Practica SPSolutions

Elaboración de un sistema web para dar solución a la problemática existente el control de las experiencias y habilidades de un postulante. Con la creación e implementación de dicho sistema se busca realizar un manejo de la información del postulante de una manera más eficiente.

- BackEnd desarrollado en lenguaje de programación Java.
- Uso de Framework Spring Boot.
- REST.
- JPA.
- FrontEnd desarrollado en Framework Angular.
- Boots
- Uso de gestor de bases de datos MySQL.

Spring Boot

Tecnología basada en el framework Spring que facilitan el proceso de creación de aplicaciones basadas en Spring, además de ofrecer mayor facilidad para personalizar cualquier proyecto al inicio y durante su desarrollo, debido a las ventajas que ofrece Spring Boot se decidió hacer uso de este framework para el desarrollo del proyecto.

REST

La API de tipo REST es una herramienta que nos permite conectar sistemas basados en el protocolo HTTP, en la comunicación que se lleva a cabo entre el BackEnd y el FrontEnd del proyecto se hizo uso de esta tecnología ya que en el proceso de enviar y recibir datos se usa el formato JSON que nos permite una mejor manera de controlar los datos que se necesitan.

JPA

Java Persistence API (JPA) es una herramienta que nos permite convertir las entidades de una base de datos a un sistema orientado a objetos, permitiendo una mayor facilidad para su manejo.

Angular

Angular es un framework JavaScript con múltiples ventajas, ideal para aplicaciones basadas en la estructura MCV. Angular ofrece un fácil mantenimiento en proyectos que pueden a llegar a crecer conforme el desarrollo, además de implementar componentes, que son una porción de código que dan la oportunidad de ser reutilizables.

Bootstrap

Bootstrap es un framework que nos permite desarrollar interfaces web haciendo uso de CSS y JavaScript, con el objetivo de adaptar las interfaces al tamaño del dispositivo en el que se visualice el sistema.

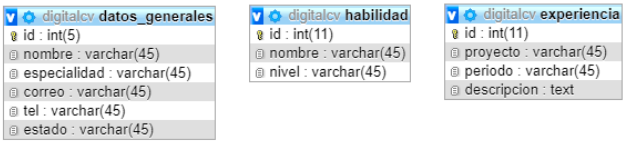


Ilustración 1 Diagrama Entidad - Relación

Para comenzar es necesario ejecutar los comandos contenidos en la carpeta database del repositorio.

Desarrollo BackEnd

- El desarrollo del sistema comienza en la elaboración del BackEnd por medio de Spring Boot, para eso es necesario entrar al sitio https://start.spring.io, se nos pedirá llenar la información del proyecto a crear, además de agregar las dependencias que se usaran, en este caso serán Spring Web, Spring Data JPA y MsQL Driver, una finalizado podemos dar clic en la opción de generar proyecto lo que nos descargará un archivo zip.

- Posteriormente descomprimiremos el archivo zip y podremos abrir el fichero en nuestro IDE, en este caso NetBeans, para comenzar con el desarrollo.

- Una vez abierto el proyecto se debe configurar el archivo "application.properties" para poder conectar el proyecto con la base de datos, para este caso debe llevar la siguiente configuración.

```
server.port = 9898


spring.datasource.url = jdbc:mysql://localhost:3306/digitalcv?serverTimezone=UTC
spring.datasource.username = cvuser
spring.datasource.password = cvuser
spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
```

- Para comenzar el desarrollo del BackEnd es necesario crear las entidades en la carpeta en que se encuentra el archivo "Aplication" del proyecto.
- Solamente es necesario crear tres clases, DatosGenerales, Experiencia y Habilidad. Para crear una entidad se debe seguir la sintaxis que se muestra a continuación.

```
@Entity
@Table(name = "datos_generales")
public class DatosGenerales {

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

    @Column
    private String nombre;

    @Column
    private String especialidad;

    @Column
    private String correo;

    @Column
    private String tel;

    @Column
    private String estado;

    public int getId() {
        return id;
    }

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

- Una vez creadas las entidades se debe continuar a crear las interfaces que contendrán los métodos para permitir la administración de los datos, las interfaces de tener la terminación “Repository”, a continuación, se muestra un ejemplo.

```
public interface HabilidadRepository extends Repository<Habilidad, Integer>{

   public List<Habilidad> findAll();

   public Habilidad findById(Integer id);

   public Habilidad save(Habilidad habilidad);

   public void delete(Habilidad habilidad);

}
```

- Una vez creadas las interfaces se deben desarrollar las clases controller que tendrán comunicación con el FrotEnd.
- Es necesario añadir al inicio de la clase los atributos: **RestController** que especifica que la aplicación hará uso de la tecnología REST, **CrossOrigin** que nos permitirá enlazar con la url del proyecto FrontEnd y **RequestMapping** que nos permite indicar el url del controlador.

```
@RestController
@CrossOrigin(origins = "http://localhost:4200", maxAge = 3600)
@RequestMapping("/api/sps/helloworld/v1/experiencias/")
public class ExperienciaController {
```

- Posteriormente es necesario tener acceso a los métodos de las interfaces que se crearon, para este paso es necesario incluir las interfaces de la siguiente manera.

```
@Autowired
ExperienciaRepository experienciaRepository;
```

- Una vez realizado el paso anterior se puede continuar a la creación de los métodos, los cuales deben tener una sintaxis específica para hacer uso de la tecnología REST.
- **Método GET:** @GetMapping().
- **Método POST:** @PostMapping(@RequestBody Object object).
- **Método PUT:** @PutMapping(@RequestBody Object object).
- **Método DELETE: @**DeleteMapping()

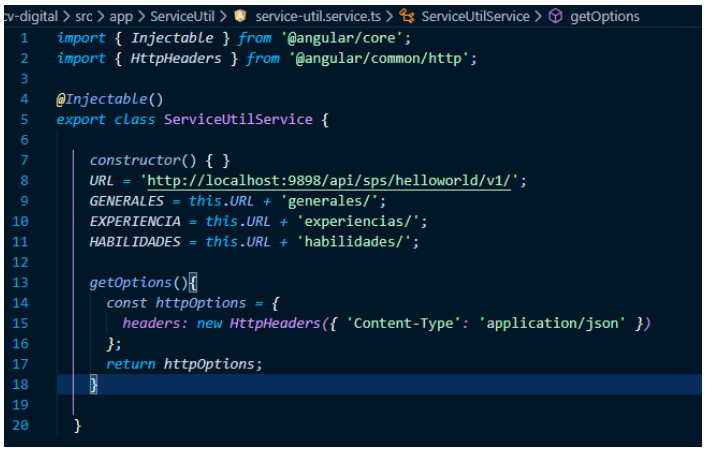
```
@GetMapping("obtenerExperiencia/{idExperiencia}")
public Experiencia obtenerDatosGenerales(@PathVariable("idExperiencia") int idExperiencia){
    return experienciaRepository.findById(idExperiencia);
}

@PostMapping("registrarExperiencia")
public Experiencia registrarExperiencia(@RequestBody Experiencia experiencia){
    return experienciaRepository.save(experiencia);
}

@PutMapping("editarExperiencia")
public Experiencia editarExperiencia(@RequestBody Experiencia experiencia){
    return experienciaRepository.save(experiencia);
}

@DeleteMapping("eliminarExperiencia/{idExperiencia}")
public void eliminarExperiencia(@PathVariable("idExperiencia") int idExperiencia){
    Experiencia experiencia = experienciaRepository.findById(idExperiencia);
    if(experiencia != null){
        experienciaRepository.delete(experiencia);
    }
}
```

- Finalizado estos podemos correr el proyecto BackEnd y nos desacrgara automáticamente las dependencias necesarias y en la terminal de nuestro IDE nos mostrara un mensaje de que el proyecto se encuentra funcionando correctamente.

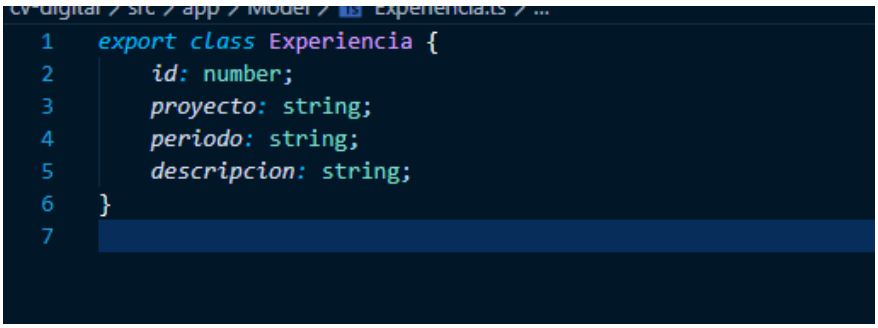
Desarrollo FrontEnd

- Para comenzar el desarrollo FrontEnd es necesario crear una carpeta en donde se quiere guardar el proyecto, una vez creada la carpeta mediante el IDE Visual Studio Code se debe abrir una terminal para ejecutar el comando: **ng new nombredelproyecto**.

- Una vez creado el proyecto crearemos dentro de la carpeta **app** del proyecto una carpeta llamada **ServiceUtil** que contendrá el archivo donde estableceremos las rutas de nuestro proyecto BakcEnd, para crear el archivo se debe usar el siguiente comando: **ng generate service service-util**. El archivo debe contener el código que se muestra en la imagen.

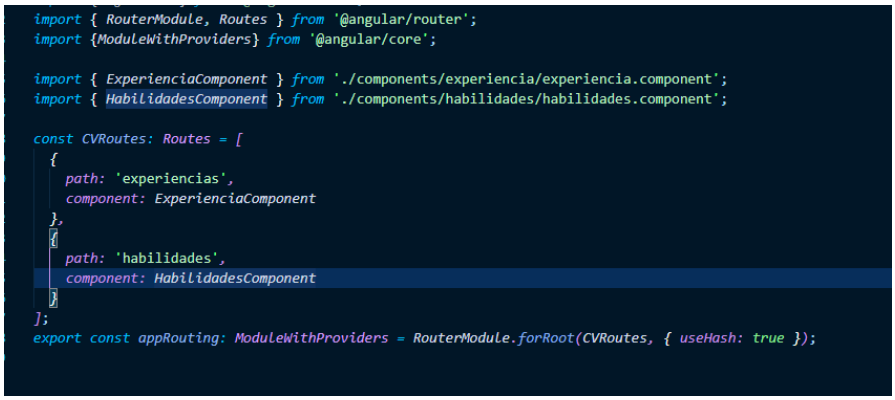


- Posteriormente se debe crear una carpeta llamada **Model** al mismo nivel que la carpeta anterior, esta carpeta contendrá los modelos de los objetos que se recibirán y se enviarán al BackEnd.



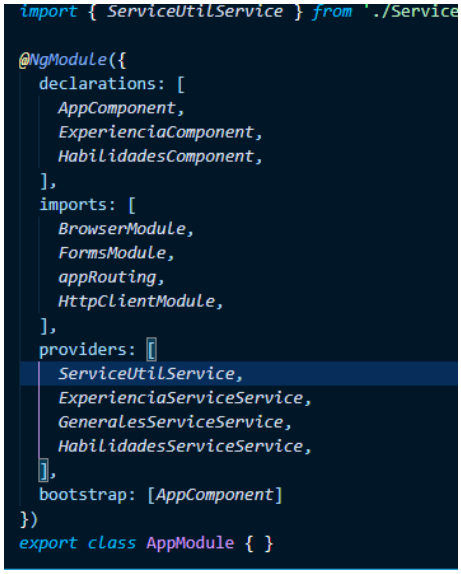
- Una vez creados los modelos se deben crear dos carpetas, **components y service**, al mismo nivel que las anteriores carpetas, en la carpeta de service se crearan las clases que se encargaran de tener los métodos de tipo REST, para crear los archivos de esta carpeta se debe usar el comando: **ng generate service nombre-archivo**.
- La carpeta components tendrá los componentes de la vista del proyecto, para crear los archivos de esta carpeta se debe usar el comando: **ng generate component nombre-componente**.
- El código que deben contener cada uno de los archivos está en el repositorio del proyecto.

- Para finalizar se debe configurar los archivos **app-routing.module.ts y app.module.ts**, en el primer archivo se deben agregar los componentes que se crearon como se muestra en la imagen.



```
import { RouterModule, Routes } from '@angular/router';
import {ModuleWithProviders} from '@angular/core';

import { ExperienciaComponent } from './components/experiencia/experiencia.component';
import { HabilidadesComponent } from './components/habilidades/habilidades.component';

const CVRoutes: Routes = [
  {
    path: 'experiencias',
    component: ExperienciaComponent
  },
  {
    path: 'habilidades',
    component: HabilidadesComponent
  }
];
export const appRouting: ModuleWithProviders = RouterModule.forRoot(CVRoutes, { useHash: true });
```

- El archivo **app.module.ts** se deben agregar los archivos service y components que se crearon como se muestra a continuación.



```
import { ServiceUtilService } from './Service

@NgModule({
  declarations: [
    AppComponent,
    ExperienciaComponent,
    HabilidadesComponent,
  ],
  imports: [
    BrowserModule,
    FormsModule,
    appRouting,
    HttpClientModule,
  ],
  providers: [
    ServiceUtilService,
    ExperienciaServiceService,
    GeneralesServiceService,
    HabilidadesServiceService,
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- Para finalizar solo queda correr el proyecto FrontEnd con el comando **ng serve** y se podrá hacer uso del sistema.