ChatGPT

# Plan de Desarrollo e Implementación de la Integración GPT ↔ ServiceDesk Plus (MCP)

## 0 · Objetivo

Permitir que los modelos de OpenAI (p.ej. **o3**) puedan **crear, consultar y anotar tickets** de ServiceDesk Plus (`https://helpdesk.frv.com`) mediante un **servidor MCP** interno seguro y estandarizado.

## 1 · Requisitos previos

1. **API Key de ServiceDesk Plus**
2. *Ruta:* **Admin → Technicians** → técnico-bot → **Generate API Key**.
3. Guardar en gestor de secretos como `SDP_API_KEY`.
4. **Acceso de red**
5. El host del micro-servicio debe resolver `helpdesk.frv.com` (TCP 443).
6. Puerto libre **8001** para exponer MCP.
7. **Cuenta OpenAI** con acceso a modelos *o-series* y **Responses API**.
8. **Stack de ejecución**

```
Python 3.11
poetry 1.8
Docker 24.x
kubectl / systemd
```

## 2 · Estructura de proyecto

```
helpdesk-mcp/
├── README.md
├── pyproject.toml
├── .env.example
├── server.py
├── tools_schema.py
├── tests/
│   ├── unit/
│   └── integration/
└── infra/
    ├── Dockerfile
    └── k8s-deployment.yaml
```

## 3 · Fase 1 · Bootstrap del servicio (1 día)

```
git clone ssh://gitlab.frv.com/ai/helpdesk-mcp.git
cd helpdesk-mcp

poetry init -n
poetry add fastapi uvicorn openai-mcp==0.5.* requests python-dotenv
poetry add --group dev pytest pytest-cov black isort

cp .env.example .env          # rellenar secretos
git add .
git commit -m "chore: initial scaffolding"
```

## 4 · Fase 2 · Implementación de las tools (2 días)

### 4.1 Definir JSON Schema (`tools_schema.py`)

```
create_ticket_schema = {
    "name": "create_ticket",
    "description": "Crea un ticket en ServiceDesk Plus",
    "parameters": {
        "type": "object",
        "properties": {
            "subject": {"type": "string"},
            "description": {"type": "string"},
            "priority": {
                "type": "string",
                "enum": ["Low", "Medium", "High", "Urgent"]
            },
            "site": {"type": "string"}
        },
        "required": ["subject"]
    }
}
```

*Definir esquemas similares para ``.*

### 4.2 Servidor MCP (`server.py`)

```
import os, json, requests
from fastapi import FastAPI
from openai_mcp import MCPServerHTTP
from dotenv import load_dotenv
from tools_schema import create_ticket_schema, get_ticket_schema,
add_note_schema

load_dotenv()
```

```
SDP_API_KEY = os.environ["SDP_API_KEY"]
SDP_URL     = os.getenv("SDP_URL", "https://helpdesk.frv.com/api/v3")

def call_sdp(method, path, payload=None):
    headers = {"Authtoken": SDP_API_KEY, "Content-Type": "application/json"}
    resp = requests.request(method, f"{SDP_URL}{path}", headers=headers,
                            data=json.dumps(payload) if payload else None,
                            timeout=15)
    resp.raise_for_status()
    return resp.json()

class HelpdeskServer(MCPServerHTTP):
    def list_tools(self):
        return [create_ticket_schema, get_ticket_schema, add_note_schema]

    def call_tool(self, name, arguments):
        if name == "create_ticket":
            data = {"request": arguments}
            res = call_sdp("POST", "/requests", data)
            return {"ticket_id": res["request"]["id"]}
        if name == "get_ticket":
            rid = arguments["ticket_id"]
            return call_sdp("GET", f"/requests/{rid}")
        if name == "add_note":
            rid = arguments["ticket_id"]
            note = {"note": {"description": arguments["note_text"]}}
            res = call_sdp("POST", f"/requests/{rid}/notes", note)
            return {"note_id": res["note"]["id"]}

app = FastAPI()
app.mount("/", HelpdeskServer())
```

### 4.3 Logs y errores

- `logging.basicConfig(level=os.getenv("LOG_LEVEL", "INFO"))`
- Capturar `requests.exceptions.Timeout` y devolver `{"error": "timeout"}`.

### 4.4 Pruebas unitarias

*Usar ``.*

## 5 · Fase 3 · Pruebas End-to-End (1 día)

```
MCP_PORT=8001 poetry run uvicorn server:app --reload --port $MCP_PORT
```

```
import openai, os
openai.api_key = os.getenv("OPENAI_API_KEY")
resp = openai.responses.create(
    model="o3",
```

```
    tools=[{
        "type": "mcp",
        "server_label": "frv_helpdesk",
        "server_url": f"http://localhost:{os.environ['MCP_PORT']}"
    }],
    input="Crea un ticket urgente: 'VPN caída site Madrid'"
)
print(resp)
```

**Aceptación:** se devuelve `ticket_id` y el ticket existe con prioridad *High*.

## 6 · Fase 4 · Contenerización y despliegue (1 día)

### 6.1 Dockerfile

```
FROM python:3.11-slim
WORKDIR /app
COPY pyproject.toml poetry.lock* /app/
RUN pip install poetry && poetry install --no-dev
COPY . /app
CMD ["uvicorn", "server:app", "--host", "0.0.0.0", "--port", "8001"]
```

### 6.2 Despliegue Kubernetes (`infra/k8s-deployment.yaml`)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helpdesk-mcp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: helpdesk-mcp
  template:
    metadata:
      labels:
        app: helpdesk-mcp
    spec:
      containers:
      - name: server
        image: registry.frv.com/ai/helpdesk-mcp:1.0.0
        envFrom:
        - secretRef:
            name: helpdesk-mcp-secrets
        ports:
        - containerPort: 8001
        readinessProbe:
          httpGet:
```

```
            path: /healthz
            port: 8001
          periodSeconds: 5
---
apiVersion: v1
kind: Service
metadata:
  name: helpdesk-mcp
spec:
  selector:
    app: helpdesk-mcp
  ports:
  - port: 8001
    targetPort: 8001
```

## 7 · Fase 5 · Observabilidad (½ día)

| Componente | Acción |
|---|---|
| **Logging** | Enviar stdout a Loki/Elastic. |
| **Métricas** | Exponer `/metrics` (Prometheus), métrica `mcp_request_latency_seconds`. |
| **Alerting** | `p95(mcp_request_latency_seconds) > 5` s durante 5 min ⇒ alerta. |

## 8 · Fase 6 · Seguridad y control de acceso (½ día)

1. Secretos como **Kubernetes Secret** o `systemd-creds`.
2. **NetworkPolicy**: sólo tráfico TCP 8001 desde la subred AI.
3. Auditoría: registrar `user`, `tool_name`, `arguments` (ofuscando PII).

## 9 · Fase 7 · Puesta en producción & rollback (½ día)

1. **Blue/Green**: desplegar v1.0.0 como “green”, enrutar 10 % del tráfico.
2. Promover a 100 % tras 1 h sin errores ≥ 1 %.
3. Rollback automático si `5xx > 2 %` en 10 min.

## 10 · Mantenimiento continuo

| Frecuencia | Tarea |
|---|---|
| Semanal | Revisar logs y latencia. |
| Mensual | `poetry update` de dependencias. |
| Trimestral | Versionar nuevas tools ( `v2` , …). |
| Ad-hoc | Añadir tests para nuevos campos de SDP. |

## Checklist final

-

---

**Resultado:** Integración reproducible, segura y lista para producción entre GPT y ServiceDesk Plus usando MCP.