Análise de Requisitos da Plataforma Nexus

Versão atual do Documento: 1.0

Equipe de Desenvolvimento

Engenheiro de Software: Igor Bonfim

Tecnologias utilizadas no Projeto

Linguagem de programação: Golang
Banco de Dados: PostgreSQL
FrontEnd utilizado: Next.JS
BackEnd utilizado: Golang

INTRODUÇÃO

Descrição Geral do Sistema

Nexus é um rede social para League of Legends projetada para conectar jogadores por meio de interações sociais. Nesta primeira versão, o foco é permitir que os usuários criem, visualizem, comentem e interajam com posts de outros jogadores, oferecendo um feed dinâmico e intuitivo.

Público Alvo

Jogadores de League of Legends:
Pessoas que desejam compartilhar experiências, clipes de partidas e ideias relacionadas ao jogo. O foco inicial são jogadores que buscam interagir de forma simples e direta.

Objetivo:
Oferecer uma plataforma que permita interações sociais através de posts, criando um ambiente inicial para conexões entre jogadores de League of Legends.

TIPOS DE REQUISITOS

Os requisitos do sistema são divididos em duas categorias: **Funcionais** e **Não Funcionais.**

- **Requisitos Funcionais** definem o que o sistema deve fazer.

- **Requisitos Não Funcionais (ou Atributos de Qualidade)** definem como o sistema deve funcionar.

Níveis de Prioridade dos Requisitos

Os requisitos são classificados em três categorias:

Obrigatório -> O sistema não funciona sem este requisito. É imprescindível e deve ser implementado obrigatoriamente.

Necessário -> O sistema funciona, mas de forma insatisfatória sem este requisito. Deve ser implementado, mas o sistema ainda pode ser usado sem ele.

Opcional -> O sistema funciona bem sem este requisito. Pode ser deixado para uma versão futura se não houver tempo para implementá-lo agora.

REQUISITOS FUNCIONAIS

RF 001 - Autenticação de Usuário

Descrição:O sistema deve permitir que os usuários se autentiquem usando um e-mail e uma senha. O usuário deve ser capaz de criar uma conta, fazer login e recuperar a senha em caso de esquecimento. Após a autenticação, o usuário será redirecionado para a tela principal do aplicativo.

Entradas e Pré-condições:

- O usuário deve fornecer um endereço de e-mail e uma senha.
- O e-mail deve ser único e não deve estar registrado em outra conta.

Saídas e Pós-condições:

- Após a autenticação bem-sucedida, o usuário será direcionado para a tela principal do aplicativo.
- Se a autenticação falhar, uma mensagem de erro será exibida.

Prioridade: Obrigatório

RF 002 - Gerenciar Posts e Conteúdo

Descrição: O sistema deve permitir que os usuários criem posts com texto, imagens ou vídeos. Eles também devem poder curtir e comentar nos posts de outros usuários.

Entradas e Pré-condições:

- O usuário deve estar autenticado no sistema.

Saídas e Pós-condições:

- Os posts criados devem ser exibidos no feed do usuário e de seus seguidores. Os comentários e curtidas devem ser registrados no sistema e exibidos em tempo real.

Prioridade: Obrigatório

RF 003 - Feed de Posts

Descrição:
O sistema deve exibir um feed dinâmico com os posts criados pelos usuários, ordenados por ordem cronológica ou relevância.

Entradas e Pré-condições:

- O usuário deve estar autenticado para acessar o feed.

Saídas e Pós-condições:

- Os posts aparecem em ordem cronológica ou com base em relevância. A paginação deve ser utilizada para carregar mais posts conforme necessário.

Prioridade: Obrigatório

RF 004 - Integração com a Riot Games

Descrição:
O sistema deve integrar-se à API oficial da Riot Games para buscar informações do perfil do invocador, como elo, campeões favoritos e partidas recentes.

Entradas e Pré-condições:

- O usuário deve fornecer seu nome de invocador. A API da Riot Games deve estar disponível.

Saídas e Pós-condições:

- As estatísticas do jogador são exibidas em seu perfil. O progresso do jogador é atualizado automaticamente.

Prioridade: Opcional

REQUISITOS NÃO FUNCIONAIS

RNF 001 - Arquitetura de Software

Descrição: O aplicativo vai utilizar uma arquitetura de software moderna, possibilitando que o aplicativo seja escalável, sustentável e totalmente testável por outros desenvolvedores futuros.

RNF 002 - Escalabilidade

Descrição: O sistema deve suportar pelo menos 5.000 usuários simultâneos e utilizar técnicas de caching para manter a performance.

RNF 003 - Segurança

Descrição: Todas as senhas devem ser armazenadas com hashing seguro (bcrypt) e o sistema deve operar exclusivamente via HTTPS.

RNF 004 - Desempenho

Descrição: Desempenho Descrição: As respostas do backend devem ter latência inferior a 300ms para consultas simples e 500ms para carregamento de posts no feed.

RNF 005 - Tecnologia Utilizada

Descrição: A plataforma será desenvolvida utilizando a linguagem Golang, com banco de dados PostgreSQL e Next.js.

Stakeholders

Definição: Partes interessadas que influenciam ou são afetadas pelo projeto. Inclui usuários finais, clientes, gerentes, e equipes de desenvolvimento, testes e suporte.

Stakeholders Importantes:

- **Usuários Finais:** Utilizadores diários do software.
- **Clientes:** Solicitantes do desenvolvimento do software.
- **Gerentes:** Responsáveis pela supervisão do projeto.
- **Equipe de Desenvolvimento:** Criadores e implementadores do software.
- **Equipe de Testes:** Garantidores da qualidade e funcionamento do software.
- **Equipe de Suporte:** Suporte técnico pós-implementação.

Assinatura dos Stakeholders envolvidos

Stakeholder 1 _____
Stakeholder 2 _____
Stakeholder 3 _____

Histórico de Alterações

Data: 26/01/2025
Versão: 1.0
Descrição: Documento inicial criado para especificar os requisitos do sistema "Nexus" com foco em posts.
Autor: Igor Bonfim