

Node.js

Criando Web APIs



Introdução ao Node.js

import

```
const Baby =  
require ( './Baby.js' );  
  
const baby = new Baby ();  
baby.cry ();
```

```
// class Baby.js  
class Baby {  
    name = 'Enzo';  
    cry(yes = true) {  
        Console  
            .log('Crying..');  
    }  
}  
  
module.exports = Baby;
```

Introdução ao ExpressJS

O que é?

ExpressJS é um framework open-source minimalista para Node.js

- Utilizado para criação de aplicações e APIs
- Framework popular
- Utiliza o padrão MVC

Instalação

Softwares

- Node.js
- Npm ou Yarn

```
$ yarn init
```

```
$ yarn add express  
body-parser  
path
```

Hello, World!

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');

const app = express();

app.get('/', (req, res) => {
});

app.listen(3000, () => {
  console.log('Server running on port 3000.');
```

Hello, World!

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');

const app = express();

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

app.listen(3000, () => {
  console.log('Server running on port 3000.');
```


Hello, World!

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');

const app = express();

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

app.listen(3000, () => {
  console.log('Server running on port 3000.');
```

Hello, World!

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');

const app = express();

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

app.listen(3000, () => {
  console.log('Server running on port 3000.');
```

Middleware

```
[...imports]
```

```
const app = express();
```

```
const logger = (req, res, next) => {  
  console.log("Request!");  
  next();  
};
```

```
app.use(logger);
```

```
[...routes]
```

```
[...serve]
```

Middleware

```
[...imports]
```

```
const app = express();
```

```
const logger = (req, res, next) => {  
  console.log("Request!");  
  next();  
};
```

```
[...routes]
```

```
app.use(logger);
```

```
[...serve]
```

Middleware: bodyParser

```
[...imports]
```

```
const bodyParser = require('body-parser');
```

```
const app = express();
```

```
app.use(bodyParser.json());
```

```
app.use(bodyParser.urlencoded({extended: false}));
```

```
[...routes]
```

```
[...serve]
```

Middleware: Static Path

```
[...imports]
```

```
const app = express();
```

```
app.use(bodyParser.json());
```

```
app.use(bodyParser.urlencoded({extended: false}));
```

```
app.use( express.static( path.join(__dirname, 'public') ) );
```

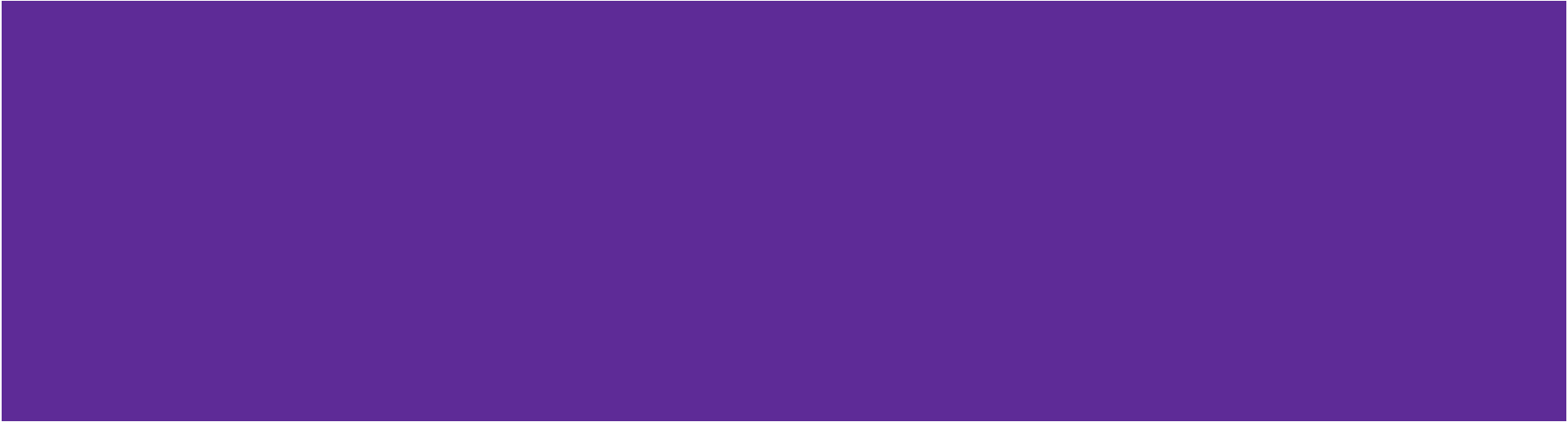
```
[...routes]
```

```
[...serve]
```

"FECHAR E ABRIR O SERVER TODA HORA QUE EU
SALVAR O ARQUIVO É UMA MERDA." - VOCÊ



```
npm install -g nodemon  
$ nodemon
```

A large solid purple rectangle occupies the bottom half of the image, extending from the left edge to the right edge and from the bottom edge up to the terminal text.

JSON

```
[...imports]
const app = express();

let batata = {
  isSweet: false,
  name: 'Fernando'
};

app.get('/', (req, res) => {
  res.json ( batata );
});

[...routes]
[...serve]
```

JSON JSON! JSON! JSON.. JSON?

```
[...imports]
```

```
const app = express();
```

```
let batata = [  
  {isSweet: false, name: 'Fernando'},  
  {isSweet: true, name: 'Tomate'},  
  {isSweet: false, name: 'Larissa'}  
];
```

```
app.get('/', (req, res) => {  
  res.json ( batata );  
});
```

```
[...routes]
```

```
[...serve]
```

Como criar uma batata

```
[...imports]
```

```
let batata =
```

```
  [{ isSweet: false, name: 'Fernando', email: 'fer.batata@horta.io' }];
```

```
[...get]
```

```
app.post('/', (req, res) => {
```

```
  const newBatata = {...req.body};
```

```
  res.json(newBatata);
```

```
});
```

```
[...routes] [...serve]
```

Toda batata tem nome

```
[...imports]
```

```
let batata = [];
```

```
[...get]
```

```
app.post('/', (req, res) => {  
  const newBatata = {...req.body};  
  res.json(newBatata);  
});
```

```
[...routes] [...serve]
```

```
$ yarn add express-validator
```



Toda batata tem nome

```
app.post(
  '/', batataValidator, (req, res) => {
    const errs = validationResult(req);
    if(!errs.isEmpty()) {
      res.status(422)
        .json({
          errors: errs.mapped()
        });
      return;
    }
    const newBatata = {...req.body};
    res.json(newBatata);
  });
```

```
const { check, validationResult } =
  require('express-validator/check');

const batataValidator = [
  check('name')
    .notEmpty()
    .withMessage('Need a name.'),

  check('email')
    .isEmail()
    .withMessage(
      'Must be an email.')
];
```

"NOSSA PAULO. VOCÊ MANJA
MUITO DE PROGRAMAÇÃO EM
ESPAGUETE." - VOCÊ



Estrutura

- ❏ src
 - ❏ models
 - ❏ controllers
 - ❏ public
 - ❏ server.js
 - ❏ server.routes.js
- ❏ index.js

Models

Todos os modelos e schemas vão nesta pasta.

Controllers

Todos os controllers vão nesta pasta.

Public

Pasta com arquivos publicos. Ex: imagens.

server.js

Arquivo para configuração do servidor.

server.routes.js

Arquivo com as rotas para os controllers.

Estrutura

- ❏ src
 - ❏ models
 - ❏ Batata.js
 - ❏ Horta.js
 - ❏ controllers
 - ❏ BatataController.js
 - ❏ HortaController.js
 - ❏ public
 - ❏ server.js
 - ❏ server.routes.js
- ❏ index.js

Batata.js

Schema e model de uma batata.

Horta.js

Schema e model de uma horta.

BatataController.js

Controlador e rotas para controlar a batata.

HortaController.js

Controlador e rotas para controlar a horta.

MongoDB

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

const mongoUri = "mongodb://localhost/hortai0";

mongoose.connect(mongoUri);

mongoose.connection.on('error', () => {

    throw new Error(`unable to connect to database: ${mongoUri}`);

});
```

server.routes.js

```
const express = require('express');
const { batataRoutes } = require('./controllers/batata');

const router = express.Router();

router.get('/', (req, res) => {
  res.send("Hello, World!");
});

router.use('/users', batataRoutes );

module.exports = router;
```

server.js

```
[...imports]  
const cors = require('cors');  
const routes = require('./server.routes');
```

```
ConfigureMongoDB();
```

```
app.use(cors);  
app.use('/api', routes);
```

```
[...serve]
```

BatataController.js

```
const express = require('express');  
const router = express.Router();
```

```
const batataController = {  
  list(req, res) {},  
  get(req, res) {},  
  post(req, res) {},  
  put(req, res) {},  
  delete(req, res) {}  
};
```

[ROUTER]

```
module.exports = {  
  batataController: controller,  
  batataRouter: router  
};
```

BatataController.js

```
const express = require('express');
const router = express.Router();

router.route('/')
  .get(batataController.list.bind(batataController))
  .post(batataController.create.bind(batataController));

router.route('/:userId')
  .get(batataController.get.bind(batataController))
  .put(batataController.update.bind(batataController))
  .delete(batataController.delete.bind(batataController));

module.exports = {
  batataController: controller,
  batataRouter: router
};
```

Models

```
type Batata = {  
  name: { String, Required },  
  isSweet: Boolean = false,  
  email: { String, Required, Unique },  
  horta: Horta  
}
```

```
type Horta = {  
  owner: { String, Required },  
  batatas: [Batata]  
}
```