SmartHire — Relatório de Status (60h)

Objetivo do Projeto

Criar um processo seletivo mais automatizado, reduzindo tempo e viés na triagem de candidatos por meio de:

- Análise de currículos e entrevistas com IA
- Etapas configuráveis por vaga com thresholds e pesos
- Pontuação objetiva e painel de resultados

Visão Geral da Arquitetura

- Frontend/BFF: Next.js (App Router) + TypeScript hospedado no Vercel
- Banco/Storage/Auth: Supabase (Postgres + RLS + Storage)
- Serviço de IA: FastAPI (Python) em AWS ECS Fargate, integrações com Deepgram (ASR) e modelo compatível com OpenAI
- Filas/Jobs: Estratégia assíncrona via SQS descrita (pós-MVP em partes), com polling em `/api/ai/runs/[id]`
- Observabilidade/Segurança: Sentry/OTel (planejado), RLS por `company_id`, RBAC básico por usuário

Referências: `docs/DECISOES.md`, `docs/SCHEMA.md`, `docs/API_BFF.md`, `docs/API_IA.md`.

Requisitos Funcionais

- RF1: Gerenciar vagas (criar, listar, buscar, excluir)
- RF2: Cadastrar e listar candidatos, com paginação e busca
- RF3: Atribuir candidatos a vagas (applications)
- RF4: Configurar etapas do processo por vaga com threshold e peso
- RF5: Definir templates de prompt (padrão e por etapa)
- RF6: Upload de currículo (PDF), áudio e transcrição com URLs assinadas (Supabase Storage)
- RF7: Disparar análise de IA por etapa (transcrição/opcional, RAG/extração e scoring) e acompanhar status
- RF8: Visualizar análises, pontos fortes/fracos, requisitos atendidos e pontuação
- RF9: Painel com notas por candidato e média ponderada por etapas
- RF10: Configurar credenciais e parâmetros da IA (modelo, temperatura, tokens)

Requisitos Não Funcionais

- RNF1: Autenticação via Supabase Auth; autorização por `company_id` (RLS) e ownership
- RNF2: Escalabilidade horizontal do serviço de IA (ECS Fargate); fallback assíncrono (SQS) planejado
- RNF3: Observabilidade com logs estruturados, tracing e métricas (planejado)
- RNF4: LGPD: classificação de PII, retenção e auditoria (planejado)
- RNF5: UX clara, feedbacks (toasts), estados de loading e erros amigáveis

Banco de Dados (alto nível)

Principais tabelas (ver `docs/SCHEMA.md` e migrações em `db/migrations/`):

- `companies`, `users`, `jobs`, `candidates`, `applications`, `job_stages`
- `application_stages`, `stage_ai_runs`, `prompt_templates`, `stage_prompt_templates`
- `scores`, `score_overrides`, `documents`

RLS por `company_id` e uso de `created_by` em várias entidades. Armazenamento de arquivos em buckets: `resumes`, `audios`, `transcripts`.

Páginas Implementadas (web/src/app)

- Home (`/`): landing com CTA para `jobs` e `settings/ai`.
- Login (`/login`): email/senha e link mágico (Supabase Auth) com sync de sessão em `/api/auth/session`.
- Vagas (`/jobs`):
  - Criar vaga; listar/buscar/paginar; excluir vaga
  - Link para configurar etapas em `/jobs/[id]/stages`
- Vaga > Etapas (`/jobs/[id]/stages`):
  - CRUD de etapas (nome, descrição, threshold, peso)
  - Seleção de template por etapa; fallback para template padrão do usuário
  - Atribuir candidatos à vaga; listar candidates atribuídos
  - Upload de documentos (currículo/áudio/transcrição) e envio para avaliação da IA
  - Visualização da análise mais recente por etapa, com painel dedicado
  - Painel de notas por candidato (média ponderada pelos pesos das etapas)
- Candidatos (`/candidates`): criar/listar/buscar com paginação
- Configurações (`/settings`): cards de navegação
- Configurações da IA (`/settings/ai`): chave, modelo, temperatura, max tokens, teste de conexão
- Templates de Prompt (`/settings/prompts`): CRUD, marcar como padrão
- Análise por run (`/analysis/[runId]`): página de status e resultado (fluxo complementar)

API BFF (Next.js App Router)

- Jobs
  - `GET /api/jobs` — lista com busca e paginação
  - `POST /api/jobs` — cria vaga
  - `GET /api/jobs/[id]` — detalhes/sanity e ownership
  - `DELETE /api/jobs/[id]` — remove vaga e dependências diretas
  - `GET /api/jobs/[id]/stages` — lista etapas
  - `POST /api/jobs/[id]/stages` — cria etapa
  - `GET /api/jobs/[id]/applications` — lista applications com dados do candidato
  - `POST /api/jobs/[id]/applications` — atribui candidato
- Applications
  - `DELETE /api/applications/[id]` — remove atribuição
- Candidates
  - `GET /api/candidates` — lista/busca/paginação
  - `POST /api/candidates` — cria candidato
- Uploads (Supabase Storage, URLs assinadas)
  - `POST /api/uploads/resume` , `audio` , `transcript`
- Stages / IA
  - `POST /api/stages/[stageId]/evaluate` — dispara análise completa no serviço de IA; cria `stage_ai_runs`
  - `GET /api/stages/[stageId]/analysis` — busca análise mais recente para application/etapa
  - `GET /api/stages/[stageId]/analysis/latest` — busca análise mais recente da etapa (qualquer application)
  - `PUT/DELETE /api/stages/[stageId]` — atualiza/remove etapa
  - `GET/POST /api/stages/[stageId]/prompt-template` — obtém/define template da etapa
- Prompt Templates
  - `GET /api/prompt-templates` , `POST /api/prompt-templates`
  - `PUT/DELETE /api/prompt-templates/[id]` (implementação correlata no projeto)

- Runs de IA
  - `GET /api/ai/runs/[id]` — proxy de status ao serviço de IA

Autorização: `requireUser()` em praticamente todas as rotas; checagens por `company_id` e `created_by` antes de mutações.

Serviço de IA (contratos)

- Endpoints de referência em `docs/API_IA.md` (transcribe, rag, score, runs). No fluxo de avaliação usado pelo frontend, há endpoint consolidado `/v1/evaluate` (serviço) acionado por `/api/stages/[stageId]/evaluate`.
- Persistência do run e resultados em `stage_ai_runs` via BFF.

Fluxos Principais

1. Criar vaga → configurar etapas (threshold/peso/descrição) → criar/atribuir candidatos → enviar documentos → disparar avaliação → acompanhar status do run → visualizar análise/score.
2. Definir templates de prompt (um padrão por usuário e opcionais por etapa).
3. Ajustar configurações de IA (chave OpenAI, modelo, temperatura, tokens) e testar conexão.

O que já foi entregue (60h)

- UI/UX das páginas principais (Home, Jobs, Candidates, Settings, Settings/AI, Settings/Prompts, Stages, Analysis)
- BFF com autenticação/authorization, CRUDs essenciais e integração com Supabase Storage
- Integração frontend → BFF → Serviço de IA (evaluate) com polling do run
- Persistência de runs e exibição de análises e scores; painel de candidatos com média ponderada
- Documentação de arquitetura, schema e contratos (docs/*)

Pontos pendentes/Próximos passos

- Webhooks/SSE para atualização em tempo real (reduzir polling)
- Integrações de calendário (entrevistas) e trilhas de auditoria completas
- Exports/Relatórios em PDF/CSV direto do app (há print/export básico na página de análise) +- Polimento de RLS/Policies e LGPD (retenção, consentimento)
- Observabilidade (Sentry/OTel) e custos por run/token

Métricas e Tempo

- Esforço contabilizado até o momento: 60 horas
- Ganhos esperados: redução de ~80% no tempo de triagem (estimado, conforme copy da UI)

— Relatório gerado automaticamente a partir do código e docs do repositório.