Bachelor's thesis

Infromation Technology

2019

Ilja Shustov

EVALUATION AND DEVELOPMENT OF A PROGRESSIVE WEB APP

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Infromation Technology

2019 | 30 pages

Ilja Shustov

EVALUATION AND DEVELOPMENT OF A PROGRESSIVE WEB APP

With the evolution of the web, and global popularization of smartphones, the market was missing a layer of something that could combine the features of native apps and the reach of the web. Progressive Web Applications is a response to this market demand.

This thesis was carried out to research and learn about Progressive Web Applications, with the implementation of one of them, as a simple news aggregator.

This thesis is divided into two parts: theoretical part and practical part. The theoretical part consists of information gathered mostly from the developers.google.com website and from other reliable online sources. It includes a comparison between Progressive Web Applications and its competitors. It also gives a description of the main aspects of Progressive Web Applications and the technologies they use.

The practical part describes the implementation of a Progressive Web App. Thus, a news aggregator application was developed with the use of such technologies as Service Worker, Web App Manifest, App Shell and Push Notifications. The developed app supports offline mode, is able to be installed on the home screen and has high performance, which was proven when the testing of it was carried out.

The thesis concludes that a Progressive Web App is a great combination of technologies that has a bright future despite facing some obstacles which are currently being solved with the support of such influential companies as Google.

KEYWORDS:

Progressive web application, multiplatform, App Manifest, Service Worker, App Shell Model, Push Notifications, HTTPS

CONTENTS

FIGURES

LIST OF ABBREVIATIONS

CSS	Cascading Style Sheets
GPS	Global Positioning System
HTTPS	HyperText Transfer Protocol Secure
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
JS	JavaScript
PWA	Progressive Web Application
SEO	Search Engine Optimization
SSL	Secure Sockets Layer
UI	User Interface
URL	Uniform Resource Locator

1 INTRODUCTION

1.1 Background

The web has changed significantly in the last years; it is upgrading and the functionality increased significantly. Some years ago, we had static web pages, and that was what the web was, but nowadays, with the advent of new technologies and the updates of the existing ones, we have this rich functionality.

One more thing which changed in the last years is that due to the global popularization of smartphones, and their enhanced performance, nowadays we use phones rather than desktop computers. It is an interesting fact, that while using smartphones, time spent in native apps is almost 7 times higher than in the web based on comscore.com[1] research.

Unfortunately, native app development is costly both from resources and time perspectives, due to the platform independence of native apps. Moreover, for the users installing a native app, it is a large commitment, due to the memory space it will take and the time they will spend on this app.

Overall, it seems that there should be something in the middle to meet the user's expectations.

Progressive Web Application (PWA) combines the capability of the native app and the reach of the web. It provides a great user experience that feels integrated. PWA is a web app, which moved to a new level, and provides new opportunities, thanks to several technologies combined, such as Service Worker, App Shell, Web App Manifest and Push Notifications.

Everything considered, it seems to be an interesting topic to research and the implementation of PWA with the crucial aspects highlighted, should help in better understanding the specifics of technologies.

1.2 Thesis Structure

This thesis is divided into 6 chapters. Chapter 1 is an introduction part, where the overview of the thesis topic is given and the initial idea is presented.

Chapter 2 is the theoretical part, where the main advantages and disadvantages of PWA are presented, moreover, research and description of PWA competitors are introduced.

Chapter 3 is the theoretical part too, where the core technologies of PWA are reviewed. Such technologies as Service Worker, Web App Manifest, Push Notifications and App Shell are reviewed.

Chapter 4 is the practical part where the implementation process of the PWA is described, with all main technologies implemented. It presents the application flow of this project.

In Chapter 5 the results of the practical part are evaluated, in particular, thanks to testing conducted with Lighthouse Tool. The final view of PWA News UI is shown.

Chapter 6 is the final chapter. It consists of summarizing the outcomes and describing achieved goals.

2 WEB AND MOBILE APPLICATIONS

This section briefly describes web, mobile and progressive web applications, showing their strengths and weaknesses. This is how a comparison between them takes place in this thesis.

2.1 Native Apps

Native Apps are applications that are developed for a particular platform, with the use of specific programming language for each platform, thus this specificity provides optimized performance and gives an opportunity to take full advantage of device features such as camera, Bluetooth, GEO-location, accelerometer and etc. Native apps have to be installed on the user's device separately and downloaded from the platform specific app store.

Due to platform dependence of native apps the time and the cost of their development increases. Due to the same reason, later updates are also becoming costly. Native apps are occupying a limited internal memory on user's devices. Especially noticeable and annoying for users it is becoming when app is used rarely or even just one single time.

2.2 Web Apps

Web apps are apps which the user runs in a web browser. Web apps are written using web coding languages, such as HTML, CSS, JS and etc. They have limited functions and cannot take full control of the device. Although a user has to be online to use those apps, and even poor internet connection may cause application failure.

The significant advantage of the web app is that they are multiplatform. Their development time is lower in comparison with native apps, therefore the cost is also decreasing. Since that they run in a web browser, a user does not have to download and install them separately.

2.3 Progressive Web Apps

PWA is a web app that combining new technologies with best practices for reaching reliability, performance, be engaging and to give a user native-like feelings. PWA characteristics are:

Reliable App loading and shows up immediately, regardless of the status and the quality of the network connection. The level of offline functionality will depend on the application – some applications will be able to function completely offline, while others will display meaningful placeholder data informing the user that they are offline. In no case should the application break or become unresponsive.[2]

Performance Network data exchange is fast, the UI is smooth and responsive. Opportunity to be able to adapt to any kind of device: mobile, desktop and tablet are important for PWA because it ensures compatibility even with new devices in the future.

Engaging PWA makes the user experience with the application comfortable and enjoyable, prompting him to want to experience it again and again. It ensures that the whole experience is still lightful. Making it easy for the user to do what they need to do. Using features like a web Push it is always up-to-date, keeping the user informed with notifications.

Safe It served with HTTPS. Having a secure connection really means 3 things: identity, confidentiality, and integrity. Keeping user safe is hugely important, due to high risks of getting into trouble.

PWA looks like native applications are, this cosmetic solution is important for the user from a psychological point of view. All the main resources of the application can be stored on the client, only dynamic content will be transmitted over the network.

There are differences between PWA and native applications - mainly in the rights of access to system resources, but work in this direction goes even on the field of pure HTML5. [3]

Hybrid Apps

Hybrid apps are built with web technologies and platform independent. Same as native apps they run locally on the user’s device. ’’Hybrid apps run inside a native container and leverage the device’s browser engine (but not the browser) to render the HTML and process the JavaScript locally. A web-to-native abstraction layer enables access to device capabilities that are not accessible in Mobile Web applications, such as the accelerometer, camera and local storage.'' [4]

Same as native apps, hybrid apps have to be downloaded and installed on the device separately. Even though the functionality of hybrid apps is higher than web apps, they still loosing to native apps, especially when the things are coming to some complicated logic of the app behavior.

3 ANALYSIS OF PWA CORE TEHNOLOGIES

In this chapter will be shown the main components of PWA. Such technologies as Service Worker, Web App Manifest, App Shell and Push Notifications are described. The use of those technologies is actually, what is making the usual web app to become a Progressive Web App.

3.1 Service Worker

The Service Worker can be called the heart of PWA. It presents the proxy layer between the frontend and the backend that is compiled in the browser and all of the browser requests go through it. This division into two independent layers allowed developers to make the transition from a regular web site to PWA as simple as possible.

From the repositories, the Service Worker has access to Cache Storage for web resources, and IndexDB for data. But, most importantly, complete freedom to implement business logic.

It gives an opportunity, for example, to accept a request from the browser, check the network status, take data from the storage, perform operations with them and return some result back to the browser and based on the developers's logic implemented it will either think that answers came from the server or from the internet. Two browser layers (client frontend and Service Worker) allow writing full-fledged applications.[3]

‘‘Using service worker you can hijack connections, fabricate, and filter responses.’’[5] So for the security reasons, Service worker works only over HTTPS, to avoid man-in-the-middle attacks.

Service worker lifecycle is separated from your web page. Below in Figure 1 is illustrated ‘‘a simplified version of the service worker lifecycle on its first installation.’’[5]

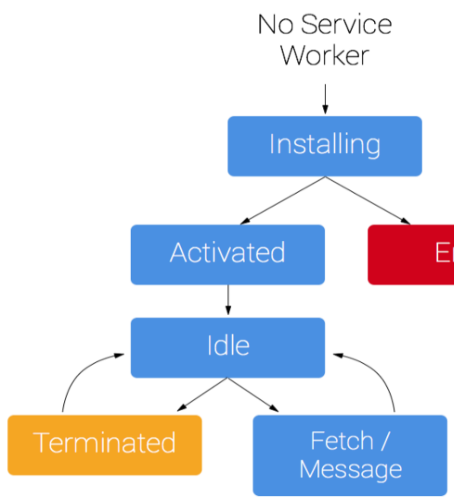
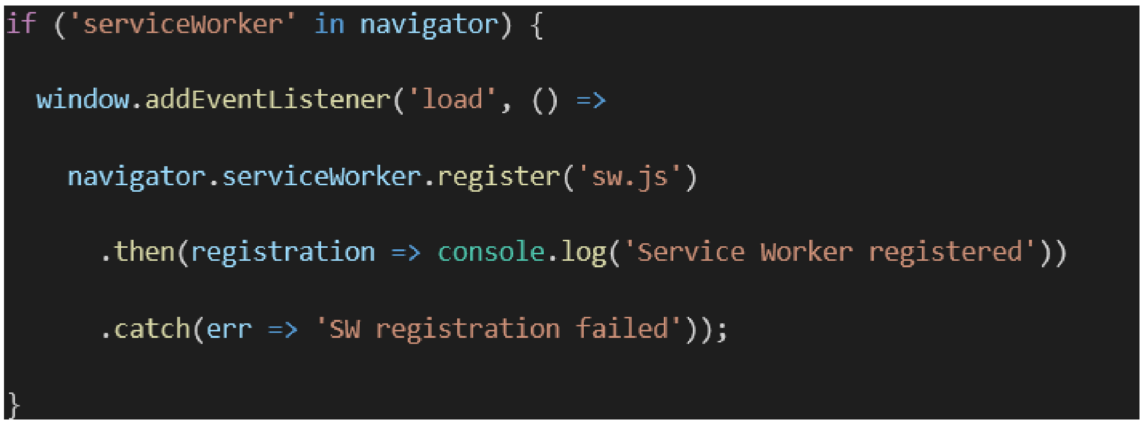


Figure 1. Simplified version of service worker lifecycle. Copied from [5]

From a programmer's point of view, the Service Worker is a javascript file that is included in the html code of the page. In it, the developer defines the logic of working with requests coming from the frontend and other functionality.

The registration process of Service Worker starts in app.js file, with declaring in it the location of service worker Javascript file, as illustarated in Figure 2.

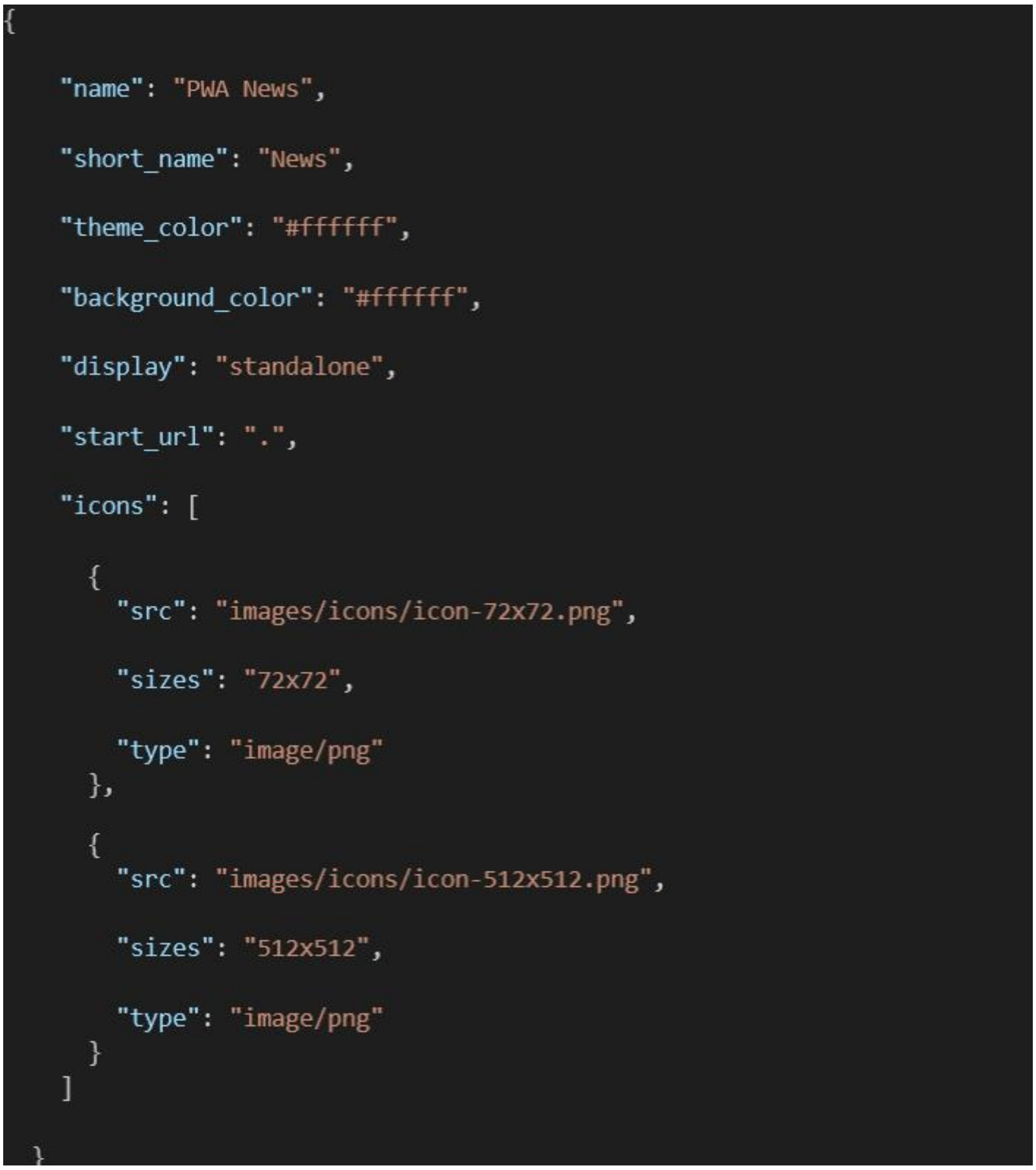


```
if ('serviceWorker' in navigator) {

  window.addEventListener('load', () =>

    navigator.serviceWorker.register('sw.js')

      .then(registration => console.log('Service Worker registered'))

      .catch(err => 'SW registration failed'));

}
```

Figure 2. Service worker registration process.

3.2 Web App Manifest

Web App Manifest is a simple JSON file, which provides to a browser some basic information about the app, such as the name of the app, icon, view of the app (standalone, fullscreen and etc.) and some other parameters. It gives an opportunity to install PWA in the home screen of a smartphone, which will look like a native app, and will give to a user native-app feel. Figure 3 below shows the manifest file for PWA.



```
{

    "name": "PWA News",

    "short_name": "News",

    "theme_color": "#ffffff",

    "background_color": "#ffffff",

    "display": "standalone",

    "start_url": ".",

    "icons": [

      {
        "src": "images/icons/icon-72x72.png",

        "sizes": "72x72",

        "type": "image/png"
      },

      {
        "src": "images/icons/icon-512x512.png",

        "sizes": "512x512",

        "type": "image/png"
      }
    ]

  }
```

Figure 3. Manifest.json file for PWA News app.

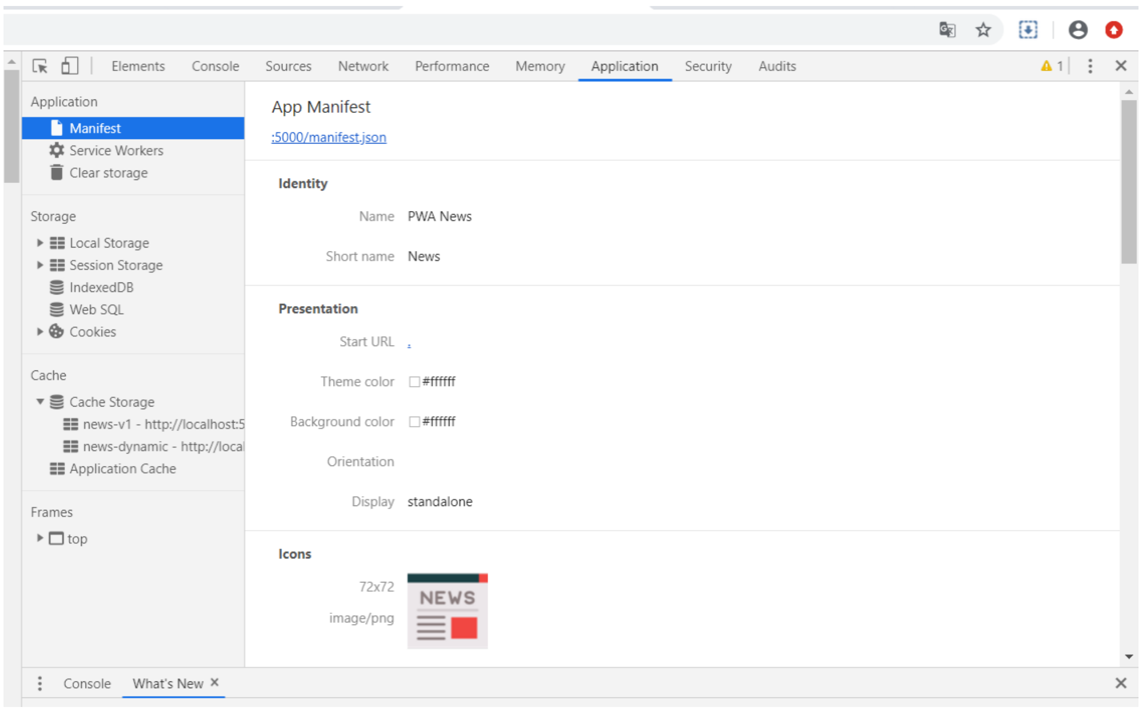
As it could be seen from Figure 3, the terms used in the manifest file are clear and have the explanations in their names. It is obligatory to have a name or short_name and recommended to have both, the short_name is used in cases of limited space on the user’s device. 'Icons' is an array of image objects. Each object should include the src, a sizes property, and the type of image. [6] Mainly icons are used in the home screen, but they also could be seen in the task switcher, splash screen and etc. Background_color is used when the user first time opens the app and the content is loading. 'Display' gives an option to set the UI based on application needs. Start_url setting the URL which will be open after the user will click on an icon in the home screen.

To start using a manifest file developer first have to register it in the index file, simply by adding a link to it, as it can be seen on the Figure 4 below.

```
<head>

<link rel="manifest" href="manifest.json">

</head>
```

Figure 4. Linking manifest.json file in the head of index.html file

With Google Chrome using DevTools developer can easily check the manifest file, as shown below in Figure 5.



App Manifest

:5000/manifest.json

Identity

Name	PWA News
Short name	News

Presentation

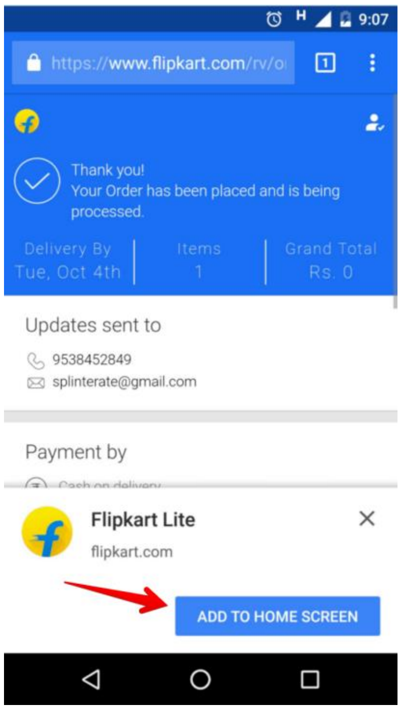
Start URL	.
Theme color	#ffffff
Background color	#ffffff
Orientation	
Display	standalone

Icons

72x72
image/png

Figure 5. App Manifest in Google Chrome Developer Tools

In order to have an ‘Add to Home Screen’ banner, some requirements have to be met, such as site have to use HTTPS, Service Worker has to be registered and the manifest file should have all mandatory parts (name, icons, start_url and display mode). Finally, when all of the conditions are fulfilled end-user will be able to see a banner prompting him to install the app to the home screen, as is shown in Figure 6.



Thank you!
Your Order has been placed and is being processed.

Figure 6. PWA installation banner on Chrome for Android (taken from [7]).

Web App manifest faced with a lack of support in the initial stage in some browsers, due to it PWA was not supported on IOS devices. But after a new upgrade in the middle of 2018 from Apple, it has the support of Web App manifest, even though it is still limited and some functions like installation events are not supported. Due to that Google released a new library called PWACompact, which allows using Web App manifest for the non-compliant browsers. Implementation of this library consists of adding a script to your code, as it could be seen on the Figure 7.

```
<link rel="manifest" href="manifest.webmanifest" />
<script async
src="https://cdn.jsdelivr.net/npm/pwacompat@2.0.8/pwacompat.min.js"
    integrity="sha384-
uONtBTCBzHKF84F6XvyC8S0gL8HTkAPeCyBNvfLfsqHh+Kd6s/kaS4BdmNQ5ktp1"
    crossorigin="anonymous"></script>
```

Figure 7. PWACompact library implementation script.

Nowadays most of the browsers having support of the Web App Manifest. Browsers like Chrome for Android, Edge, Chrome and Android Browser having the full support of all features of the Web App Manifest. Meanwhile, IOS Safari, Opera Mobile and Firefox for Android are still having partial support.

3.3 Application Shell

App Shell is a basic template for a website which consists of a minimum of HTML, CSS and Javascript. App Shell is cashed after the first visit of the user, then it saves on the users' device and loads instantly when the user opens the app next time. So it giving an opportunity to show for the user something immediately while dynamic information of the website is loading. It makes a user have a feeling of using a native app and increase the performance of the app. In other words, app shell could be described as a skeleton of UI, which stored locally. The difference between how app shell looks like with and without content could be seen in Figure 8 below.

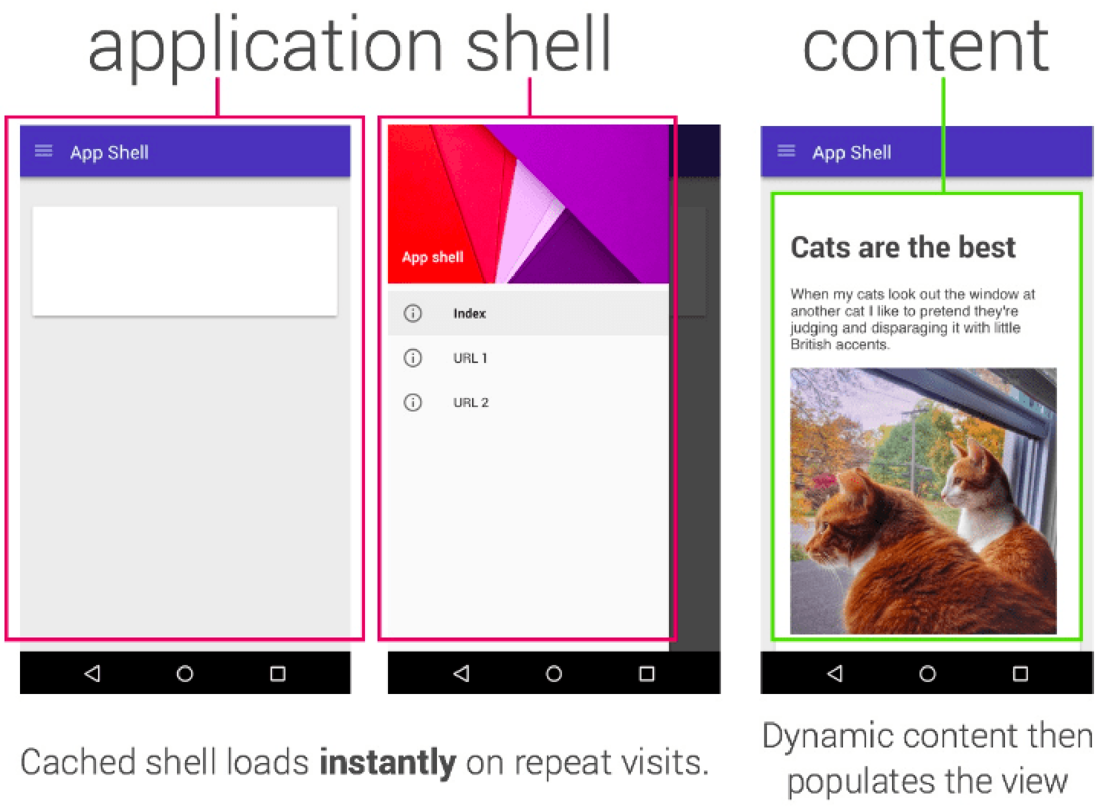

application shell
content
App Shell
App shell
Index
URL 1
URL 2
App Shell
Cats are the best
When my cats look out the window at another cat I like to pretend they're judging and disparaging it with little British accents.
Cached shell loads **instantly** on repeat visits.
Dynamic content then populates the view

Figure 8. App Shell with and without content. Copied from [8].

There are 3 main advantages of using an app shell with the service worker. The first one is a performance, high speed of showing to the user some animation on the repeat visits after the caching strategy is executed. For complete clarity, Google developers have made a test to see the gap between the first and repeat visits of the same website with app shell architecture and service worker enabled, the results of it are shown in Figure 9 below.

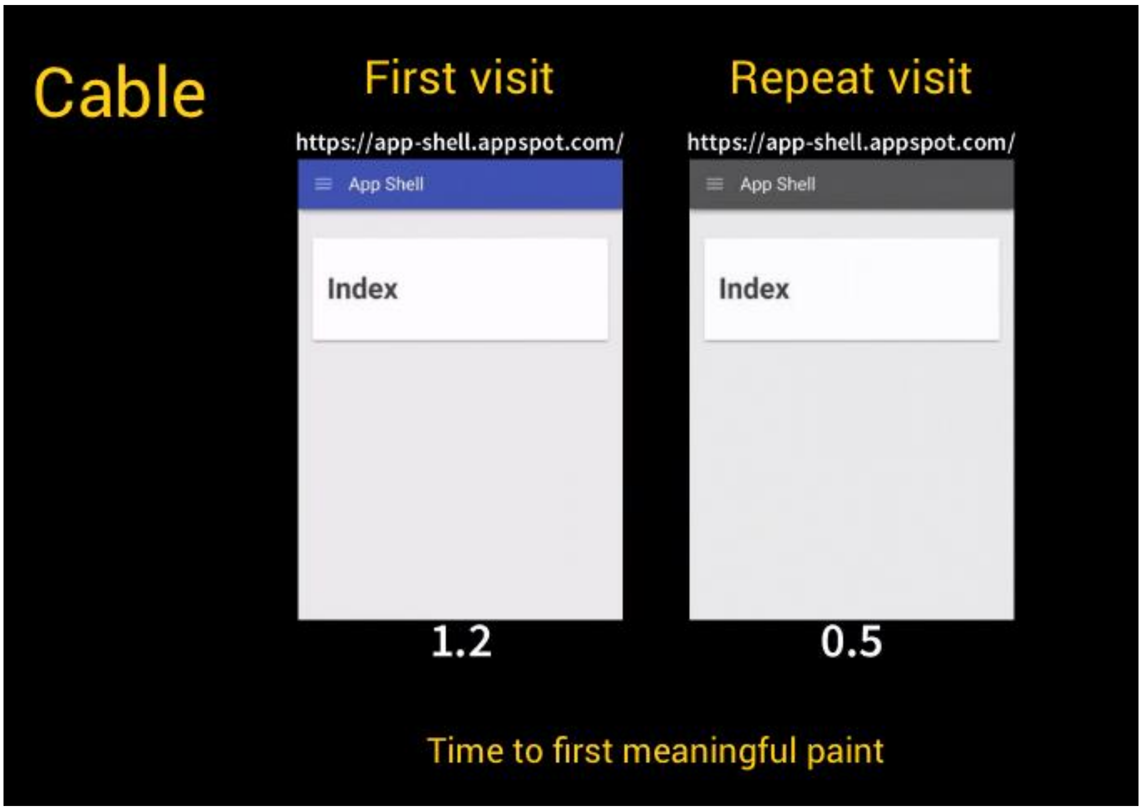

Cable
First visit
Repeat visit
https://app-shell.appspot.com/
App Shell
Index
1.2
https://app-shell.appspot.com/
App Shell
Index
0.5
Time to first meaningful paint

Figure 9. The loading speed of the app on the first and repeat visits (copied from [9]).

As illustrated in Figure 9, it took 1.2 seconds in the first case scenario and only 0.5 second in the second one, due to properly configured and implemented the work of service worker caching. Tests were carried out on cable with Nexus 5 using Chrome Dev Tools.

The second benefit of App shell gives to a user native-like interactions. ‘‘By adopting the app shell model, you can create experiences with instant, native-application-like navigation and interactions, complete with offline support.’’ [8]

Modest use of data is another advantage of the app shell combined with the service worker. During the development time, a web app is created to use a minimum amount of data, plus the result is achieved by using static assets from the cache and overall aggressive but at the same time reasonable cache strategy.

3.4 Push Notifications

Push Notifications was the thing which was missing in the web before, and made a big gap between the native and web apps, in the ability to reach the users' attention.

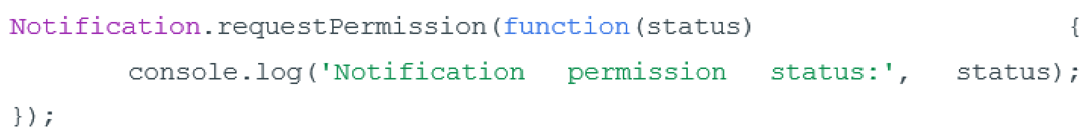
Push Notifications is one of the most popular technologies of PWA. In the initial stage of PWA gaining its popularity, most of the websites were using PWA only for Push Notifications. And if back then when you were visiting those websites, the first thing you were doing is searching for the ‘Close’ button, nowadays it changed, and the idea of wise use of notifications, that it is better do not bother the user and much better solution would be making those offers after the second or third time of user visiting the website.

There are several options that could be push notifications used for. ‘‘They can do simple things, such as alert the user to an important event, display an icon and a small piece of text that the user can then click to open up your site. You can also integrate action buttons in the notification so that the user can interact with your site or application without needing to go back to your web page.’’ [10]

Push notifications consist of two main technologies, it is Notifications API and Push API.

Notifications API

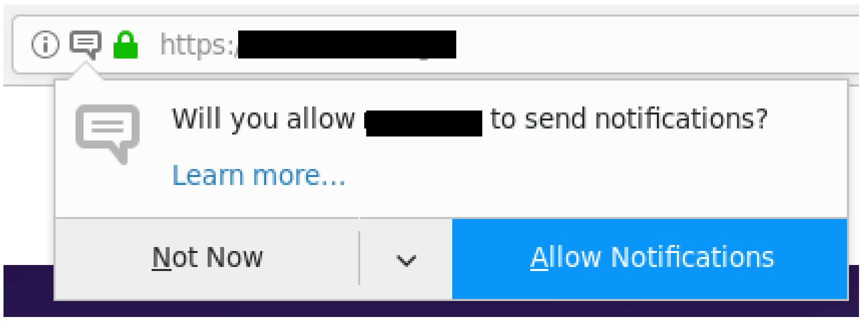
Notifications API allows us to show to the user some pop-ups or display any other information while he is using the browser. Before we could display those notifications, permission from the user have to be granted. It usually consists from adding two lines of code to our main Javascript file, as it is shown on the Figure 10 below.



```
Notification.requestPermission(function(status)                {
        console.log('Notification   permission   status:',   status);
});
```

Figure 10. Adding request permission for the Notifications to main.js file.

From the users’ point of view, it looks like it is shown in Figure 11 below.


https:[redacted]

Will you allow [redacted] to send notifications?

Learn more...

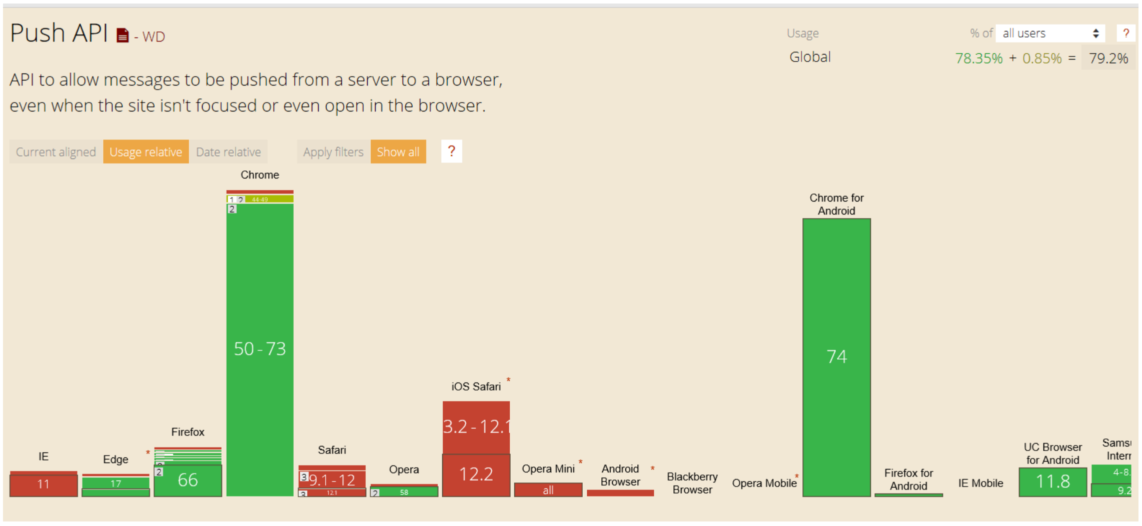
Figure 11. Push Notification request for later notifications.

Push API

Push API allows us to display notifications even when the browser is closed. It gives us an opportunity to engage the user. Implementation of Push API is more complex work in comparison with Notification API, but as a result, it provides the user the ability to interact with the app even without using the browsers itself. Overall, increasing the native app feeling.

Push API support

Push API faces a lack of support in some browsers, especially on IOS devices, as it could be seen from Figure 12 below.


Push API - WD

API to allow messages to be pushed from a server to a browser, even when the site isn't focused or even open in the browser.

Usage % of all users

Global 78.35% + 0.85% = 79.2%

Figure 12. Push API browsers support. [11]

4 IMPLEMENTATION OF PWA NEWS APP

This chapter will describedthe path of creating a PWA News App. The application was developed to show all of the features of PWA and all of the main technologies, such as Service Worker, App Shell, Push Notifications and Web App Manifest were used. It is a simple news aggregator with a wide choice of different news sources. News API (newsapi.org) was used to fetch the data from various sources. The app has an add to Home screen feature with one click from the pop-up notification, it also supports offline mode.

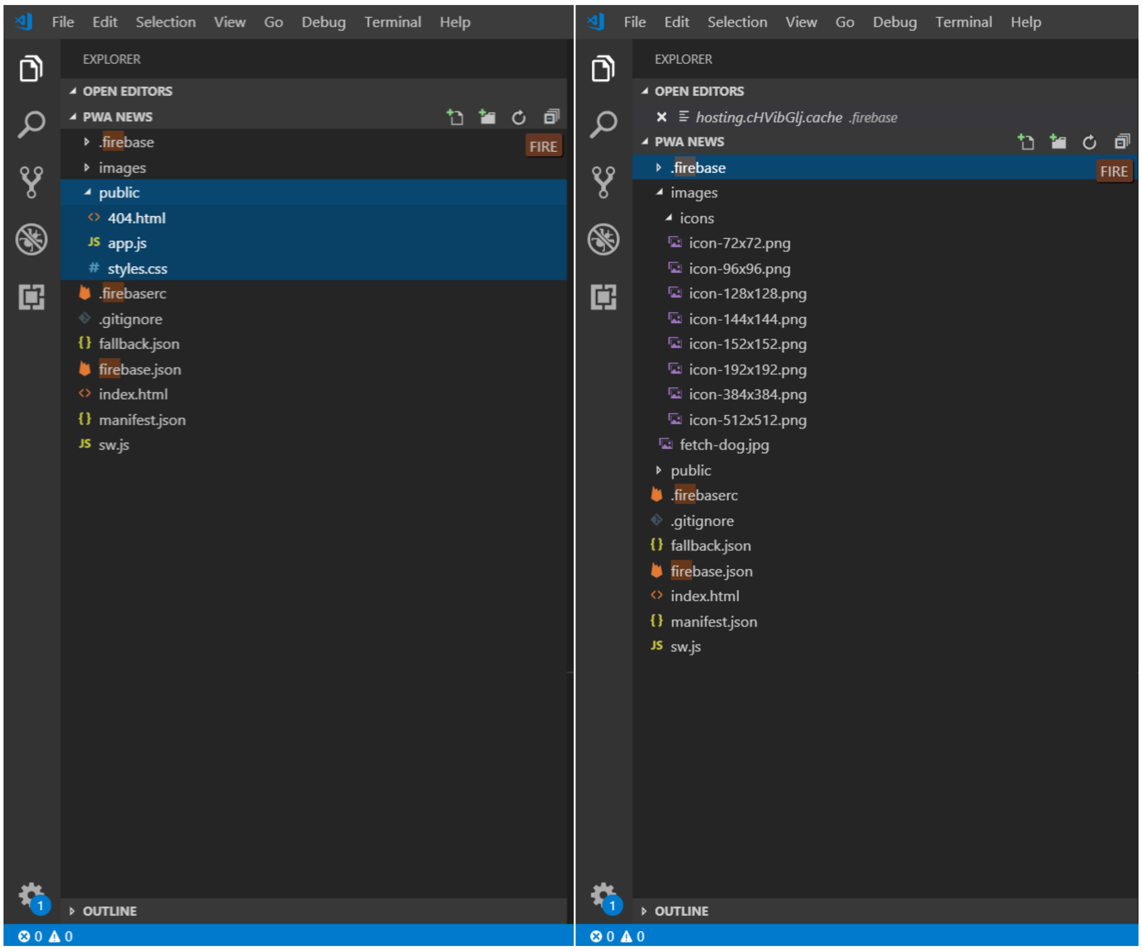
4.1 Development Environment

The application was developed using a HP Pavilion with Windows 10 as an operating system. Node.js was used to execute the Javascript code. NPM was installed and used as a package manager for Node modules. The code itself was written in Visual Studio Code, it is a code editor redefined and optimized for building and debugging modern web and cloud applications. [12] It has an impressive amount of different features and a great number of programming languages supported, overall fully covers our needs in this project.

Firebase was used for the hosting of our app. Firebase was chosen because it is free to use and has the support of all vital functions of the PWA, such as secure connection, API support and etc.

4.2 Application architecture

App implementation started with creating a new main folder. Structure of this folder shown in Figure 13 below.


File Edit Selection View Go Debug Terminal Help
EXPLORER
OPEN EDITORS
FIRE
OUTLINE
0 0
File Edit Selection View Go Debug Terminal Help
EXPLORER
OPEN EDITORS
hosting.cHVibGlj.cache .firebase
FIRE
OUTLINE
0 0

```
PWA NEWS
  .firebase
  images
  public
    404.html
    app.js
    styles.css
  .firebaserc
  .gitignore
  fallback.json
  firebase.json
  index.html
  manifest.json
  sw.js
```

```
PWA NEWS
  .firebase
  images
    icons
      icon-72x72.png
      icon-96x96.png
      icon-128x128.png
      icon-144x144.png
      icon-152x152.png
      icon-192x192.png
      icon-384x384.png
      icon-512x512.png
    fetch-dog.jpg
  public
  .firebaserc
  .gitignore
  fallback.json
  firebase.json
  index.html
  manifest.json
  sw.js
```

Figure 13. Structure of the PWA News app.

As it is shown in Figure 13 main folder consists of several subfolders and files.

Public

Public directory was created for hosting reasons during the initialization of the project with the Firebase project. This directory consist of the files served publicly - static assets.

404.html file was created when the initialization progress of Firebase hosting was running. It is a boilerplate web page that will appear in case anything goes wrong.

App.js file keeps in yourself the logic of the app. Here we have settled the creation of articles process and their updates.

Images

It is a folder with images, mainly consist from various size icons that need to fulfill the need of Web App Manifest, and all images and icons looked smooth in different types of devices and fit the screen size.

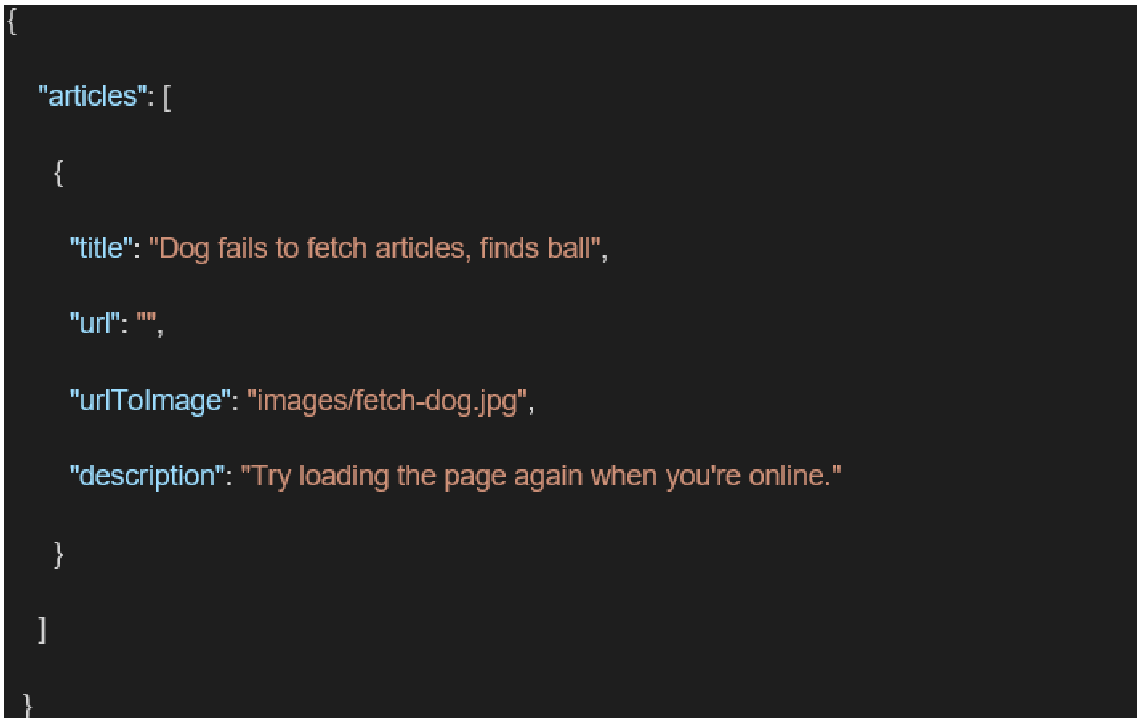
Firebase.json, .firebaserc, and folder .firebase were added automatically when the initialization process of Firebase Hosting was running.

.gitignore

It is a file which used to decide which files will be ignored, by the version control system.

Fallback.json

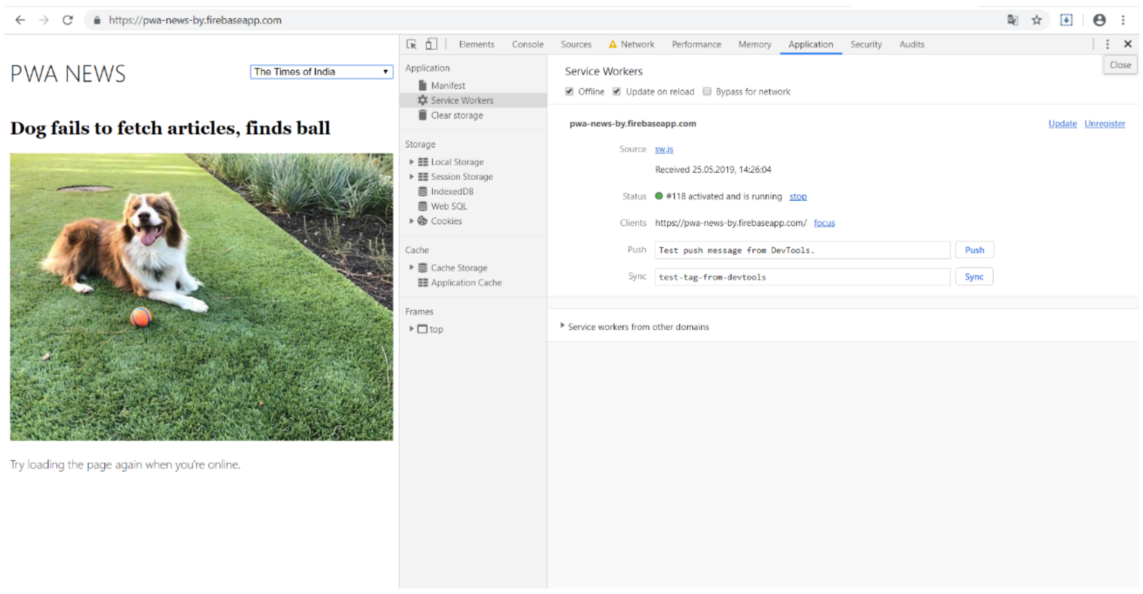
It is a file which used to display to the user some information when the internet connection is lost.



```json
{
  "articles": [
   {
     "title": "Dog fails to fetch articles, finds ball",
     "url": "",
     "urlToImage": "images/fetch-dog.jpg",
     "description": "Try loading the page again when you're online."
   }
  ]
 }
```

Figure 14. Fallback.json file of PWA news application.

From the user point of view, it looks the way it is shown in Figure 15 below.



PWA NEWS

The Times of India

Dog fails to fetch articles, finds ball

Try loading the page again when you're online.

Service Workers

✓ Offline ✓ Update on reload ☐ Bypass for network

pwa-news-by.firebaseapp.com

Source sw.js

Received 25.05.2019, 14:26:04

Status ● #118 activated and is running stop

Clients https://pwa-news-by.firebaseapp.com/ focus

Push `Test push message from DevTools.`

Sync `test-tag-from-devtools`

Service workers from other domains

Figure 15. Offline mode of the PWA News application.

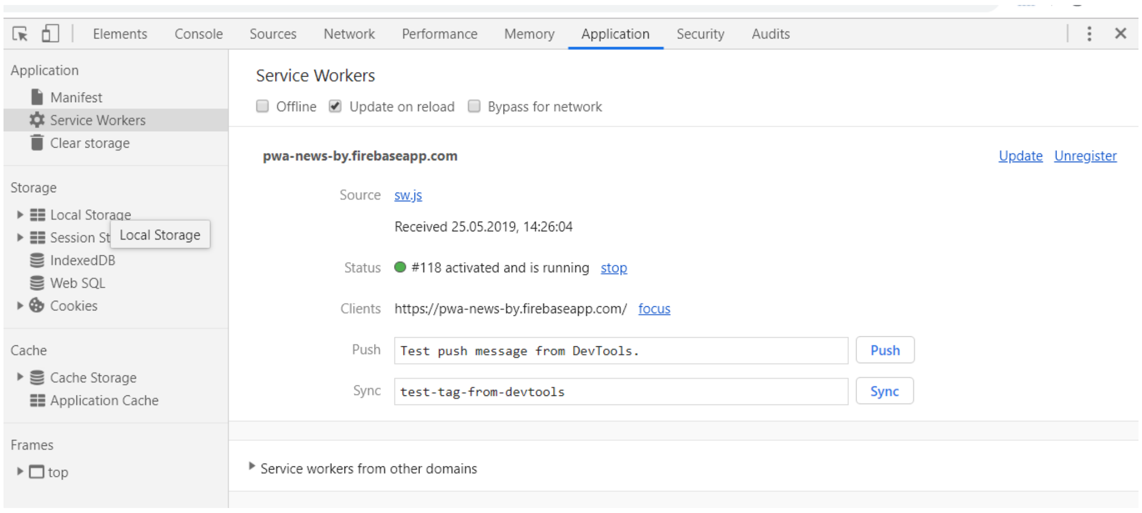
On the Figure 15 shown the offline mode. It appears when the user is offline and he changing the source of the news to the one he never was visited before, so it was not cached. It was used to get rid of the standard connection interrupted messages.

Manifest.json

This file was generated using Web App Manifest Generator (https://app-manifest.firebaseapp.com/). This tool is free to use and facilitates the work of the developer.

Sw.js

In this file was identified events for Service worker. Its functionality was checked using Chrome DevTools, as it is shown in Figure 16 below.



Service Workers

☐ Offline ☑ Update on reload ☐ Bypass for network

pwa-news-by.firebaseapp.com Update Unregister

Source sw.js

Received 25.05.2019, 14:26:04

Status ● #118 activated and is running stop

Clients https://pwa-news-by.firebaseapp.com/ focus

Push `Test push message from DevTools.` Push

Sync `test-tag-from-devtools` Sync

Service workers from other domains

Figure 16. Service worker status in Chrome Dev tools.

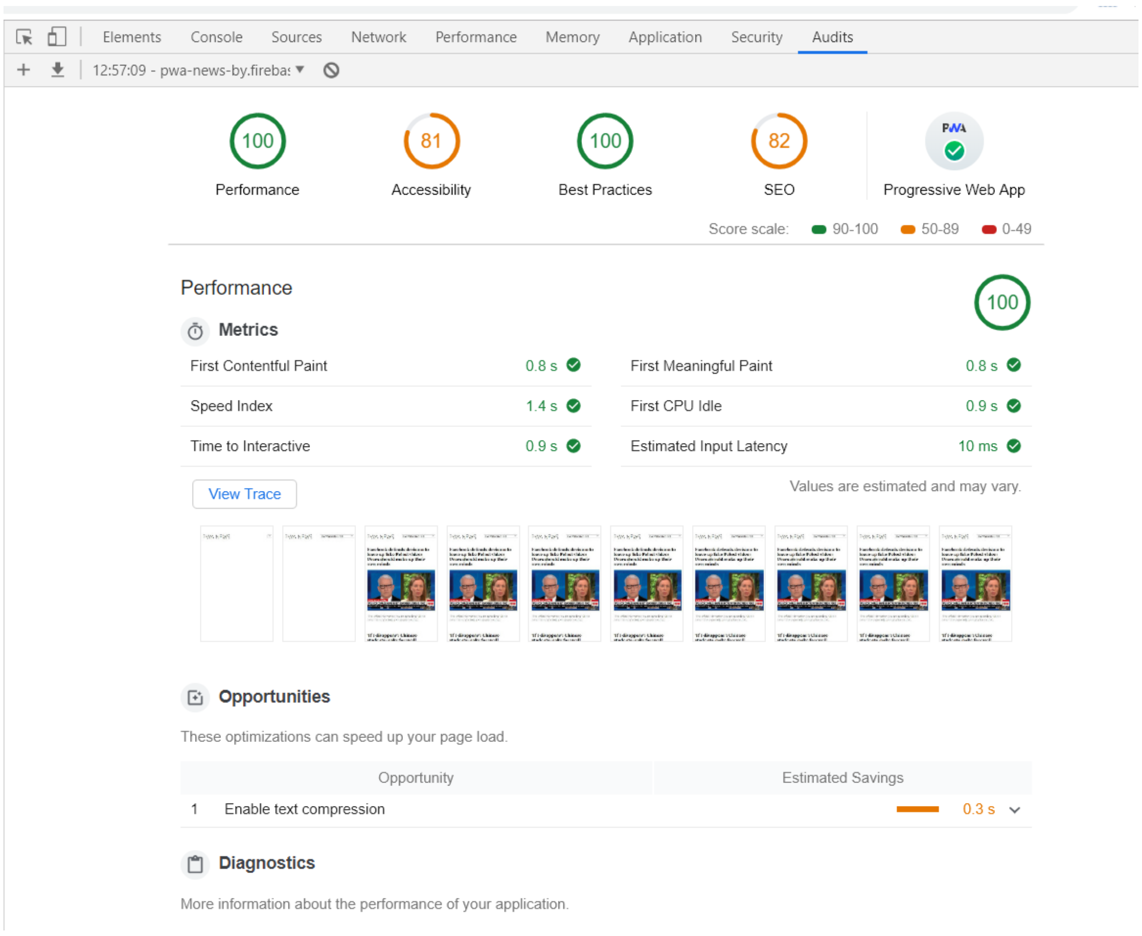
Service Worker was identified and registered and as it is shown in Figure 16 running without any errors.

5 RESULTS

After successful finishing the implementation process of PWA, when all debugging issues were solved and the app was hosted. The testing was carried out, for it was used LightHouse tool

Lighthouse is an open source tool which is used for the evaluation of websites. ‘’It can be run in Chrome DevTools, from the command line or as Node Module. It runs a series of audits against the page, and then it generates a report on how well the page did.’’ [13] This report gives an evaluation of the page in such categories as performance, accessibility, best practices, Search Engine Optimization (SEO) and PWA. It gives a great opportunity for the developer to see what is already good and what can be improved, moreover each audit having an explanation on why it is important and how to fix it. [13]

PWA News app successfully passed the Lighthouse audition process, as it is shown in Figure 17 below.



100 Performance | 81 Accessibility | 100 Best Practices | 82 SEO | Progressive Web App

Score scale: 🟩 90-100 🟧 50-89 🟥 0-49

Performance 100

Metrics

First Contentful Paint	0.8 s ✓	First Meaningful Paint	0.8 s ✓
Speed Index	1.4 s ✓	First CPU Idle	0.9 s ✓
Time to Interactive	0.9 s ✓	Estimated Input Latency	10 ms ✓

Values are estimated and may vary.

Opportunities

These optimizations can speed up your page load.

	Opportunity	Estimated Savings
1	Enable text compression	0.3 s

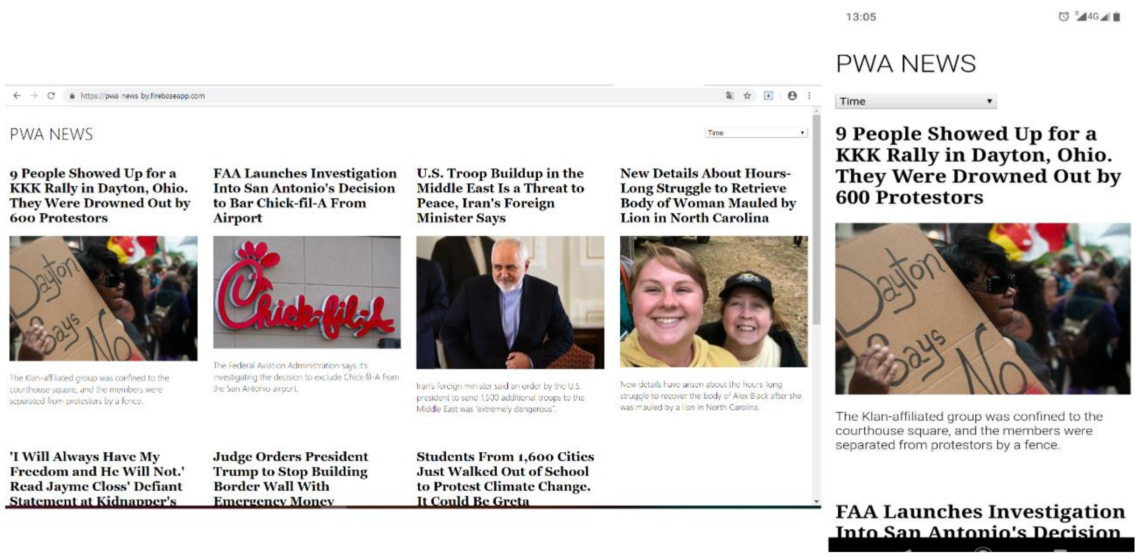
Diagnostics

More information about the performance of your application.

Figure 17. Lighthouse test results.

As it can be seen from Figure 17, the app received 100 points in a Performance category, 81 percent in accessibility, 100 in best practices, 82 in SEO and 100 percent for PWA, which means that completely fulfill all of the aspects of a Progressive Web App.

The final view of PWA news application is shown in Figure 18 below.



https://pwa-news-by.firebaseapp.com

PWA NEWS

Time

9 People Showed Up for a KKK Rally in Dayton, Ohio. They Were Drowned Out by 600 Protestors

The Klan-affiliated group was confined to the courthouse square, and the members were separated from protestors by a fence.

FAA Launches Investigation Into San Antonio's Decision to Bar Chick-fil-A From Airport

The Federal Aviation Administration says it's investigating the decision to exclude Chick-fil-A from the San Antonio airport.

U.S. Troop Buildup in the Middle East Is a Threat to Peace, Iran's Foreign Minister Says

Iran's foreign minister said an order by the U.S. president to send 1,500 additional troops to the Middle East was "extremely dangerous".

New Details About Hours-Long Struggle to Retrieve Body of Woman Mauled by Lion in North Carolina

New details have arisen about the hours-long struggle to recover the body of Alex Black after she was mauled by a lion in North Carolina.

'I Will Always Have My Freedom and He Will Not.' Read Jayme Closs' Defiant Statement at Kidnapper's

Judge Orders President Trump to Stop Building Border Wall With Emergency Money

Students From 1,600 Cities Just Walked Out of School to Protest Climate Change. It Could Be Greta

13:05

PWA NEWS

Time

9 People Showed Up for a KKK Rally in Dayton, Ohio. They Were Drowned Out by 600 Protestors

The Klan-affiliated group was confined to the courthouse square, and the members were separated from protestors by a fence.

FAA Launches Investigation Into San Antonio's Decision

Figure 18. Desktop and mobile app view of PWA News application.

As a result, we created a functioning PWA news app that can be added to the home screen, thanks to the web app manifest, and supports offline mode, thanks to the service worker and cache strategy.

6 CONCLUSION

In conclusion, the main goal of the thesis was to learn about PWA and create an app which would fulfill all of the aspects of PWA. While researching about PWA technologies, pros and cons were analysed, as well as a comparison between native, PWA and hybrid apps was made.

After studying the theory of PWA and having an understanding of PWA core technologies, the implementation of the app itself started. A News app was decided to be a good choice, due to simple app logic itself and the opportunity to use all of PWA technologies. The developed app is fully consistent with the initial idea, and the test result confirms it.

PWA seems to be a good set of technologies that moving web to a new level, and helping the web to increase the engagement with the users. It could replace some of the native apps and will be beneficial for all of the sides, but it does not look like something that is going to replace them completely, but it will occupy its niche.

Challenge in gaining the popularity for PWA is still remaining some lack of support on IOS and non-acquaintance of end users of the existence of those features. Anyway, it is noticeable how the developers are trying to solve these problems, releasing new updates and optimizing technologies for their greater approach.

In any case, Google is actively promoting PWA and prompting to master the PWA technologies, by providing tutorials in particular. Moreover, the fact that such giant companies as Facebook, Twitter, Instagram and many others have, besides their main native apps, PWA versions of it, means only that PWA is a great technology, which will gain popularity.

To sum up, from the author’s perspective, it was interesting to research and implement PWA and this knowledge has strengthened the author’s overall web development experience.

REFERENCES

[1] Comscore, Inc. (2019). *Comscore's 2015 U.S. Mobile App Report Available for Download*. [online] Available at: https://www.comscore.com/Insights/Press-Releases/2015/9/comScores-2015-US-Mobile-App-Report-Available-for-Download [Accessed 9 May 2019].

[2] Vaadin.com. (2019). [online] Available at: https://vaadin.com/pwa?utm_term=pwa&utm_campaign=PWA+search+EU+-+Mar+20,+2018&utm_source=adwords&utm_medium=ppc&hsa_net=adwords&hsa_tgt=kwd-303145221062&hsa_ad=258324572117&hsa_acc=7040932438&hsa_grp=55768637720&hsa_mt=e&hsa_cam=1258602860&hsa_kw=pwa&hsa_ver=3&hsa_src=g&gclid=Cj0KCQjwh6XmBRDRARIsAKNInDEzvhXLrjTOTQg5bt9deU1TizJhE0l75RZJWIEiZi2E95ARja8186kaApbwEALw_wcB [Accessed 28 Mar. 2019].

[3] Weekly-geekly.github.io. (2019). PWA is just. [online] Available at: https://weekly-geekly.github.io/articles/418923/index.html [Accessed 2 Aprill 2019].

[4] Telerik Blogs. (2019). What is a Hybrid Mobile App?. [online] Available at: https://www.telerik.com/blogs/what-is-a-hybrid-mobile-app- [Accessed 10 Apr. 2019].

[5] Google Developers. (2019). Service Workers: an Introduction | Web Fundamentals | Google Developers. [online] Available at: https://developers.google.com/web/fundamentals/primers/service-workers/ [Accessed 25 Apr. 2019].

[6] Google Developers. (2019). The Web App Manifest | Web Fundamentals | Google Developers. [online] Available at: https://developers.google.com/web/fundamentals/web-app-manifest/ [Accessed 26 Feb. 2019].

[7] Love2dev.com. (2019). [online] Available at: https://love2dev.com/blog/beforeinstallprompt/ [Accessed 10 May 2019].

[8] Google Developers. (2019). The App Shell Model | Web Fundamentals | Google Developers. [online] Available at: https://developers.google.com/web/fundamentals/architecture/app-shell [Accessed 28 Apr. 2019].

[9] Google Developers. (2019). Instant Loading Web Apps with an Application Shell Architecture | Web | Google Developers. [online] Available at: https://developers.google.com/web/updates/2015/11/app-shell [Accessed 5 May 2019].

[10] Google Developers. (2019). Introduction to Push Notifications | Web | Google Developers. [online] Available at: https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications [Accessed 11 Apr. 2019].

[11] Caniuse.com. (2019). Can I use... Support tables for HTML5, CSS3, etc. [online] Available at: https://caniuse.com/#feat=push-api [Accessed 28 Apr. 2019].

[12] Code, (2019). Visual Studio Code - Code Editing. Redefined. [online] Code.visualstudio.com. Available at: https://code.visualstudio.com/ [Accessed 14 Apr. 2019].

[13] Google Developers. (2019). Lighthouse | Tools for Web Developers | Google Developers. [online] Available at: https://developers.google.com/web/tools/lighthouse/ [Accessed 28 Apr. 2019].