SERIOUS GAMES POR UN DESARROLLO SOSTENIBLE

Desarrollo colaborativo de un videojuego web educativo usando metodologías ágiles

Objetivo de la práctica

Esta práctica consiste en el desarrollo colaborativo de un videojuego y una página web, a través de metodologías ágiles, utilizando herramientas de control de versiones e IA integrada en el entorno de desarrollo para agilizar la generación de código.
El proyecto servirá como iniciación al desarrollo colaborativo de una pieza de software, emulando en clase un entorno de trabajo realista.
Utilizaréis **Git como herramienta de control de versiones**, y aplicaréis el método **KANBAN**.

Parte 1. Explicación del proyecto

Organización del trabajo

- Grupos de **2 ó 3 personas**.
- Cada grupo debe acordar el diseño y reparto de tareas.
- Desarrollaréis un videojuego web incrustado en una página.

Requisitos del proyecto

Diseño de juego

- Menú principal desde el que se accede al juego.
- Al principio, el jugador dispone de **4 vidas**:
- El juego principal encadena **microjuegos**.

> Videojuego de muy corta duración, con una única mecánica y objetivo que debe completarse en un tiempo limitado.

- Cada fallo en un microjuego resta una vida.

Diseño de juego

- Debe contener al menos 3 microjuegos por integrante:
  - Un total de 6 en los proyectos desarrollados por parejas y 9 para grupos de 3.
- Al perder todas las vidas, el juego termina.
- La puntuación aumenta con cada microjuego superado.

Microjuegos

Cada microjuego debe cumplir los siguientes criterios:

- Un objetivo único y claramente definido.
- Un tiempo límite para completarlo.
- Interacción directa mediante ratón y/o teclado.
- Cada microjuego debe poder añadir variaciones para diferenciar **tres niveles de dificultad**, aumentando al superar los 10 y 20 puntos.

Microjuegos

- Un ejemplo de microjuego:

> Título: "¡No malgastes agua!"
> La interfaz indica: "¡Recoge!"
> Un vaso sustituye al cursor, el jugador debe moverlo para recoger partículas que caen de arriba a abajo de la pantalla, en forma de gotas de agua durante un tiempo limitado (10 segundos). Si el tiempo termina y el vaso no se ha llenado, el jugador pierde.

- Al fallar el microjuego, el jugador pierde una vida. Si todavía conserva alguna vida, continúa al siguiente microjuego.
- Al completar exitosamente, aumenta el contador.

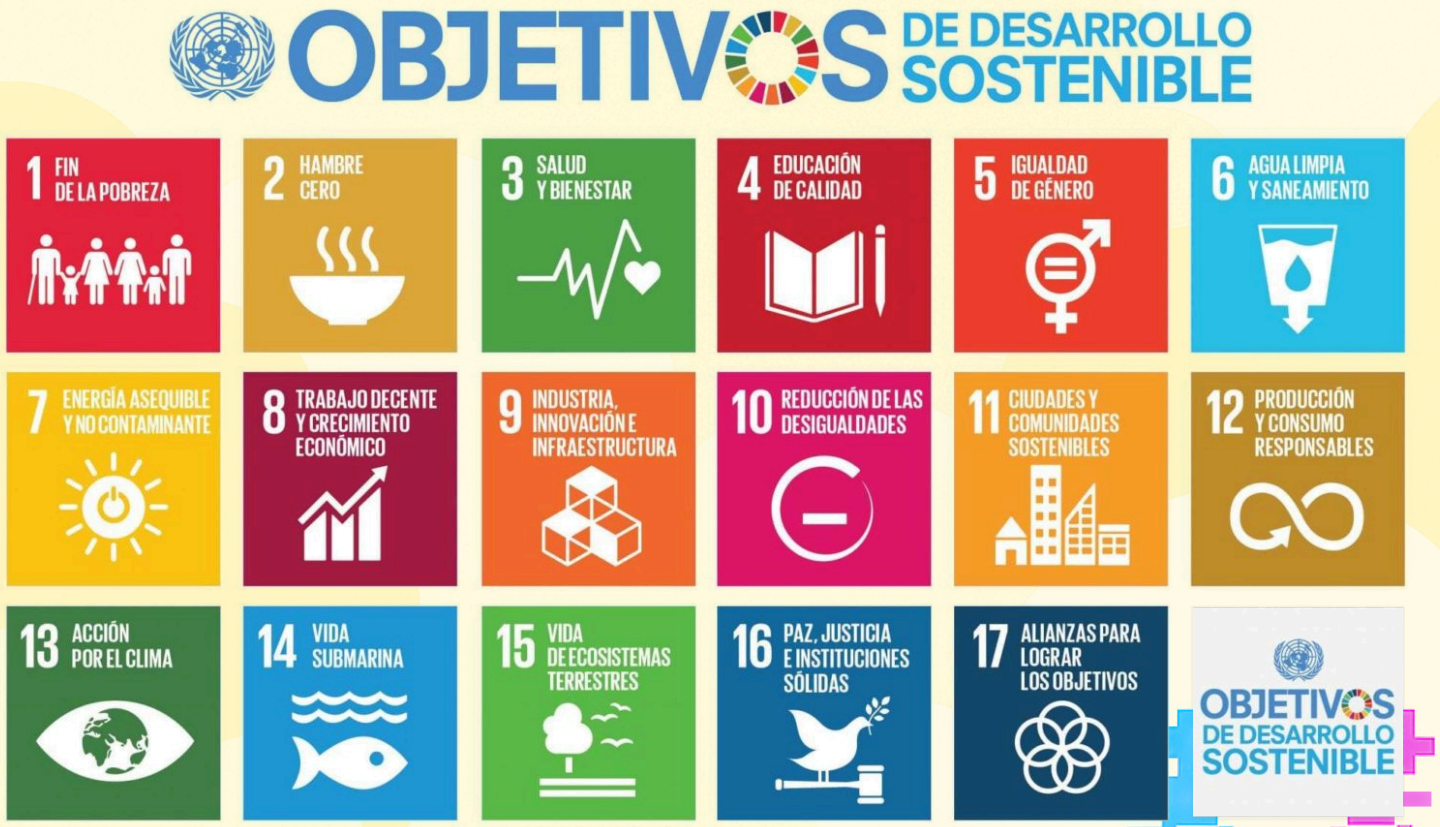
Microjuegos

- Tras superar 10, al enfrentar un mismo microjuego, el reto debe ser más difícil.
- Al superar los 20 puntos, la dificultad debe volver a aumentar.

> Para aumentar la dificultad, por ejemplo, el vaso puede ser sensiblemente más profundo o estrecho, las gotas de agua más escasas o rápidas y el tiempo para recoger agua más corto.

Temática del juego

- Cada microjuego debe estar relacionado con uno o varios **Objetivos de Desarrollo Sostenible**.



Ejemplo de proyecto

Repositorio:

github.com/javalbo/ProyectoIntermodularDAM

GitHub Pages:

javalbo.github.io/ProyectoIntermodularDAM

Parte 2. Preparación del entorno de trabajo

1. Configuración inicial

Cada alumno debe asegurarse de tener:

- Git instalado.
- Una cuenta activa en GitHub.
- El IDE instalado.

2. Creación del repositorio

- Cada grupo debe crear un repositorio nuevo usando GitHub.
- Cada uno, clonad el repositorio en vuestro equipo, utilizando el IDE.
- Abre el proyecto en el IDE.

3. Configuración

- Instala extensión **Live Server** en el IDE para visualizar los cambios en tiempo real.
- Configura **Git** desde el terminal:

```
git config --global user.name "[username]"
git config --global user.email "[your.email@example.com]"
```

4. Primer contacto con el flujo de trabajo

1. Crea una ***branch*** (rama) de nombre, por ejemplo, "prueba-[tus iniciales]".
2. Crea un archivo de texto de prueba. Escribe en él, por ejemplo, tu nombre.
3. Para publicar los cambios:

- Usa el apartado *Source Control* del IDE.
- Selecciona los cambios (asegúrate de que contiene el archivo de texto).
- Escribe un mensaje y descripción claros.
- Haz ***commit*** para registrar los cambios.
- Haz ***push*** para publicar los cambios.

Primer contacto con el flujo de trabajo

4. Para mezclar cambios definitivos en la rama principal:

- Debes asegurarte de que has implementado todos los cambios planeados y funcionan correctamente.
- Crea una ***pull request*** de tu rama a main. Asegúrate de que añades una descripción de la funcionalidad añadida.
- **Otro integrante** del grupo deberá confirmar que **no haya conflictos** y el programa funcione correctamente.
- Una vez validado, deberá proceder al ***merge***.

Ciclo del flujo de trabajo

Para cada tarea o funcionalidad desarrollada:

1. Crear una ***branch*** al comenzar una tarea o *funcionalidad*.
2. Hacer los cambios necesarios. Publícalos cada vez que tengas un avance significativo (***commit***).
3. Cuando la funcionalidad esté completa, crea una ***pull request***.
4. Un compañero/a deberá validar los cambios y proceder a mezclarlos en la rama principal (***merge***).

Configuración del Personal Access Token

Es posible que debáis configurar un ***Personal Access Token***.

- Generar un *Personal Access Token* en GitHub (usando la plantilla sugerida).
- Copiaremos el token ("ghp_...") para pegarlo cuando el IDE se solicite.

Parte 3. Gestión del proyecto

Planificación

- Crear un *Trello* con 4 columnas: *Backlog*, *In progress*, *Validate*, *Done*
- Realizar un documento de diseño o *ten pages* que contenga la documentación sobre el videojuego a desarrollar.
- Crear las tareas esperables a partir del documento de diseño en el backlog.
- Diario de desarrollo: Incluid información sobre el trello, reparto de tareas, progreso y dificultades en el desarrollo etc.

Control de versiones

- Al comenzar una tarea se debe:

1. **Asignarla en Trello** al integrante del grupo correspondiente y pasarla de la columna *backlog* a *in progress*.
2. **Crear una rama** con el esquema: "features/[nombre-de-la-tarea]"
3. Realizar los cambios necesarios.
4. **Hacer commit** cada vez que se añadan *cambios significativos* en la funcionalidad.

Control de versiones

5. Al terminar la tarea, se debe mover a tarea a la columna *validate*.
6. Otro integrante debe validarla, comprobando que no haya problemas de funcionalidad haciendo merge de *main* en la rama de desarrollo.
7. Una vez validada, se debe hacer merge de la rama de desarrollo a *main* y se mueve la tarea a *done*.

Ejemplo de planificación

- Una **tarea** podría ser: *Implementar microjuego "¡No malgastes agua!"*
- La **rama** en la que se desarrollará deberia llamarse "feartures/microjuego-no-malgastes-agua"
- Un **commit** de dicha rama podria incluir el mensaje "se añade dificultad media" y describirse como "ahora caen menos gotas y más rápido a partir de los 10 puntos"

Documentación

- Todo el proceso debe documentarse e incluirse en el repositorio y accederse en **diferentes apartados de la página web**.
- Debe incluirse:
  - Definición del proyecto en un documento ***ten pages*** que incluya una explicación del juego y de la mecánica de cada microjuegos
  - Una memoria de proyecto que incluya una explicación de la organización del trabajo, quién ha hecho cada tarea, quién la ha validado, capturas del tablero de Trello, etc.
  - Incluir link a Trello.

4. Entrega

- Activar **GitHub Pages**:
  - Rama: `main`
  - Carpeta: raíz del proyecto.
- Comprobar que la web es accesible desde la siguiente URL:https://USUARIO.github.io/NOMBRE_DEL_REPO/

Deberéis entregar los links para acceder al producto final del proyecto:

- Página web y juego en github pages
- Repositorio de código público

5. Exposición

Para poder evaluar esta actividad, deberéis realizar una exposición final de máximo 10 minutos, en la que explicaréis el desarrollo y producto final del proyecto.

- Debéis realizar una presentación en clase de formato libre.
- Máximo 10 min.
- Deberéis evaluar a vuestros compañeros durante las presentaciones utilizando una rúbrica que encontraréis en *Aules*.

6. Evaluación y calificación

La nota de esta actividad se calculará a partir de los siguientes apartados:

- Un 80% provendrá de la evaluación docente a partir de la exposición realizada.
- Un 20% provendrá de la evaluación de vuestros compañeros:
  - Para optar a este 20% deberéis evaluar al resto de equipos a través de Aules.

> Ambos apartados de la nota se calificarán a partir de la misma rúbrica, que encontraréis en Aules y en el repositorio de ejemplo.

7. Ampliación (opcional y no evaluable)

- Investigar el uso de un servidor para permitir la competición entre varios jugadores:
- Ranking de puntuaciones.
- Resultados compartidos.
- Multijugador básico.
- Crear elementos gráficos y añadirlos al juego.