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

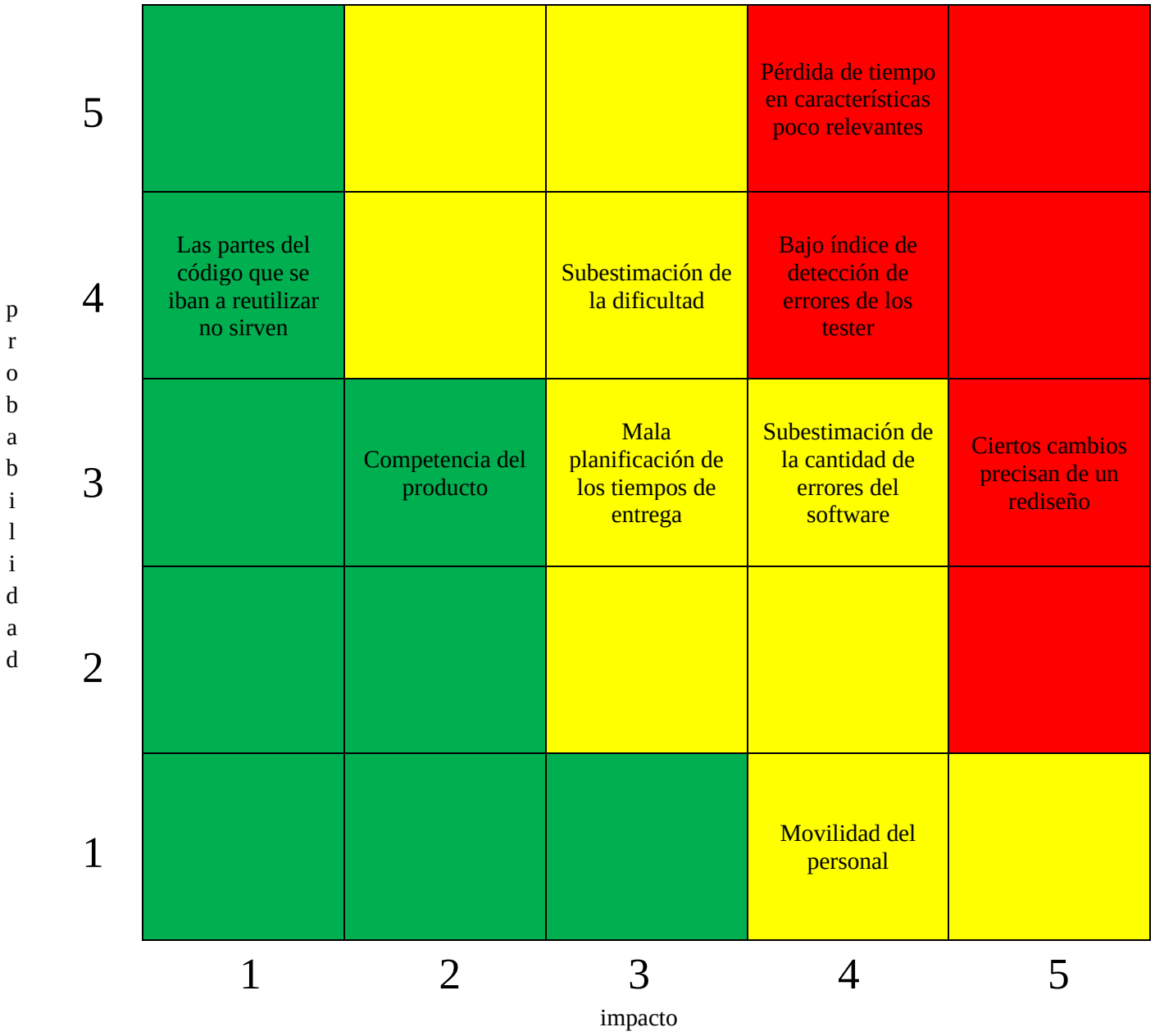
- Tipo: tecnológico.
- Descripción: los desarrolladores crean un código con demasiados errores que los tester no son capaces de encontrar.
- Probabilidad: alta.
- Efecto del riesgo: serio.
- Estrategia para mitigarlo: adoptar una estrategia de detección de errores efectiva por parte de los tester, realizando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

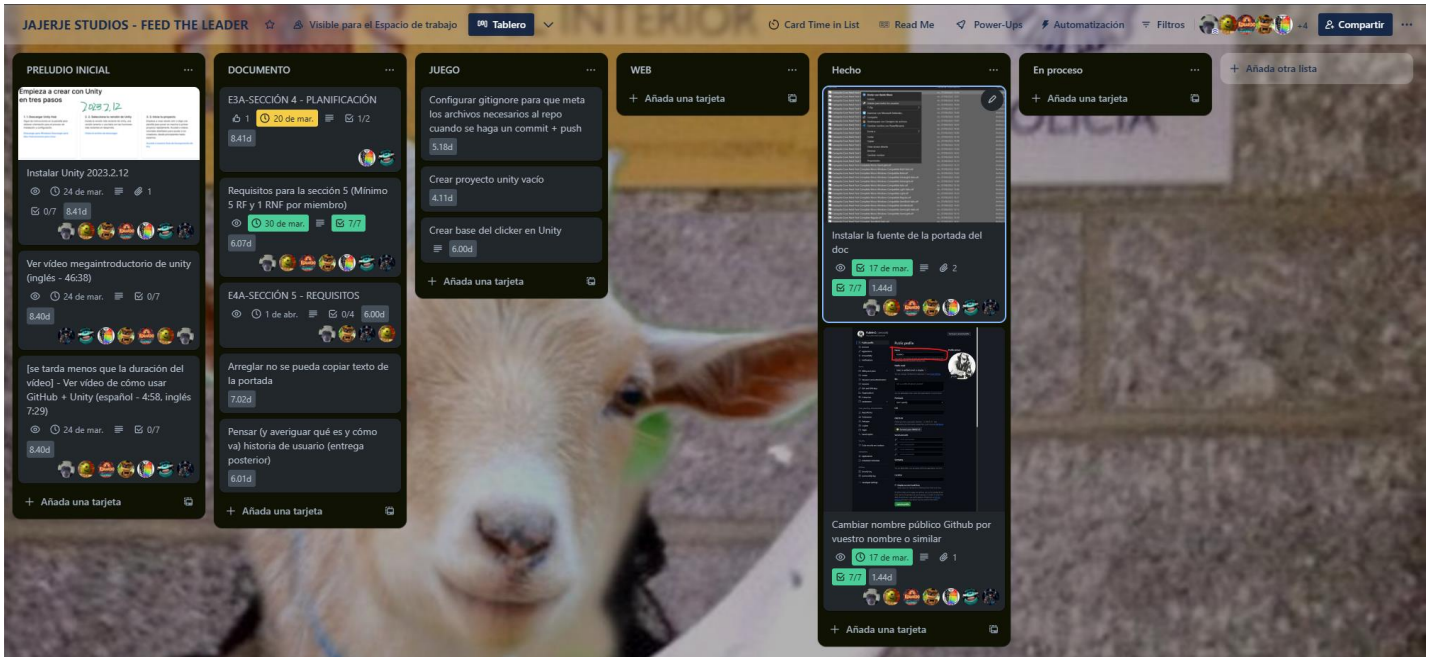
Se ha elegido la metodología ***Scrum*** porque es un modelo que se adecúa perfectamente a nuestras necesidades en el proyecto. Esta **metodología** es **iterativa** e **incremental**. Que sea iterativa quiere decir que el trabajo se divide en iteraciones, llamadas *sprints* en el caso de *Scrum*, que 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.

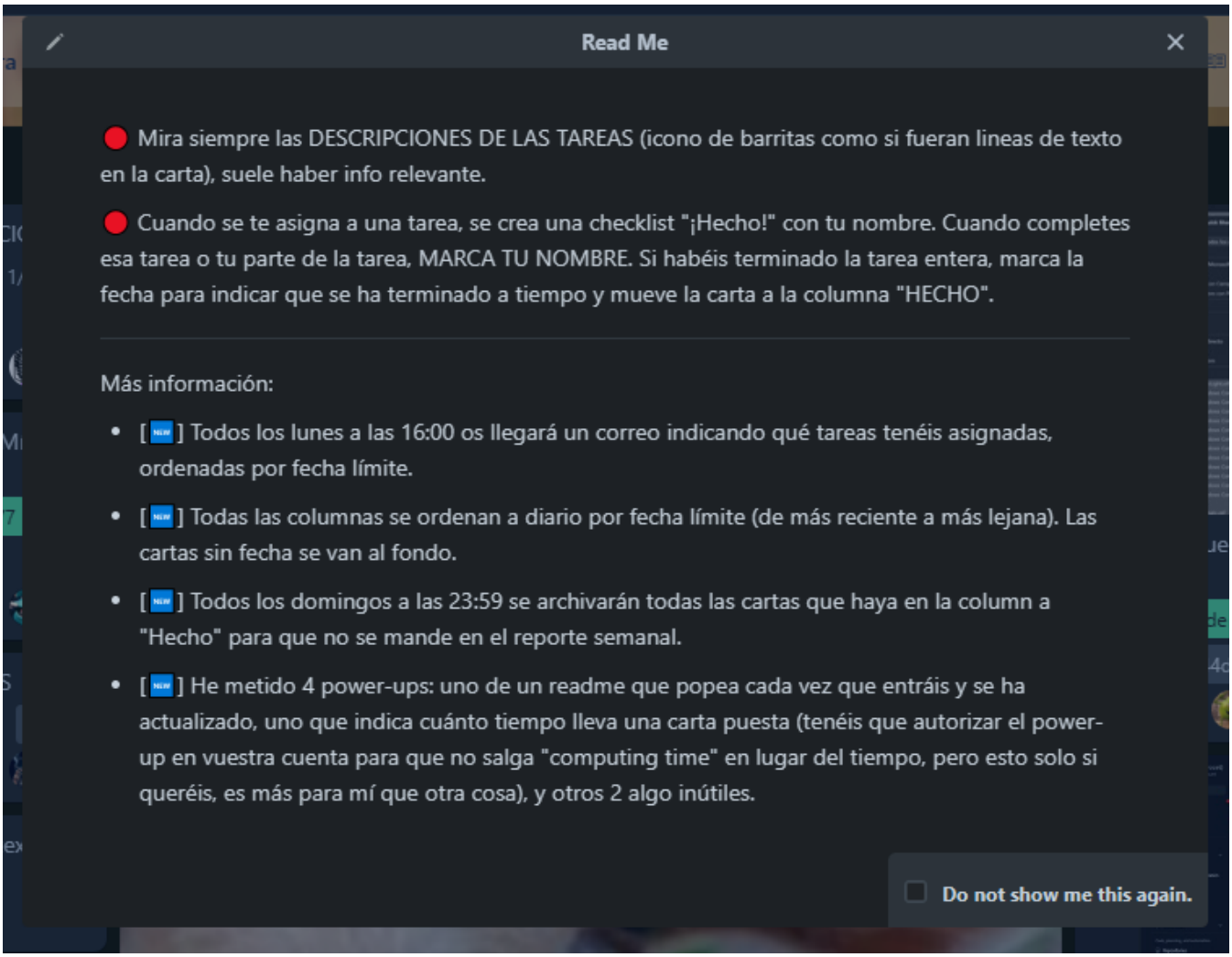
Con respecto a la columna ***Juego***, como su nombre indica, se encuentran las actividades con la realización de nuestro juego. Actualmente, vemos tareas básicas como son la creación de un proyecto vacío en *Unity*, la base del *clicker* y la configuración de *gitignore*.

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

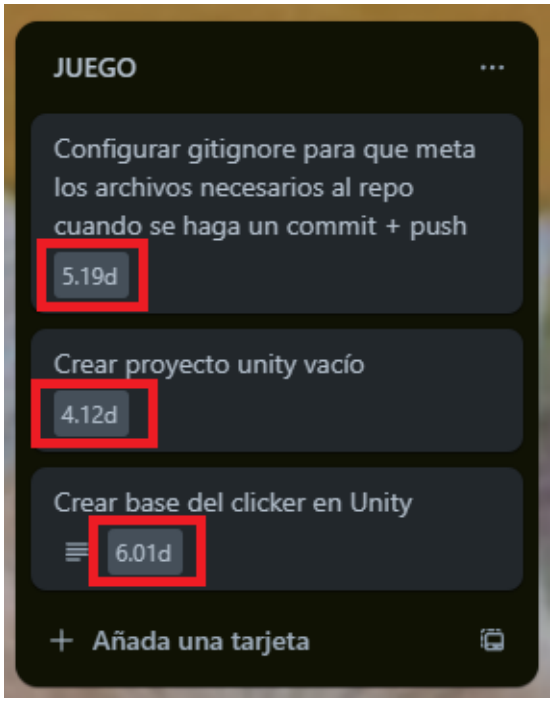
Se ha decidido añadir los siguientes *power-ups* para conseguir una organización todavía más efectiva:

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



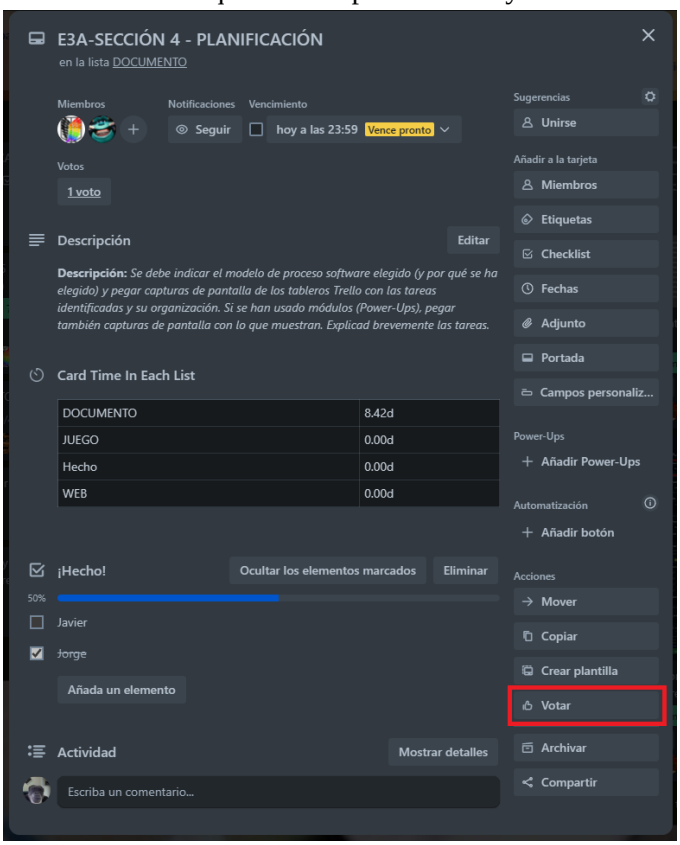
2 Pop-up de Read Me

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



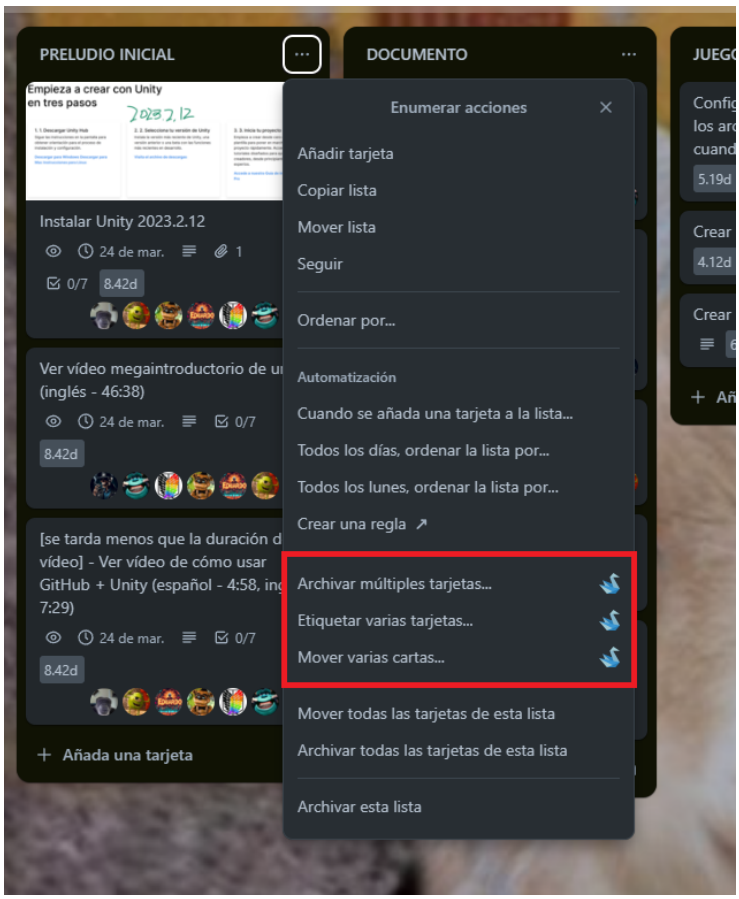
3 Muestra del power-up Card time

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.

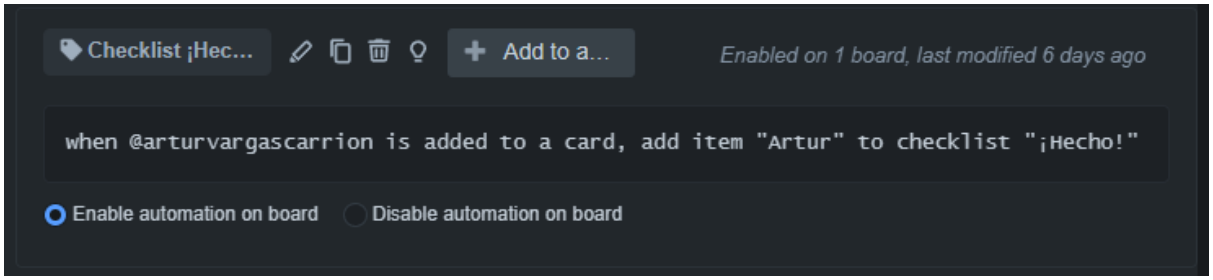


5 Muestra del power-up "Manny"

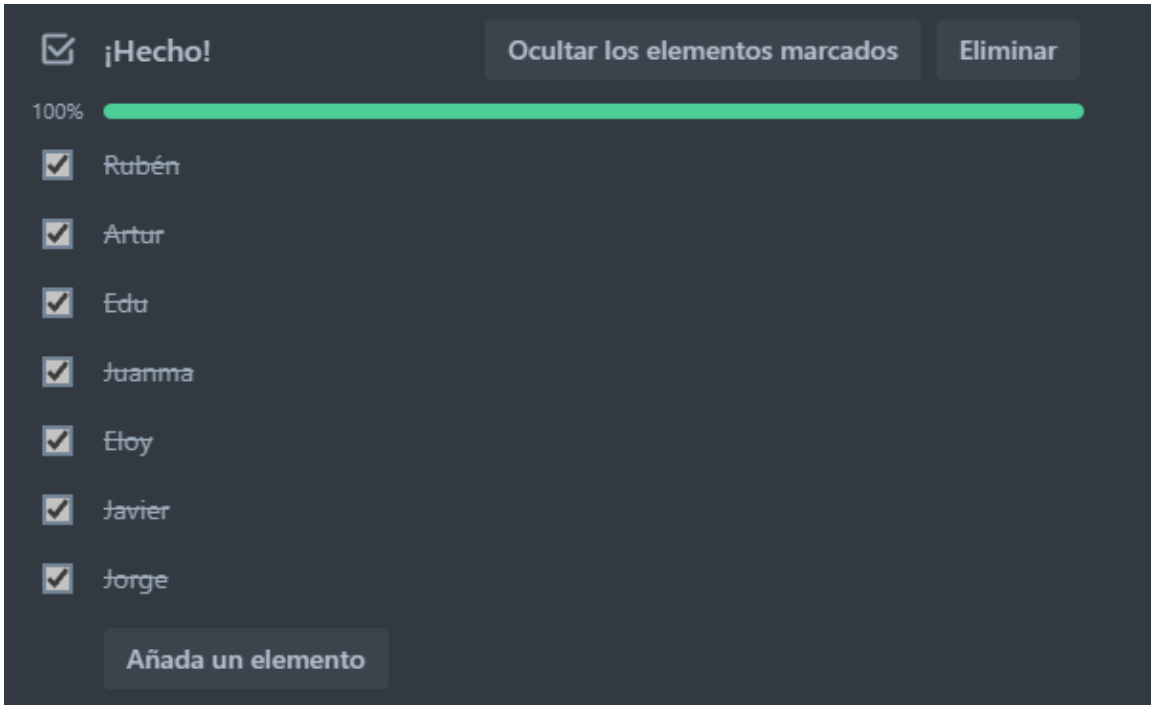
4.4. Automatizaciones

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.

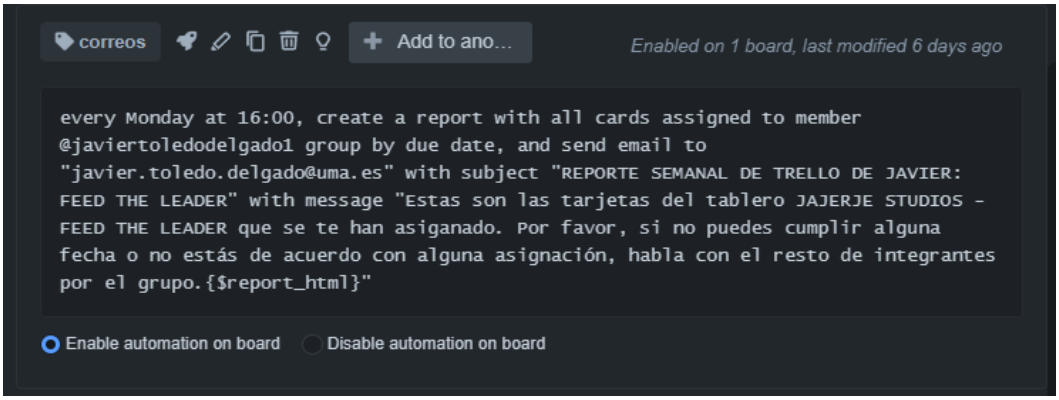


6 Automatización para crear el Checklist ¡Hecho!



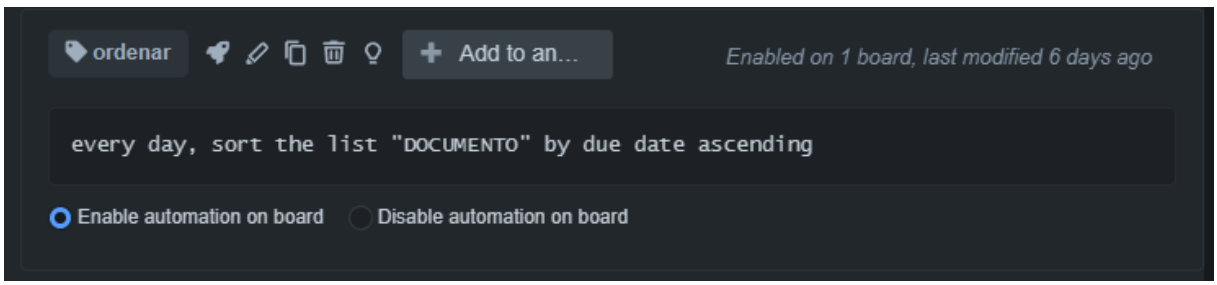
7 Checklist ya creado

- **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.



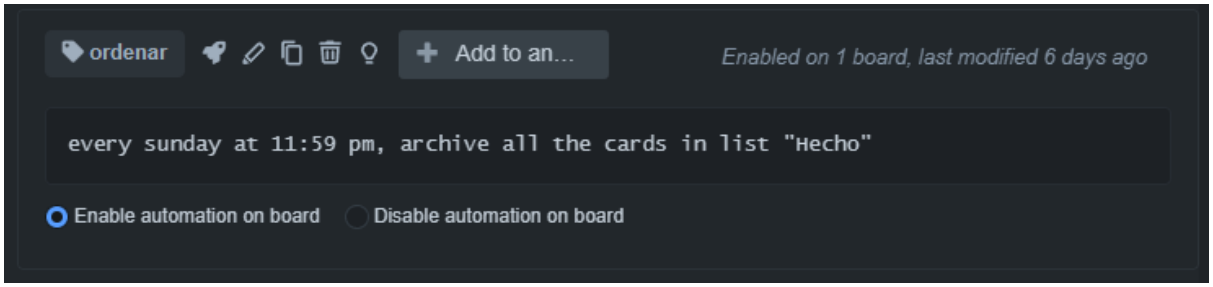
8 Automatización para enviar correos

- **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. 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.
- **Trello**: organizador de tareas.