Minchapp

Tablas en PostgreSQL

```
CREATE TABLE Usuario (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(256) NOT NULL,
    apellido VARCHAR(256) NOT NULL,
    fechaNacimiento DATE,
    ciudadResidencia VARCHAR(256),
    urlImagenPerfil VARCHAR(512),
    telefono VARCHAR(16),
    email VARCHAR(256),
    contrasena BYTEA
);

-- Definimos diferentes roles como "Cuidador", "Host", "Administrador"
CREATE TABLE TipoUsuario (
    id SERIAL PRIMARY KEY,
    descripcion VARCHAR(128) NOT NULL  -- Ejemplo: 'Cuidador', 'Host',
'Administrador'
);

-- Esta tabla permitirá la asignación de múltiples roles a un mismo usuario.
CREATE TABLE UsuarioTipo (
    idUsuario INTEGER REFERENCES Usuario(id),
    idTipoUsuario INTEGER REFERENCES TipoUsuario(id),
    PRIMARY KEY (idUsuario, idTipoUsuario)
);

CREATE TABLE InfoUsuario (
    idUsuario INTEGER REFERENCES Usuario(id),
    cedula VARCHAR(64),
    hojaDelincuencia BOOLEAN,
    PRIMARY KEY (idUsuario)
);

-- DIRECCION
CREATE TABLE Pais (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(128) NOT NULL
);

CREATE TABLE Estado (
    id SERIAL PRIMARY KEY,
    idPais INTEGER REFERENCES Pais(id),
    nombre VARCHAR(128) NOT NULL
);

CREATE TABLE Ciudad (
```

```
    id SERIAL PRIMARY KEY,
    idEstado INTEGER REFERENCES Estado(id),
    nombre VARCHAR(128) NOT NULL
);

CREATE TABLE Direccion (
    id SERIAL PRIMARY KEY,
    idCiudad INTEGER REFERENCES Ciudad(id),
    calle1 VARCHAR(256),
    calle2 VARCHAR(256),
    codigoPostal VARCHAR(16),
    latitud DECIMAL(9,6),
    longitud DECIMAL(9,6)
);
---

CREATE TABLE Contacto (
    id SERIAL PRIMARY KEY,
    idUsuario INTEGER REFERENCES Usuario(id),
    tipoContacto VARCHAR(64),  -- Ejemplo: 'personal', 'emergencia'
    nombre VARCHAR(256),
    numeroContacto VARCHAR(16),
    email VARCHAR(256)
    deleted BOOLEAN DEFAULT FALSE  -- Para eliminaciones lógicas
);

CREATE TABLE TipoPlataforma (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(128) NOT NULL
);

CREATE TABLE RedSocial (
    id SERIAL PRIMARY KEY,
    idUsuario INTEGER REFERENCES Usuario(id),
    idPlataforma INTEGER REFERENCES TipoPlataforma(id),
    urlPerfil VARCHAR(512)
);

-- Se une depósitos de garantía y pagos en una sola tabla
CREATE TABLE Transaccion (
    id SERIAL PRIMARY KEY,
    idUsuario INTEGER REFERENCES Usuario(id),
    fecha TIMESTAMP DEFAULT NOW(),
    monto DECIMAL(10, 2),
    descripcion TEXT,
    tipo VARCHAR(64),  -- Ejemplo: 'depósito', 'pago'
    numeroReferencia VARCHAR(64),
    checksum VARCHAR(64)
);

-- Una bitácora para registros de contacto y de cuidados con los cuidadores.
CREATE TABLE Bitacora (
    id SERIAL PRIMARY KEY,
    idUsuario INTEGER REFERENCES Usuario(id),
```

```
    idCuidador INTEGER REFERENCES Usuario(id),
    tipo VARCHAR(64),  -- Ejemplo: 'contacto', 'cuidados'
    fecha TIMESTAMP DEFAULT NOW(),
    observaciones TEXT,
    checksum VARCHAR(64)
);

CREATE TABLE Favorito (
    id SERIAL PRIMARY KEY,
    idUsuario INTEGER REFERENCES Usuario(id),
    idCuidador INTEGER REFERENCES Usuario(id)
);

CREATE TABLE ProtocolosEmergencia (
    id SERIAL PRIMARY KEY,
    idInfoCasa INTEGER REFERENCES InfoCasa(id),
    situacionEmergencia TEXT,
    solucion TEXT
);

CREATE TABLE ServiciosAdicionales (
    id SERIAL PRIMARY KEY,
    idUsuario INTEGER REFERENCES Usuario(id),
    descripcion TEXT NOT NULL,
    deleted BOOLEAN DEFAULT FALSE  -- Campo para eliminaciones lógicas
);
```

Tablas MongoDB

```
import mongoose, { Schema, Document } from 'mongoose';

// Interfaz para el modelo `Post`
interface IPost extends Document {
  idUsuario: number;
  motivo: string;
  idInfoCasa: number;
  ofertaPago: number;
  fechaInicio: Date;
  fechaFin: Date;
  estadoReservado: boolean;
  deleted: boolean;  // Eliminación lógica
}

// Esquema de `Post`
const PostSchema = new Schema<IPost>({
  idUsuario: { type: Number, required: true },
  motivo: { type: String, required: true },
  idInfoCasa: { type: Number, required: true },
  ofertaPago: { type: Number, required: true },
  fechaInicio: { type: Date, required: true },
```

```
  fechaFin: { type: Date, required: true },
  estadoReservado: { type: Boolean, default: false },
  deleted: { type: Boolean, default: false }
});

// Crear modelo
const PostModel = mongoose.model<IPost>('Post', PostSchema);

export { PostModel };
```

```
import mongoose, { Schema, Document, model } from 'mongoose';

// Interfaz para el modelo InfoCasa
interface IInfoCasa extends Document {
  idUsuario: mongoose.Types.ObjectId;  // Relación con el Usuario
  descripcionBase: string;
  numHabitaciones: number;
  numBanos: number;
  piscina: boolean;
  jardin: boolean;
  mascotas: boolean;
  caracteristicasHabitaciones: Array<{
    nombre: string;
    valor: string;
  }>;
}

// Esquema de InfoCasa
const InfoCasaSchema = new Schema<IInfoCasa>({
  idUsuario: { type: mongoose.Types.ObjectId, ref: 'Usuario', required: true },
  descripcionBase: { type: String, required: true },
  numHabitaciones: { type: Number, required: true },
  numBanos: { type: Number, required: true },
  piscina: { type: Boolean, default: false },
  jardin: { type: Boolean, default: false },
  mascotas: { type: Boolean, default: false },
  caracteristicasHabitaciones: [
    {
      nombre: { type: String, required: true }, // Ej: 'tipoCama', 'tamaño',
'vistas'
      valor: { type: String, required: true }   // Ej: 'King Size', '30 m²',
'Vista al mar'
    }
  ]
});

// Crear el modelo
const InfoCasaModel = model<IInfoCasa>('InfoCasa', InfoCasaSchema);
```

```
export { InfoCasaModel };
```