# Technologies

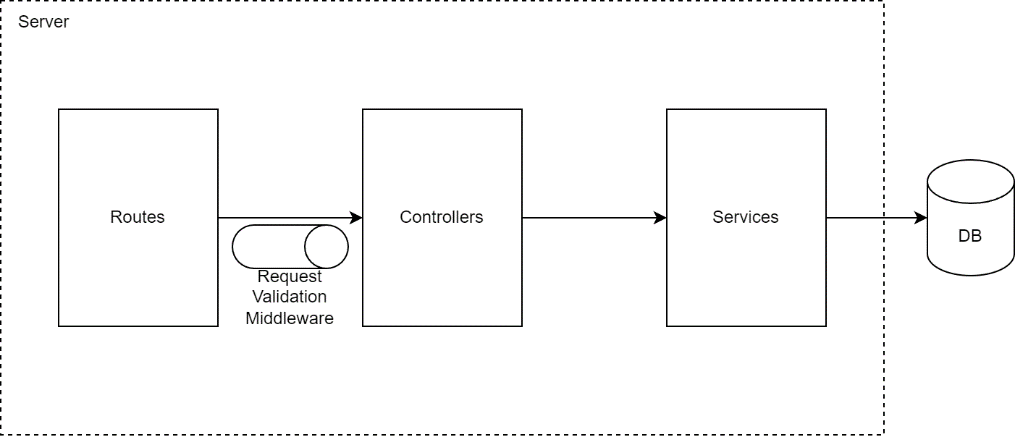
For this assessment, I will be using NodeJs and Express to build the API, Zod for validating requests, Node-Postgres for connecting to the database and Typescript.

# Layers

My solution consists in the following modules:

* **Route** – Each route consists of the pair of path and http method and matches the pair to the correspondent controller.
* **Controller** – The controller will extract the necessary data from the request and will execute the logic of the route it corresponds. In the controller it’s not necessary to verify the existence of parameters nor if they follow the convention defined, this is because using a package called Zod we can verify the schema of the request and return an error in case there is any parameter missing.
* **Service** – The service will execute the connection to the database and do the different operations, from fetching data to updating rows.

Usually a Controller, Service or a different module do the verifications on whether the request has the correct data, however, as I’m using Zod we can remove that need, adding this verification as a middleware, ending up with the following project architecture:



# Data model

From the prompt, I can extract the following tables:

* **Route** – Represents a course. This table has the following fields:
  + Identifier – Unique identifier.
  + Name – For easier identification from the user.
  + Description – For an in-depth explanation of the route.
  + Start Location – The initial location for the route, represented by a string. [[1]](#Note1)
  + End Location – The end location for the route, represented by a string. [[1]](#Note1)
* **Landmark** – Represents interesting points that will appear through the routes. As landmarks aren’t specific to a single route, this must be a separate table. This table has the following fields:
  + Identifier – Unique identifier.
  + Name – Name of the landmark.
  + Description – Description of the landmark.
  + Longitude – Numeric value representing the longitude in which the landmark is located (part of the extra functionality of the 2nd phase). [[2]](#Note2)
  + Latitude – Numeric value representing the latitude in which the landmark is located (part of the extra functionality of the 2nd phase). [[2]](#Note2)
* **Route Landmark** – Represents the junction between Route and Landmark. Besides the Route and Landmark identifiers, it also has the following field:
  + Highlight – Defines whether a Landmark is a highlight for that Route.

**NOTES**:

1. The start location and end location of a route are defined as strings. An alternative would be to abstract these values, creating a table, manually adding predefined values and add these values as foreign keys in the Route table.
2. The longitude and latitude will have to be manually verified and inserted, since in this assessment there are no external APIs being used that could be utilized to fetch these values.

Having structured what tables will exist, we can formulate the ER model:

