ГУАП

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

преподаватель		Опалева У.С.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТЫ О ЛАБОРАТОРНЫХ РАБОТАХ

По дисциплине: МДК 02.02 Инструментальные средства разработки

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	021к		Панков Вася
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

1 Лабораторная работа №2-4

Тема: Разработка структуры проекта. Диаграмма модулей. Перечень артефактов и протоколов. Настройка работы системы контроля версий. Разработка проекта на основании шаблона.

Цель: создание модульной структуры проекта в виде диаграммы, разработка перечня артефактов (основных файлов) и протоколов, настройка Git

Выполнение работы:

1. Требуется создать веб-приложение (сайт) на основе имеющегося шаблона. Выбрать Файл > Создать > Проект (Create a new project). Далее в открывшемся окне выбрать Bottle Web Project и нажать Next:

Задать название проекта (в наименование должна быть включена фамилия студента) > Create.

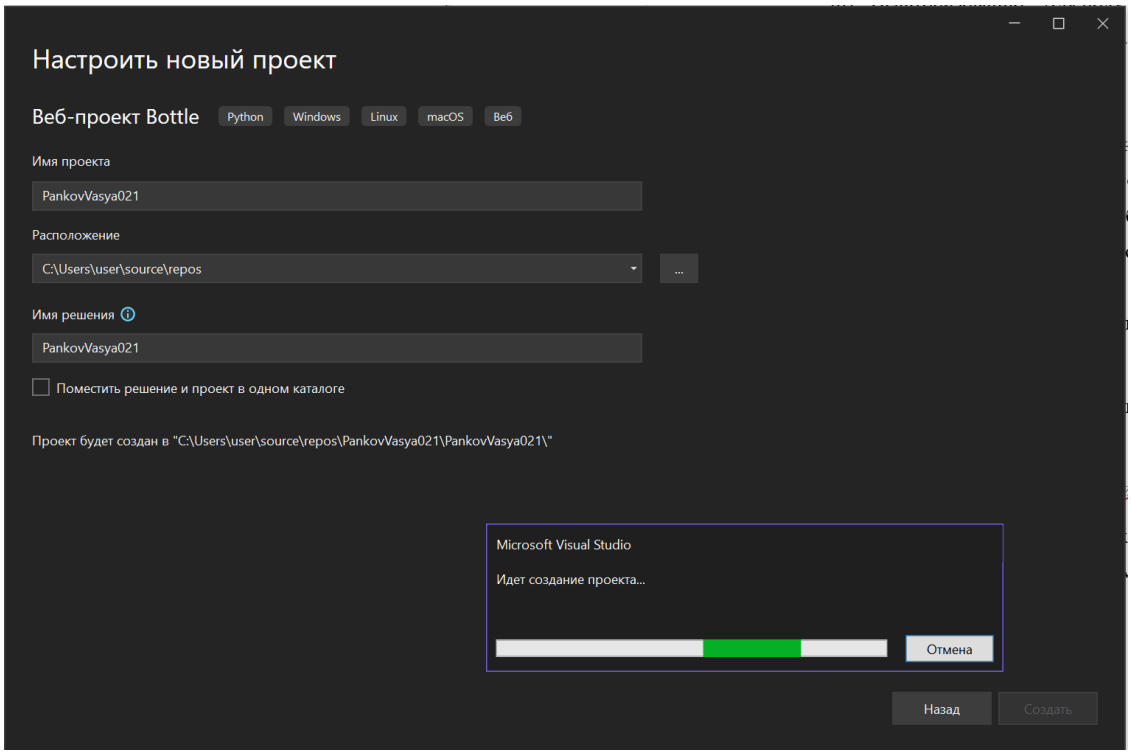

Рисунок 1 – Создание решения

2. По умолчанию в редакторе кода откроется файл app.py. Для корректной интерпретации содержимого файла requirements.txt будет предложено создать виртуальное окружение.

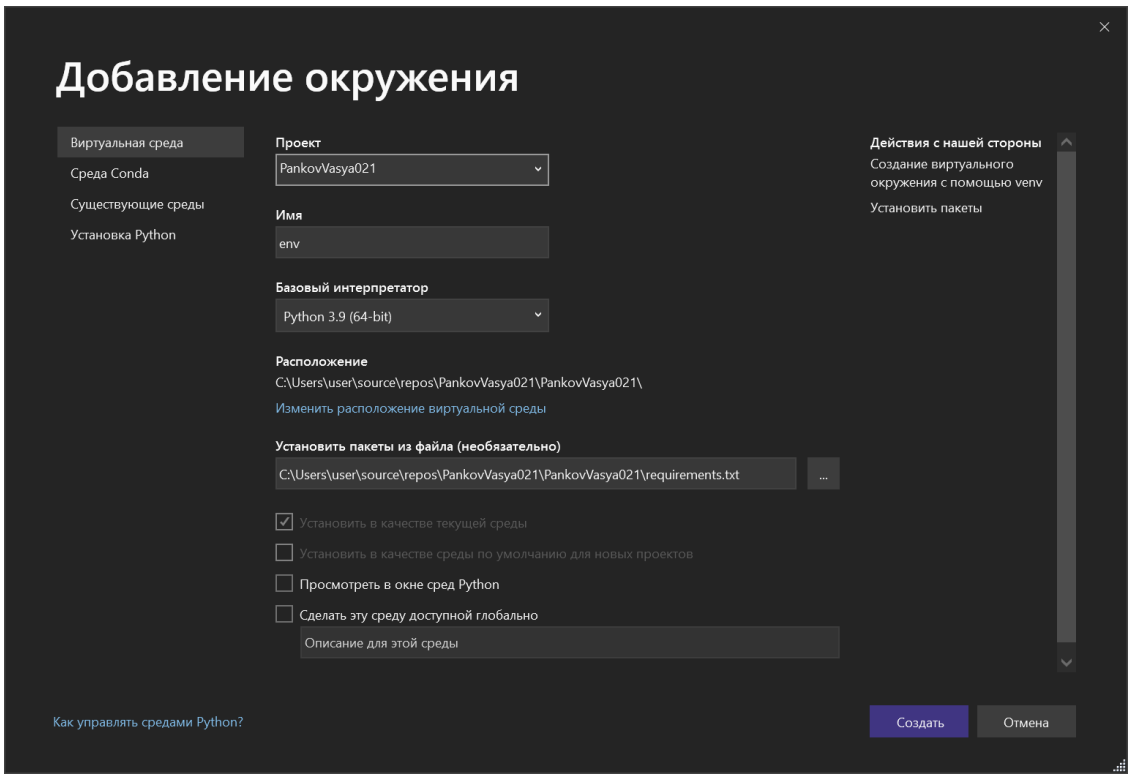

Рисунок 2 – Создание виртуальной среды

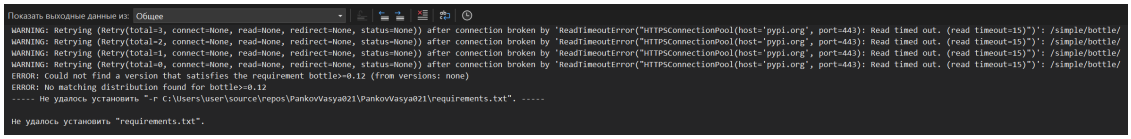

Рисунок 3 – Неудачная разработка среды

3. Ознакомиться со структурой проекта, открыв Обозреватель решений (Solution Explorer):

В папке static содержатся стилевые css-файлы для оформления внешнего вида веб-страниц, шрифтовые наборы и js-скрипты.

Выбрав для просмотра файлы из папки views, можно убедиться, что они представляют собой макеты веб-страниц: главной (index.tpl), о нас (about.tpl), контакты (contact.tpl), а файл layout.tpl – не что иное, как шаблон обёртки с панелью навигации, содержащей четыре гиперссылки: Application name, Home, About, Contact в «шапке» сайта и небольшим «подвалом».

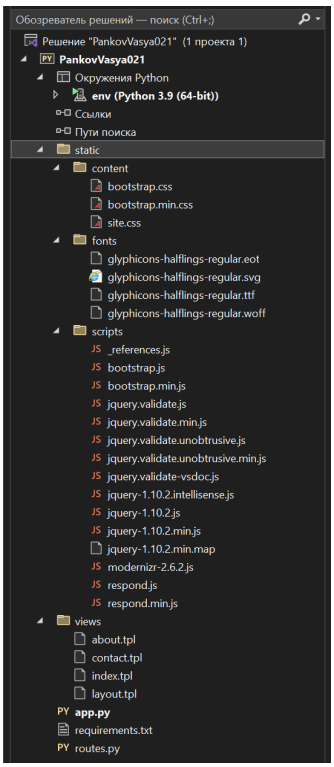

Рисунок 4 – Структура проекта

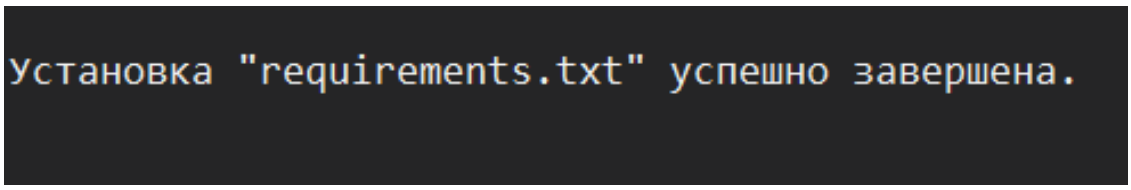

Рисунок 5 – Удачная установка

4. Выполнить запуск проекта

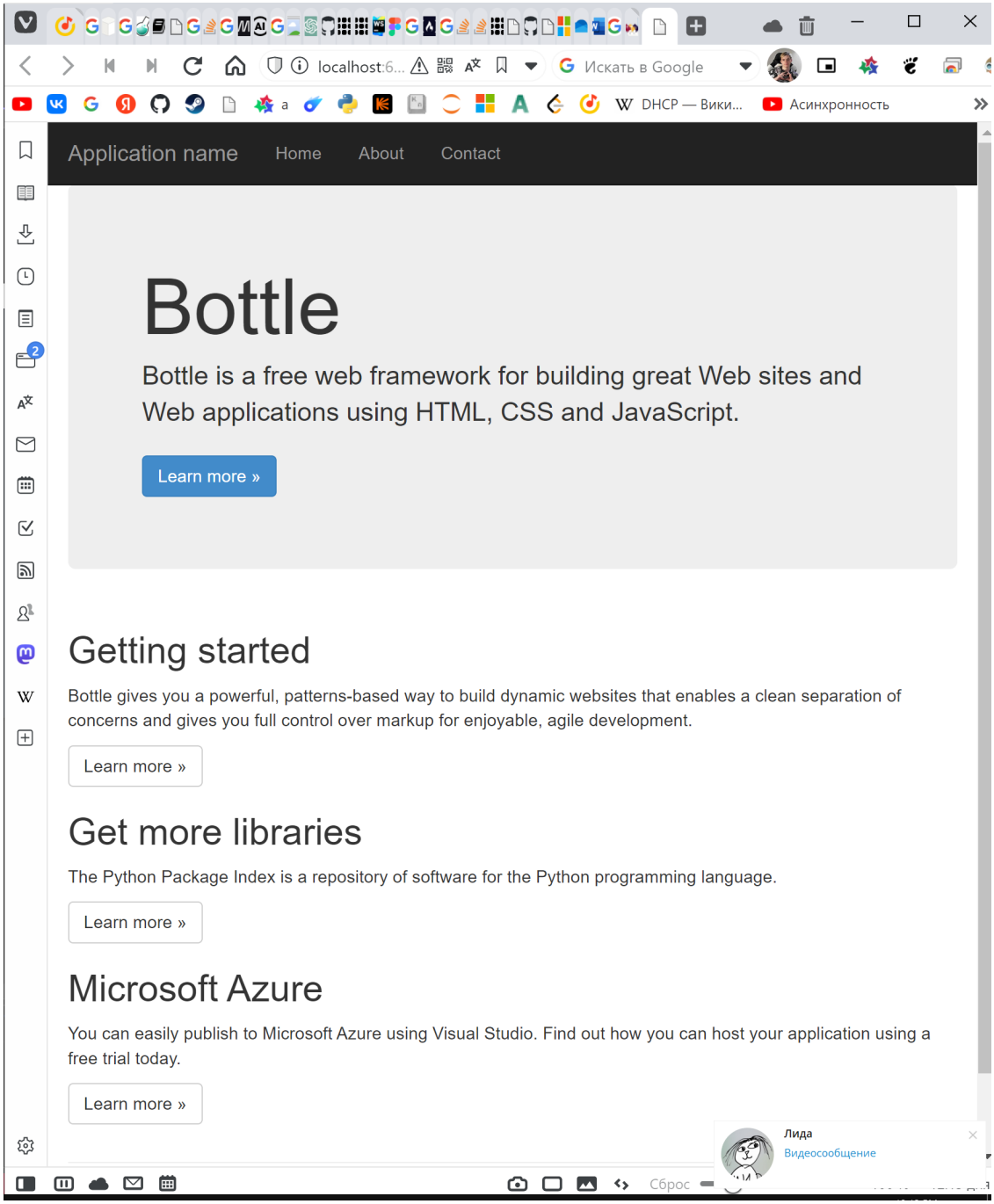

Рисунок 6 – Запуск проекта

5. Например, вместо текста заголовка «Bottle» на главной (домашней) странице требуется разместить картинку. Для этого сначала в структуре каталога static создаётся папка images, в которую копируется файл изображения $\text{logo}_{\text{nav.png}}$. Затем в коде index.tpl удалить строку <h1> Bottle </h1>, добавить <img src="$\text{static}_{\text{nav.png}}$» (файл изображения можно скачать с сайта `http://bottlepy.org/docs/dev/index.html`) и пустой абзац

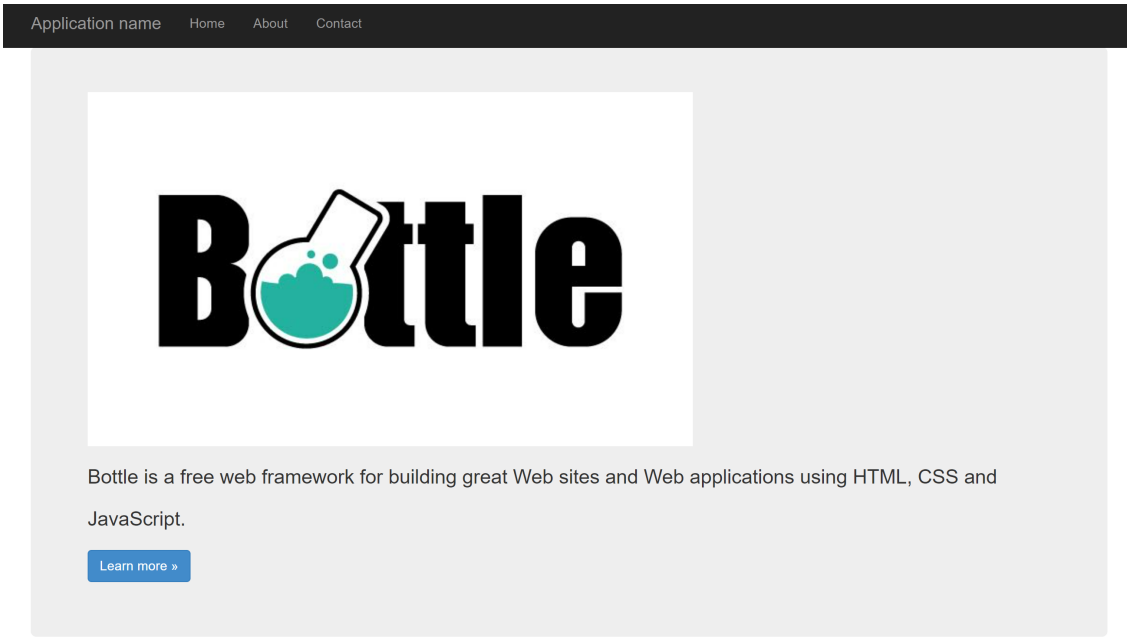

Рисунок 7 – Отображение сайта

```
<div class="jumbotron">
    <img src="/static/images/bottle.jpg">
    <p></p
    <p class="lead">Bottle is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.</p>
    <p><a href="http://bottlepy.org/docs/dev/index.html" class="btn btn-primary btn-large">Learn more &raquo;</a></p>
</div>
```

Рисунок 8 – Изменения кода

6. Требуется создать веб-приложение (тематический сайт) на основе шаблона фреймворка Bottle. Придерживаться следующих этапов:

- Выбрать тематику приложения, согласовать с преподавателем (темы не повторяются!).

  ReportViwer - сайт для просмотра отчётов

- Подобрать текстовые и графические (иллюстрации, иконки, карты) материалы для сайта.

  Был разработан дизайн в Figma:

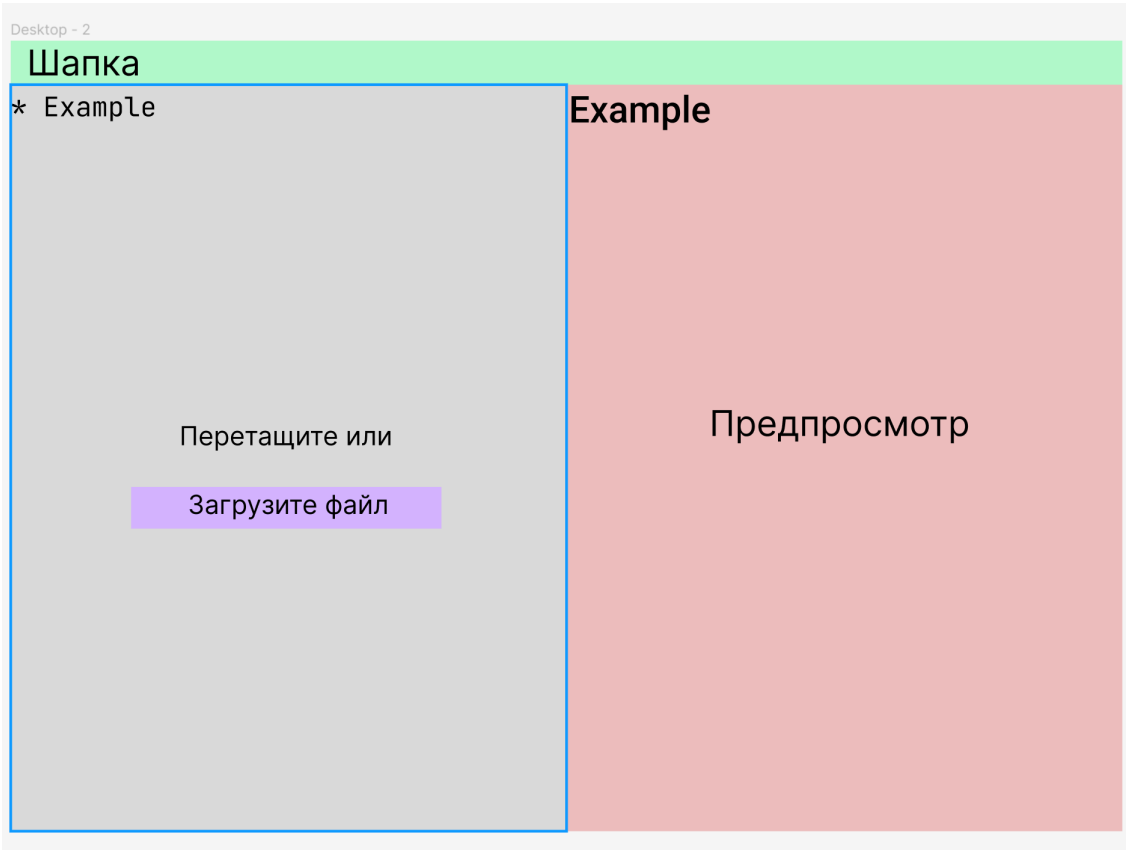

Рисунок 9 – Warframe

Рисунок 10 – Дизайн в Figma

- Построить UML-диаграмму компонентов приложения (css-файлы считать одним укрупнённым структурным компонентом, аналогично js-скрипты, их подключение осуществляется в файле layout.tpl).

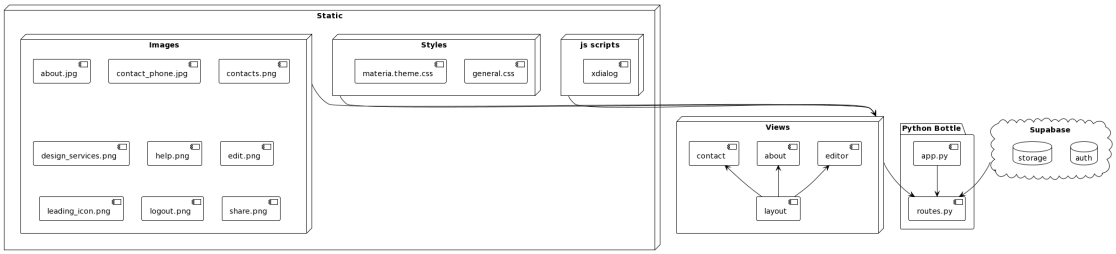

Рисунок 11 – UML-диаграммы

- Настроить Git. Хранение проекта осуществлять в репозитории на Github одного из участников, остальные клонируют проект в свой локальный репозиторий. Распределить задачи редактирования файлов шаблонов между членами команды. Изменить содержимое этих файлов согласно выбранной тематике (ссылки на кнопках поставить на нужные страницы похожих по тематике сайтов).

  Репозитоий был создан мной на GitHub

- Вести контроль версий. Синхронизировать все локальные изменения с удалённым репозиторием на Github.

  В итоге были получены такие изменения от всех пользователей:

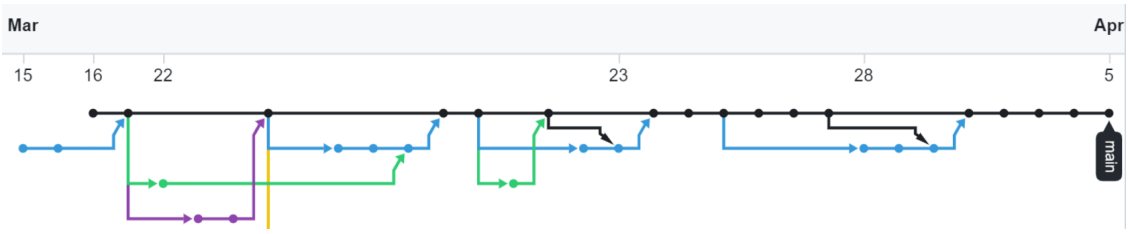

Рисунок 12 – Изменения от всех пользователей

Моя часть работы состояла, в создание основной части приложения, то есть редактора текстовых документов:

– Вид без авторизации:

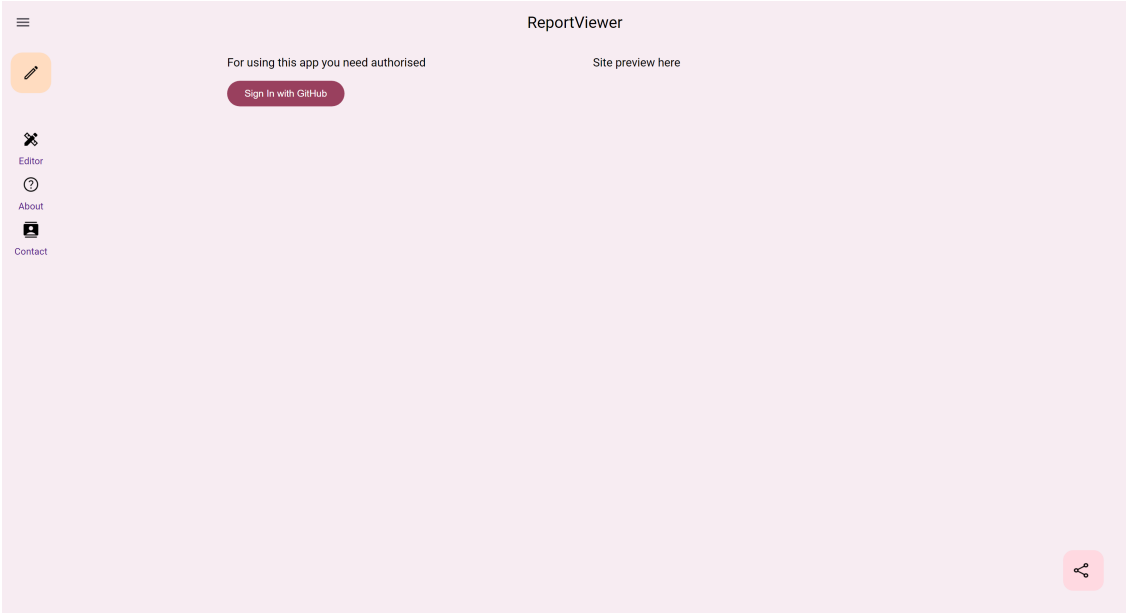

Рисунок 13 – Вид, без авторизации

– Вид, после авторизации, с выбранным файлом

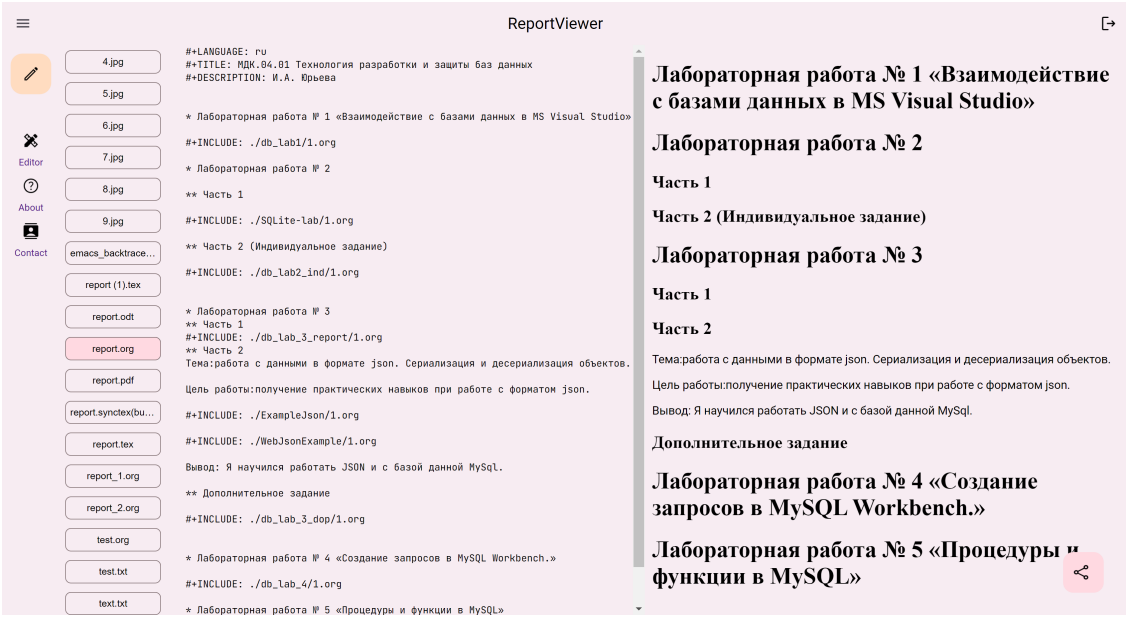

Рисунок 14 – Просмотр текстового файла

– Просмотр картинки

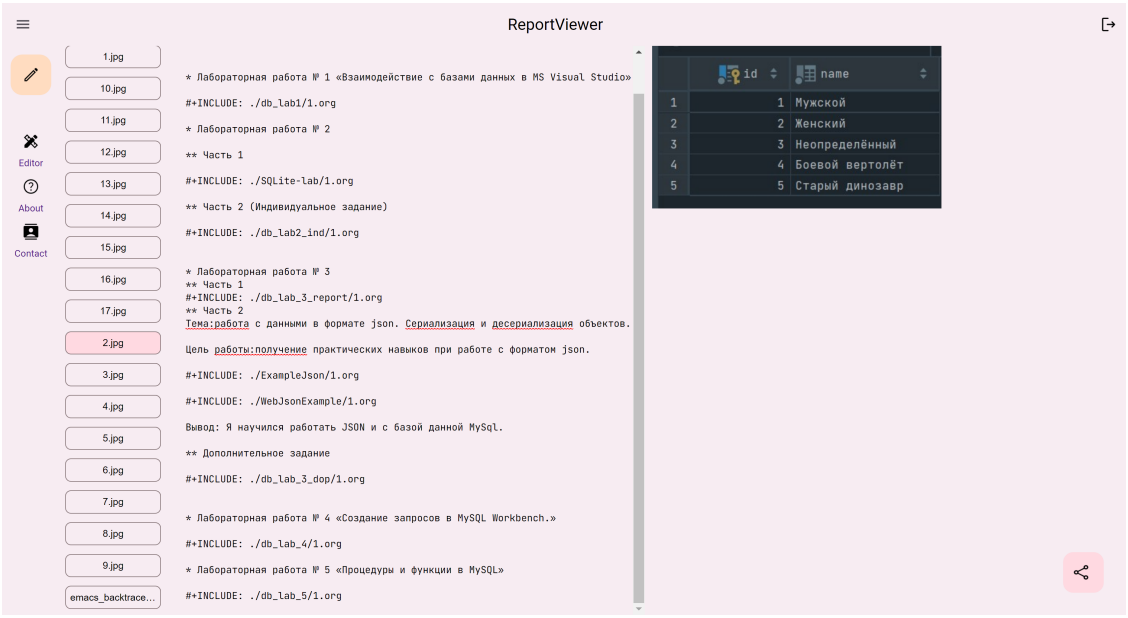

Рисунок 15 – Просмотр картинок

- Заполнить файл README.md. Примерный вариант описания можно найти в шаблоне `https://gist.github.com/PurpleBooth/109311bb0361f32d87a2` или по ссылке `https://github.com/Sinclear/default_readme`

README.md

Report_Viewer

Данный проект представляет собой сайт, который предоставляет пользователям возможность делиться отчётами, просматривать отчёты других пользователей, загружать отчёты в любом формате.

При разработке использовались:

- Среда разработки Visual Studio 2019 и JetBrains Pycharm;
- Язык программирования Python;
- Сайт figma https://www.figma.com/;
- Система контроля версий git;
- Веб-сервис для хостинга GIT-репозиториев GitHub https://github.com/;
- Конвертер текстовых документов Pandoc https://pandoc.org;
- Сервис Supabase
  - auth - авторизация
  - storage - хранение данных

Особенность проекта:

- Интерфейс в стиле Material You
- Крутой, авторский, авангардный и "вкусный" дизайн
- Навигация по страницам;
- Возможность посмотреть контакты разработчиков;
- Возможность посмотреть информацию о сайте;
- Сохранение данных
- Быстрая авторизация

Для использования требуется:

- Python **не ниже 3.7**
- Библиотеки Python
  - Bottle 0.12
  - pypandoc 1.11
  - supabase 1.0.2

Шаги по установке и запуске программы

```
git clone https://github.com/pank-su/report_viewer
cd report_viewer
pip install -r requirements.txt
python installer.py
python app.py
```

Разработчики:

- Василий Панков - главная страница, навигация и логика
- Тимофей Мерлин - страница "О нас";
- Алексей Челноков - страница контактов;

Рисунок 16 – README файл

Листинг программы:
app.py

```
"""
This script runs the application using a development server.
"""

import bottle
import os
import sys

# routes contains the HTTP handlers for our server and must be imported. **NOT DELETE**
import routes

if '--debug' in sys.argv[1:] or 'SERVER_DEBUG' in os.environ:
    # Debug mode will enable more verbose output in the console window.
    # It must be set at the beginning of the script.
    bottle.debug(True)


def wsgi_app():
    """Returns the application to make available through wfastcgi. This is used
           when the site is published to Microsoft Azure."""
    return bottle.default_app()


if __name__ == '__main__':
    PROJECT_ROOT = os.path.abspath(os.path.dirname(__file__))
    STATIC_ROOT = os.path.join(PROJECT_ROOT, 'static').replace('\\', '/')
    HOST = os.environ.get('SERVER_HOST', 'localhost')
    try:
        PORT = int(os.environ.get('SERVER_PORT', '5555'))
    except ValueError:
        PORT = 5555


    @bottle.route('/static/<filepath:path>')
    def server_static(filepath):
        """Handler for static files, used with the development server.
                    When running under a production server such as IIS or Apache,
                    the server should be configured to serve the static files."""
        return bottle.static_file(filepath, root=STATIC_ROOT)


    # Starts a local test server.
    bottle.run(server='wsgiref', host=HOST, port=PORT)
```

routes.py

```
"""
Routes and views for the bottle application.
"""
import os
import shutil
from datetime import datetime

import pypandoc
import storage3.utils
from bottle import route, view, request, response, FileUpload, BaseRequest, redirect,
↪  HTTPResponse
from supabase import create_client, Client

BaseRequest.MEMFILE_MAX = 1024 * 1024  # ограничение по памяти
```

```python
url: str = os.environ.get("SUPABASE_URL")
key: str = os.environ.get("SUPABASE_KEY")

supabase: Client = create_client(url, key)

# переменная, содержащая секретный ключ, используемый для защиты куков
SECRET = os.environ.get("SECRET_TOKEN")
# это переменная, содержащая путь к директории, в которой будут храниться файлы превью
savable_path = "./save/"
temporary_path = "./temp/"


def check_path(path: str) -> None:
    """Функция, которая проверяет есть ли путь, если нет, то создаёт его"""
    if not os.path.exists(path):
        os.makedirs(path)


check_path(savable_path)


@route('/')
@route('/editor')
@view('editor')
def editor():
    query = request.query
    user_id = None
    if request.query_string != "":
        try:
            supabase.auth.refresh_session(query["refresh_token"])
            user_id = supabase.auth.get_user().user.id
        except Exception as e:
            user_id = None
        if user_id is None:
            supabase.auth.set_session(query["access_token"])
    elif user_id is None:
        try:
            user_id = request.get_cookie("user_id", secret=SECRET)
        except Exception as e:
            user_id = None
    if user_id is not None:
        response.set_cookie("user_id", user_id, secret=SECRET)
        try:
            supabase.storage.create_bucket(user_id)
        except storage3.utils.StorageException:
            pass
        storage = supabase.storage.get_bucket(user_id)
        if len(storage.list()) == 0:
            storage.upload("test.org", "./tested_file/test.org")
        return dict(
            year=datetime.now().year,
            userExist=True,
            files=storage.list(),
            user_id=user_id
        )
    else:
        return dict(
            year=datetime.now().year,
            userExist=False,
```

```
            files=[],
            user_id=""

        )


@route('/contact')
@view('contact')
def contact():
    """Renders the contact page."""
    return dict(
        title='Contact',
        year=datetime.now().year
    )


@route('/about')
@view('about')
def about():
    """Renders the about page."""
    return dict(
        title='About us',
        message='Here you can see the possibilities of our site',
        year=datetime.now().year,
        functions='At this site you could:',
        func1='share reports',
        func2='watch reports',
        func3='download reports',
        func4='change report format'
    )


@route('/upload', method='POST')
def do_upload():
    """Обработчик маршрута, которая обрабатывает POST-запрос на загрузку файла. Она
        сохраняет загружженный файл во
            временную директорию, конвертирует его в формат org с помощью pypandoc,
        генерирует случайный хэш-код и сохраняет
            конвертированный файл с использованием этого хэш-кода в постоянную
    ↪   директорию"""
    upload: FileUpload = request.files.get('file')
    check_path(temporary_path)
    upload.save(temporary_path + upload.filename)
    org_text = pypandoc.convert_file(temporary_path + upload.filename, 'org')
    user_id = request.get_cookie("user_id", SECRET)
    save_path = savable_path + user_id + "/"
    if not os.path.exists(save_path):
        os.makedirs(save_path)
    filename = ".".join(upload.filename.split('.')[:-1]) + ".org"
    new_filename = filename
    with open(savable_path + user_id + "/" + filename, "w", encoding="utf-8") as f:
        f.write(org_text)
    shutil.rmtree(temporary_path)
    is_sent = False
    file_id = 0
    while not is_sent:
        try:
            supabase.storage.get_bucket(user_id).upload(new_filename, savable_path +
            ↪   user_id + "/" + filename)
            is_sent = True
```

```
        except storage3.utils.StorageException:
            file_id += 1
            new_filename = ".".join(upload.filename.split('.')[:-1]) +
            ↪  f"_{file_id}.org"
    return "ok"


def generate_preview(user_id, filename):
    """Функция, которая генерирует превью"""
    save_path = savable_path + user_id + "/"
    check_path(save_path)
    with open(save_path + filename, "wb") as f:
        res = supabase.storage.from_(user_id).download(filename)
        f.write(res)
    html_text = "С файлом что-то не так"
    try:
        html_text = pypandoc.convert_file(save_path + filename, 'html')
    except RuntimeError:
        """Если файл не текст"""
        url = supabase.storage.from_(user_id).get_public_url(filename)
        html_text = f"<img src='{url}' />"
    return html_text


@route("/preview")
@view("preview")
def preview():
    return dict(
        previewContent="<p>Site preview here</p>"
    )


@route('/preview/<filename>')
@view("preview")
def preview(filename):
    user_id = request.get_cookie("user_id", secret=SECRET)
    return dict(previewContent=generate_preview(user_id, filename))


@route('/s/<user_id>/<filename>')
@view("preview")
def preview(user_id, filename):
    return dict(previewContent=generate_preview(user_id, filename))


@route("/preview_reload", method='POST')
def preview_reload():
    """Обработчик маршрута, который вызывается при изменении содержимого файла
    ↪  пользователем в редакторе"""
    filename = request.json["filename"]
    data = request.json["data"]
    print(data)
    user_id = request.get_cookie("user_id", secret=SECRET)
    supabase.storage.from_(user_id).remove(filename)
    check_path(temporary_path)
    with open(temporary_path + filename, "w", encoding="utf-8") as f:
        f.write(data)
    supabase.storage.from_(user_id).upload(filename, temporary_path + filename)

    # shutil.rmtree(temporary_path)
```

```
    return "ok"


@route("/content/<filename>")
def get_content(filename: str):
    """Сохраняет файл в директории save_path и возвращает значение"""
    if filename.endswith((".png", ".jpg")):
        return HTTPResponse(status=400, body="This is not text")
    user_id = request.get_cookie("user_id", SECRET)
    save_path = savable_path + user_id + "/"
    check_path(save_path)
    if user_id is None:
        return None

    with open(save_path + filename, "wb") as f:
        res = supabase.storage.from_(user_id).download(filename)
        f.write(res)
    try:
        with open(save_path + filename, "r", encoding="utf-8") as f:
            return f.read()
    except UnicodeDecodeError:
        return HTTPResponse(status=400, body="This is not text")


@route("/logout")
def logout():
    supabase.auth.sign_out()
    response.set_cookie("user_id", "")
    redirect("/")
```

general.css

```
@import url('https://fonts.googleapis.com/css2?family=Roboto&display=swap');
@import url('https://fonts.googleapis.com/css2?family=JetBrains+Mono&display=swap');
@import url('../material-theme/theme.css');

.myAppBar {
    display: flex;
    flex-direction: row;
    align-items: center;

    gap: 6px;
    height: 7%;
}

:root p {
    font-family: "Roboto";
}

.title {
    text-align: center;
    margin: auto;
    padding-right: 24px;
}

body {
    margin: 0;
}

.icon-button {
```

```
    background: transparent;
    border: none;
    height: 48px;
    justify-content: center;
    align-items: center;
    width: 48px;
    display: flex;
    flex-direction: column;
}

.icon-button img {
    height: 46px;
    width: 46px;
    margin: auto;
}

/*Тут будет крутой hovered эффект*/
.icon-button img:hover {
    background: #49454F14;
    background-size: 48px;
    border-radius: 50%;
}

.navigation-rail {
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    padding: 0 0 -56px;
    width: 80px;
    height: 100%;
}

.fab {
    display: flex;
    flex-direction: row;
    align-items: flex-start;
    width: 56px;
    height: 56px;
    margin: 12px;
    border: none;
    border-radius: 16px;
}

.fab img {
    margin: auto;
    width: 24px;
    height: 24px;
}

.fab:hover {
    background: #EFCCB3;

}

.navs {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    width: 56px;
```

```
    margin: 40px 12px 12px;
}

.icon {
    width: 24px;
    height: 24px;
}

.naaav {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin: auto;
}

.nav.active img {

}

.nav p {
    /*font-weight: bold;*/
    margin-top: 10px;
    font-size: 13px;
}

.nav:link {
    text-decoration: none;
}

.nav:visited {
    text-decoration: none;
}

#editor {
    display: flex;
    height: 100%;
    justify-content: center;
    align-items: center;
    width: 100%;
}

#editor form {
    display: flex;
    flex-direction: column;
}


#inputFile::file-selector-button, #submit {
    background: var(--md-sys-color-primary);
    border-radius: 20px;
    color: var(--md-sys-color-on-primary);
    padding: 5px 20px;
    border: none;
}

#inputFile, #submit {
    margin: 10px 0px;
}

#preview {
```

```
    width: 100%;
    height: 100%;
    border: none;
}

.content {
    width: 100%;
    height: 100%;
}

.content div {
    height: 100%;
}

body {
    height: 100vh;
}

iframe::content {
    padding: 0;

}

.area {
    height: 100%;
    width: 100%;
    background: transparent;
    border: none;
    margin: 10px;
    font-family: "JetBrains Mono", serif;
    padding: 0;
    resize: none;
}


#shareButton {
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
    position: absolute;
    z-index: 2;
    bottom: 31px;
    right: 31px;
    width: 56px;
    height: 56px;
    border: none;
    border-radius: 14px;

}

#shareButton img {
    padding-right: 8px;
    height: 24px;
    width: 24px;
}

#profile_img {
    border-radius: 50%;
    width: 100px;
```

```
    height: 100px;
}

#userInfo {
    margin: auto;
}

.twos {
    grid-template-columns: 1fr 1fr;
}

.threes {
    grid-template-columns: 0.5fr 2fr 2fr;
}

#files {
    display: grid;
    /*flex-direction: column;*/
    overflow-y: scroll;
    max-height: 100%;
    gap: 12px;
}

#files::-webkit-scrollbar {
    visibility: hidden;
}

.fileButton {
    height: 32px;
    width: 90%;
    margin: auto;
    border-radius: 8px;
    text-overflow: ellipsis;
    overflow: hidden;
    white-space: nowrap;

    border: 1px solid var(--md-sys-color-outline-light);
}

.selected {
    background: var(--md-sys-color-secondary-container);
}

.filledButton {
    /*height: 40px;*/
    /*width: fit-content;*/
    padding: 10px 24px 10px 24px;
    color: var(--md-sys-color-on-primary);
    border-radius: 100px;
    border: none;
}
```

Контрольные вопросы:

1. Каково основное назначение UML-диаграмм и их роль в разработке приложений?

   Основное назначение UML-диаграмм и их роль в разработке приложений заключается в хорошем проектировании приложения перед его

непосредственным кодированием.

2. Можно ли считать артефактами веб-страницы сайта? Обоснуйте ответ

   Да, веб-страницы сайта можно считать артефактами, так как это информационные элементы, которые тем или иным способом используются при работе программной системы и входят в ее состав.

3. Предпочтительнее ли протокол HTTPS протоколу HTTP? Почему (да / нет)?

   Да, так как протокол HTTPS является расширением протокола HTTP, добавляющий поддержку шифрования в целях повышения безопасности. Также протокол HTTPS является стандартом даже для не самых крупных сайтов сейчас.

4. Что значит код ответа сервера «200»? Какие ещё группы и коды ответов вы знаете (привести по 1-2 примера из каждой группы)?

   Коды сгруппированы в 5 классов:

   (a) Информационные 100 – 199

   101 Switching Protocol "В обработке". Этот код указывает, что сервер получил запрос и обрабатывает его, но обработка ещё не завершена.

   100 Continue "Продолжить". Этот промежуточный ответ указывает, что запрос успешно принят и клиент может продолжать присылать запросы либо проигнорировать этот ответ, если запрос был завершён.

   (b) Успешные 200 – 299

   200 OK "Успешно". Запрос успешно обработан. Что значит "успешно зависит от метода HTTP, который был запрошен.

   202 Accepted "Принято". Запрос принят, но ещё не обработан. Не поддерживаемо, т.е., нет способа с помощью HTTP отправить асинхронный ответ позже, который будет показывать итог обработки запроса. Это предназначено для случаев, когда запрос обрабатывается другим процессом или сервером, либо для пакетной обработки.

   (c) Перенаправления 300 – 399

   301 Moved Permanently "Перемещён на постоянной основе". Этот код ответа значит, что URI запрашиваемого ресурса был изменён. Возможно, новый URI будет предоставлен в ответе.

   304 Not Modified "Не модифицировано". Используется для кеширования. Это код ответа значит, что запрошенный ресурс не был

изменён. Таким образом, клиент может продолжать использовать кешированную версию ответа.

(d) Клиентские ошибки 400 – 499

400 Bad Request "Плохой запрос". Этот ответ означает, что сервер не понимает запрос из-за неверного синтаксиса.

401 "Неавторизованно". Для получения запрашиваемого ответа нужна аутентификация. Статус похож на статус 403, но, в этом случае, аутентификация возможна.

(e) Серверные ошибки 500 – 599

500 Internal Server Error "Внутренняя ошибка сервера". Сервер столкнулся с ситуацией, которую он не знает, как обработать.

504 Gateway Timeout Этот ответ об ошибке предоставляется, когда сервер действует как шлюз и не может получить ответ вовремя.

5. Какую систему защиты и сертификации данных использует Git по умолчанию?

   По умолчанию Git использует для защиты и сертификации данных систему SSH.

6. Что значит «клонировать» репозиторий в терминах Git?

   «Клонировать» репозиторий в терминах Git означает полный перенос удалённого репозитория на локальную машину с помощью команды git clone.