Modelo Relacional para Passkeeper (Sitio Web)

Entidades:

➔ **User:**

◆ **Atributos:**

- `idUser` (INT AUTO_INCREMENT PRIMARY KEY): Identificador único del usuario.
- `email` (VARCHAR(50) UNIQUE NOT NULL): Correo electrónico del usuario.
- `password` (VARCHAR(50) NOT NULL): Contraseña del usuario (debe cumplir con los requisitos de seguridad: mínimo 8 caracteres, una minúscula, una mayúscula y números).
- `is_staff` (TINYINT): Indica si el usuario es miembro del personal (administrador): 1 (true) o 0 (false).
- `is_premium` (BOOLEAN NOT NULL DEFAULT FALSE): Indica si el usuario tiene una suscripción premium: 1 (true) o 0 (false).

➔ **Payment:**

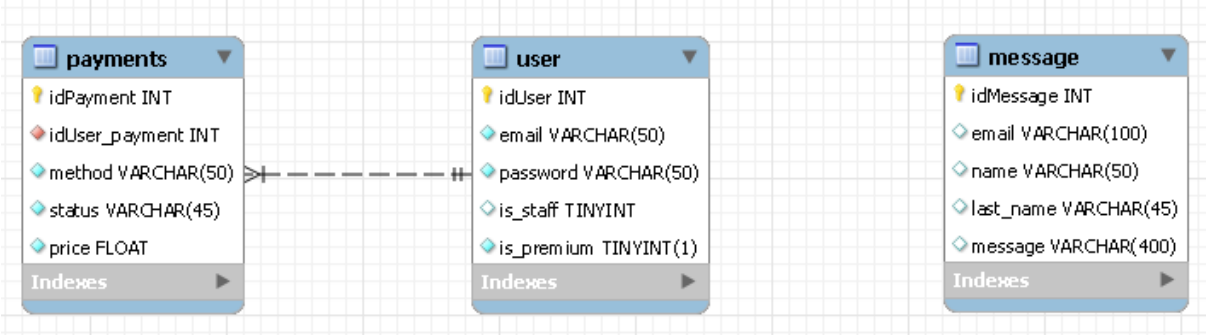
◆ **Atributos:**

- `idPayment` (INT PRIMARY KEY AUTO_INCREMENT): Identificador único del pago.
- `idUser_payment` (INT NOT NULL): Identificador del usuario asociado al pago.
- `method` (VARCHAR(50) NOT NULL): Método de pago utilizado (por ejemplo, tarjeta de crédito, PayPal).
- `status` (VARCHAR(45) NOT NULL): Estado del pago (por ejemplo, pendiente, completado, cancelado).
- `price` (FLOAT NOT NULL): Precio del pago.

➔ **Message:**

◆ **Atributos:**

- `idMessage (INT PRIMARY KEY AUTOINCREMENT): Identificador del mensaje enviado.`
- `email (VARCHAR(100)): Correo electronico del usuario.`
- `name (VARCHAR (50)): Nombre del usuario.`
- `last_name (VARCHAR(50)): Apellido del usuario.`
- `message(VARCHAR(400)): Contenido del mensaje enviado por el usuario.`



<u>Relaciones:</u>

- **User (1:N) Payments:** Un usuario puede tener varios pagos, pero un pago solo pertenece a un usuario.
    - Se implementa mediante una clave foránea `idUser_payment` en la tabla `Payments` que hace referencia a la clave primaria `idUser` en la tabla `User`.

Consideraciones de seguridad:

- **Almacenamiento de contraseñas:** Las contraseñas nunca deben almacenarse en texto plano. Se recomienda utilizar un algoritmo de hash seguro como bcrypt o PBKDF2 para almacenar hashes de contraseñas.
- **Validación de entrada:** Se debe validar la entrada del usuario para garantizar que las contraseñas cumplan con los requisitos mínimos de seguridad (mínimo 8 caracteres, una minúscula, una mayúscula y números).
- **Prevención de ataques de inyección SQL:** Se deben utilizar consultas SQL parametrizadas para evitar ataques de inyección SQL.
- **Protección contra ataques XSS:** Se debe escapar el código HTML del usuario para evitar ataques XSS.

Implementación en SQL:

SQL

```
CREATE DATABASE passkeeper;
USE passkeeper;

CREATE TABLE User (
  idUser INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(50) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  is_staff TINYINT,
  is_premium BOOLEAN NOT NULL DEFAULT FALSE
);

CREATE TABLE Payments (
  idPayment INT PRIMARY KEY AUTO_INCREMENT,
  idUser_payment INT NOT NULL,
  FOREIGN KEY (idUser_payment) REFERENCES User(idUser),
  method VARCHAR(50) NOT NULL,
  status VARCHAR(45) NOT NULL,
  price FLOAT NOT NULL
);

CREATE TABLE Message (
idMessage INT AUTO_INCREMENT PRIMARY KEY,
email VARCHAR(100),
name VARCHAR(50),
```

```
last_name VARCHAR(45),
message VARCHAR(400)
)
```

Nota:

- Este modelo relacional es un punto de partida y puede requerir modificaciones adicionales según los requisitos específicos de la aplicación PassKeeper.
- Es importante implementar las medidas de seguridad adecuadas para proteger los datos de los usuarios.