Rodrigo Cichetto Monteiro

Desenvolvimento de aplicações com JavaScript

Brasil

2018, v-1.0.0

Rodrigo Cichetto Monteiro

Desenvolvimento de aplicações com JavaScript

Trabalho apresentado a UNIP - UNIVERSIDADE PAULISTA como pré-requisito para obtenção da Certificação de Conclusão do Curso de Bacharelado em Ciência da Computação. Orientador: Prof. Leandro Carlos Fernandez

Universidade Paulista – UNIP

Faculdade de Ciência da Computação

Programa de Graduação

Orientador: Leandro Carlos Fernandez

Coorientador: Danielle Colturato

Brasil

2018, v-1.0.0

Rodrigo Cichetto Monteiro
Desenvolvimento de aplicações com JavaScript/ Rodrigo Cichetto Monteiro. – Brasil, 2018, v-1.0.0-
31 p. : il. (algumas color.) ; 30 cm.

Orientador: Leandro Carlos Fernandez

Tese (Graduação) – Universidade Paulista – UNIP
Faculdade de Ciência da Computação
Programa de Graduação, 2018, v-1.0.0.

1. JavaScript. 2. Desenvolvimento de aplicações. I. Orientador Leandro Carlos Fernandez. II. Universidade Paulista – UNIP. III. Faculdade de Ciência da Computação. IV. Desenvolvimento de aplicações com JavaScriptV. Rodrigo Cichetto Monteiro

CDU 02:141:005.7

Rodrigo Cichetto Monteiro

Desenvolvimento de aplicações com JavaScript

Trabalho apresentado a UNIP - UNIVERSIDADE PAULISTA como pré-requisito para obtenção da Certificação de Conclusão do Curso de Bacharelado em Ciência da Computação. Orientador: Prof. Leandro Carlos Fernandez

Trabalho aprovado. Brasil, 24 de novembro de 2018:

Leandro Carlos Fernandez
Orientador

Professor
Convidado 1

Professor
Convidado 2

Brasil
2018, v-1.0.0

"Cada sonho que você deixa pra trás,
é um pedaço do seu futuro que deixa de existir.
(Steve Jobs)

Resumo

Fazendo parte das três principais tecnologias que movem a internet, sendo elas HTML, CSS e claro o JavaScript, não é mais uma linguagem voltada somente para desenvolvedores front-end, mas sim uma obrigação para todo desenvolvedor, que tem como obrigação conhecer pelo menos o básico de JS. Nos últimos tempos a linguagem ganhou muita importância em quaisquer cenários, e vem sendo utilizada em sites, aplicações, mobile, servidores, automação de testes, automação de tarefas, internet das coisas, entre outros. Este trabalho tem como principal objetivo atualizar o leitor através de um compilado de informações sobre as tendências do que mais vem sendo utilizado para a construção de aplicações utilizando a linguagem, mostrando os frameworks mais recentes e mais famosos. Mas lembre-se com grandes poderes vem grandes responsabilidades.

Palavras-chaves: javascript, typescript, frameworks

Abstract

Being part of the three main technologies that move the internet, being HTML, CSS and clear JavaScript, is no longer a language aimed only at front-end developers, but rather an obligation for every developer, who has to know at least the basics of JS. In recent times the JavaScript language has gained a lot of importance in any scenarios involving programming, and has been used in websites, applications, servers, automation of tests, automation of tasks, internet of things, among others. This work has as main objective to update the reader through a compilation of information about the trends of what is being used to build applications using the language, showing the latest and most famous frameworks. But remember with great powers comes great responsibilities.

Key-words: javascript, typescript, frameworks

Lista de ilustrações

Lista de abreviaturas e siglas

JS	JavaScript
TS	TypeScript
HTML	Abreviação para *HyperText Markup Language*, que em português significa Linguagem de Marcação de Hipertexto.
DOM	Abreviação para *Document Object Model*, que em português significa Modelo de Objetos e Documentos.
CSS	Abreviação para *Cascading Style Sheets*, que em português significa Folhas de Estilo em Cascata.
ECMA	Abreviação para *European Computer Manufacturers Association*, que em português significa Associação Européia de Fabricantes de Computadores.
API	Abreviação para *Application Programming Interface*, que em português significa Interface de programação de aplicações.

Sumário

1 Introdução

Inicialmente implementada com o objetivo no desenvolvimento web para o lado do cliente, a linguagem criada por Brendan Eich enquanto trabalhou na Netscape se tornou uma das linguagens mais populares da atualidade, sendo a terceira camada do bolo quando se fala de tecnologias web, das quais HTML e CSS também fazem parte.

Com navegadores cada vez mais modernos a tendência é que nossos sites também fiquem cada vez mais sofisticados.

Todo projeto que iniciamos existe uma origem, uma necessidade e um problema a ser resolvido. Há algum tempo, um grande problema ganhou atenção: a reutilização de trechos de código. São módulos de código (trechos em HTML, CSS, JavaScript, etc) que juntos criam padronizações e organizações, mas sobretudo flexibilidade. São os chamados frameworks.

Nos últimos anos a linguagem JavaScript ganhou maior importância, com o surgimento de bibliotecas e frameworks que possibilitaram o desenvolvimento de não somente web sites mas também aplicativos, single page applications, programas desktop, progressive web apps e muito mais. Sendo alguns deles Angular, React, Vue, jQuery e Node.js.

Talvez nenhuma outra linguagem tenha conseguido ganhar tanta atenção dos desenvolvedores como o JavaScript. Em busca de sua identidade a linguagem foi a única que conseguiu se enraizar nos navegadores, e atualmente também passou a se empoderar dos servidores de alta performance através do Node.js.

O surgimento do Node.js possibilitou um novo mundo para os desenvolvedores, levando a linguagem para um novo patamar. Criado com um modelo não bloqueante na entrada e saída de dados, possibilitou que a linguagem fosse levada agora também para aplicações back-end, ou seja, para o server-side (lado do servidor).

Com o tempo a Microsoft desenvolveu o TypeScript, que de uma forma simples é o JavaScript acrescido de tipagem de dados, ou seja, agora erros podem ser detectados durante a digitação do código, sem dúvidas podemos considerar o TypeScript como uma evolução do JavaScript.

Este trabalho tem como objetivo atualizar o leitor do que há de mais novo no desenvolvimento com a linguagem JavaScript, para melhor compreensão do conteúdo apresentado é necessário conhecimento prévio básico da linguagem.

2 Pesquisa bibliográfica

2.1 Arquitetura

2.1.1 Banco de dados

2.1.2 Serviço

2.1.3 Cliente

2.1.4 Servidor

2.2 JavaScript

JavaScript é uma linguagem de programação dinâmica interpretada, inicialmente utilizada pelos navegadores para execução de *scripts* no lado do cliente, ou seja, no seu *browser*. Os *scripts* são incorporados a páginas HTML tendo como função adicionar interatividade para o usuário.

Atualmente, é praticamente impossível imaginar a internet sem a existência do JavaScript, certamente você já se deparou com alguns dos exemplos como os famosos carrosseis (Figura 1, página 21), as tão importantes validações de formulários (Figura 2, página 21), ou até mesmo o *modal* (Figura 3, página 21) que mostram a aplicação da linguagem em *web sites*.

A web está infestada de códigos JavaScript, pode ter certeza de que diariamente muitos *scripts* estão sendo executados durante uma pesquisa no Google, seu tempo no Facebook, acesso a seu banco na internet, ou até mesmo para leitura de uma notícia em um portal. Seu uso primário na web é de funções incluídas no HTML que interagem com o Modelo de Objetos e Documentos (DOM) da página.

DOM é uma interface de programação utilizada para documentos HTML, ele fornece a representação e interação com objetos os estruturando em nós de uma árvore chamada de Árvore DOM. É assim que o JavaScript consegue manusear as estruturas, estilos e conteúdos dos elementos presentes no HTML. Vale ressaltar que ele não faz parte da linguagem JavaScript, pois também pode ser acessado por outras linguagens.

O JavaScript apresenta uma sintaxe simples que facilita o aprendizado, mas não confunda pois a primeira vista muitos desenvolvedores podem acreditar que a linguagem é defeituosa ou esquisita, pois não compreendem o real poder que se esconde por trás desta simplicidade.

Inicialmente classificada como linguagem do tipo *client side*, é por si só uma linguagem compacta, mas muito flexível, com comportamentos diferenciados das demais ela permite, por exemplo, que um objeto tenha seus atributos adicionados ou removidos em tempo de execução, o que não é muito comum para desenvolvedores de outras linguagens.

É uma linguagem interpretada, pois seus comandos são executados sem que haja necessidade de compilação, tendo como interpretador de *script* o *browser* do usuário. Sendo assim independente de plataformas, como os comandos são interpretados pelo navegador do usuário é irrelevante se o usuário está utilizando Windows, Linux ou Mac OS.

Sua tipagem é dinâmica, ou seja, tipos são associados com valores. Por exemplo, uma variável pode ser associada a um número e posteriormente associada a um texto.

Baseada em objetos, no JavaScript os objetos são matrizes associativas, ou seja `obj.a = 10` e `obj["a"] = 10` são equivalentes, sendo a única diferença a própria sintaxe.

As funções em JS são tratadas como objetos, possuindo métodos e propriedades, podendo assim serem atríbuidas a variáveis, retornadas como objetos, ou até mesmo passadas como argumentos para outras funções, o que é chamado de aninhamento de funções.

Com JavaScript também é possível a detecção de eventos, sempre que algo de importante acontece é disparado um evento, o clique de um botão, o preenchimento de um campo de formulário, a movimentação do mouse, são alguns exemplos dos eventos que são disparados. Isso nos permite reagir a estes eventos deixando assim que nossa aplicação deixe de ser estática.

A linguagem possui grande tolerância a erros, uma vez que conversões automáticas são realizadas durante suas operações.

Podemos executar códigos JS de várias formas na web, sendo uma delas pelo próprio console do navegador pressionando as teclas F12, importando um script em uma página HTML, ou até mesmo envolvendo o trecho de código na tag <script>.

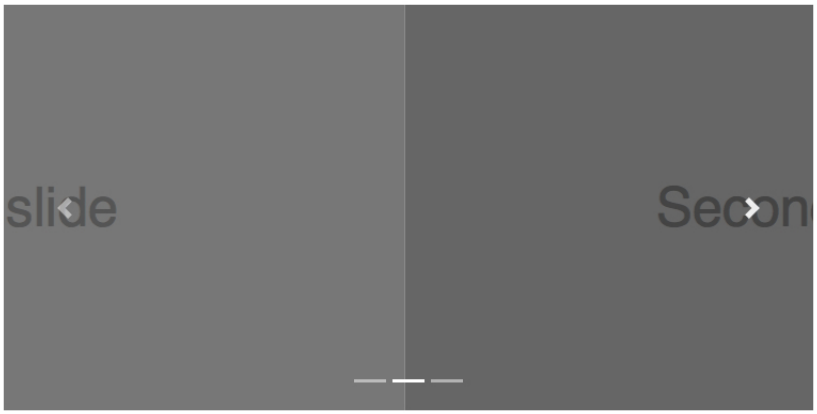
No início da internet as páginas não eram nada interativas, documentos apresentavam seu conteúdo exatamente como foram criados para serem exibidos no navegador e só. O JavaScript revolucionou o que podemos fazer, hoje não só na web, mas praticamente em todas as áreas que se possa programar.

No próximo capítulo abordaremos de como surgiu a linguagem, como ela evoluiu e atua no presente e a necessidade de padronizações.

2.2.1 A linguagem nos dias atuais

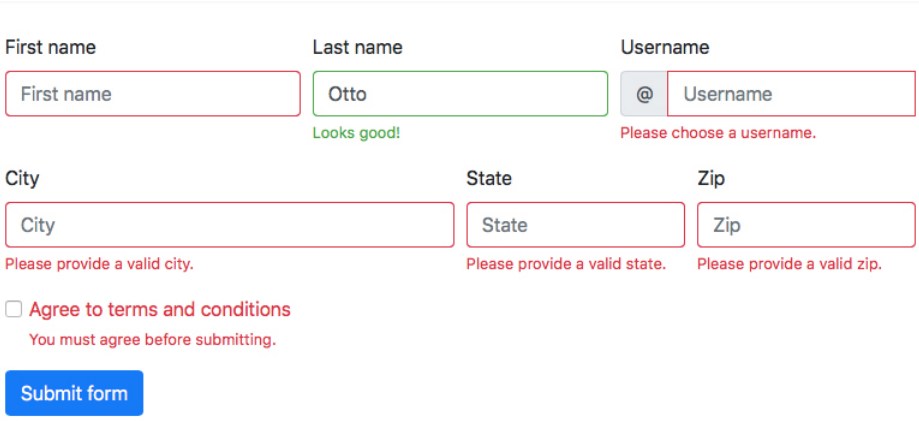
Se existe alguma linguagem que evoluiu nos últimos tempos, essa linguagem é o JavaScript. Conforme a linguagem evoluiu se tornou mais poderosa e independente do

Figura 1 – *Carrousel* - Um componente de apresentação de slides para percorrer imagens ou slides de texto - como um carrossel. – Exemplo de aplicação do JS



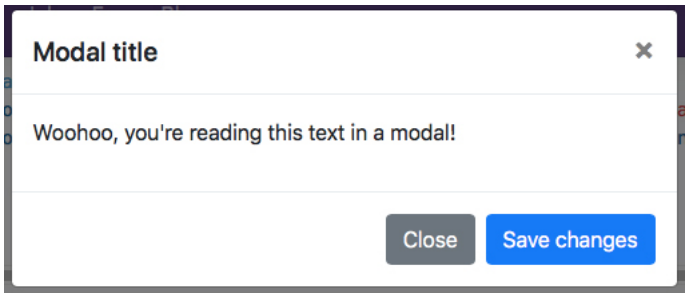
Fonte: Bootstrap <https://getbootstrap.com/docs/4.1/components/carousel/>

Figura 2 – *Validation* - Validação de formulários – Exemplo de aplicação do JS



Fonte: Bootstrap <https://getbootstrap.com/docs/4.1/components/forms/>

Figura 3 – *Modal* - Caixas de diálogo para notificações ao usuário. – Exemplo de aplicação do JS



Fonte: Bootstrap <https://getbootstrap.com/docs/4.1/components/modal/>

navegador. Isso possibilitou que a linguagem fosse utilizada não somente para a web como *client side* mas agora em vários lugares.

2.2.1.1 JavaScript além dos navegadores

Inicialmente tratada como um extra para os navegadores, podemos dizer que a linguagem caminhou com o avanço da tecnologia, isso possibilitou o uso da linguagem em diversas áreas sendo elas:

a) Aplicações web

b) Aplicativos mobile

c) Automação de testes e de tarefas

d) Controle de hardware

e) Desenvolvimento de jogos

f) Internet das coisas

g) Realidade virtual e aumentada

h) Softwares desktop

i) Servidores

Praticamente tudo que envolve programação o JavaScript está presente, para criar um software desktop existe por exemplo o *framework* Electron (https://electronjs.org), desenvolvido pelo GitHub ele possibilita criar aplicativos desktop multiplataforma através do JavaScript. No desenvolvimento de jogos a *engine* Unity (https://unity3d.com/pt) é uma das plataformas que oferecem suporte a linguagem.

O ecossistema JavaScript é gigante e tem atraído empresas de todos os portes. Hoje grandes empresas como Google, Microsoft, Netflix, Uber e Linkedin usam JavaScript até mesmo no *back-end*.

Isso acabou impactando o mundo dos desenvolvedores, fazendo com que seja obrigatório todo programador saber pelo menos o básico da linguagem, mesmo atuando na área de *back-end* ou até mesmo de teste.

2.2.1.2 ECMA

Atualmente a linguagem se encontra na 9ª edição chamada de ECMAScript 2018, finalizada em Junho de 2018, suas últimas contribuições de maior expressão foram a inclusão dos operadores *await* e *async*, possibilitando agora que a linguagem trabalhe com funções assíncronas de uma maneira simples.

Pela linguagem rodar em ambientes que podem variar, algo importante a considerar é a compatibilidade entre os navegadores. Para isso é necessário um padrão a seguir, tal criado e mantido até hoje pelo ECMA Internacional.

Já em 1996 a Netscape, detentora do JavaScript, anúnciou que submetia a linguagem para o ECMA Internacional como candidata a padrão industrial, resultando então no ECMAScript.

Padronização que define a estrutura da linguagem, seus comportamentos e comandos, dando assim um padrão aos interpretadores da linguagem.

Com participação colaborativa de empresas que implementam o *run-time* da linguagem, como Mozilla, Google, Microsoft e Apple, além da participação de desenvolvedores da comunidade, o ECMA coordena e faz o trabalho de desenvolvimento contínuo e descentralizado do JS.

Em seu site <http://www.ecma-international.org/publications/standards/Ecma-262.htm>, a ECMA disponibiliza informações como versão atual e a documentação da linguagem, porém aconselho a buscar documentações na internet, uma boa referência a seguir é o site da Mozilla <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>, ou também o site da W3Schools <https://www.w3schools.com/jsref/default.asp>.

A 10ª edição já está em desenvolvimento, devendo chegar até o final de Junho no ano de 2019, sendo chamada de ECMAScript 2019.

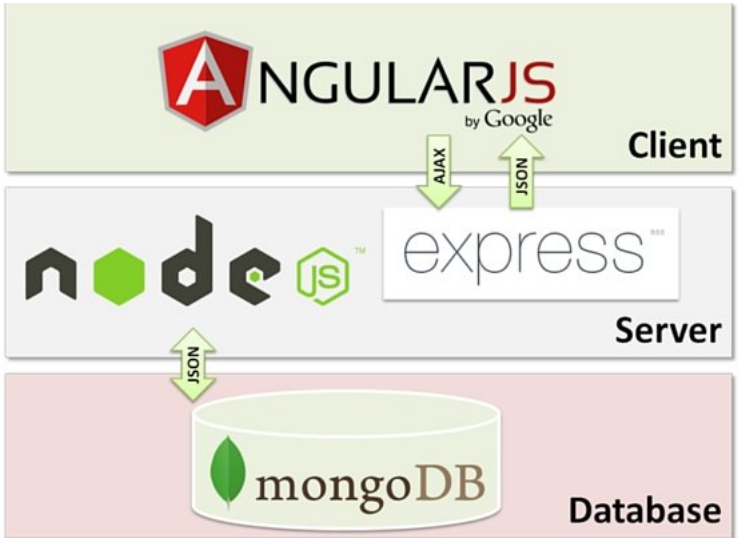
3 Metodologia e ferramentas

3.1 A pilha MEAN

É bem atraente a possibilidade de usar apenas uma linguagem em todo o desenvolvimento, ganhando não só reaproveitamento de recursos humanos como de código, sendo assim possível criar códigos que podem ser executados em qualquer plataforma que interprete JavaScript, isso é o que acontece com o MEAN.

Das iniciais das ferramentas MongoDB, Express, Angular e Node.js, MEAN é o nome dado quando integramos todas essas ferramentas para desenvolver uma aplicação. Sendo que todas utilizam como linguagem o JavaScript, possibilitando que um programador da linguagem tenha maior facilidade para trabalhar em todas as partes da aplicação, seja front-end, back-end ou banco de dados.

Figura 4 – *MEAN Stack* - Fluxo das ferramentas que integram o MEAN



Fonte: orangemantra <https://www.orangemantra.com/blog/utilize-the-simplicity-of-mean-stack-technology-for-your-next-project/>

Por utilizar quatro tecnologias distintas, podemos pensar que dará mais trabalho no desenvolvimento, mas ao contrário disso a MEAN Stack é utilizada até mesmo em curtas competições de programação conhecidas como Hackathon, conseguindo protótipos de forma rápida, provando assim sua produtividade.

3.1.1 MongoDB

Um banco de dados flexível, poderoso, escalonável e de alta performance orientado a documentos. Lançado em 2009, escrito em C++, o MongoDB é gratuito e de código

aberto.

Por ser orientado à documentos JSON, ou seja, retém os dados usando pares de chave/valor, podemos modelar dados de forma mais natural, utilizando a forma como os dados realmente serão utilizados em nossa aplicação, ao invés de criar várias ligações entre tabelas, o que o dá a característica ao MongoDB de banco não-relacional.

Para a execução de comandos no Mongo, existe um console que executa códigos JavaScript. Por esse motivo, desenvolvedores da linguagem terão facilidade em manter um banco MongoDB.

Em uma aplicação desenvolvida em cima do MEAN Stack o Mongo tem a responsabilidade de persistir os dados, ou seja, permite armazenar e recuperar dados.

3.1.1.1 Mongoose

Com o Mongo é possível inserir qualquer formato JSON nas coleções, por esse motivo não há nenhum controle sobre os dados inseridos. Isso pode causar inconsistência nos dados, inserindo informações inválidas podendo quebrar a aplicação, deixando a responsabilidade para o dev garantir a exatidão das informações.

Para resolver esse problema existe o Mongoose que nos ajuda a modelar objetos de forma elegante para o MongoDB, ele fornece uma solução direta e baseada em esquema para modelar os dados da nossa aplicação, incluindo conversão de tipo incorporada, validação, criação de consulta, ganchos de lógica de negócios e muito mais.

3.1.1.2 Instalação

Para instalar o MongoDB, basta acessar o link <https://www.mongodb.com/download-center> e baixar o instalador de acordo com seu sistema operacional.

É necessário criar um diretório para o Mongo escrever seus arquivos, crie a pasta "data"e dentro dela a pasta "db", se você utiliza Linux ou Mac pode utilizar os seguintes comandos:

- `mkdir -p /data/db`
- `chown -R $USER:$USER /data/db`

Para iniciar o servidor Mongo, acesse o diretório de instalação pelo terminal e execute o comando bin/mongod.exe se utilizar Windows ou bin/mongod se utilizar Linux ou Mac. Logo após é possível iniciar o MongoDB pelo comando bin/mongo , caso esteja em um ambiente Windows execute bin/mongo.exe .

Através deste terminal podemos criar bancos de dados, documentos e coleções. Se em qualquer momento for preciso obter ajuda, basta dar o comando `help` na linha de comando do terminal do Mongo.

3.1.1.3 Primeiros passos

Por padrão o terminal do Mongo inicia conectado ao banco de dados `test`. Para mudarmos para outro banco de dados utilizamos o comando `use nome_do_banco`. Caso o banco indicado não exista, o Mongo criará um novo assim que dados forem incluídos nele.

Insert, através da função `insert()` insere novos dados e pode receber como parâmetro um objeto JSON ou um array de objetos.

```
1 // Exemplo insert
2 db.nome_do_banco.insert()
```

Find, através da função `find()`, busca por objetos inseridos em uma *collection*, podendo receber como parâmetro um objeto com os critérios de filtragem, ou nenhum para retornar todos os objetos.

```
1 // Exemplo find
2 db.nome_do_banco.find()
```

Update, através da função `update()` atualiza os dados, recebendo dois parâmetros, sendo o primeiro a condição para achar o documento, e o segundo o novo documento.

```
1 // Exemplo update
2 db.nome_do_banco.update()
```

Remove, através da função `remove()` remove os dados, podendo passar um parâmetro para informar qual objeto remover ou nenhum para remover todos, caso esse seja seu objetivo, também é possível utilizar o comando `drop()`.

```
1 // Exemplo remove
2 db.nome_do_banco.remove()
3 // Exemplo drop
4 db.nome_do_banco.drop()
```

3.1.2 Express

Express é um *framework* web rápido, flexível e minimalista para Node.js inspirado no Sinatra, um *framework* para Ruby. Ele facilita o desenvolvimento de aplicações web e APIs, tanto pequenas quanto mais robustas, tornando fácil escalonar aplicações criadas com ele.

Com um conjunto de métodos utilitários HTTP e middlewares a seu dispor, criar uma API robusta utilizando Express é rápido e fácil.

Na MEAN Stack, o Express tem a responsabilidade de disponibilizar endpoints REST, que serão consumidos pela aplicação cliente.

3.1.2.1 Primeiros passos

Para criarmos um servidor com o Express, necessitamos primeiro do Node.js e o NPM instalados (veja como instalalos no capítulo ...) e então criar um arquivo com as seguintes linhas de código.

No terminal execute o comando `npm install express` dentro do diretório do seu projeto. Desta maneira conseguimos importar o express para nosso código. E então é só criar um arquivo `app.js` com o código abaixo.

```
1  import express from 'express';
2
3  let app = express();
4  const PORT = 3000;
5
6  app.get('/', (req, res) => {
7     res.send('Hello World');
8  });
9
10 app.listen(PORT);
11 console.log(`Servidor disponivel na porta ${PORT}`);
```

3.1.2.2 Express Generator

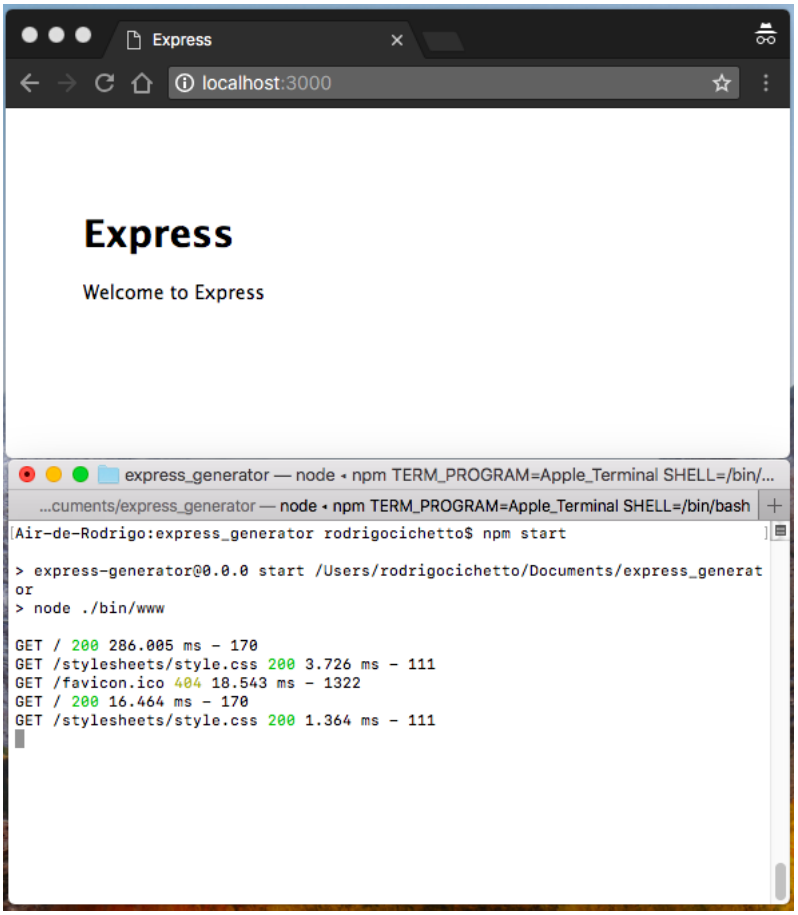
Com apenas o comando `express nome_da_app` do Express Generator ele já estrutura uma aplicação Express simples. Após a execução do comando ele irá criar um novo diretório com o nome da aplicação informado anteriormente, contendo arquivos e dependências necessárias.

Antes de iniciarmos o servidor Express devemos instalar as dependências através do comando `npm install` que irá baixar as dependências de acordo com o arquivo `package.lock` gerado anteriormente. Enfim é hora de iniciar a aplicação executando o comando `npm start` e acessá-la pelo endereço http://localhost:3000.

Essa página só é exibida porque no arquivo `routes/index.js` está configurado a rota padrão para que renderize o arquivo `views/index.jade` passando o título com o valor Express, sendo interpretado pela *template engine* JADE, configurada por padrão pelo Express Generator.

Para saber mais opções de geração de estruturas através do Express Generator, execute o comando `express -h`.

Figura 5 – Aplicação gerada pelo Express Generator



3.1.3 Angular

Angular é um framework JavaScript de código aberto, mantido pela Google. E tem como princípio "*One framework. Mobile and desktop*", que basicamente significa utilizarmos o Angular para criar nossa aplicação web e também aplicações para dispositivos móveis.

O *framework* utiliza TypeScript como linguagem, isso pode nos ajudar a manter melhor um projeto grande, já que a tipagem de variáveis nos força a sempre utilizar uma váriavel ou objeto do mesmo tipo, dando assim de certa forma uma garantia de que não teremos problemas ao acessar algum dado.

Assim como o React o novo Angular trabalha com componentes, que são trechos de códigos que definem elementos com aparência e comportamentos da interface. A ideia é que componentes sejam genéricos para que sejam utilizados e principalmente reutilizados em qualquer lugar da aplicação.

Na MEAN Stack, o Angular é responsável pela aplicação do usuário, ele cria interfaces dinâmicas sem manipulação de DOM, permite execução de lógica no client-side além de facilitar a troca de dados REST.

3.1.4 Angular CLI

Com o Angular CLI é possível criar projetos e estruturá-los a partir do comando `ng`. Por exemplo, para criar um novo projeto, basta executar o comando `ng new app-name`, assim não precisamos nos preocupar com a estrutura inicial do projeto e suas dependências cruciais para o funcionamento do Angular. Para iniciar o servidor com o Angular, basta executar o comando `ng serve` e abrir seu navegador no endereço http://localhost:4200.

Ainda é possível criar componentes, pipes, módulos, diretivas e serviços de forma rápida com o comando `ng generate`.

3.1.5 Node.js

O Node.js é uma plataforma de desenvolvimento construída sobre a linguagem JavaScript, por sua natureza assíncrona, exectuada pelo interpretador V8 criado pela Google e utilizado no Google Chrome, focado em migrar a linguagem JS para servidores. Foi criado pensando em um modelo não bloqueante para as operações de entrada e saída de dados (I/O - *Input and Output*).

Neste modelo, as operações de I/O não bloqueiam o atendimento aos outros clientes, ou seja, quando são feitas operações como uma leitura no disco ou consulta de banco de dados, as requisições de outros clientes vão sendo enfileiradas. Após o processamento ser finalizado e respondido ao primeiro cliente, o próximo cliente é atendido.

Na MEAN Stack, o Node é o core...

3.1.5.1 NPM

O *Node Package Manager*, mais conhecido como npm é o gerenciador de pacotes do Node.js. Com ele podemos fazer download de códigos, bibliotecas e *frameworks* que nossos projetos podem usar, ou até mesmo ferramentas que exerçam alguma função em nosso sistema operacional.

O arquivo `package.json` é responsável por gerenciar os pacotes cujo nosso projeto depende. Ele basicamente serve como uma lista para as dependências que nosso projeto necessita. Executando o comando `npm install` automaticamente todas as dependências listadas serão baixadas.

Isso também nos ajuda a padronizar as versões de nossas dependências, facilitando informar os requisitos mínimos e atualizações, garantindo assim que todos os desenvolvedores terão a mesma versão em suas máquinas. Outra vantagem é para que outro desenvolvedor tenha acesso a nosso código, basta enviar o código em si, sem se preocupar com as dependências, pois basta ele executar o comando `npm install` para fazer o download.

4 Conclusão