Implementing Progressive Web Apps using React

A Practical Guide to create Web Apps that provides a native experience to the users

Enrique Pablo Molinari

Contents

About the Author

My name is Enrique Pablo Molinari. I have been working in the software industry for the last 22 years, working in different software projects from different companies as developer, technical lead and architect. I'm a passionate developer and also a passionate educator. In addition to my work on the software industry, I'm teaching Object Oriented Design and Advance Database Systems at Universidad Nacional de Río Negro.

Implementing Progressive Web Apps is my third book. I have also written Understanding React, a book to start learning React. And Coding an Architecture Style, a book about hands-on software architecture. You can find more about my thoughts on software development at my blog: Copy/Paste is for Word. I would be very happy if you want to ping me by email at enrique.molinari@gmail.com to send thoughts, comments or questions about this book, the others or my blog.

What is this book about?

In this book we are going to implement a Progressive Web App (PWA) using a step-by-step approach. This learning experience starts with an already crafted React application called Task List. To create this application I have used the create-react-app tool with the `cra-template-pwa` template, which gives us a good starting point for building a progressive web app.

We will go into the details about how to make the web app installable and after that how to improve the user experience by adding offline support to the web app. To make the learning process smooth I will first add offline support for the read-only use cases of the application. After that, I will show how to provide full offline support by using the IndexedDB database and the Background Sync.

I will also explain how to update progressive web apps. How users get notified that there is a new release of the application waiting to be installed and how that actually happens.

This book requires prior knowledge of JavaScript and React.

Development Environment

There are many development environments out there, and you can choose the one you are more comfortable with. In any case if you don't have a preference, I recommend Visual Studio Code (VS Code). And to be more productive I suggest installing the extension VS Code ES7 React/Redux/React-Native/JS snippets which provides JavaScript and React snippets. I would also suggest installing Prettier, which is a JavaScript/React code formatter.

To install an extension, in Visual Studio Code, go to the File menu, then Preferences and then Extensions. You will see a search box that will allow you to find the extensions that you want to install.

Finally, I really recommend configuring VS Code to format your source files on save. You can do that by going to the File menu, then Preferences and then Settings. On the search box type Editor: Format On Save. This will format your code right after you save it.

Chapter 1

Introduction

1.1 Progressive?

No one has doubts about what a Web App mean. But progressive? What does it mean?

The word **progressive** comes from the design philosophy known as progressive enhancement coined by Steven Champeon and Nick Finck in 2003. The idea is that you have to design your Web App to work in old devices, old browsers, with bad connectivity or no connectivity at all, to reach as many users as possible. But, it provides the best user experience for newer devices and browsers. In other words, as described by MDN (Mozilla Development Network):

> "The word progressive in progressive enhancement means creating a design that achieves a simpler-but-still-usable experience for users of older browsers and devices with limited capabilities, while at the same time being a design that **progresses** the user experience up to a more-compelling, fully-featured experience for users of newer browsers and devices with richer capabilities".

As we will see later in this book, there are many new features implemented by modern browsers. For instance, there is one particularly nice that allows your Web App to work offline. However, this and many other capabilities have been added during recent years to browsers at different times and some of them partially supported. So, feature detection is the technique that we usually use to determine if that functionality can be used or not. In this way, you can build a Web App that **progresses**.

1.2 And... Progressive Web Apps?

THIS IS A SAMPLE BOOK

1.3 Capabilities of Progressive Web Apps

THIS IS A SAMPLE BOOK

Chapter 2

Crafting your First PWA

In this chapter we will convert with a step-by-step approach an existing React application, called Task List, into a progressive web app. It is a full stack application with back-end services written in Java.

2.1 Configure the Development Environment

THIS IS A SAMPLE BOOK

2.2 Task List React Application

The main screenshots of the Task List application are illustrated in figures **??** and **??**. In order to access their task lists a user must type first their username and password (in the login screen **??**). That will generate a request to the UserAuth back-end service which validates the user's credentials, and if successful it will return an access token. The access token allows the user to access their tasks consuming the Task List back-end services. The token is stored in an **httpOnly** cookie. Once authenticated, the user can retrieve their tasks. They are presented in a list as shown in figure **??** with their expiration date. The expiration date will have different background colors depending how close to the deadline they are. To mark a task as done (or in progress), the end user can click on the checkbox. The second task on figure **??** shows how a task looks like when it is marked as completed. You can also delete tasks or add new tasks.

THIS IS A SAMPLE BOOK

2.3 The Service Worker

You might have used the tool called create-react-app to create and manage React applications. Luckily for us, this tool allows us to create Progressive Web Apps too. To create a new React application with PWA support, you can run the following command:

```
$ npx create-react-app my-app --template cra-template-pwa
```

The Task List application we have described before, was created using that template[1]. However, the PWA capabilities are not enabled yet, we will start with that in a few more paragraphs. When you create a React application using this tool and with this template, the PWA capabilities are not enabled by default, you have to enable them.

THIS IS A SAMPLE BOOK

2.4 Making the Web App Installable

THIS IS A SAMPLE BOOK

2.5 Supporting Offline

At this point the Task List application is installable, cool!. But if we try to use the application, while we are **offline**, it will crash. This experience is not really nice for a web app that should look like a native app. Let's fix it.

THIS IS A SAMPLE BOOK

[1]After creation I have upgraded all the workbox packages to latest version, which at the time of writing this book is 6.4.2

Chapter 3

Handling New Releases

Once you have the first release of a progressive web app on the market and installed on your clients, how can you handle a new release? Think about it. You will have all the static assets of the application stored in the browser's cache of your clients and the application works with a cache-first strategy. What should you do to make your application know that there is a new release deployed on the server?

THIS IS A SAMPLE BOOK

Chapter 4

Incorporating the Sync capability