Reporte final

Datos eliminados: matrícula, correo, celular, teléfono; con fundamento en lo establecido en los artículos 65, fracción II, 98, fracción III, 113, fracción I y último párrafo de la Ley Federal de Transparencia y Acceso a la Información Pública, 44, fracción II, 106, fracción III y 116 de la Ley General de Transparencia y Acceso a la Información Pública, así como a lo establecido en los Lineamientos Generales en Materia de Clasificación y Desclasificación de la Información, así como para la elaboración de versiones públicas, por tratarse de datos personales concernientes a una persona física identificada, en la modalidad de confidencial. Área Dirección Adjunta de Innovación y Conocimiento. Periodo indefinido, toda vez que no está sujeta a temporalidad alguna y solo podrán tener acceso a ella los titulares de la misma. Aprobada en la Primera Sesión Extraordinaria del Comité de Transparencia de INFOTEC ejercicio 2025.

Índice

Introducción

Durante la estancia en el programa de TSU en Desarrollo de Software, aplicamos conocimientos esenciales como la programación orientada a objetos y utilizamos diversos lenguajes y herramientas para el desarrollo de proyectos. Entre los lenguajes empleados tenemos HTML, CSS, JavaScript, y framework como Bootstrap y Angular. También hicimos uso de tecnologías como SQL, junto con herramientas como Visual Studio Code, Spring Boot Tools, PostgreSQL, Postman y la consola, lo que nos permitió plasmar nuestras habilidades y llevar a cabo desarrollos funcionales.

En conjunto con los lenguajes y herramientas mencionados, desarrollaremos un 'Sistema Gestor de Tareas' basado en una base de datos relacional creada con PostgreSQL. Esta base de datos incluirá cinco entidades y estará conectada a nuestro proyecto frontend desarrollado en Angular. A través de esta integración, implementaremos funcionalidades como consultas, adiciones, actualizaciones y eliminación de tareas, siguiendo el modelo CRUD e incorporando validaciones para garantizar la calidad y consistencia del sistema.

Este proyecto no solo nos permitirá consolidar los conocimientos adquiridos previamente, sino también aprender nuevas habilidades que contribuirán a desarrollar un sistema más robusto y funcional.

Desarrollo

Instalación de Herramientas de trabajo

Para el manejo del código que estaremos utilizando para el frontend, instalaremos Visual Studio Code.

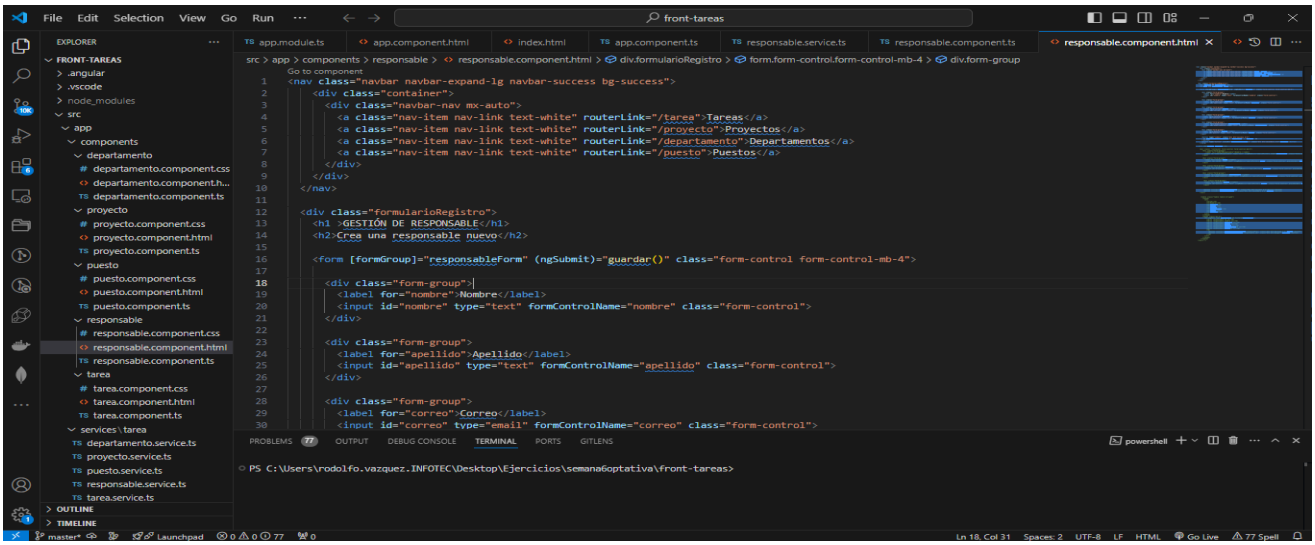


Imagen de elaboración propia.

Para el manejo del código del que estaremos utilizando para el backend, instalaremos Spring Tools.

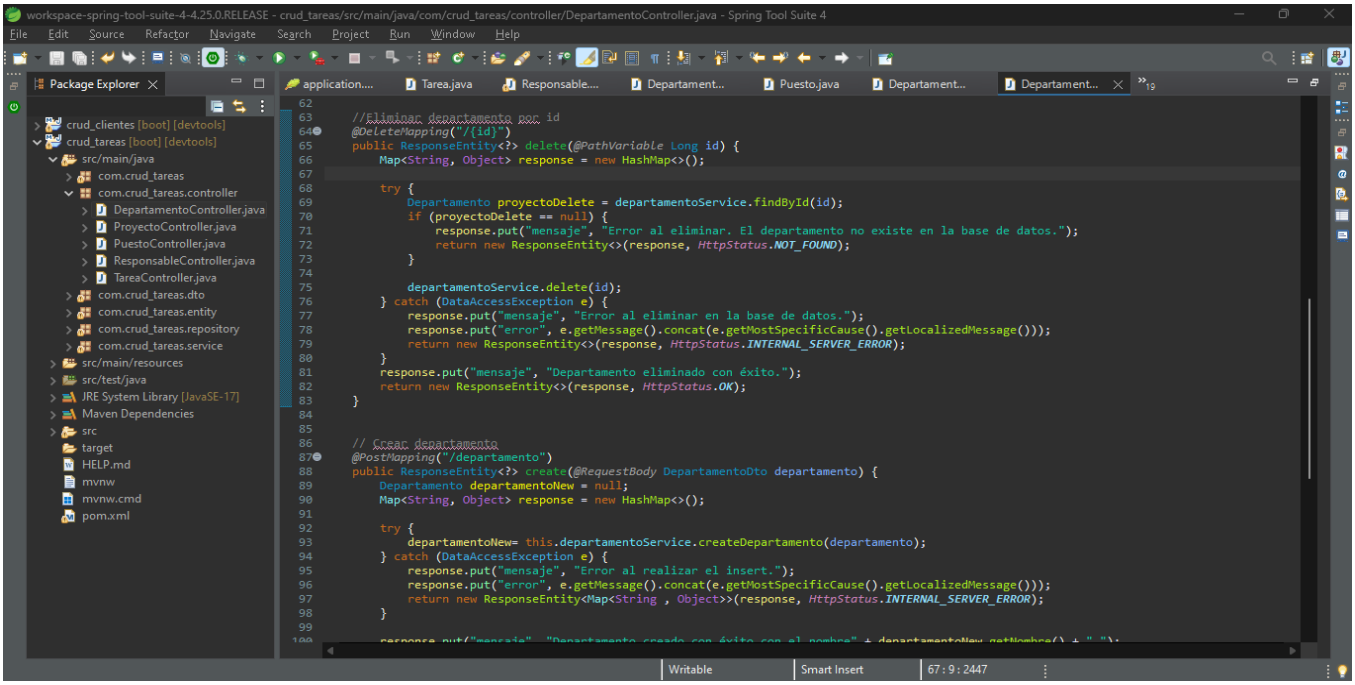


Imagen de elaboración propia.

Para la administración de la base de datos que estaremos utilizando en conjunto con lo generado en el backend, instalaremos pg Admin de PostgreSQL.

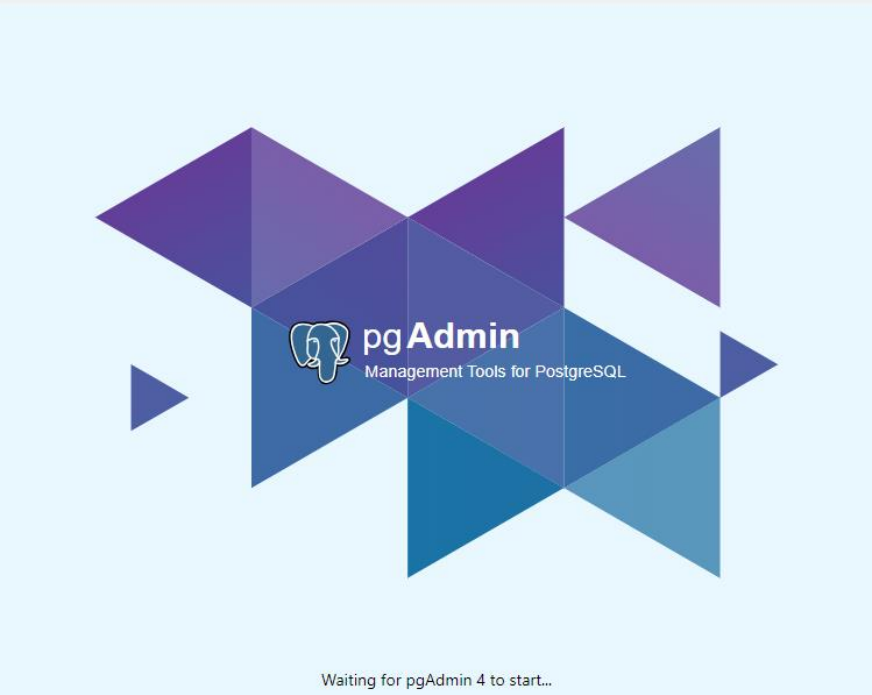


Imagen de elaboración propia.

Para realizar las pruebas de la base de datos que estaremos utilizando Postman.

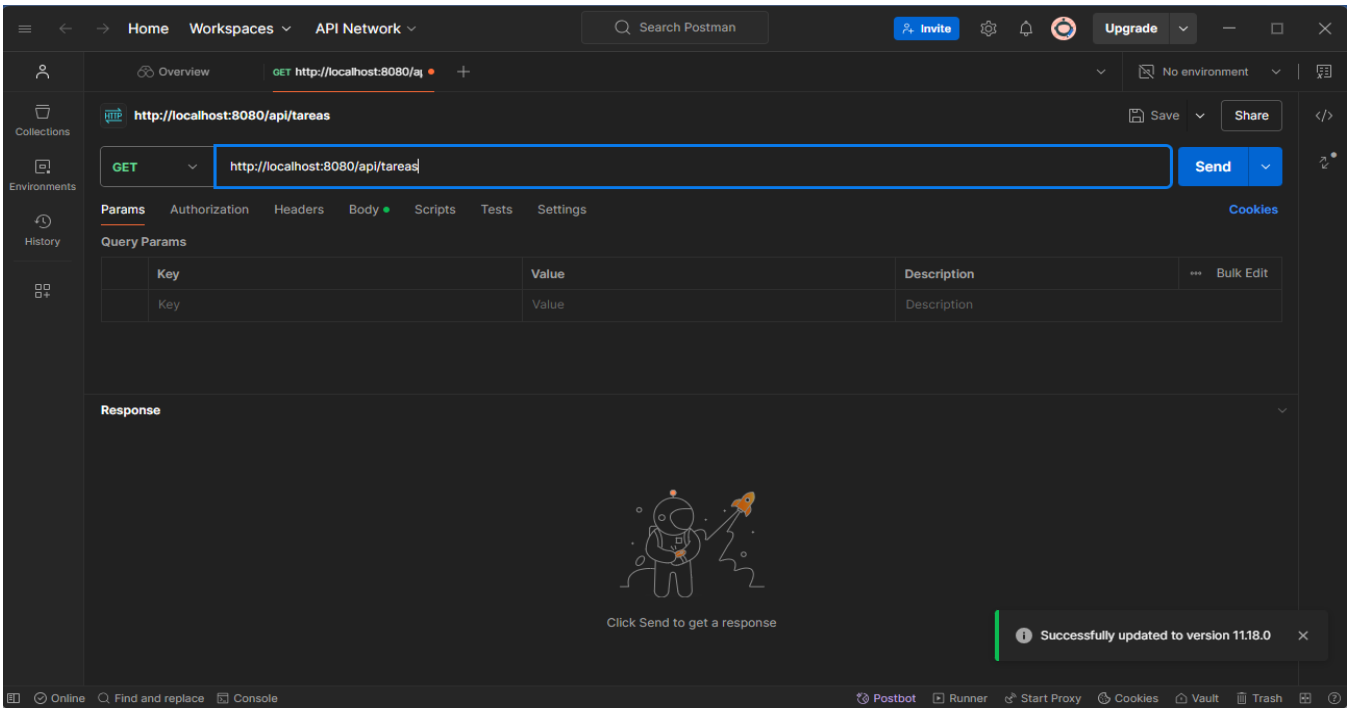


Imagen de elaboración propia.

Iniciaremos con la creación de nuestra base de datos con Spring Tools, en donde inicialmente crearemos nuestro proyecto, dando clic en **File, New y Spring Starter Project**.

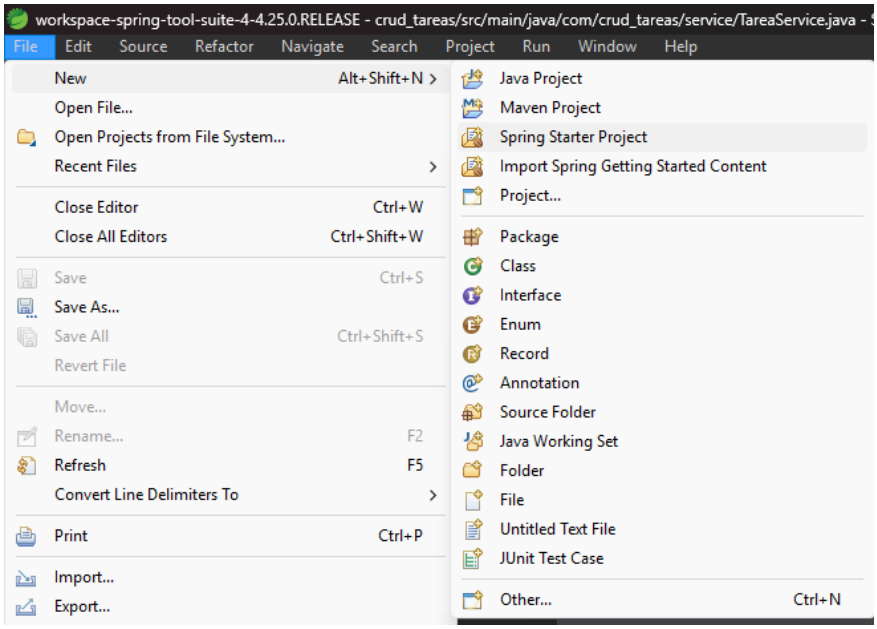


Imagen de elaboración propia.

Ahora le daremos un nombre a nuestro proyecto y daremos clic en **Next**.

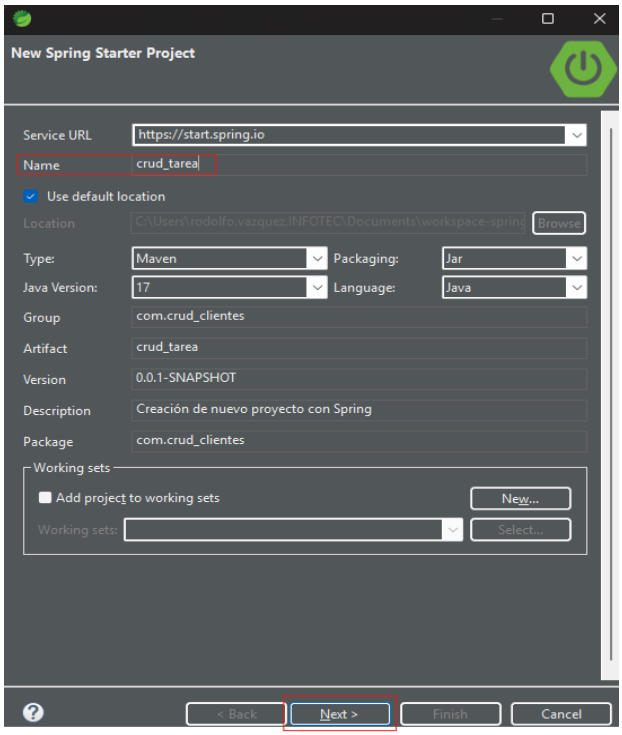


Imagen de elaboración propia.

Seleccionaremos las dependencias que estaremos usando y damos clic en **Finish**.

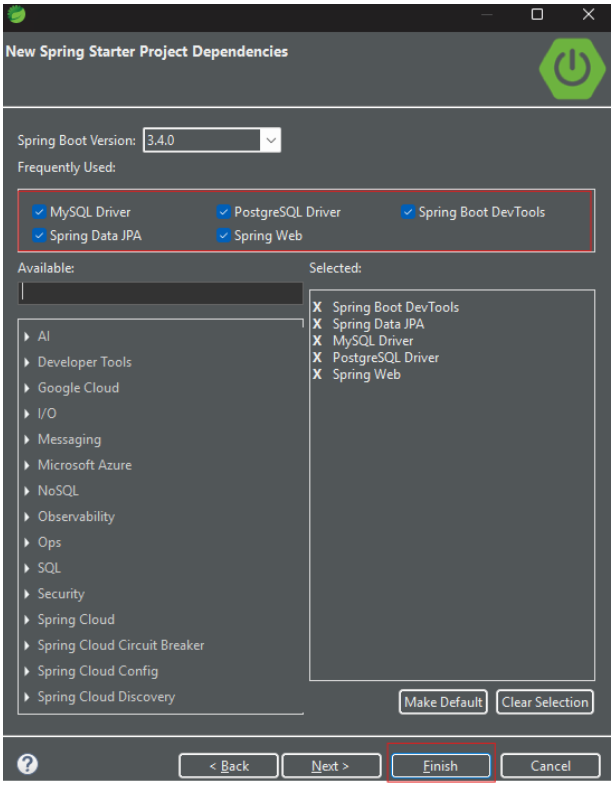


Imagen de elaboración propia.

Una vez creado nuestro proyecto, iniciaremos agregando nuestros paquetes de nuestra base de datos, los cuales serán nuestro paquete **Controller, Dto, Entity, Respository y Service**.

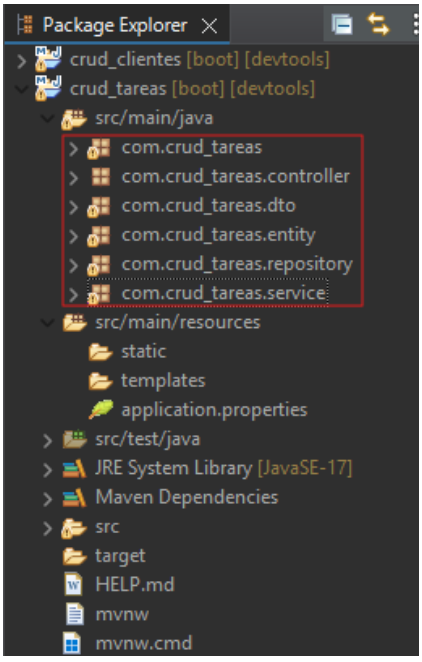


Imagen de elaboración propia.

En este caso estaremos usando 5 entidades (Tarea, Responsable, Proyecto, Departamento y Puesto), por lo que dentro de cada paquete crearemos un archivo por cada entidad correspondiente a cada paquete que generamos.

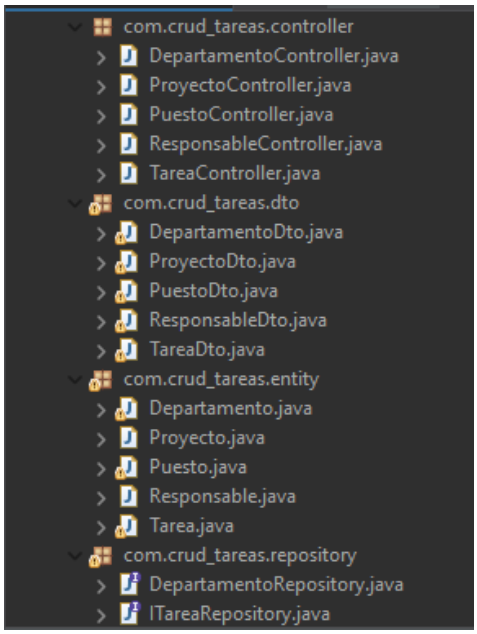


Imagen de elaboración propia.

Bien, una vez que tengamos creados nuestros paquetes y nuestros archivos, iniciaremos con la generación del código de nuestra base de datos.

Package crud_tareas.controller

Tarea.controller

Aquí vamos a declarar la ruta en la que vamos a tener mapeado nuestra página, y en donde declararemos nuestros métodos de búsqueda, creación, actualización y eliminación del CRUD de nuestra base de datos.

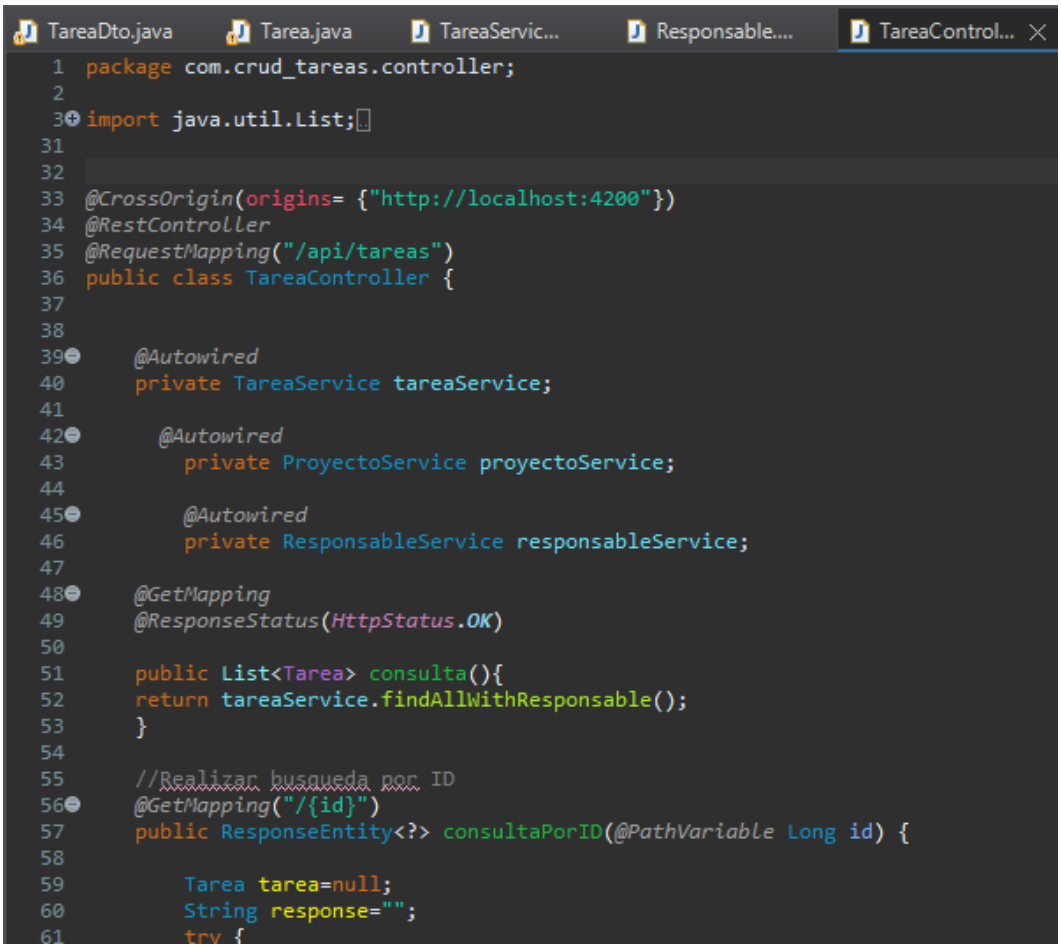


Imagen de elaboración propia.

Package crud_tareas.dto

Tarea.dto

Aquí es en donde vamos a declarar las variables para los campos de nuestras entidades, así como sus correspondientes "getters y setters".

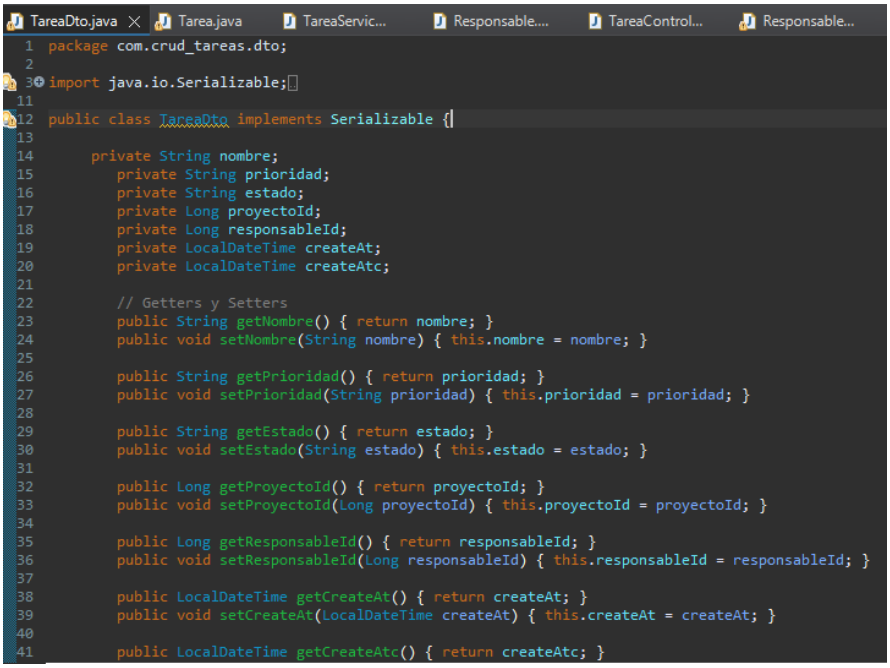


Imagen de elaboración propia.

Package crud_tareas.entity

Tarea.entity

Aquí vamos a definir las columnas de nuestras tablas y los valores que va a estar tomando de acuerdo a nuestros campos declarados, así como las relaciones que se van a tener con otras tablas y que tipo de relación van a tener estas.

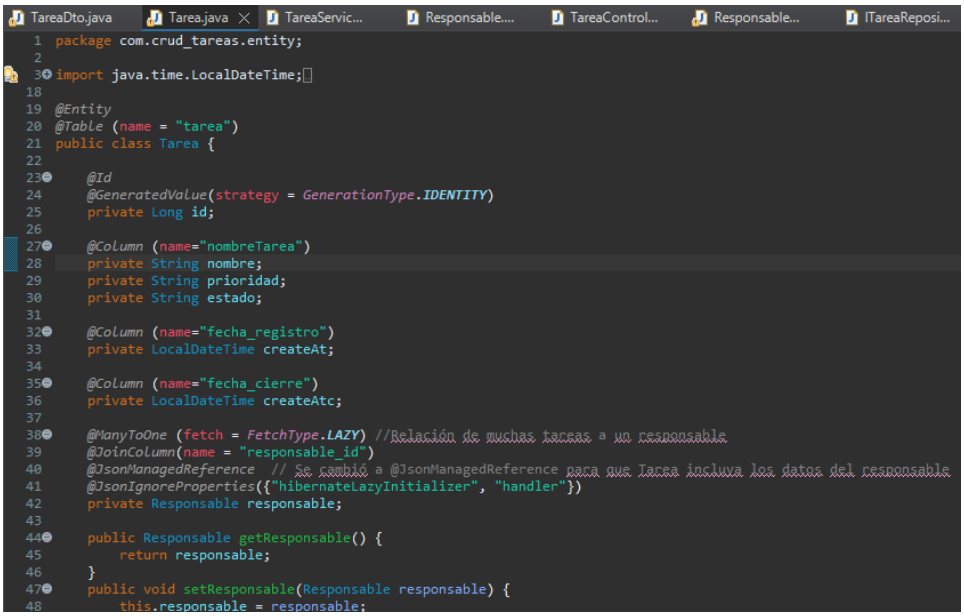


Imagen de elaboración propia.

Package crud_tareas.repository

Tarea.repository

Aquí estamos creando una interfaz como un componente que interactúa con la base de datos, permitiendo que Spring gestione la inyección de dependencias de este repositorio en otras partes de tu aplicación, y permitiéndonos agregar métodos para el manejo de la entidad Tarea (Tarea.entity).

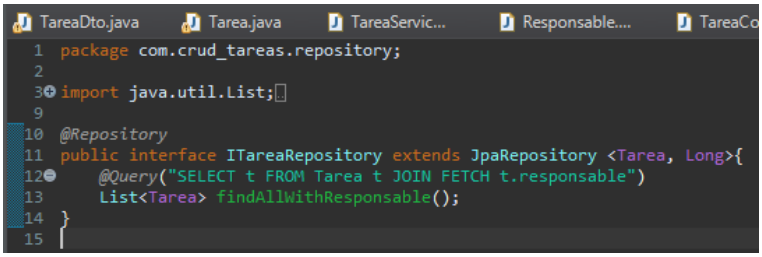


Imagen de elaboración propia.

Package crud_tareas.service

Tarea.service

Aquí nos ayudar a interactuar con el controlador quien es quien maneja las solicitudes y respuestas (solicitudes del cliente), y con el repositorio que es el que interactúa directamente con la base de datos.

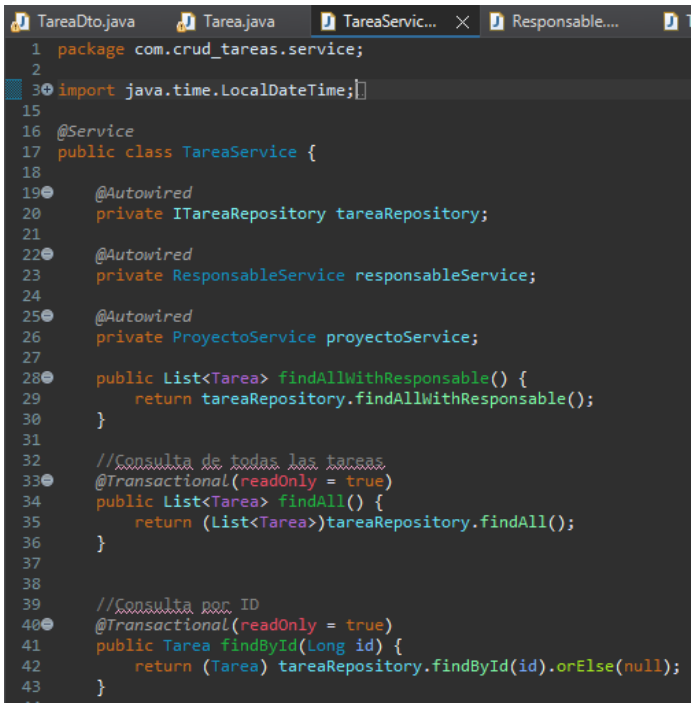


Imagen de elaboración propia.

A continuación, dejare la evidencia del resto de los archivos generados.

Responsable.controller

TareaDto.java | Tarea.java | TareaServic... | Responsable.... | Depar

```java
package com.crud_tareas.dto;

import java.io.Serializable;


public class ResponsableDto implements Serializable {

    private String nombre;
    private String apellido;
    private String correo;
    private String celular;
    private DepartamentoDto departamento;
    private PuestoDto puesto;
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellido() {
        return apellido;
    }
    public void setApellido(String apellido) {
        this.apellido = apellido;
    }
    public String getCorreo() {
        return correo;
    }
    public void setCorreo(String correo) {
        this.correo = correo;
    }
    public String getCelular() {
        return celular;
    }
    public void setCelular(String celular) {
        this.celular = celular;
    }
    public DepartamentoDto getDepartamento() {
        return departamento;
    }
    public void setDepartamento(DepartamentoDto departamento) {
        this.departamento = departamento;
    }
    public PuestoDto getPuesto() {
        return puesto;
    }
    public void setPuesto(PuestoDto puesto) {
```

Imagen de elaboración propia.

Proyecto.controller

```java
package com.crud_tareas.controller;

import java.util.HashMap;

@CrossOrigin(origins= {"http://localhost:4200"})
@RestController
@RequestMapping("/api/proyecto")
public class ProyectoController {

    @Autowired
    private ProyectoService proyectoService;


    //consulta de todos los proyectos
    @GetMapping("")
    @ResponseStatus(HttpStatus.OK)

    public List<Proyecto> consulta(){
    return proyectoService.findAll();
    }

    //consulta de proyecto por Id
    @GetMapping("/{id}")
    public ResponseEntity<?> consultaPorId(@PathVariable Long id) {
        Proyecto proyecto = null;
        String response = "";
        try {
            proyecto = proyectoService.findById(id);
        } catch (DataAccessException e) {
            response = "Error al realizar la consulta: " + e.getMessage();
            return new ResponseEntity<>(response, HttpStatus.INTERNAL_SERVER_ERROR);
        }

        if (proyecto == null) {
            response = "El proyecto con el id " .concat(id.toString()).concat(" no existe en la base de datos");
            return new ResponseEntity<>(response, HttpStatus.NOT_FOUND);
        }

        return new ResponseEntity<>(proyecto, HttpStatus.OK);
    }
```

Imagen de elaboración propia.

Departamento.controller

```java
package com.crud_tareas.controller;

import java.util.HashMap;

@CrossOrigin(origins= {"http://localhost:4200"})
@RestController
@RequestMapping("/api/departamento")
public class DepartamentoController {

    @Autowired
    private DepartamentoService departamentoService;

    //Obtener a todas los departamentos
    @GetMapping("")
    @ResponseStatus(HttpStatus.OK)

    public List<Departamento> consulta(){
        return departamentoService.findAll();
    }

    //Obtener departamento por id
    @GetMapping("/{id}")
    public ResponseEntity<?> consultaPorId(@PathVariable Long id) {
        Departamento departamento = null;
        String response = "";

        try {
            departamento = departamentoService.findById(id);
        } catch (DataAccessException e) {
            response = "Error al realizar la consulta: " + e.getMessage();
            return new ResponseEntity<>(response, HttpStatus.INTERNAL_SERVER_ERROR);
        }

        if (departamento == null) {
            response = "El departamento con el id ".concat(id.toString()).concat(" no existe en la base de datos");
            return new ResponseEntity<String>(response, HttpStatus.NOT_FOUND);
        }

        return new ResponseEntity<Departamento>(departamento, HttpStatus.OK);
    }
```

Imagen de elaboración propia.

Puesto.controller

```java
package com.crud_tareas.controller;

import java.util.HashMap;

@CrossOrigin(origins= {"http://localhost:4200"})
@RestController
@RequestMapping("/api/puesto")
public class PuestoController {

    @Autowired
    private PuestoService puestoService;


    //Busqueda de todos los puestos
    @GetMapping("")
    @ResponseStatus(HttpStatus.OK)

    public List<Puesto> consulta(){
    return puestoService.findAll();
    }


    //Busqueda de los puestos por ID
    @GetMapping("/{id}")
    public ResponseEntity<?> consultaPorId(@PathVariable Long id) {
        Puesto puesto = null;
        String response = "";
        try {
            puesto = puestoService.findById(id);
        } catch (DataAccessException e) {
            response = "Error al realizar la consulta: " + e.getMessage();
            return new ResponseEntity<String>(response, HttpStatus.INTERNAL_SERVER_ERROR);
        }

        if (puesto == null) {
            response = "El puesto con el id ".concat(id.toString()).concat(" no existe en la base de datos");
            return new ResponseEntity<>(response, HttpStatus.NOT_FOUND);
        }

        return new ResponseEntity<Puesto>(puesto, HttpStatus.OK);
    }
```

Imagen de elaboración propia.

Responsable.dto

```java
package com.crud_tareas.dto;

import java.io.Serializable;


public class ResponsableDto implements Serializable {

    private String nombre;
    private String apellido;
    private String correo;
    private String celular;
    private DepartamentoDto departamento;
    private PuestoDto puesto;
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellido() {
        return apellido;
    }
    public void setApellido(String apellido) {
        this.apellido = apellido;
    }
    public String getCorreo() {
        return correo;
    }
    public void setCorreo(String correo) {
        this.correo = correo;
    }
    public String getCelular() {
        return celular;
    }
    public void setCelular(String celular) {
        this.celular = celular;
    }
    public DepartamentoDto getDepartamento() {
        return departamento;
    }
    public void setDepartamento(DepartamentoDto departamento) {
        this.departamento = departamento;
    }
```

Imagen de elaboración propia.

Proyecto.dto

```java
package com.crud_tareas.dto;

import java.io.Serializable;

public class ProyectoDto implements Serializable {

    private String nombre;

    private String descripcion;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

}
```

Imagen de elaboración propia.

Departamento.dto

```java
package com.crud_tareas.dto;

import java.io.Serializable;

public class DepartamentoDto implements Serializable {

    private Long id;

    private String nombre;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
}
```

Imagen de elaboración propia.

Puesto.dto

```java
package com.crud_tareas.dto;

import java.io.Serializable;

public class PuestoDto implements Serializable {

private Long id;

    private String nombre;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

}
```

Imagen de elaboración propia.

Responsable.entity

```java
package com.crud_tareas.entity;

import java.util.List;

@Entity
@Table (name = "responsable")
public class Responsable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column (name="nombreResponsable")
    private String nombre;
    @Column (name="apellidoResponsable")
    private String apellido;
    private String correo;
    @Column(name = "celular")
    private String celular;

        @ManyToOne (fetch = FetchType.LAZY) //Relación de un responsable a un departamento
        @JoinColumn(name = "departamento_id")
        @JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
        private Departamento departamento;

        @OneToOne (fetch = FetchType.EAGER) //Relación de un responsable a un puesto
        @JoinColumn(name = "puesto_id")
        @JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
        private Puesto puesto;

        @OneToMany(mappedBy = "responsable", fetch =FetchType.LAZY) // Relación de un responsable a muchas tareas
        @JsonBackReference  // Se mantiene @JsonBackReference aquí para evitar ciclos de referencia
        private List<Tarea> tareas; // Relación con Tarea


        public Long getId() {
            return id;
        }

        public void setId(Long id) {
            this.id = id;
        }

        public String getNombre() {
            return nombre;
        }
```

Imagen de elaboración propia.

Proyecto.entity

```java
package com.crud_tareas.entity;

import java.util.List;

@Entity
@Table (name = "proyecto")
public class Proyecto {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column (name="nombreProyecto")
    private String nombre;
    private String descripcion;
    public Long getId() {
        return id;
    }

    @OneToMany(mappedBy = "proyecto", fetch =FetchType.EAGER) // Relación de un proyecto a muchas tareas
    private List<Tarea> tareas;

    public void setId(Long id) {
        this.id = id;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getDescripcion() {
        return descripcion;
    }
    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

}
```

Imagen de elaboración propia.

Departamento.entity

```java
package com.crud_tareas.entity;

import java.util.List;

@Entity
@Table (name = "departamento")
public class Departamento {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @Column (name="nombreDepartamento")
    private String nombre;


    @OneToOne(mappedBy = "departamento", fetch =FetchType.EAGER) // Relación de un responsable a un departamento
    private Responsable responsable;


    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

}
```

Imagen de elaboración propia.

Puesto.entity

```java
package com.crud_tareas.entity;

import java.util.List;

@Entity
@Table (name = "puesto")
public class Puesto {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column (name="nombrePuesto")
    private String nombre;

    @OneToOne(mappedBy = "puesto", fetch =FetchType.EAGER) // Relación de un responsable a un puesto
    @JsonIgnoreProperties({"puesto"}) // Ignora la propiedad "puesto" de Responsable para evitar el bucle
    @JsonIgnore
    private Responsable responsable;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public Responsable getResponsable() {
        return responsable;
    }
    public void setResponsable(Responsable responsable) {
        this.responsable = responsable;
    }

}
```

Imagen de elaboración propia.

Responsable.repository

```java
package com.crud_tareas.repository;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface ResponsableRepository extends JpaRepository<Responsable, Long> {

    Responsable save(ResponsableDto responsable);}

```

Imagen de elaboración propia.

Proyecto.repository

```java
package com.crud_tareas.repository;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface ProyectoRepository extends JpaRepository<Proyecto, Long> {}

```

Imagen de elaboración propia.

Departamento.repository

```java
package com.crud_tareas.repository;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface DdepartamentoRepository extends JpaRepository<Departamento, Long> {
}
```

Imagen de elaboración propia.

Puesto.repository

```java
package com.crud_tareas.repository;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface PuestoRepository extends JpaRepository<Puesto, Long> { }
```

Imagen de elaboración propia.

Responsable.service

```java
package com.crud_tareas.service;

import java.util.List;

@Service
public class ResponsableService {

    @Autowired
    private ResponsableRepository responsableRepository;

     @Autowired
     private PuestoService puestoService;

     @Autowired
     private DepartamentoService departamentoService;

    //Consulta de todos los responsables
    @Transactional(readOnly = true)
    public List<Responsable> findAll() {
        return (List<Responsable>)responsableRepository.findAll();
    }

    //Consulta por ID
        @Transactional(readOnly = true)
        public Responsable findById(Long id) {
            return (Responsable) responsableRepository.findById(id).orElse(null);
        }


    //Crear nuevo responsable
    @Transactional
    public Responsable createResponsable (ResponsableDto responsable) {
        Responsable responsableEntity= new Responsable();
        responsableEntity.setNombre(responsable.getNombre());
        responsableEntity.setApellido(responsable.getApellido());
        responsableEntity.setCorreo(responsable.getCorreo());
        responsableEntity.setCelular(responsable.getCelular());

        // Obtener el departamento y el puesto a partir de sus IDs
        if (responsable.getDepartamento() != null) {
            Departamento departamento = departamentoService.findById(responsable.getDepartamento().getId());
            responsableEntity.setDepartamento(departamento);
        }
```

Imagen de elaboración propia.

Proyecto.service

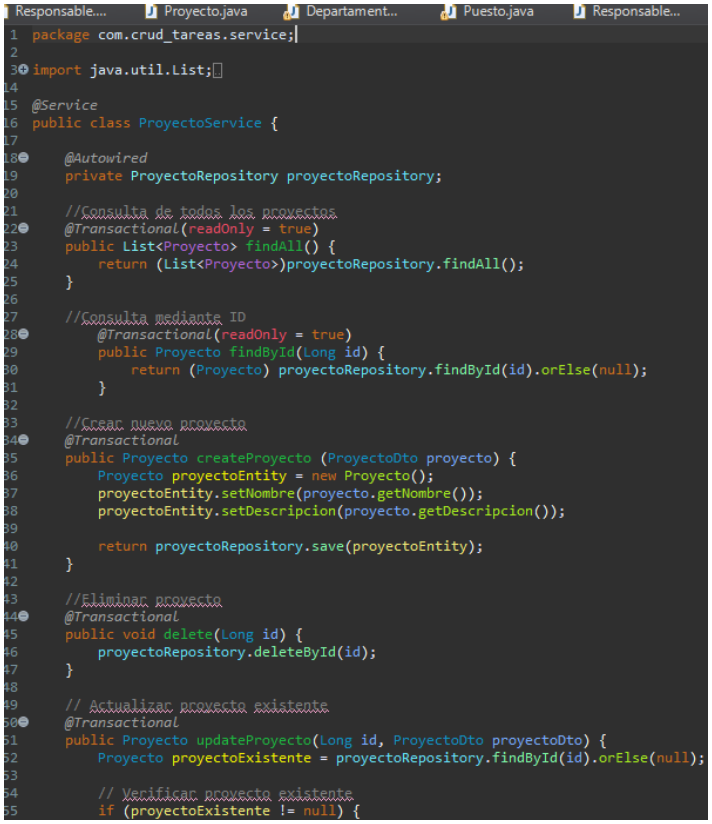


Imagen de elaboración propia.

Departamento.service

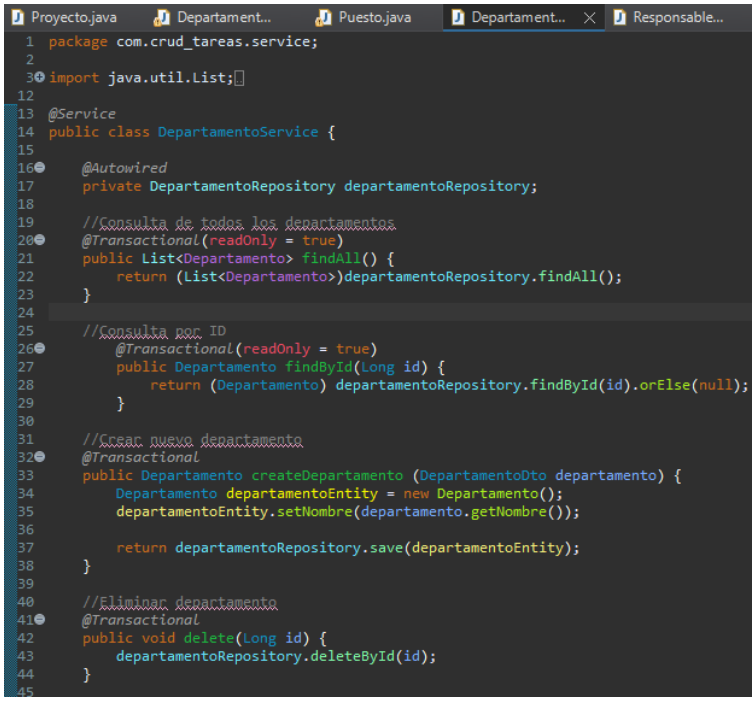


Imagen de elaboración propia.

Puesto.service

```java
private PuestoRepository puestoRepository;

//Consulta de todos los puestos
@Transactional(readOnly = true)
public List<Puesto> findAll() {
    return (List<Puesto>)puestoRepository.findAll();
}

//Consulta por ID
    @Transactional(readOnly = true)
    public Puesto findById(Long id) {
        return (Puesto) puestoRepository.findById(id).orElse(null);
    }

//Crear nuevo puesto
@Transactional
public Puesto createPuesto (PuestoDto puesto) {
    Puesto puestoEntity = new Puesto();
    puestoEntity.setNombre(puesto.getNombre());

    return puestoRepository.save(puestoEntity);
}

//Eliminar puesto
@Transactional
public void delete (Long id) {
    puestoRepository.deleteById(id);
}

// Actualizar puesto existente
@Transactional
public Puesto updatePuesto(Long id, PuestoDto puesto) {
    Puesto puestoExistente = puestoRepository.findById(id).orElse(null);

    // Verificar puesto existente
    if (puestoExistente != null) {
```

Imagen de elaboración propia.

Ahora correremos nuestro proyecto dando clic derecho en nuestro proyecto, clic en "run as" y seleccionamos "Spring Boot App".

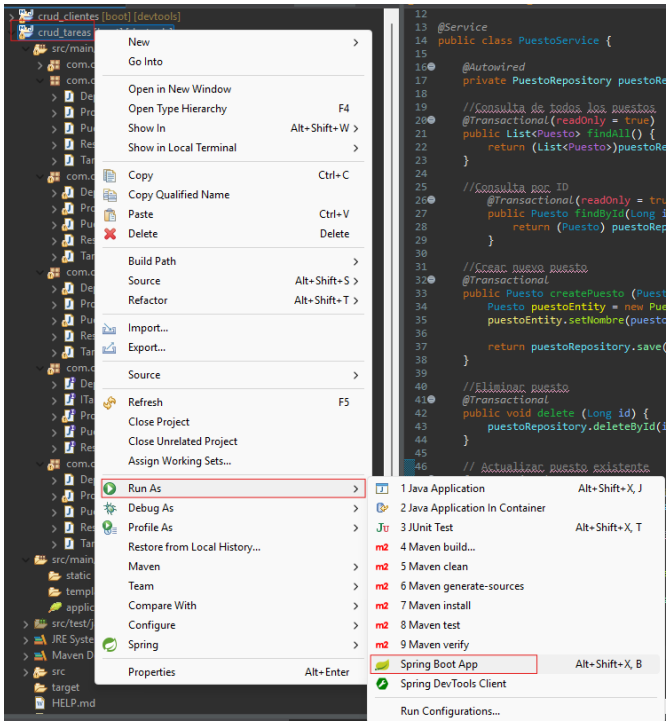


Imagen de elaboración propia.

Nos mostrará en consola como se va generando el proyecto y en caso de existir errores los mostrará como “error”, de lo contrario se mostrará todo lo que se generó.

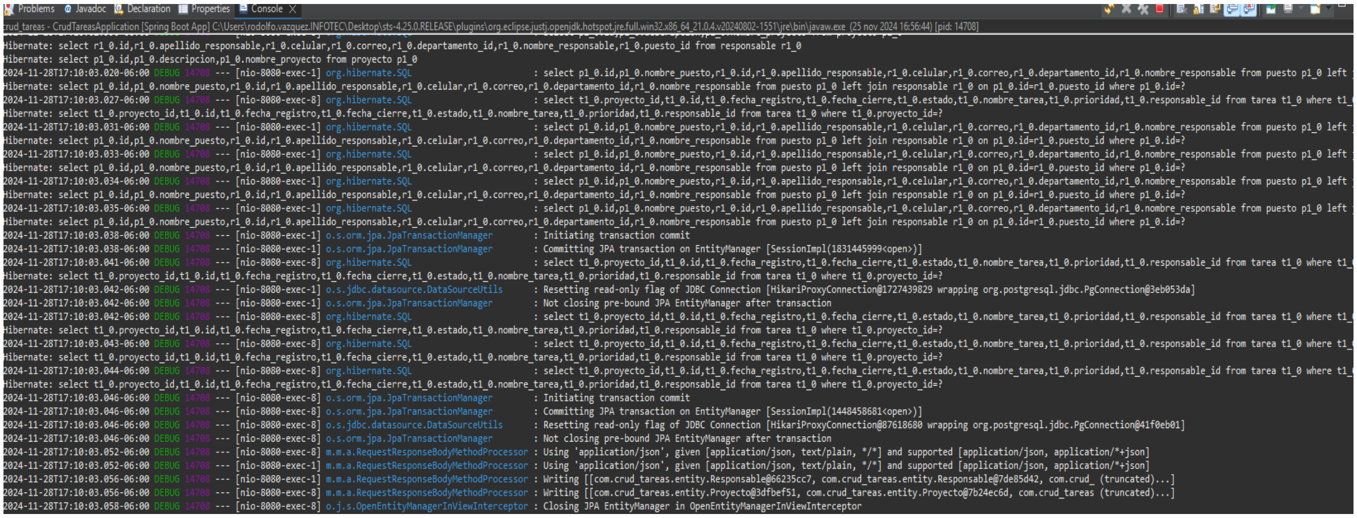


Imagen de elaboración propia.

Una vez que nos corre en consola sin errores, tenemos que validar que en nuestra base de datos de PostgreSQL se hayan creado las tablas.

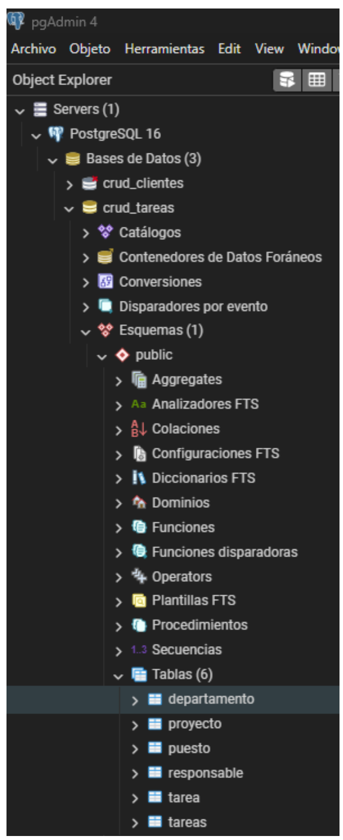


Imagen de elaboración propia.

Una vez que las tablas nos aparecen en nuestra base de datos, podremos insertar datos en cada una de las tablas, a continuación, se muestra un ejemplo del INSERT INTO en una de las tablas de la base de datos.

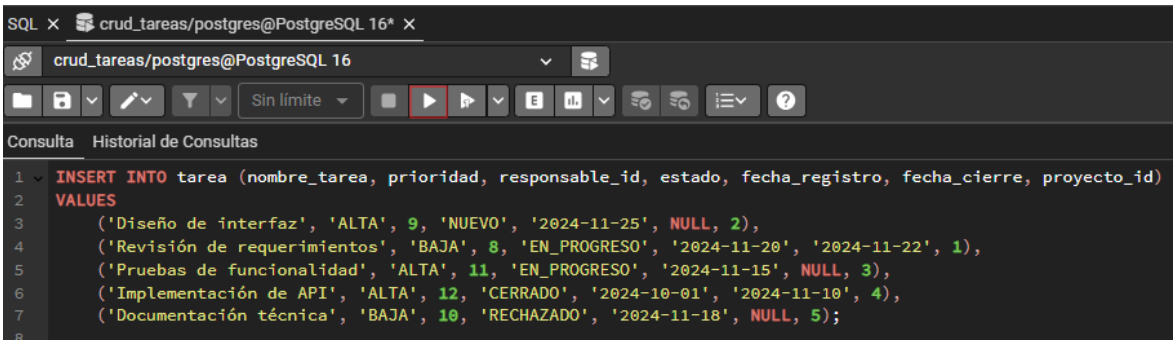


Imagen de elaboración propia.

Una vez que corremos nuestro query para el INSERT de la tabla tarea realizaremos nuestra consulta de los datos guardados con SELECT * FROM tarea.

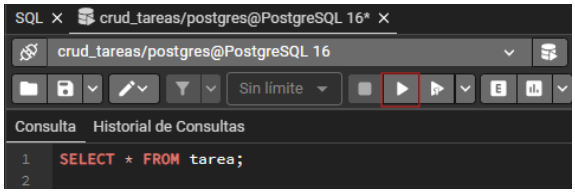
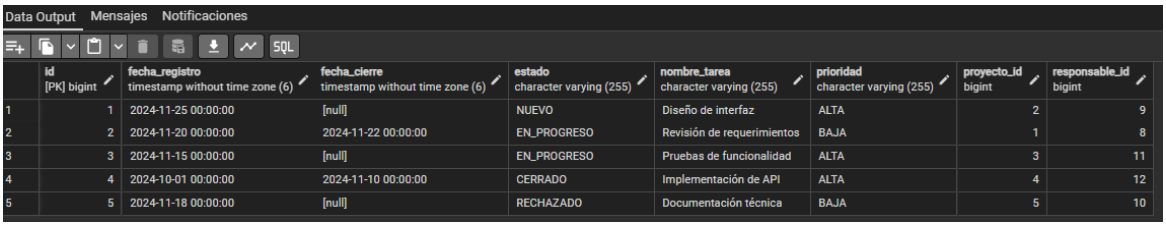


Imagen de elaboración propia.

Nos debería mostrar el siguiente resultado.

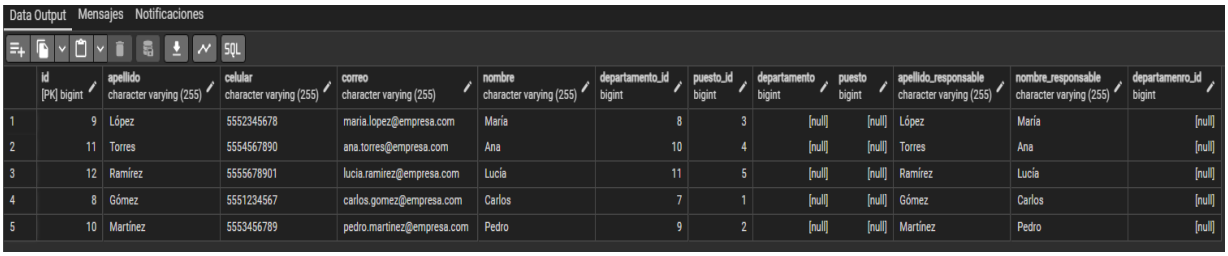


	id [PK] bigint	fecha_registro timestamp without time zone (6)	fecha_cierre timestamp without time zone (6)	estado character varying (255)	nombre_tarea character varying (255)	prioridad character varying (255)	proyecto_id bigint	responsable_id bigint
1	1	2024-11-25 00:00:00	[null]	NUEVO	Diseño de interfaz	ALTA	2	9
2	2	2024-11-20 00:00:00	2024-11-22 00:00:00	EN_PROGRESO	Revisión de requerimientos	BAJA	1	8
3	3	2024-11-15 00:00:00	[null]	EN_PROGRESO	Pruebas de funcionalidad	ALTA	3	11
4	4	2024-10-01 00:00:00	2024-11-10 00:00:00	CERRADO	Implementación de API	ALTA	4	12
5	5	2024-11-18 00:00:00	[null]	RECHAZADO	Documentación técnica	BAJA	5	10

Imagen de elaboración propia.

Así mismo se fueron realizando los registros de las demás tablas (responsable, proyecto, departamento y puesto).

Responsable



	id [PK] bigint	apellido character varying (255)	celular character varying (255)	correo character varying (255)	nombre character varying (255)	departamento_id bigint	puesto_id bigint	departamento bigint	puesto bigint	apellido_responsable character varying (255)	nombre_responsable character varying (255)	departamenro_id bigint
1	9	López	5552345678	maria.lopez@empresa.com	María	8	3	[null]	[null]	López	María	[null]
2	11	Torres	5554567890	ana.torres@empresa.com	Ana	10	4	[null]	[null]	Torres	Ana	[null]
3	12	Ramírez	5555678901	lucia.ramirez@empresa.com	Lucía	11	5	[null]	[null]	Ramírez	Lucía	[null]
4	8	Gómez	5551234567	carlos.gomez@empresa.com	Carlos	7	1	[null]	[null]	Gómez	Carlos	[null]
5	10	Martínez	5553456789	pedro.martinez@empresa.com	Pedro	9	2	[null]	[null]	Martínez	Pedro	[null]

Imagen de elaboración propia.

Proyecto

Data Output | Mensajes | Notificaciones

	id [PK] bigint	descripcion character varying (255)	nombre character varying (255)	nombre_proyecto character varying (255)
1	1	Desarrollo de un sistema integral para gestión empresari...	Sistema de Gestión	Sistema de Gestión
2	2	Creación de una app móvil para clientes.	Aplicación Móvil	Aplicación Móvil
3	3	Planificación y ejecución de una campaña publicitaria.	Campaña Publicitaria	Campaña Publicitaria
4	4	Implementación de procesos automatizados en finanzas.	Automatización Financiera	Automatización Financiera
5	5	Mejora de procesos logísticos y de producción.	Optimización de Operaciones	Optimización de Operaciones

Imagen de elaboración propia.

Departamento

Data Output | Mensajes | Notificaciones

	id [PK] bigint	nombre character varying (255)	nombre_departamento character varying (255)
1	7	Recursos Humanos	RH
2	8	Tecnología de la Información	TI
3	9	Marketing	MKT
4	10	Finanzas	FNZ
5	11	Operaciones	OPS

Imagen de elaboración propia.

Puesto

Data Output | Mensajes | Notificaciones

	id [PK] bigint	nombre character varying (255)	nombre_puesto character varying (255)
1	1	Gerencia	Gerente
2	2	Analisis	Analista
3	3	Desarrollo	Desarrollador
4	4	Diseño	Diseñador
5	5	Soporte	Soporte Técnico

Imagen de elaboración propia.

Como ya funciona nuestra base de datos, resta realizar nuestras pruebas en **postman** para validar que está tomando la base de datos en las rutas designadas en Spring, y así mismo en el puerto que indicamos. Realizaremos nuestra búsqueda con nuestro método GET y colocamos la ruta localhost en el puerto 8080/tareas.

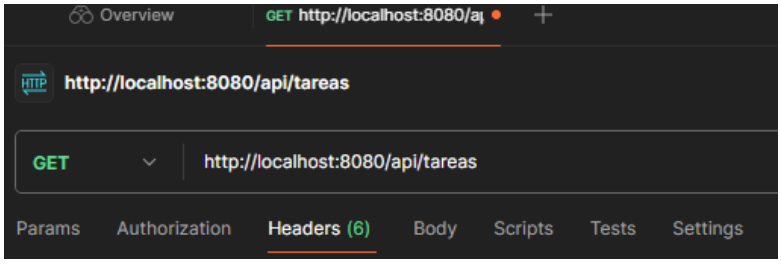


Imagen de elaboración propia.

Una vez que ingresamos la ruta de tareas, damos clic en SEND para visualizar la consulta.

Imagen de elaboración propia.

De esta manera realizaremos las pruebas para las otras entidades (responsable, proyecto, departamento y puesto). Así mismo haciendo uso de los métodos Post, Put y Delete.

Para continuar, ahora se mostrará lo realizado para la parte del frontend con el IDE Visual Studio Code y los frameworks Angular, Bootstrap, así como el uso de HTML.

Abriremos la carpeta donde guardaremos nuestro proyecto y abriremos la consola ya sea de powershell, o la misma que nos proporciona el IDE VSC, y crearemos nuestro proyecto en Angular.

NOTA: Anteriormente ya debimos haber hecho la instalación de la versión 16 de Angular y la correspondiente soportada para Bootsatrap.

Una vez que nos abrió la terminal en la ruta de la carpeta donde guardaremos el proyecto, vamos a crear nuestro proyecto con el comando "**ng new front-tareas**", es decir, el comando **ng new** de angular y en seguida el nombre que le vamos a poner al proyecto.

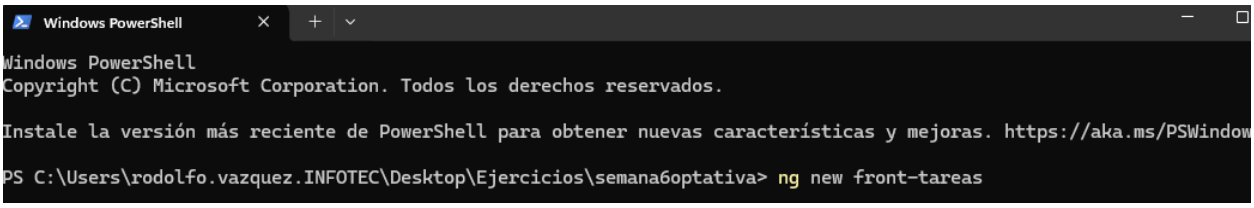


Imagen de elaboración propia.

Una vez que creamos el proyecto, de una vez crearemos también nuestro primer componente **tarea** con el comando "ng generate component components/tarea".

```
PS C:\Users\rodolfo.vazquez.INFOTEC\Desktop\Ejercicios\semana6optativa\front-tareas> ng generate component components/tarea
```

Imagen de elaboración propia.

Una vez que se crea el proyecto Angular, podemos correr el servidor con el comando "**ng serve -o**" para asegurarnos de que el proyecto fue creado correctamente.

```
PS C:\Users\rodolfo.vazquez.INFOTEC\Desktop\Ejercicios\semana6optativa\front-tareas> ng serve -o
```

Imagen de elaboración propia.

También podemos crear nuestras carpetas para los componentes y nuestros archivos des el IDE, dando clic derecho sobre nuestro proyecto.

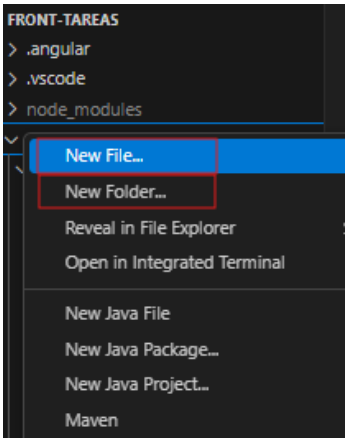


Imagen de elaboración propia.

Solo tenemos que asegurarnos que clase de archivo vamos a crear, para archivos HTML, con la extensión "html", para archivos de estilos con la extensión "css", y para los componentes y servicios con la extensión de TypeScrip "ts".

Como vamos a trabajar con otras 4 entidades aparte de la entidad "tarea", crearemos los componentes y servicios de cada una del resto de las entidades (responsable, proyecto, departamento y puesto). De tal manera que nuestro explorador de archivos de VSC se deberá ver de la siguiente manera:

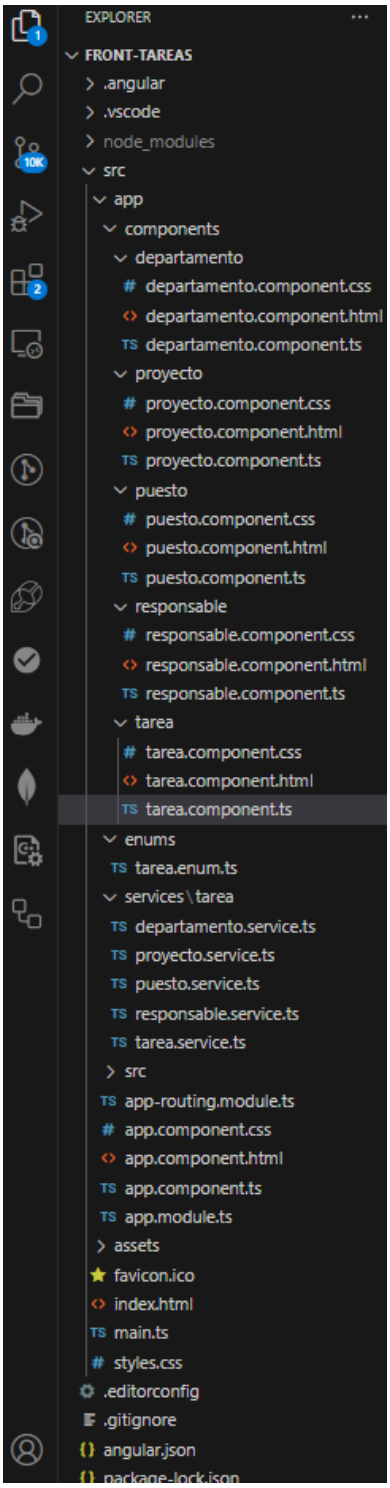


Imagen de elaboración propia.

De esta manera es como empezaremos a trabajar nuestro código dentro de nuestros archivos.

Con los archivos que inicialmente vamos a trabajar es con los "app.component", los cuales se generan automáticamente cuando creamos el proyecto desde la consola, pero antes de ello vamos a insertar nuestro enlace para poder hacer uso de **Bootstrap** mediante una etiqueta "link".

```
src > index.html > html > body > app-root
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Front-Tareas</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
</head>
<body>
  <app-root>
  <app-responsable></app-responsable>
</app-root>
</body>
</html>
```

Imagen de elaboración propia.

En nuestro archivo app.component iniciamos dándole un poco de forma a lo que va a ser nuestra página web, integrando una barra de navegación nuestro contenedor que vamos a usar como el body, y nuestro pie de página.

```
src > app > app.component.html > footer.bg-dark.text-center.text-white.py-3
      Go to component
<!-- Navbar -->
<nav class="navbar navbar-expand-lg navbar-success bg-success text-white">
  <div class="container">
    <div class="navbar-nav mx-auto">
      <a class="nav-item nav-link text-white" routerLink="/tarea">Tareas</a>
      <a class="nav-item nav-link text-white" routerLink="/responsable">Responsables</a>
      <a class="nav-item nav-link text-white" routerLink="/proyecto">Proyectos</a>
      <a class="nav-item nav-link text-white" routerLink="/departamento">Departamentos</a>
      <a class="nav-item nav-link text-white" routerLink="/puesto">Puestos</a>
    </div>
  </div>
</nav>

<div class="container my-4">
  <!-- Contenido Dinámico de las Entidades -->
  <router-outlet></router-outlet>
</div>

<!-- Footer -->
<footer class="bg-dark text-center text-white py-3">
  <p>CRUD de Entidades © 2024</p>
</footer>
```

Imagen de elaboración propia.

Seguimos con un poco de estilos generales de nuestro proyecto agregando un fondo de gradiente.

```
src > # styles.css > body
1   body{
2       background: #2980B9;  /* fallback for old browsers */
3       background: -webkit-linear-gradient(to right, #FFFFFF, #6DD5FA, #2980B9);  /* Chrome 10-25, Safari 5.1-6 */
4       background: linear-gradient(to right, #FFFFFF, #6DD5FA, #2980B9); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
5
6   }
```

Imagen de elaboración propia.

Continuamos generando nuestro constructor para el formulario y tarea service, así como nuestras variables y nuestros métodos que vamos a utilizar para que consuma los datos para crear las consultas y las demás acciones del CRUD.

```
src > app > TS app.component.ts > ...
1   import { Component, OnInit } from '@angular/core';
2   import { FormGroup, FormBuilder, Validators } from '@angular/forms';
3   import { Prioridad, Estado } from './enums/tarea.enum'; // Importar los ENUMs
4   import { TareaService } from "./services/tarea/tarea.service";
5
6   @Component({
7     selector: 'app-root',
8     templateUrl: './app.component.html',
9     styleUrls: ['./app.component.css']
10  })
11  export class AppComponent implements OnInit{
12
13    tareaForm!: FormGroup;
14    prioridades = Object.values(Prioridad); // Cargar las opciones del ENUM
15    estados = Object.values(Estado); // Cargar las opciones del ENUM
16    tarea: any[] =[];
17    filteredTarea: any[] = [];
18    tareaSeleccionada: any = null;
19
20
21    constructor(
22      public fb: FormBuilder,
23      public tareaService: TareaService){ }
24
25    ngOnInit(): void {
26
27      this.tareaForm = this.fb.group({
28
29        nombre : ['', Validators.required],
30        prioridad : ['', Validators.required],
31        responsableId : ['', Validators.required],
32        estado : ['', Validators.required],
33        fechaRegistro: ['', Validators.required],
34        fechaCierre: ['', Validators.required],
35        proyectoId : ['', Validators.required]
36
37      });
38
39      this.cargarTareas();
40    }
41
42    cargarTareas(): void {
43      this.tareaService.getAllTarea().subscribe(resp => {
44        this.tarea = resp;
45        this.filteredTarea = this.tarea;
46      },
```

Imagen de elaboración propia.

Nos aseguramos que nuestros archivos estén en ruta adecuadamente en nuestro archivo app-routing.module.ts

src > app > TS app-routing.module.ts > ...

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { TareaComponent } from './components/tarea/tarea.component';
import { ResponsableComponent } from './components/responsable/responsable.component';
import { ProyectoComponent } from './components/proyecto/proyecto.component';
import { DepartamentoComponent } from './components/departamento/departamento.component';
import { PuestoComponent } from './components/puesto/puesto.component';

const routes: Routes = [
  { path: 'tarea', component: TareaComponent },
  { path: 'responsable', component: ResponsableComponent },
  { path: 'proyecto', component: ProyectoComponent },
  { path: 'departamento', component: DepartamentoComponent },
  { path: 'puesto', component: PuestoComponent },
  { path: '**', redirectTo: 'tarea' }
];



@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Imagen de elaboración propia.

De la misma manera nos aseguramos de que estemos importando nuestros archivos correctamente en nuestro archivo app.module.ts y así mismo se lo estemos exportando a nuestro AppModule.

src > app > TS app.module.ts > AppModule

```ts
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ResponsableComponent } from './components/responsable/responsable.component';
import { ProyectoComponent } from './components/proyecto/proyecto.component';
import { DepartamentoComponent } from './components/departamento/departamento.component';
import { PuestoComponent } from './components/puesto/puesto.component';
import { TareaComponent } from './components/tarea/tarea.component';

import { FormsModule } from '@angular/forms';
import { ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { TareaService } from './services/tarea/tarea.service';
import { ResponsableService } from './services/tarea/responsable.service';
import { ProyectoService } from './services/tarea/proyecto.service';
import { DepartamentoService } from './services/tarea/departamento.service';
import { PuestoService } from './services/tarea/puesto.service';


@NgModule({
  declarations: [
    AppComponent,
    TareaComponent,
    ResponsableComponent,
    ProyectoComponent,
    DepartamentoComponent,
    PuestoComponent,

  ],

  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
  ],

  providers: [TareaService,
              ResponsableService,
              ProyectoService,
              DepartamentoService,
              PuestoService
            ],
  bootstrap: [AppComponent, ResponsableComponent]
})
export class AppModule { }
```

Imagen de elaboración propia.

Continuamos creando la parte visual con html y bootstrap, como nuestros formularios, campos de texto, labels, botones de las entidades que estaremos usando.

src > app > components > tarea > tarea.component.html > div.container > form.mb-4 > div.row > div.col-md-6.mb-3

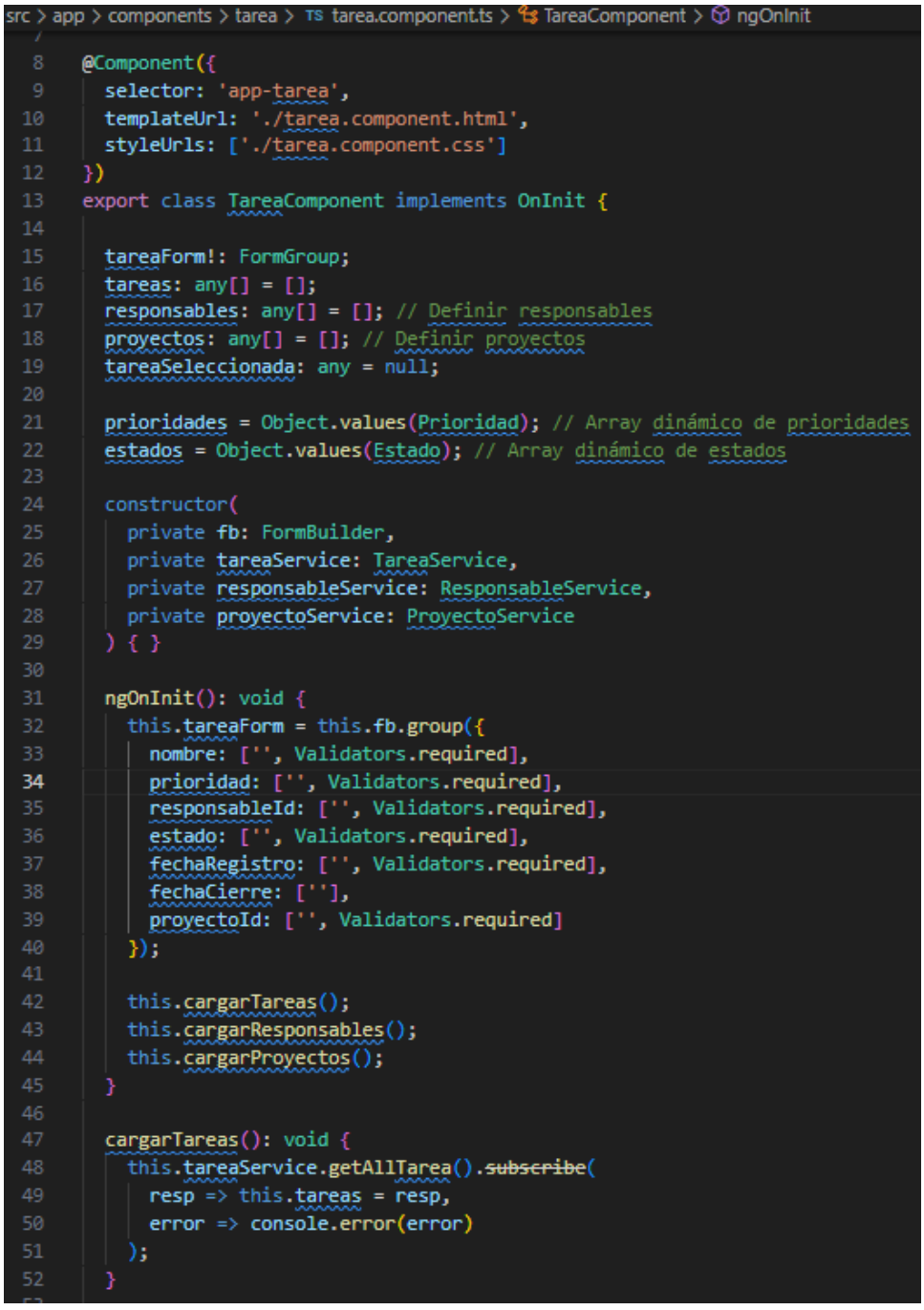
Go to component

```
<div class="container">
  <h2 class="text-center my-3">Gestión de Tareas</h2>

  <!-- Formulario -->
  <form [formGroup]="tareaForm" (ngSubmit)="guardar()" class="mb-4">
    <div class="row">

      <!-- Nombre -->
      <div class="col-md-6 mb-3">
        <label for="nombre" class="form-label">Nombre</label>
        <input id="nombre" type="text" class="form-control" formControlName="nombre">
        <div *ngIf="tareaForm.get('nombre')?.invalid && tareaForm.get('nombre')?.touched" class="text-danger">
          El nombre es obligatorio.
        </div>
      </div>

      <!-- Prioridad -->
      <div class="col-md-6 mb-3">
        <label for="prioridad" class="form-label">Prioridad</label>
        <select id="prioridad" class="form-select" formControlName="prioridad">
          <option value="">Seleccione</option>
          <option *ngFor="let prioridad of prioridades" [value]="prioridad">
            {{ prioridad }}
          </option>
        </select>
        <div *ngIf="tareaForm.get('prioridad')?.invalid && tareaForm.get('prioridad')?.touched" class="text-danger">
          La prioridad es obligatoria.
        </div>
      </div>

      <!-- Responsable -->
      <div class="col-md-6 mb-3">
        <label for="responsableId" class="form-label">Responsable</label>
        <select id="responsableId" class="form-select" formControlName="responsableId">
          <option value="">Seleccione</option>
          <option *ngFor="let responsable of responsables" [value]="responsable.id">
            {{ responsable.nombre }} {{ responsable.apellido }}
          </option>
        </select>
        <div *ngIf="tareaForm.get('responsableId')?.invalid && tareaForm.get('responsableId')?.touched" class="text-danger">
          El responsable es obligatorio.
        </div>
      </div>

      <!-- Estado -->
      <div class="col-md-6 mb-3">
```

Imagen de elaboración propia.

De esta misma manera vamos a estar generando nuestro código de las entidades de los archivos responsable.component.html, proyecto.component.html, departamento.component.html y puesto.component.html

Ahora vamos a generar nuestra parte lógica y funcional en nuestro archivo tarea.component.ts generando nuestros métodos para cargar tareas, responsables, proyectos, guardar y eliminar tareas.




```ts
@Component({
  selector: 'app-tarea',
  templateUrl: './tarea.component.html',
  styleUrls: ['./tarea.component.css']
})
export class TareaComponent implements OnInit {

  tareaForm!: FormGroup;
  tareas: any[] = [];
  responsables: any[] = []; // Definir responsables
  proyectos: any[] = []; // Definir proyectos
  tareaSeleccionada: any = null;

  prioridades = Object.values(Prioridad); // Array dinámico de prioridades
  estados = Object.values(Estado); // Array dinámico de estados

  constructor(
    private fb: FormBuilder,
    private tareaService: TareaService,
    private responsableService: ResponsableService,
    private proyectoService: ProyectoService
  ) { }

  ngOnInit(): void {
    this.tareaForm = this.fb.group({
      nombre: ['', Validators.required],
      prioridad: ['', Validators.required],
      responsableId: ['', Validators.required],
      estado: ['', Validators.required],
      fechaRegistro: ['', Validators.required],
      fechaCierre: [''],
      proyectoId: ['', Validators.required]
    });

    this.cargarTareas();
    this.cargarResponsables();
    this.cargarProyectos();
  }

  cargarTareas(): void {
    this.tareaService.getAllTarea().subscribe(
      resp => this.tareas = resp,
      error => console.error(error)
    );
  }
```

Imagen de elaboración propia.

De esta misma manera vamos a estar generando nuestro código de las entidades de los archivos responsable.component.ts, proyecto.component.ts, departamento.component.ts y puesto.component.ts

Seguido de nuestro deberemos tener las siguientes vistas en el navegador.

Tarea

Imagen de elaboración propia.

Responsable

Imagen de elaboración propia.

Proyecto

Imagen de elaboración propia.

Departamento

Imagen de elaboración propia.

Puesto

Imagen de elaboración propia.

A continuación, se muestran nuestras relaciones que utilizamos para el gestor de tareas:

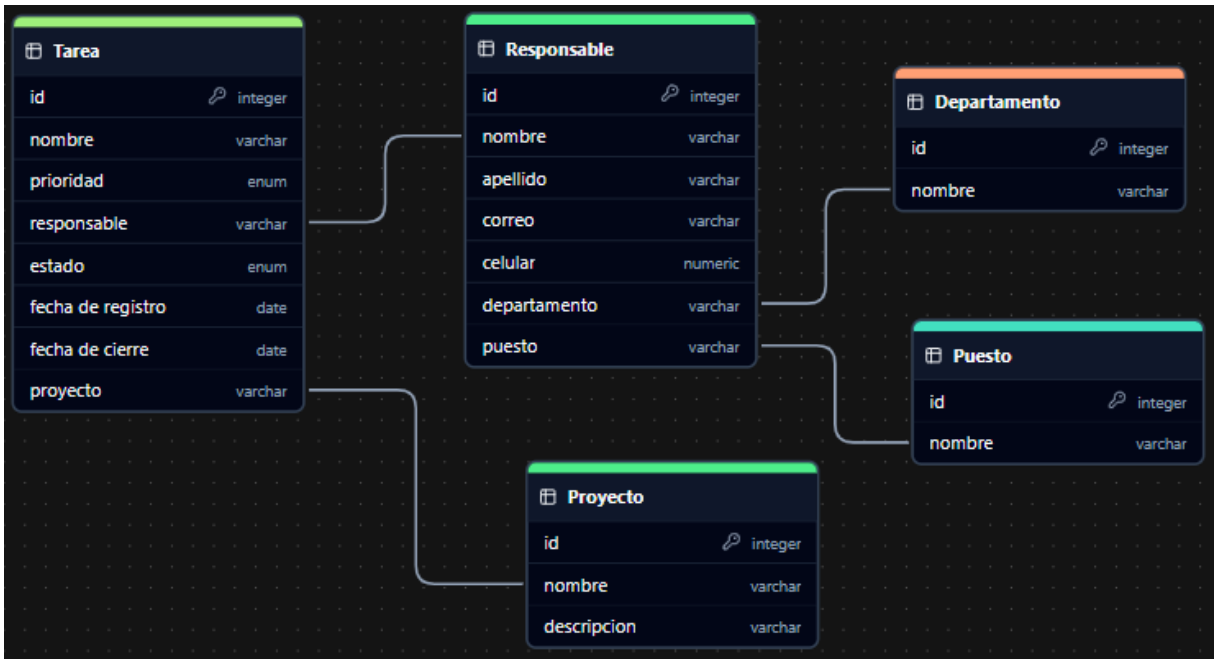


Imagen de elaboración propia.

Práctica profesional

Datos personales del estudiante			
Nombre completo:	Rodolfo Gael Vazquez Ubaldo		
Matrícula:	Eliminado	Teléfono:	Eliminado
Correo:	Eliminado		
Especialidad de TSU:	Desarrollo de Software de Código Abierto		
Año:	2024	Celular:	Eliminado
Número de horas de práctica profesional:		280 horas	

Datos de la institución	
Nombre de la institución:	Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación (INFOTEC)
Departamento en el que realizarás tus prácticas:	Dirección Adjunta de Administración de Proyectos - Desarrollo de Software
Nombre de la persona responsable del departamento:	Claudio Moran Ponce

Estructura organizacional

- **El sector de actividad**

El INFOTEC, tiene contribución en la Transformación Digital para todo México, por medio de investigación, innovación, educación o formación académica y ofrece productos como desarrollo y servicios de tecnologías de la información. Sus alcances abarcan al sector público y privado, habilitando caminos que conduzcan hacia un México moderno y de inclusión digital.

- **Presentación de la empresa e historia empresarial**

INFOTEC es una institución con más de 40 años de vida que, desde sus orígenes en 1974, se ha dedicado a la distribución del conocimiento; primero, en materia de innovación en la industria, proyectos específicos sobre las TIC´s, con la intención de incrementar los avances en México.

En la actualidad, se cuenta con un par de sedes, una en la CDMX y, a partir del 2013 la segunda en Aguascalientes. Con respecto al portafolio de soluciones, las principales líneas de negocio incluyen desarrollo de aplicaciones, IoT (internet de las cosas), tecnología en infraestructura, así como docencia y temas de investigación.

Nuestros servicios de desarrollo están acreditados con la certificación CMMI nivel 5 y contamos con un Centro de Datos en la ciudad de Aguascalientes, reconocido como el primer data center del sector público mexicano certificado internacionalmente como Tier III por el Uptime Institute.

Nuestra oferta académica incluye programas especializados de maestría y doctorado en dirección estratégica, gestión de innovación, y derecho de las TIC, así como en ciencia de datos y sistemas embebidos, entre otras.

La investigación que se desarrolla en INFOTEC tiene por finalidad la generación de conocimiento y su aplicación en el campo de las TIC para el desarrollo de productos y servicios tecnológicos, así como de modelos y metodologías destinados al crecimiento nacional.

Por esta razón, contamos con diversos ámbitos de conocimiento, los cuales representan tendencias en TIC a nivel global y derivan en proyectos de investigación que inciden de manera directa en el desarrollo de sectores clave del país.

La organización de la empresa (incluir organigrama)

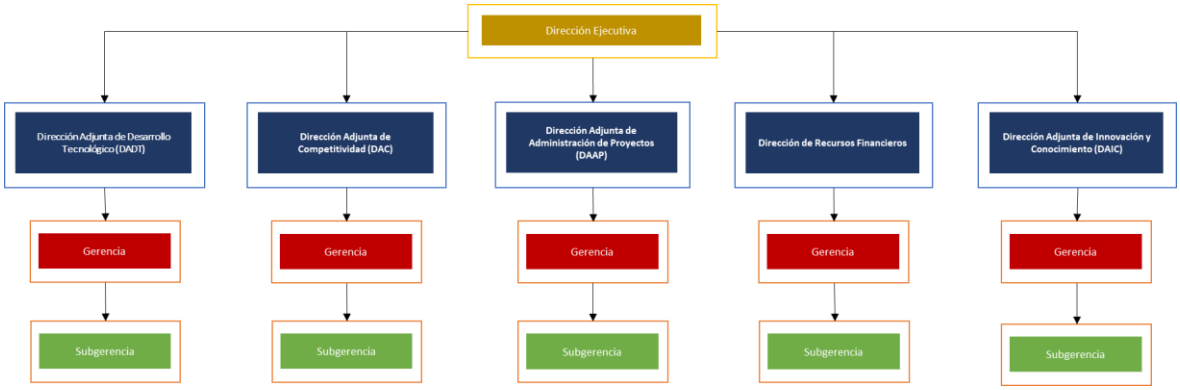


Imagen de elaboración propia.

Informe de la práctica profesional

- **Descripción de las funciones**

- Dailys semanales con Scrum Master y Líder Técnico, revisión de material de apoyo semanal y optativo, realización de actividades por semana.

- Revisión semanal de recursos de apoyo y aplicación de lo revisado en visual Studio Code (directivas, formularios basados en plantillas, formularios reactivos, pipes, tipado estático, clases, funciones, métodos, interfaces)

- Revisión semanal de recursos de apoyo y aplicación de lo revisado dentro del proyecto (generación de bases de datos con Spring, CRUD, JPA).

- Inserciones de datos a las tablas generadas dentro de la base de datos (INSERT INTO) y consultas de los datos de las entidades (SELECT *FROM "entidad").

- Validaciones de los métodos de consultas, creación, actualización y eliminación de los datos.

- Implementación de relación entre entidades en Spring (FetchType LAZY y EAGER).

- Generación de formularios con CRUD únicamente con Angular y arrays.

- Avances de semanales de la conclusión del proyecto "Gestor de tareas".

- **Desglose de actividades**

- Creación de página web con HTML, Bootstrap + CSS + JS
- Instalación de ambiente de trabajo para Angular
- Instalación de Spring Tool Suite
- Aplicación "presupuesto" – Uso de observables

- Creación de página web con HTML, Bootstrap + CSS + JS
- Instalación de ambiente de trabajo para Angular
- Instalación de Spring Tool Suite
- Aplicación "presupuesto" – Uso de observables

- Se realizó ejercicio práctico acerca de formularios reactivos en donde conocimos una mejor funcionalidad de estos formularios y una manera distinta de aplicar un required de los campos de un formulario – 30 de septiembre al 03 de oct.

- Se realizó una API REST con conexión a Postgresql/MySQL (Personalizado), es decir, retomamos la primera conexión de la Base de Datos que habíamos realizado anteriormente y se realizaron implementaciones adicionales y/o diferentes a las que se hicieron inicialmente- 07 al 13 de oct.

- Realizamos una Aplicación CRUD en Angular en donde al guardamos la información capturada mediante arreglos para poder consultarla, editarla y eliminarla- 07 al 20 de oct.
- Se revisó el material de "Relación entre entidades" con el objetivo de realizar implementaciones de relación a nuestro CRUD y dar avances al proyecto final – 21 al 27 de oct.

- Se va a llevar a cabo durante esta semana el "Cuestionario teórico" con conceptos y temas que se han visto y aprendido durante la estancia, y ver el "Material complementario "curso de Angular" – 21 de oct al 03 de nov.

Aplicación Web Spring Boot y Angular – 28 de oct al 03 de nov.

- **Bitácora de prácticas**

Semanas	Actividades realizadas	Observaciones
1 a 2	• En el transcurso de la semana se han revisado los materiales de apoyo y se realizan ejercicios prácticos con base al material revisado, así mismo, con lo visto se llevan a cabo las actividades que se entregan al final de la semana, en este caso lo que se ha visto en los materiales de apoyo es HTML, Bootstrap, la instalación de los ambientes de Angular y Spring, y TypeScript.	
3 a 4	• Introducción a Angular – 16 al 22 de septiembre ("¿Qué es Angular-Herramientas Necesarias?", "Creación de un proyecto, archivos y carpetas básicas", "Directivas y capturas de eventos", "Componentes", "Módulos: Creación y consumo", "Petición de un archivo JSON a un servidor", "Instalación de ambiente de trabajo para Angular"). • Ejercicio: Instalación de Spring Tool	

	Suite – 16 al 22 de sep. • TypeScript – 23 al 29 de septiembre ("Router: definición de rutas", "Servicios: concepto y pasos para crearlos", "Servicios: recuperación de datos de un servidor web", "Pipes: definición", "Pipes: creación de pipes personalizadas", "TypeScript", "TypeScript: tipado estático", "TypeScript: clases", "TypeScript: funciones y métodos", "TypeScript: herencia"," TypeScript: interfaces", "TypeScript: clases genéricas").	
5 a 6	• Ejercicio: Aplicación presupuesto_ uso de observables– 23 al 29 de sep. • Spring Boot – 23 de septiembre al 03 de oct ("Curso Spring Boot CRUD + JPA + Postgresql", "Clase Spring - BDD", "Ejercicio: API REST con conexión a Postgresql/MySQL"). • Ejercicio: Formularios reactivos – 30 de septiembre al 03 de oct. • Ejercicio: API REST con conexión a Postgresql/MySQL (Personalizado) - 07 al 13 de oct. • Aplicación CRUD en Angular - 07 al 20 de oct.	
7 a 8	• Relación entre entidades – 21 al 27 de oct. • Cuestionario teórico, Material complementario "curso de Angular" – 21 de oct al 03 de nov. • Aplicación Web Spring Boot y Angular – 28 de oct al 03 de nov.	
9 a 10	• Semana 9 – Avance Proyecto "Gestor de tareas" – 28 de oct al 28 de nov. • Semana 10 – Avance Proyecto "Gestor de tareas" – 28 de oct al 28 de nov.	

11 a 12	• Semana 11 – Avance Proyecto "Gestor de tareas" – 28 de oct al 28 de nov. • Ejercicio: Integración Spring Boot - Angular – 28 de oct al 28 de nov.	

Conclusiones

De manera general puedo decir que en lo personal fue una gran experiencia el haber tomado las prácticas de este TSU en desarrollo de software de código abierto, pues to que lo realizado fue algo que supero mis expectativas vs lo que se vio en el transcurso de los cuatrimestres a lo largo del curso, ya que nos enfocamos a algo en específico, haciendo uso de herramientas que no tuvimos oportunidad de conocer en las materias vistas y lo que considero mejor, hicimos uso de lenguajes específicos y frameworks durante todas las prácticas, lo cual me permitió enfocar la atención a un solo objetivo desarrollando conocimientos con herramientas y lenguajes populares pero robustos dentro de la industria del desarrollo de software.

Claro, no es de más mencionar que todo esto conllevo complicaciones en el transcurso del aprendizaje y sobretodo en la parte de la implementación, definitivamente no es lo mismo la teoría que la parte práctica, y esto me lleva a concluir que ¿el desarrollo de software es algo complejo de aprender y llevar a cabo?, puedo decir que si, ¿es algo imposible?, para nada imposible, pero como todo en lo absoluto, mientras más se practica se vuelve más fácil de entender y así mismo de llevas a cabo, me parece que la parte más satisfactoria es ver lo que lograste al final visualmente y funcionalmente hablando, después de arduas jornadas de código en donde seguramente vi muchos errores y la mayoría costaron trabajo resolverlos, pero llegar a ese resultado final, es lo que emociona y crea las ganas de continuar en este ámbito.

Cuadro CQA de mi estancia en la organización

C ¿Qué **C**onozco?	Q ¿**Q**ué aporte?	A ¿Qué **A**prendí?
Lo que conozco al inicio de estas prácticas era un poco sobre las bases de datos relacionales, el lenguaje html y un poco sobre java script, realmente un	Realmente no puedo decir que aporte conocimientos, ya que estos fueron los que realmente quien nos instruía en las prácticas era quien nos despejaba	Aprendí muchísimo en estas prácticas, como a usar TypeScript que está basado en JavaScript, Bootstrap, Agular y la generación de proyectos,

poquito más de lo básico, pero sabía que eso no sería suficiente para lo que veía venir en el proyecto.	de las dudas y nos apoyaba cuando teníamos complicaciones de cualquier tipo, pero lo que puedo decir, es que aporte mis ganas de cumplir este objetivo final y tomar lo mejor de las prácticas para llevarlo a un nivel más profesional, y que a pesar de las complicaciones personales que pude llegar a tener, no me quitaron el compromiso de intentar terminal de la mejor manera toda esta parte práctica del TSU y de la estancia.	componentes y servicios, tenía nociones de SQL pero aprendía a usar al menos de manera básica PostgreSQL, Spring haciendo uso de Java y realizando la conexión con la base de datos, a realizar la relación entre entidades, a generar las pruebas de lo anteriormente mencionado con Postman, en donde no iba a funcionar el GET, POST, PUT o DELETE si no había funcionado algo de lo anteriormente mencionado, y aprendí muchas más funcionalidades de html conto con el framework Bootstrap.

Evaluación de desempeño

Considero que mi evaluación de desempeño en la estancia dentro de la organización ha sido bastante buena, si bien no era de los mejores de la clase, me parece que siempre fui constante en todo lo que se pedía, como anteriormente comentaba, tuve situaciones personales que no me permitieron avanzar al ritmo que me hubiera gustado, siempre he hecho el esfuerzo por realizar todas las entregas en tiempo y forma y con la mejor calidad posible, y yo creo que en eso me doy más valor, porque siempre he sido una persona perseverante y que siempre y bajo la situación que sea intenta lograr sus objetivos, he tenido algunas lecciones de vida que en sus momento me complicaron el avance en mi vida profesional, y parte de lo que he aprendido es que nada es gratis y todo logro conlleva un esfuerzo a veces de más del 100%, pero que nada es imposible cuando quieres crecer y ser una persona más preparada para el día a día.

Referencias

- INFOTEC (s/f). "*Trayectoria con más de cuatro décadas de experiencia*". https://www.infotec.mx/Infotec

- *Referencia desde el punto de vista y conocimiento personal de Rodolfo Gael Vazquez Ubaldo.*
- TSU de "INFOTEC" (2023 - 2024). "*Desarrollo de Software de Código Abierto*".
- TSU de "INFOTEC" (2023 - 2024). "*Estancias Profesionales - Desarrollo de Software de Código Abierto*".
- INFOTEC (2024*). "¿Qué es INFOTEC?"*. https://www.infotec.mx/Infotec
- Bootcamp INFOTEC (2024). *"¿Qué es HTML y para qué sirve?"*. https://www.youtube.com/watch?v=eN39GOwCWzQ
- Bootcamp INFOTEC (2024). *"¿Qué es una etiqueta en HTML5 y cómo funciona?"*. https://youtu.be/q8ibtKwbt6c
- Victor robles web (2020). *"Aprende HTML5 en 15 minutos?"*. https://www.youtube.com/watch?v=mNbnV3aN3KA
- Bootcamp INFOTEC (2024). *"¿Qué es CSS?"*. https://youtu.be/pSM43Sj5vlg
- Bootcamp INFOTEC (2024). *"Aplicar estilos usando propiedades de CSS"*. https://youtu.be/idn1ybWFaeM
- Bootcamp INFOTEC (2024). *"¿Qué es un framework?"*. https://youtu.be/J1hmAIrEya4
- Bootcamp INFOTEC (2024). *"Funcionalidades de Bootstrap"*. https://youtu.be/iyU_g6DL5dU
- Dostin Hurtado (2023). *Creación de página web con HTML, Bootstrap + CSS + JS"*. https://youtu.be/-ZInT4p9hH8
- Bootcamp INFOTEC (2024). *Creación de un proyecto, archivos y carpetas básicas en Angular"*. https://youtu.be/yMgCrv-631w
- Bootcamp INFOTEC (2024). *"Directivas y capturas de eventos"*. https://youtu.be/Z5UJn95z8p4
- Bootcamp INFOTEC (2024). "*Componentes"*. https://youtu.be/QZDwe7jAOU
- Bootcamp INFOTEC (2024). *"Módulos, creación y consumo"*. https://youtu.be/ySXPFF78hJQ

- Electronikoin Team (2023). "*Instalación de ambiente de trabajo para Angular*". https://youtu.be/Z6kiJkqiCFI
- Cristian Ballesteros (2023). "*Instalación de Spring Tool Suite*".
- Bootcamp INFOTEC (2024). "*Servicios, conceptos y pasos para crearlos*". https://youtu.be/bf2__sIDsL8
- Bootcamp INFOTEC (2024). "*Pipes: definición*". https://youtu.be/9BStgKzeKY0
- Bootcamp INFOTEC (2024). "*Pipes: creación de pipes personalizadas*". https://youtu.be/AM063v6DTk8
- Bootcamp INFOTEC (2024). "*TypeScript*". https://youtu.be/NQwedBch0YM
- Bootcamp INFOTEC (2024). "*TypeScript: tipado estático*". https://youtu.be/mXCnVblTRFI
- Bootcamp INFOTEC (2024). "*TypeScript: clases*". https://youtu.be/44uSc-S_7xQ
- Bootcamp INFOTEC (2024). "TypeScript: *Funciones y métodos*". https://youtu.be/FdhgBhEJBYs
- Bootcamp INFOTEC (2024). "*TypeScript: herencia*". https://youtu.be/Kb-1bIG124I
- Tomas Ruíz Díaz (2021). "Ejercicio: *Aplicación presupuesto – uso de observables*". https://youtu.be/JsNMnXD-NRM
- Yo Androide (2020). "*Curso Spring Boot Crud + jpa + Postgresql*". https://youtu.be/liGDWn55fxw
- Coders Free (2023). "*Formularios reactivos*". https://www.youtube.com/playlist?list=PLZ2ovOgdI-kWDh3jDh-GvgToRlVfwIUFw
- Fazt Code (2024). "*Ejercicio: Aplicación CRUD en Angular*". https://youtu.be/arGRTVdG--c
- JAAX (2023). "*Cascading, FetchType*". https://youtu.be/AnBXZK-WA1U