Hono REST API Project Structure

Structured REST API with route modules, middleware, and validation. Works on edge and Node.js.

#hono #typescript #rest #api #middleware #validation

PNG PDF Copy Prompt

Project Directory

```
myproject/
  src/
    index.ts  Entry point
    app.ts  Hono app factory
    routes/  Route modules
      index.ts  Route aggregator
      users.ts
      posts.ts
      health.ts
    middleware/
      auth.ts  JWT or Bearer v…
      logger.ts
      error-handler.ts
    services/  Business logic
      user.service.ts
      post.service.ts
    schemas/  Zod schemas
      user.schema.ts
      post.schema.ts
    lib/
      db.ts  Database client
      env.ts  Environment con…
    types/
      index.ts
  tests/
    users.test.ts
    posts.test.ts
  package.json
  tsconfig.json
  wrangler.toml  For Cloudflare
  .dev.vars  Local secrets
  .gitignore
```

Why This Structure?

Organized Hono for real APIs. Routes are modular and composable using `app.route()`. Middleware handles cross-cutting concerns. Zod validates request/response schemas with full type inference. Works identically on edge and Node.js.

Key Directories

`routes/` - Each file exports a Hono sub-app mounted via `app.route()`

`middleware/` - Auth, logging, error handling composed per-route or globally

`schemas/` - Zod schemas for request validation and OpenAPI generation

`services/` - Business logic separated from HTTP handling

`lib/` - Database, environment, and shared utilities

Route Module Pattern

```
// src/routes/users.ts
import { Hono } from 'hono'
import { zValidator } from '@hono/zod-validator'
import { createUserSchema } from '../schemas/user.schema'
import { UserService } from '../services/user.service'

const users = new Hono()

users.get('/', async (c) => {
  const users = await UserService.getAll()
  return c.json(users)
})

users.post('/', zValidator('json', createUserSchema), asyn
  const data = c.req.valid('json')
  const user = await UserService.create(data)
  return c.json(user, 201)
})

export { users }
```

Getting Started

1. `npm create hono@latest`
2. `npm install zod @hono/zod-validator`
3. Create route modules in `src/routes/`
4. `npm run dev`

When To Use This

- Production APIs with 10+ endpoints
- Need request/response validation
- Teams working on the same API
- APIs that may run on edge or Node.js
- Projects needing OpenAPI docs

Trade-offs

`More files` - Route modules add navigation overhead

`Edge DB limits` - Workers need edge-compatible databases

`Learning curve` - Hono idioms differ from Express

Naming Conventions

`Routes` - `{resource}.ts` (users.ts, posts.ts)

`Schemas` - `{resource}.schema.ts` (user.schema.ts)

`Services` - `{resource}.service.ts` (user.service.ts)

Best Practices

- Use `zValidator` for all request validation
- Export sub-apps from route files, compose in `app.ts`
- Use Hono's `Env` type for typed environment bindings
- Add `@hono/swagger-ui` for API documentation
- Use `c.env` for runtime-specific bindings