UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUIZ FERNANDO SOARES VILAS BOAS

VERSIONAMENTO DE ESQUEMAS RELACIONAIS EM AMBIENTES MULTI-MÓDULO

CAMPINA GRANDE - PB

2023

LUIZ FERNANDO SOARES VILAS BOAS

VERSIONAMENTO DE ESQUEMAS RELACIONAIS EM AMBIENTES MULTI-MÓDULO

Trabalho de Conclusão de Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientadores: Professor Dr. Claudio de Souza Baptista

CAMPINA GRANDE - PB

2023

LUIZ FERNANDO SOARES VILAS BOAS

VERSIONAMENTO DE ESQUEMAS RELACIONAIS EM AMBIENTES MULTI-MÓDULO

Trabalho de Conclusão de Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Claudio de Souza Baptista

Orientador – UASC/CEEI/UFCG

Professor Dr. Andrey Elisio Monteiro Brito

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em 14 de Fevereiro de 2023.

CAMPINA GRANDE - PB

ABSTRACT

New technologies are increasingly present in the daily lives of humanity, consequently, this leads to a large increase in the production of digital data. Thus, database management systems are being increasingly demanded. In this context, specifically in modularized environments, there was a need to analyze the challenges in dealing with version control in relational schemas. The construction of this research was divided into three stages: an analysis of a real case of a large tax solutions company; a study of the alternatives currently available on the market; and, finally, the suggestion of initial solutions for versioning in this type of environment. From the studies carried out, it was seen that the company's system is extremely dense considering the issues of practices, processes and tools used. Therefore, although there are alternatives on the market, adapting an integration of an external solution to the company's system would entail radical changes and in some cases even unfeasible. Therefore, it is advisable to look for a mixed solution, to maintain the company's processes and adhere to external tools that optimize the quality and efficiency of version control.

Versionamento de esquemas relacionais em ambientes multi-módulo

Luiz Fernando Soares Vilas Boas
luiz.boas@ccc.ufcg.edu.br
Universidade Federal
de Campina Grande
Campina Grande, Paraíba

Claudio de Souza Baptista
baptista@computacao.ufcg.edu.br
Universidade Federal
de Campina Grande
Campina Grande, Paraíba

Hugo Feitosa de Figueiredo
hugo.figueiredo@ifpb.edu.br
Universidade Federal
de Campina Grande
Campina Grande, Paraíba

RESUMO

As novas tecnologias estão cada vez mais presentes no cotidiano da humanidade, consequentemente isso acarreta grande aumento na produção de dados digitais. Dessa forma, os sistemas de gerenciamento de bancos de dados estão sendo cada vez mais demandados. Nesse âmbito, especificamente em ambientes modularizados, notou-se uma necessidade em analisar os desafios em lidar com o controle de versão nos esquemas relacionais. A construção desta pesquisa foi dividida em três etapas: uma análise de um caso real de uma grande empresa de soluções fiscais; um estudo sobre as alternativas atualmente disponíveis no mercado; e, finalmente, a sugestão de soluções iniciais para o versionamento neste tipo de ambiente. A partir dos estudos realizados, foi visto que o sistema da empresa é extremamente denso considerando as questões de práticas, processos e ferramentas utilizadas. Logo, apesar de existir alternativas no mercado, adaptar uma integração de uma solução externa ao sistema da companhia acarretaria mudanças radicais e em alguns casos até inviáveis. Assim, é recomendado buscar uma solução mesclada, para manter os processos da companhia e aderir a ferramentas externas que otimizem a qualidade e a eficiência do controle de versão.

Palavras-chave

Versionamento, Modularização, Prática, Ferramenta

1. INTRODUÇÃO

Em meio a grande transformação digital e o avanço das novas tecnologias na atualidade, estima-se que são criados todos os dias cerca de 2,5 quintilhões de bytes de dados, de modo que o volume de dados produzidos nos últimos anos já representa 90% de todos os dados fabricados pela humanidade [1]. Dessa forma, os Sistemas de Gerenciamento de Banco de Dados (SGBDs) estão sendo cada vez mais demandados, decorrente do grande aumento da produção de dados. Assim, nota-se um grande desafio em desenvolver alternativas para otimizar o funcionamento desses sistemas.

Desse modo, em bancos de dados, é ideal que as alterações realizadas no projeto sejam controladas por algum sistema de versionamento. Antigamente, o versionamento era utilizado apenas para salvar versões antigas do sistema para, se caso fosse necessário, uma restauração. Porém, atualmente, esse método é utilizado de forma ampla para facilitar o desenvolvimento compartilhado por vários usuários de um sistema. Dessa maneira, o controle de versão pode incluir a capacidade de criar uma lista de diferenças entre duas versões para que elas sejam usadas para ambientes distintos dentro da lógica da empresa que o adota, como ambientes de teste, produção e desenvolvimento [2].

Geralmente, para o desenvolvimento de uma aplicação, utiliza-se um sistema de controle versão (e.g., Git, Apache Subversion (SVN), Concurrent Version System (CVS), Mercurial) para os códigos fonte do front end, do back end e para a documentação [5]. Entretanto, o banco de dados requer um elevado custo de armazenamento e gerenciamento, por isso, normalmente, são alocados em robustos servidores locais da própria empresa que detém os dados ou essa tarefa é terceirizada para companhias que oferecem o serviço de armazenamento em nuvem. Por esse motivo, o controle de versão do esquema relacional não pode ser o mesmo das outras partes da aplicação (front end, back end e documentação). Pois, deve existir uma integração entre o controle de versão do código e esquema, inclusive para evitar problemas durante o desenvolvimento por incompatibilidade de versões. Logo, algumas empresas optam por desenvolver softwares internamente ou, em alguns casos, optam por usar uma ou mais ferramentas do mercado.

Logo, dentre as várias arquiteturas para aplicações web, uma delas é um ambiente modularizado ou multi-módulo, que consiste em um software que constrói uma solução que está fragmentada em vários módulos dentro das suas especificidades. De forma que esses módulos funcionam independentemente, ou seja, um cliente que apenas utiliza um módulo de toda a solução, não precisa ter os outros módulos para que o mesmo funcione. Entretanto, esses módulos se relacionam entre si, por meio de um mesmo banco de dados, e existindo a possibilidade de compartilharem acesso a uma mesma estrutura de dados, como uma tabela ou um index.

Dessa maneira, especificamente em bancos implementados em ambientes multi-módulos, podemos identificar muitas dúvidas sobre como construir um versionamento nesses sistemas. A título de exemplo, “quando uma alteração no banco de dados é realizada para solucionar um problema em um módulo específico, essa alteração pode afetar negativamente outros

módulos?" (visto que nesse tipo de sistema um único banco de dados alimenta todos os módulos).

Além disso, "em uma circunstância que um cliente solicite uma correção específica no banco de dados, até que ponto publicar essa modificação em uma nova versão geral para que todos os clientes tenham acesso seria válido?". E mais: "considerando que isso poderia afetar algo no sistema dos outros clientes, ou se a empresa decidir por não lançar as alterações para outros usuários, como ela conseguiria manter um controle da versão do banco que cada cliente utiliza?".

Desse modo, em um sistema de uma empresa que adota um ambiente modularizado, percebe-se os desafios para adaptar uma solução de versionamento. Esses desafios ultrapassam as questões relatadas nos parágrafos anteriores, sendo preciso adaptar um método de controle de versão que altere o mínimo possível os processos e práticas seguidas pela empresa, além de ser compatível com os sistemas de banco de dados.

Assim, este trabalho tem como objetivo realizar uma análise para a implementação de sistemas de versionamento em bancos multi-módulo, por meio de um estudo de caso, buscando identificar as características e limitações a nível de tecnologias, de processos pré-estabelecidos e práticas da comunidade que desenvolve o sistema. A partir desta análise, será realizada uma revisão sobre o que o mercado oferece de soluções para esse tipo de sistema e assim serão elaboradas sugestões e debates para solucionar os obstáculos que forem evidenciados.

A pesquisa é motivada pela importância de colaborar na área acadêmica e profissional da Ciência da Computação, mais especificamente na ciência de dados, que, segundo "Bureau of Labor Statistics" dos EUA, prevê um crescimento neste campo de atuação de cerca de 28% até 2026 [3], dessa maneira, impactando diretamente no processo da transformação digital da atualidade. Além disso, existe o estímulo em inicializar um estudo para analisar e compreender os desafios de incorporar e otimizar arquiteturas modularizada integrada ao banco de dados versionado.

2. ASPECTOS METODOLÓGICOS

Esta pesquisa foi realizada em um processo de 3 etapas, cuja representação do fluxo de passos pode ser observada na Figura 1.

Figura 1. Etapas do processo de estudo para a estruturação da pesquisa.

Na primeira etapa, foi realizada uma análise de um caso real de versionamento de esquemas relacionais em uma empresa chamada "Synchro Soluções Fiscais" que adota um modelo de sistema multi-módulo. Essa companhia é a maior provedora nacional de soluções de conformidade, reunindo uma carteira de mais de 360 grupos econômicos, 14.000 usuários e 44.000 estabelecimentos fiscais [8], assim podemos perceber o elevado tamanho do sistema que a empresa adota. Foram extraídas as informações a partir de reuniões com colaboradores da instituição que têm um amplo conhecimento do sistema, além de um contato direto com partes do processo de alterações no banco de dados.

O estudo de caso teve o foco direcionado em analisar o funcionamento do sistema em uma ótica geral, assim não se limitando a apenas entender sua arquitetura, como também entender como os desenvolvedores lidam com as tecnologias e quais são as práticas e processos intrínsecos da cultura da comunidade da instituição.

Logo, baseado na conclusão desse estudo de caso, a segunda etapa foi partir para uma análise de ferramentas, processos e práticas que o mercado oferece atualmente. Após isso, o terceiro passo da pesquisa consistiu em: com base no estudo de caso e na análise do mercado, sugerir e debater sobre possíveis soluções iniciais para a otimização do versionamento em um banco de dados em um ambiente modularizado.

3. ANÁLISE DO ESTUDO DE CASO

Nesta seção, é abordada a análise do estudo de caso feito no sistema de banco de dados de uma empresa de software. Para fins de um entendimento inicial, esse é um sistema com ambiente modularizado em que existe um controle de versão limitado, que aplica métodos para lidar com a concorrência de versões distintas no código fonte do esquema relacional.

3.1 Concorrência na aplicação de scripts

Nesse ambiente, a aplicação de scripts no banco de dados é executada por uma ferramenta construída e mantida pela própria empresa. Logo, para lidar com a questão versionamento, existem padrões no software para que não ocorra concorrência de alterações no código, ou seja, que dois ou mais usuários não estejam lançando ao mesmo tempo modificações em um mesma estrutura ou estruturas dependentes.

Para isso, quando o desenvolvedor abre uma requisição de alteração em uma estrutura do banco de dados, a ferramenta mapeia os metadados que o usuário está modificando e as suas estruturas dependentes. A partir desse mapeamento, o sistema não permite que outros usuários abram um pedido para alterar nenhuma dessas estruturas. Apenas após a liberação da requisição inicial, esses trechos do código são liberados para serem modificados.

Portanto, esse tipo de procedimento, apesar de funcionar e estar bem enraizado na cultura de processos da empresa, é uma alternativa que evidentemente produz um atraso natural no processo de alterações, visto que, diferente de um sistema com um versionamento adequado, em que mais usuários podem alterar ao mesmo tempo a mesma região do código, na ferramenta da empresa é possível que apenas um desenvolvedor por vez faça este processo.

É importante salientar que, segundo a empresa, estima-se que cerca de 70% das regras de negócio do sistema estão localizadas no banco de dados, logo o volume de manutenção no banco da aplicação tende a ser maior que em outros sistemas. Assim, corroborando a importância de otimizar o controle de alterações do banco, pois a solução atual gera um gargalo no desenvolvimento da aplicação.

3.2 Processamento na aplicação de scripts

No processamento da aplicação de scripts da ferramenta, foi realizada uma análise dentro da empresa que demonstrou que o processamento é lento no lançamento de migrations com um grande volume de dados, comparando a ferramentas do mercado. Um dos fatores que ajudam a reforçar o problema do carregamento, é que o funcionamento desse processamento não é realizado de forma totalmente paralela para aplicar os scripts para os clientes, mas é feito de forma semi paralela, ou seja, ele aplica para um quantidade limitada de clientes de cada vez.

Além disso, também foi visto que é necessário que todo o processo de atualização dos clientes que iniciaram juntos o carregamento termine, para que o processo com os outros clientes que estavam na fila se iniciem. Logo, não existe uma otimização na ferramenta para assim que um cliente finalizar a atualização, outro cliente que está na fila já inicie o processo, logo tem que todos terminarem para que os novos entrem no processo de atualização.

3.3 Pacote global e específico

O sistema utiliza um processo de pacotes para fazer o lançamento das alterações para um ambiente de produção. De maneira que, para que todos os módulos tenham acesso a algumas informações gerais, é lançado sempre dois pacotes de alterações no código. O pacote global contém objetos do banco de dados, como colunas, tabelas, índices, entre outros semelhantes, e isso é feito com intuito de atualizar a estruturas de dados do banco, para estarem idêntica para todos módulos, o que é de vital importância, uma vez que existem módulos distintos que usam uma mesma estrutura de dados.

Por outro lado, existe também um pacote específico para ser lançado apenas no módulo em que a alteração foi requisitada. Esse pacote contém, em sua grande maioria, as regras de negócio do módulo que está sendo realizado a alteração. É importante salientar que essa estratégia de pacotes está extremamente consolidada dentro da empresa, tanto como processo de lançamento de versões, como quanto a cultura da companhia. Desse modo, é importante que, em possíveis propostas de soluções, essa estratégia seja mantida.

3.4 Práticas da comunidade

Além das questões de tecnologias e de processos que podemos apontar como obstáculos do sistema, existem também as questões de práticas da comunidade. Uma das práticas que podem ser consideradas negativas é que a empresa contém várias equipes de desenvolvimento, e, apesar de todos usarem a mesma ferramenta de alteração do banco de dados, cada equipe incorpora padrões distintos para lidar com esse processo. Logo, isso acaba gerando muitas dificuldades em implantar modificações no sistema, uma vez que cada equipe lida de forma diferente com o sistema atual.

Outro ponto importante de destacar é uma má prática que envolve não apenas os desenvolvedores do sistema, mas também os clientes que utilizam a aplicação. Um dos casos que foram relatados é que existem situações em que os próprios clientes não atualizam o sistema de banco para a versão mais recente. Esse tipo de prática geralmente acontece pelo fato do cliente entender que, se o sistema está funcionando de forma normal, não há necessidade de empregar um tempo realizando o processo de atualização e que algumas operações que ele pode está habituado a usar mude a forma de funcionamento.

Assim, quando esse mesmo cliente abre um chamado com algum problema que envolve componentes do banco de dados, acaba gerando um custo a mais de análise que em clientes que usam um sistema atualizado, uma vez que eles têm que analisar se, em versões mais recentes da aplicação, o problema do cliente foi resolvido ou não.

4. ANÁLISE DE MERCADO

Nesta seção, o objetivo foi o aprofundamento no entendimento das tecnologias, processos e práticas que o mercado adota para o versionamento em banco de dados, logo como critério para esse estudo foram considerados especificamente soluções que portassem uma correlação com os obstáculos apresentados na seção 3. Dentro do tópico de quais meios foram utilizados para realização da análise de mercado, o estudo foi feito a partir da investigação das características dos principais tipos de ferramentas de versionamento utilizadas atualmente. Além da análise de alguns processos e práticas que dentro da literatura são consideradas importantes de serem adotadas para se ter um controle de versão de esquemas relacionais saudável.

4.1 Ferramentas de versionamento

Existem muitas ferramentas de versionamento para banco de dados, de modo que podemos classificar os tipos de controle de versão em “Whole-Schema” e “Change Script” [9].

As ferramentas classificadas como "whole-scheme" são as que adotam um controle de versão que fornece aos usuários ferramentas de implantação, desta forma o desenvolvedor não precisa escrever na maioria das vezes os scripts, pois a própria ferramenta identifica as alterações entre as versões. Esse processo, que já é muito utilizado em desenvolvimento em back end e front end, permite que as alterações sejam feitas de forma otimizada, logo, grande parte do trabalho que seria realizado manualmente é feito pela ferramenta.

Deste modo, exemplos deste sistema são o SQL Server Data Tools (SSDT) e o Entity Framework Migrations, duas ferramentas diferentes, mas que funcionam com padrões que caracterizam um controle de versão “Whole-Schema”. Por outro lado, em um contexto de grandes alterações em uma tabela no banco de dados, pode ocasionar que a ferramenta não suporte a migração do script, gerando assim a necessidade de modificações manuais do desenvolvedor [4]. Logo, no estudo de caso da pesquisa, caso fosse implementada essa solução de controle de versão, a falha supracitada seria um ponto de atenção, pelo tamanho e complexidade do banco de dados.

Já o controle de versão baseado no padrão “Change Script” segue um método de armazenar arquivos com o passo a passo para criar os objetos do banco de dados. Esse modelo tem como um de seus representantes no mercado o Liquibase. Esse processo garante um controle mais completo do usuário do script de alteração no sistema, assim facilitando no controle das migrações, principalmente, as mais complexas [9].

Uma das desvantagens é que, diferente do processo ágil de alteração do "Whole-Schema", o "Change Script" tem um processo de alterações do banco considerado mais lento, uma vez que o desenvolvedor terá que elaborar as alterações manualmente no script. Além disso, esse processo requer um passo a mais quando se necessita uma alteração, pois existe o custo da análise manual do banco de dados por meio dos scripts em arquivos [9]. Deste modo, em grandes sistemas isto pode ser um grande obstáculo, pelo tamanho e complexidade que os scripts podem chegar neste tipo de sistema.

4.2 Processos e práticas

Na literatura sobre os bancos de dados, o livro "Refactoring Databases: Evolutionary Database Design", de Scott Ambler e Pramodkumar Sadalage [7], traz uma visão ampla da importância de não apenas priorizar as escolhas das ferramentas e tecnologias que serão adotadas ao projeto, mas do quão fundamental são as questões de processos e práticas que serão adotados dentro da comunidade da empresa.

4.2.1 Relação desenvolvedor-dba

No momento que se faz necessário uma alteração no banco de dados, uma das relações mais importantes é entre o DBA (Administrador de Banco de Dados) e o desenvolvedor, de modo que toda tarefa em que um desenvolvedor trabalha no banco de dados potencialmente precisa da ajuda de um DBA [6]. Esse tipo de prática deveria funcionar de forma constante nas empresas, já que o desenvolvedor quando for realizar uma alteração de banco idealmente necessitaria de uma consultoria do DBA que tem uma visão geral do problema, assim o DBA ajudaria a implementar da melhor forma a modificação. Porém, ainda em muitos ambientes, são vistas muitas barreiras entre o DBA e as funções de desenvolvimento de aplicativos [6], assim acarretando em dificuldades no processo de desenvolvimento e suporte ao banco de dados.

5.2.2 Instâncias em banco de dados

Quando falamos de versionamento em back end e front end, é natural que as principais soluções de mercado adotem a funcionalidade de criar instâncias do código fonte do projeto, criando assim branchs para cada usuário e que posteriormente elas serão mescladas com a versão de produção do projeto.

Entretanto, em banco de dados a maioria das organizações de desenvolvimento compartilha um único banco de dados de desenvolvimento [6], que é usado por todos os membros da organização. Especificamente em bancos de grande porte, como o apresentado na seção 3, esta é uma consequência da dificuldade não apenas de gerenciamento de múltiplos bancos, mas principalmente do alto custo financeiro de uma solução. Para exemplificar, se projetamos uma solução próxima do que hoje se usa no mercado para back end e front end, em que cada desenvolvedor tenha uma versão própria do banco, de forma geral seria tecnicamente uma solução ideal para tratarmos do versionamento, uma vez que essa já é uma solução consolidada no mercado para o desenvolvimento. Entretanto, para que isso seja possível em um banco de dados volumoso, cada usuário teria que executar o seu banco localmente e isso geraria um custo de hardware extremamente alto e consequentemente um custo financeiro que impossibilitaria a solução.

5. RESULTADOS

Como exposto na Seção 3, a ferramenta interna que a empresa utiliza contém ineficiências que afetam diretamente a velocidade e qualidade do gerenciamento do banco de dados da aplicação. Desse modo, em um primeiro momento uma proposta de solução baseada em melhorar a ferramenta interna, aparenta ser uma boa escolha, principalmente, pelo fato de ser uma alternativa menos radical, que não provocaria grandes mudanças nos processos e práticas da empresa. Entretanto, por ser uma ferramenta mantida pela própria empresa, o desenvolvimento dessas alterações acarretaria um alto custo, além do aumento nas despesas de manutenção da ferramenta ao longo do tempo.

Logo, a própria empresa compreende que uma excelente alternativa consiste em migrar completamente todo o processo de alterações no banco de dados para ferramentas externas que pudessem manter o sistema. Entretanto, a pesquisa mostra que existem muitos obstáculos para que ferramentas externas sejam adotadas ao processo da empresa, em detrimento do sistema conter um ambiente modularizado e com um banco de dados muito volumoso. Além disso, mudar um processo cultural da empresa não é algo simples, por isso o ideal é buscar processos que alterem o mínimo possível na empresa, mas que auxiliem também em ajustar más práticas que a empresa adota.

Dessa forma, muitas alternativas podem ser seguidas para uma solução, como a utilização de um sistema "change-script" como o Liquibase, que atenderia grande parte dos problemas de versionamento apresentados, principalmente por ser uma ferramenta que garante um controle mais completo das alterações no banco. Porém, existe um ponto de atenção que a própria empresa já percebeu, que é a questão se a ferramenta seria capaz de suportar o tamanho do sistema. Para ter uma percepção da quantidade de dados do sistema, a empresa hoje é a maior cliente do Brasil de um dos maiores serviços de computação em nuvem do mundo, a "Oracle Cloud". Logo, seria importante uma análise a nível técnica aprofundada, para estudar a possibilidade de adotar essa tecnologia.

Porém, mesmo que essa barreira técnica fosse superada, ser uma ferramenta que suporta a aplicação é apenas um requisito, pois os maiores desafios envolvem a adaptação aos processos e práticas para receber essa ferramenta na empresa. Portanto, percebemos que não será apenas uma tecnologia que ajustará todos os problemas. Apesar da ideia da própria empresa de descontinuar por completo a ferramenta interna, provavelmente pelas questões dos processos, uma solução mais eficiente caminharia para uma adaptação dessa ferramenta interna para receber as tecnologias externas, e que de certa forma tenha mudanças menos radicais no processo da empresa, como, por exemplo, não mudar a lógica dos lançamento de scripts com os pacotes global e específico, pois esse método feito pela empresa é de suma importância pelo funcionamento do banco no ambiente modularizado. Além disso, há também questões de práticas, que estão dentro do que o mercado segue que a empresa não adota. Logo, há muitas questões culturais dentro da empresa que devem passar por reestruturação, como a questão da padronização do processo de alterações no banco de dados.

6. CONCLUSÃO

A pesquisa teve limitações principalmente em detrimento da análise do caso real, uma análise de um sistema grande que envolve tecnologias, processos e práticas é algo extremamente denso. Logo, não foi possível se aprofundar em todos os tópicos que o estudo apontou, além disso não foi possível ter contato direto com a ferramenta interna da empresa, apenas a interação com alguns estágios do processo de alteração, mas sem acesso à etapa final, na aplicação dos scripts feitos pelo software.

Ademais, na análise de mercado, por questões naturais de confidencialidade, não tivemos acesso a muitas informações de como outras empresas lidam com a questão do versionamento, em ambientes parecidos com o que foi analisado. Logo, isso sem dúvida é uma entrave em entender as melhores práticas e processos para o problema que foi destacado.

Já que o objetivo do trabalho refere-se ao aprofundamento na análise de soluções para versionamento em bancos multi-módulo, o ideal era seguir essa base tendo um foco maior na parte do mercado, para que se proponha soluções mais gerais para que qualquer sistema possa se espelhar, já que nesta pesquisa o ajuste foi focado no sistema específico do estudo de caso.

Outra alternativa poderia seguir os passos de estudar o sistema da empresa de forma mais aprofundada, podendo propor soluções mais robustas para o sistema da empresa. Como, por exemplo, fazer uma análise mais técnica na compatibilidade em ferramentas externas que suportem o sistema. Além disso, existe a questão das instâncias em banco de dados, que é uma solução que tecnicamente é considerada ideal, pois é o método que o mercado já usa de forma ampla para front end e back end. Entretanto, na prática, por obstáculos encontrados em nível de hardware, é inviável no banco de dados, logo um estudo que inicie uma análise da compatibilidade da utilização de instância no banco seria de suma importância.

7. REFERÊNCIAS

[1] MARR, Bernard. **How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read**. Forbes, 2018. Disponível em: <https://www.forbes.com/sites/forbesdigitalcovers/2018/07/12/why-the-rocks-social-media-muscle-made-him-hollywoods-highest-paid-actor/?sh=6bc7a040136b>. Acesso em 29 dez. 2022.

[2] SEHN, Tim. **So you want Database Versioning?**. Dolthub, 2022. Disponível em: <https://www.dolthub.com/blog/2022-08-04-database-versioning/>. Acesso em 17 dez. 2022.

[3] SCHROEDER, Bernhard. **The Data Analytics Profession And Employment Is Exploding—Three Trends That Matter**. Forbes, 2021. Disponível em: <https://www.forbes.com/sites/bernhardschroeder/2021/06/11/the-data-analytics-profession-and-employment-is-exploding-three-trends-that-matter/?sh=56c0ac703f81>. Acesso em 17 dez. 2022.

[4] ALLEN, Jonathan. **How to Source Control Your Databases for DevOps.** InfoQ , 2018. Disponível em: <https://www.infoq.com/articles/DevOps-Databases/&sa=U&ved=2ahUKEwjB-ZHtn8f0AhWKmWoFHWODDGAQxfQBegQIBRAB&usg=AOvVaw0vRmaV78_gCZGoxEEJjCaP>. Acesso em 22 out. 2022.

[5] RAWSON, Rob. **2020 Version Control Software Comparison: SVN, Git, Mercurial**. Time Doctor, 2020. Disponível em: <https://www.timedoctor.com/blog/git-mecurial-and-cvs-comparison-of-svn-software>. Acesso em 13 nov. 2022.

[6] SADALAGE, Pramodkumar; FOWLER, Martin. **Evolutionary Database Design**. Martinfowler.com, 2016. Disponível em: <https://www.martinfowler.com/articles/evodb.html#AllDatabaseArtifactsAreVersionControlledWithApplicationCode>. Acesso em 10 out. 2022.

[7] AMBLER, Scott; SADALAGE, Pramodkumar. **Refactoring Databases: Evolutionary Database Design.** 1.ed. Londres: Addison-Wesley Professional, 2006.

[8] SYNCHRO. **Sobre a Synchro**. Empresa. Disponível em: <https://www.synchro.com.br/empresa/>. Acesso em 1 jan. 2023.

[9] SINGH, Gursimran. **DevOps for Databases and Why it is Important?**. Xenonstack: A Stack Innovator, 2021. Disponível em: <https://www.xenonstack.com/blog/devops-for-databases>. Acesso em 10 dez. 2022.