**Explicación prueba Sistema hotelero**

**ESTRUCTURA DEL PROYECTO**

bash

CopiarEditar

hotelAdmin/

├── hotel-backend/ # Backend Laravel 10

├── hotel-frontend/ # Frontend Vue 3 + Vite

└── README.md # Documentación del proyecto

|

**1. BACKEND (Laravel)**

**Tecnologías usadas:**

PHP 8.x

Laravel 10.x

MySQL

Laravel Sanctum (autenticación API)

RESTful API

**Estructura funcional:**

HotelController: CRUD de hoteles

HabitacionController: CRUD de habitaciones

Modelos: Hotel, Habitacion, TipoHabitacion, Acomodacion

Relaciones:

Un **Hotel** tiene muchas **Habitaciones**

Una **Habitación** pertenece a un **Hotel**

Una **Habitación** tiene un **TipoHabitación**

Una **Habitación** tiene muchas **Acomodaciones** (relación many-to-many)

**Validaciones:**

Validación vía Request->validate()

Manejo de errores con código HTTP 422

Mensajes devueltos al frontend con detalles de los errores

**Base de datos:**

Migraciones para todas las tablas

Seeder para datos iniciales (TipoHabitacion, Acomodacion)

Archivo init.sql generado con datos ya cargados

**Comandos utilizados:**

bash

CopiarEditar

composer create-project laravel/laravel hotel-backend

php artisan make:model Hotel -mcr

php artisan make:model Habitacion -mcr

php artisan make:migration create\_tipo\_habitacions\_table

php artisan make:migration create\_acomodacions\_table

php artisan migrate

php artisan db:seed

**2. FRONTEND (Vue 3)**

**Tecnologías:**

Vue 3

Vite

Composition API

Axios

Bootstrap (o estilos personalizados)

Vue Router

**Componentes principales:**

HotelList.vue: Listado, creación y edición de hoteles

HabitacionForm.vue: Formulario dinámico según tipo de habitación

App.vue: Vista principal

router/index.js: Rutas (si aplica)

main.js: Configuración principal del frontend

**Funcionalidades:**

Crear, editar, eliminar hoteles

Crear habitaciones asociadas a un hotel

Formulario dinámico (acomodaciones cambian según tipo de habitación)

Validación de errores (muestra mensajes por alerta o en campos)

**Axios configurado para consumir /api**

**3. PRUEBAS**

**Backend:**

Testeado con Postman para cada endpoint:

GET /api/hoteles

POST /api/hoteles

PUT /api/hoteles/{id}

DELETE /api/hoteles/{id}

**Frontend:**

Pruebas manuales en desarrollo local (npm run dev)

Validación de formularios y flujos de edición

**4. BASE DE DATOS**

**Modelo ER básico:**

plaintext

CopiarEditar

HOTEL

- id

- nombre

- ciudad

- dirección

HABITACION

- id

- hotel\_id (FK)

- tipo\_habitacion\_id (FK)

- precio

- número

TIPO\_HABITACION

- id

- nombre

ACOMODACION

- id

- nombre

habitacion\_acomodacion (pivot)

- habitacion\_id

- acomodacion\_id

**Exportación:**

Desde MySQL Workbench: archivo init.sql contiene estructura + datos

**5. DESPLIEGUE**

**FRONTEND (Render):**

Carpeta: hotel-frontend

Build command: npm install && npm run build

Publish directory: dist

Tipo: Static Site

Link generado: https://hoteladmin-front.onrender.com (ejemplo)

**BACKEND (Render o Railway):**

Laravel configurado con:

.env conectado a PostgreSQL o MySQL en la nube

CORS habilitado para frontend

Comando de arranque en Render:

bash

CopiarEditar

php artisan serve --host=0.0.0.0 --port=10000

**6. README.md (resumen de despliegue y uso)**

Incluye:

Instrucciones para correr localmente (npm install, php artisan serve)

Instrucciones para poblar base de datos

Enlaces de despliegue

Datos de ejemplo para probar

**7. RECOMENDACIONES**

Usar Laravel Sanctum o Passport para autenticar usuarios si se escalan permisos

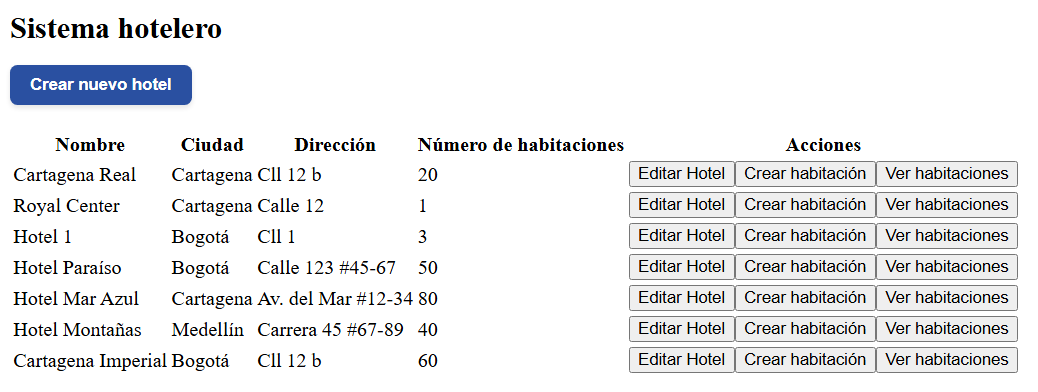
Implementar roles en backend para admin vs usuario común

Aplicar paginación a listados extensos

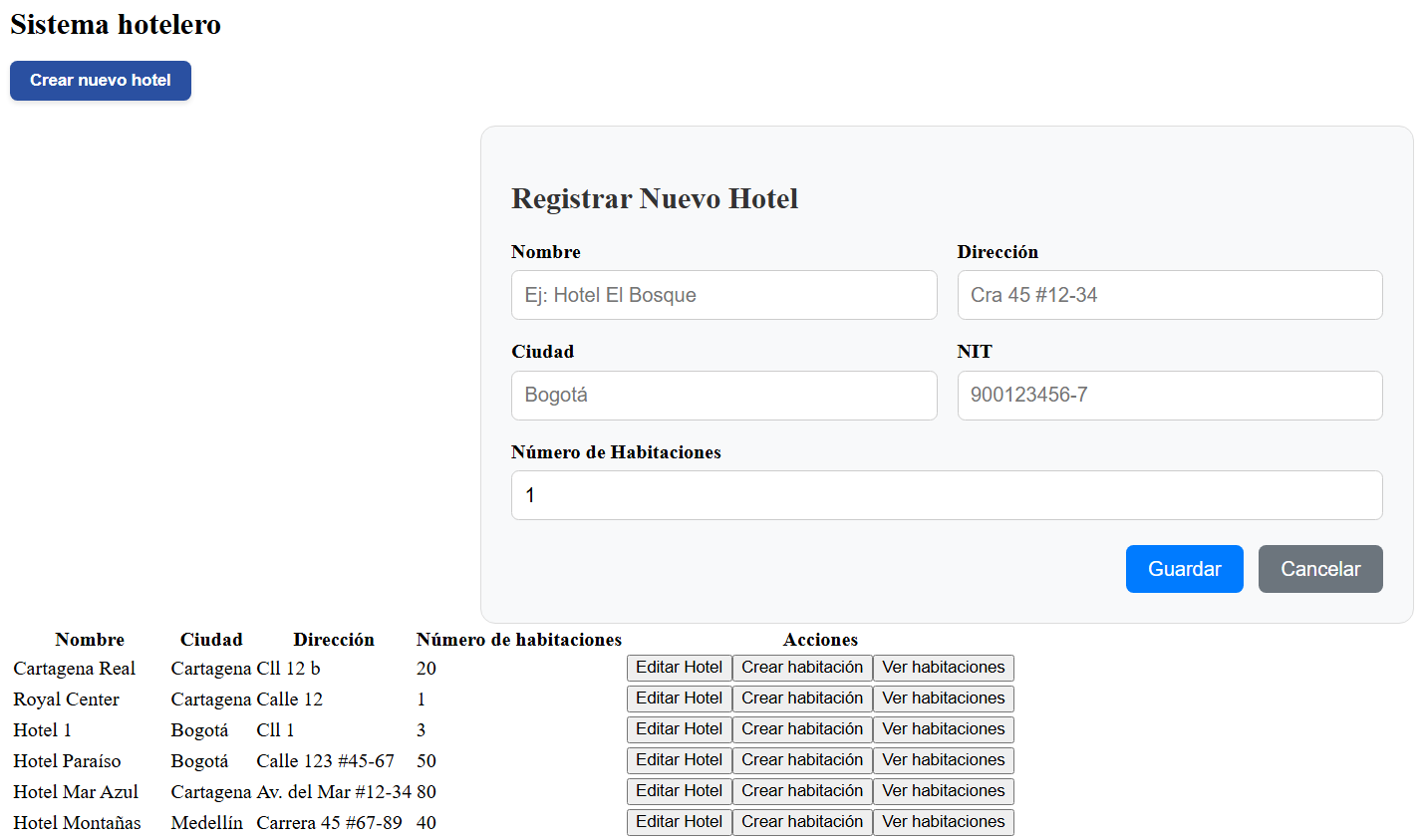
Validar desde backend el número de acomodaciones según tipo de habitación

**8. CAPTURAS DE PANTALLA DEL SISTEMA**

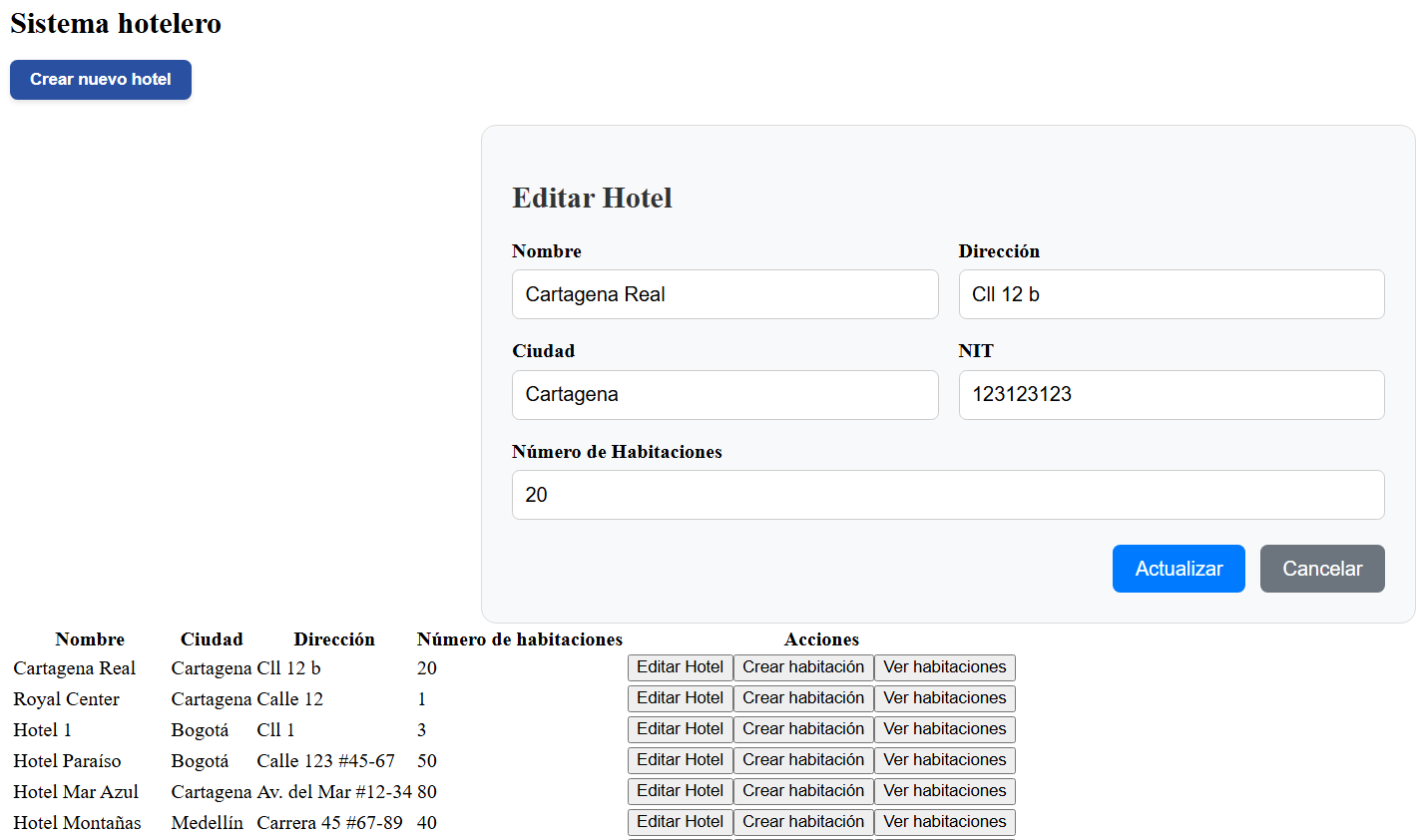
IMG 1: Vista general del sistema



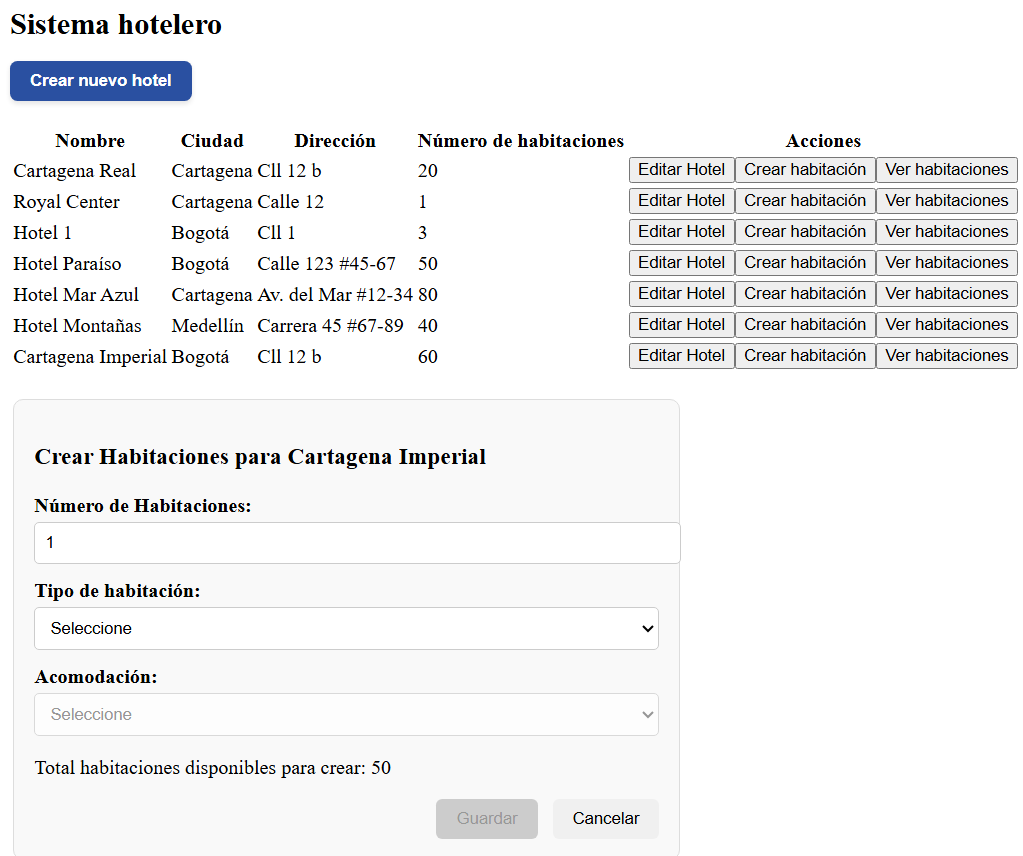
Img 2: Formulario para crear un hotel



Img 3: Formulario para editar un hotel



IMG 4: Formulario para crear una habitación asociándola a un hotel



IMG 5: Consultar habitaciones por hotel

