CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC	
(X) PRÉ-PROJETO ( ) PROJETO	ANO/SEMESTRE: 2021/2

SISTEMA GERENCIADOR DE BASE DADOS E DICIONÁRIO

William Mello

Prof. Luciana Pereira de Araújo Kohler – Orientadora

1 INTRODUÇÃO

Em muitos sistemas legados, a manutenibilidade se torna difícil devido a indisponibilidade de documentação adequada e a complexidade dos softwares dificultam ainda mais manter esses sistemas (ZAFAR *et al.*, 2019). A engenharia reversa de software é aplicada a esses sistemas legados para extrair informações úteis ocultas de um baixo nível de abstração para um alto nível de abstração afim de melhorar a manutenibilidade desses sistemas (ZAFAR *et al.*, 2019). Sistemas legados são sistemas críticos que estão em uso a um longo período, que possivelmente foram desenvolvidos com tecnologias consideradas atualmente ultrapassadas e que são peças fundamentais para a organização que o mantem (BARBOSA; CANDIDO, 2017). Devido a essas tecnologias ultrapassadas, compromete o processo interno da organização de readaptação e evolução do mesmo perante as novas mudanças (BARBOSA; CANDIDO, 2017).

A tecnologia evolui constantemente e alguns sistemas precisam evoluir junto para se adaptar as necessidades das organizações (SANTOS; BIANCHINI, 2019). Essas evoluções podem ocorrer devido a mudanças comerciais, técnicas, legislativas, de sistemas operacionais ou necessidade de operar em outras plataformas (SANTOS; BIANCHINI, 2019). Um exemplo é a computação em nuvem, que vem sendo adotada em larga escala entre as organizações, que viram como importante estratégia de negócios obrigando a evoluir seus sistemas legados as estruturas da nuvem (SANTOS; BIANCHINI, 2019).

Santos e Bianchini (2019) descrevem a engenharia reversa como:

> "[...] essencial para as pesquisas que objetivam entender e melhorar as tecnologias existentes, pois é possível que concorrentes realize o estudo de um objeto mesmo que este seja protegido por leis de propriedade intelectual. De forma ampla, engenharia reversa é o processo de entender como algo funciona através da análise de sua estrutura, função e operação permitindo conhecer como um produto foi pensado e desenhado mesmo sem possuir o projeto original.[...] Engenharia reversa pode ser usada para entender como algo funciona, ou como foi feito, mesmo sem possuir a sua documentação [...]." (SANTOS; BIANCHINI, 2019).

O resultado da engenharia reversa é a especificação de um produto que é gerada a partir das análises realizadas no sistema. Com isso permite entender o funcionamento do mesmo, para alterar ou realizar manutenções futuras ou até mesmo acrescentar novas funcionalidades ao software em questão (SANTOS; BIANCHINI, 2019). Essa técnica de engenharia reversa cresceu muito e possibilita o entendimento e melhoria do software, até na descoberta de segredos industriais e comerciais (SANTOS; BIANCHINI, 2019).

A Linguagem de Consulta Estruturada (SQL) é utilizada para administrar Sistemas Gerenciadores de Banco de Dados (SGBD) e foi desenvolvida pela IBM na década de 1970 para gerenciar os bancos de dados internos (VIEIRA, 2020). Com o passar do tempo a SQL foi aceita internacionalmente por todos os SGBDs relacionais para realização de processamento de informações (VIEIRA, 2020). A SQL está dividida em duas partes, Linguagem de Definição de Dados (DDL) e Linguagem de Manipulação de Dados (DML). Na DDL é definido o esquema do banco de dados, estrutura de armazenamento, métodos de acesso, propriedades específicas dos dados e restrições de consistência e integridade do banco de dados. Na DML é possível acessar e realizar nos dados as operações de recuperação, inserção, exclusão e alteração de informações (VIEIRA, 2020).

O SGBD é uma ferramenta que utilizada da linguagem SQL para criar, processar e gerenciar um banco de dados, tal como criar tabelas e estruturas de suporte (VIEIRA, 2020). Uma das finalidades de um SBGD é fornecer uma ferramenta ao usuário para definição de estruturas para armazenar e manipular informações, além de garantir segurança dos dados, evitar falhas de sistema, acessos não autorizados, entre outros (VIEIRA, 2020). Um SGBD é uma aplicação que interage com o usuário e banco de dados fazendo uma ponte entre um e outro, para obter e analisar dados (VIEIRA, 2020). Segundo Vieira (2020), os bancos de dados mais utilizados no mercado são: MySQL, Oracle Database, SQL Server, IBM Db2 e PostgreSQL (VIEIRA, 2020).

A partir dessas informações, este projeto pretender disponibilizar uma ferramenta com interface gráfica Web para auxílio no processo de engenharia de software, por meio do gerenciamento das tabelas, de seus

campos e de seus relacionamentos, utilizando a técnica de consulta na estrutura de um SGBD. Essa ferramenta também permitirá a conversão e atualização automática do SGBD com base nos dados das tabelas que se encontram na interface gráfica Web, importar tabelas de um arquivo de extensão pas através da engenharia reversa e gerar código fonte pas com as tabelas existentes na ferramenta.

1.1 OBJETIVOS

O objetivo deste trabalho é disponibilizar uma ferramenta web que permita a manipulação de tabelas de um banco de dados sem a necessidade da compreensão de sua estrutura e de conhecimento em código SQL.

Os objetivos específicos são:

a) fornecer uma Unit em Pascal conforme a estrutura configurada na interface gráfica;
b) disponibilizar a aplicação web com boa usabilidade seguindo os padrões de usabilidade para web.

2 TRABALHOS CORRELATOS

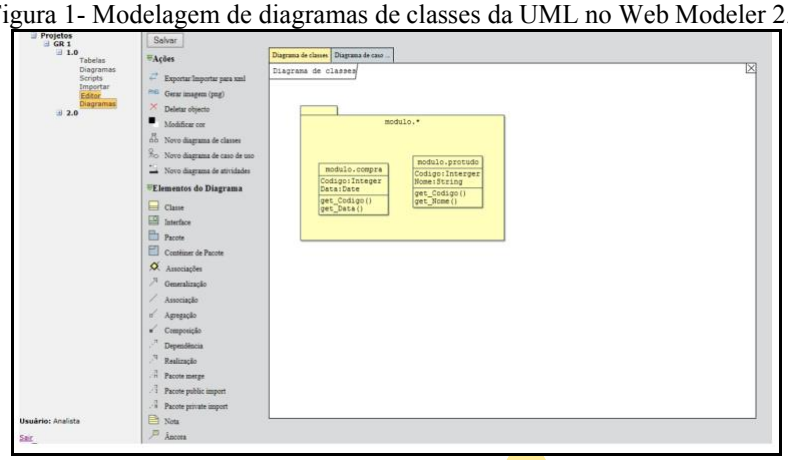
Neste capítulo são apresentados três trabalhos correlatos que possuem características semelhantes a proposta desse trabalho. Os três trabalhos são ferramentas CASE para gerenciamento de banco de dados, afim de facilitar o cotidiano de programadores e analistas de sistema.

2.1 FERRAMENTA WEB PARA MODELAGEM LÓGICA EM PROJETOS DE BANCOS DE DADOS RELACIONAIS

Bugmann (2012) apresenta uma ferramenta Web para modelagem lógica em projetos de banco de dados relacionais, de nome Web Modeler 2.0, com o objetivo de melhorar projetos de banco de dados de grandes sistemas e dar continuidade ao "aplicativo web para definição do modelo lógico no projeto de banco de dados relacional" de Bachmann (2007). A ferramenta foi melhorada para permitir ao usuário criar, gerenciar um banco de dados, independente do SGBD utilizado, e construir "[...] diagramas da UML, casos de uso, classes e atividades." (BUGMANN, 2012, p. 28), controlar projetos de banco de dados e suas versões, com restrição de acesso a usuários em determinados projetos e criar digramas para modelar tabelas, colunas e relacionamentos de forma gráfica. O Web Modeler 2.0 foi desenvolvido em linguagem de programação Java, com arquitetura Model View Controller (MVC) e alguns padrões de projeto, como *Data Access Object* (DAO) e *Command*. De alguns aspectos que foram melhorados no Web Modeler 2.0, pode-se citar a funcionalidade de aproximar, distanciar, imprimir diagramas com geração de scripts para criação/atualização de bancos de dados e adicionar uma rotina que permita realizar engenharia reversa de um banco de dados, capaz de varrer a estrutura do banco de dados e descobrir as tabelas existente (já criadas) no banco de dados em que o usuário está conectado (BUGMANN, 2012).

Na Figura 1tem-se o diagrama de classes carregado na ferramenta Web Modeler 2.0. Nessa ferramenta há disponível as opções de gerenciar tabelas, diagramas, scripts, importar e editor de diagramas. No menu de tabelas, é possível manter as tabelas do SGBD. Na opção de diagramas, é possível consultar as tabelas por meio de diagramas da Unified Modeling Language (UML). Na opção de scripts, é possível gerar os scripts para converter a base de dados. Na opção importar, é possível importar diagramas criados na ferramenta. Por último, o editor de diagramas, que há as ações necessárias para a construção de diagramas de classes, atividades e de casos de uso, como por exemplo, inserir associações, composições, atores, casos de uso, fluxo, atividade final, entre outros.

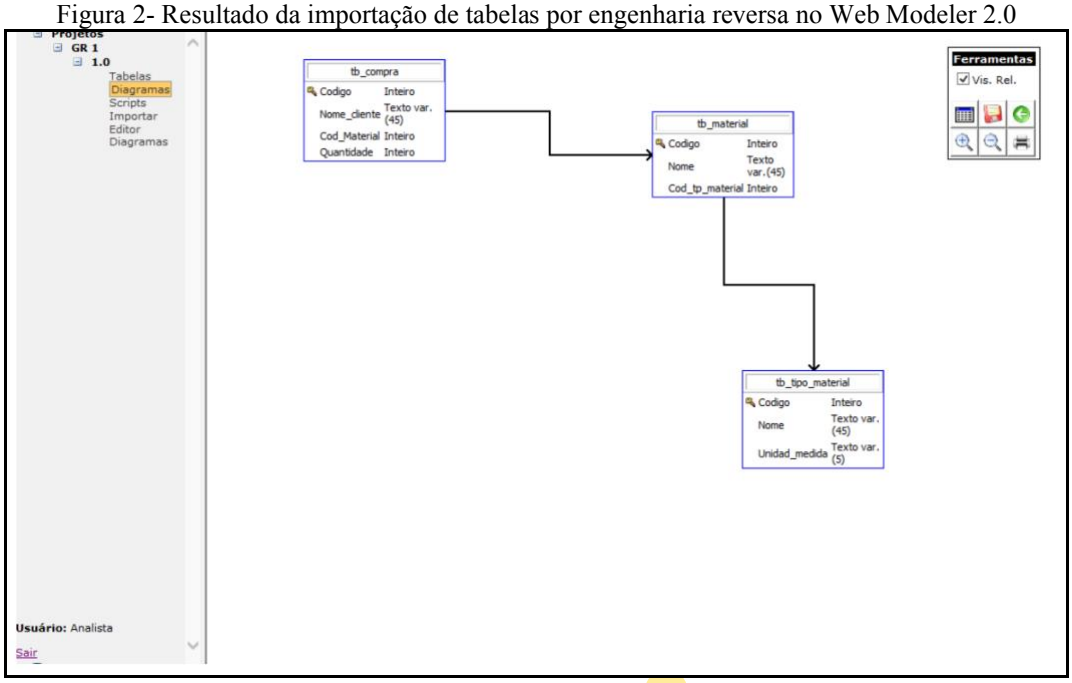
Figura 1- Modelagem de diagramas de classes da UML no Web Modeler 2.0



Fonte: Bugmann (2021, p. 26)

Ao final do trabalho, Bugmann (2012) conclui que os objetivos iniciais foram alcançados e que o aplicativo Web Modeler 2.0 desenvolvido permite que um banco de dados relacional seja modelado, por meio de diagramas, independente do SGBD utilizado. Na Figura 2 pode-se ver o resultado da importação de tabelas por meio da técnica de engenharia reversa do banco de dados utilizado por Bugmann (2012), ou seja, o Web Modeler 2.0 identificou as tabelas tb_compra, tb_material e tb_tipo_material no SGBD, importou elas para o Web Modeler 2.0 (por meio da engenharia reversa) e gerou uma representação gráfica na ferramenta para visualizar o resultado e permitir alterá-las (BUGMANN, 2021).

Figura 2- Resultado da importação de tabelas por engenharia reversa no Web Modeler 2.0



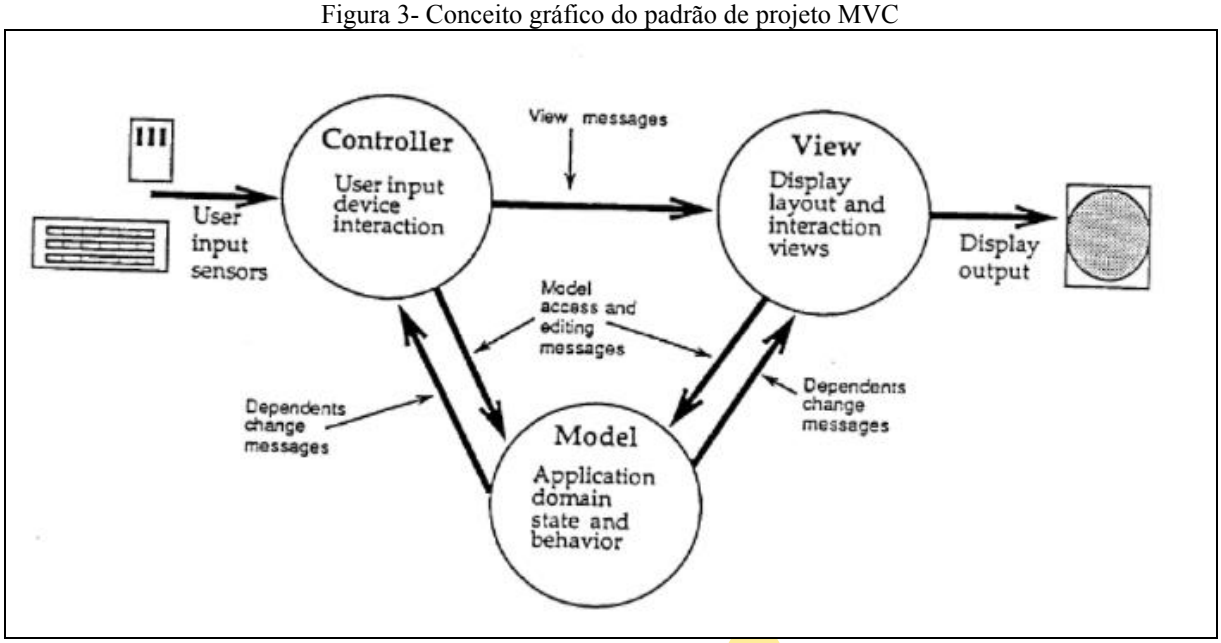
Fonte: Bugmann (2021, p. 62)

2.2 UMA FERRAMENTA DE SOFTWARE PARA GERAR UMA ARQUITETURA MODEL-VIEW-CONTROLLER BASEADA NO MODELO ENTIDADE-RELACIONAMENTO

Soto *et al.* (2020) apresentam uma ferramenta CASE capaz de gerar código Model-View-Controller (MVC) com base em um diagrama de Entidade-Relacionamento (ER) e um *script* de banco de dados correspondente ao diagrama traduzido. A ferramenta é desenvolvida na linguagem de programação Angular e tem por objetivo aprimorar o aprendizado do padrão de projeto MVC, promovendo boas práticas de programação e a reutilização de código (SOTO *et al.*, 2020).

Na Figura 3, Krasner e Pope (1988) explicam como funciona o padrão de projeto MVC e suas camadas, sendo: o modelo (*Model*) tem a função de assumir a parte do sistema que gerencia todas as tarefas relacionadas aos dados, como por exemplo, validação e persistência de dados ligadas ao banco de dados; a visão (*View*) possui a função de interagir com o usuário por meio de interfaces gráficas gerenciáveis, ou seja, formulários, botões e todos os outros elementos gráficos da aplicação (parte visual); controlador (*Controller*),tem a responsabilidade de cuidar dos eventos disparados pela camada de visão ou acionados por um processo do sistema, ou seja, gerencia os eventos entre a camada de visão e a camada de modelo (KRASNER; POPE, 1988).

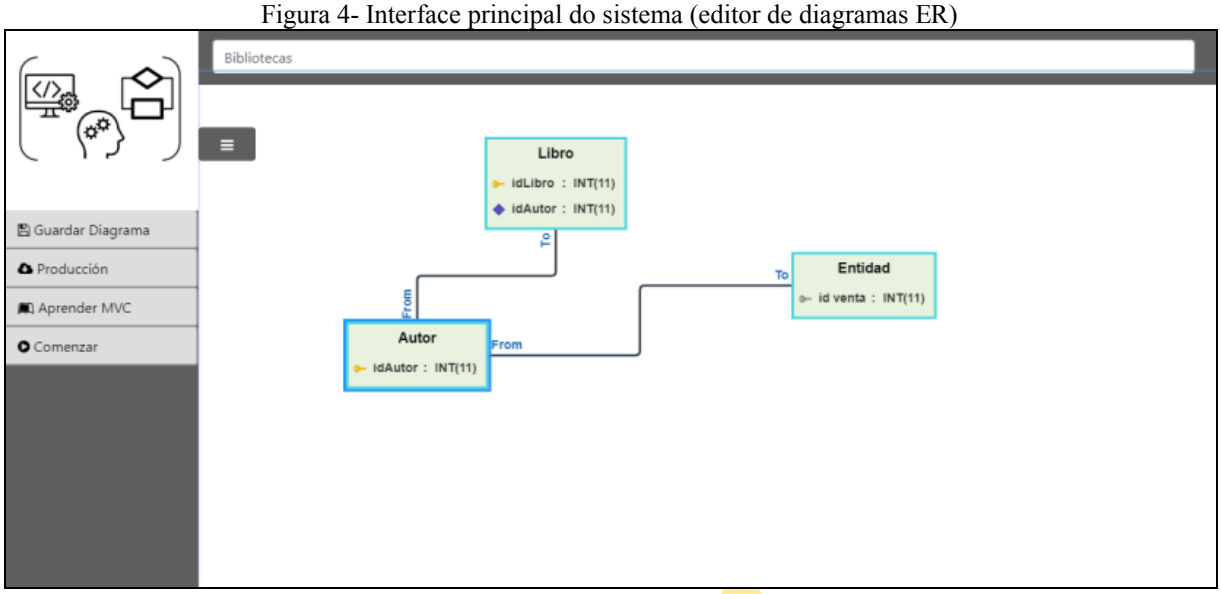
Figura 3- Conceito gráfico do padrão de projeto MVC



Fonte: Krasner e Pope (1988, p. 5)

Na Figura 4 tem-se as funcionalidades principais da ferramenta, sendo elas: criação de diagrama ER pelo usuário com apoio do editor do sistema no centro da tela acessada pela opção "Producción"; geração do projeto base em Angular com as camadas de visões, modelos, controladores e *script* do banco de dados compactada para *download* localizada a esquerda na opção "Guardar Diagrama" e aprendizado, permitindo o usuário gerar o código fonte, modifica-lo e estudar o funcionamento do padrão MVC a esquerda na opção "Aprender MVC" (SOTO *et al.*, 2020).

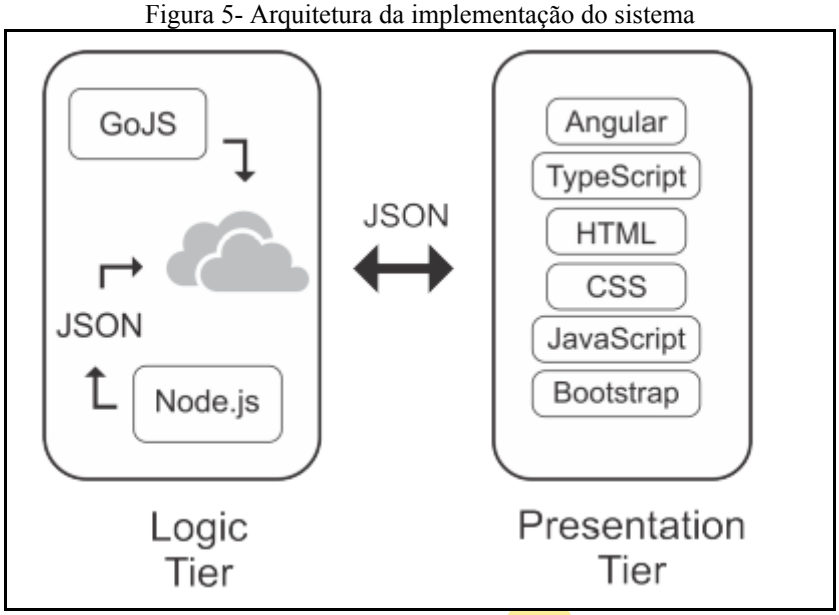
Figura 4- Interface principal do sistema (editor de diagramas ER)



Fonte: Soto *et al.* (2020, p. 61)

Na Figura 5 é ilustrada a arquitetura de implementação do trabalho dividida em duas camadas, sendo elas: camada lógica de dados e camada de apresentação. A camada lógica de dados (na esquerda) possui: o servidor NodeJS2, representando o serviço para criação do projeto base em Angular; biblioteca GoJS3 para JavaScript e TypeScript permitindo modelar e projetar os diagramas ER na camada de apresentação do sistema; e JSON4, utilizado para transformar o diagrama ER em um objeto JSON para ser enviado e processado pelo servidor. Na camada de apresentação foram utilizados os componentes Angular 7, TypeScript, HTML, CSS, JavaScript e Bootstrap com intuito de tornar a aplicação visualmente amigável (SOTO *et al.*, 2020).

Figura 5- Arquitetura da implementação do sistema



Fonte: Soto *et al.* (2020, p. 61)

Como método de validação, Soto *et al.* (2020) utilizaram a escala de usabilidade do sistema (System Usability Scale - SUS) para quantificar o grau da usabilidade da ferramenta. Essa escala engloba 10 itens em sua avaliação, cujo os valores de cada item podem variar de 0 a 100 (SOTO *et al.*, 2020), sendo eles: acho que gostaria de usar este sistema com frequência; achei o sistema desnecessariamente complexo; achei o sistema fácil de usar; acho que precisaria de apoio de um técnico para usar este sistema; achei que as várias funções deste sistema funcionavam bem integradas; eu pensei que havia muita inconsistência neste sistema; eu imagino que a maioria das pessoas aprenderiam a usar este sistema muito rapidamente; achei o sistema muito complicado de usar; me senti muito confiante ao usar o sistema e eu precisava aprender muitas coisas antes de conseguir usar este sistema (SOTO *et al.*, 2020). Ao final do trabalho Soto *et al.* (2020) concluem que a ferramenta desenvolvida obteve ótimos resultados, pois após ter sido testada e qualificada por 18 usuários aleatórios, a média de pontos entre os 10 itens da escala de usabilidade do sistema (SUS) foi superior a 70 pontos.

Soto *et al.* (2020) também concluem que a adoção do padrão de projeto MVC em desenvolvimento de sistemas aumenta a produtividade, evita tarefas monótonas e reduz significativamente os tempos de desenvolvimento, mesmo que eles não tenham testado diretamente isso. Nesse primeiro trabalho eles focaram a avaliação na usabilidade e pretendem posteriormente avaliarem aspectos de linhas de código economizadas e também o quanto pode impactar no tempo de desenvolvimento de sistemas (SOTO *et al.*, 2020).

2.3 ENGENHARIA REVERSA DE BANCO DE DADOS RELACIONAL PARA MODELO UML

Zafar *et al.* (2019) apresenta uma ferramenta de conversão de esquema de banco de dados SQL para arquivos eXtensible Markup Language Metadata Interchange (XMI) 2.1, por meio da engenharia reversa de entidades relacionais existentes em bancos de dados. O arquivo resultante dessa técnica é utilizado como entrada de dados em sistemas gráficos de modelagem UML, como por exemplo, o MagicDraw.

A técnica de engenharia reversa é utilizada para extrair informações ocultas de um baixo nível de abstração do banco de dados (descobri-las) e transformá-las em informações de alto nível de abstração (desenhos, gráficos, etc.), como por exemplo, modelos de diagramas da UML, aproximando as camadas de desenvolvimento (desenvolvedores) com as camadas de administradores de banco de dados (DBAs). Essa representação gráfica pode auxiliar a manutenção dos sistemas, pois as funcionalidades da aplicação e banco de dados podem ser modeladas em uma única linguagem usando UML, permitindo aos envolvidos em gestão de software, expressar seus pensamentos, ideias e requisitos em um só idioma comum a todos (ZAFAR *et al.*, 2019).

Zafar *et al.* (2019) esclarecem os principais objetos do artigo, sendo eles: identificar as entidades através de suas chaves primárias e identificar os relacionamentos que podem ser associações, generalizações, especializações e agregações. Para identificar os relacionamentos de associação entre duas entidades E1 e E2, foi utilizada a técnica de que se existe uma chave estrangeira entre E1 e E2, porém a chave primária for diferente entre elas, uma associação existe entre essas entidades. Para identificar os relacionamentos de generalização/especialização, foi utilizada a técnica de que se a chave primária das entidades E1 e E2 forem iguais e com chave estrangeira nessas chaves primárias, apontando de E1 para E2 ou vice-versa, significa que

uma herança existe entre essas duas entidades. Para descobrir as cardinalidades de relacionamentos de associação, foi utilizada a técnica de que entre duas entidades E1 e E2, se a chave primária de E1 é uma chave estrangeira de E2, então cada instância de E2 é associada a exatamente uma instância de E1 (um para um). Se houver entre duas entidades E1 e E2, a chave primária de E1 e E2 serem composição de uma chave estrangeira correspondente a E1 e E2, então existe uma cardinalidade de muitos-muitos (ZAFAR *et al.*, 2019).

No processo de tradução, um SQL de Linguagem de Definição de Dados (DDL) é analisado para gerar *tokens* e depois o XMI é elaborado para o modelo Unified Modeling Language (UML) com os *tokens* gerados. Nessa abordagem, para gerar o arquivo XMI 2.1 foram extraídas informações explícitas e implícitas, como o mapeamento das cardinalidades, generalização, tipo de entidade, relacionamentos de dependência, associação entre outras. Em seguida, utilizou-se esse arquivo XMI como entrada na ferramenta MagicDraw para montar os diagramas da UML conhecidos (ZAFAT *et al.*, 2019).

Zafar et al. (2019) implementaram a ferramenta na linguagem de programação Java, com analisador de SQL e conversor de *tokens*, a fim de gerar o documento XMI 2.1 para importação no MagicDraw. A abordagem utilizou Modelo de Objeto de Documento (DOM), padrão *Application Programming Interface* (API) em suporte com analisadores XML, para trabalhar com XMI e esse DOM é representado como árvores de nós. Para testar a ferramenta, Zafar *et al.* (2019) realizaram a geração forçada de tabelas, que consiste em criar manualmente uma estrutura de tabelas e relacionamentos no SGBD fictícia, ou seja, que não é utilizada por ninguém, cujo propósito serviu apenas para validar suas técnicas e teorias (ZAFAR *et al.*, 2019).

Ao final do trabalho, Zafar *et al.* (2019) concluem que a ferramenta é capaz de descobrir com sucesso, os seguintes componentes da UML: entidade, relação, generalização, dependência e mapeamento de cardinalidades. Contudo, Zafar *et al.* (2019) destacam que a incorreta modelagem do banco pode acabar causando divergências nos resultados e que seguiram os padrões da UML para realizar a engenharia reversa das entidades do banco de dados. Zafar *et al.* (2019) comentam que o diferencial do trabalho é gerar um arquivo universal para as ferramentas visuais, ao contrário de outros projetos que geram apenas arquivos em formatos próprios. Zafar *et al.* (2019) ressaltam que melhorias ao projeto são bem-vindas futuramente, afim de enriquecer os diagramas, como por exemplo, resgatar informações sobre atributos multivalorado, relação N-ária, entre outros.

3 PROPOSTA

Nas seções seguintes serão apresentadas as justificativas da proposta do projeto, os Requisitos Funcionais (RF), Requisitos Não Funcionais (RNF) e por fim as metodologias adotadas para a confecção. Na seção 3.1 está descrita a justificativa para o desenvolvimento do projeto. Na seção 3.2 estão os detalhes dos requisitos que englobarão o sistema. Por fim, na seção 3.3 estão as metodologias empregadas.

3.1 JUSTIFICATIVA

O Quadro 1 mostra o comparativo entre os trabalhos correlatos apresentados nas seções anteriores. Nas linhas são referenciadas as funcionalidades enquanto que nas colunas ficam os trabalhos correlatos.

Quadro 1- Comparativo dos trabalhos correlatos

Características \ Trabalhos Correlatos	Web Modeler 2.0 (BUGMANN, 2012)	Ferramenta de geração Model-View-Controller (SOTO *et al.*, 2020)	Ferramenta de engenharia reversa de banco de dados (ZAFAR *et al.*, 2019)
Engenharia reversa de banco de dados	X	X	X
Geração de código-fonte		X	
Criação e edição de diagramas da UML	X	X	
Utilizado padrão de projeto MVC	X		
Ferramenta Web	X	X	
Geração de arquivo XMI 2.1			X
Gerenciamento de estruturas de banco	X	X	X

Fonte: elaborado pelo autor.

Conforme apresentando no Quadro 1 todas as ferramentas realizam engenharia reversa e gerenciam as estruturas de SGBDs, sendo as principais funcionalidades do projeto proposto. Dentre elas, apenas as ferramentas Web Modeler (2012) e Ferramenta de geração Model-View-Controller (2020) estão desenvolvidas na plataforma Web. Apenas a Ferramenta de geração Model-View-Controller (2020) permite a geração de

código-fonte e apenas o Web Modeler (2012) utilizou o padrão de projeto MVC. A geração de arquivos XMI 2.1 só foi implementada na Ferramenta de engenharia reversa de banco de dados (2019).

O projeto apresentado busca englobar parcialmente algumas das funcionalidades de cada trabalho correlato apresentado no Quadro 1. Permitirá por meio de uma interface gráfica Web, manter as tabelas, seus campos e seus relacionamentos presentes nos SGBDs Oracle, realizar consultas na base de dados com o propósito de descobrir informações da estrutura dos SGBDs e transformá-las em informações gráficas de alto nível de abstração, como por exemplo, desenhos, gráficos, etc. Possibilitará gerar código-fonte para Delphi com as estruturas de tabelas e seus campos. O sistema também permitirá, através de interface gráfica, a conversão e atualização automática dos SGBDs com as alterações realizadas nas tabelas, campos e relacionamentos dentro da ferramenta gráfica, para reduzir desgastes com execução unitária de comandos DML e DDL.

No âmbito social e tecnológico esse sistema contribuirá com a melhor organização estrutural de bancos de dados dos sistemas interessados em utilizar a ferramenta de auxílio no processo de engenharia de software, possibilitando a redução do tempo gasto no processo de engenharia de sistemas e melhorando a manutenibilidade dos mesmos.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Nesta seção são apresentados os requisitos que acompanharão o projeto. Esses requisitos são divididos em Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF), conforme segue:

a) o sistema deverá possuir interface central com um menu lateral intuitivo (RF);

b) o sistema deverá manter tabelas (RF);

c) o sistema deverá manter campos de tabelas (RF);

d) o sistema deverá manter o relacionamento entre as tabelas (RF);

e) o sistema deverá gerar uma interface gráfica das tabelas por meio da descoberta dos dados do SGBD utilizando consulta na base de dados (RF);

f) o sistema deverá permitir realizar login no SGBD Oracle escolhido (RF);

g) o sistema deverá permitir consultar as tabelas e suas referências com outras tabelas (RF);

h) o sistema deverá permitir realizar conversão e atualização automática da base de dados conforme informações das tabelas existentes na ferramenta (RF);

i) o sistema deverá permitir carregar um arquivo fonte de extensão `.pas` por meio de engenharia reversa dele e importar as tabelas na ferramenta (RF);

j) o sistema deverá ser desenvolvimento para plataforma Web (RNF);

k) o sistema deverá ser desenvolvimento para comunicação nativa com SGBDs Oracle (RNF);

l) o sistema deverá ser desenvolvido em *frontend*, utilizando as linguagens HTML, CSS, Javascript e Ajax na IDE Visual Studio Code;

m) o sistema deverá ser desenvolvido em *backend*, utilizando a linguagem de programação C#, com Rest Web Api na IDE Visual Studio.

3.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: realizar o levantamento bibliográfico sobre aplicação Web para gerar interface gráfica e comunicação nativa entre *frontend*, *backend* e SGBDs Oracle, comandos SQLs para efetuar consultas nas estruturas da base de dados e complementar informações já vistas nos trabalhos correlatos;
b) levantamento de requisitos: complementar o levantamento de RFs e RNFs já definidos na seção 3.2;
c) especificação: elaborar diagramas de caso de uso, atividades, Modelo Entidade e Relacionamento (MER), assim como os diagramas que forem necessários durante a construção do projeto, utilizando Astah UML;
d) desenvolvimento da aplicação Web: será desenvolvido conforme proposto anteriormente nos RF e RNF. A aplicação será desenvolvida no Visual Studio e Visual Studio Code possuindo interface gráfica responsiva;

e) testes e validação: serão realizados testes juntamente com a construção da ferramenta para validar que todas as funcionalidades do sistema estão bem projetadas. Serão realizados testes de comunicação com o *frontend*, *backend* e base de dados, além de teste de usabilidade.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

	2022									
	fev.		mar.		abr.		mai.		jun.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
levantamento bibliográfico										
levantamento de requisitos										
especificação										
desenvolvimento da aplicação Web										
testes e validações										

Fonte: elaborado pelo autor.

4 REVISÃO BIBLIOGRÁFICA

Este capítulo tem como objetivo explorar os conceitos e fundamentos mais importantes para realização desse trabalho: sistema legado e engenharia reversa, mas especificadamente de banco de dados.

Define-se sistema legado como sistema crítico que está em uso a um longo período, onde possivelmente foi desenvolvido com tecnologias ultrapassadas e que é peça fundamental para a organização que o mantem (BARBOSA; CANDIDO, 2017). Na contramão da tecnologia, em constante evolução estes sistemas costumam entrar em produção desatualizados tecnologicamente, devido a atrasos no desenvolvimento (PINTO; BRAGA, 2004). Uma das possíveis explicações do por que esses sistemas ainda existem até hoje é uma questão de custo, pois seu desenvolvimento inicial pode ter levado a um custo muito alto e que só haverá retorno desse investimento após vários anos de uso (PINTO; GRABA, 2004). O tempo de duração de sistemas de software é muito variável e sistemas de grande porte permanecem em uso por mais de dez anos (PINTO; BRAGA, 2004). Outra possível explicação pode estar ligada ao quão importante esse sistema é para organização que o desenvolveu e para as empresas que o utilizam, onde uma simples falha dos serviços desse sistema possa causar sérios efeitos colaterais no dia-a-dia de ambos (PINTO; BRAGA, 2004).

Engenharia reversa para banco de dados é o processo de entender como algo funciona através da análise de sua estrutura, função e operação permitindo conhecer ele foi pensado e desenhado mesmo sem possuir o projeto original. Engenharia reversa pode ser usada para entender como algo funciona, ou como foi feito, mesmo sem possuir a sua documentação (SANTOS; BIANCHINI, 2019). O resultado da engenharia reversa é a especificação de um produto que é gerada a partir das análises realizadas no sistema. Com isso nos permite entender o funcionamento do mesmo, para alterar ou realizar manutenções futuras, ou até mesmo acrescentar novas funcionalidades ao software em questão (SANTOS; BIANCHINI, 2019). Essa técnica de engenharia reversa cresceu muito, e hoje possibilita o entendimento e melhoria do software, até na descoberta de segredos indústria e comerciais (SANTOS; BIANCHINI, 2019). Para Barbosa e Candido (2017), a manutenção e evolução do sistema pode eventualmente se tornar financeiramente inviável quando não se tem a documentação do sistema, pois qualquer modificação pode impactar em outras funcionalidade do software, assim como o tempo pode acabar se estendendo por longos prazos (BARBOSA; CANDIDO, 2017). Para isso pode-se fazer uso da engenharia reversa para documentar o que já está em funcionamento, no intuito de criar uma documentação para entender seu funcionamento e facilitar as alterações (BARBOSA; CANDIDO, 2017).

REFERÊNCIAS

BARBOSA, Pedro L. S.; CANDIDO, Adriano L. Diferenças entre engenharia reversa e reengenharia nos sistemas de informação. **Revista interfaces**, Icó-CE, v. 4, n. 13, Dez. 2017.

BUGMANN, Paulo A. Ferramenta web para modelagem lógica em projetos de bancos de dados relacionais. **FURB**. Blumenau-SC, p. 7-84, Dez. 2012.

KRASNER, Glenn E.; POPE, Stephen. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk80 System. **Journal Of Object Oriented Programming**, Santa Bárbara-CA, vol. 1, Jan. 1988.

VIEIRA, Luiz Flavio. COMPARAÇÃO DE PERFORMANCE DE SISTEMAS GERENCIADORES DE BANCO DE DADOS. 104 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2020.

PINTO, Herbert L. M.; BRAGA, José L. Sistemas legados e as novas tecnologias: técnicas de integração e estudo de caso. **Informática Pública**,, v. 8, n. 1, p. 47-69, 2005.

SANTOS, Ademir C.; BIANCHINI, Calebe P. Engenharia reversa de um sistema pdv. **Scholar.google**, Consolação-SP, p. 2-15, Nov. 2019.

SOTO, Jésus et al., A software tool to generate a Model-View-Controller architecture based on the Entity-Relationship Model. **IEEE Xplore**, Chetumal-Mexico, p. 1-7, Abr./Jun. 2020.

ZAFAR, Sherin et al. Reverse Engineering of Relational Database Schema to UML Model. **IEEE Xplore**, Aligarh-India, Ago./Out. 2019.