## Why would you consider a Scripting Language as JavaScript as your backend platform

To simplify the process of going from frontend to backend, and use the same language for both parts. Node.js is also faster than Java so that is another thing to consider when choosing what language to use for your backend.

## Explain Pros & Cons in using Node.js + Express to implement your backend compared to a strategy using for example Java/JAX-RS/Tomcat

### Pros:

* Using the same language for both frontend and backend. This is great as it makes the same programmer capable of writing both the frontend and the backend without extensive knowledge in two different languages.
* Database queries. Related to the first pro, as for writing database queries you don’t have to change language into SQL. This is due to NOSQL databases and that newer databases like CouchDB are written in JavaScript.
* JSON which stands for JavaScript object notation is easy to use in Node.js as no parsers are needed to change the format of the code. This is needed in Java when you get a query back from a database or a rest API.
* Node.js is faster than Java as it doesn’t set up separate threads so there is no overhead to slow anything down.

### Cons:

* New. When using Java you have more than 20 years of testing that the Java Virtual Machine works as it’s supposed to. You don’t have that certainty when working with Node.js
* The IDEs for Node.js is not as great as they are for Java. Both Netbeans and Ecplipse are great tools with debuggers, decompilers and servers. Writing code for Node.js is done in a text editor by many programs and even though Webstorm is a good IDE, it’s just not as well developed as Netbeans or Eclipse.
* Debugging is a lot easier to do in a language like Java as it’s been used by big companies that demand they can fail proof their systems, and a good debugger is needed for that.
* Java libraries are simply better as they have been worked on for longer. There are great libraries for JavaScript but the depth and quality of the Java code base is superior.
* Deadlocks. If one request is slow it slows down everything else. This is because JavaScript doesn’t use threads. You hardly ever notice this after the new V8 engine for JavaScript

## Explain using a relevant example your strategy for implement a REST-API with Node/Express and show how you can “test” all the four CRUD operations programmatically using for example the Request package

The request package makes it possible to make http calls. This is very useful when testing CRUD operations as CRUD operations are bound to specific urls that can be called via http calls.

## Explain using relevant examples about testing JavaScript code, relevant packages (Mocha etc.) and how to test asynchronous code

Mocha is a test framework made for JavaScript. To use Mocha you need to use an assertion library with it, in my case I’ve used Chai, but it’s also possible to use should.js, expect.js and more.

When using the Mocha framework the first thing you want to do is describe what you want to test. That is done with the describe(‘What needs to be tested’, function()). To actually test something the “it” notation should be used. This will look like this: it(“will do something”, function(done)). It’s important to use the done() call after each test otherwise the test will never proceed to the next test in line.

In the it notation an expect notation should be used to actually write what you expect to happen. As an example it could be expect(“something”).to.be.equal(“something”).

## Explain using relevant examples concepts related to the testing of a REST-API using Node/JavaScript + relevant packages

Check RestAPI.

## Explain using relevant examples different ways to mock out databases, HTTP-request etc.