**Manual**

**Arquitectura y Proyecto Base**



|  |  |
| --- | --- |
| **Elaborado por:** | Javier Lusarreta |
| **Versión:** | 1.0 |

CONTROL DE CAMBIOS

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Autor | Observación del cambio |
| 21/12/2020 | 1.0 | Javier Lusarreta | Versión inicial del manual |

|  |
| --- |
|  |
|  |

INDICE DE CONTENIDO

[1. Situación 3](#_Toc59446273)

[2. Objetivo 3](#_Toc59446274)

[3. Arquitectura 3](#_Toc59446275)

[3.1 Gateway – API 3](#_Toc59446276)

[3.2 Angular 3](#_Toc59446277)

**Copyright EFOR:** Este documento, y las informaciones que contiene, son propiedad exclusiva de EFOR. Las informaciones suministradas son las últimas disponibles en el momento de la publicación. Toda reproducción, copia a publicación de este documento, incluso parcial, y con cualquier fin, está prohibida sin el acuerdo escrito de EFOR.

# Situación

En Efor se tiene una arquitectura nueva y un nuevo proyecto base que hay que entender para trabajar de manera correcta.

# Objetivo

Poder entender esta metodología se ha elaborado esta documentación muy sencilla de lo que supone la nueva arquitectura.

Hay documentación más elaborada, pero esta supone una primera lectura y aproximación a los desarrolladores de Efor.

# Arquitectura

La nueva arquitectura se basa…

## Gateway – API

Gateway se encarga de conectar las partes de back y front de la aplicación. Gateway contiene los controladores que recibirán peticiones HTTP, se comunicaran con los servicios de la aplicación y devolverán una respuesta.

## Angular

Se encarga de generar la parte visual de la aplicación web. Dentro de la aplicación de angular podemos encontrar distintos componentes que corresponderian a cada página o plantilla de páginas de la web. Cada componente contiene:

Component.html: Contiene la vista que se mostrará en la página del componente.

Component.scss: Contiene la hoja de estilos que se aplicará tanto a la página del componente como a las páginas de los componentes contenidos dentro de él.

Component.module.ts: Contiene las rutas definidas de la vista y las librerías que importa necesarias para el código en el fichero TS.

Component.ts: Contiene la parte de negocio de la vista, donde se programarán las funciones y eventos de la misma.

La comunicación entre las variables de typescript y el HTML se realiza de la siguiente forma:

(): Salida (de HTML a TS).

[]: Entrada (de TS a HTML).

[()]: Bidireccional.

Dentro de la aplicación de Angular podemos encontrar el archivo package.json, ubicado en la raíz del proyecto y el cual contendrá las librerías usadas y los comandos ejecutables en la consola para llamar a los scripts.

Dentro de la carpeta src se encuentra el archivo styles.scss que contiene todos los estilos comunes a todas las vistas, también una carpeta app donde se guardan los componentes de cada vista y una carpeta assets donde se guardan recursos como imágenes o los archivos i18n para la traducción de literales.

El proceso de ejecución se iniciará con el usuario descadenando un evento, por ejemplo de clic. Este evento estará asociado a un método del componente. En este método se llamará al servicio que corresponda dentro de api/services. El servicio constará de un código autogenerado que se comunicará con la capa Gateway por medio de una llamada HTTP, esperando una respuesta. Gateway se comunicará con los servicios que se encuentran dentro de la capa Bussiness, la cuál se encargará de comunicarse con la base de datos.

COMANDOS ÚTILES

npm run build -> Ejecuta el proyecto de Angular

npm run swagger -> Genera la documentación que comunica Angular con C#

Crear un módulo nuevo -> cd ruta\_de\_la\_pagina -> ng g c nombre\_de\_la\_pagina

## SQL

Contiene las sentencias SQL que interactúan con la base de datos para proporcionarlos a la capa Bussines. En este proyecto no se implementa al usarse una conexión Oracle.

## Bussiness

Contiene la capa de negocio de la backend. Conecta Gateway con la base de datos.

Incluye una serie de servicios los cuales tienen: un automapper, al cual se le indican que clases se pueden mapear unas con otras, una interfaz que define los métodos que tendrá el servicio y el propio servicio que se encarga de implementar los métodos y se comunica con la base de datos (SELECT/INSERT/UPDATE/DELETE).

La clase interceptor que tiene cada proyecto dentro de la capa business, hereda de DbConnectionInterceptor, y sirve para recoger los datos de la conexión del cliente a la base de datos.

## Data

Capa que conecta con la base de datos y proporciona la información a la capa de Bussiness. Contiene un SSMContext autogenerado, que hereda de BDContext y que especifica la estructura con la que se generarán las entities.

El servicio SSMContext que hereda de BDContext, que está dentro de esta capa, y que sirve para generar la estructura de la base de datos, las tablas. También hereda de una clase ubicada dentro de la capa Crosscutting, que nos brindará un apoyo y además, crea la conexión con la base de datos, es este caso, Oracle.

## Infraestructure

Capa que almacena las entities de la base de datos como los objetos DTO con los que se trabajará.

Las entities creadas albergarán las propiedades de cada campo, tales como el tipo de dato, las longitudes de ese dato, si es requerido o no, el nombre de la columna, etc.

## Crosscutting

Contiene librerías que son utilizadas en los diferentes servicios de la capa Bussiness. Añade funcionalidades extra.

Dentro de crosscutting están las librerías usadas en los diferentes servicios. Entre estas librerías están las de Auth que contiene unos controladores para el acceso, así como el login, el logout. Además, esta librería contiene las políticas requeridas. También se incluyen servicios de autenticación propios creados por la empresa, un control de roles...

## Logging

Crea un listado de logs con los procesos realizados en la ejecución.

