**МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**СРЕДНЯЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ШКОЛА № 3**

**С УГЛУБЛЁННЫМ ИЗУЧЕНИЕМ ОТДЕЛЬНЫХ ПРЕДМЕТОВ**

**ПРИКЛАДНОЙ ПРОЕКТ**

**«Twitter.ru. Российский аналог твиттера»**

Выполнил:

Ученик 11А класса

Пичугин Янислав Дмитриевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

Научный руководитель:

Добряков Михаил Андреевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

2024 г

**Оглавление**

[Введение 3](#_Toc162445619)

[Устройство сайта 5](#_Toc162445620)

# Введение

Первоочередная цель этого проекта создание пет-проекта в качестве портфолио. Темой было выбрано русифицирование широко известной социальной сети **Twitter**. Важнейшим пунктом в достижении данной цели является создание сайта.

Как работает веб-пространство и что вообще из себя представляет сайт. Если кратко, то сайт – набор нескольких документов, объединённых одним стилем и оформлением. Интернет-соединения работают по принципу клиент – сервер, где клиент отправляет какой-либо запрос посредством протоколов к серверу. Сервер обрабатывает этот запрос, совершает некоторые алгоритмы, заданные программой, и отправляет обработанные данные в ответ (не всегда). Существуют два протокола передачи данных: протокол связи и сетевой протокол. Протокол связи описывает правила взаимодействия между программами, а сетевой протокол является набором правил и действий, позволяющий осуществлять соединение и обмен данными. В случае работы сайтов клиент – это браузер, которым открывают странички, а сервер – компьютер, который обрабатывает наш запрос и отправляет запрашиваемую страницу. Однако, Интернет не ограничивается только запросами страниц и ответом HTML-документом, но это уже другая тема.

Для этого проекта были использованы различного рода инструменты и технологии. Основными для создания внешнего вида сайта стали: HTML (Hyper Text Markup Language «язык разметки гипертекста»), CSS (Cascading Style Sheets «каскадные таблицы стилей») и JS (JavaScript – скриптовый язык программирования). Для описания алгоритмов обработки запросов выбран язык программирования Python и фреймворк Django.

Почему именно эти технологии, а не другие? Потому что второстепенная цель проекта – обучение. Python подходит больше прочих так как является одним самых простых в освоении ЯП, а также в Интернете много обучающего материала по этой теме, как на русском, так и на английском языках. Ещё просто потому, что мне захотелось писать на Python, а не, например, PHP, который является самым популярным ЯП для веба.

Для достижения поставленной цели необходимо решить следующие задачи:

* Изучить функционал Twitter.com
* Реализовать основной функционал в этом проекте:
  + Система пользователей (регистрация, авторизация, подписка пользователей друг на друга)
  + Создание, чтение, удаление постов
  + Система рекомендаций для выдачи релевантного контента для каждого пользователя

# Устройство сайта

Для начала немного о Django. Во фреймворке Django существует удобный менеджер, который помогает управлять проектом. Он позволяет создавать приложения, предоставляет удобный инструментарий для работы с БД, имеет при себе тестовый сервер для отладки приложений. Мой Твиттер состоит из одного такого приложения.

Составлю небольшой глоссарий в рамках моего проекта на основе философии Django:

* Приложение – модуль, который выполняет свою определённую роль. Пример: приложение **customUser** занимается всем, что связано с регистрацией, авторизацией и т.д. Система пользователей.
* Вьюшка (view – представление) – файл, который отвечает за обработку входящего объекта Request. Обработчик запроса (request – запрос)

Архитектура сайта очень проста. Каждое приложение может состоять из разных компонентов, однако можно выделить практически в каждом следующие:

* вьюшки (views.py) – в них, как я уже писал раннее, обрабатываются переданные запросы.
* apps.py – служит в основном для описания данного модуля. Нужно в качестве архитектурного решения для удобства разработки.
* модели (models.py) – в них описываются, кто бы мог подумать, модели. Модель – это некоторый шаблон, по которому создаются объекты и который говорит, как эти объекты могут обмениваться данными. Каждая модель создаёт свою таблицу в базе данных.
* urls.py – в нём описываются какие вьюшки должны обработать конкретный url.

Остальные компоненты добавляются в зависимости от предназначения модуля

Все запросы передаются в одно приложение и обрабатываются там же, и сразу отдают ответ или перенаправляются в другие модули для дальнейшей обработки.

Так выглядит мой сайт изнутри.

Приложение **api** предназначено для представлений, которые получают и отдают запросы в основном в формате JSON (однако, не всегда).

Приложение **azzb** самое важное. Оно является корневым который обрабатывает и обеспечивает связь с другими приложениями.

Приложение **customUser** занимается регистрацией, авторизацией, хранит посты, которые пользователь лайкнул, на кого пользователь подписан.

Приложение **post** служит для обработки постов. Создание, редактирование, удаление, просмотр (CRUD – create, read, update, delete), а также возможность лайкнуть, прокомментировать. Имеет некоторые метрики (характеристики) по которым можно сортировать наиболее релевантные посты в выдаче в поиске или на главной.

В каждом приложении есть файл **tests**, который нужен для автоматизированных тестов. Они позволяют отлаживать работу отдельных компонентов сайта. Поскольку разработка ведётся в течение долгого периода времени, что-то может переосмысляться и меняться, либо просто в процессе разработки изменения в одном компоненте могут влиять на другие. Это может вызвать ошибки, которые часто не замечаются сразу. Именно задачу «найти ошибку вовремя» решают автоматизированные тесты.

Папочка **media** не была пропущена. Она не является модулем. В ней хранятся данные, которые загружают пользователи.

**db.sqlite3** – файл базы данных. Хранит всю информацию.

**manage.py** – файл, позволяющий администрировать проект. Нужен для оптимизации разработки.

.env – файл, в котором прописаны некоторые переменные окружения. Одна из переменных называется SECRET\_KEY, которая нужна для генерации CSRF токенов, которые обеспечивают некоторый уровень безопасности.

Рассмотрим каждое приложение поближе.

**API**.

Это приложение предоставляет api для взаимодействия с остальным приложением. Обрабатывает все запросы на /api/.

API (Application programming interface) – это контракт (интерфейс), который предоставляет программа. «Ко мне можно обращаться так и так, я обязуюсь делать то и это».

Когда создаётся, лайкается, удаляется пост, запрос отправляется именно в это приложение.

Ручки, или же эндпоинты (endpoints) – некоторый адрес, по которому можно обратиться к определённому методу API.

**AZZB**.

Как уже писалось раннее, это основное приложение. В нём прописаны глобальные настройки. Какой обработчик шаблонов использовать, в каком приложении прописаны главные пути сайта, установленные приложения, подключенные базы данных, настройки почтового обработчика и прочие.

В нём реализовано несколько вьюшек:

* index – главная сайта, выводи последние несколько постов
* follows\_updates – последние посты от пользователей, на которые подписан пользователь

**CUSTOMUSER**.

Данное приложение хранит пользователей, обрабатывает регистрацию, авторизацию пользователя, редактирование профиля, смену пароля, а также возможность подписываться на профили. Обрабатывает /edit/, /sign-up/, /login/, а также /users/<slug>/, где slug – логин пользователя (не отображаемое имя!) пути.

**POST**.

Приложение, которое обрабатывает и хранит посты. Создание, удаление, лайк. Вывод их, визуальное отображение. У каждого поста имеется счётчик лайков, дата его создания

**TEST**.

В данном случае я использую это приложения для тестирования некоторых предварительных фишек. Никак не влияет на работу всех остальных приложений.