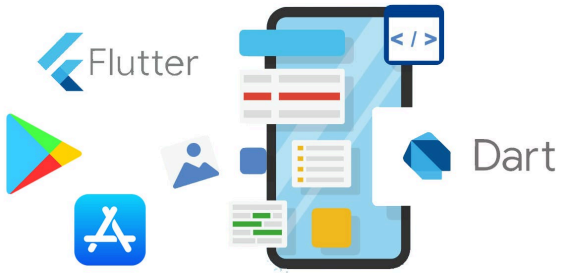
Treball de recerca

DESENVOLUPAMENT D'UNA APLICACIÓ MULTIPLATAFORMA

Cranium, un joc de taula per a mòbil



Ferran Torres

Cornellà de Llobregat, 31 d'octubre del 2023

Abstract

In line with the interest in board games and mobile application development, the idea was to adapt a board game to current technologies.

To develop the application, several technologies have been used, including Flutter, a software development kit created by Google, which is used to create applications for mobile, web and desktop from the same code base, written in the Dart programming language.

With these tools, an application has been created that adapts a board game, with the board, the cards with the questions and the dynamics of the game, adapted to the mobile screen. In this way, it has been proved that Flutter is a useful tool when developing multiplatform applications.

Conforme amb l'interès pels jocs de taula i el desenvolupament d'aplicacions mòbil, s'ha plantejat adaptar un joc de taula a les tecnologies de l'actualitat.

Per desenvolupar l'aplicació, s'ha utilitzat diverses tecnologies entre les quals es troba Flutter, un kit de desenvolupament de programari creat per Google, que es fa servir per crear aplicacions per a mòbil, web i escriptori des d'una mateixa base de codi, escrit en el llenguatge de programació Dart.

Amb aquestes eines, s'ha creat una aplicació que adapta un joc de taula, amb el taulell, les cartes amb les preguntes i les dinàmiques del joc, adequades a la pantalla mòbil. D'aquesta manera, s'ha comprovat que Flutter és una eina útil a l'hora de desenvolupar aplicacions multiplataforma.

ÍNDEX

1. Introducció

1.1 La idea del treball

La idea del treball és desenvolupar una aplicació mòbil per a Android que adapti un joc de taula a les tecnologies de l'actualitat. Per això, utilitzaré Flutter, una "eina" creada per Google que permet desenvolupar aplicacions multiplataforma amb la mateixa base de codi. D'aquesta manera, encara que la idea principal sigui llançar el joc per a Android, gràcies a aquesta llibreria en un futur podria adaptar-la per a altres plataformes com el sistema operatiu iOS, navegadors web o aplicacions d'escriptori (per a Windows, Linux o Mac).

El joc de taula en el qual em vull basar per a crear una aplicació es diu *Cranium*. Aquest té quatre categories de proves:

- preguntes de cultura general
- proves que consisteixen a dibuixar o fer escultures amb plastilina
- proves d'imitar o cantar
- proves que consisteixen en anagrames, jocs de paraules i definicions

La idea és que l'aplicació sigui el suport del joc, és a dir, com el tauler o les cartes de preguntes d'un joc de taula tradicional. Perquè, tot i que es necessiti un mòbil, per a jugar és necessari fer-ho amb més persones. És a dir, és important la companyia (en persona), ja que si no, no es pot jugar. D'aquesta manera, es fomenta la interacció social, aspecte molt necessari des del meu punt de vista als jocs de taula o als videojocs.

1.2 La motivació

Quant a la motivació, em fa especial il·lusió adaptar el *Cranium*, ja que és un joc de taula que m'agrada molt i em sembla molt divertit. A més, el fet de poder crear una aplicació que pugui portar a qualsevol lloc en el meu mòbil i, així, poder jugar sense necessitat de portar tota la caixa amb el tauler, les preguntes i el material necessari, també és un aspecte que em motiva.

D'altra banda, un dels motius principals de fer aquest treball és aprendre sobre aquestes tecnologies, pel fet que en un futur una de les meves possibilitats d'estudi passa per la informàtica i, poder comprovar si m'agrada programar, és un pas important a l'hora de decidir definitivament el meu futur pel que fa als estudis.

2. Marc teòric

El desenvolupament d'aplicacions mòbils és el procés de crear programari per a telèfons intel·ligents, tauletes i assistents digitals, més comunament per als sistemes operatius Android i iOS. El programari es pot preinstal·lar al dispositiu, descarregar des d'una botiga d'aplicacions mòbils o accedir-hi a través d'un navegador web mòbil. Els llenguatges de programació i de marcatge utilitzats per a aquest tipus de desenvolupament de programari inclouen Java, Swift, C# i HTML5.

2.1 Android i iOS

Android és un sistema operatiu de codi obert[1] basat en Linux inicialment dissenyat per a telèfons mòbils. Va ser desenvolupat per Android Inc. en els seus inicis (2003) i més tard per Google (Open Handset Alliance[2]) després que la multinacional adquirís la petita companyia en 2005. En l'actualitat, trobem aquest sistema operatiu en 2500 milions de dispositius actius com a telèfons intel·ligents, tauletes, televisors intel·ligents, dispositius portàtils i altres dispositius electrònics.

fig 2.1 Logo d'Android

La plataforma de programació d'Android inclou el kit de desenvolupament de programari (SDK) d'Android, que conté eines i recursos, com una àmplia gamma de biblioteques, per al desenvolupament d'aplicacions per a Android. El SDK d'Android inclou un entorn de desenvolupament integrat (IDE)[3] anomenat Android Studio, que és la principal eina utilitzada per a desenvolupar aplicacions per a Android.

[1] **Codi obert:** el codi font està disponible públicament i de manera gratuïta per a tothom. Qualsevol persona o empresa pot accedir al codi font, modificar-lo i distribuir-lo en les seves pròpies versions del sistema operatiu.
[2] **Open Handset Alliance:** consorci de 84 marques per desenvolupar estàndards oberts per dispositius mòbils (telèfons, tauletes, etc.)
[3] **IDE:** entorn integrat de desenvolupament o IDE, és una eina informàtica per al desenvolupament de programari de manera còmoda i ràpida.

Java i Kotlin són els llenguatges de programació principals fets servir per al desenvolupament d'aplicacions per a Android. Per a desenvolupar aplicacions per a Android s'utilitza, principalment, Android Studio. Aquest és l'IDE que serveix per a crear la interfície d'usuari de l'aplicació i escriure el codi necessari per a la lògica de l'aplicació i per a accedir als recursos del sistema, com la càmera, l'emmagatzematge intern i la connexió a Internet. Una vegada que s'ha desenvolupat una aplicació per a Android, es pot provar fent ús d'un emulador d'Android o un dispositiu físic. Finalment, si es desitja distribuir l'aplicació, es pot publicar en la Play Store de Google, que és la botiga d'aplicacions d'Android.

fig 2.2 Logo d'iOS

D'altra banda, està iOS. iOS és el sistema operatiu desenvolupat per Apple per als seus dispositius mòbils, com l'iPhone, l'iPad i l'iPod Touch. És un sistema operatiu tancat i exclusiu d'Apple, cosa que significa que només es pot executar en dispositius fabricats per Apple.

Swift o Objective-C són els llenguatges de programació per al desenvolupament d'aplicacions per a iOS. Per a desenvolupar una aplicació per a iOS, primer es necessita un sistema Mac[4]. A més es necessita una eina de desenvolupament anomenada Xcode, que és l'IDE oficial d'Apple. En Xcode, es pot dissenyar la interfície d'usuari de l'aplicació, així com escriure el codi de l'aplicació. Una vegada que s'ha desenvolupat l'aplicació, es pot provar en un simulador d'iOS. Finalment, si es desitja distribuir l'aplicació en l'App Store d'Apple, s'ha de crear un compte de desenvolupador d'Apple i seguir les pautes i requisits de l'App Store.

Android i iOS són, actualment, els dos sistemes operatius més utilitzats en el món. I és que segons les dades actualitzades d'abril de 2023 de StatCounter, Android té una quota de mercat del 68,61% i iOS del 30,61%, la qual cosa suposa que la resta de sistemes (Samsung, KaiOS, Windows...) tenen, entre tots ells, només un 0,78%.[5]

[4] **Mac:** sistema operatiu creat per Apple per a la seva línia de computadores Macintosh
[5] ***Mobile Operating system market share worldwide | StatCounter Global Stats*. (s. f.). StatCounter Global Stats.** https://gs.statcounter.com/os-market-share/mobile/worldwide

2.2 Desenvolupament multiplataforma

Com hem vist en el punt anterior, cada sistema operatiu té els seus llenguatges de programació per a desenvolupar les seves aplicacions. Llavors, caldria aprendre's diversos llenguatges per a crear aplicacions per a diferents sistemes. Aquest inconvenient es podria resoldre amb el desenvolupament multiplataforma.

El desenvolupament multiplataforma és la creació d'aplicacions que es poden executar en diverses plataformes diferents (com Android, iOS, Windows, Linux...) utilitzant un mateix codi font. Això permet als desenvolupadors crear aplicacions per a múltiples plataformes de manera més eficient, ja que no és necessari escriure codi específic per a cada plataforma ni aprendre els diferents llenguatges de programació per a cada sistema o plataforma.

Hi ha diverses eines i tecnologies disponibles per al desenvolupament multiplataforma: entre elles es troba Flutter, la que jo utilitzaré, un SDK de desenvolupament d'aplicacions mòbils multiplataforma de codi obert desenvolupat per Google. Flutter fa servir el llenguatge de programació Dart i proporciona una àmplia gamma d'eines i ginys per al desenvolupament d'aplicacions per a iOS, Android, navegador web i aplicacions d’escriptori. Altres exemples podrien ser Xamarin que empra el llenguatge de programació C# i .NET Framework o React Native que és una biblioteca de JavaScript.

El desenvolupament multiplataforma pot ser beneficiós, en part per als desenvolupadors i les empreses, ja que els permet arribar a més usuaris amb menys esforç i cost, gràcies a no haver de tornar a escriure el codi específic per a cada plataforma. No obstant això, també pot tenir alguns inconvenients i limitacions, com la possible falta d'accés a unes certes característiques específiques de cada plataforma i un rendiment lleugerament inferior en comparació amb les aplicacions natives.

2.3 Bases de dades

Una base de dades és una recopilació organitzada d'informació o dades, normalment emmagatzemada de manera electrònica en un sistema informàtic, de manera estructurada i accessible per al seu posterior ús, cerca i manipulació. Les bases de dades, generalment, estan controlades per un sistema de gestió de bases de dades (DataBase Management System (DBMS)) i s'utilitzen per a emmagatzemar i gestionar grans quantitats d'informació, permetent a les persones i aplicacions accedir a les dades de manera eficient i precisa. Aquesta informació emmagatzemada pot tractar-se de dades sobre persones, productes, comandes o altres coses.

2.3.1 Bases de dades relacionals

Les bases de dades relacionals són un tipus de base de dades que aconsegueixen emmagatzemar i proporcionar accés a punts de dades relacionades entre si en taules.

Les taules estan compostes per files i columnes. Cada fila d'una taula representa un registre que conté una sèrie d'atributs, que són les columnes. Quan es defineix una columna s'ha d'indicar el tipus de dades que contindrà. Per exemple, es pot tenir una columna "Nom" que emmagatzema dades de tipus text o una altra anomenada "Edat", on les dades seran un nombre sencer. També es pot indicar si el valor de la columna pot ser null (buit) o no. A més, cada taula ha de tenir una columna que actua com a identificador inequívoc d'aquella fila. Aquesta columna es coneix com a clau primària.

Per a relacionar les taules entre si, es defineixen columnes que actuen com a claus foranes. D'aquesta manera, una columna d'una taula fa referència a una columna d'una altra taula.

En la següent imatge es mostra un exemple per il·lustrar aquest concepte de base de dades relacional:

fig 2.3 Exemple de base de dades relacional

Es tenen dues taules, una es diu "Clientes" i l'altra "Pedidos". La taula "Clientes" té una sèrie de columnes definides com a "Nombre", "Dirección", "Teléfono"... i "Identif/C.". Aquesta taula té tres files on cadascuna d'elles representa a un client i les seves dades. La columna "Identif/C." actua com a clau primària o identificador. D'altra banda, la taula "Pedidos" té les seves respectives files i columnes amb les seves dades i, en aquest cas, la columna "Identif/C." està definida com a clau forana. D'aquesta manera podem relacionar ambdues taules i saber les comandes que ha fet cada client.

Aquest model de base de dades va ser dissenyat per a resoldre el problema causat per estructures de dades múltiples. En els inicis de les bases de dades, cada aplicació emmagatzemava dades en la seva pròpia estructura única. A l'hora de crear aplicacions usant aquestes dades, els desenvolupadors havien de conèixer bé l'estructura concreta per a poder trobar la informació que necessitaven. Aquestes estructures de dades eren poc eficaces i el manteniment era complicat. El model relacional va proporcionar una forma estàndard, intuïtiva i eficient de representar i consultar dades que podria utilitzar qualsevol aplicació.

fig 2.4 Logo d'SQL

El llenguatge de consulta estructurada (SQL, en anglès, Structured Query Language) és un llenguatge de programació per a emmagatzemar

i processar informació dins d'aquest model de base de dades. Es basa en l'àlgebra relacional i proporciona un llenguatge matemàtic d'uniformitat interna que facilita la millora del rendiment de totes les consultes en bases de dades. Es pot usar les instruccions SQL per a emmagatzemar, actualitzar, eliminar, buscar i recuperar informació de la base de dades i també per a mantenir i optimitzar el seu rendiment.

Les bases de dades relacionals, llavors, s'utilitzen per a processar transaccions de comerç electrònic, administrar quantitats enormes i essencials d'informació de clients, rastrejar inventaris... Aquest tipus de bases de dades es pot fer servir per a qualsevol aplicació de dades en la qual els punts de dades es relacionin entre si i hagin de gestionar-se de manera segura, conforme a normes i d'una manera uniforme.

2.3.2 Bases de dades NoSQL

Les bases de dades NoSQL ('Not only SQL') són un plantejament de disseny de bases de dades que permeten emmagatzemar i consultar dades d'una forma diferent de les estructures tradicionals relacionals.

Existeixen diversos tipus de bases de dades NoSQL, dissenyades específicament per a models de dades concretes, que proporcionen alternatives per a organitzar les dades. En oferir aquestes diverses estructures, NoSQL es pot aplicar a la gestió del big data, les xarxes socials, aplicacions web en temps real, a l'anàlisi de dades... Entre aquests models es troben:

Magatzem de parells clau-valor: considerada la forma més simple de bases NoSQL, aquest model de dades s'organitza en un diccionari de parells clau-valor, on cada element té una clau, que serveix com un identificador únic, i un valor. S'utilitza normalment per a emmagatzemar en la memòria cau i guardar informació de sessió de l'usuari (perfil de l'usuari, missatges, recomanacions, carret de la compra...).

fig 2.5 Esquema magatzem de parells clau-valor


fig 2.6 Esquema magatzem de documents

Magatzem de documents: aquest model està dissenyat per a emmagatzemar i consultar dades com a documents, usualment de tipus JSON[6]. Funciona bé en casos com a catàlegs, perfils d'usuari i sistemes d'administració de contingut en els quals cada document és únic i evoluciona amb el temps.

Magatzem de grafs: aquest tipus de base de dades normalment alberga dades d'un graf de coneixement. Els elements, és a dir, les dades, s'emmagatzemen com a nodes, arestes i propietats. Qualsevol objecte, ubicació o persona pot ser un node. Una aresta defineix la relació entre els nodes. Això permet una representació de les dades més rica i completa. Les bases de dades de grafs s'utilitzen per a emmagatzemar i gestionar una xarxa de connexions entre elements dins del graf, com en les xarxes socials o sistemes de reserves.


fig 2.7 Esquema magatzem de grafs

Cada tipus de base de dades NoSQL presenta qualitats per a casos d'ús específics. No obstant això, totes comparteixen una gran flexibilitat que aconsegueix abordar grans volums de dades que canvien ràpidament, la qual cosa les fa ideals per a un desenvolupament àgil, a més de comptar amb un alt rendiment.

[6] **JSON** (JavaScript Object Notation): és un format de text que forma part del sistema de JavaScript.

2.4 Disseny d'interfícies

La UI (User interface en anglès o Interfície d'usuari) és un mitjà de comunicació entre un sistema i un usuari que permet als usuaris interactuar amb ell i dur a terme tasques. En altres paraules, la interfície és el lloc on l'usuari i el sistema es troben i es comuniquen de manera efectiva. La interfície d'usuari inclou elements com a botons, menús, icones, text, imatges i qualsevol altra cosa que l'usuari pugui veure o tocar en la pantalla del seu dispositiu mòbil.

La UI, llavors, és molt important, ja que és un dels elements principals de l'experiència d'usuari o User Experience (UX). En molt bona mesura, que una persona quedi satisfeta amb la seva experiència en una app o utilitzant un programa informàtic depèn de com d'útil, fàcil d'usar, intuïtiva i estètica resulta la interfície.

Per tant, perquè una UI resulti atractiva hi ha algunes característiques o conceptes a tenir en compte:

- Disseny visual: El disseny visual de la UI és important perquè pot influir en la percepció de l'usuari sobre la qualitat de l'aplicació.
- Disseny d'interacció: l'aplicació ha de ser fàcil d'usar perquè els usuaris puguin fer les seves tasques de manera eficient. Això inclou la disposició dels botons i els menús, la navegació, la retroalimentació i els efectes visuals.
- Proves d'usabilitat: És important provar la interfície d'usuari amb els usuaris per a identificar problemes i fer millores. Això pot fer-se mitjançant proves d'usabilitat i retroalimentació dels usuaris.
- Flexibilitat: l'usuari pot desfer fàcilment les accions en la interfície i que existeix tolerància als errors. Dit d'una altra manera, s'evita que una equivocació de l'usuari el porti a un "atzucac o carrer sense sortida".

A l'hora de desenvolupar aplicacions mòbils, existeixen diverses eines per al disseny d'interfícies d'usuari. Entre elles es troben:

- Adobe XD: una eina potent que permet als dissenyadors fer un prototip de les seves idees de manera ràpida i també la creació de dissenys interactius.

- Sketch: una altra eina de disseny de UI popular i fàcil d'usar, utilitzada per molts dissenyadors. Ofereix moltes opcions, com a eines d'estils de text i maneres de fusió, que et permeten crear interfícies d'usuari de manera ràpida i eficient. A més, compta amb funcions de sincronització en el núvol per a assegurar la seguretat del teu treball, independentment d'on et trobis.

- Figma: també és una eina de disseny d'interfície d'usuari popular i potent. Ofereix una àmplia gamma de característiques, incloent-hi l'edició col·laborativa i la capacitat de crear prototips, la qual cosa fa que el procés de disseny sigui més eficient. També compta amb una biblioteca UI que permet als dissenyadors emmagatzemar i accedir als recursos fàcilment.

Per a projectes d'equips petits (els quals no poden permetre's molt temps per a fer prototips) aquestes eines tenen un inconvenient: són purament de disseny. En general, no permeten exportar els dissenys a codi de la plataforma en la qual s'està desenvolupant. Per tant, una vegada els dissenyadors han acabat el disseny de l'aplicació, el passen als desenvolupadors, que són els que implementen, en la mesura que sigui possible, el disseny utilitzant el codi i les eines que cada plataforma posa a la seva disposició.

En el meu cas, pel fet que solament treballaré jo en l'aplicació i no disposo de molt de temps per a treballar en ella, no utilitzaré aquest tipus d'eines de creació de prototips, sinó que el disseny l'implementaré directament en Flutter.

3. Àrea d'estudi

3.1 Flutter

Flutter és un kit de desenvolupament de programari (SDK) d'IU de codi obert creat per Google[7]. La primera versió de Flutter va ser coneguda com a "Sky" i s'executava en el sistema operatiu Android. Es va donar a conèixer en la 2015 Dart Developer Summit, amb la

fig 3.1 Logo de Flutter

[7] ***Flutter - crea hermosas aplicaciones nativas en tiempo récord.*** (s. f.). https://esflutter.dev/

intenció de poder renderitzar consistentment a 120 quadres per segon[8]. El maig de 2017 Flutter va ser presentat en la conferència Google I/O, però no va ser fins al 4 de desembre de 2018 on es va llançar la primera versió estable de Flutter, Flutter 1.0 en l'esdeveniment Flutter Live a Londres[9]. Aquesta SDK és utilitzada per a desenvolupar aplicacions multiplataforma per a Android, iOS, Linux MacOS, Windows, web... a partir d'un únic codi base. Com que no s'ha de crear el codi específic per a cada plataforma, s'accelera el procés de desenvolupament de les aplicacions i es redueix el cost.

fig 3.2 Exemples de les plataformes a les quals Flutter pot desenvolupar aplicacions

Flutter està construït amb C, C++, Dart, Skia (un motor de renderitzat 2D) i Impeller (el motor de renderitzat per defecte en iOS).

Estructura

Flutter està dissenyat com un sistema extensible per capes. Existeix com una sèrie de biblioteques independents que depenen de la capa subjacent.

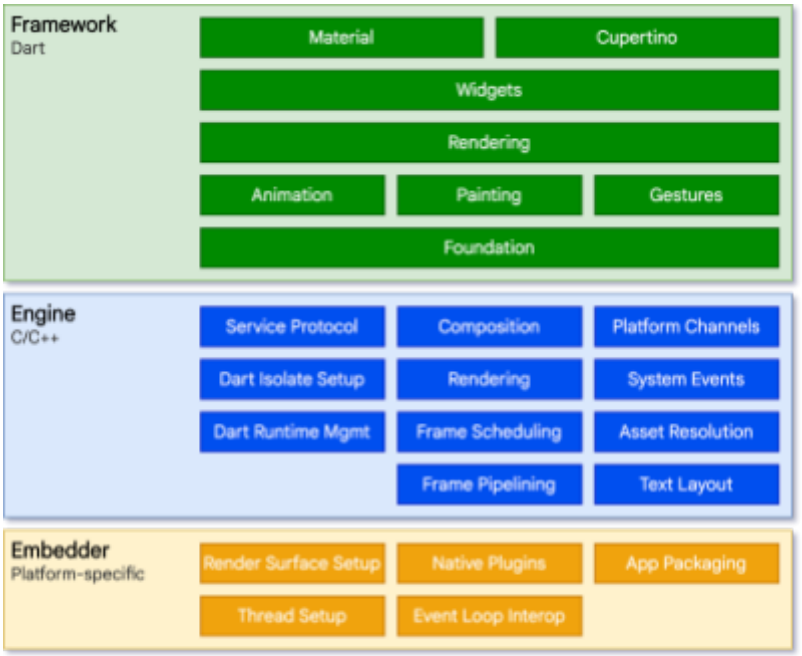


fig 3.3 Estructura de Flutter

[8] **Amadeo, R. (2015, 1 mayo).** ***Google's Dart language on Android aims for Java-free, 120 FPS apps*****. Ars Technica.** https://arstechnica.com/gadgets/2015/05/googles-dart-language-on-android-aims-for-java-free-120-fps-apps/

[9] **De Sonora | El Imparcial, N. (2019, 24 abril).** ***Google anunció el primer lanzamiento estable de Flutter 1.0. Noticias de Sonora | EL IMPARCIAL.*** https://www.elimparcial.com/sonora/tecnologia/Google-anuncio-el-primer-lanzamiento-estable-de-Flutter-1.0-20181209-0062.html

Per al sistema operatiu subjacent, les aplicacions de Flutter s'empaqueten de la mateixa manera que qualsevol altra aplicació nativa. És a dir, si el sistema operatiu fos Android, tant en les aplicacions creades amb Flutter com les nadiues d'Android s'empaquetarien com una APK[10].

Un ‘embedder’ específic de la plataforma es coordina amb el sistema operatiu subjacent per a accedir a serveis com a superfícies de renderitzat, accessibilitat i entrada… El ‘embedder’ està escrit en un llenguatge apropiat per a la plataforma: actualment Java i C++ per a Android, Objective-C/Objective-C++ per a iOS i macOS, i C++ per a Windows i Linux.

En el nucli de Flutter es troba el Flutter ‘Engine’, que està escrit en la seva major part en C++ i suporta les bases necessàries per a suportar totes les aplicacions Flutter. El motor s'encarrega de rasterizar les escenes compostes cada vegada que és necessari pintar un nou fotograma.[11]

Widgets

Flutter utilitza els widgets com a unitat de composició. Els widgets són els blocs de construcció de la interfície d'usuari d'una aplicació Flutter, i cada widget és una declaració immutable de part de la interfície d'usuari[12].

Els widgets formen una jerarquia, anomenada sovint ‘Widget tree’ basada en la composició. Cada widget es nia dins del seu pare i pot rebre context del pare. Aquesta estructura arriba fins al widget arrel (el contenidor que allotja l'aplicació Flutter, normalment MaterialApp per a Android o CupertinoApp per iOS).

Les aplicacions actualitzen la seva interfície d'usuari en resposta a esdeveniments (com una interacció de l'usuari) indicant al framework que substitueixi un widget de

[10] **Android Package Kit**: és un format per a paquets utilitzat pel sistema operatiu Android per a la distribució i instal·lació d'aplicacions mòbils i programari intermediari.

[11] ***Flutter Architectural Overview*. (s. f.). Flutter.** https://docs.flutter.dev/resources/architectural-overview#:~:text=Flutter%20is%20designed%20as%20an,to%20be%20optional%20and%20replaceable.

[12] ***Flutter Architectural Overview*. (s. f.-b). Flutter**. https://docs.flutter.dev/resources/architectural-overview#widgets

la jerarquia per un altre giny. A continuació, el framework compara els widgets nou i antic i actualitza de manera eficaç la interfície d'usuari.

Entre la gran quantitat de widgets que existeixen destaquen els següents:

- Text(): s'utilitza per escriure text a la pantalla. Dintre d'aquest widget tens possibilitats per customitzar-lo, amb per exemple la propietat style i, dintre d'style, un altre widget anomenat Text Style. Amb aquest últim es pot cambiar el color de fons, el color de la lletra, la font…

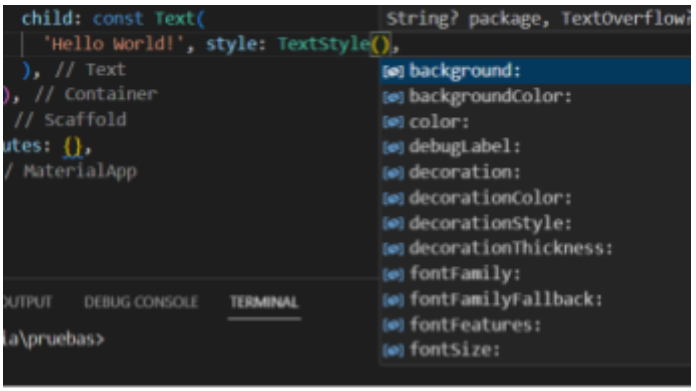


fig 3.4 Extret de la programació del widget Text( )

- Column () : serveix per disposar una llista de widgets verticalment, en aquest exemple podem veure que dins de la Columna tenim un text i a sota un filera amb més widgets.

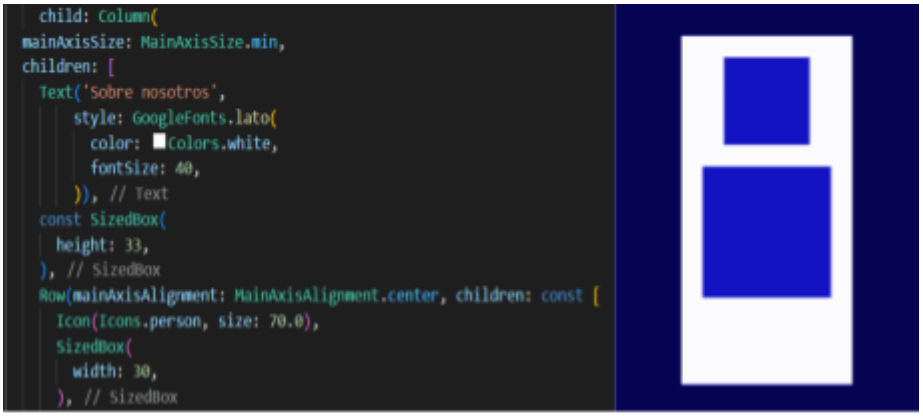


fig 3.5 Extret de la programació del widget Column ( ) i un esquema de la seva aparença

- ElevatedButton(): disposa un botó a la pantalla. Hi existeixen més widgets que creen botons , però l'Elevated Button té una estètica que el converteix en un dels més utilitzats.

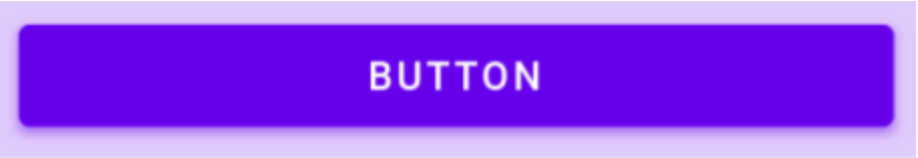


fig 3.6 Exemple d'un Elevated Button

- Icon(): crea una icona. Flutter té una col·lecció o catàleg d'icones predefinides que es poden utilitzar.

fig 3.7 Exemple d'un Icon

- Row (): Com fa el widget Column(), la Row serveix per disposar una llista de widgets, però aquesta vegada, horitzontalment.

fig 3.8 Esquema de l'aparença d'una Row ( )

3.2 Dart

Dart és un llenguatge de codi obert desenvolupat per Google en 2011 amb l'objectiu de fer el procés de desenvolupament el més còmode i ràpid possible per als desenvolupadors.

fig 3.9 Logo de Dart

Amb la finalitat d'oferir un llenguatge de programació més productiu per al desenvolupament multiplataforma, especialitzat al voltant de les necessitats de creació d'interfícies d'usuari, Dart compta amb un conjunt considerablement extens d'eines integrades, com el seu propi gestor de paquets, diversos compiladors i un analitzador. A més, disposa d'una eina molt útil a l'hora de desenvolupar aplicacions, el "hot reload" o recàrrega en calenta, que permet veure el resultat a l'instant en la teva aplicació en execució. Una vegada en producció, el codi es pot compilar en llenguatge natiu, per la qual cosa no és necessari un entorn especial per a executar-lo. En cas que es faci desenvolupament web, Dart es compila a JavaScript.

fig 3.10 Dart i Flutter

Dart constitueix la base de Flutter, que ho utilitza a causa de les qualitats d'aquest. Flutter necessita un llenguatge de programació que funcioni en diverses plataformes i ofereixi una experiència de desenvolupament ràpida, i Dart compleix amb aquests requisits gràcies, en part, al "hot reload" i a la seva capacitat de ràpida compilació, permetent iniciar aplicacions ràpidament en diferents plataformes.

3.3 SQLite

SQLite és una llibreria que implementa una base de dades SQL. Al contrari que moltes altres implementacions SQL, no requereix tenir un procés que actuï com a servidor, en canvi, SQLite llegeix i escriu directament sobre un sol arxiu emmagatzemat en disc. A més, requereix molt poca configuració i admet transaccions. L'arxiu que conté la base de dades és multiplataforma.

SQLite no ha de comparar-se amb altres bases de dades SQL com MySQL, Oracle, PostgreSQL..., ja que estan tractant de resoldre un problema diferent. Aquestes, se centren a proveir una base de dades compartida entre múltiples aplicacions, persones, departaments, etc. Per tant, se centren més en conceptes com a escalabilitat, concurrència, centralització i control d'accés (permisos). D'altra banda, SQLite tracta de proveir un magatzem de dades local per a aplicacions i/o dispositius individuals. Per això, són més importants conceptes com a economia, eficiència, fiabilitat, independència i simplicitat[13].

El fet que una base de dades SQLite no requereixi administració fa que sigui apropiada per a dispositius que operen sense el suport d'experts en la matèria. SQLite, per tant, és molt útil en casos d'ús que involucren telèfons mòbils, televisions, consoles, càmeres, rellotges i dispositius varis de IoT[14].

Per a poder utilitzar SQLite en aplicacions de Flutter, és necessari utilitzar una llibreria anomenada sqflite[15], que es pot trobar en pub.dev. Aquesta proveeix les eines necessàries per a manejar una base de dades SQLite en un dispositiu mòbil Android o iOS. La base de dades estarà llavors emmagatzemada en un arxiu SQLite que s'allotja en la memòria del telèfon mòbil.

[13] ***Appropriate uses for SQLite*. (s. f.).** https://www.sqlite.org/whentouse.html
[14] **IoT** o Internet de les coses: fa referència a la xarxa col·lectiva de dispositius connectats i a la tecnologia que facilita la comunicació entre els dispositius i el núvol, així com entre els propis dispositius.
[15] ***SQFlite | Flutter Package*. (s. f.).** Dart packages. https://pub.dev/packages/sqflite

3.4 Altres tecnologies o llibreries

3.4.1 Visual Studio Code

Visual Studio Code és un editor de codi desenvolupat per Microsoft en 2015. S'executa en l'escriptori i està disponible per a Windows, macOS i Linux. Ve amb suport integrat per a JavaScript i TypeScript i compta amb un ecosistema d'extensions per a altres llenguatges com a C++, Python, Dart... Visual Studio Code disposa d'un suport per a operacions de desenvolupament com a depuració 'debugging', execució de tasques i control de versions. El seu objectiu és proporcionar les eines necessàries per als desenvolupadors per a un cicle ràpid de codi-construcció-depuració[16].

fig 3.11 Logo Visual Studio Code

D'altra banda, per als programadors primerencs, Visual Studio Code compta amb unes eines realment útils per a aprendre programant. Entre elles es troben els suggeriments per a completar línies de codi, les solucions ràpides per a errors comuns "quick fix" i el fet de destacar les paraules clau del codi en diferents colors per a ajudar a identificar fàcilment els patrons de codificació. A més, del depurador de VSCode per a recórrer cada línia de codi i comprendre el que està succeint.

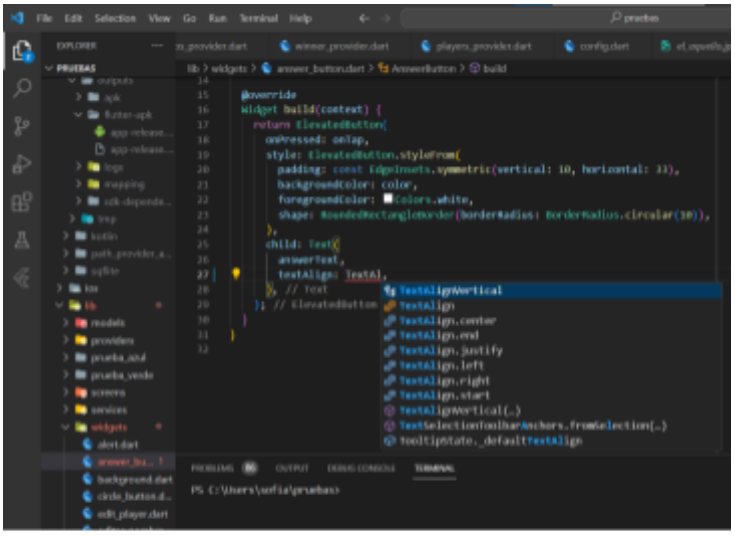


fig 3.12 Captura de pantalla extreta de Visual Studio Code

[16] ***Visual Studio Code Frequently asked questions*. (2021b, noviembre 3).** https://code.visualstudio.com/docs/supporting/faq#_what-is-the-difference-between-visual-studio-code-and-visual-studio-ide

3.4.2 Android Studio

fig 3.13 Logo Android Studio

Android Studio és l'entorn de desenvolupament integrat (IDE) oficial, basat en el potent editor de codi i les eines per a desenvolupadors d'IntelliJ IDEA[17], que s'utilitza en el desenvolupament d'apps per a Android. Android Studio ofereix diverses funcions que milloren la teva productivitat quan compiles apps per a aquest sistema operatiu, com, per exemple, Android Emulator.

Android Emulator simula dispositius Android perquè puguis provar la teva app en diferents dispositius sense necessitat de comptar amb els dispositius físics. L'emulador ofereix diversos avantatges com:

- Flexibilitat: es pot simular una varietat de dispositius incloent configuracions predefinides.
- Alta fidelitat: l'emulador proporciona gairebé totes les funcions d'un dispositiu Android real. Pots simular crides i missatges de text entrants, especificar la ubicació del dispositiu, fer servir diferents velocitats de xarxa, provar sensors de rotació i altres sensors de maquinari, accedir a Google Play Store...
- Velocitat: provar les aplicacions en l'emulador és més ràpid i fàcil que realitzar-lo en un dispositiu físic.

Gràcies a aquestes qualitats i a Android Emulator, desenvolupar aplicacions i provar-les és molt més senzill i pràctic.

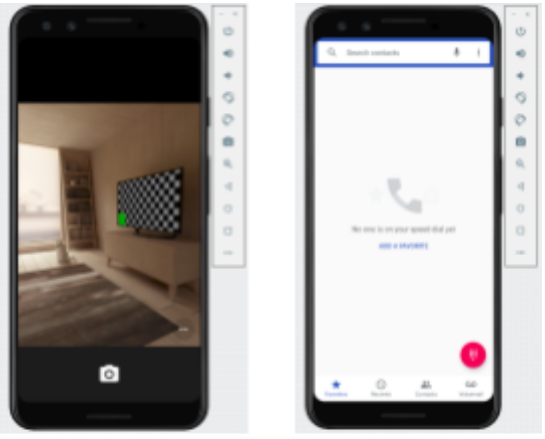


fig 3.14 i fig 3.15 Exemples d'Android Studio Emulator

[17] **Introducción a Android Studio. (s. f.-b).** ***Android Developers*****.** https://developer.android.com/studio/intro?hl=es-419

3.4.3 DBeaver

DBeaver és una eina de gestió de bases de dades universal, gratuïta i de codi obert per a desenvolupadors i administradors de bases de dades.

fig 3.16 Logo de DBeaver

DBeaver permet manipular les dades, crear informes analítics basats en registres de diferents emmagatzematges de dades i exportar informació en un format adequat. A més, ofereix un editor SQL, nombroses funcions d'administració, capacitats de migració de dades i esquemes i supervisió de sessions de connexió a bases de dades. Així mateix, DBeaver compta amb una interfície d'usuari acuradament dissenyada i implementada, que proporciona una interfície gràfica intuïtiva que facilita la creació, modificació i visualització de dades en les bases de dades.

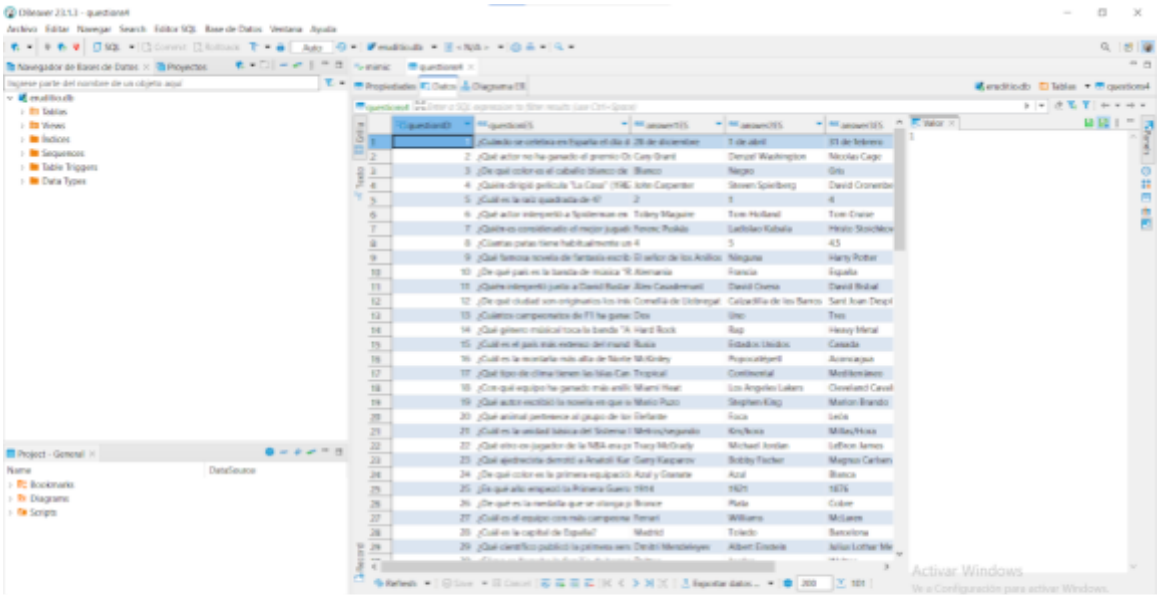


fig 3.17 Captura de pantalla extreta de DBeaver on es mostra una taula d'una base de dades

DBeaver és compatible amb més de 80 bases de dades, entre les quals es troba SQLite.

3.4.4 Riverpod

La gestió d'estat és un aspecte crucial en el desenvolupament d'aplicacions, ja que es refereix a com s'emmagatzemen i s'actualitzen les dades que afecten la interfície d'usuari i el comportament de l'aplicació. Per exemple, imaginem que en la nostra IU tenim un comptador que és un número que reflexa quantes vegades ha pres un usuari un botó. L'estat d'aquest widget està controlat per una variable que és de tipus int. Cada cop que es prem el botó, s'afegeix 1 al valor d'aquesta variable. Aleshores, com l'estat ha canviat, la IU ha de reflexar aquest canvi i "repintar-se" per mostrar el nou valor.

Existeixen multitud d'alternatives per gestionar l'estat en Flutter, en el meu cas he acabat utilitzant Riverpod.

fig 3.18 Logo de Riverpod

Riverpod és una biblioteca de gestió d'estat en Flutter. Va ser desenvolupat com una alternativa al paquet "Provider" (recomanat per Google fa anys) de Flutter i ofereix una manera més moderna i potent de gestionar l'estat en les teves aplicacions, que millora la claredat del codi i l'eficiència en comparació amb altres enfocaments.

En Riverpod, un provider és un objecte que encapsula un estat i permet escoltar aquest estat. En embolicar una part de l'estat en un provider, això permet accedir fàcilment a aquest estat en múltiples ubicacions, simplifica combinar aquest estat amb uns altres i permet optimitzacions de rendiment.

3.4.5 Pub.dev

fig 3.19 Logo de pub.dev

Pub.dev és el repositori oficial de paquets per a aplicacions de Dart i Flutter.[18] Els desenvolupadors poden utilitzar Pub.dev per a buscar, compartir i descarregar paquets i biblioteques que poden ser incorporats en els seus projectes. Aquests paquets i

[18] ***Dart packages*. (s. f.). Dart packages.** https://pub.dev/

biblioteques poden contenir codi, widgets personalitzats, eines i altres recursos que acceleren i faciliten el desenvolupament d'aplicacions en Dart i Flutter.

És a dir, Pub.dev és com una "biblioteca" on es poden trobar uns certs paquets compartits per altres desenvolupadors que permet crear ràpidament una aplicació sense haver de desenvolupar tot des de zero. Per exemple, el següent paquet et permet afegir una extensió a Flutter i afegir un geolocalitzador.

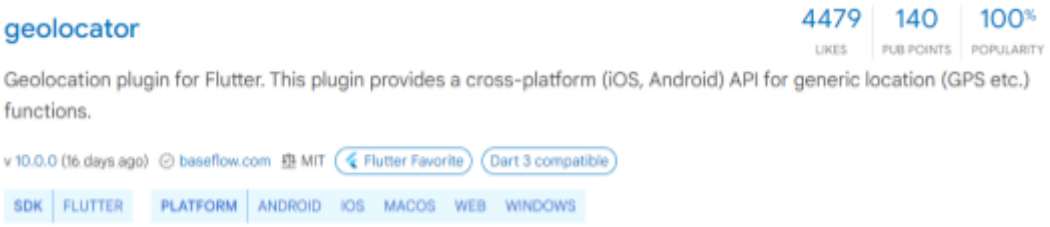


fig 3.20 Exemple d'un paquet de pub.dev

Dins d'aquest repositori existeixen una gran quantitat de paquets. Amb la finalitat de trobar els millors per a utilitzar en el teu projecte, dins de Pub.dev hi ha un sistema de puntuació. Les tres dimensions de puntuació d'un paquet pub són:

fig 3.21 Exemple del sistema de puntuació de pub.dev

- Likes: Una mesura de quants desenvolupadors els ha agradat un paquet.
- Punts pub: Inclou diverses dimensions de qualitat, com l'estil del codi, la compatibilitat amb la plataforma i la facilitat de manteniment.
- Popularitat: Mesura del nombre de desenvolupadors que utilitzen un paquet. Reflecteix el nombre d'aplicacions que depenen del paquet en els últims seixanta dies. L'escala normalitzada va del 100% (el paquet més fet servir) al 0% (el paquet menys emprat).

4. Marc pràctic

Per tal de dissenyar l'aplicació que volia crear he hagut d'aprofundir en el plantejament de diversos aspectes tals com: la digitalització del joc (és a dir, com es podrien adaptar les diverses mecàniques del joc de taula a una aplicació mòbil), la interfície (és a dir, l'aspecte de la pantalla i els botons que connecten a altres pantalles de manera que compleixi unes característiques bàsiques i necessàries del disseny d'IU) i la definició de la meva base de dades (ja que les diverses proves del joc consten de preguntes o opcions a escollir).

4.1 El meu disseny

4.1.1 Adaptació (digitalització) d'un joc de taula a una aplicació mòbil

Tal com he esmentat en l'apartat anterior, per poder crear l'aplicació basada en el joc de taula ja existent, havia de pensar com adaptar les mecàniques físiques i de les proves al mòbil.

El *Cranium* és un joc de taula que consta d'un taulell i uns daus per jugar. A més, durant les diferents proves utilitza més material físic com una espècie de bloc de notes o llibreta i un llapis per la prova de dibuixar, també té una prova de fer escultures amb plastilina, les preguntes es troben en targetes, hi ha un temporitzador per controlar el temps... Davant d'aquest fet, vaig fer el plantejament de l'aplicació amb alguns canvis i modificacions.

fig 4.1 Joc de taula *Cranium*

El primer que vaig fer va ser pensar una forma de sustituir el taulell i els daus, ja que em sería difícil incloure'ls en la programació. Per aquest motiu, en l'aplicació actual podem veure una ruleta o roda (en substitució del dau) i un sistema de puntuació

que consisteix a omplir uns buits, amb forma d'estrelles. Dit d'una altra forma, aconseguir el nombre rondes per superar escollit a la configuració de la partida. Aquest funcionament de seleccionar el nombre de proves superades necessari per poder guanyar també és una modificació, ja que al no tenir taulell, aquest aspecte del joc podia ser més flexible.

fig 4.2 Taulell del joc de taula *Cranium*

fig 4.3 Dau del joc de taula *Cranium*

fig 4.4 Ruleta de la aplicació

fig 4.5 Sistema de puntuació de l'aplicació

D'altra banda, també he fet un canvi en la forma de guanyar el joc, ja que s'ofereix la possibilitat de triar quantes proves has de passar favorablement i una vegada s'assoleix aquest número, es guanya. Al *Cranium*, en canvi, s'arriba a una final on has de dur a terme encara més proves, allargant bastant el temps de joc. Aquesta modificació està pensada per poder dinamitzar les partides.

Un concepte que s'ha mantingut igual ha sigut el nombre d'equips i de jugadors per equip. Tot i així, s'ha fet una variació. Al joc original, s'escull qui de l'equip fa algunes proves i qui mira d'endevinar. A l'aplicació mòbil, et diu directament qui ha de realitzar l'acció.

Pel que fa a les diferents proves, he fet diverses modificacions. El *Cranium* consta de quatre "seccions" de diferents colors. La prova vermella, que s'anomena *"Dato Nauta"* i consta de preguntes de cultura general on et donen quatre opcions i preguntes de vertader o fals. La prova blava, anomenada *"Gato Creativo"* intenta posar a prova les habilitats artístiques dels jugadors a través del dibuix i de l'escultura amb plastilina. La prova verda, anomenada *"Star Estelar"*, té diverses variacions com imitar a un famós, taral·lejar una cançó o fer mímica. I l'última, *"Lombriletras"*, la prova groga, que consisteix en diversos reptes lingüístics com enigmes, lletrejar paraules a l'inrevés, endevinar definicions... Amb la finalitat de no

fer unes modificacions excessives, el plantejament de les proves de l'aplicació mòbil va quedar bastant semblant.

La prova vermella consta d'una pregunta amb quatre opcions de resposta. Tant la pregunta com les quatre opcions de resposta surten a la pantalla del telèfon, a més d'un temporitzador. L'equip té trenta segons per posar-se d'acord i triar la resposta correcta prement el botó d'aquesta al mòbil.

La prova blava té tres variants de dibuixar (normal, amb els ulls tancats i amb la mà no dominant), però, en canvi, s'ha eliminat la de fer escultures, ja que es necessitaria plastilina i la intenció era que només es necessités un mòbil per jugar. Per aquest motiu, la pantalla de dibuixar ve incorporada amb una "pissarra" per poder pintar.

La prova verda té les mateixes versions (imitar, cantussejar i fer mímica) i en aquesta pantalla només es veu el temporitzador i els botons d'encertat o de rendir-se.

I finalment, la groga ha sigut eliminada, ja que a banda de ser difícil d'adaptar, no era gaire entretinguda.

Per últim, un altre afegit a l'aplicació ha sigut la possibilitat d'escollir entre tres opcions tant en les variants de la prova verda com en les de la blava. Aquest afegiment em semblava necessari, perquè moltes vegades no es coneixia a la persona que havies d'imitar o la cançó que havies de taral·lejar i, per tant, perdies directament aquella ronda. Amb la intenció de fer més accessible el joc en aquest aspecte, es va afegir aquesta opció d'escollir.

4.1.2 Interfície (UI)

Com s'explica a l'apartat 2.4 Disseny d'interfícies, hi existeixen unes característiques necessàries per poder idear l'IU de forma òptima. Aquests conceptes a prendre en consideració són l'aspecte visual de l'aplicació, el disseny d'interacció (fàcil d'usar amb bona disposició dels botons) i flexibilitat, permetent als usuaris un marge d'error.

Envers el primer punt, l'aspecte visual, vaig tractar que l'aplicació tingués uns colors i unes formes determinades prèviament. D'aquesta manera, a la programació hi ha diversos apartats especialitzats en aquest tema. Entre ells, es troba la "Paleta" de colors, on hi són guardades les referències dels colors sovint utilitzats perquè sigui més fàcil de fer servir la mateixa tonalitat a les diferents pantalles i, d'aquesta forma, l'aplicació tingui, visualment, una aparença sòlida. Així, també tinc un apartat on descric com han de ser els textos, tant la mida, com el color i la font. I a més, alguns widgets com certs botons, també estan definits des d'un inici per poder reutilitzar-los i perquè la gran part dels botons compleixin les mateixes característiques de mida, color, arrodoniment en la forma, etc.

```
class Palette {
  static const Color answerButton = Color.fromARGB(235, 206, 122, 255);
  static const Color fondoOscuro = Color.fromARGB(255, 53, 4, 116);
  static const Color fondoClaro = Color.fromARGB(255, 136, 63, 214);
  static const Color myPurple = Color.fromARGB(255, 58, 27, 91);
}
```

fig 4.6 Extracte de la programació on es mostra la Paleta de colors

```
    labelLarge: TextStyle(
        fontSize: 18,
        fontFamily: "Lato",
        fontWeight: FontWeight.bold,
        color: Colors.white),
    titleLarge: TextStyle(
        fontSize: 24,
        fontFamily: "Lato",
        fontWeight: FontWeight.w900,
        color: Colors.white),
    headlineSmall: TextStyle(
        fontSize: 30,
        fontFamily: "Lato",
        fontWeight: FontWeight.w900,
        color: Colors.white),
```

fig 4.7 Extracte de la programació on es mostra alguns tipus de fonts i la seva definició

Per tal que interactuar amb l'aplicació fos fàcil, s'havia de pensar la disposició dels botons i l'aspecte d'aquests. Per aquest motiu, molts d'ells contenen o bé un text explicatiu o una icona (per tractar de fer-ho més visual) o les dues.

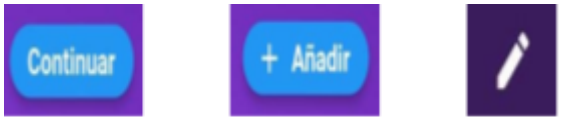


fig 4.8 Botons de continuar, afegir i editar de l'aplicació

D'altra banda, i pensant en el joc en si, també calia fer alguna explicació de les proves, per poder facilitar l'ús de l'aplicació i del contingut d'aquesta. Per aquest fet, abans d'algunes proves s'explica com funcionen i què s'ha de fer.

Finalment, respecte a la flexibilitat, hi ha diverses funcions. La primera, i més important, és el fet de deixar retrocedir. Tal com està programada l'aplicació, quan es passa des d'una pantalla a una altra, aquesta nova pantalla se sobreposa a l'anterior, formant com una espècie de "capes". Així, si l'usuari prem el botó de retrocedir del mòbil, l'aplicació mostrarà la pantalla anterior. Aquesta dinàmica, però, canvia una vegada comença la partida. Si es vol sortir mentre es juga, hi apareixerà primer un avís al mòbil per assegurar-se de què de veritat es vol sortir de la partida i no s'ha pres el botó sense voler. D'aquesta forma, donem l'anomenat "marge d'error" a l'usuari. Un altre mecanisme pensat per fer el disseny més flexible el trobem a les pantalles dels equips i jugadors. Aquí, es poden editar tant el nom dels equips i jugadors com el nombre d'aquest. Però si per cap motiu hi ha hagut una errada, es pot tornar a editar el nom i també, esborrar els elements afegits involuntàriament.

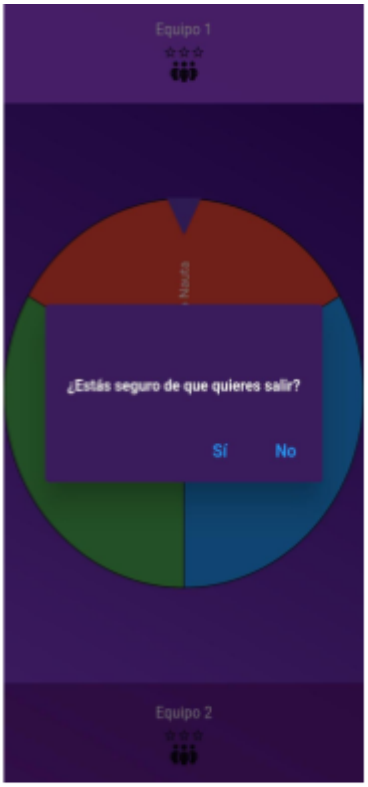


fig 4.9 Pantalla amb el missatge de comprovació

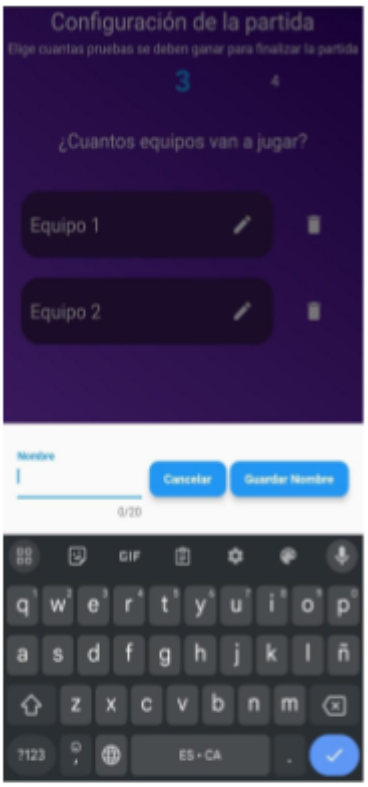


fig 4.10 Pantalla d'editar el nom

4.2 Implementació o programació

Start-Screen

En la primera pantalla es veuen diversos elements ordenats en una columna. A la part superior de la pantalla es veu una imatge (el logo de l' aplicació), seguit per un títol, un botó i, finalment, una fila amb dos botons.

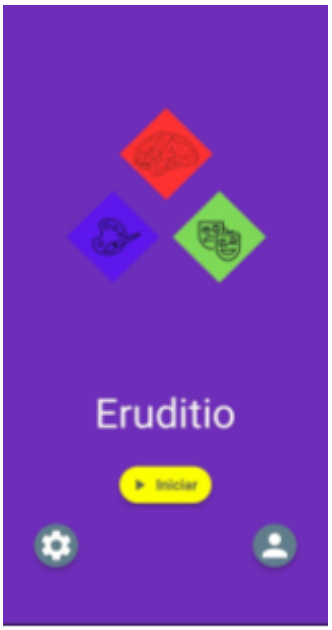


fig 4.11 Pantalla inicial de l'aplicació

El primer botó, que compta amb una icona, és el botó d'iniciar la partida. Si ho prenem, ens porta a la següent pantalla. Els botons de la fila, porten a altres pantalles. El de l'esquerra, a una de configuració i el de la dreta, a una de crèdits.

La pantalla de configuració compta amb dues "Sliders" o barres per a modificar el so, però com l'aplicació encara no conté cap so, realment en aquests moments no fan res. Sota d'aquestes barres, hi ha dos botons per a canviar l'idioma, que tampoc canvien res encara.

D'altra banda, la pantalla de crèdits disposa informació sobre els creadors de l'aplicació (és a dir, sobre mi).

Tant la pantalla de configuració com la de crèdits disposen d'una "AppBar", és a dir, una barra superior que apareix gràcies al "Scaffold", que té un botó incorporat de retrocedir, per a tornar a la pantalla anterior, és a dir, la "Start Screen".

fig 4.12 Pantalla de paràmetres

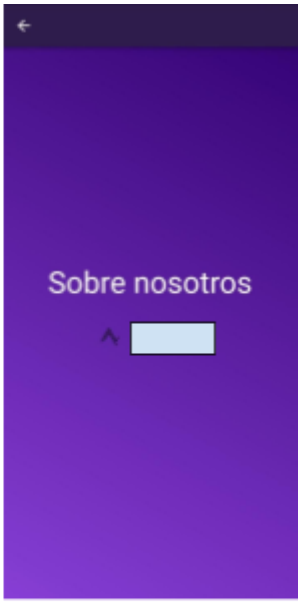


fig 4.13 Pantalla de crèdits

Configuració de la partida

Una vegada s'ha premut el botó d'iniciar de la pantalla anterior, comença la configuració de la partida. En aquesta pantalla veiem primer un "Number picker", un widget que compta amb un "spinner" (ruleta per a triar un número), en el que es tria el nombre de rondes.
Sota, veiem la llista dels equips gràcies al widget "ListView.builder". Aquest, el que fa és disposar una llista d'altres widgets, en aquest cas els contenidors on posa el nom de l'equip i les dues icones, de manera que es pot fer "scroll", és a dir, desplaçar a dalt o a baix per a poder veure el contingut de la llista, que no es mostra sencera, solament alguns elements.

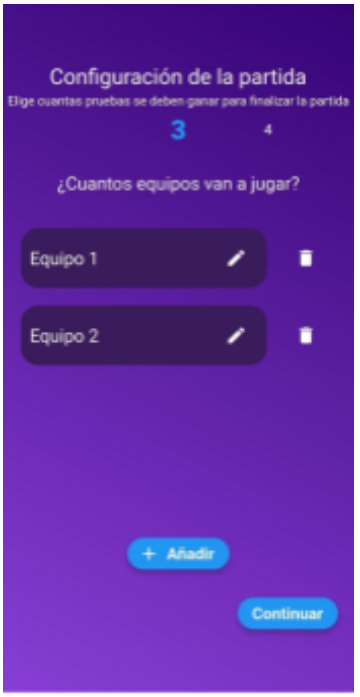


fig 4.14 Pantalla per configurar el nombre de rondes i d'equips

Cada element d'aquesta llista compta de tres altres elements, com he esmentat abans. El nom de l'equip, un IconButton per editar el nom i l'altre per eliminar l'equip, amb forma de paperera.

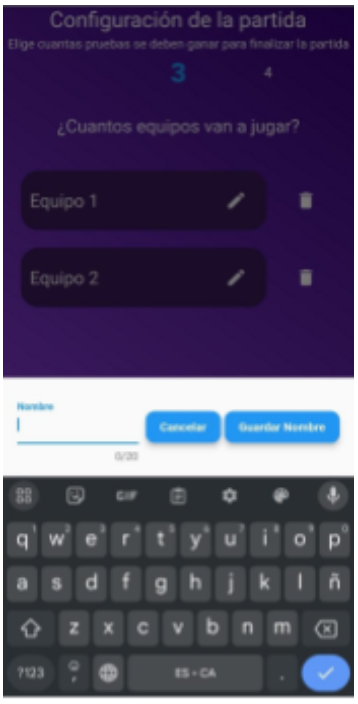


fig 4.15 Pantalla d'edició de noms

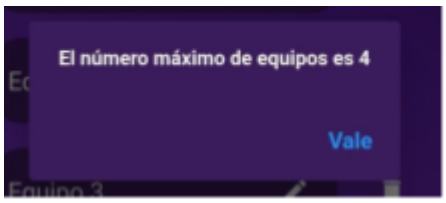
Quan es clica el botó d'editar el nom, apareix un BottomSheet, una pantalla petita que apareix des de baix, on podem editar el nom. Dins d'aquesta BottomSheet podem o bé guardar aquest nou nom o bé cancel·lar l'opció d'editar, pressionant el botó per a cadascuna d'aquestes opcions.

Així mateix, en pressionar el botó d'eliminar equip, s'eliminarà aquest de la llista i desapareixerà de la pantalla.

D'altra banda, tenim el botó d'afegir equips. Aquest sumarà un element a la llista fins a tenir un màxim de quatre equips. Si es torna a prémer per a afegir un altre, sortirà un missatge d'avís alertant que el nombre màxim d'equips és quatre.

Finalment, el botó de continuar portarà a la següent pantalla per a configurar els integrants de cada equip. Així com surt una alerta per a advertir del

màxim d'equips, també surt una per a avisar del mínim. Si es prem el botó de continuar, però hi ha menys de dos equips en la llista, sortirà aquest missatge d'avís.



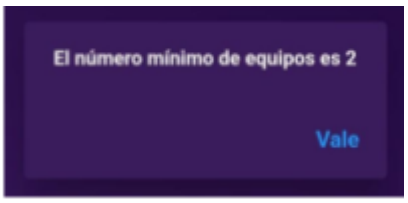


fig 4.16 i fig 4.17 Missatges d'alerta del màxim i el mínim d'equips

La pantalla individual de cada equip compta amb el nom triat del grup, a la part superior de la pantalla. Sota el nom, apareixen la llista dels jugadors (sent també una ListView.builder), i es poden afegir, eliminar i modificar els membres del grup. També surten les alertes d'un mínim de 2 jugadors i un màxim de 4. Apareixen tantes pantalles individuals de cada equip com equips s'hagin afegit en la pantalla anterior.

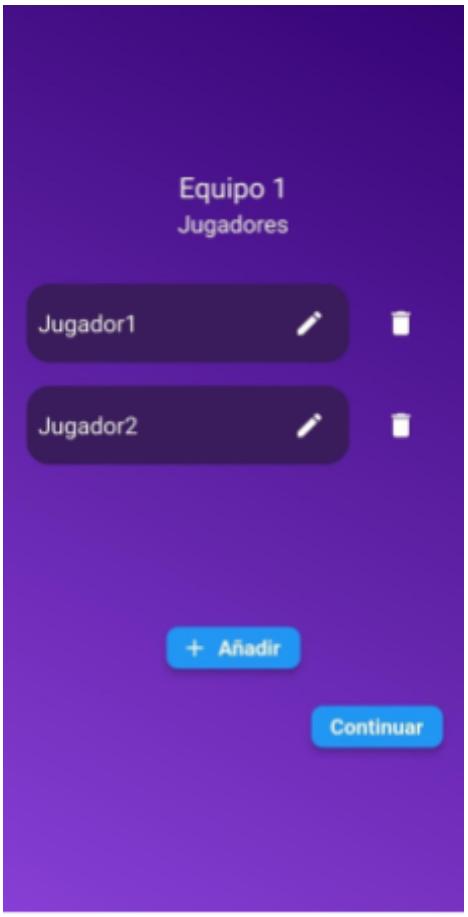
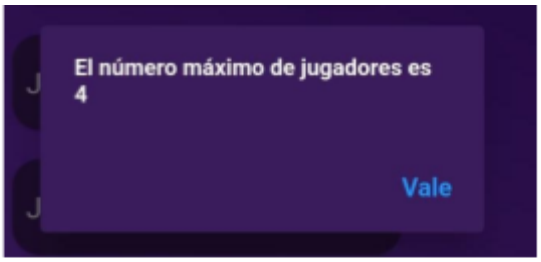


fig 4.18 Pantalla individual de l'equip



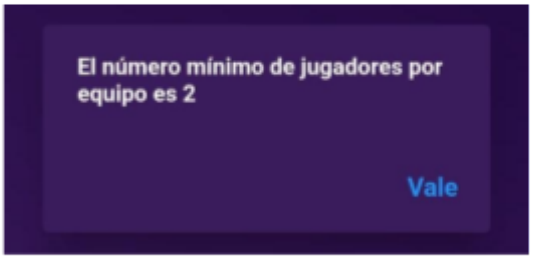


fig 4.19 i fig 4.20 Missatges d'alerta del màxim i el mínim de jugadors per equip

Game Screen

Aquesta pantalla consta de diversos elements importants.

Primerament, els marcadors, que es troben en la part superior i inferior de la pantalla. Els marcadors són individuals de cada equip i mostren el nom d'aquest i la seva puntuació. La puntuació s'indica a través d'un seguit d'icones amb forma d'estrella, inicialment buits, que aniran emplenant en passar satisfactòriament les diferents proves. Depenent del nombre de rondes triades en la configuració de la partida, apareixen més o menys estrelles.

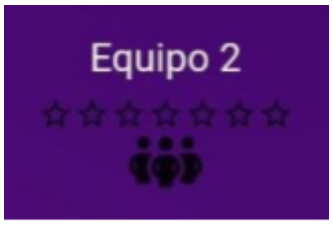


fig 4.21 Marcador de punts

A més, en funció de quants equips juguin, aquests marcadors es disposaran d'una manera o d'una altra.

fig 4.22, fig 4.23 i fig 4.24 Diferentes disposicions dels marcadors a la pantalla de joc

fig 4.25 Alerta de torn

D'altra banda, el joc indica el torn de l'equip que ha de jugar mitjançant un missatge d'alerta. Aquest missatge compta amb un text (que assenyala de quin equip és el torn) i un botó per a continuar. Així mateix, cada torn s'anirà mostrant en il·luminar el marcador de l'equip al qual li toca jugar.

Finalment, tenim la ruleta. La ruleta disposa de tres seccions diferents, corresponents a cada prova, cadascuna d'un color diferent. La ruleta es gira arrossegant una mica per, a continuació, deixar anar. Després de girar durant 1 segon, la ruleta es para, donant a conèixer quina és la prova a realitzar.

fig 4.26 Ruleta

Prova vermella

La prova vermella, anomenada “Dada Nauta”, és una prova de cultura general on es fa una pregunta i es donen quatre opcions. Aquesta pantalla compta, doncs, amb un temporitzador en la part superior, seguit de la pregunta i quatre botons, que corresponen a cada possible resposta.

Per a respondre a la pregunta, simplement s'ha de pressionar el botó que conté l'opció que creiem correcta. Si la resposta és correcta, aquest botó canviarà de color a verd. Per contra, si fallem, es tornarà vermell i la resposta correcta és marcarà en verd.

Una vegada hem respost, a la part inferior de la pantalla apareixerà un nou botó per a continuar la partida. Si el temporitzador arriba a 0 abans que s'hagi donat una resposta, també comptarà com a un error i es mostrarà quina era la resposta correcta.

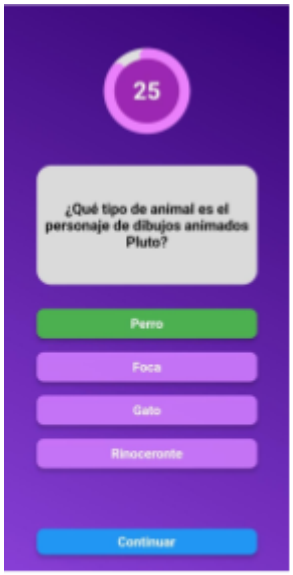


fig 4.27 Pantalla de la prova vermella amb la resposta encertada

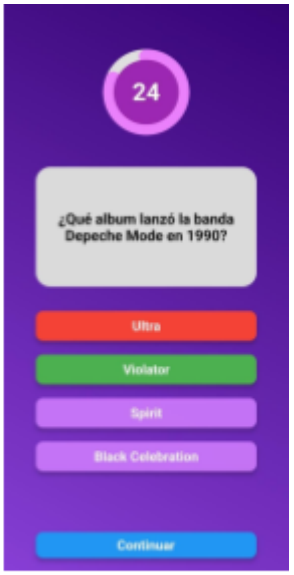


fig 4.28 Pantalla de la prova vermella amb la resposta fallada

Prova verda

La prova verda, anomenada “Star Estelar”, té 3 variacions: Mímica, Imitació o Taral·lejar.

Abans de mostrar la pantalla de joc, apareix una pantalla amb el nom de la prova que toca i una breu explicació de com jugar. Sota, apareix un contenidor que, en primer lloc, anuncia a quin jugador de l'equip li toca fer la prova, i posteriorment, les opcions.

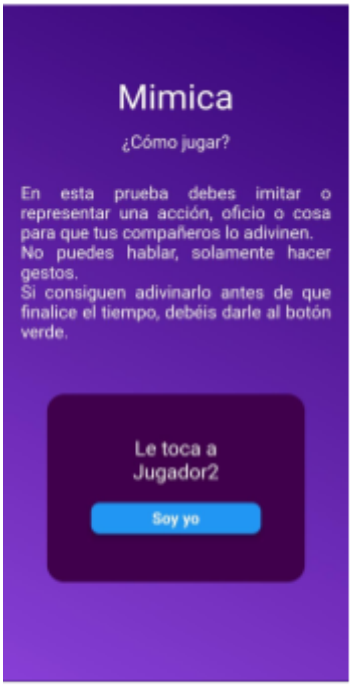


fig 4.29 Pantalla precedent a la prova verda

fig 4.30 Pantalla precedent a la prova verda amb les opcions

D'aquesta manera, primer hi ha un text indicant qui ha de tenir el mòbil i un botó que hi ha sota per confirmar que és ell. Després, apareixen 3 botons amb les opcions a triar per a dur a terme la prova.

Cada variació de la prova té les seves pròpies opcions; la Mímica té accions o objectes, la Imitació famosos i la Taral·lejar, cançons.

Una vegada s'hagi premut el botó per a triar l'opció que es prefereixi, apareixerà una altra pantalla.

Aquesta segona pantalla és bastant senzilla. En la part superior compta amb un temporitzador de 45 segons. Durant aquest temps els jugadors hauran de fer la prova. Si s'encerta, s'ha de prémer el botó verd i apareixerà un missatge, amb un fons verd, que indica quina era la resposta correcta i un botó per a continuar amb la partida. Si s'acaba el temps i no s'encerta o si l'equip decideix rendir-se prement el botó vermell, apareixerà el mateix missatge que en el cas anterior, però aquesta vegada de color vermell.

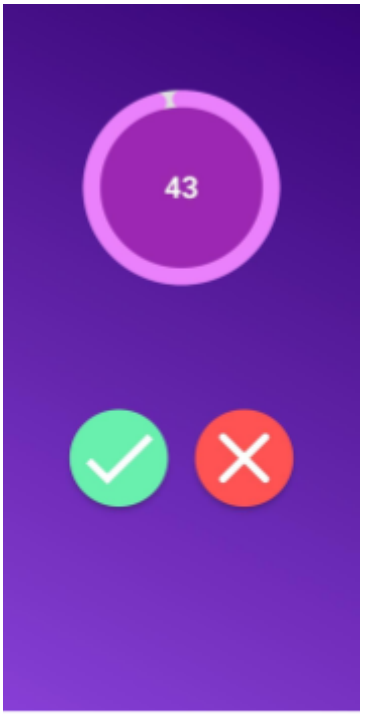


fig 4.31 Pantalla de la prova verda

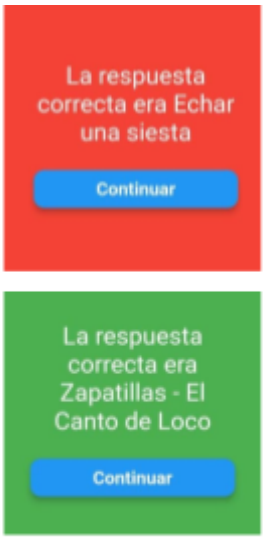


fig 4.32 i fig 4.33 Missatges amb la solució de la prova

Prova blava

La prova blava, anomenada “Gato Creativo”, té també 3 variacions; Dibuixar, Dibuixar amb els ulls tancats i Dibuixar amb la mà no dominant. Totes les variacions comparteixen les mateixes opcions a l'hora de triar.

Així com en la verda, abans de començar la prova, es mostra una pantalla amb l'explicació de la prova a realitzar. També apareix el missatge del jugador a qui li toca i, després, les opcions entre les quals triar.

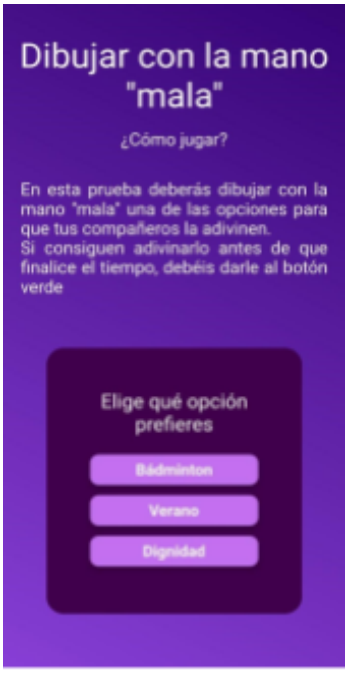


fig 4.34 Pantalla precedent a la prova blava amb les opcions

La pantalla de dibuixar compta amb diversos elements. En la part superior, d'un temporitzador de 45 segons i a la part inferior, els botons d'encertar o rendir-se.

Respecte a la zona de dibuix, hi ha una espècie de llenç (realment és un Contenidor). Aquesta zona blanca és on es pot dibuixar. Sota el llenç, hi ha dos botons senyalitzats amb icones. El de l'esquerra, obre un panell per a canviar de color la brotxa. El de la dreta, elimina el que havia dibuixat en el llenç.

fig 4.35 Pantalla de la prova blava amb el llenç

fig 4.36 Panell per cambiar el color de la brocha

Tal com en la prova verda, després d'encertar o fallar apareixen els missatges amb la resposta que era correcta i els seus respectius colors, vermell en cas de fallar i verd en cas d'encertar. Sota, apareix el botó de continuar, que porta una altra vegada a la pantalla de joc.

Winner Screen

Una vegada un dels equips hagi aconseguit guanyar tantes rondes com es va triar a l'inici, aquest equip haurà guanyat i apareixerà una pantalla per a indicar-lo.

Aquesta pantalla manté els marcadors, amb els noms dels equips i les seves puntuacions, repartits entre la part superior i inferior de la pantalla. En el centre trobem el missatge de l'equip guanyador, indicant el nom d'aquest i disposant una icona d'una corona en la pantalla. Al final del tot, apareix un botó que si premem, tornarà al menú, és a dir, a la Start Screen i des d'allí es podrà tornar a jugar si així es desitja.

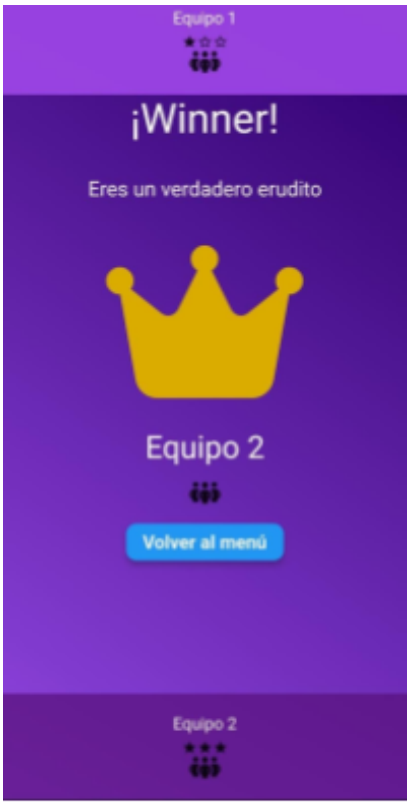


fig 4.37 Pantalla final que indica el guanyador del joc

4.2.1 Definició de la meva base de dades

Com el joc compta amb un elevat nombre de proves i preguntes, he hagut d'utilitzar una base de dades. Com que no són unes dades massives ni tenen una estructura molt particular ni es necessita connexió a Internet, la millor opció era fer servir una base de dades SQL.

En la base de dades, per cada prova he creat una taula que conté la informació necessària per jugar.

A l'hora d'escriure les diferents preguntes, com a primer pas, he fet ús de Fulls de Càlcul de Google. Allà, he creat un full per cada taula. Dintre del full, he creat una capçalera que conté el nom de les columnes de la base de dades i he omplert les files amb la informació de les preguntes i les opcions per a cada prova.

Un cop he omplert les dades, he exportat els fulls de càlcul en arxius CSV (*). Després, he utilitzat un script de Python per llegir els arxius CSV i introduir la informació a la base de dades SQLite. El script fa servir una llibreria anomenada Pandas que facilita llegir de l'arxiu CSV i convertir-ho en un format que entengui la base de dades.

```
21  if __name__ == '__main__':
22      # aquí es crea la base de dades SQLite en un arxiu que s'anomena eruditio.db
23      conn = create_connection(r"./eruditio.db")
24
25      # es crea un cursor per poder interactuar amb la base de dades
26      cursor = conn.cursor()
27
28      ##### QUESTIONS4
29      # s'executa el codi SQL que esborra la taula questions4 si ja existeix
30      cursor.execute("DROP TABLE IF EXISTS questions4")
31
32      # codi SQL que crea la taula questions 4
33      questions4_table = """
34          CREATE TABLE "questions4"(
35              "questionID" Integer NOT NULL PRIMARY KEY,
36              "questionES" Text NOT NULL,
37              "answer1ES" Text NOT NULL,
38              "answer2ES" Text NOT NULL,
39              "answer3ES" Text NOT NULL,
40              "answer4ES" Text NOT NULL
41          );
42      """
43      cursor.execute(questions4_table)
44
45      # llegir l'arxiu csv i guardar el contingut en una estructura de
46      # daates que s'anomena dataframe
47      questions4 = pd.read_csv('./csv/questions4.csv')
48
49
50      # s'escriuen les dades del csv en la taula questions4 que hem creat just abans
51      # el nom de les columnes del csv ha de ser el mateix que el de la definició
52      # de la base de dades
53      questions4.to_sql('questions4', conn, if_exists='append', index=False)
54      #pr_quest = cursor.execute('''SELECT * FROM questions4''').fetchall()
55
```

fig 4.38 Script de Python

A la línia 33 del codi, es crea la taula, en aquest cas anomenada questions4, amb el comandament create table. Dins del create table, especifiquen el nom de les columnes, el tipus i alguna restricció més. Per exemple, a la línia 35 es troba el "questionID" de tipus int, que actua com la nostra clau primària. A la següent línia veiem el "questionES" de tipus text i que no pot ser null, que seria una pregunta del joc, en aquest cas, en espanyol.

La base de dades consta, doncs, de cinc tables. La de preguntes amb quatre possibles respostes, la de dibuixar que es troben totes a la mateixa taula i unes altres tres per cada prova verda (imitar, taral·larejar i fer mímica).

Una vegada el script de Python ha finalitzat, tinc l'arxiu eruditio.db que conté la base de dades SQLite. Aquest arxiu es copia dins d'una carpeta del projecte de Flutter del desenvolupament de l'aplicació. El codi de l'aplicació de Flutter el primer que fa en iniciar-se és mirar si existeix l'arxiu que conté la base de dades a la memòria interna del mòbil i, si no existeix, el copia.

Aleshores, a partir d'aquest moment que la base de dades existeix al mòbil i es pot utilitzar per treure la informació de les preguntes per jugar.

5. Conclusions

5.1 Dificultats

Durant el desenvolupament del treball i, sobretot, el de l'aplicació, han sorgit i he hagut de fer front a diverses dificultats. El primer problema va ser fer el curs, que a banda de ser bastant llarg i durar el doble del que inicialment em pensava, en certes classes em costava seguir, no pel fet de fer el curs en anglès, sinó perquè molts conceptes eren totalment nous per a mi. Això, m'ha obligat a parar sovint a reflexionar i processar el que acabaven d'explicar.

A més, una vegada fet el curs, realment no m'he sentit tan preparada com m'hagués agradat per desenvolupar el joc.

Finalment, òbviament he tingut dificultats durant la programació en si. Però he pogut solucionar els problemes buscant en pub.dev, en tutorials, al curs... Al final, Flutter té un bon entorn amb ajudes en el qual pots recolzar-te per desenvolupar els teus projectes i, a mi, personalment, m'ha sigut molt útil.

5.2 Treball futur

De cara al futur, si continuo estudiant alguna cosa relacionada amb la programació i la informàtica, tinc algunes idees per ampliar l'aplicació mòbil actual. Entre elles, es troben les següents:

- La primera és afegir la possibilitat de canviar d'idioma, ja que inicialment el joc es troba en castellà i m'agradaria que hi hagués també una versió en català o, inclús, en anglès.
- La segona és afegir so tant per fer girar la ruleta com a l'hora de prémer els botons indicant si s'ha encertat o no i, fins i tot, una música de fons. Tant la proposta d'abans com aquesta es podrien modificar a la pantalla de "Settings" o ajustos, que actualment ja existeix, però realment no té cap funció.
- L'última idea es tracta d'afegir alguna prova nova per substituir la prova groga del joc de taula original, la qual va quedar eliminada durant el plantejament de l'aplicació.

5.3 Conclusions personals

Flutter és una eina molt útil. Compta amb un entorn d'aprenentatge amb una gran varietat d'instruments o ajudes per poder crear aplicacions multiplataforma com la pàgina oficial de Flutter, el seu canal oficial de Youtube on pugen vídeos explicatius de les seves funcions, cursos que es troben en llocs com Udemy, tutorials d'altres programadors que es poden veure a internet, la pàgina de pub.dev on pots trobar projectes i widgets creats per altres persones que et poden ajudar, etc.

Certamen, algunes funcions de Flutter no són tan fàcils d'entendre o aplicar, com bé podria ser el fet de passar informació, on es pot utilitzar Riverpod, que, igualment, no és tampoc fàcilment comprensible, però sí que actua com ajuda.

De totes maneres, i encara tenint uns certs defectes, si en un futur desenvolupés una altra aplicació, segurament faria servir Flutter.

Com a conclusions més personals, penso que probablement aquest projecte ha estat considerablement ambiciós, tenint en compte que no tenia coneixements de programació i, per tant, durant algunes fases del treball m'he pogut sentir frustrada o que el que estava fent sortia de "el meu abast". Això ha pogut ser perquè després de fer el curs no m'he sentit del tot preparada per fer front a tot el desenvolupament d'una aplicació des de zero. De totes maneres, he sigut capaç de sortir-me de la situació i acabar el joc, tenint un resultat satisfactori des del meu punt de vista.

Per aquest motiu, del treball puc treure conclusions personals positives, ja que m'ha ajudat a aclarir una mica el meu futur acadèmic o els meus possibles estudis.

6. Agraïments

Primerament, volia agrair la meva tutora pels consells i les propostes de millora donades, ja que m'han ajudat molt a l'hora de fer el treball.

En segon lloc, volia agrair a la meva família pel suport donat durant aquests mesos i per ajudar-me a provar l'aplicació. I especialment, gràcies al meu germà, per ajudar-me amb els conceptes que no acaba d'entendre, per haver-me donat ànims durant aquest temps i per tenir tanta paciència amb mi.

7. Bibliografia

Per tal de realitzar el treball he utilitzat diverses pàgines web:

- *Flutter_Fortune_Wheel | Flutter Package*. (s. f.). Dart packages. https://pub.dev/packages/flutter_fortune_wheel
- *FAQ*. (s. f.). Flutter. https://docs.flutter.dev/resources/faq#who-is-flutter-for
- *¿Qué es una base de datos relacional?* (s.f.). https://www.oracle.com/es/database/what-is-a-relational-database/
- *¿Qué es una base de datos?* (s. f.). https://www.oracle.com/es/database/what-is-database/

- *Bases de datos no relacionales | Bases de datos de gráficos | AWS*. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/nosql/
- *¿Qué son las bases de datos NoSQL? | IBM*. (s.f.). https://www.ibm.com/es-es/topics/nosql-databases
- *Mejores prácticas de seguridad de bases de datos*. (s.f.). https://www.oracle.com/es/database/nosql/what-is-nosql/
- *¿Qué es una base de datos clave-valor?* (s.f.). Amazon Web Services, Inc. https://aws.amazon.com/es/nosql/key-value/
- *¿Qué es una base de datos de documentos?* (s.f.). Amazon Web Services, Inc. https://aws.amazon.com/es/nosql/document/
- *Mejores prácticas de seguridad de bases de datos*. (s.f.). https://www.oracle.com/es/database/nosql/what-is-nosql/#benefits-of-a-nosql-database
- *About SQLite*. (s.f.). https://www.sqlite.org/about.html
- *Appropriate Uses For SQLite*. (s.f.). https://www.sqlite.org/whentouse.html
- *SQLite: la base de datos embebida*. (s. f.). SG Buzz. https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida
- KeepCoding, R. (2023, 14 marzo). ¿Qué es SQLite? | KeepCoding Bootcamps. *KeepCoding Bootcamps*. https://keepcoding.io/blog/que-es-sqlite/
- *Dart Overview*. (s. f.). Dart. https://dart.dev/overview
- *¿Qué es el lenguaje de programación DART?* (2020, 30 octubre). inLab FIB. https://inlab.fib.upc.edu/es/blog/que-es-el-lenguaje-de-programacion-dart
- *Get started with Visual Studio code*. (2021, 3 noviembre). https://code.visualstudio.com/learn
- *Visual Studio Code Frequently asked questions*. (2021, 3 noviembre). https://code.visualstudio.com/docs/supporting/faq#_what-is-the-difference-between-visual-studio-code-and-visual-studio-ide
- *Documentation for Visual Studio code*. (2021, 3 noviembre). https://code.visualstudio.com/docs
- Introducción a Android Studio. (s. f.). *Android Developers*. https://developer.android.com/studio/intro?hl=es-419
- Cómo ejecutar apps en Android emulator. (s. f.). *Android Developers*. https://developer.android.com/studio/run/emulator?hl=es-419

- Wadhwani, P. (2023, May 18). A Step-By-Step Tutorial Guide on Flutter Riverpod. *IT Blog | Mobile App Development India | Offshore Web Development - Bacancytechnology.com*. https://www.bacancytechnology.com/blog/flutter-riverpod-tutorial
- *Providers | RiverPod*. (s. f.). https://riverpod.dev/es/docs/concepts/providers
- Dbeaver. (s. f.). *Home*. GitHub. https://github.com/dbeaver/dbeaver/wiki
- *About | DBeaver Community*. (s. f.). https://dbeaver.io/about/
- *Introduction to mobile application Development | IBM*. (s. f.). https://www.ibm.com/topics/mobile-application-development
- Gonzalez, A. N. (2011b). ¿Qué es Android? *Xataka Android*. https://www.xatakandroid.com/sistema-operativo/que-es-android
- *What is Android Development? | IBM*. (s. f.). https://www.ibm.com/topics/android-development
- Filgueira, J. M. (s. f.). *¿Qué es iOS? | Gabit*. https://www.gabit.org/gabit/?q=es/que-es-ios
- Workana, & Workana. (2021). User Interface UI: qué es, para qué sirve y cómo desarrollar. *Glosario - Workana | El Glosario Workana explica terminología del mundo freelance, conceptos fundamentales del marketing y los negocios.* https://i.workana.com/glosario/que-es-user-interface-ui/
- Gómez, P. (2021, 10 diciembre). *¿Qué es UI? - DevCamp*. DevCamp. https://devcamp.es/que-es-ui/
- García, M. (2022, 16 diciembre). Lo que debes saber de diseño de interfaces y cómo aplicarlo en tu proyecto. *ESDESIGN*. https://www.esdesignbarcelona.com/actualidad/diseno-web/lo-que-debes-saber-de-diseño-de-interfaces-y-como-aplicarlo-en-tu-proyecto
- Thomsen, M. (2021, 15 diciembre). Introducing a brand new Pub.Dev | DART. *Medium*. https://medium.com/dartlang/pub-dev-redesign-747406dcb486

A més del curs de Udemy:

- Maximilian Schwarzmüller (2023). Flutter & Dart - The Complete Guide (2023 Edition)

8. Annexos

Programació

Screens

Start-Screen

```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:pruebas/screens/config_partida/config.dart';
import 'package:pruebas/screens/primeras/creditos_screen.dart';
import 'package:pruebas/screens/primeras/settings_screen.dart';
import 'package:pruebas/widgets/alert.dart';
import 'package:pruebas/widgets/background.dart';
import 'package:pruebas/widgets/circle_button.dart';

class StartScreen extends StatelessWidget {
  const StartScreen({super.key});
  static const routeName = '/menu';

  @override
  Widget build(context) {
    return Scaffold(
      body: SafeArea(
        child: Container(
          decoration: const BoxDecoration(
            gradient: LinearGradient(colors: [
              Color.fromARGB(255, 111, 46, 185),
              Color.fromARGB(255, 111, 46, 185),
            ], begin: Alignment.bottomLeft, end: Alignment.topRight),
          ),
          child: Center(
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                Image.asset(
                  'assets/images/logo_uno.png',
                  width: 300,
                  height: 400,
                ),
                const SizedBox(
                  height: 33,
                ),
                Text('Eruditio',
                    style: GoogleFonts.comicNeue(
                      color: Colors.white,
                      fontSize: 50,
                    )),
                const SizedBox(
                  height: 33,
```

```
                ),
                ElevatedButton.icon(
                  onPressed: () {
                    Navigator.of(context)
                        .pushNamed(ConfiguracionScreen.routeName);
                  },
                  style: ElevatedButton.styleFrom(
                    foregroundColor: Color.fromARGB(255, 93, 93, 93),
                    minimumSize: const Size(50, 45),
                    backgroundColor: const Color.fromARGB(255, 251,
255, 0),
                    elevation: 5.0,
                    shape: RoundedRectangleBorder(
                        //side: BorderSide(width: 1.0, color:
Colors.black54),
                        borderRadius: BorderRadius.circular(32)),
                  ),
                  icon: const Icon(Icons.play_arrow),
                  label: const Text('Iniciar'),
                ),
                const SizedBox(
                  height: 24,
                ),
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    CircleButton(
                        icon: Icons.settings,
                        iconSize: 45.0,
                        buttonFunction: () {
                          Navigator.of(context)
                              .pushNamed(SettingsScreen.routeName);
                        }),
                    const SizedBox(
                      width: 200,
                    ),
                    CircleButton(
                        icon: Icons.person,
                        iconSize: 45.0,
                        buttonFunction: () {

Navigator.of(context).pushNamed(Creditos.routeName);
                        }),
```

```
                  ],
                )
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Settings Screen

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:pruebas/widgets/background.dart';
import 'package:pruebas/widgets/circle_button.dart';

class SettingsScreen extends StatefulWidget {
  const SettingsScreen({super.key});

  static const routeName = 'settings';

  @override
  State<SettingsScreen> createState() {
    return _SettingsScreenState();
  }
}

class _SettingsScreenState extends State<SettingsScreen> {
  var rating = 50.0;
  var ratingMusic = 50.0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text('Ajustes'),
          backgroundColor: Color.fromARGB(255, 46, 28, 76),
        ),
        body: SafeArea(
            child: Container(
          decoration: const BoxDecoration(
            gradient: LinearGradient(colors: [
```

```
            Color.fromARGB(255, 136, 63, 214),
            Color.fromARGB(255, 53, 4, 116),
          ], begin: Alignment.bottomLeft, end: Alignment.topRight),
        ),
        child: Center(
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              Text(
                'Sonido',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 33,
                ),
              ),
              const SizedBox(
                height: 10,
              ),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  const Icon(Icons.volume_up),
                  Slider(
                    value: rating,
                    onChanged: (newRating) {
                      setState(() => rating = newRating);
                    },
                    min: 0,
                    max: 100,
                  ),
                ],
              ),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  const Icon(Icons.music_note),
                  Slider(
                    value: ratingMusic,
                    onChanged: (newRating) {
                      setState(() => ratingMusic = newRating);
                    },
                    min: 0,
                    max: 100,
```

```
        ),
      ],
    ),
    const SizedBox(
      height: 24,
    ),
    Text(
      'Idioma',
      style: GoogleFonts.lato(
        fontSize: 33,
        color: Colors.white,
      ),
    ),
    const SizedBox(
      height: 30,
    ),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        TextButton(
          child: const Text(
            'Català',
            style: TextStyle(
              color: Color.fromARGB(255, 0, 0, 0),
              fontSize: 25,
            ),
          ),
          onPressed: () {},
        ),
        const SizedBox(
          width: 30,
        ),
        TextButton(
          onPressed: () {},
          child: const Text(
            'Castellano',
            style: TextStyle(
              color: Colors.black,
              fontSize: 25,
            ),
          ),
        ),
      ],
```

```
                ),
                const SizedBox(
                  height: 170,
                )
              ],
            ),
          ),
        )));
  }
}
```

Creditos Screen

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:pruebas/widgets/background.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';

class Creditos extends StatelessWidget {
  const Creditos({super.key});

  static const String routeName = "creditos";

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Color.fromARGB(255, 46, 28, 76),
      ),
      body: SafeArea(
        child: Container(
          decoration: const BoxDecoration(
            gradient: LinearGradient(colors: [
              Color.fromARGB(255, 136, 63, 214),
              Color.fromARGB(255, 53, 4, 116),
            ], begin: Alignment.bottomLeft, end: Alignment.topRight),
          ),
          child: Center(
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                Text('Sobre nosotros',
                    style: GoogleFonts.lato(
```

```
                      color: Colors.white,
                      fontSize: 40,
                    )),
                const SizedBox(
                  height: 33,
                ),
                Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: const [
                      FaIcon(FontAwesomeIcons.thinkPeaks),
                      SizedBox(
                        width: 30,
                      ),
                      Text(
                        '____',
                        style: TextStyle(
                          fontSize: 23,
                        ),
                      ),
                    ]),
                const SizedBox(
                  height: 50,
                ),
                                ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Configuration 1

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:numberpicker/numberpicker.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:pruebas/providers/alerta_turno_provider.dart';
import 'package:pruebas/providers/num_rondas_provider.dart';
import 'package:pruebas/providers/players_provider.dart';
import 'package:pruebas/providers/teams_provider.dart';
import 'package:pruebas/providers/turno_provider.dart';
import 'package:pruebas/providers/winner_provider.dart';
import 'package:pruebas/screens/config_partida/config_team.dart';
```

```dart
import 'package:pruebas/widgets/background.dart';
import 'package:pruebas/widgets/equipos.dart';
import 'package:pruebas/providers/puntos_provider.dart';

import '../../constants/palette.dart';

class ConfiguracionScreen extends ConsumerStatefulWidget {
  const ConfiguracionScreen({super.key});

  static const String routeName = "config-teams";

  @override
  ConsumerState<ConfiguracionScreen> createState() {
    return _ConfiguracionScreenState();
  }
}

class _ConfiguracionScreenState extends
ConsumerState<ConfiguracionScreen> {
  int _currentValue = 3;

  void addTeam() {
    var nombreEquipos = ref.read(teamsNameProvider);
    var teamsNameNotifier = ref.read(teamsNameProvider.notifier);

    if (nombreEquipos.isEmpty) {
      teamsNameNotifier.addTeam('Equipo 1');
    } else if (nombreEquipos.length == 1) {
      teamsNameNotifier.addTeam('Equipo 2');
    } else if (nombreEquipos.length == 2) {
      teamsNameNotifier.addTeam('Equipo 3');
    } else if (nombreEquipos.length == 3) {
      teamsNameNotifier.addTeam('Equipo 4');
    } else {
      showDialog(
        context: context,
        builder: (ctx) => AlertDialog(
          backgroundColor: Palette.myPurple,
          content: const Text('El número máximo de equipos es 4'),
          actions: [
            TextButton(
                onPressed: () {
                  Navigator.pop(ctx);
```

```
                },
                child: const Text('Vale'))
          ],
        ),
      );
    }
  }

  bool checkNumTeams() {
    var numTeams = ref.read(teamsNameProvider).length;
    if (numTeams < 2) {
      return false;
    }
    return true;
  }

  void loadNextScreen() {
    var checkeo = checkNumTeams();

    if (checkeo) {
      var numTeams = ref.read(teamsNameProvider).length;
      ref.read(playersNameProvider.notifier).initializeLists(numTeams);
      //ref.invalidate(puntosProvider);
      ref.invalidate(turnosProvider);
      ref.invalidate(alertaTurnoProvider);

      ref.invalidate(winnerProvider);
      ref.read(puntosProvider.notifier).initializeLists(numTeams);

Navigator.of(context).pushNamed(ConfiguracionTeamsScreen.routeName,
          arguments: {'teamIndex': 0});
    } else {
      showDialog(
        context: context,
        builder: (ctx) => AlertDialog(
          backgroundColor: Palette.myPurple,
          content: const Text('El número mínimo de equipos es 2'),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.pop(ctx);
              },
              child: const Text('Vale'),
```

```
            ),
          ],
        ),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Background(
      child: SafeArea(
        child: LayoutBuilder(builder: (context, constraints) {
          return Center(
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                Text(
                  'Configuración de la partida',
                  style: GoogleFonts.lato(
                    color: Colors.white,
                    fontSize: 24,
                  ),
                ),
                const SizedBox(
                  height: 5,
                ),
                Text(
                  'Elige cuantas pruebas se deben ganar para finalizar
la partida',
                  style: GoogleFonts.lato(
                    color: Colors.white,
                    fontSize: 14,
                  ),
                ),
                NumberPicker(
                  minValue: 3,
                  maxValue: 7,
                  value: ref.watch(numRondasProvider),
                  onChanged: (value) {
                    /*setState(
                      () {
                        _currentValue = value;
                      },
```

```
                    );*/
                    var numRondasNotifier =
                        ref.read(numRondasProvider.notifier);
                    numRondasNotifier.editNumRondas(value);
                  },
                  axis: Axis.horizontal,
                ),
                const SizedBox(
                  height: 24,
                ),
                Text(
                  '¿Cuantos equipos van a jugar?',
                  style: GoogleFonts.lato(
                    fontSize: 20,
                    color: Colors.white,
                  ),
                ),
                const SizedBox(
                  height: 30,
                ),
                Equipos(
                  constraints:
                      BoxConstraints(maxHeight: constraints.maxHeight *
0.45),
                ),
                ElevatedButton.icon(
                  onPressed: () {
                    addTeam();
                  },
                  icon: const Icon(Icons.add),
                  style: ElevatedButton.styleFrom(
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(20)),
                  ),
                  label: const Text('Añadir'),
                ),
                const SizedBox(
                  height: 20,
                ),
                Row(
                  mainAxisAlignment: MainAxisAlignment.end,
                  children: [
                    ElevatedButton(
```

```
                    onPressed: loadNextScreen,
                    style: ElevatedButton.styleFrom(
                      shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(20)),
                    ),
                    child: const Text('Continuar'),
                  ),
                  const SizedBox(
                    width: 18,
                  ),
                ],
              ),
            ],
          ),
        );
      }),
    ),
  );
 }
}
```

Configuration 2 (teams)

```
import 'package:flutter/material.dart';
import 'package:numberpicker/numberpicker.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:pruebas/providers/teams_provider.dart';
import 'package:pruebas/screens/game/game_screen.dart';
import 'package:pruebas/widgets/background.dart';
import 'package:pruebas/widgets/edit_player.dart';
import 'package:pruebas/widgets/equipos.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/providers/players_provider.dart';

import '../../constants/palette.dart';

class ConfiguracionTeamsScreen extends ConsumerStatefulWidget {
  const ConfiguracionTeamsScreen({super.key});

  static const String routeName = "config-team";
  @override
  ConsumerState<ConfiguracionTeamsScreen> createState() {
    return _ConfiguracionTeamsScreenState();
```

```
  }
}

class _ConfiguracionTeamsScreenState
    extends ConsumerState<ConfiguracionTeamsScreen> {
  late int teamIndex;

  void addPlayer(int teamIndex) {
    var nombreJugadores = ref.read(playersNameProvider);
    var playersNameNotifier = ref.read(playersNameProvider.notifier);
    if (nombreJugadores[teamIndex].isEmpty) {
      playersNameNotifier.addPlayer('Jugador 1', teamIndex);
    } else if (nombreJugadores[teamIndex].length == 1) {
      playersNameNotifier.addPlayer('Jugador 2', teamIndex);
    } else if (nombreJugadores[teamIndex].length == 2) {
      playersNameNotifier.addPlayer('Jugador 3', teamIndex);
    } else if (nombreJugadores[teamIndex].length == 3) {
      playersNameNotifier.addPlayer('Jugador 4', teamIndex);
    } else {
      showDialog(
        context: context,
        builder: (ctx) => AlertDialog(
          backgroundColor: Palette.myPurple,
          content: const Text('El número máximo de jugadores es 4'),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.pop(ctx);
              },
              child: const Text('Vale'),
            ),
          ],
        ),
      );
    }
  }

  bool checkNumPlayers() {
    var playerNames = ref.read(playersNameProvider);

    var numPlayers = playerNames[teamIndex].length;

    if (numPlayers < 2) {
```

```
      return false;
    }
    return true;
  }

  void loadNextScreen() {
    var checkeo = checkNumPlayers();

    if (checkeo) {
      var numTeams = ref.read(teamsNameProvider).length;
      if ((teamIndex + 1) < numTeams) {

Navigator.of(context).pushNamed(ConfiguracionTeamsScreen.routeName,
            arguments: {'teamIndex': teamIndex + 1});
      } else {
        Navigator.of(context).pushAndRemoveUntil(
            MaterialPageRoute(
              builder: (context) => GameScreen(),
            ),
            (route) => route
                .isFirst // ir hacia atrás hasta que solo quede una
pantalla
            );
      }
    } else {
      showDialog(
        context: context,
        builder: (ctx) => AlertDialog(
          backgroundColor: Palette.myPurple,
          content: const Text('El número mínimo de jugadores por equipo
es 2'),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.pop(ctx);
              },
              child: const Text('Vale'),
            ),
          ],
        ),
      );
    }
  }
```

```dart
@override
Widget build(BuildContext context) {
  var arguments =
      ModalRoute.of(context)!.settings.arguments as Map<String, int>;
  teamIndex = arguments['teamIndex']!;

  var teamsName = ref.watch(teamsNameProvider);

  return Background(
    child: Center(
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          Text(
            teamsName[teamIndex],
            style: GoogleFonts.lato(
              color: Colors.white,
              fontSize: 24,
            ),
          ),
          const SizedBox(
            height: 5,
          ),
          Text(
            'Jugadores',
            style: GoogleFonts.lato(
              color: Colors.white,
              fontSize: 20,
            ),
          ),
          const SizedBox(
            height: 30,
          ),
          PlayerNameEditable(
            teamIndex: teamIndex,
          ),
          ElevatedButton.icon(
            onPressed: () {
              addPlayer(teamIndex);
            },
            icon: const Icon(Icons.add),
            label: const Text('Añadir'),
```

```
            ),
            const SizedBox(
              height: 20,
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.end,
              children: [
                ElevatedButton(
                  onPressed: () {
                    loadNextScreen();
                  },
                  child: const Text('Continuar'),
                ),
                const SizedBox(
                  width: 18,
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

Game Screen

```
import 'package:flutter/material.dart';
import 'package:pruebas/constants/palette.dart';
import 'package:pruebas/providers/alerta_turno_provider.dart';
import 'package:pruebas/providers/num_rondas_provider.dart';
import 'package:pruebas/providers/puntos_provider.dart';
import 'package:pruebas/providers/teams_provider.dart';
import 'package:pruebas/providers/turno_provider.dart';
import 'package:pruebas/providers/winner_provider.dart';
import 'package:pruebas/screens/game/custom_alert_dialog.dart';
import 'package:pruebas/screens/winner_screen.dart';
import 'package:pruebas/widgets/alert.dart';
import 'package:pruebas/widgets/background.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:pruebas/widgets/scoreboard.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/widgets/wheel.dart';
```

```dart
import 'package:collection/collection.dart';

class GameScreen extends ConsumerStatefulWidget {
  const GameScreen({super.key});

  static String routeName = 'game';

  @override
  ConsumerState<GameScreen> createState() => _GameScreenState();
}

class _GameScreenState extends ConsumerState<GameScreen> {
  Future<bool?> showWarning(BuildContext context) async =>
showDialog<bool>(
      context: context, builder: (context) => CustomAlertDialog());

  Future<bool> onBackPressed() async {
    final bool shouldPop = await showWarning(context) ?? false;

    return shouldPop;
  }

  @override
  Widget build(BuildContext context) {
    var teams = ref.watch(teamsNameProvider);
    var numTeams = teams.length;
    var numRondas = ref.watch(numRondasProvider);
    var puntos = ref.watch(puntosProvider);
    var turno = ref.watch(turnosProvider);
    var showAlertaTurno = ref.watch(alertaTurnoProvider);
    var winner = ref.watch(winnerProvider);
    return WillPopScope(
      onWillPop: onBackPressed,
      child: Background(
          child: SafeArea(
        child: Column(
          children: [
            Row(
                children: teams
                    .sublist(0, teams.length ~/ 2)
                    .mapIndexed((index, team) => Expanded(
                          child: GameScoreboard(
                            numRondas: numRondas,
```

```
                          index: index,
                        ),
                      ))
                  .toList()),
            Expanded(
              child: Stack(
                alignment: AlignmentDirectional.center,
                children: [
                  Wheel(),
                  showAlertaTurno
                      ? AlertaTurno(
                          teamName: teams[turno],
                        )
                      : SizedBox.shrink(),
                  winner != -1 ? WinnerScreen() : SizedBox.shrink(),
                ],
              ),
            ),
            Row(
                children: teams
                    .sublist(teams.length ~/ 2)
                    .mapIndexed((index, team) => Expanded(
                          child: GameScoreboard(
                            numRondas: numRondas,
                            index: teams.length ~/ 2 + index,
                          ),
                        ))
                    .toList())
          ],
        ),
      )),
    );
  }
}
```

Questions4 Screen

```
import
'package:circular_countdown_timer/circular_countdown_timer.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:pruebas/constants/palette.dart';
```

```dart
import 'package:pruebas/widgets/answer_button.dart';
import 'package:pruebas/widgets/background.dart';

import '../providers/alerta_turno_provider.dart';
import '../providers/puntos_provider.dart';
import '../providers/turno_provider.dart';
import 'game/custom_alert_dialog.dart';

class QuestionsScreen extends ConsumerStatefulWidget {
  const QuestionsScreen({
    super.key,
    required this.question,
    required this.answer1,
    required this.answer2,
    required this.answer3,
    required this.answer4,

    //required this.onSelectAnswer,
  });

  static const String routeName = "/questions";
  final String question;
  final String answer1;
  final String answer2;
  final String answer3;
  final String answer4;

  // final void Function(String answer) onSelectAnswer;

  @override
  ConsumerState<QuestionsScreen> createState() {
    return _QuestionsScreenState();
  }
}

class _QuestionsScreenState extends ConsumerState<QuestionsScreen> {
  //var currentQuestionIndex = 0;late List<String> shuffledAnswers;
  var controlador = CountDownController();

  var selectedResponse = -1;
  var correctResponse = -1;
  var responseColor = Colors.green;
```

```
  var normalColor = Palette.answerButton;
  late List<String> listaRespuestas;
  @override
  void initState() {
    super.initState();
    listaRespuestas = [
      widget.answer1,
      widget.answer2,
      widget.answer3,
      widget.answer4,
    ]..shuffle();

    WidgetsBinding.instance.addPostFrameCallback((timeStamp) {
      controlador.restart();
      //esto hace que justo despues de cargar el controller haga
restart
    });
  }

  void answerTimer() {
    for (int i = 0; i < listaRespuestas.length; ++i) {
      if (listaRespuestas[i] == widget.answer1) {
        setState(() {
          correctResponse = i;
        });
        break;
      }
    }
    setState(() {
      responseColor = Colors.red;
      selectedResponse = -2;
    });
    var turnonotifier = ref.read(turnosProvider.notifier);
    turnonotifier.pasarTurno();
    ref.read(alertaTurnoProvider.notifier).showAlertaTurno();
  }

  void answerQuestion(int selectedAnswer) {
    controlador.pause();
    for (int i = 0; i < listaRespuestas.length; ++i) {
      if (listaRespuestas[i] == widget.answer1) {
        setState(() {
          correctResponse = i;
```

```
        });
        break;
      }
    }
    if (listaRespuestas[selectedAnswer] == widget.answer1) {
      setState(() {
        responseColor = Colors.green;
      });

      var indiceTurno = ref.read(turnosProvider);

      //sumamos un punto al equipo
      var puntosNotifier = ref.read(puntosProvider.notifier);
      puntosNotifier.addPuntos(indiceTurno);
      // esto pasara el turno al siguiente equipo
      var turnonotifier = ref.read(turnosProvider.notifier);
      turnonotifier.pasarTurno();
      // y esto enseña el mensaje
      ref.read(alertaTurnoProvider.notifier).showAlertaTurno();
    } else {
      setState(() {
        responseColor = Colors.red;
      });
      var turnonotifier = ref.read(turnosProvider.notifier);
      turnonotifier.pasarTurno();
      ref.read(alertaTurnoProvider.notifier).showAlertaTurno();
    }
  }

  Future<bool?> showWarning(BuildContext context) async =>
showDialog<bool>(
      context: context, builder: (context) => const
CustomAlertDialog());

  Future<bool> onBackPressed() async {
    final bool shouldPop = await showWarning(context) ?? false;

    return shouldPop;
  }

  @override
  Widget build(context) {
    //final currentQuestion = questions[currentQuestionIndex];
```

```
    final double width = MediaQuery.of(context).size.width;
    final double height = MediaQuery.of(context).size.height;

    return WillPopScope(
      onWillPop: onBackPressed,
      child: Background(
        child: SizedBox(
          width: double.infinity,
          //otra forma para centrarlo es wrap la columna en un sized
box y ponerle esta medida de double.infinity
          child: Container(
            margin: const EdgeInsets.symmetric(horizontal: 45,
vertical: 20),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              // para que ocupe este espacio y este centrado
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: [
                Center(
                  child: CircularCountDownTimer(
                    duration: 30,
                    initialDuration: 30,
                    controller: controlador,
                    width: MediaQuery.of(context).size.width / 4,
                    height: MediaQuery.of(context).size.height / 4,
                    ringColor: Colors.grey[300]!,
                    ringGradient: null,
                    fillColor: Colors.purpleAccent[100]!,
                    fillGradient: null,
                    backgroundColor: Colors.purple[500],
                    backgroundGradient: null,
                    strokeWidth: 20.0,
                    strokeCap: StrokeCap.round,
                    textStyle: TextStyle(
                        fontSize: 33.0,
                        color: Colors.white,
                        fontWeight: FontWeight.bold),
                    textFormat: CountdownTextFormat.S,
                    isReverse: true,
                    isReverseAnimation: true,
                    isTimerTextShown: true,
                    autoStart: false,
                    onStart: () {
```

```
                debugPrint('Countdown Started');
              },
              onComplete: () {
                debugPrint('Countdown Ended');

                answerTimer();
              },
              onChange: (String timeStamp) {
                debugPrint('Countdown Changed $timeStamp');
              },
            ),
          ),
          Container(
            height: height * 0.20,
            width: width * 0.85,
            decoration: BoxDecoration(
                color: Color.fromARGB(255, 218, 218, 218),
                borderRadius: BorderRadius.circular(20)),
            child: Center(
              child: Text(
                //currentQuestion.text,
                widget.question,
                style: GoogleFonts.lato(
                  color: Color.fromARGB(255, 0, 0, 0),
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
                textAlign: TextAlign.center,
              ),
            ),
          ),
          const SizedBox(height: 30),
          /*...currentQuestion.shuffledList.map((answer) {
            return AnswerButton(
              answerText: answer,
              onTap: () {
                answerQuestion(answer);
              },
            );
          }),*/
          AnswerButton(
              answerText: listaRespuestas[0],
              color: selectedResponse == 0
```

```
                ? responseColor
                : correctResponse == 0
                    ? Colors.green
                    : normalColor,
            onTap: selectedResponse == -1
                ? () {
                    answerQuestion(0);
                    setState(() {
                      selectedResponse = 0;
                    });
                  }
                : () {}),
        const SizedBox(
          height: 12,
        ),
        AnswerButton(
            answerText: listaRespuestas[1],
            color: selectedResponse == 1
                ? responseColor
                : correctResponse == 1
                    ? Colors.green
                    : normalColor,
            onTap: selectedResponse == -1
                ? () {
                    answerQuestion(1);
                    setState(() {
                      selectedResponse = 1;
                    });
                  }
                : () {}),
        const SizedBox(
          height: 12,
        ),
        AnswerButton(
            answerText: listaRespuestas[2],
            color: selectedResponse == 2
                ? responseColor
                : correctResponse == 2
                    ? Colors.green
                    : normalColor,
            onTap: selectedResponse == -1
                ? () {
                    answerQuestion(2);
```

```
                            setState(() {
                              selectedResponse = 2;
                            });
                          }
                        : () {}),
                const SizedBox(
                  height: 12,
                ),
                AnswerButton(
                    answerText: listaRespuestas[3],
                    color: selectedResponse == 3
                        ? responseColor
                        : correctResponse == 3
                            ? Colors.green
                            : normalColor,
                    onTap: selectedResponse == -1
                        ? () {
                            answerQuestion(3);
                            setState(() {
                              selectedResponse = 3;
                            });
                          }
                        : () {}),
                const Expanded(
                  child: SizedBox(),
                ),
                selectedResponse == -1
                    ? SizedBox.shrink()
                    : ElevatedButton(
                        onPressed: () {
                          Navigator.of(context).pop();
                        },
                        child: const Text('Continuar'))
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Preverde Screen

```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:pruebas/widgets/background.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:pruebas/widgets/indicar_jugador.dart';

import '../screens/game/custom_alert_dialog.dart';

class PreVerde extends StatefulWidget {
  const PreVerde({
    super.key,
    required this.nombrePrueba,
    required this.player,
    required this.explicacion,
    required this.opcion1,
    required this.opcion2,
    required this.opcion3,
  });

  final String nombrePrueba;

  final String player;

  final String explicacion;

  final String opcion1;

  final String opcion2;

  final String opcion3;

  static const String routeName = "preverde";

  @override
  State<PreVerde> createState() => _PreVerdeState();
}

class _PreVerdeState extends State<PreVerde> {
  Future<bool?> showWarning(BuildContext context) async =>
showDialog<bool>(
      context: context, builder: (context) => const
CustomAlertDialog());
```

```
Future<bool> onBackPressed() async {
  final bool shouldPop = await showWarning(context) ?? false;

  return shouldPop;
}

@override
Widget build(BuildContext context) {
  return WillPopScope(
    onWillPop: onBackPressed,
    child: Background(
      child: Center(
          child: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          Text(widget.nombrePrueba,
              style: GoogleFonts.lato(
                color: Colors.white,
                fontSize: 40,
              )),
          const SizedBox(
            height: 16,
          ),
          Text(
            '¿Cómo jugar?',
            style: TextStyle(color: Colors.white, fontSize: 20),
          ),
          const SizedBox(
            height: 16,
          ),
          Container(
              margin: EdgeInsets.all(20),
              child: Center(
                child: Text(
                  widget.explicacion,
                  textAlign: TextAlign.justify,
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 20,
                  ),
                ),
              )),
```

```
            const SizedBox(
              height: 16,
            ),
            IndicarJugador(
              player: widget.player,
              opcion1: widget.opcion1,
              opcion2: widget.opcion2,
              opcion3: widget.opcion3,
              verde: true,
            )
          ],
        )),
      ),
    );
  }
}
```

Prueba Verde

```
import 'package:flutter/material.dart';
import
'package:circular_countdown_timer/circular_countdown_timer.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/widgets/background.dart';
import 'package:pruebas/widgets/circle_button.dart';

import '../providers/alerta_turno_provider.dart';
import '../providers/puntos_provider.dart';
import '../providers/turno_provider.dart';
import '../screens/game/custom_alert_dialog.dart';

class PruebaVerde extends ConsumerStatefulWidget {
  const PruebaVerde({super.key, required this.answer});

  final String answer;

  @override
  ConsumerState<PruebaVerde> createState() => _PruebaVerdeState();
}

class _PruebaVerdeState extends ConsumerState<PruebaVerde> {
  @override
  void initState() {
```

```
    super.initState();

    WidgetsBinding.instance.addPostFrameCallback((timeStamp) {
      controlador.restart();
      //esto hace que justo despues de cargar el controller haga
restart
    });
  }

  bool response = false;
  bool showAnswer = false;
  var controlador = CountDownController();

  Future<bool?> showWarning(BuildContext context) async =>
showDialog<bool>(
      context: context, builder: (context) => const
CustomAlertDialog());

  Future<bool> onBackPressed() async {
    final bool shouldPop = await showWarning(context) ?? false;

    return shouldPop;
  }

  @override
  Widget build(BuildContext context) {
    return WillPopScope(
      onWillPop: onBackPressed,
      child: Background(
        child: Stack(
          children: [
            Column(
              children: [
                Center(
                  child: CircularCountDownTimer(
                    duration: 45,
                    initialDuration: 45,
                    controller: controlador,
                    width: MediaQuery.of(context).size.width / 2,
                    height: MediaQuery.of(context).size.height / 2,
                    ringColor: Colors.grey[300]!,
                    ringGradient: null,
                    fillColor: Colors.purpleAccent[100]!,
```

```
                    fillGradient: null,
                    backgroundColor: Colors.purple[500],
                    backgroundGradient: null,
                    strokeWidth: 20.0,
                    strokeCap: StrokeCap.round,
                    textStyle: const TextStyle(
                        fontSize: 33.0,
                        color: Colors.white,
                        fontWeight: FontWeight.bold),
                    textFormat: CountdownTextFormat.S,
                    isReverse: true,
                    isReverseAnimation: true,
                    isTimerTextShown: true,
                    autoStart: false,
                    onStart: () {
                      debugPrint('Countdown Started');
                    },
                    onComplete: () {
                      debugPrint('Countdown Ended');
                      setState(() {
                        showAnswer = true;
                        response = false;
                      });
                      var turnonotifier =
ref.read(turnosProvider.notifier);
                      turnonotifier.pasarTurno();

ref.read(alertaTurnoProvider.notifier).showAlertaTurno();
                    },
                    onChange: (String timeStamp) {
                      debugPrint('Countdown Changed $timeStamp');
                    },
                  ),
                ),
                const SizedBox(
                  height: 40,
                ),
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    CircleButton(
                        icon: Icons.done,
                        iconSize: 100,
```

```
                        buttonFunction: () {
                          if (!showAnswer) {
                            // se para el tiempo
                            controlador.pause();

                            setState(() {
                              // saldra la respuesta que era correcta
                              showAnswer = true;
                              // para el color de fondo en la respiesta
correcta
                              response = true;
                            });
                            // sacamo el indice del equipo que ha hecho
la prueba gracias al provider del turno
                            var indiceTurno = ref.read(turnosProvider);

                            //sumamos un punto al equipo
                            var puntosNotifier =
                                ref.read(puntosProvider.notifier);
                            puntosNotifier.addPuntos(indiceTurno);

                            // esto pasara el turno al siguiente equipo
                            var turnonotifier =
                                ref.read(turnosProvider.notifier);
                            turnonotifier.pasarTurno();
                            // y esto enseña el mensaje
                            ref
                                .read(alertaTurnoProvider.notifier)
                                .showAlertaTurno();
                          }
                        },
                        buttonColor: Colors.greenAccent),
                    const SizedBox(
                      width: 30,
                    ),
                    CircleButton(
                      icon: Icons.close_rounded,
                      iconSize: 100,
                      buttonFunction: () {
                        if (!showAnswer) {
                          controlador.pause();
                          setState(() {
                            showAnswer = true;
```

```
                            response = false;
                          });
                          var turnonotifier =
ref.read(turnosProvider.notifier);
                          turnonotifier.pasarTurno();
                          ref
                              .read(alertaTurnoProvider.notifier)
                              .showAlertaTurno();
                        }
                      },
                      buttonColor: Colors.redAccent,
                    ),
                  ],
                )
              ],
            ),
            showAnswer
                ? Container(
                    padding: EdgeInsets.all(50),
                    margin: EdgeInsets.symmetric(horizontal: 50,
vertical: 230),
                    decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(20),
                        color: response ? Colors.green : Colors.red),
                    child: Column(
                      mainAxisAlignment: MainAxisAlignment.center,
                      crossAxisAlignment: CrossAxisAlignment.stretch,
                      children: [
                        Center(
                            child: Text(
                          'La respuesta correcta era ${widget.answer}',
                          textAlign: TextAlign.center,
                          style: const TextStyle(
                            color: Colors.white,
                            fontSize: 23,
                          ),
                        )),
                        const SizedBox(
                          height: 16,
                        ),
                        ElevatedButton(
                          onPressed: () {
                            Navigator.of(context).pop();
```

```
                          },
                          child: const Text(
                            'Continuar',
                            style: TextStyle(fontSize: 17),
                          ),
                        ),
                      ],
                    ))
                : const SizedBox.shrink()
          ],
        ),
      ),
    );
  }
}
```

PreAzul Screen

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:pruebas/widgets/background.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:pruebas/widgets/indicar_jugador.dart';

import '../screens/game/custom_alert_dialog.dart';

class PreAzul extends StatefulWidget {
  const PreAzul({
    super.key,
    required this.nombrePrueba,
    required this.player,
    required this.explicacion,
    required this.opcion1,
    required this.opcion2,
    required this.opcion3,
  });

  final String nombrePrueba;

  final String player;

  final String explicacion;
```

```
  final String opcion1;

  final String opcion2;

  final String opcion3;

  static const String routeName = "preazul";

  @override
  State<PreAzul> createState() => _PreAzulState();
}

class _PreAzulState extends State<PreAzul> {
  Future<bool?> showWarning(BuildContext context) async =>
showDialog<bool>(
      context: context, builder: (context) => const
CustomAlertDialog());

  Future<bool> onBackPressed() async {
    final bool shouldPop = await showWarning(context) ?? false;

    return shouldPop;
  }

  @override
  Widget build(BuildContext context) {
    return WillPopScope(
      onWillPop: onBackPressed,
      child: Background(
        child: Center(
            child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            Text(widget.nombrePrueba,
                textAlign: TextAlign.center,
                style: GoogleFonts.lato(
                  color: Colors.white,
                  fontSize: 40,
                )),
            const SizedBox(
              height: 16,
            ),
            const Text(
```

```
              '¿Cómo jugar?',
              style: TextStyle(color: Colors.white, fontSize: 20),
            ),
            const SizedBox(
              height: 16,
            ),
            Container(
                margin: const EdgeInsets.all(20),
                child: Center(
                  child: Text(
                    widget.explicacion,
                    textAlign: TextAlign.justify,
                    style: const TextStyle(
                      color: Colors.white,
                      fontSize: 20,
                    ),
                  ),
                )),
            const SizedBox(
              height: 16,
            ),
            IndicarJugador(
              player: widget.player,
              opcion1: widget.opcion1,
              opcion2: widget.opcion2,
              opcion3: widget.opcion3,
              verde: false,
            )
          ],
        )),
      ),
    );
  }
}
```

Prueba Azul

```
import 'dart:ui';

import
'package:circular_countdown_timer/circular_countdown_timer.dart';
import 'package:flutter/material.dart';
```

```dart
import 'package:flutter_colorpicker/flutter_colorpicker.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/widgets/background.dart';

import '../providers/alerta_turno_provider.dart';
import '../providers/puntos_provider.dart';
import '../providers/turno_provider.dart';
import '../screens/game/custom_alert_dialog.dart';
import '../widgets/circle_button.dart';

class DrawScreenDos extends ConsumerStatefulWidget {
  const DrawScreenDos({super.key, required this.answer});

  static const routeName = 'draw';
  final String answer;

  @override
  ConsumerState<DrawScreenDos> createState() {
    return _DrawScreenDosState();
  }
}

class _DrawScreenDosState extends ConsumerState<DrawScreenDos> {
  List<OffsetColor?> points = [];
  Color selectedColor = Colors.black;

  void selectColor() {
    showDialog(
      context: context,
      builder: (ctx) => AlertDialog(
        title: const Text('Color choser'),
        content: SingleChildScrollView(
          child: BlockPicker(
            pickerColor: selectedColor,
            onColorChanged: (color) {
              setState(() {
                selectedColor = color;
              });
            },
          ),
        ),
        actions: [
          TextButton(
```

```
              onPressed: () {
                Navigator.pop(ctx);
              },
              child: const Text('vale!'))
        ],
      ),
    );
  }

  @override
  void initState() {
    super.initState();

    WidgetsBinding.instance.addPostFrameCallback((timeStamp) {
      controlador.restart();
      //esto hace que justo despues de cargar el controller haga
restart
    });
  }

  bool response = false;
  bool showAnswer = false;
  var controlador = CountDownController();

  Future<bool?> showWarning(BuildContext context) async =>
showDialog<bool>(
      context: context, builder: (context) => const
CustomAlertDialog());

  Future<bool> onBackPressed() async {
    final bool shouldPop = await showWarning(context) ?? false;

    return shouldPop;
  }

  @override
  Widget build(BuildContext context) {
    final double width = MediaQuery.of(context).size.width;
    final double height = MediaQuery.of(context).size.height;

    return WillPopScope(
      onWillPop: onBackPressed,
      child: Background(
```

```
          child: Stack(children: [
        Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              Center(
                child: CircularCountDownTimer(
                  duration: 45,
                  initialDuration: 45,
                  controller: controlador,
                  width: MediaQuery.of(context).size.width * 0.15,
                  height: MediaQuery.of(context).size.height * 0.15,
                  ringColor: Colors.grey[300]!,
                  ringGradient: null,
                  fillColor: Colors.purpleAccent[100]!,
                  fillGradient: null,
                  backgroundColor: Colors.purple[500],
                  backgroundGradient: null,
                  strokeWidth: 20.0,
                  strokeCap: StrokeCap.round,
                  textStyle: const TextStyle(
                      fontSize: 33.0,
                      color: Colors.white,
                      fontWeight: FontWeight.bold),
                  textFormat: CountdownTextFormat.S,
                  isReverse: true,
                  isReverseAnimation: true,
                  isTimerTextShown: true,
                  autoStart: false,
                  onStart: () {
                    debugPrint('Countdown Started');
                  },
                  onComplete: () {
                    debugPrint('Countdown Ended');
                    setState(() {
                      showAnswer = true;
                      response = false;
                    });
                    var turnonotifier =
ref.read(turnosProvider.notifier);
                    turnonotifier.pasarTurno();
```

```
ref.read(alertaTurnoProvider.notifier).showAlertaTurno();
                  },
                  onChange: (String timeStamp) {
                    debugPrint('Countdown Changed $timeStamp');
                  },
                ),
              ),
              const SizedBox(
                height: 8,
              ),
              Container(
                width: width * 0.80,
                height: height * 0.60,
                decoration: BoxDecoration(
                  borderRadius: const BorderRadius.all(
                    Radius.circular(20.0),
                  ),
                  boxShadow: [
                    BoxShadow(
                      color: Colors.black.withOpacity(0.4),
                      blurRadius: 5.0,
                      spreadRadius: 1.0,
                    ),
                  ],
                ),
                child: GestureDetector(
                  onPanDown: (details) {
                    setState(() {
                      points.add(
                        OffsetColor(
                            offset: details.localPosition,
                            color: selectedColor),
                      );
                    });
                  },
                  onPanUpdate: (details) {
                    setState(() {
                      points.add(
                        OffsetColor(
                            offset: details.localPosition,
                            color: selectedColor),
                      );
```

```
                    });
                  },
                  onPanEnd: (details) {
                    setState(() {
                      points.add(null);
                    });
                  },
                  child: ClipRRect(
                    borderRadius: const
BorderRadius.all(Radius.circular(20.0)),
                    child: CustomPaint(
                      painter: MyCustomPainter(
                          points: points,
                          color: selectedColor,
                          strokeWidth: 5.0),
                    ),
                  ),
                ),
              ),
              SizedBox(
                height: height * 0.02,
              ),
              Container(
                width: width * 0.25,
                decoration: const BoxDecoration(
                  color: Colors.white,
                  borderRadius: BorderRadius.all(
                    Radius.circular(20.0),
                  ),
                ),
                child: Row(children: [
                  IconButton(
                    onPressed: selectColor,
                    icon: Icon(
                      Icons.color_lens,
                      color: selectedColor,
                    ),
                  ),
                  Expanded(
                    child: IconButton(
                      onPressed: () {
                        points.clear();
                      },
```

```
                    icon: const Icon(
                      Icons.layers_clear,
                    ),
                  ),
                ),
              ]),
            ),
            const SizedBox(
              height: 20,
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                CircleButton(
                    icon: Icons.done,
                    iconSize: 40,
                    buttonFunction: () {
                      if (!showAnswer) {
                        // se para el tiempo
                        controlador.pause();

                        setState(() {
                          // saldra la respuesta que era correcta
                          showAnswer = true;
                          // para el color de fondo en la respuesta
correcta
                          response = true;
                        });
                        // sacamo el indice del equipo que ha hecho
la prueba gracias al provider del turno
                        var indiceTurno = ref.read(turnosProvider);

                        //sumamos un punto al equipo
                        var puntosNotifier =
                            ref.read(puntosProvider.notifier);
                        puntosNotifier.addPuntos(indiceTurno);

                        // esto pasara el turno al siguiente equipo
                        var turnonotifier =
ref.read(turnosProvider.notifier);
                        turnonotifier.pasarTurno();
                        // y esto enseña el mensaje
                        ref
```

```
                            .read(alertaTurnoProvider.notifier)
                            .showAlertaTurno();
                      }
                    },
                    buttonColor: Colors.greenAccent),
                const SizedBox(
                  width: 30,
                ),
                CircleButton(
                  icon: Icons.close_rounded,
                  iconSize: 40,
                  buttonFunction: () {
                    if (!showAnswer) {
                      controlador.pause();
                      setState(() {
                        showAnswer = true;
                        response = false;
                      });
                      var turnonotifier =
ref.read(turnosProvider.notifier);
                      turnonotifier.pasarTurno();
                      ref
                          .read(alertaTurnoProvider.notifier)
                          .showAlertaTurno();
                    }
                  },
                  buttonColor: Colors.redAccent,
                ),
              ],
            )
          ],
        ),
      ),
      showAnswer
          ? Container(
              padding: EdgeInsets.all(50),
              margin: EdgeInsets.symmetric(horizontal: 50, vertical:
220),
              decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(20),
                  color: response ? Colors.green : Colors.red),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
```

```
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  Center(
                      child: Text(
                    'La respuesta correcta era ${widget.answer}',
                    textAlign: TextAlign.center,
                    style: const TextStyle(
                      color: Colors.white,
                      fontSize: 23,
                    ),
                  )),
                  SizedBox(
                    height: 16,
                  ),
                  ElevatedButton(
                    onPressed: () {
                      Navigator.of(context).pop();
                    },
                    child: Text(
                      'Continuar',
                      style: TextStyle(fontSize: 17),
                    ),
                  ),
                ],
              ))
          : SizedBox.shrink(),
    ])),
    );
  }
}

class OffsetColor {
  OffsetColor({required this.offset, required this.color});

  Offset offset;
  Color color;
}

class MyCustomPainter extends CustomPainter {
  MyCustomPainter({
    required this.points,
    required this.color,
    required this.strokeWidth,
```

```
  });
  List<OffsetColor?> points;
  Color color;
  double strokeWidth;

  @override
  void paint(Canvas canvas, Size size) {
    Paint background = Paint()..color = Colors.white;
    Rect rect = Rect.fromLTWH(0, 0, size.width, size.height);
    canvas.drawRect(rect, background);

    for (int x = 0; x < points.length - 1; x++) {
      if (points[x] != null && points[x + 1] != null) {
        Paint paint = Paint();
        paint.color = points[x]!.color;
        paint.strokeWidth = 3.0;
        paint.isAntiAlias = true;
        paint.strokeCap = StrokeCap.round;
        canvas.drawLine(points[x]!.offset, points[x + 1]!.offset,
paint);
      } else if (points[x] != null && points[x + 1] == null) {
        Paint paint = Paint();
        paint.color = points[x]!.color;
        paint.strokeWidth = 3.0;
        paint.isAntiAlias = true;
        paint.strokeCap = StrokeCap.round;
        canvas.drawPoints(PointMode.points, [points[x]!.offset],
paint);
      }
    }
  }

  @override
  bool shouldRepaint(covariant CustomPainter oldDelegate) {
    return true;
  }
}
```

Winner Screen

```
import 'package:flutter/material.dart';
import 'package:flutter/src/widgets/framework.dart';
import 'package:flutter/src/widgets/placeholder.dart';
```

```dart
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:pruebas/providers/teams_provider.dart';
import 'package:pruebas/providers/winner_provider.dart';
import 'package:pruebas/widgets/background.dart';

import 'package:font_awesome_flutter/font_awesome_flutter.dart';

class WinnerScreen extends ConsumerWidget {
  const WinnerScreen({super.key});

  static const routeName = 'winner';

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    var winnerTeamIndex = ref.watch(winnerProvider);
    var teamsNotifier = ref.watch(teamsNameProvider);

    return Background(
        child: Center(
      child: Column(
        children: [
          Text(
            '¡Winner!',
            style: GoogleFonts.lato(color: Colors.white, fontSize: 40),
          ),
          const SizedBox(
            height: 33,
          ),
          Text('Eres un verdadero erudito',
              style: GoogleFonts.lato(color: Colors.white, fontSize:
20)),
          const SizedBox(
            height: 33,
          ),
          const FaIcon(
            FontAwesomeIcons.crown,
            size: 170,
            color: Color.fromARGB(255, 219, 172, 0),
          ),
          const SizedBox(
            height: 20,
          ),
```

```
          Text(
            teamsNotifier[winnerTeamIndex],
            style: GoogleFonts.lato(color: Colors.white, fontSize: 30),
          ),
          const SizedBox(
            height: 15,
          ),
          const FaIcon(FontAwesomeIcons.peopleGroup),
          const SizedBox(
            height: 12,
          ),
          ElevatedButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: const Text('Volver al menú'),
          ),
        ],
      ),
    ));
  }
}
```

Widgets

Alerta turno

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/providers/alerta_turno_provider.dart';

class AlertaTurno extends ConsumerWidget {
  const AlertaTurno({
    super.key,
    required this.teamName,
  });

  final String teamName;

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    return Container(
      padding: const EdgeInsets.all(50),
```

```
      margin: const EdgeInsets.symmetric(horizontal: 50, vertical:
100),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(20),
        color: const Color.fromARGB(255, 64, 0, 75),
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          Center(
              child: Text(
            'Es el turno de $teamName',
            textAlign: TextAlign.center,
            style: const TextStyle(
              color: Colors.white,
              fontSize: 18,
            ),
          )),
          const SizedBox(
            height: 16,
          ),
          ElevatedButton(
            onPressed: () {
              ref.read(alertaTurnoProvider.notifier).showAlertaTurno();
            },
            child: Text('¡Vale!'),
          ),
        ],
      ),
    );
  }
}
```

Answer Button

```
import 'package:flutter/material.dart';
import 'package:pruebas/constants/palette.dart';

class AnswerButton extends StatelessWidget {
  const AnswerButton({
    super.key,
    required this.answerText,
```

```
    required this.onTap,
    this.color = Palette.answerButton,
  });

  final String answerText;
  final void Function() onTap;
  final Color color;

  @override
  Widget build(context) {
    return ElevatedButton(
      onPressed: onTap,
      style: ElevatedButton.styleFrom(
        padding: const EdgeInsets.symmetric(vertical: 10, horizontal:
33),
        backgroundColor: color,
        foregroundColor: Colors.white,
        shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10)),
      ),
      child: Text(
        answerText,
        textAlign: TextAlign.center,
      ),
    );
  }
}
```

Background

```
import 'package:flutter/material.dart';
import 'package:pruebas/constants/palette.dart';

class Background extends StatelessWidget {
  const Background({
    super.key,
    required this.child,
  });

  final Widget child;

  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      body: SafeArea(
        child: Container(
          decoration: const BoxDecoration(
            gradient: LinearGradient(
                colors: [Palette.fondoClaro, Palette.fondoOscuro],
                begin: Alignment.bottomLeft,
                end: Alignment.topRight),
          ),
          child: child,
        ),
      ),
    );
  }
}
```

Circle Button

```
import 'package:flutter/material.dart';

class CircleButton extends StatelessWidget {
  const CircleButton(
      {Key? key,
      required this.icon,
      required this.iconSize,
      this.iconColor = Colors.white,
      this.buttonColor = Colors.blueGrey,
      this.pressedButtonColor,
      required this.buttonFunction})
      : super(key: key);
  final IconData icon;
  final Color iconColor;
  final double iconSize;
  final Color buttonColor;
  final Color? pressedButtonColor;

  final Function() buttonFunction;

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: buttonFunction,
      style: ElevatedButton.styleFrom(
```

```
        foregroundColor: null,
        shape: const CircleBorder(),
        backgroundColor: buttonColor,
        padding: const EdgeInsets.all(4),
      ),
      child: Icon(
        icon,
        color: iconColor,
        size: iconSize,
      ),
    );
  }
}
```

Edit Player

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/constants/palette.dart';
import 'package:pruebas/providers/players_provider.dart';
import 'editar_nombre.dart';

class PlayerNameEditable extends ConsumerStatefulWidget {
  PlayerNameEditable({super.key, required this.teamIndex});

  int teamIndex;

  @override
  ConsumerState<PlayerNameEditable> createState() {
    return _PlayerNameEditableState();
  }
}

class _PlayerNameEditableState extends
ConsumerState<PlayerNameEditable> {
  void editarNombreSheet(int teamIndex, int playerIndex) {
    showModalBottomSheet(
      useSafeArea: true,
      isScrollControlled: true,
      context: context,
      builder: (ctx) => EditarNombre(
        onEditName: (String editName) {
```

```
          var playersNameNotifier =
ref.read(playersNameProvider.notifier);
          playersNameNotifier.editPlayer(teamIndex, playerIndex,
editName);
        },
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    var playersName = ref.watch(playersNameProvider);
    return Container(
      height: 300,
      child: ListView.builder(
        itemCount: playersName[widget.teamIndex].length,
        itemBuilder: (context, playerIndex) {
          return Row(
            children: [
              Container(
                width: 275,
                margin:
                    const EdgeInsets.symmetric(vertical: 10,
horizontal: 20),
                alignment: AlignmentDirectional.center,
                padding: const EdgeInsets.all(10),
                decoration: BoxDecoration(
                  color: Palette.myPurple,
                  borderRadius: BorderRadius.circular(20),
                ),
                child: Row(
                  children: [
                    Expanded(
                      child: Text(
                        playersName[widget.teamIndex][playerIndex],
                        style:
                            const TextStyle(color: Colors.white,
fontSize: 20),
                      ),
                    ),
                    IconButton(
                      onPressed: () {
```

```
                        editarNombreSheet(widget.teamIndex,
playerIndex);
                    },
                    icon: const Icon(
                      Icons.edit,
                      color: Colors.white,
                      size: 27,
                    ),
                  )
                ],
              ),
            ),
            IconButton(
              onPressed: () {
                var playersNameNotifier =
                    ref.read(playersNameProvider.notifier);
                playersNameNotifier.removePlayer(
                    widget.teamIndex, playerIndex);
              },
              icon: const Icon(
                Icons.delete,
                color: Colors.white,
                size: 27,
              ),
            ),
          ],
        );
      },
    ),
  );
 }
}
```

Editar Nombre (teams)

```
import 'package:flutter/material.dart';

//import 'editar_texto.dart';

class EditarNombre extends StatefulWidget {
  const EditarNombre({super.key, required this.onEditName});
```

```
  final void Function(String editName) onEditName;

  @override
  State<EditarNombre> createState() {
    return _EditarNombreState();
  }
}

class _EditarNombreState extends State<EditarNombre> {
  final _titleController = TextEditingController();

  void _submitExpenseData() {
    if (_titleController.text.trim().isEmpty) {
      showDialog(
        context: context,
        builder: (ctx) => AlertDialog(
          title: const Text(
            'Entrada no válida',
            style: TextStyle(color: Colors.black, fontSize: 18),
          ),
          content: const Text(
            'Introduce un nombre',
            style: TextStyle(color: Colors.black),
          ),
          actions: [
            TextButton(
                onPressed: () {
                  Navigator.pop(ctx);
                },
                child: const Text('Vale'))
          ],
        ),
      );
      return;
    }

    widget.onEditName(_titleController.text);
    Navigator.pop(context);
  }

  @override
  void dispose() {
    _titleController.dispose();
```

```
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    final keyboardSpace = MediaQuery.of(context).viewInsets.bottom;

    return LayoutBuilder(builder: (ctx, constraints) {
      return SizedBox(
        child: SingleChildScrollView(
          child: Padding(
            padding: EdgeInsets.fromLTRB(16, 16, 16, keyboardSpace +
16),
            child: Column(children: [
              Row(
                children: [
                  Expanded(
                    child: TextField(
                      style: const TextStyle(color: Colors.black),
                      controller: _titleController,
                      maxLength: 20,
                      decoration: const InputDecoration(
                        label: Text(
                          'Nombre',
                          style: TextStyle(fontSize: 15),
                        ),
                      ),
                    ),
                  ),
                  const SizedBox(
                    width: 8,
                  ),
                  //const Spacer(),
                  ElevatedButton(
                    onPressed: () {
                      Navigator.pop(context);
                    },
                    child: const Text(
                      'Cancelar',
                      style: TextStyle(fontSize: 13),
                    ),
                  ),
```

```
                  const SizedBox(
                    width: 4,
                  ),
                  ElevatedButton(
                    onPressed: _submitExpenseData,
                    child: const Text(
                      'Guardar Nombre',
                      style: TextStyle(fontSize: 13),
                    ),
                  )
                ],
              ),
            ]),
          ),
        ),
      );
    });
  }
}
```

Equipos

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/constants/palette.dart';
import 'package:pruebas/providers/teams_provider.dart';
import 'editar_nombre.dart';

class Equipos extends ConsumerStatefulWidget {
  Equipos({super.key, required this.constraints});

  BoxConstraints constraints;

  @override
  ConsumerState<Equipos> createState() {
    return _EquiposState();
  }
}

class _EquiposState extends ConsumerState<Equipos> {
  void editarNombreSheet(int index) {
    showModalBottomSheet(
      useSafeArea: true,
      isScrollControlled: true,
```

```
      context: context,
      builder: (ctx) => EditarNombre(
        onEditName: (String editName) {
          var teamsNameNotifier = ref.read(teamsNameProvider.notifier);
          teamsNameNotifier.editTeam(index, editName);
        },
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    var _nombreEquipos = ref.watch(teamsNameProvider);

    return Container(
      height: widget.constraints.maxHeight,
      child: ListView.builder(
        itemCount: _nombreEquipos.length,
        itemBuilder: (context, index) {
          return Row(
            children: [
              Container(
                width: 275,
                margin:
                    const EdgeInsets.symmetric(vertical: 10,
horizontal: 20),
                alignment: AlignmentDirectional.center,
                padding: const EdgeInsets.all(10),
                decoration: BoxDecoration(
                  color: Palette.myPurple,
                  borderRadius: BorderRadius.circular(20),
                ),
                child: Row(
                  children: [
                    Expanded(
                      child: Text(
                        _nombreEquipos[index],
                        style: Theme.of(context)
                            .textTheme
                            .bodyMedium!
                            .copyWith(fontSize: 20),
                      ),
                    ),
```

```
                    IconButton(
                      onPressed: () {
                        editarNombreSheet(index);
                      },
                      icon: const Icon(
                        Icons.edit,
                        color: Colors.white,
                      ),
                    )
                  ],
                ),
              ),
              IconButton(
                onPressed: () {
                  var teamsNameNotifier =
ref.read(teamsNameProvider.notifier);
                  teamsNameNotifier.removeTeam(index);
                },
                icon: const Icon(
                  Icons.delete,
                  color: Colors.white,
                ),
              ),
            ],
          );
        },
      ),
    );
  }
}
```

Indicar Jugador

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/constants/palette.dart';
import 'package:pruebas/providers/alerta_turno_provider.dart';
import 'package:pruebas/prueba_verde/prueba_verde.dart';
import 'package:pruebas/prueba_azul/draw_screen2.dart';

class IndicarJugador extends StatefulWidget {
  const IndicarJugador({
    super.key,
```

```
    required this.player,
    required this.opcion1,
    required this.opcion2,
    required this.opcion3,
    required this.verde,
  });

  final String player;
  final String opcion1;
  final String opcion2;
  final String opcion3;
  final bool verde;

  @override
  State<IndicarJugador> createState() => _IndicarJugadorState();
}

class _IndicarJugadorState extends State<IndicarJugador> {
  bool showOptions = false;

  @override
  Widget build(
    BuildContext context,
  ) {
    return Container(
        padding: const EdgeInsets.all(50),
        margin: const EdgeInsets.symmetric(horizontal: 50, vertical:
25),
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(20),
          color: const Color.fromARGB(255, 64, 0, 75),
        ),
        child: showOptions
            ? Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  const Center(
                      child: Text(
                    'Elige qué opción prefieres',
                    textAlign: TextAlign.center,
                    style: TextStyle(
                      color: Colors.white,
```

```
                    fontSize: 23,
                  ),
                )),
                const SizedBox(
                  height: 16,
                ),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Palette.answerButton,
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10)),
                  ),
                  onPressed: () {
                    if (widget.verde) {

Navigator.of(context).pushReplacement(MaterialPageRoute(
                        builder: ((context) => PruebaVerde(
                              answer: widget.opcion1,
                            )),
                      ));
                    } else {

Navigator.of(context).pushReplacement(MaterialPageRoute(
                        builder: ((context) => DrawScreenDos(
                              answer: widget.opcion1,
                            )),
                      ));
                    }
                  },
                  child: Text(widget.opcion1),
                ),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Palette.answerButton,
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10)),
                  ),
                  onPressed: () {
                    if (widget.verde) {

Navigator.of(context).pushReplacement(MaterialPageRoute(
                        builder: ((context) => PruebaVerde(
                              answer: widget.opcion2,
```

```
                            )),
                      ));
                    } else {

Navigator.of(context).pushReplacement(MaterialPageRoute(
                          builder: ((context) => DrawScreenDos(
                                answer: widget.opcion2,
                            )),
                      ));
                    }
                  },
                  child: Text(widget.opcion2),
                ),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Palette.answerButton,
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10)),
                  ),
                  onPressed: () {
                    if (widget.verde) {

Navigator.of(context).pushReplacement(MaterialPageRoute(
                          builder: ((context) => PruebaVerde(
                                answer: widget.opcion3,
                            )),
                      ));
                    } else {

Navigator.of(context).pushReplacement(MaterialPageRoute(
                          builder: ((context) => DrawScreenDos(
                                answer: widget.opcion3,
                            )),
                      ));
                    }
                  },
                  child: Text(widget.opcion3),
                ),
              ],
            )
          : Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.stretch,
```

```
                children: [
                  Center(
                      child: Text(
                    'Le toca a ${widget.player}',
                    textAlign: TextAlign.center,
                    style: const TextStyle(
                      color: Colors.white,
                      fontSize: 23,
                    ),
                  )),
                  const SizedBox(
                    height: 16,
                  ),
                  ElevatedButton(
                    onPressed: () {
                      setState(() {
                        showOptions = true;
                      });
                    },
                    child: Text('Soy yo'),
                  ),
                ],
              ));
  }
}
```

Scoreboard

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:pruebas/providers/puntos_provider.dart';
import 'package:pruebas/providers/teams_provider.dart';
import 'package:pruebas/providers/turno_provider.dart';

class GameScoreboard extends ConsumerStatefulWidget {
  GameScoreboard({
    super.key,
    required this.numRondas,
    required this.index,
  });

  final int numRondas;
```

```
  final int index;

  @override
  ConsumerState<GameScoreboard> createState() {
    return _GameScoreboardState();
  }
}

class _GameScoreboardState extends ConsumerState<GameScoreboard> {
  @override
  Widget build(BuildContext context) {
    var teamName = ref.watch(teamsNameProvider)[widget.index];
    var puntos = ref.watch(puntosProvider)[widget.index];
    var turno = ref.watch(turnosProvider);

    return Container(
      padding: const EdgeInsets.symmetric(horizontal: 40, vertical:
20),
      decoration: BoxDecoration(
          color: turno == widget.index
              ? const Color.fromARGB(229, 160, 71, 234)
              : const Color.fromARGB(146, 82, 9, 106)),
      child: Column(
        children: [
          Text(teamName),
          const SizedBox(
            height: 6,
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              for (int i = 0; i < widget.numRondas; ++i)
                i < puntos
                    ? const Icon(
                        Icons.star,
                        size: 15,
                      )
                    : const Icon(
                        Icons.star_border,
                        size: 15,
                      ),
            ],
```

```
          ),
          const FaIcon(FontAwesomeIcons.peopleGroup),
        ],
      ),
    );
  }
}
```

Wheel

```
import 'dart:async';
import 'dart:math';

import 'package:flutter/material.dart';
import 'package:flutter_fortune_wheel/flutter_fortune_wheel.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/providers/players_provider.dart';
import 'package:pruebas/providers/turno_provider.dart';
import 'package:pruebas/prueba_azul/preazul.dart';
//import 'package:pruebas/screens/draw_screen2.dart';
import 'package:pruebas/screens/draw_screen.dart';
import 'package:pruebas/prueba_azul/draw_screen2.dart';
import 'package:pruebas/screens/questions_screen.dart';
import 'package:pruebas/services/db_helper.dart';

import '../prueba_verde/preverde.dart';

class Wheel extends ConsumerStatefulWidget {
  const Wheel({super.key});

  @override
  ConsumerState<Wheel> createState() => _WheelState();
}

class _WheelState extends ConsumerState<Wheel> {
  StreamController<int> controller = StreamController<int>();

  var nextRound = -1;

  List<int> generateUniqueRandomNumbers(int count, int min, int max) {
    if (count > (max - min + 1) || max < min) {
      throw ArgumentError("Invalid arguments");
```

```
    }

    List<int> randomNumbers = [];
    Random random = Random();

    while (randomNumbers.length < count) {
      int randomNumber = min + random.nextInt(max - min + 1);
      if (!randomNumbers.contains(randomNumber)) {
        randomNumbers.add(randomNumber);
      }
    }

    return randomNumbers;
  }

  Future<Widget> generarPrueba() async {
    var nextWidget;
    //nextRound = 1;
    if (nextRound == 0) {
      // Dato Nauta
      var nextPrueba = Random().nextInt(3);
      if (nextPrueba == 0) {
        var maxID = await DBHelper.getMaxID('questions4');
        List<int> randomQuestions = generateUniqueRandomNumbers(1, 1,
maxID);
        var question = await
DBHelper.getQuestions4(randomQuestions[0]);
        nextWidget = QuestionsScreen(
          question: question!.questionES,
          answer1: question.answer1ES,
          answer2: question.answer2ES,
          answer3: question.answer3ES,
          answer4: question.answer4ES,
        );
      } else if (nextPrueba == 1) {
        var maxID = await DBHelper.getMaxID('questions4');
        List<int> randomQuestions = generateUniqueRandomNumbers(1, 1,
maxID);
        var question = await
DBHelper.getQuestions4(randomQuestions[0]);
        nextWidget = QuestionsScreen(
          question: question!.questionES,
          answer1: question.answer1ES,
```

```
          answer2: question.answer2ES,
          answer3: question.answer3ES,
          answer4: question.answer4ES,
        );
      } else {
        var maxID = await DBHelper.getMaxID('questions4');
        List<int> randomQuestions = generateUniqueRandomNumbers(1, 1,
maxID);
        var question = await
DBHelper.getQuestions4(randomQuestions[0]);
        nextWidget = QuestionsScreen(
          question: question!.questionES,
          answer1: question.answer1ES,
          answer2: question.answer2ES,
          answer3: question.answer3ES,
          answer4: question.answer4ES,
        );
      }
    } else if (nextRound == 1) {
      // Gato Creativo
      var nextPrueba = Random().nextInt(3);
      var turnoEquipo = ref.read(turnosProvider);
      var players = ref.read(playersNameProvider);
      var numPlayers = players[turnoEquipo].length;

      var playerIndex = Random().nextInt(numPlayers);
      var playerName = players[turnoEquipo][playerIndex];

      var maxID = await DBHelper.getMaxID('draw');

      List<int> randomQuestions = generateUniqueRandomNumbers(3, 1,
maxID);
      var question1 = await DBHelper.getDraw(randomQuestions[0]);
      var question2 = await DBHelper.getDraw(randomQuestions[1]);
      var question3 = await DBHelper.getDraw(randomQuestions[2]);

      if (nextPrueba == 0) {
        nextWidget = PreAzul(
          nombrePrueba: 'Dibujar',
          player: playerName,
          explicacion:
```

```
              'En esta prueba deberás dibujar una de las opciones para
que tus compañeros la adivinen.\nSi consiguen adivinarlo antes de que
finalice el tiempo, debéis darle al botón verde',
          opcion1: question1!.drawES,
          opcion2: question2!.drawES,
          opcion3: question3!.drawES,
        );
      } else if (nextPrueba == 1) {
        nextWidget = PreAzul(
          nombrePrueba: 'Dibujar con los ojos cerrados',
          player: playerName,
          explicacion:
              'En esta prueba deberás dibujar con los ojos cerrados una
de las opciones para que tus compañeros la adivinen.\nSi consiguen
adivinarlo antes de que finalice el tiempo, debéis darle al botón
verde',
          opcion1: question1!.drawES,
          opcion2: question2!.drawES,
          opcion3: question3!.drawES,
        );
      } else {
        nextWidget = PreAzul(
          nombrePrueba: 'Dibujar con la mano "mala"',
          player: playerName,
          explicacion:
              'En esta prueba deberás dibujar con la mano "mala" una de
las opciones para que tus compañeros la adivinen.\nSi consiguen
adivinarlo antes de que finalice el tiempo, debéis darle al botón
verde',
          opcion1: question1!.drawES,
          opcion2: question2!.drawES,
          opcion3: question3!.drawES,
        );
      }
    } else if (nextRound == 2) {
      //star stelar
      var nextPrueba = Random().nextInt(3);
      var turnoEquipo = ref.read(turnosProvider);
      var players = ref.read(playersNameProvider);
      var numPlayers = players[turnoEquipo].length;

      var playerIndex = Random().nextInt(numPlayers);
      var playerName = players[turnoEquipo][playerIndex];
```

```
      if (nextPrueba == 0) {
        var maxID = await DBHelper.getMaxID('impersonate');

        List<int> randomQuestions = generateUniqueRandomNumbers(3, 1,
maxID);
        var question1 = await DBHelper.getImiton(randomQuestions[0]);
        var question2 = await DBHelper.getImiton(randomQuestions[1]);
        var question3 = await DBHelper.getImiton(randomQuestions[2]);

        nextWidget = PreVerde(
          nombrePrueba: 'Imitón',
          explicacion:
              'En esta prueba debes imitar a un famoso para que tus
compañeros lo adivinen.\n Puedes hablar, pero no puedes decir el nombre
de esta persona.\nSi consiguen adivinarlo antes de que finalice el
tiempo, debéis darle al botón verde.',
          player: playerName,
          opcion1: question1!.impersonateES,
          opcion2: question2!.impersonateES,
          opcion3: question3!.impersonateES,
        );
      } else if (nextPrueba == 1) {
        var maxID = await DBHelper.getMaxID('sing');

        List<int> randomQuestions = generateUniqueRandomNumbers(3, 1,
maxID);
        var question1 = await DBHelper.getSing(randomQuestions[0]);
        var question2 = await DBHelper.getSing(randomQuestions[1]);
        var question3 = await DBHelper.getSing(randomQuestions[2]);
        nextWidget = PreVerde(
          nombrePrueba: 'Tarasilba',
          explicacion:
              'En esta prueba debes tararear o silbar una canción para
que tus compañeros la adivinen.\nSi consiguen adivinarla antes de que
finalice el tiempo, debéis darle al botón verde.',
          opcion1: question1!.singES,
          opcion2: question2!.singES,
          opcion3: question3!.singES,
          player: playerName,
        );
      } else {
        var maxID = await DBHelper.getMaxID('mimic');
```

```
        List<int> randomQuestions = generateUniqueRandomNumbers(3, 1,
maxID);
        var question1 = await DBHelper.getMimica(randomQuestions[0]);
        var question2 = await DBHelper.getMimica(randomQuestions[1]);
        var question3 = await DBHelper.getMimica(randomQuestions[2]);
        nextWidget = PreVerde(
          nombrePrueba: 'Mimica',
          explicacion:
              'En esta prueba debes imitar o representar una acción,
oficio o cosa para que tus compañeros lo adivinen.\nNo puedes hablar,
solamente hacer gestos.\nSi consiguen adivinarlo antes de que finalice
el tiempo, debéis darle al botón verde.',
          player: playerName,
          opcion1: question1!.mimicES,
          opcion2: question2!.mimicES,
          opcion3: question3!.mimicES,
        );
      }
    }
    await Future.delayed(
      Duration(seconds: 1),
    );
    return nextWidget;
  }

  @override
  Widget build(BuildContext context) {
    return FortuneWheel(
      animateFirst: false,
      selected: controller.stream,
      onAnimationEnd: () async {
        var nextRoute = await generarPrueba();
        Navigator.of(context).push(
          MaterialPageRoute(
            builder: ((context) => nextRoute),
          ),
        );
        //var result = await DBHelper.getData('questions4');
        //print(result);
      },
      onFling: () {
        setState(() {
```

```
          nextRound = Random().nextInt(3);
          controller.add(nextRound);
          print(nextRound);
        });
      },
      indicators: const <FortuneIndicator>[
        FortuneIndicator(
          alignment: Alignment.topCenter,
          child: TriangleIndicator(color: Colors.deepPurple),
        ),
      ],
      items: const [
        FortuneItem(
          child: Text('Dato Nauta'),
          style: FortuneItemStyle(color: Colors.red),
        ),
        FortuneItem(
          child: Text('Gato Creativo'),
          style: FortuneItemStyle(color: Colors.blue),
        ),
        FortuneItem(
          child: Text('Star Estelar'),
          style: FortuneItemStyle(color: Colors.green),
        ),
      ],
    );
  }
}
```

Providers (Riverpod)

Alerta Turno Provider

```
import 'package:flutter_riverpod/flutter_riverpod.dart';

class AlertaTurnoNotifier extends StateNotifier<bool> {
  AlertaTurnoNotifier() : super(true);

  void showAlertaTurno() {
    state = !state;
  }
}

final alertaTurnoProvider =
```

```
    StateNotifierProvider<AlertaTurnoNotifier, bool>((ref) {
  return AlertaTurnoNotifier();
});
```

Num Rondas Provider

```
import 'package:flutter_riverpod/flutter_riverpod.dart';

class NumRondasNotifier extends StateNotifier<int> {
  NumRondasNotifier() : super(3);

  void editNumRondas(int rondas) {
    state = rondas;
  }
}

final numRondasProvider = StateNotifierProvider<NumRondasNotifier,
int>((ref) {
  return NumRondasNotifier();
});
```

Players Provider

```
import 'package:flutter_riverpod/flutter_riverpod.dart';

class PlayersNameNotifier extends StateNotifier<List<List<String>>> {
  PlayersNameNotifier() : super([]);

  void initializeLists(int numTeams) {
    List<List<String>> result = [];
    for (int i = 0; i < numTeams; i++) {
      result.add(["Jugador1", "Jugador2"]);
    }

    state = result;
  }

  void addPlayer(String playerName, int indexTeam) {
    List<List<String>> newState = [];
    for (int i = 0; i < state.length; ++i) {
      newState.add([]);
      for (int j = 0; j < state[i].length; ++j) {
        newState[i].add(state[i][j]);
```

```
      }
    }

    newState[indexTeam].add(playerName);
    state = newState;
  }

  void removePlayer(int teamIndex, int playerIndex) {
    List<List<String>> newState = [];
    for (int i = 0; i < state.length; ++i) {
      newState.add([]);
      for (int j = 0; j < state[i].length; ++j) {
        newState[i].add(state[i][j]);
      }
    }
    newState[teamIndex].removeAt(playerIndex);
    state = newState;
  }

  void editPlayer(int teamIndex, int playerIndex, String newName) {
    List<List<String>> newState = [];
    for (int i = 0; i < state.length; ++i) {
      newState.add([]);
      for (int j = 0; j < state[i].length; ++j) {
        newState[i].add(state[i][j]);
      }
    }
    newState[teamIndex][playerIndex] = newName;
    state = newState;
  }
}

final playersNameProvider =
    StateNotifierProvider<PlayersNameNotifier,
List<List<String>>>((ref) {
  return PlayersNameNotifier();
});
```

Puntos Provider

```
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/providers/num_rondas_provider.dart';
import 'package:pruebas/providers/winner_provider.dart';
```

```
class PuntosNotifier extends StateNotifier<List<int>> {
  PuntosNotifier(
      {required this.numRondasNotifier, required this.winnerNotifier})
      : super([]);

  NumRondasNotifier numRondasNotifier;
  WinnerNotifier winnerNotifier;

  void initializeLists(int numTeams) {
    List<int> result = [];
    for (int i = 0; i < numTeams; i++) {
      result.add(0);
    }

    state = result;
  }

  void addPuntos(int index) {
    List<int> newState = [];
    for (int s in state) {
      newState.add(s);
    }
    var numPuntos = newState[index] + 1;
    if (numPuntos == numRondasNotifier.state) {
      newState[index]++;
      //aqui ha ganado un equipo
      winnerNotifier.setWinner(index);
    } else {
      newState[index]++;
    }

    state = newState;
  }
}

final puntosProvider = StateNotifierProvider<PuntosNotifier,
List<int>>((ref) {
  var numRondasNotifier = ref.watch(numRondasProvider.notifier);
  var winnerNotifier = ref.watch(winnerProvider.notifier);

  return PuntosNotifier(
```

```
      numRondasNotifier: numRondasNotifier, winnerNotifier:
winnerNotifier);
});
```

Teams Provider

```
import 'package:flutter_riverpod/flutter_riverpod.dart';

class TeamsNameNotifier extends StateNotifier<List<String>> {
  TeamsNameNotifier()
      : super([
          'Equipo 1',
          'Equipo 2',
        ]);

  void addTeam(String teamName) {
    state = [...state, teamName];
  }

  void removeTeam(int index) {
    List<String> newState = [];
    for (String s in state) {
      newState.add(s);
    }
    //copia del state
    newState.removeAt(index);
    state = newState;
  }

  void editTeam(int index, String newName) {
    List<String> newState = [];
    for (String s in state) {
      newState.add(s);
    }
    newState[index] = newName;
    state = newState;
  }
}

final teamsNameProvider =
    StateNotifierProvider<TeamsNameNotifier, List<String>>((ref) {
  return TeamsNameNotifier();
});
```

Turno Provider

```dart
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:pruebas/providers/teams_provider.dart';

class TurnosNotifier extends StateNotifier<int> {
  TurnosNotifier({required this.teamsNotifier}) : super(0);

  final TeamsNameNotifier teamsNotifier;

  void pasarTurno() {
    if ((state + 1) == teamsNotifier.state.length) {
      state = 0;
    } else {
      state++;
    }
  }
}

final turnosProvider = StateNotifierProvider<TurnosNotifier, int>((ref)
{
  var teamsNotifier = ref.watch(teamsNameProvider.notifier);
  return TurnosNotifier(teamsNotifier: teamsNotifier);
});
```

Winner Provider

```dart
import 'package:flutter_riverpod/flutter_riverpod.dart';

class WinnerNotifier extends StateNotifier<int> {
  WinnerNotifier() : super(-1);

  void setWinner(int indexTeam) {
    state = indexTeam;
  }
}

final winnerProvider = StateNotifierProvider<WinnerNotifier, int>((ref)
{
  return WinnerNotifier();
});
```

Constants

Palette

```dart
import 'package:flutter/material.dart';

class Palette {
  static const Color answerButton = Color.fromARGB(235, 206, 122, 255);
  static const Color fondoOscuro = Color.fromARGB(255, 53, 4, 116);
  static const Color fondoClaro = Color.fromARGB(255, 136, 63, 214);
  static const Color myPurple = Color.fromARGB(255, 58, 27, 91);
}
```

Models

Draw

```dart
import 'package:flutter/material.dart';

class Palette {
  static const Color answerButton = Color.fromARGB(235, 206, 122, 255);
  static const Color fondoOscuro = Color.fromARGB(255, 53, 4, 116);
  static const Color fondoClaro = Color.fromARGB(255, 136, 63, 214);
  static const Color myPurple = Color.fromARGB(255, 58, 27, 91);
}
```

Imiton

```dart
class Imiton {
  final int questionID;
  final String impersonateES;
  Imiton({required this.questionID, required this.impersonateES});

  factory Imiton.fromMap(Map<String, dynamic> data,
      {String languageCode = 'ES'}) {
    if (data == null) {
      throw StateError('Missing Data');
    }
    return Imiton(
        questionID: data['questionID'],
        impersonateES: data['impersonate$languageCode']);
  }
}
```

Mimica

```
class Mimica {
  final int questionID;
  final String mimicES;
  Mimica({required this.questionID, required this.mimicES});

  // una funcion

  factory Mimica.fromMap(Map<String, dynamic> data,
      {String languageCode = 'ES'}) {
    if (data == null) {
      throw StateError('Missing Data');
    }
    return Mimica(
        questionID: data['questionID'], mimicES:
data['mimic$languageCode']);
  }
}
```

Questions4

```
class Questions4 {
  final int questionID;
  final String questionES;
  final String answer1ES;
  final String answer2ES;
  final String answer3ES;
  final String answer4ES;
  Questions4({
    required this.questionID,
    required this.questionES,
    required this.answer1ES,
    required this.answer2ES,
    required this.answer3ES,
    required this.answer4ES,
  });

  factory Questions4.fromMap(Map<String, dynamic> data,
      {String languageCode = 'ES'}) {
    if (data == null) {
      throw StateError('Missing Data');
    }
    return Questions4(
      questionID: data['questionID'],
```

```
      questionES: data['question$languageCode'],
      answer1ES: data['answer1$languageCode'],
      answer2ES: data['answer2$languageCode'],
      answer3ES: data['answer3$languageCode'],
      answer4ES: data['answer4$languageCode'],
    );
  }
}
```

Sing

```
class Sing {
  final int questionID;
  final String singES;
  Sing({required this.questionID, required this.singES});

  factory Sing.fromMap(Map<String, dynamic> data,
      {String languageCode = 'ES'}) {
    if (data == null) {
      throw StateError('Missing Data');
    }
    return Sing(
        questionID: data['questionID'], singES:
data['sing$languageCode']);
  }
}
```

Assets

Dins d'aquest apartat va la base de dades i una foto (el logo de l'aplicació)

Services

db_helper

```
import 'dart:io';
// import 'dart:typed_data';

import 'package:flutter/services.dart';
// ignore: depend_on_referenced_packages
import 'package:path/path.dart';
import 'package:pruebas/models/imiton.dart';
import 'package:pruebas/models/sing.dart';
import 'package:sqflite/sqflite.dart' as sql;

import 'package:logging/logging.dart';
```

```dart
import '../models/draw.dart';
import '../models/mimica.dart';
import '../models/questions4.dart';

final logging = Logger('SQLiteDB');

class DBHelper {
  static Future<void> copyDB() async {
    // Construct a file path to copy database to
    final dbPath = await sql.getDatabasesPath();
    String path = join(dbPath, "eruditio.db");

    // Only copy if the database doesn't exist
    if (FileSystemEntity.typeSync(path) ==
FileSystemEntityType.notFound) {
      logging.info('Copy db file');
      // Load database from asset and copy
      ByteData data = await rootBundle.load(join('assets/db/',
'eruditio.db'));
      List<int> bytes =
          data.buffer.asUint8List(data.offsetInBytes,
data.lengthInBytes);

      // Save copied asset to documents
      await File(path).writeAsBytes(bytes);
    } else {
      logging.info('File already exists');
      //await sql.deleteDatabase(path);
    }
  }

  static Future<sql.Database> database() async {
    logging.info("INICIO BIEN");
    final dbPath = await sql.getDatabasesPath();
    return sql.openDatabase(join(dbPath, 'eruditio.db'));
  }

  static Future<void> insertPlayer(Map<String, Object> data) async {
    final db = await DBHelper.database();
    logging.info(data);
    db.insert(
      'players',
```

```
      data,
      conflictAlgorithm: sql.ConflictAlgorithm.replace,
    );
    logging.info("Tudoben");
  }

  static Future<void> updatePlayer(Map<String, Object> data) async {
    final db = await DBHelper.database();
    logging.info(data);
    db.update(
      'players',
      data,
      conflictAlgorithm: sql.ConflictAlgorithm.replace,
    );
    logging.info("Tudoben");
  }

  static Future<List<Map<String, dynamic>>> getData(String table) async
{
    final db = await DBHelper.database();
    return db.query(table);
  }

  static Future<int> getMaxID(String table) async {
    final db = await DBHelper.database();

    // Query the database to get the maximum value from the questionID
column
    List<Map<String, dynamic>> queryResult = await db
        .rawQuery('SELECT max(questionID) as maxQuestionID FROM
$table');

    // Extract and return the maximum questionID value
    int maxQuestionID = queryResult[0]['maxQuestionID'];
    return maxQuestionID;
  }

  static Future<Imiton?> getImiton(int questionID) async {
    final db = await DBHelper.database();

    List<Map<String, dynamic>> results = await db
        .rawQuery('SELECT * FROM impersonate WHERE questionID=?',
[questionID]);
```

```
    List<Imiton> questions = List.generate(results.length, (i) {
      return Imiton.fromMap(results[i]);
    });
    if (questions.isEmpty) {
      logging.info("Question not found");
      return null;
    } else {
      logging.info(questions[0].impersonateES);
      return questions[0];
    }
  }

  static Future<Mimica?> getMimica(int questionID) async {
    final db = await DBHelper.database();

    // La lista eran las rows, las strings son el nombre de las
columnas y los objects la info de las columnas
    // lo del await espera hasta que el future se resuelva

    List<Map<String, dynamic>> results = await db
        .rawQuery('SELECT * FROM mimic WHERE questionID=?',
[questionID]);

    //tenia una lista de Map<String, dynamic> y la convierta en una
lista de Mimica

    List<Mimica> questions = List.generate(results.length, (i) {
      //conviertes el map<String, object> en un Mimica
      return Mimica.fromMap(results[i]);
    });
    if (questions.isEmpty) {
      logging.info("Question not found");
      return null;
    } else {
      logging.info(questions[0].mimicES);
      return questions[0];
      //te retorna la questions de indice 0 pq la questionID es unica y
lo del select arriba solo dara una opcion (indice 0, primer elemento de
la lista)
    }
  }
```

```
  static Future<Sing?> getSing(int questionID) async {
    final db = await DBHelper.database();

    List<Map<String, dynamic>> results = await db
        .rawQuery('SELECT * FROM sing WHERE questionID=?',
[questionID]);

    List<Sing> questions = List.generate(results.length, (i) {
      return Sing.fromMap(results[i]);
    });
    if (questions.isEmpty) {
      logging.info("Question not found");
      return null;
    } else {
      logging.info(questions[0].singES);
      return questions[0];
    }
  }

  static Future<Questions4?> getQuestions4(int questionID) async {
    final db = await DBHelper.database();

    List<Map<String, dynamic>> results = await db
        .rawQuery('SELECT * FROM questions4 WHERE questionID=?',
[questionID]);

    List<Questions4> questions = List.generate(results.length, (i) {
      return Questions4.fromMap(results[i]);
    });
    if (questions.isEmpty) {
      logging.info("Question not found");
      return null;
    } else {
      logging.info(questions[0].questionES);
      return questions[0];
    }
  }

  static Future<Draw?> getDraw(int questionID) async {
    final db = await DBHelper.database();

    List<Map<String, dynamic>> results = await db
```

```
        .rawQuery('SELECT * FROM draw WHERE questionID=?',
[questionID]);

    List<Draw> questions = List.generate(results.length, (i) {
      return Draw.fromMap(results[i]);
    });
    if (questions.isEmpty) {
      logging.info("Question not found");
      return null;
    } else {
      logging.info(questions[0].drawES);
      return questions[0];
    }
  }
}
```

Main

```
import 'package:flutter/material.dart';
import 'package:logging/logging.dart';
import 'package:pruebas/app.dart';
import 'package:pruebas/services/db_helper.dart';

Future<void> main() async {
  Logger.root.level = Level.ALL; // defaults to Level.INFO
  Logger.root.onRecord.listen((record) {
    // ignore: avoid_print
    print(
        '${record.loggerName} ${record.level.name}: ${record.time}:
${record.message} ');
  });

  WidgetsFlutterBinding.ensureInitialized();
  await DBHelper.copyDB();
  runApp(const Quiz());
}
```

App

```
import 'package:flutter/material.dart';
import 'package:pruebas/prueba_verde/preverde.dart';
import 'package:pruebas/screens/config_partida/config_team.dart';
import 'package:pruebas/screens/game/game_screen.dart';
```

```dart
import 'package:pruebas/screens/primeras/settings_screen.dart';
import 'package:pruebas/screens/questions_screen.dart';
import 'package:pruebas/screens/winner_screen.dart';
import 'package:pruebas/widgets/equipos.dart';
import './screens/primeras/start_screen.dart';
import 'package:pruebas/prueba_azul/draw_screen2.dart';
import 'package:pruebas/screens/primeras/creditos_screen.dart';
import 'package:pruebas/screens/config_partida/config.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class Quiz extends StatefulWidget {
  const Quiz({super.key});

  @override
  State<Quiz> createState() {
    return _QuizState();
  }
}

class _QuizState extends State<Quiz> {
  @override
  Widget build(BuildContext context) {
    return ProviderScope(
      child: MaterialApp(
        home: StartScreen(),
        routes: {
          ConfiguracionTeamsScreen.routeName: (context) =>
              ConfiguracionTeamsScreen(),
          SettingsScreen.routeName: (context) => SettingsScreen(),
          ConfiguracionScreen.routeName: (context) =>
ConfiguracionScreen(),
          Creditos.routeName: (context) => Creditos(),
          GameScreen.routeName: (context) => GameScreen(),
          //DrawScreenDos.routeName: (context) => DrawScreenDos(),
          WinnerScreen.routeName: (context) => WinnerScreen(),
          StartScreen.routeName: (context) => StartScreen(),
        },
        theme: ThemeData(
          textTheme: const TextTheme(
            bodyMedium: TextStyle(
                fontSize: 16,
                fontFamily: "Lato",
                fontWeight: FontWeight.normal,
```

```
            color: Colors.white),
        bodyLarge: TextStyle(
            fontSize: 18,
            fontFamily: "Lato",
            fontWeight: FontWeight.normal,
            color: Colors.white),
        titleMedium: TextStyle(
            fontSize: 16,
            fontFamily: "Lato",
            fontWeight: FontWeight.bold,
            color: Colors.white),
        //botones por defecto
        labelLarge: TextStyle(
            fontSize: 18,
            fontFamily: "Lato",
            fontWeight: FontWeight.bold,
            color: Colors.white),
        titleLarge: TextStyle(
            fontSize: 24,
            fontFamily: "Lato",
            fontWeight: FontWeight.w900,
            color: Colors.white),
        headlineSmall: TextStyle(
            fontSize: 30,
            fontFamily: "Lato",
            fontWeight: FontWeight.w900,
            color: Colors.white),
        headlineMedium: TextStyle(
            fontSize: 40,
            fontFamily: "Lato",
            fontWeight: FontWeight.bold,
            color: Colors.white),
        displaySmall: TextStyle(
            fontSize: 48,
            fontFamily: "Lato",
            fontWeight: FontWeight.bold,
            color: Colors.white),
        displayMedium: TextStyle(
            fontSize: 60,
            fontFamily: "Lato",
            fontWeight: FontWeight.w900,
            color: Colors.white),
    ),
```

```
          elevatedButtonTheme: ElevatedButtonThemeData(
            style: ElevatedButton.styleFrom(
              elevation: 5.0,
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10),
              ),
            ),
          ),
          textButtonTheme: TextButtonThemeData(style:
TextButton.styleFrom()),
        ),
      ),
    );
  }
}
```