**Backend Roadmap to Master NodeJS –**

[Dash ICT](https://medium.com/@dash.ps?source=post_page-----c508742ab74f--------------------------------) Jun 7, 2022

Web Development has taken over the world, developers no longer strive to improve their knowledge in building desktop applications, it is exhausting indeed to build 3 versions of the same application for 3 different environments.

Since the establishment of SaaS (Software as a Service), web applications are now the de facto technology for creating any new product.

Products can be very complicated and very difficult to create, that’s why we have multiple roles: Front end, Backend, QA, Database Engineer, DevOps, UI/UX and many more fancy job titles.

In this article, my colleague[Umar Salah](https://medium.com/@umars.salah) and I ([Hatem Saadallah](https://medium.com/@hatem.saadallah)) will try to break down the Backend Developer scary requirements into small, digestible pieces to make it easier for new aspiring back end developers to get right on track and self-learn their ways into successful Engineers.

This article is supported by, and based on[dash.ps](http://dash.ps/) company stack. Some technologies that will be mentioned might not be used by other companies, but in general these technologies can be used to create and ship large scale products.

So brace yourself, and let’s get started.

*Note: This is a reference article and is designed as such, you can bookmark it and constantly visit it once you finish each step.*

**Prerequisites:**

In order to get the most out of this article, please make sure that you have the basic requirements in order to benefit from it.

* Basic Web Development tools: HTML, CSS , and JavaScript.
* Basic Understanding of Networking
* Basic Understanding of Computer Operating Systems.

You can search on the internet, read some articles, do some small projects, and get back to this article ready to get some real work done.

**Introduction to NodeJS:**

JavaScript has existed long before NodeJS, but it was only used as a front end language directly compiled on the browser, until 2009, when Ryan Dahl decided to make it finally a back end language, and the rest is history.

***Actions:***

Learn some advanced JS Concepts

* Asynchronous Functions
* Namespaces
* Scope and Execution Context
* Garbage Collection
* Follow the tutorial in this[FreecodeCamp Article](https://www.freecodecamp.org/news/before-you-bury-yourself-in-packages-learn-the-node-js-runtime-itself-f9031fbd8b69/)
* Install NodeJS
* Become familiar with package management (NPM)
* Create a simple project in NodeJS (Handle routes and render some pages).

**Relational Databases:**

Databases are the cornerstone of any back end application, the data that you receive has to be stored somewhere. Otherwise, it is just simple input/output route handling :).

Your job in this part is the become familiar with any famous DBMS (Database Management System); some famous examples:

* MySQL
* PostgreSQL
* MariaDB
* MS SQL Server
* Oracle SQL
* and SQLite

My personal recommendation is MySQL, it is the most famous DBMS out there, it scales well, and can handle millions of rows of data efficiently. It is also open source, so it is a favorable option for companies as it costs no money.

**ExpressJS:**

ExpressJS is the most famous NodeJS framework, it is un-opinionated, which means there are no predefined code structure, no best practices, it is up to the company or the team to decide what is best for their needs. This of course gives freedom for developers, but can be confusing for new developers who join the team as they have what are the practices of their team.

* Here’s the MySQL documentation:[HERE](https://dev.mysql.com/doc/)
* And here’s the official documentation for ExpressJS:[HERE](https://expressjs.com/en/guide/database-integration.html#mysql)

**Book Directory API Using Express JS**

It is time for your first project.

Please, read the project specification[here](https://hackmd.io/@NsgF3MelQ1yF1bCvC1-2Nw/rJtwhT4_c).

Use[Postman](https://www.postman.com/) for testing the API.

The stack needed:

* JavaScript
* ExpressJS
* MySQL/PostgresQL
* Sequelize

You need to have an idea about some ExpressJS Concepts ([Routing](https://expressjs.com/en/guide/routing.html),[Middleware](https://expressjs.com/en/guide/writing-middleware.html),[Error Handling](https://expressjs.com/en/guide/error-handling.html)).

**Introduction to NestJS**

NestJS is a NodeJS framework based on ExpressJS and Fastify. By default, NestJS uses ExpressJS under the hood, but you can specify whatever NodeJS framework manually.

*Key Points:*

1. Learn about NestJS and different types of JS Frameworks.
2. Why NestJS & not other JS Frameworks like Fastify.
3. Read the first 2 sections of the nestjs documentation (overview, & fundamentals)
4. Familiarize yourself with TypeScript language.

* [Official Documentation](https://docs.nestjs.com/)
* [Sample Applications](https://github.com/nestjs/nest/tree/master/sample)
* [Unofficial Tutorial](https://wanago.io/courses/api-with-nestjs/)

Your job is to read about NestJS & answer the questions above.

***Sequelize:***

Writing SQL queries can be tricky, it can be not well optimized, and it can be very long when you have a lot of join statements. Fortunately, we have ORM (Object Relational Mapper).

It is basically a framework for writing database commands in your own programming language.

Sequelize is the most famous ORM framework.

Check out the documentation and tinker around with it:[HERE](https://sequelize.org/docs/v6/getting-started/)

**TODO List Application (Authentication)**

Your first project in NestJS, this project is of course a TODO List, it is the Hello World for web development, this project is partitioned into sections, the first part of it is creating a user authentication system.

*Key Points:*

1. General Structure for a NestJs project.
2. Add an Authentication System Using JWT.
3. Database Schema using[DBDiagram](https://dbdiagram.io/home)

*Readings:*

* [Nestjs Authentication](https://docs.nestjs.com/security/authentication)
* [DB Diagrm Tutorial](https://www.youtube.com/watch?time_continue=254&v=rx6CPDK_5mU&feature=emb_logo)

**TODO List Application (CRUD Operations)**

The actual project now is on. Read the detailed description[HERE](https://hackmd.io/@NsgF3MelQ1yF1bCvC1-2Nw/B1ACx1S_5)

Also, check out[this](https://docs.nestjs.com/guards)

**TODO List Application (Using MongoDB)**

Change the TODO List DB into MongoDB using TypeORM

*Readings:*

* [MongoDB Documentation](https://www.mongodb.com/docs/)
* [NestJS Mongoose](https://docs.nestjs.com/techniques/mongodb)
* [Typeorm MongoDB](https://orkhan.gitbook.io/typeorm/docs/mongodb)

**Introduction to AWS**

AWS (stands for Amazon Web Services) is as the name indicates a set of software solutions provided by Amazon to help businesses remove the overhead cost of setting up basic services, like hosting and domains, security and the parts related to availability and data redundancy.

AWS provides PaaS (Platform as a Service), this can help the business hit the ground running when it comes to setting up the environment and focusing on the business.

Graphical user interface, PowerPoint

Description automatically generated

AWS Components

Imagine that a Startup with very limited resources has to purchase a physical server, hire a systems engineer and worry about backups, electricity and security; most businesses will not be able to afford it, plus that when the business fails, it’s going to be too much wasted resources.

Amazon came up with an ingenious solution because in 2006, instead of purchasing a server, you can purchase an EC2, which is a virtual server that exists as a part of Amazon servers. It acts and feels exactly like a normal server, but it’s easily expandable, highly available and relatively cheaper. A new business doesn’t have to buy a new server, they can purchase small specs and expand them once the scale-up is necessary.

*Key Points:*

1. What is AWS?
2. What are the services provided by AWS?
3. What is the difference between Server & Serverless?
4. Why’s AWS become so popular among businesses

A nice video to watch:

[Top 50+ AWS Services Explained in 10 Minutes](https://www.youtube.com/watch?v=JIbIYCM48to)

**Introduction GraphQL**

GraphQL is a new system of API created by Facebook to fix a long running problem with the conventional RESTful APIs. The front-end developer can query exactly the fields they want, nothing more and nothing less.

Diagram

Description automatically generated

Let’s think of an example of the TODO List Application, there is a table for the user data with the hashed password, so the frontend developer will query the respective route, and perform the operation, when the user is logged in, the front-end developer will query the tasks for the respective user, which is in another route.

This could pile up into multiple routes and queries. Sometimes, it takes 5 queries to get all the needed data from the backend server.

GraphQL can make it much easier, with a single route endpoint, the front-end developer can query exactly all the needed data.

*Key Points:*

1. What is GraphQL?
2. What’s the difference between GraphQL & RESTful APIs?
3. What is AWS AppSync? And how can it be used in systems and production environments?

*Links:*

* [Learn GraphQL](https://graphql.org/learn/)

*Videos:*

* [What is GraphQL ?](https://www.youtube.com/watch?v=eIQh02xuVw4)

**TODO List Application (GraphQL)**

Create A GraphQL schema for your Database Schema.

*Links:*

* [Tutorial: Designing a GraphQL API](https://github.com/Shopify/graphql-design-tutorial/blob/master/TUTORIAL.md)

**Project : Social Media Simulator**

Read the project description and implement your own version:[Project Description 4](https://hackmd.io/@NsgF3MelQ1yF1bCvC1-2Nw/ryiTj1BOq)

**AWS Cognito**

AWS Cognito is one of the services provided by Amazon, its sole job is to provide authentication for the developer’s application without having to store passwords in the database or worry about hashing or dehashing. Using its API, the developer sends the credentials and they’re going to be stored/retrieved automatically, this of course saves time and provides a safer environment to store sensitive data.

Diagram

Description automatically generated

*Task:*

Convert the authentication system in your TODO App to Cognito

*Resources:*

* [What is Cognito ?](https://aws.amazon.com/about-aws/whats-new/2014/07/10/introducing-amazon-cognito/#:~:text=Amazon%20Cognito%20is%20a%20simple,users%20across%20their%20mobile%20devices.)
* [Cognito API Reference](https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/Welcome.html)

*Videos :*

* [What is cognito ?](https://www.youtube.com/watch?v=abTy-Yyo6lI)

**Lambda Functions**

Read about AWS Lambda Functions:[HERE](https://aws.amazon.com/lambda/)

Try to implement a lambda function in your TODO Application. Make it auto verifies the user from Cognito without having to verify through email.

**Conclusion:**

I know this is rather intensive and a huge article, but it is the complete roadmap for learning and deploying a backend application on a high level of professionalism, it is the roadmap for new trainees implemented by dash.ps. It contains all of the stack, you might find it slightly different from one company to another, some companies use express js without a high layer framework like NestJS. Some companies might use different frameworks, but you’ll find that the concepts are very similar.

The time frame for this road map is 12 weeks, so save up this article and return to it constantly for reference while you are on your path to become a Backend Ninja.