**Technological University of Tijuana**
Software Development and Management

# **Title:** Tools and Execution of Progressive Web Applications (PWAs)

**Author:** Jose Eduardo Ruiz Alvarez
**Group:** 10B
**Subject:** Progressive Web Applications

**Date:** February 2, 2024

# Contents

**1 Introduction** **3**
1.1 Purpose . . . 3
1.2 Scope . . . 3
1.3 Definitions, Acronyms, and Abbreviations . . . 3
1.4 References . . . 3
1.5 Overview of the Document . . . 3

**2 Development and Execution of Progressive Web Apps (PWAs)** **4**
2.1 Requirements . . . 4
2.2 Installation . . . 4
2.3 Browser Compatibility . . . 5
2.4 Development Tools . . . 5

**3 Conclusion** **6**

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to explore the development and execution tasks of Progressive Web Applications (PWAs), focusing on requirements, installations, browser compatibility, and development tools.

## 1.2 Scope

This document covers the specific aspects related to the development and execution of PWAs, including necessary requirements and tools.

## 1.3 Definitions, Acronyms, and Abbreviations

- **PWA:** Progressive Web Application
- **HTTPS:** Hypertext Transfer Protocol Secure
- **Node.js:** A JavaScript runtime for server-side programming
- **IDE:** Integrated Development Environment
- **Git:** A distributed version control system

## 1.4 References

1 Mozilla Developer Network (MDN). (2023, December 15). *Aplicaciones Web Progresivas | MDN*. Retrieved from `https://developer.mozilla.org/es/docs/Web/Progressive_web_apps`

2 IBM. (2022, February 10). *IBM Documentation: Service-Oriented Architecture (SOA).* Retrieved from `https://www.ibm.com/docs/en/rbd/9.5.1?topic=overview-service-oriented-`

3 Amazon Web Services, Inc. (2023). *Web Apps vs. Native Apps vs. Hybrid Apps - Difference Between Types of Web and Mobile Applications - AWS.* Retrieved from `https://aws.amazon.com/compare/the-difference-between-web-apps-native-apps-and-hybrid-ap`

## 1.5 Overview of the Document

This document will provide insights into the requirements, installations, browser compatibility, and development tools for Progressive Web Applications (PWAs).

# 2 Development and Execution of Progressive Web Apps (PWAs)

In this section, we will delve into the development and execution aspects of Progressive Web Apps (PWAs), covering requirements, installations, browser compatibility, and development tools.

## 2.1 Requirements

To effectively develop and run PWAs, certain requirements need to be met. These include:

1. **Web Browser:** Ensure that you have a modern web browser that supports PWA features. Popular choices include Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.
2. **Text Editor or IDE:** Choose a text editor or integrated development environment (IDE) for coding. Common options include Visual Studio Code, Atom, Sublime Text, or any IDE of your preference.
3. **Node.js:** Install Node.js to leverage npm (Node Package Manager) for managing dependencies and building scripts.
4. **Version Control System:** Consider using a version control system like Git to track changes and collaborate with others.
5. **HTTPS:** PWAs often require secure connections. Ensure your development environment supports HTTPS.

## 2.2 Installation

The installation process for setting up a PWA development environment involves the following steps:

1. **Node.js:** Download and install Node.js from the official website (`https://nodejs.org/`). Follow the installation instructions based on your operating system.
2. **Text Editor or IDE:** Choose and install a text editor or IDE according to your preference. Download and install it from the official website.
3. **Git:** If not already installed, download and install Git from `https://git-scm.com/`.
4. **Create a New Project:** Use the terminal or command prompt to create a new directory for your PWA project. Navigate into the project folder.
5. **Initialize Node.js:** Run ‘npm init‘ to initialize a new Node.js project. Follow the prompts to set up your project.
6. **Install Development Dependencies:** Use ‘npm install‘ to install necessary dependencies such as a service worker library or a build tool.
7. **Start Coding:** Begin coding your PWA using HTML, CSS, and JavaScript. Implement features like service workers for offline capabilities and responsive design for various devices.

8. **Test Locally:** Test your PWA locally by running it in your web browser.
9. **Set Up HTTPS:** If your PWA requires HTTPS, set it up in your development environment. Tools like ‘mkcert‘ can simplify this process.

## 2.3 Browser Compatibility

PWAs are designed to work across various web browsers. However, it's essential to be aware of potential variations in support. Here are some considerations:

- **Google Chrome:** Generally has robust support for PWA features.
- **Mozilla Firefox:** Provides good support for PWA features.
- **Microsoft Edge:** Offers solid compatibility with PWA features.
- **Safari:** While Safari supports many PWA features, some functionalities may differ. Always check and adapt for Safari compatibility.

## 2.4 Development Tools

Effective development of PWAs requires the use of appropriate tools. Some essential tools include:

- **Lighthouse:** A Chrome DevTools extension for auditing and improving the quality of PWAs.
- **Workbox:** A set of libraries and Node modules that make it easy to build scalable and feature-rich service workers.
- **PWACompat:** A set of web components and polyfills that provide PWA features in unsupported browsers.
- **Webpack or Parcel:** Build tools that help bundle and optimize your PWA code for production.

By leveraging these tools and following best practices, you can streamline the development and ensure the success of your Progressive Web App.

# 3 Conclusion

This document has provided insights into the requirements, installations, browser compatibility, and development tools for Progressive Web Applications (PWAs). Understanding these aspects is crucial for successfully developing and executing PWAs.