**Manual de Implantação e Arquitetura da Plataforma IAprovas**

**Versão 1.1 - 19 de Junho de 2025**

Este documento descreve o passo a passo completo para configurar, desenvolver e implantar a plataforma IAprovas a partir do zero, incluindo uma descrição da função de cada arquivo no projeto.

**Parte 1: Configuração do Ambiente de Desenvolvimento Local**

Estas são as ações que você executa no seu computador para poder trabalhar no projeto.

**1.1. Procedimento de Início de Dia:** Sempre que for começar a trabalhar, abra o VS Code e o terminal integrado e siga estes passos:

* Navegue até a pasta do projeto:

Bash

cd caminho/para/seu/projeto/plataforma\_concursos-8a639e1bed3825304733bd4ec7746cc5cfe6fa11

* Ative o ambiente virtual do Python:

Bash

source venv/bin/activate

* Inicie o servidor de backend:

Bash

python3 app.py

* Inicie o servidor do frontend: No VS Code, clique com o botão direito no arquivo index.html e selecione "Open with Live Server".

**1.2. Procedimento de Fim de Dia (Salvar no GitHub):** Ao final de cada sessão de trabalho, para salvar seu progresso de forma segura:

* No terminal, pare o servidor app.py com Ctrl + C.
* Verifique as alterações: git status
* Adicione todas as alterações: git add .
* Crie um "pacote" com uma mensagem descritiva (Commit):

Bash

git commit -m "feat: Descrição da funcionalidade feita"

(Use feat: para novas funcionalidades, fix: para correções, chore: para configurações).

* Envie para o GitHub: git push
* Desative o ambiente virtual: deactivate
* Feche o VS Code.

**Parte 2: Integração de Serviços Externos**

Configuramos três serviços principais: Firebase, Stripe e OpenAI.

**2.1. Firebase (Banco de Dados e Autenticação):**

* **Configuração no Código:** As credenciais de frontend do Firebase (apiKey, authDomain, etc.) estão no arquivo firebase-config.js. Isso é seguro, pois a proteção dos dados é feita pelas Regras de Segurança no painel do Firestore.
* **Configuração no Painel do Firebase:**
  + **Domínios Autorizados:** Para que o login e cadastro funcionem em iaprovas.com.br, fomos em Authentication > Settings > Domínios autorizados e adicionamos iaprovas.com.br e www.iaprovas.com.br.
  + **Chave de API:** Para evitar o erro 403 (Forbidden), fomos no Google Cloud Console > APIs e serviços > Credenciais, selecionamos a chave de API do nosso app web e, em "Restrições de aplicativos", adicionamos iaprovas.com.br e \*.iaprovas.com.br aos "Referenciadores HTTP".
  + **Conta de Serviço (Backend):** Para que nosso app.py pudesse acessar o Firestore, geramos uma chave de serviço em Configurações do projeto > Contas de serviço. O arquivo JSON baixado foi renomeado para serviceAccountKey.json e colocado no ambiente de produção através dos "Secret Files" da Render.

**2.2. Stripe (Pagamentos e Assinaturas):**

* **Estrutura de Produtos e Preços (Atualizado):** A estrutura foi refeita. Agora utilizamos um **único Produto na Stripe** chamado "IAprovas Assinatura". Associados a este produto, foram criados múltiplos Preços para cada plano oferecido (Básico, Intermediário, Premium Mensal, Premium Anual). Esta abordagem centraliza a gestão.
* **Portal do Cliente (Customer Portal) (Novo):** Foi implementada a funcionalidade de "Gerenciar Assinatura". Para isso, o webhook checkout.session.completed no app.py foi atualizado para salvar o stripeCustomerId de cada assinante no Firestore. Uma nova rota, /create-portal-session, foi adicionada ao app.py para gerar um link de acesso seguro para o portal do cliente.
* **Stripe CLI e Webhooks:**
  + Instalamos a Stripe CLI para criar um "túnel" para nosso ambiente de desenvolvimento.
  + Com o comando stripe listen --forward-to localhost:5000/stripe-webhook, obtivemos um segredo de webhook de teste (whsec\_...), que foi colocado no nosso arquivo .env local.
  + Para produção, criamos um Endpoint de Webhook no painel da Stripe, apontando para https://iaprovas-backend.onrender.com/stripe-webhook. O segredo de produção gerado foi adicionado como variável de ambiente na Render.

**2.3. OpenAI (Inteligência Artificial):**

* **Controle de Custos:** Configuramos um limite de uso mensal (budget) em Settings > Limits no painel da OpenAI para receber alertas e controlar os gastos. Os Rate Limits (limites de velocidade) padrão foram mantidos como proteção automática contra picos de uso.

**2.4. Resend (E-mails Transacionais):**

* **Configuração:** Optou-se pelo Resend devido à sua API moderna e por oferecer e-mails "white-label" em seu plano gratuito.
* **Chave de API:** Uma chave de API foi gerada no painel do Resend e configurada como variável de ambiente (RESEND\_API\_KEY) no backend.
* **Domínio Verificado:** O domínio iaprovas.com.br foi verificado no Resend para garantir a máxima entregabilidade dos e-mails.

**Parte 3: Deploy na Render (Produção)**

Colocamos a aplicação no ar usando a plataforma Render.

**3.1. Arquivos de Configuração para Deploy:** Criamos três arquivos na raiz do projeto:

* requirements.txt: Lista todas as bibliotecas Python que o backend precisa.
* build.sh: Um script que diz à Render para executar pip install -r requirements.txt.
* Procfile: Um arquivo que diz à Render como iniciar o servidor: web: gunicorn app:app.

**3.2. Deploy do Backend (Web Service):**

* Criamos um "Web Service" na Render, conectado ao nosso GitHub.
* **Configurações:** Runtime: Python 3, Build Command: bash build.sh, Start Command: gunicorn app:app.
* **Variáveis de Ambiente:** Configuramos um "Environment Group" na Render onde adicionamos todas as nossas chaves: OPENAI\_API\_KEY, STRIPE\_SECRET\_KEY (de produção), STRIPE\_WEBHOOK\_SECRET (de produção) e o conteúdo do serviceAccountKey.json como um "Secret File".

**3.3. Deploy do Frontend (Static Site):**

* Criamos um "Static Site" na Render, também conectado ao GitHub.
* **Configurações:** Publish Directory: . (ponto), para publicar todos os nossos arquivos HTML/CSS/JS.

**3.4. Configuração do Domínio:**

* **Na Render:** Em "Settings" do nosso Static Site, adicionamos iaprovas.com.br e www.iaprovas.com.br como "Custom Domains". A Render nos forneceu os valores de IP (para o registro A) e CNAME (para o www).
* **No Registro.br:** Fomos na "Configuração de Zona DNS" e adicionamos os registros A e CNAME fornecidos pela Render.

**Parte 4: Arquitetura do Projeto (O que cada arquivo faz)**

**4.1. Arquivos HTML (A Estrutura e o Conteúdo das Páginas):**

* **index.html:** Página inicial, atualizada com a nova estrutura de 5 planos (Trial, Básico, Intermediário, Premium, Anual).
* **login.html**, **cadastro.html**: Páginas de autenticação.
* **home.html**, **cronograma.html**, **exercicios.html**, **discursivas.html**, **dicas-estrategicas.html**: Páginas de funcionalidades do dashboard.
* **meu-perfil.html (Atualizado):** Estrutura redesenhada para organizar os formulários e incluir a seção de "Excluir Conta". Os links da barra lateral agora contêm ids para controle de acesso.
* **Páginas de Fluxo:** termos.html, privacidade.html, sucesso.html.
* **contratacao.html (Atualizado):** Inclui um "modal de decisão" para guiar usuários não logados entre login e cadastro.
* **cancelado.html (Atualizado):** Página redesenhada com a identidade visual da plataforma para uma melhor UX.

**4.2. Arquivos de Estilo CSS (A Aparência Visual):**

* **styles.css:** Estilo principal para páginas públicas.
* **dashboard-lovable.css (Atualizado):** Estilo para o painel do usuário. Foram adicionados novos estilos para padronizar os formulários da página de perfil e para a seção de "Excluir Conta".

**4.3. Arquivos JavaScript (A Interatividade e a Lógica):**

* **main-app.js (Atualizado):** Agora dispara um evento global userDataReady após carregar os dados do usuário para sincronizar a inicialização de outras páginas. Implementa a função controlarAcessoFuncionalidades que bloqueia/libera o acesso aos links da barra lateral com base no plano do usuário.
* **auth.js (Atualizado):** Lógica de login e cadastro aprimorada para usar o parâmetro de URL ?returnTo= para redirecionar o usuário de volta ao fluxo de compra após a autenticação.
* **api.js (Atualizado):** A API\_BASE\_URL agora é dinâmica para funcionar em ambiente local e de produção. Foram adicionadas as funções criarSessaoPortal e deletarContaUsuario.
* **perfil-page.js (Atualizado):** Lógica reestruturada para "ouvir" o evento userDataReady antes de configurar os botões e preencher os dados, corrigindo bugs de temporização. Contém toda a lógica para salvar perfil, alterar senha, gerenciar assinatura e excluir conta.
* **contratacao.js (Atualizado):** Exibe o modal de decisão e utiliza o ?returnTo= para o redirecionamento.
* **stripe-config.js (Novo):** Arquivo criado para armazenar a chave publicável da Stripe, agora incluído no controle de versão.

**4.4. Backend e Arquivos de Servidor (app.py):**

* **app.py (Atualizado):** O motor da plataforma. O webhook da Stripe agora salva o stripeCustomerId no Firestore. Adicionadas duas novas rotas: /create-portal-session e /delete-user-account para gerenciar assinaturas e exclusão de contas.

**4.5. Configuração do Projeto e Deploy:**

* **.gitignore (Atualizado):** A linha que ignorava o stripe-config.js foi removida.
* **.env**, **requirements.txt**, **build.sh**, **Procfile**: Arquivos de configuração do projeto, sem alterações recentes.