Adrián Solís León 2ºDAM

Proyecto Hibernate

Voy a hacer el proyecto de Hibernate siguiendo el proyecto que estoy haciendo con Javi, que es sobre una aplicación de motos.

Aplicación para crear un perfil único para su moto, registra todo de la moto que le ocurra. Con esto podemos verificar cuando vayamos a comprar una moto todo lo que haya pasado, tantos km, modificaciones, accidentes, rutas… y todo verificado para evitar estafas. Tendrá cada moto un QR único donde poder verificar los datos de ella, habrá logros y recompensas por km, mantenimiento, rutas especiales que te aconseje la aplicación hacer dependiendo del tipo de moto. También podrás compartir tus datos en la app con los demás usuarios. Hay apps de ruta y comunidad, como MyRide, otras de mantenimiento como Drivvo y otras de seguridad como MotoSafe, pero ninguna que lo englobe todo en una.

1. Entidad: Usuario

Representa a cualquier persona registrada: usuario, taller o admin.

Atributos:

id_usuario (PK)
nombre
apellidos
email (único)
teléfono
contraseña_hash
rol (usuario / taller / admin)
verificado (boolean)
fecha_registro

2. Entidad: Moto

Cada moto registrada por un usuario.

Atributos:

id_moto (PK)
marca
modelo
matrícula (única)
cilindrada
itv_fecha
qr_codigo (único)
id_propietario (FK → Usuario)
RELACIÓN: Usuario — Moto

Usuario (1) —— (N) Moto

Un usuario puede tener varias motos.
Cada moto solo tiene un propietario activo.

Cardinalidad: 1:N

Adrián Solís León 2ºDAM

3. Entidad: Evento

Un evento registrado en la moto (reparación, mantenimiento, accidente, ITV, ruta, etc.)

Atributos

id_evento (PK)
tipo_evento
descripción
fecha_evento
km
es_validado (boolean)
fecha_validacion
id_moto (FK → Moto)
id_usuario_creador (FK → Usuario)
id_taller_validador (FK → Taller) (opcional)

RELACIONES DE EVENTO
Moto (1) —— (N) Evento

Una moto tiene muchos eventos.
Cada evento pertenece a una moto.

Usuario (1) —— (N) Evento (creador)

Un usuario puede registrar muchos eventos.
Cada evento tiene un solo creador.

Taller (1) —— (N) Evento (validador)

Un taller puede validar muchos eventos.
Un evento tiene 0 o 1 taller validador.

Cardinalidad:

Moto → Evento = 1:N
Usuario → Evento = 1:N
Taller → Evento = 1:N

4. Entidad: Taller

Talleres profesionales autorizados para validar eventos.

Atributos

id_taller (PK)
nombre_taller
dirección
telefono
cif
estado_verificación
id_usuario (FK → Usuario)

RELACIÓN: Usuario — Taller

Adrián Solís León 2ºDAM

Usuario (1) —— (1) Taller

Cada taller tiene una sola cuenta de usuario.
Cada usuario tipo taller corresponde con un taller.

Cardinalidad: 1:1

5. Entidad: Transferencia

Registro de cambio de propietario (compraventa).

Atributos

id_transferencia (PK)
id_moto (FK → Moto)
id_prop_anterior (FK → Usuario)
id_prop_nuevo (FK → Usuario)
fecha_transferencia
estado

RELACIONES DE TRANSFERENCIA
Moto (1) —— (N) Transferencia

Una moto puede tener muchas transferencias a lo largo de su vida.

Usuario (1) —— (N) Transferencia (como viejo propietario)

Un usuario puede vender muchas motos.

Usuario (1) —— (N) Transferencia (como nuevo propietario)

Un usuario puede comprar muchas motos.

6. Entidad: Ruta

Representa una ruta oficial o comunitaria que puede ser realizada por distintas motos.

Atributos:

id_ruta (PK)
nombre
distancia_km
dificultad
tipo_terreno
descripcion
es_oficial (boolean)

RELACIÓN: Moto — Ruta

Moto (N) —— (M) Ruta

Una moto puede realizar muchas rutas.
Una ruta puede ser realizada por muchas motos.

Adrián Solís León 2ºDAM

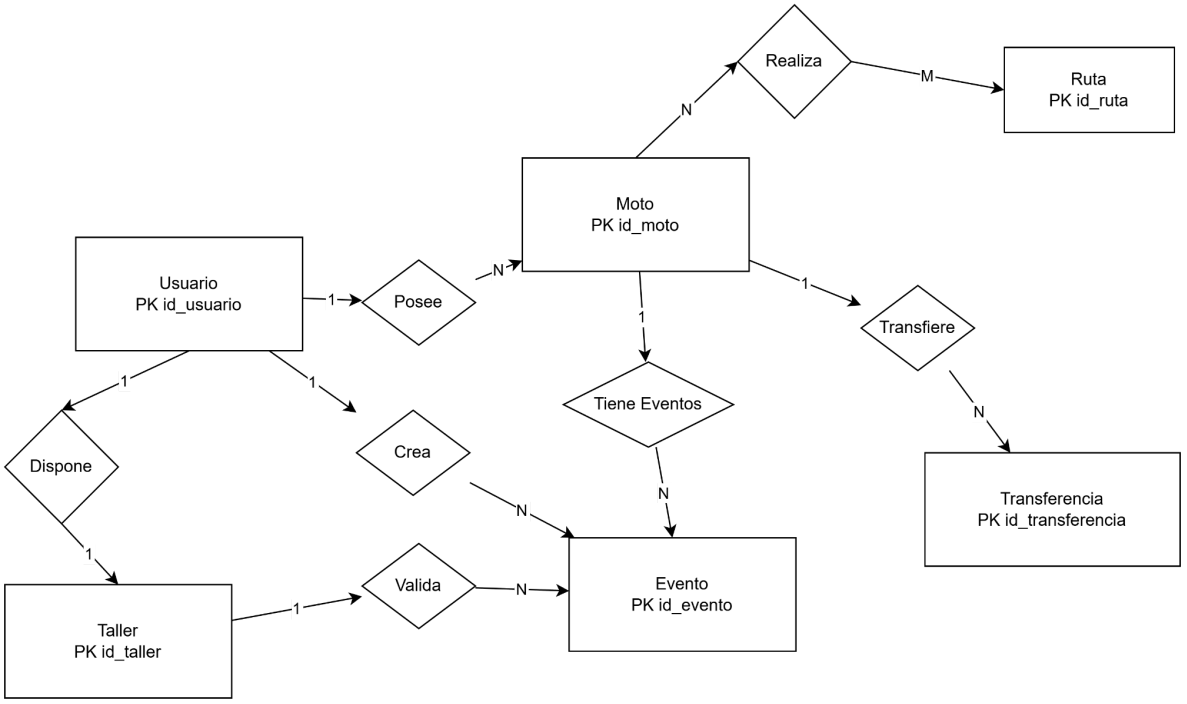

Imagen de como quedaría el Modelo E/R

Explicación de Consultas Implementadas (Sprint 2):

HQL Eficiente: Se implementó una consulta que devuelve sólo las matrículas (String) para optimizar la memoria al cargar listados.

Agregación: Usamos AVG en Rutas para calcular la media de dificultad y COUNT en Eventos para las estadísticas del perfil.

CriteriaBuilder: Se ha implementado la actualización de nombres y borrado de usuarios no verificados de forma programática, lo que permite que la aplicación sea escalable y fácil de mantener si el modelo de datos cambia en el futuro.