S.I.G.P.D.

Ingeniería de software

JurassiCode

Rol	Apellido	Nombre	C.I	Email
Coordinador	Fianza	Ignacio	5.690.153-1	businessignaciofianza@gmail.com
Sub-Coordinador	Benítez	Sebastián	5.652.044-4	sebastianbenitez2505@gmail.com
Integrante 1	Fleitas	Joaquín	5.570.982-3	joacolambru7@gmail.com
Integrante 2	Paz	Tomás	5.700.344-1	tomaslautaropaz@gmail.com

Docente: Cairús, Brandon

Fecha de culminación

15/9/2025

SEGUNDA ENTREGA

ÍNDICE

Ingeniería de software

REPOSITORIO DE GitHub:

https://github.com/JurassiCode/PROYECTO

Relevamiento aplicado al Juego Draftosaurus

1. Introducción

Durante una clase práctica, se utilizó el juego de mesa *Draftosaurus* como una dinámica lúdica con el fin de aplicar técnicas de relevamiento de requerimientos en un entorno simulado. A pesar de que el juego no constituye un sistema informático, su análisis permitió ejercitar habilidades esenciales para la comprensión, identificación y documentación de requerimientos funcionales y no funcionales, como se realiza en proyectos reales de Ingeniería de Software.

2. Resumen del Juego

Draftosaurus es un juego de mesa basado en la mecánica de selección por “draft”, en el cual cada jugador debe construir un parque jurásico personal seleccionando dinosaurios. Las decisiones están condicionadas por reglas específicas de colocación, y un dado introduce restricciones aleatorias en cada ronda. El objetivo del juego consiste en obtener la mayor cantidad de puntos al finalizar la partida, dependiendo de la distribución estratégica de los dinosaurios en las diferentes zonas del tablero individual.

3. Aplicación de Técnicas de Relevamiento

A continuación, se describen las técnicas de relevamiento aplicadas durante la experiencia y su correspondencia con prácticas reales en el desarrollo de software.

3.1 Cuestionarios (Implícitos)

Si bien no se utilizaron cuestionarios estructurados o escritos, se formularon preguntas concretas entre los jugadores antes y durante la partida. Estas consultas permitieron aclarar reglas y validar el entendimiento compartido del sistema.

Estas preguntas espontáneas ayudaron a identificar ambigüedades y divergencias en la interpretación de las reglas por parte de los usuarios.

Ejemplos:

- “¿Se puede repetir dinosaurio en esta zona?”
- “¿Cuándo se usa el dado?”

Este tipo de interacción simula el uso de cuestionarios abiertos y cerrados aplicados a usuarios reales para comprender funcionalidades, necesidades y expectativas en entornos de desarrollo.

3.2 Entrevistas (Informales)

Se llevaron a cabo conversaciones entre los participantes, especialmente con aquellos que ya conocían el juego. Estas entrevistas informales fueron fundamentales para comprender reglas complejas o mal interpretadas, así como para recoger experiencias previas.

Ejemplos:

- “¿Cómo jugaste vos la otra vez?”
- “¿Qué pasa si no me queda un lugar válido para poner el dinosaurio?”

Este tipo de diálogo reproduce las entrevistas realizadas en el contexto de proyectos informáticos, donde se consulta a usuarios o expertos en el dominio para clarificar requerimientos.

3.3 Etnografía (Observación Participativa)

Durante la partida, se observó el comportamiento de los jugadores, permitiendo identificar patrones, decisiones intuitivas y errores frecuentes. La observación participativa permitió comprender cómo interactúan los usuarios con un sistema, en este caso el juego, desde una perspectiva interna.

Ejemplos:

- Algunos jugadores evitaban consultar el manual y se guiaban por la experiencia de otros.
- Otros priorizaban ciertos tipos de dinosaurios de acuerdo con estrategias personales.

Esta técnica es equivalente a la etnografía en Ingeniería de Software, donde se observa el uso real de un sistema sin interferir en la actividad de los usuarios, a fin de comprender su comportamiento auténtico.

3.4 Análisis Bibliográfico (Manual del Juego)

Se consultó el manual oficial del juego para corroborar reglas y comportamientos observados durante la partida. El análisis documental es una técnica esencial para adquirir información precisa y estructurada sobre un sistema ya existente.

Ejemplos:

- Verificación de las condiciones para colocar dinosaurios en determinadas zonas.
- Confirmación del uso del dado y sus efectos en el desarrollo del juego.

Este análisis refleja la práctica común en el desarrollo de software de recurrir a documentación técnica o manuales de referencia para complementar y validar la información obtenida de otras fuentes.

4. Conclusión

La dinámica realizada permitió aplicar, en un entorno lúdico, diversas técnicas de relevamiento vistas en clase: cuestionarios, entrevistas, observación etnográfica y análisis bibliográfico. Aunque el sistema analizado no era informático, su estructura y reglas definidas facilitaron una simulación útil y representativa del trabajo de relevamiento en proyectos reales. Esta experiencia no solo contribuyó a comprender el juego, sino que también sirvió como ejercicio práctico para el desarrollo de competencias clave en Ingeniería de Software.

Entrevistas

Ingeniería de software

1. ¿Qué criterios debería usar el equipo para priorizar los requerimientos funcionales y no funcionales en base a la etapa de desarrollo actual?

2. ¿Cómo sugiere documentar correctamente las restricciones derivadas de las reglas del juego dentro del sistema, sin que afecte la flexibilidad del código?

3. Considerando que aún no se completó el modelo entidad-relación ni la normalización, ¿qué errores lógicos y de redundancia suelen aparecer en sistemas sin una base sólida?

4. ¿Qué técnicas de especificación considera más adecuadas para representar reglas lógicas complejas como la asignación de puntajes y validación de turnos?

5. ¿Cómo evaluar si la lógica implementada con estructuras condicionales cumple efectivamente con lo establecido en los árboles de decisión?

6. ¿Qué herramientas considera útiles para validar que se están cumpliendo correctamente los requerimientos no funcionales, como usabilidad, rendimiento o portabilidad?

7. ¿Qué estructura recomienda seguir para dejar documentado todo el diseño del sistema, desde su lógica interna hasta los criterios de instalación?

Tutoria de proyecto utu-lab

1. ¿Qué nivel de funcionalidad debería alcanzar un prototipo como JurassiDraft para considerarse válido desde el punto de vista institucional o pedagógico?

2. ¿Qué errores conceptuales suelen verse en proyectos escolares que simulan juegos, y cómo se pueden corregir o prevenir?

3. ¿Qué aspectos considera más importantes al evaluar la participación y el trabajo grupal en un proyecto interdisciplinario como este?

4. ¿Cómo se puede justificar ante un jurado o directivos que el sistema cumple una función educativa, más allá de lo lúdico?

5. ¿Qué documentación considera obligatoria entregar si este proyecto se quisiera presentar ante una institución educativa real?

6. ¿Cómo se puede asegurar la continuidad del proyecto si se busca implementarlo en otras generaciones o centros educativos?

Emprendedurismo y gestión

1. ¿Qué modelo de negocio considera más viable para un sistema como JurassiDraft que podría usarse tanto en instituciones como en actividades recreativas privadas?

2. ¿Qué herramientas recomienda para analizar si el mercado educativo estaría dispuesto a adoptar una plataforma de este tipo?

3. ¿Qué estrategias de validación del problema y de la solución podrían aplicar los estudiantes para confirmar que el producto tiene utilidad real?

4. ¿Qué tipo de pitch comercial recomendaría preparar si se quiere presentar JurassiDraft ante incubadoras de ideas o ferias estudiantiles?

5. ¿Qué análisis económico básico debe realizarse para saber si el proyecto es sustentable, incluso en su versión educativa gratuita?

6. ¿Qué aspectos legales deberían considerarse si el sistema comienza a recopilar datos personales de los usuarios en instituciones reales?

Programación full stack

1. Dado que JurassiDraft utiliza PHP como backend, ¿qué estructura de controladores y rutas considera adecuada para evitar duplicación de lógica o código espagueti?

2. ¿Cómo sugiere modularizar correctamente las funciones que validan reglas del juego, calculan puntuaciones y gestionan turnos?

3. ¿Qué técnicas de validación de entrada recomienda aplicar al manejar registros de usuarios y partidas, especialmente ante múltiples conexiones simultáneas?

4. ¿Qué patrón de arquitectura considera más mantenible para este tipo de proyectos educativos con múltiples vistas y funcionalidades CRUD?

5. ¿Qué herramientas de depuración o testing considera esenciales para validar el comportamiento del sistema antes de presentarlo?

6. ¿Qué nivel de documentación interna del código (comentarios, docblocks, estructuras) debería tener el backend para ser considerado “correctamente desarrollado”?

Inglés

1. ¿Qué vocabulario técnico considera imprescindible dominar si los estudiantes deben presentar el sistema en inglés ante docentes o jurados?

2. ¿Cómo debería organizarse un pitch oral en inglés para presentar JurassiDraft como un sistema educativo y digital?

3. ¿Qué errores comunes se deben evitar al traducir términos técnicos como "dashboard", "scoreboard", "game state", o "user login"?

4. ¿Qué recomendaciones da para generar una documentación en inglés clara, profesional y bien estructurada para uso académico?

5. ¿Qué ejercicios recomienda para mejorar la pronunciación y la seguridad al presentar un sistema como este frente a una audiencia internacional?

Sociología

1. ¿Qué impacto social puede tener un sistema como JurassiDraft si se implementa en centros educativos como herramienta de aprendizaje gamificado?

2. ¿Qué aspectos del uso de tecnología en el aula deberían evaluarse para saber si realmente mejora la participación y el trabajo en grupo?

3. ¿Qué indicadores sociológicos se pueden usar para medir el efecto de este tipo de sistemas en la inclusión educativa?

4. ¿Qué riesgos ve en la digitalización excesiva de experiencias lúdicas dentro del aula y cómo equilibrar lo virtual con lo presencial?

5. ¿Cómo puede justificarse que este proyecto promueve habilidades blandas como colaboración, respeto por las reglas y toma de decisiones?

Sistemas Operativos

1. ¿Qué ventajas tiene instalar el entorno LAMP en Fedora Server con DNF de forma manual frente al uso de soluciones como XAMPP, especialmente en entornos educativos?

2. ¿Qué configuraciones básicas deberían asegurarse en Apache y PHP para garantizar seguridad, rendimiento y compatibilidad?

3. ¿Cómo recomendaría estructurar y automatizar la verificación del estado de los servicios en un entorno de producción educativo?

4. ¿Qué consideraciones recomienda tener al configurar reglas de FirewallD para proteger el servidor sin impedir el acceso desde redes autorizadas?

5. ¿Qué errores comunes ve al instalar sistemas manualmente en Linux, y cómo evitarlos para que el entorno de JurassiDraft sea replicable sin fallos?

Las 4P

Las 4P del marketing (Producto, Precio, Plaza y Promoción) son una herramienta clásica para analizar y planificar la oferta de un producto o servicio. Aunque suelen utilizarse en contextos comerciales, también pueden adaptarse a proyectos académicos o tecnológicos para reflexionar sobre cómo se entrega valor, cómo se distribuye, a quién se dirige y cómo se comunica. En este caso, se aplican al sistema JurassiDraft, desarrollado como proyecto educativo.

Producto:

JurassiDraft es una plataforma web desarrollada con el objetivo de digitalizar y acompañar partidas del juego de mesa *Draftosaurus*. El sistema permite registrar las jugadas de cada participante, llevar control de puntajes en tiempo real y visualizar el desarrollo del tablero en formato digital, lo cual mejora la organización y seguimiento de las partidas tanto en espacios recreativos como educativos.

Está diseñado con una interfaz intuitiva, accesible desde cualquier dispositivo con navegador web, y adaptable a distintos perfiles de usuario (jugadores, administradores o docentes). Incluye funcionalidades específicas para la gestión de usuarios, visualización de estadísticas y generación de reportes. Su arquitectura modular permite futuras mejoras, adaptaciones o incluso su aplicación a otros juegos de mecánica similar.

Precio:

El sistema no tiene un costo económico, ya que fue desarrollado como parte de un proyecto académico. Sin embargo, su valor radica en su funcionalidad, utilidad educativa y potencial de reutilización. En un escenario futuro, podría ofrecerse bajo licencias libres, esquemas de implementación personalizada o incluso un modelo freemium con funciones ampliadas. La gratuidad en esta etapa refleja un compromiso con la innovación abierta y la accesibilidad en entornos educativos.

Plaza (distribución):

JurassiDraft puede instalarse en servidores locales de una institución educativa o ser alojado en un servidor web accesible desde cualquier navegador. Esto permite su uso tanto en aulas físicas como en entornos virtuales. Su distribución es digital, a través de plataformas como GitHub, copias locales o entornos de despliegue automatizado. Esto facilita su acceso por parte de docentes, estudiantes u otras instituciones interesadas en implementar herramientas lúdico-educativas.

Promoción:

La difusión del sistema se realiza principalmente dentro del entorno educativo.

Incluye presentaciones del equipo de desarrollo en clase, participación en ferias o muestras tecnológicas, y el uso por parte de docentes en actividades académicas. Además, el sistema puede promocionarse mediante portfolios personales, redes sociales o publicaciones académicas, mostrando tanto el resultado como el proceso de desarrollo. Estas vías permiten visibilizar el proyecto, demostrar habilidades del equipo y fomentar el interés en herramientas de gamificación educativa.

Interesados/StakeHolders

En el marco del desarrollo del sistema *JurassiDraft*, se identificaron diversos interesados o stakeholders. Son todas aquellas personas, grupos u organizaciones que afectan o se ven afectadas por el proyecto. Reconocer sus necesidades y gestionar su participación fue clave para garantizar un producto alineado a los objetivos académicos y funcionales del sistema.

Clasificación de stakeholders

Stakeholders primarios (directamente afectados):

- **Usuarios administradores:** personas encargadas de configurar el sistema, gestionar usuarios y acceder a reportes. Tienen interés alto y poder medio.
- **Jugadores (usuarios finales):** estudiantes o personas que interactúan directamente con el sistema durante las partidas. Interés alto, poder bajo.
- **Equipo de desarrollo:** integrantes del grupo encargado del diseño, implementación y pruebas del sistema. Interés y poder altos.
- **Docentes evaluadores:** profesores encargados de revisar los avances, orientar metodológicamente y calificar el proyecto. Poder alto, interés medio.

Stakeholders secundarios (afectados indirectamente):

- **Institución educativa (UTU):** puede beneficiarse de la reutilización o expansión del sistema en futuras generaciones.
- **Docentes que podrían usar el sistema** en actividades didácticas con gamificación.
- **Futuros desarrolladores:** estudiantes que podrían heredar o mejorar el código como parte de nuevos proyectos.

Stakeholders clave (con alto poder de decisión):

- **Coordinadores del curso / tribunal docente:** pueden influir directamente en la validación, cambios o aceptación final del proyecto.
- **Docentes técnicos de materias clave** (como Administración de S.O. e Ingeniería): pueden pedir correcciones o mejoras sustanciales.

Expectativas y necesidades comunes de los stakeholders

- Que el sistema funcione correctamente y sea intuitivo.
- Que las funcionalidades cubran las necesidades de uso reales en el aula o en partidas.
- Que el desarrollo se mantenga dentro de los plazos acordados y con buena documentación.
- Que se escuche y aplique el feedback recibido en las presentaciones.
- Que exista claridad sobre los roles, entregas y responsabilidades del equipo.

Técnicas aplicadas para su identificación y gestión

- Lluvia de ideas inicial entre los integrantes para detectar posibles actores involucrados.
- Observación y análisis del flujo del juego Draftosaurus para identificar tipos de usuarios.
- Validación constante con docentes para asegurar alineación con objetivos educativos.
- Uso de prototipos visuales y presentación de avances para recoger feedback temprano.
- Comunicación por WhatsApp y reuniones en clase para ajustar requerimientos y prioridades.

Constantine

Durante el desarrollo de JurassiDraft, el equipo adoptó un modelo de comunicación cercano al **paradigma abierto** definido por Constantine. Este enfoque permite que todos los miembros del equipo se comuniquen libremente entre sí, sin depender exclusivamente de un líder central para canalizar la información o tomar decisiones.

En este paradigma, las decisiones importantes se discuten en grupo, promoviendo una dinámica horizontal y participativa. A lo largo del proyecto, esto se manifestó mediante el uso constante de grupos de WhatsApp, reuniones presenciales en clase y la distribución flexible de tareas según la disponibilidad y capacidades de cada integrante.

Justificación de la elección:
El paradigma abierto fue el más adecuado debido a la naturaleza académica y colaborativa del proyecto. Al no haber una jerarquía formal impuesta, todos los integrantes (Sebastian, Tomas, Ignacion y Joaquin) pudieron aportar ideas, discutir decisiones técnicas y dividirse el trabajo de forma equitativa. Además, este modelo fomentó el compromiso de todos los miembros al sentirse partícipes reales del rumbo del proyecto.

Ventajas del paradigma abierto (según Constantine):

- Fomenta la participación activa y el compromiso del equipo
- Permite un equilibrio entre organización y libertad
- Favorece la creatividad, el consenso y la responsabilidad compartida
- Es adecuado para proyectos educativos donde el aprendizaje colectivo es un objetivo central

Desventajas a tener en cuenta:

- La toma de decisiones puede volverse lenta si no se llega a acuerdos fácilmente
- Puede haber conflictos o desorganización si no existe buena comunicación interna

- Requiere que todos los integrantes sean responsables y se mantengan activos

Estudio de Factibilidad

Proyecto: Software de Gestión de Partidas para Draftosaurus

1. Objetivo del Proyecto

El presente proyecto tiene como finalidad desarrollar una aplicación web que funcione como asistente digital para el juego de mesa *Draftosaurus*. Esta herramienta permitirá a los jugadores registrar sus jugadas, calcular los puntos de forma automática, llevar el control de las rondas, visualizar reglas específicas según la partida y almacenar el historial de sesiones jugadas. El sistema está diseñado para mejorar la experiencia de juego, minimizar errores manuales y ofrecer estadísticas que enriquezcan las partidas posteriores.

2. Tipos de Factibilidad

2.1 Factibilidad Técnica

Viabilidad: Alta

Desde el punto de vista técnico, el proyecto es completamente viable. Se utilizarán tecnologías accesibles y acordes a los conocimientos desarrollados en el curso:

- HTML y CSS: para la estructuración y estilización de la interfaz del sistema.
- Bootstrap: para asegurar un diseño responsivo, profesional y adaptable a diferentes dispositivos.
- PHP: para implementar la lógica del backend, incluyendo validaciones, reglas del juego y gestión de sesiones.
- MySQL: como sistema de almacenamiento de datos de jugadores, partidas y estadísticas.
- XAMPP: como entorno local de desarrollo que integra Apache y MySQL.

Ventajas Técnicas:

- Todas las tecnologías mencionadas son de código abierto y cuentan con amplia documentación.
- El equipo de trabajo ya ha utilizado o está familiarizándose con estas herramientas.
- Es posible construir los módulos esenciales (jugadores, partidas, puntuaciones) sin necesidad de tecnologías externas adicionales.

Desafíos Potenciales:

- Estructurar de forma eficiente la base de datos y la lógica del juego.
- Validar reglas dinámicamente según la condición del dado en cada ronda.
- Mantener la fluidez de la interfaz durante el desarrollo del juego.

Recomendaciones Técnicas:

- Dividir el desarrollo en módulos claramente definidos (jugadores, puntuación, reglas, historial).
- Utilizar Bootstrap para lograr una interfaz clara y simple.
- Realizar pruebas desde el comienzo en entorno local con XAMPP.

2.2 Factibilidad Económica

Viabilidad: Muy Alta

Este proyecto no implica costos económicos significativos para el equipo de desarrollo.

Costos Estimados:

Licencias: No se requiere inversión en licencias, dado que todas las tecnologías utilizadas son gratuitas.

Infraestructura: El desarrollo puede mantenerse en entorno local o utilizar alguna plataforma gratuita de alojamiento si se desea publicación.

Oportunidades Futuras:

- El sistema podría evolucionar hacia una herramienta en línea de acceso público.
- Existe potencial para ofrecer el software como recurso libre para la comunidad, siempre y cuando se respeten los aspectos legales correspondientes.

2.3 Factibilidad Operativa

Viabilidad: Alta

El sistema cumple una función clara y complementaria al juego físico. No intenta reemplazarlo, sino asistir a los jugadores en el desarrollo de la partida.

Beneficios Operativos:

- Registro automatizado de puntos y jugadas.
- Aplicación correcta y uniforme de las reglas del juego.
- Generación de estadísticas y/o rankings para jugadores frecuentes.

Riesgos Identificados:

- Algunos jugadores pueden resistirse al uso de tecnología durante un juego de mesa tradicional.
- Una interfaz mal diseñada o poco fluida puede entorpecer el ritmo del juego.

Soluciones Propuestas:

- Diseñar una interfaz simple, ligera y optimizada para su uso en dispositivos móviles.
- Desarrollar una versión mínima viable (MVP) enfocada en las funcionalidades esenciales: cálculo de puntos y control de rondas.

2.4 Factibilidad Legal y Ética

Viabilidad: Moderada

El juego Draftosaurus es una propiedad intelectual registrada, por lo que su uso implica ciertas restricciones legales.

Riesgos Legales:

- El uso del nombre del juego o material gráfico oficial puede infringir derechos de autor.
- No se debe presentar el software como herramienta oficial sin el correspondiente permiso.

Medidas Preventivas:

- Utilizar nombres alternativos como “Asistente de Partidas de Dinosaurios”.
- Evitar el uso de imágenes, íconos o contenido oficial del juego.
- Incluir un descargo de responsabilidad que indique el carácter no oficial, educativo y sin fines comerciales del proyecto.

2.5 Factibilidad de Tiempo

Viabilidad: Alta

El desarrollo del proyecto es factible dentro del plazo del semestre si se gestiona de forma organizada.

Plan Tentativo de Trabajo:

Etapa	*Duración Estimada*
Análisis y diseño inicial	***1 a 2 semanas***
Maquetado e interfaz (HTML/CSS/Bootstrap)	***2 a 3 semanas***
Programación de lógica con PHP	***2 a 3 semanas***
Integración con base de datos (MySQL)	***1 a 2 semanas***
Pruebas, ajustes y validaciones	***1 a 2 semanas***
Documentación y presentación final	***1 semana***

La planificación contempla márgenes de ajuste ante imprevistos, asegurando una entrega a tiempo con una versión funcional del sistema.

Actas de reunión:
https://drive.google.com/drive/folders/1I2-vKdz0Ktuk7k-5ynO1J2Hk5SaBG8dX?usp=sharing

Diagrama de gantt (descargar de drive e importar en https://www.onlinegantt.com/)
https://drive.google.com/drive/folders/1UJDQHNPj8VvL3SPgjLr2CrDLkEZjSdUw?usp=drive_link

Definición de Roles de Usuario, Permisos y Privilegios

Rol: Administrador

Descripción:

Usuario con acceso completo al sistema. Encargado de gestionar a otros usuarios, controlar las partidas y supervisar el funcionamiento general de la plataforma.

Permisos y privilegios:

- Crear, editar y eliminar usuarios (jugadores y otros administradores).
- Ver listado completo de usuarios registrados.
- Crear, editar y eliminar partidas.
- Asignar jugadores a partidas.
- Ver estadísticas de todas las partidas.
- Acceder al panel de administración completo.
- Ver y modificar configuraciones generales del sistema.
- No participa como jugador en una partida.

Rol: Jugador

Descripción:

Usuario final que participa en partidas del juego Draftosaurus. Tiene acceso limitado al sistema, centrado en su experiencia de juego.

Permisos y privilegios:

- Registrarse y autenticarse en el sistema.
- Ver su propio perfil y modificar datos personales (nombre, usuario, contraseña).
- Ingresar a partidas a las que fue asignado.
- Cargar sus movimientos o jugadas dentro de una partida.
- Ver resultados de partidas en las que participó.
- No puede crear, editar ni eliminar partidas.
- No puede ver ni administrar otros usuarios.

Especificación de Requerimientos (Funcionales, No Funcionales, Alcance y Limitaciones).

Alcance del sistema

El sistema a desarrollar es una aplicación web para la gestión de partidas del juego Draftosaurus, orientada al uso académico y recreativo. Permitirá a los usuarios registrarse como jugadores o administradores, gestionar partidas y registrar jugadas en cada turno. A su vez, los administradores podrán controlar tanto los usuarios como las partidas del sistema.

Requerimientos funcionales

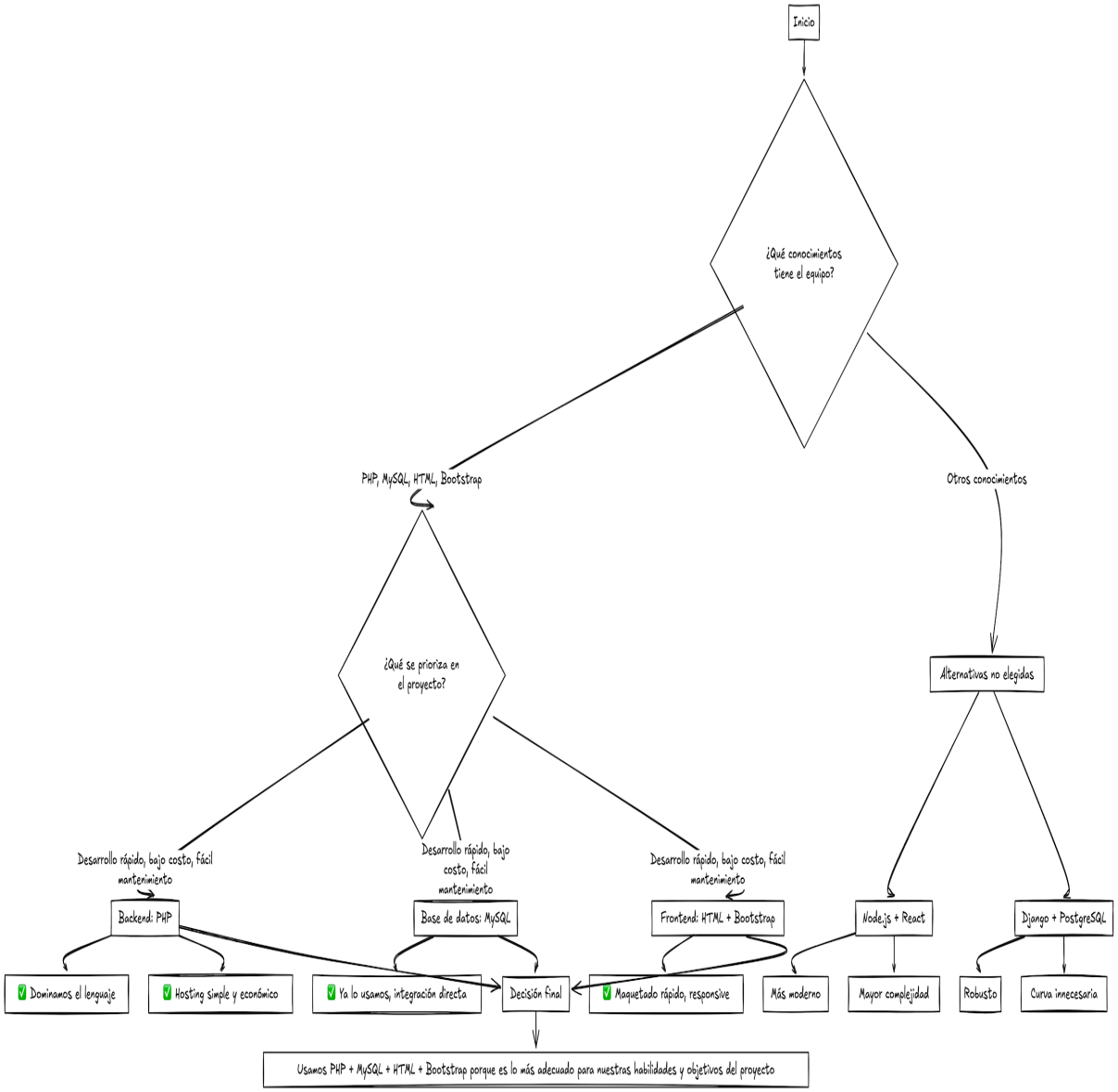
RF1: El sistema debe permitir el registro de nuevos jugadores.
RF2: El sistema debe permitir el inicio de sesión con validación de credenciales.
RF3: El sistema debe distinguir entre jugadores y administradores al iniciar sesión.
RF4: El administrador debe poder crear, editar y eliminar usuarios.
RF5: El administrador debe poder crear, editar y eliminar partidas.
RF6: El administrador debe poder asignar jugadores a partidas.
RF7: El administrador debe poder ver estadísticas generales del sistema.
RF8: El jugador debe poder ver y editar su perfil.
RF9: El jugador debe poder ver las partidas en las que participa.
RF10: El jugador debe poder ingresar a partidas activas y registrar sus jugadas.
RF11: El jugador debe poder ver los resultados de sus partidas.
RF12: El sistema debe registrar y guardar los datos de cada partida para su posterior consulta.
RF13: El sistema debe permitir al administrador acceder a reportes de uso, partidas jugadas y usuarios activos.
RF14: El sistema debe contar con un sistema de validación para evitar el acceso de usuarios no autorizados.
RF15: El sistema debe mostrar mensajes de error claros ante credenciales inválidas u operaciones no permitidas.

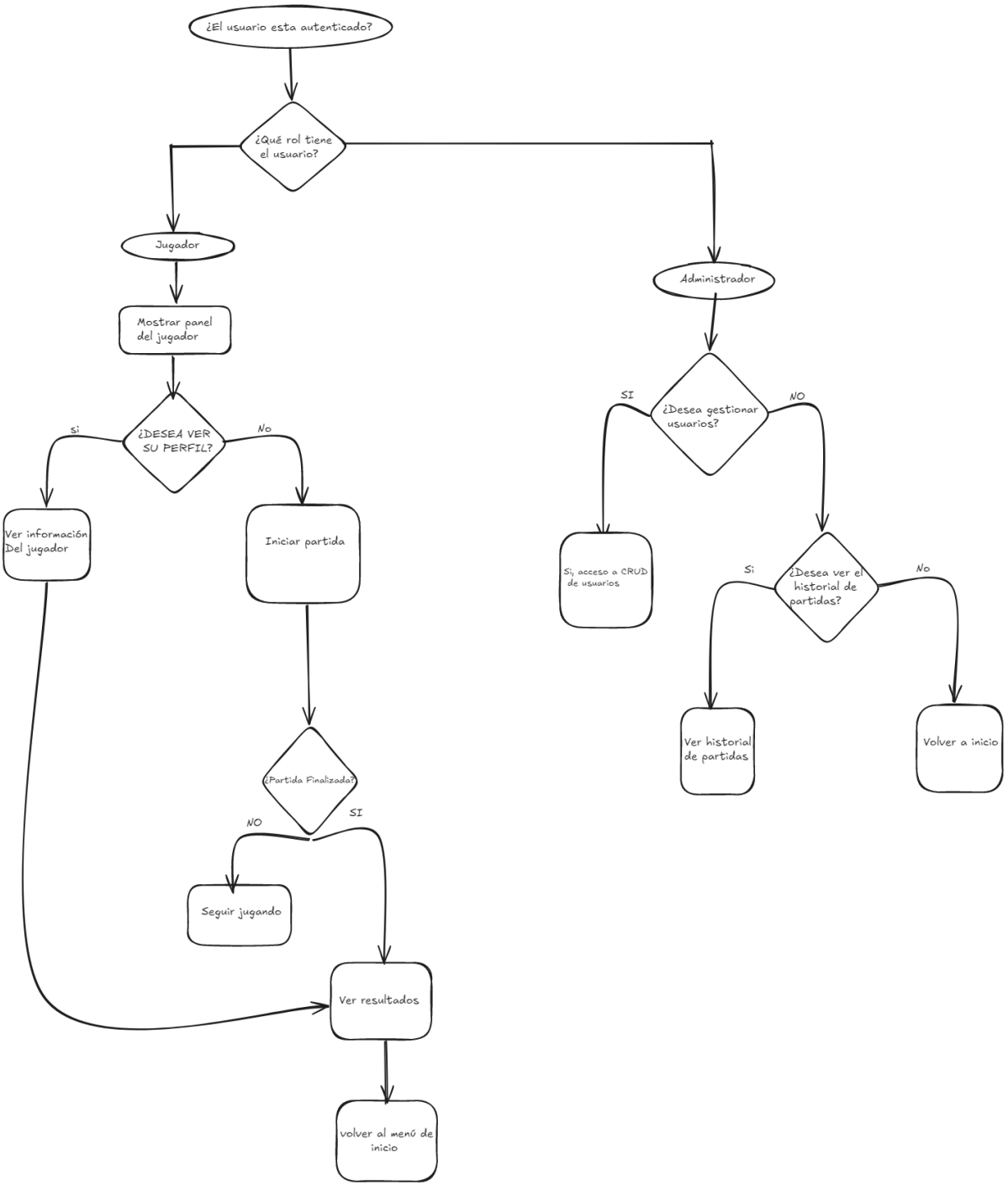
RF16: El sistema debe permitir cerrar sesión de forma segura desde cualquier perfil de usuario.

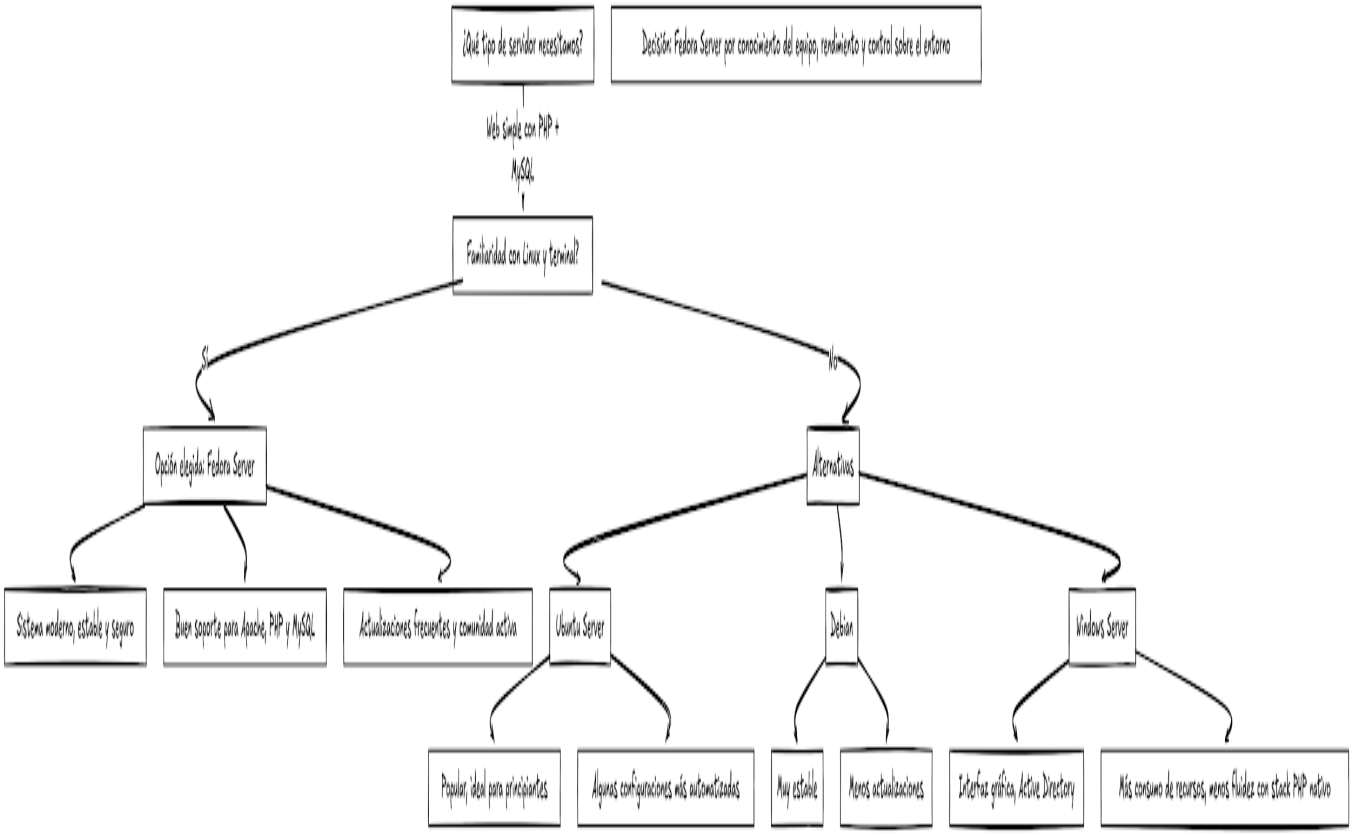
Requerimientos no funcionales

RNF1: El sistema debe cargar completamente en menos de 3 segundos.
RNF2: La interfaz debe ser responsive y adaptarse correctamente a dispositivos móviles y de escritorio.
RNF3: Las contraseñas deben almacenarse utilizando algoritmos de hashing seguro (como bcrypt).
RNF4: El acceso a funcionalidades debe estar restringido según el rol del usuario (jugador o administrador).
RNF5: Las sesiones deben expirar automáticamente tras un período de inactividad.
RNF6: El sistema debe desarrollarse de forma modular para facilitar el mantenimiento y futuras mejoras.
RNF7: La base de datos debe estar normalizada hasta la tercera forma normal (3FN).
RNF8: El sistema debe ser compatible con los principales navegadores modernos (Chrome, Firefox y Edge).
RNF9: El sistema debe ser accesible desde una red local o vía internet.
RNF10: Debe incluir documentación técnica y manual de usuario.
RNF11: La interfaz debe ser intuitiva y clara para el usuario final.
RNF12: Debe poder instalarse en un entorno LAMP (Linux, Apache, MySQL, PHP).
RNF13: Debe permitir realizar copias de seguridad manuales antes de cada entrega.
RNF14: La arquitectura debe permitir futuras expansiones sin grandes reestructuras.
RNF15: El sistema debe tener disponibilidad del 90% durante las pruebas.
RNF16: Las operaciones críticas deben incluir confirmaciones para evitar errores.
RNF17: No debe revelar información sensible en caso de fallos o errores.
RNF18: Debe cumplir principios básicos de accesibilidad y usabilidad.
RNF19: Primera entrega: 14/07/2025.
RNF20: Segunda entrega: 15/09/2025.
RNF21: Documentación final del proyecto: 10/11/2025.
RNF22: Instalación final del sistema: 12/11/2025.

Lógica del sistema y árboles de decisión




¿El usuario esta autenticado?
¿Qué rol tiene el usuario?
Jugador
Mostrar panel del jugador
¿DESEA VER SU PERFIL?
Si
No
Ver información Del jugador
Iniciar partida
¿Partida Finalizada?
NO
SI
Seguir jugando
Ver resultados
volver al menú de inicio
Administrador
¿Desea gestionar usuarios?
SI
NO
Si, acceso a CRUD de usuarios
¿Desea ver el historial de partidas?
Si
No
Ver historial de partidas
Volver a inicio



En caso de que las imágenes no se vean correctamente, dejamos acceso a la carpeta en drive de las mismas: https://drive.google.com/drive/folders/1qgAxLnHvzQJKm7OZR2NriBfwGfRdbp4S?usp=sharing

Ciclo de Vida Tradicional – Modelo en Cascada aplicado al proyecto

Idea central

El desarrollo del sistema se realiza paso a paso, sin retroceder. Cada etapa debe completarse totalmente antes de comenzar la siguiente.

1. Análisis

Objetivo:
Relevar y documentar todos los requisitos del sistema desde el inicio. En esta etapa se define qué debe hacer el sistema, sin pensar todavía en cómo lo va a hacer.

Aplicación al proyecto Draftosaurus:
En nuestro caso, lo primero fue entender el juego de mesa Draftosaurus para poder adaptarlo a una versión digital. Analizamos sus reglas, mecánicas y objetivos. A su vez, consultamos a docentes para conocer cómo pensaban usar el sistema con los estudiantes.

De esta forma, establecimos los **requisitos funcionales**, como:

- Registro y login de usuarios.
- Asignación de roles (jugador o administrador).
- Creación y administración de partidas.
- Registro de turnos y puntajes por jugador.

También definimos **requisitos no funcionales**, como:

- Buena velocidad de carga.
- Diseño intuitivo.
  Compatibilidad con dispositivos escolares.
- Seguridad básica de acceso.

Entregables de esta etapa:

- Documento con requisitos funcionales y no funcionales.
  Lista de funcionalidades que debe tener el sistema.
- Alcance y objetivos claros.

2. Diseño

Objetivo:
Planificar cómo se va a construir el sistema: tanto la parte visual como la estructura técnica. Se decide cómo se van a organizar las pantallas, formularios, funciones y base de datos.

Aplicación al proyecto:
En esta etapa pensamos cómo se iba a ver y comportar el sistema. Diseñamos los elementos principales de la interfaz (como formularios de registro, botones de acción y pantallas de juego), y los organizamos en bocetos simples o wireframes.

Además, planificamos cómo se iban a conectar las diferentes partes: la interfaz del usuario con la lógica del sistema y con la base de datos. También se definieron los permisos y funciones para cada tipo de usuario.

Entregables de esta etapa:

- Bocetos de cada pantalla del sistema.
- Plan general de organización del código.
- Definición de la estructura de base de datos.
- Documento técnico con decisiones de diseño.

3. Implementación

Objetivo:
Codificar todo el sistema siguiendo el diseño previamente planificado. Acá se transforma lo pensado en realidad.

Aplicación al proyecto:
Comenzamos escribiendo el código de las funciones básicas, como el registro, el inicio de sesión y el acceso a las distintas partes del sistema según el tipo de usuario. Se programó la lógica para administrar las partidas, registrar los turnos y almacenar los puntos de cada jugador.

Todo el desarrollo se hizo utilizando herramientas conocidas por el equipo, como PHP para la parte lógica, MySQL para la base de datos, y HTML/CSS/JS para la interfaz. El trabajo se organizó en módulos para mantener el código más claro y ordenado.

Entregables de esta etapa:

- Código completo del sistema.
- Sitio funcional en entorno local.
- Base de datos creada y conectada correctamente.

4. Testing

Objetivo:
Probar el sistema ya terminado para verificar que funcione como se espera. Se busca encontrar errores y asegurar que se cumplen los requisitos definidos en el análisis.

Aplicación al proyecto:
Realizamos pruebas con distintos tipos de usuarios, simulando situaciones reales de uso. Probamos todos los formularios, los botones, la lógica del turno por jugador y el sistema de puntajes. También hicimos pruebas en distintos dispositivos para asegurarnos de que la interfaz sea entendible y se vea bien.

Los errores encontrados fueron corregidos antes de dar por finalizado el sistema.

Entregables de esta etapa:

- Informe con resultados de las pruebas.
- Lista de errores corregidos.
- Sistema estable, listo para presentar.

Documentación de Inicio y Planificación del Proyecto

Nombre del Proyecto

JurassiDraft (Asistente Web para Partidas de Draftosaurus)

Integrantes del Grupo

- Ignacio Fianza (Coordinador)
- Sebastian Benitez (Sub Coordinador)
- Tomás Paz (Integrante 1)
- Joaquín Fleitas (Integrante 2)

Objetivo General

Desarrollar una aplicación web que sirva como asistente digital para el juego de mesa Draftosaurus, permitiendo registrar jugadores, jugadas, calcular puntos automáticamente y llevar el control de partidas, mejorando así la experiencia y la precisión durante el juego.

Alcance del Proyecto

- Registro de jugadores y creación de partidas.
- Carga de jugadas y rondas según las reglas del juego.
- Cálculo automático de puntajes.
- Visualización de resultados y estadísticas simples.
- Posibilidad de consultar el historial de partidas.
- Interfaz responsiva, usable desde celulares o computadoras.

Tecnologías Utilizadas

- Frontend: HTML, CSS, Bootstrap
- Backend: PHP
- Base de datos: MySQL
- Entorno de desarrollo: XAMPP (Apache + MySQL local)
- Organización del grupo: Hoja de cálculo compartida (Google Sheets)
- Comunicación: WhatsApp, Discord y reuniones presenciales en la UTU
- Control de Versiones: Git

Reglas del Grupo

1. Comunicación directa y continua
   Se mantiene contacto regular por WhatsApp o en persona durante horarios en la UTU. Cualquier duda o avance importante se comunica de inmediato.

2. Organización basada en tareas claras
   Las tareas están registradas en una hoja de cálculo compartida, con responsables asignados. Todos los integrantes pueden ver y actualizar el

estado de las tareas.

3. Compromiso con el trabajo
   Cada integrante debe cumplir con las tareas asignadas en los tiempos acordados. En caso de imprevistos, se debe avisar con antelación al grupo.

4. Participación equitativa
   Todos colaboran tanto en el desarrollo como en el análisis y documentación del proyecto. Las decisiones importantes se discuten en conjunto.

5. Respeto y trabajo en equipo
   Se fomenta un clima de respeto mutuo, diálogo abierto y apoyo entre compañeros. Las diferencias se resuelven conversando.

6. Revisión antes de integrar
   Cada componente debe ser probado en local (XAMPP) antes de integrarse al proyecto general. El código debe estar validado para asegurar compatibilidad.

7. Gestión de versiones básica
   Aunque no se utiliza un sistema formal como Git, se conservarán copias funcionales del proyecto en cada avance importante para evitar pérdida de información.

Plan de contingencias

1. Introducción

Este plan tiene como objetivo anticipar y gestionar posibles fallos o imprevistos que afecten el desarrollo, funcionamiento o seguridad del sistema JurassiDraft-Code. Al ser un sistema web pensado para digitalizar el juego Draftosaurus, que será usado en contextos educativos por docentes y estudiantes, se prioriza la disponibilidad, la integridad de los datos y la facilidad de recuperación ante incidentes.

2. Objetivos del plan

- Prever los principales riesgos técnicos, humanos y organizacionales.
- Minimizar los tiempos de inactividad del sistema.
- Establecer acciones claras y rápidas para responder ante imprevistos.
- Proteger la información crítica del sistema y mantener el servicio activo.
- Designar responsables para cada tipo de contingencia.

3. Identificación y clasificación de riesgos

R1 - Caída del servidor o de la base de datos. Tipo: técnico. Probabilidad: media. Impacto: alto.
R2 - Pérdida o corrupción de datos. Tipo: técnico. Probabilidad: baja. Impacto: alto.
R3 - Ataques de seguridad (como SQLi o XSS). Tipo: seguridad. Probabilidad: baja. Impacto: alto.
R4 - Errores de puntuación automática. Tipo: software. Probabilidad: medio. Impacto: medio.
R5 - Docentes o usuarios con poca capacitación. Tipo: humano. Probabilidad: media. Impacto: medio.
R6 - Caída o suspensión del hosting gratuito. Tipo: externo. Probabilidad: alta. Impacto: alto.
R7 - Saturación del sistema por exceso de conexiones. Tipo: técnico. Probabilidad: media. Impacto: medio.
R8 - Eliminación accidental de una partida. Tipo: humano. Probabilidad: media. Impacto: medio.
R9 - Fallo en la carga o descarga de partidas. Tipo: lógico. Probabilidad: media. Impacto: medio.
R10 - Incompatibilidad con dispositivos móviles. Tipo: software/UX. Probabilidad: baja. Impacto: medio.

4. Planes de acción por contingencia

R1 - Caída del servidor: se revisa el estado del servidor, se recupera backup y se documenta el incidente. Responsable: Sebastian,Ignacio. Tiempo estimado: 1 hora.
R2 - Pérdida de datos: se restaura la base de datos desde el último backup. Responsable: Sebastian,Ignacio. Tiempo estimado: 2 horas.
R3 - Ataque de seguridad: se suspende el servicio, se cambian credenciales, se analiza el ataque y se soluciona. Responsable: Ignacio,Tomas,Sebastian,Joaquin. Tiempo estimado: 6 horas.
R4 - Error en la puntuación: se pausa la puntuación automática y se corrige el código PHP. Responsable: Sebastian. Tiempo estimado: 3 horas.
R6 - Hosting caído: se traslada temporalmente el sistema a un servidor local o alternativo. Responsable: Ignacio,Tomas,Sebastian,Joaquin. Tiempo estimado: entre 12 y 24 horas.

5. Procedimiento general ante incidentes

Primero se detecta el incidente, ya sea por monitoreo técnico o por reporte de algún usuario o docente.

Luego se realiza un diagnóstico rápido para identificar la causa del problema y su impacto.

A continuación, se notifica al equipo y a los usuarios si corresponde.

Se ejecuta el plan de acción específico según el tipo de fallo, como se detalló anteriormente.

Una vez resuelto, se documenta el incidente, incluyendo fecha, duración, causa, solución aplicada y recomendaciones futuras.

Finalmente, se evalúa si es necesario ajustar alguna parte del sistema para evitar que el mismo problema vuelva a ocurrir.

6. Recursos y backups

El proyecto cuenta con respaldo periódico de todos los recursos críticos.

La base de datos MySQL se exporta manualmente cada semana como archivo .sql.

El código fuente del sistema está almacenado localmente y en una copia de seguridad externa (USB o nube).

Existe una versión portable del sistema adaptada para ejecutarse en entornos locales como WAMP o XAMPP, lista para ser usada en caso de emergencia.

También se utiliza Git como sistema de control de versiones para rastrear cambios y facilitar la recuperación ante errores de programación.

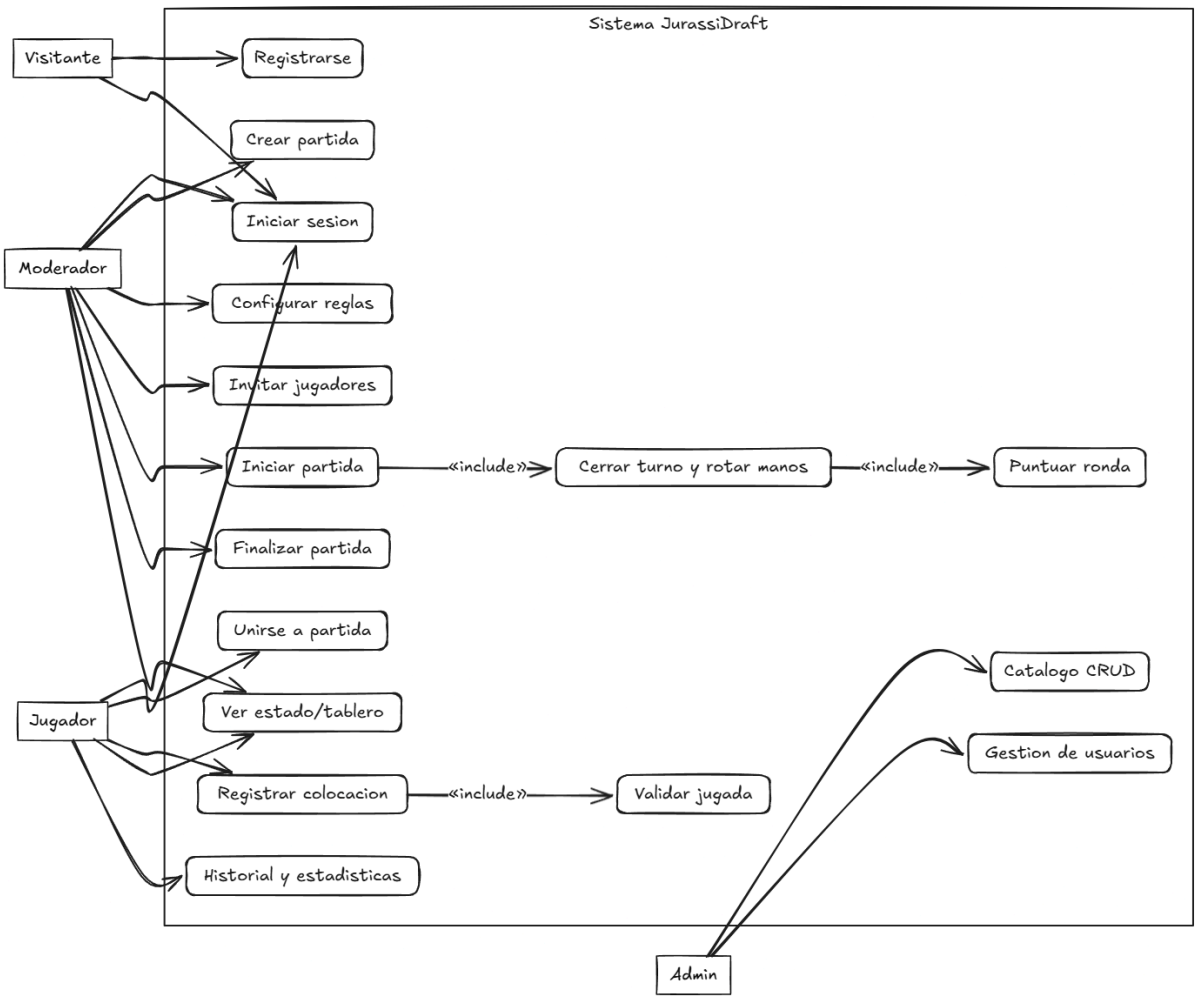
Diagrama UML (Casos de Uso) (Planilla y Diagramación)

ID	Caso de uso	Actor primario	Objetivo	Precondiciones	Flujo principal (resumen)	Postcondiciones	Excepciones
CU01	Registrarse	Visitante	Crear cuenta para jugar/gestionar	No estar autenticado	Completa formulario → valida email → alta	Usuario creado y logueado	Email inválido; usuario ya existe
CU02	Iniciar sesión	Jugador/Moderador/Admin	Acceder al sistema	Cuenta existente/verificada	Ingresa credenciales → autenticación	Sesión iniciada	Credenciales inválidas; cuenta bloqueada
CU04	Crear partida	Moderador	Abrir partida nueva	Sesión iniciada	Define nombre, rondas y modo → guardar	Partida en config	Datos incompletos
CU05	Configurar reglas	Moderador	Definir restricciones y tiempos	Partida en config	Establece restricciones (ej. dado) y tiempos → guardar	Reglas asociadas a la partida	Restricciones inválidas
CU06	Invitar/agregar jugadores	Moderador	Armar mesa	Partida en config	Busca usuarios o genera código → asigna orden	Jugadores vinculados	Usuario inexistente; cupo lleno

CU07	Unirse a partida	Jugador	Entrar a mesa existente	Cuenta activa; código válido	Ingresa código → validación → lobby	Jugador agregado a partida	Código inválido; partida llena
CU08	Iniciar partida	Moderador	Pasar a en_curso	Configuración y jugadores OK	Confirma inicio → reparte manos → tira dado	Partida en_curso, turno 1	Faltan jugadores/ configuración
CU09	Registrar colocación	Jugador	Jugar su turno	Turno activo	Selecciona dino de la mano y recinto → validar jugada → guardar	Colocación registrada	Jugada inválida (motivo)
CU10	Validar jugada	Sistema	Verificar reglas/restricciones	Propuesta de colocación	Chequea dado, recinto y reglas	Aprobada o rechazada	Datos inválidos
CU11	Cerrar turno y rotar manos	Sistema	Avanzar juego	Jugadas del turno completas	Cierra turno → rota manos → si fin de ronda puntúa	Turno/ronda actualizados	Faltan jugadas
CU12	Puntuar ronda	Sistema	Asignar puntos por ronda	Ronda cerrada	Calcula puntos según reglas → acumula	Puntos actualizados	Falta info de reglas
CU13	Finalizar partida	Moderador	Cerrar partida y ranking	Última ronda puntuada	Puntuar final; cambiar	Partida cerrada con ranking	Inconsistencias de datos

					estado a cerrada		
CU14	Ver estado/tablero	Jugador/Moderador	Visualizar progreso	Partida existente	Abre vista de tablero y puntos	Información mostrada	Sin permisos
CU15	Historial & estadísticas	Jugador/Moderador	Ver partidas previas	Sesión iniciada	Lista partidas → filtros	Historial visible	—
CU17	Gestión de catálogo	Admin	CRUD dinos/recintos	Sesión Admin	Crear/editar/eliminar ítems → validar	Catálogo actualizado	Impacto en partidas activas
CU18	Gestión de usuarios	Admin	Altas/bloqueos/roles	Sesión Admin	Buscar usuario → aplicar acción	Cambios aplicados	Restricciones FK; permisos

Diagrama de casos de uso



Desarrollo del diagrama del caso de uso

CU01 – Registrarse
Actor: Visitante
Propósito: crear cuenta.
Pre: no autenticado.
Flujo: abre formulario, completa datos, valida email, crea usuario.
Excepciones: email ya registrado.
Post: usuario creado y logueado.

CU02 – Iniciar sesión
Actor: Jugador/Moderador/Admin
Propósito: acceder al sistema.
Pre: cuenta existente/verificada.
Flujo: ingresar credenciales, validar, abrir sesión.
Excepciones: credenciales inválidas, cuenta bloqueada.
Post: sesión iniciada.

CU04 – Crear partida
Actor: Moderador
Propósito: abrir partida.
Pre: sesión iniciada.
Flujo: define nombre, rondas y modo, guarda.
Excepciones: datos incompletos.
Post: partida en estado config.

CU05 – Configurar reglas
Actor: Moderador
Propósito: definir restricciones y tiempos.
Pre: partida en config.
Flujo: establecer dado/tiempo, validar y guardar.
Excepciones: restricciones inválidas.
Post: reglas guardadas.

CU06 – Invitar/agregar jugadores
Actor: Moderador
Propósito: armar mesa.
Pre: partida en config.
Flujo: buscar usuarios o generar código, asignar orden.
Excepciones: usuario inexistente, cupo lleno.
Post: jugadores vinculados.

CU07 – Unirse a partida
Actor: Jugador
Propósito: unirse a mesa.

Pre: cuenta activa, código válido.
Flujo: ingresa código, sistema valida, entra al lobby.
Excepciones: código inválido, partida llena.
Post: jugador agregado.

CU08 – Iniciar partida
Actor: Moderador
Propósito: comenzar partida.
Pre: configuración y jugadores listos.
Flujo: confirmar inicio, repartir manos, tirar dado, setear turno/ronda.
Excepciones: faltan jugadores.
Post: partida en curso.

CU09 – Registrar colocación
Actor: Jugador
Propósito: jugar turno.
Pre: turno activo.
Flujo: elegir dino y recinto, validar jugada, guardar.
Excepciones: jugada inválida, tiempo agotado.
Post: colocación registrada o rechazada.

CU10 – Validar jugada
Actor: Sistema
Propósito: verificar reglas.
Pre: jugada propuesta.
Flujo: chequear dado, recinto, reglas; devolver válida o inválida.
Excepciones: datos inconsistentes.
Post: resultado validado.

CU11 – Cerrar turno y rotar manos
Actor: Sistema
Propósito: avanzar juego.
Pre: jugadas completas.
Flujo: cerrar turno, rotar manos, si fin de ronda puntuar.
Excepciones: faltan datos.
Post: turno/ronda avanzados.

CU12 – Puntuar ronda
Actor: Sistema
Propósito: asignar puntos.
Pre: ronda cerrada.
Flujo: calcular según reglas, acumular.
Excepciones: falta info.
Post: puntos actualizados.

CU13 – Finalizar partida
Actor: Moderador
Propósito: cerrar partida y ranking.
Pre: última ronda puntuada.
Flujo: confirmar final, puntuar final, estado cerrada.
Excepciones: inconsistencias.
Post: partida cerrada con ranking.

CU14 – Ver estado/tablero
Actor: Jugador/Moderador
Propósito: visualizar progreso.
Pre: partida existente.
Flujo: abrir vista, mostrar tablero y puntos.
Excepciones: sin permisos.
Post: información visible.

CU15 – Historial & estadísticas
Actor: Jugador/Moderador
Propósito: consultar resultados previos.
Pre: sesión iniciada.
Flujo: listar partidas, filtrar.
Excepciones: —
Post: historial mostrado.

CU17 – Gestión de catálogo
Actor: Admin
Propósito: administrar dinos/recintos.
Pre: sesión admin.
Flujo: crear, editar o eliminar; validar.
Excepciones: ítem en uso.
Post: catálogo actualizado.

CU18 – Gestión de usuarios
Actor: Admin
Propósito: administrar usuarios.
Pre: sesión admin.
Flujo: buscar usuario, cambiar rol o estado.
Excepciones: no degradar último admin.
Post: cambios aplicados.

Análisis costo-beneficio

El análisis costo–beneficio es una herramienta de evaluación utilizada en la gestión de proyectos para determinar la viabilidad económica y estratégica de una iniciativa. Su objetivo es comparar los recursos invertidos (costos) con los resultados positivos que se esperan obtener (beneficios), tanto tangibles como intangibles.

Aplicar este análisis en el proyecto **JurassiCode / JurassiDraft** permite fundamentar la conveniencia de su desarrollo, ya que se trata de una plataforma académica y recreativa orientada a la gestión digital de partidas del juego Draftosaurus, con potencial de aplicación en entornos educativos y en experiencias lúdicas innovadoras.

Costos

Los costos representan el dinero que hay que invertir para que el sistema funcione. Se pueden dividir en:

- **Costos iniciales:** lo que se paga al comienzo, como dominio y hosting.
- **Costos de operación:** gastos que podrían surgir con el uso, por ejemplo algún software adicional.

En nuestro proyecto se estiman:

- Dominio y hosting: 300 USD/año
- Software adicional eventual: 200 USD/año

Total costos: 500 USD/año

Beneficios

Los beneficios son las ventajas que se obtienen al usar el sistema. Se clasifican en:

- **Beneficios tangibles:** se pueden medir en dinero, como el ahorro de tiempo o la reducción de errores.
- **Beneficios intangibles:** no siempre se traducen en dinero, pero generan valor, como la satisfacción de los usuarios o la innovación educativa.

En este caso:

- Reducción de errores en cálculos y validaciones: 250 USD/año
- Ahorro de tiempo y mayor productividad: 350 USD/año

Total beneficios: 600 USD/año

ROI (Retorno sobre la inversión)
El ROI indica cuánto se gana por cada unidad de dinero invertido.
Fórmula: (Beneficios – Costos) / Costos × 100

Aplicado a este proyecto:
(600 – 500) / 500 × 100 = **20%**

Esto significa que por cada dólar invertido, se recupera el dólar original y se gana un 20% extra. Es una rentabilidad moderada y razonable.

Periodo de recuperación
Mide el tiempo que tarda en recuperarse la inversión.
Fórmula: Costos / Beneficios

500 / 600 = 0,83 años ≈ **10 meses**

En menos de un año se recupera lo invertido.

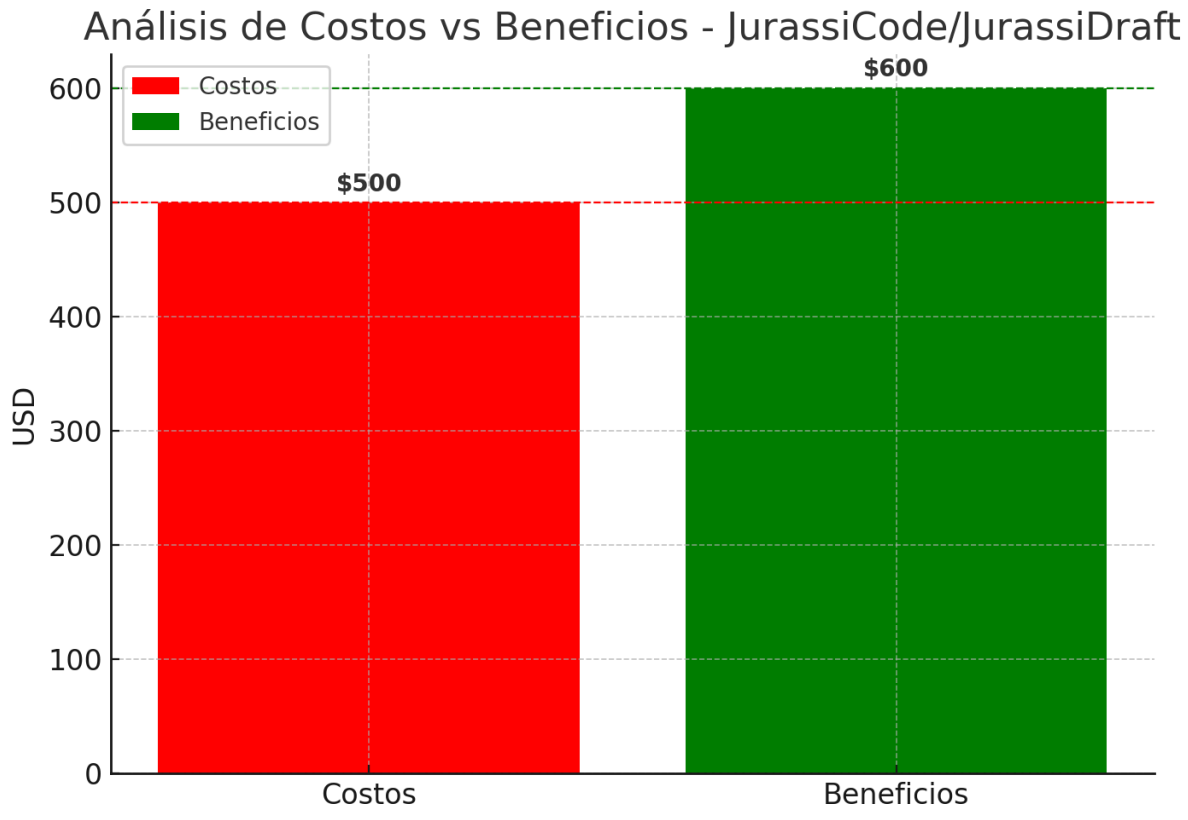


Diagrama de navegabilidad

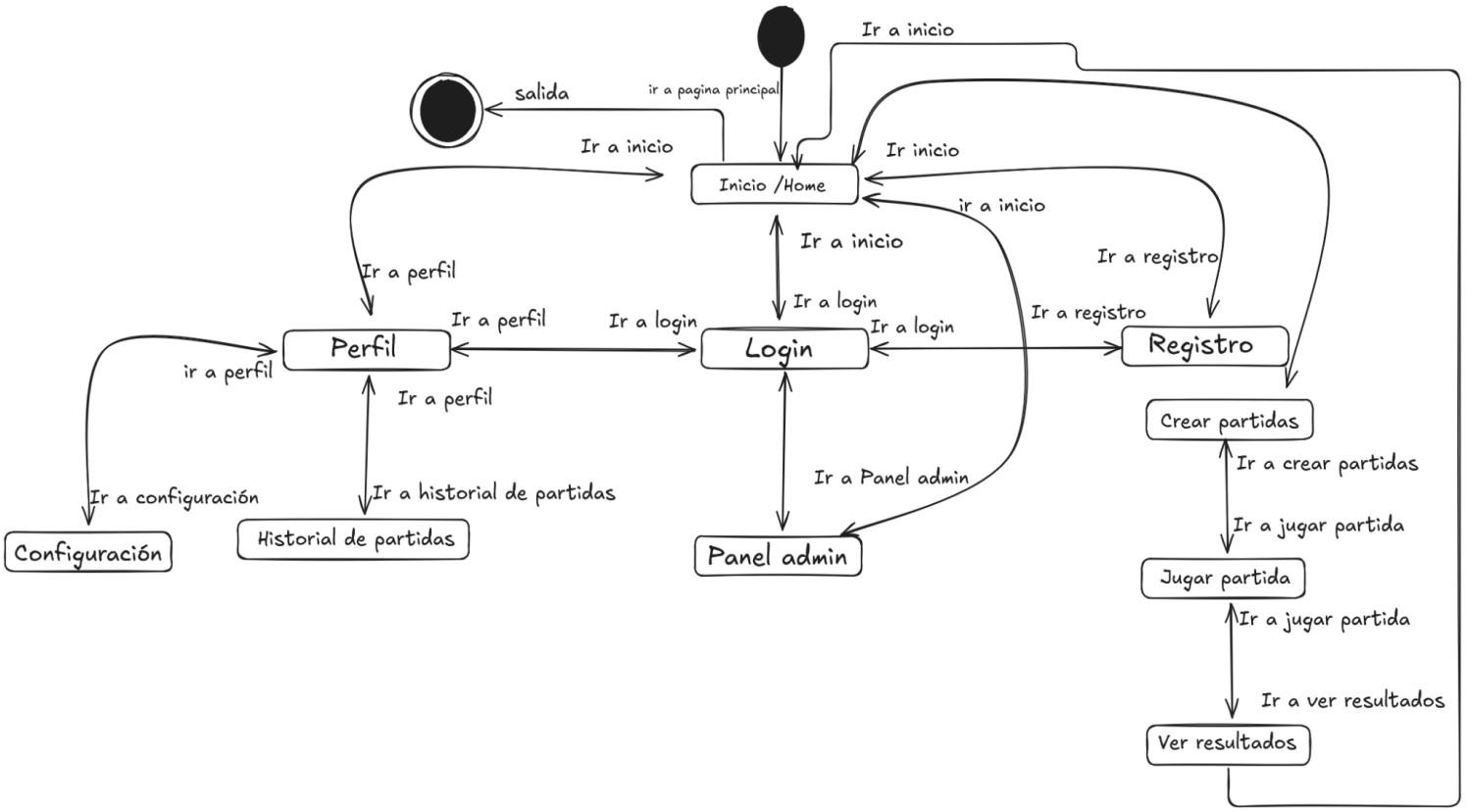
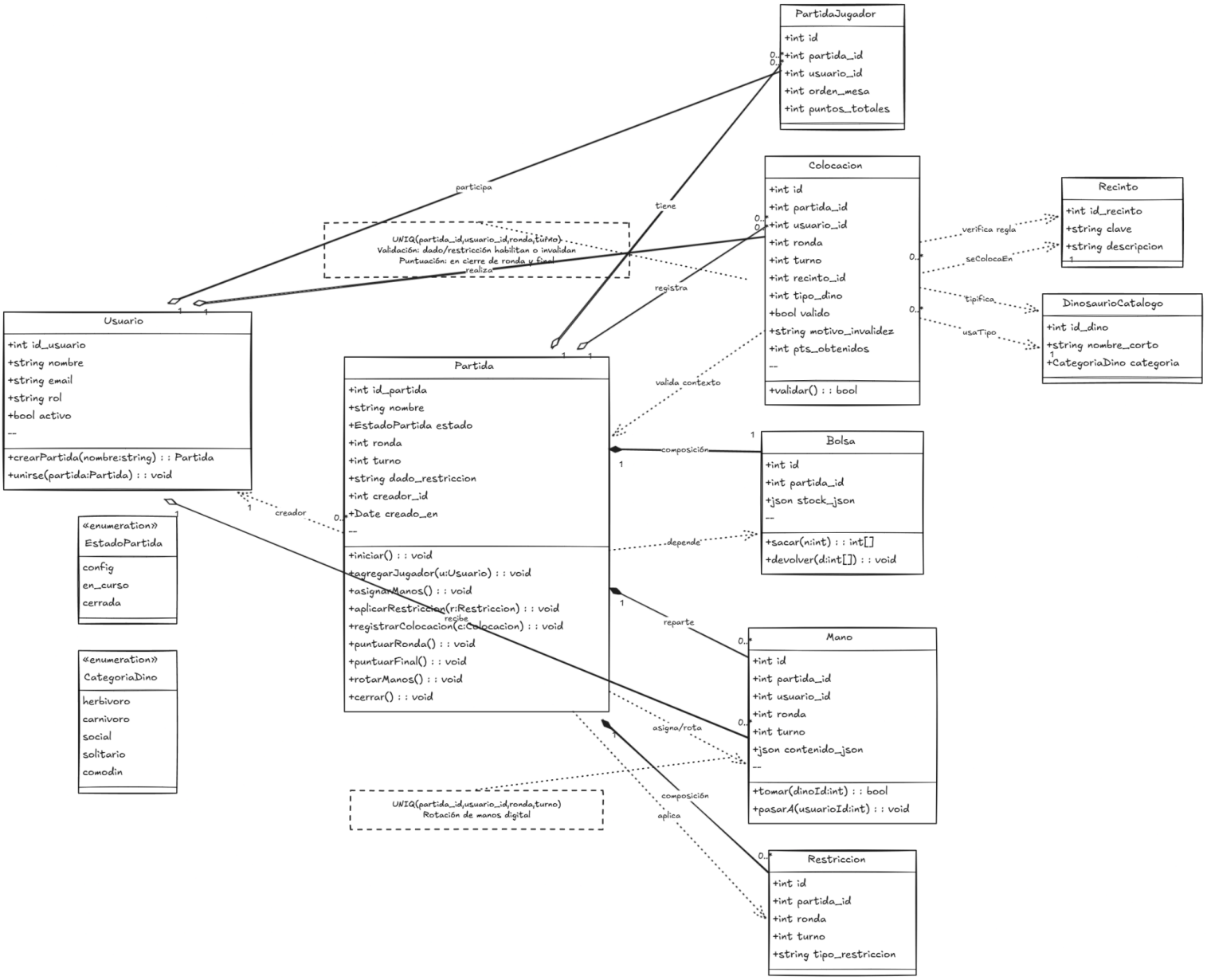


Diagrama de Clases



Cálculo de métricas del proyecto

1) Alcance y funcionalidades incluidas

Funcionalidades implementadas/ a implementar o definidas en el modelo de datos:

- **Usuarios**: registro, login y gestión de sesión.
- **Partidas**: crear, unirse, estado (config | en_curso | cerrada), ronda, turno, dado de restricción.
- **Jugadores en partida**: relación usuario–partida, orden de mesa, puntos.
- **Bolsas y manos**: asignación/rotación de manos; stock de dinosaurios por partida.
- **Colocaciones**: registrar colocación de dinosaurios en recintos, validaciones y puntos por jugada.
- **Recintos y catálogo de dinosaurios**: catálogos base.
- **Restricciones**: reglas activas por ronda/turno (dado).
- **Puntuación**: por ronda y final (tabla/visuales en interfaz).

2) Base de cálculo: categorías y pesos

- EI (Entradas externas): Baja 3 | Media 4 | Alta 6
- EO (Salidas externas): Baja 4 | Media 5 | Alta 7
- EQ (Consultas externas): Baja 3 | Media 4 | Alta 6
- ILF (Archivos lógicos internos): Baja 7 | Media 10 | Alta 15
- EIF (Archivos externos): Baja 5 | Media 7 | Alta 10

Criterios de complejidad: baja cuando intervienen pocos datos y lógica simple, media con más atributos/relaciones y validaciones, alta cuando hay múltiples ILF referenciados, reglas de negocio y/o algoritmos (p. ej., rotación de manos, validaciones de recintos, scoring).

3) Conteo detallado de Puntos de Función (PFSA)

3.1 ILF — Archivos lógicos internos (total 104)

ILF	Complejidad	Puntos	Justificación breve
usuarios	Media	10	Alta utilización (auth), varios atributos y validaciones.
partidas	Alta	15	Múltiples estados, ronda/turno, dado, relaciones (creador, jugadores, manos, etc.).
partida_jugadores	Media	10	Relación usuario–partida, orden de mesa, puntos totales.
manos	Alta	15	Estado por ronda/turno, JSON de contenido, reglas de rotación.
bolsas	Alta	15	Stock dinámico por partida y uso por turnos.
colocaciones	Alta	15	Registra jugadas con validación y puntaje por acción.
recintos	Baja	7	Catálogo estático base.
dinosaurios_catalogo	Baja	7	Catálogo estático base.
restricciones	Media	10	Reglas activas por ronda/turno (tipo de restricción).

Subtotal ILF = 104

3.2 EI — Entradas externas (total 34)

EI	Comp.	Puntos	Justificación breve
Registro de usuario	Baja	3	Formulario estándar con validación básica.
Login de usuario	Baja	3	Credenciales y sesión.
Crear partida	Media	4	Metadatos de partida + validaciones.
Unirse a partida	Media	4	Chequeos de estado/aforo.
Colocar dinosaurio en recinto	Alta	6	Toca manos, recintos, restricciones, colocaciones y reglas de negocio.
Cargar bolsa/mano inicial	Alta	6	Inicialización de estado con coherencia de stock.
Rotar manos	Media	4	Reglas de rotación por ronda/turno.
Registrar restricción de dado	Media	4	Alta/actualización de la restricción vigente por turno.

Subtotal EI = 34

3.3 EO — Salidas externas (total 24)

EO	**Comp.**	**Puntos**	**Justificación breve**
Tabla de posiciones	Alta	7	Cálculo de puntos, GF/GC, desempates, agregaciones.
Vista de mano actual del jugador	Media	5	Lectura y formateo del estado actual.
Vista de colocaciones por jugador/partida	Media	5	Agregación y ordenamientos por filtros.
Vista de recintos y puntajes parciales	Alta	7	Cálculos parciales por reglas de recinto.

Subtotal EO = 24

3.4 EQ — Consultas externas (total 11)

EQ	Comp.	Puntos	Justificación breve
Listado de partidas abiertas	Media	4	Filtros por estado, orden, paginado.
Historial de rondas/turnos	Media	4	Filtros por ronda/turno con ordenamientos.
Búsqueda de jugadores registrados	Baja	3	Búsqueda simple por nombre/email.

3.5 EIF — Archivos externos (total 0)

No se consideran integraciones externas en esta versión (sin import/export de terceros).

4) Factor de Ajuste (FA)

Evaluando las 14 características, JurassiDraft obtuvo 38 puntos.

Fórmula: PFA = PFSA × (0,65 + 0,01 × FA)

Cálculo: PFA = 173 × (0,65 + 0,38) = 178 PF

Justificación: el sistema tiene procesos internos complejos (rotación de manos, validación de recintos, cálculo de puntajes), mucha entrada de datos y seguridad básica, pero casi nada de integración externa.

5) Estimación de esfuerzo

Fórmula: Esfuerzo = (PF / 150) × (PF^0,4)

Cálculo: Esfuerzo = (178 / 150) × (178^0,4) ≈ 9,4 meses-persona

En horas: ≈ 1322 horas-persona

Justificación: el tamaño funcional del sistema es alto; por eso el esfuerzo calculado es cercano a 9,5 meses si lo hiciera una sola persona.

6) Duración del proyecto (según equipo)

Equipo de 4: ~9 semanas.

Equipo de 3: ~12 semanas.

Equipo de 2: ~19 semanas.

Justificación: a mayor cantidad de integrantes, se reduce la duración porque se reparten las horas de trabajo.

7) Conclusión

PFSA = 173 → PFA = 178

Esfuerzo = 1322 horas-persona (~9,4 meses-persona)

Con el equipo actual, el desarrollo es viable en unas 9–12 semanas.Diagrama de secuencia

Diagrama de secuencia

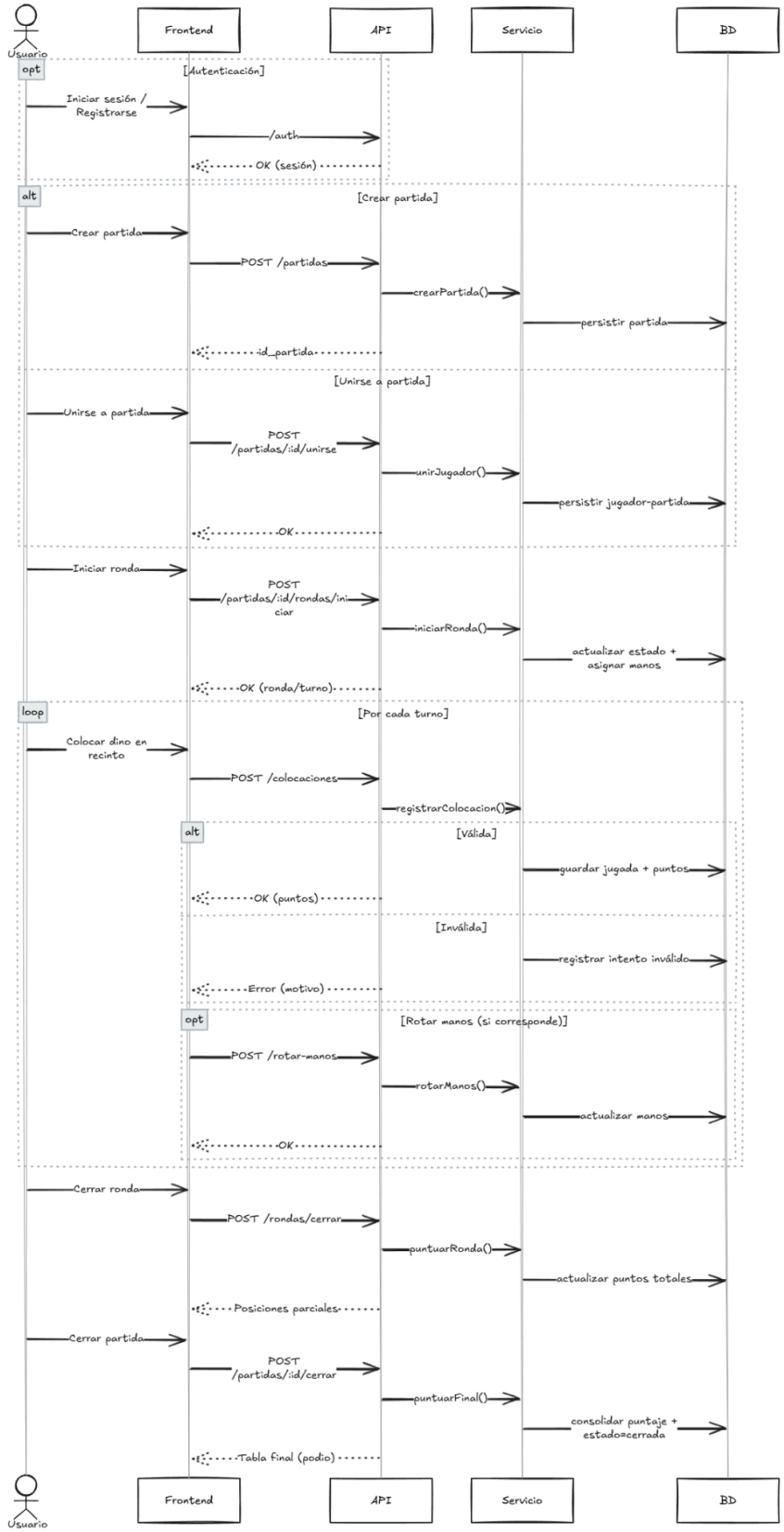


Diagrama de actividad

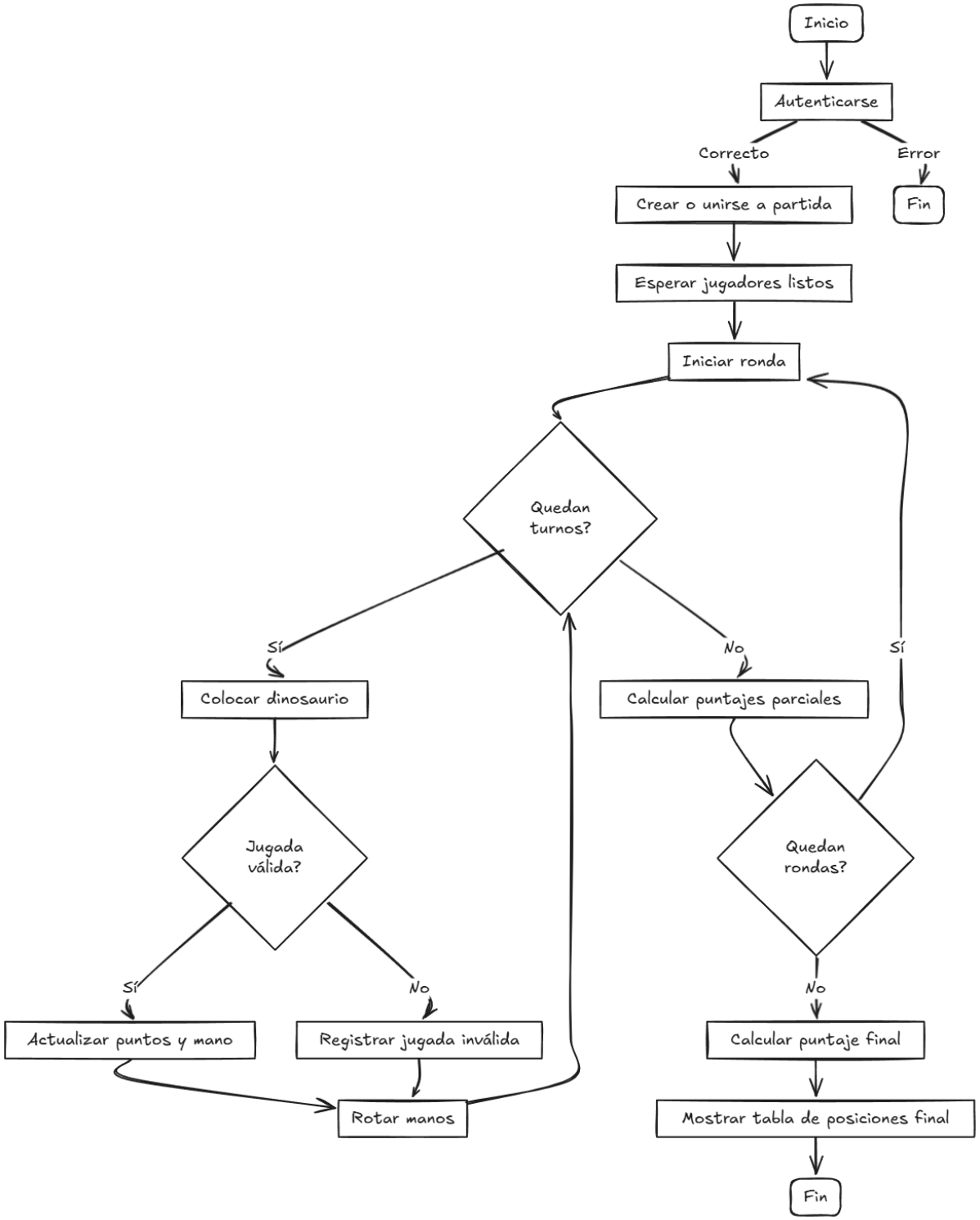


Diagrama de Gantt actualizado a segunda entrega

LINK A DIAGRAMA DE GANTT

Descargar e importar en onlinegantt.com