Building APIs

Terms

API endpoint
Data validation libraries
HTTP methods
HTTP status codes
Postman
Route handlers

Summary

- To build APIs, we add a route file (route.tsx) in a directory. Note that within a single directory, we can either have a page or a route file but not both.
- In route files, we add one or more route handlers. A *route handler* is a function that handles an HTTP request.
- HTTP requests have a *method* which can be GET (for getting data), POST (for creating data), PUT/PATCH (for updating data), and DELETE (for deleting data).
- HTTP protocol defines standard status codes for different situations. A few commonly used ones include: 200 (for success), 201 (when a resource is created), 400 (indicating a bad request), 404 (if something is not found), and 500 (for internal server errors).
- To create an object, the client should send a POST request to an API endpoint and include the object in the body of the request.
- We should always validate objects sent by clients. We can validate objects using simple if statements but as our applications get more complex, we may end up with complex and nested if statements.

- Data validation libraries, such as Zod, allow us to define the structure of our objects using a simple syntax, taking care of all the complexity involved in data validation.

- To update an object, the client should send a PUT or PATCH request to an API endpoint and include the object in the request body. PUT and PATCH are often used interchangeably. However, PUT is intended for replacing objects, while PATCH is intended for updating one or more properties.

- To delete an object, the client should send a DELETE request to an API endpoint. The request body should be empty.

- We can use Postman for testing APIs. With Postman we can easily send HTTP requests to API endpoints and inspect the responses.

Creating Route Handlers

```
export async function GET(request: NextRequest) {
  const body = await request.json();

  return NextResponse.json(body, { status: 200 });
}
```

