INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS PICOS
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RAFAEL PEREIRA BORGES

DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DE CATÁLOGOS PESSOAIS DE JOGOS DIGITAIS

PICOS, PIAUÍ
2026

RAFAEL PEREIRA BORGES

DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DE CATÁLOGOS PESSOAIS DE JOGOS DIGITAIS

Trabalho de Conclusão de Curso (Relatório Técnico de Software) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Orientador: Prof. M^{e}. Anthony Irlan Marques Luz

PICOS, PIAUÍ
2026

RAFAEL PEREIRA BORGES

DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DE CATÁLOGOS PESSOAIS DE JOGOS DIGITAIS

Trabalho de Conclusão de Curso (Relatório Técnico de Software) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Aprovada em 12 de agosto de 2026.

BANCA EXAMINADORA:

Prof. M^{e}. Anthony Irlan Marques Luz(Orientador)
Instituto Federal do Piauí (IFPI)

Prof. Dr. Rogério Figueredo de Sousa
Instituto Federal do Piauí (IFPI)

Prof. M^{e}. João Paulo Nascimento
Instituto Federal do Piauí (IFPI)

PICOS, PIAUÍ
2026

Dedico este trabalho a todos que estiveram ao meu lado ao longo desta jornada, oferecendo apoio, incentivo e confiança, tornando possível a concretização desta conquista.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus pela oportunidade de chegar até este momento. Aos professores que fizeram parte da minha trajetória acadêmica, expresso minha sincera gratidão pelos ensinamentos e por toda a contribuição para minha formação. Aos meus amigos, agradeço pelo apoio, incentivo e companheirismo ao longo dessa jornada. Por fim, agradeço à minha família, pelo amor, compreensão e apoio incondicional, estando sempre ao meu lado em todos os momentos.

RESUMO

O crescimento do mercado de jogos digitais e a ampla disponibilidade de títulos tornam a organização de bibliotecas pessoais e a descoberta de novos jogos tarefas cada vez mais desafiadoras para os usuários. Nesse contexto, este trabalho apresenta o desenvolvimento de um sistema *web* para gerenciamento de catálogos pessoais de jogos digitais, capaz de integrar funcionalidades de organização da biblioteca com um mecanismo de recomendação baseado em modelos de linguagem de grande porte. A solução foi desenvolvida utilizando Java, Spring Boot, Hibernate, Flyway e PostgreSQL, além da integração com a API RAWG para obtenção de informações dos jogos e com a plataforma Groq para geração das recomendações. Inicialmente, foram levantados os requisitos do sistema e elaborados os modelos que orientaram sua implementação. Em seguida, foi desenvolvida uma API REST contemplando funcionalidades para autenticação de usuários, gerenciamento de perfis, busca e organização da biblioteca de jogos e geração de recomendações personalizadas a partir de um título ou de uma descrição em linguagem natural. A validação da aplicação foi realizada por meio de testes funcionais dos *endpoints* da API e da avaliação da qualidade das recomendações utilizando a abordagem LLM as a Judge, baseada em critérios relacionados à similaridade da experiência, *gameplay*, ambientação, justificativas e qualidade geral das recomendações. Os resultados obtidos demonstraram que a solução desenvolvida atendeu aos objetivos propostos, apresentando desempenho satisfatório tanto na implementação das funcionalidades previstas quanto na geração de recomendações coerentes com as preferências informadas pelos usuários. Dessa forma, o trabalho evidencia a viabilidade da utilização de modelos de linguagem como apoio a sistemas de recomendação de jogos, contribuindo para a organização de bibliotecas digitais e para a descoberta de novos títulos.

Palavras-chaves: Jogos digitais; Sistemas de recomendação; API REST; Aplicações web.

ABSTRACT

The growth of the digital gaming market and the widespread availability of titles make organizing personal libraries and discovering new games increasingly challenging tasks for users. In this context, this work presents the development of a web system for managing personal catalogs of digital games, capable of integrating library organization functionalities with a recommendation engine based on large-scale language models. The solution was developed using Java, Spring Boot, Hibernate, Flyway and PostgreSQL, in addition to integration with the RAWG API to obtain game information and with the Groq platform to generate recommendations. Initially, the system requirements were raised and the models that guided its implementation were developed. Next, a REST API was developed with features for user authentication, profile management, game library search and organization, and the generation of personalized recommendations from a title or natural language description. The application was validated through functional testing of the API endpoints and recommendation quality assessment using the LLM as a Judge approach based on criteria related to similarity of experience, gameplay, setting, justifications and overall quality of recommendations. The results obtained demonstrate that the developed solution met the proposed objectives, presenting satisfactory performance both in implementing the planned functionalities and in generating recommendations consistent with the preferences informed by users. Thus, the work highlights the feasibility of using language models to support game recommendation systems, contributing to the organization of digital libraries and the discovery of new titles.

Keywords: Digital games; Recommender systems; REST API; Web applications.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicity, Consistency, Isolation, Durability (Atomicidade, Consistência, Isolamento e Durabilidade)
API	Application Programming Interface (Interface de Programação de Aplicações)
HTTP	Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)
JSON	JavaScript Object Notation (Notação de Objetos JavaScript)
JVM	Java Virtual Machine (Máquina Virtual Java)
JWT	JSON Web Token (Token Web em JSON)
LLM	Large Language Model (Modelo de Linguagem de Grande Porte)
ORM	Object-Relational Mapping (Mapeamento Objeto-Relacional)
PC	Personal Computer (Computador Pessoal)
REST	Representational State Transfer (Transferência de Estado Representacional)
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
DTO	Data Transfer Object (Objeto de Transferência de Dados)

LISTA DE SÍMBOLOS

%	Porcentagem

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o mercado de jogos digitais passou por uma expansão significativa, impulsionada pela popularização do acesso à internet e pela evolução das tecnologias computacionais (Newzoo, 2025). Embora esse cenário amplie as possibilidades de entretenimento, a elevada quantidade de opções pode dificultar a tomada de decisão dos usuários, em razão de um conceito conhecido como sobrecarga cognitiva provocada pelo excesso de alternativas (Iyengar; Lepper, 2000).

Embora a disponibilidade de jogos digitais tenha aumentado significativamente, muitos usuários ainda enfrentam dificuldades para organizar suas bibliotecas e encontrar títulos alinhados às suas preferências, especialmente em plataformas com catálogos extensos (Ricci; Rokach; Shapira, 2015). Esse excesso de informação pode comprometer a experiência do usuário, reduzindo o engajamento e dificultando a exploração de novos conteúdos digitais (Schwartz, 2004; Bawden; Robinson, 2009).

Nesse contexto, sistemas de recomendação têm sido amplamente empregados em plataformas digitais para auxiliar os usuários na descoberta de conteúdos alinhados às suas preferências, reduzindo as dificuldades associadas à grande quantidade de opções disponíveis (Adomavicius; Tuzhilin, 2005). Mais recentemente, trabalhos promissores têm investigado a aplicação de modelos de linguagem de grande porte em sistemas de recomendação, explorando capacidades como compreensão de linguagem natural, interpretação de preferências e geração de recomendações contextualizadas (Harte *et al.*, 2023; Wang *et al.*, 2024).

Nesse cenário, o desenvolvimento de aplicações *web* oferece grandes vantagens como acesso a multiplataforma, centralização de dados e integração com serviços externos, sendo amplamente adotadas para sistemas de gerenciamento de informações (Pressman; Maxim, 2016). Sistemas de bibliotecas digitais permitem o armazenamento, organização e recuperação eficiente de conteúdos digitais, promovendo melhor gerenciamento da informação (Borgman, 1999).

Diante disso, este trabalho apresenta o relatório técnico do desenvolvimento de um sistema voltado para o gerenciamento de catálogos pessoais de jogos digitais que incorpora uma camada de recomendação com auxílio de inteligência artificial. O foco deste estudo consiste na implementação e documentação da arquitetura *backend*, detalhando a prática da engenharia de *software* aplicada.

Este trabalho está organizado em quatro seções principais, estruturadas de forma a apresentar, de maneira gradual, o desenvolvimento do projeto. Inicialmente, são apresentados os Objetivos da pesquisa, que guiam o estudo. Em seguida, a seção Tecnologias Envolvidas descreve as principais ferramentas, *frameworks* e serviços utilizados no desenvolvimento da aplicação, justificando sua adoção. A seção Mode-

lagem do Projeto apresenta os requisitos do sistema, os diagramas e a arquitetura da solução, detalhando o planejamento realizado antes da implementação. Na seção Software, são descritos o processo de implantação da aplicação e os procedimentos de testes empregados para validar tanto o funcionamento do sistema quanto a qualidade da funcionalidade de recomendação baseada em inteligência artificial. Por fim, a seção Considerações Finais reúne uma análise dos resultados obtidos, as principais contribuições do trabalho, suas limitações e possíveis direções para trabalhos futuros.

1.1 OBJETIVOS

A presente seção detalha os objetivos que guiam este trabalho, os quais são subdivididos em um objetivo geral e quatro objetivos específicos, a fim de estabelecer o escopo e as metas do estudo.

1.1.1 Geral

Desenvolver e avaliar um sistema *web* para gerenciamento de coleções de jogos eletrônicos, capaz de fornecer recomendações contextualizadas.

1.1.2 Específicos

1. Levantar e especificar os requisitos funcionais e não funcionais do sistema a partir da análise do problema, da revisão da literatura e da investigação de funcionalidades presentes em plataformas similares.
2. Modelar o banco de dados, arquitetura e os demais artefatos necessários ao desenvolvimento do sistema.
3. Desenvolver o protótipo da aplicação, implementando as funcionalidades propostas para gerenciamento da biblioteca de jogos e recomendações.
4. Avaliar o sistema por meio de testes de *software* e da análise de qualidade das recomendações geradas.

2 TECNOLOGIAS ENVOLVIDAS

Esta seção apresenta as tecnologias utilizadas no desenvolvimento deste projeto. A escolha dessas ferramentas foi realizada considerando o domínio sobre as tecnologias adotadas, bem como o escopo e o prazo de desenvolvimento estabelecidos para o projeto.

2.1 TECNOLOGIAS DO BACKEND

O *backend* do sistema foi desenvolvido utilizando tecnologias consolidadas no desenvolvimento de aplicações corporativas, priorizando desempenho, manutenibilidade, escalabilidade e facilidade de integração.

2.1.1 Java

A linguagem Java(Oracle Corporation, 2026) foi escolhida por sua maturidade, portabilidade e ampla adoção no desenvolvimento de aplicações corporativas. Segundo a *Stack Overflow Developer Survey 2024*, Java permanece entre as linguagens de programação mais utilizadas por desenvolvedores profissionais, refletindo sua relevância no desenvolvimento de sistemas de médio e grande porte (Stack Overflow, 2024). Além disso, sua orientação a objetos favorece a organização do código e a reutilização de componentes, enquanto seu ecossistema disponibiliza diversas bibliotecas e *frameworks* que simplificam o desenvolvimento de aplicações *web* (Deitel; Deitel, 2017).

No contexto deste trabalho, Java constitui a base para implementação da lógica de negócio da aplicação.

2.1.2 Spring Boot

O Spring Boot(VMware, Inc., 2026) é um *framework* amplamente utilizado para o desenvolvimento de aplicações Java, fornecendo recursos como configuração automática, gerenciamento de dependências e suporte nativo à construção de aplicações *web* e *APIs REST* (Walls, 2022). De acordo com a pesquisa *State of Developer Ecosystem 2024*, realizada pela JetBrains, o Spring Boot permanece como o *framework* mais utilizado pelos desenvolvedores Java para aplicações *backend*, evidenciando sua consolidação no ecossistema da linguagem (JetBrains, 2024).

Sua utilização permitiu reduzir a quantidade de configurações necessárias durante o desenvolvimento, além de disponibilizar módulos para autenticação, injeção de dependências, acesso a dados e tratamento de requisições HTTP.

2.1.3 Hibernate

O Hibernate(Red Hat, Inc., 2026) foi utilizado como *framework* de Mapeamento Objeto-Relacional (ORM), responsável por realizar a conversão entre objetos Java e tabelas do banco de dados. Essa abordagem reduz a necessidade de escrita manual de comandos SQL para operações de persistência, simplificando o acesso aos dados (Bauer; King; Gregory, 2015).

No contexto da aplicação, o Hibernate é utilizado para gerenciar as entidades da aplicação, seus relacionamentos e operações de criação, consulta, atualização e remoção de dados.

2.1.4 Flyway

O Flyway(Redgate Software, 2026) é uma ferramenta para controle de versões do banco de dados baseada em migrações. Seu principal objetivo é garantir que a estrutura do banco permaneça sincronizada entre os diferentes ambientes de desenvolvimento e implantação, registrando todas as alterações realizadas no esquema do banco de dados.

Durante o desenvolvimento do sistema, o Flyway foi utilizado para automatizar a criação e evolução da estrutura do banco de dados, permitindo maior controle das modificações realizadas e reduzindo inconsistências entre diferentes versões da aplicação.

2.1.5 PostgreSQL

O PostgreSQL(PostgreSQL Global Development Group, 2026) foi escolhido como sistema gerenciador de banco de dados devido à sua confiabilidade, conformidade com o padrão SQL e suporte a recursos avançados, como integridade referencial, transações ACID e extensibilidade. Além dessas características, o PostgreSQL figura entre os sistemas gerenciadores de banco de dados mais utilizados mundialmente, ocupando posições de destaque no *DB-Engines Ranking*, o que demonstra sua ampla adoção em aplicações acadêmicas e corporativas (DB-Engines, 2025).

Sua utilização permitiu armazenar de forma estruturada as informações da aplicação, incluindo dados dos usuários, bibliotecas de jogos e demais entidades do sistema.

2.2 INTEGRAÇÃO COM SERVIÇOS EXTERNOS

Esta seção detalha a estratégia de integração com serviços externos, abordando a utilização de APIs para ampliar as funcionalidades e garantir a comunicação da aplicação com plataformas de terceiros.

2.2.1 Integração com Base de Dados de Jogos

Para obtenção das informações dos jogos, foi utilizada a API pública RAWG (RAWG, 2026), uma plataforma que disponibiliza um extenso catálogo de jogos digitais contendo dados como nome, descrição, plataformas, gêneros, desenvolvedoras, publicadoras, imagens e avaliações.

A escolha da RAWG deve-se à abrangência de sua base de dados, à disponibilidade de uma API REST bem documentada e à variedade de informações disponibilizadas para cada título em uma única requisiçãos. Além disso, o plano gratuito da plataforma permite a realização de até 20.000 requisições mensais, quantidade considerada suficiente para as necessidades de desenvolvimento e validação do sistema durante sua fase inicial. Essa característica possibilitou a integração de uma base de dados abrangente sem custos adicionais, tornando a solução adequada ao contexto acadêmico do projeto.

2.2.2 Integração com Inteligência Artificial

Para a implementação da funcionalidade de recomendação de jogos, foi utilizada a plataforma Groq (Groq, Inc., 2026), um serviço de inferência para modelos de linguagem de grande porte (LLMs) que disponibiliza uma API compatível com o padrão OpenAI. A plataforma oferece baixa latência e suporte à execução de diferentes modelos de inteligência artificial, facilitando sua integração com aplicações *web*.

A escolha da Groq deve-se ao seu desempenho na execução de modelos de linguagem, à simplicidade de integração por meio de API REST e à disponibilidade de modelos de código aberto. Além disso, a plataforma disponibiliza um plano gratuito com uma quantidade significativa de *tokens* para utilização diária, característica que atende às necessidades de desenvolvimento e validação do sistema sem custos adicionais. Esses fatores permitiram incorporar funcionalidades de inteligência artificial à aplicação sem a necessidade de infraestrutura própria para hospedagem e execução dos modelos.

2.3 INFRAESTRUTURA DE IMPLANTAÇÃO

Esta seção detalha as ferramentas de infraestrutura e implantação utilizadas para colocar o projeto em funcionamento, com foco nas ferramentas escolhidas para hospedar a aplicação e o banco de dados.

2.3.1 Render

O Render (Render, 2026) foi utilizado como plataforma de hospedagem da API REST desenvolvida neste trabalho. Trata-se de um serviço de computação em

nuvem que permite a implantação de aplicações diretamente a partir de repositórios Git, automatizando etapas como compilação, inicialização da aplicação e disponibilização do serviço na internet.

A escolha do Render deve-se à facilidade de integração com projetos Spring Boot, à disponibilidade de um plano gratuito adequado ao contexto acadêmico e ao processo simplificado de implantação contínua. Essas características permitiram disponibilizar a API em um ambiente acessível externamente, facilitando a realização de testes e a validação das funcionalidades implementadas.

2.3.2 Neon

O Neon (Neon, Inc., 2026) foi utilizado como serviço de hospedagem do banco de dados PostgreSQL da aplicação. A plataforma oferece um banco de dados PostgreSQL gerenciado em nuvem, eliminando a necessidade de configuração e manutenção de um servidor dedicado.

A escolha do Neon deve-se à disponibilidade de um plano gratuito, à compatibilidade nativa com o PostgreSQL e à facilidade de integração com aplicações hospedadas em nuvem. No sistema desenvolvido, o Neon foi utilizado para armazenar de forma persistente as informações da aplicação, permitindo que a API acessasse os dados em um ambiente de produção com segurança e confiabilidade.

3 MODELAGEM DO PROJETO

Esta seção detalha como o projeto foi planejado e modelado, incluindo a estrutura e os processos que foram implementados.

3.1 LEVANTAMENTO DE REQUISITOS

Esta seção detalha o processo de levantamento de requisitos, etapa fundamental para compreender as necessidades do projeto e definir as funcionalidades que o sistema deve abranger.

3.1.1 Requisitos funcionais

O levantamento dos requisitos funcionais teve como objetivo identificar e documentar as principais funcionalidades que o sistema deve oferecer para atender às necessidades dos usuários e aos objetivos propostos pelo projeto (Sommerville, 2011).

Tabela 1 – Requisitos Funcionais do Sistema

Referência	**Título**	**Descrição**
RF01	Cadastro de usuário	O sistema deve permitir o cadastro de novos usuários por meio do fornecimento das informações obrigatórias.
RF02	Autenticação de usuário	O sistema deve permitir que usuários autenticados realizem *login*, retornando um *token* JWT para acesso aos recursos protegidos da API.
RF03	Atualização de credenciais	O sistema deve permitir que o usuário altere suas credenciais de acesso, como nome, *e-mail* e senha.
RF04	Exclusão de conta	O sistema deve permitir que o usuário exclua permanentemente sua conta e os dados associados.
RF05	Criação de perfil	O sistema deve permitir que o usuário crie um perfil contendo informações complementares às fornecidas durante o cadastro.
RF06	Visualização de perfil	O sistema deve permitir a visualização das informações do perfil do usuário.
RF07	Atualização de perfil	O sistema deve permitir a atualização das informações presentes no perfil do usuário.

Referência	Título	Descrição
RF08	Busca de jogos	O sistema deve permitir a busca de jogos pelo nome, consultando inicialmente a base de dados local e, caso o jogo não seja encontrado, realizando a consulta em uma API externa.
RF09	Visualização de detalhes do jogo	O sistema deve permitir a visualização das informações detalhadas de um jogo, como descrição, plataformas, gêneros, desenvolvedora e publicadora.
RF10	Adição de jogos à biblioteca	O sistema deve permitir adicionar jogos à biblioteca do usuário, registrando informações como status, progresso, nota, avaliação textual e data de conclusão.
RF11	Listagem da biblioteca	O sistema deve permitir listar todos os jogos pertencentes à biblioteca do usuário.
RF12	Pesquisa na biblioteca	O sistema deve permitir pesquisar jogos existentes na biblioteca do usuário por meio do nome do título.
RF13	Filtragem da biblioteca	O sistema deve permitir filtrar os jogos da biblioteca utilizando critérios como *status*, plataforma, gênero ou avaliação.
RF14	Atualização de informações do jogo	O sistema deve permitir atualizar as informações associadas aos jogos presentes na biblioteca do usuário, como progresso, status, nota e avaliação.
RF15	Remoção de jogos da biblioteca	O sistema deve permitir remover jogos da biblioteca do usuário.
RF16	Recomendação por título	O sistema deve gerar recomendações de jogos a partir de um título informado pelo usuário, utilizando um modelo de linguagem integrado à aplicação.
RF17	Recomendação por descrição	O sistema deve gerar recomendações de jogos a partir de uma descrição ou preferência informada em linguagem natural pelo usuário.

Fonte: Elaborado pelo autor (2026).

3.1.2 Requisitos não Funcionais

Além das funcionalidades previstas, foram definidos requisitos não funcionais com o objetivo de estabelecer os critérios de qualidade e as restrições técnicas que orientam o desenvolvimento da aplicação (Sommerville, 2011).

Tabela 2 – Requisitos Não Funcionais do Sistema

Referência	**Título**	**Descrição**
RNF01	API REST	O sistema deve disponibilizar seus recursos por meio de uma API seguindo os princípios da arquitetura REST, utilizando o protocolo HTTP, retornando os dados no formato JSON e empregando códigos de status HTTP adequados ao resultado de cada requisição.
RNF02	Segurança	O acesso aos recursos protegidos da aplicação deve exigir um *token* JWT válido, devendo requisições sem autenticação ou com *token* inválido serem rejeitadas com código HTTP 401.
RNF03	Persistência de dados	O sistema deve armazenar as informações de forma persistente utilizando o PostgreSQL como sistema gerenciador de banco de dados.
RNF04	Integridade dos dados	O banco de dados deve impedir o armazenamento de dados que violem as restrições definidas no modelo, como campos obrigatórios não preenchidos, valores únicos duplicados e referências inexistentes entre entidades relacionadas.
RNF05	Migração do banco de dados	O versionamento e a criação da estrutura do banco de dados devem ser realizados utilizando o Flyway.
RNF06	Desempenho	As requisições que dependam exclusivamente da base de dados local devem apresentar tempo de resposta inferior a 2 segundos em condições normais de utilização.

Referência	Título	Descrição
RNF07	Integração externa	A aplicação deve realizar com sucesso a comunicação com a API RAWG para obtenção de dados de jogos e com a API do modelo de linguagem para geração de recomendações, tratando respostas inválidas ou indisponibilidade dos serviços externos sem interromper a execução da aplicação.
RNF08	Escalabilidade	A arquitetura da aplicação deve permitir a inclusão de novas funcionalidades e integrações sem alterações significativas na estrutura existente.
RNF09	Manutenibilidade	O código-fonte da aplicação deve ser organizado em componentes com responsabilidades separadas, incluindo *controllers*, *services*, *repositories*, entidades e objetos de transferência de dados (DTOs).
RNF10	Portabilidade	A aplicação deve poder ser executada em qualquer ambiente que disponibilize uma versão da JVM compatível com a versão do Java utilizada no desenvolvimento, sem necessidade de alterações no código-fonte.
RNF11	Disponibilidade	A API deve apresentar disponibilidade de 100% durante o período de monitoramento em ambiente de produção.
RNF12	Compatibilidade	A integração com o modelo de linguagem deve estar isolada em um componente específico da aplicação e utilizar uma API compatível com o padrão OpenAI, permitindo a substituição do provedor sem alterações nos *controllers* responsáveis pelas funcionalidades de recomendação.

Fonte: Elaborado pelo autor (2026).

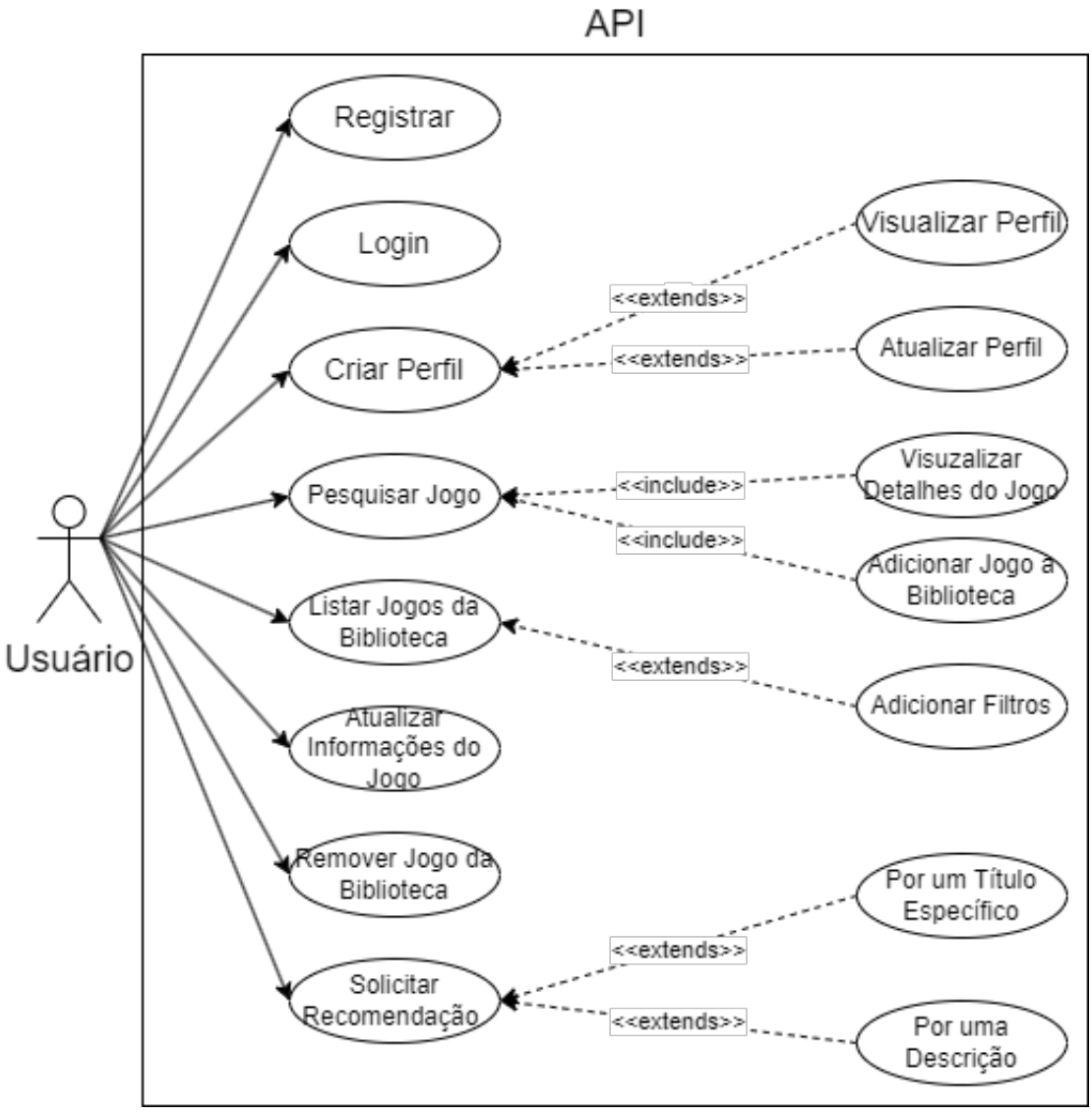
3.2 DIAGRAMA DE CASOS DE USO

O diagrama de casos de uso apresentado pela Figura 1 representa as principais funcionalidades disponibilizadas pela API e as interações realizadas pelo usuário com o sistema. Seu objetivo é fornecer uma visão geral dos serviços oferecidos, evidenciando

as operações que podem ser executadas e suas relações, sem detalhar aspectos de implementação.

As funcionalidades foram organizadas em quatro grupos principais. O primeiro contempla o gerenciamento da conta do usuário, incluindo cadastro, autenticação e alteração de credenciais. O segundo reúne as operações relacionadas ao perfil, permitindo sua criação, visualização e atualização. O terceiro grupo corresponde ao gerenciamento da biblioteca de jogos, abrangendo a pesquisa de títulos, visualização de informações, adição e remoção de jogos, atualização de dados como progresso e avaliação, além da listagem e filtragem da biblioteca. Por fim, o sistema disponibiliza uma funcionalidade de recomendação de jogos, que pode ser utilizada tanto a partir de um título específico quanto por meio de uma descrição fornecida pelo usuário.

Figura 1 – Diagrama de Casos de Uso



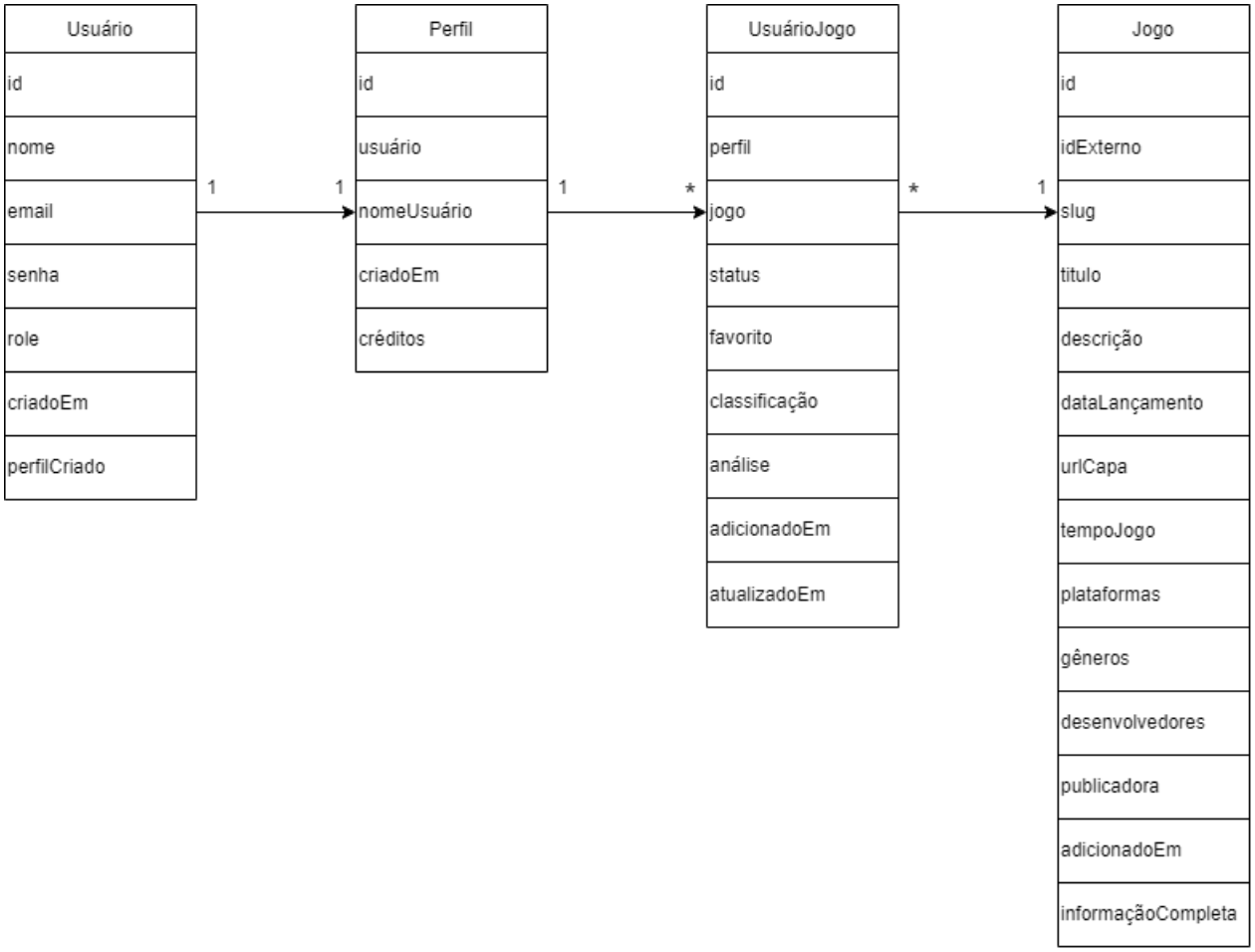
Fonte: Elaborado pelo autor (2026).

3.3 DIAGRAMA DE CLASSES

O diagrama de classes apresentado na Figura 2 representa a estrutura estática do sistema, evidenciando as principais entidades do domínio, seus atributos e os relacionamentos existentes entre elas. Esse modelo serviu como base para a implementação da camada de persistência da aplicação, permitindo organizar os dados de forma consistente e facilitar o desenvolvimento das funcionalidades previstas nos requisitos do sistema.

As classes foram definidas de acordo com as responsabilidades de cada componente da aplicação. A entidade Usuário representa as informações relacionadas à autenticação e ao gerenciamento das contas dos usuários, enquanto Perfil armazena os dados complementares do perfil. A entidade Jogo concentra as informações dos jogos obtidas por meio da integração com a API RAWG, e a entidade responsável pela biblioteca do usuário estabelece a associação entre perfis e jogos, armazenando informações específicas como status, avaliação, progresso e demais dados relacionados à experiência individual de cada usuário. Essa organização permitiu separar as responsabilidades de cada entidade, reduzir redundâncias e facilitar a manutenção e evolução da aplicação.

Figura 2 – Diagrama de Classes



Fonte: Elaborado pelo autor (2026).

3.4 ARQUITETURA DO SISTEMA

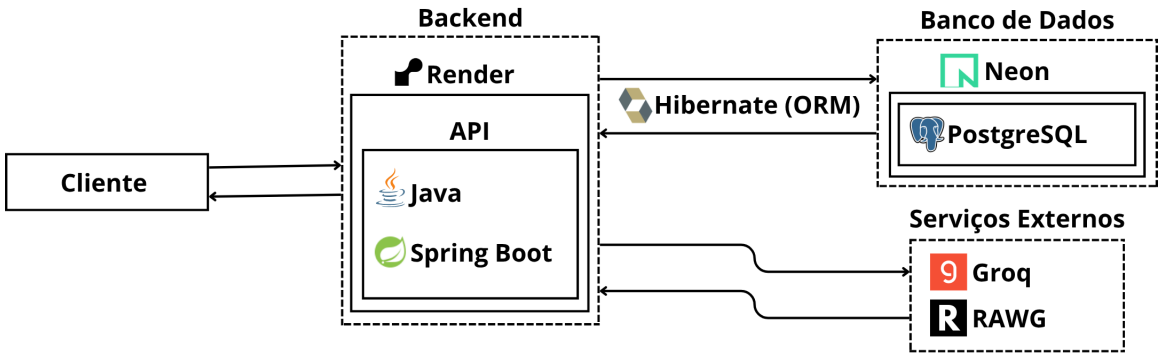
Esta seção apresenta a arquitetura do sistema, detalhando a interação entre os componentes e a forma como a solução está estruturada para atender aos requisitos propostos.

3.4.1 Fluxo do Sistema

O fluxo de processamento do sistema inicia-se quando um cliente realiza uma requisição HTTP à API. Após receber a solicitação, o Spring Boot direciona a requisição ao componente responsável pela lógica de negócio, que valida os dados recebidos, aplica as regras da aplicação e determina quais recursos serão utilizados para atender à operação solicitada. Quando a requisição envolve informações persistidas, o acesso ao banco de dados é realizado por meio do Hibernate, responsável pelo mapeamento objeto-relacional entre as entidades Java e o PostgreSQL, simplificando as operações de consulta e persistência.

Para operações que dependem de informações externas, a aplicação adota uma estratégia de priorização da base de dados local. Inicialmente, a API verifica se os dados do jogo já estão armazenados no banco de dados; caso contrário, realiza uma consulta à API RAWG, persistindo as informações obtidas para reutilizações futuras e reduzindo o número de requisições externas. De forma semelhante, as funcionalidades de recomendação utilizam a plataforma Groq para a geração das sugestões com base nas preferências informadas pelo usuário. Após o processamento da solicitação, a API retorna ao cliente uma resposta em formato JSON contendo os dados requisitados ou o resultado da operação realizada.

Figura 3 – Arquitetura do Sistema



Fonte: Elaborado pelo autor (2026).

3.5 DIAGRAMA DE ENTIDADES-RELACIONAMENTOS

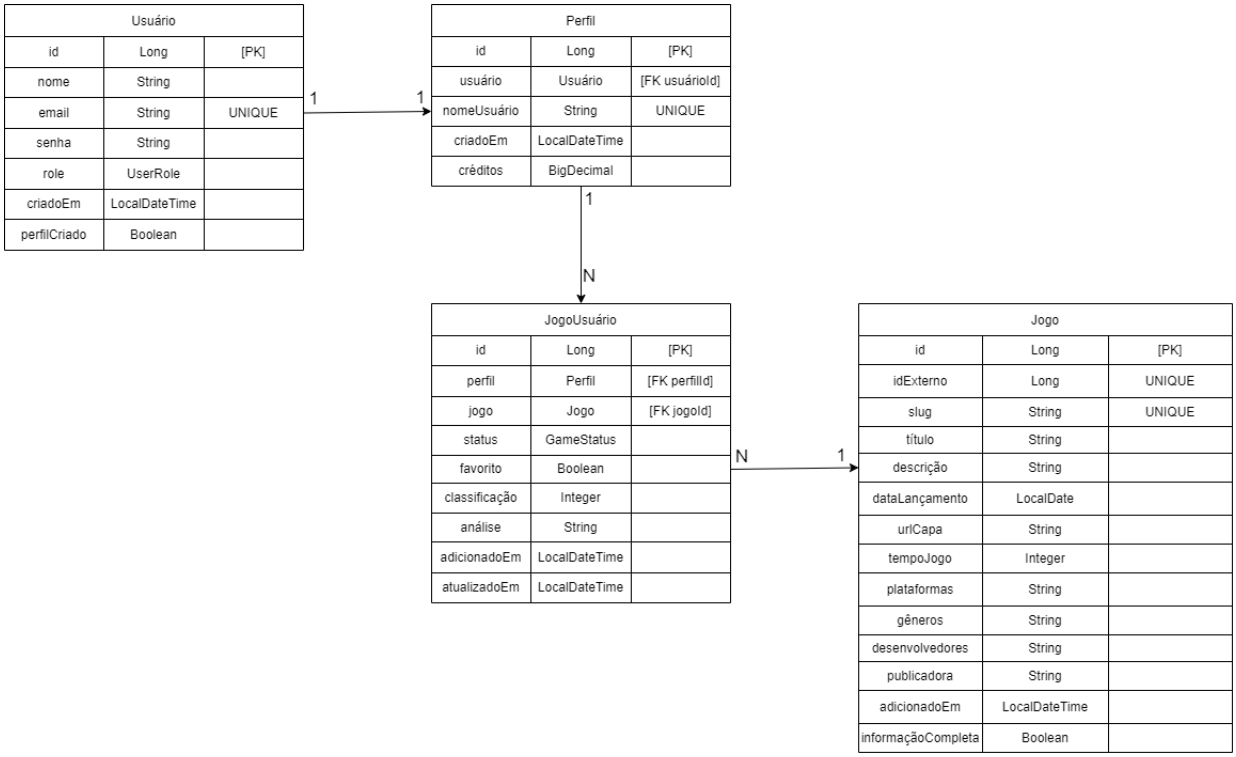
O diagrama apresentado na Figura 4 representa a estrutura lógica do banco de dados da aplicação, evidenciando as entidades responsáveis pelo armazenamento das informações e os relacionamentos existentes entre elas. Esse modelo serviu como base para a implementação do esquema relacional no PostgreSQL, garantindo a organização dos dados e a integridade referencial entre as tabelas.

O modelo é composto pelas entidades Usuário, Perfil, UsuárioJogo e Jogo. A entidade Usuário armazena os dados necessários para autenticação e gerenciamento das contas, enquanto Perfil concentra as informações complementares de cada usuário. A entidade Jogo reúne os dados dos títulos armazenados pelo sistema, provenientes da integração com a API RAWG. Por sua vez, UsuárioJogo atua como uma entidade

associativa entre perfis e jogos, armazenando informações específicas da interação do usuário com cada título, como status, avaliação, comentário e indicação de favorito.

A modelagem adotada buscou reduzir a redundância de dados e favorecer a normalização do banco, permitindo que informações gerais dos jogos sejam armazenadas apenas uma vez, enquanto os dados específicos de cada usuário sejam registrados separadamente na entidade associativa. Essa organização facilita a manutenção da base de dados, melhora a consistência das informações e torna o modelo mais adequado para futuras expansões da aplicação.

Figura 4 – Diagrama de Entidades-Relacionamentos



Fonte: Elaborado pelo autor (2026).

4 SOFTWARE

Esta seção detalha a implementação prática do *software*, incluindo sua implantação e testes.

4.1 IMPLANTAÇÃO

O desenvolvimento do sistema foi realizado em ambiente local utilizando a linguagem Java e o *framework* Spring Boot, com gerenciamento de dependências pelo Apache Maven e controle de versão por meio do Git. Durante essa etapa, a aplicação foi executada localmente para validação das funcionalidades implementadas, enquanto o banco de dados PostgreSQL foi acessado por meio do serviço em nuvem Neon, permitindo o armazenamento persistente dos dados da aplicação.

Para a implantação em produção, a API foi hospedada na plataforma Render, um serviço de computação em nuvem que automatiza o processo de compilação, implantação e publicação de aplicações a partir de repositórios Git. A integração com o GitHub possibilitou que novas versões da aplicação fossem disponibilizadas por meio de *deploys* automáticos após a atualização do código-fonte, simplificando o processo de publicação e manutenção do sistema.

O banco de dados permaneceu hospedado na plataforma Neon, que oferece instâncias gerenciadas de PostgreSQL em nuvem. A comunicação entre a API e o banco de dados foi realizada por meio de conexão segura utilizando variáveis de ambiente para armazenamento das credenciais de acesso. Da mesma forma, informações sensíveis, como as chaves de acesso às APIs externas e o segredo utilizado para geração dos *tokens* JWT, também foram configuradas como variáveis de ambiente na plataforma de hospedagem, evitando sua exposição no código-fonte e seguindo boas práticas de segurança.

4.2 TESTES

Para assegurar que o sistema atenda aos padrões esperados de estabilidade e exatidão, foram implementadas diferentes frentes de validação. Inicialmente, realizaram-se os testes unitários, focados na verificação de unidades individuais de código, como funções e componentes isolados, garantindo que a lógica interna de cada módulo opere corretamente de forma independente. Adicionalmente, empregou-se a avaliação baseada em *LLM as a judge*, utilizada para validar as respostas geradas pelos recursos de inteligência artificial, como as recomendações de jogos. Essa abordagem utiliza um modelo de linguagem para analisar e julgar critérios de relevância, coerência e qualidade das saídas fornecidas ao usuário.

4.2.1 Validação Funcional da API

Durante o desenvolvimento da aplicação, foram realizados testes funcionais contínuos com o objetivo de validar o comportamento da API a cada funcionalidade implementada. À medida que novas funcionalidades eram implementadas, eram realizados testes funcionais com o objetivo de verificar se os requisitos especificados estavam sendo atendidos e se os componentes do sistema se comportavam conforme o esperado. Essa abordagem permitiu identificar falhas ainda durante a implementação, facilitando sua correção antes da integração com os demais módulos da aplicação.

Para a execução desses testes foi utilizada a ferramenta Postman [1], responsável pelo envio de requisições HTTP aos *endpoints* da API e pela análise das respostas retornadas. Cada rota desenvolvida foi validada individualmente, verificando aspectos como o correto recebimento dos parâmetros enviados pelo cliente, a execução da lógica de negócio, a persistência e recuperação dos dados no banco de dados, os códigos de status HTTP retornados e a estrutura dos objetos JSON produzidos pela aplicação. Também foram testados cenários de erro, como requisições inválidas, ausência de autenticação, acesso não autorizado e tentativas de consulta ou alteração de recursos inexistentes, garantindo que a API respondesse adequadamente em situações excepcionais.

Além da validação isolada de cada *endpoint*, também foram realizados testes envolvendo o fluxo completo das principais funcionalidades do sistema. Foram verificados processos como cadastro e autenticação de usuários, gerenciamento de perfis, busca e adição de jogos à biblioteca, atualização das informações armazenadas e geração de recomendações por meio da integração com o serviço de inteligência artificial. Esses testes possibilitaram validar a comunicação entre os diferentes componentes da aplicação, assegurando que as funcionalidades implementadas operassem de forma integrada e em conformidade com os requisitos estabelecidos para o projeto.

Como os testes foram realizados de maneira incremental, acompanhando a evolução do desenvolvimento da API, foi possível corrigir inconsistências de forma antecipada e validar continuamente a estabilidade das funcionalidades implementadas. Essa estratégia contribuiu para reduzir a ocorrência de erros durante a integração entre os módulos e proporcionou maior confiabilidade ao comportamento final da aplicação.

4.2.2 Metodologia de Avaliação das Recomendações

A avaliação da qualidade das recomendações foi realizada por meio da abordagem *LLM as a Judge*, na qual um modelo de linguagem é utilizado como avaliador das respostas produzidas por outro modelo. Essa abordagem tem sido empregada como uma alternativa para a avaliação de respostas geradas por modelos de linguagem,

[1] `https://www.postman.com/`

permitindo considerar aspectos relacionados à qualidade e à adequação das respostas a partir de critérios previamente estabelecidos (Zheng *et al.*, 2023). Diferentemente de métricas tradicionais, que normalmente verificam aspectos mais restritos das respostas, essa abordagem possibilita uma avaliação mais ampla do conteúdo gerado.

Neste trabalho, o modelo avaliador recebeu como entrada o jogo de referência, a descrição da experiência desejada pelo usuário e a lista de jogos recomendados pelo sistema. Com base nessas informações, o modelo analisou a qualidade das recomendações utilizando critérios previamente definidos, atribuindo notas individuais para cada um deles e apresentando uma justificativa para a avaliação realizada.

Os critérios considerados na avaliação contemplam a similaridade da experiência proposta, a coerência entre as recomendações e a solicitação do usuário e a qualidade das justificativas apresentadas. A partir das notas obtidas, foi possível analisar quantitativamente e qualitativamente o desempenho da funcionalidade de recomendação implementada.

4.2.3 Construção da Base de Testes

Para avaliar a funcionalidade de recomendação de jogos, foi elaborado um conjunto de 20 jogos de referência, abrangendo diferentes gêneros, estilos visuais e propostas de jogabilidade, com o objetivo de representar uma variedade de perfis e preferências de jogadores. Para cada título, foi definida uma breve descrição sintetizando a principal experiência que poderia motivar um usuário a buscar jogos semelhantes. Essa etapa teve como objetivo estabelecer uma base diversificada para a realização dos testes, reduzindo vieses relacionados à concentração em um único gênero ou estilo de jogo.

A seleção dos jogos e a descrição das experiências foram geradas com o auxílio do modelo GPT-5.5, disponibilizado pela OpenAI, buscando representar de forma objetiva os principais aspectos que caracterizam a experiência proporcionada por cada título. A Tabela 3 apresenta os jogos utilizados durante a avaliação e a respectiva experiência considerada como referência para a geração das recomendações.

Tabela 3 – Base de jogos utilizada nos testes.

Nome do Jogo	**Experiência**
The Legend of Zelda: Breath of the Wild	Exploração livre em um mundo aberto vivo.
Dark Souls III	Combate desafiador e sensação de superação.
Stardew Valley	Vida tranquila, fazenda e relacionamentos acolhedores.

Nome do Jogo	Experiência
DOOM Eternal	Combate frenético com mobilidade intensa.
Minecraft	Criatividade ilimitada e exploração sem limites.
Hades	Progressão viciante em partidas rápidas.
Portal 2	Quebra-cabeças inteligentes com mecânicas criativas.
Resident Evil 2	Tensão constante e sobrevivência assustadora.
Forza Horizon 5	Corridas livres em cenários deslumbrantes.
Age of Empires II: Definitive Edition	Estratégia clássica e gerenciamento de impérios.
Celeste	Plataforma precisa e desafio recompensador.
The Sims 4	Criar histórias e controlar vidas.
Disco Elysium	Narrativa profunda com escolhas impactantes.
Subnautica	Exploração submarina com mistério constante.
Red Dead Redemption 2	Imersão profunda no velho oeste.
Overcooked! 2	Cooperação caótica e divertida.
Monster Hunter: World	Caçadas cooperativas contra monstros gigantes.
Spider Man Remastered	Combate intenso e viagem satisfatória.
Outer Wilds	Descobertas movidas pela curiosidade.
NieR:Automata	Narrativa filosófica e combate hack-in-slash.

Fonte: Elaborado pelo autor (2026).

4.2.4 Geração das Recomendações

Após a definição da base de testes, foi realizada a execução da funcionalidade de recomendação por meio da API desenvolvida neste trabalho. Para cada um dos 20 jogos selecionados, foram enviados o título do jogo e a respectiva descrição da experiência que o usuário desejaria reviver, conforme apresentado na Tabela 3. A aplicação processou essas informações e solicitou ao modelo de linguagem a geração de uma lista de recomendações de jogos semelhantes.

Para a geração das recomendações foi utilizada a plataforma Groq, empregando o modelo openai/gpt-oss-120b, integrado à aplicação por meio de sua API. A escolha desse modelo deve-se à sua capacidade de interpretar descrições em linguagem natural e identificar relações semânticas entre diferentes jogos, permitindo

recomendar títulos com base não apenas em características técnicas, mas também na experiência desejada pelo usuário.

Como resultado desse processo, foram obtidas 20 listas de recomendações, contendo cinco jogos para cada caso de teste. Essas recomendações serviram como base para a etapa de avaliação da qualidade dos resultados, apresentada na subseção seguinte. A Tabela 4 apresenta, como exemplo, a saída estruturada gerada pelo sistema para o jogo de referência *Red Dead Redemption 2*, com base na experiência "Imersão profunda no velho oeste.".

Tabela 4 – Exemplo de saída de recomendações gerada pelo sistema.

Jogo Recomendado	**Plataformas**	**Similaridade**	**Justificativa**
Red Dead Redemption: Undead Nightmare	PC, PlayStation 3, Xbox 360, PlayStation 4, Xbox One	MUITO	Expande a mesma imersão no Velho Oeste com missões adicionais e atmosfera sombria.
Red Dead Redemption	PC, PlayStation 3, Xbox 360, PlayStation 4, Xbox One	MUITO	Apresenta o mundo aberto e a narrativa profunda que definem a experiência *western*.
Desperados III	PC, PlayStation 4, Xbox One, Nintendo Switch, PlayStation 5, Xbox Series X/S	MÉDIO	Combina estratégia tática e exploração em um cenário western detalhado.
Call of Juarez: Gunslinger	PC, PlayStation 4, Xbox One, Nintendo Switch	MÉDIO	Foca em tiroteios rápidos e narrativa de pistoleiro dentro do Velho Oeste.
Hard West	PC, PlayStation 4, Xbox One, Nintendo Switch	POUCO	Mistura elementos sobrenaturais com ambientação *western*, oferecendo uma atmosfera distinta.

Fonte: Elaborado pelo autor (2026).

4.2.5 Avaliação dos Resultados

Após a geração das recomendações, cada um dos 20 casos de teste foi submetido a uma etapa de avaliação utilizando a abordagem *LLM as a Judge*. Nessa etapa, o modelo de linguagem Gemini 3.6 Flash (Google, 2026), foi empregado para analisar a qualidade das recomendações produzidas pelo sistema, considerando o jogo de origem, a experiência desejada pelo usuário e a lista de jogos recomendados. O objetivo dessa avaliação foi obter uma medida qualitativa padronizada sobre a capacidade do sistema em sugerir títulos coerentes com as preferências informadas.

Para garantir consistência entre todas as avaliações, foi elaborado um *prompt* contendo instruções detalhadas sobre o papel do modelo avaliador, os critérios de análise e o formato esperado da resposta. O *prompt* orienta o modelo a atuar exclusivamente como avaliador, sem gerar novas recomendações ou modificar os resultados produzidos pela aplicação.

A avaliação das recomendações foi estruturada a partir de cinco critérios, definidos com o objetivo de analisar diferentes aspectos da qualidade das respostas geradas pelo sistema. Os critérios contemplam desde características técnicas, como a similaridade das mecânicas e da ambientação dos jogos recomendados, até aspectos relacionados à capacidade de reproduzir a experiência desejada pelo usuário e à coerência das justificativas fornecidas pelo modelo. A Tabela 5 apresenta os critérios considerados durante o processo de avaliação e uma breve descrição de cada um deles.

Tabela 5 – Critérios de avaliação das recomendações.

Critério	**Descrição**
Gameplay	Similaridade das mecânicas principais entre o jogo original e os recomendados.
Ambientação	Similaridade do universo, atmosfera, estilo visual e temática.
Experiência	Capacidade das recomendações reproduzirem a experiência desejada pelo usuário.
Justificativas	Coerência das justificativas apresentadas para cada recomendação.
Qualidade Geral	Avaliação global considerando todos os critérios anteriores.

Fonte: Elaborado pelo autor (2026).

Para cada critério, o modelo avaliador atribuiu uma nota inteira em uma escala de 1 a 5, em que 1 representa um desempenho muito ruim e 5 representa um desempenho excelente, conforme apresentado na Tabela 6. Além da pontuação, foi gerada uma breve análise justificando a nota atribuída em cada critério. Ao final da

avaliação, o modelo calculou automaticamente a média aritmética simples das cinco notas e apresentou uma conclusão resumindo a qualidade geral das recomendações produzidas pelo sistema.

Tabela 6 – Escala de avaliação dos critérios.

Nota	**Interpretação**
1	Muito ruim
2	Ruim
3	Regular
4	Boa
5	Excelente

Fonte: Elaborado pelo autor (2026).

Após a execução do processo de avaliação, foram obtidas as pontuações atribuídas pelo modelo para cada um dos 20 casos de teste. Para cada jogo de referência, o avaliador analisou a lista de recomendações produzida pelo sistema e atribuiu notas aos cinco critérios definidos, além de calcular automaticamente a média aritmética das avaliações e gerar uma breve análise sobre a qualidade geral das recomendações.

A Tabela 7 apresenta os resultados obtidos em cada caso de teste, incluindo as notas atribuídas aos critérios de *Gameplay*, Ambientação, Experiência, Justificativas e Qualidade Geral, bem como a média final calculada pelo modelo avaliador. Esses resultados constituem a base para a análise do desempenho da funcionalidade de recomendação apresentada na seção seguinte.

Tabela 7 – Resultados detalhados da avaliação das recomendações por jogo.

Jogo	**Gameplay**	**Ambientação**	**Experiência**	**Justificativas**	**Qualidade Geral**	**Média**
The Legend of Zelda: Breath of the Wild	4	4	5	5	5	4,6
Dark Souls III	5	4	5	5	5	4,8
Stardew Valley	4	4	5	5	5	4,6
DOOM Eternal	4	3	5	5	5	4,4
Minecraft	4	3	5	5	5	4,4
Hades	5	3	5	5	5	4,6
Portal 2	4	3	5	5	5	4,4
Resident Evil 2	4	4	5	5	5	4,6
Forza Horizon 5	4	4	5	4	4	4,2
Age of Empires II: Definitive Edition	4	4	5	5	5	4,6
Celeste	4	3	5	5	5	4,4

Jogo	Gameplay	Ambientação	Experiência	Justificativas	Qualidade Geral	Média
The Sims 4	3	3	5	5	4	4,0
Disco Elysium	3	3	5	5	5	4,2
Subnautica	4	5	5	5	5	4,8
Red Dead Redemption 2	3	5	5	5	5	4,6
Overcooked! 2	4	3	5	5	5	4,4
Monster Hunter: World	4	3	5	5	5	4,4
Spider Man Remastered	4	3	5	5	5	4,4
Outer Wilds	4	4	5	5	5	4,6
NieR:Automata	4	4	5	5	5	4,6

Fonte: Elaborado pelo autor (2026).

Com o objetivo de obter uma visão consolidada do desempenho da funcionalidade de recomendação, foram calculadas estatísticas descritivas a partir das avaliações realizadas em todos os casos de teste. Essas métricas permitem sintetizar o comportamento geral do sistema, facilitando a identificação de tendências e a comparação entre os diferentes critérios avaliados.

A Tabela 8 apresenta as estatísticas gerais obtidas, incluindo a média das pontuações atribuídas a cada critério, a média geral das avaliações, os valores máximo e mínimo, bem como o desvio padrão amostral calculado. Essas informações possibilitam analisar o desempenho da aplicação de forma agregada, complementando a análise individual apresentada anteriormente e fornecendo uma visão mais abrangente sobre a qualidade das recomendações geradas.

Tabela 8 – Estatísticas gerais da avaliação das recomendações.

Critério	Média	Máximo	Mínimo	Desvio Padrão
Gameplay	3,95	5	3	0,51
Ambientação	3,60	5	3	0,68
Experiência	5,00	5	5	0,00
Justificativas	4,95	5	4	0,22
Qualidade Geral	4,90	5	4	0,31

Fonte: Elaborado pelo autor (2026).

De forma geral, os resultados consolidados na Tabela 8 indicam que o sistema pode ser capaz de produzir recomendações altamente coerentes com as experiências descritas pelos usuários, alcançando uma média geral de 4,48 com um desvio padrão de 0,20. As maiores pontuações concentraram-se no critério Experiência, que atingiu nota máxima em todos os casos de teste (média 5,00 e desvio padrão 0,00), seguido pelos critérios de Justificativas (média 4,95 e desvio padrão 0,22) e Qualidade Geral

(média 4,90 e desvio padrão 0,31), sugerindo que o modelo conseguiu interpretar adequadamente as preferências expressas em linguagem natural e apresentar explicações consistentes. Em contrapartida, critérios como Gameplay (média 3,95 e desvio padrão 0,51) e Ambientação (média 3,60 e desvio padrão 0,68) apresentaram médias inferiores e maior variação entre os casos de teste, evidenciando que alguns jogos compartilham experiências semelhantes mesmo pertencendo a gêneros, mecânicas ou universos distintos.

É importante destacar que a abordagem *LLM as a Judge* constitui uma avaliação automatizada baseada no conhecimento do modelo de linguagem, devendo ser interpretada como um instrumento de apoio à análise e não como uma medida absoluta da qualidade das recomendações. Apesar dessa limitação, o método permitiu realizar uma avaliação consistente e reproduzível para todos os 20 casos de teste considerados neste trabalho.

5 CONSIDERAÇÕES FINAIS

O presente trabalho apresentou o desenvolvimento de um sistema *web* para gerenciamento de catálogos pessoais de jogos digitais, integrando funcionalidades para organização da biblioteca de jogos e geração de recomendações por meio de modelos de linguagem. Ao longo do desenvolvimento, foi possível aplicar conceitos de engenharia de *software*, modelagem de sistemas, desenvolvimento de APIs REST, integração com serviços externos e persistência de dados, consolidando conhecimentos adquiridos durante a formação acadêmica.

Os resultados obtidos demonstram que os objetivos propostos foram alcançados, uma vez que a aplicação implementa as funcionalidades planejadas e foi validada por meio de testes funcionais e da avaliação da qualidade das recomendações geradas. Entre os principais desafios enfrentados destacam-se a integração entre diferentes serviços, o planejamento da arquitetura da aplicação e a definição de uma estratégia para avaliação das recomendações, aspectos que contribuíram para o aprimoramento técnico durante o desenvolvimento do projeto.

Como limitações deste trabalho, destacam-se o escopo definido para o Relatório Técnico de Software e o tempo disponível para seu planejamento e desenvolvimento, fatores que direcionaram os esforços para a implementação, documentação e validação da camada *backend* da aplicação e escrita do tranalho.

Como perspectivas para a continuidade do projeto, destacam-se o desenvolvimento de uma interface gráfica integrada à API, proporcionando uma experiência mais completa ao usuário, bem como o aprimoramento da funcionalidade de recomendação por meio da utilização de informações do histórico e das preferências dos usuários. Além disso, a realização de avaliações com usuários reais poderá complementar a validação da solução desenvolvida.

DECLARAÇÃO DO USO DE INTELIGÊNCIA ARTIFICIAL GENERATIVA

Durante a elaboração deste trabalho, foram utilizadas ferramentas de inteligência artificial generativa para apoio em tarefas específicas. O modelo ChatGPT GPT-5.5 [1] foi utilizado para revisão gramatical e como auxílio na busca e organização de referências bibliográficas e na discussão de conceitos técnicos durante a fase de pesquisa. Todo o conteúdo técnico, as análises, os resultados e as conclusões apresentados são de responsabilidade do autor, que revisou e validou integralmente as contribuições geradas com o apoio dessas ferramentas.

[1] https://chatgpt.com/

REFERÊNCIAS

ADOMAVICIUS, Gediminas; TUZHILIN, Alexander. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. **IEEE Transactions on Knowledge and Data Engineering**, v. 17, n. 6, p. 734–749, 2005. DOI: 10.1109/TKDE.2005.99.

BAUER, Christian; KING, Gavin; GREGORY, Gary. **Java Persistence with Hibernate**. 2. ed. Shelter Island: Manning Publications, 2015. ISBN 9781617290459.

BAWDEN, David; ROBINSON, Lyn. The Dark Side of Information: Overload, Anxiety and Other Paradoxes and Pathologies. **Journal of Information Science**, v. 35, n. 2, p. 180–191, 2009. DOI: 10.1177/0165551508095781.

BORGMAN, Christine L. What are Digital Libraries? Competing Visions. **Information Processing & Management**, v. 35, n. 3, p. 227–243, 1999. DOI: 10.1016/S0306-4573(98)00059-4.

DEITEL, Paul; DEITEL, Harvey. **Java: How to Program, Early Objects**. 11. ed. Boston: Pearson, 2017. ISBN 9780134743356.

DB-ENGINES. **DB-Engines Ranking**. 2025. Disponível em: https://db-engines.com/en/ranking.

GOOGLE. **Gemini API: Models**. [*S. l.: s. n.*], 2026. Google AI for Developers. Documentação oficial da API Gemini. Disponível em: https://ai.google.dev/gemini-api/docs/models.

GROQ, INC. **Groq API Documentation**. 2026. Disponível em: https://console.groq.com/docs.

HARTE, Jesse; ZORGDRAGER, Wouter; LOURIDAS, Panos; KATSIFODIMOS, Asterios; JANNACH, Dietmar; FRAGKOULIS, Marios. Leveraging Large Language Models for Sequential Recommendation. *In:* PROCEEDINGS of the 17th ACM Conference on Recommender Systems (RecSys '23). [*S. l.*]: Association for Computing Machinery, 2023. p. 1096–1102. DOI: 10.1145/3604915.3610639.

IYENGAR, Sheena S.; LEPPER, Mark R. When Choice Is Demotivating: Can One Desire Too Much of a Good Thing? **Journal of Personality and Social Psychology**, v. 79, n. 6, p. 995–1006, 2000. DOI: 10.1037/0022-3514.79.6.995.

JETBRAINS. **The State of Developer Ecosystem 2024**. 2024. Disponível em: https://www.jetbrains.com/lp/devecosystem-2024/.

NEON, INC. **Neon Documentation**. 2026. Disponível em: https://neon.com/docs.

NEWZOO. **Year in Review: 2025 to Date**. 18 dez. 2025. Disponível em: https://newzoo.com/articles/year-in-review-2025.

ORACLE CORPORATION. **Java Documentation**. 2026. Disponível em: https://docs.oracle.com/en/java/.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL Documentation**. 2026. Disponível em: https://www.postgresql.org/docs/.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software: Uma Abordagem Profissional**. 8. ed. Porto Alegre: AMGH, 2016. ISBN 9788580555332.

RAWG. **RAWG Video Games Database API Documentation**. 2026. Disponível em: https://rawg.io/apidocs.

RED HAT, INC. **Hibernate ORM Documentation**. 2026. Disponível em: https://hibernate.org/orm/documentation/.

REDGATE SOFTWARE. **Flyway Documentation**. 2026. Disponível em: https://documentation.red-gate.com/fd.

RENDER. **Render Documentation**. 2026. Disponível em: https://render.com/docs.

RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha (ed.). **Recommender Systems Handbook**. 2. ed. New York: Springer, 2015. ISBN 978-1-4899-7636-9. DOI: 10.1007/978-1-4899-7637-6.

SCHWARTZ, Barry. **The Paradox of Choice: Why More Is Less**. New York: Ecco, 2004. ISBN 9780060005683.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

STACK OVERFLOW. **2024 Stack Overflow Developer Survey**. 2024. Disponível em: https://survey.stackoverflow.co/2024/.

VMWARE, INC. **Spring Boot Reference Documentation**. 2026. Disponível em: https://docs.spring.io/spring-boot/reference/.

WALLS, Craig. **Spring in Action**. 6. ed. Shelter Island: Manning Publications, 2022. ISBN 9781617297571.

WANG, Qi; LI, Jindong; WANG, Shiqi; XING, Qianli; NIU, Runliang; KONG, He; LI, Rui; LONG, Guodong; CHANG, Yi; ZHANG, Chengqi. Towards Next-Generation LLM-based Recommender Systems: A Survey and Beyond. **CoRR**, abs/2410.19744, 2024. DOI: 10.48550/arXiv.2410.19744.

ZHENG, Lianmin; CHIANG, Wei-Lin; SHENG, Ying; ZHUANG, Siyuan; WU, Zhanghao; ZHUANG, Yonghao; LIN, Zi; LI, Zhuohan; LI, Dacheng; XING, Eric P.; ZHANG, Hao; GONZALEZ, Joseph E.; STOICA, Ion. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. **Advances in Neural Information Processing Systems**, v. 36, 2023.

APÊNDICE A – PROMPT UTILIZADO PARA GERAÇÃO DA BASE DE DADOS

```
Busque 20 jogos variados, cobrindo diferentes gêneros, estilos visuais e propostas de gameplay, para que eu possa fazer testes em uma funcionalidade de recomendação.

As experiências devem ser curtas, objetivas e representar o principal motivo pelo qual alguém gostaria de reviver aquele jogo. Exemplos:

The Legend of Zelda: Breath of the Wild: "Exploração livre em um mundo aberto vivo."
Dark Souls III: "Combate desafiador e sensação de superação."

Busque incluir jogos de estilos bastante diferentes, como RPGs, FPS, jogos de corrida, estratégia, terror, simuladores, roguelikes, plataforma, aventura, puzzle, sandbox, entre outros.

Retorne apenas objetos JSON, um após o outro (sem array, sem numeração e sem texto adicional), exatamente no seguinte formato:

{
  "game": "Nome completo do jogo",
  "experience": "Descrição curta da experiência."
}

A descrição da experiência deve conter no máximo 8 palavras.
```

APÊNDICE B – PROMPT UTILIZADO PARA GERAÇÃO DAS RECOMENDAÇÕES

```
Seu objetivo é recomendar jogos que proporcionem uma experiência semelhante àquela que o jogador deseja reviver, e não apenas jogos do mesmo gênero ou da mesma franquia.

Considere principalmente a descrição do jogador e utilize o jogo informado apenas como contexto. Analise aspectos como mecânicas, exploração, combate, narrativa, atmosfera, progressão, liberdade, desafio e ritmo de gameplay.

Entrada:

Jogo escolhido:
(Nome)

Experiência que o jogador deseja reviver:
(Experiência)

Regras:

- Retorne exatamente (Quantidade) jogos.
- Priorize a experiência desejada, não apenas gênero ou franquia.
- Distribua os jogos nas seguintes categorias de semelhança:
  - MUITO
  - MÉDIO
  - POUCO
- Priorize sempre recomendar os jogos mais semelhantes.
- O motivo deve conter apenas uma frase curta explicando a recomendação.
- Não retorne nenhuma explicação fora do JSON.

Formato da resposta:

[
  {
    "name": "Nome completo do jogo",
    "platforms": [
      "PC",
      "PlayStation 5",
```

```
      "Xbox Series X/S"
    ],
    "similarity": "MUITO",
    "reason": "Combina pela liberdade de exploração e pelo foco em
   missões abertas."
  },
  {
    "name": "Nome completo do jogo",
    "platforms": [
      "PC",
      "PlayStation 5"
    ],
    "similarity": "MUITO",
    "reason": "Compartilha elementos importantes da experiência
   desejada."
  },
  {
    "name": "Nome completo do jogo",
    "platforms": [
      "PC",
      "Xbox Series X/S"
    ],
    "similarity": "MUITO",
    "reason": "Entrega uma experiência semelhante em seus principais
   aspectos."
  },
  {
    "name": "Nome completo do jogo",
    "platforms": [
      "Nintendo Switch",
      "PC"
    ],
    "similarity": "MÉDIO",
    "reason": "Possui semelhanças relevantes, mas com foco diferente
   ."
  },
  {
    "name": "Nome completo do jogo",
    "platforms": [
      "PC"
    ],
    "similarity": "POUCO",
```

```
    "reason": "Compartilha apenas parte da experiência procurada."
  }
]
```

APÊNDICE C – PROMPT UTILIZADO PARA AVALIAÇÃO DAS RECOMENDAÇÕES

```
Sua função é avaliar exclusivamente a qualidade das recomendações produzidas por outro sistema de recomendação.

Você não deve gerar novas recomendações, alterar a lista recebida ou corrigir informações. Apenas avaliar.

Considere o jogo original, a experiência desejada pelo usuário e a lista de jogos recomendados.

Avalie os seguintes critérios:

1. Gameplay
Avalie o quanto as mecânicas centrais dos jogos recomendados se assemelham às do jogo original.

2. Ambientação
Avalie o quanto o estilo visual, atmosfera, temática, mundo e direção artística dos jogos recomendados são semelhantes ao jogo original.

3. Experiência
Avalie o quanto as recomendações permitem ao jogador reviver a experiência descrita pelo usuário. Este critério deve considerar principalmente o campo "Experiência", mesmo que os jogos pertençam a gêneros diferentes.

4. Justificativas
Avalie se a justificativa apresentada para cada recomendação é coerente, plausível e realmente explica a relação entre o jogo recomendado e o jogo original.

5. Qualidade Geral
Considere todos os critérios anteriores e atribua uma nota geral para a lista de recomendações.

Escala de notas:

```

```
1 = Muito ruim
2 = Ruim
3 = Regular
4 = Boa
5 = Excelente

Regras:

- Utilize apenas seu conhecimento sobre os jogos e as justificativas fornecidas.
- Não sugira novos jogos.
- Não altere a lista recebida.
- Não penalize a lista apenas porque alguns jogos possuem níveis diferentes de similaridade ("MUITO", "MÉDIO" ou "POUCO"), caso essa distribuição seja coerente com as justificativas apresentadas.
- Seja consistente e imparcial.

Para cada critério:

- Atribua uma nota inteira de 1 a 5.
- Escreva uma análise com no máximo uma frase.

Ao final:

- Calcule a média aritmética simples das cinco notas.
- Escreva uma conclusão resumindo a qualidade geral das recomendações.

Retorne EXCLUSIVAMENTE um JSON válido exatamente no seguinte formato:

{
  "gameplay": {
    "score": 0,
    "analysis": ""
  },
  "setting": {
    "score": 0,
    "analysis": ""
  },
  "experience": {
    "score": 0,
```

```
    "analysis": ""
  },
  "justification": {
    "score": 0,
    "analysis": ""
  },
  "overall_quality": {
    "score": 0,
    "analysis": ""
  },
  "average_score": 0.0,
  "overall_analysis": ""
}

Dados para avaliação:

Jogo original: (Jogo)

Experiência desejada: (Experiência)

Lista de recomendações: (Lista)
```