MCP SERVER | NO CODE | CLOUD HOSTED

iFood MCP

Automate Order Flow and Menu Management

iFood MCP connects your AI agent directly to your restaurant's backend operations on iFood. Manage everything from real-time order flow and status updates to updating your full menu catalog, store hours, and delivery logistics—all through natural language commands.

A+ Quality Score 100/100

food-delivery | restaurant-operations | menu-management | order-tracking | real-time-sync

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/ifood

SCAN TO VIEW

The infrastructure that powers AI agents in the real world.

Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

iFood MCP

11 tools available
Cloud-hosted on Vinkius

Running a multi-location food business means juggling orders, menus, and schedules across multiple platforms. This MCP lets you automate all of that using only your AI agent. Instead of logging into the iFood portal to check statuses or manually changing prices, you simply ask your agent to perform the task. It reads incoming order details, helps you progress an order from pending through preparation, and tracks delivery assignments when needed. You can also update your store's operating hours for any day or modify specific items in your menu catalog without touching a single button. If managing multiple locations is complex, connecting iFood via Vinkius gives you one central point of control to handle all these operations.

Core Capabilities

01 — Monitor and progress orders

Watch incoming order statuses in real time, from listing new arrivals to updating the fulfillment status (e.g., accepting or preparing) through the pipeline.

02 — Manage store information

Retrieve detailed store configuration and list all associated locations using the `get_stores` tool before making any changes.

03 — Update menu pricing and availability

Change item prices, toggle availability, or update descriptions within your digital catalog using the `update_catalog_item` tool.

04 — Handle store scheduling

Set or change your operating hours for specific days of the week with the `update_business_hours` tool.

05 — Track deliveries and personnel

Get real-time logistics information, track order fulfillment progress, and assign drivers to orders using tools like `get_logistics` and `assign_driver`.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/ifood** — connect your AI agent in three steps.

01 You connect your iFood merchant account credentials through the Vinkius Marketplace.

02 You tell your AI agent exactly what you need to do, like 'Check all pending orders and update their status to prepared.'

03 The agent executes the necessary commands using the available tools, handling the order flow or catalog changes automatically.

The bottom line is that natural language instructions replace clicking through multiple complex operational dashboards.

Built For

This MCP is essential for restaurant owners and operations managers who spend too much time manually checking order statuses, updating menus, or managing store hours across different shifts. If you deal with more than a handful of orders daily, this saves critical labor time.

Restaurant Owner

Uses the MCP to quickly check if all locations have correct operating hours and ensures menu pricing is consistent across all iFood listings.

Operations Manager

Directs the agent to monitor order status changes, ensuring that every incoming order moves correctly through acceptance, preparation, and dispatch without manual oversight.

Ghost Kitchen Operator

Manages high-volume operations by automating catalog updates (e.g., marking out-of-stock items) and monitoring the flow of many orders simultaneously.

What Changes When You Connect

01 Stop wasting time clicking through status updates. You can use `get_orders` to pull all pending items, then tell your agent to update the status for each one using `update_order_status`.

02 Never manually update menus again. Use `get_catalog` first to review existing prices and descriptions; then use `update_catalog_item` to mass-adjust pricing or toggle availability instantly.

03 Manage multiple locations without switching portals. First, run `get_stores` to get all your facility IDs, and then target specific stores with tools like `update_business_hours`.

04 Improve dispatch coordination by using `assign_driver`. The agent handles the flow of assigning a delivery person once an order is ready for pickup.

05 Keep operations running smoothly even when staff are swamped. Your agent can pull necessary data, such as customer information from `get_order_details`, and draft confirmations automatically.

Real-World Applications

The morning rush of order acceptance

A manager needs to process 50 orders that came in overnight. Instead of opening the app, clicking each order, and updating its status manually, they ask their agent to 'Check for all pending orders and set them to ACCEPTED.' The agent runs `get_orders` and executes `update_order_status` on every item.

Updating prices due to ingredient shortages

The kitchen ran out of a key ingredient, meaning the 'Mega Burger' needs a price reduction and must be marked unavailable. The user asks their agent, which uses `get_catalog` to verify the item ID before executing `update_catalog_item` to change both the price and availability.

Handling store closures for holidays

The team needs to make sure all physical locations are correctly marked as closed for a public holiday. They ask their agent, which uses `get_stores` to find every location ID, then runs `update_business_hours` for each one with the correct schedule.

Coordinating last-minute logistics changes

A delivery driver calls and says they need a different pickup point. The manager asks their agent to use `get_logistics` to verify the current plan, then uses that data to communicate with the appropriate personnel.

Patterns to Avoid

Trying to update orders without checking status

✗ AVOID

A user just types 'Update order 123.' and expects it to happen. This fails because the agent doesn't know the current state of the order, risking incorrect data writes.

✓ INSTEAD

Always start by running `get_order_details` or `get_orders` first. Then, use that verified information to tell your agent to run `update_order_status` on the correct item.

Updating menus without listing all stores

✗ AVOID

A user updates a price for one location's catalog but forgets about the other three branches, leaving inconsistent pricing and confusing customers.

✓ INSTEAD

First, run `get_stores` to get every store ID. Then, use that list of IDs as context when performing any bulk update using `update_catalog_item` or `update_business_hours`.

Assuming the delivery driver is assigned

✗ AVOID

The user simply tells the agent to 'send the order out' without ensuring a person is ready for pickup, which causes delays and confusion.

✓ INSTEAD

Always use `get_logistics` first to check if tracking information exists. If not, tell your agent to run `assign_driver` with both the required Order ID and the driver ID.

The Right Fit

Use this MCP if your primary pain point is operational consistency across multiple physical locations or handling high-volume order processing flow. You need to move data from a manual, dashboard-based workflow (checking status in 10 different tabs) into conversational commands. Don't use it if you only need to read static information—if you just want to know the current menu

catalog, `get_catalog` is enough. However, if you need to *change* that catalog or *act* on an order (like updating status), this MCP is necessary. If your needs are purely related to payment processing or inventory stock counts outside of iFood's structure, look for a dedicated Inventory Management system connector instead.

Dealing with the 'Where Did That Order Go?' Chaos

Every day, restaurant managers spend hours logging into different portals just to track one thing: an order. They have to click through status screens, copy IDs, cross-reference times, and manually update every step—from 'Pending' to 'In Preparation,' which is tedious, repetitive work that kills productivity.

With this MCP, you ask your agent what happened with the order instead of clicking it into existence. The system uses tools like `get_orders` and `get_order_details` to pull the status immediately, giving you a clear narrative flow of exactly where the order is right now.

iFood MCP: Complete Control Over Your Operations

The biggest time sinks are updating core data points: changing prices or setting business hours. Traditionally, this meant logging into the merchant portal and navigating complex forms for every single location, risking human error.

Now, you tell your agent to 'Set our hours from 10 AM to 10 PM next week.' It handles finding all locations via `get_stores` and executes the change using `update_business_hours`. The complexity disappears.

iFood: 11 Tools for Operations Management

These tools let you programmatically handle every aspect of your restaurant's iFood presence, including order flow, catalog changes, and scheduling.

#	TOOL	DESCRIPTION
01	assign_driver	Assigns a specific delivery driver to an iFood order ID to coordinate fulfillment.
02	get_business_hours	Retrieves the current operating hours for a specified iFood store location.
03	get_catalog	Provides a full review of the menu structure, allowing you to check item availability or plan price updates.
04	get_logistics	Tracks the delivery progress and provides logistics information for an existing iFood order.
05	get_order_details	Retrieves complete details, including customer info and item totals, for a single specified iFood order.
06	get_orders	Lists all orders for the store, filtering by status like PENDING or ACCEPTED, to track overall order volume.
07	get_store_details	Fetches detailed configuration and information about a specific iFood store location.
08	get_stores	Lists all associated store IDs, names, and addresses linked to the merchant account.
09	update_business_hours	Modifies the operating schedule for a store across multiple days of the week with a structured JSON payload.
10	update_catalog_item	Allows you to change specific fields like prices, availability, or descriptions for an existing menu item.
11	update_order_status	Moves a specific order through the fulfillment pipeline by updating its status (e.g., from PENDING to DISPATCHED).

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U Show me all pending orders and accept them automatically

I'll check your pending orders and accept them right away.

U Update the price of 'Combo Burger Especial' to R$32.90 and mark it as unavailable until tomorrow

I'll update the item price and toggle its availability in your catalog.

U Change my store hours to open at 11am and close at 11pm every day this week

I'll update your business hours for the entire week right now.

Frequently Asked Questions

01 **How do I check if an order is ready for dispatch using iFood MCP?**

You use the `get_order_details` tool to review all contents and status. If everything looks good, you can ask your agent to run `update_order_status` to move it to DISPATCHED.

02 **Can iFood MCP help me update prices for multiple menu items?**

Yes. You use the `get_catalog` tool first to view all item IDs, then direct your agent to execute `update_catalog_item` with a list of IDs and their new prices.

03 **What if I need to change my store hours for only one day?**

You can use the `get_business_hours` tool to view the current schedule, then instruct your agent to run `update_business_hours` specifying the exact date and times needed.

04 **How does iFood MCP handle multiple store locations?**

It first requires you to use `get_stores` to list all location IDs. Then, any subsequent operation like setting hours or viewing details must reference those specific IDs.

05 **Can I track a driver assignment after an order is placed?**

Yes, run the `get_logistics` tool with the order ID. This provides current tracking data and helps coordinate fulfillment efforts.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"ifood": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

iFood is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by iFood. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	iFood MCP
Server ID	`019d75b7-f00c-7341-a008-39eee1801867`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/ifood.

© 2026 Vinkius · vinkius.com · support@vinkius.com