GERAÇÃO AUTOMÁTICA DE CÓDIGO EM C# POR MEIO DE DIAGRAMA DE UML

Wellington Henrik Martins de Oliveira Costa[1], Luciano Antunes[2]

Resumo: No desenvolvimento de software há muitos procedimentos, sendo um destes a criação de tela e os métodos de inserir, editar e excluir uma aplicação, onde o tempo que se leva para tais tarefas pode tornar-se mais demorado que o tempo previsto para a entrega. Esse artigo apresenta uma proposta de solução a partir da prototipagem via UML, com intuito de automatizar o desenvolvimento inicial de um projeto, otimizando o tempo investido no produto para enfoque em outras frentes do mesmo. Onde os resultados obtidos foram os métodos Create, Read , Update, Delete (CRUD) a partir do diagrama de classe UML.

Palavras chave: UML. Diagrama. Automatização. CRUD. Estrutura.

Abstract: In software development there are many procedures, one of which is the creation of a screen and the methods of inserting, editing and deleting within an application, where the time taken for such tasks can become longer than the expected delivery time. This article presents a proposal for a solution based on prototyping via UML, with the aim of automating the initial development of a project, optimizing the time invested in the product to focus on other fronts of the same. Where the results obtained were the Create, Read, Update, Delete (CRUD) methods from the UML class diagram.

Keywords: UML. Diagram. Automation. CRUD. Structure.

1 INTRODUÇÃO

Ao longo dos anos, aumentou a demanda sobre o conhecimento de tecnologias que contribuem para o ecossistema digital existente hoje. Linguagens de programação, Rede de computadores, Sistemas de Gerenciamento de Banco de

[1] Acadêmico do Curso de Ciência da Computação, Unesc. E-mail: wellingtonhenrik13@gmaill.com
[2] Professor do Curso de Ciência da Computação, Unesc. Email: luciano@unesc.net

Dados e Arquitetura de Software são exemplos de temas cada vez mais enfatizados pelo mercado tecnológico, assumindo papéis importantes para o mundo digital. Tal crescimento requer refinamento nos fundamentos para a utilização dessas ferramentas e conceitos, para melhor atribuí-las ao seu exercício.

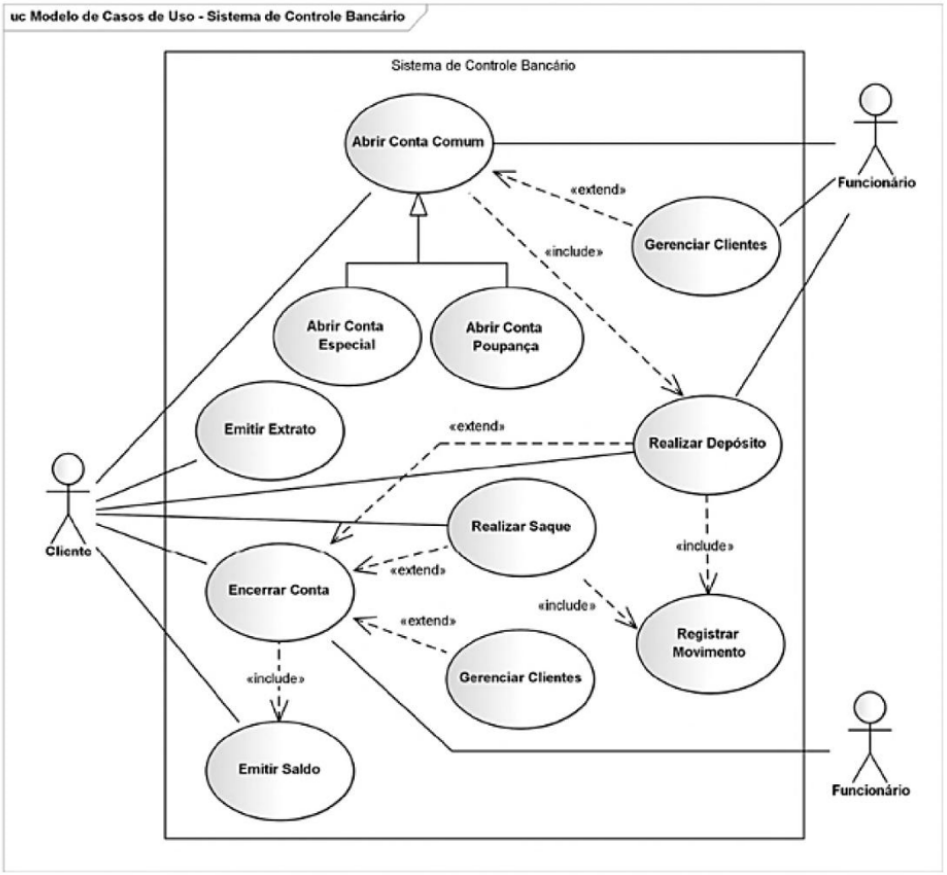
Dentre os tópicos mencionados, surgem diversos recursos que podem ser utilizados para acompanhar sua aplicação, como Designer e projetos, que auxiliam a composição visual da estrutura do software para melhor apuração de lógica e conceituação de resultados. O designer de projeto pode ser por meio de diagrama Unified Modeling Language (UML), criado em 1996, e é a ser uma linguagem para elaboração da estrutura de projeto de software, com o intuito de contribuir e auxiliar a organização do projeto por meio de diagramas estruturais e comportamentais. Para FOWLER (2007), UML “é uma família de notações gráficas, apoiada por um metamodelo único, que ajuda na descrição e no projeto de sistemas de software, particularmente daqueles construídos utilizando a abordagem orientada a objetos (OO)”.

A aplicação da UML auxilia no desenvolvimento de diversos sistemas atualmente. Sua criação surgiu da junção de três conhecedores de modelagem orientada a objetos, onde aplicaram seus conhecimentos para o aperfeiçoamento desta técnica. Desenvolvida por Grady Booch, James Rumbaugh e Ivar Jacobson, profissionais altamente qualificados em modelagem orientada a objetos, a UML é a junção de três metodologias criadas pelos mesmos, utilizando de melhorias nos conceitos da linguagem (BARROS, 1998).

De acordo com Booch, Rumbaugh e Jacobson, a UML possui definição em três pilares, sendo Visualização (modelagem textual e gráfica, facilitando a comunicação visual), Especificação (criação específica de protótipo) e Documentação (coleta de requisitos documentada). A partir da definição em três pilares, percebe-se a influência da UML para resultados positivos com sua utilização. Ao “Visualizar” o diagrama, compreende-se as fases da prototipação, com duas divisões gráficas. Na “Especificação”, apura-se as particularidades de cada parte do protótipo, para definição de regras e especificação de resultados. Enfim, com a “Documentação”, têm-se a distribuição escrita (e cria-se histórico) para expressar as aderências do projeto, a fim de sua execução. (Booch, Rumbaugh e Jacobson, 2006, apud MARGARIDA, 2018).

A UML se enquadra na prototipação das tabelas e fluxos que serão usados para compor um sistema. Essa prototipação também é conhecida como diagrama. A Figura 1 demonstra exemplo de diagrama de caso e uso para a criação de uma conta bancária, de GUEDES:

Figura 1 - Exemplo de diagrama de casos de uso.



Fonte: GUEDES, 2018

Entende-se que a visualização antecipada das compreensões do sistema facilita a interação do desenvolvedor com a execução do trabalho, devido a distribuição visual das etapas a serem executadas para alcançar os objetivos do projeto. Para que a utilização da UML seja efetiva, faz-se uso dos recursos da Arquitetura de Software, porém na etapa de planejamento, também conhecida por Projeto de Arquitetura. MARGARIDA (2018) explica a importância dessa etapa e suas influências no decorrer do projeto, indicando o Projeto de Arquitetura como uma fase de organização de estrutura geral, a fim de pontuar os principais elementos e suas correlações.

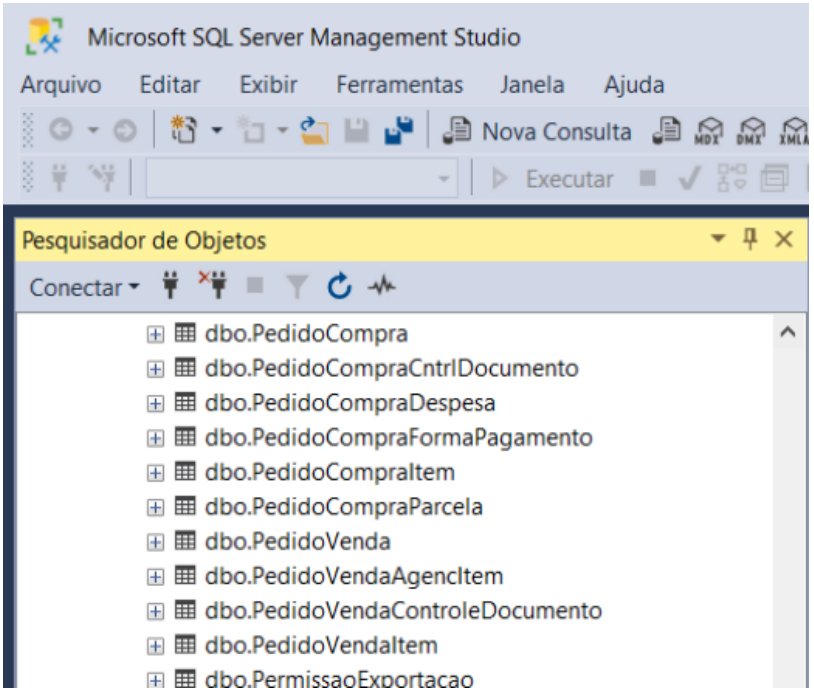
A arquitetura de software divide a composição do sistema em estruturas, nas quais norteiam as divisões de um software em camadas. Um exemplo desta divisão, é o MVC - Model, View e Controller, que são as partições de uma estrutura maior, onde cada parte é responsável por uma fase do funcionamento geral do sistema. Para LUCIANO e ALVES (2011), as definições e funcionalidades do MVC são:

- Modelo (Model) – mantém o estado da aplicação, é mais que uma classe para armazenar dados, nele deve estar todas as regras de negócios e também comunicar-se com o banco de dados se necessário;
- Visão (View) – especifica exatamente como o modelo deve ser apresentado. É a interface do usuário. A visão é dinâmica se adequando a qualquer modificação do modelo;
- Controlador (Controller) – traduz as interações do usuário com a visão, mapeando-as para tarefas que o modelo irá realizar.

Para que a estrutura funcione, ela estará totalmente interligada à estruturação de Banco de Dados e seus relacionamentos, onde o banco será responsável por armazenar os dados salvos da aplicação. O Banco de Dados é um conjunto de dados estruturados e divididos para compor um esquema de informações, podendo estar dispostos em tabelas relacionadas ou não relacionadas. Para PRESCOTT (2015), pode-se definir banco de dados como um conjunto de informações arquivadas dentro de um computador, em meios de fácil manipulação e gestão. Para que o banco de dados seja manipulado, utiliza-se de ferramentas como SGBDs que são sistemas de gerenciamento de um banco, à exemplo de Oracle e SQL Server. Esta segunda, é um sistema que gerencia banco de dados relacionais, onde é possível criar tabelas e manipular seus dados, com alterações exclusivas e listagem.

A Figura 2 demonstra um exemplo de relação de tabelas criadas e gerenciadas via SQL Server:

Figura 2 - Exemplo de tabelas no SQL Server.



Fonte: Elaborado pelo autor.

A Figura 2 esboça exemplos de estrutura de tabelas criadas no SQL Server, e sua visualização. Na mesma, é possível identificar a lista de modelo gerados nesse banco de dados, em forma padronizada nas nomenclaturas.

Assim como SQL Server, há outros tipos de SGBDs, cada uma podendo conter abordagens, sintaxes e interfaces distintas. Em questão de programação com interface de fácil utilização para o desenvolvedor e maior atratividade na experiência do usuário, onde as linguagens de alto nível são mais comuns no mercado tecnológico, em função de serem mais intuitivas. À exemplo, percebe-se o C# que é uma linguagem de programação, manipulado corriqueiramente via IDE (Integrated Development Environment - Ambiente de desenvolvimento integrado) Visual Studio, sob extensão “.cs”.

As linguagens de programação permitem criar uma estruturação lógica do código, onde são escritos passos de execução para uma aplicação, podendo utilizar de metodologias de desenvolvimento, além dos recursos da linguagem e modelo de estruturação que a aplicação conterá. Sendo uma das IDE’s mais comuns no mercado, o Visual Studio permite a programação em diversas estruturas.

Para o desenvolvimento desta pesquisa foi realizada a análise de trabalhos semelhantes sobre o tema proposto, utilizando outros projetos como base para maior fundamento.

O trabalho de Alairton Dendena (2021) teve como objetivo a criação do código fonte a partir de uma base de dados já existente, utilizando a arquitetura MDA, onde ele extraiu os dados do banco de dados, e os transformou em entidades gerando o código fonte, disponibilizado para o desenvolvedor o CRUD que seria os métodos de Create (criar), Read (ler), Update (atualizar), Delete (deletar).

Outro trabalho que seria o de SPINELLI, onde o objetivo de SPINELLI (2015) foi facilitar a criação dos métodos CRUD e tabelas no banco de dados, onde seu Framework faz a geração de código web em linguagem de programação Java.

Lodemar José Hafemann (2000), buscou apresentar como desenvolver uma ferramenta baseada no modelo de diagrama sequencial utilizando a ferramenta Rational Rose, em que a partir do diagrama a ferramenta gera o código na linguagem Delphi, utilizando as especificações da UML.

Diante disso, o desenvolvimento do protótipo de gerador de código demonstra como é possível, a partir de um diagrama, montar uma aplicação sem regras de negócio e permitindo ao desenvolvedor visualizar o escopo geral do projeto, além de otimizar tempo de projeção e admitir livre acesso ao código fonte.

O objetivo desta pesquisa foi desenvolver funcionalidades em cima de um protótipo Nclass de gerador de código fonte e tabelas no banco de dados, utilizando diagrama de classe UML junto com as linguagens C# e o SGBDs SQL Server, propondo uma alternativa facilitadora de criação de código, de fácil entendimento de modelagem de diagramas.

Com a geração do código fonte, o desenvolvedor terá total acesso para futuras manutenções, além de diminuir o tempo de programação do sistema, tal que, criar-se-á o diagrama e o gerador criará as entidades, métodos e telas para o usuário. Desta forma, poupa-se tempo nesta fase trabalhosa e repetitiva.

2 MATERIAIS E MÉTODOS

Para o desenvolvimento deste projeto, foi utilizado um notebook com as seguintes configurações: processador Intel Core 7° Geração, juntamente a uma memória de 16gb.

Com relação às ferramentas de software, contou-se com sistema operacional Windows 10, IDE Visual Studio 2022, para fazer a implementação do código em C#, responsável por gerar todo o código fonte e sua conexão com o banco de dados, SQL server, utilizado para a persistência de dados.

O protótipo elaborado para demonstrar a utilização do UML foi proposto em um modelo de inserção de dados, com objetivo de permitir ao usuário definir o relacionamento de tabelas conforme a sua necessidade. O modelo foi considerado com os campos e tabelas conforme demonstrado na Tabela 1:

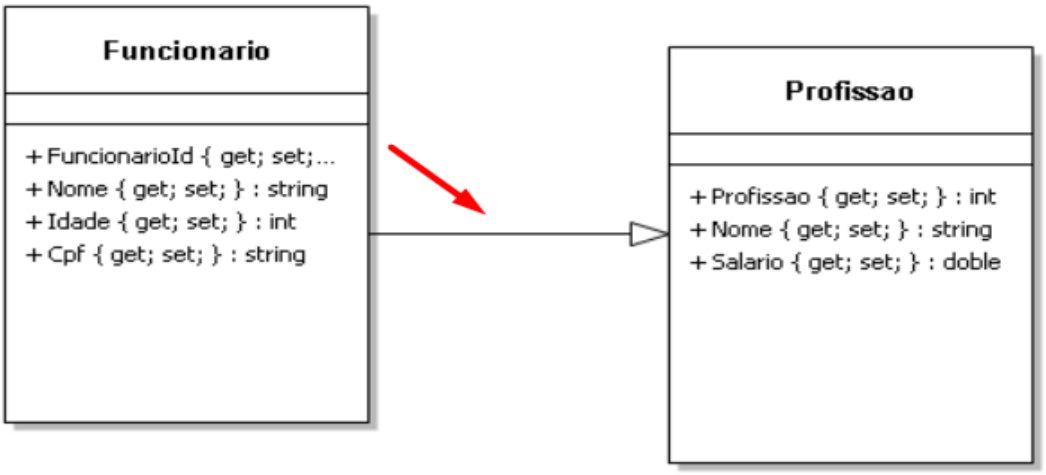
Tabela 1 - Quadro de tabelas disponíveis para relacionamento no protótipo

Tabela	**Campo**	**Tipo campo**	**Descrição**
Funcionário	FuncionarioId	Int	Código gerado automaticamente conforme a inserção de registro
	Nome	String	Campo para inserção do nome do funcionário
	Idade	Int	Campo para inserção da idade do funcionário
	Cpf	String	Campo para inserção do CPF do funcionário
Endereço	EnderecoId	Int	Código gerado automaticamente conforme a inserção de registro
	Rua	String	Campo para inserção da rua do endereço
	Número	String	Campo para inserção do número do endereço
	Bairro	String	Campo para inserção do bairro do endereço
Empresa	EmpresaId	Int	Código gerado automaticamente conforme a inserção de registro
	Nome	String	Campo para inserção do nome da empresa
	QuantidadeFuncionario	Int	Campo para inserção de quantidade de funcionário

Fonte: Elaborado pelo autor.

Uma vez definidos os parâmetros de inserção de acordo com a Tabela 1, é possível identificar os campos que foram tratados no relacionamento junto aos seus respectivos tipos e finalidades. As informações das tabelas foram incluídas e configuradas de acordo no programa NClass, para visualização relacional dos campos, como mostra a Figura 3:

Figura 3 - Tabelas relacionadas utilizando a ferramenta OpenSource NClass

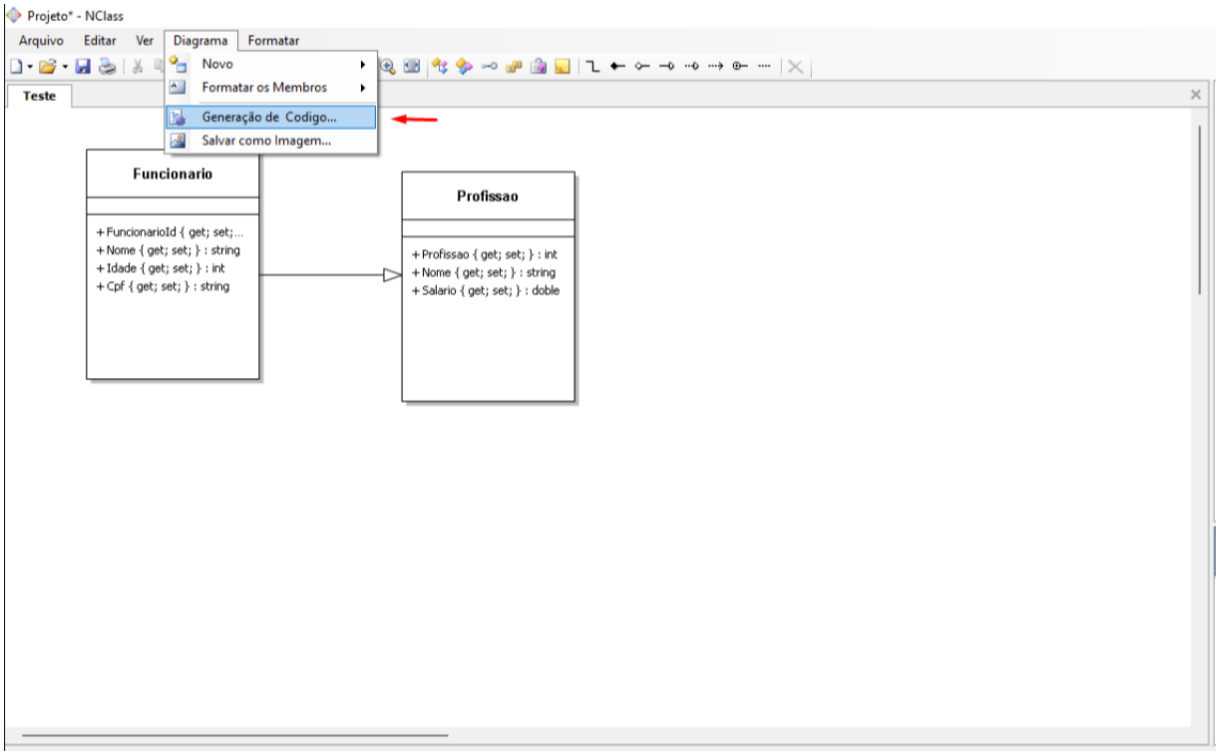


Fonte: Elaborado pelo autor

Na Figura 3, é expresso, em forma de diagrama, o relacionamento entre as tabelas propostas. Por meio da “dependência” demarcada, constata-se o vínculo entre a tabela Funcionário e Profissão. Na mesma, também notam-se as propriedades e tipos de cada tabela, assim, visualiza-se naturalmente configuração esperada para as mesmas, posteriormente utilizadas na geração do código onde obrigatoriamente tem que conter uma propriedade na classe com o mesmo nome da classe mas Id, como mostra a imagem.

Com o diagrama criado, gera-se o código fonte, como exibido na Figura 4, onde na própria ferramenta NClass seguem-se os passos: Menu Superior, Guia “Diagrama” e Opção “Geração de Código Fonte”

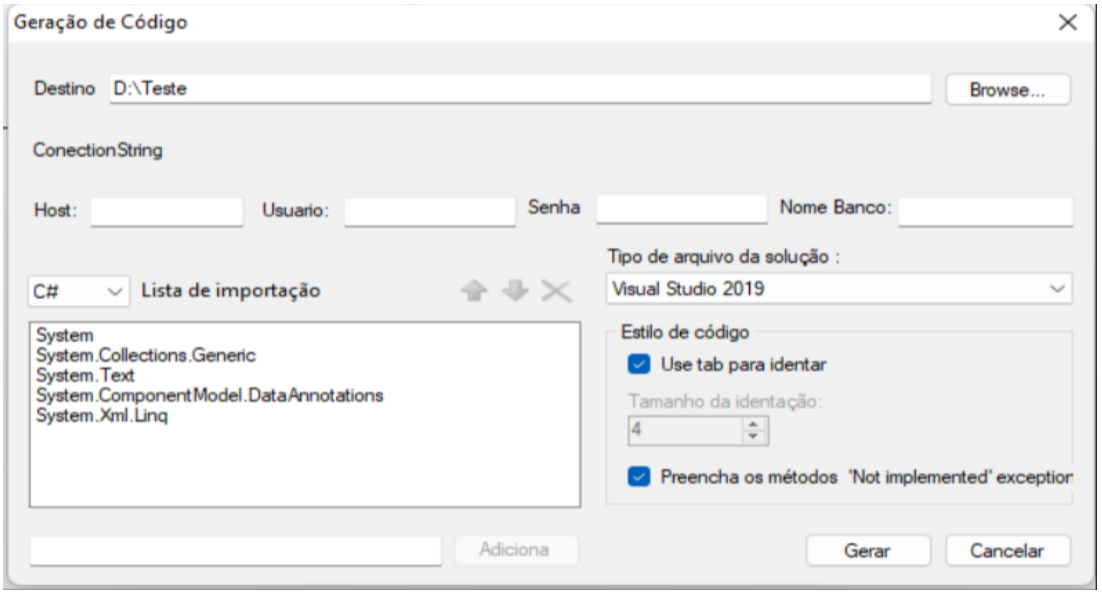
Figura 4 - Demonstração de passo a passo para gerar código fonte



Fonte: Elaborado pelo autor

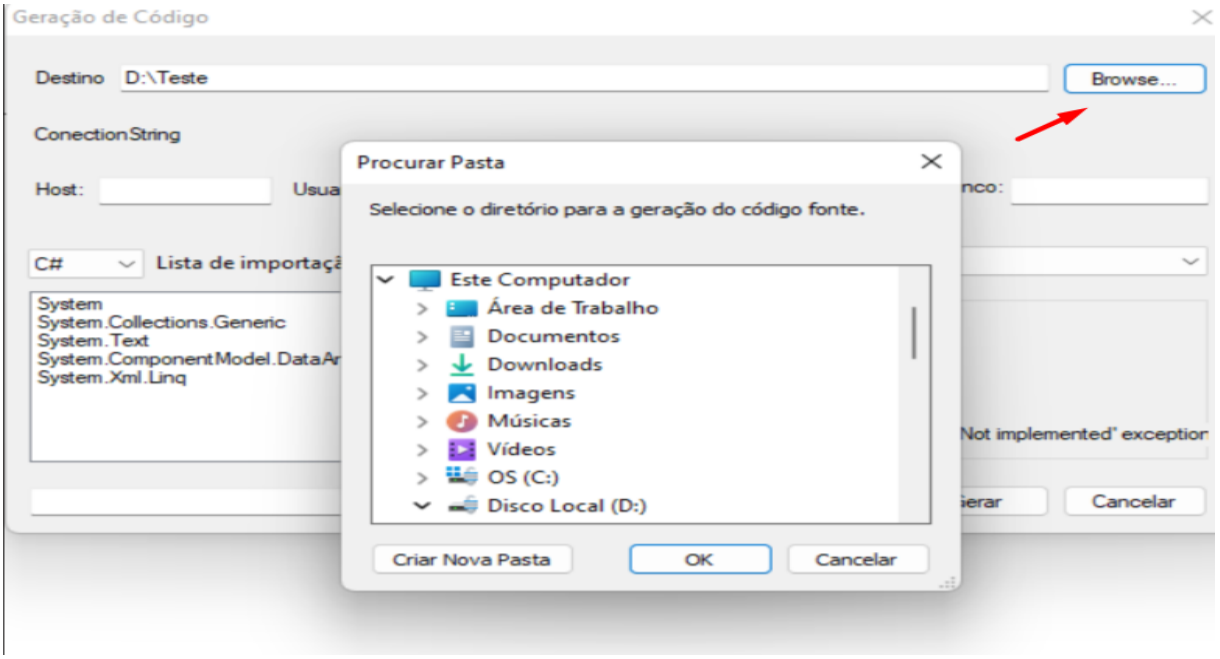
Seguindo os passos exemplificados na Figura 4, foi aberta a tela de acordo com a Figura 5, e como mostrado na Figura 6, deve-se clicar no botão “Browser” .

Figura 5 - Local de criação do código fonte e conexão com banco de dados, via NClass



Fonte: Elaborado pelo autor

Figura 6 - Demonstração de acesso ao diretório do computador

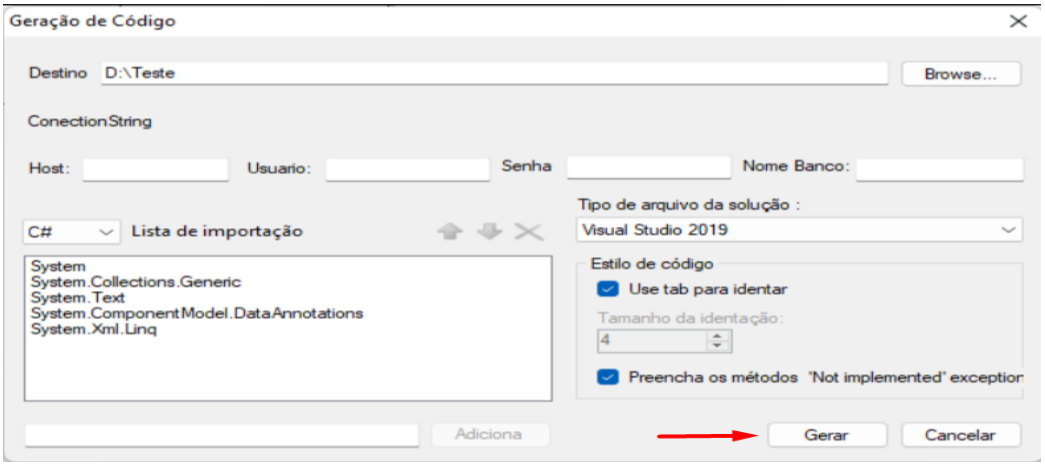


Fonte: Elaborado pelo autor.

Ao selecionar uma “Pasta”, de acordo com exemplo acima, é permitido ao usuário escolher o local em que será gerado o código e sua conexão/criação com o banco de dados, a ser utilizado.

Após clicar em “Ok”, seleciona-se o botão “Gerar”, onde o código será criado no local escolhido (indicação na Figura 7).

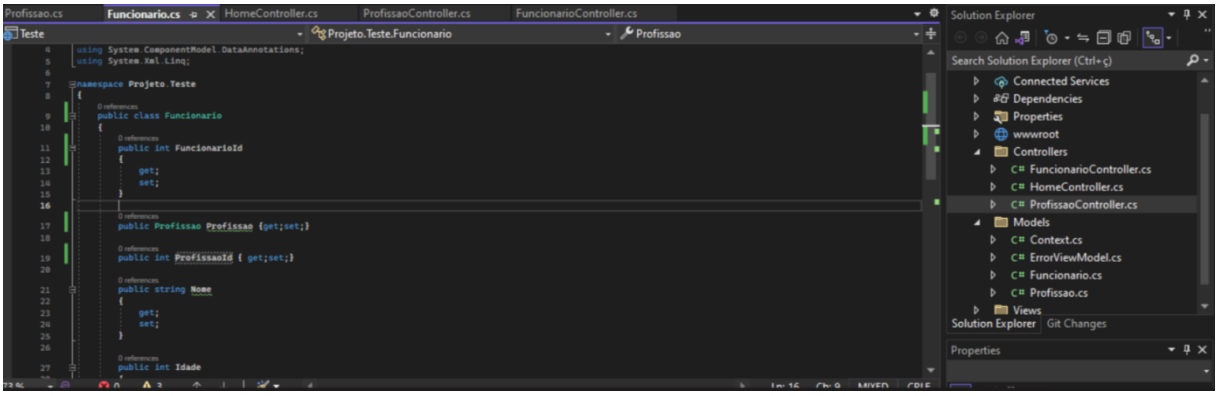
Figura 7 - Indicação de como gerar o código fonte.



Fonte: Elaborado pelo autor

Finalizado o procedimento, apresenta-se a mensagem de “Conclusão com Sucesso”. Acessando a pasta local escolhida para o projeto, observa-se um arquivo de extensão “.csproject”. Ele será o objeto de projeto a ser continuado, como expõe a Figura 8:

Figura 8 - Projeto de código gerado à partir do NClass



Fonte: Elaborado pelo autor

O projeto é aberto conforme Figura 8, que automaticamente será na linguagem de programação C#. No exemplo acima, a IDE utilizada para demonstração é o programa VisualStudio versão 2022.

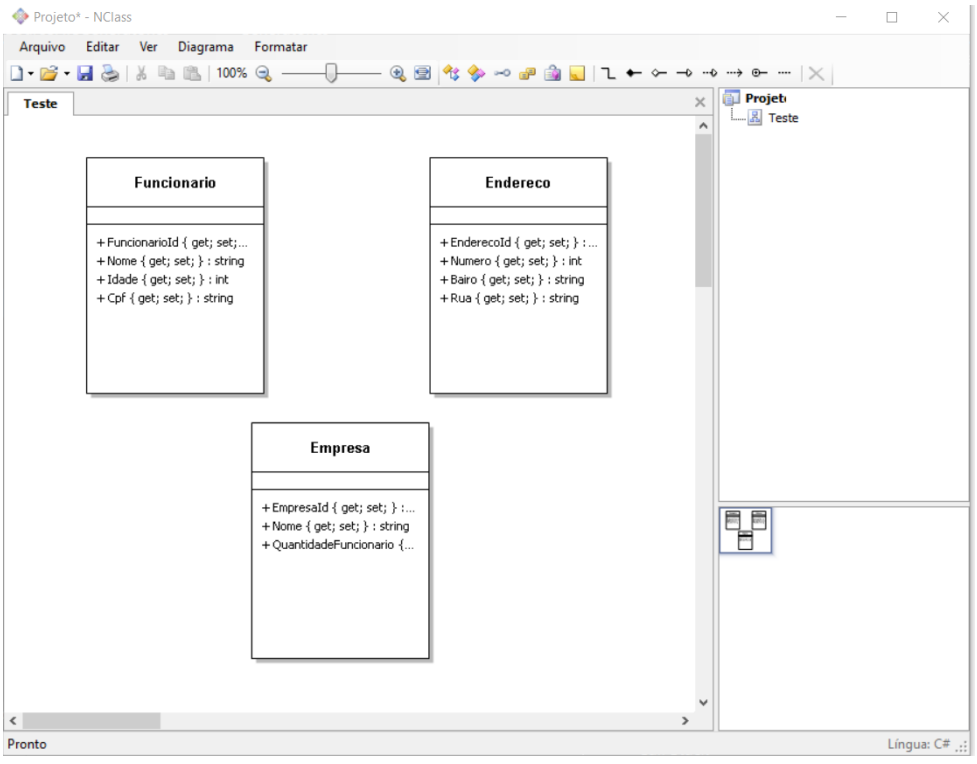
Com o projeto aberto, o usuário poderá seguir com suas implementações e incrementos de regras de negócios, para atendimento das demandas necessárias à sua aplicação.

3 RESULTADOS E DISCUSSÕES

Os resultados deste trabalho foram obtidos por meio de testes que avaliaram o comportamento do protótipo em relação a diversas propriedades e classes, permitindo verificar o relacionamento entre elas.

O primeiro teste foi com três classes que não se relacionavam, para identificar como o protótipo se comportaria. Como mostra a Figura 9, verifica-se a tela com as três classes criadas no diagrama:

Figura 9 - Classes sem relacionamento

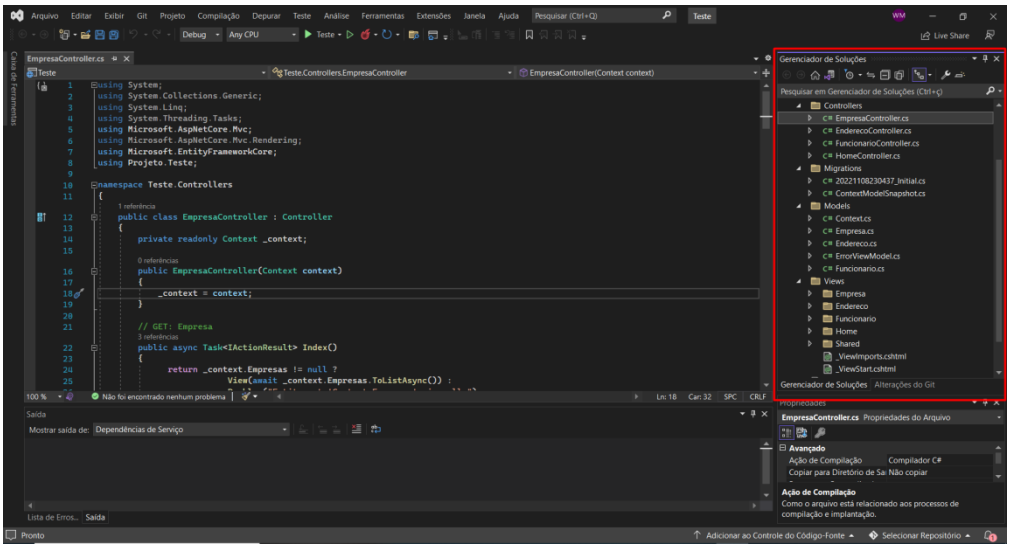


Fonte: Elaborado pelo autor

Após a criação do projeto à partir do diagrama na Figura 9, que representa o NClass (onde foi feita a inserção das tabelas e suas propriedades), a estrutura foi gerada de acordo Figura 10, sendo a representação do código fonte em sua estrutura.

A visualização da estrutura é aferida via IDE Visual Studio, que permite ao usuário reconhecer a linguagem de programação em que a aplicação foi criada. À partir desta, é possível verificar a estrutura dividida com o MVC que seria uma arquitetura de software dividida em Model, Controller e Viiew contendo pastas e classes:

Figura 10 - Estrutura do projeto



Fonte: Elaborado pelo autor.

Conforme representação acima, estrutura foi criada contendo:

Tabela 2 - Detalhamento de estrutura

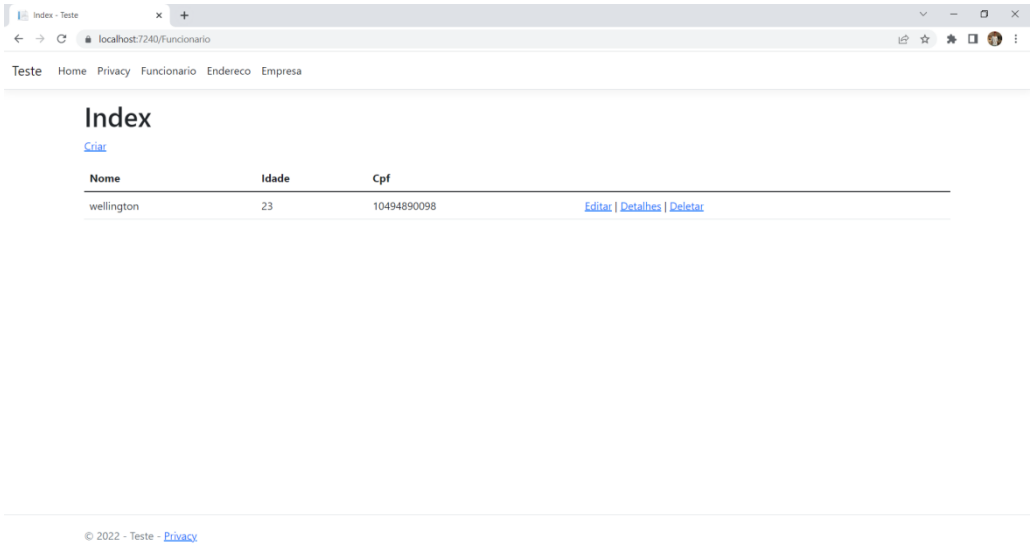
Item de estrutura	**Pasta/Classe**	**Arquivos/Métodos**	**Navegação**
View	Empresa	Index, Inserir, Editar, Deletar	
	Endereco	Index, Inserir, Editar, Deletar	
	Funcionario	Index, Inserir, Editar, Deletar	
	Home	Index	Empresa, Endereco,

			Funcionario
Controller	EmpresaController.cs	Index, Inserir, Editar, Deletar	
	Endereco.cs	Index, Inserir, Editar, Deletar	
	Funcionario.cs	Index, Inserir, Editar, Deletar	
	Home.cs	Index	
Classe Context			
Banco de dados			

Fonte: Elaborado pelo autor

Com a estrutura criada, é possível compilar o projeto clicando no botão “Run” (ou pelo atalho F5). Será aberta a versão executável do projeto, via navegador. Afere-se na Figura 11 a tela aberta à partir do código, no qual contém os campos mapeados na tabela do diagrama, juntamente aos botões Criar, Editar, Detalhes e Deletar, visualizados pela tela Index:

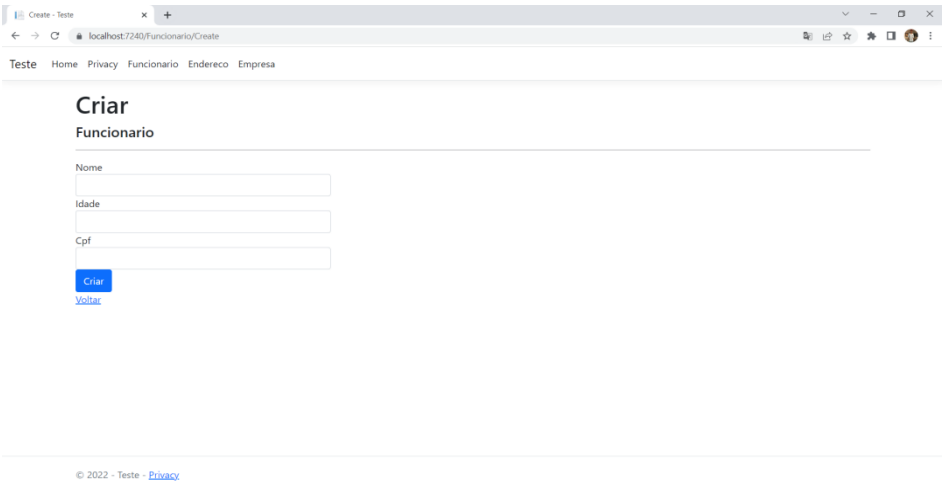
Figura 11 - Tela Inicial de uma classe



Fonte: Elaborado pelo autor

Na tela exposta na Figura 11 simboliza-se o diagrama compilado de acordo com a estrutura gerada a partir dele, também observa-se a listagem dos registros cadastrados utilizando a rotina do botão Criar, evidenciado na Figura 12.

Figura 12 - Tela de Cadastro (botão Criar)



Fonte: Elaborado pelo autor

A Tela de Cadastro (botão Criar) exibe os campos para inserção dos dados, que serão registrados conforme propriedades das classes, sendo: Nome; Idade; CPF; Botão Criar e Botão Voltar.

Ao clicar no botão Criar, posteriormente ao preenchimento dos dados, o mesmos serão criados no banco de dados, ficando registrados para futuras consultas.

Pressionando o botão Voltar, retorna-se para a tela inicial com os registros já salvos no banco, permitindo que o usuário possa visualizá-los por meio da opção Detalhes ou Editar, ou excluir via opção Deletar.

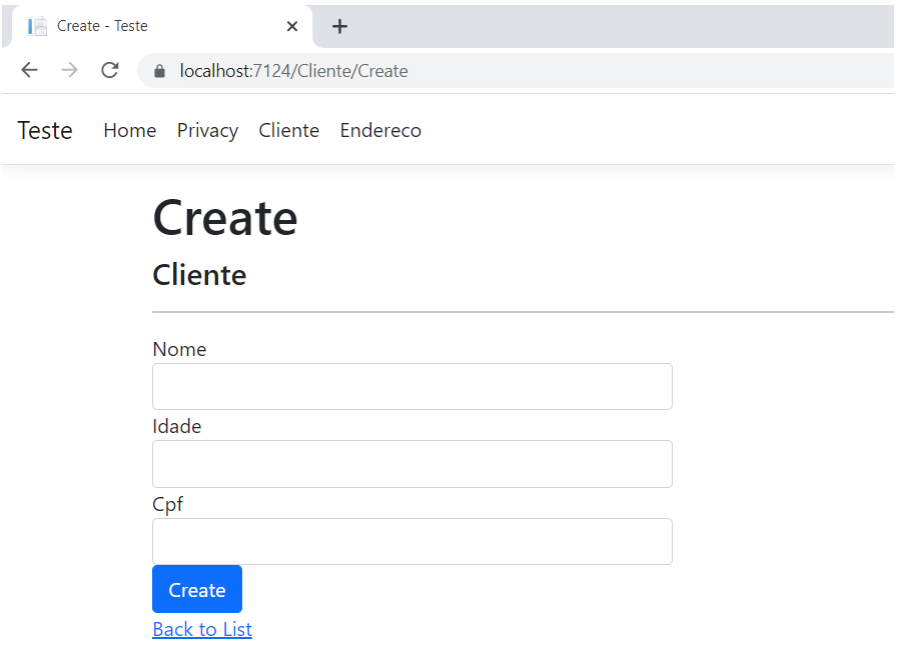
Ao decorrer dos testes, observou-se que na relação de classes, nem todos os relacionamentos ocorreram de acordo com o esperado. Um dos problemas encontrados foi relacionamento entre listas, uma vez que no protótipo não foi possível montar a tela com a lista, para fazer a seleção e salvar em banco. Como demonstrado na Figura 13, a classe foi criada com uma lista, contudo não foi levada para a tela, por não conseguir fazer a identificação do tipo especificado.

Figura 13 - Propriedade lista criada na classe

Fonte: Elaborado pelo autor

Como o protótipo não conseguiu identificar o tipo lista, não foi levado para a tela para fazer o cadastro do mesmo, pois não obedeceu seu tipo de propriedade:

Figura 14 - Erro de listagem: classe sem a lista em tela



Fonte: Elaborado pelo autor

Dessa forma, no cenário de usar lista de N:N (muitos para muitos), não seria possível em função de não estar na tela para realizar a seleção, como mostra a Figura 14.

Ao analisar o método de criação do protótipo proposto em relação ao de Dendena (2021), notam-se diferenças de concepção de projeto à partir da sequência de procedimentos, visto que o autor criou primeiramente o banco de dados com suas tabelas e relacionamentos, onde requer conhecimento prévio do usuário sobre a linguagem SQL, como exemplo de chaves primárias e index.

No protótipo elaborado neste artigo, não obriga o usuário a compreender artifícios técnicos para executar a criação do projeto, visto que em primeira instância foi desenhado o relacionamento em UML, em seguida já foi possível gerar o código fonte, juntamente com o banco de dados. Em vista de agilidade e otimização de tempo, o protótipo elaborado neste artigo permite dinamismo de entrega distinto em relação ao de Dendena, considerando: documentação (com aplicação de diagrama UML separada do código), geração de código e banco de dados (à partir da ferramenta usada).

Outro ponto foi o desenvolvimento do autor Hafemann que utilizou a ferramenta Rational Rose onde depois de ser gerado o sistema o desenvolvedor não tem acesso à fonte do sistema, assim dificultando a manutenção e mudanças.

4 CONCLUSÃO

Esta pesquisa visou demonstrar como toda a estruturação, prévia ao desenvolvimento de um projeto, pode contribuir com a visualização do produto a ser criado, permitindo ao desenvolvedor estabelecer as conexões e fluxos necessários antes de iniciá-lo. Segundo DIAS (2016) Tal visualização estabelece relação com o planejamento de protótipos, a fim de economizar tempo e recursos de desenvolvimento para executar operações iniciais repetitivas do sistema, devido a previsibilidade que a prática expõe.

No modelo proposto, foi utilizado como objeto de estudo a implementação da UML, no qual foi possível identificar as vantagens do recurso por meio da ferramenta NClass. Nela, foi desenhada a estrutura de protótipo, para

relacionamento entre tabelas. No exemplo aferido, não foi implementada a utilização de regras de negócio, focando a proposta à sua aplicação inicial o CRUD

Os objetivos de gerar o sistema com o protótipo foram atingidos, quando se tem diagramas sem relacionamento e com relacionamento um para um. Contudo, não se comportou da maneira que deveria em relacionamentos muitos para muitos, em função da necessidade de uma tabela intermediária. Todavia, devido não ser o objetivo deste trabalho, não deu-se seguimento ao seu desenvolvimento.

Apesar de o objetivo ter sido alcançado, ao explorar o uso do UML em prototipagem de estruturas, o modelo apresentado é funcional para abordagem simples, entretanto pode não ser funcional para cenários de listas, devido sua limitação em relação a tipagem de muitos para muitos. Assim, sugere-se, para pesquisas futuras, a exploração do tema visando finalização e incremento do protótipo, com abrangência de mais tipos de propriedades.

5 REFERÊNCIAS

BALAZS, Tihanyi. **Nclass**. 2010. Disponível em <http://nclass.sourceforge.net/>. Acesso em 18 mai. 2022.

BARROS, Pablo Fernando do Rêgo, **UML Linguagem de Modelagem Unificada**, 1998. Disponível em: <http://www.eribeiro.com.br/pablo/uml/index.html>. Acesso em 01.05.2023.

BENTO, Luiz Henrique Ten Caten. UML: **Um estudo sobre uso em empresas de desenvolvimento de software em São Carlos**, São Carlos, 2020. Acesso em: 20 de Out. De 2021

COIMBRA, Everton. **ASP.NET MVC 5 - Crie Aplicações Web Na Plataforma Microsoft, 2016**. Acesso em: 06 jun. 2022.

DENDENA, Alairton. **Utilização dos conceitos de Low Code e no Code na geração de Web Services com arquitetura MDA**, 2021. Acesso em: 25 de Mai. de 2022

DIAS, Francisco Alberto de Castro e. **Gerador de código para uma Framework C++ a partir de diagramas de estado UML**, 2016. Acesso em: 18 de Out. De 2021

FILHO, Alcides, **Adicionando um controlador a um aplicativo ASP.NET Core MVC com o Visual Studio 2017**, São Paulo, 21 Ago de 2017. Disponível em: <https://coremvc.com.br/adicionando-um-controlador-um-aplicativo-asp-net-core-mvc-com-o-visual-studio-2017/>

FOWLER. Martin, **UML Essencial. Um breve guia para a linguagem padrão de modelagem de objetos, 3 ed., p.25**, São Paulo, Bookman, 2007

GUEDES, Gilleanes T. A., **UML 2 - Uma abordagem prática**, 3ed, São Paulo, 2018.

HAFEMANN. **Protótipo de Gerador de Código Fonte baseado em Diagramas de Seqüências**, 2000. Acesso em: 20 de Out. De 2021

KAMIENSKI, Carlos Alberto. **Introdução ao paradigma de Orientação a Objeto**. CEFET-PB, João Pessoa, 1996.

MARGARIDA, Marina de Oliveira. **Avaliação da aplicabilidade da UML como uma ADL de software**, 2018. Acesso em 01.05.2023

PAPOTTI, Paulo Eduardo. um processo dirigido a modelos para geração de código, São Carlos, 2013. Acesso em: 27 de Out de 2021

PARADA, Abilio et al. **Geração automática de código Android eficiente a partir do modelo uml**, 2014. Disponivel em: <https://www.researchgate.net/publication/236504500_Geracao_Automatica_de_Codigo_Android_Eficiente_a_Partir_de_Modelos_UML>. Acesso em: 05 de Nov. De 2021

PRESCOTT, Preston. **SQL para Iniciantes**, Babelcube 2015.

ROCHA, Kassio. **Entenda o Modelo Entidade Relacionamento para o concurso da PF**, São Paulo. Disponivel em: <https://www.estrategiaconcursos.com.br/blog/modelo-entidade-relacionamento/>

SOMMERVILLE, Ian. **Engenharia de Software.** 9. ed., p.16-522, São Paulo: Pearson Addison Wesley, 2011.

SOUZA, Vitor e Silva; ZUPELI Breno Leite. **Integração de um Gerador de Código ao FrameWeb Editor**, 2018. Acesso em: 25 de Mai. de 2022

SPINELLI, Lucas Pompeo Pontes. **Desenvolvimento de uma ferramenta para geração automática de código aberto em Java Server Faces**, 2015. Disponível em: <https://docplayer.com.br/55592459-Lucas-pompeo-pontes-spinelli-desenvolvimento-de-uma-ferramenta-para-geracao-automatica-de-codigo-aberto-em-java-server-faces.html>. Acesso em: 27 de Out. De 2021