с settings.py и подключить в нем все маршруты из приложения projects.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include("projects.urls"))
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Рисунок 4 – подключение маршрутов других приложений

Создание приложения

В первую очередь надо создать модель для ORM. Приложение будет работать с данными, описанными именно в этой модели. В случае портфолио необходимо указать: название, описание, технологии, картинки, и возможные ссылки. Первичный ключ же указывать не нужно, потому как ORM создает его автоматически.

После создания модели необходимо провести все миграции, чтобы ORM могла создать все необходимые таблицы в БД, в данном случае в SQLite3.

```python
from django.db import models

# Create your models here.
class Project(models.Model):
    title = models.CharField(max_length=256)
    description = models.TextField()
    technology = models.CharField(max_length=256)
    image = models.FileField(upload_to='project_images/', blank=True)
    githublink = models.CharField(max_length=256, null=True, blank=True)
    replitlink = models.CharField(max_length=256, null=True, blank=True)
    anotherlink = models.CharField(max_length=256, null=True, blank=True)

    def __str__(self):
        return(self.title)
```

Рисунок 5 – Модель ORM

Следующим шагом укажем все необходимые маршруты, которые будут использоваться приложением.

```python
urlpatterns = [
    path("", views.project_index, name='project_index'),
    path("<int:pk>/", views.project_detail, name="project_detail")
]
```

Рисунок 6 – Указание маршрутов

После всего этого необходимо параллельно разрабатывать шаблоны HTML страниц (с использованием шаблонизатора Jinja2 и фреймворка Bootstrap[1]) (Приложение 1), (Приложение 2), (Приложение 3) и части представлений.

Представления в большей степени описывают какие данные и в каком виде должен получить пользователь. В данном случае представление передает контекст для отрисовки страниц, в котором хранятся необходимые данные, взятые из базы данных.

```python
from django.shortcuts import render
from projects.models import Project
# Create your views here.

def project_index(request):
    projects = Project.objects.all()
    context = {
        "projects": projects,
    }
    return render(request, "projects/project_index.html", context)

def project_detail(request, pk):
    project = Project.objects.get(pk=pk)
    context = {
        "project": project
    }
    return render(request, "projects/project_detail.html", context)
```

Рисунок 7 – Описание представления

Вся базовая часть разработки приложения закончена. Следующим шагом будет размещение приложения на хостинг.

Хостинг и деплой проекта

Самый простой способ задеплоить проект на сервер, это запустить его из дебаг режима по широковещательному каналу. Однако такой способ совсем не безопасный, потому как все пользователи будут видеть ошибки, возникающие на сайте. И этим могут воспользоваться злоумышленники.

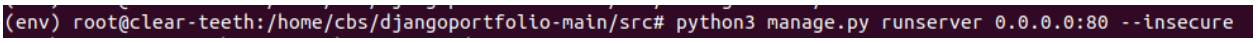

Рисунок 8 - Деплой проекта со встроенного веб-сервера Django

Другой же вариант, это развернуть nginx, в качестве прокси сервера. В таком случае злоумышленники смогут получить доступ только до Nginx[5], дальше пройти они просто не смогут.

Эти варианты доступны в том случае, если есть личный или арендованный сервер, на котором будут работать все компоненты. Если такой возможности нет, то одним из вариантов будет ручной проброс портов через роутер с машины, которая будет работать на нашей стороне.

Если такой вариант тоже не подходит, то одним из последних вариантов остается Ngrok. Он позволит запустить наш проект с общим доступом, но на локальной машине. В таком случае локальная машина становится некоторым аналогом сервера, что, к сожалению, в некоторых случаях невозможно.

Оптимальным решением же будет использовать личный или арендованный сервер (например VDS/VPS). Кроме того строго необходим перевод приложения из режима дебага, в продовый режим. Для этого в файле settings.py нужно поменять флаг DEBUG.

Так как Django не предназначен для раздачи статических файлов, то нам необходимо раздавать статические файлы с помощью Nginx. С его же помощью можно организовать перенаправление трафика. Для этого необходимо изменить файл конфигурации /etc/nginx/sites-available/default (Приложение 4).

Таким образом мы получаем собственное электронное портфолио (рисунок 9), которое теперь довольно просто можем развернуть на любых

машинах, редактировать через панель администратора (рисунок 10) и пользоваться им напрямую.

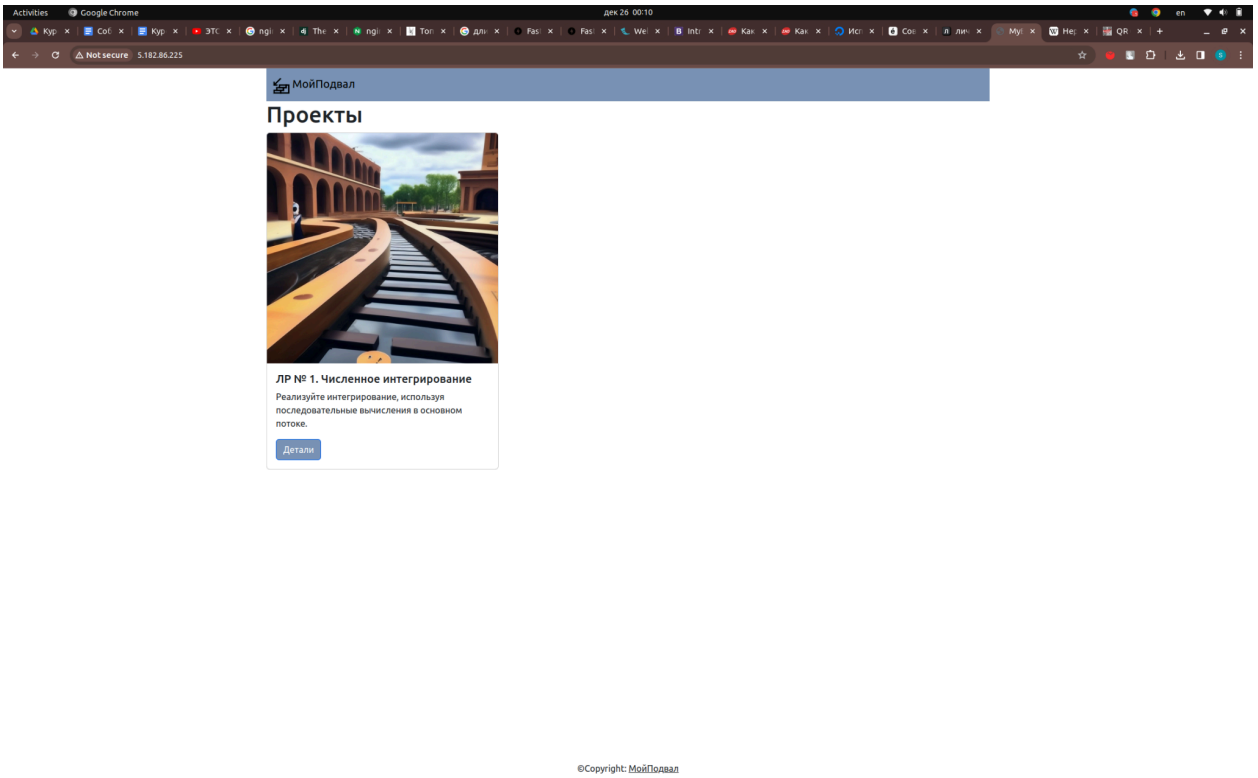

Рисунок 8 - Электронное портфолио

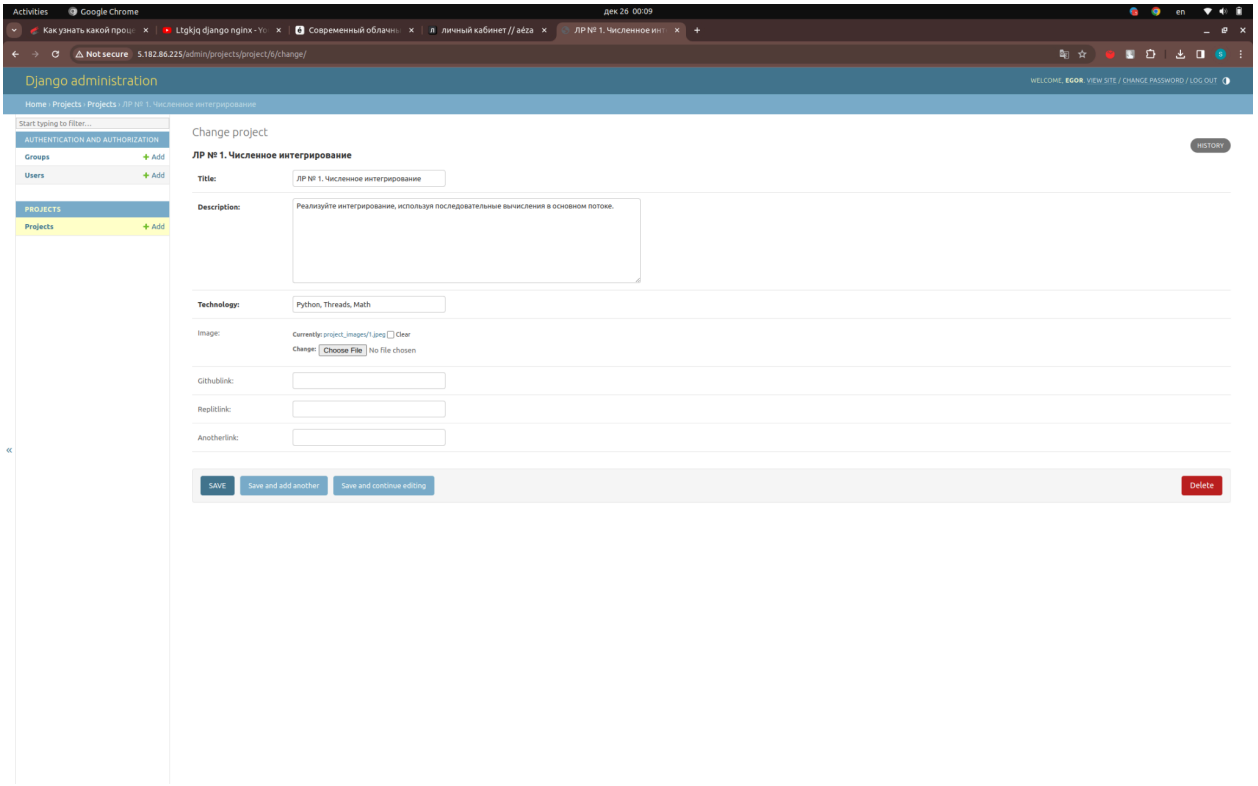

Рисунок 9 - Панель администратора

Как можно развивать портфолио дальше

С точки зрения пользовательского интерфейса можно добавить функционал пагинации страниц, для того чтобы отображать только часть работ, а пользователь в свою очередь мог сам листать страницы с нашими проектами. В случае портфолио студента, можно добавить разделы по разным дисциплинам, чтобы размещать все учебные работы в портфолио.

Так же можно подумать о другом способе реализации блока «Используемые технологии», может быть реализовать их в формате тегов, с возможностью поиска по ним. Если портфолио станет очень большим, то следует сделать отдельную страницу с наиболее важными работами, чтобы они показывались раньше других. Так же при разрастании портфолио следует реализовать поиск по названиям проектов.

С точки зрения отказоустойчивости нашего проекта можно задуматься о внедрении балансировщика нагрузки на несколько веб-серверов. Так же можно построить систему с использованием VRRP, что позволит нам потерять несколько веб-серверов, но при этом не потерять доступность нашего портфолио.

По возможности лучше получить и зарегистрировать доменное имя и настроить сертификаты безопасности для протокола HTTPS.

Кроме того можно настроить систему логирования, с помощью сервисов Grafana или Graylog. Это поможет вовремя понять необходимо ли расширять нашу систему вертикально и/или горизонтально.

Так же при многократном увеличении нашей БД, следует поменять SQLite3 на что-то более производительное (например, PostgreSQL).

Однако хочу заметить, что в таком случае портфолио скорее станет личным блогом, и превратиться из компактной витрины для демонстрации работ в большое веб-приложение, которым уже скорее всего захотят пользоваться значительно больше людей, чем целевая группа преподавателей и/или работодателей, для которых оно изначально разработано

Заключение

В ходе данной курсовой работы были проанализированы различные инструменты и технологии, предназначенные для создания электронного портфолио. Был изучен обзор существующих решений, а также проведено исследование их функциональности и возможностей.

Один из наиболее популярных инструментов для разработки веб-приложений Django — был выбран для создания электронного портфолио. Django предоставляет мощные функции для работы с базами данных, аутентификацией, обработкой форм и маршрутизацией запросов, что делает его идеальным выбором для такого проекта.

В процессе реализации электронного портфолио с использованием Django было создано приложение, которым может воспользоваться каждый человек, для размещения там своего портфолио, путем создания карточек в панели администратора.

Результаты проекта продемонстрировали, что использование Django для создания электронного портфолио обладает несколькими преимуществами. Django предоставляет быструю разработку приложений, хорошую масштабируемость и надежность. Более того, Django обеспечивает высокую степень безопасности, что особенно важно при работе с личными данными пользователей.

Таким образом, на основе проведенного анализа и реализации электронного портфолио с использованием Django можно сделать вывод о том, что данная технология предоставляет мощный инструмент для создания и управления электронным портфолио. Проект успешно сочетает в себе функциональность, простоту использования и безопасность, делая его привлекательным решением для тех, кто стремится представить свои работы и достижения в электронной форме.

Список используемой литературы

1. Документация Bootstrap: [Электронный ресурс]. URL: https://getbootstrap.com/docs/4.1/getting-started/introduction/ (Дата обращения 21.12.2023)
2. Документация Django: [Электронный ресурс]. URL: https://www.djangoproject.com/ (Дата обращения 15.12.2023)
3. Документация FastAPI: [Электронный ресурс]. URL: https://fastapi.tiangolo.com/ (Дата обращения 15.12.2023)
4. Документация Flask: [Электронный ресурс]. URL: https://flask.palletsprojects.com/en/3.0.x/ (Дата обращения 15.12.2023)
5. Документация Nginx: [Электронный ресурс]. URL: https://nginx.org/ru/
6. Дронов В. А. Django 4. Практика создания веб-сайтов на Python. — СПБ.: БХВ-Петербург, 2023. — 800 с.
7. Кетов Д. В. Внутреннее устройство Linux. — 3-у изд., перераб. и доп. — СПБ.: БХВ-Петербург, 2023. 281 — 289
8. Круг С. Не заставляйте меня думать / Стив Круг ; [пер. с англ. М. А. Райтмана]. — 3-е издание. — М.: Эксмо, 2022. — 256с.
9. Меле А. Django 4 в примерах / пер. с англ. А. В. Логунова. — М.: ДМК Пресс, 2023. — 800с.
10. Ромальо Л. Python — к вершинам мастерства: Лаконичное и эффективное программирование / пер. с англ. А. А. Слинкина. 2-е изд. — М.: МК Пресс, 2022. — 898 с.

Приложение 1

```
<!-- templates/base.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %} MyBasement {% endblock title %}</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css"
        rel="stylesheet"
    >
    <title>Document</title>
</head>
<body class="container">
    {% load static %}
        <nav class="navbar navbar-light sticky-top" style="background-color:
#7893b4;">
        <div class="container-fluid">
          <a class="navbar-brand" href="/">
                <img src="{% static 'pages/img/downstairs-svgrepo-com.svg' %}"
alt="Logo" width="30" height="30" class="d-inline-block align-text-top">
            МойПодвал
          </a>
```

```
      </div>
    </nav>
   {% block page_content %}{% endblock page_content %}


  <footer class="bg-light text-center d-block d-sm-none">
    <div class="text-center p-3" style="background-color: white;">
      ©Copyright:
      <a class="text-dark" href="/">МойПодвал</a>
    </div>
  </footer>
    <footer class="bg-light text-center fixed-bottom d-none d-sm-block">
      <div class="text-center p-3" style="background-color: white;">
        ©Copyright:
        <a class="text-dark" href="/">МойПодвал</a>
      </div>
    </footer>
</body>
</html>
```

Приложение 2

```
{% extends "base.html" %}


{% block page_content %}
<h1>Проекты</h1>
<div class="row gy-5  ">
   {% for project in projects %}
    <div class="col-md-4">
      <div class="card d-flex align-items-stretch h-100">
        {% if project.image %}
          <img class="card-img-top" src="{{ project.image.url }}">
        {% endif %}
        <div class="card-body">
          <h5 class="card-title">{{ project.title }}</h5>
          <p class="card-text">{{ project.description }}</p>
             <a href="{% url 'project_detail' project.pk %}" class="btn btn-primary" style="background-color: #7893b4">Детали</a>
        </div>
      </div>
    </div>
   {% endfor %}
  </div>
{% endblock %}
```

Приложение 3

```
<!-- projects/templates/projects/project_detail.html -->
{% extends "base.html" %}
{% block page_content %}
<h1>{{ project.title }}</h1>
<div class="row">
  <div class="col-md-8">
    {% if project.image %}
      <img src="{{ project.image.url }}" width="85%">
    {% endif %}
  </div>
  <div class="col-md-4">
    <h5>Описание:</h5>
    <p>{{ project.description }}</p>
    <br>
    <h5>Технологии:</h5>
    <p>{{ project.technology }}</p>
    {% if project.githublink or project.replitlink or project.anotherlink%}
      <h5>Полезные ссылки:</h5>
    {% endif %}
    {% if project.githublink %}
      Ознакомиться на <a href="{{ project.githublink }}">GitHub</a>
    {% endif %}
    {% if project.replitlink %}
```

```
          Ознакомиться на <a href="{{ project.replitlink }}">Replit</a>
        {% endif %}
        {% if project.anotherlink %}
          <a href="{{ project.anotherlink }}">Ссылка на файлообменник</a>
        {% endif %}
    </div>
</div>
{% endblock page_content %}
```

Приложение 4

```
server {
    listen 80;
    server_name your_domain.com;

    location /media/ {
        root /var/www/djangoportfolio-main/src;
    }
    location /static/ {
        root /var/www/djangoportfolio-main/src;
    }
    location / {
            proxy_pass http://127.0.0.1:8000;  # Порт, на котором запущен ваш Django-проект
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Приложение 5

Ссылка на Git репозиторий https://github.com/EgrSobn/portfoliodjango

Ссылка на портфолио http://5.182.86.225/