Matheus Alves de Resende

Protótipo de Software: Beautiful Book Generator

Belo Horizonte

2019/2

Matheus Alves de Resende

Protótipo de Software: Beautiful Book Generator

Apresentado como requisito da disciplina de Projeto Orientado a Computação II DCC/UFMG

Orientador:
Marco Túlio de Oliveira Valente - Departamento de Ciência da Computação

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Belo Horizonte

2019/2

Resumo

O relatório a seguir descreve a criação, sob a metodologia Lean Startup, de um protótipo de produto chamado Beautiful Book Generator. O sistema tem como função tratar a publicação de textos na web realizada pela aplicação Web Google Docs(1), que é considerada inadequada em termos de layout e segurança por uma grande parte dos usuários, republicando o texto com um design universal e mais profissional.

Sumário

LISTA DE FIGURAS

1 INTRODUÇÃO

Nós, enquanto sociedade, presenciamos nas últimas décadas o desuso gradual de documentos manuscritos e datilografados, em detrimento de textos digitados em computadores pessoais. Neste contexto, várias foram as soluções de software que tinham como proposta auxiliar a redação de textos neste novo paradigma. Não obstante o inegável progresso trazido pela revolução tecnológica na criação de textos variados. Novos problemas e desafios surgiram para as pessoas quando pensamos em como os documentos hoje em dia são divulgados entre as múltimas plataformas e programas existentes. A falta de compatibilidade entre editores/visualizadores de texto traz problemas na visualização usuário, que em tempos passados apenas precisava de uma boa iluminação (e possivelmente um óculos) para realizar uma leitura, possuindo a certeza de que estava lendo exatamente da forma que o escritor queria que o texto estivesse escrito. O Google Docs, nascido com este nome no ano de 2012, é uma aplicação Web amplamente usada para a redação de textos e é extremamente emblemática para este novo dinamismo que a tecnologia trouxe à escrita conteporânea. Entratando os usuários da plataforma enfrentam problemas na publicação de textos na web. A formatação feita no ambiente de edição da ferramenta, se perde na web, resultado em documentos pouco agradáveis aos olhos dos leitores. O trabalho proposto neste documento, será uma aplicação Web com nome Beautiful Book Generator (BBG), e tem como intuito tratar esta dificuldade na publicação de livros e textos em geral no meio digital. Em especial, a forma de que o Google Docs publica seus textos através da internet.

2 CONTEXTUALIZAÇÃO E TRABALHOS RELACIONADOS

O incômodo dos usuários com o design quebrado gerado pela publicação na web feita pelo Docs já foi motivação para o surgimento de algumas aplicações. Um exemplo é o website Google Doc Publisher(2), onde você digita a url da publicação do seu texto gerada pela Google e recebe uma republicação com o layout corrigido. Entretanto, diferente da solução proposta aqui, não existe uma reforma estrutural e nem de design da publicação em questão, sendo a tarefa realizada pelo site resumida na remoção do cabeçalho e do rodapé adicionado pela Google, e redimensionamento da publicação em um espaço retangular semelhante a uma folha A4.

Outra aplicação web, que possui uma proposta um pouco diferente, mas que se assemelha no que diz respeito a manipulação e conversão de mídias para publicação na Web, é o Designnr(3). Diferentemente do primeiro site, que é gratuito, os serviços oferecidos são pagos. A página cria eBooks, Show Notes e Web Pages através de Blogs, Podcasts e Vídeos a partir de documentos das mais variadas extensões.

3 DESENVOLVIMENTO DO TRABALHO

3.1 Método Lean

O desenvolvimento da aplicação/sistema foi inspirado no processo Lean de desenvolvimento de produto. Um MVP contendo os requisitos mínimos da aplicação foi criado e publicado atráves de uma plataforma cloud, de tal modo que o mesmo pudesse ser acessado por qualquer pessoa interessada no produto.

Para criar a aplicação foi utilizado o framework React, e todos os componentes do produto foram programados usando somente Javascript.

3.2 Primeira versão

A primeira versão do produto implementava uma solução semelhante ao que foi observado na aplicação Gpub, onde o usuário entrava na própria página da aplicação e fornecia o link da sua publicação. A abordagem inicial do produto tinha como objetivo seguir a filosofia WYSIWYG (What You See is What You Get), implementando na republicação todas o design presente no texto original do usuário. O código era sanitizado e os demais scripts perigosos eram removidos, entretanto o design era mantido. Havia também a opção de editar diretamente a publicação, caso o usuário achasse que era necessário aplicar algum *hotfix* na republicação. Além disso, textos de diferentes publicações poderiam ser fornecidos, formando capítulos para o livro. Ao final seria necessário que o criador do livro exportasse a página HTML gerada com todos os capítulos para ser distribuida para os leitores. A imagem abaixo mostra como foi a primeira versão do produto.

Figura 3.1: Primeira versão

Esta primeira proposta enfrentou uma série de problemas, também apontado pelos usuários que ajudaram a validar a aplicação. Era constante a reclamação de que em determinadas configurações a aplicação não conseguia reaplicar o estilo que o usuário tento aplicar no seu texto enquanto estava no editor. Ficou claro no processo de validação, que a solução proposta para o problema não se demonstrava escalável, uma vez que a criatividade das pessoas para criar diferentes estilos era enumerável,e, tão importante quanto, muitas vezes este estilo não poderia ser completamente recuperado através da tradução para HTML gerada pela Google. O design também foi totalmente rejeitado pelos usuários.

Os resultados desta validação junto aos usuário levou ao retrabalho da abordagem feita. Levando em consideração as limitações já expostas aqui uma segunda versão da aplicação foi criada e submetida para o ciclo Lean. Desta forma foi realizado o pivot da aplicação, que agora ao invés de apenas manter o design e o estilo do texto do usuário, agora tem como objetivo fornecer uma apresentação universal e robusta para os textos republicados.

3.3 Segunda versão

Neste ponto a abordagem pensada foi utilizar o html em sua forma mais pura, ou seja, fazer com o que o design da página seja criado a partir da semântica das próprias tags HTML. Entretanto, era necessário uma aplicação de estilo mínima para gerar de fato uma publicação agradável. Para tal, fora usada uma versão modificada do miniCSS(4), alterando principalmente o estilo aplicado a imagem e tabelas.

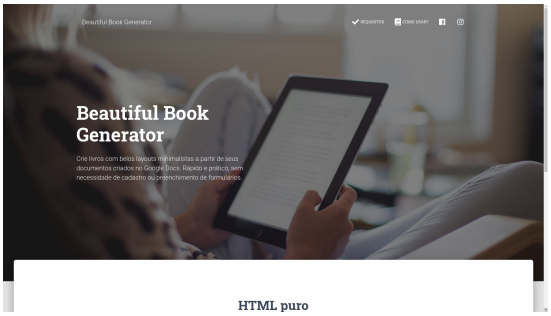
Uma página web separada da aplicação em si foi criada com o intuito de divulgar o produto, instruir o usuário de como distribuir seu livro/texto retrabalhado e sugerir algumas boas práticas de escrita de textos no Google Docs que poderiam facilitar a aplicação a gerar a estrutura HTML adequada.

Desta forma, retirou-se toda a parte gráfica do webpage da aplicação original. que não disesse respeito ao texto republicado em si. Para acessar o seu texto republicado, o usuário deveria pegar o url da publicação da Google, retirar o string gerado ao final, e entrar na publicação concatenar esta string ao final da url da pagína da aplicação.

Para se distribuir uma versão do livro modificado então, o autor do mesmo precisa apenas, ao invés de divulgar para seus leitores a url gerada pela Google, divulgar a URL modificada da aplicação. Qualquer modificação feita no Google Docs por parte do autor é propagada para o texto gerado pela aplicação, no momento que o leitor atualizar o seu navegador.

Para representar código no livro gerado a aplicação requeria que o escritor modificasse o seu documento, uma vez que se algum código fosse escrito de forma solta em um texto o Beautiful Book Creator certamente consideraria este trecho apenas como uma parte mal formatada do documento, e toda a identação seria desfeita.

Para a aplicação, qualquer tabela que contivesse apenas uma linha, não deve ser interpretada como uma tabela, e qualquer trecho de texto dentro desta linha será representado como pedaço de código de uma linguagem de programação. Então era necessário que sempre que o autor quisesse que seu livro contivesse código de programação qualquer. O usuário deve então criar no Docs uma tabela com apenas uma célula (ou com várias células na mesma linha, se quisesse representar lado a lado as instruções) para ”encapsular” o código.



(a) Parte superior

(b) Parte inferior

Figura 3.2: Landing page da segunda versão

pausa de 10 minutos para descanso. Na próxima sessão, os pares e papéis (líder vs revisor) são trocados. Assim, se em uma sessão você atuou como revisor do programador X, na sessão seguinte você passará a ser o líder, mas tendo outro desenvolvedor Y como revisor.

Programação em pares, tal como proposto por XP (foto da Wikipedia, licença CC-BY)

Mundo Real: Em 2008, dois pesquisadores da Microsoft Research, Andrew Begel e Nachiappan Nagappan, realizaram um survey com 106 desenvolvedores da empresa, para capturar a percepção deles sobre programação em pares (link). Quase 65% dos desenvolvedores responderam positivamente a uma primeira pergunta sobre se programação em pares estaria funcionando

(a) Visão vertical

seguir. No resto desta seção, vamos explicar cada um deles.

Papéis	Artefatos	Eventos
Dono do Produto Scrum Master Desenvolvedor	Backlog do Produto Backlog do Sprint Quadro Scrum Gráfico Burndown	Planejamento do Sprint Sprint Reuniões Diárias Revisão do Sprint Retrospectiva

Papéis: Times Scrum são formados por um Dono de Produto (*Product Owner*), um Scrum Master e de três a nove desenvolvedores.

O **Dono do Produto** tem exatamente o mesmo papel do Representante dos Clientes em XP, por isso não vamos explicar de novo a sua função em detalhes. Mas ele, como o próprio nome indica, deve possuir a visão do produto que será construído, sendo responsável também por maximizar o retorno do investimento feito no projeto. Assim como em XP, cabe ao Dono do Produto escrever as estórias do usuário e, por isso, ele deve estar sempre disponível para tirar dúvidas do time.

(b) Visão horizontal

Apesar disso atingimos nosso objetivo: temos um teste compilando, executando e falhando! Ou seja, chegamos ao estado vermelho.

Estado Verde: o teste anterior funciona como uma especificação. Isto é, ele define o que temos que implementar em ShoppingCart. Logo, mãos à obra:

```
public class ShoppingCart {
 public ShoppingCart() {}
 public void add(Book b) {}
 double getTotal() {
  return 30.0;
 }
}
```

Porém, o leitor deve estar agora surpreso: essa implementação está incorreta! A construtora de ShoppingCart está vazia, a classe não possui nenhuma estrutura de dados para armazenar os itens do carrinho, getTotal retorna sempre 30.0, etc. Tudo

(c) Representação de código

Figura 3.3: Visualização mobile de um livro publicado

4 RESULTADOS E DISCUSSÃO

4.1 Infraestrutura

A versão final do produto era constituida de dois servidores Web. Um para servir a aplicação em si e outro para o website do produto. Essa divisão do sistema em duas instâncias de servidores diferentes se fez necessária para eliminar a possibilidade de que a url gerada pela aplicação conflite com alguma url já existente para o site do produto, com o crescimento da aplicação.

O processamento de fato era totalmente realizado no browser do usuário, e nenhuma informação do texto do usuário era enviado de para o servidor. Isto também eliminava também a necessidade de qualquer infraestrutura adicional para servir de backend para a aplicação.

A comunicação com o Google Docs é realizado diretamente da máquina do usuário que fará a leitura do texto. A imagem abaixo esquematiza a infraestrutura final do produto.

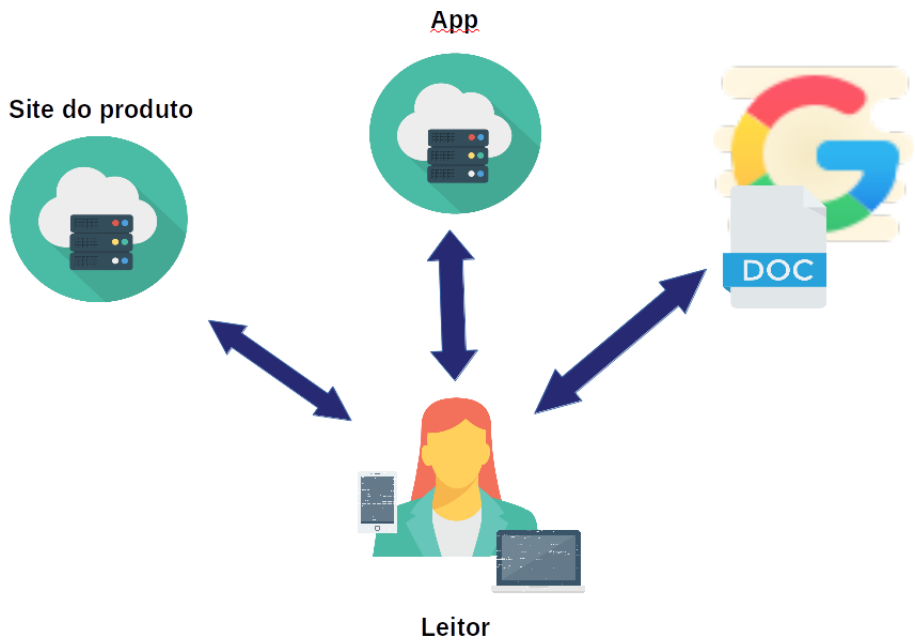


Figura 4.1: Infraestrutura do sistema

4.2 Aplicação Principal e Landing Page

A landing page contendo as instruções para acessar a aplicação principal pode ser conferida ao final deste relatório(5). Nela também estão presentes os requisitos mostrando como que um texto precisa ser redigido no Docs para que a aplicação gere adequadamente o livro.

A versão final da aplicação principal consiste em um página web, que ao ser acessada, verifica a url pelo qual foi acessada e realiza uma chamada GET ao servidores da Google requisitando a publicação. O código completo é aberto e pode ser visto no github(6).

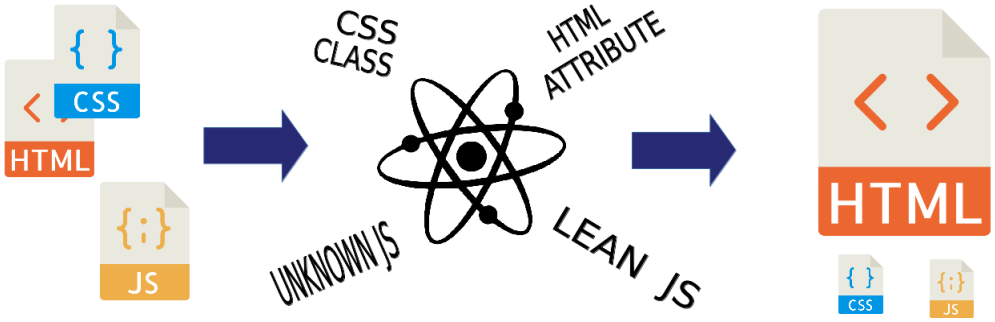


Figura 4.2: Transformação da página via React App

Os processos aplicados para realizar a transformação do texto são descritos no fluxograma abaixo.

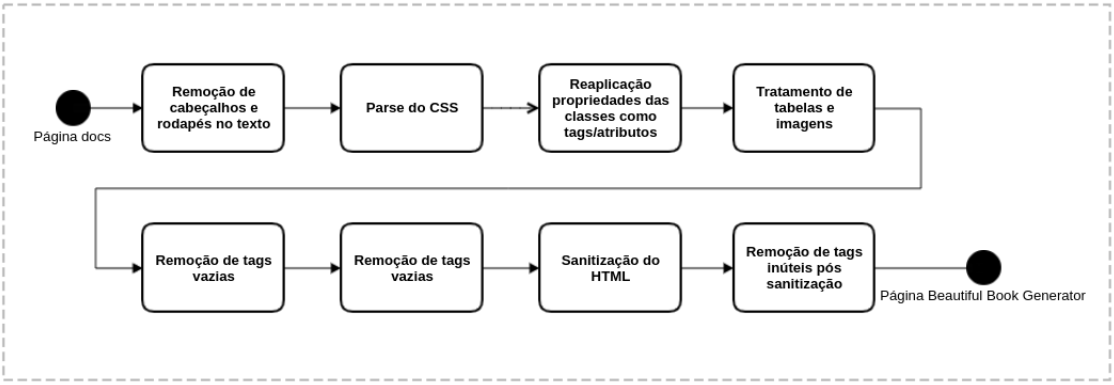


Figura 4.3: Fluxograma da transformação do texto

5 CONCLUSÕES E TRABALHO FUTUROS

A evolução do trabalho e o impacto do processo de *validated learning* do processo Lean Startup foi essencial para a qualidade final do protótipo. A recepção do produto final foi bem positiva e mostra que a evolução do protótipo para um produto de fato é factível.

Trabalhos futuros se resumem na continuação do ciclo Lean, eliminando bugs da aplicação e implementando features sugeridas pelos usuários. Alguns designs mais exóticos (imagens dentro de tabelas, posicionamente de imagens ao lado de texto) também precisam ter seu suporte incluído na aplicação. Outra possibilidade é a adição de mais estilos de textos, além da versão modificada do minicss que é aplicada.

Referências Bibliográficas

1 DOCUMENTOS Google: crie e edite documentos on-line gratuitamente. https://www.google.com/intl/pt-BR/docs/about/. Accessed: 2019-12-04.

2 GOOGLE doc publisher. https://gdoc.pub/. Accessed: 2019-12-04.

3 DESIGNRR. https://designrr.io/. Accessed: 2019-12-04.

4 MINICSS. https://minicss.org/. Accessed: 2019-12-04.

5 LANDING Page do Beatiful Book Generator. https://landing-257921.appspot.com/. Accessed: 2019-12-04.

6 APLICAçãO GitHub. https://github.com/maresende/beatifulbookgenerator. Accessed: 2019-12-04.