

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

Facultad de Ingeniería de Sistemas e Informática

Escuela Académica Profesional Ingeniería de Software



“Informe Final”

Autores:

- Bayona Vera, Elizabeth Ashley
- Giron Altamirano, Miguel Alejandro
- Melendez Cava, Andre Ivan
- Padilla Arellano Alejandro Manuel
- Torres Lezama, Mathias James

Profesor:

Chávez Soto, Jorge Luis

Curso:

Base de Datos II

Lima, Perú

2025

Presentación Técnica

El sistema SMART es una plataforma transaccional de alto rendimiento diseñada para la gestión y reserva de recintos temporales bajo un modelo de arquitectura centrado en la base de datos. A diferencia de las aplicaciones web convencionales que delegan la integridad de los datos a la capa de servidor, SMART centraliza la lógica de negocio, las reglas de validación y el control transaccional directamente en el motor de base de datos Oracle.

Esta decisión de diseño permite que el sistema opere como una "API de Base de Datos", exponiendo interfaces programáticas robustas (Paquetes PL/SQL) que garantizan que ninguna operación crítica (como una reserva o un pago) pueda eludir las reglas de negocio, independientemente de la interfaz de usuario que las invoque.

- **Ingeniería de Datos y Lógica Centralizada**

El núcleo del sistema no es un simple almacén de datos pasivo, sino un motor activo de procesamiento. La arquitectura se fundamenta en la encapsulación de procesos complejos mediante programación procedural en base de datos. Módulos críticos como la autenticación de usuarios, la búsqueda geoespacial de propiedades y el cálculo de disponibilidad en tiempo real se ejecuta nativamente dentro del motor, minimizando la latencia de red y maximizando el rendimiento mediante el uso de cursores optimizados y gestión eficiente de memoria.

- **Integridad y Automatización**

Para garantizar la consistencia en un entorno de alta concurrencia, el sistema implementa mecanismos de defensa en profundidad a nivel de datos. Se utilizan disparadores (Triggers) para la automatización de flujos de trabajo —como la generación automática de perfiles de usuario o la auditoría de cambios— y bloqueos pesimistas para prevenir condiciones de carrera en el proceso de reserva. Esto asegura que la integridad referencial y lógica del sistema se mantenga inalterable, ofreciendo una base sólida, segura y escalable para la operación comercial de anfitriones y huéspedes.

Objetivos técnicos del Trabajo Final

El objetivo principal de este proyecto es diseñar e implementar una arquitectura de base de datos robusta y escalable para la plataforma SMART (Sistema Multi-Tenant de Alquiler de Recintos Temporales), garantizando la integridad de los datos y el alto rendimiento mediante el uso avanzado de tecnologías Oracle.

Objetivos Específicos

- Centralización de la Lógica de Negocio: Implementar la lógica crítica (Reservas, Disponibilidad, Autenticación) directamente en la base de datos mediante Paquetes PL/SQL (BOOKING_PKG, AUTH_PKG, PROPERTY_PKG), reduciendo la latencia y garantizando consistencia independientemente del cliente que consuma los datos.
- Seguridad en Profundidad: Asegurar el sistema mediante una estrategia de capas que incluye perfiles de usuario (PRF_SMART), roles de privilegio mínimo (ROL_SMART) y auditoría activa de transacciones sensibles (Tabla AUDIT_LOGS).
- Optimización de Consultas: Desarrollar mecanismos de búsqueda eficientes para el filtrado de propiedades (FILTER_PKG) utilizando cursores de referencia (SYS_REFCURSOR) y optimización de índices para consultas geoespaciales y de rango de fechas.
- Integridad Transaccional: Garantizar propiedades ACID en procesos complejos, como la creación de reservas y pagos, implementando control de concurrencia pesimista para evitar conflictos de "doble reserva".

Resumen de funcionalidades, alcances y limitaciones de la base de datos

Funcionalidades Principales (Base de Datos)

- Gestión de Disponibilidad: Algoritmos capaces de calcular días libres vs. ocupados en tiempo real, considerando bloqueos de mantenimiento y reservas confirmadas (CALENDAR_AVAILABILITY_PKG).
- Búsqueda Avanzada: Procedimiento almacenado SP_SEARCH_PROPERTIES capaz de filtrar por ubicación, precio, amenities y disponibilidad de fechas en una sola ejecución.
- Dashboard de Anfitrión: Generación de estadísticas financieras y métricas de ocupación (HOST_DASHBOARD_PKG) calculadas directamente en el motor de base de datos.
- Sistema de Reseñas: Flujo automatizado mediante triggers (TRG_BOOKING_COMPLETED_REVIEWS) que habilita las reseñas solo cuando una estancia ha concluido.

Alcance

El proyecto abarca el diseño del Modelo Entidad-Relación (Lógico y Físico), la implementación de scripts DDL/DML en Oracle Database 21c/23c, y la creación de

una capa de API PL/SQL que sirve como backend para una aplicación web Next.js. Se incluye la gestión completa de usuarios, propiedades, reservas, pagos y mensajería.

Limitaciones

- **Gestión de Archivos:** Las imágenes de las propiedades no se almacenan como BLOBs en la base de datos para no afectar el rendimiento; se almacenan únicamente las referencias (URL) en la tabla `PROPERTY_IMAGES`.
- **Pasarela de Pagos:** La tabla `PAYMENTS` registra las transacciones con fines de consistencia y auditoría, pero el procesamiento bancario real se simula, ya que requiere integración con APIs externas (Stripe/PayPal) fuera del alcance de la base de datos.
- **Mensajería:** Aunque se diseñó un esquema para el chat (`MESSAGES`, `CONVERSATIONS`), el almacenamiento histórico de mensajes masivos podría migrar a una base de datos NoSQL en una fase futura para optimizar el almacenamiento de texto no estructurado.

Procesos de negocio

- **Gestión de Usuarios y Roles:** El sistema distingue entre un usuario base (**USERS**) y dos roles especializados: **TENANTS** (huéspedes) y **HOSTS** (anfitriones). El disparador `TRG_AUTO_CREATE_TENANT` automatiza la creación del perfil de huésped al registrar un usuario, lo que simplifica el *onboarding*.
- **Autenticación y Registro (Paquete `AUTH_PKG`):** Se manejan dos flujos: login/registro con credenciales (`SP_LOGIN_WITH_CREDENTIALS`, `SP_REGISTER_WITH_CREDENTIALS`) y login/registro con OAuth (`SP_FIND_OR_CREATE_USER_OAUTH`), garantizando un punto centralizado para la gestión de identidades (**`USER_AUTH_IDENTITYES`**).
- **Gestión de Propiedades (Paquete `PROPERTY_PKG`):** Proceso completo desde la creación (`SP_CREATE_PROPERTY`), modificación (`SP_UPDATE_PROPERTY`), hasta la

visualización detallada (SP_GET_PROPERTY_PAGE_DETAILS). Las propiedades (**PROPERTIES**) se asocian directamente a un Host.

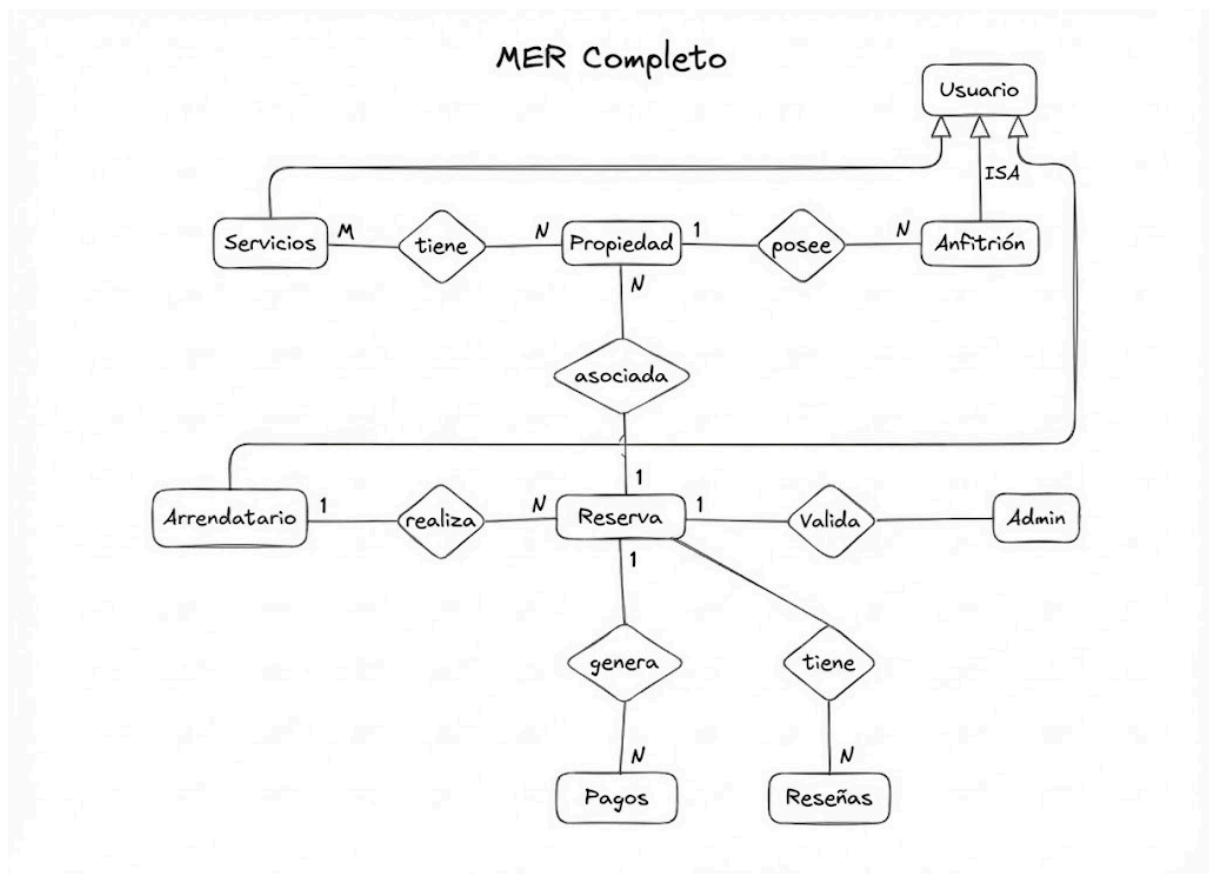
- **Disponibilidad y Calendario (AVAILABILITIES, Paquete PROPERTY_PKG, CALENDAR_AVAILABILITY_PKG):** El sistema permite al host definir rangos de fechas de bloqueo/precios especiales (SET_AVAILABILITY, SP_REMOVE_AVAILABILITY). La función **GET_CALENDAR** genera una vista día por día con precios y estados, manejando la lógica de disponibilidad.
- **Búsqueda y Filtros (Paquete FILTER_PKG):** Soporta búsquedas complejas por ubicación, precio, número de habitaciones/camas y, crucialmente, la validación de disponibilidad para un rango de fechas (SP_SEARCH_PROPERTIES).
- **Reservas (Paquete BOOKING_PKG):** Proceso de consulta de reservas por rol (get_bookings_by_tenant, get_bookings_by_host) y obtención de detalles (get_detailed_booking_info), que incluye una capa de seguridad para que solo el huésped o el host puedan ver la información.
- **Pagos (PAYMENTS, PAYMENT_DETAILS):** Existe una estructura para registrar pagos, su estado y el desglose de tarifas/comisiones.
- **Reseñas (REVIEWS, Paquete REVIEW_PKG):** El disparador **TRG_BOOKING_COMPLETED_REVIEWS** crea *placeholders* de reseña una vez la reserva se completa, forzando un flujo bidireccional de opinión (Host al Huésped, Huésped al Host/Propiedad). El paquete REVIEW_PKG gestiona el flujo de creación y consulta de reseñas.

Reglas de negocio

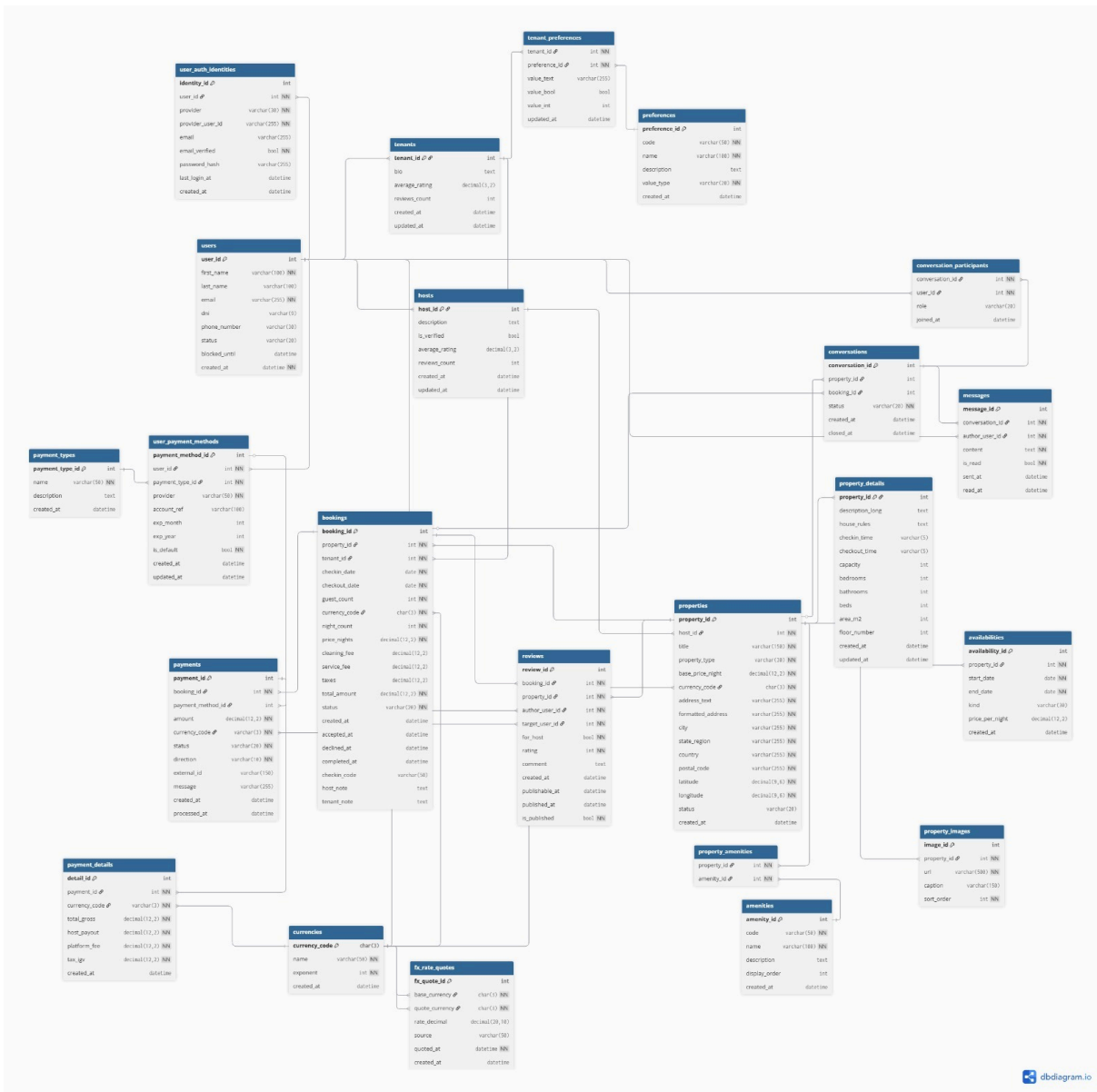
- **Rol Dual de Usuario:** **USERS** es la tabla maestra. Las tablas **HOSTS** y **TENANTS** usan **USER_ID** como clave primaria (Foreign Key a **USERS**), asegurando que un Host o Tenant es siempre un Usuario.
- **Creación Automática de Tenant:** **Automática de Tenant** El *trigger* **TRG_AUTO_CREATE_TENANT** inserta automáticamente una fila en **TENANTS** tras el registro de un nuevo usuario, haciéndolos Huéspedes por defecto.
- **Validación de Capacidad:** Las tablas **PROPERTY_DETAILS** y la lógica de búsqueda (**SP_SEARCH_PROPERTIES**) almacenan y validan capacidades máximas de ocupación (Adultos, Niños, Bebés, Mascotas).
- **Integridad de Reservas:** El procedimiento **SET_AVAILABILITY** (Regla 1) evita la creación de bloques de no-disponibilidad (**BLOCKED**, **MAINTENANCE**) que se solapen con reservas ya **CONFIRMED** (Error Code 1).
- **Asegurar la No-Solapamiento de Disponibilidad:** El procedimiento **SET_AVAILABILITY** (Regla 2) evita que un nuevo ajuste de disponibilidad se solape con otros ajustes ya existentes (Error Code 2).
- **Creación Bidireccional de Reseñas:** El *trigger* **TRG_BOOKING_COMPLETED_REVIEWS** crea automáticamente dos entradas en **REVIEWS** (una para que el Huésped califique, y otra para que el Host califique) cuando el estado de un *booking* cambia a **COMPLETED**.
- **Registro de Auditoría de Reservas:** El *trigger* **TRG_AUDIT_BOOKINGS** registra las operaciones de **INSERT**, **UPDATE** y **DELETE** en la tabla **AUDIT_LOGS** para mantener un historial de cambios en las reservas.

- **Comprobación de Privacidad de Reserva:** El procedimiento `get_detailed_booking_info` en `BOOKING_PKG` solo permite acceder a la información si el `P_USER_ID` es igual al `TENANT_ID` o al `HOST_ID` de la propiedad.

Modelo de Datos Conceptual



Modelo de Datos Lógico



Modelo de Datos Físico

A. Introducción al Modelo Físico

El Modelo Físico de Datos (MFD) es la traducción del modelo lógico a una implementación específica para el SGBD Oracle Database. Este modelo no solo define la estructura, sino que también optimiza el rendimiento y la integridad de los

datos. En esta transición, se han aplicado las convenciones de nomenclatura definidas por la consultora y se han definido los tipos de datos físicos de Oracle, índices, secuencias y restricciones de integridad.

B. Convenciones de Nomenclatura

Se ha adoptado una política de nomenclatura estricta y profesional, manteniendo todo el código y los objetos en idioma inglés, tal como se especificó en los requisitos del proyecto. Las convenciones principales son:

- Tablas: Nombres en plural, mayúsculas (ej. USERS, PROPERTIES).
- Columnas: Un prefijo de 3-4 letras de la tabla, guion bajo y el nombre del atributo (ej. USR_FIRST_NAME, PROP_TITLE).
- Claves Primarias (PK): Prefijo PK_ seguido del nombre de la tabla (ej. PK_USERS).
- Claves Foráneas (FK): Prefijo FK_ seguido de la tabla origen y la tabla destino (ej. FK_BOOKINGS_USERS).
- Índices: Prefijo IDX_ seguido de la tabla y columnas (ej. IDX_USERS_EMAIL).
- Secuencias: Prefijo SEQ_ seguido del nombre de la tabla (ej. SEQ_USERS).

C. Definiciones Físicas (Implementación en Oracle)

El MFD incluye las siguientes definiciones técnicas:

- Tipos de Datos Específicos: Se han ajustado los tipos de datos lógicos a los tipos nativos de Oracle. Por ejemplo:
 - varchar se convierte en VARCHAR2(n).
 - int se convierte en NUMBER(p, s).

- datetime se convierte en TIMESTAMP o DATE.
- bool se implementa como NUMBER(1) con una restricción CHECK (columna IN (0, 1)).
- text se convierte en CLOB para almacenar grandes volúmenes de texto (como descripciones o mensajes).
- Índices: Se han creado índices (B-Tree por defecto) en todas las claves foráneas para optimizar las operaciones de JOIN. Además, se han añadido índices en columnas de alta frecuencia de búsqueda, como USERS(EMAIL) o PROPERTIES(STATUS), para mejorar el rendimiento de las consultas.
- Secuencias y Disparadores (Triggers): Para manejar las claves primarias autoincrementales (ej. users_id), se ha creado una SEQUENCE de Oracle para cada tabla (SEQ_USERS). Se utiliza un disparador (trigger) BEFORE INSERT para asignar automáticamente el siguiente valor de la secuencia a la columna ID en cada nueva inserción.
- Integridad de Datos: Se han implementado todas las restricciones de integridad:
 - PRIMARY KEY para garantizar la unicidad de las filas.
 - FOREIGN KEY con la cláusula ON DELETE (ej. RESTRICT o SET NULL) para mantener la integridad referencial.
 - NOT NULL en columnas obligatorias.
 - UNIQUE para campos que no pueden repetirse (ej. USERS(EMAIL)).
 - CHECK para validar reglas de negocio a nivel de base de datos (ej. BOOKINGS(CHECK_OUT_DATE > CHECK_IN_DATE)).

A continuación, se presenta el diagrama del Modelo Físico de Datos (MFD). Este diagrama ilustra la estructura final de las tablas implementadas en Oracle Database, sus columnas con

tipos de datos específicos, las relaciones (claves foráneas), índices y secuencias, aplicando las convenciones de nomenclatura definidas. Representa la traducción técnica del modelo lógico. Para una mejor visualización y análisis detallado del diagrama, se puede consultar el archivo de imagen en alta resolución en el repositorio de GitHub del proyecto.



Relación de Objetos de Base de Datos:

El sistema SMART se estructura alrededor de cinco módulos principales: Usuarios y Autenticación, Propiedades, Gestión de Ubicaciones y Políticas, Reservas y Transacciones, y Comunicación y Feedback.

I. Módulo de Usuarios y Autenticación

El corazón del sistema es la tabla **users**, la cual almacena todos los perfiles de la plataforma (huéspedes, anfitriones y administradores). Cada usuario posee una tabla secundaria, **user_profile_details**, que mantiene una relación uno a uno (1:1) para guardar información extendida del perfil, como descripciones o fotos de perfil.

La seguridad y gestión de acceso se maneja mediante la tabla **permissions**, donde se definen las acciones posibles en el sistema (ej. `edit_property`, `read_booking`). Los permisos se asignan a los usuarios a través de una tabla pivote de muchos a muchos (M:N) denominada **user_permissions**, vinculando `user_id` con `permission_id`. Finalmente, la gestión de sesiones y seguridad incluye la tabla **user_tokens** para manejar tokens de acceso y restablecimiento de contraseñas.

II. Módulo de Propiedades y Recintos

La tabla central de este módulo es **properties**, que contiene toda la información de un alojamiento o recinto publicado. Cada propiedad está inherentemente ligada a un anfitrión a través de una clave foránea (`host_id`) que apunta a la tabla **users** (relación N:1).

Los detalles geográficos se separan en dos tablas para organización:

1. **locations**, que almacena ciudades o regiones para facilitar la búsqueda. La tabla **properties** tiene una relación N:1 con **locations**.
2. **addresses**, que detalla la dirección física exacta y mantiene una relación 1:1 con **properties**.

La funcionalidad de la propiedad se complementa con tablas auxiliares:

- **amenities** lista los servicios disponibles (Wi-Fi, Piscina). La relación con **properties** es de muchos a muchos (M:N) y se gestiona mediante la tabla pivote **property_amenities**.
- **property_images** (N:1 con **properties**) almacena las URLs de las fotografías.
- **property_rules** (N:1 con **properties**) define las normas de convivencia del recinto.
- **property_availability** (N:1 con **properties**) gestiona los rangos de fechas bloqueados por el anfitrión para evitar reservas no deseadas.

III. Módulo de Reservas y Transacciones

La tabla **bookings** es el registro fundamental de cada reserva. Mantiene una relación N:1 con la tabla **properties** y una relación N:1 con la tabla **users** (el inquilino que realiza la reserva). Además, para la gestión de políticas, la tabla **properties** está enlazada con **cancellation_policies** (N:1), definiendo las reglas de reembolso.

El aspecto financiero se maneja con la tabla **booking_transaction**, que registra el pago asociado a una reserva en una relación uno a uno (1:1) con **bookings**. Para el pago a los anfitriones, existen dos tablas: **host_payment_details**, que guarda la información bancaria del anfitrión (1:1 con **users Host**), y **host_payout_history**, que registra el historial de pagos realizados al anfitrión (N:1 con **users Host** y N:1 con **bookings**).

IV. Módulo de Comunicación y Feedback

El feedback del sistema se almacena en la tabla **reviews**. Esta tabla tiene claves foráneas que la relacionan con la **properties** (sobre qué se reseña) y con la tabla **users** (para identificar al revisor y al usuario revisado).

Finalmente, la comunicación interna se soporta con:

- **messages**, para la mensajería directa entre usuarios.
- **notification_types**, que categoriza las alertas del sistema.
- **notifications**, que envía alertas a los usuarios y se relaciona N:1 con **users** y **notification_types**.

Esquema de Base de Datos:

- **Scripts de generación de esquemas de Base de Datos.**
https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/Segundo_entregable/Esquema_Base_Datos/01_crear_esquema.sql
- **Scripts de generación de objetos de Base de Datos.**
<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/ScriptCreacionBD.sql>
- **Scripts de Creación de Objetos de Programación almacenados.**
<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/ScriptObjetosProgramaci%C3%B3n.sql>
- **Scripts de carga de Datos.**
<https://github.com/Matrix1102/Informes-proyecto-G6/tree/main/ScriptsCargaDatos>

- **Scripts de transacciones de Base de Datos.**

<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/ScriptTransaccionalidad.sql>

- **Scripts de Control de Concurrencia.**

<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/SMART/ControlConcurrencia.sql>

- **Base de Datos implementada y con datos de prueba.**

<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/ArchivoExportacionTotal.sql>

- **Scripts de Inyección de SQL**

<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/SMART/PruebaInyeccionSQL.sql>

- **Scripts de Backup y Restore de Base de Datos**

<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/SMART/BackupRespaldo.sql>

- **Scripts de auditoría de transacciones.**

<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/SMART/AuditoriaTransacciones.sql>

- **Scripts de auditoria de Base de Datos**

<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/SMART/AuditoriaBaseDatos.sql>

- **Gestión de Usuarios**

<https://github.com/Matrix1102/Informes-proyecto-G6/blob/main/SMART/GestionUsuarios.sql>

