Capítulo 3

Esto es más capítulo 4 (=tu solución al problema planteado en el capítulo 2), que 3 (= infraestructura en la que te has apoyado).

Aplicación web

ClassCity

ClassCity es una aplicacion que consiste en facilitar la comunicacion entre profesores y alumnos para que puedan reunirse. Es el claro ejemplo de una aplicación API REST donde tenemos una parte cliente(front-end) y otra parte servidor(backend), las cuales se comunican a partir de ficheros en formato JSON.

3.1 Diseño!!!
a vista de pájaro todo el sistema, sin demasiado detalle. Vaya dedicar cada una de las secciones siguientes a explicar cada parte de modo más prolijo.

Quizá movería 2.2 a una sección de diseño

3.1. Front-End

Esta parte de la aplicación corresponde al modelo cliente, el cual ha sido desarollado con el framework Angular2.

3.1.1. Fichero Raíz

Cuando el cliente accede a www.classcity.tk el fichero raiz de la app es index.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>ClassCity</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1"
     >
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs
     /twitter-bootstrap/4.0.0-alpha.5/css/bootstrap.css">
  <link rel="icon" type="image/x-icon" href="./assets/images/favicon.
     ico">

</head>
<body>
  <app-root>Loading...</app-root>
</body>
</html>
```

Este fichero html no es el unico que se descarga cuando nos bajamos una aplicación en angular.

1. **Main.ts** Este es el fichero raiz encargado de montar todo el cliente de nuestra app.

```
    import './polyfills';

    import { platformBrowserDynamic } from '@angular/platform-
       browser-dynamic';
    import { enableProdMode } from '@angular/core';
    import { environment } from './environments/environment';
    import { AppModule } from './app/app.module';

    if (environment.production) {
      enableProdMode();
    }

    platformBrowserDynamic().bootstrapModule(AppModule);
```

2. **App.Module.ts** Llegamos a uno de los ficheros mas importantes de toda la aplicación, donde se concentra todo lo necesario para que la app funcione. Si analizamos el fichero nos encontramos con un monton de importaciones, las cuales las he intentado separar por espacios en blancos dividiendolas en tres partes: las primeras importaciones están relacionadas con el framework angular2, las segundas importaciones estas relacionadas con servicios que he utilizado y las ultimas importaciones corresponden a la diferentes partes de la aplicación.

   Si continuamos profundizando en el codigo encontramos una variable ROUTES, la cual se encargara de asignar a cada path un componente especifico.

   Por último encontramos el motor de la aplicación

```
  @NgModule({
declarations: [
  AppComponent,
  Intro,
  LoginAlumno,
  LoginProfesor,
  SignupAlumno,
  SignupProfesor,
  HomeAlumno,
  HomeProfesor,
  ProfesorDetail
],
imports: [
  BrowserModule,
  FormsModule,
  HttpModule,
  FileUploadModule,
  AgmCoreModule.forRoot({
    apiKey: 'AIzaSyCYUVL5zFNpT0vaziTcpEUUbsmqZ7YRERM'
  }),
  RouterModule.forRoot(ROUTES)
],
```

```
  providers: [AuthGuard, {provide: AuthHttp, useFactory:
      authHttpServiceFactory,
    deps: [ Http, RequestOptions ]}, AlumnoService],
  bootstrap: [AppComponent]
})
```

Si observamos las importaciones, encontramos modulos como:

1. **FormsModule** Lo utilizamos para los formularios.
2. **HttpModule** Agiliza las peticiones tipo AJAX.
3. **FileUploadModule** Libreria necesaria para el envio de imagenes.
4. **AgmCoreModule** Libreria GoogleMaps para Angular, apiKey es la clave necesaria para realizar peticiones al servidor de google.

Justo después de las importaciones llegan los "Providers", que como su propio nombre indica son "proveedores de servicios". En este caso utilizaremos servicios como AuthHttp para una autenticación mas segura.

3.1.2. Servicios

Classcity es una aplicación web que necesita de la comunicación con varios backends para poder funcionar. De aquí los servicios que interactúan con nuestro backend y con otros como el de google.

1. **Angular/router** Este es un servicio interno en la libreria de angular, el cual capacita a la aplicación a poder enrutar cualquier URL interna.
2. **Angular2-jwt y Authguard**
3. **Angular2-google-maps**
4. **Imagenes**
5. **Google Service Geolocation**

3.1.3. Componentes

A continuación se presentan los hitos en orden cronológico:

1. **Login**
2. **Sign-up**
3. **Buscar Profesores**
4. **Navigator Geolocation**
5. **Logout**
6. **Notificacion**
7. **Chat**

¿No hay pantallazos?
Dónde está el lado servidor?

Quizá conviene partir la descripción de la aplicación en dos capítulos, uno enfocado en el lado cliente (interacción con los humanos...), y el otro en el lado servidor (base de datos, implantación en la nube...)