

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

**APLICAÇÃO WEB PARA
GERENCIAMENTO E APOIO EM
TAREFAS**

LUCAS GONÇALVES VICENTE

Trabalho de Conclusão II apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Cristina Moreira Nunes

**Porto Alegre
2022**

APLICAÇÃO WEB PARA GERENCIAMENTO E APOIO EM TAREFAS

RESUMO

Com o advento do trabalho a distância diversos benefícios surgiram, porém junto com eles, também algumas dificuldades. Mesmo que algumas atividades possam ser realizadas nesta modalidade sem que o trabalho em si seja penalizado, surge um novo problema, que é manter a comunicação entre as equipes de forma dinâmica e com alta disponibilidade. Levando em consideração a quantidade de soluções que podem ser implementadas por meio da utilização das tecnologias mais recentes no mercado, foi desenvolvido o sistema *Kanbro*, no formato de uma aplicação *Web*. Para cumprir com o seu objetivo, o *Kanbro* conta com um quadro *Kanban*, possibilitando que atividades sejam listadas e gerenciadas por uma equipe e um sistema de *chat* integrado, tornando possível a comunicação em tempo real e unindo a gestão de atividades. Este documento detalha as funcionalidades presentes no sistema, as tecnologias utilizadas no seu desenvolvimento e a arquitetura utilizada para a sua implementação.

Palavras-Chave: Aplicações *Web*, *Kanban*, Comunicação.

WEB APPLICATION FOR TASK SUPPORT AND MANAGEMENT

ABSTRACT

With the advent of remote work, several benefits have emerged, but along with them, some difficulties have also shown up. Even though some activities can be carried out in this modality without the work itself being penalized, a new problem emerged, namely, how to maintain communication between teams dynamically and with high availability. Taking into account the number of solutions that can be implemented using the latest technologies on the market, the *Kanbro* system was developed, in the form of a *web* application. To fulfill its objective, *Kanbro* has a *Kanban* board, allowing activities to be listed and managed by a team and an integrated *chat* system, making real-time communication possible uniting activity management. This document details the features present in the system, the technologies used in its development and the architecture used for its implementation.

Keywords: Web Applications, Kanban, Communication.

LISTA DE FIGURAS

Figura 3.1 – Quadro de tarefas.....	14
Figura 3.2 – Quadro <i>Kanban</i> no Monday.....	15
Figura 3.3 – Quadro <i>Kanban</i> no Trello.....	15
Figura 3.4 – Tela de adição de <i>checklists</i>	16
Figura 3.5 – Quadro com cartões de <i>e-mails</i>	16
Figura 3.6 – Barra de cronograma.....	17
Figura 4.1 – Tela de Cadastro.....	19
Figura 4.2 – Tela de <i>Login</i>	20
Figura 4.3 – Tela Gerenciamento de time.....	20
Figura 4.4 – Tela Cadastro de Time.....	21
Figura 4.5 – Tela Quadro <i>Kanban</i>	21
Figura 4.6 – Tela de Espera.....	22
Figura 4.7 – Opção Adicionar Membros.....	22
Figura 4.8 – Tela Adição de Membros.....	23
Figura 4.9 – Tela Quadro <i>Kanban</i>	23
Figura 4.10 – Tela Criação de Tarefa.....	24
Figura 4.11 – Opção Listar Membros.....	24
Figura 4.12 – Tela Lista de Membros.....	25
Figura 4.13 – Criar <i>Chat</i> em Cartão.....	25
Figura 4.14 – Tela de <i>Chat</i>	26
Figura 4.15 – Tela de <i>Chat</i> com Mensagens.....	26
Figura 5.1 – Senha criptografada.....	29
Figura A.1 – Diagrama de casos de uso.....	36
Figura A.2 – Diagrama de Classes.....	38

LISTA DE TABELAS

Tabela 3.1 – Tabela comparativa de funcionalidades	17
--	----

LISTA DE SIGLAS

API – Application Programming Interface

DTO – Data Transfer Object

HTTP – Hypertext Transfer Protocol

JPA – Java Persistence API

JSON – JavaScript Object Notation

MVC – Model-View-Controller

REST – Representational State Transfer

UC – User Case

SUMÁRIO

1	INTRODUÇÃO	8
2	FUNDAMENTAÇÃO TEÓRICA	10
2.1	METODOLOGIA DE DESENVOLVIMENTO ÁGIL UTILIZANDO <i>KANBAN</i>	10
2.2	APLICAÇÃO WEB	11
2.2.1	REST	11
2.2.2	PROTOCOLO HTTP	11
2.3	JSON	12
2.4	PADRÃO MVC	12
2.5	REACT	13
2.6	WEBSOCKET	13
3	TRABALHOS RELACIONADOS	14
3.1	MONDAY	14
3.2	TRELLO	15
3.3	FLOW-E	16
4	VISÃO GERAL DO SISTEMA	18
4.1	MOTIVAÇÃO E OBJETIVO GERAL	18
4.2	OBJETIVOS ESPECÍFICOS	18
4.3	FUNCIONAMENTO DO SISTEMA EM DETALHES	19
5	IMPLEMENTAÇÃO	27
5.1	SERVIDOR	27
5.1.1	PERSISTÊNCIA DE DADOS	28
5.1.2	AUTENTICAÇÃO	28
5.2	CLIENTE	29
5.2.1	REQUISIÇÕES AO CLIENTE VIA HTTP	29
5.2.2	LOCAL STORAGE	30
5.2.3	CHAT COM WEBSOCKET	30
5.2.4	CONTEXT API	31
6	CONCLUSÃO	32

6.1	TRABALHOS FUTUROS	32
	REFERÊNCIAS BIBLIOGRÁFICAS	34
	APÊNDICE A – MODELAGEM	36

1. INTRODUÇÃO

Com o grande crescimento das empresas existentes e o surgimento de novas corporações, a prática de trabalhar em escritórios, que teve início entre os séculos XIX e XX, foi necessária devido ao aumento de burocracia e a concentração de um maior número de pessoas em um mesmo espaço de trabalho. Com o passar dos tempos o uso dos escritórios foram se modificando e partir da década de 90, a busca por espaços otimizados que pudessem motivar e fazer com que os colaboradores produzissem resultados melhores, se tornou uma prática realizada até os dias de hoje, [Silva 2018]. Porém mesmo com diversas melhorias e cuidados para atender as necessidades dos colaboradores, o crescimento dos grandes centros urbanos, a vida corrida do século XXI e as novas formas de trabalho estão revolucionando a forma como as pessoas lidam com o uso dos escritórios. Novas tecnologias como Skype ¹, Zoom ², Discord ³, Microsoft Teams ⁴ e muitas outras ferramentas de comunicação virtual possibilitaram a implementação do sistema de trabalho remoto. Segundo matéria escrita no G1 por [Silveira 2019], temos a seguinte afirmação:

“Um levantamento divulgado nesta quarta-feira (18) pelo Instituto Brasileiro de Geografia e Estatística (IBGE) mostra que, em 2018, 3,8 milhões de brasileiros trabalhavam dentro de casa, o chamado *home office*.”

No entanto, uma situação totalmente inesperada pegou o mundo todo de surpresa. Em 2020 se deu início a uma nova pandemia causada pelo Corona vírus, tendo como uma das primeiras e principais recomendações para impedir a disseminação do vírus, o isolamento social da população, fazendo com que o trabalho remoto para muitos trabalhadores se tornasse a nova realidade. Com o advento da pandemia de Corona vírus, diversas empresas tiveram que adotar o sistema de trabalho remoto em caráter emergencial e sem preparo prévio, mudança essa que trouxe diversas dificuldades e novos desafios, sendo alguns deles lidar com falta de estrutura, manter a cooperação e a comunicação a distância, [Nawaz 2021].

Este trabalho apresenta uma solução em forma de uma aplicação *web*, denominada **Kanbro**, que tem por objetivo auxiliar a superar os desafios apresentados anteriormente, conciliando a organização das tarefas e a comunicação ativa para a resolução de qualquer impedimento que venha a ocorrer durante a jornada de trabalho. O objetivo desse trabalho é definir uma aplicação *web* que entregue capacidade de gerenciamento de tarefas ao mesmo tempo que permite a qualquer integrante do projeto reportar de forma instantânea qualquer dificuldade que esteja tendo na resolução da tarefa. A solução entregue visa atender estes requisitos disponibilizando um quadro *Kanban*, utilizado para gerenciamento

¹www.skype.com

²www.zoom.us

³www.discord.com

⁴www.microsoft.com/pt-br/microsoft-teams/group-chat-software

de tarefas, que conterà uma funcionalidade de *chat* integrado diretamente em cada atividade do quadro, possibilitando a capacidade de apontar uma dificuldade ou dúvida, onde todos os membros da equipe podem se comunicar e auxiliar na resolução do problema de forma rápida e intuitiva.

Além desta introdução, este trabalho de conclusão foi dividido em capítulos. No Capítulo 2 será apresentado um estudo teórico relevante para o desenvolvimento da solução. O Capítulo 3, que apresenta as principais características de alguns trabalhos relacionados utilizados como referência. No Capítulo 4 é apresentada uma visão geral com detalhes do funcionamento. Já no Capítulo 5 é apresentada a forma como a aplicação foi implementada.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os principais conceitos utilizados para desenvolvimento e o entendimento do trabalho.

2.1 Metodologia de desenvolvimento ágil utilizando *Kanban*

De acordo com a afirmação descrita no *blog* [RedHat 2020] a metodologia ágil:

“É uma abordagem do desenvolvimento de software que busca a entrega frequente de aplicações funcionais criadas por meio de iterações rápidas.”

O termo “Metodologia Ágil” surgiu a partir da criação do Manifesto Ágil, um documento que tem como seus principais conceitos:

- indivíduos e interações ao invés de processos e ferramentas;
- *software* executável ao invés de documentação;
- colaboração do cliente ao invés de negociação de contratos;
- respostas rápidas a mudanças ao invés de seguir planos.

Outros componentes como ferramentas, documentações, contratos, não foram excluídos do processo, porém recebem menos atenção pois o foco da agilidade esta nas pessoas, nas interações, no sistema e na participação do cliente durante o processo de desenvolvimento.

O *Kanban* é uma metodologia dentre diversas outras presentes no desenvolvimento ágil, que foi desenvolvida com o objetivo de tornar simples e rápidas as atividades de programação, controle e acompanhamento de sistemas de produção em lotes, [COSTA 2008]:

Um dos tipos de implementação de *Kanban* se dá por três murais, que são eles:

- *To do*: Tarefas que estão esperando para serem realizadas;
- *Doing*: Tarefas que estão em desenvolvimento;
- *Done*: Tarefas concluídas.

Para o contexto do projeto, esta foi a abordagem do *Kanban* escolhida pois abrange as necessidades do produto.

2.2 Aplicação Web

A aplicação *web* diz respeito a uma solução que é executada diretamente no *browser*, ou navegador, não sendo preciso realizar uma instalação na máquina do usuário. Pode-se, também, utilizar como definição “tudo aquilo que é processado em um servidor terceiro”, [Noletto 2020].

Um dos maiores benefícios encontrados em aplicações *web* se dá pelo fato de não precisar ser desenvolvido para cada plataforma como nas configurações tradicionais. Uma vez que muitas destas aplicações são concebidas para serem executadas dentro de um navegador *web*, a arquitetura subjacente é em grande parte sem importância, [Macêdo 2017].

Dentro do escopo da solução desenvolvida, a implementação de uma aplicação *web* se faz necessária devido a falta de necessidade de instalações complementares para a utilização do sistema, tornando acessível ao maior número de usuários possível.

2.2.1 REST

REST, ou *Representational State Transfer*, é um estilo arquitetônico aplicado para fornecer padrões entre sistemas de computador na *web*, facilitando a comunicação entre eles. Os sistemas em conformidade com REST, muitas vezes conhecidos como sistemas *RESTful*, são reconhecidos pelo jeito como separam as preocupações do cliente e do servidor, [de Souza 2020].

A ideia do REST é utilizar de maneira mais eficiente e em sua plenitude as características do protocolo HTTP, principalmente no que diz respeito à semântica do protocolo [Campomori 2016].

Devido a separação de responsabilidades disponibilizadas pela arquitetura REST, entre *back-end* e *front-end*, sua alta escalabilidade, onde é possível acoplar novos recursos sem dificuldade e sua troca de informações padronizada que garante maior confiabilidade a arquitetura foi utilizada no projeto.

2.2.2 Protocolo HTTP

O protocolo HTTP, ou *Hypertext Transfer Protocol*, é uma forma que temos de obter recursos de um servidor através de comunicações codificadas na camada de aplicação. A principal função do protocolo HTTP é justamente garantir uma troca de informações cliente-servidor [Garcia 2021].

O protocolo HTTP foi utilizado, pois esta presente na arquitetura REST, que foi escolhida para o desenvolvimento da aplicação.

2.3 JSON

JSON, ou *JavaScript Object Notation*, é um formato simples de troca de informações entre cliente-servidor. Hoje em dia este padrão muito popular em aplicações *web* devido ao fato de ser de fácil escrita e compreensão para seres humanos. O JSON é construído basicamente em duas estruturas, sendo elas uma coleção de pares nome/valor e uma lista ordenada de valores, também conhecidos como *arrays* ou vetores. Por ser uma estrutura de dados universal, praticamente todas as linguagens de programação modernas a suportam.

2.4 Padrão MVC

O padrão de projeto MVC ou *Model-View-Controller* é um padrão arquitetural que possibilita a divisão do projeto em camadas muito bem definidas. Os conceitos e responsabilidades dos componentes dessa arquitetura são:

- *Model* ou Modelo: Esta camada é responsável por representar os dados, modelando o comportamento por trás do processo de negócio, tendo como responsabilidade definir a manipulação e geração de dados;
- *Controller* ou Controlador: Serve de camada intermediária, determinando e intermediando o fluxo de troca de mensagens entre *Model* e *View*;
- *View* ou Visão: É responsável apenas por exibir a informação solicitada durante a troca de dados, sem se preocupar como esta informação foi obtida, sendo usada para receber a entrada de dados e apresentar informações ao usuário.

Cada uma delas, o *Model*, o *Controller* e a *View* executam apenas as funcionalidades que lhes foram definidas e nada mais além disso, tornando o código mais limpo, de fácil leitura e com alta coesão. A utilização deste padrão trás um grande benefício por isolar as regras de negócio da lógica de apresentação, proporcionando assim mais flexibilidade e oportunidades de reuso das classes.

2.5 React

ReactJs ou *React* é uma biblioteca usada para construir interfaces de usuário, permitindo a elaboração de aplicativos baseados em *JavaScript*. Os aplicativos *React* são executados no navegador e não precisam esperar por uma resposta do servidor [HostGator 2021].

Criado em 2011, o *React* surgiu com o objetivo de otimizar a atualização e a sincronização de atividades simultâneas, chamadas de estados, tendo principal foco inicial em *feed* de notícias de redes sociais, *chats*, *status*, listagem de contatos e outros [Roveda 2020].

Sendo uma linguagem com amplo suporte, de fácil utilização e aprendizado e capaz de entregar uma experiência amigável ao usuário, o *React* foi a linguagem escolhida para o desenvolvimento da interface gráfica da aplicação.

2.6 WebSocket

A especificação *WebSocket* define uma API que estabelece conexões entre um navegador da web e um servidor. Em outras palavras, há uma conexão persistente entre o cliente e o servidor e ambas as partes podem começar a enviar dados a qualquer momento [Ubl 2010].

Com *WebSocket*, pode-se transferir a quantidade de dados que for necessária, sem incorrer na sobrecarga associada às solicitações HTTP tradicionais. Os dados são transferidos por meio de um *WebSocket* como mensagens, e cada uma delas consiste em um ou mais *frames* (quadros) contendo os dados enviados (a carga útil) [Oliveira 2019].

A utilização deste protocolo esta diretamente relacionada ao problema que este trabalho se propôs a solucionar. A criação de um *chat* que possibilita a comunicação em tempo real entre todos os integrantes de um time, é um dos focos principais da solução. A possibilidade de enviar mensagens sem limites e a qualquer momento foi crucial para atingir este objetivo.

3. TRABALHOS RELACIONADOS

Neste capítulo, serão apresentadas pesquisas realizadas sobre ferramentas com propostas semelhantes disponíveis no mercado atual, a fim de apresentar algumas características dessas ferramentas. As buscas foram realizadas em pesquisas na internet e a partir delas foram localizadas diversas aplicações que se propõem a auxiliar na gestão de tarefas. A partir da pesquisa realizada, foram selecionadas três aplicações semelhantes à solução desenvolvida, sendo apresentadas a seguir.

3.1 Monday

Monday¹ é uma ferramenta de gerenciamento de projetos que permite que as organizações criem fluxos de trabalho para executar e gerenciar projetos de forma personalizada. Por ser multiplataforma, pode ser acessada tanto por computador ou dispositivo móvel como *smartphones* ou *tablets*. A ferramenta conta com diversas funcionalidades e configurações diferentes dependendo da necessidade do usuário, porém para o contexto do trabalho analisamos apenas o quadro *Kanban* disponível na plataforma devido a afinidade com a solução proposta.

Na Figura 3.1 podemos visualizar o quadro de tarefas dividido por grupos que podem ser editados de acordo a necessidade do usuário, onde também é possível definir prioridade para a tarefa e o *status* em que se encontra. O sistema também conta com a definição do colaborador que esta responsável pela tarefa e algumas informações referentes ao desenvolvimento como data de início e tempo de espera para finalização.

Figura 3.1 – Quadro de tarefas.

The screenshot shows a Monday.com Kanban board titled 'Planejamento'. It is divided into three sections, each with a table of tasks. The first section is 'Melhorias de Funcionalidades' (Improvements of Functionalities), the second is 'Bugs encontrados' (Bugs found), and the third is 'Débito técnico' (Technical debt). Each table has columns for Element, Reporter, Problem/Opport..., Priority, Status, Type, Work Start Date, Lead Time, WIP Indication, and Original Requi... (Original Request).

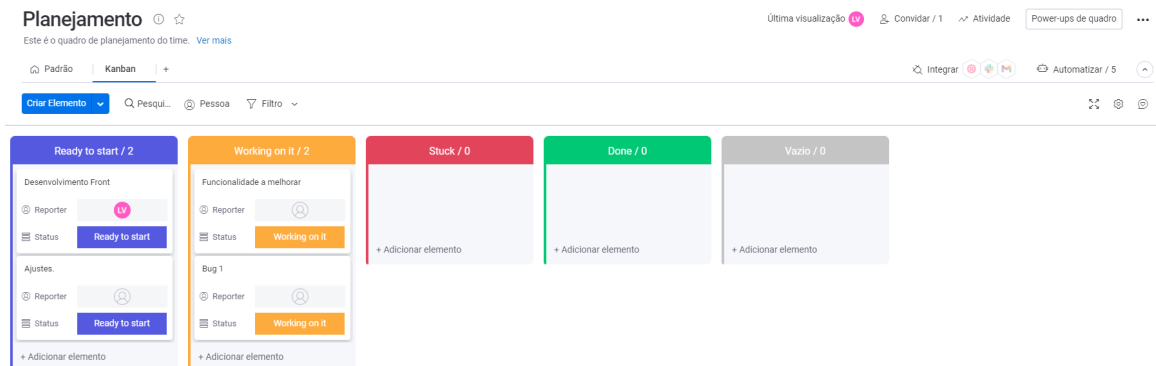
Elemento	Reporter	Problem/Opport...	Priority	Status	Type	Work Start Date	Lead Time	WIP Indication	Original Requi...
Funcionalidade a melhorar			High	Working on it	Feature Enhance...	jan 24	69	1	-
Desenvolvimento Front				Ready to start			69	-1	-
+ Adicionar Elemento									
+ Adicionar novo grupo									
Elemento	Reporter	Problem/Opport...	Priority	Status	Type	Work Start Date	Lead Time	WIP Indication	Original Requi...
Bug 1			Medium	Working on it	Bug	jan 22	71	1	Kanban Onbar...
+ Adicionar Elemento									
Elemento	Reporter	Problem/Opport...	Priority	Status	Type	Work Start Date	Lead Time	WIP Indication	Original Requi...
Ajustes			Low	Ready to start	Other		0	0	-
+ Adicionar Elemento									
+ Adicionar novo grupo									

Fonte (Próprio autor).

¹<https://monday.com>

Após realizar a organização destas tarefas, pode-se visualizá-las no quadro *Kanban* conforme a Figura 3.2.

Figura 3.2 – Quadro *Kanban* no Monday.

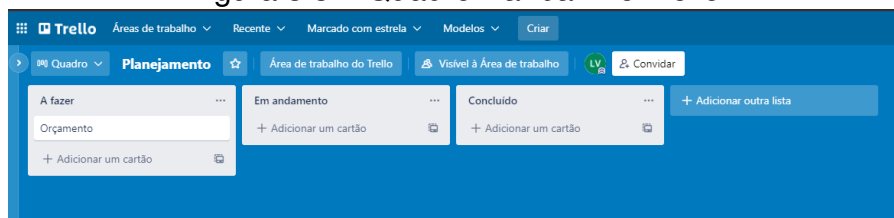


Fonte (Próprio autor).

3.2 Trello

Assim como a ferramenta descrita anteriormente, o Trello² é uma aplicação de gerenciamento de projetos, porém com foco maior na utilização do *Kanban* como metodologia de gestão, diferentemente de sua concorrente, Monday, que conta com muitas outras funcionalidades sendo o quadro *Kanban* apenas uma delas. Cada tarefa é representada por um cartão que pode ser movido, copiado ou removido dos murais de acordo com o progresso relativo ao desenvolvimento da mesma, conforme pode ser visto na Figura 3.3, contando também como a indicação do usuário responsável pela atividade.

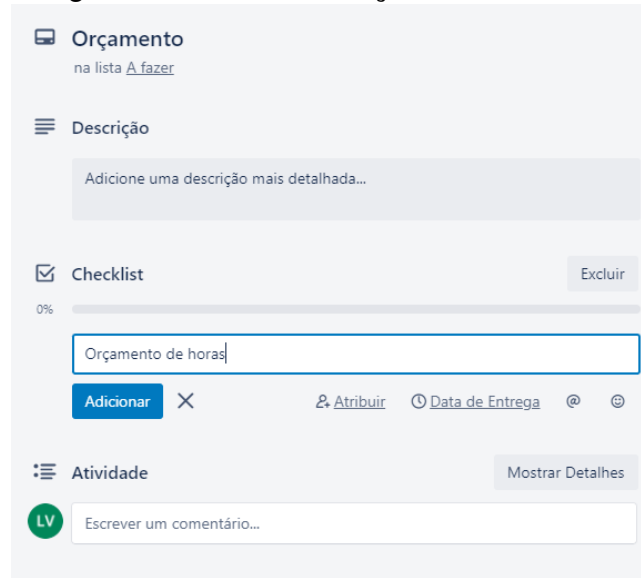
Figura 3.3 – Quadro *Kanban* no Trello.



Fonte (Próprio autor).

Outra funcionalidade interessante presente na ferramenta é a possibilidade de criar *checklists* necessárias para a finalização da tarefa, como demonstrado na Figura 3.4.

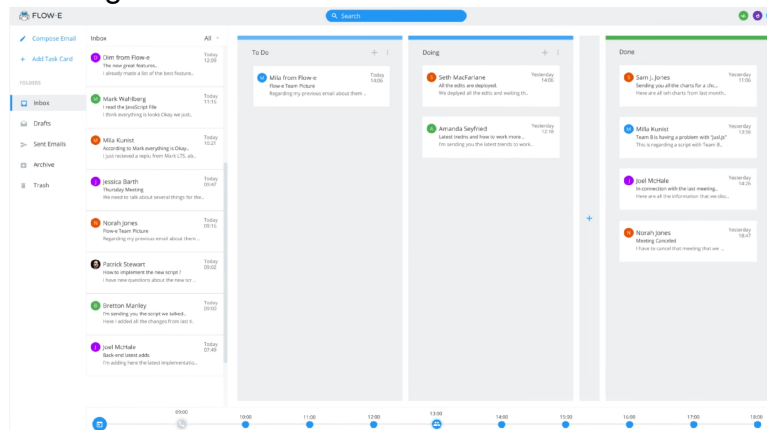
²<https://trello.com>

Figura 3.4 – Tela de adição de *checklists*.

Fonte (Próprio autor).

3.3 Flow-e

Diferentemente das outras ferramentas citadas neste documento, além do quadro *Kanban* que elas tem em comum, o Flow-E³ apresenta uma funcionalidade muito interessante, por meio de integração com o Outlook⁴, um serviço de *e-mails*, a ferramenta permite com que os usuários criem cartões com o conteúdo dos *e-mails* para utilizarem no quadro, podemos ver um exemplo a seguir na Figura 3.5

Figura 3.5 – Quadro com cartões de *e-mails*.

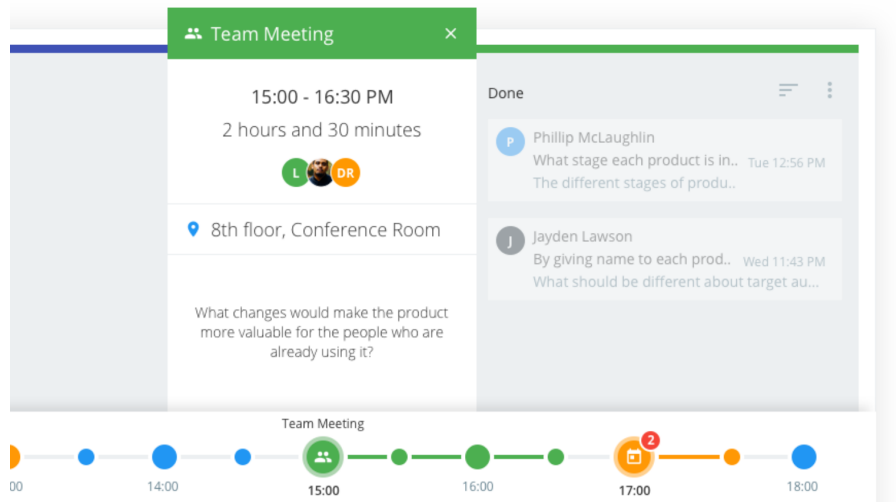
Fonte [Flow-e 2017].

³<https://flow-e.com>

⁴<https://outlook.live.com/>

Além desta funcionalidade anteriormente citada, conta com uma barra onde um cronograma de reuniões e lembretes podem ser visualizados de forma sincronizada e rápida com os a caixa de *e-mails* tornando assim o controle de agenda muito mais fluído, podemos visualizar na Figura 3.6 esta funcionalidade.

Figura 3.6 – Barra de cronograma.



Fonte [Flow-e 2017].

A seguir será mostrada uma tabela em que estão sendo comparadas as funcionalidades presentes na aplicação desenvolvida neste trabalho, denominada **Kanbro**, com as funcionalidades dos aplicativos apresentados nos trabalhos relacionados, como mostrado na Tabela 3.1.

Tabela 3.1 – Tabela comparativa de funcionalidades

	Monday	Trello	Flow-e	Kanbro
Quadro Kanban	X	X	X	X
Multiplataforma	X	X	X	X
Agrupamento por tipo de tarefa	X	-	-	-
Integração externa	X	X	X	-
Criação de blocos customizados	X	X	X	X
Definição de usuário responsável	X	X	X	X
Check-list em tarefas	-	X	-	-
Chat entre usuários	-	-	-	X

Fonte (Próprio autor).

Um ponto em comum entre todas estas ferramentas é o foco na colaboração e gerenciamento de tarefas e este foi o fator determinante para que elas servissem de base para a solução desenvolvida, que será explicada a seguir.

4. VISÃO GERAL DO SISTEMA

Neste capítulo, será apresentada a proposta do trabalho detalhando a motivação e os objetivos atingidos, funcionamento do sistema e os recursos utilizados.

4.1 Motivação e Objetivo Geral

Este trabalho foi motivado pelas dificuldades encontradas no sistema de trabalho remoto. Com a implementação do trabalho remoto proveniente do distanciamento social adotado como forma de contenção da disseminação do vírus Covid-19 que afetou o mundo inteiro, se tornando inclusive uma pandemia, diversas pessoas foram forçadas a trabalhar em casa. O sentimento de solidão, desamparo, falta de saber o que fazer ou até mesmo a falta do convívio social tornou o trabalho para os afetados por essa nova realidade algo estressante, exaustivo e melindroso,[Brito 2021]. Estes problemas afetam diretamente a produtividade dos profissionais, tendo em perspectiva esse cenário este trabalho apresenta uma aplicação *web* que entrega capacidade de gerenciamento de tarefas ao mesmo tempo que permite a qualquer integrante do projeto reportar de forma instantânea qualquer dificuldade que esteja tendo na resolução da tarefa.

4.2 Objetivos específicos

Os objetivos específicos do trabalho foram os seguintes:

- estudar e aperfeiçoar linguagens e tecnologias *web*;
- elaborar sistema de comunicação;
- melhorar e facilitar trabalho em equipe;
- modelar e desenvolver uma aplicação, denominada **Kanbro**, com as seguintes funcionalidades:
 - registro de usuário;
 - autenticação de usuário;
 - criação de times;
 - capacidade de adicionar usuários ao um time;
 - criação de um quadro para definição de tarefas referentes ao time;


- adição de cartões com tarefas;
- possibilidade de definir um responsável para cada tarefa;
- sistema de movimentação entre murais das tarefas no quadro;
- funcionalidade onde o usuário define que necessita de apoio na tarefa;
- criação do *chat* onde os usuários poderão interagir para sanar problemas;

A seguir, serão detalhadas as funcionalidades listadas anteriormente.

4.3 Funcionamento do sistema em detalhes

O sistema permite o cadastro de usuários, onde deverão ser informados um nome, um *e-mail* válido e uma senha de acesso para a realização do cadastro.

Figura 4.1 – Tela de Cadastro



A imagem mostra a tela de cadastro do sistema KANBRO!. À esquerda, há um banner azul com o logotipo 'KANBRO!' em letras brancas e azuis. À direita, o formulário de cadastro é exibido com o título 'Cadastro' em negrito. O formulário contém três campos de entrada: 'Nome', 'E-mail' e 'Senha', cada um com um ícone de lupa para busca. Abaixo dos campos, há um botão azul com o texto 'CADASTRAR' em branco.

Fonte (Próprio autor).

Após realizar o registro, os usuários terão a opção de utilizar as credenciais definidas durante o cadastro para acessar por meio de autenticação as funcionalidades que serão explicadas a seguir. Esta funcionalidade está demonstrada na Figura 4.2

Figura 4.2 – Tela de *Login*

The screenshot shows a login interface for KANBRO!. On the left, there is a blue vertical bar with the KANBRO! logo in white, bold, italicized font. On the right, the word "Login" is centered at the top. Below it, there are two input fields: "E-mail" and "Senha". Below the input fields, there are two buttons: "ENTRAR" and "CADASTRAR".

Fonte (Próprio autor).

Estando devidamente autenticado o usuário que não faz parte de nenhum time é redirecionado para uma página com duas opções, criar time e esperar convite, conforme pode ser visto na figura 4.3.

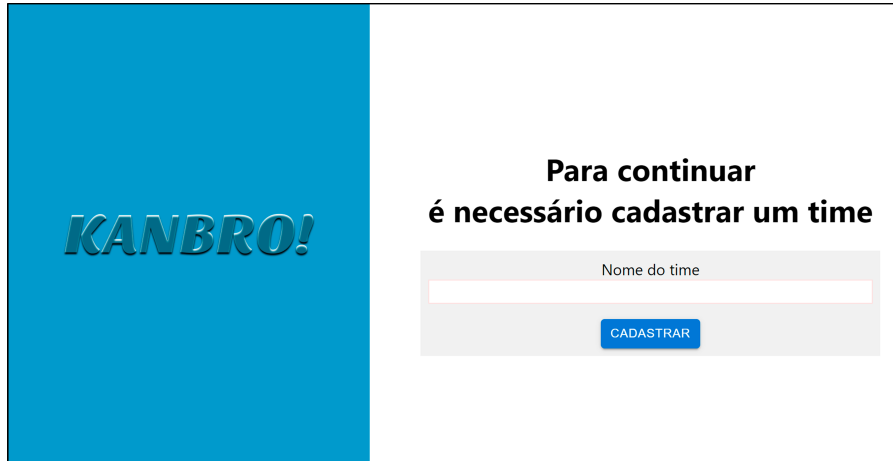
Figura 4.3 – Tela Gerenciamento de time

The screenshot shows a team management interface for KANBRO!. On the left, there is a blue vertical bar with the KANBRO! logo in white, bold, italicized font. On the right, the text "Você não faz parte de um time:" is centered. Below this text, there are two buttons: "CRIAR" and "ESPERAR CONVITE".

Fonte (Próprio autor).

Os usuários que desejam criar um time, devem selecionar a opção correspondente e informar um nome para o time que será criado. Esta funcionalidade esta demonstrada na Figura 4.4

Figura 4.4 – Tela Cadastro de Time

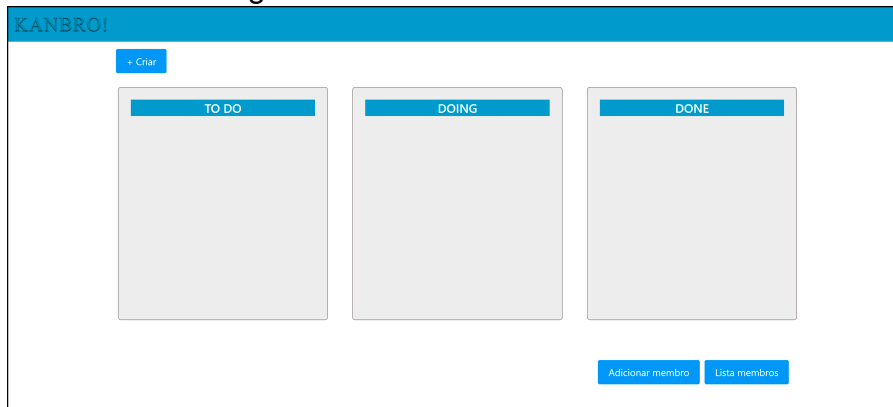


A tela de cadastro de time apresenta o logo 'KANBRO!' em um fundo azul à esquerda. À direita, há um formulário com o título 'Para continuar é necessário cadastrar um time'. O formulário contém um campo de texto rotulado 'Nome do time' e um botão azul 'CADASTRAR'.

Fonte (Próprio autor).

No momento da criação do time um quadro *Kanban* contendo os três murais básicos, citados no Capítulo 2.1 deste documento, estará disponível automaticamente.

Figura 4.5 – Tela Quadro Kanban

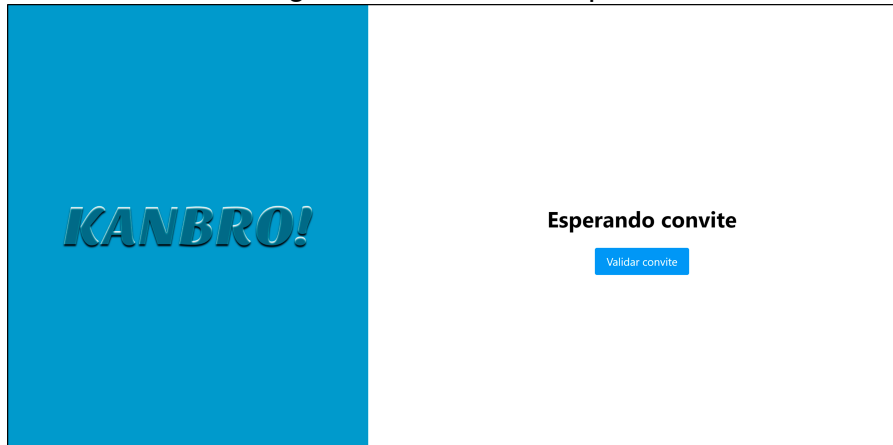


A tela do quadro Kanban possui um cabeçalho azul com o logo 'KANBRO!'. Abaixo dele, há um botão '+ Criar'. O quadro principal é dividido em três colunas: 'TO DO', 'DOING' e 'DONE', cada uma com um cabeçalho azul e um espaço cinza para as tarefas. Na base da tela, há dois botões: 'Adicionar membro' e 'Lista membros'.

Fonte (Próprio autor).

O usuário que decidir ingressar em um time já existente, deve selecionar a opção de espera sendo redirecionado a uma página onde fica esperando o convite. O sistema dispõe de uma opção para validar se o usuário foi adicionado a um time e caso tenha sido adicionado, o usuário é redirecionado para a página do quadro *Kanban* de seu novo time. Esta funcionalidade está demonstrada na Figura 4.6.

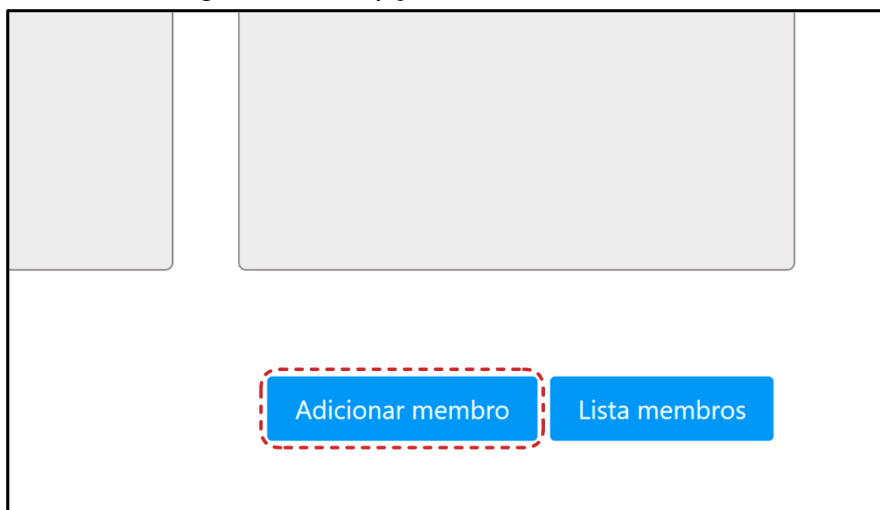
Figura 4.6 – Tela de Espera



Fonte (Próprio autor).

Para que seja possível adicionar novos membros ao time, possibilitando assim que recebam acesso ao quadro, uma opção de adição de membros deverá ser selecionada por meio da opção que esta disponível no quadro do time, conforme destaque em pontilhado na figura 4.7.

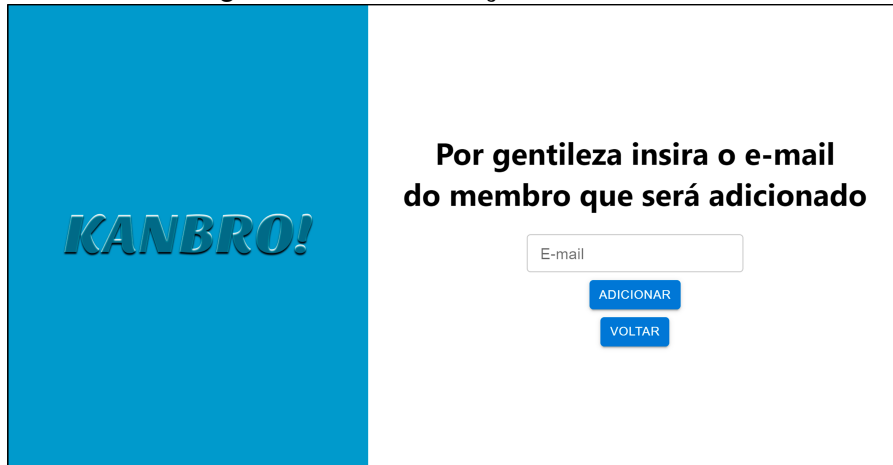
Figura 4.7 – Opção Adicionar Membros



Fonte (Próprio autor).

O usuário credenciado que já faz parte de um time informa o e-mail de outro usuário que está em espera. Ao adicionar um novo usuário ao time, o sistema registra e armazena de qual time este usuário convidado faz parte e libera acesso ao quadro Kanban e as tarefas do time. Esta funcionalidade esta demonstrada na Figura 4.8.

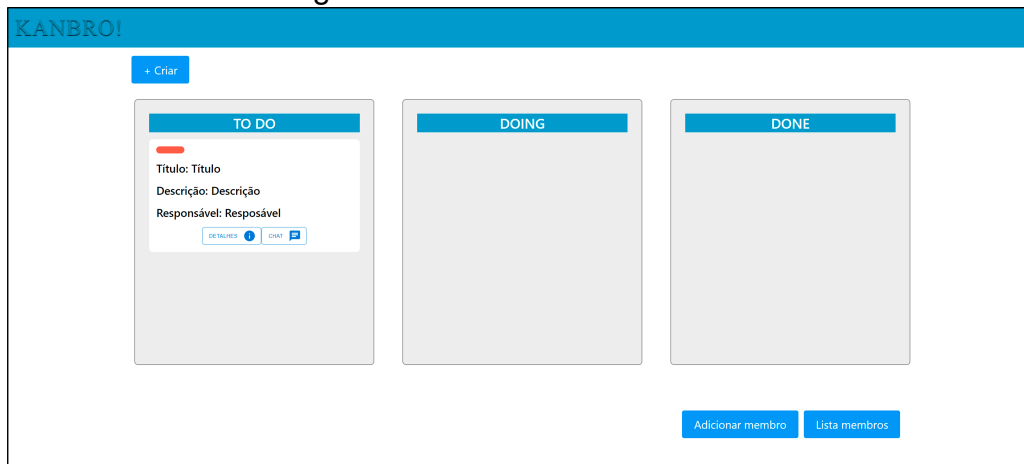
Figura 4.8 – Tela Adição de Membros



Fonte (Próprio autor).

Após os usuários serem devidamente adicionados ao time, todos poderão visualizar o quadro *Kanban* do time e as tarefas disponíveis. Esta funcionalidade esta demonstrada na Figura 4.9.

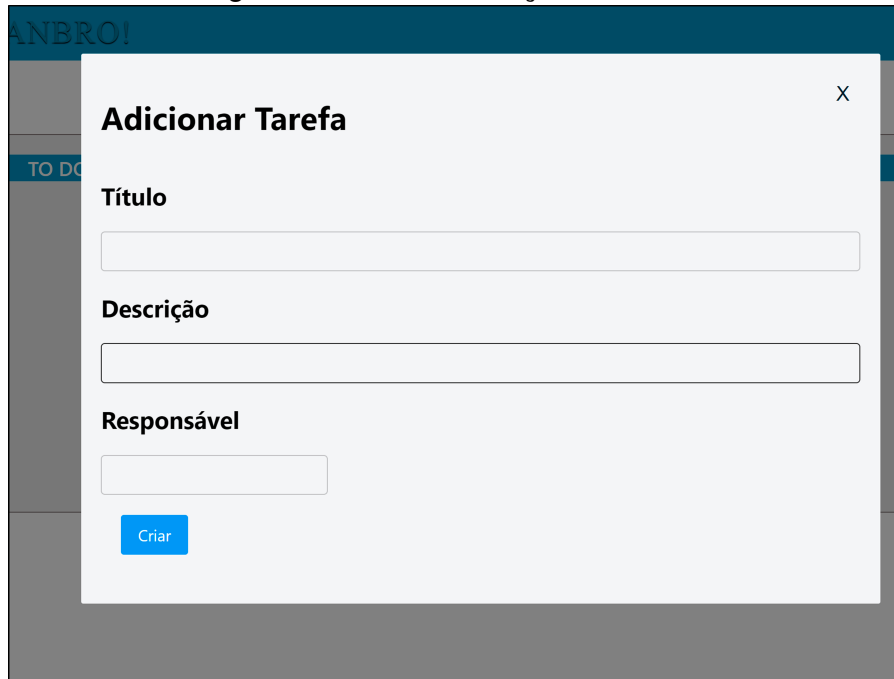
Figura 4.9 – Tela Quadro Kanban



Fonte (Próprio autor).

Utilizando a opção de criar, disponível no quadro, será possível adicionar uma nova tarefa ao quadro informando um título, uma descrição e qual membro do time será responsável por realizar aquela tarefa. Esta funcionalidade esta demonstrada na Figura 4.10.

Figura 4.10 – Tela Criação de Tarefa



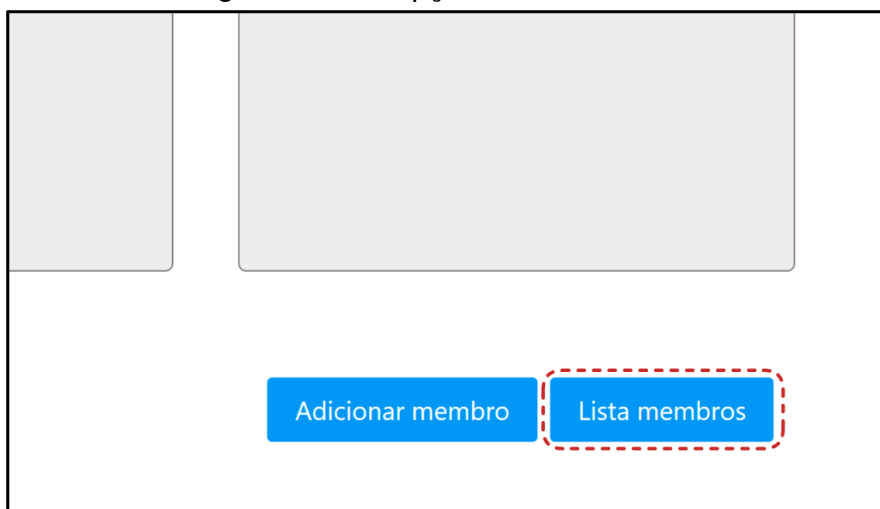
The image shows a modal window titled "Adicionar Tarefa" with a close button "X" in the top right corner. The form contains three input fields: "Título", "Descrição", and "Responsável". A blue button labeled "Criar" is positioned at the bottom left of the form.

Fonte (Próprio autor).

Todos os usuários membros do time serão capazes de criar os cartões do quadro, podendo movê-los entre os murais de acordo com o progresso do desenvolvimento de cada um. Novos cartões sempre serão adicionados inicialmente ao mural to do, que é utilizado para armazenar tarefas que estão esperando para serem realizadas.

Para visualizar os membros do time há disponível uma opção no quadro, onde quando selecionada o usuário é redirecionado a página com a lista de membros. O sistema informa detalhes de cada integrante. Esta funcionalidade esta demonstrada na Figura 4.11.

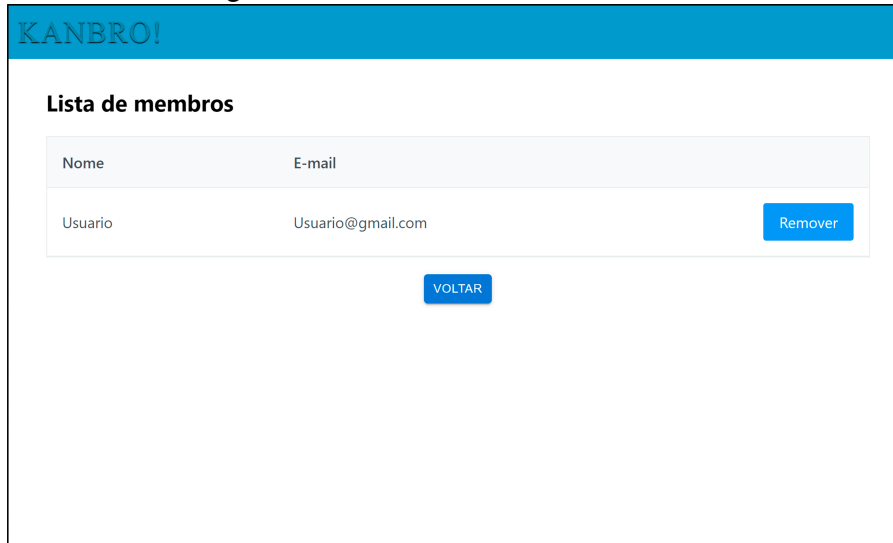
Figura 4.11 – Opção Listar Membros



Fonte (Próprio autor).

A lista de membros esta apresentada na Figura 4.12.

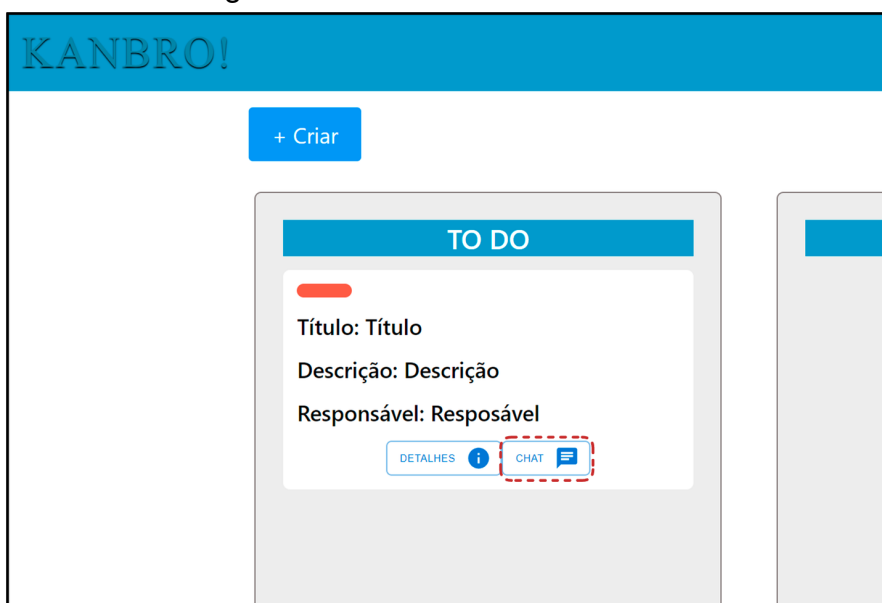
Figura 4.12 – Tela Lista de Membros



Fonte (Próprio autor).

No momento em que o usuário designado para a tarefa encontrar algum problema durante o progresso que o impeça de dar continuidade no desenvolvimento da atividade, como por exemplo, necessitar de ajuda tecnicamente, estiver com alguma dúvida referente aos requisitos da tarefa, ou qualquer outra questão que necessite ser resolvida, é possível acionar a opção de chat no cartão. Esta funcionalidade esta destacada em pontilhado na Figura 4.13.

Figura 4.13 – Criar *Chat* em Cartão

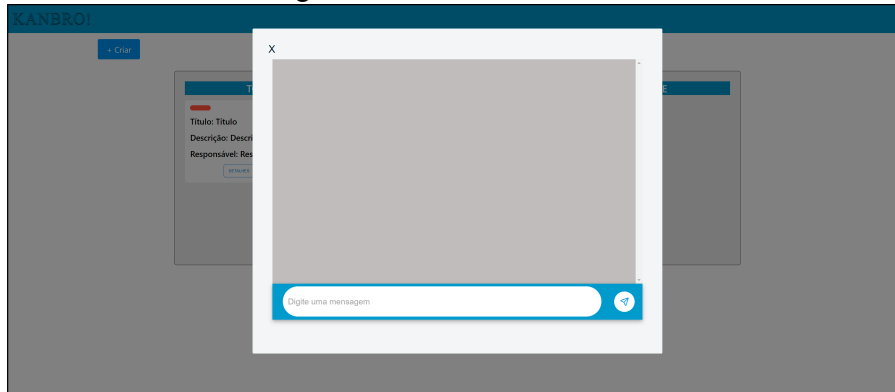


Fonte (Próprio autor).

Após a opção de *chat* ser ativada em um cartão o botão terá sua cor modificada para vermelho, indicando assim que há problemas no andamento da atividade relacionada ao cartão, que necessitam de atenção pelo time.

O sistema então habilitará, no cartão onde foi solicitado apoio, um *chat* onde o usuário com dificuldade poderá relatar o problema encontrado. Esta funcionalidade esta demonstrada na Figura 4.14.

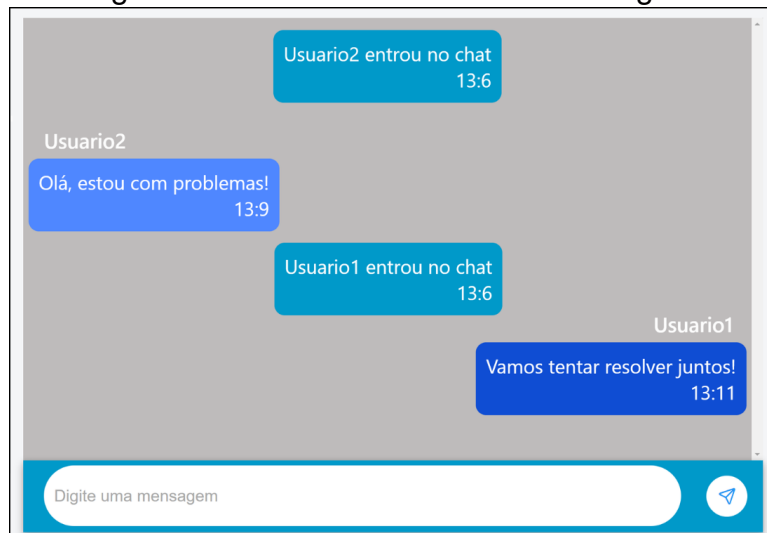
Figura 4.14 – Tela de *Chat*



Fonte (Próprio autor).

Todos os membros do time serão capazes de navegar até este *chat*, onde será possível trocar mensagens com todos a fim de solucionar este problema. Esta funcionalidade esta demonstrada na Figura 4.15.

Figura 4.15 – Tela de *Chat* com Mensagens



Fonte (Próprio autor).

Após a resolução do problema será possível voltar à conversa para revisar as sugestões de apoio ou visualizar o histórico de problemas referente à aquela tarefa.

5. IMPLEMENTAÇÃO

Seguindo os requisitos necessários em uma arquitetura de aplicações *web*, o desenvolvimento do sistema Kanbro foi dividido em duas camadas, cliente e servidor. A comunicação entre estes dois agentes torna possível o processamento e armazenamento das informações e o acesso do usuário por meio de uma interface gráfica a dados e funcionalidades que compõem a aplicação como um todo. A seguir serão descritos quais as responsabilidades de cada camada e de uma forma mais aprofundada e quais ferramentas foram utilizadas para tornar este produto realidade.

5.1 Servidor

Um Servidor tem como função receber requisições e a partir delas oferecer respostas para o cliente. Ele encapsula a lógica de negócios da aplicação, controlando as transações e coordenando as respostas na implementação de suas operações.

O servidor foi desenvolvido na linguagem Java em conjunto com o *framework Spring*¹. O *framework Spring* tem como uma das suas principais características a disponibilização de componentes pré-configurados, diminuindo assim o esforço no momento da implantação. Com a utilização das configurações disponibilizadas pelo uso do *framework* é possível garantir uma clara separação de responsabilidades entre todas as partes do sistema que compõem o servidor. Com base nisso foi possível segmentar nas seguintes camadas:

- *Controller* ou Controlador: O controlador é a classe responsável por disponibilizar recursos ao cliente por meio de URLs, podendo informar dados para recuperar alguma informação ou enviar dados em formato JSON, explicado anteriormente na Seção 2.3, para que sejam armazenados no sistema;
- DTO ou *Objeto de Transferência de Dados*: É um padrão utilizado para transporte de dados entre diferentes componentes de um sistema. Geralmente tem atributos relacionados ao modelo de uma classe, que será melhor detalhado adiante, porém filtra e contém apenas aquilo que é necessário para um determinado processo;
- *Model* ou Modelo: É o componente do sistema responsável por conter atributos ou conjunto de informações relacionados a uma necessidade de negócio. Também é utilizado como uma representação do banco de dados, sendo necessário que seu uso seja utilizado no momento de recuperação ou armazenamento de informações;

¹<https://spring.io/>

- *Mapper* ou Mapeador: No sistema utilizamos um DTO para receber dados ou requisições do cliente, porém no momento em que estes dados são recuperados ou armazenados é necessário que sejam convertidos em seus respectivos modelos para que possam ser processados. O mapeador serve exatamente para este propósito, converter um DTO em um modelo e vice-versa;
- *Service* ou Serviço: Esta camada tem como principal responsabilidade cuidar das regras de negócio e possíveis validações que se façam necessárias. Por fim, realiza a comunicação com o repositório para consultas e armazenamento na base de dados;
- *Repository* ou Repositório: É responsável por acessar diretamente a base de dados por meio direto de recuperação ou inserção de dados;
- *Security* ou Segurança: Nesta camada ficam inseridas as classes de configuração da autenticação do sistema que será abordado de forma detalhada mais adiante.

5.1.1 Persistência de dados

A modelagem da base de dados do sistema foi desenvolvida no modelo relacional, onde tabelas, utilizadas para armazenar os dados, tem sempre ao menos uma informação em comum. Desta forma, pode ser realizada uma relação entre as tabelas permitindo assim que dados distribuídos possam ser acessados de acordo com a necessidade.

A forma como a aplicação faz o gerenciamento desta base de dados foi realizado por meio da utilização do *Hibernate*², que é uma ferramenta de mapeamento objeto-relacional em conjunto com a API JPA. O *Hibernate* é responsável por realizar a correspondência entre objetos Java e uma tabela de banco de dados, tendo como principais funções executar operações de criação, exclusão, consulta e atualização no banco de dados.

5.1.2 Autenticação

A fim de identificar o usuário durante o uso da aplicação e permitir que ela possa ser utilizada apenas por usuários credenciados, foi implementado um sistema de autenticação. Para que fosse possível configurar o sistema de autenticação foi utilizado o *Spring Security*³, que é um dos recursos disponíveis no *framework Spring*.

Pensando na segurança do usuário e aproveitando as funcionalidades contidas no *framework*, foi realizada uma configuração para que a senha do usuário fosse salva de forma criptografada, conforme a Figura 5.1:

²<https://hibernate.org/>

³<https://spring.io/projects/spring-security>

Figura 5.1 – Senha criptografada

EMAIL	NOME	SENHA
usuario@gmail.com	Usuario	\$2a\$10\$2tMMdtuWnlfZs95mbY4D2Ov7.VywHzHWry8AXLiPzsCUj.Z7krrhK

Fonte (Próprio autor).

No momento em que o usuário informa suas credenciais para se autenticar na aplicação, o próprio *framework* é responsável por interpretar a senha informada com a informação salva de forma criptografada.

5.2 Cliente

O cliente é responsável por apresentar a interface gráfica da aplicação, solicitar dados para o servidor e gerenciar toda a lógica relacionada as funcionalidades que o usuário pode utilizar. *React*, que é uma biblioteca baseada em *Javascript*, foi escolhida para implementação do cliente por ser de fácil utilização e dispor de diversos recursos, sendo alguns deles:

- **Reutilização de Componentes:** O *React* possibilita a criação de componentes que podem ser reutilizados de acordo com a necessidade do desenvolvedor, fazendo com que o mesmo componente possa ser usado em mais de um lugar sem que necessite desenvolver dois componentes separadas, ou seja um componente genérico;
- **Componentes dinâmicos:** Os objetos contidos em uma página podem ser dinâmicos, alterando seu formato ou conteúdo, baseado nos dados informados no momento da utilização;
- **Props:** Considerado como propriedades que cada componente possui, podendo ser um ou vários, são valores informados a cada componente durante a utilização da aplicação que não podem ser alterados pelo usuário e tem como principal funcionalidade transferir dados entre componentes;
- **States:** São estados dos valores presentes na aplicação que podem ser modificados mediante ação do usuário em tempo real, possibilitando assim uma rápida visualização em objetos na interface durante o uso.

5.2.1 Requisições ao Cliente via HTTP

A aplicação necessita de fornecimento de dados para seu funcionamento e eventualmente realizar o armazenamento de dados pertinentes ao usuário ou de negócio, que

é responsabilidade do cliente. Para possibilitar a comunicação entre o cliente e o servidor, foi utilizada uma biblioteca chamada *Axios*⁴, que é responsável por realizar requisições ao cliente e acessar seus recursos.

A escolha para a utilização da biblioteca *Axios* se deu principalmente por possuir nomes de funções que correspondem com os métodos HTTP, conseguindo operar de forma fácil com padrão *JSON* e acessando os recursos do cliente apenas informando sua URL. Ao fazer uma solicitação ao cliente via protocolo HTTP, deve ser informado o tipo de operação que está sendo realizada e, caso estejam presentes, os dados via *JSON*. Os métodos utilizados na implementação foram os seguintes:

- *GET*: É utilizado para recuperar dados do cliente. Também possibilita a busca de um recurso específico informando o dado pertinente na URL, como um filtro;
- *POST*: É utilizado para enviar dados ao servidor. Na maioria dos casos, é utilizado para criar um novo registro, porém também possibilita atualizar alguma informação referente a um recurso.

5.2.2 Local Storage

Durante a utilização do sistema por parte do usuários, diversas informações são necessárias para realizar requisições ao cliente, como por exemplo, de qual time o usuário faz parte. A forma como esse problema foi contornado se deu pela utilização do *Local Storage*, que é um compartimento de memória disponível nos navegadores e pode ser utilizado para armazenar dados não sensíveis do usuário. No momento da autenticação na aplicação, os dados necessários para a realização das requisições ficam registrados neste compartimento e são acessados no momento em que precisam serem utilizados por ela.

5.2.3 Chat com WebSocket

Para tornar possível a comunicação em tempo real entre integrantes de um mesmo time foi implementado um chat na aplicação. Esta implementação foi realizada por meio de *WebSockets*.

WebSocket é um protocolo de comunicação bi-direcional que permite a comunicação entre cliente e servidor em tempo real. Diferentemente do protocolo HTTP que abre uma nova conexão para cada requisição e encerra a conexão com o servidor após receber

⁴<https://axios-http.com/>

uma resposta, o *WebSocket* mantém essa conexão aberta até que ela seja finalizada pelo cliente.

Para armazenar os dados referentes as mensagens enviadas no chat, foram utilizados os seguintes atributos:

- **Usuário:** Informação referente ao usuário que está mandando a mensagem;
- **Sala:** Armazena de qual a sala as mensagens fazem parte e utiliza o *id* do cartão do quadro *Kanban* como identificador único da sala;
- **Tipo da mensagem:** Podem ser de dois tipos, cliente ou servidor. Cliente, quando é enviada por um usuário. Servidor, quando é gerado a partir de um registro de atividades no *chat*;
- **Conteúdo:** É utilizada de duas formas, na primeira sendo o texto da mensagem em si que foi enviada por um usuário e na segunda, quando gerada pelo servidor, armazenando informações referentes a entrada e saída dos usuários no *chat*.

Utilizando esses atributos, no momento em que um usuário acessa e envia mensagens em um *chat*, o servidor é capaz de armazenar quem enviou e para qual *chat* as mensagens foram enviadas. Quando outros usuários acessam o *chat* que já contém um histórico de mensagens, a listagem de mensagens é exibida baseado no identificador único gerado para cada *chat*.

5.2.4 Context API

O acesso às funcionalidades da aplicação só pode ser realizado por um usuário devidamente autenticado. Durante o uso do sistema, esta informação deve estar disponível e carregada em tempo real para validação de qual usuário está autenticado. Este controle foi realizado por meio de outro recurso disponível no *React*, chamado de *Context API*, que consiste basicamente em um gerenciador de estados globais dentro da aplicação.

Após realizar a autenticação, o sistema guarda as informações do usuário e as mantém em constante compartilhamento por toda a aplicação. Desta forma, o usuário pode acessar todas as funcionalidades sem que o sistema perca a identificação de quem está utilizando-o.

6. CONCLUSÃO

Este trabalho teve como objetivo criar uma aplicação *web* capaz de facilitar a colaboração e a comunicação entre profissionais que trabalham em suas empresas remotamente. A forma como se propôs a resolver estes problemas foi por meio do desenvolvimento de um quadro *kanban* que contém um *chat* integrado em seus cartões.

Durante o desenvolvimento do sistema *Kanbro* diversas dificuldades foram sendo encontradas, dentre elas o tamanho na aplicação que havia sido previamente pensado e as tecnologias escolhidas para a sua implementação. Inicialmente o projeto contava com mais funcionalidades, mas devido à complexidade de sua implementação, foi necessário reduzir o escopo mantendo apenas aquelas necessárias para atingir o objetivo. A maior dificuldade encontrada foi que ao longo do desenvolvimento se fez necessário estudar diversas tecnologias, como uso o avançado do *React* por exemplo, fazendo assim com que o tempo para implementar a solução completa fosse maior do que o inicialmente planejado.

O conhecimento adquirido durante a produção do sistema se fez muito importante, pois a cada problema novo que surgia o aprendizado acumulado possibilitava que a solução fosse encontrada de forma mais rápida. A decisão de criar uma arquitetura com responsabilidades bem definidas e separadas auxiliou muito neste processo, pois nos momentos em que foi necessário encontrar erros, a separação de cada funcionalidade tornou mais direta a análise do problema.

Ao longo do desenvolvimento diversas ideias e ferramentas precisaram ser descartadas ou substituídas, pois a forma como funcionavam era descoberta durante a implementação e muitas vezes não atendia as necessidades ou não se relacionavam com a arquitetura geral do projeto.

Para que fosse possível cumprir os prazos e entregar a solução proposta, foi necessário abrir mão de algumas funcionalidades, porém o resultado obtido foi muito satisfatório. Mesmo tendo um número menor de funcionalidades do que foi inicialmente projetado, a possibilidade dos integrantes de um time realizarem uma comunicação em tempo real, facilitando a resolução de problemas, foi atingida na solução entregue, atendendo aos requisitos mínimos necessários para resolver o problema proposto.

6.1 Trabalhos Futuros

Levando em consideração que todo o sistema tem a possibilidade de melhorar, a seguir estão listadas algumas melhorias como trabalhos futuros:

- permitir que novos estados sejam criados no quadro *Kanban*, possibilitando que uma personalização maior seja viável e auxiliando no controle de estado das atividades;
- adicionar uma categoria em cada cartão, fazendo com que seja possível analisar em qual tipo de atividade o time tem mais dificuldade;
- implementar um sistema de notificações, auxiliando o time no atendimento de novas dúvidas ou solicitações de auxílio nos *chats*;
- disponibilizar uma barra de pesquisa para facilitar encontrar cartões antigos;
- possibilitar o uso de imagens no cadastro de cada usuário para facilitar a identificação.

REFERÊNCIAS BIBLIOGRÁFICAS

- [Brito 2021] Brito, S. (2021). Home office: o desafio de trabalhar distante da empresa. Disponível em: <<https://veja.abril.com.br/cultura/home-office-como-administrar-os-desafios-longe-do-escritorio/>>.
- [Campomori 2016] Campomori, C. (2016). Rest não é simplesmente retornar json: indo além com apis rest. Disponível em: <<https://www.treinaweb.com.br/blog/rest-nao-e-simplesmente-retornar-json-indo-alem-com-apis-rest>>.
- [COSTA 2008] COSTA, H. G. (2008). *PCP - Sistema de Controle da Produção*. In: LUSTOSA, L. et al. *Planejamento e Controle da Produção*. Elsevier.
- [de Souza 2020] de Souza, I. (2020). Saiba o que é rest (representational state transfer) e como usá-lo neste tutorial. Disponível em: <<https://rockcontent.com/br/blog/rest/>>.
- [Flow-e 2017] Flow-e (2017). Flow-e - make your email great again! Disponível em: <https://www.youtube.com/watch?v=DBCNxzDLf_g&ab_channel=Flow-e/>.
- [Garcia 2021] Garcia, G. (2021). Saiba o que protocolo http e conheça seus componentes e vulnerabilidades. Disponível em: <<https://mercadoonlinedigital.com/blog/protocolo-http/>>.
- [HostGator 2021] HostGator (2021). React: o que é e como usar a ferramenta? Disponível em: <<https://www.hostgator.com.br/blog/react-como-usar-a-ferramenta/>>.
- [Macêdo 2017] Macêdo, D. (2017). Entendendo as aplicações web. Disponível em: <<https://www.diegomacedo.com.br/entendendo-as-aplicacoes-web/>>.
- [Nawaz 2021] Nawaz, S. (2021). 4 motivos para o baixo desempenho no home office e como superá-los. Disponível em: <<https://forbes.com.br/carreira/2021/06/4-motivos-para-o-baixo-desempenho-no-home-office-e-como-supera-los/>>.
- [Noletto 2020] Noletto, C. (2020). Aplicações web: entenda o que são e como funcionam! Disponível em: <<https://blog.betrybe.com/desenvolvimento-web/aplicacoes-web/>>.
- [Oliveira 2019] Oliveira, R. (2019). Como o javascript funciona: Aprofundando em websockets e http/2 com sse + como escolher o caminho certo. Disponível em: <<https://medium.com/reactbrasil/como-o-javascript-funciona-aprofundando-em-websockets-e-http-2-com-sse-como-escolher-o-caminho-d4639995ef85>>.
- [RedHat 2020] RedHat (2020). O que é metodologia Ágil. Disponível em: <<https://www.redhat.com/pt-br/devops/what-is-agile-methodology>>.
- [Roveda 2020] Roveda, U. (2020). React: O que é, como funciona e porque usar e como aprender. Disponível em: <<https://kenzie.com.br/blog/react/>>.

[Silva 2018] Silva, R. (2018). Escritório: História e a sua relação com produtividade. Disponível em: <https://www.catho.com.br/carreira-sucesso/colunistas/gestao-rh/escritorio-historia-e-a-sua-relacao-com-productividade/>.

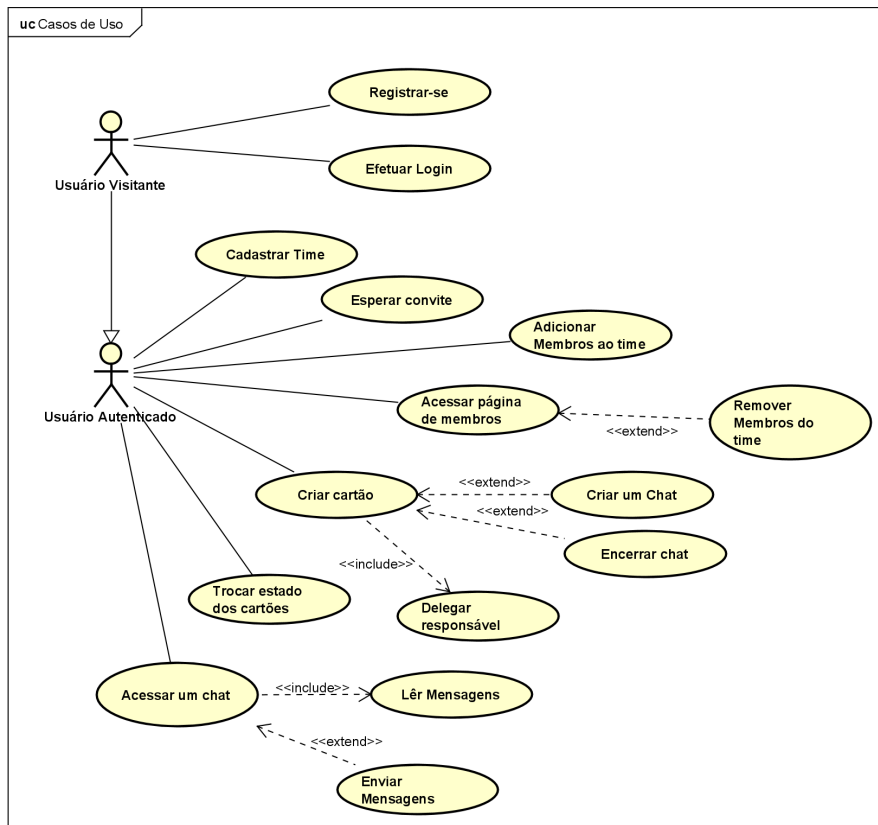
[Silveira 2019] Silveira, D. (2019). Home office bateu recorde no brasil em 2018, diz ibge. Disponível em: <https://g1.globo.com/economia/concursos-e-emprego/noticia/2019/12/18/home-office-bateu-recorde-no-brasil-em-2018-diz-ibge.ghtml>.

[Ubl 2010] Ubl, M. (2010). Apresentando websockets: trazendo soquetes para a web Disponível em: <https://www.html5rocks.com/pt/tutorials/websockets/basics/>.

APÊNDICE A – MODELAGEM

Neste apêndice, serão apresentados os casos de uso (UCs) da aplicação, ilustrado pelo diagrama de casos de uso da Figura 4.1 e a modelagem do sistema representado pelo diagrama de classes apresentado na Figura 4.14, contendo suas devidas descrições.

Figura A.1 – Diagrama de casos de uso.



Fonte (Próprio autor).

UC01 – Registrar-se: O usuário visitante informa um *e-mail*, seu nome de usuário e uma senha ao sistema para realizar seu registro. Após selecionar a opção de criar conta, estes dados são armazenados e podem ser utilizados como credenciais de *login*.

UC02 – Efetuar login: O usuário visitante informa seu *e-mail* e senha de acesso previamente cadastrados. Após o sistema validar as credenciais, é liberado o acesso ao sistema.

UC03 – Cadastrar time: O usuário informa um nome para o time que deseja criar, o sistema cadastra um novo time e disponibiliza um novo quadro *Kanban* vazio para o time.

UC04 – Esperar convite: O usuário que não deseja criar um time e sim receber um convite, é redirecionado a uma página onde fica esperando o convite para um time que já existe. O sistema verifica periodicamente por convites e caso o usuário seja adicionado a um time, o usuário é redirecionado para a página do quadro *Kanban* de seu novo time.

UC05 – Adicionar membros ao time: O usuário credenciado que já faz parte de um time informa o *e-mail* de outro usuário que está em espera. Ao adicionar um novo usuário ao time, o sistema registra e armazena de qual time este usuário convidado faz parte e libera acesso ao quadro *Kanban* e as tarefas do time.

UC06 – Acessar página de membros: O usuário seleciona a opção de visualizar membros, sendo então redirecionado à página de membros do time. O sistema informa a lista dos membros, possibilitando visualizar os detalhes de cada integrante.

UC07 – Remover membros do time: Após a opção de remoção de um membro ser disparada, que está disponível na tela de lista de membros de um determinado quadro *Kanban* do time, o sistema remove do usuário a informação de qual time ele faz parte e o mesmo é removido do time.

UC08 – Criar cartão: No quadro *Kanban* o usuário adiciona um novo cartão de tarefa, informando um título, uma descrição e definindo um membro do time como responsável por aquela atividade descrita no cartão. O sistema cria e armazena o novo cartão com as informações disponibilizadas e com estado inicial *to do*.

UC09 – Criar um chat: No cartão desejado o usuário seleciona a opção de criar um *chat*. O sistema cadastra e inicia um novo *chat* com estado “aberto” referente ao cartão informado.

UC10 – Acessar um chat: O usuário seleciona a opção de acessar um *chat* já existente no cartão desejado e é redirecionado pelo sistema ao chat referente ao cartão. Ao acessar o *chat*, o usuário pode ler as mensagens anteriormente enviadas por outros membros do time.

UC11 – Encerrar chat: O usuário seleciona a opção de encerrar o *chat*. O sistema modifica o estado do *chat* para “finalizado” impossibilitando que novas mensagens sejam enviadas, apenas permitindo visualização do histórico de mensagens.

UC12 – Enviar mensagens: Estando no *chat*, o usuário envia novas mensagens que são cadastradas pelo sistema no histórico do *chat* e se tornam visíveis para os outros usuários.

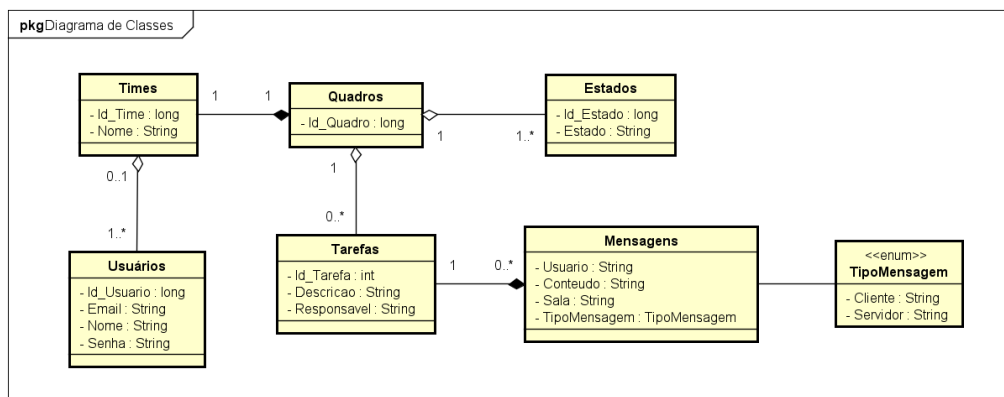
UC13 – Trocar estado dos cartões: O usuário arrasta o cartão desejado para um outro estado dentre os disponíveis no quadro *Kanban*. O sistema atualiza o estado do cartão para este novo estado.

Na Figura 4.14 esta apresentado o diagrama de classes da aplicação. Este diagrama foi construído com o intuito de demonstrar a estrutura de banco de dados necessária para comportar a arquitetura de persistência da aplicação.

A seguir serão descritas as responsabilidades de cada classe com o objetivo de entender como os dados serão organizados.

Usuários: Tabela responsável por armazenar as informações de cadastro dos usuários, contendo as colunas:

Figura A.2 – Diagrama de Classes



Fonte (Próprio autor).

- **Id_Usuario (Chave)**: Identificador único gerado automaticamente pelo sistema para identificação interna de cada usuário;
- **Email**: *E-mail* informado no momento de criação da conta, utilizado posteriormente para *login* na aplicação;
- **Nome**: Nome escolhido pelo usuário para ser utilizado na aplicação;
- **Senha**: Conjunto de caracteres informado no momento de criação da conta, armazenados de forma criptografada, utilizado posteriormente para *login* na aplicação.

Times: Tabela responsável por armazenar todos os times cadastrados no sistema. Faz referência a tabela de usuários, utilizando a informação de **Id_Usuario** como índice de uma lista de usuários cadastrados no time. Contém as colunas:

- **Id_Time (Chave)**: Identificador único gerado automaticamente pelo sistema para identificação interna de cada time;
- **Nome**: Nome informado no momento da criação do time, não podendo conter valores repetidos.

Quadros: Tabela responsável por armazenar o quadro de cada time, é automaticamente gerado no momento de criação de cada time. Faz referência a tabela de times, recebendo sua chave como chave estrangeira, permitindo que apenas seja cadastrado um quadro para cada time. Contém a coluna:

- **Id_Quadro (Chave)**: Identificador único gerado automaticamente pelo sistema para identificação interna de cada quadro;

Estados: Tabela responsável por armazenar os estados disponíveis em cada quadro. Sempre que um quadro é criado, são gerados 3 registros iniciais sendo eles, *To do*, *Doing* e *Done*. Faz referência a tabela de quadros. A medida que novos estados são criados, armazena em conjunto a informação da chave do quadro e o novo estado criado, mantendo a relação de quais estados fazem parte do quadro.

- **Id_Estado (Chave):** Identificador único gerado automaticamente pelo sistema para identificação interna de cada estado;
- **Estado:** Nome de cada estado disponível no quadro.

Tarefas: Tabela responsável por armazenar as tarefas de cada quadro. Faz referência a tabela de quadros, recebendo a chave do quadro para gerenciar de qual quadro cada tarefa faz parte.

- **Id_Tarefa (Chave):** Identificador único gerado automaticamente pelo sistema para identificação interna de cada tarefa;
- **Descrição:** Descrição utilizada para informar qual atividade deve ser realizada na tarefa.
- **Responsável:** Armazena qual é o usuário responsável por determinada tarefa.

Mensagens: Tabela responsável por armazenar todas as mensagens enviadas na aplicação. Faz referência a tabela de tarefas, recebendo sua chave para a identificação de qual tarefa as mensagens fazem parte. Contém também uma referência para a tabela de *chat*, de onde recebe sua chave para identificar de qual *chat* as mensagens fazem parte.

- **Usuário:** Responsável por armazenar o nome do usuário que enviou a mensagem;
- **Conteúdo:** Mensagem enviada pelo usuário ao *chat*;
- **Sala:** Armazena qual a sala que as mensagens percentem;
- **TipoMensagem:** Enumeração com valores fixos.

TipoMensagem: Objeto do tipo enumeração, utilizado para permitir que os tipos de mensagens possam ser apenas valores fixos pré-definidos, sendo eles:

- **Cliente:** Utilizado para mensagens provenientes do cliente.
- **Servidor:** Mensagem gerada pelo servidor.