kamranahmedse/developer-roadmap

kamranahmedse

Back-end

1 **Pick a Language**
There are myriads of different options

- Scripting Languages
  - Python
  - Ruby
  - PHP
  - Node.js
  - TypeScript (Optional)
- Functional Languages
  - Elixir
  - Scala
  - Erlang
  - Clojure
  - Haskell
- Other Options
  - Golang
  - Rust
  - Java
  - .NET

For the beginners, if you are just getting into backend development, I would recommend you to pick one of the scripting languages. For the quick-and-easy, go with **Node.js** or **PHP**. If you have already been doing backend, with some scripting language then don't pick another scripting language and have a look at Golang, Rust or Clojure, it will definitely give you a new perspective.

2 **Practice what you learnt**
Excercise and make some command line applications with your picked language

Sample Ideas
Implement some command that you use e.g. `ls`
Command that fetches and saves reddit posts on /r/programming
Command that gives you directory structure in JSON format
Command that reads JSON from above and creates director structure
Think of some task that you do every day and try to automate that

3 **Learn Package Manager**
Learn how to use package manager for the language that you picked, e.g. **PHP has composer, Node.js has NPM and yarn, Python has pip, Ruby has gems** etc

Package managers help you bring external dependencies in your application and to distribute your own packages.

Make sure to read about the best practices for security. Read the OWASP guidelines and understand different security issues and how to avoid them in language of your choice

4 **Standards and Best Practices**
Each of the language has its own standards and best practices of doing things. Study them for your picked language. For example **PHP has PHP-FIG and PSRs**, with Node.js there are many different driven by community etc.

There are several different options, each having different uses, depending upon the language of your choice. **Google** around, see different options and pick the one suitable for your needs.

For PHP – PHPUnit, PHPSpec, Codeception
For Node.js – Mocha, Chai, Sinon, Mockery, Ava, Jasmine

For others, I don't want to start any flamewars so I am not going to make any recommendations here, so look around and find the ones suitable

5 **Make and Distribute Some Package/Library**
Now go ahead and create a package and distribute it for others to use, and make sure to follow the standards and best practices that you have learnt this far.

Contribute to Some Opensource Project
Search for some projects on github and open some pull requests in opensource projects. Some ideas for that :

Refactor and implement the best practices that you learnt
Look into the open issues and try to resolve
Add any additional functionality

6 **Learn about Testing**
There are several different testing types, but for now learn about how to write **Unit and Integration tests** in the language you picked. Understand different testing terminologies such as mocks, stubs etc

Learn how to calculate test coverage

7 **Write Tests for the practical steps above**
Go ahead and write the unit tests for the practical tasks that you implemented in the steps before.

8 **Learn Relational Databases**
There are several options here. However if you learn one, others should be fairly easy. Pick **MySQL** for now but learn how they are different and the usecases

- Oracle
- MySQL
- MariaDB
- PostgreSQL
- MSSQL

9 **Practical Time**
Create a simple application using everything that you have learnt this far. It should have registration, login and CRUD. Create a blog, for example. Where anyone can register and get a public profile page create, update and delete posts and public page will show the posts created by them.

Make sure to write tests, follow the standards and best practices. Also for the database, add the indexes, use proper storage engines and make sure to analyze the queries before using them in the application.

10 **Learn a Framework**

Depending upon the project and the language you picked, you might or might not need a framework. There are several different options

For PHP – Laravel or Symfony and Slim or Lumen for micro-frameworks
For Node.js – Express.js, Hapi.js
For Golang – I prefer to code without framework

For others, search and find the suitable ones for the language you picked

11 **Practical time**
Make the same application you made in 9 to the framework of your choice

12 **Learn a NoSQL Database**
First understand what they are, how they are different from relational databases and why they are needed. There are several different options Have a look at different options and see how they differ. If you have to pick one, pick **MongoDB**

Learn MongoDB for now but make sure to look how it compares with others

- MongoDB
- RethinkDB
- Cassandra
- Couchbase

Once you have learnt, implement caching strategy in application you built in step 11

- Memcached
- Redis

13 **Caching**
Learn how to implement app level caching using Redis or Memcached

14 **Creating RESTful APIs**
Understand REST and learn how to make RESTful APIs and make sure to read the part about REST from the original paper of **Roy Fielding**

- OAuth
- Basic Authentication
- Token Authentication
- JWT
- OpenID

15 **Authentication/Authorization Methodologies**
Learn about the differences and how to implement them

16 **Message Brokers**
Learn about the message brokers, understand the "Why" and pick one. There are multiple options but I would go for **RabbitMQ** or **Kafka**. Learn how to use RabbitMQ for now, if you want to pick one.

- RabbitMQ
- Kafka

17 **Learn a Search Engine**
As the application grows, simple queries on your database aren't going to cut it out and you will have to resort to a search engine. There are multiple options, each having it's own differences.

- ElasticSearch
- Solr
- Sphinx

18 **Learn How to Use Docker**

- Apache
- Nginx
- Caddy
- MS IIS

19 **Knowledge of Web Servers**
There are several different options here, look at the different options, understand their differences and limitations

20 **Learn how to use Web Sockets**

21 **Learn GraphQL**
While it is not required, feel free to have a look at it and see what it is all about and why they are calling it the new REST

22 **Look into Graph Databases**
Again not required but you should have a little understanding of what they have to offer

23 **All the things that weren't mentioned above**
Profiling, Static Analysis, DDD, SOAP. Go Figure!

Keep Exploring