Deployment and Hosting

Next, we have to deploy our PWA to a server in order to make it available to the public and be sure that it works in a remote environment.

We'll cover the following

Nowadays, there are plenty of hosting possibilities to choose from, and all are almost equivalent in terms of performance and services offered.

Therefore, we are free to choose the provider we prefer, but in this lesson, we will use the Firebase platform to host our application. Since we used Cloud Firestore for its offline persistence capabilities in a previous demo, using Firebase hosting is a natural choice.
This is not a rule, however. We can still use Cloud Firestore as NoSQL DB but host our application in Azure or Netlify, for instance. The decision is ours.

Firebase hosting#

Firebase hosting is a web content hosting offered by the Firebase platform. It has many interesting advantages and, thanks to its CLI, setting up and deploying projects are extremely quick and easy.

Firebase hosting provides great features, not only for static web content, but also for more complex and dynamic apps:

- **Secure**: SSL is provided for the hosted data out of the box. We do not have to purchase a certificate. Our web app will be delivered through HTTPs protocol, according to Google's goal to have an overall *secure web*.

- **Fast**: The deployed files are cached on SSDs at CDN edges in different locations. This means that the closest CDN location will deliver the data to our clients, ensuring that our content is delivered fast.

- **Easy to deploy**: Thanks to the Firebase CLI, we can simply setup our local project with basic Firebase parameters and deploy it to the cloud with one command. Moreover, through the Firebase console, we can get a set of important details about the downloads or the in-/outbound traffic from our application.
  If we find an error in the latest release, we can also rollout a previous version with a mouse click.

Setup Firebase#

Below we describe all the needed packages and commands to deploy your application to Firebase step by step.

Create a Firebase project#

First of all, we need to create a new Firebase project, which can be seen as a root container for web or mobile (iOS/Android) applications.
Once we create a project, we can choose to insert all the services that are compatible with our application. Considering our PWA, we might be interested in adding Google Analytics, Cloud Firestore, and eventually performance monitoring. Toggling features are very easy within the Firebase platform, as Firebase provides several SDKs that can be adapted according to our requirements.

Accessing the Firebase console, we can login with a Google account and select "Add new project":

We need to provide a project name. I chose "Educative PWA." The name must be unique. In case a project with the same name already exists in Firebase, an alternative one will be suggested.

At this stage, we can edit the `Project ID`, which is reported just under the input field where we entered the project name.

If we are satisfied with the given name, and it is unique, we don't have to change anything, but keep in mind that we cannot change this parameter once the project is created.

We can opt to enable analytics or not. Analytics are a prerequisite for other services. Therefore, we might prefer to have analytics enabled if we plan to use features like Crashlytics or A/B Testing.

That's it! Our Firebase project will be created for us. After the project is ready, we can explore its dashboard, where we can have full control over settings, billing, and available features:

Add our app to the Firebase project#

We can select to register a mobile or web app from the dashboard.

In our case, we will choose the web option and give a nickname for our app, here "my-demo-pwa".
The nickname is not publicly visible, and it is only used within Firebase to identify the project.

At this point, Firebase provides us configuration keys for the project we just created. These keys will be used to bind our PWA to different services, like Cloud Firestore, for instance.

We need these keys in our project, and we must keep them secret. Considering the Angular demo we wrote in the previous lessons, we can paste these keys in the files in the environments folder:

```
export const environment = {
    production: false,

  // Your web app's Firebase configuration

  firebaseConfig = {
    apiKey: "SyAK3jHFQdpF7xS42AxJ7Gh2wSBChs2svvcJ",
    authDomain: "educative-pwa.firebaseapp.com",
    databaseURL: "https://educative-pwa.firebaseio.com",
    projectId: "educative-pwa",
    storageBucket: "educative-pwa.appspot.com",
    messagingSenderId: "3364045128815",
    appId: "1:3364045128815:web:h37d2810a4bdr99j14e251"
  };
};
```

environment.ts

This way it is easier to access them within the web application. To connect our Angular app to Firebase, we need the `firebase` and `@angular/fire` npm packages.
We can install them in one single command: `npm i --save firebase @angular/fire`

AngularFire is the official Angular library for Firebase. Within our PWA project, we can register the installed packages in the `app.module.ts` file.

```
import { environment } from "src/environments/environment";
import { AngularFireModule } from "@angular/fire";
import { AngularFirestoreModule } from "@angular/fire/firestore";

@NgModule({
    declarations: [AppComponent],
    imports: [
    ...
    AngularFireModule.initializeApp(environment.firebaseConfig),
    AngularFirestoreModule
   ]
})

```

> ⚠️ **Note!** We pass the Firebase project configuration object as parameter of the `initializeApp()` method, binding in this way our PWA with the Firebase project.

If we don't use Angular, the generic Firebase initialization is very similar.

```
import * as firebase from "firebase/app";

const firebaseConfig = {
    apiKey: "SyAK3jHFQdpF7xS42AxJ7Gh2wSBChs2svvcJ",
    authDomain: "educative-pwa.firebaseapp.com",
    databaseURL: "https://educative-pwa.firebaseio.com",
    projectId: "educative-pwa",
    storageBucket: "educative-pwa.appspot.com",
    messagingSenderId: "3364045128815",
    appId: "1:3364045128815:web:h37d2810a4bdr99j14e251"
  };

// Initialize Firebase
firebase.initializeApp(firebaseConfig);
```

Install Firebase CLI#

Among other tasks, like setting up a local project, Firebase CLI is the easiest way to deploy our code to Firebase.

We can install it from npm with the following command:

```
npm install -g firebase-tools
```

Once installed, let's login into our Firebase project from the terminal with the command `firebase login`, providing the firebase credentials. This allows you to make Firebase projects visible within the Firebase Tools scope.

Init Firebase locally#

At this point, in order to initialize our PWA project, we need to run the `firebase init` instruction.
This is needed to connect our local project to the one previously created in the Firebase platform.

> ⚠️ **Note!** Be sure to run the `firebase init` from the root of your app.

During this phase, we are prompted to choose among different options:

- We will select `Hosting` (We can add further functionalities, like Cloud Functions, later.).

- Since we already created a project in the Firebase console, we choose the `Use an existing project` option and then select the right project.

- We have to provide the directory to use as a public root directory. Considering our Angular demo, it will be the `/dist` folder or any folder we choose as the target for the production build. This folder will contain the compiled bundles and static assets that we want to deploy to Firebase.

- We, then, need to choose a configuration for our site. In our case, we will select a one page app.

Once the initialization process is terminated, Firebase creates two files:

- `firebase.json` configuration file
- `.firebaserc` file that stores the project aliases

Deploy the PWA to Firebase#

All these configurations might sound a bit tedious and long, but we need to implement them only once and all its steps are pretty straightforward.

Once everything is set up, we can run the `firebase deploy` command in the terminal to deploy our PWA to the Firebase hosting.

Once we receive a confirmation message, our application will be live at the following addresses:

educative-pwa.web.app
educative-pwa.firebaseapp.com

`educative-pwa` is the project ID that we defined while creating the Firebase project.
Clearly, we can also connect our custom domain name to a Firebase hosted site.

From this moment, any time we implement new features in our PWA, we simply need to run the `firebase deploy` command to send our latest release to Firebase and have it live.