

Sergio Misas Vírveda

Informe de desarrollo

FutDAM

The screenshot shows the FutDAM application window. It features a menu bar with 'Archivo' and 'Ayuda'. Below the menu is a search bar with 'Nombre' and a dropdown for 'Equipo'. A table lists players with columns for 'Nombre', 'Apellidos', and 'Equipo'. To the right, there are form sections for 'Info Jugador' and 'Info Equipo', each with input fields and 'Crear', 'Editar', and 'Borrar' buttons.

Nombre	Apellidos	Equipo
Thibaut	Courtois	Real Madrid
Robert	Lewandowski	FC Barcelona
Jan	Oblak	Atletico Madrid
Jon	Karrikaburu	Leganes SAD
Leon	Goretzka	Bayern Munchen
Vincent	Aboubakar	Besiktas
Anderson	Talisca	Al-Nassr

Info Jugador

Nombre:
Apellidos:
Edad:
Dorsal:
Nacionalidad:
Apodo:
Posicion:

Info Equipo

Nombre:
Pais:
Año fund.:
Liga:

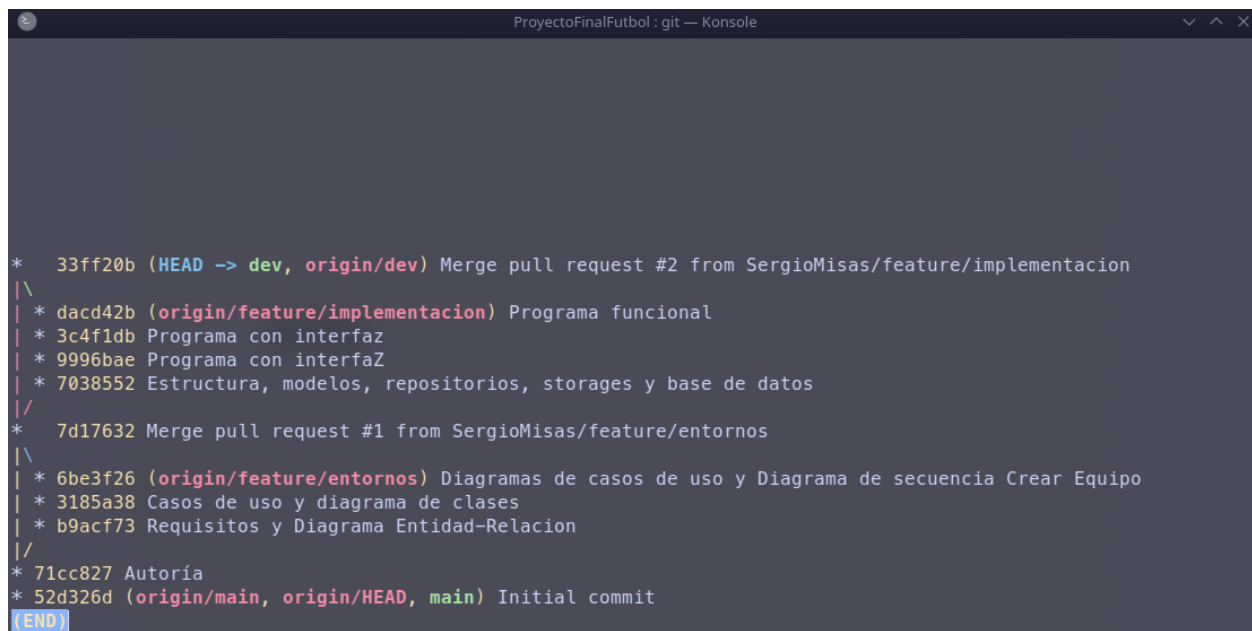
Introducción

Hay múltiples webs que acceden a bases de datos de jugadores hoy en día, y como apasionado del fútbol que soy he elegido desarrollar un pequeño proyecto de gestión de jugadores y equipos al que he llamado FutDam.

Introducción	1
Git Flow	3
Requisitos funcionales	5
Requisitos no funcionales	6
Requisitos de información	7
Digrama Entidad-Relación	8
Diagrama de Clases	9
Diagramas de Casos de Uso	10
Gestión de jugador	10
Gestión de equipo	10
Diagrama de secuencia de caso de uso Crear equipo	11
Casos de uso Jugador	12
Casos de uso Equipo	13
Implementación	15
Modelos	16
Inyección de dependencias	17
Vistas	18
Principal	18
Acerca de	18
Exportar	19
Crear Jugador	19
Editar Jugador	20
Crear Equipo	20
Editar Equipo	21
Controladores	21
ViewModel	23
Repositorios	23
Servicio de exportación a Json	24
Configuración de la aplicación	26
Testing	27

Git Flow

Se trabaja con una rama main y otra dev, de las cuales salen las ramas feature de Implementacion y de Entornos.



```
ProyectoFinalFutbol : git — Konsole

* 33ff20b (HEAD -> dev, origin/dev) Merge pull request #2 from SergioMisas/feature/implementacion
|\
| * dacd42b (origin/feature/implementacion) Programa funcional
| * 3c4f1db Programa con interfaz
| * 9996bae Programa con interfaz
| * 7038552 Estructura, modelos, repositorios, storages y base de datos
|/
* 7d17632 Merge pull request #1 from SergioMisas/feature/entornos
|\
| * 6be3f26 (origin/feature/entornos) Diagramas de casos de uso y Diagrama de secuencia Crear Equipo
| * 3185a38 Casos de uso y diagrama de clases
| * b9acf73 Requisitos y Diagrama Entidad-Relacion
|/
* 71cc827 Autoría
* 52d326d (origin/main, origin/HEAD, main) Initial commit
(END)
```

Requisitos funcionales

Desde la concepción de la idea del proyecto se han tenido en claro los puntos fundamentales de este proyecto:

- Se deben crear, editar y borrar jugadores y equipos
- Se debe crear una base de datos para almacenar la información
- Se debe crear un programa de escritorio que represente los datos de forma gráfica y permita acceder de forma interactiva a los métodos

Requisitos funcionales

RF-01: Crear una base de datos

RF-02: Actualizar BBDD

RF-03: Realizar búsquedas en la BBDD

RF-04: Crear un jugador

RF-05: Editar un jugador

RF-06: Borrar un jugador

RF-07: Creación de modelos solicitados -> Jugador, Equipo

RF-08: Generar un informe

RF-09: Exportar un informe

RF-10: Realizar una interfaz gráfica

RF-11: Crear un equipo

RF-12: Borrar un equipo

RF-13: Editar un equipo

RF-14: Poner un jugador en agentes libres

Requisitos no funcionales

Es imprescindible que se exportaran los datos en algún tipo de formato así que se eligirá JSON ya que es claro.

También hay jugadores que no están en un equipo, estos estarán el equipo de jugadores libres el cual no tiene tamaño máximo y tiene especificaciones especiales.

Requisitos no funcionales

RNF-01: Verificar la Información a la hora de de gestionar un jugador

RNF-02: Filtrar por nombre

RNF-03: Filtrar por equipo

RNF-04: Filtrar por posicion

RNF-05: Verificar la información a la hora de gestionar un jugador

RNF-06: Verificar la información a la hora de gestionar un equipo

RNF-07: Generación de informes de equipo en JSON

RNF-08: Tener pestaña acerca de

RNF-09: Tener vistas distintas para creación y edición de jugador y equipo

RNF-10: El "equipo" agentes libres no va a cumplir las normas del resto de equipos: No tiene capacidad máxima , no se les pude poner dorsal a los jugadores y no se puede borrar ni editar.

Requisitos de información

Aquí vienen los distintos datos de las clases que van a existir en el programa:

Requisitos de información

RI-01: Datos del equipo

- Id [Int,Único]
- Nombre [String]
- Pais [String]
- Año Fundación[Int(4)]
- Liga[String]

RI-02: Datos del jugador

- Id [Int, Único]
- Nombre[String]
- Apellidos[String]
- Edad[Int]
- Nacionalidad[String]
- Dorsal[Int]
- Apodo[String]
- Posición[enum]
- Id del equipo[Long]

RI-03: Tipos de posición

- PORTERO
- DEFENSA
- CENTROCAMPISTA
- DELANTERO

Digrama Entidad-Relación

Ya que se va a utilizar una base de datos para almacenar la información, es necesario hacer un diagrama entidad relación para estructurarla bien.

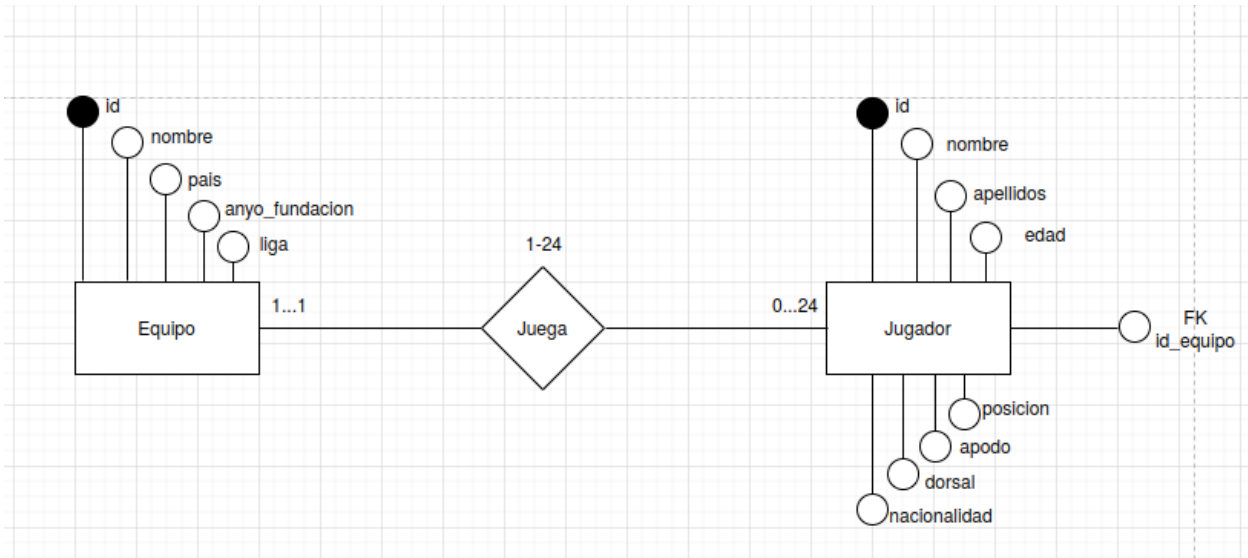
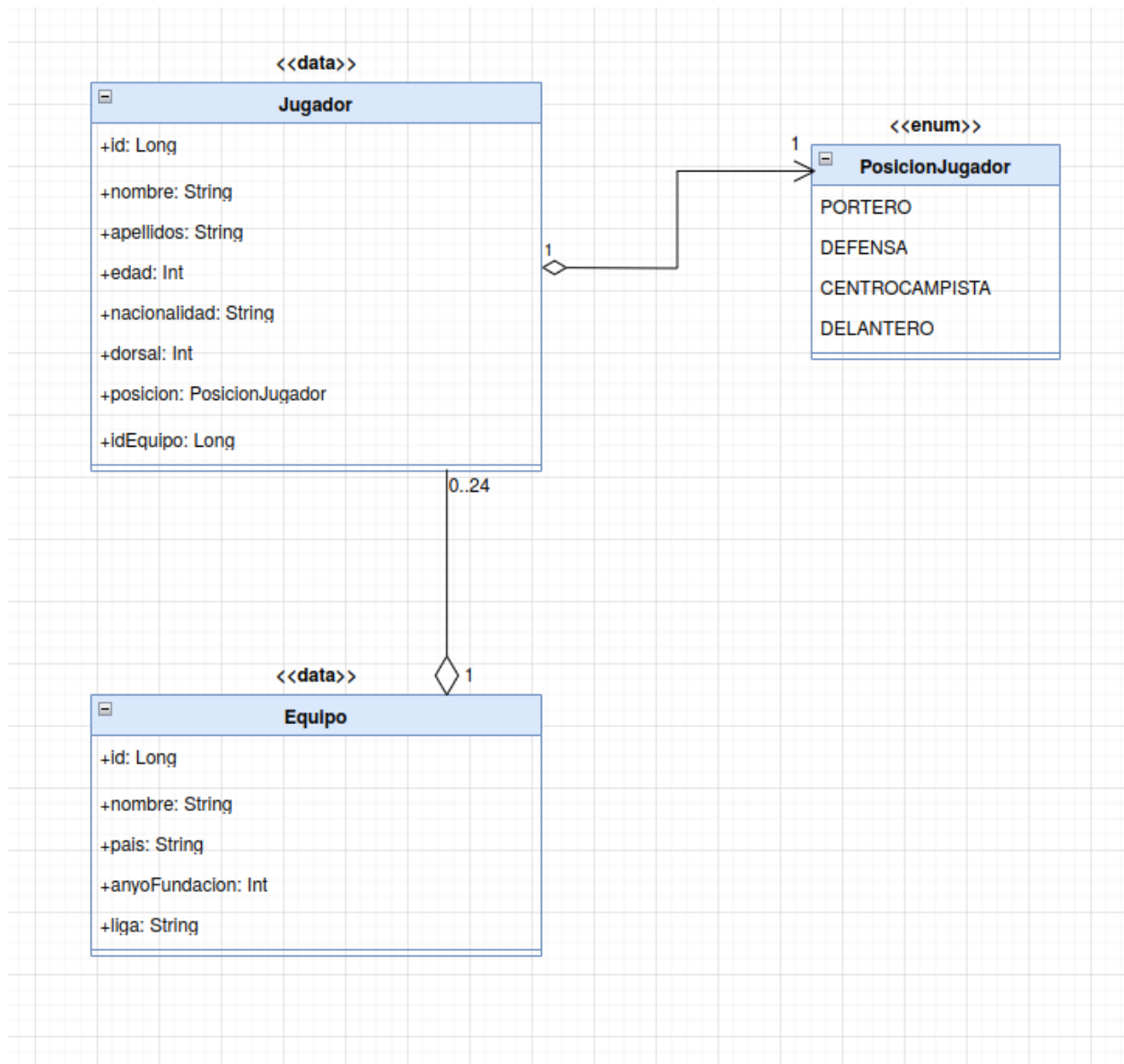


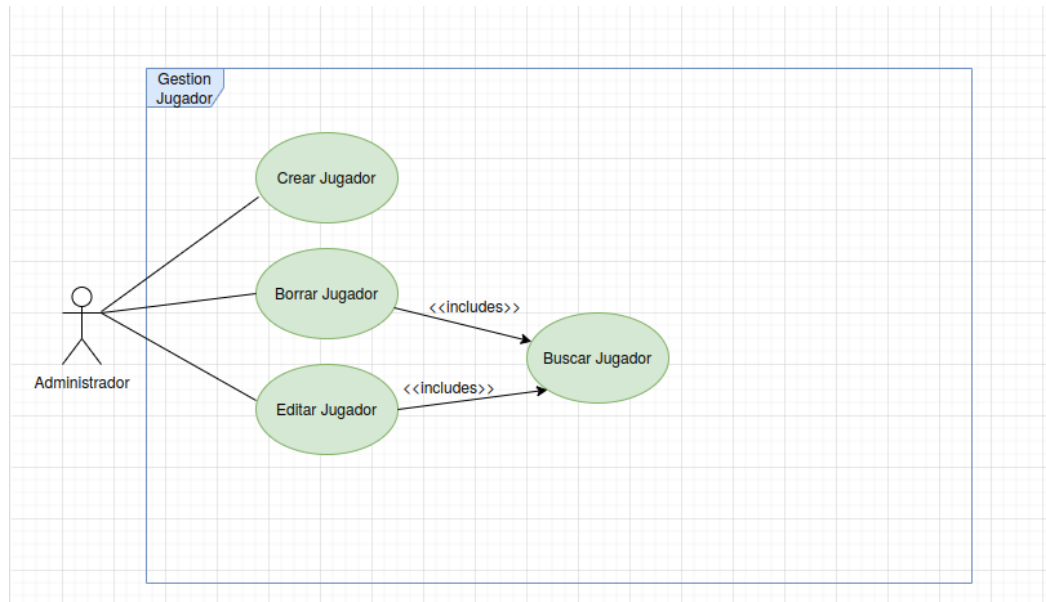
Diagrama de Clases

Una vez hecho esto toca realizar el diagrama de clases para la estructura del programa



Diagramas de Casos de Uso

Gestión de jugador



Gestión de equipo

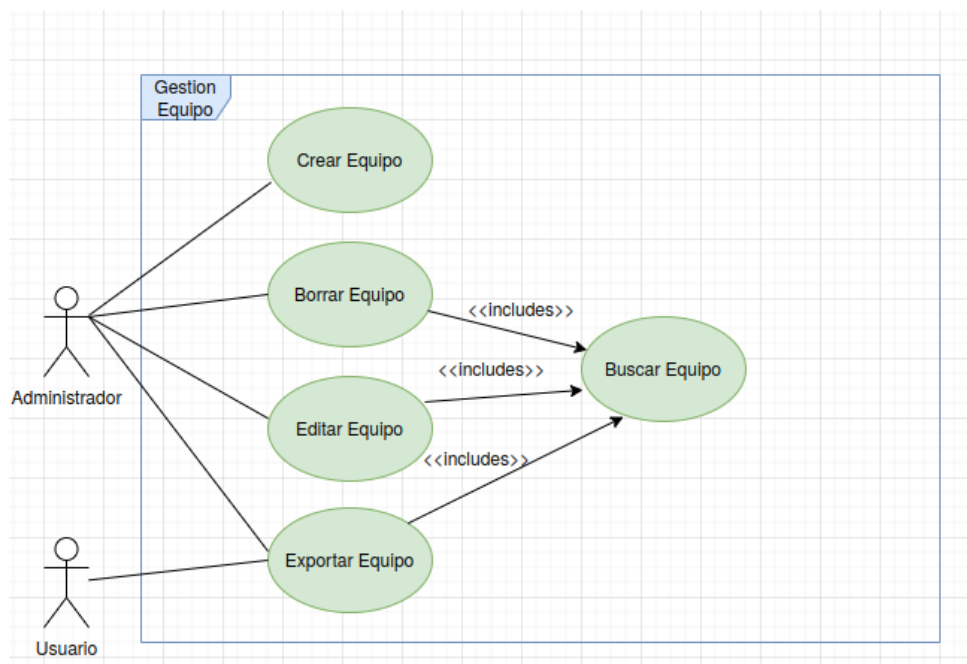
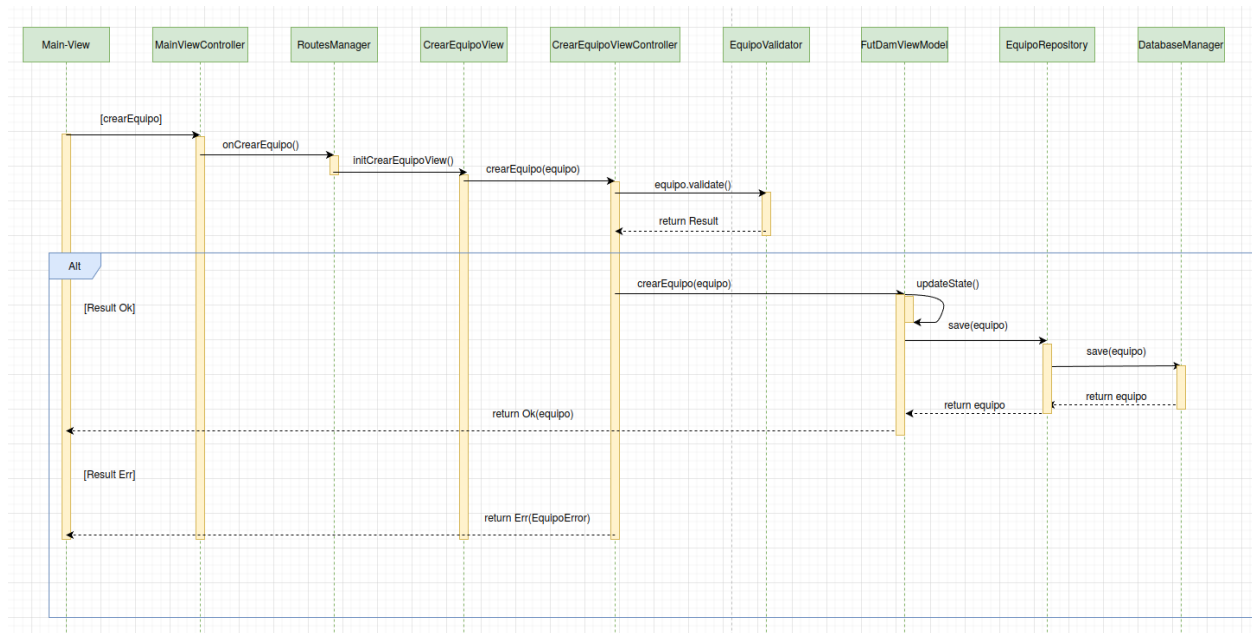


Diagrama de secuencia de caso de uso Crear equipo



Casos de uso Jugador

Se han documentado los casos de uso de crear, borrar y editar jugador.

He aquí un ejemplo:

Borrar jugador

Resumen

Este caso de uso hace que un administrador pueda borrar un jugador que exista en la BBDD

Código

CU-02

Nombre

Borrar jugador

RF

RF-06

Actor(es)

Administrador

Precondiciones

- 1 El administrador ha encendido el programa
- 2 El administrador ha seleccionado un jugador de la tabla
- 3 El administrador pulsa el botón de borrar jugador

Postcondiciones

- 1 Se ha borrado el jugador de la BBDD

Secuencia de pasos

- Flujo principal

- 1 Sale una ventana de confirmación
- 2 El administrador pulsa continuar
- 3 El sistema borra al jugador de la base de datos, fin de caso de uso

- Flujo Alternativo

- 2.1 Se pulsa el botón de volver y se cancela el caso de uso

Casos de uso Equipo

Se han documentado los mismos casos de uso que en jugador pero con el añadido de exportar equipo.

Exportar equipo

Resumen

Este caso de uso hace que un cliente pueda exportar un equipo que exista en la BBDD

Código

CU-07

Nombre

Exportar equipo

RF

RF-08

RF-09

Actor(es)

Cliente

Precondiciones

- 1 El cliente ha encendido el programa
- 2 El cliente ha pulsado la opción de exportar equipo

Postcondiciones

- 1 Se ha exportado el equipo en un formato seleccionado

Secuencia de pasos

- Flujo principal

- 1 El sistema abre una ventana para seleccionar formato y equipo
- 2 El cliente selecciona formato y equipo
- 3 El cliente pulsa aceptar
- 4 El sistema exporta el equipo al formato seleccionado
- 5 El sistema cierra la ventana, fin del caso de uso

- Flujo Alternativo

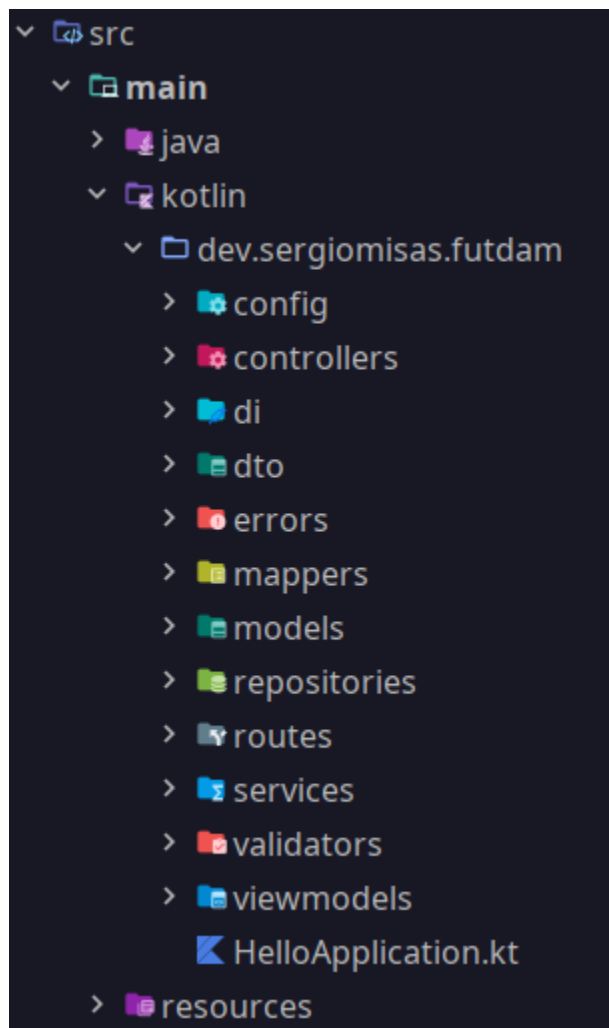
-
- 1.1 Se pulsa el botón de volver y se cancela el caso de uso
 - 2.1 El cliente elige entre las distintas formas
 - 2.1.1 El cliente exporta en JSON
 - 2.1.2 El cliente exporta en XML

Implementación

Se han seguido las instrucciones del diseño y el análisis para realizar la implementación del programa.

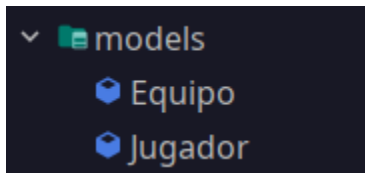
Para la interfaz gráfica se ha decidido utilizar JavaFX

También se ha hecho uso de la estructura de Model-View-ViewModel para este proyecto.



Modelos

Se han implementado las clases de Jugador y Equipo



```
SergioMisas
data class Equipo( Tipo: In word 'Equipo'.
    val id: Long = 0L,
    val nombre: String, Tipo: In word 'nombre'.
    val pais: String,
    val anyoFundacion: Int, Tipo: In word 'anyo'. Tipo: In word 'Fundacion'.
    val liga: String Tipo: In word 'liga'.
)
```

```
SergioMisas
data class Jugador( Tipo: In word 'Jugador'.
    val id: Long = 0L,
    val nombre: String, Tipo: In word 'nombre'.
    val apellidos: String, Tipo: In word 'apellidos'.
    val edad: Int, Tipo: In word 'edad'.
    val nacionalidad: String, Tipo: In word 'nacionalidad'.
    val dorsal: Int = 0,
    val apodo: String, Tipo: In word 'apodo'.
    val posicion: PosicionJugador, Tipo: In word 'posicion'.
    val idEquipo: Long = 0 Tipo: In word 'Equipo'.
) {
    SergioMisas
    enum class PosicionJugador { Tipo: In word 'PosicionJugador'
        | PORTERO, DEFENSA, CENTROCAMPISTA, DELANTERO Ty
    }
}
```

Inyección de dependencias

Se ha utilizado Koin para la inyección de dependencias

```
package dev.sergiomisas.futdam.di

import ...

val koinModule = module { this: Module
    single { AppConfig() }
    single { DatabaseManager(get()) }
    single<JugadorRepository> { JugadorRepositoryImpl(get()) }
    single<EquipoRepository> { EquipoRepositoryImpl(get()) }
    single { JsonStorageService() }
    single { FutDamViewModel(get(), get(), get()) }
}
```

```
package dev.sergiomisas.futdam

> import ...

@SergioMisas
class HelloApplication : Application() {
    @SergioMisas
    override fun start(stage: Stage) {
        startKoin { this: KoinApplication
            printLogger()
            modules(koinModule)
        }
        RoutesManager.apply { this: RoutesManager
            app = this@HelloApplication
        }.run { this: RoutesManager
            initMainView(stage)
        }
    }
}

@SergioMisas
fun main() {
    Application.launch(HelloApplication::class.java)
}
```

Vistas

Principal

Nombre	Apellidos	Equipo
Thibaut	Courtois	Real Madrid
Robert	Lewandowski	FC Barcelona
Jan	Oblak	Atletico Madrid
Jon	Karrikaburu	Leganes SAD
Leon	Goretzka	Bayern Munchen
Vincent	Aboubakar	Besiktas
Anderson	Talisca	Al-Nassr

Info Jugador

Nombre:

Apellidos:

Edad:

Dorsal:

Nacionalidad:

Apodo:

Posicion:

Info Equipo

Nombre:

Pais:

Año fund.:

Liga:

Acerca de



Exportar



Exportar equipo

Elija un equipo para exportar en formato JSON

Equipo a seleccionar

Numero de jugadores

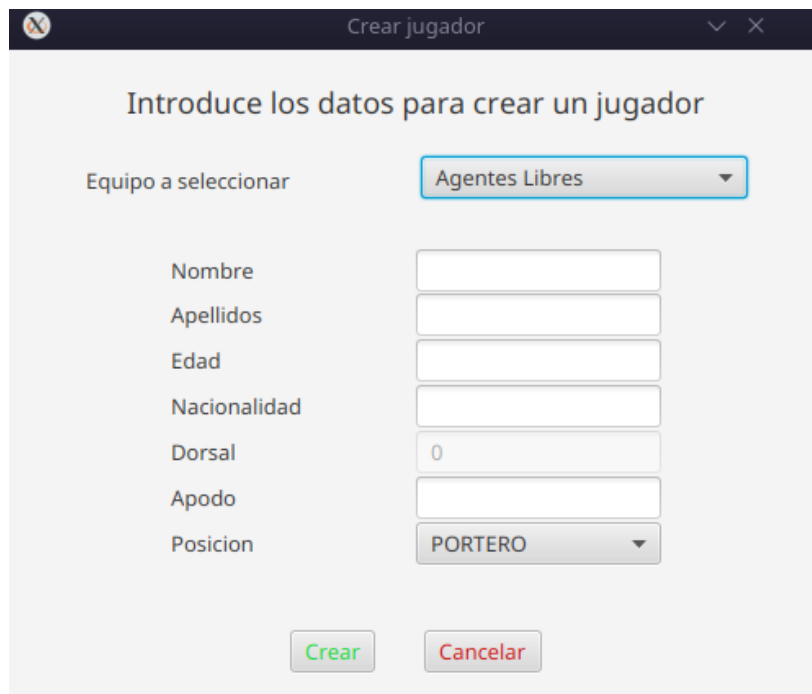
País

Año de Fundacion

Liga

This is a modal dialog box titled "Exportar equipo". It contains a title bar with a close button and a dropdown icon. The main content area has a heading "Elija un equipo para exportar en formato JSON". Below the heading, there are five input fields: "Equipo a seleccionar" (a dropdown menu), "Numero de jugadores", "País", "Año de Fundacion", and "Liga". At the bottom, there are two buttons: "Exportar" (green text) and "Cancelar" (red text).

Crear Jugador



Crear jugador

Introduce los datos para crear un jugador

Equipo a seleccionar

Nombre

Apellidos

Edad

Nacionalidad

Dorsal

Apodo

Posicion

This is a modal dialog box titled "Crear jugador". It contains a title bar with a close button and a dropdown icon. The main content area has a heading "Introduce los datos para crear un jugador". Below the heading, there are eight input fields: "Equipo a seleccionar" (a dropdown menu with "Agentes Libres" selected), "Nombre", "Apellidos", "Edad", "Nacionalidad", "Dorsal" (with "0" entered), "Apodo", and "Posicion" (a dropdown menu with "PORTERO" selected). At the bottom, there are two buttons: "Crear" (green text) and "Cancelar" (red text).

Editar Jugador

Editar jugador

Modifique los datos del jugador

Equipo a seleccionar: Al-Nassr

Nombre: Anderson

Apellidos: Talisca

Edad: 28

Nacionalidad: Brasil

Dorsal: 94

Apodo:

Posicion: CENTROCAMP...

Editar Cancelar

Crear Equipo

Crear equipo

Introduce los datos para crear un equipo

Nombre:

Pais:

Año de fundacion:

Liga:

Crear Cancelar

Editar Equipo



Editar equipo

Modifique los datos del equipo

Nombre	Al-Nassr
País	Arabia Saudí
Año de fundacion	1955
Liga	Saudi Pro League

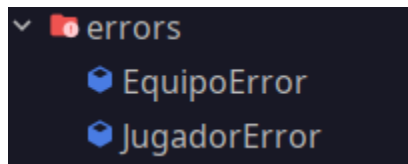
Editar Cancelar

Controladores

Cada vista tiene su controlador asociado excepto la acerca-de-view



Los controladores utilizan railway oriented programming para lanzar results o errores personalizados.



```
equipo.validate().andThen { viewModel.editarEquipo(it) }  
    .onSuccess { equipoNuevo → Typo: In word 'equipo'.  
        viewModel.state.value = viewModel.state.value.copy(  
            |   equipos = viewModel.state.value.equipos - equipoActual + equipoNuevo  
            |  
            |  
            |  
            |  
        )  
        val alert = Alert(Alert.AlertType.INFORMATION)  
        alert.title = "Editar Equipo" Typo: In word 'Editar'. Typo: In word 'Equipo'.  
        alert.headerText = "Equipo editado correctamente" Typo: In word 'Equipo'. Typo: In word 'editado'.  
        alert.contentText = "Se ha editado el equipo ${equipoNuevo.nombre} correctamente" Typo: In word 'ed:  
        alert.showAndWait()  
        cerrarVentana()  
    }.onFailure { error →  
        val alert = Alert(Alert.AlertType.ERROR)  
        alert.title = "Editar Equipo" Typo: In word 'Editar'. Typo: In word 'Equipo'.  
        alert.headerText = "Error al editar el equipo" Typo: In word 'editar'. Typo: In word 'equipo'.  
        alert.contentText = error.message  
        alert.showAndWait()  
    }
```

ViewModel

Hay un ViewModel para todo el programa

```
package dev.sergiomisas.futdam.viewmodels

import ...

@SergioMisas
class FutDamViewModel() {
    private val jugadorRepository: JugadorRepository, Tipo: In word 'jugador'.
    private val equipoRepository: EquipoRepository, Tipo: In word 'equipo'.
    private val jsonStorageService: JsonStorageService
} {
    val state = SimpleObjectProperty(FutDamState())

    @SergioMisas
    init {
        initState()
    }

    @SergioMisas
    data class FutDamState(
        val jugadores: List<Jugador> = emptyList(), Tipo: In word 'jugadores'.
        val equipos: List<Equipo> = emptyList(), Tipo: In word 'equipos'.
        val jugadorSeleccionado: JugadorFormulario = JugadorFormulario(), Tipo: In word 'jugador'. Tipo: In word 'Seleccionado'.
        val posiciones: List<String> = mutableListOf("") + Jugador.PosicionJugador.values().map { it.name } Tipo: In word 'posiciones'.
    )

    @SergioMisas
    data class JugadorFormulario( Tipo: In word 'Jugador'. Tipo: In word 'Formulario'.
        val nombre: String = "", Tipo: In word 'nombre'.
        val apellido: String = "", Tipo: In word 'apellido'.
    )
}
```

Repositorios

Uno por cada modelo y cada uno hereda de un CRUD repository para las operaciones de gestión

```
▼ repositories
  ▼ equipo
    ✚ EquipoRepository
    📦 EquipoRepositoryImpl
  ▼ jugador
    ✚ JugadorRepository
    📦 JugadorRepositoryImpl
    ✚ CrudRepository
```

Servicio de exportación a Json

Se ha utilizado gson como librería para llevar el exporte a cabo

```
package dev.sergiomisas.futdam.services.storage.json

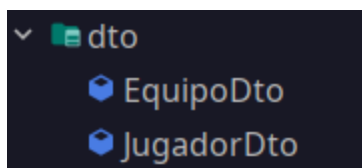
import ...

@SergioMisas
class JsonStorageService : StorageService<EquipoDto> {

    @SergioMisas
    override fun load(file: File): EquipoDto {
        val gson = GsonBuilder().setPrettyPrinting().create()
        val json = file.readText()
        val item = gson.fromJson(json, EquipoDto::class.java) Variable used only in follow
        return item
    }

    @SergioMisas
    override fun save(file: File, item: EquipoDto) {
        val gson = GsonBuilder().setPrettyPrinting().create()
        val json = gson.toJson(item)
        file.writeText(json)
    }
}
```

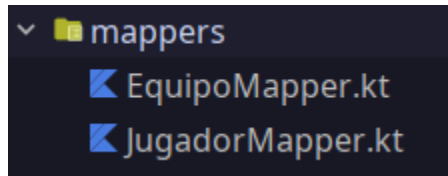
Se realizan dto de los modelos



```
package dev.sergiomisas.futdam.dto

@SergioMisas
data class EquipoDto(
    val id: Long,
    val nombre: String,
    val pais: String,
    val anyoFundacion: Int,
    val liga: String,
    val jugadores: List<JugadorDto>
)
```

Y se mapean:



```
package dev.sergiomisas.futdam.mappers

import ...

@SergioMisas
fun Jugador.toDto(): JugadorDto {
    return JugadorDto(
        id = id,
        nombre = nombre,
        apellidos = apellidos,
        edad = edad,
        nacionalidad = nacionalidad,
        dorsal = dorsal,
        apodo = apodo,
        posicion = posicion.name
    )
}

@SergioMisas
fun List<Jugador>.toDto(): List<JugadorDto> {
    return map { it.toDto() }
}
```

Configuración de la aplicación

```
class AppConfig {

    lateinit var dbUrl: String
    lateinit var dbPath: String
    lateinit var dataPath: String Property 'dataPath' could be private.
    lateinit var jsonPath: String

    val logger = KotlinLogging.logger {} Property "logger" is never used.

    @SergioMisas
    init {
        loadProperties()
        initStorage()
    }

    @SergioMisas
    private fun loadProperties() {
        val properties = Properties()
        properties.load(ClassLoader.getSystemResourceAsStream(name = "config.properties"))
        dbUrl = properties.getProperty(key = "db.url", defaultValue = "jdbc:sqlite:futdam.db") Typo: In word 'fu
        dbPath = properties.getProperty(key = "db.path", defaultValue = "futdam.db") Typo: In word 'futdam'.
        dataPath = properties.getProperty(key = "data.path", defaultValue = "data")
        jsonPath = dataPath + File.separator + properties.getProperty(key = "json.path", defaultValue = "json")
    }

    @SergioMisas
    private fun initStorage() {
        Files.createDirectories(Paths.get(jsonPath))
    }
}
```

Testing

Se ha testado con mockito

```
05      @Test
06      fun getJugadorByFormulario() { Typo: In word 'Jugador'. Typo: In word 'Formulario'.
07          `when`(jugadorRepositoryMock.findAll()).thenReturn(jugadorData)
08
09          viewModel.setJugadorSeleccionado(jugadorData[0])
10
11          assertEquals(jugadorData[0], viewModel.getJugadorByFormulario(jugadorFormulario))
12      }
13
14      @Test
15      fun borrarJugadorSeleccionado() { Typo: In word 'borrar'. Typo: In word 'Jugador'. Typo: In word 'Equipo'.
16          `when`(jugadorRepositoryMock.findAllByTeam(any())).thenReturn(jugadorData)
17          `when`(equipoRepositoryMock.findAll()).thenReturn(equipoData)
18
19          viewModel.setJugadorSeleccionado(jugadorData[0])
20
21          viewModel.borrarJugadorSeleccionado()
22
23          assertEquals(emptyList<Equipo>(), viewModel.state.value.equipos)
24      }
25
26      @Test
27      fun getNumJugadoresEquipo() { Typo: In word 'Jugadores'. Typo: In word 'Equipo'.
28          `when`(jugadorRepositoryMock.findAll()).thenReturn(jugadorData)
29
30          assertEquals( expected: 1, viewModel.getNumJugadoresEquipo(equipoData[0].id))
31      }
```

over FutDamViewModelTest x

Tests passed: 6 of 6 tests – 51 ms

Test Results 51 ms