UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
Prof. Seiji Isotani

Documento de Requisitos e Funcionalidades

1 Introdução

1.1 Objetivos do Documento

Este documento descreve os requisitos para o desenvolvimento do sistema RCEI, visando oferecer uma aplicação integrada, intuitiva e eficiente para pesquisadores, professores e alunos gerenciarem e visualizarem informações acadêmicas a partir da API do ORCID.

1.2 Público-Alvo

Professores, pesquisadores e alunos (graduação e pós-graduação) interessados em gestão, visualização e análise de dados acadêmicos e científicos.

1.3 Justificativa

A atual plataforma ORCID apresenta limitações em usabilidade, gerenciamento e análise de dados, o que torna a experiência dos usuários complexa e ineficiente. O RCEI propõe suprir essas lacunas com uma aplicação amigável e funcional.

2 Escopo do Produto

2.1 Funcionalidades para Pesquisadores

- Visualização de estatísticas (publicações, citações, índice H).
- Gerenciamento visual e interativo de publicações (adicionar, editar, remover).
- Análise gráfica da rede de colaboração acadêmica.
- Alertas e notificações em tempo real.
- Gerenciamento de projetos de pesquisa.
- Visualização cronológica do histórico acadêmico.

2.2 Funcionalidades para Usuários Comuns

- Busca personalizada por pesquisadores, palavras-chave, áreas.
- Visualização de perfis e detalhes completos de projetos.
- Sistema de avaliação e comentários sobre publicações e projetos.

2.3 Limitações e Exclusões

- Aplicativo móvel e integração com ERP serão tratados em versões futuras.

3 Descrição Geral do Produto

3.1 Arquitetura do Sistema

- Arquitetura em camadas: apresentação (front-end), lógica de negócio (back-end) e persistência (banco de dados).
- Comunicação RESTful via API com ORCID e outros serviços externos.

3.2 Tecnologias

- Front-end: React.js para interface Web, com design responsivo.
- Back-end: Node.js com Express para gerenciamento da lógica e API interna.
- Banco de dados: PostgreSQL para armazenamento de dados persistentes.
- Controle de versão: GitHub com fluxo Git Flow.
- Ferramentas de design: Figma para prototipação e UX/UI.

3.3 Interfaces de Usuário

Tela Principal

- Dashboard com resumo das publicações, alertas, e atalhos para funções principais.

Tela de Estatísticas

- Gráficos interativos exibindo métricas como número de publicações, citações e índice H.

Tela de Gerenciamento de Publicações

- Lista de publicações com opções de adicionar, editar e remover. Formulário detalhado para cadastro.
-

Tela de Rede de Colaboração

- Visualização gráfica da rede de autores e colaboradores, com filtros e zoom.

Tela de Projetos de Pesquisa

- Gestão dos projetos, com campos para status, prazos, participantes e progresso.

Tela de Busca e Perfil

- Busca personalizada com filtros avançados e visualização detalhada do perfil do pesquisador e projetos.

Tela de Avaliações e Comentários

- Interface para que usuários comuns deixem avaliações e comentários em publicações e projetos.

4 Requisitos Funcionais

4.1 Visualização de Estatísticas

- RF001: O sistema deve mostrar o número total de publicações, citações e índice H do pesquisador.

- Critério de Aceitação: Estatísticas exibidas claramente e atualizadas em tempo real.

4.2 Gerenciamento de Publicações

- RF002: Permitir a inclusão, edição e remoção de publicações com todos os metadados.

- Critério de Aceitação: Adicionar uma publicação em menos de 5 minutos.

4.3 Análise da Rede de Colaboração

- RF003: Exibir graficamente os colaboradores frequentes e a rede de parcerias.

- Critério de Aceitação: Rede clara, com identificação dos principais colaboradores.

4.4 Alertas e Notificações

- RF004: Enviar notificações sobre novas citações e publicações dos colaboradores em tempo real.

4.5 Gerenciamento de Projetos

- RF005: Permitir criar, editar e acompanhar o progresso de projetos de pesquisa.
- Critério de Aceitação: Criar um projeto completo em até 10 minutos.

4.6 Busca Personalizada

- RF006: Buscar pesquisadores por nome, palavra-chave e área, retornando resultados em até 2 segundos.

4.7 Sistema de Avaliações e Comentários

- RF007: Usuários comuns podem avaliar e comentar publicações e projetos com interface intuitiva.

5 Requisitos Não Funcionais

5.1 Desempenho

- Tempo máximo de resposta para buscas: 2 segundos.
- Processamento de dados para gráficos: até 3 segundos.

5.2 Usabilidade

- Interface amigável, intuitiva, com design responsivo e acessível.

5.3 Segurança

- Autenticação e autorização para pesquisadores e usuários comuns.
- Criptografia de dados sensíveis.

5.4 Manutenibilidade

- Código modular e documentado para facilitar futuras manutenções.

5.5 Portabilidade

- Aplicação Web responsiva, suportada em navegadores modernos (Chrome, Firefox, Edge).

6 Plano de Testes

6.1 Tipos de Testes

- Testes unitários automatizados para backend e frontend.
- Testes de integração entre front-end e back-end.
- Testes de sistema para validar funcionalidades completas.
- Testes de usabilidade com usuários reais.
- Testes de segurança (vulnerabilidades e controle de acesso).

6.2 Critérios de Aceitação

- Todas as funcionalidades devem estar implementadas conforme requisitos funcionais.
- Passagem em todos os testes de segurança.
- Satisfação do cliente na aceitação final.

7 Gerenciamento de Riscos

Risco	Probabilidade	Impacto	Mitigação
Atualização da API do ORCID	Médio	Alto	Monitoramento constante, versionamento, testes autom.
Problemas com sistemas open source	Baixo	Alto	Avaliação contínua, backups, segurança reforçada
Falta de comunicação na equipe	Médio	Médio	Reuniões diárias, comunicação clara via ferramentas
Falhas de segurança	Baixo	Alto	Testes de segurança regulares e hardening
Mudanças nos requisitos	Médio	Médio	Metodologia ágil para adaptação rápida

---

8 Gerência de Configuração

- Uso do GitHub com Git Flow para versionamento e controle.
- Padronização de nomenclaturas para branches, commits e arquivos.
- Gerenciamento de dependências via npm (frontend/backend).

9 Plano de Manutenção

- Registro de problemas e sugestões em ferramenta de gerenciamento (ex: Trello).
- Análise e priorização para inclusão em sprints futuros.
- Testes e implantação controlada das correções.
- Monitoramento pós-implantação.