**UNIVERSIDADE FEDERAL DO PAMPA**

**Vinicius de Souza Silva**

# Entertainment Tracker

Alegrete
2025

**Vinicius de Souza Silva**

# Entertainment Tracker

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Claudio Schepke

Alegrete
2025

**Vinicius De Souza Da Silva**

# ENTERTAINMENT TRACKER

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação

Monografia defendida e aprovada em: 30 de junho de 2025.

Banca examinadora:

_____

Claudio Schepke

Orientador

Unipampa

_____

Fábio Paulo Basso

Unipampa

_____

Silvio Ereno Quincozes

Unipampa

Assinado eletronicamente por **CLAUDIO SCHEPKE**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 30/06/2025, às 22:46, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **SILVIO ERENO QUINCOZES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 30/06/2025, às 22:58, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **FABIO PAULO BASSO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 01/07/2025, às 15:09, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1770939** e o código CRC **3D5D6ACE**.

Dedico este trabalho à minha família, ao meu orientador e aos amigos, pelo apoio, e companheirismo ao longo desta jornada.

“Não se deixem vencer pelo mal,
mas vençam o mal com o bem.
(Bíblia Sagrada, Romanos 12:21)

## RESUMO

Este trabalho propõe *Entertainment Track*, uma ferramenta destinada à indexação e acompanhamento de recursos audiovisuais, incluindo filmes, séries, músicas, livros e jogos. O projeto visa oferecer uma solução robusta para o gerenciamento de diversos tipos de mídia e permitir contribuições dos usuários para enriquecer o sistema, com a capacidade de personalização e interação. A motivação central parte da constatação de que, atualmente, não existe uma ferramenta capaz de realizar esse rastreamento de forma centralizada, sendo necessário o cadastro e acompanhamento em múltiplas plataformas distintas, o que prejudica a experiência do usuário. A arquitetura do sistema utiliza *.NET 9.0* para a *API*, *Angular 20* para o *front-end* e *MongoDB* como banco de dados. Essas tecnologias foram escolhidas com base em experiências anteriores no mercado de trabalho. Os requisitos funcionais e não funcionais principais foram definidos e incluem segurança, desempenho e escalabilidade. Concluiu-se a configuração inicial do ambiente de desenvolvimento e a definição dos componentes principais do sistema, a documentação da API, que foi refatorada para alinhar os *endpoints* aos padrões de mercado, a introdução da documentação do *front-end* utilizando *Storybook*, além da modelagem e documentação do banco de dados com o uso do *Hackolade*. Ainda restam algumas atividades que estão em andamento, como a implementação das funcionalidades completas, testes e validação.



**Palavras-chave**: Tracker. Indexação de recursos audiovisuais. Indexação de recursos de entretenimento. Arquitetura de Sistema. API. Angular. dotNET 9. Swagger. Storybook. Hackolade.

## ABSTRACT

This work proposes *Entertainment Track*, a tool designed to index and track audiovisual resources, including movies, series, music, books, and games. The project aims to provide a robust solution for managing various types of media. It allows user contributions to enrich the system, with the ability for customization and interaction. The main motivation comes from the observation that, currently, there is no tool capable of performing this tracking in a centralized manner, requiring registration and tracking over multiple distinct platforms, which impairs the user experience. The system architecture uses *.NET 9.0* for the *API*, *Angular 20* for the *front-end*, and *MongoDB* as the database. These technologies were chosen based on previous experiences in the job market. The main functional and non-functional requirements were defined. It includes security, performance, and scalability. The initial configuration of the development environment and the definition of the main system components have been completed, as well as the API documentation, which was refactored to align the endpoints with market standards, the introduction of the front-end documentation using Storybook and the modeling and documentation of the database using Hackolade. There are still some activities in progress, such as complete functionalities implementation, testing, and validation.



**Key-words**: Indexing of audiovisual resources. Indexing of entertainment resources. System Architecture. API. Angular. dotNET 9. Swagger. Storybook. Hackolade.

## LISTA DE FIGURAS

## LISTA DE TABELAS

SUMÁRIO

# 1 INTRODUÇÃO

A disseminação da internet de banda larga e dos dispositivos móveis facilitou o acesso à rede, impulsionando significativamente o consumo de conteúdos audiovisuais e serviços de streaming. Como exemplos de conteúdos há recursos audiovisuais como: livros, audiobooks, podcasts, músicas, filmes, séries, animes, conteúdos asiáticos (*mangás* (japonês), *manhwa* (coreano), *manhua* (chinês)), cartoons e jogos. Segundo uma pesquisa de 2023 que analisa internet, público, aplicativos e tráfego (Global AD, 2023), 69,6% dos usuários brasileiros usam a Internet para assistir vídeos, séries ou filmes, 47,5% para jogar, 53,8% têm interesse em educação e estudos, enquanto 77,6% da população brasileira usa a Internet para buscar informação.

Por haver uma vasta gama de conteúdos, existem rastreadores de conteúdo (*trackers*). Nele, as pessoas podem anotar e registrar qual conteúdo elas estão consumindo e qual parte elas estão acompanhando. Dependendo do tipo de rastreador, funcionalidades e do acesso do usuário, é possível avaliar a qualidade das obras, debater com outros usuários, acompanhar o progresso, adicionar novos conteúdos ou pedir aos administradores para incluir algo que esteja faltando.

Porém muitos dos rastreadores existentes possuem um maior foco em um único conteúdo de acordo com o objetivo do sistema. Por exemplo, *Trakt* (trakt, inc, 2010 - 2024) basicamente engloba os conteúdos áudio-visuais de filmes, séries, cartoons e animes. Já para livros, temos *Skoob* (SKOOB, 2024), onde seu foco são livros comuns. Porém, possuem alguns mangás. Para animes, filmes de animes, mangas, manhwa e manhua existe o rastreador *Kitsu* (KITSU, 2024), extremamente fechado para estes conteúdos (veja *Anexo* A). Para jogos, existe o *LaunchBox* (Unbroken Software, LLC, 2024), onde o sistema possui um único foco, para a área de jogos, não possuindo qualquer flexibilidade a outros conteúdos. Como pode-se perceber pelos exemplos, embora exista um foco principal dos rastreadores, outros tipos de recursos, em menor escala, estão começando a ser adicionados, muitas vezes pela própria requisição dos usuários.

Do ponto de vista do usuário, atualmente é necessário que o mesmo faça o registro em muitos sistemas. Além disso, alguns sistemas acabam eventualmente sendo encerrados, sem recursos de backup ou exportação de dados. Por fim, em alguns casos, apenas usuários pagantes (como *Trakt* (trakt, inc, 2010 - 2024) e *LaunchBox* (Unbroken Software, LLC, 2024)) possuem mais funcionalidades disponíveis dentro do sistema, como, por exemplo, aplicação de filtro aos títulos dos conteúdos acompanhados listados no *Trakt*.

Diante desses problemas, proponho o desenvolvimento de um sistema que possa englobar uma maior quantidade desses tipos de conteúdos de maneira centralizada, chamada *Entreteniment Tracker* (SILVA, 2024). A ideia desse sistema é que o próprio usuário possa adicionar conteúdo novo para os mais diversos nichos de conteúdos. As principais vantagens que serão incluídas nesse novo sistema incluem a concentração de conteúdos distintos em um único lugar, o próprio usuário registrado pode incluir novos itens, com

limitações de segurança e sob uma posterior supervisão, criação de listas de informação que podem ser compartilhadas com outros usuários através de *url's*, além das clássicas que os sistemas em geral possuem (assistindo, quer assistir, terminado, em pausa e abandonado). Oferecer opções de avaliação personalizáveis, como sistema de likes ou notas de 1 a 10, conforme a preferência do usuário.

Para o financiamento do sistema, é necessário que haja primeiramente um certo número de usuários engajados. Posteriormente, poderá haver monetização quando houver redirecionamento para os provedores de conteúdo. Além disso, recursos como afiliados estão em alta (DAVINSON, 2024) e também podem representar uma forma de obtenção de recursos.

O restante do documento está estruturado da seguinte forma. O Capítulo 2 apresenta algumas ferramentas de rastreamento relacionadas à proposta *Entertainment Tracker*. A metodologia de desenvolvimento e avaliação dos experimentos é definida no Capítulo 3. No Capítulo 4 é descrita a implementação da proposta. No Capítulo 5 são mostrados os resultados obtidos na fase de desenvolvimento do sistema. O Capítulo 6 apresenta a documentação da aplicação organizada conforme as camadas da arquitetura. Por fim, a conclusão e os trabalhos futuros estão apresentados no Capítulo 7.

# 2 FERRAMENTAS RELACIONADAS

A Tabela 1 lista alguns trackers e avalia o foco, funcionalidades, existência de funções pagas e flexibilidade dos sistemas. Percebe-se que cada sistema possui um nicho em foco, o que acaba mudando de um para o outro é a flexibilidade com que o sistema permite ou não a adição de conteúdo não relacionado ao seu nicho em foco. Porém, apesar disso, suas funcionalidades são muito similares. Alguns trackers apenas adaptam poucos aspectos da implementação aos conteúdos em foco, para uma maior coerência da *UI* (*User Interface*) (GALITZ, 2007). Um dos maiores exemplos é a implementação de listas de acompanhamento de conteúdos, que está presente na maioria dos trackers pesquisados.

As ferramentas listadas na Tabela 1 foram encontradas através de buscas no *Google* (Google Inc, 2024) sem utilizar palavras-chaves específicas, indicações de amigos, discussões sobre conteúdos como filmes e séries em redes sociais, como *Reddit* (reddit, 2024), *Instagram* (Instagram, 2024) e *Facebook* (Facebook, 2024).

Além das listas de acompanhamento, os trackers tendem a possuir algum tipo de funcionalidade para interação dos usuários nos conteúdos ou com os próprios usuários. Neste último caso, deseja-se gerar discussões, debates e conversas, com o intuito de obter engajamento no sistema e, desta forma, atrair e aumentar a quantidade de usuários. Alguns possuem serviços de assinatura ou funcionalidades exclusivas como comutabilidade de conteúdos registrados. Apesar das vastas funcionalidades, esses sistemas tendem a ser menos flexíveis com relação à adição de conteúdo. Em geral, os sistemas tendem a manter o foco em seu nicho, salvo algumas vezes quando um conteúdo possui uma repercussão de mídia considerável, o que pode ser benéfico ao sistema, trazendo novo público ou pelo menos alguns novos usuários.

Apesar de existirem diversos trackers disponíveis, é importante notar que alguns sistemas, como o *1337x* (1337X, 2007 - 2024), apesar de aparentarem ser trackers, na verdade funcionam como plataformas para download de conteúdo de forma ilícita, utilizando tecnologia de torrent nesse caso. Esses sites não são, de fato, trackers no sentido tradicional de acompanhamento de conteúdo; em vez disso, eles servem como repositórios para compartilhamento de conteúdos, muitas vezes desrespeitando direitos autorais.

Tabela 1 – Ferramentas relacionadas

| **Tracker** | **Conteúdo Foco** | **Funcionalidades Disponíveis** | **Funções Pagas** | **Flexibilidade do Sistema** |
|---|---|---|---|---|
| Kitsu | Animes, Mangas, Manhwa, Manhua | Listas de acompanhamento, comentários nos conteúdos e subitens, grupos, seguidores, solicitação de adição de conteúdo | Não | Baixa |
| Launch Box | Jogos | Listagens de jogos, avaliações e organização | Sim | Não possui |
| Skoob | Livros | Listas de acompanhamento, amigos, grupos, comentários nos conteúdos, adição de novos conteúdos | Não | Média |
| Trakt | Filmes, Séries | Listas de acompanhamento, comentários nos conteúdos e subitens, listas personalizadas, listas colaborativas, seguidores | Sim | Baixa |
| Scribd | Apresentações de Slides, PDFs, Livros | Marcar conteúdo, Adicionar conteúdo novo, Sistema de likes | Sim | Baixa |
| 1337x | Aplicativos, Filmes, Séries, Jogos, Músicas entre outros | Download de conteúdos por torrent | não | Não possui |

# 3 METODOLOGIA

## 3.1 Definição dos Requisitos Funcionais

Para identificar as funcionalidades essenciais e comuns em diferentes plataformas de acompanhamento de conteúdos audiovisuais, foi realizada uma análise comparativa com outros sistemas de *tracking*. Esta análise visa definir as necessidades mínimas dos usuários, garantindo a implementação de funcionalidades adequadas para o acompanhamento dos mais variados tipos de mídia. Os principais requisitos funcionais definidos incluem:

- **Listagem de Acompanhamento**: Ferramentas que permitam aos usuários criar listas de acompanhamento, listas de acompanhamento personalizadas, categorizando conteúdos e marcando progresso.
- **Interação Social**: Mecanismos que possibilitem aos usuários comentar nos conteúdos, interagir com outros membros via comentários ou mensagens mensagens privadas e participar de discussões.
- **Sistema de *Feedback***: Métodos de avaliação dos conteúdos, utilizando opções simples como *likes* ou sistemas mais detalhados com notas atribuídas a diversos aspectos do conteúdo (por exemplo, narrativa, cinematografia, entre outros).

## 3.2 Requisitos Não Funcionais

### 3.2.1 Segurança

A aplicação conta com mecanismos voltados à proteção de dados sensíveis, controle de acesso e conformidade legal. Estão sendo utilizadas boas práticas de desenvolvimento seguro, autenticação baseada em *tokens* e criptografia de dados, com o objetivo de preservar a integridade, confidencialidade e disponibilidade das informações.

O sistema implementa tokens *JWT (JSON Web Token)* protegidos com *JWE (JSON Web Encryption)* e usa cookies com atributos seguros e medidas alinhadas a *LGPD (Lei Geral de Proteção de Dados)*, visando restringir acessos não autorizados e mitigar riscos relacionados a falhas de segurança, como roubo de sessão e ataques *CSRF (Cross-Site Request Forgery)* e exposição indevida.

#### 3.2.1.1 Tokens

A autenticação e autorização são realizadas por meio de *tokens JWT* (MACORATTI.NET, 2024a). Para aumentar a segurança, esses *tokens* têm o conteúdo criptografado utilizando o padrão *JWE*.

Com o uso de *JWE*, foi adotada criptografia assimétrica baseada em chaves *RSA (Rivest-Shamir-Adleman)*, na qual a chave pública é utilizada para criptografar os dados

e a chave privada, mantida em segurança no servidor, é utilizada para decriptá-los. Isso garante que mesmo que o *token* seja interceptado, seu conteúdo permaneça inacessível.

Os tokens são armazenados em cookies configurados com os atributos `HttpOnly` e `Secure`, evitando o acesso por scripts maliciosos e garantindo a transmissão apenas por conexões seguras (HTTPS). Com isso, é possível verificar de forma segura a identidade dos usuários a cada requisição e proteger o acesso a recursos restritos.

#### 3.2.1.2 LGPD

O tratamento de dados pessoais é realizado em conformidade com a *LGPD*, respeitando princípios como finalidade, necessidade, transparência, segurança e responsabilização.

O sistema disponibiliza ao usuário opções de controle sobre seus dados em seu perfil, incluindo a possibilidade de excluir a conta de forma imediata ou desativá-la por um período de 30 dias, após o qual a exclusão será efetuada automaticamente, caso não ocorra a reativação.

O processamento dos dados pessoais é condicionado à autorização expressa do usuário. No momento do registro, é exigida a leitura e aceitação dos Termos de Uso, os quais descrevem de forma clara as finalidades do tratamento, as responsabilidades do sistema e os direitos dos usuários. O consentimento será devidamente registrado e associado à conta do usuário.

Tais medidas asseguram a transparência nas operações, o respeito à autonomia dos usuários e o alinhamento às normas vigentes de privacidade e proteção de dados.

### 3.2.2 Desempenho

Para otimizar o desempenho e minimizar os custos operacionais do *frontend*, foi utilizado o *framework Angular 20* (ANGULAR, 2024). *Angular*, com renderização no lado do cliente, reduz o consumo de recursos no servidor, diminuindo a carga e proporcionando uma melhor experiência ao usuário, com menor tempo de resposta (ANGULAR, 2024).

### 3.2.3 Escalabilidade

A arquitetura do sistema foi projetada para garantir escalabilidade. Todos os serviços estão hospedados na *AWS (Amazon Web Services)* (AMAZON. . . , 2024), permitindo a expansão dinâmica dos recursos computacionais, conforme a demanda do sistema. A *AWS* oferece soluções de escalabilidade horizontal, essenciais para suportar altos volumes de tráfego e grandes quantidades de dados sem comprometer o desempenho.

## 3.3 Banco de Dados

O banco de dados escolhido foi o *MongoDB* (MongoDB, Inc., 2024), pela sua flexibilidade e capacidade de escalabilidade. *MongoDB* (MongoDB, Inc., 2024) permite adicionar novos campos e dados sem reestruturações complexas, o que facilita a evolução do sistema ao longo do tempo. Sua escalabilidade horizontal e resiliência o tornam ideal para sistemas que manipulam grandes volumes de dados e possuem altas taxas de tráfego. Para apoiar o processo de modelagem e documentação do banco de dados, foi utilizada a ferramenta *Hackolade* (DESMARETS, 2016), que oferece recursos compatíveis com bancos orientados a documentos como o *MongoDB* (MongoDB, Inc., 2024), contribuindo para uma estrutura mais clara e bem organizada.

## 3.4 Desenvolvimento da API

A *API* foi desenvolvida utilizando *.NET 9.0 (C#)* (Microsoft, 2024), seguindo o padrão de arquitetura *Web API (Web Application Programming Interface)*. A documentação da *API* é gerada com *Swagger* (SmartBear Software, 2024) e *Data Annotations*, assegurando que os *endpoints* e as interações estejam claramente documentados (MACORATTI.NET, 2024b). O desenvolvimento da *API* foi organizado em três camadas principais:

- **Models**: Representações dos dados persistidos no *MongoDB* (MongoDB, Inc., 2024).
- **Controllers**: Responsáveis por receber as requisições *HTTP (HyperText Transfer Protocol)* e realizar a lógica de controle.
- **DTOs (Data Transfer Objects)**: Usados para transferir dados entre as camadas da aplicação de forma eficiente.

## 3.5 Serviços de Segurança e Autenticação

Foi implementado um serviço de segurança para gerar *hashs* a partir das senhas dos usuários, garantindo que senhas não sejam armazenadas em texto simples no banco de dados. Um gerador de códigos foi desenvolvido para fornecer *tokens* de autenticação para usuários recém-cadastrados ou que realizarem alterações de segurança na conta. Um serviço de e-mail foi integrado para o envio desses códigos, garantindo a validação da autenticidade e propriedade dos e-mails cadastrados.

Essa abordagem visa garantir que o sistema seja robusto, seguro e escalável, atendendo tanto às demandas dos usuários quanto aos requisitos técnicos do projeto.

# 4 IMPLEMENTAÇÃO

A implementação da aplicação é composta por *front-end Angular*, o tracker que será usando pelos usuários, o back-end sendo a *API .NET* (Microsoft, 2024) e o banco de dados *MongoDB* (MongoDB, Inc., 2024), conforme pode ser visto na Figura 1.

Para implementar a *API*, foi utilizado o *.NET 9.0* (Microsoft, 2024), criando um novo projeto *WebAPI* com *controllers*. Este modelo foi escolhido pelo fato de já possuir experiência de mercado de trabalho em *APIs* que utilizavam a mesma estrutura. A forma, a simplicidade e como é estabelecida a organização facilitam a implementação.

Para implementar o *front*, foi utilizado o *Angular 20* (ANGULAR, 2024), criando um novo projeto com rota (para as páginas do sistema). Esse modelo também foi escolhido pelo fato de já possuir experiência de mercado de trabalho. Inicialmente, foi desenvolvida a *controller* dos usuários, chamada *UsersController*, responsável por concentrar todos os *endpoints* relacionados ao gerenciamento de usuários, como criação de conta, confirmação de *e-mail* para ativação, *login* e verificação da validade das credenciais de acesso. No entanto, visando uma melhor separação de responsabilidades e organização do código, as funcionalidades relacionadas à autenticação foram posteriormente movidas para uma nova *controller*, denominada *AuthController*, que passou a centralizar processos como registro, *login*, confirmação de e-mail e *logout*.

Para garantir a segurança dos dados dos usuários, algumas medidas foram adotadas. A primeira foi criar um *service* de segurança com um método para se obter um hash da senha do usuário, com a finalidade de não salvar a senha do usuário no banco de dados, para a própria segurança do mesmo, conforme apresentado no Algoritmo 1. Além disso, foi implementada uma geração de códigos de segurança pseudo-randômica, validação de requisitos mínimos de senha, geração, codificação e decodificação dos *tokens JWT* e *JWE*. Depois, outro *service* foi criado para o envio de emails, onde o objetivo é enviar um código pseudo-randômico para o email registrado, a fim de requisitá-lo no sistema para ativar a conta e validar a propriedade do email registrado.

Após a correta ativação da conta, o usuário consegue realizar login no sistema. Assim que o login é efetuado, o sistema verifica se o usuário está devidamente autenticado. Caso a autenticação seja bem-sucedida, a requisição retorna para o *frontend* o *cookie* correspondente à sessão daquele usuário, contendo o *token JWE*. Juntamente com esse retorno, são enviados também o nível de acesso e o nome de usuário, permitindo que o *frontend* personalize a experiência conforme o nível de acesso.

Mesmo que o usuário, de alguma forma, tente alterar seu nível de acesso pelo *frontend*, o *backend* realiza a reverificação do nível de acesso nos métodos onde isso é necessário, garantindo assim a segurança e integridade do sistema.

Todos os *tokens* gerados são salvos no banco de dados, com o objetivo de, futuramente, possibilitar o planejamento e desenvolvimento de uma estratégia de confiabilidade de dispositivos e revogação de acesso com base nessa confiança.

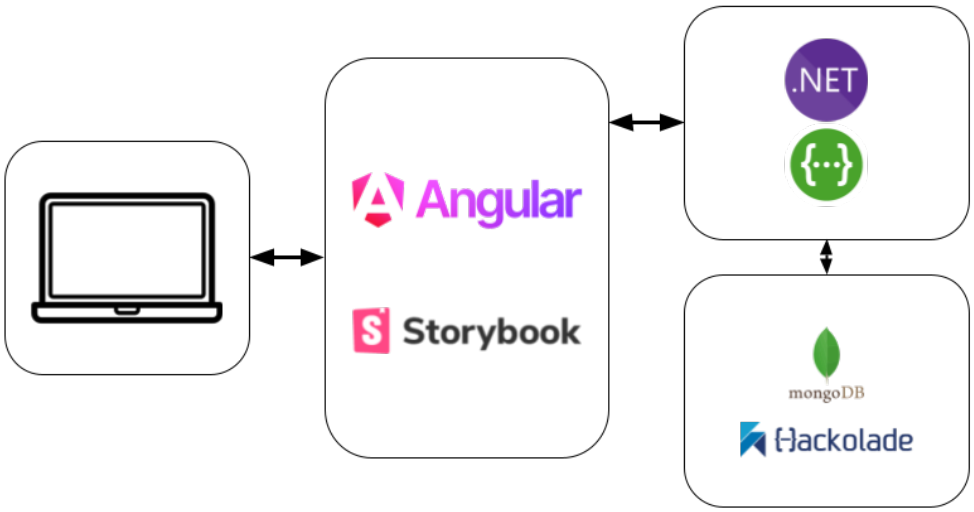



Figura 1 – Arquitetura

**Algoritmo 1:** Método responsável por retornar a hash

```
1 public string CalculateSHA256Hash(string input)
2 {
3     using (SHA256 sha256 = SHA256.Create())
4     {
5         byte[] inputBytes = Encoding.UTF8.GetBytes(input);
6         byte[] hashBytes = sha256.ComputeHash(inputBytes);
7         StringBuilder builder = new StringBuilder();
8         for (int i = 0; i < hashBytes.Length; i++)
9             builder.Append(hashBytes[i].ToString("x2"));
10        return builder.ToString();
11    }
12 }
```

Atualmente, esse controle foi planejado e desenvolvido diretamente na *API .NET*. No entanto, caso essa abordagem não se mostre suficientemente eficaz ou surja a necessidade de uma camada adicional de proteção, também foi considerada a utilização do *WAF (Web Application Firewall)* da *AWS (Amazon Web Services)*, que permite a criação de regras específicas para filtragem de tráfego web, ampliando as possibilidades de segurança do sistema.

# 5 RESULTADOS

A fase de desenvolvimento do sistema alcançou importantes marcos em termos de funcionalidades e infraestrutura, demonstrando a viabilidade e a solidez da arquitetura proposta. A seguir, são detalhados os principais resultados obtidos:

## 5.1 Backend

A API foi desenvolvida utilizando o framework *.NET 9*, com autenticação robusta implementada por meio da utilização de *tokens JWT* com encriptação *JWE*, garantindo tanto a integridade quanto a confidencialidade dos *tokens* de acesso. O sistema utiliza criptografia assimétrica baseada em chaves *RSA* de 2048 bits, garantindo um nível de segurança compatível com boas práticas de mercado.

As funcionalidades disponíveis incluem:

- **Registro de usuário** com envio automático de códigos pseudo aleatórios de confirmação por e-mail e persistência no banco de dados.
- **Confirmação de e-email** utilizando códigos e *endpoint* dedicado.
- ***Login***, com validação e resposta segura por meio de *JWE*.
- ***Logout***, com revogação de sessão ativa.
- ***Swagger*** totalmente funcional e documentado, com suporte a autenticação *JWE* e testes iterativos, facilitando o consumo da *API* por outras camadas da aplicação.

## 5.2 Frontend

A aplicação cliente foi desenvolvida com *Angular 20*, com foco em responsividade, simplicidade e experiência do usuário. Um diferencial da implementação é a centralização das funcionalidades de registro, *login*, e confirmação de *e-mail* em uma única interface dinâmica, onde os diferentes fluxos são controlados por meio de troca de *cards*, reduzindo a navegação e proporcionando uma experiência mais fluida.

Cards disponíveis na página de autenticação:

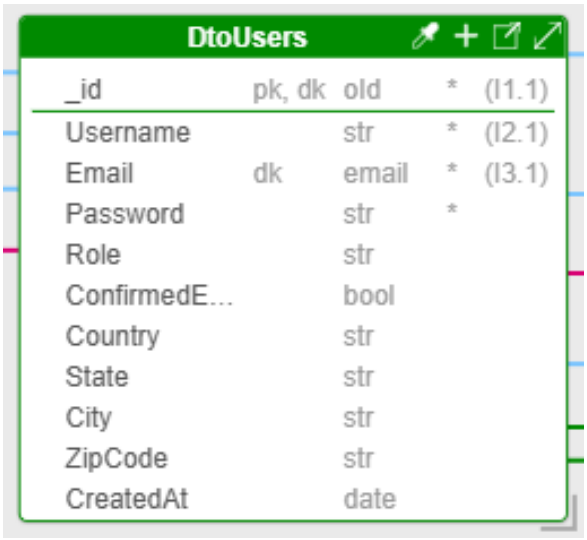
- **Card de *login***, com autenticação direta na API.
- **Card de registro**, com validação de entrada e submissão.
- **Card de confirmação de propriedade de email**, ativado após o envio do código, com feedback visual.

Além disso, foi utilizado o Storybook para documentar visualmente todos os componentes de interface, permitindo a visualização e testes isolados de botões, formulários e inputs, contribuindo para a consistência da UI e a facilidade de manutenção futura.

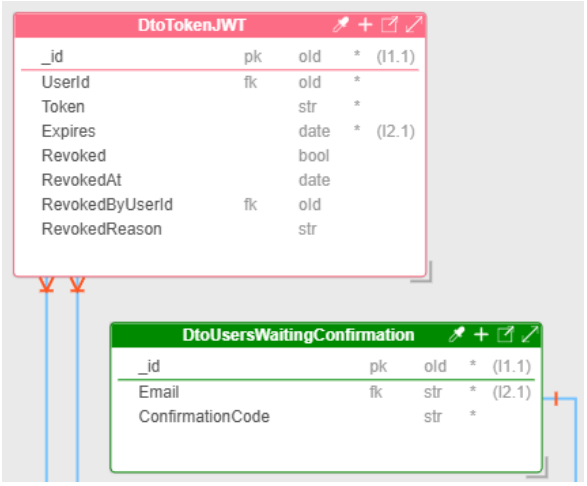
## 5.3 Modelagem e Banco de dados

A estrutura de dados foi modelada utilizando o *Hackolade*, ferramenta especializada em bancos de dados NoSQL. A modelagem contemplou as entidades principais do sistema como:

- DtoUsers;
- DtoTokenJWT;
- DtoUsersWaitingConfirmation;
- DtoComments;
- DtoMedias;
- DtoUsersProgress.




DtoUsers




DtoTokenJWT / DtoUsersWaitingConfirmation

**2.1.2.1.3.2.1 Username properties**

| PROPERTY | VALUE |
|---|---|
| Name | Username |
| Activated | true |
| Type | string |
| Description | The unique name chosen by the user to identify themselves within the system. |
| Min length | 5 |
| Max length | 20 |
| Required | true |
| Primary key | false |

Propriedades da variável Username

**2.1.2.1.4 DtoUsers Indexes**

| PROPERTY | _ID | USERNAME | EMAIL |
|---|---|---|---|
| Name | _id | Username | Email |
| Activated | true | true | true |
| Key | _id('ascending') | Username('ascending') | Email('ascending') |
| Unique | true | true | true |
| Drop duplicates | false | false | false |
| Sparse | false | false | false |
| Background indexing | false | false | false |
| Storage engine | WiredTiger | WiredTiger | WiredTiger |

Índices definidos na coleção *DtoUsers*: _id, Email e Username

Foram definidos os relacionamentos, regras de validação e índices, otimizando a estrutura para consultas rápidas e armazenamento seguro dos dados sensíveis dos usuários.

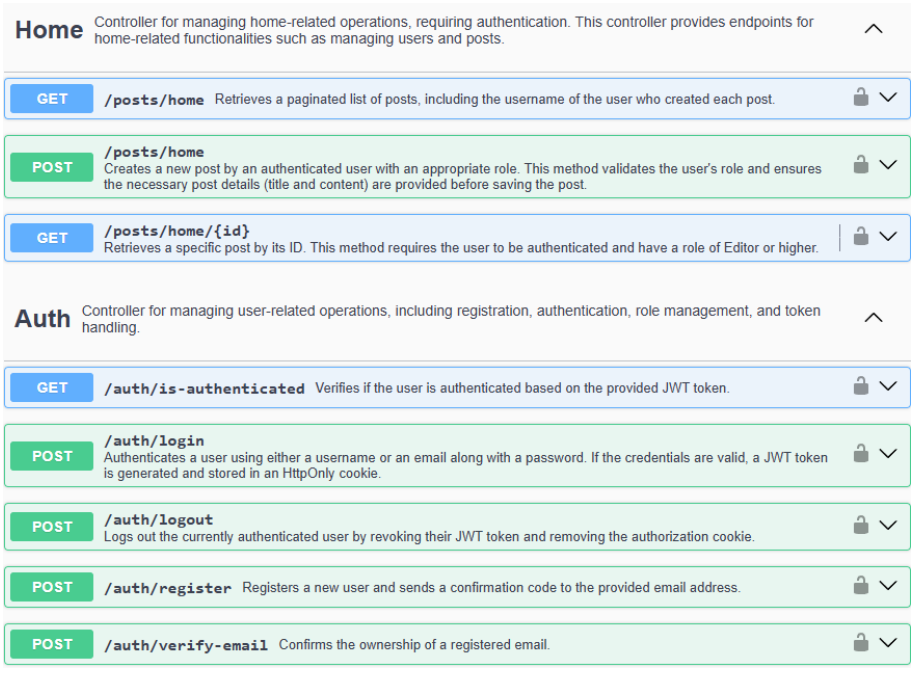
# 6 DOCUMENTAÇÃO

A documentação da aplicação está organizada conforme as camadas da arquitetura:

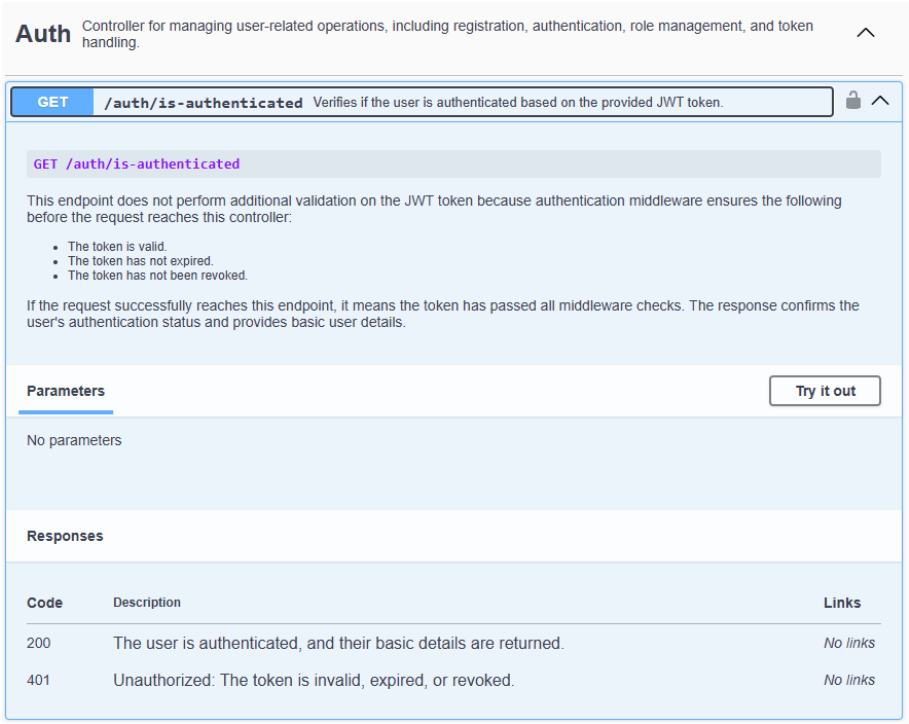
- ***API (.NET)***: Utiliza *Swagger* para documentação automática dos *endpoints REST*. Isso permite a visualização interativa, testes diretos na interface e facilita a comunicação entre equipes técnicas. A documentação é gerada automaticamente a partir dos atributos e comentários nos *controllers* e *services*, conforme ilustrado nas figuras abaixo.

```
/// <summary>
/// Verifies if the user is authenticated based on the provided JWT token.
/// </summary>
/// <remarks>
/// ```http
/// GET /auth/is-authenticated
/// ```
/// This endpoint does not perform additional validation on the JWT token because authentication middleware
/// ensures the following before the request reaches this controller:
/// - The token is valid.
/// - The token has not expired.
/// - The token has not been revoked.
///
/// If the request successfully reaches this endpoint, it means the token has passed all middleware checks.
/// The response confirms the user's authentication status and provides basic user details.
/// </remarks>
/// <returns>
/// Returns a 200 OK response with the authenticated user's information if the token is valid.
/// Returns a 401 Unauthorized response if authentication fails due to an invalid, expired, or revoked token.
/// </returns>
/// <response code="200">The user is authenticated, and their basic details are returned.</response>
/// <response code="401">Unauthorized: The token is invalid, expired, or revoked.</response>
[Authorize, HttpGet, Route("is-authenticated")]
[Produces(ContentTypes.Json)]
public async Task<IActionResult> IsAuthenticated()
{
```

Captura de tela do código da API .NET documentada



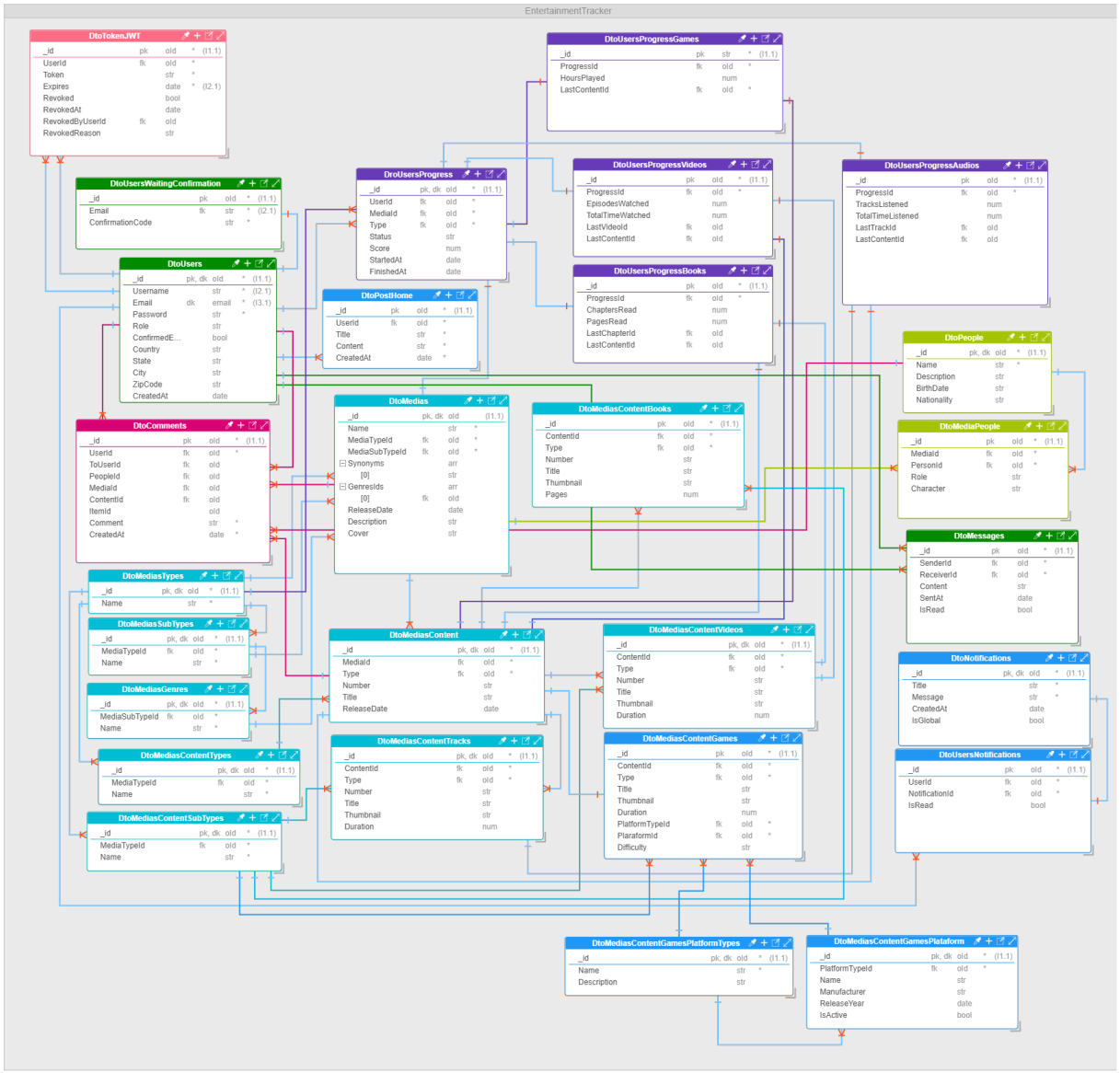

Captura de tela gerada pelo Swagger rodando a API .NET




Exibição detalhada da documentação de um endpoint no Swagger

- ***Front-end (Angular)***: A interface do usuário está documentada utilizando *Storybook*, ferramenta que permite visualizar e testar componentes de forma isolada. Com isso, é possível documentar variações de estado, validar usabilidade e promover reutilização e consistência no design da aplicação (veja os Anexos B e C).

- ***Banco de Dados (MongoDB)***: A modelagem de dados (figura logo abaixo) e os relacionamentos entre coleções são documentados por meio do *Hackolade*, uma ferramenta visual específica para bancos de dados. Essa documentação facilita o entendimento da estrutura dos dados, servindo tanto para desenvolvedores quanto para analistas de dados e responsáveis por compliance (como no caso da LGPD) (veja Anexo D).




Exibição detalhada da modelagem do banco de dados

# 7 CONSIDERAÇÕES FINAIS

O presente trabalho apresentou o desenvolvimento do sistema *Entertainment Tracker*, uma ferramenta voltada à indexação e acompanhamento de conteúdos audiovisuais. A proposta se destaca por abranger múltiplos tipos de mídia em uma única plataforma, suprimindo uma necessidade existente no mercado: a falta de uma plataforma centralizada, flexível e extensível para o gerenciamento de entretenimento multimídia. A análise comparativa com as outras ferramentas existentes mostrou as limitações dos sistemas tradicionais quanto à flexibilidade, foco e restrição a nichos específicos e, em alguns casos, funcionalidades exclusivas apenas para usuários pagantes.

A implementação do sistema foi realizada com base em uma arquitetura moderna escalável, utilizando tecnologias amplamente adotadas na indústria, como *.NET 9.0* para a *API*, *Angular 20* para o *front-end* e *MongoDB* como banco de dados orientado a documentos. Foram observadas boas práticas de engenharia de *software*, segurança da informação e conformidade com a *LGPD*, além da preocupação com a documentação técnica e a manutenção do sistema por meio de ferramentas como *Swagger*, *Storybook*, e *Hackolade*. A modelagem de dados, os mecanismos de autenticação e a estrutura modular da aplicação garantem a continuidade do projeto e sua possível evolução.

Como trabalhos futuros, será feita a finalização de outras funcionalidades planejadas, bem como a realização de testes de usabilidade e validação junto a potenciais usuários. A aplicação poderá ser expandida com recursos adicionais, como mecanismos avançados de recomendação, integração com APIs externas.

Por fim, este trabalho contribui para o campo da Ciência da Computação ao propor e documentar uma solução inovadora e tecnicamente fundamentada, com potencial de aplicação prática no contexto de sistemas de gerenciamento de mídia e plataformas digitais de consumo cultural.

# REFERÊNCIAS

1337X. *1337x*. 2007 – 2024. Disponível em: <https://www.1377x.to/>. Citado na página 21.

AMAZON Web Services. 2024. Disponível em: <https://aws.amazon.com/>. Citado na página 24.

ANGULAR. **Introduction: What is Angular?** 2024. Disponível em: <https://angular.dev/overview>. Citado 2 vezes nas páginas 24 and 27.

DAVINSON, R. **Marketing de afiliados continua em expansão no Brasil**. 2024. Disponível em: <https://valor.globo.com/patrocinado/dino/noticia/2024/08/27/marketing-de-afiliados-continua-em-expansao-no-brasil.ghtml>. Citado na página 20.

DESMARETS, P. **Hackolade**. 2016. Disponível em: <https://hackolade.com/>. Citado na página 25.

Facebook. 2024. Disponível em: <https://www.facebook.com/>. Citado na página 21.

GALITZ, W. O. **The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques**. 3th. ed. New Jersey, USA: Wiley, 2007. Citado na página 21.

Global AD. **Digital Brazil 2023: report anual de plataformas digitais**. 2023. Disponível em: <https://globalad.com.br/blog/digital-brazil-2023/>. Citado na página 19.

Google Inc. **Google**. 2024. Disponível em: <https://www.google.com/>. Citado na página 21.

Instagram. 2024. Disponível em: <https://www.instagram.com/>. Citado na página 21.

KITSU. **Kitsu**. 2024. Disponível em: <https://kitsu.app/>. Citado na página 19.

MACORATTI.NET. **ASP.NET Core - Implementando Json Web Tokens(JWT) - I**. 2024. Disponível em: <https://www.macoratti.net/19/04/aspncore_jwt1.htm>. Citado na página 23.

MACORATTI.NET. **Swagger - Aprimorando a documentação com anotações - II**. 2024. Disponível em: <https://macoratti.net/22/04/swagger_aprdoc2.htm>. Citado na página 25.

Microsoft. **dotNET**. 2024. Disponível em: <https://dotnet.microsoft.com/>. Citado 2 vezes nas páginas 25 and 27.

MongoDB, Inc. **MongoDB**. 2024. Disponível em: <https://www.mongodb.com/>. Citado 2 vezes nas páginas 25 and 27.

reddit. 2024. Disponível em: <https://www.reddit.com/>. Citado na página 21.

SILVA, V. de Souza da. **Entertainment Tracker**. 2024. Disponível em: <https://entertainment-tracker.com>. Citado na página 19.

SKOOB. **SKOOB**. 2024. Disponível em: <https://www.skoob.com.br/>. Citado na página 19.

SmartBear Software. **Swagger**. 2024. Disponível em: <https://swagger.io/>. Citado na página 25.

trakt, inc. **Trakt**. 2010 – 2024. Disponível em: <https://trakt.tv/dashboard>. Citado na página 19.

Unbroken Software, LLC. **LaunchBox**. 2024. Disponível em: <https://www.launchbox-app.com/>. Citado na página 19.

# Anexos

# ANEXO A – EXEMPLO DE TENTATIVA DE SOLICITAÇÃO DE ADIÇÃO DE CONTEÚDO AO BANCO DE DADOS



Gmail **Vinicius Souza da Silva <viniciusufx@gmail.com>**

**Request for X-Men is Closed**
1 mensagem

**help@kitsu.io** <help@kitsu.io> 29 de abril de 2024 às 12:52
Para: viniciusufx@gmail.com

## Request for X-Men is Closed.

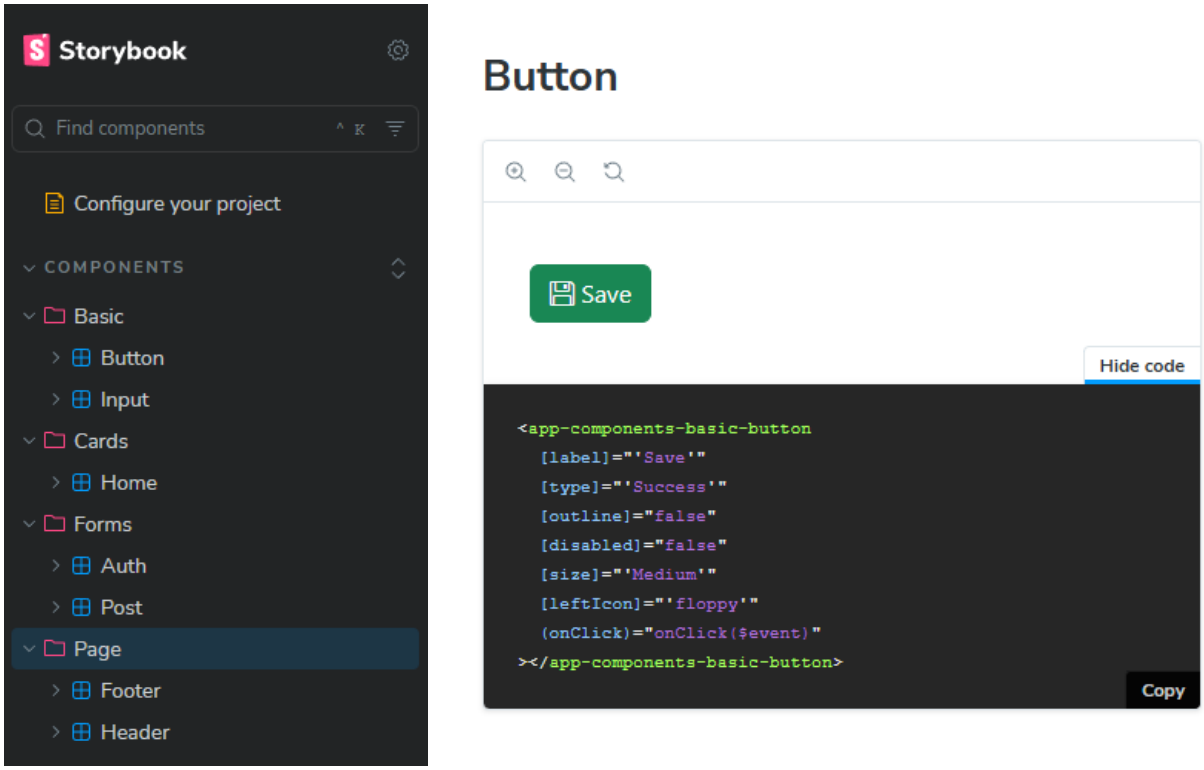
We wanted to let you know that your request has been moved to a status of: Closed.

Additional Comments:

For western animated content to be added you need to provide a source that the title in question is anime influenced.

Have a great day,
Erik_The_Red - Kitsu Database Mod

# ANEXO B – CAPTURA DE TELA GERADA PELO STORYBOOK



# ANEXO C – CAPTURA DE TELA GERADA PELO STORYBOOK

| Name | Description | Default | Control |
|---|---|---|---|
| **INPUTS** | | | |
| **label** | Text displayed inside the button<br>`string` | | Button |
| **type** | Defines the button style (e.g., Primary, Danger, Success, etc.)<br>`"Primary"` `"Secondary"` `"Success"` `"Danger"` `"Warning"` `"Info"` `"Light"` `"Dark"` Show 1 more... | | ◉ Primary ○ Secondary ○ Success ○ Danger ○ Warning ○ Info ○ Light ○ Dark ○ Link |
| **outline** | If true, renders the button with an outline style<br>`boolean` | `false` | False True |
| **disabled** | If true, disables the button interaction<br>`boolean` | `false` | False True |
| **size** | Defines the button size (e.g., Small, Medium, Large)<br>`"Small"` `"Medium"` `"Large"` | | ○ Small ◉ Medium ○ Large |
| **leftIcon** | Optional icon to be displayed on the left side of the button<br>`string` | | Set string |
| **rightIcon** | Optional icon to be displayed on the right side of the button<br>`string` | | Set string |
| **OUTPUTS** | | | |
| **onClick** | Event triggered when the button is clicked<br>`EventEmitter` | | Edit JSON string... |

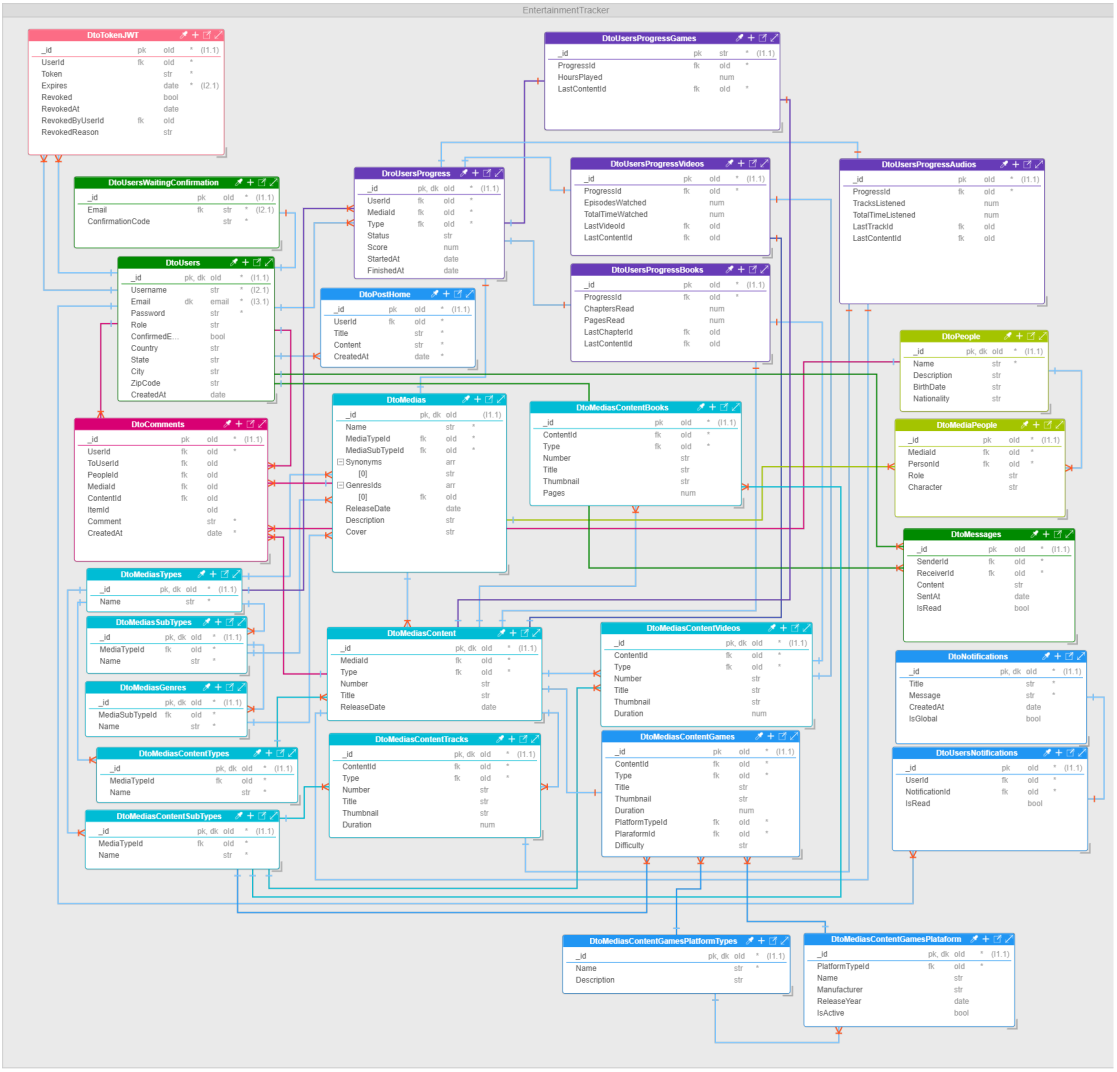
# ANEXO D – CAPTURA DE TELA GERADA PELO HACKOLADE

Hackolade

## MongoDB Physical Model

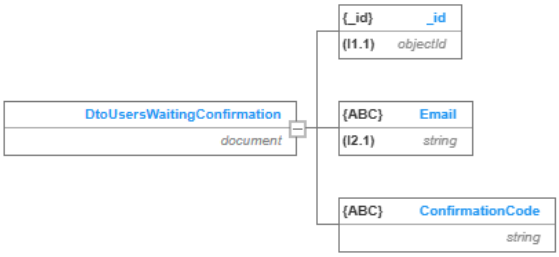
**Schema for:**

Model name: EntertainmentTracker
Author: Vinicius de Souza da Silva
Version: 1
File name: Entertainment Tracker.hck.json
File path: C:\Users\viniciusufx\Desktop\Entertainment Tracker.hck.json
Printed On: Fri Apr 18 2025 19:41:57 GMT-0300 (Horário Padrão de Brasília)
Created with: Hackolade - Polyglot data modeling for SQL and NoSQL databases, APIs, and storage formats in RDBMS



#### 2.1.2.2 Collection **DtoUsersWaitingConfirmation**

##### 2.1.2.2.1 **DtoUsersWaitingConfirmation** Tree Diagram




##### 2.1.2.2.2 **DtoUsersWaitingConfirmation** Properties

| PROPERTY | VALUE |
|---|---|
| Collection name | DtoUsersWaitingConfirmation |
| Activated | true |
| Database | EntertainmentTracker |
| Storage engine | WiredTiger |
| Validation level | Off |
| Validation action | Warn |
| Additional properties | false |

##### 2.1.2.2.3 **DtoUsersWaitingConfirmation** Fields

| FIELD | TYPE | REQ | KEY | DESCRIPTION | COMMENTS |
|---|---|---|---|---|---|
| _id | objectId | true | pk | Unique identifier for the pending confirmation entry. | |
| Email | string | true | fk | Email address of the user who is waiting for confirmation. Acts as a foreign key linked to the main user record. | |
| ConfirmationCode | string | true | | Code sent to the user's email, used to verify and activate the account. | |

###### 2.1.2.2.3.1 Field _id

####### 2.1.2.2.3.1.1 _id properties

| PROPERTY | VALUE |
|---|---|
| Name | _id |
| Activated | true |
| Type | objectId |
| Description | Unique identifier for the pending confirmation entry. |
| Required | true |
| Primary key | true |