Illiad: Sistema web de compartilhamento de vídeos de código aberto

Gustavo D. M. S. Duarte[1], **Fernando Sambinelli**[1]

[1]Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas –
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP)
Campus Hortolândia – SP - Brasil

Abstract. *Given the growing demand for digital video-sharing platforms, this paper presents the development of Illiad, an open-source web system for video sharing. The objective of this work is to provide an accessible and free alternative for users who wish to publish and consume videos without the restrictions of traditional platforms. The project was developed using technologies such as Spring, Spring Security, JPA, PostgreSQL, HTML, CSS, and JavaScript.*

Resumo. *Diante da crescente demanda por plataformas de compartilhamento de conteúdo digital, este artigo apresenta o desenvolvimento do Illiad, um sistema web de código aberto para compartilhamento de vídeos. O objetivo do trabalho foi criar uma alternativa acessível e livre para usuários que desejam publicar e consumir vídeos sem as restrições das plataformas tradicionais. O projeto foi desenvolvido utilizando as tecnologias Spring, Spring Security, JPA, PostgreSQL, HTML, CSS e Javascript.*

1. Introdução

O compartilhamento de vídeos tornou-se essencial, atendendo propósitos diversificados como entretenimento, educação e uso profissional. A maior plataforma de compartilhamento de vídeos online, *YouTube*, acumula mais de 32 bilhões de visitas mensais no seu *website* [Similarweb 2024], sendo um sistema de geração de conteúdo pelo usuário (*UGC*, do inglês *User-Generated Content*) que permite que usuários compartilhem conteúdo na plataforma para outros usuários.

As principais plataformas de compartilhamento de vídeos, como *YouTube*, *Dailymotion* e *Rumble*, operam com código fechado, o que limita o acesso ao código-fonte das plataformas, por sua vez, softwares de código aberto são mais flexíveis e – a depender da licença – possibilitam o acesso, modificação, estudo e distribuição do código fonte parcial ou total do projeto [Free Software Foundation 2024]. A natureza do código fechado limita a capacidade de auditoria dos usuários, o que pode gerar preocupações em questões de privacidade. Casos como o escândalo envolvendo o *Facebook* e a *Cambridge Analytica* demonstram os perigos da manipulação de dados, onde informações pessoais de milhões de usuários foram usadas sem consentimento para influenciar processos eleitorais [Solon 2018].

Nesse contexto, softwares que adotam código aberto aparecem como uma solução viável, oferecendo maior transparência e segurança. Um exemplo disso é o caso do *Audacity*, aplicativo de edição de áudio, que após uma atualização em 2021 passou a coletar dados pessoais dos usuários. A transparência proporcionada pelo código aberto permitiu

que essas mudanças fossem rapidamente identificadas por colaboradores do projeto, destacando a importância de alternativas *open-source* para garantir a privacidade e segurança dos usuários [Humphries 2021].

Diante dessas questões, propõe-se o desenvolvimento de um sistema web de código aberto para compartilhamento de vídeos, com objetivo de oferecer uma alternativa transparente e segura às plataformas proprietárias, permitindo que o usuário tenha maior controle sobre seus dados e promovendo a privacidade no meio digital.

2. Referencial teórico

Nesta seção são apresentadas as referências teóricas utilizados para o desenvolvimento deste projeto.

2.1. Sistemas de compartilhamento de vídeos

Sistemas de compartilhamento de vídeos, *video sharing sites* ou *online video platforms*, são sistemas que permitem que usuários disponibilizem vídeos para que outras pessoas ao redor do mundo possam assistir. Em comparação ao tradicional serviço cabeado, os conteúdos disponíveis no site são mais recentes e práticos [Xu et al. 2018], necessitando ao usuário apenas uma conexão a internet e um aparelho que tenha acesso a um navegador web.

Em 2022, o instituto de pesquisa *Pew Research* entrevistou mais de 10.000 americanos, dos entrevistados com acesso a internet, 49% obtinha notícias do dia-a-dia utilizando um computador, *tablet* ou celular, e 82% destes utilizavam a plataforma *YouTube* [Pew Research 2022], a maior plataforma de compartilhamento de vídeos. Portanto, os sistemas de compartilhamento de vídeo são amplamente utilizados e desempenham um papel fundamental tanto na economia quanto na sociedade.

2.2. Software livre

O conceito de software livre é fundamental para o desenvolvimento de plataformas abertas como o *Illiad* pois possibilita que o usuário possa modificar, estudar e distribuir o código fonte do software. A *Free Software Foundation* (*FSF*), fundação responsável pelo início do movimento de software aberto e responsáveis pelo desenvolvimento do sistema operacional GNU, vê software como uma ferramenta para empoderar as pessoas e garantir uma sociedade justa ao contrário da visão tradicional de software como um produto a ser comercializado por empresas privadas [Free Software Foundation 2024].

A *FSF* promove o uso de software livre como uma forma de assegurar a liberdade digital, mas também combate práticas que ameaçam essa liberdade, como o gerenciamento de restrições digitais (*DRM*). Digital Rights Management, ou *DRM*, limita a distribuição e modificação de conteúdos digitais, contrário aos fundamentos do software livre que promovem a liberdade digital [Stallman 2006]. A FSF luta pela transparência de software e pela autonomia dos usuários, buscando por um ambiente onde softwares podem ser compartilhados de forma livre, estimulando a colaboração e a participação de pessoas por um bem comum. O Illiad segue os princípios de código aberto, disponível no *GitHub* sob a licença *GNU General Public License 3* (*GPLv3.0*).

O movimento de software livre é essencial para combater o monopólio digital – que ocorre quando poucas empresas controlam e limitam o uso de tecnologias essenciais – seja por meio de preços exorbitantes e assinaturas mensais, ou censura e limitação

de uso em certas regiões. O movimento de código aberto busca democratizar a tecnologia [The Open Source Way 2019], garantindo que todos – seja desenvolvedor ou não – tenham acesso ao código-fonte e utilizem-o de forma a atender às suas próprias necessidades.

2.3. Spring

O *Spring Framework* é um *framework* amplamente utilizado para o desenvolvimento de aplicações em *Java*, especialmente em ambientes corporativos, por sua flexibilidade e capacidade de simplificar o desenvolvimento de software robusto e escalável [Johnson 2004]. Sua estrutura modular permite que os desenvolvedores utilizem apenas os componentes necessários, promovendo uma abordagem leve e de fácil manutenção.

Entre os principais módulos, destacam-se o *Spring Core*, que fornece o núcleo de injeção de dependência e inversão de controle (*IoC*), e o *Spring MVC*, amplamente utilizado para construção de aplicações web seguindo o padrão *Model-View-Controller* (*MVC*). Além disso, o *Spring* também oferece suporte para transações e integração com bancos de dados, utilizando JPA, e segurança, por meio do módulo *Spring Security*, o que o torna uma escolha robusta para o desenvolvimento de sistemas complexos, como plataformas de compartilhamento de vídeos [Souza 2024].

2.4. API REST

REST, ou *Representational State Transfer*, é um estilo arquitetural para sistemas distribuídos, baseado em princípios como a utilização de recursos identificáveis por *URIs* e operações padronizadas por métodos HTTP [Massé 2011].

Essa arquitetura baseia-se em princípios como a utilização de métodos *HTTP* (*GET, POST, PUT, DELETE*) para operações de manipulação de dados e em uma estrutura de *URIs* intuitiva que facilita a identificação de recursos. Além disso, *APIs REST* são projetadas para serem *stateless*, ou seja, cada requisição do cliente ao servidor é independente, contendo todas as informações necessárias para ser processada, o que torna a comunicação mais eficiente e escalável.

A arquitetura *REST* desempenha um papel fundamental na construção das APIs do sistema, permitindo operações de criação, leitura, atualização e remoção (*CRUD*) de vídeos, interações de usuários e gerenciamento de conteúdo de forma eficiente. Com uma *API REST* bem projetada, o *Illiad* pode oferecer um serviço escalável e de fácil integração com outras aplicações e outros clientes *front-end*.

2.5. PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados (*DBMS*) – um programa que gerencia e armazena um grande conjunto de dados – de código aberto amplamente reconhecido por sua robustez e escalabilidade [PostgreSQL 2024]. Criado inicialmente em 1986 no projeto *POSTGRES* da Universidade da Califórnia, Berkeley, o *PostgreSQL* evoluiu e tornou-se uma das mais principais opções para o desenvolvimento de aplicações que exigem alta performance e confiabilidade.

Para o *Illiad*, que demanda gestão eficiente de grande volume de vídeos e interações entre usuários, a escalabilidade do PostgreSQL é essencial. Seus recursos – como

replicação, particionamento de tabelas e otimização de consultas – permitem que o sistema suporte operações em grande escala, garantindo a escalabilidade horizontal e vertical. Dessa forma, o PostgreSQL assegura uma base robusta para atender às exigências de performance e crescimento do *Illiad*.

3. Trabalhos correlatos

Nessa seção são apresentados os trabalhos correlatos utilizados para o desenvolvimento deste projeto.

3.1. YouTube

YouTube é um sistema de compartilhamento de vídeos amplamente reconhecida e utilizada globalmente [Similarweb 2024]. Fundada em 2005, a plataforma permite que usuários carreguem, visualizem, avaliem, comentem e compartilhem vídeos. Ela suporta uma vasta gama de formatos de vídeo e oferece funcionalidades robustas para gerenciamento de conteúdo e interação social.

A plataforma não apenas transformou a maneira como consumimos conteúdo de vídeo, mas também influenciou significativamente o desenvolvimento de outras plataformas de compartilhamento de vídeo. Seu sucesso demonstra a viabilidade e o potencial de modelos de negócios baseados em conteúdo gerado pelo usuário.

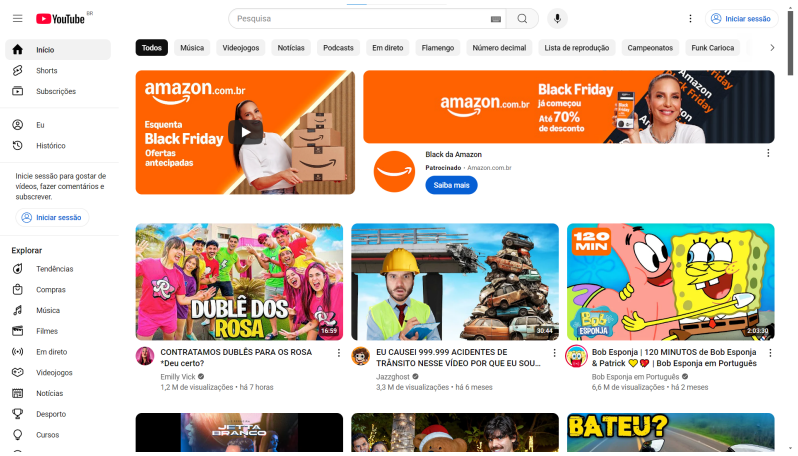

Figura 1. Tela inicial do *YouTube*.

As características principais incluem o *upload* fácil e intuitivo de vídeos, a interação entre usuários e criadores de conteúdo e gerenciamento robusto de canais. No entanto, por ser uma plataforma centralizada de código fechado e voltada para a monetização por anúncios, a exibição de propagandas intrusivas impacta severamente a experiência de usuário e as restrições no controle dos dados dos usuários podem gerar insegurança e receio em relação ao uso na plataforma. Essas lacunas incentivaram a busca por alternativas que priorizem a liberdade do usuário.

3.2. Odysee

Odysee é uma plataforma de mídia baseado em *blockchain*, conhecida principalmente por hospedar vídeos. Todo o conteúdo encontrado na plataforma é hospedado na rede

LBRY, inspirado pela rede *Bitcoin*, oferecendo uma camada de transparência incomum em plataformas convencionais [Odysee 2024].

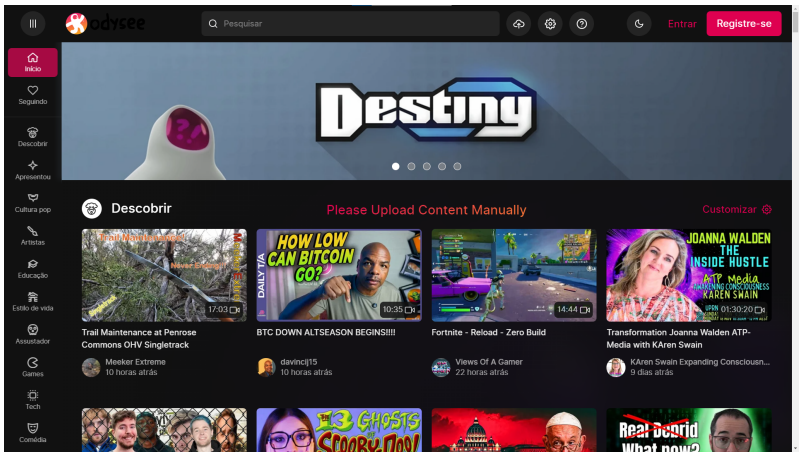

Figura 2. Tela inicial do *Odysee*.

Odysee destaca-se como uma alternativa que promove a liberdade de expressão e a diversidade de conteúdo, reafirmando seu compromisso com uma experiência de usuário transparente em um ambiente digital cada vez mais monopolizado.

Entre os diferenciais da plataforma destacam-se a ausência de propagandas, código-fonte aberto e descentralização do conteúdo. Contudo, é dependente a *block-chain* o que pode dificultar a adoção em larga escala, principalmente para usuários menos familiarizados com conceitos técnicos.

3.3. Comparação entre sistemas

Para facilitar a compreensão das semelhanças e diferenças entre os sistemas de compartilhamento de vídeos introduzidos e o que foi desenvolvido neste trabalho, a Tabela 1 apresenta um comparativo das funcionalidades oferecidas por cada plataforma. Essa análise permite identificar as características que distinguem cada sistema e como elas se alinham às necessidades específicas dos usuários.

Tabela 1. Análise comparativa entre os sistemas.

Funcionalidades	*YouTube*	*Odysee*	*Illiad*
Upload de vídeos	Sim	Sim	Sim
Web	Sim	Sim	Sim
Comentários	Sim	Sim	Não
Open-source	Não	Sim	Sim
Sem anúncios (*ad-free*)	Não	Sim	Sim
Servidor centralizado	Sim	Não	Sim

A Tabela 1 compara as plataformas de compartilhamento de vídeos, destacando critérios como modelo de negócios, privacidade e acessibilidade. O *upload* de vídeos está presente em todas as plataformas, permitindo que usuários postem seus próprios vídeos, acesso via web, também presente em todos os sistemas analisados, significa que todas as

plataformas podem ser acessadas por um navegador, e a funcionalidade de comentários permite interações entre os usuários, não presente no *Illiad*.

Tratando de código aberto, o *YouTube* não permite acesso ao código fonte, enquanto *Odysee* e *Illiad* são plataformas *open-source*, oferecendo mais transparência a seus usuários. Em questão de monetização o *Illiad* e o *Odysee* não incluem propagandas nos vídeos, ao contrário do *YouTube*. Quanto à arquitetura, o *YouTube* e o *Illiad* utilizam servidores centralizados, enquanto o *Odysee* adota uma arquitetura descentralizada baseada em *blockchain*.

O *Illiad* adota uma abordagem híbrida que busca oferecer acessibilidade, transparência e simplicidade para o usuário final, diferenciando-se do *Youtube* por ser um sistema de código aberto e pela ausência de propagandas, e do *Odysee* pois adota uma arquitetura cliente-servidor centralizada e é independente da tecnologia *blockchain*.

4. Metodologia

Foi adotada a metodologia incremental, pois ela permite o desenvolvimento iterativo do sistema, possibilitando entregas parciais e contínua validação das funcionalidades implementadas [Sommerville 2011]. Os artefatos gerados incluem o canvas de valor, requisitos funcionais e não funcionais, histórias de usuário, diagrama entidade relacionamento, diagrama de casos de uso e protótipo de telas.

O processo iniciou-se com o levantamento de requisitos, que determinou quais funcionalidades seriam adotados no sistema; seguida da prototipagem de telas, que foram utilizadas como referência para a usabilidade do sistema e destacariam possíveis requisitos não levantados anteriormente; na codificação do sistema foram aplicadas as funcionalidades principais como cadastro e login de usuários e visualização e postagem de vídeos; por fim, os testes da *API* validaram a consistência e asseguraram o comportamento esperado do sistema.

O *back-end* foi desenvolvido em *Java*, utilizando o *framework Spring* adotando a arquitetura *API REST*, enquanto o *front-end* foi construído com *HTML, CSS* e *JavaScript* com *AJAX*. Ferramentas utilizadas incluem: *Docker* para criação do contêiner de banco de dados, *VSCodium* e *IntelliJ Idea* para desenvolvimento de código e *Insomnia* para testes da *API*.

5. Desenvolvimento

Nesta seção é apresentado o desenvolvimento desse projeto, descrevendo as atividades realizadas em cada etapa do processo de criação.

5.1. Canvas de valor

A primeira etapa do desenvolvimento do projeto foi o desenvolvimento de um canvas de valor. O canvas de valor, idealizado por Alexander Osterwalder no livro *Value Proposition Design* [Osterwalder et al. 2014], é uma modelo de negócios que permite auxiliar os valores e necessidades do produto e dos clientes, expondo as dores, ganhos e as tarefas. No contexto do desenvolvimento da plataforma *Illiad* o canvas de valor foi utilizado para alinhar as funcionalidades da plataforma com as expectativas do público-alvo.

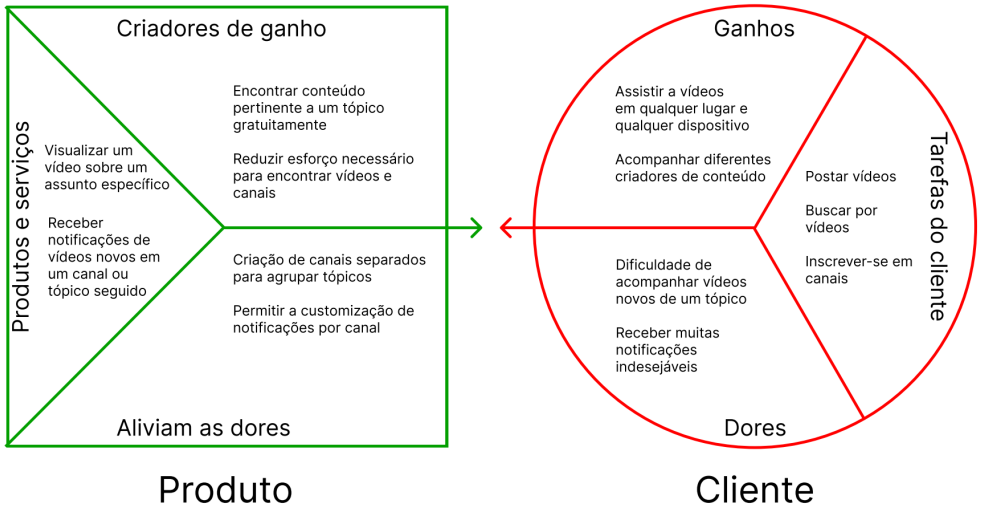

Figura 3. Canvas de valor.

O canvas de valor foi fundamental para identificar as principais funcionalidades e recursos que o sistema deveria possuir para atender as necessidades do cliente. Contribuiu também, para priorizar elementos essenciais no desenvolvimento para atender os requisitos de usuário e usabilidade do sistema.

5.2. Requisitos funcionais e não funcionais

Conforme descrito por Sommerville [Sommerville 2011], os requisitos representam as características e restrições que um sistema qualquer deve atender para satisfazer as necessidades dos usuários e cumprir os objetivos do projeto. Eles podem ser divididos em duas categorias: requisitos funcionais, especificam os serviços e comportamentos do sistema, e requisitos não funcionais, definem propriedades internas como desempenho, confiabilidade e segurança.

Esses requisitos formam a base do desenvolvimento, servindo como um guia para projetar, implementar e validar o sistema. A clareza na definição dos requisitos é essencial para evitar ambiguidades e assegurar que todas as partes interessadas estejam alinhadas em relação ao que deve ser entregue, garantindo um desenvolvimento eficiente e direcionado.

Tabela 2. Requisitos funcionais do *Illiad*.

ID	Nome	Descrição
1	Visualizar vídeo	Um usuário pode visualizar um vídeo disponível
2	Gerenciar vídeo	Um canal pode postar, editar e deletar um vídeo
3	Criar canal	Um usuário pode cadastrar um canal
4	Configurar canal	Um usuário pode editar e deletar informações de seu canal
5	Seguir canal	Um usuário pode seguir um canal

Tabela 3. Requisitos não funcionais do *Illiad*.

ID	Nome	Descrição
1	Acesso à internet	O usuário precisa de acesso à internet para acessar o sistema
2	Segurança	O sistema deve possuir autenticação de usuário e criptografia de dados pessoais
3	Performance	O sistema deve apresentar alta performance para assegurar uma experiência de usuário fluida.

A Figura 4 ilustra um diagrama de fluxo de armazenamento e postagem de vídeos que apresenta as etapas principais envolvidas no processo de *upload* do conteúdo pelo usuário até sua disponibilização para visualização na plataforma. Descrevendo como os vídeos são enviados, armazenados no servidor e disponibilizados para acesso pelos demais usuários. Esse fluxo garante eficiência e confiabilidade no gerenciamento de vídeos, alinhando-se às necessidades funcionais do sistema.

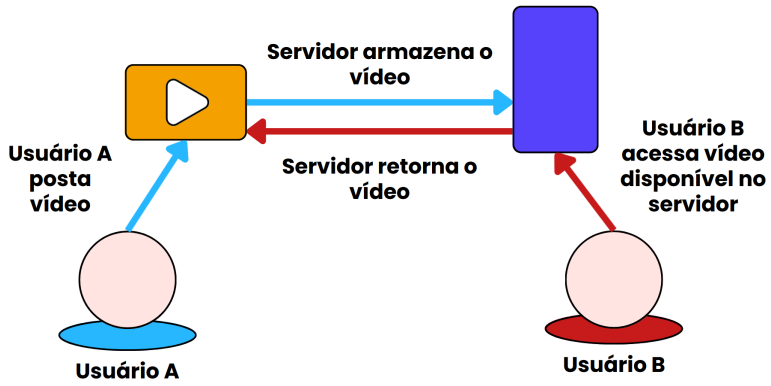

Figura 4. Diagrama de fluxo de *upload*, armazenamento e compartilhamento de vídeos entre usuários em plataformas de vídeo online.

5.3. Histórias de usuário

As histórias de usuário são fundamentais no desenvolvimento, elas descrevem funcionalidades essenciais para um usuário ou um cliente [Cohn 2004]. Elas são usadas para destacar as necessidades e expectativas dos usuários finais, orientando o desenvolvimento das funcionalidades mais impactantes de um sistema. Uma história segue o formato "Como [tipo de usuário], eu quero [necessidade], para que eu possa [benefício]".

Todas histórias de usuário possuem elementos em comum como: título, descreve a funcionalidade de forma sucinta; descrição, detalha o tipo de usuário, sua necessidade e o benefício que essa funcionalidade vai trazer ao usuário; critérios de aceitação, usados para definir as condições que devem ser atendidas para que a funcionalidade seja considerada completa. O uso desses elementos é crucial e garante uma documentação consistente durante todo o projeto e auxilia no desenvolvimento de funcionalidades consideradas prioridade.

A seguir estão descritas três das histórias de usuário mais importantes do projeto e todas as histórias de usuário podem ser acessadas no Anexo I deste trabalho.

5.3.1. Visualizar vídeo

“Como usuário, eu quero assistir um vídeo na plataforma, para que eu possa consumir conteúdo de forma fácil e intuitiva”.

Critérios de aceitação:

- O vídeo deve ser reproduzido em um *player* com opções de tocar e pausar, volume e tela cheia;
- O sistema deve exibir informações sobre o vídeo, como título, descrição, data de postagem, número de visualizações e nome do canal;
- O contador de visualizações deve ser atualizado automaticamente sempre que o vídeo for assistido.

5.3.2. Postar vídeo

“Como usuário, eu quero postar um vídeo no meu canal, para que eu possa compartilhar conteúdo com outros usuários”

Critérios de aceitação:

- O usuário deve estar autenticado no sistema;
- O sistema deve permitir que o usuário insira um título, uma descrição e selecione um arquivo de vídeo para *upload*;
- O vídeo postado deve ser processado e armazenado no sistema, garantindo que esteja disponível para visualização por outros usuários;
- O usuário deve receber uma confirmação visual de que o vídeo foi postado com sucesso.

5.3.3. Gerenciar canal

“Como criador de conteúdo, eu quero poder alterar e deletar dados do meu canal para que eu possa manter as informações atualizadas e ter controle total sobre o meu canal."

Critérios de aceitação:

- O usuário deve estar logado em uma conta existente;
- O sistema deve impedir o usuário de editar dados pessoais como nome do canal ou data de nascimento repetidas vezes em um curto período de tempo;
- O usuário terá de autenticar-se novamente se ele desejar deletar sua conta.

5.4. Prototipação de interfaces de usuário

Nessa etapa foi feita a prototipação da interface de usuário com objetivo de proporcionar uma melhor experiência de usuário (*UX*) e interface de usuário (*UI*) web. Para isso, utilizou-se a ferramenta Figma, que permite a criação de protótipos de tela, facilitando a visualização da estrutura e interação da interface de forma prática e eficiente.

O Figma foi escolhido devido a sua capacidade de criar protótipos rápidos e de fácil modificação e facilitar a realização de testes de usabilidade antes da implementação

final no *front-end*. Essa abordagem foi importante para garantir que a interface atendesse às necessidades dos usuários e proporcionasse uma navegação intuitiva e fluida, ajudando a identificar e corrigir possíveis falhas no design.

A Figura 5 apresenta a tela inicial contendo uma *grid* de vídeos recomendados para o usuário, com duas abas de navegação, para tornar a experiência mais intuitiva. Cada vídeo é apresentado com uma imagem de miniatura (*thumbnail*), título, nome do canal e ícone do canal.

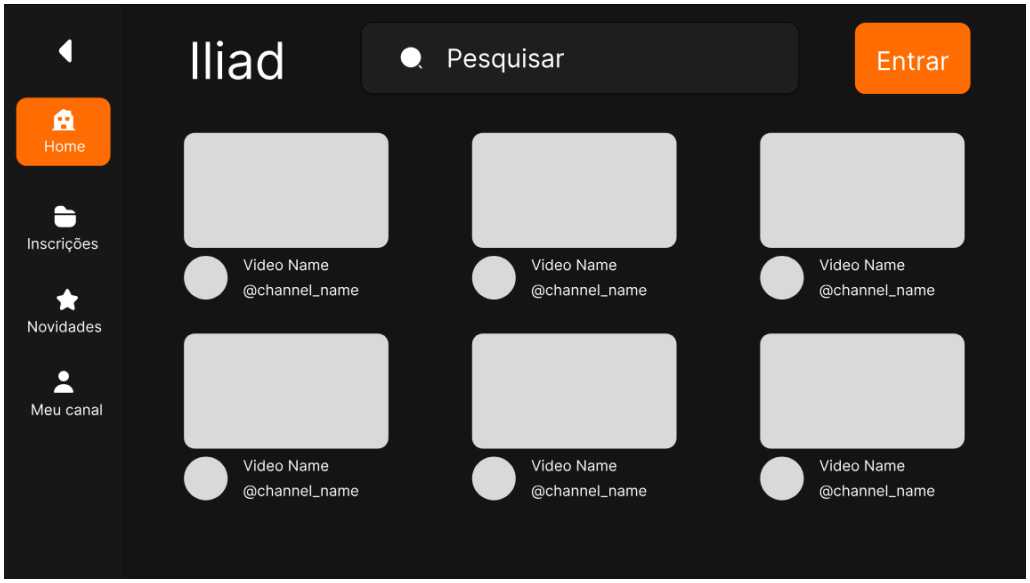

Figura 5. Protótipo da tela inicial.

A tela da Figura 6 é dedicada à reprodução de um vídeo, destacando o conteúdo principal com uma interface limpa e funcional. Abaixo do vídeo, são exibidas opções como título, descrição, gostei e não gostei, e à lateral, sugestões de vídeos relacionados.

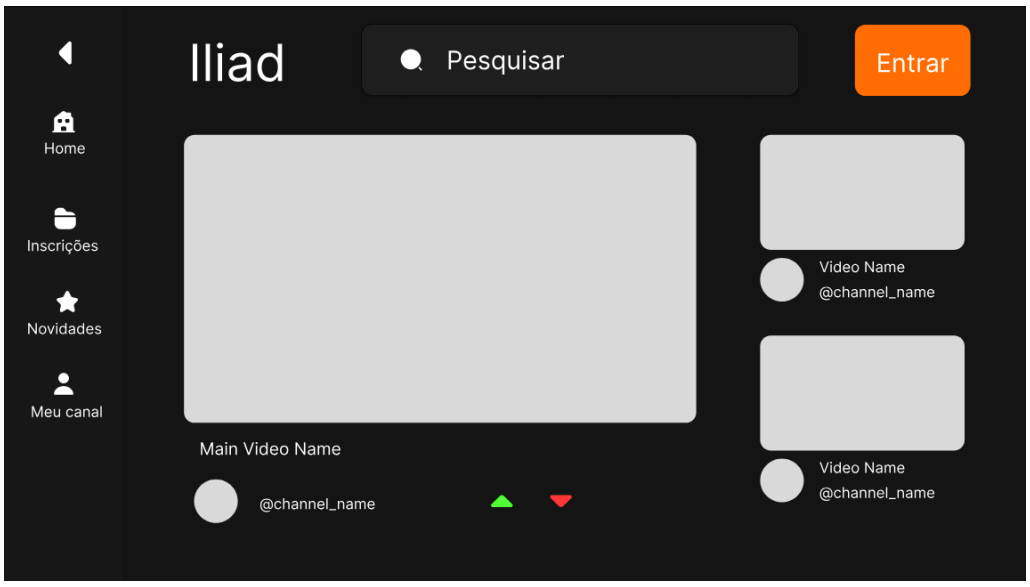

Figura 6. Protótipo da tela de visualização de vídeo.

A tela de canal (Figura 7) proporciona uma visão do perfil do criador de conteúdo, com diversas abas para separar o conteúdo: destaque, para os vídeos destacados pelo canal; recente, os vídeos postados pelo canal em ordem decrescente; *playlists*, lista de vídeos configuradas pelo canal; sobre, informações sobre o canal; e comunidade, seção para comunicação entre o criador de conteúdo e outros usuários.

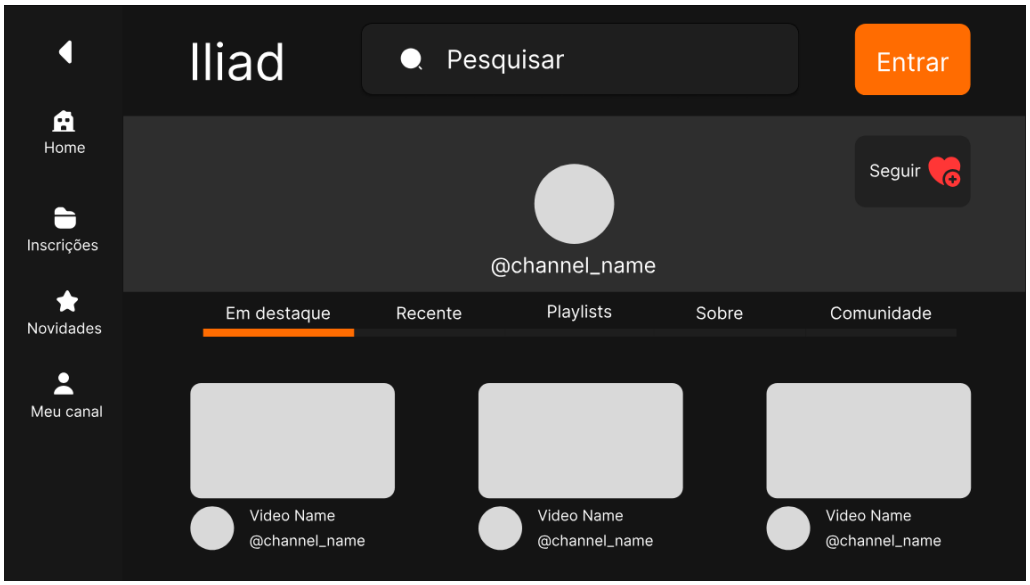

Figura 7. Protótipo da tela canal.

5.5. Modelagem de dados

Para o modelagem de dados do sistema foi utilizado a aplicação web *dbdiagram.io* que facilita a criação de Diagramas Entidade Relacionamento.

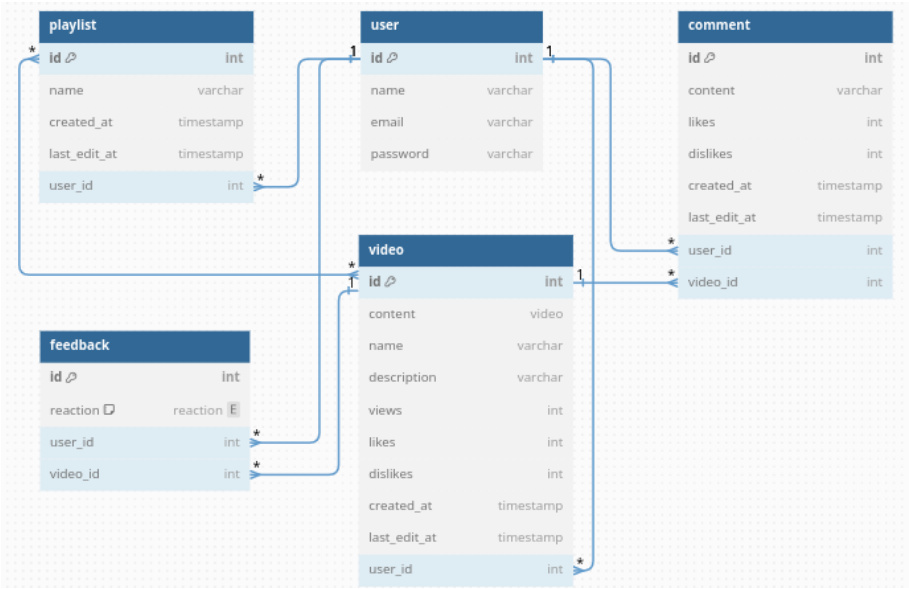

Figura 8. Diagrama Entidade Relacionamento do *Illiad*.

No diagrama entidade relacionamento da Figura 8 foi criado 4 (quatro) entidades: *user*, que representa a conta de autenticação do usuário, com os atributos *name, email* e *password*; *playlist*, que representa as *playlists* de um usuário; video, que representa todo o conteúdo de um vídeo como conteúdo, título, descrição e visualizações; *comment*, um comentário de um usuário em um vídeo qualquer; e *feedback*, que representa o *feedback* (gostei e não gostei) de um usuário em um vídeo.

5.6. API RESTful

No *back-end* do sistema foi desenvolvida uma *API RESTful* utilizando *Java* com os *frameworks Spring, Spring Security* e *Spring Data JPA*. A *API* foi projetada seguindo os princípios REST, garantindo uma arquitetura escalável, modular e de fácil manutenção. O *Spring* foi escolhido por sua capacidade de simplificar o desenvolvimento de aplicações robustas, oferecendo suporte nativo para injeção de dependências, configuração simplificada e integração com outras bibliotecas do ecossistema *Spring*.

Para garantir a segurança da aplicação, foi implementado um sistema de autenticação baseado em *JSON Web Tokens* (*JWT*). O *JWT* é um padrão aberto que permite a transmissão segura de informações entre o cliente e o servidor. No *Illiad*, o *JWT* é utilizado para autenticar usuários e autorizar o acesso a recursos protegidos (como postagem e deleção de vídeos). A autenticação envolve validar o *email* e a senha provida pelo usuário, a seguida da geração de um *token* válido contendo informações essenciais do usuário e a verificação desse *token* em requisições subsequentes. O *Spring Security* foi utilizado para configurar e gerenciar a autenticação e a autorização, garantindo que apenas usuários autenticados possam acessar determinados recursos.

```
@RequiredArgsConstructor
public class SecurityConfig {
    private final JWTAuthFilter jwtAuthFilter;
    private final AuthenticationProvider authenticationProvider;

    @Bean  no usages
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {
        httpSecurity
                .csrf(AbstractHttpConfigurer::disable)
                .cors(withDefaults())
                .authorizeHttpRequests( AuthorizationManagerRequestMat... auth -> auth
                        .requestMatchers( ...patterns: "/auth/**") AuthorizedUrl
                        .permitAll() AuthorizationManagerRequestMat...
                        .requestMatchers( ...patterns: "/videos/**") AuthorizedUrl
                        .permitAll() AuthorizationManagerRequestMat...
                        .requestMatchers( ...patterns: "/users/**") AuthorizedUrl
                        .permitAll()
                )
                .sessionManagement( SessionManagementConfigurer<HttpSecurity> session ->
                        session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
                .authenticationProvider(authenticationProvider)
                .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);

        return httpSecurity.build();
    }
```

Figura 9. Filtro de segurança usando *token JWT*.

A *API* oferece operações *CRUD* para gerenciar recursos no banco de dados. O *Spring Data JPA* foi utilizado para simplificar o acesso ao banco de dados *PostgreSQL*, abstraindo a camada de persistência e tornando a criação de tabelas e relacionamentos entre tabelas mais simples; cada entidade do sistema é mapeada para uma tabela e cada atributo de uma entidade é mapeada para uma linha na tabela do banco de dados.

As rotas do *Illiad* (Figura 10 foram definidas a intuito de seguir os padrões *RESTful*. Para garantir a qualidade e a confiabilidade da *API*, foram realizados testes na ferramenta *Insomnia* para validar a lógica de negócios e a integração com o banco de dados.

A utilização de uma *API RESTful* com autenticação *JWT* e *Spring Data JPA* permitiu que o *Illiad* oferecesse uma solução segura, escalável e de alto desempenho, alinhada com as necessidades dos usuários e as melhores práticas de desenvolvimento de software.

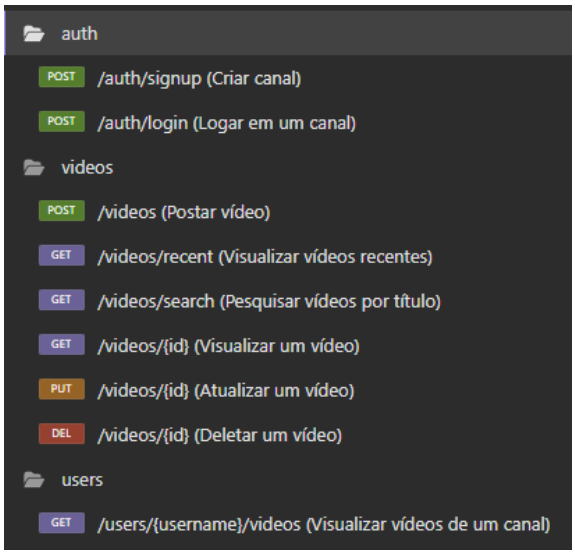

Figura 10. Rotas da *API*.

5.7. Aplicação web

Para o desenvolvimento *front-end* web do *Illiad* foi utilizado *HTML, CSS* e *Javascript*. A utilização de *AJAX* (*Asynchronous JavaScript and XML*) foi essencial para melhorar a experiência do usuário, permitindo que as páginas fossem atualizadas dinamicamente sem a necessidade de recarregar toda página ao fazer uma requisição na *API*.

```javascript
getRecentVideos();
function getRecentVideos() {
    const videosContainer = document.getElementById("videos");
    fetch("http://localhost:8080/videos/recent")
        .then(response => {
            if (!response.ok) {
                throw new Error("Erro ao buscar vídeos recentes.");
            }
            return response.json();
        })
        .then(data => {
            const videos = data.content;

            videos.forEach((video) => {
                const videoElement = `
                    <a href="video.html?id=${video.id}">
                        <div class="video">
                            <div class="thumbnail">
                                <img src="" alt="" srcset="">
                            </div>
                            <div class="info">
                                <img src="" alt="" srcset="" class="icon">
                                <div class="details">
                                    <p>${video.title}</p>
                                    <p>@${video.channel}</p>
                                </div>
                            </div>
                        </div>
                    </a>
                `;
                videosContainer.insertAdjacentHTML("beforeend", videoElement);
            });
        }).catch(error => console.error("Erro ao carregar vídeos recentes:", error));
}
```

Figura 11. Requisição na *API* para o *endpoint /videos/recent*, que retorna os vídeos recentemente postados.

5.7.1. Visualização de vídeos

A página inicial do sistema exibe os vídeos mais recentes postados pelos usuários. Cada vídeo é apresentado com uma miniatura (*thumbnail*), o título do vídeo, a foto de perfil do canal e o nome do canal que postou. Essa organização permite que os usuários naveguem facilmente pelo conteúdo disponível e escolham o que desejam assistir.

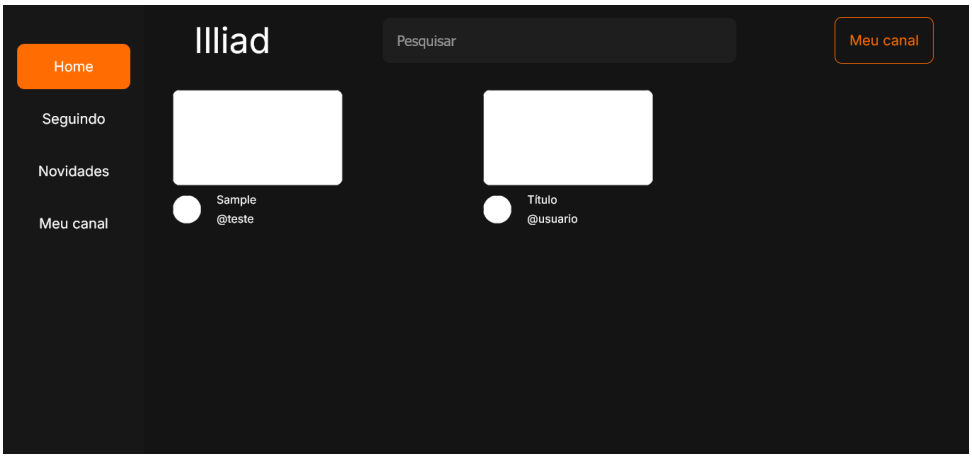

Figura 12. Tela de vídeos recentes (*home*).

Na tela de visualização de um vídeo, o usuário encontra o *player* de vídeo, o título, a descrição, a data de postagem, visualizações, um botão para compartilhar o vídeo e o

link do canal que postou o conteúdo. Além disso, na lateral da tela, há um modelo de carrossel para vídeos recomendados, funcionalidade não implementada no sistema.

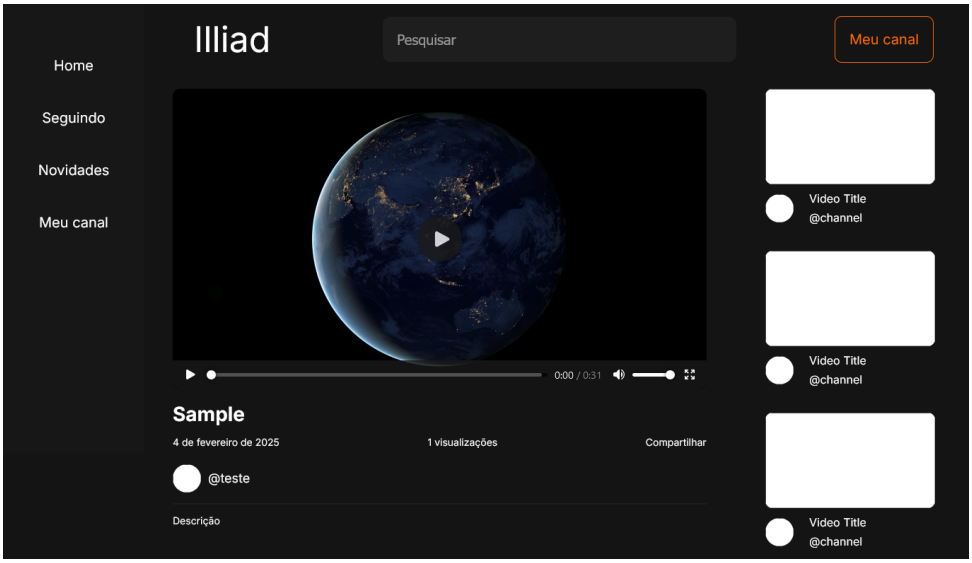

Figura 13. Tela de visualização de um vídeo.

5.7.2. Login e cadastro

A tela de login e cadastro foi consolidada em uma interface para facilitar o processo de autenticação para o usuário. Para criar uma conta o usuário fornece o nome de usuário (que será utilizada como nome do canal), *email* e senha. Para realizar login, o usuário informa apenas seu *email* e senha.

Há validações *Javascript* para assegurar que o usuário preencheu todos os campos, assim como validações no *back-end* para garantir que o sistema não permita o cadastro de usuários com informações inválidas.

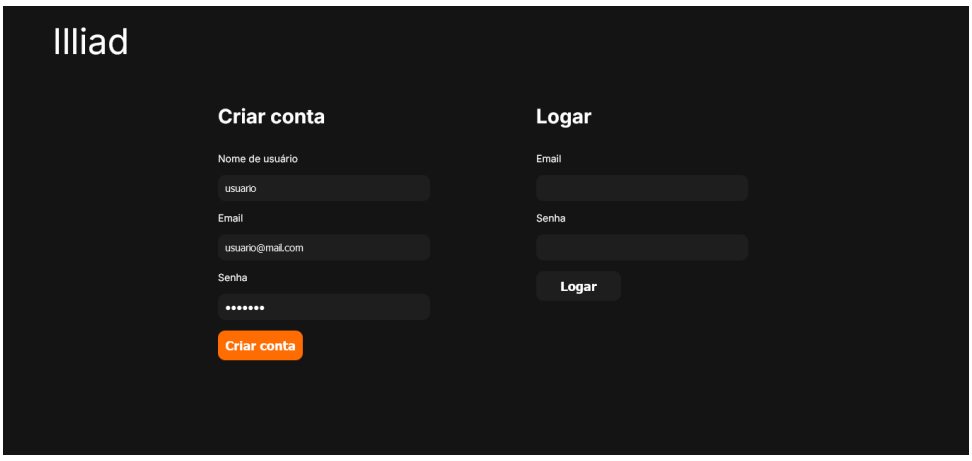

Figura 14. Tela de autenticação.

5.7.3. Canal

Na tela de canal o usuário é apresentado todos os vídeos recentes do canal acessado, organizados por data de postagem. Além dos vídeos são exibidas informações essenciais do canal, nome, foto de perfil e banner, a fim de proporcionar uma melhor experiência visual. A ordenação dos vídeos por data de postagem garante que o usuário sempre tenha acesso ao conteúdo mais recente do canal.

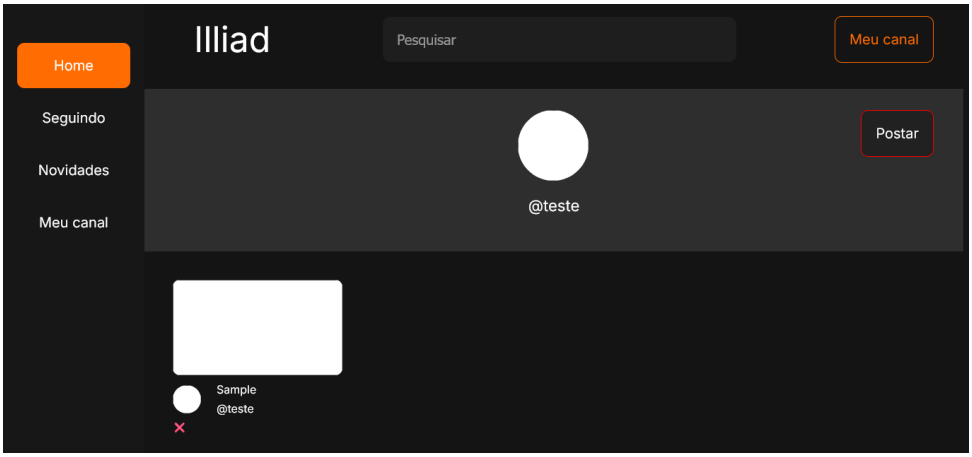

Figura 15. Tela de canal.

Ao entrar no seu próprio canal, o usuário se depara com a opção de "Postar", permitindo que ele compartilhe novos vídeos com seus seguidores. Ao clicar nessa opção, ele é redirecionado para a tela de postagem de vídeos, onde pode escolher o título, a descrição e anexar o vídeo que deseja compartilhar. O usuário também tem a opção de deletar os vídeos ao acessar seu canal, em baixo de cada um de seus vídeos há um botão X vermelho, ao clicar o usuário recebe uma mensagem de confirmação antes da deleção do conteúdo.

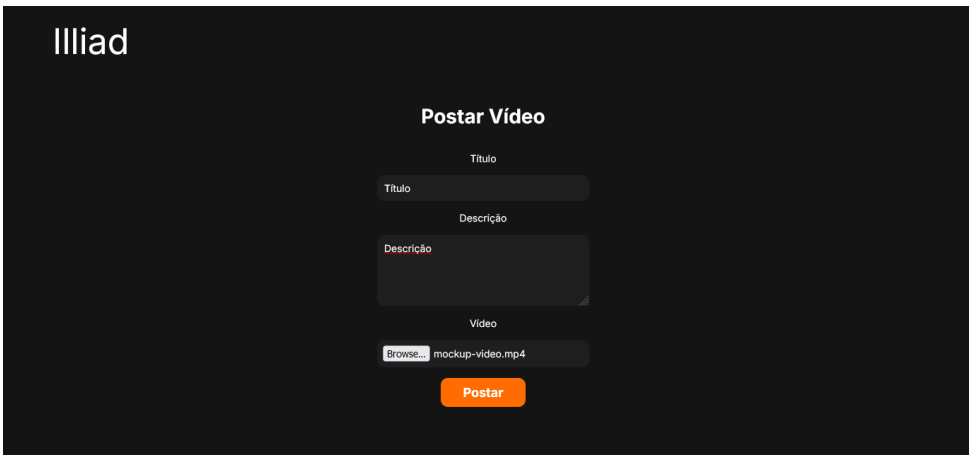

Figura 16. Postar vídeo.

O sistema válida as informações (como formato de arquivo inválida ou campos não preenchidos) e exibe uma mensagem de erro caso necessário. Caso a postagem seja

bem-sucedida, o usuário recebe uma confirmação visual e é redirecionado para a página do seu canal, onde o novo vídeo será exibido na lista.

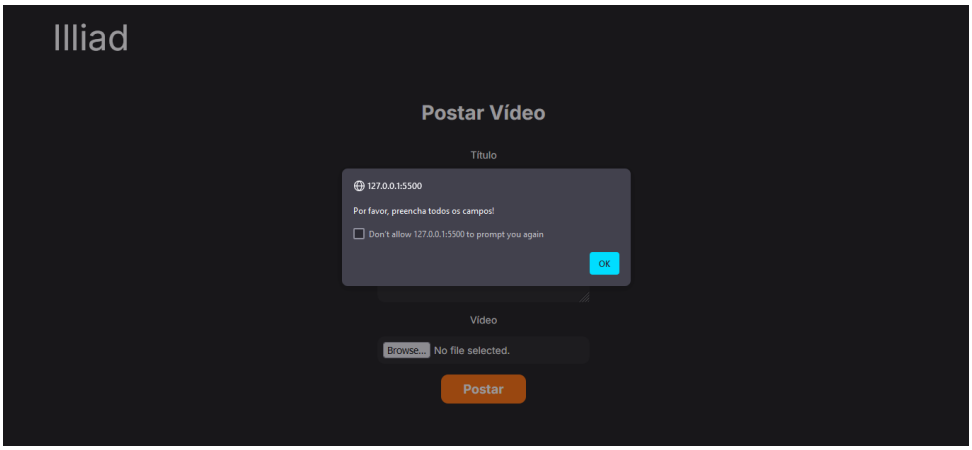

Figura 17. Mensagem de erro.

5.7.4. Testes funcionais

Os testes funcionais têm como objetivo validar se o sistema atende aos requisitos especificados. Eles verificam o comportamento da aplicação em relação às funcionalidades esperadas, garantindo que cada recurso opere conforme planejado. Segundo Sommerville (2011), testes funcionais são essenciais para assegurar que um sistema cumpra seus requisitos de maneira eficiente e sem falhas críticas.

Testes funcionais manuais envolvem a comparação dos requisitos das histórias de usuário com o comportamento da aplicação. O objetivo é verificar se as funcionalidades implementadas atendem aos critérios de aceitação de cada história de usuário. Na história de usuário "Visualizar vídeo", por exemplo, foi testado manualmente se as funcionalidades de reprodução do vídeo, com as opções de *play*, *pause*, volume e tela cheia, estavam funcionando corretamente e se as informações do vídeo eram exibidas conforme esperado.

5.7.5. Testes de API

Os testes da *API* do sistema foram realizados utilizando a ferramenta *Insomnia*, permitindo a validação das requisições HTTP da *API*. Por meio dessa abordagem, foi possível testar as rotas do sistema, avaliando o funcionamento das operações de requisição e resposta.

Os cenários testados foram cadastro e login de usuários, visualização, busca, postagem e deleção de vídeos e navegação entre canais. Através desses testes foi possível identificar e corrigir inconsistências nas respostas da *API* antes que as funcionalidades fossem integradas no *front-end*, assegurando que as funcionalidades principais e a autenticação por token *JWT* operassem conforme esperado.

5.7.6. Código fonte

O código fonte do *front-end* e *back-end* do projeto e seus artefatos estão disponíveis no seguinte link no *GitHub*: https://github.com/duarte50/illiad sob a licença *GNU General Public License 3* (*GPLv3.0*).

6. Conclusão

O desenvolvimento do *Illiad*, uma plataforma web de compartilhamento de vídeos de código aberto, representa uma alternativa às plataformas proprietárias disponíveis no mercado, tratando questões como privacidade e controle de dados do usuário. Ao adotar uma arquitetura cliente-servidor centralizada e um modelo de software livre, o projeto garante transparência e liberdade aos usuários em relação às suas informações e facilita o acesso a conteúdo na plataforma. O uso de tecnologias robustas e modernas – como o *framework Spring* e banco de dados *PostgreSQL* – garante confiabilidade e desempenho do sistema.

O *Illiad*, com interface simples e intuitiva, visa proporcionar uma melhor experiência de usuário ao mesmo tempo em que contribui para uma internet livre. Embora o *Illiad* represente uma alternativa viável para compartilhamento de vídeos de código aberto, há limitações, como a necessidade de infraestrutura para hospedagem e escalabilidade. Trabalhos futuros podem explorar a descentralização da plataforma, um algoritmo de vídeos recomendados, a implementação de técnicas avançadas de compressão de vídeo, imagens de perfil e miniaturas para otimizar o desempenho, implementação de um sistema de *feedback* de usuário com comentários e gostei/não gostei e maior customização para o espectador e criador de conteúdo.

As disciplinas do curso superior de Tecnologia em Análise e Desenvolvimento de Sistemas aplicadas neste projeto incluem: Projeto de Sistemas I e II, disciplinas fundamentais para o desenvolvimento deste artigo; Programação Orientada a Objetos, que contribuiu para a implementação do sistema utilizando Java Orientado a Objetos; Arquitetura de Software, que auxiliou na documentação e elaboração dos artefatos do sistema; Desenvolvimento Web e Desenvolvimento de Sistemas Web, que consolidaram uma base sólida em desenvolvimento web *front-end* e *back-end*; e Banco de Dados I e II, que contribuíram no uso de bancos de dados relacionais *SQL*.

Referências

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Free Software Foundation (2024). Free software is a matter of liberty, not price. https://www.fsf.org/about. Acesso em 30 de outubro de 2024.

Humphries, M. (2021). Audacity is Being Called Spyware After Privacy Policy Update. https://www.pcmag.com/news/audacity-is-being-called-spyware-after-privacy-policy-update. Acesso em 28 de novembro de 2024.

Johnson, R. (2004). *Expert One-on-One J2EE Development without EJB*. Wrox.

Massé, M. (2011). *REST API Design Rulebook*. O'Reilly Media.

Odysee (2024). What is Odysee?. https://help.odysee.tv/category-basics/whatisodysee/. Acesso em 10 de julho de 2024.

Osterwalder, A., Pigneur, Y., Bernarda, G., e Smith, A. (2014). *Value Proposition Design: How to Create Products and Services Customers Want*. Wiley.

Pew Research (2022). PEW RESEARCH CENTER'S AMERICAN TRENDS PANEL WAVE 112 SOCIAL MEDIA AND PLATFORM SURVEY FINAL TOPLINE. https://www.pewresearch.org/journalism/wp-content/uploads/sites/8/2022/09/Social-media-and-platform-fact-sheet-TOPLINE.pdf. Acesso em 7 de março de 2024.

PostgreSQL (2024). PostgreSQL 17.2 Documentation. https://www.postgresql.org/files/documentation/pdf/17/postgresql-17-US.pdf. Acesso em 28 de novembro de 2024.

Similarweb (2024). Top Websites Ranking. Most Visited Streaming & Online TV Websites. https://www.similarweb.com/top-websites/art s-and-entertainment/tv-movies-and-streaming/. Acesso em 7 de março de 2024.

Solon, O. (2018). Facebook says cambridge analytica may have gained 37m more users' data. *The Guardian*. Acesso em 28 de novembro de 2024.

Sommerville, I. (2011). *Software Engineering*. Addison-Wesley, Boston, MA, 9 edition.

Souza, M. (2024). Introdução ao Spring Framework 3. https://www.devmedia.com.br/introducao-ao-spring-framework-3/23148. Acesso em 10 de dezembro de 2024.

Stallman, R. (2006). Opposing digital rights mismanagement. *Free Software Foundation*. Acesso em 24 de fevereiro de 2024.

The Open Source Way (2019). THE OPEN SOURCE WAY 2.0. https://www.theopensourceway.org/the_open_source_way-guidebook-2.0.html. Acesso em 24 de fevereiro de 2025.

Xu, K., Li, T., Wang, H., Li, H., Wei, Z., Liu, J., e Lin, S. (2018). Modeling, analysis, and implementation of universal acceleration platform across online video sharing sites. *IEEE Transactions on Services Computing 2018-may 01 vol. 11 iss. 3*, 11.

ANEXO I: Histórias de usuário

1. Visualizar vídeo

“Como usuário, eu quero assistir um vídeo na plataforma, para que eu possa consumir conteúdo de forma fácil e intuitiva”.

Prioridade máxima

Critérios de aceitação:

- O vídeo deve ser reproduzido em um *player* com opções de tocar e pausar, volume e tela cheia;
- O sistema deve exibir informações sobre o vídeo, como título, descrição, data de postagem, número de visualizações e nome do canal;
- O contador de visualizações deve ser atualizado automaticamente sempre que o vídeo for assistido.

2. Postar vídeo

“Como usuário, eu quero postar um vídeo no meu canal, para que eu possa compartilhar conteúdo com outros usuários”

Prioridade máxima

Critérios de aceitação:

- O usuário deve estar autenticado no sistema;
- O sistema deve permitir que o usuário insira um título, uma descrição e selecione um arquivo de vídeo para *upload*;
- O vídeo postado deve ser processado e armazenado no sistema, garantindo que esteja disponível para visualização por outros usuários;
- O usuário deve receber uma confirmação visual de que o vídeo foi postado com sucesso.

3. Gerenciar canal

“Como criador de conteúdo, eu quero poder alterar e deletar dados do meu canal para que eu possa manter as informações atualizadas e ter controle total sobre o meu canal."

Prioridade máxima

Critérios de aceitação:

- O usuário deve estar logado em uma conta existente;
- O sistema deve impedir o usuário de editar dados pessoais como nome do canal ou data de nascimento repetidas vezes em um curto período de tempo;
- O usuário terá de autenticar-se novamente se ele desejar deletar sua conta.

4. Seguir um canal

“Como usuário, eu quero poder inscrever-me em outros canais, para que eu possa acompanhar o conteúdo de outros usuários”

Prioridade mínima

Critérios de aceitação:

- O usuário deve estar logado em uma conta existente;
- O usuário deve receber notificações se o canal postar um vídeo novo.

5. Visualizar estatísticas

“Como criador de conteúdo, eu quero poder visualizar as estatísticas dos meus vídeo em meu canal, para que eu possa analisar o desempenho do meu conteúdo e melhorar minhas postagens ”

Prioridade mínima

Critérios de aceitação:

- O usuário deve estar logado em uma conta existente;
- O usuário deve ter postado vídeos em seu canal.

Documento Digitalizado Público

Artigo - Versão Final do TCC

Assunto: Artigo - Versão Final do TCC
Assinado por: Fernando Sambinelli
Tipo do Documento: Formulário
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Fernando Sambinelli**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/03/2025 18:52:03.

Este documento foi armazenado no SUAP em 10/03/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1961079
Código de Autenticação: a332558a96