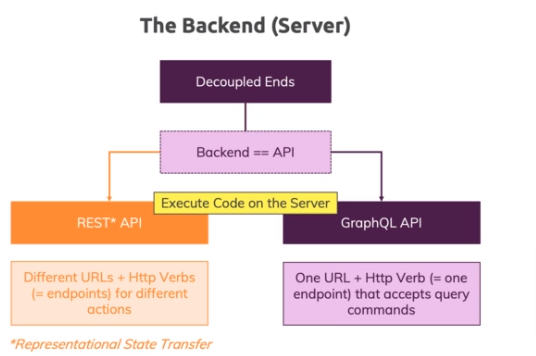
**Back End (Server Side)**

****

**Decoupled Ends:** Which means we have the frontend, powered by React and with the backend powered by Node Express and then we also have the database server with MongoDB.

Now because of that decoupled ends, thing, our backend is built as an API, which stands for Application Programming Interface.  
API describes thing which expose certain entry points we could say, which other things can use.  
For example if you’re building a third-party library that should do some user input validation, you might offer a certain functions the application that uses your library can call in a certain way to use your library. So basically, any library you create needs clearly defined entry points and a way of using them so that other users who did not build your library can interact with it and for our backend, its’ basically the same.

We build a Node Express application which define some entry pointes, some ways of communication with it and only these ways are supported thereafter.

If some client which tries to talk to our backend, for example the react app later wants to interact with some entry point we didn’t define, it will get an error.

**APIs:**

There are two major kinds of APIs which we can build when we build such a backend.

You can build a **REST API** where REST stands for Representations State Transfer.

And we can build **GraphQL API**.

Now both kinds of APIs can be built with any server side language, not just with Node.js and both kinds of APIs can do anyting on the server, store data in the database, validate user input, get data from database and so on.

Now they work differently, when it comes to how requests are received or how requests which are sent to the API should be formatted.

A REST API which is by far the most common and used form of backend web API uses a combination of different URLs or paths which are the things after the domain and HTTP verbs, which is this GET, Post, Patch, Delete things you might have heard of, to build so –called end points which trigger different actions.

For GraphQL API there we have one URL and one HTTP verb, typically a POST request, so we have one endpoint which then in turn however accepts query commands.  
So, since this HTTP verb is a POST request, when you build a GraphQL API, the body of the request contains a query expressions that adheres to the GraphQL standard in the end, which describes the operation you want to perform.

Still on the server sie when you build such an API, you define which query commands you want to support. So you still don’t support everything but you don’t work with different path ver combinations but instead with that query langauges to trigger different actions and so on.

Now in both scenarios, we execute code on the server and in both scenarios, we don’t directly talk to the database.

From your React app, you’ll always send requests to your Node Express app no matter if that’s built as a REST API or a GraphQL API, it just influences how the request look what you send. But in both cases you talk to your Node Express app will then do something based on the action which is triggered because your path verb combination or because of your command and then it’s your Node Express server which will talk to a database.