UNIVERSIDAD DEL VALLE DE GUATEMALA

Bases de Datos 1 – Sección 20

Ing. Alexander Bolaños

Proyecto 1: Integración con Ingeniería de Software

Osman Emanuel de León García – 23428

Milton Giovanni Polanco Serrano - 23471

Gadiel Amir Ocaña Véliz - 231270

Guatemala, 24 de marzo de 2025

Descripción del Proyecto de Ingeniería de Software

El proyecto consiste en el desarrollo de una aplicación web para la gestión de finanzas personales, llamada MoneyFlow. Esta aplicación está diseñada para ayudar a los usuarios a registrar, analizar y optimizar sus ingresos y gastos, fomentando la toma de decisiones financieras más informadas. Está dirigida a un público diverso, que incluye:

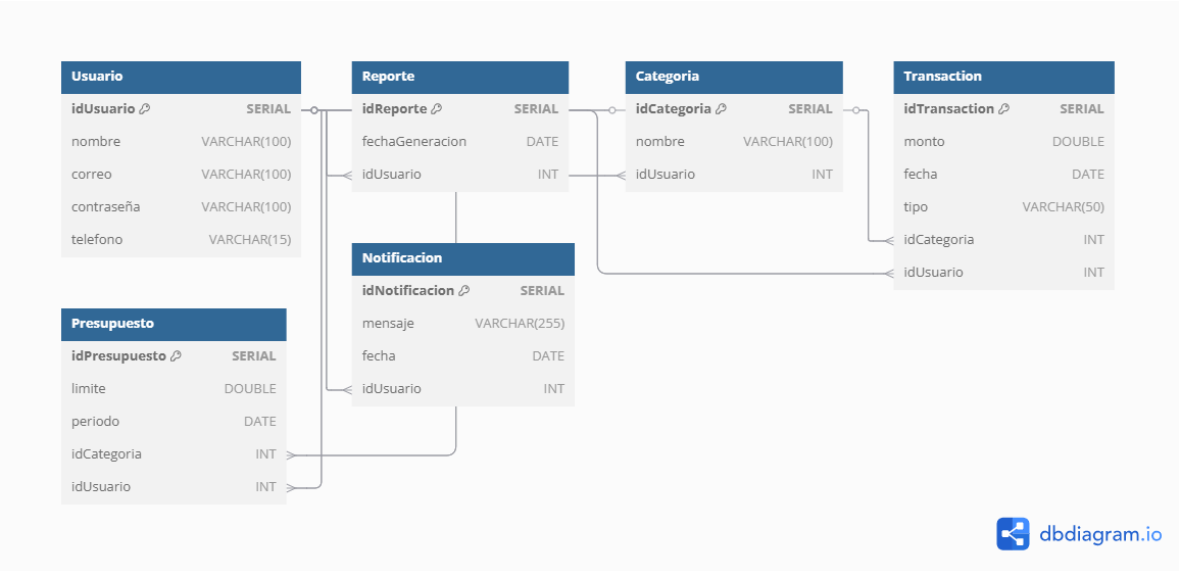
- **Estudiantes universitarios**: Que buscan aprender a manejar su dinero y establecer metas de ahorro.
- **Profesionales independientes**: Que necesitan gestionar ingresos irregulares y múltiples cuentas.
- **Familias y amas de casa**: Que desean controlar los gastos del hogar y planificar ahorros a largo plazo.

La aplicación permitirá a los usuarios:

- Registrar manualmente sus ingresos y gastos.
- Clasificar transacciones en categorías personalizadas.
- Establecer y monitorear presupuestos.
- Generar reportes financieros con gráficos interactivos.
- Recibir notificaciones y alertas sobre su situación financiera.

La base de datos es un componente central del proyecto, ya que almacena toda la información financiera de los usuarios y permite realizar análisis y consultas para generar reportes y notificaciones.

Diagrama Entidad-Relación (ER)



El diagrama ER representa la estructura de la base de datos para el sistema de gestión de finanzas personales. Aquí están las entidades principales y sus relaciones:

- **Usuario**: Almacena la información de los usuarios, incluyendo nombre, correo, contraseña y teléfono. Es la entidad central, ya que todas las demás entidades están relacionadas con ella.
- **Categoría**: Representa las categorías de gastos o ingresos que los usuarios pueden crear. Cada categoría está asociada a un usuario.
- **Transaction**: Registra las transacciones financieras (ingresos y gastos) de los usuarios. Cada transacción está asociada a una categoría y a un usuario.
- **Presupuesto**: Define los límites de gasto para cada categoría. Cada presupuesto está asociado a una categoría y a un usuario, y ahora incluye un campo periodo para definir el período del presupuesto (mes, año, etc.).
- **Reporte**: Almacena los reportes generados por los usuarios. Cada reporte está asociado a un usuario.
- **Notificacion**: Registra las notificaciones enviadas a los usuarios, como alertas de presupuesto o recordatorios. Cada notificación está asociada a un usuario.

Relaciones:

- Un Usuario puede tener múltiples Categorías, Transacciones, Presupuestos, Reportes y Notificaciones.
- Una Categoría puede tener múltiples Transacciones y Presupuestos.
- Una Transacción y un Presupuesto están asociados a una Categoría y a un Usuario.

Scripts SQL (DDL)

El script DDL se encargó de crear la estructura inicial de la base de datos, definiendo las tablas y sus relaciones. A continuación, se describe lo que se hizo:

- **Creación de tablas**: Se crearon las tablas principales, como Usuario, Categoria, Transaction, Presupuesto, Reporte y Notificacion. Cada tabla fue diseñada con campos específicos para almacenar la información relevante, como nombres, fechas, montos y tipos de transacciones.
- **Definición de claves primarias y foráneas**:
  - Cada tabla tiene una clave primaria autoincremental (por ejemplo, idUsuario, idCategoria, etc.).
  - Se establecieron relaciones entre las tablas mediante claves foráneas, como idUsuario en Categoria y idCategoria en Transaction.
- **Restricciones y valores por defecto**:
  - Se aplicaron restricciones como NOT NULL en campos esenciales (por ejemplo, nombre, correo, monto, etc.).
  - El campo correo en la tabla Usuario se definió como único (UNIQUE) para evitar duplicados.

Script de Datos de Prueba (INSERT)

Para validar el diseño de la base de datos, se insertaron datos de prueba en cada una de las tablas. A continuación, se describe el proceso:

- **Inserción de usuarios**: Se agregaron registros ficticios en la tabla Usuario, incluyendo nombres, correos electrónicos, contraseñas y números de teléfono.

- **Inserción de categorías**: Se crearon categorías como "Comida" y "Transporte", asociadas a usuarios específicos.
- **Inserción de transacciones**: Se registraron transacciones de ingresos y gastos, asociadas a categorías y usuarios.
- **Inserción de presupuestos**: Se definieron límites de gasto para categorías específicas, incluyendo el nuevo campo periodo.
- **Inserción de reportes**: Se generaron reportes ficticios con fechas de generación y usuarios asociados.
- **Inserción de notificaciones**: Se agregaron notificaciones de alertas y recordatorios, asociadas a usuarios.

Estos datos permitieron probar las relaciones entre las tablas y asegurar que la base de datos funciona correctamente.

Versionamiento y Migraciones

El versionamiento de la base de datos se gestionó mediante dos versiones principales:

1. **Versión Inicial (v1.0)**:
    - Creación de las tablas básicas: Usuario, Categoria, Transaction, Presupuesto y Reporte.
2. **Versión 1.1**:
    - Se agregó la tabla Notificacion para manejar notificaciones de presupuestos y recordatorios.
    - Se modificó la tabla Presupuesto para incluir el campo periodo, que permite definir el período del presupuesto (mes, año, etc.).

Documentación de Cambios:

- **Tabla Notificacion**: Se agregó para mejorar la experiencia del usuario con alertas y recordatorios.
- **Campo periodo en Presupuesto**: Se agregó para permitir una mejor gestión de presupuestos por período.

Reglas de Negocio y Restricciones

Reglas de Negocio:

1. Cada usuario debe tener un correo electrónico único.
2. Las transacciones deben estar asociadas a una categoría y un usuario.
3. Los presupuestos deben estar asociados a una categoría y un usuario.
4. Los reportes deben estar asociados a un usuario.
5. Las notificaciones deben estar asociadas a un usuario.

Restricciones:

- **NOT NULL**: Campos como nombre, correo, contraseña, monto, fecha, etc., no pueden ser nulos.
- **UNIQUE**: El campo correo en la tabla Usuario debe ser único.
- **FOREIGN KEY**: Las claves foráneas aseguran la integridad referencial entre las tablas.

Reflexiones sobre el Proceso

Osman de León

1. **¿Por qué eligieron este sistema de gestión de bases de datos (DBMS)? ¿Qué ventajas y desventajas tiene en comparación con otros?**
   Elegimos PostgreSQL por su robustez, soporte para transacciones ACID y su capacidad para manejar grandes volúmenes de datos. Es de código abierto y tiene una gran comunidad, lo que facilita su uso y resolución de problemas. En comparación con otros DBMS como MySQL o SQLite, PostgreSQL ofrece un mejor soporte para transacciones complejas y tipos de datos avanzados, aunque requiere más configuración inicial y recursos de hardware.
2. **¿Qué estándares o criterios usaron para diseñar su base de datos?**
   Utilizamos técnicas de normalización hasta la tercera forma normal para evitar redundancias y garantizar la integridad de los datos. También aplicamos restricciones como NOT NULL, UNIQUE y claves foráneas para mantener la consistencia de los datos. Además, diseñamos la base de datos pensando en la escalabilidad, para permitir futuras expansiones.
3. **¿Cuáles son las entidades más importantes del modelo y por qué?**
   Las entidades más importantes
   son Usuario, Categoría, Transaction, Presupuesto y Notificacion. Usuario es central, ya que todas las demás entidades están relacionadas con
   ella. Categoría y Transaction permiten clasificar y registrar los gastos e ingresos, mientras que Presupuesto y Notificacion ayudan a los usuarios a controlar sus finanzas y recibir alertas.
4. **¿Cómo aplicaron las técnicas de normalización en su diseño? ¿Qué problemas evitaron gracias a esto?**
   Aplicamos normalización hasta la tercera forma normal para eliminar redundancias y dependencias innecesarias. Esto evitó problemas como la duplicación de datos, inconsistencias y complejidad en las consultas, lo que resultó en una base de datos más eficiente y fácil de mantener.
5. **¿Cómo definieron restricciones y valores por defecto para garantizar la integridad de los datos?**
   Definimos restricciones como NOT NULL para campos esenciales, UNIQUE para el correo electrónico y claves foráneas para mantener las relaciones entre tablas. Estas restricciones aseguran que los datos sean válidos y consistentes, lo que es crucial para una aplicación de finanzas personales.
6. **¿Cómo abordaron los cambios en la estructura de la base de datos?**
   Gestionamos los cambios mediante versionamiento. La versión inicial incluyó las tablas básicas, y en la versión 1.1 agregamos la tabla Notificacion y el
   campo periodo en Presupuesto. Cada cambio se documentó y se implementó en un script SQL separado, lo que permitió simular migraciones y mantener un historial de modificaciones.
7. **¿Cómo seleccionaron los datos de prueba para garantizar que el diseño es funcional?**

Seleccionamos datos de prueba que cubrieran todos los escenarios posibles, incluyendo transacciones de ingresos y gastos, presupuestos con diferentes períodos y notificaciones de alertas. Estos datos permitieron validar las relaciones entre las tablas y asegurar que la base de datos funciona correctamente.

8. **¿Cuál fue tu contribución específica en el desarrollo del proyecto? ¿Cómo se organizó el trabajo en el equipo?**
   Mi contribución incluyó el diseño del diagrama ER, la creación de scripts DDL y la inserción de datos de prueba. El trabajo se organizó en tareas específicas asignadas según las habilidades de cada miembro.
9. **¿Sientes que trabajaste equitativamente en comparación con tus compañeros? ¿Qué hubieras hecho diferente en este proyecto?**
   Sí, siento que trabajé equitativamente en comparación con mis compañeros. Cada miembro contribuyó de manera significativa. Si pudiera hacer algo diferente, habría explorado herramientas para automatizar pruebas y dedicado más tiempo a documentar cada paso del proceso para facilitar futuras actualizaciones.

Gadiel Ocaña

1. **¿Por qué eligieron este sistema de gestión de bases de datos (DBMS)? ¿Qué ventajas y desventajas tiene en comparación con otros?**
   Desde mi punto de vista, optamos por PostgreSQL porque es una solución confiable para manejar datos de manera estructurada sin comprometer la integridad. Su soporte para transacciones complejas y su flexibilidad para adaptarse a distintos escenarios lo convierten en una gran elección. Aunque hay opciones más ligeras como SQLite o más populares en entornos web como MySQL, PostgreSQL nos brindó herramientas más avanzadas, aunque requirió mayor esfuerzo en su configuración inicial.

2. **¿Qué estándares o criterios usaron para diseñar su base de datos?**
   El enfoque que seguimos al diseñar la base de datos estuvo basado en la organización y optimización del almacenamiento de la información. Nos aseguramos de reducir redundancias mediante normalización y establecimos restricciones para evitar inconsistencias. Además, consideramos la estructura pensando en que, si en el futuro se necesitaban más funcionalidades, la base pudiera crecer sin necesidad de reestructurarla por completo.

3. **¿Cuáles son las entidades más importantes del modelo y por qué?**
   Para mí, las entidades clave son Usuario, Categoría, Transaction, Presupuesto y Notificación porque son el núcleo del sistema financiero que desarrollamos. Usuario es esencial porque conecta todo, mientras que Transaction y Categoría permiten llevar un control detallado de los movimientos económicos. Presupuesto y Notificación agregan valor al permitir que los usuarios administren sus finanzas de manera más organizada y reciban recordatorios importantes.

4. **¿Cómo aplicaron las técnicas de normalización en su diseño? ¿Qué problemas evitaron gracias a esto?**
   Consideramos la normalización como una estrategia clave para estructurar bien la información. Gracias a ella, evitamos datos duplicados y aseguramos que cualquier cambio en un registro se reflejara de forma consistente en toda la base. Sin este proceso, podríamos haber tenido problemas como datos repetidos en distintas tablas o dificultades al actualizar información importante.

5. **¿Cómo definieron restricciones y valores por defecto para garantizar la integridad de los datos?**
   Para garantizar que los datos fueran precisos y no hubiera errores en el ingreso, establecimos reglas que limitaban valores inválidos o inconsistentes. Por ejemplo, restringimos ciertos campos para que no quedaran vacíos y asignamos valores por defecto en lugares donde era necesario. También nos aseguramos de que la información se relacionara correctamente entre tablas, evitando registros huérfanos o datos que no tuvieran sentido dentro del sistema.

6. **¿Cómo abordaron los cambios en la estructura de la base de datos?**
   Cada vez que surgía la necesidad de hacer modificaciones en la base de datos, adoptamos un enfoque ordenado para implementar los cambios sin afectar lo que ya estaba construido. Documentamos cada ajuste y utilizamos scripts para aplicarlos progresivamente, asegurándonos de que cada nueva versión fuera compatible con la anterior. Esto nos permitió hacer mejoras sin comprometer la estabilidad del sistema.

7. **¿Cómo seleccionaron los datos de prueba para garantizar que el diseño es funcional?**
   Para comprobar que el modelo de base de datos funcionaba correctamente, diseñamos datos de prueba que simularan situaciones reales. Nos aseguramos de incluir registros que pusieran a prueba las reglas que habíamos definido, como transacciones con distintos montos y presupuestos con diferentes configuraciones. De esta forma, pudimos verificar que la base respondía de manera adecuada en cada caso.

8. **¿Cuál fue tu contribución específica en el desarrollo del proyecto? ¿Cómo se organizó el trabajo en el equipo?**

Mi participación se enfocó en estructurar la base de datos y definir cómo se relacionaban las distintas entidades. También trabajé en asegurar que las reglas de integridad estuvieran bien implementadas y en probar que los datos se almacenaban correctamente. La distribución de tareas en el equipo se hizo de manera equitativa, asegurándonos de que cada uno aportara desde su área de experiencia para lograr un mejor resultado.

9. **¿Sientes que trabajaste equitativamente en comparación con tus compañeros? ¿Qué hubieras hecho diferente en este proyecto?**
   Sí, creo que el esfuerzo fue balanceado entre todos. Cada quien asumió una parte del trabajo y nos apoyamos mutuamente para completar cada fase del proyecto. Si pudiera mejorar algo, me habría gustado dedicar más tiempo a optimizar las **consultas para** mejorar la eficiencia del sistema y asegurar que la documentación del proceso estuviera aún más detallada para futuras referencias.

Milton Polanco

1. **¿Por qué eligieron este sistema de gestión de bases de datos (DBMS)? ¿Qué ventajas y desventajas tiene en comparación con otros?**
   Elegimos PostgreSQL porque ofrece una robustez y flexibilidad excelentes para gestionar transacciones complejas y grandes volúmenes de datos. Su naturaleza de código abierto y la activa comunidad de soporte facilitan la resolución de problemas y la incorporación de nuevas funcionalidades. Sin embargo, reconocemos que, en comparación con opciones como SQLite o MySQL, PostgreSQL puede requerir una configuración inicial más detallada y mayores recursos de hardware.

2. **¿Qué estándares o criterios usaron para diseñar su base de datos?**
   Nuestro enfoque se basó en aplicar las mejores prácticas de normalización (hasta la tercera forma normal) para evitar redundancias y asegurar la integridad de la información. Además, consideramos criterios de escalabilidad y flexibilidad, estableciendo restricciones como NOT NULL y UNIQUE, y definiendo claves foráneas para garantizar que las relaciones entre las entidades se mantuvieran consistentes.

3. **¿Cuáles son las entidades más importantes del modelo y por qué?**
   Las entidades fundamentales para el sistema son:

- **Usuario:** Es la base del sistema, ya que todas las demás entidades se relacionan con él.
- **Categoría y Transaction:** Permiten clasificar y registrar de manera organizada los movimientos financieros.
- **Presupuesto y Notificación:** Son esenciales para ofrecer un control eficiente y alertas oportunas que ayuden a los usuarios a gestionar sus finanzas personales de forma proactiva.

4. **¿Cómo aplicaron las técnicas de normalización en su diseño? ¿Qué problemas evitaron gracias a esto?**
   Implementamos la normalización hasta la tercera forma normal, lo cual eliminó la duplicación de datos y redujo las dependencias innecesarias. Gracias a esta metodología se evitó la redundancia, se facilitó la actualización de la información y

se redujeron las inconsistencias, permitiendo que la base de datos fuese más eficiente y fácil de mantener.

5. **¿Cómo definieron restricciones y valores por defecto para garantizar la integridad de los datos?**
   Establecimos restricciones clave, como NOT NULL para campos esenciales, UNIQUE en el correo electrónico para prevenir duplicados, y claves foráneas para asegurar la integridad referencial entre tablas. Además, definimos valores por defecto en ciertos campos, lo que ayudó a mantener la coherencia y validez de los datos desde el momento de su inserción.

6. **¿Cómo abordaron los cambios en la estructura de la base de datos?**
   El manejo de cambios se realizó mediante un proceso de versiones y migraciones. Documentamos cada modificación en scripts SQL, lo que permitió una transición ordenada desde la versión inicial (v1.0) a la versión 1.1. Así se integraron de manera segura nuevas funcionalidades, como la tabla Notificación y el campo "periodo" en Presupuesto, sin comprometer la estabilidad del sistema.

7. **¿Cómo seleccionaron los datos de prueba para garantizar que el diseño es funcional?**
   Elegimos datos de prueba que reflejaran situaciones reales: incluimos transacciones de diferentes montos y tipos, presupuestos con variados periodos y notificaciones de alertas que cubrieran distintos escenarios. Esto nos permitió verificar que todas las relaciones y restricciones estuvieran correctamente implementadas y que el sistema respondiera adecuadamente ante diversas condiciones.

8. **¿Cuál fue su contribución específica en el desarrollo del proyecto? ¿Cómo se organizó el trabajo en el equipo?**
   Nuestra contribución se centró en la integración y validación de la base de datos. Nos encargamos de la implementación de los scripts DDL, la inserción de datos de prueba y la actualización de la documentación relacionada con las migraciones. Trabajamos de manera coordinada, asegurándonos de que cada tarea se complementara para lograr una solución integral y coherente.

9. **¿Sienten que trabajaron equitativamente en comparación con sus compañeros? ¿Qué hubieran hecho diferente en este proyecto?**
   Consideramos que nuestro aporte fue equilibrado y complementó de manera significativa el trabajo en equipo. De haber tenido la oportunidad, habríamos profundizado en la automatización de pruebas y en la optimización de consultas, además de ampliar la documentación de los procesos de migración para facilitar futuras actualizaciones y mantenimientos.