Programação Padronizada com Alta Flexibilidade de Configuração

Mateus Eduardo Dotto[1], **Aline Duarte Riva**[1]

[1]Curso de Ciência da Computação, Universidade La Salle
92.010-000 – Canoas – RS – Brasil

{mateus.dotto0555,aline.riva}@unilasalle.edu.br

Abstract. *This paper addresses the development of a software capable of assisting in the programming of programs in general by allowing programmers to create and define code standards and models to be generated dynamically with the existence of variable code snippets in order to speed up the development process.*

Resumo. *Este trabalho trata do desenvolvimento de um software capaz de auxiliar na programação de programas em geral ao possibilitar que programadores criem e definem padrões e modelos de código a serem gerados dinamicamente com a existência de trechos de código variáveis com o intuito de agilizar o processo de desenvolvimento.*

1. Introdução

1.1. Problema de Pesquisa

É recorrente de projetos grandes com mais de um programador que o código final não esteja bem organizado ou seguindo apenas um padrão de escrita, uma vez que cada programador poderá programar de acordo com o seu estilo de escrita. Isso evidentemente afetará diretamente nos custos envolvidos em manutenção do software [Singh et al. 2020]. Além disso, há outros fatores técnicos que impactam nos custos de manutenção, como a complexidade do software e sua estrutura e a capacidade humana de analisar o código e interpretá-lo [Singh et al. 2020], fatores que também estão ligados à existência de padronização do código ou não.

Essa preocupação com os custos ligados à manutenção de software existe desde o início da era da computação [Ogheneovo 2014], porém, devido a alta velocidade do avanço da tecnologia, os programas de softwares de hoje estão cada vez maiores e mais complexos, aumentando os custos de manutenção envolvidos nos mesmos.

Inicialmente os programas de computadores normalmente não passavam de 1000 instruções de máquina, sendo escritos normalmente apenas por um programador e o processo inteiro de desenvolvimento do programa geralmente não custaria mais que $5000 dólares americanos. Já nos dias atuais, existem grandes sistemas de software com mais de 50 milhões de linhas de código, com mais de 1500 funcionários técnicos envolvidos e pode levar mais de 3 anos para a finalização de tais sistemas [Ogheneovo 2014].

1.2. Questão de Pesquisa

Como já discutido, os programas e softwares atuais podem ser muito maiores e mais complexos que antigamente e os custos envolvidos em manutenção dos mesmos é cada vez mais elevado. Dessa forma, seria possível desenvolver um programa ou software que auxiliaria um ou mais programadores a criar, pelo menos, partes do código de forma mais organizada e padronizada com maior rapidez (e com alta flexibilidade na definição das padronizações de código), e assim diminuir o número de manutenções no código e os custos relacionados a tais manutenções?

1.3. Objetivos

O objetivo geral deste trabalho é desenvolver um programa ou software que auxilie programadores a programar com maior rapidez e a padronizar código sem perder a flexibilidade na definição de tais padrões. Como consequência, isso também irá diminuir a quantidade de erros que poderiam existir no código pela ausência de um padrão de escrita, evitando-se custos relacionados a manutenções corretivas no mesmo.

Os objetivos específicos deste trabalho são:

- Escolher uma plataforma ou framework que irá servir como intermédio para desenvolver o software deste trabalho;
- Desenvolver o software atendendo os requisitos já mencionados, que são permitir a padronização de código sem perder flexibilidade e programar com maior rapidez dentro de tais padrões de escrita de código;
- Realizar um estudo comparativo entre usar e não usar o software desenvolvido para se analisar se o objetivo geral deste trabalho foi alcançado;
- Disponibilizar o software desenvolvido de forma livre e gratuita para a comunidade de desenvolvedores em geral.

1.4. Justificativa

Este trabalho se justifica por situações em que o autor desta pesquisa se deparou ao desenvolver soluções para sistemas web, onde notou-se que a existência de um software que auxilie na escrita de códigos poderia facilitar na padronização do código (evitando-se, assim, erros que poderiam surgir a partir de erro humano na escrita de alta quantidade de linhas de código) e, também, ser mais rápido do que escrever todo o código manualmente, sendo que tal software deva permitir alta flexibilização nas suas definições de padronização. Desta forma, o autor decidiu desenvolver tal software para auxiliar não apenas na programação do mesmo, mas também na programação da comunidade de desenvolvedores em geral.

Além disso, o trabalho também se justifica como um aprendizado, uma vez que o processo de desenvolvimento de um software é uma experiência diferente do processo de desenvolvimento de um sistema web.

2. Revisão Bibliográfica

2.1. Histórico da Programação

Exemplos das primeiras linguagens de programação de mais alto nível que existiram com a finalidade de representar instruções de linguagem de máquina em maior alto nível são

FORTRAN, ALGOL e COBOL. Essas linguagens são caracterizadas pela sua maior facilidade de entendimento por um ser humano e dispõem de recursos como variáveis, declarações condicionais, declarações iterativas, declarações de atribuições e estruturas de dados. Mais tarde, outras linguagens que podem ser citadas de alto nível que surgiram são C, C++ e Java [O'Regan 2016].

Linguagens de alto nível permitem que o programador foque na solução de problemas em vez de detalhes de programação associadas a linguagens de baixo nível [O'Regan 2016]. Como essas linguagens são de alto nível, também necessitam de menor quantidade de linhas de programação para se representar uma instrução ou para atingir um certo objetivo dentro de um programa em relação às linguagens de baixo nível.

Apesar disso, a complexidade e a quantidade de linhas de código necessárias nos programas sendo desenvolvidos vem aumentando (de 1000 instruções de máquina para 50 milhões de linhas de código) [Ogheneovo 2014]. Outra consequência disso é o tempo necessário para o desenvolvimento desses programas, que aumenta juntamente com o tamanho do programa.

Pode-se dizer que, quanto maior o tempo necessário para se desenvolver um programa, maior é o custo necessário para desenvolvê-lo. Além disso, a probabilidade de existirem erros no programa devido a erro humano também aumenta, onde a correção desses erros implica em mais tempo de desenvolvimento e, consequentemente, maior custo.

2.2. Sobre Softwares

O objetivo de um software é implementar uma solução para um problema definido pela intenção humana [Simonyi et al. 2006]. Além disso, especialistas de domínio são definidos como aqueles que têm conhecimento no domínio do problema e programadores aqueles que têm conhecimento na criação de software, e, juntos, eles desenvolvem e criam o software (sendo possível que existam pessoas pertencentes aos dois grupos ao mesmo tempo) [Simonyi et al. 2006].

Sobre a ciência de software, esta estuda as propriedades formais e modelos matemáticos do software, inclusive de teorias de geração de comportamento inteligente, onde recentes desenvolvimentos permitiram preparar o caminho para novas tecnologias para Inteligência Artificial [Wang 2022].

Já engenharia de software adota abordagens de engenharia para desenvolver softwares em alta escala com alta produtividade, baixo custo, de qualidade confiável e de planejamento de desenvolvimento controlável, cobrindo princípios heurísticos, ferramentas e ambientes, melhores práticas, estudos de caso, experimentos, testes e análise comparativa de performance [Wang 2022]. Além disso, a engenharia de software é um dos ramos mais complicados da engenharia, pois o seu objeto de estudo é altamente abstrato e intangível [Wang 2022].

Sobre a relação entre ciência de software e engenharia de software, essa pode ser explicada com uma analogia entre física pura e aplicada, onde sem a física teórica não haveria a maturidade da física aplicada. Da mesma forma ocorre entre a ciência e a engenharia de software [Wang 2022].

2.3. Sobre Frameworks

O termo framework normalmente é utilizado na engenharia de software, especialmente tratando-se de design e implementação de complexos softwares orientados a objetos [Stanojević et al. 2011].

Também, frameworks são utilizados para facilitar o reuso de código, sendo construídos desde a década de 80 até os dias atuais, com o intuito de criar o framework perfeito, porém ainda não há uma metodologia única para documentação e desenvolvimento do mesmo devido ao seu amplo domínio de uso [Stanojević et al. 2011].

Dessa forma, o objetivo da utilização de frameworks é auxiliar programadores no desenvolvimento de seus programas, sobretudo frente a questões que frequentemente se deparam, como reuso de código, padronização de código (que também facilita o trabalho em equipe), formatação de código, prevenção de erros, correção de erros e, sobretudo, gerenciamento de tempo e agilidade de programação.

2.4. Criação ou Utilização de Frameworks

A utilização de frameworks, sobretudo em empresas, tem suas vantagens, porém existe a pergunta se é necessário a criação de um framework específico para o caso encontrado ou se há a possibilidade de se utilizar um já existente no mercado. Várias organizações decidem desenvolver e arcar com os custos envolvidos na criação de um framework novo sem analisar primeiramente a necessidade de se utilizar um framework e sem avaliar os produtos comercialmente disponíveis [Fayad e Hamu 2000].

Além disso, a escolha de desenvolver ou comprar um framework para a organização é uma decisão fundamental que poderá persistir por até quinze anos ou mais [Fayad e Hamu 2000].

Frequentemente, a decisão de comprar ou desenvolver um framework se reduz às vantagens e desvantagens dessas opções, posteriormente citando-as [Fayad e Hamu 2000]. Sobre essas, inicialmente, como pontos positivos de se comprar um framework do mercado, há a possibilidade da utilização do mesmo quase imediatamente e, em geral, possuem uma qualidade maior devido à alta abrangência de potenciais clientes que devem atender. Além disso, a manutenção de um framework comprado do mercado é mais simples, uma vez que na compra desses produtos, é comum estar inclusa a existência de suporte técnico [Fayad e Hamu 2000]. Porém há necessidade de treinamento da equipe para que se possa utilizar tal framework de forma efetiva [Fayad e Hamu 2000].

Sobre a decisão de se desenvolver um framework, pode ser necessário uma equipe de desenvolvedores e uma estimativa de um a três anos para desenvolver um framework que atenda as funcionalidades requeridas [Fayad e Hamu 2000]. Entretanto há vantagens de se desenvolver o framework, principalmente quando a área de atuação requerida pelo framework for muito específica. Além disso, há a possibilidade da comercialização do framework criado pela organização, se for o caso.

Dessa forma, a tomada de decisão entre desenvolver ou comprar um framework do mercado não é simples. A organização deve analisar requisitos, valores e pesquisar profundamente as opções que dispõe, uma vez que essa decisão não é de baixo custo e deverá persistir por um grande intervalo de tempo.

Não obstante, a utilização de frameworks na programação também pode ter suas desvantagens, como perda de flexibilização para implementação de certa funcionalidade desejada. Esses tipos de situações são mais frequentes ao se utilizar frameworks já existentes de mercado, uma vez que esses são desenvolvidos para atender a maior parte de seus clientes e seus requisitos de forma abrangente.

Apesar disso, esse problema pode ser aliviado através do desenvolvimento e construção de um framework para atender o problema inicial, facilitando inclusive na alteração do mesmo para atender outros casos específicos não planejados inicialmente, porém, essa solução normalmente irá envolver maiores custos e tempo de desenvolvimento, como já comentado.

3. Metodologia

3.1. Tipo de Pesquisa

Sobre este trabalho, este será de natureza aplicada, uma vez que o objetivo do mesmo é gerar conhecimento para a criação e aplicação de uma solução prática para um problema específico. Sobre a sua abordagem, esta será de forma quantitativa, visto que os resultados obtidos pelo trabalho serão transformados em gráficos para posteriormente serem analisados.

Quanto aos objetivos deste trabalho, o trabalho se classifica como descritivo, pois busca descrever e analisar as consequências da aplicação do resultado deste trabalho. As pesquisas descritivas também têm como objetivo estabelecer o relacionamento entre variáveis [Gil 2002], que, neste trabalho, se referem aos dados coletados e ao uso ou não do software criado.

Em relação aos procedimentos técnicos, este trabalho é do tipo participante, uma vez que o autor está inserido no grupo afetado pelo problema de pesquisa descrito neste trabalho e que se caracteriza pela interação entre os pesquisadores e o grupo pertencente às situações investigadas [Gil 2002].

3.2. Delimitação

As delimitações deste trabalho são:

- O editor de texto que será utilizado para desenvolver o software será o Visual Studio Code (sem extensões);
- Será utilizado o framework Electron para desenvolver o software;
- Devido ao framework escolhido, a linguagem de programação utilizada será Javascript, juntamente com o uso de HTML e CSS;
- O software será desenvolvido de forma local, no computador do autor;
- O sistema operacional do computador utilizado será o Manjaro, que é uma distribuição Linux;
- O software permitirá a criação de modelos ou padrões a serem criados a partir do que for definido pelo usuário;
- Na etapa final, o software irá gerar arquivos no computador do usuário, os quais atenderão os requisitos iniciais deste, sem necessitarem mais nenhuma alteração ou edição;
- O local de geração dos arquivos será predeterminado pelo usuário, sendo obrigatório informar este antes da geração dos arquivos;

- O software permitirá inserir trechos de código variáveis, os quais serão preenchidos dinamicamente durante a geração dos arquivos;
- Os trechos de código que serão gerados dinamicamente seguirão regras e definições predeterminadas pelo usuário;
- Os resultados serão obtidos através da criação de um sistema web, comparando métricas ao se utilizar e não utilizar o software desenvolvido neste trabalho;
- As métricas escolhidas para comparação serão o tempo total para a criação do sistema, tempos parciais da criação de cada arquivo utilizado que irá compor o sistema como um todo, quantidade de erros encontrados durante o desenvolvimento do sistema e tempo gasto com correções de erros no código;
- As métricas descritas serão apresentadas e comparadas neste trabalho através de gráficos no formato de barras;
- Os códigos finais escritos, tanto de forma manual, quanto pelo software, serão idênticos e terão a mesma forma de padronização e estilo de escrita;
- O sistema web a ser criado irá representar, de forma simplificada, uma loja de carros;
- As informações disponibilizadas pelo sistema serão os clientes e os carros a serem vendidos, onde os clientes podem ter ou não comprado um dos carros;
- O sistema web será composto por três partes: um front-end, um back-end e um banco de dados;
- O front-end do sistema será escrito na linguagem Javascript e utilizará o React para a sua construção;
- O back-end do sistema também será escrito na linguagem Javascript e utilizará o Node.js para sua construção;
- O banco de dados utilizado será o MySQL e os dados serão armazenados localmente;
- Sobre o front-end do sistema, serão criadas cinco páginas, das quais, uma será uma página introdutória do sistema, duas páginas irão servir para listar dados de duas tabelas diferentes em forma de tabela na tela e as outras duas irão servir para mostrar de forma detalhada algum registro escolhido pelo usuário, em forma de um formulário HTML, permitindo alterar e salvar os mesmos no banco de dados;
- Sobre o back-end do sistema, serão criados dois arquivos para o tratamento de busca e alteração dos dados de cada tabela e outro arquivo de conexão com o banco de dados;
- Sobre as tabelas utilizadas, serão utilizadas duas: uma para os clientes e outra para os carros;
- A tabela de clientes conterá os seguintes campos e tipos: id (numérico), nome (texto), cpf (numérico) e data de nascimento (data);
- A tabela de carros conterá os seguintes campos e tipos: id (numérico), nome (texto), placa (texto) e proprietário (chave estrangeira referenciando o id de um registro da tabela de clientes);
- O processo de desenvolvimento do sistema descrito (tanto desenvolvido de forma manual quanto com o uso do software criado) será gravado para se obter as métricas de tempo e de quantidade de erros;
- O processo de desenvolvimento do sistema será realizado três vezes para cada forma de criação do sistema (com o uso e sem o uso do software), com um intervalo de 1 dia para cada processo de desenvolvimento;

- Todas as métricas obtidas serão apresentadas de forma individual para cada tentativa e posteriormente de forma coletiva para serem analisadas e comparadas;
- O software desenvolvido por esse trabalho será disponibilizado de forma gratuita para utilização da comunidade de programadores em geral através de um repositório público na página do GitHub do autor, encontrando-se em https://github.com/medotto/CodeStandardizationGenerator.

3.3. Técnicas de Coleta de Dados

A coleta de dados deste trabalho será realizada a partir das gravações dos desenvolvimentos do sistema web mencionado, utilizando e não utilizando o software desenvolvido. A partir das gravações, serão anotados os tempos parciais e totais da programação do sistema (de cada arquivo e, após, a sua soma total), bem como será contada a quantidade de erros que ocorreram durante a programação e o tempo gasto para corrigir os mesmos.

3.4. Técnicas de Análise de Dados

A partir dos dados coletados, serão criados gráficos em formato de barras para representar os dados que serão posteriormente analisados. Cada gráfico representará uma das métricas mencionadas, havendo, após, gráficos em barra que trarão a média dos valores correspondentes.

Após isso, serão feitas comparações entre os gráficos de mesmas métricas entre os resultados de utilizar e não utilizar o software criado. O critério de julgamento utilizado será se existiram diferenças nessas métricas.

Se encontradas diferenças, essas serão analisadas realizando-se uma proporção simples para se encontrar percentualmente qual a diferença entre utilizar e não utilizar o software.

O objetivo final esperado será encontrar uma diminuição nas métricas de tempo e de quantidade de erros ao se utilizar o software em relação a não se utilizar o mesmo.

4. Software Desenvolvido e seu Funcionamento

O software inicialmente permite ao usuário informar o template de código que pretende gerar dinamicamente (Figura 1 (a)). A seguir, o usuário clica no botão “Analisar”, onde o software irá ler o template digitado e gerar campos para o usuário preencher os trechos de código dinâmicos encontrados no template, para posteriormente serem substituídos com o digitado (Figura 1 (b) e Figura 2 (a)).

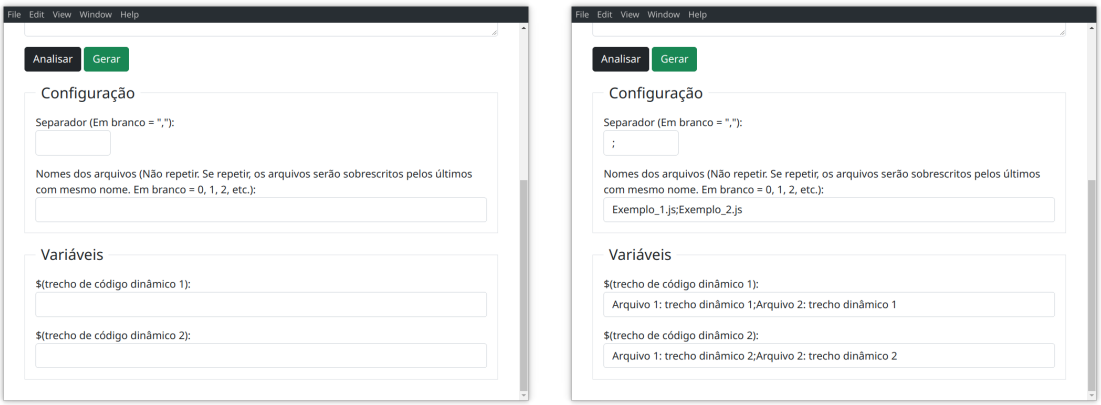
Também são gerados campos de configuração, onde o usuário pode preencher para informar um separador específico a ser utilizado pelo software, e outro campo para preencher com os nomes dos arquivos a serem gerados (Figura 2 (a) e Figura 2 (b)).

O usuário pode não preencher os campos de configuração. Se não preenchidos, o separador assumido será o caractere “,” e os nomes dos arquivos gerados serão números crescentes a partir do número um.

Após preenchidos os campos dos trechos dinâmicos do template (Figura 2 (b)), o usuário pode clicar no botão “Gerar”, onde deve escolher um local de seu computador para salvar os arquivos a serem gerados (Figura 3 (a)). O nome do arquivo informado nessa janela não importa. Finalmente os arquivos foram gerados e salvos conforme configuração e preenchimento informados pelo usuário (Figura 3 (b)).

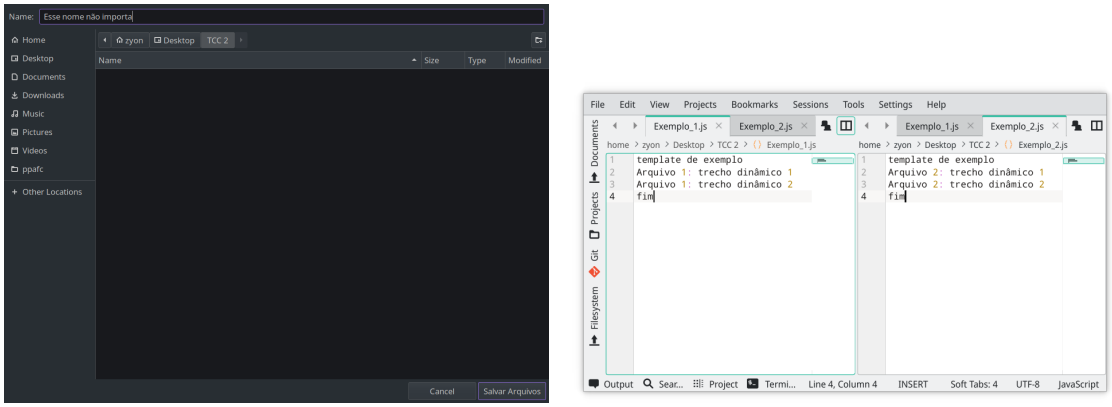
(a) (b)

Figura 1. (a) Estado inicial do software, (b) Template inserido e analisado



(a) (b)

Figura 2. (a) Campos gerados após análise, (b) Campos gerados preenchidos



(a) (b)

Figura 3. (a) Escolha do local para salvar os arquivos, (b) Arquivos gerados conforme especificação do usuário

5. Resultados Parciais

Não obstante, durante o desenvolvimento do software, foram notadas algumas considerações no momento de exportar e salvar os arquivos no computador do usuário: se existir mais de um arquivo a ser salvo com o mesmo nome, estes irão ser substituídos pelos próximos arquivos de mesmo nome.

Dessa forma, foi acrescentada uma observação no campo referente ao nomeamento dos arquivos (Figura 2 (a)), para que o usuário tenha conhecimento dessa peculiaridade. Apesar disso, o software desenvolvido está funcionamento corretamente de forma a atender os objetivos deste trabalho.

6. Considerações Finais e Trabalhos Futuros

De conclusão, ainda restam algumas considerações a serem finalizadas futuramente. Em particular, o estudo comparativo entre utilizar e não utilizar o software desenvolvido para se analisar a eficácia do mesmo e a posterior disponibilização de forma livre e gratuita do mesmo para a comunidade em geral, ambas considerações também já definidas nos objetivos específicos deste trabalho.

Ficam definidos como trabalhos futuros, então, realizar o estudo comparativo do software desenvolvido. Mais especificamente, a coleta de métricas de tempo e erros utilizando e não utilizando o software para o desenvolvimento de um sistema web básico de teste.

Após coletadas essas métricas, essas serão analisadas através de gráficos para comprovação ou não de que a utilização do software desenvolvido trará benefícios para os programadores, e que estes consigam programar com maior eficiência.

Referências

Fayad, M. e Hamu, D. (2000). Enterprise frameworks: Guidelines for selection. *ACM Comput. Surv.*, 32:4.

Gil, A. C. (2002). *Como elaborar projetos de pesquisa*. Atlas São Paulo.

Ogheneovo, E. (2014). On the relationship between software complexity and maintenance costs. *Journal of Computer and Communications*, 02:1–16.

O'Regan, G. (2016). *History of Programming Languages*, pages 189–211. Springer International Publishing, Cham.

Simonyi, C., Christerson, M., e Clifford, S. (2006). Intentional software. *SIGPLAN Not.*, 41(10):451–464.

Singh, C., Sharma, N., e Kumar, N. (2020). Analysis of software maintenance cost affecting factors and estimation models. *International Journal of Scientific & Technology Research*, 8:276–291.

Stanojević, V., Vlajić, S., Milić, M., e Ognjanović, M. (2011). Guidelines for framework development process. In *2011 7th Central and Eastern European Software Engineering Conference (CEE-SECR)*, pages 1–9.

Wang, Y. (2022). On the frontiers of software science and software engineering. *Frontiers in Computer Science*, 3.