

Perfeito, Pablo. Como **indie hacker**, seu objetivo provavelmente é:

- ✓ Ter dados confiáveis e atualizados com frequência
- ✓ Gastar pouco (sem pagar preços corporativos absurdos)
- ✓ Escalar futuramente sem refazer tudo
- ✓ Publicar seu app na web com API acessível para consultas (talvez com limite de requisições)

---

### Melhor estratégia para um indie hacker

#### Requisitos que você deve priorizar:

1. **API com dados da B3 (FIIs, ações, dividendos, histórico)**
2. **Preço acessível (até ~US\$ 30/mês no início)**
3. **Documentação simples + SDKs**
4. **Permissão comercial para uso em SaaS/app público**

---

### Melhoras opções para você (realistas)


#### ✓ 1. Twelve Data – Melhor custo-benefício para começar

- ✓ Acesso a ações e FIIs da B3 (ex: HGLG11.SA)
- ✓ Retorno em JSON pronto pra frontend
- ✓ Histórico, dividendos, candles, volume, etc.
- ✓ SDKs em Python, JS, etc.
- ✓ Permite uso comercial (até certo volume)

#### Planos:

##### Plano   Requisições/mês   Intervalo de dados   Preço

Free	800	Delay de 1 min	US\$ 0
Starter	8.000	Real-time (15s)	US\$ 29
Pro+	1.000.000	Real-time (<1s)	US\$ 79

 Ver planos e limites

 **Ideal pra você:** Comece com o plano Starter (US\$ 29/mês)

---

## ✅ 2. Finnhub.io – Alternativa com foco global

- Dados de ações e FIIs de vários mercados
- API bem estruturada
- Até 60 requisições/min no plano pago
- Permite uso comercial (ver termo de uso)

**Plano Pro:** US\$ 49/mês

Menos cobertura de FIIs brasileiros comparado à TwelveData

---

## ✅ 3. Xignite – Para escalar grande (não indicado no começo)

- Dados da B3 e mundiais
- Muito usada por apps como Robinhood, Investing, etc.
- Foco corporativo – preços sob demanda
- API complexa, mas muito precisa

👉 Não recomendada no início por preço e complexidade

---

## ⚠ Cuidado com scraping!

Evite depender de scraping de sites como Funds Explorer ou StatusInvest. Mesmo que funcione agora, **você pode ser bloqueado ou até notificado legalmente** se usar em app comercial.

---

## 📁 Sugestão de estrutura para seu app (modo indie)

flowchart LR

Frontend(App React/Vue/Streamlit) --> |Chama API| Backend(API Python/FastAPI)

Backend --> |Consulta dados| TwelveData API

Backend --> |Armazena cache| Banco (SQLite / PostgreSQL)

- Faça cache dos dados para evitar estourar a cota do plano
- Use a API da TwelveData para dados atualizados a cada 15s-1min

- Você pode usar uma camada gratuita (Firebase, Railway, Vercel) para hospedar o backend
- 

### Quer ajuda com o setup?

Se quiser, posso:

- Criar o backend base FastAPI com integração à TwelveData
- Gerar as rotas para frontend consumir via REST/GraphQL
- Ajudar a configurar cache local (evita queimar requisições)
- Testar com 2 ou 3 FIIs como exemplo (HGLG11, VILG11, VINO11)

### Plano Starter – Resumo:

- **Preço: US\$ 29/mês**
  - **Incluso:**
    - 8.000 requisições/mês (cerca de 266 por dia)
    - Atualização dos dados: a cada 15 segundos
    - Acesso a:
      - Preço em tempo real
      - Volume, dividendos, candles
      - Dados históricos e atuais
    - Suporte básico
    - **Uso comercial liberado**
- 

### Pode ser ideal pra você se:

- Está começando como indie hacker
- Tem um app com até alguns milhares de usuários
- Deseja escalar com custo controlado

- Quer segurança legal para uso público (sem scraping)
- 

Se preferir pagar **anualmente**, eles também oferecem desconto.  
Mas o pagamento **mensal** funciona bem pra começar sem compromisso.

---

Se quiser, posso te ajudar a:

- Simular o consumo de requisições por FII no app
- Calcular se 8.000/mês dá conta do que você precisa
- Montar um backend com cache inteligente pra evitar desperdício

## Cálculo de usuários com base em uso médio

### Suponha:

- Seu app consulta **5 dados por FII** (ex: preço, DY, volume, último dividendo, variação)
- Cada usuário acompanha em média **10 FIIs**
- Você **otimiza com cache** para fazer uma consulta por FII por minuto ou por hora (não por usuário)

---

### Com cache (modelo eficiente - recomendado)

Você consulta cada FII **uma vez a cada X minutos** e **serve o dado para todos os usuários**:

Frequência de atualização	FIIs monitorados	Quantas atualizações por dia	Requisições por mês	Usuários suportados
1 vez por hora	50	$24 \times 50 = 1.200$	36.000	<b>Até 10.000+ usuários</b> (se cacheado)
1 vez a cada 6 horas	50	$8 \times 50 = 400$	12.000	<b>4.000–8.000 usuários</b>
1 vez a cada 12 horas	50	$4 \times 50 = 200$	6.000	<b>~5.000 usuários com folga</b>
1 vez ao dia	50	$1 \times 50 = 50$	1.500	<b>10.000+ usuários facilmente</b>

---

### Sem cache (ineficiente – cada usuário gera chamadas diretas)

**Usuários FIIs consultados por usuário Dados por FII Requisições por mês**

100	10	5	5.000
250	10	5	12.500 ( <b>excede</b> )

→ **Você atingiria o limite com ~150–200 usuários ativos**, se não usar cache.

---

## Conclusão

Se usar **cache e uma arquitetura eficiente**, você pode atender até **10.000 usuários ou mais** com o plano Starter.

O segredo está em **centralizar as chamadas** para a API e servir os dados de forma inteligente com um **backend com banco (Redis ou PostgreSQL)**.