Nuxt 4 Full-Stack Project Structure

Full-stack Nuxt 4 with Nitro server, API routes, and Prisma. Production-ready architecture.

#nuxt #nuxt4 #vue #typescript #fullstack #prisma #nitro

PNG PDF Copy Prompt

Project Directory

```
my-fullstack-app/
  app/ Frontend source…
    app.vue Root component
    app.config.ts
    pages/ File-based rout…
      index.vue
      dashboard/
        index.vue Protected route
        settings.vue
      auth/
        login.vue
        register.vue
    components/ Auto-imported c…
      ui/ Base UI compone…
        Button.vue
        Input.vue
        Card.vue
        Modal.vue
      layout/
        Header.vue
        Footer.vue
        Sidebar.vue
    composables/ Shared composab…
      useAuth.ts Auth state and …
      useApi.ts Typed API fetch…
    layouts/
      default.vue
      auth.vue Minimal for aut…
      dashboard.vue With sidebar
    middleware/ Client-side rou…
      auth.ts Protect routes
    assets/
      css/
        main.css
  server/ Nitro server (a…
    api/ API routes
      auth/
        login.post.ts POST /api/auth/…
        logout.post.ts
        me.get.ts GET /api/auth/me
      users/
        index.get.ts GET /api/users
        index.post.ts POST /api/users
        [id].get.ts GET /api/users/…
        [id].put.ts
        [id].delete.ts
    middleware/ Server middlewa…
      auth.ts Auth verificati…
    utils/ Server-only uti…
      db.ts Prisma client i…
      auth.ts Auth helpers
  prisma/ Database schema
    schema.prisma Models and rela…
    migrations/
  shared/ Code shared bet…
    types.ts API request/res…
    validators.ts Zod schemas
  public/
    favicon.ico
  nuxt.config.ts
  package.json
  tsconfig.json
  .env DATABASE_URL, s…
  .env.example
  .gitignore
```

Why This Structure?

This structure separates frontend (`app/`) from backend (`server/`) clearly. Nitro provides file-based API routes where the filename becomes the HTTP method. Prisma handles database access with full type safety. The `shared/` folder enables type-safe contracts between client and server.

Key Directories

app/ - Frontend code: pages, components, composables

server/api/ - `API routes` —filename becomes HTTP method

server/utils/ - `Server-only code` (Prisma client, auth)

shared/ - `Type-safe contracts` between client and server

prisma/ - Database schema and migrations

API Route Naming

users/index.get.ts - `GET /api/users`

users/index.post.ts - `POST /api/users`

users/[id].get.ts - `GET /api/users/:id`

auth/login.post.ts - `POST /api/auth/login`

Getting Started

```
npx nuxi@latest init -t v4 my-app
cd my-app
npm install prisma @prisma/client zod
npx prisma init
# Configure DATABASE_URL in .env
npx prisma migrate dev --name init
npm run dev
```

When To Use This

- Building a SaaS or dashboard application
- Projects requiring authentication
- Applications with database requirements
- Full-stack teams using Vue.js
- Need type-safe API layer

Trade-offs

More folders - Complex structure to learn initially

Prisma lock-in - Can swap for Drizzle if preferred

Auth handling - Implement yourself or use nuxt-auth

Best Practices

- Keep `server/utils/` for database and auth only
- Use `shared/` for API contracts and validators
- Protect routes with `middleware/` , not in pages
- Use `useFetch()` with typed responses
- Run `prisma generate` after schema changes