Documento de Especificação de Requisitos de Software (ERS)

Projeto: SemNome
Versão: 1.0
Data: 10/05/2025
Analista Responsável: Mateus de Araujo Almeida

---

1. Introdução

1.1 Propósito do Documento

Este documento tem como objetivo detalhar os requisitos funcionais e não funcionais do sistema "SemNome", uma plataforma inteligente de análise de conhecimento técnico para sugestão de vagas e trilhas de estudo personalizadas.

1.2 Escopo do Sistema

A plataforma permitirá que o usuário cadastre suas habilidades, receba sugestões de vagas compatíveis, visualize gaps de conhecimento e obtenha trilhas de estudo com base em fontes abertas como roadmap.sh e Universidade-Livre.

---

2. Requisitos Funcionais (RF)

RF01 - Cadastro de Usuário

Permitir o cadastro de usuário com nome e lista de habilidades técnicas.

RF02 - Listagem de Usuários

Exibir todos os usuários cadastrados no sistema.

RF03 - Matching com Vagas

Calcular a compatibilidade entre as habilidades do usuário e os requisitos das vagas (compatibilidade direta e semântica via modelo BERT).

RF04 - Armazenamento de Matches

Armazenar os matches (compatíveis e incompatíveis) no banco de dados com seus respectivos scores.

RF05 - Análise de Habilidades

Comparar as habilidades do usuário com os requisitos da vaga e retornar:

- Habilidades em comum (match)
- Gaps (requisitos não atendidos)

- Sugestão de estudo por tecnologia

RF06 - Geração de Trilha Personalizada

Gerar uma trilha de estudo baseada nos gaps identificados, consultando fontes roadmap.sh e Universidade-Livre.

RF07 - Armazenamento de Trilhas

Armazenar trilhas geradas por usuário, evitando recomputação desnecessária.

RF08 - Busca por Vagas em APIs Externas

Fazer coleta de vagas em tempo real nas APIs do GitHub, Reed e Adzuna, quando disponível.

RF09 - Consulta de Trilhas Salvas

Exibir todas as trilhas já salvas para um determinado usuário.

RF10 - Geração de Resumo da Trilha

Utilizar IA (OpenAI) para gerar um resumo textual motivacional e estruturado da trilha sugerida.

RF11 - Consulta de Recursos por Tecnologia

Listar links e materiais sugeridos para cada tecnologia registrada.

---

3. Requisitos Não Funcionais (RNF)

RNF01 - Desempenho

Tempo de resposta inferior a 3 segundos para a maioria das requisições.

RNF02 - Persistência

Uso de banco SQLite para armazenamento local leve, com possibilidade de migração futura.

RNF03 - Responsividade

Interface responsiva para desktop e dispositivos móveis.

RNF04 - Segurança

Uso de variáveis de ambiente (.env) para proteger chaves de API.

RNF05 - Modularidade

Código separado em camadas (modelos, rotas, serviços, IA).

RNF06 - Versionamento

Todo o código do projeto deve estar versionado no GitHub.

---

4. Regras de Negócio

- RN01: Uma vaga é considerada "compatível" se tiver índice de similaridade semântica > 0.6 ou compatibilidade direta ≥ 70%
- RN02: O usuário pode cadastrar entre 1 e 50 habilidades
- RN03: Não será gerada nova trilha para tecnologia já registrada no histórico do usuário
- RN04: Cada trilha deve conter link, fonte e lista de tópicos

---

5. Casos de Uso (resumo)

UC01 - Cadastrar Perfil do Usuário
Ator: Usuário
Fluxo Principal: Preenche nome e habilidades → Envia para API → Dados salvos no banco

UC02 - Analisar Vagas
Ator: Sistema
Fluxo Principal: Consulta vagas mock/API externa → Calcula compatibilidade → Classifica → Salva matches

UC03 - Gerar Trilha de Estudo
Ator: Sistema
Fluxo Principal: Verifica gaps → Consulta roadmap.sh / Universidade-Livre → Gera trilha → Salva e exibe

UC04 - Gerar Resumo Motivacional da Trilha
Ator: Sistema
Fluxo Principal: Envia dados para OpenAI → Recebe e exibe texto motivador

---

6. Modelagem e Diagramas

6.1 Diagrama de Casos de Uso (UML - descrito)

- **Usuário**:
  - Cadastrar perfil
  - Visualizar trilhas
- **Sistema**:
  - Buscar vagas compatíveis
  - Analisar compatibilidade semântica

    - Armazenar matches e trilhas
    - Consultar conteúdos externos
    - Gerar resumo de trilha com IA

6.2 Modelo Entidade Relacionamento (MER)

Entidades principais:

- **Usuário** (id, nome, habilidades)
- **Vaga** (id, titulo, empresa, descricao, requisitos, local, url_aplicacao)
- **MatchVaga** (id, usuario_id, vaga_id, score, score_semantico, status, analise_texto, data_match)
- **Trilha** (id, usuario_id, tecnologia, origem, link, topicos, data_criacao)
- **ConteudoExterno** (id, tecnologia, origem, link, fonte_nome)

Relacionamentos:

- Usuário 1:N MatchVaga
- Vaga 1:N MatchVaga
- Usuário 1:N Trilha

6.3 Diagrama de Navegação (Wireflow)

- Tela Inicial: Cadastro/Login
- Tela Perfil: Habilidades e trilhas sugeridas
- Tela Vagas: Lista compatíveis/incompatíveis
- Tela Trilha: Tópicos e links por tecnologia
- Tela Admin (futura): Gerenciar fontes externas e vagas

---

7. Documentação Técnica

7.1 Arquitetura Geral

A aplicação é estruturada em camadas:

- **Front-end:** React + TailwindCSS (em desenvolvimento)
- **Back-end:** FastAPI (Python), responsável por toda lógica de negócio, persistência e integração com IA
- **Banco de Dados:** SQLite, com modelos ORM definidos via SQLAlchemy
- **IA/NLP:** Sentence-BERT (via SentenceTransformers) e OpenAI GPT-3.5 para análise textual e resumos motivacionais

7.2 Tecnologias Utilizadas

Camada	Tecnologia
Front-end	HTML + CSS + javascript
Back-End	FasAPI
Banco	SQLLite

IA semntico	Sentence Transformers
Geração de texto	OPENAI GPT-3.5
Dados externo	roadmap.sh, GitHub/universidade-livre

7.3 Endpoints REST (principais)

- POST /usuario - Cadastro de novo usuário
- GET /usuarios - Listagem de todos os usuários
- POST /vagas/compativel - Buscar vagas baseadas em habilidades
- GET /vagas/{usuario_id} - Buscar vagas compatíveis e incompatíveis
- GET /matches/{usuario_id} - Listar todos os matches do usuário
- POST /analise/habilidades - Analisar compatibilidade entre habilidades e requisitos
- POST /trilha - Gerar e armazenar trilha de estudo por tecnologia
- GET /trilha/{usuario_id} - Listar trilhas de um usuário
- POST /trilha/resumo - Gerar resumo textual motivacional da trilha
- POST /trilha/recursos - Retornar links e materiais relacionados a gaps
- POST /vagas/coletar - Coleta e persistência de vagas externas

7.4 Banco de Dados

- Implementado com SQLite
- Modelo relacional com integração via SQLAlchemy ORM
- Relacionamentos definidos entre Usuários, Vagas, Matches, Trilhas e ConteúdosExternos

7.5 Integrações com IA

- **IA Semântica:** modelo BERT all-MiniLM-L6-v2, carregado sob demanda, usado para gerar embeddings de textos (perfil e descrição de vaga)
- **IA Geradora:** OpenAI GPT-3.5-Turbo usada para:
  - Análise textual de compatibilidade
  - Geração de resumo da trilha de aprendizado

7.6 Controle de Versão

- GitHub com push protegido por .env

---

8. Dependências

- OpenAI (para análise de compatibilidade textual e resumo)
- SentenceTransformers (modelo BERT para similaridade semântica)
- FastAPI (API REST)
- SQLite (banco leve)
- roadmap.sh e GitHub Universidade-Livre (fontes de conteúdo)

---

9. Considerações Finais

Este documento será atualizado conforme a evolução do projeto. Todos os requisitos descritos foram extraídos da implementação atual da API e servem como base para manutenção, testes e evolução da plataforma.