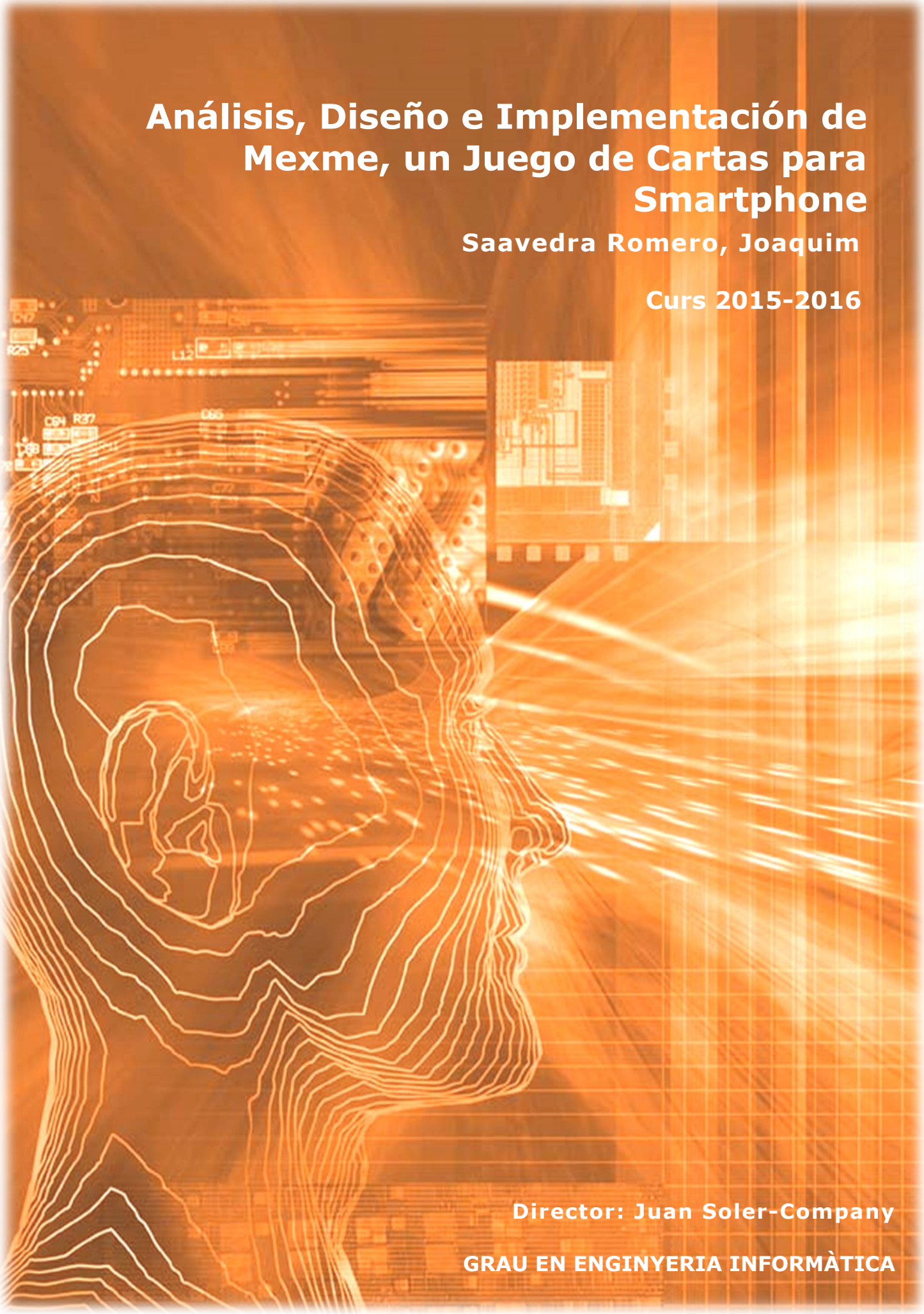
Análisis, Diseño e Implementación de Mexme, un Juego de Cartas para Smartphone

Saavedra Romero, Joaquim

Curs 2015-2016



Director: Juan Soler-Company

GRAU EN ENGINYERIA INFORMÀTICA

Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

Análisis, Diseño e Implementación de Mexme, un Juego de Cartas para Smartphone

Joaquim Saavedra Romero

TREBALL FI DE GRAU

GRAU EN ENGINYERIA INFORMÀTICA

ESCOLA SUPERIOR POLITÈCNICA UPF

CURS 2015 - 2016

DIRECTOR DEL TREBALL

Juan Soler-Company

Para ver esta película, debe disponer de QuickTime™ y de un descompresor .

Este trabajo solo es un peldaño más en esta ardua etapa de mi vida, pero representa mucho más, es el último peldaño antes de acreditarme como graduado y el que valida que por fin soy ingeniero.

En primer lugar, me gustaría dedicar este trabajo a mi familia, por ser mi sustento, apoyo y por la insistencia en los momentos complicados, en especial aquellos en los que estuve a punto de tirar la toalla.

A mis amigos por los buenos consejos cuando he dudado, por hacerme más llevaderos estos últimos cuatro años, por respetar mi espacio y ausencia cuando no pude estar ahí.

Pero sobretodo, a aquellos profesores que en su día dudaron de mí y nunca creyeron en mis cualidades. Este trabajo, entre muchos otros, es para demostrarles a ellos lo muy equivocados que estaban.

Agradecimientos

Quiero agradecer, en primer lugar, a Miguel Ballesteros por contactarme con mi tutor y despejar aquellas dudas que tenía cuando aún nadie me llevaba el trabajo final de grado.

También a mi actual tutor, Juan Soler, por tenerme en consideración cuando creía que era demasiado tarde para sacar el proyecto adelante, por su compromiso y directrices para mejorarlo.

A mis compañeros por la empatía y soporte.

Y, por último, mi familia, aunque no han participado directamente en éste trabajo, lo han cubierto en otros aspectos poniéndomelo todo un poco más fácil.

A todos ellos, Gracias.

Resumen

En este trabajo estudiaremos algunos aspectos para llevar a cabo la ejecución de una idea hasta dar con un producto final, una aplicación para smartphones. Por esta razón, nos centraremos principalmente en las buenas prácticas de la ingeniería del software, desde la identificación de la necesidad hasta el diseño e implementación de ésta sin obviar la importancia de algunas de las tareas de un business plan como el marketing estratégico, estudio de mercado y la planificación para acabar de definir cómo haremos nuestra app.

El resultado al que pretendemos llegar es un juego de cartas cuya mecánica y lógica implementada sea sencilla, pero a su vez, resulte un rompecabezas entretenido para para el jugador. La aplicación será orientada principalmente a plataformas Android, sin descartar otras como iOS Apple y Windows.

Por ello, la tecnología que haremos uso para producir el paquete de software será PhoneGap/Cordova empleando APIs JavaScript, HTML5 y CSS3.

Abstract

In this project we study some aspects to carry out the execution of an idea to find a final product, an application for smartphones. For this reason, we will focus primarily on best practices of software engineering, from identifying the needs to design and implement them without forgetting the importance of some of the tasks of a business plan as strategic marketing, study market and planning to finish defining how we will make our app.

The result which we attempt to reach is a card game whose mechanics and implemented logic is simple, but it turns become an entertaining puzzle for the player. The application will be mainly oriented to Android platforms, without discarding others like Apple iOS and Windows.

Therefore, the technology that we use to produce the software package will be PhoneGap/Cordova APIs using JavaScript, HTML5 and CSS3.

Prólogo

Al empezar a realizar prácticas en una empresa consultora de las TIC, viví en mis propias carnes como se producía el software desde dentro. Realicé diferentes tareas, desde el testeo, validaciones, desarrollo hasta el análisis y despliegue y es entonces cuando entendí cómo funcionaba la cadena de producción, la manera en la que se iban escalando las tareas, la planificación de los hitos en la producción, en definitiva, el ciclo de vida de un producto.

Me fascinó la idea de convertir una necesidad del cliente en algo "tangible" como es una aplicación empleando los conocimientos de la ingeniería del software. Creía que los equipos de desarrollo se aferraban fuertemente a las metodologías de trabajo y patrones establecidos, pero lo cierto es que cada proyecto requería su propia metodología y es cuando entendí que nosotros adaptamos las metodologías no ellas a nosotros, pudiendo sacar lo mejor de cada una.

Desde la ignorancia, creía que eran centrales en las que exclusivamente solo se producía código, pero mi perspectiva cambio totalmente cuando vi que la documentación y la gestión de los proyectos y operaciones toma un papel importantísimo además de la ejecución correcta de la planificación.

Éste proyecto para mi significa mi iniciación no oficial como Project Manager para poner en práctica mis capacidades como analista y diseñador con el fin de saberlas transmitir a un equipo de developers.

Índice

Lista de figuras

Lista de tablas

1. INTRODUCCIÓN Y MOTIVACIONES

1.1. Smartphones en la actualidad

Desde inicios de la década de los 90, la comunicación móvil ha estado muy presente en nuestras vidas. Los primeros ejemplares móviles poseían las funcionalidades más elementales para realizar llamadas y gestionar la agenda y no es hasta el 92 cuando aparecen los primeros PDA (Personal Digital Assistant), con otras funcionalidades que un móvil básico no ofrecía.

El primer dispositivo considerado Smartphone fue el Ericsson GS88, además de teléfono móvil disponía de navegación web, correo, reloj mundial, modo avión, infrarrojos, etc. Fueron apareciendo diferentes derivados añadiendo más funciones a lo largo de la década del 2000 pero el que marcó un antes y un después en la industria fue el iPhone en 2007 y a día de hoy hay un mercado vigente donde la competencia entre fabricantes y plataformas es muy alta.

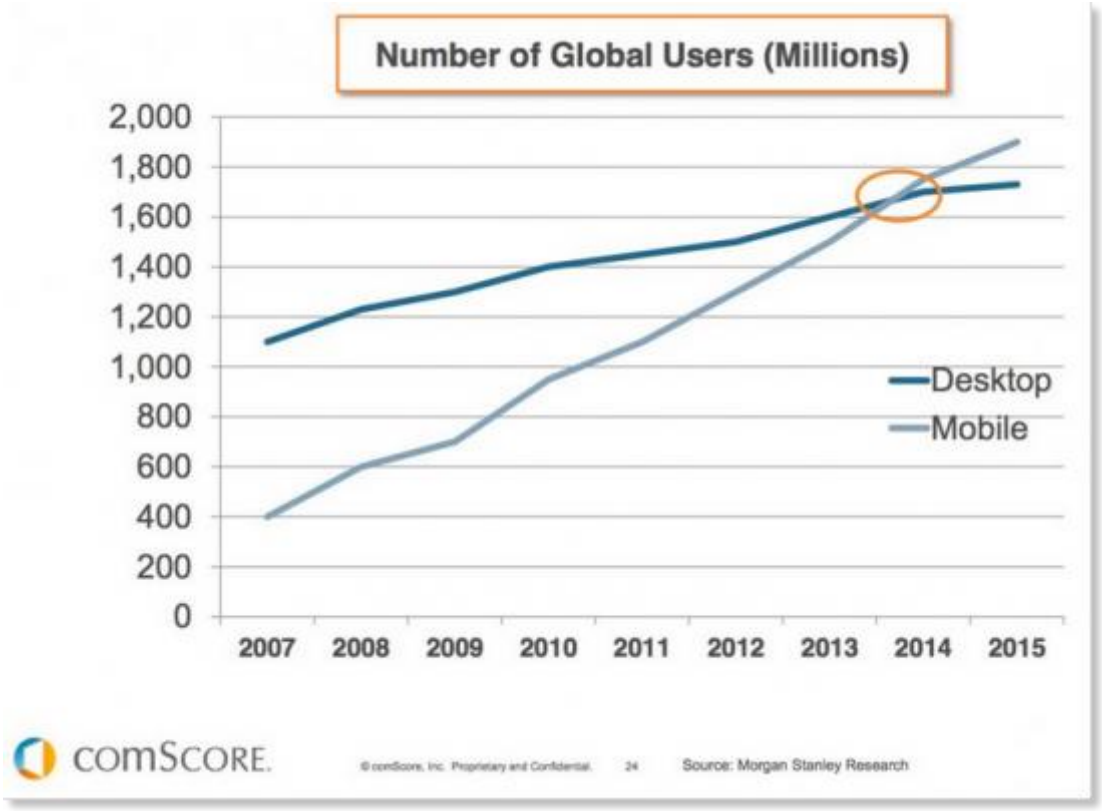


Figura 1. Gráfica de usuarios con dispositivos de escritorio/móviles [41]

Según los datos estadísticos de la mano de ComScore, los picos que alcanza el número de usuarios con dispositivos móviles a día de hoy es más de 4 veces y media respecto 2007, que es cuando se disparó el mercado. La pendiente de crecimiento de la tasa de usuarios es más pronunciada para aplicaciones móviles, de hecho, desde 2014 nos encontramos con más usuarios con smartphones que con equipos de escritorio.

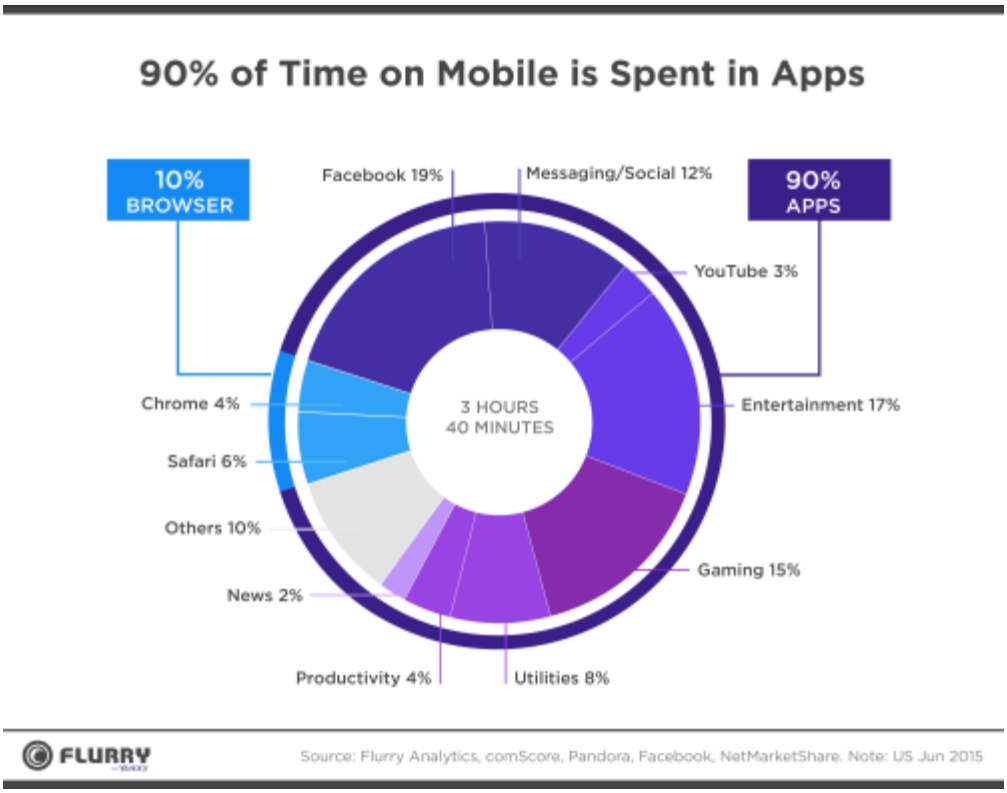


Figura 2. Gráfica del tiempo dedicado a aplicaciones por categoría [39]

Ahora bien, en términos de utilidad, el tiempo que dedicamos a un Smartphone lo abarcan mayormente las aplicaciones en concreto las de mensajería, redes sociales, entretenimiento y juegos. Sin embargo, ¿Cuánto tiempo dedicamos al ocio móvil en comparación a otras actividades ajenas?

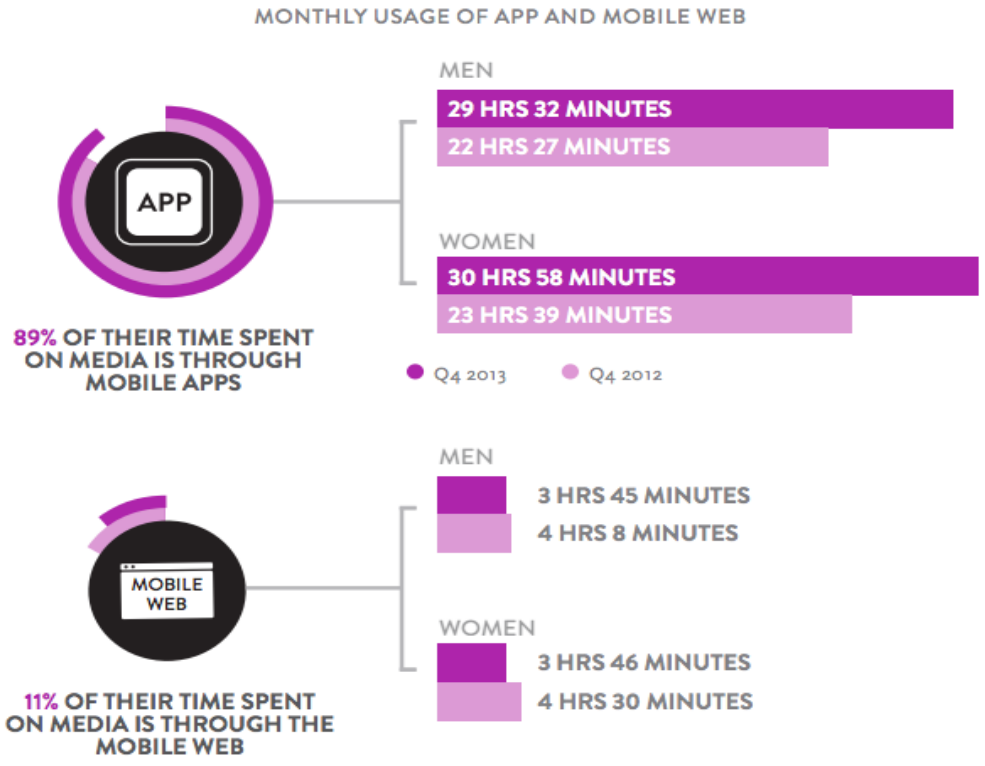


Figura 3. Gráfica por géneros del tiempo dedicado a aplicaciones y navegador móvil [40]

Nada más y nada menos un promedio de 30 horas por mes, aproximadamente una hora diaria, según la estadística del último cuarto de año en 2013 e in crescendo. Con todo esto queda demostrado que el impacto de los smartphones y las apps es muy grande en nuestro tiempo.

1.2. Análisis del mercado

Anteriormente describimos el panorama de los smartphones y qué tan presentes son en los consumidores y también describimos las horas dedicadas en cuanto a las diferentes categorías.

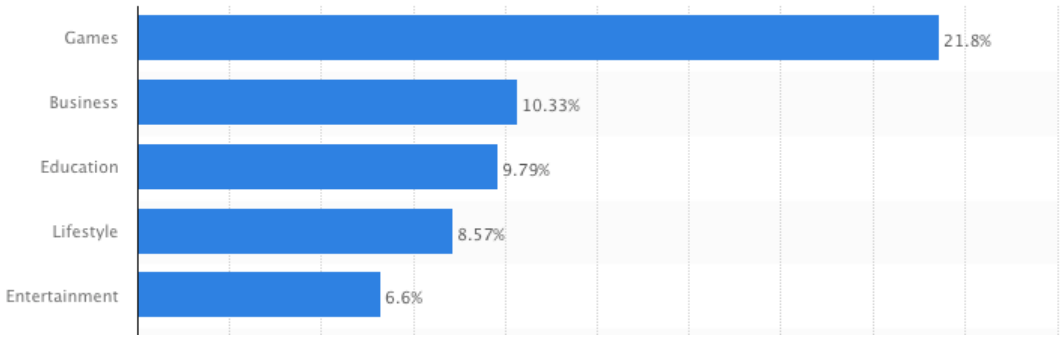


Figura 4. Gráfica de aplicaciones descargadas en la App Store por categoría [42]

Desde Julio de 2015 Marketplaces como la App Store y Google Play cuentan con 1,5 y 1,6 millones de aplicaciones respectivamente. Como podemos observar en la gráfica anterior muestra las categorías de aplicaciones más populares de la App Store. Si bien es cierto, el mercado de Games está muy masificado y muchas aplicaciones se quedan en el olvido o tienen poca longevidad en los dispositivos del consumidor, ya sea por qué carecen de ese gancho o retención o bien aparecen otras nuevas.

Nuestro proyecto queremos conducirlo a ese mercado, y en vistas generales es un arma de doble filo. Por una parte, es la categoría de mercado en la que hay más competencia y la que más diversidad de géneros que hay, por otra es la que tiene más demanda y supone una oportunidad aprovechable.

Uno de los géneros o categorías que se han hecho un hueco en los últimos tiempos son los de cartas. Ofrecen toda la parte de la jugabilidad, además de permitir al usuario ser el propietario de su propio contenido pudiendo modificarlo a su antojo y dado que son cartas tienen ese hándicap de ofrecer contenido coleccionable además de la parte competitiva.

HearthStone, es un ejemplo de juego de cartas RPG, spin off de la saga World of Warcraft uno de los MMORPG (Massive Multiplayer Online Rol Playing Game) más populares de la última década entre otros como **Magic 2015**, el mítico juego de cartas de rol ahora en un entorno digital además de muchos otros de la temática RPG-Cartas. **UNO**, también sería otro de los grandes a considerar, aunque no cuenta con ese factor de expandir la colección cuenta con una gran comunidad detrás por el juego de cartas original que Mattel puso a la venta desde 1992, además de las apps de juegos de cartas clásicos de la baraja española y la americana.

1.3. El impacto de la temática

Un juego por si solo difícilmente puede sobrevivir si no es vistoso y no es escalable. Si no se desarrollan nuevas actualizaciones para una app es complicada su permanencia en la lista de descargadas del usuario por muy bien pensada que esté la jugabilidad, por ello es necesario expandir el contenido para cuidar esa permanencia de la que hablamos.

Sin embargo, basándonos en el mercado del mismo género, ¿Cómo podemos llamar la atención del consumidor?

Los anteriores ejemplos sirven como precedentes, muchos de ellos provienen de temáticas que han tenido y tienen cierta repercusión y por eso el impacto que han tenido por el simple hecho de seleccionar una temática que atraiga. En primera instancia es un punto a considerar porque una vez la atención del usuario quede captada el siguiente paso es "seducirlo" con la aplicación en sí.

Ahora viene la parte complicada, ¿Qué temática seleccionamos para nuestra app? (En caso de querer monetizarla)

Corremos el riesgo de escoger temáticas ya existentes y con cierta repercusión que estén protegidas por copyright. En ese sentido no nos conviene enzarzarnos en asuntos legales. Por consiguiente, una temática que se adapta a nuestras necesidades es la de los memes.

Los memes se han convertido en una poderosa forma de divulgación de todo tipo slogans, frases hechas, parodias, modas, ideas, acontecimientos de una cultura, pudiendo adoptar la forma de un video, gif, audios, canciones, imágenes, sonidos, etc. La finalidad de éstos no es más que entretener y/o hacer reír a cualquier persona, y para lograrlo la manera de representarlos debe ser simple, efectiva y comprensible. Si se cumplen éstos tres puntos de la fórmula se pueden lograr sorprendentes reacciones por parte de los consumidores hasta el punto de convertir un meme en algo viral que en cuestión de clics haya sido difundido por cientos de miles de personas.

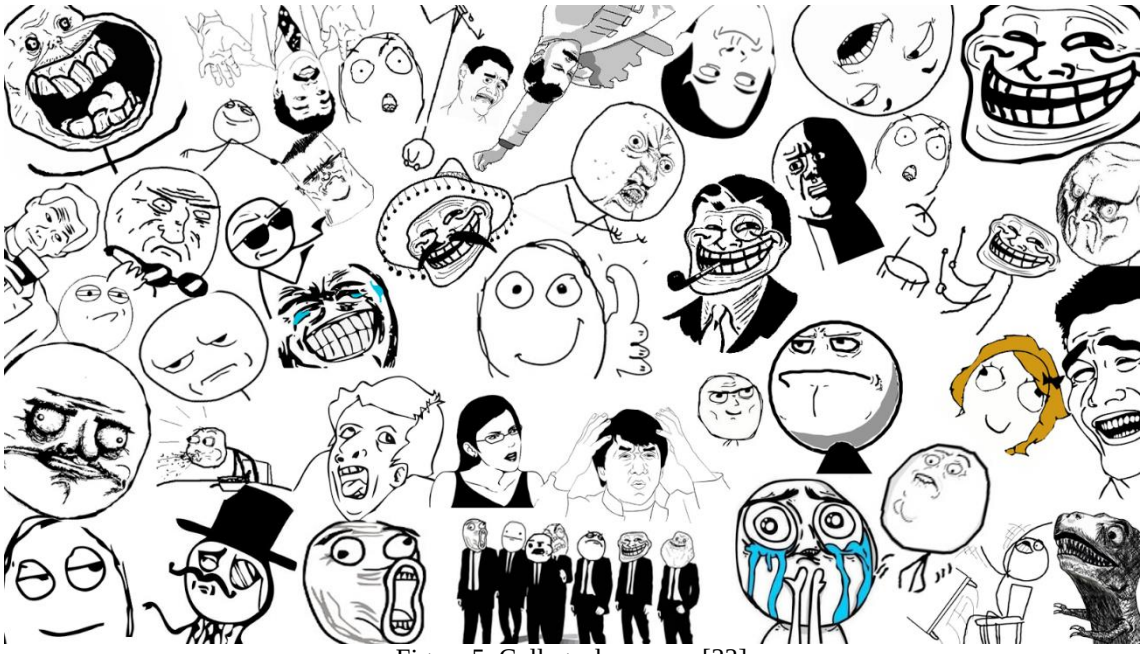


Figura 5. Collage de memes [33]

Ya que los memes tratan de alguna manera reproducir sucesos, cosas e ideas de una cultura, es esta la que da sustento a que se den nuevos memes y contra más progresa más memes se dan, por lo que podríamos afirmar que tienen caducidad, aunque algunos se reflotan y convierten en cliché, sigue siendo un "movimiento" que siempre va a generar nuevo contenido.

Por otra parte, nos ahorra los temas legales de los que hemos hablado porque la gran mayoría son libres de derechos de autor y siempre podemos contar con una comunidad que aporta nuevos memes al día.

1.3. Idea y necesidad identificada

La idea para nuestro proyecto es una aplicación móvil disponible para plataformas iOS, Android y Windows Phone. Será un juego de cartas / estrategia por turnos orientado al multijugador y también partidas contra la CPU locales cuya temática estará relacionada con los memes.

El juego premiará la buena competitividad con nuevas cartas a los jugadores, que contarán con múltiples modos de juego. No solamente podrán ampliar su colección y personalizar su baraja jugando, también pueden adquirir paquetes de cartas y adquirir nuevos packs de expansión del juego pagando un módico precio.

El juego ofrece un nuevo enfoque para divulgar y compartir memes y es a través del juego. La aplicación para el servicio Lite (libre de pago), contará con anuncios entre partida y partida, sin embargo, si el usuario decide suscribirse al servicio Premium podrá omitir los anuncios.

El propósito del proyecto es que sea escalable a corto y largo plazo, poniendo a disposición del usuario nuevos packs de expansión, desde nuevas modalidades de juego hasta nuevas cartas para la colección garantizando la robustez en el funcionamiento y rendimiento de la aplicación.

2. ANÁLISIS Y DISEÑO DE LA APLICACIÓN

2.1. Requisitos del sistema

Antes de formalizar todas las funcionalidades técnicas que compondrán Mexme el primer paso es definir el listado de especificaciones funcionales y operacionales del sistema, los requisitos funcionales.

Los requisitos no funcionales, al contrario que los funcionales, no describen que aplicaciones tiene un sistema, más bien lo que definen son las restricciones y las características. Provienen de la necesidad de satisfacer al usuario unas necesidades que van ligadas con los reglamentos de seguridad, políticas de privacidad i confort junto con otras necesidades relativas a la operatividad entre los componentes del sistema i la organización.

a) Requisitos funcionales

Los requisitos funcionales del sistema son los siguientes:

- Mexme es una aplicación cartas orientada a partidas de 2 jugadores y contra la CPU, dotada de IA.
- Los jugadores podrán entrar en salas de dos para jugar contra otros jugadores.
- Hay diferentes modos de juego disponibles en los que los que se aplican diferentes reglas.
- Cada usuario contará con una cuenta, identificada con un nickname único, en la que se registraran las partidas ganadas, empatadas, perdidas, ranking, la colección de cartas, el listado de amigos…
- Alternativamente cada usuario podrá gestionar su lista de amigos e invitarlos a jugar partidas siempre y cuando acepten la petición.
- Los jugadores podrán ampliar la colección de cartas conforme vayan jugando y/o ganando partidas.
- Todo usuario podrá acceder a su propio perfil y editar sus credenciales (foto de perfil, contraseña, etc…) además de visualizar su registro de victorias, derrotas, empates y ranking. Por otra parte, podrá acceder al perfil de otro usuario para ver sus datos (salvo la contraseña).
- De entre todas las cartas coleccionadas el usuario podrá modificar su propia baraja que será la que usará para las partidas.

b) Requisitos no funcionales

Algunos de los requisitos a satisfacer en términos de fiabilidad, usabilidad, interfaz, rendimiento, mantenibilidad, entre muchos otros son los siguientes:

- La Interfaz, a nivel de disposición de elementos en pantalla, debe ser accesible e intuitiva de manera que las interacciones sean lo más sencillas posibles y no obstaculicen las acciones independientemente del dispositivo en el que se den.
- Los elementos deben ser heterogéneos, es decir, el usuario al operar con ellos debe saber diferenciarlos e identificarlos.

- La paleta de colores debe ser consistente a lo largo de la aplicación y no afectar a la visibilidad de los elementos ni supongan perjudiciales para la vista del usuario.
- El tiempo de respuesta del sistema a las acciones debe ser razonable. Ni demasiado rápido ni demasiado lento, el usuario debe sentir que lleva el control de la aplicación y por ello será notificados cuando sea necesario.
- El usuario, como tal, es dueño de su privacidad y solo él puede decidir sobre la visibilidad de ciertos datos. Además, el sistema debe garantizar que la integridad y confidencialidad del usuario no sea afectada por factores ajenos.
- Los mensajes de comunicación entre componentes de todo el sistema deben ser lo más ligeros posibles para evitar todo tipo de incidencias.
- La aplicación debe ser consistente para reducir o evitar el desarrollo de parches de errores.
- El diseño técnico e implementación de la aplicación debe facilitar su escalabilidad para dar lugar a posibles evolutivos.
- Los costes deben ser los mínimos posibles en términos de herramientas de desarrollos y servicios proveídos (Hosting, base de datos…). Preferentemente el uso de recursos abiertos o servicios promocionales para estudiantes.
- El acceso a datos será eficiente y no habrá redundancia en la estructura de éstos.
- El sistema debe estar preparado a prueba de errores y ser capaz de reestablecerse si se dan.

2.2. Actores

Los actores son los miembros que se ven implicados en el uso de la aplicación y que por tanto interaccionan con ella. Dicho de otra manera, son los perfiles de usuario que ejecutan los casos de uso del sistema en distintos escenarios. Al ser perfiles distintos de usuario pueden tener diferentes privilegios, restricciones y acciones, aunque también pueden compartir algunos.

Para nuestro sistema a implementar los dos principales actores son el Usuario Administrador y el Usuario Jugador.

a) Usuario Jugador

El **Usuario Jugador**, un poco más limitado respecto al Administrador, puede darse de alta en la aplicación, identificarse, editar su baraja de cartas, jugar, invitar a otros jugadores a partidas, entrar en las salas, gestionar su listado de contactos y editar sus credenciales de perfil.

b) Usuario Administrador

El **Usuario Administrador**, además de poseer todos los privilegios de Jugador, tiene visibilidad de todos los usuarios que se han dado de alta en la aplicación, puede editar sus credenciales e incluso es capaz de editar algunas que un mismo Jugador no puede en su propio perfil.

2.3. Casos de uso

Los casos de uso definen escenarios en los que los actores de la aplicación ejecutan las funcionalidades. Una de las maneras de representarlos es a través diagramas donde aparece el actor, los casos de uso que desempeña y las relaciones que guardan.

a) Listado de casos de uso

Ahora que conocemos en profundidad los requisitos del sistema podemos listar todas las funcionalidades que debería implementar:

- Login
- Registro
- Editar Baraja
- Visitar mi Perfil
- Editar mi Perfil
- Ver Listado de Amigos
- Agregar Amigo
- Eliminar Amigo
- Visitar Perfil Amigo
- Invitar Partida
- Seleccionar Modo de Juego
- Seleccionar Oponente
- Ver Listado de Salas
- Entrar en Sala
- Ver Listado de Invitaciones
- Obtener Carta
- Jugar Partida

b) UML

El **UML** (Lenguaje Modelado Unificado) denota gráficamente un modelo de relaciones para visualizar y construir un sistema de software. De entre todas las relaciones posibles, nos centraremos en dos (extends y include) para vincular dos casos de uso diferentes. Veamos con los siguientes ejemplos como podemos reproducir las relaciones:

Usamos la relación <<**extends**>> para indicar que un caso de uso se puede dar otro, es decir la ejecución de uno no implica la de otro, más bien es alternativa. Por ejemplo, un usuario desea ver un vídeo y opcionalmente puede darle al botón Like:

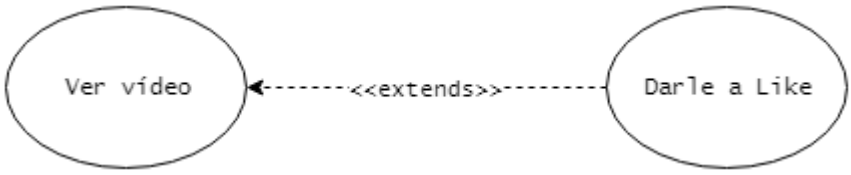


Figura 6. Diagrama de ejemplo de un extends

Para especificar que un caso de uso solo es posible realizarlo si estrictamente antes se ha ejecutado otro, usamos la relación **<<include>>**. Por ejemplo, un usuario desea realizar el pago de algo, pero antes necesariamente debe haber demandado algo:

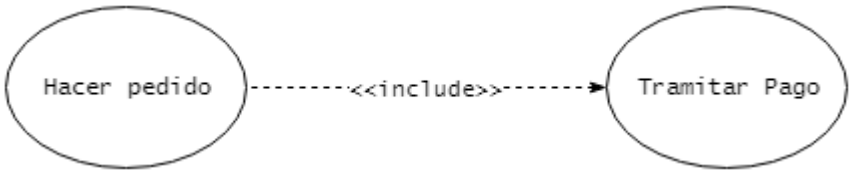


Figura 7. Diagrama de ejemplo de un include

c) Diagrama de casos de uso

Anteriormente declaramos y explicamos con ejemplos los tipos de relación que usaríamos, ahora es posible elaborar el diagrama para cada actor del sistema.

Figura 8. Diagrama de casos de uso para el usuario Jugador

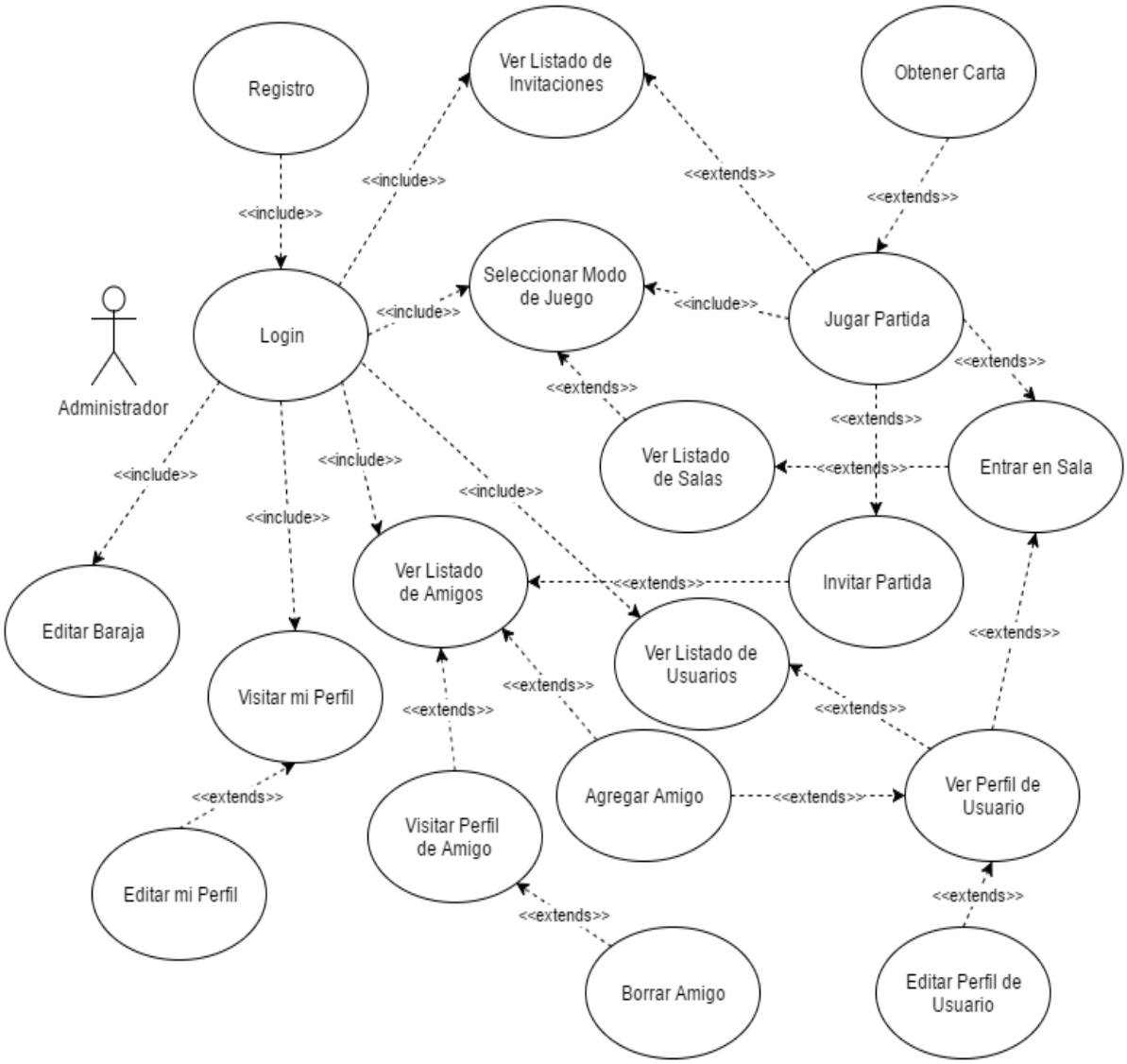


Figura 9. Diagrama de casos de uso para el usuario Administrador

c) Descripción de los casos de uso

Anteriormente hemos definido las relaciones entre casos de uso y los actores implicados en ellos. Ahora detallaremos el comportamiento de los distintos casos de uso. La siguiente plantilla muestra la estructura que usaremos para describir cada uno de los casos de uso del sistema:

Caso de uso	*Nombre del caso de uso*
Descripción	*Breve descripción del caso de uso*
Actores	*Usuarios implicados en la ejecución del caso de uso*
Precondición	*Estado o restricción necesaria para llevar a cabo el caso de uso*
Flujo principal	*Secuencia de pasos o estados para llevar a cabo el caso de uso hasta dar con la Postcondición*
Postcondición	*Output o estado final esperado en acabar el caso de uso*
Flujo alternativo	*Secuencia de pasos alternativa para un mismo user case que puede derivar a otra a Postcondición*

Tabla 1. Plantilla de casos de uso

Caso de uso	Registro
Descripción	Dar de alta un nuevo usuario para la aplicación
Actores	Admin, Jugador
Precondición	El usuario no se autentificado en el sistema y se encuentra en la pantalla de registro con todo cargado correctamente
Flujo principal	- El usuario rellena los campos de texto siguiendo el formato: ▪ Nick: alfanumérico [3-15 caracteres]. ▪ Password: alfanumérico [8-20 caracteres]. Debe tener una mayúscula y un carácter numérico mínimo. ▪ Repeat Password: alfanumérico [8-20 caracteres]. Debe tener una mayúscula y un carácter numérico mínimo y coincidir con el campo Password. ▪ Phone: numérico [9 caracteres]. - El usuario hace click en el botón *Sign In* y envía el formulario. - Si los campos cumplen el formato la aplicación enviará un mensaje conforme el alta ha sido realizada con éxito.
Postcondición	Registro completado y mensaje de aviso
Flujo alternativo	- El usuario rellena los campos sin respetar el formato de éstos u omitiendo algunos y envía el formulario. - El sistema retorna un mensaje de error notificando de los campos ausentes y/o los que no siguen el formato.

Tabla 2. Caso de uso Registro

Caso de uso	Login
Descripción	El usuario se autentifica con sus datos de acceso para hacer uso de la aplicación
Actores	Admin, Jugador
Precondición	El usuario debe disponer de una cuenta registrada y encontrarse en pantalla de Login sin la sesión iniciada y con los datos cargados correctamente
Flujo principal	- El usuario rellena los campos de texto siguiendo el formato: ▪ Nick: alfanumérico [3-15 caracteres]. ▪ Password: alfanumérico [8-20 caracteres]. Debe tener una mayúscula y un carácter numérico mínimo. - El usuario hace click en el botón *Login* y envía el formulario. - Si los datos de acceso funcionan se iniciará sesión con la cuenta del usuario.
Postcondición	Se inicia sesión con la cuenta del usuario y se accede a pantalla de bienvenida
Flujo alternativo	- El usuario rellena los campos de texto. - Al hacer click en Login recibe un mensaje conforme el nick y/o la contraseña son incorrectos.

Tabla 3. Caso de uso Login

Caso de uso	Editar baraja
Descripción	Personalización de la baraja, ésta será empleada para jugar las partidas
Actores	Admin, Jugador
Precondición	El usuario debe haber iniciado sesión y encontrarse en pantalla de bienvenida. Posteriormente accede a pantalla de Edición de Baraja
Flujo principal	- El usuario presenta a mano izquierda una columna con las 5 cartas que componen la baraja y en el cuerpo el listado de cartas. - Si hace clic cualquier carta de la baraja está quedará seleccionada. - Si ahora toca una carta de la colección, la de la colección pasará al mazo y la del mazo pasará a la colección.
Postcondición	La baraja ha sido actualizada y retorna a pantalla de bienvenida
Flujo alternativo	- El usuario en lugar de marcar una carta de la baraja marca una del mazo. - Ahora al marcar una carta del mazo, una pasa a colección y la otra pasa al mazo.

Tabla 4. Caso de uso Editar Baraja

Caso de uso	Ver Listado de Amigos
Descripción	Acceso al listado de contactos del usuario desde el que se podrá además de añadir otros, invitar a éstos a partidas y acceder a sus perfiles
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación y encontrarse en pantalla de bienvenida
Flujo principal	- Desde pantalla de bienvenida accedemos a través de un botón a pantalla del listado de amigos. - Se muestra un listado con el nombre del usuario y un botón para invitarlo a partida por cada entrada, así como otro botón para agregar nuevos.
Postcondición	Carga todos los usuarios (si los hay) en la pantalla del listado
Flujo alternativo	

Tabla 5. Caso de uso Ver Listado de Amigos

Caso de uso	Visitar mi Perfil
Descripción	Acceso al Perfil del usuario de la cuenta en el que se visualizan sus datos
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación y encontrarse en pantalla de bienvenida
Flujo principal	- Desde pantalla de bienvenida accedemos a través de un botón a pantalla de mi Perfil. - En la pantalla se visualiza la imagen del jugador, el nick, el teléfono, partidas ganadas/perdidas/empatadas, ranking, un botón para cambiar de contraseña, otro para cambiar la imagen y otro para cambiar la privacidad del número de teléfono.

Postcondición	Carga de la pantalla de mi Perfil con los datos de la cuenta actualizados
Flujo alternativo	

Tabla 6. Caso de uso Visitar mi Perfil

Caso de uso	Editar mi Perfil: Cambio de contraseña
Descripción	Modificación de credenciales y datos de la cuenta de usuario
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación y encontrarse en pantalla de mi Perfil
Flujo principal	- Al hacer clic sobre el botón cambio de contraseña se abrirá un pop-up en el que aparecen 3 campos. - Los rellenamos bajo el siguiente formato: ▪ Contraseña actual: alfanumérico [8-20 caracteres]. Debe tener una mayúscula, un carácter numérico mínimo y coincidir con la contraseña del usuario de la sesión. ▪ Nueva contraseña: alfanumérico [8-20 caracteres]. Debe tener una mayúscula y un carácter numérico mínimo. ▪ Repite contraseña: alfanumérico [8-20 caracteres]. Debe tener una mayúscula, un carácter numérico mínimo y coincidir con el campo Nueva contraseña. - Los campos cumplen los requisitos y el usuario es notificado.
Postcondición	Datos de la cuenta de la sesión actualizados
Flujo alternativo	- Si no se respeta el formato de las contraseñas modificadas debería aparecer un mensaje de error

Tabla 7. Caso de uso Editar mi Perfil: Cambio de contraseña

Caso de uso	Editar mi Perfil: Cambio de avatar
Descripción	El usuario fija una nueva imagen para su perfil
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación y encontrarse en pantalla de mi Perfil
Flujo principal	- Si el usuario hace clic en cambiar imagen del perfil se abrirá un pop-up con el listado de imágenes disponibles de la aplicación. - Al seleccionar una y confirmar se actualizará como nueva imagen del perfil.
Postcondición	Datos de la cuenta de la sesión actualizados
Flujo alternativo	- El usuario puede cancelar la operación de cambio de avatar cuando tenga el pop-up abierto y prevalecerá la imagen que ya tenía fijada.

Tabla 8. Caso de uso Editar mi Perfil: Cambio de avatar

Caso de uso	Editar mi Perfil: Fijar privacidad del teléfono
Descripción	Modificación de credenciales y datos de la cuenta de usuario
Actores	Admin, Jugador

Precondición	Haber iniciado sesión con la aplicación y encontrarse en pantalla de mi Perfil
Flujo principal	- El usuario puede cambiar la visibilidad del campo de su número de teléfono. Puede hacerlo público, privado o público para sus contactos a través de un switch de botones.
Postcondición	Datos de la cuenta de la sesión actualizados
Flujo alternativo	

Tabla 9. Caso de uso Editar mi Perfil: Fijar privacidad del teléfono

Caso de uso	Invitar Partida
Descripción	Envía una petición de partida a uno de los usuarios agregados
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación, encontrarse en pantalla del Listado de Amigos y tener al menos uno agregado
Flujo principal	- Para cada entrada del listado de usuarios agregados hay un botón para invitar a partida. - Al presionarlo se abrirá un pop-up para indicar el modo de juego de la partida y un botón para confirmar el envío. - Una vez haga clic en el botón de envío se enviará la petición.
Postcondición	- El usuario al que se le envía la petición recibe una notificación registrada en el listado de invitaciones. - El usuario de la sesión recibe un aviso conforme el envío de la petición ha sido exitoso y queda a la espera de la confirmación del oponente.
Flujo alternativo	- El usuario cancela la operación y decide no enviar la petición devolviéndole a pantalla del Listado de Amigos

Tabla 10. Caso de uso Invitar Partida

Caso de uso	Visitar Perfil Amigo
Descripción	Acceso a pantalla del perfil amigo para visualizar sus datos de cuenta
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación, encontrarse en pantalla del Listado de Amigos y tener al menos uno agregado
Flujo principal	- Cuando se hace clic sobre el nombre o el icono del listado el usuario accede a pantalla del perfil de su contacto agregado. - Los datos que puede visualizar son la imagen de usuario, nick, partidas ganadas/perdidas/empatadas, ranking y opcionalmente el número de teléfono dependiendo de la privacidad que haya fijado además del botón borrar amigo.
Postcondición	La pantalla se carga correctamente con los datos más recientes y respetando la privacidad (contraseña y/o número de teléfono)
Flujo alternativo	

Tabla 11. Caso de uso Visitar Perfil Amigo

Caso de uso	Borrar Amigo

Descripción	Eliminar un contacto agregado del listado
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación, encontrarse en pantalla del Perfil del Amigo y tener al menos uno agregado
Flujo principal	- En presionar el botón de borrado aparecerá un mensaje para confirmar definitivamente el borrado. - Si el usuario acepta borrará el usuario de su lista de contactos y volverá a la pantalla del listado. - Si por el contrario rechaza se encontrará de nuevo en pantalla del perfil.
Postcondición	El usuario es eliminado del listado y posteriormente redireccionado a la pantalla del listado de amigos con el listado actualizado
Flujo alternativo	

Tabla 12. Caso de uso Borrar Amigo

Caso de uso	Agregar Amigo
Descripción	Agregar un usuario de la aplicación al listado de amigos
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación, encontrarse en pantalla del Listado de Amigos o en pantalla Perfil del Usuario y que el sistema tenga al menos otro usuario diferente al de la sesión y no esté en lista de amigos..
Flujo principal	- En presionar el botón de agregar aparecerá una pop-up con un campo de texto para indicar el nick. - Si el nombre del usuario a agregar está correctamente escrito y ésta no figura en el listado actual, al confirmar será añadido al listado. - En ser exitosa la operación recibirá una notificación.
Postcondición	El usuario recibe una notificación de éxito y es enviado a pantalla del Listado de Amigos con el nuevo usuario añadido además de los existentes
Flujo alternativo	- Al indicar el nombre del usuario y confirmar, la operación no se realiza por uno de los siguientes motivos: ▪ El usuario no existe en la aplicación. ▪ El usuario está en el listado. ▪ No se puede añadir al administrador.

Tabla 13. Caso de uso Agregar Amigo

Caso de uso	Seleccionar Modo de Juego
Descripción	Conmuta el modo de juego para poder jugar
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación, encontrarse en pantalla de bienvenida.
Flujo principal	- Presionando cualquiera de los botones de cambio del modo de juego el usuario fijará un nuevo modo. - Si pulsa el botón derecho se fijará el siguiente modo de juego, en cambio, si pulsa el izquierdo fijará el anterior.
Postcondición	Una caja indica el nombre del modo fijado
Flujo alternativo	

Tabla 14. Caso de uso Seleccionar Modo de Juego

Caso de uso	Ver Listado de Invitaciones
Descripción	Lista las invitaciones de partidas enviadas por otros usuarios
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación, encontrarse en pantalla de bienvenida
Flujo principal	- Al hacer clic en el botón de acceso al Listado de Invitaciones el usuario es redireccionado. - El usuario visualizará cada entrada del listado (si las hay) con el nombre del usuario que invita, el modo de juego de la partida a la que invita y un botón para iniciarla
Postcondición	El listado se carga correctamente con las peticiones más recientes
Flujo alternativo	

Tabla 15. Caso de uso Ver Listado de Invitaciones

Caso de uso	Ver Listado de Salas
Descripción	Lista las salas para un modo de juego concreto
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación, encontrarse en pantalla de bienvenida
Flujo principal	- Al hacer clic en el botón de iniciar partida aparecerá una nueva vista con dos opciones, IA y Solo. - Al presionar Solo para un modo de juego aparecerá un listado de salas con el aforo disponible. Dado que es un juego PVP (Jugador contra Jugador) el aforo máximo será de dos plazas.
Postcondición	El listado se carga correctamente mostrando la disponibilidad de las salas
Flujo alternativo	

Tabla 16. Caso de uso Ver Listado de Salas

Caso de uso	Entrar en Sala
Descripción	Entrar en una de las salas para iniciar una partida de cartas o esperar a otro jugador. La sala debe tener menos de 2 jugadores
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación, encontrarse en pantalla del Listado de Salas.
Flujo principal	- Si la sala a acceder no tiene el aforo completo se entrará al hacer clic sobre la sala del listado. - De haber un oponente esperando se visualizará su nombre y el botón para ver su perfil y un botón para empezar partida.
Postcondición	El listado se carga correctamente mostrando la disponibilidad de las salas
Flujo alternativo	

Tabla 17. Caso de uso Entrar en Sala

Caso de uso	Visitar Perfil Usuario

Descripción	Acceso a pantalla del perfil usuario para visualizar sus datos de cuenta
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación y encontrarse dentro de una sala
Flujo principal	- Cuando se hace clic sobre el botón ver perfil, el usuario accede a pantalla del perfil del usuario de la sala. - El usuario visualizará la imagen de usuario, nick, partidas ganadas/perdidas/empatadas, ranking y opcionalmente el número de teléfono dependiendo de la privacidad que haya fijado además del botón de agregar amigo.
Postcondición	La pantalla se carga correctamente con los datos más recientes y respetando la privacidad (contraseña y/o número de teléfono)
Flujo alternativo	

Tabla 18. Caso de uso Visitar Perfil de Usuario

Caso de uso	Jugar Partida (Contra la CPU)
Descripción	Realizar una partida de dos jugadores siguiendo las reglas del modo de juego establecido
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación y encontrarse en pantalla de bienvenida.
Flujo principal	- Desde pantalla de bienvenida si el usuario presiona Jugar Partida con un modo de juego ya seleccionado. - Ahora en una nueva pantalla tendrá la opción de jugar contra la IA (Inteligencia Artificial) o bien entrar al listado de salas. - Al presionar el botón IA se iniciará la partida contra la CPU. - El usuario juega la partida correctamente hasta su fin.
Postcondición	El usuario se encuentra en pantalla de Partida y las animaciones e interacciones se aplican correctamente
Flujo alternativo	- El usuario decide rendirse durante el transcurso de la partida.

Tabla 19. Caso de uso Jugar Partida (Contra la CPU)

Caso de uso	Jugar Partida (Multijugador)
Descripción	Realizar una partida de dos jugadores siguiendo las reglas del modo de juego establecido
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación y satisfacer cualquiera de las siguientes precondiciones, cada una correspondiente a un flujo diferente: - Encontrarse en una sala con un oponente preparado para iniciar la partida. - Haber enviado una invitación a un amigo y que este la haya aceptado. - Haber recibido una invitación, estar en la pantalla del listado de invitaciones y aceptar la invitación.
Flujo principal	- Los usuarios juegan la partida correctamente hasta su fin.

Postcondición	La partida ha finalizado y el resultado queda registrado para cada jugador.
Flujo alternativo	Alguno de los jugadores decide rendirse durante el transcurso de la partida.

Tabla 20. Caso de uso Jugar Partida (Multijugador)

Caso de uso	Obtener Carta
Descripción	Adquisición de una carta para ampliar la colección tras ganar la partida
Actores	Admin, Jugador
Precondición	Haber iniciado sesión con la aplicación, jugar una partida y haberla finalizado saliendo exitoso.
Flujo principal	- El jugador finaliza la partida como vencedor e irá a nueva vista que mostrará las las 5 cartas del oponente. - El usuario debe seleccionar una y confirmar la acción. - En confirmar será reenviado a pantalla de bienvenida con los datos registrados.
Postcondición	La carta se suma a la colección, se registran los datos de la partida y redirecciona a pantalla de bienvenida
Flujo alternativo	- El usuario que finaliza la partida como perderdor esperará a que el oponente elija una de sus 5 cartas.

Tabla 21. Caso de uso Obtener Carta

Caso de uso	Editar Perfil de Usuario: Cambio de credenciales
Descripción	Modificación de credenciales y datos de la cuenta de un usuario
Actores	Admin
Precondición	Haber iniciado sesión con la aplicación como administrador y encontrarse en pantalla de Perfil de Usuario
Flujo principal	- Al hacer clic sobre el botón Editar los campos pasarán a estar habilitados para edición. - Los rellenamos siguiendo el formato: ▪ Nick: alfanumérico [3-15 caracteres]. ▪ Password: alfanumérico [8-20 caracteres]. Debe tener una mayúscula y un carácter numérico mínimo. ▪ Repeat Password: alfanumérico [8-20 caracteres]. Debe tener una mayúscula y un carácter numérico mínimo y coincidir con el campo Password. ▪ Phone: numérico [9 caracteres]. - Los campos que se han modificado cumplen con los requisitos, quedan actualizados y el usuario recibe notificación conforme la operación ha sido exitosa.
Postcondición	Datos de la cuenta de la sesión actualizados
Flujo alternativo	- Si el formato no cumple los requisitos y el formato anteriores debería aparecer un mensaje de indicando los campos erróneos.

Tabla 22. Caso de uso Editar Perfil de Usuario: Cambio de credenciales

Caso de uso	Editar Perfil de Usuario: Cambio de avatar
Descripción	Modificación de credenciales y datos de la cuenta de un usuario

Actores	Admin
Precondición	Haber iniciado sesión con la aplicación como administrador y encontrarse en pantalla de Perfil de Usuario
Flujo principal	- El usuario administrador hace clic en cambiar imagen del perfil y se abre un pop-up con el listado de imágenes disponibles de la aplicación. - Al seleccionar una y confirmar se actualizará como nueva imagen del perfil.
Postcondición	Datos de la cuenta de la sesión actualizados
Flujo alternativo	- Si por el contrario no se desea cambiar, puede cancelar la operación haciendo clic en el botón atrás

Tabla 23. Caso de uso Editar Perfil de Usuario: Cambio de avatar

Caso de uso	Editar Perfil de Usuario: Fijar privacidad del teléfono
Descripción	Modificación de credenciales y datos de la cuenta de un usuario
Actores	Admin
Precondición	Haber iniciado sesión con la aplicación como administrador y encontrarse en pantalla de Perfil de Usuario
Flujo principal	- El administrador cambia la visibilidad del campo de su número de teléfono con un switch de botones. - Puede hacerlo público, privado o público para sus contactos.
Postcondición	Datos de la cuenta de la sesión actualizados
Flujo alternativo	

Tabla 24. Caso de uso Editar Perfil de Usuario: Fijar privacidad del teléfono

Caso de uso	Ver Listado de Usuarios
Descripción	Acceso al listado de usuarios del sistema desde el que se podrá acceder a sus perfiles y editarlos.
Actores	Admin
Precondición	Haber iniciado sesión con la aplicación como administrador y encontrarse en pantalla de bienvenida
Flujo principal	- Desde pantalla de bienvenida accedemos a través de un botón a pantalla del listado de usuarios. - Se muestra un listado con el nombre del usuario al que se puede acceder y editar el perfil.
Postcondición	Carga el listado de todos los usuarios del sistema
Flujo alternativo	

Tabla 25. Caso de uso Ver Listado de Usuarios

2.4 Arquitectura del sistema

a) Patrones de diseño

Los patrones de diseño plantean soluciones a problemas de la ingeniería del software. En nuestro caso el problema o más bien la cuestión es ¿Qué arquitectura lógica seguirá nuestra aplicación?

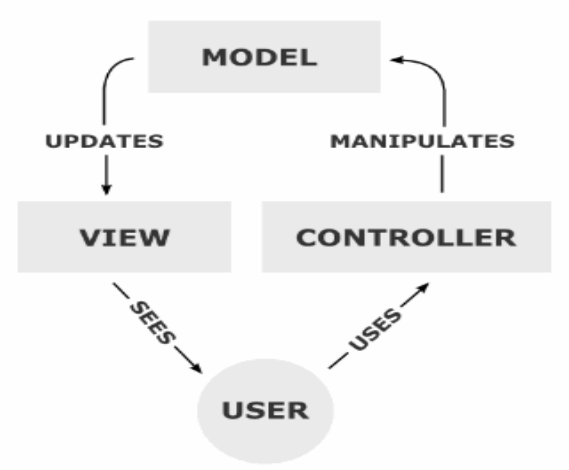


Figura 10. Diagrama MVC [38]

El patrón que más se adapta a nuestra necesidad es el MVC (Modelo Vista Controlador). Está dividido en tres capas, tal y como indica su nombre una corresponde a la capa del Modelo (Persistencia), Controlador (Lógica o de Negocio), Vista (Presentación).

- Modelo: La capa de modelo es la que controla todo lo referente al modelo de datos y que tiene que ver con los accesos a la base de datos, desde los privilegios del usuario que accede hasta las inserciones, borrados y consultas de los datos.
- Controlador: Se encarga de actualizar la vista del usuario y de manejar los eventos cuando el usuario interactúa con ella. También se encarga de invocar accesos a la base de datos y mediante los datos recogidos actualiza las vistas.
- Vista: Es la capa de presentación final y por tanto la interfaz final con la que interactúa el usuario.

b) Arquitectura lógica

En la ésta sección vamos a explicar cómo hemos aplicado el Modelo-Vista-Controlador a nuestra aplicación, como se relacionan los diferentes componentes de software, que jerarquía guardan y el flujo de trabajo cuando se dan operaciones.

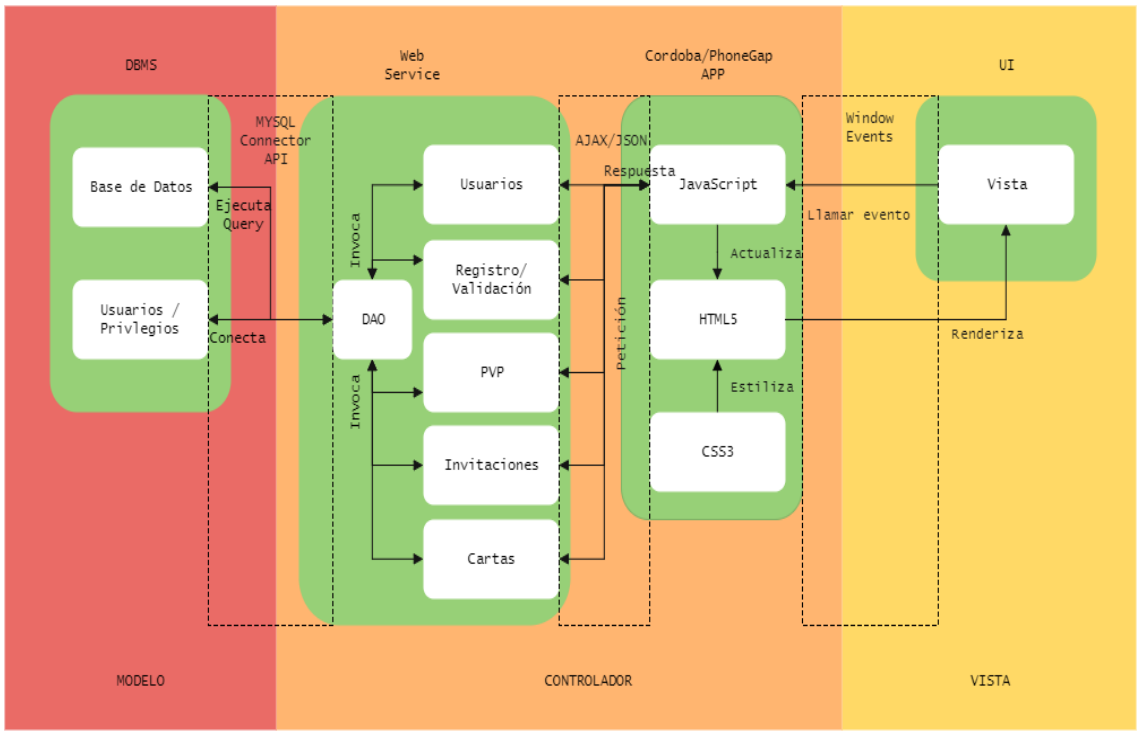


Figura 11. Diagrama de bloques

El diagrama de bloques anterior reproduce gráficamente como se comunican los bloques y subbloques de software para percibir una visión amplia de la arquitectura lógica.

Capa de presentación

Es el único bloque que compone la capa de Presentación. Siendo nuestra aplicación de móvil la pantalla será el elemento principal de interacción con el usuario a través todos los elementos que se rendericen en ella: slide bars, botones, cajas de texto, etc…

Al realizar ciertas acciones sobre la pantalla se disparan ciertos eventos en la ventana que pueden invocar el código JavaScript implementado en la aplicación. Éste tipo de acciones de eventos se definen a través de atributos en el HTML.

Capa de negocio

Tal y como hemos diseñado la aplicación la capa de lógica del negocio la componen dos bloques, por una parte, el Servicio Web y por otra la aplicación en sí en el lado cliente.

Normalmente en las aplicaciones webs, desde la vista mostrada en el Web Browser se llama al Web Service para que genere una nueva vista o al menos una parte de la vista a través de los datos de persistencia. También desde el lado del cliente actúa el JavaScript y sus respectivas librerías (JQuery/JQuery Mobile) para hacer el contenido más dinámico sin necesidad de pasar por el Servidor.

Por ejemplo, la validación de los campos de texto al enviar un formulario se podría revisar en el lado del cliente, por lo que respecta el formato de los campos. Si deseamos verificar y/o recuperar datos ya es estrictamente necesario pasar por el servidor.

Para las peticiones Client-Side usaremos AJAX para enviar y recibir datos al servidor usando JSON como formato de datos entre el cliente y el servidor donde los datos se encapsulan en pares llave-valor.

En nuestra aplicación todas las vistas y los controladores de las vistas están ubicados en el contenedor de PhoneGap/Cordova (lado cliente), por el simple hecho de no sobrecargar el sistema haciendo peticiones donde el tipo de contenido esperado en la respuesta, que viene dado en la cabecera, acostumbra a ser más denso.

Por ejemplo, al hacer una petición http al servidor con un MIME type *text/html*, la response puede contener imágenes incrustadas en el HTML, además de toda la estructura en sí y los estilos. En cambio, si el contenido que solicitamos es *application/json*, *la respuesta* será un objeto que serializará pares Key-Value.

Los datos que precisamente recogeremos de la response, los usaremos para completar la vista a través del JavaScript usando el contenido estático (HTML, CSS, imágenes, etc…) en el contenedor PhoneGap/Cordova.

Una vez el Web Service arranque instanciará en el contenedor cada uno de los Servlets implementados (una vez cada uno) esperando recibir peticiones y en caso de que lleven tiempo en desuso la instancia será eliminada para ahorrar memoria.

Como ya hemos comentado con anterioridad el tipo MIME de los datos de la response no serán *txt/html*, si no, *application/json*. Lo que se retornará serán los resultados de las consultas ejecutadas serializadas en un objeto en formato JSON.

En cada módulo de la aplicación se realizarán por separado diferentes funciones por lo que tendrá diferentes Servlets esperando ser llamados:

- **Módulo de Usuarios**: Se encargará de todo lo relacionado con los datos de los usuarios, desde el acceso a los credenciales y listados de amigos hasta sus modificaciones.
- **Módulo Registro/Autentificación**: Es el encargado de dar de alta nuevas cuentas de usuario en la aplicación, de validar el inicio de sesión del usuario, así como los privilegios y concesión al usuario los datos de sesión que se almacenarán en el *sessionStorage* de la aplicación en el lado del cliente.
- **Módulo PVP**: La principal función es gestionar las salas y su disponibilidad. Alternativamente se encarga del progreso de una partida entre dos jugadores interconnectados, actuando los WebSocket como intermediarios en cada jugada. Más adelante lo explicaremos con más detención.
- **Módulo de Invitaciones**: Se ocupa de gestionar las invitaciones activas entre usuarios para iniciar partidas y de limpiarlas de la base de datos una vez caduquen o sean denegadas.
- **Módulo de Cartas**: Se encarga de gestionar todas las operaciones implicadas en la modificación de la colección de cartas de los usuarios, así como la baraja.

Ahora bien, ¿Cómo se comunica la base de datos y el Web Service?

A través de la API MySQL Connector tendremos un nexo entre ambas capas. La clase DAO (Data Access Object) será invocada por los Servlets cuando quieran realizar consultas a la base de datos.

```
Connection connection = DriverManager.getConnection("[URL del SQL
Server]/[nombre del Schema]?user=[nombre del
usuario]&password=[contraseña del usuario]");
```

Tabla 26. Ejemplo de conexión con la base de datos en Java

La clase Connection nos permitirá iniciar una nueva sesión con la base de datos pudiendo ejecutar operaciones en ese mismo contexto establecido, siempre y cuando el usuario de la sesión cumpla con los privilegios.

```
connection.createStatement().executeQuery("[String de la Query]");

/* Ejecuta la query siguiendo la forma bàsica de una Selección
SELECT [ALL | DISTINCT ]
             <nombre_campo> [{,<nombre_campo>}]
FROM <nombre_tabla>|<nombre_vista>
        [{,<nombre_tabla>|<nombre_vista>}]
[WHERE <condición> [{ AND|OR <condición>}]]
[GROUP BY <nombre_campo> [{,<nombre_campo >}]]
[HAVING <condición>[{ AND|OR <condición>}]]
[ORDER BY <nombre_campo>|<indice_campo> [ASC | DESC]
                [{,<nombre_campo>|<indice_campo> [ASC | DESC ]}]]
*/
```

Tabla 27. Ejemplo de invocación de query en Java

Con los métodos *executeQuery* y *executeUpdate*, es posible ejecutar consultas, inserciones y/o actualizaciones en la base de datos pasando como argumento el String de la consulta respetando la sintaxis.

Una vez se recogen los resultados tras la ejecución, se introducen en un ResultSet que contiene todas las filas con todos los atributos (columnas) de la selección para devolvérselos al Servlet que ha invocado la query. Una vez el Servlet recoja los datos los enviará de nuevo al cliente, pero no sin antes formatearlos a JSON.

Capa de presentación

Esta capa del MVC solo la compone el gestor de base de datos alojado en un servidor SQL. Simplemente ejecutará las querys que se envíen desde la aplicación dada una sesión establecida según los privilegios que la rigen.

c) Arquitectura física

En el esquema inferior se puede observar cómo están relacionados los componentes físicos que conforman Mexme en el escenario de una partida. Es una aplicación orientada a la conexión multijugador por medio de un servidor intermediario, donde está alojado el Web Service, por tanto, deberá ser accesible a la red en un Hosting desde nuestros smartphones (también conectados).

El hecho de tener un servidor intermediario favorece el control del progreso de una partida para garantizar la estabilidad. Por ser prácticos las partidas contra la CPU no necesitarán de conexión a internet ejecutándose todas en el dispositivo local.

Separadamente es preferible separar en otro servidor el almacenamiento de la base de datos (accesible por IP) en lugar de alojarlo todo en un mismo equipo para evitar daños colaterales en caso de cualquier caída del sistema.

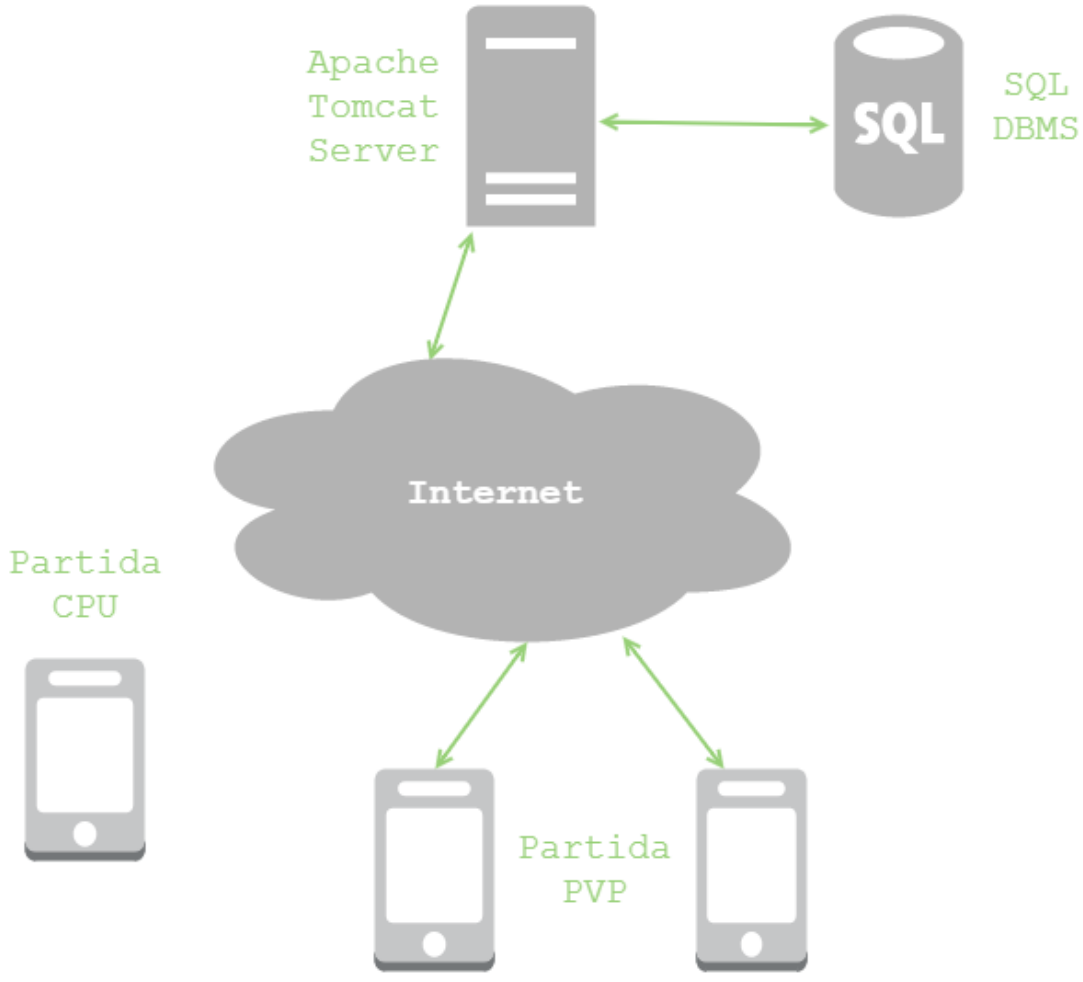


Figura 12. Diagrama de arquitectura física

2.5. Modelo de datos

El modelo de Datos de una aplicación define la información estructurada que la compone, por tanto, todo lo relacionado con la manipulación y acceso a ésta no tiene nada que ver. Solamente define como se organizan y agrupan los datos. ¿Qué tiene de bueno?

- Los datos son independientes y su acceso es eficiente.
- Garantiza la integridad de los datos y consistencia durante su ciclo de vida.
- La administración de los datos se hace de manera uniforme.

a) El modelo Entidad-Relación

El modelo ER (Entidad-Relación) representa las relaciones entre las diferentes Entidades (Tablas) dado un Schema de la base de datos. Los atributos (columnas) de las entidades también vienen dados, sin embargo, no llega al detalle de especificar los tipos de cada uno además de ciertas constraints. Más bien lo que especifica es la cardinalidad y la participación en relaciones binarias, reflexivas, n-arias e incluso la herencia, aunque en nuestro modelo solo usaremos las dos primeras.

En una Entidad subrayaremos los atributos para designarlos como clave primaria. Las claves foráneas en cambio vienen implícitas al representar una relación entre entidades.

En las capturas siguientes, a través de un ejemplo, denotamos la cardinalidad para una relación binaria:

N:1 / 1:N

Figura 13. Relación N a 1

Se puede leer como “varios empleados trabajan para un único departamento”. En este caso la relación es N:1 si fuese 1:N basta con cambiar el sentido.

1:1

Figura 14. Relación 1 a 1

Cada despacho solo puede estar ocupado por un único empleado.

N:N

Figura 15. Relación N a N

Un empleado puede participar en varios proyectos y del mismo modo en un proyecto pueden participar varios empleados.

Como ya hemos dicho también usaremos relaciones reflexivas. Las podemos representar gráficamente del mismo modo salvo que la relación de una entidad será consigo misma. Si una entidad posee participación total todas las instancias de dicha aparecerán en la relación, si por el contrario es parcial, algunas instancias pueden aparecer en la relación. Ahora que sabemos esto, ¿Cómo lo representamos?

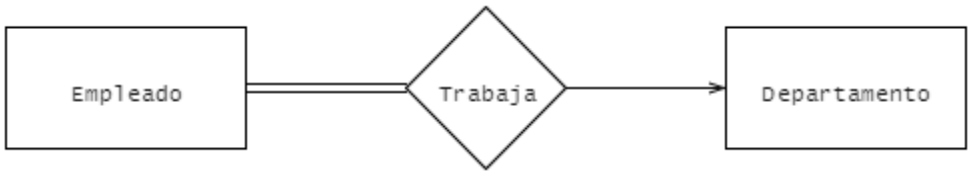


Figura 16. Participación total

Con la doble línea representamos la participación total. Podremos transcribir este ejemplo como que cualquier empleado pertenece a un departamento (único).

Figura 17. Participación parcial

Volviendo al ejemplo anterior, la participación parcial se representa con una única línea, así pues, indicamos que algunos empleados pertenecen a un departamento.

Hay varias maneras de representar los diagramas ER usando diferentes nomenclaturas, para nuestro caso hemos optado por usar las que acabamos de explicar.

b) Especificaciones del Schema

En la sección anterior mediante ejemplos hemos explicado el modelo que seguiremos para construir nuestro modelo de datos de la aplicación en un diagrama ER. Ahora no hay más que transcribir las siguientes especificaciones a diagrama:

- Los usuarios pueden tener varios usuarios como amigos.
- Los usuarios pueden jugar varias partidas con otros usuarios.
- Los usuarios pueden invitar varias partidas a otros usuarios.
- Pueden existir varias instancias de una misma carta.
- Cada usuario tendrá una colección de al menos 5 cartas.

- La baraja de un usuario estará compuesta por 5 cartas de su colección.
- Cada usuario puede asignarse una foto como avatar.

c) Diagrama ER

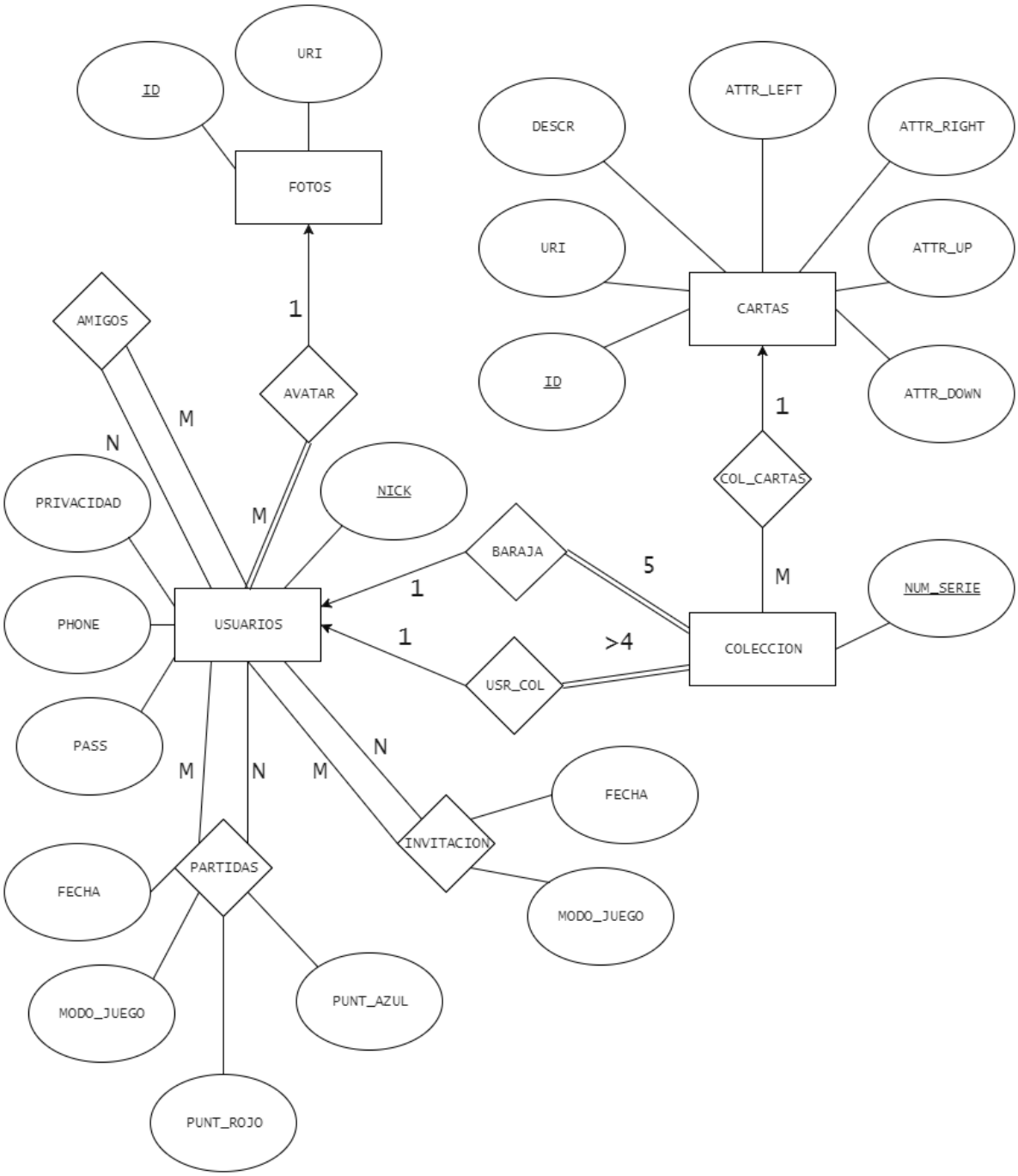


Figura 18. Diagrama ER

d) Descripción del Schema

Cuando elaboramos el Diagrama ER tenemos una visión periférica de las entidades y atributos del modelo lógico de los datos. El siguiente paso pues es traducirlo al RQL (Relational Query Language) en el que se define una sintaxis y una semántica para fortalecer la consulta de datos.

Técnicamente lo que haríamos es construir un Script SQL para conformar las tablas del Schema, sin embargo, describiremos formalmente las tablas a continuación.

Tabla **USUARIOS**: Agrupa toda la información relacionada con el perfil, así como sus credenciales, principalmente serán usados como datos de la sesión del usuario y por tanto todos los atributos deberán ser no nulos.

- **NICK**: Clave primaria de la tabla empleada para designar el username del usuario. En los wireframes la longitud máxima de la caja de texto del nick era de 15 caracteres, así pues, usaremos un VARCHAR (15) para que coincidan.
- **PHONE**: En los wireframes los indicamos como caracteres numéricos de longitud 9, el tipo empleado será DECIMAL (10,0).
- **PASS**: Referente a la contraseña, en operaciones previas controlaremos el formato definido, sin embargo, a nivel de longitud sabemos que será un VARCHAR (20).
- **AVATAR**: En lugar de crear una nueva tabla como a cada usuario le corresponde una imagen de avatar lo que haremos con éste atributo será hacerlo clave foránea del ID de la tabla FOTOS en lugar de crear una tabla para la relación entre la entidad FOTOS y USUARIOS. El tipo de este atributo (INT) debe ser el mismo que el de la tabla FOTOS.
- **PRIVACIDAD**: Usado para indicar la visibilidad del teléfono en la aplicación. Usaremos un VARCHAR (1) y tomará el valor ‘A’ para que solo los amigos puedan verlo, ‘T’ para hacerlo visible para cualquier usuario y ‘P’ para hacerlo privado.

Tabla **AMIGOS**: En el diagrama se representa como una relación reflexiva para designar quién tiene a quién agregado como amigo en la aplicación. La participación de la tabla USUARIOS es parcial por lo que podremos encontrarnos entradas vacías dado un usuario.

- **AGREGANTE**: Clave foránea del atributo NICK en la tabla USUARIOS para designar al usuario que ha agregado al amigo, por tanto, VARCHAR (15).
- **AGREGADO**: Clave foránea del atributo NICK en la tabla USUARIOS para designar al usuario que ha sido agregado como amigo, VARCHAR (15).

Ambos atributos formarán un par UNIQUE para evitar pares duplicados.

Tabla **FOTOS**: Agrupa datos relacionados con las fotos que se usan en la aplicación, principalmente para avatares de perfiles. Ambos campos deben ser no nulos.

- **ID**: Clave primaria para identificar la imagen, será del tipo INT, siendo referenciado en la tabla USUARIOS tal y como hemos explicado.

- **URI**: Referente a la ubicación de la imagen en la aplicación (directorios usados para contenido estático). Resulta más eficiente que guardar la imagen en si dentro de la tabla por su ligereza, el tipo definido será VARCHAR (20).

Tabla **PARTIDAS**: Para registrar los datos referentes a la partida, a través de este en la aplicación podremos esbozar el marcador de victorias, derrotas, empates y rango.

- **ROJO**: Clave foránea referenciando al atributo NICK (VARCHAR (15)) de USUARIOS para indicar el jugador rojo.
- **AZUL**: Clave foránea referenciando al atributo NICK (VARCHAR (15)) de USUARIOS para indicar el jugador azul.
- **FECHA**: Usaremos un DATETIME para fecha en la que se inicia la partida.
- **MODO_JUEGO**: Tipo INT para indicar el Modo de Juego de la Partida.
- **PUNT_ROJO:** Para contabilizar la puntuación del Jugador Rojo usaremos un INT.
- **PUNT_AZUL:** Para contabilizar la puntuación del Jugador Azul usaremos un INT.

Tabla **INVITACION**: Almacena todos los datos referentes para iniciar partidas entre dos jugadores. La participación de usuarios es parcial, no tienen por qué existir todos en la relación.

- **ANFITRION**: Clave foránea referenciando al atributo NICK (VARCHAR (15)) de USUARIOS para indicar el usuario que envía la solicitud.
- **INVITADO**: Clave foránea referenciando al atributo NICK (VARCHAR (15)) de USUARIOS para indicar el usuario que envía la solicitud.
- **FECHA**: Usaremos un DATETIME para fecha en la que envía la petición.
- **MODO_JUEGO**: Tipo INT para indicar el Modo de Juego de la Partida.

Tabla **CARTAS**: Registra todos los datos de cada uno de los tipos de cartas del juego.

- **ID**: Clave primaria para el identificador de carta (INT).
- **URI**: VARCHAR (20) para indicar la ubicación en la aplicación de la representación pictórica de la carta.
- **DESCR**: Breve descripción de la carta para ser distinguida por los usuarios en lugar de usar el identificador en base de datos, VARCHAR (20).
- **ATTR_LEFT**: Atributo numérico empleado en la mecánica del juego, DECIMAL (1,0).
- **ATTR_RIGHT**: Atributo numérico empleado en la mecánica del juego, DECIMAL (1,0).
- **ATTR_UP**: Atributo numérico empleado en la mecánica del juego, DECIMAL (1,0).
- **ATTR_DOWN**: Atributo numérico empleado en la mecánica del juego, DECIMAL (1,0).

Tabla **COLECCION**: Almacena cada una de las instancias de los diferentes tipos de cartas identificados por un número de serie único. Dado que cada instancia será para una clase de carta, podemos ahorrarnos la tabla de la relación COL_CARTAS.

- **NUM_SERIE**: Clave primaria del tipo INT para identificar la instancia de una carta.
- **TIPO**: Clave foránea del atributo ID en la tabla CARTAS para el identificador de la clase de carta.

Tabla **USR_COL**: Salvaguarda la colección de cartas de un usuario. Para poder jugar al juego se necesitan al menos 5 cartas, así que el identificador del usuario aparecerá 5 veces al menos.

- **NUM_SERIE**: Clave foránea referenciando NUM_SERIE en la tabla COLECCION para las instancias de las cartas.
- **USUARIO**: Clave foránea referenciando NICK en la tabla USUARIOS para indicar el propietario de la carta.

Tabla **BARAJA**: Guarda las 5 cartas que el usuario usará para jugar. El usuario aparecerá en la tabla 5 veces, una para cada carta de la baraja. Cada entrada de la tabla debe existir en la tabla USR_COL.

- **NUM_SERIE**: Clave foránea referenciando NUM_SERIE en la tabla COLECCION para las instancias de las cartas.
- **USUARIO**: Clave foránea referenciando NICK en la tabla USUARIOS para indicar el propietario de la carta.

2.6. Lógica del juego

En éste capítulo explicaremos como hemos aplicado la mecánica de una partida desde que se inicia hasta que acaba, es decir, como hemos planteado e implementado las reglas del juego.

Nuestra aplicación tiene diferentes modos de juego donde las reglas varían en cada uno, para no extendernos en la práctica únicamente explicaremos el modo de juego Normal. Así pues, el primer paso es listar las reglas:

- Hay dos jugadores, Rojo y Azul con el marcador a 0.
- Se juega sobre un tablero de 9 casillas, una matriz 3x3, donde en cada casilla se ubicará una carta.
- Aleatoriamente sea asignará el primer turno a uno los jugadores.
- Cada jugador jugará una carta de su mano en su turno en el tablero identificada con su color.
- Cada uno cuenta con 5 Cartas, con 4 atributos numéricos [0-9].
- Cada atributo numérico apunta a una casilla adyacente a casilla dónde se juega la carta (arriba, abajo, izquierda y derecha).
- Al jugar una carta en el tablero se comprobará si hay cartas con el color del oponente en las casillas adyacentes.

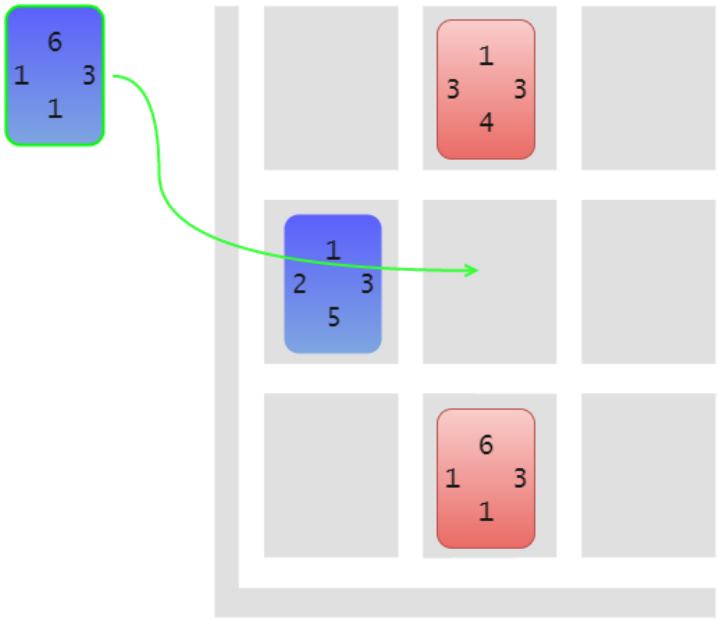


Figura 19. Jugando una carta

- En caso que la carta adyacente sea del color oponente, se comparará el atributo numérico que apunta a la casilla de la carta del color oponente con el atributo numérico de la carta oponente que apunta a la casilla de la carta jugada.

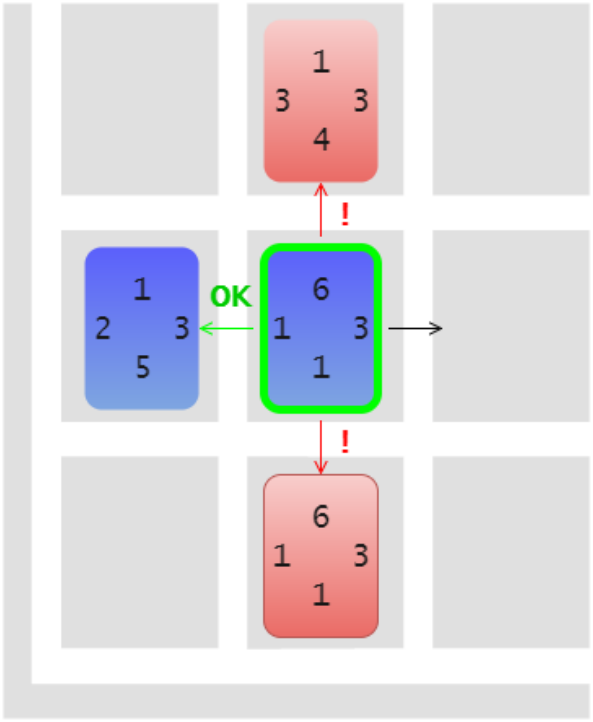


Figura 20. Verificando amenazas

- Si el valor del atributo de la carta jugada es superior al valor de la carta oponente, ésta se convertirá al color del jugador del turno actual e incrementará en 1 el marcador.

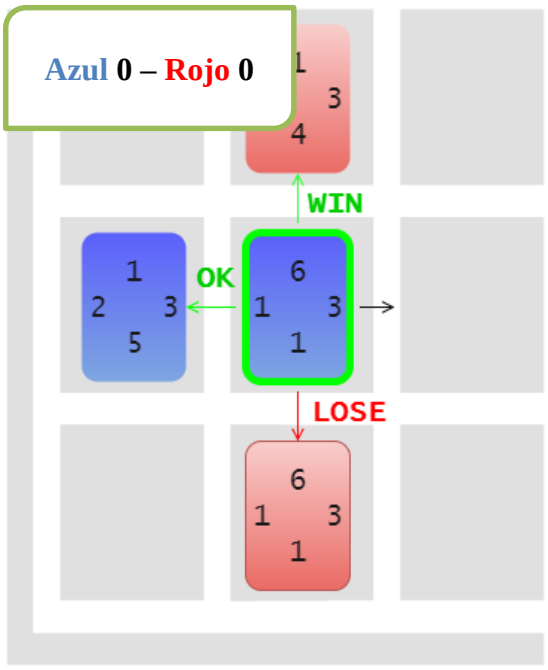


Figura 21. Convirtiendo carta

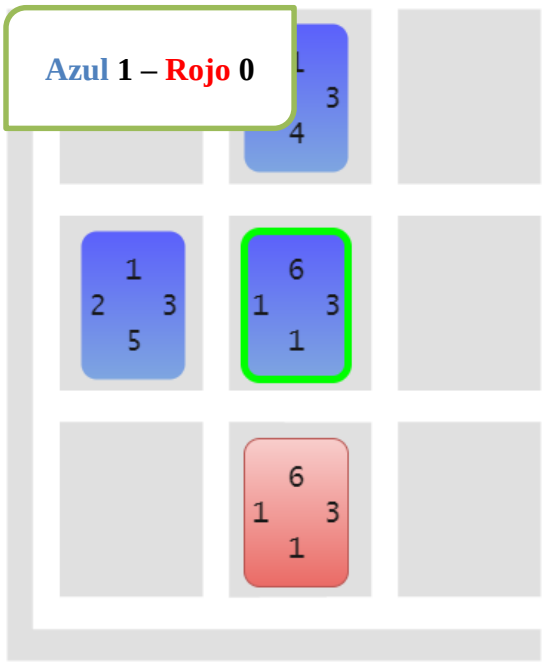


Figura 22. Incrementando el marcador

- El turno acaba cuando el jugador juega su carta en una posición vacía.
- La partida acaba cuando todas las posiciones del tablero están ocupadas.
- Gana el jugador cuyo marcador es superior.

Sabiendo por fin las reglas, la mejor manera de implementar el juego es plantear la partida como una sucesión de estados. Un jugador en su turno tiene varias posibilidades o jugadas depende de la que juegue le llevará a un estado u otro.

Por tanto, sabemos que las partidas tienen un estado inicial, un estado final y acciones que nos harán ir de estado en estado. Es imprescindible capturar que parámetros son los necesarios para la sucesión de estado.

Para nuestro sistema tenemos:

- **Carta**: Array de 5 enteros.
  - Carta [0]: para el atributo que apunta a la casilla izquierda, valores [0-9].
  - Carta [1]: para el atributo que apunta a la casilla superior, valores [0-9].
  - Carta [2]: para el atributo que apunta a la casilla derecha, valores [0-9].
  - Carta [3]: para el atributo que apunta a la casilla inferior, valores [0-9].
  - Carta [4]: para el color actual de la carta (0 para el rojo, 1 para el azul).
- **Baraja**: Array de 2 arrays de 5 cartas (Baraja [0] para el array de 5 cartas del jugador rojo y Baraja [1] para el azul).
- **Tablero**: Array de 3 arrays de 3 Cartas.
- **Turno**: Entero (0 para el rojo, 1 para el azul),
- **Marcador**: Array de 2 enteros (Marcador [0] para el rojo, Marcador [1]).

En el estado inicial los parámetros quedarán de la siguiente manera:

- [1]**Baraja** [0] = [[x,x,x,x,0], [x,x,x,x,0], [x,x,x,x,0], [x,x,x,x,0], [x,x,x,x,0]]
- [2]**Baraja** [1] = [[y,y,y,y,1], [y,y,y,y,1], [y,y,y,y,1], [y,y,y,y,1], [y,y,y,y,1]]
- **Tablero** = [[, ,] , [, ,] , [, ,]]
- **Marcador** = [0,0]
- **Turno** = Aleatorio entre 0 y 1

Ahora que hemos designado un inicio y un fin para el algoritmo podemos implementar, en pseudocódigo, el hilo de ejecución del proceso:

```
WHILE (!Tablero = vacio):
       Carta = Baraja[Turno][x];
// x toma cualquier valor entre 0 y longitud de Baraja[Turno]-1
       Tablero[i][j] = Carta;
// i, j toman valores [0-2], corresponden a una posición del tablero vacía
       FOR it = 1 to -1:
              IF (i+it < 3 AND i+it > -1):
// Miramos en las casillas horizontales
                     IF (!Tablero[i+it][j] = vacio AND Tablero[i+it][j][4] !=
Turno):
// Si la carta hay una carta de diferente color
                           IF (Tablero[i][j][1+it] > Tablero[i+it][j][1-it]):
// Si el atributo enfrentado es superior al del oponente
                                  Marcador[Turno]++;
                                         // Incrementa el marcador
                                  Tablero[i+it][j][4] = Turno;
                                         // Cambia de color la carta
                           END IF;
                     END IF;
              END IF;
       END FOR;
       FOR it = 1 to -1:
              IF (j+it < 3 AND j+it > -1):
// Miramos en las casillas verticales
                     IF (!Tablero[i][j+it] = vacio AND Tablero[i][j+it][4] !=
Turno):
// Si la carta hay una carta de diferente color
                    IF (Tablero[i][j][2+it] > Tablero[i][j+it][2-it]):
// Si el atributo enfrentado es superior al del oponente
                                  Marcador[Turno]++;
                                         // Incrementa el marcador
                                  Tablero[i][j+it][4] = Turno;
                                         // Cambia de color la carta
                           END IF;
                     END IF;
              END IF;
       END FOR;
       DELETE Baraja[Turno][x];
// Eliminamos la Carta de la mano, la longitud Baraja será igual a la actual
-1
       Turno = Abs(Turno - 1);
// Cambiamos el turno al otro jugador
END WHILE;
```

Tabla 28. Algoritmo en pseudocódigo de una partida en modo normal

[1] x puede tomar un valor u otro depende de las cartas seleccionadas. La condición final para llegar al estado final es que todo el tablero esté ocupado de cartas.
[2] y puede tomar un valor u otro depende de las cartas seleccionadas. La condición final para llegar al estado final es que todo el tablero esté ocupado de cartas.

2.7. Conexión Multijugador

Durante una partida se aplican las reglas del juego transfiriendo el turno entre los jugadores y en ausencia de uno se ejecuta un algoritmo (CPU). En una partida multijugador la comunicación que se establece toma un papel muy importante y en todo momento permanece abierta a respuestas.

a) Server Side

Los Sockets mantienen un canal de comunicación abierto por TCP entre el Cliente y el Servidor. En nuestra práctica usaremos la API de WebSockets introducida en Java EE7 y es en el contenedor de la aplicación donde crearemos la clase PvpWebSocket para establecer la comunicación.

```
@ServerEndpoint("/PVP")
public class PvpWebSocket {
   ...
}
```

Tabla 29. Implementación de la clase PvpWebSocket

Con @ServerEndpoint podemos indicar el sufijo referencial para el acceso a la conexión haciendo la petición HTTP. Necesitamos almacenar todas las sesiones abiertas entre el servidor y cada uno de los clientes.

```
Map<String, Session> Clients = new HashMap<String, Session>();
```

Tabla 30. Mapa de los clientes conectados

La clave almacena el identificador de sesión mientras que el valor la sesión en sí.
Además, necesitamos cada una de las salas donde se alojarán los jugadores durante la partida.

```
Map<Integer, String[]> Rooms = new HashMap<Integer, String[]>();
```

Tabla 31. Mapa de las salas para dos jugadores

El entero indica el número de la sala mientras que el array de Strings se guardaran los dos identificadores de sesión.

```
@OnOpen
    public void onOpen(Session session){
          Clients.put(session.getId(), session);
     ...
```

Tabla 32. Implementación del método abstracto onOpen

Con @OnOpen recibiremos solicitudes para la creación de nuevas sesiones y las añadiremos al mapa Clients. Respecto al mapa Rooms añadiremos el identificador de

sesión a la room siempre que el valor (array String) no tenga longitud > 1. Así pues, teniendo constancia de los usuarios dentro de la sala podemos establecer la comunicación entre ambos.

```
@OnMessage
public void onMessage(String message, Session session){
    ...
     session.getBasicRemote().sendText(message);
}
@OnClose
public void onClose(Session session){
    ...
}
```

Tabla 33. Implementación de los métodos abstractos onMessage y onClose

A través de @OnMessage podemos interceptar los mensajes que el usuario envía y redirigirlos por ejemplo al otro usuario de la sala. El primer argumento indica el mensaje enviado y el segundo la sesión del usuario que la envía. También podemos cortar la comunicación si recibimos una llamada @OnClose, en el código del método eliminaríamos al usuario del mapa de clientes y por tanto de la sala.

b) Client Side

En el lado cliente usaremos la API de WebSockets de la interfaz de JavaScript y disponible para HTML5. Al entrar en una Sala concreta iniciaremos crearemos un Web Socket tal que:

```
var Socket = new WebSocket(url, [protocol] );
```

Tabla 34. Conexión con el servidor a través de un Socket en el cliente

El primer argumento es la URI para conectar con la instancia del Socket en el contenedor, por tanto, la url acabará en "/PVP" tal y como viene indicada en el @ServerEndpoint de la clase PvpWebSocket.

El atributo readyState del Socket nos permite ver el estado actual de la conexión i además podemos captar varios eventos que ocurran en la conexión:

- onopen: cuando se fija la conexión.
- onmessage: cuando el cliente recibe un mensaje del servidor.
- onerror: cuando hay errores de comunicación.
- onclose: cuando la conexión se finaliza.

Los métodos disponibles desde el Socket son send(data), para enviar datos al Socket, y close(), para finalizar la comunicación.

Al inicio de una partida dos jugadores entraran en una sala, se disparará para cada uno el evento onopen. Deberán confirmar que ambos quieren jugar por lo que la partida no

empezará si los dos no envían el send(data) para validar el comienzo, si deciden salir de la sala ejecutaran un close().

En todo momento el Socket conoce a los dos usuarios así que cuando uno juegue carta lo que ocurrirá es que se pasará la instancia del objeto del estado del juego a JSON String y se enviará como mensaje, en ese momento el usuario que ha jugado debe esperar a que el oponente juegue por lo que no tiene control de su interfaz. El Socket al recibir el mensaje lo reenviará al otro usuario de la sala, se disparará el evento onmessage en el lado del cliente, actualizará el estado del juego y tomará control de la aplicación para jugar.

Este flujo pasará sucesivamente hasta que la partida finalice y para entonces ambos usuarios interceptaran un evento onclose.

c) Diagrama de secuencia (PVP)

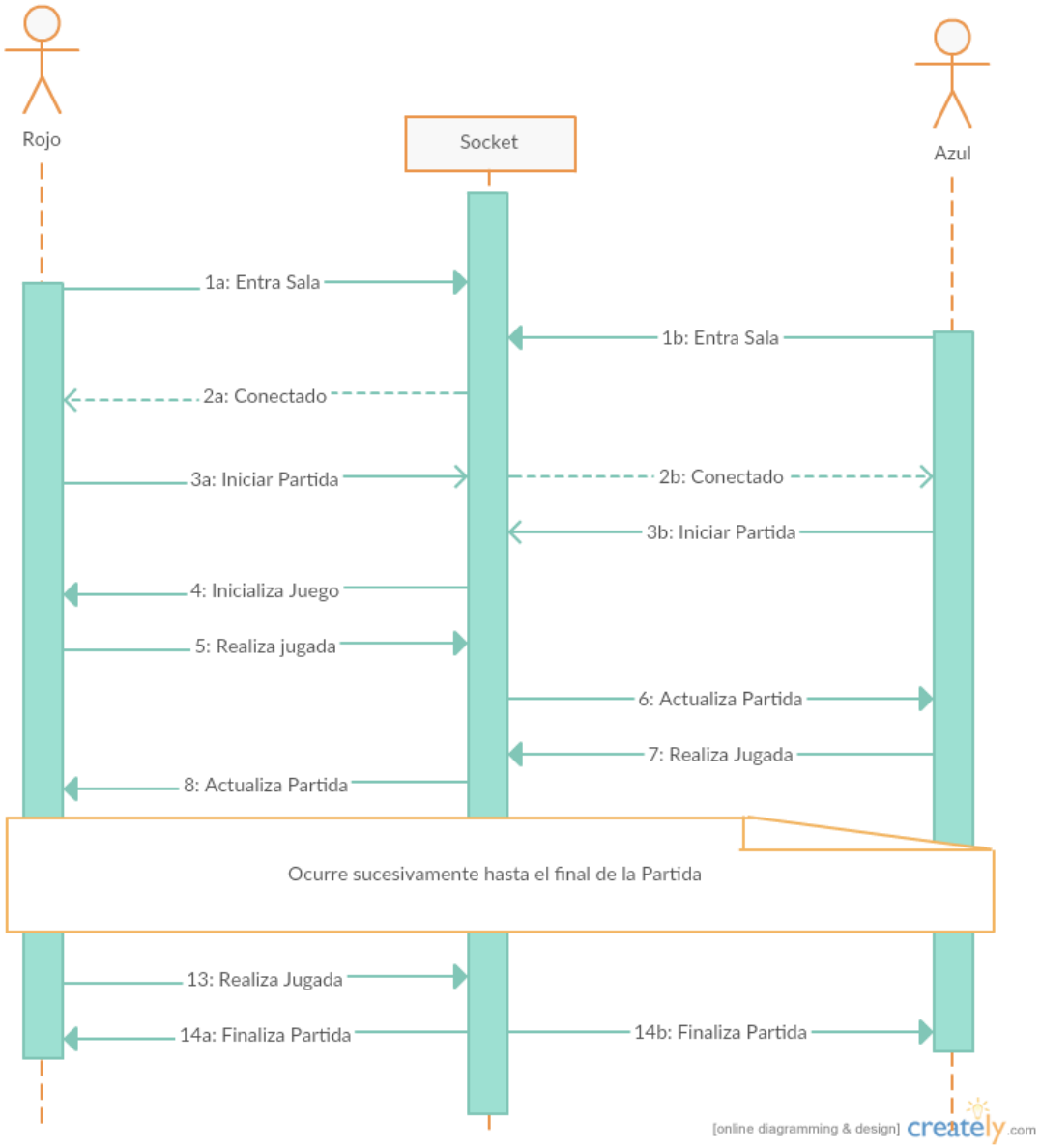


Figura 23. Diagrama de secuencia de una partida multijugador

3. TECNOLOGÍAS EMPLEADAS

3.1. El paradigma del desarrollo de aplicaciones

En el momento de desarrollar aplicaciones móviles es muy importante definir las herramientas y tecnologías para la construcción y uno de los paradigmas más frecuentes es el del desarrollo nativo o híbrido. Por eso antes que nada es conveniente discutir las ventajas/desventajas de cada vertiente:

a) Desarrollo nativo

+ El hecho de usar un framework nativo permite sacar todo el partido de una plataforma.
+ Es una manera óptima de aprovechar los recursos de la plataforma desde bajo nivel sobre todo cuando se consumen en gran cantidad.
+ Se puede sacar mejor rendimiento de los diferentes componentes del sistema (bluetooth, GPS, pantalla, etc…) además de obtener actualizaciones para éstos al actualizar el sistema.
+ Cada SDK está montado para un lenguaje (Java para Android y Objective C y Swift para iOS) en específico según la plataforma y con una interfaz ya dedicada.
+ En el proceso de marketing y distribución son más fáciles de integrar en un Marketplace.
- El software no es tan reutilizable, por la simple razón de que cada plataforma tiene un lenguaje.
- En algunas ocasiones es necesario un coste para poder desarrollar para una plataforma.
- Según la plataforma se ha de invertir más o menos tiempo en dominar un lenguaje en específico por lo que la curva de aprendizaje puede ser poco pronunciada a corto plazo.

b) Desarrollo híbrido

+ La multicompatibilidad es la principal ventaja. Podemos generar un paquete de la aplicación y ejecutarlo en cualquier plataforma.
+ La mayoría de frameworks disponibles son de código abierto al abasto de todos y el soporte que ofrecen es muy extenso.
+ Están construidos con HTML5, CSS3 y Javascript tecnologías particularmente enfocadas a web apps.
+ Es accesible para cualquier dispositivo, de hecho, permite acceder a recursos del sistema.
+ Generalmente se usan lenguajes que usan a diario los desarrolladores web.
+ El código al ser multiplataforma siempre será reutilizable, un aspecto importante en la ingeniería del software.
+ En caso de desarrollar una versión web de una app, sería fácilmente escalable a una resolución mayor con tan solo modificar las hojas de estilo.
- La interacción es más lenta (por las respuestas del sistema) y limitado respecto un framework nativo.

- Una app híbrida no se promociona con tanta facilidad respecto una nativa, por el tema de los Marketplaces.

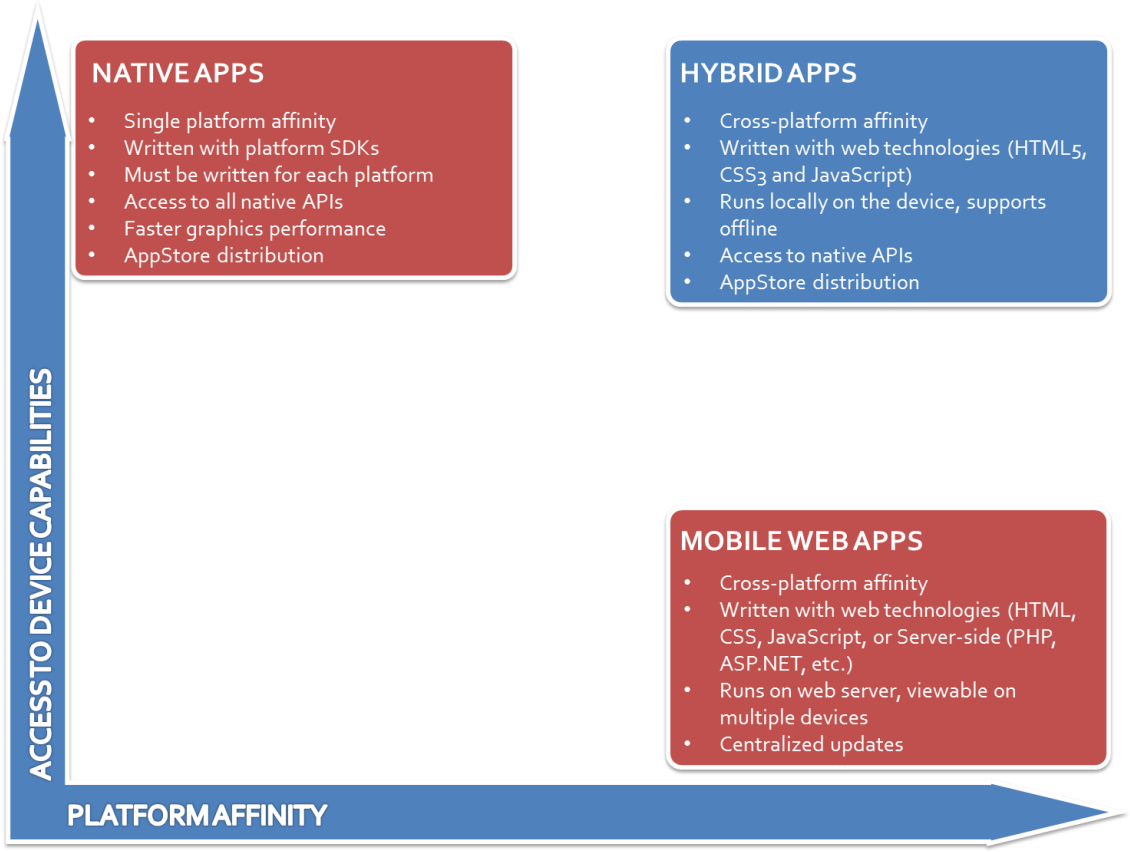


Figura 24. Relación Capacidad/Afinidad de los diferentes tipos de desarrollo [34]

3.2. Tiempo de desarrollo y curva de aprendizaje

El hecho de desarrollar una aplicación nativa implica una dedicación adicional para dominar un lenguaje, librerías, frameworks y técnicas para una plataforma en concreto y en cierto modo dota al desarrollador de cierta especialidad.

Normalmente para aplicaciones nativas los proveedores facilitan un IDE dedicado que garantice el confort del developer sin embargo el asunto se complica más si queremos desplegar para diferentes plataformas.

Con aplicaciones híbridas podemos adquirir muy buenos resultados en relación a los de una nativa y además en términos de tiempo y aprendizaje HTML5, CSS3 y Javascript son tecnologías imprescindibles si eres web developer, y por si fuera poco el acceso a funciones nativas (geologalización, notificaciones, etc…) es fácil y accesible a través de plugins.

Feature or Point	Web App	Hybrid App	Native App
Single Effort	☑☑	☑	☐
Server Side Updates	☑☑	☑	☐
Feels Like an App	☑☑	☑☑	☑☑
Deploy to App Store	☐	☑	☑
Microphone	☑	☑☑	☑☑
Speaker	☑☑	☑☑	☑☑
Camera Image/Video	☑☑	☑☑	☑☑
GPS	☑☑	☑☑	☑☑
Accelerometer	☑☑	☑☑	☑☑
Local Storage	☑☑	☑☑	☑☑
App Store Approvals Required	☐	☑	☑
Access to Contacts	☐	☑	☑
Inter App Communications	☑	☑	☑☑
Full OS API Access	☐	☐	☑☑
Notifications	☐	☑	☑☑
Local File System	☐	☑	☑☑
Network Connection	☑	☑☑	☑

Figura 25. Accesibilidad a recursos según el tipo de desarrollo [35]

3.3. Frameworks de aplicaciones móviles

a) PhoneGap/Apache Cordova

De acuerdo con todos los aspectos discutidos anteriormente la opción que más se adapta para nuestra aplicación es hacerla híbrida y PhoneGap/Cordova será el framework que usaremos para desarrollarla.

Solemos escuchar PhoneGap y Cordova como herramientas por separado pero que se usan conjuntamente como framework para aplicaciones híbridas. En 2009 la empresa Nitobi creó PhoneGap para el desarrollar aplicaciones mobiles con HTML5, CSS3 y JavaScript para acceso a recursos nativos. Más adelante la empresa fue absorbida por la fundación Apache para impulsar y ofrecer el código como Open Source dando lugar a PhoneGap/Apache Cordova. Así pues, PhoneGap no es más que una distribución de Apache Cordova y la compañía decidió extender la herramienta con nuevas actualizaciones.

PhoneGap está implementado sobre Apache Cordova, incluyendo comandos y herramientas adicionales que extienden la funcionalidad de Cordova. PhoneGap, en sí, puede hacer todo lo que hace Cordova y adicionalmente se encarga de la compilación y construcción remota de los proyectos pudiendo construir los proyectos en la nube pudiendo ahorrarnos un SDK local para cada una de las principales plataformas móviles (iOS, Android, Windows Phone, etc…).

Algunos de los comandos que ofrece PhoneGap exclusivos son:

```
$ phonegap remote login
$ phonegap remote build ios
$ phonegap remote install android
```

```
$ phonegap remote run ios
$ phonegap remote logout
```

Tabla 35. Comandos adicionales de PhoneGap

Nos hemos decantado por esta opción porqué ya tenemos cierta experiencia con la tecnología y habiendo librerías como JQuery, JQuery Mobile podemos manejar el workflow de eventos y dejar un aspecto visible de la aplicación de lo más profesional ahorrándonos código de una manera más simplificada. Podríamos optar por JavaScript puro, pero tenemos a nuestra disposición alternativas más sencillas.

El aspecto que podemos obtener no deja de ser el de un sitio web, pero contando con la cantidad de soporte y recursos abiertos de los frameworks web que usaremos el acabado final será lo más cercano a una aplicación móvil.

Somos conscientes de alternativas existentes cuyo rendimiento es posiblemente superior, dado que usando HTML5, CSS3 y JS el rendimiento de un lenguaje interpretado no es el mismo que el de uno compilado.

Recordamos que el comportamiento del contenedor actúa de manera parecida al de un Web Browser ejecutando lenguaje interpretado. De todos modos, Mexme no es una aplicación pesada y en nuestro diseño intentaremos compensar esa carencia en cuanto al rendimiento.

Toda la infraestructura que compone la aplicación móvil se puede hacer a través de los comandos de PhoneGap/Cordova. Teniéndolo ya instalado y con los permisos del root, ejecutamos lo siguiente:

```
$ cordova create hello com.example.hello HelloWorld
```

Tabla 36. Comando de creación de una aplicación con PhoneGap/Cordova

Nos generará un esqueleto de directorios donde la vista de inicio será *index.html* en el directorio www. PhoneGap/Cordova actúa como contenedor nativo de nuestra aplicación y en diferentes directorios guardaremos nuestro código JavaScript, hojas de estilo y además del HTML ya mencionado.

Otro paso importante a tener en cuenta a la hora de desplegar nuestra aplicación es que tipo de paquete vamos a generar para instalar en nuestro dispositivo. Previamente dentro del directorio de la aplicación por línea de comandos ejecutamos lo siguiente:

```
$ cordova platform add [nombre de la plataforma]
```

Tabla 37. Comando para añadir plataformas al proyecto

Con esto nuestra aplicación podrá ser desplegada generando una apk para plataformas Android si lanzamos cualquiera de las siguientes comandas:

```
$ cordova build [nombre de la plataforma]
```

Tabla 38. Comando para compilar el proyecto

Para desplegar para una plataforma concreta (debe haber sido añadida) o bien:

```
$ cordova run [nombre de la plataforma]
```

Tabla 39. Comando para compilar y ejecutar el proyecto

…Además del despliegue, ejecutará la aplicación en un emulador o un dispositivo configurado en modo developer.

Por lo que respecta los plugins, para acceder a las APIs de los recursos de un dispositivo, es posible instalarlos ejecutando:

```
$ cordova plugin add [nombre del plugin]
```

Tabla 40. Comando para añadir plugins

b) Alternativas de desarrollo

Además de PhoneGap/Apache Cordova hay otras alternativas para el desarrollo y construcción de aplicaciones híbridas:

AppCelerator Titanium

Con Appcelerator Titanium podemos crear aplicaciones para diferentes plataformas móviles (iOS, Android…) e incluso de escritorio a través de JavaScript y su IDE basado en eclipse. Permite editar ficheros PHP, Python y Ruby y convertirlos a JavaScript a través de otros frameworks.

Titanium también al igual que PhoneGap nos permite desplegar aplicaciones multiplataforma como si de aplicaciones nativas se trataran, consiguiendo buenos resultados a nivel de rendimiento, aspecto y controles nativos. Sin embargo, a nivel de soporte y documentación dejan mucho que desear por no hablar de su IDE poco consistente.

Xamaris

Con Xamaris podemos desarrollar aplicaciones híbridas con un nivel de rendimiento nativo para varias plataformas (iOS, Android, Windows Phone…) con acceso total a recursos del sistema.

Su licencia de uso es de pago, pero sin llegar a los altos precios de frameworks de desarrollo nativo puro y al contrario que Titanium la comunidad da un buen soporte.

También ofrece una colección de librerías reutilizables, sin embargo puede haber librerías

muy concretas que necesitemos y no se puedan incluir por tener un listado de opciones de compilación muy limitado.

Exclusivamente, permite el uso de código nativo para cubrir las carencias nativas donde el código fuente de Xamaris no puede llegar. Hacer uso de Xamaris requiere tener un diseño muy claro además de un dominio especializado del framework para poder aprovechar el código para cualquier Sistema Operativo.

3.4. Aplicación Cliente

a) JavaScript, HTML5 y CSS3

JavaScript, HTML5 y CSS3 son tecnologías imprescindibles para cualquier web developer y en nuestro caso particular tenemos cierta familiaridad lo que nos alivia cierta carga de aprendizaje.

JavaScript es el lenguaje con el que podremos modificar el contenido web dinámicamente pudiendo acceder a elementos el documento para implementar su comportamiento. Gracias a que el documento es tratado como un objeto DOM (Data Object Model) es mucho más accesible para JavaScript.

Con HTML5 (HyperText Markup Language) es posible estructurar el documento a través de tags clausulados para la disposición de los elementos. Cada uno de los tags cuenta con sus respectivas propiedades modificables y preferentemente usaremos aquellos que gocen de cierta semántica y estén predefinidos por la simple razón de ahorrarnos tener que crear unos de nuevos y porque el contenedor web híbrido tendrá menos problemas para interpretarlo.

CSS3 (Cascading Style Sheet) nos permite editar y acceder a los elementos que hay en el HTML5 y de alguna manera modificar su diseño, apariencia y efectos.

Por otra parte, usaremos otras APIs de JavaScript como son la de WebStorage y WebSocket para almacenar datos de sesión del usuario y crear conexiones abiertas para establecer la comunicación entre jugadores en las salas para las partidas.

b) AJAX / JSON

AJAX (Asynchronous JavaScript And Xml) es una metodología que hace uso de una infraestructura de tecnologías como son el DOM, el objeto XMLHttpRequest de JavaScript para la conexión asíncrona, HTML/CSS y el formato de los datos solicitados del servidor en XML o JSON, aunque nosotros los solicitaremos en JSON. AJAX, así mismo, permite realizar peticiones http en segundo plano desde el JavaScript sin necesidad de cambiar todo el DOM pudiendo actualizar una parte de éste.

Gracias a JQuery la implementación del código al llamar a AJAX resulta mucho más sencilla e intuitiva.

```
$.ajax({
    data: {"parametro1" : "valor1", "parametro2" : "valor2"},
    type: "POST",
```

```
    dataType: "json",
    url: [url de la petición],
})
 .done(function(data) {
         console.log(JSON.stringify(data));
 });
```

Tabla 41. Ejemplo de llamada AJAX en JQuery

Podemos indicar el tipo de petición (Get, Post…), el tipo de datos esperados en la response (json, xml…), la url a la que hacemos la petición y los datos que enviamos junto con la petición. En nuestro caso el formato de los datos enviados/recibidos será en JSON.

Mediante la API de JSON para JavaScript podemos jugar con los métodos estáticos JSON.stringify y JSON.parseJSON para pasar de String a objeto Javascript (en formato JSON) y viceversa.

```
var obj = JSON.parseJSON('{"key": "value" }'); // obj = { "key":
"value" };
var str = JSON.parse({"key": "value"}); // str = '{"key": "value"}';
```

Tabla 42. Ejemplo de conversión de objeto JSON a String y de String a JSON

c) JQuery / JQuery Mobile

JQuery es una de las librerías de JavaScript que usaremos y nos van a ayudar a ahorrarnos trabajo y líneas de código en el Client Side. Se ocupa de todo tipo de tareas relacionadas con el tratado del HTML, captación de eventos y AJAX. Asimismo, también da soporte a CSS3 como selector de elementos, clases y estilos, todo un avance que hace que el contenido web en el lado cliente sea más dinámico.

```
// JQuery
$( "button.continue" ).html( "Next Step..." );

// JavaScript
var x = document.getElementsByTagName("button");
var i;
for (i = 0; i < x.length; i++) {
    if (x[i].className == "continue") x[i].innerHTML = "Next Step…";
}
```

Tabla 43. Comparativa código JavaScript y JQuery

En el fragmento de código anterior muestran de dos maneras distintas podemos implementar lo mismo y salta a la vista cómo JQuery puede simplificarnos el trabajo.

Una de las ventajas de JQuery es su compatibilidad, las funciones de la API pueden ser llamadas y ejecutadas sin tener de preocuparnos del tipo de navegador que haya por detrás, simplemente se hará compatible en función navegador empleado (siempre y cuando soporte JavaScript), aunque en nuestro caso no es un Browser Container lo que tenemos más bien es un contenedor híbrido.

Otra de las cosas que hacen rico a JQuery es toda la comunidad que hay detrás proporcionando plugins para la reutilización de código.

Por otro lado, JQuery Mobile es un framework basado en HTML5 para la creación de UI de sitios web y aplicaciones mobiles accesibles y que den soporte para cualquier plataforma en dispositivos táctiles (Smartphone, Tablet o derivados…) o de escritorio.

Figura 26. UI implementada con JQuery Mobile para tablet y smartphone [33]

JQuery Mobile encapsula JQuery siguiendo el modelo RWD (Responsive Web Design) en el que se plantean soluciones de adaptabilidad debido a las diferentes prestaciones que ofrecen los dispositivos como son la resolución, la densidad de pantalla y modos de interacción para un mismo código fuente y ponen al abasto del programador diferentes widgets para la construcción de la interfaz.

Los tres principios fundamentales en los que se basa son los siguientes:

- Con los Selectores de estilos y clases de JQuery se gestionan características de pantalla a los dispositivos (amplitud, resolución, etc…).
- Los elementos y widgets empleados se ubican y ajustan en función del elemento contenedor.
- Al igual que los widgets y elementos gozan de flexibilidad según el contenedor, los elementos multimedia también serán redimensionables.

Otra ventaja que ofrece JQuery mobile es que no está sujeto a una temática o estilo, ofrece un esqueleto sobre el que se pueden customizar diferentes posibilidades temáticas y multiplexarlas sin necesidad de reconstruir todo el código.

A través de aplicaciones web como ThemeRoller podemos diseñar nuestros temas que mejoran el aspecto visual de los widgets.

d) Alternativas de desarrollo

Angular JS

En primer lugar, debemos añadir que JQuery y Angular JS son frameworks con diferentes enfoques y abastos. Mientras que con JQuery podemos manejar y editar con total solvencia los elementos del DOM, Angular JS va un paso más allá.

Con Angular JS también podemos acceder a elementos de documento y editarlos a través de JQLite sin llegar al nivel de JQuery (por sus niveles de acceso y compatibilidad). Sin embargo, permite extender la funcionalidad del HTML y dicta una serie de pautas basadas en el MVC (dedicadas al uso de Factorías, Servicios y Controladores).

Si quisiésemos explotar al máximo JavaScript en un sistema más complejo, Angular sería la alternativa más apropiada, pero en nuestro caso daremos un uso moderado del JavaScript usando JQuery para la interactividad y la carga de datos.

Por tanto, Angular JS sería un framework muy útil, pero se quedaría grande para lo que queremos desarrollar en consideración a la curva de aprendizaje.

Backbone

Es un framework MVC, mucho más reducido respecto a Angular JS. Podríamos obtener buenos resultados, pero no iríamos más lejos de lo que podríamos hacer con JQuery y JavaScript. Un punto a favor es que podemos ganar más control sobre cualquier elemento pudiendo alterar el comportamiento y aunque dota al desarrollador de cierta versatilidad y libertad también le obliga a seguir un nivel de exigencia programando superior, donde la estructuración y refactorización del código trasciende en cuanto a su complejidad.

Dado que gozamos de cierta familiaridad con el JQuery resulta ineficiente invertir tiempo en asimilar la sintaxis y la semántica de Backbone pudiendo obtener los mismos resultados en una herramienta en la que ya tenemos experiencia sin necesidad de exprimir a la gota el código.

3.5. Aplicación Servidor y base de datos

a) Aplicación Web

En el servidor remoto estará ejecutándose un WAR (Web Application Archive) y escuchando las peticiones http que le lleguen a través de un puerto determinado. Los Servlets instanciados en el contenedor ampliarán la funcionalidad del servidor a través de la API Servlet y la API WebSockets.

Por tanto, crearemos Servlets extendiendo de la clase HttpServlet para los diferentes módulos funcionales que componen nuestro servicio, cada uno identificado de manera diferente para poder ser llamado.

Cuando queramos establecer conexiones que nos lleguen desde el lado de cliente las gestionaremos desde una clase que implemente los métodos (onOpen, onClose,

onMessage…) de la API de WebSockets. En el subcapítulo a) de la sección 2.7 explicamos con más detencion como lo hemos llevado a cabo.

La API MySQL Connector de Java será imprescindible si queremos invocar querys, inserciones y actualizaciones desde la aplicación además de recuperar los resultados de una selección.

Además de las APIs ya mencionadas y las Java Core libraries, que ya vienen importadas por defecto, también necesitaremos la API de JSON para Java. Los datos demandados por el cliente vienen encapsulados en formato JSON y a través de su API realizaremos la conversión previa al envío.

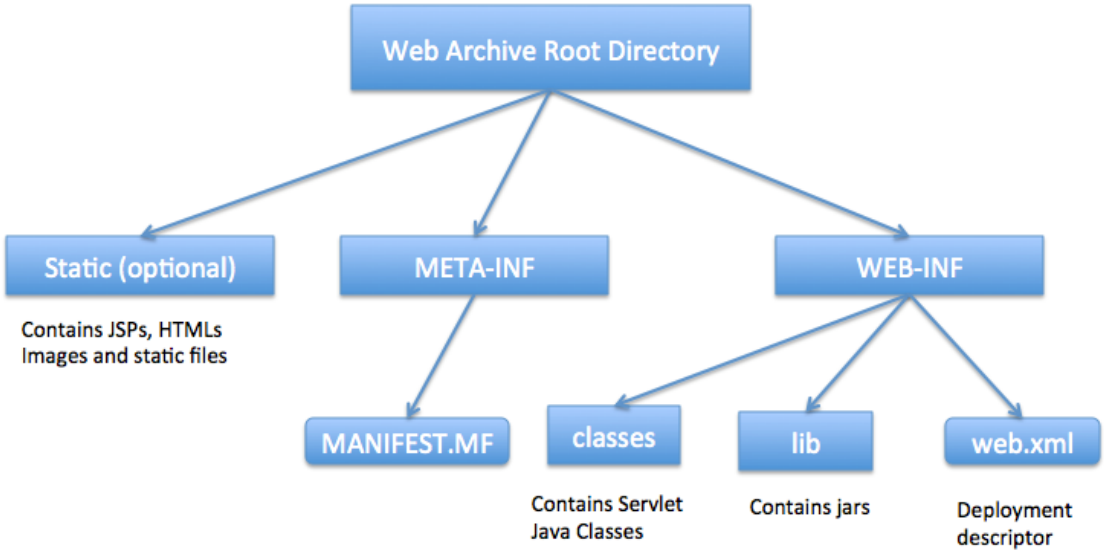


Figura 27. Esqueleto de directorios de un WAR [36]

Dado que desplegaremos WAR en el servidor, en Java, necesitamos uno que pueda contener Servlets y JSP (Java Server Pages), por esta razón usaremos Apache Tomcat, es uno de los servidores más populares por ser de código abierto, la versión más reciente hasta el momento es la 9.0 para desplegar aplicaciones y servicios web. Admite las APIs de WebSockets 1.1, Servlet 3.1 y JSP 2.3.

b) Base de datos

En el momento de realizar las operaciones con la base de datos haremos uso de SQL (Syntax Query Language) donde nuestro Schema de datos, privilegios de usuarios, usuarios, restricciones de las tablas y, si es necesario, PL/SQL (para invocar procesos con múltiples consultas para así poder evitar accesos correlativos y colapsar el sistema) estarán alojadas en un servidor SQL remoto.

4. WIREFRAMES

En el diseño de la UI (User Interface) es muy habitual hacer uso de los conocidos wireframes para representar en primera instancia el esqueleto de todas las pantallas de la aplicación a nivel conceptual y funcional.

En los wireframes se detallan los elementos de las pantallas, como el usuario interacciona con ellos, cuál es el comportamiento, cómo están dispuestos y cómo están comunicadas las pantallas, es decir la jerarquía de los contenidos. Es una práctica muy común en el prototipaje para aplicaciones de interacción persona-máquina.

4.1. Listado de pantallas

Anteriormente hemos descrito los casos de uso. Éstos representaban un flujo de interacciones que el usuario hacía con los elementos dispuestos en pantalla. El siguiente sumario lista de A hasta S todas las pantallas del sistema:

- Pantalla A: Login
- Pantalla B: Registro
- Pantalla C: Bienvenida
- Pantalla D: Listado de usuarios
- Pantalla E: Listado de amigos
- Pantalla F: Partida
- Pantalla G: Listado de Salas
- Pantalla H: Sala
- Pantalla I: Perfil de usuario
- Pantalla J: Mi Perfil
- Pantalla K: Perfil amigo
- Pantalla L: Editar Perfil usuario
- Pantalla M: Agregar amigo
- Pantalla N: Cambio de contraseña
- Pantalla O: Listado de invitaciones
- Pantalla P: Editar Baraja
- Pantalla Q: Selección oponente
- Pantalla R: Selección de carta
- Pantalla S: Selección de Avatar

4.2. Mockups de las pantallas

Una vez declaradas todas las pantallas del sistema es el momento de describir su diseño conceptual a través de wireframes:

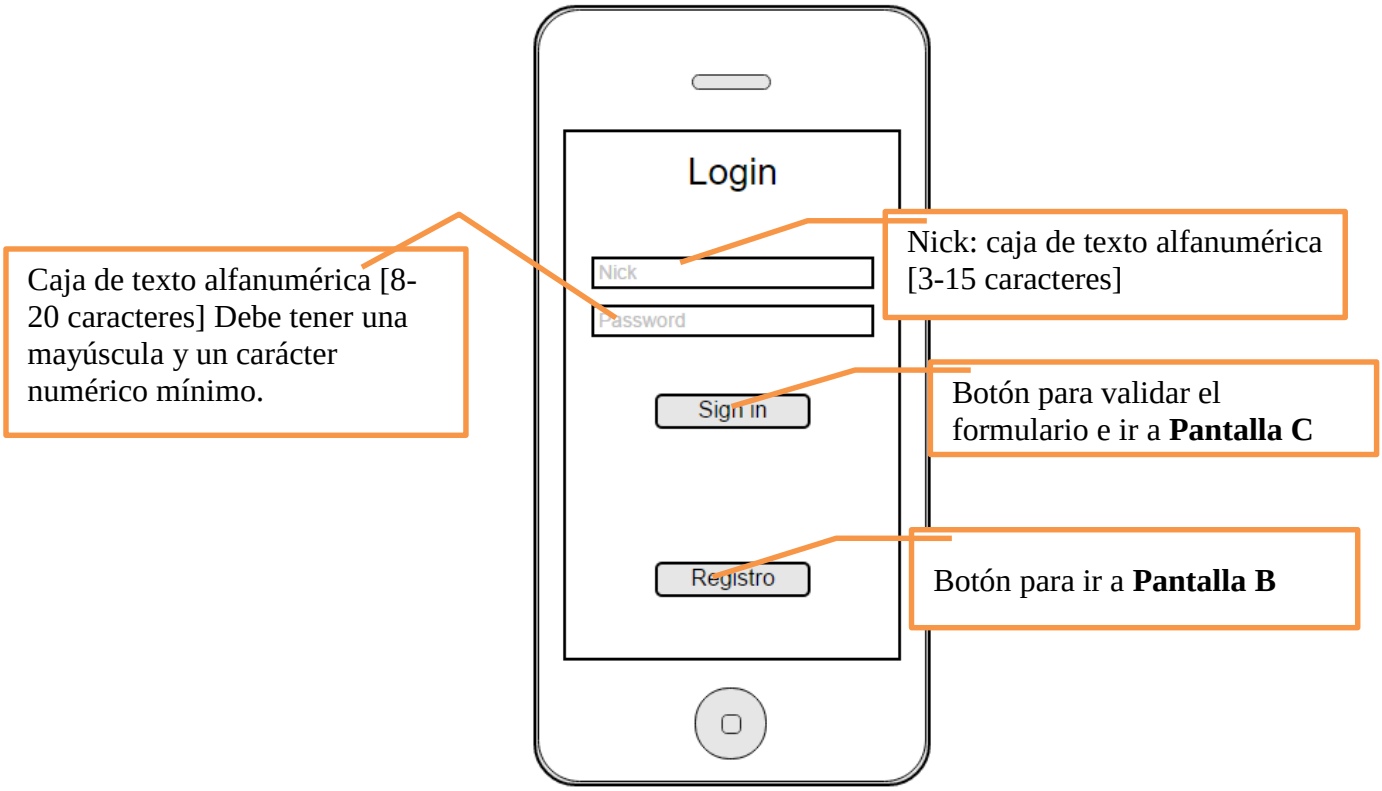


Figura 28. Pantalla A: Login

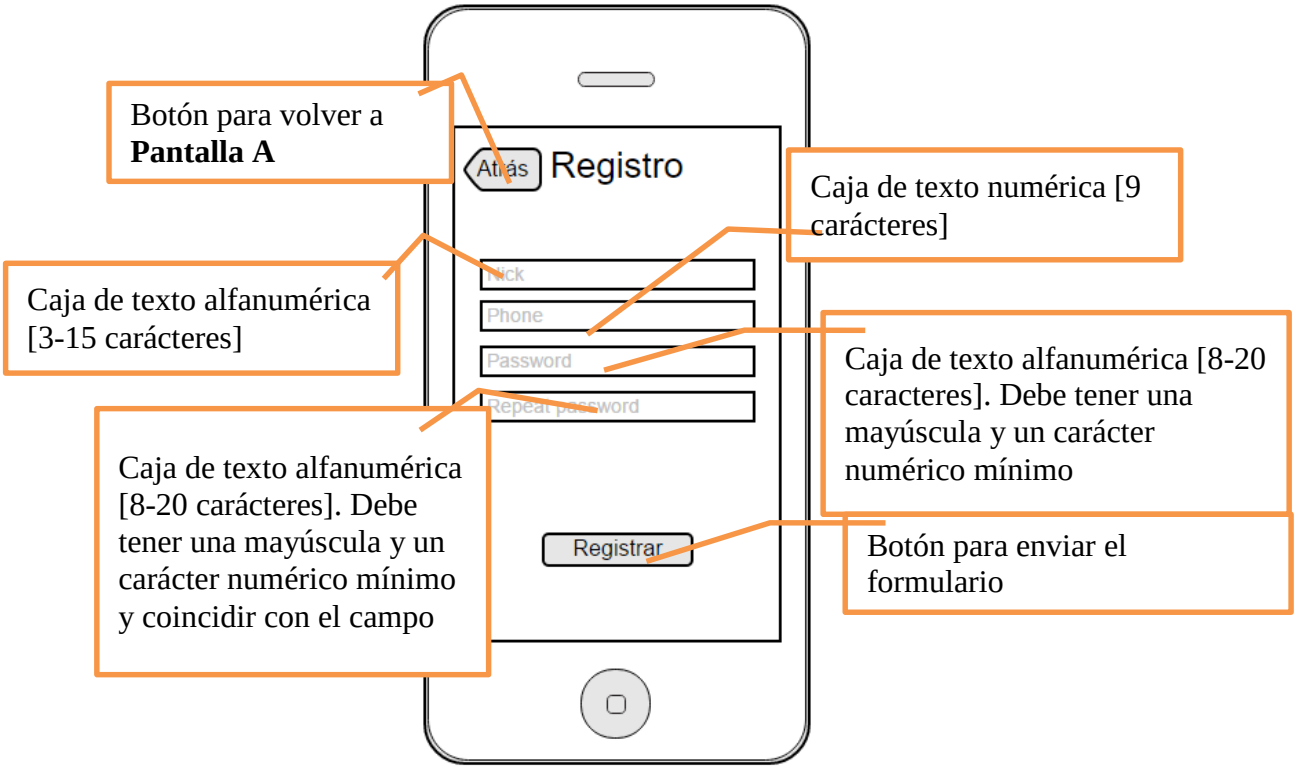


Figura 29. Pantalla B: Registro

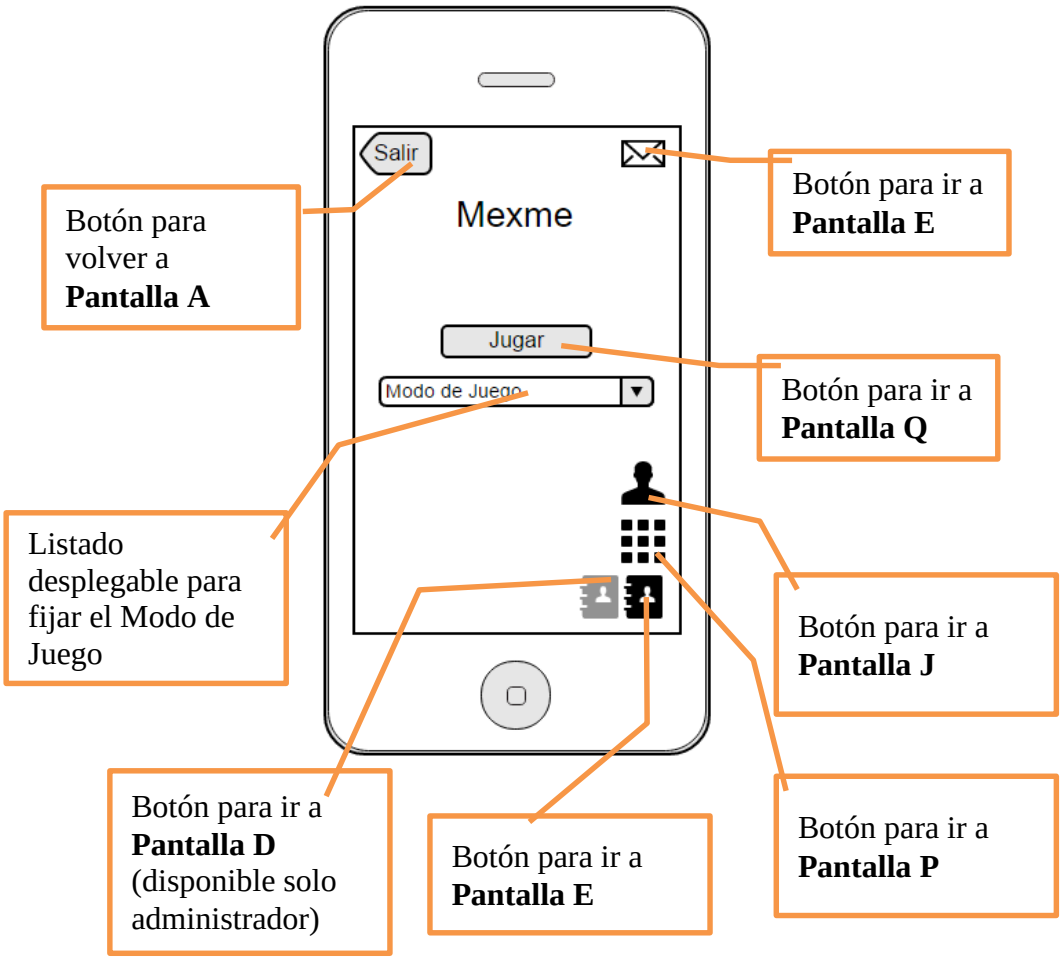


Figura 30. Pantalla C: Bienvenida

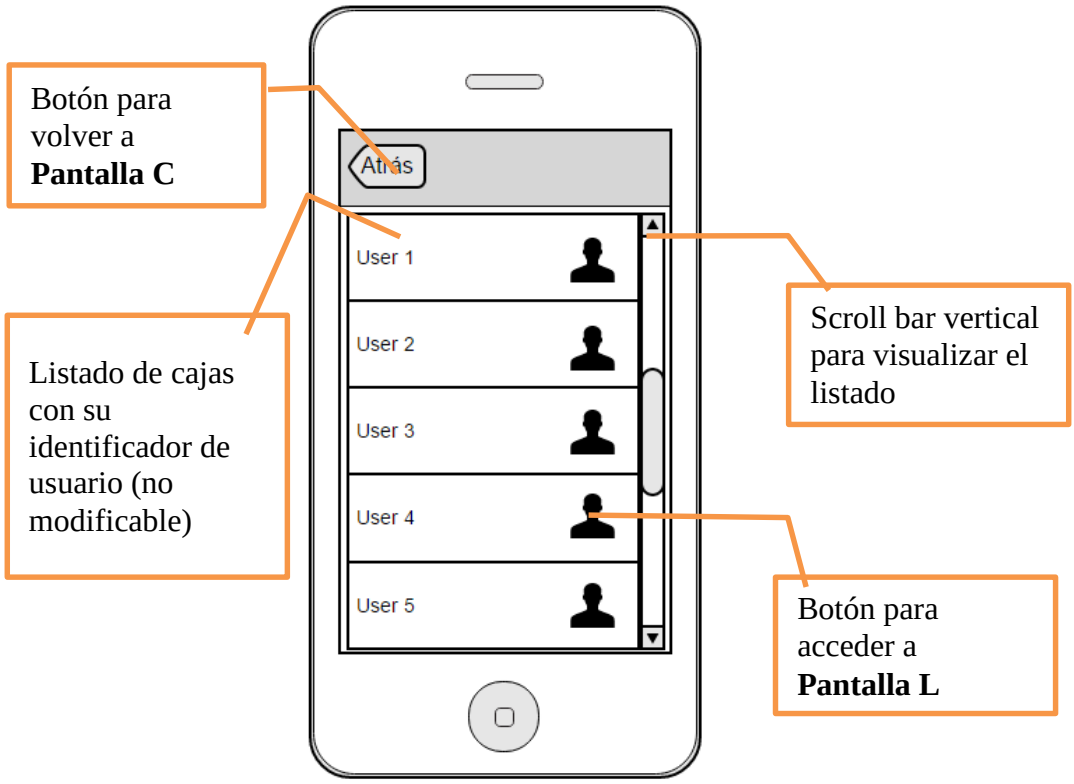


Figura 31. Pantalla D: Listado de usuarios

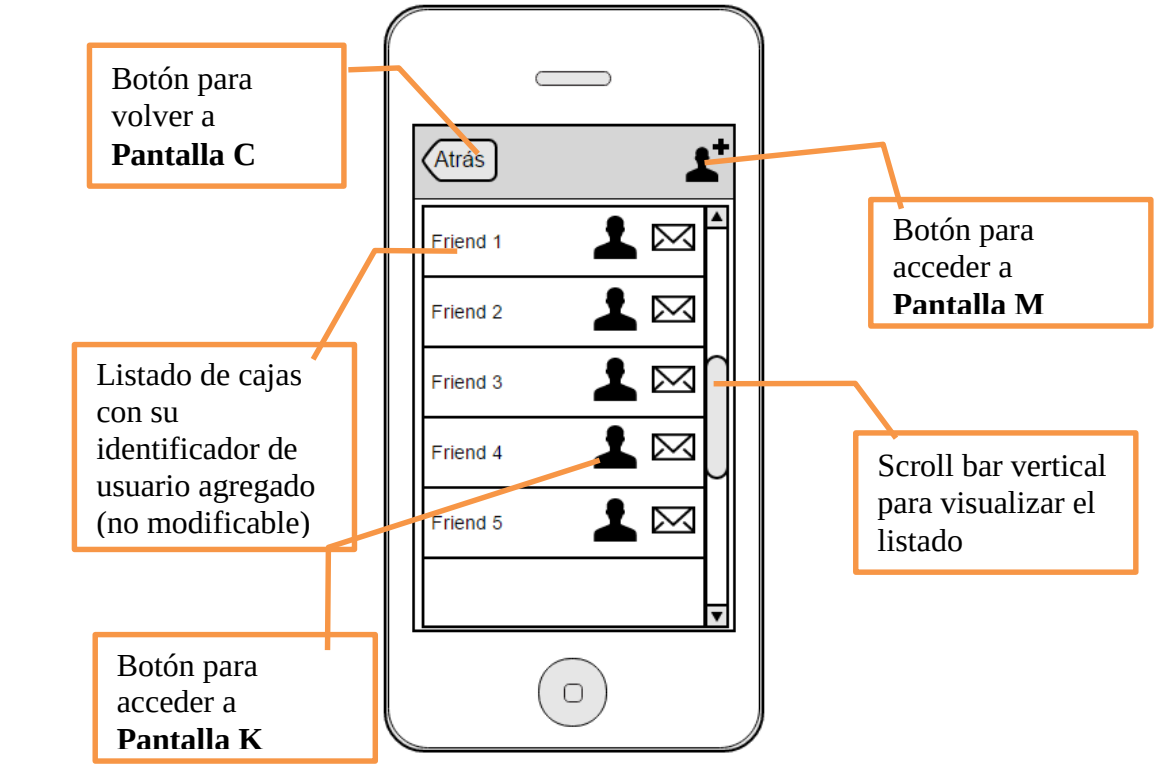


Figura 32. Pantalla E: Listado de amigos

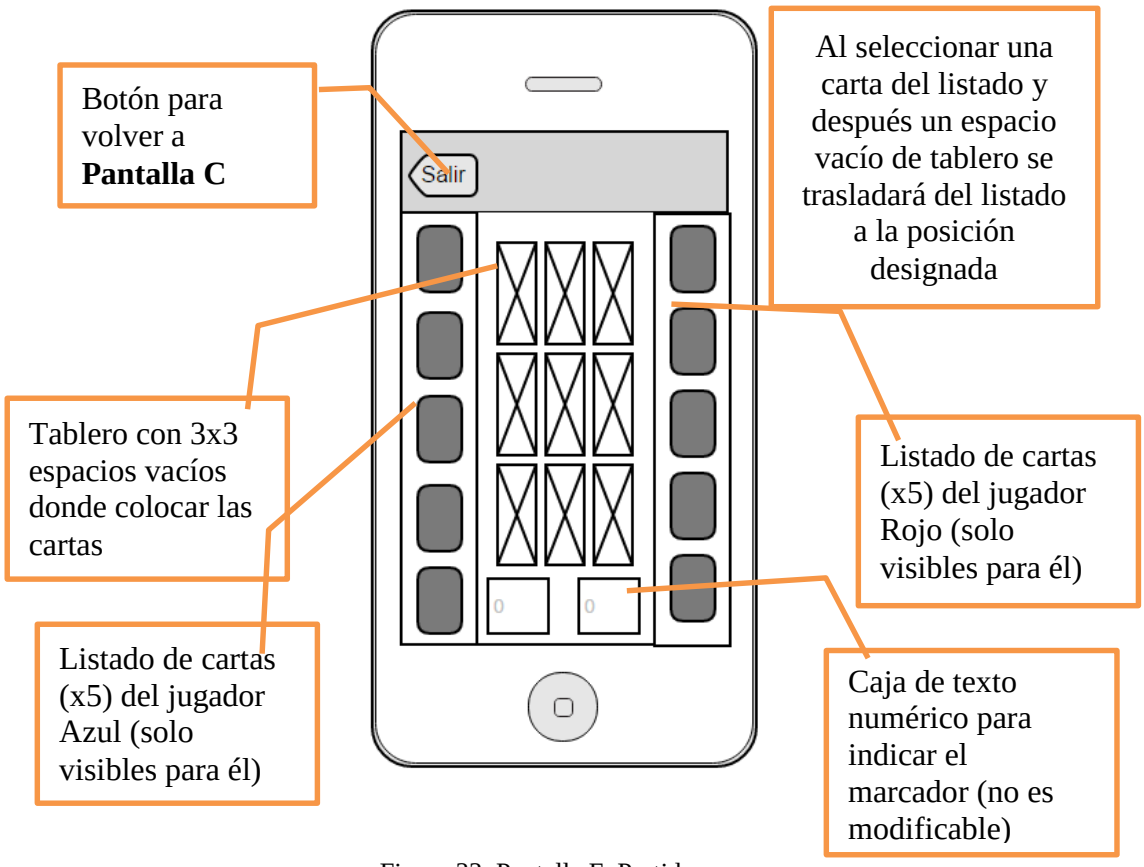


Figura 33. Pantalla F: Partida

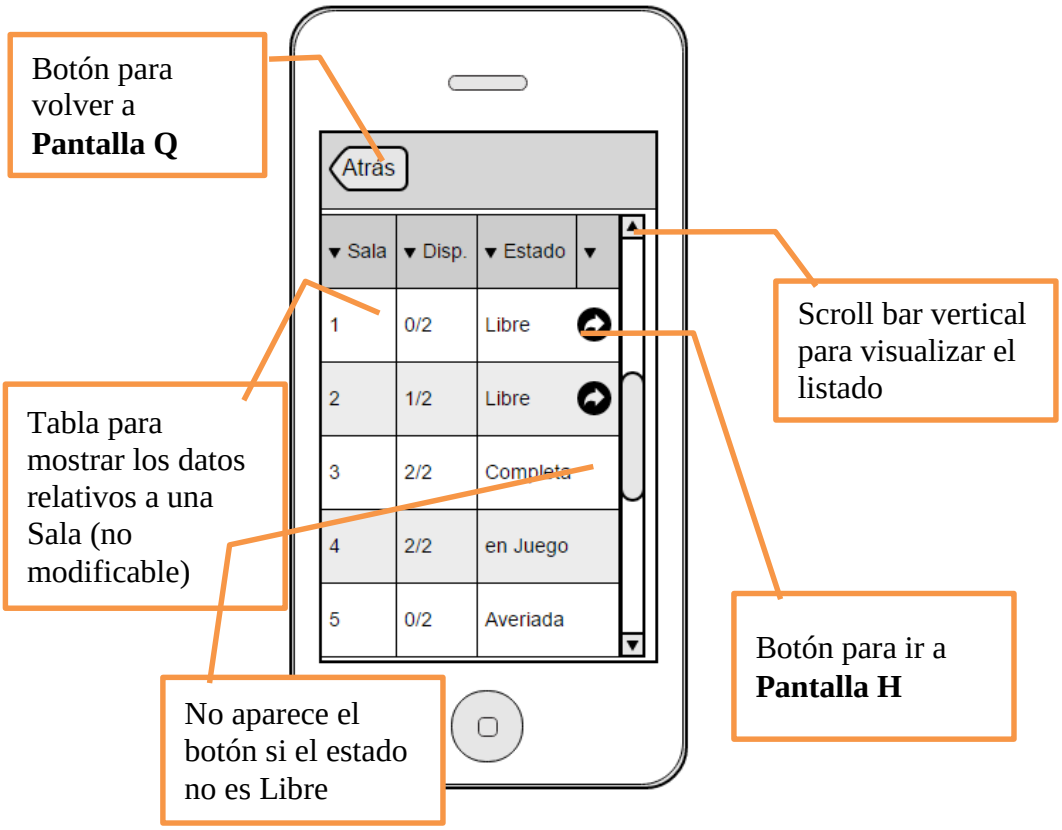


Figura 34. Pantalla G: Listado de salas

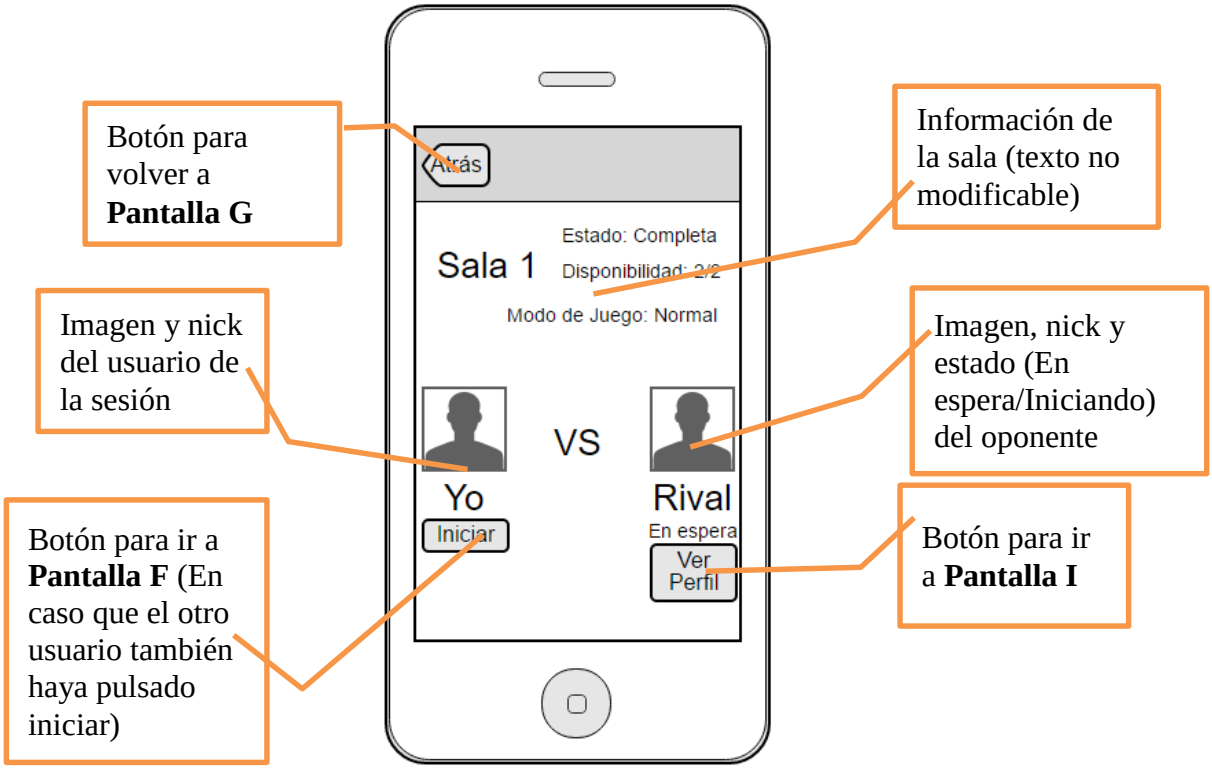


Figura 35. Pantalla H: Sala

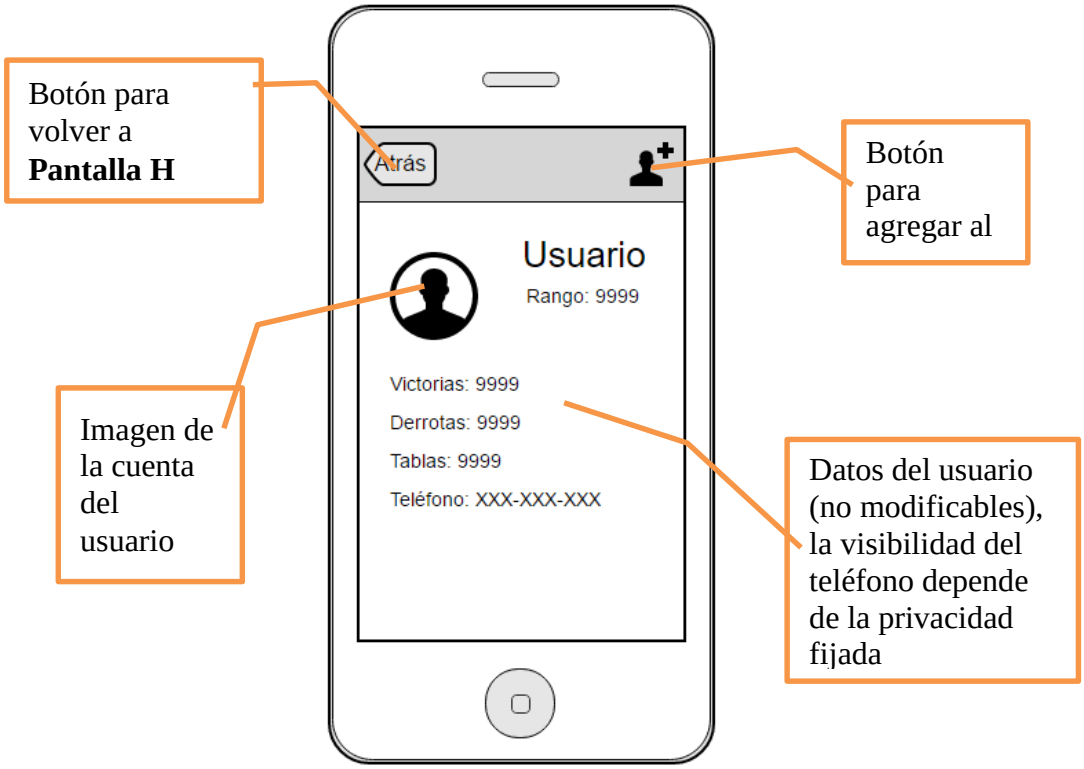


Figura 36. Pantalla I: Perfil de usuario

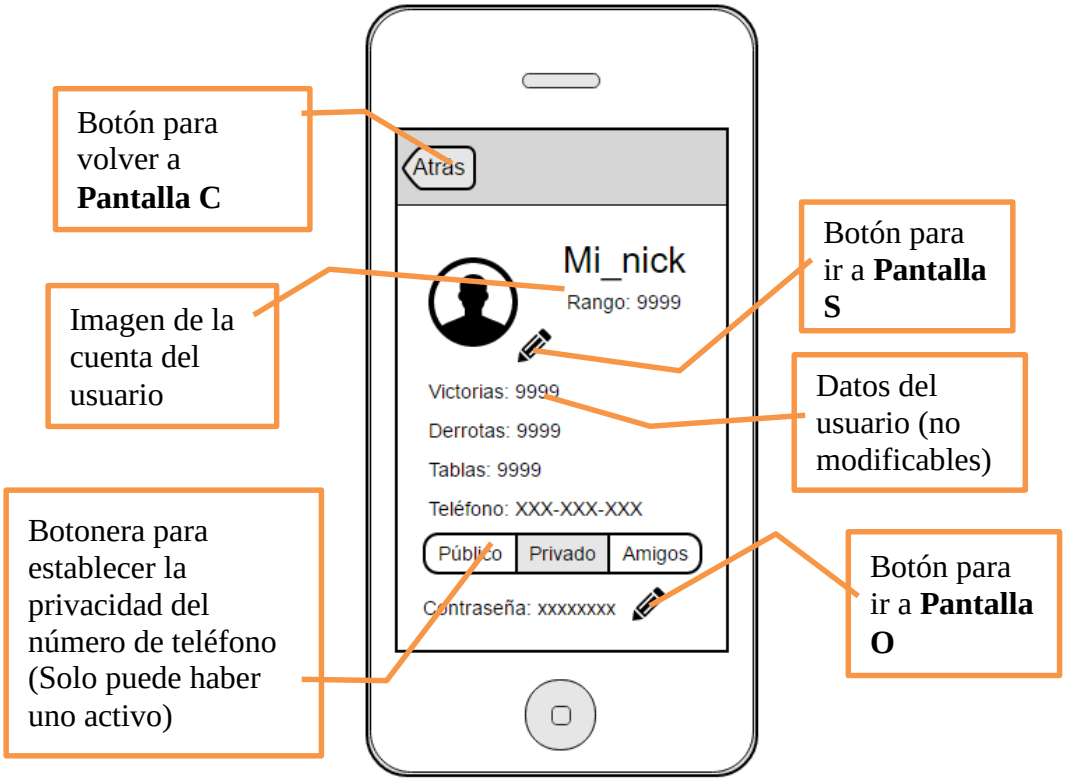


Figura 37. Pantalla J: Mi Perfil

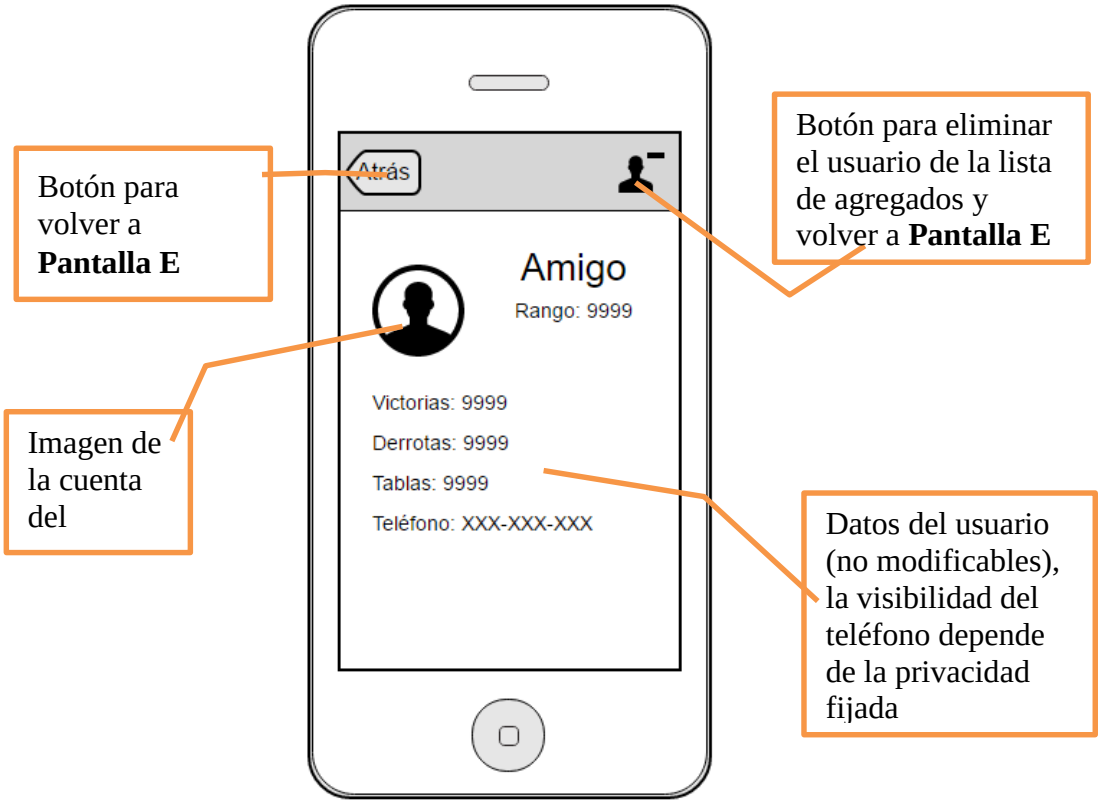


Figura 38. Pantalla K: Perfil amigo

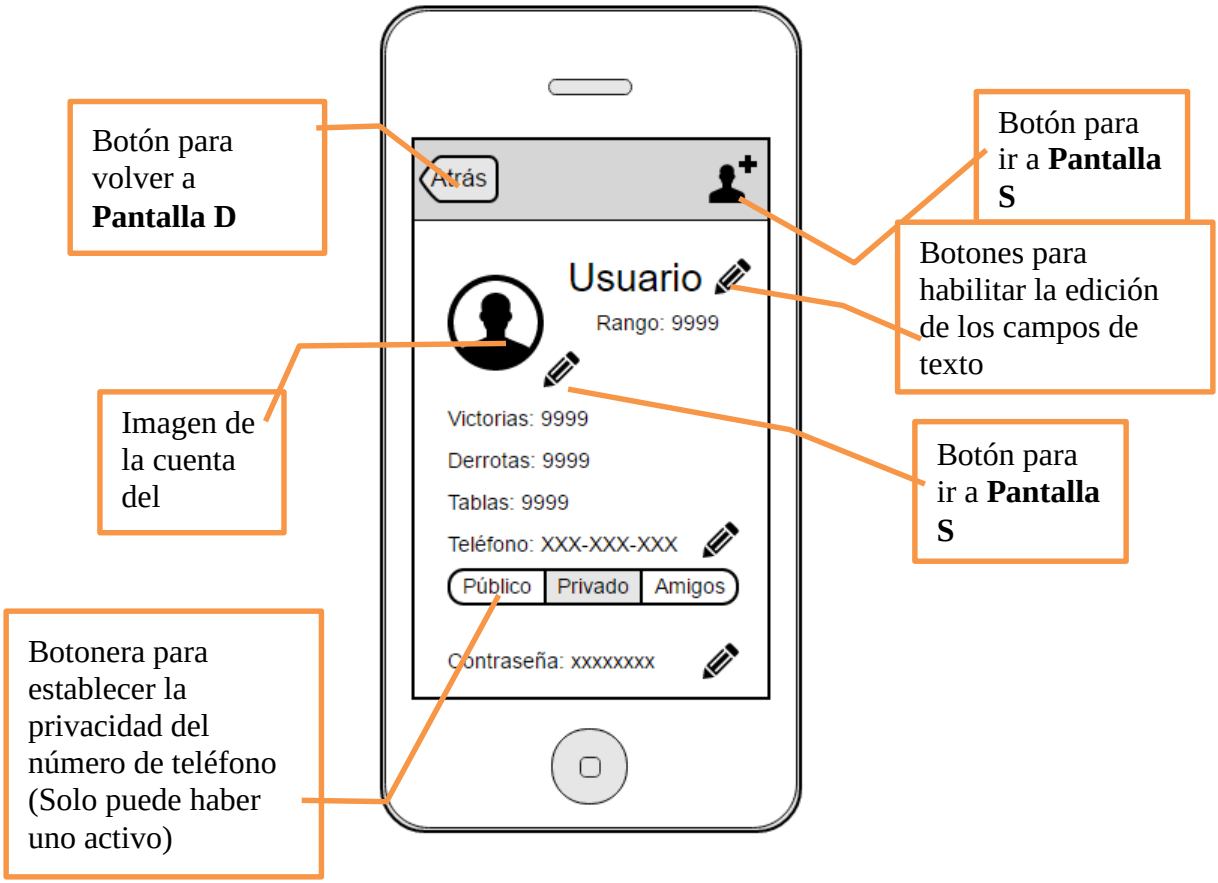


Figura 39. Pantalla L: Editar Perfil de Usuario

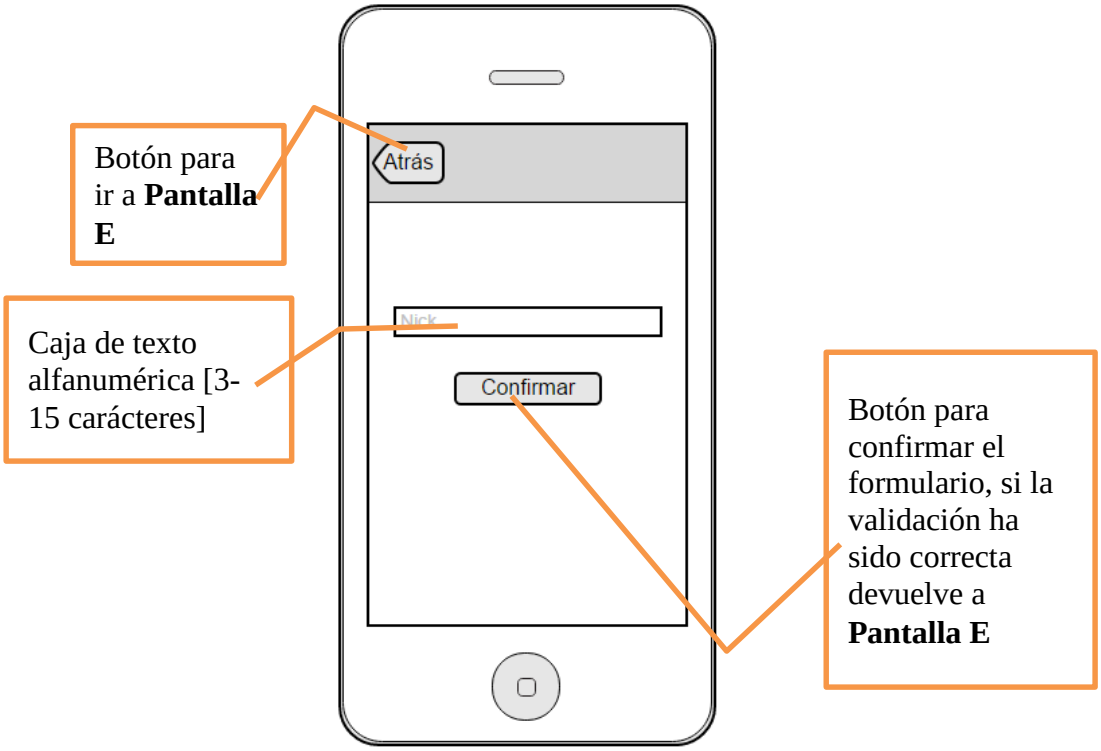


Figura 40. Pantalla M: Agregar Amigo

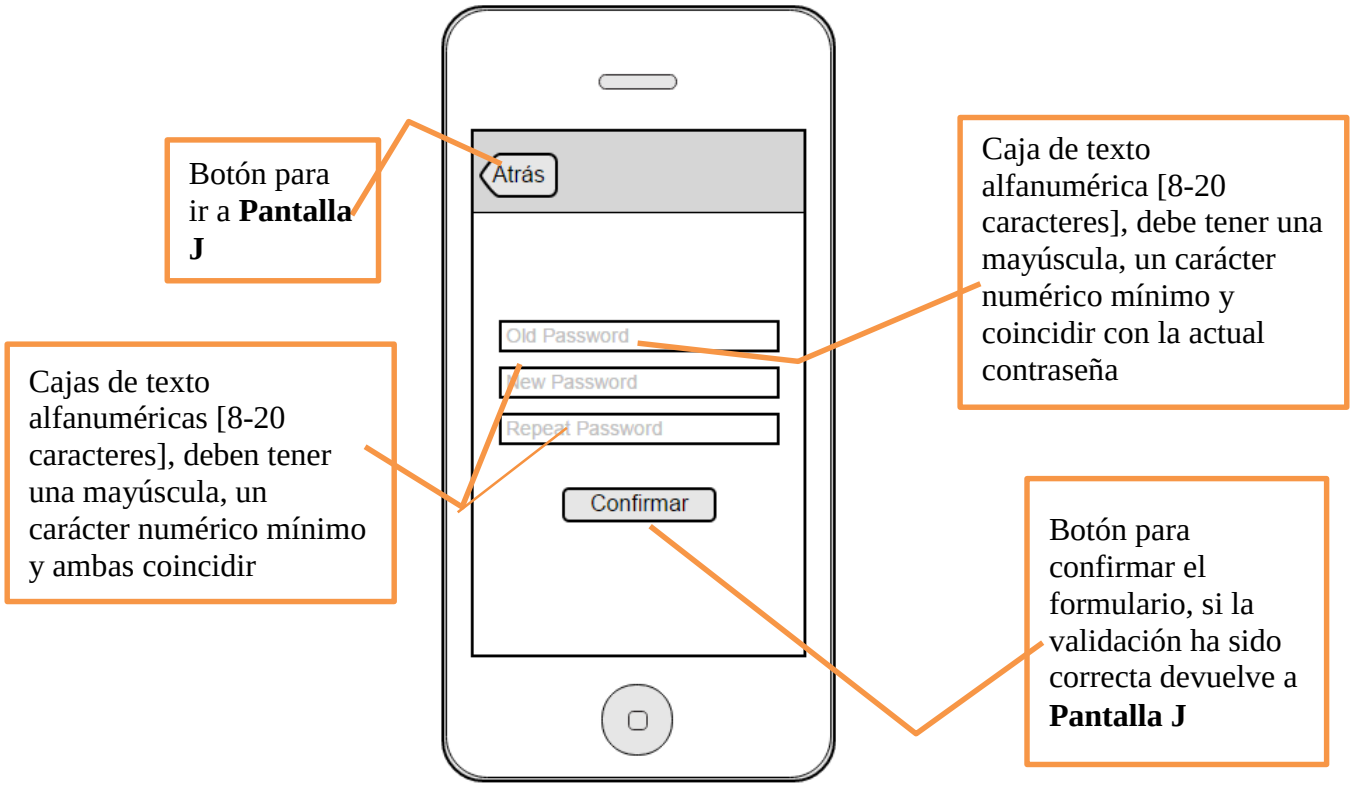


Figura 41. Pantalla N: Cambio de Contraseña

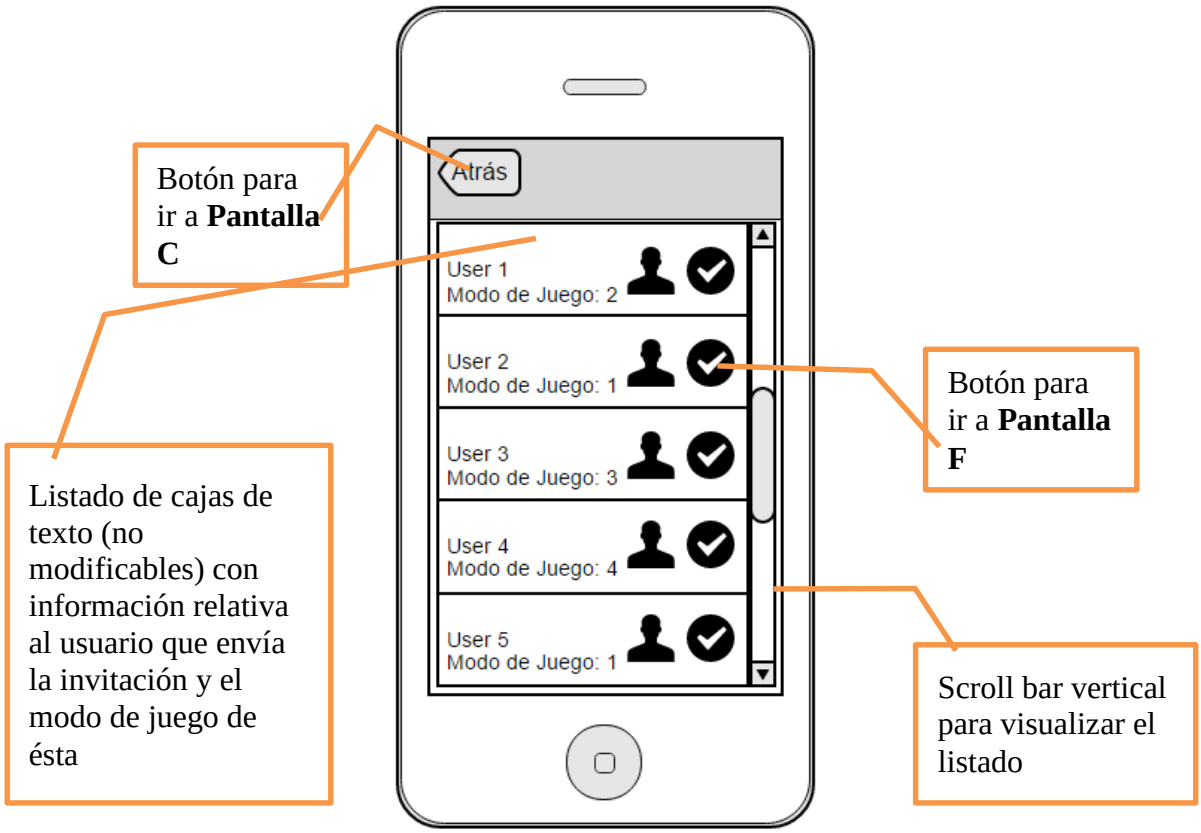


Figura 42. Pantalla O: Listado de Invitaciones

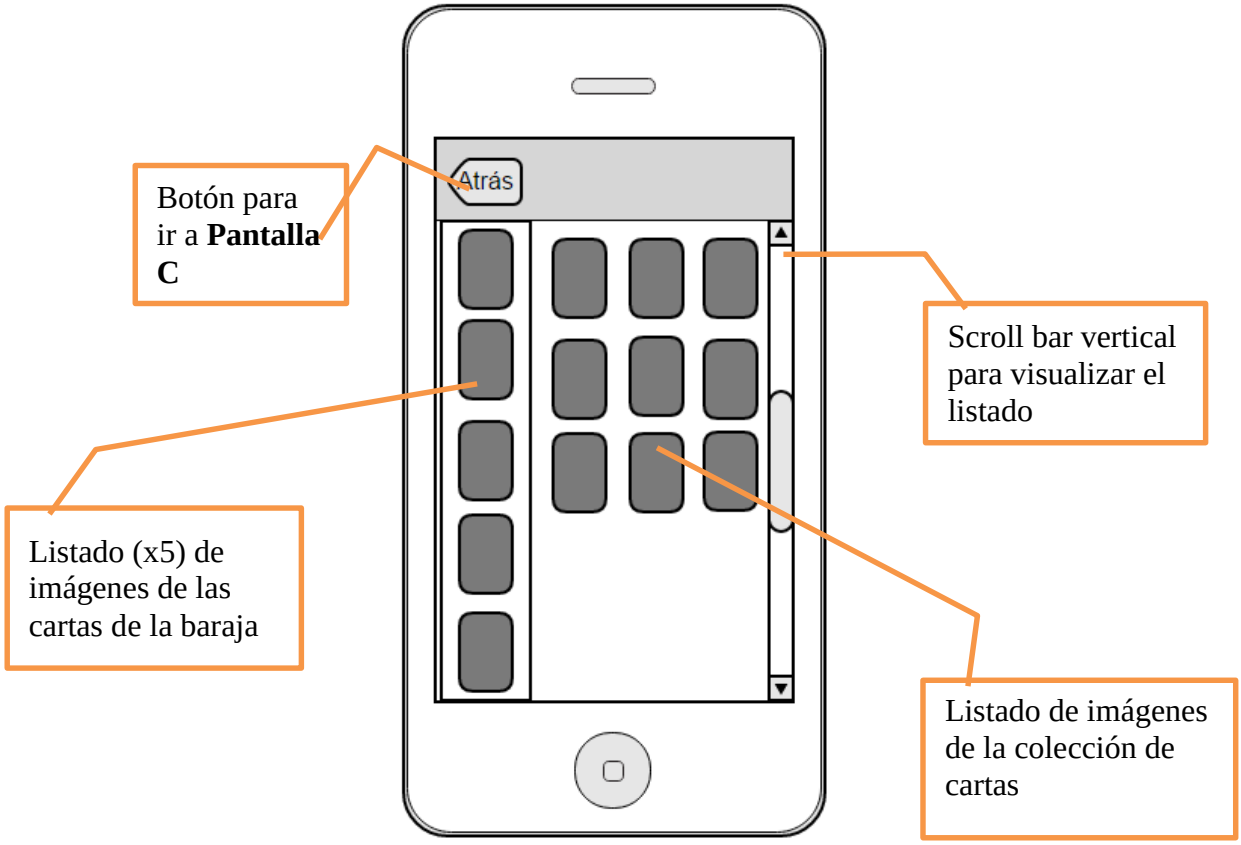


Figura 43. Pantalla P: Editar Baraja

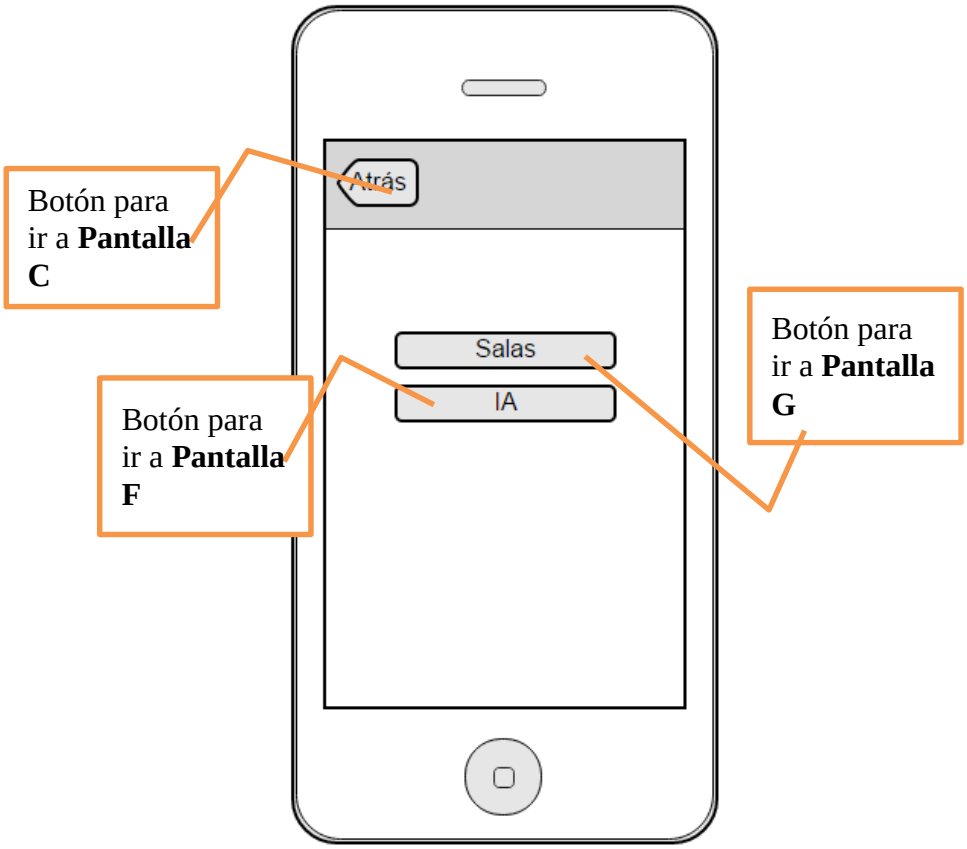


Figura 44. Pantalla Q: Seleccionar oponente

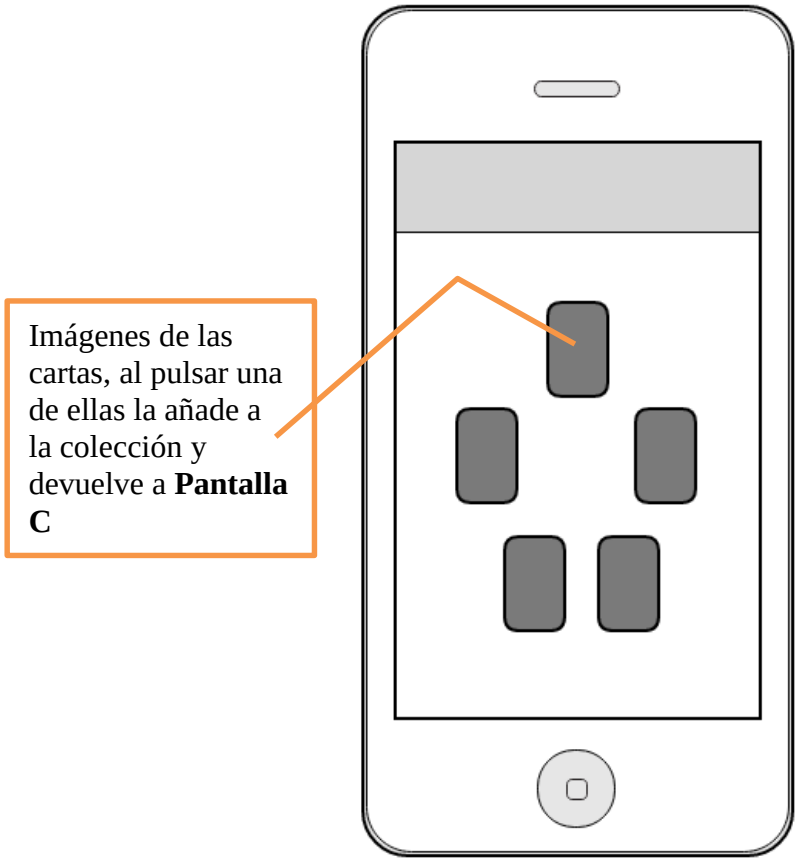


Figura 45. Pantalla R: Selección de Carta

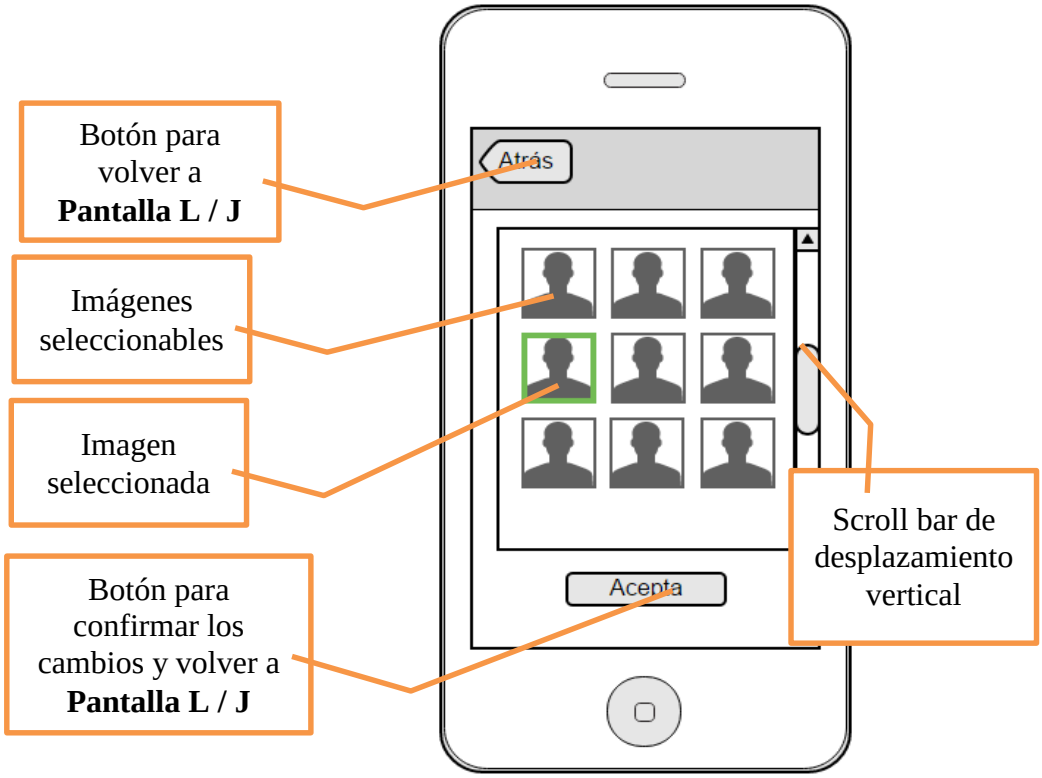


Figura 46. Pantalla S: Selección de Avatar

4.3. Mapa de navegación

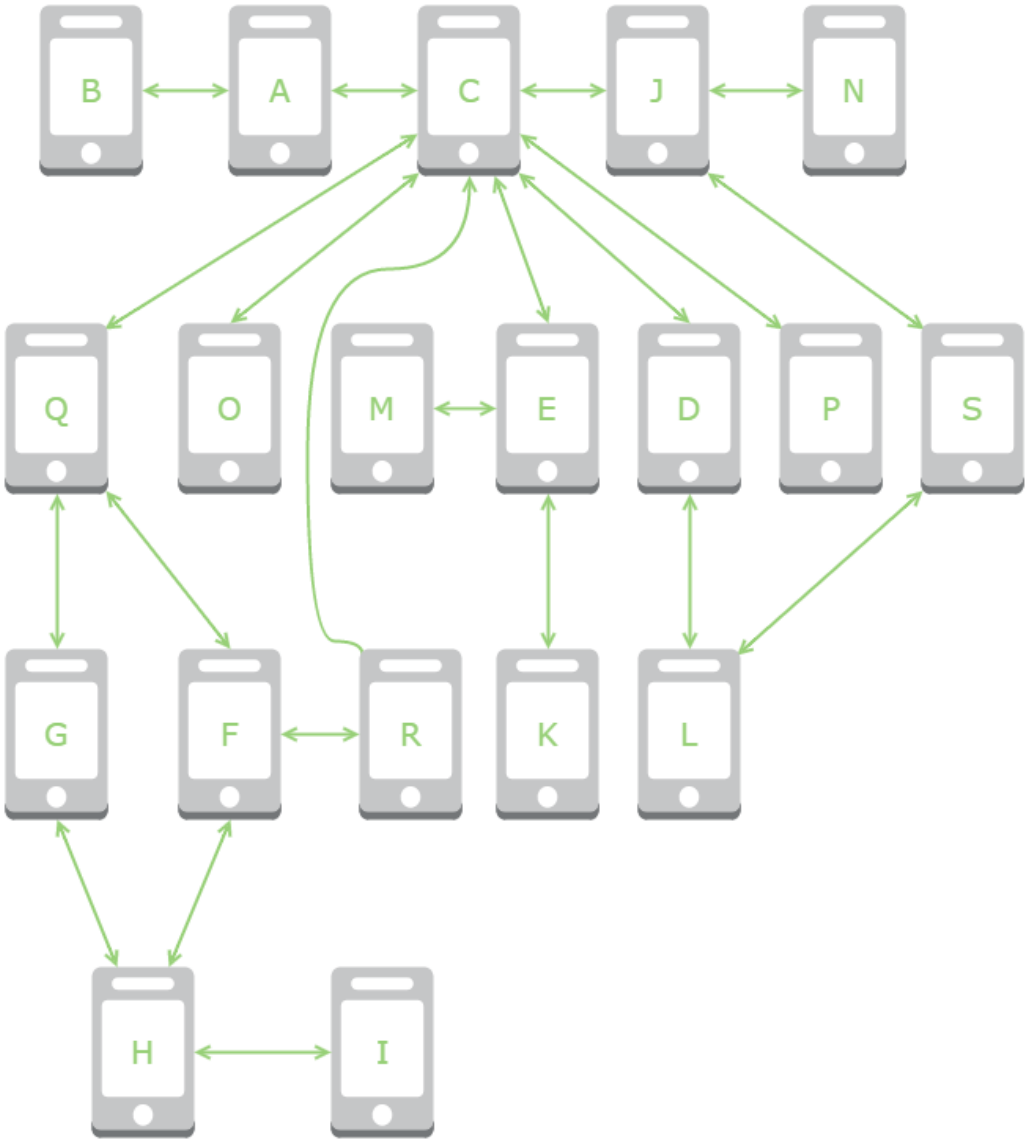


Figura 47. Mapa de navegación

5. CONCLUSIONES

5.1. ¿Qué hemos logrado?

A nivel de aplicación hemos conseguido crear las funcionalidades básicas para poder jugar a nivel local contra la CPU. Las jugadas de la IA implementadas no son aleatorias, más bien buscarán como batir al jugador. Todo lo construido es a nivel local así que usará memoria local para los datos, sin embargo, aunque no haya conectividad se pueden agregar usuarios ya predefinidos.

También se ha fijado una temática, estilos, fuentes, colores que identifican la aplicación a través de HTML5 y CSS3 puro. El código implementado en JavaScript está ya preparado para que fácilmente sean adaptables nuevos modos de juegos.

Principalmente lo que se ha construido es una demostración para ver en primera instancia cómo es y cómo funciona Mexme.

A nivel académico, hemos entendido y aplicado los pasos que intervienen en las prácticas de la ingeniería del software y como redactarlos en una memoria para su desarrollo a posteriori.

5.2. ¿Qué nos queda pendiente?

La aplicación se encuentra incompleta, ya no solo por las funcionalidades si no por la ausencia de ciertas librerías y herramientas como JQuery Mobile que habría dado un acabado más profesional y consistente a la aplicación.

Como anteriormente hemos explicado la aplicación híbrida trabaja a nivel local, por tanto, no hay un Web Service implementado y en funcionamiento alojado en un Hosting accesible por la red ni tampoco una base de datos.

Falta también un servicio hosting, a poder ser gratuito, donde desplegar el Web Service y la base de datos e integrar sus respectivas librerías, además de dar de alta la aplicación en el Marketplace.

A nivel de documentación podríamos extendernos mucho más y llegar a un nivel de especificación técnica mucho más complejo, pero con lo ya descrito es más que suficiente para tener algo sobre lo que partir.

La aplicación ha sido principalmente testeada en plataformas Android, hubiera estado bien probar otras.

5.2. ¿Qué podríamos mejorar?

Para mejorar la aplicación nos habría gustado mejorar el abasto funcional y buscar puntos negros que mejorar, por ejemplo:

- Establecer un chat para comunicar los usuarios.
- Añadir animaciones para darle frescura a la aplicación
- Un módulo funcional para reportar el mal comportamiento de los usuarios de la comunidad.
- Más modos de juego que den diversidad de ocio para el jugador.
- Un buscador integrado dentro de la aplicación para agilizar las tareas.
- Expandir la colección de cartas, actualmente es un poco pobre.

En cuanto al diseño gráfico de la aplicación nos hemos orientado por sentido común, gusto personal y referencias de otras aplicaciones. Habría estado muy bien estudiar qué aspectos hacen buena una GUI, qué criterios hemos seguido para crear la nuestra y cómo puede afectar a nivel de marketing y captación de nuevos consumidores.

Tampoco hemos hecho inmersión en la seguridad de la aplicación ni hemos hablado de protocolos de cifrado de datos, no hemos considerado si quiera que datos habría que cifrar y pese a ser un aspecto que no es perceptible debería ser considerado también.

Los fines de éste proyecto eran puramente técnicos, y aunque se han pensado maneras de monetizar la aplicación, hubiese estado bien desarrollar las ideas para elaborar un plan de marketing.

Bibliografía

[1] J. I. Herranz and A. M. Gómez, "¿Qué tecnología utilizo en mi aplicación móvil?," Paradigma Digital, 2014. [Online]. Available: https://www.paradigmadigital.com/dev/que-tecnologia-utilizo-en-mi-aplicacion-movil/. [Accessed: 12-May-2016].

[2] T. Rodríguez, "El futuro del desarrollo móvil: crear una web móvil o una aplicación nativa," GenBeta Dev, 2011. [Online]. Available: http://www.genbetadev.com/desarrollo-aplicaciones-moviles/el-futuro-del-desarrollo-movil-crear-una-web-movil-o-una-aplicacion-nativa. [Accessed: 12-May-2016].

[3] T. Rodríguez, "Lo mejor del 2012 para Genbeta Dev: tecnología más interesante para aplicaciones móviles," GenBeta Dev, 2013. [Online]. Available: http://www.genbetadev.com/desarrollo-aplicaciones-moviles/lo-mejor-del-2012-para-genbeta-dev-tecnologia-mas-interesante-para-aplicaciones-moviles. [Accessed: 12-May-2016].

[4] "Modelo–vista–controlador," Wikipedia, 2004. [Online]. Available: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador. [Accessed: 18-May-2016].

[5] "¿Cuánto cuesta crear una App móvil y cómo se desarrolla?," LanceTalent Blog. [Online]. Available: https://www.lancetalent.com/blog/cuanto-cuesta-crear-una-app-como-se-desarrolla/. [Accessed: 18-May-2016].

[6] "Los 3 tipos de aplicaciones móviles: ventajas e inconvenientes," LanceTalent Blog. [Online]. Available: https://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/. [Accessed: 19-May-2016].

[7] "Apps Hibridas vs Nativas vs Generadas. ¿Qué decisión tomar?," InnovaAge. [Online]. Available: http://www.innovaportal.com/innovaportal/v/696/1/innova.front/apps-hibridas-vs-nativas-vs-generadas-que-decision-tomar. [Accessed: 21-May-2016].

[8] M. Arlandy, "WebSockets con Java y Tomcat 7," AdictosAlTrabajo, 2012. [Online]. Available: https://www.adictosaltrabajo.com/tutoriales/web-sockets-java-tomcat/. [Accessed: 21-May-2016].

[9] S. Lissack, "WebSockets – A Quick Introduction and a Sample Application," IDR Solutions, 2013. [Online]. Available: https://blog.idrsolutions.com/2013/12/websockets-an-introduction/. [Accessed: 21-May-2016].

[10] M. Salazar and E. Moranchel, "Java EE 7: Building Web Applications with WebSocket, JavaScript and HTML5," Oracle. [Online]. Available: http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/HomeWebsocket/WebsocketHome.html. [Accessed: 21-May-2016].

[11] "jQuery.ajax()," JQuery. [Online]. Available: http://api.jquery.com/jquery.ajax/. [Accessed: 27-May-2016].

[12] “MIME Types List,” FreeFormater. [Online]. Available: http://www.freeformatter.com/mime-types-list.html. [Accessed: 27-May-2016].

[13] “AJAX,” Wikipedia, 2005. [Online]. Available: https://es.wikipedia.org/wiki/AJAX. [Accessed: 27-May-2016].

[14] “Platform Support,” Apache Cordova. [Online]. Available: https://cordova.apache.org/docs/en/latest/guide/support/index.html#core-plugin-apis. [Accessed: 27-May-2016].

[15] “Create your first Cordova app,” Apache Cordova. [Online]. Available: https://cordova.apache.org/docs/en/latest/guide/cli/. [Accessed: 27-May-2016].

[16] “Responsive Web Design,” JQuery Mobile. [Online]. Available: http://demos.jquerymobile.com/1.4.5/rwd/. [Accessed: 27-May-2016].

[17] “Introduction,” JQuery Mobile. [Online]. Available: http://demos.jquerymobile.com/1.4.5/intro/. [Accessed: 27-May-2016].

[18] “Manual de JQuery,” desarrolloweb. [Online]. Available: http://www.desarrolloweb.com/manuales/manual-jquery.html. [Accessed: 27-May-2016].

[19] “3. Técnicas para Identificar Requisitos Funcionales y No Funcionales,” Metodología Gestión de Requerimientos Metodología Gestión de Requerimientos. [Online]. Available: https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales. [Accessed: 5-May-2016].

[20] “HTML5 - WebSockets,” tutorials point. [Online]. Available: http://www.tutorialspoint.com/html5/html5_websocket.htm. [Accessed: 5-May-2016].

[21] D. Chaffey, “Mobile Marketing Statistics compilation,” Smart Insights, 2016. [Online]. Available: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/. [Accessed: 5-May-2016].

[22] D. Ledbetter, “Why You Should Avoid Developing These 4 Types Of Apps,” Bussines 2 Community, 2015. [Online]. Available: http://www.business2community.com/mobile-apps/why-you-should-avoid-developing-these-4-types-of-apps-01292516#kieRRIYAHMHav2kK.97. [Accessed: 5-May-2016].

[23] C. Maltez, “Historia y evolución del smartphone,” EL SMARTPHONE. [Online]. Available: http://smartphoneavancetecnologico.blogspot.com.es/p/historia-y-evolucion-del-smartphone.html. [Accessed: 5-May-2016].

[24] C. Martínez, “¿Cómo elegir entre Phonegap y Titanium?,” HTML5FÁCIL, 2015. [Online]. Available: http://html5facil.com/tips/como-elegir-entre-phonegap-y-titanium/. [Accessed: 5-May-2016].

[25] A. Yelmo, “Desarrollo de apps híbridas vs. apps nativas: ventajas e inconvenientes de cada técnica,” 2016. [Online]. Available: http://www.bitbotic.com/desarrollo-de-apps-hibridas-vs-apps-nativas-ventajas-e-inconvenientes-de-cada-tecnica/. [Accessed: 5-May-2016].

[26] “Remote Usage,” Adobe PhoneGap. [Online]. Available: http://docs.phonegap.com/references/phonegap-cli/remote-usage/. [Accessed: 12-May-2016].

[27] J. GM, “Los mejores juegos para iOS y Android, por géneros: para todos los gustos,” Tablet Zona, 2015. [Online]. Available: http://tabletzona.es/2015/09/26/mejores-juegos-generos-android-ipad/. [Accessed: 12-May-2016].

[28] “El impacto de los MEMES en la sociedad,” Noticias Kuriosas. [Online]. Available: http://www.noticiaskuriosas.com/el-impacto-de-los-memes-en-la-sociedad/. [Accessed: 12-May-2016].

[29] “El Poder De Memes De Internet,” SendBlaster. [Online]. Available: http://blog.sendblaster.com/es/2012/03/27/el-poder-de-memes-de-internet/. [Accessed: 12-May-2016].

[30] M. A. Alvarez and A. Basalo, “AngularJS Vs jQuery,” desarrolloweb, 2014. [Online]. Available: http://www.desarrolloweb.com/articulos/angularjs-vs-jquery.html. [Accessed: 12-May-2016].

[31] J. GM, “Los mejores RPG de cartas para tablets Android y iPad,” Tablet Zona, 2015. [Online]. Available: http://tabletzona.es/2015/03/28/los-mejores-rpg-de-cartas-para-tablets-android-y-ipad/. [Accessed: 12-May-2016].

[32] C. Vereau, “AngularJS vs BackboneJS, Jquery, ReactJS y otros,” DevCode. [Online]. Available: https://devcode.la/tutoriales/angularjs-vs-backbonejs-jquery-reactjs/. [Accessed: 12-May-2016].

[33] L. A. Gomez, “curso-desarrollo-de-aplicaciones-web-moviles-con-html5-y-jquery-mobile.” HTML5FÁCIL, 2013.

[34] A. Fernández, “Native-apps-vs-hybrid-apps-vs-web-apps.” Blogthinkbig, 2013.

[35] N. Hampson, “web_apps_vs_hybrid_apps_-vs_native_apps_comparison.” looksoftware, 2012.

[36] Pankaj, “WAR-directory-structure.” JournalDev, 2016.

[37] “Memes-y-Entre-Otros-510128.” pelacableando, 2014.

[38] R. Frey, “MVC-Process.” Wikipedia, 2010.

[39] D. Chaffey, “percent-time-spent-on-mobile-apps-2016.” Smart Insights, 2016.

[40] D. Chaffey, “Nielsenmonthlusagemobileappdevices1.” Smart Insights, 2016.

[41] D. Chaffey, “Mobile-stats-vs-desktop-users-global-700x515.” Smart Insights, 2016.

[42] D. Ledbetter, “Screen-Shot-2015-08-03-at-12.” Business 2 Community, 2015.