Бюджетное образовательное учреждение г. Омска «Лицей №149»

Создание веб-приложения на примере приватного хранилища фотографий

(практико-ориентированный)

Выполнил : ученик 10^4 класса
Прокофьев Максим
Руководитель: Ловыгина Надежда Васильевна, кандидат технических наук, доцент

Омск, 2022

ОГЛАВЛЕНИЕ

Введение

1. О веб-приложениях

2. Создание своего веб-приложения

Заключение

Список литературы и Интернет-ресурсов

Приложение

ВВЕДЕНИЕ

Актуальность темы:

В последнее время у класса, в котором я учусь, появляется все больше совместных фотографий. Их становится настолько много, что нужно целое хранилище, чтобы вместить их все.

Я решил создать сайт-хранилище для всех этих файлов. На нем мои одноклассники смогут зарегистрироваться и просматривать фотографии, скачивать себе на устройство или загружать на сайт, чтобы не потерять их. Для похожих целей, несомненно, существуют облачные хранилища файлов, но, в отличие от них, хранение файлов на собственном сайте является бесплатным, а также при создании своего сайта я получу больше опыта в этой сфере, что является для меня очень важным, так как я хочу и дальше развиваться в сфере создания сайтов.

Цель:

Создание веб-приложения, позволяющего загружать фото- и видеоматериалы, хранить их и просматривать в любое время.

Для достижения своей цели я поставил следующие **задачи**:

1. Собрать и структурировать информацию о веб-приложениях и их создании
2. Определить требования для моего сайта
3. Составить структуру приложения
4. Разработать макет дизайна
5. Реализовать серверную и пользовательскую части веб-приложения
6. Запустить сайт в сети «Интернет»

1. О ВЕБ-ПРИЛОЖЕНИЯХ

1.1 Понятие веб-приложения

Веб-приложение — клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами. Веб-приложения стали широко использоваться в конце 1990-х — начале 2000-х годов.

Существенное преимущество построения веб-приложений для поддержки стандартных функций браузера заключается в том, что функции должны выполняться независимо от операционной системы данного клиента. Вместо того, чтобы писать различные версии для различных операционных систем, приложение создаётся один раз для произвольно выбранной платформы и на ней разворачивается.

1.2 Структура веб-приложений

Веб-приложения можно разделить на несколько типов, в зависимости от разных сочетаний его основных составляющих:

Backend (бэкенд или серверная часть приложения) — работает на удаленном сервере и обрабатывает все запросы пользователя (браузера). При каждом переходе по ссылке браузер отправляет запрос на сервер. Сервер в свою очередь обрабатывает полученный запрос и формирует ответ, который отправляется на клиентскую часть приложения и обрабатывается непосредственно в ней. Бэкенд может быть написан на разных языках программирования: PHP, Python, Ruby, C# и других.

Frontend (фронтенд или клиентская часть приложения) — выполняется в браузере пользователя. Фронтенд это графический интерфейс, то, что вы видите на странице. Графический интерфейс отображается в браузере. Пользователь взаимодействует с веб-приложением именно через браузер, кликая по ссылкам и кнопкам. Эта часть написана на языке программирования Javascript. Приложение может состоять только из клиентской части, если не требуется хранить данные пользователя дольше одной сессии. Это могут быть, например, фоторедакторы или простые игры.

В некоторых случаях веб-приложение может быть создано без использования веб сервера

и серверной части в целом, например, когда вся информация обрабатывается прямо в браузере у пользователя. Также, если у приложения присутствует серверная часть, и оно должно обрабатывать какие-либо данные, то важной частью для его создания является база данных:

База данных (БД, или система управления базами данных, СУБД) — программное обеспечение на сервере, занимающееся хранением данных и их выдачей в нужный момент. База данных располагается на сервере, это может быть как сервер, на котором работает вся серверная часть приложения, так и на удаленный от него сервер. Серверная часть веб-приложения обращается к базе данных, извлекая данные, которые необходимы для формирования страницы, запрошенной пользователем.

1.3 Устройство серверной части веб-приложения

1.3.1 Веб-сервер

Серверная часть веб-приложения может быть написана на различных языках программирования, например Ruby, PHP, Java, Python, Javascript и других. Javascript — это язык программирования, изначально предназначенный для написания графического интерфейса на клиентской части, но на нем можно написать и серверную часть, благодаря существованию Node.js — платформы, превращающей Javascript в язык общего назначения посредством трансляции его в машинный код.

Все веб-приложения должны запускаться на веб-сервере. Веб-сервер — это сервер, который принимает HTTP (HyperText Transfer Protocol — «Протокол передачи гипертекста») запросы, обрабатывает их и выдает клиенту HTTP ответ, содержащий, какую-либо информацию. Самыми популярными веб-серверами на данный момент являются Apache, Nginx, Cloudflare. Node.js также может являться веб-сервером благодаря фреймворку ExpressJS, созданного специально для создания веб-приложений на платформе Node.js.

1.3.2 База данных

База данных — это совокупность данных, которые хранятся в соответствии со схемой данных. Они используются для постоянного хранения каких-либо данных на сервере. Базы данных классифицируются по модели хранения данных. Наиболее частыми видами являются иерархические и реляционные базы данных.

Иерархическая база данных — база данных, которая использует модель данных с

древовидной структурой, состоящей из данных различных уровней. Например, иерархической базой данных является практически любая файловая система, которая состоит из корневого каталога и имеет иерархию в виде подкаталогов и файлов.

Реляционная база данных — база данных, которая использует модель данных, в которой различные объекты могут быть взаимосвязаны друг с другом. Это достаточно понятный, табличный способ представления данных, в котором каждая строка имеет свой уникальный идентификатор, благодаря которому можно устанавливать взаимосвязь между данными.

1.4 Устройство клиентской части веб-приложения

Клиентская часть веб-приложения — это скрипты, написанные на языке программирования Javascript. Эти скрипты выполняются в браузере пользователя, фактически от них зависит то, что сможет увидеть и сделать пользователь внутри приложения.

В классическом варианте создания сайтов клиентская часть используется 3 составляющие:

1. HTML (HyperText Markup Language — «язык гипертекстовой разметки») — страницы, содержащие контент, показывающийся пользователю
2. CSS (Cascading Style Sheets — «каскадные таблицы стилей») стили , которые описывают внешний вид веб-страницы
3. Javascript скрипты — программный код, который позволяет управлять поведением веб-страницы и ее содержанием.

Однако, концепция веб-приложения предполагает динамичное изменение контента, что становится возможно благодаря DOM (Document Object Model — «объектная модель документа»). DOM — программный интерфейс, который позволяет программам и скриптам получать доступ к содержимому HTML документов и изменять их содержимое, структуру и оформление во время выполнения программы. Для упрощения создания веб-приложения с использованием DOM были созданы Javascript библиотеки (*Фреймворки*), содержащие наборы классов и функций, обеспечивающих динамическое изменение отображения контента на веб-странице. Наиболее популярными фреймворками для создания веб-приложений на данный момент являются Angular, React, Vue.js. Все они различаются между собой, но глобально предоставляют примерно одинаковые возможности.

1.5 Общение сервера с клиентом

Для полноценной работы веб-приложения клиент должен обмениваться с сервером информацией, например при перезагрузке страницы или нажатии на кнопку. Общение клиента и сервера происходит по протоколу HTTP, основой которого является отправка клиентом запросов на сервер и получение различных ответов.

Современные веб-приложения в основном используют протокол HTTPS (HyperText Transfer Protocol Secure), который расширяет возможности протокола HTTP и является более защищенным благодаря поддержке шифрования. Использование зашифрованного канала данных считается хорошим тоном веб-разработки, независимо от важности передаваемых данных.

HTTP запросы бывают разных типов, но на практике с основном используется только GET (получить) и POST (отправить).

GET-запрос — метод, предназначенный для чтения данных с сервера. Например, при загрузке страницы клиент отправляет на сервер запрос на получение HTML документа, расположенного по указанному адресу

POST-запрос — метод, с помощью которого данные отправляются с клиента на сервер. Чаще всего с его помощью отправляются данные различных форм.

Все HTTP запросы имеют определенную структуру и состоят из двух частей — заголовка и тела запроса. В заголовке запроса описывается конечный URL-адрес на который придет запрос, его тип и различная техническая информация. В теле запроса могут находиться различные параметры, данные, если это POST запрос, или тело запроса может быть пустым, если это GET запрос.

Когда сервер получает запрос, он должен его обработать каким-либо образом, а затем отправить ответ, в котором находится ответ с какими-либо данными и код ответа. Наиболее часто встречающимися кодами являются 200 — если все хорошо, 404 — если страница не найдена и 500 — если сервер не смог обработать запрос и произошла ошибка.

1.6 Вывод по первой главе

Теперь, когда у нас есть знания о том, что такое веб-приложение, какова структура серверной и клиентской частей, что такое веб-сервер и база данных, как сервер взаимодействует с клиентом, можно определить задачи для своего веб-приложения и начать его разработку.