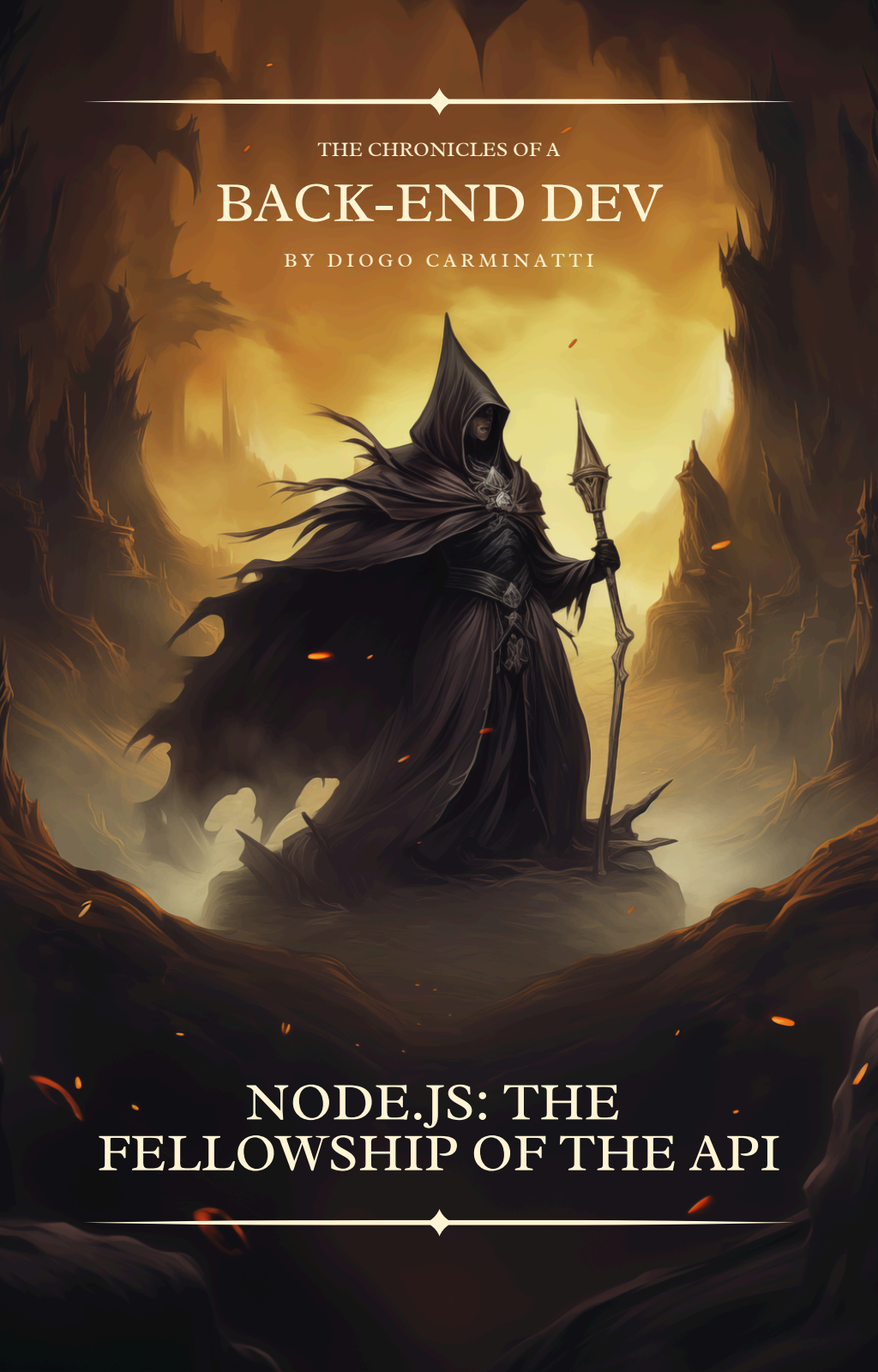
THE CHRONICLES OF A

BACK-END DEV

BY DIOGO CARMINATTI



NODE.JS: THE FELLOWSHIP OF THE API

Introduction

In the dawn of the digital age, when the web was in its infancy, developers sought a way to connect different parts of the vast digital world. It was then that APIs emerged, magical doors allowing applications to communicate with each other. At the heart of this magic lies Node.js, a powerful tool that enables the creation of robust and efficient APIs.

Before the journey

Before embarking on our journey, we need to prepare our environment. First, we will install Node.js. Visit the official site (nodejs.org) and download the latest version. After installation, confirm that everything is working, in a terminal type:

```bash
node -v
npm -v
```

If you don't have node in your machine, follow the link to install:
Windows
Linux

Forging the server

Let's start by creating a simple server. In a new folder, create a file named server.js and add the following code:

server.js

```
const http = require('http');

const server = http.createServer((req, res) => {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello, world!');
});

server.listen(3000, '127.0.0.1', () => {
    console.log('Server running at http://127.0.0.1:3000/');
});
```

This is the starting point, where our server is born, ready to listen for requests on port 3000.

Invoking the Express

To simplify our work, we will use Express, a library that makes creating APIs easier. Install Express with npm:

bash

```bash
npm install express
```

Next, let's modify our server.js to use Express:

server.js

```js
const express = require('express');
const app = express();

app.get('/', (req, res) => {
    res.send('Hello, world with Express!');
});

app.listen(3000, () => {
    console.log('Express server running at http://localhost:3000/');
});
```

The routes of Middle Earth

Let's create some routes for our API. Routes are magical paths that direct requests to the correct responses.

server.js

```javascript
// ... code

app.get('/hobbits', (req, res) => {
    res.json([{ name: 'Frodo' }, { name: 'Sam' }]);
});

app.post('/hobbits', (req, res) => {
    // Logic to add a new hobbit
    res.status(201).send('Hobbit created!');
});

// ... code
```

The power of MongoDB

To store data persistently, we need a database. We will use MongoDB. First, install mongoose, a library for interacting with MongoDB:

```bash
npm install mongoose
```

Now, connect to MongoDB and define a data model:

```js
const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/middleearth',
{ useNewUrlParser: true, useUnifiedTopology: true });

const Hobbit = mongoose.model('Hobbit', { name: String });

app.post('/hobbits', (req, res) => {
    const newHobbit = new Hobbit({ name: 'Pippin' });
    newHobbit
    .save()
    .then(() =>
    res.status(201)
    .send('Hobbit created with MongoDB!'));
});
```

Conclusion

And so, dear reader, we come to the end of this initial journey in creating an API with Node.js. We have laid the foundations of a vast and powerful system. Explore, experiment, and create your own adventures in the digital Middle-earth.