Informe UT5 - TFU

Identificación de los endpoints para cumplir con los casos de uso planteados en el Figma

Para el armado de la api implementamos los 3 primeros endpoints, ya que el resto eran de complejidad similar y consideramos que no agregaba valor a la propuesta del TFU, de todas formas quedaron definidos todos los contratos en este documento a modo de relevamiento.

CU 1.1: Login

Login	
Verbo	**`POST`**
path	`user/auth/login`
queryParams	

body	`{`
	`  “user” : “ejemplo”,`
	`  “password” : “ejemplo”`
	`}`

response codes	`200 | 400 | 401 …`
response body	`{ “token” : “ejemplo” }`

CU 1.2: Dado un juez, Devolver lista de competencias a puntuar

Dado un juez, Devolver lista de competencias a puntuar	
Verbo	**`GET`**
path	`/judges/{judgeId}/competitions`
queryParams	
body	
response	`200 | 400 | 401 …`

codes	
response body	{ “competitions” : [{“competitionId” : “1232321231”, “name” : “100mts libre”, “date” : “19/06/24” }, {“competitionId” : “1232321212”, “name” : “100mts mariposa”, “date” : “19/06/24”} }] }

CU 1.3: Dado una competencia, devolver la lista de participantes

Dado una competencia, devolver la lista de participantes	
Verbo	**GET**
path	sports/competitions/{competitionId}/participants
queryParams	
body	
response codes	200 | 400 | 401 …
response body	{ “results” : [{“athleteId” : “1231212113”, “picture” : “url picture example”, “name” : “H. Barbosa”, “country” : “Uruguay”, “time” : “1:26”, “confirmed” : “NotYet” | “Confirmed” | “Declined” }, {“athleteId” : “1232321231”, “picture” : “url2 picture example”, “name” : “M. Perez”, “country” : “USA”, “time” : “1:17”, “confirmed” : “NotYet” | “Confirmed” | “Declined” }] }

CU 1.4: Dado un atleta con sus datos, aceptar o rechazar puntuación

Dado un atleta con sus datos, aceptar o rechazar puntuación	
Verbo	**POST**
path	competitions/{competitionId}/participants/{athleteId}/score
queryParams	

body	{
	"score": "number",
	"accepted": "boolean"
	}

response codes	200 | 400 | 401 …

response body	{
	"message": "string"
	}

CU 2.1: Devolver lista de deportes

Devolver lista de deportes	
Verbo	**GET**
path	/sports
queryParams	
body	
response codes	200 | 400 | 401 …

response body	{ "results" :
	[
	{
	"sportId": "string",
	"name": "string"
	}
	]
	}

CU 2.2: Dado un deporte, devuelve una lista con las competencias para elegir qué resultado ver

Dado un deporte, devuelve categorías	
Verbo	**GET**
path	/sports/{sportId}/competitions
queryParams	
body	
response codes	200 | 400 | 401 …

response body	[
	  {
	  "competitionId": "string",
	  "nameCategory": "string",
	  "nameSubcategory": "string",
	  "date": "string",
	  },
	  {
	  "competitionId": "string",
	  "nameCategory": "string",
	  "nameSubcategory": "string",
	  "date": "string",
	  }
	]

CU 2.3 Dado una competencia competencia devuelve los resultados

Dado una subcategoría, devuelve las fechas de desa competencia	
Verbo	**GET**
path	/competitions/{competitionId}/results
queryParams	
body	
response codes	200 | 400 | 401 …
response body	[{

	<pre>"athleteId": "123123123",
	“date” : “1/1/2024”
	"name": "Helen Barbosa",
	"score": "1:07",
	"position": "1"
	}
	]</pre>

Mejoras al Figma

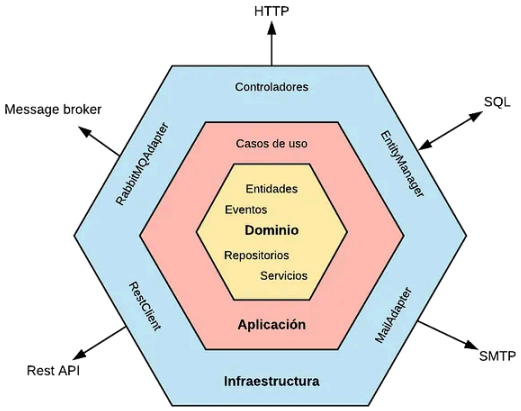
Link al Figma - UT3 TFU

Mejoras que implementamos en base al feedback recibido durante la presentación en clase del Figma:

1. **Reducción de la navegación**: Simplificar la navegación para que los usuarios puedan acceder a los resultados más rápidamente.
2. **Tamaño de iconos**: Reducir el tamaño de los iconos para que no ocupen tanto espacio visual.
3. **Utilizar dispositivos modernos**: Incluir dispositivos más modernos.
4. **Claridad en indicadores**: Mejorar los indicadores de estado (tick y cruz) para que sean más claros y no dejen lugar a interpretaciones erróneas.
5. **Comentarios en evaluaciones**:Dejar comentarios sobre por qué algo no cumplió con lo que esperaban los usuarios.
6. **Ajuste de paddings**: Revisar y ajustar los paddings en las secciones de deportes para mejorar la apariencia y la usabilidad.

Organización de la API

- Para la api usamos Java 21 con SpringBoot 3 como framework.
- El proyecto está organizado en 3 capas, siguiendo una arquitectura Clean, la arquitectura Hexagonal.
- La misma utiliza Domain Driven Design como patrón de diseño principal, pretende separar la lógica de negocio y dominio de la capa de implementación o infraestructura de modo que sea más escalable y que tu lógica de negocio no dependa de implementaciones concretas.
- Los packages son:
    - Dominio
    - Aplicación
    - Infraestructura



Patrón de diseño utilizados

Builder

Este es un patrón de diseño creacional que nos permite construir objetos complejos paso a paso. El patrón nos permite producir distintos tipos y representaciones de un objeto empleando el mismo código de construcción.
Los utilizamos para la creación de Athlete y Judge.
Este patrón nos permite crear varios Athlete y Judge distintos que implementen la misma serie de pasos de construcción, pero de forma diferente.
Por ejemplo, diferentes jueces pueden especializarse en distintos deportes que deben calificar, y varios atletas pueden participar en diferentes deportes.

Refactorización

Algún patrón que podría ser implementado en una futura refactorización
sería el patrón Strategy.

Este es un patrón de diseño de comportamiento que te permite definir una familia de algoritmos, colocar cada uno de ellos en una clase separada y hacer sus objetos intercambiables.
El patrón Strategy sería especialmente útil para calcular la calificación de un atleta en un deporte determinado. Con este patrón, se pueden crear diferentes estrategias de calificación para distintos deportes, y cambiar la estrategia utilizada sin modificar el código del cliente que usa estas estrategias.