Organización en 3 equipos

Ramas sugeridas: main (estable) · dev (integración) · trabajo diario en team-1/*, team-2/*, team-3/* + PRs a dev.
Convención commits: feat|fix|test|docs|refactor|chore: descripción.

Equipo 1 — Dominio & Contratos

Objetivo: definir el modelo del negocio y los contratos (interfaces) que usarán los otros equipos. Sin dependencias externas.

Paquetes / Carpetas

```
src/main/java/com/.../turnos/
  ├─ domain/            # entidades e invariantes
  │   ├─ Clinica.java
  │   ├─ Sucursal.java
  │   ├─ Profesional.java
  │   ├─ Paciente.java
  │   ├─ Especialidad.java
  │   └─ Turno.java
  ├─ domain/value/      # value objects
  │   ├─ DNI.java
  │   ├─ Email.java
  │   └─ Id.java
  ├─ validation/        # validadores de dominio
  │   ├─ TurnoValidator.java
  │   └─ PacienteValidator.java
  ├─ exception/         # excepciones de dominio
  │   ├─ DomainException.java
  │   ├─ ValidationException.java
  │   └─ NotFoundException.java
  └─ repository/        # CONTRATOS (interfaces)
      ├─ TurnoRepository.java
      ├─ PacienteRepository.java
      └─ ProfesionalRepository.java
```

Contratos mínimos (métodos guía)

```
interface TurnoRepository {
  Turno guardar(Turno turno);
  Optional<Turno> porId(Id id);
  List<Turno> porDiaYSucursal(LocalDate fecha, Id sucursalId);
  List<Turno> porProfesionalYDia(Id profesionalId, LocalDate fecha);
  boolean existeSolapamiento(Id profesionalId, LocalDateTime inicio, Duration
duracion);
  void eliminar(Id id);
}
```

Entregables

- Entidades con invariantes (p. ej., un Turno no puede iniciar en el pasado; sin solapes).
- *Repository definidos y documentados (JavaDoc) con ejemplos de uso.
- Tests unitarios del dominio (sin persistencia).

Criterios de aceptación
- Los tests de dominio pasan (./gradlew test).
- Ninguna clase en domain depende de service, persistence o app.
- Interfaces publicadas permiten a Equipo 2 y 3 trabajar sin bloquearse.

Equipo 2 — Persistencia & Config (Adaptadores)

Objetivo: implementar los repositorios definidos por Equipo 1. Empezar en memoria, luego CSV/JSON, y después JDBC (H2/SQLite).

Paquetes / Carpetas

```
src/main/java/com/.../turnos/
  └─ persistence/
      ├─ memory/
      │   ├─ InMemoryTurnoRepository.java
      │   ├─ InMemoryPacienteRepository.java
      │   └─ InMemoryProfesionalRepository.java
      ├─ file/
      │   ├─ CsvTurnoRepository.java
      │   └─ JsonTurnoRepository.java
      └─ jdbc/
          ├─ DataSourceFactory.java
          └─ JdbcTurnoRepository.java
src/main/resources/
  ├─ data-ejemplo/*.csv|json
  └─ sql/ (cuando llegue JDBC)
config/app.properties        # rutas de archivos, flags, etc.
```

Entregables por iteración
1. Iteración 1 (ahora): repos in-memory 100% funcionales + datos semilla.
2. Iteración 2: repos file (CSV/JSON) con lectura/escritura atómica y validaciones.
3. Iteración 3: JDBC (H2/SQLite) con DataSourceFactory, DDL y pruebas de integración.

Criterios de aceptación
- Implementaciones pasan los tests de Equipo 3 que usan los repos.
- persistence/* no depende de app; solo de domain/repository.
- Para file/JDBC: manejo de errores (IO/SQL) con excepciones claras y logs.

Equipo 3 — Servicios & Interfaz (CLI)

Objetivo: implementar casos de uso sobre los contratos de repos y entregar una

```
CLI utilizable.

Paquetes / Carpetas
  src/main/java/com/.../turnos/
    ├─ service/
    │   ├─ TurnoService.java         # crear/listar/cancelar/reprogramar
    │   ├─ PacienteService.java
    │   └─ AgendaService.java
    ├─ app/
    │   ├─ Main.java                 # entry point
    │   ├─ CliMenu.java              # menú principal
    │   └─ ConsoleIO.java            # util de entrada/salida
    ├─ dto/
    │   └─ TurnoDTO.java
    ├─ mapper/
    │   └─ TurnoMapper.java
    └─ util/
        └─ DateTimeUtils.java

Contratos de servicio (guía)
  class TurnoService {
    Turno crearTurno(Id sucursalId, Id profesionalId, Id pacienteId,
                     LocalDateTime inicio, Duration duracion);
    List<Turno> listarPorDiaYSucursal(LocalDate fecha, Id sucursalId);
    void cancelarTurno(Id turnoId);
    Turno reprogramarTurno(Id turnoId, LocalDateTime nuevoInicio);
  }

Entregables
  - CLI con opciones: Crear / Listar / Cancelar / Reprogramar (por
clínica/sucursal).
  - Tests de servicio y e2e contra repos in-memory.
  - application configurado (mainClass) y scripts de ejecución.

Criterios de aceptación
  - ./gradlew run permite la operación básica por consola.
  - Servicios usan solo interfaces de repos (no implementaciones concretas).
  - Tests de servicio cubren reglas de negocio y errores esperables.


Dependencias entre equipos (acuerdo)
  Equipo 3 (Servicios/CLI) --> usa --> repository.* (definido por Equipo 1)
  Equipo 2 (Persistencia)  --> implementa --> repository.* (Equipo 1)
  Equipo 1 (Dominio)       --> no depende de 2 ni 3 (núcleo independiente)


Backlog por iteración (sugerido)
  Iteración 0 (hoy)
    - Equipo 1: publicar entidades + TurnoRepository (firmas definitivas) +
tests.
```

    - Equipo 2: InMemory*Repository con datos semilla.
    - Equipo 3: TurnoService mínimo + Main + CliMenu con Crear y Listar.
  Iteración 1
    - Equipo 1: reglas extra (cancelación, reprogramación, ventanas de atención).
    - Equipo 2: Repos file (CSV/JSON) + config de rutas.
    - Equipo 3: CLI completa (Cancelar/Reprogramar + filtros) + tests e2e.
  Iteración 2
    - Equipo 2: JDBC (H2/SQLite) + migraciones SQL + pruebas integración.
    - Equipo 3: logging, manejo de errores amable en CLI, reporte simple.
    - (Opcional) Preparar módulo REST o UI según avance.

Roles de revisión y ramas
  Code owners
    - Equipo 1 → domain/*, validation/*, exception/*, repository/*
    - Equipo 2 → persistence/*, config/*, resources/data-*, resources/sql/*
    - Equipo 3 → service/*, app/*, dto/*, mapper/*, util/*
  Ramas de trabajo
    - Equipo 1: team-1/domain-contratos-*
    - Equipo 2: team-2/persistencia-*
    - Equipo 3: team-3/servicios-cli-*
  Definition of Done (DoD)
    - Build y tests pasan (./gradlew clean test).
    - Sin TODO críticos.
    - JavaDoc en interfaces y servicios públicos.
    - README/CHANGELOG actualizado si aplica.

Scripts de verificación (para todos)
  scripts/verificar-entorno.sh
    java -version
    ./gradlew --version
    ./gradlew clean test

Plan maestro paso a paso (de Fase 0 a Anti-patrones)
==========

Glosario mínimo
  - Contrato (interface): firma de métodos que otros implementan/usan. Permite trabajar en paralelo.
  - Repositorio (Repository): interfaz que define cómo accedemos a datos (guardar, buscar, listar).
  - DTO: objeto para transportar datos hacia/desde interfaz (CLI/REST) sin exponer dominio interno.
  - Invariante: regla que siempre debe cumplirse (p. ej., turno no empieza en el pasado).

```
  - E2E: prueba “extremo a extremo” (del menú a la persistencia real).
  - Seed: datos de ejemplo para probar rápido.
  - ADR: registro de decisiones de arquitectura (por qué elegimos X y no Y).

Convenciones
  - Ramas: main (estable), dev (integración), team-1/*, team-2/*, team-3/*.
  - Commits: feat|fix|test|docs|refactor|chore: descripción corta.
  - Paquetes: com.github.comechingones.turnos.<capa>.
  - DoD: build y tests verdes (./gradlew clean test), sin TODOs críticos,
JavaDoc pública, README/CHANGELOG actualizado.

FASE 0 — Kick-off y base de proyecto (Día 0–0.5)
  Equipo 1 (Dominio & Contratos)
    1) Crear docs/arquitectura.md, docs/decisiones-ADR/0001-elige-gradle.md.
    2) Acordar nombres de paquetes/capas.
    Entregable: documentos breves, estructura vacía.
  Equipo 2 (Persistencia & Config)
    1) Configurar Gradle con Java 21 toolchain y JUnit 5.
    2) Añadir .gitignore y scripts/verificar-entorno.sh.
    Entregable: build pasa, script corre.
  Equipo 3 (Servicios & CLI)
    1) Crear README.md inicial.
    2) Crear app/Main.java con “Hola” y application.mainClass.
    Entregable: ./gradlew run imprime “Hola”.
  Criterios F0
    - ./gradlew --version, ./gradlew clean test, ./gradlew run OK.
    - Docs/README mínimos en repo.
    - PRs a dev revisados por 1 reviewer.

FASE 1 — Publicación de contratos (Día 0.5–1)
  Equipo 1
    1) Crear domain/* con POJOs básicos e invariantes.
    2) Crear domain/value/* y validation/*.
    3) Crear repository/* interfaces.
    4) exception/* base.
    5) Tests de dominio.
    Entregable: contrato estable y entidades mínimas con tests.
  Equipo 2
    1) Stubs de persistence/memory/* que compilen.
    2) config/app.properties y resources/data-ejemplo/.
    Entregable: compila contra contratos.
  Equipo 3
    1) Stubs de service/* que llamen repos.
    2) app/CliMenu y ConsoleIO (navegación).
    3) dto/* y mapper/* (stubs).
    Entregable: CLI navega, sin lógica real aún.
  Criterios F1
    - Interfaces estables y documentadas.
    - Todo compila y corre menús.
    - ./gradlew test ejecuta tests de dominio.
```

```
FASE 2 — MVP en memoria (Día 1–2)
  Equipo 1
    - Afinar invariantes (solapamiento, duración mínima, horarios válidos).
    - Extender validadores y mensajes.
  Equipo 2
    - Implementar InMemory*Repository con índices y seed.
    - Tests unitarios de repos.
  Equipo 3
    - Implementar TurnoService (crear/listar) y conectar CLI.
    - Tests de servicio y E2E (CLI→service→in-memory).
  Criterios F2
    - ./gradlew run permite crear y listar.
    - Cobertura base ≥ 70% unidad/repos/servicios.
    - Solapamiento prevenido.

FASE 3 — Persistencia en archivo (CSV/JSON) (Día 2–3)
  Equipo 1
    - Definir estados: PENDIENTE, CONFIRMADO, CANCELADO.
    - ADR de “archivo primero”.
  Equipo 2
    - CsvTurnoRepository o JsonTurnoRepository.
    - Lectura/escritura atómica (tmp + rename).
    - config/app.properties: persistence=memory|file, rutaArchivos=...
    - Tests de integración (archivo temporal).
  Equipo 3
    - AppConfig: seleccionar implementación por config.
    - CLI: Cancelar y Reprogramar.
    - Mensajes de error claros.
  Criterios F3
    - Cambiando config corre en memory o file.
    - CRUD básico con archivo OK.
    - Logs mínimos IO.

FASE 4 — JDBC (H2/SQLite) (Día 3–5)
  Equipo 1
    - Modelo relacional (tablas, FKs, índices, UNIQUE anti-solape).
    - ADR: SQLite/H2 por simplicidad embebida.
  Equipo 2
    - DataSourceFactory y JdbcTurnoRepository.
    - resources/sql con DDL + seeds.
    - Transacciones, PreparedStatement, manejo de SQLException.
    - Tests integración con H2 en memoria.
  Equipo 3
    - AppConfig: selector jdbc.
    - CLI: import/export (opc.) y filtros.
    - Reporte simple por día/sucursal.
  Criterios F4
    - Operaciones en JDBC.
    - Índices/constraints evitan solapes.
```

```
    - Tests integración verdes.

FASE 5 — Reglas de negocio avanzadas (Día 5–6)
  Equipo 1
    - Políticas: reprogramación (≥24h), cancelación con estados, ventanas de
atención.
    - Actualizar validadores + tests.
  Equipo 2
    - Consultas eficientes, posibles migraciones SQL.
  Equipo 3
    - CLI: nuevas reglas (mensajes, bloqueos).
    - Tests E2E casos límite.
  Criterios F5
    - Reglas avanzadas aplicadas y testeadas.
    - Mensajes al usuario precisos.

FASE 6 — Logging, errores y DX (Día 6)
  Equipo 1
    - Jerarquía de excepciones y mensajes útiles.
  Equipo 2
    - Logs en persistencia (INFO/WARN/ERROR) y sanitización.
  Equipo 3
    - Manejo de excepciones en CLI; modo --verbose.
  Criterios F6
    - logback.xml con niveles por paquete.
    - Errores claros; stacktraces en verbose.

FASE 7 — Calidad, estilo y cobertura (Día 6–7)
  Todos
    - Checkstyle y SpotBugs en Gradle.
    - Jacoco: ≥ 80% unidad, ≥ 60% integración.
    - JavaDoc pública, limpieza de warnings y TODOs.
  Criterios F7
    - ./gradlew check verde.
    - PRs sin code smells graves.

FASE 8 — Empaquetado y entrega (Día 7)
  Equipo 2
    - ./gradlew installDist crea bin/turnos.
    - Config externa (app.properties) fuera del JAR.
  Equipo 3
    - scripts/run.sh y --config=/ruta/app.properties.
    - README final de uso.
  Equipo 1
    - CHANGELOG.md 1.0.0 y docs/arquitectura actualizados.
  Criterios F8
    - Se puede instalar y correr sin IDE.
    - README permite operar en 5 minutos.

FASE 9 — (Opcional) REST o UI (Día 8+)
```

```
  Variante REST (Spring Boot)
    - Módulo api-rest/ con controladores.
    - Reusar services y repositories.
    - Tests @WebMvcTest y @SpringBootTest (perfil H2).
    - Documentación OpenAPI.
  Variante UI (JavaFX)
    - ui-javafx/ con escenas de listado/creación/filtros.
    - Reusar services.

Checklist por equipo (resumen)
  Equipo 1 — Dominio & Contratos
    - Publicar entidades e interfaces repos (F1).
    - Invariantes + validaciones (F2–F5).
    - Estados y políticas (F3–F5).
    - Tests de dominio y ADRs.
  Equipo 2 — Persistencia & Config
    - Stubs InMemory (F1), InMemory real (F2).
    - File (CSV/JSON) conmutables por config (F3).
    - JDBC H2/SQLite + DDL + seeds + transacciones (F4).
    - Logs y manejo de errores IO/SQL (F6).
    - Distribución installDist (F8).
  Equipo 3 — Servicios & CLI
    - Stubs servicios + menús (F1).
    - MVP Crear/Listar (F2).
    - Cancelar/Reprogramar + filtros (F3–F4).
    - UX de errores + verbose (F6).
    - E2E + README final + scripts (F8).

Coordinación (para no pisarse)
  - Hito F1 (contratos publicados) ocurre rápido (en horas). 2 y 3 arrancan con
stubs.
  - Cualquier cambio de firma en repository/* se anuncia con ADR y PRs espejo.
  - Daily de 10' y tablero con Fases/Issues asignadas.

Anti-patrones a evitar
  - Acoplar service a una implementación de repo (siempre a la interfaz).
  - Lógica de negocio dentro de CLI o Repository (debe ir en service/domain).
  - Saltarse tests o logs.
  - Mezclar formato de archivo con reglas de dominio (separar capas).
```