



TITANS

Plan de gestión, análisis, diseño y memorial del proyecto Zodiac

ÍNDICE

Introducción	2
Organización del proyecto	3
Plan de gestión del proyecto	4
Procesos	4
Procesos de inicio	4
Procesos de ejecución y control	5
Procesos técnicos	6
Planes	7
Plan de gestión de configuraciones	7
Plan de construcción y despliegue de software	8
Plan de aseguramiento de calidad	9
Calendario del proyecto y división de trabajo	10
Análisis y diseño del sistema	12
Análisis de requisitos	12
Diseño del sistema	13
Vista de módulos	13
Base de datos	15
Tecnologías	15
Interfaz	16
Memoria del proyecto	19
Conclusiones	19
Anexo I. Glosario	20
ANEXO II. Reglas del juego	21
Zodiac	21
Objetivo de Zodiac	21
Gameplay de Zodiac	22

1. Introducción

Zodiac, un juego de cartas similar al 'Uno' disponible para móvil y navegador web con crossplay. Las características más destacables de la aplicación se encuentran:

- Crossplay entre plataformas
- Partidas online de 2 a 6 jugadores
- Creación de salas privadas mediante código de acceso las cuales podrán ser configuradas dentro de unas opciones preestablecidas.
- Posibilidad de jugar partidas públicas
- Chat en partida
- Creación de usuario y edición de perfil.
- Seguir a tus amigos
- Modo practica con IA
- Tienda de cosméticos y personalización del juego
- Ranking de jugadores
- Posibilidad de pausar las partidas privadas
- Crossave de las partidas privadas a medias así como de los datos del perfil.

Se pretenden marcar cinco reuniones con el cliente para facilitar el seguimiento de su inversión así como para cruzar ideas y estado de desarrollo. Como el periodo de desarrollo estimado para completar el proyecto son 4 meses, dichas reuniones se establecerán en periódicamente cada 3 semanas, en la primera reunión se presentará una versión previa de los documentos referentes al plan de gestión, en las siguientes tres se procederá a enseñar una DEMO y una versión más refinada de los documentos, actualizando el estado del proyecto así como realizando las correcciones que parezcan pertinentes. Pasados 4 meses se realizará la entrega de todo lo generado, es decir, todo el código y documentación generado en ese periodo.

En cuanto a la estructura de este documento, en el siguiente punto se concretan los integrantes del equipo de desarrollo así como sus roles y responsabilidades dentro del proyecto.

Siguiendo, en el punto 3, nos encontramos el plan de gestión del proyecto en el cual por una parte se explican cómo se llevarán a cabo las distintas tareas a realizar en distintos momentos del proyecto como pueden ser los procesos de inicio, procesos de ejecución y control y los procesos técnicos. Así mismo en este punto se establecen los planes/objetivos que necesitamos para llevar a cabo el proyecto.

En el punto 4 nos encontramos vistas detalladas de los componentes, la arquitectura, también podemos encontrar el análisis de los requisitos. El punto 5 sirve para mostrar como se ha llevado a cabo el proyecto y como se ha desarrollado el proyecto frente a los planes y estimaciones realizados.

Por último el punto 6 recoge las conclusiones extraídas una vez se ha terminado el desarrollo así como las reflexiones e ideas que se pueden mejorar para futuros proyectos de esta envergadura.

2. Organización del proyecto

La organización TITANS S.L. es la encargada de llevar a cabo este proyecto, se ha constituido en el contexto de la asignatura Proyecto Software del tercer curso del grado de Ingeniería Informática de la Universidad de Zaragoza.

El grupo de trabajo está formado por 6 ingenieros informáticos: Hayk Kocharyan, Enrique Ruiz, Juan Jose Tambo, Alejandro Terron, Ana Alvarez y Adolfo Terron. Este último por experiencia será el director del proyecto, ya que tiene experiencia liderando equipos y conoce y sabe implementar numerosas técnicas de metodologías ágiles lo que puede ayudar a la consecución de los objetivos en las fechas estimadas.

También se han designado dos personas que serán las encargadas de asegurar que se están documentando los procesos seguidos y revisar periódicamente los documentos, estas personas son Adolfo Terrón y Ana Alvarez.

En cuanto a la división por subgrupos, se crearán 3 destinados a desarrollar lo máximo posible cada una de las distintas partes del sistema (Front-web, Front-movil, Back) hasta el punto en el cual empiecen a depender unas de otras.

El reparto de las tareas teniendo en cuenta esto es:

- Front:
 - Web:
 - Enrique Ruiz: GUI dentro de las partidas
 - Juan Jose Tambo: GUI navegación por la app
 - Móvil:
 - Ana Alvarez:
- Back:
 - Base de datos: Adolfo Terron
 - Lógica aplicación y API Rest: Alejandro Terron y Hayk Kocharyan

A pesar de esta organización cualquiera es libre de ayudar a cualquiera que se quede atascado en el transcurso de terminar sus tareas asignadas así como cambiar de campo en el que está siempre y cuando se hable y estudie con el jefe del proyecto.

Las decisiones serán tomadas en consenso con los componentes de la rama afectada así como con el propio jefe de proyecto le afecte o no a sus tareas. Dichas decisiones tendrán que ser comunicadas al resto de los componentes ya que como se ha comentado en el párrafo anterior existe libertad de movimientos entre las distintas ramas.

3. Plan de gestión del proyecto

3.1. Procesos

3.1.1. Procesos de inicio

El control de versiones se realizará a través de Git o en su defecto GitHub, software de control de versiones pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones.

Para aunar las distintas partes en las que se subdivide el proyecto se crea una organización en Github, así todos los participantes del equipo tiene acceso a cualquiera de los distintos sistemas en desarrollo facilitando la revisión y ayuda entre los mismos.

Para evitar que varias personas trabajen en lo mismo el director del proyecto asignará uno de los subsistemas y tareas específicas a esa parte a cada uno de los desarrolladores. De esta manera se evita la duplicidad de código y que se realice formación innecesaria.

Para la aplicación móvil se utilizará Flutter con Dart. Como se puede ver en el organigrama del punto anterior de esta parte en principio se encargará solo Ana, puesto que ella tiene experiencia en el desarrollo de aplicaciones móviles con este SDK. Aun así deberá dejar documentado los pasos que ha seguido para configurar el entorno y el proyecto.

El desarrollo de la app móvil se realizará mediante un emulador de android incluido en el IDE Android Studio y el propio dispositivo móvil del desarrollador una vez sea seguro probarlo en un dispositivo físico sin provocar daños permanentes en el mismo. Así mismo para esta tarea se encontrarán disponibles todos los dispositivos del grupo.

Para el desarrollo web al contrario que en el caso anterior se encuentran dos personas puesto que aunque existe experiencia en el grupo de desarrollo para web, no se tienen conocimientos muy amplios. Enrique Ruiz y Juan Jose Tambo se formarán a lo largo del desarrollo a través de videos de youtube así como búsquedas en la red de aquellos cosas que sean más concretas. Como en el caso anterior tendrán que documentar la configuración del entorno Visual Studio Code y ReactJS.

En el back ambas partes tienen suficiente formación previa como para desenvolverse sin muchos problemas tanto en el desarrollo de PostgreSQL como con Django. Como se ha indicado en los dos puntos anteriores existe documentación del proceso de configuración del entorno Visual Studio Code con Django (Python).

Finalmente el deploy se realizará en Heroku, a pesar que algunos de los componentes del equipo tiene experiencia en esta PaaS como cada uno tiene que ser capaz de desplegar el proyecto para realizar las pruebas que crea pertinentes se realizará una quedada para explicar la propia plataforma, configurarla y leer la documentación que los que ya tienen experiencia crean conveniente.

3.1.2. Procesos de ejecución y control

Las comunicaciones internas se realizarán a través de Whatsapp y google meet y/o discord. Para ello es necesario de crear grupos en el caso de Whatsapp o un servidor en el caso de discord. Cada una de las ramas del organigrama presentado en el apartado 2 es libre de utilizar los canales que le parezcan más cómodos, pero los canales generales donde se encuentren todos los miembros del grupo serán el grupo de texto y si se necesita realizar reuniones de carácter general, como las de seguimiento, una sala de google meet.

Así pues se dispondrá de un grupo de Whatsapp con carácter general y dos servidores de discord uno para back y otro para front en los cuales se realizarán las reuniones internas y dispondrán de canales de texto donde poder guardar los enlaces a la documentación que han accedido.

Todas estas reuniones tienen que llevar un registro de duración y temas discutidos, para ello los responsables de la documentación, Adolfo y Ana, deberán tener conocimiento y al menos estar uno presente en ellas.

Adolfo, como se ha dicho anteriormente, tiene experiencia como líder de grupo y en la utilización de Scrum entre otros, por lo que esta será la metodología que optaremos y él tomará el papel de Scrum Master. Debido a esto sus funciones principales a parte del desarrollo de la base de datos es la gestión de las cargas de trabajo en los distintos sprints. Los dailys serán obligatorios, en caso de que la persona en cuestión no pueda unirse a la llamada lo realizará mediante mensajes de texto en el grupo.

En cada daily se condensara en menos de 5 minutos lo que hemos realizado el día anterior y donde nos encontramos atascados si es el caso, a partir de esta información se reasignan tareas o apoyo para solventar la situación.

Cada dos semanas se realizará un sprint review en el cual se verá el progreso con respecto a los hitos logrados frente a los establecidos en el propio periodo así como en relación al cómputo global del proyecto.

Tanto los hitos como las deudas técnicas serán plasmadas en tableros de Kanban que se encontrarán dentro de la propia organización de github, existirá uno para el conjunto total y dos más, uno para el front y otro para el back. Los encargados de la documentación también desempeñarán el cargo de mantener actualizado este apartado.

La deuda técnica a no ser que sea excesiva es entendible puesto que pueden surgir imprevistos, a pesar de ello en el sprint review se realizará un análisis de dicha deuda y se hará una reflexión del por qué su existencia.

Las rencillas que puedan aparecer dentro de los distintos integrantes del grupo de desarrollo se intentarán resolver en privado, en caso que esto no resulte el director del proyecto intervendrá para buscar una solución.

3.1.3. Procesos técnicos

Para la aplicación móvil usamos el IDE Android Studio junto con el SDK de Flutter que hace uso del lenguaje Dart. Este IDE ha sido el seleccionado puesto que facilita la creación de un dispositivo móvil virtual y te permite elegir la versión de Android así como algunas características como tamaño de la pantalla, dispositivo al que asemejarse, etc.

Como se ha indicado con anterioridad se hará uso de dicho dispositivo virtual hasta que la aplicación sea lo suficientemente estable como para poderlo probar en un dispositivo real. Se ha descartado la implementación de la aplicación para sistemas operativos IOS debido a la gran diferencia entre este y Android, si es cierto que existen herramientas que te miran el código de un SO a otro, pero personalmente ninguno se fía de que lo haga perfecto y tanto en los plazos como en la estimación del precio de este proyecto no se ha tenido en cuenta.

Para la aplicación web, hacemos uso de ReactJS una biblioteca para desarrollo de sitios web basada en JavaScript. Este parte se desarrolla bajo el IDE de Visual Studio Code elegida puesto que todos los participantes del proyecto la usábamos por su adaptabilidad a diversos lenguajes y facilidad para instalar plugins que facilitan el desarrollo y debug del código. Hasta que el código no es estable como para subir una versión funcional al heroku este apartado es probado a través de LocalHost y el navegador Chromium. Una vez la versión es estable se sube a heroku para que todos puedan tener acceso a ella.

Finalmente para el backend se va a utilizar Visual Studio por las cualidades anteriormente nombradas con un plugin de Django y otro de PostgreSQL para poder tener acceso a la base de datos desde el propio entorno así como poder lanzar el servidor. Además para el testeo preliminar de los endpoint se hará uso de la herramienta PostMan que facilita el mockeo de las llamadas del front al back.

3.2. Planes

3.2.1. Plan de gestión de configuraciones

Para asegurar una buena gestión de versiones se hará uso de google drive para todo lo relacionado con la documentación puesto que contiene un control de versiones de los ficheros. Así mismo se creará un árbol de directorios y documentos para mantener un orden y poder acceder a ellos sin tener que perder tiempo buscando.

Para la gestión de versiones del código se hará uso de Git/Github. La política a seguir será que dentro de la organización cada subgrupo tendrá un repositorio específico dentro de los cuales existirán mínimo dos ramas, una de producción donde se encontrará la última versión estable y otra de desarrollo donde cada uno subirá sus cambios previo testeo en local. Para poder subir de desarrollo a estable será necesario realizar una pull request y que la acepten mínimo dos personas.

Esto está relacionado con las incidencias puesto que cada vez que se sube el código a la rama de producción es porque se ha finalizado una de las tareas estipuladas para el sprint. Debido a esto los hitos serán modificados por la segunda persona que acepte la petición de pull request.

Para mantener una cohesión y como todos nos sentimos más cómodos toda la documentación, variables en el código, etc se encontraran en español.

En cuanto a los estándares en el código, la parte de maquetación del código se dejará a el propio IDE, es decir, se hará uso de las opciones de formatear el código que presenten los propios entornos. No se seguirá ningún estándar en relación a la propia escritura del código.

Para facilitar la unión entre el front y el back, el repositorio correspondiente al mismo tendrá un apartado de Wiki en el que cada entrada estará formada por un endpoint en concreto con formas de uso, un ejemplo de todas las llamadas posibles y JSONs que devuelve. De la misma manera existirá una entrada referente a la base de datos donde existirá un esquema de los datos almacenados y sus relaciones.

Para todo aquello que no se encuentre recogido en el drive o en github, el propio desarrollador será el responsable de la gestión de copias de seguridad y control de versiones en local.

3.2.2. Plan de construcción y despliegue de software

Para la construcción de la parte del servidor mediante Django se realizará uso del compilador Python 3 por defecto, para la instalación de librerías se usará el comando “pip install” además de un fichero “.txt” para la gestión de estas cuya responsabilidad de actualizarlo recae sobre aquel que añada la biblioteca, si se le olvida la actualización al intentar desplegar el servidor en local saldrá un mensaje en la terminal de que falta por instalar x librería.

El despliegue en local se realizará a través de la integración de Django con el entorno, del mismo modo la utilización de la base de datos también se realizará a través de la integración.

En la parte de Web se usará el “npm” como instalador de paquetes gracias al cual se gestionan solos y no es necesario añadirlos a mano facilitando así que otra persona pueda coger el proyecto y únicamente mediante la ejecución del comando “npm install” tenga descargadas todas las dependencias necesarias, se hará uso también de este comando para el despliegue en local de la página web mediante el comando “npm start”.

Estas dos son las partes que necesitan ser subidas a la nube a través de Heroku. En esta plataforma existirán dos proyectos separados, uno para el servidor y otro para la página. A Heroku solo se subirá una nueva versión cuando exista una actualización en la rama de producción asegurando así que la parte desplegada está chequeada y funciona sin dar errores. El despliegue de la aplicación móvil no es necesario puesto que se hace uso de un dispositivo real.

A partir de que tengamos por primera vez dos versiones estables, una del servidor y otra de la web, se realizarán deploys semanalmente para comprobar que no se ha excedido la cuota de tamaño de la versión gratuita de Heroku y para verificar que se puede realizar el despliegue sin errores.

El despliegue de ambos contenedores puede realizarse mediante comandos en una terminal o mediante la interfaz que ofrece la propia plataforma.

3.2.3. Plan de aseguramiento de calidad

Para asegurar la calidad de la documentación se realizarán revisiones periódicas por los encargados de la documentación y eventualmente por uno de los demás integrantes que no haya participado, de esta manera se busca que a través de la iteración y la revisión por personas ajenas a la concepción del propio documento, el refinamiento y corrección de la documentación.

Para asegurar la calidad del backend se realizarán test unitarios manuales de todos los endpoints a través de la herramienta PostMan, asimismo se realizarán pruebas con respecto al modelo de la base de datos.

La aplicación móvil será testeada con test unitarios automatizados a través del paquete que proporciona el propio SDK de flutter_test. También se hará uso de Gherkin para simular el uso que le puede dar un usuario con la aplicación y así diseñar escenarios de interacción realistas.

Al igual que en el dispositivo móvil en la web se crearán test unitarios de la interfaz mediante React Testing Library.

Cada subgrupo será el encargado de gestionar y documentar las pruebas de su código, en aquellas pruebas que se necesite una coordinación entre distintas ramas primero se realizará un mock del otro extremo y una vez se corrijan los posibles errores se realizarán los test conjuntos.

En caso que una persona se encuentre con un problema que no sabe solucionar se procederá a la revisión por pares (él y otro compañero) del código en cuestión.


Para asegurar que la calidad final sea similar en ambos frontend se establecerán reuniones periódicas en las que se mostrará el funcionamiento de ambas y cruzaran ideas de implementación de ciertas partes.

3.2.4. Calendario del proyecto y división de trabajo

En el siguiente Diagrama de Gantt se recoge las tareas a realizar y la repartición de los requisitos en las dos iteraciones. Las tareas han sido divididas en documentación, testing, backend, frontend móvil y frontend web.

El reparto de trabajo queda de esta forma:

- La documentación se encargará cada uno de su parte y aquellas cosas que se realicen para el proyecto en general, mínimo tiene que haber presente un representante del front y otro del back.
- Desarrollo Front:
 - o Web: Enrique Ruiz y Juan Jose Tambo
 - o Móvil: Ana Alvarez:
- Desarrollo Back:
 - o Base de datos: Adolfo Terron
 - o Lógica aplicación y API Rest: Alejandro Terron y Hayk Kocharyan
- Del testing se encargará cada uno de su parte

Para una mejor visualización:  Diagrama de Gantt

En la figura las líneas rojas denotan los sprints y las amarillas las reuniones intermedias con los clientes. Columna negra del final es la deadline para entregar el proyecto

Consejo de Smartsheet →

Consejo de Smartsheet →

NOMBRE DE LA E

4. Análisis y diseño del sistema

4.1. Análisis de requisitos

Requisitos funcionales

1. El sistema obligará al usuario a registrarse con un email, nickname, foto elegida entre un lote proporcionado por la app y contraseña para poder jugar.
2. El sistema obligará al usuario a estar logueado para poder jugar.
3. El usuario podrá modificar en su perfil el nickname y/o la foto de perfil.
4. El usuario podrá eliminar su cuenta borrando así todos sus datos.
5. El usuario deberá poder buscar una partida pública
6. El usuario deberá poder abandonar cualquier tipo de partida en cualquier momento.
7. El usuario deberá poder crear una partida privada customizando el tiempo de turno, el nombre de la sala y el número máximo de jugadores.
8. El usuario deberá poder unirse a una partida privada a través de un código de cinco caracteres.
9. El sistema deberá administrar un código único de cinco caracteres una vez el usuario haya configurado la partida privada.
10. El sistema deberá ser capaz de crear una partida pública siempre y cuando mínimo hayan aceptado 2 usuarios la partida en menos de un minuto o se hayan alcanzado el máximo de 6.
11. El sistema deberá permitir al usuario revisar las normas del juego dentro de una partida.
12. El sistema deberá ser capaz de gestionar pausas en partidas privadas y reanudar la partida cuando el usuario que ha provocado dicha pausa lo decida.
13. El usuario deberá ser capaz de pausar una partida privada.
14. El usuario deberá ser capaz de salir de una partida.
15. El sistema deberá ser capaz de gestionar los abandonos de partidas devolviendo las cartas al mazo y barajando.
16. El sistema deberá dar por ganador a un jugador cuando este se queda solo.
17. El sistema deberá ser capaz de parar una partida privada si uno de sus participantes sale.
18. El sistema permitirá partidas de entre 2 a 6 jugadores
19. El sistema deberá permitir al usuario cambiar de dispositivo en medio de una partida privada.
20. El sistema deberá ser capaz de gestionar el tiempo de turno y en caso de vencimiento darle una carta más al jugador que ha perdido el turno
21. El sistema deberá finalizar la partida en caso que uno de los jugadores se quede sin cartas al cual dará por ganador.
22. El sistema deberá permitir la creación de partidas contra una IA.
23. El usuario deberá poder crear una partida contra la IA customizando el tiempo de turno, el nombre de la sala y el número máximo de jugadores.
24. El sistema deberá permitir la comunicación en partida mediante el chat.
25. El sistema deberá entregar una recompensa al ganador de la partida una vez finalizada.
26. El sistema deberá permitir la compra de cosméticos.
27. El usuario deberá poder equiparse cosméticos de dorso de carta o tapete desde su perfil
28. El usuario deberá poder realizar cualquier jugada siempre que no vaya contra las reglas.

Requisitos no funcionales

1. El sistema debe funcionar en entornos Android y web (Chromium).
2. La aplicación necesitará de acceso a internet como mínimo de 10 Mbps para poder jugar a cualquiera de sus modos.
3. El sistema guardará cualquier información proveniente del usuario de manera confidencial

4.2. Diseño del sistema

4.2.1. Vista de módulos

Las vistas de módulos sirven para ver la organización del código durante el desarrollo, señalando sus relaciones y estructura. A continuación se presentan dos vistas de módulos correspondientes con el backend y el frontend, en este caso no se ha hecho diferenciación dentro del front puesto que la estructura es muy similar

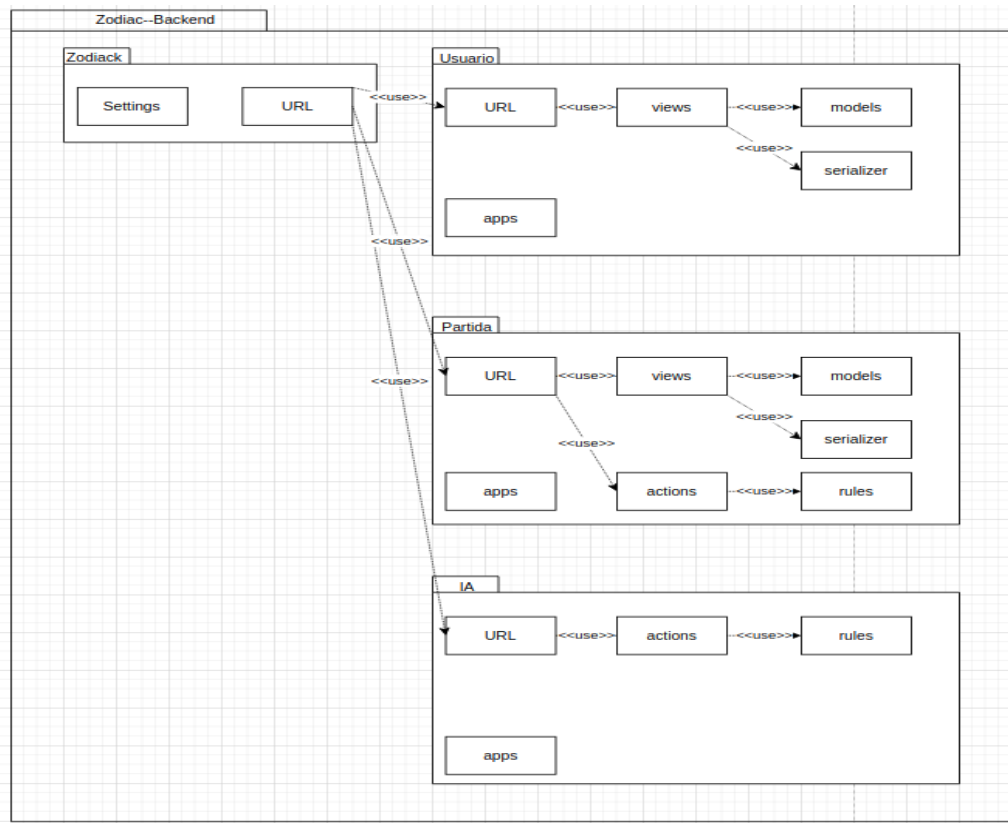


Figura 1: Vista de módulos back

Como se puede apreciar en la Figura 1 el backend está formado por un módulo denominado Zodiac que a través de urls es el encargado de determinar cuál de los paquetes restantes es el que se necesita. El paquete usuarios, partida y IA gestionan los accesos a usuario, partida en curso y a la IA respectivamente.

- urls son los encargados de gestionar las llamadas a la API Rest.
- views implementan el controlador del patrón MVC invocando los recursos de la base de datos necesarios y devuelve JSONS
- Models implementa el patrón ORM para la traducción de objeto a entidades en la base de datos PostgreSQL.
- acción son los que implementan los algoritmos de cálculo de movimientos.
- rules son la implementación de las reglas del juego.

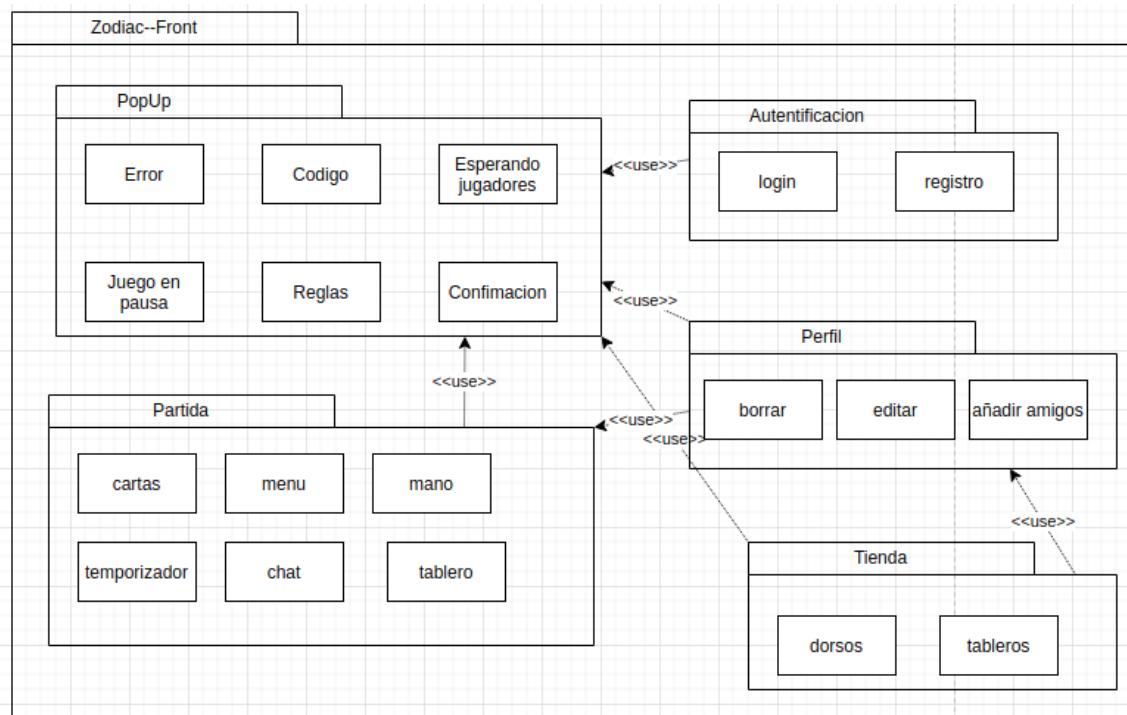


Figura 2: Vista modulos Front

- PopUp implementa todos los popups que pueden aparecer al usuario
- Partida contiene los módulos relacionados con el transcurso de la partida
- Perfil engloba los módulos relacionados con la gestión frontal del perfil del usuario.
- Autenticación engloba las llamadas al back referentes al registro y el login.
- Tienda: contiene los módulos referentes a la compra de dorsos y skins para los tableros.

4.2.2. Base de datos

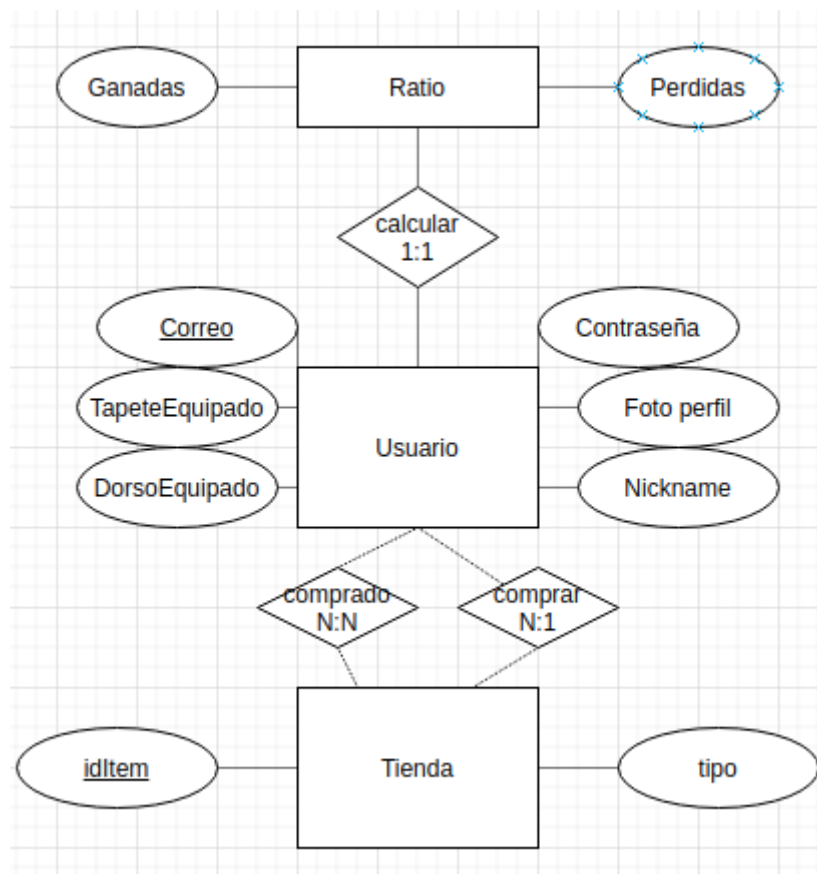


Figura 3: Esquema E-R

Este es el esquema entidad relación de la base de datos usado en el proyecto, se trata de una base de datos de tipo SQL al cual en el servidor se le aplica un patrón ORM para facilitar la traducción de objetos memoria usuario al esquema aquí presente.

4.2.3. Tecnologías

Django: Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo–vista–controlador. Se opta por este framework debido a que algunos de los miembros ya tiene experiencia en su uso. Además la sintaxis de python es más legible por lo que facilita el aprendizaje de aquellas personas que no lo han usado o no están familiarizadas con el lenguaje.

Flutter: Es un SDK de código fuente abierto de desarrollo de aplicaciones móviles. Suele usarse para desarrollar interfaces de usuario para aplicaciones en Android. El uso de este SDK es el mismo que en el caso anterior pero además este suscita curiosidad por aquellos que no tienen conocimiento en él.

ReactJS: React es una biblioteca Javascript diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Esta quizá sea la tecnología usada de la cual se tenía menos conocimiento pero al hacer algunas búsquedas vimos que existen numerosos tutoriales y documentación sobre ella.

4.2.4. Interfaz

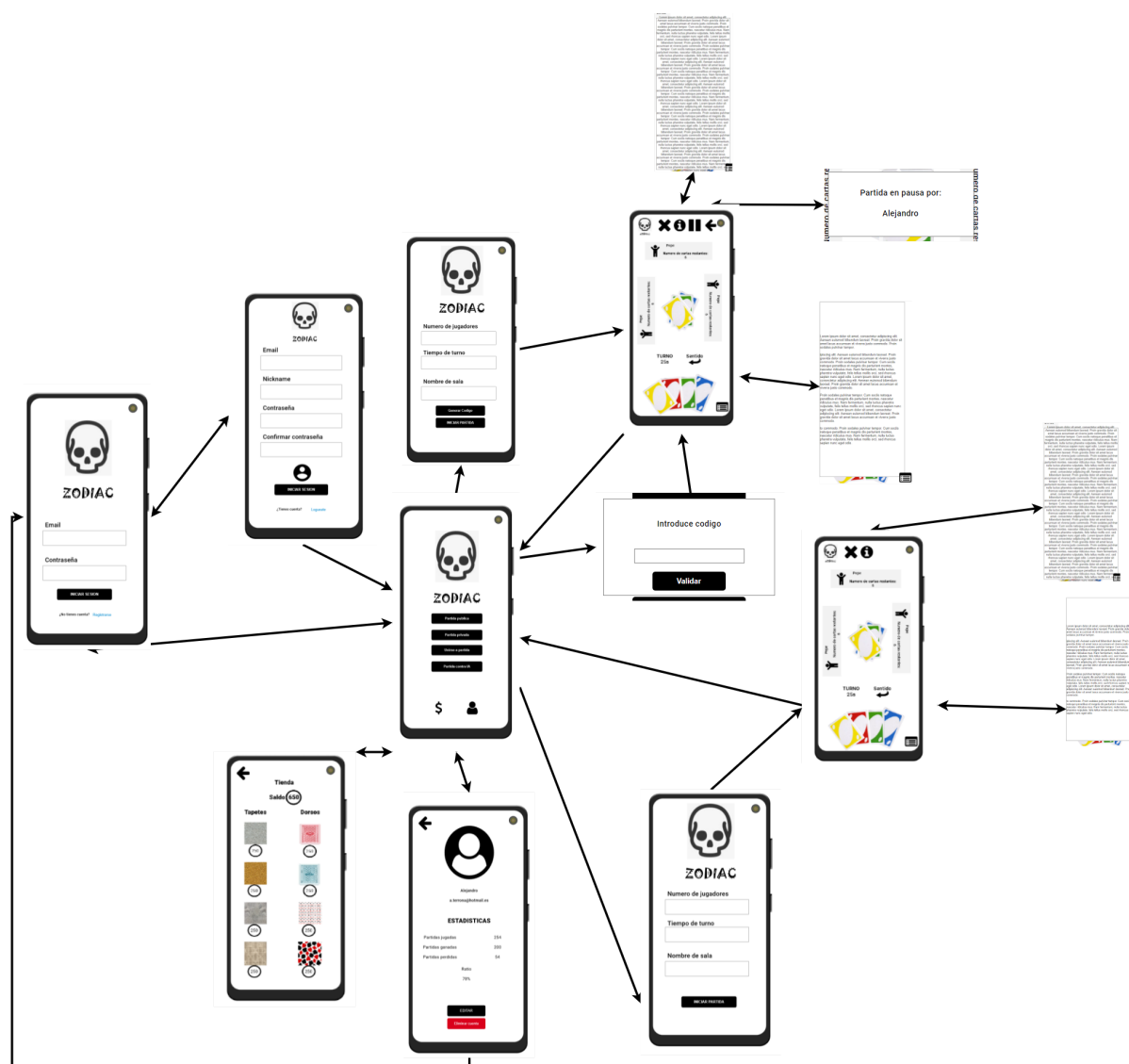


Figura 4: Mapa de navegación

En esta imagen se puede ver los recorridos que se pueden realizar a lo largo de la GUI. Solo se muestra la GUI de móvil puesto que la navegación y diseño son muy similares al web. Se ha omitido poner la alerta de error puesto que dificulta la visualización de la Figura 4.

- **Casos de uso**

Caso de uso Registrarse:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el usuario selecciona registrarse
- 2) El usuario introduce email, nombre de usuario, contraseña, confirma la contraseña y elige una imagen.
- 3) El sistema registra al usuario con dichos datos

Flujo de eventos alternativo:

- 3.1) El usuario confirma mal la contraseña.
- 4) El sistema lanza un mensaje de error.

Caso de uso Iniciar sesión:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el usuario abre la aplicación
- 2) El usuario introduce email y contraseña
- 3) El sistema identifica al usuario

Flujo de eventos alternativo:

- 3.1) El usuario pone email y contraseña no válidos.
- 4) El sistema lanza un mensaje de error

Caso de uso comprar un dorso:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el usuario selecciona la tienda
- 2) El usuario seleccionar un dorso para las cartas
- 3) El sistema guarda el color que ha comprado el usuario

Flujo de eventos alternativo:

- 3.1) El usuario no tiene monedas suficientes para comprar el color
- 4) El sistema lanza un mensaje de error

Caso de uso editar perfil:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el usuario selecciona editar perfil
- 2) El usuario modifica el nombre y/o cambia la foto de perfil
- 3) El sistema actualiza el perfil del usuario

Caso de uso crear partida contra la IA:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el usuario selecciona la opción de partida contra IA
- 2) El usuario selecciona el número de jugadores, el tiempo de turno
- 3) El usuario selecciona iniciar partida

Caso de uso unirse a partida pública:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el usuario selecciona la opción partida pública
- 2) El sistema introduce al usuario en una partida existente

Flujo de eventos alternativo:

- 2.1) No se puede formar una partida tras un minuto de espera.
- 3) El sistema vuelve a la página de inicio

Caso de uso crear partida privada:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el usuario selecciona la opción de partida privada
- 2) El usuario selecciona el número de jugadores, el tiempo de turno y el nombre de la sala
- 3) El usuario pide al sistema que genere el código
- 4) El sistema crea la partida

Caso de uso unirse a partida privada:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el usuario selecciona la opción de unirse a partida
- 2) El usuario introduce código de una partida ya creada
- 3) El sistema introduce al usuario en una partida existente

Flujo de eventos alternativo:

- 2.1) El usuario introduce de manera errónea el código
- 3) El sistema lanza un mensaje de error

Caso de uso jugar partida:

Flujo de eventos principal:

- 1) El caso de uso comienza al entrar en la partida
- 2) El usuario comienza el turno con 7 cartas y una en el centro
- 3) El usuario juega una serie indeterminada de turnos
- 4) Uno de los jugadores se queda sin cartas y la partida termina.
- 5) El sistema abona los puntos al ganador

Flujo de eventos alternativo:

- 3.1) El usuario abandona la partida
- 4) El sistema devuelve el mazo a la pila de cartas y baraja

5. Memoria del proyecto

He procedido al prototipado de la GUI de la aplicación para móvil en Axure, para que te permite loguearte tienes que usar email: a.terróna@hotmail.es y contraseña: alejandro, para el registro ambas contraseñas tienen que ser las mismas. He intentado implementar PopUP que duren un tiempo en pantalla y posteriormente te cargue otra página pero me ha resultado imposible.

6. Conclusiones

La planificación y organización de Zodiac ha resultado menos costosa de lo que cabía esperar debido a que ya se tenía experiencia. Además que a la hora de imaginar la implementación como en mi caso he usado las mismas tecnologías que en el proyecto fallido de brainbook he podido tener en cuenta los problemas que tuvimos tanto con las limitaciones tecnológicas como el falso control que poseíamos sabiendo que estaba desarrollada cada una de los integrantes pero no su avance.

En este proyecto he optado por el uso de metodologías ágiles, especialmente Scrum, para evitar una posible situación parecida. Gracias a esta metodología y a sus ceremonias como las dailys en todo momento sabes además de quien desarrolla que, el estado al día. Además te permite tener una visión global de lo que queda por hacer facilitando así la planificación a futuro y a mi tener estructurado y conocer en todo momento qué es lo que tengo que hacer y cuando tengo que tenerlo terminado me ayuda a no procrastinar.

Como en la vez anterior he optado por una figura central que sea el que maneje la carga de los springs, y esté atento a todos los grupos porque me parece imprescindible la figura de un líder que guíe al equipo. Por otro lado, aunque en papel puede que haya dado la impresión errónea he tratado de que los equipos sean muy flexibles, es decir, que por ejemplo cualquier persona en cualquier momento se pueda poner con otra parte del sistema siempre y cuando no vaya en declive del trabajo y la carga de los demás.

Esto me ha servido para darme cuenta de lo esencial que es tener un diseño inicial del sistema estipulado y documentado e iterar sobre el mismo en vez de empezar a desarrollar código intentando replicar un mapa mental.

De la misma manera la planificación a largo plazo aunque difícil de realizar ayuda a tener una visión global y ha estimado el esfuerzo total de desarrollo, también ayuda a tener todas las ideas importantes claras desde un inicio y a ponerte a ti mismo objetivos alcanzables.

En general, los proyectos me han resultado interesantes y el trabajo tanto en equipo como individual ha sido mejor de lo esperado.

Anexo I. Glosario

Crossplay: Crossplay significa juego cruzado. En el mundo de los videojuegos se refiere a compartir una partida entre consolas o dispositivos distintos.

Deploy: es utilizada para describir que algo fue colocado en su posición, cuando un sistema es habilitado para su uso para realizar pruebas o producción.

Nickname: apodo/alias

Backend: es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione.

Frontend: la parte del sitio web con la que el usuario interactúa

API REST: conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el software de las aplicaciones la cual se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web

Crosssave: La función de guardado cruzado entre cuentas nos permite vincular cuentas de dos plataformas diferentes y compartir el progreso, así como los datos en el juego entre las dos cuentas.

DEMO: la versión reducida en prestaciones de un programa para poder utilizarlo y evaluarlo antes de su compra.

IDE: entorno de desarrollo integrado

PaaS: Plataforma como servicio. Se trata de un conjunto de servicios basados en la nube que permiten a los usuarios empresariales y desarrolladores crear aplicaciones de forma rápida y rentable.

SO: Sistema operativo

IA: Inteligencia artificial

SQL: lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.

Framework: estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de *software*, que puede servir de base para la organización y desarrollo de software.

ANEXO II. Reglas del juego

Zodiac

El Zodiac es un juego que se juega con un mazo especialmente diseñado y que cuenta con 108 cartas. En ese mazo hay dos tipos de cartas: las normales y las especiales, también llamadas cartas comodines o cartas de acción.

El mazo del Zodiac está compuesto de cuatro colores: azul, verde, rojo y amarillo. Las cartas comunes están representadas por 9 signos del zodiaco y cada color tiene repetido 2 veces cada uno,

Con lo cual, el mazo cuenta con 18 cartas de cada color. También vienen 8 cartas especiales Roba Dos (un 2 antecedido de un signo +, y vienen dos de cada color); 8 cartas especiales Cambio de Sentido (2 de cada color); 8 cartas especiales de Bloqueo (2 de cada color); 4 cartas especiales Comodín Cambio de Color; 2 cartas especiales Comodín Visibilidad cartas de mazo (tiene un ojo en el centro); 2 cartas especiales Comodín Cambio de Color e intercambio de mazos tiene dos flechas formando un círculo); y 4 cartas especiales Comodín Cambio de Color y Roba Cuatro (tiene un 4 antecedido del signo +).

Objetivo de Zodiac

El Zodiac se puede jugar con un número de 2 a 10 integrantes. Pero es mucho más dinámico y atractivo jugar con 3 o 4 personas. El objetivo del juego es deshacerse de todas las cartas con las que se juegan desde el inicio (7 cada jugador) más las que se van robando en el medio del juego.

Gameplay de Zodiac

En el inicio, una vez barajado el mazo y repartidas las 7 cartas a cada jugador, se pone junto al mazo una carta dada vuelta como descarte, que será el color o signo que deberá seguirse. Y se empiezan a descartar las cartas entre los jugadores, en sentido de las agujas del reloj. Eso sí, una de las cartas es Cambio de Sentido con lo cual, si se está jugando -por caso- de izquierda a derecha con un color azul y alguien tira el cambio de sentido azul, se comienza a jugar de derecha a izquierda.

Cuando llega el turno de cada jugador (30 segundos) , este está obligado a tirar una carta del color que se está jugando o una carta con el mismo signo que han tirado antes de ese turno. Si no se tiene una carta del color o del signo que nos han tirado y tampoco se tiene una carta especial o si se termina el turno, el jugador debe llevarse una carta del mazo como penalidad.