



XTART

Fecha:

03/12/2024

Memoria

PROYECTO final

CRONOS XTART

Nombre del proyecto:
CRONOS

Organizado por:
Omar, Renzo, Diego, Alvaro

ASIGNATURAS

LENGUAJES DE MARCA

PÁG 3-13

PROGRAMACIÓN

PÁG 14- 28

BASE DE DATOS

PÁG 29-38

SISTEMAS INFORMÁTICOS

PÁG 39-44

ENTORNOS DE DESARROLLOS

PÁG 45-59

LENGUAJES DE MARCAS

Índice:

1. Introducción

2. Estructura general del sitio web

3. Maquetación y diseño en CSS

 3.1. Uso de Flexbox

 3.2. Uso de Grid Layout

 3.3. Estilos visuales aplicados

 3.4. Gestión de fondos e imágenes

4. Recursos multimedia

 4.1. Vídeos

 4.2. Imágenes

 4.3. Audio

5. Componentes desarrollados

 5.1. Botones interactivos

 5.2. Cajas de contenido (Boxes)

 5.3. Sistema de navegación narrativa

6. Responsividad

7. Validación y buenas prácticas

8. Conclusión del capítulo

1. Introducción

En este capítulo explico el trabajo realizado en la parte de Lenguajes de Marcas, donde he creado las diferentes páginas web que forman el proyecto CRONNOS-1. Aquí es donde se construye toda la parte visual: lo que el usuario ve, cómo se organiza la información y cómo se navega entre las distintas escenas de la historia.

Para ello he utilizado HTML5 y CSS, que son las bases para construir una web. A lo largo del proyecto he creado varias páginas que representan momentos de la aventura, como la introducción, los actos principales, las rutas alternativas o las pantallas de muerte. Cada página está hecha por separado, lo que facilita trabajar en ellas y mantener el orden.

Los estilos se controlan desde un único archivo CSS, donde he definido colores, tamaños, transparencias, fondos, maquetación y efectos. Esto me ha permitido que todas las páginas sigan una estética parecida y se sientan como parte del mismo proyecto.

En resumen, en este apartado se recoge todo el trabajo visual necesario para que la historia pueda verse y utilizarse de forma cómoda, clara y atractiva para el usuario.

2. Estructura general del sitio web

El sitio web de CRONNOS-1 está organizado en varias páginas HTML que se conectan entre sí según las decisiones que toma el usuario. Cada escena de la historia tiene su propio archivo, lo que hace que la estructura sea fácil de mantener y ampliar si fuese necesario. Todas las páginas comparten la misma hoja de estilos (stylefinal.css), donde se controla la apariencia general: colores, cajas, botones, fondos y distribución del contenido. Esto ayuda a que toda la web tenga un estilo coherente.

El proyecto se divide en varios tipos de pantallas:

- Pantallas iniciales, donde se presenta la historia y se pide la información del viajero.
- Actos y escenas principales, como el viaje temporal, la revisión del sistema o la elección del camino.
- Rutas alternativas, como el bar, la tienda de empeños o el laboratorio.
- Pruebas y minijuegos, donde el usuario debe responder preguntas para avanzar.
- Pantallas de muerte, que aparecen cuando el jugador falla una decisión o una prueba.

Cada una tiene su propio diseño, pero siguen la misma línea de maquetación.

El uso de videos, imágenes y sonidos ayuda a darle ambiente a cada escena, logrando que la navegación sea más entretenida y fácil de entender.

3. Maquetación y diseño en CSS

El diseño del proyecto se ha hecho con CSS, usando principalmente Flexbox y Grid para colocar los elementos de forma ordenada. Todo el estilo se guarda en un único archivo, lo que facilita modificar cosas rápidamente sin tener que tocar cada página una por una.

3.1. Uso de Flexbox

He usado Flexbox para centrar el contenido en la mayoría de las páginas, como la introducción, la revisión o el viaje temporal. Con Flexbox es más fácil alinear cajas, botones o textos sin depender de medidas fijas. También sirve para colocar elementos uno al lado del otro o distribuirlos con espacios iguales.

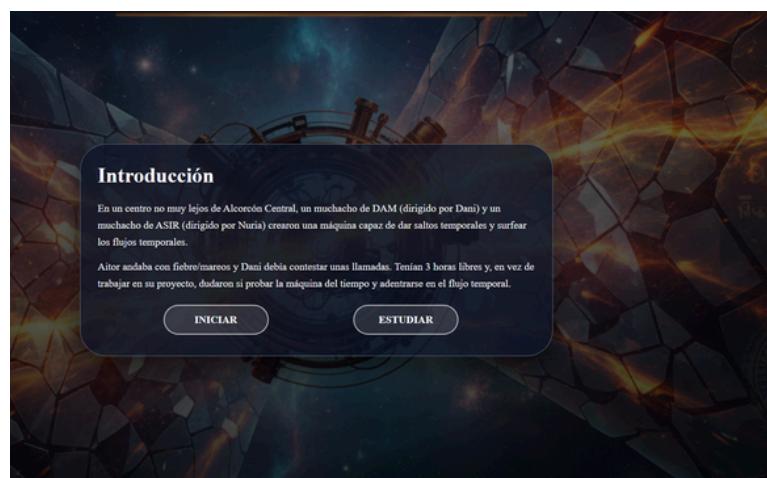
```
<body>
  <section class="introduccion">
    <div class="introduccion__contenido">
      <h2 class="introduccion__titulo">Introducción</h2>

      <p class="introduccion__texto">
        En un centro no muy lejos de Alcorcón Central, un muchacho de DAM (dirigido por Dani) y un muchacho de ASIR (dirigido por Nuria) crearon una máquina capaz de dar saltos temporales y surfear los flujos temporales.
      </p>

      <p class="introduccion__texto">
        Aitor andaba con fiebre/mareos y Dani debía contestar unas llamadas. Tenían 3 horas libres y, en vez de trabajar en su proyecto, dudaron si probar la máquina del tiempo y adentrarse en el flujo temporal.
      </p>

      <div class="introduccion__opciones">
        <div class="introduccion__opcion">
          <a href="act01.html" class="introduccion__boton">INICIAR</a>
        </div>

        <div class="introduccion__opcion">
          <a href="#" class="introduccion__boton">ESTUDIAR</a>
        </div>
      </div>
    </div>
  </section>
```



3.2. Uso de Grid Layout

Grid se ha utilizado cuando necesitaba una estructura más definida, por ejemplo en las pantallas de los quizzes, donde la pregunta y las respuestas se colocan en diferentes zonas de la pantalla. Con Grid puedo dividir la página en filas y columnas, y colocar cada elemento justo donde quiero.

```
<main class="R-quiz__wrapper">
  <header class="R-quiz__header">
    <div class="R-quiz__headerQuestion">
      <p>¿Hace cuántos años se extinguieron los dinosaurios?</p>
    </div>
  </header>

  <section class="R-quiz__options">
    <a href="tumba.html" class="R-quiz__option R-quiz__option--wrong">
      <span>Hace 93 millones de años</span>
    </a>

    <a href="CAPITULO_2B.html" class="R-quiz__option R-quiz__option--correct">
      <span>Hace 66 millones de años</span>
    </a>
  </section>
</main>
```



3.3. Estilos visuales aplicados

El estilo general del proyecto es oscuro para que los textos destaqueen sobre los vídeos de fondo. Se han utilizado cajas semitransparentes, sombras y bordes redondeados para darle un aspecto más moderno.

Los botones cambian al pasar el ratón por encima y se han reutilizado en varias páginas. También hay tarjetas y contenedores diferentes para escenas como el bar, las rutas o los quizzes.

3.4. Gestión de fondos e imágenes

En muchas escenas se utilizan vídeos como fondo ocupando toda la pantalla. Esto hace que el proyecto sea más visual y dé la sensación de estar dentro de la historia. También se han usado imágenes para iconos (como la botella o los puñetazos), para fondos de quizzes o para representar elementos históricos.

Las imágenes y vídeos se han colocado de forma que no molesten a la lectura, añadiendo capas oscuras y ajustando tamaños.



```
<video id="bg-video" autoplay muted loop playsinline>
  <source src="imagenes/bar.mp4" type="video/mp4">
</video>
<section class="bar">
```

4. Recursos multimedia

En este proyecto se han utilizado diferentes tipos de recursos multimedia para hacer que la historia de CRONNOS-1 sea más dinámica y entretenida. Estos elementos ayudan a que cada escena tenga su propio ambiente y a que el usuario se sienta más dentro de la aventura.

Los principales recursos que se han usado han sido vídeos, imágenes y audio, combinados con el código HTML y los estilos de la hoja stylefinal.css.

4.1. Vídeos

Los vídeos son uno de los elementos más importantes del proyecto.

Se utilizan como fondo en muchas páginas para dar la sensación de estar realmente en otro lugar o en otra época.

Por ejemplo:

- En la introducción aparece un vídeo que ambienta el inicio. htm introduccion
- En la escena del bar hay un vídeo del interior del bar. bar
- En las rutas históricas se muestran vídeos según la época. rutaB1
- En las pantallas de muerte aparece un vídeo de fondo más oscuro. viaje

Los vídeos se muestran con la etiqueta <video> y están configurados para reproducirse en bucle, ocupar toda la pantalla y no llevar sonido.

4.2. Imágenes

Además de los vídeos, también se han usado imágenes para completar el diseño de la web.

Estas imágenes aparecen en distintos sitios:

- Como iconos dentro de opciones (por ejemplo, la botella o los puñetazos en el bar). bar
- En los quizzes, donde se ven personajes históricos o imágenes relacionadas con la pregunta.
- Como fondos estáticos en algunas escenas, usando background-image desde el CSS.

Las imágenes ayudan a que las páginas sean más visuales y entiendas mejor lo que pasa en la historia.

4.3. Audio

El audio también forma parte de la experiencia.

Aunque no es obligatorio en todos los proyectos, en este se han añadido sonidos para dar más ambiente.

Por ejemplo:

- Música suave en la introducción. htm introduccion
- Sonidos del bar o escenas de tensión. bar
- Sonidos más serios en las pantallas de muerte. viaje
- Efectos en escenas como la revisión de la máquina o el viaje temporal. revision

El audio solo empieza cuando el usuario hace clic en la página, ya que los navegadores no permiten reproducir sonido automáticamente. Por eso se usa un pequeño script en JavaScript.

5. Componentes desarrollados

En el proyecto CRONNOS-1 he creado varios componentes que se repiten a lo largo de las páginas y que ayudan a que la web sea más fácil de usar y más coherente visualmente. Estos componentes están hechos con HTML y CSS, y algunos también utilizan pequeñas funciones de JavaScript para mejorar la experiencia del usuario.

Los principales componentes que se han desarrollado son: botones, cajas de contenido, opciones interactivas y el sistema de navegación narrativa.

5.1. Botones interactivos

Los botones se usan en casi todas las escenas para avanzar, elegir una opción o volver atrás.

Todos comparten el mismo estilo para que el usuario los reconozca fácilmente:

- Tienen un color base que encaja con el diseño oscuro del proyecto.
- Se iluminan o cambian ligeramente al pasar el ratón por encima (hover).
- Tienen bordes redondeados y un tamaño cómodo para hacer clic.

Se reutilizan en escenas como:

- Introducción
- Bar
- Tumba (pantalla de muerte)
- Rutas de la historia
- Minijuegos

Gracias a esto, el usuario entiende rápido cómo moverse por la web.

5.2. Cajas de contenido

Las cajas son los bloques donde aparece el texto principal de cada escena (narrativa, preguntas, advertencias, etc.).

Estas cajas comparten características:

- Fondo semitransparente para dejar ver el video detrás.
- Bordes redondeados y sombras para destacar.
- Espaciado interno para que el texto no quede pegado a los bordes.

Se usan en:

- Introducción
- Viaje temporal
- Revisión
- Bar
- Empeños
- Laboratorio
- Quizzes

Estas cajas permiten que el texto se lea bien aunque el fondo sea dinámico.

5.3. Opciones interactivas

En escenas como el bar, las rutas históricas o los minijuegos, el usuario debe elegir una opción para avanzar.

Para esto se han creado tarjetas interactivas que contienen:

Un ícono o imagen representativa (por ejemplo, botella o puñetazos).

Un pequeño texto explicativo.

Un enlace que lleva a la siguiente página de la historia.

Las tarjetas se agrandan ligeramente o cambian de sombra al pasar el ratón, para indicar que son clicables.

Este tipo de componente ayuda a que las decisiones sean más visuales e intuitivas.

5.4. Sistema de navegación narrativa

La historia se divide en varias páginas que se conectan mediante enlaces.

Este sistema funciona como un "árbol de decisiones":

- Si el usuario elige una opción → va a una escena concreta.
- Si falla un quiz → aparece una pantalla de muerte.
- Si responde bien → pasa a la siguiente parte del viaje.
- Algunas rutas vuelven a un camino principal y otras terminan.

Este sistema se ha construido únicamente con enlaces HTML (``), y permite crear rutas alternativas sin necesidad de programación avanzada.

6. Responsividad

La responsividad del proyecto consiste en que las páginas se adapten lo mejor posible a distintos tamaños de pantalla. Aunque el diseño está pensado principalmente para ordenador, se han aplicado algunas prácticas que ayudan a que el contenido no se descoloque cuando se ve en ventanas más pequeñas.

Para conseguirlo, he utilizado valores en porcentajes (%), unidades relativas como vh y vw, y elementos centrados mediante Flexbox. Esto hace que las cajas principales, los textos y los botones mantengan su posición incluso cuando el usuario cambia el tamaño del navegador.

También se han usado imágenes y vídeos con las propiedades object-fit: cover y background-size: cover, lo que permite que el contenido visual ocupe toda la pantalla sin deformarse. De esta forma, el vídeo de fondo y las imágenes se ajustan correctamente dependiendo del espacio disponible.

Aunque no es una responsividad completa como la de una página profesional, sí se han seguido varias buenas prácticas para evitar que la interfaz se vea desordenada o difícil de usar en distintos tamaños de pantalla.

7. Validación y buenas prácticas

Durante el desarrollo del proyecto he comprobado que el código HTML y los estilos CSS estuvieran bien organizados y sin errores básicos. Aunque no he usado herramientas profesionales de validación, sí he seguido una serie de buenas prácticas que hemos visto en clase para asegurar que la web funcione correctamente y sea fácil de mantener.

Entre las cosas que he revisado están:

- Que todas las etiquetas HTML estén bien cerradas.
- Mantener la estructura de cada página clara y ordenada (head, body, section, etc.).
- Usar clases reutilizables para no repetir estilos innecesarios en el CSS.
- Comprobar que los enlaces entre páginas funcionen sin errores.
- Revisar que los vídeos, imágenes y audios se carguen bien desde sus rutas.
- Probar los botones y las opciones interactivas para asegurar que respondan correctamente.

También he procurado que el nombre de las clases sea claro (por ejemplo, bar__contenido, panel__container, quiz__opcion) para entender fácilmente qué hace cada parte del código. Esto me ayudó a modificar cosas sin romper otras páginas.

En general, aunque el proyecto no es complejo a nivel técnico, aplicar estas buenas prácticas me ha permitido trabajar de forma más ordenada y evitar muchos errores mientras lo iba construyendo.

8. Conclusión del capítulo

En este capítulo he explicado cómo he creado toda la parte visual del proyecto CRONNOS-1 utilizando HTML y CSS. Gracias a las distintas páginas, estilos y componentes, he podido convertir la historia en una experiencia interactiva donde el usuario puede avanzar, tomar decisiones y ver diferentes escenas según sus elecciones.

Durante el desarrollo he usado técnicas como Flexbox, Grid Layout, fondos con vídeo, imágenes, iconos y efectos de audio para darle más ambiente a cada pantalla. También he tratado de mantener un código ordenado y reutilizable, siguiendo buenas prácticas para evitar errores y facilitar los cambios.

Aunque el proyecto aún puede ampliarse con nuevas rutas o finales, la parte de Lenguajes de Marcas queda completa y funcional, y me ha servido para comprender mejor cómo se estructura y diseña un sitio web real.

PROGRAMACIÓN

Índice:

1. Introducción
2. Estructura general del programa
3. Elementos de programación utilizados
 - 3.1. Variables
 - 3.2. Condicionales
 - 3.3. Bucles
 - 3.4. Arrays
 - 3.5. Métodos
 - 3.6. Manejo de entradas
 - 3.7. Uso de Random
 - 3.8. Organización del código
4. Narrativa interactiva del programa
5. Minijuegos desarrollados
 - 5.1. Adivina el número
 - 5.2. El Ahorcado
6. Validación y control de errores
7. Integración de narrativa y jugabilidad
8. Flujo general del programa
9. Conclusión

1. Introducción

En este capítulo explico la parte de Programación del proyecto CRONNOS-1. Todo el juego interactivo está hecho en Java y se ejecuta por consola. El objetivo era crear una aventura en la que el usuario pudiera tomar decisiones, jugar a minijuegos y llegar a distintos finales según lo que hiciera.

Para desarrollarlo he utilizado los contenidos básicos que hemos aprendido en clase:

- variables y constantes,
- condicionales if y switch,
- bucles (for, while, do-while),
- arrays y arrays bidimensionales,
- métodos,
- y lectura de datos con Scanner.

También he añadido algunos detalles extra, como usar códigos de color ANSI para que los textos de la consola sean más llamativos. Esto no es obligatorio, pero ayuda a que el juego sea más visual.

La historia está dividida en varias partes (prólogo, actos, minijuegos y finales), y cada una está en un método distinto para que el código esté más organizado. A lo largo del capítulo explico cómo está estructurado el programa y qué elementos de programación se han utilizado.

2. Estructura general del programa

El programa está organizado dentro de una única clase llamada HistoriaAventura. Desde aquí se controla toda la aventura, los minijuegos y los finales posibles del juego. El método principal es main, que es el que se ejecuta primero cuando arrancamos el programa. Dentro de main se van llamando, en orden, a los distintos métodos que representan cada parte de la historia.

La estructura general funciona como si fueran capítulos:

primero aparece la introducción, luego la decisión inicial, más tarde los actos donde se viaja en el tiempo, y finalmente los minijuegos y el laboratorio con preguntas. De esta forma, la historia avanza paso a paso de manera organizada.

Además de los métodos que representan escenas, el programa tiene otros métodos que sirven de apoyo, como los que imprimen texto lento, los que añaden pausas entre mensajes, o los que controlan la entrada de datos del usuario. Para manejar objetos que el jugador va consiguiendo, se utiliza un pequeño sistema de inventario basado en un array.

En resumen, la estructura del programa está dividida en bloques bien separados:

- Métodos narrativos, que cuentan la historia (como introduccion(), decisiones(), sigloXVIII(), etc.).
- Métodos de minijuegos, como adivinaNumero() y ahorcado().
- Métodos de utilidades, como leerNumero(), escribirLinea() o esperar().
- Métodos del inventario, como añadirAllInventario() y mostrarInventario().
- Gracias a esta organización, el código se entiende mejor, es más fácil de modificar y las distintas partes del juego quedan bien separadas unas de otras.

```
    import java.util.Random;
    import java.util.Scanner;

    public class HistoriaAventura {
```

```
        public static void main(String[] args) {
            banner();
            introduccion();
            decisiones();
            maquinaDelTiempo();
            actoNumero1();
            colores();
            sigloXVIII();
            laTienda();
            adivinaNumero();
            ahorcado();
            pregunta1();
        }
```

3. Elementos de programación utilizados

En este proyecto he utilizado prácticamente todos los elementos básicos de programación que hemos visto en clase. Gracias a ellos he podido crear una historia interactiva, controlar las decisiones del usuario, manejar un inventario y programar los minijuegos. A continuación explico cada grupo de elementos y cómo los he aplicado en mi código.

3.1 Variables y constantes

A lo largo del programa utilizo distintos tipos de variables:

- int para contadores, intentos, fallos y límites numéricos.
- String para decisiones, nombres, respuestas y textos de la historia.
- boolean para controlar estados como si el jugador ha acertado o si se debe reiniciar un minijuego.

También uso constantes (final String) para los colores ANSI que dan formato al texto de la consola. Esto hace que sea más fácil cambiar un color sin modificar todo el programa.

```
static final String RESET = "\u001B[0m"; 71 usages
static final String CYAN = "\u001B[36m"; 11 usages
static final String GREEN = "\u001B[32m"; 11 usages
static final String YELLOW= "\u001B[33m"; 14 usages
static final String BOLD = "\u001B[1m"; 5 usages
static final String RED = "\u001B[31m"; 25 usages
static final String BROWN = "\u001B[38;5;94m"; 4 usages
```

3.2 Condicionales (if, else, switch)

Las decisiones del jugador se controlan con condicionales. Algunos ejemplos:

Elegir entre iniciar la aventura o estudiar (if).

Elegir entrar al bar o a la tienda en el siglo XVIII.

Comprobar si el nombre introducido pertenece a un alumno válido.

Verificar si una respuesta del laboratorio es correcta o incorrecta.

En la parte del laboratorio utilice switch, que es útil cuando hay varias opciones como A, B, C o D.

```
while (!decisiones.equalsIgnoreCase(JUGAR) && !decisiones.equalsIgnoreCase(NO_JUGAR)) {
    escribirLinea( texto: "Deseas jugar? Pulsa 1 para jugar o 2 para rechazar la oferta.", pausa: 700);
    decisiones = sc.nextLine().trim();

    if (decisiones.equalsIgnoreCase(JUGAR)) {
        escribirLinea( texto: YELLOW + "ADELANTE." + RESET, pausa: 600);
        break;
    }

    } else if (decisiones.equalsIgnoreCase(NO_JUGAR)) {
        escribirLinea( texto: RED + "¿Estás seguro? Pulsa 1 para que te lo pregunte de nuevo o 2 para rendirte." + RESET, pausa: 650);
        reiniciar();
        decisiones = "";
    }
}
```

3.3 Bucles (for, while, do-while).

Los bucles permiten repetir acciones mientras se cumplen ciertas condiciones. Se utilizan en varios puntos:

while para pedir opciones hasta que el usuario introduzca un valor válido.

do-while en el minijuego de “Adivina el número”, para permitir reiniciar la partida.

for en el ahorcado, para recorrer todas las letras de la palabra.

for para recorrer el inventario o la lista de alumnos.

3.4 Arrays y arrays bidimensionales

En este proyecto utilice dos tipos de arrays:

Array unidimensional

```
String[] inventario = new String[10];
```

Sirve para guardar los objetos que el jugador va consiguiendo.

Uso la variable numObjetos para saber cuántos huecos están ocupados.

Array bidimensional

String[] alumnosXTART

Contiene dos filas: una con alumnos de DAM y otra con alumnos de ASIR.

Esto se usa para comprobar si el nombre que escribe el usuario existe en su curso.

```
String[][] alumnosXTART = {
    {"omar", "aitor", "dani", "laura", "ivan", "javier", "sergio", "diego", "juan", "alvaro"},  
    {"nuria", "helen", "carlos", "alejandro", "gonzalo", "noemi", "renzo", "clemente", "berta"}  
};
```

3.5 Métodos y modularización

El programa está dividido en muchos métodos, cada uno con una función concreta:

Métodos narrativos: introduccion(), decisiones(), sigloXVIII(), etc.

Métodos de minijuegos: adivinaNúmero(), ahorcado(), juego()

Métodos de utilidad: leerNúmero(), escribirLinea(), esperar().

Métodos del inventario: añadirAllInventario() y mostrarInventario().

Esto permite ordenar mejor el código y evitar que el método main sea demasiado largo.

```
public static void maquinaDelTiempo() { 1 usage & Omar1676
separador( titulo: "ACTO II · LA MÁQUINA DEL TIEMPO");

String maquina =
"-----\n" +
"/- ----- \n" +
"|- ⊖ MÁQUINA TEMPORAL \"CRONOS-1\" ⊖ |\n" +
"|- |-----| |\n" +
"|- ||-----|| |\n" +
"|- || ENERGÍA TEMPORAL: 99% || |\n" +
"|- || ESTABILIDAD: OK || |\n" +
"|- |-----| |\n" +
"|- [::::::| ACTIVANDO PORTAL ||::::::] |\n" +
"|- |-----| |\n" +
"----- / / / / \n" +
"----- + + + + + + \n" +
"----- \n"
```

3.6 Entrada de datos y control de errores

Para leer lo que escribe el usuario uso la clase Scanner.

El método más importante aquí es:

- Que el usuario no deje la entrada vacía.
 - Que solo escriba números válidos.
 - Que el número esté dentro del rango permitido.

Gracias a esto, el programa no se rompe aunque el usuario escriba algo incorrecto.

```
public static int leerNumero(int min, int max) { 3 usages & Omar1676
    while (true) {
        String entrada = sc.nextLine().trim();

        if (entrada.isEmpty()) {
            System.out.println(RED + ">>> No has escrito nada." + RESET);
            continue;
        }

        boolean esNumero = true;
        for (int i = 0; i < entrada.length(); i++) {
            char c = entrada.charAt(i);
            if (c < '0' || c > '9') {
                esNumero = false;
                break;
            }
        }

        if (esNumero) {
            int numero = Integer.parseInt(entrada);
            if (numero >= min && numero <= max) {
                return numero;
            } else {
                System.out.println(YELLOW + ">>> Por favor, introduce un número entre " + min + " y " + max + "." + RESET);
            }
        } else {
            System.out.println(RED + ">>> Eso no es un número válido. Solo dígitos." + RESET);
        }
    }
}
```

3.7 Uso de Random

En el minijuego “Adivina el número” utilizo la clase Random para generar un número aleatorio del 1 al 10:

```
int secreto = aleatorio.nextInt( bound: 10 ) + 1;
```

4. Narrativa interactiva del programa

La narrativa del proyecto está construida como una aventura interactiva donde el jugador avanza tomando decisiones que afectan directamente al desarrollo de la historia. La idea principal es que el usuario pueda sentirse dentro de un viaje temporal con distintos caminos posibles y varios finales. Para conseguirlo, el programa utiliza una estructura dividida en actos, donde cada uno representa una parte diferente del viaje.

La aventura empieza con una introducción que presenta a los personajes y el contexto inicial. Despues, el jugador debe tomar una decisión importante que determina si continúa con la historia o si finaliza el juego de manera temprana. A partir de ese punto, cada acto del programa presenta una escena nueva, con su propio ambiente, diálogos y opciones.

Para gestionar esta narrativa, cada escena está desarrollada dentro de métodos independientes, como `introduccion()`, `decisiones()`, `sigloXVIII()` o `laTienda()`. Esto hace que la historia sea más fácil de organizar y que el código sea más claro. Además, gracias a este sistema modular, se pueden añadir nuevas rutas o modificar escenas sin afectar al resto del programa.

Durante la historia, el jugador también participa en minijuegos, que forman parte de las pruebas necesarias para avanzar. Estas mecánicas ayudan a mantener la narrativa activa y aportan variedad al recorrido. También hay momentos donde la elección del usuario modifica el camino narrativo, como cuando decide entrar en el bar o en la tienda, o cuando escoge cómo enfrentarse en una pelea.

En conjunto, la narrativa del juego combina decisiones, escenas independientes y elementos interactivos para crear un recorrido que cambia según el jugador. Aunque el programa va guiando paso a paso, son las elecciones del usuario las que determinan qué final alcanza y cómo se desarrolla su viaje temporal.

5. Minijuegos desarrollados

Dentro de la aventura incluí dos minijuegos sencillos programados en Java. Estos minijuegos forman parte de las pruebas que el jugador debe superar para conseguir las piezas necesarias para reparar la máquina del tiempo. Cada uno está programado en un método independiente y utiliza estructuras básicas del temario: bucles, condicionales, arrays y lectura por teclado.

5.1. Adivina el número

Este minijuego consiste en que el programa genera un número aleatorio entre 1 y 10.

El usuario tiene tres intentos para acertarlo. Después de cada intento, el programa indica si el jugador se ha pasado o se ha quedado corto. Si acierta dentro del límite de intentos, obtiene un objeto para el inventario. Si falla los tres intentos, el juego ofrece la posibilidad de reiniciar.

Este minijuego me ha servido para practicar:

- La clase Random
- El uso de if y else
- Control de intentos con un bucle for
- Validación de entrada numérica

5.2. El Ahorcado

```
for (int i = 1; i <= intentos; i++) {
    escribirLinea("Indica el número que crees que es (1-10)", pausa: 500);
    int n1 = leerNumero(1, 10);
    contador++;
    if (n1 == secreto) {
        escribirLinea("¡ACERTASTEEEEE! Con " + contador + " intentos." + RESET, pausa: 650);
        escribirLinea("Te falta la última parte para conseguir tu objetivo sin que me pagues. ¿serás capaz?", pausa: 700);
        adquirirInventario("lleva oxígeno del anciano");
        acertado = true;
        break;
    } else if (n1 > secreto) {
        escribirLinea("No no, te has ido alto, jóven.", pausa: 550);
    } else {
        escribirLinea("No no, te has ido bajo, muchacho.", pausa: 550);
    }
}
```

Este minijuego es más completo. El programa tiene una palabra fija (“talisman”) y el jugador debe ir escribiendo letras.

El sistema comprueba si la letra ya se usó antes, si coincide con la palabra y va rellenando un array con los aciertos. Cada fallo suma uno, y si se alcanzan 6 fallos, el jugador pierde.

Con este ejercicio practiqué:

- Arrays de caracteres
- Comprobación de letras repetidas
- Bucles para recorrer la palabra
- Contador de fallos
- Impresión del dibujo del ahorcado según los errores

Para que la pieza has de adivinar, un pequeño juego has de jugar.
El anciano te mira serio y te propone un último reto...

La palabra tiene 8 letras.
Se trata de un objeto que te da suerte
Progreso: _ - - - - -
Introduce una letra: |

Relación con la narrativa

Los minijuegos no están puestos al azar. Forman parte de la historia y avanzan el guion.

Superarlos significa conseguir piezas necesarias para reparar la máquina temporal. Si el jugador falla repetidamente, queda atrapado en la época, lo que conecta mecánicas y narrativa.

6. Control de errores y validación de datos

En el proyecto he tenido que controlar varias situaciones en las que el usuario podía escribir valores incorrectos. Para evitar fallos y que el programa no se cierre de golpe, añadí validaciones básicas, siempre usando cosas del temario visto hasta ahora.

6.1. Validación de números con rangos

Para evitar que el usuario meta texto o números fuera del rango, hice un método propio llamado leerNúmero(int min, int max).

Este método comprueba:

- Si la entrada está vacía
- Si tiene algún carácter no numérico
- Si el número está dentro del rango permitido

```
El primer juego será: ADIVINA EL NÚMERO
Sencillo, ¿no? Para que a la segunda prueba has de jugar,
un número del 1 al 10 tendrás que adivinar.
Indica el número que crees que es (1-10):
17
>> Por favor, introduce un número entre 1 y 10.
|
```

```
¿Nombre del que viajará?
chin chan
Ese nombre no está registrado en XTART en ese curso.

El primer juego será: ADIVINA EL NÚMERO
Sencillo, ¿no? Para que a la segunda prueba has de jugar,
un número del 1 al 10 tendrás que adivinar.
Indica el número que crees que es (1-10):
abc
>> Eso no es un número válido. Solo dígitos.
```

7. Integración de elementos narrativos y jugables

En este proyecto he intentado mezclar programación básica con una pequeña historia interactiva. La idea era que el código no fuera solo un conjunto de funciones sueltas, sino que todo formara parte de una aventura con decisiones y consecuencias.

Para conseguir esto, organicé el programa en actos (como si fuera un juego narrativo). Cada acto se ejecuta como una función distinta (`introduccion()`, `decisiones()`, `actoNumero1()`, etc.) y cada uno contiene partes de la historia, preguntas y opciones.

7.1. Estructura tipo "historia por actos"

Cada acto representa un paso del jugador en la aventura. Por ejemplo:

- En el Acto I, el jugador decide si iniciar la aventura o estudiar.
- En el Acto II, entra en la máquina del tiempo.
- En el Acto IV, toma decisiones importantes (bar o tienda).
- En los actos finales, supera preguntas y minijuegos.

Esto permite que el programa avance de forma lineal, pero con pequeñas decisiones que afectan a lo que ocurre.

```
private static void banner() { 1 usage & Omar1676
    String title = CYAN + "==== CRONOS-1: PROTOCOLO DE SALTO TEMPORAL ===" + RESET;
    escribirLinea(title, pausa: 600);
    System.out.println();
}
```

7.2. Transiciones entre escenas

Para darle más sensación de "historia", usé:

- Textos impresos lentamente (`escribirLento`)
- Pausas entre mensajes
- Separadores visuales
- ASCII art en momentos importantes

Todo esto hace que el usuario sienta que avanza en una aventura.

```
Elige entre iniciar (1) o estudiar (2)
¿Qué decides? ¿Escaparte a una aventura como mandan los cánones,
o quedarte avanzando en el proyecto?
>
```

7.3. Relación entre narrativa y programación

Aunque la historia es creativa, los elementos técnicos están basados en el temario:

- Los minijuegos ayudan a avanzar en la narrativa.
- Las decisiones cambian la ejecución del programa usando if y switch.
- El inventario se va llenando acorde a lo que ocurre.
- Los errores del usuario se integran como "riesgos" en la historia.

De esta forma, la parte narrativa sirve para justificar las decisiones, validaciones y estructuras del código.

```
omar de DAM cogió sus cosas y se aventuró hacia 2001.  
Inició su máquina y comenzó a viajar.  
Una ráfaga del flujo temporal desvió la CRONOS-1 exactamente 0,5 cm a la derecha.
```

7.4. Impacto en la jugabilidad

La historia no es solo decoración; afecta a la programación porque:

- Si el usuario elige mal, puede "morir".
- Si falla un minijuego, debe reiniciar.
- Si no responde correctamente a las preguntas, la historia termina diferente.

La narrativa es el hilo conductor que une todos los conceptos que hemos visto en clase: arrays, funciones, condicionales, bucles y manejo de entradas.

8. Flujo general del programa

El funcionamiento global del programa sigue un orden concreto para que la historia avance de forma lógica y el jugador no se pierda. Todo empieza en el método `main()`, que se encarga de llamar a las distintas partes del juego en el orden correcto. Gracias a esta organización, el código queda más fácil de entender y cada escena está separada en su propio método.

8.1. Inicio del programa

- `banner()`; Muestra el título y la introducción visual.
- `introduccion()`; Presenta la historia y el contexto.
- `decisiones()`; Primera elección del usuario.
- `maquinaDelTiempo()`; Transición narrativa al viaje temporal.
- `actoNumero1()`; Elección del alumno y validación de datos.
- `colores()`; Efectos visuales de la máquina.
- `sigloXVIII()`; Escena con decisiones importantes.
- `laTienda()`; Parte narrativa previa a los minijuegos.
- `adivinaNumero()`; Primer minijuego.
- `ahorcado()`; Segundo minijuego
- `pregunta1()`; Preguntas finales y resolución del viaje.

8.2. Decisiones que alteran el recorrido

El flujo del programa no es completamente lineal.

Hay momentos donde las decisiones cambian la ruta:

- Si el jugador elige estudiar, el programa termina.
- Si pierde un minijuego, puede elegir reiniciar o salir.
- Las respuestas del laboratorio determinan el final que obtiene.

Gracias a estas bifurcaciones, el programa se comporta como una pequeña aventura interactiva.

8.3. Uso de funciones para controlar el flujo

Cada parte del juego está dentro de su propio método. Esto permite:

- Tener el código ordenado
- Modificar escenas sin afectar a otras
- Reutilizar funciones como `leerNumero()` o `escribirLinea()`
- Manejar el ritmo del juego con pausas y mensajes narrativos

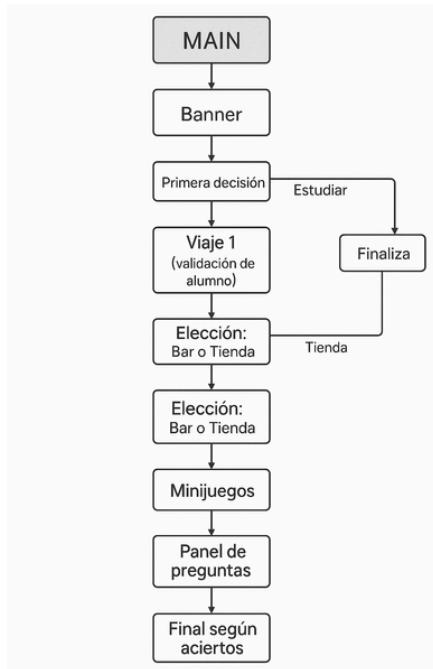
8.4. Finalización del programa

La aventura puede terminar de varias maneras según las decisiones y aciertos del usuario:

- Final bueno → Si supera casi todas las preguntas.
- Finales alternativos → Si falla preguntas del panel.
- Final inmediato → Si toma decisiones incorrectas o pierde minijuegos.

Algunos finales usan System.exit(0) para cerrar el juego en el momento exacto.

8.5. Esquema visual del flujo



9. Conclusión

El desarrollo de este proyecto de programación me ha servido para practicar de forma real todo lo que hemos aprendido durante el trimestre. Aunque la historia es creativa, detrás hay estructuras básicas del lenguaje Java como condicionales, bucles, arrays, métodos y manejo de entradas por teclado. Gracias a esto he podido ver cómo cada parte del temario se puede aplicar en un programa más grande y con varias funcionalidades.

Organizar el código en funciones separadas ha sido fundamental para que la aventura tuviera sentido y no se volviera un bloque de código difícil de entender. También he aprendido la importancia de validar los datos que escribe el usuario, ya que sin esas comprobaciones el programa podría fallar o cerrarse de forma inesperada. Otro punto importante han sido los minijuegos, donde he puesto en práctica la lógica de control de intentos, uso de números aleatorios y el trabajo con caracteres.

En general, este proyecto me ha ayudado a comprender mejor cómo estructurar un programa y cómo combinar diferentes conceptos para crear algo más completo. Aunque todavía me queda mucho por mejorar, siento que he avanzado bastante en la parte práctica de Java y que ahora entiendo mejor cómo funciona un programa con múltiples rutas y decisiones.

BASE DE DATOS

Índice:

1. Introducción
2. Estructura general de la base de datos
 - 2.1. Organización de la base de datos
 - 2.2. Funciones principales
3. Elementos de Base de datos utilizados
 - 3.1. Creación de tablas
 - 3.2. Primaries Keys y Foreign Keys
 - 3.3. Añadir información
4. Creación del diagrama E-R
 - 4.1. Elementos usados y representación de tablas
 - 4.2. Tipo de relaciones
5. Consultas
 - 5.1. Tablas de relaciones principales
 - 5.2. Tablas usadas en JOIN
 - 5.3. Tablas usadas en agrupaciones
 - 5.4. Ejemplos consultas

1. Introducción

Para la presente evaluación trimestral, el presente proyecto, denominado Cronnos, consiste en el diseño, desarrollo e implementación de una base de datos relacional.

Se ha tenido como objetivo entender el concepto de lo que es una base de datos así como es uso del lenguaje SQL y el como se relacionan las tablas llevando a cabo los distintos pasos tanto para su construcción como para las relaciones trabajando con las distintas entidades por las cuales se relacionan siendo coherentes en su relación.

Hemos elaborado mediante el lenguaje SQL varias consultas por las cuales determinamos los conocimientos adquiridos a lo largo de la evaluación.

Mediante el diagrama de entidad-relación hemos dado a conocer de forma más gráfica el tipo de relación que hay entre las tablas.

2. Estructura general de la base de datos

En la base de datos que hemos creado nos hemos inspirado en la base de datos de videojuegos que ya teníamos.

Hemos añadido mas tablas manualmente y aparte tuvimos que añadir las primary keys necesarias y las foreing keys necesarias.

En la primera captura podemos ver como hacemos una de las primary keys y una de las foreing keys necesarias.

```
`id_partida` int NOT NULL AUTO_INCREMENT,  
`id_jugador` int NOT NULL,  
`id_juego` int NOT NULL,  
`horas_juego` double DEFAULT NULL,  
`fecha_partida` date DEFAULT NULL,  
PRIMARY KEY (`id_partida`),  
KEY `id_jugador_idx` (`id_jugador`),  
CONSTRAINT `id_jugador` FOREIGN KEY (`id_jugador`) REFERENCES `jugador` (`id_jugador`)
```

También creamos las tablas necesarias para el funcionamiento y las relaciones necesarias después estuvimos preguntándole a chat gpt si la base de datos tenía algún error o necesitaba alguna información mas para que estuviese tal cual la pedía Dani, estaba todo bien a si que después le añadimos la información necesaria para que en las consultas no saliese vacío y también añadimos información de nuestro juego para que también saliese.

```
LOCK TABLES `plataformas` WRITE;  
/*!40000 ALTER TABLE `plataformas` DISABLE KEYS */;  
INSERT INTO `plataformas` VALUES (1,'Nintendo Switch','Cronos',1),  
(2,'PC','Cronos',1),  
(3,'PS4','Cronos',1),  
(4,'Xbox Series','Cronos',1),  
(5,'PS5','Cronos',1);  
/*!40000 ALTER TABLE `plataformas` ENABLE KEYS */;  
UNLOCK TABLES;
```

3. Estructura general de la base de datos

En este apartado explicaremos como he hizo y que elementos se han usados para la creación de la base de datos.

3.1. Creación de la base de datos

Lo primero que hicimos fue crear un nuevo esquema y nombrarlo en el programa utilizado durante esta evaluación que es el Workbench de mySQL. Lo siguiente fue crear las diferentes tablas relacionadas con el proyecto que vamos a necesitar y añadir las distintas columnas de cada tabla así como determinar de que tipo de dato iban a almacenar.



Column Name	Datatype
id_juego	INT
titulo	VARCHAR(45)
genero	VARCHAR(45)
precio	DOUBLE
id_desarrollador	INT
lanzamiento	DATE

3.2. Primary keys y foreign keys

Lo siguiente que hemos hecho después de crear las tablas y sus columnas fue determinar las primary Keys y las Foreign Keys y encasillar las propiedades de algunas columnas como las Unique Key.

Table: videojuegos										
Table Name:		videojuegos								
Charset/Collation:		utf8mb4			utf8mb4_0900_ai_ci					
Comments:										
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expr
id_juego	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
titulo	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
genero	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
precio	DOUBLE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
id_desarrollador	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
lanzamiento	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Table: partidas										
Table Name:		partidas								
Charset/Collation:		utf8mb4			utf8mb4_0900_ai_ci					
Comments:										
Reign Key Name	Referenced Table	Schema:								
d_game	'timemachine'.`videojuegos`									
d_jugador	'timemachine'.`jugador`									

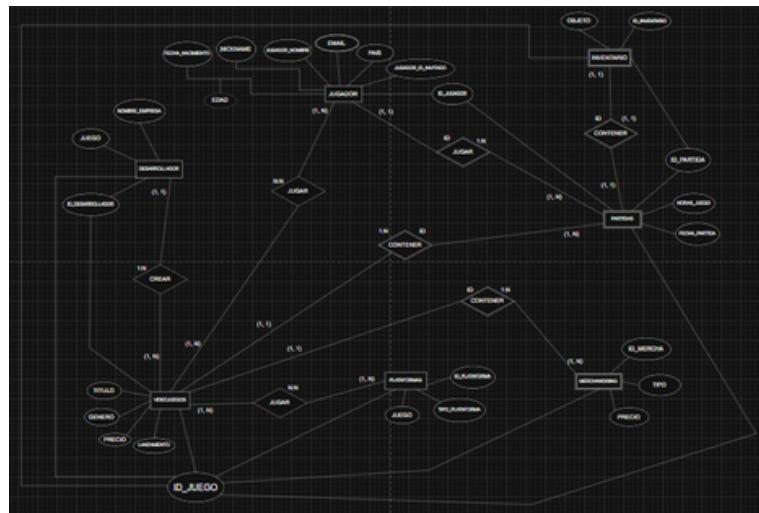
3.3. Añadir información

Para terminar de crear la base de datos se añadieron los datos según el tipo de los mismo para llenar las columnas de las tablas que usariamos más adelante para las consultas realizadas con el lenguaje SQL.

```
1 •  SELECT * FROM timemachine.jugador;
```

4. Creación del diagrama E-R

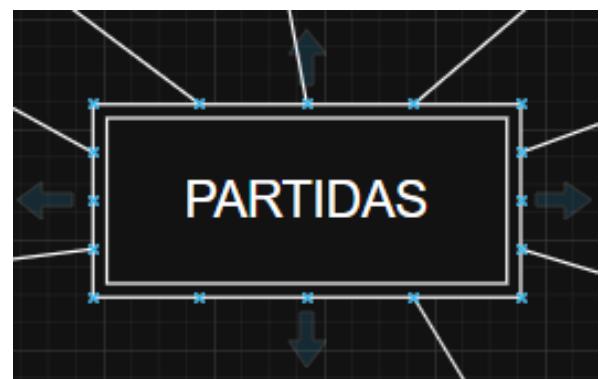
En esta parte del proyecto en base a la tabla hicimos el diagrama Entidad - Relación de la base de datos de nuestro juego.



4.1. Elementos usados y representación de tablas

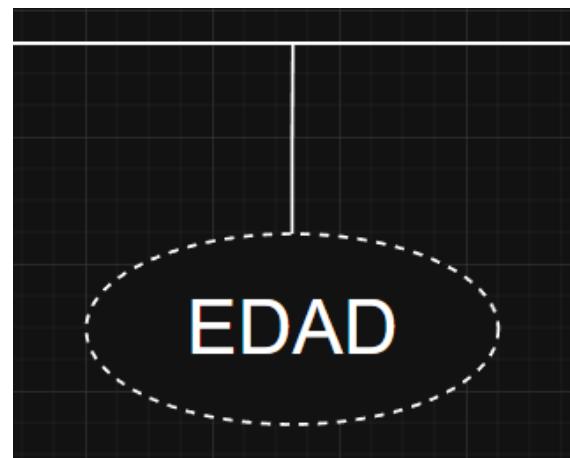
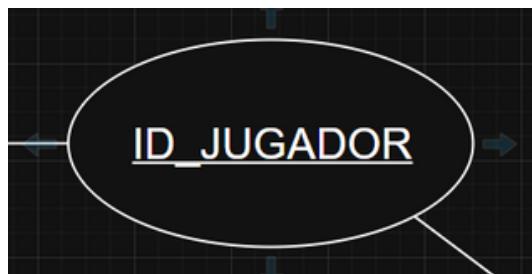
Para empezar hemos usado un programa llamado draw.io para hacer el diagrama.

Dentro del programa usamos rectángulos simples para representar las tablas, dentro de esa selección separamos las entidades fuertes que van en el rectángulo simple de las entidades débiles que se representan con un doble rectángulo.

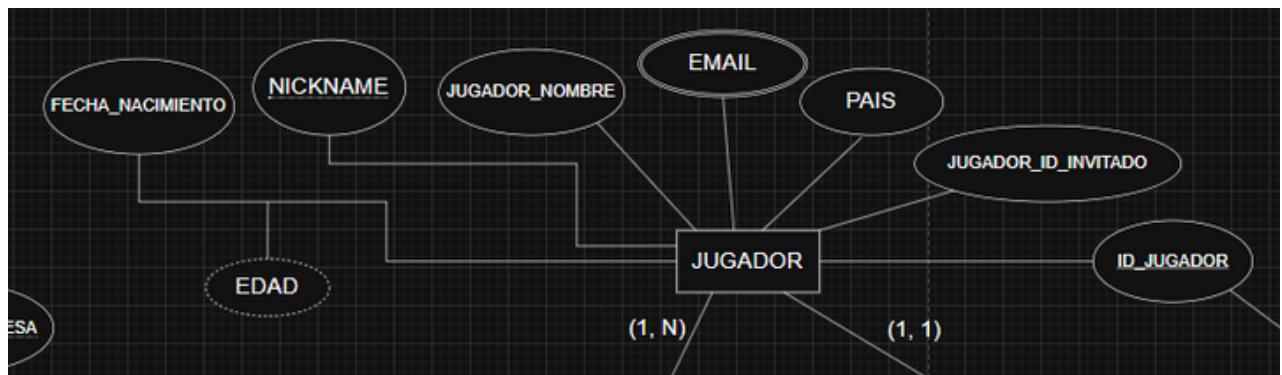


Para representar a las distintas columnas de las tablas usamos óvalos en disposición horizontal y los cuales también separamos por sus distintos tipos:

- Un óvalo simple representa a una columna normal.
- Un óvalo simple con el nombre subrayado es la representación de una primary key en nuestra tabla en la base de datos.
- Un óvalo doble representa a una columna multievaluada.
- Un óvalo simple con el nombre subrayado por una línea de puntos representa a una Unique Key.
- Un óvalo hecho por puntos representa una columna con un dato derivado.



Para terminar de representar una tabla en el diagrama unimos las tablas con sus respectivas columnas mediante líneas.



4.2. Tipos de relación

Para representar la relación cardinal que tienen entre las tablas usamos los rombos:

- Usamos un rombo simple para relacionar dos entidades fuertes cuyas existencias no dependen de otra tabla para existir.
- Usamos un doble rombo para relacionar una entidad fuerte con una débil que es aquella que sin la entidad fuerte no podría existir.

Resumen de Consultas SQL

En el sistema de gestión de videojuegos se han desarrollado diversas consultas SQL destinadas a analizar la información almacenada y a facilitar la interpretación de los datos. Estas consultas permiten obtener una visión clara del uso de la plataforma, de la actividad de los jugadores y de la relación entre los diferentes elementos del sistema.

En primer lugar, se incluyen consultas orientadas al análisis del comportamiento de los usuarios, como la que muestra las partidas jugadas por cada jugador, junto con el tiempo que han dedicado a cada título. También se identifican los jugadores que han invitado a otros, así como aquellos que han acumulado más horas que la media o que han realizado la partida más larga registrada. Estas consultas proporcionan una imagen general del nivel de actividad dentro de la comunidad.

Por otro lado, se han creado consultas que relacionan videojuegos con otros elementos del sistema. Entre ellas se incluyen las que muestran el merchandising vinculado a cada juego, las plataformas donde están disponibles y la empresa desarrolladora responsable de cada título. Gracias a estas relaciones es posible analizar el catálogo desde múltiples perspectivas, tanto comerciales como técnicas.

Además, se han diseñado consultas que permiten obtener métricas globales, como el total de horas jugadas por videojuego, el número de jugadores por país o la comparación del precio de cada título respecto al promedio general. Estas métricas resultan útiles para identificar tendencias y patrones de uso, así como para apoyar la toma de decisiones dentro del proyecto. En conjunto, todas estas consultas proporcionan una base sólida para el análisis del sistema, permitiendo estudiar la actividad de los usuarios, conocer mejor el catálogo de videojuegos y obtener indicadores relevantes sobre el funcionamiento de la plataforma. Constituyen una parte esencial del proyecto, ya que facilitan la comprensión y explotación de los datos almacenados.

```
/*Jugadores que han jugado más horas que el promedio de horas invertidas en cada juego*/
SELECT j.nickname, j.pais, p.horas_juego, v.titulo
FROM partidas p
JOIN jugador j
ON p.id_jugador = j.id_jugador
JOIN videojuegos v
ON p.id_juego = v.id_juego
WHERE p.horas_juego > (
    SELECT AVG(p2.horas_juego)
    FROM partidas p2
    WHERE p2.id_juego = p.id_juego
)
ORDER BY v.titulo, p.horas_juego DESC;
/*Juegos más caros que el precio promedio*/
SELECT titulo, precio
FROM videojuegos
WHERE precio = (
    SELECT AVG(precio)
    FROM videojuegos
);
```

SISTEMAS INFORMATICOS

Índice: Elementos de programación utilizados

1. Introducción

1.1 Objetivo del script

1.2 Contexto del uso

2. Estructura general del script

2.1. Funcionamiento básico

2.2. Estructura general del menú

3. Componentes y elementos de programación utilizados

3.1. Comandos principales

3.2. Uso de etiquetas para organizar el menú

3.3. Manejo de entradas del usuario

3.4. Uso de ASCII art y formato visual

4. Estructura detallada del script

4.1. Configuración inicial

4.2. Generación del menú principal

4.3. Sistemas de selección y validación

4.4 Ejecución de cada bloque

4.5 Retorno al menú

5. Elementos de programación utilizados

5.1. Herramientas de presentación

5.2. Herramientas de desarrollo

5.3. Bases de datos

5.4. Virtualización

5.5. Sistemas operativos

5.6. Documentación y archivos del proyecto

6. Ventajas del script

7. Limitaciones del script

8. Conclusión

1. Introducción

El script tiene como finalidad centralizar en un menú único todos los accesos necesarios para el proyecto. Como herramientas de programación (Visual Studio Code, IntelliJ), instaladores, documentos, máquinas virtuales (VMware, VirtualBox). Esto pretende facilitar el flujo de trabajo y reducir el tiempo invertido en buscar recursos.

1.2 Contexto de uso

El script se creó para un proyecto académico donde los recursos estaban dispersos entre programas instalados, carpetas locales y páginas web. Funciona como un panel de control que simplifica el acceso a programas como Visual Studio Code, IntelliJ o MySQL Workbench, plataformas externas como GitHub o Canva, instaladores y documentación del proyecto, mejorando la organización y eficiencia del usuario.

2. Objetivo del script

2. Descripción general del script

El script es un menú interactivo desarrollado en Batch que permite acceder de forma rápida y centralizada a todas las herramientas, programas, documentos y recursos web del proyecto. Cada opción del menú abre directamente un software, un archivo, un instalador o una página web, y al finalizar la acción regresa automáticamente al menú principal. Su diseño sencillo y eficiente facilita la organización del trabajo y reduce el tiempo invertido en buscar recursos dispersos.

2.1. Funcionamiento básico

El script usa un menú interactivo donde el usuario elige una opción numérica. Según la selección, se ejecuta una acción: abrir un programa, iniciar un instalador, abrir una carpeta o acceder a una web oficial.

2.2. Estructura general del menú

El menú está dividido en bloques temáticos:

- Presentaciones (**Canva**)
- Desarrollo (**VS Code, GitHub, IntelliJ**)
- Bases de datos
- Virtualización (**VirtualBox y VMware**)
- Sistemas operativos
- Documentación del proyecto



Cada opción envía al usuario a una sección específica mediante saltos (**goto**).

3. Componentes y elementos de programación utilizados

3.1. Comandos principales

@echo off → oculta instrucciones.

```
@echo off
```

chcp 65001 → habilita caracteres UTF-8.

```
chcp 65001
```

start → abre rutas o direcciones web.

```
start ""
```

set /p → captura la entrada del usuario.

```
set /p
```

if → compara la opción elegida.

```
if /i
```

goto → redirige a etiquetas.

```
goto
```

3.2. Uso de etiquetas para organizar el menú

Cada opción del menú corresponde a una etiqueta ([:github](#), [:canva](#), [:vscode](#)). Esto permite dividir el script en secciones independientes y fáciles de ampliar.

3.3. Manejo de entradas del usuario

El script utiliza entradas del usuario para decidir qué acción ejecutar en cada momento. Al mostrar el menú principal, solicita al usuario que ingrese un número correspondiente a la opción deseada mediante el comando set /p. Esta entrada se compara con las opciones disponibles usando condicionales if /i, que permiten reconocer la respuesta sin importar si se ingresa en mayúsculas o minúsculas.

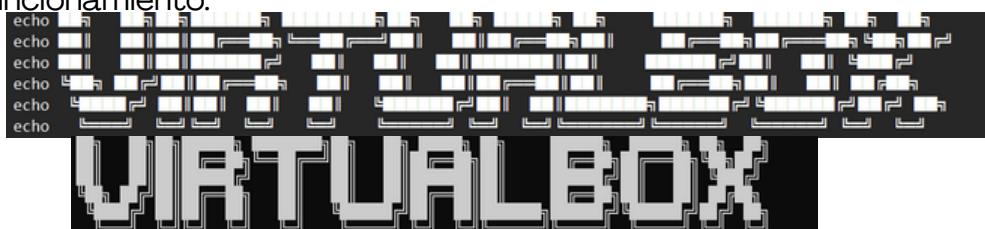
Además, dentro de cada sección, el script pregunta al usuario si desea abrir un recurso específico (programa, archivo o página web) mediante otra entrada con set /p. Según la respuesta, normalmente "s" para sí o "n" para no, se ejecuta la acción correspondiente con el comando start o se regresa al menú principal.

```
set /p respuesta=
if /i "%respuesta%"=="s" start ""
```

```
if "%opcion%"=="1"
if "%opcion%"=="2"
if "%opcion%"=="3"
if "%opcion%"=="4"
if "%opcion%"=="5"
if "%opcion%"=="6"
if "%opcion%"=="7"
if "%opcion%"=="8"
if "%opcion%"=="9"
```

3.4. Uso de ASCII art

Las secciones incluyen encabezados decorativos para hacer la interfaz más atractiva sin afectar el funcionamiento.



4. Estructura detallada del script

4.1. Configuración inicial

El script comienza configurando el entorno de ejecución. Ajusta el título de la ventana para identificar el proyecto, cambia la codificación de caracteres a UTF-8 (chcp 65001) para permitir el uso de caracteres especiales y limpia la pantalla. Además, se desactiva el eco de comandos (@echo off) para que solo se muestren los mensajes relevantes al usuario, ofreciendo una interfaz más limpia y profesional.

```
@echo off  
title Trabajo Final  
chcp 65001  
cls
```

4.2. Generación del menú principal

A continuación, el script despliega un menú interactivo con todas las opciones disponibles, numeradas para facilitar la navegación. Cada opción corresponde a un recurso específico: programas, documentos, páginas web, instaladores o máquinas virtuales. El menú está diseñado para ser claro y visualmente organizado, de manera que el usuario pueda identificar rápidamente la opción deseada.

```
echo =====
echo 1. Acceso a presentacion Canva (online)
echo 2. Acceso a Canva descargado (PDF)
echo 3. Acceso a GitHub
echo 4. Abrir Visual Studio Code
echo 5. Acceso a descarga de Visual Studio Code
echo 6. Abrir documentos HTM, CSS y JS
echo 7. Abrir IntelliJ
echo 8. Acceso descarga IntelliJ
echo 9. Abrir PDF (Adobe/Chrome/Opera)
echo 10. Abrir MySQL Workbench
echo 11. Acceso descarga MySQL Workbench
echo 12. Acceso descarga VirtualBox
echo 13. Instalar VirtualBox
echo 14. Abrir VirtualBox
echo 15. Acceso descarga VMware
echo 16. Instalar VMware
echo 17. Abrir VMware
echo 18. Descarga Windows 11 y Ubuntu Server
echo 19. Acceso a Documentación/Memoria/Aplicación
echo 0. Salir
echo =====
echo.
set /p opcion=Selecciona una opcion:
```

4.3. Sistema de selección y validación

El usuario selecciona una opción ingresando un número mediante set /p. Esta entrada se guarda en una variable y se compara con una serie de condicionales if /i para determinar la acción correcta. Este sistema permite validar la entrada y asegurarse de que el script ejecute únicamente la acción correspondiente, evitando errores por entradas inválidas.

```
if "%opcion%"=="1" goto canva
if "%opcion%"=="2" goto canvas_pdf
if "%opcion%"=="3" goto github
if "%opcion%"=="4" goto vscode
if "%opcion%"=="5" goto vscode_download
if "%opcion%"=="6" goto webdocs
if "%opcion%"=="7" goto intellij
if "%opcion%"=="8" goto intellij_download
if "%opcion%"=="9" goto pdf
if "%opcion%"=="10" goto mysql
if "%opcion%"=="11" goto mysql_download
if "%opcion%"=="12" goto vbox_download
if "%opcion%"=="13" goto vbox_install
if "%opcion%"=="14" goto vbox_open
if "%opcion%"=="15" goto vmware_download
if "%opcion%"=="16" goto vmware_install
if "%opcion%"=="17" goto vmware_open
if "%opcion%"=="18" gotoisos
if "%opcion%"=="19" goto docs
if "%opcion%"=="0" exit
goto menu
```

4.4. Ejecución de cada bloque

Cada bloque de acción sigue un patrón similar:

Muestra un encabezado decorativo para identificar el recurso.

Solicita confirmación al usuario antes de ejecutar la acción.

Si el usuario confirma, se utiliza start para abrir la ruta del programa, archivo o enlace web.

Una vez ejecutada la acción, el bloque finaliza y el control se devuelve al menú principal.

Este diseño modular garantiza que cada recurso se maneje de forma independiente y controlada.

```
:canva
echo
echo
echo
echo
echo
echo
set /p respuesta=¿Quieres iniciar Canva? (s/n):
```

```
:vscode_download
echo
echo
echo
echo
echo
echo
echo.
set /p respuesta=¿Quieres abrir la descarga de VS Code? (s/n):
```

```
:intellij_download
echo
echo
echo
echo
echo
echo
set /p respuesta=¿Quieres abrir la descarga de IntelliJ? (s/n):
```

```
:vbox_install
echo
echo
echo
echo
echo
echo.
set /p respuesta=¿Quieres instalar VirtualBox? (s/n):
```

4.5. Retorno al menú

Tras ejecutar cualquier acción, el script utiliza goto menu para regresar automáticamente al menú principal. Esto permite que el usuario continúe utilizando el panel de control sin necesidad de reiniciar el script, asegurando un flujo de trabajo continuo y eficiente.

```
if /i "%respuesta%"=="s" start "" "https://www.canva.com/"
```

```
if /i "%respuesta%"=="s" start "" "https://github.com/"
```

5. Elementos de programación utilizados

El script hace uso de varias herramientas y estructuras propias de Batch para lograr su funcionalidad de manera sencilla y eficiente:

5.1. Variables

Se utilizan variables para capturar la entrada del usuario (set /p) tanto en la selección del menú principal como en la confirmación dentro de cada bloque. Esto permite almacenar temporalmente la respuesta y usarla en condicionales para determinar qué acción ejecutar.

set /p

5.2. Condicionales

Se emplean condicionales if /i para comparar las entradas del usuario, ignorando mayúsculas y minúsculas. Esto garantiza que el script responda correctamente independientemente de cómo el usuario escriba su respuesta y evita errores en la ejecución.

if /i

5.3. Saltos de sección

El comando goto se utiliza para dirigir la ejecución a distintas secciones del script según la opción seleccionada. Esto organiza el código de manera modular y permite retornar al menú principal tras ejecutar cada acción.

goto menu

5.4. Comandos del sistema

start para abrir aplicaciones, documentos o enlaces web.

cls para limpiar la pantalla y mantener la interfaz ordenada.

title para establecer el título de la ventana.

chcp 65001 para habilitar la codificación UTF-8 y permitir caracteres especiales.

@echo off para ocultar la visualización de los comandos y mostrar únicamente los mensajes relevantes al usuario.

```
@echo off  
title Trabajo Final  
chcp 65001  
cls
```

6. Ventajas

- Reúne todos los recursos en un único menú.
- Ahorra tiempo al evitar búsquedas repetitivas.
- Interfaz simple y accesible.
- Portátil y fácil de usar.
- Modular y ampliable gracias a su estructura con etiquetas.

7. Limitaciones del script

- Las rutas dependen del usuario y su equipo.
- Solo funciona en Windows.
- No verifica si las aplicaciones están instaladas.
- Requiere que los instaladores estén en la ubicación esperada.

8. Conclusión

Este script proporciona una forma rápida y organizada de acceder a todas las herramientas necesarias para el proyecto. Aunque Batch es limitado, su simplicidad permite crear un menú interactivo eficaz, ampliable y muy útil en contextos académicos. Sirve como centro de operaciones para abrir programas, instaladores, sistemas operativos y documentos, mejorando la eficiencia en el trabajo cotidiano del proyecto.

GIT – Proyecto “Máquina del Tiempo / Cronos-1”

INDICE:

- 1. Introducción**
- 2. Objetivos del uso de Git**
- 3. Instalación y configuración inicial**
- 4. Creación del repositorio**
- 5. Comandos básicos utilizados**
- 6. Estructura del repositorio del proyecto**
- 7. Flujo de trabajo (Workflow)**
- 8. Gestión de versiones y commits**
- 9. Conexión con GitHub y repositorio remoto**
- 10. Conclusión**

1. Introducción al uso de Git en el proyecto

Durante el desarrollo de mi proyecto final del trimestre he utilizado Git como sistema de control de versiones para organizar el código, guardar las diferentes versiones del juego y mantener un registro del avance.

El repositorio se alojó en GitHub y he trabajado con él desde Visual Studio Code e IntelliJ IDEA.

Git me ha permitido:

- Evitar pérdida de código
- Volver a una versión anterior si algo fallaba
- Trabajar en diferentes pantallas del proyecto sin romper la estructura
- Mantener una copia sincronizada entre casa y el aula

2. Instalación y configuración inicial

Comandos utilizados en la configuración:

```
git config --global user.name "Omar El Yamani"  
git config --global user.email "tuemail@ejemplo.com"
```

(solo una vez para saber desde que cuenta se operara de git)

Creación del repositorio en local:

```
PS C:\Users\usuario\Desktop\Proyecto final> git init
```

(para inciarlo)

Conexión con GitHub:

```
git remote add origin https://github.com/omar1676/Trabajo-Fin-Trimestre.git
```

(para enlazarlo todo a un mismo repositorio)

3. Estructura del repositorio

Mi repositorio contiene:

- Archivos HTML
- Carpeta del proyecto Java (HistoriaAventura / Cronos-1)
- Archivo CSS en conjunto
- Carpeta de imágenes y videos
- README del proyecto

ESTRUCTURA REPRESENTADA:

/Proyecto-Final/

 /html

 /css

 /img

 /java

 README.md

4. Creación del repositorio

Para gestionar el proyecto con Git, el primer paso fue crear un repositorio.

Este repositorio me permite almacenar todas las versiones del proyecto y mantener un control completo sobre la evolución del código.

4.1. Creación del repositorio local

Desde la carpeta de mi proyecto ejecuté el siguiente comando:

```
PS C:\Users\usuario\Desktop\Proyecto final> git init
```

Este comando inicializa un repositorio vacío en local y crea la carpeta .git, donde Git guardará toda la información del historial del proyecto.

4.2. Añadir archivos al repositorio

Después de crear el repositorio, añadí todos los archivos del proyecto a la zona de preparación:

```
PS C:\Users\usuario\Desktop\Proyecto final> git add .
```

Con este comando Git detecta todos los archivos del proyecto y los deja listos para ser registrados.

4.3. Primer commit

Para guardar formalmente la primera versión del proyecto ejecuté:

```
PS C:\Users\usuario\Desktop\Proyecto final> git commit -m "Versión inicial del proyecto"
```

Con este commit comienzo el historial del proyecto.

4.4. Creación del repositorio remoto en GitHub

Para subir mi proyecto a la nube y tener una copia segura, creé un repositorio en GitHub llamado:

Trabajo-Fin-Trimestre

Una vez creado, conecté el repositorio local con el remoto:

```
PS C:\Users\usuario\Desktop\Proyecto final> git remote add origin https://github.com/omar1676/Trabajo-Fin-Trimestre.git
```

4.5. Subida del proyecto a GitHub

Finalmente, envié la primera versión del proyecto a GitHub usando:

```
PS C:\Users\usuario\Desktop\Proyecto_final> git push -u origin main
```

Esto permitió sincronizar mi repositorio local con el remoto y a partir de ese momento pude subir cambios fácilmente.

5. Comandos básicos utilizados

Durante el desarrollo del proyecto utilicé una serie de comandos básicos de Git que permitieron registrar cambios, gestionar versiones y sincronizar el repositorio con GitHub.

A continuación, se explican los comandos principales y su función dentro del flujo de trabajo.

```
PS C:\Users\usuario\Desktop\Proyecto final> git status
```

Muestra el estado del repositorio:

- Archivos modificados
- Archivos no añadidos
- Archivos listos para commit

```
PS C:\Users\usuario\Desktop\Proyecto final> git add
```

Añade todos los archivos modificados a la zona de preparación .

Con esto Git sabe qué archivos quiero incluir en el próximo commit.

```
PS C:\Users\usuario\Desktop\Proyecto final> git push
```

Envía los commits al repositorio remoto de GitHub.

Permite hacer copia en la nube y compartir el proyecto

```
PS C:\Users\usuario\Desktop\Proyecto final> git pull
```

Descarga actualizaciones del repositorio remoto.

```
PS C:\Users\usuario\Desktop\Proyecto final> git remote -v
```

Permite ver a qué repositorio está conectado el proyecto.

6. Estructura del repositorio

El proyecto está organizado dentro de un repositorio Git que contiene varios archivos y carpetas distribuidos de forma lógica.

La estructura del repositorio permite separar correctamente el código HTML, CSS, recursos multimedia y la documentación del proyecto.

A continuación se muestra la estructura general del repositorio:

omar1676	he añadido un comentario	2ea77ce · 4 hours ago	27 Commits
📁 Sistemas informáticos	Reemplazo total con contenido nuevo	3 days ago	
📁 imágenes	he añadido imágenes y juntado la parte final	4 hours ago	
📁 java	He añadido un archivo README	3 days ago	
📁 sonidos	He añadido nueva pagina de html y nuevas funciones (músi...	3 days ago	
📁 Base de datos.zip	Reemplazo total con contenido nuevo	3 days ago	
📁 CAPITULO_2A.HTML	eliminación sobrante código	yesterday	
📁 CAPITULO_2B.html	He modificado las imágenes y el código	2 days ago	
📁 Script.java	He separado los archivos	3 days ago	
📁 acto1.html	He metido la carpeta java con el proyecto casi lista	last week	
📁 acto2.1.html	He añadido nueva pagina de html y nuevas funciones (músi...	3 days ago	
📁 bar.html	He añadido nueva pagina de html y nuevas funciones (músi...	3 days ago	
📁 creditos.html	he añadido un comentario	4 hours ago	
📁 emperios.html	He añadido nueva pagina de html y nuevas funciones (músi...	3 days ago	
📁 escapada.html	He añadido nueva pagina de html y nuevas funciones (músi...	3 days ago	
📁 finalbueno.html	He modificado rutas y unido todo a un css	5 hours ago	
📁 finalbueno2.html	he añadido imágenes y juntado la parte final	4 hours ago	
📁 finalmalo.html	he añadido imágenes y juntado la parte final	4 hours ago	
📁 finalmedio.html	he añadido imágenes y juntado la parte final	4 hours ago	
📁 finalmedio2.html	he añadido imágenes y juntado la parte final	4 hours ago	
📁 htm introducción.html	He metido la carpeta java con el proyecto casi lista	last week	
📁 labonatorio.html	he modificado y unificado trabajos	3 days ago	
📁 pelea_mano.html	He añadido nueva pagina de html y nuevas funciones (músi...	3 days ago	
📁 revision.html	He añadido nueva pagina de html y nuevas funciones (músi...	3 days ago	
📁 rutaA1.html	He modificado las imágenes y el código	2 days ago	
📁 rutaB1.html	He cambiado el color, la sombra de los textos de las opciones...	2 days ago	
📁 rutaC1.html	He modificado la barra progresiva y la ventana de texto en el...	2 days ago	
📁 rutaC1.js	He modificado rutas y unido todo a un css	5 hours ago	
📁 stylefinal.css	he añadido imágenes y juntado la parte final	4 hours ago	
📁 tumbah.html	He juntado y modificado los archivos ruta 2A	3 days ago	
📁 viaje.html	eliminación sobrante código	yesterday	

7. Flujo de trabajo

Para desarrollar el proyecto utilizando Git seguí un flujo de trabajo claro y constante.

Este método me permitió mantener el código ordenado, evitar pérdidas de información y controlar cada cambio realizado.

A continuación se explica paso a paso el proceso que utilicé durante todo el desarrollo del proyecto.

7.1. Comprobar el estado del repositorio

Con este comando podía ver si tenía archivos nuevos, modificados o pendientes de commit.

```
PS C:\Users\usuario\Desktop\Proyecto final> git status
```

7.2. Realizar modificaciones en el proyecto

Aquí editaba:

- HTML (pantallas del juego)
- CSS (diseño, BEM, pantallas finales...)
- Código Java del juego
- Imágenes o recursos

Cada modificación se hacía de manera incremental para no romper el proyecto.

7.3. Añadir cambios a la zona de preparación

una vez modificados los archivos, los preparaba para guardarlos:

```
○ PS C:\Users\usuario\Desktop\Proyecto final> git add .
```

7.4. Crear un commit con descripción clara

Los mensajes de commit eran importantes para identificar qué se había cambiado.

He modificado y arreglado problemas con java, tambien añadido parte de la historia	0878a73		
omar1676 committed last week			
He añadido una carpeta con sonidos en wav para poner la musica en java	63b19d2		
omar1676 committed last week			
He metido la carpeta java con el proyecto casi listo	cf57d89		
omar1676 committed last week			
He añadido un nuevo html y lo lo linkeado tanto con acto2.1 como con escapada	3da8f9f		
omar1676 committed last week			

7.5. Subir los cambios a GitHub

```
PS C:\Users\usuario\Desktop\Proyecto final> git push
```

De esta forma aseguraba tener una copia de seguridad actualizada y accesible desde cualquier dispositivo.

7.6. Descargar cambios del remoto

Cuando el repositorio remoto tenía actualizaciones o si Git detectaba desincronización, usaba:

```
PS C:\Users\usuario\Desktop\Proyecto final> git pull
```

8. Gestión de versiones y commits

La gestión de versiones fue una parte fundamental en el desarrollo del proyecto.

Git me permitió mantener un historial completo de todos los cambios realizados, identificar errores, recuperar versiones anteriores y documentar la evolución del proyecto de forma clara y ordenada.

8.1. Uso de commits frecuentes

Realicé commits de manera constante cada vez que completaba una parte del proyecto.

Esto me permitió mantener un historial limpio y fácil de entender.

Ejemplos de buenas prácticas que seguí:

- Hacer commits pequeños en lugar de uno muy grande
- Commit por cada pantalla nueva
- Commit por corrección importante
- Commit por cambios estructurales en CSS o HTML
- Commit cuando avanzaba en el código Java

8.2. Mensajes de commit descriptivos

utilicé mensajes claros y descriptivos para indicar qué cambios se habían realizado.

The screenshot shows a portion of a GitHub repository's commit history. It displays three commits from a user named 'omar1676' made 'last week'. Each commit has a clear, descriptive message:

- He mejorado errores de linea y añadido un apartado en el cssy creado un nuevo html
- he implementado mas sonidos y una carpeta de sonidos
- he añadido canciones

Each commit is timestamped as being made 'last week'.

8.3. Revisión del historial de versiones

8.3. Revisión del historial de versiones

```
PS C:\Users\usuario\Desktop\Proyecto final> git log
```

para revisar el historial completo del repositorio.

Esto me permitió:

- Ver qué cambios hice cada día

- Comprobar quién realizó los commits (yo mismo)

- Detectar versiones estables

- Volver a un commit anterior si algo fallaba

9. Conexión con GitHub y repositorio remoto

Para asegurar que el proyecto estuviera respaldado en la nube y disponible desde cualquier dispositivo, conecté el repositorio local creado en mi ordenador con un repositorio remoto en GitHub.

Esta conexión permitió sincronizar el trabajo, subir versiones actualizadas del proyecto y mantener un historial completo accesible online.

9.1. Creación del repositorio remoto en GitHub

Primero creé un repositorio en GitHub llamado:

Trabajo-Fin-Trimestre.

GitHub generó una URL única para el repositorio remoto:

<https://github.com/omar1676/Trabajo-Fin-Trimestre.git>

Este enlace sirve como punto de conexión entre mi repositorio local y GitHub.

9.2. Vinculación del repositorio local con el remoto

Una vez creado el repositorio en GitHub, añadí la conexión desde mi proyecto usando el comando:

```
git remote add origin https://github.com/omar1676/Trabajo-Fin-Trimestre.git
```

Con este paso, Git sabe que la “rama principal” del proyecto está asociada a ese repositorio remoto.

9.3. Subida inicial del proyecto

Después de añadir los archivos y hacer el primer commit, envié toda la información al repositorio remoto:

El parámetro -u establece la relación entre la rama local main y la rama main del repositorio remoto, facilitando futuros comandos.

9.4. Sincronización continua durante el desarrollo

Durante el proyecto continué subiendo nuevas versiones mediante:

```
PS C:\Users\usuario\Desktop\Proyecto_final> git push origin main
```

Esto garantizó:

- Copia de seguridad constante
- Acceso al proyecto desde cualquier ordenador
- Historial actualizado en GitHub
- Evitar pérdida de información

```
PS C:\Users\usuario\Desktop\Proyecto_final> git pull origin main
```

Además, cuando GitHub tenía cambios que mi ordenador no tenía, actualizaba el repositorio local con:

9.5. Ventajas de usar GitHub como repositorio remoto

- Permite guardar el proyecto en la nube
- Facilita mostrar el proyecto al profesor o futuros empleadores
- Aporta profesionalidad y buenas prácticas
- Permite trabajar desde varios dispositivos
- Hace posible colaborar en equipo mediante pull requests

10. Conclusión

El uso de Git ha sido fundamental para el desarrollo y la organización de este proyecto. Gracias al control de versiones he podido trabajar de una forma estructurada, segura y mucho más profesional, permitiéndome registrar cada cambio, corregir errores rápidamente y mantener una copia de seguridad actualizada en GitHub.

Git no solo ha facilitado la gestión técnica del proyecto, sino que también me ha ayudado a comprender cómo se trabaja realmente en el entorno del desarrollo web y software: commits claros, historial de versiones, trabajo ordenado por etapas y sincronización con un repositorio remoto.

Estas prácticas son esenciales en cualquier proyecto colaborativo o profesional.

En resumen, Git ha sido una herramienta clave para garantizar la estabilidad, la trazabilidad y la evolución del proyecto. Su integración con GitHub ha permitido mantener el trabajo siempre accesible y respaldado, demostrando la importancia del control de versiones dentro del ciclo de desarrollo moderno.