TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica
Superior de Jaén

Prototipo de aplicación para juegos de cartas basados en microservicios

Autor: Juan Manuel Valcárcel Sánchez

Grado: Ingeniería Informática

Director: Víctor Manuel Rivas Santos
Departamento de informática

Fecha: 16/05/2024

Universidad de Jaén

Departamento de Informática

Víctor Manuel Rivas Santos, del Trabajo Fin de Grado titulado: '**Prototipo de aplicación para juegos de cartas basados en microservicios**', que presenta Juan Manuel Valcárcel Sánchez, da el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2024

Firma estudiante:	Firma tutor:
Firmado por Juan Manuel Valcárcel Sánchez el día 11/06/2024 con un certificado emitido por AC FNMT Usuarios	*Firmado por RIVAS SANTOS VICTOR MANUEL - 29798183G el día 26/06/2024 con un certificado emitido por AC FNMT Usuarios*
Juan Manuel Valcárcel Sánchez	Víctor Manuel Rivas Santos

Agradecimientos

Agradezco a mi tutor por la gran ayuda que me ha dado durante todo el proceso de la realización de este proyecto.

Ficha del Trabajo Fin de Título	
Titulación	Ingeniería Informática
Modalidad	Proyecto de ingeniería
Especialidad (solo TFG)	Ingeniería del Software
Mención (solo TFG)	Sin mención.
Idioma	Español
Tipo	TFG general
TFT en equipo	No
Autor/a	Juan Manuel Valcárcel Sánchez
Fecha de asignación	22-11-2023

Descripción corta	Se realizará un proyecto de pequeño alcance con el propósito de crear una aplicación con una arquitectura basada en microservicios. El propósito principal de este TFG será discernir cuáles son los principales inconvenientes y ventajas de este tipo de arquitecturas en proyectos de esta escala. Además, se detallarán los problemas generados en el desarrollo del proyecto para discernir si han sido debido a la arquitectura escogida.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
Estas normas se han utilizado como base o referencia para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como proyecto la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.	

Contenido

Índice de ilustraciones

Índice de tablas

1 ESPECIFICACIÓN DEL TRABAJO

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*" [1].

1.1 Introducción

El objetivo principal del trabajo consiste en el estudio de la arquitectura basada en microservicios [2], para ello desarrollaremos una aplicación que nos permita definir las ventajas e inconvenientes de este tipo de arquitectura.

Aparte de lo anterior, se definirá y ejecutará una metodología ágil para solventar los posibles problemas que se puedan presentar en el proyecto, debido a su fuerte tolerancia a los cambios y a que permite el desarrollo de una aplicación de forma incremental.

Por último, el principal punto de desventaja de los microservicios [3] se encuentra en su aplicación a proyectos pequeños (debido al elevado coste inicial que conlleva construir la estructura) y de corta duración (cómo es la naturaleza de este proyecto) lo que permitirá definir los límites de esta arquitectura en los proyectos de este tipo.

1.2 Objetivos del trabajo

1. Creación de una aplicación en la cuál se dispongan varios tipos de juegos de cartas.
2. Usar una metodología ágil[1].
3. Implementar la aplicación con una arquitectura de microservicios [2].
4. Definir los puntos a favor y en contra de esta clase de arquitecturas en este tipo de proyectos.

1.3 Antecedentes y estado del arte

En este apartado se describirán las aplicaciones ya existentes que han servido de inspiración para el desarrollo del prototipo, las cuales son relevantes para comprender el proceso de diseño.

Finalmente, se destaca la importancia de estas aplicaciones en el desarrollo del prototipo y cómo éstas han contribuido a su diseño final.

[1] https://www.atlassian.com/es/agile

Solitaire Paradise[2]

Esta aplicación web nos ofrece un catálogo de juegos disponible a través de su página sin necesidad de descarga alguna.

Su catálogo de juegos, que podemos observar en la imagen 1, nos ofrece juegos de cartas (aparte de otra variedad de juegos), pero su principal desventaja es que haciendo alusión a su nombre, sólo ofrece juegos de modalidad un jugador o solitario.

Por tanto, cuando nos ofrece un juego de varios jugadores, la página web solo nos da la posibilidad de enfrentarnos contra una IA cerrando completamente el juego a las opciones multijugador, ya sean locales o remotas.

Imagen 1: Página principal de la aplicación web Solitaire Paradise

[2] https://www.solitaireparadise.com/es/

Isla de juegos[3]

En esta aplicación web se halla un catálogo de juegos variado de todas las diferentes categorías ya sean multijugador (local o remoto) o solitario.

En cambio, como podemos observar en la imagen 2, el contenido específico a juegos de cartas es casi inexistente, siendo este el punto dónde queremos centrar nuestro proyecto.

Se observa también en los juegos ofrecidos la falta de un sistema de puntos o un ranking dentro de estos que es bastante común en los juegos de cartas.

Imagen 2: Página principal de la aplicación web isla de juegos

[3] https://www.isladejuegos.com

Microsoft Solitaire Collection[4]

En contraposición a las dos anteriores, esta aplicación es de escritorio, lo que nos referencia que este tipo de aplicaciones no se limita sólo al escenario web, como podemos observar en la imagen 3.

En relación a su contenido, nos encontramos una especialización en los tipos de juego llamados solitario, teniendo un catálogo con muchas variedades de este juego. Pero aparte de este juego y sus variantes no encontramos otro tipo de juegos, siendo estos los únicos existentes. Cabe recalcar, que tienen un buen sistema de puntuación de partidas y personalización de los estilos de barajas, pero no es posible jugar con la baraja española.

Imagen 3: Página de la microsoft store del juego

[4] https://www.microsoft.com/store/productId/9WZDNCRFHWD2?ocid=pdpshare

En el presente, no se han hallado otras aplicaciones que recopilan distintos juegos de cartas, sino que se limitan a recopilar “minijuegos” o sólo se especializan en un conjunto pequeño de juegos de cartas.

Este proyecto plantea satisfacer estas necesidades, para poder recopilar en una sola aplicación los distintos juegos de cartas, siendo su único objetivo este tipo de juegos.

Se satisfacerá la inexistencia de una aplicación especializada en este tipo de juegos, además de dar la posibilidad de que terceros amplíen el catálogo de la aplicación,dando un manual de usuario para que puedan implementar juegos, que posteriormente se puedan añadir a la aplicación.

Además gracias a la arquitectura de microservicios se podrán implementar distintas interfaces de usuario para la aplicación, facilitando llevarla a distintas plataformas.

1.4 Descripción de la situación de partida

Para la creación de esta aplicación se ha decidido partir de cero, siendo un proyecto totalmente nuevo. Esta decisión se ha tomado debido a las restricciones del proyecto, ya que debido a la restricción de la arquitectura, se concluyó que iniciar el proyecto de cero es la mejor opción para implementar este tipo de arquitectura.

El proyecto se realizará con una metodología ágil, Kanban, elegida debido a la naturaleza unipersonal del proyecto y a la alta adaptabilidad característica de los métodos ágiles. Además se utilizará una arquitectura basada en microservicios [2] debido a que el proyecto se basa en añadir diferentes juegos de cartas a la aplicación y que estos tienen bastante similitudes, se estima favorable la escalabilidad y reutilización de los distintos microservicios.

Al utilizar microservicios, posibilita el disponer de diversos tipos de clientes para los distintos juegos que se propongan [4], lo que facilita llevar la aplicación a otros dispositivos o plataformas con facilidad.

1.5 Requisitos iniciales

- Poder expandir con facilidad el catálogo de juegos de la aplicación, además de poder exportar sus funcionalidades a otras plataformas.

- Tener una aplicación dónde poder jugar distintos juegos de cartas.

- Definir las características de las arquitecturas basadas en microservicios en proyectos pequeños.

1.6 Alcance

Debido a que se utilizará una metodología ágil en este proyecto, el alcance se irá definiendo para cada una de las iteraciones realizadas durante el mismo.

Cada iteración definirá su propio alcance (que lo definirá las historias de usuarios seleccionadas para esa iteración), siendo el alcance del proyecto la sumatoria de los alcances de las distintas iteraciones.

1.7 Hipótesis y restricciones

El TFG se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

Debido a que uno de los objetivos del proyecto se basa en hacer un estudio de las arquitecturas basadas en microservicios en proyectos pequeños no se han estudiado más alternativas con el objetivo de realizar el estudio ya mencionado.

1.8 Descripción de la solución propuesta

Se realizará una aplicación con una arquitectura basada en microservicios debido a la naturaleza idónea de estos para cumplir los objetivos de escalabilidad del programa. Además, se pretende realzar la capacidad de reutilización de los microservicios aprovechando las similitudes de los distintos juegos que se pretenden implementar en el proyecto.

- Como lenguaje de programación se usará JavaScript[5] debido a experiencias previas con microservicios implementados en este lenguaje.
- Cómo framework se usará Express[6] debido a experiencias previas con microservicios implementados en este lenguaje.
- Cómo entorno de ejecución se usará NodeJs[7] debido a experiencias previas con microservicios implementados en este lenguaje.
- Se usará el enrutamiento de puertos de Visual Studio Code[8] para probar la aplicación de manera remota.

Se usará la metodología ágil Kanban[9] debido al carácter unipersonal del proyecto, siendo una metodología ágil la elegida en consecuencia de la necesidad de recolectar información retrospectiva de terceros para la mejora de la aplicación en el menor tiempo posible.

1.9 Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFG, no existen restricciones de tipo económico, sino de tipo temporal (un número aproximado de horas). Por consiguiente, los cálculos de tamaño del proyecto están supeditados al tiempo disponible. En cuanto al esfuerzo, se dispone de tan solo un efectivo (la persona autora del trabajo).

Debido al uso de la metodología ágil Kanban las iteraciones no tienen un tiempo definido, sino que varían según la dificultad de las funcionalidades seleccionadas para ser implementadas en cada iteración.

[5] https://developer.mozilla.org/es/docs/Web/JavaScript
[6] https://expressjs.com/es/
[7] https://nodejs.org/en
[8] https://code.visualstudio.com
[9] https://kanbantool.com/es/metodologia-kanban

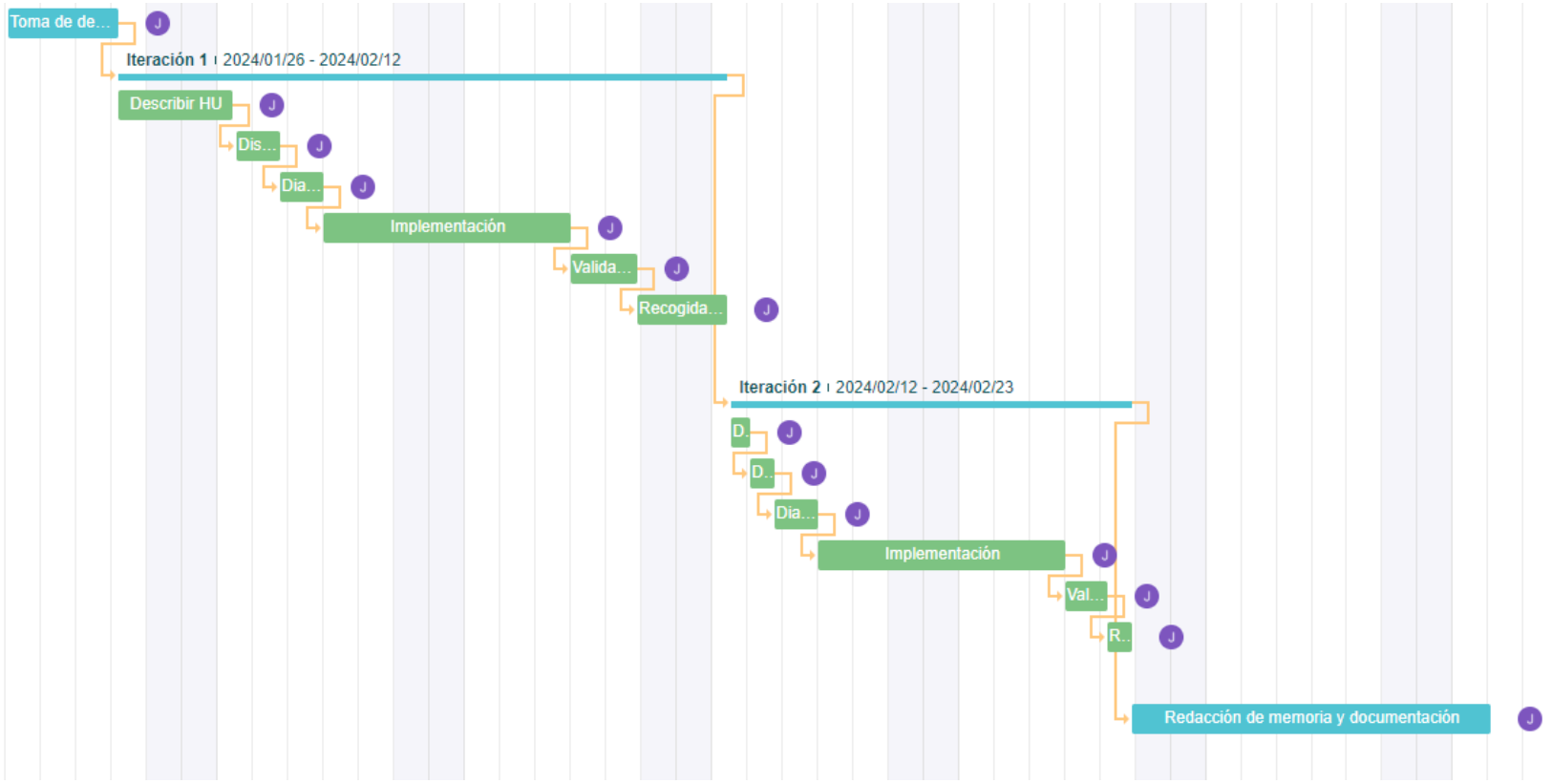


Imagen 4: Diagrama de Gantt del proyecto

En el diagrama de Gantt referido en la anterior imagen, definimos las cuatro etapas principales por las cuales va a pasar el proyecto:

1. **Toma de decisiones**: en esta etapa definimos los principales factores limitantes del proyecto, siendo elementos como el alcance del mismo, el presupuesto a gastar, las limitaciones en los recursos técnicos y las restricciones de implementación y diseño, ya sea aplicando arquitecturas de software específicas (cómo es el caso de este proyecto), qué metodología seguir o las herramientas se utilizarán.

2. **Iteración 1**: en esta etapa se comenzará la creación del programa, siendo su principal objetivo crear un PMV[10] (producto mínimo viable), a través del cuál poder tener un punto de partida que irá mejorando en las sucesivas iteraciones.

3. **Iteración 2**: definimos, por el tiempo estipulado, que sólo dará tiempo a realizar dos iteraciones del proyecto, dejando esta segunda iteración para mejorar el PMV creado en la primera iteración y para poder retroalimentarse del feedback obtenido por los usuarios de la iteración anterior.

4. **Redacción de memoria y documentación**: para finalizar el proyecto se redactará una memoria que englobará y explicará las decisiones tomadas en él. Además se documentará el código creado, dando instrucciones para que, si se diese el caso, se pudiera proseguir con el proyecto a posteriori.

[10] https://es.eserp.com/articulos/producto-minimo-viable/#:~:text=siglas%20en%20inglés).-,¿Qué%20es%20el%20producto%20mínimo%20viable%3F,su%20interés%20por%20el%20mismo.

Las dos iteraciones propuestas presentan las mismas subfases por generalizar las iteraciones, esto no elimina la posibilidad de que en una iteración se añada o elimine un paso de esta, pues el contenido de las historias de usuario que se realizarán en cada iteración pueden demandar el cambio de la estructura de la iteración en sí.

Dicho esto, ahora se presentan las fases que se han dictaminado como componentes de una iteración:

1. Descripción completa de las historias de usuario de la iteración.
2. Diseño de las interfaces de usuario.
3. Creación del diagrama de secuencia del sistema.
4. Implementación.
5. Validación de las historias de usuario implementadas.
6. Obtención de feedback por parte de los usuarios.

Debido a que se delimitan unas 300 horas de trabajo para el TFG, se pondrá de presupuesto el cobro del salario medio de un Ingeniero Informático en España[11] que son 13,85€/h.
El presupuesto, por tanto, del proyecto es: 13,85 € * 300h = **4.155,50€**.

En el caso de querer desplegarlo en producción podremos comprar un dominio[12] .es con coste de **33,80€** (renovándose por el mismo precio) y realizar un hosting con cloudflare[13] de manera gratuita.

[11] https://es.talent.com/salary?job=ingeniero+informático#:~:text=¿Cuánto%20gana%20un%20Ingeniero%20informático%20en%20España%3F&text=El%20salario%20ingeniero%20informático%20promedio.hasta%20€%2036.250%20al%20año.

[12] https://www.dominios.es/es/registra-un-dominio/cuanto-registrarlo
[13] https://www.cloudflare.com/es-es/plans/

1.10 Herramientas utilizadas

A continuación, se presentan las herramientas software que han sido de aplicación en la elaboración del trabajo y/o en la puesta en práctica del mismo.

- Para la organización de las iteraciones se ha usado Trello[14], definiendo un tablero de trabajo virtual (imagen 4) para definir mejor el flujo de trabajo dentro de las iteraciones.

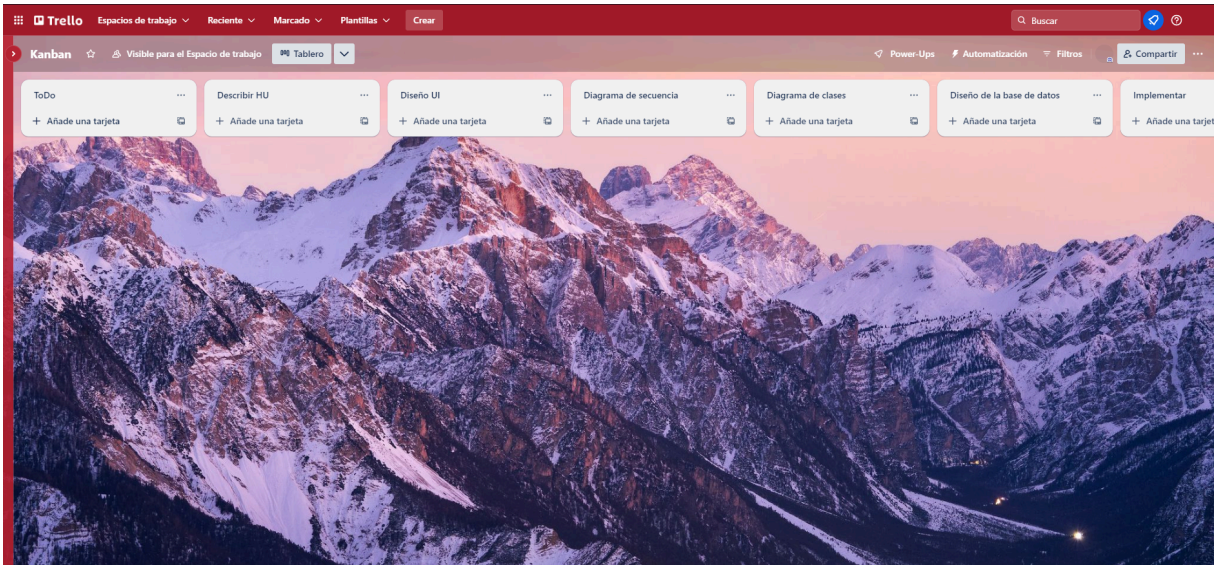


Imagen 5: Tablero Kanban creado en Trello

- Para el control de versiones y como repositorio se usó GitHub[15] https://github.com/jmvs0006/TFG.

[14] https://trello.com
[15] https://github.com

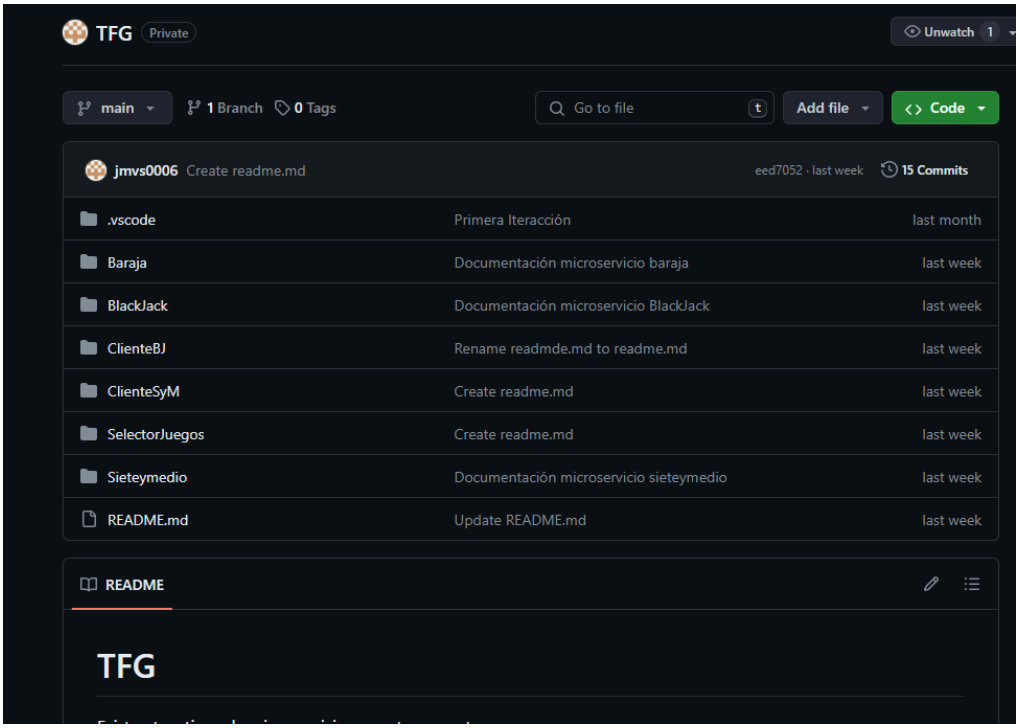


Imagen 6: Repositorio en GitHub

- Para la creación de diagramas necesarios para definir el diseño de la aplicación se usaron Visual Paradigm[16] y Creately[17].

- Para edición de código se usó Visual Studio Code.

- Para la gestión de tiempos del proyecto se ha usado GanttPro[18].

[16] https://www.visual-paradigm.com
[17] https://creately.com/es/home/
[18] https://app.ganttpro.com

2 Diseño inicial

Se definirán funcionalidades que el programa podría tener o que los usuarios podrían querer dentro de la aplicación. Además, se definirá la arquitectura de la aplicación y las diferentes decisiones de diseño realizadas.

2.1 Especificaciones del sistema

Debido al uso de una metodología ágil, nuestros requisitos se van a estructurar como historias de usuario (HU) [5]. Estas HU han sido catalogadas en importancia según el sistema MoSCoW[19] definiendo así la relevancia de estos requisitos para el usuario, con la finalidad de organizar el trabajo según la prioridad de las historias de usuario dada por un hipotético cliente.

Estas son las historias de usuario iniciales:

Código	Nombre	Prioridad
HU-1	Registrar Usuario	S
HU-2	Loguear Usuario	S
HU-3	Jugar a las Parejas	M
HU-4	BlackJack	M
HU-5	Elegir Juego	M
HU-6	Mazo aleatorio	S
HU-7	Puntuación de partidas	C
HU-8	Consultar puntuación	C
HU-9	Selector de dificultad IA	C

19 https://www.productplan.com/glossary/moscow-prioritization/#:~:text=MoSCoW%20prioritization%2C%20also%20known%20as,will%20not%20have%20right%20now.

HU-10	Tutorial	W
HU-11	Multijugador	C
HU-12	Estilos de carta	W
HU-13	Diferentes tipos de barajas	C
HU-14	Reloj de arena	C
HU-15	Top jugadores con mejor puntuación	W

Tabla 1: Historias de usuario definidas

Historia de usuario			
Número	1	Nombre	Registrar Usuario
Descripción			
Como usuario **quiero** poder registrarme **para** poder crear una cuenta.			
Prioridad			
S			

Tabla 2: HU-1 Registrar Usuario

Historia de usuario			
Número	2	Nombre	Loguear Usuario
Descripción			
Como usuario **quiero** poder iniciar sesión **para** poder acceder a mi cuenta.			
Prioridad			
S			

Tabla 3: HU-2 Loguear Usuario

Historia de usuario			
Número	3	Nombre	Jugar a las parejas
Descripción			
Como usuario **quiero** poder jugar al juego de las parejas **para** tener más variedad de juegos.			
Prioridad			
M			

Tabla 4: HU-3 Jugar a las parejas

Historia de usuario			
Número	4	Nombre	BlackJack
Descripción			
Como usuario **quiero** poder jugar al BlackJack contra la máquina **para** probar un juego con baraja francesa			
Prioridad			
M			

Tabla 5: HU-4 BlackJack

Historia de usuario			
Número	5	Nombre	Elegir juego
Descripción			
Como usuario **quiero** poder elegir qué juego jugar **para** elegir que quiero jugar.			
Prioridad			
M			

Tabla 6: HU-5 Puntuación de partidas

Historia de usuario			
Número	6	Nombre	Mazo aleatorio
Descripción			
Como usuario **quiero** poder aleatorizar el mazo de juego **para** diferenciar cada partida.			
Prioridad			
S			

Tabla 7: HU-6 Mazo aleatorio

Historia de usuario			
Número	7	Nombre	Puntuación de partidas
Descripción			
Como usuario **quiero** poder tener una puntuación de cada partida **para** saber cómo he jugado.			
Prioridad			
C			

Tabla 8: HU-7 Puntuación de partidas

Historia de usuario			
Número	8	Nombre	Consultar puntuación
Descripción			
Como usuario **quiero** poder consultar mi puntuación en un juego determinado **para** saber mi progreso.			
Prioridad			
C			

Tabla 9: HU-8 Consultar puntuación

Historia de usuario			
Número	9	Nombre	Selector de dificultad
Descripción			
Como usuario **quiero** poder cambiar la dificultad de las partidas **para** que se pueda ajustar a mi nivel.			
Prioridad			
C			

Tabla 10: HU-9 Selector de dificultad

Historia de usuario			
Número	10	Nombre	Tutorial
Descripción			
Como usuario **quiero** poder tener un tutorial del juego seleccionado **para** aprender a jugar.			
Prioridad			
S			

Tabla 11: HU-10 Tutorial

Historia de usuario			
Número	11	Nombre	Multijugador C
Descripción			
Como usuario **quiero** poder jugar partidas contra otros jugadores **para** enfrentarme con ellos.			
Prioridad			
C			

Tabla 12: HU-11 Multijugador

Historia de usuario			
Número	12	Nombre	Estilos de carta W
Descripción			
Como usuario **quiero** poder cambiar la imagen de las cartas con las que estoy jugando **para** que se ajusten a mis gustos.			
Prioridad			
W			

Tabla 13: HU-21 Estilos de carta

Historia de usuario			
Número	13	Nombre	Diferentes tipos de barajas
Descripción			
Como usuario **quiero** poder jugar juegos con distintos tipos de barajas **para** tener más variedad de juegos.			
Prioridad			
C			

Tabla 14: HU-13 Diferentes tipos de barajas

Historia de usuario			
Número	14	Nombre	Reloj de arena
Descripción			
Como usuario **quiero** poder tener un límite de tiempo de turno **para** controlar que el rival no se quede AFK (Away from Keyboard) o pierda el tiempo.			
Prioridad			
C			

Tabla 15: HU-14 Reloj de arena

Historia de usuario			
Número	15	Nombre	Top puntuaciones
Descripción			
Como usuario **quiero** poder ver las mejores puntuaciones de un juego **para** compararme con los otros jugadores.			
Prioridad			
W			

Tabla 16: HU-15 Top puntuaciones

2.2 Análisis y diseño del sistema

Este proyecto se realizará con una metodología ágil (Kanban), por lo que a lo largo de las iteraciones se irá creando y modificando el diseño del sistema.

La arquitectura elegida será una basada en microservicios, debido a que el principal objetivo del proyecto es que terceras personas puedan añadir nuevos juegos en forma de microservicios para ir aumentando la biblioteca que hay en la aplicación. Este proyecto será, por tanto, una base para que se satisfagan las principales necesidades para que otros desarrolladores puedan adjuntar sus microservicios a este proyecto e ir haciendo la aplicación más completa.

Al usar microservicios, se ha de dividir la carga del programa en diferentes programas más pequeños para adaptarse a la arquitectura [2].

En este proyecto se aprovechará la división de microservicios para crear tipos de microservicios, dotando de una estructura ya estandarizada a los microservicios que vengan en versiones posteriores, de modo que se alineen con las funcionalidades dictadas a la estructura ya creada.

Para poder identificar mejor los tipos de microservicios, se les ha asignado un color distintivo a cada tipo (ver imágenes 12 y 14).

- Persistencia: estos microservicios sólo se encargarán de almacenar datos y proporcionar datos a otros microservicios (verde).

- Lógica de negocio: estos microservicios sólo se encargarán de procesar los datos que le lleguen de entrada para generar la salida, es decir, son los microservicios que harán todas las operaciones necesarias para que funcione la lógica de la aplicación (azul).

- Interfaces: estos microservicios sólo se encargarán de proveer al usuario con una interfaz con la cual pueda comunicarse con el sistema (rojo).

Al hacer esta distinción, ayudamos a delimitar claramente las diferentes capas de la aplicación, facilitando observar la arquitectura interna del sistema además de dar plantillas para crear microservicios futuros.

Como se ha descrito anteriormente, se crearán microservicios exclusivos para las interfaces de usuario, dando lugar a la posible creación de distintas interfaces de usuario para la misma funcionalidad del programa, dotando a este una facilidad de trasladar los distintos juegos creados a distintas plataformas.

Por último, una de las partes más importantes de los microservicios radica en cómo se comunican entre ellos. En este proyecto se ha decidido optar por la creación de APIs [4] que manejan datos en tipo JSON[20], teniendo en la documentación de cada microservicio qué datos de entrada necesita y qué datos de salida genera. Se puede ver la documentación de cada microservicio en su correspondiente carpeta del github.

[20] https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON

3 DESARROLLO

Descripción en detalle del proceso del desarrollo de cada una de las iteraciones que se han realizado en este proyecto.

3.1 Primera Iteración

Historias de usuario elegidas

En esta primera iteración he decidido realizar las historias de usuario mínimas para producir el PMV (producto mínimo viable) a partir del cual hacer evolucionar el programa. Por ello, sólo se ha realizado la historia de usuario BlackJack puesto que constituye el PMV.

CÓDIGO	NOMBRE	Descripción	PRIORIDAD
HU-4	BLACKJACK	**Como** usuario **quiero** poder jugar al BlackJack contra la máquina **para** probar un juego con baraja francesa	M

Tabla 17: historias de usuario elegidas en esta iteración

Criterios de validación

Una vez descrita la historia de usuario, se han definido los criterios de validación:

- Poder iniciar el Juego
- Poder elegir entre robar carta y pasar
- Poder ver la puntuación acumulada
- Ver quién ha ganado la ronda
- Poder indicar que se desea jugar otra ronda más

Diseño de la Interfaz de Usuario

Una vez definidos los criterios de validación, se ha hecho un boceto (que se refleja en la imagen 5) de la interfaz de usuario para satisfacer estos criterios de validación.

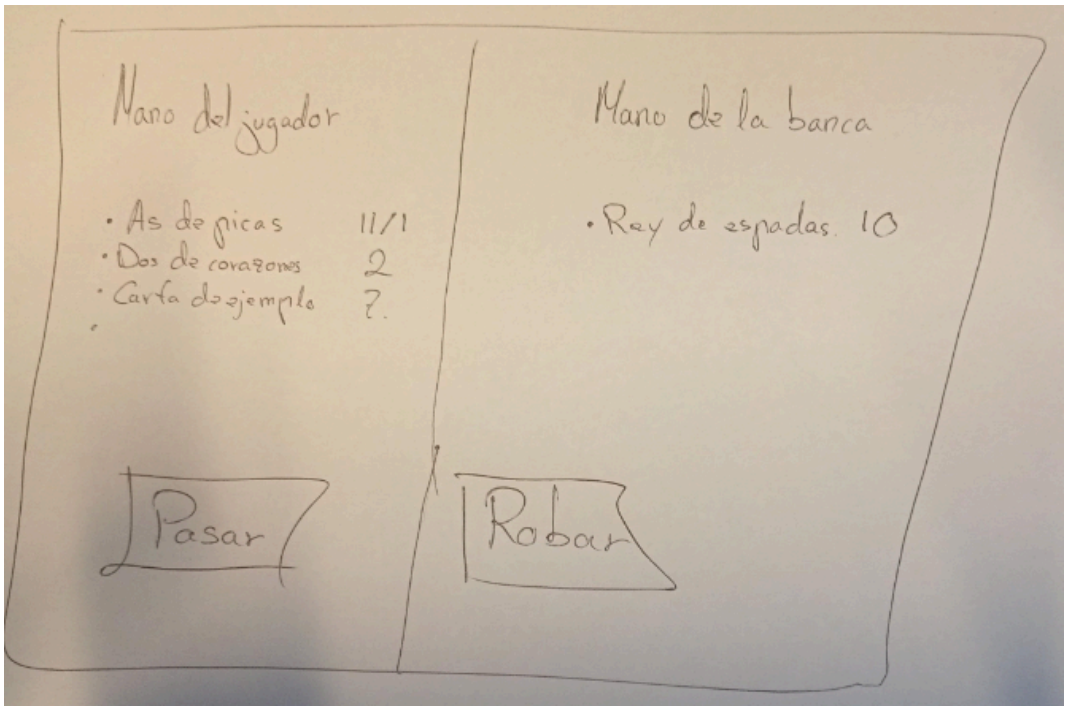


Imagen 7 : Diseño de la interfaz de Usuario de la HU-4

Se pueden observar dos partes: la parte izquierda que muestra la mano del jugador con los puntajes de cada carta y debajo de ella el botón para terminar el juego; y la parte derecha, en la que se alojaría la mano de la banca con los puntajes de sus cartas, con el botón para robar otra carta debajo.

Diagrama de secuencia de la HU

Luego se ha realizado el diagrama de secuencia (mostrado en la imagen 6) para definir las diferentes comunicaciones entre las clases y cómo se ha de realizar la comunicación con el usuario.

Podemos ver que primero el microservicio de **blackjack** pide la baraja al microservicio msBaraja, luego interacciona con el usuario hasta que este deja de robar cartas y, cuando le dé al botón de pasar el microservicio msBlackJack calculará los puntos para decidir y devolver quién es el ganador de la partida.

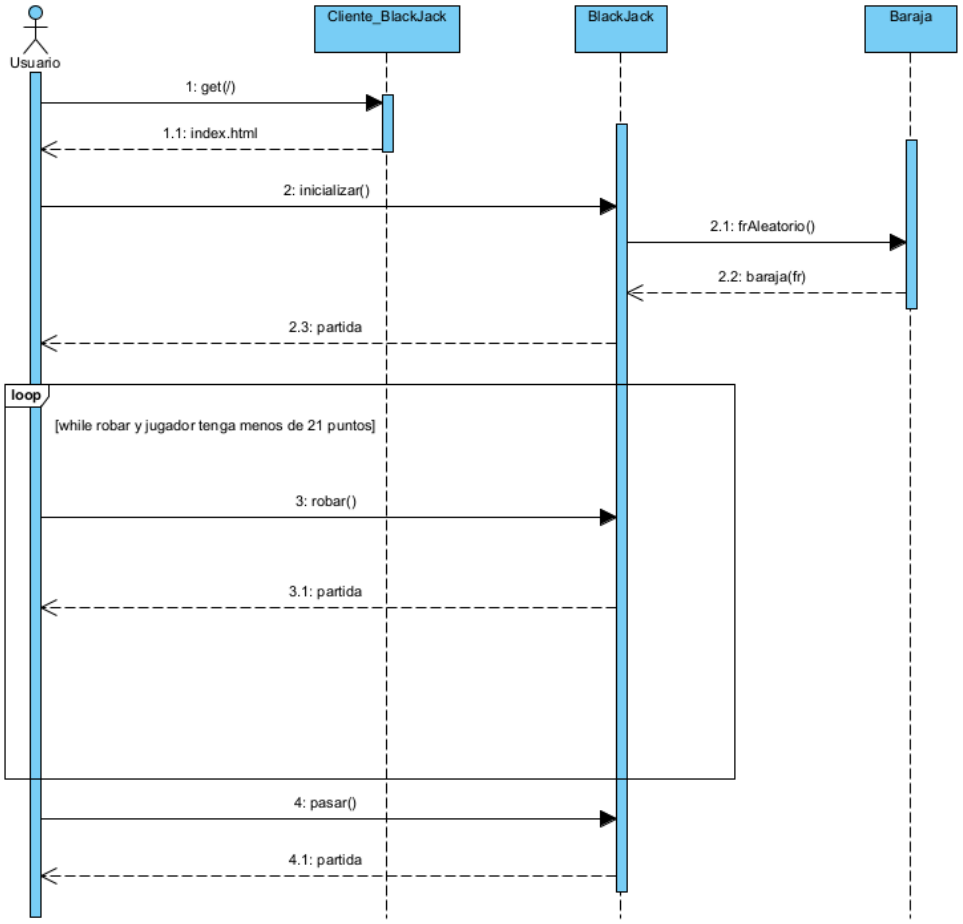


Imagen 8: Diagrama de secuencia de la HU-4

Implementación

En esta primera iteración he creado 3 microservicios para satisfacer las funcionalidades deseadas. En la siguiente imagen (imagen 7) se describe cómo se comunican entre sí los distintos microservicios.

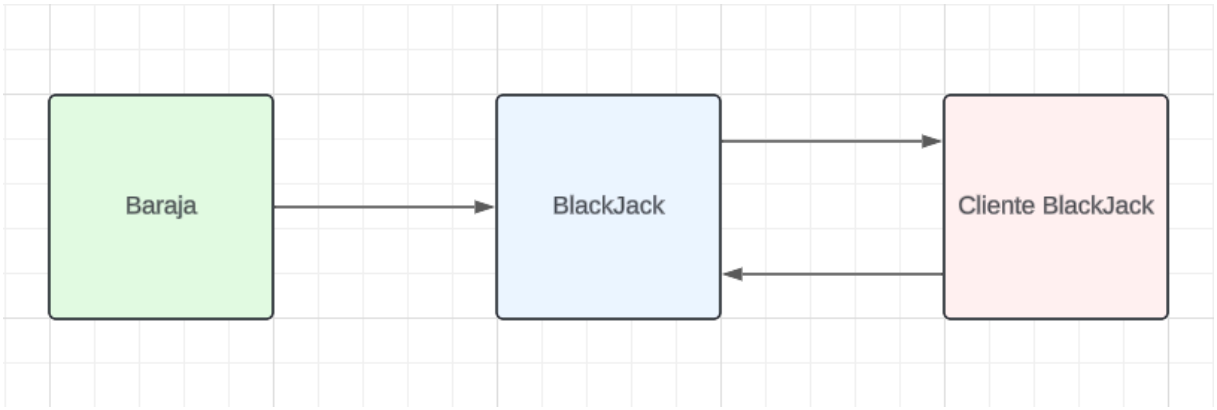


Imagen 9: Diagrama de flujo de datos entre microservicios

- Baraja: nos proporciona la propia baraja de cartas, pudiendo solicitarla que esté mezclada aleatoriamente. En este primer incremento, solo se dispone de la baraja francesa.

  Este microservicio no necesita ningún parámetro de entrada debido a que su naturaleza es proporcionar el objeto baraja de forma diferente según el endpoint al que se llame. El objeto baraja será una colección de objetos carta, como se muestra en la imagen 10, que se conforman de dos parámetros: un id y un nombre, los dos parámetros de tipo cadena de caracteres.

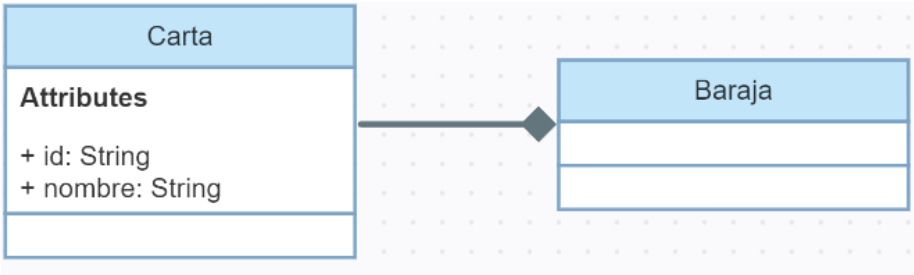


Imagen 10: Diagrama de clases de los objetos del microservicio

Se han creado dos endpoint [6] para este microservicio:

- ➔ **get /fr**: nos proporciona un objeto baraja conteniendo todas las cartas de la baraja francesa de manera ordenada por palos.
- ➔ **get /fr/aleatorio**: nos proporciona un objeto baraja conteniendo todas las cartas de la baraja francesa ordenadas de forma aleatoria.

- BlackJack: microservicio que realiza toda la lógica del juego ya sea inicializando la partida desde 0, robando cartas para el jugador o pasando para terminar la partida.

  Este microservicio necesita que se le pase el contexto de la partida para que pueda procesar los datos. El contexto que necesita es: el estado de la baraja en esa partida, la mano del jugador y la mano de la banca. Los tres parámetros de entrada se recibirán como objetos baraja, es decir, colecciones de objetos carta.

Como salida, el microservicio genera otra clase partida (imagen 11) que consta de los siguientes elementos: tres colecciones de objetos simulando la baraja de la partida, la mano del jugador y la mano de la banca; y tres valores numéricos que representan la puntuación del jugador, la puntuación de la banca y el ganador (si hay) en ese momento de la partida.

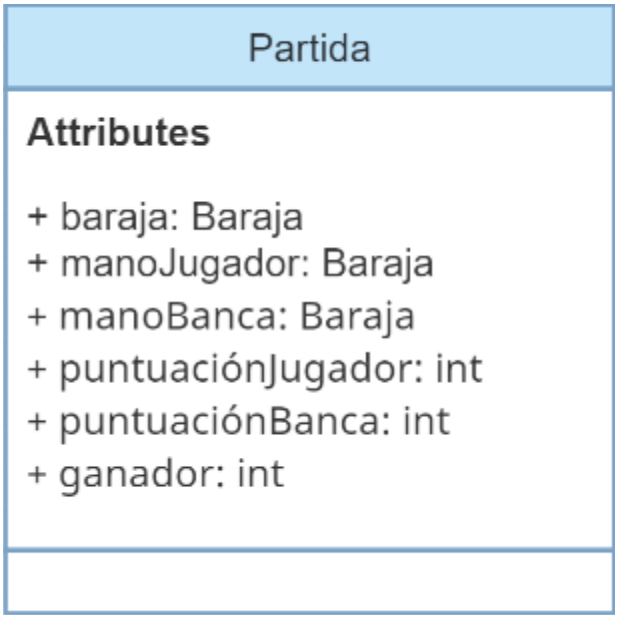


Imagen 11: Diagrama de clases del objeto partida de salida

Se han creado tres endpoint [6] para este microservicio:

- ➔ **get** */*: este endpoint no necesita ningún parámetro de entrada. Contacta con el microservicio msBaraja para obtener el objeto baraja inicial; el cuál permitirá hacer las operaciones necesarias para crear el estado inicial de la partida. Devuelve un objeto partida con el estado inicial del juego.

➔ **post /robar**: este endpoint necesita los parámetros de entrada definidos con anterioridad para funcionar. Realiza las acciones que conlleva que el jugador robe una carta. Devuelve un objeto partida con el estado correspondiente después de robar una carta por parte del jugador.

➔ **post /pasar**: este endpoint necesita los parámetros de entrada definidos con anterioridad para funcionar. Realiza las acciones que conlleva que el jugador pase de ronda. Devuelve un objeto partida con el estado final de la partida.

- Cliente_BlackJack: este microservicio solo contiene la interfaz necesaria para poder comunicarse con los otros microservicios y jugar al juego. A través de él sólo se genera la interfaz web.
  Este microservicio sólo contiene un endpoint [6]:

  ➔ **get /**: devuelve el html que contiene la interfaz de usuario web correspondiente a su juego.

  Dentro de la interfaz de usuario no se encuentra ningún tipo de lógica de negocio, pues todas las acciones que se producen en la interfaz de usuario (los distintos botones que hay en ella) son llamadas al microservicio msBlackJack que se encarga de procesarlas.

  Al ser una interfaz de usuario, sólo guarda el contexto del juego que se está jugando en ese momento, para mostrárselo al usuario y poder mandarlo al microservicio msBlackJack cuando lo necesite.

Verificación de HU hecha

Para poder considerar la HU hecha, mostramos evidencias (en forma de captura de pantalla) del resultado final una vez implementado.

La principal diferencia con el diseño de la interfaz inicial vista en la imagen 12 es la posición del puntaje de las cartas, que en vez de estar al lado del nombre de las cartas de cada mano, sólo se mostrará el puntaje acumulado de cada una de las diferentes manos.

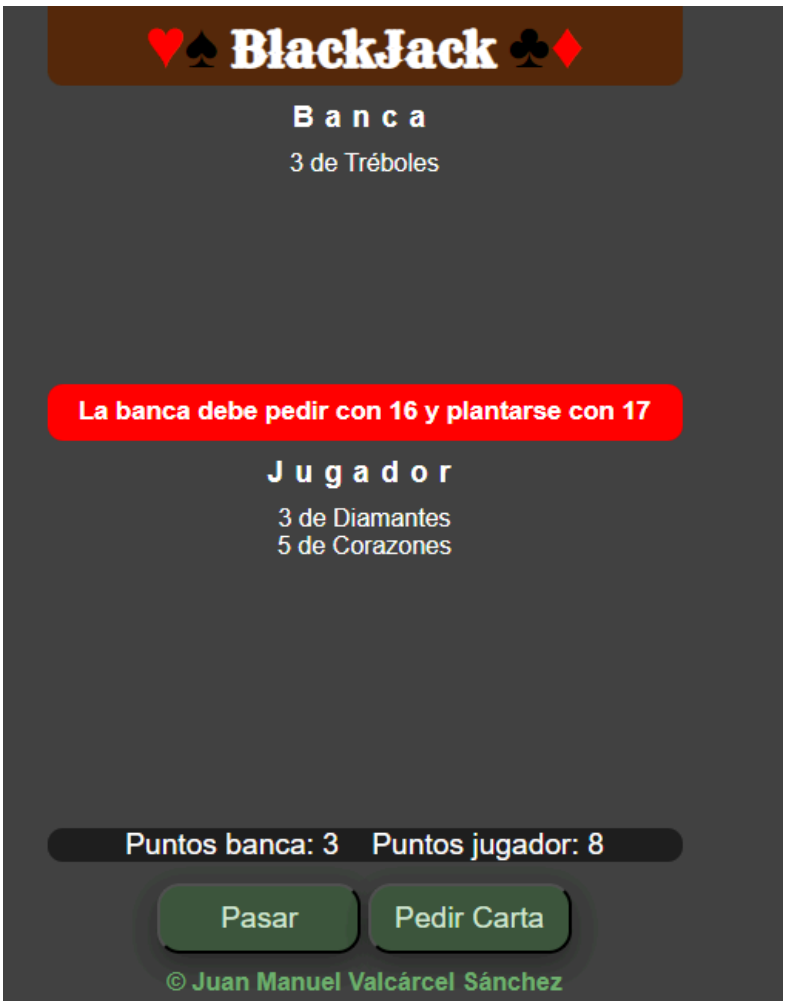


Imagen 12: Interfaz del juego BlackJack

En la imagen 13 se puede observar como el jugador puede pasar y pedir carta con los botones correspondientes; después del mensaje amarillo, indicando las acciones posibles, se muestran las puntuaciones que acumulan tanto la banca como el jugador. Para iniciar el juego sólo es necesario acceder a la página, iniciándose éste automáticamente al entrar.

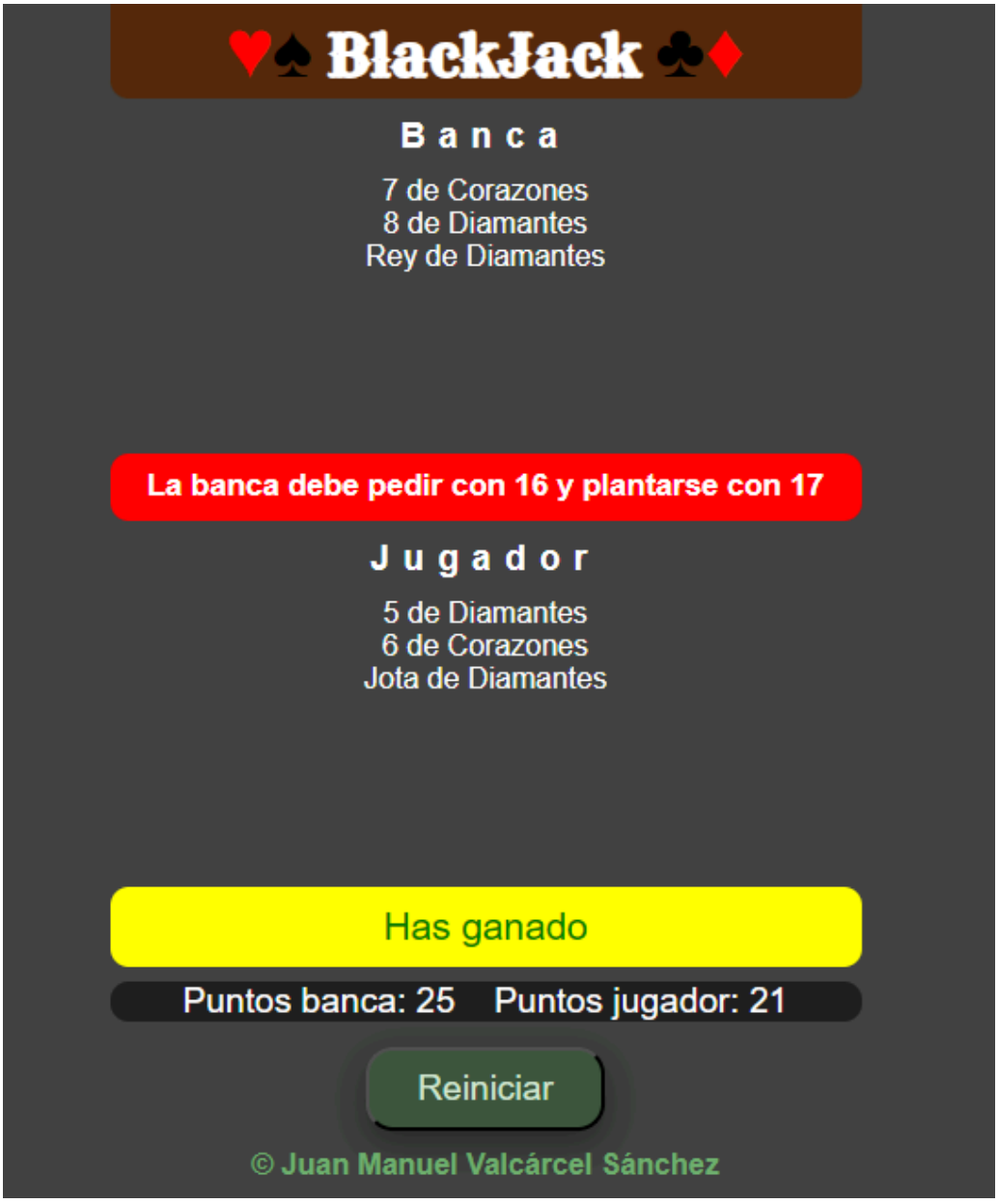


Imagen 13: Partida ganada por el jugador

En la imagen 13 podemos ver que en el mensaje con fondo amarillo muestra quién gana la partida (en este ejemplo es el jugador) y cómo solo esta disponible el botón para volver a jugar otra partida, el botón reiniciar.

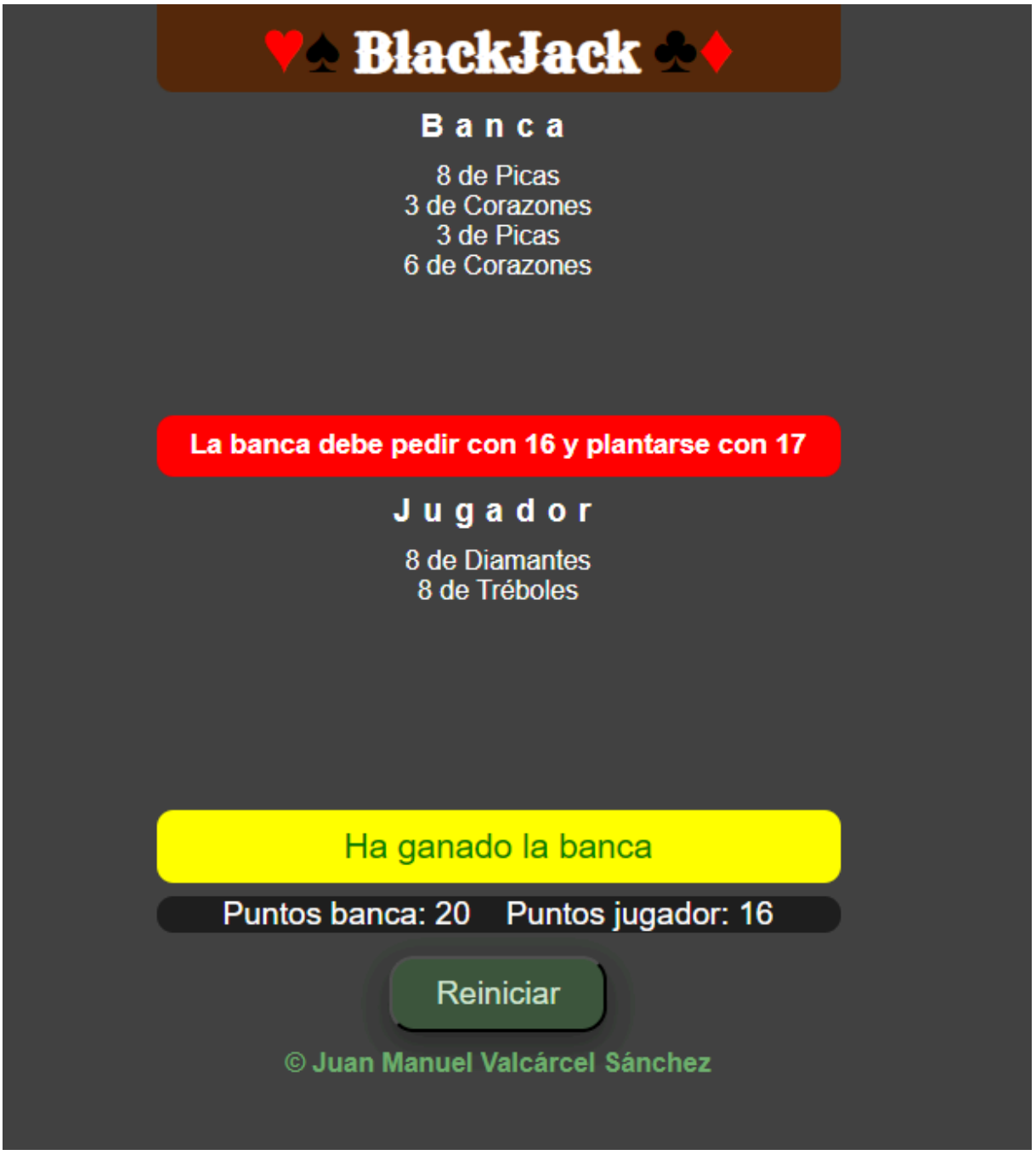


Imagen 14: Partida ganada por la banca

En la imagen 14 se muestra un caso en el que el jugador pierde la partida.

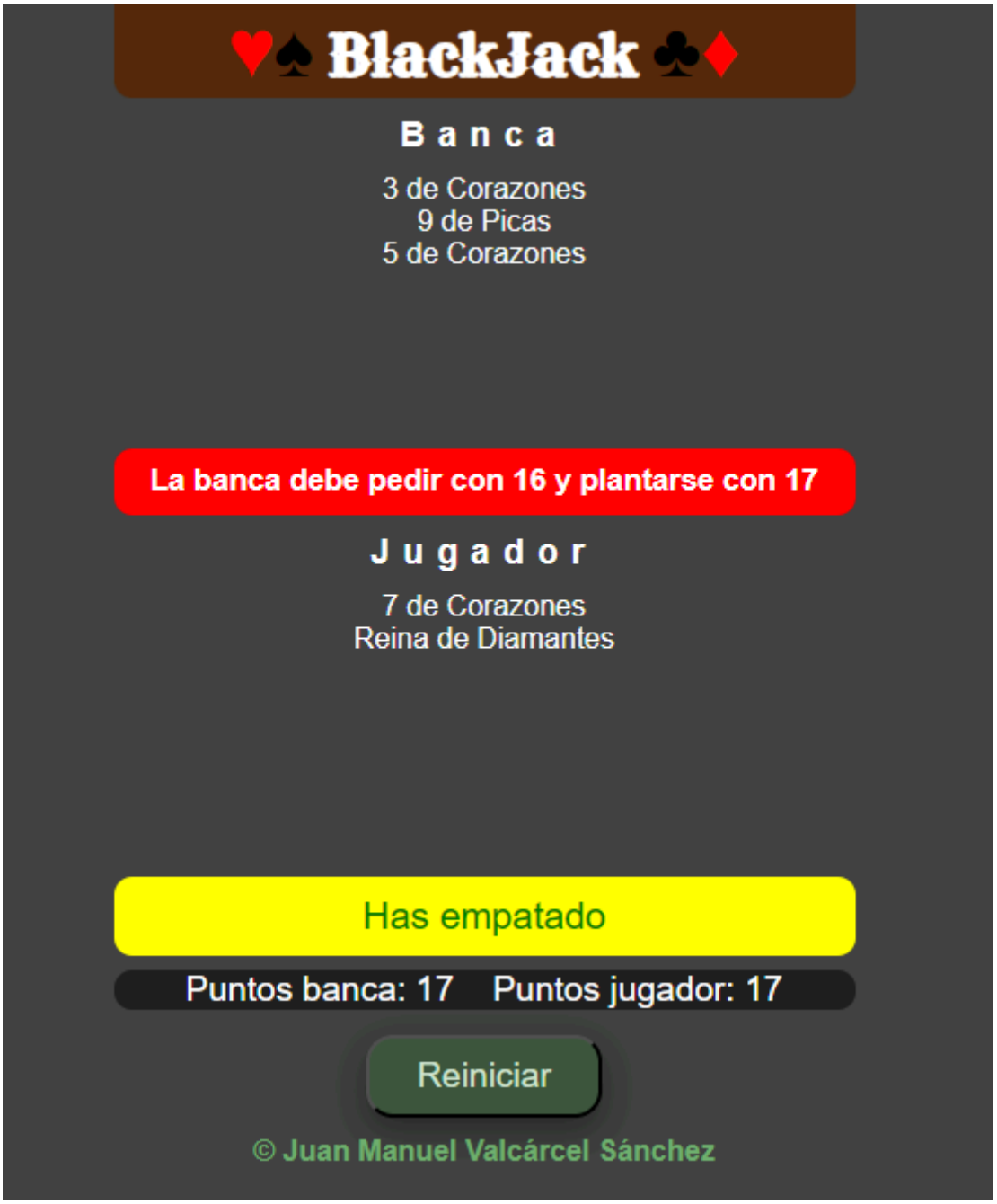


Imagen 15: Partida con resultado de empate

En la imagen 15 se muestra un caso en el que el jugador empata con el crupier.

Validación y feedback

Para validar el incremento generado, se ha recurrido a un conjunto de terceros para que comprueben si se cumplen los criterios de validación, además de recoger sus comentarios para mejorar el programa. Siendo estos usuarios de prueba personas de edades dispares y con núcleos de conocimiento distintos.

La validación ha sido exitosa teniendo todos los criterios de validación marcados como realizados por todos los comprobadores.

Se encontraron problemas a la hora de hacer llegar el programa a los usuarios probadores puesto que se tuvo que hacer yendo uno por uno y en físico. Para las siguientes iteraciones se debería poder realizar telemáticamente y con varios usuarios al mismo tiempo.

Con respecto al feedback hemos tenido varias peticiones de los usuarios que podrían convertirse en historias de usuario:

- Mostrar cartas por imágenes en vez de por texto.
- Enfatizar el resultado final de la partida.
- Mostrar reglas del juego.
- Diferenciar textos de ganar y perder la partida.

Debido a estas peticiones se han creado más historias de usuario para satisfacer el feedback de los usuarios:

Historia de usuario			
Número	16	Nombre	Imagenes de cartas
Descripción			
Como usuario quiero **poder** jugar ver las cartas por imagen **para** ayudar a visualizar el tipo de carta.			
Prioridad			
S			

Tabla 18: HU-16 Imágenes de cartas

Historia de usuario			
Número	17	Nombre	Conexión Online
Descripción			
Como usuario quiero **poder** jugar desde cualquier dispositivo **para** jugar donde yo quiera.			
Prioridad			
M			

Tabla 19: HU-17 Conexión online

Historia de usuario			
Número	18	Nombre	Siete y medio
Descripción			
Como usuario **quiero** poder jugar al Siete y medio contra la máquina **para** probar un juego con baraja española			
Prioridad			
S			

Tabla 20: HU-18 Siete y medio

Historia de usuario			
Número	19	Nombre	Clarificar UI
Descripción			
Como usuario **quiero** poder distinguir perfectamente el resultado de la partida **para** poder apreciar el resultado sin dificultad.			
Prioridad			
C			

Tabla 21: HU-19 Clarificar UI

También hemos tomado en cuenta el feedback obtenido con referencia a las historias de usuario escogidas con anterioridad, aumentando la prioridad de las HU más solicitadas:

Historia de usuario			
Número	10	Nombre	Tutorial
Descripción			
Como usuario **quiero** poder tener un tutorial del juego seleccionado **para** aprender a jugar.			
Prioridad			
W			

Tabla 22: HU-10 Tutorial

3.2 Segunda Iteración

Historias de usuario elegidas

En esta segunda iteración he decidido realizar las siguientes historias de usuario, mostradas en la tabla 3, para comprobar las ventajas de poder reutilizar microservicios.

Código	Nombre	Descripción	Prioridad
HU-5	Elegir Juego	**Como usuario** **quiero** poder elegir qué juego jugar **para** elegir que quiero jugar.	M
HU-6	Mazo aleatorio	**Como usuario** **quiero** poder aleatorizar el mazo de juego **para** diferenciar cada partida.	S
HU-13	Diferentes tipos de barajas	**Como usuario** **quiero** poder jugar juegos con distintos tipos de barajas **para** tener más variedad de juegos.	C
HU-17	Conexión Online	**Como usuario** **quiero** poder jugar desde cualquier dispositivo **para** jugar donde yo quiera.	M
HU-18	Siete y medio	**Como usuario** **quiero** poder jugar al Siete y medio contra la máquina **para** probar un juego con baraja española	S

Tabla 23: HU escogidas en la segunda iteración

Criterios de validación

En este caso, la iteración consiste en más de una historia de usuario, por ello he agrupado los criterios de validación en la tabla 4 según la historia de usuario a la que corresponden.

Código	Nombre	Criterios de validación
HU-5	Elegir Juego	-Poder ver los distintos juegos disponibles para jugar. -Poder discernir con facilidad que juegos puede el usuario elegir. -Poder escoger qué juego jugar de los disponibles.
HU-6	Mazo aleatorio	-Al jugar partidas distintas, las cartas que proporciona la aplicación son diferentes.
HU-13	Diferentes tipos de barajas	-Un juego que utilice y muestre la baraja francesa. -Un juego que utilice y muestre la baraja española.
HU-17	Conexión Online	-Poder acceder de manera online a la aplicación.
HU-18	Siete y medio	-Poder iniciar el Juego -Poder elegir entre robar carta y pasar -Poder ver la puntuación acumulada -Ver quién ha ganado la ronda -Poder indicar que se desea jugar otra ronda más

Tabla 24: Criterios de validación asociados a cada HU

Diseño de la Interfaz de Usuario

Para esta iteración las únicas HU con interfaz de usuario son la HU-5 y la HU-18.
En el caso de la HU-18 (Siete y medio) se ha elegido imitar la interfaz de usuario ya creada para el juego de Blackjack, debido a su similitud.

En el caso de la HU-5 (Elegir juego) la interfaz de usuario está dispuesta en la imagen 16.

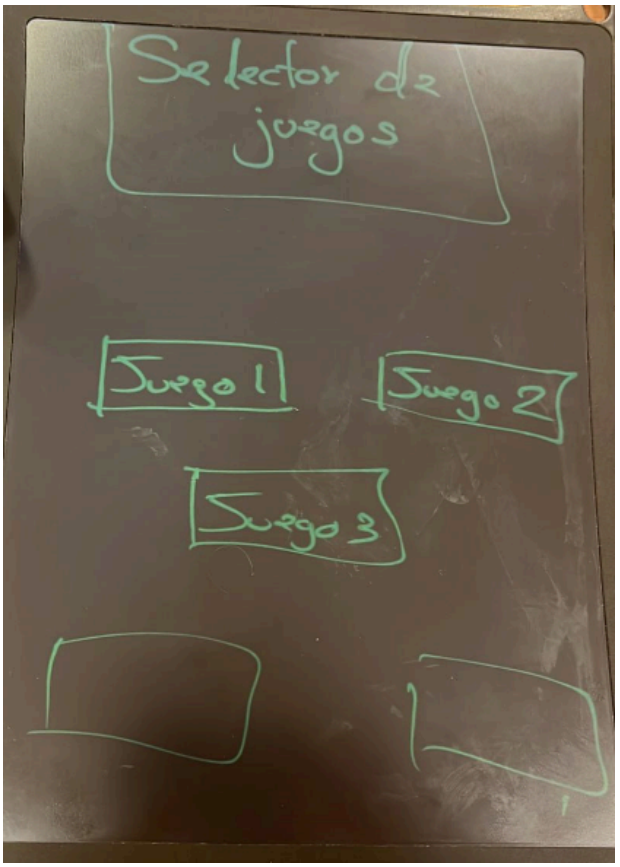


Imagen 16: Diseño preliminar del selector de juegos

En esta UI observamos en la parte superior de la pantalla el título que representa al selector de juegos para enfatizar lo que se está viendo en ese momento.

En la parte inferior observamos distintos botones que representan los distintos juegos de la aplicación, dándonos acceso a ellos desde esta interfaz pulsandolos.

Diagrama de secuencia de la HU

El diagrama de secuencia de la aplicación no cambia demasiado, puesto que solo se debe añadir el proceso de selección de juego antes de jugar a algún juego.

Por ello en el diagrama de la imagen 17 solo se representa la interacción con el microservicio SelectorJuegos, debido a que la interacción posterior a ésta está definida en diagrama de secuencia ya creado (se referencia en las notas del diagrama).

Imagen 17: Diagrama de secuencia del sistema

Implementación

En la imagen 18 se muestra cómo se comunican los distintos microservicios, pudiendo observar la capacidad de un microservicio de administrar funcionalidades a más de otro microservicio.

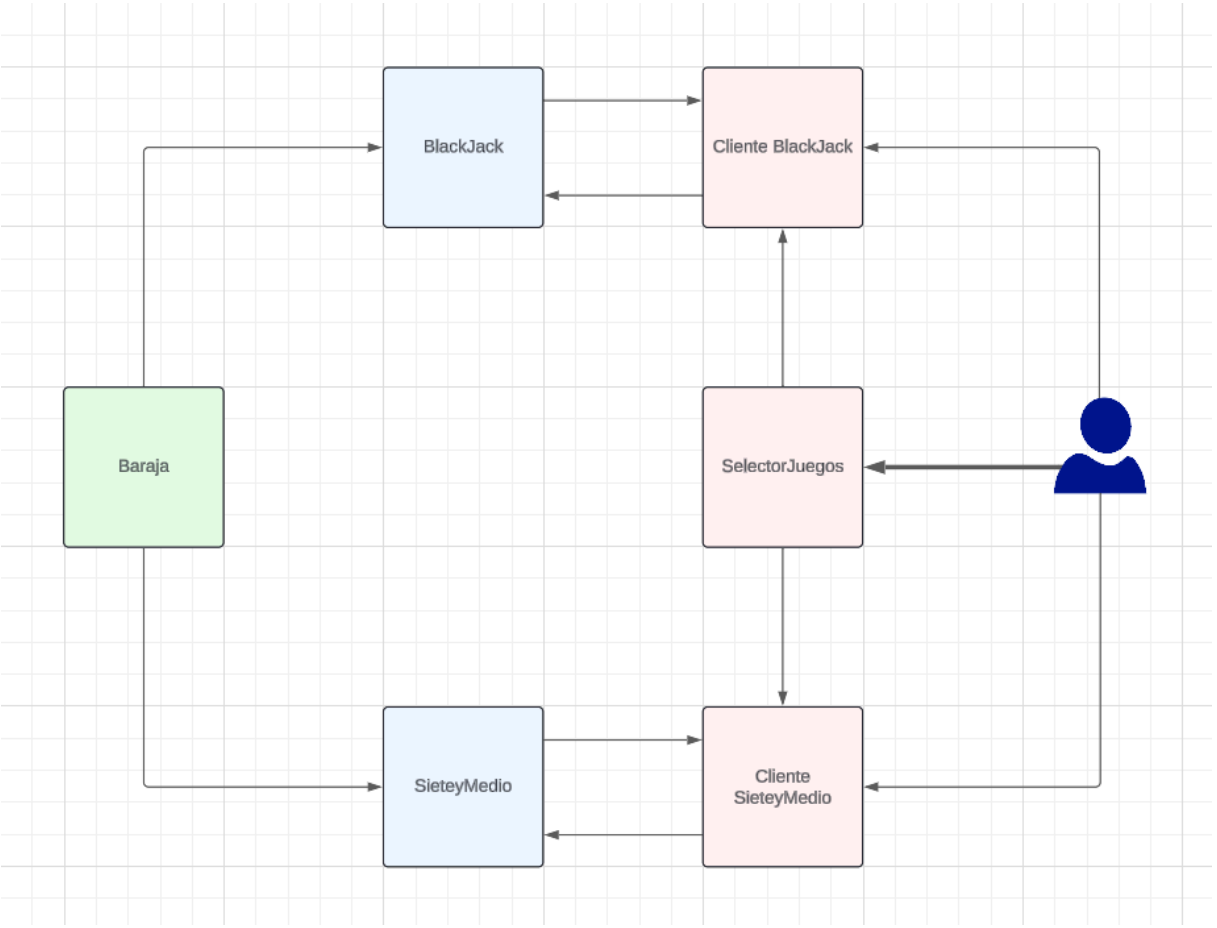


Imagen 18: Diagrama de flujo de datos entre microservicios

Ahora se describirán las diferentes implementaciones de esta iteración agrupadas por la HU a la que satisfacen:

HU-05: Elegir Juego

Se ha creado un nuevo microservicio que proporciona una página estática que muestra botones con enlaces a los distintos juegos que se pueden elegir.

Este microservicio sólo contiene un endpoint [6]:

- ➔ **get** */*: devuelve el html que contiene la interfaz de usuario web correspondiente al selector de juegos.

```html
<!DOCTYPE HTML>
<html lang="es">

<head>
  <title>BlackJack</title>
  <meta charset="utf-8">
  <meta name="author" content="Juan Manuel Valcárcel Sánchez">
  <meta name="description" content="Selector de Juegos">
  <meta name="keywords" content="Selector de Juegos">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <!-- Fuente título -->
  <link href="https://fonts.googleapis.com/css?family=Rye" rel="stylesheet">
  <link rel="stylesheet" href="CSS/index.css">
  <script src="js/index.js"></script>
</head>
<body>
  <noscript>
    <p>La página que estás viendo requiere para su funcionamiento el uso de
      JavaScript. Si lo has deshabilitado intencionadamente, por favor vuelve a
      activarlo.
    </p>
  </noscript>
  <div id="marco_exterior">
    <header id="cabecera">
      <h1>
        Selector de juegos
      </h1>
    </header>
    <div id="botones">
      <button id="pasar" onclick="window.location.href='https://vg3jdmms-8000.uks1.devtunnels.ms/';">BlackJack</button>
      <button id="pedir_carta" onclick="window.location.href='https://vg3jdmms-8004.uks1.devtunnels.ms/';">Siete y medio</button>
    </div>
    <footer id="pie">
      &copy; Juan Manuel Valcárcel Sánchez
    </footer>
  </div>
</body>
</html>
```

Imagen 19: UI del Selector de juegos

Como se puede observar en la imagen 19, los botones de este microservicio se limitan a redirigirlos a los diferentes microservicios que contienen las UI de los diferentes juegos mediantes los endpoint correspondientes.

HU-06: Mazo Aleatorio

En esta HU se ha recreado la forma de que el microservicio msBaraja devuelva los objetos baraja con un orden aleatorio. Para ello, se han creado endpoints con la terminación **/aleatorio**. Estos endpoint, luego de obtener el tipo de baraja correspondiente a ellos, ordenarán aleatoriamente el contenido de los objetos baraja que obtengan para devolverlos después de la aleatorización.

Hay dos endpoints [6] de este tipo:

- ➔ **get /fr/aleatorio**
- ➔ **get /es/aleatorio**

HU-13: Diferentes tipos de barajas

El BlackJack utiliza la baraja francesa que ya está implementada, y para la baraja española en esta misma iteración se creará el juego del siete y medio que se definirá en su historia de usuario (HU-18).
Para poder dar servicio a este nuevo microservicio se crearán nuevos endpoints [6] en el microservicio msBaraja:

- ➔ **get /es**: nos proporciona un objeto baraja conteniendo todas las cartas de la baraja española de manera ordenada por palos.

- ➔ **get /es/aleatorio**: nos proporciona un objeto baraja conteniendo todas las cartas de la baraja española ordenadas de forma aleatoria.

HU-17: Conexión online

Para que se pueda acceder a la aplicación de forma on-line se ha utilizado el mecanismo que nos proporciona el mismo Visual Studio Code para reenviar puertos[21]:

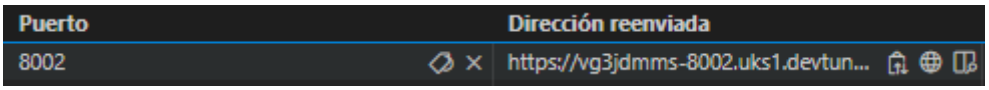


Imagen 20: Reenvío de puertos de VS Code

Este mecanismo nos permite tener una url que siempre es la misma, lo que nos permite que los usuarios accedan a la aplicación por los distintos microservicios, y a los propios microservicios que se puedan comunicar entre ellos también con las url proporcionadas.

```
const RUTA_MS_SIETEYMEDIO = "https://vg3jdmms-8003.uks1.devtunnels.ms/";
```

Imagen 21: Url estática

HU-18: Siete y medio

Para la realización de esta HU [5], realizamos el mismo proceso que utilizamos para el BlackJack, creamos dos microservicios, uno para la lógica del juego y uno que nos proporciona la interfaz de usuario.

[21] https://docs.github.com/es/codespaces/developing-in-a-codespace/forwarding-ports-in-your-codespace

❖ **Microservicio con la lógica de negocio**

Este microservicio es muy similar al microservicio msBlackJack debido a que la lógica de negocio es semejante. Por ello, los parámetros de entrada necesarios para su funcionamiento son los mismos que los del microservicio msBlackJack.
Además, los valores de salida siguen siendo el mismo objeto partida que daba el microservicio msBlackJack. La principal diferencia existente entre los parámetros de entrada y salida, es el formato de los objetos baraja con los que trabaja el microservicio. En el caso de este microservicio, se usa la baraja española.

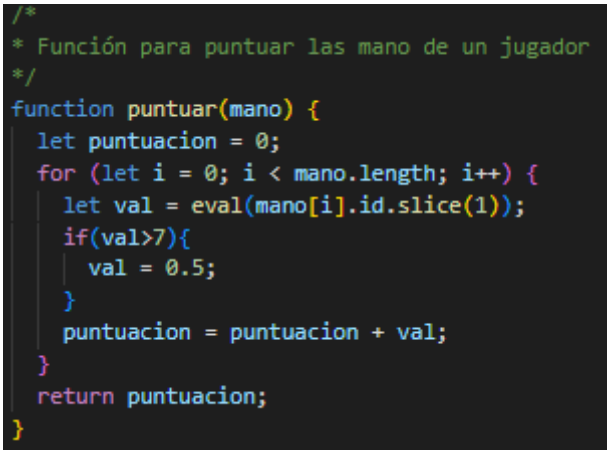
Se han creado tres endpoint [6] para este microservicio:

➔ **get /**: este endpoint no necesita ningún parámetro de entrada. Contacta con el microservicio msBaraja para obtener el objeto baraja inicial para hacer las operaciones necesarias para crear el estado inicial de la partida. Devuelve un objeto partida con el estado inicial del juego.

➔ **post /robar**: este endpoint necesita los parámetros de entrada definidos con anterioridad para funcionar. Realiza las acciones que conlleva que el jugador robe una carta. Devuelve un objeto partida con el estado correspondiente después de robar una carta por parte del jugador.

➔ **post /pasar**: este endpoint necesita los parámetros de entrada definidos con anterioridad para funcionar. Realiza las acciones que conlleva que el jugador pase de ronda. Devuelve un objeto partida con el estado final de la partida.

Además de estos endpoint, el microservicio cuenta con una función auxiliar para ayudar a conseguir la puntuación de una mano de cartas llamada **puntuar** a la que se le pasa una colección de objetos carta. Esta función se puede observar en la imagen 22.



```
/*
* Función para puntuar las mano de un jugador
*/
function puntuar(mano) {
  let puntuacion = 0;
  for (let i = 0; i < mano.length; i++) {
    let val = eval(mano[i].id.slice(1));
    if(val>7){
      val = 0.5;
    }
    puntuacion = puntuacion + val;
  }
  return puntuacion;
}
```

Imagen 22: Función que puntúa el valor de una mano

- **Microservicio con UI**

Este microservicio sólo contiene la UI del juego Siete y Medio, a través de él sólo se podrá obtener la UI web del juego.

Este microservicio sólo contiene un endpoint [6]:

- **get** */*: devuelve el html que contiene la interfaz de usuario web correspondiente al juego Siete y Medio.

Dentro de la interfaz de usuario no se encuentra ningún tipo de lógica de negocio, pues todas las acciones que se producen en la interfaz de usuario (los distintos botones que hay en ella) son llamadas al microservicio msSieteyMedio que se encarga de procesarlas. Esto se puede observar mejor en la imagen 23.

Al ser una interfaz de usuario, sólo guarda el contexto del juego que se está jugando en ese momento, para mostrárselo al usuario y poder mandarlo al microservicio msSieteyMedio cuando lo necesite.

```
    </div>
    <div id="botones">
      <button id="reiniciar" onclick="location.reload()" hidden>Reiniciar</button>
      <button id="pasar" onclick="Pasar();">Pasar</button>
      <button id="pedir_carta" onclick="PedirCarta();">Pedir Carta</button>
    </div>
    <footer id="pie">
      &copy; Juan Manuel Valcárcel Sánchez
    </footer>
  </div>
</body>
</html>
```

Imagen 23: index.html del Cliente Siete y Medio

Verificación de HU hecha

HU-05: Elegir Juego

Se muestra en la imagen 24 la interfaz para seleccionar los distintos juegos disponibles en la aplicación, siendo cada botón un juego distinto.

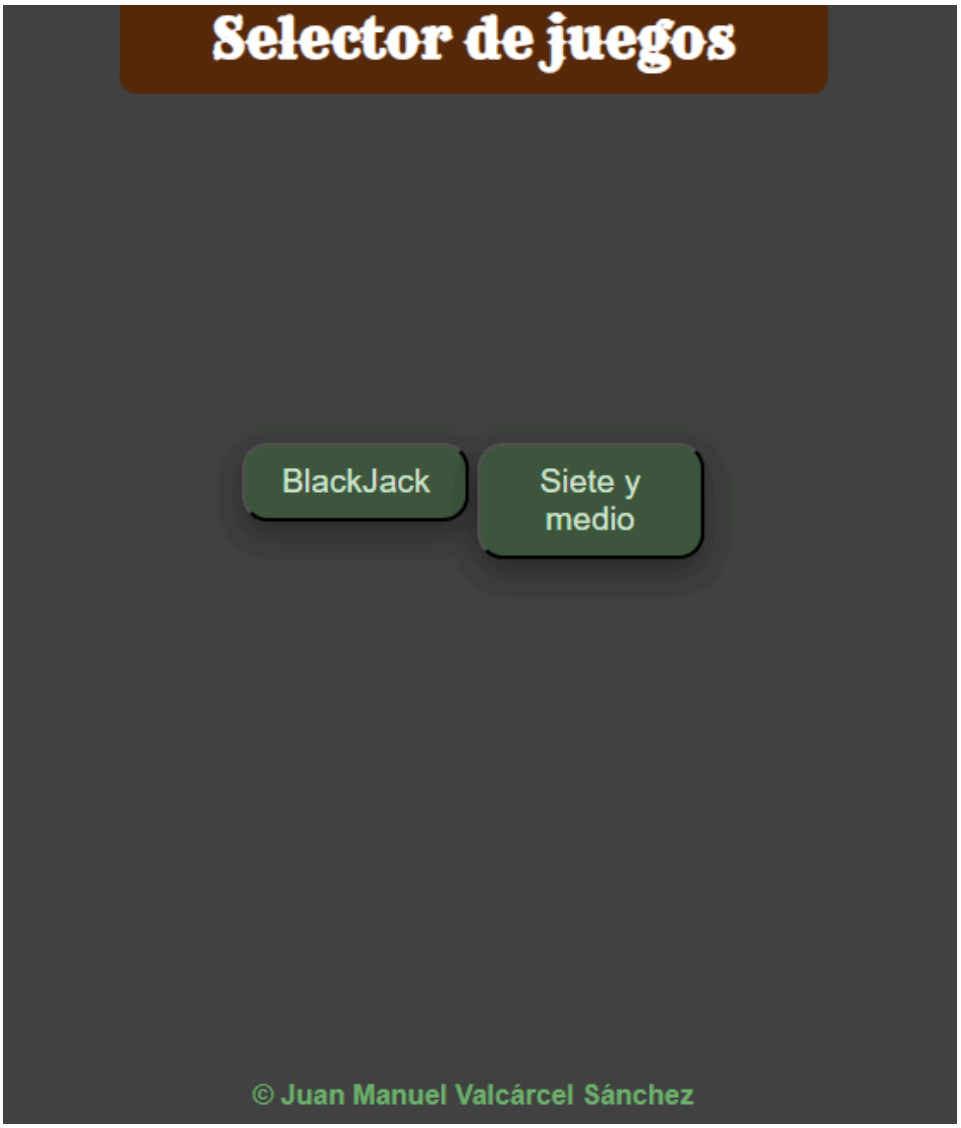


Imagen 24: UI del selector de juegos

HU-06: Mazo aleatorio

En las capturas de pantalla 25 y 26 se observa cómo en dos partidas distintas la asignación de cartas es aleatoria.

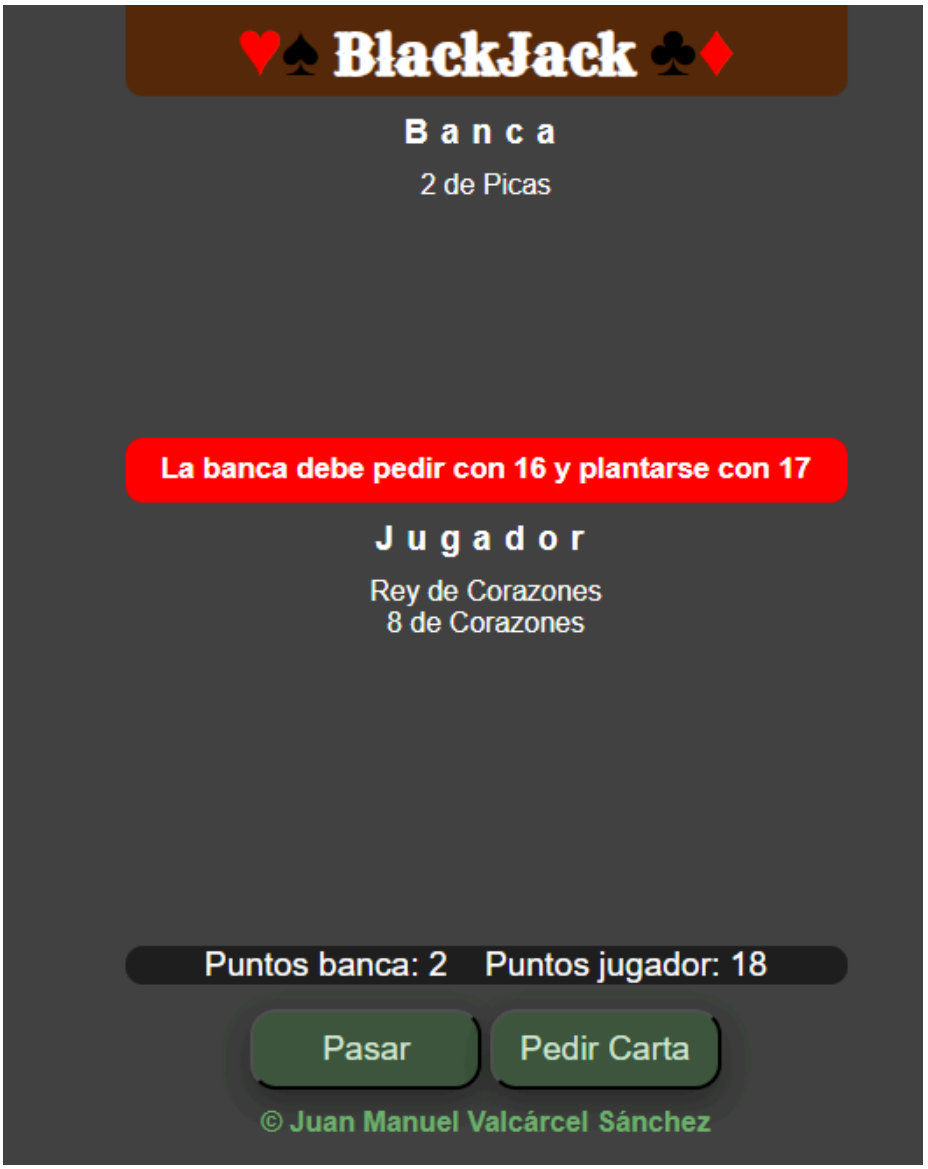


Imagen 25: Asignación inicial aleatoria 1

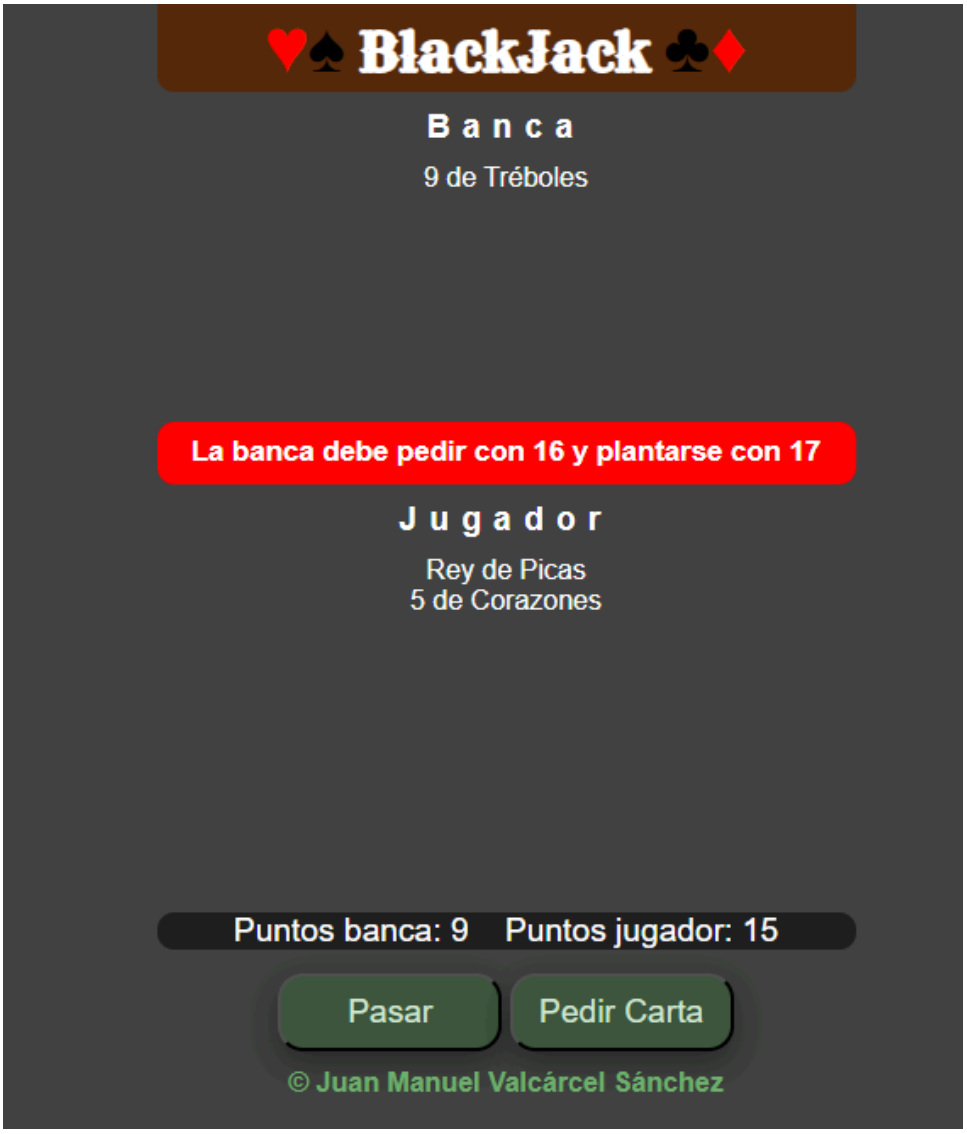


Imagen 26: Asignación inicial aleatoria 2

HU-13: Diferentes tipos de barajas

En las imágenes 27 y 28 se muestran dos juegos distintos que usan a su vez la baraja francesa y la baraja española respectivamente.

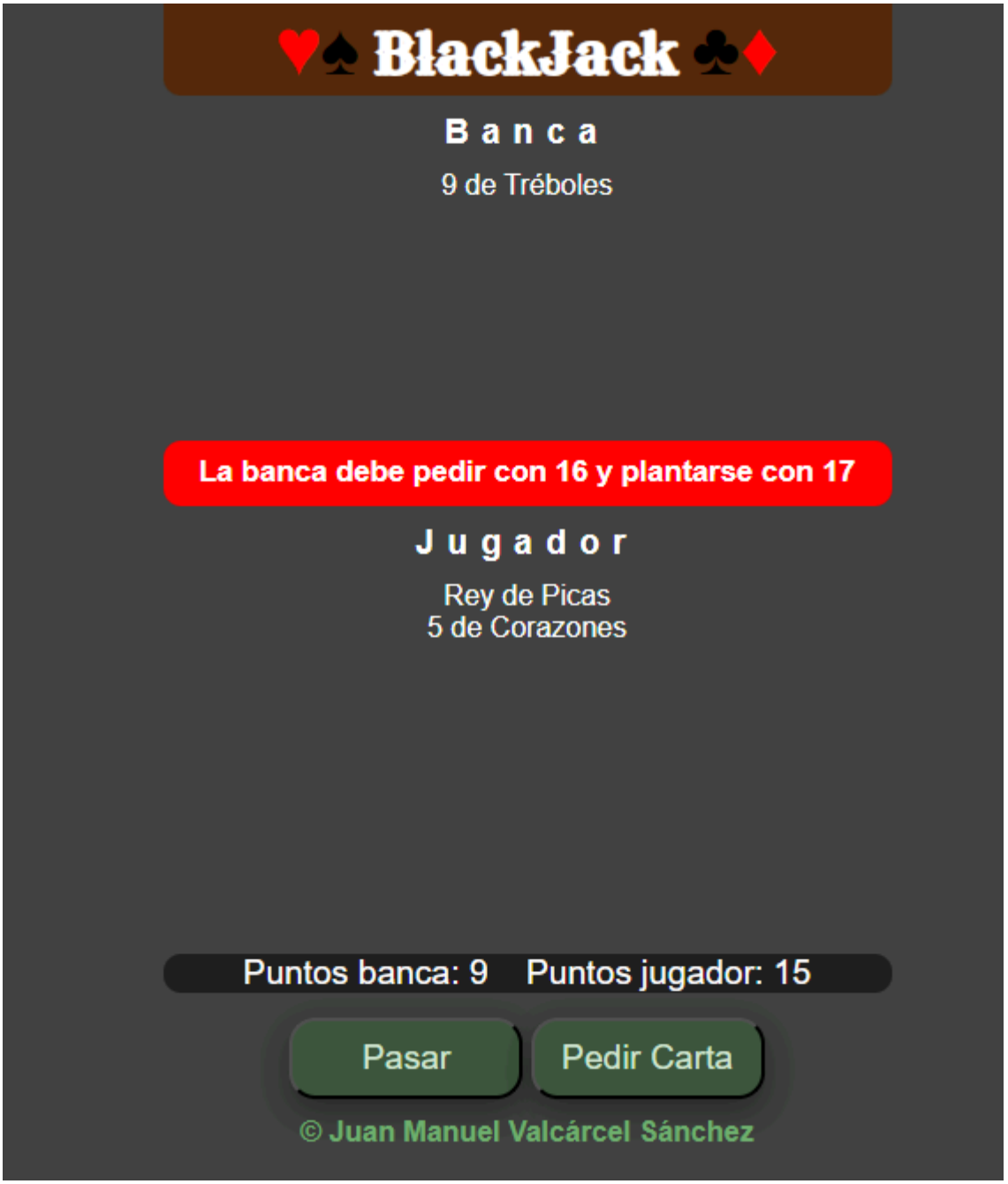


Imagen 27: UI del BlackJack

Imagen 28: UI del siete y medio

HU-17: Conexión Online

Se puede observar cómo accedemos al microservicio a través de una url estática remota en la imagen 29.

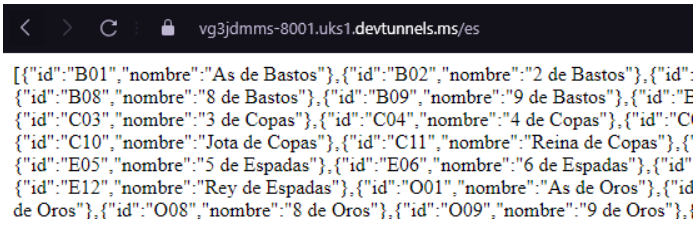


Imagen 29: Acceso a los microservicios remotamente a través del sistema de reenvío de puertos que incluye VSCode

HU-18: Siete y medio

En la imagen 30 se muestra la interfaz general del juego Siete y medio.

Imagen 30: Interfaz del juego siete y medio

En la imagen 31 se muestra cómo se perdió una partida.

Imagen 31: Partida ganada por la banca

En la imagen 32 se muestra cómo el jugador ganó una partida.

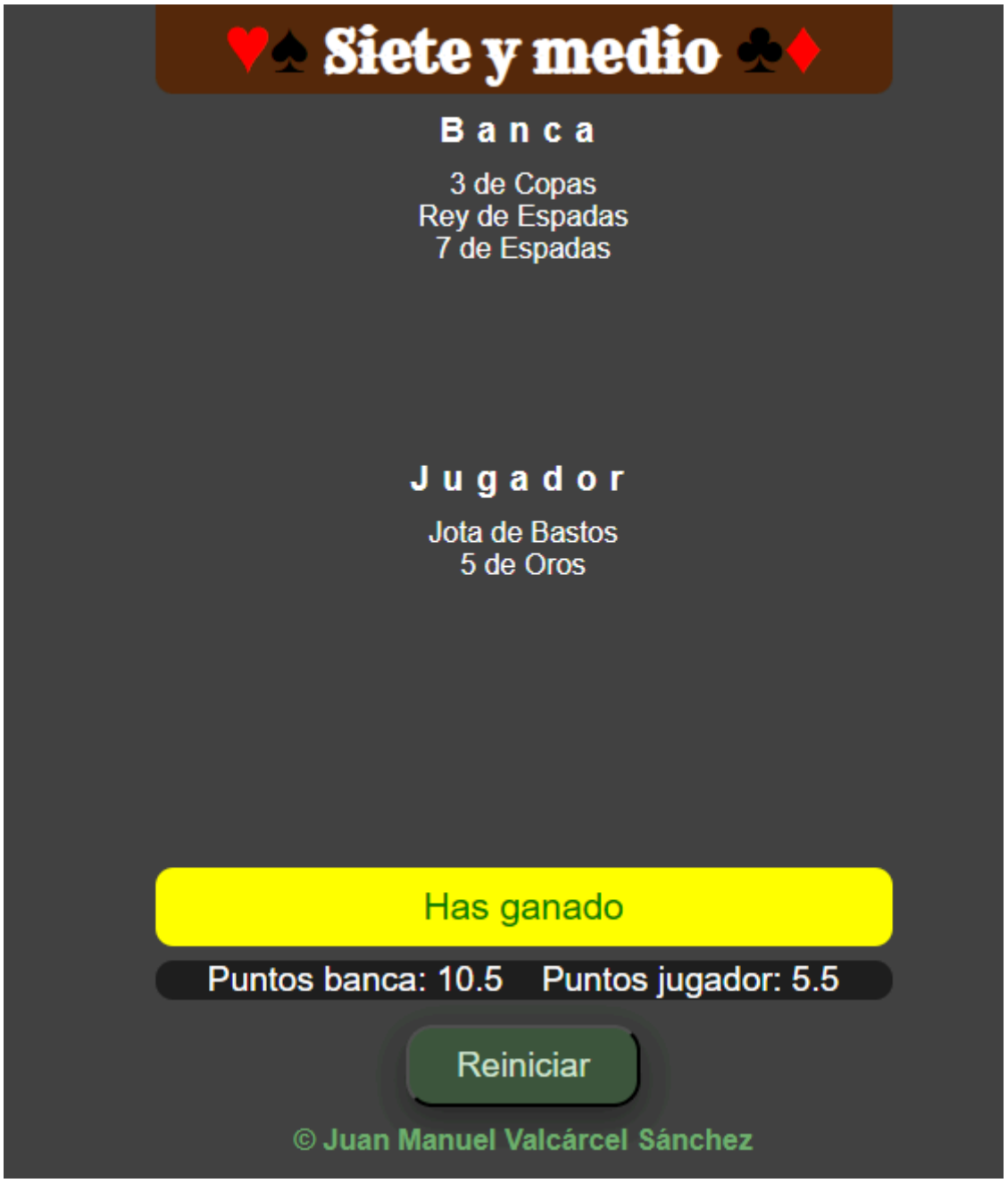


Imagen 32: Partida ganada por el jugador

En la imagen 33 se muestra cómo se empató una partida.

Imagen 33: Partida acabada en empate

Validación y feedback

Aunque esta es la iteración final de este proyecto por el límite de tiempo, se ha recogido aún así el feedback de los usuarios para que si en un futuro se sigue el desarrollo de la aplicación o alguien la utiliza cómo inspiración lo tenga en cuenta.

Para esta iteración los usuarios han tenido más tiempo y facilidades para probar el programa debido a la posibilidad de probarlo online, lo que ha permitido aumentar el feedback para esta iteración.

La mayoría de los comentarios han sido referidos a la UI; se reitera la necesidad de tener las imágenes de las cartas en vez de texto, además de la necesidad de declarar de manera más clara quién es el ganador de la partida, no olvidando la necesidad de un tutorial para los distintos juegos que incluye el programa.

Debido a que todos estos comentarios ya estaban en historias de usuario se ha procedido a aumentar su prioridad con respecto a las otras HU[5].

Con respecto a nuevos comentarios, se han recogido varios de una necesidad de un botón para volver a la anterior pantalla desde los juegos (botón para ir al selector de juegos).

Un pequeño porcentaje de los usuarios refiere la posibilidad de poder jugar sin necesidad de estar conectado a la red o si fuera posible como aplicación móvil o de escritorio además de la versión web.

Por último, determinan que gustaría tener juegos de modalidad multijugador siendo este online o en local, para poder tener un catálogo que no sea sólo de un jugador.

4 CONCLUSIONES Y TRABAJOS FUTUROS

Debido a la naturaleza ágil del proyecto y al limitado tiempo del TFG, se ha desarrollado una aplicación totalmente funcional en un entorno de desarrollo; si se quisiera poner en producción, se deberían alojar los distintos microservicios en servicios de hosting o crear servidores propios.

Con respecto al estudio del uso de arquitecturas de microservicios en proyectos pequeños se saca en claro la facilidad de generar nuevos microservicios a partir de los ya creados; puesto que la estructura ya está hecha, sólo se debe cambiar el contenido de ellos.

Aboga por la reutilización de los propios microservicios, debido a que si una función es usada por varios microservicios podemos crear un microservicio nuevo que provea de esta funcionalidad a los anteriores.

Esta misma ventaja la podemos ver en el desarrollo del proyecto debido a que en el diagrama de Gantt inicial estimamos un tiempo de setenta y cinco horas para realizar la segunda iteración, como se observa en la imagen 34.

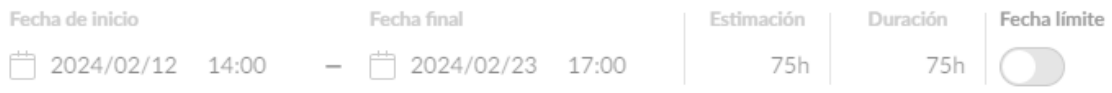


Imagen 34: Estimación inicial de tiempo de la segunda iteración

Pero debido a que ya estaba hecha la estructura del proyecto y gracias a los microservicios ya implementados, ha habido funcionalidades que ya estaban satisfechas gracias a estos microservicios. Lo que ha conducido a una implementación mucho más rápida, recortando el tiempo de implementación a cincuenta horas, en vez de las setenta y cinco horas estimadas.

Todo ello supone que, debido a la utilización de microservicios, se ha recortado un 30% del tiempo previsto para la iteración, adelantando el proyecto en general.

Y por último, simplifica la detección de errores ya que cada microservicio define sus entradas y salidas pudiendo aislar el error en un sólo microservicio o en la comunicación entre ellos, lo que delimita el ámbito de búsqueda.

Los principales inconvenientes han sido la complejidad de implementar la infraestructura al inicio de la creación de la aplicación, puesto que la comunicación de los microservicios y la generalización de su estructura consume un tiempo valioso al empezar el proyecto.

Al enfocarnos en proyectos pequeños, debemos indicar que la mayoría de ellos no aprovechan los principales beneficios de los microservicios, pues la escalabilidad se resalta con un gran número de microservicios al igual que la reutilización de estos, haciendo que los verdaderos beneficios se noten a mediano y largo plazo en el proyecto, tiempos en los que los proyectos pequeños tengan fuera de alcance o casi en su finalización.

El mayor problema de estas arquitecturas es el despliegue de los microservicios, pues la mayoría, sino todos ellos, para funcionar dependen de la comunicación entre ellos, y en una máquina es fácil su comunicación (sólo se necesita saber en qué puertos están desplegados los distintos microservicios), pero en producción se debe poder acceder a las distintas url que generan cada microservicio en sus respectivos hosting.

5 BIBLIOGRAFÍA

1. UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*"
   https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0039577

2. Martinekuan. (s. f.). *Estilo de arquitectura de microservicios - Azure Architecture Center*. Microsoft Learn.
   https://learn.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices

3. Atlassian. (s. f.). *Comparación entre la arquitectura monolítica y la arquitectura de microservicios*.
   https://www.atlassian.com/es/microservices/microservices-architecture/microservices-vs-monolith

4. Jamesmontemagno. (2022, 21, Septiembre). *The API gateway pattern versus the direct client-to-microservice communication - .NET*. Microsoft Learn.
   https://learn.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern

5. Rehkopf, D. M. (s. f.). *Historias de usuario | Ejemplos y plantilla | Atlassian*. Atlassian.
   https://www.atlassian.com/es/agile/project-management/user-stories

6. Pons, L. (2021, 18 junio). *¿Qué son los Endpoints y para qué sirven?* ICM.
   https://www.icm.es/2021/06/15/que-son-endpoints/

6 APÉNDICES

6.1 Instalación y configuración del sistema

Para que el programa funcione se deben poner en marcha todos los distintos microservicios y cambiar las url que están dentro de los microservicios para que concuerden con las url a través de las cuales se podrán acceder los microservicios en el nuevo entorno.

Pasos para poner en marcha el programa:

1. Descargar los microservicios desde github.
2. Cambiar las url de comunicación entre los distintos microservicios por las url que se utilizan en su contexto (si es para usar usar en la misma máquina se puede utilizar localhost).
3. Poner en marcha cada uno de los microservicios con ***node init***.

Se deberá de tener en cuenta que los distintos microservicios deben de tener acceso entre ellos, lo que determinará que, para funcionar, se recomiende al principio tenerlos todos en una misma máquina.

En el caso de los microservicios de lógica de negocio que se comunican con microservicios de UI, deberán estar disponibles para acceder de manera pública pues, si se accede a las UI desde otros dispositivos, estas interfaces deberán de poder acceder a ellos remotamente.

Los microservicios que no son necesarios exponer al exterior son los microservicios que no se comunican con microservicios de interfaces de usuario, ya que, si están en la misma máquina que los microservicios con los que se comunican, pueden tener acceso de manera local a ellos sin necesidad de exponerlos.

6.2 Manuales de usuario

Para utilizar el programa deberemos acceder a través del navegador por la url del selector de juegos que nos proporcionará la interfaz mostrada en la imagen 35:

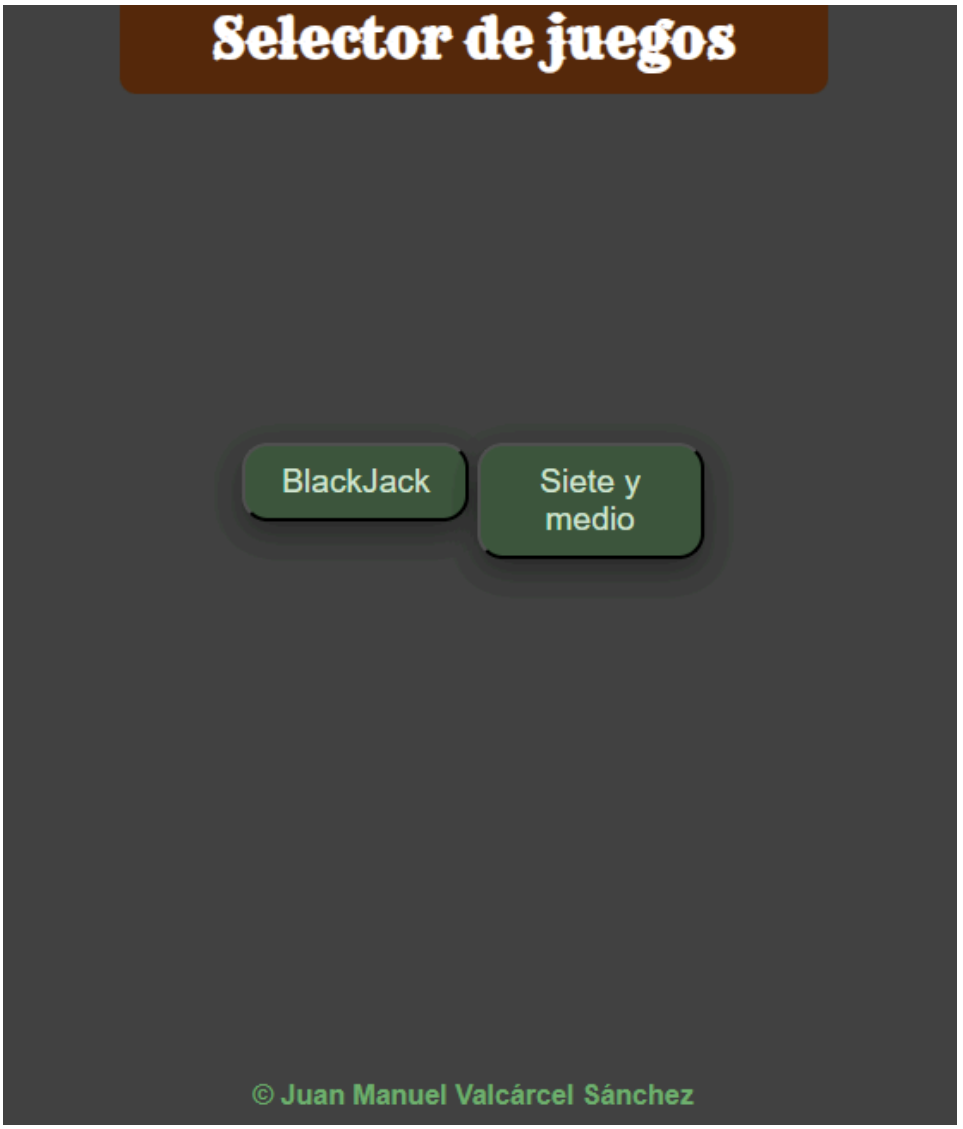


Imagen 35: UI del selector de juegos

En esta interfaz disponemos de diferentes botones que corresponden a cada uno de los juegos disponibles, los cuales, nos llevarán a cada uno de los juegos:

Imagen 36: Botón para jugar al BlackJack

Imagen 37: Botón para jugar al Siete y medio

Al presionar el botón resaltado en la imagen 36, se mostrará la interfaz que se muestra en la imagen 38 para jugar al BlackJack:

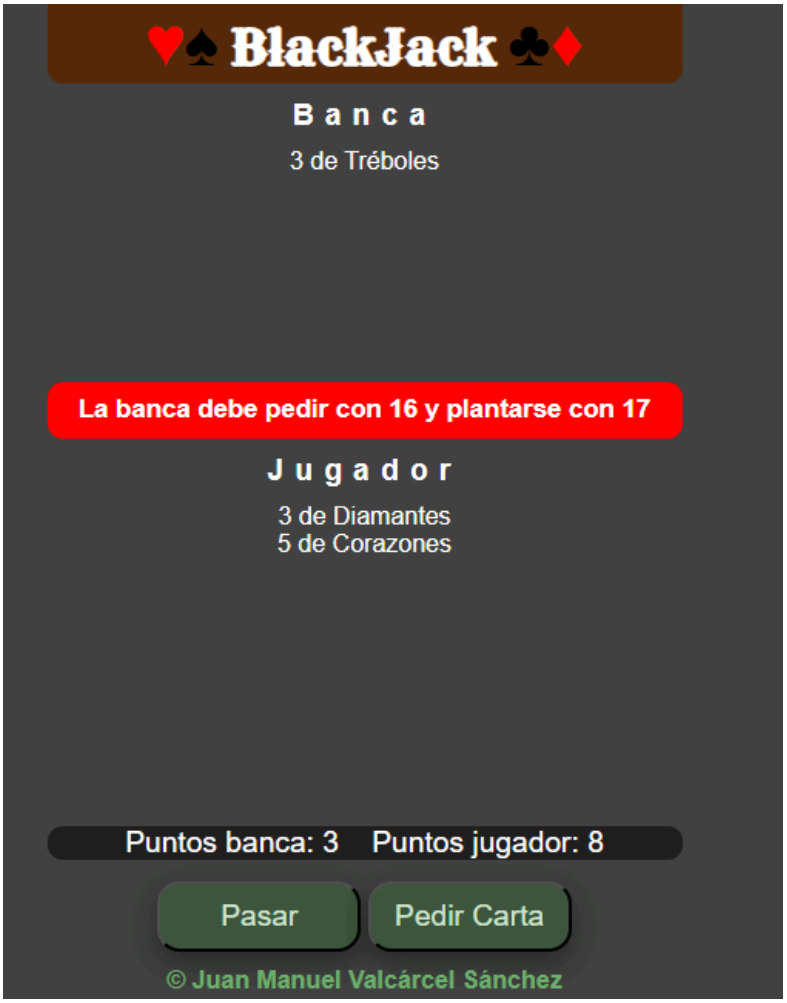


Imagen 38: Interfaz del juego BlackJack

En ella podemos observar en la parte superior de la imagen los datos de la partida, tales como las cartas de la banca y del jugador tanto como sus puntuaciones.

En la parte inferior nos encontramos con dos botones que corresponden a las acciones que podemos realizar en el juego.

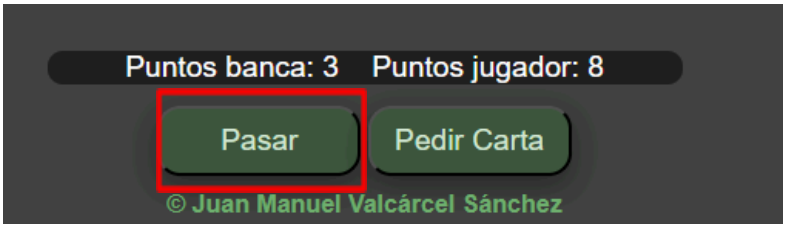


Imagen 39: Botón para pasar el turno

Si pulsamos el botón marcado en la imagen 39, pasaremos nuestro turno para robar, dando por finalizada la partida mostrándonos la interfaz que está en la imagen 40 pero con el resultado que hayamos obtenido en esa partida.

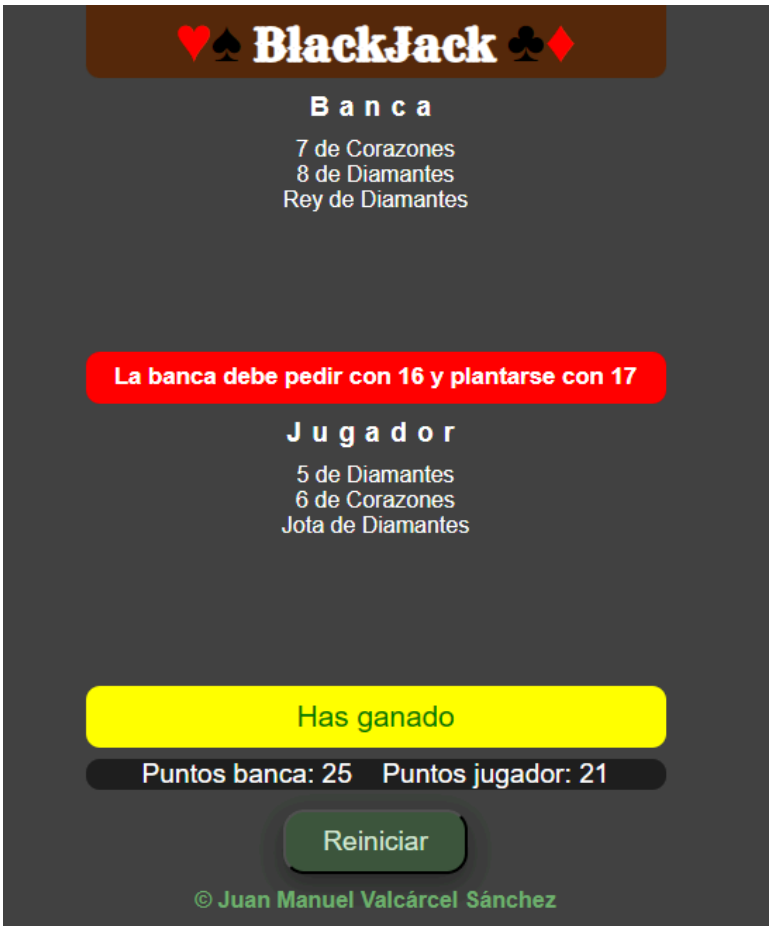


Imagen 40: Ejemplo de partida finalizada

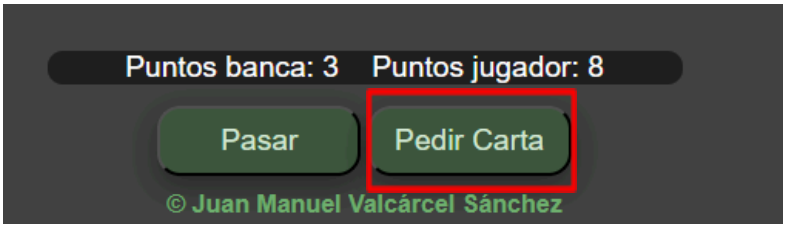


Imagen 41: Botón para pedir otra carta

Si pulsamos el botón de la imagen 41, robaremos otra carta del mazo volviendo a la interfaz de la imagen 42 o en su defecto, si nos hemos pasado de la puntuación máxima permitida, se nos mostraría la imagen 40 con una indicación de que hemos perdido la partida.

Imagen 42: Interfaz después de robar una carta

Si pulsamos el botón de la imagen 37 en la interfaz del selector de juegos, ahora obtendremos el juego siete y medio que mostramos en la imagen 43:

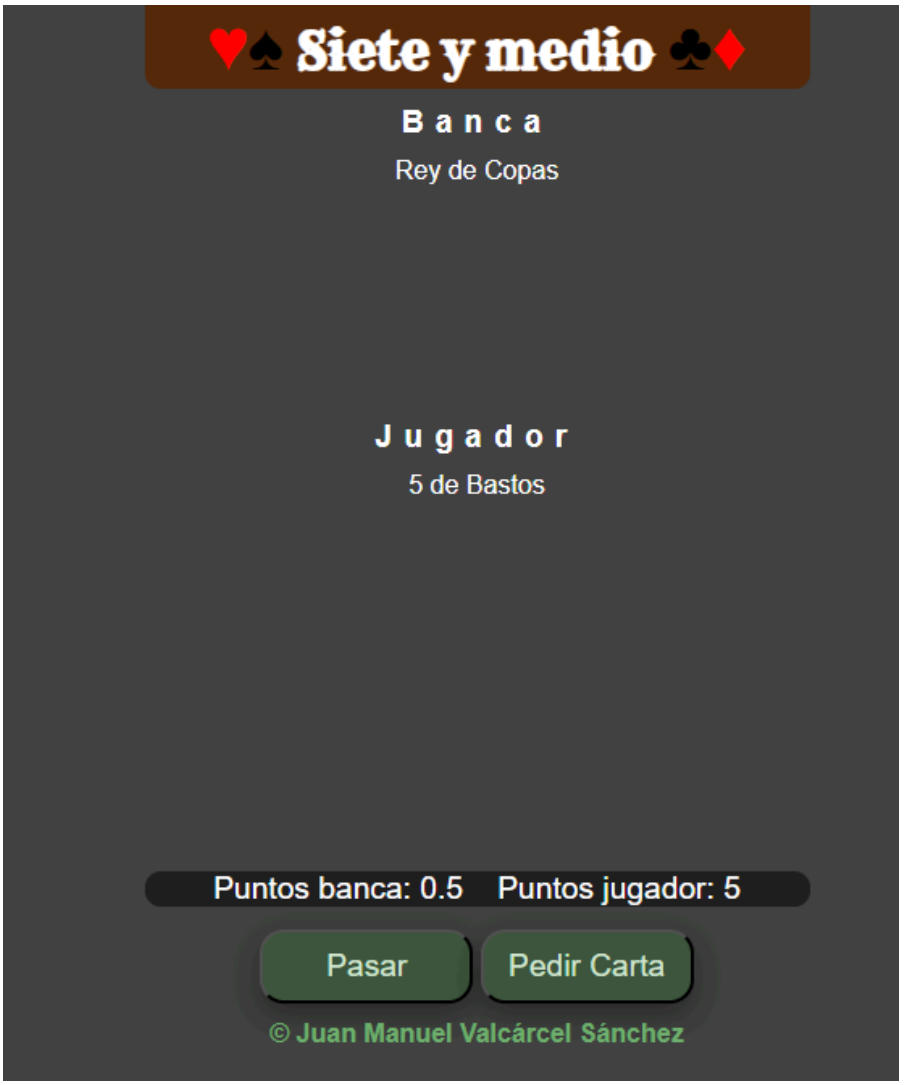


Imagen 43: Interfaz del juego siete y medio

En esta interfaz tenemos en la parte superior los datos de la partida que necesitamos para jugar, tales como las cartas de cada jugador, así como las puntuaciones de estas cartas.

En la parte inferior encontramos dos botones que representan las distintas acciones a realizar durante el juego.

Imagen 44: Botón para pasar el turno

El botón de la imagen 44 haría pasar el turno al jugador, terminando así la partida mostrando la imagen 45 con el resultado correspondiente a la finalización de la partida actual.

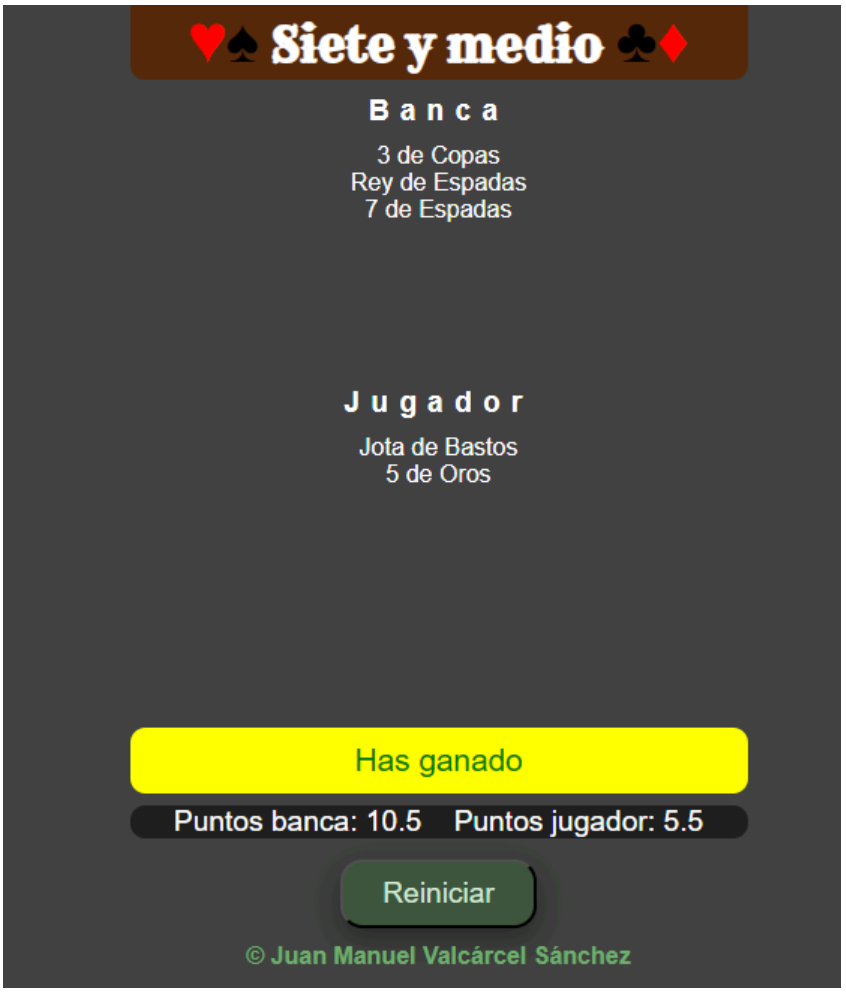


Imagen 45: Ejemplo de partida finalizada

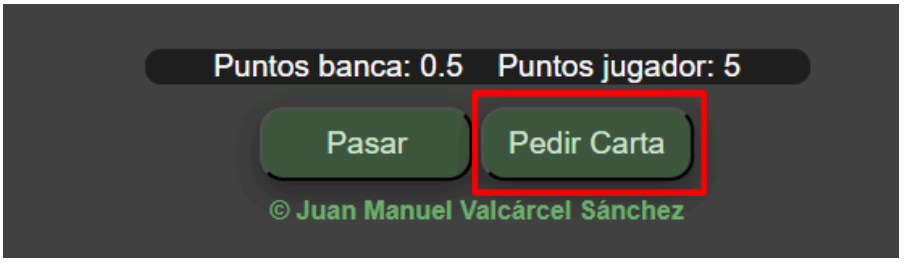


Imagen 46: Botón para pedir otra carta

Si pulsamos el botón de la imagen 46 estaríamos pidiendo robar otra carta del mazo pudiendo tener dos resultados: el primero sería que nos hubiéramos pasado de la puntuación máxima permitida, lo que nos traslada a la imagen 45 pero con la interfaz de perder la partida; el segundo resultado sería el de poder seguir jugando que nos mostraría la interfaz por defecto del juego con la carta robada ya visible en la mano del jugador como esta en la imagen 47.

Imagen 47: Ejemplo de robo de una carta

Por último, al finalizar la partida en ambos juegos aparecerá el botón de la imagen 48, lo que recargará de nuevo la página, dándonos la posibilidad jugar otra partida después de terminar la partida actual.

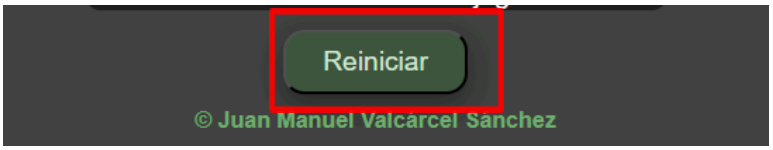


Imagen 48: Botón para reiniciar el juego