# Веб-приложение

## Общие сведения

Веб-приложение – это разновидность программного обеспечения, доступ к которому обеспечивается средствами веб-браузера. Веб-браузер – это программа, которая отправляет запросы, получает и обрабатывает ответы от имени пользователя на его устройстве. Обязательным условием функционирования веб-приложения является наличие стабильного соединения с глобальной сетью Интернет. Одно из главных отличительных особенностей такого вида приложений является кроссплатформенность и независимость от аппаратной конфигурации устройства, у которого есть доступ в сеть.

В этом разделе изложены основные идеи и концепции построения веб-приложения, начиная от описания клиент-серверной архитектуры и краткого обзора протокола HTTP, заканчивая описанием реализации обработчиков запросов фреймворка Flask. При помощи этого фреймворка реализуется веб-приложение для идентификации динамических систем.

## Описание архитектуры веб-приложения

Архитектура программного обеспечения (ПО) – это организация и структура системы, определяющая ее компоненты, их взаимодействие и принципы организации данных. Она определяет основные аспекты системы, такие как разделение ответственности, модульность, коммуникацию и распределение функций.

Глобально любое веб-приложение основано на клиент-серверной архитектуре. В такой архитектуре есть устройство, которое обслуживает запросы, и есть устройство, которые эти запросы направляет. Сервер – это устройство, на котором установлены программы, которые обслуживают запросы клиентов. Клиент – это программа пользователя, которая удовлетворяет его потребностям.

В основе вышеописанного взаимодействия лежит протокол прикладного уровня модели OSI: протокол HTTP (HyperText Transfer Protocol). HTTP был разработан как протокол обмена данными между веб-сервером и веб-браузером. Который используется для передачи между клиентом и сервером (хранителем) файлов HTML, CSS, JS, изображений, аудио, видео, введённых пользователем данных и прочего. Клиент (веб-браузер) отправляет серверу (веб-серверу) запросы и получает от него ответы. Сервер в рамках взаимодействия всегда «ожидает» инициации от клиента.

На этапе инициации клиент устанавливает соединение с сервером с помощью протокола транспортного уровня TCP. Клиент может использовать одно и то же соединение для работы с сервером или инициировать новое соединение каждый раз. Это зависит от задачи, конфигурации сети и конкретных настроек оборудования. После установки соединения клиент посылает HTTP-сообщение с телом и параметрами запроса. Сервер принимает это сообщение и на основании логики работы формирует и отправляет HTTP-сообщение ответа. Таким образом, организовывается сеанс сессии.

## Виды веб-приложений

Все веб-приложения разрабатываются с помощью одних и тех же технологий: HTML, CSS и JavaScript. Одновременно все три используются, или по выбору? Это же три технологии. И есть выбор их применения. Однако есть много вариаций организации работы приложения. Выбор способа зависит от цели приложения. Основных подхода всего два: многостраничные приложения и Single Page Applications (одностраничные приложения), но каждый из них делится на подвиды.

### Многостраничные приложения

Многостраничные приложения состоят в основном из статичных страниц. В таких страницах нет изменяемого контента. Их просто разрабатывать, но если страниц становится много (сотни или тысячи), или данные на странице меняются, то придётся генерировать их в режиме реального времени. Для этого нужно затрачивать программные и аппаратные ресурсы: подключать серверные мощности и писать дополнительный код и на каждый переход от одной страницы к другой нужно генерировать и загружать новую страницу, а это занимает время.

Рассмотрим подвиды многостраничных приложений: приложение с набором готовых страниц (файлы расширения .HTML), и приложение, с множеством страниц, которые генерируются на сервере.

*Совокупность готовых статичных страниц*, которые связаны между собой ссылками друг на друга, также имеют право называться веб-приложением, но точнее их будет назвать сайтом. Как правило, такие сайты содержат в себе данные, которые не меняются или слабо меняются с течением времени: это может быть персональный блог или страница с индивидуальными достижениями. Для организации доступа к таким страницам используются веб-сервера, которые по запросу пользователя вернут ему нужный готовый ресурс.

*В случае приложения с динамической генерацией HTML* файлов есть возможность размещать на страницах нестатичные данные, то есть такие, которые меняются с течением времени в зависимости от действий пользователя или в зависимости от внешних изменений. Красная строка. Новая мысль

В данной работе используется именно такой подход с помощью шаблонизатора Jinja2 – это шаблонный движок на языке Python, который используется для генерации HTML-страниц и других текстовых форматов. Предложение опять нескладно. См.падежи! И… ранее об Jinja2 не было ни слова. В этом же абзаце – итог написанного выше. Он предоставляет инструменты для управления данными и создания динамических веб-страниц. Jinja2 может быть использован вместе с различными фреймворками веб-приложений, такими как Flask, Django и Pyramid. Здесь такое же замечание.

### Одностраничные приложения

Одностраничные приложения (SPA –раз пишете аббревиатуру, дайте ее английское обозначение) дают возможность разрабатывать клиентские приложения со сложной логикой с помощью JavaScript. В этом подходе отрисовкой содержимого на странице управляет JavaScript. При помощи этого языка программирования можно написать функции, которые будут посылать запросы веб-серверу на получение данных, затем они будут размещены на странице в соответствующих местах. Переходы между экранами будут мгновенными, и пользователь сразу увидит результат своих действий. Однако такой подход создаёт новые проблемы, а именно: большая нагрузка на браузер, вследствие этого падение производительности.

Разработка таких приложений часто сложнее, поэтому в данной выпускной работе вид веб-приложения – многостраничный, но с размещением на них нестатичных элементов, например, интерактивный график переходного процесса. Этот выбор себя оправдывает, так как само приложение не нагружено большим количеством страниц. Здесь вывод хорош

## Архитектура и паттерны проектирования веб-приложений

**Архитектура приложения** — это набор решений о том, как модули приложения будут связаны друг с другом и с внешними компонентами. **Паттерны проектирования** – шаблонное решение частной архитектурной проблемы. Важно спроектировать веб-приложения согласно определенном паттерну, исходя из архитектурного решения. Рассмотрим паттерн проектирования MVC, который используется в веб-приложении для идентификации динамических систем. Здесь и ранее – опять не вижу ссылок на литературу, откуда взяты определения. Вы хотя бы расставьте скобки [ ]. А кокретную литературу проставите потом

### Паттерн проектирования MVC

MVC (Model-View-Controller) — структура приложения, в которой за данные, их обработку и их вывод отвечают три разных сущности:

* Модель (model) отвечает за данные и их структуру, cодержит в себе схему данных приложения;
* Представление (view) отвечает за отображение данных, показывает данные в понятном для пользователя виде;
* Контроллер (controller) или обработчик – отвечает за обработку данных.

Таким образом разделяется ответственность между разными программными модулями. Важно упомянуть о потоке данных между этими сущностями (см. рис. 1):

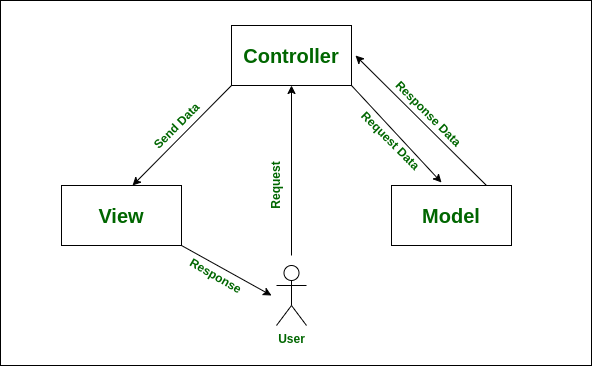
* от пользователя передаются представлению;
* от представления — контроллеру;
* через контроллер обновляется модель;
* модель уведомляет представление о том, что что-то изменилось.

Рисунок – Паттерн проектирования MVC

### Архитектурный стиль RESTful API

Это архитектурный стиль, который используется для создания веб-сервисов. Возможно, так и должно быть. Но стиль и инструмент – это разное. Инструментом пользуются для создания. А стиль – это на что должен быть похож результат. Как минимум. Он опирается на принципы REST (Representational State Transfer) и включает в себя использование HTTP-методов (GET, POST, PUT, DELETE) для управления ресурсами.

*Информация в этом подразделе будет дописываться.*

## Фреймворки для веб-разработки

Фреймворк – набор готовых программных пакетов, программных модулей для облегчения труда программистов и экономии времени разработки программного обеспечения. Фреймворк конкретизирует архитектурное решение и позволяет реализовать выбранный паттерн проектирования. Рассмотрим один из фреймворков.

*Информация в этом подразделе будет дописываться.*

### Flask

Flask – это минималистичный фреймворк для создания веб-приложений на языке Python. Он предоставляет широкий спектр возможностей и инструментов для быстрой и качественной разработки отзывчивых приложений.

Вот некоторые ключевые особенности данного фреймворка: Здесь и выше: наверное, отступы можно уменьшить. Выполню в качестве примера:

1. Легкость использования: Flask имеет простую и интуитивно понятную структуру, которая делает его легким в освоении и использовании. Он предоставляет только базовый функционал, позволяя разработчикам гибко выбирать сторонние инструменты и библиотеки для своих проектов. Или какой-то промежуточный вариант. Но… тут решение принимаете Вы
2. Маршрутизация URL: Flask предлагает простой способ определения URL-маршрутов и связывания их с соответствующими функциями-обработчиками. Это позволяет определить, как приложение должно отвечать на запросы по определенным URL-адресам.
3. Встроенный сервер разработки: Flask поставляется с встроенным сервером разработки, который упрощает запуск и тестирование веб-приложений без необходимости настройки отдельного веб-сервера.
4. Расширяемость: Flask имеет модульную архитектуру, позволяющую разработчикам добавлять функциональные возможности с помощью расширений (extensions). Существуют множество расширений для Flask, которые обеспечивают поддержку баз данных, авторизацию, формы, обработку файлов, асинхронные запросы и многое другое.
5. Шаблонизация: Flask предоставляет инструменты для использования шаблонов, которые упрощают создание динамических веб-страниц. Шаблонизация позволяет разделить логику и представление веб-приложения, что делает код более удобочитаемым и обеспечивает повторное использование кода. Такой движок шаблонизации называется Jinja2.
6. Поддержка HTTP-методов: Flask обрабатывает различные HTTP-методы, такие как GET, POST, PUT, DELETE и другие, что позволяет легко определять обработчики для разных типов запросов.
7. Отладочные возможности: Flask предлагает отладочные инструменты, которые помогают разработчикам в процессе отладки приложений. Это включает в себя режим разработки, отображение отладочной информации при ошибке и возможность интерактивной отладки.
8. Flask является очень популярным фреймворком для разработки веб-приложений на Python благодаря своей простоте и гибкости. Он хорошо подходит для небольших и средних проектов, где требуется быстрый старт и минимальная настройка.

*Информация в этом подразделе будет дописываться*