Arquitetura de Aplicação Multiplataforma - Gestão de Horários de Aulas

Arquitetura Geral (MVC)

- Model (Data): PostgreSQL + Prisma/TypeORM.

- Controller/Service (Backend): Node.js + Express (TS).

- View (Frontends):

  - Web: React + Vite + MUI/Tailwind + Redux Toolkit.

  - Mobile: React Native (Expo) + Redux Toolkit.

1) Modelo de Dados (normalizado)

Entidades principais:

- lab (id, name, capacity, location)

- term (id, code, start_date, end_date)

- course (id, name)

- subject (id, name, course_id)

- instructor (id, name)

- class_group (id, subject_id, instructor_id, label)

- timeslot_template (id, weekday, start_time, end_time)

- reservation (id, term_id, lab_id, class_group_id, weekday, start_time, end_time, note)

- exception (id, reservation_id, date, action, new_lab_id, new_start_time, new_end_time)

ER Simplificado:

lab ——< reservation >—— class_group >—— instructor

                |

└─── term

class_group >─── subject >─── course

reservation ──< exception

```mermaid

erDiagram

  lab ||--o{ reservation : "possui"

  reservation }o--|| class_group : "aloca"

  class_group ||--|| instructor : "ministrado por"

  reservation }o--|| term : "pertence a"

  class_group }o--|| subject : "referente a"

  subject }o--|| course : "faz parte de"

  reservation ||--o{ exception : "pode ter"

## 2) Ingestão da Planilha (ETL)

- Identificar labs pelas abas da planilha.

- Mapear dias da semana (2ª a Sábado → 1–6).

- Ler faixas de horário (linhas: 07h40–08h30, etc).

- Parsear conteúdo da célula: disciplina, professor, curso.

- Criar reservas para o termo ativo.

- Validar conflitos e normalizar nomes.

## 3) Backend (NestJS/Express) – Services, Controllers & Endpoints

Endpoints principais:

- Labs: GET /labs, GET /labs/:id

- Term: GET /terms/active, POST /terms

- Schedule: GET /schedule/lab/:id, GET /schedule/day, GET /schedule/now, GET /schedule/search
```

- Reservations: CRUD completo

- Exceptions: POST /reservations/:id/exceptions

- Catálogos: cursos, disciplinas, professores, turmas

- Export: GET /export/lab/:id/ics, GET /export/term/:term/ics

Regras de negócio:

- Detecção de conflitos (horários sobrepostos).

- Consultas "Now" e próximas aulas.

- Exceções sobrepõem reservas recorrentes.

Autenticação/Autorização:

- JWT + roles (admin, coordenação, professor, viewer).

4) Frontend Web (React)

Rotas/Páginas:

- / : visão geral “Agora”

- /labs : lista de laboratórios

- /labs/:id : grade semanal (grid)

- /search : busca

- /admin : importação da planilha e CRUD reservas

Componentes: ScheduleGrid, TimeslotColumn, NowBar, LabCard, ReservationForm.

Estado: RTK Query + Redux slice para filtros.

Acessibilidade e dark mode incluídos.

5) Mobile (React Native/Expo)

Fluxos:

- Home: “Agora”

- Grade por laboratório (scroll por dia/horário)

- Busca por professor/disciplina

UI:

- FlashList ou SectionList

- expo-dev-client

- Deep linking

Offline opcional: sincronização com SQLite.

6) Estrutura de Pastas

/horarios-monorepo

/apps

/web

/mobile

/backend

7) Fluxo Dev → Prod

- Dev local: docker-compose (Postgres + Redis), npm run dev.

- CI: GitHub Actions → lint, test, build, publish containers.

- Prod: ECS Fargate/Railway, Postgres RDS, domínio + HTTPS, CDN web.

- Observabilidade: logs, métricas, Sentry.

8) Exemplos de Payloads

GET /schedule/lab/3?term=2025_1 → JSON com grade semanal.

POST /reservations → JSON cria reserva, upsert de entidades relacionadas.

9) Segurança, Qualidade e Extras

- Helmet + CORS restritivo no backend.

- Rate limit básico.

- Swagger com schemas e exemplos.

- Export iCal para labs/termos.

- WebSocket/SSE opcional para atualização em tempo real.