CARONTE — Manual Técnico v1.0

> 🚣 "O barqueiro que guia famílias pelo rio burocrático pós-óbito"

---

1. Visão Geral da Arquitetura

O CARONTE é uma aplicação web full-stack composta por:

Camada	Tecnologia	Porta Padrão
Frontend	Next.js 14 (App Router) + React 18 + Tailwind CSS	3102
Backend	FastAPI (Python 3.12) + SQLAlchemy Async	8102
Banco de Dados	PostgreSQL (produção) / SQLite (desenvolvimento)	5432
Proxy Reverso	Nginx + Cloudflare	80/443
Process Manager	PM2	—

Diagrama de Arquitetura

```
[Cloudflare CDN/DNS]
        ↓
    [Nginx :80]
     ├── /api/*  → FastAPI :8102
     └── /*      → Next.js :3102
```

```
                    ↓
              [PostgreSQL :5432]
```

---

2. Estrutura de Pastas

```
caronte/
├── backend/
│   ├── app/
│   │   ├── main.py              # Entry point FastAPI
│   │   ├── __init__.py
│   │   ├── api/v1/              # Rotas da API
│   │   │   ├── auth.py          # Autenticação (registro, login, /me)
│   │   │   ├── familias.py      # CRUD famílias, membros, falecidos
│   │   │   ├── checklist.py     # Checklist burocrático
│   │   │   ├── beneficios.py    # Scanner de benefícios
│   │   │   ├── documentos.py    # Geração de documentos PDF
│   │   │   └── dashboard.py     # Dashboard consolidado
│   │   ├── core/
│   │   │   ├── config.py        # Configurações (Pydantic Settings)
│   │   │   ├── database.py      # Engine SQLAlchemy async
│   │   │   └── security.py      # JWT, bcrypt, OAuth2
│   │   ├── models/              # Modelos SQLAlchemy
│   │   │   ├── usuario.py
│   │   │   ├── familia.py       # Familia + MembroFamilia
│   │   │   ├── falecido.py
│   │   │   ├── checklist.py
│   │   │   ├── beneficio.py
│   │   │   └── documento.py
│   │   ├── schemas/
│   │   │   └── schemas.py       # Pydantic schemas (request/response)
│   │   ├── services/
│   │   │   ├── checklist_engine.py    # Geração automática de checklist
│   │   │   ├── beneficio_scanner.py   # Scanner de benefícios elegíveis
│   │   │   └── document_generator.py  # Geração de PDFs (ReportLab)
│   │   └── data/
│   │       └── checklist_templates.json  # Templates de checklist
│   ├── uploads/                 # PDFs gerados
│   ├── requirements.txt
│   └── seed_data.py
├── frontend/
│   ├── src/
│   │   ├── app/
```

```
│   │   │   ├── layout.js        # Layout global
│   │   │   ├── page.js          # Página inicial
│   │   │   ├── login/page.js
│   │   │   ├── registro/page.js
│   │   │   ├── dashboard/page.js
│   │   │   └── familia/[id]/
│   │   │       ├── page.js          # Detalhe da família
│   │   │       ├── checklist/page.js
│   │   │       ├── beneficios/page.js
│   │   │       └── documentos/page.js
│   │   ├── components/
│   │   │   └── Sidebar.js
│   │   └── lib/
│   │       └── api.js           # Cliente HTTP para a API
│   ├── package.json
│   ├── tailwind.config.js
│   └── postcss.config.js
├── docs/                        # Documentação
├── start-backend.sh
└── start-frontend.sh
```

---

3. Modelos de Dados

3.1 Usuario

Campo	Tipo	Descrição
id	Integer (PK)	Auto-increment
nome	String	Nome completo
email	String (unique)	Email de login
senha_hash	String	Hash bcrypt
telefone	String (nullable)	Telefone
created_at	DateTime	Auto

3.2 Familia

Campo	Tipo	Descrição
id	Integer (PK)	Auto-increment
nome	String	Nome da família
usuario_id	Integer (FK)	Usuário responsável
created_at	DateTime	Auto

3.3 MembroFamilia

Campo	Tipo	Descrição
id	Integer (PK)	Auto-increment
familia_id	Integer (FK)	Família
nome	String	Nome do membro
parentesco	String	Grau de parentesco
cpf, telefone, email	String (nullable)	Dados de contato

3.4 Falecido

Campo	Tipo	Descrição
id	Integer (PK)	Auto-increment
familia_id	Integer (FK)	Família
nome	String	Nome completo

Campo	Tipo	Descrição
cpf	String (nullable)	CPF
data_nascimento	Date (nullable)	Nascimento
data_obito	Date	Data do óbito
causa_obito	String (nullable)	Causa
tipo_vinculo	String	empregado, aposentado, autonomo, servidor
empregador	String (nullable)	Nome do empregador
tinha_carteira_assinada	Integer	0/1
tinha_fgts	Integer	0/1
era_aposentado	Integer	0/1
tinha_seguro_vida	Integer	0/1
tinha_imoveis	Integer	0/1
tinha_veiculos	Integer	0/1
salario_estimado	Float (nullable)	Último salário

3.5 ChecklistItem

Campo	Tipo	Descrição
id	Integer (PK)	Auto-increment
familia_id	Integer (FK)	Família

Campo	Tipo	Descrição
falecido_id	Integer (FK)	Falecido
titulo	String	Título da tarefa
descricao	Text (nullable)	Descrição detalhada
fase	String	imediato, primeira_semana, 30_dias, 60_dias
categoria	String	certidoes, inss, fgts, inventario...
status	String	pendente, andamento, concluido
prazo_dias	Integer (nullable)	Prazo em dias
ordem	Integer	Ordem de execução
dependencia_id	Integer (nullable)	Item que precede

3.6 Beneficio

Campo	Tipo	Descrição
id	Integer (PK)	Auto-increment
familia_id	Integer (FK)	Família
falecido_id	Integer (FK)	Falecido
tipo	String	fgts, pis, pensao_morte, seguro_vida, rescisao
nome	String	Nome do benefício
descricao	String (nullable)	Descrição
status	String	identificado, em_processo, sacado
valor_estimado	Float (nullable)	Valor estimado

Campo	Tipo	Descrição
valor_sacado	Float (nullable)	Valor efetivamente sacado

3.7 Documento

Campo	Tipo	Descrição
id	Integer (PK)	Auto-increment
familia_id	Integer (FK)	Família
falecido_id	Integer (FK, nullable)	Falecido
tipo	String	procuracao, requerimento_fgts, peticao_pensao
nome	String	Nome do documento
arquivo_path	String (nullable)	Caminho do PDF
status	String	gerado, enviado, aprovado

---

4. API — Endpoints

Base URL: `/api/v1`

4.1 Autenticação

Método	Endpoint	Descrição	Auth
POST	`/auth/registro`	Criar conta	❌
POST	`/auth/login`	Login (OAuth2 form) → JWT	❌

Método	Endpoint	Descrição	Auth
GET	`/auth/me`	Dados do usuário logado	✅

4.2 Famílias

Método	Endpoint	Descrição	Auth
GET	`/familias/`	Listar famílias do usuário	✅
POST	`/familias/`	Criar família	✅
GET	`/familias/{id}`	Detalhe da família	✅
POST	`/familias/{id}/membros`	Adicionar membro	✅
GET	`/familias/{id}/membros`	Listar membros	✅
POST	`/familias/{id}/falecido`	Registrar falecido (gera checklist + benefícios)	✅
GET	`/familias/{id}/falecidos`	Listar falecidos	✅

4.3 Checklist

Método	Endpoint	Descrição	Auth
GET	`/familias/{id}/checklist/`	Listar itens do checklist	✅
PUT	`/familias/{id}/checklist/{item_id}`	Atualizar status	✅
GET	`/familias/{id}/checklist/proximo`	Próximo passo recomendado	✅

4.4 Benefícios

Método	Endpoint	Descrição	Auth
GET	`/familias/{id}/beneficios/`	Listar benefícios	✅
POST	`/familias/{id}/beneficios/scan`	Re-escanear benefícios	✅

4.5 Documentos

Método	Endpoint	Descrição	Auth
GET	`/familias/{id}/documentos/`	Listar documentos	✅
POST	`/familias/{id}/documentos/gerar`	Gerar PDF	✅
GET	`/familias/{id}/documentos/{doc_id}/download`	Download PDF	✅

Tipos de documento suportados: `procuracao`, `requerimento_fgts`, `peticao_pensao`

4.6 Dashboard

Método	Endpoint	Descrição	Auth
GET	`/dashboard/`	Resumo consolidado do usuário	✅

---

5. Autenticação

- **Método:** JWT Bearer Token (OAuth2 Password Flow)
- **Hash:** bcrypt via passlib
- **Algoritmo:** HS256
- **Expiração:** 1440 minutos (24h)
- **Header:** `Authorization: Bearer <token>`

---

6. Variáveis de Ambiente

Variável	Descrição	Padrão
`DATABASE_URL`	URL de conexão ao banco	`sqlite+aiosqlite:///./caronte.db`
`SECRET_KEY`	Chave secreta para JWT	`caronte-secret-key-change-in-production`
`ALGORITHM`	Algoritmo JWT	`HS256`
`ACCESS_TOKEN_EXPIRE_MINUTES`	Expiração do token	`1440`
`UPLOAD_DIR`	Diretório de uploads	`./uploads`

Produção (PostgreSQL):

```
DATABASE_URL=postgresql+asyncpg://caronte:Caronte2026!@localhost:5432/caronte
```

> ⚠️ Para PostgreSQL, adicionar `asyncpg` ao requirements.txt e usar driver `postgresql+asyncpg://`

---

7. Requisitos de Sistema

Backend

- Python 3.12+
- Dependências: FastAPI, SQLAlchemy, aiosqlite/asyncpg, python-jose, passlib, reportlab, pydantic-settings

Frontend

- Node.js 18+
- Next.js 14, React 18, Tailwind CSS, lucide-react

Produção

- Ubuntu 22.04+ / Debian 12+
- PostgreSQL 15+
- Nginx
- PM2 (Node.js process manager)
- Cloudflare (DNS + CDN)

---

8. Deploy de Produção

8.1 Backend

```
cd /opt/caronte/backend
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
pip install asyncpg  # driver PostgreSQL

# Criar .env
echo 'DATABASE_URL=postgresql+asyncpg://caronte:Caronte2026!@localhost:5432/caronte' > .env
echo 'SECRET_KEY=<gerar-chave-segura>' >> .env

# Iniciar com PM2
pm2 start "source .venv/bin/activate && uvicorn app.main:app --host 0.0.0.0 --port 8102" --r
```

8.2 Frontend

```
cd /opt/caronte/frontend
npm install
```

```
npm run build   # NUNCA usar next dev em produção
pm2 start "npm start -- -p 3102" --name caronte-frontend
```

8.3 Nginx

```
server {
    listen 80;
    server_name caronte.aivertice.com;

    location /api/ {
        proxy_pass http://127.0.0.1:8102/;
        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;
    }

    location / {
        proxy_pass http://127.0.0.1:3102;
        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;
    }
}
```

---

9. Serviços Inteligentes

9.1 Checklist Engine

Ao registrar um falecido, o sistema gera automaticamente um checklist burocrático personalizado baseado no perfil (tipo de vínculo, tinha FGTS, era aposentado, etc.). Os templates são carregados de `checklist_templates.json` e filtrados por condições.

9.2 Benefício Scanner

Analisa o perfil do falecido e identifica automaticamente benefícios elegíveis: - **FGTS** (se tinha FGTS) — estimativa: 3.5x salário - **PIS/PASEP** (se tinha carteira assinada) — estimativa: 0.8x salário - **Pensão por Morte** (sempre) — estimativa: 12x salário - **Seguro de Vida** (se tinha) — estimativa: 24x salário - **Verbas Rescisórias** (se tinha carteira assinada) — estimativa: 2x salário

9.3 Document Generator

Gera PDFs profissionais via ReportLab: - **Procuração** — representação perante órgãos - **Requerimento FGTS** — saque por falecimento (Lei 8.036/90) - **Petição Pensão por Morte** — requerimento INSS (Lei 8.213/91)

Documento gerado em 08/02/2026 — CARONTE v1.0