Документация по разработке плагинов для системы умного дома

# 1. Что такое плагин?

Плагин — это модуль, который регистрирует устройство и отправляет его данные в state для отображения в веб-интерфейсе.

# 2. Как создать плагин?

Создай папку в modules/:

modules/  
└── my\_plugin/  
 ├── \_\_init\_\_.py # Пустой файл  
 └── plugin.py # Логика плагина

Напиши plugin.py. Минимальный пример:

from core.interfaces.plugin\_interface import PluginInterface  
from core.logger import logger  
  
class Plugin(PluginInterface):  
 def initialize(self, event\_bus, state):  
 self.event\_bus = event\_bus  
 self.state = state  
 # Добавляем устройство в список  
 devices = self.state.get\_data("devices") or []  
 if "Моё устройство" not in devices:  
 devices.append("Моё устройство")  
 self.state.set\_data("devices", devices)  
 # Добавляем данные устройства  
 sensor\_data = self.state.get\_data("sensor\_data") or {}  
 sensor\_data["Моё устройство"] = {"example": 42}  
 self.state.set\_data("sensor\_data", sensor\_data)  
  
 def register(self):  
 pass # Подписка на события, если нужно

# 3. Как добавить данные?

Используй self.state.set\_data("sensor\_data", {...}) для записи данных. Данные должны быть словарем, где ключ — имя устройства, а значение — словарь с его метриками.

Пример:

sensor\_data = self.state.get\_data("sensor\_data") or {}  
sensor\_data["Моё устройство"] = {"temperature": 25.0}  
self.state.set\_data("sensor\_data", sensor\_data)

# 4. Пример реального плагина: Мониторинг ПК

import asyncio  
import psutil  
from core.interfaces.plugin\_interface import PluginInterface  
from core.logger import logger  
  
class Plugin(PluginInterface):  
 def \_\_init\_\_(self):  
 self.pc\_name = "Мой ПК"  
  
 def initialize(self, event\_bus, state):  
 self.event\_bus = event\_bus  
 self.state = state  
 # Добавляем устройство  
 devices = self.state.get\_data("devices") or []  
 if self.pc\_name not in devices:  
 devices.append(self.pc\_name)  
 self.state.set\_data("devices", devices)  
 # Инициализируем данные  
 self.update\_sensor\_data()  
 asyncio.create\_task(self.monitor\_pc())  
  
 def register(self):  
 pass  
  
 async def monitor\_pc(self):  
 while True:  
 self.update\_sensor\_data()  
 logger.info(f"Метрики ПК обновлены")  
 await asyncio.sleep(10)  
  
 def update\_sensor\_data(self):  
 sensor\_data = self.state.get\_data("sensor\_data") or {}  
 sensor\_data[self.pc\_name] = {  
 "cpu\_usage": round(psutil.cpu\_percent(interval=1), 1),  
 "memory\_usage": round(psutil.virtual\_memory().percent, 1),  
 "status": "Онлайн"  
 }  
 self.state.set\_data("sensor\_data", sensor\_data)

# 5. Полезные советы

Логи: Пиши сообщения через logger.info("Сообщение").

События: Подписывайся через self.event\_bus.subscribe("event\_name", self.handler).

Фоновая работа: Используй asyncio.create\_task() для периодических задач.

Ошибки: Обрабатывай исключения.

# 6. Что дальше?

1. Создай папку в modules/.

2. Напиши plugin.py с классом Plugin.

3. Добавь устройство в devices и данные в sensor\_data.

4. Перезапусти main.py — готово!