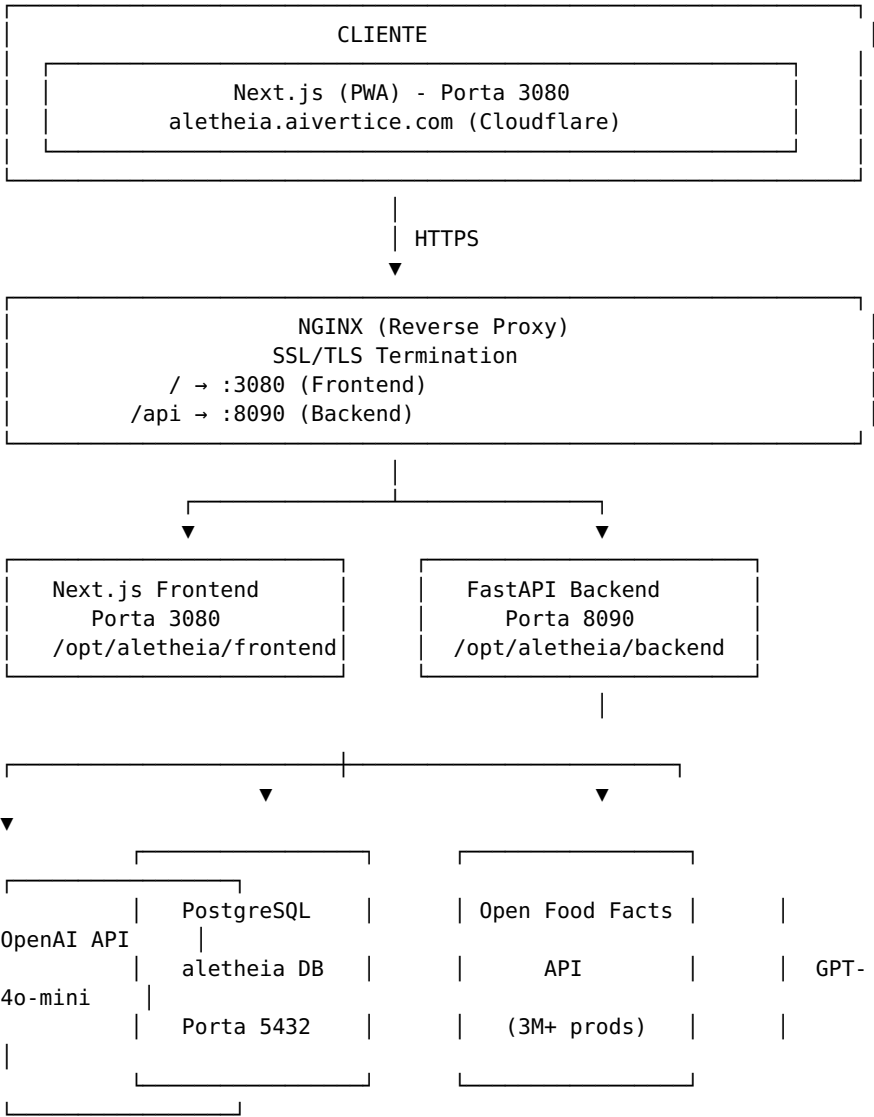


ALETHEIA - Manual Técnico

Arquitetura Geral



Stack Tecnológico

Camada	Tecnologia	Versão
Frontend	Next.js (React)	14.x
Backend	FastAPI (Python)	0.100+
Banco de Dados	PostgreSQL	15+
IA	OpenAI GPT-4o-mini	-

Process Manager	PM2	5.x
Proxy/SSL	Nginx	1.24+
CDN/DNS	Cloudflare	-

□ Deploy e Infraestrutura

Estrutura de Diretórios

```

/opt/aletheia/
├── backend/
│   ├── main.py
│   ├── requirements.txt
│   ├── .env
│   └── ...
├── frontend/
│   ├── package.json
│   ├── .env.local
│   ├── public/
│   │   └── sw.js (Service Worker)
│   └── ...
└── docs/

```

PM2 - Gerenciamento de Processos

Processos ativos: - aletheia-backend - FastAPI (porta 8090) - aletheia-frontend - Next.js (porta 3080)

Comandos úteis:

```

# Status dos processos
pm2 status

# Logs em tempo real
pm2 logs aletheia-backend
pm2 logs aletheia-frontend

# Reiniciar processos
pm2 restart aletheia-backend
pm2 restart aletheia-frontend

# Reiniciar tudo
pm2 restart all

# Salvar configuração
pm2 save

```

Nginx - Configuração

Arquivo: /etc/nginx/sites-available/aletheia

```

server {
    listen 443 ssl http2;
    server_name aletheia.aivertice.com;

```

```

    ssl_certificate
/etc/letsencrypt/live/aletheia.aivertice.com/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/aletheia.aivertice.com/privkey.pem;

# Frontend
location / {
    proxy_pass http://127.0.0.1:3080;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}

# Backend API
location /api {
    proxy_pass http://127.0.0.1:8090;
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
}

server {
    listen 80;
    server_name aletheia.aivertice.com;
    return 301 https://$server_name$request_uri;
}

```

SSL/TLS

- **Provedor:** Let's Encrypt (Certbot)
- **Renovação:** Automática via cron
- **CDN:** Cloudflare (SSL Full Strict)

```

# Renovar certificado manualmente
sudo certbot renew

```

```

# Verificar certificado
sudo certbot certificates

```

Banco de Dados

Conexão

Host: localhost
 Porta: 5432
 Database: aletheia
 User: aletheia
 Password: Aletheia2026!

Connection String:

postgresql://aletheia:Aletheia2026!@localhost:5432/aletheia

Tabelas Principais

users

```
CREATE TABLE users (  
  id SERIAL PRIMARY KEY,  
  email VARCHAR(255) UNIQUE NOT NULL,  
  password_hash VARCHAR(255) NOT NULL,  
  name VARCHAR(255),  
  plan VARCHAR(50) DEFAULT 'free', -- 'free' ou 'premium'  
  scans_today INTEGER DEFAULT 0,  
  last_scan_date DATE,  
  created_at TIMESTAMP DEFAULT NOW(),  
  updated_at TIMESTAMP DEFAULT NOW()  
);
```

products

```
CREATE TABLE products (  
  id SERIAL PRIMARY KEY,  
  barcode VARCHAR(50) UNIQUE NOT NULL,  
  name VARCHAR(255),  
  brand VARCHAR(255),  
  categories TEXT,  
  ingredients TEXT,  
  nutrition_data JSONB,  
  image_url TEXT,  
  source VARCHAR(50), -- 'openfoodfacts', 'manual'  
  created_at TIMESTAMP DEFAULT NOW(),  
  updated_at TIMESTAMP DEFAULT NOW()  
);
```

scans

```
CREATE TABLE scans (  
  id SERIAL PRIMARY KEY,  
  user_id INTEGER REFERENCES users(id),  
  product_id INTEGER REFERENCES products(id),  
  barcode VARCHAR(50),  
  score INTEGER, -- 0-100  
  analysis JSONB, -- Análise completa da IA  
  recipe TEXT, -- Receita sugerida  
  scanned_at TIMESTAMP DEFAULT NOW()  
);
```

Comandos Úteis

```
# Conectar ao banco  
psql -U aletheia -d aletheia -h localhost  
  
# Backup  
pg_dump -U aletheia -d aletheia > backup_$(date +%Y%m%d).sql  
  
# Restore  
psql -U aletheia -d aletheia < backup_20260210.sql
```

□ APIs e Endpoints

Autenticação

POST /api/auth/register

Registra novo usuário.

Request:

```
{
  "email": "usuario@email.com",
  "password": "senha123",
  "name": "Nome do Usuário"
}
```

Response (201):

```
{
  "id": 1,
  "email": "usuario@email.com",
  "name": "Nome do Usuário",
  "plan": "free",
  "token": "eyJhbGciOiJIUzI1NiIs..."
}
```

POST /api/auth/login

Autentica usuário existente.

Request:

```
{
  "email": "usuario@email.com",
  "password": "senha123"
}
```

Response (200):

```
{
  "token": "eyJhbGciOiJIUzI1NiIs...",
  "user": {
    "id": 1,
    "email": "usuario@email.com",
    "name": "Nome do Usuário",
    "plan": "free",
    "scans_today": 2
  }
}
```

Scans

POST /api/scan

Analisa um produto pelo código de barras.

Headers:

Authorization: Bearer <token>

Request:

```
{
  "barcode": "7891000100103"
}
```

Response (200):

```
{
  "id": 123,
  "product": {
    "barcode": "7891000100103",
    "name": "Leite Condensado",
    "brand": "Moça",
    "image_url": "https://..."
  },
  "score": 25,
  "classification": "Péssimo",
  "analysis": {
    "summary": "Produto com alto teor de açúcar...",
    "positives": ["Fonte de cálcio"],
    "negatives": ["Alto teor de açúcar", "Calorias elevadas"],
    "additives": [],
    "nutrition": {
      "calories": 321,
      "sugar": 55,
      "sodium": 128
    }
  },
  "recipe": {
    "title": "Leite condensado caseiro saudável",
    "ingredients": ["1 litro de leite desnatado", "..."],
    "instructions": "..."
  }
}
```

Erros: - 403: Limite de scans atingido (plano free) - 404: Produto não encontrado - 500: Erro na análise

Histórico

GET /api/history

Lista histórico de scans do usuário.

Headers:

Authorization: Bearer <token>

Query Params: - limit (opcional): Número de resultados (default: 20) - offset (opcional): Paginação

Response (200):

```
{
  "total": 45,
  "scans": [
    {
      "id": 123,
      "barcode": "7891000100103",
      "product_name": "Leite Condensado",
      "score": 25,
      "scanned_at": "2026-02-10T14:30:00Z"
    },
    ...
  ]
}
```

GET /api/history/{id}

Retorna detalhes de um scan específico.

Response (200):

```
{
  "id": 123,
  "product": { ... },
  "score": 25,
  "analysis": { ... },
  "recipe": { ... },
  "scanned_at": "2026-02-10T14:30:00Z"
}
```

□ Integrações

Open Food Facts API

Base de dados aberta com mais de 3 milhões de produtos.

Endpoint:

<https://world.openfoodfacts.org/api/v2/product/{barcode}.json>

Exemplo:

```
import requests

def get_product(barcode: str):
    url =
f"https://world.openfoodfacts.org/api/v2/product/{barcode}.json"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        if data.get("status") == 1:
            return data["product"]
    return None
```

OpenAI GPT-4o-mini

Usamos o modelo GPT-4o-mini para análise inteligente dos ingredientes.

Uso:

```
from openai import OpenAI

client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

def analyze_product(product_data: dict) -> dict:
    prompt = f"""
    Analise este produto alimentício e forneça:
    1. Score de saúde (0-100)
    2. Pontos positivos
    3. Pontos negativos
    4. Análise dos aditivos
    5. Sugestão de receita saudável alternativa

    Produto: {product_data['name']}
    Ingredientes: {product_data['ingredients']}
    Valores nutricionais: {product_data['nutrition']}
    """

    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[{"role": "user", "content": prompt}],
        response_format={"type": "json_object"}
    )

    return json.loads(response.choices[0].message.content)
```

❑ PWA / Service Worker

O ALETHEIA é uma Progressive Web App (PWA), permitindo: - Instalação na home screen - Funcionamento parcial offline - Notificações push (futuro)

Service Worker

Arquivo: /opt/aletheia/frontend/public/sw.js

```
const CACHE_NAME = 'aletheia-v1';
const urlsToCache = [
    '/',
    '/scan',
    '/history',
    '/offline.html'
];

self.addEventListener('install', (event) => {
    event.waitUntil(
        caches.open(CACHE_NAME)
            .then((cache) => cache.addAll(urlsToCache))
    );
});

self.addEventListener('fetch', (event) => {
    event.respondWith(
```



```
        caches.match(event.request)
          .then((response) => response || fetch(event.request))
    );
  });
```

Manifest

Arquivo: /opt/aletheia/frontend/public/manifest.json

```
{
  "name": "ALETHEIA",
  "short_name": "ALETHEIA",
  "description": "Scanner de rótulos com IA",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#22c55e",
  "icons": [
    {
      "src": "/icon-192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icon-512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

⚙ Variáveis de Ambiente

Backend (.env)

```
# Database
DATABASE_URL=postgresql://aletheia:Aletheia2026!@localhost:5432/aletheia

# OpenAI
OPENAI_API_KEY=sk-...

# JWT
JWT_SECRET=sua-chave-secreta-muito-longa
JWT_ALGORITHM=HS256
JWT_EXPIRATION_HOURS=24

# App
APP_ENV=production
DEBUG=false
CORS_ORIGINS=https://aletheia.aivertice.com
```

Frontend (.env.local)

```
NEXT_PUBLIC_API_URL=https://aletheia.aivertice.com/api
```

NEXT_PUBLIC_APP_NAME=ALETHEIA

📊 Monitoramento

PM2

```
# Dashboard em tempo real
pm2 monit

# Status detalhado
pm2 show aletheia-backend

# Métricas
pm2 info aletheia-backend
```

Logs

```
# Logs do backend
tail -f ~/.pm2/logs/aletheia-backend-out.log
tail -f ~/.pm2/logs/aletheia-backend-error.log

# Logs do frontend
tail -f ~/.pm2/logs/aletheia-frontend-out.log

# Logs do Nginx
tail -f /var/log/nginx/access.log
tail -f /var/log/nginx/error.log

# Logs do PostgreSQL
tail -f /var/log/postgresql/postgresql-15-main.log
```

🗄 Backup e Manutenção

Backup Automático (Cron)

```
# Editar crontab
crontab -e

# Adicionar backup diário às 3h
0 3 * * * pg_dump -U aletheia -d aletheia >
/opt/aletheia/backups/backup_$(date +%Y%m%d).sql

# Limpar backups antigos (manter últimos 30 dias)
0 4 * * * find /opt/aletheia/backups -name "*.sql" -mtime +30 -
delete
```

Manutenção do Banco

```
-- Vacuum e análise (rodar semanalmente)
VACUUM ANALYZE;

-- Verificar tamanho das tabelas
```

```
SELECT
    relname as table,
    pg_size_pretty(pg_total_relation_size(relid)) as size
FROM pg_catalog.pg_statio_user_tables
ORDER BY pg_total_relation_size(relid) DESC;
```

Atualizações

```
# Backend
cd /opt/aletheia/backend
git pull
pip install -r requirements.txt
pm2 restart aletheia-backend

# Frontend
cd /opt/aletheia/frontend
git pull
npm install
npm run build
pm2 restart aletheia-frontend
```

☐ Troubleshooting

Problema: Backend não inicia

```
# Verificar logs
pm2 logs aletheia-backend --lines 50

# Verificar porta em uso
lsof -i :8090

# Testar manualmente
cd /opt/aletheia/backend
python -m uvicorn main:app --host 0.0.0.0 --port 8090
```

Problema: Erro de conexão com banco

```
# Verificar se PostgreSQL está rodando
sudo systemctl status postgresql

# Testar conexão
psql -U aletheia -d aletheia -h localhost -c "SELECT 1"
```

Problema: Certificado SSL expirado

```
# Renovar
sudo certbot renew

# Reiniciar Nginx
sudo systemctl restart nginx
```

Problema: Open Food Facts não responde

```
# Testar API diretamente
```

```
curl
"https://world.openfoodfacts.org/api/v2/product/7891000100103.json"

# Verificar rate limiting (máx 100 req/min)
```

☐ Contatos Técnicos

- **Infraestrutura:** infra@aivertice.com
 - **Desenvolvimento:** dev@aivertice.com
-

Última atualização: Fevereiro 2026