INTERFACE TECNOLÓGICA

MEAN STACK: uma solução para o desenvolvimento de aplicações Web

MEAN STACK: a solution for Web applications' development

Ariane Rodrigues Gomes de Oliveira – ro0drigues.ariane@gmail.com
Jederson Donizete Zuchi – jederson.zuchi@fatectq.edu.br
Faculdade de Tecnologia de Taquaritinga (FATEC) – São Paulo – Brasil

RESUMO

Este estudo tem como objetivo abordar a pilha de desenvolvimento MEAN e suas tecnologias, as mais populares do mercado de desenvolvimento, e seus benefícios bem como sua implementação em conjunto traz uma solução efetiva para alguns problemas recorrentes na área de desenvolvimento Web. Para este estudo é levantado um conteúdo teórico sobre *MEAN Stack* e suas tecnologias, MongoDB, ExpressJS, AngularJS e NodeJS. É mostrado como cada uma dessas tecnologias funciona e como se integram na pilha de desenvolvimento MEAN. É abordado como é feito o desenvolvimento Web utilizando MEAN e suas tecnologias e qual o padrão seguido. É abordado quais as vantagens de se utilizar MEAN e em quais cenários sua aplicação não se mostra eficiente ou pouco eficiente com base na proposta das tecnologias abordadas. Também é apresentado um estudo de caso que aplica o conhecimento abordado e a utilização da pilha MEAN para o desenvolvimento de melhoras em um aplicativo desenvolvido para Web. Por fim, são apresentadas as últimas considerações sobre o caso de estudo e como as melhorias feitas utilizando MEAN são significativas para o desempenho do aplicativo.

Palavras-chave: Desenvolvimento. MEAN. Tecnologias. Web. JavaScript.

ABSTRACT

This study aimed to approach MEAN stack development, its most popular technologies in the development world and its benefits as well as its implementation as a set of an effective solution to some problems in the area of Web development. For this study, a theoretical content about MEAN Stack and its technologies, such as MongoDB, ExpressJS, AngularJS and NodeJS, is gathered. It shows how each of the technologies works and how they integrate into the stack development. It is studied how the development is made through MEAN and its technologies and which development pattern was followed. This article presents the advantages of using MEAN stack and in which scenarios its application proves to be efficient or inefficient based on the proposal of the aforementioned technologies. A case study was also made in which the knowledge regarding the MEAN stack was applied in the development of improvements for a Web application. The final considerations describe the case study and how the MEAN stack improvements are meaningful for the application performance.

Keywords: Development. MEAN. Technologies. Web. JavaScript.

1 INTRODUÇÃO

Este trabalho traz um estudo sobre novas tecnologias para o desenvolvimento Web que são destaque no mercado e quais as melhorias proporcionadas por tal. *MEAN Stack* é uma nova pilha de desenvolvimento que envolve as tecnologias mais recentes e em destaque no mercado para o desenvolvimento de aplicações para Web. Esta pilha abrange quatro tecnologias emergentes que abordam soluções para alguns problemas recorrentes na área de desenvolvimento Web como complexidade do código tornando difícil a manutenção e integração e sobrecarga de serviços para os servidores Web.

Este estudo tem como objetivo trazer uma solução definitiva para os problemas encontrados utilizando a pilha de desenvolvimento MEAN. As tecnologias abordadas trazem consigo algumas soluções para os problemas citados e quando postas em conjunto fornecem uma solução definitiva e eficaz. A pesquisa se baseou nas necessidades encontradas e qual seria a melhor solução.

Para este estudo foi levantado um conteúdo teórico sobre as tecnologias abordadas e como cada uma se apresenta na pilha de desenvolvimento. Foi apresentado como funciona o desenvolvimento Web utilizando a pilha MEAN e suas tecnologias juntamente com suas vantagens e desvantagens. Por último foi apresentado um estudo de caso que aplica o conhecimento abordado.

2 REVISÃO BIBLIOGRÁFICA SOBRE MEAN STACK

O termo *Mean Stack* foi usado pela primeira vez por Valeri Karpov, engenheiro do time MongoDB, em Abril de 2013 em uma publicação no *blog* oficial do MongoDB (ALMEIDA, 2015). A expressão foi adotada pela sua equipe para trabalhar em uma maratona de programação utilizando MongoDB, ExpressJS, Angular JS e Node.js e se refere à junção das iniciais de todas as tecnologias utilizadas para o desenvolvimento da aplicação durante a maratona, conforme descrito em seu artigo: "*The MEAN Stack: MongoDB, ExpressJS, AngularJS and Node.js*".

Segundo Holmes (2015), "o *MEAN stack* é composto por quatro principais tecnologias [...]", são elas:

- MongoDB – o banco de dados.
- Express – o *framework Web*.
- Angular JS – o *framework front-end*.

- Node JS – o servidor *Web*.

2.1 Node JS

De acordo com o mesmo autor, o Node.js é uma plataforma de desenvolvimento que permite a criação de um servidor Web. Além disso,

> Node.js não é um servidor Web, nem mesmo uma linguagem. Ele contém uma biblioteca de servidor HTTP embutida, ou seja, não é preciso executar um programa de servidor Web separadamente como Apache [...]. Isso lhe traz um controle grande sobre como o seu servidor Web funciona. (HOLMES, 2015, p.7).

Portanto, o Node.js é uma biblioteca que disponibiliza recursos, porém não define como esses recursos serão utilizados. O Node não estabelece nenhuma regra para a sua utilização ou de configuração interna, ele somente disponibiliza recursos. Para tal configuração é utilizado o Express.

2.2 Express

O Express é descrito em seu *Website* como "um framework para aplicativo da Web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativo Web e móvel" [1]; isso significa que o Express é a estrutura para configurar o servidor Node.js. Utilizando este *framework*, é possível configurar e estabelecer regras de utilização para o seu servidor, ou seja, utilizando o Express, podemos configurar um servidor Node próprio. (HOLMES, 2015)

2.3 Mongo DB

Havendo um servidor onde é possível armazenar dados é preciso um banco de dados que torne viável o armazenamento. O banco de dados utilizado nesta pilha é o MongoDB, um banco de dados No-SQL baseado em documentos. Ou seja, é um banco de dados que armazena dados em forma de documentos ao invés de tabelas e não utiliza a Linguagem de Consulta Estruturada, do inglês *Structured Query Language* - SQL (HOLMES, 2015).

[1] Ver: http://expressjs.com/pt-br/.>

2.4 Angular JS

Por fim, é preciso montar toda a parte da aplicação que será exibida ao usuário final e responsável por coletar os dados, o *front-end*. Para o desenvolvimento do *front-end* é utilizado um *framework* chamado AngularJS. Holmes (2015) descreve Agular JS como "um framework JavaScript para trabalhar com dados direto do *front-end*", isso significa que Angular é uma biblioteca em JavaScript que permite a captura dos dados inseridos no navegador antes de serem enviados ao servidor. Esses dados são coletados e tratados para que sejam compatíveis com a estrutura estabelecida no *back-end*.

2.4 JavaScript

Todas essas tecnologias utilizadas em conjunto para o desenvolvimento de uma aplicação formam consigo um novo paradigma onde somente uma única linguagem de programação é utilizada para todo o desenvolvimento. A linguagem em questão é a JavaScript. Haviv (2014) reforça que o principal pilar para este paradigma é a linguagem JavaScript, pois é ela o centro de toda pilha.

Ainda de acordo com Haviv (2014), o desenvolvimento utilizando somente uma linguagem de programação é a principal vantagem, pois facilita a sincronização do código como um todo e permite melhor integração da equipe tornando o trabalho mais produtivo. Holmes explica que isso somente é possível utilizando Node.js porque tal tecnologia tornou viável a utilização da linguagem para o desenvolvimento do lado *back-end*, onde é feito o armazenamento dos dados e conexão com os servidores. Como JavaScript é uma linguagem interpretada, somente os navegadores Webs tinham essa funcionalidade e podiam interpretá-la até então.

3 A STACK

Na história do desenvolvimento Web temos dois campos bem distintos, mas que se complementam, o *front-end* e *back-end.* No início, esses dois campos surgiram com a necessidade de melhorar as páginas Web, dá-lhes mais funcionalidades e consequentemente aumentado a complexibilidade do código. Sendo assim, os desenvolvedores se especializavam

e em uma das duas categorias para atender melhor às necessidades de mercado. (HOLMES, 2015).

Com o passar do tempo, *frameworks* foram construídos para facilitar o desenvolvimento das páginas Web e abstraindo assim algumas complexidades do próprio desenvolvimento. Com isso, se tornou mais fácil desenvolver páginas Web e entender ambos os campos (HOLMES, 2015).

Com o passar do tempo, a forma de desenvolvimento Web foi mudando, e funcionalidades que antes eram destinadas ao desenvolvimento *back-end*, foram movidas para o desenvolvimento *front-end*. Isso se deu porque a lógica de carregamento das páginas passou a ser feita no cliente (navegador Web) e não mais no servidor como antes. Isso ajudou a diminuir o tempo de resposta do próprio servidor, diminuindo assim o custo. Com isso, a divisão entre *front* e *back* se tornou menos visível, pois mudou a forma como cada um trabalha mudando com isso as responsabilidades de cada. (HOLMES, 2015).

Atualmente, o desenvolvimento Web se baseia num modelo de três camadas que são chamadas de data, lógica e apresentação. Data seria a camada responsável pelo armazenamento de dados; a camada de lógica seria responsável por estabelecer as regras de tratamento dos dados e regra de negócio; e apresentação seria a camada responsável pela apresentação das páginas Web. Essas camadas também podem ser subdivididas em banco de dados, lógica do servidor, lógica do cliente e interface do cliente. (HAVIV, 2014).

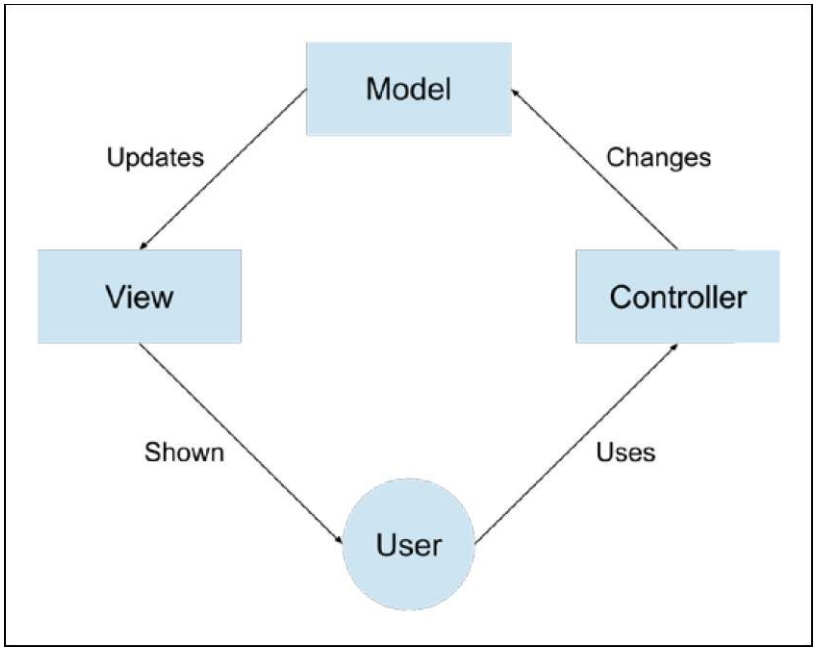
Há vários padrões de desenvolvimento que seguem esse modelo de três camadas, um dos mais populares é o modelo MVC (do inglês *Model-View-Controler*). (HAVIV, 2014).

O paradigma MVC distribui as três camadas em três objetos, que são *View*, *Controller* e *Model* (HAVIV, 2014):

- *View*: tem a tarefa de cuidar da interação do usuário, a parte visual.
- *Controller*: tem a tarefa de comandar a *View* e a *Model* de acordo com os eventos do sistema, a parte da lógica.
- *Model*: tem a tarefa de manipulação e armazenamento dos dados, a parte dos dados.

A figura demonstra como funciona o paradigma MVC.

Figura 1 – Arquitetura MVC de comunicação.



Fonte: Haviv (2014, p.8)

Todas as mudanças significativas durante a história do desenvolvimento Web circulam em volta da linguagem JavaScript. Foi com essa linguagem que as páginas em HTML se tornaram dinâmicas. JavaScript é uma linguagem interpretada que foi criada para a Web. Seu crescimento se iniciou nos anos 2000 e só aumentou com o surgimento de comunidades de desenvolvedores que se especializaram nessa linguagem. (HAVIV, 2014)

3.1 Vantagens de se utilizar *MEAN Stack*

As principais vantagens giram em torno da sintaxe, objetos, produtividade e integração da equipe.

3.1.1 Sintaxe e Objetos

De acordo com Haviv (2014), a principal vantagem em utilizar o *MEAN Stack* é a utilização de uma única linguagem por toda a pilha: JavaScript. Karpov utilizou essa pilha de desenvolvimento em uma maratona de programação e afirmou, em sua publicação, que há tanto ganho de performance no *software* quanto na produtividade do time. O autor explica que

> Com MongoDB, nós podemos armazenar nossos documentos em um formato como o de um JSON, escrever consultas em JSON no nosso servidor baseado em ExpressJS e NodeJS, e passar documentos JSON para o nosso *front-end* em Angular perfeitamente. (KARPOV, 2013)

Umas das vantagens citadas por Karpov (2013) em sua publicação é manter os dados armazenados no banco em um formato igual aos dados inseridos no *front-end*. O tipo de dado em questão é o JSON. Como MongoDB é um banco de dados que armazena dados e um formato baseado em JSON, se torna muito mais fácil armazenar os dados vindos nesse mesmo formato do *front-end*. Com isso, facilita-se a leitura do código do lado do servidor e das *queries* do banco de dados por que a sintaxe usada é a mesma e os objetos (dados) passados entre *front* e *back-end* são do mesmos tipo.

Tudo isso contribui para maior agilidade no processo de desenvolvimento de uma aplicação pois não há necessidade de abranger mais de uma linguagem para uma única aplicação o que consequentemente dificultaria o entendimento do código (KARPOV, 2013).

3.1.2 Produtividade e integração da equipe

Como já citado, a utilização de somente uma linguagem por todo o desenvolvimento da aplicação contribui para maior agilidade no processo e facilita o entendimento do código. Contribui também para melhor sincronização e integração da equipe, pois todo o planejamento e desenvolvimento envolvem somente uma linguagem.

Quando se tem uma equipe, é fundamental que haja sincronização entre todas as partes e para isso todos os integrantes precisam estar, se não completamente, muito bem integrados. Segundo Karpov (2016), “usando a mesma sintaxe e os mesmos objetos por todo o caminho o libera de ter que considerar vários conjuntos de melhores práticas de programação e reduz a barreira para a compreensão do seu código base.“.

Portanto, com prazos curtos como de uma maratona, é muito importante optar por soluções que facilitem o desenvolvimento e a integração da aplicação.

3.2 Desvantagens em utilizar MEAN

A seguir são apresentadas as algumas desvantagens envolvendo *MEAN stack* em relação a cenários específicos.

3.2.1 *Single Page Application* (SPA)

O termo *single page application* significa “aplicação de página única”. É uma metodologia empregada no desenvolvimento de páginas Web onde o carregamento da página é feito somente uma única vez, quando a página é carregada pela primeira vez. Qualquer alteração feita na página não exige um novo carregamento, porque o mecanismo por traz reproduz o funcionamento da arquitetura MVC, onde são feitas somente requisições de dados ao servidor e a página e partes dela são construídas do lado do cliente. Portanto, uma aplicação SPA é uma aplicação MVC.

A princípio pode ser muito atrativo desenvolver utilizando SPA, Holmes (2015) compara esse tipo de aplicação a um carro conversível. Ambos são lindos, atraentes, potentes, rápidos e muito eficientes, mas nem sempre eles são adequados. Holmes complementa fazendo uma comparação entre diferentes situações: quando se está sozinho dirigindo pela costa um conversível é a melhor escolha, mas quando é preciso sair com a família de férias o carro esportivo não é apropriado. O mesmo se emprega a páginas SPAs, nem sempre esse tipo de desenvolvimento é o melhor. As circunstancias do projeto precisam ser avaliadas para identificar se todas as páginas precisam ser carregadas completamente para decidir pelo desenvolvimento ou não de SPAs.

3.2.1.1 Rastreamento e Index

Segundo Holmes (2015), páginas SPAs são difíceis de rastrear e indexar. Geralmente os motores de busca olham o conteúdo em HTML da página, mas não executam o JavaScript, ou seja, o conteúdo em HTML que é carregado depois de executar o JavaScript não é analisado, somente o conteúdo do HTML estático. Os motores de buscam que executam o JavaScript não fazem um rastreamento tão eficiente com o conteúdo executado pelo JavaScript em comparação ao rastreamento feito no HTML estático.

Portanto, SPAs não são recomendadas para projetos que exigem páginas rastreáveis (HOLMES, 2015).

3.2.1.2 Análise de histórico

Holmes (2015) acrescenta que como SPAs não recarregam completamente uma página, ferramentas como *Google Analises* e o histórico do navegador não conseguem rastrear

o que se passa com o site. O *Google Analises* e o histórico do navegador dependem do carregamento completo da página para executarem suas funções, mas SPAs não funcionam desta forma. Portanto, se for preciso que as páginas do seu projeto sejam completamente carregadas, SPAs não se encaixam.

4 ESTUDO DE CASO

Com a finalidade de aplicar o conteúdo teórico apresentado, o estudo de caso a seguir demonstra as melhorias feitas em uma aplicação existente utilizando a pilha de desenvolvimento MEAN. Este estudo foi realizado por Adrian Shiokawa Alvarez (2015) para o trabalho de conclusão do curso de Computação do Instituto Tecnológico de Aeronáutica em São José dos Campos. Seu intuito é de comparar as formas tradicionais de desenvolvimento com o desenvolvimento utilizando *MEAN Stack* (ALVAREZ, 2015).

Seu estudo foi motivado pela busca de uma solução para o problema da escalabilidade no desenvolvimento Web. Para solucionar o problema da escalabilidade, Alvarez sugere a utilização das tecnologias que envolvem a pilha MEAN.

4.1 Cenário atual

Para seu estudo, Alvarez (2015) utilizou o aplicativo COLAB - Atividade Colaborativa. De acordo com o autor, este aplicativo foi desenvolvido como protótipo para a dissertação de mestrado da aluna Luma Maia Ferreira. O aplicativo foi desenvolvido somente para plataforma *Android*. Ainda segundo Alvarez, o desenvolvimento *back-end* foi feito utilizando a plataforma PaaS (do inglês, *Platform as a Service*, “Plataforma como serviço”) que consiste em auxiliar no desenvolvimento e implementação do código, ou seja, o desenvolvimento *back-end* da aplicação foi feito utilizando esta plataforma de serviços. Serviços esses que auxiliam no desenvolvimento abstraindo a responsabilidade de criá-los do desenvolvedor tornando o trabalho mais rápido.

4.2 O problema

A plataforma de serviço utilizada para o desenvolvimento *back-end* da aplicação proporciona agilidade para o desenvolvimento, mas a sua abstração não permite que o

desenvolvedor a customize. Ou seja, os serviços consumidos pela aplicação não são customizáveis, caso seja necessário (ALVAREZ, 2015).

4.3 Solução

Para solucionar o problema, Alvarez desenvolveu uma nova estrutura para o *back-end* utilizando NodeJS, ExpressJS e MongoDB. Seu desenvolvimento abrange as plataformas *Android* e iOS. Alvarez desenvolveu uma nova estrutura para o servidor implementando uma nova lógica, roteamento e acesso ao baco de dados, o que não era possível utilizando o método antigo. Alvarez afirma que umas das principais vantagens desse novo método é a criação de API Rest, pois seu acesso é feito de maneira simples, uma requisição HTTP, e pode ser feito por qualquer navegador cliente que possa interpretar os dados enviados. (ALVAREZ, 2015).

4.4 Análise dos resultados

Foi observado que para o desenvolvimento *front-end* foram utilizados aplicativos nativos das plataformas *Android* e iOS e não AngularJS. De acordo com Holmes (2015), MEAN é uma pilha muito flexível e essa flexibilidade se dá pelo uso da linguagem JavaScript, pois qualquer *framework front-end* em JavaScript pode ser encaixado na pilha. Tornando possível a utilização de outra plataforma *front-end* em JavaScript.

Outro ponto a ser observado é que Alvarez (2015) construiu uma nova lógica e um roteamento para o servidor para nova a versão da aplicação. Isso foi possível utilizando ExpressJS que permite a customização de *templates*, roteamento e gerenciamento de sessões. A utilização do ExpressJS em conjunto com as outras tecnologias integrantes do MEAN permitem que a solução desenvolvida seja independente da plataforma. Alvarez valida esta afirmação com a implementação da versão para iOS. Com a criação de um servidor Web em conjunto com a API Rest tornou a implementação viável, pois qualquer requisição HTTP feita para o servidor Web criado consegue se comunicar (HOLMES, 2015).

Alvarez (2015) acrescenta, “Ainda que seja totalmente em JavaScript, pode-se apontar como desvantagem a curva de aprendizado das tecnologias”. Para ele, o uso do paradigma de orientação a evento e código assíncrono da linguagem requerem uma forma diferente de pensar e por isso muitas funcionalidades não foram desenvolvidas. Isso demandaria mais tempo (ALVAREZ, 2015).

5 CONCLUSÃO

MEAN Stack surgiu como a solução de um problema: desenvolver e integrar diferentes partes de uma aplicação de forma rápida para uma maratona de programação. Essas competições são estimuladas tanto por instituições de educação quanto por empresas a fim de trazer o melhor de seus competidores e elucidar soluções para os mais variados problemas. Essas soluções sempre são valorizadas e postas em prática em projetos futuros, não seria diferente com o MEAN.

A utilização da linguagem JavaScript é o principal ponto para a utilização dessa pilha, pois possibilitou o surgimento da pilha. A integração e desenvolvimento de todos os componentes flui de forma mais rápida e fácil por causa da linguagem. Por este motivo essa pilha se mostra muito atrativa, mas há casos em que sua implementação não se mostre tão vantajosa, exigindo uma análise cuidadosa sobre o projeto.

Os seus pontos de desvantagem são facilmente contornados, uma vez que a pilha se mostra muito versátil a mudanças, justamente pela utilização da linguagem JavaScript. A linguagem é amplamente aceita e utilizada pela sua versatilidade. Mas a utilização da pilha tem que ser analisada com cautela, pois diferentes situações possuem diferentes soluções.

A solução encontrada para o estudo de caso citado foi desenvolver melhorias para uma aplicação já criada, foi identificado que as melhorias poderiam ser feitas utilizando a pilha MEAN. Nota-se que o desenvolvimento em ambas as plataformas foi possível utilizando esta pilha, pois a estrutura desenvolvida para o servidor foi determinante para esta implementação.

Conclui-se que a utilização da pilha MEAN traz uma vantagem muito grande por se utilizar da linguagem JavaScript. E sua aplicação no desenvolvimento Web traz significativas melhorias tanto para o desenvolvimento em si quanto para o projeto e para os desenvolvedores envolvidos, mas mostra-se necessário um estudo e conhecimento prévio da linguagem para se atingir todos os objetivos visados.

REFERÊNCIAS

ALMEIDA, F. **MEAN Full stack JavaScript para aplicações Web com MongoDB, Express, Angular e Node**. 1 ed. São Paulo: Casa do Código, 2015.

ALVAREZ, A. S. **Desenvolvimento de aplicação Web utilizando a *solution stack* MEAN**. 2015. 32f. Trabalho de Conclusão de Curso. (Graduação) - Instituto Tecnológico de Aeronáutica, São José dos Campos. Disponível em:

<http://www.bdita.bibl.ita.br/TGsDigitais/lista_resumo.php?num_tg=69679.> Acesso em: 18 fev. 2017, 15:11

EXPRESS. **Guia, Referencia da API, Tópicos Avançados, Recursos, etc**. Disponível em: <http://expressjs.com/pt-br/.> Acesso em: 25 jan. 2017, 20:01

HAVIV, A. Q. **MEAN Web Development: Master real-time Web application development usin a mean combination of MongoDB, Express, AngularJS, and Node.js**. 1 ed. Birmingham: Pack Publiching, 2014.

HOLMES, S. **Getting MEAN with Mongo, Express, Angular, and Node**. ed. New York: Manning Publications Co., 2015.

KARPOV, V. **The MEAN Stack: MongoDB, ExpressJS, AngularJS and Node.js**. Disponível em: <http://blog.mongodb.org/post/49262866911/the-mean-stack-mongodb-expressjs-angularjs-and.MongoDB>. Acesso em: 29 set. 2016, 14:36