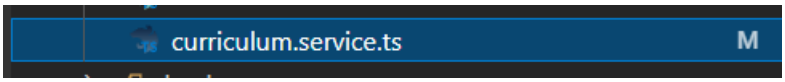
INTITUTO TECNOLOGICO DE PATRIMONIO Y TURISMO "YAVIRAC"

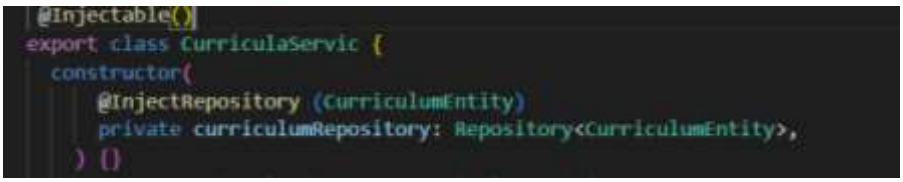
NOMBRE: Tania Paola Acosta Calle
CURSO: 5to "A"

TEMA: DOCUMENTACIÓN DE SERVIDORES CONTROLADORES, MODELOS, ENTIDADES, DTO

El archivo service se va a encargar se va a encargar de comunicar con el backed



Lo que se realiza es un injectable el cual se inyecta una dependencia



Metodos:

Se crea lo que son los metodos y el nomnre de nuestra tabla en este caso tenemos el Create

Remove FindALl , FindOne y delete

Se realiza la comunicacon con el componente el cual se realiza mediante el contructor

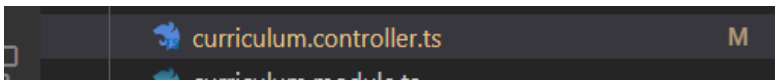
Y se coloca frente de cada metodo que sea asincronico y espere una respuesta el cual se realiza con cada metodo respectivo.

```
) {}
async create(payload: CreateCurriculumDto) {
  const newCurriculum= this.curriculumRepository.create(payload);

  return await this.curriculumRepository.save(newCurriculum);
}

async remove(id: number) {
  return await this.curriculumRepository.softDelete(id);
}

async findAll() {
  return await this.curriculumRepository.find();
}

async findOne(id: number) {
  const curriculum = await this.curriculumRepository.findOne({
    where: {
      id: id,
    },
  });
```

```
  if (curriculum === null) {
    throw new NotFoundException('El producto no se encontro');
  }

  return curriculum;
}

async update(id: number, payload: UpdateCurriculumDto) {
  const curriculum = await this.curriculumRepository.findOne({
    where: {
      id: id,
    },
  });

  if (curriculum === null) {
    throw new NotFoundException('El producto no se encontro');
  }

  this.curriculumRepository.merge(curriculum, payload);

  return this.curriculumRepository.save(curriculum);
}
}
```

Controladores

Responde las solicitudes que se realiza al servidor



Para esto tenemos lo que son decoradores el cual se inicializa con un arroba cual agregan funcionalidades

Los nombres de las rutas en el controlador debe ir siempre en plural

Y tiene una clase el cual llamamos como **curriculaController** y tienemos un constructor

El cual nos sirve para dar un valor a la clase .

El el sigueinte codigo tenemos un decorador que es un metodo el cual es el metodo GET

El decorador @Httpcode() me va a permitir modificar el estatus de la respuesta

El la sigueinte linea tenemos un @Query el cual nos sirve para enviar un parametro el cual se hace mediente con el signo de **?**

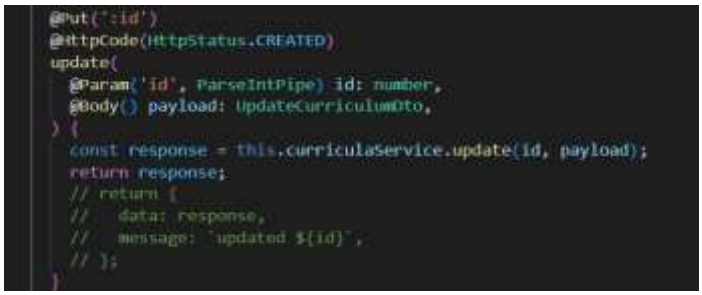
tenemos una variable const que tiene un valor como response el cual hace una referencia con **this.CurriculaService** y el metodo **FINDALL()** el cual devuelve una lista de cadena.

Tenemos lo mismo de la anterior linea pero si con la diferencia del metoto @param el cual envia parametros con : y algo podemos enviar en nuestro cliente por ejemplo {{host}}/curriculum/algo.

-tenemos el metodo FindOne el cual nos sirve para buscar un objeto en especifico

Tenemos el metodo **@Post** el cual nos sirve para actualizar **@Httpcode()** me va a permitir modificar el estatus de la respuesta en este caso tenemos como create **@Body** es el cuerpo y el **payloa**d nos va a servir como una variable y guardar toda la data

Tenemos el método @Put que es para actualizar y es casi similar al código anterior



Tenemos el método @Delete el cual nos sirve para eliminar en este caso esta el ID para nosotros colocar mediante el método @param que id vamos a borrar.

MODULO CURRICULUM

El import nos sirve para importar las funciones que han sido exportadas desde un moduli externo

@module que es un decorador

El cual tenemos lo que son clases de JavaScript el cual se nos crea ya por defecto

Y tenemos una clase con nombre CurricularModule

ENTITIES

Las entidades nos sirven para mapear y poder conectar entre la base de datos y el backend en cual podemos crear actualizar borrar y consultar.

El cual tenemos un decorador con nombre de @entity y el nombre el plural y con una clase

En este caso tenemos un ID porque es incrementable y tiene de tipo NUMBER

Tenemos un decorador entonces estamos diciendo que es una @PrimaryGeneralColumn el cual convierte en dos cosas en una primerary fk y que sea incrementable

```
@Entity('curricula')
export class CurriculumEntity {

@PrimaryGeneratedColumn()
id:number;
```

Tenemos que importar un decorador el cual es el @column y el nombre de nuestra tabla

Y colocar que tipo de dato es en nuestra base de datos en mi caso yo tengo como string y seguimos haciendo con los demás campos de la tabla

```
  @Column('varchar', {
    name: 'code',
    length: 255,
    default: 'SN',
    comment: 'Nombre del producto',
  })
  code:string;
```

Aquí colocamos los decoradores con los nombres respectivos campos de la base de datos y colocando el tipo de dato

DTO

Me van a servir para validad la data que viene en el cuerpo de la petición el cual se inicializa con un decorador y colocamos dependiendo los campos de nuestra tabla que estemos trabajando.

Tenemos las 10 validaciones

El cual digo que sea de tipo String

Que sea mínimo de 2 y máximo de 100

Que sea sea opcional y que no sea alfanumérico Que sea de tipo type , Y que no sea una dirección