УДК 004.91
DOI: 10.18101/978-5-9793-1626-0-101-105

РАЗРАБОТКА ПРОГРЕССИВНОГО ВЕБ-ПРИЛОЖЕНИЯ

© **Мархакшинов Аюр Лувсаншаравович**
кандидат технических наук, доцент,
Бурятский государственный университет имени Доржи Банзарова
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
ayurmar@yandex.ru

© **Богидаева Кристина Михайловна**
студент,
Бурятский государственный университет имени Доржи Банзарова
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
ibogidaeva81@gmail.com

Аннотация. В статье описывается процесс создания прогрессивного веб-приложения (PWA, Progressive Web Application). Приложения данного типа используют возможности современных браузеров для повышения качества пользовательского взаимодействия с веб-приложениями и являются альтернативой нативным мобильным и десктопным приложениям. К основным функциям PWA, недоступным традиционным веб-приложениям, относятся возможность автономной работы, поддержка всплывающих уведомлений на устройствах Android, доступ к периферийному оборудованию устройства (например, к камере смартфона, геолокации, акселерометру и т. д.). В данной статье рассмотрены этапы, необходимые для преобразования стандартного веб-сайта в PWA-приложение с функцией работы в офлайн-режиме.
Ключевые слова: прогрессивное веб-приложение, progressive web application, манифест PWA, Service Worker, разработка веб-приложений

Для цитирования
Мархакшинов А. Л., Богидаева К. М. Разработка прогрессивного веб-приложения // Информационные системы и технологии в образовании, науке и бизнесе: материалы региональной научно-практической конференции с международным участием (Улан-Удэ, 1 июля 2021 г.) / отв. ред. А. А. Тонхоноева, науч. ред. Е. Р. Урмакшинова. Улан-Удэ: Изд-во Бурят. гос. ун-та, 2021. С. 101–105.

Введение

Прогрессивные веб-приложения позволяют сделать традиционные веб-сайты «устанавливаемыми», то есть сохранить определенную часть контента в памяти устройства, добавить иконку приложения на экран и обеспечить запуск страницы в отдельном окне с минимальным интерфейсом, что делает взаимодействие с веб-сайтом похожим на работу с нативным приложением.

Очевидно, что процесс разработки мобильного приложения для Android и/или iOS требует дополнительных затрат и существенно отличается от разработки веб-приложения используемым стеком технологий. Созданное мобильное приложение фактически является отдельным программным продуктом, которому необходима поддержка и сопровождение.

При публикации мобильного приложения неизбежно возникают временные и финансовые издержки, связанные с оплатой аккаунта разработчика App Store или Google Play, а также прохождением рецензирования приложения.

PWA-приложения позволяют отказаться от услуг магазинов мобильных приложений и предложить посетителям веб-сайта бесплатную альтернативу. При этом вся необходимая кодовая база относится непосредственно к проекту веб-приложения и размещается на хостинге.

Преобразование стандартного веб-сайта в PWA-приложение выполняется в три этапа:

- добавление манифеста PWA-приложения;
- разработка скрипта Service Worker;
- обеспечение доступа к сайту по протоколу HTTPS.

Манифест PWA-приложения

Манифест является ключевым элементом технологии PWA и представляет собой текстовый файл, содержащий информацию о приложении в формате JSON. Ссылка на файл манифеста должна быть размещена в HTML-разметке веб-страницы в тэге <head>:

```
<head>
        …
        <link rel="manifest" href="manifest.json" />
        …
</head>
```

Структура и содержимое манифеста описываются соответствующим стандартом W3C[1], поэтому ограничимся рассмотрением лишь наиболее важных полей:

- *name* — поле, в котором указывается строка с названием приложения. Значение используется для отображения пользователю, например в списке приложений или в качестве подписи к иконке приложения;
- *short_name* — сокращенная версия названия приложения. Используется, если для отображения полной версии названия недостаточно места на экране устройства;
- *start_url* — начальный URL-адрес, который должен загружаться при запуске PWA-приложения пользователем;
- *display* — режим отображения приложения, определяет вариант интерфейса браузера при просмотре. Значения варьируются от обычного браузерного варианта просмотра до полноэкранного режима;
- *icons* — массив, содержащий сведения об изображениях, предназначенных для использования в качестве иконок PWA-приложения. Рекомендуется предоставлять набор иконок для наиболее популярных разрешений.

Ниже приведен пример файла PWA-манифеста:

[1] Web Application Manifest. URL: https://w3c.github.io/manifest/ (дата обращения: 24.05.2021). Текст: электронный.

```
{
 "name": "Кулинарные рецепты",
 "short_name": " PWA Рецепты",
 "start_url": "index.html",
 "display": "standalone",
 "background_color": "#f8f7f5",
 "theme_color": "#ffa500",
 "orientation": "portrait-primary",
 "icons": [
  {"src": "/images/icons/recipes_icon_48.png",
   "type": "image/png", "sizes": "48x48"},
  {"src": "/images/icons/recipes_icon_96.png",
   "type": "image/png", "sizes": "96x96"},
  {"src": "/images/icons/recipes_icon_144.png",
   "type": "image/png", "sizes": "144x144"},
  {"src": "/images/icons/icon-512x512.png",
   "type": "image/png", "sizes": "512x512"}
 ]
}
```

Современные браузеры автоматически распознают наличие PWA-манифеста на веб-странице и предоставляют функцию добавления приложения на домашний экран устройства.

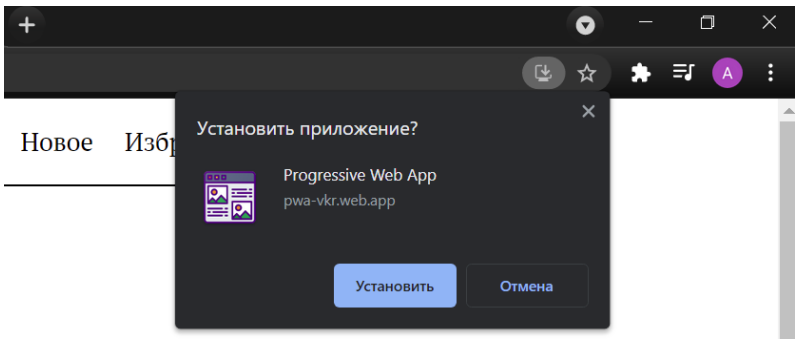

Рис. 1. Диалоговое окно установки PWA-приложения в Google Chrome

Скрипт Service Worker

Service Worker фактически представляет собой прокси-сервер между веб-приложением и сетью, реализованный в виде файла на языке JavaScript[1]. Первым шагом к использованию Service Worker является его регистрация в браузере, при которой происходит привязка скрипта к заданному URL. После регистрации Service Worker получает возможность контролировать запросы браузера к PWA-приложению. Например, для обеспечения автономной работы приложения за-

[1] Service Worker API. URL: https://developer.mozilla.org/ru/docs/Web/API/Service_Worker_API (дата обращения: 24.05.2021). Текст: электронный.

просы браузера перехватываются, проверяется доступность сети, и, в случае отсутствия интернет-подключения, запрашиваемый ресурс возвращается из локального хранилища. При этом для браузера результат выглядит как полноценный ответ от сервера[1].

Все остальные функции PWA-приложения также реализовываются в скрипте Service Worker – фоновая синхронизация данных, отправка всплывающих уведомлений, предварительная загрузка ресурсов и др. Service Worker разрабатывается на чистом JavaScript и не требует подключения дополнительных фреймворков или библиотек.

Пример кода Service Worker, обеспечивающего работу PWA-приложения в офлайн-режиме:

```
const cacheName = "pwa-example";
const assets = [
 "/",
 "/index.html",
 "/recipe.html",
 "/css/style.css",
 "/scripts/recipe.js",
 "/images/background.png",
 "/images/menu.svg",
 "/images/recipe_01.webp",
 "/images/recipe_02.webp",
 "/images/recipe_03.webp",
 "/images/recipe_04.webp"
];

self.addEventListener("install", installEvent => {
 installEvent.waitUntil(
  caches.open(cacheName).then(cache => {
   cache.addAll(assets)})
 )});

self.addEventListener("fetch", fetchEvent => {
 fetchEvent.respondWith(
  caches.match(fetchEvent.request).then(res => {
   return res || fetch(fetchEvent.request)})
 )});
```

Заключение

Доля веб-приложений, поддерживающих технологию PWA, постоянно растет, хоть и относительно медленными темпами. Сказывается тот факт, что пользователи за много лет привыкли скачивать мобильные приложения из магазинов App Store и Google Play и считают их единственными площадками распростра-

[1] Introduction to Service Worker. URL: https://developers.google.com/web/ilt/pwa/introduction-to-service-worker (дата обращения: 24.05.2021). Текст: электронный.

нения подобного программного обеспечения. Тем не менее PWA-приложения по функционалу становятся все ближе к нативным приложениям, предлагая гораздо более удобный и дешевый способ разработки, сопровождения и дистрибуции.

Литература

1. Ater T. Building Progressive Web Apps. New York: O'Reilly Media, 2017. 275 p.

2. Киселев П. В. Прогрессивные веб-приложения: объединяющая технология для веб- и нативных приложений // Политехнический молодежный журнал. 2020. № 2(43). С. 1–9. Текст: непосредственный.

PROGRESSIVE WEB APPLICATION DEVELOPMENT

Ayur L. Marhakshinov
Cand. Sci. (Engineering), A/Prof.,
Dorzhi Banzarov Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: ayurmar@yandex.ru

Kristina M. Bogidaeva
Student,
Dorzhi Banzarov Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: ibogidaeva81@gmail.com

Abstract. Progressive web application (PWA) development process is described in the article. These applications use modern browsers capabilities to enrich user experience of interacting with web applications and provide an alternative to native mobile and desktop applications. Main features of PWA are offline work, push-notifications support for Android devices, access to additional device functions (i. e. smartphone camera, geolocation, accelerometer and so on). This article describes steps required in order to convert standard web site to the PWA with offline work feature.

Keywords: progressive web application, PWA manifest, Service Worker, web-development