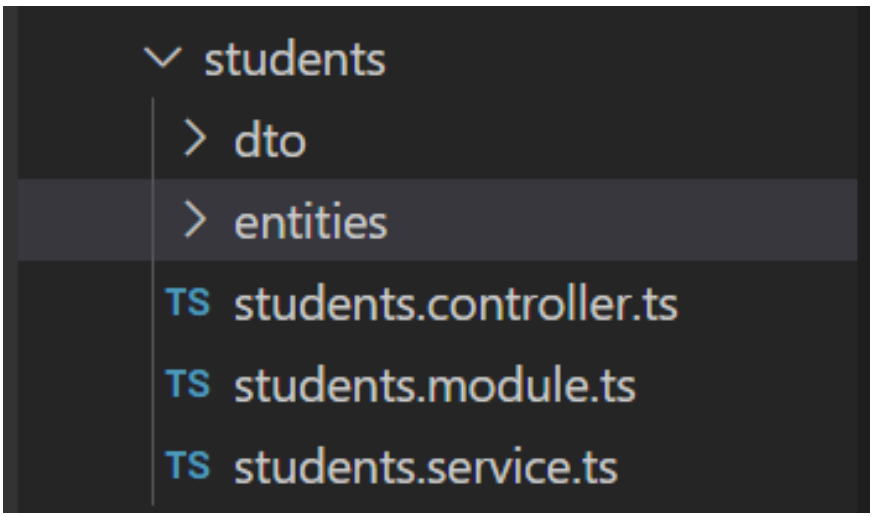
INSTITUTO SUPERIOR TECNOLÓGICO YAVIRAC
Dirección: García Moreno S435 y Ambato
Quito – Ecuador

| Carrera | Asignatura | Datos del estudiante | fecha |
|---|---|---|---|
| Tecnología en Desarrollo de Software | Tendencias actuales de programacion | Apellidos:Motoche Roman | 26/junio/2022 |
| | | Nombres: kevin Joel | Docente: Ing. Hernan Mejia |

# Documentación app NESTJS (backend)




## Students.service.ts

El archivo. service.ts nos sirve para poder conectarnos a la base de datos y este a su vez nos permitirá realizar un CRUD.

1.-Dependencias del archivo o librerías usadas para la factibilidad de interacción con el lenguaje

```
import { Injectable } from '@nestjs/common';
import { CreateStudentDto } from './dto/create-student.dto';
import { UpdateStudentDto } from './dto/update-student.dto';
import { InjectRepository } from '@nestjs/typeorm';
import { StudentEntity } from './entities/student.entity';
import { Repository } from 'typeorm';
import { InformationStudentsService } from '../information_students/information-stud
```

| Carrera | Asignatura | Datos del estudiante | fecha |
|---|---|---|---|
| Tecnología en Desarrollo de Software | Tendencias actuales de programacion | Apellidos:Motoche Roman | 26/junio/2022 |
| | | Nombres: kevin Joel | Docente: Ing. Hernan Mejia |

1.1.-Hacemos una inyección de dependencias que este a su vez con la ayuda de un constructor nos permite crearnos un repositorio aquí es donde uno la obtiene en la conexión de base de datos pero primero necesitaremos una entidad.

@Injectable = Decorador de nestjs.Proveedor asíncrono esperará hasta que Promisese resuelva

```
@Injectable()
export class StudentsService {
  constructor(
    @InjectRepository(StudentEntity)
    private studentRepository: Repository<StudentEntity>,
    private informationStudentsService: InformationStudentsService,

  ) {}
```

1.2.-Create : este método tiene un payload:CreateStudentDto de tipo Dto por que este Dto permite que cualquier ruta que use CreateStudentDto aplicará estas reglas de validación

```
  async create(payload: CreateInformationStudentDto) {
    const newInformationsStudent = this.informationStudentRepository.create(payloa
    return await this.informationStudentRepository.save(newInformationsStudent);

  }
```

| Carrera | Asignatura | Datos del estudiante | fecha |
|---|---|---|---|
| Tecnología en Desarrollo de Software | Tendencias actuales de programacion | Apellidos:Motoche Roman | 26/junio/2022 |
| | | Nombres: kevin Joel | Docente: Ing. Hernan Mejia |

FindOne: Método toma un argumento que representa un id parámetro de ruta extraído

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

  if (informationStudent === null) {
    throw new NotFoundException('La informacion no se encontro');
  }

  return informationStudent;
}
```

Update: Permite modificar los campos pero este a su vez hace uso de un dto llamado UpdateStudentDto y aplicara igual las reglas de validación

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

  if (informationStudent === null) {
    throw new NotFoundException('La informacion no se encontro');
  }

  this.informationStudentRepository.merge(informationStudent, payload);

  return this.informationStudentRepository.save(informationStudent);
 }
}
```

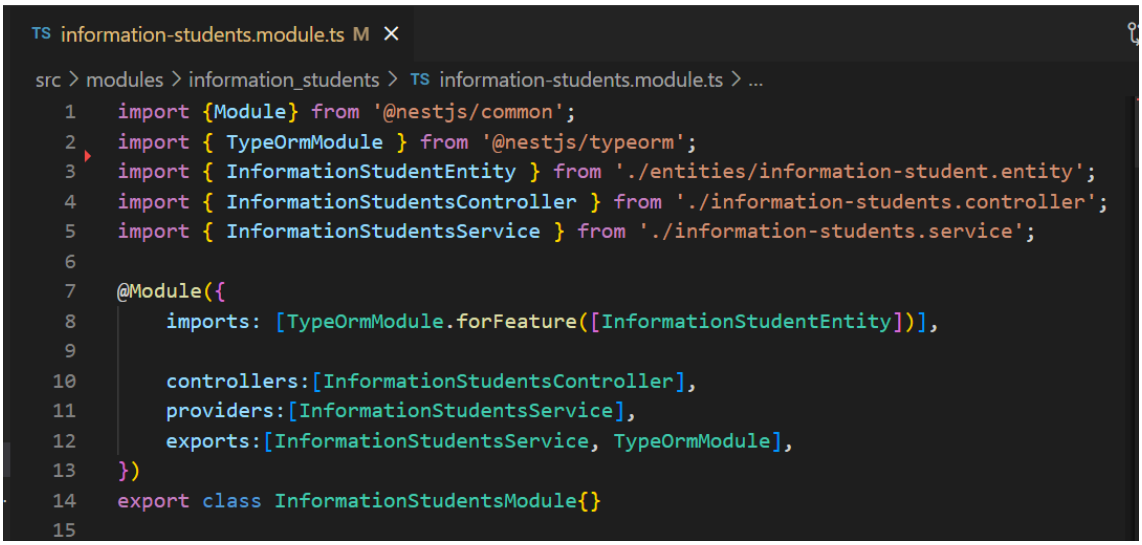
| Carrera | Asignatura | Datos del estudiante | fecha |
|---|---|---|---|
| Tecnología en Desarrollo de Software | Tendencias actuales de programacion | Apellidos:Motoche Roman | 26/junio/2022 |
| | | Nombres: kevin Joel | Docente: Ing. Hernan Mejia |

Delete: Permite borrar un dato que hemos creado

```
async delete(id: number) {
  return await this.informationStudentRepository.softDelete(id);
}
```

## .module.ts

El modulo simplemente organiza el código relevante para una función específica, manteniendo el código organizado y estableciendo límites claros



| Carrera | Asignatura | Datos del estudiante | fecha |
|---|---|---|---|
| Tecnología en Desarrollo de Software | Tendencias actuales de programacion | Apellidos:Motoche Roman | 26/junio/2022 |
| | | Nombres: kevin Joel | Docente: Ing. Hernan Mejia |

## .controller.ts

El controlador permite recibir solicitudes específicas para la aplicación.

1.1 Primero haremos un constructor que se relacione con el service

```
@Controller('information-students')
export class InformationStudentsController {
  constructor(private informationstudentsService:InformationStudentsService ) {}
```

2.2 El @Get()decorador del método de solicitud HTTP antes del findAll()método le dice a Nest que cree un controlador para un punto final específico para las solicitudes HTTP.Este también ayuda para poder buscar solo un id , aquí también definiremos los @Query y @Params . Hacemos uso de los Pipes para validar la ID

```
@Get('')
@HttpCode(HttpStatus.OK)
findAll(@Query() params: any) {
  const response = this.informationstudentsService.findAll();

  return response;
  // return {
  //   data: response,
  //   message: `index`,
  // };
}

@Get(':id')
@HttpCode(HttpStatus.OK)
findOne(@Param('id', ParseIntPipe) id: number) {
  const response = this.informationstudentsService.findOne(id);
  return response;
  // return {
  //   data: response,
  //   message: `show`,
  // };
}
```

| Carrera | Asignatura | Datos del estudiante | fecha |
|---|---|---|---|
| Tecnología en Desarrollo de Software | Tendencias actuales de programacion | Apellidos:Motoche Roman | 26/junio/2022 |
| | | Nombres: kevin Joel | Docente: Ing. Hernan Mejia |

El método post y put envían atraves del body los campos ya validados donde el cliente enviara la data y este a su vez le permitirá un dato o actualizarlo

```
@Post('')
@HttpCode(HttpStatus.CREATED)
create(@Body() payload: CreateInformationStudentDto) {
  const response = this.informationstudentsService.create(payload);
  return response;
  // return {
  //   data: response,
  //   message: `created`,
  // };
}

@Put(':id')
@HttpCode(HttpStatus.CREATED)
update(
  @Param('id', ParseIntPipe) id: number,
  @Body() payload: UpdateInformationStudentDto,
) {
  const response = this.informationstudentsService.update(id, payload);
  return response;
  // return {
  //   data: response,
  //   message: `updated ${id}`,
  // };
}
```

| Carrera | Asignatura | Datos del estudiante | fecha |
|---|---|---|---|
| Tecnología en Desarrollo de Software | Tendencias actuales de programacion | Apellidos:Motoche Roman | 26/junio/2022 |
| | | Nombres: kevin Joel | Docente: Ing. Hernan Mejia |

Delete aquí es donde se destruye un dato a través del id

```
@Delete(':id')
@HttpCode(HttpStatus.CREATED)
delete(@Param('id', ParseIntPipe) id: number) {
  const response = this.informationstudentsService.delete(id);
  return response;
  // return {
  //   data: response,
  //   message: `deleted`,
  // };
}
```



## Entities

Sirve para poder administrar de mejor manera los datos al enviar a la base de datos este a su vez puede definir una entidad y sus columnas directamente en el modelo, utilizando decoradores.

information-student.entity.ts

src > modules > information_students > entities > information-student.entity.ts > InformationStudentEntity

```
  PrimaryGeneratedColumn,
  UpdateDateColumn,
} from 'typeorm';

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

  @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 so
  })
  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 e
  })
```

| Carrera | Asignatura | Datos del estudiante | fecha |
|---|---|---|---|
| Tecnología en Desarrollo de Software | Tendencias actuales de programacion | Apellidos:Motoche Roman | 26/junio/2022 |
| | | Nombres: kevin Joel | Docente: Ing. Hernan Mejia |

## DTO

Son la validaciones de datos que viene del body , aquí hacemos uso de decoradores para dar las respectivas restricciones