



# **Manual de Técnico**

**Prueba de Aptitud Entrevista**

**Realizado por Ing. Bryan Chalacán**

**05 de marzo de 2023**

# Control de cambios

| Versión | Lugar*            | Tipo** | Fecha      | Autor                      | Descripción             |
|---------|-------------------|--------|------------|----------------------------|-------------------------|
| 1.0     | Todo el documento | A      | 05/03/2023 | Ing. Bryan Chalacán Araujo | Creación del documento. |

\* Sección del documento, Tabla, Figura.

\*\* **A** Alta; **B** Baja; **M** Modificación

## Contenido

|   |    |
|---|----|
| Control de cambios .....  | 2  |
| Introducción y Herramientas Utilizadas.....                         | 4  |
| Creación de la Base de Datos .....                                  | 4  |
| Creación del API (Back-End) .....                                   | 4  |
| Configuración de JWT(JSON WEB TOKEN) en la parte del back-end. .... | 8  |
| Creación y Configuración de Proyecto en Angular(Front-End) .....    | 10 |
| Combinar Proyecto Back-End y Front-End .....                        | 12 |
| COMO LEVANTAR EL PROYECTO .....                                     | 14 |

# Introducción y Herramientas Utilizadas

En el presente documento se describen los pasos a seguir para crear, configurar y correr la aplicación. Las herramientas tecnológicas que se utilizaron fueron las siguientes:

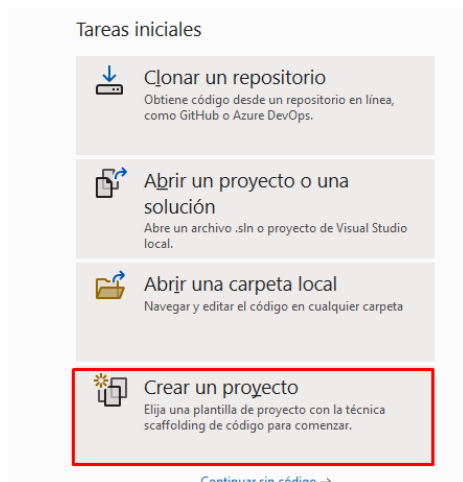
- Para el back-end y creación del API se utilizó .NET en su versión de SDK 6.0
- El IDE de desarrollo para el API fue Visual Studio 2022
- Para el front-end se utilizó Angular el cual combina el lenguaje de Type Script, HTML5 y Css.
- El IDE de desarrollo para el front-end fue Visual Studio Code en su versión 1.76 año 2022.
- Para la creación de un proyecto en Angular debemos instalar NodeJs, para mi caso utilicé la versión 14.17.6.
- Además debemos instalar la CLI de angular que es una línea de comandos que nos va a facilitar el desarrollo de mi aplicación, utilizamos la versión 12.0.0
- Para el almacenamiento de datos utilice el motor de base de datos PostgreSQL en su versión 14 que viene acompañado de su interfaz pgAdmin4 en su versión 6.1.0.

## Creación de la Base de Datos

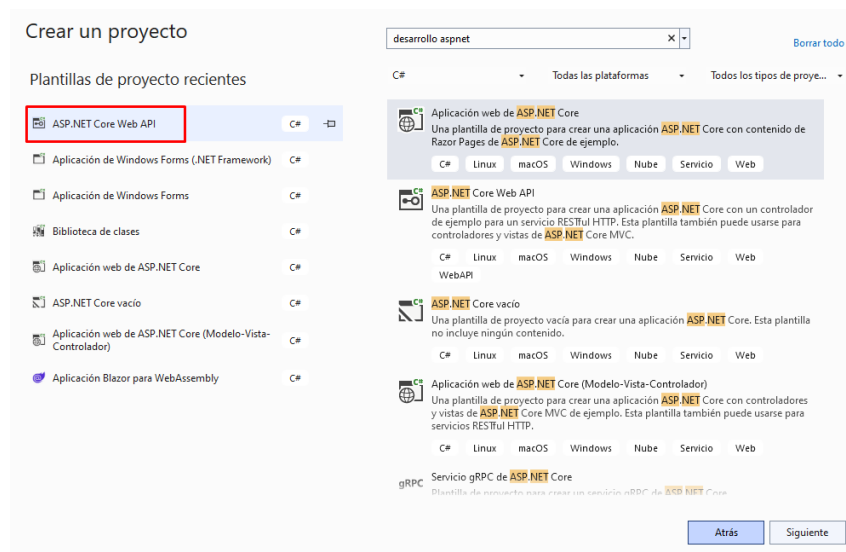
- Para la creación de la base de datos vamos a guiarnos en el documento de guía de la prueba, generaremos un script el cual va a ir acompañado de este documento para la creación de la base y sus respectivas tablas.

## Creación del API (Back-End)

- Vamos a abrir el IDE y vamos a crear un nuevo proyecto:



- Vamos a crear un proyecto de tipo ASP.NET Core Web API en caso de no tener en el inicio rápido vamos a buscarlo , cuando nos aparezca lo seleccionamos y damos clic en siguiente:



- Nos aparecerá la pantalla para asignarle un nombre y la ruta donde guardar el proyecto, como nombre lo he colocado Test\_Enterprise\_Back, cuando este todo lleno damos clic en siguiente:



- En la siguiente vamos a seleccionar la versión del framework, para mi ejemplo he seleccionado la versión .NET 6.0, una vez seleccionado procedemos a dar clic en CREAR :

### Información adicional

ASP.NET Core Web API   C#   Linux   macOS   Windows   Nube   Servicio   Web   WebAPI

Framework ⓘ

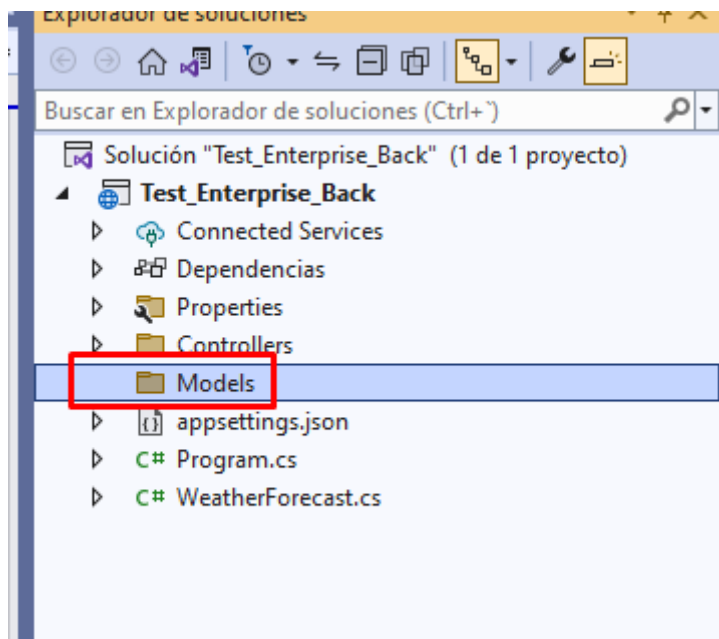
Authentication de campo ⓘ

☒ Configurar para HTTPS ⓘ  
☐ Habilitar Docker ⓘ

Sistema operativo de Docker ⓘ

☒ Usar controladores (desactivar para usar API mínimas) ⓘ  
☒ Habilitar compatibilidad con OpenAPI ⓘ  
☐ No usar instrucciones de nivel superior ⓘ

- Así finalizamos la creación de la base de nuestro proyecto y tendremos una estructura parecida a la imagen inferior pero debemos añadir dos carpetas mas una que se llame Models y otra que se llama Data.



- Procedemos a instalar las dependencias o Nuggets necesarios, que nos van a servir para diferentes funcionalidades de nuestro proyecto, las dependencias que utilizamos son las siguientes:
  - Microsoft.EntityFrameworkCore.Tools Version="6.0.14"
  - Npgsql.EntityFrameworkCore.PostgreSQL Version="6.0.8" : permite la conexión con el motor de base de datos PostgreSQL
  - Microsoft.AspNetCore.Authentication.JwtBearer Version="6.0.14" Nos permite la implementación de la seguridad ante el envío de peticiones con JWT.

```
</PackageReference>
<PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="6.0.14"/>
<PackageReference Include="Npgsql.EntityFrameworkCore.PostgreSQL" Version="6.0.8"/>
<PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="6.0.14"/>
```

- En el archivo appsettings.json colocamos la configuración hacia la base de datos y una constante que nos va a ayudar para el tema del Json Web Token.

```
{
  "AppSettings": {
    "Code": "1254784saf1585236ddddd115d5d2d2d56a32df4f5e5a2d1f4f5ff21fd2s15e1f2s1d5s"
  },
  "Logging": {
    "IncludeScopes": false,
    "Debug": {
      "LogLevel": {
        "Default": "Warning"
      }
    },
    "Console": {
      "LogLevel": {
        "Default": "Warning"
      }
    }
  },
  "ConnectionStrings": {
    "TestEnterpriseContext": "Server=localhost;Port=5432;Database=TestEnterpriseDB;User Id=postgres;Password="
  }
}
```

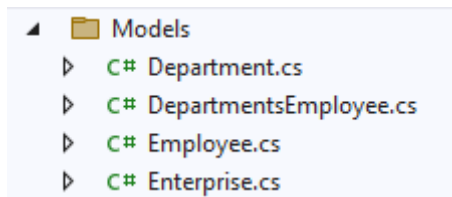
- En el archivo Program.cs vamos a añadir una configuración para asignarle la url y el puerto con el que queremos que se levante nuestra aplicación esto servirá después para combinar los archivos del back y los que se creen del front y tengamos todo en un solo proyecto. La arquitectura los considera como proyectos por separado ya que se van a levantar en diferentes instancias pero “físicamente” en el mismo proyecto.

```
Test_Enterprise_Back | Test_Enterprise_Back.Program
using Test_Enterprise_Back;

namespace Test_Enterprise_Back
{
    public class Program
    {
        public static void Main(string[] args)
        {
            BuildWebHost(args).Run();
        }

        public static IWebHost BuildWebHost(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>()
                .UseUrls("http://localhost:5056")
                .Build();
    }
}
```

- Vamos a crear las clases modelo de la base de datos en la carpeta Models:



- Crearemos dentro de la carpeta Data una clase que nos permita generar el Contexto de datos para poder utilizar los objetos del modelo en los diferentes controladores.

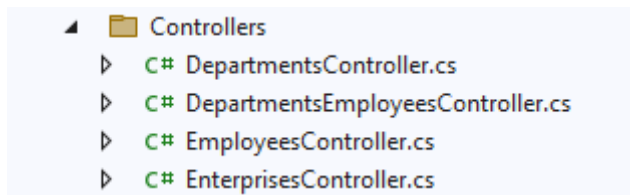
```

public class TestEnterpriseContext : DbContext
{
    public TestEnterpriseContext(DbContextOptions<TestEnterpriseContext> options)
        : base(options)
    {
    }

    public DbSet<Test_Enterprise_Back.Models.Department> Department { get; set; }
    public DbSet<Test_Enterprise_Back.Models.DepartmentsEmployee> DepartmentsEmployee {
    public DbSet<Test_Enterprise_Back.Models.Employee> Employee { get; set; }
    public DbSet<Test_Enterprise_Back.Models.Enterprise> Enterprise { get; set; }
}

```

- **Creamos** dentro de la carpeta Controllers los controladores para poder implementar los métodos o EndPoints de cada objeto (CRUDS).



## Configuración de JWT(JSON WEB TOKEN) en la parte del back-end.

- Creamos carpetas y clases que nos van a servir para configurar JWT:
- Método Encryptar Contraseña.- Dentro de la clase Encrypt se encuentra este método que permite encryptar cualquier texto que pase como parámetro de acuerdo al algoritmo de SHA256 , recorre la cadena o el parámetro de entrada para ir encriptando carácter por carácter.
- La clase AuthRequets. - Sirve para determinar cuáles serán los parámetros que el usuario deberá ingresar en el login.
- La clase UsuarioResponse. - Esta clase contiene los elementos que van a conformar el cuerpo del token.
- La clase usuario Interfaz. - Esta clase contiene el método que va a ser llamado por el controlador para la generación del token.
- La clase Usuario Service. - contiene la sobreescritura del método que está en la interfaz para implementar la lógica de la generación del token.
  - o Primero instanciamos el parámetro que debe ser colocado en el archivo appsettings.json, el cual tendrá una variable denominada CLAVE que



serán una cadena de número aleatorios que la librería bearerToken usara para formar el código principal del token.

- Luego instanciamos al dbContext para realizar la consulta a la tabla usuario para saber si el nombre y contraseña que se ingresaron son correctos.
- En caso de ser correcto llenamos la instancia del objeto usuario Reponse con su respectivo nombre de usuario y token.
- El método GETTOKEN genera el token y lo configura con tiempo de caducidad para que el usuario que lo haya generado exitosamente lo pueda usar mientras el token se encuentre vigente.
- La clase usuarioController.- Contiene el método http put "login" denominado autenticar, este atreves de una instancia a la clase usuarioInterfaz Accede al método descrito en el anterior punto el cual nos devuelve una respuesta del objeto usuarioRespinse, si este llega a ser exitoso instanciamos al objeto respuesta el cual es llenado por una serie de parámetros correctos para que sean la respuesta a la petición del usuario.
- Configuración en el archivo StartUp.cs : en el método ConfigureServices se debe colocar las siguientes líneas:

```
services.AddScoped<UserInterface, UserService>();
var appSettingsSection = Configuration.GetSection("AppSettings");
services.Configure<AppSettings>(appSettingsSection);
//JWT
var appSettings = appSettingsSection.Get<AppSettings>();
var llave = Encoding.ASCII.GetBytes(appSettings.Code);
services.AddAuthentication(d =>
{
    d.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    d.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(d =>
{
    d.RequireHttpsMetadata = false;
    d.SaveToken = true;
    d.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuerSigningKey = true,
        IssuerSigningKey = new SymmetricSecurityKey(llave),
        ValidateIssuer = false,
        ValidateAudience = false
    };
});
```

- El configure nos quedaría de la siguiente manera:

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseDefaultFiles();
    app.UseStaticFiles();
    app.UseRouting();
    app.UseAuthorization();
    app.UseAuthentication();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

- Finalmente en nuestros controladores colocamos la siguiente línea para garantizar que se permita el ingreso a los diferentes EndPoints siempre que se tenga el token de seguridad correcto.

```
[Route("api/[controller]")]
[ApiController]
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
```

## Creación y Configuración de Proyecto en Angular(Front-End)

- En la ruta en la que vayamos a crear nuestro proyecto vamos a abrir una consola de comandos y con la ayuda del comando ng new “nombre de proyecto” vamos a crearlo.

```
PS D:\TestEnterprise> ng new TestEnterpriseFront
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE TestEnterpriseFront/angular.json (3117 bytes)
CREATE TestEnterpriseFront/package.json (1083 bytes)
CREATE TestEnterpriseFront/README.md (1009 bytes)
CREATE TestEnterpriseFront/tsconfig.json (783 bytes)
CREATE TestEnterpriseFront/.editorconfig (274 bytes)
CREATE TestEnterpriseFront/.gitignore (604 bytes)
CREATE TestEnterpriseFront/.browserslistrc (703 bytes)
CREATE TestEnterpriseFront/karma.conf.js (1436 bytes)
CREATE TestEnterpriseFront/tsconfig.app.json (287 bytes)
CREATE TestEnterpriseFront/tsconfig.spec.json (333 bytes)
CREATE TestEnterpriseFront/src/favicon.ico (948 bytes)
CREATE TestEnterpriseFront/src/index.html (305 bytes)
CREATE TestEnterpriseFront/src/main.ts (372 bytes)
CREATE TestEnterpriseFront/src/polyfills.ts (2820 bytes)
CREATE TestEnterpriseFront/src/styles.css (80 bytes)
CREATE TestEnterpriseFront/src/test.ts (743 bytes)
CREATE TestEnterpriseFront/src/assets/.gitkeep (0 bytes)
CREATE TestEnterpriseFront/src/environments/environment.prod.ts (51 bytes)
CREATE TestEnterpriseFront/src/environments/environment.ts (658 bytes)
CREATE TestEnterpriseFront/src/app/app-routing.module.ts (245 bytes)
CREATE TestEnterpriseFront/src/app/app.module.ts (393 bytes)
CREATE TestEnterpriseFront/src/app/app.component.html (23809 bytes)
CREATE TestEnterpriseFront/src/app/app.component.spec.ts (1096 bytes)
CREATE TestEnterpriseFront/src/app/app.component.ts (223 bytes)
CREATE TestEnterpriseFront/src/app/app.component.css (0 bytes)
Installing packages (npm)...
```

- Abrimos nuestro proyecto con la ayuda de visual studio code y empezamos a configurarlo:
- Abrimos el archivo tsconfig.json y colocamos la siguiente configuración con la finalidad de que Angular no sea tan estricto al momento de codificar y evitarnos errores en tiempo de ejecución y complicación.

```

/* To learn more about this file see:
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "moduleResolution": "node",
    "importHelpers": true,
    "target": "es2017",
    "module": "es2020",
    "lib": [
      "es2018",
      "dom"
    ],
  },
  "angularCompilerOptions": {
    "preserveWhitespaces": true
  }
}

```

- Vamos a instalar las siguientes librerías con la ayuda del comando npm el cual es un repositorio de librerías que se pueden utilizar para nuestro proyecto angular:
- ng-bootstrap: es la Librería de npm y bootstrap que permite utilizar estilos personalizados.
- ng-select: una Librería que nos permite utilizar combos dinámicos.
- Bootstrap: la librería propia de bootstrap
- ngb-modal: librería que permite abrir modales o pantallas emergentes.
- ngx-toaster: librería que permite mostrar mensajes personalizados.
- 
- Podemos ver el conjunto de estas librerías en el archivo package. json.
- Procedmos a configurar el archivo angular. json:
- Lo primero es el outputPath colocar el nombre wwwroot en este se almacenará el compilado de nuestro proyecto front-end.

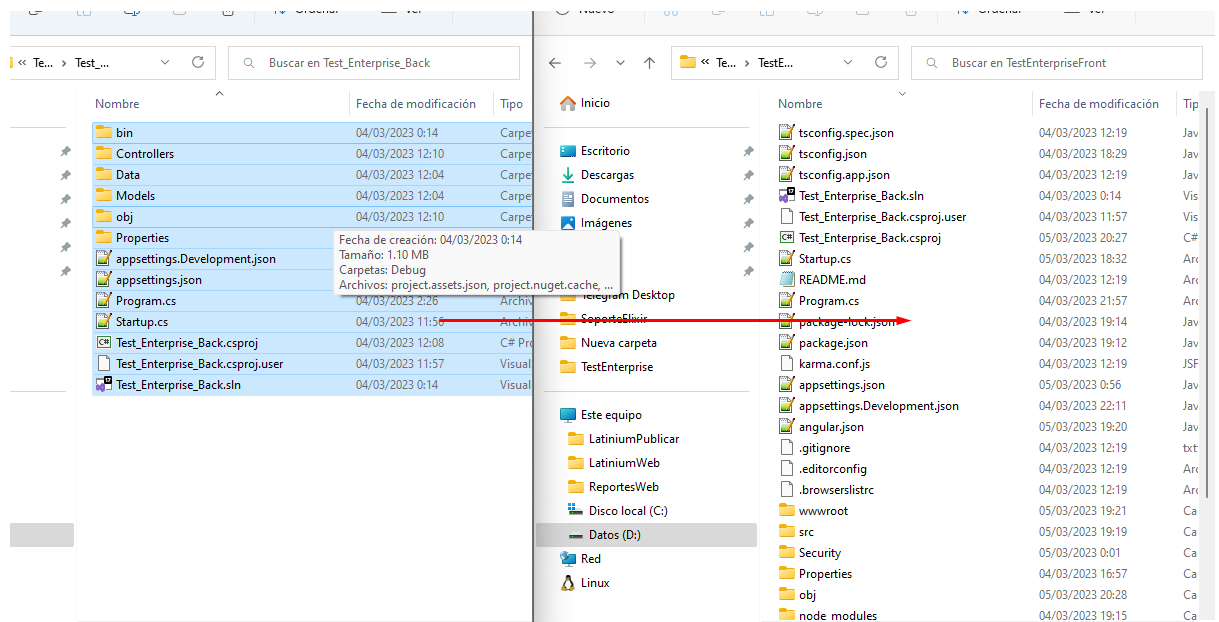
```

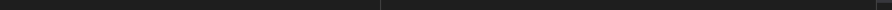
angular.json > {} projects > {} TestEnterpriseFront > {} architect > {} test > {} options > [ ] styles > 0
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "TestEnterpriseFront": {
      "projectType": "application",
      "schematics": {
        "@schematics/angular:application": {
          "style": "scss",
          "spec": false
        }
      },
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "wwwroot",

```

- ```
src/assets
],
"styles": [
  "node_modules/bootstrap/dist/css/bootstrap.min.css",
  "src/styles.scss",
  "node_modules/ngx-toastr/toastr.css"
],
"scripts": []
```

- Vamos a abrir la ruta de ambos proyectos en dos carpetas diferentes y vamos a copiar los archivos de la una a la otra, la del back en la del front:



- 
- PROBLEMS 12 OUTPUT TERMINAL DEBUG CONSOLE
- PS D:\TestEnterprise\TestEnterpriseFront> ng b --watch
- PS D:\TestEnterprise\TestEnterpriseFront> dotnet run
- powershell
- powershell

- Se recomienda primero levantar el front y luego el back. En el terminal del back nos aparecerá una URL daremos clic a esa URL y se nos abrirá la pantalla que tengamos asignado por defecto en las rutas de nuestro proyecto angular, en mi caso del login.

```

PROBLEMS 12 OUTPUT TERMINAL DEBUG CONSOLE

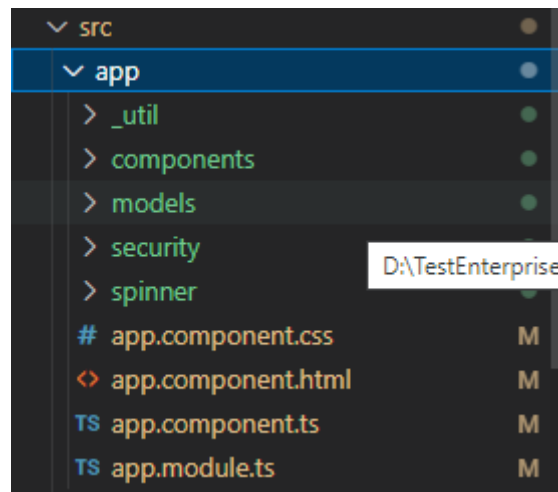
styles.css | styles | 146.75 kB
polyfills.js | polyfills | 33.13 kB
runtime.js | runtime | 1.14 kB
Initial Total | 783.36 kB

Build at: 2023-03-06T02:21:02.648Z - Hash: ac57708e0d226bebb9f6 - Time: 35241ms

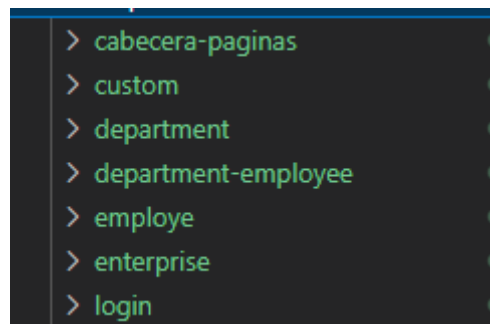
D:\TestEnterprise\TestEnterpriseFront\Security\Services\Us
erService.cs(34,45): warning CS8603: Posible tipo de valor
devuelto de referencia nulo. [D:\TestEnterprise\TestEnter
priseFront\Test_Enterprise_Back.csproj]
Hosting environment: Development
Content root path: D:\TestEnterprise\TestEnterpriseFront
Now listening on http://localhost:5056
Application started. Press Ctrl+C to shut down.

```

- Estructura de carpetas del proyecto Angular : La estructura que considere es la siguiente:



- \_UTIL: aquí almacenaremos los archivos que podremos utilizar en todo el proyecto , dentro de estos tendremos métodos como formateos de fechas, validares personalizados, etc.
- Components: dentro de esta tenemos la estructura principal del proyecto ya que se encuentran los componentes de departamentos , trabajadores, empresas, logeo, y la cabecera.




- Models: tendremos dos archivos que van a servir de referencia para la respuesta y parámetros de respuesta del token cuando nos hayamos logeado con éxito o no.

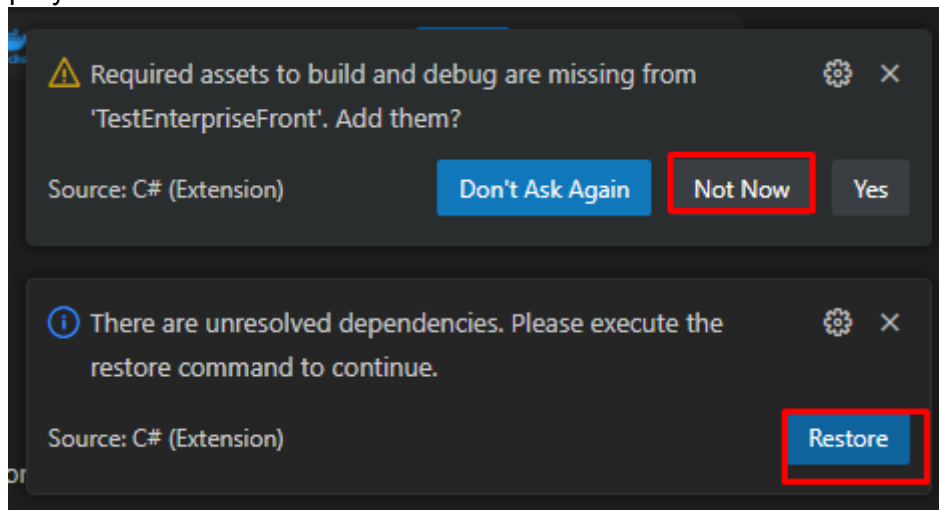
- Security: tendremos los archivos interceptores y el archivo que va a generar el authGuard.
- Spinner: finalmente el spinner contendrá el componente que va a permitir generar un spinner de espera cuando exista un tiempo entre el envío de petición del front al back y la respuesta devuelta del back al front, con esto el usuario puede saber que aún está cargando su pantalla.

## COMO LEVANTAR EL PROYECTO

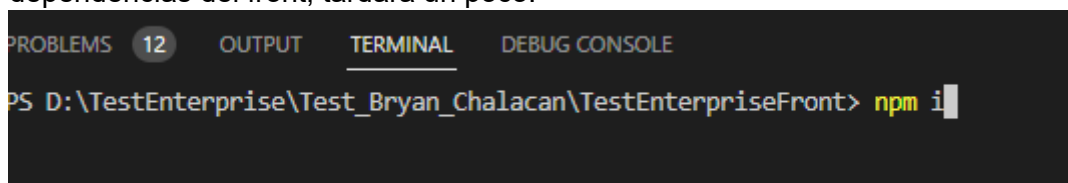
- El Proyecto lo podemos descargar de GIT y se enviará por correo en un comprimido

|                                                                                                           |                  |                |        |
|-----------------------------------------------------------------------------------------------------------|------------------|----------------|--------|
|  TestEnterpriseFront.rar | 05/03/2023 22:06 | Archivo WinRAR | 177 KB |
|-----------------------------------------------------------------------------------------------------------|------------------|----------------|--------|

- Vamos a descomprimirlo y a abrirlo con visual studio code , nos aparecerán dos mensajes que no debemos ignorar el primero colocaremos NOT NOW y en el otro colocaremos RESTORE eso ayudara a restaurar las dependencias del proyecto de back:



- Vamos a abrir un terminal y ejecutamos “npm i” lo cual ayudara a instalar las dependencias del front, tardará un poco.



- Nos aparecerá unos mensajes de vulnerabilidad pero son debido a que algunas librerías están obsoletas:

```
PROBLEMS 12 OUTPUT TERMINAL DEBUG CONSOLE

4"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: esbuild-android-arm64
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for
":arm64"} (current: {"os":"win32","arch":"x64"})

added 1339 packages from 1167 contributors and audited 1363 packages

104 packages are looking for funding
  run `npm fund` for details

found 17 vulnerabilities (2 low, 2 moderate, 10 high, 3 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
PS D:\TestEnterprise\Test_Bryan_Chacalan\TestEnterpriseFront>
```

- Abrimos dos terminales y ejecutamos por separado los siguientes comandos primero levantamos el front y luego el back:

```
PROBLEMS 12 OUTPUT TERMINAL DEBUG CONSOLE
PS D:\TestEnterprise\Test_Bryan_Chacalan\TestEnterpriseFront> ng b --watch
PS D:\TestEnterprise\Test_Bryan_Chacalan\TestEnterpriseFront> dotnet run
```

- Una vez que termina de cargar el front veremos un mensaje como este:

```
PROBLEMS 12 OUTPUT TERMINAL DEBUG CONSOLE
✓ Index html generation complete.

Initial Chunk Files | Names          | Size
main.js             | main           | 602.35 kB
styles.css           | styles         | 146.75 kB
polyfills.js         | polyfills      | 33.13 kB
runtime.js           | runtime        | 1.14 kB

| Initial Total | 783.36 kB

Build at: 2023-03-06T03:22:46.422Z - Hash: e5de003482f17d567
: Generating browser application bundles (phase: sealing)...
✓ Browser application bundle generation complete.
```

- Cuando termina de levantar el back veremos un mensaje como el de abajo y daremos click en en la url:

```
PS D:\TestEnterprise\Test_Bryan_Chacalan\TestEnterpriseFront> dotnet run
Compilando...
Hosting environment: Development
Content root path: D:\TestEnterprise\Test_Bryan_Chacalan\TestEnterpriseFront
Now listening on: http://localhost:5056
Application started. Press Ctrl+C to shut down.
```