项目进度报告

# **1 项目进度报告**

## **1.1 项目概述**

本项目旨在开发一个基于 Python 的 Markdown 转 Word 文档 MCP 服务器，完整复刻 Java 版本的所有功能。

## **1.2 功能实现**

### **1.2.1 核心功能**

1. \*\*Markdown 解析\*\*：支持 H1-H6 标题、段落、表格

2. \*\*ECharts 图表转换\*\*：将 ```echarts 代码块转换为 Word 图表

3. \*\*动态 Word 模板生成\*\*：根据 Markdown 结构创建带占位符的模板

4. \*\*标题自动编号\*\*：多级标题自动添加序号

### **1.2.2 技术架构**

**表格 1：**

|  |  |  |
| --- | --- | --- |
| 组件 | 功能 | 状态 |
| 数据模型层 | ChartColumn、ChartTable、WordParams | ✅ 完成 |
| 解析器层 | MarkdownParser、TableParser | ✅ 完成 |
| ECharts 转换器 | JSON 解析、数据提取 | ✅ 完成 |
| 模板创建器 | 动态生成 Word 模板 | ✅ 完成 |
| Word 生成器 | 占位符替换、内容插入 | ✅ 完成 |
| 主转换器 | 协调所有组件 | ✅ 完成 |
| MCP 服务器 | 定义 tools、集成转换器 | ✅ 完成 |

## **1.3 数据统计**

### **1.3.1 开发进度**

**图表 1：**

**【图表】开发进度统计**

|  |  |
| --- | --- |
| **类别** | **完成度** |
| 数据模型 | 100 |
| 解析器 | 100 |
| 转换器 | 100 |
| 模板 | 100 |
| 生成器 | 100 |
| 服务器 | 100 |

### **1.3.2 代码行数统计**

**表格 2：**

|  |  |  |
| --- | --- | --- |
| 模块 | 文件数 | 代码行数 |
| 数据模型 | 3 | 150 |
| 解析器 | 2 | 200 |
| 核心转换 | 4 | 800 |
| MCP 服务器 | 1 | 200 |
| 测试 | 1 | 300 |
| \*\*总计\*\* | \*\*11\*\* | \*\*1650\*\* |

## **1.4 测试结果**

### **1.4.1 功能测试**

- ✅ Markdown 标题解析

- ✅ 表格解析和转换

- ✅ ECharts 图表转换

- ✅ 标题自动编号

- ✅ Word 文档生成

- ✅ MCP 协议支持

### **1.4.2 性能测试**

**图表 2：**

**【图表】转换性能测试**

|  |  |
| --- | --- |
| **类别** | **转换时间(ms)** |
| 简单文档 | 50 |
| 包含表格 | 120 |
| 包含图表 | 200 |
| 复杂文档 | 350 |

## **1.5 部署说明**

### **1.5.1 安装步骤**

1. 克隆项目到本地

2. 安装 Python 依赖

3. 配置 Claude Desktop

4. 启动 MCP 服务器

### **1.5.2 配置示例**

```json

{

"mcpServers": {

"md2doc": {

"command": "python",

"args": ["-m", "md2doc\_mcp.server"]

}

}

}

```

## **1.6 总结**

本项目成功将 Java 版本的 Markdown 转 Word 服务转换为 Python MCP 服务，保持了所有核心功能：

- 完整的 Markdown 解析能力

- ECharts 图表支持

- 动态模板生成

- 标题自动编号

- 格式化样式

- MCP 协议兼容

项目已通过全面测试，可以投入使用。