Apostila NodeJS - 16 horas - CT Novatec

Esta é a apostila do curso presencial de NodeJS de 16 horas que irei ministrar no Centro de Treinamento Novatec (http://ctnovatec.com.br/cursos/trilha-front-end/curso-de-nodejs/).

Instrutor

Eu sou o **William Bruno**, aka **wbruno**, desenvolvedor web apaixonado por JavaScript. Você pode ler alguns artigos que escrevi no meu **blog pessoal** (http://wbruno.com.br), para o portal **iMasters**(http://imasters.com.br/perfil/william_bruno/), para o blog **UOL Host** e mais alguns outros lugares por ai. Caso você participe do **fórum imasters**, poderá me encontrar por lá também. (http://forum.imasters.com.br/user/69222-william-bruno/).

Outras formas de contato:
https://about.me/wbruno
<wbrunom@gmail.com>

O conteúdo

Eu gravei um Workshop que está publicado na **Eventials** sobre como **Construir uma API REST com ExpressJS** (https://www.eventials.com/wbruno.moraes/construindo-uma-api-rest-com-expressjs-nodejs-2/), que é exatamente uma parte do foco desse curso presencial.

Nessa apostila irei abordar com mais “calma” e mais detalhadamente cada etapa desde a construção da API até a listagem dos produtos no site e da edição no admin. Depois escreveremos alguns testes unitários e funcionais para garantir que o nosso código seja confiável e sólido.

Sempre que possível irei colocar links relacionados para que você consiga complementar seus estudos e use como guia de referência em dúvidas sobre o tema.

Desafios

O HackerRank é uma plataforma online com desafios de programação:
https://www.hackerrank.com/domains

Introdução

- O que é NodeJS?
- Instalando o NodeJS;
- O arquivo package.json;
- O npm (Node Package Manager);
- GruntJS
- ExpressJS;

Criando sua primeira aplicação

- Iniciando um servidor;
- Servindo estáticos;
- MongoDB;

O que é o No-SQL?

- Como utilizar o MongoDB?
- CRUD no terminal do MongoDB;

REST

- Criando o model e um DAO;
- Criando controllers;
- Construindo o admin;

Formulários (Listagem de dados)

- Angular ou Backbone;
- Adicionar autenticação;
- A vitrine;

Listar os produtos

- Pagina interna;

Testes

- Mocha;
- Assert;

Controle de versão

- GitHub;

Deploy

- Colocando sua aplicação disponível na internet;

O que é o NodeJS?

NodeJS é uma plataforma escrita em cima da engine JavaScript do Chrome, a poderosa **V8**, para construir escaláveis, utilizamos a linguagem JavaScript. https://nodejs.org/

Primeiros passos
http://tableless.com.br/o-que-nodejs-primeiros-passos-com-node-js/

Instalando o NodeJS

Vou deixar alguns links sobre como instalar, basta seguir as instruções corespondentes ao seu sistema operacional.

- **Todas** http://nodebr.com/instalando-node-js-atraves-do-gerenciador-de-pacotes/
- **Mac OSx** http://udgwebdev.com/node-js-para-leigos-instalacao-e-configuracao/
- **Windows** http://mateussouzaweb.com/blog/nodejs/tutorial-instalando-nodejs-no-windows

O npm (Node Package Manager)

O que é
https://docs.npmjs.com/getting-started/what-is-npm

O npm (https://www.npmjs.com) é um serviço grátis (para pacotes públicos) que possibilita que os desenvolvedores NodeJS criem e compartihem códigos com a comunidade. Facilitando a distribuição de módulos e ferramentas escritas em JavaScript.

Por exemplo, se você quiser utilizar o **ExpressJS** (http://expressjs.com) que é um framework rápido e minimalista para construção de rotas em NodeJS, você não precisa entrar no github deles, baixar o código, e "instalar" (colar) no seu projeto. Basta digitar no seu terminal:

```
$ npm install express --save
```

Que o módulo será baixado dentro de uma pasta chamada `node_modules` e pronto, você já pode utilizar o express.

O lado bom

O lado “bom” do npm é que existem muitos módulos a disposição. Então sempre que você tiver que fazer alguma coisa em NodeJS, vale a pena dar uma pesquisada no https://www.npmjs.com, se já existe algum módulo que faça o que você quer, ou uma parte do que você precisa. Agilizando muito assim o desenvolvimento da sua aplicação.

O lado ruim

O lado “ruim” do npm é que existem muitos módulos a disposição! Opa, mas eu não acabei de falar que esse era o lado bom ? pois é, acontece que por ser completamente público e colaborativo (*Open Source*), você encontrará diversos módulos com o mesmo propósito, que realizam as mesmas tarefas. Cabe a nós conseguir escolher aquele que melhor resolve a nossa questão, e para isso eu costumo seguir alguns simples passos:

- Procure um módulo que esteja sendo mantido (com atualizações frequentes, olhe se o último commit não foi a anos atrás,mas sim alguns dias ou semanas).
- Se existe alguma comunidade ativa utilizando o módulo (olhe o número de estrelas, forks, issues abertas e fechadas…)
- Importante que ele possua uma boa documentação dos métodos públicos e da forma de uso, assim você não terá que ler o código fonte do projeto para realizar uma tarefa simples. (ler o fonte pode ser interessante, quando você tiver um tempo para isso, ou precisar de alguma otimização mais baixo nível).
- Procure por benchmarks onde são comparados módulos alternativos, e decida por aquele com melhor performance.

Assim você foca na sua aplicação e em desenvolver os códigos da regra de negócio.

O arquivo package.json

Todo projeto NodeJS contém um arquivo chamado `package.json` na raiz. Esse arquivo contém informações sobre a aplicação, assim como as dependências dela.
Sempre use o comando `npm init` para criar a estrutura inicial do seu `package.json`.

```
$ npm init
```

Basta ir respondendo as perguntas, uma a uma, como “nome da aplicação”, “versão”, “link do repositório git”, no final, confirme com um `yes`, e um arquivo mais ou menos parecido com esse será criado:

```
{
  "name": "curso-nodejs-16h-novatec",
  "version": "0.0.1",
  "description": "Loja Virtual criada durante o curso de NodeJS
presencial no CT Novatec",
  "main": "app.js",
  "scripts": {
     "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "William Bruno <wbrunom@gmail.com>
(http://wbruno.com.br/)",
  "license": "ISC"
}
```

Feito isso, podemos começar.

GruntJS

Nem só para aplicações web vive o NodeJS. O GruntJS é um ótimo exemplo de como podemos utiliza-lo como uma ferramenta de linha de comando.
http://gruntjs.com

A idéia por trás do GruntJS é automatizar tarefas. Desde as mais simples como concatenar arquivos .css e .js, minificar o código (remover espaços, tabulações, encurtar nomes de variáveis) até fazer deploy!

Tudo depende de quais plugins você utilizar junto com ele.O GruntJS sozinho, apenas gerencia esses módulos e organiza a ordem e quais tarefas você irá utilizar, mas serão os módulos que você plugar que irão realizar os trabalhos, como por exemplo:

https://github.com/gruntjs/grunt-contrib-cssmin
https://github.com/gruntjs/grunt-contrib-uglify
https://github.com/gruntjs/grunt-contrib-watch

Um artigo complementar: http://simplesideias.com.br/usando-gruntjs
Você poderá encontrar muitos outros sobre esse assunto, caso sinta alguma dúvida ou precise implementar mais coisas no teu grunt.

Utilizando

Após instalarmos o NodeJS, em uma aba do terminal, precisamos do grunt-cli como um módulo global do computador, para podermos utilizar o comando 'grunt-cli', sem precisar identificar o diretório em que ele está instalado, e para qualquer uma das nossas aplicações ter acesso aos comandos.

```
$ npm install grunt-cli -g
```

Criamos o `package.json` desse projeto com o comando

```
$ npm init
```

E já podemos instalar as dependências locais, tais como:

```
$ npm install grunt grunt-contrib-watch grunt-contrib-uglify
grunt-contrib-cssmin --save-dev
```

Note que utilizei a flag `--save-dev` para que esses módulos entrem no package.json, mas na sessão "`devDependencies`", afinal não precisamos minificar css e javascript na hora de rodar a aplicação em produção, precisamos apenas em desenvolvimento, enquanto estivermos codando.

Todos esses módulos do grunt, ficarão dentro da pasta `node_modules` na raiz da nossa aplicação, no mesmo nível do arquivo `package.json`.

Gruntfile.js

O comando `grunt` irá sempre buscar um arquivo chamado `Gruntfile.js` na raiz do projeto. É nele onde iremos declarar as nossas tasks: concatenar e minificar css e js.

A estrutura inicial desse arquivo é:

```
module.exports = function (grunt) {
  'use strict';

  grunt.initConfig({
     pkg: grunt.file.readJSON('package.json')
  });

  grunt.registerTask('default', []);
};
```

Utilizando o CSS min

Basta adicionar essa propriedade no json do grunt:

```
cssmin: {
  with_banner: {
    options: {
      banner: '/*\n' +
           'Minified CSS of <%= pkg.name %>\n' +
           '*/\n'
    },
    files: {
      'public/css/all.min.css': [
           'src/css/normaset.css',
           'src/css/main.css'
      ]
    }
  }
},
```

O nosso objetivo é pegar os arquivos .css da pasta `src/css` e minifica-los em `public/css`

Utilizando o Uglify

Segue a mesma idéia:

```
uglify: {
  options: {
     banner: '/* Minified JavaScript of <%= pkg.name %> */\n'
  },
  my_target: {
     files: {
     'public/javascript/all.min.js': [
     'src/js/vendor/*.js',
     'src/js/*.js'
     ]
     }
  }
},
```

E por fim, vamos implementar o watch, para deixarmos o comando:

```
$ grunt watch
```

Rodando em uma aba no terminal, e assim que algum arquivo fonte `.css` ou `.js` for modificado, o grunt irá minificá-lo para nós.

```
watch: {
  options: {
     livereload: true,
     spawn: false
  },
  css: {
     files: 'src/css/*.css',
     tasks: ['cssmin']
  },
  js: {
     files: 'src/js/*.js',
     tasks: ['uglify']
  }
}
```

Além disso, adicionaremos o suporte ao livereload no nosso html, apenas incluindo essa tag script no html:

```
<script src="//localhost:35729/livereload.js"></script>
```

Assim, sempre que o grunt terminar alguma tarefa do watch, a nossa página será automaticamente recarregada, e não precisaremos quebrar a tecla F5 de tanto usá-la.

Deploy

http://imasters.com.br/front-end/javascript/configurando-nodejs-em-producao-em-um-cloud-server/