

Universidad Tecnológica del Centro  
**ORGANISMO PÚBLICO DESCENTRALIZADO DEL  
GOBIERNO DEL ESTADO DE YUCATÁN**

**CARRERA:**

INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE

**ASIGNATURA:**

APLICACIONES WEB PREGRESIVAS

**CUATRIMESTRE:** 10°

**GRUPO:** A

**ELABORADO POR:**

BR. CAN OXTE LICSI ASUCENA (19211955)

BR. IRABIEN CRIOLLO RAFAEL ANTONIO (19211986)

BR. MEDINA PISTE DANIEL CONCEPCIÓN (19211967)

BR. JOSIAS EMMANUEL FRANCO PUERTO (19211982)

BR. ANGEL LUIS CANCHE CHAN (19211956)

**MAESTRO:**

MTRO. IVÁN VEGA UC

**IZAMAL, YUCATÁN**

**NOVIEMBRE 2022**

## Contenido

Plataformas y herramientas utilizadas:.....	3
Visual estudio.....	3
Para el desarrollo del código .....	3
Chrome y libre Wolf.....	3
Angular.....	3
Node js.....	4
Parámetros de configuración de las herramientas .....	4
Reporte .....	5
Configuración de la aplicación y la forma de configurar.....	5
Splash screen .....	5
Código.....	8
Instalación de service worker .....	8
Asignación de la ruta.....	8
Registrar el service worker.....	9
Política de almacenamiento en caché de rendimiento .....	9
Angular app Shell.....	10
Instalación.....	10
Instalar los módulos necesarios.....	11
Clases generadas con el comando.....	11
Comandos para iniciar el Shell en la aplicación.....	12
Notificaciones.....	13
Instalación del servicio de mensajes push.....	13
Configuración de las rutas .....	14
Uso dispositivo físico del teléfono .....	15
Estado offline de la aplicación.....	16
Link del proyecto .....	16

## **Plataformas y herramientas utilizadas:**

Se ha utilizado una serie de herramientas cuyas funciones agregaran modalidades útiles y otorgaron facilidades al momento de desarrollar del proyecto,

### **Visual estudio**

Se opto por esta herramienta por el grado de familiaridad y conocimiento de esta que poseen los desarrolladores que conforman el equipo de trabajo, esta misma resulta cómoda al momento de usarse, así como facilitar tareas que de otro modo o con otra herramienta serían más tediosas de realizar

### **Para el desarrollo del código**

#### **Chrome y libre Wolf**

Contando con estos dos navegadores para verificar los avances al momento de la codificación del proyecto para probar cada actualización realizando los cambios y actualizaciones requeridas, al utilizar siempre el navegador de Chrome presentaba fallos que no parecían ser errores de la programación si se usaba libre Wolf que no cache la página para comprobar el funcionamiento así manteníamos una visión más abierta del progreso de la aplicación desarrollada

#### **Angular**

Angular es una plataforma de desarrollo, construida sobre TypeScript. Es un framework basado en componentes para crear aplicaciones web escalables. Una colección de bibliotecas bien integradas que cubren una amplia variedad de características, que incluyen enrutamiento, administración de formularios, comunicación cliente-servidor y más. Un conjunto de herramientas para desarrolladores que permiten desarrollar, compilar, probar y actualizar el código fuente de la aplicación las ventajas que esta permite para las desarrolladoras y la facilidad de conversión de sus aplicaciones a progresivas de una manera más sencilla que con otros lenguajes nos orillo a enfocarnos en ella para el desarrollo de este proyecto.

## Node js

Node. js sirve para crear sitios web dinámicos muy eficientes, escritos con el lenguaje de programación JavaScript. Normalmente, los desarrolladores se decantan por este entorno de ejecución cuando buscan que los procesos se ejecuten de forma ágil y sin ningún tipo de bloqueo cuando las conexiones se multiplican, la razón por la que se optó por esta herramienta es ya que permite el manejo de paquetes de javascript de una manera más sencilla y esto fue muy valioso para el desarrollo del proyecto.

### Parámetros de configuración de las herramientas

Herramienta	Utilidad	Versión configuraciones o
Visual studio.	Se uso una extensión de typescript para ayudar con la sintaxis de la programación lo que ayudo a prevenir errores gramaticales.	Extensión de typescript.
Node.	Se usó la versión 12 de node js para garantizar compatibilidad con el curso que se estaba siguiendo.	Versión 12 de node js.
Angular CLI.	Se usó la versión 9.0.2 equivalente a la versión del angular lo garantiza compatibilidad con los comandos.	La versión 9.0.2

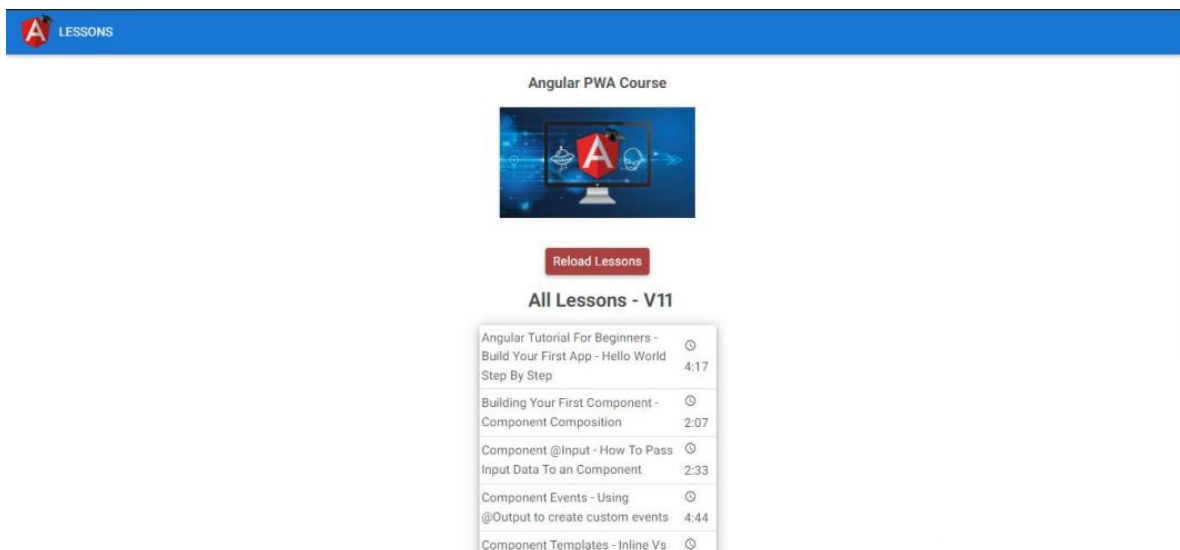
# Reporte

## Configuración de la aplicación y la forma de configurar

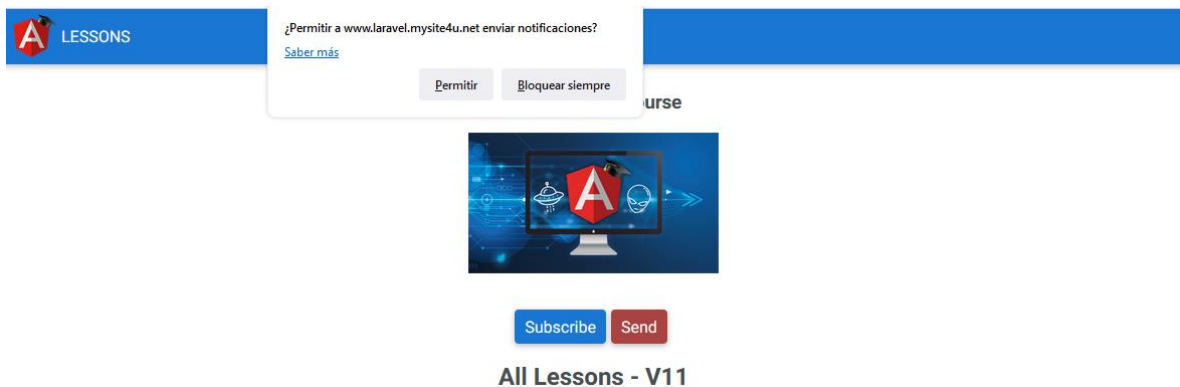
### Splash screen



Se muestra en esta imagen la Pantalla de inicio que los usuarios verán para poder iniciar con los servicios de la aplicación.



El reporte de Notificaciones permite avisar y notificar a los usuarios de un cambio, reporte o noticia, que se haya presentado en la página.



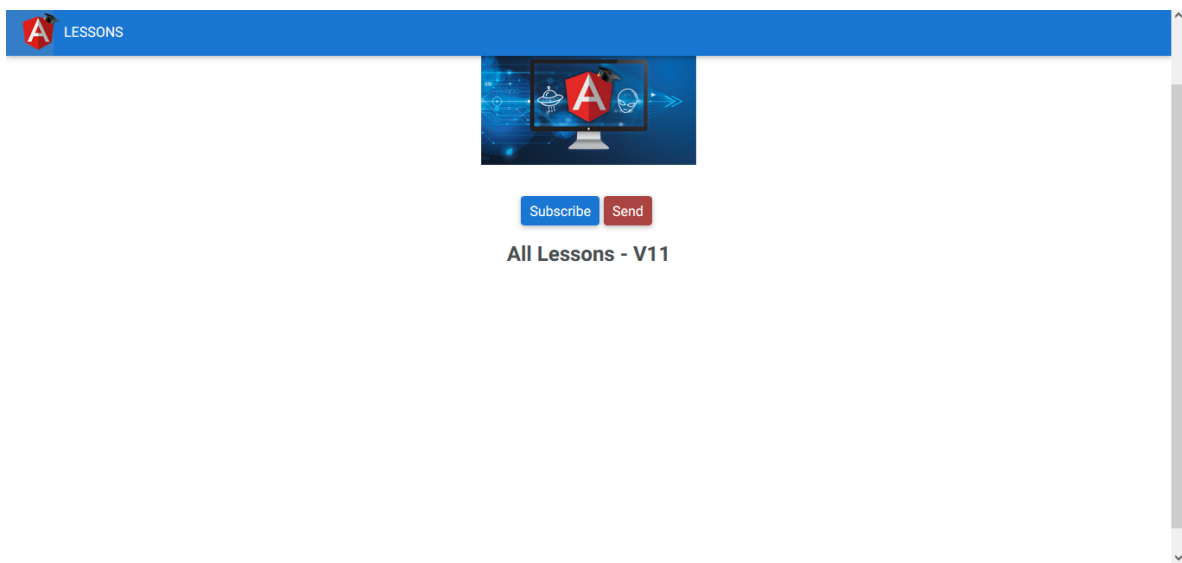
Al tener una estructura y dimensión diferentes la Vista en dispositivo móvil cambia ajustándose a sus propias características.



En la siguiente imagen se contempla los Datos del servidor con la que se comunica y almacena la página estos comparten la información requerida para el funcionamiento de la página.

```
localhost:3000/api/lessons
{
  lessons: [
    {
      id: 1,
      description: "Angular Tutorial For Beginners - Build Your First App - Hello World Step By Step",
      longDescription: "<p>This is step by step guide to create your first application. <b>Its aimed at beginners</b> just starting out with the framework.This lesson will show how",
      tags: "BEGINNER",
      duration: "4:17",
      url: "https://www.youtube.com/watch?v=iVrF-a06Nv0",
      videoUrl: "https://www.youtube.com/embed/du6skvEFrhQ"
    },
    {
      id: 2,
      description: "Building Your First Component - Component Composition",
      duration: "2:07",
      longDescription: "<p>In this lesson we are going to see how to include a component inside another component. We are going to create a simple search box component and include it",
      tags: "BEGINNER",
      url: "angular2-build-your-first-component",
      videoUrl: "https://www.youtube.com/embed/VF51eTNx1s"
    },
    {
      id: 3,
      description: "Component @Input - How To Pass Input Data To an Component",
      duration: "2:33",
      longDescription: "<p>In this lesson we are going to learn how to use the template syntax for properties, and learn how we can use it to pass input data to a component. We are",
      tags: "BEGINNER",
      url: "angular2-passing-data-to-component-using-input",
      videoUrl: "https://www.youtube.com/embed/Yfeb2mFrTU"
    },
    {
      id: 4,
      description: "Component Events - Using @Output to create custom events",
      duration: "4:44",
      longDescription: "<p>In this lesson we are going to see how components can emit custom events via EventEmitter and the @Output decorator. We are going to see how we can subscri",
      tags: "BEGINNER",
      url: "angular2-component-events",
    }
  ]
}
```

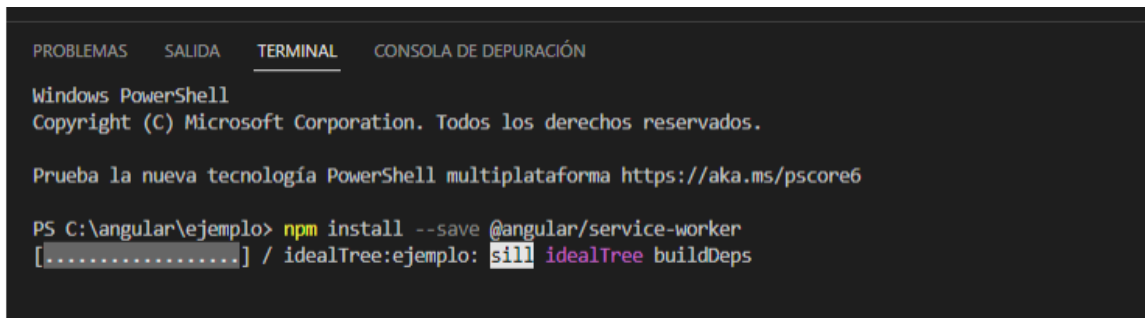
Modo offline sin la cache, algunos datos son guardados de manera local, lo que le permite a la página mostrar está sección aún sin tener conexión con el servidor.



## Código

### Instalación de service worker

Service worker es una librería de angular que permite la migración de una aplicación común a una aplicación progresiva, para su instalación basta con introducir en la consola el comando “**npm install --save @angular/service worker**”.



```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

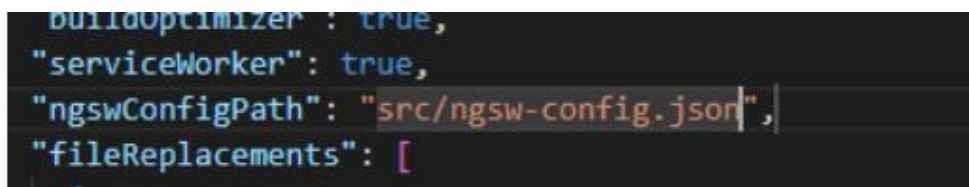
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/powershell

PS C:\angular\ejemplo> npm install --save @angular/service-worker
[.....] / idealTree:ejemplo: sill idealTree buildDeps
```

El administrador de paquetes de node js se encarga de realizar la instalación de los archivos que van ser utilizados para ofrecer el servicio dentro de este framework. Adicionalmente es necesario el realizar una configuración dentro de los archivos específicamente en la archivo que se encuentra en el directorio raíz denominado “angular.json”. Aquí se agrega las líneas “serviceWorker” : true.

### Asignación de la ruta

Ahora que service worker ya se encuentra instalado es necesario definir la ruta en donde se va encontrar su archivo de configuración para ello se abre el archivo angular.json de nuevo y se agrega la línea “ngswConfigPath”: “src/ngsw-config.json”, esto hace referencia a la ubicación en donde se encuentra el archivo de configuración de serviceworker.



```
    "buildOptimizer": true,
    "serviceWorker": true,
    "ngswConfigPath": "src/ngsw-config.json",
    "fileReplacements": [
```



## Registrar el service worker

Como último paso de la instalación se puede proseguir con el registro del componente instalador por ello en el archivo de configuración de angular este se encuentra dentro de “src>app>app.module.ts”, aquí se agrega como uno de los módulos dentro de la sección imports.

```
@NgModule({
  declarations: [
    AppComponent,
    LessonsComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    BrowserModuleAnimationsModule,
    AppRoutingModule,
    ReactiveFormsModule,
    ServiceWorkerModule.register('/ngsw-worker.js', { enabled: environment.production })
  ],
  providers: [
    LessonsService
  ],
  bootstrap: [AppComponent]
})
```

Despues de realizar la configuración anterior la aplicación de angular ya se encuentra convertida en una aplicación web progresiva. Únicamente falta compilarla para que pueda procesar los cambios que se han hecho.

## Política de almacenamiento en caché de rendimiento

Una de las cualidades de las aplicaciones web es el poder utilizar los datos que trae desde el servidor de forma fuera de línea, por ello se debe de realizar una configuración dentro del archivo de la configuración del service worker. Aquí se debe de agregar un nuevo arreglo con la configuración que en este caso es la dirección del api y como se va almacenar la configuración, este caso haciendo uso de los datos que obtenga de la primera visita de forma temporal, lo que limita las peticiones al servidor.

```

    "DataGroups": [
      {
        "name": "lessson-api",
        "urls": [
          "/api/lessson"
        ],
      },
    ],
  },

```

```

    "cacheConfig": {
      "strategy": "perfomance",
      "maxAge": "1d",
      "maxSize": 100
    }
  }
}

```

Ahora ya se puede compilar de nuevo la app, solo cambiando la version para ver como los cambios son aplicados.

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2193]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\angular\ejemplo>npm run server

> angular-pwa-course@0.0.0 server
> ts-node -P ./server/server.tsconfig.json ./server/server.ts

HTTP Server running at http://localhost:9000
¿Desea terminar el trabajo por lotes (S/N)? s

C:\angular\ejemplo>

```

## Angular app Shell

Este permite la carga de un loader dentro de la aplicación cuando este se encuentra en el proceso de carga de la aplicación, pero es necesario una pequeña configuración del servicio.

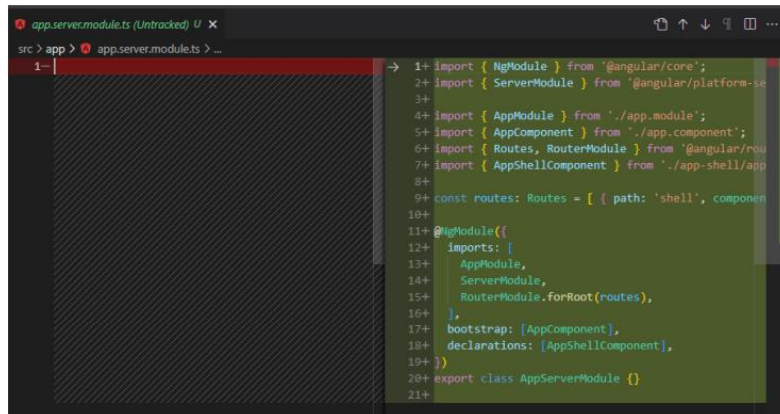
### Instalación

Para comenzar con la configuración se debe de realizar una instalación del paquete usando el terminal de angular con el siguiente comando **“ng generate --client-project angular-pwa-course --universal-project serve app”**.

```
PS C:\angular\ejemplo> ng generate app-shell --client-project angular-pwa-course --universal-project server-app
DEPRECATED: The 'defaultProject' workspace option has been deprecated. The project to use will be determined from the current working directory.
Your global Angular CLI version (14.2.1) is greater than your local version (9.0.2). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".
Option "universalProject" is deprecated: This option has no effect.
```

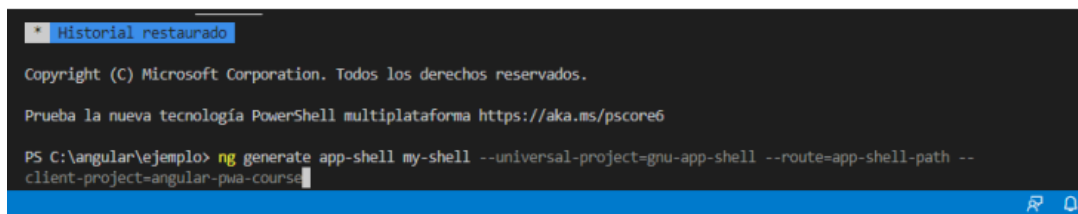
EL comando anterior generara una serie de clases y componentes necesarios para la instalación del Shell



```
app.server.module.ts (Untracked) U X
src > app > app.server.module.ts > ...
1-
→ 1+ import { NgModule } from '@angular/core';
2+ import { ServerModule } from '@angular/platform-server';
3+
4+ import { AppModule } from './app.module';
5+ import { AppComponent } from './app.component';
6+ import { Routes, RouterModule } from '@angular/router';
7+ import { AppShellComponent } from './app-shell/app-shell.component';
8+
9+ const routes: Routes = [{ path: 'shell', component: AppShellComponent }];
10+
11+ @NgModule({
12+   imports: [
13+     AppModule,
14+     ServerModule,
15+     RouterModule.forRoot(routes),
16+   ],
17+   bootstrap: [AppComponent],
18+   declarations: [AppShellComponent],
19+ })
20+ export class AppServerModule {}
21+
```

## Instalar los módulos necesarios

Ya que el servicio se encuentre listo es necesario realizar la instalación del Shell con unas configuraciones adicionales dentro del proyecto, esto se realiza desde la terminal de la aplicación por medio de un comando.



```
Historial restaurado
Copyright (C) Microsoft Corporation. Todos los derechos reservados.
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6
PS C:\angular\ejemplo> ng generate app-shell my-shell --universal-project=gnu-app-shell --route=app-shell-path --client-project=angular-pwa-course
```

## Clases generadas con el comando

Despues de que el comando ya este terminado se puede apreciar que genero unos archivos de typescript y un archivo html que contiene cierta información del enrutamiento en la aplicación, estos archivos son los que tienen la conexión con el Shell de la aplicación.

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "project": {
    "name": "angular-security-course"
  },
  "apps": [
    {
      "root": "src",
      "outDir": "dist",
      "assets": [
        "assets",
        "favicon.ico",
        "manifest.json"
      ],
      "index": "index.html",
      "main": "main.ts",
      "polyfills": "polyfills.ts",
      "test": "test.ts",
      "tsconfig": "tsconfig.app.json",
      "testTsconfig": "tsconfig.spec.json",
      "prefix": ""
    }
  ]
}
```

Módulos en donde se modificará la salida del componente

```
import { NgModule } from '@angular/core';
import { ServerModule } from '@angular/platform-server';

import { AppModule } from './app.module';
import { AppComponent } from './app.component';
import { Routes, RouterModule } from '@angular/router';
import { AppShellComponent } from './app-shell/app-shell.component';

const routes: Routes = [ { path: 'app-shell-path', component: AppShellComponent } ];

@NgModule({
  imports: [
    AppModule,
    ServerModule,
    RouterModule.forRoot(routes),
  ],
  bootstrap: [AppComponent],
  declarations: [AppShellComponent],
})
```

## Comandos para iniciar el Shell en la aplicación

Ahora que la configuración ya se encuentra lista se procede a la compilación de los archivos desde la terminal de angular empleando la Shell de la aplicación.

```
Run `npm audit` for details.
PS C:\angular\ejemplo> ng run angular-pwa-course:app-shell
```

Lín. 42, col. 49    Espacios: 2    UTF-8    LF

## Notificaciones

### Instalación del servicio de mensajes push

Para la instalación del servicio de notificaciones se prosiguió con la instalación de un paquete npm llamado web-push, el cual su función es suscribirse a un canal de eventos y estar a la escucha para realizar alguna actividad, este paquete requiere de una llave publica y una llave privada para funcionar los cuales pueden ser generados desde la consola npm como se puede observar en las instrucciones del archivo.

#### ## Generating VAPID keys

In order to generate a public/private VAPID key pair, we first need to install the [web-push] (<https://github.com/web-push-libs/web-push>) library globally:

```
npm install web-push -g
```

We can then generate a VAPID key pair with this command:

```
web-push generate-vapid-keys --json
```

And here is a sample output of this command:

```
```json
{
  "publicKey":
    "BF1BhDhSW89yKw6pWbLlzcDpCR3I3ViSCEiS_z0q_RP9-ablo5Up8HDIEP1-GauARtU7MxB6Yl_7FI8UvczPm
    aQ",
  "privateKey": "6XaIXj1cbSoaCpxSbOA-xYwHSISVSMCPucSvEcczxkg"
}
```
```

## Configuración de las rutas

Dentro de la configuración del servidor se tiene que asignar unas rutas del tipo post que permite lograr la conectividad entre los usuarios, estas se encontraran con la api lo que permite lograr un mayor control en la app al separar los componentes

```
app.use(bodyParser.json());

// REST API
app.route('/api/lessons')
  .get(readAllLessons);

app.route('/api/notifications')
  .post(addPushSubscriber);

app.route('/api/newsletter')
  .post(sendNewsletter);
```

Ya por último se enlaza con la vista, las rutas correspondientes conectadas por medio de un botón que indique la acción a realizar.

```
@Injectable()
export class NewsletterService {

  constructor(private http: HttpClient) {

  }

  addPushSubscriber(sub:any) {
    return this.http.post('/api/notifications', sub);
  }

  send() {
    return this.http.post('/api/newsletter', null);
  }
}
```

## Uso dispositivo físico del teléfono

La forma en la que la app hace uso del dispositivo físico con el que cuenta un dispositivo electrónico es accediendo a su sistema de vibración si es que el dispositivo cuenta con él, específicamente al momento de realizar una notificación es cuando la app enciende ese sensor. Esto se logra por medio de una promesa que el javascript que a su vez depende de un payload (carga util), dentro de este se componen ciertos componentes que permiten generar la notificación que se recibe y en uno de ellos se especifica la duración de la vibración del dispositivo

```
// sample notification payload
const notificationPayload = {
  "notification": {
    "title": "Angular News",
    "body": "Newsletter Available!",
    "icon": "assets/main-page-logo-small-hat.png",
    "vibrate": [100, 50, 100],
    "data": {
      "dateOfArrival": Date.now(),
      "primaryKey": 1
    },
    "actions": [{
      "action": "explore",
      "title": "Go to the site"
    }]
  }
};
```

Cuando el payload ya se encuentra definido se procede a pasarlo a una promesa que busca a todos los usuarios suscritos y por medio del servicio de mensajes push se intenta enviar la notificación, en donde el navegador interpreta las instrucciones del payload para realizar las acciones.

```
Promise.all(USER_SUBSCRIPTIONS.map(sub => webpush.sendNotification(
  sub, JSON.stringify(notificationPayload) )))
  .then(() => res.status(200).json({message: 'Newsletter sent successfully.'}))
  .catch(err => {
    console.error("Error sending notification, reason: ", err);
    res.sendStatus(500);
  });
```

## Estado offline de la aplicación

Cuando la aplicación se encuentra ausente de internet mientras se encuentra con archivos de la cache es capaz de mostrar un mensaje de alerta que le indica al usuario para que se pueda reconectar y ver el contenido, el mensaje se encuentra en la raíz de la aplicación con el nombre de offline.html para ser recogido por el componente correspondiente y renderizarlo.

```
<h1> You are currently offline - check your network connection!</h1>
```

## Link del proyecto

<https://github.com/Daniel-Medina/angular-pwa>