Poyecto Final Base de Datos

Jonathan Cazco[00136618] and Erick Ñauñay[00136877]

Universidad San Francisco de Quito, Quito-Ecuador

Abstract. El proyecto FORTEX DB se enfocó en digitalizar los datos del gimnasio Fortex Train Center. Es decir, se transfirió información desde la base de datos física hacia la base de datos del proyecto. Esto permitió un mejor control por sobre los clientes, personal administrativo e inventario de la sucursal. La interfaz gráfica facilito el reporte y visualización de datos de los clientes, así como tareas de manutención y actualización de datos. Se tomó en consideración otras áreas del gimnasio para mejorar la atención al cliente y formalizar los historiales (Fisioterapia, Nutrición, Entrenamiento). La implementación de módulos de reporte grafico de datos fueron vitales para la presentación de graficas construidas sobre el historial de medidas de cada cliente. De igual manera se proyectó la creación de un módulo de facturación, presentado a manera de registros en tabla en la actual aplicación, para el futuro mejoramiento, desarrollo y mantenimiento del proyecto.

Keywords: Digitalizar · Reporte · Visualización.

1 Introducción

El gimnasio FORTEX Training Center, ubicado en la Av. Amazonas y Oyacachi, es un establecimiento dedicado al trabajo personalizado de programas de ejercicio y alimentación para una salud integral. A pesar de ser un establecimiento relativamente nuevo, el centro cuenta con más de 170 clientes tanto en el área de gimnasio como en el área fisioterapéutica. El principal problema que conllevó el incremento de volumen en su clientela fue ineficiencia en la recuperación y actualización de registros e historiales debido a que toda la información de los clientes se encuentra tabulada en hojas de papel. El dueño del establecimiento no tuvo mayor problema con su forma original de almacenar datos, pero le gustaría una solución tecnológica para centralizar y organizar de mejor manera su

gimnasio. El objetivo principal del proyecto es la crear una aplicación de escritorio que simplifique el manejo de datos del gimnasio. La aplicación ofrece una mejor administración del establecimiento incluyendo aspectos antes no considerados por el propietario como nomina de personal e inventario del gimnasio. En conjunto con la aplicación, se plantearon tres objetivos secundarios enfocados en sus funcionalidades:

- Implementar de un módulo de reporte visual para el seguimiento en medidas de los clientes.
- Construir y presentar vistas resumidas de tablas con sus respectivas funciones de administración (añadir, eliminar, editar) para cada área del gimnasio.
- Probar un modulo de facturación para futuro desarrollo en la aplicación.

2 Materiales y Métodos

2.1 Materiales

Para el desarrollo de la aplicación se determinó el uso de MySQL como gestor de la base de datos del gimnasio, el cual utiliza InnoDB como su motor. Esta base de datos se encuentra alojada en la nube en un servidor remoto de una instancia de Google Cloud Platform con acceso a internet, para garantizar versatilidad y una recuperación de datos segura y siempre disponible. En conjunto, se gestionó la creación de tres usuarios: dos para el desarrollo de la aplicación y un administrador para el propietario. Los primeros cuentan con todos los permisos para el acceso y modificación de tablas en información desde la parte programática del proyecto. El segundo, cuenta con permisos de escritura y lectura sobre las tablas en la base de datos para el mantenimiento y actualización de datos por parte del dueño del establecimiento.

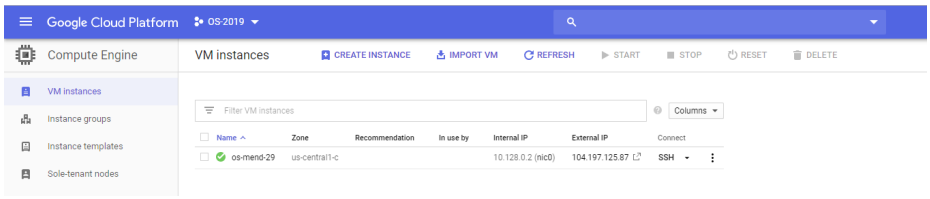

Fig. 1. Intancia en la nube la cual aloja la base de datos para el proyecto.

Fig. 2. Estatus del servidor en la nube.

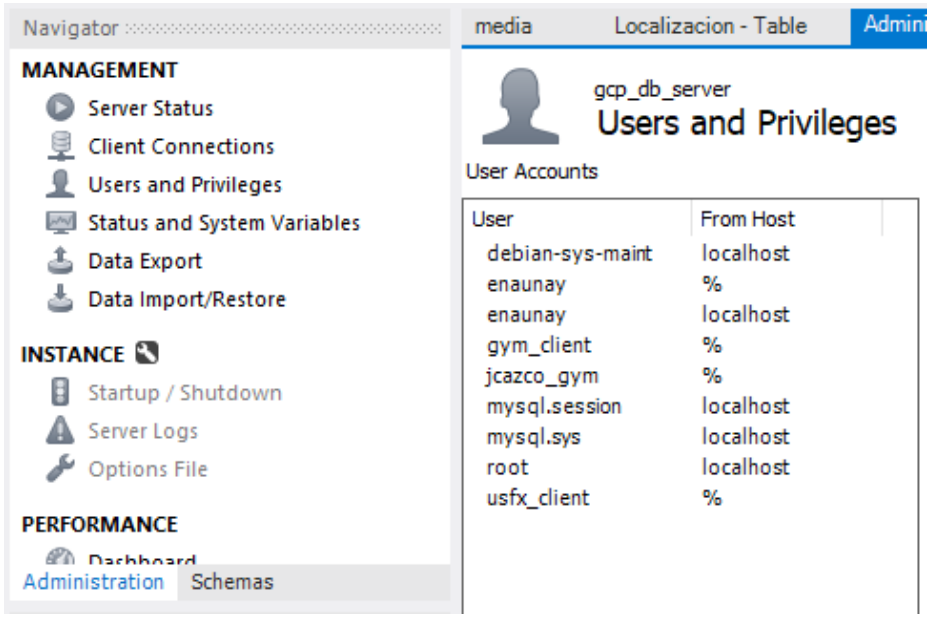

Fig. 3. Usuarios registrados en la conexion remota. Los usuarios enaunay , jcazco_gym y gym_client corresponden a los usuarios con permisos sobre la base de datos del proyecto.

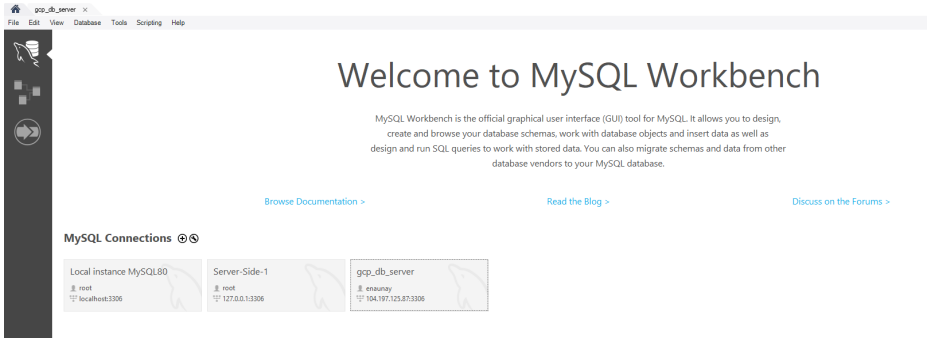

Fig. 4. Conexión desde MySQL Workbench para el usuario enaunay.

En paralelo, el desarrollo de la aplicación se lo hizo en el lenguaje de Java, versión 11. Para el diseño del GUI se utilizó SceneBuilder con JavaFX12 mientras que la parte programática fue desarrollada en el IDE Eclipse 2019.

2.2 Métodos

El modelamiento de la base de datos se realizo utilizando el asistente de diagramas de MySQL. Este modelo cuenta con 14 tablas que representan las entidades principales y secundarias del establecimiento. El modelo fue adaptado y diseñado en conjunto con el propietario del gimnasio para semejar, pero a la vez optimizar, el método tradicional de almacenamiento. De igual manera, se tomo en cuenta un diseño lógico para recuperar trabajar con dicha información desde la parte programática y su presentación en el GUI.

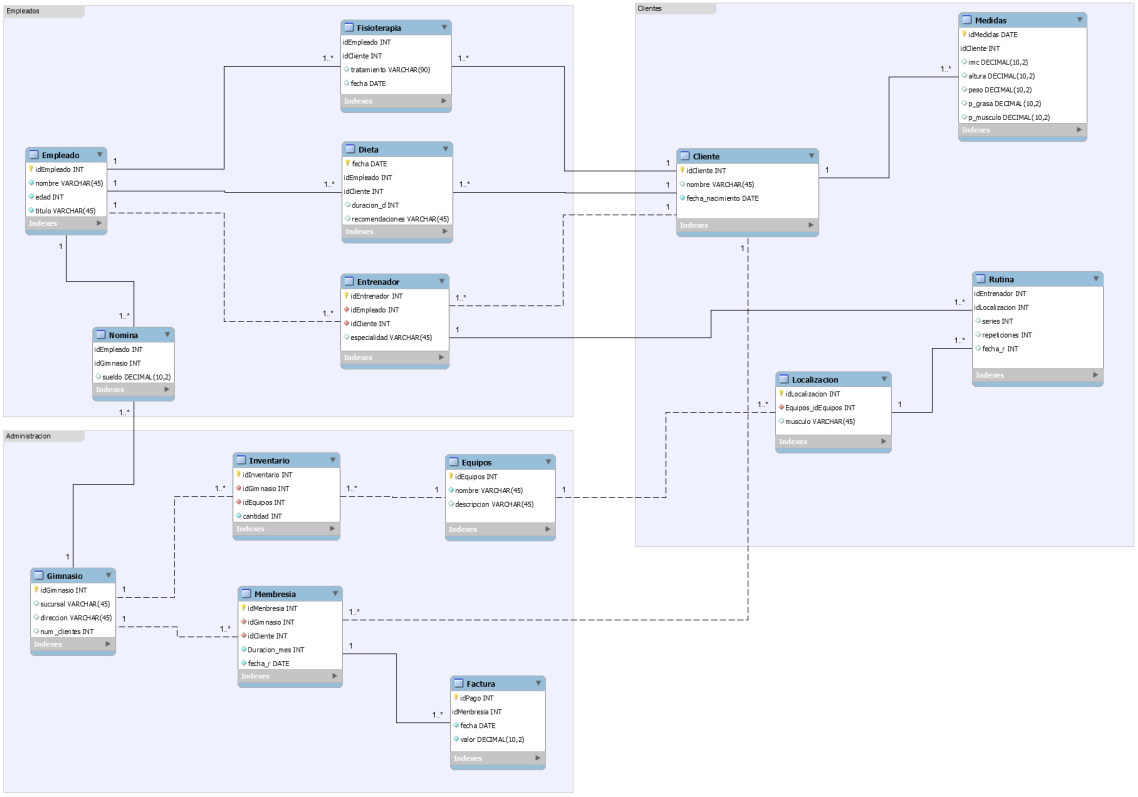

Fig. 5. Conexión desde MySQL Workbench para el usuario enaunay.

A continuacion, se presentan los procedimientos de mayor importancia para el propietario del gimnasio. Cada procedimiento implementa transacciones para garantizar la correcta operacion del procedimiento. Estos procedimientos corresponden a las tareas de registro de clientes, sus membresías y posteriormente la asignación de rutinas de enternamiento.

1. **Agregar nuevo cliente, valida existencia previa**

```sql
-- 1. Agregar nuevo cliente. valida existencia
CREATE PROCEDURE nuevo_cliente (
    in new_id INT,
    in new_nombre VARCHAR(45),
    in new_fecha DATE,
    out return_val INT
)
BEGIN
    declare id_search int default -1;

    START TRANSACTION;

        SELECT
            idCliente
        INTO id_search FROM
            Cliente
        WHERE
            idCliente = new_id;

        IF id_search != new_id THEN
            INSERT INTO
                Cliente
            VALUES
                (new_id, new_nombre, new_fecha);
        ELSE
            SET return_val = 1;
            rollback;

        END IF;

    COMMIT;
    SET return_val = 0;

END $$
```

Fig. 6. SQL que resuelve el ingreso de un nuevo cliente a la base de datos, validando existencia previa.

2. **Ingreso de membresia de un cliente si es nuevo cliente o si quiere contratar una nueva memebresia**

```
-- 2. Ingreso de membresia de un cliente si es nuevo cliente o si quiere contratar una nueva memebresia

CREATE PROCEDURE nueva_membresia (
    in new_gym INT,
    in new_cliente INT,
    in new_duracion INT,
    in new_fecha DATE,
    out return_val INT
)
BEGIN
    declare id_1 int default -1;

    START TRANSACTION;

        SELECT
            Duracion_mes
        INTO id_1
        FROM
            Membresia
        WHERE
            idCliente = new_cliente and idGimnasio = new_gym;

        IF id_1 != -1 AND id_search != new_duracion THEN
            INSERT INTO
                Membresia (idGimnasio, idCliente, Duracion_mes, fecha_r)
            VALUES
                (new_gym, new_cliente, new_duracion, new_fecha);
        ELSE
            SET return_val = 1;
            rollback;

        END IF;

    COMMIT;
    SET return_val = 0;

END $$
```

Fig. 7. SQL que resuelve la asignación de membresias por cliente.

3. **Crear relacion entre cliente y entreador para poder crear rutinas, valida existencia y nueva especialidad**

```sql
-- 3. crea Relacion entre cliente y entreador para poder crear rutinas, valida existencia y nueva especialidad

CREATE PROCEDURE relacion_entrenador (
    in new_empleado INT,
    in new_cliente INT,
    in new_especialidad VARCHAR(45),
    out return_val INT
)
BEGIN
    declare id_1 int default -1;
    declare id_2 int default -1;

    declare id_rel int default -1;
    declare rel_esp VARCHAR(45);

    START TRANSACTION;

        SELECT
            idEntrenador, especialidad
        INTO id_rel, rel_esp
        FROM
            Entrenador
        WHERE
            idCliente = new_cliente and idEmpleado = new_empleado;

        IF id_rel = -1 THEN

            SELECT
                idEmpleado
            INTO id_1
            FROM
                Entrenador
            WHERE
                idEmpleado = new_empleado;
```

```
            SELECT
                idCliente
            INTO id_2
            FROM
                Cliente
            WHERE
                idCliente = new_cliente;

            IF id_1 != -1  and id_2 != -1 THEN
                INSERT INTO
                    Entreador (idEmpleado, idCliente, especialidad)
                VALUES
                    (new_gym, new_cliente, new_duracion, new_fecha);
            ELSE
                SET return_val = 1;
                rollback;
            END IF;
        ELSE

            IF rel_esp != new_especialidad THEN
                INSERT INTO
                    Entreador (idEmpleado, idCliente, especialidad)
                VALUES
                    (new_empleado, new_cliente, new_especialidad);
            ELSE
                SET return_val = 1;
                rollback;
            END IF;

        END IF;

    COMMIT;
    SET return_val = 0;

END $$

DELIMITER ;
```

Fig. 8. SQL que resuelve la asignación de clientes a entrenadores para su futuro uso en la designación de rutinas de entrenamiento.

De igual manera se incluyen las vistas más usadas, o de datos más recuperados, para el empleador. Estas views también trabajan a nivel programatico de la aplicacion para los reportes.

4. **Equipos más utilizados**

```sql
-- 1. Reporte de las maquinas mas usadas por entrendaores en rutinas de upper body
CREATE VIEW equipos_usados_en rutinas_upper_body
AS
SELECT
    Equipo.nombre, Equipo.descripcion
FROM
    Rutina
INNER JOIN
     Localizacion USING (idLocalizacion)
INNER JOIN
    Equipos USING (idEquipos)
WHERE Localizacion.musculo = "Bicep" OR Localizacion.musculo = "Antebrazo"
    OR Localizacion.musculo = "Tricep" OR Localizacion.musculo = "Espalda"
    OR Localizacion.musculo = "Abdomen" OR Localizacion.musculo = "Hombros"
```

Fig. 9. SQL que retorna una vista con los equipos más utilizados según localización muscular.

5. **Rutinas de un cliente específico**

```sql
CREATE VIEW rutinas_entrenadas_entrenador
AS
SELECT
    em.nombre, em.especialidad, idLocalizacion, series, repeticiones, fecha
FROM
    Rutina
INNER JOIN
    Entrenador as ent using (idEntrenador)
INNER JOIN
    Empleado as em using (idEmpleado)
WHERE
    em.nombre = "Camilo"
```

Fig. 10. SQL que retorna una vista con la o las rutinas asignadas a un cliente en cierta fecha.

6. **Rutinas por especialidad**

```sql
CREATE VIEW clientes_entrenados_kickboxing
AS
SELECT
    cl.nombre, em.especialidad, idLocalizacion, series, repeticiones, fecha
FROM
    Rutina
INNER JOIN
    Entrenador as ent using (idEntrenador)
INNER JOIN
    Cliente as cl using (idCliente)
WHERE
    ent.especialidad = "KickBoxing"
```

Fig. 11. SQL que retorna una vista con la o las rutinas asignadas de a cuerdo a la especialidad del entrenador.

7. **Registro de Dietas**

```sql
CREATE VIEW nutricionista_informacion_cliente
AS
SELECT
    Cliente.nombre as paciente, Empleado.nombre as tratante, duracion as ultima_dieta, recomendaciones
FROM
    Dieta
INNER JOIN
    Cliente  using (idCliente)
INNER JOIN
    Empleado using (idEmpleado)
```

Fig. 12. SQL que retorna una vista con las dietas asignadas a cada cliente en conjunto con su nutricionista responsable.

8. **Historial de fisioterapia**

```sql
CREATE VIEW fisioterapista_informacion_cliente
AS
SELECT
    Cliente.nombre as paciente, Empleado.nombre as tratante, fecha as ultima_revision, tratamiento
FROM
    Fisioterapia
INNER JOIN
    Cliente  using (idCliente)
INNER JOIN
    Empleado using (idEmpleado)
```

Fig. 13. SQL que retorna una vista con el historial completo de fisioterapias realizadas.

3 Conclusiones

Una completa digitalización de los datos físicos no fue posible ya que primero se debe llegar a un acuerdo de confidencialidad con el empleador para manejar datos personales de los clientes. Para fines prácticos, se utilizó datos ficticios basados en la información recopilada en el modelamiento del proyecto. Estos datos fueron los utilizados en el demo de la aplicación. Sin embargo, si se llegase a dicho acuerdo, se debería comenzar un proceso de migración de la base de datos física entera para completar totalmente dicho objetivo.

Las funciones de reporte grafico y tabular se implementaron correctamente y fueron del agrado del dueño del gimnasio. El incluir áreas como fisioterapia, dietas y el uso de equipos en el gimnasio propulso mas ideas relacionadas con el análisis de los datos para el mejoramiento del servicio de calidad hacia el cliente, basado en los historiales ahora registrados en la base de datos del proyecto.

El módulo de facturación quedara como un registro de pagos de las membresías ya que el propietario aun no se ha decidido por emitir facturas electrónicas y así utilizar dicha información.

En general, el modelamiento del proyecto a las necesidades del empleador es fundamental ya que, con un buen diseño, específico y enfocado a las necesidades del cliente, se logra una buena modelación de base de datos y por ende, un fácil manejo programático de los datos para los reportes.

4 Recomendaciones

En etapa de construcción del proyecto se tuvieron varios problemas al instalar el servidor remoto y administrarlo correctamente, este proceso debe ser cuidadosa y responsablemente ejecutado con unas buenas políticas de seguridad, ya que para poder conectarse al servidor de forma remota para administrarlo o acceder a las bases de datos internas la maquina donde se localice el MySQL Server debe tener una IP estática, por ende, es visible en toda la Internet. Al levantar el servidor se debe verificar los puertos de entrada y salida y si se esta utilizando la nube como plataforma de maquinas virtuales, verificar que ambos firewalls (máquina y plataforma) estén con los puertos deseados abiertos y bien configurados.

Como segunda recomendación, se debe hacer un análisis exhaustivo de que permisos otorgar dependiendo el rol de la persona que va a acceder a los datos del servidor, además de asegurar roles estrictos y que no puedan ver información delicada que comprometa la integridad del servicio. Como tercera recomendación, la conexión de la aplicación a la base de datos debe estar data por un usuario independiente, no usar usuarios con privilegios innecesarios, ya que esto compromete la información guardada en el servidor.

Finalmente, la aplicación como depende de un servicio externo debe manejar correctamente las excepciones, si en alguna ocasión existiese problemas de conexión o inconsistencia en los datos enviados al servidor. Se debe tener protocolos bien definidos para ataques a la base de datos como inyección SQL.

References

1. Matthews, M., Cole, J., & Gradecki, J. D. (2003). MySQL and Java developer's guide. John Wiley & Sons.
2. Yarger, R. J., Reese, G., King, T., & Oram, A. (1999). MySQL and mSQL. O'Reilly & Associates, Inc..
3. Ronstrom, M., & Thalmann, L. (2004). MySQL cluster architecture overview. MySQL Technical White Paper.
4. Zoratti, I. (2006). MYSQL security best practices.
5. Thibaud, C. (2006). MySQL 5: instalación, implementación, administración, programación. Ediciones ENI.