Utilizando MEAN Stack para o desenvolvimento de API REST

Adriano F. De Araújo1, Leonardo Sommariva2

1Departamento de Sistemas e Computação  
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil

2Departamento de Sistemas e Computação  
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil

flachadriano@gmail.com, lsommariva@gmail.com

**Abstract.** This paper presents the MEAN Stack framework, developed to make easier to build web applications, focusing on develop front-end and back-end with one programming language, the JavaScript. This framework a few others to do it, AngularJS é used to front-end, back-end use MongoDB, Express and NodeJS.

**Resumo.** Este artigo apresenta o framework MEAN Stack, desenvolvido para facilitar o desenvolvimento de aplicações web visando o desenvolvimento de front-end e back-end com apenas uma linguagem de programação, o JavaScript. Para isto é realizada a utilização de diferentes outros frameworks em conjunto, sendo o AngularJS para o front-end e para o back-end é utilizado MongoDB, Express e NodeJS.

# 1. MEAN Stack *framework*

Durante muitos anos JavaScript foi considerada por muitos como uma linguagem para amadores, porém sua arquitetura de desenvolvimento e potencial fez com que seus desenvolvedores mostrassem o poder desta linguagem. Com o surgimento do AJAX vislumbrou-se a possibilidade de transformar *sites* simples em aplicações *web*, o que inspirou o desenvolvimento de bibliotecas utilitárias, como jQuery e Prototype, para agilizar o desenvolvimento dessas aplicações. Google contribuiu para o contínuo crescimento da linguagem com o Chrome V8(HAVIV, 2014)*.* Este último, lançado em 2008, é uma máquina interpretadora de código JavaScript feita em C++, possibilitando o desenvolvimento de código JavaScript em processadores que suportem a linguagem C++ (GOOGLE, 2015).

Hoje em dia, *JavaScript* tornou-se ubíquo para o desenvolvimento de aplicações *web* *client-side*, porém para o desenvolvimento do servidor dessas aplicações *web*, muitas linguagens, *frameworks* e APIs entram em voga. Várias dessas opções atenderam as expectativas e estão decolando entre os desenvolvedores e empresas, enquanto outras ficaram obsoletas com o tempo. Em 2009, as pessoas já haviam se dado conta do potencial que JavaScript tinha como linguagem para o desenvolvimento de aplicações para o navegador, quando Ryan Dahl vislumbrou o potencial que esta linguagem tinha para o desenvolvimento de aplicações no servidor, então nascia o Node (BROWN, 2014).

Conforme Almeida (2015), O acrônimo MEAN foi cunhado em 2013 por Valeri Karpov do time do MongoDB para denotar o uso de uma stack completa para desenvolvimento de aplicações incluindo MongoDB, Express, AngularJS e Node.js.

A letra M do termo MEAN denota o MongoDB, um banco de dados orientado a documentos, que traz um novo conceito de armazenamento de dados, onde não há um esquema fixo definindo como cada dado armazenado deve ser (CHODOROW, 2013). A forma de armazenamento utilizada é muito similar ao JavaScript Object Notation (JSON) o que ajuda a realizar o armazenamento e reaver os dados, pois JSON é o formato comumente utilizado para prover e consumir APIs. Este formato de armazenamento realiza poucas validações em relação aos dados recebidos, tendo a aplicação a maior parte da responsabilidade de validar estes dados (ALMEIDA, 2015). Mantendo funcionalidades disponibilizadas por um banco relacional, como por exemplo índices e ordenação.

Express, criado em 2009, é responsável pela organização da aplicação no lado do servidor, utilizando a arquitetura MVC (ALMEIDA, 2015). Inspirado no *framework* Sinatra, desenvolvido em Ruby, que preza por desenvolvimento rápido, eficiente e manutenível. Seguindo esta ideia, Express disponibiliza uma camada mínima para o desenvolvimento da aplicação, porém, sua grande força está em permitir que sejam acoplados *middlewares*, que são responsáveis por executar alguma tarefa maior para a aplicação. Permitindo assim, que o framework evolua constantemente através de seus *middlewares*, assim como ocorre com Sinatra (BROW, 2014).

AngularJS implementa a letra A do MEAN Stack, que é responsável pelo desenvolvimento de aplicações no lado do cliente utilizando conceito de *Single Page Application* (SPA) onde a aplicação necessita ser carregada completamente apenas uma vez, sendo as demais chamadas realizada apenas para buscar partes necessárias para realizar a apresentação dos dados ao usuário (ALMEIDA, 2015). Este *framework* preza por desenvolver a aplicação de forma declarativa, extendendo as *tags* HyperText Markable Language (HTML), as tags adicionadas através deste framework são responsáveis por modularizar a aplicação (BRANAS, 2014). Este artigo focará no desenvolvimento da parte de servidor de uma aplicação *web*, sendo assim não será abordado o funcionamento do angular no MEAN Stack.

NodeJS é uma plataforma para aplicações JavaScript que roda sobre o Chrome V8. Nesta plataforma é possível utilizar bibliotecas desenvolvidas pela comunidade através do gerenciador de pacotes NPM. NodeJS seria o *core* da aplicação *web*.

# 2. MongoDB

MongoDB é um banco de dados orientado a documentos e não um banco relacional, desta forma o conceito de “linha” de uma tabela é substituído pelo modelo de “documento”, assim é possível prover maior facilidade e flexibilidade para alterações na modelagem dos dados, podendo armazenar um documento com uma lista de outros documentos em um mesmo registro, algo muito comum de ser observado no desenvolvimento orientado a objetos. Este banco se utiliza de esquemas não fixos onde chaves e valores não tem tamanhos ou tipos fixos, o que permite que facilmente seja realizada adição e remoção de campos, fazendo com que seja facilitado escalar este banco, sendo um de seus maiores destaques entre os bancos de dados atuais (CHODOROW, 2013).

Documento é um conjunto de chaves associadas a valores, representado de diferentes formas em diversas linguagens, em formas de mapas, *hash*s, dicionários, entre outros, em JSON são representados como objetos, conforme é apresentado no Quadro 1, neste documento pode ser verificado que há três chaves: nome, framework e versão. Como pode ser verificado na chave versão, este formato comporta diferentes tipos de dados, onde os dois primeiros são textos e o terceiro é numérico (CHODOROW, 2013).

|  |
| --- |
| {“nome” : “Adriano”, “framework” : “MEAN Stack”, “versao” : 1} |

Quadro 1. JSON

O formato de armazenamento utilizado pelo MongoDB funciona de forma muito similar ao apresentado no Quadro 1, sendo que devem ser observadas algumas regras e orientações sobre o funcionamento:

* Chaves não devem conter o caracter *\0* (caracter conhecido como *null*), pois é utilizado para identificar o fim do identificador de uma chave;
* Os caracteres “.” e “$” são comumente utilizados como chave para identificar propriedades especiais, tendo sua utilização restringida a alguns casos;
* As chaves são *case-sensitive*, ou seja, quando utilizado letras maiúscula como chave será diferente de se utilizar somente caracteres minúsculos;
* Não são aceitas chaves repetidas dentro de um documento;
* O *schema* não deve depender da ordem em que os campos são ordenados, pois o MongoDB pode ser realizar alterações em alguns casos.

Conjuntos de documentos que representem a modelagem de algo são agrupados em coleções. Assim como um documento está para uma linha no banco relacional, uma coleção está para uma tabela. Porém o MongoDB não controla o que é adicionado em uma coleção, sendo assim, é possível que haja dentro de uma mesma coleção dois documentos com chaves totalmente diferentes entre si. Apesar de o MongoDB prover esta liberdade para armazenamento de objetos de diferentes modelos dentro de uma coleção, não é aconselhável utilizar-se disto, pois conforme a massa de dados evolui serão observadas maiores dificuldades para identificar objetos dentro de uma coleção acarretando na necessidade de haver restrições muito genéricas para varrer uma grande massa de dados (CHODOROW, 2013).

Este banco de dados organiza as coleções de documentos em bases de dados, sendo que uma instância do MongoDB pode armazenar várias bases de dados. Sendo que cada uma das bases de dados tem permissões de acesso específicas e é salva em um arquivo diferente no disco. Bases de dados são comumente utilizadas para armazenar separadamente os dados de diferentes aplicações ou de usuários (CHODOROW, 2013).

# Referências

BROWN, Ethan. **Web Development with Node & Express:** Leveraging the JavaScript Stack. Sebastopol: O’Reilly Media, 2014. 306 p.

HAVIV, Amos Q. **MEAN Web Development:** Master real-time web application development using a mean combination of MongoDB, Express, AngularJS, and Node.js. Birmigham: Packt Publishing, 2014. 456 p.

ALMEIDA, Flávio. **MEAN:** Full stack JavaScript para aplicações web com MongoDB, Express, Angular e Node. São Paulo: Casa do Código, 2015. 377 p.

GOOGLE. **Chrome V8:** Google's high performance, open source, JavaScript engine. 2015. Disponível em: <https://developers.google.com/v8/>. Acesso em: 24 mar. 2016.

CHODOROW, Kristina. **MongoDB:** The Definitive Guide. 2. ed. Sebastopol: O’Reilly Media, 2013. 410 p.

BRANAS, Rodrigo. **AngularJS Essentials:** Design and construct reusable, maintainable, and modular web applications with AngularJS. Birmigham: Packt Publishing, 2014. 164 p.

NODEJS. **NodeJS.** 2016. Disponível em: <https://nodejs.org>. Acesso em: 24 mar. 2016.

\_\_\_\_\_\_. **NodeJS v5.9.1 Documentation.** 2016. Disponível em: <https://nodejs.org/api/modules.html>. Acesso em: 25 mar. 2016.