EJERCICIO 1:

1) Funcionalidades (Requisitos funcionales)

Vista general por rol

Paciente

- RF1. Registro e inicio de sesión (email/telefono + verificación).
- RF2. Gestión de perfil (datos personales, EPS/aseguradora, contacto, consentimiento).
- RF3. Búsqueda de médicos por especialidad/horario/sede/modalidad (presencial/virtual).
- RF4. Consulta de disponibilidad en calendario.
- RF5. Reserva de cita (fecha, modalidad, motivo, medio de pago si aplica).
- RF6. Reprogramación/cancelación con política de tiempo.
- RF7. Historial de citas y estado (reservada, confirmada, atendida, cancelada).
- RF8. Recordatorios y notificaciones (email/SMS/WhatsApp/push).
- RF9. Adjuntar información previa (ej. órdenes médicas, exámenes) opcional.

Médico

- RF10. Autenticación y roles.
- RF11. Configurar agenda: bloques de atención, duración por tipo de consulta, tiempos de descanso, días no laborables, cupos virtuales/presenciales.
- RF12. Aprobar/rechazar solicitudes especiales (si se usa “pendiente de aprobación”).
- RF13. Ver agenda (día/semana) y detalles de cada cita.
- RF14. Bloquear/desbloquear espacios (reunión interna, congreso, vacaciones).
- RF15. Marcar cita como atendida/no asistida, añadir notas mínimas (no-HCE completa en esta fase).

Administrador

- RF16. CRUD de usuarios (pacientes, médicos, recepcionistas) y especialidades.
- RF17. Reglas globales: política de cancelación, horarios de la sede, tipos de citas.
- RF18. Supervisión: panel de ocupación, no-shows, tiempos de espera.
- RF19. Configurar canales de notificación (proveedores), plantillas de mensajes.
- RF20. Auditoría y reportes básicos (reservas por médico/sede/mes, cancelaciones).

Transversales

- RF21. Autorización por rol (RBAC).
- RF22. Logs de auditoría (quién hizo qué y cuándo).
- RF23. Internacionalización básica (es/en) y zona horaria.
- RF24. Accesibilidad AA.

2) Modularización (componentes)

Frontends

- **Web Paciente** (React/Angular/Vue): búsqueda, reservas, perfil.
- **Web Médico**: configuración de agenda y vista de calendario.
- **Web Admin**: catálogo, reglas, reportes.

Backoffice/API (modular monolito recomendado para Fase 1)

- **Auth & Identity**: registros, login, RBAC, sesiones, verificación 2FA opcional.
- **Gestión de Usuarios**: pacientes, médicos, especialidades, sedes.
- **Agenda y Disponibilidad**:
  - Generación de slots y reglas (duración, buffers, feriados).
  - Bloqueos, sobrescrituras locales.
- **Reservas**:
  - Orquestación de *booking* con concurrencia (locking/optimistic).
  - Políticas de cancelación/reprogramación.
  - Estados y trazabilidad.
- **Notificaciones**:
  - Envío asíncrono (email/SMS/WhatsApp/push).
  - Plantillas y programación (recordatorios T-24h, T-2h).
- **Reportes & Auditoría**:
  - Métricas operativas, logs inviolables.
- **Integraciones** (futuro):
  - Calendarios externos (iCal/Google Calendar).
  - Pasarela de pagos.
  - HCE/facturación (si entra a alcance).

Datos & Infra

- **DB relacional** (PostgreSQL/MySQL): Users, Doctors, Specialties, Slots, Appointments, Policies.
- **Cache** (Redis): calendarios generados, disponibilidad próxima.
- **Mensajería** (RabbitMQ/Kafka/SQS) para eventos de dominio (AppointmentBooked).

- **Almacenamiento** (S3/Blob) para adjuntos.
- **Observabilidad**: logs, métricas, trazas

3) Interfaces y conectores

API (REST/JSON; GraphQL opcional)

- **Auth**
  - POST /auth/register (paciente), POST /auth/login, POST /auth/verify
- **Usuarios**
  - GET /me, PATCH /me
  - GET /doctors?specialty={id}&site={id}
  - GET /specialties, GET /sites
- **Disponibilidad & Agenda**
  - GET /doctors/{id}/availability?from=2025-09-18&to=2025-09-25
  - POST /doctors/{id}/schedule (médico): bloques, buffers, vacaciones
  - POST /doctors/{id}/blocks (bloqueos ad-hoc)
- **Reservas**
  - POST /appointments (body: doctorId, slotId, modality, reason) → *lock transaccional*
  - GET /appointments/{id}
  - PATCH /appointments/{id} (reprogramar/cancelar con política)
  - GET /patients/{id}/appointments
- **Admin**
  - POST /specialties, POST /sites, POST /users (rol=doctor/recep)
  - GET /reports/occupancy?from&to&site

Eventos de dominio (mensajería)

- AppointmentBooked, AppointmentRescheduled, AppointmentCancelled
  - Consumidos por **Notificaciones** y **Reportes**.

Conectores externos

- **Email/SMS**: webhooks de estado; reintentos exponenciales.
- **Calendario** (futuro): iCal feed por médico; sincronización pull/push opcional.
- **Pagos** (futuro): POST /payments/intent antes de confirmar reserva; *webhook* de confirmación.

Seguridad

- OAuth2/OIDC + JWT; refresco de tokens; *scopes* por rol.
- Rate limiting por IP/usuario; protección CSRF en vistas web.
- Auditoría inmutable (tabla *audit_events*).

4) Análisis y justificación de diseño

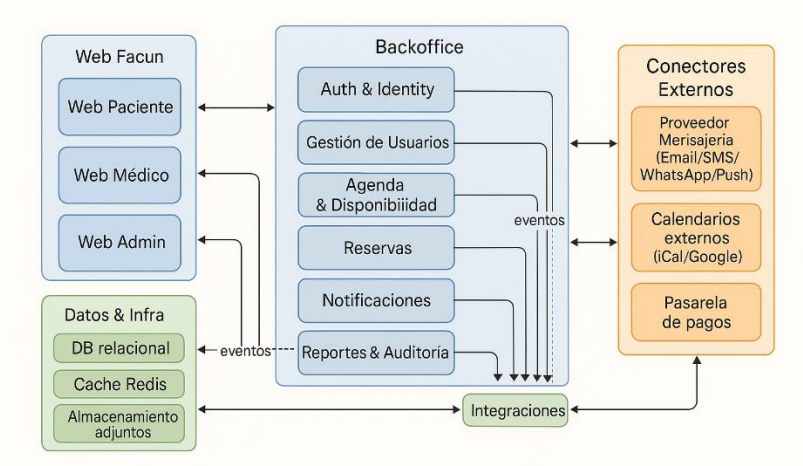
- **Modular monolito** inicial: reduce complejidad operativa, mantiene transacciones fuertes para reservar un *slot* (evita doble asignación). Se

separan *bounded contexts* (Agenda, Reservas, Notificaciones) dentro del mismo despliegue.

- **Consistencia**: transacción BEGIN → check slot free (SELECT … FOR UPDATE) → insert appointment → commit. Alternativa: bloqueo optimista con versión de slot.
- **Escalabilidad pragmática**: cache de disponibilidad próxima (horizonte 14 días) y *read models* para calendarios; *warm-up* nocturno de slots.
- **Resiliencia**: eventos de dominio alimentan Notificaciones y Reportes de forma asíncrona; reintentos idempotentes.
- **Experiencia de usuario**: recordatorios y reglas claras bajan *no-show*.
- **Seguridad y privacidad**: principio de mínimo privilegio; PII cifrada en reposo; logs sin datos sensibles.

Riesgos y mitigaciones

- **No-shows altos** → depósitos/restricciones de reprogramación tardía (fase 2 pagos).
- **Contención en reservas pico** → colas cortas y retries con *backoff*, plus *circuit breaker*.
- **Complejidad de agenda** (excepciones, feriados, multi-sede) → motor de reglas declarativas en “Agenda”.



EJERCICIO 2

1) Funcionalidades (Requisitos funcionales)
Por rol
Cliente (usuario final)

- RF1 Registro/inicio de sesión, recuperación de cuenta.
- RF2 Gestión de perfil, direcciones múltiples y métodos de pago tokenizados.
- RF3 Descubrimiento: listar restaurantes por zona, filtros (tiempo de entrega, costo envío, rating), búsqueda por plato.
- RF4 Menú y personalización (tamaños, extras, notas).

- RF5 Carrito, cálculo de precio (productos, impuestos, tarifas, propina, cupones).
- RF6 Checkout: pago (autorización/captura) y confirmación.
- RF7 Seguimiento en tiempo real: preparación → recogida → entrega (mapa/ETA).
- RF8 Reprogramación/cancelación dentro de ventana permitida.
- RF9 Calificaciones/reseñas y soporte (chat/tickets/reembolsos).

Restaurante

- RF10 Onboarding y verificación (documentos, cuenta bancaria).
- RF11 Horarios, zonas de cobertura (si propio delivery), tiempos de preparación.
- RF12 CRUD de menú/catálogos, combos, disponibilidad e inventario básico.
- RF13 Bandeja de pedidos (en vivo): aceptar/rechazar, tiempos estimados, marcar “listo”.
- RF14 Integración POS opcional; reportes de ventas, liquidaciones.

Repartidor

- RF15 Onboarding/KYC (documento, vehículo, antecedentes).
- RF16 Estado en línea/fuera de línea, zonas activas.
- RF17 Ofertas de viaje (ping), aceptar/declinar; navegación a restaurante/cliente.
- RF18 Prueba de entrega (foto, firma, OTP), incidencias (cliente ausente).
- RF19 Balance de ganancias, propinas, retiro/cash-out.

Administrador/Operaciones

- RF20 Gestión de catálogos globales, comisiones, tarifas y “surge” por zona/tiempo.
- RF21 Incentivos (bonos a repartidores), promociones/cupones.
- RF22 Monitoreo en vivo (pedidos, SLAs), soporte y reembolsos.
- RF23 Detección de fraude/riesgo, auditoría y reportes.

Transversales

- RF24 RBAC, auditoría, privacidad/PII, accesibilidad.
- RF25 Notificaciones (push/SMS/email/WhatsApp).
- RF26 Localización/zonas geográficas (geofencing), cálculo de ETA.

2) Modularización del sistema (componentes)

Frontends

- App **Cliente** (iOS/Android) + Web responsive.
- **Panel Restaurante** (web) con feed en vivo.
- App **Repartidor** (iOS/Android).
- **Backoffice Admin** (web).

Backend (modular monolito para MVP, con límites claros de dominio)

- **Auth & Identity** (OIDC/JWT, 2FA opcional).
- **KYC/Compliance** (repartidores y restaurantes).
- **Catálogo & Menús** (productos, variantes, disponibilidad).
- **Búsqueda & Descubrimiento** (ranking por ETA/costo/rating).
- **Carrito & Pricing** (tarifas, impuestos, cupones, propina).
- **Pedidos & Pagos** (autorización/captura, reembolsos; idempotencia).

- **Logística & Despacho** (matching pedido-repartidor, batching, re-asignación).
- **Ubicaciones & Tracking** (telemetría GPS, ETA).
- **Notificaciones** (plantillas, multicanal).
- **Promociones & Lealtad** (cupones, créditos).
- **Soporte & Reembolsos** (tickets, flujos de resolución).
- **Reportes & Auditoría**.

Datos & Plataforma

- **DB relacional** (PostgreSQL/MySQL) + **PostGIS** para geoespacial.
- **Redis** (sesiones, caché de catálogos/menús, carritos).
- **Broker** (Kafka/RabbitMQ) para eventos de dominio.
- **Almacenamiento** (imágenes de menú, prueba de entrega).
- **Observabilidad** (logs, métricas, trazas), **Feature Flags**.
- **Motor de ruteo/ETA** (propio simple o API de mapas externa).

3) Interfaces – Conectores

APIs (REST/JSON; GraphQL opcional para frontends ricos)

Cliente

- GET /restaurants?lat=&lng=&filters=...
- GET /restaurants/{id}/menu
- POST /cart / PUT /cart/items
- POST /checkout (idempotency-key)
- GET /orders/{id} (estado)
- GET /orders/{id}/tracking (SSE/WebSocket para actualizaciones)

Restaurante

- GET /merchant/me/orders?status=pending|in_progress
- PATCH /merchant/orders/{id} {accept|reject|ready}
- PUT /merchant/menu (batch upsert)
- PUT /merchant/hours (horarios)

Repartidor

- POST /courier/online / POST /courier/offline
- GET /courier/offers/stream (WebSocket/SSE)
- POST /courier/offers/{id}/accept
- POST /courier/tasks/{id}/status {arrived, picked_up, delivered}
- POST /courier/proof (foto/firma/OTP)
- GET /courier/balance

Admin

- PUT /fees (envío, servicio, surge)
- POST /promotions/coupons
- GET /ops/monitoring?city=...
- GET /reports/*

Eventos de dominio (pub/sub)

- OrderPlaced, PaymentAuthorized, OrderAcceptedByMerchant, OrderReady, CourierAssigned, OrderPickedUp, OrderDelivered, PaymentCaptured, OrderCancelled, RefundIssued.

Consumidores: Notificaciones, Reportes, Incentivos, Anti-fraude.

Conectores externos

- **Pagos** (pasarela con tokenización/vault), **Mapas/Geocoding/ETA**, **SMS/Email/Push**, **POS** (webhooks), **KYC** (validación de identidad), **3PL** (flota externa).

Seguridad

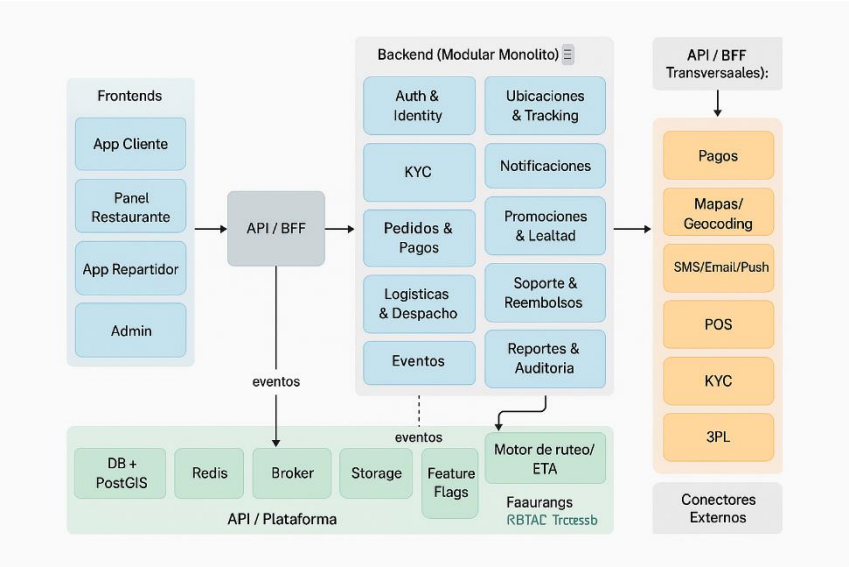
- OAuth2/OIDC + JWT con *scopes* por rol; **idempotency keys** en checkout; **rate limiting**; **cifrado** PII en reposo; **PCI**: nunca almacenar PAN sin tokenizar.

4) Análisis y justificación del diseño

- **Transaccionalidad crítica** en Pedidos & Pagos: separamos **autorización** (al crear pedido) de **captura** (tras entrega/aceptación), permitiendo cancelaciones seguras.
- **Despacho** es el núcleo de complejidad: matching multi-objetivo (tiempo, distancia, costo, rating del repartidor, batching). Para MVP: heurística "más cercano con capacidad" + penalización por desvío; dejamos *reinforcement/ML* para fases posteriores.
- **Tracking y ETA**: consistencia eventual aceptable; actualizaciones cada 3–5 s con compresión para ahorrar datos.
- **Escalabilidad pragmática**: catálogos y carritos cacheados; *read models* denormalizados para listados rápidos; colas para notificaciones y liquidaciones.
- **Resiliencia**: reintentos idempotentes en pagos y cambios de estado; *dead-letter queue*.
- **Privacidad**: ocultar teléfono mediante relé; limitar precisión de ubicación compartida al cliente (p. ej., ~30 m).
- **Métricas operativas** (para iterar producto): tasa de aceptación restaurante/repartidor, tiempo preparación, pickup wait, delivery time, cancelaciones, reintentos de pago, NPS.

Riesgos & mitigaciones

- **Desbalance oferta/demanda** → *surge*, incentivos temporales, throttling de pedidos en zonas saturadas.
- **Fraude** (tarjetas, triangulación) → scoring de riesgo en checkout + 3DS selectivo.
- **No-shows del repartidor** → re-asignación automática con SLA; créditos al cliente.
- **Inventario desactualizado** → "agotado" rápido en panel + API POS opcional.



EJERCICIO 3

1) Funcionalidades (Requisitos funcionales)

Usuarios

- RF1 Autenticación y gestión de roles (bibliotecario, docente, estudiante, admin).
- RF2 Alta/edición/baja de usuarios y sus estados (activo, suspendido).
- RF3 Historial de actividad por usuario (préstamos, reservas, multas).

Catálogo

- RF4 Gestión de títulos (ISBN, autor, materia, editorial).
- RF5 Gestión de ejemplares físicos asociados al título (código de barras, estado).
- RF6 Búsqueda y filtrado (por título, autor, materia, disponibilidad).

Circulación (préstamos/reservas)

- RF7 Registro de préstamo y devolución con fechas y política de vencimiento.
- RF8 Renovación de préstamos según reglas de la biblioteca.
- RF9 Reservas/colas de espera de ejemplares no disponibles.
- RF10 Cálculo y registro de multas por retraso; condonación/control por bibliotecario.

Reportes

- RF11 Reportes operativos: préstamos por periodo, morosidad, títulos más prestados.
- RF12 Reportes de catálogo e inventario: altas, bajas, existencias por estado.
- RF13 Exportación a CSV/Excel y programaciones periódicas (diario/semanal/mensual).

Auditoría y seguridad

- RF14 Bitácora de acciones (quién, qué, cuándo).

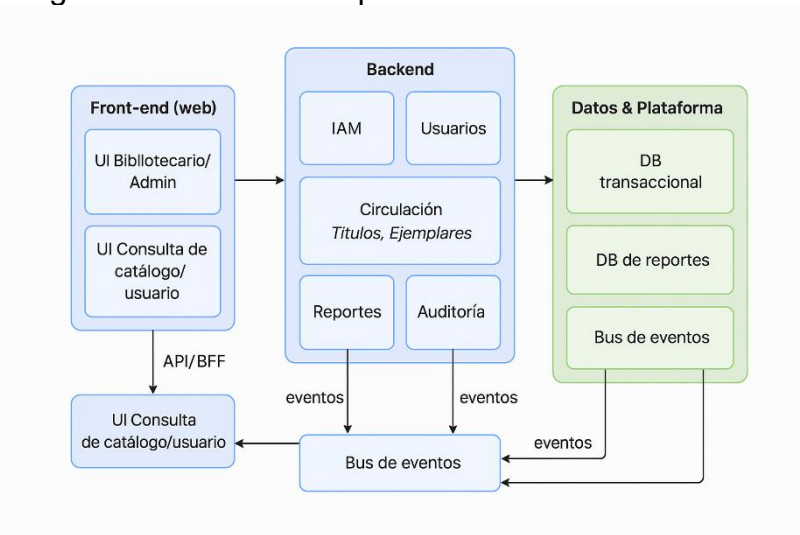
- RF15 Control de acceso por rol a operaciones críticas (condonar multa, baja de ejemplares).

2) Modularización (componentes propuestos)
Front-end (web)
- UI Bibliotecario/Admin.
- UI Consulta de catálogo/usuario (para ver disponibilidad e historial propio).

Back-end (modular y desacoplado por dominio)
1. **Identidad y Acceso (IAM)**
   Autenticación, autorización, gestión de roles y sesiones.
2. **Usuarios (Patrons)**
   Alta/edición de usuarios, estados, historial por usuario.
3. **Catálogo**
   Títulos (metadatos) y **Ejemplares** (inventario físico) como submódulo.
4. **Circulación**
   Préstamos, devoluciones, renovaciones, reservas, multas.
5. **Reportes**
   Generación y distribución de reportes; consultas analíticas.
6. **Auditoría**
   Registro inmutable de eventos/acciones administrativas.

Datos & Plataforma
- **DB transaccional** (relacional) para Usuarios, Catálogo y Circulación.
- **DB de reportes** (read model) optimizada para consultas y agregaciones.
- **Bus de eventos** para integrar módulos y poblar reportes.
- **Registro de auditoría** en tabla/almacen dedicado.

Diagrama textual de componentes:



3) Interfaces – Conectores
REST API (por módulo)
IAM
- POST /auth/login

- POST /auth/logout
- GET /me (roles, permisos)

Usuarios

- GET /users?role=&status=
- POST /users | PATCH /users/{id} | PATCH /users/{id}/status

Catálogo

- GET /titles?query=&author=&subject=
- POST /titles | PATCH /titles/{id}
- GET /titles/{id}/items
- POST /items | PATCH /items/{barcode} (estado: disponible, prestado, dañado, baja)

Circulación

- POST /loans (body: userId, itemBarcode, dueDate)
- POST /loans/{id}/return
- POST /loans/{id}/renew
- POST /holds (userId, titleId) | DELETE /holds/{id}
- GET /users/{id}/loans | GET /users/{id}/holds
- GET /items/{barcode}/status
- POST /fines (loanId, amount, reason) | POST /fines/{id}/waive

Reportes

- GET /reports/loans?from&to&groupBy=day|title|user
- GET /reports/overdues?asOf=YYYY-MM-DD
- GET /reports/catalog/stock
- POST /reports/schedules (crear programación de reporte)

Auditoría

- GET /audit?entity=Loan&entityId=...

Eventos de dominio (pub/sub)

- LoanCreated, LoanReturned, LoanRenewed, LoanOverdue
- HoldPlaced, HoldCancelled, HoldFulfilled
- ItemCreated, ItemStateChanged
- UserCreated, UserStatusChanged
- FineAssessed, FineWaived

Flujos clave (ejemplo)

- **Préstamo**: POST /loans → valida usuario/ejemplar → registra préstamo → emite LoanCreated → Reportes consume y actualiza métricas; Auditoría registra acción.
- **Devolución**: POST /loans/{id}/return → calcula multa si aplica → LoanReturned (+ FineAssessed si corresponde).
- **Reserva**: POST /holds → en devolución de un ejemplar, si hay cola, HoldFulfilled.

4) Análisis breve y justificación del diseño

- **Separación por dominios** (Usuarios, Catálogo, Circulación, Reportes, IAM) elimina el acoplamiento del monolito heredado y facilita el mantenimiento y la evolución de reglas de negocio específicas.

- **Arquitectura hexagonal por módulo**: puertos/adaptadores para exponer REST y consumir almacenamiento/eventos, manteniendo lógica de dominio independiente de frameworks.
- **Eventos de dominio + modelo de lectura para reportes (CQRS liviano)**: las operaciones transaccionales permanecen en la DB principal; los reportes se actualizan de forma asíncrona con eventos, evitando consultas pesadas sobre tablas operativas.
- **Auditoría append-only**: garantiza trazabilidad de operaciones sensibles (multas, bajas de ejemplares, condonaciones).
- **Compatibilidad con refactor incremental**: permite aislar primero **Circulación** (núcleo de negocio), luego **Catálogo**, y finalmente **Reportes**, manteniendo una fachada hacia el sistema legado mientras se sustituye módulo por módulo (patrón “strangler”).