# **1) Como aplicar o projeto (do zero ao demo)**

<https://drive.google.com/drive/folders/1n5z2wR9LKpbllRgZ8XG7KUCSnVSi5iXp?usp=sharing>

## **Fase 0 — Preparação (antes do evento)**

* **Ambiente**: crie um repo (Git) e defina .devcontainer/conda com requirements.txt.
* **Dados**: use o *starter kit* que te entreguei:  
  1. python src/ingest.py --download all (baixa KOI/K2/TOI)
  2. confirme colunas-chave: disposition, period, duration, planet\_radius.
* **Baseline**:  
   3) python src/train\_baseline.py --mission all --target\_column disposition  
   4) revise AUC/F1 no terminal e rode o notebook notebooks/baseline\_exoseeker.ipynb para ver SHAP.

## **Fase 1 — MVP científico (Dia 1)**

1. **Normalização & features**
   * Padronize rótulos (CONFIRMED/CANDIDATE/FALSE\_POSITIVE).
   * Crie *features* derivadas (ex.: *odd-even depth ratio*, SNR do trânsito, razão período/duração).
2. **Modelo tabular**
   * Treine LightGBM com class\_weight="balanced".
   * Métricas iniciais (AUC macro, F1 por classe).
3. **API**
   * Suba a FastAPI (uvicorn app.main:app --reload) e valide /api/train + /api/predict.
4. **UI (web) – primeiro corte**
   * Página “Upload & Analisar”: enviar CSV, receber tabela com predições e probabilidades.
   * Painel de métricas (AUC, F1, matriz de confusão).

## **Fase 2 — IA multimodal + explicabilidade (Dia 2)**

1. **Curva de luz (sequência)**
   * Gere *janelas* centradas no trânsito (phase-fold).
   * **Modelo 1D-CNN + encoder leve** para a sequência; mantenha LightGBM para tabular.
   * **Fusão tardia** (concat) → cabeça de classificação.
2. **Adaptação entre missões (domain shift)**
   * Treine com rótulo de domínio (Kepler/K2/TESS) e *DANN* (camada adversarial) para robustez.
3. **Grad-CAM 1D**
   * Registre gradientes na última *conv*; pese filtros pelos gradientes médios e gere o *heatmap* 1D.
   * Exponha via /api/explain/{target\_id}: retorna indices, flux, cam normalizado.
4. **Rotulagem Ativa**
   * Crie fila de amostras de maior entropia para “human-in-the-loop” (Candidate/FP/Artifact).
5. **Relatórios**
   * Exportação PDF por amostra: gráfico da curva (com *highlight* do Grad-CAM) + top-features (SHAP) + decisão.

## **Fase 3 — “pronto pra demo”**

* **Casos de uso**: pegue 10–20 alvos (mix PC/FP/Confirmed).
* **Script da demo**: upload → predição → abrir *Explainer* → mostrar *heatmap* sobre o trânsito → baixar laudo PDF.
* **Story**: “como evitamos um FP clássico (odd-even)” e “como generalizamos para outra missão”.

# **2) Como apresentar para jurados (pitch + demo)**

**Tempo total: 3–4 min (adaptável para 90s e 2 min):**

1. **Gancho (15–20s)** — “Ainda perdemos horas separando planetas de artefatos. Hoje, isso fica explicável e rápido.”
2. **Problema (20s)** — Heterogeneidade inter-missões + falsos positivos.
3. **Solução (45s)** — Multimodal (curva + tabular), DANN, Grad-CAM 1D, SHAP, UI com rotulagem ativa.
4. **Demo (60–75s)** — Subir CSV, ver predição, abrir *Explainer*, *highlight* no trânsito, baixar laudo.
5. **Resultados (30–40s)** — AUC macro, F1 por classe, *leave-one-mission-out*.
6. **Impacto (20s)** — Ciência reprodutível + modo educacional + comunidade.
7. **Fecho (10–15s)** — “Open-source, pronto para ampliar com observação de follow-up.”

**Dicas de slide:**

* 1 gráfico bom > 5 bullet points.
* Mostre **1** caso Candidate e **1** caso FP (odd-even).
* Tenha a página de “métricas globais” aberta para perguntas.

# **3) Como faremos a página web (site oficial)**

## **Stack & deploy**

* **Next.js + Tailwind** (SSR rápido, SEO) ou Astro se preferir estático.
* Deploy em qualquer PaaS (Vercel/Netlify/Render).
* Conecta na API FastAPI já pronta.

## **Sitemap (mínimo)**

* **Home**: hero (“Caçando exoplanetas com IA explicável”), CTA “Testar agora”.
* **Explorar**: mini-galeria com 6–10 casos (gráficos + card do laudo).
* **Analisar**: página de upload → tabela de predições → botão “Explainer”.
* **Docs**: endpoints da API, formato de CSV, exemplos (com *curl*).
* **Equipe/Contato**: bios curtas, e-mail, link do repo.

## **Componentes principais**

* **Uploader** (CSV/ID de alvo): valida schema e mostra preview.
* **Tabela de resultados**: Target, Missão, Predição, Prob., *badge* de incerteza; filtro por classe/ missão.
* **Explainer Modal/Drawer**:  
  + Gráfico interativo (Plotly): série flux + *heatmap* Grad-CAM 1D sobreposto.
  + Top-features SHAP (bar chart).
  + Botão **Exportar PDF**.
* **Métricas globais**: AUC, F1, matriz de confusão; *tabs* por missão.
* **Modo Edu**: passo-a-passo do método do trânsito + mini-quiz.

## **Integração com API (exemplo de contrato)**

* POST /api/predict → { predictions: [...], proba: [...], ids: [...] }
* GET /api/explain/{id} → { time: [...], flux: [...], cam: [...], shap: [{feature, value, impact}, ...] }
* POST /api/label → grava rótulo humano e notas (para *active learning*).

## **Design (rápido e bonito)**

* Layout em **cards** com sombras suaves (rounded-2xl), tipografia Inter, escala de cinzas + acentos em roxo/azul.
* Ícones (lucide) para estados: Confirmed (★), Candidate (◐), FP (⚠).

# **4) Como faremos o aplicativo móvel**

## **Objetivo do app**

* Visualizar e “marcar” candidatos em mobilidade (outreach + ciência cidadã).
* Subir *amostras pequenas* (ou IDs) para inferência e ver explicações.

## **Stack**

* **React Native (Expo)** para agilidade e build iOS/Android.
* Libs: react-native-svg + victory-native (gráficos), expo-file-system (CSV local), expo-router (navegação).

## **Telas**

1. **Home** — CTA “Analisar” + últimos resultados.
2. **Upload/Selecionar** — escolher CSV do dispositivo ou digitar ID (TOI/KOI).
3. **Resultados** — lista com Predição, Prob., Missão, “ver detalhes”.
4. **Explainer** — gráfico da curva + *overlay* do Grad-CAM 1D, lista de top-features (chips).
5. **Rotulagem** — Candidate/FP/Artifact + notas → envia para /api/label.

## **Fluxo técnico**

* O app chama o mesmo backend (/api/predict, /api/explain, /api/label).
* Para *offline demo*, inclua 2 arquivos embarcados (os CSVs simulados que te dei).
* Cache leve (AsyncStorage) de últimos 50 resultados.

## **Segurança & privacidade**

* Rate-limit no backend e *API key* simples para endpoints de treino.
* Logs de proveniência (ID do dataset, versão do modelo) — úteis para auditoria.

# **5) Grad-CAM 1D na prática (resumo técnico)**

1. **No modelo sequencial (PyTorch)**, pegue a **última camada conv1d**.
2. **Forward pass**: guarde a *feature map* dessa camada.
3. **Backward pass**: compute gradiente da *logit* da classe prevista w.r.t. *feature map*.
4. **Pesos**: média global dos gradientes por canal.
5. **CAM 1D**: combinação linear dos mapas por canal com esses pesos; *ReLU*; normalize 0–1.
6. **UI**: transforme o vetor CAM em um *overlay* (alpha) sobre o gráfico da curva (tempo vs fluxo).

*(Se quiser, eu integro esse módulo ao starter kit e exponho em /api/explain já pronto.)*

# **6) Banco de dados & logs (mínimo viável)**

**Tabelas**

* datasets(id, source, mission, created\_at, schema\_hash)
* inferences(id, dataset\_id, target\_id, pred, proba, model\_version, created\_at)
* explanations(inference\_id, cam\_blob, shap\_blob, meta)
* labels(inference\_id, user, label, note, created\_at)
* models(version, type, train\_hash, metrics\_json, created\_at)

**Boas práticas**

* Salve *hash* dos CSVs e seed de treino.
* Toda release do modelo tem *artifact* + *metrics.json*.

# **7) Cronograma sugerido (para hackathon 48h)**

* **H0–H6**: ingestão, limpeza, baseline tabular + métricas.
* **H6–H14**: API + UI (upload, resultados, métricas).
* **H14–H24**: 1D-CNN simples + Grad-CAM 1D.
* **H24–H32**: DANN (adaptação de domínio) + validação *leave-one-mission-out*.
* **H32–H40**: Rotulagem ativa + relatórios PDF.
* **H40–H48**: ensaio do pitch, roteiro de demo, *backup video* da demo.

# **8) Próximos passos que posso te entregar agora**

* Incorporar **Grad-CAM 1D real** no starter kit (endpoint /api/explain).
* Gerar **página Next.js** inicial (Home, Analisar, Docs) com o design proposto.
* Gerar **app Expo** com as 5 telas (usando os CSVs simulados para demo offline).
* Refinar **slides** (versões 90s/2min/4min) e **PDF** one-pager do projeto.

1. **Site Next.js (App Router) — scaffold** • Páginas: Home, **/analisar** (upload + resultados + Explainer), **/docs** • Tailwind configurado e layout base com header/footer  
    • Contrato de integração com a API já previsto  
    👉 Baixar exoseeker\_site\_scaffold.zip
2. **App mobile (Expo React Native) — scaffold** • Telas: Home, Upload (demo offline), Results (lista + Explainer mock)  
    • Pronto para rodar com expo start e evoluir para chamadas reais da API  
    👉 Baixar exoseeker\_app\_scaffold.zip
3. **API atualizada com Grad-CAM 1D (versão simples para demo)** • Novo endpoint: GET /api/explain/{idx} que retorna { time, flux, cam }  
    • Implementação: destaca a região de trânsito simulada (mínimo de fluxo + gaussiana) para servir de **mock de Grad-CAM** até plugarmos o modelo 1D-CNN real.  
    • Já embutido no starter kit que você baixou antes.

Dicas para rodar tudo local:

* **Backend (API FastAPI)** # na pasta do starter kit
* pip install -r requirements.txt
* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
* Endpoints: /api/upload, /api/predict, /api/train, **/api/explain/{idx}**.
* **Site Next.js** unzip exoseeker\_site\_scaffold.zip -d exoseeker\_site
* cd exoseeker\_site
* npm install
* npm run dev
* Obs.: a página **/analisar** aponta para http://localhost:8000 (ajuste se necessário).
* **App Expo** unzip exoseeker\_app\_scaffold.zip -d exoseeker\_app
* cd exoseeker\_app
* npm install
* npm run start
* No início ele roda com **demo offline**; depois você pode trocar o mock por fetchs reais para o backend.

Extras já prontos pra você usar:

* Dataset simulado e gráficos:  
   • Candidate CSV | False Positive CSV  
   • Plot Candidate | Plot False Positive
* Slides:  
   • ExoSeeker.AI — Slides (PPTX)
* Starter kit (código/base):  
   • Baixar Starter Kit (ZIP)

###### 

1. **Grad-CAM 1D real (PyTorch) integrado à API**
   * Modelo **1D-CNN** em src/models/seq\_model.py
   * Utilitário **Grad-CAM** em src/utils/grad\_cam.py
   * Endpoint **GET /api/explain/{idx}** agora roda o modelo e retorna time, flux, cam, pred\_proba, pred\_class.
   * Pacote completo atualizado: **Baixar Starter Kit – UPDATED**
2. **Gráfico interativo (Plotly) no site /analisar**
   * Componente app/components/LightCurveCAM.tsx (usa react-plotly.js).
   * Página **/analisar** já consome /api/explain/{idx} e plota **Fluxo + Grad-CAM**.

* Para funcionar no seu Next.js, instale as dependências:  
    
   npm i react-plotly.js plotly.js
* npm run dev
  + O site scaffold que você já tem só precisa desses npm i e de o backend estar em http://localhost:8000.

1. **One-pager PDF para jurados/organizadores**
   * Resumo enxuto (problema, solução, UI, resultados, impacto, tech stack).
   * **Baixar One-Pager (PDF)**

Como rodar rapidamente:

* **API (FastAPI) com Grad-CAM 1D** unzip exoseeker\_starter\_kit\_UPDATED.zip -d exoseeker\_starter
* cd exoseeker\_starter
* pip install -r requirements.txt # pode ser necessário: pip install torch
* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
* **Site Next.js** unzip exoseeker\_site\_scaffold.zip -d exoseeker\_site
* cd exoseeker\_site
* npm install
* npm i react-plotly.js plotly.js
* npm run dev
* # abra http://localhost:3000/(site)/analisar
* **App Expo (mobile)** unzip exoseeker\_app\_scaffold.zip -d exoseeker\_app
* cd exoseeker\_app
* npm install
* npm run start

## **✅ 1) Relatórios PDF via API (com curva + Grad-CAM no PDF)**

* Novo endpoint: **POST /api/report/{idx}** Body esperado:  
    
   { "time": [...], "flux": [...], "cam": [...], "meta": { "mission": "tess", "target\_id": "TIC123" } }
* Resposta: {"ok": true, "pdf": "reports/ExoSeeker\_Report\_<id>.pdf"}
* A API renderiza um **PNG** (curva + Grad-CAM sobreposto) e embute no PDF (A4) com metadados.
* Já integrei o botão **“Salvar relatório PDF”** na página **/analisar** (site Next.js).
* Está tudo no pacote atualizado: **Baixar Starter Kit UPDATED v2**

## **✅ 2) Rotulagem Ativa na UI (fila de incerteza)**

* Endpoints novos:  
  + **GET /api/active/queue** → retorna uma **fila mock** com itens incertos (tempo/flux/cam/meta/uncertainty).
  + **POST /api/label** → grava rótulo do humano (CANDIDATE | FALSE\_POSITIVE | ARTIFACT) e salva em reports/labels.jsonl.
* Na página **/analisar**:  
  + Bloco “**Rotulagem Ativa**” mostra cartões da fila.
  + Botões enviam rótulos e atualizam a fila.
  + Mensagem de confirmação exibe quantos itens restam.
* Isso já serve para **demo** e para instrumentar um fluxo de *active learning* real depois.

## **✅ 3) Script de treino reprodutível (1D-CNN, seed, metrics.json e .pt)**

* Arquivo: src/train\_seq.py
* O que faz:  
  + **Semente fixa** e determinismo de cudnn.
  + Gera um **dataset simulado** de curvas (3 classes: Candidate, False Positive, Other).
  + Treina **SimpleCNN1D** (PyTorch) e salva o melhor checkpoint em checkpoints/model\_seed<seed>.pt.
  + Salva **metrics.json** com histórico de *loss* (train/val), hiperparâmetros e timestamp.
* Exemplo de uso:  
    
   cd exoseeker\_starter
* pip install -r requirements.txt # e, se preciso, pip install torch
* python src/train\_seq.py --seed 1337 --epochs 10 --batch\_size 64 --lr 0.001 --outdir checkpoints

## **Como rodar o fluxo ponta-a-ponta agora**

### **Backend (FastAPI)**

* unzip exoseeker\_starter\_kit\_UPDATED\_v2.zip -d exoseeker\_starter
* cd exoseeker\_starter
* pip install -r requirements.txt
* # se faltar, instale PyTorch: pip install torch --index-url https://download.pytorch.org/whl/cpu
* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

### **Site (Next.js)**

* unzip exoseeker\_site\_scaffold.zip -d exoseeker\_site
* cd exoseeker\_site
* npm install
* npm i react-plotly.js plotly.js
* npm run dev
* # abra http://localhost:3000/(site)/analisar

No **/analisar**:

1. Faça upload de um CSV (ou use qualquer arquivo pra demo de predição mock).
2. Clique em **Explainer** → aparece o gráfico **Fluxo + Grad-CAM** (via /api/explain/{idx}).
3. Clique em **Salvar relatório PDF** → a API devolve o caminho do PDF gerado.
4. Em **Rotulagem Ativa**, rotule itens incertos para o *loop* de aprendizado.

## **1) Persistência real (SQLite) para Rotulagem Ativa + rótulos**

* Fila e rótulos agora ficam no **SQLite** (exoseeker.db), em vez de memória.
* Endpoints:  
  + **GET /api/active/queue** → lê da tabela active\_queue (se vazio, faz seeding).
  + **POST /api/label** → grava na tabela labels e remove item da fila.
* O banco já está criado no pacote atualizado.

## **2) Carregamento de pesos do 1D-CNN**

* Endpoint novo: **POST /api/model/load?version=<path>**
  + Ex.: version=checkpoints/model\_seed42.pt
  + Carrega state\_dict e coloca o modelo em eval().

## **3) Relatório PDF com SHAP opcional**

* Endpoint **POST /api/report/{idx}** (atualizado):  
  + Gera **PDF A4** com **gráfico da curva + Grad-CAM**.
  + Se enviar shap: [{feature, impact}, ...] no body, embute um **gráfico de barras** com top features.
* Corpo esperado (exemplo):  
    
   {
* "time": [ ... ],
* "flux": [ ... ],
* "cam": [ ... ],
* "shap": [
* {"feature": "period", "impact": 0.42},
* {"feature": "duration", "impact": 0.35},
* {"feature": "planet\_radius", "impact": 0.27}
* ],
* "meta": {"mission": "tess", "target\_id": "TIC123"}
* }
* Resposta: {"ok": true, "pdf": "reports/ExoSeeker\_Report\_<id>.pdf"}

## **Downloads**

* **Starter Kit (v3) com tudo isso integrado**:  
   👉 Baixar exoseeker\_starter\_kit\_UPDATED\_v3.zip  
   (Contém exoseeker.db, API com novos endpoints, e as integrações anteriores.)

## **Como usar agora**

### **Backend (API FastAPI)**

* unzip exoseeker\_starter\_kit\_UPDATED\_v3.zip -d exoseeker\_starter
* cd exoseeker\_starter
* pip install -r requirements.txt
* # se necessário: pip install torch reportlab
* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

### **Carregar pesos do 1D-CNN (quando tiver um .pt salvo)**

* curl -X POST "http://localhost:8000/api/model/load?version=checkpoints/model\_seed42.pt"

### **Fila de Rotulagem Ativa**

* curl "http://localhost:8000/api/active/queue"
* # rotular
* curl -X POST "http://localhost:8000/api/label" \
* -H "Content-Type: application/json" \
* -d '{"id":0,"label":"CANDIDATE","notes":"trânsito claro"}'

### **Gerar PDF (com SHAP, se quiser)**

* curl -X POST "http://localhost:8000/api/report/0" \
* -H "Content-Type: application/json" \
* -d '{
* "time":[0,1,2], "flux":[1,0.998,1.001], "cam":[0.1,0.9,0.2],
* "shap":[{"feature":"period","impact":0.42},{"feature":"duration","impact":0.35}],
* "meta":{"mission":"tess","target\_id":"TIC123"}
* }'

## **✅ O que foi adicionado agora**

### **1) SHAP tabular via API (para LightGBM)**

* Endpoint novo: **POST /api/shap/tabular** Body:  
    
   {
* "rows": [
* {"period": 3.14, "duration": 2.1, "planet\_radius": 1.8}
* ],
* "topk": 10
* }
* Retorno: top features por linha com magnitude de impacto (|SHAP|).  
   Obs.: Se o modelo LightGBM (MODEL) ainda não tiver sido treinado via /api/train, a API cria um **modelo dummy** apenas para demonstrar SHAP.

### **2) Nomenclatura padronizada para PDFs**

* Agora os PDFs são salvos como:  
   reports/ExoSeeker\_Report\_<MISSION>\_<TARGET>\_<UTC-TS>.pdf  
   (ex.: ExoSeeker\_Report\_TESS\_TIC123\_20251002T140305Z.pdf)
* Basta enviar meta: { "mission": "...", "target\_id": "..." } no corpo do /api/report/{idx}.

### **3) Admin endpoints para auditoria**

* **GET /api/admin/labels** → últimos rótulos (tabela labels no SQLite).
* **GET /api/admin/reports** → lista de PDFs em reports/.

### **4) Página /admin no site (Next.js)**

* Nova rota: **/(site)/admin**
* Mostra **rótulos recentes** e **lista de PDFs** (consome os endpoints acima).

## **Downloads**

* Starter kit atualizado com tudo acima:  
   👉 **exoseeker\_starter\_kit\_UPDATED\_v4.zip**
* A página de admin foi criada aqui no scaffold do site:  
   /app/(site)/admin/page.tsx

## **Como usar**

### **Backend**

* unzip exoseeker\_starter\_kit\_UPDATED\_v4.zip -d exoseeker\_starter
* cd exoseeker\_starter
* pip install -r requirements.txt
* # se necessário: pip install torch shap reportlab
* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

### **Site**

* # usando o scaffold que você já baixou
* cd exoseeker\_site
* npm i
* npm i react-plotly.js plotly.js
* npm run dev
* # visite http://localhost:3000/(site)/admin para ver auditoria

### **Exemplos rápidos de API**

**SHAP Tabular**

* curl -X POST "http://localhost:8000/api/shap/tabular" \
* -H "Content-Type: application/json" \
* -d '{"rows":[{"period":3.14,"duration":2.1,"planet\_radius":1.8}],"topk":5}'

**Gerar PDF padronizado (com SHAP opcional)**

* curl -X POST "http://localhost:8000/api/report/0" \
* -H "Content-Type: application/json" \
* -d '{
* "time":[0,1,2], "flux":[1,0.998,1.001], "cam":[0.1,0.9,0.2],
* "meta":{"mission":"tess","target\_id":"TIC123"},
* "shap":[{"feature":"period","impact":0.42},{"feature":"duration","impact":0.35}]
* }'

**Auditoria**

* curl "http://localhost:8000/api/admin/labels"
* curl "http://localhost:8000/api/admin/reports"

## **✅ Novidades deste pacote**

### **1) Admin com download de PDFs e filtros**

* Endpoint: **GET /api/admin/report/{filename}** → baixa o PDF diretamente.
* Endpoint: **GET /api/admin/labels?label=...** → filtra rótulos por classe (ex.: FALSE\_POSITIVE).
* Página web **/admin** atualizada:  
  + Filtro por label (Todos, CANDIDATE, FALSE\_POSITIVE, ARTIFACT).
  + Links clicáveis para baixar PDFs gerados.

### **2) Carregamento do melhor checkpoint do 1D-CNN**

* Endpoint: **POST /api/model/load\_best**
  + Lê checkpoints/metrics.json e carrega automaticamente o .pt correspondente (fallback: primeiro .pt encontrado).
* Continua disponível o **POST /api/model/load?version=...** para apontar um arquivo específico.

### **3) /docs enriquecido (site)**

* Documentação com **exemplos de curl** para cada endpoint.
* Dica para abrir a documentação interativa do FastAPI: **/docs** e **/redoc**.

## **📦 Download**

* Pacote atualizado (com API, SQLite, relatórios, SHAP tabular, admin, etc.):  
   👉 **exoseeker\_starter\_kit\_UPDATED\_v5.zip**

## **Como rodar**

### **Backend**

* unzip exoseeker\_starter\_kit\_UPDATED\_v5.zip -d exoseeker\_starter
* cd exoseeker\_starter
* pip install -r requirements.txt
* # se precisar: pip install torch shap reportlab
* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

### **Site**

* cd exoseeker\_site
* npm install
* npm i react-plotly.js plotly.js
* npm run dev
* # abra http://localhost:3000/(site)/docs e http://localhost:3000/(site)/admin

## **✅ Novidades**

### **1) Autenticação simples (token) nos endpoints “sensíveis”**

* Novo header: **X-Admin-Token**
* Configure o token via env: EXOSEEKER\_ADMIN\_TOKEN (padrão: changeme)
* Protegidos com token:  
  + GET /api/admin/labels
  + GET /api/admin/reports
  + GET /api/admin/report/{filename}
  + POST /api/admin/upload-model
  + POST /api/model/load
  + POST /api/model/load\_best

### **2) Upload de modelo pelo navegador (admin)**

* Endpoint: **POST /api/admin/upload-model**
* Envie um .pt (multipart/form-data) + header X-Admin-Token
* Salva em checkpoints/uploaded\_<timestamp>.pt

### **3) Script de avaliação com PR curves direto no PDF**

* **src/eval\_seq.py**
* Carrega pesos .pt, avalia em dataset simulado, gera:  
  + checkpoints/eval\_metrics.json (AP por classe + classification report)
  + checkpoints/pr\_\*.png (PR de cada classe)
  + checkpoints/EvalReport\_<UTC-TS>.pdf com os gráficos embutidos
* Exemplo:  
    
   cd exoseeker\_starter
* pip install -r requirements.txt # e se faltar: pip install torch scikit-learn reportlab
* python src/train\_seq.py --seed 42 --epochs 6 --outdir checkpoints
* python src/eval\_seq.py --weights checkpoints/model\_seed42.pt --seed 1337 --outdir checkpoints

## **📦 Download**

👉 **exoseeker\_starter\_kit\_UPDATED\_v6.zip**

## **Como usar rapidamente**

### **Backend**

* unzip exoseeker\_starter\_kit\_UPDATED\_v6.zip -d exoseeker\_starter
* cd exoseeker\_starter
* export EXOSEEKER\_ADMIN\_TOKEN="meu-token-seguro"
* pip install -r requirements.txt
* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

### **Upload do modelo e carregamento**

* # upload (no terminal: use httpie ou curl)
* curl -X POST "http://localhost:8000/api/admin/upload-model" \
* -H "X-Admin-Token: meu-token-seguro" \
* -F "file=@checkpoints/model\_seed42.pt"
* # carregar o melhor checkpoint
* curl -X POST "http://localhost:8000/api/model/load\_best" \
* -H "X-Admin-Token: meu-token-seguro"

### **Avaliação e relatório de PR curves**

* python src/eval\_seq.py --weights checkpoints/model\_seed42.pt --seed 1337 --outdir checkpoints
* # abre o PDF gerado em checkpoints/EvalReport\_<UTC-TS>.pdf

Se quiser, faço mais dois acabamentos:

* UI no **/admin** com formulário de **upload de modelo** e campo para o token,
* botão na página **/analisar** para **baixar diretamente** o PDF gerado (em vez de só mostrar o caminho).

##### **Você disse:**

sim

###### **O ChatGPT disse:**

Perfeito — finalizei os dois acabamentos de UI que você pediu.

## **✅ O que ficou pronto agora**

### **1) /admin com upload de modelo e token**

* Campo para **X-Admin-Token** (fica em memória local da página).
* **Upload de modelo (.pt)** direto pelo browser (envia para POST /api/admin/upload-model).
* Botão **“Carregar melhor (metrics.json)”** que chama POST /api/model/load\_best.
* Auditoria continua com **filtro por label** e **links pra PDFs** (respeitando o token).

### **2) /analisar com botão “Gerar & Baixar PDF”**

* Ao clicar, a página:  
  1. chama POST /api/report/{id} para gerar o PDF padronizado;
  2. baixa o arquivo usando GET /api/admin/report/{filename} **incluindo o token** informado.
* Adicionei um campo **X-Admin-Token** também em **/analisar** só para este download.

## **📦 Downloads atualizados**

* **Site (v2) com as novas páginas** 👉 exoseeker\_site\_scaffold\_v2.zip

*(O backend já foi atualizado nos pacotes v5/v6 anteriores com os endpoints e segurança.)*

## **Como testar em 2 passos**

1. **Backend**

* unzip exoseeker\_starter\_kit\_UPDATED\_v6.zip -d exoseeker\_starter
* cd exoseeker\_starter
* export EXOSEEKER\_ADMIN\_TOKEN="meu-token-seguro"
* pip install -r requirements.txt
* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

1. **Site**

* unzip exoseeker\_site\_scaffold\_v2.zip -d exoseeker\_site
* cd exoseeker\_site
* npm install
* npm i react-plotly.js plotly.js
* npm run dev
* # / (home), /(site)/analisar e /(site)/admin

Dica de fluxo para a demo:

* em **/analisar**, faça um upload de CSV (ou qualquer arquivo pra mock), clique em **Explainer** e depois **Gerar & Baixar PDF**.
* Informe o **X-Admin-Token** igual ao do backend para baixar o PDF imediatamente.
* em **/admin**, use o mesmo token para **upload** de um .pt e **carregar o best** via metrics.json.

## **✅ O que entrou agora**

### **1) Token lembrado no navegador (localStorage)**

* Criei o hook useAdminToken() e integrei em **/analisar** e **/admin**.
* O valor do **X-Admin-Token** é persistido (localStorage) e reutilizado automaticamente.

### **2) Links públicos assinados para PDFs (sem token)**

* API:  
  + **GET /api/admin/report/signed?filename=...&ttl=600** → gera URL pública com assinatura HMAC e expiração (padrão 10 min).
  + **GET /api/public/report?f=...&exp=...&sig=...** → faz o download sem token, validando assinatura/expiração.
* Variável de ambiente: EXOSEEKER\_PUBLIC\_LINK\_SECRET (default seguro de demo: "public-secret").
* UI (página **/admin**): botão “**Gerar link público**” copia a URL para a área de transferência.

### **3) Onboarding wizard (3 passos) na Home**

* Uma caixinha “Onboarding” com os passos: **Upload & Explainer → PDF & Compartilhar → Aprimorar modelo**.
* Botão **Concluir** que esconde o wizard (salva exoseeker\_wizard\_done=1 no localStorage).

## **📦 Downloads atualizados**

* **Site (v3)** com wizard, token persistido e geração de link público:  
   👉 exoseeker\_site\_scaffold\_v3.zip
* **Starter Kit (v7)** com endpoints de link assinado e tudo anterior:  
   👉 exoseeker\_starter\_kit\_UPDATED\_v7.zip

## **Como rodar (resumo)**

### **Backend**

* unzip exoseeker\_starter\_kit\_UPDATED\_v7.zip -d exoseeker\_starter
* cd exoseeker\_starter
* export EXOSEEKER\_ADMIN\_TOKEN="meu-token-seguro"
* export EXOSEEKER\_PUBLIC\_LINK\_SECRET="minha-chave-de-assinatura"
* pip install -r requirements.txt
* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

### **Site**

* unzip exoseeker\_site\_scaffold\_v3.zip -d exoseeker\_site
* cd exoseeker\_site
* npm install
* npm i react-plotly.js plotly.js
* npm run dev
* # Home (wizard), /(site)/analisar e /(site)/admin

### **Dica rápida**

* Gere um relatório em **/analisar** → na **/admin** gere o **link público** e compartilhe (sem expor o token).
* O token agora é lembrado automaticamente entre as páginas.

showtime! aqui vai um **guia passo-a-passo, do zero**, para você colocar **todo o projeto no ar**: backend (FastAPI + PyTorch), site (Next.js + Plotly), e app (Expo), com GitHub, secrets, build e deploy.

# **0) Pré-requisitos (local)**

* **Git** e uma conta no **GitHub**
* **Python 3.10+** (com pip)
* **Node 18+** e **npm**
* **Expo CLI** (instala depois com npm i -g expo-cli, opcional)
* (Opcional) **conda** ou **uv/venv** para isolar ambiente Python

# **1) Crie o repositório no GitHub**

1. Acesse GitHub → **New repository** Nome sugerido: exoseeker-ai (público ou privado).
2. Não precisa README/licença agora (vamos subir arquivos completos).

# **2) Prepare os artefatos locais**

Você já tem os pacotes prontos para baixar aqui no chat:

* **Starter Kit (v7)** — API + scripts + banco SQLite:  
  + exoseeker\_starter\_kit\_UPDATED\_v7.zip
* **Site (v3)** — Next.js com wizard, Plotly, /admin:  
  + exoseeker\_site\_scaffold\_v3.zip
* (Opcional) App mobile (Expo) — caso ainda não tenha baixado:  
  + exoseeker\_app\_scaffold.zip (ou o último que te forneci antes)

Crie uma pasta de trabalho e extraia:

* mkdir exoseeker-ai
* cd exoseeker-ai
* # backend
* unzip ~/Downloads/exoseeker\_starter\_kit\_UPDATED\_v7.zip -d exoseeker\_starter
* # site
* unzip ~/Downloads/exoseeker\_site\_scaffold\_v3.zip -d exoseeker\_site
* # app (se quiser rodar também)
* unzip ~/Downloads/exoseeker\_app\_scaffold.zip -d exoseeker\_app

Se os nomes dos arquivos diferirem, ajuste o caminho/arquivo.

# **3) Inicie o Git e primeiro commit**

* git init
* git branch -M main
* echo -e "exoseeker\_starter/\_\_pycache\_\_/\nexoseeker\_starter/.venv/\nexoseeker\_site/.next/\nexoseeker\_site/node\_modules/\nexoseeker\_app/node\_modules/\n.DS\_Store\n" > .gitignore
* git add .
* git commit -m "ExoSeeker.AI: backend+site+app scaffolds v7"
* git remote add origin https://github.com/<seu-usuario>/exoseeker-ai.git
* git push -u origin main

# **4) Rodar localmente — Backend (FastAPI)**

## **4.1 Criar e ativar ambiente Python**

* cd exoseeker\_starter
* python -m venv .venv
* source .venv/bin/activate # Windows: .venv\Scripts\activate
* pip install -r requirements.txt
* # se necessário:
* # pip install torch --index-url https://download.pytorch.org/whl/cpu
* # pip install shap reportlab scikit-learn

## **4.2 Variáveis de ambiente**

* export EXOSEEKER\_ADMIN\_TOKEN="meu-token-seguro"
* export EXOSEEKER\_PUBLIC\_LINK\_SECRET="minha-chave-de-assinatura"

Windows (PowerShell):

* $env:EXOSEEKER\_ADMIN\_TOKEN="meu-token-seguro"
* $env:EXOSEEKER\_PUBLIC\_LINK\_SECRET="minha-chave-de-assinatura"

## **4.3 Subir a API**

* uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

Endpoints úteis:

* GET /docs (Swagger)
* GET /api/active/queue, POST /api/label
* POST /api/report/{id} (gera PDF) e GET /api/admin/report/{filename} (baixa PDF)
* POST /api/admin/upload-model (upload de .pt) — exige X-Admin-Token
* POST /api/model/load\_best (carrega melhor checkpoint)
* GET /api/admin/report/signed?filename=... (gera link público assinado)

# **5) Rodar localmente — Site (Next.js)**

Em outro terminal:

* cd exoseeker\_site
* npm install
* npm i react-plotly.js plotly.js
* npm run dev
* # abre http://localhost:3000

Páginas:

* **Home** (com wizard onboarding)
* **/(site)/analisar** — upload, predição mock, **Explainer** com Plotly, **Gerar & Baixar PDF** (usa token)
* **/(site)/admin** — auditoria, filtro de rótulos, **Upload de modelo** com token, **Gerar link público** para PDF

Dica: informe o mesmo **X-Admin-Token** na UI (persistido no navegador).

# **6) (Opcional) Rodar o App mobile (Expo)**

* cd exoseeker\_app
* npm install
* npm run start
* # ou: npx expo start

Abra no iOS/Android com o app Expo Go, ou rode em emulador.

# **7) Treinar, avaliar e carregar um modelo 1D-CNN**

## **7.1 Treino reprodutível (dataset simulado)**

* cd exoseeker\_starter
* source .venv/bin/activate
* python src/train\_seq.py --seed 42 --epochs 8 --batch\_size 64 --outdir checkpoints
* # gera: checkpoints/model\_seed42.pt e checkpoints/metrics.json

## **7.2 Avaliação + PDF de PR-curves**

* python src/eval\_seq.py --weights checkpoints/model\_seed42.pt --seed 1337 --outdir checkpoints
* # gera: checkpoints/EvalReport\_<timestamp>.pdf e eval\_metrics.json

## **7.3 Carregar o melhor checkpoint na API**

* curl -X POST "http://localhost:8000/api/model/load\_best" -H "X-Admin-Token: meu-token-seguro"

# **8) Subir tudo para produção (deploy)**

## **8.1 Backend (Render/railway/fly.io) — exemplo Render**

1. Faça login em **render.com** → **New +** → **Web Service** → **Build from GitHub** → selecione exoseeker-ai.
2. Root directory: exoseeker\_starter

* **Start Command**:  
    
   uvicorn app.main:app --host 0.0.0.0 --port $PORT

1. **Environment**:  
   * EXOSEEKER\_ADMIN\_TOKEN=...
   * EXOSEEKER\_PUBLIC\_LINK\_SECRET=...
2. **Python version** e **build**: Render detecta requirements.txt.

Dica: garanta pasta reports/ criada na inicialização (pode estar no código; já criamos quando gera o PDF).

## **8.2 Site (Vercel) — Next.js**

1. Vercel → **New Project** → importe do GitHub.
2. Root directory: exoseeker\_site
3. Build preset: **Next.js**
4. Em **Environment Variables**, configure:  
   * NEXT\_PUBLIC\_API\_BASE (opcional: use http(s)://<seu-backend> se quiser externalizar; por padrão está localhost:8000 no código, ajuste os fetches se precisar)
5. Deploy.

Se seu backend for HTTPS em outro domínio, ajuste as URLs nos fetch() do site (http://localhost:8000/... → https://api.sua-url.com/...).

## **8.3 App (Expo) — produção**

* Ajuste as URLs do backend no app para seu domínio público.
* Gere builds:  
    
   npx expo prebuild
* expo run:android # ou run:ios
* (ou use **EAS Build** para enviar às lojas)

# **9) GitHub: Secrets + CI (opcional)**

### **9.1 GitHub Secrets**

No repositório → **Settings → Secrets and variables → Actions**:

* EXOSEEKER\_ADMIN\_TOKEN
* EXOSEEKER\_PUBLIC\_LINK\_SECRET

### **9.2 Workflow simples (API) — .github/workflows/api-ci.yml**

* name: api-ci
* on: [push]
* jobs:
* test-api:
* runs-on: ubuntu-latest
* defaults:
* run:
* working-directory: exoseeker\_starter
* steps:
* - uses: actions/checkout@v4
* - uses: actions/setup-python@v5
* with: { python-version: "3.11" }
* - run: python -m pip install -r requirements.txt
* - run: python -m pyflakes app || true

### **9.3 Workflow (Site) — .github/workflows/site-ci.yml**

* name: site-ci
* on: [push]
* jobs:
* build-site:
* runs-on: ubuntu-latest
* defaults:
* run:
* working-directory: exoseeker\_site
* steps:
* - uses: actions/checkout@v4
* - uses: actions/setup-node@v4
* with: { node-version: "18" }
* - run: npm ci
* - run: npm run build

Para deploy contínuo, conecte o GitHub ao Render (API) e Vercel (site). Eles constroem a cada push.

# **10) Fluxo de demo oficial (como apresentar)**

1. Abra o site (deploy Vercel) → **Home** (wizard) → **Analisar**.
2. Faça upload de um CSV (pode ser o simulado), clique **Explainer** → veja **Flux + Grad-CAM**.
3. Clique **Gerar & Baixar PDF** → na **/admin**, gere **link público** e copie.
4. Em **/admin**, faça **upload** de um .pt custom (se quiser) e clique **Carregar melhor** (via metrics.json).
5. Mostre **/admin** com rótulos (Rotulagem Ativa) e a lista de PDFs gerados.

# **11) Problemas comuns & soluções rápidas**

* **Torch não instala**: use a **wheel CPU**:  
   pip install torch --index-url https://download.pytorch.org/whl/cpu
* **SHAP/LightGBM** em ambiente restrito: comente a rota POST /api/shap/tabular até instalar shap/lightgbm.
* **CORS entre site e API**: se hospedar domínios diferentes, ative CORS no FastAPI (middleware) para o domínio do site.
* **PDF não baixa**: verifique o token na UI (localStorage) e se o backend tem EXOSEEKER\_ADMIN\_TOKEN configurado.
* **Links públicos**: defina EXOSEEKER\_PUBLIC\_LINK\_SECRET, gere link em **/admin** e valide se exp não expirou.

# **12) Próximos upgrades (quando quiser)**

* Persistir **fila de Rotulagem Ativa** diretamente a partir das predições reais.
* Etapas de **fine-tuning** com os rótulos novos (loop de *active learning*).
* **Dark mode** e gráficos de métricas globais (PR/AUC) na **/admin**.
* Dockerfiles e docker-compose.yml para um deploy containerizado.