Informe Tópicos avanzados de datos

Base de datos UCEN

Alumnos: Emilio Moyano, Javier Villafaña, Matias Auriol

Profesor: Mario Ortiz

Fecha estimada de entrega: 14 Julio 2023

Índice

Introducción

En el presente informe se realizó una base de datos con las herramientas de Visual Studio Code, phpmyadmin y xampp, en donde se imitó una base de datos real tematizada en nuestra institución educativa actual.

Se hablará de las bases que sentaron al proyecto tales como el modelo MER como de los problemas presentados, técnicas para su solución y buenas prácticas para optimizar el flujo del trabajo. Este proyecto fue realizado tanto en el horario de clases como también de forma remota.

Los códigos utilizados fueron adaptados para el correcto funcionamiento de la base de datos, en donde se utilizaron herramientas de inteligencia artificial como ChatGPT para optimizar las horas de trabajo.

División de grupo

Emilio Moyano: **GETS Y PLANI / Apartado visual / Desarrollo de informe**

Matías Auriol: **CRUDS / Apartado visual / Desarrollo de informe**

Javier Villafaña: **CRUDS / Apartado visual / Desarrollo de presentaciones**

Desarrollo

Las bases del proyecto fueron los códigos otorgados por el profesor, estos vendrían a sentar nuestra primera base de datos para tener un indicio de cómo tendríamos que progresar en el proyecto. También estos códigos abrieron la puerta a tener nuestras primeras plataformas web plani y fullcallender, en donde podríamos ver las modificaciones que le podríamos hacer a la BD directamente.

En nuestra BD agregamos diferentes funciones get, las cuáles nos ayudarán a traer datos que solicitemos, y plani nos mostrará una interfaz gráfica de estos datos.

En base a lo aprendido con la estructura get_levels, agregamos a nuestra BD los siguientes get:

- Get_alumnos

```php
<?php
// Conexion a la base de datos
$conexion = mysqli_connect("localhost", "root", "", "planeacion");
$query = "SELECT rut, nombre, apellido_paterno, apellido_materno FROM alumnos";
$resultado = mysqli_query($conexion, $query);
$alumnos = array();
while ($row = mysqli_fetch_assoc($resultado)){
    $alumnos[] = $row;
}
header('Content-Type: application/json');
echo json_encode($alumnos);
?>
```

- Get_asignaturas

```php
<?php
// Conexion a la base de datos
$conexion = mysqli_connect("localhost", "root", "", "planeacion");
$query = "SELECT id_asignatura, nombre_asignatura FROM asignaturas";
$resultado = mysqli_query($conexion, $query);
$asignaturas = array();
while ($row = mysqli_fetch_assoc($resultado)){
    $asignaturas[] = $row;
}
header('Content-Type: application/json');
echo json_encode($asignaturas);
?>
```

- Get_bloques

```php
<?php
// Conexion a la base de datos
$conexion = mysqli_connect("localhost", "root", "", "planeacion");
$query = "SELECT id_bloque, hora_ini, hora_fin FROM bloques";
$resultado = mysqli_query($conexion, $query);
$bloques = array();
while ($row = mysqli_fetch_assoc($resultado)){
    $bloques[] = $row;
}
header('Content-Type: application/json');
echo json_encode($bloques);
?>
```

- Get_carreras

```php
<?php
// Conexion a la base de datos
$conexion = mysqli_connect("localhost", "root", "", "planeacion");
$query = "SELECT id_carrera, nombre_carrera, facultad FROM carreras";
$resultado = mysqli_query($conexion, $query);
$carreras = array();
while ($row = mysqli_fetch_assoc($resultado)){
    $carreras[] = $row;
}
header('Content-Type: application/json');
echo json_encode($carreras);
?>
```

- get_profesores

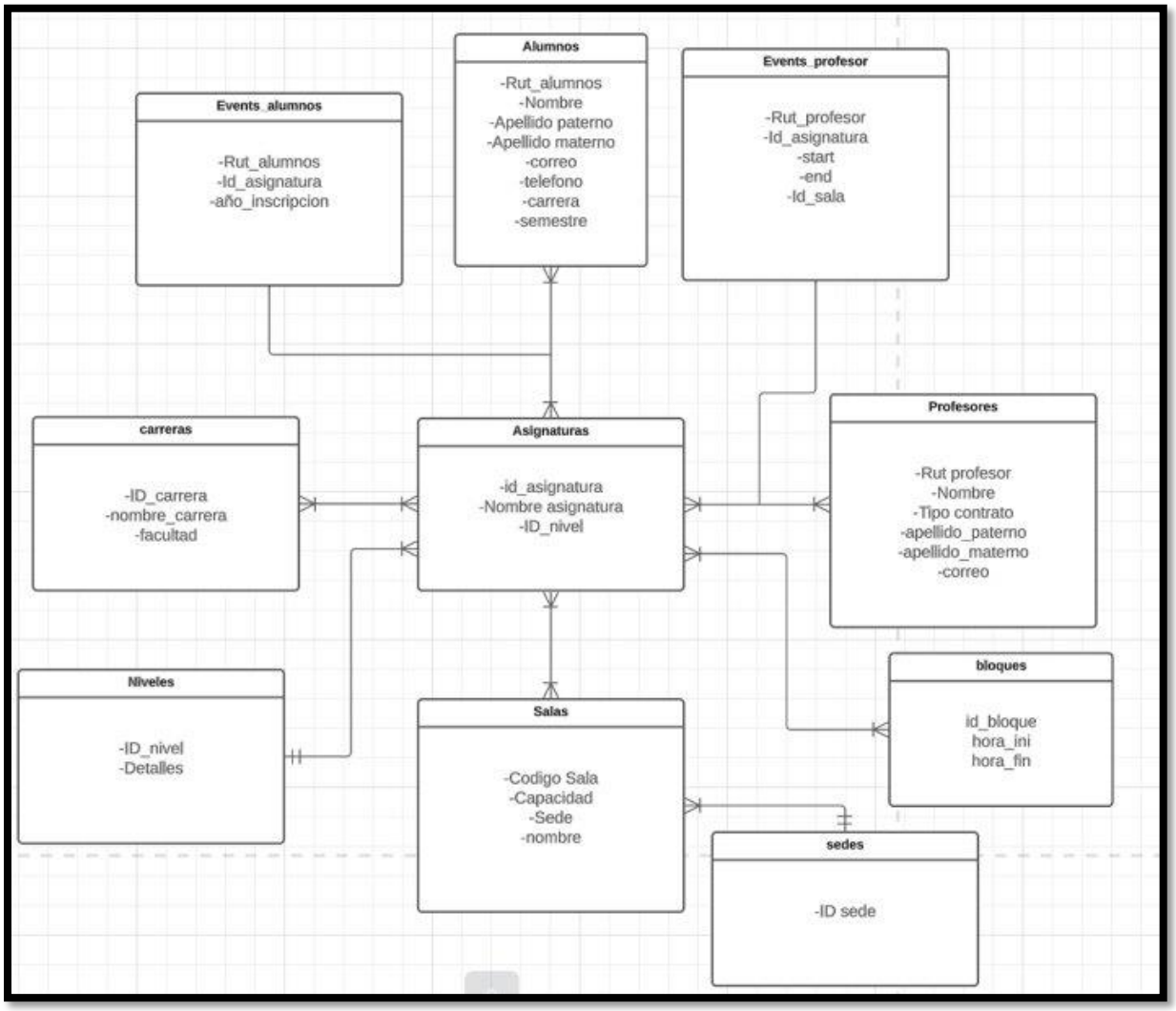
```php
<?php
// Conexion a la base de datos
$conexion = mysqli_connect("localhost", "root", "", "planeacion");
$query = "SELECT rut, nombre, tipo_contrato, apellido_paterno , apellido_materno, correo FROM profesores";
$resultado = mysqli_query($conexion, $query);
$profesores = array();
while ($row = mysqli_fetch_assoc($resultado)){
    $profesores[] = $row;
}
header('Content-Type: application/json');
echo json_encode($profesores);
?>
```

Una vez creados, se procedió a la creación de los plani faltantes, los cuáles son:

- Planigen
- Planialumnos
- Planiprof

Modelo MER:

Para crear los “planos” de nuestra base de datos recurrimos a un modelo MER, en dónde se obtuvieron todas las variables a crear, generando un orden más limpio a la hora de organizarnos con el proyecto y con sus distintas funcionalidades.



Creación Menú:

Para navegar por nuestra Base de datos se creó un menú general, que nos daría una interfaz gráfica bastante intuitiva y siguiendo los modelos estándares actuales. El código generado es el siguiente:

```
<body>
    <div>
        <h1>Menu</h1>

        <h2>Horarios</h2>
        <a href="planialumno.html">Alumnos</a>
        <a href="planiprof.html">Profesores</a>
        <a href="planigen.html">Asignaturas / Niveles</a>

        <h2>CRUDS</h2>
        <a href="CRUDS/profesores/menu.php">Profesores</a>
        <a href="CRUDS/alumnos/menu.php">Alumnos</a>
        <a href="CRUDS/asignaturas/menu.php">Asignaturas</a>
        <a href="CRUDS/carreras/menu.php">Carreras</a>
        <a href="CRUDS/Bloques/menu.php">Bloques</a>
    </div>
</body>
</html>
```

CRUDS

Se agregaron diferentes funciones para agregar, eliminar, actualizar datos de nuestra BD, además se creó un menú para contener todas las funciones de manera organizada debido a que se separan por:

- Alumnos
- Profesores
- Asignaturas
- Bloques
- Carrera

Cabe mencionar que cada apartado tiene su propio menú, que muestran las funciones que estos pueden realizar. Este es el menú de los CRUDS:

Menú

```
</head>
<body>
    <div>
        <h1>Menú</h1>

        <h2>Horarios</h2>
        <a href="planialumno.html">Alumnos</a>
        <a href="planiprof.html">Profesores</a>
        <a href="planigen.html">Asignaturas / Niveles</a>

        <h2>CRUDS</h2>
        <a href="CRUDS/profesores/menu.php">Profesores</a>
        <a href="CRUDS/alumnos/menu.php">Alumnos</a>
        <a href="CRUDS/asignaturas/menu.php">Asignaturas</a>
        <a href="CRUDS/carreras/menu.php">Carreras</a>
        <a href="CRUDS/Bloques/menu.php">Bloques</a>
    </div>
</body>
</html>
```

Agregar

Esta función nos permitirá agregar nuevas variables al sistema con sus respectivas clases, teniendo en cuenta sus llaves primarias y sus demás atributos, como ejemplo del código, usaremos alumnos:

```
public function agregarAlumnos() {
    // Obtiene los datos del formulario y los inserta en la base de datos
    $rut = $_POST['rut'];
    $nombre = $_POST['nombre'];
    $apellidoPaterno = $_POST['apellido_paterno'];
    $apellidoMaterno = $_POST['apellido_materno'];
    $correo = $_POST['correo'];
    $telefono = $_POST['telefono'];
    $carrera = $_POST['carrera'];
    $semestre = $_POST['semestre'];

    $resultado = $this->model->insertAlumnos($rut, $nombre, $apellidoPaterno, $apellidoMaterno, $correo, $telefono, $carrera, $semestre);

    if ($resultado) {
        echo "<script>
        Swal.fire({
            title: 'Éxito',
            text: 'Acción realizada correctamente',
            icon: 'success',
            showConfirmButton: false,
            timer: 2000
        }).then(function() {
            window.location.href = 'Menu.php';
        });
    </script>";
    } else {
        echo "Error al agregar el alumno";
        echo "<br>";
        echo "<a href='menu.php'>Volver al menú</a>";
    }
```

Eliminar

Esta función nos permitirá eliminar datos de nuestra BD. Usaremos la variable Alumnos

```php
public function eliminarAlumnos($rut) {
    $query = "DELETE FROM alumnos WHERE rut = :rut";
    $statement = $this->conexion->prepare($query);
    $statement->bindParam(':rut', $rut);
    $result = $statement->execute();
    return $result;

}
```

```php
    public function eliminarAlumnos($rut) {
        $rut = $_POST['rut'];
        $resultado = $this->model->eliminarAlumnos($rut);

        if ($resultado) {
            echo "<script>
                Swal.fire({
                    title: 'Éxito',
                    text: 'Acción realizada correctamente',
                    icon: 'success',
                    showConfirmButton: false,
                    timer: 2000
                }).then(function() {
                    window.location.href = 'Menu.php';
                });
            </script>";
        } else {
            echo "Error, el registro no existe";
            echo "<br>";
            echo "<a href='menu.php'>Volver al menú</a>";
        }

}
}
```

Actualizar:

Esta función permitirá editar los datos de cualquier variable. Usaremos la variable Alumnos

```php
public function updateAlumnos($rut, $nombre, $apellidoPaterno, $apellidoMaterno, $correo, $telefono ,$carrera,$semestre) {
    // Actualiza los datos de un alumno en la base de datos
    $query = "UPDATE alumnos SET nombre = :nombre, apellido_paterno = :apellidoPaterno, apellido_materno = :apellidoMaterno, correo = :correo, telefono = :telefono, carrera = :carrera, semestre = :semestre WHERE rut = :rut";
    $statement = $this->conexion->prepare($query);
    $statement->bindParam(':rut', $rut);
    $statement->bindParam(':nombre', $nombre);
    $statement->bindParam(':apellidoPaterno', $apellidoPaterno);
    $statement->bindParam(':apellidoMaterno', $apellidoMaterno);
    $statement->bindParam(':telefono', $telefono);
    $statement->bindParam(':correo', $correo);
    $statement->bindParam(':carrera', $carrera);
    $statement->bindParam(':semestre', $semestre);
    $result = $statement->execute();
    return $result;
}
public function verificarRut($rut) {
    $query = "SELECT * FROM alumnos WHERE rut = :rut";
    $statement = $this->conexion->prepare($query);
    $statement->bindParam(':rut', $rut);
    $statement->execute();
    $alumno = $statement->fetch(PDO::FETCH_ASSOC);

    if ($alumno) {
        // El alumnos existe, se redirige a la página de edición con los datos del alumnos
        $url = "update_alumnos.php?rut=" . urlencode($rut);
        header("Location: $url");
        exit(); // Se recomienda salir del script después de redirigir
    } else {
        // El alumnos no existe, se puede mostrar un mensaje de error o redirigir a otra página de manejo de errores
        header("Location: error");
        exit();
    }
}
public function getErrorInfo() {
    $errorInfo = $this->conexion->errorInfo();
    return $errorInfo[2]; // Devuelve solo el mensaje de error
}
```

```php
public function verificarRut($rut) {
    $existeRut = $this->model->verificarRut($rut);

    if ($existeRut) {
        echo "El RUT existe en la base de datos.";
    } else {
        echo "El RUT no existe en la base de datos.";
    }
}

public function actualizarAlumnos() {
    $rut = $_POST['rut'];
    $nombre = $_POST['nombre'];
    $apellidoPaterno = $_POST['apellido_paterno'];
    $apellidoMaterno = $_POST['apellido_materno'];
    $correo = $_POST['correo'];
    $telefono = $_POST['telefono'];
    $carrera = $_POST['carrera'];
    $semestre = $_POST['semestre'];

    $resultado = $this->model->updateAlumnos($rut, $nombre, $apellidoPaterno, $apellidoMaterno, $correo, $telefono, $carrera, $semestre);

    if ($resultado) {
        echo "<script>
            Swal.fire({
                title: 'Éxito',
                text: 'Acción realizada correctamente',
                icon: 'success',
                showConfirmButton: false,
                timer: 2000
            }).then(function() {
                window.location.href = 'Menu.php';
            });
        </script>";
    } else {
        echo "Error al actualizar los datos del alumno";
        echo "<br>";
        echo "<a href='menu.php'>Volver al menú</a>";
    }
}
```

Problemas

- El primer problema fue la falta de práctica de los lenguajes utilizados y requeridos, los cuales eran php y html junto a javascript. La solución fue mejorar la práctica como equipo en estos lenguajes y así solucionar el inconveniente
- Posteriormente la interpretación del plani fue poco familiar al inicio, a medida que se agregaron funciones este problema se fue eliminando.
- Existieron problemas en los códigos de actualización, dando errores de formato de variable en estos. Se utilizaron diversas herramientas como ChatGPT para solucionar algunos de estos errores, además de consultas con el profesor de la asignatura.
- A la hora de asociar el CRUD con el plani existieron problemas al momento de enlazarlos, ya que no se lograba mostrar la plataforma de horarios requeridos en un principio. Este error se corrigió al seguir practicando con el código realizado y consultas al profesor de la asignatura.
- La conexión de events no conectaba con los CRUDS, dando así un problema notorio a la hora de querer realizar más acciones en la BD.
- Problemas a la hora de crear la estética del menú principal y de los CRUDS, utilizamos también sweet alert para dar un mejor resultado.

Conclusiones y beneficios.

Para concluir este proyecto, se aprendieron muchísimos conceptos de cómo funciona una base de datos por detrás, cómo obtiene sus datos e intercambia información sin mezclarse, además de mejorar como grupo las técnicas con los lenguajes de programación y del proyecto en sí.

Se aprendió el correcto manejo de herramientas y sus técnicas, mejorando así nuestro progreso en este proyecto y en trabajos futuros relacionado a base de datos o que involucren este tipo de softwares.

Buenas prácticas utilizadas fue anotar cada cambio que se generaba en los códigos utilizados, además de guardar páginas web de donde se sacó información para así facilitar este informe.

Algunas de las problemáticas se conservaron del avance anterior entregado, ya que se estimó conveniente nombrar el paso de todos los errores ocurridos en el transcurso de este proyecto.

El conocimiento adquirido se podrá utilizar en proyectos laborales, creación de páginas webs, extracción de información en BD, etc.

El proyecto fue bastante más llevadero con el conocimiento que se nos entregó y con la ayuda tanto del profesor de la asignatura como compañeros a quienes consultamos, logramos realizar todo lo solicitado.

Bibliografía

https://chat.openai.com

Consultas realizadas en clases con profesor Mario Ortiz

https://sweetalert2.github.io