UNIVERSIDAD CATÓLICA BOLIVIANA “SAN PABLO”

PROYECTO SEGUNDO PARCIAL - IOT

Nombre: Beymar Mauricio Coronel Condori
Sergio Juan Ticona Mamani
Brian Joel Zapana Pariguana
Brandon Enrique Silva Pinto

Docente: Germán Jesús Pereira Muñoz

Fecha de entrega: 11/ 11 / 2024

La Paz – Bolivia

1. INTRODUCCION

1.1 Descripción general del proyecto y su contexto.

Este proyecto está diseñado para ayudar a personas mayores que han experimentado una pérdida progresiva de movilidad en las manos y requieren rehabilitación física. Para ello, hemos desarrollado un sistema interactivo que utiliza cuatro joysticks y luces LED, pensado para guiar a los usuarios a través de ejercicios y actividades lúdicas. Mediante juegos orientados a estimular el movimiento de los dedos de forma atractiva y entretenida, el sistema busca aumentar la motivación de los usuarios, mejorando la adherencia a los ejercicios y optimizando la experiencia de recuperación.

El sistema incorpora una base de datos que permite registrar información relevante de cada usuario, así como sus sesiones de rehabilitación, progreso y lecturas de movimientos específicos. Esta estructura facilita la recolección de datos sobre movimientos, tiempos de reacción y avances en fuerza y precisión, lo cual permite un seguimiento personalizado y continuo del progreso de cada usuario. Gracias a esta herramienta, los profesionales de la salud pueden ajustar los tratamientos de manera eficaz, proporcionando un apoyo integral y motivador para mejorar la calidad de vida y la movilidad de los adultos mayores.

1.2 Objetivos del proyecto, tanto generales como específicos.

1.2.1 Objetivo General

Desarrollar un sistema interactivo de rehabilitación para personas mayores que, mediante ejercicios con joysticks y luces LED, mejore la movilidad y facilite el monitoreo personalizado de su progreso.

1.2.2 Objetivos Específicos

- Diseñar una base de datos que estructure y garantice la integridad de los datos de usuarios, sesiones y progreso de rehabilitación.
- Crear un script en PHP para la inserción y consulta de datos en la base de datos, con validaciones y manejo de errores.
- Implementar un script en MicroPython que permita al microcontrolador ESP32 insertar datos de manera confiable.
- Desarrollar un script en Python para operaciones ABM (Alta, Baja, Modificación) en la base de datos mediante una interfaz interactiva.
- Construir un dashboard en Python para visualizar el progreso de rehabilitación en tiempo real, con gráficos interactivos y actualizaciones automáticas.

1.3 Importancia del proyecto

Este proyecto es importante por su capacidad de mejorar la calidad de vida de las personas mayores que, con el tiempo, han perdido movilidad. La falta de movilidad reduce su independencia y afecta su bienestar físico y emocional. Este sistema interactivo ofrece una herramienta accesible y motivadora para la rehabilitación, combinando ejercicios lúdicos con tecnología de retroalimentación visual que facilita la adherencia a una rutina constante y entretenida.

Además, el sistema recoge datos de movimiento y progreso, proporcionando a los terapeutas información clave para personalizar el tratamiento y adaptar los ejercicios a las necesidades específicas de cada persona. Esto no solo optimiza el proceso de rehabilitación, sino que permite un monitoreo detallado de la evolución del paciente, mejorando la efectividad de la terapia y favoreciendo una recuperación más completa y satisfactoria.

2. DISEÑO DE LA BASE DE DATOS

2.1 Descripción de la estructura

La base de datos que se maneja es una relacional, la base de datos tiene como nombre: “db_ejercicios”.
(Anexo 1, Figura 1) Se tienen 4 tablas las cuales son:

- **Tabla usuarios :** Tabla usada para almacenar datos del usuario que realiza la prueba.
- **Tabla de sesiones :** Tabla usada para almacenar datos de la sesión de práctica realizada.
- **Tabla lecturas:** Tabla usada para la lectura de datos en base a la sesión.
- **Tabla progreso:** Tabla usada para medir el progreso del paciente según el usuario.

2.2 Descripción detallada

(Anexo 1, Figura 1) En todas las tablas se incluye un campo denominado “id_(db_ejercicios)”, que representa el código único asignado automáticamente para cada fila. Este campo actúa como clave primaria o "Primary Key" (PK), asegurando unicidad y evitando valores duplicados o nulos en la misma tabla. Los campos están configurados como “serial”, de modo que el servidor asigna automáticamente un número entero secuencial iniciando en 1 para cada nuevo registro.

- **Tabla usuarios**
    - **idUsuario**: Campo de tipo serial, clave primaria de esta tabla, que representa un identificador único asignado a cada usuario en el sistema. Es generado automáticamente y no permite valores nulos.

    - **Nombre**: Campo de tipo VARCHAR(50). Almacena el nombre del usuario registrado. Es un campo obligatorio y no puede ser nulo.
    - **Edad**: Campo de tipo INT. Representa la edad del usuario. No permite valores nulos, y debe contener un valor entero positivo.
    - **Genero**: Campo de tipo VARCHAR(50). Almacena el género del usuario registrado. Este campo es obligatorio y no permite valores nulos.
    - **Fecha_registro**: Campo de tipo DATE. Registra la fecha en la que el usuario se inscribió en el sistema, permitiendo el seguimiento temporal de cada cuenta. Este campo es obligatorio y no permite valores nulos.
    - **Comentarios_medicos**: Campo de tipo VARCHAR(200). Sirve para almacenar comentarios médicos relevantes del usuario, que pueden ser utilizados para referencias de salud en las sesiones de ejercicio. Es obligatorio y no permite valores nulos.
    -
- **Tabla de sesiones**
    - **idSesion:** Campo de tipo serial, clave primaria de esta tabla, que identifica de forma única cada sesión de ejercicio registrada. Es generado automáticamente y no permite valores nulos.
    - **idUsuario:** Campo de tipo INT. Representa el código único asociado al usuario que realizó la sesión. Es una clave foránea que se refiere al idUsuario en la tabla Usuarios. Si este campo es nulo, puede significar una falta de asociación en la toma de datos o un error en la conexión.
    - **Fecha_sesion:** Campo de tipo DATE. Registra la fecha en la que se realizó la sesión de ejercicio. Este campo es obligatorio y no permite valores nulos.
    - **Duracion:** Campo de tipo INT. Almacena la duración de la sesión en minutos. Es obligatorio y no permite valores nulos, aunque puede ser igual a 0 en situaciones de prueba o sesiones canceladas.
    - **Tipo_ejercicio:** Campo de tipo VARCHAR(50). Define el tipo de ejercicio realizado durante la sesión, lo cual facilita la clasificación de las sesiones entre diferentes categorías de práctica. Este campo no permite valores nulos.
- **Tabla lecturas**
    - **idLectura:** Campo de tipo serial, clave primaria de esta tabla, que identifica de forma única cada registro de lecturas del joystick durante una sesión. Es generado automáticamente y no permite valores nulos.
    - **idSesion:** Campo de tipo INT. Representa el código único de la sesión de práctica asociada, conectándose mediante una clave foránea a la tabla Sesiones. Si es nulo, puede haber habido un error en la captura de datos.

    - **Joystick_numero:** Campo de tipo INT. Identifica el número del joystick utilizado para capturar las lecturas. Este campo es obligatorio y no permite valores nulos.
    - **Eje_x y Eje_y:** Campos de tipo INT. Almacenan las posiciones de los ejes x e y en cada lectura realizada, midiendo así el desplazamiento en ambas direcciones. Ninguno de estos campos permite valores nulos.
    - **Nivel_x y Nivel_y:** Campos de tipo INT. Indican los niveles de fuerza aplicados en cada eje, registrando la intensidad de presión o movimiento. Estos campos son obligatorios y no pueden ser nulos.
    - **Boton:** Campo de tipo BOOLEAN. Indica si el botón del joystick fue presionado (TRUE) o no (FALSE) durante la sesión. Este campo es obligatorio y no permite valores nulos.
    - **Direccion:** Campo de tipo VARCHAR(50). Almacena la dirección del movimiento registrada (por ejemplo, "izquierda", "derecha"). Este campo no puede ser nulo y proporciona contexto adicional sobre el tipo de movimiento.
    - **Diagonal:** Campo de tipo VARCHAR(150). Describe si el movimiento tuvo componentes diagonales y registra los detalles correspondientes, si los hay.
- **Tabla progreso**
    - **idProgreso**: Campo de tipo serial, clave primaria de esta tabla, que identifica de manera única cada registro de progreso de un usuario. Es generado automáticamente y no permite valores nulos.
    - **idUsuario**: Campo de tipo INT. Representa el identificador único del usuario, conectándose mediante una clave foránea a la tabla Usuarios para asegurar la relación con el progreso individual. No puede ser nulo.
    - **Fecha_calculo**: Campo de tipo DATE. Registra la fecha en la que se calcula el progreso de un usuario en una sesión o conjunto de sesiones. Este campo no permite valores nulos.
    - **Fuerza_promedio**: Campo de tipo FLOAT. Almacena la fuerza promedio registrada para el usuario, calculada en función de los datos de la sesión. No puede ser nulo, aunque puede ser 0 en caso de sesiones de práctica iniciales.
    - **Movilidad_horizontal**: Campo de tipo INT. Representa la medida de movilidad horizontal del usuario registrada en una sesión específica. No permite valores nulos.
    - **Movilidad_vertical**: Campo de tipo INT. Mide la movilidad vertical del usuario. Este campo es obligatorio y no permite valores nulos.
    - **Precision_diagonal**: Campo de tipo INT. Registra la precisión diagonal alcanzada por el usuario durante la práctica. No permite valores nulos.

- **Mejora_fuerza**: Campo de tipo FLOAT. Almacena el porcentaje de mejora en fuerza respecto a sesiones anteriores, útil para seguimiento de progreso a lo largo del tiempo. Puede ser 0 en sesiones iniciales o en caso de falta de mejoras.

2.3 Diagrama de entidad-relación

El diagrama Entidad-Relación de la base de datos bd_ejercicios puede ser encontrado en el Anexo 1, Figura 1.

3. DESARROLLO DEL SCRIPT EN PHP

3.1 Inserción de datos

El script PHP para la inserción de datos comienza estableciendo una conexión con la base de datos mediante los parámetros de conexión (hostname, usuario, contraseña y nombre de la base de datos). Una vez establecida la conexión, el script verifica si la conexión es exitosa; en caso contrario, muestra un mensaje de error y detiene la ejecución del script.

Para la inserción, se define una función que utiliza consultas SQL preparadas. Este enfoque mejora la seguridad al proteger contra inyecciones SQL y permite insertar valores específicos, como joystick, movimiento y tiempo de reacción, en la tabla datos_movilidad. Cada uno de estos valores es procesado antes de la inserción para asegurar su integridad. Si ocurre un error durante la inserción, el script captura el error y muestra un mensaje adecuado.

3.2 Lectura de datos

La lectura de datos se realiza a través de una función que ejecuta una consulta SELECT para obtener todos los registros de la tabla datos_movilidad. Si la consulta obtiene resultados, el script recorre cada registro y muestra la información correspondiente (joystick, movimiento y tiempo de reacción). Si no hay registros, el script muestra un mensaje indicando que no existen datos disponibles.

3.3 Validaciones y Seguridad

El script PHP incluye validaciones para asegurar que los datos sean válidos y estén correctamente formateados antes de la inserción. Esto se logra mediante funciones de filtrado que verifican y sanitizan los valores de entrada, como la sanitización de cadenas y la validación de números. Además, se aplican medidas de seguridad, como el uso de consultas SQL preparadas, para prevenir ataques de inyección SQL. El script también cuenta con un sistema de manejo de errores que facilita la identificación y resolución de problemas en tiempo real.

4. DESARROLLO DEL SCRIPT EN MICROPYTHON

4.1 Conexión y configuración

El microcontrolador se conecta a la red Wi-Fi utilizando un archivo de configuración secrets.py, que contiene las credenciales necesarias (SSID y contraseña). Se emplea una función wifi_init() para establecer la conexión, que gestiona la configuración del dispositivo de acuerdo con el modelo utilizado (como Raspberry Pi Pico W, ESP32, ESP8266, entre otros). La función se asegura de que los pines y la configuración sean correctos según el modelo de placa.

4.2 Inserción de datos

Una vez establecida la conexión con la red Wi-Fi, el script toma los valores del joystick (movimientos hacia la izquierda, derecha, arriba, abajo y centro), la duración del uso, junto con el nombre y correo electrónico del usuario. Estos datos se envían a un servidor mediante una solicitud HTTP POST.

El servidor espera los datos en formato JSON, lo que facilita la integración con aplicaciones web. Los datos enviados incluyen información sobre los movimientos del joystick y la duración del uso, organizados bajo las claves definidas en el diccionario data.

4.3 Manejo de excepciones y reconexión

Para asegurar la estabilidad de la conexión, el script está diseñado para manejar posibles errores de red. En caso de que el microcontrolador pierda la conexión a la red Wi-Fi o al servidor, el sistema intentará reconectar automáticamente. Esto se logra mediante un bloque try-except, que captura los errores de conexión y activa la reconexión utilizando la función reconectar_wifi(). Esta función reintenta la conexión a la red Wi-Fi después de un breve retraso.

5. DESARROLLO DEL SCRIPT EN PYTHON PARA ABM

5.1 Descripción general de las operaciones ABM

Este script en Python gestiona las operaciones ABM (Alta, Baja, Modificación) sobre la tabla de usuarios en una base de datos MySQL. Las operaciones son las siguientes:

- **Alta (Creación):** Permite agregar un nuevo usuario proporcionando datos como nombre, correo electrónico y contraseña. Esta operación inserta un nuevo registro en la base de datos.
- **Baja (Eliminación)**: Permite eliminar un usuario seleccionando su ID desde la lista. La eliminación es permanente y requiere confirmación antes de proceder.
- **Modificación:** Permite actualizar los datos de un usuario existente (como nombre, correo y contraseña). Si el usuario ya existe, se actualiza; si no, se crea un nuevo registro.

5.2 Interfaz de usuario

El formulario de usuario permite ingresar o modificar los datos de un usuario. Este formulario incluye:

- **ID de Usuario:** Solo lectura cuando se selecciona un usuario existente.
- **Nombre:** Campo para ingresar el nombre del usuario.
- **Correo:** Campo para ingresar el correo electrónico.
- **Contraseña:** Campo para ingresar la contraseña, que se oculta al escribir.

Botones de Acción: Los botones permiten ejecutar las operaciones de ABM:

- **Nuevo Usuario:** Limpia el formulario para crear un nuevo usuario.
- **Guardar Usuario:** Guarda o actualiza el usuario, dependiendo de si el ID está vacío o no.
- **Eliminar Usuario:** Elimina el usuario seleccionado.
- **Limpiar Usuario:** Borra los datos en el formulario.

Tabla de Usuarios: La tabla muestra una lista de usuarios con los siguientes campos:

- **ID:** Identificador único del usuario.
- **Nombre:** Nombre del usuario.
- **Correo:** Correo electrónico del usuario.
- **Fecha Registro:** Fecha y hora de creación del usuario.

Lógica detrás de cada operación de la interfaz:

- **Nuevo Usuario:** Limpia los campos del formulario para permitir al usuario ingresar los datos de un nuevo usuario.
- **Guardar Usuario:** Si el ID de Usuario está vacío, se considera que se está creando un nuevo usuario. Si el ID de Usuario tiene valor, el sistema actualiza los datos del usuario correspondiente.La información de nombre, correo y contraseña se guarda en la base de datos, y el formulario se limpia tras el éxito de la operación.
- **Eliminar Usuario:** El usuario debe seleccionar un ID de usuario de la lista antes de eliminarlo. Al confirmar la eliminación, se procede a eliminar el usuario de la base de datos y se actualiza la tabla de usuarios.
- **Limpiar Usuario:** Borra todos los valores en el formulario, permitiendo al usuario empezar de nuevo.

5.3 Manejo de errores y validaciones

El manejo de errores en el código se realiza principalmente en las siguientes situaciones:

- **Conexión a la Base de Datos:** Si hay un error de conexión a MySQL, el programa muestra un mensaje de error indicando que no se pudo conectar a la base de datos.
- **Operaciones de Base de Datos:** Durante las operaciones de inserción, actualización o eliminación, cualquier error en la consulta SQL es capturado y se muestra un mensaje informativo al usuario.
- **Errores Generales:** Si ocurre un error inesperado, como al intentar realizar una operación sin la selección de un usuario o con datos incompletos, el sistema muestra mensajes de advertencia o confirmación.

Validaciones:

- **Validación de Campos Vacíos:** Se verifica que los campos de nombre, correo y contraseña no estén vacíos antes de crear o actualizar un usuario. Si algún campo está vacío, se muestra una advertencia.
- **Validación de Selección de Usuario para Eliminación:** Se asegura que el usuario haya seleccionado un registro antes de intentar eliminarlo. Si no es así, se muestra un mensaje solicitando la selección.
- **Confirmación de Eliminación:** Al intentar eliminar un usuario, se muestra una ventana de confirmación para evitar eliminaciones accidentales.
- **Manejo de la Selección de la Tabla:** Se valida que un usuario esté seleccionado en la tabla antes de permitir la edición o eliminación. Si no hay selección, se muestra una advertencia.

6. DESARROLLO DEL DASHBOARD EN PYTHON PARA VISUALIZACIÓN EN TIEMPO REAL

6.1 Diseño del Dashboard

El dashboard fue diseñado en Python utilizando herramientas de visualización como matplotlib, que permiten representar los datos de movilidad en gráficos actualizados automáticamente cada pocos segundos. El diseño incluye gráficos lineales que muestran los movimientos y tiempos de reacción en función del tiempo, permitiendo una visualización clara y precisa de los datos recolectados. Además, se utilizan características interactivas que permiten una mejor comprensión de los datos.

6.2 Explicación del Código del Dashboard

El código del dashboard se conecta a la base de datos y extrae los datos almacenados. Los datos son visualizados en tiempo real mediante gráficos que se actualizan de manera continua. El script realiza consultas periódicas a la base de datos para obtener los datos más recientes y actualizar los gráficos automáticamente. Esta operación se lleva a cabo sin necesidad de intervención manual, garantizando que los usuarios siempre tengan acceso a información actualizada.

6.3 Interactividad y Actualización

La interactividad del dashboard permite a los usuarios aplicar filtros y seleccionar rangos específicos de datos para su análisis. Esta característica facilita un análisis más detallado y personalizado. La actualización automática se asegura mediante una pausa configurada en el script, que permite obtener los datos más recientes sin interrumpir la visualización en tiempo real. Además, el dashboard cuenta con un sistema que maneja posibles desconexiones o problemas de conexión, garantizando una experiencia de usuario estable y continua.

7. CONCLUSIÓN Y LECCIONES APRENDIDAS

7.1 Resumen de los logros y conclusiones del proyecto

El proyecto permitió desarrollar un sistema completo para gestionar usuarios y visualizar datos de movilidad en tiempo real. Se implementaron operaciones ABM seguras y un dashboard interactivo que se actualiza automáticamente con datos de la base de datos. El sistema garantizó la integridad de los datos y una experiencia de usuario estable.

7.2 Descripción de los principales desafíos enfrentados y cómo se resolvieron.

Uno de los principales desafíos fue el armado del circuito, que implicó integrar correctamente el microcontrolador, joystick y sensores de movimiento. Para solucionarlo, se realizaron pruebas iterativas y se verificó la conectividad Wi-Fi.

Otro desafío fue la extracción de datos, que incluía movimientos del joystick, duración de uso y datos del usuario. Estos datos se enviaron en tiempo real al servidor en formato JSON. Se implementaron validaciones para asegurar que solo se enviaran datos correctos y evitar errores.

7.3 Reflexión sobre el aprendizaje obtenido a través de la adaptación de este proyecto.

El proyecto permitió mejorar habilidades técnicas en bases de datos, visualización en tiempo real y manejo de errores. También se aprendió sobre la importancia de la experiencia del usuario y la gestión de proyectos, enfocándose en soluciones eficientes y seguras.

8. ANEXOS

8.1 Anexo 1: Diagrama entidad - relación de la base de datos

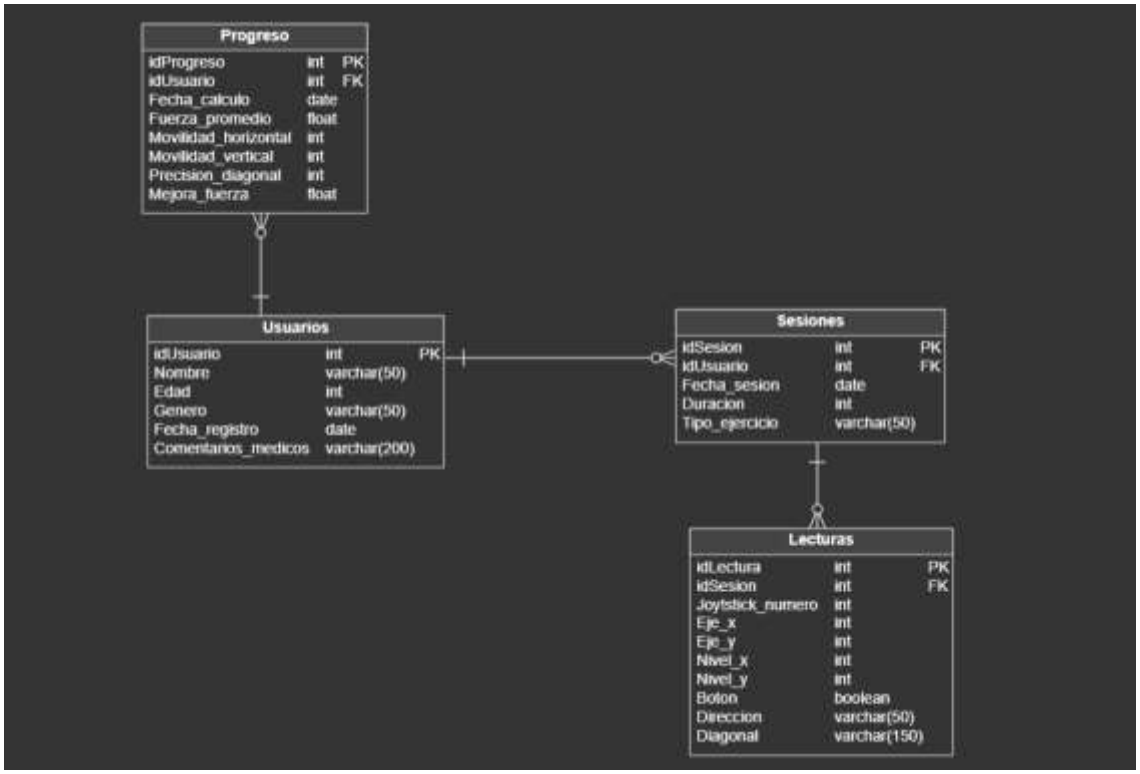

Figura 1.

8.2 Anexo 2: Código en PHP

- **8.2.1** https://github.com/Serghio07/2do-Parcial/blob/main/InsercionDatos.php

8.3 Anexo 3: Código en Python

- **8.3.1** https://github.com/Serghio07/2do-Parcial/blob/main/ABM.py
- **8.3.2** https://github.com/Serghio07/2do-Parcial/blob/main/dashboard.py
- **8.3.3** https://github.com/Serghio07/2do-Parcial/blob/main/InsercionDatos.py

8.4 Anexo 4: Diseño dashboard

- **8.4.1** https://github.com/Serghio07/2do-Parcial/blob/main/templates/index.html

Figura 2.

8.5 Anexo 5: Repositorio GitHub

- **8.5.1** https://github.com/Serghio07/2do-Parcial.git