Práctica: Desarrollo colaborativo de un videojuego web usando metodologías ágiles

Objetivo de la práctica

Iniciar al alumnado en el desarrollo colaborativo de un proyecto software realista utilizando **GitHub como control de versiones**, **VS Code como entorno de desarrollo** y **Antigravity como asistente de generación de código**, aplicando una organización basada en el método **KANBAN**.

Parte 1. Preparación del entorno de trabajo

1. Configuración inicial

Cada alumno debe asegurarse de tener:

- Git instalado.
- Una cuenta activa en GitHub.
- Antigravity instalado.

2. Creación del repositorio

- Crear un repositorio nuevo en GitHub.
- Clonar el repositorio utilizando Antigravity.
- Abrir el proyecto usando el IDE de Antigravity.

3. Configuración:

- Instalar extensión **Live Server** en el IDE para visualizar los cambios en tiempo real.
- **Git** desde la consola de VS Code:

```
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 un archivo de texto de prueba. Escribe en él, por ejemplo, tu nombre.
2. Crea una rama de prueba.
3. Para hacer *commit*:

- Usa el apartado *Source Control* del IDE.
- Selecciona los cambios (asegúrate de que contiene el archivo de texto).
- Escribe un mensaje de commit claro y descriptivo.
- Ejecuta *commit* para registrar los cambios.

4. Una vez se haya procesado el commit, haz *push* para que se suba al repositorio.

Configuración del 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 2. Inicio del proyecto

Organización del trabajo

- Grupos de **4 personas**.
- Cada grupo debe acordar:
- Diseño y reparto de tareas.

Requisitos del proyecto

Diseño de juego

- Menú principal desde el que se accede al juego.
- Al principio, el jugador dispone de **4 vidas**:
- Un juego principal que 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.
- Debe contener 12 microjuegos diferentes.
- Al perder todas las vidas, el juego termina.
- Cada microjuego superado añade una unidad al contador de puntuación.

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

Microjuegos

- Si el contador supera los 10 puntos, al enfrentar el 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 debe 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**.

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 los microjuegos 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:
  · **Asignarla en Trello** al integrante del grupo correspondiente y pasarla de la columna *backlog* a *in progress*.
  · **Crear una rama** con el esquema: "features/[nombre-de-la-tarea]"
  · Realizar los cambios necesarios.
  · **Hacer commit** cada vez que se añadan *cambios significativos* en la funcionalidad.
  · Al terminar la tarea, se debe mover a tarea a la columna *validate*.
  · Otro integrante debe validarla, comprobando que no haya problemas de funcionalidad tras hacer merge de *main* en la rama de desarrollo.
  · Una vez validada, se mergea 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 titularse "se añade dificultad media" y describirse como "ahora las gotas caen menos y más rápido a partir de los 10 puntos"

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/

Ampliación (opcional)

- 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.