**Proyecto: Sistema de Gestión de Hoteles**

**1. Introducción**

* **Objetivo**: Desarrollar un sistema para administrar hoteles y sus configuraciones de habitaciones.
* **Alcance**: Ingreso de hoteles con datos básicos y tributarios; asignación de tipos y acomodaciones de habitaciones.
* **Limitaciones**: Sin autenticación/roles; ambiente local.

**2. Requisitos Funcionales**

* Registrar hoteles con:
  + Nombre
  + NIT (formato: 8 dígitos, un guion y 1 dígito; regex ^\d{8}-\d$)
  + Dirección
  + Ciudad (catálogo pre-cargado)
  + Número máximo de habitaciones
* Asociar a cada hotel uno o varios tipos de habitación y cantidad según reglas:
  + **Estándar** → Sencilla, Doble
  + **Junior** → Triple, Cuádruple
  + **Suite** → Sencilla, Doble, Triple
* Validaciones:
  + La suma de cantidades no supera max\_rooms.
  + No duplicar hoteles (única combinación nombre+ciudad+NIT).
  + No repetir combinación tipo/acomodación en un mismo hotel.

**3. Requisitos Técnicos**

* **Backend**: PHP 8+ con Laravel (RESTful API).
* **Frontend**: React con Next.js (framework profesional para producción).
* **Base de datos**: PostgreSQL.
* **Servidor**: Local con Laragon u otro similar (no cloud).
* **IDE**: Visual Studio Code.
* Navegadores soportados: Firefox y Chrome.

**4. Arquitectura y Diseño**

* **Patrón**: MVC en Laravel; Componentes y páginas en Next.js.
* **Principios**: SOLID, Clean Architecture (capas: Controllers, Services, Repositories).
* **Diagramas**: UML de clases, casos de uso y secuencias.

**5. Esquema de Base de Datos**

* **hotels**: id, name, nit, address, city\_id, max\_rooms, timestamps
* **cities**: (migración inicial y seeder) id, name.
* **room\_types** (catálogo): id, code, name (Estándar, Junior, Suite).
* **accommodations** (catálogo): id, code, name (Sencilla, Doble, Triple, Cuádruple).
* **hotel\_rooms**: id, hotel\_id, room\_type\_id, accommodation\_id, quantity, timestamps.

**6. Endpoints REST API**

* POST /api/hotels → Crear hotel
* GET /api/hotels → Listar todos
* GET /api/hotels/{id} → Detalle con habitaciones
* PUT /api/hotels/{id} → Actualizar hotel
* DELETE /api/hotels/{id} → Eliminar hotel
* POST /api/hotels/{id}/rooms → Asignar habitaciones
* DELETE /api/hotels/{id}/rooms/{rid} → Eliminar habitación

**7. Frontend (Next.js)**

* **Estructura**:
  + pages/ → rutas y SSR/SSG si aplica
  + components/ → FormularioHotel, ListaHoteles, FormAsignacion
  + lib/ → llamadas a API con Axios
  + styles/ → Tailwind CSS
* **Estado**: React Context o SWR para datos
* **Validación**: Yup + React Hook Form

**8. Pruebas**

* **Backend**: PHPUnit (unitarias e integración, uso de RefreshDatabase).
* **Frontend**: Jest + React Testing Library (componentes, servicios).
* **Cobertura**: ≥ 80%.

**9. Documentación**

* **UML**: Clases, casos de uso, secuencia.
* **README**:
  + Descripción
  + Pre-requisitos (PHP, Node, Laragon, PostgreSQL)
  + Guía paso a paso de instalación y ejecución
  + Comandos Git y ramas (git-flow)
  + Enlace al repositorio público
* **Dump**: Archivo SQL listo para importar.

**10. Despliegue Local**

* **Instrucciones para la "abuelita"**:
  1. Instalar Laragon y clonar repo.
  2. Copiar .env.example a .env y configurar DB.
  3. Ejecutar composer install y npm install.
  4. Ejecutar php artisan migrate --seed.
  5. Ejecutar npm run dev y php artisan serve.
  6. Abrir http://localhost:3000 en el navegador.