Progressive web app

https://dev.to/paco\_ita/service-workers-and-caching-strategies-explained-step-3-m4f

https://www.freecodecamp.org/news/build-a-pwa-from-scratch-with-html-css-and-javascript/

Service worker:

A SW is similar to a web worker, both a simple javascript file.  
A web worker does not have a specific task and it is typically used to offload the main thread (where the main web app is running on).

On the other side, a service worker has a specific task and it is to act as a proxy between our web application and the network. It can intercept http requests and serve the responses from the network or from a local cache, according to which caching strategy we implemented (more details later).

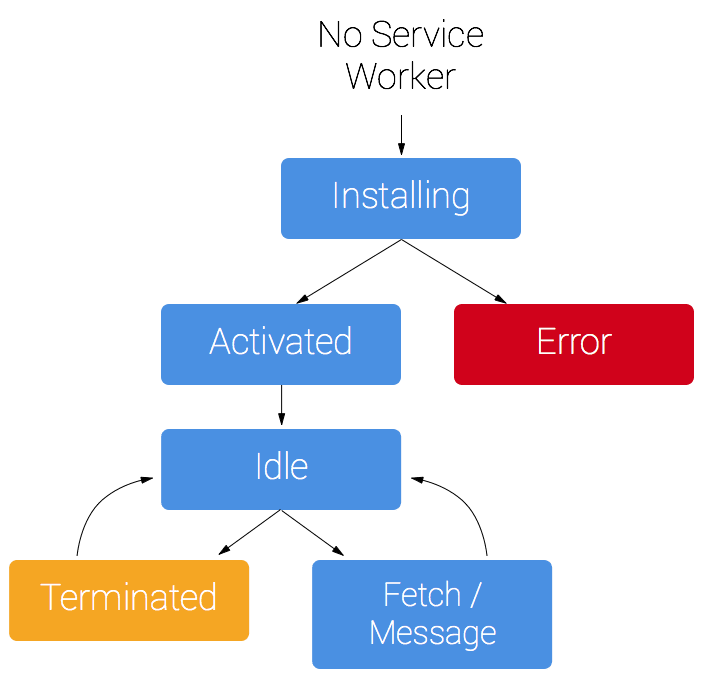
**SW are Secure**

Service workers will **function only on HTTPs connection**.

This is a safe decision, because otherwise we would easily expose our application to man-in-the-middle attacks. Let's just imagine what might happen if anybody could substitute our SW with a manipulated one...scary, isn't it?

On the other side, localhost is considered secure, allowing to test the application before deploying it.  
If we work with Angular though, we cannot use the ng serve command to build and serve our application locally, as it does not work with service workers. In this case we have to use an HTTP Server of our choice, for example [http-server](https://www.npmjs.com/package/http-server) package or the [Web Server](https://chrome.google.com/webstore/detail/web-server-for-chrome/ofhbbkphhbklhfoeikjpcbhemlocgigb?hl=en) Chrome extension.

Service Workers lifecycle



The schema above describes the different lifecycle steps of a service worker.  
During the registration the whole operation is canceled if an error occurs or the SW file cannot be fetched.  
The register method will be newly triggered when the user loads the page again. The browser is able to identify whether the SW is already installed or not and call the method accordingly.  
  
Once registered, a SW does not remain constantly active. The browser can unpredictably terminate it and reactivate it again when an event needs to be triggered. That's the reason why, if we need to persist a state used within the service worker (I do not mean caching assets or API requests here), we should better use IndexeDB, or a similar solution.  
   
In the install step, pre-fecth operations are typically executed. Their goal is to ensure target assets are downloaded and made already available in the cache for the SW. These assets are commonly static files (eg. js, css) representing the core shell of our application, the minimum files and styles that should be available immediately to the user, even when offline.

https://danielk.tech/home/angular-app-shell-ultimate-guide

https://blog.angular-university.io/angular-app-shell/

ng new my-app-shell --routing

ng build --prod

ng serve --prod

ng generate universal

ng generate app-shell --route=app-shell-path

ng run AppShellDemo:app-shell:production

ng run app-shell-test:app-shell:production

npm install --save @angular/cdk @angular/material ngx-spinner

npm install cors

const cors = require('cors');

const express = require('express');

const cors = require('cors');

const app = express();

// Allow requests from all origins

app.use(cors());

// Or, specify allowed origins explicitly

app.use(

cors({

origin: ['http://localhost:4200', 'https://example.com'],

})

);

// Your other route handlers here

// ...

npm install web-push -g

web-push generate-vapid-keys

ng g c components/notification

cd server

npm i --save web-push

ng build --prod  
http-server -p4200 -c-1 dist/<name-of-app>node server/server.js

https://arjenbrandenburgh.medium.com/angulars-pwa-swpush-and-swupdate-15a7e5c154ac

https://dev.to/paco\_ita/service-workers-and-caching-strategies-explained-step-3-m4f

https://github.com/anuroopjoy/pwa-sample/blob/master/server/src/main.ts

https://www.freecodecamp.org/news/build-a-pwa-from-scratch-with-html-css-and-javascript/