

# Programació de Dispositius Mòbils

*Projecte Final - Aplicació mòbil iOS*

**WikiFilms**

**Escola Tècnica Superior d'Enginyeria**

**La Salle - Universitat Ramon Llull**

25 de gener de 2026

	<b>Nom i Cognoms</b>
<b>Alumne</b>	Nil Bagaria Nofre (nil.bagaria@students.salle.url.edu) Alvaro Bello Garrido (alvaro.bello@students.salle.url.edu)
<b>Grup</b>	Grup 08

## Taula de continguts

<b>1. Introducció al projecte.....</b>	<b>2</b>
<b>2. SDKs utilitzats.....</b>	<b>3</b>
<b>2.1 UIKit.....</b>	<b>3</b>
<b>2.2 Firebase Authentication i Firestore.....</b>	<b>3</b>
<b>2.3 Alamofire.....</b>	<b>3</b>
<b>2.4 SDWebImage.....</b>	<b>4</b>
<b>2.5 LocalAuthentication.....</b>	<b>4</b>
<b>3. Explicació de l'API seleccionada.....</b>	<b>5</b>
<b>3.1 Funcionalitats Principals de la API.....</b>	<b>5</b>
<b>3.2 Integració de la API.....</b>	<b>6</b>
<b>4. Disseny de l'aplicació.....</b>	<b>7</b>
<b>4.1 Disseny de les vistes.....</b>	<b>7</b>
<b>4.2 Diagrama de classes.....</b>	<b>10</b>
<b>4.3 Responsabilitats de classes.....</b>	<b>13</b>
<b>4.3.1 Classes Base.....</b>	<b>13</b>
<b>4.3.2 Autenticació.....</b>	<b>13</b>
<b>4.3.3 Vistes principals i interfície d'usuari.....</b>	<b>14</b>
<b>4.3.4 Gestió de dades (TMDB).....</b>	<b>15</b>
<b>5. Funcionament de l'aplicació.....</b>	<b>16</b>
<b>6. Problemes trobats.....</b>	<b>24</b>
<b>7. Conclusions.....</b>	<b>27</b>
<b>8. Bibliografia.....</b>	<b>28</b>

## 1. Introducció al projecte

WikiFilms és una aplicació mòbil desenvolupada per a dispositius iOS utilitzant el llenguatge de programació Swift, dissenyada per ser compatible tant en iPhone com amb iPad.

L'aplicació permet a l'usuari consultar pel·lícules destacades del moment i accedir a informació detallada sobre cadascuna d'elles, com ara el títol, la descripció, la valoració i altres dades que puguin ser d'interès.

A més, també ofereix a l'usuari la possibilitat de crear llistes personalitzades (Watchlist) on pot afegir les pel·lícules que ja ha vist o que vulgui veure més tard.

L'objectiu principal del projecte és desenvolupar una aplicació completa i preparada per a producció que integri tant serveix externs com funcionalitats pròpies del dispositiu.

En aquest sentit, WikiFilms incorpora la connexió amb una API pública per a l'obtenció dinàmica de les dades sobre les pel·lícules, un sistema d'autenticació d'usuaris mitjançant Firebase Authentication i la integració de diferents SDKs propis del sistema operatiu iOS per gestionar la interfície d'usuari, la navegació i la seguretat de l'aplicació..

La motivació del projecte neix amb l'objectiu de posar en pràctica els coneixements adquirits durant el curs, tant a nivell de programació, com en el disseny d'interfícies d'usuari atractives, funcionals i orientades a un entorn real d'ús, tenint en compte l'experiència d'usuari, la correcta gestió de les dades i la separació de responsabilitats entre les diferents parts del sistema.

En quant al marc del projecte, aquest s'emmarca en el context acadèmic i té com a finalitat consolidar els conceptes treballats durant l'assignatura, posant en pràctica tot el que s'ha vist al llarg del semestre per tal d'apropar-se el màxim possible al desenvolupament d'una aplicació mòbil en iOS professional.

## 2. SDKs utilitzats

Durant el desenvolupament de WikiFilms s'han utilitzat diversos SDKs i frameworks amb l'objectiu d'aconseguir una aplicació funcional i segura.

### 2.1 UIKit

**Web:** <https://developer.apple.com/documentation/uikit>

Per una banda, com que hem treballat amb l'SDK de iOS, s'ha utilitzat el framework UIKit, que forma part d'aquest SDK d'Apple, per al desenvolupament de la interfície d'usuari de l'aplicació. UIKit ens ha permès crear interfícies mitjançant storyboards i controlar la lògica de navegació i interacció a través de classes com UIViewController, UINavigationController o UITabBarController, entre d'altres.

S'ha escollit la utilització d'UIKit ja que ha sigut el framework que ens han ensenyat a utilitzar al llarg de l'assignatura i permet una integració directa amb Xcode, així com el control detallat de la interfície de l'aplicació utilitzant els storyboards.

### 2.2 Firebase Authentication i Firestore

**Web:** <https://firebase.google.com/?hl=es-419>

Pel que fa la gestió d'usuaris de l'aplicació, s'ha integrat Firebase Authentication i Firestore. Firebase Authentication s'ha utilitzat per implementar de manera senzilla i segura el registre i inici de sessió d'usuaris mitjançant correu electrònic i contrasenya.

D'altra banda, Firestore s'ha utilitzat com a base de dades per emmagatzemar informació extra associada als usuaris, la qual no podem gestionar directament amb Firebase Authentication, com ara el nom d'usuari (username) o les llistes personalitzades de pel·lícules (watchlists). Aquesta combinació permet separar la gestió d'autenticació de l'usuari de l'emmagatzematge de dades pròpies de l'aplicació.

### 2.3 Alamofire

**Web:** <https://github.com/Alamofire/Alamofire>

Per a la comunicació amb l'API externa utilitzada, s'ha fet ús del framework Alamofire. Aquest SDK facilita la realització de peticions HTTP (GET, POST, etc.) i la gestió de respostes de manera més senzilla i llegible.

---

Alamofire s'ha utilitzat per obtenir informació sobre pel·lícules a partir de l'API pública TMDB, gestionar els paràmetres de les peticions i processar les respostes.

## 2.4 SDWebImage

**Web:** <https://github.com/SDWebImage/SDWebImage>

Per a la càrrega i gestió d'imatges remotes, com ara els pòsters de les pel·lícules, s'ha utilitzat el framework SDWebImage. Aquest SDK permet descarregar imatges des d'URLs de manera asíncrona.

Gràcies a SDWebImage, l'aplicació pot mostrar imatges de forma eficient sense bloquejar la interfície d'usuari, millorant així el rendiment i l'experiència d'ús. A més, la gestió de la memòria caché també ajuda a reduir el nombre de peticions repetides a l'API, optimitzant el consum de dades i el temps de càrrega.

## 2.5 LocalAuthentication

**Web:** <https://developer.apple.com/documentation/localauthentication>

Per implementar l'autenticació biomètrica a l'aplicació, s'ha utilitzat el framework LocalAuthentication, que forma part de l'SDK d'iOS.

Aquest framework permet utilitzar els sistemes d'autenticació biomètrica del dispositiu, com ara Face ID o Touch ID, de manera segura i integrada amb el sistema operatiu.

En el projecte WikiFilms, LocalAuthentication s'ha utilitzat a la vista de Login per permetre a l'usuari iniciar sessió mitjançant biometria, sempre que el dispositiu ho permeti. Aquesta funcionalitat millora l'experiència d'usuari oferint un accés més ràpid i còmode, alhora que reforça la seguretat de l'aplicació.

S'ha escollit aquest SDK per tal d'utilitzar un SDK propi del dispositiu, tenint en compte que pogués encaixar amb l'objectiu de l'aplicació sense fer una implementació d'alguna funcionalitat molt forçada i no relacionada amb el projecte.

### **3. Explicació de l'API seleccionada**

Per a l'obtenció de la informació de les pel·lícules, el projecte utilitza una API pública especialitzada en dades cinematogràfiques.

**Nom de l'API:** The Movie Database (TMDB) API.

**Web:** <https://developer.themoviedb.org/docs/getting-started>

Aquesta API proporciona accés a una gran base de dades de pel·lícules, sèries i actors, oferint funcionalitats com la cerca de pel·lícules, la obtenció de detalls, valoracions, dates d'estrena i imatges promocionals. TMDB és una de les plataformes més utilitzades en aplicacions audiovisuals, ja que manté la informació actualitzada i en múltiples idiomes.

#### **3.1 Funcionalitats Principals de la API**

Les funcionalitats principals que aporta la API de TMDB al projecte se centren en la consulta i visualització de contingut cinematogràfic. L'aplicació pot recuperar les pel·lícules més populars del moment i també aquelles que tenen una millor valoració per part dels usuaris. Aquestes dades es mostren a la pantalla principal mitjançant "collections" les quals permeten una navegació intuïtiva.

A més, la API permet realitzar cerques de pel·lícules a partir del text introduït per l'usuari. Aquesta funcionalitat fa possible que l'usuari pugui localitzar ràpidament qualsevol pel·lícula simplement escrivint una part del seu títol. Un cop seleccionada una pel·lícula, l'aplicació pot obtenir tota la seva informació detallada, com la sinopsi, la valoració mitjana i les imatges associades.

Aquest conjunt de funcionalitats garanteix que l'aplicació disposi d'un contingut ric, actualitzat i rellevant per a l'usuari, sense necessitat de mantenir una base de dades pròpia.

### 3.2 Integració de la API

La integració de la API de TMDB s'ha realitzat mitjançant peticions HTTP utilitzant la llibreria Alamofire, que facilita la comunicació amb serveis web i el tractament de respostes en format JSON. Per accedir a l'API és necessari utilitzar un sistema d'autenticació mitjançant un Bearer Token, el qual s'envia en la capçalera de cada petició.

Per tal de mantenir una arquitectura neta, modular i fàcilment escalable, s'ha creat una estructura basada en tres components principals: el router, el client i els models. En primer lloc, el fitxer TMDBRouter centralitza totes les rutes de l'API i defineix els diferents endpoints disponibles, com ara les pel·lícules populars, les més valorades o la cerca. Aquest router també gestiona els paràmetres de cada consulta, com la pàgina o el text de cerca, i construeix les peticions HTTP.

A partir del router, el projecte disposa d'un client de xarxa anomenat TMDBClient, encarregat de realitzar les peticions, validar les respostes i decodificar-les en un array d'objectes TMDBMovie. Aquest client segueix el patró Singleton, ja que només existeix una única instància accessible mitjançant TMDBClient.shared, amb un inicialitzador privat que impedeix crear-ne de noves. D'aquesta manera, tota l'aplicació utilitza el mateix punt d'accés a l'API, garantint una gestió centralitzada de les comunicacions.

Models.swift conté les estructures de les dades rebudes de l'API. En el projecte s'utilitzen TMDBMovie i TMDBMovieResponse, que conformen una capa intermèdia entre les dades en brut (JSON) i la interfície gràfica. Gràcies al protocol Decodable, aquestes estructures permeten convertir automàticament les respostes de l'API en objectes utilitzables dins de l'aplicació.

Per obtenir un nombre més gran de resultats, l'aplicació fa ús del sistema de paginació proporcionat per la pròpia API de TMDB, sol·licitant diverses pàgines i combinant-ne els resultats en un sol array. Finalment, un cop les dades són rebudes i processades, la interfície gràfica es refresca automàticament per mostrar el nou contingut a l'usuari, assegurant una experiència fluida i dinàmica.

## 4. Disseny de l'aplicació

En aquest apartat es descriu el disseny general de l'aplicació WikiFilms, tant des del punt de vista visual com estructural.

S'explica l'organització de les diferents vistes que conformen la interfície d'usuari, així com l'arquitectura interna de l'aplicació mitjançant el diagrama de classes i la distribució de responsabilitats entre les diferents classes del projecte.

L'objectiu és mostrar com hem planificat i estructurat el disseny per tal de garantir una experiència d'usuari clara i coherent, alhora que es manté una separació adequada de les responsabilitats, facilitant l'escalabilitat i el manteniment del codi.

### 4.1 Disseny de les vistes

Pel que fa el disseny de les vistes, s'ha plantejat amb l'objectiu d'ofrir una navegació clara, coherent i intuïtiva a l'usuari, tot mantenint una estètica el màxim d'homogènia possible en totes les pantalles de l'aplicació.

L'aplicació segueix una estructura seqüencial inicial d'autenticació, i un cop superada, una navegació principal basada en diferents seccions accessibles per l'usuari mitjançant un sistema de TabBar.

Primer de tot, l'aplicació disposa d'una pantalla de benvinguda (Welcome View) que actua com a punt d'entrada i permet a l'usuari escollir entre iniciar sessió o registrar-se. A continuació, es presenten les vistes de Login i Register, dissenyades de forma senzilla i funcional, amb els camps de formulari clars perquè l'usuari pugui introduir les seves dades i/o credencials de forma intuïtiva.

Un cop autenticat, l'usuari accedeix a la vista principal (Home View), que actua com a nucli de l'aplicació. En aquesta vista es mostren diferents llistats de pel·lícules (pel·lícules millor valorades i pel·lícules populars) mitjançant una combinació de taules i col·leccions, permetent una navegació visual cap a la vista detallada de cadascuna de les pel·lícules.

La vista de detall (Details View) mostra la informació completa de la pel·lícula seleccionada, incloent la imatge principal (poster), el títol, la valoració i la descripció. Des d'aquesta vista, l'usuari pot interactuar amb la pel·lícula afegint-la a la seva llista personal (Watchlist/MiLista).

---

En quant a la valoració de les pel·lícules, s'ha utilitzat un sistema d'escala de color segons la valoració, permetent així visualitzar-ho amb més qualitat, sent el color verd una valoració alta, seguida de groc, taronja i, finalment, vermell per les valoracions amb nota més baixa.

L'aplicació també incorpora una vista de cerca (Search View), que permet a l'usuari buscar pel·lícules mitjançant text, mostrant resultats dinàmics obtinguts a partir de l'API externa. Per tal que en accedir a aquesta vista, no es mostri la pantalla buida, s'utilitza també el listat de pel·lícules populars mentre l'usuari no ha realitzat cap cerca.

La vista de Watchlist (o MiLista) permet consultar i gestionar les pel·lícules guardades per l'usuari, mostrant-les en format de llista amb el nom, la imatge i la valoració, i permetent la seva eliminació mitjançant un “swipe” cap a l'esquerra i mostrant el botó d'eliminar.

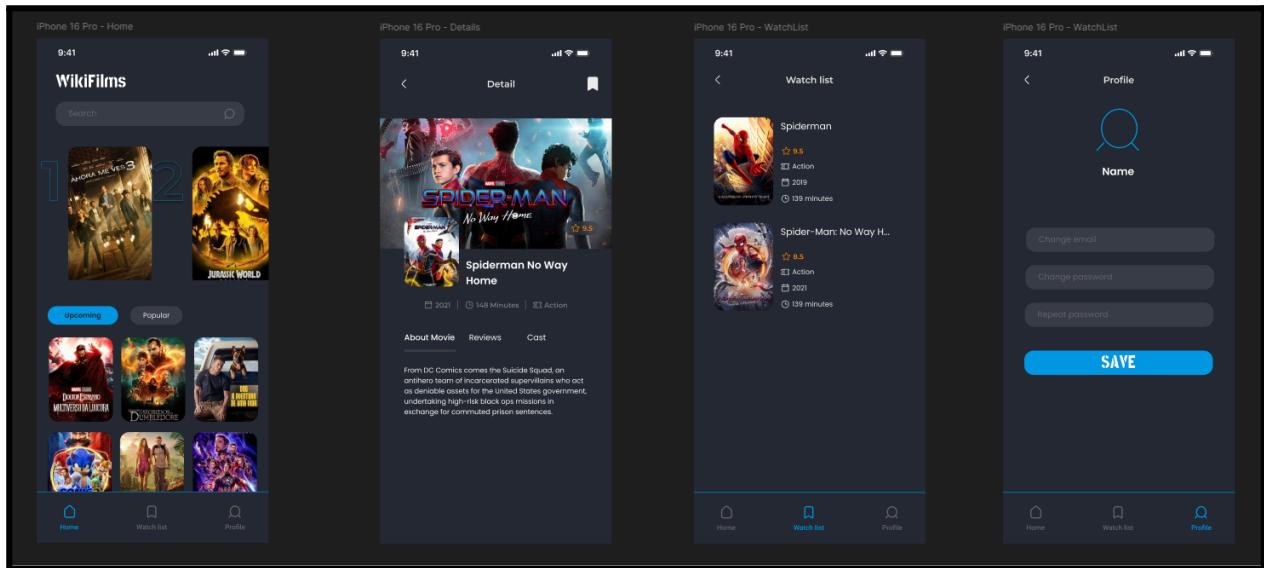
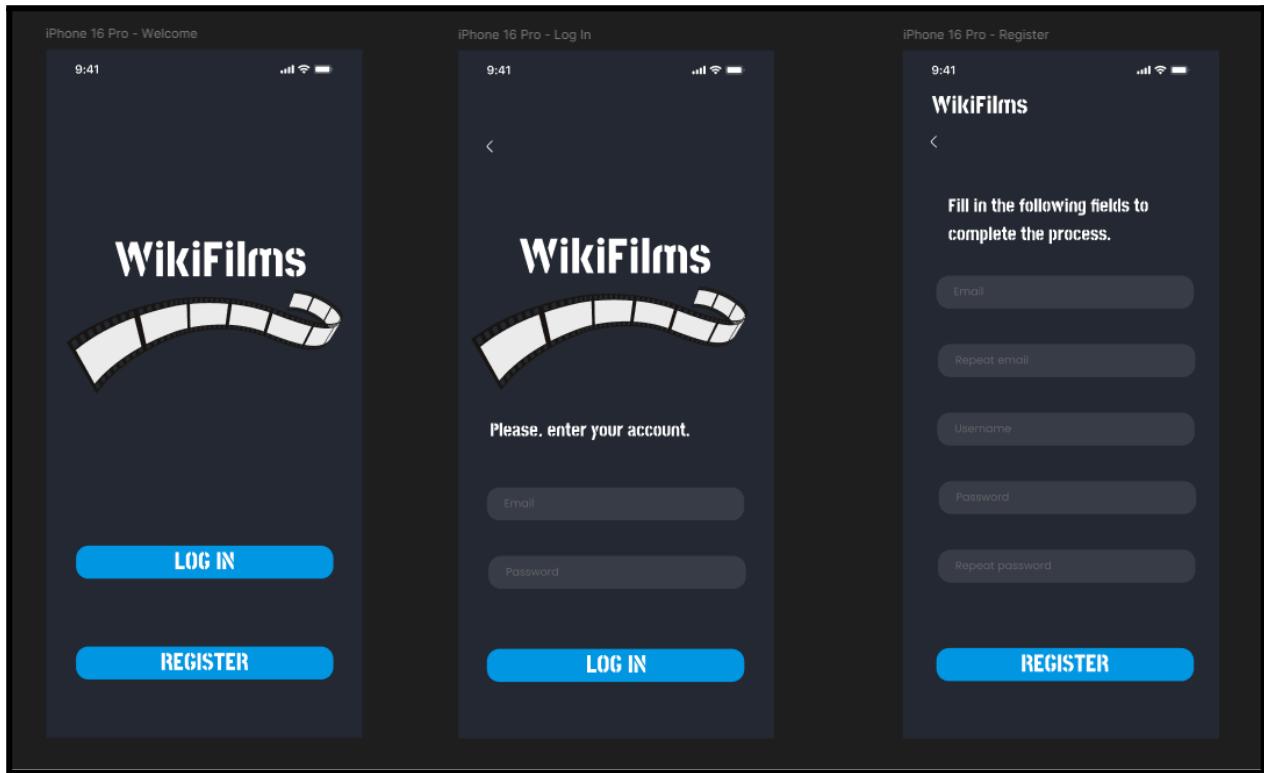
Finalment, la vista de Perfil (Profile) permet a l'usuari visualitzar i modificar les seves dades personals bàsiques, com el correu electrònic o la contrasenya, mitjançant la integració amb FirebaseAuth i Firestore.

Pel que fa a l'aspecte visual de tota l'aplicació, totes les vistes mantenen coherència en colors, tipografia i disposició d'elements, oferint una experiència uniforme i orientada a un ús senzill.

El prototip inicial de la interfície es va realitzar mitjançant Figma, servint de guia visual durant el desenvolupament de les diferents vistes de l'aplicació.

A més, l'aplicació incorpora suport per a múltimes idiomes (castellà i anglès) mitjançant fitxers Localizable, permetent adaptar tant la interfície com les peticions a l'API segons l'idioma del dispositiu.

A continuació, podem veure una imatge del disseny del prototip en Figma, així com l'enllaç al disseny.

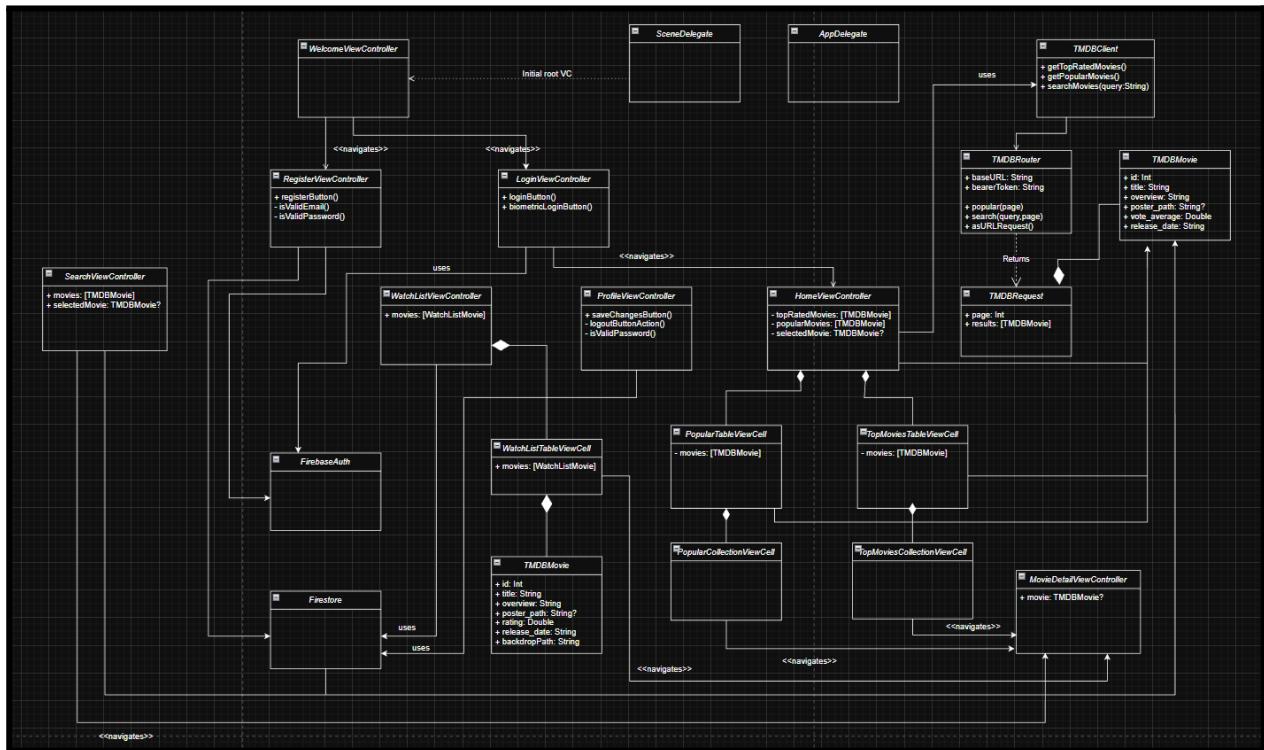


### Web Figma de la Interfície de l'App:

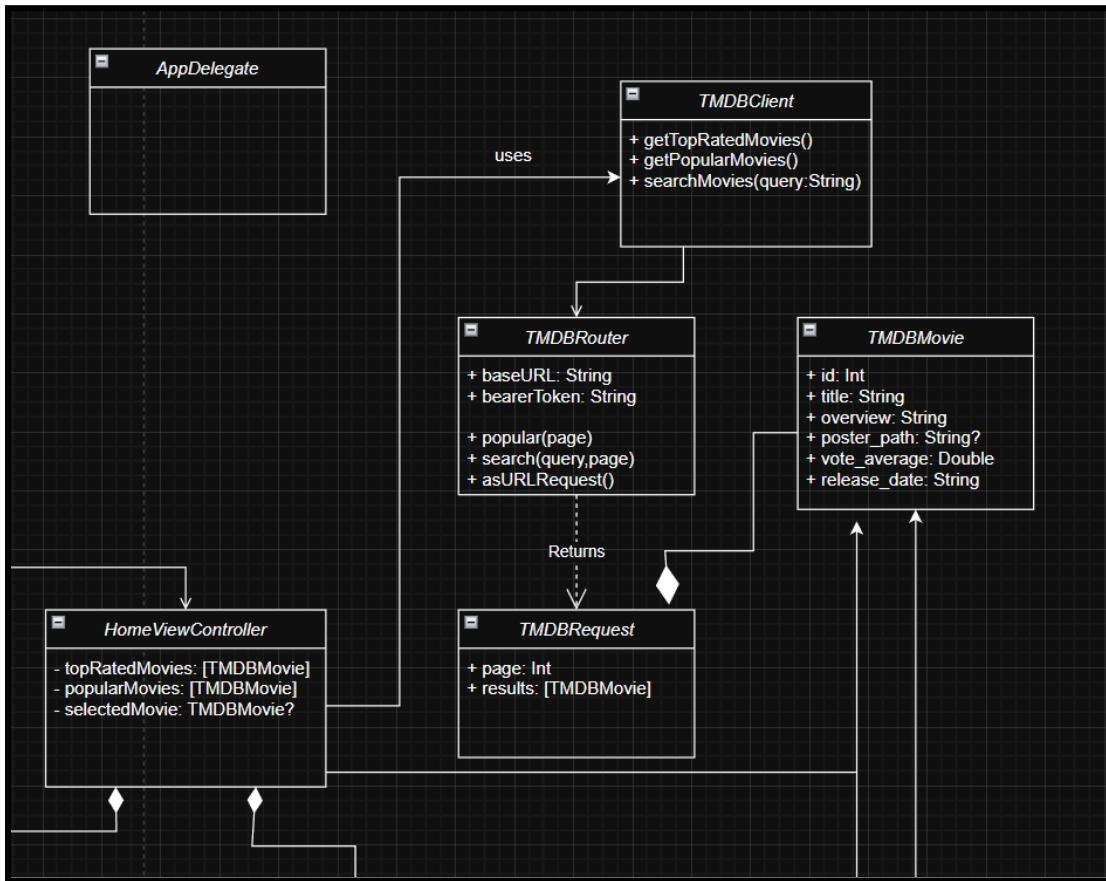
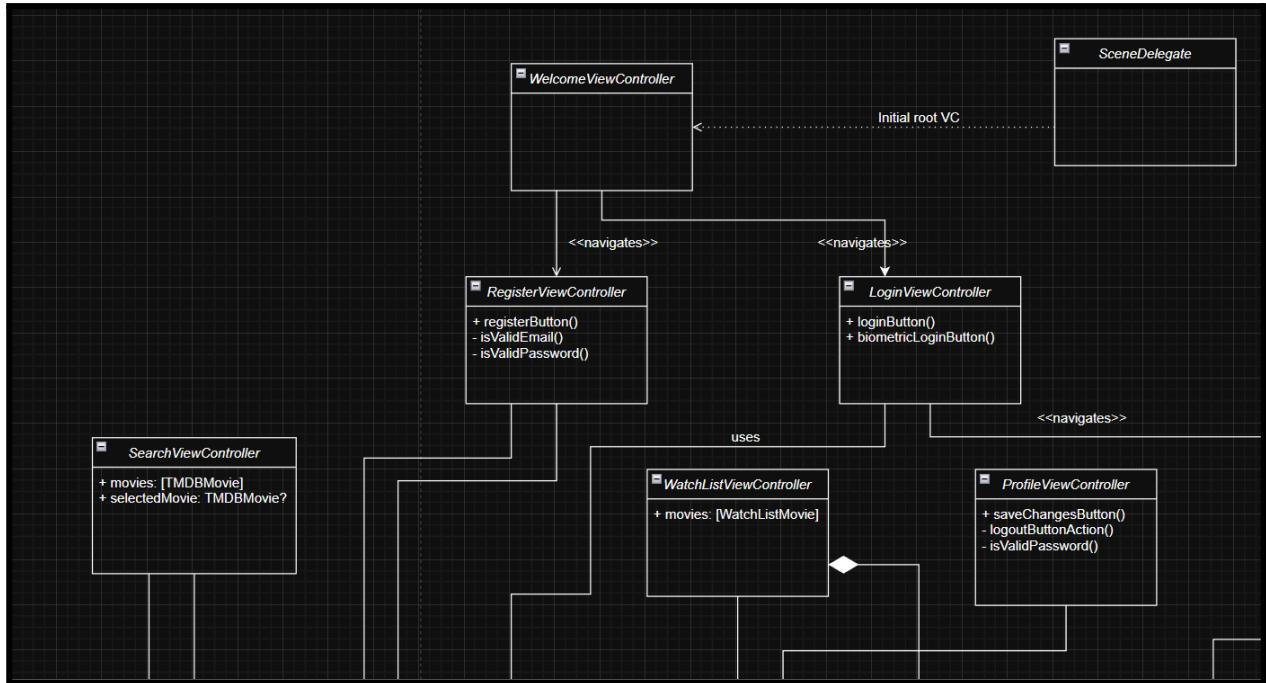
<https://www.figma.com/design/3wGtDe9z4RlYiNZxtpOF4t/Projectes-Mobils?node-id=0-1&p=f>

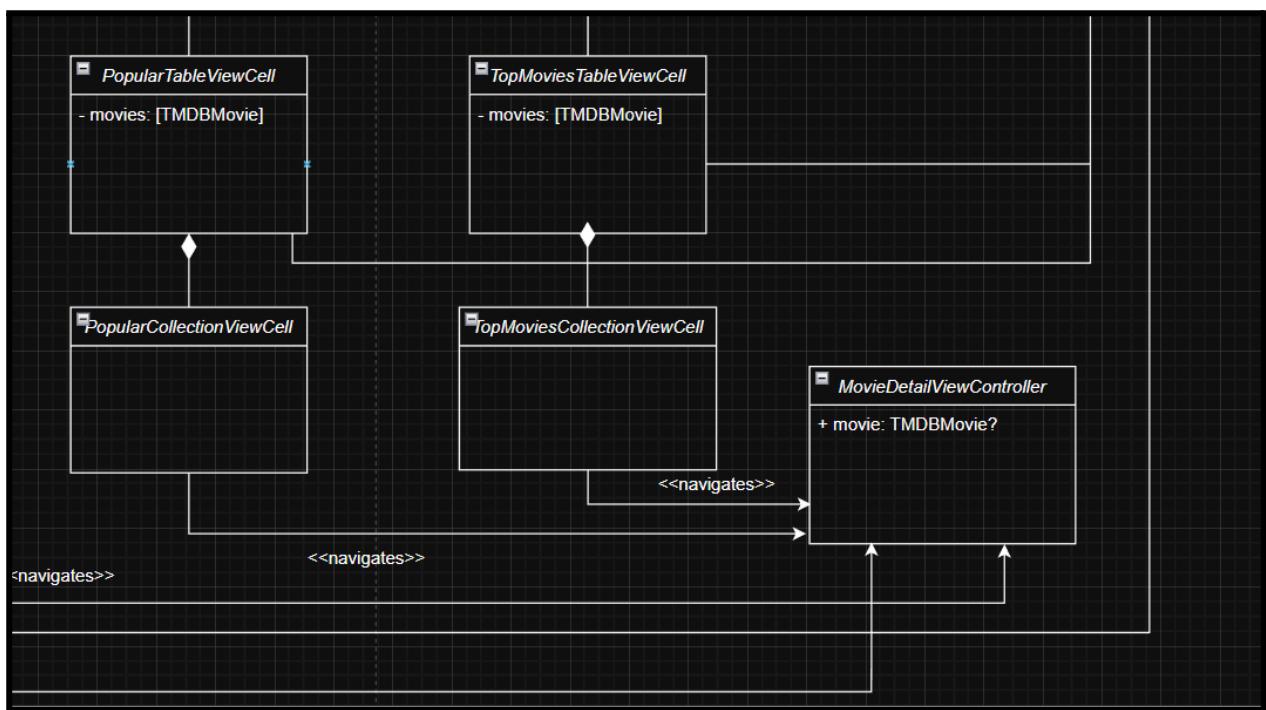
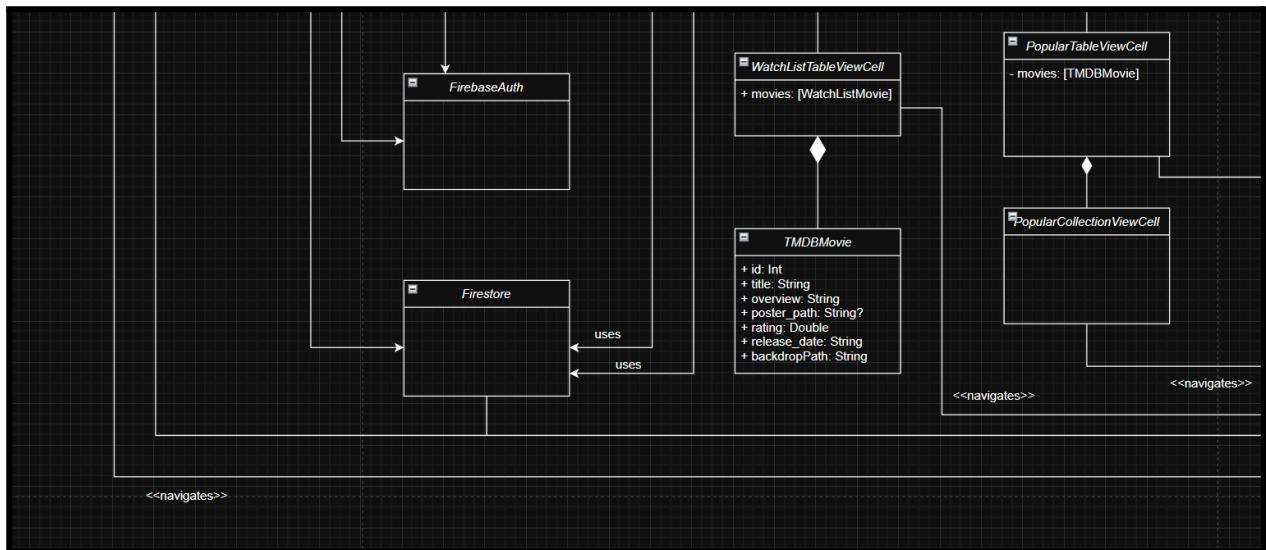
## 4.2 Diagrama de classes

Captura de pantalla del Diagrama de Classes general:



Captures de pantalla del Diagrama de Classes ampliat:





## 4.3 Responsabilitats de classes

L'arquitectura del projecte es divideix en diferents blocs funcionals, cadascun amb les seves responsabilitats clarament definides.

1. Classes Base (Cicle de vida de l'aplicació)
2. Autenticació
3. Vistes principals i interfície d'usuari
4. Gestió de Dades (API TMDB i Models)

A continuació es detallen les diferents responsabilitats de cadascuna de les classes.

### 4.3.1 Classes Base

#### AppDelegate

És el punt d'entrada de l'aplicació i no depèn de cap ViewController en si mateix. Té com a responsabilitats la inicialització global de l'app, incloent la configuració de Firebase durant l'arrencada.

#### SceneDelegate

Gestiona el cicle de vida de les escenes de l'aplicació i s'encarrega de la gestió de les finestres i les escenes actives.

### 4.3.2 AUtenticació

#### WelcomeViewController

Actua com a pantalla inicial de l'aplicació i controla la navegació cap a les vistes de Login o Registre.

#### LoginViewController

Gestiona l'inici de sessió de l'usuari mitjançant correu electrònic i contrasenya, així com l'autenticació biomètrica utilitzant LocalAuthentication. Un cop l'usuari és autenticat correctament, permet l'accés al contingut principal de l'aplicació.

#### RegisterViewController

Gestiona el registre de nous usuaris. Implementa la validació dels camps correu electrònic i contrasenya, per verificar que compleixen les següents condicions:

Correu Electrònic: ha de contenir ambdós caràcters '@' i '.'.

Contrasenya: ha de tenir mínim 6 caràcters, 1 minúscula, 1 majúscula i 1 número.

També implementa l'emmagatzematge de dades extres, com el nom d'usuari (username) a Firestore.

#### **4.3.3 Vistes principals i interfície d'usuari**

##### **HomeViewController**

És la vista principal de l'aplicació. Mostra diferents llistats de pel·lícules obtingudes de l'API (populars i millor valorades).

Permet realitzar una navegació cap a la vista detallada de cadascuna de les pel·lícules seleccionades per l'usuari.

##### **SearchViewController**

Permet la cerca de pel·lícules mitjançant text, mostrant resultats dinàmics obtinguts de l'API de TMDB.

##### **MovieDetailViewController**

Mostra la informació detallada d'una pel·lícula i permet afegir-la a la watchlist de l'usuari, emmagatzemant-la a Firestore.

##### **WatchListViewController**

Gestiona la visualització i administració de la llista de pel·lícules guardades per l'usuari. Realitza lectures i escriptures a Firestore i permet eliminar elements de la llista.

##### **ProfileViewController**

Permet mostrar les dades de l'usuari i modificar-les (correu electrònic i contrasenya). Realitza tant lectura com escriptura de les dades emmagatzemades a FirebaseAuth i Firestore.

##### **Cel·les**

Les classes de cel·les personalitzades (PopularTableViewCell, TopMoviesTableViewCell, PopularCollectionViewCell, TopMoviesCollectionViewCell, WatchListTableViewCell) tenen com a única responsabilitat la representació visual de la informació, separant la lògica de presentació de la lògica de negoci.

#### **4.3.4 Gestió de dades (TMDB)**

##### **TMDBClient**

Centralitza totes les peticions HTTP a l'API de TMDB. Utilitza Alamofire per realitzar les peticions i decodificar les respostes.

##### **TMDBRouter**

Defineix els endpoints de l'API i construeix les peticions HTTP, separant la definició de rutes de la lògica de xarxa, URLs.

##### **Models**

Les estructures de dades representen els objectes de domini de l'app i són utilitzades per la majoria de vistes, les cel·les i els controllers per mostrar la informació relacionada amb les pel·lícules.

## 5. Funcionament de l'aplicació

En aquest apartat descriurem com utilitzar l'aplicació, així com les captures de pantalla de cadascun dels passos, per tal de poder visualitzar les diferents funcionalitats implementades.

Comentar que, tot i que les captures de l'aplicació estan realitzades des d'un simulador iPhone, l'aplicació està dissenyada per poder funcionar també correctament amb iPad.

De la mateixa manera, l'aplicació funciona tant en idioma Anglès com en Castellà, tot i així les captures estan realitzades en la versió en castellà.

Primer de tot, un cop engeguem l'aplicació se'n obrirà la WelcomeView, amb el logotip que hem dissenyat de l'aplicació i dos botons, un per iniciar sessió i un altre per registrar-se.

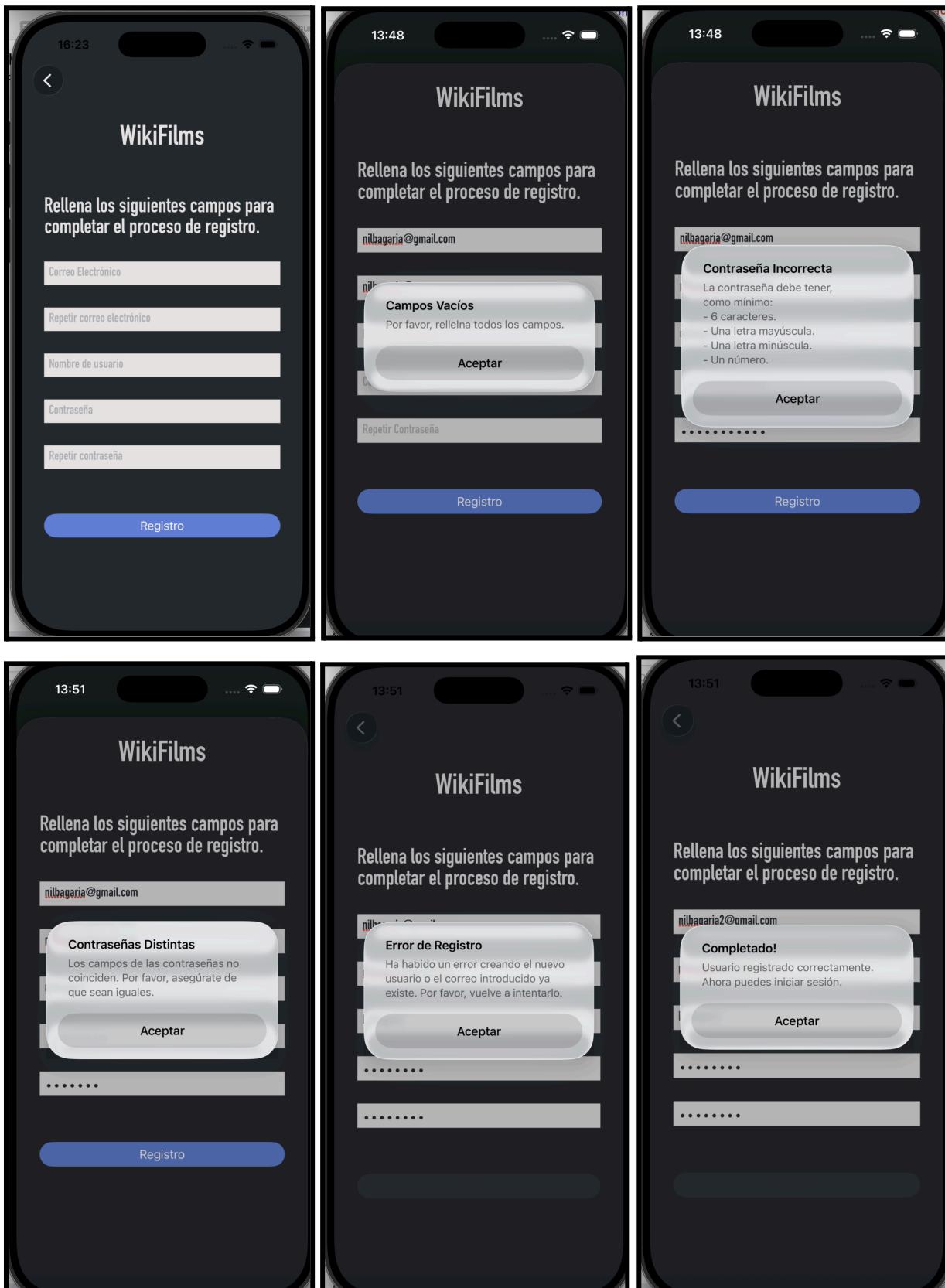


A continuació, seleccionem l'opció “Registro” per accedir a la vista per registrar un nou usuari.

En aquesta pantalla, caldrà omplir tots els camps per registrar un nou usuari.

S'han implementat diverses verificacions i alertes per informar a l'usuari de les accions que no realitza correctament i així poder ajudar-lo a realitzar el procés de registre. Un cop l'usuari s'ha registrar correctament, automàticament es fa un “pop” i es retorna a la pàgina inicial per facilitar la navegació cap al “Iniciar Sesión”.

Podem veure la pantalla de registre i algunes de les diferents alertes configurades a continuació.



Un cop registrat correctament, l'aplicació ens retorna a la WelcomeView. Seguidament, si seleccionem “Iniciar Sesión” se’ns dirigeix a la pantalla per iniciar sessió al nostre compte.

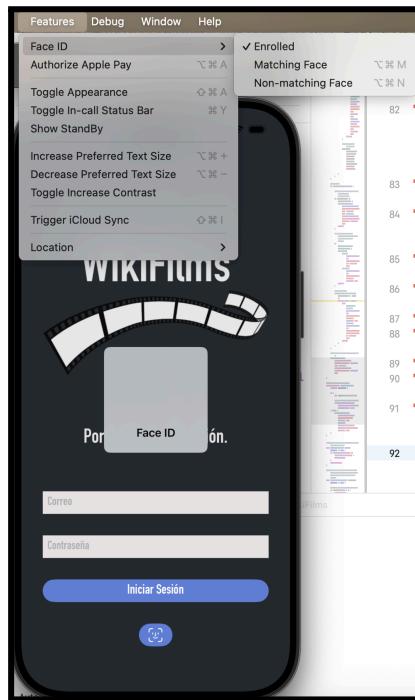
Aquí, tenim dues opcions (dos botons): iniciar sessió amb les credencials, iniciar sessió utilitzant la biometria (FaceID).

En el cas de ser el primer accés a l'aplicació (o haver tancat sessió prèviament) caldrà accedir mitjançant les credencials. En cas de ja haver accedit prèviament al nostre compte i, per exemple, haver sortit de l'aplicació deixant-la en segon pla, podrem iniciar sessió utilitzant el FaceID seleccionant el botó corresponent.

Podem veure les diferents imatges a continuació.



Seguidament, observem una imatge de l'accés amb biometria, un cop enrolat el dispositiu (utilitzant el simulador). Observem com s'obre l'icana de FaceID i, en cas de seleccionar Matching Face s'accedeix correctament. En canvi, si es selecciona Non-matching Face, falla l'autenticació:



Un cop iniciada la sessió a l'aplicació, entrem a la vista Home. Aquí tenim per una banda el llistat de pel·lícules millor valorades (de l'1 al 20 “swipejant” cap a l'esquerra) i un altre llistat inferior amb pel·lícules populars del moment (“scrollejant” cap a baix).

Podem observar també a la part inferior, un Tab Bar, per moure'n's a través de les diferents pantalles de l'aplicació (Home, WatchList, Search, Profile).



A continuació, una altra funcionalitat de la qual disposa l'usuari és la vista detallada.

En cas de seleccionar una pel·lícula, ja sigui del llistat de millors valorades, com del llistat de populars, s'accedeix a la vista detallada de la pel·lícula on, en cas de disposar-ne (si la API n'ofereix) es mostra la imatge de la pel·lícula (poster), una imatge de background, el títol, la valoració i la sinopsis. En el cas de la valoració, segons el valor concret d'aquesta, es mostrerà de color verd, groc, taronja o vermell (de millor a pitjor valoració).



Tal com podem observar, a la part superior dreta de la vista detallada de les pel·lícules hi tenim un botó anomenat “Guardar” amb el símbol de WatchList(MiLista).

En aquest cas, si seleccionem aquest botó, el que farem es guardar aquella pel·lícula en concret a la nostra llista de pel·lícules, tal i com veiem a continuació.

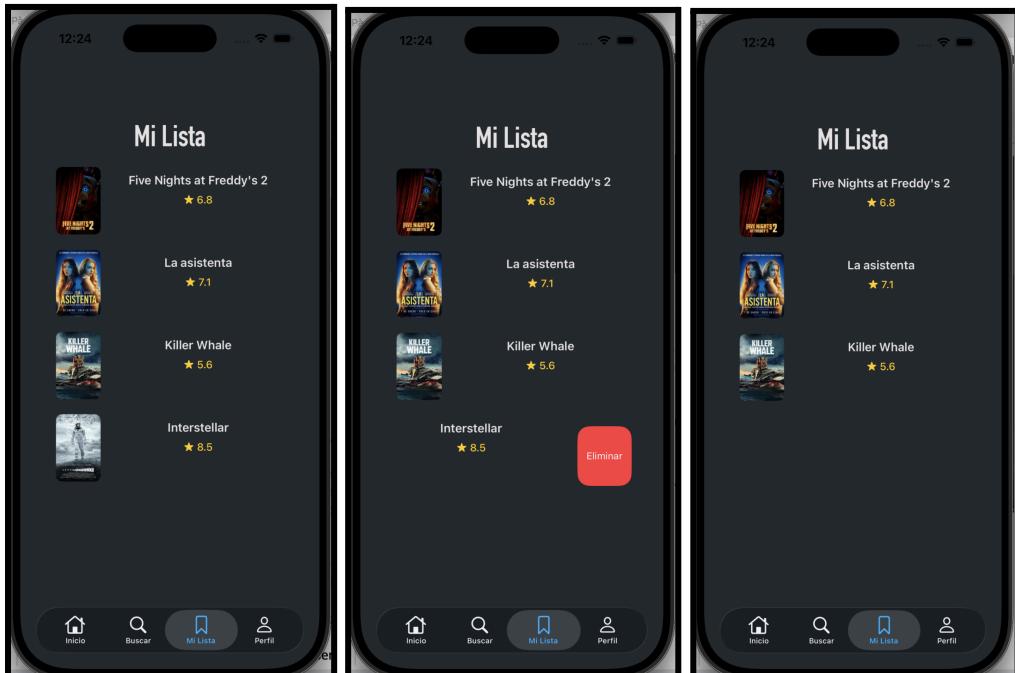
En aquest exemple, guardarem la pel·lícula “Interstellar” i després podrem comprovar com aquesta apareix a la nostra WatchList guardada.



A continuació, si accedim mitjançant el Tab Bar a la WatchList, podem observar com apareix la nova pel·lícula guardada a la llista.

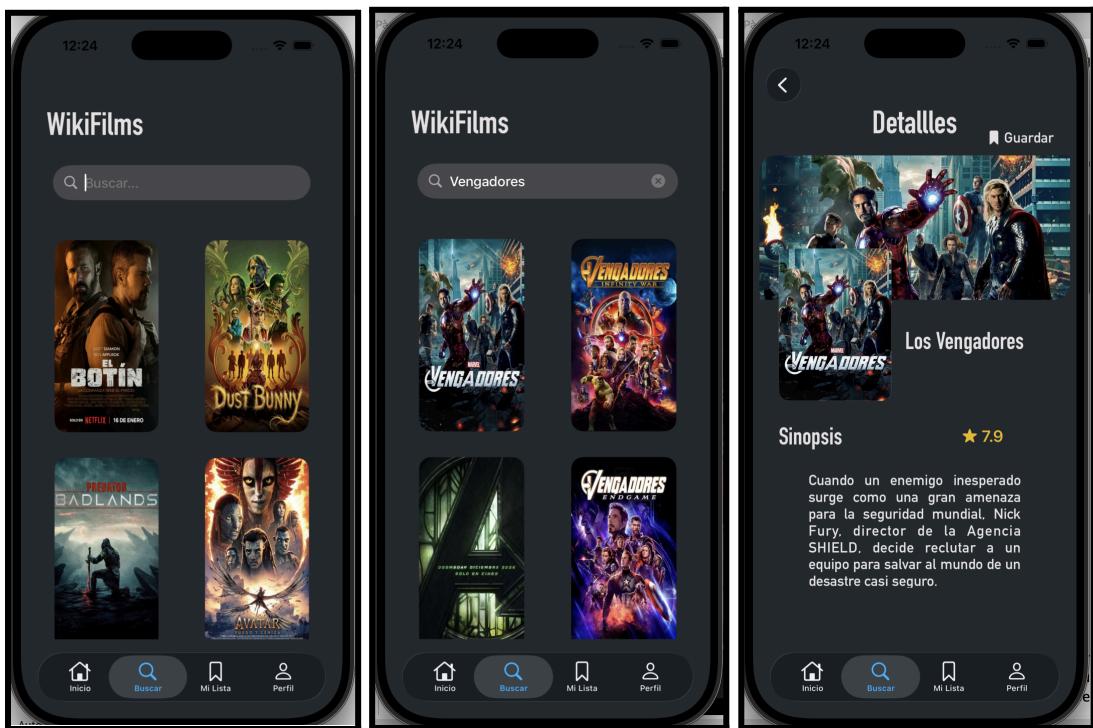
Des d'aquest menú, de nou també podem accedir a la vista detallada de les pel·lícules seleccionant-les (igual que des de la Home page) i, també, podem eliminar les pel·lícules que vulguem de la llista fent un “swipe” cap a l'esquerra al damunt de la pel·lícula.

Fent això, ens apareixerà un botó “Eliminar” per esborrar la pel·lícula de la llista.



Ara parlarem d'una altra funcionalitat, en aquest cas de la vista de Search. Aquí si hi accedim a través del TabBar anirem a una vista on ens apareixeran pel·lícules populars (com al Home Page), però també hi tindrem una barra de cerca per buscar pel nom les pel·lícules que ens interessen, com podem veure a continuació.

De nou, un cop cercada una pel·lícula, també podem accedir a la seva vista detallada per veure la seva informació i/o guardar-la a la nostra llista.



Finalment, tenim l'última de les opcions que correspon al Perfil de l'usuari.

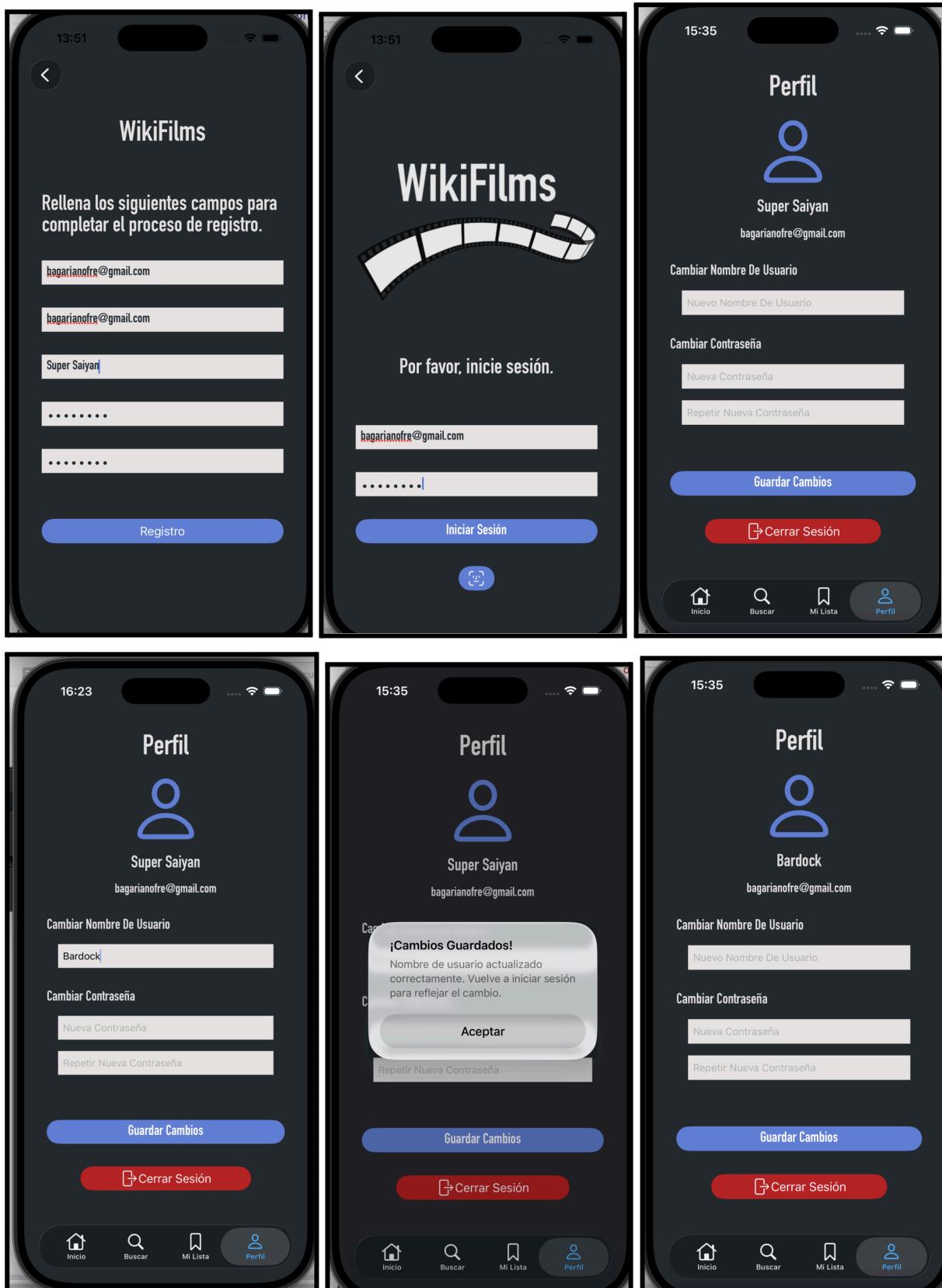
En aquesta vista hi podem observar les dades de l'usuari com el correu i el nom d'usuari. A més d'això, disposem de la possibilitat de modificar el nom d'usuari o la contrasenya de l'usuari, aplicant el canvi immediat en ambdues coses un cop loguejat de nou.

Per últim, disposem del botó per tancar sessió i sortir de l'aplicació.

A continuació, veurem un exemple des del registre fins a la vista de perfil per mostrar que les dades es corresponen correctament i que la modificació dels camps també funciona de forma correcta.

En aquest exemple farem el registre d'un usuari amb correu [bagarianofre@gmail.com](mailto:bagarianofre@gmail.com) i nom d'usuari "Super Saiyan" i el modificarem perquè el mateix usuari amb el mateix correu passi a tenir el nom d'usuari "Bardock".

Tant per aplicar els canvis de nom d'usuari com de contrasenya, cal tancar sessió i tornar a iniciar sessió. En cas de la contrasenya cal iniciar sessió ja amb la nova contrasenya.



## 6. Problemes trobats

Durant el desenvolupament del projecte han sorgit diversos problemes tant a nivell de disseny com d'implementació tècnica. A continuació, es descriuen els més importants, així com les solucions o els enfocs que hem adaptat per solucionar-ho.

Un dels primers problemes important va aparèixer durant la implemtació de la funcionalitat de cerca de pel·Lícules. Inicialment, la cerca s'havia plantejat per fer-la dins la mateixa vista principal (Home), on ja es mostraven diferents collections de pel·Lícules. Aquesta combinació però ens va resultar difícil de gestionar, tant a nivell de codi com d'organització de la interfície, ja que implicava controlar les diferents collections i els diferents estats dins de la mateixa vista.

Com a solució, vam replantejar el disseny inicial i vam decidir separar la funcionalitat de cerca en una vista independent, afegint una nova pestanya al Tab BAr (Search). Aquest canvi va simplificar considerablement la implementació i ens va permetre treballar la cerca de forma més senzilla i estructurada, tot i haver de sacrificar parcialment el disseny inicial definit a Figma.

Relacionat amb això, un altre problema concurrent va ser la diferència entre el disseny inicial de la interfície realitzat a Flgma i la seva posterior implementació mitjançant Storyboards i codi.

Algunes idees de disseny, com ara l'ús de submenús dinàmics o el canvi de contingut dinàmic dins d'una mateixa vista, va resultar-nos massa complex d'implementar. Per aquest motiu, es va optar per simplificar una mica l'estruatura de la interfície, mantenint les mateixes funcionalitats però utilitzant una organització més "estable", com per exemple mostrant collections diferenciades en lloc de contingut dinàmic intercanviable.

Per altra banda, un dels problemes més rellevants va ser el fet de desenvolupar una aplicació per a un dispositiu físic del qual no es disposava d'una unitat real per poder fer proves. Tot i que el simulador d'iOS permet executar i provar la majoria de funcionalitats de l'aplicació, algunes característiques, com l'autenticació biomètrica, no es comporten de manera totalment realista.

En el cas de la biometria, el simulador permet iniciar sessió sense una associació clara amb un usuari concret, fet que pot generar comportaments que no es corresponen amb un entorn real. Sense ser usuari d'iPhone, entenem que en un dispositiu físic, el framework LocalAuthentication únicament verifica que la persona que interactua amb el dispositiu és el seu propietari, però no autentica directament un usuari de l'aplicació. Per aquest motiu, l'autenticació biomètrica hauria de permetre l'accés només quan existeix una sessió prèvia d'usuari activa.

Tot i així, durant les nostres proves amb el simulador, es va detectar que era possible accedir a l'aplicació mitjançant Face ID sense haver-se registrat ni iniciat sessió prèviament. Aquest comportament provocava un accés sense un usuari associat i generava problemes posteriors en funcionalitats que depenen d'un identificador d'usuari, com ara l'emmagatzematge de dades a Firestore (per exemple, la watchlist de pel·lícules).

Per solucionar aquest problema, es va implementar una comprovació prèvia abans d'iniciar l'autenticació biomètrica, verificant que existís una sessió d'usuari activa. En cas contrari, es força l'usuari a iniciar sessió de manera manual. D'aquesta manera, es garanteix que l'autenticació biomètrica només s'utilitza com a mètode d'accés ràpid per a usuaris ja autenticats prèviament, intentant reproduir així un comportament més proper al d'una aplicació real.

Un altre dels problemes que va aparèixer, va ser en la gestió de la modificació de les dades del perfil de l'usuari, concretament en el canvi del correu electrònic mitjançant Firebase Authentication.

Inicialment, es va implementar la possibilitat que l'usuari pogués modificar tant el correu electrònic com la contrasenya des de la pantalla de perfil. Tot i que el canvi de contrasenya es realitzava correctament i Firebase aplicava el canvi sense incidències, el canvi de correu electrònic presentava comportaments incorrectes. En molts casos, tot i executar el mateix flux de codi, Firebase rebutjava la modificació del correu electrònic i mantenia el correu original de l'usuari.

Després de buscar informació, vam comprovar que aquest comportament es deu al fet que Firebase aplica diferents nivells de seguretat segons el tipus de dada que es vol modificar. El correu electrònic és considerat un element crític d'identitat de l'usuari i, per motius de seguretat, Firebase requereix que l'usuari s'hagi autenticat recentment per permetre aquest tipus de canvi. En situacions en què la sessió havia estat restaurada automàticament, l'usuari havia accedit mitjançant autenticació biomètrica o havia passat un cert temps des de l'inici de sessió, Firebase bloquejava el canvi del correu electrònic.

Davant d'aquesta limitació, es va optar per realitzar una alerta per l'usuari informant que era necessari tornar a iniciar sessió per aplicar canvis sensibles i forçar el tancament de la sessió quan Firebase ho requeria, garantint així una gestió coherent del canvi. Tot i fer això, es va observar que aquesta funcionalitat afegia complexitat innecessària a l'experiència d'usuari i generava confusió, ja que tampoc quedava molt clar si el correu s'havia pogut modificar correctament o no, a més de que el correu electrònic rarament és una dada que l'usuari necessita modificar amb freqüència en les aplicacions.

Per aquest motiu, i seguint el funcionament habitual de moltes aplicacions reals, es va decidir eliminar la possibilitat de modificar el correu electrònic des de la pantalla de perfil i mantenir-lo únicament com a dada informativa. En el seu lloc, es va implementar la possibilitat de modificar el nom d'usuari (username), una dada que en aquest cas no es crítica i que s'emmagatzema a Firestore i que, per tant, pot ser actualitzada de manera senzilla i immediata sense afectar el sistema d'autenticació.

Aquesta decisió ens va permetre simplificar la lògica del codi, millorar l'experiència d'usuari i evitar conflictes derivats de les restriccions de seguretat de Firebase, mantenint alhora una gestió del perfil coherent amb el que havíem dissenyat inicialment.

Finalment, pel que fa al desenvolupament general, també vam tenir algun problema relacionat amb el renombrat del projecte i de la carpeta principal, fet que va provocar que Xcode no trobés correctament els paths dels recursos i els fitxers. Per tal de solucionar això, l'únic que vam poder fer va ser modificar el nom de la carpeta exterior però mantenint el nom del projecte a la carpeta interior, ja que sense fer-ho així no vam aconseguir que el projecte compilés de forma correcta.

## 7. Conclusions

El desenvolupament de l'aplicació WikiFilms ens ha permès aplicar de manera pràctica amb un objectiu d'entorn real, els coneixements adquirits al llarg de l'assignatura de Programació de Dispositius Mòbils, tant pel que fa al disseny d'interfícies, com a la programació, la gestió de dades, l'autenticació d'usuaris i la comunicació amb serveis externs mitjançant APIs, d'aplicacions per a iOS.

A nivell tècnic, la integració de Firebase Authentication i Firestore ens ha permès poder implementar un sistema de control d'usuaris (registre i inici de sessió), així com també ens ha permès poder gestionar les dades pròpies de l'usuari.

Al mateix temps, l'ús de l'API de TMDB, juntament amb Alamofire i SDWebImage, ens ha permès treballar amb dades reals i contingut dinàmic, apropiant el projecte a un entorn més professional.

Pel que fa als aprenentatges adquirits, podem destacar especialment la comprensió del cicle de vida de l'aplicació, la navegació entre les diferents vistes mitjançant storyboards i controllers, així com la separació de responsabilitats entre les diferents classes. També ha sigut important l'experiència de detectar problemes reals durant el desenvolupament i adaptar el disseny inicial a solucions més viables, com ha estat el cas de la gestió del perfil d'usuari o l'organització de les vistes principals en comparació al disseny plantejat incialment en Figma.

Tot i que l'aplicació compleix els objectius plantejats incialment, existeixen diverses possibles millores i ampliacions de cara a futures versions. Una de les principals seria ampliar la funcionalitat de la watchlist, permetent a l'usuari crear diverses llistes personalitzades (per exemple, "Pendents", "Vistes", "Preferits"), en lloc de disposar d'una única llista. Aquesta millora aportaria més flexibilitat i una experiència d'ús més rica i personalitzada per a l'usuari.

Una altra possible ampliació seria permetre a l'usuari afegir una imatge de perfil, emmagatzemada a Firebase, per personalitzar encara més el compte d'usuari. Així mateix, es podrien implementar funcionalitats addicionals com filtres avançats de cerca, recomanacions personalitzades segons l'historial de l'usuari o una millora del disseny visual amb animacions i transicions més elaborades, per fer més visual l'experiència de l'usuari.

En conclusió, podem dir que WikiFilms és un projecte complet i funcional que reflecteix i posa en pràctica els coneixements adquirits durant l'assignatura. Alhora, reflecteix la importància de totes les decisions preses prèviament en el disseny, així com la intenció d'apropar-se el màxim possible i adaptar-se a possibles limitacions amb la intenció de desenvolupar una aplicació destinada a l'entorn de producció.

## 8. Bibliografia

Forts d'informació referenciades segons la normativa ISO 690.

**La Salle – Universitat Ramon Llull.** *Materials de l'assignatura de Programació de Dispositius Mòbils.*

Disponible a:

<https://estudy2526.salle.url.edu/course/view.php?id=5887&sectionid=17221>

**Apple Inc.** *UIKit Documentation.*

Disponible a:

<https://developer.apple.com/documentation/uikit>

**Google LLC.** *Get started with Firebase Authentication on iOS.*

Disponible a:

<https://firebase.google.com/docs/auth/ios/start?hl=es-419>

**The Movie Database (TMDB).** *Getting Started with TMDB API.*

Disponible a:

<https://developer.themoviedb.org/docs/getting-started>

**CodezUp.** *Swift Firebase RealTime Database Integration.*

Disponible a:

<https://codezup.com/swift-firebase-real-time-database-integration/>

**Creole Studios.** *SwiftUI Firebase Authentication Guide.*

Disponible a:

<https://www.creolestudios.com/swiftui-firebase-authentication-guide/>

**Apple Inc.** *LocalAuthentication Framework.*

Disponible a:

<https://developer.apple.com/documentation/localauthentication/>

**Apple Inc.** *Logging a user into your app with Face ID or Touch ID.*

Disponible a:

<https://developer.apple.com/documentation/localauthentication/logging-a-user-into-your-app-with-face-id-or-touch-id>

**Apps Developer Blog.** *Integrate Touch ID and Face ID for biometric security in iOS.*

Disponible a:

<https://www.appsdeveloperblog.com/integrate-touch-id-and-face-id-for-biometric-security-in-ios/>

**Google LLC.** *Delete data from Cloud Firestore (Swift).*

Disponible a:

[https://firebase.google.com/docs/firestore/manage-data/delete-data?hl=es-419#swift\\_2](https://firebase.google.com/docs/firestore/manage-data/delete-data?hl=es-419#swift_2)

**Apple Inc.** *UITableViewDelegate Protocol.*

Disponible a:

<https://developer.apple.com/documentation/uikit/uitableviewdelegate>

**Apple Inc.** *Table Views.*

Disponible a:

<https://developer.apple.com/documentation/uikit/table-views>

**Stack Overflow.** *Add swipe to delete UITableViewCell.*

Disponible a:

<https://stackoverflow.com/questions/24103069/add-swipe-to-delete-uitableviewcell>

**Apple Inc.** *CharacterSet — whitespacesAndNewlines.*

Disponible a:

[https://developer.apple.com/documentation/foundation/characterset/whitespacesandnewline\\_s](https://developer.apple.com/documentation/foundation/characterset/whitespacesandnewline_s)

**Apple Inc.** *String — trimmingCharacters(in:).*

Disponible a:

[https://developer.apple.com/documentation/foundation/nsstring/trimmingcharacters\(in:\)/](https://developer.apple.com/documentation/foundation/nsstring/trimmingcharacters(in:)/)