MCP-Tools Implementierungsplan für AGENT_LAND_SAARLAND

1. Vorbereitungen

Backup der vorhandenen Konfiguration

bash

```
# Backup der aktuellen MCP-Server-Konfiguration erstellen
tar -czvf mcp_servers_backup_$(date +%Y%m%d).tar.gz ~/.claude/mcp.json
```

Umgebung vorbereiten

bash

```
# Projektverzeichnis navigieren
cd /Users/deepsleeping/agentland.saarland

# Abhängigkeiten aktualisieren
npm install

# Benötigte MCP-Tools installieren
npm install -g @smithery/cli@latest
```

2. MCP-Server Integration

Desktop Commander Integration

bash

```
# Desktop Commander MCP-Server hinzufügen
claude mcp add '{"mcpServers": {"desktop-commander": {"command": "npx", "args": ["-y
```

Context7-MCP Integration

bash

```
# Context7-MCP Server hinzufügen
claude mcp add '{"mcpServers": {"context7-mcp": {"command": "npx", "args": ["-y", "@
```

Clear Thought Integration

bash

```
# Clear Thought Server hinzufügen
claude mcp add '{"mcpServers": {"clear-thought": {"command": "npx", "args": ["-y", "
```

Sequential Thinking Server

bash

```
# Sequential Thinking Server hinzufügen
claude mcp add '{"mcpServers": {"server-sequential-thinking": {"command": "npx", "ar
```

GitHub Integration

bash

```
# GitHub MCP-Server hinzufügen
claude mcp add '{"mcpServers": {"github": {"command": "npx", "args": ["-y", "@smithe
```

Shadcn UI Integration

bash

```
# Shadcn UI MCP-Server hinzufügen
claude mcp add '{"mcpServers": {"shadcn-ui-mcp-server": {"command": "npx", "args": [
```

DeepView Integration

bash

```
# DeepView MCP-Server hinzufügen
claude mcp add '{"mcpServers": {"deepview-mcp": {"command": "npx", "args": ["-y", "@
```

Gemini MCP Integration

bash

```
# Gemini MCP-Server hinzufügen
claude mcp add '{"mcpServers": {"geminimcptest": {"command": "npx", "args": ["-y", "
```

DeepResearch MCP

bash

```
# DeepResearch MCP-Server hinzufügen
claude mcp add '{"mcpServers": {"DeepResearchMCP": {"command": "npx", "args": ["-y",
```

3. Verifikation der Installation

Überprüfung des Status

bash

```
# MCP-Server Status überprüfen
claude mcp list

# MCP-Server starten (falls nicht automatisch gestartet)
claude mcp start
```

Funktionstest

bash

```
# Eigenes Testskript ausführen
node test-a2a-security.js

# Verbindung zu den MCP-Servern testen
node tools/scripts/mcp/test-connections.js
```

4. Projektspezifische Integration

Framework-Anpassungen für AGENT_LAND_SAARLAND

In der Datei `/Users/deepsleeping/agentland.saarland/libs/mcp/src/setup_mcp.js`:

javascript

```javascript
// MCP-Tools in das AGENT_LAND_SAARLAND Framework integrieren
const setupMCPIntegration = async () => {
  try {
    console.log('Initialisiere MCP-Tools für AGENT_LAND_SAARLAND...');

    // Desktop Commander integrieren
    const desktopCommander = await initializeMCPService('desktop-commander');

    // Sequential Thinking in Workflow integrieren
    const sequentialThinking = await initializeMCPService('server-sequential-thinkin
    workflowEngine.registerThinkingService(sequentialThinking);

    // GitHub-Integration für Code-Repositories
    const github = await initializeMCPService('github');
    registerCodeRepository(github);

    // DeepView für Code-Analyse
    const deepview = await initializeMCPService('deepview-mcp');
    registerCodeAnalyzer(deepview);

    // DeepResearch für komplexe Recherchen
    const deepResearch = await initializeMCPService('DeepResearchMCP');
    registerResearchService(deepResearch);

    // Logging
    console.log('MCP-Tools erfolgreich integriert!');
    return true;
  } catch (error) {
    console.error('Fehler bei der MCP-Integration:', error);
    return false;
  }
};

module.exports = { setupMCPIntegration };
```

MCP-Integration in die Agent-Architektur

In der Datei `/Users/deepsleeping/agentland.saarland/libs/agents/src/agent-base/mcp-capable-agent.ts`:

typescript

```
import { MCPClient } from '../../mcp/src/client';

export class MCPCapableAgent extends BaseAgent {
  protected mcpClient: MCPClient;
  protected availableTools: string[] = [];

  constructor(config) {
    super(config);
    this.mcpClient = new MCPClient();
    this.initializeTools();
  }

  async initializeTools() {
    // Verfügbare MCP-Tools abfragen und registrieren
    const tools = await this.mcpClient.listTools();
    this.availableTools = tools.map(tool => tool.name);

    // Agent-spezifische Tools aktivieren
    this.setupAgentSpecificTools();
  }

  setupAgentSpecificTools() {
    // Wird von spezialisierten Agenten überschrieben
  }

  async useThinkingTool(query, approach = '', options = {}) {
    if (this.availableTools.includes('sequentialthinking')) {
      return this.mcpClient.callTool('sequentialthinking', {
        query,
        approach,
        ...options
      });
    }

    if (this.availableTools.includes('geminithinking')) {
      return this.mcpClient.callTool('geminithinking', {
        query,
        approach,
        ...options
      });
    }

    throw new Error('Kein Thinking-Tool verfügbar');
  }
```

```
  // Weitere Tool-spezifische Methoden...
}
```

5. Einrichtung für spezialisierte Agenten

Navigator-Agent MCP-Konfiguration

In der neu zu erstellenden Datei `/Users/deepsleeping/agentland.saarland/libs/agents/src/specialized/navigator-agent.ts`:

typescript

```typescript
import { MCPCapableAgent } from '../agent-base/mcp-capable-agent';

export class NavigatorAgent extends MCPCapableAgent {
  constructor(config) {
    super({
      ...config,
      agentType: 'navigator',
      description: 'Zentraler Navigator-Agent für AGENT_LAND_SAARLAND'
    });
  }

  setupAgentSpecificTools() {
    // Navigator-spezifische MCP-Tools aktivieren
    this.requiredTools = [
      'sequentialthinking',
      'github',
      'deepview-mcp',
      'DeepResearchMCP'
    ];

    // Überprüfen, ob alle benötigten Tools verfügbar sind
    const missingTools = this.requiredTools.filter(
      tool => !this.availableTools.includes(tool)
    );

    if (missingTools.length > 0) {
      console.warn(
        `Navigator-Agent: Fehlende MCP-Tools: ${missingTools.join(', ')}`
      );
    }
  }

  // Navigationsspezifische Methoden implementieren...
}
```

Tourismus-Agent MCP-Konfiguration

In der neu zu erstellenden Datei `/Users/deepsleeping/agentland.saarland/libs/agents/src/specialized/tourismus-agent.ts`:

typescript

```typescript
import { MCPCapableAgent } from '../agent-base/mcp-capable-agent';

export class TourismusAgent extends MCPCapableAgent {
  constructor(config) {
    super({
      ...config,
      agentType: 'tourismus',
      description: 'Tourismus-Agent für AGENT_LAND_SAARLAND'
    });
  }

  setupAgentSpecificTools() {
    // Tourismus-spezifische MCP-Tools aktivieren
    this.requiredTools = [
      'sequentialthinking',
      'DeepResearchMCP'
    ];

    // Überprüfen, ob alle benötigten Tools verfügbar sind
    const missingTools = this.requiredTools.filter(
      tool => !this.availableTools.includes(tool)
    );

    if (missingTools.length > 0) {
      console.warn(
        `Tourismus-Agent: Fehlende MCP-Tools: ${missingTools.join(', ')}`
      );
    }
  }

  // Tourismus-spezifische Methoden implementieren...
}
```

6. Automatisierung und Deployment

Automatisierte MCP-Setup-Skript

Erstellen eines Skripts in `/Users/deepsleeping/agentland.saarland/scripts/setup-mcp-tools.sh`:

bash

```
#!/bin/bash
# AGENT_LAND_SAARLAND MCP-Tools Setup

# Fehlerbehandlung
set -e

echo "==== AGENT_LAND_SAARLAND MCP-Tools Setup ===="
echo "Vorbereitung der Umgebung..."

# Backup erstellen
mkdir -p backups
tar -czvf backups/mcp_servers_backup_$(date +%Y%m%d).tar.gz ~/.claude/mcp.json 2>/de

# MCP-Server-Konfiguration hinzufügen
echo "MCP-Server-Konfigurationen werden hinzugefügt..."

# JSON-Datei erstellen
cat > temp_mcp_config.json << EOL
{
  "mcpServers": {
    "desktop-commander": {
      "command": "npx",
      "args": ["-y", "@smithery/cli@latest", "run", "@wonderwhy-er/desktop-commander
    },
    "context7-mcp": {
      "command": "npx",
      "args": ["-y", "@smithery/cli@latest", "run", "@upstash/context7-mcp", "--key"
    },
    "clear-thought": {
      "command": "npx",
      "args": ["-y", "@smithery/cli@latest", "run", "@waldzellai/clear-thought", "--
    },
    "server-sequential-thinking": {
      "command": "npx",
      "args": ["-y", "@smithery/cli@latest", "run", "@smithery-ai/server-sequential-
    },
    "github": {
      "command": "npx",
      "args": ["-y", "@smithery/cli@latest", "run", "@smithery-ai/github", "--key",
    },
    "shadcn-ui-mcp-server": {
      "command": "npx",
      "args": ["-y", "@smithery/cli@latest", "run", "@ymadd/shadcn-ui-mcp-server", "
    },
    "deepview-mcp": {
      "command": "npx",
```

```
      "args": ["-y", "@smithery/cli@latest", "run", "@ai-1st/deepview-mcp", "--key",
    },
    "geminimcptest": {
      "command": "npx",
      "args": ["-y", "@smithery/cli@latest", "run", "@palolxx/geminimcptest", "--key
    },
    "DeepResearchMCP": {
      "command": "npx",
      "args": ["-y", "@smithery/cli@latest", "run", "@ameeralns/DeepResearchMCP", "-
    }
  }
}
EOL

# Claude MCP hinzufügen - einzeln für Fehlerbehandlung
for server in desktop-commander context7-mcp clear-thought server-sequential-thinkin
  echo "MCP-Server hinzufügen: $server"
  jq -r ".mcpServers.\"$server\"" temp_mcp_config.json > temp_server.json
  claude mcp add "{\"mcpServers\": {\"$server\": $(cat temp_server.json)}}" || echo
  sleep 1
done

# Aufräumen
rm temp_mcp_config.json temp_server.json

# Status überprüfen
echo "Überprüfe MCP-Server Status..."
claude mcp list

echo "MCP-Server werden gestartet..."
claude mcp start

echo "Setup abgeschlossen!"
echo "==== AGENT_LAND_SAARLAND MCP-Tools bereit ===="
```

Integration in Projekt-Setup

Anpassung der Datei `/Users/deepsleeping/agentland.saarland/package.json`:

```json
{
  "scripts": {
    "setup:mcp": "bash scripts/setup-mcp-tools.sh",
    "start:mcp": "claude mcp start",
    "stop:mcp": "claude mcp stop",
    "test:mcp": "node tools/scripts/mcp/test-connections.js"
  }
}
```

7. Dokumentation

Neue Dokumentationsdatei erstellen in `/Users/deepsleeping/agentland.saarland/docs/mcp-tools.md`:

```markdown
# MCP-Tools Integration in AGENT_LAND_SAARLAND

Dieses Dokument beschreibt die Integration und Nutzung der MCP-Tools (Model Control

## Überblick

Die MCP-Tools erweitern die Fähigkeiten der KI-Agenten durch spezialisierte Dienste

- **Desktop Commander**: Systeminteraktion und Dateimanagement
- **Sequential Thinking**: Strukturierte Denkprozesse für komplexe Probleme
- **GitHub Integration**: Code-Repository-Management
- **DeepView**: Code-Analyse und -Verständnis
- **DeepResearch**: Komplexe Recherchen und Informationsaggregation

## Einrichtung

Die MCP-Tools können mit dem Setup-Skript installiert werden:

```bash
npm run setup:mcp
```

Einzelne Tools können auch manuell hinzugefügt werden:

```bash
claude mcp add '{"mcpServers": {"tool-name": {...}}}'
```

Verwendung in Agenten
```

Die spezialisierten Agenten von AGENT_LAND_SAARLAND greifen über die `MCPCapableAgent`-Basisklasse auf die MCP-Tools zu:

```typescript
// Beispiel für die Nutzung des Sequential Thinking Tools
const denkprozess = await agent.useThinkingTool(
  "Wie optimiere ich die Tourismusroute im nördlichen Saarland?",
  "Berücksichtige Verkehrsanbindungen und lokale Sehenswürdigkeiten"
);
```

Verfügbare Tools

Tool-Name	Primäre Funktion	Empfohlene Agenten
desktop-commander	Dateisystem-Interaktion	Alle
server-sequential-thinking	Strukturierte Denkprozesse	Navigator, Verwaltung
github	Code-Repository-Management	Wirtschaft, KI-Schmiede
deepview-mcp	Code-Analyse	KI-Schmiede
DeepResearchMCP	Informationsrecherche	Tourismus, Kultur

Fehlerbehebung

Bei Problemen mit den MCP-Tools:

1. Status prüfen: `claude mcp list`
2. Neustart: `claude mcp restart`
3. Logs prüfen: `claude mcp logs <server-name>`
4. Verbindungstest: `npm run test:mcp`

Weitere Ressourcen

- Claude MCP Dokumentation
- AGENT_LAND_SAARLAND Framework-Dokumente
- Agentenarchitektur