CLAUDE.md - Google Maps Lead Finder

Contexto do Projeto

Sou uma agência de marketing digital e preciso de uma ferramenta de prospeção de leads B2B que extraia dados de negócios do Google Maps. O objetivo é identificar potenciais clientes que possam beneficiar dos nossos serviços de marketing digital.

Objetivo Principal

Criar uma CLI tool (e opcionalmente uma interface web simples) que permita:

1. Pesquisar negócios no Google Maps com filtros avançados
2. Guardar os resultados numa base de dados local
3. Detetar novos negócios (comparando snapshots ao longo do tempo)
4. Exportar leads qualificados em vários formatos

Stack Tecnológico

- **Linguagem**: Python 3.11+
- **Base de Dados**: SQLite (simplicidade) ou PostgreSQL (se escalar)
- **API**: Google Places API (New) - https://developers.google.com/maps/documentation/places/web-service
- **CLI Framework**: Typer ou Click
- **HTTP Client**: httpx ou requests
- **ORM**: SQLAlchemy ou raw SQL
- **Export**: pandas para CSV/Excel, openpyxl para Excel formatado
- **Config**: python-dotenv para variáveis de ambiente
- **Optional Web UI**: FastAPI + HTMX ou Streamlit

Estrutura do Projeto

```
google-maps-lead-finder/
├── README.md
├── requirements.txt
├── .env.example
├── .gitignore
├── setup.py
├── src/
│   ├── __init__.py
```

```
│   ├── main.py             # Entry point CLI
│   ├── config.py           # Configurações e env vars
│   ├── api/
│   │   ├── __init__.py
│   │   ├── google_places.py # Client para Google Places API
│   │   └── models.py       # Pydantic models para responses
│   ├── database/
│   │   ├── __init__.py
│   │   ├── db.py           # Conexão e setup
│   │   ├── models.py       # SQLAlchemy models
│   │   └── queries.py      # Queries reutilizáveis
│   ├── services/
│   │   ├── __init__.py
│   │   ├── search.py       # Lógica de pesquisa
│   │   ├── tracker.py      # Deteção de novos negócios
│   │   ├── scorer.py       # Lead scoring
│   │   └── exporter.py     # Exportação de dados
│   └── utils/
│       ├── __init__.py
│       └── helpers.py
├── tests/
│   └── ...
└── data/
    └── leads.db            # SQLite database
```

Funcionalidades Requeridas

1. Pesquisa de Negócios (CORE)

```bash
# Exemplos de uso esperado:
leadfinder search --location "Lisboa, Portugal" --radius 5000 --type "restaurant"
leadfinder search --location "41.1579,-8.6291" --radius 10000 --type "dentist" --min-reviews 0 --max-reviews 10
leadfinder search --query "agência imobiliária Porto"
```

Filtros a implementar:

Filtro	Descrição	Prioridade
`--location`	Cidade, morada ou coordenadas (lat,lng)	Alta
`--radius`	Raio em metros (max 50000)	Alta
`--type`	Tipo de negócio (Google Place Types)	Alta
`--query`	Texto livre de pesquisa	Alta
`--min-reviews`	Mínimo de reviews	Alta
`--max-reviews`	Máximo de reviews (poucos = oportunidade)	Alta
`--min-rating`	Rating mínimo	Média
`--max-rating`	Rating máximo	Média
`--has-website`	Filtrar por ter/não ter website	Alta
`--has-phone`	Filtrar por ter/não ter telefone	Média
`--price-level`	Nível de preço (1-4)	Baixa
`--open-now`	Apenas abertos agora	Baixa
`--language`	Idioma dos resultados	Média

2. Base de Dados de Leads

Schema principal - Tabela `businesses`:

```sql
```

```
CREATE TABLE businesses (
    id TEXT PRIMARY KEY,                -- Google Place ID
    name TEXT NOT NULL,
    formatted_address TEXT,
    latitude REAL,
    longitude REAL,
    place_types TEXT,                   -- JSON array
    business_status TEXT,               -- OPERATIONAL, CLOSED, etc.

    -- Contactos
    phone_number TEXT,
    website TEXT,
    google_maps_url TEXT,

    -- Métricas
    rating REAL,
    review_count INTEGER,
    price_level INTEGER,

    -- Análise interna
    has_website BOOLEAN,
    has_photos BOOLEAN,
    photo_count INTEGER,

    -- Lead scoring
    lead_score INTEGER,                 -- 0-100
    lead_status TEXT DEFAULT 'new',     -- new, contacted, qualified, converted, rejected
    notes TEXT,

    -- Timestamps
    first_seen_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    last_updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    last_search_query TEXT              -- Para saber de que pesquisa veio
);

CREATE TABLE search_history (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    query_params TEXT,                  -- JSON dos parâmetros usados
    results_count INTEGER,
    new_businesses_count INTEGER,
    executed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE business_snapshots (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    business_id TEXT,
```

```
    snapshot_data TEXT,                -- JSON completo
    captured_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (business_id) REFERENCES businesses(id)
);
```

3. Deteção de Novos Negócios

```bash
# Comparar com última pesquisa
leadfinder track --location "Lisboa" --type "restaurant"

# Ver novos negócios desde uma data
leadfinder new --since "2024-01-01"

# Configurar alerta automático (cron job)
leadfinder watch --location "Porto" --type "gym" --interval daily --notify email
```

A lógica deve:

- Guardar cada negócio encontrado com timestamp `first_seen_at`
- Marcar como "novo" se não existia na base de dados
- Permitir queries por data de descoberta

4. Lead Scoring

Criar um sistema de pontuação automática (0-100) baseado em:

Critério	Pontos	Lógica
Sem website	+30	Precisa de presença digital
Poucos reviews (<10)	+20	Negócio novo ou pouca visibilidade
Rating baixo (< 4.0)	+15	Pode precisar de gestão de reputação
Sem fotos ou poucas (<5)	+15	Precisa de conteúdo visual
Price level alto (3-4)	+10	Maior budget potencial
Negócio recente (< 6 meses)	+10	Startups precisam de marketing

```bash
leadfinder score --recalculate  # Recalcular todos
leadfinder list --min-score 60  # Ver leads quentes
```

5. Exportação

```bash
```

```
# Exportar para CSV
leadfinder export --format csv --output leads.csv

# Exportar para Excel com formatação
leadfinder export --format xlsx --output leads.xlsx --template professional

# Exportar apenas leads qualificados
leadfinder export --min-score 50 --status new --format csv

# Formato para importação em CRMs
leadfinder export --format hubspot  # Colunas compatíveis com HubSpot
leadfinder export --format pipedrive
```

Colunas de exportação padrão:

- Nome do Negócio
- Endereço
- Telefone
- Website
- Email (se disponível)
- Rating
- Nº Reviews
- Lead Score
- Data de Descoberta
- Link Google Maps
- Notas

6. Comandos Úteis Adicionais

bash

```
# Estatísticas da base de dados
leadfinder stats

# Listar leads com filtros
leadfinder list --type restaurant --city Lisboa --min-score 50 --status new

# Marcar lead como contactado
leadfinder update <place_id> --status contacted --notes "Enviado email em 15/01"

# Remover duplicados
leadfinder dedupe

# Backup da base de dados
leadfinder backup --output backup_2024.db
```

Configuração (.env)

```
env

# Google API
GOOGLE_PLACES_API_KEY=your_api_key_here

# Database
DATABASE_URL=sqlite:///data/leads.db

# Defaults
DEFAULT_RADIUS=5000
DEFAULT_LANGUAGE=pt

# Export
EXPORT_DIR=./exports

# Optional: Notifications
SMTP_HOST=
SMTP_USER=
SMTP_PASS=
NOTIFY_EMAIL=
```

Google Places API - Notas Técnicas

Endpoints a usar:

1. **Text Search (New)**: `POST https://places.googleapis.com/v1/places:searchText`
   - Para pesquisas por query text

2. **Nearby Search (New)**: `POST https://places.googleapis.com/v1/places:searchNearby`
    - Para pesquisas por localização + raio

Field Masks importantes:

```
places.id,
places.displayName,
places.formattedAddress,
places.location,
places.types,
places.businessStatus,
places.nationalPhoneNumber,
places.internationalPhoneNumber,
places.websiteUri,
places.googleMapsUri,
places.rating,
places.userRatingCount,
places.priceLevel,
places.photos
```

Custos aproximados (2024):

- Text Search: $32 por 1000 requests
- Nearby Search: $32 por 1000 requests
- Place Details: $17 por 1000 requests

Rate Limits:

- Implementar rate limiting (max 10 requests/segundo recomendado)
- Implementar retry com exponential backoff
- Cache de resultados para evitar requests duplicados

Considerações Importantes

1. **Termos de Serviço**: A Google proíbe armazenamento de dados por mais de 30 dias sem refresh. Implementar lógica de refresh automático.
2. **Paginação**: A API retorna max 20 resultados por request. Usar `nextPageToken` para obter mais.
3. **Geocoding**: Se o user passar cidade/morada em vez de coordenadas, usar Geocoding API para converter.
4. **Dados de Email**: A Google Places API NÃO fornece emails. Para isso seria necessário scraping do website (outra fase).
5. **RGPD**: Os dados recolhidos são públicos, mas ter cuidado com como são usados para marketing direto em Portugal/UE.

Milestones de Desenvolvimento

Fase 1 - MVP (Prioridade)

- [ ] Setup do projeto e estrutura
- [ ] Integração básica com Google Places API
- [ ] Comando `search` funcional
- [ ] Guardar resultados em SQLite
- [ ] Exportação CSV básica

Fase 2 - Core Features

- [ ] Todos os filtros de pesquisa
- [ ] Sistema de lead scoring
- [ ] Deteção de novos negócios
- [ ] Comando `list` com filtros
- [ ] Exportação Excel formatada

Fase 3 - Polish

- [ ] Rate limiting e error handling robusto
- [ ] Comandos de gestão (update, dedupe, backup)
- [ ] Estatísticas e relatórios
- [ ] Templates de exportação para CRMs

Fase 4 - Optional

- [ ] Interface web com Streamlit/FastAPI
- [ ] Notificações por email
- [ ] Scraping de emails dos websites
- [ ] Integração direta com CRMs via API

Como Começar

1. Criar ambiente virtual e instalar dependências
2. Configurar `.env` com API key do Google
3. Implementar o cliente da Google Places API primeiro
4. Testar com uma pesquisa simples
5. Depois adicionar database e ir expandindo

Pronto para começar! Pergunta-me o que precisares ao longo do desenvolvimento.