MCP Tool Development Strategy

🎯 Overview

This document outlines the strategies and best practices for developing and maintaining MCP tools in a scalable and maintainable way.

🏗️ Architecture Principles

1. Separation of Concerns

- Tool Definition: Each tool is a separate class implementing `McpTool`

- Tool Registry: Central registry manages all tools dynamically

- Base Implementation: Common functionality in `BaseMcpTool`

- Category Organization: Tools grouped by domain (products, search, financial)

2. Single Responsibility

- Each tool class handles one specific API operation

- Clear input/output contracts

- Self-contained validation and error handling

3. DRY (Don't Repeat Yourself)

- Common patterns abstracted into base classes

- Shared utility methods for argument handling

- Standardized response formats

📁 Directory Structure

enterpriseMcp/src/main/java/tools/

├── McpTool.java # Base interface

├── BaseMcpTool.java # Base implementation

├── ToolRegistry.java # Tool management

├── ToolFactory.java # Tool registration

├── products/ # Product-related tools

│ ├── GetProductsTool.java

│ ├── GetProductTool.java

│ ├── CheckProductTool.java

│ ├── AddProductTool.java

│ └── UpdateProductTool.java

├── search/ # Search-related tools

│ ├── SearchProductTool.java

│ ├── SearchOfficeTool.java

│ ├── SearchVendorTool.java

│ └── ...

└── financial/ # Financial tools

├── GetAdvanceDetailsTool.java

├── GetVendorAdvancesTool.java

└── ...

🛠️ Tool Development Guidelines

... (Truncated for readability, will be included in the full document)