🏨 🚀 Plan Maestro de Sprints – Proyecto ReservaHotel

Duración estimada: 8-10 semanas (si trabajas solo con Codex, podría ser más rápido).

🌱 🧮 Sprint 1: Backend Básico (Infraestructura sólida)

🎯 **Objetivo:**
Construir las bases del backend con NestJS + Prisma + PostgreSQL usando Clean Architecture. Dejarlo preparado para escalar a microservicios y soportar carga futura.

✅ **Actividades:**

- Inicializar proyecto con **NestJS** y estructura Clean Architecture.
- Conectar con **PostgreSQL** usando **Prisma ORM**.
- Definir los modelos base en schema.prisma:
  - **Usuario** (Admin, Recepcionista, Cliente).
  - **Habitación** (número, estado, precio).
  - **Reserva** (fechas, estado).
- Crear **API REST básica**:
  - GET /habitaciones
  - POST /habitaciones
- Configurar **Docker** para desarrollo y producción.
- Preparar scripts para **migraciones automáticas** con Prisma.

🛡️ **Ciberseguridad Sprint 1:**

- Configurar variables de entorno con **dotenv-safe** (validación).
- Usar **helmet** para cabeceras HTTP seguras.
- Bloquear métodos HTTP innecesarios (solo permitir GET/POST/PUT/DELETE).
- Agregar **rate-limiting** con express-rate-limit.

- Configurar **CORS restrictivo** (solo frontend permitido).

🎨 **Diseño y calidad Sprint 1:**

- Estandarizar código con **ESLint + Prettier + Husky**.
- Definir **paleta de colores y tipografía** para la app (empezar en Figma).
- Crear wireframes básicos para: login, dashboard, reservas.

📦 **Entregables:**

✅ API básica funcional (NestJS + PostgreSQL).
✅ Seguridad inicial (rate-limiting, CORS, helmet).
✅ Docker Compose listo.
✅ Wireframes iniciales en Figma.

🔐🗓️ Sprint 2: Autenticación y Roles

🎯 **Objetivo:**
Implementar autenticación segura y autorización con roles (Admin, Recepcionista, Cliente).

✅ **Actividades:**

- Registro y login usando **JWT** con refresh tokens.
- Middleware para validar JWT y permisos por rol.
- Hash de contraseñas con **bcrypt**.
- Endpoints protegidos:
  - Solo Admin puede crear/editar recepcionistas.
  - Recepcionistas pueden gestionar reservas.
  - Clientes solo ven y gestionan sus reservas.
- Soporte para “recordarme” (tokens persistentes).

🛡️ **Ciberseguridad Sprint 2:**

- **Validación de inputs** con class-validator (evitar SQLi y XSS).
- Protección contra **brute-force** login con express-rate-limit.
- Forzar **contraseñas fuertes** (mínimo 12 caracteres, 1 mayúscula, 1 número, 1 especial).
- Implementar **2FA (Two Factor Auth)** opcional para Admins.
- Guardar contraseñas con **argon2** como upgrade futuro.

🎨 **Diseño y calidad Sprint 2:**

- Diseñar pantallas de **login** y **registro** (Angular Material/TailwindCSS).
- Animaciones sutiles para transiciones.
- Mockups en Figma para flujo de autenticación.

📦 **Entregables:**

✅ Autenticación y roles funcionales.
✅ JWT con refresh tokens.
✅ Validación y seguridad robusta en endpoints.

🧾 🧮 Sprint 3: CRUD Reservas + Lógica de negocio

🎯 **Objetivo:**
Implementar la lógica principal para crear, consultar, modificar y cancelar reservas.

✅ **Actividades:**

- CRUD completo para **reservas**:
  - POST /reservas
  - GET /reservas
  - PUT /reservas/:id
  - DELETE /reservas/:id
- Validación de fechas (no permitir solapamientos).
- Cálculo automático de costo por habitación y días.
- Estado de reserva: pendiente, confirmada, cancelada.

- Recepcionista puede crear reservas a nombre de clientes.

🛡️ **Ciberseguridad Sprint 3:**

- Validar que un cliente **solo puede modificar sus propias reservas**.
- Sanitizar comentarios/notas en reservas para evitar **XSS**.
- Proteger contra **IDOR (Insecure Direct Object References)**.

🎨 **Diseño y calidad Sprint 3:**

- Pantalla de listado de reservas con filtros (fecha, estado).
- Formulario de reservas intuitivo con calendario (Angular Material).
- Indicadores de estados (badge: pendiente, confirmada, etc.).

📦 **Entregables:**

✅ CRUD reservas funcional.
✅ Validaciones de negocio y seguridad.
✅ UI atractiva para gestionar reservas.

🌐 🗓️ Sprint 4: Frontend conectado

🎯 **Objetivo:**
Conectar el frontend Angular con el backend usando HttpClient.

✅ **Actividades:**

- Crear servicios en Angular para consumir la API.
- Autenticación frontend con JWT (almacenar en **HttpOnly cookies**).
- Componentes:
    - Login/Register
    - Dashboard cliente
    - Lista de habitaciones

    - Formulario de reservas

🛡️ **Ciberseguridad Sprint 4:**

- Usar **HttpOnly cookies** para tokens (mitigar XSS).
- Implementar **guards en Angular** para proteger rutas.
- Validar en frontend datos antes de enviar.

🎨 **Diseño y calidad Sprint 4:**

- Aplicar **tema responsive** (desktop, tablet, móvil).
- Animaciones con Angular Animations.
- Pantalla atractiva de dashboard con estadísticas (ocupación hotelera).

📦 **Entregables:**

✅ Frontend funcional y conectado.
✅ UI profesional y responsive.
✅ Seguridad en cliente (guards, cookies HttpOnly).

🛡️🧮 Sprint 5: Seguridad avanzada y pruebas

🎯 **Objetivo:**
Asegurar el sistema y validar todos los flujos con pruebas automáticas.

✅ **Actividades:**

- Pruebas unitarias con **Jest** (backend).
- Pruebas de integración con **Supertest** (backend).
- Pruebas unitarias con **Jasmine/Karma** (frontend).
- Configurar CI/CD para:
    - Lint
    - Tests automáticos

- Deploy a staging (Railway + Vercel).

🛡️ **Ciberseguridad Sprint 5:**

- Configurar **CSRF tokens**.
- Hacer auditoría básica con **OWASP ZAP**.
- Configurar cabeceras seguras (HSTS, Content Security Policy).
- Revisar permisos en la base de datos (principio de mínimo privilegio).

📦 **Entregables:**

✅ Cobertura de pruebas +80%.

✅ Seguridad avanzada lista para producción.

🚀 🗓️ Sprint 6: Despliegue y documentación

🎯 **Objetivo:**

Lanzar la app a producción y documentar todo el sistema.

✅ **Actividades:**

- Despliegue automático con GitHub Actions:
  - Backend → **Railway/Render**
  - Frontend → **Vercel/Netlify**
- Configurar monitoreo con **Sentry** (errores) y **Grafana** (métricas).
- Documentar API con **Swagger**.
- Crear README.md profesional con instrucciones para developers.

🎨 **Diseño y calidad Sprint 6:**

- Animaciones finales en frontend (entradas, loaders).
- Branding: logo, favicon y esquema de colores final.

📦 **Entregables:**

✅ App desplegada (producción).

✅ Documentación técnica y de usuario.

✅ Sistema profesional y escalable.

🏨 ReservaHotel – Sistema de Reservas Hoteleras Escalable y Seguro

📝 Descripción General

ReservaHotel es un sistema web diseñado para gestionar reservas hoteleras de forma eficiente, segura y escalable. Está pensado tanto para el **personal del hotel** (administradores y recepcionistas) como para los **clientes finales**, ofreciendo una experiencia moderna y profesional.

El sistema está construido con un enfoque **Clean Architecture** y está preparado para crecer hacia microservicios en el futuro. Incorpora las mejores prácticas de desarrollo seguro (OWASP Top 10) y cuenta con un diseño UI/UX atractivo y responsive.

Es ideal como **proyecto empresarial** o para **portafolio profesional**, ya que cubre un ciclo completo de desarrollo: desde infraestructura y CI/CD hasta despliegue en la nube.

👥 Usuarios y Roles

El sistema define tres tipos de usuarios:

1. **Cliente**
    a. Realiza reservas y gestiona su historial.
    b. Recibe correos de confirmación.
    c. Puede cancelar o modificar reservas antes de la fecha de entrada.
2. **Recepcionista**
    a. Gestiona reservas de clientes (presenciales o telefónicas).
    b. Realiza check-in y check-out.

c. Consulta la disponibilidad de habitaciones.

3. **Administrador**
    a. Crea, edita y elimina habitaciones.
    b. Gestiona cuentas de recepcionistas.
    c. Consulta reportes básicos sobre ocupación y actividad del sistema.

🎨 Características Clave

✅ **Frontend (Angular)**

- UI responsive y moderna con **Angular Material/TailwindCSS**.
- Rutas protegidas según el rol del usuario.
- Formularios dinámicos y validados para login, registro y reservas.

✅ **Backend (NestJS + Prisma)**

- API REST modular y escalable.
- Gestión de usuarios con roles y permisos.
- Conexión a base de datos PostgreSQL con Prisma ORM.

✅ **Seguridad**

- Autenticación con **JWT** (tokens de acceso y refresh).
- Validación de datos y sanitización para prevenir ataques XSS e inyecciones SQL.
- Middleware de control de roles y permisos.
- Protección contra ataques de fuerza bruta (rate limiting).

✅ **Ciberseguridad avanzada (en fases futuras)**

- Contraseñas fuertes con **argon2**.
- Soporte para 2FA (Two-Factor Authentication).
- Auditoría básica con herramientas como **OWASP ZAP**.

✅ **DevOps y Despliegue**

- Contenerización completa con Docker.
- CI/CD con **GitHub Actions**: lint, tests y despliegue automático.
- Backend desplegado en **Railway/Render**, frontend en **Vercel/Netlify**.

✅ **Monitoreo**

- Gestión de errores con **Sentry**.
- Métricas de rendimiento con **Prometheus + Grafana**.

Arquitectura Técnica

Stack Tecnológico

Capa	Tecnología
Frontend	Angular, Angular Material
Backend	NestJS, Prisma, Node.js
Base de Datos	PostgreSQL
Autenticación	JWT + bcrypt
Infraestructura	Docker, Railway, Vercel
CI/CD	GitHub Actions

Arquitectura (Clean Architecture + Hexagonal)

- **Core Layer**: Entidades y casos de uso (independiente del framework).
- **Adapters Layer**: Implementaciones (ORM, JWT, Email).
- **API Layer**: Controladores REST (NestJS).
- **Infrastructure**: Configuración de base de datos, seguridad y módulos externos.

Módulos Principales

Autenticación y Autorización

- Registro/login con validación de credenciales.
- Roles definidos: admin, recepcionista, cliente.
- Middleware para validar JWT y roles en cada endpoint.

Gestión de Habitaciones

- CRUD completo para habitaciones.
- Estado de habitación: libre, ocupada, mantenimiento.

🗓️ Gestión de Reservas

- Crear, consultar, modificar y cancelar reservas.
- Validación de disponibilidad para evitar solapamientos.
- Cálculo automático de precio por noche.

📊 Panel Administrativo

- Dashboard con métricas: ocupación, ingresos, número de reservas.
- Gestión de recepcionistas y configuración general.

🛡️ Ciberseguridad Integrada

✔️ Protección contra **OWASP Top 10**.
✔️ Validación y sanitización exhaustiva en backend y frontend.
✔️ Contraseñas encriptadas con **bcrypt** (y upgrade a argon2).
✔️ Logs de auditoría para acciones críticas (alta/baja de usuarios, reservas).

🎨 UI/UX

El diseño se centra en:

- **Experiencia de usuario fluida** (UX).
- **Interfaz moderna** con colores profesionales.
- Responsive para escritorio, tablet y móvil.

Wireframes y mockups diseñados en **Figma**.

🚀 Flujo DevOps

1. Push a rama develop → Lint + Tests automáticos.
2. Merge a main → Despliegue automático (backend + frontend).
3. Monitoreo activo con Sentry y alertas configuradas.

Características destacadas

- ✅ Código modular y mantenible.
- ✅ Preparado para multi-hotel (fases futuras).
- ✅ Escalabilidad vertical y horizontal (microservicios).
- ✅ Pensado para usarse en entornos de producción real.

Resultado esperado

Un sistema robusto, seguro y profesional para gestionar reservas de hotel que puede:

- ✔ Escalar fácilmente a múltiples hoteles o franquicias.
- ✔ Servir como referencia para aplicaciones empresariales de gran nivel.
- ✔ Impresionar a cualquier reclutador o cliente.