Титульный лист

**СОДЕРЖАНИЕ**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**ВВЕДЕНИЕ**

Разработать сайт – блог для художников и работников сферы дизайна, связанных со статическими изображениями, основанный на Flask.

**ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ**

Разработка ведется на основании официальной документации Flask и системы стандартов ЕСПД

Организация утвердившая документ и дата его утверждения: …

Наименование и(или) условное обозначение темы разработки: web-разработка.

**НАЗНАЧЕНИЕ РАЗРАБОТКИ**

Сайт – блог для художников и работников сферы дизайна, связанных со статическими изображениями.

**ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ**

Сайт должен иметь возможность:

* Регистрации
* Авторизации
* Создания постов
* Просмотра постов

Пользователи должны иметь возможность:

* Ставить/убирать лайки
* Оставлять комментарии
* Удалять собственные посты

Сайт должен обеспечивать сохранность пользовательских данных.

Эксплуатация происходит в режиме реального времени. Требований к информационной и программной совместимости нет (сайту требуется только браузер).

Требований к упаковке и маркировке нет (сайту не требуется упаковка).

Требования к транспортированию и хранению: сайт должен храниться на хостинге pythonanywhere.com или на любом другом хостинге, поддерживающим WSGI-ориентированные Python-приложения.

Специальных требований нет.

**ВВЕДЕНИЕ**

На сегодняшний день сфера дизайна и CG-графики востребована практически во всех областях промышленного дизайна и маркетинга. Художники и дизайнеры обеспечивают привлекательный внешний вид продукта, производимого компанией.

Но как же художнику или дизайнеру реализовать и преподать себя? Для этого нужна платформа, на которой возможно было бы просматривать работы вышеуказанных личностей.

Целью данной работы является создание сайта, где возможно было бы выкладывать работы CG-направленности.

Можно выделить следующие задачи по выполнению цели:

* Разработка верстки сайта
* Нахождение необходимых библиотек
* Разработка backend-а сайта

Характеристики:

…..

…..

…..

Перечень исходных данных: …

Результатов работы программы: …

**ОСНОВНАЯ ЧАСТЬ**

**Описание программы:**

Название разработки: ArtStart

Язык программирования: Python, JavaScript

Основное назначение программы: просмотр и выкладывание постов.

**Функциональные возможности:**

На сайте возможно регистрироваться, авторизовываться, выкладывать посты, просматривать посты, комментировать посты, лайкать посты.

Список функций:

Стандартные:

Функции обертки(@\*.route(), @login\_manager.\*)

Пользовательские:

Функция для генерации слага: slugify()

Функция проверки загружаемых файлов: def allowed\_file(filename)

Функции обработки данных на странице:

* def home()

Функция для просмотра пользователем собственной домашней страницы.

* def edit\_profile()

Функция для редактирования пользователем собственной домашней страницы (возможно изменение описания страницы и аватара)

* def index()(view.py)

Индексная страница проекта(краткое описание сайта)

* def reg\_user():

Функция для регистрации пользователя.

* def logout()

Функция для выхода пользователя из сессии.

* def index()(posts/blueprint.py)

Функция вывода постов на страницу.

* def create\_post()

Функция для создания поста.

* def post\_detail()

Функция для отображения страницы поста

* def add\_like(id)

Функция для обработки AJAX запросов и добавления лайка впоследствии.

* def add\_comment(id)

Функция для обработки AJAX запросов и добавления комментария впоследствии.

* def tag\_detail(slug)

Функция для отображения на странице постов отфильтрованных по тегу.

* def index()(users/blueprint.py)

Функция вывода существующих пользователей.

* def user\_profile(slug)

Функция отображения страницы стороннего пользователя.

**Требования к программе:**

Сайт должен обеспечивать сохранность пользовательских данных, а также предоставлять возможность:

* Регистрации
* Авторизации
* Создания постов
* Просмотра постов

Пользователи должны иметь возможность:

* Ставить/убирать лайки
* Оставлять комментарии
* Удалять собственные посты

**Структура приложения:**

**Классы и компоненты:**

Классы:

* class AdminMixin

Класс-шаблон для администрирования.

* class AdminView

Класс для входа в режим администрирования.

* class HomeAdminView

Класс для определения компетенции администратора .

* class Post

Модель поста для ORM.

* class User

Модель пользователя для ORM.

* class Picture

Модель изображения для ORM.

* class Comment

Модель комментария для ORM.

* class Tag

Модель тэга для ORM.

* class Role

Класс роли для ORM

Библиотеки:

* alembic==1.2.1

Взаимодействие с миграциями БД

* Babel==2.7.0

Локализация интерфейсов библиотек

* bcrypt==3.1.7

Хорошее хэширование паролей для вашего программного обеспечения и серверов

* Flask-Admin==1.5.4

Расширение администрирования

* Flask-BabelEx==0.9.3
* Flask-JSON==0.3.4

Flask-JSON-это простое расширение, которое добавляет лучшую поддержку JSON в приложение Flask.

* Flask-Login==0.4.1

Flask-Login обеспечивает управление сеансами пользователя для Flask.

* Flask-Mail==0.9.1

Расширение Flask-Mail предоставляет простой интерфейс для настройки SMTP с вашим приложением Flask и отправки сообщений из ваших представлений и скриптов.

* Flask-Migrate==2.5.2

Flask-Migrate-это расширение, которое обрабатывает миграции базы данных SQLAlchemy для приложений Flask с использованием Alembic. Операции с базой данных становятся доступными через интерфейс командной строки Flask или через расширение Flask-Script.

* Flask-Script==2.0.6

Расширение Flask-Script обеспечивает поддержку написания внешних скриптов в Flask. Оно включает в себя запуск сервера разработки, настраиваемой оболочки Python, скриптов для настройки базы данных, cronjobs и других задач командной строки, которые принадлежат за пределами самого веб-приложения.

* Flask-Security==3.0.0

Flask-Security позволяет быстро добавлять общие механизмы безопасности в приложение Flask.

* Flask-SQLAlchemy==2.4.1

Flask-SQLAlchemy является расширением Flask, в которую добавлена поддержка с SQLAlchemy. Оно направлено на упрощение использования SQLAlchemy с Flask, предоставляя полезные возможности по умолчанию и дополнительные возможности, которые облегчают выполнение общих задач.

* Jinja2==2.10.3

Это шаблонизатор для языка программирования Python.

* Mako==1.1.0

Mako — это стандартный шаблонизатор для фреймворка [Pylons](http://docs.pylonsproject.org/projects/pylons-webframework/en/latest/), написанный Майком Байером

* MarkupSafe==1.1.1

MarkupSafe реализует текстовый объект, который экранирует символы, поэтому его можно безопасно использовать в HTML и XML. Символы, имеющие специальные значения, заменяются так, чтобы они отображались как фактические символы. Это смягчает инъекционные атаки, что означает, что ненадежный пользовательский ввод может безопасно отображаться на странице.

* passlib==1.7.1

Passlib-это библиотека хэширования паролей для Python 2 & 3.

* Pillow==6.2.0

PIL-это библиотека изображений Python, созданная Фредриком Лундом и другими участниками.

* pillowcase==2.0.0
* pycparser==2.19

pycparser-это полный синтаксический анализатор языка C, написанный на чистом Python с использованием библиотеки парсинга PLY.

* python-dateutil==2.8.0

Модуль dateutil предоставляет мощные расширения для стандартного модуля datetime, доступного в Python.

* python-editor==1.0.4

python-editor это библиотека, которая предоставляет модуль редактора для программного взаимодействия с $EDITOR вашей системы.

* pytz==2019.3

Эта библиотека позволяет выполнять точные и кросс-платформенные вычисления часовых поясов с использованием Python 2.4 или выше.

* six==1.12.0

Библиотека предоставляет служебные функции для сглаживания различий между версиями Python с целью написания кода Python, совместимого с обеими версиями Python.

* speaklater==1.3
* SQLAlchemy==1.3.10

[SQLAlchemy](https://ru.wikipedia.org/wiki/SQLAlchemy) — это программное обеспечение с открытым исходным кодом для работы с базами данных при помощи языка [SQL](https://ru.wikibooks.org/w/index.php?title=SQL&action=edit&redlink=1). Оно реализует технологию программирования [ORM](https://ru.wikipedia.org/wiki/ORM) (Object-Relational Mapping), которая связывает базы данных с концепциями объектно-ориентированных языков программирования. SQLAlchemy позволяет описывать структуры баз данных и способы взаимодействия с ними прямо на языке [Python](https://ru.wikibooks.org/wiki/Python" \o "Python). SQLAlchemy реализована в виде пакета для Python под лицензией MIT, а значит возможно ее использование в [проприетарном ПО](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5" \o "w:Проприетарное программное обеспечение).

* Werkzeug==0.16.0

Werkzeug-это всеобъемлющая библиотека веб-приложений WSGI (маршрутизатор).

**Алгоритмы:**

Описание алгоритмов будет проводиться по конкретным url страниц:

* /posts

Индексная страница, на которой происходит вывод постов.

1. Программа берёт данные постов из базы данных (а конкретно, изображение поста (при желании возможно выводить любую другую информацию) и через цикл выводит их на страницу.
2. Далее для каждого из постов автоматически генерируется ссылка для перехода.

* /posts/create

1. Пользователь вводит данные в предоставленную страницей форму, из которой, впоследствии берутся данные и отправляются на сервер.
2. Сервер обрабатывает данные, формирует из них экземпляр класса поста, отправляет в базу данных и производит коммит.
3. Данные сохраняются в базу данных и далее выводятся на всех страницах, где они потребуются.

* /posts/<id>

На странице присутствуют функции удаления поста (если авторизованный пользователь является автором поста), добавление комментария к посту (если пользователь авторизован), возможность поставить лайк (если пользователь авторизован).

1. Переход осуществляется по ссылке, которая генерируется из id поста.
2. Осуществляется запрос в базу данных.
3. Из пришедших данных формируется пост.

* /posts/<id>/delete

1.При нажатии на кнопку удаления поста производится запрос к базе данных для поиска поста по id текущего поста.

2.Далее производится дополнительная проверка авторства текущего пользователя по отношению к посту.

3.Удаляется изображение поста.

4.Удаляется экземпляр поста из базы данных.

* /posts/<id>/add\_like

1.При нажатии на кнопку добавления лайка на странице поста (/posts/<id>) производится AJAX-запрос на сервер.

2.К базе данных производится запрос для поиска поста по id текущего поста.

3.Проверяется, авторизован ли пользователь.

4.Если пользователь авторизован, то к свойству поста, отвечающего за лайки прибавляется 1.

5.Создаётся строка в реляционной таблице, отвечающей за связь пользователя и поста (для проверки, проставлял ли текущий пользователь лайк посту или нет)

6.Если строка уже создана, то она удаляется из базы данных.

7.Производится коммит базы данных.

8.Упаковывается JSON-ответ и отправляется на страницу поста(для динамического отображения страницы).

* /posts/<id>/add\_comment

1.При нажатии на кнопку добавления комментария на странице поста (/posts/<id>) производится AJAX-запрос на сервер.

2.К базе данных производится запрос для поиска поста по id текущего поста.

3.Проверяется, авторизован ли пользователь.

4.Если пользователь авторизован, то свойству поста, в котором хранится список комментариев, добавляется данные текущего комментария.

5.Далее производится коммит базы данных.

6.Упаковывается JSON-ответ и отправляется на страницу поста(для динамического отображения страницы).

* posts/tag/<slug>

1. При нажатии на странице поста на лейбл тега поста происходит переход на текущий url.
2. Далее производится выборка постов по тегу.
3. По произведённой выборке происходит последующий вывод постов и генерация url для каждого поста.

**Анализ программы:**

**Инструкция пользователя:**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Pillow [Электронный ресурс]-режим доступа: <https://pillow.readthedocs.io/en/stable/>
2. Flask-Security [Электронный ресурс]-режим доступа: <https://pythonhosted.org/Flask-Security/>
3. Flask-Login [Электронный ресурс]-режим доступа: <https://flask-login.readthedocs.io/en/latest/>
4. Flask’s documentation [Электронный ресурс]-режим доступа: <https://flask.palletsprojects.com/en/1.1.x/>
5. Bootstrap 4 компоненты [Электронный ресурс]-режим доступа: <https://bootstrap-4.ru/docs/4.3.1/components/alerts/>
6. ГОСТ 19.201 – 78
7. ГОСТ 19.105-78 [Электронный ресурс]-режим доступа: <http://docs.cntd.ru/document/gost-19-106-78-espd>
8. Стандарты ЕСПД [Электронный ресурс]-режим доступа: http://www.swrit.ru/gost-espd.html