INSTITUTO FEDERAL SUL-RIO-GRANDENSE
CURSO SUPERIOR EM SISTEMAS PARA INTERNET

PLANO DE ESTUDO E PESQUISA

Harmonic: O Próximo Gerador de Sites Estáticos

Fabrício da Silva Matté

Prof. Sérgio Luis Rodrigues
Orientador

Pelotas, agosto de 2016

SUMÁRIO

RESUMO

O projeto Harmonic tem por objetivo desenvolver um gerador de sites estáticos, utilizando como base a plataforma Node.js juntamente com os recursos mais recentes da linguagem JavaScript que foram especificados no padrão ECMAScript 2015 (TC39, 2015), o qual foi finalizado e oficializado como padrão da linguagem JavaScript em Junho de 2015. O compilador Babel é utilizado para atingir este objetivo, o qual transforma código que utiliza recursos de especificações recentes e futuras do JavaScript em código que pode ser executado nos motores JavaScript atuais.

O Harmonic roda sobre a plataforma Node.js, que é, de forma resumida, um motor JavaScript combinado com servidor Web, que permite execução do mesmo código JavaScript em todas as principais plataformas (Windows, Linux, Mac), sem possuir as restrições de segurança comumente encontradas no ambiente de navegadores. Ou seja, o código JavaScript executado pelo Node.js tem acesso completo ao sistema de arquivos e rede da máquina hospedeira, e esta é uma das principais capacidades do Node.js das quais o Harmonic utiliza para gerar sites estáticos.

Palavras-chave: Gerador de Sites Estáticos, Node.js, ECMAScript 2015.

TITLE: “HARMONIC: THE NEXT STATIC SITE GENERATOR”

ABSTRACT

The Harmonic project aims to develop a static site generator, using the Node.js platform as its base together with the most recent JavaScript features that have been specified in the ECMAScript 2015 standard (TC39, 2015), which was completed and officially published as the JavaScript language standard in June 2015. The Babel compiler is utilized to achieve this goal, which transforms code that uses features from the most recent and future JavaScript specifications into code that can be run in the current JavaScript engines.

Harmonic runs on the Node.js platform, which is, basically, a JavaScript engine combined with a Web server, which enables the same JavaScript code to run in all the major platforms (Windows, Linux, Mac), having none of the security restrictions often found in the browser environment. That is, the JavaScript code run by Node.js has full access to the host machine’s file system and network, and this is one of the main Node.js capabilities of which Harmonic uses to generate static sites.

Keywords: Static Site Generator, Node.js, ECMAScript 2015.

1 INTRODUÇÃO

O projeto Harmonic trata-se de um gerador de sites estáticos. Ou seja, é um programa capaz de gerar uma estrutura de pastas e *arquivos fonte* onde o usuário pode criar e gerenciar o conteúdo de seu site, além de poder instalar, criar ou personalizar *temas*, que são grupos de arquivos de *template*.

O Harmonic pode ser instalado de forma gratuita pelo gerenciador de pacotes *npm* (NPM, 2016). Os temas do Harmonic geralmente são disponibilizados através de pacotes também distribuídos pelo npm, mas o usuário também possui a opção de criar temas privados sem a necessidade de publicá-los no npm, assim como não compartilhá-los com o público.

Outra característica importante do projeto Harmonic é que todo seu desenvolvimento se dá de forma aberta, tendo não apenas todo seu *código fonte* acessível publicamente em um repositório hospedado no serviço do GitHub, mas também todas discussões, reportagem de problemas, tomadas de decisões e governança do projeto também se dá de forma aberta no próprio *issue tracking system* do GitHub, que é utilizado como uma espécie de sistema de gerenciamento do projeto.

O projeto Harmonic está sendo escrito sobre a plataforma *Node.js*, que possibilita a execução de programas escritos na linguagem de programação JavaScript nos principais Sistemas Operacionais (Windows, OS X e várias distribuições de Linux). Desta forma, todo projeto é escrito em uma linguagem de programação fácil de escrever e contribuir, além de não possuir limitações comumente encontradas no ambiente de um *browser*.

O foco principal deste projeto está na ergonomia do usuário e simplicidade de ser utilizado. O usuário está sempre em primeiro plano. Recursos como recompilar arquivos automaticamente ao serem modificados assim como recarregar o site em desenvolvimento automaticamente quando há mudanças, além da excelente performance da ferramenta e a simplicidade da interface de linha de comando, facilitam a vida do usuário e o deixam mais produtivo.

Um dos objetivos complementares deste projeto é aprender, explorar e demonstrar os novos recursos oferecidos pela linguagem JavaScript, como aqueles introduzidos nas novas edições da especificação ECMAScript, assim como nas propostas que estão em andamento no processo de padronização do TC39 (Technical Committee 39, grupo responsável pela evolução da linguagem JavaScript) da associação *ECMA International*.

Na próxima seção veremos um pouco mais sobre o tema e metodologia do projeto.

1.1 Justificativa

1.2 Objetivos

O objetivo geral deste trabalho é explorar a área de geradores de sites estáticos, desenvolvendo um novo gerador de sites estáticos com recursos inovadores e fazendo uso das últimas tecnologias relacionadas à linguagem JavaScript.

Os objetivos específicos são:

- estudar e aprofundar a metodologia de criação de sites estáticos;

- estudar os recursos das próxima(s) versão(ões) do JavaScript (ECMAScript 2015 e além);
- estudar a plataforma Node.js;
- criar um gerador de sites estáticos com recursos que os demais ainda não possuem;

2 FUNDAMENTAÇÃO TEÓRICA

A proposta dos sites estáticos é um tema bem popular atualmente, tendo mais de 435 geradores de sites estáticos já existentes. (STATIC SITE GENERATORS, 2016)

Ao contrário dos sites dinâmicos, que requisitam banco de dados e executam lógica no servidor para cada requisição, todo o conteúdo de um site estático é gerado instantaneamente antes de qualquer acesso por parte de usuários, assim podendo ser hospedado em um servidor que não necessita de banco de dados nem suporte a nenhuma linguagem de programação.

Além disto, como diz David Walsh, sites estáticos são muito mais rápidos que os dinâmicos, pois todo processamento de conteúdo já foi realizado no momento da geração do site estático e não a cada requisição como nos sites dinâmicos. Outro ponto é a segurança, sites estáticos são extremamente mais seguros que os dinâmicos pois não possuem lógica de programação no servidor, o que os torna a prova de falhas de segurança na programação do lado do servidor. (AN INTRODUCTION TO STATIC SITE GENERATORS, 2015)

Praticamente todos geradores de sites suportam a escrita de conteúdo através da linguagem de marcação Markdown, que é basicamente uma versão simplificada, mais fácil de escrever e ler do que a linguagem de marcação HTML. Markdown é uma ferramenta de conversão de texto para HTML para escritores Web (GRUBER, 2004). O projeto Harmonic também suporta a linguagem de marcação Markdown e o converte para HTML na momento da geração do conteúdo.

O projeto Harmonic é desenvolvido sobre a plataforma Node.js, que suporta todos os principais sistemas operacionais (OS X, Linux, Solaris, FreeBSD, OpenBSD, Windows, webOS, NonStop OS) nos permitindo executar o mesmo código em todas estas plataformas. A plataforma Node.js executa código JavaScript, que é uma linguagem de programação leve e bem simplificada (AL., 2016), assim facilitando no desenvolvimento. O Node.js também possui mais de 250.000 pacotes públicos publicados no registro de pacotes mais popular, npm (NPM, 2016), o que torna o desenvolvimento muito mais ágil permitindo reutilizar "blocos de construção"para construir sistemas maiores. (SORHUS, 2015)

3 METODOLOGIA DE PESQUISA

sub-capítulo: experiência própria
sub-capítulo: Sistemas semelhantes
Vejamos uma comparação entre os Sistemas Semelhantes.

4 MODELAGEM DO SISTEMA

Após especificar o que o sistema deveria possuir, passou-se à etapa de criação dos diagramas para identificar como deve ser o processo de comunicação entre os vários componentes do sistema e a utilização do UML facilitou a modelagem do software.

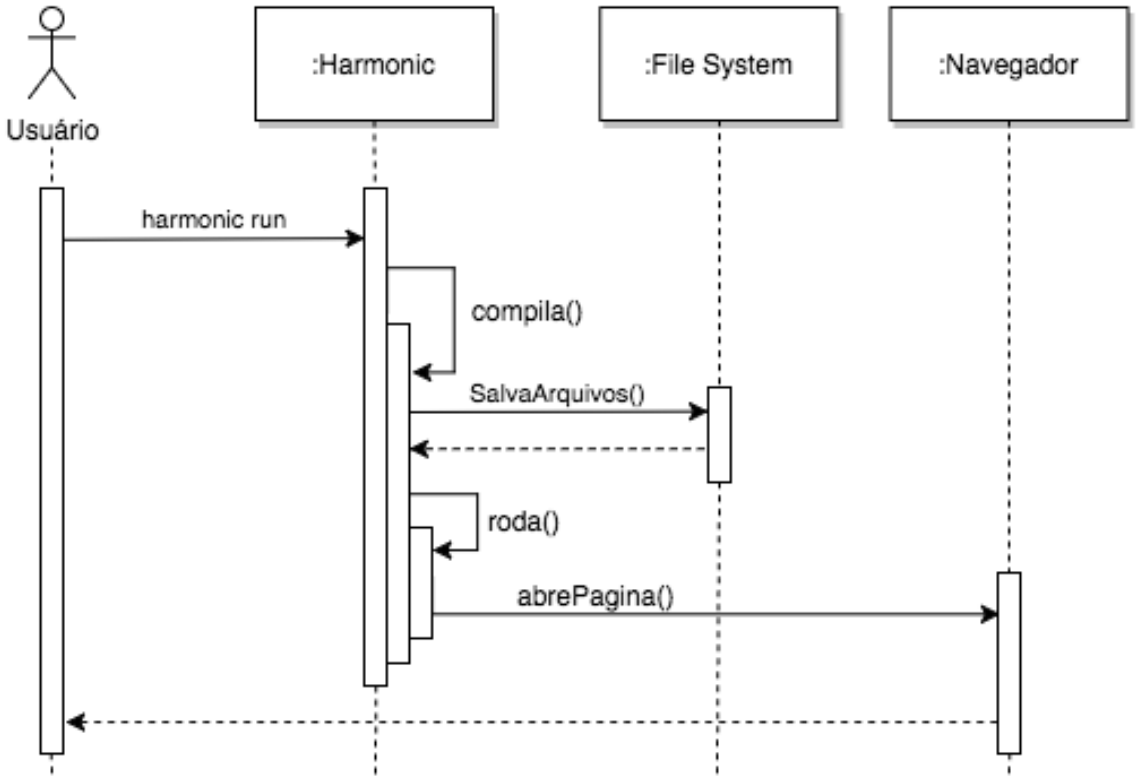
UML é uma linguagem de modelagem que auxilia os desenvolvedores na montagem dos requisitos e do comportamento dos processos no sistema, também atua na descoberta de possíveis necessidades físicas que possam surgir na implementação de uma determinada ferramenta. (GUEDES, 2011)

UML é uma metodologia que disponibiliza diversas maneiras para analisar uma determinada questão e neste projeto será utilizado uma abordagem orientada a objetos, que auxilia na reutilização de métodos ou atributos, melhorando o desenvolvimento e a manutenção do sistema. Os itens a seguir apresentam diagramas para facilitar o entendimento do leitor.

4.1 Diagrama de Sequência

Foram criados diagramas de sequência pois estes ajudam a planejar o fluxo de informações e operações a serem realizadas pelas rotinas do sistema. O diagrama de sequência da execução do site estático, que é uma das principais funcionalidades do Harmonic, pode ser conferido na figura 4.1.

Figura 4.1: Diagrama de sequência da execução de um site estático do Harmonic.

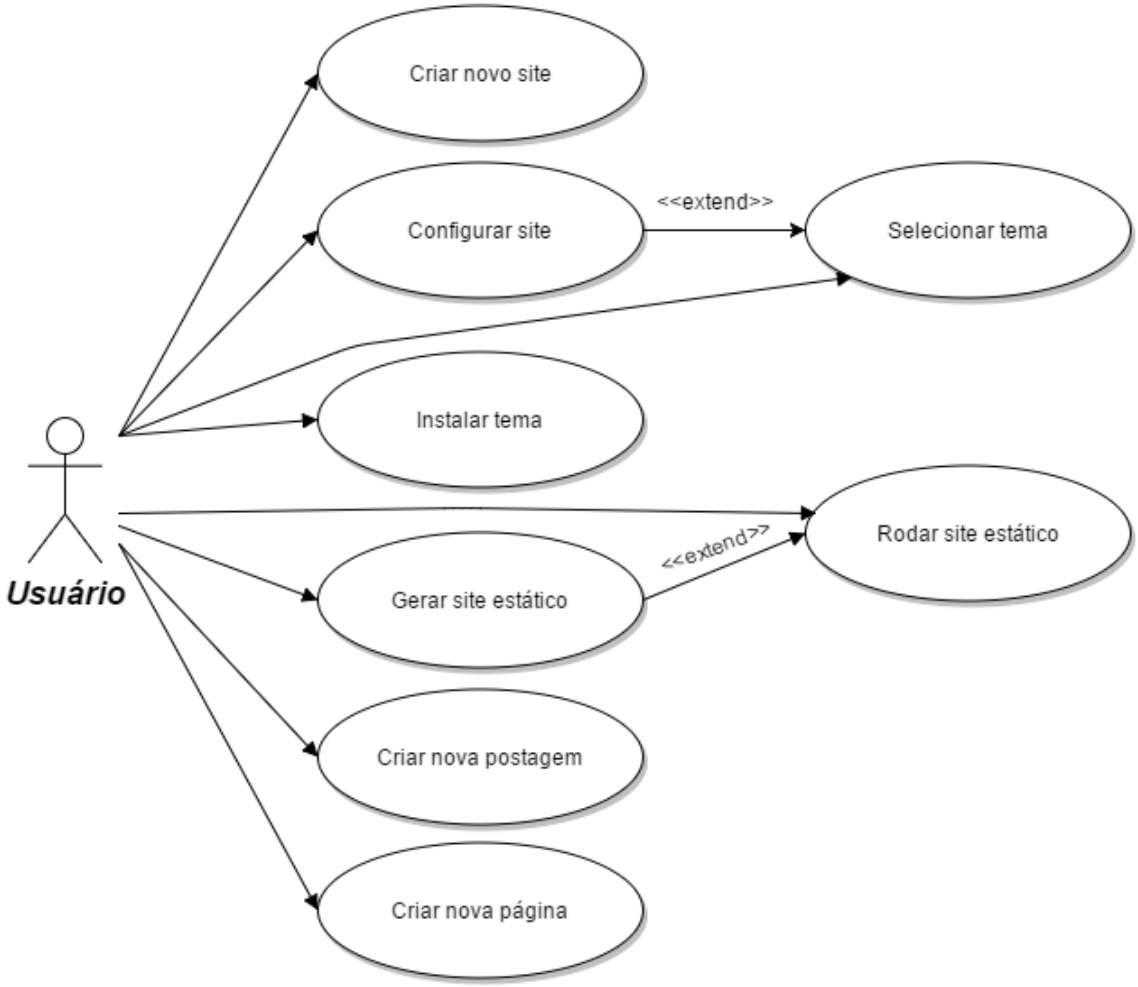


Fonte: Elaborado pelo autor.

4.2 Diagrama de Casos de Uso

Foi criado um diagrama de casos de uso pois este facilita mapear as funcionalidades que o sistema deve desempenhar assim como as relações entre elas. O diagrama de casos de uso do Harmonic pode ser conferido na figura 4.2.

Figura 4.2: Diagrama de casos de uso do Harmonic.



Fonte: Elaborado pelo autor.

5 TECNOLOGIAS E FERRAMENTAS UTILIZADAS

5.1 JavaScript

JavaScript é uma linguagem de programação orientada a objetos multiplataforma. Dentro de um ambiente hospedeiro (por exemplo, um navegador web), JavaScript pode ser conectado aos objetos de seu ambiente para prover controle sobre eles. O padrão oficial da linguagem JavaScript é o ECMAScript. (AL., 2016)

O JavaScript obteve uma evolução significativa nos últimos anos, tornando-se assim uma das linguagens de propósito geral mais utilizadas no mundo. É mais conhecida como a linguagem incorporada aos navegadores Web, mas também houve um grande crescimento em sua adoção por parte de servidores e aplicações embarcadas. (TC39, 2015)

O projeto Harmonic escolheu esta linguagem de programação devido a sua ampla aceitação no mercado e rápida evolução, além de esta linguagem ser relativamente simples e fácil de trabalhar, contendo centenas de milhares de pacotes comunitários que facilitam no desenvolvimento de praticamente qualquer sistema.

5.2 Node.js

Node.js é uma plataforma de código fonte aberto que permite construir aplicações em rede utilizando a linguagem JavaScript. Node.js é construído em cima do V8, uma máquina virtual JavaScript moderna que também é utilizada pelo navagedor Web Google Chrome. (TEIXEIRA, 2013)

Node.js é um *runtime* JavaScript assíncrono orientado a eventos. (ABOUT | NODE.JS, 2016) Isto que dizer que o Node.js pode ser considerado uma plataforma que combina um motor JavaScript com acesso a recursos do sistema operacional hospedeiro.

A plataforma Node.js roda diretamente sobre o sistema operacional, com isto é possível executar código JavaScript completamente fora do ambiente de um navegador Web. Assim sendo, o Node.js não interage com uma página da Web diretamente, mas sim com os recursos do computador hospedeiro. Por exemplo, o Node.js pode receber e realizar requisições HTTP, ler e escrever no sistema de arquivos, e se comunicar com outros processos, *sockets* e *drivers* do sistema operacional. (WEN, 2013)

Todo o projeto Harmonic é construído sobre a plataforma Node.js, que realiza toda lógica de negócios e leitura e escritas ao sistema de arquivos do usuário.

5.3 npm

O npm é o gerenciador de pacotes padrão da plataforma Node.js, contando com mais de 250.000 pacotes publicados (NPM, 2016) e provendo mais de um bilhão de instalações de pacotes semanalmente (VOSS, 2016).

O projeto Harmonic escolheu o npm como sua plataforma de distribuição e geren-

ciador de dependências devido ao seu fácil acesso, altíssima resiliência e ubiquidade na comunidade Node.js.

5.4 JSON

JSON (JavaScript Object Notation) é um formato de troca de informações eficiente. É fácil de ler e escrever manualmente. É fácil de analisar e gerar por computadores, assim como ler e escrever por desenvolvedores. (CROCKFORD, 2013)

O projeto Harmonic escolheu o formato JSON para o armazenamento de configurações, pois este formato facilita a operação dos dados tanto pelo sistema quanto pelo usuário final.

5.5 Markdown

5.6 gulp

5.7 Babel

5.8 BrowserSync

6 DESCRIÇÃO DO SISTEMA

6.1 Estrutura do projeto

O Harmonic divide-se em vários módulos, sendo eles: criação de um novo site estático, configuração do site estático, criação de *posts* e páginas, geração do site estático e execução do site estático, além do módulo de ajuda.

Assim como a maioria dos pacotes, bibliotecas e ferramentas escritas sobre a plataforma Node.js, o Harmonic está publicado no registro de pacotes do npm e pode ser instalado através do gerenciador de pacotes que faz parte da instalação padrão do Node.js, ou seja, através da ferramenta de linha de comando do npm.

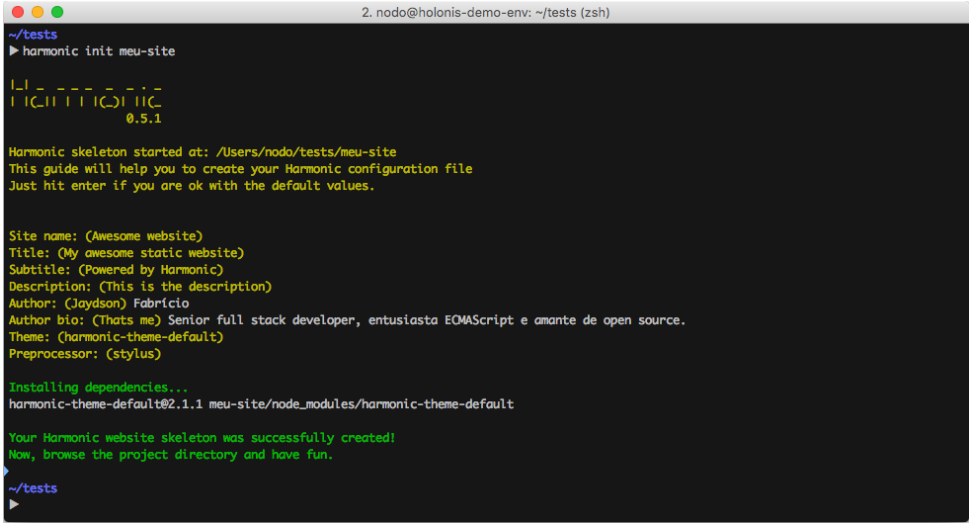
E, similarmente a outros geradores de sites estáticos, o Harmonic conta uma ferramenta de linha de comando que pode ser utilizada para criar um novo site estático, adicionar conteúdo ao mesmo, compilar o site estático e visualizá-lo no navegador.

6.2 Módulo de criação de um novo site estático

O Harmonic permite a criação de novos sites estáticos através do comando `harmonic init`. Este comando gera uma estrutura padrão de pastas e arquivos, onde são guardados todos arquivos fontes do site estático.

O comando `harmonic init` realiza uma série de questionamentos sobre as configurações e personalizações que o site deve possuir (conforme pode ser conferido na figura 6.1), então escreve estas configurações no arquivo de configuração do site estático `harmonic.json`. O usuário pode futuramente modificar estas configurações utilizando o comando `harmonic config` ou editando o arquivo de configuração `harmonic.json` em qualquer editor de texto, já que o arquivo de configuração nada mais é do que um arquivo de texto no formato *JSON*.

Figura 6.1: Exemplo do comando "harmonic init".



Fonte: Elaborado pelo autor.

6.3 Configurações disponíveis

A FAZER

6.4 Criação de novos *posts*

O usuário pode criar novos *posts* utilizando o comando `harmonic new_post "<TÍTULO DO POST>"`, que criará um arquivo *Markdown* dentro da pasta fonte de *posts* de cada linguagem que o site estático suporta.

6.4.1 Meta-informações e configurações de *posts*

Cada arquivo fonte de *post* possui um cabeçalho de meta-informações, no formato de um comentário HTML contendo pares de chaves e valores, como pode ser visto na figura 6.2.

Figura 6.2: Exemplo de cabeçalho de meta-informações de um *post* do Harmonic.

```
<!--
layout: post
title: JavaScript iterables and iterators
date: 2015-09-15T04:06:02.428Z
comments: true
published: true
keywords: iterables, iterators, ES2015
description: Understanding JavaScript ES2015 Iterables and
   Iterators
categories: iterables, iterators, ES2015, articles
authorName: UltCombo
authorLink: https://twitter.com/Ult_Combo
authorDescription: Full Stack developer, ECMAScript
   enthusiast, open source lover.
authorPicture:
   https://s.gravatar.com/avatar/326fba1c2980ce0073f6b212acf71ea0
-->
```

Fonte: Elaborado pelo autor.

- **`layout`**: nome do arquivo HTML do tema selecionado a ser utilizado na renderização deste *post*. O valor padrão é `post`.
- **`title`**: título do *post*.
- **`date`**: data de publicação deste *post*. Pode ser utilizado para agendar uma publicação futura.
- **`comments`**: Um valor booleano indicando se o *post* deve permitir comentários ou não. Note que nem todos temas suportam comentários. O valor padrão é `true`.
- **`published`**: Um valor booleano indicando se o *post* está publicado ou não. Pode ser utilizado para despublicar ou ocultar o *post*. O valor padrão é `true`.

- **`keywords`**: Palavras-chave descrevendo o *post*.
- **`description`**: Descrição resumida do *post*.
- **`categories`**: Lista de categorias a qual o *post* pertence, separadas por vírgulas. Você pode especificar qualquer nome de categoria, se a mesma não existir ela será criada.
- **`authorName`**: Nome do autor do *post*.
- **`authorDescription`**: Biografia resumida do autor do *post*.
- **`authorPicture`**: Endereço URL para a foto de perfil do autor do *post*.

6.5 Geração do site estático

Para consumir o conteúdo do site é necessário primeiro gerar o site estático, que é basicamente um processo de compilação. Para isso, pode-se utilizar o comando `harmonic build`, que gera o site estático a partir dos arquivos fontes (configurações, tema ativo e arquivos Markdown) do seu site Harmonic, assim gerando arquivos HTML, CSS e JS como saída.

Com estes arquivos, então, é possível abrir o arquivo `index.html` em seu navegador de preferência e navegar pelo site estático.

Note que qualquer edição ou mudança realizada nos arquivos fontes do site, incluindo edição de *posts* e alteração de configurações, requer uma nova compilação para que as mudanças sejam refletidas nos arquivos compilados.

6.6 Execução do site estático

O Harmonic também dispõe do comando `harmonic run`, que além de realizar o mesmo processo de compilação que o `harmonic build`, também inicia um servidor local para servir os arquivos estáticos e automaticamente abre o site estático no navegador padrão do sistema. Este é o método indicado para pre-visualização de seu site estático, pois os navegadores frequentemente bloqueiam vários recursos JavaScript ao acessar os arquivos estáticos diretamente do disco sem um servidor envolvido no processo, o que pode afetar a funcionalidade do site caso você abra os arquivos diretamente no navegador como descrito na seção anterior.

Além disso, o comando `harmonic run` também conta com um recurso de auto-geração de site e auto-recarregamento do navegador ("*live-reload*") sempre que ocorrerem alterações nos arquivos fontes do site, assim proporcionando uma experiência de criação de conteúdos otimizada e muito mais prática e eficiente.

Por padrão, o comando `harmonic run` inicia um servidor local na porta `9356`, e abre o site no navegador padrão do sistema automaticamente. É possível customizar estes recursos, passando a porta desejada como primeiro parâmetro do comando (exemplo: `harmonic run 8080`), assim como desabilitar a função de abrir o site automaticamente passando a *flag* `-no-open`.

7 CONSIDERAÇÕES FINAIS

7.1 Resultados obtidos e discussões

7.2 Trabalhos Futuros

7.2.1 Plugins

A FAZER

7.2.2 Melhorias no desenvolvimento de temas

A FAZER

REFERÊNCIAS

ABOUT | Node.js. Acesso em agosto de 2016, https://nodejs.org/en/about/.

AL., C. R. F. S. G. P. et. **Introduction - JavaScript**. Acesso em agosto de 2016, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript.

AN Introduction to Static Site Generators. Acesso em agosto de 2016, https://davidwalsh.name/introduction-static-site-generators.

CROCKFORD, D. **The JSON Data Interchange Format**. [S.l.: s.n.], 2013.

GRUBER, J. **Markdown**. Acesso em julho de 2016, https://daringfireball.net/projects/markdown/.

PRATES, R. (Ed.). **UML 2 - Uma Abordagem Prática - 2ª Edição**. Rua Luís Antônio dos Santos 110, 02460-000 – São Paulo, SP – Brasil: Novatec Editora Ltda., 2011.

NPM. **npm**. Acesso em julho de 2016, https://www.npmjs.com/.

SORHUS, S. **Micro-modules**. Acesso em agosto de 2016, https://github.com/sindresorhus/ama/issues/10#issuecomment-117766328.

STATIC Site Generators. Acesso em agosto de 2016, https://staticsitegenerators.net/.

TC39. **ECMAScript 2015 Language Specification - ECMA-262 6th Edition**. [S.l.: s.n.], 2015.

HINDLE, A. R. A. H. S. T. S. S. B. D. P. (Ed.). **Instant Node.js Starter**. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK: Packt Publishing, 2013. 3p. n.cap. 1.

VOSS, L. **The npm Blog - how many npm users are there?** Acesso em julho de 2016, http://blog.npmjs.org/post/143451680695/how-many-npm-users-are-there.

WEN, B. **6 things you should know about Node.js**. Acesso em agosto de 2016, http://www.javaworld.com/article/2079190/scripting-jvm-languages/6-things-you-should-know-about-node-js.html.

8 ASSINATURA