Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Уральский федеральный университет**

**имени первого Президента России Б.Н.Ельцина»**

**Институт фундаментального образования**

**Кафедра интеллектуальных информационных технологий**

**«Разработка прогрессивного веб-приложения «Мои задачи»»**

Пояснительная записка

Руководитель Свинцов Д.В.

Группа ФО-340001

Студенты Рагозин Н.А.

Екатеринбург – 2016

**Введение**

Представьте, что ваш сайт взаимодействует с пользователем как приложение. То есть пользователь может установить его на любой гаджет, получать уведомления и работать с ним (даже без интернет-соединения). Это уже реальная технология, которую активно продвигает Google, а также многие другие крупные компании.

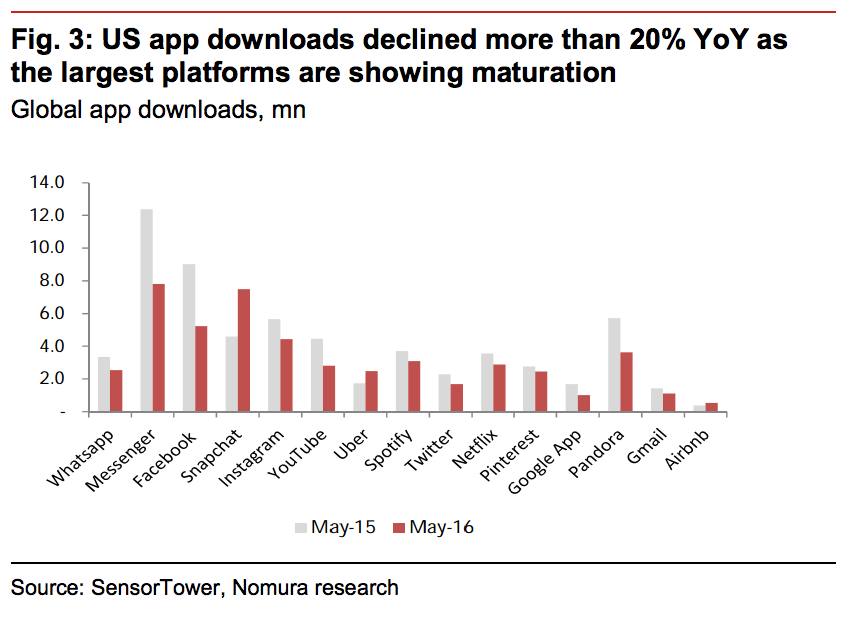
Progressive Web Apps (PWA) Google анонсировал еще в 2015 году. Этот формат создания мобильных сайтов привлек внимание благодаря относительной простоте разработки и почти мгновенному взаимодействию с пользователем. Уже в мае 2016 на конференции разработчиков Google I/O The Washington Post продемонстрировал свой мобильный гибридный [сайт-приложение](https://www.washingtonpost.com/pwa/?tid=a_inl).

Что же такое PWA? Можно воспринимать это как сайт, построенный с использованием веб-технологий, но который взаимодействует с пользователем как приложение. Достижения в разработке браузеров, кэше и Push-интерфейсах позволили устанавливать приложение на домашний экран напрямую из браузера, получать всплывающие уведомления и даже работать в автономном режиме.

У меня появилась идея: с использованием современного стека технологий создать прогрессивное веб-приложение. В качестве темы приложения мною была выбрана тема простого таск-менеджера.

**Обоснование**

В июне издание Recode недавно опубликовало исследование, которое подтвердило то, что уже у многих специалистов крутится на языке: бум традиционных приложений прошел, а процент тех, кто деинсталлирует приложения, продолжает расти.



Comscore недавно сообщил, что большинство пользователей смартфонов скачивают ноль приложений в месяц. Лишь около одной трети владельцев смартфонов скачивают хоть что-то, и большинство из них загружают от одного до трех приложений в месяц.

Прогрессивные веб приложения решают эти проблемы. И вот как:

1. Пользователи могут переходить на прогрессивные приложения из ссылок в соцсетях, во время просмотра веб-страниц или непосредственно из выдачи. Согласитесь, это гораздо удобнее, чем конкурировать за просмотры с двумя миллионами доступных приложений на IOS App Store или Google Play Store.

2. Предложение установить прогрессивное приложение показывается только тогда, когда веб-приложение отвечает определенным критериям, и пользователь продемонстрировал интерес посредством повторного посещения сайта.

3. Установка приложения происходит мгновенно. Все компоненты, которые требуют длительной загрузки, уже были установлены в кэш при первом посещении сайта пользователем.

4. Прогрессивные приложения гораздо меньше по размеру, так как они эффективно используют возможности браузера.

5. Всплывающие уведомления, работа в автономном режиме и все другие функции прогрессивного приложения будут работать, даже если посетитель никогда его не устанавливал.

6. Обычные приложения могут быть использованы только на той платформе, для которой они созданы. Вы не можете использовать приложение на вашем компьютере так же, как на Android или iOS устройстве. Кроме того, использовать приложение могут только те, кто его установил.

**Описание**  
Приложение должно иметь следующие функциональные требования:

* Авторизация.
* Пользователь может создавать, просматривать и удалять свои задачи.
* Доступ к приложению с любого устройства.
* Работа в оффлайне при загруженном приложении.

Приложение использует следующие технологии:

* React
* Gulp
* Babel
* Bower
* MaterializeCSS
* NPM
* ES6
* SCSS
* HTML5 Web Storage
* XMLHttpRequests
* Python
* Pyramid
* PostgreSQL
* Облачные хранилища Cloud9 и Heroku
* Nodejs
* Telegram API
* Jinja2
* REST API

**Инструкция пользователя**  
При входе в приложение пользователь видит перед собой форму входа, в которую вносятся регистрационные данные пользователя. Если же пользователь не зарегистрирован, то по нажатию кнопки «Зарегистрироваться» он будет перенаправлен на страницу регистрации.

Регистрация проходит вполне интуитивно: пользователь вносит данные и нажимает кнопку «Зарегистрироваться». Далее он будет перенаправлен непосредственно в главное окно приложения.

Главное окно приложения представляет из себя список задач и поле для ввода новых заданий. С помощью кнопки с «Минус» интуитивно понятно, что происходит удаление текущей задачи (у каждой задачи своя кнопка удаления). С помощью ввода осуществляется добавление заданий в общий список задач.

Выполнение задания осуществляется через двойное нажатие на прямоугольник, в котором он содержится. Задание померится как выполненное сразу на всех авторизованных устройствах одновременно.

Всплывающие подсказки, разбросанные по всей программе, позволят пользователю вносить корректные данные (также любое действие проверяется на допустимость и корректность). Если же была совершена ошибка, программа предложит свою подсказку.

Индикаторная полоса вверху приложения визуально показывает пользователю статус текущего подключения к серверу программы. Зеленый цвет говорит о штатной работе, красный о работе в оффлайн режиме. В оффлайн режиме доступны все те же действия, что и с сетью. При появлении сети у клиента данные автоматически синхронизируются между клиентом и сервером.

Также приложение доступно в качестве бота для телеграмма. В боте предусмотрена авторизация и получение текущих задач. Доступные команды указаны непосредственно в диалоге с ботом, а чтобы начать им пользоваться, достаточно что-нибудь ему написать и он готов к общению через команды.

Данное приложение можно установить на телефон. Достаточно открыть приложение в браузере и в настройках страницы выбрать пункт «Добавить на домашний экран». Браузер выполнит инструкции, содержащиеся в файле manifest.json и установит приложение вместе с кешем на ваше устройство.

**Выводы**

Мною было изучено достаточно количество технологий, однако впереди есть еще достаточно того, что можно улучшать. На текущий момент приложение доступно онлайн любому желающему, не содержит ошибок и имеет возможность установки на устройства. В связи с работой сервера на технологии pyramid не все, что мне бы хотелось, удалось реализовать. Например, для меня оказалось довольно сложным разобраться с работой сокетов под pyramid.

Изначально приложение имело server-side рендеринг React, который в тот момент находился в стадии тестирования. Изучив, как работает компоновка страницы на сервере, я перенес рендеринг непосредственно на клиент, непосредственно доработав приложение.

Поставленные изначально задачи были выполнены в полном размере, в результате получено следующее:

1. Создан веб-сервер на языке программирования Python с помощью фреймворка Pyramid, использующий шаблонизатор Jinja2.
2. Создана и подключена база данных Postgresql, которая в коде находит свое отражение в коде через ORM-систему SqlAlchemy, а также базируется на облачной PaaS-платформе Heroku.
3. Создан SPA с использованием ряда технологий, которые я перечислял выше, а также возможностью установки на мобильные системы.
4. Общение клиента и сервера осуществляется по REST API.
5. Получен навык в верстке, сборке как клиентской, так и серверной части, разработке авторизации, использование react-router и многих-многих библиотек и фреймворков.
6. Получен опыт создание клиент-серверных приложений а также выгрузка их на удаленные сервера.
7. Получен навык управление системами на linux, osx по средствам командной строки.
8. Был разработан способ рендеринга React на сервере python (через рендер-сервер, который можно найти в исходном коде).

**Литература**

1. Nodejs Documentation [Электронный ресурс] URL: <https://nodejs.org/api/> (20.12.2017)
2. PostgreSQL Documentation [Электронный ресурс] URL: <https://www.postgresql.org/docs/> (20.12.2017)
3. React Documentation [Электронный ресурс] URL:<https://facebook.github.io/react/docs> (20.12.2017)
4. Gulp Documentation [Электронный ресурс] URL: <https://github.com/gulpjs/gulp/blob/master/docs/README.md> (20.12.2017)
5. Основы веб программирования [Электронный ресурс] URL: [http://lectureswww.readthedocs.io/](http://lectureswww.readthedocs.io/6.www.sync/3.framework/pyramid/0.introduction.htmlhttp://lectureswww.readthedocs.io/6.www.sync/3.framework/pyramid/0.introduction.htmlhttp://lectureswww.readthedocs.io/) (20.12.2017)
6. React-router Documentation [Электронный ресурс] URL: <https://github.com/ReactTraining/react-router/tree/master/docs> (20.12.2017)
7. ECMAScript 2016 Language Specification [Электронный ресурс] URL:[www.ecma-international.org/ecma-262/6.0/](http://www.ecma-international.org/ecma-262/6.0/)(20.12.2017)
8. Heroku Dev Center [Электронный ресурс] URL: <https://devcenter.heroku.com/> (20.12.2017)
9. The Pyramid Web Framework [Электронный ресурс] URL: http://docs.pylonsproject.org/projects/pyramid/en/latest/index.html (20.12.2017)
10. Telegram Bot API [Электронный ресурс] URL: <https://core.telegram.org/bots/api> (20.12.2017)