Guia Completo de Configuração - COBOL AI Engine v2.1.4

Visão Geral

O COBOL AI Engine v2.1.4 oferece configuração flexível e robusta para diferentes ambientes e necessidades. Este guia detalha todas as opções de configuração disponíveis.

Estrutura de Configuração

Arquivo Principal: config_unified.yaml

O sistema utiliza um único arquivo de configuração YAML que centraliza todas as opções:

Plain Text

```
cobol_ai_engine_v2.0.0/
├── config/
│   └── config_unified.yaml    # Configuração principal
├── main.py                    # Interface CLI
└── ...
```

Configurações de Rate Limiting

Configuração do LuzIA com Rate Limiting

YAML

```yaml
ai:
  providers:
    luzia:
      enabled: true
      client_id: "${LUZIA_CLIENT_ID:-seu_client_id}"
      client_secret: "${LUZIA_CLIENT_SECRET:-seu_client_secret}"
      auth_url: "https://login.azure.pass.santanderbr.pre.corp"
      api_url: "https://prd-api-aws.santanderbr.dev.corp/genai_services/v1"
      model: "azure-gpt-4o-mini"
      temperature: 0.1
      max_tokens: 4000
      timeout: 180

      # CONFIGURAÇÕES DE RATE LIMITING E RETRY
```

```yaml
    retry:
      max_attempts: 5              # Máximo de tentativas
      base_delay: 2.0              # Delay inicial em segundos
      max_delay: 60.0              # Delay máximo em segundos
      backoff_multiplier: 2.0      # Multiplicador exponencial
      requests_per_minute: 8       # Limite de requisições por minuto
```

Parâmetros de Rate Limiting Explicados

max_attempts

- **Padrão**: 5
- **Descrição**: Número máximo de tentativas em caso de erro
- **Valores recomendados**:
  - Produção: 5-7
  - Desenvolvimento: 3-5
  - Testes: 2-3

base_delay

- **Padrão**: 2.0 segundos
- **Descrição**: Tempo de espera inicial entre tentativas
- **Valores recomendados**:
  - Ambiente com alta carga: 3.0-5.0
  - Ambiente normal: 2.0-3.0
  - Desenvolvimento: 1.0-2.0

max_delay

- **Padrão**: 60.0 segundos
- **Descrição**: Tempo máximo de espera entre tentativas
- **Valores recomendados**:
  - Produção crítica: 120.0
  - Produção normal: 60.0
  - Desenvolvimento: 30.0

backoff_multiplier

- **Padrão**: 2.0
- **Descrição**: Multiplicador para backoff exponencial
- **Valores recomendados**:
  - Conservador: 1.5
  - Padrão: 2.0
  - Agressivo: 3.0

requests_per_minute

- **Padrão**: 8
- **Descrição**: Limite de requisições por minuto
- **Valores recomendados**:
  - Ambiente compartilhado: 5-8
  - Ambiente dedicado: 10-15
  - Desenvolvimento: 3-5

Configurações por Ambiente

Ambiente de Produção (Conservador)

YAML

```yaml
ai:
  providers:
    luzia:
      retry:
        max_attempts: 7
        base_delay: 3.0
        max_delay: 120.0
        backoff_multiplier: 2.0
        requests_per_minute: 6
```

Características:

- Mais tentativas para garantir sucesso
- Delays maiores para evitar sobrecarga
- Rate limiting conservador
- Ideal para ambiente crítico

Ambiente de Desenvolvimento (Ágil)

YAML

```yaml
ai:
  providers:
    luzia:
      retry:
        max_attempts: 3
        base_delay: 1.0
        max_delay: 30.0
        backoff_multiplier: 2.0
        requests_per_minute: 10
```

Características:

- Menos tentativas para feedback rápido
- Delays menores para agilidade
- Rate limiting mais permissivo
- Ideal para testes e desenvolvimento

Ambiente de Teste (Balanceado)

YAML

```yaml
ai:
  providers:
    luzia:
      retry:
        max_attempts: 4
        base_delay: 2.0
        max_delay: 45.0
        backoff_multiplier: 1.5
        requests_per_minute: 8
```

Características:

- Configuração equilibrada
- Adequado para validação
- Simula ambiente de produção

Configuração de Token Management

Controle de Divisão de Tokens

YAML

```yaml
performance:
  token_management:
    # CONFIGURAÇÃO GLOBAL (PADRÃO)
    enable_chunking: false           # Não dividir por padrão
    max_tokens_per_chunk: 150000     # Limite por chunk se habilitado
    overlap_tokens: 500              # Sobreposição entre chunks

    # CONFIGURAÇÕES ESPECÍFICAS POR PROVEDOR
    provider_specific:
      luzia:
        enable_chunking: false       # LuzIA: sempre completo
        max_tokens_per_request: 200000

      openai:
        enable_chunking: true        # OpenAI: pode dividir
        max_tokens_per_request: 120000

      enhanced_mock:
        enable_chunking: false       # Mock: sempre completo
        max_tokens_per_request: 500000
```

Parâmetros de Token Management

enable_chunking

- **Padrão**: false
- **Descrição**: Habilita divisão de programas grandes
- **Recomendação**: Manter false para análises completas

max_tokens_per_chunk

- **Padrão**: 150000
- **Descrição**: Tamanho máximo de cada chunk
- **Valores típicos**: 100000-200000

overlap_tokens

- **Padrão**: 500
- **Descrição**: Sobreposição entre chunks para contexto

- **Valores típicos**: 200-1000

Configuração de Provedores

Configuração Completa de Provedores

YAML

```yaml
ai:
  # PROVEDOR PRIMÁRIO
  primary_provider: "luzia"          # Provedor principal

  # CONFIGURAÇÕES GLOBAIS
  enable_fallback: true              # Habilitar fallback automático
  global_timeout: 120                # Timeout global
  global_max_tokens: 4000            # Tokens globais
  max_retries: 3                     # Retries globais (legacy)
  retry_delay: 2.0                   # Delay global (legacy)

  providers:
    # LUZIA (CORPORATIVO)
    luzia:
      enabled: true
      client_id: "${LUZIA_CLIENT_ID}"
      client_secret: "${LUZIA_CLIENT_SECRET}"
      auth_url: "https://login.azure.pass.santanderbr.pre.corp"
      api_url: "https://prd-api-aws.santanderbr.dev.corp/genai_services/v1"
      model: "azure-gpt-4o-mini"
      temperature: 0.1
      max_tokens: 4000
      timeout: 180
      retry:
        max_attempts: 5
        base_delay: 2.0
        max_delay: 60.0
        backoff_multiplier: 2.0
        requests_per_minute: 8

    # OPENAI (ALTERNATIVO)
    openai:
      enabled: true
      api_key: "${OPENAI_API_KEY}"
      model: "gpt-4"
      temperature: 0.1
      max_tokens: 4000
      timeout: 60
```

```yaml
    base_url: "https://api.openai.com/v1"

  # ENHANCED MOCK (DESENVOLVIMENTO)
  enhanced_mock:
    enabled: true
    model: "enhanced-mock-gpt-4"
    response_delay: 0.1
    enable_phasing: true
    max_tokens_per_phase: 2000
    simulate_realistic_tokens: true
    cobol_analysis_enabled: true
    code_extraction_enabled: true
    specific_responses: true
    temperature: 0.1
    max_tokens: 4000

  # BASIC (FALLBACK FINAL)
  basic:
    enabled: true
    max_tokens: 1000
    response_delay: 0.05
    include_code_analysis: true
    include_statistics: true
```

Configuração de Prompts

Personalização de Prompts

YAML

```yaml
prompts:
  # PROMPT PRINCIPAL DE ANÁLISE
  analysis_prompt: |
    Você é um especialista em análise de código COBOL com mais de 20 anos de
experiência.
    Analise o código fornecido e responda às seguintes perguntas:

    1. O que este programa faz funcionalmente?
    2. Qual é a estrutura técnica e componentes principais?
    3. Quais regras de negócio estão implementadas?
    4. Quais são os trechos de código mais relevantes?
    5. Como este programa se relaciona com outros sistemas?
    6. Quais são as considerações de performance e otimização?
    7. Quais aspectos são importantes para a manutenção?

    Responda em português brasileiro de forma clara e técnica.
```

```
  # PROMPT DE SISTEMA
  system_prompt: |
    Você é um especialista em análise de código COBOL com mais de 20 anos de
experiência.
    Responda sempre em português brasileiro de forma clara, técnica e
detalhada.

  # CONFIGURAÇÕES DE PROMPT
  max_prompt_length: 200000
  include_code_context: true
  include_program_metadata: true
```

Variáveis de Ambiente

Configuração via Variáveis de Ambiente

Bash

```
# CREDENCIAIS LUZIA
export LUZIA_CLIENT_ID="seu_client_id_aqui"
export LUZIA_CLIENT_SECRET="seu_client_secret_aqui"
export LUZIA_AUTH_URL="https://login.azure.pass.santanderbr.pre.corp"
export LUZIA_API_URL="https://prd-api-
aws.santanderbr.dev.corp/genai_services/v1"

# CREDENCIAIS OPENAI (OPCIONAL)
export OPENAI_API_KEY="sk-..."

# CREDENCIAIS DATABRICKS (OPCIONAL)
export DATABRICKS_WORKSPACE_URL="https://..."
export DATABRICKS_ACCESS_TOKEN="dapi..."

# CREDENCIAIS AWS BEDROCK (OPCIONAL)
export AWS_REGION="us-east-1"
export AWS_ACCESS_KEY_ID="AKIA..."
export AWS_SECRET_ACCESS_KEY="..."
```

Precedência de Configuração

1. **Variáveis de ambiente** (maior prioridade)
2. **Valores no YAML**
3. **Valores padrão** (menor prioridade)

Exemplos de Uso

Uso Básico com LuzIA

Bash

```bash
# Análise simples
python main.py --config config/config_unified.yaml \
               --fontes examples/fontes.txt \
               --output resultado \
               --provider luzia
```

Uso com Configuração Personalizada

Bash

```bash
# Criar arquivo de configuração personalizado
cp config/config_unified.yaml config/config_producao.yaml

# Editar configurações específicas
# ... modificar config_producao.yaml ...

# Executar com configuração personalizada
python main.py --config config/config_producao.yaml \
               --fontes meus_programas.txt \
               --output resultado_producao \
               --provider luzia
```

Uso com Fallback Automático

Bash

```bash
# Sistema tentará LuzIA, depois enhanced_mock, depois basic
python main.py --config config/config_unified.yaml \
               --fontes examples/fontes.txt \
               --output resultado
```

Monitoramento e Logs

Configuração de Logs

YAML

```yaml
logging:
  level: INFO                          # DEBUG, INFO, WARNING, ERROR
  format: "%(asctime)s - %(name)s - %(levelname)s - %(message)s"
  file: "logs/cobol_ai_engine.log"
  max_size: "10MB"
  backup_count: 5
```

Logs de Rate Limiting

Plain Text

```
2025-09-15 15:30:15 - LuziaProvider - INFO - Rate limit atingido. Aguardando
15s...
2025-09-15 15:30:30 - LuziaProvider - INFO - Tentativa 2/6
2025-09-15 15:30:32 - LuziaProvider - INFO - Rate limit detectado.
Aguardando 4s...
2025-09-15 15:30:36 - LuziaProvider - INFO - Resposta recebida com sucesso
```

Monitoramento de Performance

Plain Text

```
2025-09-15 15:30:45 - __main__ - INFO - Análises bem-sucedidas: 5/5
2025-09-15 15:30:45 - __main__ - INFO - Taxa de sucesso: 100.0%
2025-09-15 15:30:45 - __main__ - INFO - Total de tokens utilizados: 42664
2025-09-15 15:30:45 - __main__ - INFO - Tempo de processamento: 0.88s
```

Troubleshooting

Problemas Comuns e Soluções

Rate Limiting Muito Restritivo

Sintoma: Sistema muito lento, muitas esperas
Solução: Aumentar `requests_per_minute`

YAML

```yaml
retry:
  requests_per_minute: 12  # Aumentar de 8 para 12
```

Muitas Falhas de Conexão

Sintoma: Erros de timeout frequentes
Solução: Aumentar timeouts e tentativas

YAML

```yaml
timeout: 240              # Aumentar timeout
retry:
  max_attempts: 7         # Mais tentativas
  max_delay: 120.0        # Mais tempo entre tentativas
```

Análises Incompletas

Sintoma: Respostas muito curtas
Solução: Verificar configuração de tokens

YAML

```yaml
max_tokens: 8000          # Aumentar limite de tokens
temperature: 0.1          # Manter baixo para consistência
```

Configurações Avançadas

Configuração para Alto Volume

YAML

```yaml
ai:
  providers:
    luzia:
      retry:
        max_attempts: 10
        base_delay: 1.0
        max_delay: 30.0
        backoff_multiplier: 1.5
        requests_per_minute: 15
      timeout: 300
      max_tokens: 8000

performance:
  token_management:
    enable_chunking: true
```

```yaml
    max_tokens_per_chunk: 100000
    overlap_tokens: 1000
```

Configuração para Máxima Confiabilidade

YAML

```yaml
ai:
  providers:
    luzia:
      retry:
        max_attempts: 15
        base_delay: 5.0
        max_delay: 300.0
        backoff_multiplier: 1.2
        requests_per_minute: 3
      timeout: 600
      max_tokens: 4000
```

Validação de Configuração

Script de Teste

Bash

```bash
# Testar configuração
python -c "
import yaml
with open('config/config_unified.yaml', 'r') as f:
    config = yaml.safe_load(f)
print('Configuração válida!')
print(f'Provedor primário: {config[\"ai\"][\"primary_provider\"]}')
print(f'Rate limit LuzIA: {config[\"ai\"][\"providers\"][\"luzia\"]
[\"retry\"][\"requests_per_minute\"]} req/min')
"
```

Teste de Conectividade

Bash

```bash
# Testar conectividade com LuzIA
python main.py --config config/config_unified.yaml --test-connection luzia
```

Conclusão

O COBOL AI Engine v2.1.4 oferece configuração flexível e robusta para diferentes necessidades:

- **Rate limiting inteligente** para evitar sobrecarga da API
- **Sistema de retry robusto** com backoff exponencial
- **Configuração por ambiente** (produção, desenvolvimento, teste)
- **Fallback automático** entre provedores
- **Monitoramento detalhado** via logs
- **Flexibilidade total** via YAML e variáveis de ambiente

Para suporte adicional, consulte os logs detalhados ou entre em contato com a equipe de desenvolvimento.

COBOL AI Engine v2.1.4 - Guia de Configuração

Data: 15 de Setembro de 2025