APLICACIÓN DE GESTIÓN DE MÁQUINAS DE VENDING

MODULO : M2

MÓDULO	0613. Desarrollo Web en Entorno Servidor
GRUPO	M2
INTEGRANTES	Javier Pastor Matesanz, Erlin Francisco Sapeg Soriano, Javier Arruabarrena Sabroso y Kevin Alexander Zurita Ojeda
FECHA	21/11/2024

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

No se encuentran elementos de tabla de ilustraciones.

1. INTRODUCCIÓN

El módulo 2 (M2) del sistema de gestión de máquinas de vending está diseñado para manejar las operaciones relacionadas con la fabricación y gestión de las máquinas y las ubicaciones en las que operan. Este módulo se centra en:

- **El mantenimiento integral de los datos de las máquinas de vending.**
- **La gestión de ubicaciones, incluyendo su alta, baja y modificación.**
- **La gestión de máquinas, incluyendo su alta, baja y modificación.**
- **La asignación de máquinas de vending a ubicaciones existentes.**

Este módulo es una parte fundamental del sistema, ya que asegura que las máquinas estén correctamente distribuidas y asociadas a ubicaciones dentro de la red de clientes.

2. ALCANCE DEL PROYECTO

Objetivo General

Proveer una solución eficiente para el manejo del parque de máquinas de vending y su asociación a ubicaciones, asegurando que:

- **Las máquinas estén siempre asignadas a ubicaciones válidas.**
- **Se puedan gestionar tanto máquinas como ubicaciones desde una interfaz web.**

Límites del Módulo

- **Solo permite la asignación de máquinas a ubicaciones previamente registradas.**
- **No incluye funcionalidades de reposición o monitoreo de stock (gestionadas por el módulo M3).**
- **El cambio de estado a "averiada" desasigna automáticamente a la máquina de su ubicación actual.**

3. FUNCIONALIDADES DEL MÓDULO

Alta, Baja y Modificación de Máquinas

- **Permite registrar nuevas máquinas con atributos como número de serie, modelo y tipo.**
- **Posibilita actualizar los detalles de las máquinas existentes.**
- **Incluye la funcionalidad para dar de baja máquinas que ya no estén en uso.**

Gestión de Ubicaciones

- **Permite agregar, modificar y eliminar ubicaciones dentro de la red.**
- **Cada ubicación está asociada a un cliente y debe contener una dirección válida.**

Asignación de Máquinas a Ubicaciones

- **Permite asignar máquinas del taller (almacén) a ubicaciones disponibles.**
- **Asegura que las máquinas solo se asignen a ubicaciones válidas y activas.**

4. DISEÑO DE APLICACIÓN

4.1. TECNOLOGÍAS UTILIZADAS

- **Backend: PHP.**
- **Base de Datos: MySQL.**
- **Frontend: HTML/CSS con generación dinámica desde PHP.**

4.2. ARQUITECTURA DE APLICACIÓN

4.2.1. CLIENTE/SERVIDOR

El sistema utiliza un modelo cliente-servidor en el que:

1. **El cliente (usuario final) interactúa con la interfaz web.**
2. **El servidor PHP procesa las solicitudes del cliente y se comunica con la base de datos MySQL para recuperar o modificar datos.**

4.2.2. ESTRUCTURA DE DIRECTORIOS

Directorios:

- `/clases`**: Contiene funciones reutilizables para la conexión a la base de datos y generación de menús.**
- `/css`**: Archivos CSS para el diseño de la interfaz.**
- `/m2`**: Código PHP específico del módulo 2.**

4.2.3. BASE DE DATOS

Tabla máquina

Registra la información de las máquinas de vending y su estado dentro del sistema.

- **idmaquina: INT, clave primaria autoincremental.**
- **numserie: VARCHAR(32), identificador único de la máquina.**
- **idestado: INT, clave foránea a la tabla estado. Representa el estado actual de la máquina.**
- **idubicacion: INT, clave foránea a la tabla ubicacion. Indica dónde se encuentra la máquina.**
- **capacidad: INT, número de artículos distintos que puede contener la máquina.**
- **stockmax: INT, capacidad máxima de unidades por artículo.**
- **modelo: VARCHAR(32), modelo de la máquina (por ejemplo: STAR30, STAR24).**
- **foto: VARCHAR(128), ruta relativa al archivo de imagen de la máquina.**

Tabla ubicación

Contiene los datos de las ubicaciones asociadas a las máquinas.

- **idubicacion: INT, clave primaria autoincremental.**
- **dir: VARCHAR(64), dirección de la ubicación (calle, número, código postal, provincia).**
- **cliente: VARCHAR(64), nombre del cliente asociado a la ubicación.**

Relación con la tabla máquina:

Las máquinas se asocian a una ubicación a través de la clave idubicacion.

Tabla estado

Define los estados posibles para las máquinas.

- **idestado: INT, clave primaria autoincremental.**
- **descripción: VARCHAR(64), descripción del estado (por ejemplo: desactivada, en servicio, averiada).**

5. DISEÑO DEL MÓDULO

El módulo de fabricación se encarga de gestionar máquinas de vending y sus ubicaciones. Esto incluye:

- **Alta, baja y modificación de máquinas.**
- **Alta y modificación de ubicaciones.**
- **Asignación de máquinas a las ubicaciones existentes.**
- **Reasignación de máquinas a ubicaciones genéricas en caso de eliminación de una ubicación.**

Requisitos Funcionales Principales:

1. **Gestión integral de las máquinas de vending (CRUD completo).**
2. **Gestión de ubicaciones, asociándolas con clientes y direcciones específicas.**
3. **Interacción sencilla a través de formularios dinámicos y validación de datos.**
4. **Automatización de reasignación de máquinas tras la eliminación de una ubicación.**
5. **Funcionalidad específica para asociar máquinas del taller con ubicaciones existentes.**

5.1. DIAGRAMA DE CLASES

El diagrama de clases se centra en las entidades principales del módulo y sus interacciones:

- **Máquina:** `idmaquina`, `numserie`, `idestado`, `idubicacion`, `modelo`, `foto`
- **Ubicación:** idubicacion, cliente, dir.
- **Relaciones :**

  Máquina tiene una relación de *uno a muchos* con Ubicación.

  Las modificaciones en Ubicación afectan directamente al atributo idubicacion de Máquina.

5.2. INTERFAZ DE USUARIO

Panel Central (`fabricación.php`):

- Dos tablas principales: máquinas y ubicaciones.
- Posibilidad de filtrar por modelo, cliente o ciudad.
- Selección de filas para redirigir a otras interfaces.

Formularios Individuales:

- Formularios dinámicos para alta, edición o eliminación de máquinas y ubicaciones.
- Campos validados con restricciones específicas (ej.: rango para código postal y número de portal).

Diseño Responsive:

- Utilización de CSS para ajustar los elementos visuales en dispositivos pequeños.
- Tablas adaptadas para evitar desbordamientos horizontales.

5.3. ESTRUCTURA DE DIRECTORIOS

/top vending

- /css
  - máquinas.css
  - modificar ubicaciones.css
  - ubicaciones.css
- /m2
  - anyadir_maquina.php
  - anyadir_ubicación.php
  - fabricación.php
  - máquinas.php
  - modificar ubicaciones.php
  - ubicaciones.php
- /resources
  - default.jpg
  - imágenes adicionales cargadas por los usuarios

5.4. ESTRUCTURA DEL CÓDIGO

Cada script sigue un diseño dividido en las siguientes secciones:

1. **Inclusión de Dependencias:**
   - **Conexión a la base de datos y funciones auxiliares.**
2. **Definición de Variables y Procesamiento de Datos:**
   - **Validación de entradas (formularios, cookies, etc.).**
3. **Consultas a la Base de Datos:**
   - **CRUD sobre tablas principales (máquina y ubicación).**
4. **Ejecución de la Lógica del Módulo:**
   - **Actualización, inserción o eliminación según la acción solicitada.**
5. **Renderizado de la Interfaz:**
   - **Generación de tablas o formularios dinámicos con PHP y HTML.**

5.5. INTERFACES CON OTROS MÓDULOS

El módulo de fabricación interactúa directamente con otros módulos:

- **Módulo de Gestión de Perfiles (M0):**
  - **Utiliza las funciones de sesión y menú dinámico generadas en el módulo de perfiles.**
- **Módulo de Gestión de Suministros (M3):**
  - **La reasignación de máquinas a ubicaciones genéricas (ID 1) asegura que el módulo de suministros pueda continuar trabajando sin interrupciones.**
- **Módulo de Gestión de Incidencias (M4):**
  - **La modificación de estados en máquinas puede influir en las incidencias registradas.**

6. IMPLEMENTACIÓN DEL MÓDULO

6.1.1. DESCRIPCIÓN GENERAL

El módulo está diseñado para ser modular, manejable y centrado en el usuario. Cada script representa una funcionalidad específica, asegurando separación de responsabilidades.

6.1.1.1. REPOSITORIOS PARA DESARROLLO

El código se almacena en un repositorio GitHub privado, con una rama específica para este módulo (`m2_fabricacion`).

6.1.1.2. DESPLIEGUE DEL MÓDULO

El código se almacena en un repositorio GitHub privado, con una rama específica para este módulo (m2_fabricación).

1. Requisitos:
   - Servidor web Apache con PHP.
   - Base de datos MySQL con el esquema proporcionado.
2. Pasos:
   - Clonar el repositorio en el directorio del servidor web.
   - Configurar la conexión a la base de datos en el archivo de configuración central.
   - Verificar las dependencias de imágenes en `/resources`.

6.1.2. DOCUMENTACIÓN DEL CÓDIGO

Cada script incluye comentarios detallados que explican las funciones y las consultas SQL utilizadas. Las variables se nombran siguiendo un estándar coherente.

6.1.2.1. ORGANIZACIÓN GENERAL

El módulo está organizado para que los cambios en los estados de las máquinas y las ubicaciones se reflejan automáticamente en la interfaz y base de datos, asegurando consistencia.

6.1.2.2. DESCRIPCIÓN BLOQUE/FUNCIÓN 1

Ejemplo Ubicaciones:

Propósito: Mover máquinas del taller a ubicaciones operativas.

Descripción: Lista las máquinas del taller (ubicación ID 1) y permite asignarlas a ubicaciones existentes mediante un selector.

Impacto: Actualiza el atributo `idubicacion` en la tabla `máquina`.

Obtención de Datos de Máquinas y Ubicaciones:

- Ejecuta una consulta SQL para obtener las máquinas en el taller (ubicación `idubicacion = 1`).
- También obtiene las ubicaciones existentes en la base de datos, excluyendo el taller.

```php
// Consulta SQL para obtener las máquinas en el taller (idubicacion = 1)
$sql = "
    SELECT m.numserie
    FROM maquina m
    JOIN ubicacion u ON m.idubicacion = u.idubicacion
    WHERE u.idubicacion = 1";
$stmt = $dbh->query($sql);
// Ejecuta la consulta SQL a través de la conexión y almacena el resultado en '$stmt'.

$maquinas = $stmt->fetchAll(PDO::FETCH_ASSOC);
// Recupera todos los resultados de la consulta como un arreglo asociativo y los almacena en la variable '$maquinas'.
```

Manejo de Solicitudes POST:

- **Si se envía un formulario (POST), verifica que se haya seleccionado una máquina y una nueva ubicación.**
- **Recupera el número de serie de la máquina y el ID de la nueva ubicación desde el formulario.**

```php
// Verifica si la solicitud es POST y si se están enviando los parámetros necesarios para asignar una ubicación
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['asignarUbicacion'], $_POST['numserie'], $_POST['idubicacion'])) {
    $numserie = $_POST['numserie'];
    // Recupera el número de serie de la máquina enviada por POST.

    $idubicacion = $_POST['idubicacion'];
    // Recupera el ID de la ubicación seleccionada de la lista desplegable enviada por POST.
```

Actualización de la Ubicación de la Máquina:

- **Si la solicitud es válida, actualiza la ubicación de la máquina en la base de datos, cambiando el campo `idubicacion` según la nueva ubicación seleccionada.**
- **Utiliza bloques `try-catch` para capturar y manejar errores en la ejecución de las consultas SQL. Si ocurre un error, muestra un mensaje y registra el error en el log.**

```php
try {
    // Inicia otro bloque try-catch para manejar errores de la actualización en la base de datos.

    $updateSql = "UPDATE maquina SET idubicacion = :idubicacion WHERE numserie = :numserie";
    // Consulta SQL para actualizar la ubicación de la máquina en la base de datos.

    $stmt = $dbh->prepare(query: $updateSql);
    // Prepara la consulta para su ejecución.

    $stmt->bindParam(param: ':idubicacion', var: &$idubicacion);
    // Asocia el valor del parámetro 'idubicacion' de la consulta con la variable '$idubicacion'.

    $stmt->bindParam(param: ':numserie', var: &$numserie);
    // Asocia el valor del parámetro 'numserie' de la consulta con la variable '$numserie'.

    $stmt->execute();
    // Ejecuta la consulta para actualizar la base de datos.

    RegistrarLog(categoría: "Data", accion: "Máquina actualizada");
    // Redirige a la página 'ubicaciones.php' después de realizar la actualización.
    header(header: "Location: ubicaciones.php");
    exit;
} catch (Exception $e) {
    // Si ocurre un error al ejecutar la actualización, muestra un mensaje de error.
    echo "<p style='color: red;'>Error al asignar la ubicación: " . $e->getMessage() . "</p>";
    RegistrarLog(categoría: "Error", accion: "Error al asignar la ubicación");
}
```

Redirección:

- **Después de actualizar la ubicación de la máquina, redirige al usuario a la página `ubicaciones.php` para reflejar los cambios.**

```php
} catch (Exception $e) {
    // Si ocurre un error al obtener las máquinas o ubicaciones, muestra un mensaje de error general.
    echo "<p>Error: " . $e->getMessage() . "</p>";
    RegistrarLog(categoría: "Error", accion: "Error en la consulta a la bbdd");
}
?>
```

Este código asegura que la máquina se asigne correctamente a una nueva ubicación y maneja errores de manera controlada durante el proceso.

6.1.2.3. DESCRIPCIÓN BLOQUE/FUNCIÓN 2

Ejemplo Añadir Ubicación:

Propósito: Crear una nueva ubicación con los inputs del usuario.

Descripción: Lista los campos a rellenar para crear una nueva ubicación.

Verificación de la Solicitud POST:

- **Verifica si el formulario ha sido enviado mediante el método `POST`.**

```php
<?php
try {
    // Verificar si la solicitud es de tipo POST (cuando se envía el formulario)
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

Sanitización de los Datos:

- **Sanitize los datos del formulario (`cliente`, `calle`, `numportal`, `codigo_postal`, `provincia`) para evitar inyecciones de código malicioso utilizando `htmlspecialchars`.**

```php
// Sanitizar los datos recibidos del formulario para evitar inyecciones de código
$cliente = htmlspecialchars(string: $_POST['cliente']);
$calle = htmlspecialchars(string: $_POST['calle']);
$numportal = htmlspecialchars(string: $_POST['numportal']);
$codigoPostal = htmlspecialchars(string: $_POST['codigo_postal']);
$provincia = htmlspecialchars(string: $_POST['provincia']);
```

Validación de Campos:

- **Asegura que todos los campos requeridos no estén vacíos antes de continuar con el procesamiento.**

```php
// Validar que todos los campos estén completos antes de procesarlos
if (!empty($calle) && !empty($numportal) && !empty($codigoPostal) && !empty($provincia) && !empty($cliente)) {
    // Formatear la dirección completa como un string delimitado por punto y coma
    $direccion = "$calle;$numportal;$codigoPostal;$provincia";

    // Crear la consulta SQL para insertar una nueva ubicación en la base de datos
    $sql = "INSERT INTO ubicacion (cliente, dir) VALUES (:cliente, :dir)";

    // Preparar la consulta para ejecutar la inserción en la base de datos
    $stmt = $dbh->prepare(query: $sql);
    // Vincular los parámetros de la consulta con las variables sanitizadas
    $stmt->bindParam(param: ':cliente', var: &$cliente);
    $stmt->bindParam(param: ':dir', var: &$direccion);
```

Inserción en la Base de Datos:

- **Prepara una consulta SQL para insertar una nueva ubicación con los datos del formulario en la tabla `ubicación`.**
- **Ejecuta la consulta usando la función `execute()`.**

```php
try {
    // Ejecutar la consulta de inserción
    $stmt->execute();
    // Mostrar un mensaje en la consola indicando que la inserción fue exitosa
    echo "<script>console.log('Nueva ubicación insertada correctamente');</script>";
    // Redireccionar al usuario a otra página después de la inserción
    header(header: "Location: ./fabricacion.php");
    // Detener la ejecución del script después de la redirección
    exit;
```

Manejo de Errores en la Inserción:

- **Si la inserción en la base de datos es exitosa, redirige al usuario a la página `fabricacion.php`.**
- **Si ocurre un error al ejecutar la consulta, se captura mediante el bloque `catch` y muestra un mensaje de error en la página, además de registrar el error en el log.**

```
    } catch (PDOException $e) {
        // Si ocurre un error al ejecutar la consulta, mostrar el mensaje de error
        echo "<p style='color: red;'>Error al insertar la ubicación: " . htmlspecialchars(string: $e->getMessage()) . "</p>";
        RegistrarLog(categoría: "Error", accion: "Error en la consulta");
    }
} else {
    // Si algún campo está vacío, mostrar un mensaje de error
    echo "<p style='color: red;'>Error: Todos los campos deben estar completos.</p>";
}
```

Validación de Campos Vacíos:

- **Si alguno de los campos del formulario está vacío, muestra un mensaje de error indicando que todos los campos deben estar completos.**

```
} catch (Exception $e) {
    // Manejar cualquier otro tipo de excepción que no haya sido capturada antes
    echo "<p style='color: red;'>Error: " . htmlspecialchars(string: $e->getMessage()) . "</p>";
    RegistrarLog(categoría: "Error", accion: "Error en el método POST");
}
?>
```

Este código asegura que los datos de la ubicación se insertan correctamente en la base de datos, gestionando tanto la sanitización como la validación de la entrada. Además, proporciona manejo de errores adecuado en caso de fallos en el proceso de inserción.

7. CONCLUSIONES Y PROPUESTAS DE MEJORA

Conclusiones:

El módulo de gestión de máquinas de vending está funcionando correctamente, permitiendo la asignación de máquinas a ubicaciones y la gestión de las mismas a través de formularios sencillos y consultas a la base de datos. El manejo de errores mediante bloques try-catch garantiza que los fallos sean controlados y no interrumpan el funcionamiento del sistema. Además, las operaciones CRUD para máquinas y ubicaciones están implementadas de manera efectiva.

Propuestas de Mejora:

1. **Optimización de Consultas: Mejorar el rendimiento de las consultas y considerar el uso de índices en tablas clave como máquina y ubicación para acelerar el acceso a datos a medida que el sistema crece.**
2. **Mejorar la UI: Mejorar el diseño visual de la interfaz y adaptarla a dispositivos móviles usando frameworks como Bootstrap. Agregar interacciones dinámicas (como actualización en tiempo real) para una mejor experiencia de usuario.**
3. **Gestión de Errores: Mejorar la gestión de errores mostrando mensajes más amigables y registrando más detalles en los logs para facilitar la resolución de problemas.**
4. **Autenticación y Roles de Usuario: Implementar un sistema de autenticación de usuarios con roles y permisos específicos, mejorando la seguridad y control de acceso al sistema.**

5. **Automatización de Asignación de Máquinas: Desarrollar un sistema que sugiera automáticamente máquinas disponibles para asignar a nuevas ubicaciones, optimizando el proceso.**
6. **Pruebas y Control de Calidad: Implementar pruebas unitarias y de integración para asegurar que el sistema funcione de manera consistente y detectar problemas antes de su implementación.**
7. **Documentación Completa: Mejorar la documentación del sistema, detallando procesos, base de datos y guías de uso para facilitar el mantenimiento y futuras ampliaciones.**

Conclusión Final:

El sistema cumple con los objetivos iniciales, pero con las mejoras propuestas en rendimiento, diseño, seguridad y pruebas, se podrá optimizar su funcionamiento y facilitar su escalabilidad y mantenimiento a largo plazo.

8. ANEXOS

8.1. ANEXO 1

8.2. ANEXO 2