CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE FRANCA

“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GABRIEL DA SILVA JESUS

GABRIEL LEMOS SIQUEIRA PEREIRA

TCG MANAGER

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me .Leonardo Henrique Raiz

FRANCA/SP

2023

TCG Manager:

plataforma de distribuição de informações sobre Pokémon TCG

Gabriel da Silva Jesus

Gabriel Lemos Siqueira Pereira

Resumo

A aplicação web aqui desenvolvida foi pensada para suprir a necessidade de uma plataforma informativa para o jogo Pokémon Estampas Ilustradas, focando principalmente em seu ambiente competitivo. O problema foi levantado analisando a quantidade de informações dispersas sobre o jogo que poderiam ser centralizadas e melhor distribuídas, aumentando o interesse e a imersão de novos jogadores dentro do ambiente competitivo. Por mais que seja inusitado, jogos de cartas colecionáveis, comumente conhecidos como jogos de Trading Card Games, movimentam uma alta quantidade monetária em todo mundo. O jogo aqui citado, na primeira metade de 2021, registrou um aumento de 536% no valor bruto de mercadoria no e-commerce internacional eBay. Tendo isto em vista, o setor em questão é sim um bom mercado para se investir visto seu crescimento exponencial ao longo dos anos.

Palavras-chave: Aplicação. Ambiente competitivo. Investir. Jogo. Necessidade. Plataforma informativa.

Abstract

The developed web application was conceived to address the need for an informative platform for the Pokémon Trading Card Game, focusing primarily on its competitive environment. The issue was identified by analyzing the abundance of scattered information about the game that could be centralized and better distributed, thereby enhancing the interest and immersion of new players within the competitive setting. Although it may seem unusual, collectible card games, commonly known as Trading Card Games, generate a substantial amount of revenue worldwide. The aforementioned game, in the first half of 2021, recorded a 536% increase in gross merchandise value on the international eBay e-commerce platform. With this in mind, it is evident that the sector in question is indeed a promising market for investment, given its exponential growth over the years.

Keywords: *Application. Competitive environment. Game. Informative platform. Investment. Need.*

1 Introdução

O desenvolvimento de aplicações web para diversas finalidades tem sido uma tendência crescente ao longo dos anos, tornando-se cada vez mais comum em uma variedade de domínios, desde transmissões de música e vídeo até sistemas

empresariais e fóruns, entre outras utilidades. Dentro desse amplo espectro de aplicações, destacam-se as plataformas de discussão online, que possibilitam o acesso global a uma riqueza de informações e a capacidade de compartilhá-las com uma audiência diversificada.

No âmbito das plataformas de discussão online, existe um potencial significativo para sua aplicação em diversos setores e para públicos variados. No entanto, neste momento, concentramos nossa atenção em um setor específico em que a informação desempenha um papel crucial, porém, muitas vezes, essa informação se encontra dispersa ou mal organizada. Este setor em destaque é o ambiente competitivo de Pokémon Estampas Ilustradas.

1.1 O que são jogos de cartas colecionáveis?

Jogos de cartas colecionáveis, internacionalmente reconhecidos pela sigla TCG, que representa Trading Card Games, são categorizados como jogos de estratégia em que cada participante dispõe de um baralho personalizado composto por diversas cartas, que são escolhidas de acordo com os objetivos de suas estratégias. Antes de sua concepção, o mercado de entretenimento de jogos alternativos sempre foi predominantemente influenciado por jogos de interpretação de personagens fictícios em cenários diversos, conhecidos como RPG (Role Play Games). O pioneiro nesse campo de jogos de cartas colecionáveis foi de 'Magic: The Gathering', criado por Richard Garfield, um professor de matemática e jogos de tabuleiro da Universidade da Pensilvânia.

1.2 O impacto dos TCGs

Não diferente de outros jogos, o cenário competitivo dos Jogos de Cartas Colecionáveis (TCGs) revela uma dinâmica na qual cartas específicas podem se destacar, conferindo-lhes um valor significativamente superior em relação às demais. Tomando novamente o Pokémon TCG como exemplo, a empresa responsável por sua produção, a The Pokémon Company, estima ter comercializado aproximadamente 27,2 bilhões de cartas desde o lançamento do jogo em 1996. Em um ambiente onde a estratégia é preeminente e cada carta pode ter um valor distinto, a determinação de um valor de venda preciso pode se mostrar complexa.

No entanto, é possível formar uma visão geral considerando a média de preços. Cartas comuns, por exemplo, costumam ser avaliadas em torno de R$8,50, enquanto cartas raras ou aquelas que se revelam fundamentais para determinada estratégia podem alcançar valores substanciais, chegando a patamares como R$1,3 milhão.

Adicionalmente, ao examinarmos o panorama competitivo dos TCGs, percebemos que diversas outras transações financeiras estão em curso. Isto se dá em razão da realização de inúmeros campeonatos oficiais que contemplam expressivas premiações para os vencedores. No contexto do Campeonato Mundial de Pokémon Estampas Ilustradas, realizado no ano de 2022, as premiações concedidas variaram significativamente, oscilando entre valores que compreendem U$1,5 mil a U$25 mil, tendo o montante a ser recebido condicionado à classificação alcançada pelos competidores. Alternativamente, a empresa distribuidora do jogo, por vezes, oferece bolsas de estudos como parte das recompensas aos vencedores, realçando ainda mais o caráter competitivo e meritocrático do cenário.

1.3 O que é o Pokémon TCG?

Pokémon Trading Card Game (TCG), conhecido como Pokémon Estampas Ilustradas em português, é um dos principais Jogos de Cartas Colecionáveis (TCGs) do mundo, juntamente com Magic: The Gathering e Yu-Gi-Oh!. Criado no Japão em 1996 e posteriormente introduzido no mercado ocidental em 1999, este jogo é notável por suas cartas traduzidas em aproximadamente 13 idiomas e sua presença oficial em 77 países e regiões distintas. No Brasil, o jogo foi inicialmente distribuído pela Devir Livraria até 2010, quando as licenças foram adquiridas pela Copag, a distribuidora atual.

Uma curiosidade fascinante relacionada a este jogo é que, em 2011, um brasileiro chamado Gustavo Wada conquistou o título de campeão mundial na categoria júnior, que é destinada a jogadores com até 12 anos de idade.

A mecânica do jogo é, em sua essência, simples e envolvente. Cada jogador utiliza um baralho composto por 60 cartas, e o foco principal recai sobre as batalhas travadas entre os Pokémon. Estas batalhas são influenciadas por diversos elementos, como tipos de energia, terrenos, cartas de treinador e itens, sendo a habilidade estratégica do jogador determinante para o desfecho das partidas.

1.4 A falta de uma plataforma

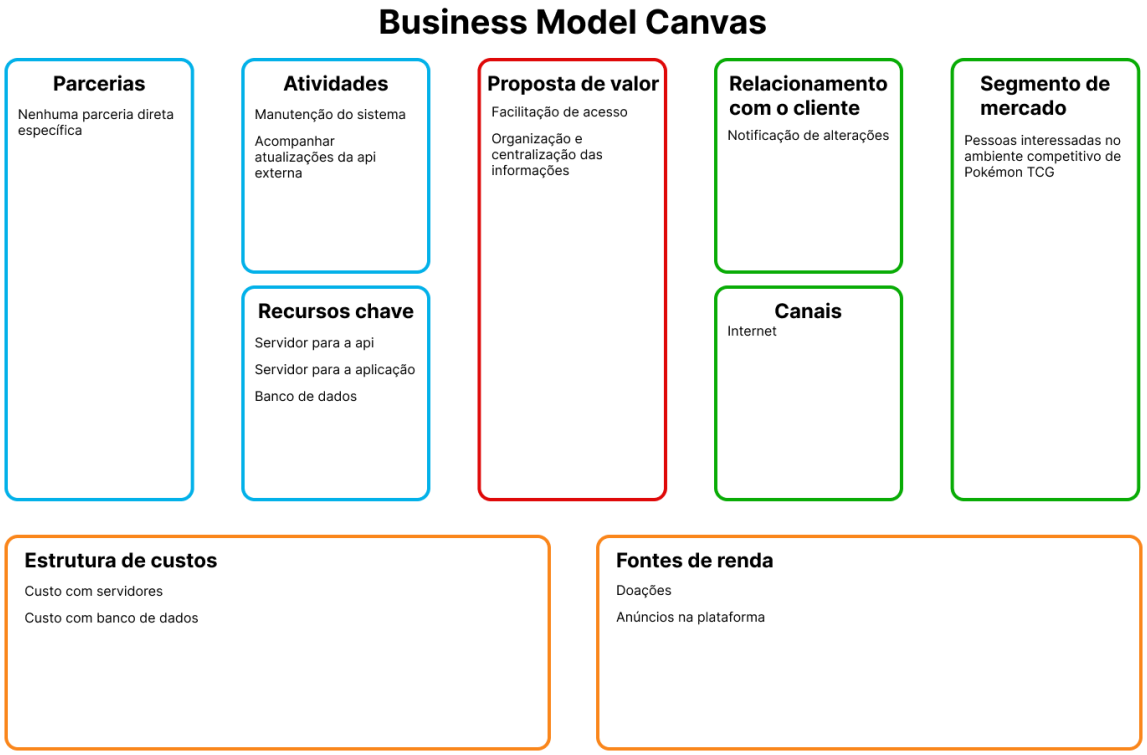
Ao realizar uma pesquisa em diversos mecanismos de busca, constatou-se uma notável lacuna na disponibilidade de plataformas destinadas à agregação de informações concisas no contexto do jogo Pokémon Trading Card Game (TCG). Este jogo, reconhecido por sua influência tanto no âmbito social quanto econômico, carece de uma infraestrutura que centralize e disponibilize informações relevantes, tanto para novos jogadores em busca de uma experiência inicial enriquecedora quanto para jogadores experientes que desejam aprimorar suas estratégias visando competições oficiais.

O propósito da criação da aplicação que estamos desenvolvendo é, portanto, facilitar a disseminação eficiente de informações entre os membros da comunidade de jogadores de Pokémon TCG. Em um primeiro plano, busca-se proporcionar uma jornada mais acessível e proveitosa aos novos jogadores, permitindo-lhes ingressar com maior confiança no ambiente competitivo. Junto a isso, a aplicação visa aperfeiçoar a experiência dos jogadores competitivos veteranos, fornecendo-lhes ferramentas e recursos que ampliem seu conhecimento estratégico, com o intuito de obter melhores resultados em competições oficiais.

É necessário destacar que o Pokémon TCG exerce um impacto significativo em diversos aspectos, tanto sociais quanto econômicos, para todos os envolvidos em sua comunidade. Portanto, a implementação desta plataforma não apenas preenche uma lacuna evidente, mas também fortalece o papel fundamental que o jogo desempenha na interação social, no desenvolvimento de habilidades cognitivas e, não menos importante, na esfera econômica, à medida que o interesse e a competitividade no cenário do Pokémon TCG continuam a crescer de forma constante.

2 Viabilidade do projeto

Figura 1 - Modelo Canvas



Fonte: desenvolvido pelos autores.

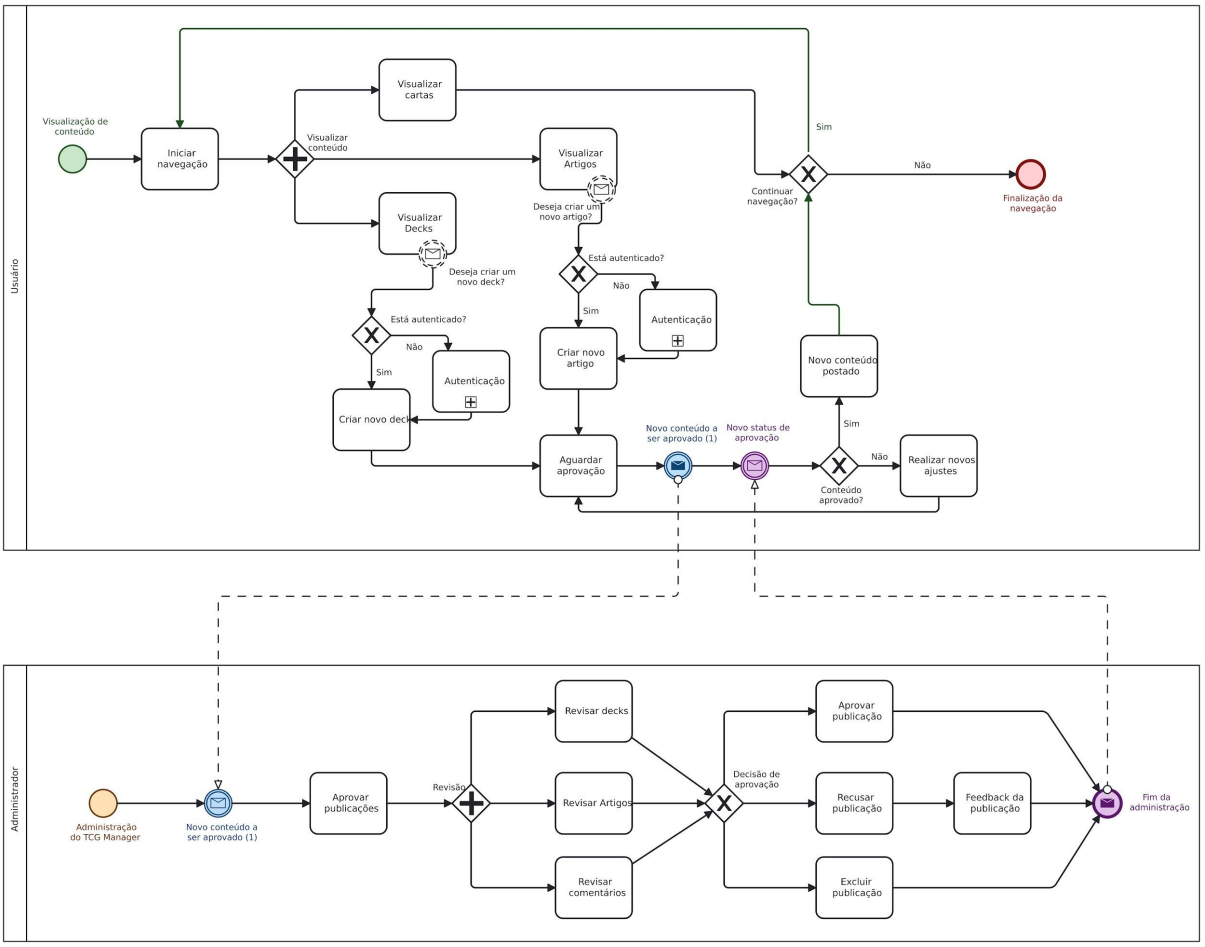
3 Levantamento de Requisitos

3.1 Elicitação e especificação dos Requisitos.

Os requisitos foram identificados com base nas necessidades fundamentais de um sistema de comunicação, levando em consideração diversas situações de interação entre o usuário e o sistema. As principais funcionalidades da aplicação foram identificadas e, com base nisso, desenvolvemos a interface que serviu como alicerce para o desenvolvimento do projeto.

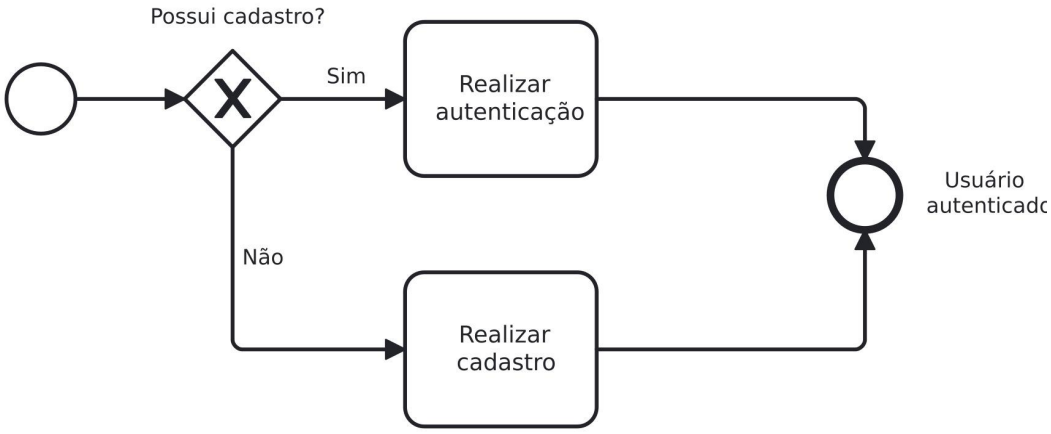
3.2 BPMN

Figura 2 - BPMN Principal



Fonte: desenvolvido pelos autores.

Figura 3 - BPMN Rotina de autenticação



Fonte: desenvolvido pelos autores.

3.3 Requisitos Funcionais

Quadro 1 - Requisitos funcionais do sistema

RF001-Cadastro	Categoria: ( ) Oculto (X)Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que um usuário consiga se cadastrar		
RF002-Autenticação	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário cadastrado consiga se autenticar.		
RF003-Alteração de dados de cadastro	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta

		( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário altere seus dados de cadastro		
RF004-Recuperar cadastro	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário recupere acesso ao seu cadastro realizado.		
RF005-Autenticação por OAuth 2.0	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima ( ) Alta ( ) Média (X) Baixa
Descrição: O sistema deve permitir que o usuário se autentique utilizando autenticação fornecida pelos serviços do google.		
RF006-Listar cartas	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve listar todas as cartas.		
RF007-Filtrar cartas	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário consiga aplicar um filtro à lista de cartas.		

RF008-Análise de carta	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O usuário deve ter acesso a todas as informações disponíveis sobre a carta em questão.		
RF009-Decks relacionados	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve mostrar ao usuário quais decks possuem a carta que está sendo analisada.		
RF010-Comentário sobre cartas	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que os usuários devidamente autenticados escrevam comentários sobre as cartas		
RF011-Favoritar cartas	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima ( ) Alta ( ) Média (X) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado salve cartas como suas favoritas		
RF012-Remover cartas favoritas	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima ( ) Alta

		( ) Média (X) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado remova cartas favoritadas por ele		
RF013-Listar cartas favoritas	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima ( ) Alta ( ) Média (X) Baixa
Descrição: O sistema deve listar cartas favoritadas pelo próprio usuário devidamente autenticado		
RF014-Listar artigos	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve listar todos os artigos disponíveis.		
RF015-Filtrar artigos	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário aplique filtros de busca dentro dos artigos disponíveis		
RF016-Visualizar artigo	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário visualize um artigo por ele selecionado.		

RF017-Comentar artigo	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado escreva um comentário sobre o artigo lido		
RF018-Avaliar artigo	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado avalie um artigo		
RF019-Publicar novo artigo	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado publique um novo artigo		
RF020-Listar artigos publicados	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado liste seus artigos publicados.		
RF021-Deletar artigos publicados	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta

		( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado remova seus artigos publicados.		
RF022-Listar decks	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve listar todos os decks disponíveis.		
RF023-Filtrar decks	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário aplique filtros de busca dentro dos decks disponíveis.		
RF024-Analisar deck	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve mostrar todos os dados disponíveis sobre o deck em questão.		
RF025-Comentar decks	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado escreva um comentário sobre o deck em questão.		

RF026-Avaliar deck	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado avalie o deck em questão.		
RF027-Publicar novo deck	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado publique um novo deck		
RF028-Remover deck publicado	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado remova um deck por ele publicado.		
RF029-Listar decks publicados	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário devidamente autenticado liste seus decks publicados.		
RF030-Listar usuários	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta

		( ) Média ( ) Baixa
Descrição: O sistema deve listar todos os usuários disponíveis		
RF031-Filtrar usuários	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir filtro de buscas na lista de usuários.		
RF032-Ver perfil de outros usuários	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que um usuário tenha acesso ao perfil de um usuário cadastrado.		
RF033-Alterar situação de artigos	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que um usuário administrador altere o status de um artigo para um dos seguintes: Publicado, recusado ou em análise		
RF034-Alterar situação de baralhos	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que um usuário administrador altere o status de um baralho para um dos seguintes: Publicado, recusado ou em		

análise		
RF035-Gerenciar usuários	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que um usuário administrador gerencie outros usuários.		

3.4 Requisitos não funcionais

Quadro 2 - Requisitos Não Funcionais do Sistema

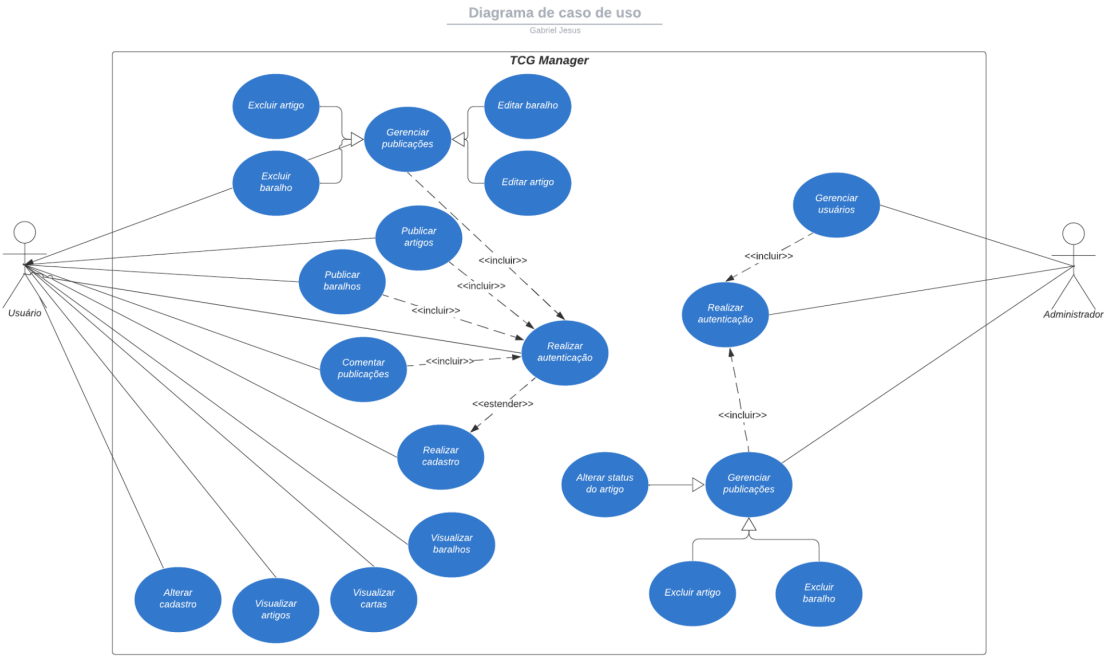
RNF001 - Responsivo	O sistema deve ser responsivo, tendo compatibilidade com qualquer dispositivo que possua acesso a ele.	**Tipo: Interface**	**( ) Desejável (X) Obrigatório**	**(X) Permanente ( ) Transitório**
RNF002 - Paleta de cores	A paleta de cores do sistema deve seguir um padrão similar ao Material Design e/ou ao Daisy Ui	**Tipo: Interface**	**( ) Desejável (X) Obrigatório**	**(X) Permanente ( ) Transitório**
RNF003 - Disponibilidade	O sistema deve informar se há a disponibilidade da informação buscada, seja ela qual for, dentro dos parâmetros do sistema.	**Tipo: Interface**	**( ) Desejável (X) Obrigatório**	**(X) Permanente ( ) Transitório**
RNF004 - Autenticação	A autenticação deve ser efetiva por 24h	**Tipo: Segurança**	**( ) Desejável (X) Obrigatório**	**(X) Permanente ( ) Transitório**

3.5 Casos de uso

Índice de casos de uso e diagramação

UC001 - Realizar cadastro
UC002 - Realizar Autenticação
UC003 - Visualizar Cartas
UC004 - Visualizar baralhos
UC005 - Visualizar artigos
UC006 - Publicar Artigos
UC007 - Publicar baralho
UC008 - Comentar publicação
UC009 - Alterar cadastro
UC010 - Excluir Baralho
UC011 - Excluir Artigo
UC012 - Editar artigo
UC013 - Editar baralho
UC014 - Gerenciar publicações
UC015 - Realizar Autenticação (Administrador)
UC016 - Gerenciar usuários
UC017 - Excluir Artigo (Administrador)
UC018 - Excluir Baralho (Administrador)
UC019 - Alterar Status de Artigo
UC020 - Gerenciar publicações (Administrador)

Figura 4 - Diagrama de casos de uso



Fonte: Desenvolvido pelos autores

3.6 Especificação dos casos de uso

Quadro 3 - Realizar cadastro

Caso de Uso – Realizar cadastro	
ID	UC 001
Descrição	Este caso de uso descreve o processo pelo qual um usuário realiza o cadastro no sistema, caso ainda não possua uma conta.
Ator Primário	Usuário do sistema
Pré-condição	O usuário não está registrado no sistema.

Cenário Principal	1.O usuário acessa a página de registro. 2.O sistema exibe um formulário de registro com campos como nome, email, senha, etc. 3.O usuário preenche o formulário com as informações necessárias. 4.O usuário clica no botão "Registrar". 5.O sistema verifica se as informações fornecidas são válidas. 6.Se as informações forem válidas, o sistema cria uma conta para o usuário. 7.O sistema redireciona o usuário para a página inicial após o registro.
Pós-condição	O usuário tem um cadastro ativo no sistema.

Quadro 4 - Realizar autenticação

Caso de Uso – Realizar Autenticação	
ID	UC 002
Descrição	Este caso de uso descreve o processo pelo qual um usuário realiza a autenticação no sistema, caso já possua uma conta registrada.
Ator Primário	Usuário do sistema
Pré-condição	O usuário possui um cadastro no sistema.
Cenário Principal	1.O usuário acessa a página de login. 2.O sistema exibe campos para inserção do nome de usuário (ou email) e senha. 3.O usuário insere seu nome de usuário (ou email) e senha. 4.O usuário clica no botão "Entrar". 5.O sistema verifica as credenciais fornecidas. 6.Se as credenciais são válidas, o sistema autentica o usuário.

	7.O sistema redireciona o usuário para a página inicial após a autenticação.
Pós-condição	O usuário está devidamente autenticado no sistema.
Cenário Alternativo	5a - O sistema verifica que as credenciais inseridas pelo usuário são inválidas. 5a1.O sistema exibe uma mensagem de erro informando que as credenciais são incorretas. 5a2.O usuário pode tentar novamente inserindo as credenciais corretas. 5b - O sistema verifica que as credenciais inseridas pelo usuário são inválidas. 5b1.O sistema oferece a opção "Esqueceu a Senha?". 5b2.O usuário clica na opção "Esqueceu a Senha?". 5b3.O sistema fornece um processo de redefinição de senha, que envolve a verificação por email ou outras etapas de segurança.

Quadro 5 - Visualizar cartas

Caso de Uso – Visualizar Cartas	
ID	UC 003
Descrição	Este caso de uso descreve como um usuário pode visualizar as informações das cartas disponíveis na aplicação.
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1.O usuário está na página principal ou na seção de cartas da aplicação. 2.O sistema exibe uma lista das cartas disponíveis, incluindo seus nomes e imagens representativas. 3.O usuário pode rolar a lista para visualizar mais cartas, se houver.

	4.O usuário pode clicar em uma carta específica para ver mais detalhes. 5.O sistema exibe informações adicionais sobre a carta selecionada, como descrição, atributos e estatísticas.
Pós-condição	Nenhuma
Cenário Alternativo	2a - O sistema não possui nenhuma carta cadastrada. 2a1.O sistema exibe uma mensagem informando que não há cartas disponíveis no momento. 4a - O usuário decide voltar à lista de cartas. 4a1.O sistema redireciona o usuário de volta à lista de cartas, permitindo a continuação da navegação.

Quadro 6 - Visualizar baralhos

Caso de Uso – Visualizar baralhos	
ID	UC 004
Descrição	Este caso de uso descreve como um usuário pode visualizar as informações do baralho de cartas Pokémon na aplicação.
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1.O usuário está na página principal ou na seção de baralhos da aplicação. 2.O sistema exibe uma lista de baralhos de cartas Pokémon disponíveis. 3.O usuário pode rolar a lista para visualizar mais baralhos, se houver. 4.O usuário pode clicar em um baralho específico para ver mais detalhes. 5.O sistema exibe informações adicionais sobre o baralho selecionado, como nome, quantidade de cartas e cartas incluídas.

Pós-condição	Nenhuma
Cenário Alternativo	2a - O sistema não possui nenhum baralho de cartas Pokémon cadastrado. 2a1.O sistema exibe uma mensagem informando que não há baralhos disponíveis no momento. 4a - O usuário decide voltar à lista de baralhos. 4a1.O sistema redireciona o usuário de volta à lista de baralhos, permitindo a continuação da navegação.

Quadro 7 - Visualizar artigos

Caso de Uso – Visualizar artigos	
ID	UC 005
Descrição	Este caso de uso descreve como um usuário pode visualizar artigos
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1.O usuário está na seção de artigos sobre Pokémon TCG na aplicação. 2.O sistema exibe uma lista de artigos disponíveis relacionados ao ambiente do jogo. 3.O usuário pode rolar a lista para visualizar mais artigos, se houver. 4.O usuário pode clicar em um artigo específico para ler o seu conteúdo. 5.O sistema exibe o conteúdo completo do artigo, incluindo texto, imagens e outros elementos relacionados ao ambiente do Pokémon TCG.
Pós-condição	Nenhuma

Cenário Alternativo	2a - O sistema não possui nenhum artigo relacionado ao Pokémon TCG cadastrado. 2a1.O sistema exibe uma mensagem informando que não há artigos disponíveis no momento. 4a - O usuário decide voltar à lista de artigos. 4a1.O sistema redireciona o usuário de volta à lista de artigos, permitindo a continuação da navegação.

Quadro 8 - Publicar artigos

Caso de Uso – Publicar Artigos	
ID	UC 006
Descrição	Este caso de uso descreve como um usuário autenticado pode criar e publicar um artigo na aplicação.
Ator Primário	Usuário do sistema
Pré-condição	O usuário está devidamente autenticado no sistema
Cenário Principal	1.O usuário está logado na aplicação. 2.O usuário navega até a seção de criação de artigos. 3.O sistema exibe um formulário de criação de artigos, incluindo campos como título, conteúdo e imagens. 4.O usuário preenche o formulário com as informações do artigo. 5.O usuário clica no botão "Publicar" para enviar o artigo. 6.O sistema registra o artigo e o marca para análise manual pela equipe responsável.
Pós-condição	O artigo é enviado para análise manual.
Cenário Alternativo	5a - O usuário opta por cancelar a publicação do artigo. 5a1.O sistema não registra o artigo e redireciona o usuário de volta à página principal. 5b- Ocorre um erro durante a tentativa de publicação do artigo (por exemplo, falha no envio).

	5b1.O sistema exibe uma mensagem de erro e permite que o usuário tente novamente ou entre em contato com o suporte.

Quadro 9 - Publicar baralho

Caso de Uso – Publicar baralho	
ID	UC 007
Descrição	Este caso de uso descreve como um usuário autenticado pode criar e publicar um baralho que contenha 60 cartas, obedecendo às regras do jogo.
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1.O usuário está logado na aplicação. 2.O usuário acessa a seção de criação de baralhos. 3.O sistema exibe uma interface de criação de baralho, permitindo ao usuário adicionar cartas. 4.O usuário adiciona cartas do baralho até que ele contenha exatamente 60 cartas, obedecendo às regras do jogo Pokémon TCG. 5.O usuário clica no botão "Publicar Baralho" para enviar o baralho. 6.O sistema registra o baralho e o marca para análise manual pela equipe responsável.
Pós-condição	Nenhuma
Cenário Alternativo	5a - O usuário opta por cancelar a publicação do baralho. 5a1.O sistema não registra o baralho e redireciona o usuário de volta à página principal de baralhos. 5b - Ocorre um erro durante a tentativa de publicação do baralho (por exemplo, falha no envio ou baralho não está de acordo com as regras). 5b1.O sistema exibe uma mensagem de erro e permite que o

	usuário tente novamente ou entre em contato com o suporte.

Quadro 10 - Comentar publicação

Caso de Uso – Comentar publicação	
ID	UC 008
Descrição	Este caso de uso descreve como um usuário autenticado pode comentar em uma publicação, que pode ser um baralho ou um artigo, na aplicação.
Ator Primário	Usuário do sistema
Pré-condição	Estar devidamente autenticado
Cenário Principal	1.O usuário está navegando em uma publicação, que pode ser um baralho ou um artigo. 2.O sistema exibe a publicação e uma área para inserção de comentários. 3.O usuário insere seu comentário na área designada. 4.O usuário clica no botão "Comentar" para enviar o comentário. 5.O sistema registra o comentário na publicação.
Pós-condição	Nenhuma
Cenário Alternativo	4a - O usuário opta por cancelar o envio do comentário. 4a1.O sistema não registra o comentário e mantém a página da publicação sem adicionar o novo comentário. 4b - Ocorre um erro durante a tentativa de envio do comentário (por exemplo, falha no envio). 4b1.O sistema exibe uma mensagem de erro e permite que o usuário tente novamente ou entre em contato com o suporte.

Quadro 11 - Alterar cadastro

Caso de Uso – Alterar cadastro	
ID	UC 009
Descrição	Este caso de uso descreve como um usuário autenticado pode alterar seus dados cadastrais, incluindo email, senha, nome, apelido e foto de perfil.
Ator Primário	Usuário do sistema
Pré-condição	O usuário está devidamente autenticado no sistema.
Cenário Principal	1.O usuário acessa a seção de gerenciamento de perfil. 2.O sistema exibe os dados cadastrais atuais do usuário, incluindo email, nome, apelido e foto de perfil. 3.O usuário escolhe qual dado cadastral deseja alterar. 4.O usuário fornece as novas informações correspondentes ao dado escolhido. 5.O usuário confirma as alterações clicando em um botão "Salvar" ou similar. 6.O sistema atualiza os dados cadastrais conforme as informações fornecidas pelo usuário.
Pós-condição	Nenhuma
Cenário Alternativo	5a - O usuário opta por cancelar as alterações. 5a1.O sistema mantém os dados cadastrais inalterados. 5b - Ocorre um erro durante a tentativa de alteração dos dados cadastrais (por exemplo, senha antiga incorreta). 5b1.O sistema exibe uma mensagem de erro e permite que o usuário tente novamente ou entre em contato com o suporte.

Quadro 12 - Excluir baralho

Caso de Uso – Excluir Baralho	
ID	UC 010

Descrição	Este caso de uso descreve como um usuário, devidamente autenticado, pode excluir um baralho que tenha criado na aplicação.
Ator Primário	Usuário do sistema
Pré-condição	O usuário está devidamente autenticado no sistema e é o autor do baralho que deseja excluir.
Cenário Principal	1.O usuário acessa a seção de gerenciamento de suas próprias publicações. 2.O sistema lista os baralhos criados pelo usuário, incluindo informações como nome do baralho, data de criação e status atual. 3.O usuário seleciona o baralho que deseja excluir. 4.O usuário escolhe a opção "Excluir Baralho". 5.O sistema remove permanentemente o baralho do sistema, incluindo todas as cartas e dados associados a esse baralho.
Pós-condição	Nenhuma
Cenário Alternativo	4a - O usuário opta por cancelar a ação. 4a1.O sistema mantém o baralho inalterado.

Quadro 13 - Excluir artigo

Caso de Uso – Excluir Artigo	
ID	UC 011
Descrição	Este caso de uso descreve como um usuário, devidamente autenticado, pode excluir um artigo que tenha criado
Ator Primário	Usuário do sistema
Pré-condição	O usuário está devidamente autenticado no sistema e é o autor do artigo que deseja excluir.

Cenário Principal	1.O usuário acessa a seção de gerenciamento de suas próprias publicações. 2.O sistema lista os artigos criados pelo usuário, incluindo informações como título, data de criação e status atual. 3.O usuário seleciona o artigo que deseja excluir. 4.O usuário escolhe a opção "Excluir Artigo." 5.O sistema remove permanentemente o artigo do sistema, incluindo todos os dados associados a esse artigo.
Pós-condição	Nenhuma
Cenário Alternativo	4a - O usuário opta por cancelar a ação. 4a1.O sistema mantém o artigo inalterado.

Quadro 14 - Editar artigo

Caso de Uso – Editar artigo	
ID	UC 012
Descrição	Este caso de uso descreve como um usuário, devidamente autenticado, pode editar um artigo criado por ele na aplicação
Ator Primário	Usuário do sistema
Pré-condição	O usuário está devidamente autenticado no sistema e é o autor do artigo que deseja editar.
Cenário Principal	1.O usuário acessa a seção de gerenciamento de suas próprias publicações. 2.O sistema lista os artigos criados pelo usuário, incluindo informações como título, data de criação e status atual. 3.O usuário seleciona o artigo que deseja editar. 4.O sistema exibe o artigo para edição, permitindo ao usuário fazer as alterações necessárias no conteúdo. 5.O usuário salva as alterações realizadas no artigo.
Pós-condição	Nenhuma

Cenário Alternativo	4a - O usuário opta por cancelar a edição. 4a1.O sistema mantém o artigo inalterado.

Quadro 15 - Editar baralho

Caso de Uso – Editar baralho	
ID	UC 013
Descrição	Este caso de uso descreve como um usuário, devidamente autenticado, pode editar um baralho criado por ele na aplicação.
Ator Primário	Usuário do sistema
Pré-condição	O usuário está devidamente autenticado no sistema e é o autor do baralho que deseja editar.
Cenário Principal	1.O usuário acessa a seção de gerenciamento de suas próprias publicações. 2.O sistema lista os baralhos criados pelo usuário, incluindo informações como nome do baralho, data de criação e status atual. 3.O usuário seleciona o baralho que deseja editar. 4.O sistema exibe o baralho para edição, permitindo ao usuário fazer as alterações necessárias nas cartas ou informações do baralho. 5.O usuário salva as alterações realizadas no baralho.
Pós-condição	Nenhuma
Cenário Alternativo	4a - O usuário opta por cancelar a edição. 4a1.O sistema mantém o baralho inalterado.

Quadro 16 - Gerenciar publicações

Caso de Uso – Gerenciar publicações

ID	UC 014
Descrição	Este caso de uso descreve como um usuário, devidamente autenticado, pode gerenciar suas próprias publicações
Ator Primário	Usuário do sistema
Pré-condição	O usuário está devidamente autenticado no sistema.
Cenário Principal	1.O usuário acessa a seção de gerenciamento de suas próprias publicações. 2.O sistema lista as publicações criadas pelo usuário, incluindo informações como título, data de criação e status atual. 3.O usuário seleciona uma de suas publicações que deseja gerenciar. 4.O usuário tem as seguintes opções de gerenciamento: - Excluir a publicação existente (Casos de uso 10 e 11). - Editar a publicação existente (Casos de uso 12 e 13).
Pós-condição	Nenhuma
Cenário Alternativo	Não possui

Quadro 17 - Realizar autenticação (Administrador)

Caso de Uso – Realizar Autenticação (Administrador)	
ID	UC 015
Descrição	Este caso de uso descreve como um usuário administrador pode realizar a autenticação no sistema para acessar funcionalidades administrativas.
Ator Primário	Administrador

Pré-condição	O usuário está devidamente autenticado no sistema.
Cenário Principal	1.O administrador acessa a página de login do sistema. 2.O administrador insere seu nome de usuário e senha. 3.O sistema verifica as credenciais do administrador. 4.Se as credenciais estiverem corretas e o usuário for um administrador, o sistema concede acesso às funcionalidades administrativas. 5.O administrador é redirecionado para a página principal do sistema como um administrador autenticado.
Pós-condição	O administrador está devidamente autenticado como administrador.
Cenário Alternativo	4a - O sistema não valida as credenciais do administrador como corretas. 4a1.O sistema exibe uma mensagem de erro informando que as credenciais são inválidas. 4a2.O administrador pode tentar novamente ou solicitar uma redefinição de senha, se aplicável.

Quadro 18 - Gerenciar usuários

Caso de Uso – Gerenciar usuários	
ID	UC 016
Descrição	Este caso de uso descreve como um usuário administrador pode gerenciar outros usuários, incluindo a capacidade de desativar o acesso em seus perfis e/ou excluí-los do sistema.
Ator Primário	Administrador
Pré-condição	O administrador está devidamente autenticado no sistema.
Cenário Principal	1.O administrador acessa a seção de gerenciamento de usuários. 2.O sistema lista os perfis de usuários cadastrados, incluindo informações como nome de usuário e status de acesso.

	3.O administrador seleciona um perfil de usuário que deseja desativar. 4.O administrador escolhe a opção "Desativar Acesso". 5.O sistema desativa o acesso ao perfil de usuário selecionado, impedindo que ele faça login no sistema.
Pós-condição	Nenhuma
Cenário Alternativo	3a - O administrador seleciona um perfil de usuário que deseja excluir. 3a1.O administrador escolhe a opção "Excluir Perfil". 3a2.O sistema remove permanentemente o perfil de usuário do sistema, incluindo todos os dados associados a esse perfil. 4a- O administrador opta por cancelar a ação. 4a1.O sistema mantém o status ou perfil de usuário inalterado.

Quadro 19 - Excluir artigo (Administrador)

Caso de Uso – Excluir Artigo (Administrador)	
ID	UC 017
Descrição	Este caso de uso descreve como um usuário administrador pode excluir um artigo que tenha sido publicado na aplicação.
Ator Primário	Administrador
Pré-condição	O usuário está devidamente autenticado no sistema.
Cenário Principal	1.O administrador acessa a seção de gerenciamento de artigos. 2.O sistema lista os artigos disponíveis, incluindo informações como título, autor e status atual (que deve ser "Publicado"). 3.O administrador seleciona o artigo que deseja excluir. 4.O administrador escolhe a opção "Excluir Artigo." 5.O sistema remove permanentemente o artigo do sistema, incluindo todos os dados associados a esse artigo.

Pós-condição	Nenhuma
Cenário Alternativo	4a - O administrador opta por cancelar a ação. 4a1.O sistema mantém o artigo inalterado.

Quadro 20 - Excluir baralho (Administrador)

Caso de Uso – Excluir Baralho (Administrador)	
ID	UC 018
Descrição	Este caso de uso descreve como um usuário administrador pode excluir um baralho que tenha sido publicado na aplicação.
Ator Primário	Administrador
Pré-condição	O administrador está devidamente autenticado no sistema.
Cenário Principal	1.O administrador acessa a seção de gerenciamento de baralhos. 2.O sistema lista os baralhos disponíveis, incluindo informações como nome do baralho, autor e status atual (que deve ser "Publicado"). 3.O administrador seleciona o baralho que deseja excluir. 4.O administrador escolhe a opção "Excluir Baralho". 5.O sistema remove permanentemente o baralho do sistema, incluindo todas as cartas e dados associados a esse baralho.
Pós-condição	Nenhuma
Cenário Alternativo	4a - O administrador opta por cancelar a ação. 4a1.O sistema mantém o baralho inalterado.

Quadro 21 - Alterar status de artigo.

Caso de Uso – Alterar Status de Artigo	
ID	UC 019

Descrição	Este caso de uso descreve como um usuário administrador pode alterar o status de um artigo para um dos seguintes: "Publicado," "Em Análise" ou "Recusado". Para os status "Em Análise" ou "Recusado," o administrador deve fornecer um motivo escrito a ser enviado junto com o novo status
Ator Primário	Administrador
Pré-condição	O administrador está devidamente autenticado no sistema.
Cenário Principal	1.O administrador acessa a seção de gerenciamento de artigos. 2.O sistema lista os artigos disponíveis, incluindo informações como título, autor e status atual. 3.O administrador seleciona um artigo que deseja gerenciar. 4.O administrador escolhe uma das opções de status disponíveis: "Publicado," "Em Análise" ou "Recusado". 5.Se o administrador escolher o status "Em Análise" ou "Recusado", ele deve fornecer um motivo escrito. 6.O sistema atualiza o status do artigo conforme selecionado pelo
Pós-condição	Nenhuma
Cenário Alternativo	O administrador opta por cancelar a ação. O sistema mantém o status do artigo inalterado.

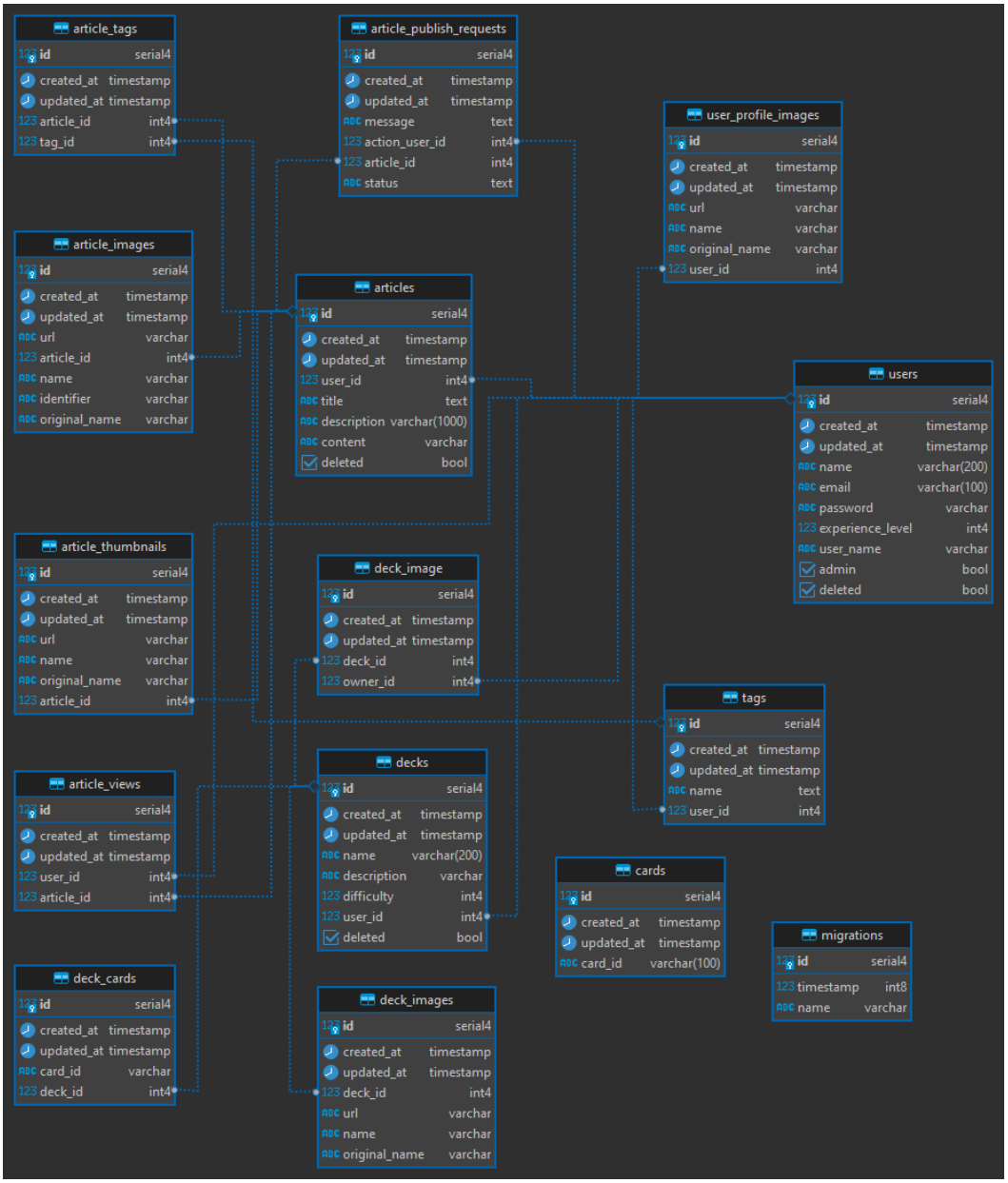
Quadro 22 - Gerenciar publicações (Administrador)

Caso de Uso – Gerenciar publicações (Administrador)	
ID	UC 020
Descrição	Este caso de uso descreve como um usuário administrador pode gerenciar uma publicação, que pode ser um baralho ou um artigo.
Ator Primário	Administrador

Pré-condição	O administrador está devidamente autenticado no sistema.
Cenário Principal	O administrador acessa a seção de gerenciamento de publicações. O sistema lista as publicações disponíveis, incluindo informações como título, autor e status atual. O administrador seleciona uma publicação que deseja gerenciar. - Excluir a publicação existente (Casos de uso 17 e 18). - Editar o status de um artigo (Caso de uso 19)
Pós-condição	Nenhuma
Cenário Alternativo	Não se aplica

3.7 Diagrama Entidade Relacionamento

Figura 5 - Diagrama Entidade Relacionamento (DER)



Fonte: desenvolvido pelos autores

4.Ferramentas e Métodos ou desenvolvimento

4.1 Ferramentas

a) Figma: Utilizado na elaboração dos protótipos das telas, com a adoção da licença MIT (Figma, 2023). Essa escolha foi respaldada por diversos fatores. Primeiramente, a plataforma foi preferida devido à elevada familiaridade dos membros da equipe com suas funcionalidades e interface. Além disso, o

Figma se destaca pela sua ampla disponibilidade de recursos, tutoriais e informações que facilitam significativamente sua utilização. Isso garante que nossa equipe possa aproveitar ao máximo essa poderosa ferramenta no processo de desenvolvimento.

b) Visual Studio Code: IDE (Integrated Development Environment) amplamente reconhecida e apreciada. Nossa escolha se deve à sua versatilidade e à vasta coleção de plugins disponíveis, que enriquecem significativamente o ambiente de desenvolvimento. A diversidade de recursos oferecidos pelo Visual Studio Code é um ativo importante para nossa equipe, pois ajuda a agilizar o processo de desenvolvimento e a melhorar a produtividade.

c) Next Js: Framework frontend que utiliza o React JS como biblioteca de UI. Essa escolha permite a criação de interfaces de usuário de alta qualidade e altamente interativas. O Next Js se destaca por sua capacidade de desenvolver aplicações web de forma eficiente, usando JavaScript. Isso nos permite fornecer aos nossos usuários uma experiência agradável e responsiva.

d) Firebase Storage: O Firebase Storage é um serviço de armazenamento em nuvem oferecido pelo Firebase, uma plataforma de desenvolvimento de aplicativos móveis e da web desenvolvida pela Google. O Firebase Storage é projetado especificamente para armazenar, servir e gerenciar arquivos e mídia, como imagens, vídeos, áudios e documentos, em aplicativos móveis e da web. A escolha dessa ferramenta se deu em virtude de sua simplicidade de uso e integração ao código, além de ser altamente confiável para o armazenamento das imagens necessárias para o projeto.

e) RabbitMQ: RabbitMQ é um sistema de mensagens de código aberto (open-source) que atua como um middleware de mensagens. Ele é projetado para ser uma plataforma de mensagens confiável e escalável, utilizada para transmitir e receber mensagens entre aplicativos e sistemas distribuídos. O RabbitMQ é amplamente empregado em arquiteturas de microsserviços e

sistemas distribuídos para comunicação assíncrona. Essa ferramenta foi selecionada devido à sua facilidade de utilização e integração no código do projeto, além de ser rápida e eficiente para situações que requerem uma fila de mensagens para serem processadas de forma síncrona e de acordo com a disponibilidade da aplicação, uma vez que essas mensagens não precisam ser processadas em tempo real, ganhando assim desempenho na aplicação.

f) Visual Studio Code: Trata-se de um editor de código utilizado para desenvolver todo o código contido na aplicação. A escolha se baseou na familiaridade prévia com a ferramenta, bem como na extensa biblioteca de extensões que a integram com diversos softwares, auxiliando no desenvolvimento de software, como o Docker, por exemplo.

g) Docker: Docker é uma plataforma de código aberto que facilita a criação, implantação e execução de aplicativos em contêineres. Os contêineres são unidades de empacotamento de software que incluem o código, suas dependências e configurações, permitindo que os aplicativos sejam executados de maneira consistente em diferentes ambientes, como desenvolvimento, teste e produção. A escolha dessa ferramenta se deu devido à sua capacidade de simplificar o desenvolvimento e simplificar a instalação das ferramentas necessárias para o funcionamento do software, como banco de dados, RabbitMQ, entre outros, além de proporcionar flexibilidade na distribuição dos recursos do servidor de maneira simples e rápida de configurar.

h) Node.js: Node.js é uma plataforma de código aberto que permite aos desenvolvedores criar aplicativos de servidor e executar código JavaScript no lado do servidor. Ela é construída sobre a engine V8 JavaScript da Google, a mesma engine que alimenta o navegador Chrome, mas é projetada para ser executada em ambientes de servidor. A escolha dessa ferramenta se deve à familiaridade prévia com ela e também porque a stack utilizada para o

desenvolvimento do front-end foi totalmente baseada em JavaScript. Portanto, optamos por usar a mesma linguagem no lado do servidor para facilitar a manutenção do código. Além disso, o Node.js acelera significativamente o desenvolvimento da API devido à sua facilidade de configuração.

i) TypeScript: TypeScript é uma linguagem de programação de código aberto desenvolvida pela Microsoft que estende o JavaScript adicionando recursos de tipagem estática e outros recursos de linguagem. É frequentemente referido como um superset do JavaScript, o que significa que todo código JavaScript válido também é código TypeScript válido. No entanto, o TypeScript oferece vantagens adicionais sobre o JavaScript, tornando-o uma escolha popular para desenvolvedores que desejam melhorar a qualidade, segurança e manutenção de seus projetos de software. Optamos pelo uso do TypeScript devido à sua capacidade de adicionar tipagem forte à linguagem JavaScript, evitando assim muitos bugs e problemas durante o desenvolvimento do software, além de aumentar a manutenção do código.
j) Express.js: Express.js é um framework web de código aberto para Node.js. Ele é projetado para simplificar o desenvolvimento de aplicativos web e APIs (Application Programming Interfaces) em Node.js, oferecendo uma variedade de recursos e funcionalidades que facilitam a criação de aplicativos web de forma rápida e eficiente. Essa ferramenta foi escolhida porque acelera as fases do desenvolvimento do software ao simplificar muitas partes do desenvolvimento de uma API, além de fornecer um sistema de roteamento completo e uma forma simples de tratamento dos erros da API.

k) TypeORM: TypeORM é um Object-Relational Mapping (ORM) de código aberto para TypeScript e JavaScript (ECMAScript). É uma biblioteca que permite aos desenvolvedores interagir com bancos de dados relacionais usando objetos e classes em vez de escrever consultas SQL manualmente. O TypeORM simplifica o acesso a bancos de dados e a manipulação de dados, tornando o desenvolvimento de aplicativos com bancos de dados

relacionais mais fácil e produtivo. Esta ferramenta foi escolhida devido à facilidade de uso e integração com o banco de dados escolhido, bem como por simplificar a criação das queries, uma vez que essa biblioteca disponibiliza várias abstrações para essa finalidade.

l) PostgreSQL: Frequentemente abreviado como "Postgres," o PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto (RDBMS). É conhecido por ser um dos bancos de dados relacionais mais poderosos, robustos e extensíveis disponíveis, sendo amplamente utilizado em uma variedade de aplicativos, desde pequenos projetos pessoais até grandes sistemas empresariais. A escolha desse banco de dados se deu devido à facilidade de instalação e uso, além de sua robustez.

m) pokemontcg.io: O pokemontcg.io é uma API pública que reúne uma base de dados sobre cartas, sets, coleções e muito mais relacionados ao jogo de cartas Pokémon TCG. Essa ferramenta foi selecionada porque disponibiliza todas as informações necessárias para o site sobre as cartas e coleções, além de possuir SDKs que facilitam a integração com o código da aplicação. Além disso, ela possui um mecanismo robusto de busca.

4.2 Métodos ou Desenvolvimento

A linguagem de programação escolhida para o desenvolvimento do back-end foi o JavaScript, utilizando o TypeScript para adição de tipagens. Essa linguagem foi escolhida devido à sua facilidade de uso e ao fato de já estar sendo utilizada no desenvolvimento do front-end, o que possibilitou manter o projeto inteiro dentro de um único ecossistema de linguagem, melhorando assim o desenvolvimento e a manutenção do código. Além disso, o Node.js foi utilizado em conjunto com o framework Express.js para criar o código que roda no lado do servidor, sendo responsável pela criação do servidor HTTP e pelo gerenciamento do roteamento da API.

Para o front end da aplicação, foi utilizado o framework Next JS fazendo uso do TypeScript. Next JS para a criação das interfaces utiliza React. Dentro desta aplicação os itens desenvolvidos foram componentes para a melhor adaptação e reaproveitamento de elementos, visto que, todo o layout das interfaces mantinham um padrão para atender a consistência do design escolhido.

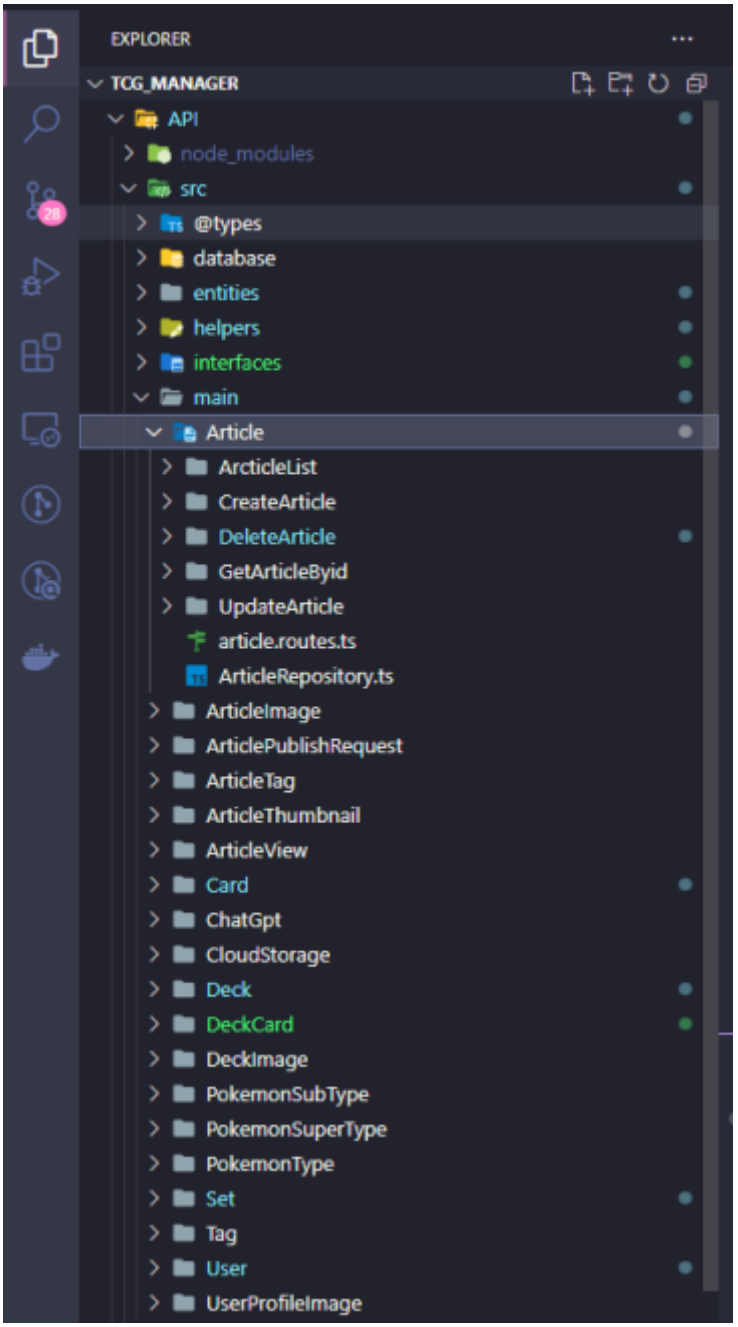
Dentro das cores escolhidas para a aplicação, a paleta foi baseada em duas paletas de sistemas de design, o Material Design e o Daisy UI. As cores baseiam-se nas cores de estado do Material, possuindo cores para status de erro, sucesso, confirmação e avisos, juntamente com cores básicas como, primárias, secundárias, acentuais e sistêmicas.

A opção de framework escolhido foi o Next JS devido a experiências anteriores dos envolvidos com a plataforma, juntamente com sua capacidade surpreendente de se adaptar a análises de SEO (Search Engine Optimization), o que é vital para uma aplicação com intuitos informativos, já o sistema de pastas e organização de arquivos utilizados no front end baseiam-se nos conceitos de Clean Architecture (MARTIN, 2019) e SOLID (MARTIN, 2011).

4.2.1 Estrutura das pastas (Back end)

A estruturação das pastas do projeto foi organizada de acordo com as necessidades de desenvolvimento, sem seguir uma convenção específica. As pastas foram separadas em "database" (que contém a conexão com o banco de dados e suas configurações, além de todas as migrações criadas), "entities" (que contém todas as entidades do projeto), "helpers" (onde ficam todos os arquivos com funcionalidades para auxiliar o desenvolvimento), "interfaces" (que contêm os arquivos das interfaces utilizadas) e "main" (que contém toda a lógica e rotas do projeto, sendo subdividida em pastas separadas por entidade e, dentro delas, separadas por casos de uso). Veja um exemplo na imagem abaixo:

Figura 6



Fonte: Desenvolvido pelos autores

Logo em seguida, temos a pasta "middlewares" (que contém os middlewares utilizados nas rotas) e "queues" (que contém os arquivos relacionados às filas de mensagens do projeto).

O desenvolvimento da API começou com a modelagem das entidades, que foram planejadas durante as reuniões de alinhamento e idealização do projeto. Todas as entidades foram modeladas utilizando decoradores da biblioteca

TypeORM. Veja um exemplo da entidade "user" na imagem abaixo:

Figura 7

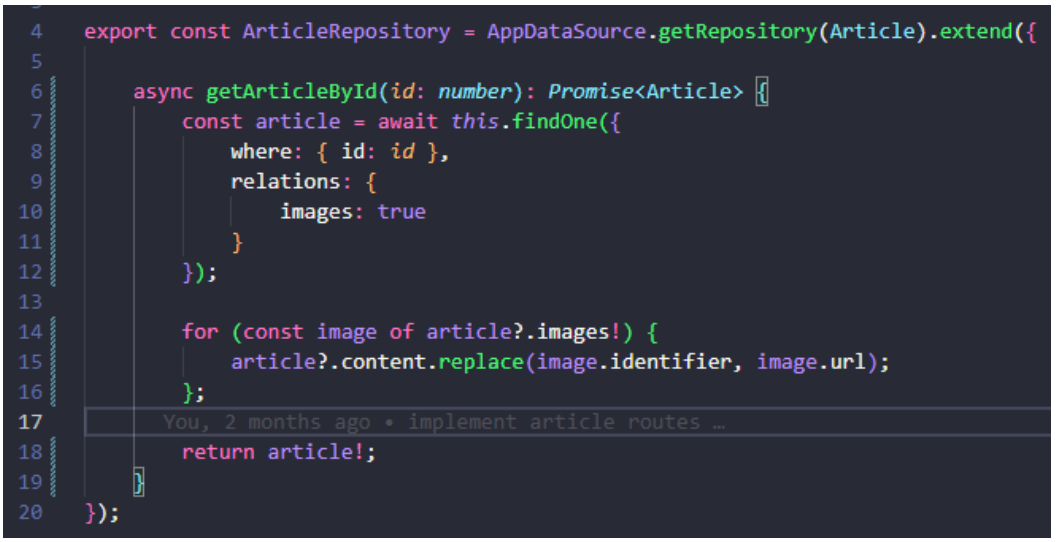
```
@Entity('users')
export class User {
    @PrimaryGeneratedColumn()
    id: number;

    @CreateDateColumn()
    created_at: Date;

    @UpdateDateColumn()
    updated_at: Date;

    @Column({ nullable: false, unique: true })
    user_name: string;

    @Column({ type: 'varchar', length: '200', nullable: false })
    name: string;

    @Column({ type: 'varchar', length: '100', nullable: false, unique: true })
    email: string;
      You, 8 months ago • Creating user's autentication …
    @Column({ type: 'varchar', nullable: false, select: false })
    password: string;

    @Column({ nullable: false, default: false })
    admin: boolean;

    @Column({ nullable: false, default: false })
    deleted: boolean;

    @OneToMany(() => Deck, deck => deck.user)
    created_decks: Deck[];

    @OneToMany(() => Article, article => article.user)
    created_articles: Article[];

    @OneToOne(() => UserProfileImage, userProfileImage => userProfileImage.user)
    profile_image: UserProfileImage;
};
```

Fonte: Desenvolvido pelos autores

Nesse arquivo de modelagem de entidades, são definidas todas as propriedades dos campos, suas tipagens, limites de caracteres e se os campos podem ser nulos ou não. Além disso, são definidas todas as relações com outras entidades.

Após a modelagem das entidades, o desenvolvimento dos casos de uso foi iniciado, incluindo listagens, criação de entidades, edição, etc. Todos esses casos de uso foram organizados na estrutura de pastas dentro da pasta "main", onde estão agrupados todos os casos de uso de todas as entidades. Nessa etapa do desenvolvimento, os repositórios das entidades também foram criados, contendo os métodos relacionados à camada de banco de dados da aplicação, como queries específicas para listagem, criação e edição das entidades às quais o repositório

Figura 8



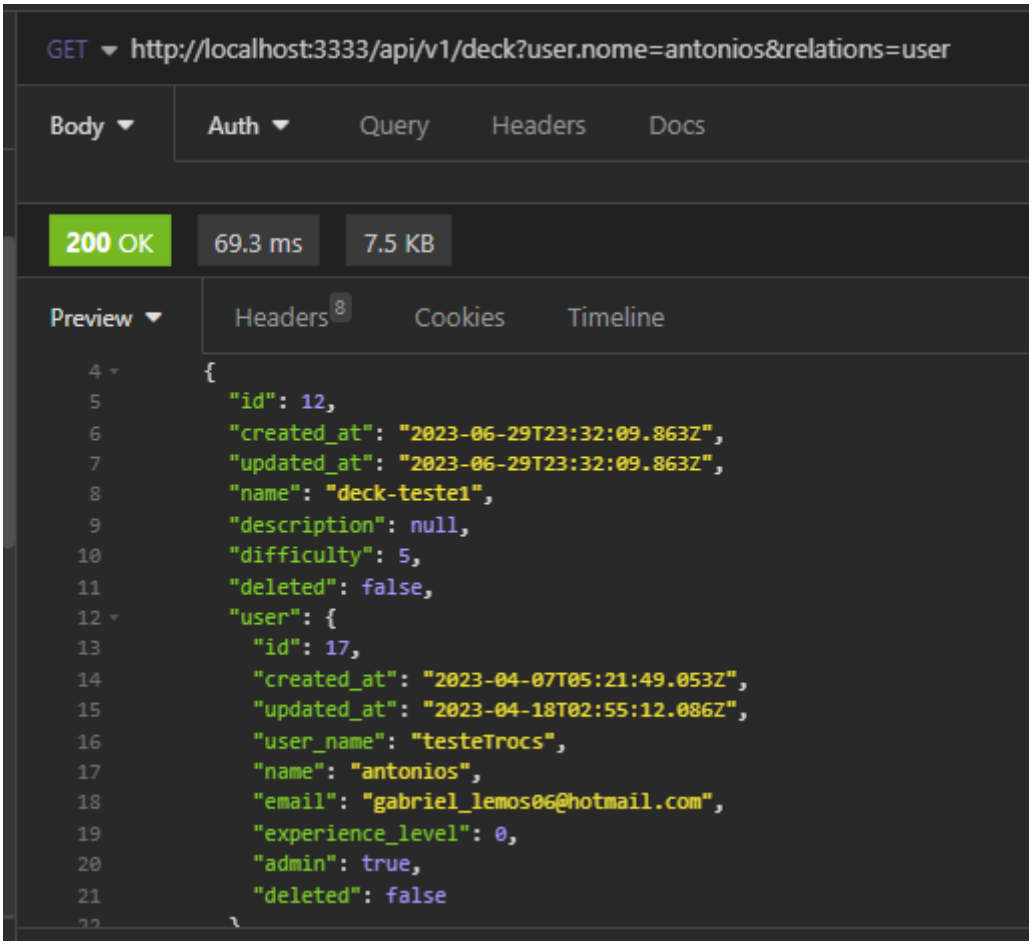
Fonte: Desenvolvido pelos autores

Nesse caso, devido à biblioteca TypeORM, ao utilizar o método getRepository(entidade).extend(), é possível obter todos os métodos genéricos fornecidos pelo repositório padrão do TypeORM, além de poder adicionar novos métodos, como mostrado na imagem acima. Essa biblioteca facilita o desenvolvimento ao criar queries automaticamente de forma simples e direta, eliminando a necessidade de escrever queries completamente cruas no código.

Durante o desenvolvimento do projeto, percebemos que todas as entidades precisam de uma rota de listagem com diversos filtros. Para agilizar o desenvolvimento e evitar a duplicação de código, foi criado um código de filtragem de entidades genérico. Esse desenvolvimento foi a parte mais complexa do projeto, pois precisava atender a todas as entidades do projeto, ser performático e validar

campos válidos para busca. Veja um exemplo de uma rota de listagem utilizando essa listagem genérica na imagem abaixo:

Figura 9



Fonte: Desenvolvido pelos autores

A rota recebe query parameters, onde cada parâmetro representa um campo de busca da entidade, e é possível passar o parâmetro "relations" para trazer a relação desejada juntamente com a entidade principal.

Outra necessidade identificada durante o desenvolvimento do projeto foi a contagem de visualizações dos artigos criados. No entanto, percebemos que, em situações de alto tráfego no site, essa rotina poderia gerar uma grande quantidade de escritas no banco de dados, comprometendo a performance da API. Para contornar esse problema, criamos uma fila de mensagens a serem processadas utilizando o RabbitMQ. Cada mensagem corresponde a uma visualização de um artigo. Além disso, foi criado um micro serviço que consome essa fila e registra as visualizações em um banco de dados não relacional, de acordo com a

disponibilidade de processamento do micro serviço. O banco de dados não relacional foi escolhido devido à sua capacidade de receber um grande número de gravações sem perder a performance, uma vez que será utilizado apenas para essa finalidade. Veja um exemplo do processamento da mensagem na imagem abaixo:

Figura 10

```
export class ArticleViewReceiver {

    private _queue = 'article_view';

    async initArticleViewReceiver(): Promise<void> {

        const createQueueConnection = new CreateQueueConection();
        const connection = await createQueueConnection.createConecction();

        const channel = await connection.createChannel();

        await channel.assertQueue(this._queue);

        channel.consume(
            this._queue,
            async (message) => {
                if (message) {
                    try {
                        const messageJson = JSON.parse(JSON.parse(message.content.toString()));
                        const articleView = new ArticleView();

                        articleView.article = messageJson.article;
                        articleView.user = messageJson.user;

                        await ArticleViewRepository.save(articleView);
                        channel.ack(message);
                    } catch (err) {
                        channel.reject(message, false);
                        throw new ApiError(`${err}`, 500);
                    };
                };
            }
        );
    };

};
```

Fonte: Desenvolvido pelos autores

Durante a concepção do projeto, percebemos que seria necessário um método de armazenamento de imagens, já que seria amplamente utilizado por várias entidades do projeto. Atualmente, não é comum armazenar as imagens diretamente no banco de dados, devido ao processamento necessário para salvá-las no banco e ao aumento significativo do espaço em disco utilizado. Para resolver esse problema, optamos por uma solução performática: o Firebase

Storage, que é um sistema de armazenamento de mídia em nuvem oferecido pelo Firebase, uma plataforma desenvolvida pelo Google. Essa ferramenta facilita a integração com o código e é muito eficiente. Após o upload de uma imagem para o Firebase Storage, é gerada uma URL de visualização instantânea, que é armazenada no banco de dados para posterior acesso pelos usuários no front-end.

Para criar uma API confiável, é essencial ter um tratamento de erros eficiente que possa lidar com os erros da API de forma adequada e evitar falhas. Para isso, desenvolvemos um middleware para capturar todos os erros síncronos e assíncronos da aplicação. Nesse momento, também utilizamos a biblioteca "Express-async-errors" para lidar com os erros assíncronos sem a necessidade de usar blocos try/catch no código, agilizando o desenvolvimento e melhorando o tratamento de erros por parte da API. Veja um exemplo do middleware de erros abaixo:

Figura 11

```
const errorHandler = (error: Error & Partial<ApiError>, req: Request, res: Response, next: NextFunction) => {
    const statusCode = error.statusCode ?? 500;
    const errorMessage = error.statusCode ? error.message : 'Internal Server Error';
    console.log('error: ' + error.message)
    return res.status(statusCode).json({ message: errorMessage });
}
```

Fonte: Desenvolvido pelos autores

4.2.2 Estrutura das pastas (Front end)

Para estruturar as pastas dentro do front end da aplicação foram aplicados os conceitos de Clean Architecture. Por mais que o front não necessite de uma organização tão complexa quanto essa, a fim de manter um padrão e facilitar a manutenção em casos futuros decidiu-se manter estes conceitos (figura 12).

Juntamente com todas essas novas diretrizes para os diretórios, foram aplicadas regras mais restritivas utilizando as bibliotecas disponibilizadas pelo ESLint, forçando um padrão de código específico em todos os arquivos.

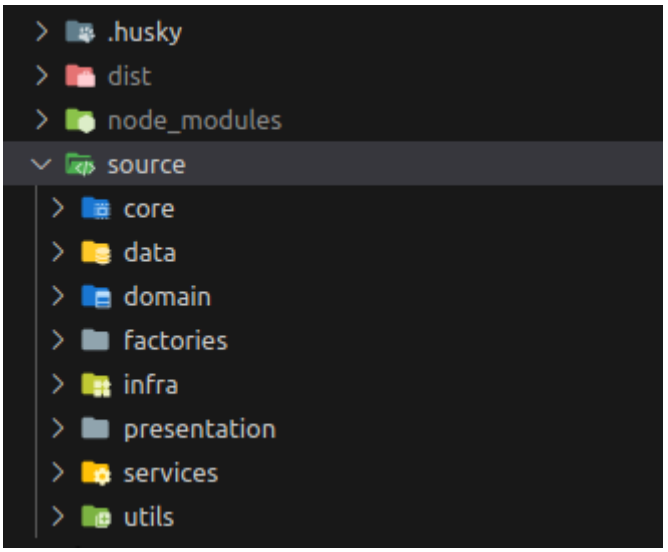
O código da aplicação encontra-se dentro do diretório source, dentro dele, existem as seguintes divisões:

- Core: Para arquivos vitais para a estrutura, com informações sobre tipos de

erros, estruturas de dados e classes abstratas

- Data: Para implementações de repositórios onde os dados podem ser requisitados.
- Domain: Onde as regras de negócio, casos de uso e repositórios são estruturados.
- Factories: Neste diretório se localizam as funções que realizam as injecções de dependências dentro dos repositórios, obedecendo o princípio de Inversão de dependência do SOLID.
- Infra: Localizam-se os arquivos de serviço que foram implementados baseados em interfaces criadas na camada de serviço.
- Apresentação: Camada de apresentação, onde se localizam todos os arquivos referentes a acesso do usuário.
- Services: Onde os serviços são determinados para serem implementados posteriormente, aplicando o princípio de inversão de dependência.
- Utils: Arquivos de funções globais que podem ser usados em toda a aplicação em qualquer camada.

Figura 12

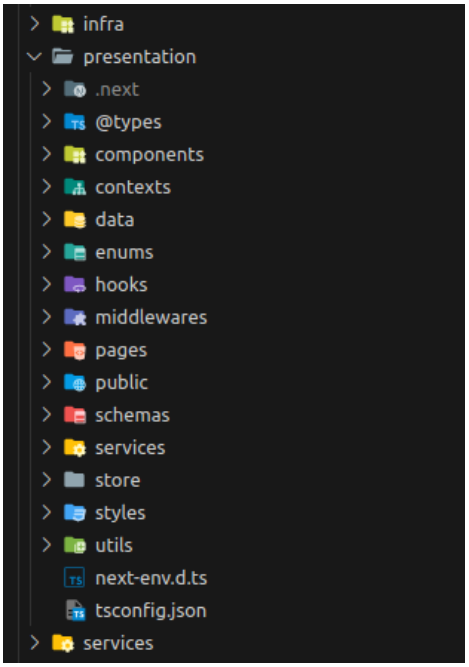


Fonte: Desenvolvido pelos autores

Por se tratar de um sistema de diretórios personalizados muitos arquivos estão mais separados e fragmentados do que o normal, entretanto, como estamos trabalhando com um framework que possui suas próprias diretrizes e

especificações, mantemos alguns diretórios padrões do Next JS encapsulados dentro do diretório de apresentação.

Figura 13



Fonte: Desenvolvido pelos autores

5 Resultados e discussão

A aplicação TCG Manager adota uma abordagem de navegação não linear, permitindo aos usuários acesso direto a qualquer página a partir de qualquer ponto. Na página inicial (figura 14), encontramos um banner central que exibe informações oficiais, seguido por seções que apresentam baralhos e artigos mais visualizados.

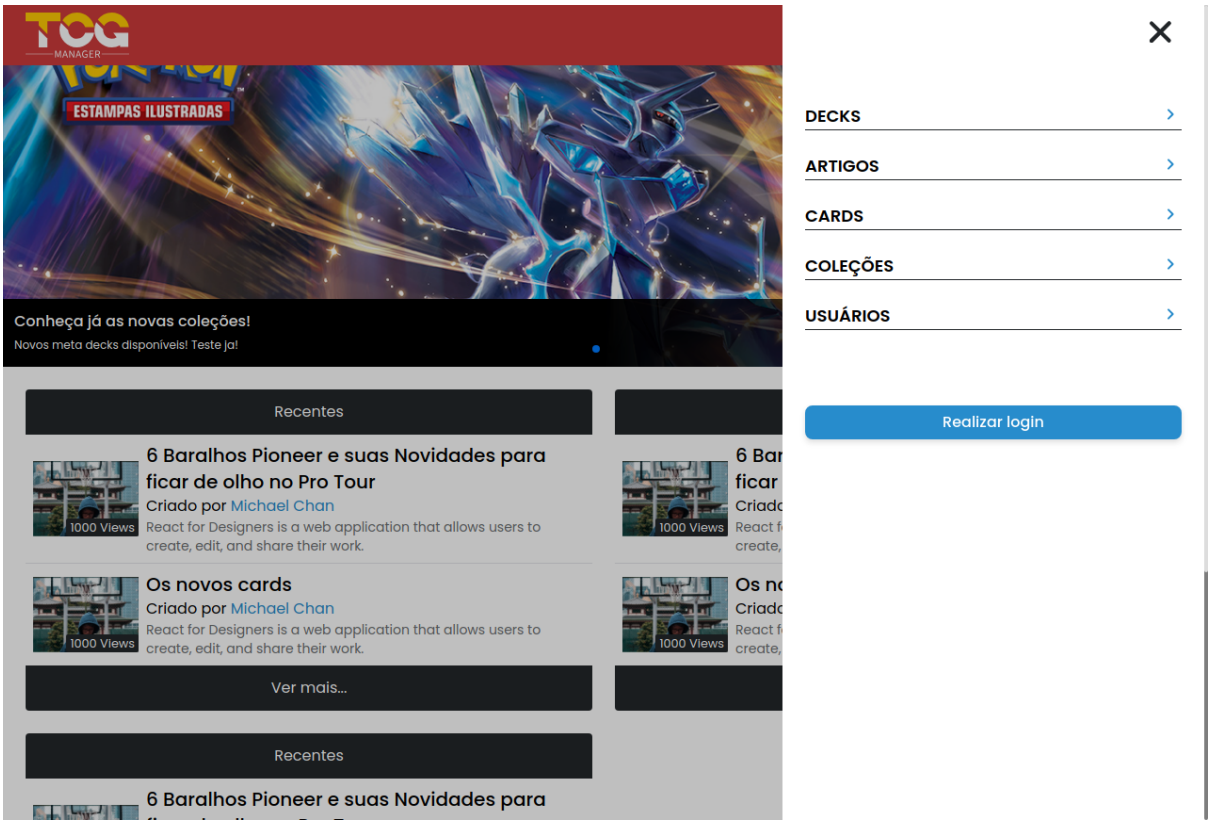
Dentro do menu de navegação, os usuários não autenticados têm acesso apenas às páginas de visualização de conteúdo. As páginas de criação de publicações estão disponíveis exclusivamente para usuários autenticados, não sendo acessíveis por visitantes não autenticados.

As páginas de listagem (figura 15, figura 18 e figura 20) contam com recursos de busca e filtro, transformados em componentes para simplificar o desenvolvimento e manter uma consistência de design. O ambiente de visualização de cartas (figura 17) é modular, se adaptando para cada tipo de carta com suas especialidades. Por outro lado, as páginas de criação apresentam características únicas.

Para a criação de baralhos, foi desenvolvido um construtor de baralhos intuitivo, acessível e responsivo (figura 19) . Esse construtor permite aos usuários criar baralhos de forma rápida e eficaz, seguindo apenas alguns passos simples.

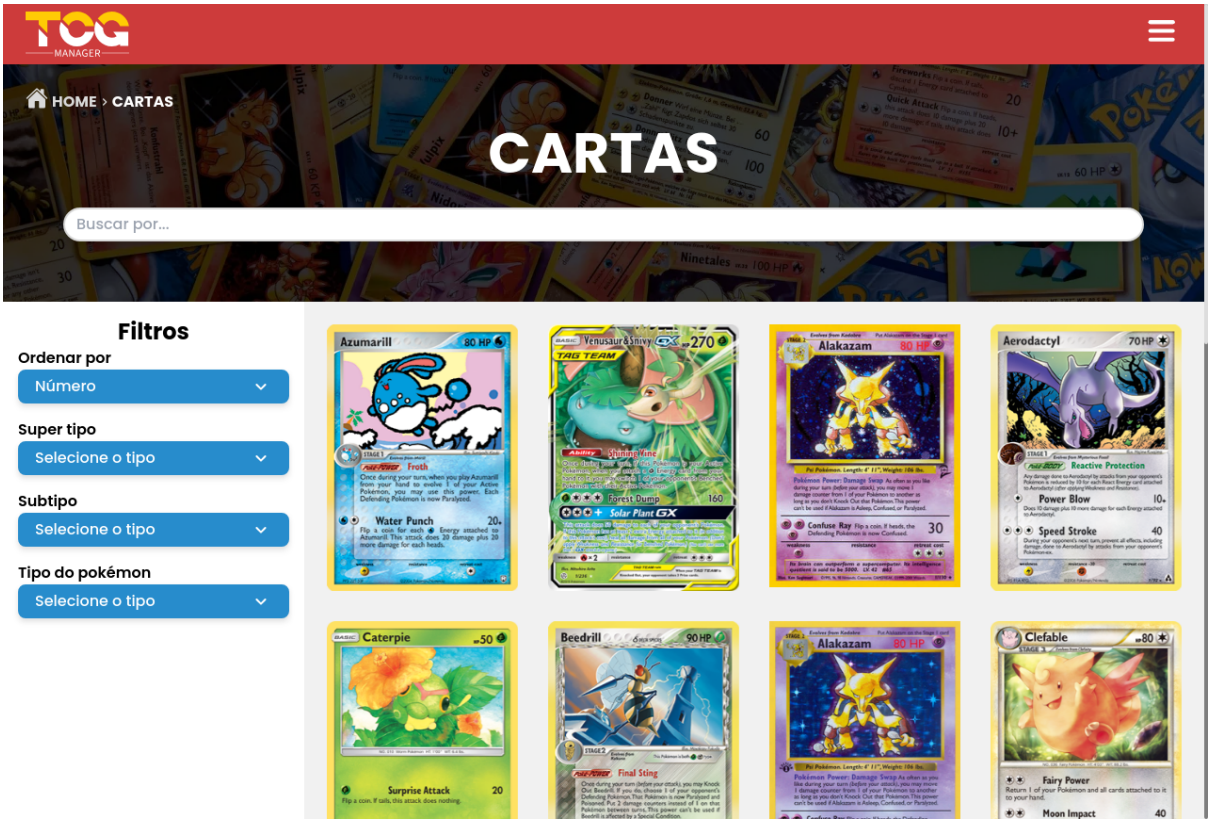
Na página de criação de artigos (figura 16), implementamos um editor de texto que converte todo o conteúdo em HTML usando o padrão WYSIWYG (What You See Is What You Get), que significa "O que você vê é o que você obtém". Isso significa que o texto se comporta como um documento HTML com tags semânticas, proporcionando uma experiência de edição mais flexível e fácil para os usuários.

Figura 14



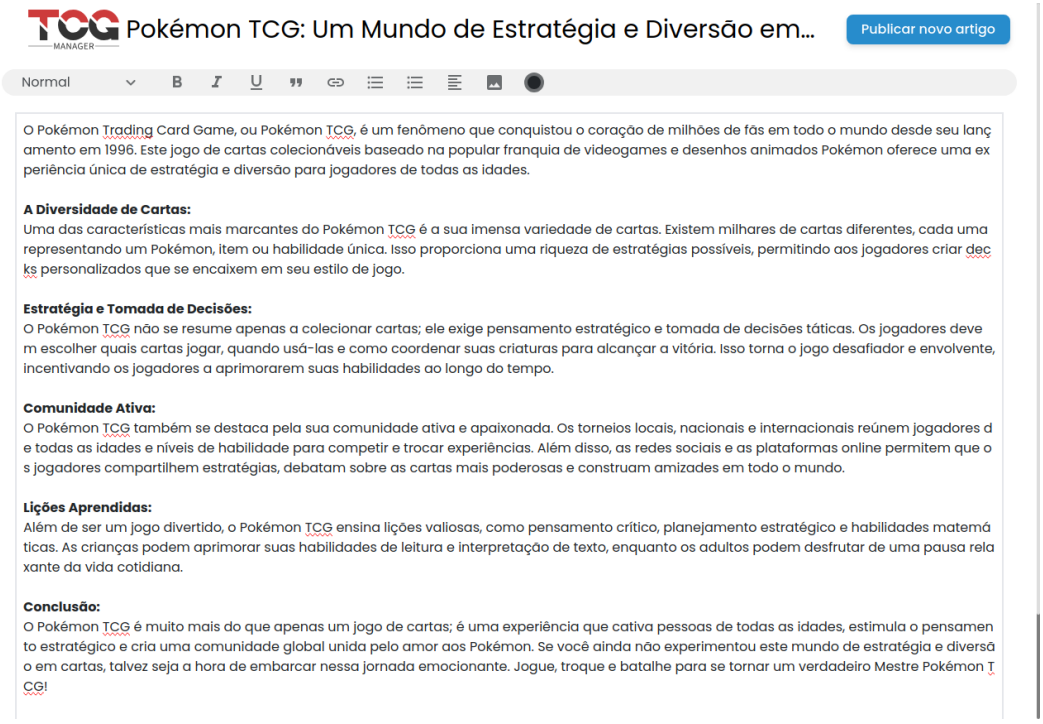
Fonte: Desenvolvido pelos autores.

Figura 15



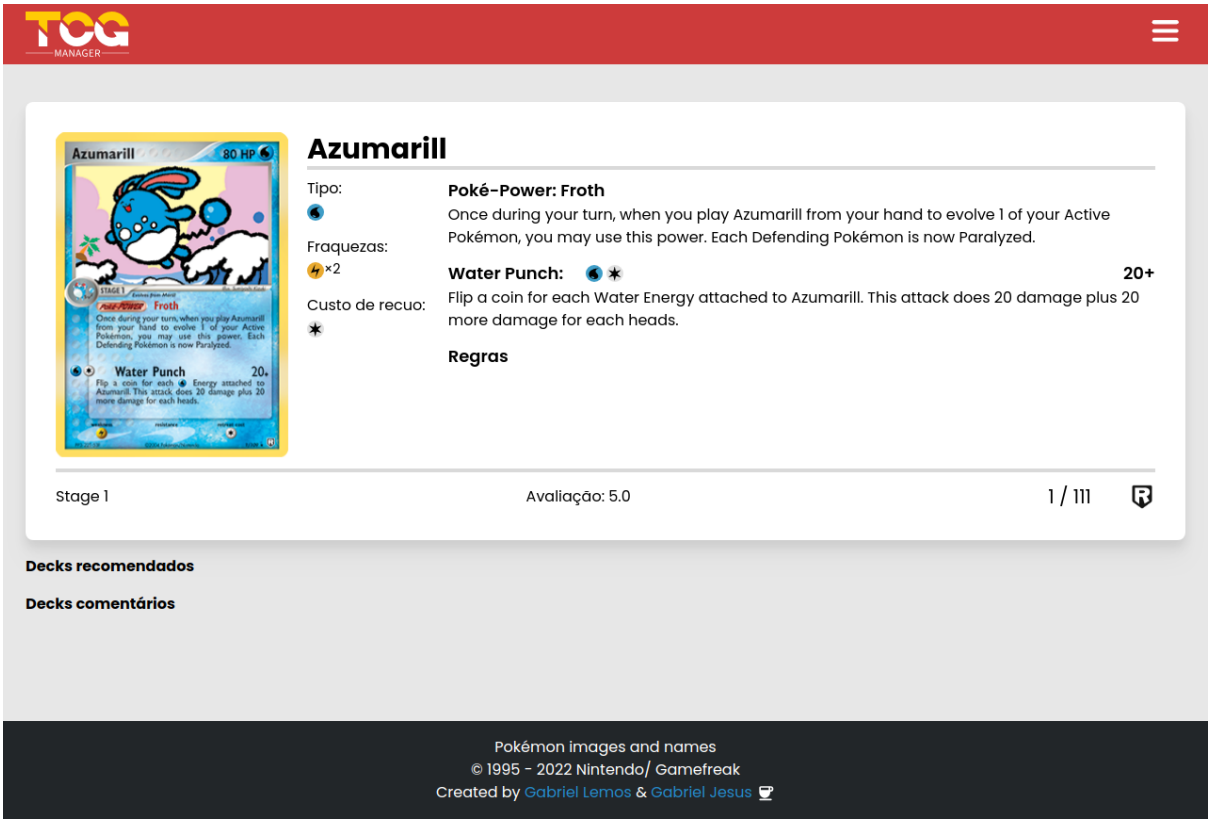
Fonte: Desenvolvido pelos autores.

Figura 16



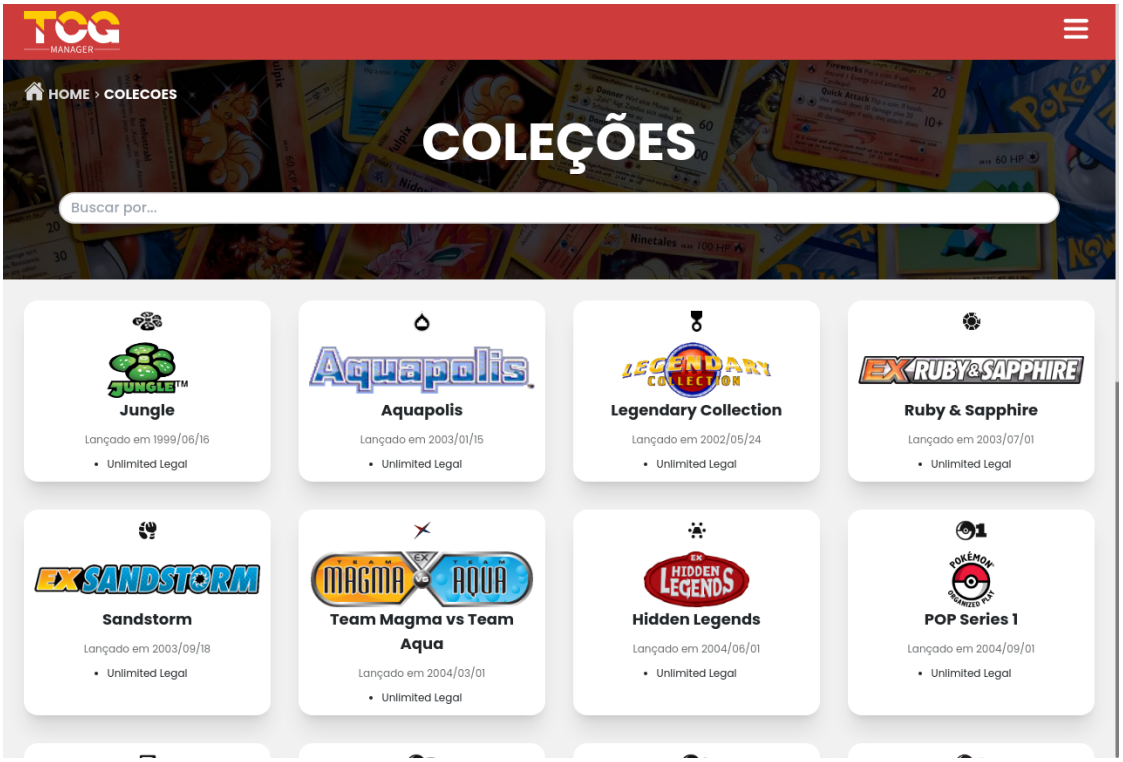
Fonte: Desenvolvido pelos autores.

Figura 17



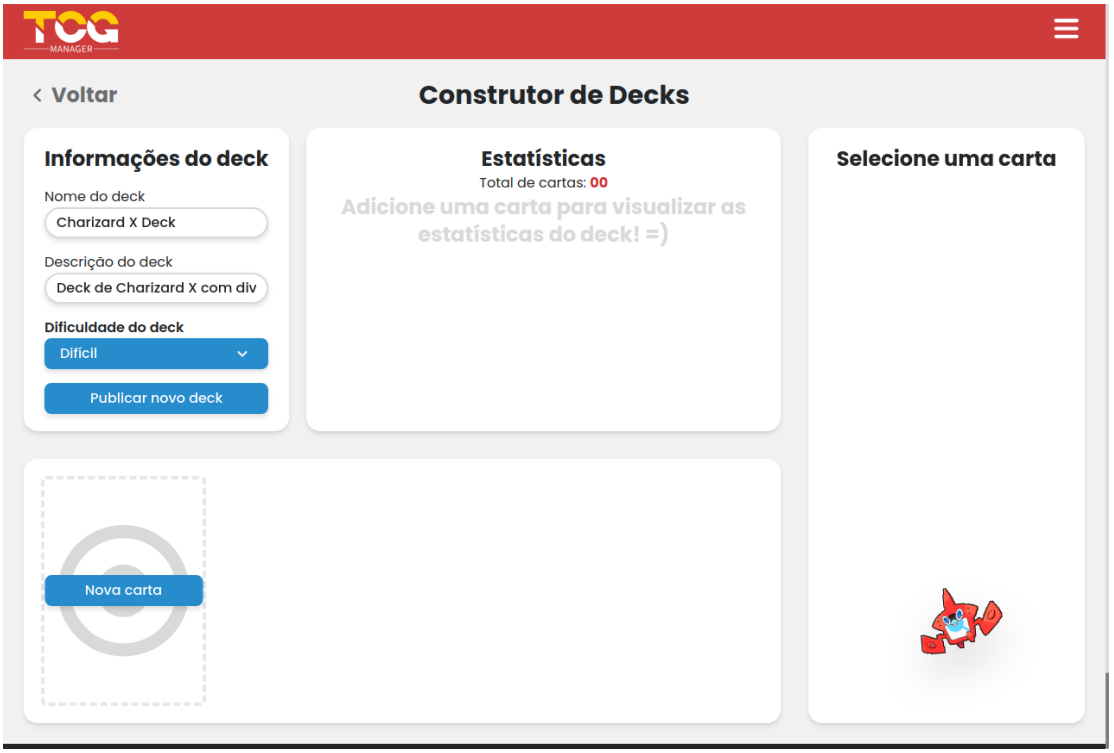
Fonte: Desenvolvido pelos autores

Figura 18



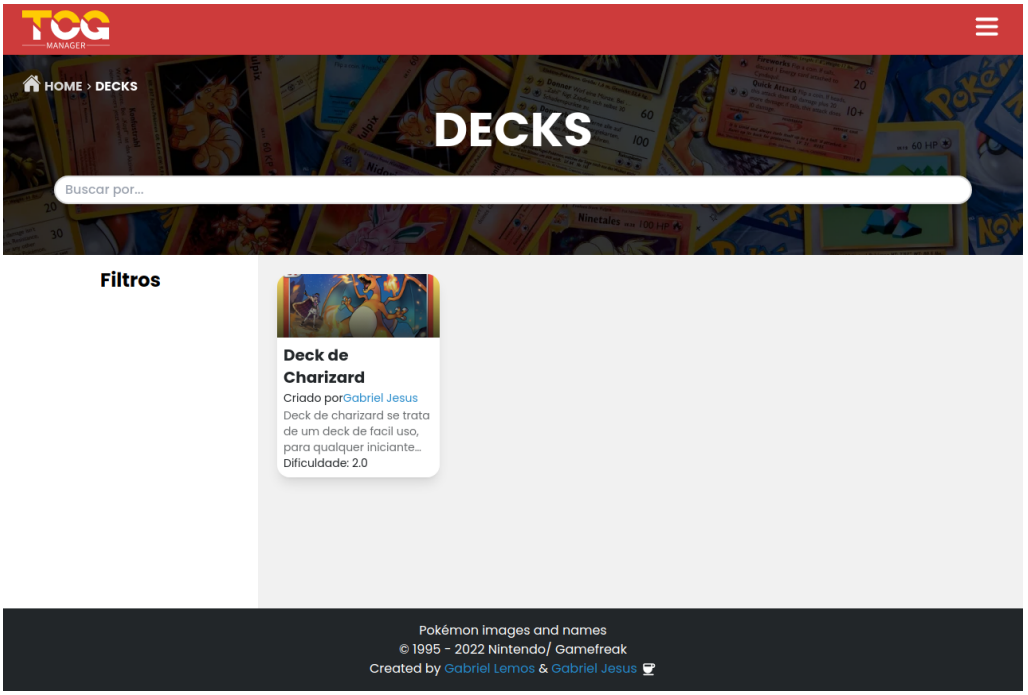
Fonte: Desenvolvido pelos autores.

Figura 19



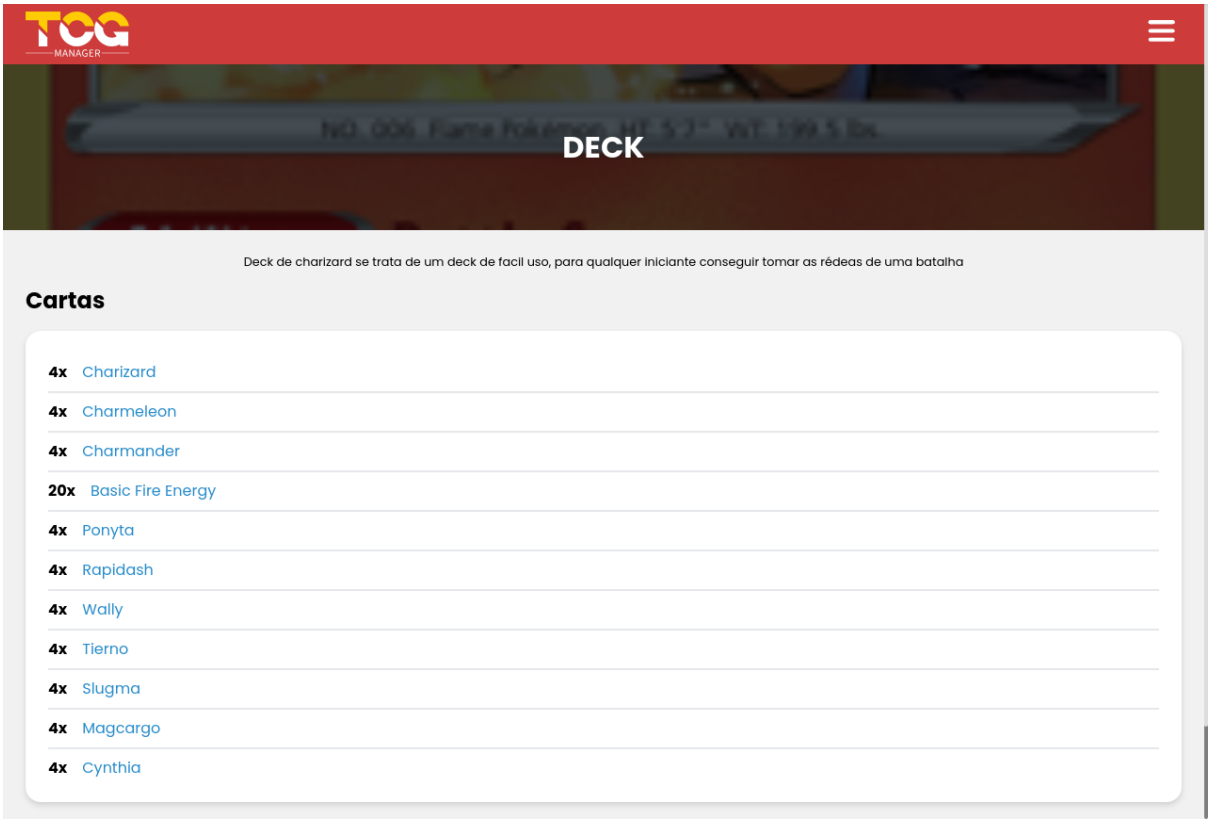
Fonte: Desenvolvido pelos autores.

Figura 20



Fonte: Desenvolvido pelos autores.

Figura 21



Fonte: Desenvolvido pelos autores.

Considerações finais

Com o intuito de tornar a disseminação de informações sobre o Pokémon TCG mais acessível tanto para a comunidade competitiva quanto para os jogadores casuais, foi desenvolvida a aplicação TCG Manager. Esta plataforma tem como propósito centralizar diversas informações e possibilitar aos usuários compartilhar esses conhecimentos de maneira mais interativa e organizada.

Enfrentamos como principal desafio a integração de informações provenientes de fontes externas e a criação de interfaces intuitivas, de modo a simplificar e tornar claro o acesso às informações, permitindo que os usuários extraiam o máximo de conhecimento com apenas alguns cliques.

Um dos próximos passos a serem tomados é a contínua melhoria da acessibilidade da plataforma. Isso inclui aprimorar a experiência para usuários que dependem de leitores de tela, garantindo que a aplicação seja acessível para todos. Além disso, planejamos adicionar traduções para línguas amplamente faladas, como inglês, espanhol, mandarim e outras. Dessa forma, ampliaremos ainda mais o alcance da nossa plataforma e iremos garantir que ela seja útil para uma audiência global.

Referências

FIGMA. **Figma**. Disponível em: https://www.figma.com/. Acesso em: 6 set. 2023.

FIREBASE. **Firebase**. Disponível em: https://firebase.google.com/. Acesso em: 5 set. 2023.

VISUAL STUDIO CODE. **Visual Studio Code**. Disponível em: https://code.visualstudio.com/. Acesso em: 20 set. 2023.

PENILHAS, Bruna. **Todos os vencedores do Pokémon World Championships 2022,** 21 ago.2022. Disponível em: https://www.terra.com.br/gameon/todos-os-vencedores-do-pokemon-world-championships-2022,cd24db0a5135c602192fb4b14d9b02505hlj7x7k.html. Acesso em 20 set.2023

ALÊS, Tamires. **No mundo das cartas Pokémon**, 10 fev.2020. Disponível em: https://www.mundook.com.br/no-mundo-das-cartas-pokemon/. Acesso em 20 set.2023

LONGO, Laelya. **Com mercado que gira US$ 2,5 bi ao ano no Brasil, bancões estão de olho no 'gamer money'**, 22/06/2022. Disponível em: https://valorinveste.globo.com/produtos/servicos-financeiros/noticia/2022/06/22/com-mercado-que-gira-us-25-bi-ao-ano-no-brasil-bancoes-estao-de-olho-no-gamer-money.ghtml . Acesso em 22 set.2023

LOURENÇO, Hugo. **Termo de Abertura do Projeto: o que é, como funciona e como podemos utilizá-lo na agilidade**, 07/10/2021. Disponível em: https://www.objective.com.br/insights/termo-de-abertura-do-projeto-o-que-e-como-funciona-e-como-podemos-utiliza-lo-na-agilidade/. Acesso em 22 set.2023

BUTCHER, Billy. **Konami encerra seu ano fiscal com muito sucesso e lucro, principalmente graças aos videogames**, 13 mai.2021. Disponível em: https://www.gamevicio.com/noticias/2021/05/konami-encerra-seu-ano-fiscal-com-muito-sucesso-e-lucro-principalmente-gracas-aos-videogames/ . Acesso em 23 set.2023

JONATHAN, **O que é Pokémon TCG?**, 31/03/2021. Disponível em: https://www.poke-blast-news.net/2010/09/o-que-e-tcg.html. Acesso em 23 set.2023

MARTIN, Robert C. **Código Limpo**: Habilidades Práticas do Agile Software. Rio de Janeiro: Alta Books, 2011.

MARTIN, Robert C. **Arquitetura Limpa**: O Guia do Artesão para Estrutura e Design de Software. Rio de Janeiro: Alta Books, 2019.