



Ubaldo tiene il filo della conversazione

Il terzo agente con ADK: sessione e stato

Nella seconda guida abbiamo dato a Ubaldo due strumenti per leggere la dispensa e aggiornarla dopo aver cucinato. Ora gli insegneremo a gestire le informazioni sulla cena che stiamo organizzando: quante persone mangiano, quali preferenze alimentari hanno e quanto tempo è disponibile.

Durante una conversazione queste informazioni possono cambiare. Se passiamo da quattro a sei persone, Ubaldo deve aggiornare il numero dei commensali senza perdere le altre preferenze.

Per farlo useremo lo **stato della sessione**: uno spazio in cui registrare dati che il programma può leggere e aggiornare durante la conversazione.

Questa guida prosegue dalla precedente. Useremo la stessa cartella `adk-svuota-frigo` e lo stesso ambiente Python, creando un nuovo progetto chiamato `step3_sessione`. Le prime due versioni resteranno disponibili.

1. Apri il terminale e crea il terzo progetto

Su Windows, apri il menu Start, cerca **Ubuntu** e avvialo. Su Linux o macOS, apri il terminale che usi normalmente.

Entra nella cartella principale della serie:

```
cd ~/adk-svuota-frigo
```

Attiva l'ambiente virtuale:

```
source .venv/bin/activate
```

Questo passaggio va ripetuto ogni volta che apri un nuovo terminale. Se l'attivazione è riuscita, all'inizio della riga vedrai `(.venv)`. Non devi ricreare l'ambiente né reinstallare ADK.

Crea il progetto:

```
adk create step3_sessione
```

Seleziona `gemini-3.8-flash`:

```
Choose a model for the root agent:
1. gemini-3.5-flash
2. gemini-3.8-flash
3. Other models (fill later)
Choose model (1, 2, 3): 2
```

Poi scegli **Google AI**, inserendo 1:

```
1. Google AI
2. Vertex AI
3. Login with Google
Choose a backend (1, 2, 3): 1
```

Quando richiesto, inserisci la chiave API di Google AI Studio già utilizzata nelle guide precedenti.

Copiamo dal secondo progetto il file con gli strumenti della dispensa:

```
cp step2_dispensa/tools.py step3_sessione/tools.py
```

Il comando `cp` copia il file nella nuova cartella. Il file originale del secondo esempio rimane invariato.

Apri il nuovo progetto con VSCode:

```
cd step3_sessione
codium .
```

Lavoreremo su quattro file:

- `dispensa.json`: contiene gli ingredienti disponibili.



- tools.py: contiene i due strumenti della seconda puntata, appena copiati.
- preferenze.py: conterrà i nuovi strumenti per leggere e aggiornare le preferenze.
- agent.py: definisce Ubaldo, le sue istruzioni e gli strumenti disponibili.

2. Prepara la dispensa

Crea dispensa.json nella stessa cartella di agent.py e inserisci:

```
{
  "uova": {
    "quantita": 12,
    "unita": "pezzi"
  },
  "zucchine": {
    "quantita": 1200,
    "unita": "g"
  },
  "ceci cotti": {
    "quantita": 1000,
    "unita": "g"
  },
  "rosmarino": {
    "quantita": 6,
    "unita": "rametti"
  }
}
```

Abbiamo aumentato le scorte per poter provare ricette per più persone.

Il file tools.py cerca la dispensa nella propria cartella. Questa terza versione userà quindi il nuovo file, senza modificare quello del secondo esempio.

Salva il file.

3. Crea gli strumenti per le preferenze

Crea preferenze.py e inserisci questo codice:

```
from google.adk.tools import ToolContext

def leggi_preferenze(tool_context: ToolContext) -> dict:
    """Legge le preferenze della cena nella sessione corrente.

    Returns:
        Numero di commensali, alimentazione e tempo disponibile.
        I dati non ancora dichiarati sono restituiti come null.
        Il tempo predefinito è 15 minuti.
    """
    return {
        "stato": "ok",
        "preferenze": {
            "commensali": tool_context.state.get("commensali"),
            "alimentazione": tool_context.state.get("alimentazione"),
            "tempo_minuti": tool_context.state.get("tempo_minuti", 15),
        },
    }

def aggiorna_preferenze(
    tool_context: ToolContext,
    commensali: int = 0,
    alimentazione: str = "",
    tempo_minuti: int = 0,
) -> dict:
    """Aggiorna soltanto le preferenze esplicitamente indicate.

    Usa questo strumento per decisioni certe dell'utente.
    Non trasformare ipotesi o possibilità in preferenze confermate.
```



```
Args:
    commensali: Numero totale aggiornato di persone, non un incremento.
                Usa 0 oppure ometti il parametro per lasciare il dato invariato.
    alimentazione: "vegetariana" oppure "nessuna".
                "nessuna" significa nessuna preferenza alimentare dichiarata.
                Usa una stringa vuota oppure ometti il parametro
                per lasciare il dato invariato.
    tempo_minuti: Tempo totale disponibile per preparazione e cottura.
                Usa 0 oppure ometti il parametro per lasciare il dato invariato.

Returns:
    Esito e preferenze aggiornate, oppure un errore senza modifiche.
    """
    if (
        isinstance(commensali, bool)
        or not isinstance(commensali, int)
        or commensali < 0
    ):
        return {
            "stato": "errore",
            "messaggio": (
                "I commensali devono essere un intero positivo. "
                "Usa 0 soltanto per lasciare il dato invariato."
            ),
        }
    if (
        isinstance(tempo_minuti, bool)
        or not isinstance(tempo_minuti, int)
        or tempo_minuti < 0
    ):
        return {
            "stato": "errore",
            "messaggio": (
                "Il tempo deve essere un intero positivo. "
                "Usa 0 soltanto per lasciare il dato invariato."
            ),
        }
    if alimentazione not in ("", "vegetariana", "nessuna"):
        return {
            "stato": "errore",
            "messaggio": (
                "Le opzioni previste sono 'vegetariana' e 'nessuna'."
            ),
        }
    # Tutti i controlli precedono le modifiche.
    if commensali != 0:
        tool_context.state["commensali"] = commensali
    if alimentazione != "":
        tool_context.state["alimentazione"] = alimentazione
    if tempo_minuti != 0:
        tool_context.state["tempo_minuti"] = tempo_minuti
    return leggi_preferenze(tool_context)
```

Come funzionano le due funzioni

`leggi_preferenze` restituisce le preferenze della cena nella sessione corrente. Se `commensali` e `alimentazione` non sono ancora stati registrati, restituisce `null` per quei dati. Per il tempo restituisce invece il valore iniziale di **15 minuti**.

Il valore iniziale viene restituito dalla funzione quando manca un tempo registrato: non viene automaticamente scritto nello stato.

`aggiorna_preferenze` controlla i valori ricevuti e modifica soltanto i dati indicati. Se cambia il numero dei `commensali`, `alimentazione` e tempo rimangono invariati.

Il parametro `tool_context` viene passato automaticamente da ADK. Attraverso questo oggetto, la funzione può accedere allo stato della sessione.

Con questa istruzione leggiamo un valore:

```
tool_context.state.get("commensali")
```



Con questa lo aggiorniamo:

```
tool_context.state["commensali"] = commensali
```

ADK registra le modifiche effettuate attraverso `tool_context.state` negli eventi della sessione.

Che cosa significano i valori predefiniti

Nella funzione di aggiornamento:

- `commensali=0` significa "lascia invariato il numero di persone".
- `tempo_minuti=0` significa "lascia invariato il tempo".
- `alimentazione=""` significa "lascia invariata la preferenza alimentare".

Sono convenzioni interne alla funzione: zero non viene salvato come numero di commensali o tempo disponibile.

Il valore "nessuna", invece, è una preferenza esplicita: significa che l'utente non richiede una proposta vegetariana. Non significa "lascia invariato" e non indica l'assenza di allergie o intolleranze.

4. Inserisci il codice di Ubaldo

Apri `agent.py` e sostituisci il contenuto con questo codice:

```
from google.adk.agents import Agent
from google.genai import types
from .tools import leggi_dispensa, registra_consumo
from .preferenze import leggi_preferenze, aggiorna_preferenze

root_agent = Agent(
    name="ubaldo_chef",
    model="gemini-3.8-flash",
    generate_content_config=types.GenerateContentConfig(
        thinking_config=types.ThinkingConfig(
            thinking_level="low"
        )
    ),
    description=(
        "Ubaldo, uno chef burbero ma dal cuore d'oro, "
        "che propone ricette usando la dispensa e le preferenze "
        "della cena registrate nella sessione."
    ),
    instruction=""
)

Ti chiami Ubaldo. Sei un cuoco di vecchia scuola, pragmatico,
schietto e intollerante agli sprechi.

Non fai complimenti inutili, parli in modo diretto e un po' ruvido,
ma aiuti sempre chi è in difficoltà davanti al frigo.

PREFERENZE DELLA CENA:
- Le preferenze comprendono commensali, alimentazione e tempo.
- Usa leggi_preferenze per conoscere i valori attuali.
- Quando l'utente comunica una decisione certa, usa
  aggiorna_preferenze per registrarla.
- Passa soltanto i parametri da cambiare.
- Non limitarti a dire di aver registrato un dato: usa il tool
  e controlla il risultato.
- Non affermare che un aggiornamento sia riuscito se il tool
  restituisce un errore.
- Se l'utente chiede il riepilogo, leggi le preferenze con il tool.

NUMERO DI COMMENSALI:
- Se non è ancora indicato, chiedilo prima di proporre una ricetta.
- Il parametro commensali rappresenta sempre il totale.
- Se l'utente dice "si aggiungono due amici", leggi il totale
  attuale e aggiorna il valore aggiungendo due.
- Se il totale attuale è sconosciuto, chiedi un chiarimento.
- Se dice "forse vengono due amici", non aggiornare subito:
  chiedi per quante persone vuole organizzarsi.
- Non registrare zero persone: se l'utente annulla la cena,
  chiedi come vuole proseguire.
```



ALIMENTAZIONE:

- In questo esempio gestiamo "vegetariana" e "nessuna".
- "nessuna" significa nessuna preferenza alimentare dichiarata, non assenza di allergie o intolleranze.
- Se il valore non è ancora indicato, chiedi se desidera una proposta vegetariana o senza questa preferenza.
- Una ricetta vegetariana esclude carne, pesce e ingredienti derivati incompatibili, come il brodo di carne.
- Non confondere vegetariano e vegano.
- Non sostituire richieste diverse, allergie o intolleranze con una delle due opzioni: spiega il limite dell'esempio.

TEMPO:

- Se non viene indicato un tempo, il valore iniziale è 15 minuti.
- Se l'utente comunica un altro tempo, registralo.
- Il tempo comprende preparazione e cottura.
- Non registrare zero minuti come tempo disponibile.

RICETTE:

1. Prima di proporre una ricetta, leggi le preferenze e la dispensa.
2. Se mancano commensali o alimentazione, chiedi i dati mancanti.
3. Usa solo gli ingredienti disponibili, nelle quantità presenti.
Gli unici extra concessi sono acqua, sale, pepe e olio extravergine d'oliva.
4. Adatta le quantità al numero di commensali.
5. Rispetta alimentazione e tempo registrati.
6. Usa al massimo una padella o pentola e un tagliere.
7. Se quantità, attrezzatura o tempo non bastano, spiegalo.
Non inventare scorte e non dichiarare sufficiente una quantità chiaramente inadeguata.
8. Se l'utente propone cose non commestibili, riprendilo e non includerle nella ricetta.

GESTIONE DELLA DISPENSA:

- Proporre o accettare una ricetta NON significa averla cucinata.
- Cambiare preferenze NON modifica le scorte.
- Non registrare consumi durante la proposta di una ricetta.
- Quando l'utente dice di aver cucinato, riepiloga gli ingredienti e le quantità utilizzati e chiedi conferma.
- Solo dopo la conferma esplicita del riepilogo, chiama registra_consumo con tutti gli ingredienti consumati insieme.
- Usa i nomi esatti e le unità presenti nella dispensa.
- Non registrare nuovamente un consumo già registrato.
- Dichiarare che l'aggiornamento è riuscito solo se il tool restituisce stato "ok".
- Se il tool restituisce un errore, spiegalo senza fingere che l'operazione sia riuscita.

FORMATO DELLE RICETTE:

- 🍴 NOME DEL PIATTO: un nome onesto ma rustico.
- 👤 PORZIONI: numero di persone.
- ⌚ TEMPO E DIFFICOLTÀ: stima realistica in minuti.
- 🍴 INGREDIENTI: quantità totali per tutte le persone.
- / PIANO D'AZIONE CRONOLOGICO: passaggi numerati e concreti.
- 💡 IL CONSIGLIO DI UBALDO: una dritta pratica.

Per chiarimenti, conferme e riepiloghi, rispondi brevemente senza utilizzare tutta la scheda della ricetta.

```
""",
    tools=[
        leggi_dispensa,
        registra_consumo,
        leggi_preferenze,
        aggiorna_preferenze,
    ],
)
```

Ubaldo ora dispone di quattro strumenti: due per la dispensa e due per le preferenze.

Il limite fisso di 15 minuti diventa un **valore iniziale modificabile**. Restano il carattere di Ubaldo e il vincolo della singola padella.

Riduciamo il tempo dedicato al ragionamento

Abbiamo aggiunto questa configurazione:



```
generate_content_config=types.GenerateContentConfig(
    thinking_config=types.ThinkingConfig(
        thinking_level="low"
    )
),
```

Il parametro `thinking_level="low"` chiede al modello un livello di ragionamento ridotto, che può aiutare a contenere i tempi di risposta. Non elimina i passaggi necessari per usare gli strumenti e non aumenta il numero di chiamate disponibili.

Prima di proseguire, salva tutti i file in VSCode.

5. Avvia l'interfaccia web

Nel terminale, torna nella cartella principale:

```
cd ~/adk-svuota-frigo
```

Se hai aperto un nuovo terminale, riattiva l'ambiente:

```
source .venv/bin/activate
```

Se `adk web` è già in esecuzione, fermalo con **Ctrl+C** nel suo terminale.

Avvia ADK:

```
adk web
```

Apri nel browser l'indirizzo indicato, di solito:

```
http://localhost:8000
```

Seleziona `step3_sessione` e crea una nuova conversazione.

6. Prova la conversazione

Google ci permette di usare gratuitamente il modello scelto per questa guida, ma con un limite di 20 chiamate al giorno. Un messaggio a Ubaldo può richiedere più chiamate al modello, quindi terremo breve la conversazione di prova.

Se compaiono errori 429 o 503

Durante le prove potresti incontrare questi errori:

- **429 — limite di utilizzo raggiunto.** Può indicare troppe richieste in poco tempo oppure l'esaurimento della quota giornaliera. Nel primo caso, aspetta il tempo indicato e riprova. Se hai esaurito le 20 chiamate giornaliere, aspettare pochi secondi non basta: devi attendere il ripristino della quota.

- **503 — servizio temporaneamente non disponibile.** Il servizio può essere sovraccarico o momentaneamente indisponibile. Aspetta qualche secondo e invia nuovamente la richiesta. Se l'errore continua, lascia passare più tempo prima di riprovare.

Questi errori non significano necessariamente che ci sia un problema nel codice di Ubaldo.

Un errore può interrompere la risposta dopo che un tool ha già modificato i dati. Prima di ripetere un aggiornamento, controlla lo stato della sessione o il file della dispensa per verificare se l'operazione è già stata eseguita.

Prima richiesta: registra le preferenze

Invia:

```
Siamo in quattro, cena vegetariana, abbiamo 30 minuti. Registra le preferenze.
```

Ubaldo deve utilizzare `aggiorna_preferenze` per registrare:

- **Commensali:** 4.
- **Alimentazione:** vegetariana.
- **Tempo disponibile:** 30 minuti.

Può confermarlo con una normale risposta nella chat. **Non deve necessariamente mostrare un JSON:** controlla i valori nello stato della sessione o nel risultato del tool.



Seconda richiesta: cambia soltanto i commensali

Nella stessa conversazione, invia:

Siamo diventati sei.

Ubaldo deve aggiornare il numero di persone a **6**, mantenendo l'alimentazione **vegetariana** e il tempo di **30 minuti**.

Controlla i dati registrati: il riepilogo nella chat potrebbe non ripetere tutte le preferenze.

Terza richiesta: cambia il tempo e chiedi una ricetta

Sempre nella stessa conversazione, invia:

Abbiamo solo venti minuti. Proponimi una ricetta con quello che c'è in dispensa.

Ubaldo deve aggiornare il tempo a **20 minuti**, mantenendo **6 commensali** e l'alimentazione **vegetariana**.

Deve poi consultare la dispensa e proporre una ricetta coerente con queste informazioni, oppure spiegare perché le scorte o il tempo non sono sufficienti.

La dispensa deve rimanere invariata: chiedere una ricetta non significa averla cucinata.

7. Prova facoltativa: apri una nuova sessione

Quando hai chiamate disponibili, apri una **nuova conversazione** nello stesso agente e chiedi:

Quali preferenze hai registrato per questa cena?

In questa nuova sessione devono risultare:

- **Commensali:** non indicati.
- **Alimentazione:** non indicata.
- **Tempo disponibile:** 15 minuti, il valore iniziale restituito dal tool.

Le preferenze della conversazione precedente appartengono alla sua sessione. **Aprire una nuova sessione non cancella quella vecchia e non ripristina la dispensa**, che rimane salvata nel file.

Tornare alla conversazione precedente

Puoi tornare a una conversazione precedente selezionandola dall'elenco delle sessioni nell'interfaccia di ADK.

Ritroverai la cronologia e le preferenze registrate in quella conversazione. La dispensa rimane invece comune a tutte le sessioni di questo agente: Ubaldo leggerà le scorte attuali, non quelle presenti quando avevi lasciato la conversazione.

8. Che cosa abbiamo costruito

Ora abbiamo tre elementi distinti:

- **La cronologia della sessione** contiene messaggi, chiamate agli strumenti e risultati, comprese le decisioni poi cambiate.
- **Lo stato della sessione** contiene i valori aggiornati delle preferenze della cena.
- **Il file della dispensa** contiene le scorte, condivise dalle conversazioni di questa versione dell'agente.

Il codice controlla i valori ammessi e aggiorna soltanto i campi richiesti. Capire correttamente le parole dell'utente e scegliere quando chiamare gli strumenti resta un compito del modello: i test servono anche a verificare questo passaggio.

Non abbiamo ancora aggiunto una memoria personale valida per tutte le conversazioni. Abbiamo dato a Ubaldo un modo esplicito per gestire i dati della cena all'interno di una sessione.

Restano inoltre i limiti della seconda versione: la conferma prima del consumo è affidata alle istruzioni del modello e la dispensa locale è pensata per una persona alla volta.

Nella prima puntata abbiamo definito il comportamento di Ubaldo. Nella seconda gli abbiamo dato strumenti per leggere e modificare la dispensa. In questa terza versione abbiamo aggiunto la gestione delle preferenze, distinguendo ciò che è stato detto durante la conversazione dai valori che vogliamo utilizzare adesso.