# **👨‍💻 ملخص للمطورين: معمارية وتصميم G-Assistant**

مشروع G-Assistant مصمم بمعمارية معيارية (Modular Architecture) قوية تهدف إلى سهولة الصيانة، التوسعة، وفصل الاهتمامات. يعتمد المشروع بشكل كبير على نمط IIFE (Immediately Invoked Function Expression) لتغليف الوحدات، مما يوفر نطاقًا خاصًا لكل وحدة ويمنع تضارب الأسماء.

## **🧠 المعمارية الأساسية**

1. **الفصل الواضح للطبقات:**
   * **طبقة الواجهة (UI Layer):** يتم التحكم فيها بواسطة ui.gs وملفات HTML (مثل Sidebar.html, BuiltinDocsSidebar.html, WorkshopSidebar.html). هذه الطبقة مسؤولة عن التفاعل مع المستخدم وعرض المعلومات.
   * **طبقة المنطق الأساسي للذكاء الاصطناعي (AI Core Logic):** تتمثل في ai\_core.gs (محرك الاتصال بـ Gemini API)، ai\_config.gs (الإعدادات)، ai\_context.gs (بناء السياق)، ai\_memory.gs (إدارة الذاكرة)، وai\_dispatcher.gs (توجيه النية واستدعاء الأدوات). هذه هي "عقل" المساعد.
   * **طبقة الأدوات (Tools Layer):** مجموعة من الوحدات المتخصصة (مثل tools\_sheets.gs, tools\_accounting.gs, tools\_code\_review.gs, tools\_ai\_helpers.gs). كل وحدة توفر مجموعة من الدوال التي يمكن لنموذج Gemini استدعاؤها لتنفيذ مهام محددة في Google Sheets أو غيرها.
   * **طبقة المساعدات العامة (Helpers Layer):** تتمثل في helpers.gs، وتقدم دوال مساعدة عامة مثل تسجيل العمليات، معالجة الأخطاء، وتنسيق البيانات.
   * **طبقة التكوين والبيئة (Configuration & Environment):** تتمثل في ENV.gs وSETTINGS\_PROVIDER.gs، وتوفر إعدادات النظام العامة والخاصة بالذكاء الاصطناعي.
2. **استدعاء الدوال (Function Calling):**
   * يستخدم المشروع ميزة function\_calling المتقدمة في Gemini API.
   * يتم تعريف جميع الأدوات المتاحة لـ Gemini في Tools Catalog.gs، والذي يولد Function Declarations المطلوبة لـ API.
   * ai\_dispatcher.gs هو المسؤول عن تحليل استجابات Gemini وتحديد الأداة المناسبة للاستدعاء وتنفيذها.
3. **إدارة السياق والذاكرة:**
   * ai\_context.gs يجمع المعلومات السياقية الهامة (دور المستخدم، الخلية النشطة، سجل العمليات) لتقديمها لنموذج Gemini.
   * ai\_memory.gs يدير ذاكرة الجلسة (قصيرة المدى) وسجل العمليات (طويلة المدى) لتمكين المحادثات المستمرة وتتبع الأداء.
4. **معالجة الأخطاء والتسجيل:**
   * يتم استخدام helpers.gs.logOperation() لتسجيل جميع العمليات الهامة، مما يسهل تتبع الأداء وتصحيح الأخطاء.
   * AI\_TOOLS\_DIALOGUE (الموجودة ضمن tools\_dialogue.gs) توحد تنسيق رسائل المستخدم (نجاح، خطأ، تحذير، معلومات) لضمان تجربة مستخدم متسقة.

## **💡 نقاط تصميم رئيسية**

* **المرونة والتوسعة:** يسهل الهيكل المعياري إضافة أدوات جديدة أو تعديل الأدوات الموجودة دون التأثير على أجزاء أخرى من النظام.
* **فصل الواجهة عن المنطق:** يتم فصل منطق الأعمال عن الواجهة الأمامية، مما يجعل الواجهة أكثر خفة وسهولة في التعديل.
* **الاستفادة من Gemini AI:** يتم استخدام نماذج Gemini بذكاء لمجموعة واسعة من المهام، من التحليل النصي إلى توليد الكود والصيغ.
* **التعامل مع البيانات المنظمة:** يتم استخدام responseMimeType: "application/json" و responseSchema في طلبات Gemini للحصول على استجابات منظمة، مما يسهل معالجة البيانات برمجيًا.

## **🛠️ أدوات التطوير**

* **ورشة العمل البرمجية (**workshop.gs**):** تساعد المطورين على تحليل الكود، اقتراح الملفات المناسبة له، وتجهيزه للمراجعة.
* **أدوات مراجعة الكود (**tools\_code\_review.gs**):** توفر قدرات مراجعة الكود، شرحه، توليد الاختبارات والتوثيق، وتحسين الأداء باستخدام الذكاء الاصطناعي.
* **نظام الاختبارات (**tests.gs**):** يضمن جودة الكود من خلال اختبارات الوحدة والتكامل.

هذا الملخص يقدم نظرة عامة على الجوانب الهندسية للمشروع، مما يساعد المطورين الجدد على فهم بنيته والمساهمة فيه بفعالية.