

스토브 개발자 센터 MCP 서버

차세대 개발자 지원 시스템

기술위원회 공유용 | 2025.06.24

개요 및 배경

MCP(Model Context Protocol) 개념

AI 모델이 외부 데이터 소스 및 도구와 안전하고 제어된 방식으로 상호작용할 수 있게 해주는 **오픈 프로토콜**

개발자 센터 문서화 현황

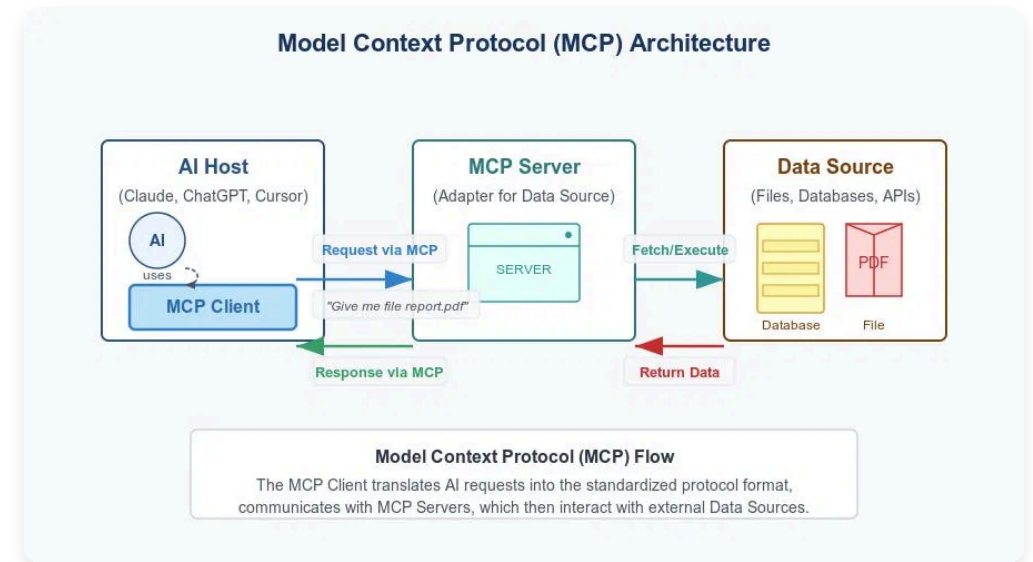
방대한 API 문서(1,360개 문서)가 존재하나, **검색 및 접근성 한계**로 개발자 경험 저하

현재 문제점

- 복잡한 API 문서 탐색 어려움
- 자연어 질의 불가능
- 개발자 지원 부담 증가

MCP 서버 솔루션

FastMCP 프레임워크 기반으로 **자연어 질의응답**을 통해 개발자 경험 혁신



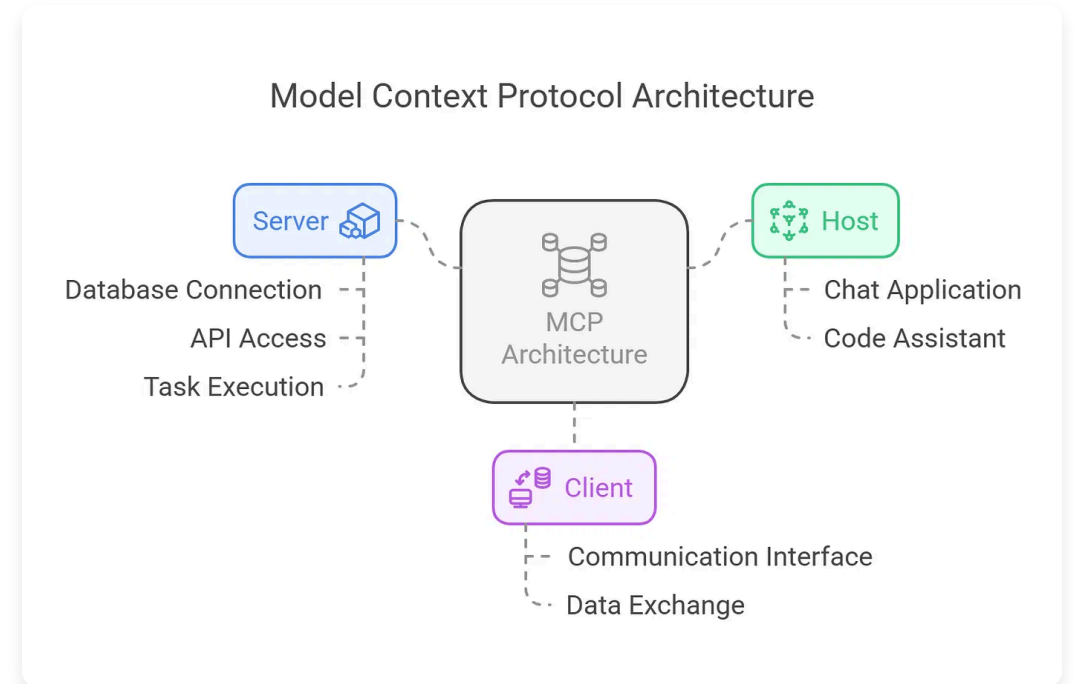
핵심 기술 아키텍처

MCP 프로토콜 구조

- ✓ **표준 프로토콜**: AI 모델과 외부 도구 간 통신 규약
- ✓ **Transport 지원**: STDIO, HTTP(SSE) 방식
- ✓ **Tool 기반 아키텍처**: @mcp.tool 데코레이터

데이터 흐름

- 1 개발자가 자연어로 질문
- 2 LLM이 질문을 분석하고 MCP를 통해 검색 요청
- 3 MCP Server가 BM25 알고리즘으로 관련 문서 검색
- 4 검색 결과를 MCP Protocol로 LLM에 전달
- 5 LLM이 컨텍스트를 활용해 개발자에게 답변 제공



연결 구조도

개발자 ↔ LLM(Claude/Cursor) ↔ MCP Protocol ↔ MCP Server ↔ STOVE Developer Center 컨텐츠

검색 엔진 기술

기본 버전

- 단순 키워드 매칭 기반 검색
- 제목/내용 문자열 포함 검색
- 간단한 구현, 제한된 정확도

고도화 버전 (BM25)

- **BM25 Okapi 알고리즘** 적용
- 제목 가중치 부여 (3배)
- 토큰 기반 정확도 향상

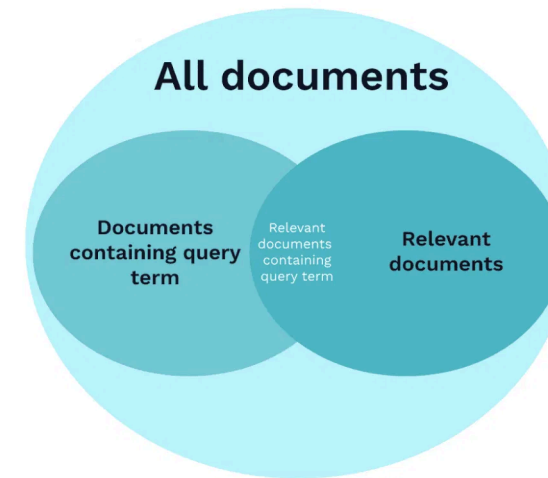
검색 품질 최적화 방안

- 문서 필드별 가중치 조정
- 한국어 형태소 분석기 적용 가능
- 문서 인덱싱 최적화

```
# BM25 인덱스 생성 코드
corpus = [text.lower().split() for text in corpus_texts]
BM25_INDEX = BM25Okapi(corpus)

# 검색 시 점수 계산
tokenized_query = query.lower().split()
doc_scores = BM25_INDEX.get_scores(tokenized_query)
```

Best Match 25



구분	기본 버전	BM25 버전
검색 정확도	낮음	높음
구현 복잡도	단순	중간
확장성	제한적	우수

기술적 핵심 가치

👤 개발자 경험(DX) 혁신

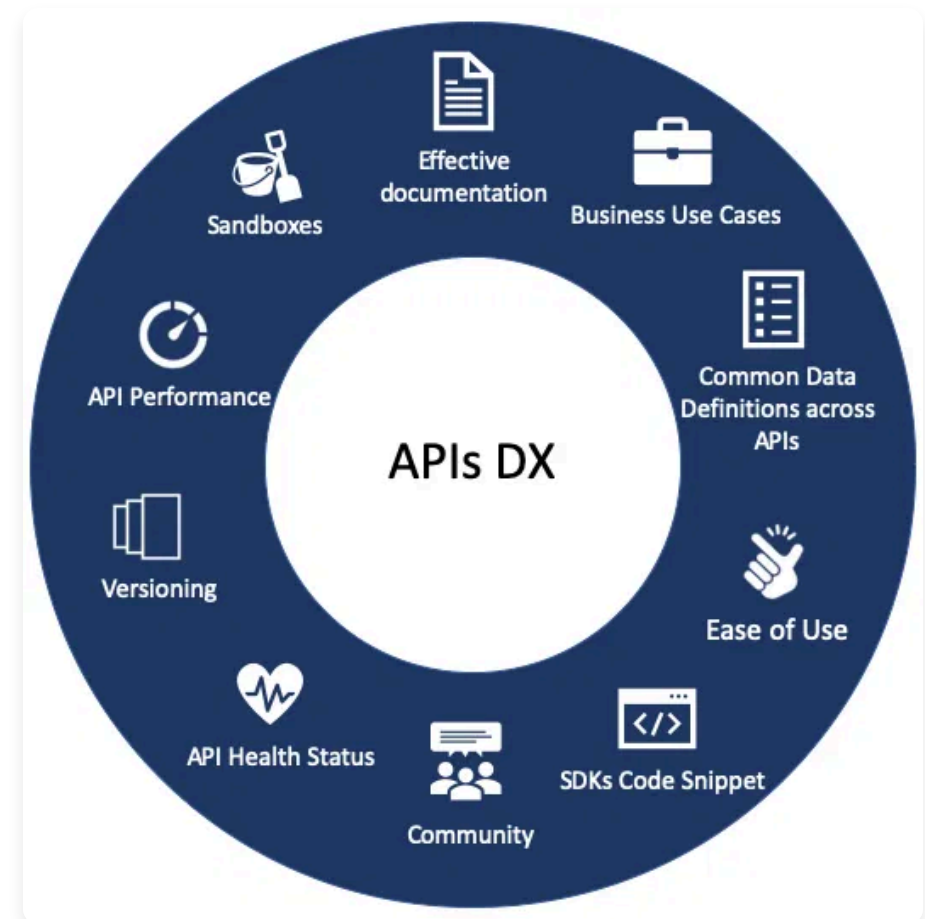
- **자연어 인터페이스**: 복잡한 API 문서를 일반 언어로 질문 가능
- **실시간 응답**: 즉시 관련 문서와 코드 예시 제공
- **컨텍스트 유지**: 연속적인 대화를 통한 깊이 있는 문제 해결

🧩 확장성 및 유연성

- **모듈화 설계**: 새로운 검색 알고리즘 쉬운 교체/확장
- **다중 Transport**: STUDIO(IDE 통합), HTTP(웹 서비스) 동시 지원
- **플랫폼 독립적**: 다양한 AI 클라이언트와 호환

검색 품질 최적화

- **BM25 알고리즘**: 정보 검색 분야의 검증된 알고리즘
- **가중치 조정**: 제목 중요도 강화로 정확도 향상
- **멀티 필드 검색**: 제목, 설명, 내용 통합 검색



활용 사례

개발자 지원 시나리오

개발자 질의응답 예시

개발자: 스토브 로그인 연동 방법 알려줘

MCP 서버: [인증] GNB 방식 연동 가이드

- JavaScript 코드 예시: `STOVE.login({ ... })`

- [관련 문서 링크](#)

통합 환경 지원



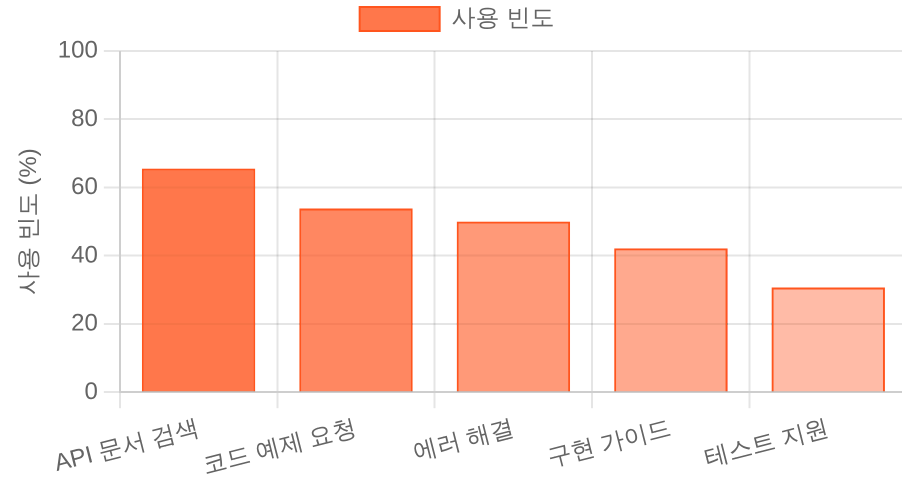
Cursor IDE



Claude



HTTP API



실제 사용 예시

API 문서 탐색

자연어 질의로 문서 검색, 코드 예제 제공

에러 해결

에러 코드 설명, 해결책 제안

구현 가이드

단계별 구현 안내, 모범 사례 공유

테스트 지원

테스트 시나리오, 디버깅 가이드

향후 발전 방향

🔍 검색 기술 고도화

임베딩 기반 시맨틱 검색

- 현재: 키워드/BM25 기반 → 향후: 벡터 유사도 기반
- 의도 파악 정확도 대폭 향상

목표 아키텍처

```
class HybridSearchEngine:
    def search(self, query):
        bm25_results = self.bm25_search(query)
        semantic_results = self.embedding_search(query)
        return self.fusion_ranking(bm25_results, semantic_results)
```

🤖 AI 기능 확장

코드 생성 및 검증

- API 호출 코드 자동 생성
- 파라미터 유효성 검사

상호작용 개선

- 멀티턴 대화 컨텍스트 관리
- 개인화된 답변 제공

📄 콘텐츠 보강

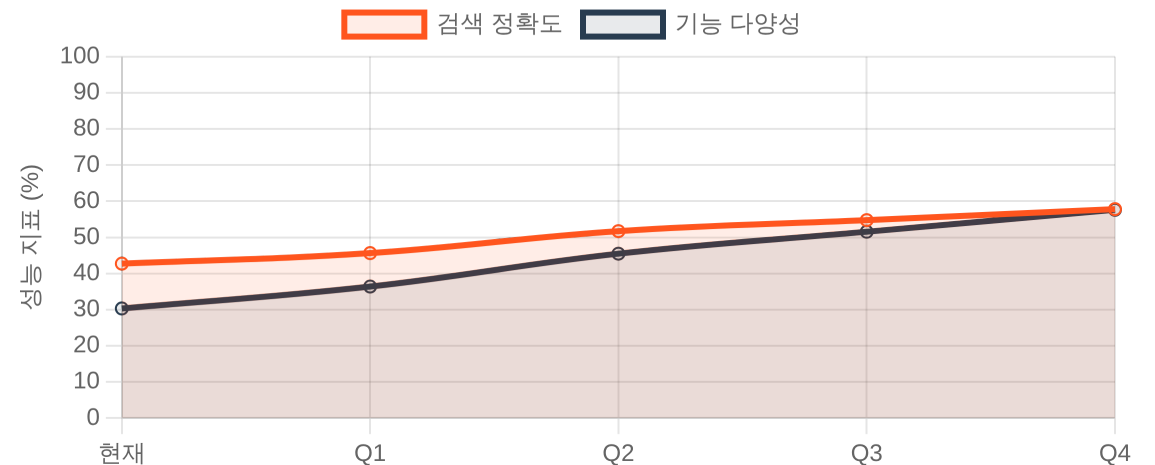
Example 코드 자동 생성

- API 호출 예시 코드 동적 생성
- 다양한 언어별 SDK 예제 제공

테스트 환경 제공

- 샌드박스 API 엔드포인트 연동
- 실시간 API 테스트 기능

기능 발전 로드맵






도입 효과 및 ROI

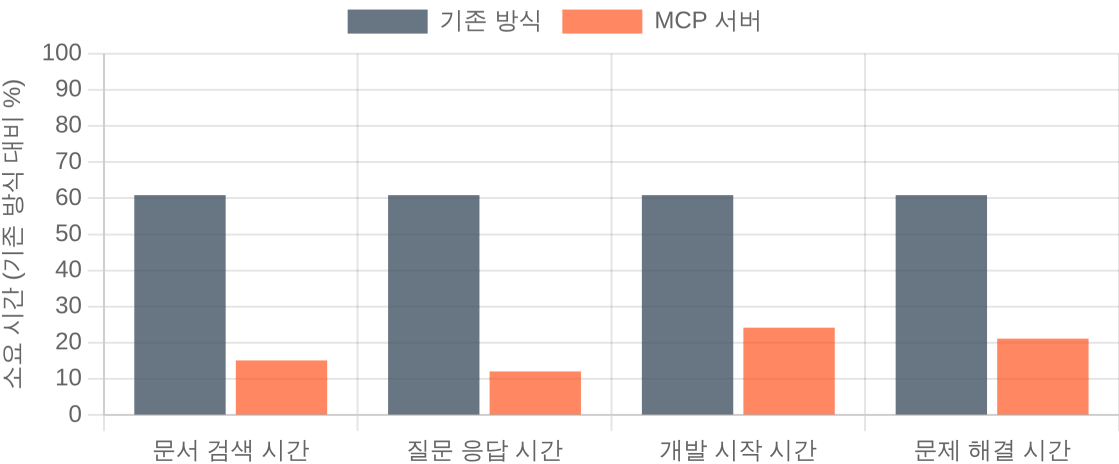
정량적 효과

-  개발자 문의 시간 단축
평균 **75% 감소** 예상
-  문서 접근성 향상
자연어 질의로 **진입 장벽 제거**
-  개발 속도 향상
API 학습 곡선 **완화**

정성적 효과

-  개발자 만족도 향상
-  플랫폼 채택률 증가
-  기술 지원 부담 감소

시간 효율성 비교



기술적 차별점

구분	기존 방식	MCP 서버
접근성	수동 문서 탐색	자연어 질의
정확성	사용자 해석 의존	AI 기반 정확한 답변
효율성	다중 페이지 확인	원스톱 솔루션
확장성	정적 문서	동적 응답

구현 로드맵

1 기반 기술 안정화

Q1

- BM25 검색 엔진 성능 최적화
- 다양한 클라이언트 호환성 테스트
- 문서 품질 관리 시스템 구축

2 지능화 기능 추가

Q2

- 임베딩 기반 시맨틱 검색 도입
- 코드 생성 기능 개발
- 사용자 피드백 학습 시스템

3 생태계 확장

Q3

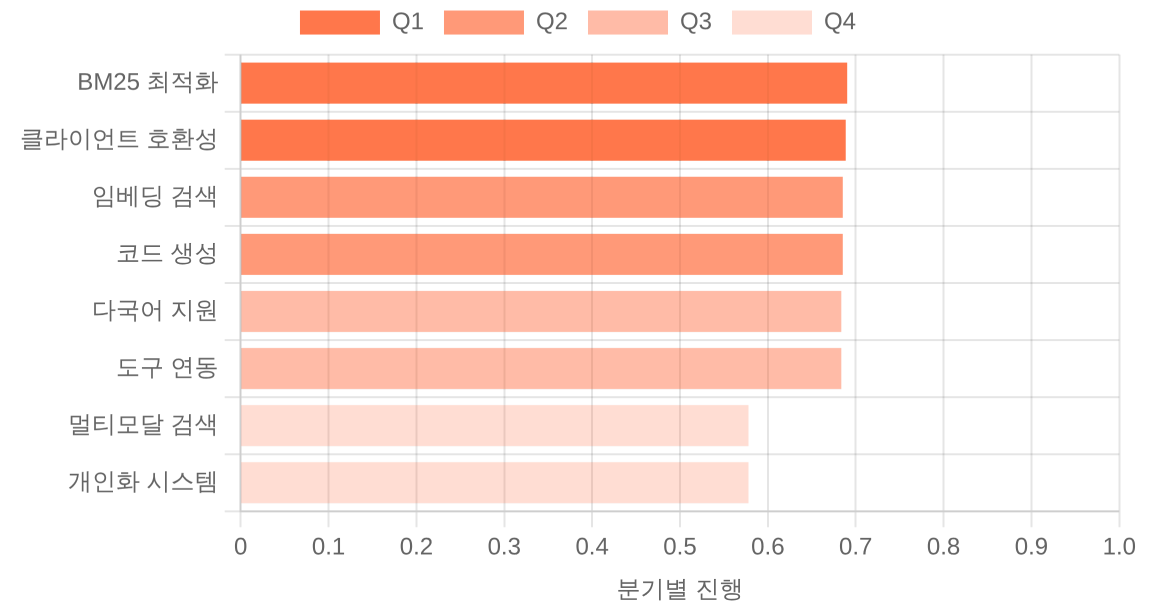
- 다국어 지원 확대
- 외부 개발 도구 연동
- 실시간 문서 동기화

4 차세대 기능

Q4

- 멀티모달 검색 (이미지, 다이어그램)
- 개인화 추천 시스템
- 자동 문서 생성 지원

기능 개발 일정



현재 (Q1 시작)

BM25 기반 검색 엔진 구현 완료, 기본 MCP 서버 구축

Q2 중간

임베딩 모델 학습 및 하이브리드 검색 시스템 테스트

Q4 말

완전한 멀티모달 지원 및 개인화 시스템 출시

결론 및 제안

"STOVE Developer Center MCP 서버는 **차세대 개발자 지원 도구**로서의 잠재력을 보여주는 혁신적인 프로젝트입니다."

💡 핵심 제안사항

- 1 **즉시 도입**: 현재 BM25 기반 시스템으로도 충분한 가치 제공
- 2 **단계적 확장**: 임베딩 기반 검색으로 점진적 고도화
- 3 **생태계 연동**: 주요 개발자 도구와의 통합 추진
- 4 **지속적 개선**: 사용자 피드백 기반의 반복적 개선

📊 기대 효과

- 개발자 생산성 향상 및 학습 곡선 완화
- 플랫폼 채택률 증가 및 생태계 확장
- 기술 지원 비용 절감 및 효율성 증대
- 개발자 경험(DX) 차별화로 경쟁력 강화

▶▶ 다음 단계

- 기술위원회 승인 및 예산 확보
- 파일럿 프로젝트 시작 (Q1)
- 개발자 피드백 수집 및 개선
- 전체 플랫폼 통합 계획 수립

이러한 기술적 투자는 **STOVE 플랫폼의 개발자 생태계 확장**과 **경쟁력 강화**에 직접적으로 기여할 것입니다.