MOORE

01/2026

Sistema Banco de Talentos e comparação de tecnologias para trabalho com FastAPI

MOORE

Moore Prisma Auditores e Consultores

Rua Milton José Robusti, 75
15º andar
CEP 14021-613
Ribeirão Preto - SP

T. 55 (16) 3019-7900
E. moorerp@moorebrasil.com.br

www.moorebrasil.com.br

01/2026

Ribeirão Preto-SP, 15 de janeiro de 2026.

Ao
Departamento de TI - Auditoria
Ribeirão Preto-SP

Prezados Senhores:

Encaminhamos-lhe o "Sistema Banco de Talentos e comparação de tecnologias para trabalho com FastAPI" elaborado com o objetivo de apresentar uma solução completa para gestão de candidaturas e processos seletivos, bem como demonstrar, de forma comparativa, diferentes abordagens tecnológicas aplicáveis ao desenvolvimento de sistemas web modernos. O material busca evidenciar as decisões técnicas adotadas, os fluxos funcionais implementados e a análise comparativa entre frameworks amplamente utilizados no mercado, contextualizando o projeto às necessidades e boas práticas aplicáveis à **Moore Prisma**.

Atenciosamente,
Pedro Camargo

MOORE

Sumário

1. Introdução

Este relatório tem como objetivo apresentar uma análise técnica e conceitual do sistema Banco de Talentos, desenvolvido como uma solução completa para gestão de candidaturas e processos seletivos. Ao longo do documento, são abordadas decisões de arquitetura, fluxos funcionais implementados e comparações entre diferentes tecnologias e frameworks amplamente utilizados no desenvolvimento de aplicações web modernas.

A proposta do relatório é contextualizar o projeto dentro de um cenário corporativo, demonstrando como determinadas funcionalidades foram concebidas e implementadas, bem como explorar abordagens alternativas utilizando outras tecnologias relevantes do mercado. Dessa forma, busca-se evidenciar não apenas o funcionamento do sistema, mas também os critérios técnicos, vantagens e limitações associados a cada escolha tecnológica.

Adicionalmente, o relatório contempla uma visão evolutiva do projeto, incluindo sugestões de melhorias para o frontend e considerações sobre escalabilidade, experiência do usuário, desempenho e manutenção. O conteúdo apresentado tem como foco fornecer uma visão clara, estruturada e comparativa, servindo tanto como documentação técnica quanto como material de apoio para avaliação, discussão e futuras evoluções do sistema.

2. O sistema Banco de Talentos e suas funcionalidades

O sistema Banco de Talentos consiste em uma plataforma web desenvolvida para gerenciar o cadastro, acompanhamento e avaliação de candidatos interessados em oportunidades profissionais dentro da organização. A aplicação permite que candidatos realizem o envio de seus dados, currículo e foto profissional de forma simples e intuitiva, além de acompanhar o status de sua candidatura ao longo do processo seletivo.

Do ponto de vista administrativo, o sistema disponibiliza um painel exclusivo para gestores, no qual é possível visualizar, filtrar e gerenciar candidaturas, registrar anotações internas, conduzir entrevistas estruturadas com questionários comuns e específicos por área, tomar decisões finais e analisar dados por meio de gráficos interativos. O projeto foi concebido com foco em usabilidade, organização do fluxo seletivo e clareza das informações, proporcionando uma experiência consistente tanto para candidatos quanto para gestores.

2.1 Funcionalidades

2.1.1 Candidatos

- Cadastro de candidatos com dados pessoais, formação, área pretendida e upload de currículo/foto;
- Consulta pública do status da candidatura via email;
- Fluxo simples e intuitivo para participação no Banco de Talentos.

2.1.2 Gestão de candidaturas

- Painel administrativo para visualização e gestão de todas as candidaturas;
- Filtros por estado, formação e área;
- Atualização de status da candidatura em tempo real;
- Exclusão de candidaturas com confirmação via modal;
- Sistema de anotações internas por candidato;
- Paginação de resultados para melhor desempenho e usabilidade.

2.1.3 Visualização de dados

- Gráficos dinâmicos (barras e pizza) para análise de candidatos por: Área, Formação, Estado;
- Alternância de tipo de gráfico diretamente na interface;
- Layout responsivo e consistente com o painel administrativo.

2.1.4 Entrevistas

- Lista dedicada de candidatos selecionados para entrevista;
- Fluxo completo de entrevista com: questionário comum e específico por área;
- Avaliação com notas e observações;
- Decisão final da entrevista: Aprovado, Reprovado, Não compareceu, Em andamento (decidir depois);
- Possibilidade de rever entrevistas já realizadas;
- Exclusão de entrevistas com confirmação;
- Feedback visual de status (“Já foi entrevistado”).

2.1.5 Páginas institucionais

- Página de carreiras com descrição detalhada de cada cargo;
- Navegação por tabs entre carreiras;
- CTA para cadastro no Banco de Talentos;
- Home page com seção hero e vídeo institucional incorporado.

2.1.6 UI / UX

- Design consistente entre páginas públicas e administrativas;
- Cards reutilizáveis, espaçamentos padronizados e tipografia uniforme;
- Modais de confirmação para ações sensíveis;
- Mensagens de sucesso e feedback visual para ações do gestor.

2.1.7 Segurança e controle de acesso

- Área administrativa protegida para gestores;
- Separação clara entre fluxo público (candidato) e interno (gestor).

MOORE

3. Propostas para melhorias no frontend

3.1 Design system e consistência visual

- Consolidar e documentar o design system utilizado (cores, tipografia, espaçamentos, botões, badges, modais, cards);
- Criar tokens visuais (CSS variables) para cores, tipografia e espaçamentos;
- Padronizar estados de componentes (hover, focus, active, disabled);
- Garantir consistência visual entre páginas públicas e administrativas.

3.2 Header e navegação

- Aplicar transparência no header na home page;
- Tornar o header “sticky” com comportamento inteligente:
    - Esconder ao rolar para baixo;
    - Reaparecer ao rolar para cima, com animação suave.
- Melhorar hierarquia visual do header em telas grandes e pequenas.

3.3 Formulários & UX

- Adicionar autocomplete para campos de localização:
    - Estado;
    - Cidade (a partir de 3 ou 4 caracteres digitados).
- Indicar campos obrigatórios visualmente;
- Melhorar feedback de carregamento em submissões (loading spinner no botão);
- Bloquear múltiplos envios acidentais do formulário.

3.4 Feedback visual e estados da interface

- Padronizar mensagens de sucesso, erro e aviso;
- Adicionar animações sutis em:
    - Abertura de modais;
    - Troca de tabs.
- Melhorar estados vazios (empty states) com mensagens e ilustrações.

3.5 Tabelas, listagens e dados

- Permitir ordenação de colunas (ex: por nome, data, status);
- Adicionar busca textual em tabelas (ex: buscar candidato por nome/email);
- Melhorar responsividade das tabelas em telas menores;
- Adicionar tooltips para ações e ícones.

3.6 Gráficos

- Permitir download dos gráficos (PNG ou PDF);
- Adicionar animação de entrada nos gráficos.

3.7 Página de entrevistas

- Exibir currículo e foto do candidato diretamente na página de entrevistas;

- Criar visualização rápida (preview) sem sair da lista;
- Destacar entrevistas pendentes vs finalizadas.

3.8 Acessibilidade

- Garantir contraste adequado entre texto e fundo;
- Preparar frontend para múltiplos idiomas;
- Externalizar textos estáticos para facilitar tradução.

3.9 Performance e qualidade

- Otimizar carregamento de imagens (lazy loading);
- Reduzir uso de CSS duplicado;
- Agrupar e minificar arquivos estáticos em produção.

3.10 Footer

- Expandir footer com:
    - Links institucionais (About, Services, Contact);
    - Links para redes sociais (LinkedIn, YouTube, Instagram, X).
- Usar ícones vetoriais (SVG) para redes sociais.

As melhorias propostas para o frontend do Banco de Talentos visam elevar o nível de maturidade do projeto, aproximando-o de padrões utilizados em aplicações corporativas de médio e grande porte. A consolidação de um Design System, aliada à padronização visual e comportamental dos componentes, contribui diretamente para a consistência da interface, facilidade de manutenção e escalabilidade futura do produto.

Além disso, os aprimoramentos em navegação, formulários, feedback visual e visualização de dados fortalecem a experiência do usuário, tornando as interações mais claras, eficientes e intuitivas tanto para candidatos quanto para gestores. As iniciativas relacionadas à acessibilidade, performance e organização do código frontend demonstram preocupação com qualidade técnica, inclusão e sustentabilidade do sistema a longo prazo.

Em conjunto, essas evoluções não apenas refinam a interface atual, mas também preparam o projeto para futuras expansões funcionais, integrações e adaptações, reforçando o Banco de Talentos como uma solução robusta, profissional e alinhada às boas práticas modernas de desenvolvimento frontend.

MOORE

4. Comparação de tecnologias

4.1 Situação 1: consulta de status de candidatura

No projeto desenvolvido, existe uma funcionalidade pública que permite ao candidato consultar o status da sua candidatura informando apenas o email utilizado no cadastro.

Requisitos da funcionalidade:

- Interface seguindo o design system com campo de preenchimento para email;
- Busca no banco de dados pelo email informado;
- Retorno dos dados básicos do candidato:
  - Nome;
  - Email;
  - Localização (cidade e estado);
  - Área pretendida;
  - Status atual da candidatura.
- Funcionalidade pública (sem autenticação);
- Resposta indicando erro para email não encontrado.

4.1.1 Como essa funcionalidade poderia ser implementada usando FastAPI

Arquitetura proposta:

- FastAPI como API REST;
- Endpoint GET /status?email=...;
- Validação automática do email usando Pydantic;
- ORM como SQLAlchemy ou SQLModel.

Fluxo:

- Cliente envia requisição com o email;
- FastAPI valida o formato do email;
- Consulta ao banco de dados;
- Retorna JSON com os dados ou erro.

Exemplo simplificado:

```
@app.get("/status")
def consultar_status(email: EmailStr):
    candidato = session.query(Candidato).filter_by(email=email).first()

    if not candidato:
        raise HTTPException(status_code=404,  detail="Candidato não
encontrado")

    return {
        "nome": candidato.nome,
        "area": candidato.area,
```

```
        "status": candidato.status
    }
```

4.2 Situação 2: atualização de status de candidatura

No projeto, cada candidatura possui um status que representa sua etapa no processo seletivo. Os status podem ser:

- Cadastro submetido;
- Selecionado para entrevista;
- Aprovado;
- Reprovado;
- Não compareceu.

O gestor pode atualizar esse status diretamente pelo Painel do Gestor, por meio de um seletor (dropdown), sem necessidade de navegar para outra página. Essa atualização impacta:

- A visualização do candidato no painel administrativo;
- A lista de entrevistas;
- A consulta pública de status feita pelo próprio candidato via email.

4.2.1 Como essa funcionalidade poderia ser implementada usando FastAPI

Arquitetura proposta:

- FastAPI como API REST;
- SQLAlchemy para ORM;
- Pydantic para validação de payloads;
- Frontend separado (HTML + JS).

Fluxo para um único candidato:

- Recebe o novo status no corpo da requisição;
- Valida se o status é permitido;
- Atualiza o registro no banco;
- Retorna o candidato atualizado.

4.3 Situação 3: fluxo de entrevista e decisão final

No projeto, após um candidato ser selecionado para entrevista, o gestor acessa uma página dedicada onde pode:

- Visualizar os dados do candidato;
- Responder um questionário dividido em:
    - Parte comum;
    - Parte específica (de acordo com a área).
- Atribuir notas e observações;
- Tomar uma decisão final:
    - Aprovado;

    - Reprovado;
    - Não compareceu;
    - Decidir depois.

Essa decisão impacta diretamente:

- O status do candidato;
- A visualização nas listas administrativas;
- O feedback ao próprio candidato via consulta pública.

4.3.1 Como essa funcionalidade poderia ser implementada usando FastAPI

Arquitetura proposta:

- FastAPI como backend REST;
- SQLAlchemy para ORM;
- Pydantic para validação de dados;
- Frontend separado (HTML + JS).

Fluxo:

- GET /interviews/{candidate_id}
    - Retorna dados do candidato;
    - Retorna perguntas comuns e específicas;
    - Retorna avaliação já salva (se existir).
- POST /interviews/{candidate_id}
    - Recebe respostas do questionário;
    - Recebe decisão final;
    - Salva respostas em estrutura JSON;
    - Atualiza campo entrevistado = true;
    - Atualiza status do candidato conforme decisão.

4.4 Situação 4: visualização de dados com gráficos

O projeto possui um Painel do Gestor que apresenta gráficos para análise das candidaturas cadastradas. Esses gráficos permitem visualizar:

- Quantidade de candidatos por Área;
- Quantidade de candidatos por Formação;
- Quantidade de candidatos por Estado.

Características importantes da implementação atual:

- Gráficos de barras e pizza;
- Alternância do tipo de gráfico diretamente na interface;
- Dados atualizados dinamicamente com base nos filtros aplicados;
- Objetivo de apoiar decisões do gestor com visão analítica rápida.

4.4.1 Como essa funcionalidade poderia ser implementada usando FastAPI

Arquitetura proposta:

- FastAPI como API;
- SQLAlchemy para agregações no banco de dados;
- Chart.js ou biblioteca similar no frontend;
- API responsável apenas por fornecer os dados agregados.

Fluxo:

- GET /dashboard/charts
  - Retorna dados agregados (labels e values);
  - Exemplo:

```
{
      "labels": ["SP", "RJ", "MG"],
      "values": [12, 8, 5]
}
```

- Frontend
  - Consome a API;
  - Renderiza gráficos dinamicamente;
  - Alterna entre tipos (barra e pizza) sem nova requisição.

4.5 Comparação tecnológica

A escolha da tecnologia de backend e frontend exerce impacto direto na arquitetura, escalabilidade, produtividade da equipe e manutenção de um sistema. No contexto deste projeto, que envolve cadastro de candidatos, gestão administrativa, entrevistas, atualização de status e visualização de dados, é relevante analisar como diferentes tecnologias se posicionam frente às necessidades funcionais e não funcionais da aplicação.

4.5.1 FastAPI vs Django

Ao comparar FastAPI e Django, observa-se que FastAPI é um framework moderno, orientado à construção de APIs de alto desempenho, com forte apoio à tipagem estática e validação de dados por meio do Pydantic. Sua arquitetura baseada em ASGI proporciona excelente escalabilidade e desempenho, tornando-o especialmente adequado para sistemas desacoplados, microserviços e aplicações com múltiplos consumidores. No entanto, FastAPI exige maior definição arquitetural desde o início e não oferece, de forma nativa, recursos prontos como painel administrativo ou sistema de templates, o que pode aumentar o tempo de desenvolvimento em aplicações completas. Por outro lado, Django apresenta-se como um framework full-stack consolidado, com ORM robusto, sistema de autenticação integrado, administração automática e suporte nativo a templates. Essas características tornam o Django especialmente produtivo para aplicações administrativas, como o painel do gestor desenvolvido neste projeto, permitindo rápida implementação de fluxos complexos com consistência e segurança. Em contrapartida, Django pode não ser a melhor escolha para aplicações exclusivamente orientadas a APIs de alta concorrência, onde FastAPI se destaca.

4.5.2 FastAPI vs React

A comparação entre FastAPI e React não representa um cenário de concorrência direta, mas sim de complementaridade. FastAPI atua exclusivamente no backend, sendo responsável pelas regras de negócio, persistência de dados e validações, enquanto React é uma biblioteca voltada à construção de interfaces de usuário altamente interativas no frontend. Em um cenário arquitetural alternativo, este projeto poderia ser estruturado com FastAPI fornecendo uma API REST e React consumindo esses dados no frontend, resultando em maior flexibilidade visual e desacoplamento entre camadas. Entretanto, a escolha por Django com templates server-side simplificou a arquitetura e reduziu a complexidade, sendo uma decisão adequada para o escopo e objetivos do projeto.

4.5.3 FastAPI vs Next.js

Ao analisar FastAPI em relação ao Next.js, percebe-se que Next.js ocupa uma posição híbrida, oferecendo tanto recursos avançados de frontend quanto funcionalidades de backend por meio de API Routes. Next.js destaca-se principalmente em aplicações onde SEO, renderização no servidor e performance de páginas públicas são requisitos críticos, como em páginas institucionais ou de carreiras. Embora seja possível implementar APIs completas com Next.js, FastAPI oferece maior robustez, organização e escalabilidade para sistemas cujo backend é o núcleo da aplicação. No contexto deste projeto, Django cumpre um papel semelhante ao Next.js ao integrar frontend e backend, enquanto FastAPI seria mais indicado em uma arquitetura totalmente desacoplada.

4.5.4 FastAPI vs Laravel

Por fim, ao comparar FastAPI e Laravel, observa-se que ambos oferecem soluções maduras para desenvolvimento backend, porém com filosofias distintas. Laravel é um framework full-stack altamente produtivo, com sintaxe expressiva e ecossistema vasto, sendo amplamente utilizado para sistemas administrativos e aplicações baseadas em CRUD. Assim como Django, Laravel seria uma alternativa viável para este projeto. FastAPI, por sua vez, apresenta vantagens significativas em cenários que exigem alta performance, integração com serviços externos e arquitetura orientada a APIs. A escolha entre Laravel e FastAPI tende a ser influenciada principalmente pela stack já dominada pela equipe e pelos requisitos de escalabilidade do sistema.

4.5.5 Conclusão

De forma geral, conclui-se que FastAPI é mais indicado para projetos orientados a APIs modernas, com necessidade de desempenho, tipagem forte e integração com múltiplos clientes, enquanto Django e Laravel são escolhas mais adequadas para aplicações full-stack, especialmente quando há necessidade de painéis administrativos, desenvolvimento rápido e forte acoplamento entre frontend e backend. React e Next.js, por sua vez, destacam-se no desenvolvimento de interfaces ricas e modernas, sendo particularmente relevantes quando a experiência do usuário e a separação de responsabilidades entre frontend e backend são prioridades estratégicas.

REDE GLOBAL MOORE

Uma rede mundial que atua há mais de 100 anos com personalidade local.

Isso garante proximidade ao cliente e profundo conhecimento da sua região de atuação, respeitando culturas e legislações.

CONTATO

Moore Prisma Auditores e Consultores

Rua Milton José Robusti, 75
15° Andar
CEP 14021-613
Ribeirão Preto - SP - Brasil

T 55 (16) 3019 7900
E moorerp@moorebrasil.com.br

MOORE

www.moorebrasil.com.br