b

UNIVERSIDAD TÉCNICA NACIONAL

SEDE DEL PACÍFICO

INGENIERIA EN TECNOLOGÍAS DE LA INFORMACIÓN

TECNOLOGÍAS Y SISTEMAS WEB ll

SEBASTIÁN ALBERTO MATA ORTEGA

WALTER ALONSO ALPÍZAR ARGÜELLO

ll CUATRIMESTRE

2023



**INTRODUCCIÓN**

Dentro de este documento encontraremos información relevante acerca del curso de Tecnologías y Sistemas Web ll, donde nos enfocaremos en la parte del back-end empleando diferentes tecnologías que nos facilitarán desarrollar aplicativos webs, velar por la seguridad e integridad del usuario que utiliza el aplicativo, los diferentes formatos de intercambio de información, la manera en la que se envían, procesan y se acceden los datos. De esta manera los usuarios que necesiten de un servicio puedan tener acceso a la información del servicio que brinda el aplicativo web.

**Semana 1: Tecnologías para creación de API’s**

**Introducción a Frameworks y otras tecnologías**

**JavaScript & Frameworks**

JavaScript es un lenguaje de programación tipado de manera dinámica, el cual se ejecuta en el lado del cliente y permite interactuar con los elementos de la página web.

**Características:**

-**Interactividad en tiempo real**: JavaScript permite que las páginas web respondan a las acciones del usuario en tiempo real.

-**Versatilidad**: : Javascript es un lenguaje muy versátil que se puede utilizar tanto en el lado del cliente como en el lado del servidor.

-**Facilidad de aprendizaje**: Javascript es relativamente fácil de aprender para aquellos que ya tienen conocimientos de programación.

-**Amplia comunidad y Recursos**: Javascript cuenta con una comunidad activa de desarrolladores en todo el mundo.

**TypeScript**

Este es un superconjunto de javascript el cual es tipado de manera estática, en donde las variables que se establezcan van a trabajar como el tipo de variable que se le indique; INT, FLOAT, STRING, etc.

**Herramientas y Frameworks populares**

Hoy en día existen distintos tipos de tecnologías las cuales nos permiten juntar los diferentes algoritmos y servicios que elaboremos, específicamente dentro del ecosistema de **JavaScript** encontramos muchos **frameworks**, los cuales nos permiten realizar aplicaciones web más complejas e intuitivas.

Algunos de los más utilizados son:

* React: Biblioteca especializada en construir interfaces de usuario dentro del lenguaje de JS.
* Angular: Enfocado en la creación de aplicativos webs en JS.
* Vue.js: Framework progresivo para construir interfaces de usuario.
* Node.js: Entorno de ejecución de JavaScript de parte del servidor.
* Express: Tiene como objetivo crear aplicaciones web y APIs en Node.js.

**Ventajas de TypeScript sobre JavaScript**

* Detección de errores temprana: El tipado estático de TypeScript permite detectar errores de tipo durante la etapa de desarrollo, lo que ayuda a evitar errores comunes antes de ejecutar el código.
* Mantenibilidad y escalabilidad: TypeScript facilita el mantenimiento y la evolución de proyectos a largo plazo, especialmente en aplicaciones complejas, gracias a su soporte para clases, interfaces y otras características de programación orientada a objetos.
* Mejor herramienta de desarrollo: TypeScript proporciona una experiencia de desarrollo más rica, con autocompletado, sugerencias de errores y refactorización de código más avanzada en comparación con JavaScript puro.
* Compatibilidad con JavaScript: Dado que TypeScript es un superconjunto de JavaScript, se puede utilizar en proyectos existentes, ya que todo el código JavaScript válido también es válido en TypeScript.

**Node.js**

Este es un entorno de tiempo de ejecución (runtime) de JS basado en el motor V8 de Chrome. A diferencia de JavaScript, el cual se ejecuta sobre el navegador, node.js es ejecutado sobre la parte del servidor, ya que Node.js permite que las funciones de JS puedan interactuar con el servidor para realizar todo tipo de tareas. De igual manera, este posee un modelo de programación basado en eventos y en I/O sin bloqueo, permitiéndole manejar diversas solicitudes de manera simultáneamente sin bloquear el hilo principal de ejecución. Otras características de Node.js son las siguientes:

1. **Amplio ecosistema de módulos y paquetes:** Se cuenta con un gestor de paquetes llamado **npm (Node Package Manager)**, el cual permite gestionar una amplia variedad de módulos.
2. **Desarrollo rápido y productivo:** Permite un desarrollo efectivo y rápido debido a la ligeres por naturaleza y a su capacidad para ejecutar varias tareas en paralelo.
3. **Escalabilidad y rendimiento:** El hecho que permite desarrollar diferentes tareas en paralelo y al modelo sin bloqueo influye en el hecho que le permite a las empresas instalarla dentro de sistemas muy concurrentes por una gran cantidad de usuarios.

**Angular**

Este es un Framework de desarrollo de aplicaciones web, basado en TypeScript, creado por Google. Proporciona un conjunto de herramientas que nos permiten crear aplicativos webs de mayor magnitud, escalables y de alto rendimiento.

**Características Principales.**

Angular posee **Componentes y Directivas** el cual se ayuda para poder crear la arquitectura basada en componentes, en donde cada **componente** es un bloque de construcción importante para la aplicación. Cada componente tiene su propio código. De igual manera Angular ofrece un poderoso sistema de **Binding** que permite mantener sincronización entre los datos y la interfaz de usuario. Así mismo, Angular cada que ves que se necesita crear una nueva instancia este posee **Inyección de dependencias** con el objetivo de reutilizar código, evitando el agotamiento en memoria, y finalmente tenemos que angular como tal posee una función de **Enrutamiento**, el cual nos permite establecer una navegación entre los diferentes componentes y vistas del trabajo al definir rutas y asociar componentes a estas rutas.

**TypeOrm**

TypeOrm es un ORM que puede correr sobre Node.js, el navegador, Ionic, entre otros. Tiene como objetivo brindar soporte las características más recientes de JS, a su ves que provee nuevas características que nos ayudarán en el desarrollo de las aplicaciones.

**Características**

TypeOrm posee un Soporte multiplataforma: TypeORM es compatible con una variedad de bases de datos relacionales, como **MySQL, PostgreSQL, SQLite, Microsoft SQL Server y Oracle**. Esto permite a los desarrolladores elegir la base de datos que mejor se adapte a sus necesidades. TypeORM facilita la comunicación entre la base de datos y los objetos de la aplicación mediante el **mapeo de tablas y columnas** de la base de datos a clases y propiedades de TypeScript, TypeORM proporciona herramientas para crear y **modificar esquemas de base de datos** de forma programática. También admite migraciones de esquemas, lo que facilita la actualización y mantenimiento de la estructura de la base de datos a medida que la aplicación evoluciona.

Algunas otras Características:

* **Consultas y Relaciones**
* **Soporte para transacciones y caching**
* **Integración con Express y otros framworks**

**MySQL**

MySQL es un sistema de gestión de bases de datos relacionales (RDBMS) muy popular y ampliamente utilizado. Es de código abierto y está basado en el modelo de datos relacional, lo que significa que almacena datos en tablas estructuradas con filas y columnas.

**Características**

1. **Escalabilidad y Rendimiento:** MySQL está diseñado para manejar grandes volúmenes de datos y es altamente escalable. Puede manejar múltiples conexiones y consultas concurrentes.
2. **Lenguaje SQL:** : MySQL utiliza el lenguaje SQL (Structured Query Language) como su lenguaje de consulta principal. SQL proporciona una sintaxis estándar para interactuar con la base de datos
3. **Soporte Multiplataforma:** : MySQL es compatible con una amplia gama de plataformas, incluyendo Windows, macOS y varias distribuciones de Linux
4. **Alta disponibilidad y replicación :** MySQL ofrece opciones para alta disponibilidad y replicación de datos. Mediante la configuración de réplicas, puedes crear copias de tus datos y distribuir la carga entre varios servidores.

**Semana 2: Proyecto de Typeorm**

Durante esta clase se aprendió acerca de como podemos modificar nuestro proyecto hecho en typescript para poder elaborar un aplicativo web de manera que atienda las necesidades del servicio que estemos brindando. Para realizar la creación de nuestro api, vamos a utilizar diferentes tecnologías, como las son:

* **TypeOrm:** Es una herramienta la cual nos permite realizar conexiones entre las tecnologías que vayamos a emplear, en este caso Node.js, TypeScript, express, entre otras.
* **Express:** Framework que posee conexiones con Node.js para realizar las funciones respectivas a nuestro aplicativo web.
* **TypeScript:** Lenguaje de tipo tipado estático, que nos servirá para poder realizar las diferentes consultas y funciones.
* **Node.JS:** Trabaja como backend y proporciona todas las funciones que van detrás del servidor.

Los pasos para crear nuestro proyecto son:

1. Instalar Node.js en nuestro dispositivo
2. Abrir ventana de cmd para instalar los paquetes correspondientes.
3. “CMD”: npm install typeorm –save
4. “CMD”: npm install reflect-metada –save
5. “CMD”: npm install @types/node --save-dev
6. “CMD”: npm install mysql –save
7. “CMD”: npx typeorm init --name MyProject --database postgres
8. “CMD”: cd MyProject
9. “CMD”: npm install
10. “CMD”: npm install express –save
11. “CMD”: npm start
12. “CMD”: code . (Para ejecutar nuestro proyecto dentro de visual Studio
13. Abrir una terminal dentro de VS para instalar los paquetes de node.modules:
14. En la terminal: npm i cors helmet jsonwebtoken class-validator bcryptjs
15. Luego: npm i -D @types/cors @types/jsonwebtoken @types/express @types/node @types/bcryptjs typescript ts-node-dev

(nota):Comandos para restablecer el usuario de git y realizar un push tranquilamente

git config --global --unset user.name

git config --global --unset user.email

git config --global --unset credential.helper

cmdkey /delete:LegacyGeneric:target=git:https://github.com

git config --global user.name “username”

git config --global user.email “email”

Ahora que tenemos nuestro proyecto montado deberíamos de ver algo así:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteA partir de aquí cuando tenemos nuestro proyecto de prueba podemos establecer nuestra propia conexión hacia nuestra base de datos preferida dentro de nuestro paquete de data.source:

Podemos modificar los parámetros de nuestra base de datos correspondiente. Para que nuestro proyecto cumpla nuestras necesidades, debemos tener en cuenta que nuestro api se maneja mediante rutas, las cuales nos permitirán extraer la información en base a las rutas o caminos donde estarán los métodos que harán los debidos request (get, post, put or delete). Donde dichos request o funciones van a estar preparados dentro de un archivo controlador que nos brindará la información con base a la ruta que se requiere.

**Semana 3: CRUD Methods**

Durante esta semana se realizaron los debidos métodos mediante los cuales vamos a realizar diferentes acciones, ya sea: mostrar la información general, mediante un filtro de ID, Insertar información, actualizar la información y borrarla. Debemos de tener en cuenta las necesidades de nuestro servicio. En el proyecto pasado creamos las rutas mediante las cuales vamos a acceder a las diferentes acciones:

Texto

Descripción generada automáticamente

Cada acción de http lleva consigo una ruta especifica y en algunos casos, trae consigo un identificador de búsqueda único para realizar acciones específicas. Cabe Recalcar que estos métodos apuntan hacía un controlador del cliente, en donde se encuentran todas las acciones.

Cada uno de estos hace una conexión hacia la base de datos creando una constante la cual va a obtener la información de la base de datos que estemos empleando, para después buscar la información dentro de ella y almacenarla dentro de otra constante.



Texto

Descripción generada automáticamenteDespués de buscar la información requerida podemos realizar diferentes validaciones con respecto a la base de datos, y a la información que se está intentando mandar o validar los datos que se brindan para poder realizar una búsqueda filtrada.

Cabe recalcar que estas validaciones traen consigo ciertos mensajes ya sea de representar errores o cuando se ejecutan las acciones como deberían de ser. Ya que se obtiene la información mediante un método get, cuando se requiere insertar nueva información se deben hacer muchas más validaciones, las cuales llamamos: **“Reglas de entrada”**, después de esto se validan lo que viene siendo las **reglas de negocio**, las cuales son propias de cada empresa con la que se vaya a trabajar. Para poder insertar dicha información debemos de mandar los detalles de lo que necesitamos en una manera concreta, ya que esto se trabaja mediante archivos de **JSON**, y dentro del código descomponer el objeto de JSON en pequeños pedazos para poder realizar las necesidades; ya sea: realizar las validaciones individuales o la inserción como tal de dicha información.

**Semana 4:**

Verificiación de Datos de entrada dentro del controlador de clientes
Durante esta semana se mejoraron los cruds para la entidad de cliente, donde se las validaciones de datos de entrada se reemplazaron por el uso de la librería de class-validator.

Variaciones de Class validator


De igual manera, se implementó dentro del proyecto todo lo relacionado para poder realizar las debidas relaciones entre las tablas según corresponde la lógica.

Semana 5:

Para poder realizar el proceso de inicio de sesión y de creación de usuarios se deben de realizar diversos pasos de autenticación para poder ingresar dentro de la aplicación que estemos creando, primero se debe de ingresar los usuarios a la base de datos, donde se establece un nombre de usuario y la contraseña, que además esta debe de poseer seguridad de encriptación para proteger la información.

Texto

Descripción generada automáticamente

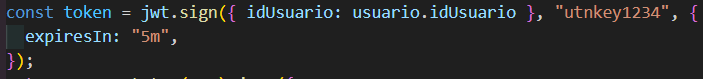
Mediante Esta función podemos encriptar las contraseñas a partir del usuario que creamos

Texto

Descripción generada automáticamente

Una ves encriptada la contraseña, debemos de comparar la contraseña que el usuario coloca con la que esta encriptada dentro de la base de datos

Una vez creado dicho usuario, se debe de realizar la parte de seguridad del log in, donde se establece una sesión para cada usuario mediante un tocken que le permitirá interactuar con el aplicativo y que este se le este reiniciando con dicha ventana establecida cuando el usuario realiza una acción.



Código para poder brindar un token, mediante una ventana que durará 5 minutos y que será desencriptada mediante el key que le brindemos

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

Código para poder generar un nuevo token cada vez que el usuario este realizando una acción dentro del aplicativo

Texto

Descripción generada automáticamenteAsí mismo, debemos de entender que puede ser que el usuario que ingrese tenga permiso de ingresar, pero debemos de saber hasta que punto tiene permitido dicho usuario para interactuar con el aplicativo.

Con dichas líneas de código podemos realizar la acción previamente mencionada, mandando los respectivos mensajes de error u de acceso restringido.

**Semana 6:**

Las relaciones entre tablas nos ayudan a poder mejorar la lógica de nuestro código, permitiendo así alterar las reglas de negocio o de entrada y acomodando las necesidades del producto con todo lo necesario para que este funcione según las necesidades del usuario final. En este caso tenemos una factura, la cual dentro de ella contiene los productos que el cliente desea comprar, sin embargo, para poder realizar la correcta relación debe de existir una tabla intermediaria llamada: DetalleFactura, la cual sale mediante el debido proceso de normalización de las tablas de bases de datos. Teniendo esto en cuenta observemos como podemos realizar la conexión entre estas 3 entidades (Factura, Producto y Detalles) para poder realizar las acciones requeridas de lo que vendría ser el proceso realizar la factura.

Las tres entidades deben de representarse de la siguiente manera: Diagrama

Descripción generada automáticamente

Como mencionamos anteriormente, la entidad detalle\_factura va a contener los detalles que van a tomar juego dentro del proceso de compra como:

* ¿Qué productos se compraron?
* ¿Cuánto de ese producto de compró?

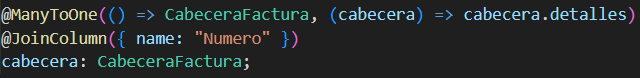
Permitiendo tener la información mucho más ordenada para poder realizar las inserciones de los productos y el hecho de realizar las facturas.

A continuación, veremos cómo se aplica dentro de TypeScript:

-Dentro de la entidad de Factura o Cabecera de Factura declaramos todas los decoradores necesarios dentro de dicha entidad, agregando además una relación con la entidad detalle de factura la cual es de tipo OneToMany()-ManyToOne() de la siguiente manera:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



Con el código anterior realizamos la relación entre estas dos entidades estableciendo el campo Numero como llave compuesta dentro de la entidad detalle. Mediante la propiedad Cascade permitimos que cuando se inserte la llave primaria dentro de la factura, este se coloca automáticamente dentro de la entidad de Detalles, al igual que la actualización de este campo.

Cuando se realiza el llamado de una factura para poder consultarla, dentro del controlador de factura debemos de llamar a los detalles de la factura dentro del mismo llamado de la factura haciendo uso del cascade para llamarlas a las dos como un todo y poder desplegar la información como lo es debido.

Texto

Descripción generada automáticamente

Con el código anterior mandamos a llamar dentro de nuestra base de datos a la entidad de factura, y a su vez dentro de este necesitamos encontrar las relaciones que esta posee para poder devolver la información requerida, cabe recalcar que dentro de las relaciones debemos especificar las necesarias para lo requerido.

**Semana 7:** Introducción a Angular

Angular se emplea principalmente para la parte visual del frontend de las páginas web, a la vez que permite realizar las debidas conexiones con lo que sería el backend, la cual es la encargada de realizar la conexión con el servidor y devolver la información correspondiente. Angular se basa en:

* Componentes
* Directivas
* Módulos
* Servicios

**Decoradores**

Un decorador permite dotar funcionalidades y responsabilidades dinámicamente a objetos. Donde los decoradores nos permiten extender dichas funciones con otra función, sin alterar la función principal, cabe recalcar que estos decoradores se forman a partir de la unión de HTML y las clases de TypeScript y estos se identifican con el “@” como indicio de un decorador.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

La estructura de un decorador de angular es:

● selector: es nombre de una nueva etiqueta.

● templateUrl: es la ruta del archivo HTML.

● styleUrls: es un arreglo de las rutas de los archivos CSS.

**Componentes**

Un componente define la funcionalidad a una sección de html, donde podemos incorporar diversos componentes dentro de un mismo modulo para poder realizas los llamados correspondientes. Los componentes son cualquier cosa relacionada con html que se le necesite mostrar al usuario final dentro de nuestro aplicativo, un componente se encuentra formado por 4 aspectos muy importantes:

* **HTML**
* **CSS**
* **Clase TypeScript**
* **Clase TypeScript Test**

La estructura de la clase de un componente está divida en:

1. Importaciones de módulos.

2. Decorador

3. Clase

Ejemplo:

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente**

Ejemplo html del componente:

Imagen que contiene Forma

Descripción generada automáticamente

**Directivas**

Una directiva básicamente es una plantilla dinámica de html Son instrucciones que alteran la estructura mediante la adición, eliminación y sustitución del DOM.

Las directivas estructurales más comunes son:

* ngif
* ngfor
* ngswitch

Ejemplo de directiva:

Interfaz de usuario gráfica, Texto, Aplicación, Teams

Descripción generada automáticamente

Directiva para correr una lista:

Interfaz de usuario gráfica

Descripción generada automáticamente

**Módulos:**

Los módulos dentro de angular son la principal herramienta que nos permite realizar la gran parte de las tareas dentro de un proyecto montado sobre angular, las ventajas de estos son:

* Organizar la aplicación de una mejor manera.
* Poseen una escalabilidad excelente
* El mantenimiento de estos es relativamente sencillos

**NgModules:**

NgModule es el módulo principal de la aplicación. Es una clase con el decorador @NgModule. Ayuda a organizar una aplicación en bloques de funcionalidad. Las propiedades más importantes son:

1. imports: se importan los módulos para la aplicación.
2. declarations: se declaran los componentes, pipes y directivas creados.
3. providers: exporta todos los servicios para el uso general en la aplicación.
4. bootstrap: es la vista principal de la aplicación, aloja todas las vistas de la aplicación.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteEjemplo de NgModule

**Servicios:**

Estos usualmente son una clase con una funcionalidad especifica, Los servicios pueden ser:

* Servicios de datos.
* Servicios de registros.
* Configuraciones de la aplicación.

Estos servicios son los que les brindan las funcionalidades a los componentes dentro de nuestro proyecto.

Ejemplo de uso de un servicio:



**Inyección de dependencias:**

Cuando se esta trabajando en proyecto que involucre el hecho que los usuarios estén realizando creaciones de objetos de manera continua, angular nos permite poder realizar este tipo de inyecciones para no tener que estar consumiendo tantos recursos del servicio y evitar en la medida de lo posible la latencia cuando se pone en pie la aplicación. La inyección de dependencias es un método que administra la instancia de una clase. La mayoría de las dependencias son servicios, los cuales proporcionan nuevos servicios a los componentes.

Un Inyector mantiene los servicios en propiedades de la clase. ● Si un servicio no está instanciado en la clase, el inyector crea una nueva instancia del servicio requerido

Imagen que contiene Escala de tiempo

Descripción generada automáticamenteEscala de tiempo

Descripción generada automáticamente con confianza media**Arquitectura de Angular**

Diagrama

Descripción generada automáticamente con confianza media

**Semana 8**: VACACIONES!

**Semana 9:**

Dentro del proyecto de angular que creamos desde cero, se van a realizar los siguientes trabajos o archivos para consumir una API de internet:

* Creación de Servicios dentro de angular
* Modelos para el consumo de una API
* Un módulo de personajes
* Las rutas respectivas para los personajes

Con el siguiente comando podemos crear un servicio dentro de nuestro proyecto:



Después, con el siguiente comando crearemos un modulo de personajes dentro de la carpeta de Personajes:



Dentro de la carpeta de src se creará una carpeta donde se almacenarán los modelos para los personajes:

Texto

Descripción generada automáticamente

Info.ts:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamentePersonaje.ts:

PersonajesResponse.ts, donde se hace el llamado de info y de personajes como un array para almacenar los personajes, para poder llamar dentro del constructor del componente de personajes y mostrar b:

Texto

Descripción generada automáticamente

Dentro de personajes.service vamos a realizar la conexión con la API de uso de Rick&Morty, esto se logra mediante un constructor y mediante la librería de HttpClient:

Texto

Descripción generada automáticamente

Luego, dentro del componente de personaje vamos a crear un constructor para personajes, donde vamos a almacenar los resultados del llamado anterior del uso de la api de Rick&Morty, donde este actúa como un subscriptor de la API, en espera de la información:

Texto

Descripción generada automáticamente

Finalmente, dentro del archivo html de personajes donde se carga cada objeto tendremos que establecer con “\*ngFor” podremos cargar los resultados que devuelve personaje, además de cargar individualmente, al igual que la manera de cargar las imágenes que contiene dicha api:

Texto

Descripción generada automáticamente

**Semana 10:**

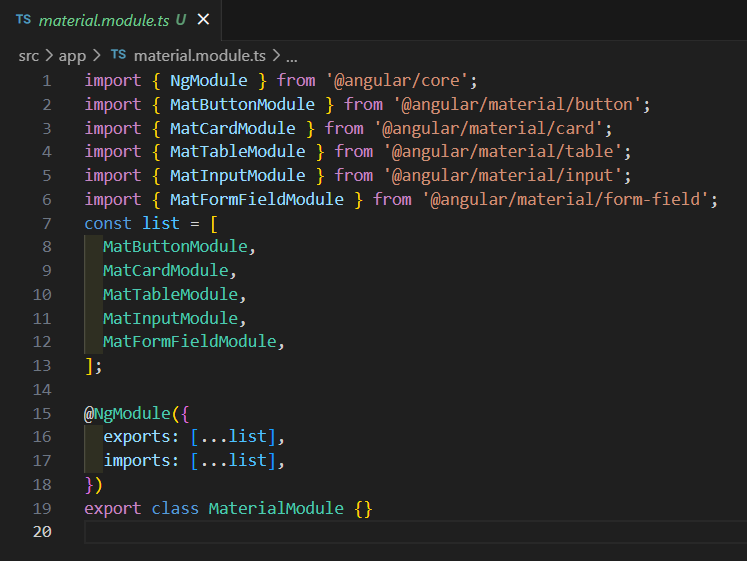
Durante esta semana se realizó uso del proyecto backend que se inicializó a principios del curso, en donde su funcionalidad principal era actuar como los servicios de una tienda, en donde se podían: Agregar Productos, Clientes, Proveedores y generar una factura final con los detalles de la compra. Por consiguiente, se implementará dicho proyecto sobre otro proyecto de Angular, donde se le brindará una interfaz gráfica a los servicios que ofrece dicho servidor, con el objetivo de poner en ejecución al mismo tiempo ambos proyectos.

Primero, dentro del proyecto de angular vamos a agregar Angular Material para poder darle una interfaz agradable a nuestra página, y además veremos como funciona angular material y sus diferentes modelos. Para agregar angular material a nuestro proyecto deberemos de abrir una terminal y ejecutar el siguiente comando:



Cuando comience la instalación deberemos de elegir el tema de colores que nos brinda angular material. Consultar página web: <https://material.angular.io/>

Una ves instalado dentro de nuestro proyecto, deberemos de realizar la conexión hacía nuestra api con el http client. Cabe recalcar que cada componente de angular material debe de ser importado hacía nuestro proyecto de angular, para efectos de un mejor manejo de estos componentes crearemos un archivo en la raíz de app, casi al mismo nivel que app module para así poder reutilizarlos dentro de nuestro proyecto a nuestro antojo.

Quedaría de la siguiente manera:

De igual manera, se creó un servicio para lo que sería los productos del backend del proyecto, y un constructor como se aprendió la semana anterior para así poder cargar todo lo relacionado con el producto desde el backend, hacía el frontend.

Ejemplo:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente