Práctica #3

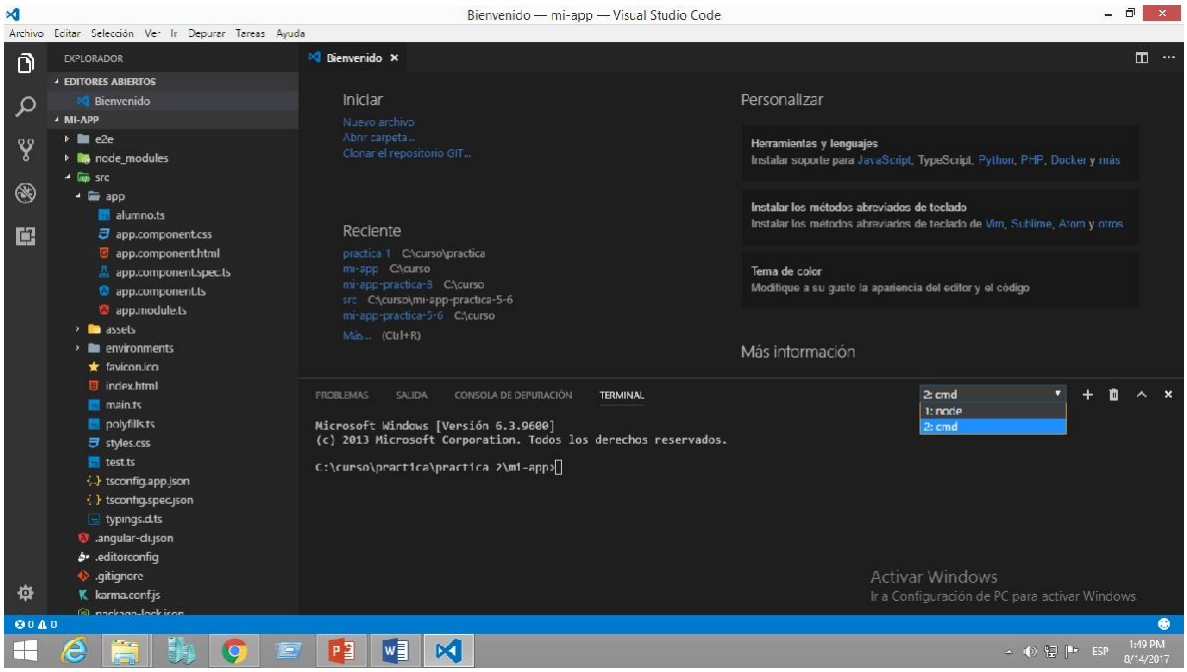
Objetivos

- Instalar y Utilizar Angular Material 2 v. 5
- Uso de eventos
- Poner en práctica ngIf, ngFor, ngClass

Para ello vamos a agregar a la aplicación ya desarrollada un array de alumnos, vamos a listar ese array de alumnos en una lista y cuando seleccionemos un alumno, cambiaremos el estilo de ese ítem de la lista y vamos a mostrar a la derecha los datos del alumno, permitiendo la edición de su nombre.

1) Instalar Angular Material 2 (ver 5.1)

Vamos a agregar una nueva ventana de TERMINAL a la que ya tenemos. Presione **CTRL+MAYUSCULA+ñ** para agregarla. Ahora en la venta de terminal podrá ver en el combo de terminales las dos abiertas



Asegúrese de estar ubicado en la carpeta **practica\mi-app**.

Desde la terminal ejecutar el siguiente comando npm:

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

Hay un doble guión antes de save.

Esto instalará en la carpeta **node_modules** las librerías de Angular Material 2 y agregará estas dependencias al nuestro proyecto (**pakage.json**). Note que además de **Angular Material** estamos agregando una librería adicional **Angular CDK**. Esta última librería es utilizada por **Angular Material**. Para más información puede ingresar a https://material.angular.io/cdk .

Otra librería adicional que debemos instalar es el módulo de animaciones de Angular. Para ello ejecute el siguiente comando en la terminal:

npm install --save @angular/animations

La librería de animación es utilizada por **Angular Material** para el efecto "ripple" sobre los botones. Con esto finalizamos el punto 1.

2) Incorporar a nuestro proyecto la librería de Material Icon

Normalmente las aplicaciones web utilizan íconos. Google desarrolló la librería **Material Icons** (https://material.io/tools/icons/) y la embebió en un font. Esta librería de íconos concuerda con las especificaciones del Material Design respecto a imágenes y símbolos. Para utilizar esta librería siga los siguientes pasos:

- Edite el archivo index.html y agregue la línea que está resaltada en amarillo

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

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

- Para ver la lista de íconos disponibles puede ingresar a https://material.io/tools/icons/. Guarde los cambios.

3) Incorporar la librería Angular Material 2 en nuestra aplicación.

Además de agregar la librería **Angular Material**, tenemos que incorporar los elementos que vamos a utilizar de ella en nuestra aplicación. Angular trabaja con **módulos (NgModules)** para dividir nuestra aplicación e incorporar funcionalidades de librerías en ella. Por lo general las librerías que están desarrolladas en Angular (como es el caso de Angular Material 2) están modularizadas permitiéndonos seleccionar los módulos que necesitamos.

Hay dos formas de incorporar los módulos de esta librería, una es incorporar todas las funcionalidades y componentes, aunque nuestra aplicación sólo utilice unos pocos. Esta opción tiene la ventaja que, desde el desarrollo, luego del setup inicial nos olvidamos de ir agregando uno por uno las partes que vamos a utilizar. La gran desventaja que tiene esta opción es que el bundle (archivos de nuestra app) que genera angular-cli es muy grande, impactando en la performance inicial de nuestra aplicación.

La otra opción es ir agregando una por una las partes que vamos a utilizar. Desde el desarrollo requiere más trabajo, pero la performance inicial de nuestra aplicación no se verá impactada drásticamente. Para nuestro caso vamos a utilizar esta última opción.

Para incorporar los módulos de Angular Material 2 tenemos que incorporarlos en el módulo principal de la aplicación.

Vamos por ahora a utilizar los componentes de íconos (https://material.angular.io/components/icon/overview) , Toolbar (https://material.angular.io/components/toolbar/overview) y Card (https://material.angular.io/components/card/overview) .

- Edite el archivo **app.module.ts** . Agregue debajo del último import las siguientes líneas de códigoL

```
import { MatIconModule } from '@angular/material/icon';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatCardModule } from '@angular/material/card';
```

Esto le permitirá al compilador de TypeScript hacer referencias a los módulos **MatIconModule**, **MatToolbarModule** y **MatCardModule** dentro del módulo de la aplicación. **MatToolbarModule** es el componente de toolbar, **MatCardModule** es el componente de tarjetas y el **MatIconModule** es para utilizar el Font de íconos. Ya veremos más adelante para que sirve cada uno.

- Además de hacer referencias a estos módulos, tenemos que configurar que cuando el runtime de Angular cargue a memoria el módulo principal de la aplicación, también cargue estos tres módulos. Para ello agregue en el mismo archivo las líneas que están resaltadas en amarillo al array de **imports** del módulo:

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    MatToolbarModule,
    MatCardModule,
    MatIconModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Como podemos ver, en el decorador de **@NgModule**, hay un JSON con varios arrays. Vamos a ir viendo cada uno de estos arrays a lo largo del curso. El array de **Imports** define cuales son los módulos dependientes que deben cargarse a memoria para que el módulo principal funcione correctamente. No confundir este array de imports con la sentencia **import** de TypeScript. En este caso las dos deben estar en sincronía.

- Guarde los cambios. Si no está ejecutándose la aplicación ejecute en la terminal **ng serve --open** y vea que no haya errores de compilación.

Para finalizar este paso, necesitamos a nivel de CSS el ***tema*** de Angular Material 2 que vamos a utilizar. EL **Tema** es un conjunto de reglas CSS que define cual es el color primario y el secundario de los componentes de **Angular Material 2**. Para más información sobre este tema puede consultar en https://material.io/collections/color/.

- Edite el archivo **styles.css** que se encuentra en la carpeta **src**. Todas las reglas CSS que definamos aquí serán heredadas por todos los componentes de nuestra aplicación. Agregue las siguientes líneas de código:

```
@import "@angular/material/prebuilt-themes/indigo-pink.css";

body {
  margin: 0px;
}
```

El **@import** carga las definiciones CSS del tema de combinación de color indigo – rosa. La otra regla **body** elimina los márgenes que vienen por default en el cuerpo de la página.

- Guarde los cambios

4) Utilización de los componentes Toolbar y Card en nuestra aplicación

- Edite el archivo **app.component.css** y **reemplace todo su contenido** por el siguiente código:

```
.content {
    margin-top: 3em;
    display: grid;
    grid-gap: 3em;
    grid-template-columns: 1fr 300px 300px 1fr;
    grid-template-rows: 1fr;
}

.avatar {
  margin-right: 0.5em;
  color: #ccc;
  font-size: 24px;
}

h2 {
  display: flex;
}
```

El código anterior simulaba el comportamiento de estos componentes. Ahora sólo tendremos reglas css que nos servirán para dar estilo a las diferentes partes de nuestro html.

Temas de auto estudio adicionales: Cuales son los efectos de las reglas h2 y .content? que es display: flex y display: grid?

- Edite el archivo **app.component.html** y reemplace el primer div que utilizamos para el toolbar de nuestra app por el siguiente código:

```
<mat-toolbar color="primary" class="mat-elevation-z3">
    <span>Gestión de cursos y alumnos</span>
</mat-toolbar>
```

- Aquí hemos utilizado el componente **Toolbar** en su forma más sencilla. Guarde los cambios y vea los resultados en el navegador.
- Cambie el valor del atributo **color** a "**secondary**" y vea los resultados. También pruebe de probar de eliminar el atributo color.
- En el mismo archivo reemplace todo el div class="alumno" por:

```
<div class="content">
  <div></div>

  <mat-card>
    <mat-card-header>
        <mat-card-title>
          <h2>
            <mat-icon class="avatar">account_circle</mat-icon>
            {{alumno.nombre}} {{alumno.apellido}}
          </h2>
        </mat-card-title>
        <mat-card-subtitle><hr></mat-card-subtitle>
    </mat-card-header>

    <mat-card-content>
      <p>
        Perfil: {{ Perfiles(alumno.perfil) }} <br>
        Sexo: {{ Sexos(alumno.sexo) }} <br>
        Activo: {{ alumno.activo }}<br><br>
      </p>
    </mat-card-content>
  </mat-card>

  <div></div>
  <div></div>

</div>
```

- Guarde los cambios y revise el cómo se ve ahora la aplicación.

También revise el cómo está estructurado el **div .content** a través de su regla css:

✓ display: grid – define la utilización de CSS GRID para el layout de este elemento
✓ grid-gap: 3em – el margen que tendrá cada elemento del layout entre sí.
✓ grid-template-columns: 1fr 300px 300px 1fr
  - El layout tendrá 4 columnas
  - La segunda y tercera columna tendrán un ancho de 300 px.

- La primera y la última serán tan anchas como el resto del ancho de la página dividido en dos. La unidad css **fr** (fractional unit) nos permite trabajar libremente sin tener en cuenta el valor real de cada dispositivo.

✓ grid-template-rows: 1fr
- El layout tendrá una fila

5) Preparar el módulo principal de la aplicación para los siguientes puntos

Vamos a agregar una lista de alumnos que vamos a poder seleccionar para mostrar sus datos en el el panel derecho en formato card. Para ello:

- Edite el archivo **app.module.ts** y agregue los siguientes **imports** marcados en amarillo:

```
import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms'; // <-- NgModel lives here
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';

import { MatIconModule } from '@angular/material/icon';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatCardModule } from '@angular/material/card';
import { MatListModule } from '@angular/material/list';
import { MatInputModule } from '@angular/material/input';
import { MatFormFieldModule } from '@angular/material/form-field';
```

✓ **FormsModule** nos permitirá utilizar two way data binding
✓ **BrowserAnimationsModule** es necesario para ciertas funcionalidades que utilizaremos de **Angular Material**
✓ **MatListModule**, **MatInputModule** y **MatFormFieldModule** son componentes que utilizaremos para mostrar listas y editar su contenido.

- Agregue en el array de **imports** del **@ngModule** los módulos importados vía **TypeScript** recientemente:

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
```

```
    MatToolbarModule,
    MatCardModule,
    MatIconModule,
    MatListModule,
    MatInputModule,
    MatFormFieldModule,
    FormsModule,
    BrowserAnimationsModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

6) Modificar la clase Alumno

Edite el archivo **alumno.ts** y agréguele la propiedad **id: number** a la clase **Alumno**. Necesitaremos esta propiedad id para identificar a los diferentes alumnos.

```
export class Alumno {
  constructor(
    public id: number,
    public nombre: string,
    public apellido: string,
    public sexo: number,
    public activo: boolean,
    public perfil: number
  ) {}
}
```

7) Agregar el array de alumnos que mostraremos en la lista

Edite el archivo **app.component.ts**. **Reemplace** la propiedad **alumno** el código de la clase por lo siguiente:

```
 alumnos: Alumno[] = [
   {id: 1, nombre: 'Juan', apellido: 'Perez', sexo: 1, perfil: 0, activo:
   true},
   {id: 2, nombre: 'Pedro', apellido: 'Garcia', sexo: 1, perfil: 1, activo:
   true},
   {id: 3, nombre: 'Ana', apellido: 'Romero', sexo: 0, perfil: 2, activo:
   true},
   {id: 4, nombre: 'Maria', apellido: 'Gutierrez', sexo: 0, perfil: 2,
   activo: true},
```

```
  {id: 5, nombre: 'Esteban', apellido: 'Smith', sexo: 1, perfil: 0,
activo: true} ];

alumnoSeleccionado: Alumno = null;
```

Ya no tendremos la propiedad **alumno** en nuestro componente. Ahora tenemos un array de alumnos que nos permitirá armar una lista a partir de allí.

También tendremos una nueva propiedad llamada **alumnoSeleccionado** que contendrá el alumno que iremos seleccionando con el mouse.

Recuerde salvar los cambios, vea que ha sucedido ahora en el navegador.

¿Por qué sucedió esto? Vea en la herramienta del desarrollador del navegador (F12) los mensajes que se encuentran en la consola.

8) Mostrar la lista de alumnos: *ngFor

Vamos a utilizar el ngFor para recorrer el array y mostrar el nombre y el apellido en una lista de Angular Material. Para ello editaremos el **app.component.html** y reemplazaremos todo el **div class="content"** por lo siguiente:

```
<div class="content">
  <div></div>

  <mat-card class="lista">
    <mat-list>
      <mat-list-item
        *ngFor="let alumno of alumnos"
        class="alumno-list-item"
    >
      <div mat-line>
        <h2>
            <mat-icon class="avatar">account_circle</mat-icon>
            {{alumno.nombre}} {{alumno.apellido}}
        </h2>
      </div>

          <p mat-line>
            Perfil: {{Perfiles(alumno.perfil)}}<br><br> </p>
      </mat-list-item>
    </mat-list>
```

```
  </mat-card>

  <div></div>
  <div></div>

 </div>
```

Este html que hemos agregado va a mostrar la lista de alumnos con un **ngFor** y iterando sobre el array de alumnos con el componente **mat-list-item.** Vea que este componente está dentro de otro componente: **mat-list**, que brinda la funcionalidad de listas. A su vez este componente esta dentro de una tarjeta (componente mat-**card**) para dar una visual más agradable.

- Guarde los cambios.
- Edite los estilos del componente (**app.component.css**) y agregue los siguientes estilos al mismo:

```
.all-width {
  width: 100%;
}

.lista {
  padding: 0;
}

.alumno-list-item {
  border-bottom: 1px solid #ddd;
  padding-top: 1em;
  padding-bottom: 1em;
}

.alumno-list-item:hover {
  background-color: #eee;
}

.alumno-list-item p {
  padding-left: 2.5em;
}

mat-list {
  padding-top: 0px;
}
```

- Guarde los cambios y pruebe la aplicación.

9) Seleccionar un alumno y mostrar sus datos en otro panel: Eventos *ngIf

Ahora agregaremos la funcionalidad de seleccionar un alumno de la lista y mostrar sus datos en el tercer panel (recuerda el **display: grid** de la clase CSS **content**?). Para ello vamos a atrapar el evento **click** que se haga sobre cada ítem de la lista:

- Edite el template **app.component.html** y agregue en el elemento **mat-list-item** la captura del evento **click** (resaltado en amarillo en las siguientes líneas de código):

```
<mat-list-item
    *ngFor="let alumno of alumnos"
    class="alumno-list-item"
    (click)="AlumnoSelect(alumno)">
```

- Agregue en el archivo **app.component.ts** la siguiente función que será llamada cada vez que hagamos un click sobre un ítem de la lista:

```
AlumnoSelect(alumno: Alumno) {
  this.alumnoSeleccionado = alumno;
}
```

- Cada vez que se seleccione un alumno, mostraremos sus datos a la derecha de la lista. Para ello reemplace el anteúltimo ** ** dentro del **div.content** (por qué el anteúltimo?) por las líneas que están en amarillo debajo del cierre del elemento **</mat-card>** :

```
</mat-card>

<mat-card *ngIf="alumnoSeleccionado">
    <h2>Alumno Seleccionado</h2>
    Nombre: {{alumnoSeleccionado.nombre}} <br>
    Apellido: {{alumnoSeleccionado.apellido}} <br>
    Sexo: {{Sexos(alumnoSeleccionado.sexo)}} <br>
    Perfil: {{Perfiles(alumnoSeleccionado.perfil)}} <br>
    Activo: {{alumnoSeleccionado.activo}}
</mat-card>
```

```
    <div></div>

</div>
```

Los datos se mostrarán en una tarjeta siempre y cuando exista un alumno seleccionado. El ***ngIf** nos permite evitar un error de runtime al momento de cargar la aplicación, justo antes de seleccionar un alumno.

- Guarde los cambios y pruebe la aplicación. Vea también de eliminar el *ngIf y vea el error obtenido en la consola de la herramienta de desarrolladores.

10) Marcar el alumno seleccionado en la lista

Vamos a cambiar el color de background del alumno cada vez que se lo seleccione.

- Edite el template **app.component.html** y agregue la línea que está en amarillo.

```
<mat-list-item
       *ngFor="let alumno of alumnos"
       class="alumno-list-item"
       (click)="AlumnoSelect(alumno)"
       [class.alumno-seleccionado] = "EstaSeleccionado(alumno)" >
          <div mat-line>
            <h2>
```

Esto hará que la clase CSS **alumno-seleccionado** se aplique al componente **mat-list-item** cada vez que la función **EstaSeleccionado** retorne verdadero.

- Edite el código del componente **app.component.ts** y agregue la siguiente función al final de la clase:

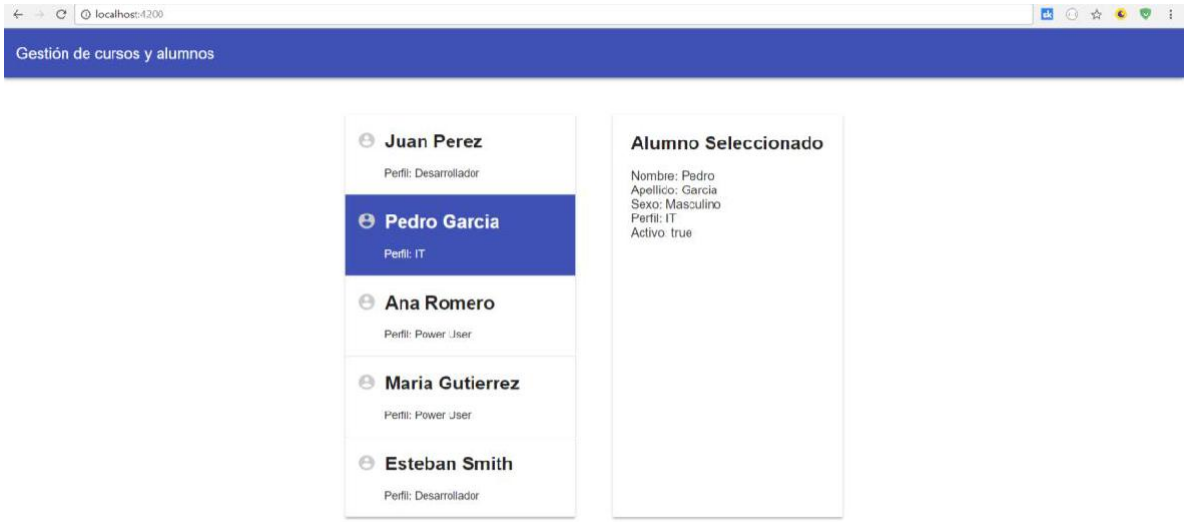
```
  EstaSeleccionado(alumno: Alumno) {
    if (this.alumnoSeleccionado) {
      return this.alumnoSeleccionado.id === alumno.id;
    } else {
      return false;
    }
  }
```

- Por último, agregue la clase CSS **alumno-seleccionado** al archivo app.component.css:

```
.alumno-seleccionado {
  background-color: rgb(63, 81, 181) !important;
```

```
  color: #fff;
  font-weight: bold;
}
```

- Pruebe la aplicación, en este punto deberá encontrarse con una página idéntica a esta:



11) Editar el nombre y apellido: Two Way Data Binding

Two Way Data Binding se utiliza en Angular a través de las prestaciones del módulo **ngModel**. Cuando veamos formularios en Angular entraremos en más profundidad en esta característica. Por ahora simplemente la utilizaremos.

- Edite el template **app.component.html** y cambie las líneas de código que usamos para mostrar el nombre y el apellido

  **Nombre: {{alumnoSeleccionado.nombre}}
**
  **Apellido: {{alumnoSeleccionado.apellido}}
**

En la tarjeta en donde mostramos todos los datos del alumno por las siguiente líneas resaltadas en amarillo:

```
  <mat-card *ngIf="alumnoSeleccionado">
      <h2>Alumno Seleccionado</h2>
      Nombre: <input name="nombre" placeholder="Nombre"
[(ngModel)]="alumnoSeleccionado.nombre">
      <br>
```

```
      Apellido: <input name=" apellido" placeholder="Apellido"
[(ngModel)]="alumnoSeleccionado.apellido">
      <hr>
      Sexo: {{Sexos(alumnoSeleccionado.sexo)}} <br>
      Perfil: {{Perfiles(alumnoSeleccionado.perfil)}} <br>
      Activo: {{alumnoSeleccionado.activo}}
  </mat-card>
```

- Guarde los cambios y pruebe la aplicación. Seleccione un alumno y edítelo. Vea cómo los cambios se propagan automáticamente a la lista.

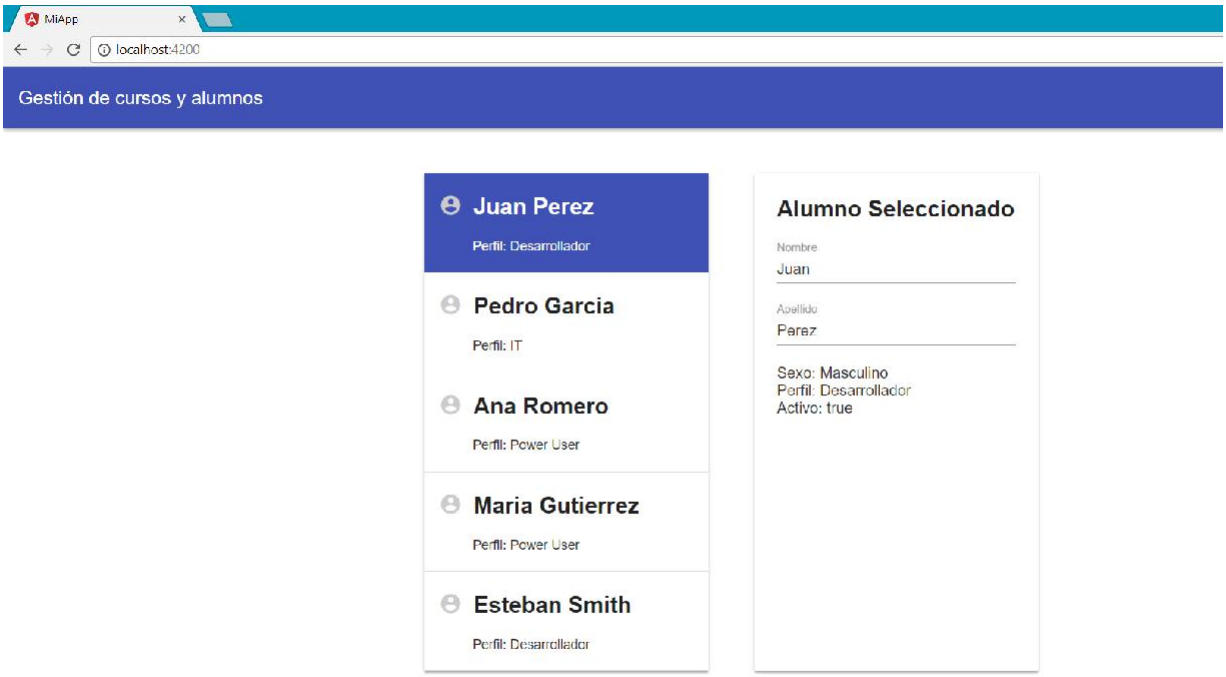
Angular Material 2 tiene su propia forma visual de mostrar los inputs boxs. Vamos a utilizarla a continuación.

- Reemplace las líneas agregadas recientemente por las siguientes resaltadas en amarillo:

```
  <mat-card *ngIf="alumnoSeleccionado">
      <h2>Alumno Seleccionado</h2>

      <mat-form-field class="all-width">
          <input matInput  name="nombre" placeholder="Nombre"
[(ngModel)]="alumnoSeleccionado.nombre">
      </mat-form-field>

      <mat-form-field  class="all-width">
        <input matInput name=" apellido" placeholder="Apellido"
[(ngModel)]="alumnoSeleccionado.apellido">
      </mat-form-field>

      Sexo: {{Sexos(alumnoSeleccionado.sexo)}} <br>
      Perfil: {{Perfiles(alumnoSeleccionado.perfil)}} <br>
      Activo: {{alumnoSeleccionado.activo}}
  </mat-card>
```

Aquí estamos utilizando el componente **mat-form-field**, que permite cambiar el look & feel de varios componentes de input html (https://material.angular.io/components/form-field/overview), y la directiva **MatInput** (https://material.angular.io/components/input/overview) en los elementos Input. La directiva y este último componente van de la mano. Entraremos en más detalle más adelante en el curso.

- Guarde los cambios y pruebe la aplicación. Al final tendría que tener un resultado idéntico al que se muestra a continuación:


MiApp
localhost:4200
Gestión de cursos y alumnos
Juan Perez
Perfil: Desarrollador
Pedro Garcia
Perfil: IT
Ana Romero
Perfil: Power User
Maria Gutierrez
Perfil: Power User
Esteban Smith
Perfil: Desarrollador
Alumno Seleccionado
Nombre
Juan
Apellido
Perez
Sexo: Masculino
Perfil: Desarrollador
Activo: true