

Suprem Tournaments

Documentación



Jorge
Llopis
Gómez

Índice

1. Abstact	4
2. Introducción	5
2.1 Objetivo	5
2.2 Justificación	5
2.3 Análisis de lo existente	5
3. Necesidades empresariales para el desarrollo	6
3.1 Recursos Humanos	6
3.1.1 Trabajadores	6
3.1.2 Contratos y puesto solicitado	7
3.1.3 Obligaciones en materia de Seguridad Social	12
3. 2 Prevención de riesgos laborales	12
3.2.1 Riesgos laborales	12
3.2.2 Medidas preventivas.	12
3.3 Iniciativa emprendedora:	13
3.3.1 Forma jurídica.	13
3.3.2 Trámites administrativos.	13
4. Requisitos funcionales de la aplicación	14
5. Análisis y diseño	14
5.1 Diagrama de arquitectura	14
5.2 Diagrama de casos de uso	15
5.3 Diagrama de clases	17
5.3.1 Clases Principales	17
5.3.2 Clases Android	17
5.3.2 Clases Windows	18
5.3.2.1 Convertidores	18
5.3.2.2 Objetos	18
5.3.2.3 Servicios	19
5.4 Diagrama de datos	21
5.5 Diseño de la interfaz de usuario	25
5.5.1 Diseño del programa de Windows	25
5.5.2 Diseño de la Aplicación Android	31
6. Codificación	32
6.1 Lenguajes de programación	32
6.2 Entornos de programación	33
6.3 Aspectos relevantes de la implementación	34
6.3.1 API	34
Suprem Tournaments Documentación	2

6.3.2 Aplicación Android	38
6.3.3 Programa de Windows	41
7. Manual de Usuario	43
7.1 Aplicación de Android	43
7.1.1 Lista de Torneos	43
7.1.2 Datos del Torneo seleccionado	45
7.2 Programa de Windows	46
7.2.1 Ventana Principal (sin login)	46
7.2.2 Torneos	47
7.2.3 Ver Torneo	48
7.2.4 Registrarse	50
7.2.5 Iniciar Sesión	51
7.2.6 Ventana principal del Gestor, Lista torneos	52
7.2.6 Ventana principal del Gestor Lista Participantes continuos.	53
7.2.7 Ventana editar torneo	55
7.2.8 Ventana Editar Participantes aceptados	55
7.2.9 Ventana Participantes Solicitados	57
7.2.10 Ventana Combates	58
8. Requisitos e instalación	60
8.1 API	60
8.2 Aplicación de Android	62
8.3 Programa de Windows	63
9 Conclusiones	67
9.1 Conclusiones sobre el trabajo realizado	67
9.2 Posibles ampliaciones y mejoras.	68
10. Bibliografía	69
11. Apéndice	69

1. Abstact

My project is about tournaments management.

Participants can use a Windows application to participate in one tournament, without having to log in.

Only tournament managers can log in. Managers have permission to:

- Create, edit and delete tournaments.
- Accept or deny participants. Managers can, also create, edit, delete and save participants.
- Create combats with accepted participants, create random combats with all accepted participants. Edit and delete existing combats.
- Create, edit and delete stand alone participant.

All of this is easy and intuitive for the manager.

The Windows application has been coded using WPF in MS Visual Studio.

The mobile application allows to get basic tournament information for the players. It has been coded using Kotlin in Android Studio.

This project is perfect to show my abilities and let me enjoy with my friends.

2. Introducción

2.1 Objetivo

El objetivo de este proyecto es que un gestor desde el programa de Windows pueda crear, editar y eliminar torneos, esto conlleva poder añadir solicitantes, aceptar solicitantes, editar solicitantes, eliminar solicitantes, añadir combates, editar combates.

Una persona que no se logue en el programa de Windows solo tendrá la opción de ver los torneos, ver la información general de estos, ver los solicitantes, ver los combates y solicitar concursar en ese torneo.

Desde una aplicación en Android solo podrá ver los torneos y la información general de este.

2.2 Justificación

Este proyecto lo elegí para poder exponer todas mis habilidades como programador y poder utilizar el resultado de este proyecto con mis amigos ya que siempre que tenemos tiempo libre hacemos algún que otro torneo.

Gracias a mi programa podremos crear, de una manera fácil e intuitiva, torneos y los demás podrán ver información desde el móvil.

2.3 Análisis de lo existente

La aplicación Android destaca por su fácil utilidad para usuarios que quieren ver los torneos.

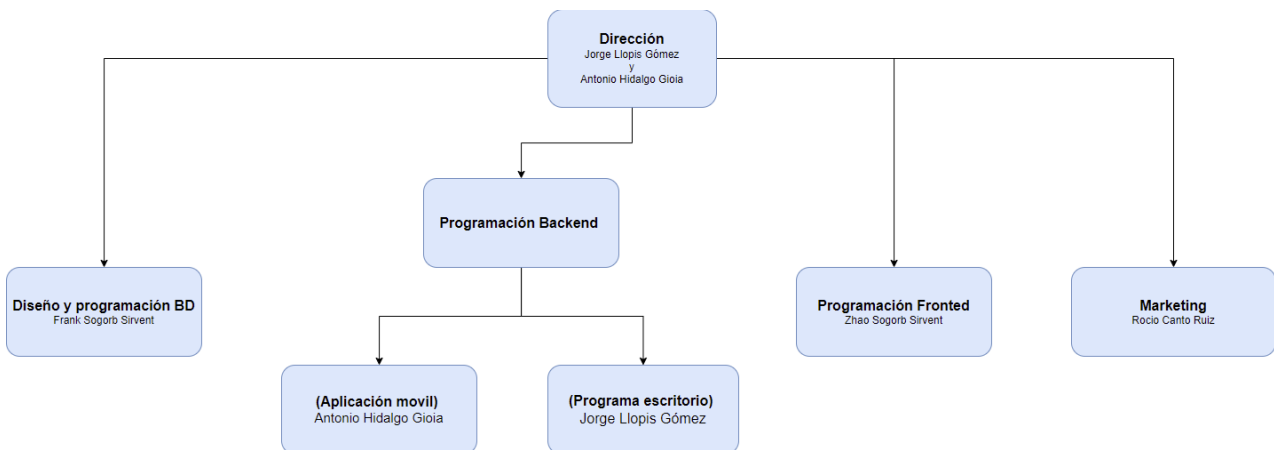
Gracias al programa de escritorio es posible crear y editar torneos fácilmente. Existe la posibilidad de poner en la entrada una pantalla para que la gente pueda solicitar concursar en un torneo, y también crear de una forma sencilla solicitantes y combates, editables.

3. Necesidades empresariales para el desarrollo

3.1 Recursos Humanos

3.1.1 Trabajadores

Organigrama



En el organigrama podemos observar que todo el equipo será dirigido por Jorge Llopis Gómez el cual también se encargará de la programación Backend del programa de escritorio (programado en C#).

Antonio Hidalgo Gioia se encargará de la programación Backend de la aplicación móvil, que será en Kotlin.

Frank Sogorb Sirvent se encargará del diseño y programación de la base de datos, la cual estará en MySQL.

Zhao Sogorb Sirvent se encargará de la programación Fronted de la aplicación móvil y de WPF.

Rocio Cantó Ruiz se encargará del marketing para llevar nuestro programa a consumidores target.

3.1.2 Contratos y puesto solicitado

Descripción del puesto	Condiciones laborales	Perfil profesional
Denominación <u>Jorge Llopis Gómez</u> Técnico Informático	Salario 1100€ Neto mes. 14 pagas	Formación y titulación Técnico Informático Grado superior desarrollo de aplicaciones multiplataforma
Departamento Programación Backend	Horario y jornada De 8:00 a 14:30 y de 18:00 a 21:00	Conocimiento específicos Programación en C#
Nivel responsabilidad Alto	Lugar de contrato <i>Suprem office</i>	Experiencias profesionales Trabajo en NTT Data
Tareas a realizar Programará todas la funcionalidades del programa de escritorio de Suprem Tournaments	Tipo de contrato Indefinido	Habilidades profesionales Buen desarrollo, cambios repentinos en el programa
		Actitudes y habilidades Habilidades de persuasión, capacidad para resolver problemas, actitud positiva

Descripción del puesto	Condiciones laborales	Perfil profesional
Denominación <u>Antonio Hidalgo Gioia</u> Técnico Informático	Salario 1100€ Neto mes. 14 pagas	Formación y titulación Técnico Informático Grado superior desarrollo de aplicaciones multiplataforma
Departamento Programación Backend	Horario y jornada De 8:00 a 14:30 y de 18:00 a 21:00	Conocimiento específicos Programación en Kotlin. Conocimiento de Android Studio.
Nivel responsabilidad Alto	Lugar de contrato <i>Suprem office</i>	Experiencias profesionales Trabajo en NTT Data
Tareas a realizar Programará todas la funcionalidades de la aplicación móvil de Suprem Tournaments	Tipo de contrato Indefinido	Habilidades profesionales Capacidad de adaptación fácilmente y racionalización
		Actitudes y habilidades Capacidad para añadir funcionalidades fácilmente

Descripción del puesto	Condiciones laborales	Perfil profesional
Denominación <u>Frank Sogorb Sirvent</u> Técnico Informático	Salario 1000€ Neto mes. 14 pagas	Formación y titulación Técnico Informático Grado superior desarrollo de aplicaciones multiplataforma
Departamento Programación y diseño de base de datos	Horario y jornada De 8:00 a 15:00	Conocimiento específicos Programación SQL.
Nivel responsabilidad Alto	Lugar de contrato <i>Suprem office</i>	Experiencias profesionales Trabajo en GTT
Tareas a realizar Programará todas la funcionalidades de la aplicación móvil de Suprem Tournaments	Tipo de contrato Indefinido	Habilidades profesionales Seguro de si
		Actitudes y habilidades Capacidad para enfocar errores desde otros puntos de vista, ayudar al equipo a hacer menos tediosa la tarea.

Descripción del puesto	Condiciones laborales	Perfil profesional
Denominación <u>Zhao Sogorb Sirvent</u> Técnico Informático	Salario 1000€ Neto mes. 14 pagas	Formación y titulación Técnico Informático Grado superior desarrollo de aplicaciones multiplataforma
Departamento Programación Fronted	Horario y jornada De 8:00 a 15:00	Conocimiento específicos Programación en WPF y en Android Studio.
Nivel responsabilidad Medio	Lugar de contrato <i>Suprem office</i>	Experiencias profesionales Trabajo en GTT
Tareas a realizar Programará la vista de la aplicación para móvil y escritorio. Encontrará la mejor paleta de colores y el mejor diseño para Suprem tournaments.	Tipo de contrato Temporal	Habilidades profesionales Seguro de sí mismo
		Actitudes y habilidades Actitud colaboradora. Actitud positiva.

Descripción del puesto	Condiciones laborales	Perfil profesional
Denominación <u>Rocio Cantó Ruiz</u> Técnico Informático	Salario 1000€ Neto mes. 14 pagas	Formación y titulación Marketing y Negocios
Departamento Marketing	Horario y jornada De 8:00 a 14:30	Conocimiento específicos Tratar con clientes de gran nivel profesional
Nivel responsabilidad Alto	Lugar de contrato <i>Suprem office</i>	Experiencias profesionales Trabajo en Indra
Tareas a realizar Encontrar clientes targeds	Tipo de contrato Temporal	Habilidades profesionales La mejor en persuadir
		Actitudes y habilidades Habilidad social, relaciones personales.

3.1.3 Obligaciones en materia de Seguridad Social

Los trabajadores contratados están dados de alta en la Seguridad Social, y contamos con el mejor plan de prevención de riesgos para la seguridad de todos los trabajadores/as de nuestra empresa.

La apertura de nuestro negocio la comunicamos a la Dirección Provisional de trabajo y con la ayuda de un notario registramos nuestra marca.

En nuestra empresa la comunicación es crucial para que nuestro desempeño sea óptimo. A los trabajadores los tratamos con el debido respeto, hay buen ambiente de trabajo, todo el mundo se lleva bien y nuestro rendimiento en el trabajo es bueno (un trabajador feliz es un trabajador eficiente).

3. 2 Prevención de riesgos laborales

3.2.1 Riesgos laborales

Los posibles riesgos laborales en los puestos de trabajo son:

- Fatiga visual o muscular.
- Contacto eléctrico.
- Carga mental.
- Dolor de espalda.
- Riesgo de golpes y choques.

3.2.2 Medidas preventivas.

Los trabajadores recibirán un curso en el que aprenderán a minimizar riesgos. También contarán con:

- Una buena ubicación de oficina con una adecuada distribución de ventanas y de luces artificiales.
- Una silla de trabajo inclinable, con soporte lumbar y ajustable a tu altura.
- Un reposapiés.
- Canaletas para ocultar cables y evitar caídas.

3.3 Iniciativa emprendedora:

3.3.1 Forma jurídica.

Somos una sociedad limitada porque era necesario unir fuerzas económicas para financiar la empresa.

Si se da el caso de necesitar más fuentes económicas podemos pedir un crédito. Cada mes guardaremos algo de dinero por si en algún momento no conseguimos llegar al resultado económico previsto. Gracias a esto podríamos seguir adelante sin problema por un tiempo hasta que nos podamos recuperar económicamente.

En nuestra sociedad Jorge Llopis invirtió unos 8.000€ y otros 8.000€ Antonio Hidalgo. Cada socio cuenta con un 50% de las acciones.

3.3.2 Trámites administrativos.

Lo primero que debemos hacer es registrar el nombre Suprem Tournaments SL solicitando la certificación negativa del nombre en el registro mercantil que corresponda por ubicación.

Necesitaremos abrir una cuenta bancaria a nombre de nuestra sociedad en el banco.

Los estatutos de la sociedad donde estarán nuestras normas deberemos solicitarlas en Hacienda, así como la obtención del NIF provisional, el Impuesto de Actividades Económicas y la declaración censal. Por último realizaremos el trámite para obtener el NIF definitivo, también en Hacienda.

Acudiremos al notario para así realizar la firma de la escritura pública de la creación de nuestra sociedad.

4. Requisitos funcionales de la aplicación

Mi aplicación de escritorio va dirigida a solventar la gestión de torneos de una forma sencilla y amigable, para que cualquier persona o entidad, siempre que esté registrada como gestor, pueda crear torneos de una forma rápida y sin dificultades.

Ofrecemos la posibilidad de poner la aplicación de escritorio en la entrada, para que los usuarios puedan solicitar participar en un torneo.

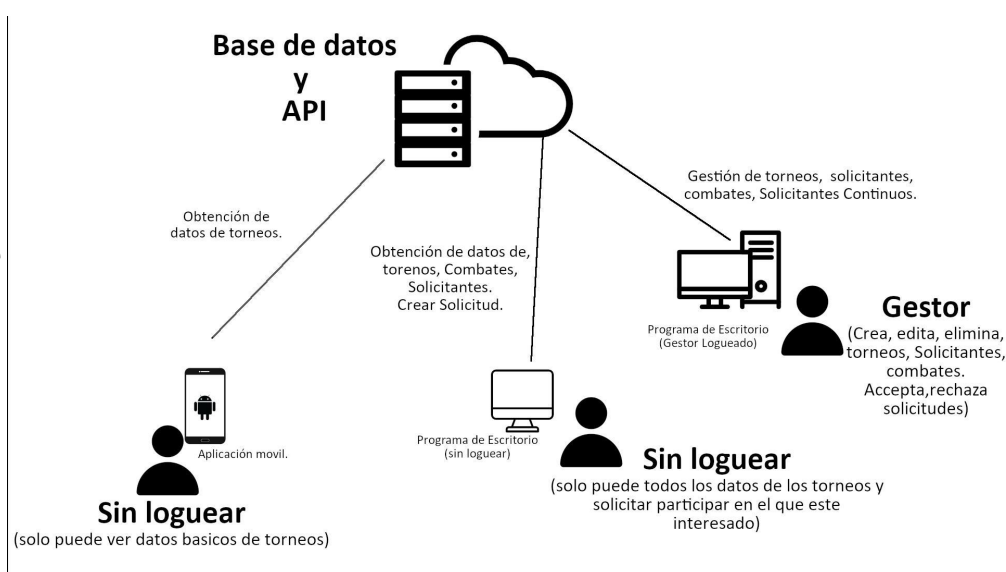
Existirá una aplicación móvil en la cual se podrá ver los torneos registrados en Suprem Tournaments.

5. Análisis y diseño

5.1 Diagrama de arquitectura

En el servidor Linux estará almacenada la base de Datos y la API, esta segunda gracias a Wildfly. Así se podrá responder peticiones de las aplicaciones.

La aplicación móvil en Android, solicitará los datos básicos de los torneos.

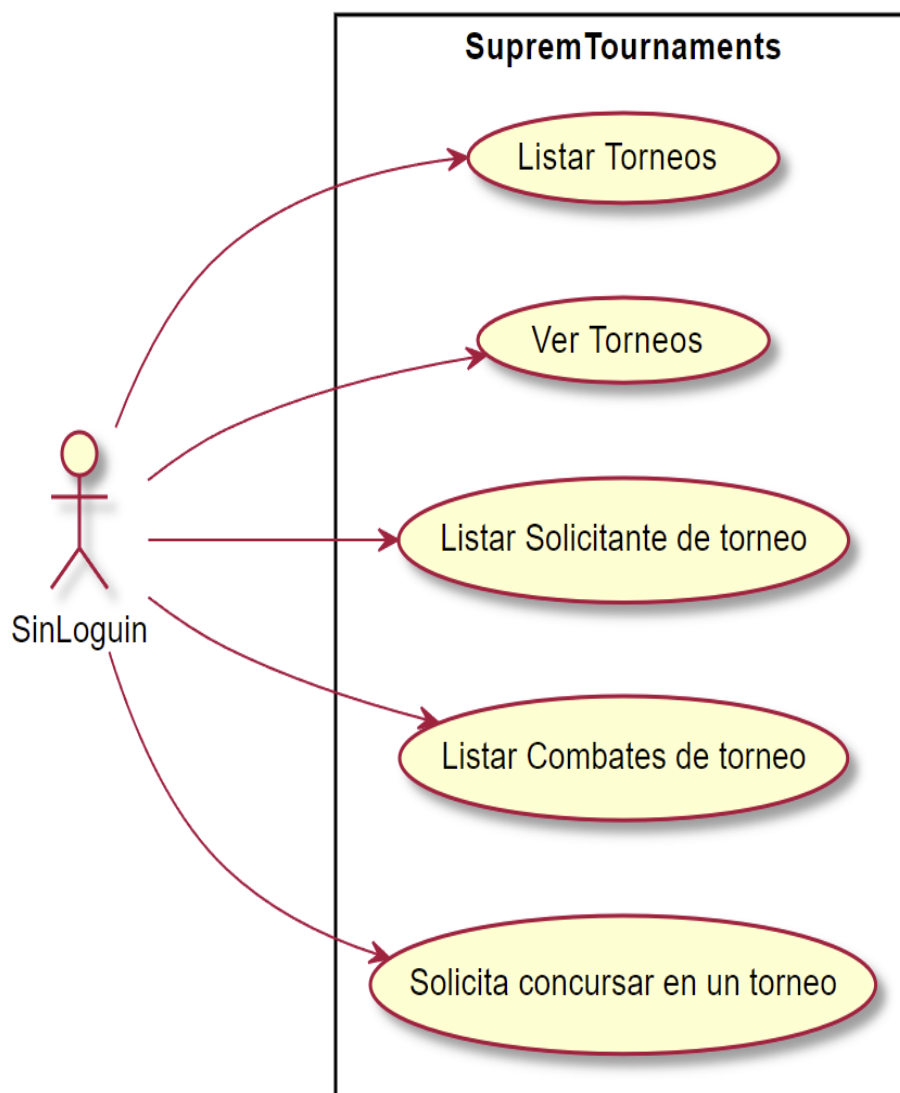


En el programa de escritorio, únicamente disponible en Windows creado con WPF, sin registrarse se podrá ver todos los torneos, sus datos, los solicitantes y los combates. Además de poder solicitar concursar en un torneo (Crear solicitud).

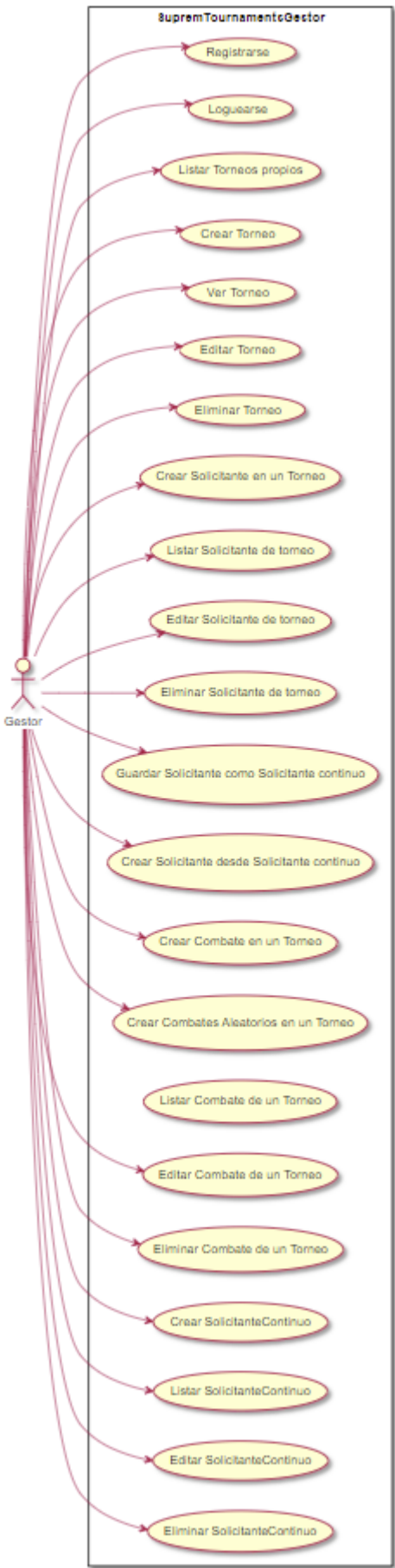
El programa de escritorio registrado, será un gestor, lo cual le da permiso para gestionar sus torneos, los solicitantes de sus torneos y los combates de sus torneos. También podrá gestionar Solicitantes Continuos.

5.2 Diagrama de casos de uso

La siguiente imagen muestra las opciones del sistema en caso de no estar logueado. Contamos con cinco opciones:



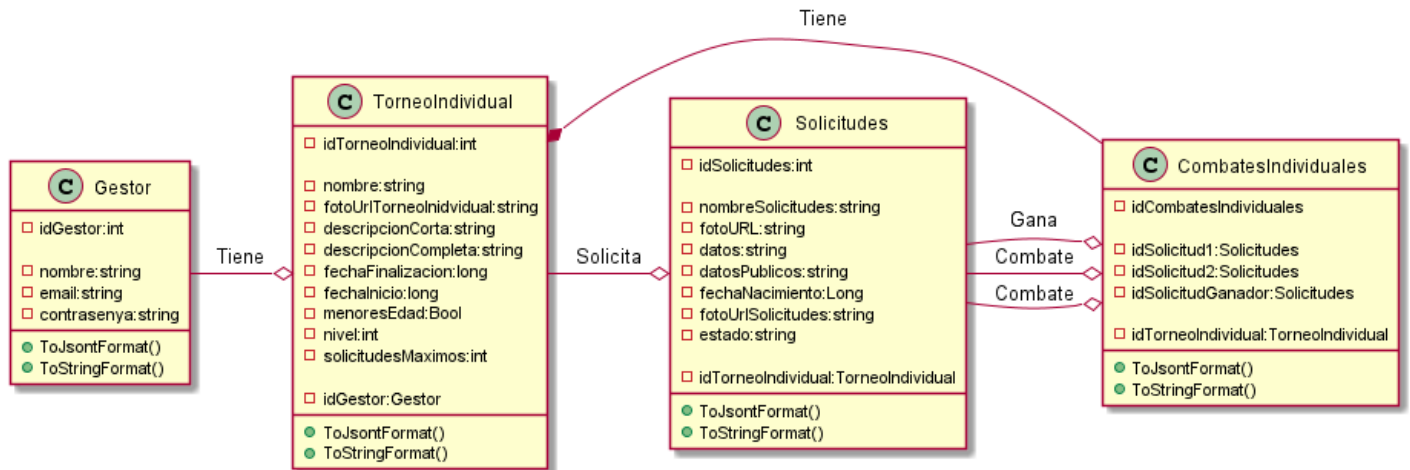
La siguiente imagen muestra las opciones del sistema. En caso de estar logueado contamos con veinte opciones.



5.3 Diagrama de clases

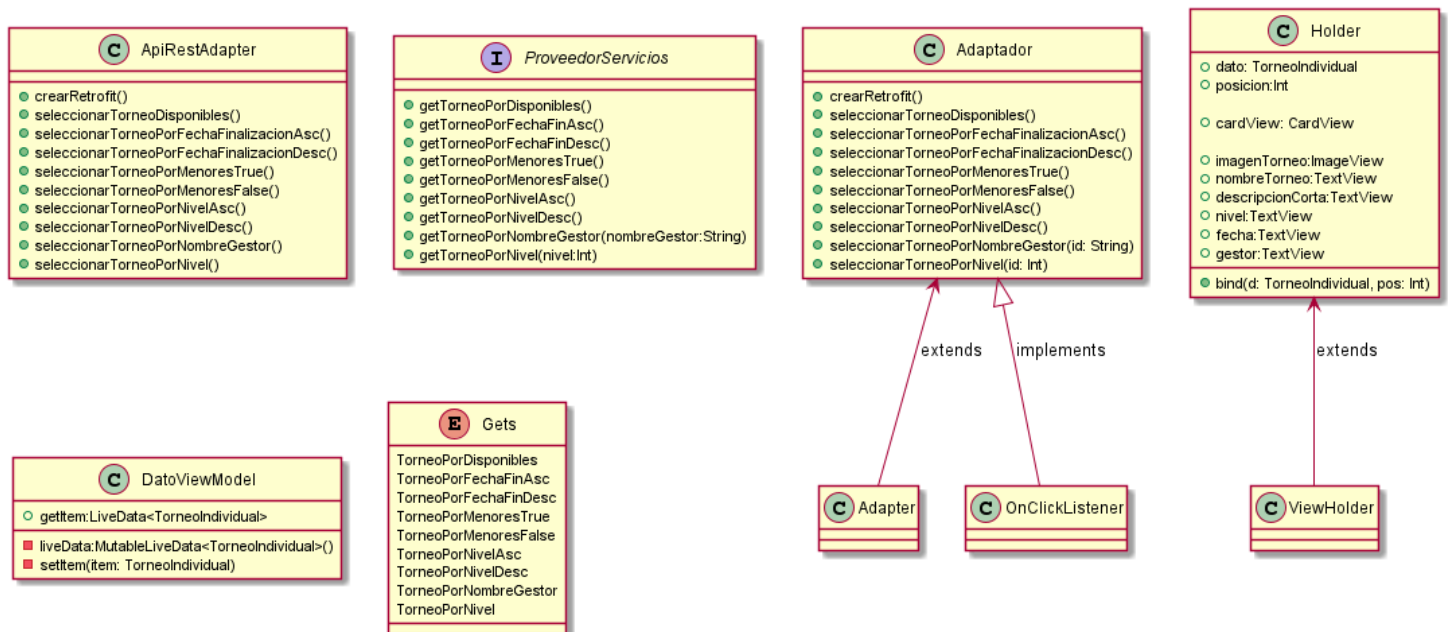
5.3.1 Clases Principales

Estas clases son las necesarias para almacenar los datos que nos suministra la API.



5.3.2 Clases Android

En el siguiente diagrama podemos observar todas las clases necesarias, aparte de las principales, que han sido usadas en la aplicación Android.

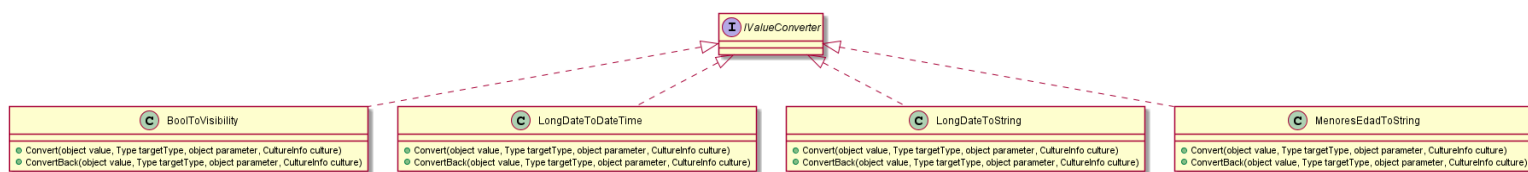


5.3.2 Clases Windows

En estos apartados se comentan las clases más importantes. No se incluyen las principales que se usan en todo momento.

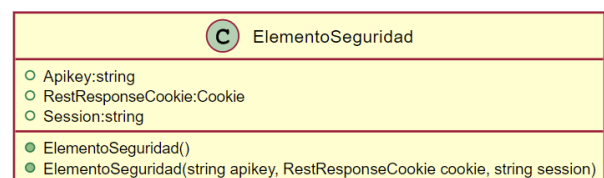
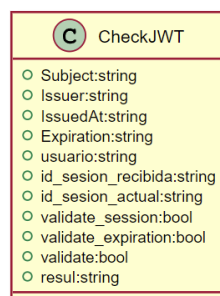
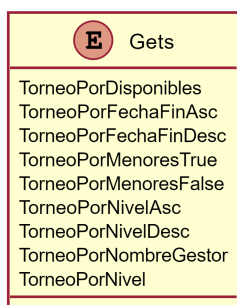
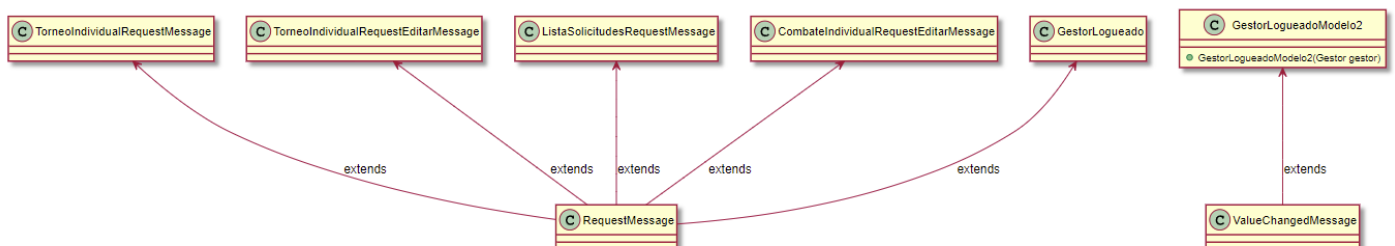
5.3.2.1 Convertidores

En el siguiente diagrama podemos observar los convertidores usados en el programa de Windows.



5.3.2.2 Objetos

En el diagrama que se observa a continuación podemos ver los objetos necesarios en el programa de Windows, aparte de los principales.



5.3.2.3 Servicios

A continuación se mostrará un diagrama donde podremos observar los servicios que han sido necesarios en el programa de Windows.



En el diagrama de abajo, podemos observar el servicio que nos comunicara con la API. Este tiene todos los tipos de request que le podemos hacer a la API. Aunque no se usen todos, son necesarios para posibles ampliaciones en el futuro.



5.4 Diagrama de datos

Tabla Gestor:

Esta tabla cuenta con:

Id gestor: columna para identificar al gestor.

Nombre: columna que guardará el nombre del gestor.

Contraseña: columna donde se guardará la contraseña en SHA256.



Tabla torneos_individuales:

Esta tabla cuenta con:

Id torneo individual: columna para identificar el torneo individual.

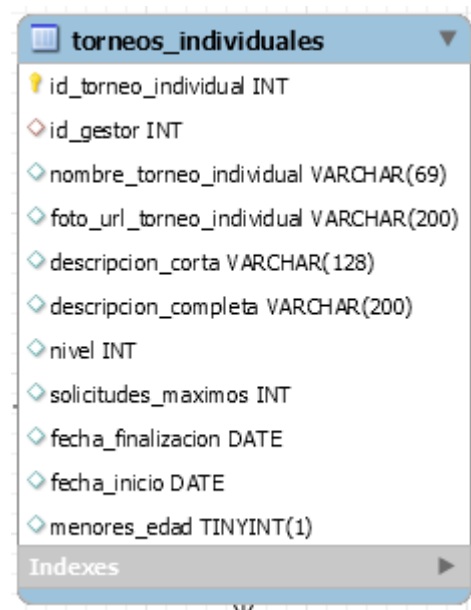
Id gestor: columna para saber qué gestor es propietario de dicho torneo.

Nombre torneo individual: el nombre del Torneo.

Foto URL: columna para almacenar la foto en url.

Descripción corta: columna para una descripción más simple necesarias en vistas sin mucho espacio.

Descripción completa: descripción con más detalle para vistas que permitan mostrar más información.



Fecha de finalización: fecha desde donde no se aceptarán más solicitudes.

Fecha de inicio: fecha de inicio del torneo.

Menores edad: esta columna será usada para saber si los menores de edad pueden concursar en dicho torneo o no.

Tabla solicitudes:

Esta tabla cuenta con:

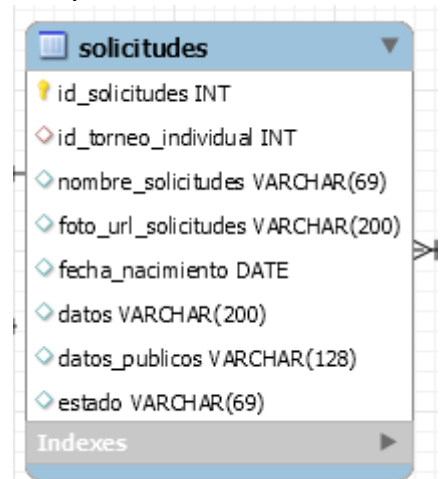
Id solicitudes: columna para identificar la solicitud.

Id torneo individual: columna para saber a qué torneo concursa el solicitante.

Nombre solicitudes: es el nombre del solicitante.

Foto url solicitudes: foto del solicitante en formato url.

Fecha de nacimiento: fecha de nacimiento del solicitante



Datos: en esta columna están los datos que solo serán visibles para el gestor, para indicar algo en concreto o un resumen de datos públicos.

Datos públicos: en esta columna están los datos que serán visibles para todos.

Estado: esta columna servirá para saber si ha sido aceptado en el torneo, rechazado, o si aún no han decidido nada.

Tabla combates individuales:

Esta tabla cuenta con:

Id combates individuales: columna para identificar el combate.

Id solicitud1: id del solicitante que será un contrincante.

Id solicitud2: id del solicitante que será el otro contrincante.

Id solicitud ganador: id del solicitante que es el ganador.



Tabla (tabla plantilla) solicitudes continuas:

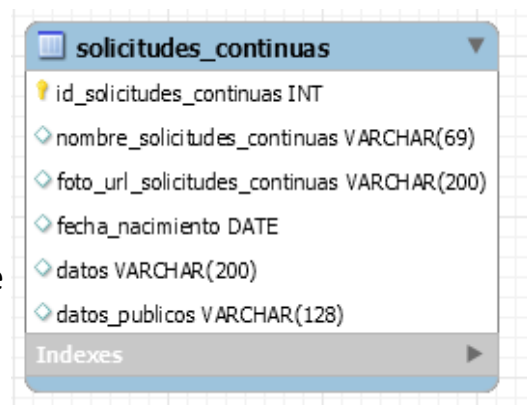
Esta tabla cuenta con:

Id solicitudes continuas: columna para identificar solicitudes continuas.

Nombre solicitudes continuas: es el nombre de la solicitud continua.

Foto url solicitudes continuas: foto del solicitante continuo en formato url.

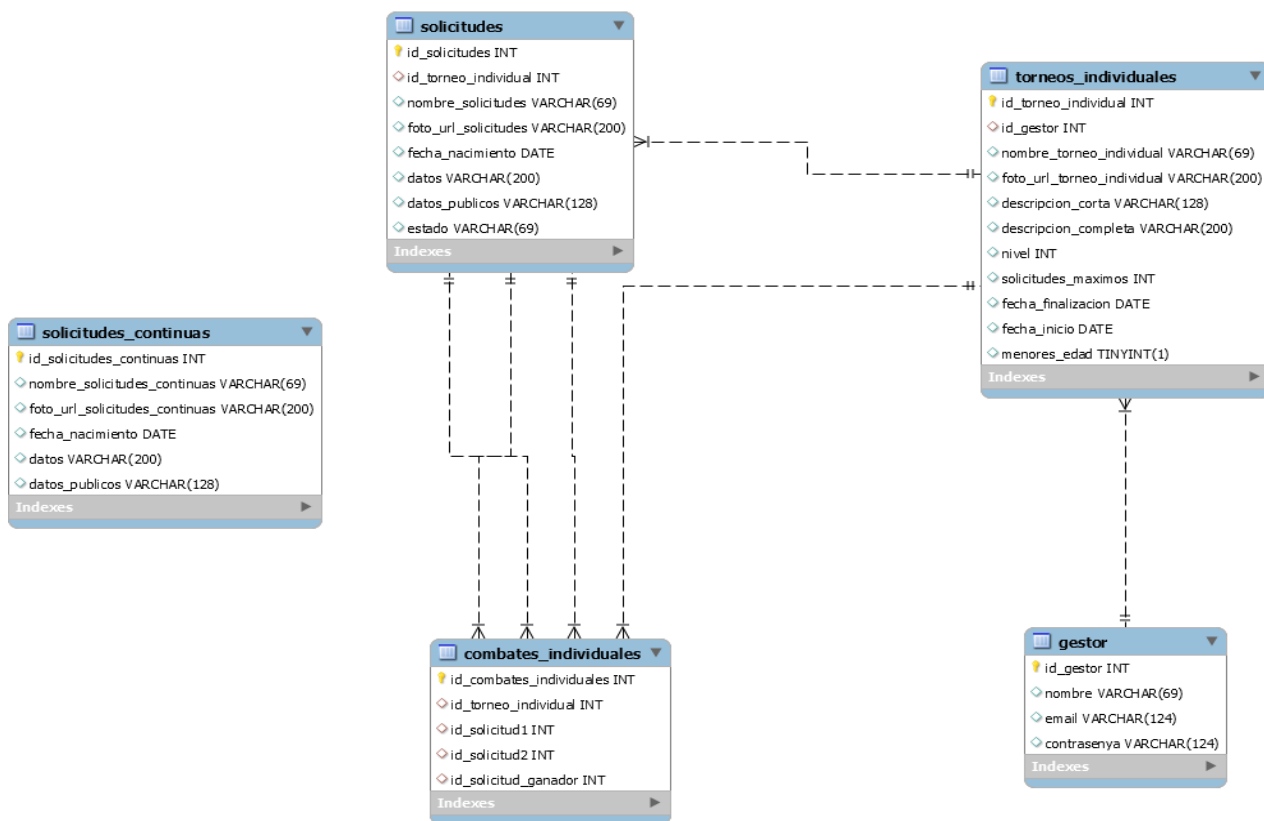
Fecha de nacimiento: fecha de nacimiento de la solicitud continua.



Datos: en esta columna están los datos que solo serán visibles para el gestor, para indicar algo en concreto o un resumen de datos públicos.

Datos públicos: en esta columna están los datos que serán visibles para todos.

Veamos el esquema completo:



Podemos observar que:

Un gestor puede tener varios torneos.

Un torneo puede tener varios solicitantes.

Un torneo puede tener varios combates.

Un solicitante puede ser varias veces un solicitud1.

Un solicitante puede ser varias veces un solicitud2.

Un solicitante puede ser varias veces un solicitudGanador.

Un solicitante solo puede participar en un torneo.

Solicitudes continuas se usa como tabla plantilla.

5.5 Diseño de la interfaz de usuario

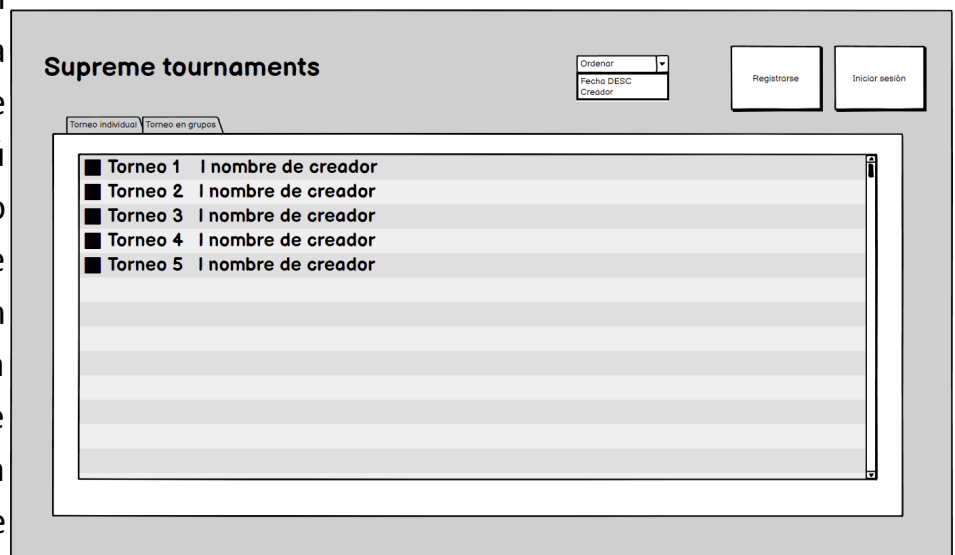
Cuando la idea principal quedó clara, se creó un prototipo con Balsamic, aunque dicho prototipo tenía la característica de Torneos Grupales que finalmente fue borrada (esta no será mostrada).

Lo más destacado será un menú lateral en el programa de Windows. Esto conlleva cambios en la interfaz final.

5.5.1 Diseño del programa de Windows

Vista principal (sin login)

Podemos observar que el diseño para la pantalla principal cambió, ya que se decidió poner un menú lateral izquierdo pudiendo poner las opciones de registrarse e iniciar sesión en esta. La vista también ha cambiado, porque ahora se usan CardViews para una mejor vista. Este cambio se debe a que esta parte del programa podría estar en pantallas táctiles.



Vista Torneo datos básicos (sin login)

Podemos observar que en el programa se ha mantenido el mismo diseño, pero con más información.

Vista Torneo datos participantes (sin login)

Podemos observar que en el programa finalmente se agregó un botón para refrescar datos. Igual que en la ventana de torneos, la vista de cada participante fue cambiada por una mayor comodidad.

Vista Formulario solicitar participar (sin login)

En la versión final se agregó la opción de cancelar por si se da el caso de pulsar sin querer el botón de la ventana anterior. La propiedad apellidos fue eliminada ya que se puede poner en la propiedad nombre.

Vista Resultados (sin login)

Esta vista se renombró por combates, ya que es mucho más lógico.

The screenshot shows a web interface titled "Datos torneo". At the top, there are three tabs: "Torneo", "Participantes", and "Resultados", with "Resultados" being the active tab. Below the tabs is a table with five rows of tournament results. Each row contains: "Participante", "vs", "Participante", "=", and "Participante Ganador". Below the table is a "Refrescar" (Refresh) button.

Participante	vs	Participante	=	Participante Ganador
Participante	vs	Participante	=	Participante Ganador
Participante	vs	Participante	=	Participante Ganador
Participante	vs	Participante	=	Participante Ganador
Participante	vs	Participante	=	Participante Ganador
Participante	vs	Participante	=	Participante Ganador

Refrescar

Vista Registrarse

Esta vista se pudo mantener igual.

The screenshot shows a registration form with four input fields: "Nombre", "Email", "Contraseña", and "Repite contraseña". Below the fields is a "Registrarse" button.

Vista iniciar sesión

Esta vista se pudo mantener igual.

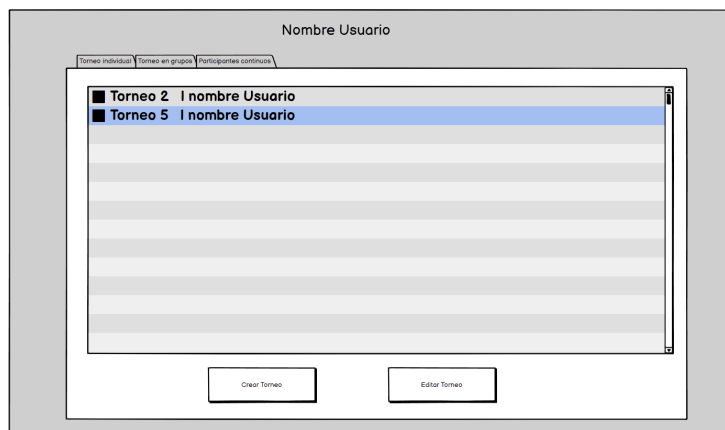
The screenshot shows a login form with two input fields: "Email" and "Contraseña". Below the fields is an "Iniciar Sesión" button.

Después de la vista anterior saldría esta, pero finalmente no fue incluida.

The screenshot shows a view titled "Sesion Iniciada" with a single "Entrar" button.

Vista principal (Gestor) Mis torneos

En esta vista se agregó la opción de eliminar un torneo y se quitó el botón para editar un torneo, ya que se utiliza el menú lateral para incluir dicha opción.

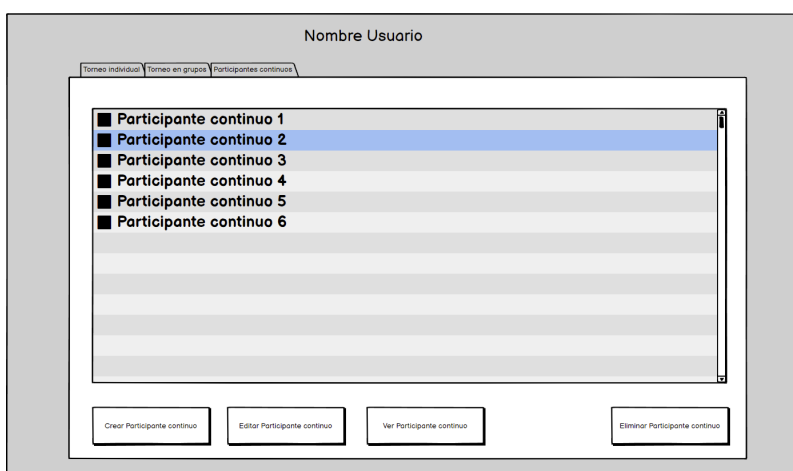


Vista crear torneo (Gestor)

Esta vista ha sufrido algunos cambios ya que no se puede agregar participantes o combates. Ahora solo se ingresan los datos básicos del torneo. Después se editará pudiendo añadir participantes y combates. También se añadió algún campo más y el botón cancelar

Vista principal (Gestor). Participantes Continuos

En esta, se eliminó la opción de ver Participante continuo y la de editar fue cambiada por guardar cambios seleccionados, ya que se aprovechó el data grid para editar las propiedades de cada objeto.



Nuevo Participante

Nombre

Apellidos

Más datos

Más datos (público)

Fecha de Nacimiento

Crear Participante

Nuevo participante continuo (Gestor).

Esta continúa de la misma forma, pero se eliminó el campo apellidos ya que se puede poner en nombre. También se añadió el botón cancelar.

Vista editar datos básicos torneo (Gestor).

Esta vista se mantiene igual pero se añadieron más campos (como el nivel de dificultad).

Nombre Torneo

Nombre Usuario

Editar Torneo

Nombre

Descripción corta

Descripción completa

Fecha finalización

☐ Menores de edad permitida

Guardar Cambios

Nombre Torneo

Nombre Usuario

Editar Torneo | Participantes | Solicitudes participantes | Resultados

- Participante 1
- Participante 2
- Participante 3
- Participante 4
- Participante 5
- Participante 6

Añadir Participante Continuo

Crear Participante

Ver Participante

Eliminar Participante

Editar Participante

Guardar como Participante Continuo

Vista editar participantes (Gestor).

En esta vista fue cambiado el botón editar ya que se pueden cambiar los datos desde la misma. También se eliminó el botón ver participante por falta de tiempo, las opciones de abajo fueron reordenadas.

Vista añadir participante continuo (Gestor).

Esta se mantuvo igual, pero también se agregó el botón cancelar.

The screenshot shows a web form titled "Añadir Participante Continuo". It features a list of participants: "Participante 1", "Participante 2" (highlighted in blue), "Participante 3", "Participante 4", "Participante 5", and "Participante 6". Below the list is a "Añadir participante" button.

The screenshot shows a web form titled "Nuevo Participante". It includes input fields for "Nombre", "Apellidos", "Máx datos", "Máx datos (public)", and "Fecha de Nacimiento". There is a "Crear Participante" button at the bottom.

Vista crear participantes (Gestor).

Esta se mantuvo igual, aunque también se agregó el botón cancelar.

Vista aceptar Solicitudes (Gestor).

Esta se mantuvo igual.

The screenshot shows a web form titled "Nombre Torneo". It has a tabbed interface with tabs for "Editar Torneo", "Participantes", "Solicitudes participantes", and "Resultados". The "Solicitudes participantes" tab is active, showing a list of requests: "Solicitud Participante 1", "Solicitud Participante 2" (highlighted in blue), "Solicitud Participante 3", "Solicitud Participante 4", "Solicitud Participante 5", and "Solicitud Participante 6". Below the list are three buttons: "Aceptar participante", "Rechazar", and "Rechazar participante".

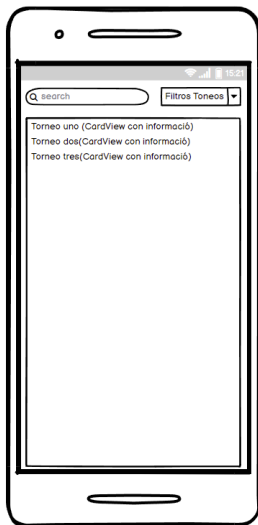
Vista resultados (Gestor).

Ahora renombrada combates, cambió el diseño. Ahora editas a través de un botón

que cambia la vista y te muestra 3 listas en las que eliges los participantes que van a combatir y el ganador. Se añadió la opción *editar*. Contrincantes aleatorios sin repetir se cambió por *Combates aleatorios*.

El botón *nuevo combate* ahora te muestra una vista parecida a editar en vez de añadirte un elemento más a la lista.

5.5.2 Diseño de la Aplicación Android



Vista principal.

Esta interfaz no sufrió cambios.

Aunque en esta no se especifica el diseño del Cardview, ya que la paleta aún no estaba elegida.

Vista datos torneo.

Esta vista fue modificada igual que la anterior.

6. Codificación

6.1 Lenguajes de programación

Los lenguajes de programación usados en Suprem Tournaments:

SQL: Necesario para crear una base de datos relacional. Fue elegido porque es en el que más conocimientos tengo para la creación de bases de datos.

Java: Usado para programar la API de Suprem Tournaments. Fue seleccionado ya que es el que más conocimientos tengo para creación de APIS.

Kotlin: Necesario para programar el Backend de la aplicación móvil en Android. Escogí Kotlin porque es necesario para programar en el famoso IDE de Google.

XML: Necesario en la parte Frontend de la aplicación Android, sencillo de entender y es usado en el IDE de Android.

C#: Usado para programar la parte de Backend en el programa de Escritorio de Windows, C#. Fue escogido gracias a todas las facilidades que ofrece Microsoft en todo su ecosistema (como WPF).

XAML: Usado en la parte de Fronted del programa de Escritorio. Necesario para WPF.

Los IDEs usados para programar los lenguajes de programación comentados, están en el siguiente punto.

6.2 Entornos de programación



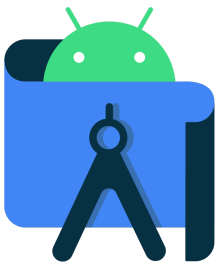
MySQL WorkBench: IDE usado para programar la base de datos.

Apache Netbeans: IDE usado para programar la API la cual se comunica con la base de datos. Además usé distintas librerías:



- MySQL Connector: Librería que permitirá tener una comunicación con la base de datos.
- JSON.simple: Librería que nos permite tratar con formato Json de una manera sencilla.
- Jjwt: Librería que nos ayudará a implementar seguridad basada en jwt.
- Swagger jaxrs2: Librería usada para hacer una documentación atractiva de la API la cual también nos permite hacer todo tipo de peticiones HTTP, gracias a sus propias anotaciones.

Android Studio: IDE usado para programar la aplicación móvil de Android. También fue usada la librería:



- Picasso: Librería que nos facilitará la obtención de imágenes de internet mediante una url, y muchas más funcionalidades.

Visual Studio Enterprise 2019 : Necesario para programar el programa de escritorio de Windows. Además también se han usado los siguientes Toolkits:



◦ Materia Desing: los dos toolkits MaterialDesingColors y MaterialDesingThemes, usados para darle una vista novedosa y limpia al Frontend del programa de escritorio.

- Azure: Core, Storage.Blobs y Storage.Common, han sido usados para poder almacenar imágenes en un contenedor Blob.
- Newtonsoft.Json: nos permite serializar a Json y deserializar objetos desde Json.
- RestSharp: usado para hacer peticiones a la API de una forma sencilla.
- Mvvm: es usado para la comunicación entre vistas y cambios de propiedades de objetos.

6.3 Aspectos relevantes de la implementación

6.3.1 API

Creación de más queries:

Pensando en el futuro creé varias queries para que sea posible usarlas mediante ServiceRest utilizando las aplicaciones. De esta forma podré usarlas en un combobox en WPF o en un Spinner en Android.

En la imagen que aparece a continuación podemos ver las queries de torneos individuales que serán las que usará el usuario.

```
@NamedQuery(name = "TorneosIndividuales.findByMenoresEdad", query = "SELECT t FROM TorneosIndividuales t WHERE t.menoresEdad = :menoresEdad"),
@NamedQuery(name = "TorneosIndividuales.findByNombreGestor", query = "SELECT t FROM TorneosIndividuales t WHERE t.idGestor.nombre = :nombreCreador"),
@NamedQuery(name = "TorneosIndividuales.findByEmailGestor", query = "SELECT t FROM TorneosIndividuales t WHERE t.idGestor.email = :emailCreador"),

@NamedQuery(name = "TorneosIndividuales.findByFechaFinalizacionDesc", query = "SELECT t FROM TorneosIndividuales t ORDER BY t.fechaFinalizacion desc"),
@NamedQuery(name = "TorneosIndividuales.findByFechaFinalizacionAsc", query = "SELECT t FROM TorneosIndividuales t ORDER BY t.fechaFinalizacion asc"),
@NamedQuery(name = "TorneosIndividuales.findByNivelDesc", query = "SELECT t FROM TorneosIndividuales t ORDER BY t.nivel desc"),
@NamedQuery(name = "TorneosIndividuales.findByNivelAsc", query = "SELECT t FROM TorneosIndividuales t ORDER BY t.nivel asc"),
@NamedQuery(name = "TorneosIndividuales.findByPlazasLibres", query = "SELECT t FROM TorneosIndividuales t WHERE size(t.solicitudesCollection) < t.solicitudesMaximos"),
@NamedQuery(name = "TorneosIndividuales.findByPlazasLibresAndAfterDate", query = "SELECT t FROM TorneosIndividuales t WHERE size(t.solicitudesCollection) < t.solicitudesMaximos")
```

Uso de la librería Swagger:

Gracias a la librería Swagger podremos tener una documentación de la API en la cual podremos probar GET, PUT, POST, DELETE.

Podemos ver el resultado en la imagen de abajo. Si nos fijamos bien, se observa que cada uno tiene una documentación.



Como ya comenté, gracias a las anotaciones que nos provee Swagger, podremos hacer la documentación de la API. Veamos cómo.

Encima de las clases ServiceRest (servicio que tendrá todos las url para acceder a la API). Veamos el código para conseguir este resultado:



```

@OpenAPIDefinition(
    info = @Info(
        description = "API REST sobre la BD de suprem tournaments",
        version = "1.0.0",
        title = "Swagger Suprem Tournaments"
    )
)
@Tag(name = "SolicitudesContinuas",
    description = "Datos de las Solicitudes continuas")
@Server(url = "/api/supremtournaments")

```

Podemos observar la información básica de la API en la anotación `@OpenAPIDefinition`, es decir la descripción de la API, la versión y el título.

`@Tag` para documentar el servicio.

`@Server` para indicarle a Swagger desde que base deberá de hacer la petición.

Ahora veamos la información que podemos agregar a una petición:

Para poner la información de la cabecera y la de abajo, usaremos la notación `@Operation`.

```

@Operation(
    summary = "Datos de solicitud continua",
    description = "Obtiene los datos de una solicitud continua en concreto"
)

```

Si nos fijamos, Swagger también nos indica qué parámetro podemos rellenar indicándonos que el parámetro `id` es obligatorio, ya que forma parte de la url. Para esto no nos hace falta poner ninguna anotación, ya que Swagger lo hará solo. Ocurre lo mismo con el tipo de petición (GET, PUT, POST, DELETE), ya que identificará las anotaciones de javax.

La anotación `@Path` también la detecta Swagger. Para poder hacer la petición hay que añadir antes la anotación ya comentada `@Server`

```

@GET
@Path("/{id}")
@Produces(MediaType.APPLICATION_JSON)
public Response getOne(@PathParam("id") int id, @QueryParam("apikey") String token)

```

También podemos especificar qué errores pueden pasar y el esquema de la clase

Responses		
Code	Description	Links
200	Acción realizada con éxito	No links
	Media type <input type="text" value="application/json"/> <small>Controls Accept header.</small> Example Value Schema	
	<pre>{ "idSolicitudesContinuas": 0, "nombreSolicitudesContinuas": "string", "fotoUrlSolicitudesContinuas": "string", "fechaNacimiento": "2022-04-29T18:14:43.751Z", "datos": "string", "datosPublicos": "string" }</pre>	
400	Error al procesar la petición	No links
404	No hay datos que mostrar	No links

Esto lo lograremos con la anotación `@ApiResponse`

```

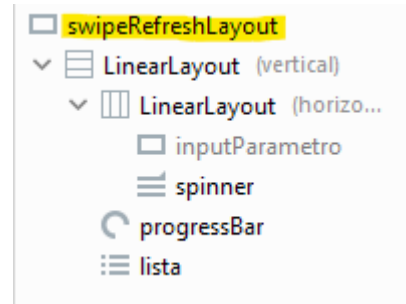
@ApiResponse (
    responseCode = "200",
    description = "Acción realizada con éxito",
    content = {
        @Content(mediaType = "application/json",
            schema = @Schema(implementation = SolicitudesContinuas.class)
        )
    }
)
@ApiResponse(responseCode = "204", description = "No hay datos que mostrar")
@ApiResponse(responseCode = "400",
    description = "Error al procesar la petición")
@GET
@Produces(MediaType.APPLICATION_JSON)

```

6.3.2 Aplicación Android

Uso de SwipeRefreshLayout:

Para que la aplicación sea mucho más interactiva y sencilla de usar para el usuario, use como layout principal `swipeRefreshLayout`, gracias a este layout podremos saber si el usuario desliza de arriba a abajo, y de esta manera actualizar los datos que está viendo.



Veamos el Backend:

```
//Con este trozo de código puede indicar que hacer al usar el swipe (deslizar el dedo de arriba a abajo)
binding.swipeRefreshLayout.setOnRefreshListener {
    cargarDatos(opcionSeleccionada)
    binding.swipeRefreshLayout.isRefreshing = false
}
```

Como podemos observar, cuando el escuchador detecte dicha acción ejecutará la función `cargarDatos` y dejará de mostrar la típica ruedecita de carga de Android.

Aclaración de la función cargar datos: esta función ha sido necesaria ya que dependiendo de la opción seleccionada en el Spinner hará un get u otro.

Opciones Spinner:

Las opciones del Spinner son obtenidas desde la creación de un adaptador de array tipo string desde un xml. Veamos el XML:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="opcionesGet">
        <item>Disponibles</item>
        <item>Fecha Finalización Asc</item>
        <item>Fecha Finalización Desc</item>
        <item>Menores y Adultos</item>
        <item>Adultos</item>
        <item>Nivel Asc</item>
        <item>Nivel Desc</item>
        <item>Nombre Gestor</item>
        <item>Nivel</item>
    </string-array>
</resources>
```

El backend para obtener los datos de mi recurso:

```
var opcionesSpinner = resources.getStringArray(R.array.opcionesGet)
binding.spinner.adapter = ArrayAdapter<String>(requireContext(), android.R.layout.simple_list_item_1, opcionesSpinner)
```

Opciones Nombre Gestor y Nivel Spinner, InputText:

El spinner tiene dos opciones, las cuales nos van a hacer visible un InputText tipo texto o numérico dependiendo de cual de esas dos opciones elijamos (también cambiará el texto del hint).

Veamos el código :

```
7 ->{
    opcionSeleccionada = Gets.TorneoPorNombreGestor
    binding.inputParametro.setVisibility(View.VISIBLE)
    binding.inputParametro.hint = opcionesSpinner.get(seleccionado)
    binding.inputParametro.inputType = InputType.TYPE_CLASS_TEXT
}

8 ->{
    opcionSeleccionada = Gets.TorneoPorNivel
    binding.inputParametro.setVisibility(View.VISIBLE)
    binding.inputParametro.hint = opcionesSpinner.get(seleccionado)
    binding.inputParametro.inputType = InputType.TYPE_CLASS_NUMBER
}
```

A diferencia de las otras opciones, esta no ejecuta la función cargar datos directamente (ya que aún no ha sido introducido el nivel o el nombre del gestor).

Para que el Input Text sea más amigable con el usuario y poder hacer una petición a la API con los datos introducidos por el usuario, no hará falta hacer un tab en un botón sino que será posible pulsar enter. Veamos el código para esta funcionalidad:

```
//Hace un get cuando se pulsa el enter del teclado de Android, también muestra una Snackbar si esta vacío
binding.inputParametro.setOnKeyListener(View.OnKeyListener { view, i, keyEvent ->
    if (i == KeyEvent.KEYCODE_ENTER && keyEvent.action == KeyEvent.ACTION_UP) {
        if (!binding.inputParametro.text.isNullOrBlank()) {
            when(seleccionado){
                7 -> {
                    opcionSeleccionada = Gets.TorneoPorNombreGestor
                    nombreGestor = binding.inputParametro.text.toString()
                    cargarDatos(opcionSeleccionada)
                }
                8 -> {
                    opcionSeleccionada = Gets.TorneoPorNivel
                    nivelTorneoIndividual = Integer.parseInt(binding.inputParametro.text.toString())
                    cargarDatos(opcionSeleccionada)
                }
            }
        } else {
            Snackbar.make(binding.root, "Es necesario introducir un " + binding.inputParametro.hint, Snackbar.LENGTH_INDEFINITE)
                .setAction(text: "Aceptar", View.OnClickListener { })
                .setBackgroundTint(Color.rgb( red: 118, green: 0, blue: 0))
                .setTextColor(Color.WHITE)
                .show()
        }
        true
    }
    false ^OnKeyListener
})
```

Si detecta que ha pulsado enter y lo está dejando de pulsar, revisa cuál opción del spinner está seleccionada, hará un get u otro. Como podemos observar, si no hay nada en el InputText nos saldrá una snackbar que nos indicará que deberemos ingresar el parámetro.

Progressbar personalizada:

Mientras se espera respuesta de la API será visible una ProgressBar personalizada.

Cuando termina la corrutina, se ocultara la Progressbar.

A la derecha, la parte frontend de la ProgressBar y abajo el resultado

```
<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyle"
    android:layout_width="133dp"
    android:layout_height="160dp"
    android:layout_gravity="clip_horizontal|center|clip_vertical"
    android:layout_marginTop="200dp"
    android:indeterminateDrawable="@drawable/pantalla_carga2"
    android:visibility="gone" />
```



Veamos el código del Backend:

```
fun cargarDatos(gets: Gets) {
    CoroutineScope(Dispatchers.Main).async { this: CoroutineScope
        binding.progressBar.visibility = View.VISIBLE
        binding.lista.visibility = View.GONE

        when(gets){
            Gets.TorneoPorDisponibles -> torneos = ApiRestAdapter.seleccionarTorneoDisponibles().await()
            Gets.TorneoPorFechaFinAsc -> torneos = ApiRestAdapter.seleccionarTorneoPorFechaFinalizacionAsc().await()
            Gets.TorneoPorFechaFinDesc -> torneos = ApiRestAdapter.seleccionarTorneoPorFechaFinalizacionDesc().await()
            Gets.TorneoPorMenoresTrue -> torneos = ApiRestAdapter.seleccionarTorneoPorMenoresTrue().await()
            Gets.TorneoPorMenoresFalse -> torneos = ApiRestAdapter.seleccionarTorneoPorMenoresFalse().await()
            Gets.TorneoPorNivelAsc -> torneos = ApiRestAdapter.seleccionarTorneoPorNivelAsc().await()
            Gets.TorneoPorNivelDesc -> torneos = ApiRestAdapter.seleccionarTorneoPorNivelDesc().await()
            Gets.TorneoPorNombreGestor -> torneos = ApiRestAdapter.seleccionarTorneoPorNombreGestor(nombreGestor).await()
            Gets.TorneoPorNivel -> torneos = ApiRestAdapter.seleccionarTorneoPorNivel(nivelTorneoIndividual).await()
        }
        cargarAdaptador()
        binding.progressBar.visibility = View.GONE
        binding.lista.visibility = View.VISIBLE
    }
}
```

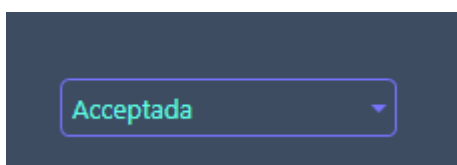

6.3.3 Programa de Windows

Para que el programa tenga una interfaz amigable, el gestor puede editar los datos desde el mismo UserControl usado para ver los solicitantes, haciendo un click sobre el campo que quieres cambiar y poniendo lo que necesite. Esto ha sido posible gracias al objeto solicitante, que hereda de ObservableObject y al grid que nos permite modificar cada columna para que pueda tener nuestro diseño personalizado.

Imagen	Nombre	Fecha nacimiento	Datos	Datos publicos	Estado
	Desdmon	18/07/2002	Desdmon conocido t	Siempre intentara contraatacar contra	Acceptada
	Marcos	05/04/2002	Marcos conocido tar	Muerte a Paskyyyyy	Acceptada
	Copo Nieve	20/09/2002	Un guerrero de nieve	Guerrero caido en la ultima guerra	Acceptada
	TheHamster	06/01/2015	Soy alergico al polvo	Preparado para la guerra.	Acceptada

Buttons: Crear Participante, Guardar cambios seleccionado, Refrescar, Añadir participante continuo, Guardar seleccionada como participante continuo, Eliminar

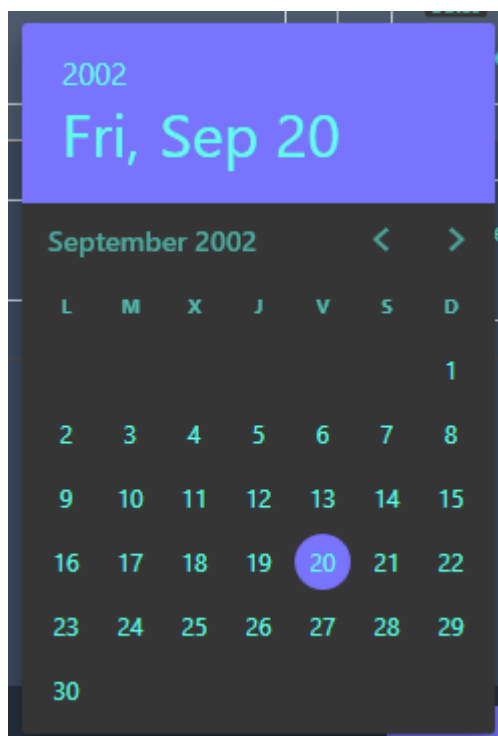
Para guardar los cambios deberás pulsar en *guardar cambios seleccionados*.



Si nos fijamos bien, el campo estado es una lista para poder elegir entre tres estados. Veamos el Frontend.

```
<DataGridTemplateColumn Header="Estado" Width="200">
  <DataGridTemplateColumn.CellTemplate>
    <DataTemplate>
      <ComboBox
        ItemsSource="{Binding DataContext.OpcionesEstado, RelativeSource={RelativeSource AncestorType=local:EditarSolicitudesAcceptadosUC}}"
        SelectedValue="{Binding Path=estado, UpdateSourceTrigger=PropertyChanged}"/>
    </DataTemplate>
  </DataGridTemplateColumn.CellTemplate>
</DataGridTemplateColumn>
```

La lista que se usa en el combobox es seleccionada desde el VM, no de la lista de objetos de tipo Solicitudes.

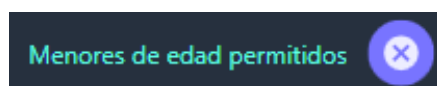
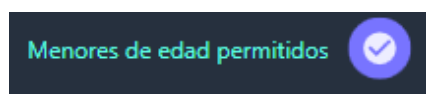


También se ha usado un datePicker para facilitar al gestor la fecha.

A la hora de editar o crear un torneo se ha facilitado la selección de nivel de dificultad con una rating bar.



También se ha usado un togglebutton para que el gestor elija si pueden concursar menores o no.

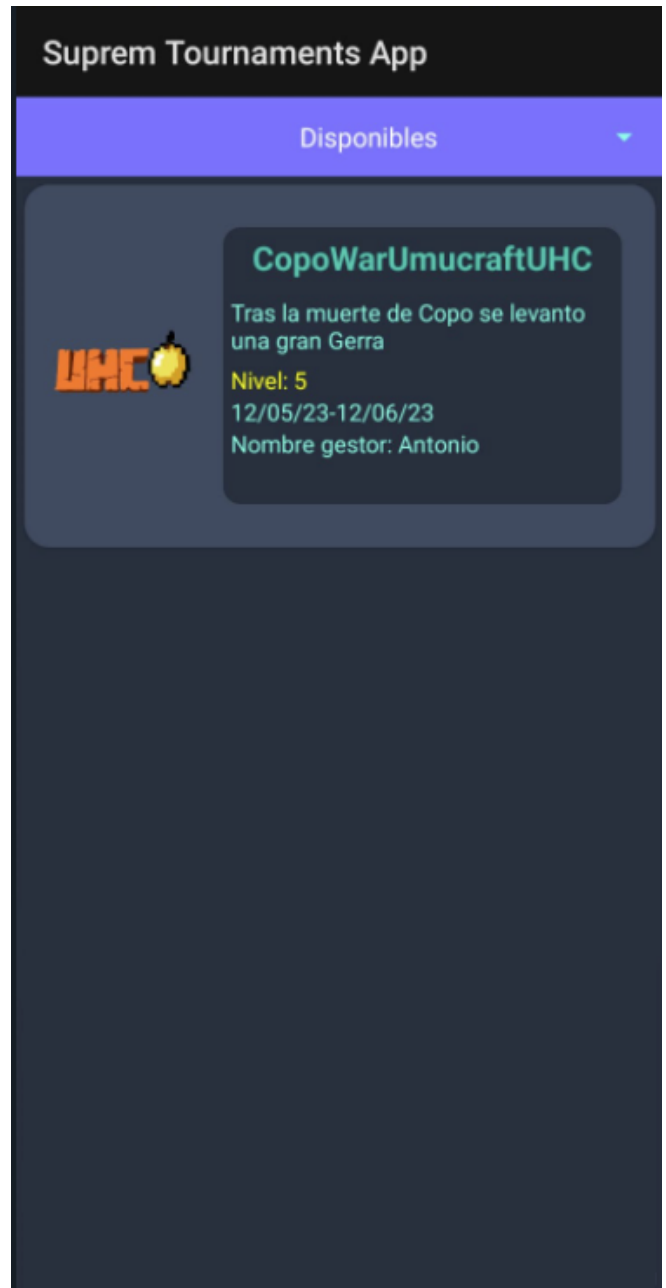


7. Manual de Usuario

7.1 Aplicación de Android

7.1.1 Lista de Torneos

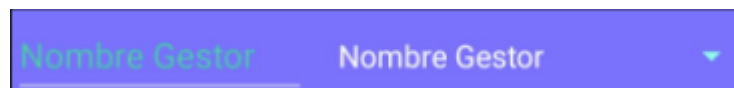
Al abrir la aplicación podemos observar una lista de Torneos que por defecto son los torneos disponibles, si nos fijamos en esta ventana arriba a la derecha tenemos un spinner con diferentes opciones:



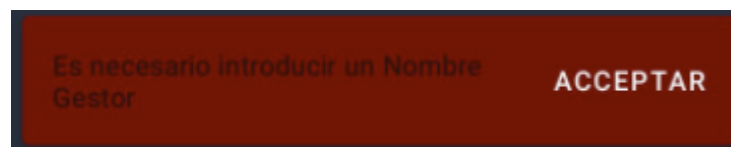
Disponibles: Torneos con una fecha de finalización superior a la actual y que aún permite participantes.

Disponibles	Fecha de finalización ASC: nos muestra todos los torneos de forma ascendente por fecha de finalización.
Fecha Finalización Asc	Fecha de finalización DES: nos muestra todos los torneos de forma descendente por fecha de finalización.
Fecha Finalización Desc	
Menores y Adultos	Menores y adultos: nos muestra todos los torneos en los que pueden participar menores y adultos.
Adultos	
Adultos	Adultos: nos muestra los torneos en los que solo pueden participar mayores de edad.
Nivel Asc	
Nivel Desc	Nivel ASC: nos muestra todos los torneos de forma ascendente por nivel.
Nombre Gestor	Nivel DES: nos muestra todos los torneos de forma descendente por nivel.
Nivel	

Nombre Gestor: habilita una barra de búsquedas para buscar por nombre de Gestor.

A blue search bar with the placeholder text "Nombre Gestor" in green on the left and "Nombre Gestor" in white on the right, followed by a small blue downward arrow icon.

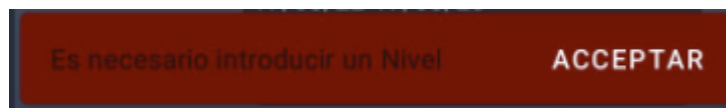
Si no indicamos ningún gestor, nos saldrá un snackbar indicado que es obligatorio.

A dark red rectangular snackbar message with the text "Es necesario introducir un Nombre Gestor" in white on the left and a white "ACEPTAR" button on the right.

Nivel: nos habilitará la barra de búsqueda para buscar por nivel de dificultad.

A blue search bar with the placeholder text "Nivel" in green on the left and "Nivel" in white on the right, followed by a small blue downward arrow icon.

Al igual que el anterior, nos mostrará un error, si no hemos introducido nada



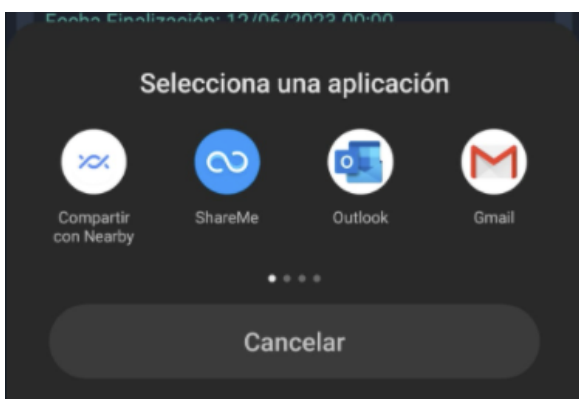
Tanto con la opción de nombre gestor como la del nivel, pulsaremos en el enter para que busque resultados.

Si queremos actualizar la lista de torneos, solo debemos hacer un gesto de arriba a abajo para que se actualice la lista (esto actualizará los datos según la opción elegida).

7.1.2 Datos del Torneo seleccionado

Por último tenemos la opción de ver los datos básicos del torneo, esto será posible pulsando sobre el torneo. El resultado será un cambio de vista donde se mostrará la información básica del torneo.

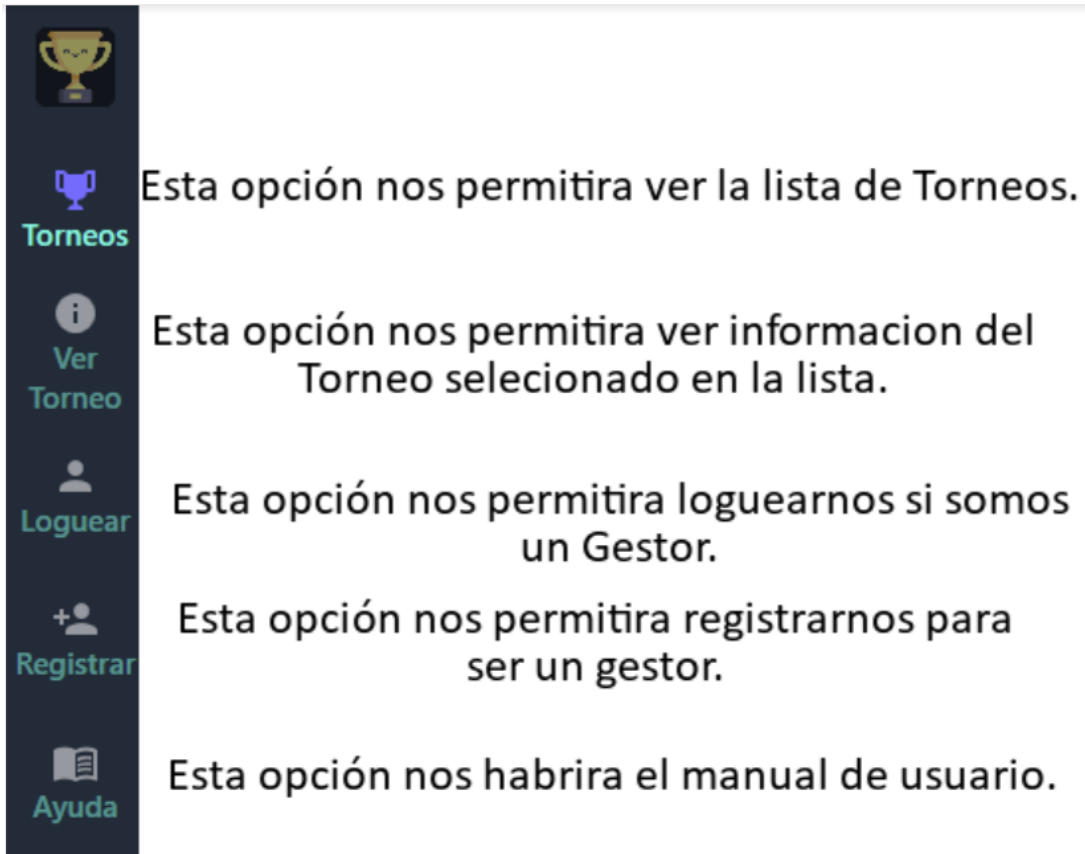
Si pulsamos sobre el botón de la carta nos dejara elegir una aplicación para mandar un email al gestor del torneo, por si tiene alguna duda.



7.2 Programa de Windows

7.2.1 Ventana Principal (sin login)

Veamos el menú de la izquierda del programa:



Si eres un gestor, para acceder a las funcionalidades disponibles es necesario loguearse.

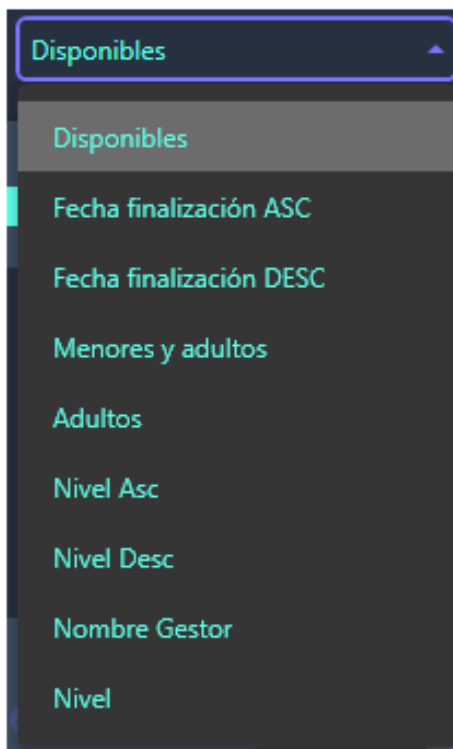
7.2.2 Torneos

En esta ventana podemos ver todos los torneos.



También podemos filtrar con el combobox de la derecha superior. Tenemos un total de 9 opciones :

Disponibles: Torneos con una fecha de finalización superior a la actual y que aún permite participantes.



Fecha de finalización ASC: nos muestra todos los torneos de forma ascendente por fecha de finalización.

Fecha de finalización DESC: nos muestra todos los torneos de forma descendente por fecha de finalización.

Menores y adultos: nos muestra todos los solicitantes en los que pueden participar menores y adultos.

Adultos: nos muestra los torneos en los que solo pueden participar mayores de edad.

Nivel ASC: nos muestra todos los torneos de forma ascendente por nivel.

Nivel DES: nos muestra todos los torneos de forma descendente por nivel.

Gestor: nos habilita una barra de búsquedas a la izquierda superior para buscar por nombre de Gestor.

Nivel: nos habilitará la barra de búsqueda para buscar por nivel de dificultad.

El funcionamiento de la barra de búsqueda es simple. Deberemos introducir el nivel o el nombre del gestor para buscar esos torneos y después pulsar en la lupa.



Si queremos ver toda la información del torneo, deberemos pulsar sobre un torneo e ir al menú lateral izquierdo a la opción Ver Torneo.

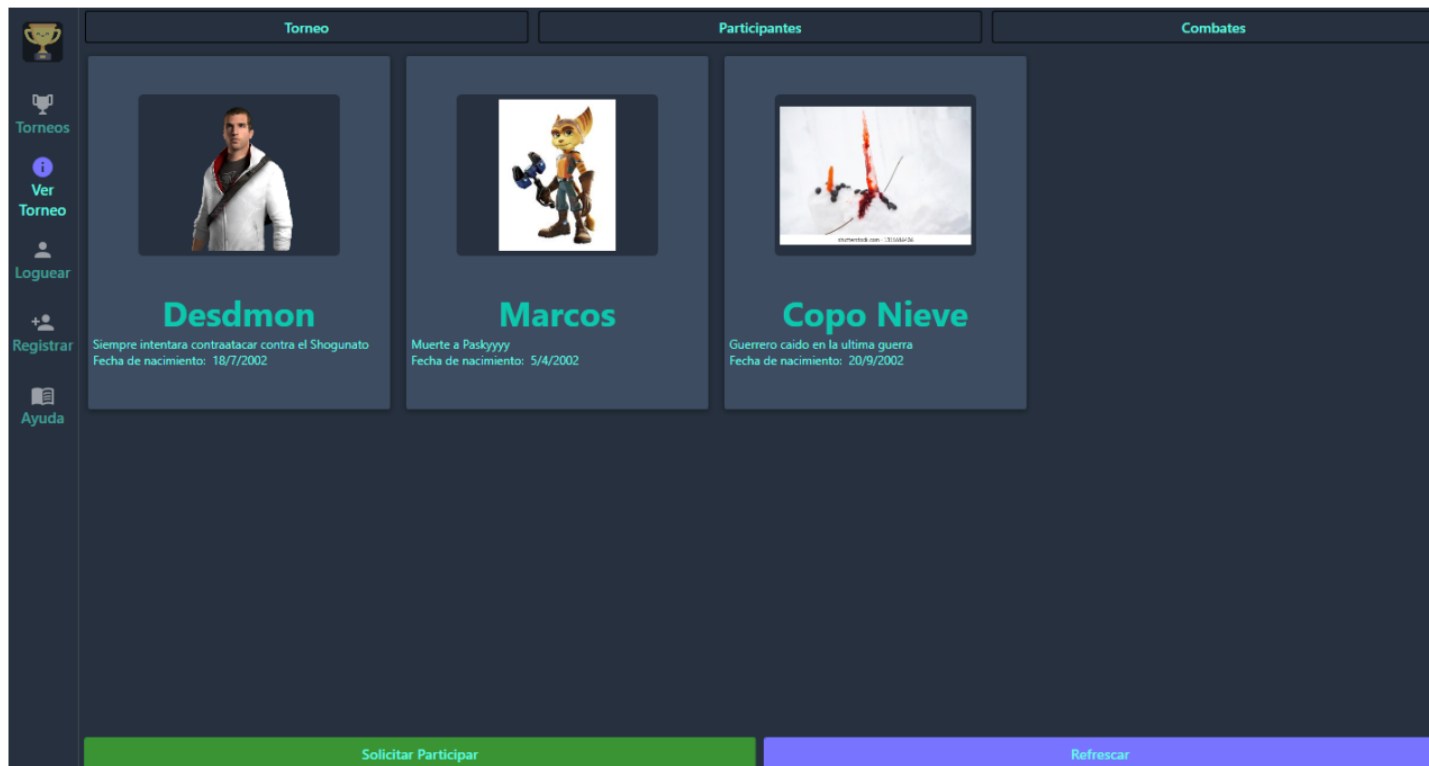
7.2.3 Ver Torneo

En esta ventana podremos ver todos los datos del torneo seleccionado en Torneos.



Lo primero que veremos por defecto será la información del torneo.

Pero si observamos, podemos ver más botones como el de Participantes



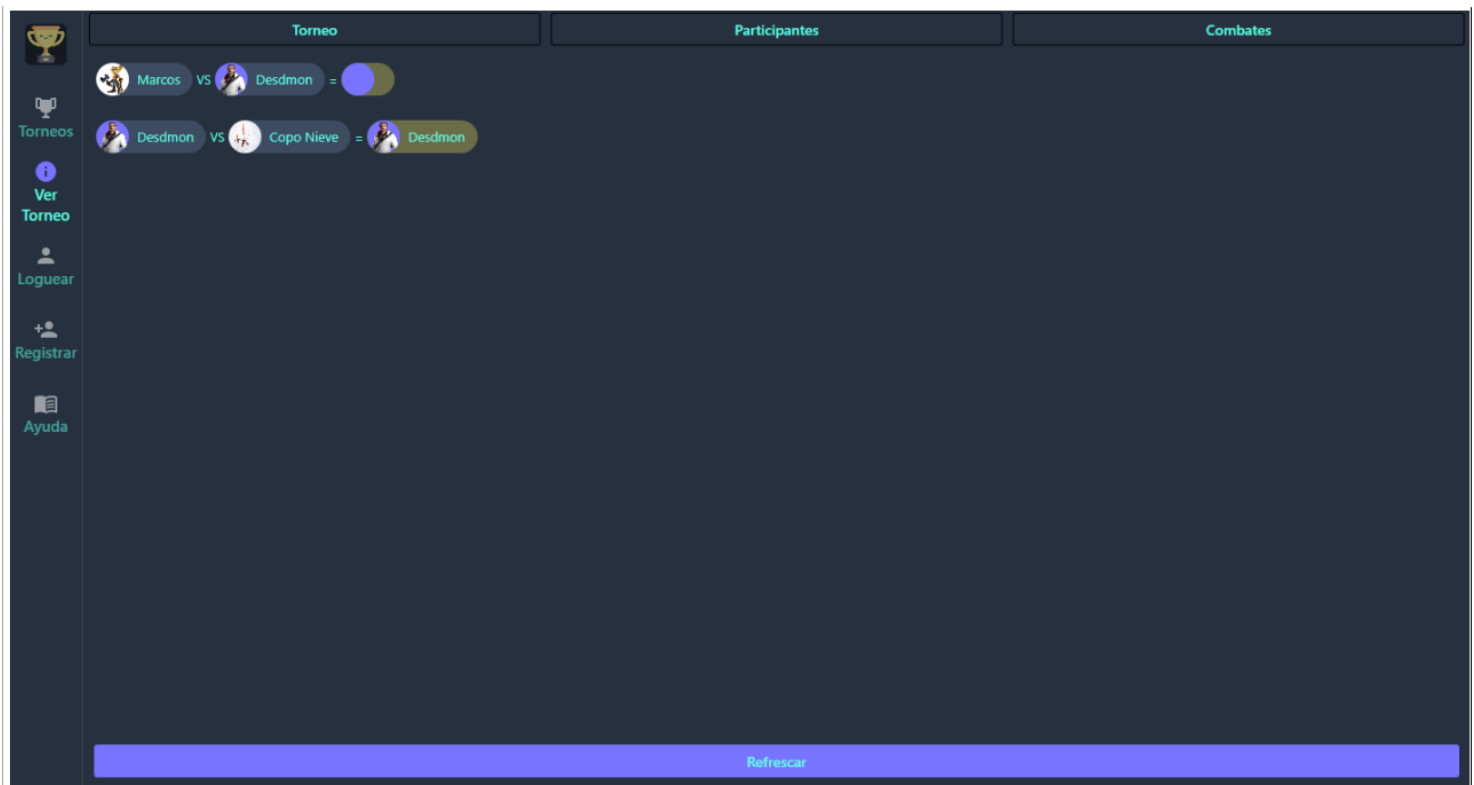
En la ventana participantes, podemos ver los participantes del torneo. Podemos solicitar participar en un torneo (solo si el Torneo no ha finalizado). Nos saldrá un diálogo para rellenar nuestros datos. Si el torneo no acepta menores de edad y solicita concursar, no será posible hacer la solicitud y nos saldrá un error indicando dicho error.

The form is for registering as a participant. It features a large blue profile picture placeholder on the left. To the right are four input fields: 'Nombre' (0 / 69), 'Datos' (0 / 200), 'Datos publicos' (0 / 128), and a date field set to '10/05/2022'. At the bottom are two buttons: 'Enviar Solicitud' (green) and 'Cancelar' (red).

Nombre	0 / 69
Datos	0 / 200
Datos publicos	0 / 128
10/05/2022	

También podremos refrescar los datos de los participantes. Si se da el caso de que el Gestor aceptó nuestra solicitud, poder verlo.

La última ventana de datos del torneo es la de combates, donde veremos los combates con ganadores y los que aún no ha ganado nadie .

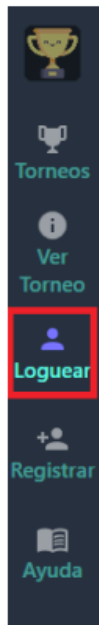


7.2.4 Registrarse

The screenshot shows the registration form with the following fields: 'Nombre' (Name), 'Email', 'Contraseña' (Password), and 'Repita la contraseña' (Repeat the password). A green 'Registrarse' (Register) button is located below the password fields. The left sidebar contains icons for 'Torneos', 'Ver Torneo', 'Loguear', 'Registrar', and 'Ayuda'.

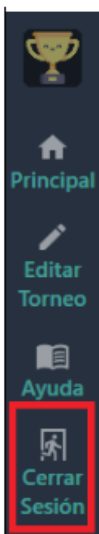
Para registrarse y tener los privilegios de un gestor solo debemos de ir al menú lateral izquierdo, pulsar sobre la opción registrarse y rellenar el formulario con un email válido y contraseña. Esta debe contener al menos 2 mayúsculas, 1 minúscula, 2 dígitos y una longitud mínima de 10.

7.2.5 Iniciar Sesión



Para iniciar sesión deberemos pulsar en el menú lateral sobre el botón Loguear

Después deberemos poner nuestro email y contraseña

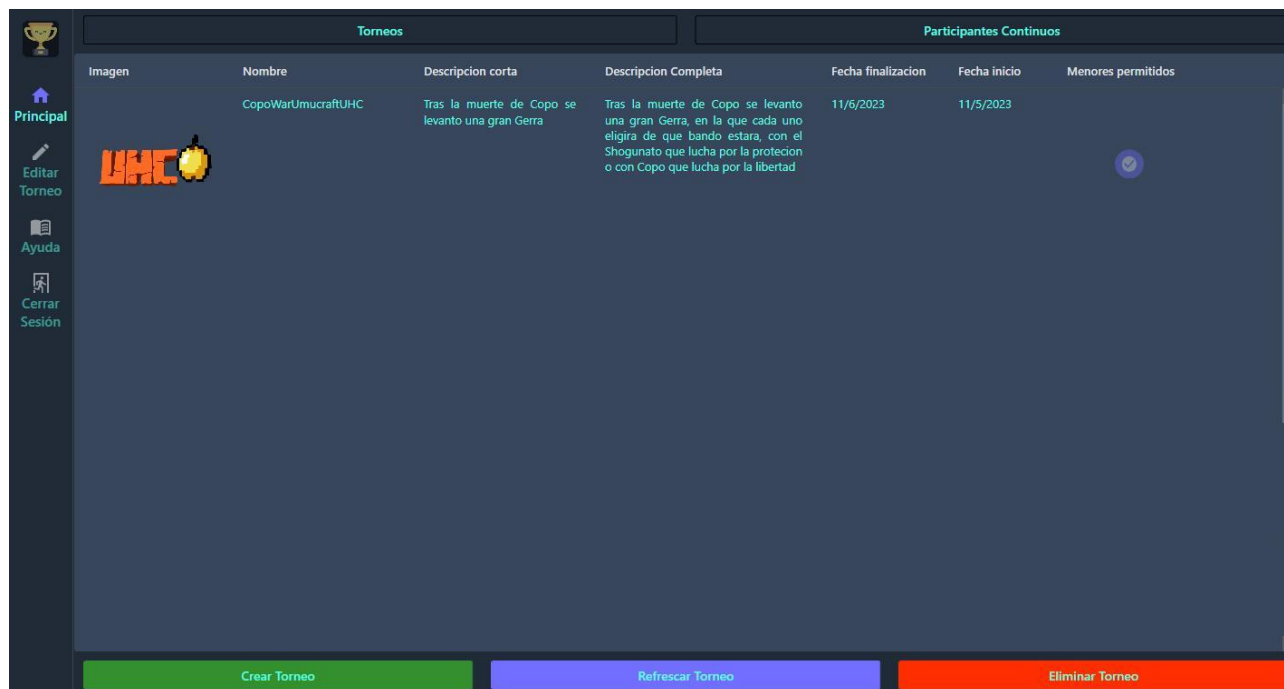
El formulario de inicio de sesión tiene un fondo oscuro. Incluye un campo de texto para 'Email', un campo de texto para 'Contraseña' con un contador de caracteres '0 / 124' a la derecha, y un botón rectangular de color verde con el texto 'Iniciar Sesión' en blanco.



Tras loguearnos el menú lateral cambiará teniendo unas opciones distintas.

Para cerrar sesión, solo le tendremos que dar en la opción del menú lateral izquierdo en cerrar sesión.

7.2.6 Ventana principal del Gestor, Lista torneos

Tras loguearnos, el menú lateral cambiará teniendo unas opciones distintas, deberemos ir a la pantalla principal donde veremos una lista de todos los torneos (los cuales podremos ordenar de forma ascendente o descendente según el dato que pulsemos en la cabecera) que tiene el gestor. En esta misma ventana, podemos observar abajo las opciones de eliminar, crear torneo y refrescar (para ver el nuevo torneo que hemos creado).

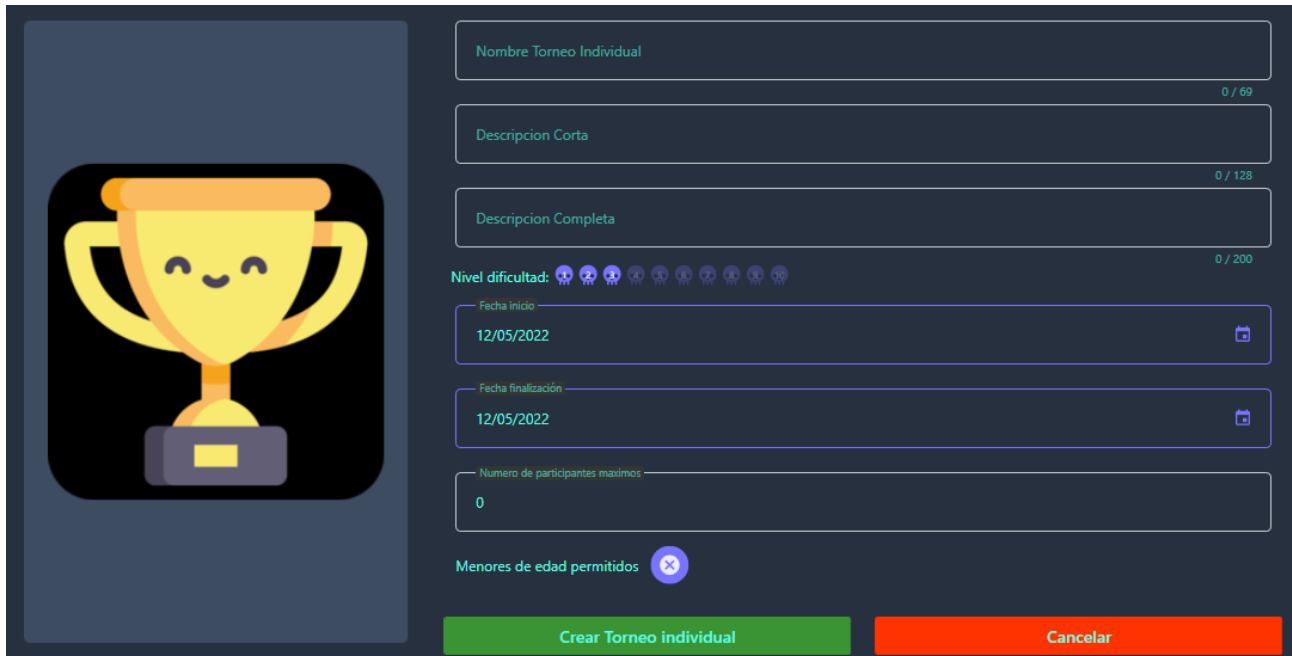


Torneos				Participantes Continuos		
Imagen	Nombre	Descripción corta	Descripción Completa	Fecha finalización	Fecha inicio	Menores permitidos
	CopoWarUmucraftUHC	Tras la muerte de Copo se levanto una gran Gerra	Tras la muerte de Copo se levanto una gran Gerra, en la que cada uno eligira de que bando estara, con el Shogunato que lucha por la proteccion o con Copo que lucha por la libertad	11/6/2023	11/5/2023	

Crear Torneo Refrescar Torneo Eliminar Torneo

Si queremos eliminar un torneo, solo deberemos seleccionar el torneo que queremos borrar y pulsar sobre el botón eliminar torneo.

Si pulsamos sobre crear torneo, nos saldrá un diálogo para rellenar con los datos básicos que necesitemos en el torneo a crear. Para elegir una imagen del torneo, deberemos clicar sobre la imagen por defecto.



Después de crearlo, volveremos a la ventana anterior y para ver el nuevo torneo deberemos pulsar en *refrescar*.

7.2.6 Ventana principal del Gestor Lista Participantes continuos.

Si volvemos a la pantalla principal del gestor, podemos observar que arriba, aparte de tener un botón para ver los torneos, también tenemos otro para ver los participantes continuos (estos participantes no están participando en ningún torneo, ya que los podremos usar en todos lo que queramos). En esta nueva ventana nos encontramos con una lista de participantes. Si pulsamos sobre una línea y en el campo que queramos cambiar, podemos cambiar la información de dicho participante continuo (incluyendo la imagen).

Si queremos guardar los cambios de la línea que acabamos de editar deberemos pulsar antes la línea y luego darle al botón de abajo para guardar cambios.

También se podría realizar usando la combinación de teclas control+s. Las otras opciones que tenemos en esta ventana es *eliminar Participante continuo*.

The screenshot shows a web application interface for managing continuous participants. On the left is a sidebar with icons for 'Principal', 'Editar Torneo', 'Ayuda', and 'Cerrar Sesión'. The main area is titled 'Participantes Continuos' and contains a table with the following columns: 'Imagen', 'Nombre', 'Fecha nacimiento', 'Datos', and 'Datos publicos'. There are three rows of participants: 'Copo Nieve' (born 20/09/2002, 'Un guerrero de nieve'), 'Vicente Rico' (born 05/02/1988, 'Vicente Rico profesor de AD'), and 'Marcos' (born 05/04/2002, 'Marcos conocido tambien como Rach'). Each row has character counts for the name, date, and public data fields. At the bottom of the window are four buttons: 'Crear Participante Continuo' (green), 'Guardar cambios seleccionado' (blue), 'Refrescar' (blue), and 'Eliminar' (red).

Imagen	Nombre	Fecha nacimiento	Datos	Datos publicos
	Copo Nieve 10 / 69	20/09/2002 20 / 200	Un guerrero de nieve 20 / 200	Guerrero caido en la ultima guerra 34 / 200
	Vicente Rico 12 / 69	05/02/1988 27 / 200	Vicente Rico profesor de AD 55 / 200	Vicente Rico profesor de AD creador de APIs Profesional
	Marcos 6 / 69	05/04/2002 65 / 200	Marcos conocido tambien como Rach	Muerte a Paskyyyy 18 / 200

Refrescar que servirá para refrescar los datos y así, de esta forma, comprobar que se ha guardado el Participante continuo que hemos creado o ver que se ha guardado la edición.

Crear Participante continuo nos abrirá un diálogo para crear un Participante continuo. Si pulsamos sobre la imagen, por defecto podremos elegir una nuestra.

The dialog box for creating a new continuous participant. It features a large placeholder image on the left. On the right, there are four input fields: 'Nombre' (0 / 69), 'Datos' (0 / 200), 'Datos publicos' (0 / 128), and 'Fecha de nacimiento' (10/05/2022). At the bottom are two buttons: 'Guardad solicitud continua' (green) and 'Cancelar' (red).

7.2.7 Ventana editar torneo

Si queremos editar un torneo deberíamos seleccionar el torneo a editar y pulsar en el menú lateral sobre la opción de editar torneo.

Al pulsar sobre esta opción, nos saldrá una ventana con la información básica del torneo, la cual podemos editar. Para guardar deberemos dar a guardar cambios o usar la combinación de teclas control+s.

Principal

Editar Torneo

Ayuda

Cerrar Sesión

Editar datos Torneo

Participantes

Solicitudes

Combates

Nombre Torneo Individual

CopoWarUmucraftUHC

Descripción Corta

Tras la muerte de Copo se levanto una gran Gerra

Descripción Completa

Tras la muerte de Copo se levanto una gran Gerra, en la que cada uno elige de que bando estara, con el Shogunato que lucha por la proteccion o con Copo c

Nivel dificultad: 178 / 200

Fecha Inicio

11/05/2023

Fecha Finalización

11/06/2023

Numero de participantes maximos




10

Menores de edad permitidos

Editar Torneo individual

7.2.8 Ventana Editar Participantes aceptados

En esta misma ventana podemos ver distintas opciones arriba. Si pulsamos sobre Participantes, la ventana cambiará para tener los participantes que concursan en el torneo. Podremos editar los datos de un participante, seleccionamos la línea y podremos cambiar los datos que necesitemos. Después deberemos pulsar en el botón de guardar cambios seleccionados o usar la combinación de teclas control+s.

	Editar datos Torneo	Participantes	Solicitudes	Combates		
	Imagen	Nombre	Fecha nacimiento	Datos	Datos publicos	Estado
Principal		Nombre Desdmon 7 / 69	18/07/2002	Datos Desdmon conocido l 68 / 200	Datos publicos Siempre intentara contraatacar contra 50 / 200	Acceptada
Editar Torneo		Nombre Marcos 6 / 69	05/04/2002	Datos Marcos conocido tar 65 / 200	Datos publicos Muerte a Paskyyyy 18 / 200	Acceptada
Ayuda		Nombre Copo Nieve 10 / 69	20/09/2002	Datos Un guerrero de nieve 20 / 200	Datos publicos Guerrero caido en la ultima guerra 34 / 200	Acceptada
Cerrar Sesión						

Crear Participante

Guardar cambios seleccionado


Refrescar

Añadir participante continuo

Eliminar

Guardar seleccionada como participante continuo

Podemos crear un Participante con el botón crear participante, el cual nos abrirá un diálogo para rellenar con los datos del participante. Podemos cambiar la imagen por defecto haciendo un click sobre esta y pulsaremos en crear si queremos crear dicho participante (si en participante no es compatible por ser menor de edad nos lo indicará un diálogo y nos impediría crearlo).



Nombre

0 / 69

Datos

0 / 200

Datos publicos

0 / 128

Fecha de nacimiento




11/05/2022

Crear Solicitud

Cancelar

Otra opción es guardar como Participante continuo, lo cual nos permite guardar el seleccionado como un participante continuo para poder usarlo en otros torneos.

Añadir participante continuo, nos mostrará un diálogo con la lista de todos los Participantes continuos que existen. Pulsamos sobre el que queramos y le daremos al botón añadir seleccionado (si el solicitante no es compatible con el torneo por ser menor de edad, saldrá un diálogo y nos impediría añadirlo), tras ello volvemos a la lista de solicitantes, le damos a refrescar y vemos que se ha añadido correctamente.

Imagen	Nombre	Fecha nacimiento	Datos	Datos publicos
	Copo Nieve	20/09/2002	Un guerrero de nieve	Guerrero caído en la última guerra
	Vicente Rico	05/02/1988	Vicente Rico profesor de AD	Vicente Rico profesor de AD creador de APis Profesional
	Marcos	05/04/2002	Marcos conocido tambien como Rached y clan	Muerte a Paskyyyy

Elegir este participante continuo

Cancelar

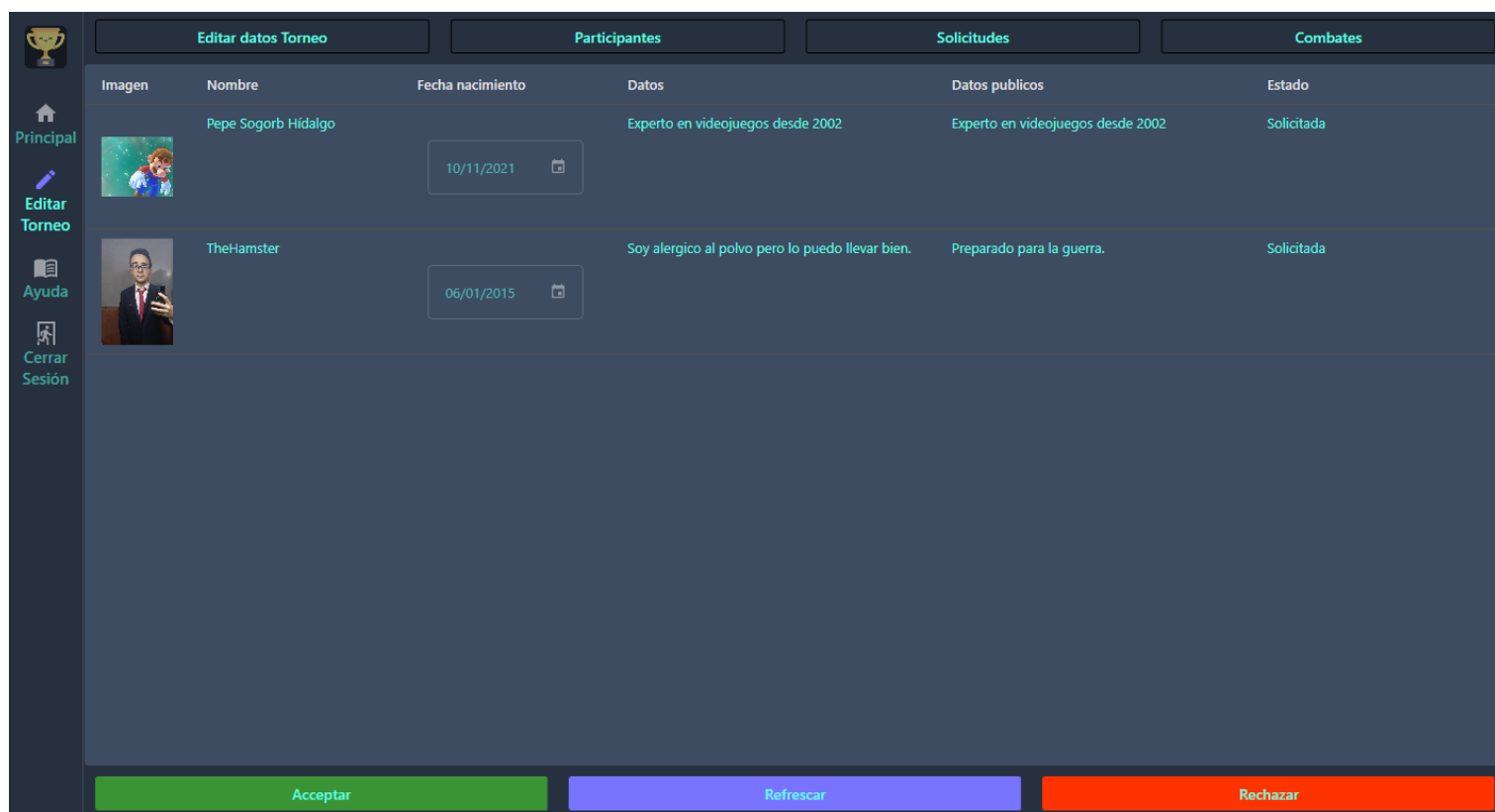
Eliminar: solo se eliminará un Participante si no está en ningún combate.

Refrescar: como ya fue recalcado en los apartados anteriores refrescará la lista de participantes.

Guardar seleccionado como participante continuo: si pulsamos este botón, se guardará el participante seleccionado como participante continuo de esta forma podremos usarlo en otros torneos.

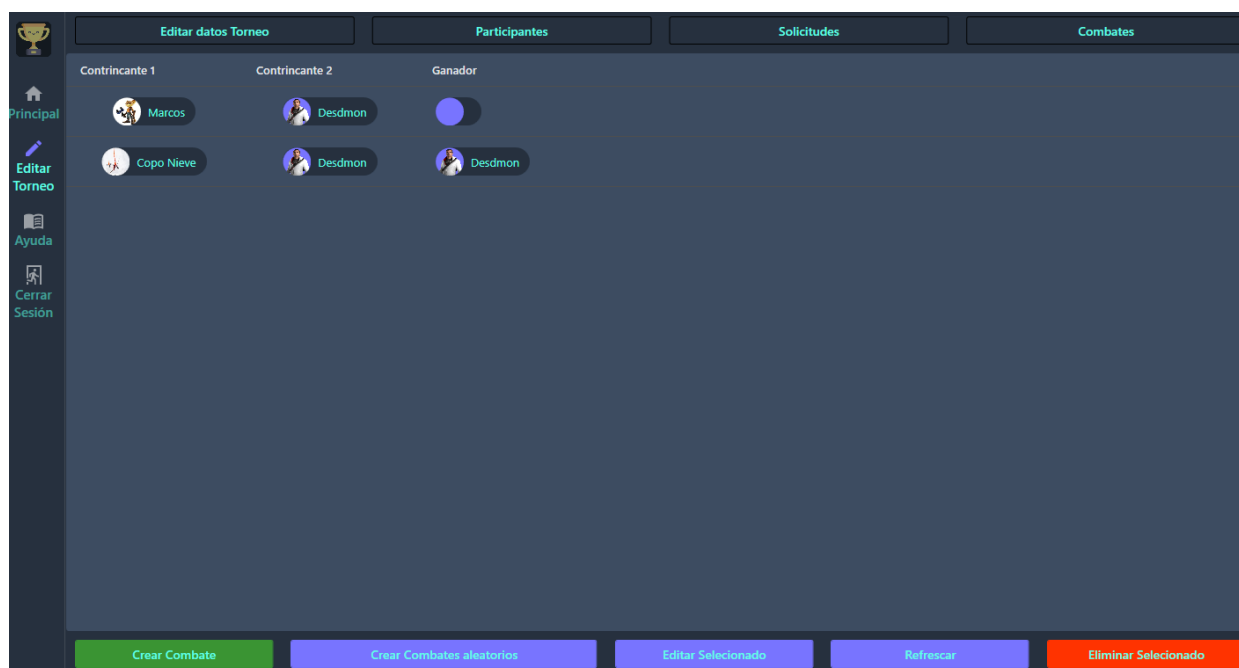
7.2.9 Ventana Participantes Solicitados

Veamos otra opción; en el menú superior podemos observar el botón de solicitudes. Si pulsamos sobre este botón, nos saldrá la lista de los participantes que ha solicitado concursar, seleccionamos el participante que queramos y en la parte inferior de la ventana podemos rechazarlo o aceptarlo (si queremos hacer algún cambio en sus datos o si nos hemos equivocado y no queríamos aceptarlo, podemos editar su estado en la sección Participantes). También contamos con un botón para refrescar los datos por si alguien acaba de solicitar concursar.



7.2.10 Ventana Combates

La última opción del menú de arriba es combates. En esta, podemos ver varias opciones como crear combate aleatorio, crear combates aleatorios sin ganador con todos los participantes aceptados y eliminar combate.



Crear combate nos mostrará un diálogo con 3 listas de solicitantes donde elegirás el contrincante 1, el contrincante 2 y el ganador.

Contrincante 1	Contrincante 2	Ganador
Desdmon	Desdmon	Desdmon
Marcos	Marcos	Marcos
Copo Nieve	Copo Nieve	Copo Nieve

[Crear combate](#) [Cancelar](#)

Editar seleccionado, nos modificará la vista y podremos editar el combate seleccionado.

Contrincante 1	Contrincante 2	Ganador
Desdmon	Desdmon	Desdmon
Marcos	Marcos	Marcos
Copo Nieve	Copo Nieve	Copo Nieve

[Guardar cambio](#) [Cancelar](#)

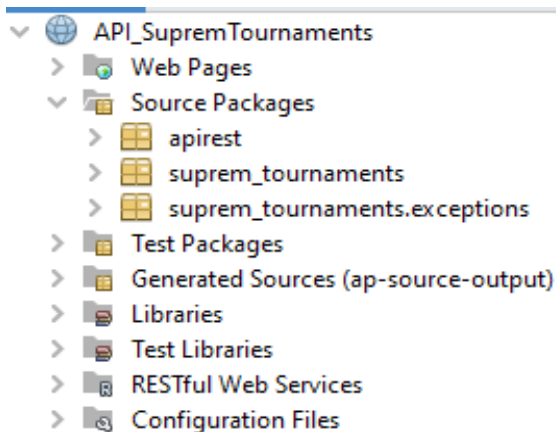
8. Requisitos e instalación

Es necesario usar el servidor de Azure para poder acceder a la API.

La aplicación de Android es posible instalarla desde la apk que ha sido generada con Android Studio.

El programa de Windows se puede instalar con el msi generado, con la ayuda de la extensión Microsoft Visual Studio Instalar.

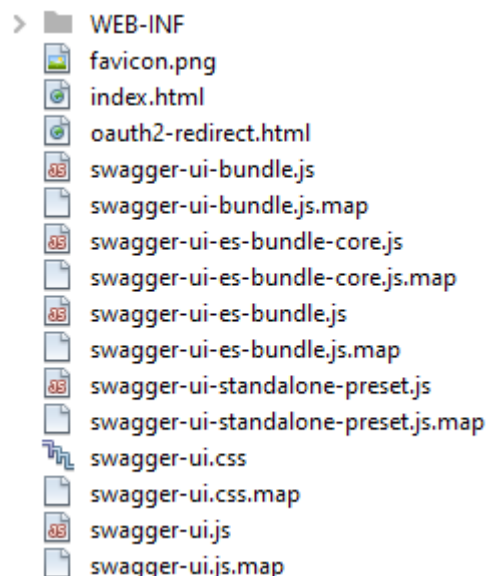
8.1 API

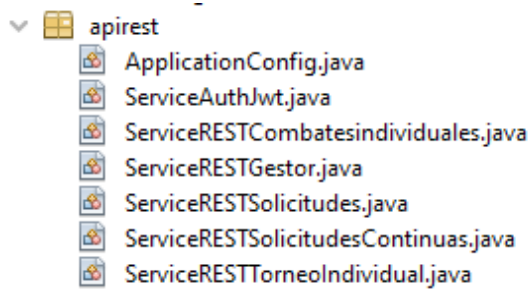


Podemos observar todos los ficheros de la Api.

Web Pages.

En esta carpeta están todos los ficheros que necesita Swagger para hacer su vista.

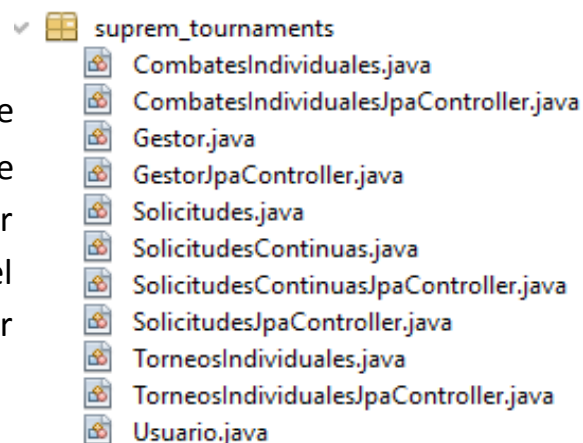




Paquete apiREST

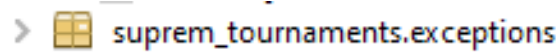
En este paquete están todos los servicios para poder acceder desde peticiones.

Paquete suprem_tournaments En este paquete están todas las clases de la base de datos y los controlers necesarios para poder modificar la base de datos. También está el objeto Usuario que fue utilizado para iniciar sesión.

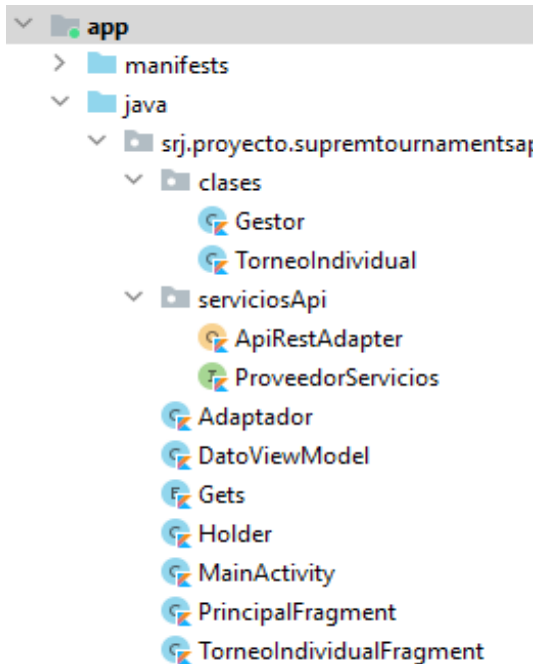


Paquete suprem_tournaments.exceptions

Este paquete fue generado automáticamente



8.2 Aplicación de Android



La estructura de la aplicación Android es la siguiente:

En la foto de la izquierda vemos todo lo necesario del Backend

Clases

Gestor: Objeto que almacena los datos de un gestor (usado por la API).

Torneos Individuales: Datos que forman un torneo individual como nombre, descripción corta...(usado por la API).

serviciosApi

ApiRestAdapter: clase para comunicarse con la API, la cual implementa la interfaz ProveedorServicios.

ProveedorServicios: interfaz para hacer Gets a la API.

Adaptador: clase que extiende de RecyclerView.Adapter<Holder>. Esta clase nos creará una lista para poder visualizar los objetos tipo torneo.

DatoViewModel: Clase para pasar un objeto tipo TorneoIndividual entre fragments.

Gets: Enum con todos los posibles Get a hacer.

Holder: Clase la cual pone los datos del torneo en un layout.

MainActivity: Clase main activity la cual tendrá los fragmentos.

PrincipalFragment: Backend para una lista de Torneos Individuales.

TorneoIndividualFragment: Backend para los datos del torneo seleccionado.

Drawable

En esta carpeta podemos observar distintos archivos necesarios como el gif o un borde redondo.

Layout

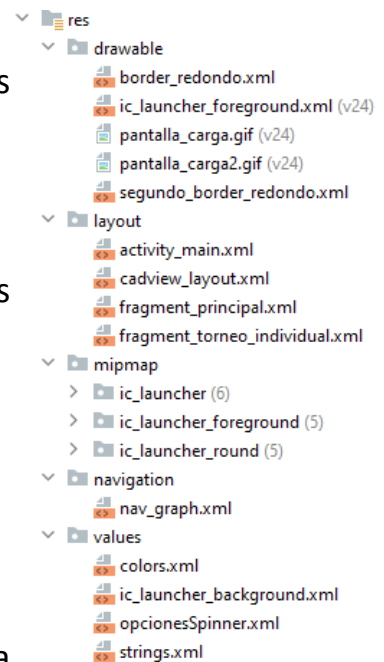
Todo el frontend de la aplicación incluyendo el de los fragments y el de cada elemento de la lista de torneos.

Mipmap

Fue modificado el icono por defecto para poner uno propio.

Values

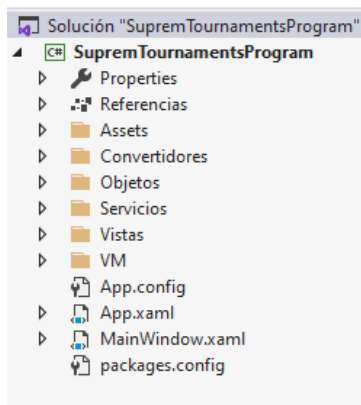
Distintos recursos como colors para tener la paleta de la aplicación, o las opciones del Spinner.



8.3 Programa de Windows

La estructura base del proyecto de WPF está dividida en varias carpetas.

Assets: imágenes usadas.



Convertidores: objetos para la comunicación específica entre variables del backend con el frontend.

Servicios: clases estáticas que se usarán en todas las clases como el servicio de navegación o el de la API.

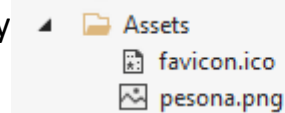
Objetos: objetos usados en el programa, como TorneoIndividual.

Vistas: ficheros xaml(windows y usercontrol).

VM: ficheros backend de cada vista.

Assets

Solo tiene dos imágenes, el icono del programa y el icono de una persona.

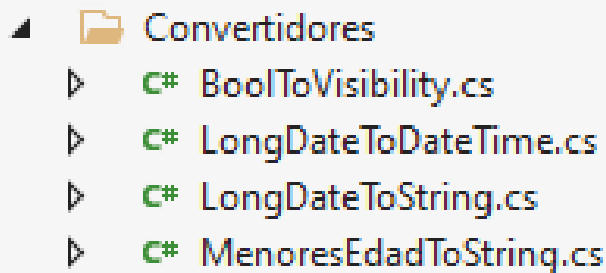


Convertidores:

BoolToVisibility: Usado para ocultar partes del frontend con un booleano.

LongDateToDateTime: para poder cambiar propiedades long usadas como date.

MenoresEdadToString: El cual nos muestra un mensaje según si es true o false.



Objetos

CheckJWK: Será usado para un objeto que devuelve que nos facilitara si el token JWT sigue siendo válido.

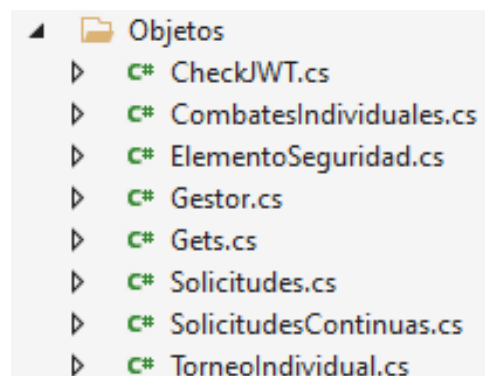
CombatesIndividuales: En este objeto se almacenará qué solicitud combatirá contra la otra solicitud y la solicitud ganadora (usado por la API).

ElementoSeguridad: Objeto que como su nombre indica es para la seguridad, ya que este almacena el token renombrado como ApiKey para verificar que puede hacer determinadas acciones en la API, la Cookie para saber que es el mismo usuario al hacer peticiones a la API y Sesion por una posible necesidad futura.

Gestor: Objeto que almacena los datos de un gestor (usado por la API).

Gets: Objeto para poder actualizar la lista de torneos.

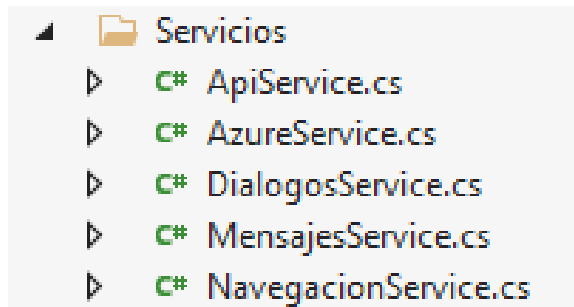
Solicitudes: Datos del solicitante como fecha de nacimiento, nombre, idTorneo... (usado por la API).



Solicitudes Continuas: Datos de una solicitud continua como fecha nacimiento,nombre...(usado por la API).

Torneos Individuales: Datos que forman un torneo individual como nombre, descripción corta...(usado por la API).

Servicios



ApiService: clase estática con métodos para poder hacer peticiones a la API (GET,POST,PUT,DELETE).

AzureService: Clase estática para poder guardar en un contenedor blob las fotos elegidas por el usuario.

DialogosService: Clase estática para informar al usuario de errores, información, o seleccionar una imagen.

MensajesServices: Fichero con clases de mensajería para intercambiar objetos entre vistas.

NavegacionService: Clase estática para crear objetos UserControl y Window y así poder cambiar de ventana fácilmente.

Vistas

Dentro del directorio Vistas hay otros dos directorios:

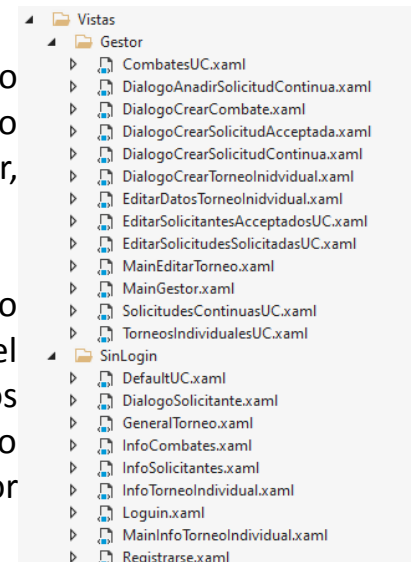


Gestor: tiene todas las vistas necesarias para un gestor.

SinLogin: tiene todas las vistas necesarias para una persona sin loguearse .

En esta imagen podemos observar todas las vistas que han sido necesarias para el programa de Windows. Vemos un claro patrón en que hay diálogos para crear objetos y para editar, visualizar.

También podemos observar algunas vistas usadas como “marcos”. Uno como este podría ser MainGestor que en el funcionamiento interno puede navegar entre distintos UserControl, otro podría ser MainEditarTorneo o MainInfoTorneoIndividual es decir todos los que empiezan por Main.



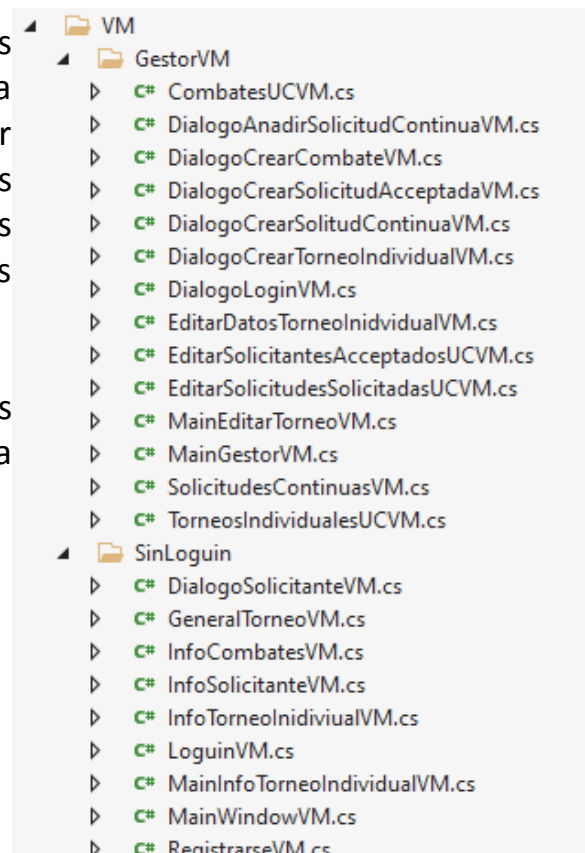
VM (acrónimo de Vista modelo)



Como hemos visto antes, esta también ha sido dividida en dos carpetas para una mayor comodidad.

Estos Vm son todo el backend de las vistas, es decir, estos serán los que tendrán toda la lógica del programa. Se encarga de utilizar apropiadamente el ApiService según las indicaciones del usuario gracias a los RelayCommand, que están bindeados a los botones del Frontend.

Estos estarán usando constantemente los objetos del fichero MensajeriaService para pasar objetos entre ellos.



9 Conclusiones

9.1 Conclusiones sobre el trabajo realizado

Lo primero que hice fue un boceto en Balsamic para aclarar las ideas y tener un prototipo como referencia. Después de ser rechazada mi idea tuve que quitar parte de la base de datos y añadir más funciones como Swagger o una aplicación para Android.

Luego creé la base de datos y la API con Swagger, en el punto de la API lo más complicado ha sido la seguridad, ya que ha conllevado muchos cambios internamente y bastante tiempo.

Tras acabar la API, fue creada la aplicación de Android para ver los datos básicos de los torneos que fue avanzando lentamente, añadiendo distintas opciones que no conocía, para que sea más amigable la interfaz con el usuario.

En la aplicación móvil también se tuvo la idea de que en la barra de búsqueda se auto completará el nombre de gestor, pero debido a la falta de tiempo no se pudo crear dicha función.

Finalmente, creé el programa de Windows donde empecé con un programa para comprobar que las comunicaciones con la API funcionaban correctamente. Después de una conexión con la API exitosa, empecé con el programa, el cual conllevó ver desde una perspectiva distinta los bocetos hechos anteriormente ya que lo que más me convenía era tener un menú lateral y este no estaba pensado. Al re diseñar mi programa con el menú hubo un error que me tuvo más de dos días intentarlo solucionar, pero finalmente se logró.

9.2 Posibles ampliaciones y mejoras.

Hay una idea que no he podido implementar por falta de tiempo, pero la API está preparada para poder mandarte los datos necesarios para esa funcionalidad. Se trata de la posibilidad de clicar sobre un solicitante y ver un gráfico de los combates ganados, a quien ha ganado y otro gráfico con los combates que ha perdido y contra quien a perdido. El resultado sería basado en este (aunque usaría gráficos de material design):

The mockup shows a user interface for a participant's combat statistics. At the top, there is a header 'Nombre Participante'. Below it, there are labels for 'Nombre', 'Apellidos', 'Más datos (publicos)', and 'Fecha de Nacimiento'. To the left of these labels is a small icon of a person. Below the labels, there are two main sections: 'Combates Perdidos contra:' and 'Combates ganados contra:'. Each section contains a list of participants with checkboxes. In the 'Combates Perdidos contra:' section, 'Participante 1' and 'Participante 5' are selected. In the 'Combates ganados contra:' section, 'Participante 2', 'Participante 3', 'Participante 4', and 'Participante 6' are selected. Below each list, there is a label 'Combates Perdidos en total = Numero' and 'Combates ganados en total = Numero' respectively. At the bottom center, there is a button labeled 'Volver'.

También considero que a mi programa de escritorio a veces le falta devolver feedback al usuario y esa sería una mejora obligatoria si llegase a comercializarse este software.

La idea principal también tenía torneos grupales, pero debido a la falta de funcionalidades base, esta fue removida (gracias a esto, se pudo hacer Swagger y la aplicación Android). Estaría bien poder añadirlo en un futuro.

Por desgracia, hay una idea que no me dio tiempo a desarrollar lo suficiente. Se trata de un diálogo que tras una hora desde que se logueó el gestor se mostrará para que tenga que volver a iniciar sesión (no fue implementada debido a la falta de tiempo para testear y pulir esta función).

10. Bibliografía

<https://es.stackoverflow.com/>

<https://developer.android.com/>

<https://docs.microsoft.com/es-es/dotnet/csharp/>

<https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit>

<https://material.io/design>

<https://color.adobe.com/es/create/color-wheel>

<https://gifmaker.me/>

<https://www.flaticon.es/>

11. Apéndice

Este proyecto es el que he querido hacer desde un principio. Si que es verdad que ha habido cambios frustrantes pero finalmente se consiguió acabar exitosamente.

Esto no significa que fue tarea fácil ya que conllevó estar desde Navidad desarrollando la idea y cuando acabe los exámenes tener como norma que cada día hasta la entrega había que adelantar el proyecto. Lo primero que tuve que hacer fue tirar gran parte de mi idea inicial a la basura y con la restante añadir muchas más funciones. Antes de tener las mañanas ocupadas cumplía con mis propias metas de avance con el proyecto pero cuando empecé las prácticas nunca logre cumplir con dichas metas, ya que acababa cansado y adelantaba la mitad de lo previsto. Esto conllevó que los fines de semana hubiera un estrés bastante alto debido a no cumplir mis propias metas.

Quitando el estrés, me gustó poder desarrollar mi propia idea desde cero, aunque si alguna vez vuelvo a hacer algo así de exigente, creo que sería conveniente tener un equipo, aunque sea pequeño, para poder apoyarnos entre nosotros y que cada uno se pudiera centrar en una cosa distinta.