Remix Full Stack Project Structure

Complete app with Prisma, sessions, and nested routes. Production-ready architecture.

#remix #react #typescript #prisma #fullstack #auth

PNG PDF Copy Prompt

Project Directory

```
my-remix-app/
  app/
    root.tsx
    entry.client.tsx
    entry.server.tsx
    routes/
      _index.tsx
      _auth.tsx  Layout route fo…
      _auth.login.tsx
      _auth.register.tsx
      _app.tsx  Layout route fo…
      _app.dashboard.tsx
      _app.settings.tsx
      posts.tsx  Parent layout f…
      posts._index.tsx
      posts.$slug.tsx
      posts.new.tsx
      api.healthcheck.tsx  Resource route
    components/
      ui/
        button.tsx
        input.tsx
        card.tsx
      forms/
        login-form.tsx
        post-form.tsx
      layout/
        header.tsx
        sidebar.tsx
        footer.tsx
    lib/  Shared utilities
      db.server.ts  Prisma client
      session.server.ts  Cookie sessions
      auth.server.ts  Auth helpers
      utils.ts
    models/  Data access lay…
      user.server.ts
      post.server.ts
    tailwind.css
  prisma/  Database layer
    schema.prisma
    seed.ts
    migrations/
  public/
    favicon.ico
    images/
  tests/
    setup.ts
    routes/
      posts.test.ts
  package.json
  remix.config.js
  tailwind.config.ts
  tsconfig.json
  .env  DATABASE_URL, S…
  .env.example
  .gitignore
```

Why This Structure?

This structure separates concerns clearly: routes handle HTTP, models handle data access, and lib contains shared utilities. The `.server.ts` suffix ensures code only runs server-side. Layout routes (`_auth.tsx` , `_app.tsx`) group related pages with shared UI.

Key Directories

app/routes/_app.* - Protected routes sharing dashboard layout

app/models/ - Data access with `.server.ts` suffix

app/lib/ - Session, auth, and database utilities

prisma/ - Database schema and migrations

Naming Conventions

_auth.tsx - Pathless layout—no URL segment, shared UI

_auth.login.tsx - Child of _auth layout at /login

***.server.ts** - Server-only code, tree-shaken from client

api.*.tsx - Resource routes without UI

Session Auth Pattern

Remix uses cookie-based sessions. Create session in `lib/session.server.ts` with `createCookieSessionStorage` . Protect routes by checking session in the loader and redirecting if not authenticated. This avoids client-side auth complexity.

Getting Started

```
npx create-remix@latest --template remix-run/indie-sta
cd my-remix-app
npm run setup  # Creates DB and runs migrations
npm run dev
```

When To Use This

- Production web applications
- Apps requiring user authentication
- Content management and CRUD apps
- Teams wanting full-stack TypeScript
- Projects benefiting from progressive enhancement

Trade-offs

More setup - Database and session configuration required

Server required - Cannot deploy as static site

Learning curve - Nested routes and conventions take time