COMMUNICATIONS AND INNOVATIONS GAZETTE

Adaptação dos Minicursos do PET para o formato de Materiais Online

Bruno Corrêa Zimmermann, Jordi Pujol Ricarte, Érika Cota

[1] PET Computação – Instituto de Informática (INF)
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, RS – Brazil

{bczimmermann, jpricarte, erika}@inf.ufrgs.br

Abstract. *This paper describes the adaptation of PET Computação UFRGS's Minicourses project to an online format, due to the COVID-19 pandemic. We address in it both the process of choosing platforms and the course creation by itself. After analyzing some possibilities, we opted for the technologies MDBook, Git, Github Pages, and Travis CI to develop and publish our online courses. The group has received positive feedback from students about the published material's format and contents, which allows a collaborative and constant update. This experience showed us that the adopted model is adequated not only for these pandemic times but also to support in-person courses.*

Resumo. *Esse artigo descreve o processo de adaptação do projeto minicursos do PET Computação UFRGS para a modalidade online, em virtude da pandemia da COVID-19. São abordados tanto o processo de escolha de plataformas e ferramentas quanto a criação dos cursos em si. Após a análise de algumas alternativas, optamos pelas tecnologias MDBook, Git, Github Pages e Travis CI para desenvolvimento e disponibilização dos cursos online. O grupo tem recebido um retorno positivo dos estudantes sobre o formato e o material disponibilizado, que permite uma atualização constante e colaborativa do material. Essa experiência mostrou que o modelo adotado é adequado não apenas para esse período de pandemia, mas também como suporte a cursos presenciais.*

1. Introdução

O Programa de Educação Tutorial (PET) permite que alunos de ensino superior enriqueçam sua formação acadêmica atuando nos três eixos que formam o tripé acadêmico: ensino, pesquisa e extensão. Dentre as inúmeras oportunidades que o programa proporciona aos seus membros, uma delas é a de estudar e aplicar conhecimentos que, muitas vezes, não são abordados diretamente em sala de aula.

Este artigo é sobre uma experiência do PET Computação UFRGS com um projeto de ensino na modalidade remota. O grupo, assim como diversos outros pelo Brasil, tem como projeto tradicional a produção de cursos sobre conhecimentos e tecnologias utilizadas em suas atividades de pesquisa ou desenvolvimento. Os membros oferecem os cursos na modalidade presencial envolvendo, normalmente, aulas teóricas acompanhadas de demonstrações ou exercícios práticos. O público-alvo típico são alunos dos cursos de Ciência e Engenharia da Computação da UFRGS, mas ocasionalmente, alunos de outros cursos e a comunidade externa são também atingidos.

Com a pandemia da COVID-19 e o isolamento social, o método tradicional de oferta dos cursos previstos no planejamento do grupo teve que ser repensado. Em um primeiro momento, o grupo cogitou manter as aulas remotas síncronas, através de uma ferramenta de videoconferência. Apesar de ser possível, foram identificadas algumas desvantagens e dificuldades em dar aula ao vivo, pela *Internet*. A maior desvantagem é que limitaria o alcance do curso, já que seria necessária uma boa conexão, tanto para quem ministra quanto para quem assiste.

Por isso, percebeu-se que era o momento de buscar novas formas de ensinar, explorando vantagens da *Internet* e minimizando os problemas causados por ela. Ao estudar sobre como disponibilizar esse conteúdo de forma assíncrona, foram identificados diversos modelos de produção, divulgação e hospedagem de materiais. Este trabalho discute o processo de escolha do modelo definido como tecnologia de base para a oferta dos cursos do PET Computação UFRGS. O relato detalha ainda a produção e disponibilização de dois cursos no modelo escolhido.

Este artigo está organizado da seguinte maneira: na Seção 2 serão abordadas as plataformas consideradas para a aplicação de cursos, bem como suas vantagens e desvantagens. Na Seção 4 será explicado o uso das tecnologias escolhidas para criar materiais de cursos. Na Seção 5, detalhará os cursos criados pelo grupo PET Computação UFRGS utilizando MDBook e Github Pages. Por fim, a Seção 6 traz algumas considerações e conclusões sobre a experiência do grupo.

2. Formatos de cursos online

Em um primeiro momento, considerou-se ministrar aulas ao vivo, de forma síncrona, similar ao modelo tradicional. Porém, como mencionado na Seção 1, um problema com esse método é a falta de garantia de um bom acesso à *Internet* tanto para quem fosse aplicar o curso quanto para quem quisesse participar do mesmo. Além disso, encontrar um horário adequado para uma atividade síncrona com um bom número de participantes é sempre um desafio, tanto na modalidade presencial quanto na remota.

Com isso em mente, ficou decidido abandonar o modelo síncrono e publicar o material no modelo assíncrono. A ideia inicial foi de gravar videoaulas sobre os conteúdos que seriam abordados. Com isso, seriam aproveitados os materiais desenvolvidos para as aulas presenciais, mas haveria a assincronicidade que era buscada.

No entanto, o modelo de videoaulas também não se mostrou muito adequado para os cursos de Git e LaTeX, programados pelo grupo. Como será visto na Seção 5, o conteúdo desses cursos é mais técnico, mais prático, onde o texto é mais sucinto. Durante o processo de gravação, ficou evidente que as aulas gravadas tornaram-se monótonas, e regravar explicações demasiadamente longas podem requerer regravar toda a aula. Enquanto isso, pedaços de textos podem ser editados sem comprometer o resto do conteúdo.

Outro fator que tornou o modelo de videoaulas inadequado foi a difícil manutenção. Por ser a primeira vez que o grupo disponibilizava aulas de forma assíncrona, foi considerado de suma importância que elas pudessem ser atualizadas em casos de possíveis erros ou identificação de pontos de melhoria. De fato, não era incomum gravar mais de uma vez cada aula para corrigir algum engano cometido e o processo como um todo tornou-se lento e trabalhoso. Além disso, a atualização e aprimoramento das au-

las a cada edição de um curso já é uma prática do grupo e o rastreamento dos materiais (fontes, slides, etc) nem sempre é uma tarefa trivial.

Pelos motivos supracitados, optou-se pela criação de um material em texto que possa ser acessado a qualquer momento, em um ritmo individual. Não foi planejado acompanhar os alunos diretamente, mas havia a intenção de possibilitar e aceitar colaborações. Diversas ferramentas e plataformas foram consideradas, tanto para escrever esse material como para postá-lo. Essas alternativas serão discutidas na seção 3.

3. Ferramentas para produção e publicação de materiais

Nesta seção são analisadas algumas soluções para a produção e publicação de material textual, considerando os requisitos levantados na Seção 2. Tais requisitos são: atualização dinâmica do conteúdo, edição em texto simples e versionamento.

Com atualização dinâmica do conteúdo, entende-se que os membros do grupo devem ser capazes de adicionar conteúdos e corrigir problemas, de tal forma que as atualizações, com o menor tempo e esforço possíveis, cheguem nos leitores. Vários cursos oferecidos pelo grupo têm demanda contínua e são re-editados a cada ano. Enquanto isso, a edição em texto simples é necessária para que seja possível usar ferramentas de manipulação de texto, especialmente para automatizar mudanças como substituição de texto.

Pela mesma razão, cada edição precisa ser facilmente rastreável, o que pode ser feito através de um sistema de controle de versão. De acordo com [Somasundaram 2013], um sistema de controle de versão é ”um sistema capaz de registrar mudanças feitas em um arquivo ou um conjunto de arquivos sobre um período de tempo de tal forma que isso nos permita voltar no tempo, do futuro, para trazer de volta uma versão específica”.

Um exemplo sistema de controle de versão é o Git. Uma vantagem de usar Git é publicar o repositório do código fonte do material no *site* Github. Por sua vez, o Github disponibiliza um serviço chamado Github Pages, que hospeda um *site* estático, gerado a partir do conteúdo de um repositório no Github.

Existem diversas ferramentas e plataformas para a criação e disponibilização de conteúdo pela web. A partir de uma pesquisa inicial, selecionou-se algumas alternativas que os autores decidiram explorar antes de selecionar a plataforma que seria utilizada. São elas:

1. publicar um ou mais documentos PDF, possivelmente usando LaTeX como plataforma de criação;
2. publicar um *website* regular, feito manualmente com HTML, CSS e JS;
3. publicar um *website* feito com Wordpress;
4. publicar um *website* usando um gerador de *site* estático, como por exemplo:
   (a) Gitbook [1]
   (b) MDBook [2]
   (c) Hugo [3]

[1] https://www.gitbook.com/
[2] https://github.com/rust-lang/mdBook
[3] https://gohugo.io/

3.1. Publicar Como Arquivos PDF

A primeira opção avaliada foi a criação de materiais em PDF. Um ponto positivo do documento PDF é que, se feito em uma linguagem de texto simples como LaTeX, seria possível versionar usando um software como Git. No entanto, não seria muito adequado utilizar o Github Pages, uma vez que o objetivo da plataforma é publicar *sites*, então seria necessário, mesmo assim, que construir um *site* para publicar o documento.

Porém, uma grande desvantagem do documento PDF é que a atualização dinâmica é difícil de atingir em alguns casos. Versionar o material utilizando um *software* como Git e publicar em um *website* próprio, por exemplo, facilitaria a publicação contínua. Porém, por mais que fosse possível publicar versões continuamente, o usuário ainda teria que ativamente buscar (ou seja, baixar) a versão mais atualizada.

3.2. Publicar Um *Website* Implementado Manualmente

A segunda alternativa analisada foi a de fazer um *website*, criando manualmente todo o *design*. Este modelo tem diversas vantagens: maior controle sobre o código que faz parte do *website*; o formato é em texto simples, o que permite usar o versionador Git; a atualização dinâmica do conteúdo também é possível pois, usualmente, o *website* é carregado dos servidores, na sua versão mais atualizada, a cada vez que o usuário o acessa; e, por fim, é possível publicar o *website* como um *site* estático no Github Pages, facilitando a disponibilização do conteúdo.

Porém, o controle total do código do *website* pode também ser uma grande desvantagem. Para desenvolver o *design*, o criador do curso dedicará parte do tempo de desenvolvimento do conteúdo trabalhando com tecnologias como HTML e CSS. Esse revés também se aplica para a manutenção do curso, cuja revisão envolveria tanto o conteúdo técnico quanto o código fonte.

3.3. Publicar um *website* implementado em wordpress

Uma alternativa ao *website* implementado manualmente é o uso de ferramentas de criação de *sites* como o Wordpress[4]. O Wordpress é um CMS (do inglês, *Content Management System* ou Sistema de Gestão de Conteúdo), ou seja, é um sistema que facilita a criação de *websites*, fornecendo *designs* pré-implementados e possibilitando ao usuário focar na criação de conteúdo. O Wordpress é um sistema que funciona a partir de um servidor dinâmico, o que o torna bastante poderoso, capaz de interagir com bancos de dados persistentes.

O Wordpress cumpre o requisito de atualização dinâmica, pelo mesmo motivo do *website* implementado manualmente, com a vantagem de que focaríamos no conteúdo. O Wordpress tem um sistema de versionamento, algo que era buscado. No entanto, o Wordpress funciona através de uma interface gráfica, e então descumpriria o requisito de ser editado com texto simples. Além disso, como não é necessário de um servidor dinâmico ou de um banco de dados para o projeto, avaliou-se que o Wordpress seria exagerado para o nosso caso de uso.

[4]https://wordpress.org/

3.4. Publicar um *website* com gerador de *Site* estático

Um *site* estático é um *site* onde o servidor envia para um cliente (como o navegador) os arquivos exatamente como estão armazenados no servidor, fixos, sem processamento dinâmico. Um *site* estático não tem acesso a banco de dados persistente. Um gerador de *site* estático é um programa que a partir de uma entrada, gera arquivos fixos para serem utilizados em um *site* estático. Um exemplo de uso de tais geradores é produzir um *site* a partir de um formato de texto conveniente para a programação. O *site* estático cumpre os três requisitos que se buscava: é versionável, permite atualização dinâmica e é em texto simples.

3.5. Definição das plataformas de criação e disponibilização de cursos

A partir da análise apresentada nas seções anteriores, avaliou-se que publicar um *Website* com gerador de site estático seria a melhor opção para o caso. De fato, para os cursos planejados não seria preciso um bancos de dados ou processamento dinâmico e todo o conteúdo pode ser gerado a partir de algo mais simples. Além disso, decidiu-se utilizar o Git como ferramenta de versionamento, o Github como repositório público para disponibilização dos cursos e o Github Pages para hospedar o *site*. Por fim, nesta infraestrutura é possível automatizar o processo de geração e publicação do *site* usando uma ferramenta de integração contínua (do inglês, CI - *continous integration*).

De acordo com [Meyer 2014], o ciclo de vida de um processo de CI envolve executar um programa que automaticamente seleciona as mudanças feitas, aplica-as, e então executa vários comandos (como testes automatizados) para verificar se há algum problema nas mudanças. Uma ferramenta para utilizar CI no Github é o Travis CI[5].

Existem duas formas de publicar um *site* no Github Pages. Uma é usando os arquivos do repositório Git exatamente como estão. A outra, como descrito no parágrafo acima, é usar uma ferramenta de CI e executar um gerador de *site* estático com base nos arquivos do repositório a cada atualização do mesmo, e publicar o resultado no Github Pages. Optou-se então por esta última forma justamente para que fosse possível usar um formato conveniente de edição.

Entre os geradores de *site* estático considerados estão: Gitbook, MDBook e Hugo. O Gitbook é uma ferramenta que gera materiais divididos em capítulos. A desvantagem do Gitbook é que a edição em texto simples foi descontinuada, somente edição gráfica tem suporte atualmente. Com um visual semelhante ao do Gitbook, há o MDBook. No entanto, MDBook ainda usa edição em texto simples. O Hugo é relativamente diferente: ao invés de fixar o conteúdo do *site* no formato de um livro, Hugo permite construir *sites* estáticos em qualquer formato.

Usar qualquer um dos geradores acima com Github Pages permite o versionamento e a atualização dinâmica do conteúdo, mas somente MDBook e Hugo funcionam com edição de texto. Por isso, descartamos o Gitbook como opção de criação do conteúdo. Para os cursos planejados, a estrutura em capítulos é pertinente. Por essa razão, o formato do MDBook pareceu mais adequado. Além disso, ele provê um leiaute padrão que agradou ao grupo. O gerador Hugo, por outro lado, exigiria a definição e configuração de um tema, o que exigiria mais tempo de desenvolvimento. Foi concluído, então, que o gerador MDBook é o mais adequado para as necessidades do grupo.

[5]https://travis-ci.org/

Outro motivo para a escolha do MDBook é o uso da linguagem Markdown. Essa linguagem permite uma formatação básica do texto de forma simples. Isso é uma vantagem para a criação de cursos utilizando essa ferramenta, pois o criador do conteúdo não terá que aprender uma plataforma ou linguagem complexa para a criação do material.

Embora não fosse um requisito inicial, existe uma vantagem bastante interessante em publicar, no Github Pages, um *site estático* associado a um repositório Git que contém o código fonte do material. Tal vantagem é que o repositório está aberto para colaborações externas através do Github. Dessa forma, se um leitor quiser relatar um problema ou sugerir algo, ele pode abrir um *issue* no repositório hospedado no Github, ou até mesmo corrigir o problema e contribuir com o material através de uma PR (do inglês, *Pull Request*).

Assim, após a análise das tecnologias disponíveis, optou-se pelo uso de um gerador de sites estáticos – mais especificamente o MDBook – em conjunto com a ferramenta TravisCI e Git. Essas ferramentas, quando utilizadas em conjunto, atendem os requisitos de atualização dinâmica, versionamento e edição em texto simples.

4. Criação de conteúdo com MDBook e GitHub Pages

O MDBook é uma ferramenta que, a partir de arquivos na linguagem Markdown, renderiza livros e materiais similares. O formato de saída principal são arquivos nas linguagens HTML, CSS e JavaScript. Também é possível gerar arquivos PDF.

A ferramenta divide o material em capítulos e subcapítulos. Quando renderizados como páginas estáticas, os capítulos têm uma página HTML de introdução, e uma página para cada subcapítulo. No entanto, quando renderizados para PDF, os capítulos e subcapítulos são dispostos um após o outro no documento de saída.

4.1. Organização de arquivos do MDBook

Um projeto do MDBook utiliza um diretório inteiro. Os itens relevantes deste diretório são ”book.toml”, ”src/”e ”book/”. O arquivo ”book.toml”é o arquivo de configuração do MDBook, e é onde são definidos itens como título do material e autor. O diretório ”src/”é o diretório que contém os diversos arquivos dos capítulos e subcapítulos, terminados com a extensão ”.md”, arquivos da linguagem Markdown.

Além de capítulos e subcapítulos, no diretório ”src/”também ficam arquivos extras usados pelo livro, tais como imagens. Um arquivo especial dentro do diretório ”src/”é ”SUMMARY.md”, que tem a função de sumário. Esse arquivo, também no formato da linguagem Markdown, lista todos títulos de capítulos e subcapítulos, assim como os arquivos correspondentes.

Por fim, o diretório ”book/”é onde ficam os arquivos HTML, CSS e JavaScript gerados pelo MDBook a partir dos arquivos em ”src/”. Esses são os arquivos que devem ser abertos por um navegador ou publicados em um *website*, enquanto os arquivos em ”src/”são os arquivos editáveis.

4.2. A linguagem Markdown

A linguagem Markdown[6] é uma linguagem de marcação para escrever documentos, divididos em seções, com formatação simples, tais como itálico e negrito. Além disso,

[6]https://daringfireball.net/projects/markdown/

Markdown suporta a inserção de imagens, código fonte, listas, tabelas simples, entre outras funcionalidades. Para usos mais avançados, a linguagem suporta inserção de HTML.

O escritor de um documento não controla detalhes da aparência do documento renderizado, pois o renderizador escolherá um estilo adequado. Por exemplo, a fonte usada não é determinada pelo escritor, a menos que ele possa escolher um renderizador que permita customização da fonte.

A Figura 1 mostra um exemplo de código escrito em Markdown. Já a Figura 2 mostra uma página gerada pelo renderizador baseada nesse código

````
# Exemplo em Markdown
Este documento contém um exemplo em Markdown. Aqui
demonstram-se **algumas** propriedades do Markdown. Veja
[este _link_](https://www.markdownguide.org/) para mais
informações. Segue a imagem de uma batata, como exemplo:

![Uma batata](./batata.jpg)

## _Features_

Algumas _features_ que vemos neste documento:

* Títulos de Seções
* Itálico
* _Links_
* Código em linguagem de programação, como abaixo:

```python
name = "Markdown"
print("Hello,", name)
```
````

Figura 1. Exemplo de Markdown

Figura 2. Renderização do Exemplo da Figura 1 usando MDBook

4.3. Invocação do programa MDBook

O programa MDBook é um programa executado com linha de comando. Em um terminal ou *prompt* de comando, o programa deve ser invocado com subcomandos e opções adequados. A Figura 3 mostra exemplos de como invocar o programa, assumindo que o executável ”mdbook”pode ser encontrado através da variável de ambiente ”$PATH”.

```
1 ~/Documents/curso-de-git$ mdbook init
2 ~/Documents/curso-de-git$ mdbook build
3 ~/Documents/curso-de-git$ mdbook watch
```

Figura 3. Invocação de alguns comandos do MDBook

Por exemplo, ao executar o subcomando ”init” (linha 1) dentro do diretório, é inicializado um projeto de livro no diretório atual. A inicialização cria um arquivo de configuração ”book.toml”e um diretório ”src/”com o sumário dentro. É necessário editar a configuração para especificar o nome do livro.

O subcomando ”build” (linha 2) constrói o livro a partir da configuração, do sumário, e dos capítulos. Por padrão, os arquivos em HTML, CSS e JavaScript são inseridos no diretório ”book/”. O subcomando ”watch” (linha 3) faz com que o MDBook observe por mudanças nos arquivos e construa o livro automaticamente toda vez que um arquivo é salvo, enquanto estiver rodando em segundo plano.

4.4. Github Pages e Integração Contínua

O Github é um serviço de hospedagem de repositórios de *software* que usam o controlador de versão Git. O Github Pages é o serviço análogo da plataforma voltado para *sites* estáticos, usando os arquivos de um repositório Git.

A estrutura do MDBook é versionada pelo Git, e a página gerada é hospedada pelo Gihub Pages. Porém, como buscamos trazer uma atualização contínua e simples, vimos a necessidade de utilizar um processo de Integração Contínua. Dentro desse contexto, foi utilizado Travis CI, que faz o processamento dos arquivos do repositório antes de publicar o site no Github Pages.

Para usar o Travis, é necessário acessar o seu site usando a conta do Github e habilitar o CI para o repositório desejado. Além disso, o repositório precisa conter o arquivo de configuração do Travis chamado ”.travis.yml”. Esse arquivo é escrito no formato YAML (*YAML Ain’t Markup Language*).

Como exemplo, a Figura 4 contém um exemplo de arquivo de configuração. As linhas de 1 a 3 definem o ambiente: sistema operacional Linux sem instalação de nenhuma linguagem de programação em específico (”*minimal*”). As linhas de 5 a 8 definem um código de Shell do Linux que será executado toda vez que o Travis for acionado. Essas linhas realizam o *download* do programa MDBook e constroem o livro com o programa baixado.

As linhas de 10 a 17 especificam como o Travis vai publicar o livro (como ele vai fazer *deploy*). A chave ”provider”especifica que será usado o Github Pages, enquanto a chave ”token”especifica um *token* de acesso. Esse *token* tem que ser gerado pelo Github e

```
language: minimal
os:
  - linux

script:
  - curl -L -o mdBook.tar.gz https://github.com/rust-lang/mdBook/releases/download/v0
      ↪ .3.7/mdbook-v0.3.7-x86_64-unknown-linux-gnu.tar.gz
  - tar -xf mdBook.tar.gz
  - ./mdbook build

deploy:
  provider: pages
  token: $GITHUB_TOKEN
  keep_history: true
  local_dir: book
  skip_cleanup: true
  on:
    branch: master
```

Figura 4. Exemplo de configuração do Travis

deve-se acessar o repositório desejado no site do Travis para definir a variável de ambiente ”GITHUB_TOKEN”contendo o *token* gerado.

Enquanto isso, a chave ”keep_history”permite manter o histórico de atualizações do *site* dentro do repositório. A chave ”local_dir”especifica onde vai estar o *site* (o diretório ”book/”é onde o MDBook gera o livro em HTML) e a chave ”skip_cleanup”evita que o Travis remova o diretório temporário ”book/”criado pelo MDBook.

Finalmente, a chave ”on”especifica as condições para publicar o site, e foi especificado para só publicar se a *branch* (isto é, o ramo do versionamento do software) for ”master”(o ramo principal). Toda nova atualização nesse ramo fará com que o Travis publique o *site* baseado nos conteúdos daquela versão do *software*, usando o *script* que foi especificado para construir o *site*.

5. Cursos do PET computação UFRGS

Com base nas análises, descritas na Seção 2, assim como das tecnologias descritas na Seção 4, o PET Computação UFRGS produziu dois cursos em formato *online*. Um dos cursos é o de Git, ferramenta usada para versionar *softwares*, como mencionado anteriormente. O outro é o curso de LaTeX, uma linguagem voltada para formatar documentos acadêmicos. Ambos os cursos seguiram uma mesma metodologia para a sua elaboração, que será descrita a seguir.

5.1. O processo de criação

O processo de escrita dos dois materiais abordados nesse artigo foram semelhantes. Em um primeiro momento, os autores estudaram os temas que iriam abordar em seus materiais. Nessa etapa, além de utilizar de fontes disponíveis na internet, muito dos materiais antigos do grupo, provenientes de aplicações antigas dos cursos no formato presencial, serviram como base para os estudos e organização do formato dos materiais que viriam a ser produzidos.

Quando os participantes do projeto obtiveram domínio sobre os respectivos temas, inciou-se o processo de escrita. Nessa etapa, os autores se utilizaram das ferramentas cita-

das nessa seção para escrever os materiais. Também houve nessa etapa um processo constante de troca de experiências entre os autores que ocorriam principalmente em reuniões periódicas, de forma que tanto as experiências positivas quanto as negativas de um autor auxiliasse os demais.

Após a escrita da primeira versão de cada material, iniciou-se o processo de revisão, que envolvia todo o grupo PET Computação UFRGS. Nessa etapa, o autor enviou o conteúdo para o grupo, onde os membros faziam a leitura e enviavam sugestões para o mesmo. É importante dizer que, nessa etapa, o material já estava publicado no mesmo domínio que seria usado para a publicação da versão final desse. Isso foi possível graças a escolha da ferramenta de versionamento, o que facilitou a correção e adição do conteúdo do material.

Após todas as revisões serem feitas, e os autores ficarem satisfeitos com a qualidade do material produzido, foi feita a divulgação do material. Nesse momento, o grupo divulgou o link do material via postagens em redes sociais e lista de email dos cursos relacionados ao grupo PET Computação UFRGS.

5.2. O curso de Git

O PET Computação UFRGS aplicou cursos da tecnologia Git em diversas edições da Semana Acadêmica da UFRGS (SEMAC), realizada anualmente. Alguns meses antes da pandemia, os membros do grupo estavam revisando o material, e se preparando para aplicá-lo mais uma vez. O material foi largamente baseado nos manuais oficiais do Git[7].

Dadas as mudanças previamente mencionadas, em decorrência da pandemia, o curso foi publicado em Julho de 2020, em uma página do Github Pages[8]. [9]. O curso aborda tópicos como instalação, comandos básicos do Git, comandos para operar em *branches*, comandos para lidar com repositórios remotos, além de demonstrar o uso de Git em um projeto de exemplo. A Figura 5 contém um exemplo de capítulo do curso de Git.

5.3. O curso de LaTeX

No final do ano de 2019, o PET Computação UFRGS fez uma pesquisa informal no Instituto de Informática para saber quais cursos seriam ministrados no ano subsequente. Dentre todas as opções que o grupo ofereceu, um dos que teve maior procura foi o da ferramenta LaTeX, que é utilizada em diversos momentos ao longo do curso, porém não é apresentado formalmente para os alunos.

Apesar do grupo já ter aplicado cursos de LaTeX algumas vezes, a última vez que um desses havia sido oferecido foi em 2012. Por isso, mesmo já havendo um material, ficou decido recriar o curso, reaproveitando o que foi aplicado em anos anteriores, mas fazendo as atualizações necessárias e tentando dar um foco maior às aplicações do LaTeX na área da computação. Além dos materiais antigos, também utilizou-se artigos sobre a ferramenta, livro [Datta 2017] e um curso em pdf [Souto 2000] sobre a plataforma, além das documentações das bilbiotecas, disponibilizadas no site CTAN[10] .

[7]https://git-scm.com/docs
[8]https://pages.github.com/
[9]https://inf.ufrgs.br/pet/curso-git
[10]https://ctan.org

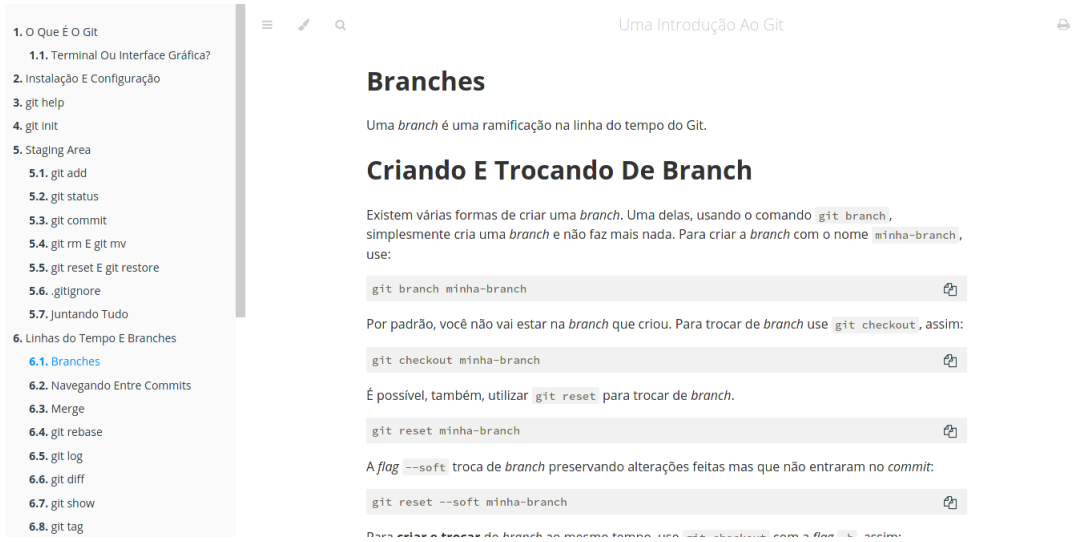

Figura 5. Renderização do Exemplo da Figura 1 usando MDBook

O grupo inciou a preparação do curso para aplicá-lo de forma presencial, utilizando apresentações de slides que alternavam entre explicações dos principais tópicos da tecnologia com exercícios práticos. Como mencionado, com o advento da pandemia da COVID-19 houve a migração do curso para a modalidade online.

Ao iniciar a migração, foi percebido que a criação de um material em texto que abordasse todos os principais tópicos do LaTeX – além de ser complexo e demorado – seria desnecessário, visto que já existem diversos livros e artigos completos e didáticos tanto na literatura nacional quanto internacional. Por isso, ao invés de criar um material que explicasse todas as possibilidades da plataforma, o grupo iria abordar os primeiros passos para que o leitor possa escrever um documento usando o LaTeX, focando nos tópicos mais usados na construção de textos da área da computação.

Com isso em mente, criou-se a primeira versão do material utilizando o MDBook, disponibilizado-a na plataforma Github Pages [11]. Nessa primeira versão, foram abordados tópicos como:

- Tipos de documentos;
- Divisão do texto em parágrafos, seções, etc;
- Formatação de texto;
- Criação de listas e tabelas;
- Inserção de fórmulas;
- Inserção de trechos de código e imagens;

Também foram pensado tópicos para serem inseridos em momentos futuros, como a inserção de bibliografia e utilização de arquivos *.bib*, assim como uma ordem de prioridade para cada uma dessas atualizações.

6. Considerações finais

Com as mudanças trazidas pela COVID-19, as instituições de ensino tiveram de adequar-se às medidas de distanciamento social, assim como os projetos desenvolvidos pelo PET

[11] https://inf.ufrgs.br/pet/curso-latex

Computação UFRGS. Dessa forma, buscou-se adaptar o projeto minicursos, para que, mesmo durante esse período anormal, fosse possível aprimorar o ensino da graduação.

Por isso, decidiu-se levar o projeto para uma plataforma *online* e assíncrona. Era desejado uma de fácil utilização, que permitisse atualizações constantes, de forma que os leitores recebessem novas versões de forma direta e automática. Entre as plataformas analisadas, a que melhor atendeu as demandas foi o uso conjunto do MDBook, Travis CI e Github Pages, para gerar os cursos como páginas *web*.

Com o MDBook gerou-se uma página completa, usando uma sintaxe simples, porém poderosa. Versionaram-se os cursos usando o *software* Git. Com o serviço GitHub, hospedaram-se os arquivos de desenvolvimento dos cursos, e no Github Pages, publicou-se as páginas. O Travis CI automatiza o processo de publicação – toda vez que uma nova versão dos arquivos de desenvolvimento é salva no GitHub, o Travis CI gera e publica as páginas do curso no GitHub Pages.

Ao escolher essas ferramentas, os autores puderam se concentrar no conteúdo dos cursos. Para construir os materiais, eles se basearam em conteúdos *online*, manuais, livros e cursos antigos do grupo, realizados presencialmente. Ambos os cursos aqui citados foram inicialmente separados em seções, cada uma com tópicos internos. Após isso, buscou-se em cada tópico trazer explicações claras com exemplos sucintos.

Foram recebidos diversos relatos informais de alunos utilizando os materiais, com *feedbacks* positivos. Como os grupos PET tem como um de seus objetivos melhorar o ensino de graduação, sentiu-se que esse foi alcançado.

Como foram alcançados todos os objetivos iniciais utilizando esse formato, pretende-se reutiliza-lo na produção de cursos futuros. Após o fim da pandemia, o grupo pretende utilizar os conhecimentos adquiridos para produção de novos materiais, que poderão ser utilizados tanto de forma independente, quanto como apoio para cursos presenciais.

Agradecimentos

Gostaríamos de agradecer a todo o grupo PET Computação da UFRGS pelas revisões de versões preliminares do artigo, pelas sugestões e pelo apoio para o escrevermos.

Referências

Datta, D. (2017). *LaTeX in 24 Hours A Practical Guide for Scientific Writing*.

Meyer, M. (2014). Continuous integration and its tools. *IEEE Software*, 31(3):14–16.

Somasundaram, R. (2013). *Git: Version control for everyone*. Packt Publishing Ltd.

Souto, G. (2000). *Curso de LATEX*.