

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

THAYNARA RODRIGUES SILVA

**PROTÓTIPO DE APLICAÇÃO *WEB* PARA GESTÃO DE ESTOQUE DE UM
LABORATÓRIO DE PESQUISA DA UNIVERSIDADE FEDERAL DE
UBERLÂNDIA**

Uberlândia,

2021

THAYNARA RODRIGUES SILVA

**PROTÓTIPO DE APLICAÇÃO *WEB* PARA GESTÃO DE ESTOQUE DE UM
LABORATÓRIO DE PESQUISA DA UNIVERSIDADE FEDERAL DE
UBERLÂNDIA**

Trabalho de Conclusão de Curso
apresentado à Faculdade de Engenharia
Elétrica da Universidade Federal de
Uberlândia como requisito parcial para
obtenção do título de bacharel em
Engenharia Biomédica.

Orientadora: Profa. Dra. Ana Cláudia
Patrocínio

Uberlândia,

2021

Dedico este trabalho à minha família.

AGRADECIMENTOS

Agradeço à minha família, principalmente à minha mãe por batalhar muito para me oferecer essa oportunidade.

À minha irmã Marcela por me acompanhar nessa jornada.

À Prof. Ana Claudia pela paciência.

Aos meus amigos de graduação, Barbara, Bruna, Danillo, Ítalo, Luísa e Mirella, obrigada por terem tornando os anos de graduação mais leves.

À Leticia, por ser minha melhor amiga e me apoiar desde sempre.

À Marianny, por me acompanhar desde o ensino médio.

E em especial, à Ana Flávia pela amizade, e por ter dividido seus conhecimentos comigo.

“Comece fazendo o que é necessário, depois
o que é possível, e de repente você estará
fazendo o impossível.”

(São Francisco de Assis)

RESUMO

A otimização do tempo sempre foi um processo essencial para a gestão de prazos em empresas, laboratórios e na vida das pessoas. Para tal, é necessário analisar como as atividades cotidianas são realizadas e elaborar métodos que aperfeiçoem a forma como o tempo é utilizado. O presente trabalho propôs o desenvolvimento de um *Software web* para localização dos itens que compõem o estoque de um laboratório de pesquisa da Universidade Federal de Uberlândia. Assim, o tempo que seria utilizado na busca de materiais, poderá ser utilizado em outros processos, resultando em uma maior produtividade da equipe que compõe o laboratório. Para o desenvolvimento da plataforma proposta foram utilizados os *framework* DjangoRest e Vue.js. O banco de dados foi obtido a partir do levantamento de todos os itens disponíveis no laboratório e organizados em forma de tabela. O protótipo foi eficiente na busca dos itens e para retornar informações de localização e armazenamento para o usuário. Embora seja necessário um aprimoramento do *software*, o protótipo já foi capaz de contribuir para a otimização do tempo de busca de um item.

Palavras-chave: Gestão de estoque; Laboratório de Pesquisa; *Software*.

ABSTRACT

Time optimization has always been an essential process for managing deadlines in companies, laboratories and in people's lives. To do this, it is necessary to analyze how daily activities are carried out and devise methods that improve the way time is used. The present work proposed the development of a *Software* to locate the items that make up the stock of a research laboratory at the Federal University of Uberlândia. Thus, the time that would be used in the search for materials, can be used in other processes, resulting in greater productivity of the team that makes up the laboratory. DjangoRest and Vue.js. *framework* were used to develop the proposed platform. The database was obtained from the survey of all items available in the laboratory and organized in a table. The prototype was efficient in the search for the items and in returning the location and storage information to the user. Although it is necessary to improve the *software*, the prototype has already been able to contribute to the optimization of the search time for an item.

Keywords: Inventory management. *Software*. Laboratory

LISTA DE ILUSTRAÇÕES

Figura 1 - Arquitetura básica de um <i>website</i>	16
Figura 2 - Arquitetura de um <i>website</i> dinâmico.	17
Figura 3 - Representação Server-side e Client-side.	17
Figura 4 - Diagrama de caso de uso de um sistema de informação de pacientes.....	22
Figura 5 - Prototipação de Pressmann.....	24
Figura 6 - Diagrama de caso de uso do sistema desenvolvido.....	27
Figura 7 - Tela inicial apresentada ao usuário.....	29
Figura 8 - Busca de item através da barra de pesquisa.....	30
Figura 9 - Detalhes de um reagente selecionado pelo usuário	30
Figura 10 - Ícones de exclusão, edição e visualização, respectivamente.	31
Figura 11 - Página de edição	31
Figura 12 - Ícone de inserção de novo reagente	32
Figura 13 - Inserção de dados de novo reagente	32

LISTA DE ABREVIATURAS E SIGLAS

BDR	Banco de Dados Relacional
CRUD	<i>Create, Read, Update e Delete</i>
CSS	Cascading Style Sheet
HTML	HyperText Markup Language
NANOBIO	Laboratório de Nanobiotecnologia
REST	Representational State Transfer
SGBDR	Sistemas Gerenciadores de Bancos de Dados Relacional
UFU	Universidade Federal de Uberlândia

SUMÁRIO

1	INTRODUÇÃO.....	12
2	OBJETIVO GERAL.....	14
2.1	Objetivos específicos	14
3	FUNDAMENTAÇÃO TEÓRICA	15
3.1	Gestão de estoques	15
3.2	Aplicação <i>web</i>	16
3.2.1	<i>Frontend</i>	Erro! Indicador não definido.
3.2.2	<i>Backend</i>	Erro! Indicador não definido.
3.3	Banco de dados relacional.....	19
3.3.1	CRUD – Create, Read, Upate and Delete	20
3.4	Desenvolvimento de <i>software</i>	20
3.4.1	Levantamento e análise de requisitos	20
3.4.2	Diagrama de casos de uso	21
3.4.3	Prototipação	22
3.4.4	Framework.....	25
3.4.4.1	<i>Vue.js</i>	25
3.4.4.2	<i>Django</i>	26
4	METODOLOGIA.....	26
4.1	Visão geral do protótipo	26
4.2	Levantamento de requisitos	26
4.3	Diagrama de casos de uso.....	27
4.4	Desenvolvimento do protótipo	28
5	RESULTADOS	29
6	DISCUSSÃO.....	33
7	CONCLUSÃO.....	36
8	REFERÊNCIAS.....	37

1 INTRODUÇÃO

Os sistemas informatizados são aqueles que utilizam *software* para tratar as informações de interesse. Os sistemas de informação computacionais se tornaram importantes ferramentas capazes de propagar a informação de maneira rápida, fácil e eficaz.

Além disso, são importantes ferramentas de apoio às decisões administrativas que necessitam de amplo conhecimento técnico-científico, isso faz com que tenham as mais diversas aplicações e sejam utilizados em momentos que haja necessidade de otimização de processos e melhora na qualidade de serviços específicos (BENITO; LICHESKI, 2019).

Nesse contexto, podemos trazer a ideia de utilização de *software* como auxiliares em etapas de gestão de empresas. Toda instituição, pública ou privada, precisa possuir um espaço destinado ao armazenamento dos materiais necessários para executar as atividades de seu cotidiano, esse espaço é denominado estoque. Os estoques são pilhas de matérias-primas, insumos, componentes, produtos em processo e produtos acabados. Também podemos definir estoque como os bens materiais que são guardados, estando inclusos os produtos acabados e as matérias-primas que aguardam utilização na produção. Dado o exposto, todas as organizações devem ter um gerenciamento de seus pertences (MARTELLI; DANDARO, 2015).

O termo gerenciamento de estoque refere-se ao planejamento de como controlar os materiais guardados, focando nas necessidades da instituição. Esse gerenciamento surgiu a partir da necessidade de controlar de forma mais eficaz os materiais estocados. Uma forma eficaz de fazer este controle é utilizando sistemas de informação. O controle que antes era feito manualmente através de fichas de papel, vem sendo substituído por sistemas informatizados que aprimoram o processo de controle (MARTELLI; DANDARO, 2015).

As organizações e instituições públicas devem prestar serviços à sociedade de forma eficiente e eficaz. Para que isso aconteça, os indivíduos que compõem essas organizações devem ter à sua disposição, ferramentas, materiais e suprimentos necessários para executar as atividades cotidianas de seu trabalho (OLIVEIRA et al, 2016).

O Laboratório de Nanobiotecnologia localizado no Bloco 2E - sala 248 do Campus Umuarama desenvolve diversas pesquisas nas áreas de Nanobiotecnologia, Genoma, Biotecnologia na Agropecuária, Dermatologia Sanitária, Hanseníase, Câncer, Imunologia Aplicada, dentre outras, incluindo pesquisas envolvendo o SARS-CoV-2, vírus causador da COVID-19. Devido à quantidade de atividades desenvolvidas, o laboratório possui diversos itens que precisam ficar armazenados de maneira correta e com fácil acesso para os

pesquisadores autorizados. Atualmente os dados de todos os itens do estoque estão armazenados em planilhas individuais para cada forma de armazenamento.

Apesar de conter os dados necessários, as planilhas mencionadas não são padronizadas, os dados não estão compilados em um único banco de dados e não oferecem um mecanismo de busca eficiente e intuitivo. Dessa forma, quando um usuário precisa usar um item que está localizado no laboratório mas não tem informações específicas, como temperatura de armazenamento, o tempo de procura é longo e ineficiente.

Dado o exposto, o objetivo deste trabalho é construir um protótipo de *software web* utilizando os princípios da engenharia de *software*, engenharia *web* e conceitos de usabilidade, para armazenar, tratar e expor de forma eficaz, os dados de estoque do laboratório de Nanobiotecnologia da Universidade Federal de Uberlândia, gerando economia do tempo de busca da localização dos itens e como consequência mais eficiência no trabalho da equipe do laboratório.

2 OBJETIVO GERAL

Desenvolver uma aplicação *web* que unifique as informações de estoque do Laboratório de Nanobiotecnologia e agilize a busca, indicando informações complementares sobre armazenamento do produto pesquisado.

2.1 Objetivos específicos

- Fazer levantamento de itens disponíveis no laboratório;
- Organizar todos os itens em tabelas de acordo com uso e armazenamento;
- Contruir o banco de dados dos insumos disponíveis;
- Desenvolver a camada de código responsável pela lógica de funcionamento, acesso e manipulação do banco de dados (*backend*) que seja capaz de realizar CRUD (*Create, Read, Update e Delete*) no banco de dados;
- Desenvolver a camada visual, ou de interação com o usuário, do sistema (*Frontend*) que mostre ao usuário as informações solicitadas;
- Permitir cadastro de novos itens;
- Retornar dados do item pesquisado: local de armazenamento, temperatura de acondicionamento, fabricante e profissional responsável pelo item;

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Gestão de estoques

O termo estoque representa a quantidade de bens físicos que são mantidos à espera da venda, da produção ou do uso por um determinado tempo. São considerados como bens em estoques as matérias-primas e os produtos acabados. Os estoques são os materiais que não estão sendo utilizados no momento, mas que serão usados em atividades futuras (LOPRETE, 2009).

A gestão de estoque consiste em um conjunto de ferramentas para auxiliar os gestores a organizar as informações de localizações, controlar as quantidades, e apoiar o manejo dos itens que o compõe. Podem ser definidos ainda como itens mantidos em disponibilidade para execução dos serviços da organização. O gerenciamento de estoques é importante para garantir o fornecimento dos materiais necessários, agilizando assim os processos e proporcionar economia de escala da organização (LOPRETE, 2009).

Para que a gestão de qualquer informação, incluindo as de estoque, gere bons resultados, é necessário que as ferramentas escolhidas sejam coerentes e possibilitem o fornecimento de informação relevante, com qualidade, no tempo correto, com um custo apropriado e facilidades de acesso por parte de quem está autorizado a utilizá-la (BRAGA, 2021).

Sob o mesmo ponto de vista, pode-se inferir que a gestão de estoques se tornou uma parte vital de qualquer cadeia de abastecimento. Sendo que, uma política de estoques eficiente não só regula os custos, mas também torna a instituição mais ágil (VEDOVATO *et al*, 2015).

A gestão de estoque muitas vezes é feita de forma manual com uso de anotações manuscritas, para posteriormente atualizar uma planilha dentro de um computador ou serem arquivados em um espaço de armazenamento físico. Apesar de cumprirem seu papel principal, proporcionar o registro e logística do estoque, tais atividades consomem tempo, geram desgaste para os colaboradores e, conseqüentemente, estão suscetíveis a falhas humanas (MARQUES, 2019).

Em virtude disso, a busca por soluções mais eficazes no gerenciamento de estoques faz-se necessária. O uso da tecnologia para auxiliar nas atividades de gerenciamento de estoque vem se tornando uma forma comum e eficiente de organização dentro de instituições públicas e privadas. O uso de aplicações voltadas para essas atividades aumentou com a popularização dos sistemas de gerenciamento e logística. Os *software* tornaram-se um dos principais recursos computacionais capazes de propagar de maneira rápida, fácil e segura as informações (BENITO, 2019).

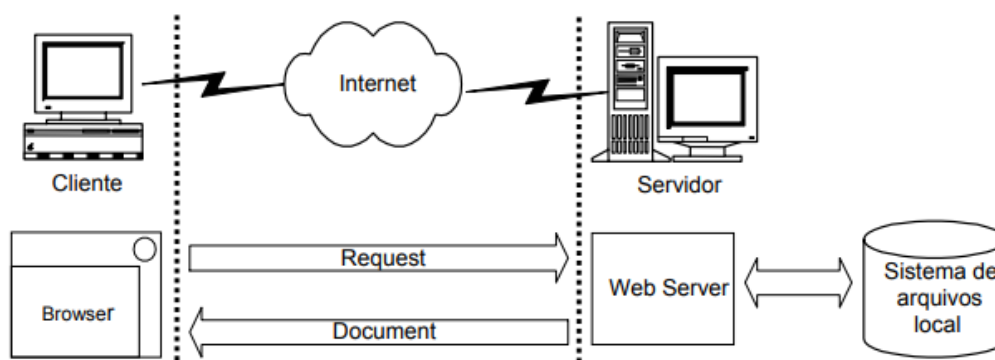
3.2 Aplicação web

Segundo Messenlehner e Coleman (2014), um site pode ser definido como um conjunto de uma ou mais páginas *web* contendo informações, podendo ser acessado através de um navegador *web*, por exemplo, sites corporativos, que exibem a história da empresa, apresentam os seus serviços e disponibilizam seu contato.

Uma aplicação *web* é uma aplicação de *software* que utiliza a *web* como ambiente de execução. Aplicações *web* envolvem sites *web* ou sistemas *web* (Conallem, 2000). De acordo com Beighley e Morrison (2010), uma aplicação *web* é um site dinâmico, elaborado para atender a um determinado propósito, podendo conter uma ou várias funcionalidades específicas para um público ou serviço prestado.

A figura 1 mostra a arquitetura básica de um site, onde um servidor *web* recebe uma solicitação de um *browser* (o navegador que utilizamos para acessar sites por meio da *internet*), localiza o documento em um sistema de arquivos local, e envia o documento de volta ao *browser*. Os recursos no sistema são interligados através de links, que são a forma usual de navegação no sistema, e podem ser documentos textuais, imagens, vídeo ou áudio (WINCKLER; PIMENTA, 2002).

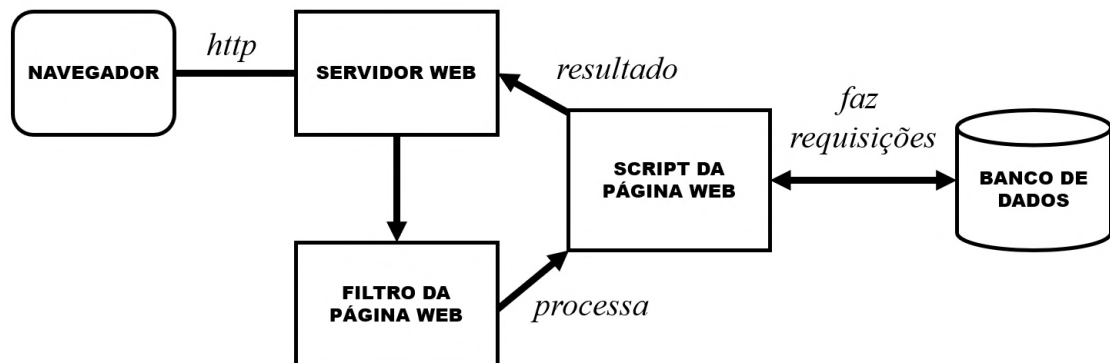
Figura 1 - Arquitetura básica de um *website*.



Fonte: Winckler e Pimenta, 2002.

Uma aplicação *web* amplia o conceito de *website*, ao adicionar funcionalidade ao sistema. Em outras palavras, aplicação *web* é um *website* dinâmico que permite aos usuários executarem lógica e usufruir de funcionalidades através do navegador, construindo e apresentando dinamicamente o conteúdo em função da interação do usuário através das páginas. A arquitetura básica é mostrada na figura 2.

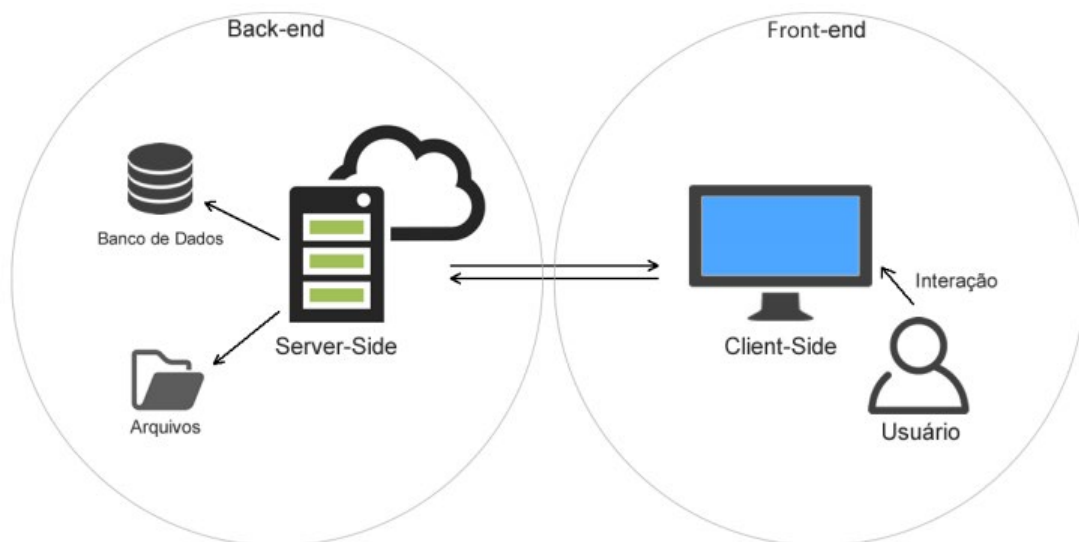
Figura 2 - Arquitetura de um *website* dinâmico.



Fonte: Adaptado de Winckler e Pimenta, 2002.

Uma aplicação *web* pode ser dividida em dois lados, o lado do cliente (*client-side*) e o lado do servidor (*server-side*), conforme ilustra a figura 3. O termo *frontend* e *backend* são usados para caracterizar as interfaces e serviços do programa de acordo com o lado em que estão servindo e serão explicados a seguir.

Figura 3 - Representação *Server-side* e *Client-side*.



Fonte: Bauer, 2018.

3.2.1 Frontend

Em aplicações *web*, a camada *frontend* é aquela com a qual os usuários interagem diretamente com a aplicação. O *frontend* da aplicação é executado no navegador (*client-side*) e pode encaminhar solicitações para a camada *backend*, localizada em outro computador ou em

um servidor local, para obter os dados solicitados ou executar um determinado serviço. (SOUTO, 2019).

O frontend é separado em 3 camadas básicas, a camada de informação (HTML), a camada de formatação (CSS), e a camada de comportamento (Javascript) (BAUER, 2018).

O HTML (Hyper Text Markup Language) é uma linguagem de marcação utilizada na construção de páginas na *web* que permite estabelecer a estrutura visual básica de uma página *web*. Portanto, ele serve para dar significado e organizar as informações de uma página na *web*. Essa estrutura possibilita o navegador exibir textos como elementos, carregar imagens e outros conteúdos (BAUER, 2018).

CSS (*Cascading Style Sheet*) é uma linguagem de estilos usada para estilizar elementos de uma página desenvolvida com uma linguagem de marcação, como o HTML. Com ela é possível separar a parte estrutural da parte visual e estilizar os itens definidos com o HTML, ou seja, o design gráfico dessa mesma página (BAUER, 2018).

O JavaScript (JS) é uma linguagem de programação utilizada principalmente em páginas *web*. Com o JS, é possível mudar dinamicamente a apresentação visual das páginas, conforme o comportamento que a aplicação necessite (BAUER, 2018).

Alguns exemplos de tecnologias que se destacam no desenvolvimento *frontend* são: HTML5, CSS3, Vue.js, Bootstrap, Materialize, Javascript, Angular, JQuery, dentre outros (BAUER, 2018).

3.2.2 Backend

Em projetos de *software*, para que a aplicação seja funcional, não basta apenas construir uma interface com as ferramentas de *frontend*. A solução precisa cumprir seu papel e colocar em prática as funcionalidades planejadas, como por exemplo realizar uma consulta no banco de dados e apresentar o resultado ao usuário. Para isso, é necessário uma camada que coloque em prática a lógica por trás da interface do usuário (SOUTO, 2019).

O *backend*, na maioria das vezes, atua como uma ponte entre os dados que são enviados do navegador para o banco de dados e vice-versa, aplicando as devidas regras de negócio, obedecendo o comando do usuário que solicitou determinada funcionalidade, realizando as validações e garantias em um ambiente que o usuário final não tem acesso, e, por fim, retornando essa informação de maneira visível para ele (SOUTO, 2019).

Um aplicativo, programa ou camada *backend* serve indiretamente como suporte aos serviços de *frontend*, residindo no servidor da aplicação *web* e podendo se comunicar com recursos de armazenamento (banco de dados) e controle de arquivos do sistema. Ele pode ser

escrito em diferentes linguagens de programação e em diferentes ambientes de desenvolvimento. PHP, Java, Django, Python, Node.js, Ruby e ASP são alguns exemplos de tecnologias utilizadas no desenvolvimento *backend* (BAUER, 2018).

3.3 Banco de dados relacional

Segundo Korth (1994), um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”, isto é, sempre que for possível agrupar informações que se co-relacionam e tratam de um mesmo assunto, podemos afirmar que se trata de um banco de dados.

Para Date (1991), é um sistema computadorizado cujo principal propósito é armazenar informações e permitir que os usuários busquem e atualizem essas informações quando quiser solicitá-las.

Banco de Dados Relacional (BDR) é um método de persistência de dados estável, seguro e rápido (COELHO; SARTORELLI, 2004). Os bancos de dados relacionais permitem aos seus usuários (incluindo desenvolvedores) escreverem consultas (*queries*) que não foram antecipadas por quem projetou a base de dados. Como resultado, bancos de dados relacionais podem ser utilizados por várias aplicações de infinitas maneiras diferentes. Isto tem tornado os bancos de dados relacionais muito populares em todas as esferas do desenvolvimento de *software* (COSTA, 2011).

Os Sistemas Gerenciadores de Banco de Dado Relacional (SGBDR) são usados em muitas aplicações, compondo uma gama inteira de *software* de computador. Os Sistemas Gerenciadores de Bancos de dados são os principais métodos de armazenamento e recuperação de dados para aplicações onde a coordenação entre muitos usuários, tabelas, e *queries* é necessária. Eles possuem um sistema robusto, com suporte a grande volume de dados, transações, controle de concorrência e otimizações; além disso, são amplamente utilizados no mundo todo. São exemplos de SGBDR: Mysql, MariaDB, PostgreSQL, SQL Server e Oracle (BATISTA, 2017).

O modelo relacional armazena as informações em tabelas unidimensionais, com linhas e colunas, onde cada linha da tabela representa um registro no banco de dados. É possível aos usuários criar, modificar e consultar dados dentro de um SGBDR através da linguagem SQL (Structured Query Language).

3.3.1 CRUD – Create, Read, Update and Delete

CRUD (*Create, Read, Update and Delete*) é um acrônimo para as operações de inserção, obtenção, atualização e exclusão de dados. Elas são as quatro operações básicas utilizadas em um banco de dados relacional e são necessárias para manipular qualquer dado persistente (AREND, 2011).

Em um sistema *web* que faça uso de banco de dados, as principais operações consistem em realizar CRUD. A utilização dessas quatro operações para a manipulação dos dados faz com que a automatização seja vantajosa para o ganho de velocidade no desenvolvimento e funcional para o usuário (PEREIRA; COGO; CHARAO, 2009).

3.4 Desenvolvimento de *software*

Para Sommerville (2011), *software* é caracterizado como um programa de computador e toda a documentação associada a ele. Os *software* podem ser de três tipos diferentes, todos eles oferecem o mesmo produto final, a informação. Porém, o que os diferencia é a forma com que o sistema obtém e entrega essas informações.

Os *software desktop* são aqueles que utilizam o sistema operacional para obter dados e fornecer informações ao usuário, já os *software web*, apresentados na seção anterior, utilizam os navegadores (*browsers*) para essa mesma função, e a *internet* para transmitir as informações.

Os *software mobile*, popularmente conhecidos como aplicativos, também utilizam os sistemas operacionais para realizar o fluxo de informações, porém, apenas em dispositivos *mobiles* (*smartphones e tablets*) (GONCALVES, 2016).

A seguir, serão apresentados os principais conceitos necessários para compreender o processo de desenvolvimento de um *software* para *web*.

3.4.1 Levantamento e análise de requisitos

Os requisitos são as descrições do que um *software* deve fazer, quais serviços disponibiliza e as restrições de seu funcionamento. O método de encontrar, avaliar e registrar esses serviços e restrições é conhecido como engenharia de requisitos (BUCKELL, 2017).

Para Sommerville (2011), podem ser definidos dois tipos de requisitos para um *software*: requisitos de sistema e requisitos de usuário. O primeiro descreve de forma detalhada e específica as funções, serviços e restrições do *software*, enquanto o segundo descreve de uma forma mais informal e geral quais os serviços e restrições o *software* fornecerá aos usuários. São também chamados de requisitos funcionais e não-funcionais, respectivamente.

Os requisitos funcionais descrevem o que este deverá fazer, ou seja, as funcionalidades do *software*, de forma geral ou específica, de acordo com o tipo de *software* a ser desenvolvido. É importante que, ao especificar os requisitos, todos os serviços e funções solicitados pelo cliente ou usuário final sejam definidos e essas definições sejam coerentes. Dessa forma, o *software* poderá ser entregue mais rápido, sem necessidade de rever o desenvolvimento por causa de requisitos mal definidos ou inconsistentes (SOMMERVILLE, 2011).

Requisitos não funcionais, são os requisitos que não estão diretamente associados aos serviços e funções específicos disponibilizados pelo *software* aos seus usuários. Existem diversos tipos de requisitos não-funcionais e eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, usabilidade, desempenho, segurança, interface, tempo de resposta e ocupação de área (SOMMERVILLE, 2011; GUEDES, 2011).

Essa distinção se torna útil para informar diferentes tipos de usuários e desenvolvedores dos diferentes tipos de detalhamento dos requisitos, que podem utilizá-los de diversas formas. Nessa etapa, as informações são obtidas com os *stakeholders* (clientes, usuários finais ou pessoas da organização que serão afetadas pelo *software*) sobre as funcionalidades que o *software* possuirá, as tarefas que ele irá ou não realizar.

As etapas que compõem o processo de levantamento de requisitos são: descoberta, classificação e organização, priorização e negociação, especificação. Para tal, geralmente são realizadas entrevistas com os *stakeholders* (interessados), que podem ser do tipo fechada, em que são definidas as perguntas da entrevista, e do tipo aberta, onde não são perguntas predefinidas, mas é feita uma análise com esses usuários para uma melhor compreensão de suas necessidades (SOMMERVILLE, 2011).

Depois do levantamento dos requisitos vem a fase da análise desses requisitos, para avaliar se todas as necessidades dos usuários foram entendidas corretamente e determinar se algo precisa ser melhorado (GUEDES, 2011).

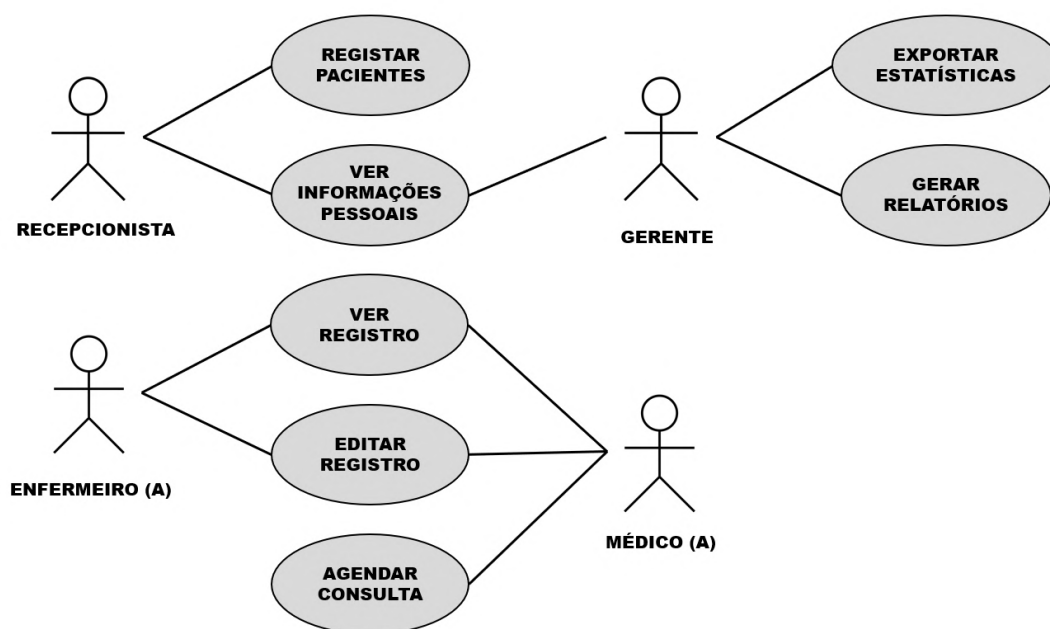
3.4.2 Diagrama de casos de uso

Segundo Guedes (2011), o diagrama de caso de uso é elaborado na etapa inicial da modelagem de um *software*. Ele pode ser consultado e possivelmente modificado durante o processo de levantamento de requisitos e serve como base para outros diagramas. O autor afirma que o diagrama de casos de uso é fundamental na identificação e compreensão dos requisitos, possibilitando especificar, visualizar, e documentar as características, funções e serviços do *software* que o usuário deseja.

Além disso, o diagrama também ajuda os desenvolvedores a pontuarem quais os tipos de usuário que irão utilizar o *software*, quais papéis esses usuários irão cumprir e quais funções um usuário específico poderá requisitar.

Sommerville (2011) pontua que o diagrama de caso de uso detalha as interações entre os usuários e o sistema. Além disso, de acordo com Gomes e Wanderley (2003), os diagramas de caso de uso são usados para demonstrar as funcionalidades que o sistema a ser criado irá prover, esse diagrama revela ainda a forma como atores (pessoa ou outro sistema) interagem com o sistema proposto. A figura 4 mostra um exemplo representativo de um diagrama de caso de uso.

Figura 4 - Diagrama de caso de uso de um sistema de informação de pacientes



Fonte: Adaptado de SOMMERVILLE (2011)

De acordo com Booch, Rumbaugh e Jacobson (2000, p. 217), o caso de uso exemplifica o comportamento desejado de um potencial usuário, mas não pontua como este será executado. Os relacionamentos ilustrados pelo diagrama ajudam a descrever casos de uso, e podem ser entre um ator e um caso de uso, entre atores ou entre casos de uso (SAMPAIO, 2007).

3.4.3 Prototipação

O rápido crescimento da indústria de desenvolvimento de *software* acarretou o surgimento de problemas mais complexos relacionados a correções, adaptações, aperfeiçoamento e manutenção. Para solucionar os novos desafios, é necessário alocar recursos,

pessoas e pensar em metodologias ágeis para a prototipação de soluções modernas e funcionais (PRESSMANN, 2006).

Um protótipo é uma das formas de se validar uma ideia, uma versão simulada ou amostra de um produto final, e tem como objetivo facilitar o entendimento dos requisitos de uma aplicação. Além disso, permite apresentar conceitos e funcionalidades do *software* de modo simplificado. A partir do protótipo, ajustes poderão ser feitos no projeto para adequar às expectativas e alinhar aos objetivos (PRESSMANN, 2006).

Tais esforços compõem o campo de atuação de uma área do conhecimento denominada Engenharia de *Software*. A Engenharia de *Software* é a criação e a utilização de princípios de engenharia com o objetivo de desenvolver *software* econômicos, confiáveis e que trabalhem eficientemente. O IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos) apresenta uma definição mais abrangente, que é a aplicação de uma abordagem sistemática, disciplinada e quantificável, para desenvolvimento, operação e manutenção de um *software* (PRESSMANN, 2006).

A Prototipação de Pressmann é um modelo de prototipação, uma das inúmeras metodologias da Engenharia de *Software*. Neste modelo, a construção do protótipo é de fundamental importância e vem após os primeiros esboços da interface do *software*. Ele possibilita experimentar diferentes ideias, identificar problemas e implementar melhorias no projeto que está sendo implementado. Além disso, a prototipação facilita a comunicação entre os desenvolvedores e os usuários da aplicação (PRESSMANN, 2006). A Figura 5 apresenta o modelo de Prototipação de Pressmann, que foi utilizado na prototipação da solução proposta nesse trabalho.

Figura 5 - Prototipação de Pressmann



Fonte: Adaptado de Pressmann (2011).

O paradigma da prototipação de Pressman auxilia os *stakeholders*, ou seja, os interessados (contratantes da solução), a entenderem melhor a forma como o *software* está sendo implantado. Como ilustrado na figura 5, o modelo se divide em cinco etapas:

- Comunicação: o desenvolvedor e o cliente encontram-se para definirem os objetivos gerais e fazer o levantamento e análise de requisitos do *software*;
- Plano rápido: nesta etapa inicia-se o planejamento do protótipo com base nos requisitos que foram definidos;
- Modelagem rápida: nesta fase é feita a modelagem do protótipo, os primeiros esboços da interface, botões, cores, etc.;
- Construção do Protótipo: a programação do *software* é feita nesta etapa, porém, nem todas as funcionalidades são implementadas, pois é apenas uma versão para testes e validação. Conforme o ciclo vai passando por esta etapa, as demais funcionalidades são implementadas, até a conclusão do projeto;
- Implantação, entrega e feedback: nesta etapa a versão do protótipo é entregue ao cliente para análise à espera de um feedback para ajustes, diversas versões podem ser entregues até a finalização da aplicação.

Como fora ilustrado, o modelo de prototipação é cíclico, ou seja, após a realização de todas as fases, o processo volta ao estágio inicial, podendo ser repetida para diferentes versões do projeto. Desta forma, torna-se possível implementar os requisitos mais importantes, e, no decorrer do projeto, os novos requisitos que forem identificados poderão ser implementados.

A medida que mais ciclos se completam, a solução vai se aperfeiçoando e então é testada pelo cliente para validar suas funcionalidades. O desenvolvedor avalia se pretende evoluir o protótipo e prossegue com o desenvolvimento para chegar à versão final, até resultar na entrega do *software* (PRESSMAN, 2006).

3.4.4 Framework

Inicialmente, as interfaces *web* nada mais eram do que documentos HTML, contendo texto e imagens interligados por links que permitiam navegar de um documento a outro. Estes recursos eram suficientes apenas para um número limitado de aplicações, a necessidade de utilizar a *web* para aplicações mais complexas como, por exemplo transações com banco de dados, motivou o desenvolvimento de novas tecnologias. Tecnologias para facilitar o desenvolvimento de aplicações *web* foram criadas como soluções ágeis para os novos desafios do mercado (WINCKLER; PIMENTA, 2002).

Para viabilizar a prototipação e se implementar um *software*, inicialmente o desenvolvedor deve escolher qual tipo de tecnologia se adequa melhor às necessidades do projeto e depois definir qual será a técnica de desenvolvimento e as linguagens de programação que serão utilizadas. Uma técnica interessante é a reutilização de trechos de código desenvolvidos por outros programadores, esse conjunto de códigos prontos que podem ser usados no desenvolvimento de aplicações é conhecido como *framework*. A literatura define um *framework* como um *software* parcialmente projetado. O *framework* define uma arquitetura geral para um conjunto de sistemas e oferece a estrutura básica para criá-los (JUNIOR, 2006).

Na Engenharia de *Software* é frequente o reuso. Através da reutilização de *software* é possível reduzir o esforço do desenvolvimento e consequentemente o aumento da qualidade do *software* (GIMENES; HUZITA, 2005).

Do ponto de vista da interface, cada tecnologia suporta um estilo de interação diferente e, potencialmente, pode incluir problemas de usabilidade diferentes. Cada *framework* oferece uma gama de benefícios ao desenvolvedor, a depender da aplicação que está sendo desenvolvida. Não é o objetivo aqui descrever exaustivamente todo o potencial de cada uma das tecnologias, mas segue abaixo uma apresentação dos dois principais *framework* que foram usados para o desenvolvimento da solução proposta neste trabalho.

3.4.4.1 Vue.js

Vue.js é um *framework* Javascript de código aberto, lançado por Evan You, desenvolvedor que atuava em um dos projetos do Google. Aplicações que utilizam Vue são

constituídas de componentes criados com a sintaxe HTML, CSS e Javascript em um único arquivo .vue, que facilitam o desenvolvimento *frontend*.

3.4.4.2 Django

O Django é um *framework* gratuito e de código aberto escrito em *Python*. A linguagem *Python* foi criada por Guido van Rossum em 1991, e é considerada uma linguagem de programação de alto nível, ou seja, se assemelha à estrutura de linguagem usada na comunicação humana. Essa linguagem possui estrutura definida como orientada a objetos e atualmente é uma das principais linguagens de programação usada. O Django é geralmente usado para desenvolvimento *backend* dentro das aplicações *web* (MILLMAN; AIVAZIS, 2011).

DjangoREST *Framework* é uma biblioteca que permite a construção de aplicações REST utilizando a estrutura do Django. Por funcionar sob a estrutura do Django, permite a construção de projetos em qualquer plataforma, seja Windows, macOS ou Linux.

4 METODOLOGIA

4.1 Visão geral do protótipo

A proposta do projeto é o desenvolvimento de uma aplicação *web*, destinado aos membros do NANOBIO, que poderá ser acessada pelos usuários através do navegador, dentro de um servidor, para localização dos itens que compõem o estoque do laboratório.

A metodologia escolhida para o desenvolvimento do protótipo, foi o modelo evolucionário de processo de *software*, denominado Prototipação de Pressman, descrito no item 2.4.3.

O desenvolvimento do protótipo foi dividido em três etapas principais: levantamento de requisitos; elaboração do diagrama de casos de uso; e a implementação do protótipo nas camadas *frontend* e *backend*.

O ambiente de desenvolvimento utilizado foi o Microsoft Visual studio, sendo que o *software* foi construído utilizando os *framework* vue.js e DjangoREST e o banco de dados SQLite.

4.2 Levantamento de requisitos

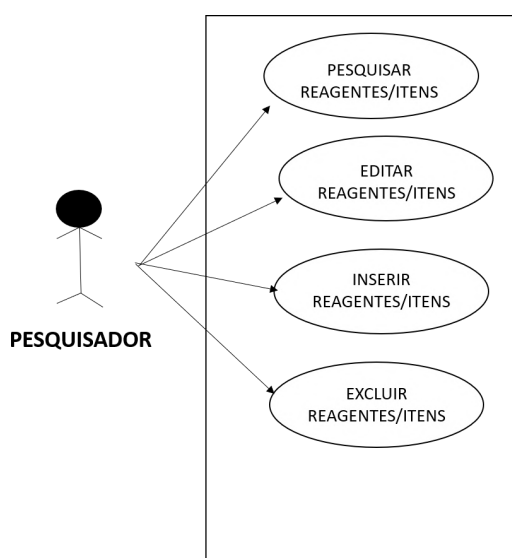
Para a coleta e refinamento dos requisitos, foram feitas reuniões com os futuros usuários, os membros do laboratório NANOBIO. Os requisitos levantados estão listados a seguir:

- Estabeleceu-se que é necessário que a aplicação permita a visualização de uma página *web*, contendo os dados dos itens em estoque no laboratório;
- O sistema deve ser capaz de pesquisar itens e mostrar um ficha de informação referente ao mesmo: temperaturas de armazenarmento, datas de preparo, data de validade e responsáveis,;
- O banco de dados deve ser iniciado com os itens atualmente presentes no laboratório;
- O sistema deve possibilitar ao usuário a exclusão e edição de itens;
- O local onde cada item está localizado deve ter destaque no sistema.

4.3 Diagrama de casos de uso

Após o levantamento e análise de requisitos, a aplicação foi planejada e modelada. O diagrama de caso de uso foi construído e pode ser visualizado na figura 6.

Figura 6 - Diagrama de caso de uso do sistema desenvolvido



Fonte: Autoria própria.

4.4 Desenvolvimento do protótipo

Após as etapas de comunicação (levantamento de requisitos), planejamento e modelagem (digrama de casos de uso), o protótipo foi desenvolvido.

Realizou-se o levantamento de todos os 2600 itens que compõem o estoque do laboratório. Depois do levantamento e análise das planilhas, os dados foram compilados em uma única tabela e posteriormente cadastrados no banco de dados do *software*, contendo todas as informações consideradas necessárias.

O *frontend* da aplicação foi desenvolvido com o *framework* Vue.js. O *Backend* foi construído utilizando o *framework* DjangoRest.

Após o desenvolvimento, o projeto prosseguiu para as etapas de implantação, entrega e *feedback*.

5 RESULTADOS

A aplicação *web*, denominada “Cadê – Encontre agora o que você precisa!” foi implantada e as imagens a seguir demonstram sua interface. Os *framework* DjangoRest e Vue.js permitiram criar um sistema de gestão de estoque onde o usuário visualiza os reagentes cadastrados e seus detalhes, busca o reagente/item na lista geral ou digitando o nome na barra de pesquisa, além de editar os itens já cadastrados e cadastrar itens.

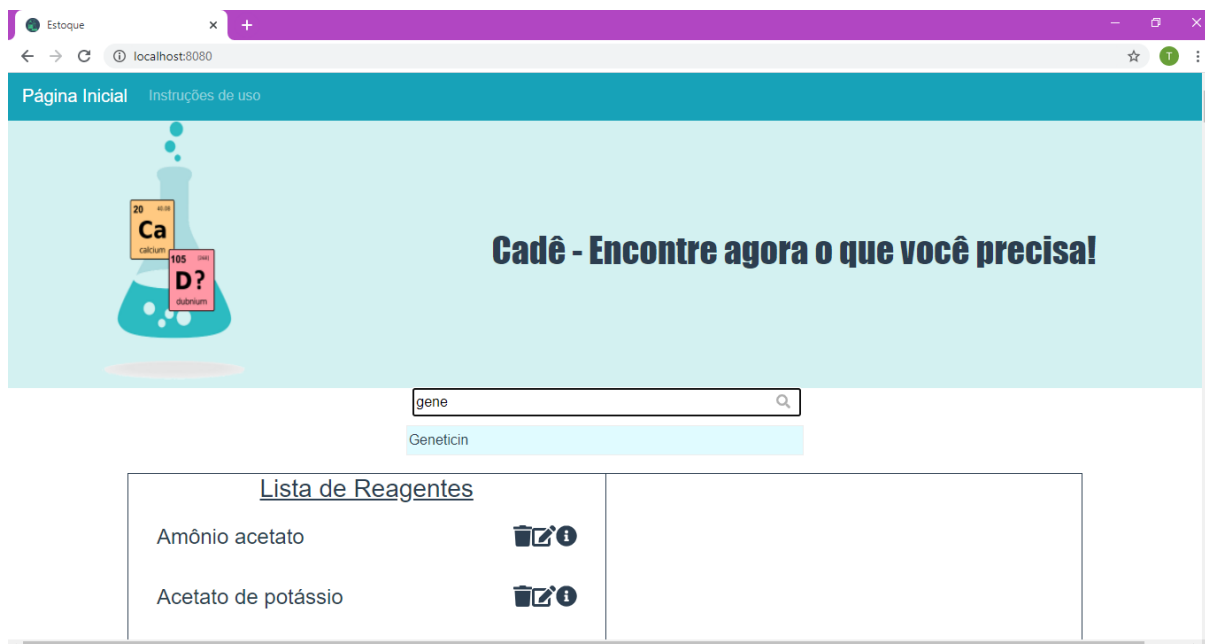
A página inicial conta com a listagem completa dos itens cadastrados e com uma barra de pesquisa, exibida na figura 7.

Figura 7 - Tela inicial apresentada ao usuário



Fonte: Autoria própria

O usuário pode buscar pelo item desejado digitando o nome do reagente na barra de pesquisa, e buscando diretamente na lista apresentada, como demonstra a figura 8. Ao digitar uma palavra na barra de pesquisa o sistema indica quais são os reagentes prováveis, através da busca por termos similares.

Figura 8 - Busca de item através da barra de pesquisa

Fonte: Autoria própria

Ao clicar sobre o nome do item desejado, é mostrado ao usuário um painel com as informações cadastradas, dando destaque ao local onde o item está armazenado. A figura 9 mostra o painel com os detalhes do item “Acetato de potássio”, após o usuário clicar sobre esse item na lista.

Figura 9 - Detalhes de um reagente selecionado pelo usuário

Fonte – Autoria própria

Ao clicar no ícone de edição o usuário pode alterar as informações do item, no ícone de exclusão ele consegue excluir o respectivo item, e no ícone de informações o usuário consegue ler a ficha informativa do item selecionado. A figura 6 mostra os referidos ícones.

Figura 10 - Ícones de exclusão, edição e visualização, respectivamente.



Fonte: Autoria própria

A imagem abaixo exibe a página de edição mostrada ao usuário após ele ter clicado sobre o ícone de edição, demonstrado na figura 10. Nele, é possível alterar os dados anteriormente cadastrados.

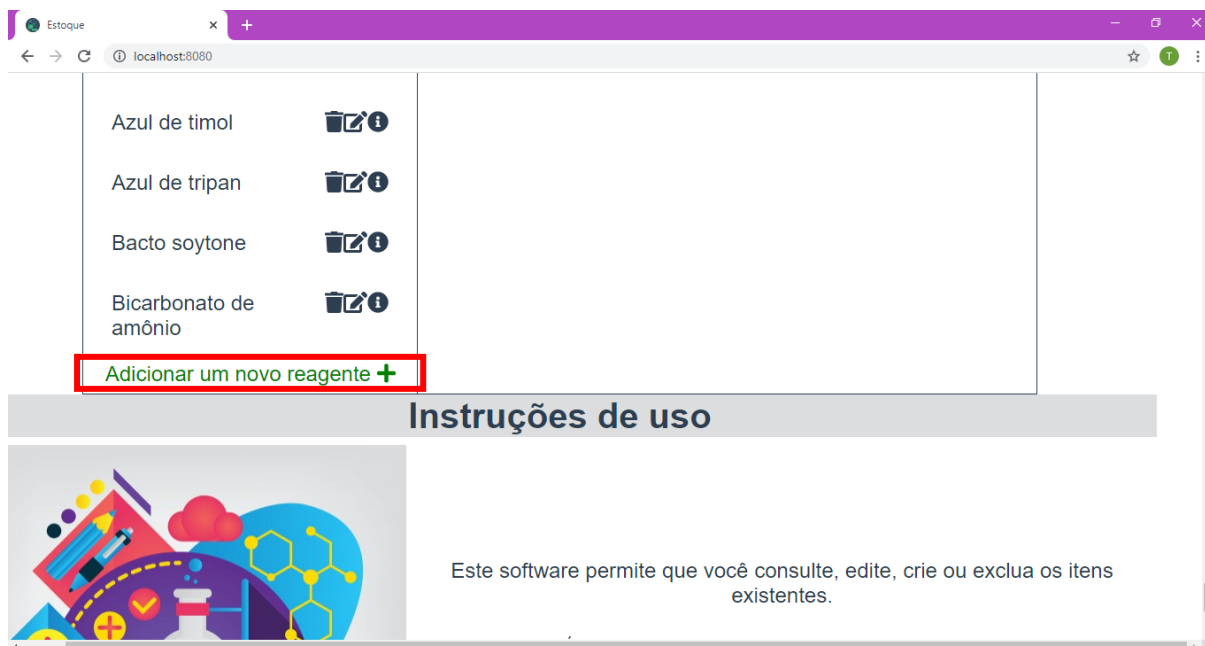
Figura 11 - Página de edição

Lista de Reagentes	Detalhes
Amônio acetato  	Name: Acetato de potássio
Acetato de potássio  	Local: ARMARIO DE SÓLIDOS EM ESTOQUE
Acetato de sódio anidro  	Temperatura: AMBIENTE
Acetato de sódio triidratado  	descricao: Marca: NÃO SE APLICA Data de Preparo: - Data de vencimento: -
Acido amino acético  	Numeração: 0
Ácido bórico  	Quantidade: 1
	Criado em: 12:28:00 - 04/12/2020
	Última modificação: 12:28:00 - 04/12/2020

Fonte: Autoria própria

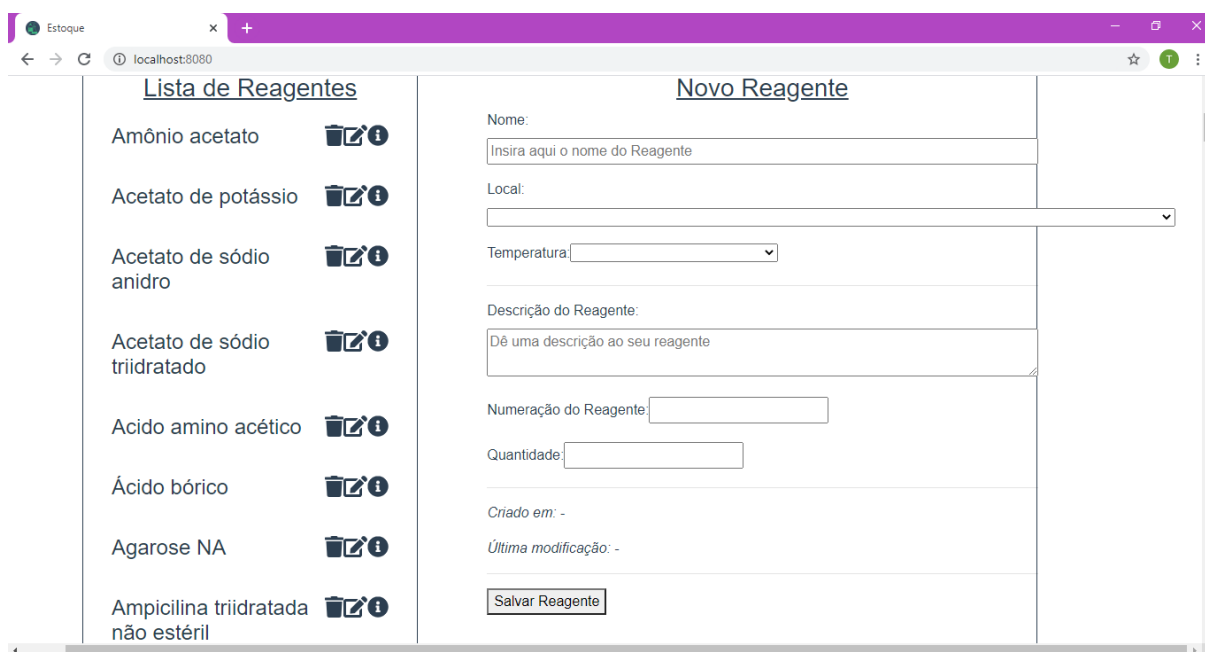
Para adicionar um novo item ao sistema o operador precisa clicar sobre o texto destacado “Adicionar um novo reagente” seguido de um ícone, que podem ser visualizados abaixo, nas figuras 12 e 13.

Figura 12 - Ícone de inserção de novo reagente



Fonte: Autoria própria

Figura 13 - Inserção de dados de novo reagente



Fonte: Autoria própria

Após clicar no ícone “Adicionar um novo reagente”, é retornada a página para preenchimento dos dados do novo item a ser incluído, como mostra a figura 13. O usuário deve digitar no respectivo campo o nome do reagente, indicar o local em que será guardado, qual a

temperatura de armazenamento e, no campo “descrição”, descrever informações adicionais como: responsável.; data de preparo e de vencimento. Essas informações compõem a ficha informativa do item, que é exibida toda vez que um reagente é consultado no sistema.

6 DISCUSSÃO

O paradigma da prototipação de Pressman, metodologia escolhida para implementação desta aplicação, mostrou-se eficaz no desenvolvimento do protótipo. A mesma técnica foi utilizada por Marcolin (2017) para a criação de uma aplicação *web* envolvendo banco de dados. Com a utilização do ciclo de vida, foi possível identificar os pontos a serem melhorados na interface e nas funcionalidades do *software* ao longo de seu desenvolvimento, tais como a inserção de um campo explicativo sobre o uso do sistema. Os testes de funcionamento foram realizados com o intuito de corrigir possíveis falhas e garantir a qualidade do protótipo e, também, para se obter a consolidação da primeira versão e disponibilizá-la aos usuários.

O levantamento de requisitos e o diagrama de caso e uso elucidaram de maneira eficiente quais eram as reais necessidades do sistema e direcionaram a escolha das ferramentas a serem usadas no desenvolvimento do sistema. A escolha de se utilizar um *framework* DjangoRest permitiu, como o esperado, o preenchimento de todos os dados iniciais de forma rápida e também realizar o CRUD (*Create, Read, Update e Delete*). Em seu trabalho, Pereira, Cogo e Charao (2009) também propuseram a construção de um sistema de controle de estoque utilizando o *framework* Django, onde foi possível construir *backend* contendo um banco de dados que possui todos os itens do estoque, realizar buscas e visualizar detalhes dos itens selecionados pelo usuário.

Em relação ao *frontend*, temos o modelo desenvolvido por Oliveira, Lima e Nunes (2020) como trabalho correlato, e, assim como no presente trabalho, foi desenvolvida uma área de interação com usuário (*frontend*) utilizando o *framework* Vue.js. Utilizando essa tecnologia,

foi possível construir interfaces intuitivas que enviem dados ao *backend*, para que seja feito o CRUD no banco de dados, e recebam informações para que sejam mostradas ao usuário. Em ambos os sistemas, o Vue.js se mostrou uma ferramenta eficaz e com estrutura bem-organizada.

O sistema desenvolvido apresenta os resultados de busca de reagentes, consulta os detalhes deles e otimiza o tempo de busca, antes feito manualmente no *software* Microsoft Excel®. Os 2600 itens cadastrados no banco de dados da aplicação web “Cadê – Encontre agora o que você precisa” foram coletados após a análise de 18 planilhas pré-existentes no laboratório. O NANOBIO é responsável por diversas pesquisas em todos os níveis acadêmicos e conta com dezenas de pesquisadores, alunos de iniciação científica, mestrado e doutorado. Ele utiliza reagentes diariamente e o gerenciamento deles é imprescindível para o andamento dos trabalhos que são realizados. A utilização de planilhas do Microsoft Excel® para o controle de estoques é realidade em muitas instituições e empresas e era a única ferramenta de gestão de estoques do laboratório. A opção pelas planilhas normalmente ocorre porque elas são muito fáceis de serem manipuladas e gratuitas, já que a maioria das instituições públicas, e grande parte das pessoas, possui o *software* Microsoft Excel® instalado em suas máquinas. Além disso, o hábito de utilização dessas ferramentas deixa muitos usuários com dúvida no momento de trocar de solução e optar por um sistema de gestão.

Entretanto, a segurança das planilhas e das informações contidas nela é baixa, pois qualquer problema no computador pode danificar seu arquivo e seus dados. Se a planilha for compartilhada com outras pessoas, os riscos aumentam ainda mais, com chances de alteração e desatualização das informações. Além disso, apenas um usuário pode utilizar a planilha por vez, aumentando os riscos de criar informações desatualizadas e duplicadas.

Diante da tendência de incorporação tecnológica em diversos setores e áreas do conhecimento, é importante que os profissionais de um laboratório de pesquisa desenvolvam habilidades para implementar, utilizar e monitorar *software* em seus processos de trabalho.

Nesse sentido, ressalta-se a importância da aplicação implementada. Quando o laboratório utiliza um *software*, todas as suas informações são criptografadas. O banco de dados fornece segurança e os dados ficam disponibilizados em tempo real. Além disso, o *software* foi projetado para ser utilizado por equipes, por isso, o armazenamento permite que diversos usuários tenham acesso às informações atualizadas em tempo real.

O *software* “Fácil estoque de código de barras e inventário”, desenvolvido pela (NOWI APPS, 2019), é uma aplicação de gestão de estoques que apresenta apenas o cadastramento via código de barras, não possibilitando a inserção de informações extras e características dos itens cadastrados. A solução proposta neste trabalho destaca-se nesse quesito, pois permite o cadastro

da ficha informativa do item cadastrado, facilitando o armazenamento de informações importantes sobre determinado reagente, como local em que será guardado, qual a temperatura de armazenamento, pessoa responsável, data de preparo e data de vencimento.

Essas lacunas para preenchimento de informações específicas para reagentes, são difíceis de serem encontradas em *software* de gerenciamento de estoque, pois são muito específicas e não fazem parte do escopo da maioria dos sistemas que são comercializados. Sendo assim, a criação de um *software* personalizado para essa finalidade, como feito nesse trabalho, é vantajosa, descomplicada e ágil para o laboratório e seus usuários.

No tocante a interface do *software*, podemos afirmar que ela é uma das principais etapas no processo de desenvolvimento de um produto, e deve ser de fácil e agradável utilização, garantindo a eficiência (OLIVEIRA; BARROS; OLIVEIRA, 2010). De acordo com Ferreira e Leite (2003), a falta de *software* bem projetados do ponto de vista de sua interface tem sido responsável por diversas perdas, e um dos motivos para tal desfecho é que a usabilidade deve ser levada em consideração. Dessa forma, uma possibilidade de melhoria futura para o *software* aqui apresentado é a aplicação de testes de usabilidade a fim de aprimorar a interface criada para que seja mais intuitiva e eficiente.

A adição de mais funcionalidades, como a separação de itens por categorias, opções avançadas de autenticação de usuário, realização automática de backup dos dados, hospedagem do *software* em um domínio *web* (permitindo acesso dentro e fora do laboratório) e o desenvolvimento de uma versão *mobile* da aplicação, fazem parte do escopo de melhorias futuras para a este *software*. Acredita-se que a adoção de tais recursos, associados à realização de testes contínuos para a melhoria do funcionamento do protótipo, resultará em uma excelente solução, podendo futuramente ser patenteada e comercializada.

7 CONCLUSÃO

O propósito deste trabalho foi desenvolver uma aplicação *web* para gestão do estoque do Laboratório de Nanobiotecnologia da Universidade Federal de Uberlândia que agilize a busca de itens do laboratório e forneça informações complementares sobre o produto pesquisado.

O levantamento de itens disponíveis no laboratório foi feito através de análise de planilhas pré-existentes. Após isso, todos os itens foram organizados em tabela única de acordo com uso e armazenamento e o banco de dados dos insumos disponíveis foi implementado com êxito.

O protótipo foi desenvolvido e através dele é possível realizar CRUD no banco de dados. A interface implementada permite que o usuário pesquise, cadastre, edite, apague e visualize os dados armazenados, contendo todas as informações relevantes do item cadastrado.

A construção do protótipo apresentado abre novos caminhos para novas soluções de *software* para gerenciamento de diversos tipos de estoque, baseado na experiência do usuário e personalizado de acordo suas necessidades, de forma que os *stakeholders* e desenvolvedores possam compreender o que será produzido. Ele também traz um conjunto de experiências aprendidas durante o desenvolvimento de sua arquitetura, baseada na melhoria contínua.

Com uma gestão de estoques eficiente é possível controlar, analisar e organizar produtos de maneira que possa garantir sua qualidade e integridade física, otimizando o trabalho das pessoas. Como trabalhos futuros, sugere-se primordialmente a continuidade da implementação de novas funcionalidades e melhorias no *software*.

8 REFERÊNCIAS

AREND, Felipe Gabriel. **Geração de Operações CRUD a partir de Metadados**. 2011. 72 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Tecnologia, Universidade Federal de Santa Maria, Santa Maria, 2011. Disponível em: <<http://wwwapp.inf.ufsm.br/bdtg/arquivo.php?id=153&download=1>>. Acesso em: 29 mai. 2021.

BATISTA, Emerson de Oliveira. **Sistemas de informação: o uso consciente da tecnologia para o gerenciamento**. 2017. São Paulo: Saraiva.

BAUER, Edinei Jacob. **Geração de interfaces de usuário para operações CRUD com base em metadados**. 2018. TCC (Graduação) – Curso de Ciência da Computação. Universidade do Extremo Sul Catarinense. Criciúma. 2018. Disponível em: <<http://repositorio.unesc.net/bitstream/1/8143/1/EDINEI%20JACOB%20BAUER.pdf>>. Acesso em: 01 jun. 2021.

BEIGHLEY, Lynn; MORRISON, Michael. **Use a cabeça! PHP & MySQL**. 2ª ed., Rio de Janeiro: Alta Books Editora, 2010.

BENITO, Gladys Amélia Véles; LICHESKI, Ana Paula. **Sistemas de Informação apoiando a gestão do trabalho em saúde**. Revista Brasileira de Enfermagem [S. l.], p. 447-450, 15 abr. 2019.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2ª ed., Rio de Janeiro, 2000.

BRAGA, A. "A gestão da informação". **Millenium Internet**, Viseu, n.19, jun. 2000. Disponível em: <https://repositorio.ipv.pt/handle/10400.19/903>. Acesso em: 27 de fev. 2021.

BUCKELL, Talissa Regina Occhi. **Protótipo de uma aplicação para a Web: Semana Nacional de Ciência e Tecnologia do IFSC Câmpus Florianópolis**. 2017. Disponível em: <<https://repositorio.ifsc.edu.br/bitstream/handle/123456789/513/TCC%20Talissa%20Buckell.pdf?sequence=1>> Acesso em: 01 jun. 2021.

COELHO, Camila Arnellas; SARTORELLI, Reinaldo Coelho. **Persistência de Objetos via Mapeamento Objeto-Relacional**. 2004. 110 f. TCC (Graduação) - Curso de Sistemas de Informação, Computação e Informática, Universidade Presbiteriana Mackenzie, São Paulo,

2004. Cap. 1. Disponível em: <<http://www.dominiopublico.gov.br/download/texto/ea000253.pdf>>. Acesso em: 01 jun. 2021.

CONALLEM, J.: **Building *Web* Applications with UML**. Addison Wesley, 2000.

COSTA, Elisângela Rocha da. **Bancos de Dados Relacionais**. 2011. TCC (Tecnólogo) – Curso de Tecnologia em Processamento de Dados. Faculdade de Tecnologia de São Paulo. São Paulo. 2011. Disponível em: <<http://www.fatecsp.br/dti/tcc/tcc0025>>. Acesso em: 01 jun. 2021.

DATE, C.J.; Int. a **Sistemas de Bancos de Dados**, tradução da 4a.edição norte-americana, Editora Campus, 1991.

FERREIRA, Simone Bacellar Leal; LEITE, Júlio Cesar Sampaio do Prado. **Avaliação da usabilidade em sistemas de informação: o caso do Sistema Submarino**. Rev. adm. contemp., Curitiba, v. 7, n. 2, p. 115-136, jun. 2003.

GIMENES, I.M.S., HUZITA, E.H.M. **Desenvolvimento Baseado em Componentes: Conceitos e Técnicas**. Rio de Janeiro: Ciência Moderna, 2005.

GOMES A. S.; WANDERLEY E. G. **Elicitando requisitos em projetos de *Software Educativo***, WIE'2003, Campinas (SP);

GONCALVES, Rodrigo Franco et al. **Uma abordagem sistêmica do processo de produção em engenharia *web*, na fase de concepção**. Prod., São Paulo, v. 26, n. 2, p. 402-416, jun. 2016.

GUEDES, Gilleanes T. A. **UML 2: uma abordagem prática**. 2ª ed., São Paulo: Novatec Editora, 2011.

JUNIOR, Celso Gomes Barreto. **Agregando *framework* de Infraestrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente AulaNet**. 2006. Dissertação de Mestrado (Mestrado em Informática) - Programa de Pós-Graduação em Informática da PUC-Rio, Rio de Janeiro, 2006.

KORTH, H.F. e SILBERSCHATZ, A.; **Sistemas de Bancos de Dados**. Makron Books, 2a. edição revisada, 1994.

LOPRETE, D. et. al. **Gestão de Estoque e a Importância da Curva ABC**, Lins, SP: 2009. Monografia (Graduação em Administração) – Centro Universitário Salesiano Auxilium, Lins.

MARCOLIN, Marcos. **Protótipo de *software web* de apoio ao ensino presencial com foco na interface do usuário**. 2017. TCC (Graduação) - Curso Superior de Licenciatura em Informática. Francisco Beltrão. 2017. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/11431/1/FB_COLIN_2017_2_06.pdf>. Acesso em: 31 mai. 2021.

MARTELLI, Leandro Lopez; DANDARO, Fernando. **Planejamento e Controle de Estoque nas Organizações**. Revista Gestão Industrial, [S. l.], p. 170-185, 2015.

MENDONÇA, Ricardo Augusto Ribeiro de. **Levantamento de requisitos no desenvolvimento ágil de software**. Coordenação de Pós-Graduação Lato Sensu Pontifícia Universidade Católica de Goiás (PUC Goiás), Goiânia – GO – Brasil. Acesso em: 16 dez. 2020.

MESSENLEHNER, Brian; COLEMAN, Jason. **Criando aplicações web com Wordpress**. São Paulo: Novatec Editora, 2014.

MILLMAN, K. J; AIVAZIS, M. "Python for Scientists and Engineers," in **Computing in Science & Engineering**, vol. 13, no. 2, pp. 9-12, March-April 2011, doi: 10.1109/MCSE.2011.36.

NOWI APPS. **Fácil - estoque de código de barras e inventário**, 2019. Disponível em: <https://play.google.com/store/apps/details?id=cz.nowi.inventory&hl=pt_BR>. Acesso em: 31 mai. 2021.

OLIVEIRA, Arthur Melo de; GADELHA, Hugo Rodrigues de Oliveira; NUNES, Denílson Ricardo de Lucena; MELO, André Cristiano Silva; BRANCO, Antônio do Nascimento. **Software para Gestão de Estoque em uma Instituição Pública de Ensino Superior**. Revista Gestão Industrial, [S. l.], p. 19-47, 2016.

OLIVEIRA, Cleiane Gonçalves; BARROS, Kátia Adriana Alves Leite de; OLIVEIRA, Ariane Gonçalves de. **Construção de um protótipo de software para apoio à Sistematização da Assistência de Enfermagem, utilizando a engenharia de software e usabilidade**. Journal of Health Informatics, [S. l.], p. 1-6, 1 mar. 2010.

OLIVEIRA, Victor Felipe dos Reis; LIMA, Joselice Ferreira; NUNES, Danilo Magalhães. **Criação de webservice RESTful em Python para consumo através de aplicação em Vuejs**. Anais dos Simpósios de Informática do IFNMG - Campus Januária, [s. l.], 14 dez. 2020.

PEREIRA, Adriano; COGO, Vinícius Vielmo; CHARAO, Andrea Schwertner. *framework para Desenvolvimento Rápido de Aplicações Web: um Estudo de Caso com CakePHP e Django*. Programa de Educação Tutorial (PET), [S. l.], p. 1-6, 2009.

PRESSMAN, R.S. **Engenharia de Software**. 6ª Ed, McGraw-Hill, 2006.

PRESSMAN, Roger S. **Engenharia de Software: Uma abordagem profissional**. 7. ed. Tradução: Ariovaldo Griesi e Mario M. Fecchio. Porto Alegre: AMGH Editora, 2011.

SAMPAIO, Marcus Costa. **Casos de uso**. 2007. Disponível em: <<http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SIII/Uml/diagramas/usecase/s/usecases.htm>> . Acesso em: 01 jun. 2021.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª ed., São Paulo: Pearson Prentice Hall, 2011.

SOUTO, Mario. **O que é frontend e backend?**. ALURA. 2019. Disponível em: <<https://www.alura.com.br/artigos/o-que-e-frontend-e-backend>>. Acesso em: 31 mai. 2021.

VEDOVATO, Ademir Junior. **Utilização da Tecnologia de Informação na gestão de estoques em uma Empresa varejista**. 2015. Disponível em: <

http://www.fecilcam.br/anais/ix_eepa/data/uploads/2-logistica/2-02.pdf>. Acesso em: 31 mai. 2021.

VOIGT, Ricardo; JUNIOR, Osmar de Oliveira Braz. **Análise de desempenho de arquitetura SOAP e REST para comunicação entre sistemas**. Congresso Sul Brasileiro de Computação, [s. l.], 11 dez. 2020.

WINCKLER, Marcos; PIMENTA, Marcelo Soares; **Avaliação de Usabilidade de Sites *Web***. Institut De Recherche En Informatique De Toulouse. cap.3 p. 11-12 . 2002.