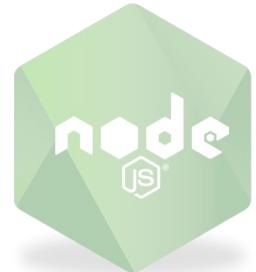


Criar uma conta nos seguintes websites

- TravisCI (Integração contínua)
 - <https://travis-ci.org>
- MLab (Banco de Dados)
 - <https://mlab.com/>
- Heroku (Plataforma de aplicações em nuvem)
 - <http://heroku.com/>
- Instalar "nodemon" globalmente
 - `npm install -g nodemon`



Adicione um usuário para o database no site mlab

The screenshot shows the mLab database interface for the 'nodejs-course' database. The 'Users' tab is selected, displaying a table with one row for the 'admin' user. A modal window titled 'Add new database user' is open, prompting for a database username ('admin'), database password, confirmation of the password, and a 'Make read-only' checkbox. The background shows connection details and a 'mongod' version message.

To connect using the mongo shell:
% mongo ds157762.mlab.com:57762/nodejs-course -u <dbuser> -p <dbpassword>

To connect using a driver via the standard MongoDB URI ([what's this?](#)):
mongodb://<dbuser>:<dbpassword>@ds157762.mlab.com:57762/nodejs-course

mongod version: 3.6.7 (MMAPv1)

WELCOME PLANS

Home Database: nodejs-course

Delete database

Collections Users Stats Backups Tools

Add database user

Database Users

NAME	READ ONLY?
admin	false

Database Users

NAME	READ ONLY?
admin	false

Add new database user

Database username*
admin

Database password*
.....

Confirm password*
.....

Make read-only

CANCEL CREATE

WELCOME PLANS & PRICING

APV1)



APIs testáveis e escaláveis de alta performance com nodejs

Renan Pupin
André Andreotti

Sobre

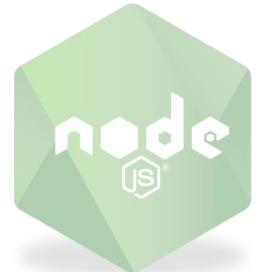


- Renan
 - Bacharel em Ciência da Computação pela FCT-UNESP, 2016
 - Aluno especial do Programa de Mestrado em Ciência da Computação
 - CTO e Fundador do FalaFreud, 2016
- André
 - Bacharel em Ciência da Computação pela FCT-UNESP, 2016
 - Mestrando em Ciência da Computação pelo IBILCE-UNESP, 2017
 - Pesquisador no projeto Radiar na Petrobrás, 2017
 - Desenvolvedor no FalaFreud, 2018



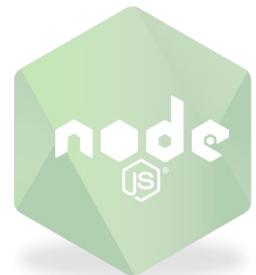
Sumário

- Javascript
- Nodejs
- APIs REST
- MongoDB e No-SQL
- Integração contínua e Cloud Computing
- TDD e Escalabilidade



Sumário

- Hands On
 - Hello World node.js
 - Desenvolvendo uma API orientada a testes
 - Executando sua API para a nuvem e disponibilizando para a internet
 - Integrar a um CI e rodar testes automatizados automaticamente

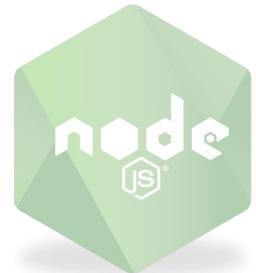


Javascript



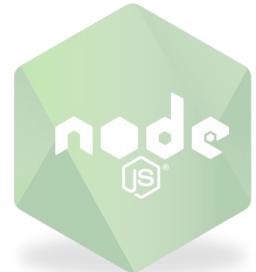
O que é?

- Linguagem de programação
- Brendan Eich, 1995 - Mozilla Corporation
- LiveScript
- Rebatizada para JavaScript
- Orientada a objetos
- Navegadores, Photoshop, GNOME, Node.js



Características

- Várias funcionalidades avançadas
- Pequeno esforço
- APIs disponíveis em navegadores web
- APIs de terceiros (twitter, facebook...)
- Frameworks e bibliotecas



Sintaxe

- Case-Sensitive
- Unicode (107 mil caracteres)
- Instruções separadas por ponto e vírgula (;
- Palavras reservadas (if, else, case, break...)
- (a.if, a['if'], a = {if : 'test'}) são permitidas pois não são identificadores



Tipos, estrutura e JSON

- var (declara uma variável que é acessível independente do escopo de bloco)
- let (declara uma variável local de escopo de bloco)
- const (declara uma constante apenas de leitura)
- Tipos de dados (Boolean, null, undefined, Number, String, Symbol (ES6), Object, Array)
- Dinamicamente tipada

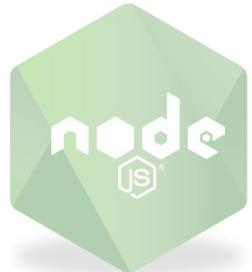
```
1  var a = 7;
2  a = "Isso é uma String.";
```



Tipos, estrutura, JSON

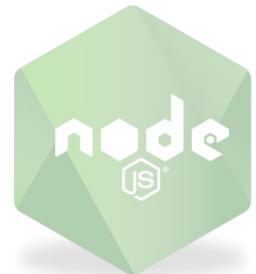
- JSON (JavaScript Object Notation)
- Sintaticamente idêntico ao código para criar objetos JavaScript
- Utilizado para troca de dados do servidor com a página web

```
1  {
2    "employees": [
3      {"firstName":"John", "lastName":"Doe"},
4      {"firstName":"Anna", "lastName":"Smith"},
5      {"firstName":"Peter", "lastName":"Jones"}
6    ]
7  }
```



ES6 - ECMAScript 6

- Ser uma linguagem melhor para construir aplicações complexas
- Resolver problemas antigos do JavaScript
- Facilidade no desenvolvimento de *libraries*



ES6 - ECMAScript 6

- Inclusão de “let” e “const” na declaração de variáveis
- Arrow Functions

(a)

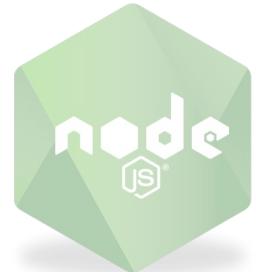
```
1  var sum = function(x, y) {  
2      return x + y;  
3  };
```

(b)

```
1  const sum = (x, y) => {  
2      return x + y  
3  };
```

(c)

```
1  const sum = (x, y) => x + y  
2
```



ES6 - ECMAScript 6

- Utilizando o “this”

```
1 function Widget() {  
2     var button = document.getElementById('button');  
3     button.addEventListener('click', function() {  
4         this.doSomething();  
5     });  
6 }
```

- O 'this' não aponta para Widget como esperado e provocará um erro



ES6 - ECMAScript 6

- Utilizando o “this” com arrow function

```
1  function Widget() {  
2      const button = document.getElementById('button')  
3      button.addEventListener('click', () => {  
4          this.doSomething()  
5      })  
6  }
```

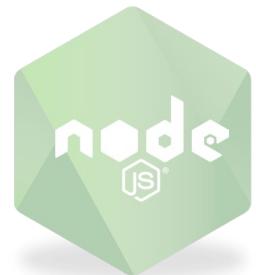
- O 'this' aponta para Widget e não provocará nenhum erro



ES6 - ECMAScript 6

- Default parameters

```
1  const multiply = (x, y = 1) => {
2    return x * y
3  }
4
5  multiply(3, 2) // 6
6  multiply(3) // 3
```



Closures

- add5 e add10 são closures
- makeAdder é uma função fábrica

```
1  function makeAdder(x) {
2    return function(y) {
3      return x + y;
4    };
5  }
6
7  var add5 = makeAdder(5);
8  var add10 = makeAdder(10);
9
10 print(add5(2)); // 7
11 print(add10(2)); // 12
```



Módulos

- Organizar o sistema
- Aumentar o reuso
- Diminuir a complexidade



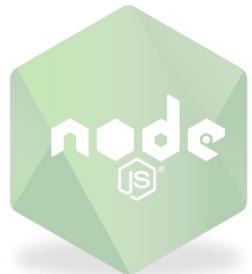
Módulos

- Exportar módulos

```
1 //lib.js
2 exports.sum = function (x, y) {
3     return x + y;
4 }
```

- Importar módulos

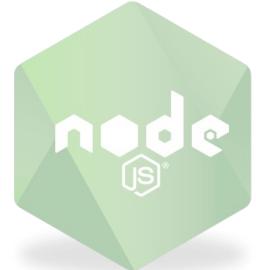
```
1 const math = require('./lib.js')
2
3 math.sum(1, 2) //3
```



Callbacks

- Função passada como argumento para outra função
- Invocada dentro da função externa para completar algum tipo de rotina

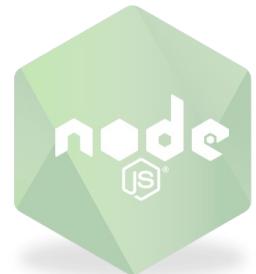
```
1  function greeting(name) {
2    alert('Hello ' + name);
3  }
4
5  function processUserInput(callback) {
6    var name = prompt('Please enter your name.');
7    callback(name);
8  }
9
10 processUserInput(greeting);
```



Promises

- Um objeto que representa a eventual conclusão ou falha de uma operação assíncrona

```
1  Let p = new Promise (function (resolve, reject) {  
2      // resolva ou rejeite  
3  });
```



Promises

Com callback

```
1 doSomething(function(result) {  
2     doSomethingElse(result, function(newResult) {  
3         doThirdThing(newResult, function(finalResult) {  
4             console.log('Got the final result: ' + finalResult);  
5         }, failureCallback);  
6     }, failureCallback);  
7 }, failureCallback);  
8 }
```



Promises

Com promise

```
1  doSomething().then(function(result) {
2    return doSomethingElse(result);
3  })
4  .then(function(newResult) {
5    return doThirdThing(newResult);
6  })
7  .then(function(finalResult) {
8    console.log('Got the final result: ' + finalResult);
9  })
10 .catch(failureCallback);
```



Nodejs

Node.js® is a JavaScript runtime platform built on Chrome's V8 JavaScript engine.



História

- Criado por **Ryan Dahl** em 2009
- **Interpretador de código JavaScript**
- Open Source
 - Atualmente seu desenvolvimento é **mantido** pela **Node.js Foundation** em parceria com a **Linux Foundation**.



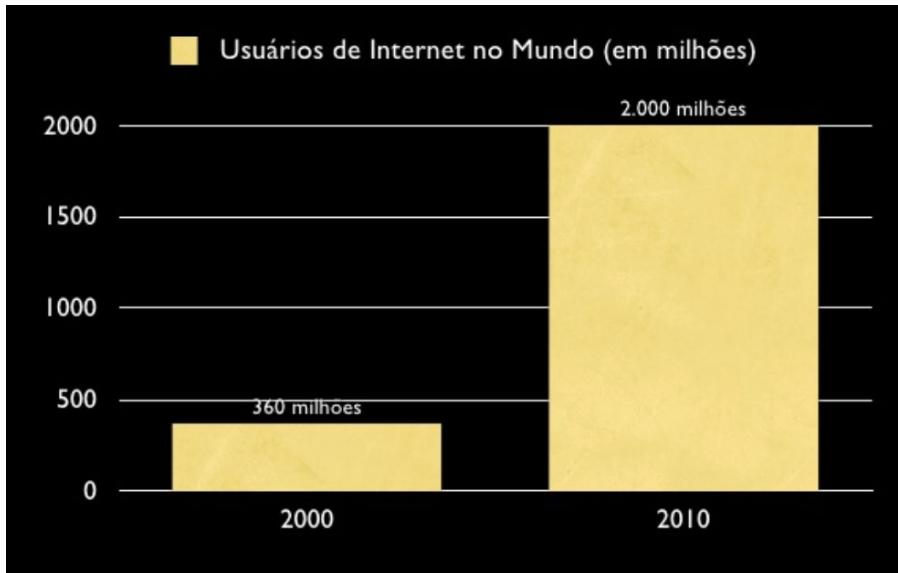
Sobre o nodejs

- Desenvolvimento **Javascript no lado do servidor**
- Plataforma de desenvolvimento de **aplicações** altamente **escaláveis**, com códigos **capazes de manipular** dezenas de **milhares de conexões simultâneas**, numa única máquina física.
- **Multiplataforma**
- **Independente de navegador**

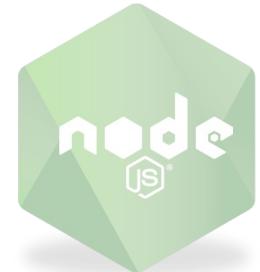


Motivação

- Anos 2000: 360 milhões de pessoas com acesso a internet.
- Em 2010: 2 bilhões



PHP	1995
Java EE	1998
ASP.Net	2002
Ruby on Rails	2005
Django	2005



Motivação

- Código costumava ser escrito assim

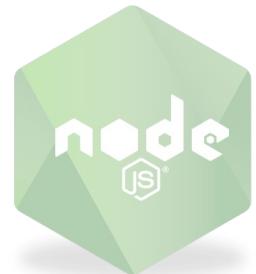
```
findUserByName(name){  
    var busca = "SELECLONE USUARIO ONDE NOME = 'NOME"';  
    var usuario = Usuario.encontre(busca);  
  
    retorno usuario;  
}
```

- O que o software está fazendo enquanto a busca executa?
- Na maioria dos casos está travado esperando a resposta do banco



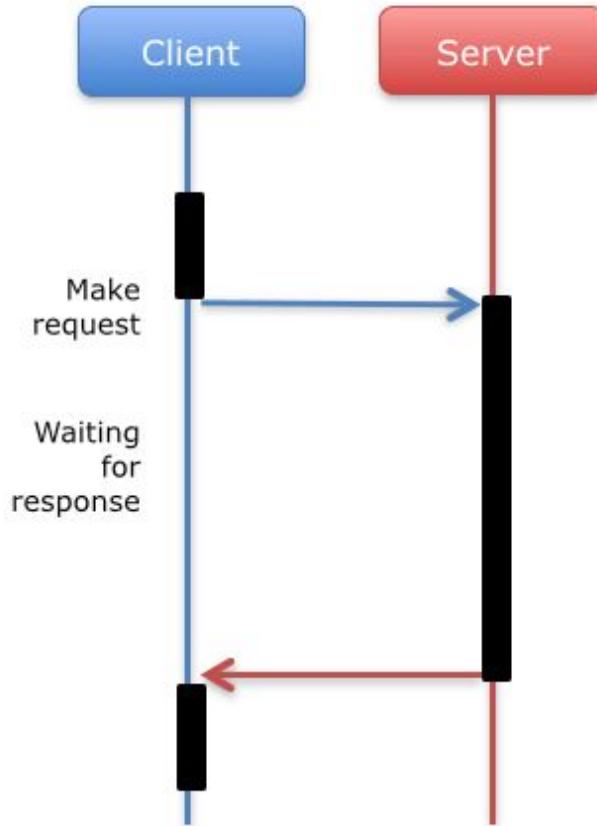
Motivação

- **Servidor web com requisições de I/O bloqueantes**
 - Leitura de disco, query no banco de dados, envio de email, requisição para API de terceiros
- **Uma Thread para cada conexão**
 - Se uma Thread usa 2mb
 - **2mb x 1000 conexões simultâneas = 2GB**
- Como manter conectados **100 mil usuários simultâneos?**
 - $2\text{mb} \times 100.000 \text{ conexões simultâneas} = \text{200GB????}$

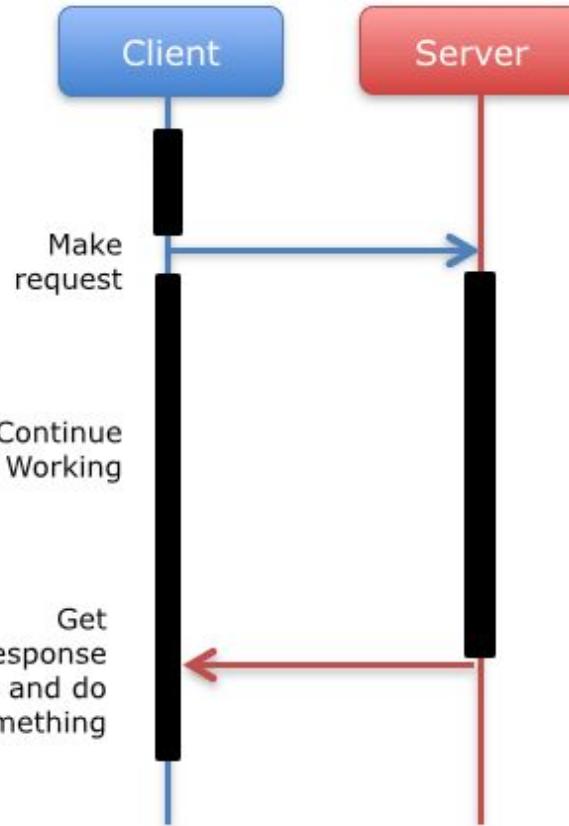




Synchronous

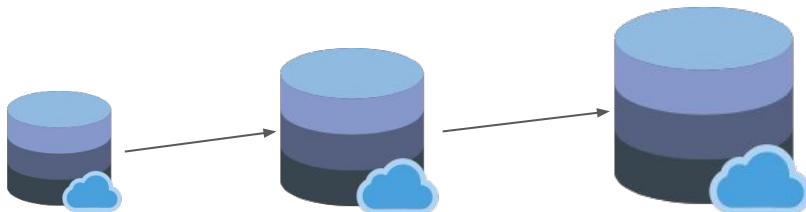


Asynchronous

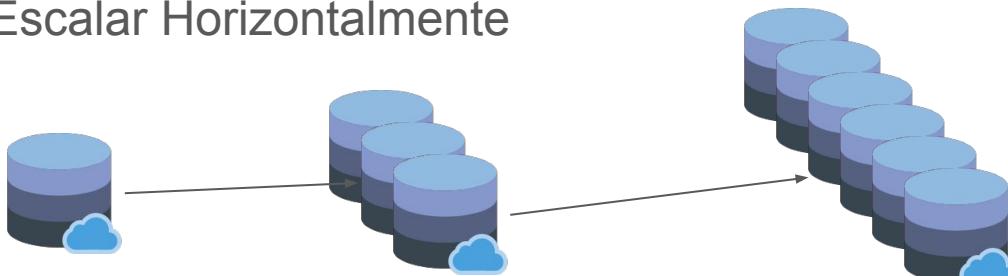


Possíveis soluções

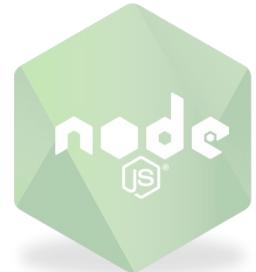
- Escalar Verticalmente



- Escalar Horizontalmente



- Desenvolver uma linguagem de programação mais rápida



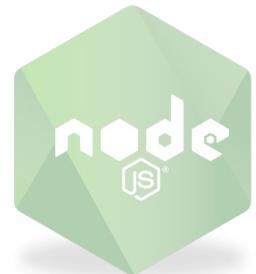
Solução: Novos Paradigmas

- **Programação orientada a evento**
- **Single Thread**
 - **Múltiplas Threads** não funcionam bem com eventos de I/O
 - **Threads escalam** em utilização de **CPU** (Processamento Paralelo) mas comprometem a memória
- **Requisições I/O**
 - **Assíncronas**
 - **Não bloqueantes**



Como funciona?

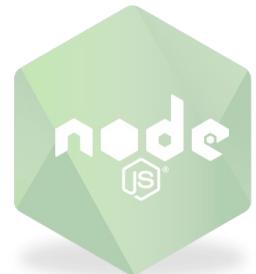
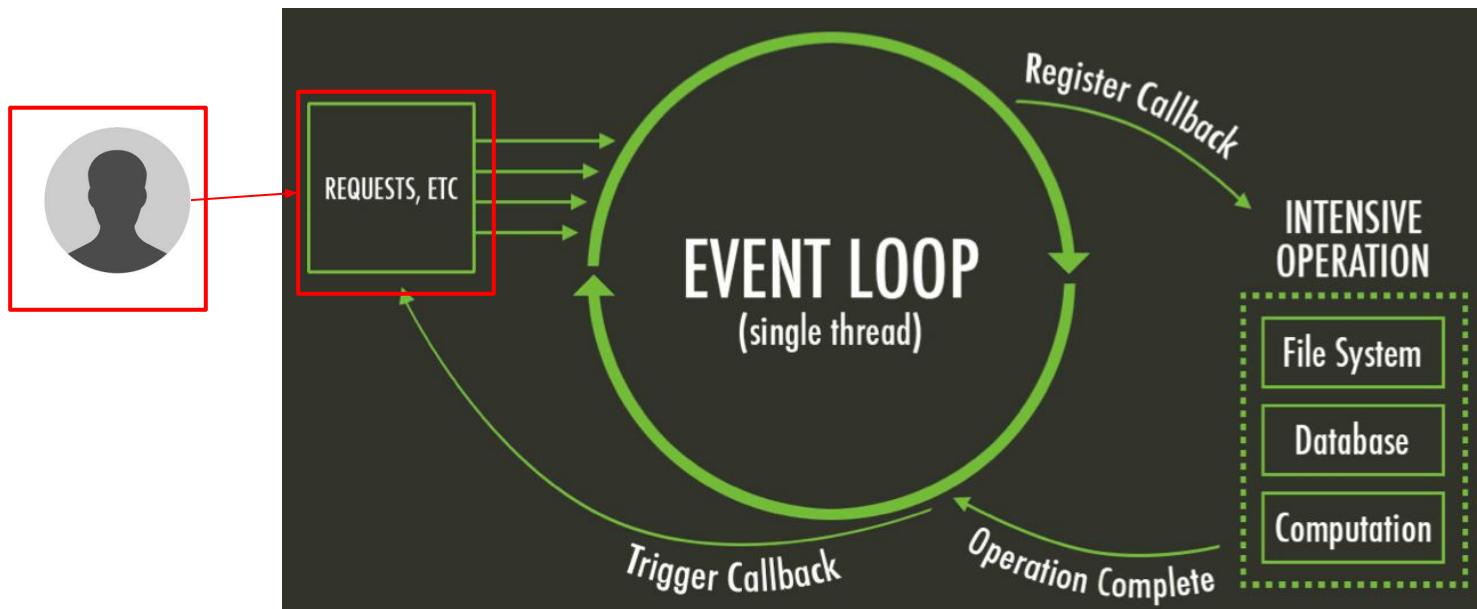
- No Nodejs as **requisições** são processadas **uma por vez**
- O Nodejs **opera em single-thread** puramente **baseado em eventos**
- Usa o **modelo de I/O** com infraestrutura **não bloqueante, eficiente e altamente escalável**



O fluxo

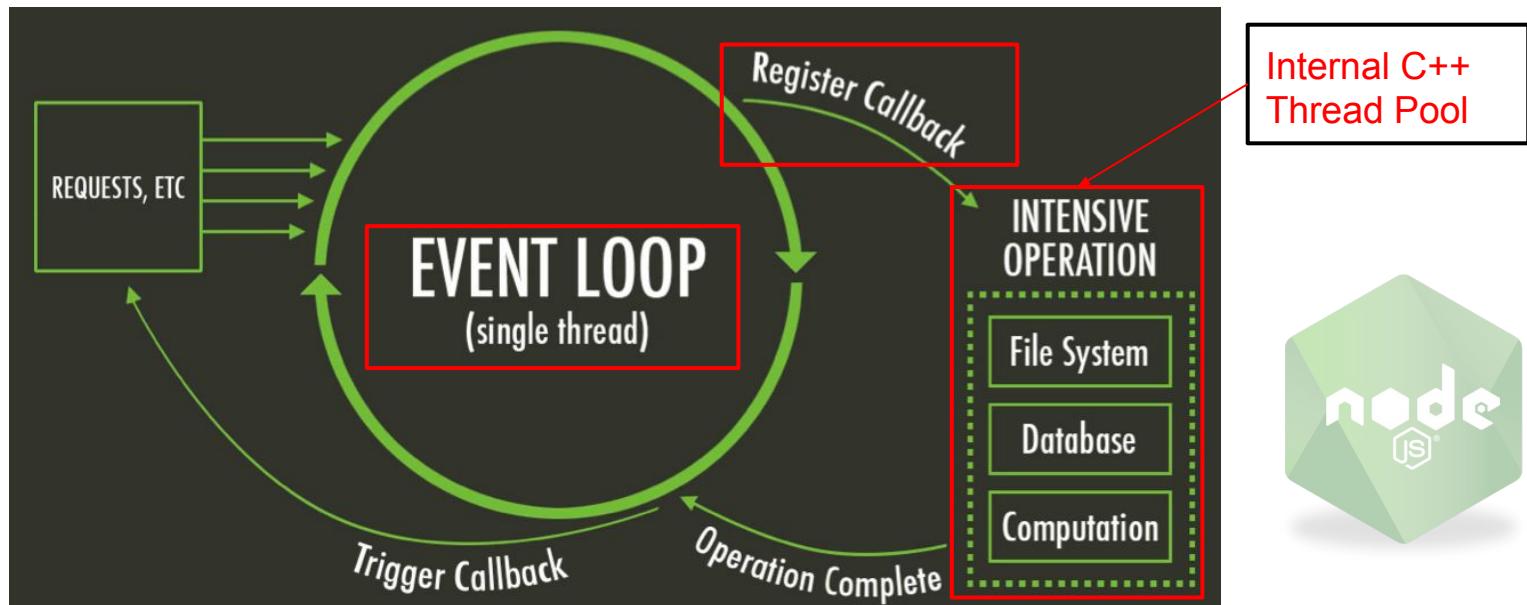
- O usuário envia uma requisição para o servidor nodejs

<https://stackoverflow.com/questions/21596172/what-function-gets-put-into-eventloop-in-nodejs-and-js>



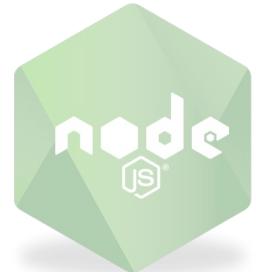
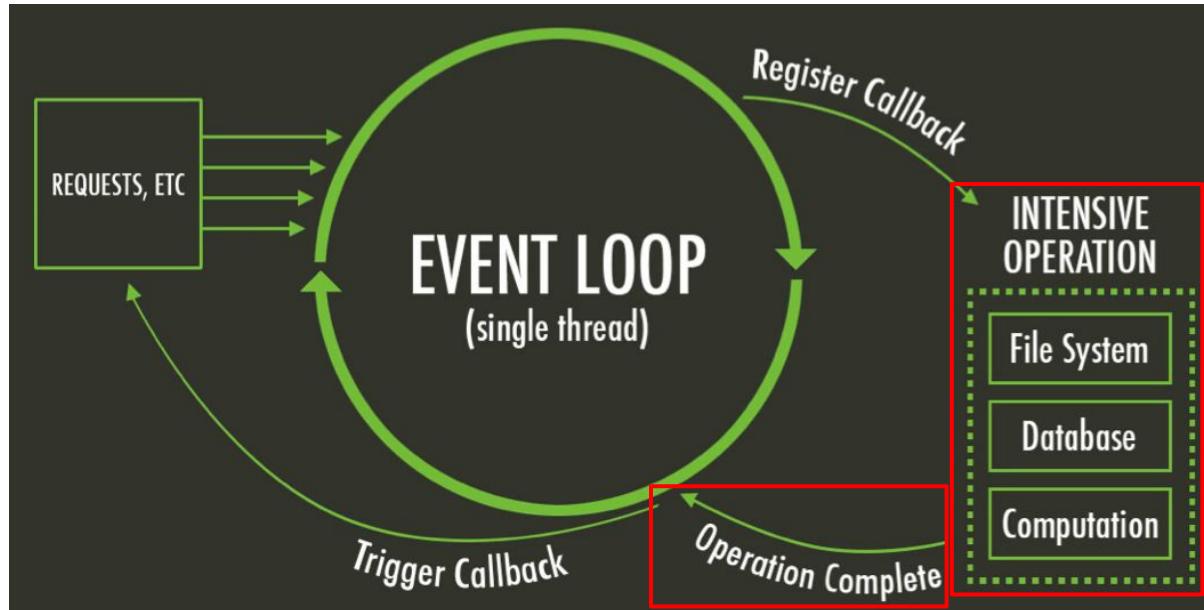
O fluxo

- O **loop de eventos** da thread principal **registra** um **callback**, encaminha o trabalho para um **pool de threads** e imediatamente **retorna** para escutar a **próxima conexão**.



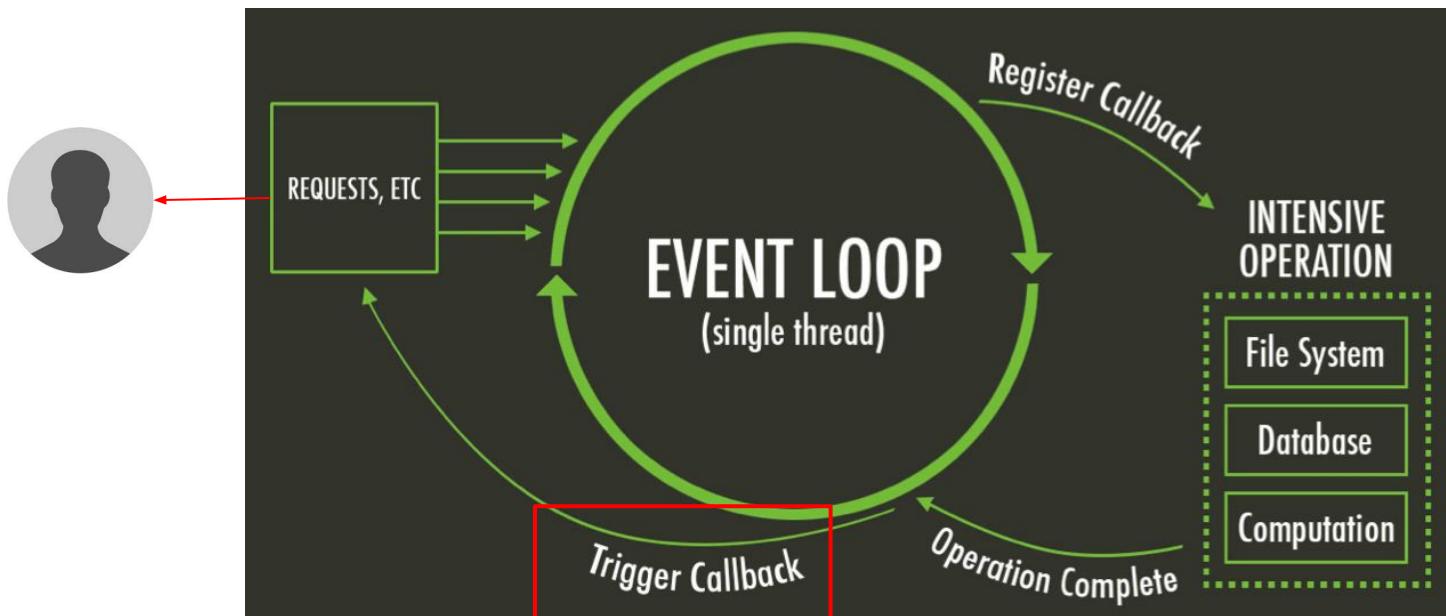
O fluxo

- As threads do pool executam as tarefas de longa duração de forma assíncrona e paralelizada. Ao finalizar uma tarefa, a resposta é enviada para a thread principal via callback.



O fluxo

- O loop de eventos retorna o resultado para o usuário.



Sobre o nodejs

- Porque não fazíamos isso antes?
 - I/O assíncrono não era suportado em todos os SOs
- Porque Javascript?
 - JS já foi estruturado pensando em ser assíncrono

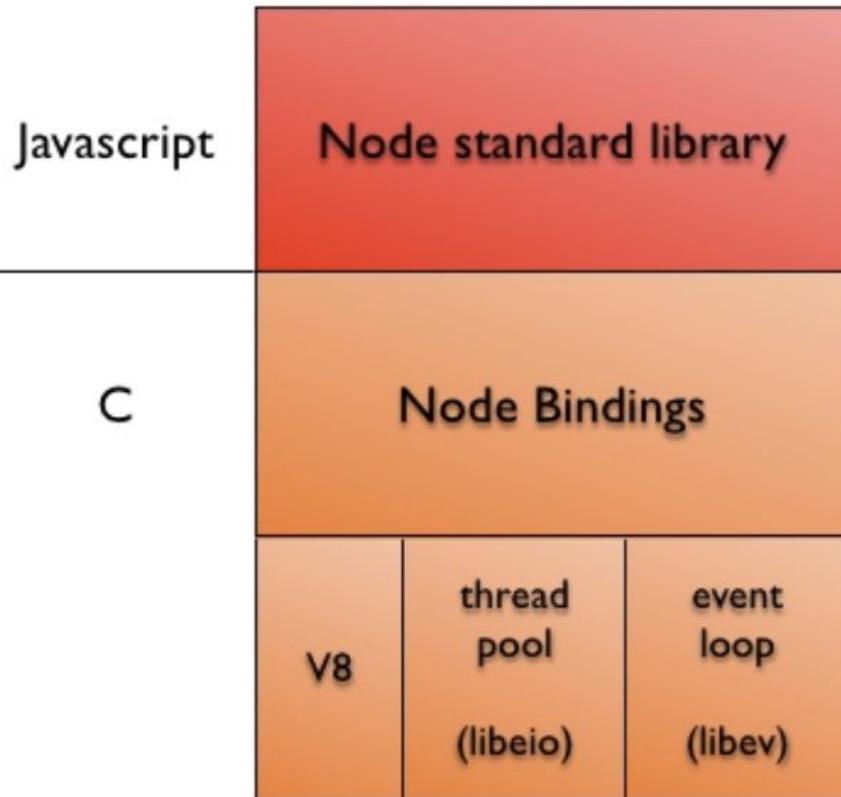


Arquitetura

- Escrito em C/C++ e javascript

- libuv - Cross-platform asynchronous I/O

- <http://libuv.org/>



Sobre o nodejs

- Nesse exemplo de "Hello World"
 - Várias conexões podem ser gerenciadas simultaneamente
 - Para cada conexão um callback é retornado
 - Porém se não tiver trabalho a ser feito, o nodejs vai "dormir"
- <https://nodejs.org/>

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

const hostname = '127.0.0.1';
const port = 3000;

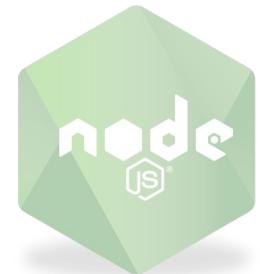
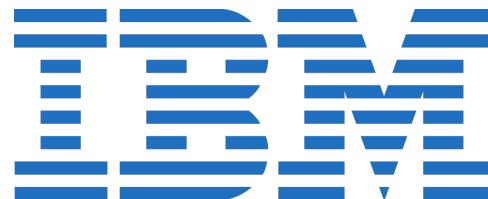
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Quem usa?

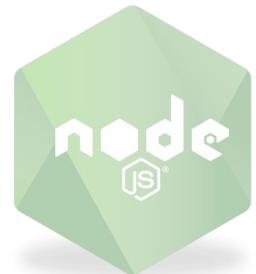


NETFLIX



Vantagens

- Node é totalmente **modular** (na verdade ele é um conjunto de módulos)
- **Simples** (basta abrir o terminal e digitar o comando "node")
- É **otimizado** para aplicações **real time**
- Permite **gerenciar** um **alto nível** de **concorrência** sem **sobrecarregar** o **CPU**
- **Multiprotocolo** (HTTP, DNS, TLS, etc)
- É **possível** criar um **cluster** e trabalhar em **multi-threading**
- Funciona bem com **banco de dados não-relacionais**



Quando não usar node?

- Quando for preciso fazer **codificação de vídeo e processamento intenso**
 - O grande uso de **CPU** anula os **benefícios do I/O não bloqueante**
 - Assim todas as **outras conexões** ficarão **bloqueadas** enquanto a **thread está ocupada com o processamento**
- Quando for preciso **utilizar muitos módulos específicos**
 - Nodejs é recente
- Se você confunde **java** com **#javascript**xD



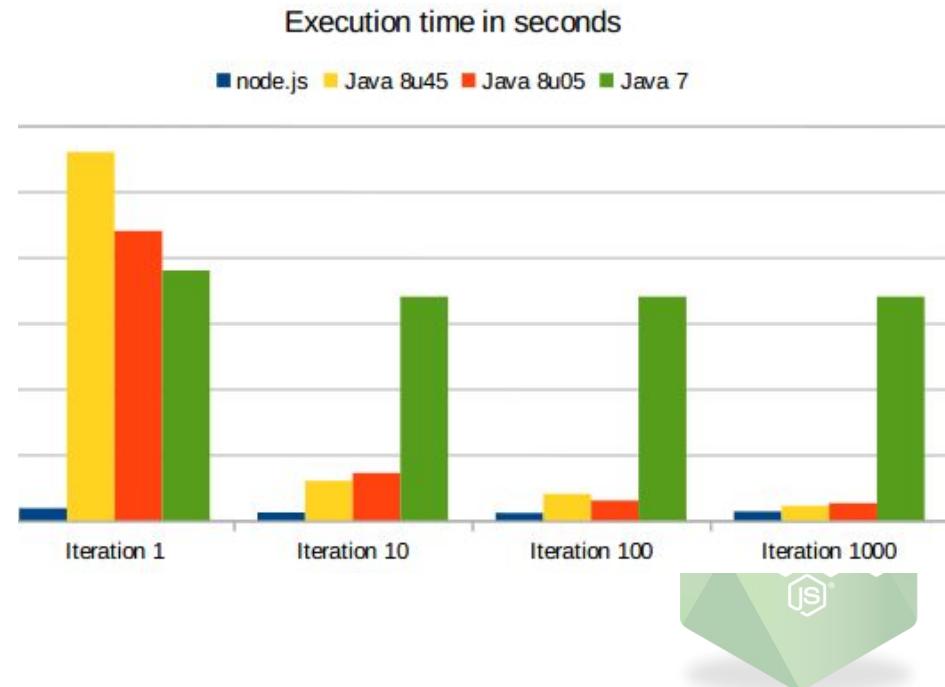
Comparativo



vs



- Permite um **desenvolvimento 70% mais rápido**
 - 40% menos arquivos e 33% menos LOC
- Possibilita **2x mais requisições** com **% do processamento usado pelo Java**
- **Carregamento de página 35% mais rápido**



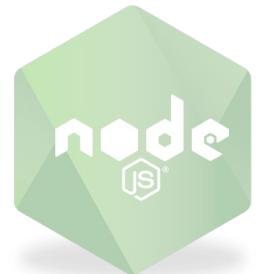
Pacotes/Módulos

- São **dependências** do projeto
 - Ficam no diretório "node_modules" da sua aplicação
- Podem ser
 - Nativos
 - De terceiros
 - Próprios



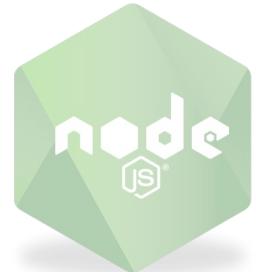
Node Package Manager (NPM)

- **Gerenciador de pacotes** do `nodejs`
- Arquivo `package.json`
 - Define um módulo
 - Informa: nome, versão, endereço do repositório, dependências, scripts
- Comandos
 - Iniciar módulo: `npm init <nome_pacote>`
 - Instalar pacote: `npm install <nome_pacote> --save` (ou `--save-dev`)
 - Remover pacote: `npm uninstall <nome_pacote>`
 - Atualizar pacote: `npm update <nome_pacote>`



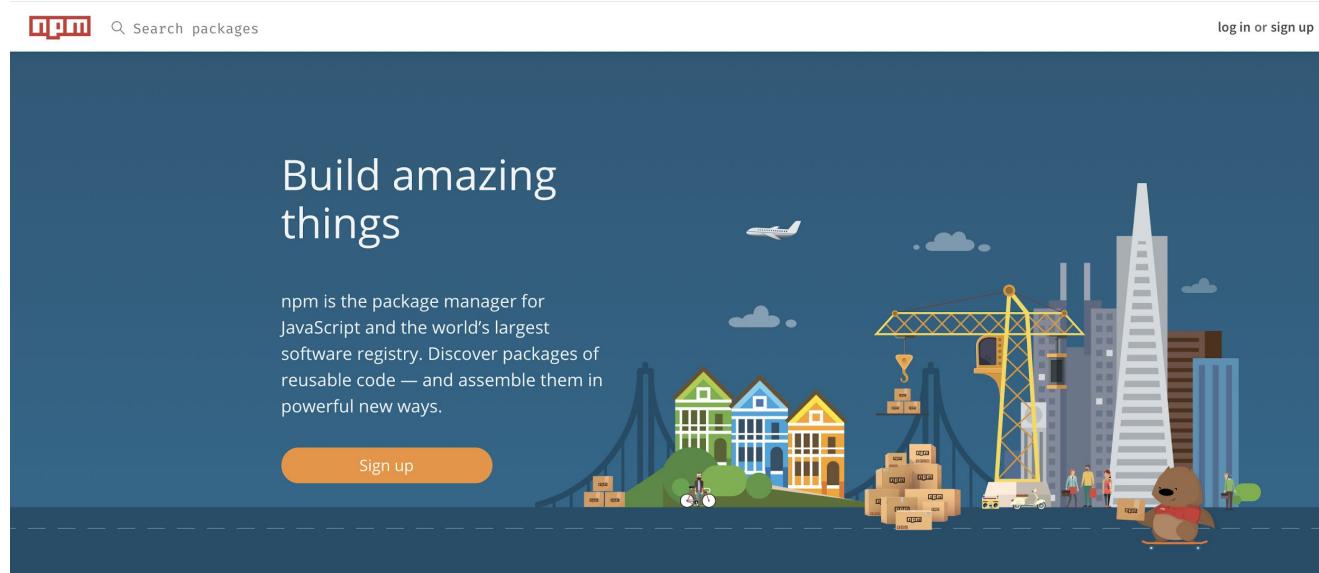
Módulos Nativos

- Escritos em
 - C, C++
 - Javascript
- Exemplo
 - Path
 - Url
 - Http
 - OS
 - Events
- <https://nodejs.org/api>



Módulos de terceiros

- Mais de 700.000 módulos publicados
- <https://www.npmjs.com/>



Módulos Próprios

- Declarando

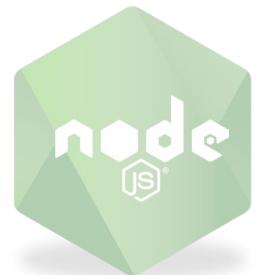
```
//math.js
module.exports = {
  soma: function (a, b) { return a + b; }
}
```

- Consumindo

```
//outro.js
var math = require('./math');
math.soma(1, 2); //3
```

```
//package.json

{
  "name": "math",
  "version": "1.0.0",
  "description": "",
  "main": "math.js",
}
```



Hello World

- Criar arquivo chamado "server.js"

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

const hostname = '127.0.0.1';
const port = 3000;

let server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log("Server running at http://"+hostname+":"+port);
});
```

- Rodar o comando "node server.js" no terminal



Hello World

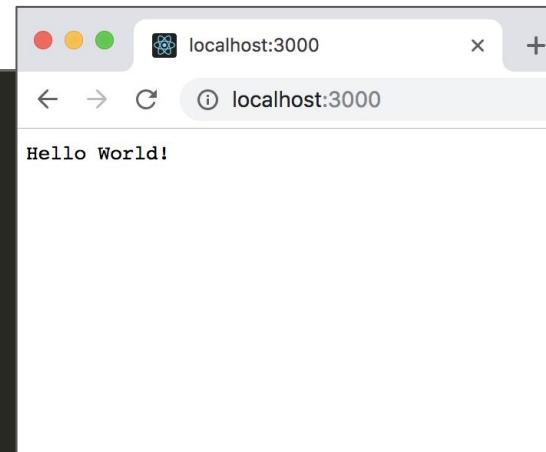
- Criar arquivo chamado "server.js"

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

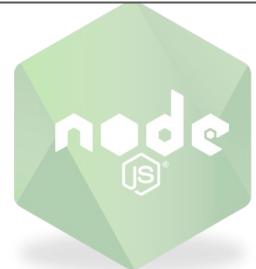
const hostname = '127.0.0.1';
const port = 3000;

let server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log("Server running at http://"+hostname+":"+port);
});
```



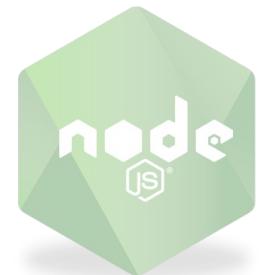
- Rodar o comando "node server.js" no terminal



Express

express

- O Express é um **framework** que **facilita criar aplicações web** utilizando **nodejs**.
- É um framework **rápido, minimalista e flexível** que fornece um **conjunto** robusto de **recursos** para **criar APIs** de maneira fácil.
- Escrito em JavaScript que **roda sobre o ambiente node.js** em tempo de execução.
- <https://expressjs.com/pt-br/>



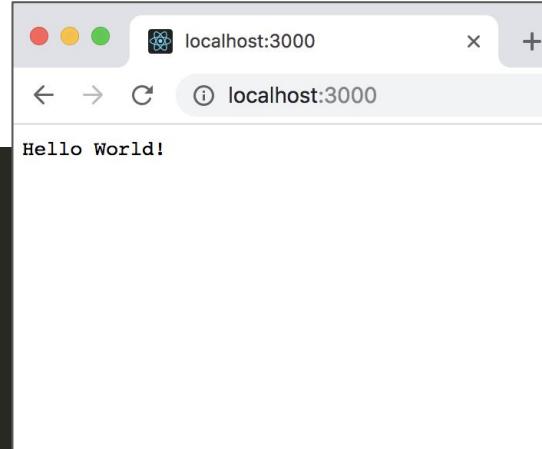
Express: Hello World

- Criar arquivo chamado "server_express.js"

```
var express = require('express');
var app = express();

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

app.listen(5000, function () {
  console.log('Example app listening on port 5000!');
});
```



- Rodar o comando "npm install express --save"
- Rodar o comando "node server_express.js" no terminal



APIs REST

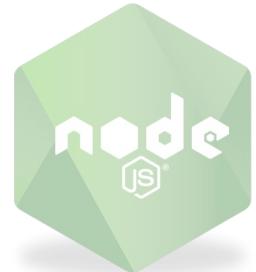
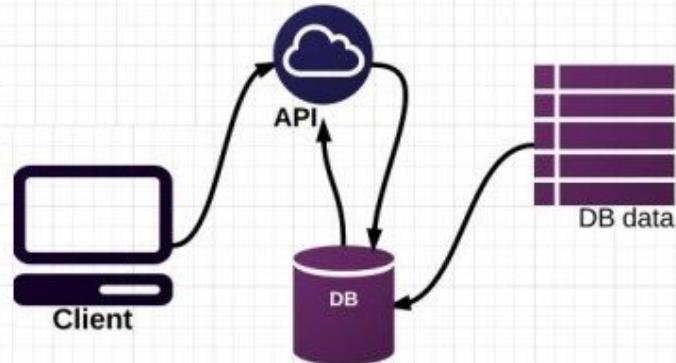
- APIs (Interface de Programação de Aplicações)
 - É um **conjunto de rotinas e padrões** estabelecidos por um **software** para a **utilização** das suas **funcionalidades** por **aplicativos** que não pretendem envolver-se em **detalhes** da **implementação** do **software**, mas apenas **usar seus serviços**.
- REST (Transferência de Estado Representacional)
 - É um **estilo de arquitetura** que **define** um conjunto **propriedades baseados em HTTP**.
 - **Padroniza URLs, e metadados da requisição**
 - **Diferencia as requisições** pelos verbos **HTTP**



APIs REST

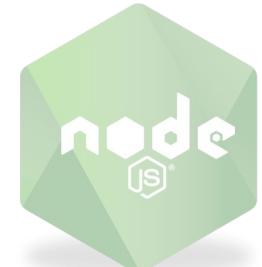
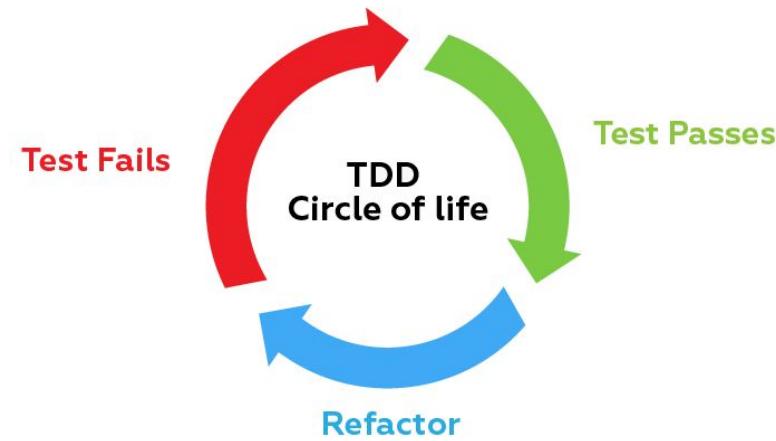
GET	/movies	Get list of movies
GET	/movies/:id	Find a movie by its ID
POST	/movies	Create a new movie
PUT	/movies	Update an existing movie
DELETE	/movies	Delete an existing movie

Web Services



Test Driven Development (TDD)

- É uma **técnica de desenvolvimento de software** que se relaciona com o **conceito de validação**.
- Primeiramente o **desenvolvedor** escreve um **caso de teste automatizado** para uma **nova funcionalidade**.
- Então, **produz o código que será validado pelo teste**, para posteriormente o **código ser refatorado**.
- Kent Beck, considerado o criador do TDD, declarou que o uso do **TDD encoraja o código simples e inspira confiança**.





Cloud Computing: Heroku

- Plataforma de **computação em nuvem** (PaaS), isso significa que você pode fazer **deploy** sem se preocupar com **configurações de hardware e SO**.
- Você só precisa se **focar na aplicação** e nos **componentes de infraestrutura** necessários (Banco de dados, cache, log, filas)
- Ele **suporta linguagens** por meio dos "**Buildpacks**"
 - Ruby, Python, Node, Php, Go, Java
- Os **aplicativos devem incluir** um arquivo "**Procfile**" que **especifica** aos **Dynos**, os **comandos** que deverão ser **executados**.

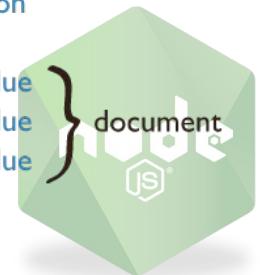


Mongodb



- É um software de **banco de dados orientado a documentos**
- De **código aberto, multiplataforma** e de **alta performance**
- Classificado como um programa de banco de dados **NoSQL**
- Usa **documentos semelhantes a JSON** com esquemas.
- Escrito na linguagem C++
 - O que o torna **portável para diferentes SOs**

```
db.users.insertOne(  
  {  
    name: "sue",      ← field: value  
    age: 26,          ← field: value  
    status: "pending" ← field: value  
  })
```



Mongoose

mongoose

- "Modelagem de objetos mongodb elegante para node.js"
- O mongoose foi criado para:
 - Facilitar as validações do MongoDB no nodejs
 - Fornecer uma solução direta baseada em esquemas para modelar os dados da aplicação
 - Construção de queries de maneira simples



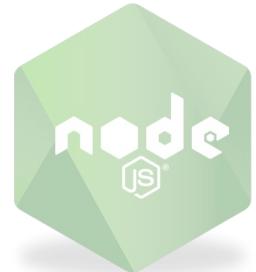
Mongoose Schema

- Tudo no **Mongoose** começa com um **esquema**.
- São **análogos** ao "Modelo" da arquitetura **MVC**.
- O **esquema mapeia** para **uma coleção** do **MongoDB** e **define a forma** que os **documentos** terão dentro **dessa coleção**.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');

let Schema = mongoose.Schema;

let AnimalSchema = new Schema({
  name: String
});
```



Operações com Mongoose

- **Operações**

- find
- findOne
- findOneAndUpdate
- count
- remove
- save

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');

let Schema = mongoose.Schema;

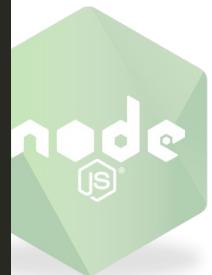
let AnimalSchema = new Schema({
  name: String
});

//create
let dog = new AnimalSchema({
  name: "Cachorro"
});

dog.save();

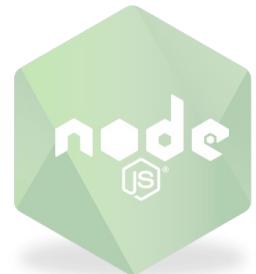
// find

let search = AnimalSchema.find({name: "Cachorro"});
console.log(search); //prints Cachorro
```



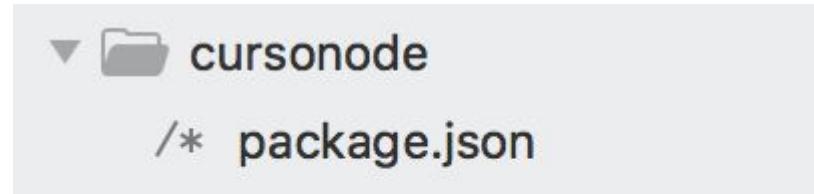
Hands on: Workflow

- Criar novo projeto
- Criar o arquivo de entrada da aplicação
- Instalar dependências
- Criar o primeiro modelo
- Implementar rotas REST e testes automatizados para o CRUD
 - C - Create
 - R - Read
 - U - Update
 - D - Delete
- Rodar aplicação na nuvem
- Rodar testes automatizados na nuvem



Hands on: Criar um novo projeto

- Criar diretório "cursonode"
 - **mkdir cursonode**
- Navegar até a pasta
 - **cd cursonode**
- Inicializar o módulo npm
 - **npm init**
 - No "entry point" digitar "**server.js**"



```
{  
  "name": "cursonode",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\"Error: no test specified\\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC"  
}
```

Hands on: Criar o arquivo de entrada da aplicação

- Abra o projeto no editor de código
- Crie o arquivo "server.js" na raiz do projeto

```
▼ curonode
  /* package.json
  /* server.js
```



Hands on: Instalando as dependências

- Vamos utilizar as seguintes dependências

- Express, Body Parser, Mongoose
 - Mocha, Chai e Chai Http

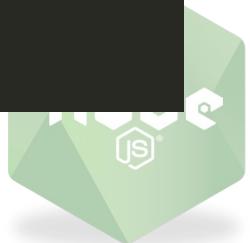
- Para adicionar ...

- npm install <nome_modulo> --save
 - Exemplo: npm install express --save

- Ou...

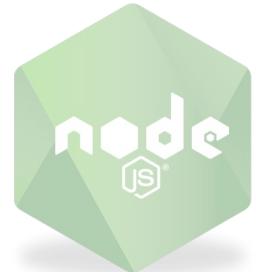
- **npm install express body-parser mongoose mocha chai chai-http --save**

```
{  
  "name": "cursonode",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\"$Error: no test specified\\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "body-parser": "^1.18.3",  
    "chai": "^4.1.2",  
    "chai-http": "^4.2.0",  
    "express": "^4.16.3",  
    "mocha": "^5.2.0",  
    "mongoose": "^5.2.17"  
  }  
}
```



Hands on: Criar API Rest

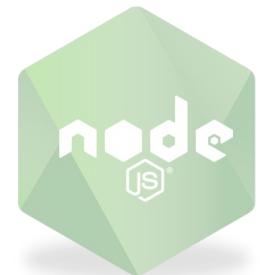
- **API REST para lista de tarefas**
 - **GET** "/todo/:id?" busca um objeto
 - **PUT** "/todo/:id" atualiza um objeto
 - **POST** "/todo" cria um novo objeto
 - **DELETE** "/todo/:id" delete um objeto



Hands on: Implementar rotas REST para o CRUD

- Importar dependências, implementar a rota default e executar o server

```
1  let express = require('express');
2  let app = express();
3
4  let bodyParser = require("body-parser");
5  app.use(bodyParser.json());
6  app.use(bodyParser.urlencoded({ extended: false }));
7
8  app.get('/', function (req, res) {
9    res.send('Hello World!'); //envia texto
10 });
11
12 //SERVER listening
13 let port = process.env.PORT || 5000;
14 app.listen(port, function () {
15   console.log('Example app listening on port ' + port);
16 });
17
18 //add exports to app at server.js
19 module.exports = app;
```



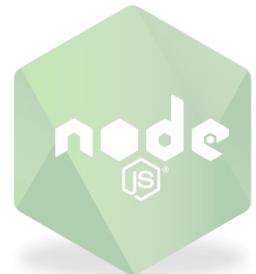
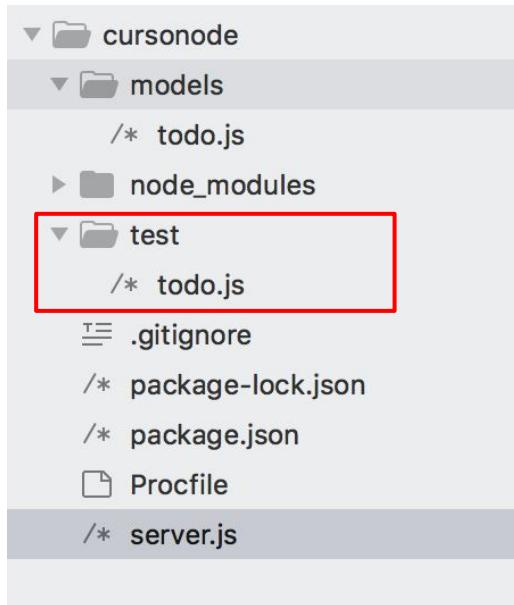
Hands on: Adicionando testes automatizados

- Vamos utilizar Mocha e Chai para executar os testes automatizados
- Vamos adicionar o mocha dentro do script de teste no package.json

```
1  {
2    "name": "cursonode",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "mocha --exit"
8    },
9    "author": "",
10   "license": "ISC",
11   "dependencies": {
12     "body-parser": "^1.18.3",
13     "chai": "^4.1.2",
14     "chai-http": "^4.2.0",
15     "express": "^4.16.3",
16     "mocha": "^5.2.0",
17     "mongoose": "^5.2.17"
18   }
19 }
```

Hands on: Adicionando testes automatizados

- Vamos criar o diretório "test" dentro da pasta raiz do projeto
- Vamos criar o arquivo "todo.js" dentro da pasta "test"



Hands on: Adicionando testes automatizados

- Vamos inicializar nosso arquivo "todo.js" com as seguintes configurações e rodar o comando:
 - npm test

```
→ cursonode git:(master) ✘ npm test
```

```
> cursonode@1.0.0 test /Users/renanpupin/workspace/cursonode
> mocha --exit
```

```
Example app listening on port 5000
Test server working
✓ should get a server message
```

```
1 passing (27ms)
```

```
let chai = require('chai');
let chaiHttp = require('chai-http');
chai.use(chaiHttp);
let expect = chai.expect;
let should = chai.should();

//carregando nossa API
let server = require('../server.js');

describe('Test server working', function(done) {
  it('should get a server message', function(done) {
    chai.request(server)
      .get('/')
      .end(function(err, res){
        expect(res.status).to.eql(200);
        done();
      });
  });
});
```

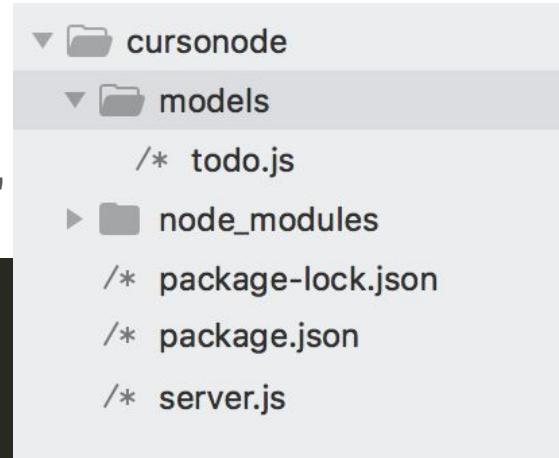
Hands on: Criando o primeiro modelo

- Vamos criar o modelo para a entidade "ToDo"
- Crie o diretório "models" dentro do projeto
- Dentro do diretório criado, crie o arquivo "todo.js"

```
let mongoose = require('mongoose');
let Schema = mongoose.Schema;

let schema = new Schema({
  title: {type: String, required: true},
  is_complete: {type: Boolean, default: false},
  created_at: {type: Date, required: true, default: Date.now},
  completed_at: {type: Date}
});

module.exports = mongoose.model('ToDo', schema);
```



Hands on: Implementar rotas REST para o CRUD

- Conectar no banco de dados (usar URL do banco de cada um) e importar modelo "ToDo"

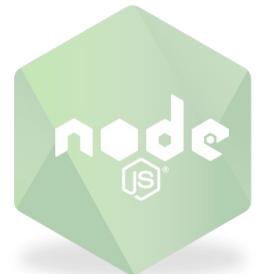
```
6   app.use(bodyParser.urlencoded({ extended: false }));
7
8   let mongoose = require('mongoose');
9   mongoose.connect(
10     "<SUA_URL_DO_MLAB>",
11     { useNewUrlParser: true }
12 );
13
14  let ToDo = require("./models/todo");
15
16  app.get('/', function (req, res) {
```



Hands on: Adicionando testes automatizados

- Vamos criar os testes para as operações da nossa API REST
- GET

```
describe('Test ToDo is Working', function(done) {  
  
    let id;  
  
    //test GET  
    it('should get ToDos', function(done) {  
        chai.request(server)  
            .get('/todo')  
            .end(function(err, res){  
                expect(res.body.success).to.eql(true);  
                expect(res.body.todos).to.be.an('array');  
                done();  
            });  
    });  
});
```



Hands on: Implementar rotas REST para o CRUD

- Implementando o GET /todo

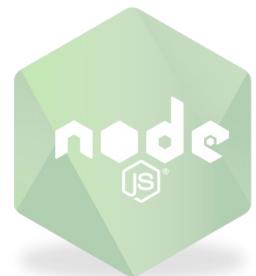
```
//get with callback
app.get('/todo', function (req, res) {
  ToDo
    .find()
    .exec((err, todos) => {
      if(!err){
        res.json({success: true, message: "ToDos buscados com sucesso.", todos }); //responde com um objeto json
      }else{
        res.json({success: false, message: err.message, todos: [] });
      }
    })
});
```



Hands on: Adicionando testes automatizados

- POST

```
describe('Test ToDo is Working', function(done) {  
  let id;  
  
  //test GET  
  it('should get ToDos', function(done) {  
    chai.request(server)  
      .get('/todo')  
      .end(function(err, res){  
        expect(res.body.success).to.eql(true);  
        expect(res.body.todos).to.be.an('array');  
        done();  
      });  
  });  
  
  //test POST  
  it('should create ToDos', function(done) {  
    chai.request(server)  
      .post('/todo')  
      .send({ title: "Todo Teste" })  
      .end(function(err, res){  
        expect(res.body.success).to.eql(true);  
        expect(res.body.todo).to.not.be.undefined;  
        expect(res.body.todo.is_complete).to.eql(false);  
  
        id = res.body.todo._id; //salvando para alterar  
  
        done();  
      });  
  });  
});
```



Hands on: Implementar rotas REST para o CRUD

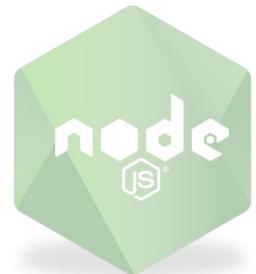
- Implementando o POST /todo

```
//post com async/await e try/catch
app.post('/todo', async(req, res) => {
  try{
    let title = req.body.title;

    let newTodo = new ToDo({
      title: title
    });

    let savedTodo = await newTodo.save();

    res.json({ success: true, message: "Successo!!!", todo: savedTodo });
  }catch(err){
    res.json({ success: false, message: err.message });
  }
});
```

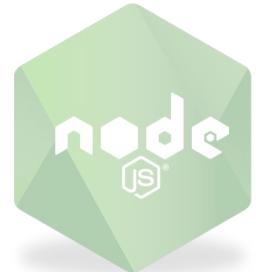


Hands on: Adicionando testes automatizados

- PUT

```
//test PUT
it('should update Todos', function(done) {
  chai.request(server)
    .put('/todo/' + id)
    .send({ is_complete: true })
    .end(function(err, res){
      expect(res.body.success).to.eql(true);
      expect(res.body.todo).to.not.be.undefined;
      expect(res.body.todo.is_complete).to.eql(true);

      done();
    });
});
```



Hands on: Implementar rotas REST para o CRUD

- Implementando o PUT /todo:id

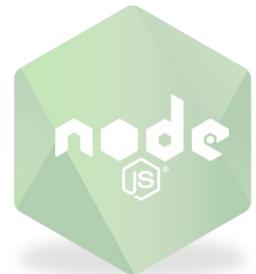
```
//put
app.put('/todo/:id', async(req, res) => {
  try{
    let id = req.params.id;
    let is_complete = req.body.is_complete;

    let todo = await ToDo.findOne({_id: id});
    console.log(todo);

    if(is_complete !== undefined){
      todo.is_complete = is_complete;
      todo.completed_at = new Date();
    }

    let updatedTodo = await todo.save();
    console.log("updatedTodo", updatedTodo);

    res.json({ success: true, message: "Successo!!!", todo: updatedTodo });
  }catch(err){
    res.json({ success: false, message: err.message });
  }
});
```

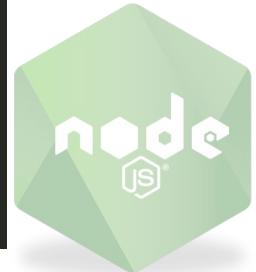


Hands on: Adicionando testes automatizados

- DELETE

```
//test DELETE
it('should remove Todos', function(done) {
  chai.request(server)
    .delete('/todo/' + id)
    .end(function(err, res){
      expect(res.body.success).to.eql(true);

      done();
    });
});
```



Hands on: Implementar rotas REST para o CRUD

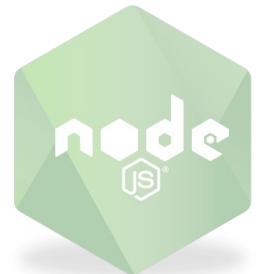
- Implementando o DELETE /todo:id

```
//delete
app.delete('/todo/:id', async(req, res) => {
  try{
    let id = req.params.id;

    let todo = await ToDo.findOne({_id: id});
    console.log(todo);

    await todo.remove();

    res.json({ success: true, message: "Successo!!!" });
  }catch(err){
    res.json({ success: false, message: err.message });
  }
});
```



Hands on: Adicionando testes automatizados

```
5 passing (2s)  
→ cursonode git:(master) x |
```



Hands on: Subindo o código para o Git

- Criar Procfile na raiz do projeto

```
//Procfile  
web: node server.js
```

- Criar arquivo ".gitignore" na raiz do projeto e adicionar "node_modules"

```
/.gitignore  
node_modules
```



Hands on: Subindo o código para o Git

- Inicializar um novo repositório Git

```
➔ cursonode2 git init  
Initialized empty Git repository in /Users/renanpupin/workspace/cursonode2/.git/
```

- Criar repositório chamado "cursonode" no Github

Create a new repository

A repository contains all the files for your project, including the revision history.

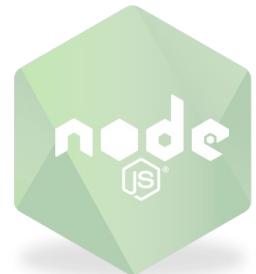
Owner	Repository name
 renanpupin ▾	/ cursonode ✓

Great repository names are descriptive and unique. Your new repository will be created as **cursonode-curso-improved-memory**.



Hands on: Subindo o código para o Git

- Setar a origem do repositório para o Github
 - `git remote add origin <meu_repositorio.git>`
- Subir o código para o git
 - `git add .`
 - `git commit -m "primeiro commit"`
 - `git push origin master`



Hands on: Subindo o código para o Git

renanpupin / cursonode

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit

Manage topics

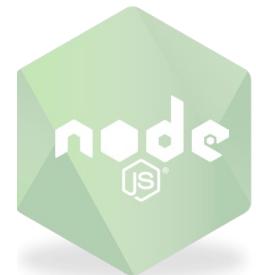
1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

renanpupin primeiro commit Latest commit c84c021 10 seconds ago

File	Commit Message	Time
models	primeiro commit	10 seconds ago
test	primeiro commit	10 seconds ago
.gitignore	primeiro commit	10 seconds ago
Procfile	primeiro commit	10 seconds ago
package-lock.json	primeiro commit	10 seconds ago
package.json	primeiro commit	10 seconds ago
server.js	primeiro commit	10 seconds ago

Help people interested in this repository understand your project by adding a README. Add a README



Hands on: Rodando na nuvem

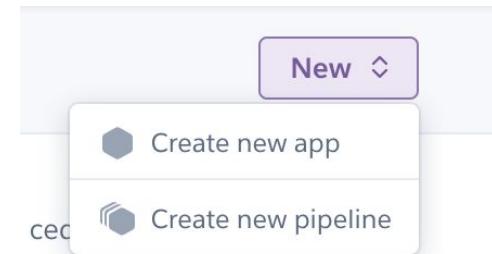
- Após fazer cadastro na plataforma Heroku, abra o link:
 - <https://dashboard.heroku.com/apps>
- Crie um novo app clicando botão do canto direito da tela



A screenshot of the "Create New App" form. The "App name" field contains "cursonode-renan", which is highlighted with a red rectangular border. Below the input field, the text "cursonode-renan is available" is displayed in green. The "App owner" field shows "Renan Oliveira (renan_athas@hotmail.com)". The "Choose a region" dropdown is set to "United States".

Add to pipeline...

Create app



Hands on: Rodando na nuvem

- Vamos conectar o Heroku ao Github

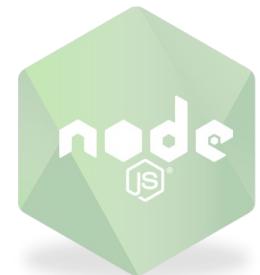
The screenshot shows the Heroku Dashboard for the app 'curonode-renan'. The top navigation bar includes 'Personal' (with a profile icon), the app name 'curonode-renan', a star icon, 'Open app', and a 'More' dropdown. Below the navigation are tabs: Overview, Resources, Deploy (which is selected), Metrics, Activity, Access, and Settings.

The main content area has two sections:

- Add this app to a pipeline**: A section for creating or selecting a pipeline. It contains a note: "Create a new pipeline or choose an existing one and add this app to a stage in it." Below this is a "Choose a pipeline" dropdown menu.
- Add this app to a stage in a pipeline to enable additional features**: A section for connecting to GitHub. It explains that pipelines let you connect multiple apps and promote code between them. It also notes that GitHub-connected pipelines enable review apps and creation of new pull requests. A callout box highlights the "GitHub" button.

Deployment method: A section for connecting to GitHub. It includes a "Heroku Git" icon and a "GitHub" icon with the text "Connect to GitHub". The "GitHub" button is highlighted with a red box.

Connect to GitHub: A section for connecting the app to GitHub. It says: "Connect this app to GitHub to enable code diffs and deploys." Below this is a search bar labeled "Search for a repository to connect to" with the text "renanpupin" and "curonode" entered. A red box surrounds this search area. Below the search bar is a message: "Missing a GitHub organization? Ensure Heroku Dashboard has team access." At the bottom is a "Connect" button.



Hands on: Rodando na nuvem

- Após conectar vamos habilitar os deploys automáticos

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to  [renanpupin/cursonode](#) by  [renanpupin](#)

 [Releases](#) in the [activity feed](#) link to GitHub to view commit diffs

[Disconnect...](#)

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more](#).

Choose a branch to automatically deploy

 master



Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

[Enable Automatic Deploys](#)

Hands on: Rodando na nuvem

- Vamos entrar na aba settings

The screenshot shows a user interface for managing a cloud resource. At the top left is a purple profile icon. Next to it are the words "Personal" and a right-pointing arrow. To the right of the arrow is a purple hexagonal icon with a white "C" inside, followed by the text "curonode-renan". Below this, there is a "GITHUB" link with a purple octocat icon, followed by the repository name "renanpupin/curonode" and a "master" branch indicator. A horizontal navigation bar below these items contains the following tabs: "Overview", "Resources", "Deploy", "Metrics", "Activity", "Access", and "Settings". The "Settings" tab is highlighted with a thick red rectangular border.



Hands on: Rodando na nuvem

- Adicionar o buildpack do nodejs

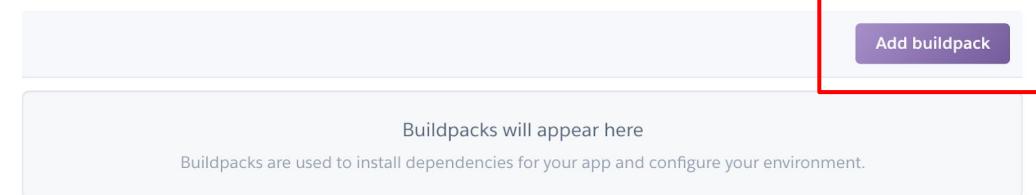
Buildpacks

Buildpacks are scripts that are run when your app is deployed. They are used to install dependencies for your app and configure your environment. [Find new buildpacks on Heroku Elements](#)

Add buildpack

Buildpacks will appear here

Buildpacks are used to install dependencies for your app and configure your environment.



Add Buildpack

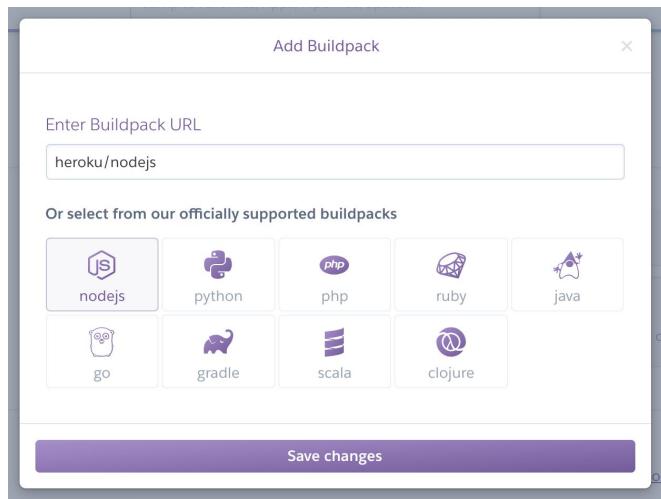
Enter Buildpack URL

heroku/nodejs

Or select from our officially supported buildpacks

nodejs	python	php	ruby	java
go	gradle	scala	clojure	

Save changes

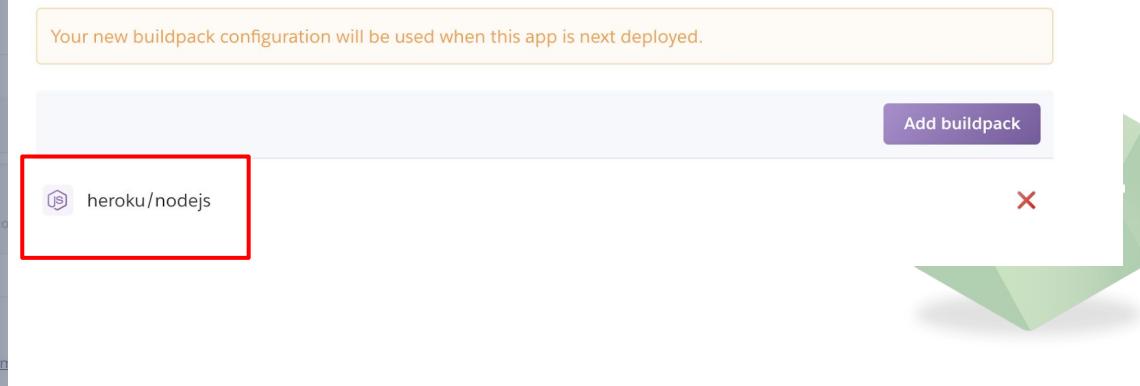


Your new buildpack configuration will be used when this app is next deployed.

heroku/nodejs

Add buildpack

X



Hands on: Rodando na nuvem

- Na aba deploy, vamos fazer um novo deploy manual para ativar o buildpack

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to automatically deploy

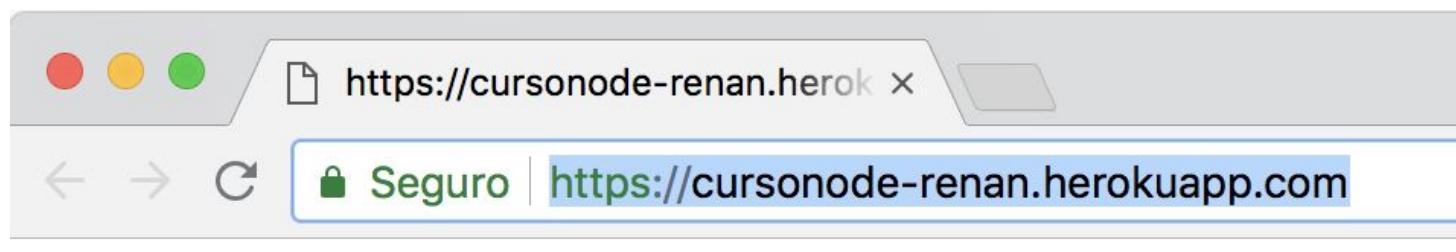
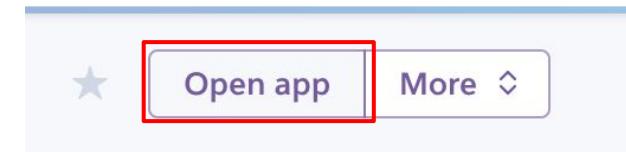
master

Deploy Branch



Hands on: Rodando na nuvem

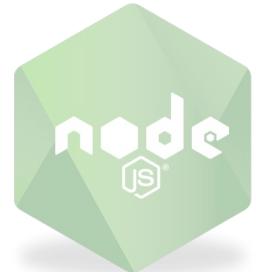
- Se abrirmos a aplicação pelo botão "Open App", podemos ver nossa aplicação Hello World.



Hands on: Rodando testes automatizados no CI

- Criar arquivo ".travis.yml" e **subir** para o **git**
 - `git add .`
 - `git commit -m "add travis.yml"`
 - `git push origin master`

```
language: node_js
node_js:
  - "9.10.0"
services:
  - mongodb
env:
  - NODE_ENV=TEST
```



Hands on: Rodando testes automatizados no CI

- Conectar no site do TravisCI

Travis CI [About Us](#) [Blog](#) [Status](#) [Documentation](#)

[Sign in with GitHub](#)

Test and Deploy with Confidence

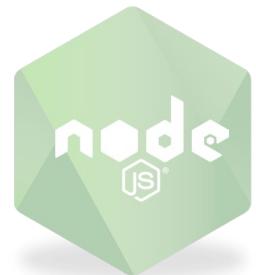
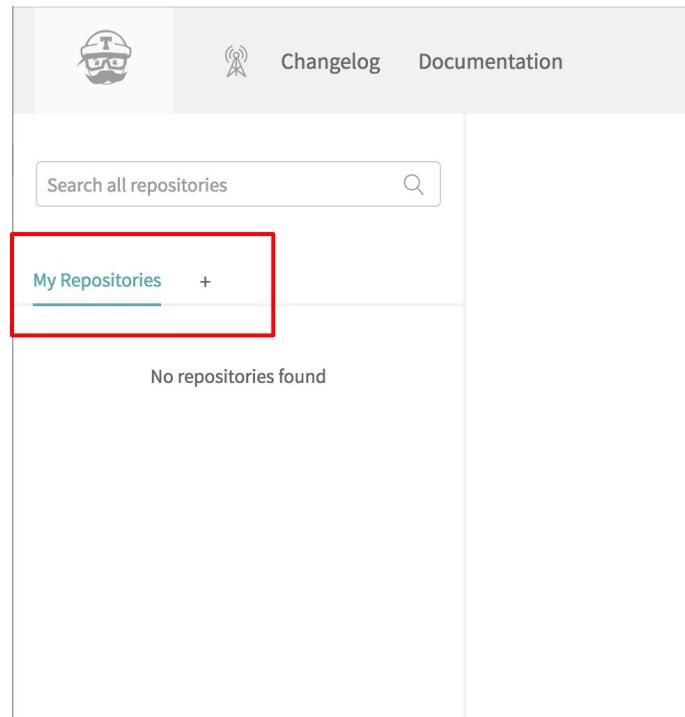
Easily sync your GitHub projects with Travis CI and you'll be testing your code in minutes!



Sign Up

Hands on: Rodando testes automatizados no CI

- Vamos conectar o github no TravisCI



Hands on: Rodando testes automatizados no CI

- Selecione o repositório e habilite

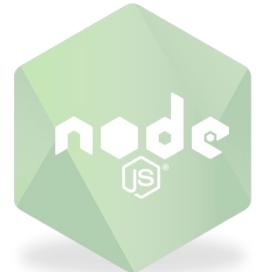
The screenshot shows the Travis CI user interface. At the top, there's a navigation bar with links for "Travis CI", "Changelog", and "Documentation". On the right, there's a user profile picture and a dropdown menu.

The main area displays the user's account information: "MY ACCOUNT" with a profile picture of Renan Pupin, the name "Renan Pupin", and the handle "@renanpupin". Below this is a "Sync account" button.

On the left, there's a sidebar titled "ORGANIZATIONS" listing "fctlabs", "ESII-Unesp", and "kingsfood". At the bottom of the sidebar, there's a link "MISSING AN ORGANIZATION? Review and add your authorized organizations."

The central part of the screen shows "Repositories" and "Settings" tabs. A message states: "We're only showing your public repositories. You can find your private projects on [travis-ci.com](#)".

Below this is a section titled "Legacy Services Integration" which lists "cursonode" twice. The second entry is highlighted with a red box. To the right of the second entry are a toggle switch and a "Settings" button.



Hands on: Rodando testes automatizados no CI

- Vamos adicionar o app do travis no github
 - <https://github.com/marketplace/travis-ci/>
- Selecione o plano opensource e pressione o botão "Install it for free"

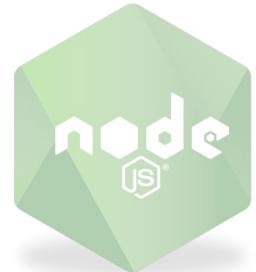
The screenshot shows the Travis CI pricing page. It features a table with four rows: 'Open Source' (free), 'ONE' (\$69/month), 'THREE' (\$199/month), and 'SIX' (\$349/month). The 'Open Source' row is highlighted with a blue background. The 'Install it for free' button in the 'ONE' row is highlighted with a red box.

Pricing and setup	
Open Source We offer free CI for Open Source projects	\$0
ONE <small>Free Trial</small> Unlimited builds, 1 job at a time. Ideal for hobby and small projects.	\$69 / month
THREE <small>Free Trial</small> Unlimited builds, 3 jobs at a time. Best suited for small teams.	\$199 / month
SIX <small>Free Trial</small> Unlimited builds, 6 jobs at a time. Great for growing teams.	\$349 / month

Travis CI
Open Source
We offer free CI for Open Source projects
✓ Unlimited public repositories
✓ Unlimited collaborators

Install it for free Next: Confirm your installation location.

Travis CI is provided by a third-party and is governed by separate [terms](#), [privacy policy](#), and [support contact](#).



Hands on: Rodando testes automatizados no CI

- Selecione o repositório e pressione "Save"

Repository access

All repositories
This applies to all current *and* future repositories.

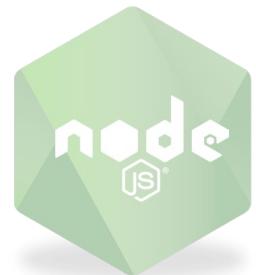
Only select repositories

Select repositories ▾

cursoronode

renanpupin/cursoronode no description

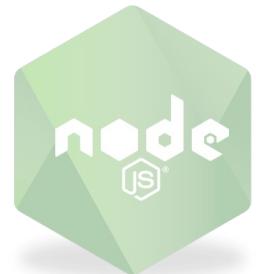
Save Cancel



Hands on: Rodando testes automatizados no CI

- Agora o Github já possui o app do travis instalado

The screenshot shows the GitHub Settings page for the repository 'renanpupin / cursonode'. The left sidebar has a 'Integrations & services' section selected. In the main area, under 'Installed GitHub Apps', there is a single entry for 'Travis CI' with a 'Configure' button. Below this, under 'Services', there is a note: 'Note: GitHub Services are being deprecated. Please contact your integrator for more information on how to migrate or replace a service with webhooks or GitHub Apps.' A yellow box highlights this note. At the bottom, a footer note states: 'Services are pre-built integrations that perform certain actions when events occur on GitHub.'



Hands on: Rodando testes automatizados no CI

- No Heroku, vamos habilitar a opção de apenas fazer deploy quando passar nos testes.
- Na aba deploy, marque a opção "Wait for CI to pass before deploy"

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

 Automatic deploys from  master are enabled

Every push to `master` will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. [Learn more](#).

Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

[Disable Automatic Deploys](#)

Hands on: Rodando testes automatizados no CI

- Vamos fazer um novo commit no git para fazer com o que o Travis execute um novo build ao receber um novo push
- Altere a resposta "Hello World!" para "Hello World Travis!" no server.js
 - `res.send("Hello World Travis!");`

The screenshot shows the Travis CI interface. At the top, there's a navigation bar with links for 'Travis CI', 'Changelog', 'Documentation', and a user profile icon. Below the navigation, a search bar says 'Search all repositories'. On the left, a sidebar titled 'My Repositories' lists 'renanpupin/cursonode' with a note '# 1' and 'Duration: -'. The main area displays the repository 'renanpupin / cursonode' with a 'build unknown' status. It has tabs for 'Current', 'Branches', 'Build History', and 'Pull Requests'. Under the 'Current' tab, it shows a single build for the 'master' branch. The build details include: 'Commit ac452c4' (with a link), 'Compare a384ae3..ac452c4' (with a link), 'Branch master' (with a link), and the author 'Renan Pupin'. To the right of the build details, there are buttons for '#1 queued' (with a cancel icon) and 'Running for -'. A green decorative element with a white 'C' logo is on the far right.

Hands on: Rodando testes automatizados no CI

- Em seguida após rodar os testes, o Travis irá completar o build

```
> cursonode@1.0.0 test /home/travis/build/renanpupin/cursonode
> mocha --exit

Example app listening on port 5000
Test server working
  ✓ should get a server message

Test ToDo is Working
  ✓ should get Todos (132ms)
  ✓ should create Todos
{ is_complete: false,
  _id: 5bae50e83390a10aa1d3f52c,
  title: 'Todo Teste',
  created_at: 2018-09-28T16:03:52.008Z,
  __v: 0 }
updatedTodo { is_complete: true,
  _id: 5bae50e83390a10aa1d3f52c,
  title: 'Todo Teste',
  created_at: 2018-09-28T16:03:52.008Z,
  __v: 0,
  completed_at: 2018-09-28T16:03:52.035Z }
  ✓ should update Todos
{ is_complete: true,
  _id: 5bae50e83390a10aa1d3f52c,
  title: 'Todo Teste',
  created_at: 2018-09-28T16:03:52.008Z,
  __v: 0,
  completed_at: 2018-09-28T16:03:52.035Z }
  ✓ should remove Todos

  5 passing (233ms)
```

Hands on: Rodando testes automatizados no CI

- Após fazer o build, em caso de sucesso nossa aplicação no heroku será atualizada

renanpupin / cursonode  build passing

Current Branches Build History Pull Requests > Build #2

✓ master test commit

- o Commit f7a75cc ↗
- Compare ac452c4..f7a75cc ↗
- Branch master ↗

Renan Pupin

Job log View config

Personal > cursonode-renan

GITHUB renanpupin/cursonode master

Overview Resources Deploy Metrics Activity

Activity Feed

renan_athas@hotmail.com: Deployed f7a75cc2 Today at 1:22 PM · v8 · Compare diff

Hands on: Rodando testes automatizados no CI

- Se forçarmos um erro no teste, o teste automatizado não irá funcionar e a aplicação não será publicada automaticamente

 [renanpupin / cursonode](#)  build passing

[Current](#) [Branches](#) [Build History](#) [Pull Requests](#)

 [master](#) [test fail deploy](#)

 #3 failed

 Commit 5455467 ↗

 Ran for 41 sec

 Compare f7a75cc..5455467 ↗

 3 minutes ago

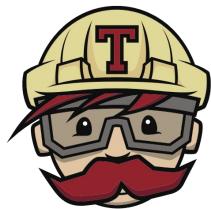
 Branch master ↗

 Renan Pupin

Hands on: Bônus?

- Testes no FalaFreud



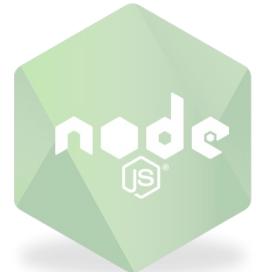


Dúvidas?



Links Úteis

- Código Fonte
 - <https://github.com/nodejs/node>
- Documentação
 - <https://nodejs.org/en/docs/>
- API
 - <https://nodejs.org/api/>
- NodeSchool
 - <https://nodeschool.io/pt-br/>
- Stack Overflow
 - <https://stackoverflow.com/questions/tagged/node.js>
- Npm
 - <https://www.npmjs.com/>



Comunidade

- We encourage all kinds of contribution from the community.
- The Node.js community is large, inclusive, and excited to enable as many users to contribute in whatever way they can. If you want to report an issue, help with documentation or contribute to the code base of the project, you've come to the right place. Explore our community resources to find out how you can help:
- <https://nodejs.org/en/about/community/>



Referências

- https://www.slideshare.net/giovanni.bassi/introduo-ao-nodejs-37802907?qid=71b312be-b14b-424c-b6c7-34fba93f5e78&v=&b=&from_search=4
- <https://www.slideshare.net/search/slideshow?searchfrom=header&q=nodejs>
- <https://www.netguru.co/blog/pros-cons-use-node.js-backend>
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Lexical_grammar
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Values,_variables,_and_literals



Referências

- <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Closures>



Obrigado!

- @renanpupin

renan@falafreud.com

- @andreoandreotti

andre@falafreud.com

