UNIVERSIDADE ESTADUAL DO MATO GROSSO DO SUL – UEMS

JOSÉ CARUZO TEODORO NETO

SISTEMA DE INFORMAÇÃO DINÂMICO PARA GERENCIAMENTO DE EQUAÇÕES - SIDGE

DOURADOS-MS

2018

SISTEMA DE INFORMAÇÃO DINÂMICO PARA GERENCIAMENTO DE EQUAÇÕES - SIDGE

JOŚE CARUZO TEODORO NETO

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso devidamente corrigida e defendida por José Caruzo Teodoro Neto e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 19 de Outubro de 2018

PROFª DRA. GLÁUCIA GABRIEL DE SASS

JOSÉ CARUZO TEODORO NETO

SISTEMA DE INFORMAÇÃO DINÂMICO PARA GERENCIAMENTO DE EQUAÇÕES – SIDGE

Outubro de 2018

Banca Examinadora:

Profa. Dra. GLAUCIA GABRIEL SASS (Orientadora)
Área de Computação – UEMS

Prof. Msc. André Chastel Lima
Área de Computação – UEMS

Prof Dr. Cleber Valgas Gomes Mira
Área de Computação – UEMS

Eu dedico este trabalho aos meus pais, que me apoiaram em toda essa minha jornada da graduação, sempre acreditaram que era possível e não me deixaram desanimar.

Além disso, gostaria de dedicar ao Prof. Dr. Odival Faccenda que mesmo não podendo ser meu Co-orientador, por estar afastado, foi a pessoa que mais me incentivou em minhas ideias, que mais me inspirou e deu dicas valiosas, sendo crucial para a conclusão deste trabalho.

AGRADECIMENTOS

Agradeço a todos que estiveram comigo neste momento da minha vida, que estiveram ao meu lado e souberam como foi gratificante para mim, concluir este trabalho. Agradeço aos meus pais, meus professores (em especial o Prof. Dr. Faccenda, a Profª Dra. Gláucia e o Prof. Msc. Chastel), meus amigos e todos que indiretamente me ajudaram de alguma maneira nesta conquista.

RESUMO

Os Sistemas de Informação (SI) com estrutura de dados temporais, principalmente no caso dos que são compostas por equações matemáticas, tem a reformulação e as alterações trabalhosas e demoradas, ter um sistema que gerencie todos os cálculos destes SIs temporais facilitaria e diminuiria muito a demanda de manutenção dos mesmos. Pensando nisso foi proposto criar o Sistema de informação Dinâmico para gerenciamento de Equações (SIDGE), um SI web planejado para armazenar equações, juntamente com seus resultados, de forma que estas interajam entre si formando uma rede hierárquica, almejando potencializar suas alterações. O projeto foi desenvolvido utilizando uma base de dados estruturada, e implementado com o conceito de Metodologia Ágil de Desenvolvimento: XP, dentro do ciclo de vida de prototipação. Por fim, ao se aplicar o SIDGE a outro SI, o mesmo realizaria o gerenciamento das equações, evitando realizar alterações "manuais" de equação a equação, pois, uma vez que as equações são cadastradas ou alteradas na base de dados do SIDGE elas consequentemente se atualizam no SI que o aplica.

Palavras chave: Sistemas de Informação, SIDGE, Gerenciamento de Equações.

ABSTRACT

The Information Systems (IS), with temporal data structure, mainly in cases that are composed by mathematical equations, the reformulation and amendments are cumbersome and time-consuming, have a system that manages all mathematical calculations hint are applied to a temporal IS would facilitate its maintenance and would decrease the demand for modification on the project's source code. With that in mind, the propose to create a Dynamic Information System for Equation Management (Sistema de Informação Dinâmico para Gerenciamento de Equações - SIDGE), a web IS planned in a way to store equations and solutions on the same database, so hint theses equations can interact with each other forming a hierarchical network, aiming at potentialize it's changes. This project was developed using a structured database, implemented with the concept of Development Agile Methodology: XP, in the prototyping life cycle. Finally, applying SIDGE to another SI, it would perform the management of the equations, avoinding changes directly in the source code whenever there were modifications to be applied. Avoinding manual changes in each equation, since these equations are registered or modified on the SIDGE database, they would be automatically applied in the SI that uses it.

Keywords: Information Systems, SIDGE, Equations Management.

SUMÁRIO

LISTA DE SIGLAS

AJAX	*Asynchronous JavaScript and XML*
CSS	*Cascading Style Sheets*
CWI	*Centrum Wiskunde & Informatica*
DER	*Diagrama de Entidade Relacionamento*
DOM	Document Object Model
ECMA	*European Computer Manufacturer's Association*
GNU	*General Public License*
GPENSI	Grupo de Pesquisa em Necessidades de Saúde do Idoso
HTML	*Hypertext Markup Language*
IVI	Índice de Vulnerabilidade do Idoso
KDD	*Knowledge Discovey in Databases*
KIS	*Keep It Simple*
MDS	Modelagem Dinâmica de Sistemas
MER	Modelo de Entidade Relacionamento
MVC	*Model*, *View* e *Controller*
POO	Programação Orientada a Objetos
RN	*Regras de Negócio*
SAD	Sistema de Apoio à Decisão
SGBD	Sistema Gerenciador de Banco de Dados
SI	Sistema de Informação
SIAMI	Sistema de Informação para Avaliação e Monitoramento do Idoso
SIDGE	Sistema de Informaçao Dinâmico para Gerenciamento de Equações
SO	*Sistema Operacional*
TI	Tecnologia da Informação
WAN	*Wide Área Network*
WHATWG	*Web Hypertext Aplication Tecnology Working Group*
W3C	*Wolrd Wide Web Consortium*
XP	*Extreming Programing*

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

O conceito de Sistema de Informação (SI) existe desde antes da popularização dos computadores, sendo baseado em arquivamento de documentos de forma física e a partir disso era feito o controle dessas informações, de acordo com a necessidade.

Normalmente um "arquivador" era responsável por esta organização, podendo registrar, catalogar, alterar e consultar os documentos, quando necessário. Mesmo sendo um método simples, exige um grande esforço para que seja realizado, podendo demandar muito tempo (TURBAN; MCLEAN; WETHERBE, 2004).

Com a evolução da tecnologia o conceito de Tecnologia da Informação (TI) tomou forma. Podendo ser caracterizada como um conjunto de recursos dedicados ao armazenamento, processamento, comunicação da informação e a organização dos recursos em um sistema computacional, capaz de executar um conjunto de tarefas (LAUDON E LAUDON, 2004).

A era da informação se iniciou no século XX, e desde então, SI's passaram a ser uma ferramenta fundamental para qualquer organização, pois com a utilização das tecnologias disponíveis, tornou-se mais fácil o gerenciamento das informações, e assim a TI ganhou o *status* de necessidade.

SI dentro da TI é baseado em *softwares* que podem ser definidos como uma sequência de instruções a serem interpretadas por um computador, com o intuito de executar tarefas em específico (O'BRIEN, 2004).

Com o tempo, os SI's passaram a realizar as mais diversas atividades, desde gerenciamento de documentos, até automatização de processos, que antes eram realizados manualmente por pessoas, e isso fez com que cada vez mais se tornassem complexos (PRATES; OSPINA, 2004).

SI's muito grandes e complexos estão em constante mudança, resultando na necessidade de manutenção e atualização. Se for um SI com embasamento matemático, a alteração dos dados de uma das equações gera impacto em diversas partes do mesmo, podendo até mesmo impactar nele como um todo.

As alterações dos requisitos para a formação de um SI é o que demandam a sua manutenção, para atender a necessidade do usuário, pois, de nada vale um SI que não acompanhe essa evolução.

Pensando no caso de SI's com estrutura de dados temporais, ou seja, dados que se mantém constantemente em mudança, uma manutenção constante de *software* é necessária, o

que acaba gerando um grande esforço para mantê-lo o mais atualizado possível.

Pensando no caso de um SI com estrutura temporal, é interessante entender onde ocorre a maior demanda de atualizações e adotar um meio de que elas ocorram de maneira que não há a constante necessidade de alterar o código fonte do SI. Diminuindo a quantidade de manutenções que o SI sofreria.

No caso de um SI com embasamento matemático/estatístico, a maior demanda do *software* ocorreria nos cálculos que o compõe, uma vez que o usuário tem melhor conhecimento matemático do que o profissional de TI que o desenvolve, torna-se mais viável que a construção dos cálculos esteja sob o controle de alteração do usuário. Quando isso é sanado, diminui a necessidade de alteração do código-fonte do SI, além de deixar o usuário mais independente, agilizando as alterações do SI.

1.1. Objetivos

1.1.1. Objetivo geral

- Realizar o gerenciamento de equações matemáticas implantadas em Sistemas de Informação, de forma padronizada.

1.1.2. Objetivos específicos

- Estudar métodos de resoluções de equações matemáticas.
- Analisar e desenvolver uma estrutura em banco de dados que gerencie dados dinâmicos.
- Desenvolver um SI que possa gerenciar as equações matemáticas em um banco de dados.
- Aplicar o SI a um *software* onde há a necessidade de reformulação constante de equações.
- Testar o SI.

2. REFERENCIAL TEÓRICO

2.1. Sistemas de Informação

Antes mesmo de se definir SI, é importante ressaltar a diferença entre, “dado”, “informação” e “conhecimento”, conceitos que muitas vezes são confundidos, pois estão interligados.

Drucker (2002) diz que informação é um conjunto de dados dotados de relevância e propósito. E então, Davenport (2003) enfatiza essa definição e resumidamente afirma que dado é tudo que facilmente possa se observar em um determinado estado do mundo, e conhecimento é uma informação que foi interpretada e contextualizada. Como pode-se observar na tabela abaixo.

Tabela 1 – Definição de dado, informação e conhecimento.

	Definição	Características
Dado	Simples observação sobre o estado do mundo.	• Facilmente estruturado. • Facilmente obtido por máquinas. • Frequentemente quantificado. • Facilmente transferível.
Informação	Dado dotado de relevância e propósito.	• Requer Unidade de análise. • Exige consenso em relação ao significado. • Exige necessariamente a mediação humana.
Conhecimento	Informação valiosa da mente humana, incluindo reflexão, síntese e contexto.	• Informação valiosa da mente humana. • Inclui reflexão, síntese e contexto. • De difícil estruturação. • De difícil captura em máquinas. • Frequentemente tático. De difícil transferência.

Fonte: Davenport (2003)

Há diversas maneiras de se definir a palavra “sistema”, mas de forma geral sistema é um conjunto de várias partes organizadas e sincronizadas e que age com o intuito de atender uma ou mais finalidades (SILVA, 2008).

Dentre os diversos tipos de sistemas, existe o chamado SI, que é definido por O´Brien (2004) como "um conjunto organizado de pessoas, *hardware*, *software*, rede de comunicação e recursos de dados que coleta, transforma e dissemina informações em uma organização". Afirmando que o sistema recebe dados como entrada, os processa e no fim retorna um ou mais produtos como saída.

De forma mais detalhada, sistemas de informação (computacionais) são compostos por *hardware*, *software*, banco de dados, telecomunicações, pessoas e procedimentos.

Hardware: Equipamento físico do computador, todos os componentes eletrônicos que o compõe.

Software: Parte de processamento do sistema, que executa as instruções do computador.

Banco de dados: Coleção de dados armazenados em uma estrutura de memória não volátil.

Telecomunicações: Responsável pela comunicação com outros dispositivos.

Pessoas: Os elementos mais significativos do sistema, pois são os que o criam, executam, gerenciam, mantém o sistema e o nutri de dados.

Procedimentos: É o que determina como devem ser operados os sistemas computacionais, o que ditam as regras e o que será executado, em qual ordem e as estratégias a serem usadas (O'BRIEN, 2004).

De forma geral todos reconhecem que SI's são essenciais para os administradores e gestores, pois a maioria das organizações precisa desses sistemas para sobreviver e prosperar, uma vez que a informação se tornou um dos bens mais valiosos nos dias atuais (LAUDON E LAUDON, 2004).

Para O´Brien (2004), com o avanço tecnológico cada vez mais as organizações precisam de uma melhor gestão para suas atividades, almejando informações ágeis, completas e precisas. E para isso um SI pode ser a solução ideal para estas necessidades.

Laudon e Laudon (2004) afirmam que existem diversos tipos sistemas de informação, pois, há inúmeros níveis, especificidades e interesses em uma organização.

Dentre os diversos tipos de SI, temos os de apoio a decisão e os sistemas dinâmicos, que serão os focos deste estudo.

2.1.1. Sistemas de informação de apoio à decisão

Em todas as áreas profissionais, o processo de tomada de decisão está presente.

E em todas as situações é importante tomar a melhor decisão, embora nem sempre isso aconteça e o intuito dos Sistemas de apoio de decisão é ajudar na tomada de decisão do usuário.

Sistemas de Apoio à Decisão (SAD) são sistemas que utilizam de tecnologia de informação com a finalidade de converter dados ou informações pouco estruturadas, de forma sistemática, em conhecimento ou informações mais estruturadas, e são destinados a apoiar a tomada de decisões dos usuários (BATISTA, 2004).

Os SI's relacionados aos SAD podem ser definidos também como "mecanismos cuja função é coletar, guardar e distribuir informações para suportar as funções gerenciais e operacionais das organizações".

Há sistemas específicos, que às vezes são agrupados em categorias de acordo com sua orientação principal, observa-se, que muitos programas analíticos associados aos SAD são, na verdade, programas independentes (FREITAS ET. AL, 2003).

Com estes conceitos do SAD podemos defini-lo resumidamente como um sistema que a partir da entrada de dados ou informações não estruturadas irá transformá-los em um resultado de informações mais estruturadas que ajudam ao usuário tomar decisões no seu dia a dia (FERREIRA, 2010).

Para tomar as decisões os sistemas SAD utilizam de recursos matemáticos e lógicos como análise dos dados, cálculos estatísticos e outros tipos de recursos.

Um conceito que caminha juntamente com o SAD é o do processo *Knowledge Discovey in Databases* (KDD) que de forma traduzida é conhecido como Descoberta de Conhecimento em Bases de Dados (DCBD) (FAYYAD, 2001).

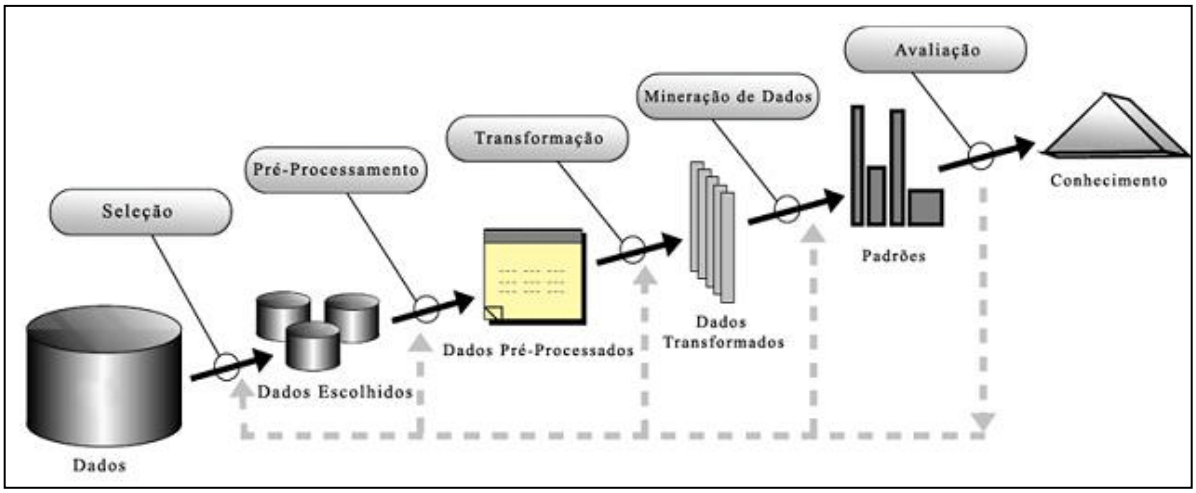
As bases de dados são compreendidas como fontes de informação eletrônicas, pesquisáveis de modo interativo ou conversacional através de um computador. (POBLACIÓN; WITTER; SILVA, 2006).

Segundo Fayyad (2001), o KDD é um processo com a tratativa da descoberta de conhecimento em bases de dados, é muito importante na extração de conhecimento a partir de uma base de dados, se tornando potencialmente útil, por isso, pode-se dizer que definição está diretamente ligada ao conceito de SAD.

O processo de KDD nada mais é do que uma série de passos compreendidos por: seleção, pré-processamento, transformação, mineração de dados (*data mining*) e avaliação. Gerando um ciclo em que o dado percorre até se tornar conhecimento ou

informação, como pode ser visto na Figura 1.

Figura 01 – Processo de transformação de dado em conhecimento.



Fonte: https://www.researchgate.net/figure/Figura-1-Processo-de-KDD-O-processo-de-KDD-consiste-em-uma-sequencia-de-etapas-que-devem_fig1_308995146, Acessado em 30 de setembro de 2018.

Para que possa se utilizar KDD é essencial que haja um Sistema Gerenciador de Banco de Dados – SGBD, sendo o responsável por armazenar os dados que serão analisados pelas aplicações que utilizam o conceito de KDD. SGBD's são sistemas computacionais que servem para controlar uma base de dados, criando e fazendo a manutenção dos dados, fazendo com que o usuário não precise realizar o controle dos dados físicos em ficheiros. Agilizando a busca, relacionando, facilitando o acesso e estruturando os dados, agilizando a extração de informações (LAUDON E LAUDON, 2010).

2.1.2. Sistemas de Informação Dinâmicos

Entende-se por SI Dinâmico qualquer processo que evolua com o tempo (ALVES, 2001).

SI's Dinâmicos são sistemas fora do equilíbrio, caracterizados por estados que mudam com o tempo. São usados para modelar e fazer previsões de sistemas físicos, biológicos, financeiros, etc.

Estas decisões são convertidas em ações, interferindo diretamente no comportamento do sistema. Desde que as informações são geradas, é possível avaliar o impacto da decisão passada no sistema em questão (FERNANDES, 2003).

Para que seja possível a criação de um SI dinâmico a modelagem do sistema

tem que ser elaborada para suportar tais mudanças.

A Modelagem Dinâmica de Sistemas (MDS) dentro de SI's dinâmicos é uma modelagem para entender como os objetos de um sistema interagem na escala de tempo. Objetos e pessoas em um sistema interagem através de laços, sendo que a mudança das informações afeta diversas áreas da aplicação. E com o tempo, as modificações vão interagindo e alterando a estrutura original (SDEP, 2005).

Sendo assim, pode-se afirmar que MDS é uma metodologia que busca mapear sistemas, com o objetivo de examinar as inter-relações e suas influências com um contexto sistêmico, ou seja, que afeta o *software* como um todo (ANDRADE, 2006).

Que também foi definido por Ogata (2003), como uma técnica na qual SI's dinâmicos e complexos podem ser entendidos e analisados, através de interações.

Além de novas políticas e estruturas serem desenhadas para melhorar o comportamento do sistema. Para aplicar o modelo dinâmico são feitas análises partindo dos ciclos de "*feedback*" das informações. (SANTOS, 2004).

2.2. Ciclo de vida voltado para Sistema de Informação Dinâmico

Existem vários tipos de ciclos de vida de um SI, eles são definidos a partir dos processos do *software*. Um processo de *software* é um conjunto de atividades, ações e tarefas a serem realizadas, para criá-lo é necessário levantar todos os conjuntos de informações descritos anteriormente e dividi-los em fases que se relacionam para gerar o produto final do *software* (PRESSMAN, 2011).

De acordo com Sommerville (2011) as fases do ciclo de vida de um SI podem ser divididas em: especificação, desenvolvimento, validação e evolução do *software*.

Especificação: Levantamento das informações do *software* com o usuário, visando entender a natureza e o domínio do problema, para definir as funcionalidades e comportamentos esperados para atender a necessidade do usuário.

Desenvolvimento: É a fase onde o software vai ser esboçado, e em seguida programado, de acordo com os requisitos que foram levantados.

Validação: Fase de testes, para comprovar se os requisitos do usuário são atendidos.

Evolução: O *software* é modificado para se adaptar às mudanças de requisitos do usuário, de acordo com a necessidade. E também por evolução para acompanhar às mudanças do mercado.

Cada uma das atividades, ações e tarefas do processo de um *software* é feita se enquadrando dentro de um modelo. O papel do modelo é determinar o relacionamento das atividades, ações e tarefas com o processo e a interação entre elas (PRESSMAN, 2011).

A escolha de um modelo de processo é muito dependente das características do projeto. Sendo importante conhecer diferentes tipos de modelos de ciclo de vida e em que situações devem ser aplicados. Os principais modelos podem ser divididos em três categorias: modelos sequenciais, modelos incrementais e modelos evolutivos (PFLEEGER, 2004).

Tendo em vista os objetivos do projeto, possível observar que dentre os três modelos citados acima, o que melhor se aplica ao SIDGE é o modelo evolutivo.

Os modelos evolutivos ou evolucionários frisam nos sistemas complexos, que evoluem com o tempo. Pois os requisitos são difíceis de ser definidos com exatidão, ou mudam com frequência.

Diferente dos outros modelos, que tem como base a entrega de versões operacionais desde o primeiro ciclo, os modelos evolutivos focam em produzir protótipos, conforme o desenvolvimento avança os protótipos vão dar lugar a versões operacionais e vai solidificando o sistema, até que esteja completamente concluído. Os modelos evolutivos são comumente usados quando o problema não consegue ser bem especificado no início do desenvolvimento ou não ficou bem definido de acordo com os requisitos apresentados (PFLEEGER, 2004).

Muitas vezes o cliente define muitos objetivos para o sistema, porém não sabe definir os requisitos ou não consegue os definir claramente, tornando difícil de aplicar os outros modelos de processo (PFLEEGER, 2004).

O ciclo de vida do sistema, no modelo evolutivo, precisa se adequar com mudanças contínuas e com incertezas. Muitas vezes os primeiros ciclos produzem versões não usuais, ou apenas modelos e conforme o desenvolvimento avança os requisitos vão ficando mais claros e estáveis e as versões começam a se tornar operacionais. (PFLEEGER, 2004).

Dois dos principais ciclos de vida do modelo evolutivo são: espiral e prototipação. Após diversos estudos, optou-se por usar o modelo de ciclo de vida de prototipação. Pois, o conceito de versionamento deste ciclo de vida se encaixa muito bem no SIDGE, pensando que em primeiro momento não é se tem um conhecimento aprofundado dos requisitos, e o ciclo de vida preza pela facilidade e agilidade de desenvolvimento a cada

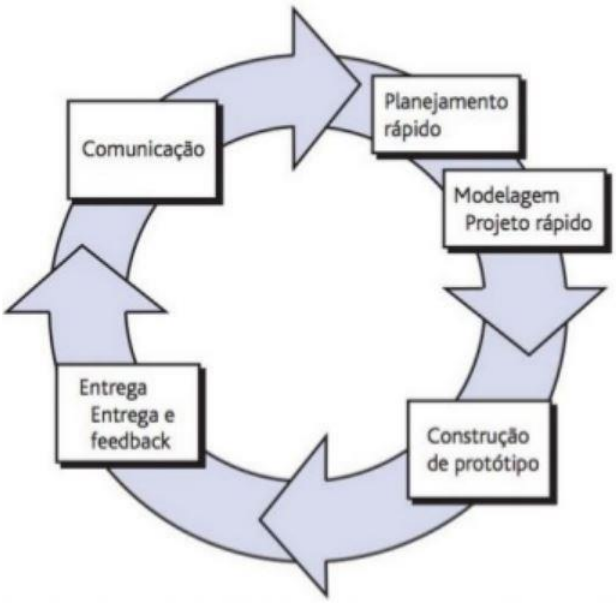
versão.

A ideia da prototipação é levantar objetivos gerais de forma rápida, gerar um protótipo, e no decorrer do processo os requisitos serão definidos com mais clareza (PRESSMAN, 2011).

Após o protótipo ser entregue para o cliente e/ou usuários, os mesmos dão um retorno (*feedback*) do que foi feito, para que seja realizado o aprimoramento do sistema, isso sucessivamente até o sistema ser finalizado por completo (PRESSMAN, 2011).

A Figura 2 demonstra o ciclo de vida de todo protótipo do modelo de prototipação, todas as atividades realizadas para gerar cada protótipo do sistema. O ciclo se repete diversas vezes até que o sistema esteja concluído.

Figura 02 – Representação do ciclo de vida do modelo de prototipação.



Fonte: PRESSMAN (2011).

2.3. Delimitação do problema para o estudo de caso

Para este projeto, foi utilizado como base do estudo um SI do Grupo de Pesquisa em Necessidades de Saúde do Idoso (GPENSI), pois por ser um sistema na área de saúde onde as equações estão em constante mudança, a aplicabilidade de um SI dinâmico ajudaria no controle do versionamento, podendo assim, futuramente ser aplicado em outros diversos SI's. Esse o SI em questão possui indicadores de vulnerabilidade de idosos, que são calculados a partir de relatórios respondidos pelos idosos.

Primeiro, para estudar os indicadores do projeto SIAMI, é necessário saber mais sobre o que é o SIAMI e sobre as questões que envolvem a vulnerabilidade dos idosos.

Com a incidência crescente de idosos vivendo com enfermidades crônicas, problemas de saúde e decréscimo na sua capacidade funcional, há um aumento significativo das despesas para suas famílias e para os serviços de saúde, no entanto, quanto mais integrados psicológica e socialmente estiverem os idosos, menos despesas e melhor será sua qualidade de vida. (ALVARENGA, 2008).

Com o aumento da população de idosos, há uma maior preocupação por parte da sociedade em geral com os problemas que afetam essa faixa etária, principalmente os de saúde, sociais, financeiros, atenção, cognitivos e abandono. Desta forma e considerando a segunda lei de diretriz da Política Nacional de Informação e Informática em Saúde, que pretende estabelecer o registro eletrônico de saúde, para recuperar as informações de saúde do indivíduo em seus diversos contatos com o sistema de saúde, para melhorar a qualidade dos processos de trabalho em saúde, (SASS et al, 2012) propuseram o SIAMI. O referido SI se propõe a fornecer resposta em tempo real aos profissionais de saúde e ao gestor quando os dados indicarem que um idoso está em situação de vulnerabilidade.

Foram implementadas fórmulas estatísticas que geram os indicadores do SIAMI de acordo com um índice, que é chamado de IVI. O IVI é uma base teórica para a indicação da vulnerabilidade do idoso, a partir dele são analisados os dados coletados pelo SIAMI.

Para verificar a vulnerabilidade de um idoso, são averiguados dois tipos de dados, onde um é independente e o outro dependente. São eles os indicadores e fatores, respectivamente.

Os indicadores são gerados a partir de embasamentos estatísticos, que apontam a situação do idoso, referente a uma análise dos dados coletadas sobre o mesmo. Os indicadores dão como resposta um status do idoso, podendo ser: bom, regular ou ruim. Mostrados para o usuário de forma intuitiva partindo de cores básicas, sendo: verde, referente a um status bom; amarelo, para o regular; e vermelho, para o ruim.

Já os fatores são os resultados da soma de indicadores para uma determinada área de avaliação e segue o mesmo intuito de resposta que os indicadores, um bom exemplo para citar é o Fator 1:

TF1 = ***0.771*I8_DMR+ 0.698*I13_ICF + 0.635 * I7_AS + 0.632*I9_RN+0.454 * I12_EDG***

O fator 1 é referente a Condição de saúde do idoso, este é composto por cinco indicadores (I8_DMR, I13_ICF, I7_AS, I9_RN, I12_EDG), onde cada um possui um peso (0.771, 0.698, 0.635, 0.632, 0.454). Depois de determinados os pesos e multiplicados os indicadores, os mesmos serão somados, chegando ao resultado final do fator.

Atualmente o SIAMI está com a sua última versão na *web*, e sistemas de *software* para internet (sistemas *web*) são modificados com dinamismo, pois conseguem alcançar os clientes, tendo então um baixo custo, comparado a sistemas que possuem sua instalação em máquinas locais sem acesso a WAN (WAN é *Wide Área Network*, significa uma rede que cobre uma área física maior, como o campus de uma universidade, uma cidade, um estado ou mesmo um país.) (TANEMBAUM, 2003).

As aplicações *web* possuem a facilidade de serem aplicadas a qualquer tipo de máquina (multiplataforma, pois depende apenas do navegador e seus serviços), pensando também no fato que a partir do momento que as alterações são realizadas já entram em uso, em tempo real, fazendo com que este tipo de *software* tenha fortes requisitos de desempenho.

No quesito atualização/evolução do SIAMI, toda alteração que precisar ser feita em seu modelo de dados, atualmente, ocasiona na alteração de diversas partes do código-fonte do sistema, para que ele possa suportar os novos requisitos de entrada.

Uma vez que a alteração se torna mais complexa, por conta da interligação dos dados do sistema, fica muito difícil manter este tipo de aplicação atualizada, os SI's "estáticos" não tendem a prever a alteração temporal (MRACK, 2003).

É possível afirmar que grande parte dos sistemas atuais passam pela mesma situação, possuem a complexidade de manutenção e gerenciamento grande demais para que seja feita manualmente. Isso promove que a construção do SI já seja feita de forma que ele possa prover as funções de autoconfiguração, autorrecuperação e autoproteção (DODONOV, 2009).

- **Autoconfiguração**: O sistema deve ser capaz de se reconfigurar de acordo com a necessidade e alterações do ambiente.
- **Autorrecuperação**: O sistema deve ser capaz de detectar e se recuperar de falhas, minimizando os danos e retornando as atividades normais.
- **Autoproteção**: O sistema deve ser capaz de detectar, identificar e proteger-se de ataques de terceiros, mantendo a integridade do sistema.

2.4. Trabalhos relacionados

2.4.1. Geogebra

GeoGebra é um aplicativo dinâmico, combinando os conceitos matemáticos geometria e álgebra em um só lugar. Sua distribuição é livre, nos termos da *General Public License* (GNU), além do fato de ser um *software* é escrito em linguagem de programação Java, o permitindo estar disponível em diversas plataformas.

O programa disponibiliza a construções geométricas a partir de pontos, retas, segmentos de reta, polígonos etc., como também é possível inserir funções, além de poder alterar todos esses objetos dinamicamente pela interface, após a construção estar finalizada. Outro ponto chave da aplicação é inserir equações e coordenadas para que sejam resolvidas de forma rápida e fácil.

Portanto, o GeoGebra é capaz de lidar com variáveis para números, pontos, vetores, derivar e integrar funções, e ainda oferecer comandos para se encontrar raízes e pontos extremos de uma função.

Geogebra foi criado por Markus Hohenwarter com objetivo acadêmico. Iniciado em 2001, na Universität Salzburg, e continuado na Florida Atlantic University (GEOGEBRA, 2018).

Figura 03 – Logo do Geogebra.

Fonte: https://help.geogebra.org/topic/geogebra-logo

2.4.2. Mathway

Mathway é um aplicativo *web* com a intenção de resolver vários tipos de problemas de matemática: matemática básica, pré-álgebra, álgebra, trigonometria, pré-cálculo, cálculo, estatística, matemática finita, álgebra linear, química e plotação de gráfico.

O aplicativo é capaz de fazer desde as operações básicas, como operação com

frações, porcentagens, área, volume, até funções polinomiais, derivadas, integrais e, ele ainda cria a solução em gráfico.

Tendo a interface bem limpa e simples de usar, basta inserir o problema matemático no editor, e ao submeter o problema matemático o aplicativo pergunta qual é o tipo de resposta que o usuário quer, dependendo do que foi submetido.

Ex.: Caso x+y = 7 seja inserido no editor para resolução. As opções que aparecem para se escolher são: gráfico, resolver para x, resolver para y, encontre a inclinação e o intercepto em y, escreva em forma de y = mx+b, encontre a solução para três pares ordenados etc.

A medida que se digita no editor, ele tenta identificar qual é o problema que está sendo proposto pelo usuário, de maneira parecida com a recomendação de pesquisa do Google. Caso ele não for capaz de identificar o tipo de problema corretamente, é possível selecionar a partir da lista predefinida pelo aplicativo.

Mathway tem uma versão paga, que mostra o passo a passo da resolução dos problemas que foram submetidos no editor (MATHWAY, 2018).

Figura 04 – *Logo do Mathw*ay

Fonte: https://play.google.com/store/apps/dev?id=6431727159568786626

2.4.3. Microsoft Mathematics

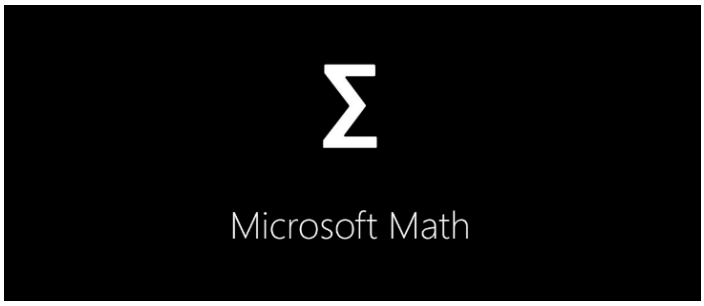
O Microsoft Mathematics (antigo Microsoft Math) é um *software* gratuito que fornece um conjunto de ferramentas matemáticas com o objetivo educacional. Resolvendo equações passo-a-passo, juntamente com a obtenção de uma melhor compreensão dos conceitos fundamentais em pré-álgebra, álgebra, trigonometria, física, química e cálculo.

Microsoft Mathematics foi desenvolvido e é mantido pela Microsoft, sendo voltado principalmente para os alunos como uma ferramenta de aprendizado. É disponível como um *plugin* para ser utilizado dentro do Microsoft Word ou no One Note.

O Microsoft Mathematics inclui uma calculadora gráfica completa, projetada para ser utilizada como uma calculadora portátil. Possui ferramentas matemáticas

adicionais ajudam a converter de um sistema de unidades para outro e resolver sistemas de equações (MICROSOFT, 2010).

Figura 05 – Logo do *Microsoft Math.*



Fonte: https://www.windowsteam.com.br/microsoft-math-conheca-esse-incrivel-app-que-te-ajudara-a-aprender-matematica/

2.4.4. Wolfram Alpha

Wolfram Alpha é um mecanismo de conhecimento computacional desenvolvido pela Wolfram Research. Foi anunciado em 2009 pelo físico britânico Stephen Wolfram, e está em funcionamento desde maio deste mesmo ano. Tem sua versão pro (versão paga), onde estende o recursos da aplicação.

Resumidamente, é um *software web* que responde às perguntas parecendo um site de buscas, e trás de retorno, mediante o processamento da resposta que será extraída de uma base de dados estruturados, todas as informações encontradas.

Ele abrange diversos conteúdos, como: matemática, ciência e tecnologia, sociedade e cultura e dia-a-dia. Mas pensando no tópico matemática, o que pode ser encontrado de retorno nas buscas do Wolfram Alpha é plotação de gráfico, resolução passo a passo, resultado solução dentro do conjunto dos números reais e também dos complexos, etc.

Wolfram Alpha se baseia em outro *software* da empresa Wolfram Research, o Wolfram Mathematics, uma plataforma computacional ou *toolkit* que abrange álgebra computacional, computação simbólica e numérica, visualização e recursos de estatística (WOLFRAM, 2009).

Figura 06 – Logo do WolframAlpha.

Fonte: https://commons.wikimedia.org/wiki/File:Wolfram_Alpha_logo.svg

Mesmo os trabalhos relacionados ao projeto, citados anteriormente, conterem aplicações que resolvam equações eles não são viáveis para a utilização por outros sistemas, pois o objetivo destes SIs é de unicamente resolver os cálculos que são propostos, em tempo real, fornecendo o resultado e muitas vezes o passo a passo ou usuário. Porém esses resultados não são armazenados e muito menos podem ser aplicados a outros SIs, sendo objetivo do SIDGE sanar esta necessidade.

2.5. Tecnologias empregadas no projeto

2.5.1. Programação orientada a objetos

É um paradigma para programação de *software* que tem embasamento na utilização de componentes chamados de classes, que individualmente colaboram para construir sistemas mais complexos. Um paradigma é um conjunto de regras que estabelecem um modelo, um padrão, e descrevem como resolver problemas a partir disto, ajuda a organizar e coordenar o jeito que vemos o mundo. (BEZERRA, 2007)

As classes que são responsáveis por compor o sistema possuem instâncias, que se chamam objetos, que se comunicam entre si e com outros objetos de outras classes, gerando uma colaboração no sistema. A colaboração entre os objetos é feita através do envio de mensagens, ou seja, envio e recebimento de comandos.

De acordo com Bezerra (2007), para entender melhor do que se trata a Programação Orientada a Objetos (POO), é necessário compreender os 4 pilares que uma linguagem desse paradigma deve ter:

- **Abstração:** tem o intuito de representar um objeto do mundo real em um objeto do sistema, onde objeto do mundo real tem que ser abstraído para o sistema, imaginando onde será aplicado e o que realizará. Todos os objetos devem possuir identidade, propriedades e/ou métodos, onde identidade é um nome que irá representá-lo, propriedades são suas características e métodos são suas ações dentro do sistema.
- **Encapsulamento:** trata-se de um dos elementos que adicionam segurança à aplicação em uma POO, onde “esconde” as propriedades. O encapsulamento normalmente é baseado em propriedades privadas, que são interligadas com métodos especiais, *get*() e *set*(), que servem para acessar e alterar as informações do objeto. Para melhor entendimento, pode-se fazer a analogia do clique de um botão “ligar” de algum eletrodoméstico, o usuário em

questão sabe que disparou um comando para realizar a ação de ligar, porém não tem conhecimento do que acontece internamente, averiguando que o método que liga o eletrodoméstico está encapsulado.

- **Herança:** Uma das grandes vantagens atribuídas ao uso de POO é o reuso de código, isso ocorre a partir da herança, característica que otimiza a produção da aplicação. Na aplicação da herança, um objeto abaixo em uma hierarquia, herda as características de todos acima dele. Existindo herança direta e indireta.

  Ex.: Animais → Mamíferos → Homo Sapiens

  "Homo Sapiens" é uma classe que herda diretamente de "Mamíferos" e indiretamente de "Animais".

- **Polimorfismo:** consiste na modificação de funcionalidades de um método para realizar uma mesma atividade. Por exemplo: O cálculo da área de um objeto. A área de um losango não é calculada da mesma maneira que a do quadrado, porém não deixa de ser um cálculo de área, o mesmo método "cálculo de área" pode possuir diferentes maneiras de ser executado para que possa abranger todos os possíveis casos.

2.5.2. Padrão MVC

O padrão de *Model*, *View* e *Controller* (MVC) é um padrão de arquitetura de *software*, que tem o objetivo de dividir a aplicação em camadas, de forma bem definida, visando separar a interface do usuário das informações do sistema, fazendo com que cada camada possa ser alterada sem interferir de forma expressiva em outra, proporcionando flexibilidade e maior chance de reuso de classes.

- *Model* - consiste nos dados da aplicação, regras de negócios, lógicas e funções.
- View - pode ser qualquer saída de representação dos dados, como uma tabela ou um diagrama (contato do cliente com o sistema).
- Controller - faz o intermédio da entrada de dados (*View*), convertendo-a em comandos para o *Model*. O *Controller* é responsável por manipular as outras camadas e tomar decisões a partir das ações realizadas pelo usuário. (ZEMEL, 2009).

2.5.3. Aplicação *web*

Uma das grandes vantagens da aplicação *web* é o fato de que é de fácil acessibilidade, e foi o principal motivo para ser escolhida para ser utilizada neste projeto. A fácil acessibilidade se dá pelo fato de que as aplicações web pelo lado do cliente são representadas pelos *browsers (Ex.: Google Chrome, Firefox, Opera, Safari, Internet Explorer/Microsoft Edge),* que são *softwares* de fácil acesso em diversos dispositivos.

A estrutura de uma aplicação web é diferente das demais estruturas de *software*, girando em torno de um conceito de arquitetura cliente/servidor, onde o cliente fará uma requisição para o servidor e o servidor irá processar a requisição e retornar para o cliente.

Na arquitetura cliente/servidor o processo das requisições é feito por dois lados da aplicação, o *client-side* e o *server-side*, também conhecidos como *font-end* e *back-end.*

Front-end: Parte do sistema de interação com o usuário, responsável pela interface e experiência do usuário, ou seja, conjunto de componentes que formam a "comunicação" entre o cliente e o servidor, que não se conectam diretamente, onde haverá o recebimento dos dados de entrada e responsável por expor os dados de retorno do servidor (dados de saída).

Um *software* pode ser controlado através de um usuário usando um dispositivo e esse dispositivo se comunica diretamente com o *software*. A interface entre o dispositivo e o usuário é a tela de comandos apresentada pelo *software*, ou seja, a interface gráfica do *software* (PREECE; ROGERS; SHARP, 2005).

Back-end: Parte do sistema que é responsável por aplicar as Regras de Negócio (RN) do sistema, executar as requisições feitas pelo lado do cliente. É o lado que processa os dados de entrada, podendo: salvar em banco de dados, analisar, calcular, gerar informações a partir dos dados, etc. Resumidamente é a parte de processamento/tomada de decisão da aplicação.

Entende-se RN, no contexto de SI, como um requisito de *software* como critérios e restrições que precisam ser atendidos para que o SI alcance um bom funcionamento, dentro dos padrões esperados, com o objetivo de estruturar, controlar ou influenciar o comportamento do SI (WIEGERS, 2003).

2.5.4. Linguagens de programação/marcação

Uma linguagem de programação é uma linguagem utilizada para gerar instruções para um computador.

Um conjunto de regras sintáticas e semânticas usadas de forma a definir um programa de computador. Permite definir com precisão em quais dados um computador vai atuar, como estes dados serão armazenados e/ou transmitidos e quais ações devem ser tomadas (MELO; SILVA, 2003).

Linguagem de marcação é uma linguagem que utiliza marcação para definir formatos, formas de exibição ou padrões para documentos, diferente das linguagens de programação não realiza ações de controle a serem seguidas, servindo basicamente para definir o conteúdo de uma tela.

Como este projeto será desenvolvido um SI para a *Web* terá linguagens de programação e marcação para atender o lado do cliente e do servidor.

Linguagens *front-end*: As linguagens utilizadas do lado do cliente são de dois tipos: linguagens de marcação, para que seja construída a interface do usuário, e linguagens de programação, para fazer as pré-validações, animações, etc melhorando a interatividade do usuário com a aplicação.

As linguagens utilizadas foram: HTML5, CSS3 e *Javascript.*

- **HTML5**: É a última versão do *Hyper Text Markup Language* (HTML) traduzido como Linguagem de Marcação de Hipertexto, versão esta que foi criada de um consórcio entre a *Wolrd Wide Web Consortium* (W3C) e a *Web Hypertext Aplication Tecnology Working Group* (WHATWG), de forma resumida, HTML é uma linguagem baseada em *tags* que estrutura a página *web.*

  Tag do inglês significa etiqueta, na programação *web* serve para "etiquetar" textos, que irá definir os componentes na página, como se comportará na página e quais características ele vai assumir.

- **CSS3:** É a última versão do *Cascading Style Sheets* (CSS), traduzido como Folha de Estilo em Cascatas, foi criado pela W3C e é uma linguagem que define a aparência dos componentes da página *web*.

  O CSS trabalha juntamente com o HTML e outras linguagens de marcação, onde usa as *tags* para definir os estilos dos componentes (Ex.: cor, fonte, tamanho do texto, posição da tela, etc).

- ***JavaScript:*** É uma linguagem de programação originalmente *client-side*, que com o tempo vai tomando forma também no *server-side*, a partir de *frameworks* como por exemplo o Node.js.

  JavaScript dentro da aplicação *web* é responsável por manipular e controlar os componentes da página, esta linguagem foi criada por Brendan Eich na Netscape e é mantida pela *European Computer Manufacturer's Association* (ECMA).

  Basicamente *Javascript* é uma linguagem baseada em *scripts*, "roteiros" que são seguidos para executar ações no SI. É uma linguagem Orientada a Objetos baseada em protótipos, que vem para manipular os componentes da página *Web*, realiza diversas atividades do lado do cliente, evitando passar pelo servidor, diminuindo a quantidade de requisições ao servidor e aumentando a experiência do usuário com o SI.

Linguagens *back-end*: A linguagem do lado servidor é responsável por receber as requisições do *client-side* e gerar uma resposta para a requisição.

Existem diversas linguagens que podem ser utilizadas para o desenvolvimento *back-end*, *Python*, PHP, C#, *Ruby* e *Java* são algumas delas. Cada uma possui suas vantagens e desvantagens em relação ao uso no desenvolvimento web.

Depois de uma análise dentre as linguagens a serem utilizadas, a linguagem de programação escolhida para a construção do sistema foi *Python*. Por possuir um padrão de Programação Orientada a Objetos e também por ser uma linguagem fácil de aplicar o padrão de modelagem MVC. A versão do *Python* aplicada ao projeto é a versão 3.

- ***Python* 3:** É uma linguagem de altíssimo nível, sendo ela orientada a objeto, com tipagem dinâmica, interpretada e iterativa. Uma linguagem que é de fácil compreensão e mais produtiva, por possuir um código limpo. Sendo assim, clara e concisa, a tornando de mais simples compreensão e legibilidade (BORGES, 2010).

  O *Python* foi criado em 1989 por Guido Van Rossum em um instituto de pesquisa, o Instituto *Centrum Wiskunde & Informatica* (CWI), nos Países Baixos. Vem como substituto da ABC, visando aumentar a produtividade dos desenvolvedores.

  A versão 3 do *Python* é a versão mais estável da linguagem, lançado em

2008, tendo em vista que a versão do *Python*2 não terá mais suporte a partir de 2020.

2.5.5. Banco de dados e SGBD’s

Banco de dados é uma estrutura utilizada para armazenamento de informações e para realizar a manipulação dessas informações é necessário de algo que o gerencie como um todo. Para tal, existem os chamados Sistemas Gerenciadores de Banco de Dados (SGBD), sendo alguns dos principais: Oracle, SQLServer, MySQL, PostgreSQL, MongoDB, SQLite, Cassandra etc (DATE, 2003).

PostgreSql foi o SGBD escolhido para ser aplicado a este projeto, manipulando os dados que serão adicionados a base de dados. Pois, possui uma arquitetura capaz de suportar sistemas grandes e complexos de maneira mais simplificada e robusta, ou seja, que possui uma boa manipulação de erros.

O SGBD PostgreSQL permite a criação de bancos de dados estruturados e não estruturados, onde o modelo estruturado é baseado em relações entre as entidades do banco e o não estruturado baseia-se no conceito *NoSQL*, não relacional. (POSTGRESQL, 2018).

As entidades são abstrações de algo do mundo real para o mundo virtual e os relacionamentos são como essas entidades interagem entre si.

Mesmo o projeto envolvendo a dinamicidade de equações, permitindo construir diferentes tipos de equações com quantidade indeterminada de variáveis, optou-se por um banco de dados estruturado.

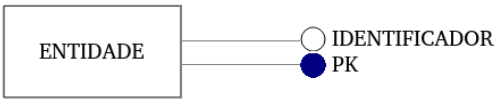
Como dito anteriormente o banco de dados estruturado é baseado em relacionamentos entre entidades, isto forma um Modelo de Entidade Relacionamento (MER). Para representar o MER é construído um diagrama expondo graficamente estas relações.

Este diagrama é chamado de Diagrama de Entidade Relacionamento (DER), e a partir dele que se definiu a estrutura do banco de dados.

Entidades**:** no DER as entidades são compostas por nome e atributos. Onde o nome é a sua identificação e os atributos são suas características. Representadas por retângulos, com o nome no interior e os atributos representados por retas com um círculo na extremidade, como pode ser observado na Figura 7.

O atributo que representa a chave primária da entidade tem o círculo da cor azul. As chaves primárias não admitem valores repetidos e podem ser usadas como um índice para as demais entidades.

Figura 07 – Entidade em um DER.



Fonte: Próprio autor.

Relacionamentos**:** no DER as relações são conexões entre as entidades, representando a interação que ocorre entre elas. Representadas por um losango contendo uma descrição da interação entre as classes no seu interior. Existem três tipo de relações entre as entidades: um para um , um para muitos e muitos para muitos.

Neste trabalho só serão utilizados relacionamentos muitos para muitos, a Figura 8 representa como está disposta uma relação de muitos para muitos no DER.

Figura 8 – Relação de muitos para muitos em um DER.

Fonte: Próprio autor

2.5.6. *Frameworks* de desenvolvimento

Um *framework* (ou biblioteca), em desenvolvimento de *software*, é uma abstração que une códigos comuns entre vários projetos de software, provendo uma funcionalidade genérica. Um *framework* pode atingir uma funcionalidade específica, por configuração, durante a programação de uma aplicação. Os *frameworks* que foram englobados no projeto foram: *Django*, *Bootstrap* e *jQuery*.

- **Django:** é um *framework* que foi construído na linguagem *Python*, que visa a praticidade de desenvolvimento. Foi criado por causa do aumento da complexidade dos sistemas web atuais, junto com questões de segurança e escalabilidade. A adequação precisava ser realizada para acompanhar a "evolução" das aplicações, o *Django* veio para realizar isso. (SANTANA

NETO, 2010).

No projeto, *Django* será usado para realizar as conexões com o banco de dados, a iteração dos modelos e dos formulários que são construídos em linguagem de marcação HTML, juntamente com CSS.

- ***Bootstrap*:** é uma estrutura web de *front-end*, é um *software* livre e de código aberto para projetar aplicações web como um todo. Possui modelos de design embasados em HTML e CSS, com os seguintes tópicos: tipografia, formas, botões, navegação e outros componentes de interface. Assim como também extensões de *JavaScript* opcionais .

  Diferentemente de muitos *frameworks* da *web*, ele se preocupa apenas com o desenvolvimento do front-end da aplicação. O *Bootstrap* é modular e visa com menos código implementar os vários componentes do conjunto de ferramentas. Suas folhas de estilo (Normalmente arquivos CSS) geralmente são compiladas em um pacote e incluídas nas páginas da Web, mas os componentes individuais podem ser incluídos ou removidos. O *Bootstrap* fornece uma série de variáveis de configuração que controlam o estilo dos componentes.

- ***jQuery*:** é implementado em *JavaScript* com o objetivo de interagir com o HTML, foi desenvolvida para simplificar os *scripts* interpretados no navegador do lado do cliente. Sendo lançada em 2006 na cidade de Nova York, por John Resig.

  jQuery é uma dentre as mais populares bibliotecas *JavaScript*. Além de possuir código aberto onde qualquer um pode colaborar com o projeto. A sintaxe do *jQuery* foi desenvolvida para tornar mais simples a navegação do documento HTML e a manipulação do *Document Object Model* (DOM), gerenciar animações e eventos, assim como desenvolver funções com chamada de *Asynchronous JavaScript and XML* (AJAX).

Tais facilidades permitem aos desenvolvedores criarem camadas de abstração, de modo simplificado em aplicações web. Além de disponibilizar diversos *plug-ins* para complementar a programação da biblioteca, desde que a biblioteca *jQuery* em si tenha sido instanciada na aplicação.

2.5.7. Metodologia Ágil

A metodologia de pesquisa deve ser entendida como o conjunto detalhado e sequencial de métodos e técnicas científicas a serem executados ao longo da mesma, de tal modo que se consiga atingir os objetivos inicialmente propostos e, ao mesmo tempo, atender aos critérios de menor custo, maior rapidez, maior eficácia e mais confiabilidade de informação (LAKATOS ; MARCONI, 2007).

Decidiu-se construir o projeto dentro do modelo de vida evolutivo, seguindo o conceito de modelo de prototipação.

O ciclo dos protótipos segue um modelo de: obtenção de requisitos, projeto rápido, construção do protótipo, avaliação do protótipo e refinamento do protótipo. Estas atividades serão realizadas diversas vezes até chegar no produto final. Para este modelo de ciclo de vida, um modelo de desenvolvimento que se adequa a curtos períodos de desenvolvimento e de rápidas entregas, é muito bem aplicado. Pensando nesse pré-requisito optou-se pelo modelo de desenvolvimento ágil *Extreming Programing* – XP.

A metodologia XP é baseada em programação simples e com organização de requisitos para que o desenvolvimento seja realizado por um grupo pequeno de pessoas.

A metodologia XP é utilizada em projetos cujas especificações são passíveis a alterações, em que as iterações entre as atividades do projeto costumam ser de curtos períodos, produzindo um protótipo a cada uma delas.

Particularmente, será seguido o ciclo de vida XP apresentado por Pressman (2011). O processo do ciclo de vida XP abrange as seguintes atividades:

- **Planejamento:** é uma atividade de levantamento de requisitos, para levantar os principais fatores de funcionalidades, objetivando entender o *software* a ser desenvolvido.
- **Projeto:** o projeto XP segue o princípio *Keep It Simple* – KIS, ou seja, preserva a simplicidade, além de subdividir o projeto em várias partes, formando vários protótipos a serem desenvolvidos até chegar à versão final.

  O desenvolvimento deve ser o mais simples possível visando que maneira que satisfaça apenas o necessário do protótipo atual e nada além disso, pois a medida que o projeto avança ele vai se complementando.

Independente de qual fase o projeto esteja, o SI está sempre aberto a mudanças.

- **Codificação:** a codificação na metodologia XP, normalmente é feita em pares e sempre está ligada a testes constantes e entregas rápidas das versões do SI, com o intuito de fazer a versão do menor tamanho possível com os requisitos com o maior valor ao projeto.

  A codificação se preocupa unicamente com os requisitos da versão atual, mas mesmo assim é feita de forma padronizada para aceitar as mudanças.

- **Testes:** O processo de realização de testes concentra-se principalmente em cada versão do *software* feita, os chamados testes unitários, garantindo que todas as instruções tenham sido testadas, como também para garantir que a entrada definida produza os resultados desejados no levantamento de requisitos e por fim feito teste de implantação para averiguar a aplicabilidade e funcionalidades do projeto como um todo.

3. METODOLOGIA

Este trabalho é do tipo pesquisa tecnológica. Dividido em duas etapas, sendo elas: estudo teórico e desenvolvimento.

O estudo teórico foi realizado a partir de livros, artigos científicos, teses etc. Estudo este feito em cima dos conceitos listados no capítulo 2. Conceitos estes muito importantes, pois, guiam o desenvolvimento do projeto, seguindo os padrões definidos.

Na etapa de desenvolvimento foi onde se implementou o projeto, aplicando os conceitos teóricos estudados.

Para o desenvolvimento do projeto foi necessário utilizar um computador para instalar e configurar todos os recursos necessários para criar o projeto (*Django*, *Python* e *PostgreSQL*). Este computador se comporta como um servidor local para hospedar o SI do projeto. O computador em questão é um notebook, com o Sistema Operacional (SO) Ubuntu 18.04.

Depois do ambiente de desenvolvimento estar pronto para uso, foi criada uma base de dados no PostgreSQL e implementado um arquivo .sql contendo a estrutura da base de dados do projeto.

A parte de implementação do aplicativo *web* que faria o gerenciamento da base de dados seguiu o ciclo de vida de prototipação e metodologia ágil de desenvolvimento. Foram implementados cinco módulos para este projeto: página de login, página principal, módulos de gerenciamento de usuário, módulos de gerenciamento de equação e consultas de usuário, equação e atividade. Cada módulo, na sequência anterior, teve sua fase de planejamento, projeto, codificação e teste. No primeiro protótipo foi construído o módulo da página de login, no segundo protótipo, o módulo da página principal foi inserido no *software/*protótipo, e assim por diante.

Cada módulo tem uma fase de teste, onde são realizados testes unitários do protótipo. Quando os novos módulos são incluídos, são realizados os testes de integração e depois que o SI estiver completo ele passa por testes de sistema e de implantação.

- **Planejamento**: serviu para definir os requisitos que deveriam ser atendidos, suas funcionalidades e quais testes que deveriam ser realizados na fase dos testes unitários.
- **Projeto**: foi elaborado como seriam as telas, como seriam feitas as validações do protótipo e quais tecnologias seriam necessárias para atender os requisitos.
- **Codificação**: fase onde foi implementado tudo que foi levantado e projetado na fase de planejamento e projeto. Construídas as telas, criadas as funcionalidades

client-side e *server-side* e feita a integração com a base de dados para realizar o gerenciamentos das tabelas da mesma.

- **Testes**: Todos os testes levantados durante a fase de planejamento, foram devidamente realizados. Sendo que, todas as alterações necessárias para cumprir com os requisitos e garantir a correta funcionalidade do sistema serão devidamente aplicadas, caso encontradas, passando novamente pela fase de codificação e retestadas.

Por fim, após todos os módulos devidamente finalizados, o projeto foi aplicado ao SIAMI e realizou-se todos os testes de sistema para averiguar a veracidade do SI. Após feitas diversas comparações dos resultados das equações armazenadas na base de dados do SI, com os resultados gerados pelo SIAMI, pode-se afirmar que o projeto supriu todas as necessidades que foram levantadas.

4. RESULTADOS

O SI do projeto foi desenvolvido de acordo com o que foi proposto nos objetivos específicos e na metodologia. A seguir serão descritos os resultados obtidos, baseado nos cinco tópicos dos objetivos específicos, que foram levantados.

4.1. Estudar métodos de resolução de equações matemáticas

Durante o estudo dos métodos de resolução foi elaborado um método para armazenar as equações em uma base de dados, com o objetivo de armazenar a composição da equação e o resultado.

Pois, na situação em que somente a composição da equação é armazenada, a equação precisaria ser calculada para poder mostrar o resultado, esse cálculo se torna desnecessário enquanto a composição da equação não se alterar, uma vez que as mudanças no resultado ocorrem apenas quando a composição da equação é alterada.

Outro ponto levantado é a questão das equações poderem ou não compor outras equações e equações que podem assumir um valor atribuído.

Basicamente, a estrutura da base de dados foi construída visando uma estrutura de pirâmide ou hierarquia, param realizar esta dependência entre equações.

Ex.: x = 1

y = x+2

z = y^2 + x

O x é a equação independente; o y é dependente de x; o z é dependente de y e x.

O x neste caso seria como a base da pirâmide, pois caso ele não exista, nem y ou z poderiam ser calculados por falta de valores. E z é o topo da pirâmide, por conta de não tem nenhuma equação que dependa do seu resultado.

Assim pode-se concluir que z pode ser facilmente removido e alterado, enquanto x não. As equações criadas nesta estrutura podem assumir dois tipos, as atribuídas e as calculadas.

As equações atribuídas têm como preceito assumir um resultado definido diretamente e as equações calculadas geram os resultados a partir da composição que o usuário fornece.

Pelo fato das equações poderem compor outras equações, a partir do momento que uma é alterada, todas que são compostas por ela são recalculadas e assim sucessivamente, até que todas as equações que tenham alguma relação com a equação alterada inicialmente

tenha sido recalculadas. Ou seja, faz com que uma alteração seja automaticamente aplicada a todas as equações necessárias.

Além disso, por conta deste método de interação entre as equações, não é possível excluir uma equação que esteja compondo outra equação, pois isto quebraria o preceito da estrutura.

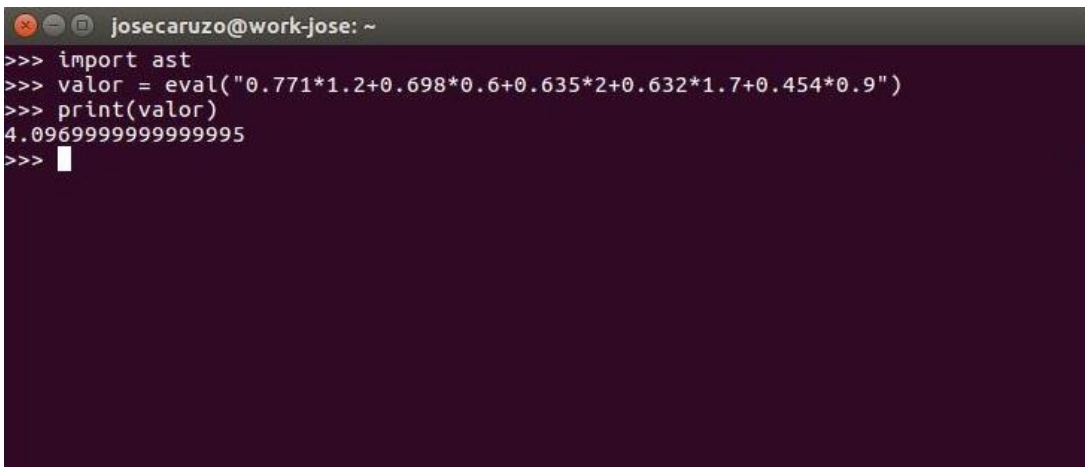
Tendo em vista a linguagem de programação *Python*, foram estudadas diversas formas de resolver equações a partir destes conceitos levantados. Onde se encontrou uma biblioteca de apoio chamada ***ast*** que permite a utilização da função ***eval(),*** função esta que pode ser utilizada para realizar cálculos.

A função ***eval()*** funciona a partir do recebimento de uma variável do tipo *string* (conjunto de caracteres) como parâmetro, a interpretando como código nativo da linguagem.

Este recurso foi visto como útil para resolver equações, onde o único empecilho foi o fato que as mesmas estejam na sintaxe da linguagem *Python*. Por fim, a função executa o código das equações passado e retorna os valores do tipo inteiro ou real.

Ao declarar a biblioteca ***ast***, basta fazer a chamada da função ***eval()*** com o conjunto de caracteres dentro dos parênteses, tudo ali será interpretado pela linguagem e retornado. Foi construído um exemplo, onde uma variável com o nome de “valor” recebe o resultado da função e em seguida a variável é impressa para verificar o resultado obtido, como segue na Figura 9.

Figura 09 – Código teste da função *eval*().



Fonte: Próprio autor.

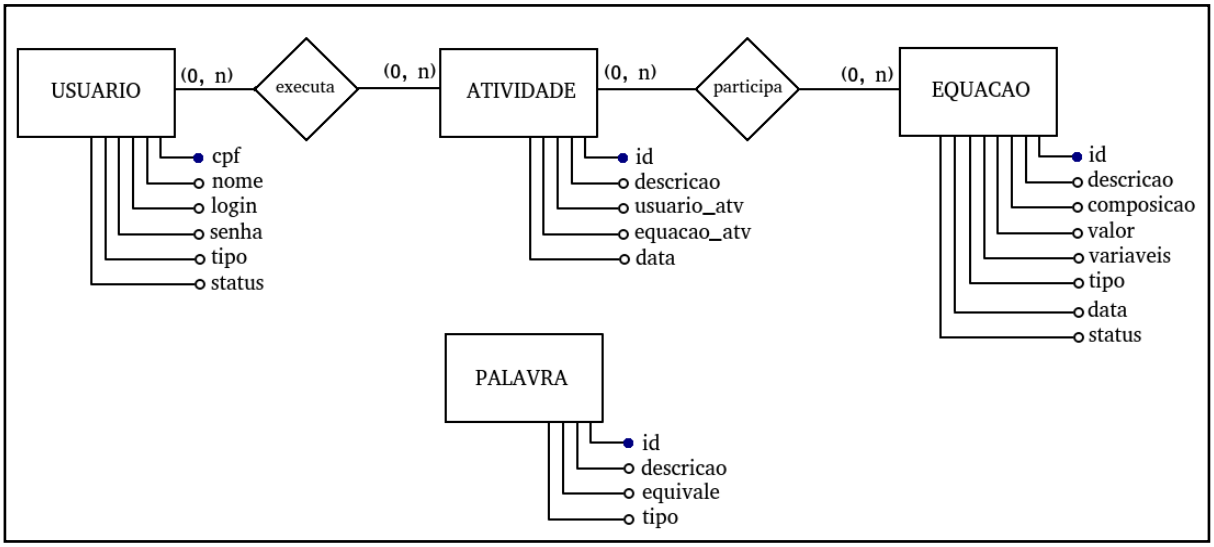
Além do meio que foi definido para ser utilizado no projeto, é interessante comentar sobre as linguagens LISP e PROLOG, que são facilmente aplicadas a sistemas de manipulação simbólica, que poderiam ser usadas para gerar uma lógica para resolução de equações.

4.2. Analisar e desenvolver uma estrutura em banco de dados que gerencie dados dinâmicos

Depois dos estudos sobre as diagramações de sistemas estruturados, foi projetado o DER (Figura 10) que serviu para criar a base de dados do SI, composto por quatro entidades: Usuário, Equação, Atividade e Palavra.

Usuário, Equação e Atividade são entidades que se relacionam e a entidade Palavra é uma entidade de auxílio que não se relaciona com as demais, servindo para armazenar informações que ajudam a formar as equações.

Figura 10 – DER do projeto.



Fonte: Próprio autor.

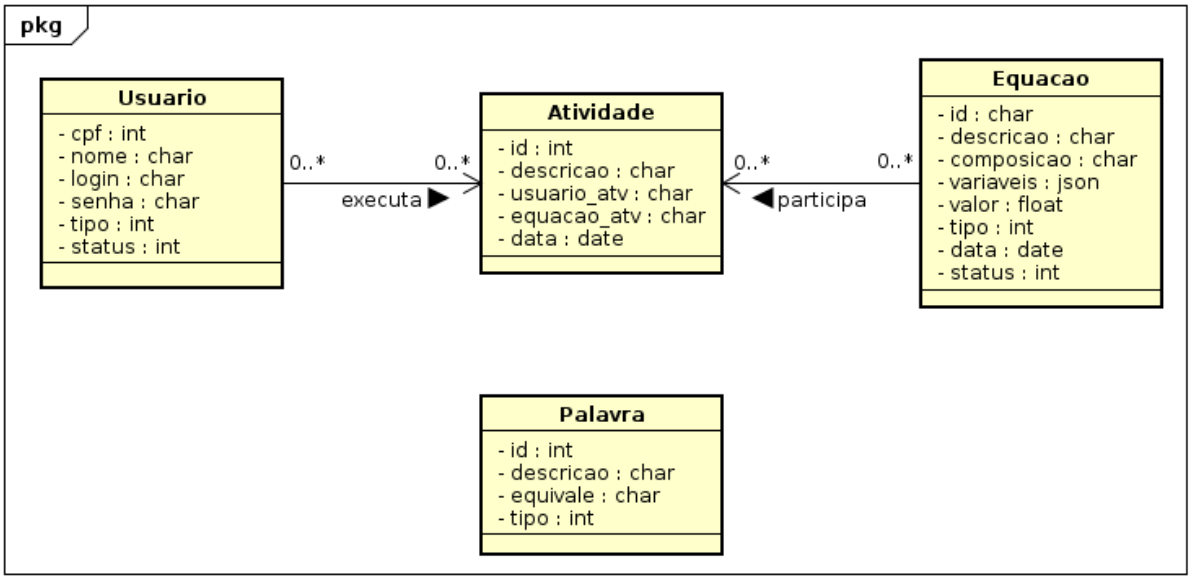
Como foi dito antes, todas as relações do DER do projeto possuem o tipo de relacionamento muitos para muitos, onde várias equações participam de várias atividades e vários usuários executam várias atividades.

Um diagrama de classe foi gerado, para definir as entidades, atributos, quais tipos que os atributos podem assumir na base dados e as relações. Sendo este diagrama definido de acordo com o que foi desenhado no DER, como mostra a Figura 11, representando como seriam compostas as tabelas no banco de dados.

As variáveis do diagrama de classe do projeto assumem diferentes tipos de dados, dos quais temos: ***char***, ***json***, ***float***, ***int*** ou ***date***.

- ***char****:* tipo de dado que representa um conjunto de caracteres.
- ***json****:* nada mais é do que um vetor de objetos, que é composto por *key* e *value* (chave e valor respectivamente).
- ***float****:* representa o conjunto dos números reais. Ex.: 1.0; 2.7; 1.765; etc.
- ***int****:* representa o conjunto dos números inteiros. Ex.: 1; 0; -5; etc.
- ***date:*** dado composto por uma data, possui uma estrutura diferente dos campos baseado em caracteres, sendo mais fácil de manipular as datas com o tipo *date* de que com o tipo *char*.

Figura 11 – Diagrama de classe do projeto.



Fonte: Próprio autor.

Usuário: contém todos os usuários que podem acessar a plataforma diferenciando-os pelo campo CPF.

Os campos da entidade **Usuario** são **cpf**, **nome**, **login**, **senha**, **tipo** e **status** que podem assumir valores do tipo ***int***, ***char***, ***char***, ***char***, ***int*** e **int** respectivamente. Todos os atributos serão definidos pelo usuário que o cadastra, exceto o **status**, que será atribuído pelo sistema.

- **cpf:** o campo é ***int***, pois todos os caracteres especiais são removidos.
- **nome:** o nome pelo qual o usuário será chamado dentro do SI.

- **login:** “apelido” do usuário, usado para acessar o SI, não pode haver mais de um usuário com um mesmo *login*.
- **senha:** a senha de acesso do usuário, para que ele possa validar a entrada dele no SI.
- **tipo:** existem dois tipos de usuário, padrão e administrador, de maneira que o usuário administrador tenha o controle total dos módulos do SI e o usuário padrão tenha acesso só a alguns módulos. O atributo tipo é um valor inteiro que assume os valores de 0 ou 1, 0 para usuário padrão e 1 para usuário administrador.
- **status:** variável de controle que determina se o usuário está ativo ou não, variável auxiliar utilizada para realizar a exclusão lógica do usuário. Quando um usuário está ativo seu valor é 1, caso contrário, o seu valor é 0.

Equação: armazena toda e qualquer equação do SI.

Os campos que compõe a entidade **Equacao** são **id**, **descricao**, **composicao**, **variaveis**, **valor**, **tipo**, **data** e **status** que podem assumir os valores do tipo ***char***, ***char***, ***char***, ***json***, ***float***, ***int***, **date**, e **int** respectivamente. Os campos definidos pelo usuário são **id**, **descricao**, **composicao** e **tipo**. Os outros atributos são gerados automaticamente pelo sistema.

- **id:** o id da equação é definido pelo usuário, sendo um conjunto de caracteres, que por padrão não pode ter caracteres especiais. Ex.: **TF1**.
- **descricao:** uma breve descrição referente a equação.
- **composicao:** a composição da equação propriamente dita. Ex.: ***0.771*I8_DMR+ 0.698*I13_ICF + 0.635 * I7_AS + 0.632*I9_RN+0.454 * I12_EDG***
- **variaveis:** todas as variáveis que compõe a equação. Um exemplo da estrutura do campo “variaveis” seria: [“I8_DMR”, “I13_ICF”, “I7_AS”, “I9_RN”, “I12_EDG”].
- **valor:** é o resultado final da equação, calculado pelo sistema a partir do que foi definido pelo usuário no atributo **composicao.**
- **tipo:** a entidade foi dividida em dois tipos, atribuída e calculada, por conta de ter modos diferentes de obter o resultado. Tipo é um atributo composto por um valor inteiro que assume os valores 0 ou 1, caso seja uma equação do tipo atribuída seu valor é 0, caso contrário, seu valor é 1.

- **data:** data do sistema no momento em que foi realizada a atividade.
- **status:** variável de controle que determina se a equação está ativa ou não, variável auxiliar utilizada para realizar a exclusão lógica das equações. Quando uma equação está ativa seu valor é 1, caso contrário o seu valor é 0.

Atividade: faz o controle de toda atividade realizada por um usuário sobre uma equação. Por conta do controle de usuários não ser o foco, por ora, ele não faz parte deste controle de atividades.

Os campos da entidade **Atividade** são **id**, **usuario**, **equacao**, **descricao** e **data**, que podem assumir os valores do tipo ***int***, ***int***, ***char***, ***char*** e ***date*** respectivamente, e todos os valores são definidos pelo sistema após o usuário realizar alguma atividade com as equações do sistema.

- **id:** é um campo inteiro, do tipo incremental, a primeira atividade terá o **id** com valor 1, a segunda com o valor 2 e assim sucessivamente, servindo para manter o controle das atividades.
- **usuario:** definido pelo cpf do usuário que realizou a atividade.
- **equacao:** composto pelo id da equação que está envolvida na atividade do usuário.
- **descricao:** contém uma breve descrição do que foi realizado pelo usuário Ex.: “José Caruzo cadastrou a equação TF1.”.
- **data:** data do sistema no momento em que foi realizada a atividade.

Palavra: Armazena as palavras chaves da linguagem *Python* que são parte da composição de equações, evitando que sejam confundidas.

Os campos da entidade **Palavra** são **id**, **descricao**, **substituto** e **tipo**, que podem assumir os valores do tipo ***int***, ***char***, ***char*** e ***int*** respectivamente.

- **id:** é um campo inteiro, do tipo incremental, a primeira atividade terá o id com valor 1, a segunda com o valor 2 e assim sucessivamente, servindo para manter o controle das palavras.
- **descricao:** é a palavra reservada de *Python* que está presente na composição da equação e que não devem ser confundidas com variáveis. Ex.: “and”, “not”, “abs”.
- **substituto:** é a palavra que substitui a palavra que está na **descricao** por outra, utilizada para evitar que palavras chaves que o usuário digite não sejam confundidas por conta de não fazer parte da linguagem *Python*. Ex.:

normalmente para operações lógicas de como “E”, “OU” e “NÃO” pode-se representar como: “&&” para “E”, “||” para “OU” e “!” para “NÃO”. Porém em *Python* estas operações lógicas são representadas por “and”, “or” e “not” respectivamente. Desta maneira, “&&” seria a descrição e “and” o substituto.

- **tipo:** é um valor inteiro para definir o tipo de palavra, palavra possui dois tipos, palavra reservada e substituta, caso seja uma palavra reservada será atribuído ao tipo o valor 0, caso contrário será atribuído o valor 1. Caso seja uma palavra nativa, o campo substituto é vazio.

As palavras reservadas representam palavras internas da linguagem *Python,* que devem ser ignoradas na composição de uma equação.

As palavras substitutivas são palavras que são substituídas por outras quando aparecem na composição da equação. Para que o usuário mesmo sem saber como compor uma equação em *Python,* consiga montar sua equação para ser interpretada de forma correta pela linguagem. Ex.:

Tabela 2 – Exemplos de possíveis registros da tabela Palavra.

id	descricao	substituto	tipo
1	&&	And	1
2	and		0

Fonte: próprio autor.

Importante ressaltar que a variável status da entidade usuário e equação existe para que não se faça a exclusão física dos mesmo, pois, para ter o histórico dos registros de todas as atividades, nenhum dos registros pode ser excluído fisicamente da base de dados.

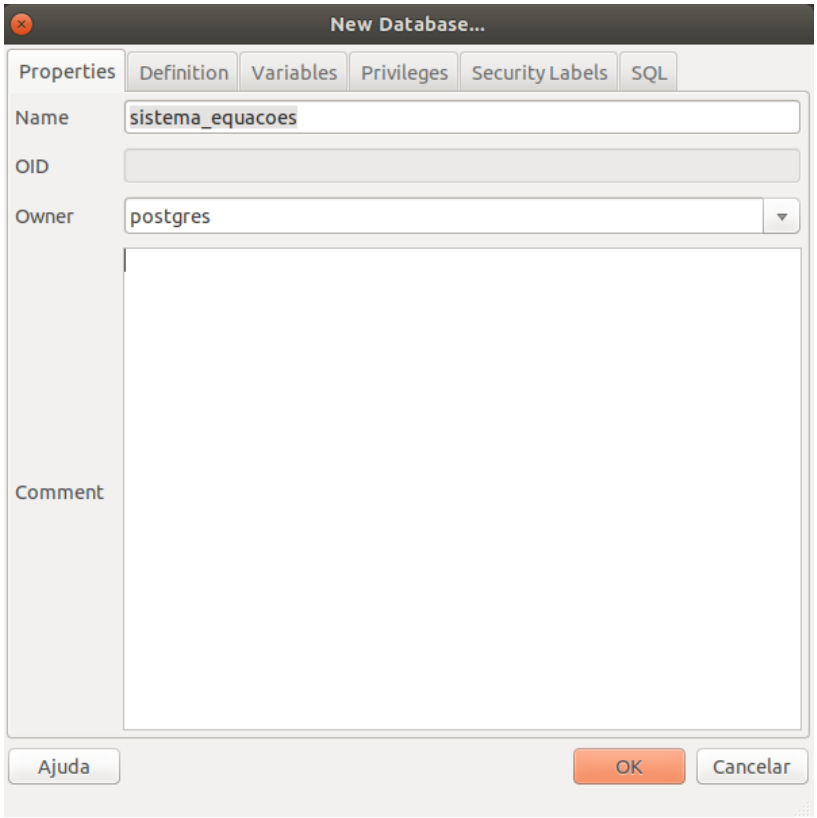
4.3. Desenvolver um SI que possa gerenciar as equações matemáticas em uma base de dados

A parte de desenvolvimento do SI foi subdividida cinco partes, conforme essas subdivisões foram desenvolvidas o projeto foi tomando forma, originou protótipos, até chegar a uma versão final do mesmo, que possa ser aplicada a *softwares* de terceiros.

4.3.1. Base de dados

A base de dados do SI foi planejada de acordo com o diagrama de classe desenhado no tópico 4.2. O primeiro passo foi criar a base de dados com a identificação “sistema_equacoes” dentro do SGBD para gerar as tabelas do SI, representado pela Figura 12.

Figura 12 – Tela de criação de base de dados.



Fonte: próprio autor.

Em seguida, criou-se um arquivo SQL (a linguagem utilizada nos SGBDs estruturados) contendo as quatro entidades descritas no diagrama de classe, que ao ser executado, gerou as tabelas que compunham a base de dados.

Na Figura 13 temos como foi construído o arquivo SQL, com os atributos definidos nos diagramas anteriores.

NOT NULL representa que o atributo não pode ser nulo, *primary key* representa a chave primária de cada tabela e *references* serve para definir as chaves estrangeiras, dizendo qual tabela e atributo que está referenciado.

Figura 13 – Código SQL com as tabelas do SI.

```sql
CREATE TABLE USUARIO(
     cpf varchar(11) primary key,
     nome varchar(50) NOT NULL,
     login varchar(50) NOT NULL,
     senha varchar(50) NOT NULL,
     tipo integer NOT NULL,
     status integer NOT NULL
);

CREATE TABLE EQUACAO(
     id varchar(50) primary key,
     descricao varchar(100) NOT NULL,
     composicao varchar(500) NOT NULL,
     variaveis json NOT NULL,
     valor float NOT NULL,
     tipo integer NOT NULL,
     data date NOT NULL,
     status integer NOT NULL
);

CREATE TABLE ATIVIDADE(
     id serial primary key,
     descricao varchar(200) NOT NULL,
     usuario_atv varchar(11) references USUARIO(cpf) NOT NULL,
     equacao_atv varchar(50) references EQUACAO(id) NOT NULL,
     data date NOT NULL
);

CREATE TABLE PALAVRA(
     id serial primary key,
     descricao varchar(50) NOT NULL,
     equivale varchar(50),
     tipo integer NOT NULL
);
```

Fonte: próprio autor.

Após isto, realizou-se alguns cadastros manualmente para testar a relação entre as entidades, usuario, equacao e atividade. Todos os campos foram devidamente preenchidos e o SGBD não alegou nenhum erro, segue nas Figuras 14, 15 e 16 os registros cadastrados.

Figura 14 – Registro da tabela “usuario” cadastrado manualmente para teste.

cpf [PK] character varying(11)	nome character varying(50)	login character varying(50)	senha character varying(50)	tipo integer	status integer
01674734255	José Caruzo	jcaruzo	admin123	1	1

Fonte: próprio autor.

Figura 15 – Registros da tabela “equacao” cadastrados manualmente para teste.

id [PK] chara	descricao character varying(100)	composic characte	variaveis json	valor double precision	tipo integer	data date	status integer
x	Incógnita x	2	[]	2	1	2018-06-14	1
y	Incógnita y	3	[]	3	1	2018-06-14	1
z	Incógnita z	x + y	["x", "y"]	5	1	2018-06-14	1

Fonte: próprio autor.

Figura 16 – Registros da tabela “atividade” cadastrados manualmente para teste.

id [PK] serial	descricao character varying(200)	usuario_atv character varying(11)	equ cha	data date
1	José Caruzo cadastrou a equação x.	01674734255	x	2018-06-14
2	José Caruzo cadastrou a equação y.	01674734255	y	2018-06-14
3	José Caruzo cadastrou a equação z.	01674734255	z	2018-06-14

Fonte: próprio autor.

Por fim, foi necessário configurar no *Django* as informações do banco de dados e da base de dados, para que ele pudesse ser devidamente utilizado pelo *framework*.

Para poder utilizar a base de dados o *framework* do *Django* precisa das informações que qual é o SGBD utilizado, o nome da base de dados, o usuário e senha, onde está hospedado o servidor e qual é a porta de acesso do servidor. Essas informações devem ser preenchidas no arquivo *settings.py*, utilizando um objeto com o nome de *DATABASES*. Na figura 17, pode-se observar o trecho de código contendo as devidas configurações realizadas.

Figura 17 – Código para configuração de base de dados do *Django*.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'sistema_equacoes',
        'USER': 'postgres',
        'PASSWORD': '123',
        'HOST':'localhost',
        'PORT':'5432'
    }
}
```

Fonte: próprio autor.

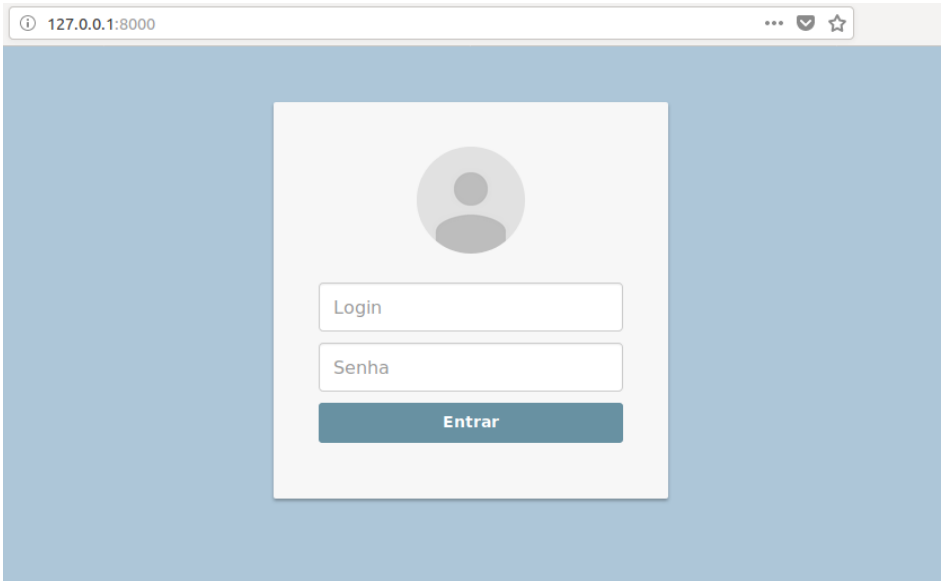
4.3.2. Página de *login*

Antes de desenvolver qualquer outra funcionalidade dentro do SI, o módulo

desenvolvido foi o de *login*, pois as funcionalidades disponíveis mudam de acordo com o tipo de usuário que está o utilizando.

A página de login é composta por duas caixas de texto, que devem ser preenchidas com o login e senha do usuário, e um botão para validar seu acesso, o *layout* final da página é o mostrado na Figura 18.

Figura 18 – Página de login do SI.



Fonte: próprio autor.

Após o usuário submeter seu login e senha, caso seja um usuário válido, o mesmo é direcionado a página principal do SI, que até este momento era uma página em branco somente com o botão de *logout*, conforme segue na imagem 19.

Figura 19 – Página teste para representar a página principal do SI.

Fonte : próprio autor.

Por fim foram realizados testes utilizando o usuário administrador cadastrado manualmente na base de dados, a fim de verificar a funcionalidade correta do login,

testando acessar tanto com o login e senha válidos, quanto inválidos.

4.3.3. Página principal do SI

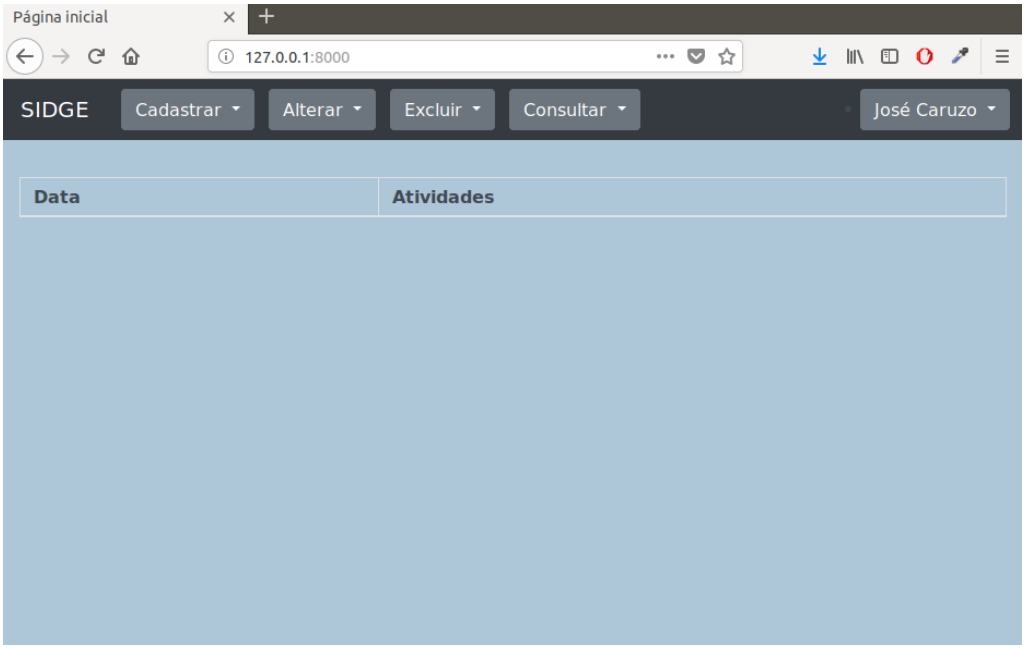
A página principal é onde estão disponibilizados todos os módulos do sistema, ou seja, todas as atividades que o usuário pode executar.

A página principal foi desenvolvida contendo botões organizados em uma aba de navegação no topo da página, com alinhamento da esquerda para a direita. Os botões criados foram “Cadastrar”, “Alterar”, “Excluir” e “Consultar”, respectivamente, que servem para direcionar o usuário da página principal para o módulo que o mesmo deseja executar. O botão de *logout* também fica na aba de navegação, porém no canto direito separado dos demais botões.

No corpo da página contém apenas uma tabela, cujo o objetivo é armazenar a data e descrição das atividades realizadas pelo usuário.

A Figura 20 representa o *layout* final da página principal, contendo todos os componentes descritos acima.

Figura 20 – Página principal do SI.



Fonte: próprio autor.

4.3.4. Módulos de gerenciamento de usuário

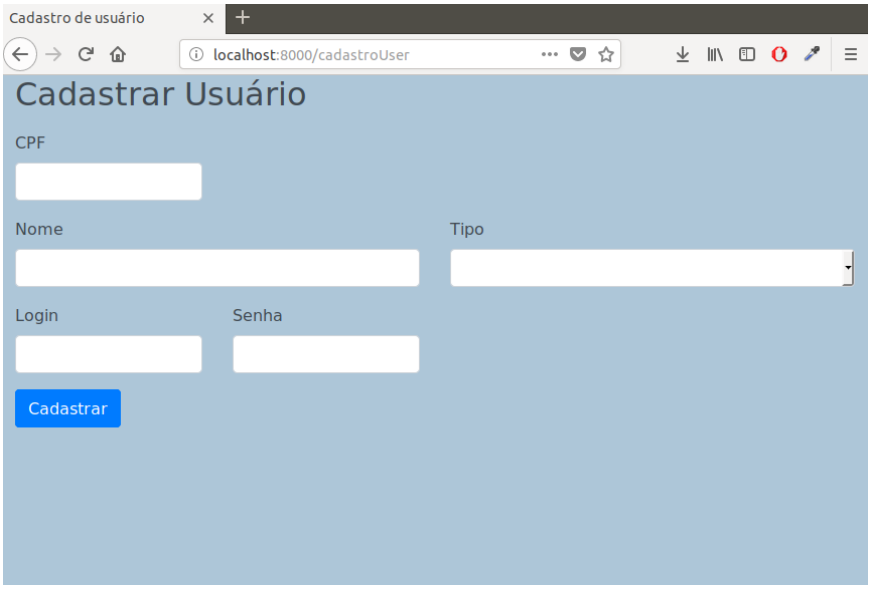
Os módulos de gerenciamento de usuário só estão disponíveis aos

administradores do SI. Os módulos disponíveis são:

- Cadastrar: todos os dados para realizar o cadastro de um novo usuário devem ser devidamente informadas, cpf, nome, login, senha e tipo, após clicar no botão “Cadastrar” o cadastro é realizado pelo sistema, o status do usuário cadastrado recebe o valor 1, para informar que o mesmo está ativo.

A Figura 21 demonstra a tela de cadastro de usuário, contendo todos os campos que precisam ser preenchidos para realizar o cadastro.

Figura 21 – Página para cadastro de usuário.



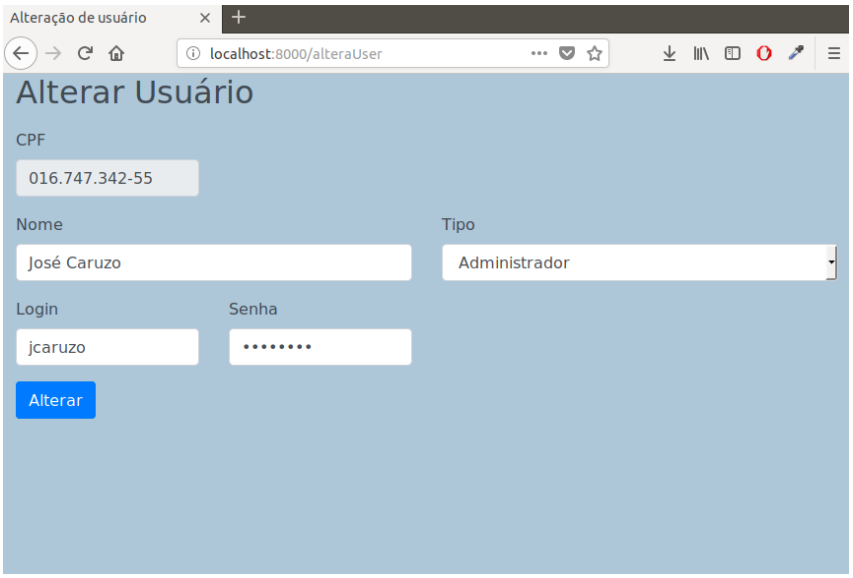
Fonte : próprio autor.

- Alterar: O cpf do usuário a ser alterado deve ser informado, caso o cpf submetido seja um cpf válido de um usuário da base de dados, o sistema retorna os dados do mesmo.

Os atributos que estão disponíveis para supostas alterações são: nome, login, senha e tipo. Após realizadas as alterações desejadas nos atributos do usuário, basta clicar no botão “Alterar” para confirmar a alteração do registro.

A Figura 22 demonstra a tela de alteração, logo após a consulta do cpf de um usuário válido.

Figura 22 – Página para alteração de usuário.



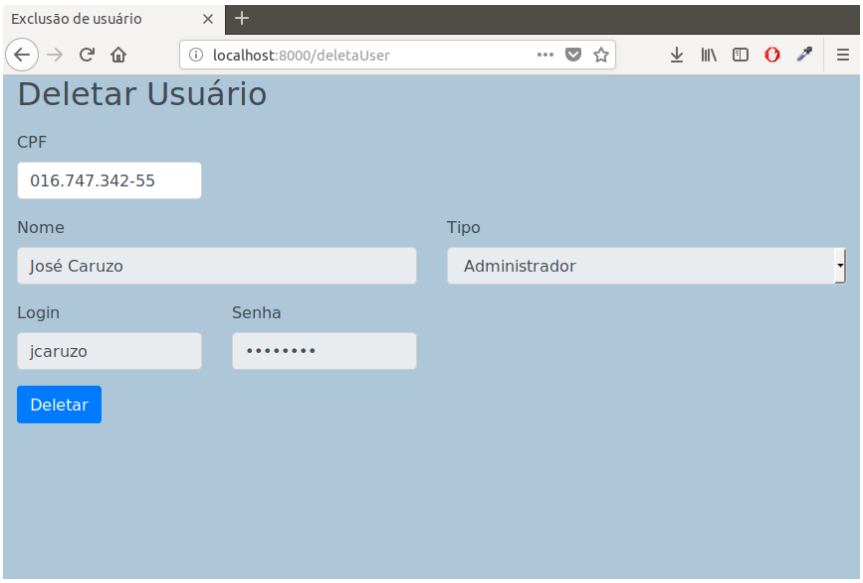
Fonte: próprio autor.

- Excluir: O cpf do usuário a ser excluído deve ser informado, caso o cpf submetido seja um cpf válido de um usuário da base de dados, o sistema retorna os dados do mesmo para a conferência.

  Ao apertar o botão “Deletar”, é feita a exclusão lógica do usuário informado, modificando o valor do atributo status para 0, dizendo desta maneira que este usuário não está mais apto a utilizar o sistema.

  A Figura 23 demonstra a tela de exclusão, logo após a consulta do cpf de um usuário válido.

Figura 23 – Página para exclusão de usuário.



Fonte: próprio autor.

Durante a fase de teste dos módulos de gerenciamento de usuário, garantiu-se que não ocorra: cadastros duplicados, *logins* duplicados, alteração de usuários já excluídos e exclusão usuários já excluídos.

4.3.5. Módulos de gerenciamento de equação

Os módulos de gerenciamento de equação estão disponíveis à todos os usuários do SI. Os módulos disponíveis são:

- Cadastrar: todos os dados para realizar o cadastro de uma nova equação devem ser devidamente preenchidos, o id, descricao, composicao e tipo, todos os outros atributos são definidos automaticamente pelo sistema.

  O atributo status da equação é cadastrado como 1 para informar que a mesma está ativa.

  O atributo variaveis é definido de acordo com todas as palavras que foram encontradas da composicao, que representam ids de outras equações na base de dados.

  O atributo valor é o resultado que o sistema obtém após calcular a equação.

  O atributo data é definido a partir da data do sistema no momento

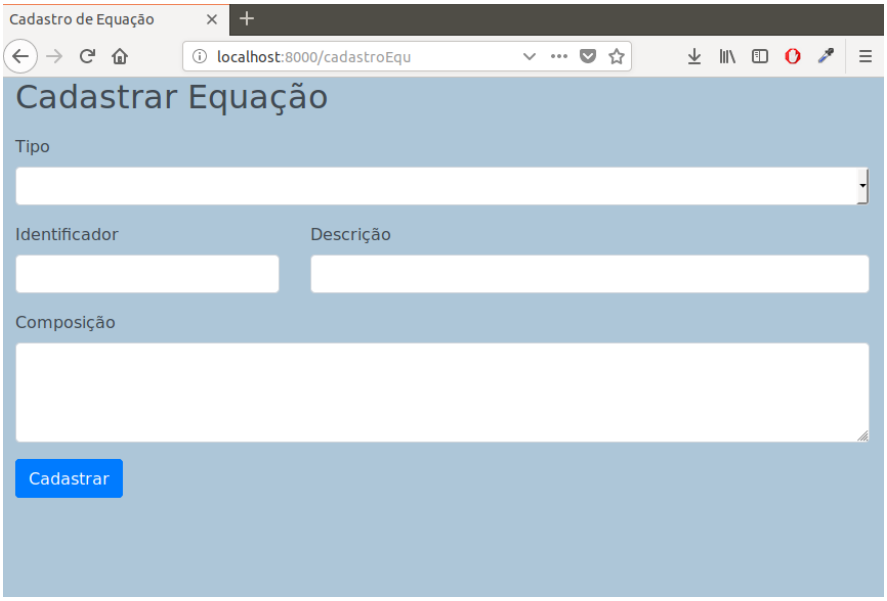
que a equação foi cadastrada.

É necessário dar uma atenção especial caso tipo seja 0, referente uma equação atribuída, porquê o modo de definir a composição da equação muda, pois, a intenção neste caso é criar equações referente a registros de outras bases de dados.

Pensando na estrutura de uma base de dados, a melhor maneira seria definir a base de dados de onde se deseja retirar as informações, qual tabela, coluna e por fim, definir o registro atribuído esta equação.

A página de cadastro é apenas uma, porém, para cada tipo de equação ela possui um *layout* diferente, como demonstrado nas Figuras 24 e 25.

Figura 24 – Página para cadastro de equação.



Fonte: próprio autor.

Figura 25 – *Layout* da página de cadastro de equação para equação atribuída.

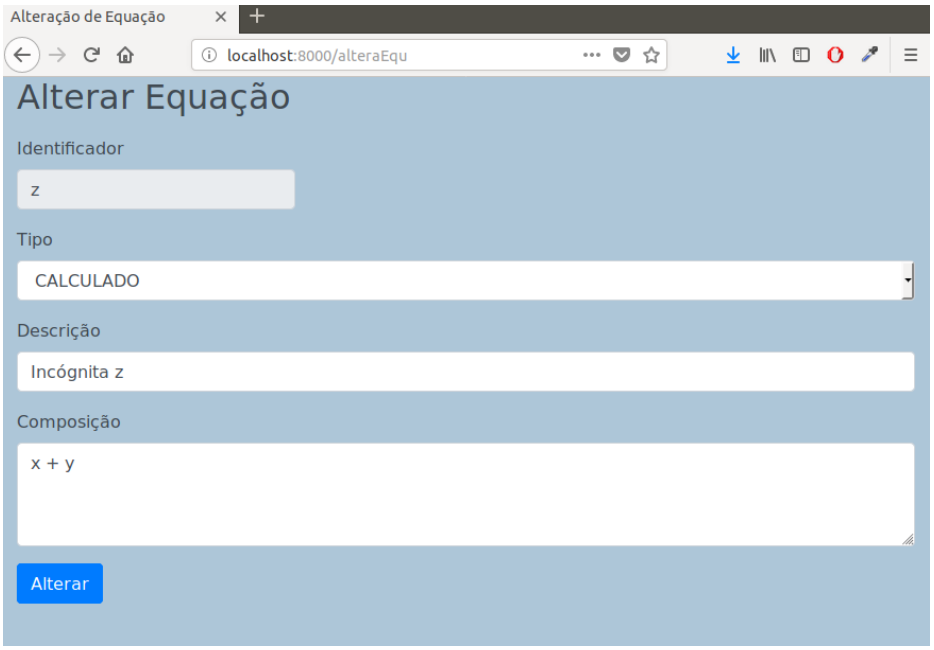
Fonte: próprio autor.

- Alterar: O id da equação a ser alterada deve ser informado, caso o id submetido seja um id válido de uma equação da base de dados, o sistema retorna os dados da mesma.

  Os atributos que estão disponíveis para supostas alterações são: descricao, composicao e tipo. Após realizadas as alterações desejadas nos atributos da equação, basta clicar no botão “Alterar” para confirmar a alteração do registro.

  A Figura 26 demonstra a tela de alteração, logo após a consulta do id de uma equação válida.

Figura 26 – Página para alteração de equação.



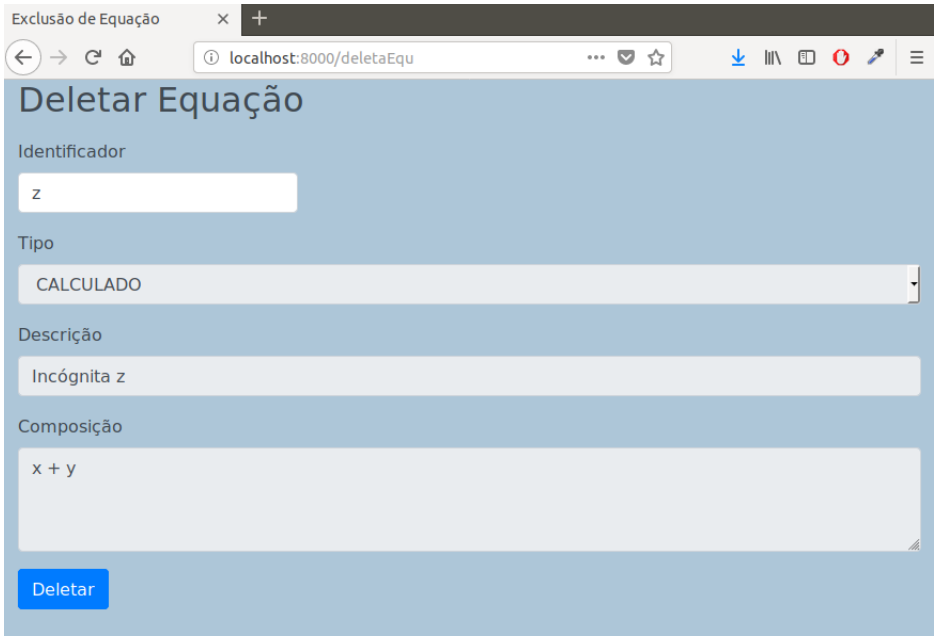
Fonte: próprio autor.

- Excluir: O id da equação a ser excluída deve ser informado, caso o id submetido seja um id válido de uma equação da base de dados, o sistema retorna os dados do mesmo para a conferência.

  Ao apertar o botão “Deletar”, é feita a exclusão lógica da equação informada, modificando o valor do atributo status para 0, dizendo desta maneira que esta equação não está mais disponível no sistema.

  A Figura 27 demonstra a tela de exclusão, logo após a consulta do id de uma equação válida.

Figura 27 – Página para exclusão de equação.



Fonte: próprio autor.

Durante a fase de teste dos módulos, garantiu-se que não ocorra: cadastros duplicados, exclusões de equações que compõe outras equações, alterações de equações já excluídas e exclusões equações já excluídas.

4.3.6. Consultas de usuário, equação e atividade

As consultas são módulos “extras”, uma vez que não realizam nenhuma atividade na base de dados, sendo meramente ilustrativas ao usuário.

As consultas de usuário estão limitadas somente aos administradores, assim como os módulos de gerenciamento do usuário.

As consultas de equações e de atividades estão disponíveis a todos os usuários do sistema.

Todas as consultas se baseiam em consultar todos os registros que não foram excluídos. Todos eles são listados em uma tabela, a partir disto o usuário pode realizar suas consultas a partir de filtros de buscas e ordenações.

As ordenações são feitas a partir das colunas, para ordená-las basta clicar no “cabeçalho” da mesma. A ordenação pode ser crescente ou decrescente, dependendo da necessidade.

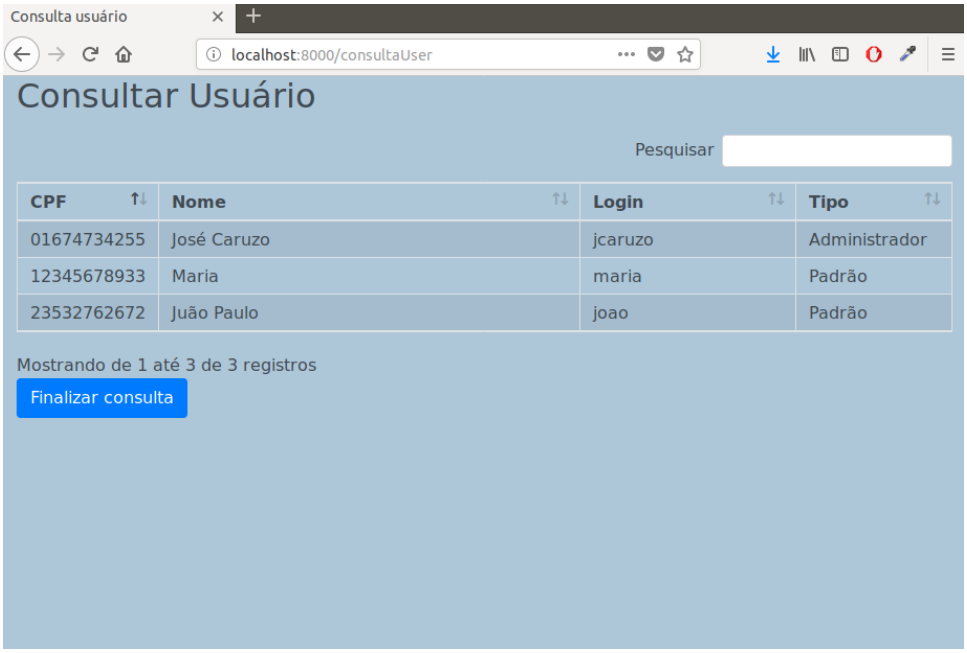
A representação de qual coluna está ordenada e em qual sentido é feita a partir de

setas que ficam ao lado da descrição da coluna. Cada coluna possui duas seta, uma seta é apontada para cima e demonstra a ordem crescente de ordenação, a outra seta é apontada para baixo e demonstra a ordem decrescente.

Para expor por qual coluna a tabela está ordenada e em qual sentido, a seta que representa esta ordenação está na cor preta.

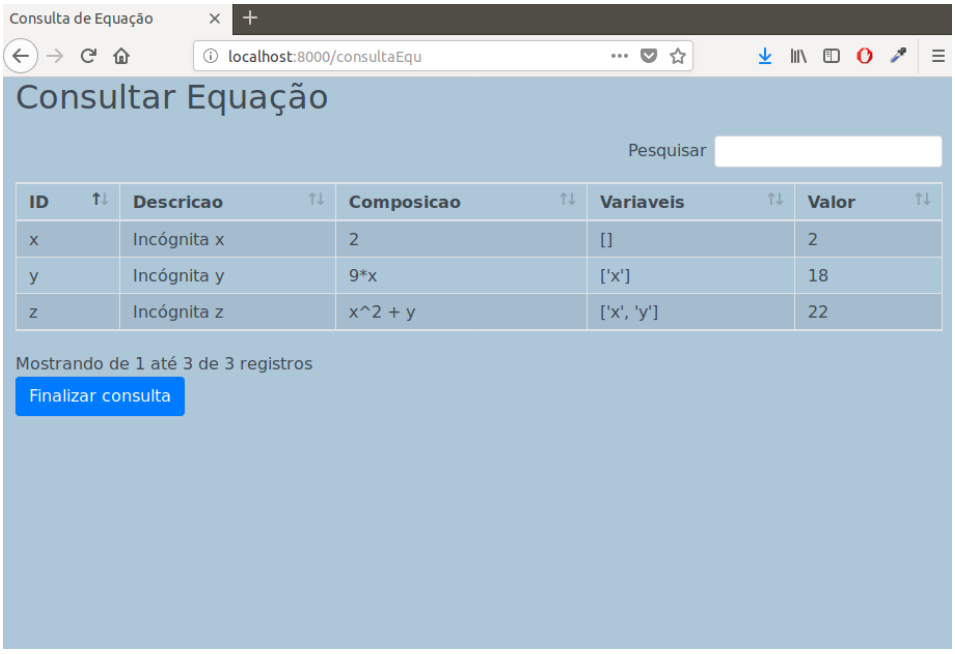
As figuras 28, 29 e 30 são referentes às consultas. Sendo que, a figura 28 mostra a tela de consulta de usuário com a tabela ordenada pelo cpf, de forma crescente, a figura 29 mostra a tela de consulta de equações com a tabela ordenada pelo id, de forma crescente e a figura 30 mostra a tela principal do SI, onde se está realizando as consultas de atividades pela data, de forma decrescente.

Figura 28 – Página para consulta de usuário.



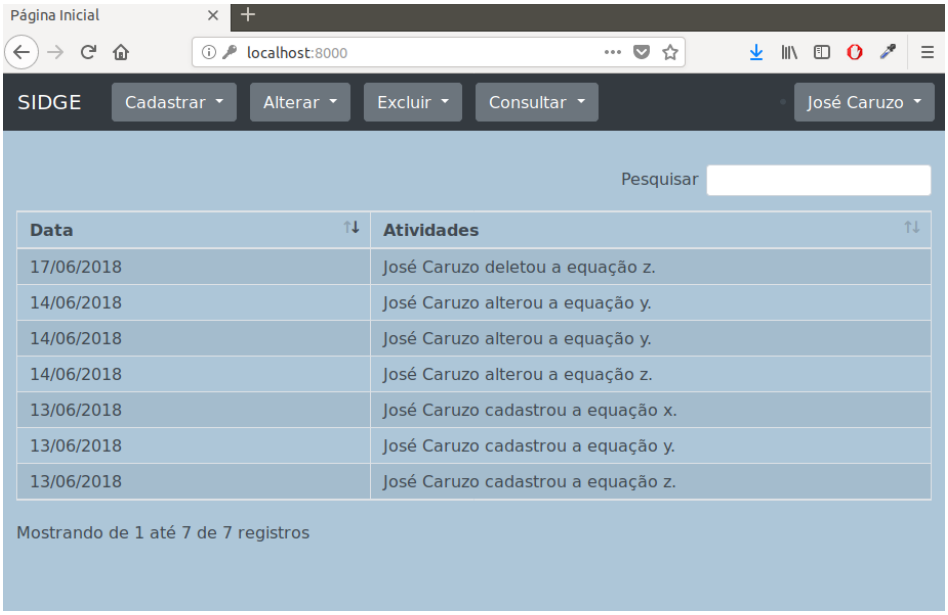
Fonte: próprio autor.

Figura 29 – Página para consulta de equação.



Fonte: próprio autor.

Figura 30 – Página principal com a consulta de atividades.



Fonte: próprio autor.

Realizaram-se testes para verificar a funcionalidade do filtro de busca e as ordenações das colunas da tabela, de acordo com as necessidades do usuário.

4.4. Aplicar o SI a um *software* onde há a necessidade de reformulação constantes de equações

Para fazer a integração do SI com o SIAMI, foi simulada a situação como se os dois sistemas estivessem no mesmo servidor, ou seja, dividindo o mesmo SGBD, mesmo que com base de dados diferentes. Além disso, fizeram-se algumas alterações nas configurações do SI do projeto, para que ele reconheça a base de dados do SIAMI.

A configuração segue o mesmo modelo do tópico 4.3.1. assim como o mostrado na figura 17, a diferença é que neste momento é adicionada outra base de dados, precisando diferenciar de cada uma das bases de dados por uma identificação, para ser reconhecido pelo *framework Django* corretamente. Sendo assim, a base de dados do SI ficou como "*default*" e a base de dados do SIAMI ficou como "siami_web" para realizar esta configuração.

O trecho de código para adicionar a base de dados do SIAMI pode ser visto na figura 31, para o melhor entendimento.

Figura 31 – Código para adicionar a base de dados do SIAMI no *Django*.

```
# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'sistema_equacoes',
        'USER': 'postgres',
        'PASSWORD': '123',
        'HOST':'localhost',
        'PORT':'5432'
    },
    'siami_web':{
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'siami_web',
        'USER': 'postgres',
        'PASSWORD': '123',
        'HOST':'localhost',
        'PORT':'5432'
    }
}
```

Fonte: próprio autor.

Após realizar as alterações no DATABASES o SI do SIDGE pode acessar a base de dados do SIAMI, disponibilizando que as construções das equações utilizem os dados do SIAMI como base. Foram feitos testes de consulta à base de dados do SIAMI, a fim de validar tais configurações.

4.5. Testar SI

Esta fase de testes é referente aos testes de sistema, verificando a aplicabilidade do SIDGE ao SIAMI, de maneira que ele consiga aplicar as equações do SIAMI e que os resultados obtidos pelo SIDGE coincidam com os resultados gerados pelo SIAMI.

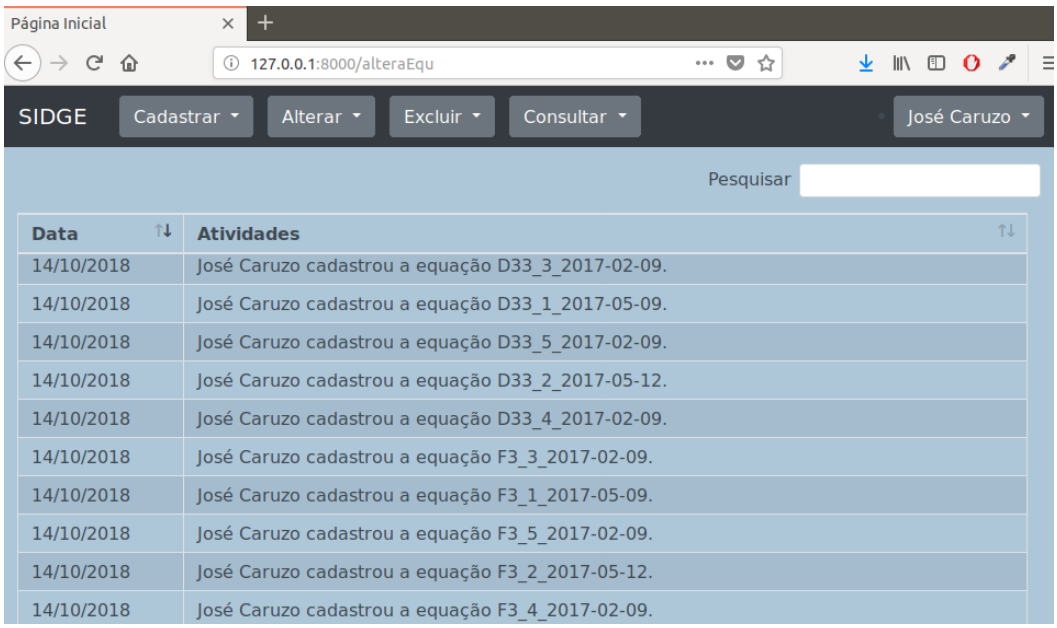
Foram realizados diversos cadastros, alterações e exclusões baseados no IVI, para realizar esta verificação.

Os resultados armazenados na base de dados foram comparados com o módulo do SIAMI de geração de fatores. As cores vermelho, amarelo e verde do módulo de geração de fatores, representam os valores 2, 1 e 0 respectivamente.

Após fazer as comparações, pode-se afirmar que todas as equações que participaram do teste foram devidamente geradas a partir do SI deste projeto, sanando todas as necessidades do SIAMI. A intenção é que o projeto consiga acompanhar não só a versão atual do sistema do SIAMI e do IVI, como todas as próximas versões que forem criadas.

Para demonstrar os resultados de forma mais clara, foram geradas várias imagens para exemplificar o resultados obtidos pelo SI, observadas nas figuras 32, 33, 34 e 35. O Fator usado para a demonstração é o fator de Condição Social (Fator 3) que é gerado no SIAMI, utilizando os dados de cinco idosos do sistema.

Figura 32 – Página principal do SI com os as atividades de cadastro do Fator 3.



Fonte: próprio autor.

Os fatores do SIAMI são gerados individualmente de idoso para idoso, obtendo os resultados após uma consulta pelo cpf do idoso. Para averiguar os resultados, nas figuras 33 e 34 foi utilizado o idoso 1 do sistema, para ficar melhor para a comparação, onde a figura 33 contém as consultas de equação filtrando pelo idoso 1 e a figura 34 contém a consulta dos fatores do mesmo idoso.

A ordem dos indicadores que geram o fator de Condição Social é I03, I04, I09, I12 e I14, lembrando que 2 equivale a vermelho, 1 a amarelo e 0 a verde.

Figura 33 – Página de consulta de equações utilizando o idoso 1 como filtro.

ID	Descricao	Variaveis	Valor
arranjofamiliar_1_2017-05-09	Arranjo familiar	[]	2
condicaomoradia_1_2017-05-09	Condição de moradia	[]	1
D31_1_2017-05-09	Centro de grupo 1	['TF3']	1.891
D32_1_2017-05-09	Centro de grupo 2	['TF3']	0.858
D33_1_2017-05-09	Centro de grupo 3	['TF3']	0.496
depressao_1_2017-05-09	depressao	[]	8
F3_1_2017-05-09	Classificação do fator 3	['D31', 'D32', 'D33']	2
I03_1_2017-05-09	Indicador 3: Renda per capita	['rendapercapita']	2
I04_1_2017-05-09	Indicador 04: Condição de moradia	['condicaomoradia']	0
I09_1_2017-05-09	Indicador 09: Risco Nutricional	['risconutricional']	1
I12_1_2017-05-09	Indicador 12: Depressão EDG	['depressao']	1
I14_1_2017-05-09	Indicador 14: Arranjo Familiar	['numpessoaresidente']	2

Fonte: próprio autor.

Figura 34 – Página de consulta do fator de Condição Social do SIAMI.

Fonte: próprio autor.

Para averiguar o cadastro correto das equações pelo SI foram realizadas diversas consultas a base de dados do SI para verificar se os resultados estavam devidamente armazenados, uma das consultas do fator 3 é demonstrada na figura 35.

Figura 35 – Registros do SI dispostos dentro da base de dados.

id [PK] character varying(50)	descricao character varying(100)	composicao character varying(500)	variaveis json	valor doubl	tipo integer	data date	status integer
F3_2_2017-05-12	Classificação do fat	(min(D31,D32,D33)==D	["D31", "D32", "D33"]	1	1	2018	1
F3_3_2017-02-09	Classificação do fat	(min(D31,D32,D33)==D	["D31", "D32", "D33"]	2	1	2018	1
F3_4_2017-02-09	Classificação do fat	(min(D31,D32,D33)==D	["D31", "D32", "D33"]	0	1	2018	1
F3_5_2017-02-09	Classificação do fat	(min(D31,D32,D33)==D	["D31", "D32", "D33"]	0	1	2018	1
I03_1_2017-05-09	Indicador 3: Renda p	(rendapercapita >= 1	["rendapercapita"]	2	1	2018	1
I03_2_2017-05-12	Indicador 3: Renda p	(rendapercapita >= 1	["rendapercapita"]	1	1	2018	1
I03_3_2017-02-09	Indicador 3: Renda p	(rendapercapita >= 1	["rendapercapita"]	2	1	2018	1
I03_4_2017-02-09	Indicador 3: Renda p	(rendapercapita >= 1	["rendapercapita"]	1	1	2018	1
I03_5_2017-02-09	Indicador 3: Renda p	(rendapercapita >= 1	["rendapercapita"]	0	1	2018	1
I04_1_2017-05-09	Indicador 04: Condiç	(condicaomoradia ==	["condicaomoradia"]	0	1	2018	1
I04_2_2017-05-12	Indicador 04: Condiç	(condicaomoradia ==	["condicaomoradia"]	1	1	2018	1
I04_3_2017-02-09	Indicador 04: Condiç	(condicaomoradia ==	["condicaomoradia"]	0	1	2018	1
I04_4_2017-02-09	Indicador 04: Condiç	(condicaomoradia ==	["condicaomoradia"]	0	1	2018	1
I04_5_2017-02-09	Indicador 04: Condiç	(condicaomoradia ==	["condicaomoradia"]	0	1	2018	1
I09_1_2017-05-09	Indicador 09: Risco	(risconutricional <	["risconutricional"]	1	1	2018	1
I09_2_2017-05-12	Indicador 09: Risco	(risconutricional <	["risconutricional"]	1	1	2018	1

Fonte: próprio autor.

5. CONCLUSÃO

Com a chegada da era da informação, cada vez mais SI's se tornam ferramentas importantes no dia a dia, onde a mudança é necessária para a melhoria. Com esta necessidade de mudança, os SI's também precisam evoluir, e acompanhá-las é um trabalho que exige tempo e dedicação.

No quesito SI com estrutura de dados temporais a tendência se torna a facilidade no reaproveitamento de módulos, códigos, estrutura, entre outros. Pois construir algo que não está aberto a mudanças nos tempos de hoje não traz benefícios, pois, atrapalha o processo evolutivo.

Este trabalho propôs gerar um projeto que gerenciasse equações utilizando o SIAMI como SI piloto, a partir de uma base de dados e um SI web.

Depois de realizar estudos sobre gerenciamento de equações foi possível constatar que a formulação de uma hierarquia entre equações era viável e que supriria a necessidade, de manter cálculos de equações diretamente na composição dos SI's com estrutura de dados temporais.

O controle de equações feito a partir de armazenamento em base de dados mostrou diversos fatores positivos, dentre eles podemos citar alguns como:

- Fácil controle de versionamento de equações.
- Evitar o recálculo desnecessário das equações, partindo do pré suposto que os valores são salvos e recalculados somente quando ocorrem alterações.
- Melhor desempenho do que utilizar as equações diretamente no código fonte de um SI, pois, em vez de acessar dados e calculá-los ele só precisa acessar o resultado da equação armazenada.
- Dar liberdade ao usuário que tem um conhecimento sobre a composição das equações, realizar as alterações quando necessário.

Importante ressaltar que este trabalho foi aplicado ao SI do SIAMI, porém ele também serve como solução de gerenciamento de equações para outros SI's que tem embasamentos em cálculos matemáticos e estatísticos em sua composição. Porém como não foi aplicado a nenhum outro sistema para averiguar a sua veracidade, pode precisar de algumas alterações para suprir todos os requisitos necessários.

Como é uma base de dados, os SI's que forem utilizar este trabalho precisam fazer consultas e acessar os resultados das equações armazenadas, e pensando nisso, para trabalhos

futuros, sugere-se que sejam feitos *plugins* para *Python* e outras linguagens de programação, para criar módulos que realizem de forma mais ágil e simples este consumo de dados.

Ao aplicar a este trabalho o modelo de vida de *software* como o de prototipação, significa que ele pode passar por novas versões e se aperfeiçoar a cada vez mais na resolução do problema encontrado, realizando melhores tratativas para diferentes tipos de equações, com melhor desempenho, maior confiabilidade e robustez.

REFERÊNCIAS

ALVARENGA, M. R. M. **Avaliação da capacidade funcional, do estado de saúde e da rede de suporte social do idoso atendido na Atenção Básica.** [tese]. São Paulo: Escola de Enfermagem da Universidade de São Paulo; 2008.

ALVES, J. F. **Caos: uma perspectiva probabilística**, Centro de Matemática da Universidade do Porto, 2001, Disponível em: <**http://www.fc.up.pt/cmup/activities/aberto.html>**, Acessado em: 15 de Setembro de 2018.

ANDRADE, A.; RODRIGUES, L.; SELEME A.; SOUTO, R. **Pensamento sistêmico: caderno de campo – o desafio da mudança sustentada nas organizações e na sociedade**. Porto Alegre: Bookman, 2006.

BARBOSA, B. G. **Estabilidade de sistemas detectáveis com custo médio a longo prazo limitado**. São Paulo: Brenno Gustavo Barbosa, 2012.

BATISTA, E. O. **Sistema de informação: o uso consciente da tecnologia para o gerenciamento.** São Paulo: Saraiva, 2004.

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML.** 2ª Ed. Rio de Janeiro: Elsevier, 2007.

BORGES, L.E. ***Python*** **para Desenvolvedores,** 2010**.** 2ª Ed. Disponível em: <**https://ark4n.files.wordpress.com/2010/01/python_para_desenvolvedores_2ed.pdf>,** Acessado em: 7 de Outubro de 2018.

DATE, C. J. **INTRODUÇÃO A SISTEMAS DE BANCOS DE DADOS.** 8ª Ed. Rio de Janeiro: Elsevier, 2003.

DAVENPORT, T.H. ; PRUSAK, L. **Conhecimento empresarial.** 15ª Ed. Rio de Janeiro: Campus, 2003.

DODONOV, E. **Uma abordagem de predição da dinâmica comportamental de processos para prover autonomia a ambientes distribuídos**. São Paulo: Evgueni Dodonov, 2009.

DRUCKER, P. F.. **Introdução à administração**. Tradução de Carlos A. Malferrari. 4ª Ed. São Paulo : Thomson/Pioneira, 2002.

FAYYAD, U. M.; GRINSTEIN, G.; WIERSE, A. **Information Visualization in Data Mining and Knowledge Discovery,** 2001, Morgan Kaufmann.

FERNANDES, A. C. **Scorecard dinâmico – em direção à integração da dinâmica de sistemas com o balanced scorecard.** Tese de Doutorado, CO PPE/UFRJ, 2003.

FERREIRA, G. E. **Avaliação de Sistemas de Apio à Decisão na perspectiva do usuário da informação: o data warehouse como suporte à estratégia organizacional**. 2010. Dissertação (Mestrado em Ciência da Computação) – Escola de Ciência da Informação, Universidade Federal de Minas Gerais, Belo Horizonte.

FREITAS, H.; OLIVEIRA, M.; MOSCAROLA, J.; LUCIANO, E.M. **A tomada de decisão e o correio eletrônico: reflexões sobre o usuário brasileiro.** XXVIII CLADEA, Lima/Peru, 2003.

GEOGEBRA, *Geogebra Classic Manual*, 2018. Disponível em: <**https://wiki.geogebra.org/en/Manual**>. Acessado em: 22 de Setembro de 2018.

LAKATOS, E. M.; MARCONI, M. A. **Metodologia do Trabalho Científico.** 7ª Ed. Atlas, 2007.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de Informação Gerenciais**. 7ª Ed. São Paulo: Prentice Hall, 2004.

MATHWAY, *About Mathway*, 2018. Disponível em: <**https://www.mathway.com/about>.** Acessado em: 22 de Setembro de 2018.

MESQUITA, J. G. **Measure functional differential equations and impulsive functional dynamic equatoins on time scales**. São Paulo: Jaqueline Godoy Mesquita, 2012.

MICROSOFT, *Microsoft Matematics,* 2010. Disponível em: <**https://www.microsoft.com/en-us/download/details.aspx?id=17786>** , Acessado em: 22 de Setembro de 2018.

OGATA, K. **System Dynamics.** 4ª Ed. Prentice Hall, 2003.

MELO, A. C. V. de; SILVA, F. S. C. da. **Princípios de Linguagens de Programação**. São Paulo: Edgard Blücher Ltda, Melo, 2003.

MRACK, M.; MOREIRA, D. **Sistemas Dinâmicos Baseados em Metamodelos**. II *Workshop* e Computação e Gestão da Informação (WCOMPI), Lajeado, 2003.

O'BRIEN, J. **Sistemas de Informação e as decisões gerenciais na era da internet.** São Paulo, 2º Ed., 2004.

PAIVA, M. R. **Matemática – Volumes 1, 2 e 3**. 1ª Ed . São Paulo: Moderna, 2009.

PFLEEGER, S.L. **Engenharia de *Software*: Teoria e Prática**, 2ª Ed. São Paulo: Prentice Hall, 2004.

POBLACION, D. A.; WITTER, G. P.; SILVA, J. F. M. da. (Orgs.). **Comunicação e produção científica: contexto, indicadores, avaliação.** São Paulo: Angellara, 2006.

POSTGRESQL, *The PostgreSQL Global Development Group*. **PostgreSQL 10.5 Official Documentation.** Disponível em: **<https://www.postgresql.org/docs/10/static/index.html>,** Acessado em: 7 de Outubro de 2018.

PRATES, G. A.; OSPINA, M. T. **Tecnologia da informação em pequenas empresas: fatores de êxito, restrições e benefícios**. Curitiba: Revista de Administração Contemporânea, 2004.

PREECE, J.; ROGERS, Y.; SHARP, H. **Design de Interação: além da interação homem-computador.** Porto Alegre: Bookman, 2005.

PRESSMAN, R. S. **Engenharia de *software* – Uma abordagem profissional**. São Paulo: Makron Books, 2011.

SANTANA NETO, O. ***Python* e *Django* – Desenvolvimento Ágil de Aplicações *Web***. São Paulo: Novatec, 2010.

SANTOS, A. C. K. dos. **Modelos Mentais e a Dinâmica de Sistemas como uma Metodologia para a pesquisa Educacional**. 2004. Fundação Universidade Federal do Rio Grande – FURG, Rio Grande do Sul.

SASS, G. G.; ALVARENGA, M. R. M.; OLIVEIRA, M. A. C.; FACCENDA, O. **Sistema de informação para monitoramento da saúde de idosos.** Journal of Health Informatics, v. 4, p. 209-215, 2012.

SDEP, *System Dynamics in Education Project*. ***Road Maps - A Guide to Learning Systems Dynamics***, *MIT Sloan School of Management – System Dynamics Group. Massachusetts, 2005.* Disponível em: **<http://web.mit.edu/sysdyn/sdep.html>**, Acessado em: 11 Agosto 2018.

SHIMABUKURO, M. H. **Visualizações Temporais em uma plataforma de *software* extensível e adaptável**, Milton Hirokazu Simabukuro, 2004.

SILVA, R. O. da. **Teorias da administração**, São Paulo: Pearson Prentice Hall, 2008.

SOMMERVILLE, I. **Engenharia de *Software***; tradução Ivan Bosnic e Kalinka G. de O. Gonçalves; revisão técnica Kechi Hirama, 9ª Ed., São Paulo : Pearson Prentice Hall, 2011.

TANENBAUM, A. S. **Redes de Computadores**, 4ª Ed., Editora Campus (Elsevier), 2003./

TURBAN, E.; MCLEAN, E.; WETHERBE, J. **Tecnologia da informação para gestão.** Tradução de Renate Schinke, 3 ª Ed., Porto Alegre: Bookman, 2004.

WIEGERS, K.E. ***Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle.*** *2nd Edition, Microsoft Press, Redmond, Washington*, 2003.

WOLFRAM, *Wolfram Alpha Blog Archive*, 2009. Disponível em: <**http://blog.wolframalpha.com/2009/05/15/wolframalpha-is-launching-made-possible-by-mathematica/>,** Acessado em: 22 de Setembro de 2018.

ZEMEL, *T. MVC (Model – View – Controller).* 2009. Disponível em**: <http://www.codeigniterbrasil.com/passos-iniciais/mvc-model-view-controller/**>. Acessado em: 10 de Julho de 2018.