# Основные Архитектуры веб сервисов

## Статические сайты

**Статический сайт** — это веб-сайт, состоящий из предварительно собранных HTML, CSS, JavaScript и других файлов, которые сервер отдает клиенту без их динамической генерации на стороне сервера для каждого запроса.

Статический сайт может делать запросы на бэкенд и отображать ответы, однако это не изменяет структуру собранного html. Появляются лишь заранее определенные структурные элементы (таблицы или числа), а их значения могут изменяться бэкендом.

### Классический статический сайт

1. Статический сайт- это сайт который состоит и файлов *.html* заранее забилденных и размещенных на сервере. При запросе сервер просто отдаёт заранее приготовленные файлы пользователю. Каждый html отвечает за одну страницу.
2. механизм работы статического сайта:

* На сервере запускают базовый простой легковесный HTTP-сервер (например python http.server). В нём реализована логика какие файлы возвращать в ответ на какие запросы.

Но можно и реализовать самостоятельно через fastAPI тоже самое.

* Перемещают html файлы в директории в соответствии с логикой http сервера.

1. Создание и запуск.

* Создание папки содержащей проект
* помещение html документов и скриптов для них
* из директории этой папки запустить http сервер “python -m http.server 8000”

### SPA

#### Описание

1. Single Page Application (SPA) - тоже статический сайт, но который состоит только из одного html и множества JavaScript скриптов, которые изменяют html. Чтобы не загружать несколько html можно сделать скрипты, чтобы создать анимации или иллюзию множества страниц.

|  |  |  |
| --- | --- | --- |
|  | **Статический сайт** | **SPA (Single Page Application)** |
| **Количество HTML-страниц** | Много (index.html, about.html, contact.html, …) | Одна (index.html) |
| **Переходы между страницами** | Браузер перезагружает страницу | Всё происходит **без перезагрузки** (через JS) |
| **Навигация** | Ссылки ведут на отдельные файлы | Ссылки просто **меняют состояние React/Vue** |
| **Контент** | Статичный, заранее написанный | Генерируется динамически через JS |
| **Пример** | Сайт-визитка, документация | Gmail, Trello, Google Docs |

1. Только один html.

Главной особенностью SPA является, что у него только один html и множество js скриптов, работающих с ним. Это позволяет динамически изменять содержимое страницы без её перезагрузки, обеспечивая плавный и быстрый пользовательский опыт. Поэтому для высокой скорости работы сайта оборачивают несколько страниц в одну (один html).

1. Проблема с запросами на ресурсы SPA.

Именно из-за специфики работы SPA — когда адресная строка меняется (для удобства пользователей), но фактически используется один index.html — HTTP-сервер нужно настраивать особым образом.

Если пользователю скинули ссылку на конкретную часть сайта ( /about ), то браузер, переходя по ссылке SPA сайта с путём вроде /about или /profile, отправляет запрос к соответствующему ресурсу, которого физически не существует в папке с сайтом (там только index.html и статические файлы).

Чтобы всё работало корректно, необходимо:

* настроить клиентский JavaScript (роутер), чтобы он **обрабатывал маршруты на стороне клиента** и отрисовывал нужный фрагмент интерфейса;
* настроить сервер так, чтобы **на любой несуществующий маршрут он возвращал index.html**, позволяя SPA-приложению загрузиться и самостоятельно разобраться, что нужно показать.

1. Правильный http сервер для SPA.

Так как SPA билдится в обычный html файл, то по итогу это обычный *статический сайт* и для него также нужен сервер прослушивающий запросы и возвращающий правильные файлы, но

* Обычные статические серверы вроде python http.server не подойдут из-за особой механики работы SPA (нужно чтобы при любом запросе возвращался index.html).
* И используется вместо этого npm server.

#### Запуск

1. Механизм работы SPA сайта:

* Во время разработки происходят множественные модификации react файлов, которые в конечном итоге должны скомпилироваться в один html.

Для тестирования ещё не скомпилированных в html файловесть встроенный HTTP сервер, который умеет работать с такими сырыми файлами: *npm start*.

* Когда файлы готовы реакт проект билдится в одну папку dist, где находится один *html* и скрипты для него, которые срабатывают во время анимаций, изменения вида элементов, появления новых страниц и другого рода интерактивностей. Это уже готовая папка к работе и переносится на сервер.
* На сервере запускается особый, подходящий для SPA, http сервер слушающий запросы.

Обычно это npm serve.

Можно и fastAPI настроить и что угодно, но легче npm serve.

1. Создание и запуск:

* Создание react проекта (npm create vite@latest – –template react)
* билд проекта в папку dist:

npm run build

* Запуск сервера (serve -s) в папке dist

#### Доп

1. Роутинг.

Обеспечение корректной навигации в SPA, когда приложение использует разные адресные строки, но при этом работает только с одним HTML-файлом (*index.html*).

Роутинг - механизм навигации по SPA сайту: при нажатии на кнопку перехода на другую “страницу” сайта (/about) SPA не делает никаких запросов, а

* рендерит определенный фрагмент index.html.
* изменяет адресную строку.

С роутингом в SPA js скрипты изменяют в браузере адресную строку и отображаемый контент, но по факту являясь только одной страницей.

Это нужно, чтобы пользователь мог ориентироваться в приложении, делиться ссылками на конкретные "страницы" и браузер корректно отображал историю навигации (например, кнопки "назад" и "вперед").

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Прямо в самом приложении реакт:

// App.jsx

import { BrowserRouter, Routes, Route } from 'react-router-dom';

import Home from './pages/Home';

import About from './pages/About';

import NotFound from './pages/NotFound';

function App() {

return (

<BrowserRouter>

<Routes>

<Route path="/" element={<Home />} />

<Route path="/about" element={<About />} />

<Route path="\*" element={<NotFound />} />

</Routes>

</BrowserRouter>

);

}

export default App;

В React SPA нужно использовать <Link> из react-router-dom, чтобы избежать перезагрузки:

jsx

Копировать

Редактировать

import { Link } from 'react-router-dom';

<Link to="/about">О нас</Link>

Нельзя использовать обычный <a href="/about"> — он приведёт к полной перезагрузке страницы.

#### \_\_\_\_\_\_\_\_\_

#### Стек

1. serve

<https://www.npmjs.com/package/serve/v/11.3.0>

npm i serve@11.3.0

npm install -g serve // установить глобально

npm утилита, которая запускает сервер и прослушивает порт.

Можно запустить в разных режимах в зависимости от того какой сайт:

* для статических файлов можно запустить как

serve public/

* Для SPA:

serve -s build/

### \_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Реализация

1. Обычно финальный html доставляют клиенту при получения запроса в рут сервера «/».

Это значит, что пользователь нажал на ссылку, чтоы открыть наш сайт

1. Этот механизм можно реализовать через

- API, прослушивающий запросы (вручную написать чтобы API возвращал html

при запросе рута).

- Специализированные сервисы вроде nginx.

## Динамические сайты

Динамический сайт — это веб-сайт, содержимое которого генерируется на стороне сервера в момент запроса, с использованием серверных языков программирования, баз данных и шаблонов.

Ключевые характеристики:

Серверная генерация — HTML генерируется динамически для каждого запроса

Работа с базами данных — данные извлекаются и обрабатываются на сервере

Пользовательские сессии — поддержка состояния пользователя (логины, корзины покупок)

Интерактивность на сервере — обработка форм, бизнес-логика на бэкенде

## Гибридные архитектуры сайтов

### SSR

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Хостинги

1. Vercel

<https://vercel.com/home>

<https://test-app-six-rouge.vercel.app/>

1. Render

<https://render.com/>