CENTRO PAULA E SOUZA

FATEC OURINHOS

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Márcio Daniel Moreira

Murilo Domanoski Carvalho

SISTEMA DE VALIDAÇÃO DE PARÂMETROS DE BANCO DE DADOS

OURINHOS (SP)

2016

MÁRCIO DANIEL MOREIRA

MURILO DOMANOSKI CARVALHO

SISTEMA DE VALIDAÇÃO DE PARÂMETROS DE BANCO DE DADOS

Projeto de Pesquisa apresentado à Faculdade de Tecnologia de Ourinhos para qualificação do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. João Mauricio Hypolito

OURINHOS (SP)

2016

Márcio Daniel Moreira

Murilo Domanoski Carvalho

SISTEMA DE VALIDAÇÃO DE PARÂMETROS DE BANCO DE DADOS

Trabalho de Graduação – TG apresentado a Faculdade de Tecnologia de Ourinhos como requisito para a conclusão do Curso de Analise e Desenvolvimento de Sistemas.

Data da aprovação: _____/_____/_____

_____ (assinatura) - _____ (nota)

Orientador: Prof. João Mauricio Hypolito – Fatec Ourinhos

_____ (assinatura) - _____ (nota)

Prof. _____ - _____

_____ (assinatura) - _____ (nota)

Prof. _____ - _____

A todos os colaboradores.

AGRADECIMENTOS

Agradecemos primeiramente à Deus, pois sem o seu sustento e sua graça não estaríamos aqui.

À nossas famílias pelo apoio, e por acreditar sempre que conseguiríamos cumprir com nossos objetivos e nos fazendo enxergar além do que nós poderíamos ver.

Agradecemos a todos amigos que caminharam conosco no decorrer do curso, nos dando forças e lutando as nossas lutas, e nos trazendo momentos de diversão em meio aos estudos.

Agradecemos em especial ao professor Emerson Rogério Alves Barea idealizador do projeto, e nosso orientador na primeira etapa de desenvolvimento, e ao professor João Mauricio Hypolito, que assumiu a responsabilidade após o desligamento do professor Emerson da unidade, ambos nos auxiliando e motivando a conseguir o nosso melhor, e com o seu conhecimento agregando a nossa carreira profissional e pessoal.

Agradecemos a todos os professores que tivemos contato ao longo do curso, pois o conhecimento e capacidade que demonstraram nos fizeram crescer como profissionais e a instituição a qual fazem parte.

“O sucesso é
ir de fracasso em fracasso sem perder entusiasmo”.
(Winston Churchill)

RESUMO

Esse estudo estabelece uma visão crítica sobre a necessidade de se ter um Banco de Dados dentro de padrões, baseado nas boas práticas de segurança publicada pelas desenvolvedoras dos SGBD's para um melhor uso do mesmo. Com o intuito de auxiliar o administrador de banco de dados no processo de otimização do banco, melhorando, assim, a definição dos parâmetros de boas práticas de segurança, que por si só não é uma tarefa fácil, mas sim uma tarefa complexa que requer habilidade e conhecimento técnico de seu administrador. Desta forma, será desenvolvido um sistema que irá obter, analisar, criticar e emitir relatórios acerca do resultado obtido na análise desses parâmetros de segurança. O estudo abrange especificamente os processos de configuração segura do produto Oracle DataBase 11g e Oracle TNS Listener levantados no Trabalho de Titulação de autoria de William Devidé Komel, e a utilização dos mesmos para alimentar o sistema. Também deixando em aberto para novos estudos que poderão ser realizados e posteriormente alimentar o sistema com seus dados. De forma geral, esse estudo irá viabilizar um sistema no qual qualquer administrador de banco de dados poderá obter resultados sobre as configurações de segurança em banco de dados Oracle, tendo tudo dentro de parâmetros que a Oracle e seus colaboradores recomendam como sendo boas práticas para a segurança.

Palavras-chave: Oracle. Segurança. Banco de Dados. Sistema. Parâmetros. Análise.

ABSTRACT

This study establishes a critical view of the need to have a database within standards, based on good security practices published by the developers of DBMS's to a better use of such. In order to assist the database administrator in the database optimization process, thus improving the definition of good safety practices parameters, which in itself is not an easy task, but rather a complex task that requires skill and knowledge technician your administrator. So it will develop a system that will obtain, analyze, criticize and report on the results obtained in the analysis of safety parameters. The study specifically covers safe product configuration processes Oracle DataBase 11g and Oracle TNS Listener raised at Work Titling William Devide Komel authorship, and the use of such to feed the system, thus also leaving open to new studies that may be carried out and also feed the system with your data. Overall this study will enable a system in which any database administrator can get results on the security settings in the Oracle database, and all within parameters that Oracle and colleagues recommend as good practice for security.

Keywords: Oracle. Safety. Database. System. Parameters. Analysis.

SUMÁRIO

1 INTRODUÇÃO

Durante o século XX, ocorreu um aumento da importância da informação no âmbito social e individual. O computador passou a ser percebido como fundamental em muitas estruturas organizacionais. A estrutura de armazenamento e processamento dos dados passou a usar de diferentes formas para armazenar dados. Em diferentes formas e maneiras, os dados passaram a ser processados desenvolvendo as informações nos mais diferentes níveis.

Os bancos de dados evoluíram, e, na década de 80, tinham uma forma consagrada de armazenamento e representação denominado modelo relacional.

A questão de segurança dos dados evoluiu de modo mais acelerado na segunda metade do século XX. Segurança de dados passou a ser um fator importante no armazenamento e na disponibilização do acesso aos dados.

Percebendo-se a necessidade de organização das informações, foi necessária a criação de padrões e métodos para guardar as informações, a fim de que, futuramente, elas possam ser utilizadas para tomar decisões, garantindo, assim, o fácil acesso, segurança e integridade das mesmas.

De acordo com Feitosa (2013, p. 279),

> O primeiro modelo estrutural de bando de dados como conhecemos hoje e o conceito de SGDB foram propostos em 1961 pela CODASYL (Conference on Data System Languages) com o chamado “Modelo em Rede”, relativamente próximo do atual Modelo Relacional, mas na prática foi implementado e comercializado por um modelo simplificado do Modelo em Rede, conhecido como “Modelo Hierárquico”, constituído de uma cadeia de relacionamentos 1:N, onde cada filho só poderia ter um pai. Caso fosse necessário o relacionamento de um filho com mais de um pai, aquele registro teria que ser replicado no banco.

Com esses conceitos colocados em prática por meio do Modelo em Rede, no inicio dos anos 60 foi criado o primeiro Sistema Gerenciador de Banco de Dados (SGBD) relacional, o Sistema IDS (*Integrated Data Store*), que foi desenvolvido por Charles Bachman da empresa *General Eletric* (FEITOSA, 2013).

Neste sentido, “o IDS foi desenvolvido para operar como base de dados de um sistema de aplicação para controle da produção batizados de MIACS” (FEITOSA, 2013, p. 278).

Sendo assim, com esses SGBDs, viu-se a necessidade de uma mão de obra especializada para gerenciar os dados de maneira inteligente, pois cada SGBD demanda de um tipo de conhecimento para ser utilizado, cada um com suas singularidades e com diferentes parâmetros de controle, e além do grande fluxo de dados gerenciado por eles.

Uma prova disso é a Oracle Corporation, que se trata de uma das maiores desenvolvedoras de Sistemas Gerenciadores de Banco de Dados do mundo, sendo a detentora de aproximadamente 45% do mercado de Banco de Dados.

De acordo com Tavares (2006), a versão do Oracle Database 11g conta com mais de 400 recursos distintos voltados para o gerenciamento da administração de seus bancos de dados.

O profissional responsável por administrar um banco de dados é chamado de administrador do banco de dados (DBA - *Database Administrator*), que é responsável por fornecer o suporte técnico necessário para implementar a tomada de decisões a uma determinada empresa, sendo responsável pelo controle geral do sistema em um nível técnico (DATE, 2004, p. 35).

O DBA é um profissional qualificado para executar essas funções, porém, possui salário cujo valor é consideravelmente maior que os demais profissionais de tecnologia da informação, recebendo cerca de 100% a mais do que um programador da linguagem Java, 105% a mais que um Analista de Infraestrutura e 20% a mais que um Consultor de Planejamento de recurso corporativo (ERP – Enterprise Resource Planning) (APINFO, 2014).

Assim, as empresas deixam de contratar um DBA, e preferem contratar outro profissional que não é especialista em SGBDs, tendendo, assim, a maior possibilidade de se ocorrerem problemas devido à menor qualificação profissional, o que pode ocasionar maior vulnerabilidade dos dados e menor confiabilidade do sistema.

Sendo assim, viu-se a necessidade de se auxiliar os profissionais melhorarem a estrutura de um banco de dados, através de estudos realizados por Komel (2011), e da criação de um sistema que possibilite a comparação entre os parâmetros por ele definidos junto aos parâmetros difundidos pela criadora do SGBD como as melhores práticas de segurança.

Dessa forma, o resultado previsto é de que, a partir de informações consistentes, se consiga reduzir a vulnerabilidade do banco de dados administrado por esse profissional, facilitando, assim, o seu trabalho.

Problema

Segundo o levantamento dos parâmetros de configuração segura nos produtos Oracle Database 11g e no Oracle TNS Listener, realizado por Komel (2011), foi cogitada a ideia de que seria viável o desenvolvimento de uma ferramenta automatizada para os mesmos fins, pois, a utilização dos parâmetros de maneira manual não é fácil.

Por isso, foi gerado um software que possibilitasse o confronto de dados do banco do usuário com os valores por ele levantados, pois não há ferramenta automatizada que verifique se as configurações estão de acordo com as melhores práticas.

Contudo, o resultado final apresentado por Komel (2011) foi apenas um texto com suas conclusões acerca da viabilidade deste software, não facilitando, assim, o trabalho de verificação dos dados que compete ao DBA.

Desta forma, o presente trabalho apresenta uma solução viável através da automatização dos estudos realizados por Komel (2011) em um sistema que facilitará os resultados apresentados pelo mesmo.

Hipótese

Viabilizar a criação de um sistema que funcionará como uma ferramenta automatizada para confrontar os parâmetros levantados por Komel (2011) de forma automatizada e rápida, gerando um relatório com os resultados obtidos.

Objetivos

Gerar um sistema de validação de parâmetros de banco de dados acessado diretamente no navegador, ou seja, via web, no qual o usuário final fará o *upload* de seus parâmetros diretamente no sistema, especificando suas configurações, para

que o sistema faça o quadro comparativo que informe as melhores práticas dos parâmetros.

Desta maneira, o sistema apresenta um resultado final capaz de informar se o parâmetro está dentro dos limites recomendados, possibilitando que sejam corrigidos os dados necessários para que se melhore o desempenho do banco de dados.

Específicos

Definir o melhor modelo que atenda ao banco de dados Oracle facilitando suas configurações de segurança.

Desenvolver um sistema para comparar as variáveis levantadas pela análise do aluno William Komel (2011), com as variáveis de um banco de dados em funcionamento criado por qualquer usuário de internet.

Gerar relatórios, nos quais são exibidas as definições utilizadas pelo usuário, e se elas estão de acordo com os parâmetros comumente utilizados.

Auxiliar na tomada de decisões após a utilização e a comparação dos parâmetros do banco, sendo assim, exibindo sugestões para uma melhor configuração do banco do usuário.

Justificativa

Com o crescente ritmo de uso de sistemas computacionais baseados em bancos de dados, é cada vez mais frequente que uma falha ou esquecimento de um parâmetro de segurança possa ocorrer. Deste modo, justifica-se o desenvolvimento que dê apoio às tarefas de manutenção dos aspectos de segurança de banco de dados.

A ferramenta possibilita, por meio de um relatório automatizado, detalhar os parâmetros de segurança informados pelo usuário. Informa, no entanto, se os mesmos estão de acordo com a melhores práticas, ou ainda, indica a necessidade de se iniciar um novo procedimento até a configuração correta.

Dessa maneira, a configuração de um banco de dados é facilitada, proporcionando um auxílio no desenvolvimento da definição dos parâmetros.

Este trabalho apresenta uma estrutura de desenvolvimento dividida por capítulos distintos, para a organização das informações coletadas buscando uma melhor explanação dos assuntos abordados.

No capítulo 1 é realizada uma introdução, definindo aspectos gerais do projeto, no qual será desenvolvido um sistema vindo de uma necessidade de uma configuração adequada de um banco de dados. No capítulo 2 explana-se uma revisão bibliográfica sobre os temas pertinentes ao desenvolvimento do projeto. No capítulo 3 é abordada a especificação da ferramenta de software objeto deste projeto. No Capítulo 4 apresenta-se uma explanação detalhada de alguns aspectos importantes dos algoritmos desenvolvidos. No capítulo 5 faz-se a Conclusão do trabalho. Nos Apêndices apresentam-se os códigos-fontes mais importantes desenvolvidos.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo demonstra a linha de pesquisa adotada pelos autores no desenvolvimento do projeto, e proporciona um melhor entendimento do trabalho por parte do leitor e do usuário da aplicação.

2.1 Banco de Dados

O termo banco de dados pode ser definido como sendo “uma coleção de dados persistentes, usada pelos sistemas de informação de uma determinada empresa” (DATE, 2004, p. 10). O autor complementa, ainda, que:

> Um sistema de banco de dados é basicamente apenas um sistema computadorizado de manutenção de registros. O banco de dados, por si só, pode ser considerado como o equivalente eletrônico de um armário de arquivamento; ou seja, ele é um repositório ou recipiente para uma coleção de arquivos de dados computadorizados. (DATE, 2004, p. 03).

Já Silberschatz, Korth e Sudarshan (1999, p. 04) se referem a um sistema de banco de dados dizendo que “é uma coleção de dados inter-relacionados e um conjunto de programas que permitem aos usuários acessar e modificar esses dados”.

Além disso, é importante destacar a importância dos sistemas de bancos de dados, e também seus benefícios, quais sejam:

> Os sistemas de bancos de dados oferecem diversos benefícios, dos quais um dos mais importantes é a independência de dados (física). A independência de dados pode ser definida como a imunidade de programas de aplicação a alterações no modo de armazenar fisicamente os dados e obter acesso a eles. (DATE, 2004, p. 25).

A partir daí, evidencia-se o surgimento da necessidade de que os dados sejam melhor organizados, para facilitar para os administradores, pois, de acordo com Heuser (1998, p. 04):

> O compartilhamento de dados tem reflexos na estrutura do software. A estrutura interna dos arquivos passa a ser mais complexa, pois estes devem ser construídos de forma a atender às necessidades dos diferentes sistemas. Para contornar este problema, usa-se um sistema de gerência de banco de dados.

Os SGBD's surgem como ferramentas que possibilitam a manipulação e facilite a interação com o usuário. Encontramos vários exemplos disponíveis no mercado, como: Oracle Database 11g, SQL server e até mesmo o Access, que ambos desempenham essas funções.

Nesse sentido, sobre o SGBD, afirma Date (2004, p. 36) ainda, que o mesmo "deve ser capaz de aceitar definições de dados (esquemas externos, o esquema conceitual, o esquema interno e todos os mapeamentos associados) em formato fonte e convertê-los para o formato objeto apropriado".

É importante diferenciar, ainda, o administrador de dados (DA – *Data Administrator*) do administrador do banco de dados (DBA – *Database Administrator*). Date (2004, p. 35) aclara os termos da seguinte maneira:

> O administrador de dados (DA – Data Administrator) é a pessoa que toma as decisões estratégicas e de normas com relação aos dados da empresa, e o administrador do banco de dados (DBA – Database Administrator) é a pessoa que fornece o suporte técnico necessário para implementar essas decisões.

Isso significa dizer que cabe ao DBA controlar de maneira geral o sistema, utilizando seus conhecimentos técnicos.

2.2 Parâmetros de banco de dados

O sistema a ser desenvolvimento foi sugerido em um trabalho de levantamento de tais parâmetros. Sendo dada continuidade à monografia de William Devidé Komel, titulado **"Software de análise de segurança de banco de dados oracle".**

Com base nesses levantamentos, são realizadas as entradas, bem como o cadastramento dos parâmetros corretos a serem avaliados, e, por fim, serão comparados com os padrões do usuário final.

> Tendo jurisdição completa sobre o banco de dados, o DBA (sob orientação apropriada do administrador de dados) pode assegurar que o único meio de acesso ao banco de dados seja através dos canais apropriados e, em consequência, pode definir restrições de segurança a serem verificadas sempre que houver uma tentativa de acesso a dados confidenciais. (DATE, 2004, p. 17).

As recomendações de segurança quanto aos parâmetros de rede do TNS Listener, dizem respeito a análise dos nós que tentam estabelecer conexão com o banco de dados e o tempo que os mesmos tem para completar o estabelecimento dessa conexão (KOMEL, 2011).

Além disso, Date (2004, p. 37) estabelece que o SGDB “deve monitorar requisições de usuários e rejeitar toda tentativa de violar as restrições de segurança e integridade definidas pelo DBA”.

2.3 Requisitos de Segurança a um Banco de Dados

A segurança é uma questão essencial para qualquer tipo de software, sendo necessário à empresa que o desenvolve criar um manual de segurança com toda a documentação, regras, e procedimento para melhor configuração do tal.

Segundo Watson (2010, p. 436) em segurança não há certo ou errado, há somente a conformidade ou a não conformidade de um procedimento, por tanto qualquer administrador que seguir as regras e orientações estipulados em um manual de segurança, caso vir ocorrer uma falha de segurança, não será sua falha.

Ainda segundo Watson (2010, p.436) nos produtos Oracle isso ocorre de uma maneira natural pois a Oracle fornece muitos recursos para implementação de segurança, muito além dos padrões especificados por qualquer legislação.

Porém, segundo Bryla e Loney (2009, p. 303), apesar de todas as configurações de segurança, ainda é necessário a segurança de acesso ao sistema operacional, além do hardware físico estar em um local seguro.

É preciso especificar os privilégios dos usuários e ainda a estrutura de contas, que, segundo Bryla e Loney (2009, p.159):

> A equipe de desenvolvimento deve especificar a estrutura de contas que a aplicação usará, incluindo o proprietário de todos os objetos na aplicação e a maneira como os privilégios serão concedidos. Todas as atribuições e privilégios deve ser claramente definidos. Os resultados dessa seção serão usados para gerar a estrutura de contas e privilégios da aplicação de produção.

O planejamento de segurança é uma área na qual o DBA é muito importante, ele deve ser capaz de projetar uma implementação que atenda todas as

necessidades de uma determinada aplicação, e ajustá-la aos planos de segurança de banco de dados da empresa (BRYLA; LONEY, 2009, p.159).

2.4 Linguagem de Programação PHP

A linguagem de programação PHP é a terceira linguagem de programação de maior empregabilidade (APINFO, 2014), sendo assim é indispensável o estudo da tal, assim como o desenvolvimento de aplicações para melhor preparo no mercado de trabalho.

Desenvolvedores PHP se beneficiam por ser uma das linguagens mais utilizadas na web. Segundo Niederauer (2011, p. 23) a principal diferença em relação às outras linguagens é a capacidade que o PHP tem de interagir com o mundo web, transformando totalmente os websites que possuem páginas estáticas. Umas das grandes vantagens do PHP é que ele é gratuito e com o código-fonte aberto, e toda sua documentação detalhada está disponível em seu site oficial.

Ainda segundo Niederauer (2013, p. 21) uma página PHP não contém apenas códigos de programação PGP, mas contem *tags* de marcação HTML, com o PHP representando a parte dinâmica da página e o HTML representando a parte estática, ou seja, a saída HTML sempre será a mesma quando acessar uma página, já a saída PHP pode ser diferente a cada acesso.

2.5 HTML

Criada em 1990 por Tim Berners-Lee, um cientista inglês, como uma linguagem que serviria para interligar computadores do laboratório com outras instituições de pesquisa e exibir documentos científicos de formas simples e fácil de acessar.

A linguagem HTML (*HyperText Markup Language*) tornou-se um padrão para o desenvolvimento de web sites na Internet, sendo uma linguagem de marcação e não de programação. É uma linguagem de fácil aprendizado, não exigindo muito de um hardware potente para o seu desenvolvimento, e nem mesmo algum software específico para tal.

> Para um bom entendimento das definições, podemos resumir hipertexto como todo o conteúdo inserido em um documento para a web e que tem como principal característica a possibilidade de se interligar a outros documentos da web. O que torna possível a construção de hipertextos são os links, presentes nas páginas dos sites que estamos acostumados a visitar quando entramos na internet. (SILVA, 2008, p. 26).

Com isso, tornou-se muito fácil navegar na web, com a facilidade do HTML, que permite títulos, cores, imagens, textos diferenciados, o HTML cumpre o papel de interação do usuário com a web.

2.6 CSS

O *Cascading Style Sheets* (CSS) ou, no português, folhas de estilo em cascata. A definição de CSS no site da W3C afirma que:

> Folha de estilo em cascata é um mecanismo simples para adicionar estilos (por exemplo: fontes, cores, espaçamentos) aos documentos web. A finalidade de se utilizar CSS é devolver ao XHTML o propósito inicial da linguagem, que era gerar apenas o texto na página. (SAVOIA, 2013, p. 29).

Ainda segundo SAVOIA (2013, p. 29), cabe às CSS as funções de apresentação de uma página web ou um documento. Para inserção de um CSS a uma página existem dois métodos: um deles é a referenciação dentro do *head*, do arquivo .css, e o outro é incluir dentro do *head* a *tag style* e colocar todo o código CSS dentro dela.

2.7 MySQL

O MySQL é um SGBD muito utilizado por todo o mundo, sendo um sistema de código aberto e o mais popular para aplicações web, sendo rápido, confiável e fácil de usar (ORACLE, 2016).

Segundo Suehring (2002, p. 43) o MySQL possui um cenário de melhor dos mundos, pois é executado em várias plataformas, e possui um baixo TCO, e alta estabilidade, contendo uma documentação completa e excelente, com bons

materiais de referência, com suporte de alta qualidade para seus produtos e serviços.

Outra vantagem que torna o MySQL um dos mais utilizados no mundo, é que os serviços são gratuitos, sendo um software livre com código fonte aberto, e livre para ser instalado em vários servidores diferentes.

2.8 Apache

Criado por um grupo chamado Apache Group, ou também conhecida como Fundação Apache. O Apache é um dos softwares mais utilizados nos dias de hoje na internet. Sendo um dos mais seguros e robustos programas desenvolvidos para ambientes TCP/IP, e faz parte da operação de mais de 60% dos sites disponíveis na web no mundo (MARCELO, 2005, p. 3).

Ainda segundo Marcelo (2005, p. 4), as principais vantagens do apache são, o suporte a HTTP 1.1, para criação de *virtual hosts* baseados em DNS, suporte a *Secured Socket Layer* (SSL) para transações seguras, como também a CGI's, Perl, autenticação HTTP, *Server Side Includes* (SSI), *Servlets Java*, logs customizáveis, e além de suporte a PHP, que foi a linguagem utilizada para desenvolvimento do Sistema de Validação de Parâmetros de Banco de Dados.

2.9 JavaScript

A linguagem JavaScript foi criada pela *Netscape Communications Corporation,* foi desenvolvida com o nome de Mocha, depois passou a se chamar LiveScript e finalmente em 1995 foi lançada como JavaScript, no navegador NetScape, visando implementar uma tecnologia de processamento no modo cliente.

É uma linguagem de script, que consiste em scripts que também são colocados em páginas web, no meio do HTML, é um tipo de programação executada no lado do cliente da aplicação, que ao abrir seu browser (navegador) e ao acessar uma página, ela é carregada na memória de sua máquina, e o código JavaScript é executado consumindo os recursos de processamento do seu computador.

Definida como um tipo de programação, pode ser vista por qualquer pessoa ao acessar uma página da web, somente ao escolher a opção no navegador exibir > código fonte (NIEDERAUER, 2011, p. 25).

Atualmente JavaScript é usado em praticamente toda web, uma empresa que utiliza JavaScript em praticamente todos os seus aplicativos é a Google Inc., como por exemplo: o site de buscas Google Search, Gmail, Google Maps, entre outros.

2.10 jQuery

jQuery é uma biblioteca JavaScript, criada por Johm Resig, e disponibilizada como software livre e aberto, significa que qualquer pessoa pode usar a biblioteca gratuitamente tanto para desenvolver projetos pessoais ou comerciais (JQUERY, 2016).

> Simplicidade é a palavra-chave que resume e norteia o desenvolvimento com jQuery. Linhas e mais linhas de programação JavaScript escritas para obter um simples efeito em um objeto são substituídas por apenas algumas, escritas com sintaxe jQuery. Intrincados e às vezes confusos códigos JavaScript destinados a selecionar um determinado elemento HTML componente da árvore do documento são substituídos por um simples método jQuery. (SILVA, 2013, p. 26).

A biblioteca jQuery apresenta uma grande facilidade e auxílio para o programador. Ao contrário de desenvolver muitas linhas de código para se alcançar um resultado, é facilitado pelo jQuery, que, ao desenvolver poucas linhas de código se desenvolve a mesma coisa, gerando, assim, maior rentabilidade e agilidade no trabalho.

2.11 UML

A UML (Linguagem Unificada de Modelagem) é uma linguagem gráfica para visualização, especificação, construção e documentação de processos de um software, proporcionando uma forma padrão na preparação de planos de arquitetura no projeto de sistemas. (BOOCH; RUMBAUGH; JACOBSON, 2006, p.13).

Ainda segundo Booch, Rumbaugh e Jacobson (2006, p.13) a UML inclui conceitos de negócios, funções do sistema, e até classes escrita em linguagem de programação, dando uma visão sistêmica do software a ser desenvolvido.

2.12 Astah

O Astah é uma ferramenta para desenvolvimento e elaboração de diagramas UML de maneira simples e prática (ASTAH,2016). Anteriormente, era denominado JUDE (Java and UML Developers Environment), desenvolvido na plataforma Java, garantindo sua portabilidade independente da plataforma.

Possui variadas versões como Astah Community, Astah UML, Astah Professional e Astah Share, diferenciando entre elas o seu tipo de licença, versão e funcionalidades, para atender variados tipos de publico (ASTAH,2016).

2.13 Modelo Cascata

O modelo cascata é o paradigma mais antigo da engenharia de software e foi proposto por Winston Royce em 1970 (PRESSMAN, 2006, p.39).

O modelo cascata, também chamado de ciclo de vida clássico, sugere uma abordagem sequencial e sistemática para o desenvolvimento de um software, desde o levantamento das necessidades do cliente que precisa do software, durante todo o planejamento, modelagem, construção, e uso, até a manutenção após o sistema ser concluído (PRESSMAN, 2011, p.59). Conforme a Figura 1 abaixo:

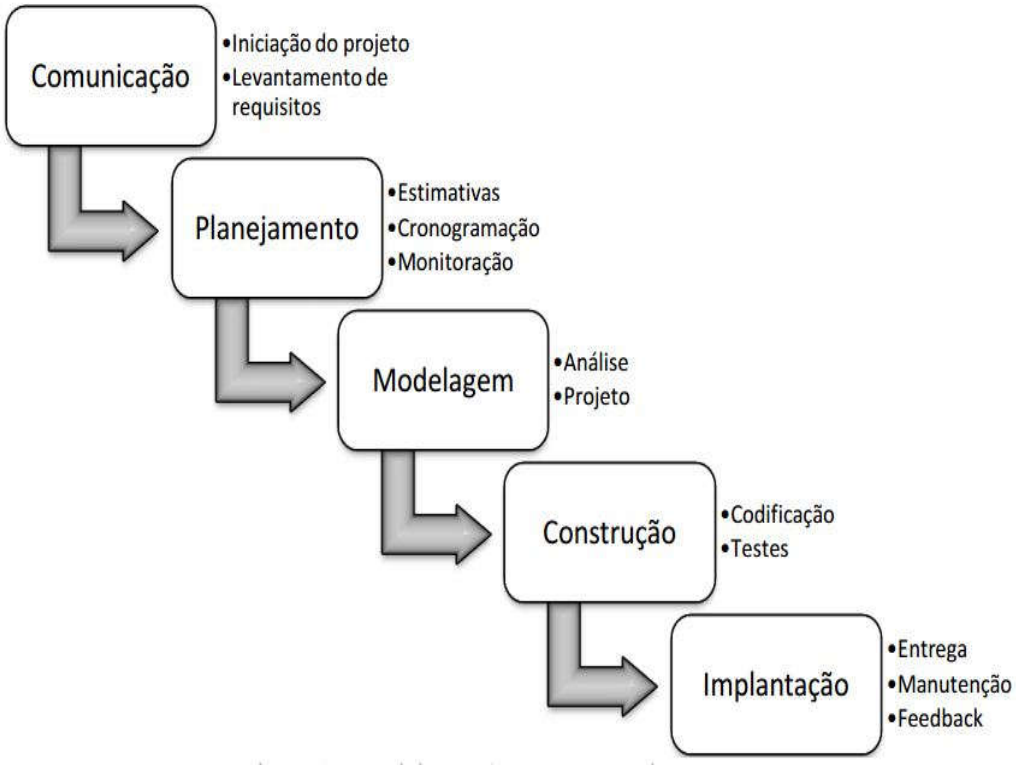

Figura 1- Modelo em cascata (PRESSMAN, 2006, p.39)

A grande vantagem desse método é que o projeto só avança de etapa após validado e com as devidas funcionalidades estando corretas, evitando assim complicações nas próximas etapas do projeto.

Porém se ocorrerem falhas, ou atraso em alguma etapa, a próxima etapa do projeto fica impedida de inicializar até que os membros responsáveis pela etapa anterior concluam devidamente a etapa necessária, e isso ocorre devido ao modelo linear de desenvolvimento do método.

2.14 Trabalhos correlatos

O trabalho de monografia de William Devidé Komel (2011), titulado "**Software de análise de segurança de banco de dados oracle",** sugere que dentro do mercado de banco de dados é viável uma ferramenta automatizada que receba

esses dados que já foram levantados por ele e os utilize para confrontar e validar os feitos pelo usuário que venha a utilizá-la.

Vendo que a implementação tanto profissional quanto educacional é aplicável, dando continuidade ao seu trabalho, também é acrescentado que a ferramenta fique disponível via WEB, para um acesso livre a todos, para automatização dos estudos realizados por ele.

Visando atender esses profissionais, viu-se a possibilidade de desenvolver um software que analise os parâmetros já configurados em um banco de dados Oracle e forneça respostas quanto a essa configuração, de acordo com resultados considerados corretos pelo fabricante. Assim, possibilitando que o profissional que utilize essa ferramenta possa determinar assim a necessidade ou não de alterações nos parâmetros analisados (KOMEL, 2011).

3 MÉTODO

Neste capítulo serão descritos os métodos e ferramentas utilizadas para o desenvolvimento do Sistema de Validação de Parâmetros de Banco de Dados.

3.1 Ambiente

O ambiente utilizado para aplicação serão os laboratórios da FATEC Ourinhos, estando também disponível via internet para ser acessado por qualquer usuário da rede que tenha interesse de utilizar.

3.2 Participantes

Serão participantes ativos do projeto os alunos da instituição e pessoas diversas que tenham interesse de contribuir com o projeto, podendo ser tanto do sexo masculino quanto feminino.

Como usuários do sistema os participantes serão divididos em:

- *Root*: terá poderes de inclusão, alteração, exclusão e consulta de todas as informações cadastradas no sistema.
- Administrador: terá poderes de inclusão, alteração e consulta de todas as informações cadastradas no sistema.
- Usuário Padrão: Consultar informações cadastradas no sistema.

3.3 Materiais e instrumentos

Para o desenvolvimento e aplicação do projeto foram utilizados equipamentos físicos como notebooks e microcomputadores, no ambiente Windows, e com instalações do ambiente de desenvolvimento (IDE) Brackets com os pacotes necessários para o desenvolvimento e conexões, sendo eles PHP, HTML, CSS,

JavaScript, JQuery, utilizando o banco de dados MySQL, servidor web apache e uso de Adobe Photoshop CS6 para edições gráficas.

3.4 Procedimento

O modelo de processo de desenvolvimento do software a ser empregado foi o modelo de cascata. Começando a criação a partir de uma abordagem sistemática e sequencial, iniciado pela especificação dos requisitos, seguido pela modelagem, construção, implantação e manutenção.

4 RESULTADOS E DISCUSSÕES

Este capítulo objetiva descrever todo o desenvolvimento e resultados obtidos na construção do sistema, bem como os requisitos, diagramas, que serviram de base para o levantamento das reais necessidades de desenvolvimento do software, e os resultados obtidos através dele.

4.1 Documentação do Sistema

Este tópico tem como objetivo descrever toda a documentação do sistema, incluindo seus diagramas, requisitos.

4.1.1 Descrição do problema

Muitos usuários e administradores de banco de dados possuem um banco ativo, entretanto, não trabalham nas melhores práticas do banco em uso, pois trabalham com práticas de experiências passadas, ou até mesmo com práticas padrões do sistema.

Com o Sistema de Validação de Parâmetros de Banco de Dados via Web, qualquer pessoa pode realizar um teste com seu banco, pode realizar o *upload* dos parâmetros do seu banco e testá-los no sistema.

Por sua vez, o sistema irá realizar uma comparação entre os parâmetros do tal banco e as melhores práticas definidas por convenção, e apontar quais as variáveis devem ser alteradas, e propor soluções úteis para um melhor desempenho.

4.1.2 Objetivo do Sistema

Este sistema permitirá a qualquer usuário realizar comparações entre parâmetros de um banco e ver as melhores práticas do sistema de banco de dados utilizado por ele, tendo assim soluções práticas para melhorar o desempenho de banco.

4.1.3 Principais Envolvidos e suas Características

Os principais envolvidos são quaisquer tipos de usuários que administre, ou estude algum banco de dados em especifico, e busca as melhores práticas e melhorar o seu próprio banco.

4.1.4 Usuários do Sistema

Livre para o uso de qualquer empresa ou pessoa física que deseja aumentar o desempenho do seu banco de dados. Isso, pois, os administradores do sistema irão gerar os padrões de melhores práticas por meio de pesquisas, e métodos para gerar o melhor desempenho no banco, e tudo isso gerenciado por um root que irá gerar as permissões dos usuários do sistema.

4.1.5 Desenvolvedores do Sistema

Os desenvolvedores envolvidos serão os alunos que iniciaram o projeto com auxilio de professores que irão aconselhar e direcionar os alunos pelo melhor método de desenvolvimento e andamento de todo o sistema.

4.1.6 Regras de Negócio

- O usuário deverá acessar o sistema via internet, fazer o upload dos parâmetros do seu banco e o sistema deverá gerar um relatório.
- O administrador irá definir os parâmetros de melhores práticas de um determinado banco de dados, e irá defini-lo como padrão no sistema.
- O *root* irá gerenciar os usuários e os acessos, e contar com o acesso a internet para o devido funcionamento do sistema.

4.1.7 Levantamento de Requisitos

Tomando por base o contexto do sistema, foram identificados os seguintes requisitos que auxiliaram no desenvolvimento do sistema:

4.1.7.1 Requisitos Funcionais

Os requisitos funcionais são os pontos levantados pelo cliente, são as condições necessárias para o funcionamento do sistema, as necessidades que o sistema deverá atender, é o que o cliente realmente quer que o sistema faça.

Esta etapa define explicitamente o que cada função do sistema irá fazer. Esse requisito é de extrema importância, uma vez que é nele onde será cumprida a real necessidade do cliente. Os requisitos levantados serão apresentados abaixo.

Observações:

Dados marcados com * (asterisco) serão considerados campos de preenchimento obrigatório.

RF01 – Manter usuários

1.1 O sistema deve permitir a inclusão, alteração e consulta dos usuários e de dados dos tais.

1.2 Para melhor organização das funções do sistema os usuários serão divididos em quatro tipos: root, administrador, e usuário padrão.

1.3 Root: terá acesso ao cadastro de usuários administradores, alteração de padrões de servidor, controlar o acesso de usuários padrões (que não necessitarão de cadastro), e terá acesso a todas as funcionalidades do sistema.

1.4 Administrador: terá acesso a uma tela exclusiva para adicionar as boas práticas de um determinado banco de dados, receberá um ID e uma senha do usuário root, que será utilizada para realizar esse acesso exclusivo.

1.5 Usuário Padrão: não precisará de cadastro, terá acesso a tela via web, acessado por um link, no qual ele fará o upload dos parâmetros do seu banco conforme RF04, e através dele o sistema irá gerar um relatório das alterações necessárias conforme RF06.

1.6 Senha: Aceitando qualquer tipo de caractere, mínimo 6, máximo 12, de preenchimento obrigatório;

1.7 Status: Aceitando apenas um caractere, sendo A (Ativo) e I (Inativo), de preenchimento obrigatório.

RF02 – Efetuar Login

2.1 Antes de um root ou administrador do sistema terem acesso a áreas exclusivas do sistema será efetuado obrigatoriamente o login, neste processo de login deverá conter o ID e a senha conforme o item RF01.

2.2 O sistema terá como usuário inicial o usuário root, com a senha root123, que deverá ser alterado posteriormente, e através desse usuário será possível o cadastro dos novos usuários administradores.

RF03 – Cadastrar boas práticas

3.1 O sistema deve permitir que os usuários administradores, realizem o cadastro dos parâmetros de boas praticas.

3.2 O cadastro das boas práticas deve conter as seguintes informações: Nome do banco*, versão do banco* e os parâmetros.

3.3 Nos parâmetros deverá conter: nome do parâmetro*, valor mínimo*, valor máximo*, solução valor mínimo*, solução valor máximo*.

RF04 – Realizar upload de arquivos

4.1 O sistema deverá permitir que o usuário padrão realize o upload do arquivo dos parâmetros gerados no banco de dados.

4.2 O tamanho do arquivo não poderá passar de 5 MB.

RF05 – Comparar parâmetros

5.1 O sistema irá realizar a leitura e fazer a comparação com as boas práticas já cadastradas no sistema, gerando, assim, um relatório para o usuário padrão, conforme RF 06.

RF06 – Gerar relatório

6.1 O sistema deverá gerar um relatório com o resultado da comparação dos parâmetros conforme RF 05, ao usuário padrão.

6.2 O relatório deverá ser gerado em um arquivo, e será enviado via email, a partir disso não sendo mais os desenvolvedores responsáveis por manter o arquivo em um servidor.

4.1.7.2 Requisitos Não Funcionais

RNF 01 Controle de acesso

O sistema deverá permitir o acesso e execução de operações quando a senha for fornecida de acordo com o cadastro de usuário.

RNF 02 Persistência

O sistema deverá permitir que os dados incluídos, alterados ou excluídos sejam atualizados em um banco de dados relacional, de acordo com as especificações designadas.

RNF 03 Interoperabilidade

O sistema devera se comunicar com outro um sistema operacional, independentemente de qual seja.

RNF 04 Hardware

O hardware terá de ter compatibilidade com o sistema, sendo necessário para execução de um navegador para acesso do sistema.

RNF05 Rede

O microcomputador deverá estar conectado com a rede mundial de computadores, por meio da internet.

RNF06 Segurança

O sistema deverá garantir o anonimato e a integridade das informações dos usuários cadastrados.

4.1.8 Modelo Entidade Relacionamento

Também conhecido pela sigla MER, é a representação gráfica da estrutura dos dados, sendo um modelo de um SGBD, que se baseia no princípio que todos os dados estão guardados em tabelas.

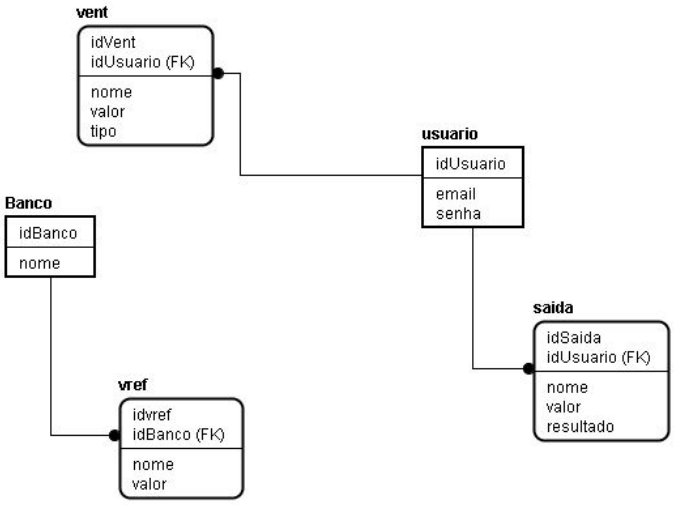

Figura 2- Modelo entidade relacionamento do Sistema

De acordo com a *Figura 2* o banco de dados utilizado no sistema apresentará quatro tabelas nas quais os dados serão armazenados.

4.1.9 Diagrama de Atividades

O diagrama de atividades tem como objetivo mostrar o fluxo de atividades em um único processo, mostra como uma atividade depende da outra.

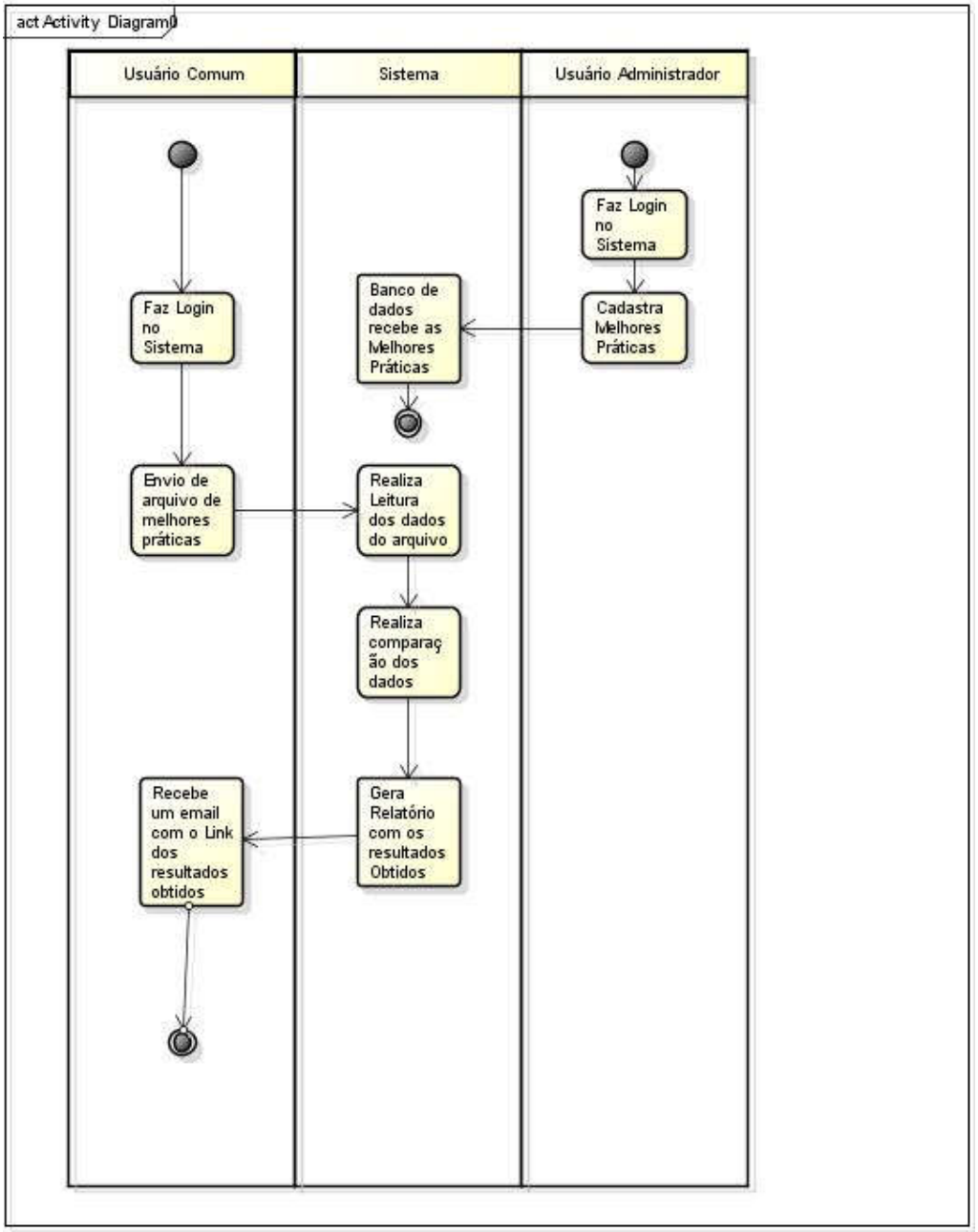

Figura 3 - Diagrama de Atividades do Sistema

4.1.10 Diagrama de Componentes

O diagrama de componentes mostra vários componentes de um sistema e suas dependências, com uma visão estática de como o sistema será implementado, em seus meios físicos e lógicos.

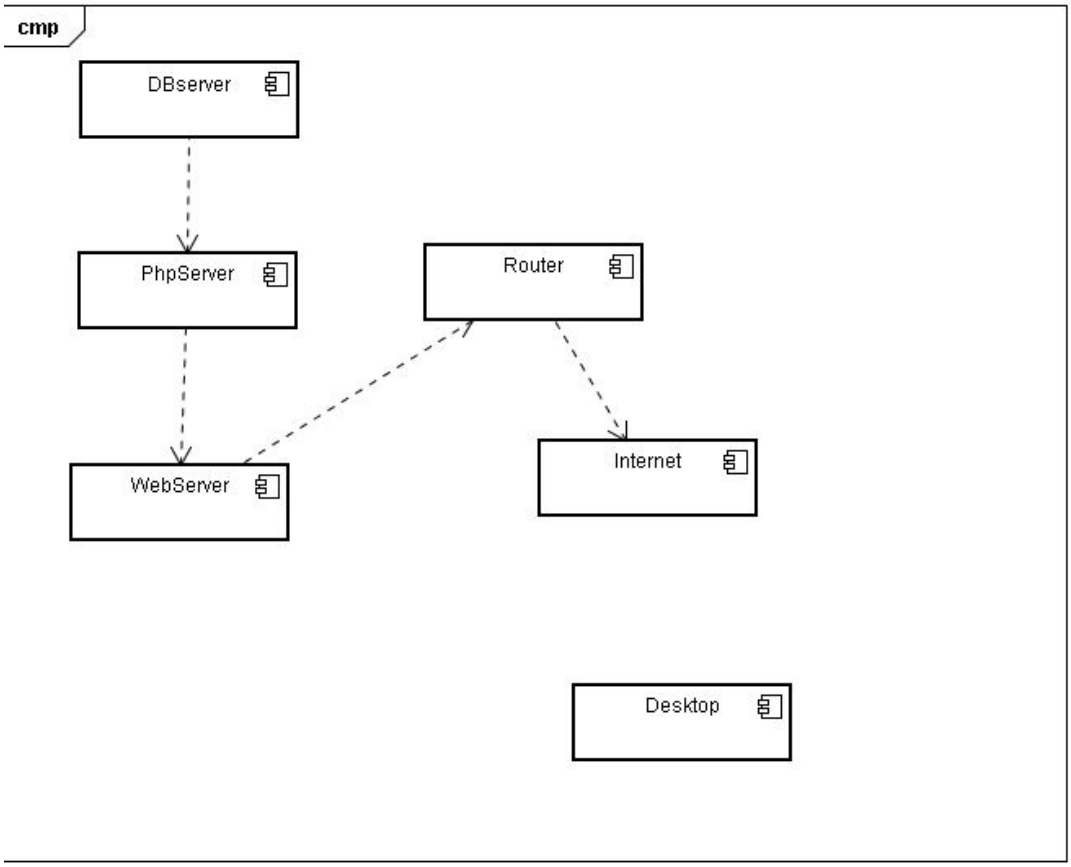

Figura 4 - Diagrama de componentes do Sistema

4.1.11 Diagrama de Sequência

Um diagrama de sequência mostra uma interação, que representa a sequência de mensagens entre as instâncias de classes, componentes, subsistemas ou atores através do tempo.

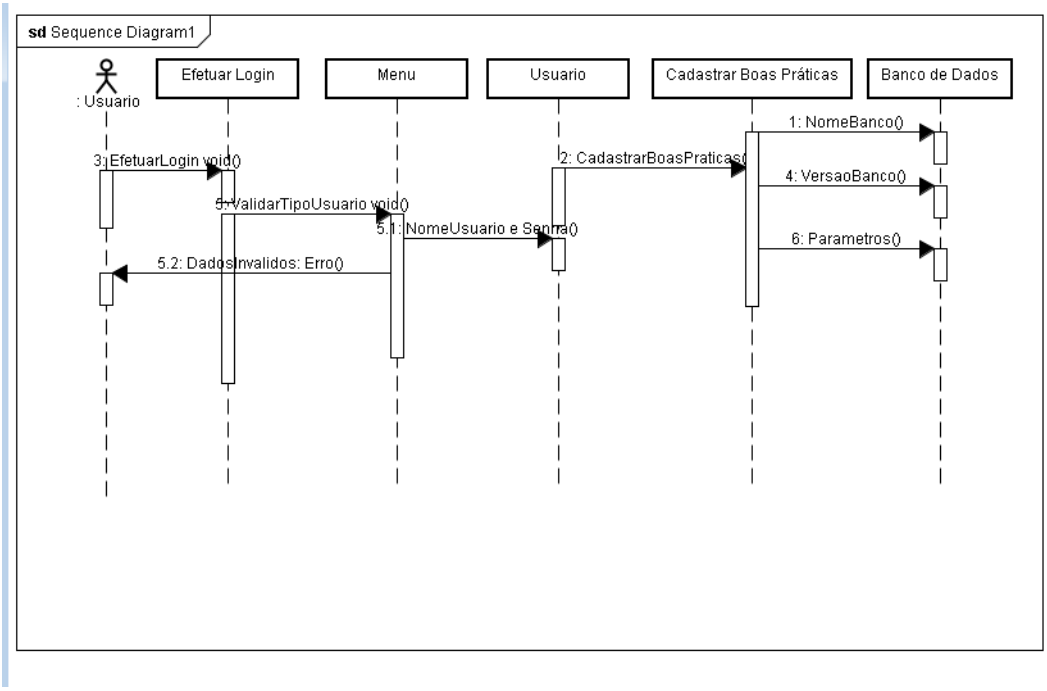

Figura 5 - Diagrama de sequência para cadastro de boas práticas

Na figura 5 é representado o cadastro de boas práticas através de um usuário com o acesso devido, na qual a informação é captada através de uma tela de cadastro, e depositada no banco de dados para usos eventuais.

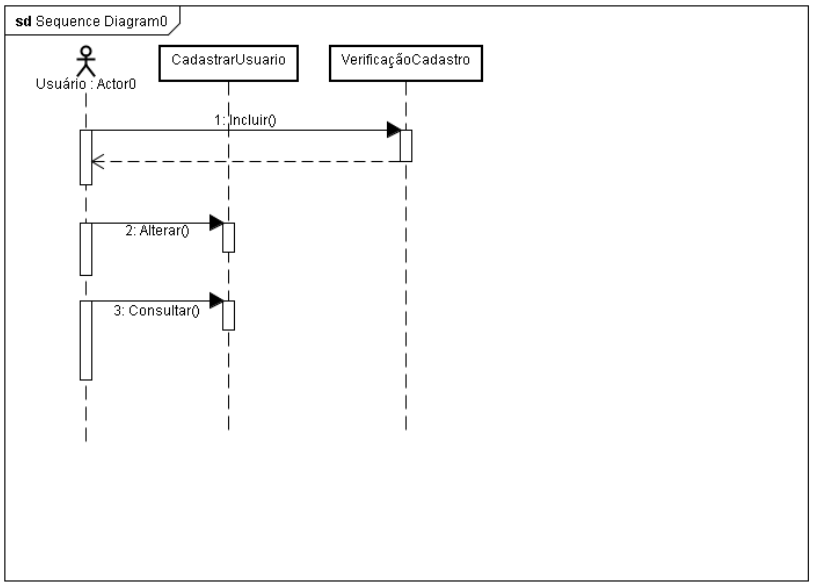

Figura 6 - Diagrama de sequência para cadastro de usuários

Na figura 6 é representado o diagrama de sequencia para cadastro de novos usuários no sistema, através de uma tela de inclusão de dados.

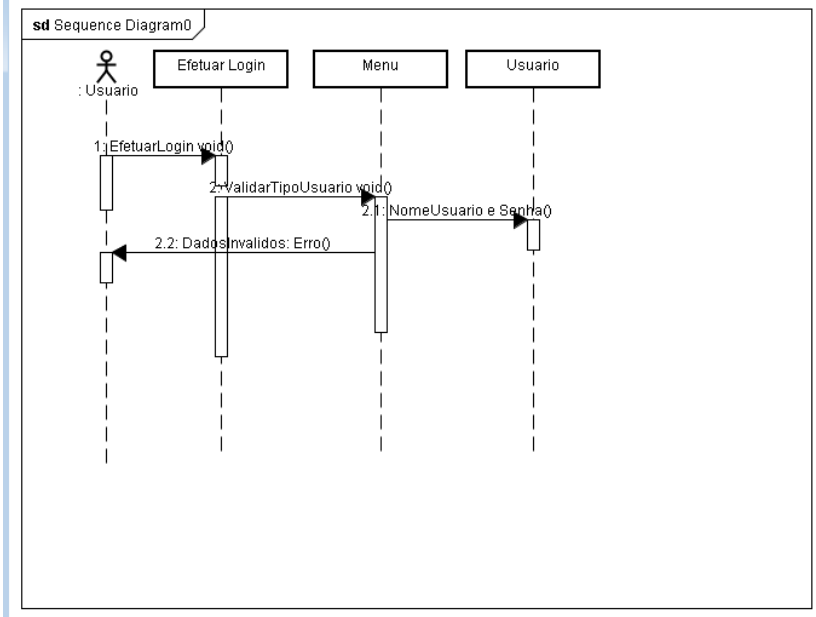

Figura 7 - Diagrama de sequência para Efetuar login

Na figura 7 é representado o diagrama de sequência para um usuário efetuar login no sistema.

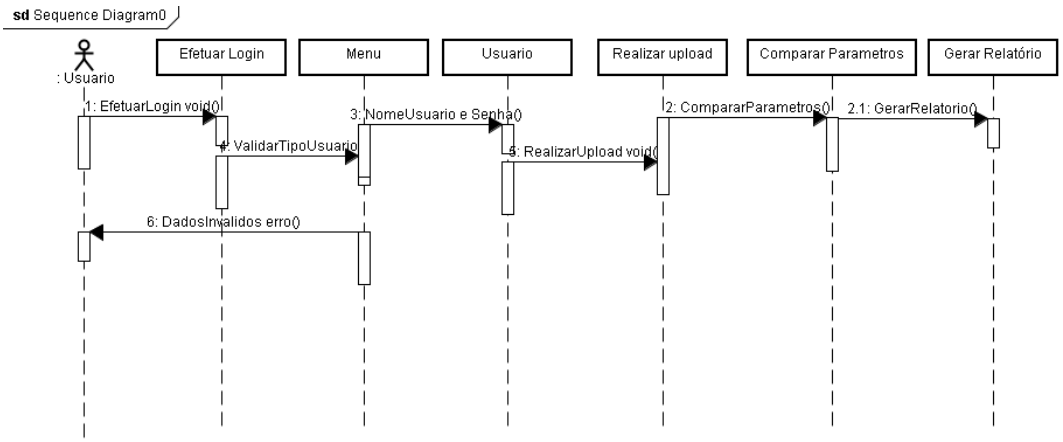

Figura 8- Diagrama de sequência para gerar relatório

Na figura 8 é representado um diagrama de sequência para a geração de relatórios no sistema, após realizar a comparação de parâmetros e gerar os resultados, é gerado um relatório que será enviado automaticamente para o email cadastrado do usuário.

4.1.12 Diagrama de Caso de Uso

O diagrama de Caso de Uso, visa capturar e descrever as funcionalidades que um sistema deve apresentar para que os usuários (atores) interajam com o mesmo.

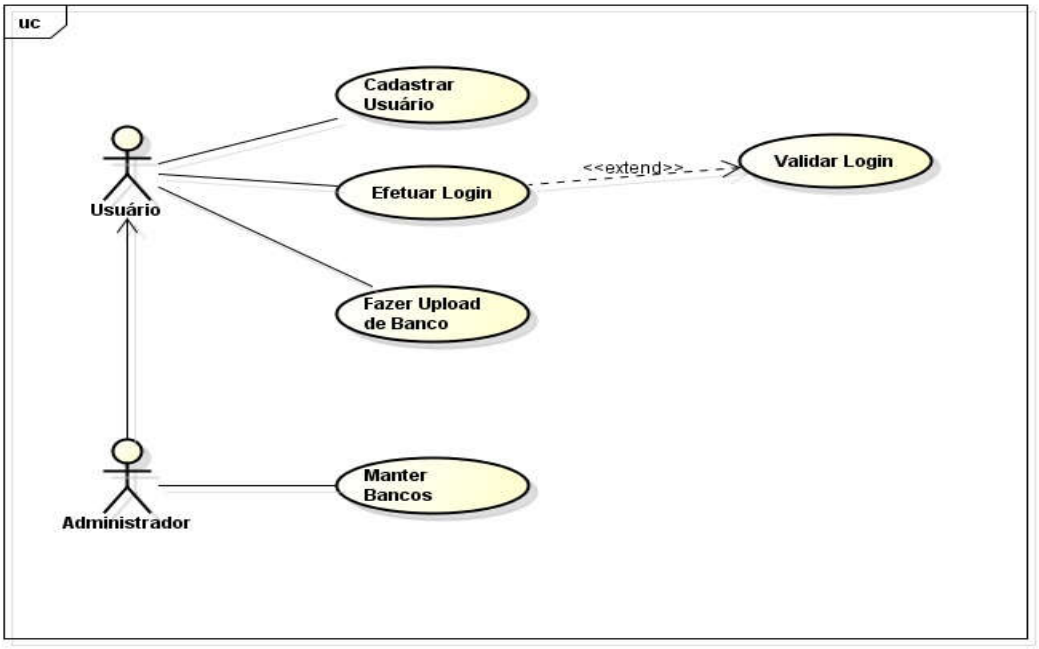

Figura 9 - Caso de uso geral do Sistema

Na figura 9 é apresentado o caso de uso geral do sistema, no qual é realizado as interações do usuário padrão, onde ele pode se cadastrar, efetuar o login, e realizar o upload de um arquivo de parâmetros de segurança para que o sistema realize os testes necessários e desenvolva o feedback. É o Administrador que mantém os bancos, fazendo o cadastro das melhores práticas.

4.2 Resultados do sistema

Com o desenvolvimento do Sistema de Validação de Parâmetros de Banco de dados obteve-se como resultado um produto final, que será abordado nesse tópico.

4.2.1 Telas

As telas principais do sistema serão abordadas nesse subtópico. Sendo elas na seguinte ordem abaixo, tela de cadastro, login, upload, e resultados sendo possibilitado o download dos resultados:

Figura 10 - Tela de login

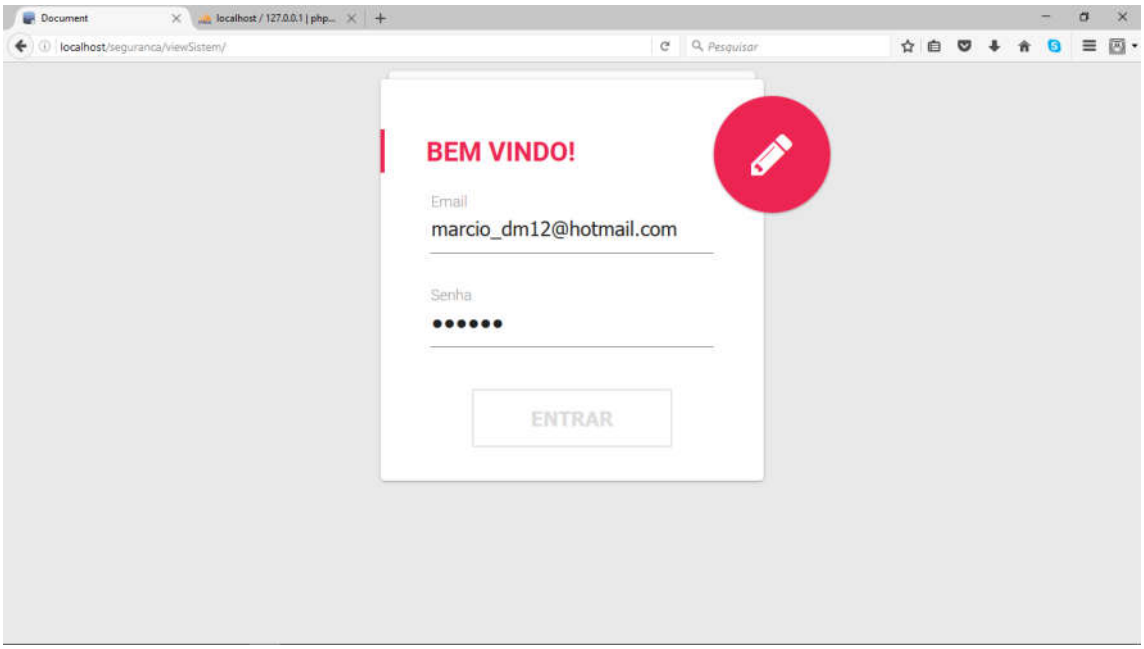

Figura 11 - Tela de cadastro

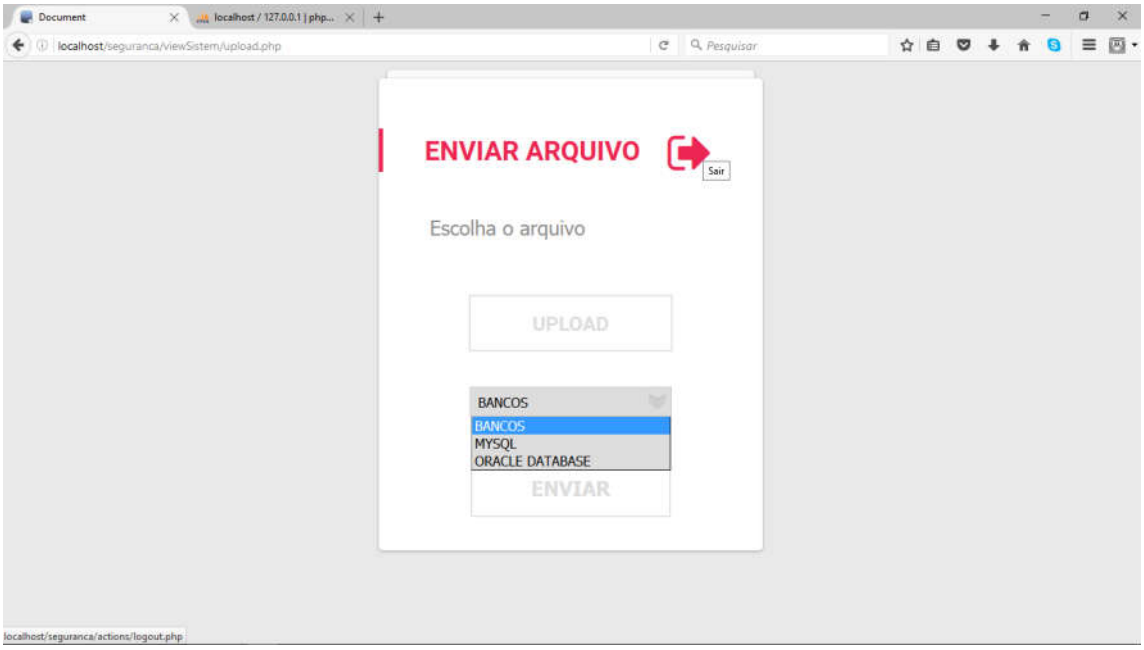

Figura 12 - Tela de upload

Figura 13 - Tela de resultados

4.2.2 Código fonte

Nesse sub tópico apresentam-se as partes lógicas importantes para o funcionamento do sistema.

A seguir é apresentado parte do código fonte utilizado para recebimento do upload de um arquivo CSV para uma estrutura de tabelas no banco através da criação de uma tabela temporária para comparação dos dados:

function temporaria($id,$file){ //Aqui jogamos passamos os dados do arquivo CSV para a tabela de entrada no Banco!

$fh = fopen($file['tmp_name'],'r+');

try{

$query = "INSERT INTO vent (idUsuario,nome,valor,tipo) VALUES ('{$id}',?,?,?);" ;

$stm = $this->conexao->prepare($query);

```
fgets($fh);
while(($row = fgetcsv($fh,0,",")) !== false){
                $stm->execute($row);
}

fclose($fh);
} catch(PDOException $e){
        die($e->getMessage());
        }
}
```

A seguir, o código que implementa a verificação se os dados recebidos estão corretos de acordo com as boas práticas de segurança:

```
        function compararOKS($id,$bancoID){ // Faz a primeira comparação dos que estão corretos
                $query = "INSERT INTO saida(idUsuario,nome,valor,resultado)
                SELECT :id,vent.nome,vent.valor,'OK' FROM vent,vref WHERE vref.idBanco = :banco AND vent.nome = vref.nome AND (vent.valor = vref.valor OR vent.valor<>'-');";

                $stm = $this->conexao->prepare($query);
                $stm->bindValue(":id", $id);
                $stm->bindValue(":banco", $bancoID);
                return $stm->execute();
        }
```

A seguir, o código que implementa a comparação dos dados que não estão de acordo com as boas práticas de segurança:

```
function compararNOKS($id,$bancoID){ // Faz a comparação dos que não estão corretos
                $query = "INSERT INTO saida(idUsuario,nome,valor,resultado)
               SELECT :id,vent.nome,vent.valor,'NOK' FROM vent,vref WHERE vref.idBanco = :banco AND vent.nome = vref.nome AND vent.valor <> vref.valor;";

                $stm = $this->conexao->prepare($query);
                $stm->bindValue(":id", $id);
                $stm->bindValue(":banco", $bancoID);
                return $stm->execute();
        }
```

Abaixo, consta a implementação da listagem de resultados obtidos pelas comparações a cima em tela:

```
        function listaResultado($idUsuario){ //Listagem para exibição na tela
                $query = "SELECT * FROM saida WHERE saida.idUsuario = :id;";

                $stm = $this->conexao->prepare($query);
                $stm->bindValue(":id",$idUsuario);
                $stm->execute();
                $variaveis = array();
                while ($variavel = $stm->fetchall(PDO::FETCH_CLASS,'Variavel')) {
                        array_push($variaveis, $variavel);
                }
                return $variaveis;
        }
```

A implementação para gerar um arquivo com os resultados obtidos pela comparação, conforme:

```
        function listaResultadoParaDownload($idUsuario){ //Listagem para gerar o download e criação do arquivo CSV!
                $query = "SELECT * FROM saida WHERE saida.idUsuario = :id;";
```

```
        $stm = $this->conexao->prepare($query);
        $stm->bindValue(":id",$idUsuario);
        $stm->execute();
        $variaveis = array();
        while ($variavel = $stm->fetch()) {
                array_push($variaveis, $variavel);
        }
        return $variaveis;
}
```

O restante do código fonte está disponível nos apêndices deste documento.

5 CONSIDERAÇÕES FINAIS

O objetivo inicial do projeto é auxiliar o profissional que trabalha ou utiliza um banco de dados a realizar de maneira clara as configurações de segurança de uma instalação de banco de dados Oracle, de acordo com as boas práticas de segurança publicadas pela Oracle e seus colaboradores, como também no estudo realizado por Komel (2011).

Com o SVPBD, esse processo se realiza de forma automática, não necessitando o usuário realizar uma busca das melhores práticas, e compará-las manualmente. Mas, sim com uma ferramenta que realizasse todo o processo automático gerando uma resposta através de um relatório maneira clara, rápida e objetiva, de acordo com o que foi proposto.

Desta forma, usuário final do sistema é auxiliado na tomada de decisões a partir da utilização das comparações de parâmetros realizadas pelo sistema, facilitando a visualização sistêmica do banco de dados que ele utiliza.

Desde o princípio, o foco do desenvolvimento do SVPBD era o desenvolvimento de uma ferramenta que possibilita a utilização para bancos de dados Oracle. Entretanto, com o desenvolvimento da ferramenta, possibilitou-se o cadastro de diversos tipos e distribuição de banco de dados com suas boas práticas, sendo sugerido o estudo de outras ferramentas a fim de levantar as boas práticas de outras distribuições de banco de dados, possibilitando o alcance ainda maior dessa ferramenta.

Assim, esse estudo e desenvolvimento de sistema cumprem o seu objetivo inicial, e fornece aos administradores de banco de dados, a possibilidade automatizada de encontrar um auxílio nas configurações e tomadas de decisão relacionadas à segurança de um banco de dados qualquer.

.

REFERÊNCIAS

APINFO. **Pesquisa salarial**. 2014. Disponível em: http://www.apinfo2.com/apinfo/informacao/p12sal-br.cfm. Acesso em 08 set. 2015.

____. **Pesquisa de procura de mercado.** 2014. Disponível em: http://www.apinfo2.com/apinfo/informacao/p14lingua.cfm. Acesso em: 29 de outubro de 2015.

ASTAH. **Versões do software Astah.** 2016. Disponível em: http://astah.net/editions. Acesso em: 03 de fevereiro de 2016.

____. **Sobre o Astah.** 2016. Disponível em: http://astah.net/about-us. Acesso em: 03 de fevereiro de 2016.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML:** Guia do Usuário. Rio de Janeiro. Editora Campus, 2006.

BRYLA, B.; LONEY, K. **OCA Oracle Database 11g Manual do DBA**. Porto Alegre. Editora Bookman, 2009.

DATE, C. J. **Introdução a Sistemas de Banco de Dados.** Tradução da 8. ed. Americana. Editora Campus, 2004.

FEITOSA, M. P. **Fundamentos de banco de dados:** uma abordagem prático-didática. São Paulo: Edição do Autor, 2013.

HEUSER, C. A. **Projeto de Banco de Dados**. 4ª edição editora Sagra, 1998.

JQUERY. **JQUERY:** Licença de uso. 2016. Disponível em: https://jquery.org/license/. Acessado em: 13 de maio de 2016.

KOMEL, W. D. **Software de análise de segurança de banco de dados oracle.** Ourinhos: FATEC, 2011.

MARCELO, A.; **Apache:** configurando o servidor WEB para Linux**.** 3ª edição. Rio de Janeiro. Editora Brasport, 2005.

NIEDERAUER, J.; **Desenvolvendo Websites com PHP.** 2ª edição. São Paulo. Editora Navatec, 2011.

____. J.; **PHP para quem conhece PHP.** 4ª edição. São Paulo. Editora Navatec, 2013.

ORACLE. 2015. Disponível em: http://www.oracle.com/br/index.html. Acesso em 19 de setembro de 2015.

____. 2016. Disponível em: http://www.oracle.com/br/products/mysql/index.html. Acesso em 21 de maio de 2016.

PRESSMAN, R. S.; **Engenharia de Software:** Uma abordagem Profissional. 6ª edição. Rio de Janeiro: Editora McGraw-Hill, 2006.

____.; **Engenharia de Software:** Uma abordagem Profissional. 7ª edição. Rio de Janeiro: Editora McGraw-Hill, 2011.

SAVOIA, H. S.; **XHTML e CSS + PHP e MySQL.** 1ª edição. Riberão Preto. Editora IELD tec, 2013.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 5 edição. São Paulo: Editora Campus, 1999.

SILVA, M. S.; **Criando sites com HTML:** Sites de alta qualidade com HTML e CSS. 1ª edição. São Paulo. Editora Navatec, 2008.

SILVA, M. S.; **JQuery:** A Biblioteca do Programador JavaScript. 3ª edição. São Paulo. Editora Navatec, 2013.

SUEHRING, S.; **MySQL Bible.** 1ª edição. USA. Wiley Publishing, Inc., 2002.

TAVARES, M. **Administração de sistemas de gerenciamento de banco de dados**: um estudo no Oracle 10g. 2006. Disponível em: www.cin.ufpe.br/~tg/2006-1/mact.ppt mercado oracle. Acesso em: 10 de setembro de 2015.

WATSON, J. **OCA Oracle Database 11g Administração I**. Porto Alegre. Editora Bookman, 2010.

APÊNDICE A – Código fonte

A classe principal para o desenvolvimento do projeto e comparação de parâmetros é apresentada da seguinte forma na integra:

```
<?php


class variavelDAO{

	private $conexao;


	function __construct($conexao){

		$this->conexao = $conexao;

	}


	function addVar(Variavel $variavel){

		$query = "INSERT INTO vref (idBanco, nome, valor) VALUES (:idBanco,:nome,:valor);";


		$stm = $this->conexao->prepare($query);

		$stm->bindValue(":idBanco",$variavel->getIdBanco());

		$stm->bindValue(":nome",$variavel->getNome());

		$stm->bindValue(":valor",$variavel->getValor());

		return $stm->execute();

	}
```

```
function listVar($id) : array {
        $query = "SELECT * FROM vref WHERE vref.idBanco = :id ;";

        $stm = $this->conexao->prepare($query);
        $stm->bindValue(":id",$id);
        $stm->execute();
        $variaveis = array();
        while ($variavel = $stm->fetchall(PDO::FETCH_CLASS,'Variavel')) {
                array_push($variaveis, $variavel);
        }
        return $variaveis;
}

function buscaVar($id){
        $query = "SELECT * FROM vref WHERE vref.idBanco = :id;";

        $stm = $this->conexao->prepare($query);
        $stm->bindValue(":id", $id);
        $stm->execute();
        $variaveis = array();
        while ($variavel = $stm->fetchall(PDO::FETCH_CLASS,'Variavel')) {
```

```
                array_push($variaveis, $variavel);
            }

            return $variaveis;
        }

        function temporaria($id,$file){ //Aqui jogamos passamos os dados do arquivo CSV para a tabela de entrada no Banco!

            $fh = fopen($file['tmp_name'],'r+');

            try{
                $query = "INSERT INTO vent (idUsuario,nome,valor,tipo) VALUES ('{$id}',?,?,?);" ;
                $stm = $this->conexao->prepare($query);


                fgets($fh);

                while(($row = fgetcsv($fh,0,",")) !== false){
                    $stm->execute($row);
                }
```

```
                fclose($fh);
            } catch(PDOException $e){
                die($e->getMessage());
            }
        }

        function compararOKS($id,$bancoID){ // Faz a primeira comparação dos que estão corretos
            $query = "INSERT INTO saida(idUsuario,nome,valor,resultado)
            SELECT :id,vent.nome,vent.valor,'OK' FROM vent,vref WHERE vref.idBanco = :banco AND vent.nome = vref.nome AND (vent.valor = vref.valor OR vent.valor<>'-');";

            $stm = $this->conexao->prepare($query);
            $stm->bindValue(":id", $id);
            $stm->bindValue(":banco", $bancoID);
            return $stm->execute();
        }

        function compararNOKS($id,$bancoID){ // Faz a comparação dos que não estão corretos
            $query = "INSERT INTO saida(idUsuario,nome,valor,resultado)
            SELECT :id,vent.nome,vent.valor,'NOK' FROM vent,vref WHERE vref.idBanco = :banco AND vent.nome = vref.nome AND vent.valor <> vref.valor;";
```

```
        $stm = $this->conexao->prepare($query);
        $stm->bindValue(":id", $id);
        $stm->bindValue(":banco", $bancoID);
        return $stm->execute();
    }

    function listaResultado($idUsuario){ //Listagem para exibição na tela
        $query = "SELECT * FROM saida WHERE saida.idUsuario = :id;";

        $stm = $this->conexao->prepare($query);
        $stm->bindValue(":id",$idUsuario);
        $stm->execute();
        $variaveis = array();
        while ($variavel = $stm->fetchall(PDO::FETCH_CLASS,'Variavel')) {
            array_push($variaveis, $variavel);
        }
        return $variaveis;
    }

    function listaResultadoParaDownload($idUsuario){ //Listagem para gerar o download e criação do arquivo CSV!
        $query = "SELECT * FROM saida WHERE saida.idUsuario = :id;";
```

```
            $stm = $this->conexao->prepare($query);
            $stm->bindValue(":id",$idUsuario);
            $stm->execute();
            $variaveis = array();
            while ($variavel = $stm->fetch()) {
                array_push($variaveis, $variavel);
            }
            return $variaveis;
        }

        function deletaSaidaUser($id){// após ser feito o download são deletadas as inserções!
            $query = "DELETE FROM saida WHERE saida.idUsuario = :id;";

            $stm = $this->conexao->prepare($query);
            $stm->bindValue(":id",$id);
            $stm->execute();
        }
}
```

Classe para leitura de arquivo CSV:

```
<?php
require_once("../viewSistem/cabecalho.php");
require_once("../Conexao/conexao.php");
require_once("../DAO/bancoDAO.php");
require_once("../DAO/variavelDAO.php");
require_once("../DAO/UsuarioDAO.php");
require_once("logica-user.php");

verificaUsuario();
$con = new Conexao();

$usuarioDAO = new UsuarioDAO($con->getConexao());
$user = $usuarioDAO->busca(usuarioLogado());

$bancoID = $_POST['idBanco'];

$con = new Conexao();
$variavelDAO = new variavelDAO($con->getConexao());
$bancoDAO = new bancoDAO($con->getConexao());
$variavelDAO->temporaria($user[0]->getId(),$_FILES['file']);
```

```
$variavelDAO->compararOKS($user[0]->getId(),$bancoID);

$variavelDAO->compararNOKS($user[0]->getId(),$bancoID);


header("Location: ../viewSistem/resultado.php");

?>

<?php require_once("../viewSistem/rodape.php"); ?>
```

Classe que adiciona um novo banco de dados:

```
<?php

require_once("../view/cabecalho.php");

require_once("../Conexao/conexao.php");

require_once("../DAO/bancoDAO.php");

$con = new Conexao();

$nome = $_POST['nome'];


$banco = new Banco();

$banco->setNome($nome);


$bancoDAO = new bancoDAO($con->getConexao());


if($bancoDAO->addBanco($banco)){ ?>
```

```
        <p class="text-success">O Banco <?= $banco->getNome()?> foi cadastrado!</p>
<?php }else{
        $msg = \PDO::errorInfo($con->getConexao());
?>
        <p class="text-danger">O Banco <?= $banco->getNome()?> não foi cadastrado : <?= $msg ?></p>
<?php
}
?>
<?php
die();
require_once("../view/rodape.php");?>
```

Classe de download de arquivo:

```
<?php
require_once("../Conexao/conexao.php");
require_once("../DAO/variavelDAO.php");
require_once("../DAO/UsuarioDAO.php");
require_once("../Classes/Usuario.php");
require_once("../Classes/Variavel.php");
require_once("../actions/logica-user.php");
```

```
verificaUsuario();
$con = new Conexao();
$variavelDAO = new variavelDAO($con->getConexao());
$usuarioDAO = new UsuarioDAO($con->getConexao());
$user = $usuarioDAO->busca(usuarioLogado());
$variaveis = $variavelDAO->listaResultadoParaDownload($user[0]->getId());


function outputCsv($fileName, $assocDataArray)
{
    ob_clean();
    header('Pragma: public');
    header('Expires: 0');
    header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
    header('Cache-Control: private', false);
    header('Content-Type: text/csv');
    header('Content-Disposition: attachment;filename=' . $fileName);
    if(isset($assocDataArray['0'])){
        $fp = fopen('php://output', 'w');
        fputcsv($fp, array_keys($assocDataArray[0]));
        foreach($assocDataArray AS $values){
            fputcsv($fp, $values);
        }
```

```
        fclose($fp);
    }
    ob_flush();
}

$variavelDAO->deletaSaidaUser($user[0]->getId());
outputCsv($user[0]->getEmail(). date('d-m-Y').".csv", $variaveis);
die();
?>
```

Sendo assim apresentadas as classes mais importantes do projeto.