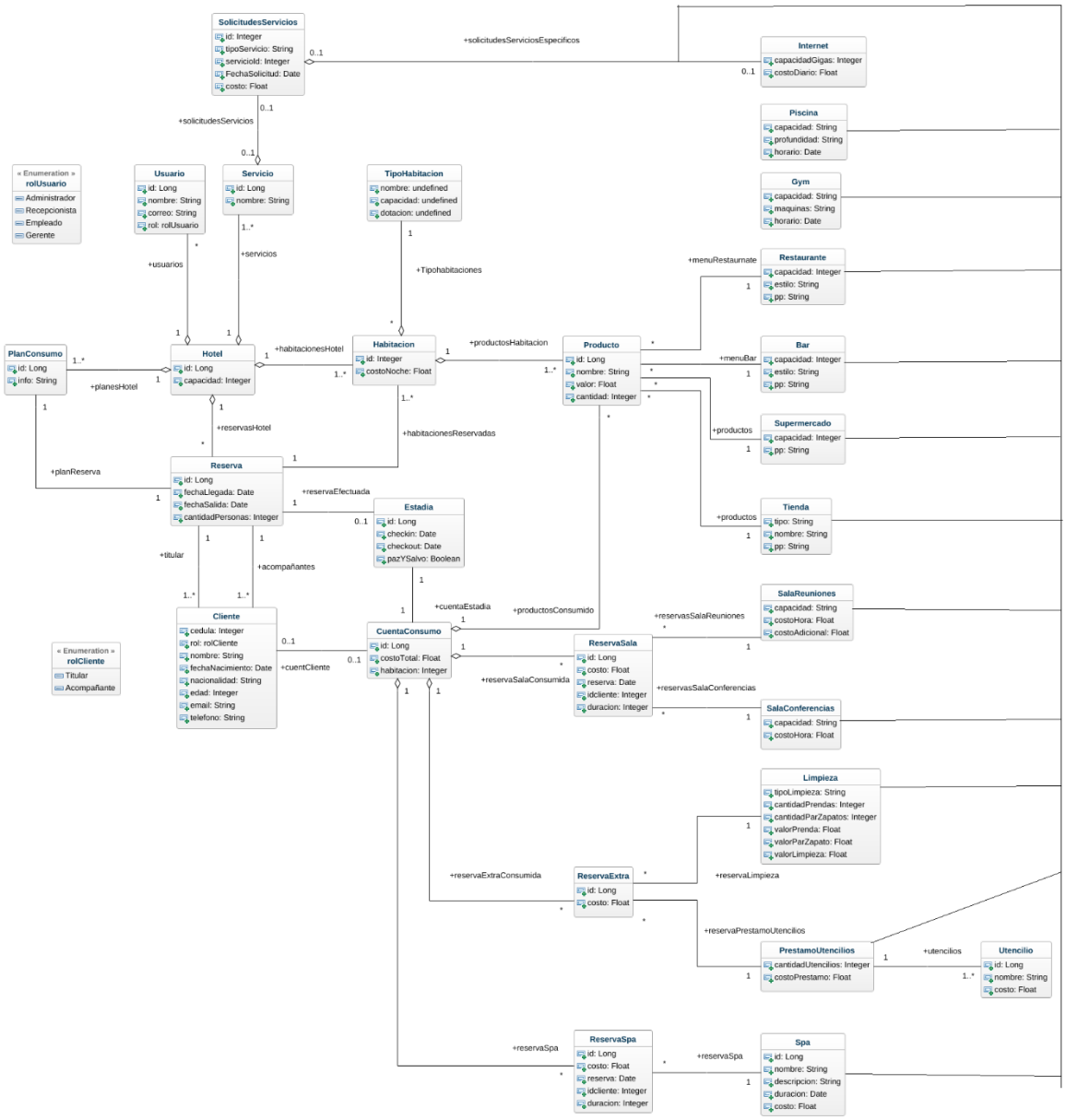
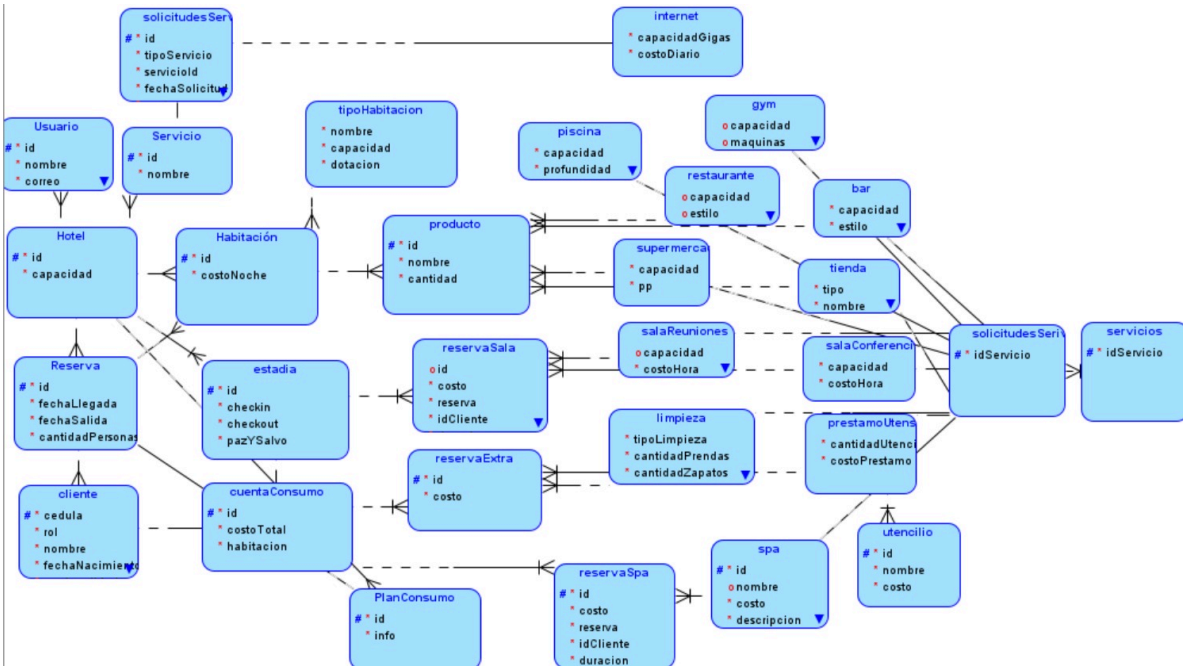
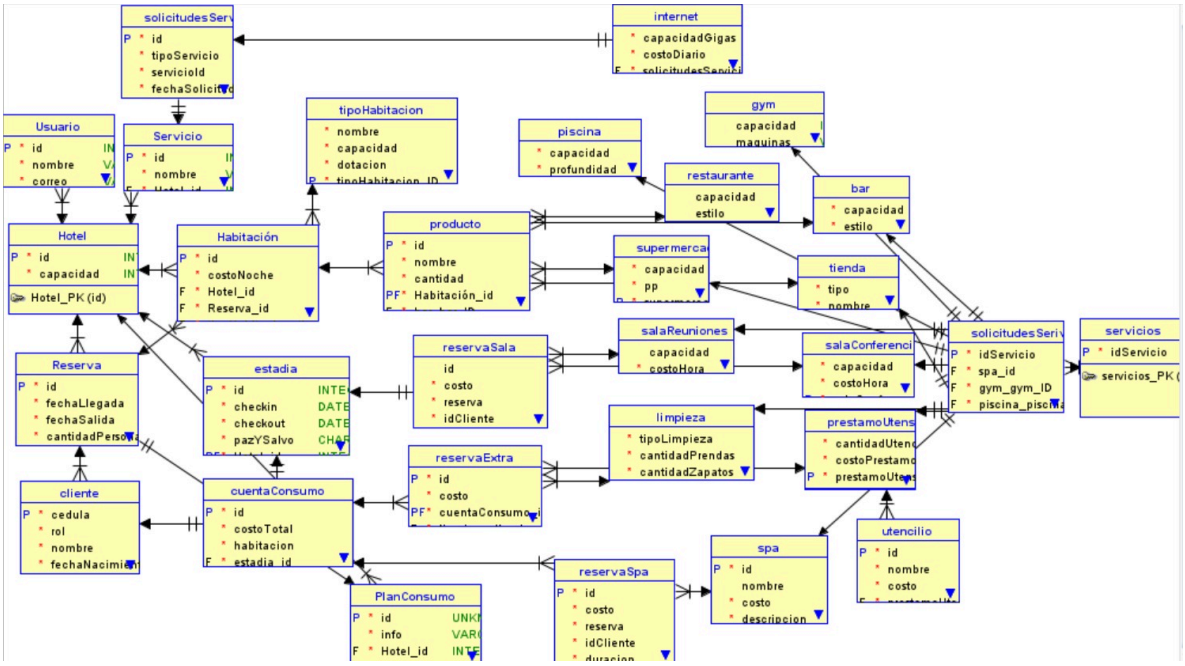
Documentacion Proyecto 3 SISTRANS

1. UML



2. E/R


solicitudesServ
* id
* tipoServicio
* servicioId
* fechaSolicitud
internet
* capacidadGigas
* costoDiario
Usuario
* id
* nombre
* correo
Servicio
* id
* nombre
tipoHabitacion
* nombre
* capacidad
* dotacion
piscina
* capacidad
* profundidad
gym
o capacidad
o maquinas
restaurante
o capacidad
o estilo
bar
* capacidad
* estilo
Hotel
* id
* capacidad
Habitación
* id
* costoNoche
producto
* id
* nombre
* cantidad
supermerca
* capacidad
* pp
tienda
* tipo
* nombre
salaReuniones
o capacidad
* costoHora
salaConferenci
* capacidad
* costoHora
solicitudesSeri
* idServicio
servicios
* idServicio
Reserva
* id
* fechaLlegada
* fechaSalida
* cantidadPersonas
estadia
* id
* checkin
* checkout
* pazYSalvo
reservaSala
o id
* costo
* reserva
* idCliente
limpieza
* tipoLimpieza
* cantidadPrendas
* cantidadZapatos
prestamoUtens
* cantidadUtenci
* costoPrestamo
reservaExtra
* id
* costo
cliente
* cedula
* rol
* nombre
* fechaNacimient
cuentaConsumo
* id
* costoTotal
* habitacion
spa
* id
o nombre
* costo
* descripcion
utencilio
* id
* nombre
* costo
PlanConsumo
* id
* info
reservaSpa
* id
* costo
* reserva
* idCliente
* duracion


solicitudesServ
P * id
* tipoServicio
* servicioId
* fechaSolicitud
internet
* capacidadGigas
* costoDiario
Usuario
P * id
* nombre
* correo
Servicio
P * id
* nombre
tipoHabitacion
* nombre
* capacidad
* dotacion
piscina
* capacidad
* profundidad
gym
capacidad
maquinas
restaurante
capacidad
estilo
bar
* capacidad
* estilo
Hotel
P * id
* capacidad
Hotel_PK (id)
Habitación
P * id
* costoNoche
F * Hotel_id
F * Reserva_id
producto
P * id
* nombre
* cantidad
PF * Habitación_id
supermerca
* capacidad
* pp
tienda
* tipo
* nombre
salaReuniones
capacidad
* costoHora
salaConferenci
* capacidad
* costoHora
solicitudesSeriv
P * idServicio
F * spa_id
F * gym_gym_ID
F * piscina_pisci
servicios
P * idServicio
servicios_PK (
Reserva
P * id
* fechaLlegada
* fechaSalida
* cantidadPerso
estadia
P * id
* checkin
* checkout
* pazYSalvo
reservaSala
id
* costo
* reserva
* idCliente
limpieza
* tipoLimpieza
* cantidadPrendas
* cantidadZapatos
prestamoUtens
* cantidadUtenc
* costoPrestamo
* prestamoUtens
reservaExtra
P * id
* costo
PF * cuentaConsumo
cliente
P * cedula
* rol
* nombre
* fechaNacimien
cuentaConsumo
P * id
* costoTotal
* habitacion
F * estadia_id
spa
P * id
nombre
* costo
* descripcion
utencilio
P * id
* nombre
* costo
PlanConsumo
P * id
* info
F * Hotel_id
reservaSpa
P * id
* costo
* reserva
* idCliente
* duracion

3. **Elección de embebido vs referenciado:**

1. Tipo de Habitación (embebido en Habitación):

- ¿Por qué embebido? Un Tipo de Habitación (como 'normal', 'suite', etc.) es una característica intrínseca de la Habitación y raramente cambia. Al embeber el Tipo de Habitación directamente en el documento de Habitación, se facilita la recuperación de toda la información de la habitación en una sola consulta, sin necesidad de realizar joins o referencias adicionales. Esto es eficiente para operaciones de lectura.

- Ventajas: Rápido acceso a datos relacionados sin necesidad de consultas adicionales. Reducción de la complejidad en las consultas y en la lógica de la aplicación.

- Consideraciones: Si los detalles del Tipo de Habitación cambian (lo cual es raro), se deben actualizar todos los documentos de Habitación que contienen esa información. Esto no es óptimo para datos que cambian con frecuencia.

2. Reserva (referenciando Tipo de Habitación y Habitación):

- ¿Por qué referenciado? Las reservas son transacciones que ocurren en un momento específico y tienen una relación directa tanto con el Tipo de Habitación como con la Habitación específica. Referenciar permite mantener la integridad de los datos sin duplicar información. Si una habitación o su tipo se actualizan, no hay necesidad de actualizar cada reserva; simplemente se sigue la referencia para obtener la información actual.

- Ventajas: Mantiene la consistencia de los datos y evita la duplicación. Facilita la actualización de los datos de las habitaciones y sus tipos, ya que no se requiere actualizar múltiples reservas.

- Consideraciones: Las consultas pueden ser ligeramente más complejas, ya que requieren seguir referencias para obtener información completa.

3. Estadías (referenciando Reserva):

- ¿Por qué referenciado? Las estancias son eventos que se derivan directamente de una reserva. Referenciar la reserva en una estancia permite mantener una clara relación entre la estancia y todos los detalles de la reserva (incluyendo la habitación y el tipo de habitación). Esto garantiza que cualquier cambio en la reserva se refleje automáticamente en la información de la estancia.

- Ventajas: Se mantiene un enlace claro entre la estancia y su reserva correspondiente. Las actualizaciones en la reserva se reflejan en la estancia sin duplicación de datos.

- Consideraciones: Al igual que con las reservas, las consultas para recuperar información completa de estancias pueden requerir seguir múltiples referencias.

4. **Ejmplo documentos:**

```
db.tipoHabitacion.insertOne({
  "_id": "1",
  "nombre": "Suite Presidencial",
  "capacidad": 4,
  "dotacion": "Cama King, Baño con Jacuzzi, Vista al mar"
})
```

```
db.habitacion.insertOne({
    "_id": "1",
    "costo": 250,
    "tipoHabitacion": db.tipoHabitacion.findOne({ "_id": "1" })
})
```

```
db.reserva.insertOne({
    "_id": "3",
    "fechaLlegada": ISODate("2023-12-24T00:00:00Z"),
    "fechaSalida": ISODate("2023-12-31T00:00:00Z"),
    "tipoHabitacion": db.tipoHabitacion.findOne({ "_id": "1" }),
    "titular": "Juan Pérez",
    "habitacion": db.habitacion.findOne({ "_id": "3" })
})
```

```
db.estadia.insertOne({
  "_id": "1",
  "fechaCheckIn": ISODate("2024-01-01T00:00:00Z"),
  "fechaCheckOut": ISODate("2024-01-07T00:00:00Z"),
  "checkInRealizado": false,
  "checkOutRealizado": false,
  "reserva": db.reserva.findOne({ "_id": "3" })
})
```

5. **Creación de las colecciones:**

```
db.createCollection("tipoHabitacion")
db.createCollection("habitacion")
db.createCollection("reserva")
db.createCollection("estadia")
```

6. **Validaciones esquema**

```
const tipoHabitacionSchema = new mongoose.Schema({
  _id: String,
  nombre: String,
  capacidad: Number,
  dotacion: String,
});

const TipoHabitacion = mongoose.model('TipoHabitacion', tipoHabitacionSchema);
```

```
const habitacionSchema = new mongoose.Schema({
  _id: String,
  costo: Number,
  tipoHabitacion: {
    type: String,
    ref: 'TipoHabitacion',
  },
});

const Habitacion = mongoose.model('Habitacion', habitacionSchema);
```

```
const reservaSchema = new mongoose.Schema({
  _id: String,
  fechaLlegada: Date,
  fechaSalida: Date,
  tipoHabitacion: {
    type: String,
    ref: 'TipoHabitacion',
  },
  titular: String,
  habitacion: {
    type: String,
    ref: 'Habitacion',
  },
});

const Reserva = mongoose.model('Reserva', reservaSchema);
```

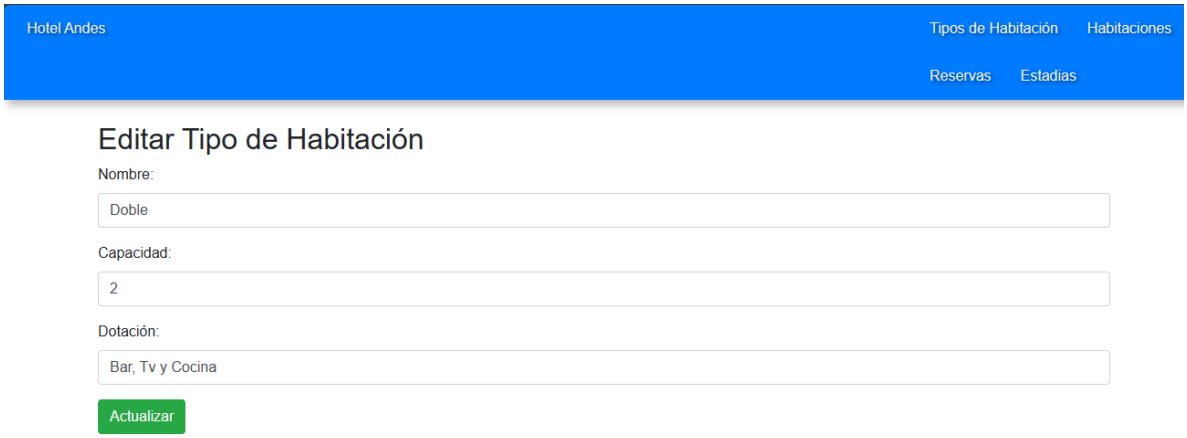
```
const estadiaSchema = new mongoose.Schema({
  _id: String,
  fechaCheckIn: Date,
  fechaCheckOut: Date,
  checkInRealizado: Boolean,
  checkOutRealizado: Boolean,
  reserva: {
    type: String,
    ref: 'Reserva',
  },
});

const Estadia = mongoose.model('Estadia', estadiaSchema);
```

```
module.exports = {
  TipoHabitacion,
  Habitacion,
  Reserva,
  Estadia,
};
```

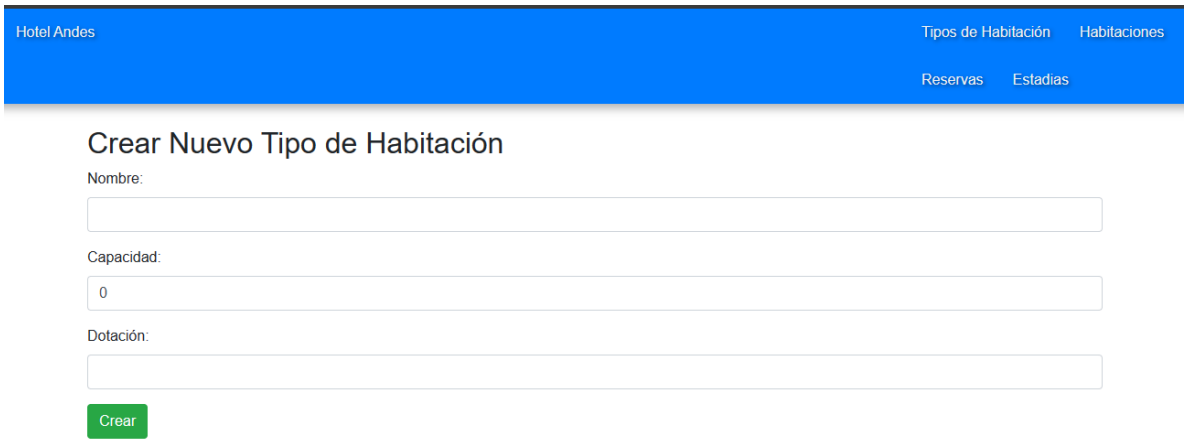
7. **Escenarios de prueba para todos los req**

Req 1:
Eliminar y mostrar

editar:

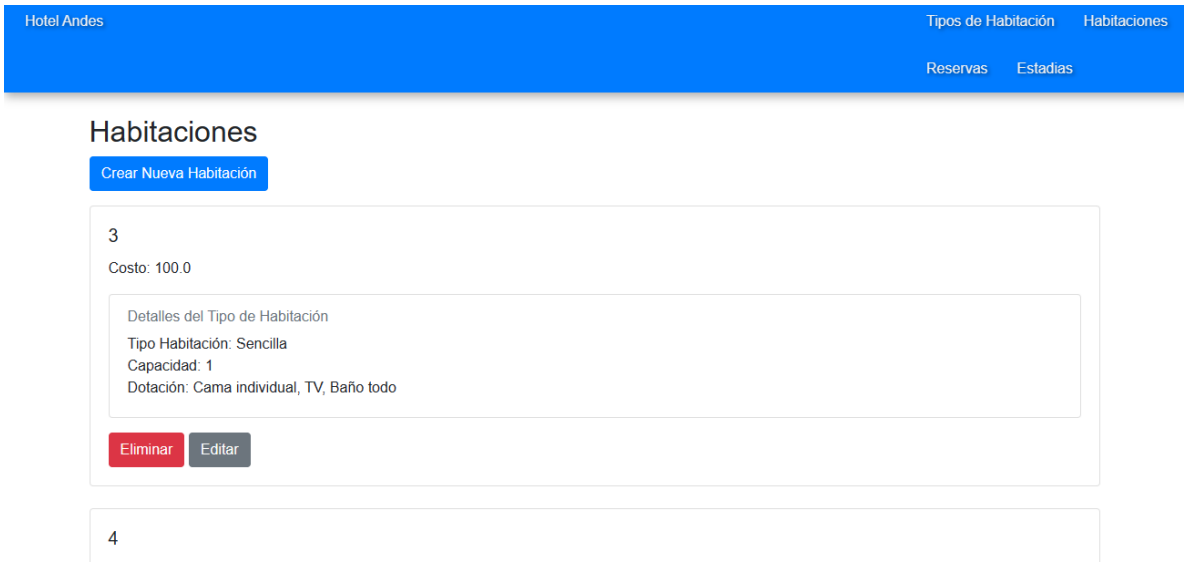


crear

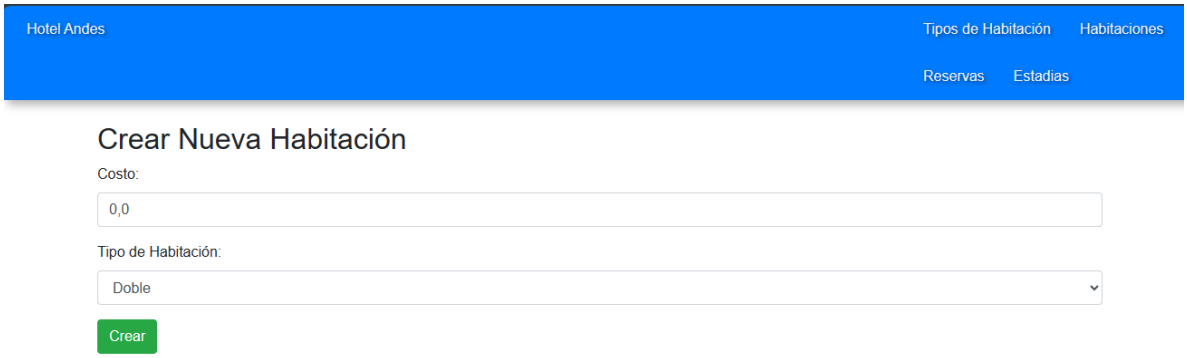


Req 2:

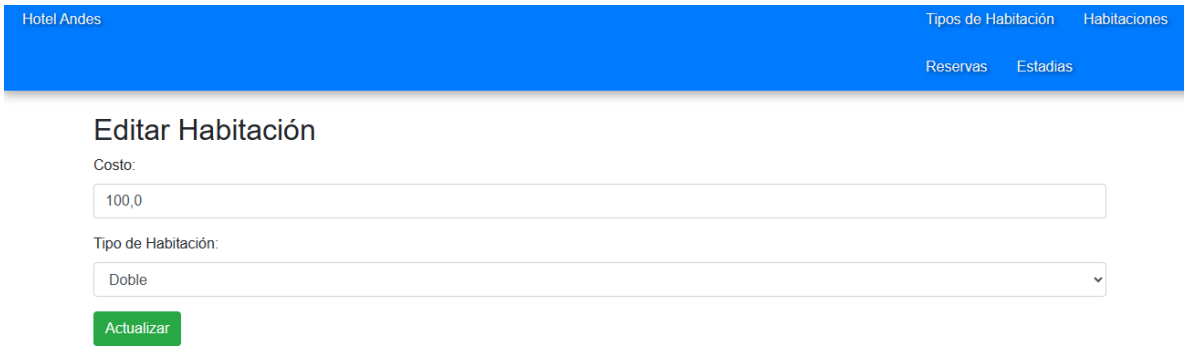
Eliminar y mostrar:



Crear:

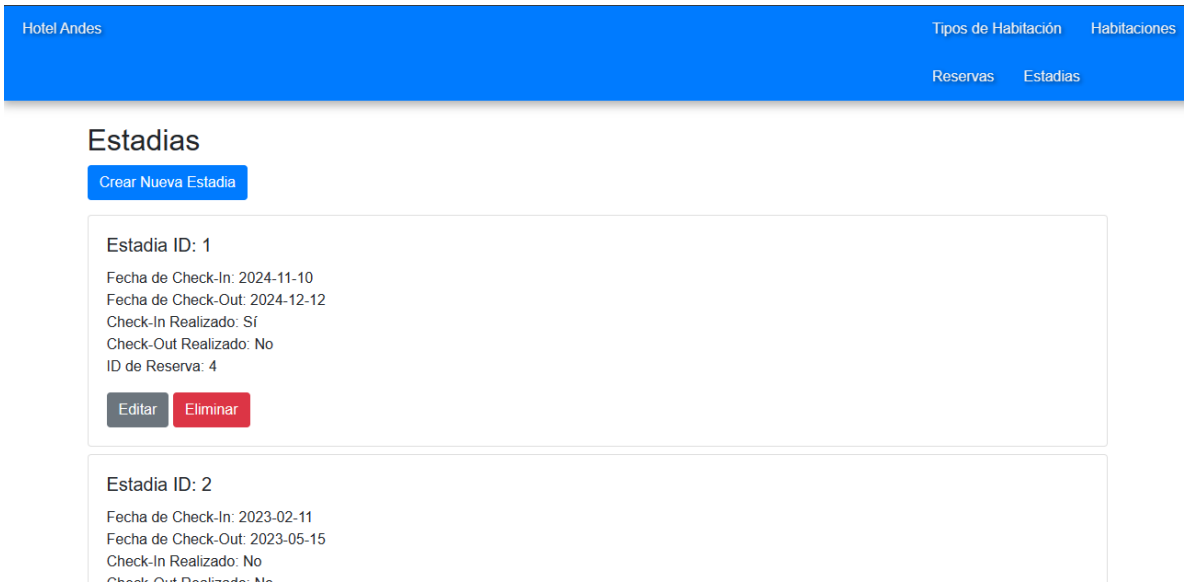


Editar:

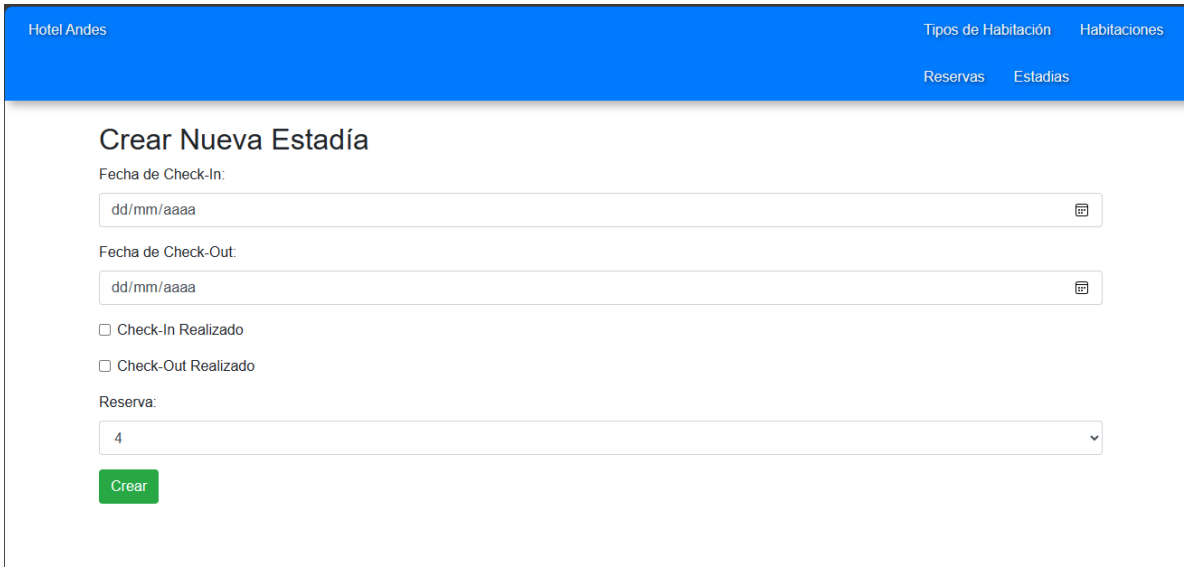


Req 4 y 7 estadia:

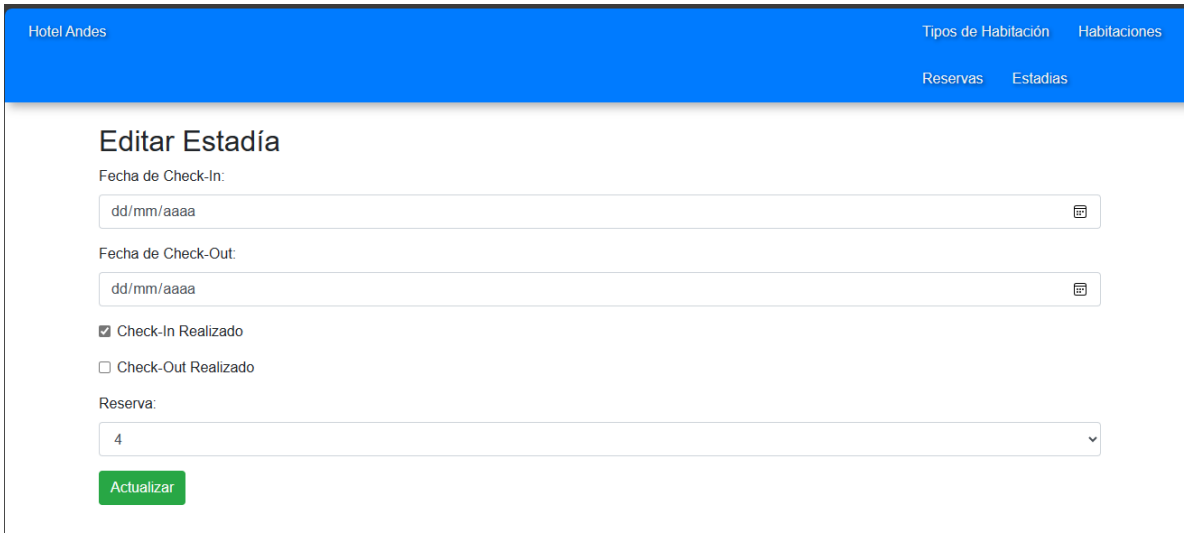
Mostrar eliminar



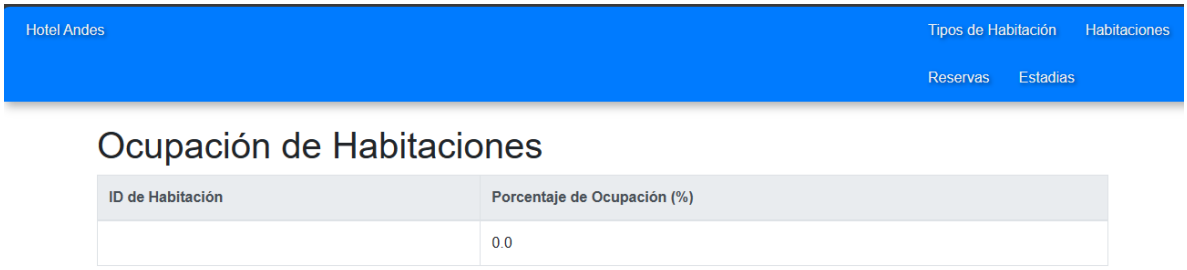
Crear:



Editar:



Req 2 consultas:



8. **Escenarios de prueba**

a. **Arriba están escenarios de que se puede realizar todo el CRUD y en la sustentación se demostrara**

b. **Para el siguiente documento incumple entonces sale este error:**
E11000 error collection: ISIS2304B08202320.tipoHabitacion

```
db.tipoHabitacion.insertOne({
    "_id": "2",
    "nombre": "Habitación Estándar"
    // Falta 'capacidad' y 'dotacion'
})
```

```
db.reserva.insertOne({
    "_id": "4",
    "fechaLlegada": ISODate("2023-12-15T00:00:00Z"),
    "fechaSalida": ISODate("2023-12-20T00:00:00Z"),
    "tipoHabitacion": db.tipoHabitacion.findOne({ "_id": "1" }),
    "titular": "Ana Gómez",
    "habitacion": db.habitacion.findOne({ "_id": "99" }) // Esta habitación no existe
  })
```

```
db.estadia.insertOne({
    "_id": "2",
    "fechaCheckIn": ISODate("2024-02-01T00:00:00Z"),
    "fechaCheckOut": ISODate("2024-02-07T00:00:00Z"),
    "checkInRealizado": false,
    "checkOutRealizado": false,
    "reserva": db.reserva.findOne({ "_id": "99" }) // Esta reserva no existe
  })
```

Así para cada una de las colecciones

C. script utilizado