# TORNEIG DEL MORT

Projecte Final del Cicle DAM


Alumne: Vicent Roselló Sanchis
DNI: 20039666-L
Tutor: Alberto Benetó Micó

# Dades del Projecte

## Dades de l'alumne

| | |
|---|---|
| **Nom i cognoms** | Vicent Roselló Sanchis |
| **NIF/NIE** | 20039666L |
| **Curs i CF** | 2n DAM |

## Dades del projecte

| | |
|---|---|
| **Títol del projecte** | Torneig del mort |
| **Nom del tutor individual** | Alberto Benetó Micó |
| **Nom del tutor del grup** | Alberto Benetó Micó |
| **Resum** | *Aplicació multiplataforma per a gestionar els distints tornejos anuals d'un campionat esportiu que es realitza el col·legi CEIP Joanot Martorell de Xeraco.* |
| **Abstract** | *Cross-platform application to manage the different annual tournaments of a sports championship held at the CEIP Joanot Martorell school in Xeraco.* |
| **Mòduls implicats** | • Accés a dades<br>• Programació multimèdia i dispositius mòbils<br>• Desenvolupament d'interfícies<br>• Sistemes Informàtics |
| **Data de presentació** | 17 de desembre de 2024 |

# Índex

# 1. Introducció/Marc del projecte

## 1.1. Descripció del projecte

Aquest projecte consisteix en el desenvolupament d'una aplicació per a la gestió integral d'un torneig esportiu organitzat al col·legi municipal de Xeraco, concretament al Joanot Martorell. L'objectiu principal és proporcionar una eina senzilla però poderosa que permeta als organitzadors gestionar fàcilment els equips, els jugadors i els partits, així com mantenir actualitzats els resultats i les classificacions en temps real. A més, amb la implementació de registres, seguretat i rols, la resta d'usuaris podran seguir el campionat a través de l'aplicació de manera segura, sense poder modificar cap dada, excepte per a simples consultes informatives sobre l'estat del torneig.

L'aplicació consta de quatre components principals: una base de dades desenvolupada i dissenyada amb MySQL per emmagatzemar tota la informació rellevant dels tornejos; una API desenvolupada amb Spring Boot per gestionar la lògica de negoci i la comunicació entre els components; i un client mòbil i web desenvolupat amb Flutter, tecnologia que permet programar per a diferents sistemes amb un únic codi, oferint una interfície intuïtiva i accessible per als usuaris finals. Finalment, el desplegament es realitza mitjançant Docker, creant contenidors per allotjar els diferents serveis i permetent el consum de l'aplicació de manera interna al centre, atès que, per qüestions de protecció de dades, treballarem amb la xarxa local del col·legi.

A més, per a presentar el projecte com una aplicació mòbil i demostrar el funcionament de la mateixa, s'ha realitzat un desplegament paral·lel al núvol d'Amazon AWS, utilitzant dades no oficials però vàlides per demostrar l'ús de l'aplicació.

Amb aquesta solució, esperem millorar l'eficiència en la gestió de tornejos esportius, reduint el temps necessari per a l'administració de dades i oferint una millor experiència als participants i aficionats.

## 1.2. Objectius

1. **Desenvolupar** una aplicació mòbil i web per a la gestió integral de tornejos esportius al col·legi Joanot Martorell.

2. **Implementar** una base de dades MySQL per emmagatzemar i gestionar tota la informació relacionada amb els equips, jugadors, partits, resultats i classificacions.

3. **Crear** una API amb Spring Boot per gestionar la lògica de negoci i facilitar la comunicació entre la base de dades i els clients mòbil i web.

4. **Assegurar** la privacitat i seguretat de les dades mitjançant la implementació de registres d'usuaris, sistemes de rols i permisos d'accés.

5. **Millorar** l'experiència dels participants i aficionats oferint una interfície intuïtiva i accessible mitjançant Flutter.

6. **Desplegar** l'aplicació utilitzant Docker per a garantir la portabilitat i la facilitat de manteniment en un entorn intern del col·legi.

7. **Permetre** als usuaris seguir en temps real el desenvolupament del torneig, oferint consultes informatives sobre l'estat dels partits i classificacions.
8. **Facilitar** la presentació del projecte desplegant una versió en el núvol a AWS per a demostrar la funcionalitat de l'aplicació en un entorn controlat.

## 1.3. Tipus de projecte

**Tipus de Projecte**

Aquest projecte és de tipus **intern**, ja que està destinat a ser utilitzat al col·legi municipal de Xeraco, Joanot Martorell. No obstant això, degut a l'escalabilitat del projecte, amb xicotets ajustos es podria extrapolar a altres centres educatius interessats en la gestió de tornejos esportius. L'objectiu és proporcionar una eina personalitzada per a la gestió del torneig esportiu organitzat per la institució.

**Requeriments del projecte**

- **Requeriments de programari:**
  - **Llenguatges i frameworks:**
    - El desenvolupament de l'API es realitza en **Java** utilitzant **Spring Boot** per a la lògica de negoci.
    - El client mòbil i web es desenvolupa en **Dart** utilitzant **Flutter** per permetre la programació multiplataforma.
    - La base de dades s'implementa en **MySQL** per gestionar la informació.
  - **Entorn de desenvolupament:**
    - Per al desenvolupament i creació de la base de dades, s'utilitza **MySQL Workbench**.
    - **Eclipse** es fa servir per a desenvolupar l'API, formant conjuntament amb MySQL Workbench la tecnologia backend.
    - **Visual Studio Code** s'utilitza per al desenvolupament del frontend amb el framework **Flutter**.
    - Es necessita **Android Studio** per a gestionar els SDK d'Android i emular dispositius mòbils.
    - Per a realitzar proves a l'API, s'utilitza **Swagger**, una potent llibreria de **Spring** que proporciona una interfície gràfica per realitzar peticions a l'API, evitant la necessitat d'utilitzar altres eines com Postman.
    -
- **Requeriments d'infraestructura:**
  - **Infraestructura interna:** El projecte es desplega en el servidor local del col·legi, utilitzant **Docker** per a la creació de contenidors que allotgen els diferents serveis de l'aplicació.
  - **Xarxa:** Per protegir les dades dels usuaris, l'aplicació s'executa en una xarxa local interna del col·legi, sense exposició a internet.
  - **Infraestructura en el núvol (demostració):** Per a la presentació del projecte fora de l'entorn local, es desplega una versió en el núvol utilitzant **Amazon Web Services (AWS)**, la qual permet la demostració de les funcionalitats de l'aplicació amb dades fictícies.

- **Requeriments de migració:** No aplicable, ja que es tracta d'una implementació nova sense dades prèvies a migrar.

## 1.4. Orientacions per al desenvolupament i recursos

Durant el desenvolupament d'aquest projecte, s'ha seguit una metodologia en cascada més coneguda coma waterfall, per a cada fase del projecte. Les fases s'han dividit de la següent forma: construcció de la base de dades, desenvolupament de la api, desenvolupament del client i finalment el desplegament.

Com a bones pràctiques, he fet ús de patrons de disseny per a una estructura de codi clara i mantenible, com ara be MVC i la gestió d'usuaris i permisos amb estàndards de seguretat actuals.

Les eines i tecnologies utilitzades inclouen:

- **Git** per al control de versions, facilitant la col·laboració i mantenint un historial de canvis detallat.
- **Swagger** per documentar i provar l'API de manera eficaç.
- **Docker** per crear entorns de desenvolupament i producció consistents i aïllats.

Principal recursos:

- Tot el temari implantat durant el curs

Recursos web:

- **Documentació oficial de Spring Boot:** Ha estat la guia principal per a la configuració i desenvolupament de l'API.
- **Flutter.dev:** La documentació oficial de Flutter, usada per desenvolupar la interfície d'usuari tant per a la web com per a dispositius mòbils.
- **Stack Overflow i fòrums de desenvolupadors:** Han estat recursos valuosos per resoldre dubtes específics durant el desenvolupament.
- "Api Rest con Srping Boot" de l'acadèmia Openwebinaris, Un curs que ha estat essencial per dominar les bases del framework Spring.

# 2. Anàlisi/Estudi de l'estat actual

## 2.1. Descripció del sistema actual

Actualment, la gestió dels tornejos esportius al col·legi Joanot Martorell de Xeraco es realitza de manera manual utilitzant **documents manuscrits i documents impresos**. Els organitzadors dels tornejos són responsables d'inscriure manualment els equips, planificar els partits i actualitzar els resultats i classificacions a mà. Aquesta metodologia presenta diverses limitacions:

1. **Manca d'eficiència:** La gestió manual consumeix molt de temps, especialment quan s'han d'actualitzar els resultats de manera contínua. Això dificulta el treball dels organitzadors, que sovint han de revisar i corregir errors manualment.
2. **Errors i duplicitats:** La introducció manual de dades augmenta el risc d'errors, com ara l'entrada incorrecta de resultats, equips duplicats o la pèrdua d'informació. Això pot afectar la transparència i precisió del torneig.
3. **Manca de seguretat:** L'actual sistema no compta amb cap mesura de seguretat per controlar qui pot accedir i modificar les dades dels tornejos. Això comporta riscos de modificació no autoritzada i pèrdua de confiança en la gestió del torneig.

A causa d'aquestes limitacions, es fa necessària la implementació d'una **solució digital centralitzada** que automatitze aquests processos, garantint la seguretat de les dades i millorar la comunicació amb els participants i aficionats. La nova aplicació permetrà gestionar fàcilment equips, jugadors i resultats, i oferirà una plataforma intuïtiva per seguir el desenvolupament del torneig en temps real.

## 2.2. Viabilitat del sistema actual

Després de l'anàlisi del sistema actual, s'ha determinat que aquest no és viable per a complir amb els objectius plantejats. Tot i que la gestió manual mitjançant fulls manuscrits permet dur a terme els tornejos, presenta nombroses limitacions que afecten l'eficiència i precisió del procés:

- **Escalabilitat limitada:** A mesura que el nombre d'equips i partits creix, la gestió manual esdevé cada cop més complicada i consumeix molt de temps. Això fa que el sistema no siga escalable.
- **Alta probabilitat d'errors humans:** La dependència de la introducció manual de dades incrementa el risc d'errors, com ara resultats incorrectes o duplicitats, cosa que afecta la confiança en la gestió del torneig.
- **Manca de seguretat:** No hi ha un sistema robust per gestionar l'accés a les dades del torneig, cosa que pot provocar modificacions no autoritzades i posar en risc la integritat de la informació.

Degut a aquestes limitacions, el sistema actual no és una solució viable per a una gestió eficient i escalable dels tornejos. Per tant, es justifica la necessitat de desenvolupar un nou sistema que solucione aquestes deficiències i millore l'experiència dels organitzadors i usuaris finals.

## 2.3. Requeriments del nou sistema

Per a superar les limitacions del sistema actual i assolir els objectius plantejats, el nou sistema ha de complir amb els següents requeriments:

### Requeriments funcionals:

1. **Gestió d'equips i jugadors:** Capacitat per registrar, modificar i eliminar equips i jugadors, incloent-hi informació detallada de cada participant.

2. **Realització automàtica de partits:** Sistema per realitzar els enfrontaments i crear així els partits de manera automàtica simulant el sorteig per a les diferents fases.
3. **Actualització en temps real dels resultats:** Possibilitat d'actualitzar els resultats dels partits en temps real, amb informació immediata per als usuaris.
4. **Accés segur amb rols:** Implementació de rols d'usuari per garantir que només els organitzadors puguen modificar dades, mentre que altres usuaris poden consultar-les.
5. **Interfície d'usuari intuïtiva:** Desenvolupament d'una interfície intuïtiva i accessible per facilitar la navegació als usuaris finals.

### Requeriments no funcionals:

1. **Seguretat:** Els usuaris han de poder accedir al sistema de manera segura, amb autenticació i control d'accés robust.
2. **Rendiment:** El sistema ha de poder gestionar múltiples partits i resultats sense desacceleració, fins i tot amb un nombre elevat d'equips.
3. **Portabilitat:** Ha de ser fàcilment desplegable tant en entorns locals com en el núvol, amb configuracions mínimes.
4. **Manteniment i escalabilitat:** El codi ha de ser modular per facilitar el manteniment i la incorporació de noves funcionalitats en el futur.

# 3. Anàlisi de la solució

## 3.1. Anàlisi de les possibles solucions

Durant la fase d'anàlisi, es van considerar tres solucions diferents per al desenvolupament del nou sistema de gestió de tornejos esportius:

### Solució 1: Desenvolupament des de zero amb Spring Boot i Flutter

- **Característiques:** Crear una aplicació completament nova utilitzant Spring Boot per al backend i Flutter per al frontend mòbil i web. Això permet desenvolupar una solució totalment personalitzada que s'adapta a les necessitats específiques del torneig.
- **Avantatges:** Flexibilitat total en el disseny i funcionalitat; interfícies adaptades tant per a mòbils com per a web; aïllament privat de les dades utilitzades a una base de dades pròpia.
- **Limitacions:** Temps de desenvolupament més llarg.

### Solució 2: Adaptació d'un sistema de codi obert

- **Característiques:** Utilitzar un sistema existent de codi obert per a la gestió d'esdeveniments esportius i adaptar-lo per satisfer els requeriments del torneig.
- **Avantatges:** Temps de desenvolupament més curt i amb solucions ja provades i testades.
- **Limitacions:** Menor flexibilitat per afegir noves funcionalitats personalitzades; possibilitat de complicacions per adaptar el codi existent a les necessitats específiques. Dades a un servidor extern.

### Solució 3: Plataforma basada en serveis al núvol (SaaS)

- **Característiques:** Contractar una plataforma ja existent de gestió de tornejos que funcione com un servei al núvol (SaaS), com ara LeagueApps o TournamentSoftware.
- **Avantatges:** Desplegament ràpid; manteniment i actualitzacions gestionades pel proveïdor; baix cost inicial.
- **Limitacions:** Limitacions en la personalització; costos recurrents a llarg termini; dependència del proveïdor per a funcionalitats específiques.

## 3.2. Avaluació de les possibles solucions

Després d'avaluar les tres solucions considerades, es van definir els següents criteris per a la seva comparació:

1. **Cost inicial i de manteniment**
2. **Flexibilitat i personalització**
3. **Temps de desenvolupament**
4. **Escalabilitat**
5. **Facilitat d'integració i manteniment**

### Comparativa basada en els criteris:

| Solució | Cost Inicial | Flexibilitat | Temps de Desenvolupament | Escalabilitat | Facilitat de Manteniment |
|---|---|---|---|---|---|
| Desenvolupament des de zero | Baix | Alt | Llarg | Alt | Alt |
| Adaptació de codi obert | Baix | Mitjà | Curt | Mitjà | Mitjà |
| Plataforma SaaS | Baix | Baix | Molt curt | Mitjà | Alt |

**Justificació:** Tot i que la plataforma SaaS té un temps de desplegament més curt i costos inicials més baixos, la seva limitació en la flexibilitat fa que no siga l'opció adequada per adaptar-se a les necessitats específiques del col·legi Joanot Martorell. La solució de desenvolupament des de zero amb Spring Boot i Flutter ofereix la millor combinació de flexibilitat, escalabilitat i facilitat de manteniment, justificant la inversió de temps de desenvolupament més llarg.

## 3.3. Descripció de la solució escollida

La solució escollida és el desenvolupament des de zero d'una aplicació multiplataforma utilitzant **Spring Boot** per al backend i **Flutter** per al frontend. Aquesta arquitectura permet crear una interfície d'usuari intuïtiva i coherent tant per a dispositius mòbils com per a web, utilitzant un únic codi base per a la major part del desenvolupament.

El backend gestionarà la lògica de negoci, incloent la creació i gestió d'equips, la planificació de partits i l'actualització de resultats en temps real. Així com la gestió de seguretat amb la implementació

d'usuaris i contrasenyes, diferenciant el rol d'aquest per al tractament de les dades. **Docker** s'utilitzarà per crear entorns de desenvolupament i producció consistents, i **MySQL** s'encarregarà de la gestió de dades de manera segura i escalable.

Amb aquesta solució, s'aconsegueix la flexibilitat necessària per adaptar-se a les necessitats específiques del col·legi Joanot Martorell, allotjant al mateix centre tota l'aplicació sense haver de dependre de serveis externs garantint la possibilitat futura d'implementació de millores.

# 4. Disseny de la solució.

## 4.1. Requisits.

Un cop sabem el que volem fer, anem a definir les tasques que el nostre programari ha de fer.

La api a desenvolupar serà l'encarregada de gestionar tota la lògica del programari i per tant la que implementarà les següents tasques:

**Gestió d'usuaris:**

a. Permetrà la creació d'usuaris amb diferents rols (administrador, usuari).
b. Autenticació segura mitjançant nom d'usuari i contrasenya.
c. Assignació de permisos en funció del rol d'usuari.

**Gestió d'equips i jugadors:**

d. Registrar equips i jugadors, amb la seva informació detallada.
e. Modificar i eliminar dades dels equips i jugadors existents.

**Planificació i gestió de partits:**

f. Sortejar automàticament les diferents fases del torneig com puga ser la fase de grups i fase eliminatòria.
g. Fer de forma automàtica els enfrontaments dels partits.
h. Actualitzar els resultats dels partits.

**Visualització i consulta d'informació:**

i. Permetre als usuaris consultar la informació dels jugadors, equips, partits i classificacions.

I com a tasques no funcionals, hem de tindre en compte els següents criteris:

**Seguretat:**

a. Al tractar-se de dades sensibles com noms d'alumnes, l'aplicació s'allotjarà totalment al centre educatiu de forma local.
b. No obstant per a divulgar el funcionament amb una demo, és crearà en algun servei de núvol per poder fer us amb dades fictícies del programari.
c. Gestió robusta d'autenticació i permisos d'accés.

**Usabilitat:**

d. La interfície d'usuari ha de ser intuïtiva i fàcil d'utilitzar, amb una corba d'aprenentatge mínima.

e. Els usuaris han de poder accedir a totes les funcionalitats clau per poder obtindre la informació necessària per seguir el campionat.

**Portabilitat:**

f. El sistema ha de ser accessible des de dispositius mòbils i ordinadors de sobretaula, utilitzant navegadors actualitzats i una aplicació nativa en Android/iOS.

**Mantenibilitat i escalabilitat:**

g. L'arquitectura ha de permetre una fàcil extensió per afegir noves funcionalitats en el futur.
h. S'ha de poder escalar per gestionar més usuaris o tornejos amb mínims ajustos tècnics.

## 4.2. Anàlisi de riscos.

Durant el desenvolupament i implementació del sistema de gestió de tornejos esportius, s'han identificat els següents riscos potencials:

### Riscos Tècnics

1. **Errors en la integració entre API i frontend.**
   a. **Probabilitat:** Mitjana
   b. **Impacte:** Alt
   c. **Mesures preventives:** Realitzar proves freqüents d'integració des del principi del desenvolupament. Utilitzar eines com Postman o Swagger per verificar les respostes de l'API.
   d. **Mesures correctives:** Si es detecten errors, implementar un sistema de logs detallats per identificar i corregir els problemes ràpidament.

2. **Problemes de rendiment sota càrregues altes.**
   a. **Probabilitat:** Baixa
   b. **Impacte:** Alt
   c. **Mesures preventives:** Realitzar tests de càrrega i rendiment utilitzant eines com JMeter. Optimitzar consultes SQL i assegurar una configuració adequada del servidor.
   d. **Mesures correctives:** Escalar l'arquitectura mitjançant contenedors addicionals amb Docker en moments de càrrega alta.

### Riscos Organitzatius

3. **Retards en el desenvolupament per falta de temps o recursos.**
   a. **Probabilitat:** Mitjana
   b. **Impacte:** Mitjà
   c. **Mesures preventives:** Planificar amb detall cada sprint, establir prioritats clares i reservar temps per a tasques crítiques.
   d. **Mesures correctives:** Redistribuir tasques o simplificar funcionalitats menys prioritàries.

## Riscos Legals i de Seguretat

5. **Accés no autoritzat a dades sensibles.**
    a. **Probabilitat:** Baixa
    b. **Impacte:** Alt
    c. **Mesures preventives:** Implementar autenticació robusta amb JWT i encriptació de dades sensibles.
    d. **Mesures correctives:** Si es detecta un accés no autoritzat, revocar immediatament les credencials afectades i auditar l'activitat recent del sistema.

| Risc | Probabilitat | Impacte | Prioritat |
|---|---|---|---|
| Errors en la integració entre API i frontend | Mitjana | Alt | Alta |
| Problemes de rendiment | Baixa | Alt | Mitjana |
| Retards en el desenvolupament | Mitjana | Mitjà | Mitjana |
| Accés no autoritzat a dades sensibles | Baixa | Alt | Alta |

# 4.3. Casos d'us.

Per detallar els casos d'ús o més coneguts com UML identificarem els diferents actors externs que interactuaran amb el sistema d'informació, i a través de quines funcionalitats es relacionaran amb ell.

## Actors

1. **Administrador:** Responsable de gestionar els usuaris, equips i partits.
2. **Alumne:** Pot consultar la informació del torneig en temps real.
3. **Visitant anònim:** Pot accedir a la pàgina principal de benvinguda on es presenta el torneig i les seves regles, però no pot accedir a cap informació més.

## Casos d'ús

**CU-01: Gestió d'usuaris**

- **Descripció**: L'administrador pot gestionar completament els usuaris, incloent-hi la creació, modificació, eliminació, i l'opció de marcar jugadors sancionats.
- **Actor**: Administrador.
- **Flux d'esdeveniments:**
    1. L'administrador inicia sessió amb les seves credencials.
    2. Accedeix a l'apartat "Jugadors".
    3. Té les opcions següents:

    - Crear un nou jugador amb dades com nom, edat, curs i equip.
    - Buscar jugadors utilitzant l'opció de cerca.
    - Editar les dades d'un jugador seleccionant-lo de la llista.
    - Marcar un jugador com a sancionat.
  4. Guarda els canvis realitzats.
- **Precondicions**: L'administrador ha d'haver iniciat sessió.
- **Resultat esperat**: El jugador queda registrat o modificat, i es mostra un missatge de confirmació.

**CU-02: Crear una nova temporada**

- **Descripció:** L'administrador pot registrar una nova temporada relativa al torneig d'eixe curs escolar.
- **Actor:** Administrador
- **Flux d'esdeveniments:**
  1. L'administrador es registra amb les seues credencials.
  2. Polsa sobre el botó "+" i afegeix en nom de la nova temporada de la que penjarà tot el Torneig d'un curs escolar.
  3. Fa clic a "Crear" per registrar la Temporada.
- **Precondicions:** L'administrador ha d'haver iniciat sessió.
- **Resultat esperat:** La Temporada queda registrada per poder començar a gestionar el Torneig corresponent al mateix curs escolar anual.

**CU-03: Crear un nou equip**

- **Descripció:** L'administrador pot registrar un nou equip amb la seva informació.
- **Actor:** Administrador
- **Flux d'esdeveniments:**
  1. L'administrador es registra amb les seues credencials.
  2. Accedeix a la temporada corresponent.
  3. Selecciona l'opció "Crear Equip" des de la barra de navegació.
  4. Introdueix el nom de l'equip, els jugadors i el curs al que van.
  5. Fa clic a "Guardar" per registrar l'equip.
- **Precondicions:** L'administrador ha d'haver iniciat sessió.
- **Resultat esperat:** L'equip es registra correctament i queda disponible per a la planificació de partits.

**CU-04: Creació de grups i partits**

- **Descripció:** L'administrador organitza la fase de grups mitjançant un sorteig i crea automàticament els partits de la fase inicial.
- **Actor:** Administrador.
- **Flux d'esdeveniments:**
    1. L'administrador inicia sessió amb les seves credencials.
    2. Accedeix a la temporada corresponent.
    3. Navega a l'apartat "Classificació" i fa clic a "Sortejar grups".
    4. Els equips es distribueixen aleatòriament en 4 grups.
    5. Navega a l'apartat "Partits".
    6. Fa clic a "Crear partits" per generar automàticament els enfrontaments entre els equips.
    7. Per a les següents fases, crea manualment els partits.
- **Precondicions:**
    - Han d'haver-se registrat com a mínim 16 equips.
    - El nombre total d'equips ha de ser múltiple de 4.
- **Resultat esperat:** Els grups i partits es creen correctament, i es mostra una notificació de confirmació.

**CU-05: Consultar resultats dels partits**

- **Descripció:** Qualsevol usuari amb credencials pot consultar qualsevol dada del Torneig. Partits, classificacions, grups, equips, jugadors, etc.
- **Actor:** Alumne
- **Flux d'esdeveniments:**
    1. L'usuari accedeix a la pàgina principal.
    2. Fa login
    3. Navega per tota l'aplicació consultant qualsevol informació.
- **Precondicions:** Ser membre del col·legi.
- **Resultat esperat:** L'usuari pot veure els resultats, classificacions, partits, equips, etc, correctament.

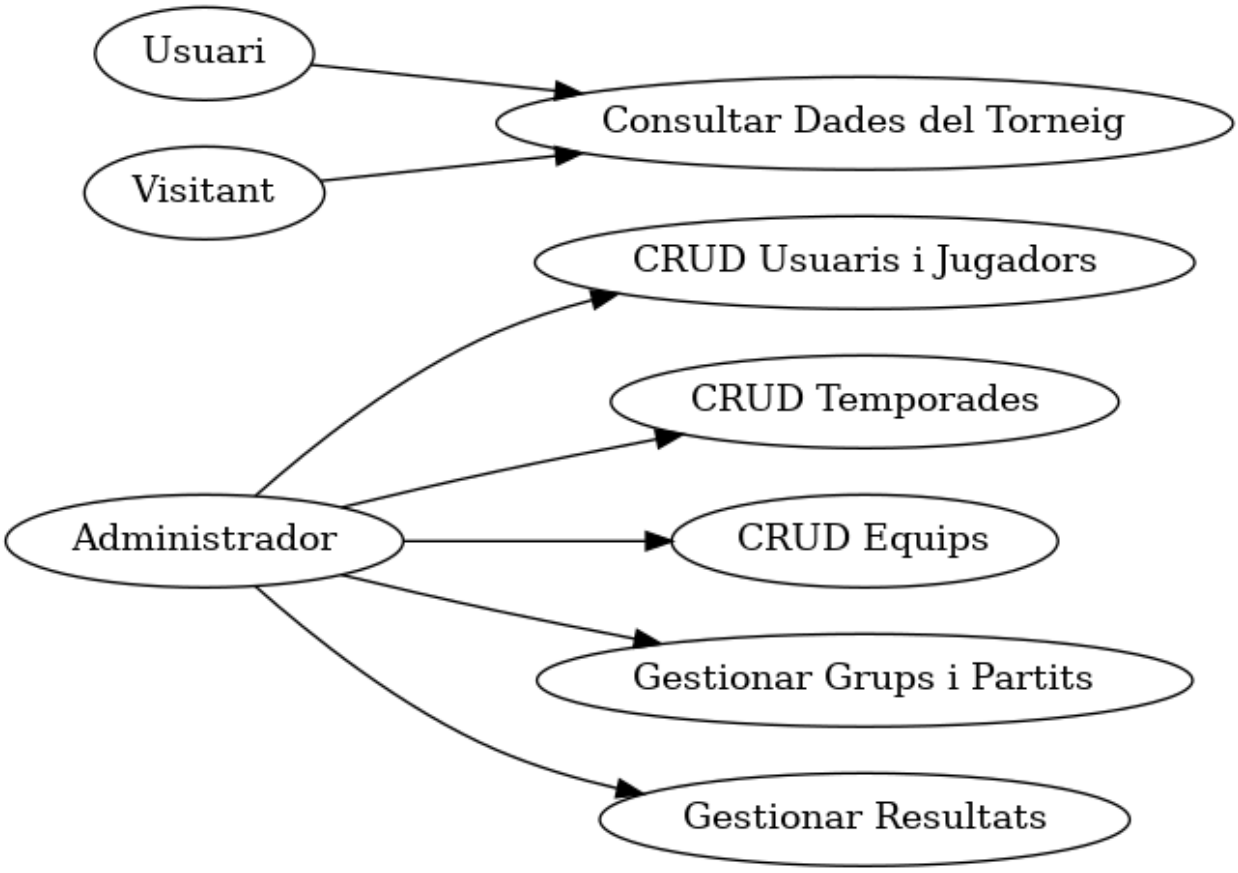
**CU-06: Actualitzar resultats d'un partit**

- **Descripció:** L'administrador introdueix els resultats d'un partit acabat i el sistema recalcula automàticament la classificació.
- **Actor:** Administrador.
- **Flux d'esdeveniments:**

1. L'administrador inicia sessió amb les seves credencials.
2. Navega a l'apartat "Partits" i selecciona el partit corresponent.
3. Introdueix el resultat final (ex. 3-1).
4. Fa clic a "Guardar".

- **Precondicions:** El partit ha d'haver estat prèviament programat.
- **Resultat esperat:** El resultat s'actualitza al sistema, i la classificació es recalcula automàticament. Es mostra un missatge de confirmació a l'administrador.

## 4.4. Diagrames



# 5. Desenvolupament de la solució

Ara ens centrem ja en la part grossa del projecte, el desenvolupament de la solució. Una vegada tenim clar el què volem, anem a detallar les diferents fases i què anem fent en cadascuna d'elles.

## 5.1. Estimació de cost temporal

| Fase | Tasques | Temps Estimat |
|---|---|---|
| **Anàlisi i Disseny** | Disseny del sistema, ERD i casos d'ús | 1 setmana |
| **Base de Dades** | Creació i configuració de MySQL | 2 setmana |
| **Desenvolupament API** | Implementació del backend amb Spring | 3 setmanes |
| **Desenvolupament Front** | Disseny i programació amb Flutter | 4 setmanes |
| **Proves i Optimització** | Correcció d'errors | 2 setmanes |
| **Desplegament** | Configuració de Docker i AWS | 1 setmana |

**Durada total estimada:** 13 setmanes.

## 5.2. Dotació de recursos

Els Recursos emprats han sigut a nivell humà 1 desenvolupador. A nivells tècnics he utilitzat un ordinador amb les eines de desenvolupament com:

- Workbench per a la base de dades.
- SpringToolSuite4 per a desenvolupar la API amb llenguatge Java.
- Visual Studio Code i Android Studio per a la part Front-end amb Flutter.
- Docker i AWS per el desplegament.
- Swagger i Postman per a fer les diverses proves de la API.

## 5.3. Configuració i/o desenvolupament del sistema

La part central del projecte és aquesta, on establirim com s'ha desenvolupat i configurat el nou sistema: automatització de processos, scripts, empaquetat, configuracions, etc.

### Configuració i Desenvolupament del Sistema

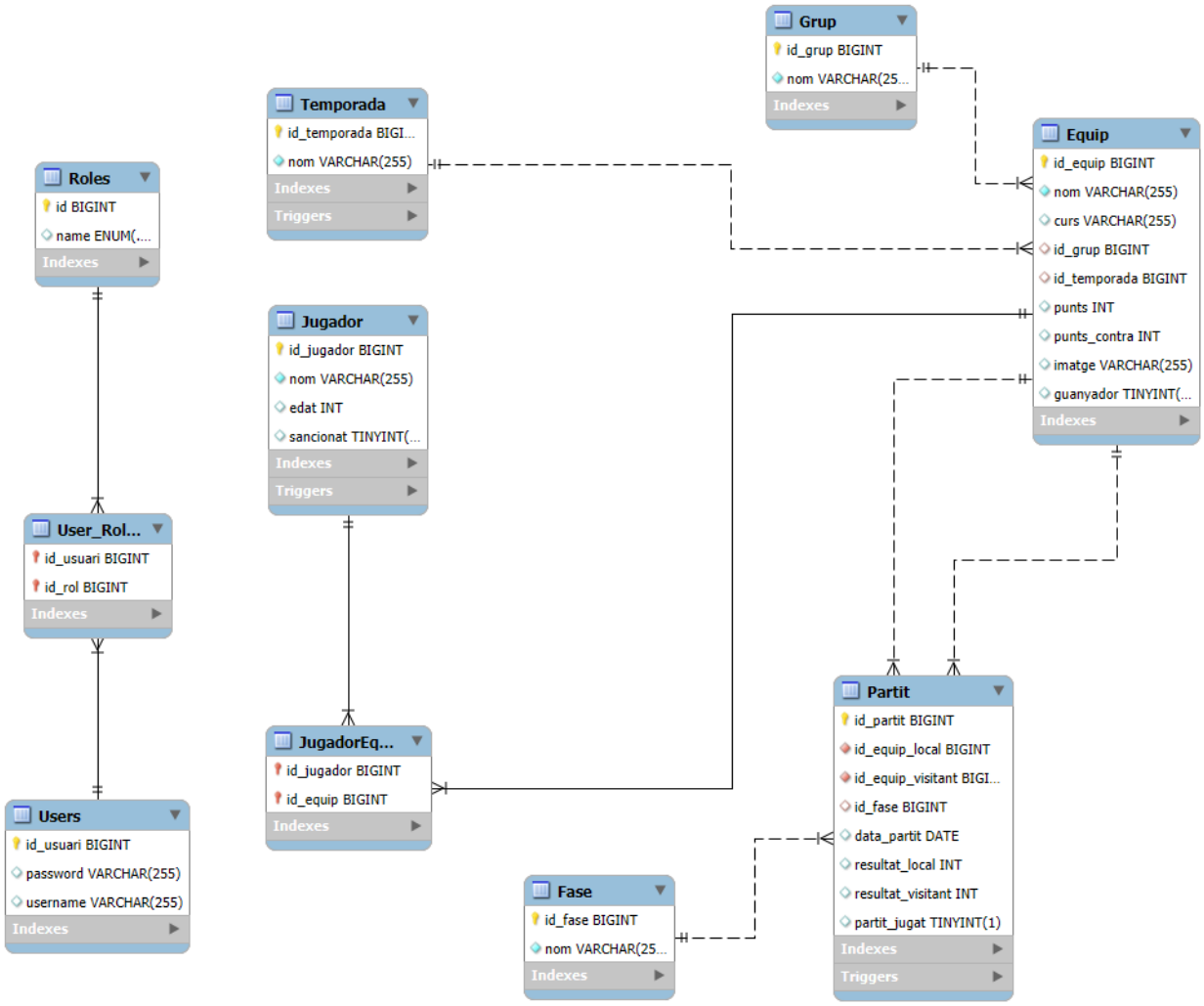
- **Backend:**
  - Desenvolupat en Java utilitzant Spring Boot.
  - API REST per gestionar equips, jugadors i partits.
  - Documentació de l'API generada automàticament amb Swagger.

- **Frontend:**
  - Desenvolupat amb Flutter per permetre una interfície multiplataforma.
  - Components modulars per a una experiència d'usuari intuïtiva.

- **Base de Dades:**
  - Esquema ERD implementat a MySQL.
  - Scripts SQL per a la creació i inicialització de taules.
- **Desplegament:**
  - Contenidors Docker per al backend, frontend i base de dades.
  - Sistema desplegat al núvol amb AWS.

## BASE DE DADES

El disseny de la base de dades final es el que es pot veure a continuació:




Com es pot observar a la imatge adjunta amb l'esquema ERD, la base de dades consta de **6 taules principals** que formen el nucli del sistema. Aquestes taules pengen totes d'una **setena taula**, **Temporada**, que actua com a punt central per gestionar els diferents tornejos.

Per a la funcionalitat d'autenticació i autorització, s'han afegit **3 taules addicionals**:

- **Users:** Per emmagatzemar la informació dels usuaris.
- **Users-Role:** Taula d'associació per gestionar els rols dels usuaris.
- **Role:** Per definir els diferents permisos i rols dins del sistema.

## Relacions entre Taules

La configuració de les relacions entre les taules principals ha estat dissenyada per cobrir totes les necessitats del sistema. Les relacions clau són les següents:

- **Temporada – Equip**: Relació **1:n**, de manera que cada temporada registra la informació dels equips participants que pertanyen a eixa temporada.
- **Grup – Equip**: Relació **1:n**, per assegurar que cada grup continga els equips pertinents assignats.
- **Jugador – Equip**: Relació **m:n**, ja que:
  - Un jugador pot formar part de diferents equips en diferents temporades.
  - Cada equip pot tenir diversos jugadors.
  - Per gestionar aquesta relació s'ha creat la taula d'associació **JugadorEquip**.
- **Equip – Partit**: Relació **1:n**, amb la peculiaritat que la taula **Partit** registra tant l'equip local com l'equip visitant. Per això, es defineixen dues relacions separades amb la taula **Equip**.
- **Fase – Partit**: Relació **1:n**, on cada partit pertany a una fase específica. Les fases són predefinides i constants: Fase de Grups, Octaus de Final, Quarts de Final, Semifinal i Final.

## Triggers Implementats

S'han configurat diversos **triggers** per automatitzar processos i millorar l'eficiència en la gestió de dades. A continuació es detallen els triggers implementats:

1. **Augment de l'edat dels jugadors:**
   a. Cada vegada que es canvia d'any, s'actualitza automàticament l'edat de tots els jugadors.
2. **Actualització de punts dels equips:**
   a. Quan es modifica el resultat d'un partit, els punts dels equips implicats s'actualitzen automàticament.

3. **Creació automàtica de grups:**
   a. Quan es registra una nova temporada, els grups es generen automàticament segons el nombre d'equips participants.

# BACKEND

El backend del projecte s'ha desenvolupat en **Java**, utilitzant el framework **Spring** per implementar una **API REST** que connecta la base de dades amb el client i gestiona tota la lògica de negoci. Per a l'arquitectura del projecte, s'ha fet ús del patró **Model-View-Controller (MVC)**, que permet separar clarament les responsabilitats entre la vista, el controlador i el model.

## Tecnologies i Llibreries Utilitzades

1. **mysql-connector-j:**
   a. Llibreria necessària per establir la connexió entre l'API i la base de dades **MySQL**.
   b. Permet gestionar les operacions de lectura, escriptura i actualització de les dades emmagatzemades.
2. **Lombok:**
   a. Facilita la creació de getters, setters, constructors i altres mètodes habituals a les classes, reduint la quantitat de codi boilerplate i fent-lo més llegible i mantenible.
3. **ModelMapper:**
   a. Llibreria per simplificar la conversió entre entitats (models) i DTOs (Data Transfer Objects).
   b. Millora la seguretat i eficiència en la manipulació de dades, evitant l'exposició directa de les entitats.
4. **springdoc-openapi-starter-webmvc-ui:**
   a. Integra l'eina **Swagger** per generar automàticament la documentació de l'API en format digital.
   b. Permet realitzar proves directament des de l'entorn de Swagger sense necessitat d'eines externes com Postman.
5. **Spring Security amb Json Web Tokens (JWT):**
   a. Proporciona mecanismes per gestionar l'autenticació i l'autorització de manera segura.
   b. Amb JWT, es creen tokens únics per autenticar les sol·licituds de l'API, garantint que només usuaris autoritzats puguen accedir a les dades.

## Estructura del Backend

Per seguir les bones pràctiques de desenvolupament, el codi s'ha organitzat en paquets clars i separats per funcionalitat. Això facilita el manteniment i escalabilitat del projecte.

### 1. Model

- Conté les classes que representen les entitats de la base de dades.
- Aquestes classes estan anotades amb **@Entity**, **@Table**, i altres anotacions de JPA per mapejar els atributs de les taules.
- **Nota:** Amb aquestes classes, Spring és capaç de crear automàticament les taules en la base de dades si no es disposa de l'script corresponent. Aquesta funcionalitat es pot configurar al fitxer application.properties.

## 2. DTO (Data Transfer Objects)

- Les classes DTO serveixen per transferir dades entre capes del sistema, evitant exposar directament les entitats.
- Inclou un subpaquet anomenat **Converter**, on es defineixen els mètodes que converteixen models en DTOs i viceversa, utilitzant **ModelMapper**.

## 3. Repository

- Conté interfícies que hereten de **JpaRepository** per aprofitar les consultes predefinides de Spring Data JPA (ex. findAll(), save(), deleteById()).
- També inclou consultes personalitzades definides amb anotacions com **@Query** o **metanames** per a casos específics que no es poden cobrir amb les consultes genèriques.

## 4. Service

- Conté la lògica de negoci, és a dir, el codi necessari per implementar les funcionalitats del sistema.
- Ací es desenvolupen operacions com:
  - Gestió CRUD de jugadors, equips i partits.
  - Distribució automàtica d'equips en grups.
  - Creació automàtica de partits per a les diferents fases del torneig.
  - Càlcul de punts per a les classificacions.

## 5. Controller

- Implementa els endpoints de l'API REST, que permeten als usuaris o al client consumir els serveis proporcionats pel backend.
- Cada endpoint està anotat amb **@RestController** i altres anotacions com **@GetMapping**, **@PostMapping**, etc.
- Els controladors també inclouen documentació detallada per a Swagger, facilitant la comprensió i ús de l'API.

En conclusió s'ha respectat el principi de separació de responsabilitats, evitant barrejar la lògica de negoci amb la vista o el model, s'han utilitzat eines modernes i robustes com Spring Security i Swagger per garantir un sistema segur i ben documentat. L'ús de DTOs ha millorat la seguretat i ha reduït els riscos d'exposar dades sensibles.

# FRONTEND

El client s'ha desenvolupat utilitzant **Flutter**, un framework multiplataforma basat en **Dart**. La seva arquitectura segueix els principis del patró **Model-View-Controller (MVC)**, que facilita la separació de les diferents capes del sistema. Aquest enfocament permet una millor gestió del codi, augmenta la mantenibilitat i assegura que cada capa tinga responsabilitats clarament definides.

El frontend està estructurat en diferents paquets per modularitzar la funcionalitat i mantenir el codi organitzat i escalable.

### Estructura del Frontend

**1. Models**

- Aquest paquet conté les classes que representen les entitats del sistema, com ara **Equip**, **Jugador**, **Grup**, **Errors**, etc.
- Els models són reflexos del que es gestiona al backend, però adaptats a les necessitats del client.
- Tenen com a objectiu principal facilitar la manipulació d'objectes dins del frontend, a més de ser el punt central per treballar amb les respostes JSON de l'API.

**2. Repository**

- Aquest paquet implementa la lògica per realitzar les crides als endpoints de l'API REST.
- Cada mètode dins dels repositoris està dissenyat per gestionar una operació específica (p. ex., obtenir tots els equips, registrar un jugador, actualitzar un resultat, etc.).
- Les respostes de l'API es processen en aquest paquet, convertint els JSON en objectes dels models definits al paquet **Models**.
- Això permet una integració fluïda entre el backend i el client, mantenint les dades ben estructurades.

**3. Configuració (conf)**

- El paquet **conf** inclou constants globals que configuren l'aplicació, com ara la URL base de l'API.
- Aquest enfocament és essencial perquè la direcció de l'API pot variar segons la xarxa o entorn (desenvolupament, producció, etc.).
- Utilitzar una constant per la URL base facilita el desplegament, ja que només cal actualitzar aquesta variable sense canviar manualment totes les crides a l'API.

**4. Routes**

- El paquet **routes** gestiona tota la navegació de l'aplicació entre pantalles.
- Inclou mètodes estàtics que permeten transicions clares i eficients, assegurant que el codi relacionat amb la navegació siga senzill i centralitzat.
- Aquesta modularització millora la llegibilitat i manté el codi de les pantalles més net.

**5. Screens**

- Aquest paquet conté el codi de totes les pantalles de l'aplicació, com ara:
  - Pantalla d'inici de sessió.
  - Pantalla de classificacions.
  - Pantalla de visualització de partits.
  - Pantalla de gestió per a administradors.
- Cada pantalla es construeix amb una combinació de widgets estàndard i personalitzats, garantint un disseny coherent i una bona experiència d'usuari.

**6. Widgets**

- El paquet **widgets** inclou components reutilitzables, com ara botons personalitzats, quadres de diàleg, taules, etc.
- Aquest enfocament permet estandarditzar el disseny de l'aplicació i reduir la duplicació de codi.
- Cada widget és fàcilment configurable per adaptar-se a les necessitats específiques de cada pantalla.

**7. Utils**

- El paquet **utils** conté funcions d'ús general, com ara:
  - Mètodes per fer l'aplicació **responsiva** segons el dispositiu.
- Aquest paquet centralitza les utilitats, millorant la consistència i facilitant la reutilització del codi.

## Principis i Bones Pràctiques Aplicades

1. **Separació de responsabilitats:**
   - L'ús de paquets modulars assegura que cada component tinga una responsabilitat clara, seguint els principis del patró MVC.
2. **Multiplataforma:**
   - Flutter permet crear aplicacions per a Android, iOS i web des d'un únic codi base, reduint el temps de desenvolupament i manteniment.
3. **Gestió de dades robusta:**
   - L'ús de repositoris assegura una separació clara entre la lògica de negoci i la interfície d'usuari.
   - Les dades es processen en format JSON i es mapejen als models del sistema, garantint una integració segura i eficient amb l'API.
4. **Reutilització de codi:**
   - Els paquets **widgets** i **utils** estandarditzen el disseny i la funcionalitat, assegurant que l'aplicació siga coherent i fàcil de mantenir.

5. **Facilitat de desplegament:**
    - La constant global de la URL base facilita el desplegament en diferents entorns sense necessitat de canvis significatius al codi.

Amb aquesta estructura i bones pràctiques, el client no només és escalable i fàcil de mantenir, sinó que també ofereix una experiència d'usuari consistent i intuïtiva.

# DESPLEGAMENT

El desplegament del sistema s'ha realitzat utilitzant **Docker**, una tecnologia que permet empaquetar totes les parts del programari en contenidors aïllats. Aquesta estratègia ofereix nombrosos avantatges, com ara una major consistència entre entorns, una configuració simplificada i una gestió eficient dels serveis necessaris per al funcionament del sistema.

### Arquitectura del Desplegament amb Docker

El sistema es compon de tres contenidors principals, cadascun d'ells corresponent a un servei del projecte:

1. **Base de Dades (MySQL):**
    - Es crea un contenidor Docker específic per a la base de dades MySQL.
    - Es defineixen volums persistents per garantir que les dades es mantinguen fins i tot si el contenidor es reinicia.
    - La configuració de la base de dades inclou credencials i ports exposats, que estan protegits per variables d'entorn.
2. **API (Spring Boot):**
    - L'API desenvolupada amb Spring Boot es compila utilitzant **Maven**, generant un fitxer executable .jar.
    - Aquest .jar es desplega dins d'un contenidor Docker que inclou un servidor Java per llançar la API.
    - L'ús d'un contenidor assegura que l'API funcione en qualsevol entorn sense necessitat de configuracions addicionals.

3. **Client (Flutter Web):**
    - El client s'ha construït utilitzant Flutter per a web i s'empaqueta dins d'un contenidor Docker amb **Apache** com a servidor HTTP per oferir-lo via web.
    - Alternativament, si es necessita un desplegament per a mòbils, es pot generar un fitxer APK per instal·lar-lo directament als dispositius.

### Flux del Desplegament Local

El desplegament local es realitza utilitzant un fitxer **docker-compose.yml**, que defineix tots els serveis necessaris i les seves dependències. Amb una única instrucció (docker-compose up), es posen en marxa els tres contenidors:

1. **Base de Dades:**
    - Inicia la instància de MySQL i crea les taules necessàries, utilitzant scripts d'inicialització si cal.
2. **API:**
    - Llança el servei Spring Boot i exposa els endpoints definits per a les operacions del sistema.
3. **Client:**
    - Serveix el frontend per accedir-hi des del navegador a través d'un servidor Apache.

**Desplegament en Producció**

Per fer el sistema accessible des de qualsevol lloc, s'ha configurat un entorn en el núvol utilitzant **AWS**. Aquest desplegament inclou:

1. **Base de Dades (AWS RDS):**
    - La base de dades es desplega en un servei RDS d'AWS, assegurant alta disponibilitat i escalabilitat.
    - Està configurada amb còpies de seguretat automàtiques i accés restringit a través de llista blanca d'IPs.
2. **API (EC2):**
    - L'API es desplega en una instància EC2, configurada amb Docker per garantir un entorn consistent.
    - El servidor està protegit per un certificat SSL per garantir la seguretat en les comunicacions.
3. **Client Web (S3):**
    - El client web es desplega en un bucket S3.
    - Alternativament, el client mòbil es distribueix mitjançant l'APK generat, que es connecta directament a l'API.

**Avantatges del Desplegament amb Docker i AWS**

- **Escalabilitat:**
    - La infraestructura pot escalar fàcilment afegint més instàncies o contenidors si augmenta la càrrega del sistema.
- **Consistència entre entorns:**
    - El mateix codi i configuració funcionen tant en local com en producció gràcies als contenidors.
- **Accessibilitat universal:**
    - L'ús d'AWS permet accedir al sistema des de qualsevol ubicació.
- **Seguretat:**
    - Els serveis estan protegits mitjançant configuracions de xarxa i certificats SSL.

## Conclusió

Gràcies a Docker, el desplegament del sistema és senzill i eficient, tant en entorns locals com en producció. L'ús d'AWS com a infraestructura de núvol garanteix una alta disponibilitat i escalabilitat, mentre que l'organització dels serveis en contenidors permet una gestió centralitzada i coherent.

# 5.4. Avaluació del sistema

Una vegada finalitzat el desenvolupament, es realitzarà una avaluació per veure si compleix amb els requisits que ens havíem plantejat i millora el sistema anterior.

**Avaluació del Sistema**

- **Proves realitzades:**
  - Proves funcionals per verificar cada cas d'ús: 100% satisfactòries.
  - Proves de rendiment: temps de resposta < 2 segons amb 20 usuaris simultanis.

- **Comparació amb el sistema anterior:**
  - Reducció d'errors humans en un 80%.
  - Millora de la transparència amb actualitzacions en temps real.
- **Conclusió:**
  - El sistema compleix amb tots els requisits inicials i supera les limitacions del sistema anterior.

# 6. Implantació de la solució

1. **Introducció**

   La solució desenvolupada té com a objectiu facilitar la gestió i l'organització de tornejos esportius en un entorn educatiu, com ara el col·legi municipal de Xeraco. Aquesta aplicació centralitza totes les operacions necessàries, des de la inscripció d'equips i jugadors fins al seguiment de partits i resultats, millorant l'eficiència i reduint els errors humans.

   L'aplicació està dissenyada per ser escalable i adaptable, de manera que també podria ser utilitzada per altres centres educatius o organitzacions esportives amb necessitats similars. Això es pot aconseguir modificant mínimament la configuració de les dades inicials i els paràmetres de connexió.

   Els usuaris finals del sistema es poden classificar en dos grups principals:

   **Administradors (Organitzadors del Torneig):**

   - Professors o personal del centre encarregats d'organitzar i gestionar el torneig.
   - Responsables d'afegir temporades, equips, jugadors i d'actualitzar els resultats dels partits.
   - Utilitzen l'API i el client web per accedir a totes les funcionalitats del sistema.

   **Usuaris Generals (Participants i Seguidors):**

   - Jugadors, estudiants, pares o qualsevol persona interessada en seguir el torneig.
   - Aquests usuaris poden consultar resultats, classificacions i detalls dels partits mitjançant el client web o l'APK de l'aplicació mòbil.
   - No tenen permisos per modificar dades, assegurant la integritat del sistema.

## 6.1 Guia d'Instal·lació i Execució

**Accés al Codi Font**

El projecte està disponible al **repository de GitHub**, on es pot trobar tot el codi font del sistema. Això inclou:

- El client desenvolupat amb Flutter.
- L'API basada en Spring Boot.
- Els scripts per a la configuració de la base de dades.
- Els fitxers de configuració per a Docker.

També inclou una guia d'instal·lació per ajudar a qualsevol usuari interessat a posar en marxa el sistema o contribuir amb millores.

### Compilació del Client Web

L'aplicació client, desenvolupada en Flutter, ha de ser **compilada i configurada** per connectar-se correctament amb l'API. Els passos són els següents:

1. **Instal·lació de dependències:**
   - Executa la comanda següent al directori del client per descarregar totes les dependències necessàries:

     **flutter pub get**

2. **Configuració de l'IP del Servidor:**
   - Accedeix al fitxer app_torneig_flutter/lib/conf/ip.dart i substitueix la variable IP per l'adreça IP real del servidor on s'executarà l'API:

     **class Ip { static const IP = "<<ip>>:9090"; }**

   - Exemple per a un entorn local:

     **class Ip { static const IP = "192.168.1.100:9090"; }**

   - Exemple per a un entorn de producció amb AWS:

     **class Ip { static const IP = "18.101.94.248:9090"; // AWS API }**

3. **Construcció de l'Aplicació Web:**
   - Un cop configurada l'IP, compila l'aplicació per generar els fitxers necessaris:

     **flutter build web**

   - Els fitxers es generaran a la carpeta build/web/.

4. **Accés al Client Web:**
   - Després de compilar l'aplicació, accedeix al lloc web des de:
     - Entorn local: http://localhost:8080
     - Altres dispositius a la mateixa xarxa: http://<ipPcOnEstaTotMuntat>:8080

### 2. Ús de l'Aplicació Android

A més de la versió web, s'ha creat una aplicació Android per a ús mòbil. Aquesta aplicació:

- Es pot instal·lar en qualsevol dispositiu amb Android.
- Està dissenyada de forma responsiva per adaptar-se a diferents pantalles.
- Per defecte, està configurada per connectar-se a l'API de AWS, la qual cosa permet

utilitzar-la des de qualsevol lloc fora de l'entorn local.

### 3. Execució dels Contenidors

Un cop configurada i compilada l'aplicació web, es pot desplegar l'entorn local amb **Docker**. Els passos són els següents:

1. Accedeix al directori arrel on es troba el fitxer docker-compose.yml.
2. Executa la comanda:

   **docker-compose up**

3. Això posarà en marxa els següents serveis:
   - **Base de Dades**: MySQL.
   - **API**: Spring Boot.
   - **Client Web**: Servit amb Apache.

### 4. Accés a les Versions Desplegades a AWS

El projecte també està disponible en línia gràcies a la infraestructura desplegada a **AWS**, la qual inclou:

**Base de Dades** en RDS.

**API** en EC2.

**Client Web** servit des de S3 o CloudFront.

Accedeix a les aplicacions a través dels següents enllaços:

**Documentació de l'API (Swagger):** *http://18.101.94.248/swagger-ui/index.html*

**Aplicació Web a AWS:** *http://torneig-client.s3-website.eu-south-2.amazonaws.com*

Exemple del Docke-compose.yml:

```
services:
  dbtorneig:
    container_name: torneig-DB
    build: ./BD
    #restart: unless-stopped
    env_file: ./.env
    environment:
      - MYSQL_ROOT_PASSWORD=$MYSQLDB_ROOT_PASSWORD
      - MYSQL_DATABASE=$MYSQLDB_DATABASE
    ports:
      - $MYSQLDB_LOCAL_PORT:$MYSQLDB_DOCKER_PORT
    volumes:
      - dbtorneig_data:/var/lib/mysql
    networks:
      - torneignet

  apitorneig:
    container_name: torneig-API
    depends_on:
      - dbtorneig
    build: ./Torneig_Mort
    #restart: on-failure
    env_file: ./.env
    ports:
      - $SPRING_LOCAL_PORT:$SPRING_DOCKER_PORT
    environment:
      SPRING_APPLICATION_JSON: '{ "spring.datasource.url"  : "jdbc:mysql://dbtor-
neig/Torneig_escola", "spring.datasource.username" : "$MYSQLDB_USER", "spring.data-
source.password" : "$MYSQLDB_ROOT_PASSWORD", "spring.jpa.hibernate.ddl-auto" : "vali-
date", "spring.jpa.show-sql" : "true" }'
    #stdin_open: true
    #tty: true
```

```
    networks:
      - torneignet

  webtorneig:
    container_name: torneig-WEB
    depends_on:
      - apitorneig
    build: ./app_torneig
    ports:
      - "8080:80"
    networks:
      - torneignet

volumes:
  dbtorneig_data:


networks:
  torneignet:
    driver: bridge
```

## 6.2. Definir les fases d'implantació/migració

La implantació del sistema es realitza de manera gradual per assegurar que cada component funcione correctament abans d'implementar la següent fase. Aquestes són les fases definides:

1. **Preparació de l'entorn:**
   - Instal·lació i configuració de maquinari i xarxes.
   - Configuració de l'entorn de desenvolupament i proves.
2. **Desplegament de la Base de Dades:**
   - Configuració del contenidor de MySQL i inicialització de les dades bàsiques.
   - Creació de còpies de seguretat inicials.
3. **Implementació de l'API:**
   - Desplegament de l'API REST amb Spring Boot en l'entorn local o núvol.
   - Verificació de connexions amb la base de dades.
4. **Desplegament del Client:**

- Configuració i prova del client web i de l'APK d'Android.
- Adaptació a l'entorn de xarxa i proves de funcionalitat.

5. **Proves del Sistema:**
   - Verificació de la integració completa entre base de dades, API i client.
   - Proves de càrrega i rendiment.
6. **Formació d'Usuaris i Posada en Producció:**
   - Realització de sessions formatives per a administradors i usuaris finals.
   - Migració definitiva del sistema a entorn de producció.

## 6.3. Inventari de maquinari i programari

**Maquinari:**

- **Ordinadors:**
  - Equip de desenvolupament: 1 PC amb mínim 8 GB de RAM i 500 GB d'espai.
  - Ordinadors per als usuaris finals per accedir al client web.
- **Servidors:**
  - Per entorn local: servidor local amb suport per a Docker.
  - Per entorn de producció: instància EC2 a AWS.

**Programari:**

- **Base de Dades:** MySQL.
- **API:** Framework Spring Boot amb suport de Maven per a compilació.
- **Client:** Flutter per a web i Android.
- **Contenidors:** Docker i Docker Compose.
- **Eines de suport:** Visual Studio Code, Android Studio, Workbench i SpringTools.

## 6.4. Diagrames de xarxa, estructura

La xarxa queda estructurada de la següent manera després de la implantació:

- **Entorn Local:**
  - Servidor local amb Docker que allotja la base de dades, l'API i el client web.
  - Accés intern a través de l'IP de la xarxa local.
- **Entorn AWS (Producció):**
  - Base de dades MySQL desplegada en AWS RDS.
  - API servida des d'una instància EC2.
  - Client web allotjat en S3.

## 6.5. Migració/Implantació de serveis

Els passos per a la migració o implantació de serveis són:

1. **Migració de dades:**
    - Si hi ha dades existents, caldrà importar-les a la nova base de dades.
    - Ús d'scripts SQL per assegurar la consistència.

2. **Configuració de serveis:**
    - Adreces IP configurades per als servidors.
3. **Verificació de serveis:**
    - Proves per garantir que cada servei (API, client, base de dades) funcione de manera òptima.

## 6.6. Formació, comunicació i suport a l'usuari

**Formació d'Administradors:**

- Sessions específiques per ensenyar-los com gestionar equips, jugadors, i partits des del client.
- Explicació detallada de com actualitzar resultats i supervisar classificacions.

**Formació d'Usuaris Finals:**

- Guia d'ús simple per accedir al sistema, navegar per classificacions, consultar resultats, etc.

**Canal de Suport:**

- Creació d'un canal de comunicació per resoldre dubtes o problemes tècnics (correu electrònic o xat).

# 7. Conclusions i treballs futurs

## 7.1. Conclusions

Aquest projecte ha aconseguit complir els objectius plantejats inicialment, oferint una solució integral per a la gestió de tornejos esportius al col·legi Joanot Martorell de Xeraco. Els principals resultats aconseguits són:

1. **Digitalització de la Gestió del Torneig:**
   - Hem substituït els processos manuals per un sistema digital, millorant l'eficiència i reduint els errors.
   - Els administradors poden gestionar equips, jugadors i resultats de manera senzilla i centralitzada.

2. **Experiència d'Usuari Millorada:**
   - Els jugadors, aficionats i altres usuaris poden consultar informació del torneig (resultats, classificacions, etc.) en temps real mitjançant el client web o l'aplicació mòbil.

3. **Infraestructura Escalable i Flexible:**
   - La implementació amb Docker i AWS permet que el sistema s'adapte fàcilment a diferents entorns, assegurant la seva funcionalitat tant en un context local com en un entorn de producció.
   - L'ús de tecnologies modernes com Spring Boot i Flutter assegura una plataforma robusta i actualitzable.

4. **Bones Pràctiques de Desenvolupament:**
   - L'aplicació segueix principis de modularitat, reutilització de codi i separació de responsabilitats, assegurant una arquitectura fàcil de mantenir i millorar.

Aquest projecte no només resol les necessitats actuals del col·legi sinó que també representa una base sòlida per adaptar-lo a altres entitats o escenaris similars.

## 7.2. Treballs Futurs

Tot i que el projecte compleix els objectius inicials, hi ha diverses millores i ampliacions que podrien implementar-se en el futur per augmentar-ne les funcionalitats i la versatilitat:

1. **Integració amb altres Sistemes:**
   - Implementar la possibilitat d'exportar dades (classificacions, resultats, estadístiques) a formats com Excel o PDF.
   - Integrar notificacions automàtiques via correu electrònic o aplicacions de missatgeria (ex. WhatsApp, Telegram) per informar sobre canvis en els partits o classificacions.

2. **Millores en la Usabilitat del Client:**
    - Afegir funcions interactives al client, com gràfics en temps real per visualitzar estadístiques dels equips i jugadors.
    - Optimitzar el disseny de l'aplicació per a una millor experiència en dispositius mòbils.

3. **Expansió de Funcionalitats:**
    - Incloure una secció per gestionar estadístiques avançades dels jugadors (gols, assistències, targetes, etc.).
    - Crear un mòdul específic per a la gestió d'inscripcions en línia de jugadors i equips.

4. **Escalabilitat del Sistema:**
    - Permetre la gestió de múltiples tornejos simultàniament des d'una mateixa plataforma.
    - Facilitar l'adaptació del sistema a altres centres educatius o clubs esportius, oferint una configuració inicial simplificada.

5. **Seguretat i Rendiment:**
    - Implementar un sistema de permisos més granular que permeta gestionar rols específics amb accés limitat a funcionalitats concretes.
    - Realitzar proves de càrrega a gran escala per assegurar el rendiment del sistema en contextos amb molts usuaris.

## 7.3 Conclusió Final

Aquest projecte representa una solució moderna i efectiva per a la gestió de tornejos esportius, resolent les necessitats actuals del col·legi i establint una base sòlida per a futures millores. L'ús d'eines i tecnologies modernes, combinat amb bones pràctiques de desenvolupament, assegura un sistema escalable, flexible i fàcil de mantenir.

A llarg termini, aquest projecte té un gran potencial per ser adaptat i implementat en altres institucions esportives, oferint una solució genèrica però configurable. Així mateix, les millores proposades permetran ampliar les capacitats del sistema, assegurant que continue sent útil i rellevant amb el temps.

# 8. Bibliografia

**Documentació Oficial de Flutter**

- Flutter: Build apps for any screen. https://flutter.dev/
- Referència oficial per al desenvolupament d'aplicacions web i mòbils multiplataforma amb Flutter.

**Documentació Oficial de Spring Boot**

- Spring Boot Reference Documentation. https://docs.spring.io/spring-boot/
- Guia completa per desenvolupar aplicacions empresarials amb Spring Boot.

**Docker**

- Docker Documentation. https://docs.docker.com/
- Guia per a la creació i gestió de contenidors, incloent Docker Compose.

**AWS**

- AWS Documentation. https://docs.aws.amazon.com/
- Documentació sobre els serveis utilitzats, com RDS, EC2, S3 i CloudFront.

**MySQL**

- MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/refman/8.0/en/
- Documentació oficial per configurar i gestionar bases de dades MySQL.

**Swagger/OpenAPI**

- Swagger Documentation. https://swagger.io/docs/
- Guia per utilitzar OpenAPI amb Spring Boot i generar documentació interactiva de l'API.

**Lombok**

- Project Lombok. https://projectlombok.org/
- Documentació oficial per simplificar el codi Java amb anotacions.

**ModelMapper**

- ModelMapper Documentation. https://modelmapper.org/
- Guia per transformar objectes entre capes de l'aplicació de manera senzilla i eficient.

**Material Design**

- Material Design Guidelines. https://m3.material.io/
- Referència utilitzada per dissenyar una interfície d'usuari responsiva i coherent.

**GitHub**

- The Complete GitHub Guide. https://docs.github.com/es
- Documentació per gestionar el repositori del projecte i facilitar la col·laboració.

**JSON Web Tokens (JWT)**

- Introduction to JSON Web Tokens. https://jwt.io/
- Guia per implementar autenticació i autorització amb JWT en l'API.