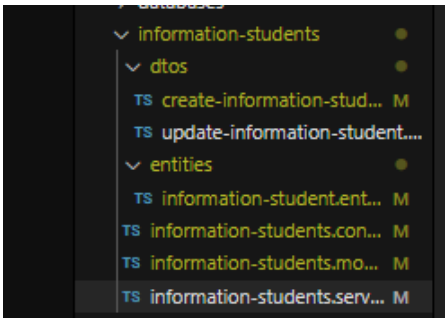
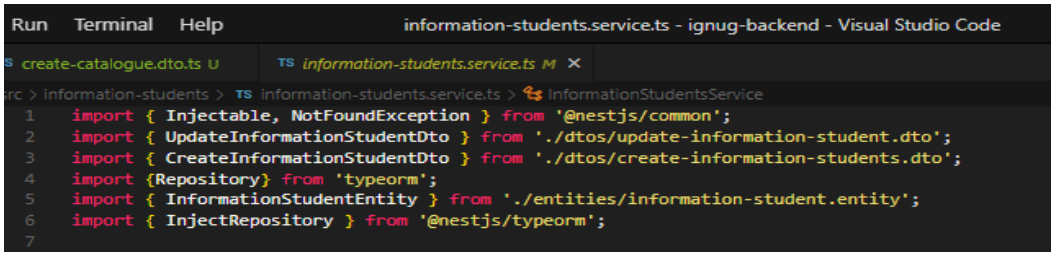
Carrera	Asignatura	Datos del estudiante	fecha
Tecnología en Desarrollo de Software	Tendencias Actuales de Programación	Apellidos: Quito Tibán	26/06/2022
		Nombres: Jhonatan Alejandro	Docente: Ing. Hernan Mejia



Information_students.service.ts

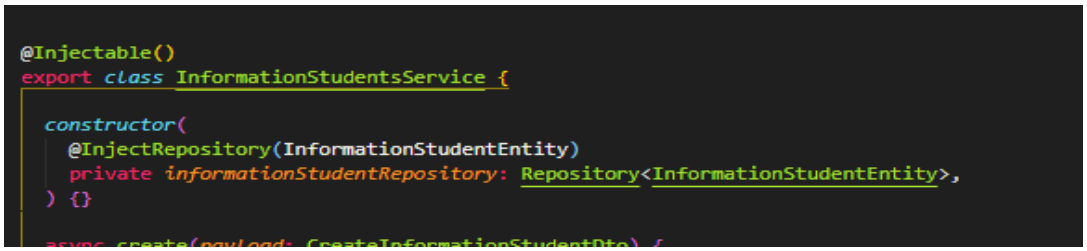
El documento service.ts sirve para poder contarnos a la base de datos para que así nos permitirá realizar un CRUD.

1.- Las librerías usadas nos ayudan a la facilidad de la interacción con el lenguaje.



1.1.- Una inyección de dependencias que a su vez nos permitirá disminuir el acoplamiento entre los componentes de una aplicación. La inversión de dependencias suele también conocerse como inversión de control.

1.2.- Los constructores son funciones o métodos que permiten realizar tareas de instanciación de objetos. Cuando un objeto es creado a partir de una clase, se llama al constructor que se encargará de inicializar los atributos del objeto.



Carrera	Asignatura	Datos del estudiante	fecha
Tecnología en Desarrollo de Software	Tendencias Actuales de Programación	Apellidos: Quito Tibán	26/06/2022
		Nombres: Jhonatan Alejandro	Docente: Ing. Hernan Mejia

1.3.- Una función declarada como **async** significa que el valor de retorno de la función será, "por dentro de javascript", una Promesa. Si la promesa se resuelve normalmente, el objeto-promesa retornará el valor.

La palabra clave **await** recibe una Promesa y la convierte en un valor de retorno. Usamos await, JavaScript esperará hasta que finalice la Promesa. Si se completa con éxito, el valor obtenido es retornado.

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

async findOne(id: number) {
  const student = await this.informationStudentRepository.findOne({
    where: {
      id: id,
    },
  });

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

  return student;
}

async update(id: number, payload: UpdateInformationStudentDto) {
  const student = await this.informationStudentRepository.findOne({
    where: {
      id: id,
    },
  });

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

  this.informationStudentRepository.merge(student, payload);

  return this.informationStudentRepository.save(student);
}

remove(id: number) {
  return  this.informationStudentRepository.softDelete(id);
}
}
```

Information_students.module.ts

2. 1.- Las librerías usadas nos ayudan a la facilidad de la interacción con el lenguaje.

```
information-students > TS information-students.module.ts > InformationStudentsModule
import {Module} from '@nestjs/common';
import { InformationStudentsController } from './information-students.controller';
import { InformationStudentsService } from './information-students.service';
import { InformationStudentEntity } from './entities/information-student.entity';
import { TypeOrmModule } from '@nestjs/typeorm';
```

Carrera	Asignatura	Datos del estudiante	fecha
Tecnología en Desarrollo de Software	Tendencias Actuales de Programación	Apellidos: Quito Tibán	26/06/2022
		Nombres: Jhonatan Alejandro	Docente: Ing. Hernan Mejia

2.2.- Un módulo es una clase anotada con un decorador. El decorador proporciona metadatos que Nest utiliza para organizar la estructura de la aplicación. @Module()

```
@Module({
    imports: [TypeOrmModule.forFeature([InformationStudentEntity])],
    controllers:[InformationStudentsController],
    providers:[InformationStudentsService],
    exports:[InformationStudentsService, TypeOrmModule],

})
```

2.3.- En Nest, la propiedad *export* en un @Module define qué *Providers* serán expuestos hacia el módulo que lo está importando, también llamado *enquirer*.

```
})
export class InformationStudentsModule{}
```

Information_students.entity.ts

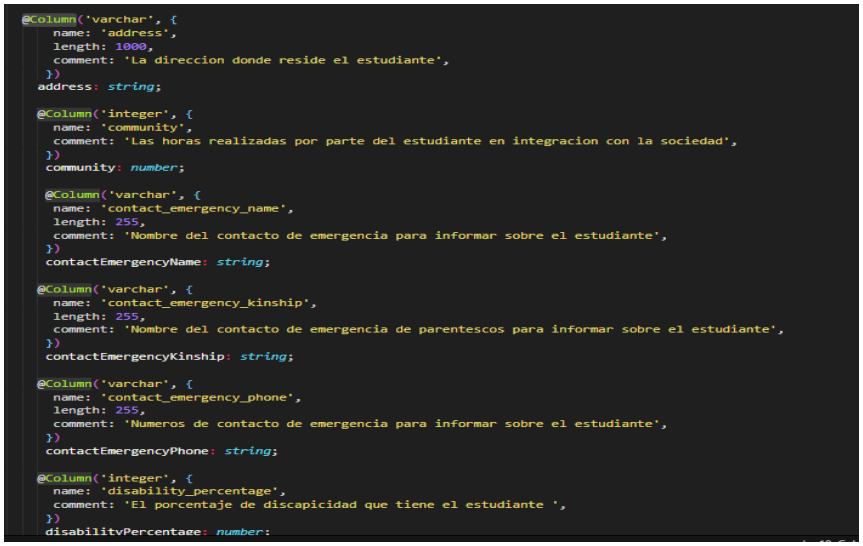
3.1.- Entity sirve para administrar donde nace la importancia de la anotación @Entity, esta anotación se debe de definir a nivel de clase y sirve únicamente para indicarle a JPA que esa clase es una Entity

3.2.- @PrimaryGeneratedColumn() crea una columna principal cuyo valor se generará automáticamente. Creará una columna int con incremento automático / serie / secuencia / identidad (dependiendo de la base de datos y la configuración que se la proporcionada).

```
@Entity('information_students')
export class InformationStudentEntity{
    @PrimaryGeneratedColumn()
    id: number;
```

Carrera	Asignatura	Datos del estudiante	fecha
Tecnología en Desarrollo de Software	Tendencias Actuales de Programación	Apellidos: Quito Tibán	26/06/2022
		Nombres: Jhonatan Alejandro	Docente: Ing. Hernan Mejia

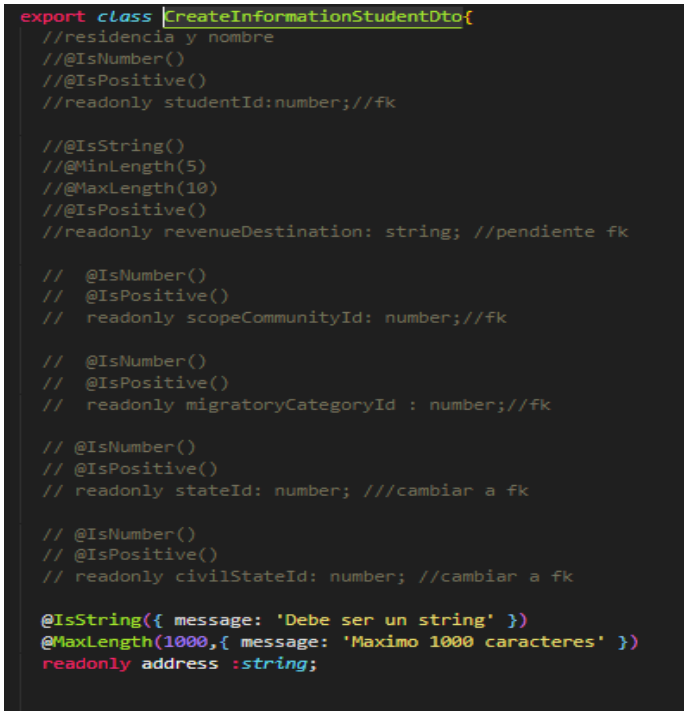
3.3.- La anotación @Column nos permitirá definir aspectos muy importantes sobre las columnas de la base de datos de la base de datos como lo es el nombre, la longitud, constrains, etc.



```
@Column('varchar', {
    name: 'address',
    length: 1000,
    comment: 'La direccion donde reside el estudiante',
  })
  address: string;

  @Column('integer', {
    name: 'community',
    comment: 'Las horas realizadas por parte del estudiante en integracion con la sociedad',
  })
  community: number;

  @Column('varchar', {
    name: 'contact_emergency_name',
    length: 255,
    comment: 'Nombre del contacto de emergencia para informar sobre el estudiante',
  })
  contactEmergencyName: string;

  @Column('varchar', {
    name: 'contact_emergency_kinship',
    length: 255,
    comment: 'Nombre del contacto de emergencia de parentescos para informar sobre el estudiante',
  })
  contactEmergencyKinship: string;

  @Column('varchar', {
    name: 'contact_emergency_phone',
    length: 255,
    comment: 'Numeros de contacto de emergencia para informar sobre el estudiante',
  })
  contactEmergencyPhone: string;

  @Column('integer', {
    name: 'disability_percentage',
    comment: 'El porcentaje de discapicidad que tiene el estudiante ',
  })
  disabilityPercentage: number;
```

Create-information-students.dto.ts

Los DTO (Data Transfer Object) permite la validación de datos que viene del body.



```
export class CreateInformationStudentDto{
  //residencia y nombre
  //@IsNumber()
  //@IsPositive()
  //readonly studentId:number;//fk

  //@IsString()
  //@MinLength(5)
  //@MaxLength(10)
  //@IsPositive()
  //readonly revenueDestination: string; //pendiente fk

  //  @IsNumber()
  //  @IsPositive()
  //  readonly scopeCommunityId: number;//fk

  //  @IsNumber()
  //  @IsPositive()
  //  readonly migratoryCategoryId : number;//fk

  // @IsNumber()
  // @IsPositive()
  // readonly stateId: number; ///cambiar a fk

  // @IsNumber()
  // @IsPositive()
  // readonly civilStateId: number; //cambiar a fk

  @IsString({ message: 'Debe ser un string' })
  @MaxLength(1000,{ message: 'Maximo 1000 caracteres' })
  readonly address :string;
```