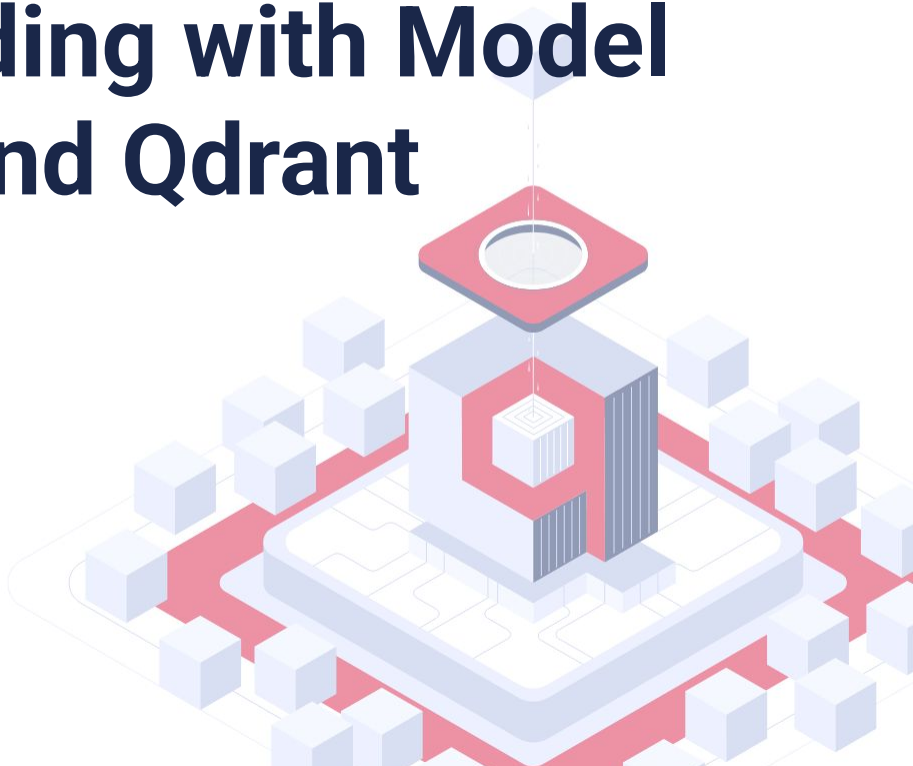
Grounded Vibe Coding with Model Context Protocol and Qdrant

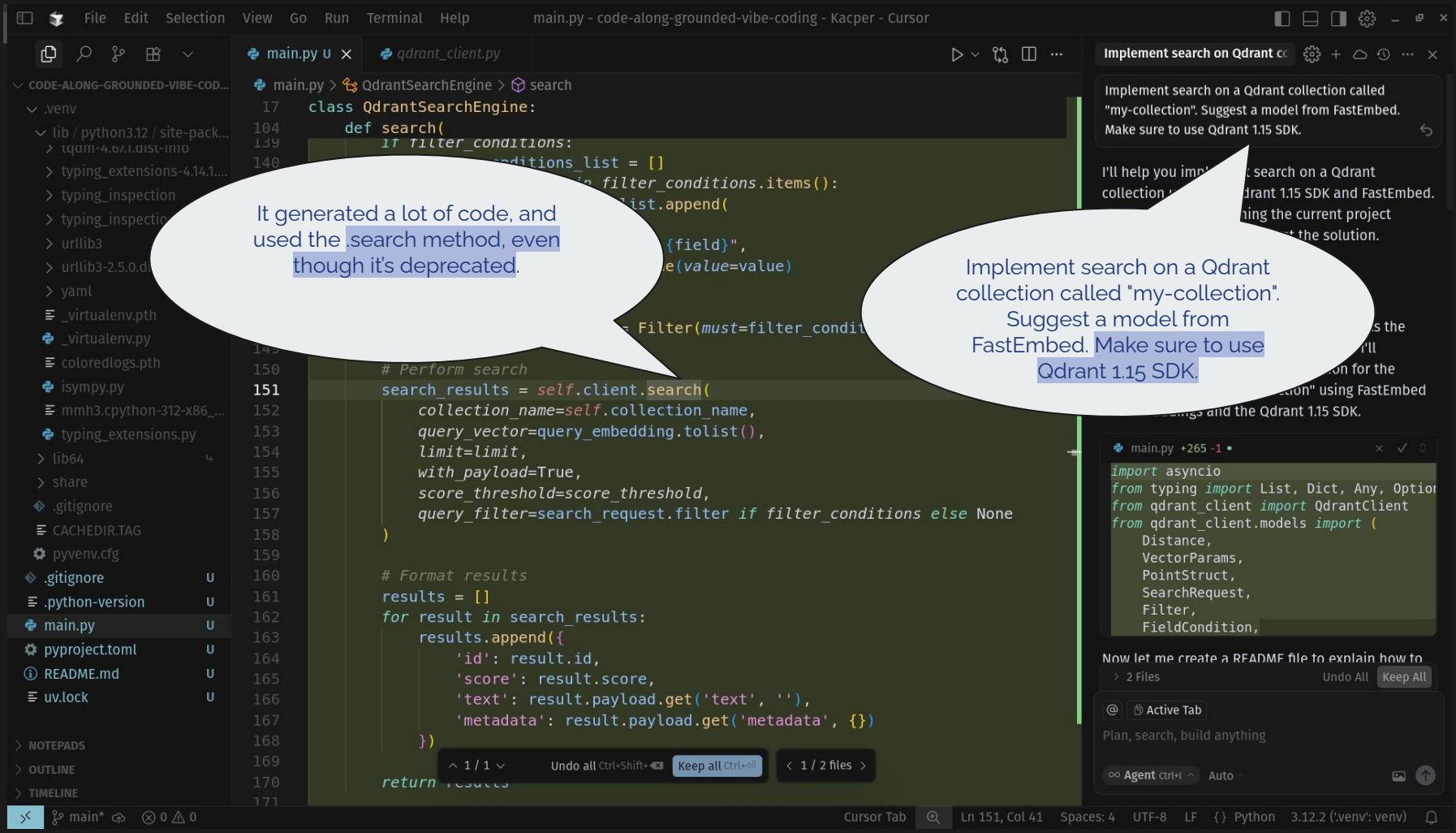
Kacper Łukawski, Senior Developer Advocate



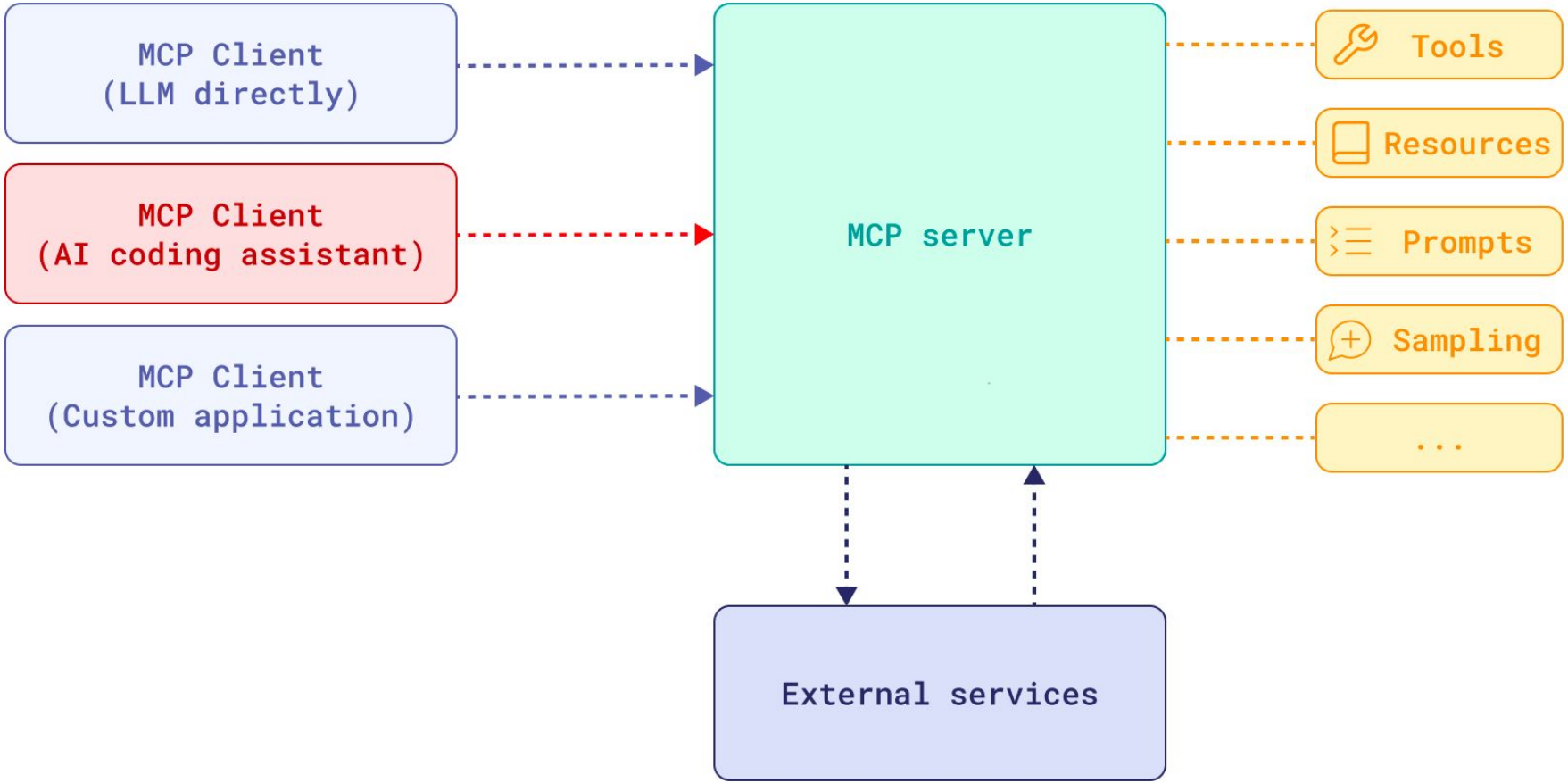
Andrej Karpathy
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

12:17 AM · Feb 3, 2025 · **4.4M** Views

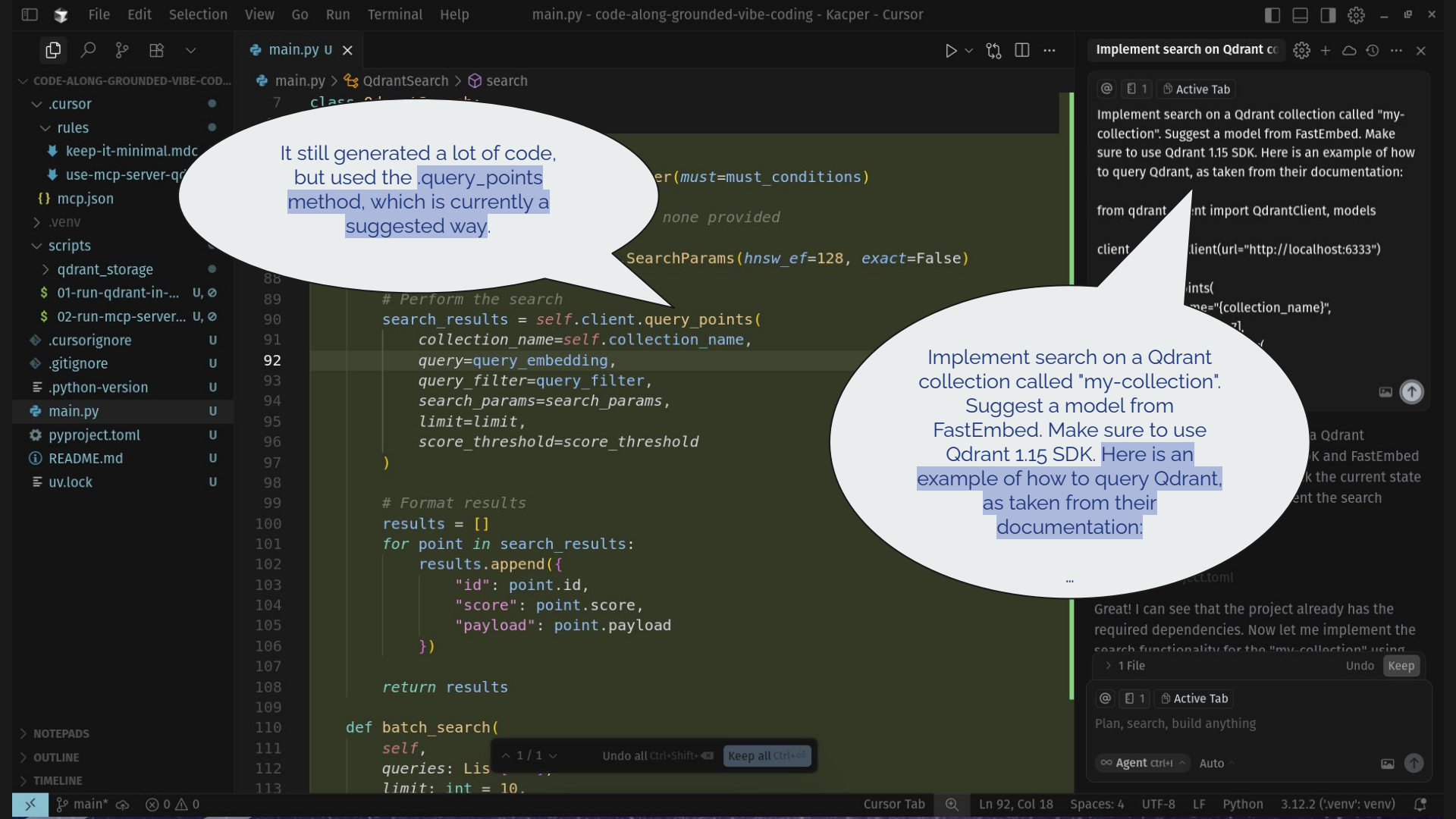

It generated a lot of code, and used the .search method, even though it's deprecated.
Implement search on a Qdrant collection called "my-collection". Suggest a model from FastEmbed. Make sure to use Qdrant 1.15 SDK.
File Edit Selection View Go Run Terminal Help
main.py - code-along-grounded-vibe-coding - Kacper - Cursor
main.py U
qdrant_client.py
main.py > QdrantSearchEngine > search
class QdrantSearchEngine:
def search(
# Perform search
search_results = self.client.search(
collection_name=self.collection_name,
query_vector=query_embedding.tolist(),
limit=limit,
with_payload=True,
score_threshold=score_threshold,
query_filter=search_request.filter if filter_conditions else None
# Format results
results = []
for result in search_results:
results.append({
'id': result.id,
'score': result.score,
'text': result.payload.get('text', ''),
'metadata': result.payload.get('metadata', {})
Implement search on Qdrant cc
Implement search on a Qdrant collection called "my-collection". Suggest a model from FastEmbed. Make sure to use Qdrant 1.15 SDK.
main.py +265 -1
import asyncio
from typing import List, Dict, Any, Optio
from qdrant_client import QdrantClient
from qdrant_client.models import (
Distance,
VectorParams,
PointStruct,
SearchRequest,
Filter,
FieldCondition,
2 Files
Undo All
Keep All
Active Tab
Plan, search, build anything
Agent
Auto
Undo all
Keep all
1 / 2 files
Cursor Tab
Ln 151, Col 41
Spaces: 4
UTF-8
LF
Python
3.12.2 ('.venv': venv)

Model Context Protocol


MCP Client
(LLM directly)
MCP Client
(AI coding assistant)
MCP Client
(Custom application)
MCP server
Tools
Resources
Prompts
Sampling
...
External services

Feature support matrix

Client	Resources	Prompts	Tools	Sampling	Roots	Notes
5ire	❌	❌	✅	❌	❌	Supports tools.
Apify MCP Tester	❌	❌	✅	❌	❌	Supports tools
BeeAI Framework	❌	❌	✅	❌	❌	Supports tools in agentic workflows.
Claude Code	❌	✅	✅	❌	❌	Supports prompts and tools
Claude Desktop App	✅	✅	✅	❌	❌	Supports tools, prompts, and resources.
Cline	✅	❌	✅	❌	❌	Supports tools and resources.
Continue	✅	✅	✅	❌	❌	Supports tools, prompts, and resources.
Copilot-MCP	✅	❌	✅	❌	❌	Supports tools and resources.
Cursor	❌	❌	✅	❌	❌	Supports tools.
Daydreams Agents	✅	✅	✅	❌	❌	Support for drop in Servers to Daydreams agents
Emacs Mcp	❌	❌	✅	❌	❌	Supports tools in Emacs.
fast-agent	✅	✅	✅	✅	✅	Full multimodal MCP support, with end-to-end tests
FLUJO	❌	❌	✅	❌	❌	Support for resources, Prompts and Roots are coming soon
Genkit	⚠️	✅	✅	❌	❌	Supports resource list and lookup through tools.
GenAIScript	❌	❌	✅	❌	❌	Supports tools.
Goose	❌	❌	✅	❌	❌	Supports tools.


It still generated a lot of code, but used the .query_points method, which is currently a suggested way.
Implement search on a Qdrant collection called "my-collection". Suggest a model from FastEmbed. Make sure to use Qdrant 1.15 SDK. Here is an example of how to query Qdrant, as taken from their documentation:
Great! I can see that the project already has the required dependencies. Now let me implement the
1 File
Undo
Keep
Plan, search, build anything
Agent
Auto

mcp-server-qdrant

mcp-server-qdrant

Public

Edit Pins | Unwatch 12 | Fork 114 | Starred 808

master | 12 Branches | 5 Tags | Go to file t | Add file | Code

joein new: bump to v0.8.0 (#73) ✓ 5a72373 · last month | 69 Commits

.github/workflows	Add mypy to pre-commit (#43)	3 months ago
src/mcp_server_qdrant	new: update fastmcp to 2.7.0 (#65)	last month
tests	new: update type hints (#64)	last month
.gitignore	Abstract the embedding providers	4 months ago
.pre-commit-config.yaml	Add mypy to pre-commit (#43)	3 months ago
.python-version	Initial commit	8 months ago
Dockerfile	Add Dockerfile and update README (#32)	4 months ago
LICENSE	Change license to Apache 2.0	4 months ago
README.md	Update the README to reflect the new default for FASTM...	last month
pyproject.toml	new: bump to v0.8.0 (#73)	last month
uv.lock	new: bump to v0.8.0 (#73)	last month

README | Apache-2.0 license

mcp-server-qdrant: A Qdrant MCP server

smithery.ai ▲ 116

> The Model Context Protocol (MCP) is an open protocol that enables seamless integration between LLM applications and external data sources and tools. Whether you're building an AI-powered IDE, enhancing a chat interface, or creating custom AI workflows, MCP provides a standardized way to connect LLMs with the context they need.

About

An official Qdrant Model Context Protocol (MCP) server implementation

qdrant.tech

mcp cursor semantic-search claude windsurf llm mcp-server

Readme
Apache-2.0 license
Activity
Custom properties
808 stars
12 watching
114 forks
Report repository

Releases 5

v0.8.0 Latest
on Jun 27

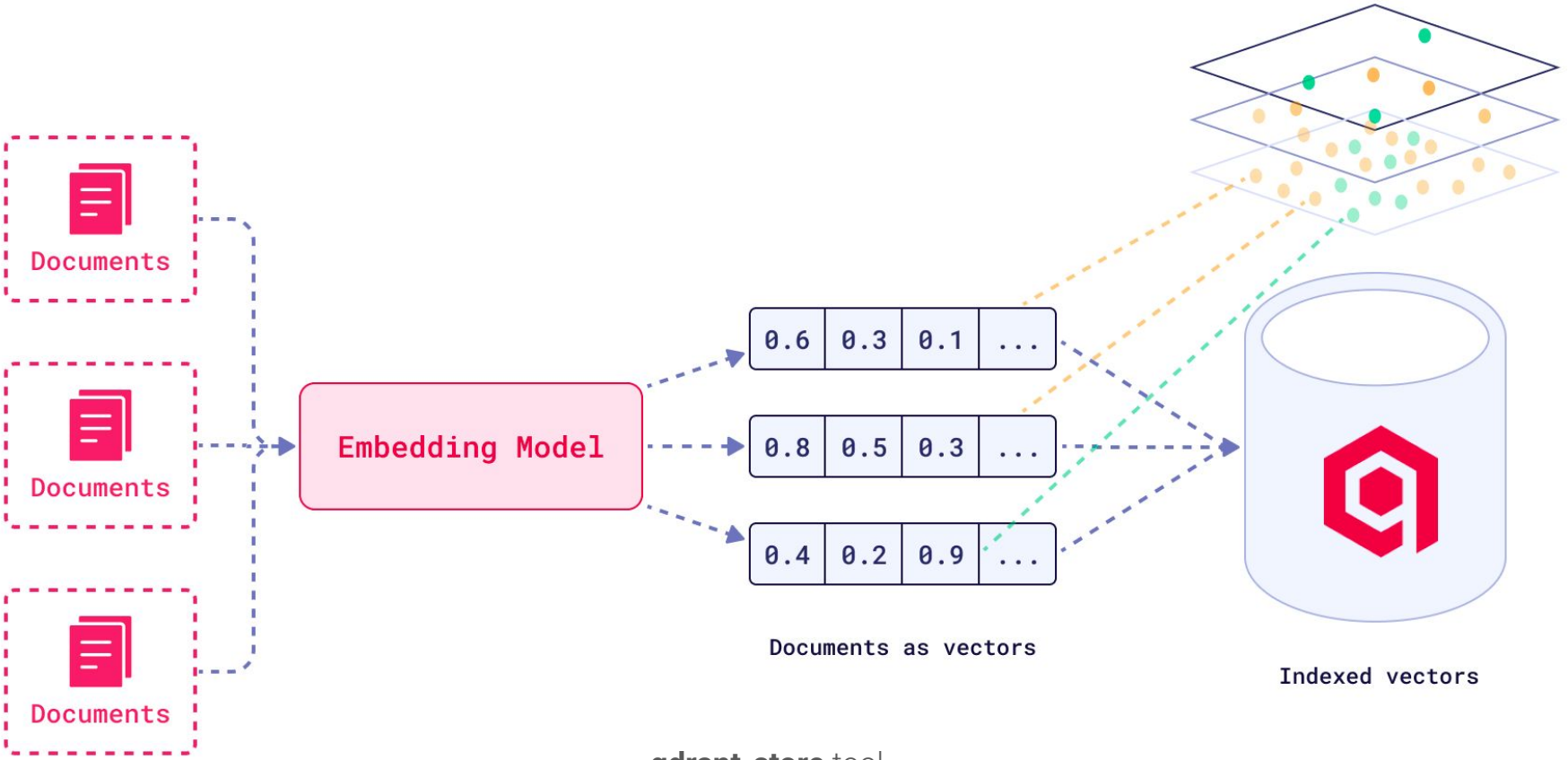
+ 4 releases

Packages

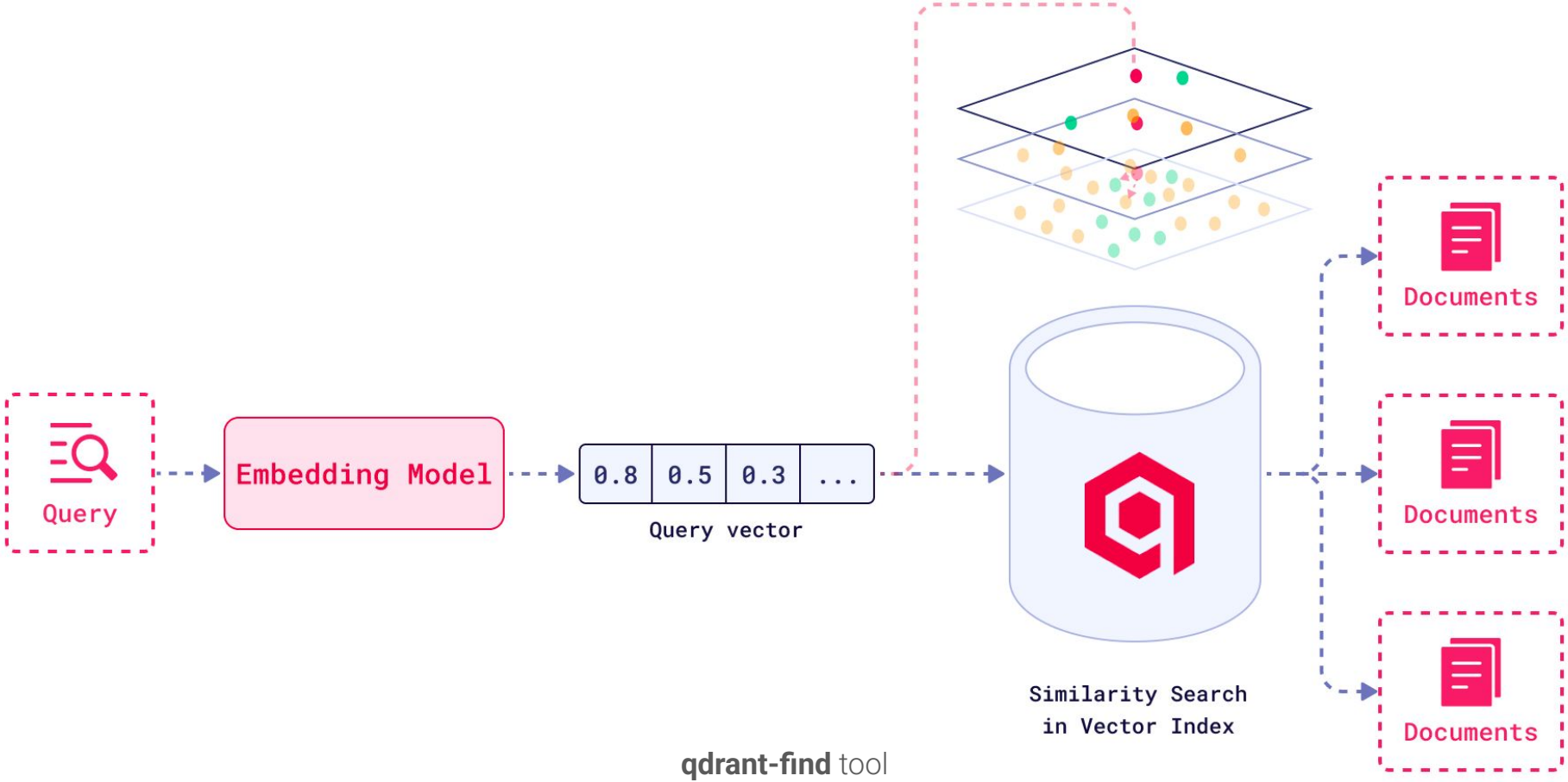
No packages published
Publish your first package

Contributors 6

Languages



qdrant-store tool



qdrant-find tool