HOW SOFTWARE WORKS · BUILDER · 15 MIN READ

Server routes: your first API endpoint

Why secret work happens on a server, how a Next.js route handler works, and how to build one safely: check the input, keep the key in an environment variable, and return JSON with the right status.

Pass the check (4 of 5) online to earn the “Endpoint Builder” badge:
https://digitalbuilders757.com/university/study/server-routes-your-first-api-endpoint

Browser code and server code

Everything in your front end is sent to every visitor's browser, where anyone can read it. Code on a server runs on computers you control: it can hold secret keys, talk to the database with full rights and call paid APIs. That's why our guides keep saying 'do it on the server'.

A server route (also called an API route or endpoint) is a URL your own app answers, like /api/ask. The page sends a request to it, your server code runs, and a response comes back.

A route handler in Next.js

In the Next.js App Router, a file named route.ts inside the app folder becomes an endpoint, and you export one function per HTTP method. app/api/hello/route.ts with export async function GET() { return Response.json({ message: "Hello" }) } answers GET /api/hello.

A POST handler reads what the page sent with const body = await request.json(). Express servers, serverless functions and Supabase Edge Functions follow the same pattern: a request comes in, your code runs, a response goes out.

Build it safely

- Route handlers are public: anyone can call them, not just your page. Check who's asking (sign-in) and whether they're allowed, on the server.
- Never trust the input. Check it's the shape and size you expect, and reject anything else with status 400.
- Read secrets from environment variables (process.env.MY_API_KEY) set in your host's settings, never from the code.

- Return the right status: 200 OK, 400 bad input, 401 not signed in, 403 not allowed, 429 too many requests, 500 your bug.
- Rate-limit anything that costs money, like an AI call.
- Send back only what the page needs: no secrets, no other people's data. Log the details on the server and show users a friendly message.

Calling it from your page

The page uses fetch: const response = await fetch("/api/ask", { method: "POST", headers: { "Content-Type": "application/json" }, body: JSON.stringify({ question }) }). Check response.ok before reading await response.json(), and show a message if something went wrong.

Open the Network panel in your browser's developer tools to watch each request: the method, the status, what was sent and what came back.

Key terms

Route handler
Server code that answers requests to a URL in your app, like route.ts for /api/hello in Next.js.

Endpoint
One URL an API answers, like /users/octocat, each doing one job.

Server
A computer (usually in the cloud) that runs your app's back end and answers requests.

Back end
Code on a server that checks permissions and does work users mustn't tamper with.

Environment variable
A setting or secret the app reads at run time, kept outside the code.

HTTP status code
A number a server sends back: 200 OK, 401 not signed in, 403 not allowed, 404 not found, 500 server error.

HTTP method
The verb of a web request: GET reads, POST creates, PATCH or PUT updates, DELETE removes.

JSON
A simple text format for data that apps send to each other: names and values in curly braces.

Rate limit
The most requests or tokens a key may use in a period, such as per minute.

Authentication
Checking who someone is (signing in).

Try it: Your first endpoint

1. Ask your coding agent (or v0) to add app/api/hello/route.ts that returns { message: "Hello" }, then open /api/hello in your browser.
2. Add a POST handler that takes { name } and returns a greeting. Reject a missing name, or one over 50 characters, with status 400.
3. Call it from a page with fetch, check response.ok, and show the reply.
4. Watch the request in the Network panel: method, status, payload and response.
5. Ask your AI helper: 'What could a stranger do with this endpoint?' Fix what it finds.

KNOWLEDGE CHECK

Server routes: your first API endpoint

5 questions. Circle one answer for each; 4 or more correct passes. Answers are at the back.

1. Why should a call to a paid AI API happen in a server route rather than in the browser?

A. Servers always answer faster than browsers, so users wait less
B. The secret key stays on the server, out of visitors' reach
C. Browsers aren't able to send requests to other websites at all
D. It makes the AI's answers more accurate

2. In Next.js, which file creates the endpoint GET /api/hello?

A. app/api/hello/page.tsx with a <form> on it
B. public/api/hello.json
C. app/api/hello/route.ts exporting a GET function
D. A GET function written inside the page's useEffect hook

3. Only your own page calls your /api/ask route. Do you still need to check the input?

A. No, because only your page knows the address
B. No, because the page already checked the form before sending it
C. Only if the route uses a database
D. Yes: anyone can call a route directly, so check every request

4. Someone sends your endpoint a 'name' that's 50,000 characters long. The best response?

A. Reject it with status 400 and a short message
B. Accept it, and trim it later when you display the name on the page
C. Return status 500 so it looks like the server crashed
D. Save it, since bigger inputs mean more data to learn from

5. Where should your route get the AI provider's key?

A. From a const in route.ts
B. From an environment variable, like process.env.AI_API_KEY
C. From the request body, sent by the page
D. From a comment in the code

Official sites

Next.js docs: Route Handlers: https://nextjs.org/docs/app/getting-started/route-handlers
Next.js docs: Next.js as a backend for your frontend: https://nextjs.org/docs/app/guides/backend-for-frontend

Last reviewed 2026-11-25. AI tools change fast; check each tool's own site for current features, plans and prices.

ANSWER KEY

Server routes: your first API endpoint

1. B: The secret key stays on the server, out of visitors' reach
Anything in browser code can be read and copied. On the server, the key stays in an environment variable.

2. C: app/api/hello/route.ts exporting a GET function
route.ts files in the app folder are route handlers; the exported function's name is the HTTP method it answers.

3. D: Yes: anyone can call a route directly, so check every request
Routes are public URLs. Browser checks can be skipped, so the server must check sign-in, permissions and input itself.

4. A: Reject it with status 400 and a short message
400 means 'your request was wrong'. Rejecting bad input early protects your database, your bill and your users.

5. B: From an environment variable, like process.env.AI_API_KEY
Keys in code end up in Git and shared screens. Environment variables live in your host's settings and are easy to rotate.