Recrea De Juerga

"Una aplicación que une fútbol, videojuegos y diseño personal."

Proyecto DAW – Jorge González Pérez

Tecnologías usadas

Angular

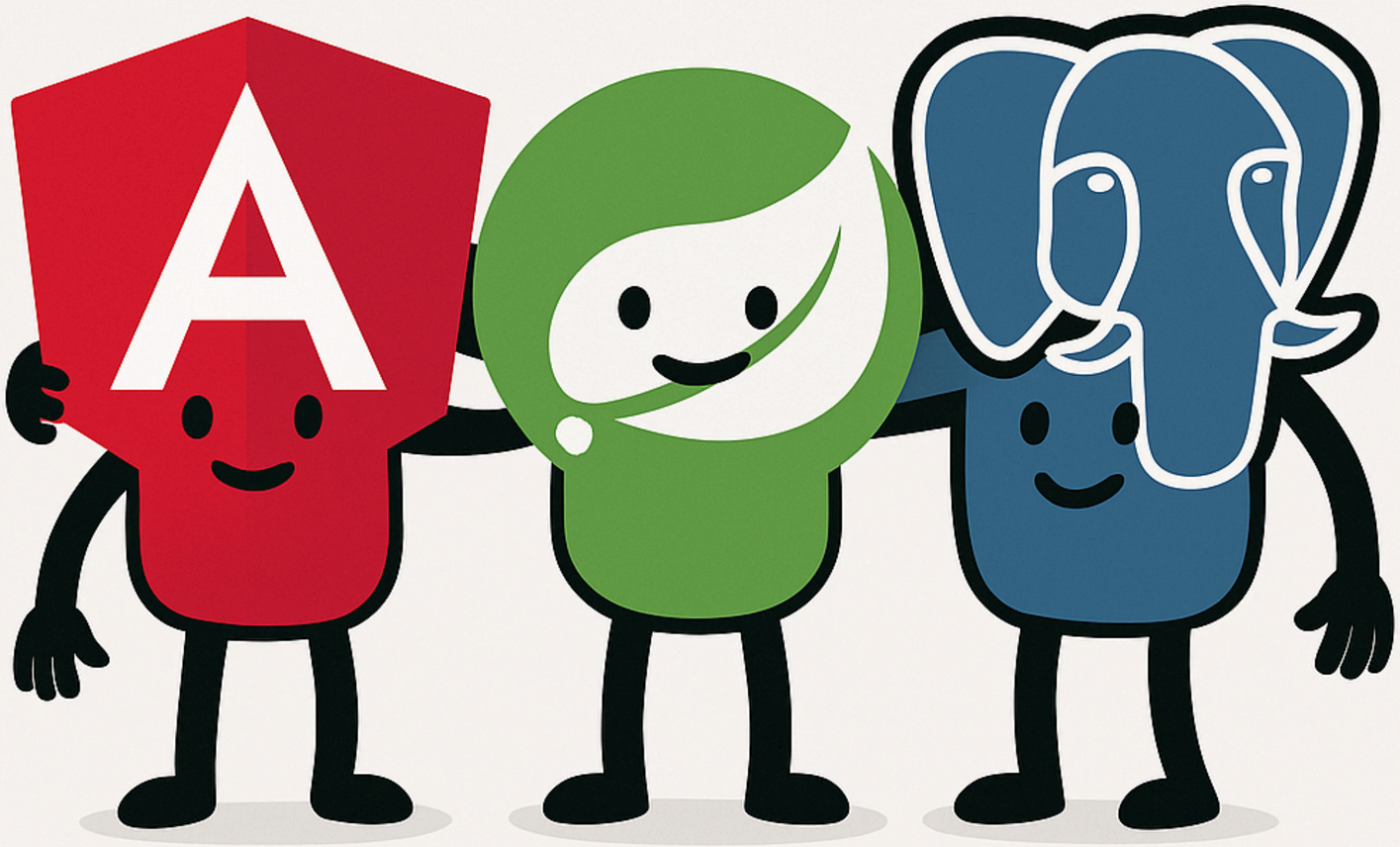
Framework frontend basado en TypeScript y SPA(Singleton Page Application)

Spring Boot

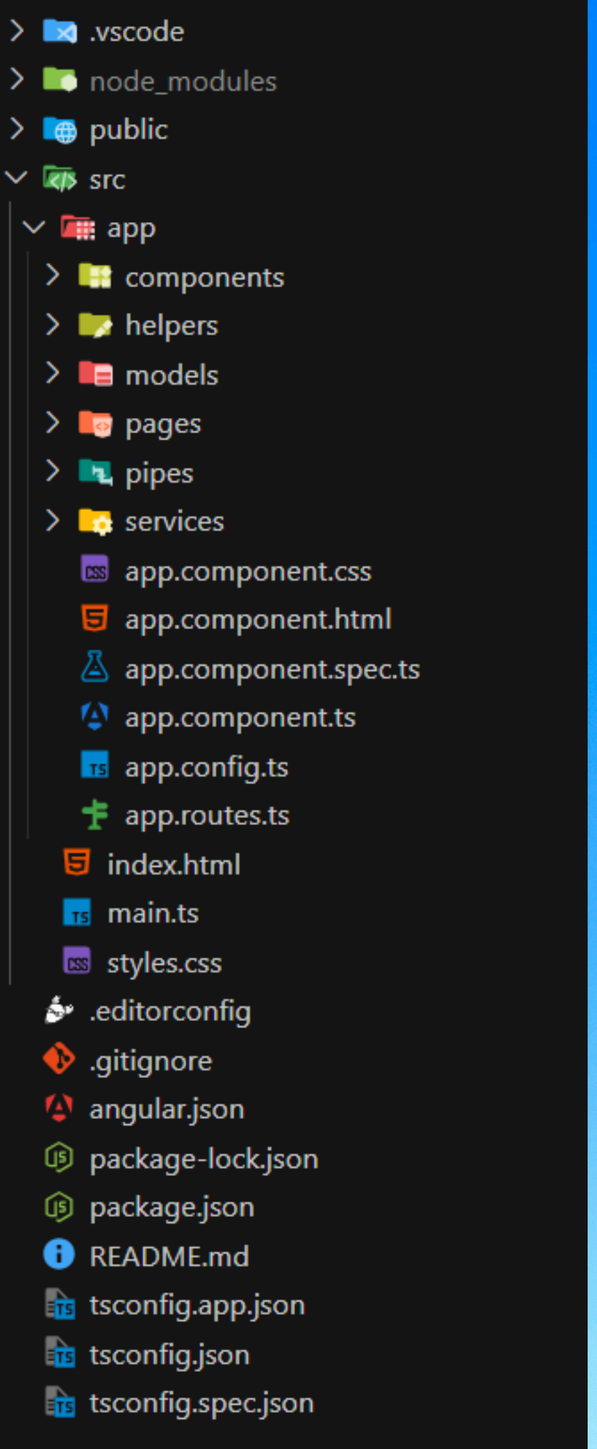
Framework backend basado en Java.
Arquitectura por capas.

PostgreSQL

Sistema gestor de base de datos relacionales
Soporta unas grandes cantidades de trabajo.



ANGULAR


.vscode
node_modules
public
src
app
components
helpers
models
pages
pipes
services
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json

Gestion de estados en Angular vs React

La propiedad del componente

```
mostrarDrawer: boolean = false;
```

Sin necesidad de métodos adicionales, cambia la propiedad de manera directa y sencilla

```
<div class="panel-footer">
  <span class="boton-a" (click)="mostrarDrawer = true">A</span> Añadir Jugador
  <span class="boton-b" (click)="volver()">B</span> Atrás
</div>
```

Necesita un método para poder modificar el estado

```
const [contador, setContador] = useState(0);
```

```
<button onClick={() => setContador(contador + 1)}>
```

ngOnInit() y ngDestroy()

```
export class PartidosComponent implements OnInit
```

```
ngOnInit(): void {
  const nombreEquipoURL = this.route.snapshot.params['nombreEquipo'];
  this.nombreEquipo = nombreEquipoURL.replace(/_/g, ' ');

  this.equipoService
    .getEquipoPorNombre(this.nombreEquipo)
    .pipe(take(1))
    .subscribe({
      next: (equipo: EquipoDTO) => {
        this.cargarPartidos(equipo);
      },
      error: (err: HttpErrorResponse) => {
        this.toastService.show({
          severity: 'error',
          summary: 'Error al obtener el equipo',
          detail: err.error?.detail || 'Ha ocurrido un error inesperado.',
        });
      },
    });
}
```

```
ngOnDestroy(): void {
  this.destroy$.next();
  this.destroy$.complete();
}
```

Modularidad de Angular

Aún mejor con los Standalone Components

```
@Component({
  selector: 'app-jugadores',
  standalone: true,
  imports: [CromoComponent, RouterOutlet, CommonModule],
  schemas: [CUSTOM_ELEMENTS_SCHEMA],
  templateUrl: './jugadores.component.html',
  styleUrl: './jugadores.component.css',
})
```

```
@Component({
  selector: 'app-menu',
  standalone: true,
  imports: [RouterLink,NgClass],
  templateUrl: './menu.component.html',
  styleUrl: './menu.component.css',
})
```

```
@Component({
  selector: 'app-footer',
  standalone:true,
  imports: [RouterLink],
  templateUrl: './footer.component.html',
  styleUrl: './footer.component.css',
})
```

```
@Component({
  selector: 'app-toast',
  standalone: true,
  templateUrl: './toast.component.html',
  styleUrls: ['./toast.component.css'],
})
```

Observables y Reactividad en Angular

```
export class EquipoService {
  private readonly api_url = 'http://localhost:8080/equipos';

  private equipoSeleccionadoSubject = new BehaviorSubject<EquipoDTO | null>(
    null
  );
  equipoSeleccionado$ = this.equipoSeleccionadoSubject.asObservable();

  constructor(private http: HttpClient) {}

  getTodosLosEquipos(): Observable<EquipoDTO[]> {
    return this.http.get<EquipoDTO[]>(this.api_url);
  }

  getEquipoPorNombre(nombre: string): Observable<EquipoDTO> {
    return this.http.get<EquipoDTO>(
      `${this.api_url}/nombre/${encodeURIComponent(nombre)}`
    );
  }

  getEquipo(equipoId: string): Observable<EquipoDTO> {
    return this.http.get<EquipoDTO>(`${this.api_url}/${equipoId}`);
  }

  altaEquipo(datosFormulario: EquipoFormularioDTO): Observable<EquipoDTO> {
    return this.http.post<EquipoDTO>(`${this.api_url}/alta`, datosFormulario);
  }
```

```
this.equipoService
  .getEquipoPorNombre(this.nombreEquipo)
  .pipe(take(1))
  .subscribe({
    next: (equipo: EquipoDTO) => {
      this.jugadorService
        .getJugadoresDeUnEquipo(equipo.id)
        .pipe(take(1))
        .subscribe({
          next: (jugadores: JugadorDTO[]) => {
            this.jugadores = jugadores.filter((j) => j.cromo != null);
          },
          error: (err: HttpErrorResponse) => {
            this.toastService.show({
              severity: 'error',
              summary: 'Error al obtener jugadores',
              detail: err.error?.detail || 'Error inesperado.',
            });
          },
        });
    },
```

Spring Boot

Ciclo de vida simplificado de una petición

Cliente ⟷ **Controlador** ⟷ **Servicio** ⟷ **Repositorio**

Petición del cliente en mi caso de Angular

```
getEquipo(equipoId: string): Observable<EquipoDTO> {
  return this.http.get<EquipoDTO>(`${this.api_url}/${equipoId}`);
}
```

Interceptor que añade el token en la cabecera

```
provideHttpClient(
  withInterceptors([authInterceptor])
),
```

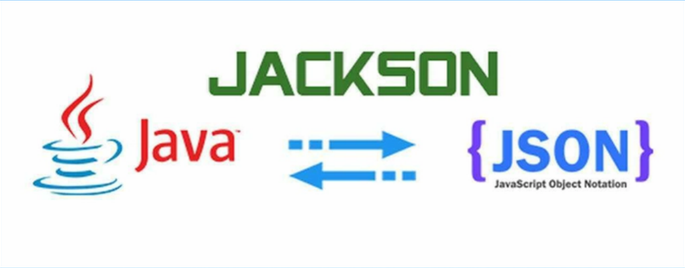
En mi caso, todos los métodos post, put y delete requieren de token

```
altaEquipo(datosFormulario: EquipoFormularioDTO): Observable<EquipoDTO> {
  return this.http.post<EquipoDTO>(`${this.api_url}/alta`, datosFormulario);
}
```

Método de un controlador

```
@GetMapping("/nombre/{nombreEquipo}")
public ResponseEntity<EquipoDTO> getEquipoByNombre(
        @Parameter(description = "Nombre exacto del equipo (puede contener espacios)")
        @PathVariable String nombreEquipo) {
    return ResponseEntity.ok(equipoServicio.getEquipoByNombre(nombreEquipo));
}
```

Serialización y deserialización



Método de un servicio

```
public EquipoDTO getEquipoByNombre(String name){  1 usage  jodric10
    return repo.buscarPorNombre(name).map(EquipoMapper::toEquipoDTO).orElseThrow(() -> new EquipoNoEncontradoException(name));
}
```

Método de un repositorio

```
@Repository  4 usages  Jorge González Pérez +1
public interface EquipoRepositorio extends JpaRepository<Equipo, UUID> {

    @Query("""  2 usages  Jorge González Pérez
            SELECT e
            FROM Equipo e
            WHERE LOWER(e.nombre) = LOWER(:nombre)
            """)
    Optional<Equipo> buscarPorNombre(@Param("nombre") String nombre);
```

Spring Boot

Lombok

Entidad

```
@Entity   jodric10 +1
@Table(name = "equipos")
@Getter
@Setter
@ToString
@NoArgsConstructor
@AllArgsConstructor
@Builder
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")
public class Equipo {
    @Id
    @UuidGenerator
    private UUID id;

    @OneToOne(mappedBy = "equipo", cascade = CascadeType.ALL, orphanRemoval = true)
    private Clasificacion clasificacion;

    @Column(name = "nombre", unique = true ,nullable = false, length = 100)
    private String nombre;

    @Column(name = "url_logo", columnDefinition = "TEXT")
    private String  url_logo;

    @Column(name = "eslogan", length = 50)
    private String  eslogan;

    @Column(name = "descripcion", columnDefinition = "TEXT")
    private String  descripcion;

    @Column(name = "url_equipo", columnDefinition = "TEXT")
```

DTO

```
@Getter   28 usages   Jorge González Pérez +1
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class EquipoDTO {
    private UUID id;
    private String nombre;
    private String url_logo;
    private String eslogan;
    private String descripcion;
    private String url_equipo;
}
```

```
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")
public class Clasificacion {
    @Id
    @Column(name = "equipo_id", nullable = false)
    private UUID id;

    @OneToOne
    @MapsId
    @JoinColumn(name = "equipo_id", foreignKey = @ForeignKey(name = "fk_equipo_clasificacion"))
    private Equipo equipo;
```

Spring Boot

Mapper

```
public class EquipoMapper {  13 usages  jodric10 +1


    public static EquipoDTO toEquipoDTO(Equipo equipo){  5 usages  Jorge González Pérez +1
        if (equipo == null) return null;
        return new EquipoDTO(equipo.getId(),equipo.getNombre(),equipo.getUrl_logo(), equipo.getEslogan(),equipo.getDescripcion(),equipo.getUrl_equipo());
    }
```

@RestControllerAdvice

```
@RestControllerAdvice  no usages  Jorge González Pérez +1
public class GlobalErrorControlador extends ResponseEntityExceptionHandler {

    @ExceptionHandler(EquipoNoEncontradoException.class)  no usages  Jorge González Pérez
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public ProblemDetail handleEquipoNoEncontrado(EquipoNoEncontradoException ex,  WebRequest request){
        ProblemDetail resultado= ProblemDetail.forStatusAndDetail(HttpStatus.NOT_FOUND, ex.getMessage());
        resultado.setTitle("Equipo no encontrado");
        resultado.setType(URI.create("https://api.recreadejuerga.com/errores/equipo-no-encontrado"));
        resultado.setInstance(URI.create(request.getDescription( includeClientInfo: false).replace( target: "uri=",  replacement: "")));
        return resultado;
    }
```

Spring Boot vs Laravel

Java

JPA
HIBERNATE

php

ELOQUENT

Gracias por tu atención

Bibliografía

Midudev

OpenWebinars

Pablo San Segundo

Alexander Paredes

Sergio Pérez González