SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL
CENTRO DE EDUCAÇÃO PROFISSIONAL EURICO DE AGUIAR SALLES
CURSO **TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS**

AUTOR:
IURI MATTEDI TOMAZINI

CONTROLE DE INDICADORES, ANÁLISES E AÇÕES PARA GESTÃO EMPRESARIAL

LINHARES
2021

AUTOR:
IURI MATTEDI TOMAZINI

CONTROLE DE INDICADORES, ANÁLISES E AÇÕES PARA GESTÃO EMPRESARIAL

Trabalho de Conclusão de Curso apresentado ao Curso Técnico de Desenvolvimento de Sistemas, do Senai – ES, como requisito parcial para obtenção do título de Técnico em Desenvolvimento de Sistemas
Orientador: Bruno Cézar Manzoli Ferreira

LINHARES
2021

IURI MATTEDI TOMAZINI

WELERSON SOUZA DOS SANTOS

CONTROLE DE INDICADORES, ANÁLISES E AÇÕES PARA GESTÃO EMPRESARIAL

SENAI – SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

Este **Trabalho de Conclusão de Curso** foi julgado e aprovado pela banca examinadora para a obtenção do grau de **Técnico em Automação do Serviço Nacional de Aprendizagem Industrial - SENAI.**

Linhares, 20 de agosto de 2021.

_____
Prof. Bruno Cézar Manzoli Ferreira
SENAI – Serviço Nacional de Aprendizagem Industrial
Orientador

_____
Prof. Carlos Alexandre
SENAI – Serviço Nacional de Aprendizagem Industrial
Instrutor

_____
Prof. Rodrigo Niero
SENAI – Serviço Nacional de Aprendizagem Industrial
Coordenador Pedagógico

RESUMO

Agilidade, usabilidade e confiabilidade são características cada vez mais solicitadas em desenvolvimento de software de gestão e controle organizacionais. Neste contexto, este trabalho se baseia numa revisão bibliográfica de recursos de programação voltados a esses quesitos, a fim de realizar o desenvolvimento de um sistema *web* que centraliza e facilita o acesso a indicadores organizacionais, análises de causas de não-conformidades e ações corretivas ou preventivas, por meio de uma plataforma funcional, usual e segura. Através do sistema, a organização poderá cadastrar os resultados e consultá-los sempre que necessário, a fim de fomentar a gestão participativa e direcionar a gestão para o cumprimento de metas pautadas nos objetivos estratégicos da empresa.

Palavras-chave: Indicadores, Ações, Gestão, Sistema *Web*, Usabilidade, Gestão Participativa.

SUMÁRIO

1 INTRODUÇÃO

Dentro de um processo de gestão, traçam-se objetivos e estratégias, controla-se o andamento deles por meio de indicadores com metas bem definidas e faz-se análises de causas e ações quando esses indicadores estão fora da meta estipulada. Os indicadores têm como intuito medir o estado presente da organização e as análises de causas e ações servem para manter a empresa dentro das metas a fim de alcançar os objetivos estratégicos. É comum, entretanto, as empresas não terem uma boa sistemática de controle para essas informações.

Muitas empresas têm seus indicadores gerados por diferentes áreas, que acessam módulos restritos para obter dados e compilá-los. Esses indicadores são em geral muito específicos para cada organização, por isso os sistemas (softwares) de gestão comumente não possuem uma seção ou módulo destinado a controlar e centralizar essas informações. Fica à mercê, portanto, das áreas guardarem evidência e valores desses indicadores, muitas vezes, salvos em planilhas despadronizadas e dispersas em pastas de rede o que pode dificultar o acesso e comprometer a segurança da informação.

Além disso, é uma exigência comum de órgãos certificadores de gestão, como a ISO (International Organization for Standardization), que esses indicadores (ou KPIs, Key Performance Indicators), análises e ações sejam documentados sistematicamente e divulgados – para fomentar a gestão participativa – e que evidências sejam arquivadas para comprovação dos resultados. Daí, surge a necessidade de controlar essas informações e torná-las acessíveis, a fim de reduzir o risco de empresas certificadas terem não-conformidades registradas em auditorias e, principalmente, a fim de promover uma gestão eficaz.

Este trabalho, portanto, tem como objetivo levantar requisitos, planejar e desenvolver um aplicativo que centraliza e tornam acessíveis as informações de indicadores, análises de causas e ações, por meio de uma plataforma funcional, usual e segura disponibilizada na plataforma *web*.

1.1 OBJETIVO GERAL

Desenvolver um software para gerenciar – de forma centralizada e acessível – indicadores, análises de causas e ações, desdobradas de objetivos de organizações.

1.2 OBJETIVOS ESPECÍFICOS

Continuando com o exemplo anterior, os objetivos específicos poderiam ser:

- Reduzir o risco de organizações receberem não-conformidades em auditorias de órgão certificadores, por conta de um mau gerenciamento de dados de indicadores, análises de causas e ações.
- Prover uma plataforma que mantém evidências de resultados de empresas.
- Reduzir o tempo consumido com a atualização e a consulta de indicadores.
- Melhorar o acesso aos relatórios, fomentando a gestão participativa.

1.3 ASPECTOS METODOLÓGICOS

Segundo Gil (2002), pesquisa é um processo racional e sistemático para obter respostas para um problema específico. Este trabalho visa levantar requisitos e analisar a problemática referente ao controle de indicadores, análises de causas e ações atrelados a objetivos estratégicos de organizações.

Para classificação das pesquisas com base em seus objetivos, Gil (2002) propõe três tipos: exploratórias, descritivas e explicativas. A pesquisa desenvolvida neste trabalho se encaixa na exploratória, pois haverá um estudo da situação problema, a fim de propor uma melhoria embasada em conceitos e fundamentos de Desenvolvimento de Sistemas. Em relação à abordagem, este

trabalho é classificado como qualitativo, pois a situação explorada fornecerá bases não-numéricas para entendimento das causas do problema.

Já o levantamento de dados ocorrerá por meio de: entrevistas com profissionais de empresas que utilizam indicadores como base para gestão de seus processos; e pesquisa de campo, isto é, verificação da situação no local em que ela acontece. Os dados obtidos serão analisados e documentados como requisitos para o sistema a ser desenvolvido. Após isso, eles serão interpretados em diagramas de caso de uso, de classe e de sequência com o intuito de esquematizar os requisitos levantados para serem ratificados a nível de usuário e para servirem de base na composição das classes e interações do sistema.

Por fim, o aplicativo será desenvolvido para centralizar o controle de indicadores organizacionais, com o objetivo de facilitar o acesso e a atualização dessas informações.

2 REFERENCIAL TEÓRICO

2.1 USO DE INDICADORES EM ORGANIZAÇÕES

Todo processo de gestão empresarial, dentre vários aspectos, se baseia em mensurar resultados, traçar metas e tomar ações para que as metas sejam alcançadas. Dessa forma, as empresas bem geridas possuem indicadores com metas claras para serem atingidas participativamente a fim de garantir a sustentabilidade do negócio.

Francischini (2017) ressalta que indicadores devem estar atrelados às ações para que deem retorno à organização. Para exemplificar a importância dos indicadores como direcionadores de gestão, o autor ainda compara: “Dirigir uma empresa multinacional contando somente com as informações que o gestor consegue coletar por seus próprios sentidos equivale a um piloto de avião fazer um voo transcontinental somente olhando pela janela do cockpit”. O autor também menciona que indicadores trazem como benefício: o controle da

empresa, a comunicação de objetivos, a motivação de funcionários e o direcionamento de melhorias na empresa. Sendo assim, os indicadores, permitem que a direção de uma determinada organização abstraia informações que se alinham aos seus objetivos estratégicos.

Lemes (2018) corrobora que empresas devem aperfeiçoar continuamente sua gestão para terem competitividade no contexto de globalização atual, com foco nos resultados e na satisfação do cliente. Segundo ele, as empresas com esse tipo de gestão têm como características: atividades precedidas de planejamento com objetivos e metas; planejamento pautado com participação de representantes do processo; avaliação de recursos; definição e divulgação do processo padronizado (como fazer); e definição e compreensão da contribuição de cada um para os resultados e preservação da cultura organizacional. Por conta disso, é evidenciada a importância de a gestão possuir um planejamento ativo e com participação de todos da empresa.

2.2 LEVANTAMENTO DE REQUISITOS PARA DESENVOLVIMENTO DE SOFTWARES

A primeira etapa de desenvolvimento de um software consiste no levantamento dos requisitos que o conceberão. Nessa etapa são levantadas informações sobre sistema atual e como ele será transformado em um sistema de informação computadorizado que atenda as necessidades dos usuários.

De acordo com Guedes (2011, p.21-24), na etapa de levantamento de requisitos do software é alcançado o domínio do problema que determinará o que o software deve conter. Esse levantamento é uma função atribuída ao Engenheiro de Software que irá levantar as informações, principalmente, por meio de entrevistas. Essas entrevistas configuram a comunicação – principal desafio do levantamento de requisitos – a fim de sobrepassar as dificuldades para reunir informações, muitas vezes insuficientes e desornadas.

Ainda segundo Guedes, após levantar essas informações, é feita a análise dos requisitos. Nesta etapa, busca-se confirmar se as necessidades do usuário

foram compreendidas corretamente e verificar se há algo relevante que não foi considerado. Para estas análises, os pontos levantados são repassados aos clientes de forma simples (não técnica), normalmente com uso de diagramas e protótipos como ferramentas de apoio à comunicação.

2.3 DIAGRAMAÇÃO E DOCUMENTAÇÃO DE PROJETOS DE SOFTWARES

No contexto de desenvolvimento de softwares, os diagramas têm duas funções principais: apoiar o levantamento de requisitos; e projetar o desenvolvimento do software. Neste trabalho serão tratados dois diagramas: o de caso de uso e o de classes.

De acordo com Boosh (2006), o diagrama de caso de uso tem a função de representar as principais funcionalidades do sistema e como ocorrerá a interação com o usuário. Este diagrama não é muito detalhado e é feito para ser de fácil compreensão por parte do usuário. Os principais elementos do caso de uso são: Autores (bonecos), que são os usuários do sistema (pessoa, empresa, entre outros); Caso de Uso (elipse), que representa o a utilização do sistema; e Associações (linhas), as quais indicam as relações entre os demais elementos. Assim, este diagrama representa como cada autor se associa às formas de utilização do sistema.

O segundo diagrama, o de classes, serve para projetar as propriedades e métodos das classes do sistema e como elas irão interagir entre si. Guedes (2011) aponta as seguintes formas de relacionamento entre as classes: herança (seta), associação simples (linha), agregação (linha com losango vazio na ponta) e composição (linha com losango pintado na ponta). Além disso, este autor também especifica que a visibilidade de cada propriedade ou método deve ser representada no diagrama da seguinte forma: “-“ (privado), “+” (públic) e “#” (protegido). Dessa forma, é possível projetar cada característica importante das classes antes de implementá-las.

2.4 MODELAGEM DE BANCO DE DADOS

2.4.1 Modelos Conceitual, Lógico e Físico

Existem três modelos de banco de dados principais: o conceitual, o lógico e o físico. A diferença básica entre eles é o nível de abstração e, a seguir, tem-se uma base conceitual disposta por Barbosa e Freitas (2018).

O modelo conceitual é um modelo mais simplificado e que pode ser entendido mais facilmente pelo usuário. Ele é composto basicamente por blocos que representam as tabelas ou entidades, por losangos que representam os relacionamentos e por setas que demonstram as conexões entre as entidades e os relacionamentos. Existe também a possibilidade de utilização de círculos pequenos ligados às entidades para representar os atributos (colunas) das tabelas, mas sua utilização é flexível. Sendo assim, a demonstração do banco de dados por meio das figuras geométricas torna este modelo mais flexível e simples de ser entendido e faz com que sua utilização seja mais voltada para facilitar a comunicação com o requisitante do software.

O segundo modelo é o lógico. Ele é composto por quadrados que representam as entidades com a lista de todos os atributos contemplados no banco de dados. Cada atributo listado deve conter o tipo de dados (por exemplo: int, decimal, varchar, char e text) e se ele é uma chave primária (Primary Key – PK) ou uma chave estrangeira (Foreign Key – FK), além das cardinalidades entre as chaves das tabelas. Desta forma, este modelo tem um nível de detalhamento maior e é utilizado para desenvolvimento do terceiro modelo: o físico.

O modelo físico é a codificação do banco de dados em si. Existem diversas linguagens ou variações para desenvolvimento desde modelo, como: SQL, MySql e MariaDB. Por meio dessas linguagens, o banco de dados e as tabelas podem ser criados no servidor, bem como os relacionamentos entre as chaves e as *views*. Além disso, é possível fazer – dentre as várias ações no banco – as quatro operações principais abordadas pela sigla CRUD: Create (inserir), Read

(ler), Update (atualizar) e Delete (deletar). Neste trabalho, a linguagem de banco de dados utilizada será a MySql.

2.4.2 MySQL

O MySql é um Sistema Gerenciador de Banco de Dados (SGBD) *open source* e uma variação da linguagem SQL que possui um boa compatibilidade com PHP, linguagem para *back-end* de sistemas *web*.

O MySql pode ser administrado via codificação, concebida pelo SQL, para as diversas funções a serem executadas no banco de dados. Alguns comandos MySql disponíveis na documentação do site (MYSQL, 2021), são:

- **CREATE DATABASE**: Criação do banco de dados.
- **CREATE TABLE**: Criação de tabelas.
- **CREATE VIEW:** Criação de *views*.
- **SELECT * FROM**: Leitura de dados.
- **JOIN**: Junção de tabelas, por meio das chaves para leitura de dados em conjunto. Esse comando possui algumas variações, como LEFT JOIN, INNER JOIN, FULL JOIN, CROSS JOIN e SELF JOIN.
- **NOT IN**: Este comando serve para acessar os dados por meio de uma consulta em que um de seus campos não se encontra em outra consulta.

2.4.3 Normalização de Banco de Dados

A normalização de banco de dados engloba algumas regras e boas práticas para que o banco de dados cresça de maneira consistente, sem a duplicidade de dados e com eficiência. Existem três formas principais de normalidade, ou três etapas, conforme explanado por Barbosa e Freitas (2018):

- **1ª Forma Normal (1FN)**: Remoção de valores multivalorados ou compostos, ou seja, na 1FN todos os valores devem ser separados em

campos indivisíveis. Por exemplo: o endereço é composto por rua, CEP, número da casa, bairro e cidade, sendo assim subdivisão do endereço deve ser um campo individual na tabela, ao invés de se ter uma coluna única para todo o endereço. Além disso, a tabela deve conter uma chave primária.

- **2ª Forma Normal (2FN)**: Nesta forma normal, o banco deve atender também às regras da 1FN e cada atributo deve ser funcionalmente dependente da chave primária (PK). Se um campo possui uma dependência parcial da PK, é necessário extrair uma nova tabela desta relação. Por exemplo: uma tabela que contém informações de produtos e de clientes deve ser separada em duas, produtos e clientes, porque os campos de cada uma delas não são funcionalmente dependentes dos campos da outra tabela.
- **3ª Forma Normal (3FN)**: Um modelo atinge a 3FN quando está nas duas formas normais anteriores e quando as tabelas não possuem chaves relacionadas funcionalmente dependentes nas relações. Neste caso, as entidades devem ser unificadas. Além disso, as tabelas devem, por obrigatoriedade, ser relacionados entre as tabelas por meio das chaves FK.

Existe ainda um conjunto de regras que dá continuidade à normalização, conhecido como Boyce-Codd, mas não será abordado neste trabalho.

2.5 LINGUAGENS E RECURSOS PARA PROGRAMAÇÃO *WEB*

2.5.1 HTML

O HTML (*Hyper Text Markup Language*) é uma linguagem utilizada por todo o site ou sistema *web*. Ela é essencial e indispensável na programação web e se caracteriza como uma linguagem de marcação de hipertexto que organiza o conteúdo da aplicação web, isto é, forma o esqueleto do sistema.

De acordo com Miletto *el. al. (*2014), HTML é uma linguagem de marcação utilizada para definir como o conteúdo de uma página deve ser exibido. A exibição ocorre pelo navegador que interpreta o código que é delimitado por *tags* com todo o conteúdo apresentado entre elas. Dessa forma, o HTML define o que é apresentado pelo *site* ou sistema *web* e é a primeira camada de desenvolvimento.

Algumas *tags* comuns do HTML são:

- **<html>**: Engloba toda a codificação em HTML do site.
- **<head>**: parte que engloba informações gerais do site, como título e tipologia de caracteres (com acento, com símbolos especiais, etc.).
- **<body>**: Corpo.
- ** **: Uma divisão ou parte do site.
- **<section>**: Seção.
- **<form>**: Formulário.
- **<input>**: Entrada de dados.
- **<button>**: Botão.
- **<nav>**: Navegação.
- **<svg>**: Ícone.
- **<header>**: Cabeçalho.

2.5.2 CSS

Segundo Duckett (2016), o CSS (Cascading Style Sheets) é uma linguagem de design usada para estilizar páginas *web* no qual possui a capacidade alterar elementos HTML das mais diversas formas que vão desde controlar estilos de fontes, cores de textos e espaçamento entre parágrafos até a forma de exibição e responsividade em diferentes tamanhos de tela.

Por meio desta linguagem é possível atribuir características visuais para os diversos elementos HTML. Dentre estas características, tem-se:

- **background-color**: Cor de fundo do elemento do site.

- **height**: Altura do elemento.
- **width**: Largura do elemento.
- **max-width**: Largura máxima do elemento.
- **position:** Tipo de posicionamento do elemento na página de visualização. Os principais tipos são: o *absolute* (posicionamento relativo ao elemento onde ele está inserido), o *relative* (deslocamento com base em sua posição original no site) e o *fixed* (posição fixa na tela).
- **margin**: Margem do elemento (externa) em relação aos demais.
- **padding**: Distanciamento do conteúdo interno do elemento com relação à sua borda.
- **border:** Borda do elemento. Esta característica pode definir a largura da borda e sua cor.
- **top**, **right**, **letf** e **bottom**: Posicionamento superior, direito, esquerdo e inferior, respectivamente. São referenciados de acordo com a característica *position*.
- **color**: Cor da fonte.
- **z-index**: Valor inteiro que define quais elementos devem se sobrepor aos outros. Um elemento com o *z-index* superior deve se sobrepor aos demais.

A atribuição dessas características pode ocorrer por meio das classes, dos IDs ou pelas próprias *tags*.

2.5.3 Java Script

O JavaScript é uma das linguagem de programação mais comuns para o desenvolvimento da camada lógica do site. Conforme Rauschmayer (2021), o JavaScript foi criado em maio de 1995 por Brendan Eich.

O JavaScript é uma linguagem de programação de alto nível, interpretada, dinâmica e não tipada. Ainda segundo Rauschmayer (2021), por meio desta linguagem é possível realizar cálculos no *site*, remover e incluir elemento

HTML, alterar as propriedades do CSS e realizar algoritmos lógicos com estruturas de repetição, como o *for*, e de condição, como o *if* e o *switch case*.

2.5.4 PHP

Segundo Miletto *el. al. (*2014), o PHP (Hypertext Preprocessor) é uma linguagem de script utilizada no lado do servidor e que pode gerar informações HTML dinamicamente.

Nos sistemas web, é comumente necessário executar lógicas e comunicações com o banco de dados de forma inacessível para o usuário – isso mantém a segurança e integridade do site. Dessa forma, o PHP é uma linguagem que roda em *back-end*, ou seja, é executada no servidor da aplicação. Segundo Saraiva & Barreto (2018, p. 57) a linguagem PHP permite a criação de sistemas confiáveis e complexos, a fim de atender diferentes tarefas do desenvolvimento de sistemas.

Esta linguagem pode ser integrada com bancos de dados e alguns dos comandos que podem ser utilizados para esta finalidade são (PHP, 2021):

- **mysqli_connect(<host>, <user>, <password>)**: Conexão com o servidor do banco de dados.
- **mysqli_select_db(<conexão>, <banco de dados>)**: Seleção do banco de dados.
- **mysqli_query(<conexão>, <comando>) or die(mysqli_error (<conexão>))**: Execução do comando MySql – query.
- **mysqli_fetch_assoc(<query>)**: Obtenção dos dados da query. Normalmente utilizado dentro de uma estrutura de repetição, como o *while*.
- **mysqli_close(<conexão>)**: Fechar a conexão com o servidor.

Outros comandos que não estão diretamente relacionados à conexão com o banco, mas que podem ser úteis para tratamento dos dados obtidos são:

- **implode(<separador>, <array>)**: Separação dos itens de um *array* em uma única string.
- **array_push(<array>, <elemento>)**: Agregação de um elemento ou valor em um *array*.
- **json_encode(<query>)**: Organiza os dados para o formato de JSON.

2.5.5 Frameworks

2.5.5.1 JQuery

Segundo Silva (2013),

> "jQuery destina-se a adicionar interatividade e dinamismo às páginas web, proporcionando ao desenvolvedor funcionalidades necessárias à criação de scripts que visem a incrementar, de forma progressiva e não obstrutiva, a usabilidade, a acessibilidade e o design, enriquecendo a experiência do usuário."

Em outras palavras, trata-se de uma biblioteca baseada em JavaScript que visa simplificar e reduzir o processo de codificação. Além disso, o JQuery é pré-requisito para várias outras *frameworks* disponíveis como *open source*.

Para obtenção das características de um elemento HTML basta lançar o objeto *$(<elemento>)* que representa o elemento pesquisado no console do navegador, por meio do código *console.log($(<elemento>))*, considerando o <*elemento*> como: a *string* do símbolo "#" com o nome do ID do elemento único; como a *string* do símbolo "." com o nome da classe ou como apenas a *string* do nome da *tag* HTML.

Alguns métodos e propriedades usuais do JQuery são:

- **$(<elemento>).css(<característica>, <valor>)**: Alteração ou inclusão de um comando CSS.
- **$(<elemento>).prop(<propriedade>, <valor>)**: Alteração ou inclusão de uma propriedade do HTML, como *disabled* ou *hidden*.

- **$(<elemento>).val(<novo valor>)**: Alteração do valor de uma entrada de dados de um elemento HTML. Pode ser utilizado sem o parâmetro do novo valor, caso o programador queira obter o valor atual.
- **$(<elemento>).html(<novo valor>)**: Alteração do conteúdo HTML de um elemento. Também pode ser utilizado sem o parâmetro do novo valor, caso o programador queira obter o conteúdo HTML atual.
- **$(<elemento>).sibling()**: Obtém os demais elementos que estão na mesma tag HTML que o item <elemento>.
- **$(<elemento>).find(<elemento a procurar>)**: Procura um elemento específico, pela classe, ID ou *tag*, que está dentro do elemento principal.

2.5.5.2 Bootstrap

A *framework* Bootstrap é o kit de ferramentas *open source*, onde é possível acessar diversos recursos lógicos e de estilização de forma simplificada. De acordo com Schmitz (2014), “(...) o bootstrap é utilizado para desenhar telas em html, que serão acessadas via navegador web ou dispositivo mobile”.

A utilização do Bootstrap se dá em duas etapas. A primeira é a importação das *frameworks* necessárias, o que inclui as do Bootstrap (CSS e JS) e as de pré-requisito, Popper e JQuery – versão completa ou *slim* (compacta). Após isso, basta utilizar as classes e atributos HTML disponibilizados pela *framework* para obtenção dos estilos, das classes (da linguagem orientada a objetos) e eventos em JQuery para a utilização da lógica. Com isso, é possível usufruir da estilização e de lógicas de diversos elementos, como:

- **Alerts**: Alertas.
- **Badges**: Distintivos.
- **Buttons**: Botões.
- **Carousel**: Carrosséis.
- **Forms**: Formulários.
- **Input groups**: Grupo de entradas de dados.
- **Moldals**: Modais – pop-up que se sobrepõe à tela.

- **Navbars**: Barra de navegação.
- **Tootips**: Dicas de ferramentas.

2.5.5.3 Animate.css

Segundo o site e documentação do “Animate.css” (ANIMATE, 2021), esta *framework* possibilita a aplicação movimentos em elementos HTML com alto dinamismo. Para utilizá-lo é necessário importar a biblioteca do “Animate.css” que é exclusivamente em CSS e atribuir as classes das animações escolhidas nos elementos. Suas as animações são divididas em: chamar atenção, entrada ou saída por traz, entrada ou saída saltitante, entrada ou saída por desbotamento, giros, velocidade da luz (aceleração), entrada ou saída rotacionada, entrada ou saída por zoom, entrada ou saída deslizante e animações especiais.

2.5.5.4 Bootbox

O Bootbox é uma *framework* que gera, de forma mais simples e dinâmica para o programador, modais do Bootstrap. Esses modais podem ser de simples alertas até confirmações, *prompts* e diálogos customizados com retorno de dados do usuário, conforme a documentação da biblioteca (BOOTBOX, 2021). A utilização dessa biblioteca traz velocidade para a programação, porém reduz a flexibilidade na modificação dos modais.

2.5.5.5 AMCharts

O AMChats é uma *framework* de gráficos em JavaScript que disponibiliza gratuitamente centenas de modelos de gráficos com vários atributos modificáveis. O gráficos são em grande maioria animados pelo “Animate.css”.

Para utilização desta *framework*, é necessário que o HTML contenha uma tag “ ” para que o gráfico seja incluído nela por meio das classes em JavaScript. As classes incluem a definição de propriedades dos eixos, dos rótulos de dados, do tipo de gráfico, das linhas de grade, entre outros. A atribuição dos dados se dá por meio de um objeto Json (AMCHARTS, 2021).

2.6 ORGANIZAÇÃO DO CÓDIGO

A organização do código é primordial para alcançar uma maior velocidade no desenvolvimento, na manutenção e na atualização do software. Para obter um bom estado de organização, deve-se seguir as boas práticas de programação e ter uma boa organização de pastas. Neste trabalho, foi abordada a estrutura de pastas e interações MVVM (Model View ViewModel).

3 PESQUISA REALIZADA

O estudo foi levantamento para uma empresa localizada no norte do estado do Espírito Santo. Esta empresa possui um sistema de gestão estruturado e certificações da ISO, incluindo a ISO 9001. Nesta empresa os esforços empresariais são direcionados por indicadores e ações.

Os indicadores desta empresa são, em geral, mantidos pelas áreas a que eles pertencem. Porém, embora a empresa possua um sistema da informação eficaz, a maioria das áreas mantém os indicadores alocados em planilhas em pastas de rede. Isso gera uma vulnerabilidade para o indicador que é uma importante evidência do cumprimento das diretrizes estratégicas.

Além disso, o mesmo problema foi observado para as ações e os planos de ação. Segundo a companhia, é uma exigência dos órgãos certificadores e do próprio sistema de gestão da empresa que todos os indicadores possuam metas claras e bem difundidas na empresa e que ações sejam tomadas caso essas metas não sejam atingidas.

Sendo assim, mediante a situação averiguada, foi desenvolvido um sistema *web* para que este controle de indicadores seja eficaz, rápido e bem difundido dentro da organização.

Para realização levantamento dos requisitos do sistema, foi realizada uma entrevista com alguns colaboradores que trabalham diretamente como a atualização e manutenção desses indicadores dentro da empresa. Após isso, as informações levantadas foram organizadas em um diagrama de caso de uso e um diagrama de classes, bem como nos modelos lógico e físico do banco de dados.

No diagrama de caso de uso, foi verificado que o colaborador deve acessar a uma página de *login*, uma página de indicadores e uma página de ações. O *login* só pode ser realizado por parte do usuário se o mesmo tiver realizado o cadastro no sistema. Uma vez logado, o usuário poderá visualizar os indicadores, cadastrar novos indicadores e incluir os resultados dos indicadores pelos quais ele é responsável. Na página de ações, ele poderá consultar todas as ações pendentes e concluir as que estiverem pendentes em seu nome. Essas informações estão sintetizadas no Diagrama de Caso de Uso da Figura 1, onde é possível ver os requisitos funcionais do sistema.

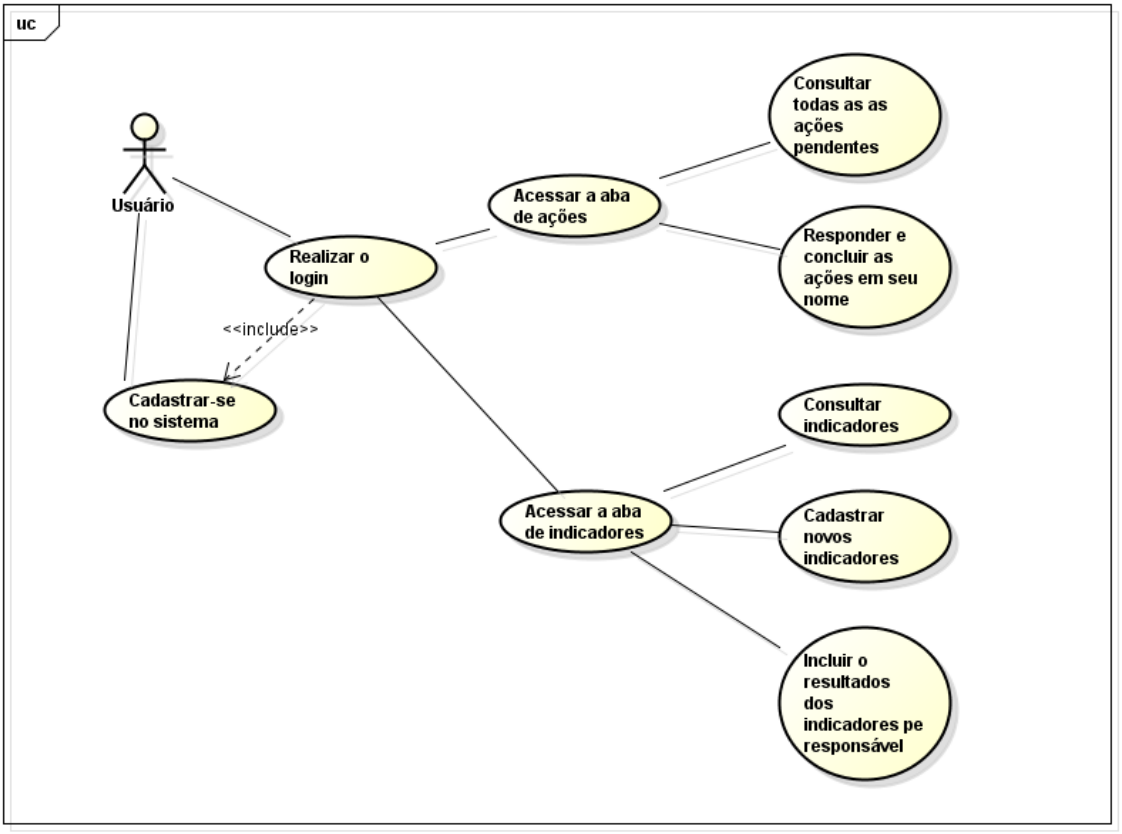

***Figura 1**: Diagrama de Caso de Uso.*

Em relação ao banco de dados, foi averiguada a necessidade de considerar quatro tabelas, especificados no modelo lógico da Figura 2: tbl_indicador, tbl_resultado, tbl_usuario e tabl_area.

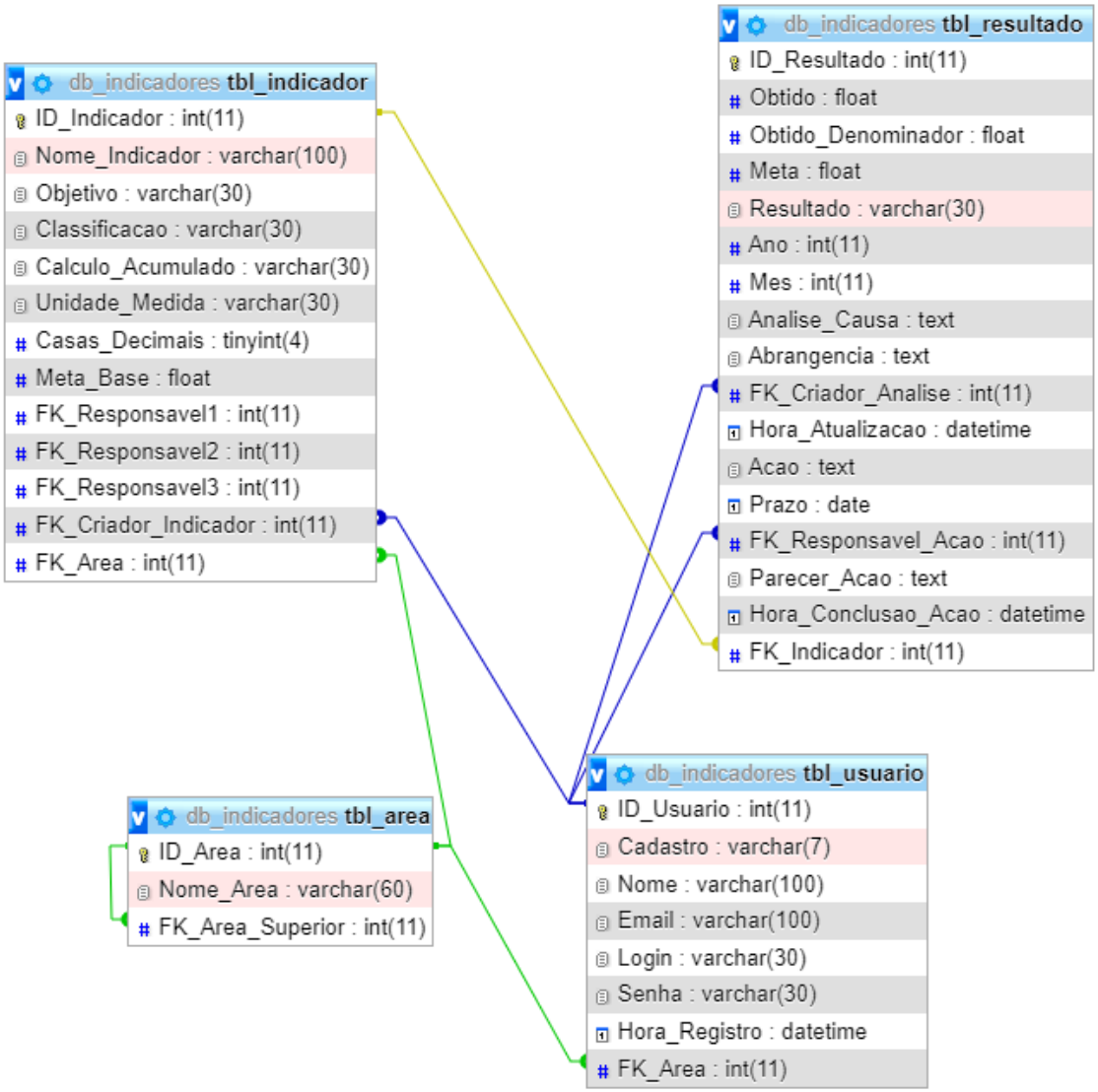

Figura 2: Modelo Lógico do Banco de Dados.

Os atributos de cada uma das tabelas estão especificados e explicados nas Tabelas 1, 2, 3, e 4.

Tabela 1: Atributos da tabela tbl_indicador

Nome do Campo	Descrição do Campo
ID_Indicador	Código único gerado automaticamente pelo sistema.
Nome_Indicador	Nome do indicador.
Objetivo	O objetivo do indicador pode conter duas opções: Aumentar ou Reduzir.
Classificacao	A classificação do indicador se enquadra nas seguintes opções: Produtividade, Qualidade, Capacidade ou Estratégico.
Calculo_Acumulado	Forma com a qual o resultado acumulado do ano é calculado: Média, Média ponderada ou Soma

Unidade_Medida	Unidade de medida do indicador.
Casas_Decimais	Número de casas decimais utilizada para o resultado acumulado.
Meta_Base	Meta do indicador ao longo do ano. Pode ser atualizada ao longo tempo.
FK_Responsavel1	Responsáveis pelo cadastramento dos resultados. A existência de três responsáveis é um requisito conformado pelas regras de gestão da empresa voltadas à versatilidade do uso da mão de obra.
FK_Responsavel2	
FK_Responsavel3	

Tabela 2: Atributos da tabela tbl_resultado.

Nome do Campo	Descrição do Campo
ID_Resultado	Código único gerado automaticamente pelo sistema.
Obtido	O valor numérico obtido por mês.
Obtido_Denominador	Trata-se do montante do indicador. Este campo é solicitado ao usuário quando o método de cálculo acumulado do indicador é a média ponderada. Assim, o montante serve como peso ponderado para o cálculo.
Meta	Como a meta-base da tabela tbl_indicador pode ser atualizada ao longo do tempo, cada resultado deve conter a meta válida (histórico) para o período a que ele se refere.
Resultado	O resultado do indicador pode ser: Satisfatório ou Insatisfatório.
Ano	Ano ao qual o resultado se refere.
Mes	Mês ao qual o resultado se refere.
Analise_Causa	Texto com a análise das possíveis causas do resultado insatisfatório.
Abrangencia	A abrangência é o local, físico ou não, onde o problema foi causado para o resultado ser insatisfatório.
FK_Criador_Analise	Chave estrangeira do usuário que cadastrou o resultado da análise.
Hora_Atualizacao	Hora em que o resultado foi cadastrado.
Acao	Ação, corretiva ou preventiva para resolver a causa do resultado insatisfatório.
Prazo	Prazo máximo para conclusão da ação.
FK_Responsavel_Acao	Responsável por realizar ou coordenar a execução da ação.
Parecer_Acao	O parecer de conclusão da ação é o que realmente foi feito para resolver o problema. Nem sempre ação levantada pode ser integralmente aplicada na prática.
Hora_Conclusao_Acao	Horário em que a ação foi concluída, isto é, que o parecer foi cadastrado.

FK_Indicador	Chave estrangeira para apontar a qual indicador o resultado se refere.

Tabela 3: Atributos da tabela tbl_area

Nome do Campo	Descrição do Campo
ID_Area	Código único gerado automaticamente pelo sistema.
Nome_Area	Nome da área, seção, departamento, diretoria, entre outros.
FK_Area_Superior	Chave estrangeira que faz um auto relacionamento com o ID_Area para indicar a área hierarquicamente superior.

Tabela 4: Atributos da tabela tbl_usuario

Nome do Campo	Descrição do Campo
ID_Usuario	Código único gerado automaticamente pelo sistema.
Cadastro	Número de cadastro do colaborador na empresa.
Nome	Nome completo do usuário.
Email	E-mail do usuário.
Login	Login para acesso ao sistema.
Senha	Senha para acesso ao sistema.
Hora_Registro	Hora em que o colaborador se cadastrou no sistema.
FK_Area	Chave estrangeira para indicar a qual área da empresa o colaborador está alocado.

A partir do modelo lógico do banco de dados, foi desenvolvido o modelo físico (codificação) pelo MySql. Além disso, foi desdobrado o diagrama de classes do sistema.

No diagrama de classes, representado pela Figura 3, são representadas as classes do sistema e como elas se relacionam. Em cada classe também é possível ver suas propriedade e métodos.

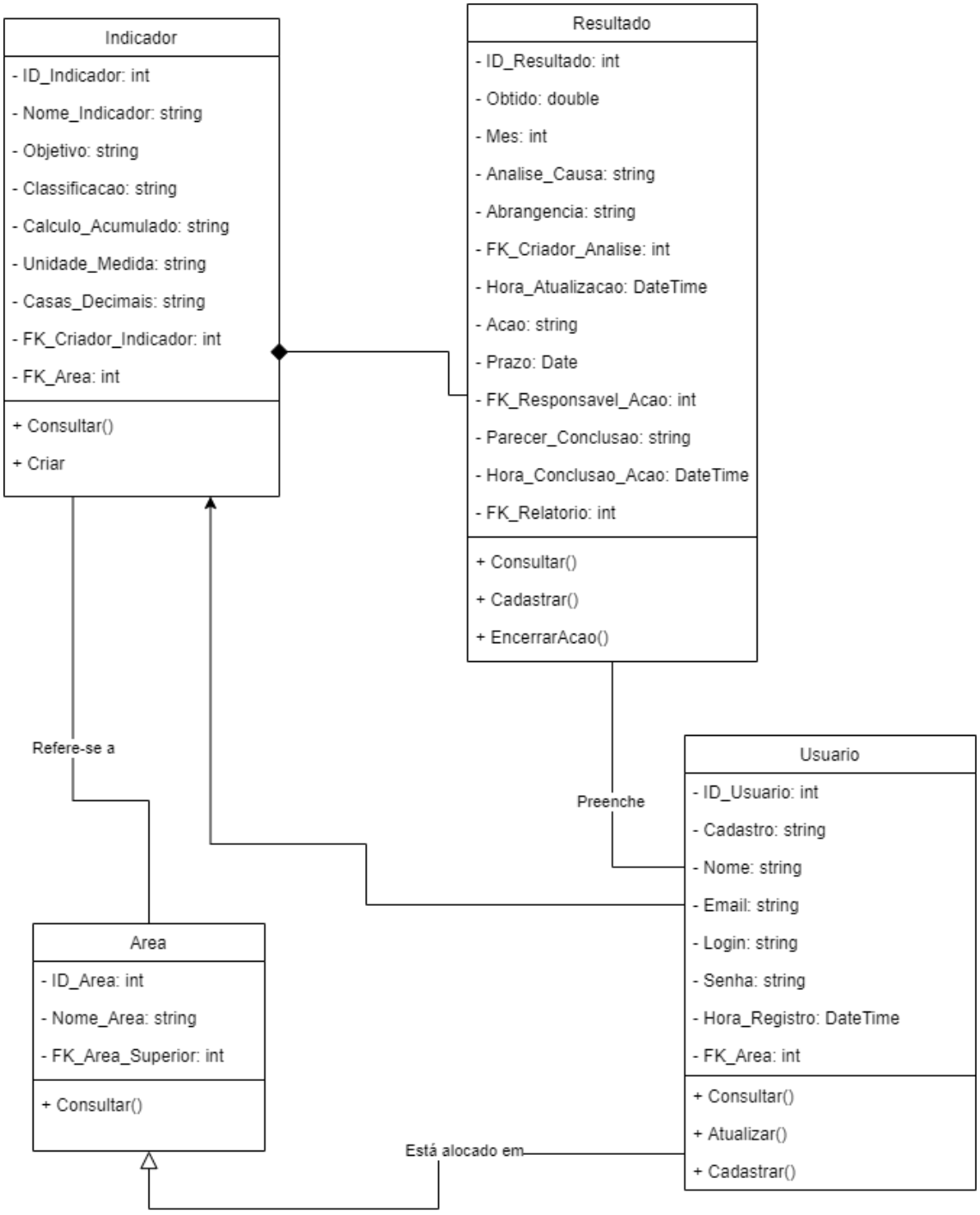

Figura 3: Diagrama de Classes.

A prototipagem do site foi realizada por meio do software Pencil com o objetivo de alinhar o layout com o usuário. As algumas imagens da prototipagem estão disponibilizadas nas Figuras 4, 5, 6 e 7.

Figura 4: Prototipagem – Aba Ações

Figura 5: Prototipagem - Consulta dos indicadores

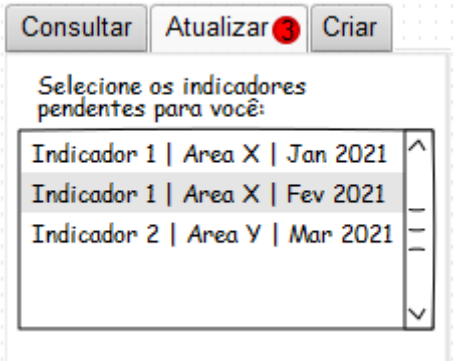

Figura 6: Prototipagem – Aba atualizar indicador

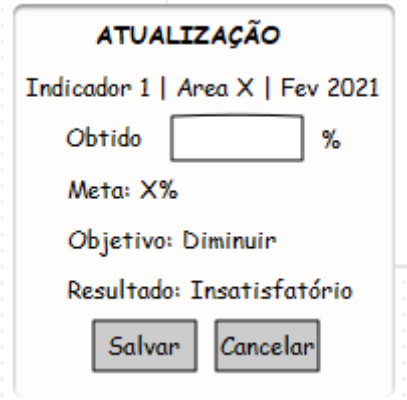

Figura 7: Prototipagem – Modelo de modal simplificado para cadastramento de resultados e conclusão de ações

Na estrutura de conexão com o banco de dados no *back-end*, foi utilizado o PHP. A organização básica do código se baseou na criação de uma classe individual para conter os dados de conexão e os métodos de execução de comandos do MySql similares aos disponibilizados no DotNet: Reader, NonQuery e Scalar. No Reader é possível consultar várias linhas de dados com diferentes campos; no Scalar essa consulta se limita a um dado; e no NonQuery a implementação foi feita para que ele retorne sempre o índice onde o dado foi salvo, alterado ou deletado, diferente do DotNet que não retorna nada para o NonQuery. Com estes três métodos executados e testados numa classe individual, os demais arquivos PHP apenas declaram a classe e chamam um desses três métodos. Assim, com essa proposta de organização do código no *back-end*, houve uma maior agilidade na codificação e nos testes

de caixa branca, pois os erros se tornaram mais fáceis de serem identificados nas camadas dissecadas.

Para o desenvolvimento lógico do *front-end*, utilizou-se majoritamente o JQuery – ao invés do JavaScript puro – e as camadas foram organizadas conforme o método MVVM. Com ele, foi possível executar ações de consulta no banco de dados simultaneamente, isto é, de forma assíncrona. Este assincronismo trouxe uma agilidade para execução conveniente, ao passo que a performance de um sistema *web* está condicionada à conexão com a rede e à disponibilidade do servidor. Além disso, o MVVM proporcionou uma maior agilidade para o desenvolvimento, atualização e manutenção do sistema.

As *frameworks* Bootstrap, Bootbox, AmChats e "Animate.css" foram utilizadas amplamente no código e também proporcionaram um desenvolvimento mais ágil e uma interface mais agradável.

As Figuras 8, 9, 10, 11 e 12 contém os layouts finais do sistema desenvolvido neste trabalho.

Figura 8: Tela de login do sistema web desenvolvido

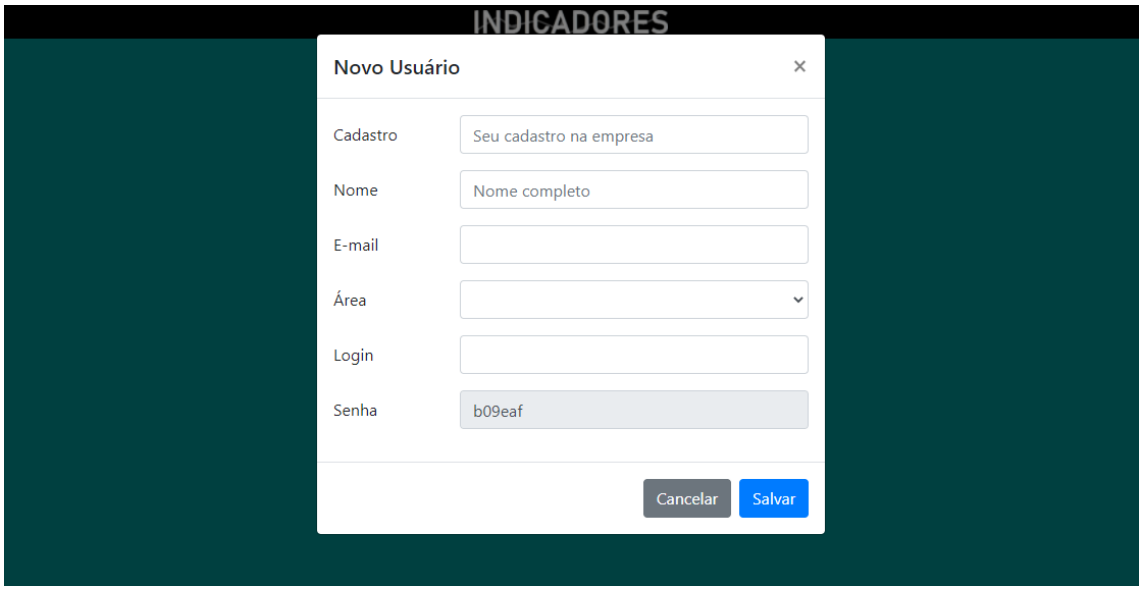

Figura 9: Cadastro um novo usuário no sistema web desenvolvido.

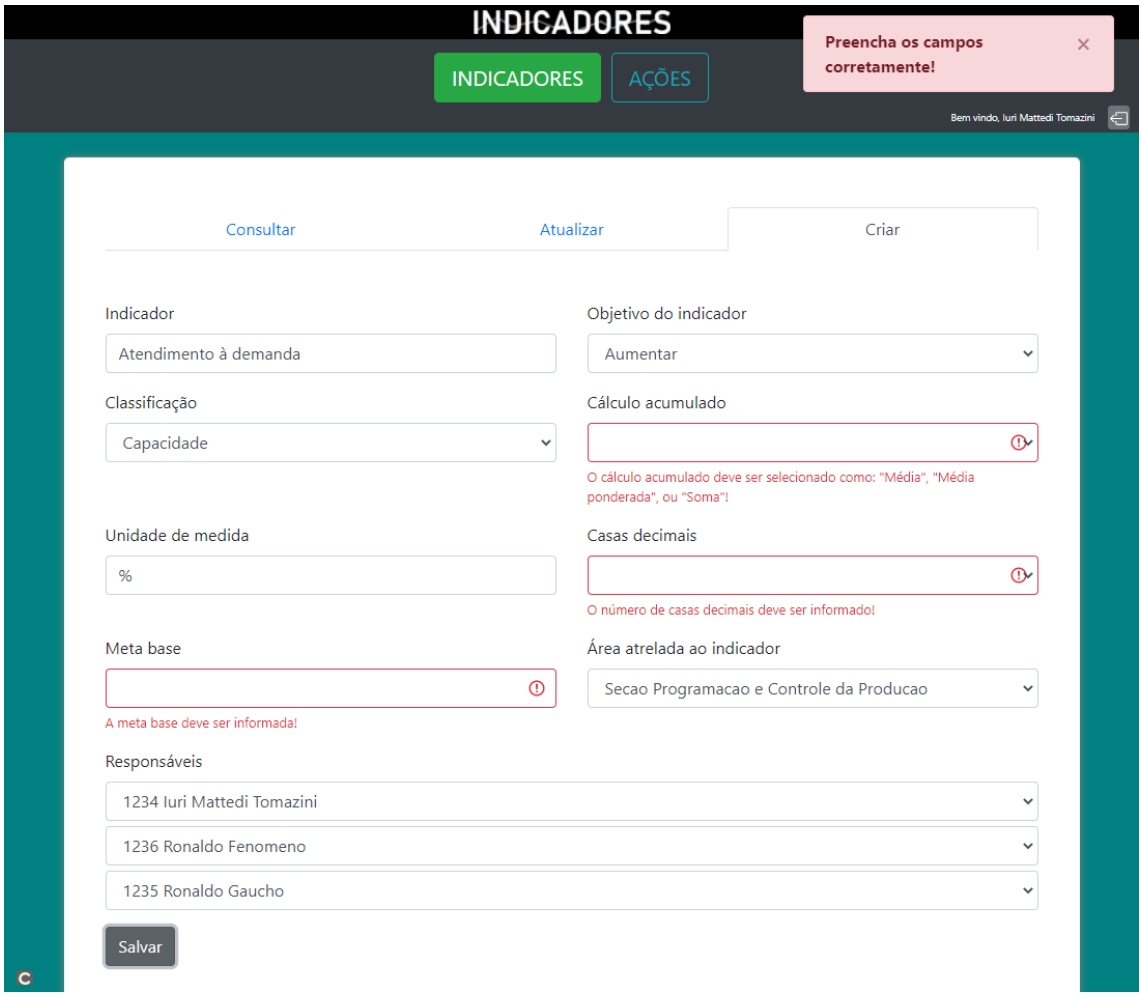

Figura 10: Criação de novos indicadores pelo sistema web desenvolvido.

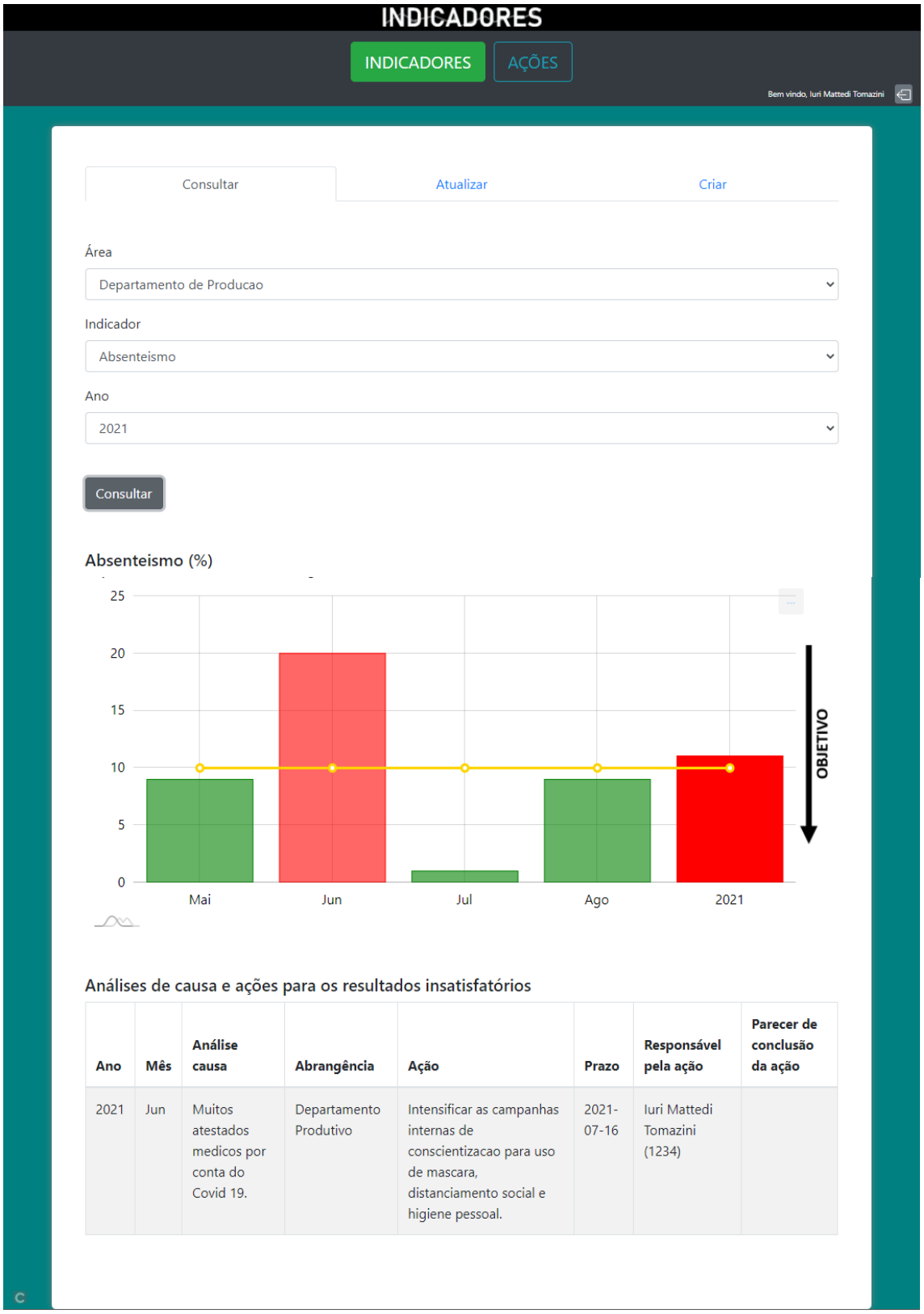

Figura 11: Consulta de indicadores pelo sistema web desenvolvido.

Figura 12: Ações pendentes (1) para o usuário logado no sistema web desenvolvido.

4 CONSIDERAÇÕES FINAIS

O objetivo geral deste trabalho foi desenvolver um software para gerenciar indicadores, análises de causas e ações. Dessa forma, foi realizada uma revisão bibliográfica dos recursos de programação, foi feito o levantamento de requisitos para o *site* em uma empresa do norte do Espírito Santo e foi desenvolvido o sistema em questão.

Na revisão bibliográfica se destacaram as *frameworks* de programação *web*. Dentre as *frameworks* pesquisadas estão o JQuery, o Bootstrap, o Bootbox, o “Animate.css” e o AMCharts. Todas estas bibliotecas representam uma metodologia atualizada de programação, pois facilitam e agilizam os desenvolvimento tornando-o com um nível mais alto, isto é, mais próximo do usuário. Além disso, as *frameworks* trouxeram um dinamismo maior para o sistema proporcionando uma melhor usabilidade em um tempo de dedicação reduzido.

Outro ponto crucial atingido por este trabalho foi a organização do código. A estrutura de pastas utilizada proporcionou uma boa manutenabilidade, garantindo mais eficiência com a execução de tarefas assíncronas. Além disso,

a organização do *back-end* (PHP) para acesso aos dados possibilitou uma maior agilidade no desenvolvimento e nos testes de caixa branca na camada de acesso aos dados. Assim, para o contexto de utilização empresarial do sistema *web* – cujo desempenho está condicionado ao acesso ao servidor e cuja manutenabilidade deve ser alta – essas propostas foram satisfatórias.

Por fim, o software desenvolvido atendeu às expectativas dos usuários entrevistados, culminando numa redução do risco da organização receber não-conformidades em auditorias de órgão certificadores, provendo um plataforma que mantém evidências de resultados, reduzindo o tempo consumido com a atualização e a consulta de indicadores e melhorando o acesso aos relatórios e a gestão participativa.

5 REFERÊNCIAS

AMCHARTS. **Documentação do site AMCharts.** [2021]. Disponível em: <https://www.amcharts.com/>. Acesso em: 17/08/2021.

ANIMATE.CSS. **Documentação do site "Animate.css".** [2021]. Disponível em: <https://animate.style/>. Acesso em: 18/08/2021.

BARBOSA, Fabrício Felipe Meleto, FREITAS, Pedro Henrique Chagas. **Modelagem e desenvolvimento de banco de dados.** Porto Alegre, Sagah Educação S.A, 2018. 188p.

BOOTBOX. **Documentação do site BootboxJS.** [2021]. Disponível em: < http://bootboxjs.com//>. Acesso em: 15/08/2021.

BOOTSTRAP. **Documentação do site Bootstrap.** [2021]. Disponível em: <https://getbootstrap.com/>. Acesso em: 15/08/2021.

BOOCH, Grady Grady. **UML – Guia do Usuário**. LTC, 2006. 2 ed. ISBN-10: 9788535217841.

DUCKETT, Jon. **HTML e CSS: projete e construa websites.** Alta Books, 2016. ISBN-10: 8576089394

FRANCISCHINI, Paulino G; FRANCISCHINI, Andressa S. N. **Indicadores de desempenho dos objetivos à ação – métodos para elaborar KPIs e obter resultados.** Rio de Janeiro: Alta Books, 2017. 448 p. Disponível em: <https://www.google.com.br/books/edition/Indicadores_de_Desempenho/O_dwDwAAQBAJ?hl=pt-BR&gbpv=1&dq=indicadores+organizacionais&printsec=frontcover>.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4ª ed. São Paulo: Atlas, 2002.

GUEDES, Gilleanes T. A. **UML 2: Uma abordagem prática.** São Paulo: Novatec Editora, ed.2 (2011).

KNAPP, Micheal. **HTML & CSS: Learn The Fundamentals In 7 days.** Draft2digital, 2017. 114 p. ISBN 10: 1520562594.

LEMES, Giovanni Bugni. Processos Organizacionais. (n.d.). (n.p.): Clube de Autores (managed). 72 p. Ed 1 (2018). Disponível em: <https://www.google.com.br/books/edition/Processos_Organizacionais/G8l5DwAAQBAJ?hl=pt-BR&gbpv=0>.

MILETTO, Evandro Manara *et. al.* **Desenvolvimento de Software II. Introdução ao desenvolviemtno weg com html, css, javascript e php.** Porto Alegre: 2014. 266 p.

MYSQL. **Documentação do site MySql.** [2021]. Disponível em: <https://dev.mysql.com/doc/>. Acesso em: 19/08/2021.

PHP. **Documentação do site PHP.** [2021]. Disponível em: <https://www.php.net/docs.php>. Acesso em: 19/08/2021.

RAUSCHMAYER, Dr. Axel. **JavaScript For Impatient Programmers**. Independently Published: 2021. ISBN 978-1-09-121009-7. Disponível em: <https://exploringjs.com/impatient-js/downloads/impatient-js-preview-book.pdf>. Acesso em: 10/08/2021.

SARAIVA, Mauricio de Oliveira; BARRETO, Jeanine do Santos. **Desenvolvimento de Sistemas com PHP.** Porto Alegre: SagahEducação S.A., 2018. 268 p.

SILVA, Maurício Samy. JQuery: A Biblioteca do Programador JavaScript. Navatec, 2013. 2 ed. ISBN-10: 8575223879 Disponível em: <https://s3.novatec.com.br/capitulos/capitulo-9788575222379.pdf>. Acesso em: 15/08/2021.