Проект на тему:

“Создание сайта на языке программирования Python и фреймворке Flask”.

Чекашов Матвей

10А класс

ВОЛГОГРАД 2021

Оглавление

Введение в проект на тему:

“Создание сайта на языке программирования Python и фреймворке Flask”

Цель проекта: создать сайт тема сайта, объяснить принцип его создания в фреймворке Flask.

Задачи проекта:

1) Рассказать о веб-разработке.
2) Рассмотреть основные архитектурные особенности веб-приложений.
3) Рассмотреть и объяснить всю важность соблюдения всех принципов веб-разработки.
4) Сравнить фреймворк Flask с его ближайшими конкурентами и объяснить почему я выбрал именно Flask.
5) Создать сайт.

Объектом проекта является процесс получения и последующего применения теоретических знаний на практике.
Предметом проекта является создание сайта.

Актуальность: Веб-разработка является самым популярным направлением в программировании, а также одним из самых востребованных и хорошо оплачиваемых. В современном мире каждому предпринимателю нужен сайт для продвижения своих услуг из-за этого количество вакансий на веб-разработчиков постоянно увеличивается и их востребованность также растёт. Уметь создавать веб-приложения сегодня это гарантированный способ получить интересную, высокооплачиваемую работу и я в своём проекте постараюсь рассказать про те навыки, которые гарантировано потребуются в этом не лёгком деле.

Теоретическая значимость проекта: Данные из этого проекта (Как теоретическую так и практическую часть), можно использовать в качестве примера того, с чего начать в разработке веб-сайтов.

Основная часть. Теория

О веб-разработке

Веб-разработка на сегодняшний момент, это самая популярная и быстрорастущая сфера в программировании.

Веб-разработка — *процесс создания веб-сайта или веб-приложения. Основными этапами процесса являются веб-дизайн, вёрстка страниц, программирование части на стороне клиента(frontend) и сервера(backend)*.
На сегодняшний момент для создания frontend“а(То есть клиентской части сайта) используют JavaScript и его фреймворки, такие как VueJS, AngularJS, ReactJS.
Процесс создания клиентской части без фреймворков JS, называется вёрсткой веб-страниц.
Но если на клиентской части всё просто, там есть только JS(В некоторых случаях TypeScript), то на серверной части(BackEnd) всё гораздо сложнее.
Но сначала нужно понять, зачем вообще нужна серверная часть, ведь есть HTML и CSS! Но, HTML как бы это грустно не было, не явдяется языком программирования, ровно как и CSS и на них невозможно сделать серверную часть сайта. Но что же такое эта „Серверная часть“?
Серверная часть(BackEnd) — Программа написанная на языке программирования и отвечающая за логику выполнения сайта, к примеру, клиентская часть, это все кнопки,

фотографии, текст, да вообще всё что вы видите на Веб-Странице. BackEnd это программа, которая взаимодейтсвует с клиентской частью и позволяет к примеру зарегистрироваться, оставить комментарий или лайкнуть запись, то есть придать любому действию клиентской части логику. Без backend“а кнопки лайка бы не работали, нельзя было бы зарегистрироваться или восстановить утерянный пароль.

В моём проекте пойдёт речь именно о серверной веб-разработке.
Для backenda можно использовать любой язык программирования, но лишь не многие действительно для этого предназначены и хорошо с этим справляются. Я выделил пару таких языков: PHP, Python, Ruby, Java, C#.

Python почему он?

Почему выбран именно Python? Давайте разберёмся.

Изначально, надо понять какие языки для каких задач подходят:

1) На чистом PHP разрабатываются небольшие сайты.
2) На CMS от PHP разрабатываются все решения ниже уровня e-commerce, простые интернет-магазины, главные странички кофеен, бистро и другого.

3) На фреймворках PHP разрабатываются e-commerce решения, пример фреймворков: Laravel, YII, Symfony.
4) На Python разрабатываются только e-commerce решения, такие как сайты спорт клубов, интернет-провайдеров, репетиторских или туристических агенств. И тп.
5) На Rube разрабатываются только e-commerce решения.

6) На Java разрабатываются сложные Enterprice решения для огромных корпораций.

7) На C# разрабатываются сложные Enterprice решения для огромных корпораций, C# является более современной версией Java.

Но почему всё таки Python, ведь тот же PHP популярнее него? Всё очень просто, PHP программистов больше чем Python программистов и это пораждает большую конкуренцию(На начальном этапе), к тому же на PHP можно писать существенно хуже чем на его ближайших „Соседях“, ну и в заключении мне не нравится синтаксис PHP.

С Ruby дела проще, он очень похож на Python, но существенно менее популярный, найти вакансию на Ruby, если и не большая удача, то уж точно удачное стечение обстоятельств.

Веб-разработка на Python Flask

Хорошо, мы определились с языком, давайте теперь разберёмся, почему именно Flask, ведь есть более популярный Django?
Это трудный вопрос на который нельзя ответить однозначно, могу лишь сказать своё субъективное мнение, мне Django кажется излишне сложным и „Тяжёлым“, чтобы написать даже не большое веб-приложение на Django потребуется сделать много лишних действий которых делать крайне не хочется. Flask же напротив, лёгкий, быстрый и не сложный в изучении фреймворк, который приятно минималистичен не только по своему синтаксису, но и по предоставляемым в комплекте с фреймворком дополнений.

Но конечно, в больших приложениях используется именно Django, так как в нём создавать правильную архитектуру веб-приложений можно прямо из „Коробки“. Вот примеры сайтов написанных на Django:

1) YouTube.
2) Google Search
3) DropBox
4) Instagram
5) Reddit
6) Yahoo Maps
7) Spotify
8) Mozilla
9) NASA
10) National Geographic

Это действительно огромные проекты которым может похвастаться не только Django, но и всё Python сообщество.

Flask и его особенности. Паттерны архитектуры

Изначально, рассмотрим чем вообще отличается фреймворк от библиотеки. И наверное главное отличие, это синтаксис.
Библиотека не изменяет синтаксис самого Python, а добавляет в него нечто новое, фреймворк напротив, меняет синтаксис самог языка, диктуя свои условия написания кода.

И главная особенность Flask, это его минималистичность. Благодаря этому, он не навязывает тебе какие либо конкретные архитектурные решения и требования, а даёт пространство творчеству. Также Flask работает с набором инструментов Werkzeug, а также шаблонизатором Jinja2.
У Flask есть огромное кол-во расширений таких как:
Flask-SQLAlchemy, Flask-Login, Flask-WTF, Flask-Security, Flask-Script, Flask-Mail и многие другие.
Каждое из этих дополнений, делает Flask ещё более удобным инструментом.
Но прежде чем рассматривать мой сайт, необходимо рассмотреть некоторые важные моменты, такие как: ООП, ORM, Архитектура, Паттерны.

Паттерны архитектуры(О самих паттернах и что это, поговорим чуть позже)**.**
MVC давно уже на рынке, но многие его используют по разному.
Проблема в том, что многие разработчики очень сильно усложняют код и его дорого обслуживать и развивать или приходится делать глобальный рефакторинг, что не выгодно для бизнеса. Основная цель — это выполнить проект в сроки. Один из важных критериев качества кода — это его простота. Но как измерять простоту? Один из вариантов — это рассчитать кол- во элементов системы. Чем меньше элементов тем система проще.

Поэтому, пришла MVC(M-Models, V- Views, C-Controller) архитектура. Она заметно облегчает жизнь не только программисту, но и заказчику, за счёт упрощения кода.

Но есть и другие виды архитектур такие как:

1)MVVM(Model-View-ViewModel)
2)MVP(Model-View-Presenter)

Также, хотел бы отметить, что строго придерживаться только одному паттерну — не всегда лучший выбор.

ООП или 90'е?

Все говорят и знают, что Python — ООП язык, но некоторые не знают что это значит, давайте разберёмся в чём дело.
Начнём с самого простого, нам всем объясняли алгоритмы и думаю это выглядело так.
Как приготовить борщ? Встать, пойти в магазин, купить продукты, придти, достать кастрюлю, сготовить борщ.
И так было и в программировании, когда в коде были только команды и их вызов, но ещё в конце 90х, поняли что это не правильно. В таком алгоритме, получалось, что мы должны мыслить как и машина последовательными командами.
Такое программирование получило название „процедурное программирование“, но мы не машины, а люди, мы мыслим объектами.
Концепция ООП к этому и сводится, у нас есть какой-то объект и у него есть какие-то свойства и возможности. К примеру машина, у неё есть колёса, кузов, двигатель, руль и есть действие — ехать.
В ООП программировании существуют классы которые и представляют сами объекты реального мира. Многие эксперты считают, что изучение программирования начинается только после изучения ООП.
После того как только появилось ООП, оно сразу завоевало популярность, код стал на порядки проще и понятнее людям и наш алгоритм будет выглядеть уже по другому.

Получаем объект класса Человек и „Говорим“ ему, Человек.СваритьБорщ и получаем уже готовый объект класса борщ, таким образом мы сильно упростили алгоритм и сделали его более понятным.
К сожалению, в школах и большинстве ВУЗов, изучается именно процедурное программирование которое бросили в конце 90х и это порождает ужасный, не читаемый и не поддерживаемый код.
Концепция ООП крайне важна и также сложна, у неё есть свои принципы которые нельзя нарушать: Наследование, Абстракция, Инкапсуляция, Полиморфизм, но это слишком большие темы для одного проекта, о них можно вести долгие, многочасовые дискуссии.

*Есть ещё несколько видов программирование такие как:

Операторное, функциональное, логическое, структурное (модульное), визуально-ориентированное.

Но все эти виды, не столь популярны(За исключением функционального программирования) и по этому мы не станем о них говорить.

ORM

После того, как мы разобрались с концепцией ООП, можно рассмотреть её производные, а именно ORM.
ORM(Object-Relational Mapping) — в дословном переводе - объектно-реляционное отображение, или преобразование.
А именно, технология связывающая ваш код и Базу Данных(БД).

Раньше, когда не было технологии ORM, обращение к БД происходило с помощью SQL запросов, но с приходом ООП, когда все начали избавляться от процедурного программирования, то изменили и концепцию связи с БД.

Теперь, вместо прописывания вручную каждого запроса и получения данных в формате списков, чисел или строк, мы получаем не что иное, как объект класса. Рассмотрим на примере, мы хотим получить пользователя который создал статью, в обычном SQL запросе, мы получили бы список данных о нём, а именно id, имя, email и тп., а в ORM, мы получим объект класса пользователей, у которого уже можно будет взять необходимую информацию.

Flask поддерживает технологию ORM и для этого у него есть дополнение Flask-SQLAlchemy, которое реализует все функции и главное не отходит от принципов ООП.

На основе этого дополнения, сделаны и другие дополнения, такие как к примеру Flask-Login, которое позволяет регистрировать пользователей в системе.

Понять более углубленно, для чего нужна ORM, можно если хорошо изучить ООП, и написать свой сайт с использованием сначало SQL запросов, а потом и ORM.

Принципы разработки

В любом языке программирования существуют свои нормы и правила, которым придерживаются все разработчики. В Python это PEP8, свод основных правил по написанию кода.
Многие думают, что только эти правила гарантируют хороший код, но это совершенно не так, на хороший код в большей степени влияют „Принципы разработки“.
Рассмотрим основные из них:

DRY(Don't Repeat Yourself - Не повторяйся) — Это принцип разработки, говорящий о необходимости не повторяться. Т.е.
Если у вас есть более одного повторяющегося блока кода, то его нужно вынести в отдельный метод или класс.

KISS(Keep it simple, stupid - Будь проще, глупый) — Принцип KISS утверждает, что большинство систем работают лучше всего, если они остаются простыми, а не усложняются. Поэтому в области проектирования простота должна быть одной из ключевых целей, и следует избегать ненужной сложности.

YAGNI(You aren't gonna need it - Вам это не понадобится) — Принцип, объясняющий, что не следует реализовывать то, что не было прописано в требованиях заказчика.
Делать следует то и только то, о чём просил ваш заказчик, ни больше и не меньше.

Также, нельзя не отметить принципы, которые выразил Роберт Мартин(Является профессионалом в области разработки ПО с 1970, а с 1990 становится международным консультантом в этой области). Он выразил огромное количество принципов хорошего кода, но первые пять принципов были выделены отдельно и получили аббревиатуру SOLID(Первая буква первого слова каждого из принципов).

SOLID:
S — SRP(Single Responsibility Principle — Принцип единой ответственности) — класс или функция, должны реализовывать что либо одно, а не выполнять множество действий. Одна функция — одно действие.
O — OCP(Open Closed Principle — Принцип открытости, закрытости)—Программные сущности (классы, функции) должны быть открыты для расширения, но закрыты для изменения.

L - LSP(The Liskov Substitution Principle - Принцип подстановки Барбары Лисков) — Объекты в программе должны быть заменяемыми на экземпляры их подтипов без изменения правильности выполнения программы.
I - ISP(The Interface Segregation Principle - Принцип Разделения Интерфейса) — Много интерфейсов, специально предназначенных для клиентов, лучше, чем один интерфейс общего назначения.
D - DIP(The Dependency Inversion Principle - Принцип инверсии зависимостей) — Зависимость на Абстракциях. Нет зависимости на что-то конкретное. „Общение“ классов должно происходить через интерфейс.

Это лишь часть принципов, но следуя даже этим, ваш код будет приближен к идеалу.

Паттерны проектирования

Паттерны(образец, шаблон.) - это тема не для новичков, а для очень серьёзных разработчиков, но всё же, предлагаю немного коснуться этой темы, что бы понимать что к чему.

Представьте, что вы хотите сделать новый автомобиль. Сколько колес и почему вы спроектируете для него? Сейчас вы уже скорее всего скажете что 4, однако почему не 3, 5, 10, 20? Потому-что практикой использования уже было выяснено, что обычные автомобили лучше всего делать на 4-х колесах — это шаблон проектирования сформированный временем. Именно такому же подходу и служат паттерны в ООП и вы не столкнетесь с ними в разработке до тех пор, пока вам не потребуется «сделать автомобиль». Однако иногда случается так, что вы создаете «трицикл», и только потом, набив несколько шишек с его устойчивостью и неудачным вписыванием в колею на дороге, узнаете что существует паттерн «автомобиль», который значительно упростил бы вам жизнь. Паттерны, не привязаны только к одному языку.

Предлагаю кратко рассмотреть Grasp паттерны проектирования.
GRASP (General Responsibility Assignment Software Patterns) — шаблоны проектирования, используемые для решения общих задач по назначению обязанностей классам и объектам.
Известно девять GRAPS шаблонов, описанных в книге Крейга Лармана «Применение UML и шаблонов проектирования». Не имеют выраженной структуры, четкой области применения и конкретной решаемой проблемы, а представляют собой обобщенные принципы, при проектировании системы.

Перечислим основные GRASP шаблоны: Предметная область, Информационный эксперт, Низкая связанность,
Высокое зацепление, Устойчивый к изменениям, Контроллер, Полиморфизм, Чистая выдумка, Перенаправление.

Опишу только, как мне кажется фундаментальный Grasp паттерн.

Information expert(Информационный эксперт) - Зачем нам нужен информационный эксперт? Затем, что если объект владеет всей нужной информацией для какой-то операции или функционала, то значить и этот объект будет выполнять либо делегировать выполнение этой операции.
Итак рассмотрим пример. Есть некая система продаж. И есть класс Sale (продажа). Нам необходимо посчитать общую сумму продаж. Тогда кто будет считать общую сумму по продажам? Конечно же класс – Sale, потому что именно он обладает всей информацией необходимой для этого.

Сайт на Flask

В заключительной части своего проекта, я применил все знания полученные в ходе работы и создал сайт на Flask.

При создании сайта использовались такие библиотеки и модули:

Flask, flash, redirect, render_template, request, url_for, session, escape, make_response
flask_sqlalchemy, flask_login
loguru, hashlib, email.mime.multipart, email.mime.text
re, os, smtplib, random, datetime

Сайт построен на паттерне MVC, с учитыванием всех принципов DRY, KISS, SOLID.

Для дизайна, по мимо стандартного CSS использовалась библиотека Bootstrap, для добавления адаптивного дизайна(То есть одинаково-работающего и на смартфонах и на компьютерах).

В качестве базы данных, используется MySQL и драйвер mysqlconnector.

Также, сайт подключен к Яндекс.Метрике, для отслеживания посещений и активности на сайте.

Для реализации связи с БД(Базой данных) использовалось система ORM, которая во Flask“е подключается с помощью Flask-SQLAlchemy.

Для обеспечения большей безопастности пользователей, пароли хранятся в БД не в открытом виде, а в зашифрованном с помощью шифрования sha256. А для регистрации, необходимо подтвердить почту с помощью кода отправленного на почту.

Основная часть. Практика

Начало работы. Установка зависимостей

Итак, когда теория изученна, можно перейти к созданию сайта на Flask.

Прежде всего надо установить библиотеки которые понадобятся в ходе написания веб-сайта, а именно: Flask, flask_sqlalchemy, flask_login, loguru, hashlib, email.mime.multipart, email.mime.text. Остальные бибилотеки уже идут в стандартную сборку Python и их не нужно ставить отдельно.

Теперь можно приниматься за работу.

Выбор стека разработки

Как я уже ранее говорил, все(Или скорее все более-менее серьёзные) проекты пишутся с использованием какого-либо архитектурного паттерна.

Несколько таких я называл(MVVM, MVP, MVC).

Я выбрал наиболее подходящий и наиболее популярный из них, а именно MVC.

Модель MVC(т.е. Model, Views, Controller) я реализовал следующим образом:

Model(M) - у меня models.py

View(V) - у меня templates

Controller(C) - у меня routes.py\.

Почему же не совпадают названия? Всё на самом деле очень просто.

Дело в то, что во Flask используется шаблонизатор Jinja2, который изначально ищет все шаблоныв папке templates(Шаблоны - динамически изменяемые HTML документы).

А название routes, потому что я в нём реализую маршрутизацию(Т.е. роутинг) запросов.

Страницы сайта и его функции

Я решил сделать новостной портал, вот с такими страницами:

Главная страница(На ней выводится список всех новостей),

Страница просмотра новостей(На ней можно прочесть новость),

Страница с рейтингом языков программирования,

Страница со списком событий сайта(Новый дизайн, функции и тп),

Страница авторизации,

Регистрации,

Подтверждения почты,

Изменения пароля,

Страница со своим профилем,

Страница с чужим профилем(Просмотр чужого профиля),

Страница для написания своих собственных новостей,

Админ-панель.

Главная

Работа с пользователями(Блокировка или запрет создавать/комментировать новости)

Работа с Администрацией сайта(Добавлять, удалять Администраторов)

Работы с событиями(Редактирование, добавление, удаление событий)

Работа с рейтингом(Добавлять или удалять языки программирования из рейтинга)

Логи сайта(Просмотр логов сайта).

Структура Базы Данных(БД)

На моём, как и на любом другом сайта, реализующего сохранение, хранение данных, используется База Данных(Далее БД).

Существует две большие группы БД.

Реляционные и не реляционные.

Реляционная ***База данных****, основанная на* ***реляционной*** *модели данных. Понятие «****реляционный****» основано на англ. relation («отношение, зависимость, связь»).*

*Использование **реляционных баз данных** было предложено доктором Коддом из компании IBM в 1970 году.*

Не реляционные соотвественно не построенно на привычных столбиках и столбцах.

Также существуют SQL и NoSQL БД.

В SQL бд, для обращения к серверу БД, используется язык SQL, в NoSQL, он соответственно не используется.

Для своего сайта я выбрал реляционную, SQl БД, а именно - MySQL.

А вот сама структура БД, я выпишу только название таблиц, т.к. описание каждого из полей таблицы не имеет смысла:

Comment(Комментарии пользователей)

Cookie_user(Хранит куки для автоматического входа в систему)

Email_confirm(Для подтверждения почты пользователя)

Event(Хранит все события на сайте)

Post(Блок содержащий все новости)

Rating(Таблица с рейтингом языков программирования)

Reset_password(Таблица для смены пароля)

User(Таблица с пользователями сайта)

Usering(Таблица в которой содержаться все пользователи с не подтверждённым email)

Дополнительно

В процессе создания сайта, я добавил несколько Python файлов:

Log.py - отвечает за логику создания, сохранения и показа логов сайта.

Buisness_logic.py - отвечает за реализацию дополнительных функций, относящихся к бизнес-логике именно моего сайта.

Трудности, проблемы и не доработки сайта

За всё время реализации этого проекта, я сталкивался с различными проблемами, но все они(А вернее, почти все) были мною устранены, но нужно и рассмотреть проблемы которые я не смог решить:
1) Автовход на сайт через куки, реализован не очень грамотно и продуманно, т.к. хранение токенов уже устаревший метод аунтификации.
2) Отправка E-mail, для подтверждения почты пользователя, реализованно с помощью сторонней библиотеки, к тому же, сама почта не доменная.
3) Сохранение и отображение логов сделано через ‘Костыль’, в начало и место до слова Traceback ставятся два греческих символа, чтобы можно было выделять текст в этих местах другим цветом. Также логи идут от самых старых(Сверху файла) к новым(Внизу файла), что также не удобно)

Интересные факты

После окончания работы над сайтом, я стал успешно продавать его движок. Сейчас его купила уже несколько человек и используют его для своих сайтов. И в этом и заключается отличие школьного проекта от настоящего. Настоящий проект нужен кому-то, он коммерчески успешен, а школьный нет. И в своём проекте я пытался создать именно настоящий, коммерчески успешный проект и у меня это получилось.

Сайт состоит из 132 файлов, из которых:
25 — html файлов
9 - Python файлов
2 — txt файла
И другие файлы(картинки, шрифты, CSS, JS)

Сайт находится по адресу https://site-hunter.ru

А также на GitHub на котором можно посмотреть его исходный код и скачать себе на компьютер: https://github.com/Ryize/it-news

(Сайт находится под лицензией Apache 2.0 и требует:

1) Упоминания автора и лицензии в своём проекте

2) Указывать изменения, внесённые в работу)

Сайт был создан за 3 недели.

Список литературы

https://tiobe.com/tiobe-index/

https://ru.wikipedia.org/wiki/Flask_(веб-фреймворк)

https://itproger.com/news/10-samih-populyarnih-saytov-napisannih-na-django

https://habr.com/ru/post/92570/

https://habr.com/ru/post/237889/