для решения проблемы уникальной идентификации как внутри отдельно взятой интегральной схемы, так и между интегральными схемами.

Список использованных источников:
1. Иванюк, А. А. Проектирование встраиваемых цифровых устройств и систем: монография / А. А. Иванюк. – Минск: Бестпринт, 2012. – 337 с.
2. Tuyls, P. Security with Noisy Data / P. Tuyls, B. Skoric, T. Kevenaar. – London: Springer, 2007. – 344 p.
3. Клыбик В. П., Иванюк А. А. Применение физически неклонируемой функции типа арбитр для решения задачи идентификации цифровых устройств. / В. П. Клыбик, А. А. Иванюк – Информатика, 2015. – в печати.

ЕДИНЫЙ ЦЕНТР ОБРАБОТКИ УДАЛЁННЫХ КОМАНД

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Скакун А. С.

Бахтизин В. В. – кандидат. техн. наук, доцент, профессор

В связи с распространением «облачных» технологий и сервисов возникает частая необходимость разработки методов взаимодействия между клиентом и сервером, что вынуждает разработчиков тратить дополнительное время при разработке ПО. Наличие какого-либо универсального решения позволило бы значительно упростить и сократить процесс разработки.

На сегодняшний день огромное количество программных продуктов строится по системе клиент-сервер. Такое решение предполагает наличие единого центра (сервера), который берёт на себя обработку поступающих команд в виде запросов и отправку ответа с результатом выполнения, и клиентов, задачи которых заключаются в формировании команд для сервера и наглядном представлении пользователю ответов с сервера.

Обычно при разработке сервера программисты создают интерфейс программирования приложения (API – Application Programming Interface), с помощью которого клиент может посылать строго определённые команды. Обычно API доступен только по протоколу HTTP или HTTPS. Это накладывает определённые требования к клиенту, ему необходимо поддерживать данный протокол. Для сложных систем это выливается в дополнительные финансовые затраты на приобретение дорогостоящего оборудования или в дополнительное время разработки программного продукта.

Решением данной проблемы может стать организация единого центра обработки удалённых команд, принимающего команды по различным протоколам и в различных форматах и приводящего их к единому формату, с которым и будут работать программисты. Разработчикам нужно будет выполнить лишь разовую настройку этого центра. Исходя из требований к разрабатываемой системе и из технических возможностей клиентов, можно определить список необходимых сетевых протоколов, которые требуется поддерживать, а также определить форматы запросов и ответов для каждого из них. Таким образом, единый центр обработки удалённых команд возьмёт на себя роль переводчика, который интерпретирует поступающие команды в понятный для сервера формат, и в обратном направлении переводит ответы сервера в понятный для клиента формат.

Главным достоинством данного решения является сокращение временных затрат при разработке серверного ПО. При проектировании многокомпонентных систем такое решение позволяет использовать устройства, работающие через разные сетевые протоколы, что может значительно сократить финансовые расходы. Использование низкоуровневых сетевых протоколов позволит увеличить их время автономной работы от батарей, т.к. существенно снизится объём передаваемых данных.

АРХИТЕКТУРА ОДНОСТРАНИЧНОГО ПРИЛОЖЕНИЯ ПО ОБРАБОТКЕ ДАННЫХ СЕРВИСА МИКРОБЛОГИНГА TWITTER

Белорусский государственный университет информатики и радиоэлектроники

г. Минск, Республика Беларусь

Череватенко Н. П.

Самаль Д. И. – канд. техн. наук, доцент

Первый веб-сайт появился двадцать лет назад и представлял из себя статическую страницу с текстом, спустя некоторое время компания Microsoft предоставила технологию ActiveX, появились каскадные таблицы стилей, веб страницы стали приобретать форму и функциональность. Они превратились в сложнейшие приложения по обмену информацией, просмотра медиа-файлов, место запуска игр и даже обработки графики. Изменились и методики их написания.

Современный веб-сайт разделен на следующие части, веб сервер для статических данных, при большом количестве пользователей применяется так же балансировщик нагрузки, сервер предоставляющий интерфейс доступа к данным, в зависимости от задач может быть использован еще один сервер для фронтенда, часть статических данных может быть отдана в сеть доставки контента. Разбивка на составляющие позволяет вести работу над каждым модулем по отдельности. Сервер данных не имеет графической составляющей и лишь предоставляет интерфейс для доступа к базе данных, может так же выполнять следующие роли, рассылка писем, выгрузка и обработка информации с других серверов данных, авторизация пользователей и контроль их прав, кеширование информации для ускорения отдачи данных. Сервер для статических данных для веб страницы используется как основная точка входа на сайт, отдает статику в виде html/js/css файлов а так же файлов картинок, аватарок пользователей и другой медиа информации не требующей обработки перед отправки пользователю веб-сайта. На нем как правило используется кеширование запросов и применения алгоритмов сжатия трафика для уменьшения размера статических файлов и соответственно ускорения загрузки страницы пользователя. Сервер для фронтенда служит буфером между сервером данных и конечным пользователем. Основная задача увеличить скорость загрузки страницы путем формирования части информации на сервере или кеширования данных для начальной загрузки страницы и вставка их в тело страницы или в виде загружаемого ресурса. Так же может выполнять роль прокси-сервера к серверу данных, переформировывать запросы данных или же их мультиплексирование, может быть использован так же для push-нотификации, при этом держа постоянное соединение с сервером данных и получая последнюю информацию об изменениях в слое данных. Может так же кешировать запросы данных. CDN - сеть доставки контента, позволяет оптимизировать доставку статических ресурсов конечному пользователю, осноное применение это доставка джаваскрипт библиотек, хранение и доставка изображений, реже видеофайлов. Популярные ресурсы такие как facebook и vkontakte в первую очередь хранят все фотографии пользователей и их аватары в CDN дабы уменьшит время загрузки данной информации, что же до библиотек приложений, все они уже загружены и доступны на соответствующих ресурсах и могут быть загружены оттуда, к примеру с сайта cdnjs.

Задачей настоящей работы является построение и развертывание веб-приложения для обработки и визуализации данных сервиса микроблогинга Twitter. В качестве сервера данных задействован java-сервер, обрабатывающий поступающие данные из сервиса Twitter, приложение написано на языке джаваскрипт с использованием библиотек Backbone, Marionette, Lodash. Backbone - предоставляет формат организации модели данных и синхронизации их с сервером. Marionette - использована для построение интерфейса пользователя, тесно интергируется с библиотекой Backbone. Lodash - универсальная библиотека для обработки данных.

Для разработки веб-сайта проекта, использованы следующие подходы. Автоматизация сборки проекта, под ней подразумевается уход от старого способа разработки и сборки сайта с использованием множества труда разработчика по компонованию всех необходимых ресурсов и сборки пакета в сторону модульной сборки и выполнения всех подготовительных операций программой сборщиком. Сейчас имеются следующие популярные инструменты для сборки: gulp, grunt, broccoli. Gulp обладает преимуществом по сравнению с остальными, а именно параллельным выполнением этапов сборки и обработки файлов. Что следует обязательно включать в этапы сборки веб-сайта, конкатенацию используемых джаваскрипт файлов и файлов стилей. Конкатенированные файлы не обязательно могут быть представлены в форматах используемых браузером, они могут быть так же компилируемы в этот формат, для этого перед этапом сборки следует выполнить компиляцию этих файлов. Помимо конкатенации может быть применена минификация сконкатенированных файлов и библиотек. Для упрощения работы с изображениями, их так же можно конкатенировать в одно большое изображение называемое спрайтом, снижая количество запросов к серверу и ускоряя отображение информации. Количество запросов в современных одностраничных приложениях критически важно, современные браузеры ограничивают количество одновременнных запросов к серверу до 6-7, в зависимости от используемого браузера, что препятствует скорости загрузки страницы и сопутствующих данных с сервера. Для этого следует уменьшить количество используемых файлов на странице, можно так же вставлять часть необходимой для начальной загрузки информации в домашнюю страницу, вынося критически важные части стилей и программного кода прямо в тело страницы и последующей загрузки остальной части файлов после загрузки и отображения веб-страницы. Помимо компрессии использованы так же инструменты по проверке качества кода и нахождения в нем уязвимых мест, что позволяет в дальнейшем избегать ошибок в разрабатываемом приложении.

Это лишь немногие компоненты архитектуры одностраничных приложений, опираясь на которые можно с легкостью разрабатывать высоконагруженные веб-сайты, при этом не теряя времени и усилий на формирования всех данных перед размещением на веб-сервер и предоставлением доступа пользователям.

Список использованных источников:

1. Zakas, N.C. High Performance JavaScript – Orelly, 2010. – 211 p.
2. Zakas, N.C. Professional javascript for web developers – Wiley, 2012. – 964 p.
3. Stefanov, S. JavaScript Patterns – Orelly, 2010. – 236 p.
4. Школа разработки интерфейсов. Яндекс [Электронный ресурс] / Компания Яндекс. Российская Федерация. - Москва, 2014. - Режим доступа: HYPERLINK "Школа разработки интерфейсов — Обучение в Яндексе" https://academy.yandex.ru/events/shri/. – Дата доступа: 25.03.2015.