ASTERISK ADMIN

Sistema de Gerenciamento VoIP

DOCUMENTACAO TECNICA COMPLETA

Versao:	1.0.0
Data:	11/01/2026
Autor:	Asterisk Admin Team
Licenca:	Proprietaria

SUMARIO

1. VISAO GERAL DO PROJETO

O Asterisk Admin e um sistema completo de gerenciamento VoIP desenvolvido para administrar centrais telefonicas baseadas em Asterisk. O sistema permite gerenciar clientes, DIDs, provedores, gateways, rotas, ramais e tarifas atraves de uma interface web moderna e intuitiva.

1.1 Principais Funcionalidades

- Dashboard com metricas em tempo real
- Gerenciamento de Clientes (Trunk e Ramal)
- Inventario de DIDs com alocacao
- Cadastro de Provedores de telefonia
- Configuracao de Gateways SIP
- Rotas de saida com prioridade
- Ramais para clientes tipo extensao
- Tarifacao com calculo de margem
- Relatorios de CDR e DIDs
- Conferencia entre numeros
- Sincronizacao automatica com Asterisk

2. ARQUITETURA DO SISTEMA

O sistema segue uma arquitetura de tres camadas: Frontend (React), Backend (FastAPI) e Banco de Dados (PostgreSQL), com integracao direta ao Asterisk via arquivos de configuracao.

2.1 Diagrama de Arquitetura

```
[Browser] <--HTTPS--> [Nginx] <---> [React SPA]
|
v
[FastAPI Backend]
|
+--------------+--------------+
| | |
v v v
[PostgreSQL] [Asterisk] [Config Files]
```

2.2 Componentes

Componente	Tecnologia	Porta	Funcao
Frontend	React + Vite	80/443	Interface do usuario
Backend	FastAPI + Uvicorn	8000	API REST
Banco de Dados	PostgreSQL 15+	5432	Persistencia
Web Server	Nginx	80/443	Proxy reverso
PBX	Asterisk 20	5060	Telefonia SIP

3. STACK TECNOLOGICO

3.1 Backend

Tecnologia	Versao	Uso
Python	3.11+	Linguagem principal
FastAPI	0.104+	Framework web async
SQLAlchemy	2.0+	ORM async
Pydantic	2.0+	Validacao de dados
Uvicorn	0.24+	Servidor ASGI
python-jose	3.3+	JWT tokens
bcrypt	4.0+	Hash de senhas
loguru	0.7+	Logging
asyncpg	0.29+	Driver PostgreSQL async

3.2 Frontend

Tecnologia	Versao	Uso
React	18+	Framework UI
Vite	5+	Build tool
TailwindCSS	3+	Estilizacao
React Query	5+	Gerenciamento de estado
Axios	1.6+	Cliente HTTP
Lucide React	0.29+	Icones
React Router	6+	Roteamento SPA

3.3 Infraestrutura

Tecnologia	Uso
Debian 12 / Ubuntu 24	Sistema operacional
Nginx	Proxy reverso e arquivos estaticos
PostgreSQL 15+	Banco de dados relacional
Asterisk 20	PBX SIP com PJSIP
Systemd	Gerenciamento de servicos

4. ESTRUTURA DO BANCO DE DADOS

O banco de dados PostgreSQL armazena todas as informacoes do sistema. Utiliza UUIDs como chaves primarias para melhor distribuicao e seguranca.

4.1 Tabelas Principais

Tabela	Descricao	Campos Principais
users	Usuarios do sistema	id, username, password_hash, role
customers	Clientes (trunk/ramal)	id, code, name, type, trunk_ip
providers	Provedores de telefonia	id, name, type, cost_per_minute
gateways	Gateways SIP	id, name, ip_address, port, codecs
dids	Numeros DID	id, number, provider_id, gateway_id, status
customer_dids	Alocacao DID-Cliente	customer_id, did_id, destination
routes	Rotas de saida	id, name, pattern, gateway_id, priority
extensions	Ramais	id, extension, secret, customer_id
tariffs	Tarifas	id, pattern, cost_per_minute, price_per_minute
cdr	Registros de chamadas	id, src, dst, duration, billsec, price

4.2 Relacionamentos

• customers 1:N extensions (cliente pode ter varios ramais)
• customers N:M dids (via customer_dids)
• providers 1:N dids (provedor fornece DIDs)
• providers 1:N gateways (provedor tem gateways)
• gateways 1:N routes (gateway usado em rotas)
• gateways 1:N dids (gateway de entrada do DID)

5. BACKEND - API FASTAPI

5.1 Estrutura de Diretorios

```
backend/
■■■ app/
■   ■■■ __init__.py
■   ■■■ main.py              # Aplicacao FastAPI
■   ■■■ api/                  # Routers/Endpoints
■   ■   ■■■ auth.py          # Autenticacao
■   ■   ■■■ customers.py     # CRUD clientes
■   ■   ■■■ dids.py          # CRUD DIDs
■   ■   ■■■ providers.py     # CRUD provedores
■   ■   ■■■ gateways.py      # CRUD gateways
■   ■   ■■■ routes.py        # CRUD rotas
■   ■   ■■■ extensions.py    # CRUD ramais
■   ■   ■■■ tariffs.py       # CRUD tarifas
■   ■   ■■■ reports.py       # Relatorios
■   ■   ■■■ dashboard.py     # Estatisticas
■   ■   ■■■ conference.py    # Conferencia
■   ■■■ core/
■   ■   ■■■ config.py        # Configuracoes
■   ■   ■■■ database.py      # Conexao DB
■   ■   ■■■ security.py      # JWT/Auth
■   ■■■ models/              # Modelos SQLAlchemy
■   ■■■ schemas/             # Schemas Pydantic
■   ■■■ services/
■       ■■■ asterisk.py      # Integracao Asterisk
■■■ requirements.txt
■■■ .env
```

5.2 Autenticacao

O sistema utiliza JWT (JSON Web Tokens) para autenticacao. O token e gerado no login e deve ser enviado no header Authorization de todas as requisicoes protegidas.

```
Header: Authorization: Bearer [token]
Expiracao: 24 horas
Algoritmo: HS256
```

6. FRONTEND - REACT

6.1 Estrutura de Diretorios

```
frontend/
■■■ src/
■   ■■■ main.jsx              # Entry point
■   ■■■ App.jsx               # Rotas principais
■   ■■■ components/
■   ■   ■■■ Sidebar.jsx       # Menu lateral
■   ■■■ pages/
■   ■   ■■■ Login.jsx
■   ■   ■■■ Dashboard.jsx
■   ■   ■■■ Customers.jsx
■   ■   ■■■ DIDs.jsx
■   ■   ■■■ Providers.jsx
■   ■   ■■■ Gateways.jsx
■   ■   ■■■ Routes.jsx
■   ■   ■■■ Extensions.jsx
■   ■   ■■■ Tariffs.jsx
■   ■   ■■■ Reports.jsx
■   ■   ■■■ Conference.jsx
■   ■   ■■■ Settings.jsx
■   ■■■ services/
■   ■   ■■■ api.js           # Cliente Axios
■   ■■■ styles/
■       ■■■ globals.css       # TailwindCSS
■■■ index.html
■■■ package.json
■■■ vite.config.js
■■■ tailwind.config.js
■■■ postcss.config.js
```

6.2 Gerenciamento de Estado

O frontend utiliza React Query (TanStack Query) para gerenciamento de estado do servidor. Isso proporciona cache automatico, revalidacao em background, e estados de loading/error.

7. INTEGRACAO COM ASTERISK

O sistema sincroniza automaticamente as configuracoes com o Asterisk atraves de arquivos de configuracao PJSIP e Dialplan.

7.1 Arquivos Gerados

Arquivo	Modulo	Conteudo
pjsip_gateways.conf	Gateways	Endpoints SIP dos gateways
pjsip_customer_trunks.conf	Clientes Trunk	Endpoints dos clientes trunk
pjsip_extensions.conf	Ramais	Endpoints dos ramais
extensions_routes.conf	Rotas	Dialplan de saida
extensions_dids.conf	DIDs	Dialplan de entrada

7.2 Processo de Sincronizacao

1. Usuario cria/edita/exclui registro no painel
2. API salva no banco de dados
3. Servico AsteriskService gera arquivo de configuracao
4. Faz backup do arquivo anterior
5. Escreve novo arquivo em /etc/asterisk/
6. Executa reload no Asterisk (pjsip reload ou dialplan reload)

7.3 Exemplo de Configuracao Gerada

```
; === Gateway: GW_Vivo_SP ===
[GW_Vivo_SP]
type=endpoint
context=from-trunk
disallow=all
allow=alaw,ulaw
aors=GW_Vivo_SP

[GW_Vivo_SP]
type=aor
contact=sip:200.200.200.200:5060

[GW_Vivo_SP]
type=identify
endpoint=GW_Vivo_SP
match=200.200.200.200
```

8. ARQUIVOS DE CONFIGURACAO

8.1 Backend (.env)

```
DATABASE_URL=postgresql+asyncpg://asterisk:senha@localhost/asterisk_admin
SECRET_KEY=sua-chave-secreta-aqui
ASTERISK_CONFIG_PATH=/etc/asterisk
ASTERISK_SPOOL_PATH=/var/spool/asterisk/outgoing
```

8.2 Nginx

```
server {
listen 80;
server_name _;

location / {
root /opt/asterisk-admin/frontend/dist;
try_files $uri $uri/ /index.html;
}

location /api {
proxy_pass http://127.0.0.1:8000;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
}
}
```

8.3 Systemd Service

```
[Unit]
Description=Asterisk Admin API
After=network.target postgresql.service

[Service]
Type=simple
User=root
WorkingDirectory=/opt/asterisk-admin/backend
ExecStart=/opt/asterisk-admin/backend/venv/bin/uvicorn app.main:app --host 0.0.0.0
--port 8000
Restart=always

[Install]
WantedBy=multi-user.target
```

9. DEPLOY E INFRAESTRUTURA

9.1 Requisitos Minimos

Recurso	Minimo	Recomendado
CPU	2 vCPUs	4+ vCPUs
RAM	2 GB	4+ GB
Disco	20 GB SSD	50+ GB SSD
SO	Debian 12 / Ubuntu 24	Debian 12

9.2 Processo de Deploy

1. Instalar dependencias do sistema (PostgreSQL, Nginx, Python, Node.js, Asterisk)
2. Criar banco de dados e usuario
3. Extrair pacote em /opt/asterisk-admin/
4. Executar migracoes SQL
5. Criar e ativar virtualenv Python
6. Instalar dependencias pip
7. Instalar dependencias npm e fazer build
8. Configurar Nginx
9. Criar service systemd
10. Adicionar includes no Asterisk
11. Iniciar servicos

9.3 Capacidade Estimada

Servidor	Canais Simultaneos	CPS
2 vCPU / 2GB	60-80	10-15
4 vCPU / 8GB	150-180	25-30
8 vCPU / 16GB	300-400	50-60
32 vCPU / 64GB	1500-2000	150-200

10. ENDPOINTS DA API

Metodo	Endpoint	Descricao
POST	/api/v1/auth/login	Autenticacao
GET	/api/v1/dashboard/stats	Estatisticas
GET/POST	/api/v1/customers/	Listar/Criar clientes
PUT/DELETE	/api/v1/customers/{id}	Editar/Excluir cliente
GET/POST	/api/v1/dids/	Listar/Criar DIDs
POST	/api/v1/dids/{id}/allocate	Alocar DID
POST	/api/v1/dids/{id}/deallocate	Desalocar DID
GET/POST	/api/v1/providers/	Listar/Criar provedores
GET/POST	/api/v1/gateways/	Listar/Criar gateways
GET/POST	/api/v1/routes/	Listar/Criar rotas
GET/POST	/api/v1/extensions/	Listar/Criar ramais
GET/POST	/api/v1/tariffs/	Listar/Criar tarifas
GET	/api/v1/reports/cdr	Relatorio CDR
POST	/api/v1/conference/	Criar conferencia

11. FLUXO DE CHAMADAS

11.1 Chamada de Entrada (DID)

1. Chamada chega no Gateway configurado
2. Asterisk recebe no contexto [from-trunk]
3. Dialplan verifica o DID chamado
4. Se DID alocado a cliente trunk: encaminha para IP do cliente
5. Se DID alocado a ramal: chama o ramal interno
6. Se DID nao configurado: toca mensagem de invalido

11.2 Chamada de Saida

1. Ramal ou Trunk do cliente origina chamada
2. Asterisk recebe no contexto [from-internal]
3. Dialplan verifica pattern das rotas por prioridade
4. Encontra rota compativel e gateway associado
5. Adiciona tech_prefix se configurado
6. Envia chamada pelo gateway

12. SEGURANCA

12.1 Medidas Implementadas

• Senhas armazenadas com hash bcrypt

• Autenticacao via JWT com expiracao

• Validacao de entrada com Pydantic

• Queries parametrizadas (SQLAlchemy)

• CORS configurado

• Proxy reverso com Nginx

12.2 Recomendacoes Adicionais

• Configurar HTTPS com certificado SSL

• Usar firewall (iptables/ufw)

• Limitar acesso SSH por IP

• Habilitar fail2ban para Asterisk

• Backup regular do banco de dados

• Monitoramento com Prometheus/Grafana

13. MANUTENCAO E TROUBLESHOOTING

13.1 Comandos Uteis

```
# Ver status do servico
systemctl status asterisk-admin

# Ver logs do backend
journalctl -u asterisk-admin -f

# Ver endpoints PJSIP
asterisk -rx "pjsip show endpoints"

# Recarregar PJSIP
asterisk -rx "pjsip reload"

# Recarregar Dialplan
asterisk -rx "dialplan reload"

# Ver canais ativos
asterisk -rx "core show channels"

# Testar conexao banco
psql -U asterisk -h localhost -d asterisk_admin
```

13.2 Problemas Comuns

Problema	Solucao
500 Internal Server Error	Verificar logs: journalctl -u asterisk-admin
Tela branca apos login	Verificar console do browser (F12)
Gateway nao aparece	Verificar pjsip_gateways.conf e dar reload
Ramal nao registra	Verificar senha e IP no pjsip_extensions.conf
Chamada nao completa	Verificar rotas e dialplan show from-internal

FIM DA DOCUMENTACAO

Documento gerado em 11/01/2026 12:04

Asterisk Admin v1.0.0