مختصر كتابي

# شغّل مساعد Seagull وراقب ما يفعله



## طلبات مَن تقرؤها الـTool، ومَن يقرّر ذلك؟

في هذا المشروع تبني **مساعد Seagull** لشركة **Seagull للتوزيع** (شركة أدوات مكتبية افتراضية بياناتها للتدريب)، لتستخدمه مريم في خدمة العملاء. هدف القسم الأول أن يجيب المساعد من **الـDatabase** لا من التخمين، وهو ما يُقاس في نهايته على 10 أسئلة.

تتلقّى مريم مكالمة من «مكتبة الفجر — فرع مدينة نصر» ( `C200` )، بعد أن تحقّق نظام الهاتف من هوية **الـCaller**. يسأل الـCaller عن طلبين: الطلب `4800` على حسابه وبلا موعد توصيل مؤكد، والطلب `4801` الذي يدّعي أنه على حساب فرع المعادي ( `C201` ) — لكن `C201` عميل آخر مستقل في الـDatabase.

يكتب **Claude Haiku** الرد، ولأنه لا يعرف طلبات الشركة أعطيناه **الـTool** المسماة `read_order` . يرسل **الـModel** طلب **الـTool Call** فقط، وينفّذ كودنا الـTool ثم يعيد **الـTool Result** بالمعرّف نفسه؛ وبذلك يقرّر الـModel الخطوة بينما ينفّذ الكود العمل — وهذا هو **الـAgent**.

في النسخة الأولى أرسل الـModel رقم الطلب ورقم العميل معًا، وإذا اختلف العميل تعيد الـTool ردّ `forbidden` . في 5 من 5 تشغيلات حقيقية على Claude Haiku أرسل الـModel طلبَي Tool Call في اللفة نفسها برقم `C200` ؛ لكن ردّ `forbidden` دفع الـModel في 5 من 5 تشغيلات لإخبار مريم بأن الطلب `4801` ليس على هذا الحساب، ما يكشف وجوده لدى عميل آخر. وعندما طلبت مريم مراجعة `4801` على حساب `C201` ، أرسل الـModel الرقم `C201` في 5 من 5 تشغيلات، فظهرت حالة الطلب وموعده (19 نوفمبر) أمام مريم — وفي تشغيلتين من 5 نبّهها الرد بعد ظهور الموعد.

الخلل ليس في الـModel بل في أن رقم العميل جاء منه، وكلام المحادثة يغيّر أي قيمة يكتبها الـModel. لذا ينتقل رقم الـCaller إلى **الـSession** التي يضبطها كودنا قبل المحادثة، وتقتصر **الـSchema** على `order_id` فقط، ليعمل الكود كـ**Contract** صريح يرفض المدخل الخاطئ قبل الـDatabase ويدمج شرط العميل في استعلام واحد:

```
def read_order(tools, args):
    if not isinstance(args, dict) or set(args) != {"order_id"}:
        return {"error": "bad_arguments"}
    order_id = args["order_id"]
    if type(order_id) is not int or not 0 < order_id <= MAX_ORDER_ID:
        return {"error": "bad_order_id"}
    customer = tools.customer
    if not customer:
        return dict(NOT_FOUND)
    row = tools.db.execute("SELECT id,status,eta FROM orders WHERE id=? AND
customer_id=?",
                           (order_id, customer)).fetchone()
    if row is None:
        return dict(NOT_FOUND)
    status = (row[1] or "").strip().lower()
    eta = _confirmed_eta(row[2])
    return {"found": True, "order_id": row[0], "customer_id": customer,
            "status": status if status in ORDER_STATUSES else "unknown",
            "eta": eta, "eta_confirmed": eta is not None}
```

| المقياس (5 تشغيلات على Claude Haiku 5.5) | النسخة الأولى (من الـModel) | النسخة الثانية (من الـSession) |
|---|---|---|
| تسريب بيانات 4801 عند سؤال مريم عن C201 | 5 من 5 | 0 من 5 |
| كشف وجود الطلب 4801 على حساب آخر | 5 من 5 | 0 من 5 |
| متوسط الـTokens (داخل / خارج) والتكلفة | 652 / 2,510 (≈ $0.0006) | 766 / 2,550 (≈ $0.0006) |

- في الـAgent يطلب الـModel الـTool عبر الـTool Call، وينفّذ كودنا ويعيد الـTool Result.
- ردّ `forbidden` يكشف وجود الطلب؛ لذا يجب أن يطابق ردّ طلب عميل آخر ردّ الطلب غير الموجود.
- تأتي هوية الـCaller من الـSession في الكود لا من مدخلات الـModel.
- عدم استخدام `int()` يمنع قبول الأرقام العربية والمسافات ويكشف أخطاء الـModel قبل الـDatabase.
- تعمل الـTests على **الـMock** (Model ردوده مكتوبة مسبقًا) لنتيجة ثابتة ومجانية مع حالات مخفية.

جرّب في ملفات المعمل (مجلد الدرس في Labs_00): شغّل سؤال مريم على النسخة الأولى، ثم عدّل `read_order` والـSchema لتطبيق الـContract واجتياز الـTests الـ13.

**الخلاصة:** في بناء أي Agent يقرّر الـModel الخطوة، لكن الكود هو الذي يحدّد ما يستطيع رؤيته.