

CorpLang - Plano de Implementação de Análise de Documentos

1. Extensões da Linguagem

1.1 Novos Tokens e Palavras-chave

```
python

# Novos tokens para análise documental
DOCUMENT = "DOCUMENT"
EXTRACT = "EXTRACT"
ANALYZE = "ANALYZE"
CLASSIFY = "CLASSIFY"
SUMMARIZE = "SUMMARIZE"
EXPORT = "EXPORT"
TEMPLATE = "TEMPLATE"
OCR = "OCR"
```

1.2 Nova Sintaxe Proposta

```
corplang

# Carregamento de documentos
document load contrato_imovel("documento.pdf");

# Extração de dados específicos
extract from contrato_imovel {
    campo: "valor_imovel",
    tipo: "currency",
    pattern: "R\\$ [0-9,.]+"
};

# Análise com IA
analyze contrato_imovel with gemini {
    prompt: "Extraia todas as cláusulas de rescisão",
    format: "json",
    schema: {
        "clausulas": ["string"],
        "condicoes": ["string"]
    }
};

# Classificação automática
classify documento {
    categories: ["contrato", "escritura", "certidao"],
    confidence_threshold: 0.8
};

# Geração de relatório
export analysis to pdf("relatorio_analise.pdf") {
    template: "template_pericia.html",
    data: resultado_analise
};
```

2. Estrutura de Arquivos do Projeto

```
corplang/
├── src/
│   ├── core/
│   │   ├── lexer.py
│   │   ├── parser.py
│   │   ├── interpreter.py
│   │   └── ast_nodes.py
│   ├── extensions/
│   │   ├── document_processor.py
│   │   ├── ai_integration.py
│   │   ├── ocr_handler.py
│   │   └── export_manager.py
│   ├── utils/
│   │   ├── file_utils.py
│   │   ├── validation.py
│   │   └── logging.py
│   └── cli/
│       ├── main.py
│       └── repl.py
├── templates/
│   ├── pericia_imovel.html
│   ├── analise_contrato.html
│   └── relatorio_base.html
├── examples/
│   ├── analise_escritura.cl
│   ├── pericia_automatica.cl
│   └── batch_processing.cl
├── tests/
│   ├── test_document_processor.py
│   ├── test_ai_integration.py
│   └── test_export.py
├── docs/
│   ├── language_reference.md
│   ├── api_documentation.md
│   └── examples.md
├── install/
│   ├── setup.py
│   ├── requirements.txt
│   └── install_scripts/
├── ide/
│   ├── vscode_extension/
│   ├── syntax_highlighting/
│   └── language_server/
└── README.md
```

3. Implementação das Extensões

3.1 Document Processor (document_processor.py)

python

```
class DocumentProcessor:
    def __init__(self):
        self.supported_formats = ['.pdf', '.docx', '.doc', '.txt', '.jpg', '.png']
        self.ocr_handler = OCRHandler()
        self.ai_client = AIClient()

    def load_document(self, file_path: str, doc_name: str):
        """Carrega e processa documento"""
        pass

    def extract_text(self, document):
        """Extrai texto do documento"""
        pass

    def extract_fields(self, document, field_config):
        """Extrai campos específicos usando regex ou IA"""
        pass

    def classify_document(self, document, categories):
        """Classifica documento usando IA"""
        pass
```

3.2 AI Integration (ai_integration.py)

```
python

class AIClient:
    def __init__(self, api_key: str):
        self.gemini_client = genai.GenerativeModel('gemini-pro')
        self.api_key = api_key

    def analyze_document(self, text: str, prompt: str, format_type: str = "json"):
        """Analisa documento usando Gemini"""
        pass

    def extract_structured_data(self, text: str, schema: dict):
        """Extrai dados estruturados baseado em schema"""
        pass

    def summarize_document(self, text: str, max_length: int = 500):
        """Gera resumo do documento"""
        pass
```

3.3 Export Manager (export_manager.py)

```
python
```

```
class ExportManager:
    def __init__(self):
        self.pdf_generator = PDFGenerator()
        self.template_engine = TemplateEngine()

    def export_to_pdf(self, data: dict, template_path: str, output_path: str):
        """Exporta dados para PDF usando template"""
        pass

    def export_to_json(self, data: dict, output_path: str):
        """Exporta dados para JSON"""
        pass

    def export_to_excel(self, data: dict, output_path: str):
        """Exporta dados para Excel"""
        pass
```

4. Casos de Uso Específicos

4.1 Análise de Escritura de Imóvel

corplang

```
# Carregamento do documento
document load escritura("escritura_imovel.pdf");

# OCR se necessário
if (document.needs_ocr(escritura)) {
  ocr process escritura;
}

# Extração de dados específicos
var dados_imovel = extract from escritura {
  "matricula": {
    "pattern": "Matrícula nº[0-9]+\s*(\d+)",
    "type": "number"
  },
  "valor": {
    "pattern": "R\$[0-9.]+\s*",
    "type": "currency"
  },
  "area": {
    "pattern": "(\d+[.,]?\d*)\s*m²",
    "type": "area"
  },
  "endereco": {
    "ai_prompt": "Extraia o endereço completo do imóvel",
    "type": "address"
  }
};

# Análise com IA para verificar irregularidades
var analise_juridica = analyze escritura with gemini {
  prompt: "Análise este documento e identifique possíveis irregularidades jurídicas, cláusulas problemáticas e pontos d
  format: "json",
  schema: {
    "irregularidades": ["string"],
    "clausulas_problematicas": ["string"],
    "pontos_atencao": ["string"],
    "nivel_risco": "string"
  }
};

# Geração de relatório
export analysis to pdf("relatorio_escritura.pdf") {
  template: "template_escritura.html",
  data: {
    "dados_imovel": dados_imovel,
    "analise_juridica": analise_juridica,
    "data_analise": current_date(),
    "usuario": _global_user
  }
};
```



4.2 Perícia Automática

```
# Processamento em lote
var documentos_pericia = ["foto1.jpg", "foto2.jpg", "laudo_anterior.pdf"];

function processar_pericia(docs) {
    var resultados = [];

    for (doc in docs) {
        document load atual(doc);

        var tipo_documento = classify atual {
            categories: ["foto", "laudo", "certidao", "planta"],
            confidence_threshold: 0.85
        };

        if (tipo_documento == "foto") {
            var analise_foto = analyze atual with gemini {
                prompt: "Analise esta foto para perícia, identifique danos, estado de conservação e pontos relevantes",
                format: "json"
            };
            resultados.append(analise_foto);
        }
    }

    return resultados;
}

var resultado_pericia = processar_pericia(documentos_pericia);

# Geração de laudo automático
export analysis to pdf("laudo_pericia.pdf") {
    template: "template_pericia.html",
    data: {
        "resultados": resultado_pericia,
        "conclusoes": analyze_all_results(resultado_pericia)
    }
};
```

5. Instalação e Distribuição

5.1 Setup Script (setup.py)

```
python
```

```
from setuptools import setup, find_packages

setup(
    name="corplang",
    version="1.0.0",
    description="Linguagem de Programação Corporativa com IA",
    author="Lucas",
    packages=find_packages(),
    install_requires=[
        "google-generativeai>=0.3.0",
        "PyPDF2>=3.0.0",
        "python-docx>=0.8.11",
        "Pillow>=9.0.0",
        "pytesseract>=0.3.10",
        "jinja2>=3.0.0",
        "weasyprint>=57.0",
        "pandas>=1.5.0",
        "openpyxl>=3.0.0"
    ],
    entry_points={
        'console_scripts': [
            'corplang=corplang.cli.main:main',
            'cl=corplang.cli.main:main',
        ],
    },
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
    python_requires='>=3.8',
)
```

5.2 Instalação Cross-Platform

```
bash

# Linux/Mac
pip install corplang
# ou
python setup.py install

# Windows
pip install corplang
# ou via installer
corplang-installer-windows.exe
```

6. IDE e Ferramentas

6.1 Extensão VSCode

```
json
```

```
{
  "name": "corplang-support",
  "displayName": "CorPLang Support",
  "description": "Suporte completo para CorPLang",
  "version": "1.0.0",
  "engines": {
    "vscode": "^1.60.0"
  },
  "categories": ["Programming Languages"],
  "contributes": {
    "languages": [{
      "id": "corplang",
      "aliases": ["CorPLang", "corplang"],
      "extensions": [".cl", ".corplang"],
      "configuration": "./language-configuration.json"
    }],
    "grammars": [{
      "language": "corplang",
      "scopeName": "source.corplang",
      "path": "./syntaxes/corplang.tmLanguage.json"
    }]
  }
}
```

6.2 Language Server Protocol

python

```
class CorPLangLanguageServer:
    def __init__(self):
        self.parser = CorPLangParser()
        self.analyzer = SemanticAnalyzer()

    def handle_completion(self, params):
        """Autocompletar código"""
        pass

    def handle_hover(self, params):
        """Informações on-hover"""
        pass

    def handle_diagnostics(self, params):
        """Análise de erros em tempo real"""
        pass
```

7. Integração com APIs

7.1 Configuração de API

corplang


```
# Configuração global
config api_keys {
    gemini: "sua_chave_gemini",
    openai: "sua_chave_openai",
    azure: "sua_chave_azure"
};

# Configuração de provider padrão
config default_ai_provider("gemini");
```

7.2 Fallback e Redundância

```
python

class AIProvider:
    def __init__(self):
        self.providers = {
            'gemini': GeminiClient(),
            'openai': OpenAIClient(),
            'azure': AzureClient()
        }
        self.fallback_order = ['gemini', 'openai', 'azure']

    def analyze_with_fallback(self, text, prompt):
        """Tenta diferentes providers em caso de falha"""
        for provider_name in self.fallback_order:
            try:
                return self.providers[provider_name].analyze(text, prompt)
            except Exception as e:
                print(f"Falha no provider {provider_name}: {e}")
                continue
        raise Exception("Todos os providers falharam")
```

8. Templates e Relatórios

8.1 Template HTML para Relatórios

```
html
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Relatório de Análise - {{documento.nome}}</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 40px; }
    .header { border-bottom: 2px solid #333; padding-bottom: 20px; }
    .section { margin: 20px 0; }
    .highlight { background-color: #ffeb3b; padding: 2px 4px; }
    .risk-high { color: #f44336; font-weight: bold; }
    .risk-medium { color: #ff9800; font-weight: bold; }
    .risk-low { color: #4caf50; font-weight: bold; }
  </style>
</head>
<body>
  <div class="header">
    <h1>Relatório de Análise Documental</h1>
    <p>Documento: {{documento.nome}}</p>
    <p>Data: {{data_analise}}</p>
    <p>Usuário: {{usuario.name}}</p>
  </div>

  <div class="section">
    <h2>Dados Extraídos</h2>
    {% for campo, valor in dados_extraidos.items() %}
    <p><strong>{{campo}}:</strong> {{valor}}</p>
    {% endfor %}
  </div>

  <div class="section">
    <h2>Análise de Risco</h2>
    <p class="risk-{{nivel_risco}}">Nível de Risco: {{nivel_risco.upper()}}</p>

    {% if irregularidades %}
    <h3>Irregularidades Identificadas</h3>
    <ul>
    {% for irregularidade in irregularidades %}
      <li class="highlight">{{irregularidade}}</li>
    {% endfor %}
    </ul>
    {% endif %}
  </div>

  <div class="section">
    <h2>Conclusões e Recomendações</h2>
    <p>{{conclusoes}}</p>
  </div>
</body>
</html>
```

9. Tratamento de Erros e Logging

9.1 Sistema de Logging

python

```
class CorpLangLogger:
    def __init__(self):
        self.setup_logging()

    def log_document_processing(self, doc_name, status, details):
        """Log específico para processamento de documentos"""
        pass

    def log_ai_request(self, provider, prompt, response_time):
        """Log de requisições AI"""
        pass

    def log_export_operation(self, format_type, output_path, success):
        """Log de operações de export"""
        pass
```

9.2 Tratamento de Erros

```
corplang

# Tratamento de erros na linguagem
try {
    document load contrato("arquivo_inexistente.pdf");
} catch (DocumentNotFoundError e) {
    print("Erro: Documento não encontrado -", e.message);
    return false;
} catch (AIServiceError e) {
    print("Erro no serviço de IA -", e.message);
    # Fallback para processamento manual
    manual_process(contrato);
}
```

10. Deployment e Distribuição

10.1 Docker Container

```
dockerfile

FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .

RUN python setup.py install

EXPOSE 8080

CMD ["corplang", "server", "--port=8080"]
```

10.2 Distribuição via PyPI

bash

```
# Preparação
python setup.py sdist bdist_wheel

# Upload
twine upload dist/*

# Instalação pelos usuários
pip install corplang
```

11. Roadmap de Desenvolvimento

Fase 1 (Fundação)

- ☐ Implementar extensões básicas de documento
- ☐ Integração com Gemini API
- ☐ Sistema básico de templates
- ☐ Exportação para PDF

Fase 2 (Melhorias)

- ☐ OCR integrado
- ☐ Múltiplos providers de IA
- ☐ IDE/Editor com syntax highlighting
- ☐ Sistema de plugins

Fase 3 (Avançado)

- ☐ Machine Learning local
- ☐ Análise de batch em paralelo
- ☐ API REST para integração
- ☐ Dashboard web

Fase 4 (Empresa)

- ☐ Autenticação e autorização
- ☐ Auditoria e compliance
- ☐ Integração com sistemas empresariais
- ☐ Suporte a workflows complexos

12. Considerações de Segurança

12.1 Proteção de Dados

```
python
```

```
class SecurityManager:
    def __init__(self):
        self.encryption_key = self.generate_key()

    def encrypt_document(self, doc_content):
        """Criptografa conteúdo sensível"""
        pass

    def sanitize_ai_prompt(self, prompt):
        """Remove informações sensíveis dos prompts"""
        pass

    def audit_log(self, action, user, document):
        """Log de auditoria"""
        pass
```

12.2 Configuração de Privacidade

```
corplang

# Configurações de privacidade
config privacy {
    anonymize_data: true,
    store_locally: true,
    encrypt_exports: true,
    audit_trail: true
};
```

Este planejamento fornece uma base sólida para implementar o sistema de análise de documentos na CorpLang, com foco em documentos imobiliários e perícias, mantendo flexibilidade para expansão futura.