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, julho de 2016

SUMÁRIO

˜HAPTER*.1RESUMO 3

˜HAPTER*.11 INTRODUÇÃO 4

˜HAPTER*.12 FUNDAMENTAÇÃO TEÓRICA 5

˜HAPTER*.13 METODOLOGIA DE PESQUISA 6

˜HAPTER*.14 TECNOLOGIAS UTILIZADAS 7

˜HAPTER*.15 MODELAGEM DO SISTEMA 8

˜HAPTER*.16 ESTRUTURA DO PROJETO 9
6.1 Módulo de criação de um novo site estático 9
6.2 Configurações disponíveis 9
6.3 Criação de novos *posts* 9
6.3.1 Meta-informações e configurações de *posts* 9
6.4 Geração do site estático 10
6.5 Execução do site estático 11

˜ECTION.6.57 ESCOPO DO TRABALHO 12
7.1 Medicina Ubíqua: Principais Demandas 12

˜ECTION.7.18 OBJETIVO 13

˜ECTION.7.19 TRABALHOS FUTUROS 14
9.1 Plugins 14
9.2 Melhorias no desenvolvimento de temas 14

˜ECTION.9.2REFERÊNCIAS 15

˜ECTION.9.210 ASSINATURA 17

RESUMO

O projeto Harmonic tem por objetivo desenvolver um gerador de sites estáticos, utilizando como base a plataforma Node.js e os recursos mais recentes da linguagem JavaScript, recursos que estão sendo especificados no padrão ECMAScript 2015 (TC39, 2015), o qual foi finalizado e oficializado como padrão da linguagem JavaScript na metade de 2015. Para isto, é utilizado o compilador Babel, que transforma código que utiliza recursos de especificações 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.

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 nem 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.

2 FUNDAMENTAÇÃO TEÓRICA

A proposta dos sites estáticos é um tema bem popular atualmente, tendo 435 geradores de sites estáticos já existentes no momento da escrita. [https://staticsitegenerators.net/]

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 inteiramente em uma única execução de uma ferramenta de geração de sites estáticos, assim podendo ser hospedado em um servidor que não necessita de banco de dados nem suporte a nenhuma linguagem de programação. Além disto, 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. [https://davidwalsh.name/introduction-static-site-generators]

Praticamente todos geradores de sites suportam a escrita de conteúdo através da linguagem de marcação Markdown (GRUBER, 2004), que é basicamente uma linguagem mais fácil de escrever e ler do que HTML. Markdown é uma ferramenta de conversão de texto para HTML para escritores Web. 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 na linguagem de programação JavaScript, que é bem simples de escrever e portanto facilita bastante no desenvolvimento. O Node.js também possui mais de 250,000 pacotes públicos publicados no registro de pacotes mais popular, npm, o que torna o desenvolvimento muito mais ágil permitindo reutilizar ”blocos de construção”para construir sistemas maiores. [https://nodejs.org/en/about/]

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.

3 METODOLOGIA DE PESQUISA

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

4 TECNOLOGIAS UTILIZADAS

A FAZER

5 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 (GILLEANES, 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.

6 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.

6.1 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 perguntas sobre as configurações e personalizações que o site deve possuir, 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*.

6.2 Configurações disponíveis

A FAZER

6.3 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.

6.3.1 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. Por exemplo:

```
<!--
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: Fabrcio S. Matt
authorLink: https://twitter.com/Ult_Combo
authorDescription: ECMAScript enthusiast, open source addict
   and Web Platform lover.
authorPicture:
   https://s.gravatar.com/avatar/326fba1c2980ce0073f6b212acf71ea0
-->
```

- **`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*.

6.4 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.

6.5 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 ESCOPO DO TRABALHO

7.1 Medicina Ubíqua: Principais Demandas

8 OBJETIVO

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;

9 TRABALHOS FUTUROS

9.1 Plugins

A FAZER

9.2 Melhorias no desenvolvimento de temas

A FAZER

REFERÊNCIAS

BARDRAM J.; BOSSEN, C. Mobile Work - The Spatial Dimension of Collaboration at a Hospital. **Computer Supported Cooperative Work**, [S.l.], v.14, n.2, p.131–140, 2005.

BROWN I.; ADAMS, A. The ethical challenges of ubiquitous healthcare. **International Review of Information Ethics**, [S.l.], v.8, n.54-59, Dezembro 2007.

CFM, . **Resolu 1.629/2002 do Conselho Federal de Medicina**. Disponel em: <¡ http://www.arnaut.eti.br/ResoCFM.htm >. Acesso em mare 2010.

COSTA, C. A.; YAMIN, A. C.; GEYER, C. F. R. Toward a General Software Infrastructure for Ubiquitous Computing. **IEEE Pervasive Computing**, Los Alamitos, CA, USA, v.7, n.1, p.64–73, 2008.

COULORIS G. ; DOLLIMORE J.; KINDBERG, T. (Ed.). **Distributed systems - concepts and design**. [S.l.]: Addison Wesley, 2005. 657 - 719p. n.cap. 6.

DINIZ, J. **UbiDoctor**: Arquitetura de Servi para Gerenciamento de Sess Adpta de Contedo em Ambientes de Medicina Ubua. 2009. 178p. Tese (Doutorado em Ciia da Computa) — Universidade Federal de Pernambuco, UFPE, Recife, PE.

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

JOHNSON, T. **Uma Arquitetura de Computa Pervasiva para Trabalho de Campo**. 2005. Tese (Doutorado em Ciia da Computa) — Universidade Federal de Pernambuco, UFPE, Recife, PE.

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

PERTMED, . **PERTMED - Sistema de TeleMedicina M, disponibilizando a informa onde ela ?ecessa**. Disponel em: <¡ http://pertmed.wkit.com.br/pertmed/doku.php >. Acesso em novembro de 2009.

RODRIGUEZ, M. D. e. a. Location-aware access to hospital information and services. **IEEE Transactions on Information Technology in Biomedicine**, [S.l.], v.8, n.4, p.448–455, 2004.

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

TENTORI, M.; FAVELA, J. Activity-aware computing for healthcare. **IEEE Pervasive Computing**, [S.l.], v.7, n.2, p.51–57, Abril 2008.

VARSHNEY, U. Pervasive Healthcare. **IEEE Computer**, [S.l.], v.36, n.12, p.138–140, 2003.

WEISER M.;GOLD, R. B. J. S. The origins of ubiquitous computing research at parc in the late 1980s. **IBM Syst. J.**, [S.l.], v.38, n.4, p.693–696, 1999.

YAMIN, A. **Arquitetura para um Ambiente de Grade Computacional Direcionado plicas Distribuas Ms e Conscientes do Contexto da Computa Pervasiva**. 2004. 195p. Tese (Doutorado em Ciia da Computa) — Instituto de Informca, UFRGS, Porto Alegre, RS.

10 ASSINATURA