Введение.

В качестве моего первого языка программирования был выбран один из самых популярных на сегодняшний день языков программирования Python - [высокоуровневый язык программирования](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения с [динамической](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [строгой](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%B3%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости [кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью [объектно-ориентированным](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) в том плане, что всё является [объектами](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (https://ru.wikipedia.org/wiki/Python). Это интерпретируемый язык, а не компилируемый, как C++ или Java. Программа на Python представляет собой обычный текстовый файл. Код можно писать практически в любом редакторе или использовать специальные IDE. Python можно встретить почти везде: в вебе, мобильных и десктопных приложениях, а также в играх. На нём пишут нейросети, проводят научные исследования и тестируют программы.

- Чаще всего Python используют в веб-разработке. Для него написано множество фреймворков: FastAPI, Flask, Tornado, Pyramid, TurboGears, CherryPy и, самый популярный, Django ( <https://skillbox.ru/media/code/dlya_chego_nuzhen_python/>).

- Благодаря лаконичности, простому синтаксису и богатому набору инструментов Python стал любимым языком учёных. Есть специальные библиотеки для Python, которые позволяют строить графики, проводить исследования и вычисления.

- Ещё одна область применения Python — автоматизация тестирования. Многие специалисты по автоматизации QA выбирают Python из-за его простоты. Он отлично подходит тем, кто имеет небольшой опыт в разработке приложений. Развитое сообщество, логичный синтаксис и удобочитаемость упрощают процесс обучения.

- Хоть язык не компилируется, с его помощью всё же можно создать десктопные программы.

- Мобильная разработка на Python менее популярна. Для Android чаще используют Java, C#, C++ или Kotlin, а для iOS — Swift или Objective-C. На Python обычно программируют серверную часть приложения. Например, клиент «Инстаграма»\* для iOS написан на Objective-C, а сервер — на Python. Тем не менее у Python есть фреймворки для разработки кросс - платформенных мобильных GUI-приложений: Kivy и BeeWare.

- Многие компьютерные игры полностью или частично написаны на Python.

В рамках обучения на образовательной платформе GeekBrains разработан проект - блог на языке программирования Python (специального сайта с контентом) для общения между пользователями. Так как целью дипломного проекта является создание блога, то немного расскажем что это такое и для чего он нужен. Слово **блог (blog)** - это сокращение от английского **web log**, что дословно означает сетевой журнал, интернет-дневник, онлайн-дневник. В широком смысле блогом считается часто обновляемая площадка в интернете. Это может быть сайт, который с определенной периодичностью наполняется контентом, группа в социальной сети или же ваш личный профиль, где вы делитесь своими мыслями или полезными материалами с другими людьми. (<https://mariyaalbi.com/prichini-sozdat-i-vesti-blog.html>). Приведем несколько причин для чего нужен блог: это отличная возможность заявить о себе широкой публике, может служить огранкой личного бренда, «открытая» книга личного творчества, инструмент для создания и развития собственного онлайн и офлайн-бизнеса, площадка для развития и демонстрации своих талантов и т.д.

**Архитектура проекта.**

В процессе анализа поставленной задачи был определен к реализации общий функционал проекта:

1. Регистрация нового пользователя
2. Авторизация и аутентификация пользователей (с помощью расширения Flask-Login)
3. Редактирование профиля пользователя (смена имени пользователя, изменение email’а, добавление изображения профиля)
4. Создание и редактирование статей пользователя (написание заголовков и текстов статей, а так же добавление изображения к ним)
5. Просмотр зарегистрированными пользователями всех статей блога, а также статей отдельного автора
6. Оставление комментариев и “лайков” к постам, а также возможность удаления своего комментария и “отозвать лайк”

В качестве основы для реализации функционала приложения использовался легковесном веб-фреймворке Flask для языка Python обладающий рядом особенностей:

**- минимальный набор инструментов из коробки.** Причём они не навязывают какую-то архитектуру или жёсткую структуру проектов. Разработчики сами решают, как и что они будут создавать.

**- гибкость.** Работая с Flask, программист может выбирать только необходимые встроенные инструменты и подключать дополнительные внешние, не перегружая проект лишними модулями.

**- расширяемость.** У Flask много расширений и плагинов, которые помогают быстро добавить новую функциональность. Например, авторизацию, управление базами данных и работу с формами.

**- простота.** У Flask простой синтаксис, что делает изучение этого фреймворка более простым, а также позволяет быстрее создавать прототипы веб-приложений.

В проекте Flask, как базовый фреймворк, функционирует в роли веб-сервера, обрабатывающего HTTP-запросы и возвращающего HTTP-ответы. Flask использует механизм маршрутизации для управления потоком данных при входящих HTTP-запросах. Благодаря этому возможно определять различные URL-шаблоны и связать их с функциями, которые будут обрабатывать соответствующие запросы. Фреймворк Flask поддерживает использование шаблонов Jinja2 для создания веб-страниц. Шаблоны представляют собой файлы, содержащие HTML-разметку с использованием специальных тегов и переменных, которые могут быть заменены на данные во время выполнения. Для обработки входящих запросов и возвращения соответствующих ответов во Flask существуют специальные функции - контроллеры. Они могут вызываться в ходе маршрутизации и могут включать в себя логику для доступа к данным из базы данных, обработки форм, отправки электронной почты и т. д.

Работы с веб-формами обеспечивается модулем Flask-WTForm - это расширение [Flask](https://translated.turbopages.org/proxy_u/en-ru.ru.ac5da8f8-65808bab-18b8d7b6-74722d776562/https/www.geeksforgeeks.org/python-introduction-to-web-development-using-flask/), интегрирующее библиотеку WTForms, которая предоставляет полезные функции для простого создания и обработки форм в веб-приложении Flask. Эта библиотека для работы с формами и полями форм, представляющая собой гибкую платформу для создания форм, обработки проверки и рендеринга форм в HTML (www.geeksforgeeks.org/flask-wtf-explained-how-to-use-it/).

В качестве способа организации кода в приложении на Flask используем Blueprint, который позволяет создавать независимые модули приложения. Каждый модуль может иметь свой собственный маршрутизатор, шаблоны, статические файлы и другие ресурсы. Это позволяет легко масштабировать приложение и делить его на логические компоненты Blueprints также упрощают тестирование и отладку приложения, так как каждый модуль может быть протестирован и отлажен отдельно ( https://flask.ivan-shamaev.ru/approaches-to-development-flask-app-in-python/).

Аутентификация пользователей реализуется модулем Flask-Login который обеспечивает управление сеансами пользователей для Flask. Он выполняет обычные задачи по входу в систему, выходу из системы и запоминанию сеансов ваших пользователей в течение длительных периодов времени (https://flask-login.readthedocs.io/en/latest/index.html#custom-login-using-request-loader), что позволяет:

* Сохраняйте идентификатор активного пользователя в сеансе Flask и позволяйте ему легко входить в систему и выходить из нее.
* Позволяет ограничить просмотр только вошедшими в систему (или вышедшими из системы) пользователями. ([login\_required](https://flask-login.readthedocs.io/en/latest/index.html" \l "flask_login.login_required" \o "flask_login.login_required))

Хранение и управление данными блога обеспечивается базой данных SQLite - встраиваемая кроссплатформенная БД, написанная на языке C (ANSI-C), которая реализует движок для работы с реляционными БД. Основное отличие SQLite от других систем управления БД, таких как MS SQL Server, MySQL и Postgres, заключается в том, что для работы с ней не требуется отдельный сервер БД. Эта СУБД представляет собой встраиваемый движок, который напрямую работает с файлом базы данных на диске (все данные хранятся и обрабатываются на одном устройстве). Поэтому нет необходимости устанавливать или конфигурировать сервер SQLite для работы с БД.

Flask имеет множество ORM-библиотек, таких как SQLAlchemy, Peewee, Pony ORM и Django ORM. **Object-Relational Mapping** - это технология, которая позволяет работать с базами данных, используя объектно-ориентированный подход. ORM-библиотеки позволяют разработчикам использовать объекты и методы, а не SQL-запросы, для работы с базами данных. Одним из основных преимуществ **ORM** является то, что она позволяет разработчикам работать с базами данных на более высоком уровне абстракции. Вместо того, чтобы писать SQL-запросы, разработчик может использовать объекты, методы и свойства для взаимодействия с данными. Это позволяет уменьшить количество ошибок и сократить время разработки. **ORM** также позволяет разработчикам создавать более гибкие и масштабируемые приложения. Они могут изменять структуру базы данных, не затрагивая код приложения, так как ORM библиотека позволяет автоматически создавать запросы на основе изменений в моделях данных. Это облегчает процесс разработки и позволяет более быстро реагировать на изменения требований к приложению. .(<https://flask.ivan-shamaev.ru/approaches-to-development-flask-app-in-python/>). В нашем приложении для взаимодействия Flask с базой данных используется расширение Flask-SQLAlchemy. Оно предоставляет удобный интерфейс для работы с базами данных, а также мощные инструменты для создания и управления моделями данных.

Отображение пользовательского интерфейса реализовывалось с помощью ……