GUION PARA EL SEGUNDO ENTREGABLE

ALBERT BELTRAN CARBONELL

Sumario

ACCESO A DATOS....2
DESARROLLO DE INTERFACES....5
PROGRAMACIÓN DE SERVICIOS Y PROCESOS....7
PROGRAMACIÓN MULTIMEDIA....8
Gestión Empresarial....9

ACCESO A DATOS

Punto A) Explica las ventajas que has visto que ha tenido el desarrollo de una clase (POO) para gestionar las conexiones de la base de datos.

El uso de una clase orientada a objetos (POO) para gestionar las conexiones a una base de datos MySQL utilizando MySQLi en PHP ofrece varias ventajas significativas. Aquí detallo algunas de las principales ventajas que he observado al desarrollar una clase para este propósito:

1. Modularidad y Reutilización del Código

Modularidad: Al encapsular la lógica de conexión y operaciones en una clase, puedes gestionar todas las operaciones de la base de datos desde un único archivo. Esto facilita la gestión y hace que el código sea más ordenado y modular.

Reutilización: Puedes reutilizar esta clase en varios proyectos sin necesidad de reescribir el código para las operaciones básicas de la base de datos (como conectar, ejecutar consultas, etc.). Solo necesitas incluir el archivo de la clase.

2. Facilidad de Mantenimiento

Cambios Centralizados: Si necesitas cambiar la configuración de la conexión (por ejemplo, cambiar el nombre de la base de datos, el usuario o la contraseña), solo debes modificar el código en un lugar, en lugar de hacerlo en cada archivo que conecta a la base de datos.

Mantenimiento más sencillo: Agrupar todas las operaciones relacionadas con la base de datos en una clase facilita la depuración y mantenimiento del código, ya que cualquier error o cambio necesario puede ser fácilmente localizado y corregido.

3. Seguridad Mejorada

Prevención de Inyecciones SQL: Al utilizar sentencias preparadas y consultas parametrizadas dentro de la clase, se reduce el riesgo de ataques de inyección SQL. La clase puede incluir métodos que gestionen automáticamente la creación de consultas preparadas, lo que hace que el uso de parámetros sea obligatorio.
Encapsulación de Información Sensible: Las credenciales de la base de datos (usuario, contraseña, host, etc.) están protegidas dentro de la clase y no expuestas directamente en el código fuente.

4. Manejo de Errores Eficiente

Puedes implementar un sistema de manejo de errores dentro de la clase, por ejemplo, utilizando excepciones (try-catch). Esto permite capturar y gestionar errores de conexión o de consulta de manera centralizada, mejorando la robustez de la aplicación.
Puedes personalizar los mensajes de error para hacerlos más comprensibles o incluso registrarlos en un archivo de logs para auditoría y seguimiento.

5. Facilita la Implementación del Patrón Singleton

Usar una clase permite la implementación del patrón Singleton, lo que asegura que solo haya una instancia de la conexión a la base de datos activa en un momento dado. Esto mejora la eficiencia al evitar múltiples conexiones innecesarias.

Punto B) Explica que hemos utilizado MySQL y que de momento toda la lógica de la clase esta en función de MySQL.

Básicamente hemos utilizado MySQL para nuestra base de datos en phpmyadmin, por lo que toda la logica de la aplicación ira en torno a MySQL mediante el conector mysqli.

Punto C) Explica que conector hemos utilizado

La lógica de la aplicación ira en torno a MySQL mediante el conector mysqli.

Punto D) Indica el funcionamiento a nivel global de la clase desarrollada, tanto a nivel de propiedades como a nivel de parametros

Explicación en el video de como funciona el código.

Punto E) Indica tanto los métodos como el enrutador, y la forma en la que hemos convertir el enrutador en un proveedor de microservicios.

Punto F) Explica tanto los métodos de seleccion, insercion, eliminacion, actualizacion, etc

Explicación en el video de como funciona el código.

Punto F) Explica que json viene, y json vuelve

Explicación en el video de como funciona el código.

Punto G) Explica la integración que ha habido entre el proyecto de acceso a datos y el proyecto de sistemas de gestión empresarial

Explicación en el video de como funciona el código.

Punto K) Explica los procedimientos almacenados que hemos realizado en la base de datos y los procedimientos con parámetros

Explicación en el video de como funciona el código. Carpeta 007

DESARROLLO DE INTERFACES

Punto A) Explica el ejercicio que realizamos en clase acerca de machine learning

Explicación del archivo 022 donde creamos una ia que reconoce mediante imagen formas y los define como pixeles en un canvas.

Punto B) Explicad las diferencias entre las interfaces de usuario graficas, de consola, y las naturales

Las interfaces de usuario pueden ser:

GUI (Gráfica): Usa ventanas, iconos, botones, y menús; es fácil de usar e intuitiva. Ejemplos: Windows, macOS.

CLI (Consola): Usa texto; requiere comandos escritos. Es rápida y potente para usuarios avanzados. Ejemplos: terminal de Linux, cmd de Windows.

NUI (Natural): Usa gestos, voz, y toques; simula la interacción humana. Ejemplos: asistentes de voz, pantallas táctiles, realidad aumentada.

Punto C) Explica el ejercicio que hicimos mediante el cual utilizamos la libreria de Google para reconocimiento de voz

Explicación de la carpeta 003. Donde utilizamos la librería pyaudio.

Punto D) Explica los ejercicios que hicimos en cuanto a mediapipe con opencv para reconocimiento de formas corporales

Explicación de la carpeta 004. Donde utilizamos la librería mediapipe para reconocimiento de manos, cuerpo y cabeza.

Punto E) Explica el ejercicio que hicimos de mover el cursor del raton utilizando la nariz - o bien si tu has hecho proyectos similares, pues tus proyectos

Explicación de la carpeta 004. Donde utilizamos la librería mediapipe para reconocimiento de manos, cuerpo y cabeza.

PROGRAMACIÓN DE SERVICIOS Y PROCESOS

Punto A) Explica el ejercicio que hicimos del chat utilizando html y php en una cola de mensajes en un archivo txt (lo hicimos en mi servidor)

Explicación de la carpeta 001

Punto E) Contadme el ejercicio de chat con python flask y la diferencia entre crear un servidor stateless con php y crear un servidor que guarda los estados con python (o con nodejs)

Explicación de la carpeta 006

PROGRAMACIÓN MULTIMEDIA

Punto A) Explica la diferencia y las similitudes de la lógica entre los juegos 2D y 3D (3d con threejs)

Similitudes entre Juegos 2D y 3D

Uso de HTML y CSS:
Ambos tipos de juegos pueden utilizar HTML para la estructura básica de la página (como elementos <canvas>) y CSS para estilos como contenedores, botones de interfaz de usuario, y HUD (heads-up display).
Por ejemplo, el sistema de menús, pantallas de inicio, o elementos como barras de vida pueden implementarse con HTML y CSS.

Lógica de Programación con JavaScript:
Tanto en 2D como en 3D, la lógica principal del juego se escribe en JavaScript. Esto incluye la gestión de eventos (teclas presionadas, clics), la detección de colisiones, el movimiento de personajes, y la lógica del juego (como puntuaciones y niveles).
En ambos casos, se sigue el patrón de bucle de juego (game loop) que se repite continuamente para actualizar la posición de los elementos, detectar colisiones y redibujar la escena. Esto se suele hacer utilizando la función requestAnimationFrame().

Sistemas de Coordenadas:
Aunque las coordenadas funcionan de manera diferente (con el eje Z añadido en 3D), tanto los juegos 2D como 3D manejan sistemas de coordenadas para posicionar objetos en el espacio.
Ambos usan la representación de posiciones con vectores (x, y para 2D y x, y, z para 3D).

Colisiones y Física:
En ambos tipos de juegos, puedes implementar colisiones para detectar cuándo dos objetos se tocan y la física para simular gravedad, rebotes, o fricción.

Punto B) En el juego 3D, comenta cómo crearmos objetos a partir de Three.mesh y Three.object

En el archivo 24 hemos creado estos objetos en creamos objetos en pantalla y externos.

Punto C) En el juego, comenta que, para que todo funcione, tiene que existir un objeto raiz llamada Three.scene, dentro del cual estarán enmarcados el resto de objetos

Explicar como funcionan las escenas.

Punto E) A lo largo del desarrollo de juegos 2D y 3D, hemos visto la simulacion de eventos físicos tales como la colisión o la gravedad. Comenta cómo hemos logrado tales efectos

En juegos 2D, se suelen utilizar sistemas de coordenadas que manejan solo dos ejes: x (horizontal) e y (vertical). La física en estos juegos se enfoca en la detección de colisiones entre objetos y la aplicación de fuerzas como la gravedad.

A. Gravedad
Para simular la gravedad, se aplica una constante que acelera continuamente al objeto hacia abajo (en dirección positiva del eje y).
En la práctica, esto se logra aumentando el valor de la posición y del objeto en cada frame del juego.

B. Detección de Colisiones
La detección de colisiones en juegos 2D suele manejarse con colisiones rectangulares (AABB - Axis-Aligned Bounding Box).
La lógica básica verifica si los bordes de dos rectángulos se intersectan.

En juegos 3D, se añade un eje adicional z, lo que complica la física debido a la necesidad de simular efectos en un espacio tridimensional. Aquí, las colisiones y la gravedad se manejan de manera similar, pero con consideraciones extra para el eje z.

A. Gravedad
La gravedad en 3D se implementa de forma similar a la 2D, pero afecta tanto la posición y como otras posibles fuerzas en los ejes x y z.

B. Detección de Colisiones
En 3D, se utilizan Bounding Boxes o Bounding Spheres para detectar colisiones entre objetos.
Three.js proporciona la clase THREE.Box3() para crear una caja delimitadora alrededor de los objetos y la función intersectsBox() para verificar colisiones.

Punto G) En los juegos 2D comenta como hicimos el efecto de desfase parallax, y en el juego 3D comenta que tuvimos que poner una cámara 3D para establecer el punto de vista del propio juego 3D

En el fichero 24, al principio del documento creamos la camara y al final creamos las luces.

Punto I) Cuando hemos desarrollado juegos, hemos tendo la consola abierta, para ver elementos tales como el consumo de CPU, el consumo de RAM, debugear, etc. Comenta todas estas operaciones realizadas

Gestión Empresarial

Punto A) Me explicáis que estamos trabajando con un supuesto modulo CRM y otro de ERP, pero que realmente estamos trabajando sobre un supercontrolador que luego nos servirá para personalizarlo para cada modulo, pero, estamos creando una base fuerte de inicio

Explicar que estamos trabajando con un supercontrolador, que en principio hace de CRM. Y que la parte de ERP todavia no tenemos nada. La idea seria que tenga diferentes funciones nuestro CRM, por ahora es solo el supercontrolador.

Punto B) En base a las tecnologías que estamos usando hasta el momento (stack LAMP), cual sería el proceso de migración previsto en el caso de que finalicemos el desarrollo en nuestro equipo, y lo queramos migrar. Contadme también que hemos tenido que instalar para crear un entorno de desarrollo local (XAMPP)

Punto C) En este apartado contadme las integraciones que hemos hecho hasta el momento. 1.-El controlador de datos que gestiona el servidor (lo creamos en acceso a datos) y el select con buscador (lo creamos en desarrollo de interfaces)

Enseñar supercontrolador con las funcionalidades integradas.

Punto E) Cada vez que realizamos un desarrollo, un cambio, una integración, lo que sea, contadme cómo probamos el software desarrollado y cómo actuamos en el caso de que no funcione. Hablad también de las operaciones de refactorización.

Explicar que cuando integramos cambios o nuevas features al codigo del supercontrolador, debemos adaptarlo correctamente para que funcione con la logica del supercontrolador. Ademas para controlar los errores, utilizamos el console.log para capturar los posibles errores.

Punto F) Comentad por una parte cómo vamos comentando el código y mostrad el esquema de funcionamiento que hice en clase (hacedlo vosotros también - está en la carpeta de documentación)

Explicar los comentarios del codigo, que se van comentando cuando nos aseguramos de que funciona correctamente. Enseñar el esquema de Jose Vicente.