Feed The Leader

jajerje studios ®

1- Jorge Repullo Serrano - jorgers4@uma.es
2- Artur Vargas Carrión - arturvargas797@uma.es
3- Juan Manuel Valenzuela González - amcgil@uma.es
4- Eduardo González Bautista - edugb@uma.es
5- Rubén Oliva Zamora - rubenoliva@uma.es
6- Javier Toledo Delgado - javier.toledo.delgado@uma.es
7- Eloy González Castillo - eloygonzalez@uma.es

GR1-04
Ingeniería del Software A
Universidad de Málaga

ÍNDICE DE CONTENIDO

1. Introducción

Nuestra idea de proyecto se basa en la creación de un videojuego del género incremental o *clicker*, similar a otros juegos famosos como *Cookie Clicker*. La premisa de este tipo de juegos es muy simple: **asignar una tarea repetitiva al jugador** que le aporte puntos (por ejemplo, hacer clic, de ahí el nombre). Dicha tarea puede parecer repetitiva y aburrida en un principio, pero la idea es ir **avanzando progresivamente** consiguiendo multiplicadores y potenciadores que hagan el avance más satisfactorio.

En otras palabras, en un **juego tipo *clicker***, el jugador tan solo necesita hacer clic de forma repetida o realizar una acción equivalente con el objetivo de **obtener recursos virtuales**. Estos recursos pueden ayudar a mejorar personajes u otros elementos dentro de la temática del juego. A largo plazo, esta repetición constante permite obtener más recursos de manera progresiva.

En nuestro caso vamos a desarrollar un *clicker* llamado ***Feed The Leader***. El líder es un ser supremo que necesita **alimentarse para aumentar su poder**. Nuestra tarea será clicar para darle comida, cuanta más comida le demos, ganaremos más **puntos de fe** (FP) con los que podremos comprar mejores comidas para el líder, potenciadores para ganar más puntos por un tiempo limitado, multiplicadores que aumenten la cantidad de comida que podemos ofrecer al líder, etc.

Nuestro objetivo será ser el **mayor adorador del líder**. Esto se alcanza al llegar a una determinada cantidad de puntos de fe y comprar todas las mejoras. Cuando lleguemos a ese punto podremos **prestigiar**, lo que implica que **empezaremos de nuevo,** pero con uno o varios **potenciadores base,** que acelerarán el progreso.

Al alimentar al líder va **aumentando su peso**, lo que se verá reflejado en su **apariencia física**. Cuando el medidor de saciedad esté lleno significa que el líder está saciado. El líder **prefiere comer cuando no está saciado**, esto implica que la **cantidad de puntos obtenida será mayor** bajo estas condiciones ―y consecuentemente menor si lo sobrealimentamos―. Conforme la cantidad de comida que podamos ofrecer al líder y la frecuencia con la que podamos alimentarlo aumente, necesitaremos una forma de **incrementar la cantidad de comida que puede soportar**. Este problema lo soluciona la **prensa cuántica**, que reduce el tamaño de la comida acercando sus átomos sin que pierdan en el proceso su valor nutricional ni energético. Podremos mejorar la prensa con puntos de fe, generando así una dependencia entre las mejoras a la prensa, a los alimentos y demás.

1.1. ¿Por qué un *clicker*? ¿Por qué *Feed The Leader*?

En la danza constante de clics, en la repetición aparentemente trivial, hallamos un eco de la vida misma. Un videojuego incremental no es solo un entretenimiento; es un **recordatorio**. Como el sol que asoma cada día, como las olas que acarician la orilla sin cesar, la consistencia es la melodía que subyace en nuestra existencia.

En el *clicker*, encontramos la paradoja: lo pequeño se vuelve grande, lo insignificante se torna poderoso. Cada clic, como una gota en el océano, suma y construye. Así también en la vida, nuestros esfuerzos diarios, aunque modestos, moldean nuestro destino. La persistencia, como un hilo dorado, teje la trama de nuestras historias.

Aumentar la moneda de canje, clic tras clic, es un ritual que trasciende lo mundano. En cada pulsación, se esconde la promesa de un mañana más brillante. No es solo un juego, sino una lección: la grandeza no surge de actos grandiosos aislados, sino de la constancia, del compromiso inquebrantable con nuestra causa.

Que este juego sea un faro en la noche, una **invitación a la perseverancia**. Que cada clic sea un voto por la grandeza, un tributo a la persistencia. En el *clicker*, en la vida, la consistencia es la llave que desbloquea puertas insospechadas. Así, clic a clic, tejemos nuestra epopeya personal, recordando que hasta el más pequeño gesto puede resonar en la vastedad de la eternidad.

Además de una **oda a la consistencia**, nuestro juego, *Feed The Leader*, emerge como una ventana a la **crítica** hacia las complejas **estructuras de poder en la sociedad contemporánea**. Al interactuar con la tarea aparentemente simple de alimentar a un ser supremo, el juego nos incita a reflexionar sobre las diferentes capas de autoridad y sumisión que impregnan nuestras vidas. Cada clic en el juego puede interpretarse como un acto de complacencia hacia una figura de autoridad, generando así una oportunidad para examinar críticamente cómo nuestras acciones cotidianas pueden reforzar o desafiar las jerarquías existentes.

2. Roles

Durante el desarrollo del proyecto a cada integrante del grupo se le asignará un rol. Cada uno será el encargado de que ese trabajo salga bien. No obstante, eso no significa que el integrante solo se dedique a su rol y no ayude con los demás.

Entre los roles elegidos encontramos: coordinador, programador, tester, analista y arquitecto.

2.1. Descripción de los roles

El **coordinador** lidera la planificación y ejecución de proyectos, asignando tareas y recursos. Su rol implica mantener la motivación y colaboración del equipo, así como resolver conflictos. Coordinan la comunicación interna y externa, asegurando la alineación con los objetivos organizacionales. En resumen, el coordinador desempeña un papel clave en la gestión efectiva, liderazgo y éxito del proyecto. Necesitamos a personas que sepan tomar buenas decisiones y que soporten la presión.

El **programador** se encargará principalmente de escribir código utilizando diferentes lenguajes de programación. Su función principal es convertir los diseños y especificaciones proporcionados por los analistas y desarrolladores en software funcional. Necesitamos a gente que sea muy hábil con el lenguaje de programación a utilizar.

El **tester** se encarga de probar aplicaciones y sistemas para identificar defectos, asegurando su calidad y rendimiento. Sus responsabilidades incluyen la creación de casos de prueba, ejecución de pruebas manuales o automáticas, documentación de resultados y colaboración con desarrolladores para corregir problemas. Además, participa en la validación de requisitos y contribuye al mantenimiento de estándares de calidad en el ciclo de desarrollo. Su objetivo es garantizar que el software cumpla con los estándares de calidad establecidos antes de su lanzamiento. Necesitamos que los tester tengan muy claro el funcionamiento del producto y que sepan identificar rápidamente de dónde pueden venir los fallos.

El **analista** es responsable de analizar las necesidades del cliente y traducirlas en requisitos técnicos y funcionales para el equipo de desarrollo. Su función implica comprender los procesos comerciales y las necesidades del usuario final, así como documentar y comunicar claramente los requisitos al equipo de desarrollo. Necesitamos a gente que sepa entender las necesidades del cliente y traducirla fácilmente a una implementación en el producto a desarrollar.

El **arquitecto** se encarga de diseñar la estructura y la arquitectura técnica de sistemas y aplicaciones. Sus responsabilidades incluyen la definición de la estructura del software, la selección de tecnologías, la elaboración de patrones de diseño y la garantía de la coherencia y la escalabilidad del sistema. Además, colabora con analistas y otros miembros del equipo para asegurar que la arquitectura cumpla con los requisitos del proyecto. Necesitamos a gente que conozca diversas aplicaciones y técnicas que puedan ser útiles en el desarrollo del proyecto.

2.2. Asignación de roles

Tras evaluar las fortalezas y debilidades de cada integrante, se ha determinado que los roles de cada uno deberían ser los siguientes:

- **Jorge Repullo Serrano:** analista y tester.
- **Artur Vargas Carrión:** analista y coordinador.
- **Juan Manuel Valenzuela González:** tester y programador.
- **Eduardo González Bautista:** programador y tester.
- **Rubén Oliva Zamora:** coordinador y arquitecto.
- **Javier Toledo Delgado:** programador y arquitecto.
- **Eloy González Castillo:** arquitecto y analista.

3. Gestión de Riesgos

Antes de comenzar con el proyecto, debemos ser conscientes de la posibilidad de que no todo el proceso de desarrollo sea en línea recta. Para ello, realizamos una lista con los potenciales riesgos que pueden surgir durante nuestro trabajo, así como su clasificación, la probabilidad de que ocurra, las consecuencias y una posible estrategia para minimizar estas consecuencias.

Movilidad del personal

- Tipo: proyecto.
- Descripción: algún integrante del grupo por motivos personales tenga que abandonar el proyecto.
- Probabilidad: muy baja.
- Efecto del riesgo: serio.
- Estrategia para mitigarlo: mantener informados a los integrantes del grupo de posibles inconvenientes, para poder distribuir la carga de trabajo y prevenir cambios en la planificación.

Subestimación de la dificultad

- Tipo: proyecto y producto.
- Descripción: subestimar la dificultad del desarrollo del software necesario para la realización del proyecto.
- Probabilidad: alta.
- Efecto del riesgo: tolerable.
- Estrategia para mitigarlo: esperar que las tareas nos van a llevar más de lo habitual a la hora de planear cualquier evento, tener en cuenta experiencias de los integrantes del grupo y realizar cualquier trabajo con tiempo de antelación.

Mala planificación con los tiempos de entrega

- Tipo: organización.
- Descripción: mala organización a la hora de dividir la carga de trabajo que provoque una mala compatibilidad a la hora de dedicarle el tiempo al proyecto junto a otras asignaturas.
- Probabilidad: moderada.
- Efecto del riesgo: tolerable.
- Estrategia para mitigarlo: comenzar a trabajar en el proyecto desde el día uno, para que cualquier inconveniente que surja, poder solucionarlo a tiempo mucho antes de la entrega.

Competencia del producto

- Tipo: negocio.
- Descripción: otro grupo realiza un proyecto parecido, que puede provocar comparaciones o incluso que el profesor decida que el proyecto no salga adelante.
- Probabilidad: moderada.
- Efecto del riesgo: tolerable.
- Estrategia para mitigarlo: plantear una idea original o de difícil realización la cual sea complicada de replicar.

Las partes del código que se iban a reutilizar tienen defectos que limitan su funcionalidad

- Tipo: tecnológico.
- Descripción: cuando se trabaja con código, lo normal es reutilizar parte de estos, pues en este caso, suponemos que parte de este código no puede funcionar en todos los casos.
- Probabilidad: alta.
- Efecto del riesgo: insignificante.
- Estrategia para mitigarlo: desarrollar código consistente y coherente con muchos comentarios que ayuden en su comprensión, teniendo en cuenta la posible reutilización de estos escribiéndolos de la forma más genérica posible para que funcionen en un mayor rango de ámbitos.

Pérdida de tiempo en características poco relevantes

- Tipo: organización.
- Descripción: los desarrolladores emplean demasiado tiempo en implementar características alejadas del esqueleto del proyecto y no emplean el tiempo necesario en pulir las partes más importantes del mismo.
- Probabilidad: muy alta.
- Efecto del riesgo: serio.
- Estrategia para mitigarlo: centrarse en desarrollar el esqueleto del proyecto, y hasta que este no esté terminado no a implementar características menos relevantes.

Subestimación de la cantidad de errores en el software del proyecto

- Tipo: tecnológico.
- Descripción: los desarrolladores crean un código con demasiados errores que no permite continuar con otras partes del proyecto.
- Probabilidad: moderada.
- Efecto del riesgo: serio.
- Estrategia para mitigarlo: crear el software poco a poco, con muchos tests para reducir errores potenciales y con numerosas revisiones.

Bajo índice de detección de errores por parte de los tester

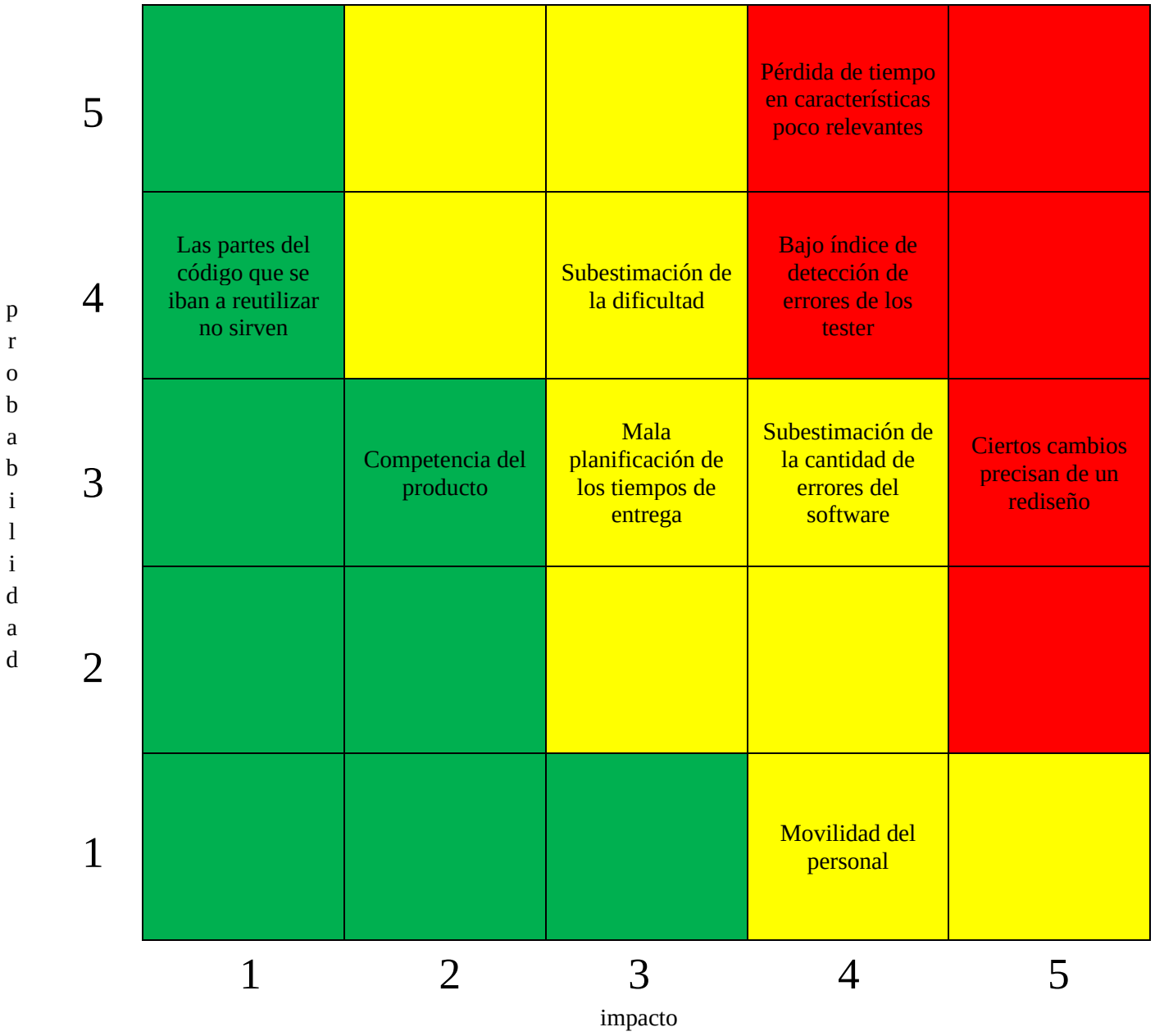
- Tipo: tecnológico.
- Descripción: los desarrolladores crean un código con demasiados errores que los tester no son capaces de encontrar.
- Probabilidad: alta.
- Efecto del riesgo: serio.
- Estrategia para mitigarlo: adoptar una estrategia de detección de errores efectiva por parte de los tester, realizando acciones poco comunes para encontrar el máximo número posible de errores.

Se proponen unos cambios de requisitos que necesitan un importante rediseño

- Tipo: organización.
- Descripción: el profesor o la forma de la estructura del trabajo, nos obliga a cambiar gran parte de lo que llevamos hecho.
- Probabilidad: moderado.
- Efecto del riesgo: catastrófico.
- Estrategia para mitigarlo: Amoldar la forma del trabajo para que los cambios no supongan un cambio de la estructura completa, y organizarnos correctamente los integrantes del grupo con objeto de evitar esto.

3.1. Matriz de riesgos

Para poder visualizar los riesgos, hemos decidido ilustrarlos en una matriz de riesgos:



4. Planificación

4.1. Modelo de proceso de software elegido

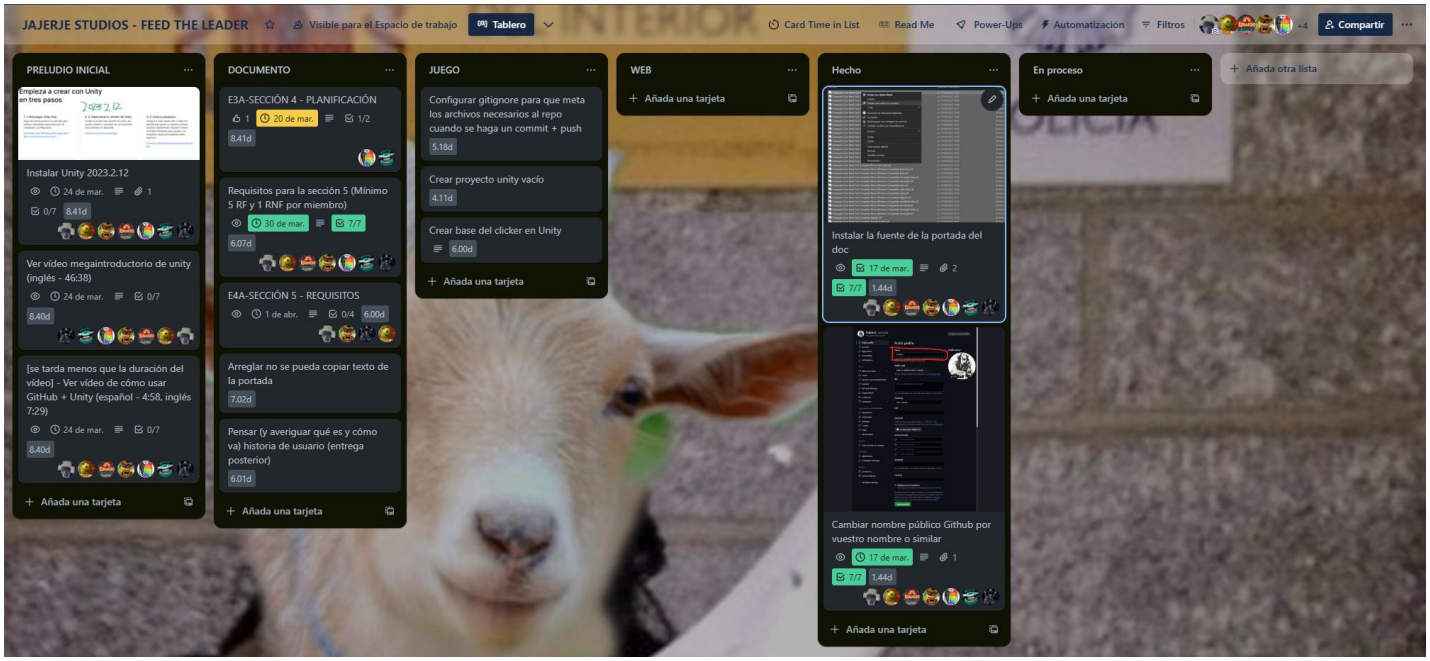
Se ha elegido la metodología ***Scrum*** porque es un modelo que se adecúa perfectamente a nuestras necesidades en el proyecto. Esta **metodología** es **iterativa** e **incremental**. Que sea iterativa quiere decir que el trabajo se divide en iteraciones, llamadas *sprints* en el caso de *Scrum*, que son períodos de tiempo predefinidos durante los cuales se realiza el trabajo planificado. Al final de cada *sprint*, se produce un incremento de software funcional y potencialmente entregable.

Nuestra organización de proyecto se crea gracias a las **reuniones regulares** en las que se decide la mejor organización posible para aumentar la productividad, y son lideradas por el "coordinador principal", llamado en la metodología *Scrum*, *Scrum Master*. Tenemos una comunicación constante y efectiva, con continuas interacciones, lo cual nos permite tener mejores análisis de la situación en cada momento del proyecto.

El uso de este método nos proveerá una mayor auto-organización, pues cada tarea asignada a uno o varios miembros es resuelta por el método que estos consideren. Además de una mayor autonomía y auto-superación, el proyecto se mejora progresivamente. De todo esto, concluimos en un enriquecimiento de todos los miembros del grupo y una transmisión del conocimiento constante.

4.2. Organización mediante tableros

Hemos usado la aplicación web Trello para organizar las tareas mediante tableros y tarjetas.



1 Tablero de Trello

Nuestro tablero de Trello tiene, actualmente, varias columnas. En primer lugar, encontramos la columna ***Preludio inicial***. En esta columna, se observan tareas que no forman parte de ninguna entrega, como

por ejemplo la instalación de *Unity*, ver vídeos para aprender lo básico sobre *Unity* y *GitHub*, instalar fuentes de Word, etc.

A continuación, tenemos la columna ***Documento***. Aquí están todas las tareas relacionadas con la creación del documento del proyecto, como pueden ser la realización las diferentes secciones por entregar cada semana, fallos a corregir de cada sección u otras partes del documento, o "subtareas" que cada integrante debe hacer.

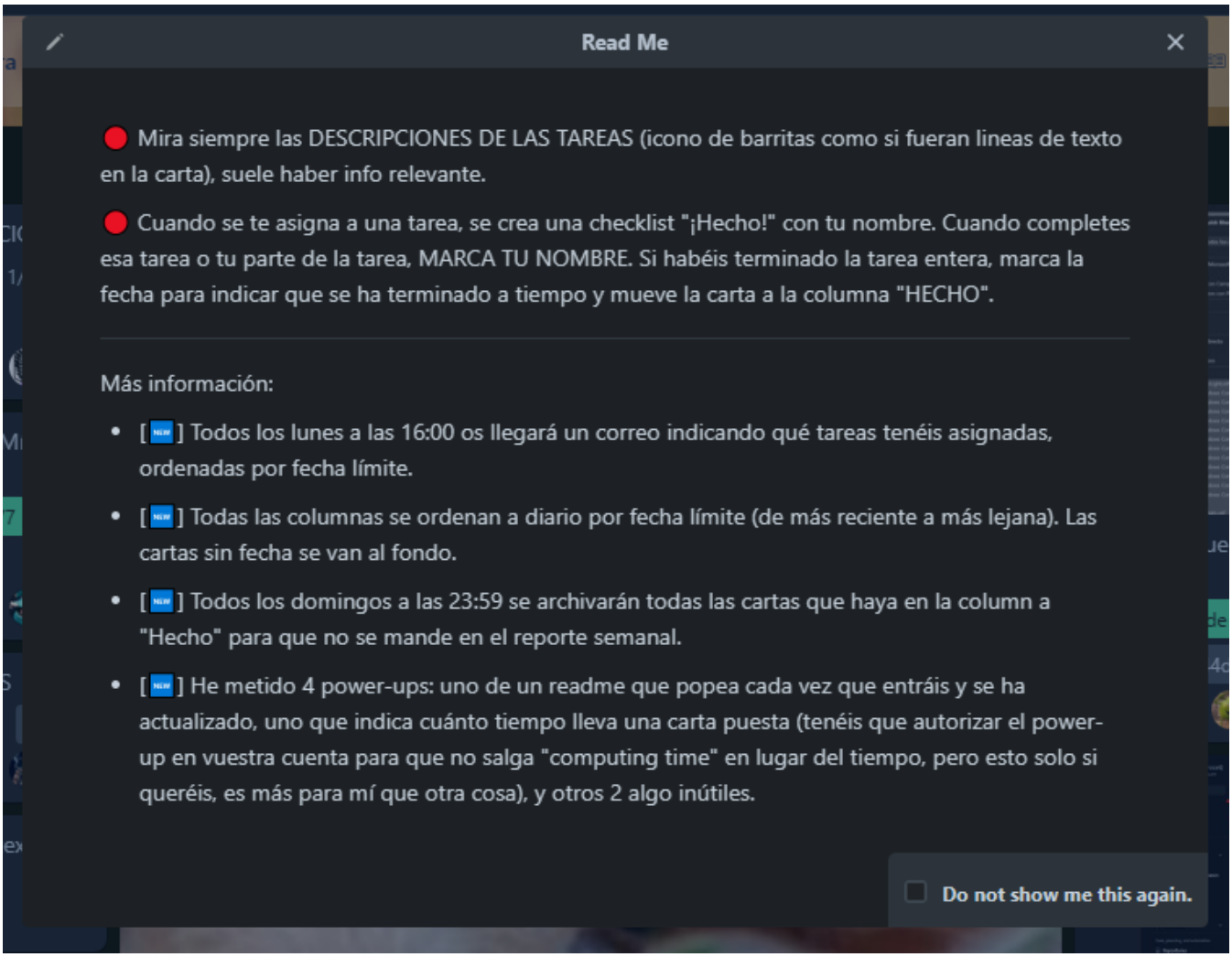
Con respecto a la columna ***Juego***, como su nombre indica, se encuentran las actividades con la realización de nuestro juego. Actualmente, vemos tareas básicas como son la creación de un proyecto vacío en *Unity*, la base del *clicker* y la configuración de *gitignore*.

Más a la derecha tenemos las columnas ***Web***, aún vacía, ***Hecho***, donde se irán introduciendo aquellas tareas que estén terminadas, y ***En proceso***, donde se introducirán tareas parcialmente terminadas o a medio hacer.

4.3. Power-ups

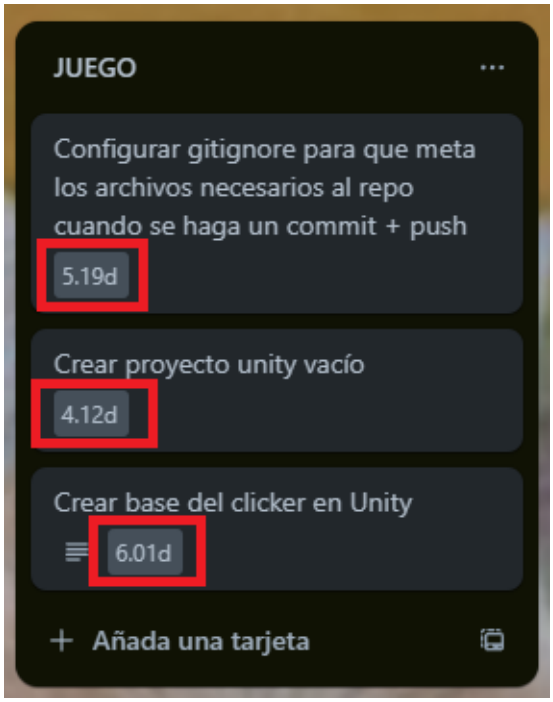
Se ha decidido añadir los siguientes *power-ups* para conseguir una organización todavía más efectiva:

- **Readme:** usado para dar información general sobre el funcionamiento de este espacio de trabajo.



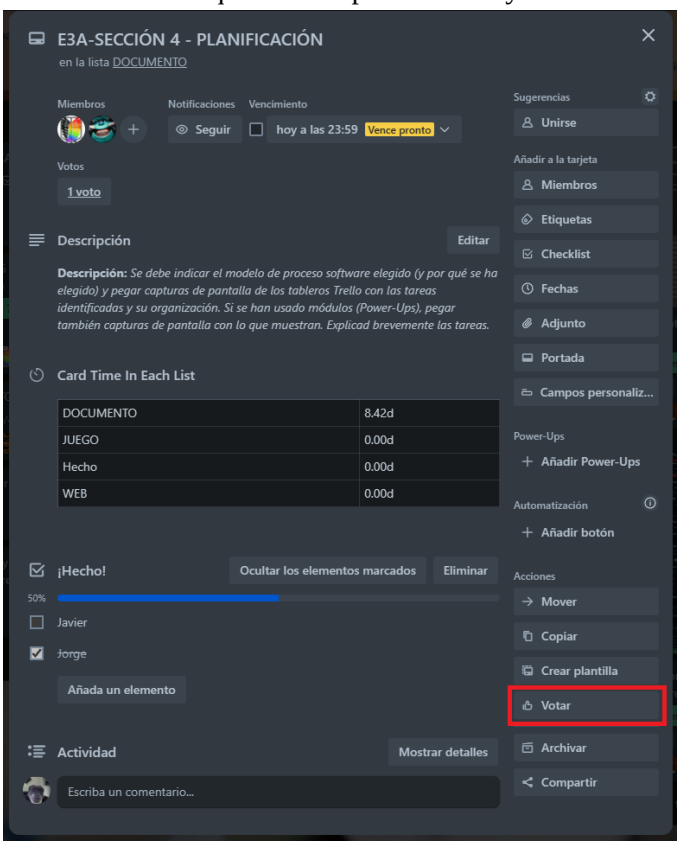
2 Pop-up de Read Me

- **Card time:** usado para ver el tiempo que una tarea lleva publicada en el tablero Trello.



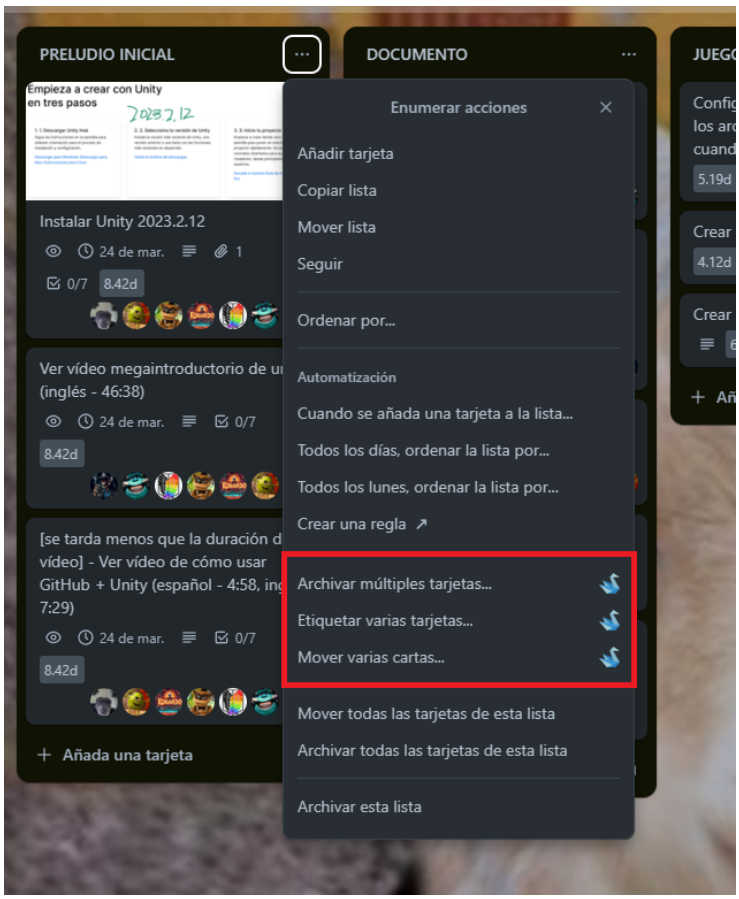
3 Muestra del power-up Card time

Votar: los miembros del grupo puedan votar si una tarea es útil para comunicar de forma efectiva que son conscientes de que forman parte de ésta y están de acuerdo con ella.



4 Muestra del power-up "Votar"

- **Manny:** añadido para ganar eficiencia a la hora de planificar en Trello. Permite mover múltiples tarjetas simultáneamente.

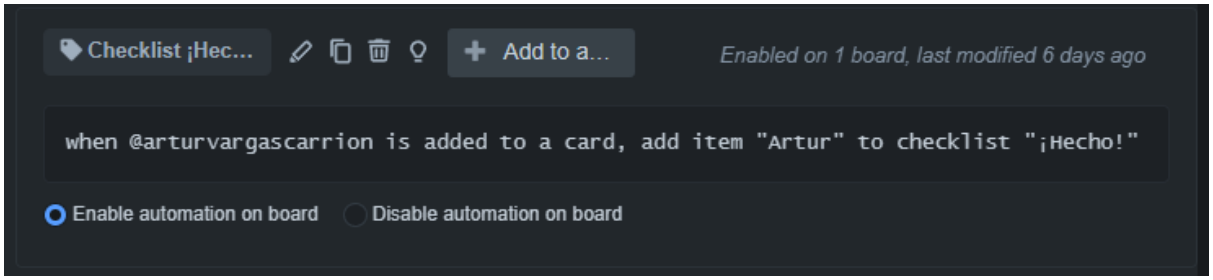


5 Muestra del power-up "Manny"

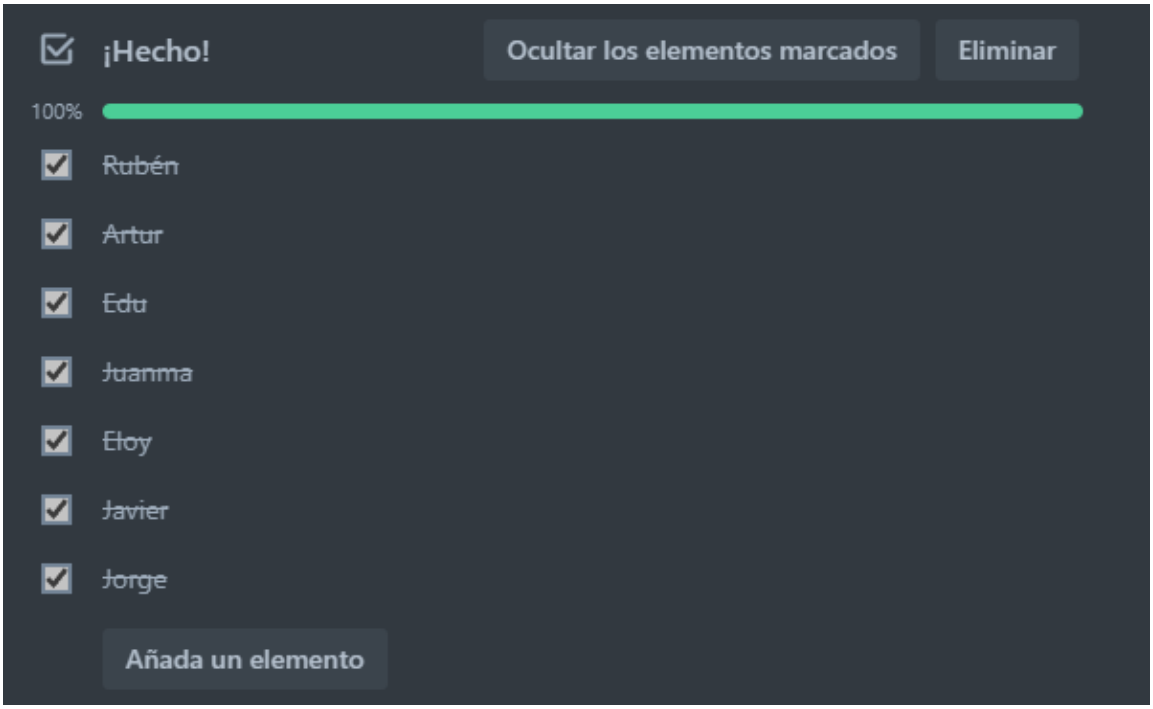
4.4. Automatizaciones

Como ocurre con el power-up Manny, se han creado diferentes automatizaciones con el fin de mejorar la eficiencia a la hora de organizar el proyecto.

- **Checklist:** cuando se añade un miembro a una tarea, automáticamente se crea una lista con el nombre de esa persona. Esta lista sirve para cuando un miembro completa su parte, tacha su nombre de la lista, así se ve visualmente quién ha terminado cada parte.

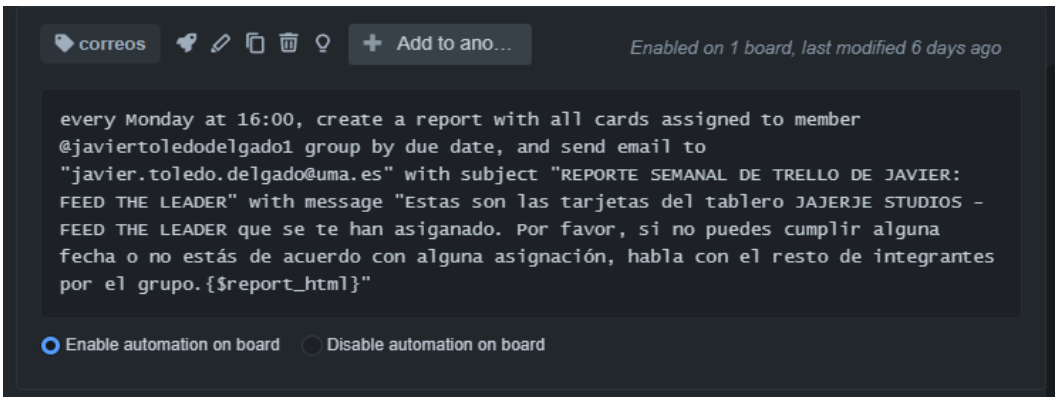


6 Automatización para crear el Checklist ¡Hecho!



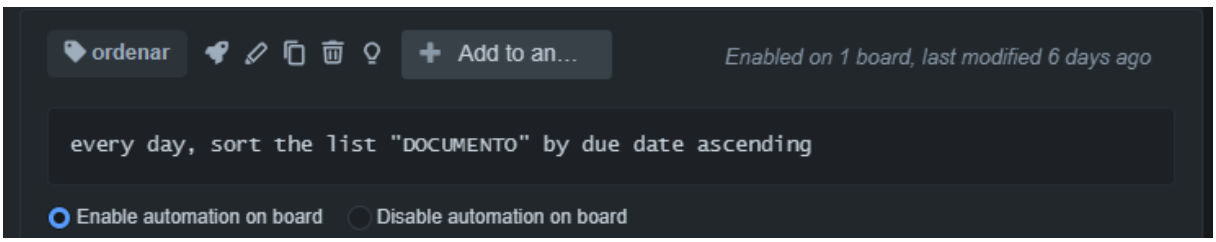
7 Checklist ya creado

- **E-mail semanal:** se ha creado una automatización para que a todos los miembros les llegue un correo semanal en el que se detallan todas las tareas no completadas de las que forma parte.



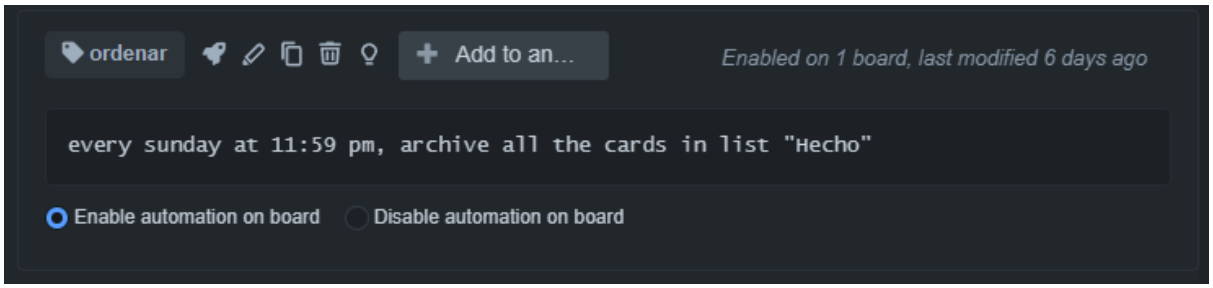
***8** Automatización para enviar correos*

- **Ordenar tarjetas por fecha límite:** cada tarjeta de cada lista se ordena según la fecha de vencimiento diariamente gracias a esta automatización.



9 Automatización para ordenar las listas según su fecha de entrega

- **Archivo automático de tareas:** semanalmente se archivarán automáticamente las tareas que ya hayan sido completadas.

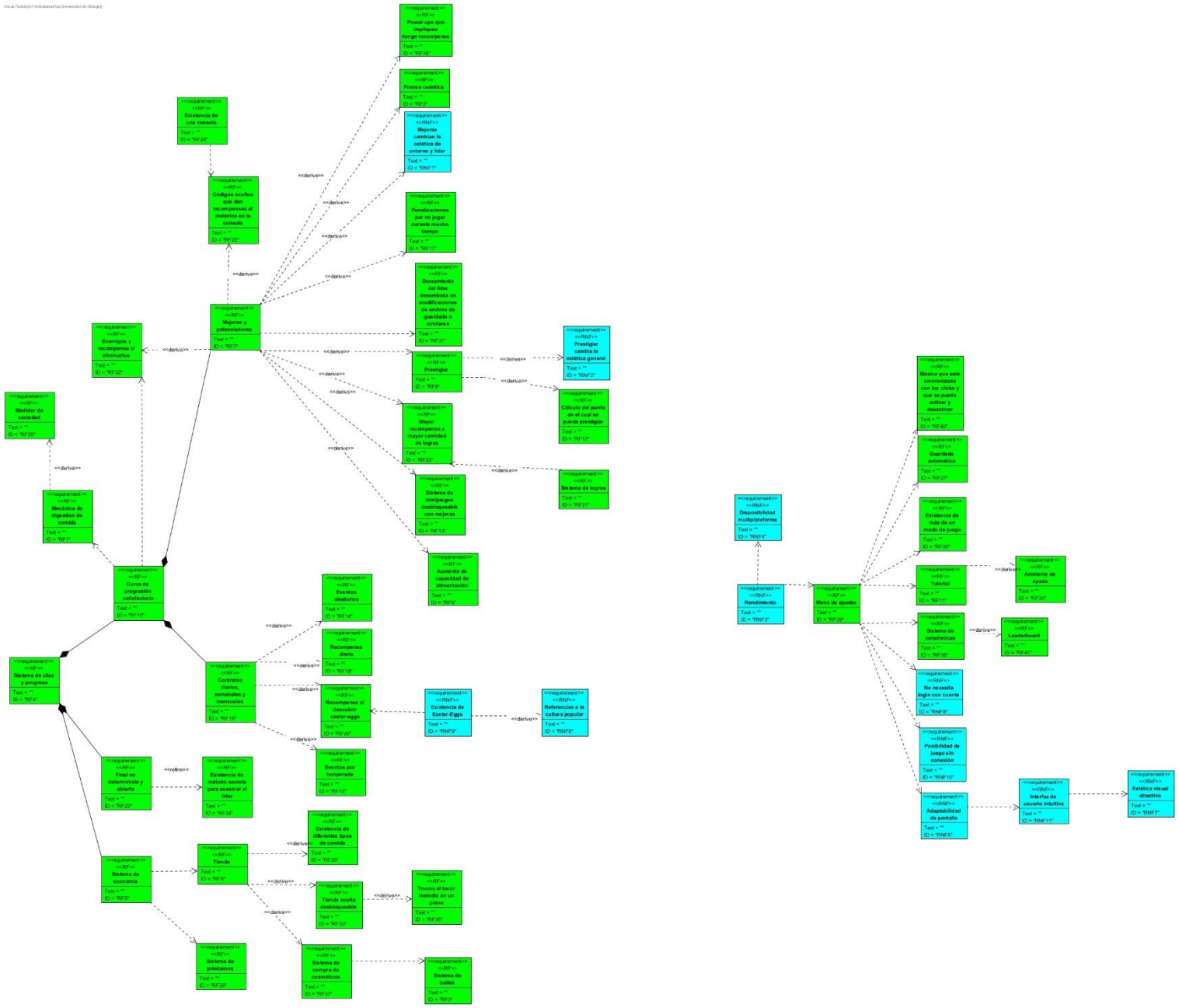


10 Automatización para el archivo de tarjetas

5. Requisitos

En la concepción y desarrollo de cualquier proyecto de software, la comprensión y especificación de requisitos son fundamentales para su éxito. En esta sección, exploraremos en detalle los requisitos que guiarán el diseño y la implementación del videojuego. Abordaremos tanto los requisitos funcionales, que describen las funciones específicas que el sistema debe realizar, como los requisitos no funcionales, que se centran en atributos de calidad como el rendimiento, la seguridad y la usabilidad.

A través de un enfoque sistemático, examinaremos cómo identificar, analizar y documentar estos requisitos para asegurar que el producto final cumpla con las expectativas y necesidades del usuario. Para una comprensión más visual, presentamos un diagrama de requisitos elaborado con Visual Paradigm.



11 Diagrama de requisitos creado en Visual Paradigm

5.1. Requisitos funcionales

RF1: Mecánica de digestión de comida

Como jugador
Quiero que el videojuego tenga mecánicas que me mantengan jugando
Para retener mi atención y conseguir que siga empleando mi tiempo en jugar.

Pruebas de aceptación

- Alimentar al líder con comida debe mostrar una animación o indicador visual que represente la digestión del alimento.
- La cantidad de puntos de fe obtenidos al alimentar al líder debe ser proporcional a la cantidad de comida digerida.
- La barra de saciedad debe incrementarse conforme se alimente al líder.

RF2: Prensa cuántica

Como jugador
Quiero mecánicas interesantes que hagan ameno el progreso
Para querer seguir jugando al juego.

Pruebas de aceptación

- La prensa cuántica debe ser un ítem disponible en la tienda del juego, que además debe poderse mejorar.
- Al comprar la prensa cuántica, la cantidad de calorías que el líder puede consumir debe aumentar significativamente.

RF3: Sistema de bailes

Como jugador
Quiero un sistema de bailes para el líder
Para poder divertirme con una mecánica que ridiculice al personaje principal.

Pruebas de aceptación

- Las animaciones serán fluidas y manteniendo la estética del personaje.
- Los bailes no taparán ningún aspecto importante del juego.
- Habrá un botón debajo del líder que desplegará los bailes disponibles, así como los que se podrán desbloquear.

RF4: Sistema de clics y progreso

Como jugador
Quiero que haya un sistema de progresión
Para querer avanzar en el juego.

Pruebas de aceptación

- Cada clic en el líder debe aumentar los puntos de fe del jugador.
- El progreso en el juego debe estar directamente relacionado con la cantidad de puntos de fe acumulados a través de los clics.

RF5: Sistema de economía

Como jugador
Quiero ver que mis puntos tienen un uso
Para querer emplearlos y seguir consiguiendo más.

Pruebas de aceptación

- El juego debe mostrar claramente la cantidad de puntos de fe disponibles para el jugador en todo momento.
- Con la compra de mejoras, se deben de obtener más puntos de fe por clic.
- Deben existir automatizaciones que permitan obtener puntos de fe sin la participación activa del usuario.

RF6: Tienda

Como jugador
Quiero tener una mecánica de tienda
Para sentirme partícipe de la progresión dando uso a los puntos comprando artículos que puedo ver físicamente en el juego.

Pruebas de aceptación

- Los elementos de la tienda deben estar organizados de manera lógica y fácil de entender para el jugador.
- Al comprar un artículo de la tienda, la cantidad de puntos de fe del jugador debe disminuir correctamente y el artículo debe estar disponible.
- No deben poderse comprar puntos de fe.

RF7: Mejoras y potenciadores

Como jugador
Quiero una variedad de mejoras y potenciadores
Para emplear los puntos y que aumente la eficiencia de la alimentación y el progreso en el juego.

Pruebas de aceptación

- Cada mejora o potenciador adquirido en la tienda debe tener un efecto claro y medible en el progreso del juego.
- Las mejoras y potenciadores deben escalarse de manera que su eficacia aumente con cada compra.

RF8: Prestigiar

Como jugador
Quiero una mecánica de prestigio
Para que el videojuego sea rejugable y además lo haga más interesante volver a completarlo.

Pruebas de aceptación

- Al elegir la opción de prestigiar, el progreso del jugador debe reiniciarse mientras que las mejoras bases se incrementan como se prometió.
- El jugador debe sentir una mejora significativa en la velocidad de progreso tras prestigiar.

RF9: Aumento de capacidad de alimentación

Como jugador
Quiero poder alimentar más al líder
Para que exista progresión en el juego y querer conseguir mejoras.

Pruebas de aceptación

- Las mejoras relacionadas con la capacidad de alimentación deben tener un impacto perceptible en la cantidad de comida que el líder puede consumir, y, por tanto, en los puntos de fe que se pueden conseguir.
- Las mejoras de capacidad de alimentación deben ser equilibradas para evitar romper el juego.

RF10: Curva de progresión satisfactoria

Como jugador
Quiero sentir que el avance en el juego es progresivo y satisfactorio
Para que el avance no sea aburrido, exagerado o inexistente.

Pruebas de aceptación

- La curva de progresión del juego debe diseñarse de manera que los jugadores sientan un aumento constante en el desafío y la recompensa a medida que avanzan, manteniéndolos enganchados con una constante sensación de recompensa.
- La dificultad del juego debe aumentar gradualmente para mantener el interés del jugador sin volverse abrumadora.
- Deben existir incentivos en caso de que el jugador deje de jugar durante un periodo de tiempo considerable (horas o incluso días).

RF11: Tutorial

Como jugador
Quiero tener una noción básica de cómo funciona el juego, preferiblemente mediante un asistente
Para entender la interfaz y las mecánicas.

Pruebas de aceptación

- El tutorial será una mascota que aparecerá cuando el jugador haga clic en algo que no sepa cómo funciona o que añada alguna característica.
- Estará siempre presente en una esquina de la pantalla para que se pueda clicar en caso de que no recuerdes algo.
- Si se clica se abre un menú con opciones de diálogo.
- Si no se clica el personaje se mantiene en una esquina con una animación sencilla.

RF12: Cálculo del punto en el cual se puede prestigiar

Como jugador
Quiero sentir que el prestigio es una mecánica bien implementada
Para que la progresión tenga una recompensa "final".

Pruebas de aceptación

- La curva de progresión debe obligar al jugador en algún punto del juego, a desbloquear las mejoras de prestigio.
- Esto será algo visible para el jugador, que notará que el progreso es demasiado lento.
- A medio camino de necesitarlo, la mascota se comunicará con el jugador para informarle de la existencia de esta opción y cuál es su cometido, de modo que el jugador siempre sepa cuáles son sus opciones disponibles.

RF13: Sistema de minijuegos desbloqueables con mejoras

Como jugador
Quiero minijuegos alternativos al juego base
Para hacer la experiencia y la progresión menos monótona y más divertida.

Pruebas de aceptación

- Ciertas mejoras llevan consigo un desplegable que abre un minijuego.
- Para desbloquear el minijuego, se debe haber mejorado esa mejora hasta cierto nivel. Si está bloqueado, el desplegable aparece bloqueado también, para que se sepa que esa mejora posee un minijuego.
- Durante el minijuego, la producción de puntos no se detiene.
- Se puede cerrar el desplegable en cualquier momento.

RF14: Eventos aleatorios

Como jugador
Quiero que ocurran eventos espontáneamente
Para quedar sorprendido y que se rompa la monotonía del juego con regularidad.

Pruebas de aceptación

- Arbitrariamente aparecen eventos por pantalla, que provocarán animaciones.
- Estos eventos, desencadenan minijuegos que tendrán un efecto notable y claro en el juego.
- Si el jugador no decide hacer lo que se le indique, puede ser castigado.
- Una vez terminado el evento, la animación termina y el juego vuelve a la normalidad (con castigo o recompensa).

RF15: Eventos por temporada

Como jugador
Quiero sentir y notar cada estación de la vida real en el juego
Para sentir al líder y el juego más conectado a mi vida y a la realidad.

Pruebas de aceptación

- Cambian la apariencia de la interfaz.
- Se desbloquean nuevas mejoras y formas de conseguir puntos de fe.
- Una vez terminado, las mejoras de temporada desaparecen, los puntos conseguidos se mantienen y el juego vuelve a la normalidad.

RF16: Power-ups que impliquen riesgo-recompensa

Como jugador
Quiero tener que barajar opciones y tomar decisiones
Para aumentar la intensidad del juego y arriesgarme para conseguir mejores recompensas con el objetivo de disfrutar más de ellas.

Pruebas de aceptación

- Ciertas mejoras llevan consigo un desplegable que abre un menú de power-ups.
- Para desbloquear los power-ups, se debe haber mejorado esa mejora hasta cierto nivel, si está bloqueado el desplegable aparece bloqueado también, para que se sepa que esa mejora posee power-ups.
- Activarlos es gratis, pero pueden provocar tanto efectos positivos como negativos.
- Estos power-ups tienen un tiempo determinado y el jugador no puede decidir eliminarlo siempre que quiera.

RF17: Penalizaciones por no jugar durante mucho tiempo
Como desarrollador
Quiero que el jugador sienta la necesidad de jugar
Para que siempre esté atento al juego y a las actualizaciones que se implementen, así como los eventos y recompensas.

Pruebas de aceptación

- Vienen relacionadas con el tiempo real en la Tierra, ninguno interno del juego.
- El jugador, debe sentir un deber de alimentar al líder en su experiencia de vida.
- Prescindir de alimentar al líder, llevará consigo eventos que provocarán un perjuicio en la producción de puntos de fe.
- El juego avisará al jugador al entrar después del tiempo definido como, "no jugar por mucho tiempo", del impacto que ha tenido no jugar y advertirá de no volver a fallar al líder.

RF18: Recompensas diarias
Como desarrollador
Quiero que el jugador sienta que jugar diariamente le proporcione recompensas
Para asegurar que no se siente obligado a entrar, sino que tiene ganas de hacerlo.

Pruebas de aceptación

- Al entrar al juego por primera vez en el día, se abrirá un menú que informará de una recompensa que obtiene el jugador.
- Este menú también informará de las recompensas que obtendrá si sigue entrando todos los días de aquí a una semana.
- Una vez reclamada la recompensa, este menú se cierra.

RF19: Contratos diarios, semanales y mensuales

Como desarrollador
Quiero un compromiso del jugador con el juego
Para asegurar tiempos más prolongados de juego.

Pruebas de aceptación

- En un lado de la pantalla. habrá un botón que abra una interfaz la cual informará al jugador de las misiones disponibles.
- Este menú, debe de ser claro y grande, indicando sin generar duda, de cuáles son las misiones del día, de la semana y del mes y cuáles son las recompensas por completar cada una.
- Cuanto más duren las misiones, mayores serán los tiempos de compleción, así como las recompensas.
- Al completar un contrato, aparecerá un tic sobre este que podrás clicar para reclamar la recompensa del mismo. Al reclamarla este contrato desaparece.
- Una vez cerrado el menú, se podrá acceder otra vez a él pulsando el botón.

RF20: Existencia de diferentes tipos de comida

Como jugador
Quiero variedad en las opciones del juego
Para no aburrirme y mantener el juego fresco el mayor tiempo posible.

Pruebas de aceptación

- Cada una de estas comidas, tendrá un sprite diferente, que la diferenciará de las demás, así como de un nombre único.
- Al pasar el ratón por encima de la comida, se indicará claramente como prepararla y los beneficios de la misma.
- Al quitar el ratón, la información desaparecerá.
- Estás comidas, van almacenadas en un libro de recetas, que no es más que otro menú desplegable que puedes abrir y cerrar y que se irá rellenando conforme el jugador, obtenga nuevos alimentos.

RF21: Sistema de logros

Como jugador
Quiero objetivos en el juego
Para sentir que progreso y llego a metas.

Pruebas de aceptación

- Cuando el jugador complete un logro, se emitirá un sonido que lo indicará, además se aparecerá una pestaña por pantalla, que informará cuál se ha completado.
- Los logros, cómo conseguirlos y sus beneficios, se podrán consultar en el libro de recetas.
- Estos logros, deben incentivar al jugador a seguir jugando y a conseguirlos a través de sus recompensas, pero sobre todo a aquellos que les guste hacer el 100% de los juegos.

RF22: Final no determinista y abierto

Como jugador
Quiero diferentes formas de acabar mi aventura
Para sentir que yo decido el cómo termina mi historia y la del líder.

Pruebas de aceptación

- Comprobar que, al alcanzar cierto hito o estado en el juego que indica el final, se desbloquea un logro correspondiente y puede ser visible por el jugador.
- Verificar que el juego continúa después del final, permitiendo al jugador seguir interactuando de alguna manera.
- Comprobar que todas las mejoras, puntos y valores se reinician al empezar el juego de nuevo.

RF23: Mayor recompensa a mayor cantidad de logros

Como jugador
Quiero una progresión más rápida y acelerada después de mucho tiempo jugando
Para no sentir que me estanco en el juego, sino que cada vez es más frenético y divertido.

Pruebas de aceptación

- Realizar pruebas para confirmar que a medida que se desbloquean más logros, el porcentaje de recompensa aumenta de manera adecuada.
- Verificar que las mejoras que otorgan los logros no desbalanceen el juego de sobremanera.

RF24: Existencia de una consola

Como tester y jugador
Quiero una forma de controlar el juego con mayor facilidad
Para ahorrarme los largos tiempos de juego y probar cosas concretas, o explorar el juego sin necesidad de disfrutarlo.

Pruebas de aceptación

- Verificar que la consola está presente en la interfaz del juego y se puede interactuar con ella.
- Confirmar que la consola no tiene funcionalidad real en cuanto al funcionamiento o código y solo simula interacción para activar eventos dentro del juego.
- Verificar que los comandos introducidos desencadenen los eventos esperados por los desarrolladores.

RF25: Códigos ocultos que dan recompensas al meterlos en la consola

Como jugador
Quiero recompensas por descubrir secretos
Para que conseguirlos sea algo mágico y satisfactorio, con idea de que lo obtenido, es algo que no todo el mundo tiene y que se siente único.

Pruebas de aceptación

- Probar la inserción de códigos específicos en la consola y verificar que proporcionan las recompensas o eventos esperados.
- Confirmar que los códigos están ocultos y no son fácilmente accesibles sin exploración.
- Comprobar que no se puede acceder a los códigos de ninguna forma que no sea mediante la exploración preparada.

RF26: Recompensas al descubrir *Easter eggs*

Como jugador
Quiero recompensas por descubrir secretos
Para que mi exploración y compromiso con el juego se vea premiada.

Pruebas de aceptación

- Verificar que se pueden buscar *Easter eggs* en el juego y confirmar que al descubrirlos se otorgan recompensas.
- Asegurarse de que los *Easter eggs* están bien ocultos, pero no imposibles de encontrar.
- Implementar alguna mecánica para interactuar con los *Easter eggs*, ya sea hacer clic, meter un comando, etc.

RF27: Guardado automático

Como jugador
Quiero un sistema de guardado sencillo y fácil
Para no perder el tiempo con conceptos externos a la propia acción de jugar.

Pruebas de aceptación

- Verificar que el progreso del juego se guarda automáticamente a intervalos regulares o en puntos clave.
- Confirmar que el guardado automático no interrumpe la experiencia del jugador.
- Comprobar que el guardado no hace conflicto con alguna otra mecánica del juego.
- Opcionalmente agregar algún indicativo visual al jugador mientras esté en progreso el autoguardado.

RF28: Sistema de préstamos

Como jugador
Quiero poder conseguir puntos sin tenerlos
Para acceder antes a mejoras, sin tener que esperar tanto y pagarlo más adelante.

Pruebas de aceptación

- Probar el proceso de solicitar y pagar préstamos en el juego.
- Asegurarse de que el sistema funcione correctamente y que los préstamos se otorguen y paguen según lo previsto.
- Comprobar que existe algún tipo de penalización al no pagar los préstamos en un tiempo o momento determinado.

RF29: Medidor de saciedad

Como jugador
Quiero una mecánica que regule la cantidad de clics que tengo que hacer
Para poder tomarme descansos y utilizar otras mecánicas del juego.

Pruebas de aceptación

- Confirmar que el medidor de saciedad refleje de manera precisa el nivel de satisfacción del líder.
- Verificar que el anti-autoclicker responda adecuadamente a los intentos de automatizar la interacción con el juego.
- Comprobar que alimentar al líder cuando ya esté saciado otorga menos puntuación.
- Opcionalmente agregar algún evento o interacción negativa cuando el líder lleva mucho con el medidor vacío.

RF30: Asistente de ayuda

Como jugador
Quiero un asistente que me guíe
Para saber qué siguiente paso tomar en el juego.

Pruebas de aceptación

- Probar la interacción con el asistente a lo largo del juego para confirmar que proporciona orientación y consejos relevantes.
- Verificar que al final del juego, el asistente se revele contra el jugador y se una al líder de acuerdo con la descripción del requisito.
- Comprobar que el asistente no es demasiado invasivo para el jugador y no perturba demasiado la experiencia del juego base.
- Confirmar que existe una alternativa a que se revele el asistente (relacionado con el final bueno).

RF31: Descontento del líder desemboca en modificaciones de archivo de guardado o similares

Como jugador
Quiero que el juego rompa la cuarta pared
Para hacer la experiencia del juego más divertida y diferente a otros productos del mercado.

Pruebas de aceptación

- Si no le das comida en más de x tiempo (por ejemplo, tres horas usando la aplicación) empieza un evento que pueda cambiar algún archivo de guardado o cerrar el juego.
- Si le das comida en menos del tiempo x, no ocurre nada.
- Cuando ocurra el evento, se notificará al usuario de alguna forma antes de que ocurra.

RF32: Enemigos y recompensa al eliminarlos

Como jugador
Quiero enemigos que penalicen mientras existan y recompensen al eliminarlos
Para romper la monotonía que pueda llegar a tener el juego y tener una mecánica entretenida más.

Pruebas de aceptación

- Si aparecen los enemigos, ocurre una penalización hasta que se eliminan.
- Si no aparecen, no ocurre nada.
- Si los clicas menos veces de las necesarias para eliminarlos, continúan la penalización.
- Si los eliminas, te dan una recompensa y termina la penalización.

RF33: Tienda oculta desbloqueable

Como jugador
Quiero una tienda secreta que dé recompensas especiales
Para aumentar la curva de progresión y añadir contenido interesante al juego.

Pruebas de aceptación

- Para desbloquear la tienda, es necesario usar una característica oculta.
- Si esta característica no se ha usado, no existirá la tienda.
- Al comprar algún objeto, la divisa disminuirá y el objeto se añadirá al inventario.
- Si no se compra un objeto no ocurre nada.
- Una vez desbloqueada, se tendrá que abrir la tienda con un botón en el menú o similar.

RF34: Existencia de método secreto para asesinar al líder

Como jugador
Quiero una forma de derrocar al líder
Para tener una mecánica distinta que permita una historia alternativa.

Pruebas de aceptación

- Una vez realizado el método, se iniciará un evento en el que el líder muera y se cambiará por tu personaje.
- Si este método no se ha realizado, no ocurrirá nada.

RF35: Trucos al tocar melodía en un piano

Como jugador
Quiero trucos desbloqueables con una secuencia de teclas secreta
Para hacer la experiencia menos monótona y más divertida.

Pruebas de aceptación

- Si se toca una cierta melodía, ocurrirá un evento en el juego.
- Si no se toca la melodía designada, o no se toca nada, no ocurre nada.
- Una vez usada una melodía, al volver a tocarla no ocurrirá nada.

RF36: Existencia de más de un modo de juego

Como jugador
Quiero distintos modos de juego
Para romper la monotonía del modo principal y poder dedicarle más horas al juego.

Pruebas de aceptación

- En cada modo debe haber una opción para cambiar a un modo distinto.
- Si no se usa la opción para cambiar el modo, se seguirá en el actual indefinidamente.
- Cada modo tendrá características que lo diferencien de los demás.

RF37: Sistema de compra de cosméticos

Como jugador
Quiero un sistema de compra de cosméticos
Para poder personalizar la interfaz del juego.

Pruebas de aceptación

- Si se compra un cosmético, este debe salir desbloqueado en el menú correspondiente.
- Si no se compra el cosmético, saldrá bloqueado y posiblemente oculto.
- Una vez comprado, el cosmético debe poder activarse y cambiarse desde el menú.

RF38: Sistema de estadísticas

Como jugador
Quiero un sistema de estadísticas
Para saber el progreso que he hecho en el tiempo que he estado jugando.

Pruebas de aceptación

- El sistema de estadística se abrirá solamente pulsando el botón designado.
- Si no se pulsa el botón, no ocurre nada.
- Los valores estadísticos se actualizarán automáticamente.

RF39: Menú de ajustes

Como jugador
Quiero un menú de ajustes
Para personalizar la configuración del juego a mis preferencias.

Pruebas de aceptación

- El menú se abre con un botón en la pantalla principal.
- Si no se pulsa el botón, no ocurre nada.
- Una vez en el menú, debe haber diferentes botones que permitan personalizar los ajustes del juego.

RF40: Música que esté sincronizada con los clics y que se pueda activar y desactivar

Como jugador
Quiero música especial
Para mejorar la experiencia del juego.

Pruebas de aceptación

- La música debe sonar automáticamente sin necesidad de activarla.
- Debe haber posibilidad de desactivarla en el menú de ajustes anterior.
- Si no se hace clic, la música permanecerá en su modo normal.
- Si se hace clic, la música sufrirá un cambio con relación al clic.

RF41: Leaderboard

Como jugador
Quiero una tabla con los mejores jugadores
Para comparar mis estadísticas con las de los mejores jugadores.

Pruebas de aceptación

- En la tabla solo se mostrarán los mejores jugadores y el perfil observador, en caso de que no esté en los mejores valores.
- Cada jugador que ha usado el programa debe poder salir en la leaderboard.

5.2. Requisitos no funcionales

RNF1: Mejoras cambian la estética de entorno y líder

Como jugador
Quiero notar que mis mejoras aparecen de forma visual
Para hacer más amena la progresión y querer seguir mejorando.

Pruebas de aceptación

- Comprando ciertas mejoras cambiará algún aspecto de la pantalla, pero no taparán ninguna información.
- Hay mejoras que son compatibles y se podrán ver las dos por pantalla.
- Hay mejoras que no son compatibles y una sustituirá a la otra.

RNF2: Prestigiar cambia la estética general

Como jugador
Quiero cambios visuales al hacer prestigio
Para notar que merece la pena llegar a ese punto del juego ya que es una mejora difícil.

Pruebas de aceptación

- Llegando a cierto nivel de prestigio aparecerán detalles en la pantalla sin tapar información.
- Si el prestigio vuelve a bajar esos detalles pueden desaparecer.

RNF3: Rendimiento

Como jugador
Quiero que funcione bien el juego
Para que la experiencia sea satisfactoria y divertida.

Pruebas de aceptación

- El juego deberá mantener unos FPS estables en la medida de lo posible.
- Si se detecta que el juego va muy lento se eliminarán algunos efectos visuales de la pantalla que no sean muy perceptibles.
- Si el juego no puede seguir ejecutándose se cerrará y lanzará un mensaje de error.

RNF4: Disponibilidad multiplataforma

Como jugador
Quiero poder jugar en distintos dispositivos
Para llevar mi progreso a todos mis dispositivos y jugar en cualquier parte.

Pruebas de aceptación

- El juego podrá jugarse en las principales plataformas como móvil o tablet, ordenador, consola y también en la web.
- Todas las funcionalidades del juego estarán disponibles y funcionales en cada plataforma compatible.
- Los datos de una partida guardada podrán volver a abrirse en cualquier plataforma.

RNF5: Adaptabilidad a la pantalla

Como jugador
Quiero que el juego se adapte a como tengo puesta la pantalla de forma cómoda y sin errores
Para jugar cómodamente desde cualquier posición en cualquier postura

Pruebas de aceptación

- La interfaz del juego se ajustará dinámicamente para adaptarse a diferentes orientaciones de pantalla (horizontal y vertical).
- Los controles táctiles se escalarán adecuadamente para ser utilizables en pantallas de diferentes tamaños.
- Los elementos de la interfaz no se solaparán en ningún tamaño de pantalla llegando a eliminar alguno si fuese necesario.

RNF6: Referencias a la cultura popular

Como jugador
Quiero sorprenderme con referencias a cosas que conozco
Para divertirme espontáneamente en ciertos momentos del juego, haciendo que me salga una sonrisa o carcajada.

Pruebas de aceptación

- Durante el avance del juego podrán aparecer algunas referencias como algún nombre o algún apartado estético.
- Las referencias de la cultura popular estarán integradas de manera orgánica en el juego.
- Las referencias serán un simple guiño y no deberán afectar a ninguna funcionalidad del juego.

RNF7: Estética visual atractiva

Como jugador
Quiero un entorno cómodo y que me incite a jugar
Para no sentirme abrumado con demasiada información o con animaciones o estética cutre que me saque del juego.

Pruebas de aceptación

- Las animaciones serán suaves y bien ejecutadas, sin artefactos visuales o problemas de sincronización.
- Todo lo que se vea en pantalla debe transmitir información al jugador sobre su progreso en el juego.

RNF8: No necesita login con cuenta

Como jugador
Quiero no tener que iniciar sesión
Para poder jugar sin necesidad de preocuparme tener una cuenta.

Pruebas de aceptación

- No se deberá guardar ninguna información no necesaria del usuario más allá de su progreso.
- Si se cambia de dispositivo se le dará un código al usuario que contenga todo su progreso y que pueda meter en la aplicación o web para seguir con su partida.

RNF9: Existencia de *Easter eggs*

Como jugador
Quiero que haya secretos en el juego
Para mejorar la experiencia y estar atento a todo lo que ocurra.

Pruebas de aceptación

- Los *Easter eggs* no deben estar a la vista y se deberán hacer acciones específicas para que aparezcan.
- Los *Easter Eggs* serán variados y sorprendentes para mantener el interés de los jugadores.
- Los *Easter Eggs* serán estéticos y no afectarán en nada a la jugabilidad.

RNF10: Posibilidad de juego sin conexión

Como jugador
Quiero poder jugar sin conexión a internet
Para poder usar el juego en cualquier situación.

Pruebas de aceptación

- El juego no necesitará internet para poder jugarse.
- Todas las acciones del juego podrán hacerse sin conectarse a la red.

RNF11: Interfaz de usuario intuitiva

Como jugador
Quiero poder entender fácilmente todos los elementos de la interfaz
Para poder interactuar con el juego de manera sencilla.

Pruebas de aceptación

- Los botones y controles del juego deben estar claramente etiquetados en el menú de ajustes y ser fácilmente reconocibles por el jugador.
- La navegación entre las diferentes secciones del juego (tienda, menú principal, pantalla de juego, etc.) debe ser intuitiva y fluida.

6. Herramientas software usadas durante la realización del proyecto

Hasta la fecha, las herramientas software usadas en la realización del proyecto son las siguientes:

- **Microsoft Word (Office 365)**: elaboración de documentos compartidos en línea para trabajo cooperativo.
- **WhatsApp**: comunicación asíncrona.
- **Discord**: comunicación síncrona y compartición de documentos.
- **Google Drive**: compartición de documentos.
- **Photopea**: edición de imágenes para el logo del grupo.
- **GIMP**: edición de imágenes para el logo del proyecto.
- **tensor.art**: elaboración de logo con estilo de arte tipo *pixel-art*.
- **8bit Painter**: elaboración de logo con estilo de arte tipo *pixel-art*.
- **remove.bg**: eliminación del fondo de pantalla de imágenes.
- **Copilot (anteriormente Bing AI)**: creación de imágenes, elaboración de texto y búsquedas en internet.
- **ChatGPT**: resumen y elaboración de textos.
- **GitHub**: repositorio y control de versiones.
- **GitHub Desktop**: aplicación gratuita de código abierto que ayuda a trabajar con código hospedado en GitHub.
- **Trello**: organizador de tareas.
- **Unity**: motor de videojuegos multiplataforma usado para crear el videojuego *Feed The Leader*.
- **Microsoft Visual Studio Community**: entorno de desarrollo integrado usado junto con Unity.
- **C#**: lenguaje de programación multiparadigma usado en el desarrollo del videojuego en Unity.
- **Visual Paradigm**: software de diagramas y solución de gráficos empleado en el apartado de *Requisitos*.