Práctica #2

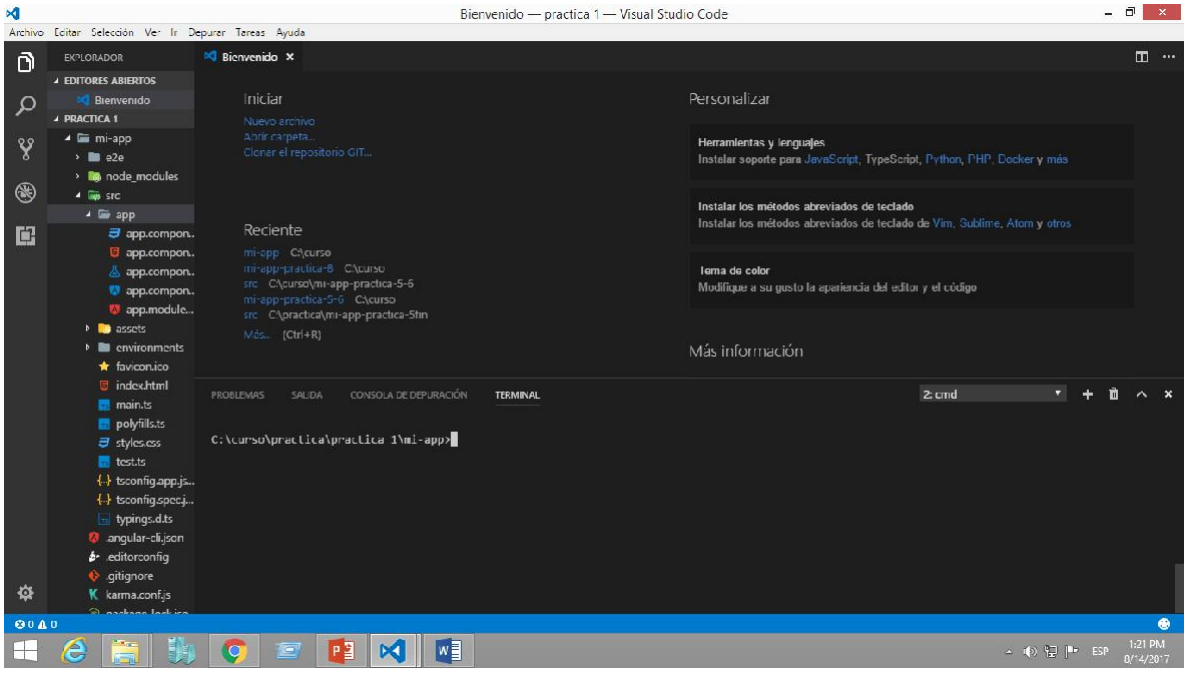
Objetivo

- Crear una clase con datos de Alumnos en TypeScript
- Crear una instancia de la clase
- Mostrar los datos de la instancia en un componente
- Este proyecto inicial será la semilla del proyecto del curso

1. Preparar el ambiente

Vamos a utilizar la aplicación creada en la práctica #1. Si tiene abierta aún la ventana de comando de la práctica anterior, ciérrela. Abra la aplicación con Visual Studio Code.

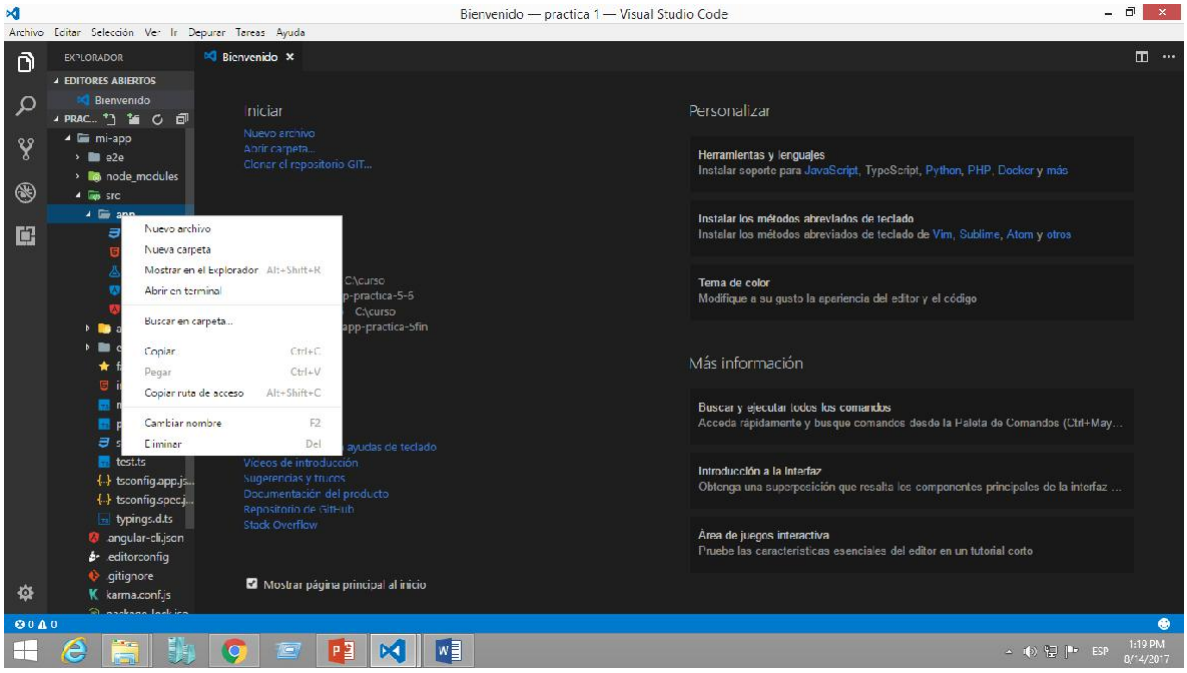
- Visual Studio Code tiene una ventana desde la cual se puede ejecutar comandos desde el propio IDE. Abra esta ventana seleccionando la opción de menú **Ver** y luego **Terminal Integrado** o presione **CTRL+ñ** (Si instaló la versión en ingles es **CTRL+~**).



- Asegúrese que este en la carpeta **mi-app** .
- Ejecute desde la ventana de **TERMINAL** el comando **ng serve** para ejecutar la aplicación.
- Desde el navegador navegue la url http://localhost:4200
- Puede cerrar la ventana de **TERMINAL** y reabrirla en cualquier momento. Con esta acción no se interrumpirá la ejecución de los comandos como el **ng serve**.

2. Crear la clase Alumno utilizando la sintaxis de TypeScript

- Desde el explorador de Visual Studio Code, cree un archivo **alumno.ts** en la carpeta **app**



- Agregue el siguiente código a este archivo:

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

Esto creará una clase **Alumno** en **TypeScript**, vea la palabra **export** delante de la clase, esto indica que la clase puede ser exportada fuera del código fuente para luego poder ser importada desde otro archivo.

Vea también la sintaxis del constructor de la clase, el constructor tiene una serie de parámetros **public** que **TypeScript** los convierte automáticamente en propiedades de la clase.

3. Crear la clase itemList en TypeScript

- Desde el explorador de Visual Studio Code, cree un archivo **itemList.ts** en la carpeta **app**
- Agregue el siguiente código a este archivo:

```
export class ItemList {
  constructor(
    public index: number,
    public descripcion: string
  ) {}
}
```

- Salve los cambios.

4. Utilizar las clases creadas en el component app.component.ts

- En el código de nuestro componente base **app.component.ts** agregue el siguiente código debajo del import:

```
  import { Alumno } from './alumno';
  import { ItemList } from './ItemList';
```

Esto es para que desde nuestro componente podamos utilizar las clases **Alumno e ItemList**. Vea como trabajan en conjunto el **export** (en la declaración) y el **import** (en el archivo en donde se va a hacer referencia).

- Debajo de **export class AppComponent** agregue la línea que está resaltada en amarillo:

```
export class AppComponent {
  alumno: Alumno = {nombre: 'Juan', apellido: 'Perez', sexo: 1,
  perfil: 0, activo: true};
```

Aquí tenemos una instancia de la clase alumno.

- Elimine la línea **title = '...';** ya no la necesitaremos más.
- Salve los cambios.

5. Modificar el template del componente app.component

Ahora vamos a modificar el template para mostrar los datos de la instancia **alumno** utilizando **one way data bindin**g. Recuerde guardar todos los cambios realizados y **verificar en este punto que el compilador no haya emitido algún error**.

- Edite el template del componente (**app.component.html**) y reemplace **todo** el código existente por el siguiente código:

```
<div style="text-align:center">
  <h1>
    Gestión de cursos y alumnos
  </h1>
</div>

<div class="alumno">
  <h2>Alumno</h2>
  Nombre: {{alumno.nombre}} <br>
  Apellido: {{alumno.apellido}} <br>
  Sexo: {{alumno.sexo}} <br>
  Perfil: {{alumno.perfil}} <br>
  Activo: {{alumno.activo}}
<div>
```

Vea el resultado en el navegador.

- Ahora estilizaremos el componente. En **app.component.css** agregue lo siguiente:

```
:host {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 16px;
}

h1 {
  padding: 1em;
  background-color: teal;
  color: white;
}

.alumno {
  color: #444;
  margin: 3em;
```

```
  padding: 1em;
  border: 1px solid #ccc;
  width: 200px;
  box-shadow: 0 10px 20px rgba(0,0,0,0.19), 0 6px 6px rgba(0,0,0,0.23);
  transition: all 0.3s cubic-bezier(.25,.8,.25,1);
}

.alumno:hover {
  box-shadow: 0 14px 28px rgba(0,0,0,0.25), 0 10px 10px rgba(0,0,0,0.22);
}
```

Vea el resultado. Que es el selector CSS **:host** ?

6. Utilizar la clase ItemList para las descripciones de perfil y sexo

- En el archivo **app.component.ts** agregue las siguientes líneas resaltadas en amarillo:

```
export class AppComponent {

  alumno: Alumno = {nombre: 'Juan', apellido: 'Perez', sexo: 1,
  perfil: 0, activo: true};

  sexos: ItemList[] = [ new ItemList( 0, 'Femenino'), new ItemList( 1,
                        'Masculino'), new ItemList( 2, 'Otro')
                      ];

  perfiles: ItemList[] = [ new ItemList( 0, 'Desarrollador'), new
                           ItemList( 1, 'IT'), new ItemList( 2, 'Power
                           User'),
                         ];

}
```

Vea como hemos definido un array de un tipo de datos y creado las instancias **ItemList** invocando al constructor de la clase.

- Vamos ahora a agregar dos funciones que realicen búsquedas sobre los arrays de **sexo** y **perfiles**. Agregue debajo de las líneas agregadas estas dos funciones que se detallan a continuación:

```
  Sexos(index: number): string {
    return this.sexos.find(item => item.index === index).descripcion;
  }

  Perfiles(index: number): string {
    return this.perfiles.find(item => item.index === index).descripcion;
  }
```

Es importante destacar la sintaxis de **TypeScript** para definir funciones:

<nombre>(parámetros tipados): tipo de dato de retorno {cuerpo de la función}

Los nombres de las funciones, por norma de estilo, es conveniente definirlos con la primera letra en mayúscula, a diferencia de los nombres de las propiedades que se escriben con la primera letra en minúscula.

Vea también como utilizamos la función find de JavaScript para buscar dentro del array ItemList y el uso de las arrow functions definidos en ECMAScript 6.

- Guarde los cambios.

- Por último, vamos a llamar estas funciones desde el template del componente. Edite el archivo **app.component.html** y modifique las líneas en las cuales mostramos el valor de la propiedad sexo y perfil por las líneas de código resaltadas en amarillo:

```
<div class="alumno">
  <h2>Alumno</h2>
  Nombre: {{alumno.nombre}} <br>
  Apellido: {{alumno.apellido}} <br>
  Sexo: {{ Sexos(alumno.sexo) }} <br>
  Perfil: {{ Perfiles(alumno.perfil) }} <br>
  Activo: {{alumno.activo}}
<div>
```

- Guarde los cambios y vea el resultado.