# **--- Django ---**

**Framework Django** – заготовка (каркас) для создания сайта, где внутри разработчик сам добавляет нужным функционал на сайте. Инструмент для back end разработки, взаемодействие пользователся с сервером на котором лежит сайт

**Node.js** - позволяет запускать код Java Script вне браузера

**Model , Temlate , View ( MTV ) -** три основные части Django приложений

**Model + Template => View => Ответ Пользователю**

**URL запроса пользователя -> URL Маршрутизатор -> View1(**предстваление1**)** или **Viev2(**представление2**) в зависимости от URL пользователя -> Формирование Ответа Пользователю, который состоит из Моделей + Шаблонов**

**Если URL запрос пользователя не найден тогда выводиться ошибка ERROR 404 - страница не найдена**

**Представление(View) -** формирует ответы для польователей основываясь на ихних запросах. Можно реализовать только или в виде функции или в виде класса

**Классы Представления (Class Based Views = CBV) -** позвоялет реализовывать больше функционала в представлениях. Можно прописывать разуную риализацию представление в записимости от типа запроса пользователся GET или POST

**Модели(Model) -** класс, который создает таблицу базы данных в проекте. Обьекты Класса - ето записи в етой таблице базе данных

**Django ORM (Object Relation Mapping)** - возможность взаемодейтствовать с моделями базы данных проекта при помощи языка Python, а не SQL

**Шаблоны(Templates) -** служат для генерации html кода, который будет показываться пользователю. Генерация внешнего вида ответа пользователю

**Приложения** - составные части сайта, которые не связаны между собой. Приложения реализуються как пакеты языка python

**Миграции** – действия, которые проделываем на базой данных. Все действия над базой храняться в папке migrations. Позволяет откатиться к предидущим версиям базы данных

**Пагинация Страниц** - постраничная навигация, разделение одной сплошной страници сайта на несколько и переключение между ними кнопками снизу

**Статические Файлы** - файлы CSS , JS , которые подключаються к проекту Django

**Медиа** – картинки или видео на нашем сайте

**CRUD (Create / Read / Update / Delete)** - создание / чтение / изменение / удаление записей в базе данных

**Индексы Базы Данных** - механизмы поиска в базе данных. Замедляет удаление, изменение и добавление значений, но ускоряет чтение етих значений.

**Менеджеры Обратного Связывания Записей Базы Данных** - механизм работы с базами данных при их вызове. По умолчанию ClassBase.objects

**Список Выборов(choices) -** когда вписываем в пользователю будут показываться одно значения, а в базу данных будут записываться другие

**Перечисления (Переопредиление Значений) -** когда опеределяем значения названием. И можем обращаться к етим значениям уже по указаным названиям. Можно переопределять в классе Meta (models.py)

**Поиски(Lookups)** - возможность прописывать характеристики для запроса из базы данных. Сопоставление по pk(id) или my\_property

**Формы Django** - пользовательский класс, который создан для автоматизации создания форм в html документах

**HTML Страница** – результатом работы фрейморка django являеться страница html, которая показываеться пользователю

Процесс Запроса  **==> Запрос Пользователя -> Web Сервер -> Django -> html страница**

Процесс Ответа **==>**  **html страница -> Django - > Web Сервер -> Ответ для пользователя**

**Виртуальнео Окружение** - изолирования среда для каждого проекта для использование конкретных версий модулей та интсрументов Django чтоб после обновления етих продуктов не возникали проблемы на сайте

**Аутентификация (Authentication)** - проверка подлиности (наличия ) пользователя по введеным данным с данными, которые находяться в базе данных. Существует ли такой пользователь в нашой базе данных

**Авторизация (Authorization)** - разрешение для пользователя на доступ к закрытой части сайта. Выполняеться после аутентификации

# **Django REST**

**Django Rest Framework (DRF)** - метод реализации RESTful API через framework Django. DRF позволяет создать API для нашего приложения

**Клиент -> Запрос -> Django REST -> file.json -> Возврат Клиенту file.json**

**API Запрос -> Маршрутизатор URL -> Представления -> Серилизатор -> База Данных**

**Сериализатор -** преобразовует файл .json формата в ячейки памяти моделей Django или наоборот. Выполянет также функцию создания , чтения , изменения или удаления записей в базе данных. Нужен для легкой передачи данных через API

**Сериализация Обьекта** - преобразования обьекта(ячейки в моделе django) в строку json

**Десериализация Обьектов** - преобразования строки json обратно в обьект (ячейку в моделе django)

**APIView(Представления)** - только обрабатывают все типи запросов от клинента на сайт

**ViewSet (Наборы Представлений)** - хранит внутри себя все типы запросов get- post- put- patch- delete. Наборы представлений выполняют методы(действия) на базой данных. Можна переопределяться все типы запросов. Меньше занимает места в коде чем просто api-представления

**Mapping (Мапинг)** – связивание указаного типа запроса от пользователя с серверу с методом, которые определен в ViewSet

**Endpoint** - URL адресса с помощью которых frontend нашего сайта или другие сайты получаються данные в JSON формате

**Контролеры** - предсталвения в views.py которые отвечают за отправку данных в формате JSON

# **Celery**

**Celery** - обработчик задач на нашем веб-сайте, который выполянет задачи, которые делаеть пользователь асинхроно(паралельно)

**Таски (Tasks)** - задачи на нашем сайте, которые приходят от нашего пользователя. Загрузить картинку , скачать музыку , зделать денежный перевод. Ети задачи асинхронные

**Брокеры (Broker)** - создает очередь из принятых тасков для выполнения Воркером. Хранит ести такси внутри себя

**Воркер (Worker** или **Consumer) -** выполняет таски. Ожыдает поступления тасков от брокера

**Task -> Broker -> Worker -> Задача Выполнена**

**Внутри Celery НЕ можна работать с ORM Django**

**Flower -** позволяет наблюдать за выполеннием тасков не из консоли

# **Redis**

**Redis** **Сервер** - специальная база данных, котора хранит асинхронные операции на нашем сайте. По сути ето брокер для Celery. Данные в етой базе хранять по типу ключ - значение

**redis-cli** инструмент для работы с redis сервер