Desenvolvimento e uso de Ferramenta CASE de geração de código automatizado baseada em *templates*

Pedro Henrique Albernaz Machado A. Tannús[1], Milton Miranda Neto[1]

[1]Ciência da Computação – Centro Universitário do Triângulo (UNITRI) – Av. Nicomedes Alves dos Santos, 4545, Bairro Gávea - CEP 38.411-106 – Uberlândia – MG – Brasil

{pedrohenriquetannus@gmail.com, voidmmn@gmail.com}

Abstract *In the software development process, one of the tools that can assist in its construction are the traditionally called CASE tools that aim to be very advantageous in all stages of the development of software systems. In this work, the focus is on exploring the benefits that can be obtained using technology for better routines, standardization, good practices, dynamism, greater deliveries, among others, with regard to supporting CASE tools, especially code generators. For this investigation, a CASE tool for automated code generation was created using the C# language, in which the development process was supported by the principles of agile. Subsequently, the dynamics of the tool developed were described, followed by a case study.*

Resumo. *No processo de desenvolvimento de software, uma das ferramentas que podem auxiliar na sua construção são as tradicionalmente chamadas ferramentas CASE que apontam ser muito vantajosas em todas as etapas do desenvolvimento de sistemas de software. Neste trabalho, o foco está em explorar os benefícios que podem ser obtidos usando da tecnologia para melhores rotinas, padronizações, boas práticas, dinamismo, maiores entregas entre outros, no que tange ao suporte às ferramentas CASE, sobretudo os geradores de códigos. Para esta investigação, foi criado a ferramenta CASE de geração de código automatizado utilizando a linguagem C#, no qual o processo de desenvolvimento foi apoiado pelos princípios de metodologias ágeis. Posteriormente, foi descrita a dinâmica de funcionamento da ferramenta desenvolvida, e em seguida um estudo de caso.*

1. Introdução

A evolução tecnológica sempre aconteceu concomitantemente com a história da humanidade, desde quando o homem inventou ferramentas para facilitar o trabalho (a pedra lascada, lanças, a roda, o relógio). Desde então, a humanidade vem constantemente, criando algum tipo de aparato, algum tipo de ferramenta que auxilie o seu trabalho.

Neste sentido, a evolução tecnológica na área do desenvolvimento não é diferente, segundo Martin (1995), é necessário uma revolução industrial do software, que provavelmente virá das técnicas orientadas a objeto combinadas com ferramentas CASE (*Computer Aided Software Engineering*), geradores de código, programação visual e desenvolvimento baseado em repositórios, uma vez que a geração de software não consegue acompanhar a produção de hardware.

Nesta perspectiva, este trabalho objetivou: o desenvolvimento de uma ferramenta CASE de geração de código automatizado, baseada em *templates* dinâmicos; implementar os artefatos gerados em um estudo de caso; analisar as vantagens e desvantagens e a viabilidade da ferramenta, na implantação em projetos de desenvolvimento de software; fazer uma comparação com ferramentas de mercado e também com a programação tradicional. A proposta da ferramenta é auxiliar no desenvolvimento de forma mais ágil para que o projeto seja desenvolvido em menos tempo, com padronização e menos erros de códigos.

A segunda seção apresenta a literatura consultada para fundamentação teórica na construção deste projeto, desenvolvendo conceitos importantes para o entendimento deste trabalho. A terceira seção descreve a arquitetura do sistema e as suas principais funcionalidades, assim como, as ferramentas e metodologias utilizadas para o seu desenvolvimento. A quarta seção demonstra um estudo de caso e pôr fim a quinta seção conclui sobre a viabilidade do uso da ferramenta desenvolvida.

2. Revisão Literária

Neste capítulo, serão apresentadas referências de outros autores que darão sustentação ao trabalho aqui desenvolvido, como por exemplo: as considerações sobre ferramentas CASE; os geradores de código automatizado e suas classes; *templates*; metodologias ágeis usadas para desenvolvimento deste trabalho e ferramentas CASE similares.

2.1 Ferramenta CASE

Segundo Sauter (2009), a utilização de ferramenta CASE foi originalmente implementada pela *Nastec Corporation de Southfield* em Michigan no ano de 1982. Pois, até então não existia uma demanda de programas que auxiliassem no desenvolvimento de software, uma vez que os softwares não eram tão complexos. No entanto, a partir dos anos 80 houve uma evolução das ferramentas CASE, que passou a ser uma realidade cada vez mais presente nas equipes e projetos. Segundo Nascimento (1993), CASE é a automação da automação, e fornece uma resposta prática aos problemas de produtividade, podendo ser vista também, como uma combinação de ferramentas de software com metodologias.

O termo CASE refere-se a diversos programas que auxiliam o profissional, desde a especificação até a manutenção do software. Por exemplo, existem ferramentas para análise de requisitos, modelagem do sistema, documentação etc. (SOMMERVILLE, 2007, p. 9).

Com o mundo cada vez mais veloz e demandando também soluções ágeis, o desenvolvimento de uma ferramenta CASE se mostra como vantajoso, uma vez que irá auxiliar na geração de código fonte para ser implementada em algum sistema, seja na sua criação ou manutenção. Como o principal objetivo destas ferramentas são facilitar o trabalho, as interfaces de comunicação com o usuário são geralmente de fácil utilização, ou seja, normalmente “as ferramentas de tecnologia CASE possuem facilidades gráficas para o planejamento e projeto de sistemas” (REZENDE, 2005, p. 181).

Ferramenta CASE é uma classificação que abrange todas as ferramentas baseadas em computadores, que auxiliam nas atividades de engenharia de software, desde análise

de requisitos e modelagem, até programação e testes. Podem ser consideradas como ferramentas automatizadas que tem como objetivo auxiliar o desenvolvedor de sistemas em uma ou várias etapas do ciclo de vida do desenvolvimento de um software, sobretudo na geração de código. Fisher (1990, p. 10) define que a geração automática do código, integral ou parcial, fornece as seguintes vantagens: redução do tempo de desenvolvimento, pois minimiza a necessidade de codificação manual, e aumento da confiabilidade do código gerado, o qual foi produzido por uma ferramenta depurada e testada, afirmando também que a geração automática de código define a capacidade de gerar automaticamente um software funcional ou compilável diretamente de uma especificação de projeto.

De acordo com Nascimento (1993), não há um padrão para categorizar as ferramentas CASE, mas pode-se identificá-las como:

a) ***Frontend* ou *Upper Case***: são aquelas que apoiam as etapas iniciais de criação do sistema;

b) ***Backend* ou *Lower Case***: são aquelas que apoiam a geração de código e os testes, isto é, a parte referente a implementação do sistema;

c) **I-CASE ou CASE Integrado:** são aquelas que apoiam todo o ciclo de vida do software.

O trabalho aqui proposto leva as características da categoria *Lower Case.*

2.1.1 Geradores de código automatizado

Durante o desenvolvimento de um software o fator humano é parte fundamental do processo, entretanto é também um fator passível de gerar erros e falhas, por isto deve-se buscar ferramentas para minimizar estas falhas e promover a padronização. Martin, (1994) afirma que sempre que possível, os programas devem ser gerados automaticamente a partir de projetos de alto nível, de especificações ou de imagens em uma tela CASE. Essa automatização de um processo que era gerado de forma manual, ou seja, em que o desenvolvedor iria escrever linha por linha de um sistema, pode ser facilitado pelos geradores de códigos, fazendo-se assim um processo mais automatizado. Segundo o dicionário Michaelis (2020), automação é,

> "Um sistema constituído por dispositivos mecânicos ou eletrônicos, utilizado em fábricas e estabelecimentos comerciais, em telecomunicações, em instituições hospitalares e bancárias, entre outros, destinado à operacionalização e controle dos processos de produção, que dispensa a intervenção direta do homem"

Entende-se por geradores de códigos como ferramentas que originam códigos-fonte de acordo com essas informações fornecidas pelo modelo de dados (MOREIRA; MRACK, 2003).

2.1.2 Classes de geradores de código automatizado

Para Herrington (2003, p. 17), geradores automáticos de código podem ser usados com o objetivo de gerar a infraestrutura do código, deixando para os engenheiros de software desafios de programação na solução dos problemas. Ainda de acordo com ele, a geração automática de código, além de eliminar o trabalho pesado, provê benefícios para a engenharia de software, tais como: qualidade, consistência do código, desenvolvimento ágil de software e flexibilidade.

Ainda afirma Herrington (2003, p. 28), que existem duas classes de geradores de código: passivos e ativos. Geradores passivos geram conjuntos de códigos e não mantém responsabilidade sobre estes códigos, deixando que os desenvolvedores possam alterar livremente. Geradores ativos de código são responsáveis pelos códigos que geram a longo prazo. Isso significa que, quando mudanças nestes códigos são necessárias, os desenvolvedores devem enviar novos parâmetros para o gerador e executá-lo novamente para obter o novo código.

2.2 *Templates*

De acordo com Dennis e Richard (1985), *template* de software é uma especificação sequencial de chamadas recursivas descritas em um modelo. Ainda sobre o trabalho de Dennis e Richard (1985), *templates* são definidos sobre valores de dados abstratos cuja sua implementação é especificada e catalogada.

Os *templates* determinam como a saída do programa (código fonte) será produzida. Neste projeto, *templates* de arquivos de *frontend*, *backend* e também procedimentos de bancos de dados são utilizados como meio para geração automática de código fonte.

Neste contexto, o modelo implementado dará grandes condições para que o operador desenvolva novas funcionalidades e propague estes de maneira rápida e personalizada, se adequando às novas demandas.

2.3 Metodologias Ágeis

Ser ágil em tempos contemporâneos é sinônimo de maiores entregas e consequentemente maiores lucros. No campo do desenvolvimento esta necessidade também se faz necessária, por isso fazer uso de metodologias ágeis podem ser uma boa alternativa para suprir esta necessidade.

A partir dos anos 90, surgiram os chamados métodos ágeis, influenciando uma nova visão na forma de desenvolvimento de software e estabelecendo princípios comuns através da criação do Manifesto Ágil. PRIKLANDNICK, (2014).

2.3.1 O método *Scrum*

O *Scrum* é uma metodologia ágil para gestão de projetos de software. Tem como uma das

principais mecânicas uma divisão de ciclos normalmente mensais ou quinzenais chamados de *Sprints*, que representa um período de tempo onde serão as executadas atividades e assim sucessivamente trazendo entregas de funcionalidades. De acordo com Schwaber (2002), as práticas gerências ou cerimônias do *Scrum* são: *Product Backlog, Daily Scrum, Sprint, Sprint Planning Meeting, Sprint Backlog e Sprint Review Meeting*.

Ainda segundo Schwaber (2002) o método tem como objetivo definir um processo para o projeto de desenvolvimento de software orientado a objetos, que seja focado nas pessoas e que seja indicado para ambientes em que os requisitos surgem e mudam rapidamente.

2.3.2 O método Kanban

Segundo Ferrari e Genari (2015), Kanban, que tem raízes nipônicas significa "cartão de visualização", é um conceito relacionados à produção enxuta. O Kanban originalmente foi utilizado na Toyota para sinalizar o progresso do trabalho na ala de manufatura. Diferentemente de outras metodologias ágeis, não possui iterações.

Adotado por times de desenvolvimento, o Kanban normalmente é empregado quando o projeto tem mudanças frequentes de escopo e priorizações.

2.4 Ferramentas CASE similares

Existem inúmeras ferramentas que oferecem algum tipo de mecanismo de geração de código. Algumas muito tradicionais como o GeneXus criado em 1989 ou outras novas a exemplo do Quixy desenvolvida em 2019, trazem interfaces muitas vezes complexas e a necessidade de grandes procedimentos e processos complexos para fazer a geração do código. Algo rápido e efetivo com as necessidades de um projeto pequeno para empresas de pequeno e médio porte, nem sempre são atendidas, levando em consideração também o alto custo como no caso do GeneXus, com R$999,00 por mês cotado em novembro de 2020. (GENEXUS, 2020).

A proposta deste trabalho propõe funcionalidades similares e tem por seu diferencial agilidade, simplicidade e customização.

3 Arquitetura

O projeto foi desenvolvido pensando na otimização do tempo laboral da equipe de desenvolvimento e de forma comumente vivenciada em qualquer *software house*: com uma boa estrutura, ferramentas adequadas, método de trabalho eficiente e uma boa proposta.

3.1 Escopo do Projeto

Entre as principais funções, o aplicativo permite que a partir de uma fonte de dados

oriunda de um SGBD SQL Server, seja feita a geração automática de código fonte e artefatos de programação via escrita de arquivos no disco, com base em *templates* texto e outros mecanismos de apoio à projetos como: Gerador de Soluções *dotNet* e Gerador de criptografia de dados.

Serão demonstrados nas próximas subseções Classes de geradores de código fonte, Métodos e Materiais, Ferramentas utilizadas nesse projeto, Controle de Versão, Estrutura e Organização e Mecânicas de funcionamento.

3.2 Classe Passiva de geradores de código automatizado

Com uma proposta de facilitar o trabalho do desenvolvedor e assim ganhar tempo e, consequentemente, trazer ganhos reais ao projeto, a ideia de criar um gerador passivo se mostrou ideal, em que o código gerado pudesse ser modificado pelo desenvolvedor e, por sua vez, promover um aumento de produtividade inicial.

O fluxo convencional de desenvolvimento de um software passa pelo processo de criação ou edição de código fonte, compilação e testes.

O fluxo para os geradores de código fonte automatizados ativos geram códigos fonte com base em *templates*, depois o procedimento de compilação e testes. Se forem encontrados problemas, ocorre um ajuste no *template* e então o processo para geração acontece novamente.

O fluxo para os geradores de código fonte automatizados passivos, que foi a escolha para este trabalho, geram códigos fonte com base em *templates* e depois o procedimento de compilação e testes. Caso encontrem problemas, é seguido o fluxo convencional. Assim, o usuário fará a adequação destes artefatos dali para a frente.

3.3 Métodos e Materiais

Neste trabalho foi utilizado para a criação da ferramenta CASE geradora de código a metodologia *Scrum* e *Kanban* como forma de orientar e gerenciar as etapas do processo de desenvolvimento, visando maior agilidade e promovendo um desenvolvimento sustentável.

A metodologia *Scrum* é bastante utilizada nas organizações e que nos permite ter um ótimo dinamismo e entregas sucessivas. Este processo foi fundamental uma vez que o mapeamento detalhado dos *templates* aconteceu durante a execução do trabalho. O objetivo macro a ser alcançado estava bem claro, mas as entregas vieram adequando o produto conforme a necessidade. Desta forma, o *Scrum* foi uma decisão de fácil aderência e trouxe a cada *Sprint* novas funcionalidades que ao serem entregues e validadas já poderiam ser incorporadas no cotidiano do desenvolvedor. O emprego do *Scrum* foi essencial para organização das demandas e priorização das tarefas, com a elaboração do *Product Backlog* e com o uso de *Sprints*.

O *Kanban*, por sua vez, que é uma grande metodologia nipônica de gestão que

pode trazer grandes ganhos de produtividade e organização.

Com foco em uma arquitetura de sistemas já existente, os arquivos de código fonte foram retirados de um projeto para serem abstraídos dando condições para a criação dos *templates*.

3.4 Ferramentas

Durante o desenvolvimento foram necessárias algumas ferramentas para o desenvolvimento desta ferramenta CASE geradora de código fonte automático.

Para a modelagem da mecânica do sistema, que é uma etapa essencial para a melhor compreensão do funcionamento da ferramenta, empregou-se o Draw.io. Trata-se de uma ferramenta web para a construção de fluxogramas e diagramas muito difundida pela comunidade e gratuita para uso.

Para construção do código fonte da ferramenta desenvolvida neste trabalho foi utilizado o Visual Studio 2019 da Microsoft, visto que é uma ferramenta muito estável e com uma versão gratuita para uso da comunidade, também é a ferramenta que os autores tem mais familiaridade e confiança. Seguindo essa mesma linha, o projeto foi escrito na linguagem C#, usando o framework 4.7 em uma solução Windows WPF (*Windows Presentation Foundation*). Essas tecnologias foram empregadas para que não houvesse dependência de um servidor de aplicação dedicado ao uso da ferramenta após sua conclusão ficando assim, a simples execução na estação de trabalho do desenvolvedor de forma autônoma.

Utilizando-se um *Kanban* na ferramenta Trello, foi possível fazer o controle das tarefas que seriam executadas em cada Sprint de maneira organizada e sem custos.

3.5 Controle de Versão do Trabalho

Desde o início da codificação, o código fonte e outros artefatos foram controlados por versão. O sistema de controle de versão utilizado foi o GitHub. Um dos maiores benefícios do controle de versão é distinguir as versões do código desenvolvidas durante o projeto. Uma cópia de trabalho é salva no computador do programador, e suas alterações são enviadas periodicamente ao repositório remoto.

3.6 Estrutura e Organização do Software

O catálogo de arquivos do projeto é baseado em alguns conceitos como gerenciamento de dependências, agrupamento de componentes visuais, armazenamento de modelos bem como isolamento de arquivos estáticos e arquivos do usuário. Suas principais pastas são Assemblies, Core, Presentation, UC e Template, conforme Tabela 1.

Tabela 1: Estrutura e Organização do Software

Diretório	**Descrição do Conteúdo**
Assemblies	Pasta para recursos usados na aplicação como o Ookii.Dialogs.Wpf.dll
Core	Classes de processamento, utilitários, provedor de acesso à dados.
Presentation	Arquivos para o *frontend* da aplicação
UC	Componentes de *frontend* reutilizáveis da aplicação
Template	Arquivos com os modelos que serão usados para geração automática de código fonte.

3.7 Mecânicas e núcleo de funcionamento

A ferramenta propõe funcionalidades que podem gerar um grande ganho laboral e isso impacta diretamente nos prazos e no orçamento dos projetos onde essa ferramenta for empregada. Assim sendo, obteve-se importantes fluxos listados na sequência.

3.7.1 Input

O sistema utiliza uma conexão com um banco de dados SQL Server para poder ler a estrutura básica de esquemas. A partir deste procedimento, é possível ter condições de mapear uma das tabelas do banco de dados que estejam disponíveis para leitura. Todos os campos e seus tipos serão utilizados para a geração automática de código fonte.

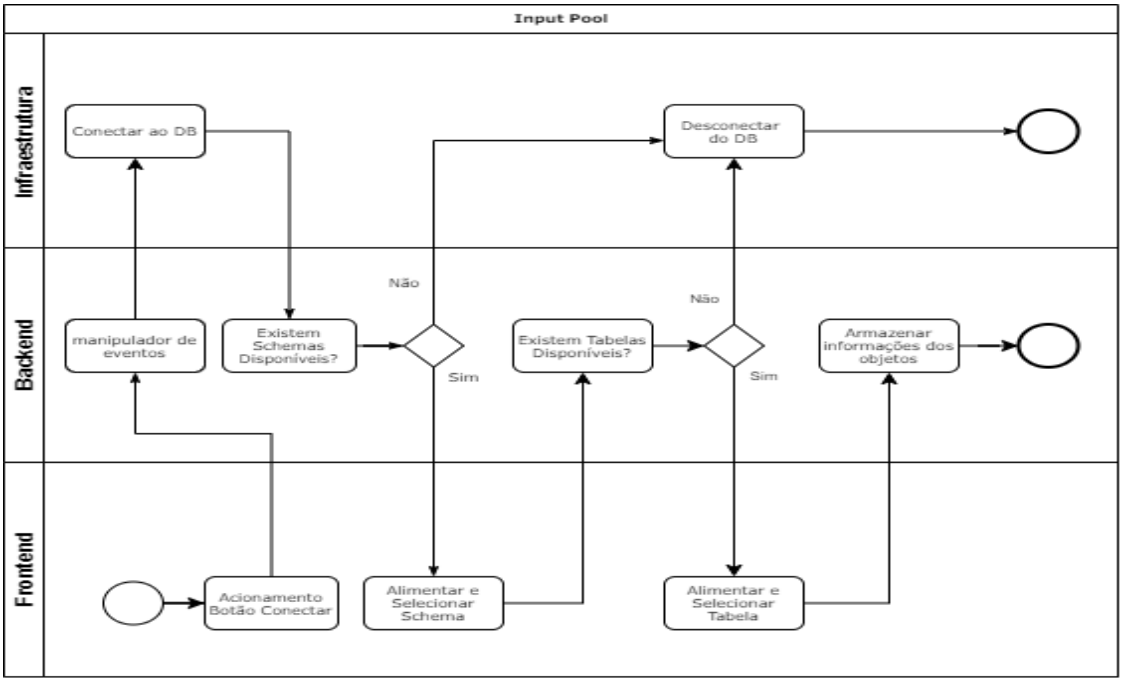

Figura 1. InputPool, Diagrama de Fluxo de entrada

Conforme a Figura 1, tem-se uma conexão estabelecida com um banco de dados, a ferramenta CASE faz uma leitura e mapeia os esquemas e depois lista em um componente para o usuário escolher. Feito isso o usuário pode selecionar uma dentre as tabelas uma para que o processo dê sequência. Após esse procedimento os dados estão armazenados e prontos para a etapa de Output.

3.7.2 Output

O sistema utiliza uma gama de *templates* para poder escrever arquivos de código fonte com uma série de palavras chave internas para que sejam feitas as substituições no processo de interpretação do *template*. Desta maneira, para cada arquivo definido pelo desenvolvedor, será criado um novo arquivo de código fonte.

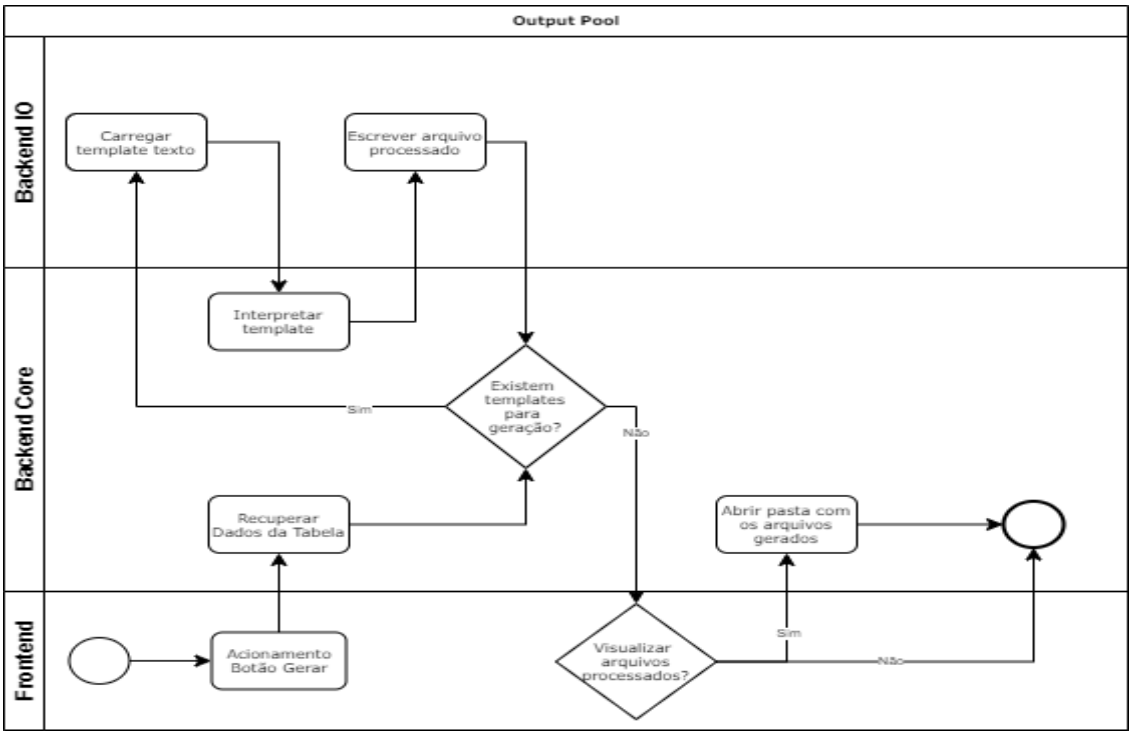

Figura 2. OutputPool - Diagrama de Fluxo de Saída

Conforme a Figura 2, o sistema recupera os dados da tabela e monta uma estrutura de repetição para cada objeto de *template*. A cada iteração faz a leitura do respectivo *template* e posteriormente executa a interpretação do mesmo (este processo faz substituições de palavras reservadas do sistema) e na sequência escreve em disco um arquivo de código fonte, correspondente aquele *template*.

3.7.3 *Template* texto dinâmico

O *template* em formato texto, mostra-se como uma grande vantagem, pois é possível trazer ajustes ou novas implementações sem que seja necessário uma nova versão da ferramenta CASE geradora de código fonte.

Este processo acontece no ato do processamento e interpretação dos *templates*, usando substituição de palavras chave do sistema por blocos de código processados de acordo com cada item.

Estas palavras reservadas foram elencadas com base em um estudo feito da arquitetura do sistema que incorporaria estes artefatos assim como cada arquivo *template*. A Tabela 2, lista os termos que, ao serem usados nos arquivos de *template*, serão substituídos por informações vindas da fonte de dados.

Tabela 2: Palavras reservadas dos *templates*

Palavra reservada	Instrução escrita	Palavra reservada	Instrução escrita
[<nome_base>]	Nome da tabela	[<proc_select>]	Nome do procedimento SQL para Select
[<nome_classe_bo>]	Nome do objeto BO	[<proc_update>]	Nome do procedimento SQL para Update
[<nome_classe_dao>]	Nome do objeto DAO	[<proc_insert>]	Nome do procedimento SQL para Insert
[<nome_classe_to>]	Nome do objeto TO	[<proc_deletet>]	Nome do procedimento SQL para Delete
[<campo_identity>]	Nome da Coluna com Chave Primária	[<parametros_select>]	Coleção contendo todas as colunas processadas
[<nom_schema>]	Nome do esquema	[<parametros_insert>]	Coleção contendo todas as colunas processadas
[<propriedades>]	Coleção contendo todas as colunas processadas	[<parametros_update>]	Coleção contendo todas as colunas processadas
[<proc_name>]	Nome do procedimento	[<proc_owner>]	Propriedade do banco

Estes *templates* devem ser escritos com foco no sistema que irá incorporar o código fonte o qual será gerado automaticamente. Para isso deve-se fazer um estudo do caso e levantar a arquitetura alvo preliminarmente. O caso é representado pela Figura 3:

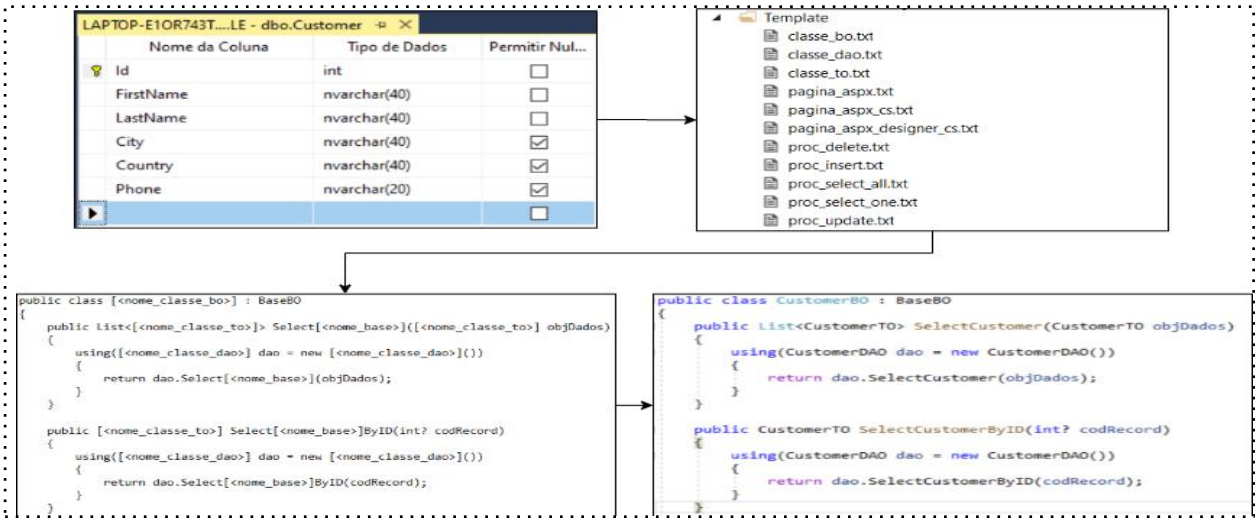

Figura 3: Transformação da fonte de dados para arquivos de código fonte

4. Estudo de caso

Para a realização deste estudo de caso, foi desenvolvido uma ferramenta CASE geradora de código fonte e posteriormente fez-se a sua aplicação em um projeto usando as tecnologias C# e AspNet. Na sequência, empregou-se os artefatos gerados e a implementação foi demonstrada através de figuras e tabelas.

Nesta seção, pretende-se fazer o emprego da ferramenta e analisar a viabilidade da ferramenta, levantando as vantagens e desvantagens que a mesma pode demonstrar.

4.1 Modelagem do banco de dados

Assim como toda ferramenta CASE geradora de código fonte, é necessário definir uma fonte de dados. Neste trabalho, o desenvolvimento foi feito a partir de um banco de dados relacional, e de maneira tradicional, a modelagem do banco de dados precedeu o desenvolvimento da aplicação, conforme Figura 4.

Figura 4: Diagrama de Entidades Relacional do banco de dados

4.2 Definição dos *templates*

Como forma de solução eficaz para uma empresa que já emprega uma arquitetura em seus produtos, o ideal é se atentar à arquitetura já existe nestes projetos para abstrair adequadamente os *templates*.

Nesta seção, será demonstrado o emprego em um projeto C# com *AspNet* contendo procedimentos de banco de dados para as operações de *CRUD*, que envolvem a criação, edição, seleção e deleção de registros, infraestrutura e também a interface gráfica para execução dos métodos e visualização dos registros.

4.2.1 Detalhamento

Com a abstração dos arquivos de código fonte, foram listados na Tabela 3 os arquivos texto que serão usados e suas respectivas atribuições.

Tabela 3: Palavras reservadas dos *templates*

Arquivo	**Descrição de Uso**
classe_bo.txt	Classe C#, camada de negócio
classe_dao.txt	Classe C#, camada de acesso à dados
classe_to.txt	Classe C#, classe de transporte, objeto simples com as propriedades usado para transferir dados
proc_delete.txt	Procedimento SQL para excluir um registro do banco de dados de uma determinada tabela
proc_insert.txt	Procedimento SQL para inserir um registro no banco de dados de uma determinada tabela
proc_select_all	Procedimento SQL para retornar todos os registros do banco de dados de uma determinada tabela
proc_select_one.txt	Procedimento SQL para retornar um registro do banco de dados de uma determinada tabela
proc_update.txt	Procedimento SQL para atualizar um registro do banco de dados de uma determinada tabela
pagina_aspx.txt	Arquivo contendo Tags CSS, HTML, Javascript e AspNet
pagina_aspx_cs.txt	Arquivo contendo código C# relacionado à página
pagina_aspx_designer_cs.txt	Arquivo com a orientação e propriedades dos objetos da página bem como eventos
template.zip	Solução genérica para ser usada em novos projetos

4.3 Interface gráfica

Por poder ser usada de forma bem intuitiva e objetiva, sua interface simples já mostra uma proposta para os nomes dos arquivos seguindo boas práticas de mercado. A Figura 5, apresenta a interface do sistema.

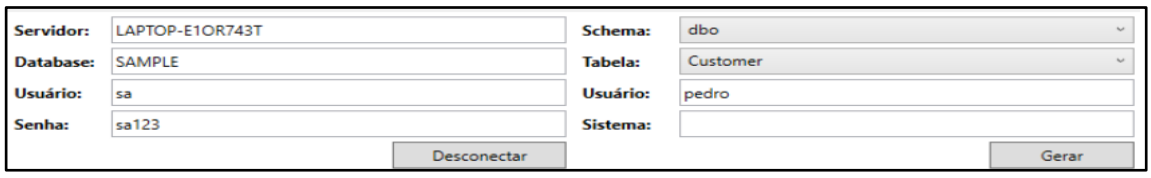

Figura 5: Consumindo o banco de dados

Seguindo um fluxo de execução de cima para baixo, o desenvolvedor tem a parte de conexão ao banco de dados e seleção do objeto alvo que é uma tabela do banco de dados, passando pela seleção de esquemas.

Logo após este procedimento na parte inferior a relação dos *templates* já com os nomes propostos dos objetos será exibida juntamente com o detalhamento das propriedades da tabela em questão representado na Figura 6.

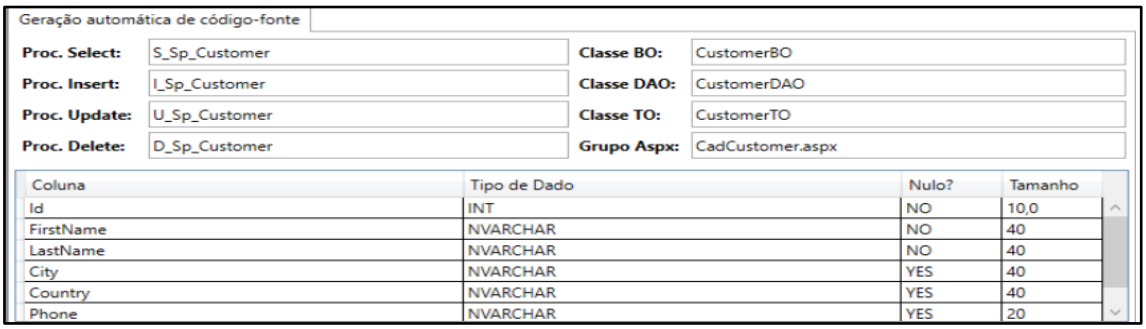

Figura 6: Sugestão de nomes dos objetos

Assim com o acionamento do botão gerar, o sistema irá abrir uma caixa de diálogo para seleção da pasta de saída e então irá disponibilizar neste caminho todos os artefatos para serem integrados em um projeto *AspNet*.

A interface gráfica demonstrada na Figura 7 a seguir.

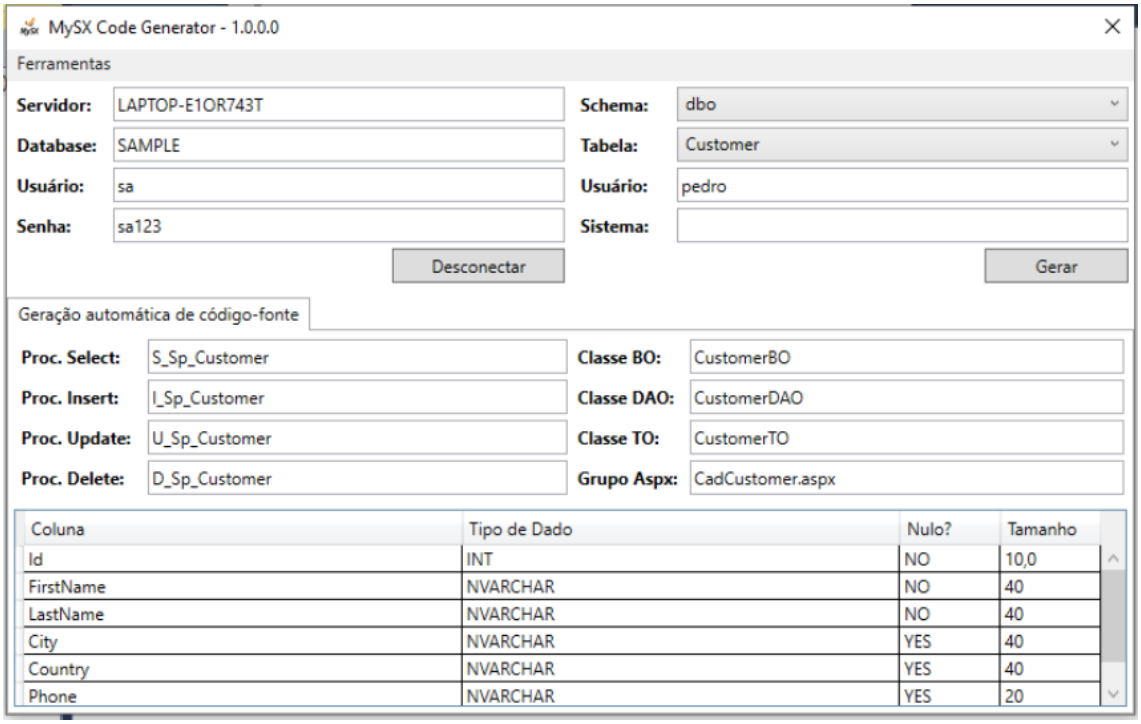

Figura 7: Interface Gráfica

4.5 Resumo do processamento

Para esse exemplo de uso consumindo uma tabela do banco de dados conforme já demonstrado, obtém-se como resultado os arquivos exibidos na Figura 8. Um resultado total de 614 linhas de código fonte gerados automaticamente em cerca de 2 segundos.

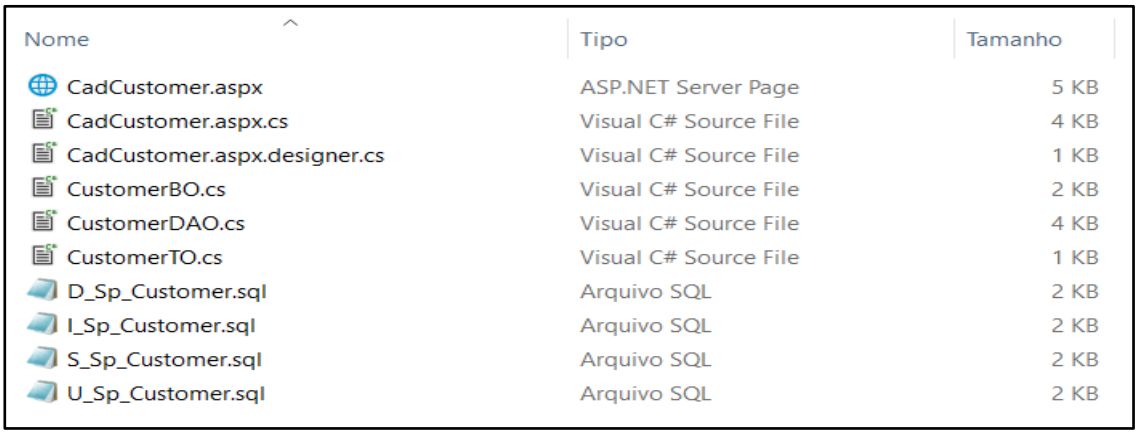

Figura 8: Resultado, código fonte gerado automaticamente

4.6 Implantação

A ferramenta CASE apresenta em seu menu “Ferramentas” um gerador de soluções *AspNet*. Esta solução tem uma infraestrutura pré configurada com a arquitetura alvo para acelerar o desenvolvimento e padronização de novos projetos conforme Figura 9.

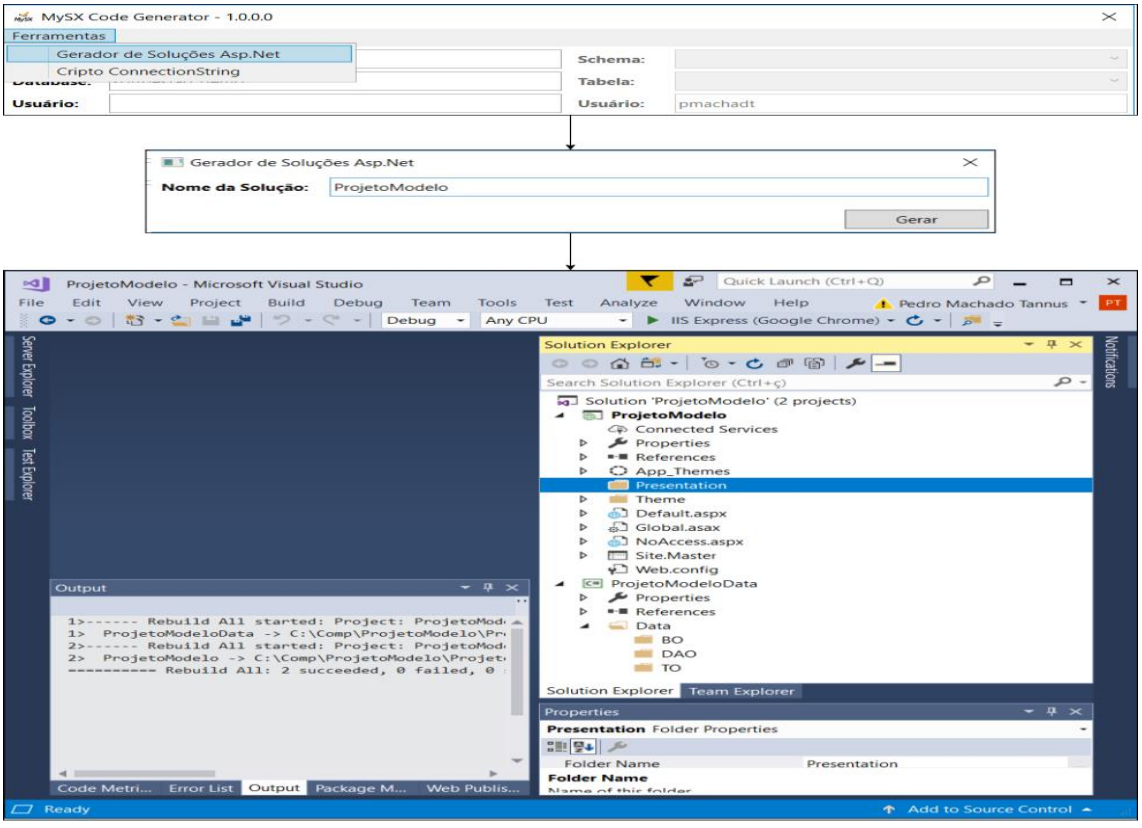

Figura 9: Resultado, solução gerada automaticamente

Com a mesma mecânica para substituições de palavras chave, esta solução será criada com um nome que o desenvolvedor fornecer.

Como a solução proposta gerada automaticamente já tem todo o arcabouço arquitetural mínimo para execução, não é preciso nenhum ajuste adicional para execução. Assim como os *templates* texto, o projeto gerado também pode ser alterado pelo desenvolvedor.

O procedimento para trazer os artefatos gerados é usual. Basta adicionar eles dentro da sua respectiva pasta dentro do projeto. A Figura 10 demonstra esse procedimento.

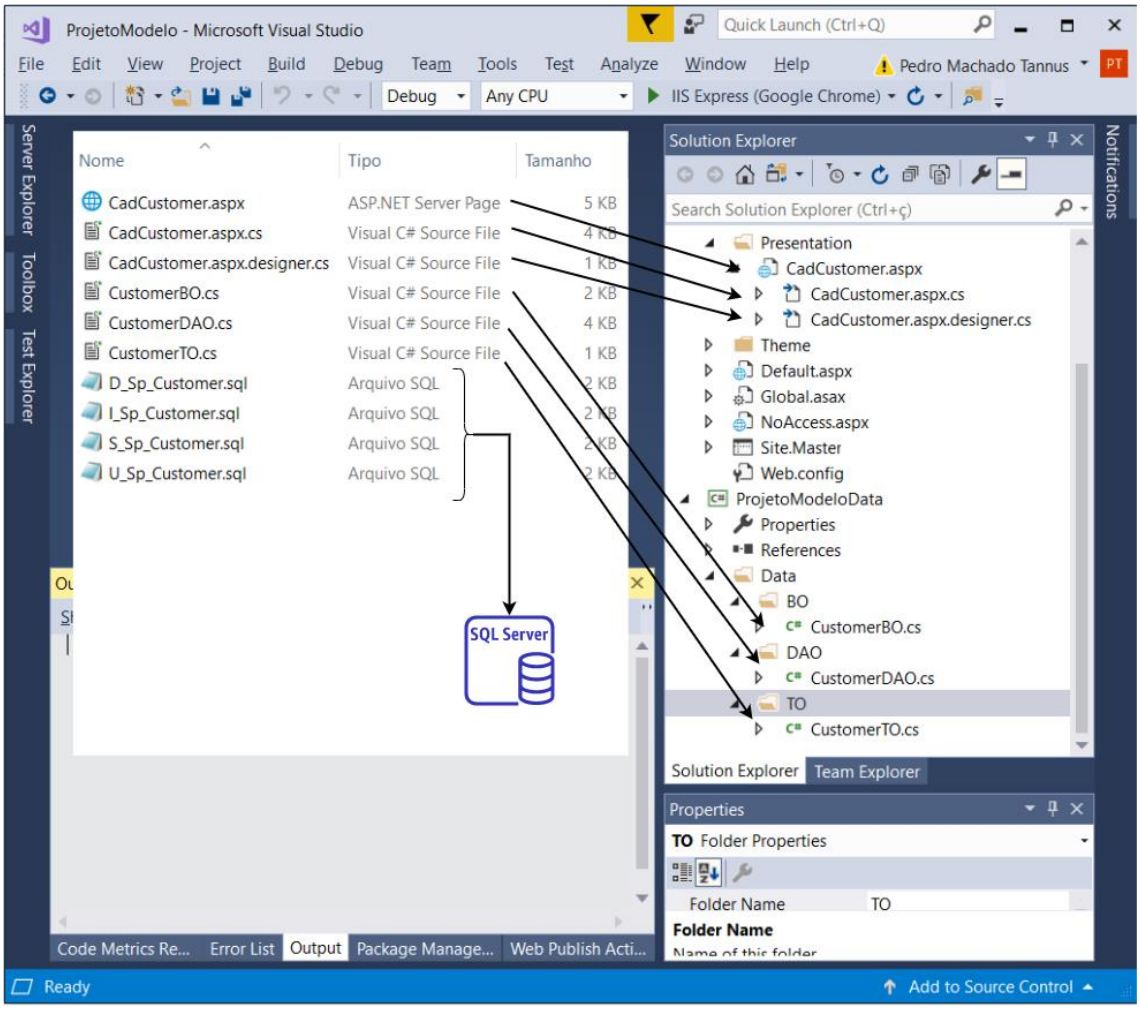

Figura 10: Importação dos artefatos

Com os arquivos devidamente dentro da solução e os scripts SQL executados adequadamente no banco de dados, pode-se testar o resultado. A Figura 11 ilustra o resultado compilado no navegador. Esta primeira interface gráfica tem como objetivo listar os registros do banco de dados e fornecer mecanismos de persistência.

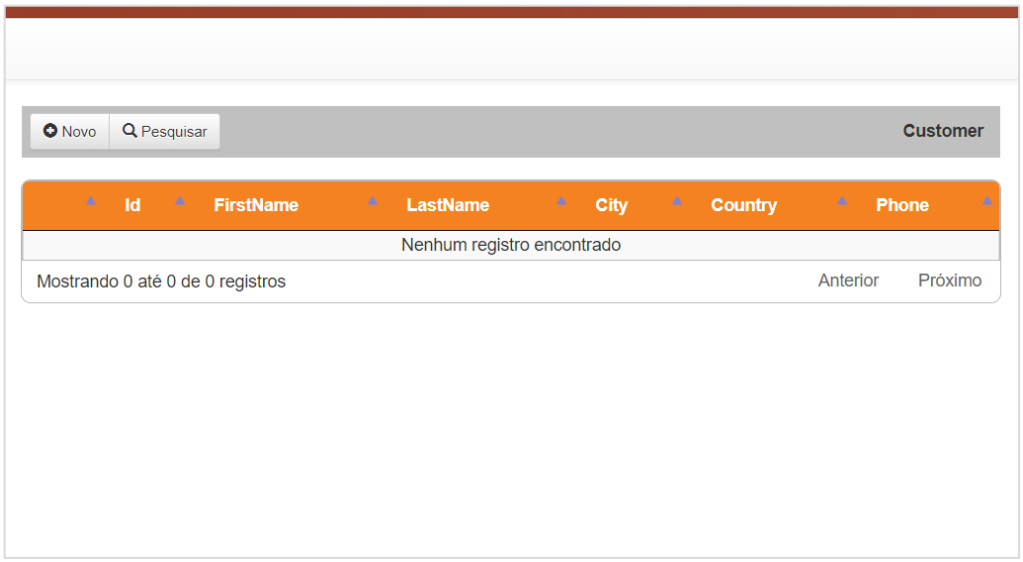

Figura 11: Primeira execução

O *template* “pagina_aspx.txt”, apresenta uma função que abre uma janela para inclusão de um novo registro, com todos os campos tendo sido criados dinamicamente com base nos campos da tabela selecionada conforme ilustrado na Figura 12.

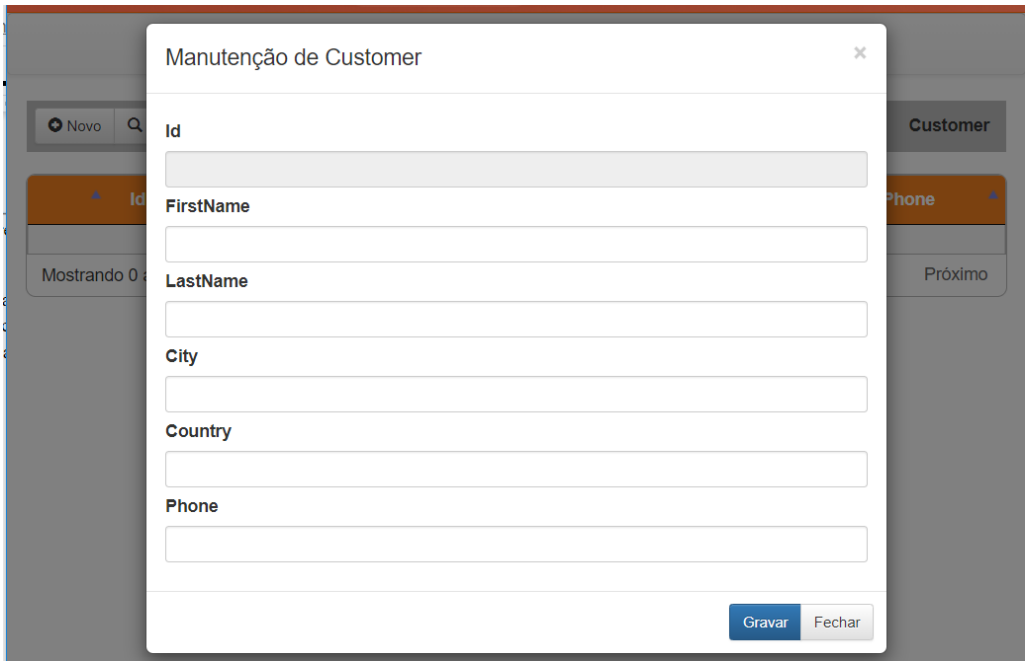

Figura 12: Primeira execução, janela de inclusão e edição

4.7 Modelo de classe Passiva, ajustes

Com o intuito de aproveitar todos os artefatos que foram criados e visto que ainda se faz necessário algumas modificações para atender os requisitos e objetivos do software na qual a ferramenta CASE geradora de código fonte automatizado está beneficiando, aqui será demonstrado o conceito para implementar uma interface gráfica mais amigável. A

Figura 13 mostra a aplicação desta técnica.

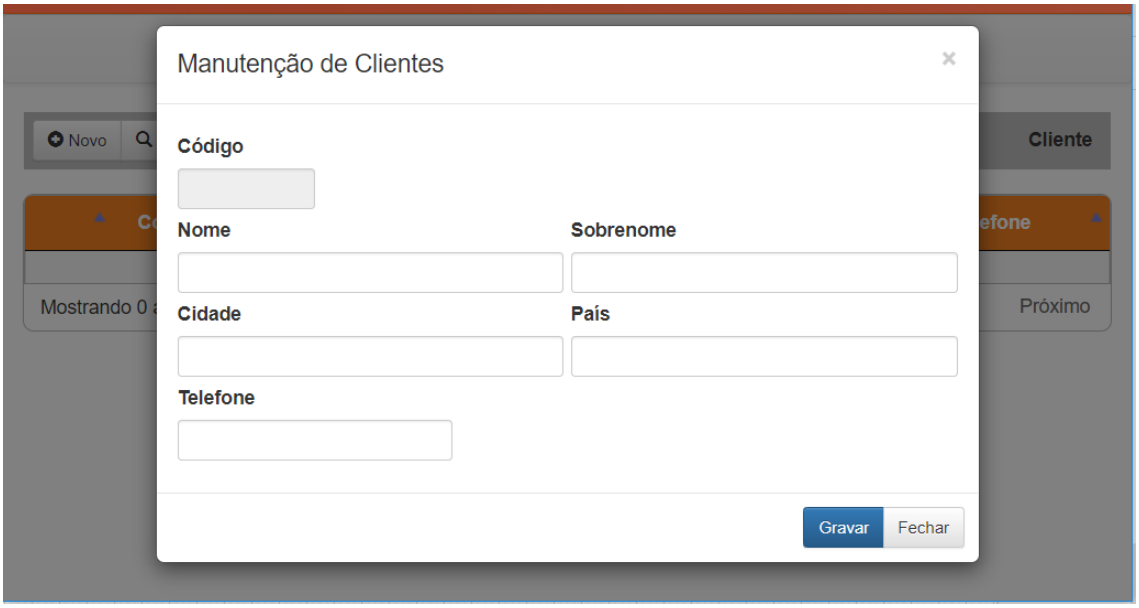

Figura 13: Ajustes nos arquivos

4.8 Análise de resultados

Nesta seção, são apresentados e analisados os resultados encontrados a partir do estudo de caso da ferramenta CASE geradora de código fonte automatizado de classe passiva, a qual busca auxiliar de maneira efetiva equipes de desenvolvimento de pequenas e médias empresas.

Esta análise pautou-se em pontos como tempo de execução, custos envolvidos, dependência da ferramenta, suporte e linguagens, investigando assim as vantagens e desvantagens da ferramenta aqui desenvolvida.

Com relação ao tempo de desenvolvimento tradicional, a ferramenta mostrou-se eficaz, porque 614 linhas de códigos foram geradas em 2 segundos, o que de maneira tradicional seria um tempo muito maior. O Genexus fez o mesmo procedimento e disponibilizou o código para download depois de 1:13 minutos. Neste sentido, o desenvolvedor ganha além de tempo de execução, uma padronização do código fonte.

Em relação às ferramentas similares pesquisadas neste trabalho, como o Genexus, elas demandam um alto custo de investimento, além de processos grandes e complexos, inúmeros cadastros e associações, e muitas vezes, depende de ambientes articulados à nuvem, requerendo assim conectividade em 100% do uso. O que a ferramenta CASE deste trabalho tem como vantagem é a simplicidade de funcionamento, podendo operar sem a necessidade de conectividade com a internet. Por outro lado, não traz consigo uma equipe de suporte para sustentação e operação, nem a possibilidade de gerar artefatos para outras tecnologias.

Outra vantagem analisada é que o código fonte gerado nela é independente dela, oportunizando ao desenvolvedor atuar nas adequações e futuras implementações dispensando o retorno à ferramenta como acontece com ferramentas de classe ativa em que se faz necessário o ajuste no *template* e um novo processo para geração e importação

do código fonte gerando assim um retrabalho.

O uso da ferramenta não gera dependência dentro do projeto caso a equipe de desenvolvimento queira parar de usar, pois existem ferramentas de classe ativa que geram dependência relevante de seu uso durante todo o processo de desenvolvimento, como é o caso de ferramentas de classe ativa.

É preciso considerar também que, o emprego de uma ferramenta CASE que usualmente demanda um custo de instalação e/ou periódico, em uma equipe de desenvolvimento, normalmente oferece um serviço de suporte ao desenvolvimento o que não é o caso desta ferramenta, pois após a instalação, a adequação fica a cargo do desenvolvedor. Desta mesma forma, outros *templates* padrões e inclusive a geração para outras linguagens, podem ser ofertadas, entre outros recursos. A proposta deste trabalho é para um único grupo de *templates* e especificamente para C# e *AspNet*, sendo essa uma clara desvantagem.

Nesse sentido, demonstrou-se a execução de um projeto modelo como forma de ilustrar o resultado dos artefatos gerados e a possibilidade de adequação após a importação, característica de um gerador de classe passiva.

5. Conclusão

Sabe-se que o mundo vem demandando soluções cada vez mais ágeis e com códigos cada vez mais refinados e padronizados, neste contexto, diante dos resultados obtidos neste trabalho, é possível concluir que a ferramenta desenvolvida e aplicada no estudo de caso, se mostra como uma alternativa viável para auxiliar no processo de desenvolvimento de software, atendendo às demandas contemporâneas.

A partir dos resultados deste trabalho, é possível concluir que, apesar da ferramenta apresentar como desvantagens não ser subsidiada por um suporte ao desenvolvedor e não atender outras linguagens além de C# e *AspNet*, as suas vantagens no que se referem a custos, tempo de execução de projetos, independência e padronização, se mostra viável para emprego em projetos em empresas de pequeno e médio porte.

Durante o desenvolvimento do trabalho, ficou claro que, a criação da ferramenta CASE geradora de código fonte automatizado, para além das vantagens citadas acima, ainda apresenta uma mecânica intuitiva com uma única interface gráfica sem a necessidade de alimentar cadastros e fazer associações para utilizá-la.

Por ser uma ferramenta de geração de código fonte de classe passiva, o código gerado pode ser incorporado e adequado, sem a necessidade de retornar a ela. Isso, significa maior flexibilidade e independência da equipe de desenvolvimento em relação à ferramenta CASE.

Com esses pontos apresentados, pode-se considerar que a ferramenta CASE demonstra uma redução do tempo de desenvolvimento e de custo na produção de um software, assim como no processo de manutenção. Uma vez que *template* esteja bem elaborado, a ferramenta irá garantir uma padronização de código fonte e também diminuir

os passíveis erros humanos, muito comum no desenvolvimento tradicional. No entanto, se esta "matéria prima", o *template*, for mal elaborado ou adequado, a qualidade gerada será equivalente.

É preciso considerar também que, o emprego de uma ferramenta CASE que usualmente demanda um custo de instalação e/ou periódico, em uma equipe de desenvolvimento, normalmente oferece um serviço de suporte ao desenvolvimento, o que não é o caso desta ferramenta, pois após a instalação, a adequação fica a cargo do desenvolvedor. Desta mesma forma, outros *templates* padrões e inclusive a geração para outras linguagens, podem ser ofertadas, entre outros recursos. A proposta deste trabalho é para um único grupo de *templates* e especificamente para C# e *AspNet*, sendo essa uma clara desvantagem.

Um trabalho similar a este é apresentado por Monteiro, Lima e Stork (2016), em que os autores apresentam um gerador de códigos para o desenvolvimento de sistemas Web a partir de uma modelagem entidade-relacionamento, no entanto, a ferramenta desenvolvida por eles geram código fonte para linguagem Java. Dessa forma, é necessário avaliar de uma forma mais sistemática a implementação no dia a dia de uma empresa durante o desenvolvimento de um projeto, ficando esse estudo como sugestão para próximos trabalhos. Ainda como sugestão de trabalhos futuros, é necessário avaliar, de uma forma mais sistemática, a implementação no dia a dia de uma empresa, durante o desenvolvimento de um projeto de software, ficando esse estudo, como sugestão para próximos trabalhos.

Outra sugestão que seria interessante ser avaliada, é a realização de um trabalho que fizesse a exposição dos métodos trabalhados para serem consumidos por sistemas externos, possibilitando o uso da mecânica aqui apresentada para consumo de outros softwares, trazendo a possibilidade de uma integração entre sistemas. Ainda como outra proposta, fazer o estudo de mecanismos que possibilitem a geração também para a tecnologia Java.

6. Referências Bibliográficas

DENNIS M. V.; Richard B. K. Software templates. IEEE Computer Society Press, Washington, DC, USA, 55–60. 1985.

FERRARI, F. C.; GENARI, J.O. S. Times de alto desempenho no contexto das metodologias Scrum e Kanban, T.I.S. São Carlos, v. 4, n. 3. p. 200-2008, 2015.

FISHER, A. S. CASE: Utilização de ferramentas para desenvolvimento de software. Rio de Janeiro: Campus, 1990. 264 p.

GENEXUS. Disponível em: <https://www.genexus.com/pt/>. Acesso em: Nov. 2020

HERRINGTON, J. Code generation in action. Greenwich: Manning, 2003.

MARTIN, J. Princípios de análise e projeto baseados em objetos. Rio de Janeiro : Campus, 1994.

MARTIN, J; ODELL, J. Análise e projetos orientados a objetos. Tradução José Carlos Barbosa dos Santos. São Paulo: Makron Books, 1995.

MAXIMIANO, A. C. A. Administração de projetos: como transformar ideias em resultados. 2a. ed. São Paulo: Atlas, 2006.

MICHAELLIS. Dicionário de língua portuguesa. Editora Melhoramentos. Disponível em: <https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/automatiza%C3%A7%C3%A3o/> Acesso em: Ago. 2020.

MONTEIRO, C. M., et.al. Gerador de códigos para o desenvolvimento de aplicações Web a partir da modelagem entidade-relacionamento. Belo Horizonte: CEFET-MG, 2016.

MOREIRA, E.; MRACK, P. Sistemas Dinâmicos Baseados em Metamodelos. In II Workshop de Computação e Gestão da Informação (WCOMPI2003), 2, 2003, Santa Cruz do Sul. Disponível em: <http://3layer.com.br/confluence/download /attachments/3571742/sistemasDinamicosEmTempoDeExecucao.pdf>. Acesso em: 10 Abr, 2020.

NASCIMENTO, J. B. Metodologias de desenvolvimento de sistemas. São Paulo: Érica, 1993.

PHILLIPS, J. Gerência de projetos de tecnologia da informação. 6a. ed. Rio de Janeiro: Elsevier, 2003.

REZENDE, D. A. Engenharia de software e sistemas de informação. 3.ed. Rio de Janeiro: Brasport, 2005.

SAUTER, V., CASE (Computer-Aided Software Engineering), 2009, disponível em: <http://www.umsl.edu/~sauterv/analysis/6840_f09_papers/Sudasong/> Acesso em Mar, 2020.

SCHWABER, Ken; BEEDLE, Mike. Agile Software Development with SCRUM. Prentice Hall, 2002.

SOMMERVILLE, I. Engenharia de software. 8. ed. São Paulo: Pearson Addison-Wesley, 2007.

TAIICHI,O. *O Sistema Toyota de Produção: além da produção em larga escala*, Bookman, Porto Alegre, 1997.