Существует множество архитектурных решений для создания сайтов. Рассмотрим наиболее распространенные из них и их преимущества и недостатки:

1. **Статические сайты** - это набор HTML-файлов, которые содержат контент и стили, без динамических элементов. Они не имеют динамической функциональности и обычно используются для простых сайтов-визиток или статических страниц.

Преимущества:

* Быстродействие: статические сайты обычно быстрее загружаются, чем динамические, т.к. не требуется обработки запросов к базе данных.
* Безопасность: статические сайты менее уязвимы к хакерским атакам, т.к. не имеют серверной логики и не могут быть атакованы через SQL-инъекции или другие уязвимости базы данных.

Недостатки:

* Ограничения в функциональности: статические сайты не поддерживают динамические элементы, такие как формы обратной связи или интерактивные элементы пользовательского интерфейса.
* Сложность редактирования: каждый раз, когда необходимо внести изменения на сайт, необходимо вручную редактировать HTML-код.

1. **CMS** (системы управления контентом) CMS - это программное обеспечение для управления содержимым веб-сайта, которое позволяет пользователям создавать, изменять и публиковать содержимое веб-сайта без необходимости знания HTML, CSS и других языков программирования.

Преимущества:

* Простота в использовании: CMS позволяют создавать сайты без необходимости знания программирования.
* Гибкость: CMS позволяют настраивать сайт и добавлять новые функции с помощью плагинов и расширений.

Недостатки:

* Безопасность: CMS имеют уязвимости, которые могут быть использованы злоумышленниками для взлома сайта.
* Сложность настройки: необходимо выбрать подходящую CMS, установить ее на сервере и настроить конфигурацию для своих нужд.

1. **SPA** (одностраничные приложения) SPA - это приложения, которые загружаются на одной странице и динамически обновляют контент без перезагрузки страницы. Они создаются с помощью фреймворков, таких как React или Angular.

Преимущества:

* Пользовательский опыт: SPA обеспечивают более быстрый и плавный пользовательский опыт, т.к. не требуют перезагрузки страницы при каждом обновлении контента.
* Масштабируемость: SPA обычно могут обрабатывать большие объемы данных и поддерживать большое количество пользователей.

Недостатки:

* SPA могут иметь проблемы с оптимизацией для поисковых систем, т.к. поисковые роботы не всегда могут обработать динамический контент.
* Нагрузка на браузер: SPA требуют больше ресурсов браузера, что может привести к более медленной работе на старых компьютерах или мобильных устройствах.

1. **SSR** (серверный рендеринг) SSR - это подход, при котором сервер отображает HTML-код страницы на стороне сервера перед отправкой его браузеру. Он позволяет создавать динамические приложения с использованием фреймворков, таких как React или Vue.

Преимущества:

* SSR обеспечивает лучшую оптимизацию для поисковых систем, т.к. все контент отображается на стороне сервера.
* Быстродействие: SSR может ускорить загрузку страницы, т.к. часть контента уже отображается на стороне сервера.

Недостатки:

* Сложность настройки: SSR требует настройки сервера и изменения кода приложения, что может быть сложно для начинающих разработчиков.
* Большая нагрузка на сервер: SSR требует больше вычислительных ресурсов на стороне сервера, что может привести к дополнительной нагрузке на сервер и повышенным затратам на хостинг.

1. **JAMstack** (JavaScript, APIs, Markup) - это подход, при котором веб-страницы создаются с использованием статических файлов HTML, CSS и JavaScript, обращаясь к API для получения динамических данных и функциональности.

Преимущества:

* Быстродействие: JAMstack позволяет создавать очень быстрые и быстрозагружаемые сайты, т.к. веб-страницы уже собраны на стороне клиента и не требуют серверной обработки при каждом запросе.
* Безопасность: JAMstack минимизирует возможность взлома сайта, т.к. не использует динамическую серверную логику.
* Масштабируемость: JAMstack может легко масштабироваться и обрабатывать большое количество пользователей.

Недостатки:

* Ограниченная динамичность: JAMstack может иметь ограниченную функциональность, т.к. динамические данные обычно получаются через API, что может затруднять интеграцию сложных приложений и функций.
  + Сложность разработки: JAMstack требует использования инструментов сборки и развертывания сайтов, что может потребовать дополнительных знаний и умений от разработчиков.

**MTV-архитектура** (Model-Template-View) - это шаблон проектирования, используемый во многих веб-фреймворках, таких как **Django**. MTV-архитектура разделяет приложение на три части:

* Модель (Model) - это представление данных, используемых в приложении. Он определяет структуру данных, связи между ними и методы доступа к этим данным.
* Шаблон (Template) - это представление того, как данные должны быть отображены на веб-странице. Он определяет, как данные должны быть отформатированы и представлены пользователю.
* Представление (View) - это код, который управляет взаимодействием между моделью и шаблоном. Он определяет, какие данные из модели должны быть переданы в шаблон и как пользователь может взаимодействовать с этими данными.

Преимущества:

* Разделение обязанностей: MTV-архитектура разделяет приложение на три отдельные части, что упрощает разработку и обеспечивает более четкое разделение обязанностей между разработчиками.
* Гибкость: MTV-архитектура позволяет легко модифицировать и расширять приложение, поскольку каждая часть приложения является отдельным компонентом, который может быть изменен независимо от других компонентов.
* Удобство тестирования: Модель, шаблон и представление могут быть тестированы независимо друг от друга, что упрощает процесс тестирования и обеспечивает более высокое качество кода.

Недостатки:

* Сложность: MTV-архитектура может быть сложной для понимания для начинающих разработчиков, поскольку она включает в себя несколько компонентов, каждый из которых имеет свои собственные функции и особенности.
* Сильная связность: MTV-архитектура может привести к сильной связности между компонентами приложения, что может затруднить изменение одного компонента без изменения других компонентов.
* Накладные расходы: MTV-архитектура может быть менее эффективной, чем другие архитектурные подходы, такие как MVC, в тех случаях, когда приложение требует множество запросов к базе данных.

**Проанализировать: архитектуры с точки зрения бизнес-драйверов, стейкхолдеров, требований, атрибутов качества - Статические сайты, CMS, SPA, SSR, JAMstack, MTV**

1. **Статические сайты**

Бизнес-драйверы:

* Снижение затрат на хостинг, так как статические сайты требуют меньше ресурсов сервера, чем динамические.
* Быстрое развертывание, так как статические сайты не требуют установки баз данных и настройки сервера.

Стейкхолдеры:

* Владельцы бизнеса, которые ищут простое и дешевое решение для веб-присутствия.
* Разработчики, которые предпочитают использовать статические сайты для быстрого развертывания.

Требования:

* Дизайн-шаблоны, чтобы создать статические страницы.
* Знания HTML, CSS и JavaScript для настройки и развертывания.

Атрибуты качества:

* Простота использования.
* Быстрое развертывание и экономия затрат.
* Высокая производительность, так как статические сайты не требуют обработки данных на стороне сервера.

1. **CMS**

Бизнес-драйверы:

* Управление содержимым веб-сайта без необходимости знания языков программирования.
* Ускорение времени публикации контента.
* Возможность реализовать сложные функции сайта, такие как блоги, форумы, магазины и т. д.

Стейкхолдеры:

* Владельцы бизнеса, которые ищут простое и эффективное решение для управления своим веб-сайтом.
* Редакторы контента, которые хотят иметь быстрый и легкий доступ к управлению содержимым веб-сайта.

Требования:

* Понимание основ CMS, чтобы настроить их для конкретных потребностей.
* Знание языков программирования, если требуется настройка дополнительных функций или создание пользовательских шаблонов.

Атрибуты качества:

* Легкость использования для редакторов контента.
* Возможность быстрого добавления и изменения содержимого веб-сайта.
* Гибкость и расширяемость для настройки дополнительных функций и пользовательских шаблонов.

1. **SPA** (Single Page Application) SPA - это веб-приложение, которое загружает только одну страницу в браузере и динамически обновляет содержимое на этой странице без перезагрузки.

Бизнес-драйверы:

* Повышение производительности и улучшение пользовательского опыта, поскольку SPA не перезагружает всю страницу при обновлении содержимого.
* Возможность реализовать более сложные функции веб-приложения, такие как интерактивные графики, диаграммы и т. д.

Стейкхолдеры:

* Владельцы бизнеса, которые хотят улучшить производительность и пользовательский опыт своего веб-приложения.
* Разработчики, которые ищут более гибкое и масштабируемое решение для веб-разработки.

Требования:

* Опыт работы с JavaScript, HTML и CSS для разработки SPA.
* Знание фреймворков и библиотек, таких как Angular, React или Vue.js.

Атрибуты качества:

* Быстрая и более плавная загрузка веб-приложения.
* Улучшенный пользовательский опыт благодаря более гладким и быстрым переходам между страницами.
* Возможность реализации более сложных функций веб-приложения.

1. **SSR**

Бизнес-драйверы:

* Улучшение производительности, так как страницы генерируются на сервере, а не на стороне клиента.
* Улучшение оптимизации поисковых систем, поскольку поисковые системы могут лучше индексировать сайты, использующие SSR.

Стейкхолдеры:

* Владельцы бизнеса, которые хотят улучшить производительность и оптимизацию поисковых систем своего веб-приложения.
* Разработчики, которые ищут более производительное решение для веб-разработки.

Требования:

* Знание языков программирования, таких как JavaScript, для разработки серверной части приложения.
* Опыт работы с фреймворками и библиотеками, такими как Next.js или Nuxt.js.

Атрибуты качества:

* Улучшенная производительность и быстрое отображение страниц на клиентской стороне.
* Лучшая оптимизация поисковых систем благодаря генерации HTML-кода на сервере.

1. **JAMstack**

Бизнес-драйверы:

* Быстрая загрузка страниц благодаря статическому контенту и использованию CDN.
* Улучшение безопасности благодаря предварительно сгенерированным страницам, которые могут быть защищены статическими правилами безопасности.

Стейкхолдеры:

* Владельцы бизнеса, которые хотят создать быстрый и безопасный веб-сайт.
* Разработчики, которые ищут гибкое и масштабируемое решение для веб-разработки.

Требования:

* Опыт работы с различными технологиями, такими как HTML, CSS, JavaScript, API-сервисы.
* Знание статических сайт-генераторов, таких как Gatsby, Jekyll или Hugo.

Атрибуты качества:

* Высокая производительность благодаря использованию статического контента и CDN.
* Более безопасный веб-сайт благодаря статическим правилам безопасности для предварительно сгенерированных страниц.

1. **MTV**

Бизнес-драйверы:

* Улучшенная масштабируемость и гибкость разработки благодаря разделению бизнес-логики, шаблонов и отображения.
* Ускорение разработки благодаря использованию фреймворка Django.

Стейкхолдеры:

* Владельцы бизнеса, которые хотят ускорить и улучшить разработку своих веб-приложений.
* Разработчики, которые ищут готовое и надежное решение для веб-разработки.

Требования:

* Знание языка Python и фреймворка Django.
* Опыт работы с базами данных и ORM.

Атрибуты качества:

* Улучшенная масштабируемость и гибкость разработки благодаря разделению бизнес-логики, шаблонов и отображения.
* Ускорение разработки благодаря использованию готового фреймворка и встроенным инструментам Django.

В целом, каждая из этих архитектур имеет свои сильные и слабые стороны, а также применение в зависимости от конкретных требований бизнеса и стейкхолдеров. Важно выбирать правильную архитектуру для каждого проекта, учитывая требования к производительности, безопасности, масштабируемости и гибкости.

Наиболее удобным для нас оказался именно Django. В основном из-за поддерживаемого языка программирования и поддержки администрирования баз данных и самого сайта.