UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

SpotSport: una aplicació mòbil per a organitzar trobades esportives

---

Adrià Valls I Morta

Director: Albert Clapes
Realitzat a: Departament de
Matemàtiques i informàtica

Barcelona, 20 de juliol de 2017

Índex

Resumen

Con un aumento en la sedentariedad en la vida moderna, el hacer ejercicio puede parecer una tarea que tienes que hacer y no una actividad de la que tengas ganas de ser partícipe y que puedas pasarlo bien, por eso he querido crear una herramienta en la forma de una aplicación móvil que te permita encontrar gente con las mismas ganas que tu de participar en actividades deportivas. También he investigado sobre los métodos de recomendación y matchmaking para dar una experiencia más ajustada a cada usuario.

Resum

Amb un augment de la sedentaritat en la vida moderna, fer activitat física pot semblar una tasca més a fer i no una activitat de la que tens ganes de ser participant i passar-ho bé, per això he volgut crear una eina en forma d'una aplicació mòbil que faciliti els esdeveniments on persones poden trobar altres persones amb les mateixes ganes de participar en activitats esportives. També he investigat sobre mètodes de recomanació i matchmaking per donar una experiència més conforme a cada usuari.

Abstract

With an increase of the sedentary lifestyle that the modern life comes with, doing some physical activities might feel like you are doing a task that you have to do and not like a fun activity that you want to be participating in, that is why i wanted to make a mobile application that allow you to find other people to share and be part of a sportive activity. I have also investigated recommendation and matchmaking services to give the user a better and closer experience.

1. Introducció i motivació

L'activitat física és un component molt important en un estil de vida saludable, la Organització Mundial de la Salut (OMS, o WHO en anglès) recomana entre 150 i 300 minuts d'activitat física moderada durant la setmana per a persones d'entre 18 i 64 anys per promoure una bona salut física i mental i menor risc de mort prematura.

La OMS també alerta en el seu *Global status report on physical activity 2022* [1] de que un 80% dels adolescents i un 30% dels adults no arriben als nivells recomanats d'activitat física i això pot provocar un increment en casos d'hipertensió, molts d'ells causats per depressió. Amb l'arribada del covid 19 aquest número va empitjorar.

Jo mateix he notat que durant aquests últims anys i sobretot després de la pandemia em vaig trobar amb que cada vegada feia menys activitat física, un sentiment que altre gent també m'ha compartit. Personalment anar a córrer, sortir en bicicleta o anar al gimnàs, tot i que efectives, no son activitats que tingui ganes de fer, com per exemple fer un partit d'algun esport que m'agrada com bàsquet o voleibol. El problema amb aquesta última opció és la dificultat per a trobar prou gent per organitzar un partit o realitzar una d'aquestes activitats.

Seguint aquest fil vaig tenir la idea de l'aplicació que presento, la idea principal es crear una aplicació mòbil on els usuaris puguin crear i unir-se a activitats esportives amb altres usuaris. Per millorar l'experiència, també he creat un sistema de matchmaking per recomanar i poder emparellar usuaris amb altres usuaris que tinguin un nivell d'habilitat similar.

2. Estat de l'art

En aquest apartat he investigat diverses aplicacions de la play store de google que ofereixen un servei similar al que jo vull desenvolupar, mirant aquestes apps m'he centrat sobretot en quatre aspectes importants que vull que la meva aplicació sigui capaç de donar:

-Varietat d'activitats: L'app ha de donar suport a varis esports per a oferir una experiència més global per a tots els usuaris i tenir un ventall de possibilitats més ampli per al usuari.

-Sistema de recomanació: Hi ha d'haver un apartat específicament on els usuaris puguin veure les activitats recomanades específicament per a ells.

-Separació per nivell: Els usuaris han de poder tenir la possibilitat de trobar altres persones amb un nivell similar per tenir partits igualats si es vol.

-Creació d'activitats: Un usuari ha de poder crear partits i unir-se a partits que han creat altres usuaris.

Aquests quatre punts són el que tinc en ment a l'hora de comparar amb les altres aplicacions, també busco elements interessants que tinguin aquestes apps.

La primera app que he investigat es diu *Find a player*, aquesta és la més semblant al que vull fer jo, té la opció de crear activitats de qualsevol tipus, de crear un partit per compartir amb la gent i integra un mapa on es poden veure les diferents activitats marcades. Aquesta part amb mapa és un punt molt interessant que,tot i no haver-lo inclós en el marc d'aquest treball, seria una opció buscada en un treball futur. El que li manca a aquesta app en comparació amb el que he vull fer, es el sistema de recomanació.

Una altra app, *Padel CPI*, és - com el seu nom indica - l'aplicació d'un establiment de pàdel. Aquesta té opció de reservar una pista amb gent coneguda, però també té l'opció de trobar gent amb un nivell similar al teu. No obstant,per a determinar el teu nivell, has de fer unes proves físicament a l'establiment. Això, tot i tenir l'avantatge de que la puntuació resulti més exacta, té l'inconvenient de necessitar persones que puguin distingir les habilitats dels usuaris. Per tant, no es un sistema que em sigui convenient ja que és massa específic i requereix de personal extra i un nivell de logística massa fort.

Una altra app semblant és *Lineup*, que té un sistema de crear i trobar partits interessant i que engloba diferents esports. Ofereix estadístiques de jugadors, però no té un sistema de nivells ni recomanació.

La app anomenada *Fubles* té una versió de puntuar als usuaris interessant, essent els propis usuaris que voten el nivell dels altres usuaris.

Padel Mates és una app que s'assembla a la *CPI Pade*l, però més general, amb la qual pots buscar partits a diferents establiments i crear-los tu mateix, però igual que amb Padel CPI, és massa específica i no té sistema de recomanació.

Hudle, permet buscar una varietat d'activitats pero no et permet crear cap partit, serveix més com a un hub on els establiments que poden donar serveis i instal·lacions on fer esport es poden anunciar als usuaris. De totes les aplicacions és potser la que dona el servei més diferent a la meva idea, ja que, el que he pensat és més per la interacció entre usuaris no entre establiment i usuari. Tampoc té sistema de recomanació ni de nivell.

Amb la informació que he recopilat de cada aplicació, presento un resum en forma de taula que ensenya les idees principals i com s'ajusta cada aplicació que he investigat.

App (nom)	Varietat d'activitats	Sistema de recomanació	Separació per nivell	Creació d'activitats
Find a player	si	no	si	si
Padel cpi	no	no	si	si
Lineup	si	no	no	si
Fubles	no	no	no	si
Sport Easy	no	no	no	si
Padel Mates	no	no	no	si
Hudle	si	no	no	no

3. Planificació

He dividit la planificació del projecte en quatre parts.
Primer tinc el segment general, on he abordat les parts menys específiques del projecte com redactar la memòria; la part de definir el projecte que és quan intento trobar l'abast del projecte inicial i saber què vull fer exactament; finalment la recerca de tecnologíes que és quan decideixo quins programes i llenguatges vull fer servir per l'elaboració del projecte.
El segon apartat es el de frontend, dividit per la programació de les pantalles com els sis primers subapartats i la programació dels serveis.
L'apartat de recomanador està subdividit en tres subapartats, el primer es la recerca dels diferents tipus de matchmaking, el segon és la implementació de l'algoritme que he decidit i el tercer es la programació de la simulació de partits.
Per últim tinc la construcció del backend, començant amb la base de dades i seguint amb la API, dividida en les tres últimes subdivisions.

En la Figura 1 es pot veure el diagrama de Gantt que he utilitzat com a planificació per el treball, es pot veure amb major resolució a l'*annex 1*.

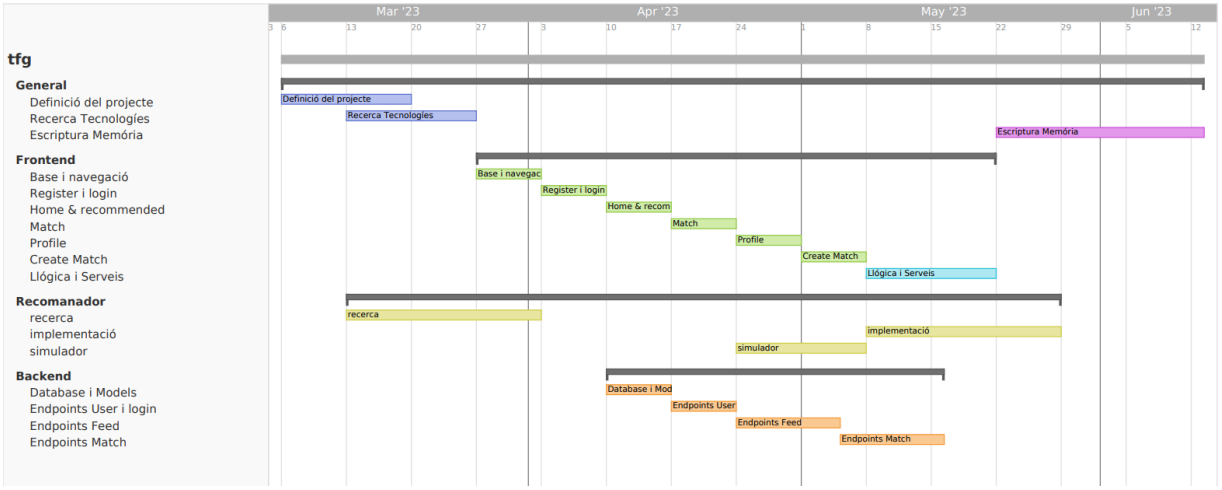


Figura 1. Taula de Gantt

El pla final va resultar ser bastant similar al planificat inicialment, excepte per la part de lògica i serveis, la pantalla de crear partit i la creació de la base de dades, desviant-se una setmana respecte la planificació inicial La resta de punts es van cenyir força igualats amb el que havia previst.

4. Disseny i Implementació

La idea principal per l'aplicació es crear un lloc, una comunitat on els usuaris siguin capaços de trobar altres persones amb qui compartir un esport. Per tant, el bloc central de l'aplicació és la creació i el disseny orbita al voltant del centre.

User stories
Per definir el disseny de l'aplicació he creat un seguit de user stories que són l'estructura que he seguit durant el desenvolupament de l'aplicació.

Com he explicat abans, jo he designat com el centre de l'aplicació, les funcions generals que ha de poder executar la meva aplicació de manera que jo considedo que la part central de la app és funcional.

Aquestes primeres user stories són anomenades Épiques i són un tipus de user stories de molt alt nivell definicional, molt generals i que descriuen una acció molt àmplia. Més endavant seran subdividides per generar les user stories més específiques.

4.1 Épiques:

1. Com a usuari vull tenir un compte d'usuari per poder fer ús de les funcions de la app

2. Com a usuari vull crear una activitat esportiva per poder buscar a gent amb qui compartir l'esport

3. Com a usuari vull veure els partits disponibles per poder-me trobar altres usuaris que busquen un company per fer esport.

4. Com a usuari vull ajuntar-me amb altres usuaris per fer esport.

4.2 User Stories

Partint de les Épiques, es pot segmentar el desenvolupament de la app en la següent llista de user stories:

1.1. Com a usuari vull crear un compte d'usuari per interactuar amb l'aplicació.
1.2. Com a usuari vull tenir un nom d'usuari únic per poder-me distingir dels altres usuaris.
1.3. Com a usuari vull poder iniciar sessió a l'aplicació amb el mau usuari per fer-ne ús.
1.4. Com a usuari vull tenir un perfil on veure la meva informació relacionada amb l'aplicació.
1.5. Com a usuari vull modificar la meva informació d'usuari per ajustar-la si és necessari.
1.6. Com a usuari vull esborrar el meu compte d'usuari si és necessari.

2.1. Com a usuari vull crear una activitat que contingui el tipus d'activitat (esport), data i hora
2.2. Com a usuari vull que les meves activitats siguin visibles als altres usuaris perquè s'hi puguin apuntar.
2.2.. Com a usuari vull poder veure els partits que he creat per poder administrar i informar-me adequadament.
2.2. Com a usuari vull vull poder veure els partits on m'he apuntat per poder administrar i informar-me adequadament.
2.3. Com a usuari vull poder apuntar dels resultats de les activitats que he creat per donar informació dels usuaris participants.

3.1. Com a usuari vull veure una llista de partits disponibles per poder-me apuntar i ser part de l'activitat.
3.2 Com a usuari vull veure la informació d'un partit específic per poder-me informar abans d'apuntar-me o formar part de l'activitat.
3.2.1 Com a usuari vull veure l'usuari que ha creat el partit per estar informat a l'hora de decidir si unir-me o sortir d'una activitat.
3.2.2 Com a usuari vull veure els usuaris que s'han apuntat en el partit per estar informat a l'hora de decidir si unir-me o sortir d'una activitat.
3.3. Com a usuari vull veure una llista de partits recomanats específicament per trobar una activitat que s'ajusti millor a les meves necessitats
3.4. Com a usuari vull veure una llista de partits segons el tipus d'esport per trobar una activitat que s'ajusti més adequadament a les meves necessitats

4.1 Com a usuari vull poder unir-me a una activitat per poder fer esport amb altres usuaris.
4.2 Com a usuari vull poder desapuntar-me d'una activitat si ja no m'interessa.

4.3 Codi

Per al bon funcionament d'una aplicació web o mòbil no vols que tota la feina la faci la part del client (frontend), ja que això omple massa el dispositiu de l'usuari i és millor tenir una aplicació més lleugera i fer que la feina per la part del client sigui més simple. Una altre problema és l'emmagatzematge de les dades i el tractament, si volem tenir més d'un usuari necessitem un lloc sigui possible la connexió entre usuaris. És per això que volem tenir diferents parts de l'aplicació separats a llocs adequats, la part de client, on l'usuari interactuara amb l'aplicació i la part del servidor, on l'aplicació va a buscar la informació que mostra a l'usuari.

El codi de l'aplicació està dividit en dues parts independents, el frontend i el backend, els quals s'executen independentment.
El frontend agrupa el que és l'aplicació que ha de correr l'usuari per utilitzar el software.
Per fer el backend necessitem una base de dades per guardar la informació dels usuaris, partits, etc. Però també necessitem un lloc on l'aplicació per part de l'usuari pugui accedir a les dades que necessita per funcionar correctament, aquí és on entra la API. Una API, o application programming interface, es una aplicació que permet connectar dos programes, en aquest cas el frontend i la base de dades utilitzant una serie de comandes, per exemple, si un usuari vol demanar la informació del partit, l'aplicació del client farà una demanda a la API, aquesta buscarà a la base de dades i retornarà la informació demanada a l'usuari.

Dins de la memòria però, utilitzaré una separació diferent. Dins del backend he investigat l'ús de recomanadors per a proposar partits als usuaris que s'adeqüi millor al nivell de l'usuari si aquest vol. Aquesta part la considero suficientment diferent de la resta del backend com per a separar-la. Per tant, estaré fent ús de la següent separació general del codi:

·Frontend
·Backend
·Recomanador

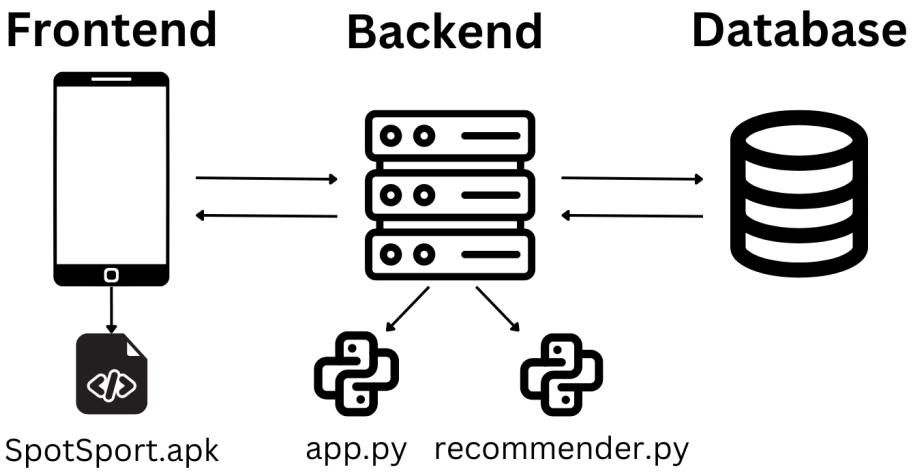


Figura 2. diagrama de l'aplicació

4.4 Frontend

Per construir el frontend d'una aplicació web hi ha tres elements que són importants. Primer tenim html, un llenguatge que dona la estructura i el contingut a l'aplicació. Després tenim el css que ens ajuda a modificar l'estil de la nostra aplicació. Per últim tenim TypeScript amb el que donarem funció a l'aplicació. Amb aquest tres elements podem crear el que vulguem, pero si el que estem és molt gran, com per exemple una web amb moltes pàgines, sempre és millor tenir una bona organització. És per això que utilitzem el que se'n diu *frameworks*, eines que ajuden a posar un ordre a tots els arxius del teu programa i donen funcionalitat extra.

El Frontend està fet amb *Ionic*, una SDK que està preparada per ajudar a crear aplicacions multiplataforma amb una sola base de codi i dona eines per la interfície d'usuari per un bon funcionament en mòbil. Això vol dir que, tot i que durant aquest treball estigui utilitzant el meu codi només com a aplicació mòbil, el mateix codi es pot adaptar per a funcionar com a pàgina web.
Ionic dona una compatibilitat més senzilla amb l'eina capacitor, la qual fa possible la interoperabilitat i transformació de l'aplicació web a aplicació mòbil i amb la qual es pot crear un apk a partir del codi d'*Angular*.

Per la programació utilitzo *Angular*, un framework de TypeScript (una vertent de Javascript) basat en components, que proporciona una forma de programar web amb html, css i TypeScript més ordenada i ofereix llibreries per a fer més fàcil la programació de diferents funcions de l'aplicació web.*Angular* funciona amb components i cada element està compost per quatre fitxers diferents un .html, un. css i dos .ts (el format per fitxers typeScript) un per la llògica interna que se li vulgui donar i l'altre per la configuració del projecte (mòdul). El que et permet fer aquesta tecnología és crear elements els quals es poden utilitzar independentment per tota l'aplicació mòbil o web.

En el meu projecte també faig servir una variant dels components que et facilita *Ionic*:les pàgines. Tot i ser molt semblants als components, són una forma de dividir les parts més importants del codi i donen accés a tenir un routing propi i una independència més marcada entre cada element principal del frontend .
A la Figura 3, s'hi poden veure les pàgines que té la meva aplicació:

-Create Match: vista on l'usuari pot crear un partit.
-Home: pàgina principal de l'aplicació.
-Login: on l'usuari s'identifica per poder accedir a la app.
-Match: on s'ensenya tota la informació d'un partit específic.
-Profile: on un usuari pot trobar la seva informació.
-Recommended: on hi apareix el llistat de partits que es recomanen a l'usuari.
-Register: on un usuari pot crear-s'hi un compte.

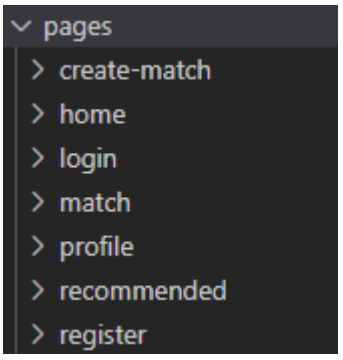


Figura 3, Pàgines

Aquest és el cos principal del frontend de l'aplicació, aquestes pàgines són encapsulades dins d'un sistema de "tabs" (o pestanyes), dins de la mateixa pàgina principal. En el meu cas, es diu tabs.page.ts i hi definim un seguit de rutes utilitzant la funció de routing amb Angular, indicant el mòdul de la pàgina associada amb cada ruta. Un cop definides les rutes, podem utilitzar una barra de navegació per decidir quina pàgina volem mostrar, donant la opció a l'usuari de navegar per l'aplicació sense la sensació de canviar de pàgina.

4.4.1 Disseny

Vull que la meva aplicació sigui simple d'utilitzar per fer la transacció d'entrar, buscar o crear un partit fàcilment.. Per això, he optat per un disseny minimalista, amb el qual busco ensenyar la informació justa per facilitar a l'usuari la interacció i enteniment de l'aplicació sense atapeir la vista.

Els colors també els he volgut mantenir simples: pel fons l'aplicació he utilitzar blanc (#FFFFFF), pel text negre (#000000) i per donar una mica de color he utilitzar un blau marí (#1A4F84)
Dins de l'*annex 2* es pot veure un diagrama de navegació de la app on es poden veure les pàgines que s'explicaran en els segments següents en el context global de navegació.

A la Figura 4, s'hi pot veure el disseny de la pàgina "tabs". Essent aquesta la base de l'aplicació i on s'hi mostrarà la resta de les pàgines; dins de l'espai entre l'encabeçat o header (Figura 5) i la barra de navegació (Figura 6).

El header ensenya el nom de l'aplicació i la imatge de perfil de l'usuari que funciona com a botó, si s'apreta es navega a la pàgina de perfil de l'usuari.

A la part d'abaix, hi tenim la barra de navegació amb quatre botons, començant per l'esquerra tenim la pàgina principal o home, la pàgina de recomanacions, el creador de partits i el perfil de l'usuari.

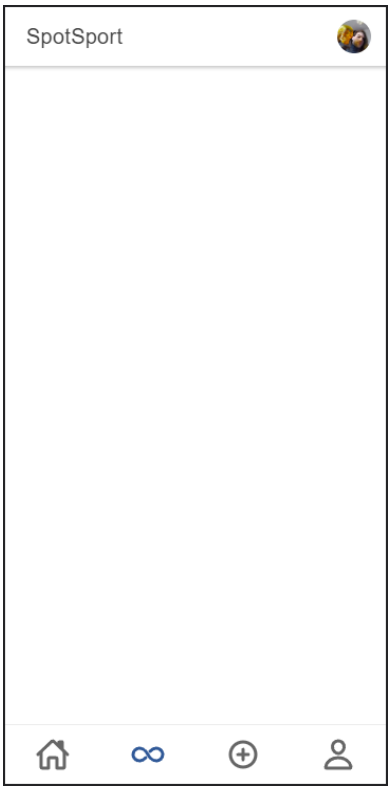


Figura 4. Base de la app

Figura 5. Header

Figura 6. Barra de navegació

Component match-card

Aquest és un component que utilitzo a diferents lloc dins de la app, es tracta d'una carta que ensenya un petit resum del partit. La informació la rep mitjançant una funció d'*Angular* anomenada Input, que serveix per passar informació. En aquest cas, un partit en forma d'objecte des de dins d'altres components que facin servir el match-card. També utilitzo una funció contraria a

Figura 7. Component match

l'anterior anomenada Output, que em permet crear un event, el que s'utilitza al apretar el botó d'abaix a la dreta, que s'emet al component receptor per passar la id del partit i permet decidir què fer amb aquesta informació. En la meva aplicació s'utilitza majoritàriament per navegar a la pàgina del partit indicat.

Pantalles de registre i login

Dues pantalles semblants, són la pantalla de registre (Figura 8) i la pantalla de login (Figura 9), amb un formulari per omplir i un botó per enviar la informació al servidor. Si l'enregistrament és correcte, et redirigeix a la pantalla de login i si el login és correcte et redirigeix a la pantalla principal "home".

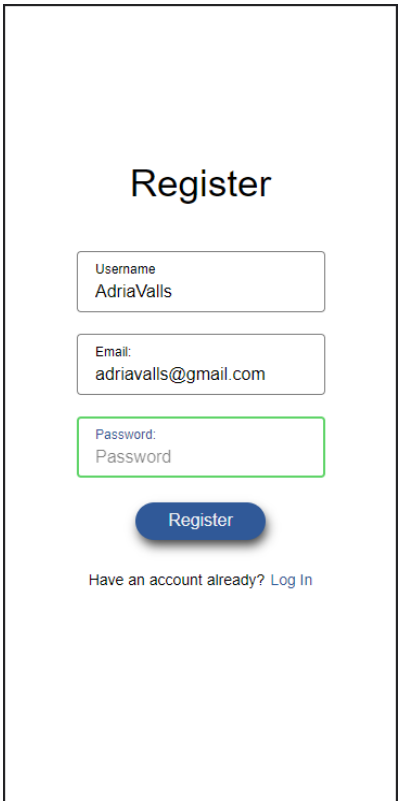


Figura 8. Pantalla register

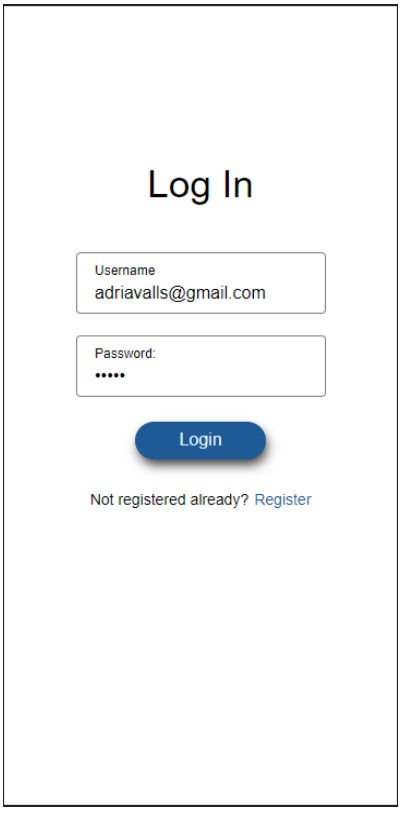


Figura 9 Pantalla login

Home Page

Aquesta és la primera pàgina que veu l'usuari, on hi he volgut ensenyar una llista amb tots els partits més recents. A sobre de la llista, hi ha un selector d'esports per a poder filtrar la llista de partits segons l'esport que l'usuari seleccioni.

La llista s'inicialitza a l'entrar a la pàgina, demanant amb el mètode getMainFeed el llistat de partits. La resposta del backend retorna la informació de cada partit i la guardo com a objecte a la llista, utilitzant un component que he creat anomenat match-card i com he explicat anteriorment li passo la informació amb Input. El component match-card té l'output de la id del match amb la que, utilitzant el mètode gotoMatch, es navega a la pàgina del partit indicat.

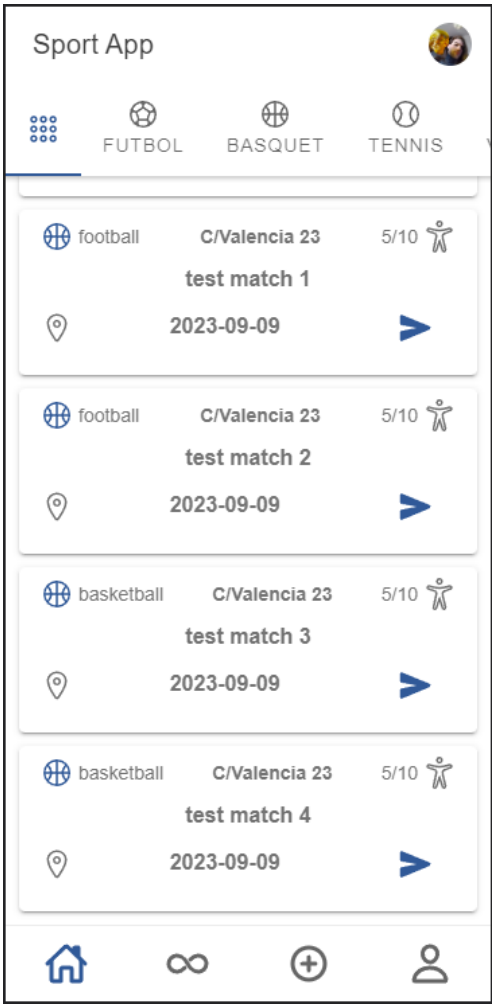


Figura 10. Pantalla home

Match

La pàgina de match ensenya tota la informació del partit, com que s'accedeix des de les altres pàgines, com home o profile. He modificat el component de header per afegir un botó que porti a la pàgina anterior i facilita la navegació. També he posat el títol del partit dins del header. A la part del contingut, hi ha dues finestres. La primera (Figura 11) es per veure tota la informació rellevant, com la descripció la data i hora, qui ha creat el partit i la direcció.
A la segona pàgina (figura 12), s'hi poden veure tots els participants que s'han apuntat al partit.
Si ets el propietari del partit i hi ha suficients jugadors a la pestanya d'informació apareixen dos botons amb els que pots indicar quin equip ha guanyat (figura 13).

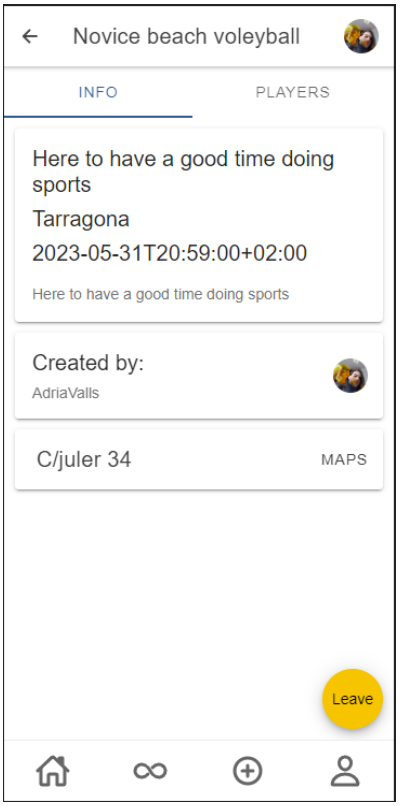


Figura 11. Pantalla match

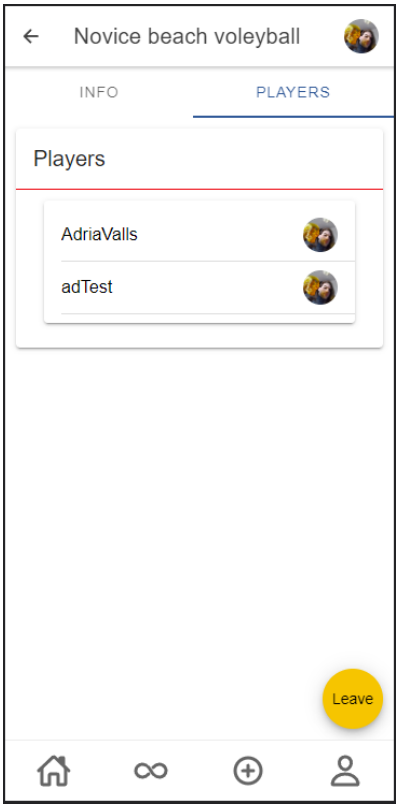


Figura 12. Pantalla jugadors

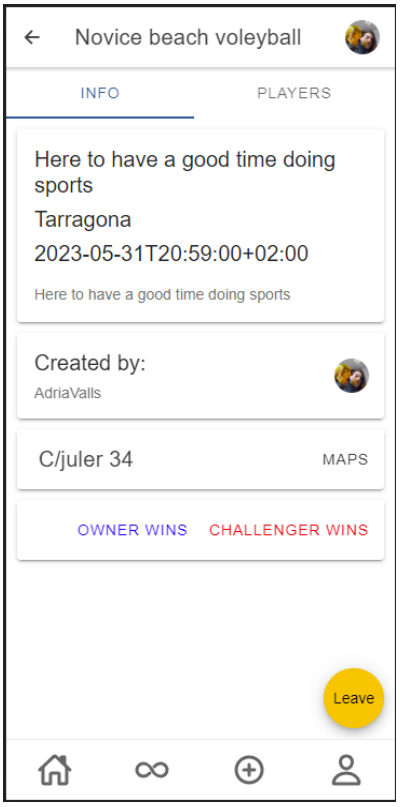


Figura 13. Pantalla owner

Create match

És la pàgina on l'usuari entra si vol crear un partit, està dividida en quatre parts, cada una demanant un requeriment per omplir la informació necessària per crear el partit, una pantalla per seleccionar quin tipus d'esport es vol jugar (Figura 14), la següent per seleccionar data i hora (Figura 15), una per saber la localització (Figura 16) i, finalment, demanar la informació general que l'usuari pot omplir amb el que li sembli més adequat (Figura 17).

La navegació dins de la pàgina es fa a través dels botons a la part inferior de la pantalla, tot i que semblin quatre pàgines diferents estan totes en el mateix component i utilitzo un joc de visualitzar el elements actius i amagar els que no vull que es vegin segons un contador que marca quina finestra vull que es vegi. Els punts que marquen el progrés a sota dels botons canvien de classe de la mateixa forma, i els botons s'amaguen igual, per exemple en la Figura 14 estan amagats el botó de previous i finish, i en la següent el botó de previous es deixa veure. Aquest control el faig cridant el mètode de selectTab, passant-li un boolean per a que canviï el punter que marca la finestra actual i segons aquest punter fan el canvi tots els elements. Si s'intenta passar de pàgina sense haver omplert la informació necessària surt un popup informant del problema i no deixa passar de pàgina fins que no s'ha omplert.

Un cop recuperada tota la informació es crida el servei de newMatch per pujar el nou partit al backend.

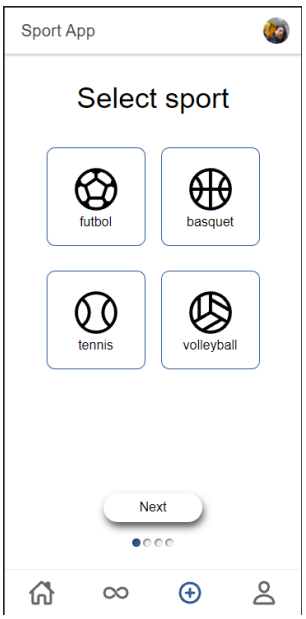


Figura 14. Escollir esport

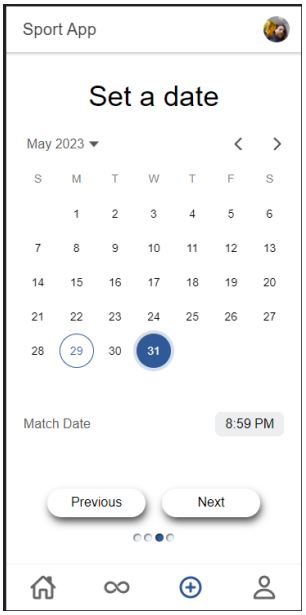


Figura 15. Escollir data

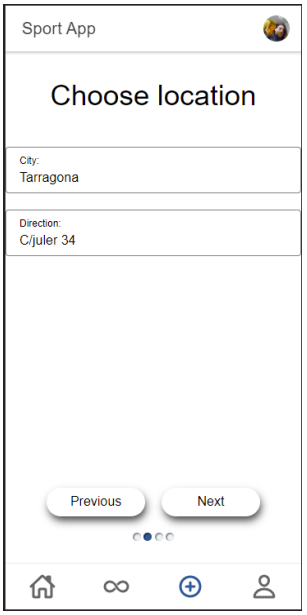


Figura 16. Escollir lloc

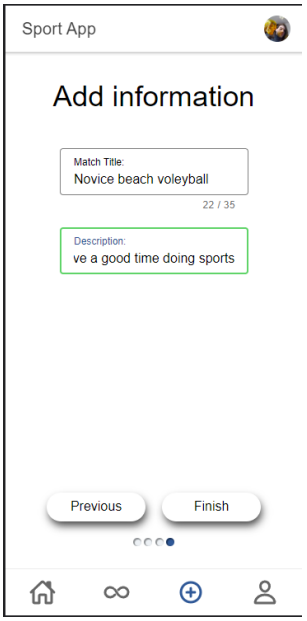


Figura 17. Afegir informació

Profile

Profile és la pàgina on l'usuari pot anar a veure la informació relacionada amb els seu, o qualsevol compte d'usuari. S'ensenya el nom d'usuari i la biografía (informació que l'usuari vol donar de si mateix), i hi ha tres pestanyes a la part inferior per veure els partits on s'ha unit (Joined), els partits que ha creat (Owned) i opcions per la configuració.

Les dues llistes funcionen de manera semblant a la pàgina principal (Home), però demanen les llistes de partits a dos serveis diferents, pels partits en què s'ha apuntat es crida a getJoinedMatches i per veure els seus partits es crida a getOwnedMatches, els dos explicats en la següent secció.

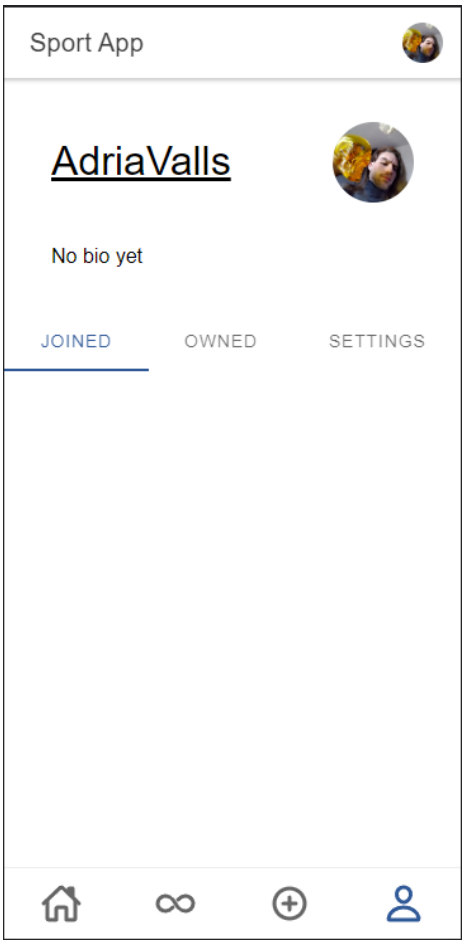


Figura 18. Pantalla perfil

Recommended

La pantalla de recomanació es força senzilla, similar a la pantalla principal té un filtre triar quin tipus d'esport es vol i un cop escollit s'ensenya el partit que el backend ha decidit recomanar a l'usuari.

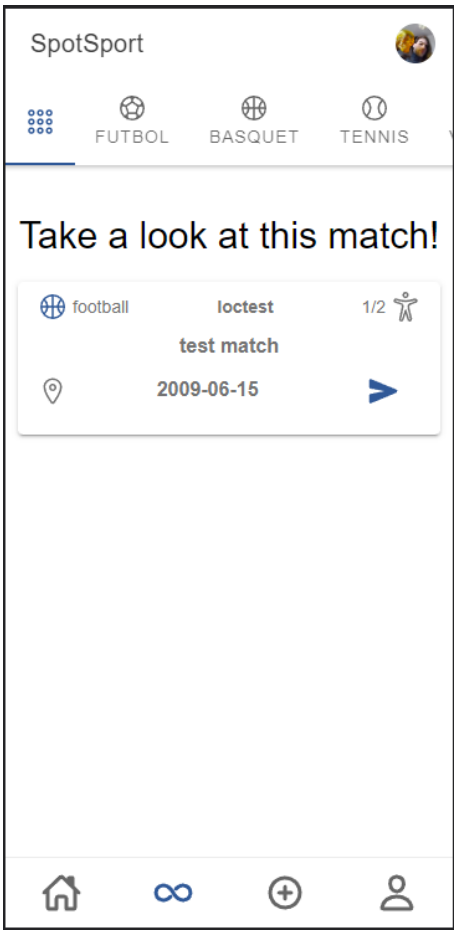


Figura 19. Pantalla recommended

4.4.2 Lògica i Serveis

Cada pàgina o component té el seu fitxer ".ts" on s'hi poden programar les funcions com les que he explicat en l'apartat anterior, pero si es depèn massa de programar a dins del component hi pot haver problemes de codi repetit, funcions i objectes massa dependents i amb problemes de coupling. Per a solucionar-ho, es poden programar serveis (services), és a dir, trossos de codi independents que es poden importar i utilitzar arreu del codi.

Els serveis que utilitzo actualment dins l'aplicació estan repartits en cinc fitxers diferents: feed.service.ts, match.service.ts, recommended.service.ts, session.service.ts i user.service.ts. Tots els serveis els utilitzo per fer la connexió amb el backend mitjançant http requests.

El Servei *Feed* s'encarrega de demanar llistes de partits, per això hi ha tres mètodes: getMainFeed, per agafar els partits que van a la pàgina principal; getOwnedMatches i getJoinedMatches, a partir d'una id de partit un demana els partits que ha creat un usuari i els partits on s'ha unit respectivament.

El Servei *Match* controla tot el que té a veure amb un partit amb els següents mètodes: newMatch, per crear un partit nou a partir d'un json amb les dades del partit que es vulgui crear; getMatch, amb una id retorna la informació associada amb el partit amb aquella id; getMatchPlayers, a partir d'una id de partit retorna la informació dels usuaris apuntats en el partit; joinMatch, donada una id de partit envia una request al backend per unir l'usuari al partit indicat; leaveMatch, amb la id donada envia una petició per desapuntar l'usuari del partit; endMatch, amb dues id una del partit i l'altre de l'equip guanyador per indicar a backend els resultats del partit.

El Servei *Recommended*, similar al main feed, recommended té getRecommendedFeed el qual demana la llista de partits que el backend li recomana a l'usuari.

Session serveix per controlar la part de la sessió d'usuari, amb el mètode de login comprovo quin usuari està fent servir l'aplicació, i amb register envio el conjunt de dades per crear un nou usuari

El Servei *User* el vaig crear per obtenir informació relacionada amb usuaris, amb dos mètodes un getUserId per saber la id de l'usuari corrent i un getUserInfo que, a partir d'una id, et dona la informació necessària de l'usuari corresponent.

4.5 Backend

El backend està programat en python, necessito una base de dades i una API per accedir-hi i modificar-la. Per fer la API, he decidit utilitzar el framework de Flask. Aquest és un framework simple que es pot ampliar utilitzant llibreries per donar-li més utilitats. Per la interacció amb la base de dades, he fet servir la llibreria que dona un servei d'object relational mapper (ORM) anomenat Sqlalchemy. Un ORM s una eina per interactuar amb una base de dades, tant fer una query com manipular dades amb un estil orientat a objectes.

El codi està situat dins del directori TFGBackend i té tres subdirectoris: models, resources, i recommender.

Dins del directori principal hi han el arxius que fan correr la API, el primer de tots i el més important es app.py. A dins es defineix la APIi és on es construeix la app de flask. Primer, s'escriuen tots els imports que faig servir, tant de la llibreria de flask com de les classes de l'ORM. Després es defineix la app com a una instància de flask, seguit de la configuració i definint la base de dades, s'inicialitza a APIi de la base de dades i es defineixen els endpoints de la API. El següent arxiu és config.py, tot i que la configuració es pot fer dins de app.py, és més segur diferenciar-ho separadament, també va bé a l'hora de fer el deployment ja que la configuració es fa des del servidor. L'últim fitxer important del primer repositori es el de db.py, on es defineix la base de dades com a una instància de SQLAlchemy.

Al directori models, és on hi han les classes que utilitzo en l'ORM per generar els objectes de la base de dades, els quals explicaré en l'apartat de models.

A resources hi guardo les classes que utilitzo com a endpoints, cada endpoint i classe té els mètodes definits necessaris per utilitzar les funcions de CRUD (Create, read, update and delete, les funcions bàsiques de les http requests).

Com he explicat anteriorment, recommender està explicat a la secció de Recomanador.

4.5.1 Base de dades

Per aquest projecte no necessito una base de dades molt complexa per això vaig optar per utilitzar una base de dades sql. Per generar-la i utilitzar-la faig servir SqlAlchemy, que em permet tractar models com a objectes dins de la base de dades.

4.5.2 Models

Els models es generen dins d'una classe. Primer s'han de definir les columnes, indicant el tipus de dada i altres variables com si l'element és una clau primària, si és únic, si pot ser null i altres "flags" que siguin necessaris. En la base de dades, hi tinc definides tres taules:

-Match
-User
-Player_in_match

Match l'utilitzo per guardar els partits, User per els usuaris i la taula Player_in_match És una taula especial, ja que, l'utilitzo per generar una connexió many-to-many entre les altres dues i, llavors, saber quins usuaris estan apuntats a cada partit. Dins dels models també es poden definir relacions entre taules, com per exemple la relació one-to-many owner/owned_matches entre user i match, on un usuari pot ser el creador de molts partits, però un partit només pot tenir un creador.

Figura 20, diagrama de la base de dades

4.5.3 API

La API és el servei que dona accés a les dades a l'aplicació, utilitzant una sèrie d'endpoints als que es poden fer peticions que permeten la interacció entre backend i frontend.

Construïts al directori de resources, els endpoints els defineixo en classes on cada mètode és una operació CRUD. He dividit les classes segons l'ús de cada una i les he posat en cinc arxius diferents:

-accounts, les classes d'aquest arxiu controlen tot el que té a veure amb l'usuari, com ara crear un compte o accedir-hi amb la classe Account, o retornar informació d'un usuari com UserInfo o UserId.

-feed, per demanar llistes de partits que formen l'aplicació amb les classes Feed i FilteredFeed.

-login, separada d'accounts per seguretat, que juntament amb la classe Login permet a un usuari iniciar sessió a l'app

-matches, totes les funcions que tenen a veure amb partits, la classe Match per exemple permet crear, demanar, modificar o eliminar un partit; l'altre classe PlayersInMatch permet saber quins usuaris estan a dins del partit.

-userMatches, és un punt mig entre users i matches. Permet accedir als partits als que un usuari s'ha unit o ha creat amb les classes OwnedMatches i JoinedMatches

4.5.4 Endpoints

Cada classe dona accés a la base de dades, però és necessari tenir un sistema que ens permet accedir a les funcions. Per això, les APIs defineixen els endpoints: unes url relatives al servidor on s'executa l'aplicació i a les quals és possible enviar-li demandes HTTP i accedir, d'aquesta manera, a les dades.

A continuació, ensenyaré tots els endpoints definits en la meva app, explicant quin tipus de request és, la url, quina funció fan, com accedir-hi i quina resposta esperar.

Request:GET
Url: /account/<id>
Resum: Retorna tota la informació de l'usuari amb la id.

-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador de l'usuari

Resposta: content-type: application/json
-200, {"user": user.json()}
-404 {"message": f"Could not find a user with that id"}

—----------

Request:POST
Url: /account/
Resum: Crea un nou usuari a partir dels paràmetres enviats

-Autorització: cap

Paràmetre	Tipus	Descripció
username	string	Nom d'usuari únic
password	string	Contrasenya
email	string	Email de l'usuari
description	string	Informació que l'usuari vol donar d'ell mateix
is_admin	boolean	Si un usuari es administrador o no

Resposta: content-type: application/json
-200, {"user": user.json()}
-409 {"message": "An account with this username already exists!"}
-409 {"message": "An account with this email already exists!"}

Request:DELETE
Url: /account/<id>
Resum: Elimina l'usuari de la base de dades
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador de l'usuari

Resposta: content-type: application/json
-200,{"message": "Account deleted successfully!"}
-500 {"message": "An error occurred deleting the account."}
-400 {"message": "No id specified."}
-403 {"message": "You can't delete someone else's account."}
-404 {"message": "Could not find the account"}

Request:DELETE
Url: /userinfo/<id>
Resum: Retorna la informació de l'usuari en format petit
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador de l'usuari

Resposta: content-type: application/json
-200, {"user": user.smallInfo()}
-404, {"message": f"Could not find a user with that id"}

Request:DELETE
Url: /userid
Resum: Retorna la id de l'usuari que té la sessió iniciada.
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador de l'usuari

Resposta: content-type: application/json
-200, {"user_id": user.id}
-404, {"message": f"Could not find a user with that id"}
Request:DELETE

Url: /login
Resum: Inicia la sessió per l'usuari.
-Autorització: cap

Paràmetre	Tipus	Descripció
username	string	Nom d'usuari
password	string	Contrasenya hash

Resposta: content-type: application/json
-200, {"token": token}
-404, {"message": f"Login failed! No account was found with username"}
-404, {"message": "Invalid password!"}

Request:POST
Url: /match/<id>
Resum: Retorna informació del partit amb la id associada.
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
title	string	Títol del partit
description	string	Informació sobre el partit
location	string	Direcció on es juga el partit
city	string	Ciutat o poble on es juga el partit
date	string	Data i hora quan es juga el partit
numPlayers	int	Número de participants
sport	string	Tipus d'esport del partit
ongoing	boolean	Si el partit ja s'ha acabat o no

Resposta: content-type: application/json
-200, {"match": match.json()}
-500, {"message": "An error occurred creating the match."}

Request:GET
Url: /match/<id>
Resum: Retorna informació del partit amb la id associada.
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador del partit

Resposta: content-type: application/json
-200, {"match": match.json()}
-404, {"message": f"Could not find an a match with that id"}

Request:PUT
Url: /match/<id>
Resum: Marca el final del partit i apunta el resultat.
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador del partit
winner_id	int	Número identificador del guanyador

Resposta: content-type: application/json
-200,{"message": "Finished match id [{}]".format(data.winner_id)}
-404, {"message": "No match found with id [{}]".format(id)}
-500, {"message": "Match is already over"}

Request:DELETE
Url: /match/<id>
Resum: Marca el final del partit i apunta el resultat.
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador del partit

Resposta: content-type: application/json
-200, {"message": "Left match successfully!"}
-404, {"message": "No match found with id [{}]".format(id)}
-500, {"message": "Error leaving match"}, 500
-200, {"message": "Player not in match"}, 200

Request:GET
Url: /players/<id>
Resum: Retorna la informació limitada dels usuaris que estan apuntats al partit.
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador del partit

Resposta: content-type: application/json
-200, {"match_players": [player.smallInfo() for player in players]}
-404, {"message": f"Could not find an a match with that id"}

Request:POST
Url: /players/<id>
Resum: Apunta a l'usuari al partit demanat.
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador del partit

Resposta: content-type: application/json
-201, {"message": "Joined match"}
-404, {"message": "No match found with id [{}]".format(id)}
-500, {"message": "Game Full"}
-500, {"message": "Error Joining Game"}

Request:GET
Url: /owned/<id>
Resum: Retorna un llistat de tots els partits creats per un usuari.
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador de l'usuari

Resposta: content-type: application/json
-200, {"owned_matches": [match.json() for match in matches]}
-404, {"message": f"User has no owned matches"}
-404, {"message": f"User does not exist"}

Request:GET
Url: /matches/<id>
Resum: Retorna un llistat de tots els partits on l'usuari amb la id s'ha apuntat.
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador de l'usuari

Resposta: content-type: application/json
-200, {"owned_matches": [match.json() for match in matches]}
-404, {"message": f"User has no joined matches"}
-404, {"message": f"User does not exist"}

—----------

Request:GET
Url: /feed
Resum: Retorna un llistat dels partits que formen el feed de l'usuari
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador de l'usuari

Resposta: content-type: application/json
-200, {"feed_matches": [match.json() for match in matches]}
-404, {"message": f"Unable to get feed"}

—----------

Request:GET
Url: /filtered
Resum: Retorna un llistat dels partits que formen el feed de l'usuari segons l'esport indicat
-Autorització: Bearer <token>

Paràmetre	Tipus	Descripció
id	int	Número identificador de l'usuari
sport	string	Tipus d'esport dels partits

Resposta: content-type: application/json
-200, {"feed_matches": [match.json() for match in matches]}
-404, {"message": f"Unable to get feed"}

4.6 Recomanador

Dins del món de l'esport, hi ha una gran variació en el que és el nivell físic i d'experiència entre persona i persona. Això pot ser un problema, ja que, si es juga un partit amb una desigualtat de nivell notable el partit pot acabar sent una mala experiència per a tothom. És per això que volia crear un sistema per recomanar partits als usuaris.

L'objectiu és poder proposar a l'usuari una sèrie de partits que s'ajustin al seu nivell i que, d'aquesta manera, els usuaris puguin trobar persones amb les que compartir un partit igualat.

La meva primera idea era implementar un sistema de recomanació per buscar quins partits s'adeqüen millor a cada usuari. Pensant en com desenvolupar el recomanador vaig trobar que el que realment estava buscant eren usuaris semblants en el fet de que estiguin en un nivell similar, per tant l'ús de recomanació col.laborativa o basada en contingut no són el que estava buscant. Tot i que poden ser útils no donen la funció de comparar usuaris que necessito.
La segona idea era un sistema similar als escacs amb un elo pero la heurística per diferents esports no es la mateixi jo vull tenir un sistema més general.

Al final, el que vaig determinar que el que buscava era - realment - un sistema de ranking i el he acabat implementant es un sistema de pairwise ranking, que ens donarà una puntuació per cada usuari i és aquest valor el que faré servir per trobar els usuaris similars.

4.6.1 Pairwise ranking

La funció principal d'un pairwise ranking és que, donat un número "n" d'ítems (en el meu cas usuaris) i utilitzant comparacions entre parelles d'ítems, permet identificar els top "k" items, o donar un rànquing de tots els ítems. És la darrera funció la que utilitzo en aquest treball, ja que, em permet tenir una puntuació per a cada usuari basat en els resultats de les comparacions entre usuaris.
L'algorisme que faig servir està basat en el copland counting algorithm [2].
He escollit aquest algorisme per la seva simplicitat, ja que, ordena els ítems segons el nombre d'emparellaments guanyats i ajuda a tenir una complexitat computacional més reduïda comparat amb altres tipus de rankings. No obstant, , tot i ser simple no vol dir que no sigui eficaç en desenvolupar la seva tasca i és força robust a l'hora de donar resultats que siguin fiables ja que no es fan suposicions a l'hora de tractar les dades.

4.6.2 Copeland counting algorithm

Volem trobar el rànquing dins d'una col·lecció n d'ítems on $n \geq 2$, indexats en un set $[n] := \{1, ..., n\}$, en aquest cas la col·lecció és el grup d'usuaris, a partir del grup construïm una matriu $M \in R\, nxn$.

Sent *i, j* dos usuaris diferents, denotem Mij com la probabilitat de que l'ítem *i* guanyi la comparació amb l'ítem *j,* assumim que de cada comparació en surt un guanyador tal que:

$$Mij + Mij = 1$$

$Mii = 1/2$, definim la diagonal per corregir el problema de puntuacions neutres.

Per cada ítem que pertany a la col·lecció, $i \in [n]$, definim una puntuació τi com a:

$$\tau i := \frac{1}{n} \sum_{j=1}^{n} Mij$$

Per tant, la puntuació τi de qualsevol ítem $i \in [n]$ correspon a la probabilitat de que l'ítem *i* guanyi un ítem escollit aleatoriament del grup de *n* ítems.

Quan es juga un partit es genera una comparació δ.I com que cada parella pot jugar més d'una vegada és necessari crear un grup de comparacións $[r]$, tal que $\delta \in [r]$. Llavors, per cada $i, j \in [n]$ i cada $\delta \in [r]$ tenim $\Upsilon_{ij}^{\delta} \in \{-1, 0, +1\}$, representant la comparació δ entre i i j definida com:

$$\Upsilon_{ij}^{\delta} = \{0, si\ no\ hi\ ha\ enparellament\ en\ la\ comparació\ \delta,\ +1\ si\ i\ guanya\ a\ j,\ -1\ si\ j\ guanya\ a\ i\}$$

Seguint aquesta definició assegura que $\Upsilon_{ij}^{\delta} = -\Upsilon_{ji}^{\delta}$.

Un cop hem fet totes les comparacions només queda calcular el τi amb la fórmula prèviament explicada.

4.6.3 Implementació

L'algorisme està escrit en python, es comença amb la part principal es una matriu de zeros amb mida #usuaris x #usuaris. Un cop tenim la matriu inicialitzada, posem el valor de tots els elements de la diagonal amb un valor de ½. Llavors, cada vegada que es juga un partit, s'envia una senyal amb el guanyador de cada partit, el resultat és el que es considera la comparació i es processa segons la fórmula explicada en l'apartat anterior. Un cop omplerta matriu, el següent pas és el de calcular les puntuacions dels usuaris.

Cada usuari té la seva puntuació, i es pot calcular en qualsevol moment si es té la matriu. El problema en un deployment real on hi hagi molts usuaris pot ser que les puntuacions s'han de calcular totes de cop. Per això, la meva idea sería tenir un procés automàtic on cada día en una hora no molt concurrent es fes el càlcul de les puntuacions. Aquesta opció no afectaría de forma dramática el funcionament del recomanador, ja que, no és gaire probable que un mateix usuari jugui més d'un partit per día i, per tant, la recomanació no es veuría molt afectada amb la manca de resultats del mateix dia. Fent el càlcul només una vegada per día ens estalviem molt cost operacional si ho comparem a fer el càlcul cada vegada que s'actualitza la matriu amb una nova comparació.

Per recomanar un partit també necessitem saber quina és la puntuació del propi partit. Per a calcular-ho, fem la mitja dels valors de cada usuari que està apuntat. Amb els valors de cada partit llavors podem començar a trobar quin és el partit que s'ajusta més a l'usuari pel qual estem buscant el partit per recomanar. Es busca per cada partit fins trobar-ne un que coincideix amb la puntuació de l'usuari que està buscant un partit o amb una diferència de 0.5 punts. Si no en troba cap, es retorna el següent més proper. És aquest últim partit el que es recomana a l'usuari.

4.6.4 Simulació

Com que no tinc una base d'usuaris amb qui testejar el recomanador, he fet una simulació d'usuaris que juguen partits entre ells.

Començo amb un número de jugadors que será la quantitat d'usuaris a simular, i cadascun amb un nivell d'habilitat realista per a poder fer la simulació el més real posible. Genero primer una distribució de punts de l'u (1) al deu (10), de tal manera que hi hagi sis nivells (1, 2, 4, 6, 8, 10). A cada nivell d'aquests li assigno una probabilitat segons una distribució normal amb una mitjana de 5 (tot i que arrodoneixo a 4 o 6) i amb una desviació estàndard de 10. Amb la distribució vaig assignant de manera aleatòria els valors a cada usuari aconseguint que els valors en global de tots els usuaris segueixin una distribució més o menys realista.

Havent-los assignat un nivell als usuaris, començo a simular els partits. La funció per simular partits genera resultats per tots els usuaris: per cada usuari, n'escull un altre de manera aleatòria i genera un resultat. El resultat és creat a partir dels nivells dels usuaris també amb un element d'aleatorietat, però tenint en compte el pes del valor de cada usuari, si hi ha diferència de nivell l'usuari que té el valor més alt també té més probabilitats de guanyar.

Els resultats es generen un cop per cada usuari i per cada iteració. Es poden escollir el número d'iteracions passant-ho per paràmetre. La sortida de la simulació de partits és una llista de partits composta de tres valors la id dels dos participants i el guanyador, que ve a ser el mateix que trobem a l'aplicació quan acabem un partit.

Un cop tenim la llista feta només queda fer els passos que he explicat en l'apartat anterior de la implementació, en el cas de la simulació és diferent que dins de l'aplicació ja que no he de tractar amb la base de dades ni la API, sinó que es fa tot directament des del mateix script de la simulació.

5. Conclusions i treball futur

Amb aquest projecte he creat una aplicació funcional que compleix amb els quatre punts principals que em vaig marcar a l'inici del treball. La meva app permet a l'usuari crear un partit, compartir-lo amb altres usuaris, buscar partits que li interessa i unir-se i buscar partits on el nivell sigui similar al seu.

A partir d'aquí, si seguís amb el desenvolupament, el següent pas sería fer el deployment de la app a un servidor online per poder comprobar el funcionament amb usuaris reals. Per la part de codi, afegiria més contingut informatiu, com pot ser el suport per imatges i mapes, també donar més opcions als usuaris com un filtratge més específic per als partits i opcions per adaptar el seu perfil més al gust de l'usuari. L'altre part que ampliaria seria l'apartat social, és a dir, que usuaris puguin seguir a altres usuaris, invitacions a partits, etc.

Per la part del matchmaking he trobat que el problema més gran es el del cold-start, un problema que molts sistemes de recomanació es troben. Consisteix en la incertesa del valor real d'un Ítem, o nivell de l'usuari, quan aquest s'introdueix inicialment. En el meu cas se li assigna un valor mitjà a cada usuari nou però aquest ha de jugar partits per tal saber realment a quin nivell està la seva habilitat.

He pensat en dues possibles solucions. La primera i possiblement més fàcil, és demanar a l'usuari que es puntuï a si mateix al crear el compte o dins del seu perfil, això ens permet començar amb un número possiblement més pròxim que el valor mitjà, però també vol dir que l'usuari s'està autoavaluant i a vegades un no és el millor jutge per ell mateix. L'altre idea és crear un sistema d'avaluació entre usuaris que complementi l'emparellament, si un usuari pot votar el nivell dels seus companys o contrincants tenim variables amb les que comparar a l'hora de calcular el nivell. És bastant probable que les dues solucions es puguin implementar simultàniament.

Hi ha molts fils a estirar per seguir el desenvolupament de la app, pero jo crec que les millores que he exposat en els paràgrafs anteriors són el camí més indicat per la meva aplicació.

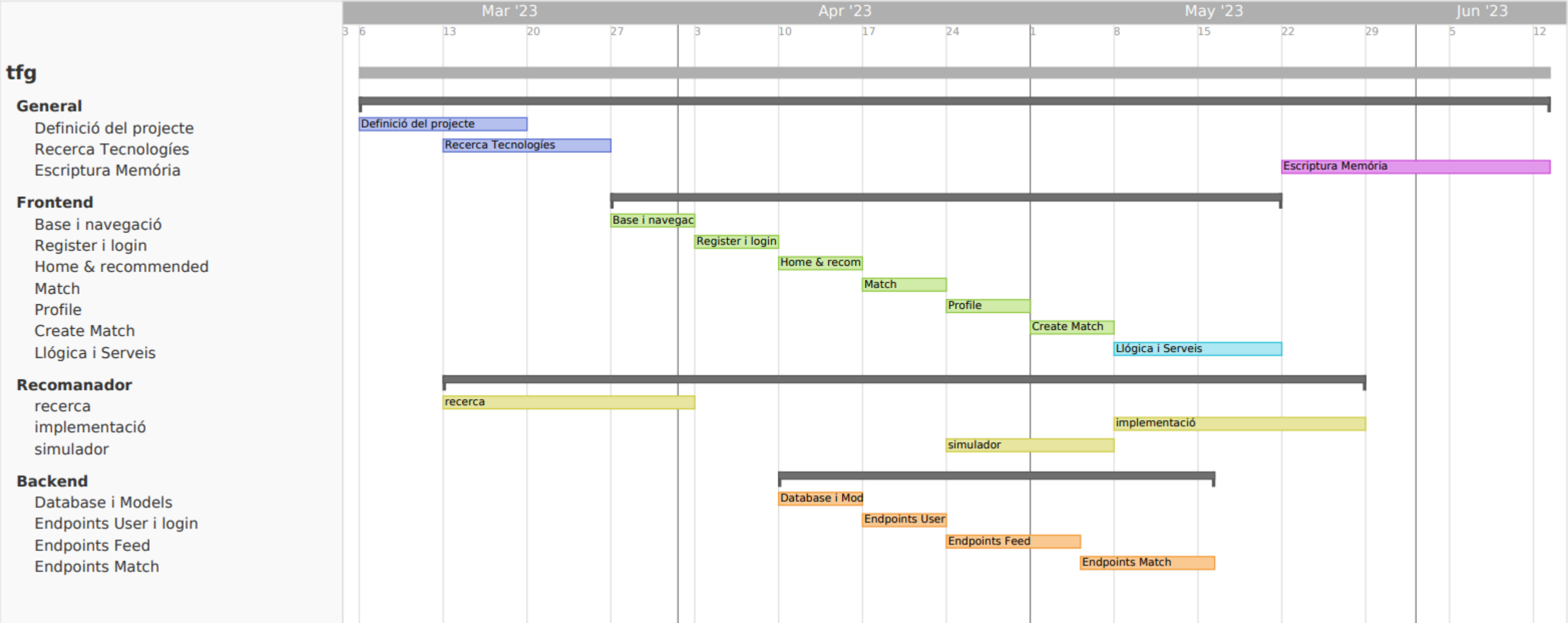
6. Referències

[1] **Global status report on physical activity 2022 -** https://www.who.int/publications/i/item/9789240059153

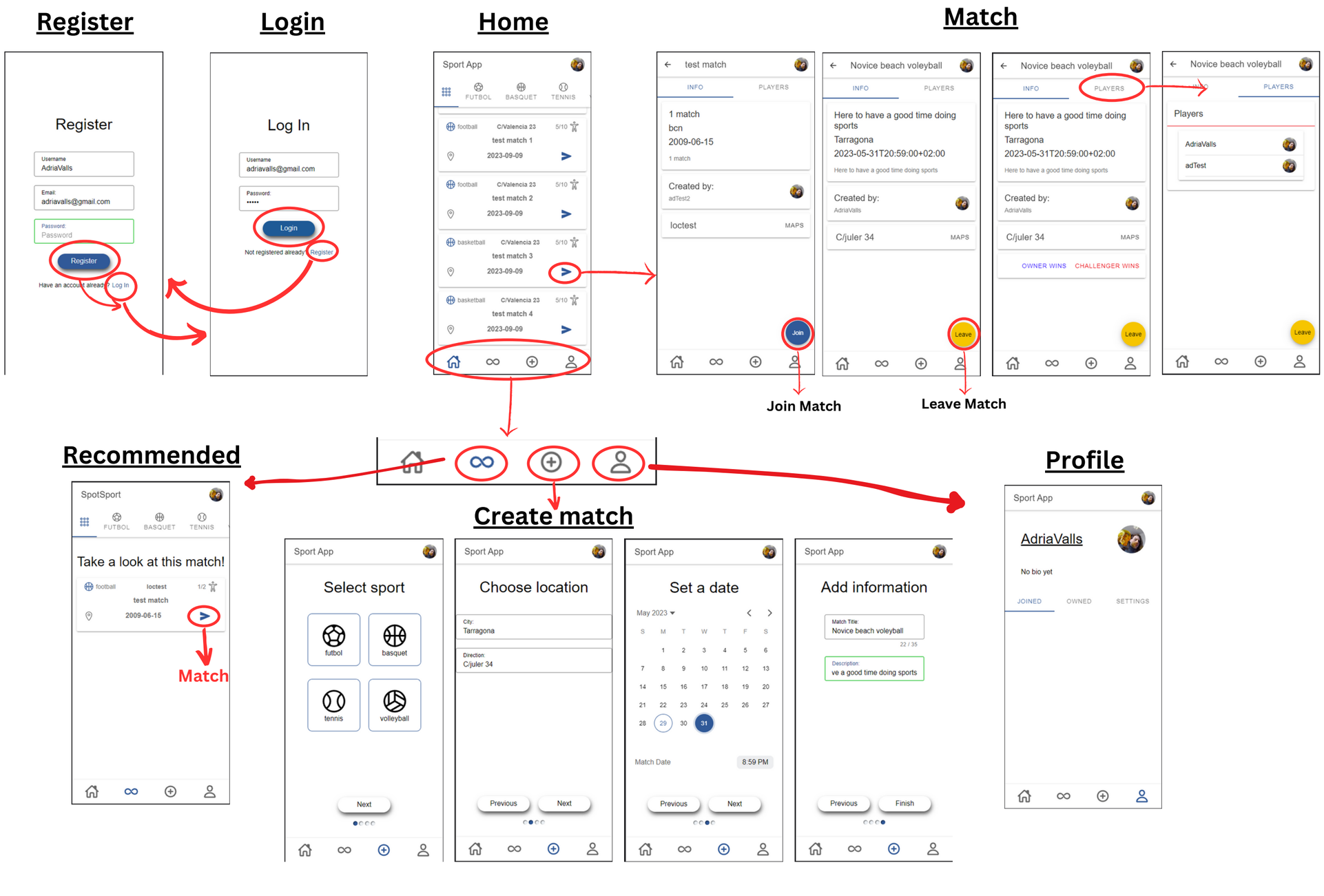
[2] **Simple, Robust and Optimal Ranking from Pairwise Comparisons** [Nihar B. Shah, Martin J. Wainwright]

7. Annexos

Annex 1. Diagrama de Gantt


Mar '23
Apr '23
May '23
Jun '23
3
6
13
20
27
3
10
17
24
1
8
15
22
29
5
12
tfg
General
Definició del projecte
Recerca Tecnologíes
Escriptura Memória
Frontend
Base i navegació
Register i login
Home & recommended
Match
Profile
Create Match
Llógica i Serveis
Recomanador
recerca
implementació
simulador
Backend
Database i Models
Endpoints User i login
Endpoints Feed
Endpoints Match
Definició del projecte
Recerca Tecnologíes
Escriptura Memória
Base i navegac
Register i login
Home & recom
Match
Profile
Create Match
Llógica i Serveis
recerca
implementació
simulador
Database i Mod
Endpoints User
Endpoints Feed
Endpoints Match

Annex 2. Diagrama de navegació


Register
Login
Home
Match
Register
Username
AdriaValls
Email:
adriavalls@gmail.com
Password:
Password
Register
Have an account already? Log In
Log In
Username
adriavalls@gmail.com
Password:
Login
Not registered already? Register
Sport App
FUTBOL
BASQUET
TENNIS
football C/Valencia 23 5/10
test match 1
2023-09-09
football C/Valencia 23 5/10
test match 2
2023-09-09
basketball C/Valencia 23 5/10
test match 3
2023-09-09
basketball C/Valencia 23 5/10
test match 4
2023-09-09
test match
INFO
PLAYERS
1 match
bcn
2009-06-15
1 match
Created by:
adTest2
loctest
MAPS
Join
Novice beach voleyball
INFO
PLAYERS
Here to have a good time doing sports
Tarragona
2023-05-31T20:59:00+02:00
Here to have a good time doing sports
Created by:
AdriaValls
C/juler 34
MAPS
Leave
OWNER WINS
CHALLENGER WINS
Players
AdriaValls
adTest
Join Match
Leave Match
Recommended
Create match
Profile
SpotSport
FUTBOL
BASQUET
TENNIS
Take a look at this match!
football loctest 1/2
test match
2009-06-15
Match
Sport App
Select sport
futbol
basquet
tennis
volleyball
Next
Choose location
City:
Tarragona
Direction:
C/juler 34
Previous
Next
Set a date
May 2023
S M T W T F S
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
Match Date
8:59 PM
Previous
Next
Add information
Match Title:
Novice beach voleyball
22 / 35
Description:
ve a good time doing sports
Previous
Finish
AdriaValls
No bio yet
JOINED
OWNED
SETTINGS