

# Tutti i prompt [COPIA] del libro

**Claude Code per Tutti — 2a Edizione** · Bonus pack lettori

Raccolta dei prompt copia-e-incolla del libro, organizzati per capitolo.

**Come usarli:** 1. Apri Claude Code nel terminale (**claude**) 2. Copia il prompt qui sotto (selezione + Ctrl+C) 3. Incollalo nel terminale e premi Invio 4. Adatta i placeholder tipo `[il tuo nome]` o `[descrivi qui]`

Per i dettagli sull'uso di ogni prompt, vai al capitolo corrispondente del libro.

---

## Capitolo 3 — Il tuo primo dialogo con Claude Code

### Prompt 1

```
/compact focus on the performance issue we discussed
```

### Prompt 2

```
/btw ho già aggiunto il test per questo metodo?
```

### Prompt 3

```
/simplify focus on reducing code duplication
```

### Prompt 4

```
! npm run build 2>&1  
Questo build ha errori - puoi identificare il problema principale?
```

### Prompt 5

```
/plan refactor the database module to use connection pooling
```

### Prompt 6

```
# Crea il CLAUDE.md di progetto automaticamente  
/init
```

### Prompt 7

```
Aggiorna il CLAUDE.md con le convenzioni che abbiamo discusso:  
il formato di commit è "tipo(scope): descrizione" in italiano.
```

### Prompt 8

```
Ricorda sempre di usare pnpm invece di npm in questo progetto.
```

### Prompt 9

```
Rivedi il modulo di autenticazione e:  
- Controlla la gestione degli errori  
- Verifica che i token scadano correttamente  
- Segnala eventuali problemi di sicurezza
```

*Totale prompt Capitolo 3 — Il tuo primo dialogo con Claude Code: 9*

## Capitolo 4 — Skills: crea i tuoi comandi personalizzati

### Prompt 10

```
# Struttura consigliata per un progetto team  
.claude/  
├─ skills/  
│   ├─ deploy/          # /deploy - esegui il deploy  
│   │   └─ SKILL.md  
│   ├─ changelog/       # /changelog - genera il changelog  
│   │   └─ SKILL.md  
│   └─ review-pr/       # /review-pr - revisiona la PR
```

```
|      └─ SKILL.md
└─ CLAUDE.md
```

## Prompt 11

```
# Template description ben strutturata
---
description: [Cosa fa in una frase].
  Usa quando [trigger phrases, frasi naturali dell'utente].
  NON usare quando [distingui da casi simili].
---
```

## Prompt 12

```
# Skill che solo tu puoi invocare
---
description: Deploy dell'applicazione in produzione
disable-model-invocation: true
---
```

## Prompt 13

```
/create-api-endpoint users GET "Retrieve a paginated list of users"
```

## Prompt 14

```
# Struttura consigliata per skill complessa
mkdir -p ~/.claude/skills/security-audit/{examples,scripts}
touch ~/.claude/skills/security-audit/SKILL.md
touch ~/.claude/skills/security-audit/checklist.md
touch ~/.claude/skills/security-audit/examples/{good,bad}-example.md
touch ~/.claude/skills/security-audit/scripts/scan.sh
```

## Prompt 15

```
# Installa tutte e tre le skill di esempio
mkdir -p ~/.claude/skills/{commit-it,docfunc}
mkdir -p .claude/skills/sec-check

# Poi crea i SKILL.md con il contenuto degli esempi sopra
```

## Prompt 16

```
# Aggiungi una nota nel CLAUDE.md del progetto
## Skill disponibili
```

```
Questo progetto include skill Claude Code in ~/.claude/skills/:

- `/commit-it` – genera commit message in italiano (conventional commits)
- `/sec-check` – security check rapido prima di committare
- `/changelog` – genera changelog da commit
- `/review-interno` – code review strutturata del branch
```

Prima di usarle, accetta il workspace trust dialog che Claude Code mostra.

*Totale prompt Capitolo 4 — Skills: crea i tuoi comandi personalizzati: 7*

## Capitolo 5 — Hooks: automazione silenziosa al servizio del tuo workflow

### Prompt 17

```
chmod +x ~/.claude/hooks/session-context.sh
chmod +x ~/.claude/hooks/rich-context.sh
```

### Prompt 18

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
```

```

        "type": "command",
        "command": "~/claude/hooks/safety-check.sh",
        "timeout": 5
    }
]
}
]
}
}
}

```

### Prompt 19

```

{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "http",
            "url": "http://localhost:3000/webhook/file-changed",
            "timeout": 5
          }
        ]
      }
    ]
  }
}

```

### Prompt 20

```

#!/bin/bash
# Log semplice delle istruzioni caricate
echo "$(date +%H:%M:%S) loaded: $(jq -r '.file_path // "?"') >> ~/.claude/debug.log
exit 0

```

### Prompt 21

```

#!/bin/bash
# Hook minimo per contesto da git
BRANCH=$(git branch --show-current 2>/dev/null)
MODIFIED=$(git status --short 2>/dev/null)
jq -n --arg ctx "Branch: ${BRANCH}\nFile modificati:\n${MODIFIED}" '{
  hookSpecificOutput: { hookEventName: "SessionStart", additionalContext: $ctx }
}'

```

### Prompt 22

```

# ~/.claude/hooks/env-context.sh
ENV=${NODE_ENV:-development}
MSG="Ambiente: ${ENV}"
[ "$ENV" = "production" ] && MSG="PRODUZIONE: operare con cautela"
jq -n --arg m "$MSG" '{
  hookSpecificOutput: { hookEventName: "SessionStart", additionalContext: $m }
}'

```

*Totale prompt Capitolo 5 — Hooks: automazione silenziosa al servizio del tuo workflow: 6*

## Capitolo 6 — Claude Code senza tastiera — modalità non-interattiva e pipeline

### Prompt 23

```

# Estrarre solo il testo dalla risposta JSON
claude -p "query" --output-format json | jq -r '.result'

# Estrarre l'ID sessione per riprenderla dopo
SESSION_ID=$(claude -p "query" --output-format json | jq -r '.session_id')

```

### Prompt 24

```

claude -p "Spiega recursione" \
  --output-format stream-json \

```

```
--include-partial-messages | \
jq -rj 'select(.type == "stream_event" and .event.delta.type? == "text_delta") | .event.delta.text'
```

## Prompt 25

```
{
  "scripts": {
    "lint:typo": "git diff main | claude -p \"sei un linter di typo. per ogni typo nel diff, riporta nomefile:riga su una riga e il problema sulla riga successiva. non restituire nient'altro.\""
  }
}
```

## Prompt 26

```
# Crea commit automatico con allowedTools esatti
claude --bare -p "Guarda i cambiamenti staged e crea un commit appropriato" \
  --allowedTools "Bash(git diff *),Bash(git log *),Bash(git status *),Bash(git commit *)"
```

## Prompt 27

```
# Analisi bug con output JSON strutturato
cat error.log | claude -p "Analizza questo log e trova i bug" \
  --output-format json \
  --json-schema '{
    "type": "object",
    "properties": {
      "bugs": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "severity": {"type": "string", "enum": ["high","medium","low"]},
            "description": {"type": "string"},
            "line": {"type": "number"}
          }
        }
      }
    }
  }' | jq '.structured_output.bugs'
```

## Prompt 28

```
# .github/workflows/claude.yml – workflow base @claude trigger
name: Claude Code
on:
  issue_comment:
    types: [created]
  pull_request_review_comment:
    types: [created]
jobs:
  claude:
    runs-on: ubuntu-latest
    steps:
      - uses: anthropics/claude-code-action@v1
        with:
          anthropic_api_key: ${ secrets.ANTHROPIC_API_KEY }
```

## Prompt 29

```
# Review automatica su ogni PR aperta o aggiornata
name: Code Review
on:
  pull_request:
    types: [opened, synchronize]
jobs:
  review:
    runs-on: ubuntu-latest
    steps:
      - uses: anthropics/claude-code-action@v1
        with:
          anthropic_api_key: ${ secrets.ANTHROPIC_API_KEY }
          prompt: "Review questa PR per qualità e sicurezza. Posta i problemi trovati come commenti inline."
          claude_args: "--max-turns 5"
```

## Prompt 30

```
# Step minimale per usare Claude in GitLab CI
code-review:
  script:
    - git diff origin/main | claude --bare -p "Review del diff" --allowedTools ""
  variables:
    ANTHROPIC_API_KEY: $ANTHROPIC_API_KEY # Da GitLab CI variables
```

## Prompt 31

```
#!/bin/bash
# Script template per analisi con Claude in CI
LOG_FILE="${1:-error.log}"

tail -200 "$LOG_FILE" | \
  claude --bare -p "Analizza questi log: identifica causa principale e azioni consigliate. Markdown." \
  --output-format text \
  --allowedTools ""
```

## Prompt 32

```
# Make target per review automatica del diff corrente
review:
  git diff HEAD~1 | claude --bare -p \
    "Review del codice: elenca i problemi per severity (Alta/Media/Bassa)." \
    --output-format text
```

## Prompt 33

```
# Script diagnostico per print mode
echo "=== Test output format ==="
claude --bare -p "Rispondi solo OK" --output-format text
echo "---"
claude --bare -p "Rispondi solo OK" --output-format json | jq .
echo "=== Fine diagnostica ==="
```

*Totale prompt Capitolo 6 — Claude Code senza tastiera — modalità non-interattiva e pipeline: 11*

## Capitolo 7 — Subagents — Specialisti dedicati

### Prompt 34

```
# Avvia sessione con subagent specializzato
claude --agent code-reviewer

# Lista i subagents disponibili
claude agents

# Visualizza subagents nell'interfaccia interattiva
# (dentro Claude Code, digita):
# /agents
```

### Prompt 35

```
---
name: read-only-analyst
description: Analyst that can only read and analyze – never modify. Use for audits and reviews.
tools: Read, Grep, Glob, Bash
disallowedTools: Write, Edit
model: sonnet
---
```

Sei un analista specializzato in audit e code review. Non modifichi mai i file – solo leggi, analizzi, e riporti. Per ogni problema trovato, descrivi cosa hai visto e perché è rilevante.

### Prompt 36

```
---
name: codebase-expert
description: Expert in this codebase. Use for architectural questions and when context about the project is needed.
tools: Read, Grep, Glob, Bash
model: sonnet
memory: project
---
```

Sei l'esperto di questo codebase.

All'inizio di ogni sessione:

1. Leggi la tua memoria per aggiornare il contesto del progetto
2. Annota le domande ricevute e le risposte date

Dopo ogni conversazione, aggiorna MEMORY.md con:

- Nuovi pattern trovati
- Decisioni architetturali spiegate
- Aree critiche da monitorare

## Prompt 37

```
---
name: quality-gate
description: Verifica test suite dopo ogni modifica al codice. Use proactively after code changes.
tools: Bash, Read
model: sonnet
---
```

Esegui `npm test`. Se passano, rispondi "[OK] OK". Se falliscono, elenca i test falliti e le cause. Non modificare mai il codice.

## Prompt 38

# Subagent temporaneo per una sessione di refactoring

```
claude --agents '{
  "refactoring-guide": {
    "description": "Guida il refactoring TypeScript. Use when refactoring or cleaning up code.",
    "prompt": "Sei un esperto di TypeScript. Suggerisci refactoring chiari, sicuri, e leggibili. Non introdurre breaking changes.",
    "tools": ["Read", "Grep", "Glob"],
    "model": "sonnet"
  }
}'
```

## Prompt 39

```
---
name: test-engineer
description: Test engineer specialist. Use proactively after implementing features to generate tests, and when existing tests fail.
tools: Read, Write, Bash, Grep, Glob
model: sonnet
---
```

Sei un test engineer esperto in TDD e testing automatico.

Quando implementi test:

1. Studia il codice da testare
2. Identifica i casi edge e i percorsi critici
3. Scrivi test unitari e di integrazione
4. Assicurati che i test passino prima di restituire

Framework preferito per questo progetto: usa quello già esistente (controlla package.json).

*Totale prompt Capitolo 7 — Subagents — Specialisti dedicati: 6*

## Capitolo 8 — Le skill da padroneggiare — il catalogo operativo

### Prompt 40

# Review standard — controlla tutto  
/simplify

# Review mirata — solo efficienza memoria  
/simplify focus on memory efficiency

# Review pre-PR — qualità e leggibilità  
/simplify focus on readability and naming conventions

### Prompt 41

```
# Migrazione framework
/batch migrate src/ from Solid to React – preserve all existing tests and behavior

# Aggiornamento API
/batch update all uses of fetch() to use our new httpClient wrapper in src/api/

# Aggiunta logging
/batch add structured JSON logging to all service functions in src/services/

# Conversione TypeScript
/batch convert all .js files in src/ to .ts with strict mode – add types based on usage
```

### Prompt 42

```
# Analisi generica
/debug

# Problema specifico con descrizione
/debug il subagent code-reviewer si avvia ma non restituisce output

# Problema di performance
/debug le risposte sono molto lente negli ultimi 10 minuti
```

### Prompt 43

```
# Aspetta che un deploy finisca (controlla ogni 5 minuti)
/loop 5m check if the deployment to production succeeded – look at the deployment logs

# Monitor test suite durante sviluppo
/loop 2m run the test suite and report any new failures

# Senza intervallo – Claude decide il ritmo
/loop watch the build output and notify me when it completes or fails

# Usa .claude/loop.md come configurazione
/loop
```

### Prompt 44

```
# Carica reference API (Claude rileva il linguaggio automaticamente)
/claude-api

# Migra il codice al modello più recente
/claude-api migrate

# Crea un nuovo Managed Agent guidato
/claude-api managed-agents-onboard
```

### Prompt 45

```
# Esegui l'ottimizzazione dei permessi
/fewer-permission-prompts
```

### Prompt 46

```
/brainstorming
Ho un negozio online di prodotti artigianali.
Voglio aumentare le vendite del 20% entro tre mesi.
Aiutami a esplorare le strategie possibili.
```

### Prompt 47

```
/systematic-debugging
Il mio form di login funziona sul mio computer ma non su quello del cliente.
L'errore nel log è "Cannot read property 'userId' of undefined".
Ho già verificato le credenziali database – sono corrette.
```

### Prompt 48

```
/content-calendar
Canali: Instagram (3/settimana), LinkedIn (2/settimana)
Periodo: giugno 2026
```

Topic: lancio del mio corso di fotografia online (297€)  
Tono Instagram: ispirazionale e personale. Tono LinkedIn: professionale.

## Prompt 49

/conversion-copywriting  
Tipo: email di benvenuto (primo messaggio dopo iscrizione)  
Prodotto: corso online fotografia, 297€, 8 settimane  
Audience: hobbisti 35-55 anni che vogliono migliorare ma non diventare professionisti  
Obiettivo: portarli alla pagina di acquisto entro 48 ore  
Framework: PAS

## Prompt 50

/skill-creator  
Voglio creare una skill per generare release notes.  
Ogni volta che faccio un deploy, voglio una nota strutturata con:  
- Versione rilasciata  
- Lista delle modifiche (dal git log)  
- Eventuale nota per i clienti  
Il file deve essere salvato in docs/releases/ con la data nel nome.

## Prompt 51

```
---
name: safe-git
description: Git operations without permission prompts. Use when working with git.
disable-model-invocation: true
allowed-tools: Bash(git status) Bash(git diff *) Bash(git log *) Bash(git add *) Bash(git commit *)
---
```

Esegui operazioni git in modo sicuro:  
- git status e git diff: sempre permessi, non chiedere  
- git add e git commit: chiedi conferma prima  
- git push: MAI senza conferma esplicita dell'utente

*Totale prompt Capitolo 8 — Le skill da padroneggiare — il catalogo operativo: 12*

## Capitolo 9 — Gli 8 MCP da installare oggi

### Prompt 52

*claude mcp add --transport stdio filesystem --npx -y @modelcontextprotocol/server-filesystem /percorso/cartella*

Sostituisci ``/percorso/cartella`` con il percorso reale della cartella a cui vuoi dare accesso. Esempi:  
- Mac/Linux: ``/home/nomeutente/Documenti`` o ``~/Documenti``  
- Windows (WSL): ``/mnt/c/Users/nomeutente/Documents``  
- Windows (nativo): usa la versione Claude Desktop che gestisce i percorsi Windows

Se vuoi dare accesso a più cartelle, aggiungi più percorsi separati da spazio alla fine del comando:

### Prompt 53

*claude mcp add --transport stdio filesystem --npx -y @modelcontextprotocol/server-filesystem ~/Documenti ~/Desktop ~/Progetti*

**\*\*Autenticazione\*\***

Nessuna. Il controllo di accesso è il percorso stesso: specificare una cartella significa che Claude può leggere e scrivere solo in quella cartella e nelle sue sottocartelle.

**\*\*Esempio di utilizzo\*\***

Immagina di avere una cartella con cinquanta file di testo – appunti presi negli ultimi sei mesi, in modo disordinato, senza una struttura precisa. Con filesystem puoi dire:

### Prompt 54

*Leggi tutti i file .txt nella cartella ~/Documenti/appunti. Per ciascuno, estraimi: data (se presente), argomento principale, eventuali azioni da fare. Poi crea un file appunti-organizzati.md con tutto organizzato in ordine cronologico.*

Claude leggerà i file uno per uno, estrarrà le informazioni richieste, e creerà il file di riepilogo – tutto senza che tu debba aprire un singolo documento.

## Prompt 55

*Cerca in ~/Progetti tutti i file README.md. Per ciascuno, dimmi: il progetto è aggiornato? Ci sono sezioni mancanti rispetto a un README standard? Dammi un report con i progetti che hanno bisogno di attenzione.*

[PRATICA] Apri il terminale, installa filesystem con il comando sopra puntando alla tua cartella Documenti, poi chiedi a Claude di listarti i dieci file modificati più di recente. Verifica che risponda con i nomi reali dei tuoi file. Se i nomi corrispondono a file che conosci, il MCP funziona correttamente. Poi chiedi di aprire il più recente e fare un breve riassunto del suo contenuto – questo testa sia la lettura della lista che la lettura del contenuto di un singolo file.

**\*\*[DIETRO] Come funziona l'accesso ai file\*\***

Quando Claude usa filesystem, invia comandi al server MCP locale (il processo npx che hai avviato). Questo server ha accesso al filesystem reale del tuo computer, ma solo nella cartella che hai specificato. Claude non vede il file system completo – vede solo quello che il server MCP decide di esporre, e il server MCP espone solo la cartella che gli hai indicato.

Questo è il meccanismo di sicurezza fondamentale: la "sandbox" non è gestita da Claude ma dal server MCP che fa da intermediario. Se dici a Claude "leggi /etc/password" e hai dato accesso solo a ~/Documenti, il server MCP rifiuterà l'operazione.

**\*\*Trappole\*\***

- [!] Non dare accesso a '/' (radice del filesystem Mac/Linux) o a 'C:\' (radice Windows) – limita sempre il percorso a una cartella specifica. Dare accesso all'intero disco significa che Claude potrebbe leggere file di sistema, password, configurazioni sensibili.
- [!] filesystem non ha cestino – i file eliminati tramite Claude Code spariscono definitivamente (non vanno nel Cestino del sistema operativo). Prima di chiedere operazioni di cancellazione massiva, fai un backup.
- [!] Le operazioni di scrittura sono immediate. "Modifica tutti i .doc in questo formato" cambia i file senza passare da "salva come bozza". Chiedi sempre prima un'anteprima delle modifiche.

**\*\*Una sessione di lavoro reale con filesystem\*\***

Ecco come si presenta una sessione tipica dopo aver installato filesystem. Il contesto: hai una cartella ~/Clienti con sottocartelle per ciascun cliente, e ogni sottocartella contiene proposte, contratti, e comunicazioni salvate come file di testo.

Utente: "Guarda la cartella ~/Clienti e dimmi quali clienti non hanno avuto aggiornamenti negli ultimi 30 giorni."

Claude accede al filesystem, lista le cartelle, controlla le date di modifica dei file, e risponde con un elenco dei clienti "inattivi". Poi puoi continuare:

"Per ciascuno di questi clienti, leggi l'ultima comunicazione e proponi una email di follow-up personalizzata."

Claude legge i file, estrae il contesto dell'ultima interazione, e genera bozze di email personalizzate per ciascun cliente – il tutto senza che tu abbia dovuto aprire un singolo file. Questa è la differenza concreta che filesystem introduce nel lavoro quotidiano.

**\*\*Troubleshooting filesystem\*\***

Se ricevi errore "ENOENT: no such file or directory": il percorso che hai specificato nel comando di installazione non esiste o è scritto in modo sbagliato. Verifica il percorso con 'ls ~/Documenti' (Mac/Linux) o aprendo Esplora file (Windows).

Se ricevi errore "Permission denied": hai tentato di accedere a un file protetto dal sistema. filesystem rispetta i permessi del filesystem del sistema operativo – se un file non è leggibile dal tuo utente, Claude non può leggerlo.

Se Claude dice "non ho accesso a quella cartella": il percorso che stai chiedendo non è incluso nella cartella che hai specificato all'installazione. Puoi reinstallare il MCP aggiungendo il nuovo percorso:

## Prompt 56

*claude mcp remove filesystem claude mcp add -transport stdio filesystem - npx -y @modelcontextprotocol/server-fileSystem ~/Documenti ~/Desktop ~/Progetti ~/Clienti*

**\*\*Alternativa se filesystem non funziona\*\***

Claude Code ha tool file integrati (Read, Write, Edit) che operano sulla directory corrente del progetto. Non richiedono configurazione MCP ma hanno scope più limitato: funzionano bene per progetti con una struttura di cartelle definita, meno bene per accedere a file sparsi in luoghi diversi del computer.

---

**## 9.2 puppeteer**

**\*\*Cosa fa\*\***

puppeteer apre un browser Chrome reale – invisibile, controllato da Claude – e gli permette di navigare pagine web, fare clic su elementi, compilare form, catturare screenshot, ed eseguire JavaScript. È il MCP per tutto quello che richiede "guardare una pagina web" e interagire con essa come farebbe un essere umano con un mouse e una tastiera.

Le pagine web che richiedono interazione (clic su tab, scorrimento, login) sono inaccessibili ai normali strumenti di scraping perché il loro contenuto viene caricato dinamicamente dopo che l'utente interagisce. puppeteer supera questo limite perché usa un browser vero.

Usalo per: fare screenshot di siti per poi analizzarli, compilare form ripetitivi, testare se una pagina funziona correttamente, estrarre dati da siti che richiedono interazione, automatizzare flussi web che esegui manualmente ogni settimana.

[VER 2026-05-03 – <<https://github.com/modelcontextprotocol/servers>>] Il pacchetto npm è `@modelcontextprotocol/server-puppeteer`. Nota: il progetto modelcontextprotocol lo classifica come "reference implementation" – un esempio di come costruire server MCP, non un prodotto enterprise. In pratica funziona bene per usi personali.

**\*\*Installazione\*\***

## Prompt 57

```
claude mcp add --transport stdio puppeteer -- npx -y @modelcontextprotocol/server-puppeteer
```

Il primo avvio richiede qualche minuto perché Puppeteer scarica automaticamente una versione di Chromium (circa 200 MB). Questo accade una sola volta.

**\*\*Autenticazione\*\***

Nessuna. Se devi accedere a siti protetti da login, puoi usare puppeteer per fare il login interattivamente (fornendo le credenziali nel prompt) ma questa operazione resta sotto la tua responsabilità e va verificata con i termini di servizio del sito.

**\*\*Esempio di utilizzo\*\***

Un caso d'uso immediato: hai un portale web aziendale o un gestionale che aggiorni manualmente ogni settimana con gli stessi dati. Con puppeteer puoi automatizzare:

## Prompt 58

```
claude mcp add --transport stdio --env FIRECRAWL_API_KEY=la-tua-api-key firecrawl -- npx -y firecrawl-mcp
```

Sostituisci `la-tua-api-key` con la chiave che hai copiato dalla dashboard firecrawl.dev. La chiave ha il formato `fc-xxxxxxxxxxxxxxxxxxxxxxxxxxxx`. Se la dimentichi, puoi sempre generarne una nuova dalla dashboard – le chiavi precedenti vengono disattivate automaticamente.

**\*\*Autenticazione\*\***

[VER 2026-05-03 – <<https://firecrawl.dev>>] API key gratuita su firecrawl.dev. Il piano free offre 500 crediti al mese. Una pagina singola costa 1 credito, un crawling di un sito piccolo (10-20 pagine) costa 10-20 crediti. Per uso personale moderato, il piano gratuito è ampiamente sufficiente.

**\*\*Esempio di utilizzo\*\***

Ricerca e analisi competitiva:

## Prompt 59

*Cerca “migliori corsi fotografia online Italia 2026” e riassumimi i primi 5 risultati con: titolo del corso, prezzo, durata, punti di forza principali. Poi dimmi quale sceglieresti tu e perché.*

Monitoraggio di contenuti:

## Prompt 60

*Leggi questa pagina: <https://esempio.com/blog/articolo> Estraimi: titolo, autore, data, sommario di 200 parole, e tutti i link esterni citati. Salvami tutto in un file markdown chiamato articolo-[data-oggi].md*

Crawling di un intero sito per documentazione:

## Prompt 61

*Fai il crawling del sito <https://docs.esempio.com> (max 50 pagine). Costruisci un indice di tutte le pagine con: URL, titolo, argomento principale. Poi dimmi quali argomenti mancano rispetto a una documentazione completa per questo tipo di prodotto.*

[PRATICA] Scegli un sito web che leggi regolarmente (un blog, un quotidiano, una newsletter). Chiedi a Claude di estrarre i titoli degli ultimi 10 articoli con un breve sommario per ciascuno. Confronta il tempo impiegato con quello che ci vorresti tu manualmente.

**\*\*[DIETRO] Come firecrawl gestisce i siti difficili\*\***

I siti web moderni usano spesso JavaScript per caricare contenuti dopo il rendering iniziale: le notizie, le liste prodotti, i commenti arrivano tramite chiamate API successive all'apertura della pagina. Un parser HTML semplice vede solo lo scheletro vuoto. firecrawl risolve questo problema usando browser headless nel suo backend – ma tu non devi gestire niente di tutto questo, avviene automaticamente. In più, firecrawl ha sistemi per aggirare le protezioni anti-bot più comuni (non per violare termini di servizio, ma per leggere contenuti pubblicamente accessibili).

La differenza rispetto al WebFetch integrato in Claude Code è sostanziale: WebFetch legge HTML grezzo (veloce ma limitato), firecrawl renderizza la pagina completa e restituisce solo il testo pulito (più lento ma molto più preciso con siti dinamici).

**\*\*Trappole\*\***

- [!] Il piano free ha 500 crediti mensili. Un crawling di un sito grande può consumare molti crediti rapidamente. Prima di fare crawling, stima le pagine: usa `firecrawl\_map` per vedere l'elenco degli URL, poi decidi se procedere.
- [!] Alcuni contenuti dinamici che si caricano solo dopo interazione specifica (clic su un tab, compilazione di un form) non vengono estratti. Per questi casi usa puppeteer.
- [!] firecrawl può accedere solo a contenuti pubblicamente visibili (o accessibili con autenticazione che fornisci). Non è adatto per siti che richiedono login complessi con 2FA.

**\*\*Rassegna stampa automatizzata: firecrawl in azione\*\***

Uno dei workflow più utili per professionisti che devono tenersi aggiornati: la rassegna stampa giornaliera automatizzata.

Crea un file `~/Rassegna/fonti.txt` con i siti che segui (uno per riga). Poi ogni mattina puoi usare filesystem + firecrawl insieme:

## Prompt 62

*claude mcp add --transport stdio context7 -- npx -y @upstash/context7-mcp*

Non richiede configurazione aggiuntiva. Il piano gratuito ha rate limiting (poche centinaia di richieste al giorno), sufficiente per uso personale.

**\*\*Autenticazione\*\***

Nessuna. context7 funziona senza account o API key per l'uso base.

**\*\*Esempio di utilizzo\*\***

Documentazione su versioni specifiche:

## Prompt 63

*Usando context7, dimmi come si installa e configura React Query v5. Voglio capire le differenze principali rispetto alla v4: quali hook sono cambiati? Quali sono gli breaking change?*

Confronto tra strumenti:

## Prompt 64

*Con context7, confrontami Zustand e Jotai per la gestione dello stato in React. Basati sulla documentazione attuale — voglio capire quale è più adatto per un progetto piccolo con 3-4 sviluppatori.*

Debug con documentazione aggiornata:

## Prompt 65

*claude mcp add --transport stdio memory -- npx -y @modelcontextprotocol/server-memory*

I dati vengono salvati localmente in un file JSON nella cartella dei dati di sistema. Non vengono mai inviati a server esterni.

**\*\*Autenticazione\*\***

Nessuna. I dati sono salvati localmente sul tuo computer.

**\*\*Esempio di utilizzo\*\***

La prima sessione: costruisci il tuo profilo.

## Prompt 66

*Ho un nuovo cliente, Evento Gamma, che mi ha chiesto un preventivo per un evento aziendale. Basandoti su quello che sai di come lavoro con clienti simili, aiutami a strutturare il preventivo e la email di risposta.*

Aggiornamento della memoria:

## Prompt 67

*Aggiorna la memoria: ho chiuso il contratto con Evento Gamma. Budget: 2.500 euro. Data evento: 15 giugno. Tipo: cena aziendale, 80 persone. Aggiungi Evento Gamma alla lista dei miei clienti attivi.*

[PRATICA] Installa memory e crea il tuo profilo di base: nome, professione, 3-4 clienti o progetti principali, le tue preferenze di comunicazione con Claude. Chiudi completamente Claude Code (non solo la finestra – termina l'applicazione), riapri in una nuova sessione, e chiedi "Cosa sai di me?". Verifica che ricordi le informazioni che hai inserito. Questo è il test fondamentale di persistenza – se funziona, la memoria sopravvive alla chiusura della sessione.

**\*\*memory nel tempo: come evolve il grafo\*\***

La memoria non è statica – deve evolvere con te. Dopo il primo mese di utilizzo, la tua memoria dovrebbe contenere:

- Il tuo profilo di base (chi sei, cosa fai)
- I tuoi clienti o collaboratori principali con note aggiornate
- I tuoi progetti in corso con stato e scadenze
- Le tue preferenze e il tuo stile di lavoro con Claude

Dopo tre mesi, aggiungi:

- Pattern che hai osservato ("i meeting del lunedì mattina sono sempre urgenti")
- Risorse che usi spesso ("il mio fornitore preferito per X è Y")
- Informazioni su tools e processi consolidati

Una volta al mese è utile fare una pulizia: chiedi a Claude "Cosa c'è nella mia memoria che è probabilmente obsoleto?" Claude elencherà entità e osservazioni con date vecchie o potenzialmente superate, e potrai decidere cosa cancellare.

**\*\*[DIETRO] Knowledge graph vs. semplici note\*\***

memory non salva testo grezzo ma un grafo strutturato. La differenza è importante: se salvi "Bruno lavora con Azienda Alpha" come nodo, memory sa che "Bruno" è una persona, "Azienda Alpha" è un'organizzazione, e tra loro esiste una relazione di tipo "cliente". Quando chiedi "chi sono i miei clienti?", memory restituisce tutte le entità collegate a te dalla relazione "cliente" – non cerca parole chiave in un testo.

Questo rende le query molto più precise: "quali sono i progetti aperti?" restituisce solo le entità di tipo progetto con stato "aperto", non tutte le volte che hai menzionato la parola "progetto" in qualsiasi contesto.

La struttura a grafo permette anche relazioni indirette: se sai che "Evento Gamma è cliente di Bruno" e "Bruno usa Lightroom per gli eventi", puoi inferire che "Evento Gamma è associato a workflow Lightroom" anche senza averlo scritto esplicitamente.

**\*\*Trappole\*\***

- [!] La memoria è locale – non si sincronizza tra computer diversi. Se lavori su laptop e desktop, hai due knowledge graph separati. Puoi copiarli manualmente (sono file JSON) ma non c'è sincronizzazione automatica.
- [!] Informazioni obsolete rimangono in memoria finché non le cancelli esplicitamente. Se un cliente termina il contratto, devi dirlo esplicitamente a Claude perché aggiorni il grafo. Una pulizia periodica (ogni tre mesi) mantiene la memoria accurata.
- [Tip] Inizia semplice: aggiungi solo le informazioni che usi spesso. Un grafo sovraffollato diventa rumoroso. Aggiungi più dettagli man mano che capisci cosa ti è utile ricordare.

**\*\*Costruire il tuo knowledge graph: cosa ricordare\*\***

Non tutto merita di essere nella memoria. Un knowledge graph efficace contiene informazioni che:

- Cambiano raramente (il tuo nome, il tuo settore, i tuoi strumenti principali)
- Ci vorresti tempo a rispiegare ogni volta (la struttura dei tuoi progetti, i tuoi processi standard)
- Servono spesso come contesto (chi sono i tuoi clienti, quali sono le tue tariffe)

Non mettere in memoria:

- Dettagli che cambiano spesso (stato di avanzamento di un task specifico)
- Informazioni che già hai in un documento organizzato
- Tutto – un grafo troppo denso diventa rumoroso

Un buon punto di partenza per chiunque:

## Prompt 68

```
claude mcp add --transport http github https://api.githubcopilot.com/mcp/
--header "Authorization: Bearer IL_TUO_GITHUB_PAT"
```

Sostituisci `IL\_TUO\_GITHUB\_PAT` con il token che hai copiato.

**\*\*Autenticazione\*\***

[VER 2026-05-03 – <<https://github.com/settings/tokens>>] Personal Access Token (PAT) generato dalla sezione Settings del tuo account GitHub. Usa i "Fine-grained tokens" – permettono di limitare l'accesso ai soli repository che ti servono e ai soli permessi necessari. Questo è più sicuro dei "Classic tokens" che danno accesso a tutto.

**\*\*Esempio di utilizzo\*\***

Analisi e miglioramento di documentazione:

## Prompt 69

*Mostrami tutte le issue aperte nel repository "mio-progetto" con label "bug". Per ciascuna: dimmi se la descrizione è sufficiente per lavorarci, o se manca informazione. Proponi un commento per chiedere i dettagli mancanti.*

Revisione di un file di configurazione:

## Prompt 70

*Leggi il file .github/workflows/deploy.yml nel mio repository "sito-produzione". Verifica che il workflow segua le best practice e dimmi se c'è qualcosa da migliorare per efficienza e sicurezza.*

[PRATICA] Installa il MCP github, poi chiedi a Claude di mostrarti la lista dei tuoi repository con il nome e la data dell'ultimo commit. Verifica che veda i repository del tuo account (pubblici e privati con il PAT configurato correttamente).

**\*\*github come strumento di documentazione aziendale\*\***

Molte piccole imprese e liberi professionisti che collaborano con team tecnici scoprono che GitHub è diventato il loro sistema di documentazione "de facto" – non per scelta, ma perché lo sviluppatore lo usa e lì finisce tutto. Con il MCP github puoi interagire con quella documentazione senza imparare l'interfaccia GitHub.

## Prompt 71

*Leggi tutti i file nella cartella "docs/" del mio repository "mia-azienda". Costruisci un indice dei contenuti: per ciascun file, dammi il titolo e un sommario di 2 righe. Poi dimmi: c'è documentazione duplicata? C'è qualcosa di chiaramente obsoleto (menziona versioni vecchie o date lontane)?*

Questo ti dà una panoramica della documentazione esistente senza doverti loggare su GitHub, navigare la struttura delle cartelle, aprire ogni file e leggerlo uno per uno.

**\*\*[DIETRO] HTTP transport vs. stdio per GitHub\*\***

GitHub offre il suo MCP server come servizio HTTP remoto – non devi installare niente sul tuo computer oltre all'autenticazione. Quando Claude invoca un tool github, la richiesta viaggia da Claude Code al server MCP di GitHub (via HTTP), che esegue le operazioni sull'API GitHub e restituisce il risultato. Il token PAT nell'header è la tua firma digitale che dice a GitHub "questo è Bruno, ha i permessi X".

Il vantaggio del transport HTTP rispetto alla vecchia versione stdio è che il server è mantenuto da GitHub stesso, aggiornato quando l'API GitHub cambia, e non richiede Node.js locale o dipendenze npm.

**\*\*Trappole\*\***

- [!] Non mettere il PAT direttamente in chiaro in un file di configurazione se quel file potrebbe finire in un repository pubblico. Usa variabile d'ambiente: prima esegui `export GITHUB\_PAT=il-tuo-token` nel terminale, poi aggiungi il MCP con `--header "Authorization: Bearer \${GITHUB\_PAT}"`.
- [!] I PAT hanno una scadenza. Quando scadono, il MCP smette di funzionare senza preavviso (vedrai errori di autenticazione). Quando crei il token, imposta una scadenza di 6-12 mesi e annotala subito in calendario.
- [!] I "Fine-grained tokens" limitano l'accesso ai repository che selezioni. Se crei un nuovo repository dopo aver configurato il token, dovrai aggiornare le permission del token per includerlo.

**\*\*github per chi non scrive codice: documentazione e gestione issue\*\***

Il MCP github è spesso percepito come uno strumento per sviluppatori. In realtà, se la tua organizzazione o il tuo team usa GitHub per qualsiasi scopo – versionare documenti, tracciare task, gestire progetti – github ti apre le stesse possibilità.

Esempio per chi gestisce un sito web con un team tecnico: il tuo sviluppatore usa GitHub per il codice del sito. Tu non sai leggere codice, ma hai bisogno di capire cosa è cambiato di recente e segnalare problemi.

## Prompt 72

```
claude mcp add --transport stdio sequential-thinking -- npx -y @modelcontextprotocol/server-sequential-thinking
```

Autenticazione: nessuna.

**\*\*Quando usarlo\*\***

Non tutti i problemi richiedono sequential-thinking. È particolarmente utile per:

- Decisioni con molte variabili e vincoli (scegliere tra alternative, pianificare)
- Analisi di situazioni complesse con molti fattori interconnessi
- Problemi dove il "perché" è importante quanto il "cosa" – vuoi capire il ragionamento
- Casi in cui hai dubbi che Claude possa semplificare troppo
- Prima di usare /plan per task di sviluppo complessi
- Quando vuoi condividere il ragionamento con altri – il formato strutturato è più comunicabile di una risposta fluida

Non serve per: domande dirette con risposta univoca, recupero di informazioni semplici, conversazioni veloci, compiti creativi dove il pensiero strutturato può limitare l'originalità.

**\*\*Esempio di utilizzo\*\***

Decisione strategica:

## Prompt 73

```
claude mcp add --transport http gmail https://gmailmcp.googleapis.com/mcp/v1 claude mcp add --transport http google-calendar https://calendarmcp.googleapis.com/mcp/v1
```

Passo 2 – Avvia Claude Code e autenticati:

## Prompt 74

```
/mcp
```

Vedrai i server gmail e google-calendar nell'elenco con stato "pending auth". Selezionali e segui il flusso OAuth nel browser: accedi al tuo account Google con le credenziali normali, concedi i permessi richiesti (leggi attentamente cosa stai autorizzando – puoi concedere solo lettura se preferisci iniziare in modo conservativo), e il browser ti reindirizzerà a Claude Code.

[VER 2026-05-03 – <https://code.claude.com/docs/en/mcp>] I token OAuth sono memorizzati in modo sicuro nel tuo sistema. Vengono rinnovati automaticamente senza che tu debba rifare il login. Se il browser non si apre automaticamente, copia l'URL fornito da Claude Code e aprilo manualmente.

**\*\*Autenticazione\*\***

[VER 2026-05-03 – <https://developers.google.com/workspace/guides/configure-mcp-servers>] OAuth 2.0 con account Google. Scope richiesti: Gmail (leggi e scrivi email, gestisci etichette), Calendar (leggi e scrivi eventi, gestisci calendari). Puoi controllare e revocare l'accesso in qualsiasi momento da myaccount.google.com/security → "App con accesso al tuo account".

**\*\*Esempi di utilizzo\*\***

Triage giornaliero delle email:

*Totale prompt Capitolo 9 — Gli 8 MCP da installare oggi: 23*

## Capitolo 11 — Creare skill personalizzate

### Prompt 75

```
/skill-creator
```

Voglio creare una skill che ogni volta che la invoco:

- Analizza i file nella cartella corrente
- Mi dice quanti file per tipo ci sono
- Suggerisce come organizzarli in sottocartelle
- Mi chiede conferma prima di fare qualsiasi spostamento

## Prompt 76

```
---
name: organizza-file
description: Analizza i file nella cartella corrente, raggruppa per tipo, e propone
  una struttura di organizzazione con spostamento confermato. Usare quando l'utente
  chiede di organizzare file, riordinare una cartella, o capire cosa c'è in una
  directory e come strutturarla.
when_to_use: >
  Trigger: "organizza questa cartella", "riordina i file", "cosa c'è in Downloads",
  "crea una struttura", "dividi i file per tipo"
disable-model-invocation: false
allowed-tools: Read Bash(ls *) Bash(find *) Bash(mv *) Bash(mkdir *)
---
```

## Cosa fare

1. Elenca tutti i file nella cartella corrente con `ls -la`
2. Raggruppa mentalmente per estensione:
  - Immagini: .jpg, .jpeg, .png, .gif, .webp, .svg, .heic
  - Documenti: .pdf, .docx, .doc, .xlsx, .pptx, .txt, .md
  - Video: .mp4, .mov, .avi, .mkv, .webm
  - Audio: .mp3, .wav, .aac, .flac, .ogg
  - Archivi: .zip, .tar, .gz, .rar, .7z
  - Altro: tutto il resto
3. Mostra la tabella riepilogativa: Tipo | Count | Dimensione totale
4. Proponi la struttura di cartelle (Immagini/, Documenti/, Video/, ecc.)
5. Chiedi conferma: "Vuoi che sposti i file in queste cartelle? Digita 's' per confermare o 'n' per annullare."
6. Se confermato: esegui gli spostamenti uno per volta mostrando ogni operazione
7. Al termine: mostra il riepilogo degli spostamenti eseguiti

## Regole

- NON spostare file nascosti (che iniziano con .)
- NON spostare file aperti o in uso (estensioni .lock, .tmp)
- Mostrare sempre il piano completo PRIMA di eseguire
- In caso di conflitto di nomi (file con stesso nome già esistente), aggiungere suffisso numerico: documento.pdf → documento-2.pdf
- Se la cartella è vuota, comunicarlo e terminare

## Prompt 77

```
/skill-creator
Testa la skill organizza-file con questi scenari:
1. Una cartella Downloads con 50 file misti (il caso normale)
2. Una cartella vuota (edge case)
3. Una cartella con solo file nascosti (edge case)
4. Una cartella dove tutti i file hanno già la struttura corretta (idempotenza)
```

## Prompt 78

```
/remotion-powered-by-truant
Reel 9:16, durata 15 secondi
Testo: "La creatività è intelligenza che si diverte." – Albert Einstein
Stile: dark mode, font bold, animazione testo reveal
```

## Prompt 79

```
---
name: analisi-legale
description: >
  Analizza documenti legali e identifica clausole potenzialmente problematiche.
  Usa quando l'utente chiede di rivedere un contratto, analizzare termini e
  condizioni, o identificare rischi in documenti legali.
when_to_use: >
  Trigger: "rivedi questo contratto", "analizza questi termini", "ci sono
  problemi in questo documento", "cosa significa questa clausola", "è pericoloso
  questo accordo?"
argument-hint: [percorso-file-o-testo]
disable-model-invocation: false
allowed-tools: Read Bash(cat *) Bash(wc *)
model: opus
effort: high
---
```

## Analisi documento legale

Esegui questi passaggi nell'ordine:

1. Leggi l'intero documento prima di commentare
2. Identifica il tipo di documento (contratto, NDA, EULA, altro)
3. Elenca le clausole potenzialmente problematiche con:
  - Citazione testuale della clausola
  - Perché potrebbe essere problematica
  - Rischio (basso/medio/alto)
4. Fornisci una valutazione generale (1-5 stelle per semplicità)
5. Raccomanda i punti da negoziare prima di firmare

**## Avvertenza**

Sei un assistente per la comprensione dei documenti, non un avvocato.  
Ricorda sempre all'utente che per decisioni legali importanti deve consultare un professionista.

## Prompt 80

```
---
name: status-progetto
description: Mostra lo stato completo del progetto Git corrente. Usa quando
  l'utente chiede lo stato, le modifiche in sospeso, o cosa è cambiato.
---
```

**## Stato corrente del progetto**

**\*\*Branch e ultimo commit:\*\***  
!`git log --oneline -5`

**\*\*Modifiche in sospeso:\*\***  
!`git status --short`

**\*\*Dimensione delle modifiche:\*\***  
!`git diff --stat HEAD`

**## Istruzioni**

Analizza questi dati e fornisci:

1. Un riassunto in 3 bullet di cosa è cambiato
2. Eventuali rischi (file critici, test mancanti, conflitti potenziali)
3. Il messaggio di commit suggerito per le modifiche in sospeso

## Prompt 81

/organizza-file  
(esegui nella cartella ~/Downloads reale)

## Prompt 82

Analizza questo SKILL.md e aiutami a migliorarlo. Cerca:

1. Istruzioni ridondanti o contraddittorie
2. Sezioni che potrebbero andare in file separati (reference.md, examples/)
3. La description: potrebbe essere più precisa per il triggering automatico?
4. Opportunità per usare dynamic context injection (!`comando`)
5. Regole che potrebbero essere semplificate

[incolla il contenuto del SKILL.md]

Poi scrivi la versione migliorata mantenendo tutto il comportamento utile,  
ma eliminando ridondanze e contraddizioni.

## Prompt 83

# Crea la directory della skill nel progetto  
mkdir -p .claude/skills/mia-skill

# Copia il SKILL.md  
cp ~/temp/mia-skill/SKILL.md .claude/skills/mia-skill/

# Aggiungi al repository  
git add .claude/skills/mia-skill/  
git commit -m "feat: aggiungi skill mia-skill per automazione X"

## Prompt 84

Leggi il file `~/.claude/skills/[nome-skill-sorgente]/SKILL.md` e dimmi:

1. Quali sezioni sono riutilizzabili per [descrizione del tuo use case]?
2. Quali sezioni sono specifiche per l'uso originale e dovrei rimuovere?
3. Cosa dovrei aggiungere per adattarla a [tuo use case specifico]?
4. Ci sono problemi strutturali di cui devo essere consapevole?

## Prompt 85

```
---
name: mio-fork-specializzato
description: >
  Fork specializzato di [skill-sorgente] per [tuo use case specifico].
  Usa quando [trigger specifici per il tuo contesto, diversi dall'originale].
# Basato su [skill-sorgente] di [autore originale]
# Versione originale: [link o riferimento]
# Modifiche principali:
# - Aggiunto: [cosa hai aggiunto]
# - Rimosso: [cosa hai tolto]
# - Modificato: [cosa hai cambiato]
---
```

*Totale prompt Capitolo 11 — Creare skill personalizzate: 11*

## Capitolo 12 — Editing video con Claude

### Prompt 86

```
sudo apt update && sudo apt install ffmpeg
```

### Prompt 87

```
winget install Gyan.FFmpeg
```

### Prompt 88

```
# Clona il toolkit
git clone https://github.com/digitalsamba/claude-code-video-toolkit.git

# Crea la directory della skill e copia il contenuto
mkdir -p ~/.claude/skills/video-use
cp -r claude-code-video-toolkit/skills/video-use/. ~/.claude/skills/video-use/
```

### Prompt 89

Installa la skill `video-use` dal Claude Code Video Toolkit su GitHub.  
Controlla che FFmpeg sia installato, poi installa la skill.  
Dimmi quando è pronta.

### Prompt 90

```
/video-use [descrizione dell'operazione che vuoi eseguire]
```

### Prompt 91

Prendi il file `conferenza.mp4` nella mia cartella `Desktop/video`.  
Taglia i primi 3 minuti (intro) e gli ultimi 2 minuti (chiusura).  
Salva il risultato come `conferenza-editata.mp4` nella stessa cartella.

### Prompt 92

Taglia il file `lezione.mp4` dal minuto 16:30 al minuto 28:45.  
Salva il risultato come `lezione-concetto-chiave.mp4`  
Usa stream copy (senza ricompressione) per velocità massima.

### Prompt 93

Converti il file `tutorial.mp4` in formato verticale 9:16 per Instagram Reel.  
Risoluzione target: 1080x1920.  
Il video è orizzontale: centra il soggetto e riempi il resto con uno sfondo sfocato (blur) preso dal video stesso.  
Non usare barre nere.  
Salva come `tutorial-reel.mp4`

## Prompt 94

Unisci questi file video nell'ordine indicato:

1. intervista-partel.mp4
2. intervista-parte2.mp4
3. intervista-parte3.mp4

Aggiungi una dissolvenza al nero di 0.5 secondi tra ogni clip.

Assicurati che l'audio sia continuo e senza salti.

Salva come intervista-completa.mp4

## Prompt 95

Estrai l'audio dal file intervista.mp4.

Formato: MP3, qualità 128 kbps, mono (un canale), frequenza 16kHz.

Salva come audio-per-trascrizione.mp3

Dimmi la dimensione del file risultante.

## Prompt 96

Comprimi il file presentazione.mp4 per ridurlo a meno di 15 MB.

Mantieni la risoluzione originale se possibile.

Se la risoluzione deve scendere, massimo 720p.

Salva come presentazione-compresa.mp4

Dimmi la qualità stimata e la dimensione prima di procedere.

## Prompt 97

Aggiungi il file logo.png come watermark al video tutorial.mp4:

- Posizione: angolo in basso a destra
- Margini: 20px dal bordo destro, 20px dal bordo inferiore
- Opacità: 70% (semitrasparente per non coprire il contenuto)
- Dimensione logo: 120px di larghezza, mantieni proporzioni

Salva come tutorial-brandizzato.mp4

## Prompt 98

Crea una GIF animata dal file demo.mp4:

- Dal minuto 0:45 al minuto 0:58 (13 secondi)
- Risoluzione 480px di larghezza (standard per GIF)
- 15 frame al secondo (bilanciamento qualità/dimensione)
- Ottimizza la palette colori

Salva come demo-clip.gif e dimmi la dimensione risultante.

## Prompt 99

Aiutami a configurare l'API Whisper di OpenAI passo per passo.

Non ho mai usato API prima d'ora.

Parti dall'inizio: dove devo andare, cosa devo cliccare, dove trovare la API key.

Aspetta la mia conferma a ogni passaggio prima di continuare.

## Prompt 100

Imposta la API key di OpenAI come variabile d'ambiente sul mio sistema.

La key è: [incolla qui la key, es. sk-proj-...]

Vuoi che funzioni anche nelle sessioni future, non solo in questa.

## Prompt 101

Verifica che la API key di OpenAI sia configurata correttamente.

Fai una chiamata di test all'API Whisper con un audio di prova (genera tu il file se non ne ho uno) e dimmi se funziona.

## Prompt 102

Aggiungi sottotitoli automatici in italiano al file mio-video.mp4

usando Whisper API con il modello gpt-4o-transcribe.

Font Arial Bold, testo bianco con contorno nero 3px, posizione 80% dal basso.

Prima mostrami il file SRT generato, poi chiedi conferma per procedere con il burn-in.

Salva il risultato come mio-video-con-sottotitoli.mp4

## Prompt 103

Aiutami a installare Whisper open-source sul mio computer per la trascrizione locale.  
Il mio sistema è [Linux/Mac/Windows con WSL].  
Voglio usarlo per trascrivere video in italiano senza usare l'API a pagamento.

## Prompt 104

Trascrivi il file audio-italiano.mp3 con Whisper locale.  
Usa il modello large-v3 (il migliore per l'italiano).  
Output: file SRT con timestamp.

## Prompt 105

Elabora il file registrazione-raw.mp4 e producilo come Reel Instagram:

1. EDITING: Taglia i primi 40 secondi (intro) e gli ultimi 20 secondi (outro).  
Mantieni tutto il resto.
2. AUDIO: Normalizza il volume a -16 LUFS (standard per social media).
3. FORMATO: Converti in verticale 9:16 (1080x1920).  
Il video è orizzontale 16:9: centra il soggetto e usa sfondo blur (prendi dal video originale).
4. SOTTOTITOLI: Genera i sottotitoli in italiano con Whisper API (gpt-4o-transcribe).  
Mostrami il file SRT e aspetta la mia approvazione prima di procedere.
5. STILE SOTTOTITOLI: Font Arial Bold, 60px, testo bianco, contorno nero 3px.  
Posizione: 15% dal basso (zona sicura da sovrimpressioni Instagram).  
Allineamento: centrato.
6. COMPRESSIONE: Target sotto 100 MB (limite upload Instagram).  
Mantieni la massima qualità possibile entro quel limite.

Salva il risultato finale come reel-pronto.mp4

Prima di iniziare: mostrami il piano completo step-by-step e chiedi conferma.

## Prompt 106

Crea una versione TikTok del file reel-pronto.mp4:

- Formato 9:16, 1080x1920 (stessa risoluzione)
- Durata: se più lungo di 60 secondi, taglia al minuto
- Sottotitoli: stessa posizione ma font 20% più grande (su TikTok si vede più piccolo)
- Aggiungi fade-in nero di 0.5 secondi all'inizio
- Encoding: H.264 High Profile, audio AAC 192kbps

Salva come tiktok-versione.mp4

## Prompt 107

Dove posizionare i sottotitoli per Instagram Reel 1080x1920  
senza che vengano coperti dall'interfaccia Instagram?  
Calcolami la posizione Y ottimale in pixel.

## Prompt 108

Crea una skill "pipeline-social" che esegue automaticamente  
la pipeline completa (taglio → audio → formato → sottotitoli → compressione)  
su qualsiasi video che le passo come argomento.  
Parametri configurabili: formato target (reel/post/tiktok),  
se generare sottotitoli (sì/no), dimensione target in MB.  
Salva la skill in ~/.claude/skills/pipeline-social/

## Prompt 109

/pipeline-social  
Elabora tutti i file .mp4 nella cartella ~/Video/settimana-corrente/  
Formato: reel 9:16  
Sottotitoli: sì, italiano  
Dimensione target: 80 MB  
Salva i risultati in ~/Video/pronti-per-upload/

### Prompt 110

Controlla l'utilizzo dell'API Whisper di questo mese:

- Totale minuti trascritti
- Costo totale
- Proiezione costo fine mese

Vai su [platform.openai.com/usage](https://platform.openai.com/usage) e guida attraverso le sezioni rilevanti.

### Prompt 111

Migliora la qualità audio del file registrazione.mp4 prima della trascrizione:

1. Riduci il rumore di fondo (filtro denoise)
2. Normalizza il volume a -16 LUF5
3. Converti in mono (un canale solo – Whisper lavora meglio)
4. Imposta frequenza di campionamento a 16kHz (ottimale per Whisper)
5. Salva come audio-ottimizzato.wav

Poi usa questo file per la trascrizione invece dell'audio originale.

### Prompt 112

Mostrami il file SRT generato dalla trascrizione di mio-video.mp4

prima di procedere con il burn-in.

Voglio revisionarlo e correggere eventuali errori.

### Prompt 113

Usa il file SRT corretto che ti mostro per il burn-in:

[incolla il contenuto SRT corretto]

Esegui il burn-in con stile standard (Arial Bold 60px, bianco, contorno nero).

### Prompt 114

Trascrivi il file podcast-tech.mp4 in italiano con Whisper API.

Il podcast parla di Claude Code, skill, MCP, subagent, e automazioni AI.

Questi termini appaiono frequentemente e devono essere trascritti correttamente.

Usa il modello gpt-4o-transcribe.

### Prompt 115

/video-use aggiungi sottotitoli da intervista.srt al file intervista.mp4 con questo stile:

Font Arial Black 64px, testo bianco puro (#FFFFFF),

contorno nero 4px, ombra leggera, posizione 82% dal basso, centrato.

Salva come intervista-con-sub-styled.mp4

### Prompt 116

Ho installato FFmpeg ma ottengo "command not found". Aiutami a risolvere.

Il mio sistema è [Linux/Mac/Windows].

### Prompt 117

Il mio file audio è più grande di 25 MB e Whisper API lo rifiuta.

Dividilo in segmenti da 5 minuti, invia ogni segmento a Whisper separatamente,

e poi unisci i file SRT risultanti in ordine corretto.

Il file si chiama audio-lungo.mp3

### Prompt 118

I sottotitoli nel video edited-final.mp4 sono desincronizzati:

arrivano circa 2 secondi in ritardo rispetto all'audio.

Correggi il file SRT spostando tutti i timestamp di -2 secondi.

### Prompt 119

La trascrizione Whisper del file riunione.mp4 ha troppi errori –

audio di qualità bassa, microfono distante, rumore di fondo.

Applica prima il filtro denoise aggressivo di FFmpeg e riprova.

Usa l'opzione afftdn per la riduzione rumore.

## Prompt 120

Il file video-senza-audio.mp4 non ha traccia audio.  
Ho una registrazione audio separata in audio-registrazione.wav.  
Sincronizzali: l'audio inizia 3 secondi dopo l'inizio del video.  
Salva come video-sincronizzato.mp4

## Prompt 121

I sottotitoli in video.mp4 spesso escono dai bordi laterali.  
Riapplica i sottotitoli con queste limitazioni:

- Massimo 35 caratteri per riga
- A capo automatico se la riga è più lunga
- Margini laterali di 50px da ogni bordo

*Totale prompt Capitolo 12 — Editing video con Claude: 36*

## Capitolo 13 — Automazione temporale — /loop, /schedule, worktrees

### Prompt 122

/loop 5m controlla se il deploy su Vercel è andato a buon fine e dimmi l'esito

### Prompt 123

/loop 30m esegui npm run test e dimmi se ci sono test falliti

### Prompt 124

/loop controlla se la CI è passata e se ci sono nuovi commenti di review sulla PR

### Prompt 125

Controlla la PR del branch `release/next`.

- Se la CI è rossa: scarica il log del job fallito, diagnostica, e fai un push con un fix minimo
- Se ci sono nuovi commenti di review: rispondili e chiudi i thread
- Se tutto è verde e silenzioso: dì solo "tutto OK" in una riga

### Prompt 126

Tra 45 minuti, controlla se i test di integrazione sono passati e dimmi l'esito.

### Prompt 127

Alle 16:30, ricordami di fare il push del branch feature/auth prima che scada la finestra di deploy.

### Prompt 128

Crea un task che ogni lunedì mattina alle 9:00 mi dia un riassunto dei commit sul branch main della settimana precedente.

### Prompt 129

Che task schedulati ho attivi in questa sessione?

### Prompt 130

Cancella il task che controlla i test ogni 30 minuti.

### Prompt 131

/schedule daily PR review at 9am

### Prompt 132

/schedule ogni lunedì mattina alle 8, genera un report dei commit della settimana precedente

### Prompt 133

/schedule in 2 weeks, apri una PR che rimuove il feature flag di autenticazione

### Prompt 134

```
/schedule run nome-routine
```

### Prompt 135

```
/schedule domani alle 9:00, scrivi un riassunto dei commit merged nella settimana corrente e aprimi una PR con il file CHANGELOG.md aggiornato
```

### Prompt 136

```
claude --worktree feature-auth
```

### Prompt 137

```
claude --worktree bugfix-123
```

### Prompt 138

```
claude --worktree  
# Crea: worktree "bright-running-fox" (nome generato automaticamente)
```

### Prompt 139

```
git remote set-head origin -a
```

### Prompt 140

```
git remote set-head origin develop
```

### Prompt 141

```
echo ".claude/worktrees/" >> .gitignore
```

### Prompt 142

```
claude --worktree react19-upgrade
```

### Prompt 143

Aggiorna React dalla versione 18 alla 19 in questo progetto.

- Aggiorna package.json con le nuove versioni
- Esegui npm install
- Risolvi tutti gli errori di tipo e i breaking change
- Esegui l'intera suite di test e correggi i test falliti
- Fammi un report finale con: le modifiche necessarie, i test che erano già falliti prima, e qualsiasi deprecation che dovrei sapere

### Prompt 144

Lavora su questi tre task in parallelo, usando un worktree isolato per ognuno:

1. Implementa il modulo di autenticazione
2. Scrivi i test di integrazione per le API esistenti
3. Aggiorna la documentazione con le nuove route

### Prompt 145

```
{  
  "permissions": {  
    "allow": [  
      "Bash(npm run test)",  
      "Bash(npm run lint)",  
      "Bash(git add .)",  
      "Bash(git commit -m *)",  
      "Bash(git push origin claude/*)",  
      "Read(**/*.ts)",  
      "Edit(**/*.ts)"  
    ],  
    "deny": [  
      "Bash(rm -rf *)",  
      "Bash(git push --force *)",  
      "Bash(git push origin main)",  
      "Bash(git push origin master)",  
    ]  
  }  
}
```

```

    "Bash(curl *)",
    "Read(./env)",
    "Read(./env.*)"
  ],
  "defaultMode": "acceptEdits"
}
}

```

### Prompt 146

```

{
  "model": "claude-sonnet-4-6",
  "effortLevel": "high"
}

```

### Prompt 147

```

{
  "model": "claude-sonnet-4-6",
  "effortLevel": "high",
  "permissions": {
    "allow": [
      "Bash(npm run *)",
      "Bash(npx *)",
      "Bash(git add *)",
      "Bash(git commit -m *)",
      "Bash(git push origin claude/*)",
      "Bash(git checkout -b claude/*)",
      "Read(**/*.ts)",
      "Read(**/*.tsx)",
      "Read(**/*.json)",
      "Edit(**/*.ts)",
      "Edit(**/*.tsx)"
    ],
    "deny": [
      "Bash(rm -rf *)",
      "Bash(git push --force *)",
      "Bash(git push origin main)",
      "Bash(git push origin master)",
      "Bash(curl * | bash)",
      "Bash(wget * -O- | bash)",
      "Read(./env*)",
      "Read(./secrets/**)"
    ],
    "ask": [
      "Bash(git push origin *)"
    ],
    "defaultMode": "acceptEdits"
  },
  "cleanupPeriodDays": 14
}

```

### Prompt 148

```

# Aggiungere al .gitignore del progetto
echo ".claude/settings.local.json" >> .gitignore

```

### Prompt 149

/schedule ogni venerdì alle 17:00, leggi tutti i commit degli ultimi 7 giorni sul branch main, raggruppa le modifiche per tipo (feature, fix, refactor), e aggiorna il file CHANGELOG.md con una nuova sezione per la settimana corrente. Committa e fai una PR con il titolo "chore: update CHANGELOG week [N]".

### Prompt 150

```

/loop 2h
Controlla se il numero di issue aperte su GitHub è sceso sotto 10.
Se sì: avvisami con un messaggio "Target raggiunto: X issue aperte" e fermati.
Se no: scrivi solo quante issue ci sono e riprova tra 2 ore.

```

### Prompt 151

/loop 3m controlla l'output di `vercel --logs` per eventuali errori o il messaggio "Deployment complete". Dimmi lo stato a ogni check. Quando vedi "Deployment complete" o un errore critico, fermati e avvisami.

### Prompt 152

/schedule ogni giorno alle 8:30, analizza il file src/index.ts e controlla se tutte le funzioni esportate sono documentate nel README.md. Per quelle mancanti, aggiungi una sezione di documentazione minima con la firma della funzione e un esempio. Apri una PR se ci sono modifiche.

### Prompt 153

/schedule ogni giorno alle 7:30, leggi tutte le issue GitHub aperte senza etichetta. Per ognuna: aggiungi l'etichetta appropriata tra "bug", "enhancement", "documentation", "question". Se non riesci a categorizzarla con certezza, aggiungi "needs-triage". Scrivi un commento breve spiegando la categorizzazione. Non chiudere nessuna issue.

### Prompt 154

/schedule ogni giorno alle 22:00, leggi tutti i file .md modificati oggi nella cartella ~/note/. Crea un file ~/note/riassunti/[data]-riassunto.md con: i concetti chiave emersi oggi, le decisioni prese, e i task in sospeso che dovrò riprendere domani. Non modificare i file originali.

### Prompt 155

claude --worktree ts-migration

### Prompt 156

/loop 20m  
Continua la migrazione dei file da JavaScript a TypeScript.  
- Lavora un file alla volta nella cartella src/  
- Dopo ogni file: esegui tsc --noEmit per verificare che compili  
- Se ci sono errori di tipo: risolvi prima di passare al file successivo  
- Fermati dopo 5 ore di lavoro totale e fammi un report di avanzamento

### Prompt 157

Elenca tutti i task schedulati attivi in questa sessione con i loro ID e la prossima esecuzione pianificata.

### Prompt 158

Perché il task schedulato "review-pr" ha fallito l'ultima volta?  
Mostrami l'errore e suggeriscimi come configurare i permessi in settings.json per permettere questa operazione.

### Prompt 159

Cancella tutti i task schedulati attivi in questa sessione.

### Prompt 160

```
# Controlla se il repository ha un remote configurato
git remote -v

# Controlla lo stato dei worktrees esistenti
git worktree list

# Rimuovi worktrees orfani se necessario
git worktree prune
```

### Prompt 161

Il loop che ho avviato sta modificando file che non dovrebbe toccare. Fermalo, mostrami cosa ha fatto nelle ultime 3 iterazioni, e suggerisci come riscrivere il prompt per limitare le sue azioni.

### Prompt 162

Dimmi quali task schedulati sono attivi in questa sessione, qual è la frequenza di ognuno, e fai una stima del numero di chiamate API che genereranno nelle prossime 24 ore.

## Prompt 163

Esegui questa operazione una volta sola e dimmi cosa faresti se fosse schedulata:  
[incolla il prompt che vorresti usare nel loop]  
Non fare niente – mostrami solo il piano.

*Totale prompt Capitolo 13 — Automazione temporale — /loop, /schedule, worktrees: 42*

## Capitolo 14 — Sei progetti end-to-end

### Prompt 164

Verifica il mio ambiente per i progetti del Capitolo 14:

1. Versione Claude Code: `claude --version` (deve essere  $\geq$  v2.1.72)
2. MCP filesystem: è configurato? Prova a leggere un file di test
3. Variabili d'ambiente configurate: `OPENAI_API_KEY` esiste?
4. Cartelle necessarie: `~/Note`, `~/Report`, `~/Dashboard`, `~/Video` esistono? Creale se mancano.

Dimmi per ognuno: OK o cosa devo configurare.

### Prompt 165

Crea questa struttura di cartelle nel mio home directory (`~`) se non esiste già:

- `~/Note/Lavoro/`
- `~/Note/Personale/`
- `~/Note/Idee/`
- `~/Note/Riferimenti/`
- `~/Note/Riassunti/`
- `~/Report/`
- `~/Dashboard/`
- `~/Video/bozze/`
- `~/Video/finiti/`
- `~/Video/`

Conferma ogni cartella creata o già esistente.

### Prompt 166

Configura il MCP filesystem per accedere alla mia cartella Downloads.

Su Linux/Mac: `~/Downloads`

Su Windows: `C:\Users\[nome_utente]\Downloads` (adatta al mio sistema)

Mostrami la configurazione da aggiungere al file MCP config e applicala.

### Prompt 167

Conta quanti file ci sono nella mia cartella Downloads e dimmi il totale.

### Prompt 168

Analizza la mia cartella Downloads e dimmi:

- Quanti file ci sono in totale
- Quali tipi di file (per estensione) e quanti per tipo
- Quanto spazio totale occupano
- Quali file sembrano duplicati (stesso nome o stessa dimensione)
- I file più vecchi di 6 mesi (candidati alla pulizia)

Non spostare o modificare nulla per ora.

### Prompt 169

Crea un file di test in `~/Downloads/test-mcp-check.txt` con il contenuto "MCP filesystem funziona correttamente - [data odierna]" poi leggilo e dimmi cosa contiene.

### Prompt 170

Crea questa struttura di cartelle dentro `~/Downloads/`:

- Immagini/ (per: `.jpg`, `.jpeg`, `.png`, `.gif`, `.webp`, `.svg`, `.heic`)
- Documenti/ (per: `.pdf`, `.doc`, `.docx`, `.xls`, `.xlsx`, `.ppt`, `.pptx`, `.txt`, `.md`)
- Video/ (per: `.mp4`, `.mov`, `.avi`, `.mkv`, `.webm`)
- Audio/ (per: `.mp3`, `.wav`, `.m4a`, `.flac`, `.aac`)
- Programmi/ (per: `.exe`, `.dmg`, `.pkg`, `.deb`, `.rpm`, `.msi`, `.appimage`)
- Archivi/ (per: `.zip`, `.rar`, `.7z`, `.tar`, `.gz`, `.tar.gz`)
- Altro/ (tutto il resto)

Prima mostrami il piano completo (quanti file andrebbero in ogni cartella) e chiedi la mia conferma prima di spostare qualsiasi file.

## Prompt 171

Aggiusta il piano: tutti i file .pdf che hanno "fattura" o "contratto" nel nome vanno in Documenti/Lavoro/ invece che in Documenti/.  
Mostrami il piano aggiornato e poi procedi.

## Prompt 172

Crea una skill "organizza-downloads" in ~/.claude/skills/organizza-downloads/SKILL.md  
La skill deve:

1. Analizzare il contenuto di ~/Downloads
2. Spostare i file nelle cartelle corrette (le stesse regole applicate oggi)
3. Loggare ogni operazione nel file ~/Downloads/.log-organizzazione.txt
4. Alla fine, mostrare un riassunto: file spostati, file saltati, spazio liberato

Non deve chiedere conferma per ogni file – funziona in automatico.

## Prompt 173

Testa la skill organizza-downloads su questa cartella di test:  
~/Downloads/test-organizer/  
(crea tu stesso 5-10 file di test di tipi diversi)  
Mostrami l'output del log dopo l'esecuzione.

## Prompt 174

/schedule ogni domenica alle 9:00, esegui la skill /organizza-downloads  
sulla mia cartella ~/Downloads e mandami un riassunto delle operazioni eseguite.

## Prompt 175

La skill organizza-downloads ha sbagliato categoria per questi file:

- documento-cliente-acme.pdf → l'ha messo in Documenti/ ma doveva andare in Lavoro/Clienti/Acme/
- database.bin → l'ha messo in Programmi/ ma doveva andare in Altro/

Aggiorna le regole della skill per gestire questi casi:

1. PDF con "cliente" o il nome di un'azienda nel nome → Lavoro/Clienti/[nomecliente]/
2. File .bin che non sono installer → Altro/Dati/

Poi mostrami il SKILL.md aggiornato.

## Prompt 176

Qual è il percorso esatto della mia cartella Downloads su questo sistema?  
Prova prima ~/Downloads, poi ~/download, poi ls ~ per elencarne le cartelle.

## Prompt 177

Aggiorna la skill organizza-downloads con queste esclusioni:

- Non spostare file con estensioni .conf, .ini, .cfg, .env
- Non spostare file che iniziano con un punto (file nascosti)
- Non spostare file più vecchi di 1 anno (probabilmente ci sono per un motivo)

Questi file rimangono in Downloads/ così come sono.

## Prompt 178

Dopo l'organizzazione, identifica i file candidati all'eliminazione:

- Duplicati esatti (stesso contenuto, nomi diversi)
- File temporanei (.tmp, .temp, .crdownload, .part)
- Screenshot numerati in sequenza (spesso ce ne sono centinaia)

Mostrami la lista con le dimensioni prima di eliminare qualsiasi cosa.

## Prompt 179

Accedi a questi tre strumenti e dimmi cosa trovi:

1. MCP gmail: elenca le ultime 5 email ricevute questa settimana
2. MCP google-calendar: elenca gli eventi di questa settimana
3. MCP filesystem: elenca i file modificati questa settimana in ~/Note/ (o la cartella che uso per le note)

Non serve il dettaglio completo – solo la conferma che hai accesso.

## Prompt 180

Crea un file template in ~/Report/template-settimanale.md con questa struttura:

```
# Report settimanale – [DATA]
```

```
## Sommario esecutivo
[3 bullet point con le cose più importanti della settimana]

## Attività completate
[Lista numerata delle cose fatte, con data]

## In corso
[Cosa è iniziato ma non concluso]

## Prossima settimana
[3-5 priorità per la settimana entrante]

## Note e osservazioni
[Riflessioni libere, lezioni imparate]

---
Generato automaticamente il [DATA E ORA]
```

## Prompt 181

Genera il report settimanale usando solo i file in ~/Note/ modificati negli ultimi 7 giorni.  
Non hai accesso a Gmail o Google Calendar – usa solo il filesystem locale.

## Prompt 182

Genera il report settimanale per questa settimana usando il template in ~/Report/template-settimanale.md.  
Fonti dati:

- Email Gmail degli ultimi 7 giorni (filtra per quelle lavorative – esclude newsletter e promozioni)
- Calendari Google di questa settimana (solo eventi passati)
- File .md in ~/Note/ modificati negli ultimi 7 giorni

Salva il report in ~/Report/[anno]-[settimana]-report.md  
Mostrami il report generato.

## Prompt 183

Crea una skill "report-settimanale" in ~/.claude/skills/report-settimanale/SKILL.md  
La skill deve:

1. Raccogliere dati da Gmail, Google Calendar, e ~/Note/ degli ultimi 7 giorni
2. Compilare il template in ~/Report/template-settimanale.md
3. Salvare il report in ~/Report/[YYYY]-W[WW]-report.md (formato anno-settimana)
4. Mostrare un'anteprima del report al termine

La skill non deve richiedere input dall'utente – funziona in autonomia.

## Prompt 184

/schedule ogni venerdì alle 17:00, esegui la skill /report-settimanale  
e dimmi quando è pronto.

## Prompt 185

Il report settimanale generato è buono ma voglio aggiungere:

1. Una sezione "Decisioni prese" – solo le scelte importanti fatte questa settimana
2. Una riga "Tempo speso in riunioni" – conta il numero di ore in eventi del calendario
3. Rimuovi la sezione "Note e osservazioni" – non la uso

Aggiorna il template in ~/Report/template-settimanale.md e rigenera il report di test.

## Prompt 186

Crea una seconda versione della skill "report-settimanale-ext"  
che usa lo stesso template ma in lingua professionale terza persona:  
"Questa settimana il team ha completato..." invece di "Ho completato..."  
Salva in ~/Report/[YYYY]-W[WW]-report-ext.md

## Prompt 187

Il report settimanale include troppe email automatiche e newsletter.  
Aggiorna il filtro Gmail per considerare solo:

- Email con risposta (ho risposto o ho scritto io)
- Email da mittenti nei miei contatti
- Email con oggetto che non contiene "unsubscribe", "newsletter", "notification", "noreply"

## Prompt 188

Elenca tutti i calendari Google a cui ho accesso tramite MCP.  
Mostrami nome e ID di ognuno.

## Prompt 189

Apri Telegram e segui questi passaggi per creare il tuo bot:

1. Cerca @BotFather nella barra di ricerca di Telegram
2. Avvia una conversazione e invia il comando: /newbot
3. Quando chiede il nome: scegli un nome descrittivo (es. "Il mio assistente personale")
4. Quando chiede l'username: scegli un username unico che finisce in "bot" (es. "tuonome\_assistente\_bot")
5. Copia il token API che BotFather ti fornisce – formato: 1234567890:AABBccDDee...

Dimmi quando hai il token.

## Prompt 190

Crea un subagent "telegram-bot" in ~/.claude/agents/telegram-bot.md

Il subagent deve:

- Usare il token Telegram dalla variabile TELEGRAM\_BOT\_TOKEN
- Fare polling per i nuovi messaggi ogni 10 secondi usando l'API Telegram getUpdates
- Per ogni messaggio ricevuto: passare il testo a Claude Code come prompt e inviare la risposta via sendMessage
- Avere accesso al MCP filesystem (per rispondere a richieste sui file)
- Rispondere sempre in italiano
- Limitare le operazioni filesystem alla cartella ~/Documenti e sottocartelle
- Logga ogni messaggio ricevuto e la risposta inviata in ~/.telegram-bot.log

## Prompt 191

Avvia il bot Telegram in modalità test: fai polling per 30 secondi  
e dimmi se ricevi messaggi. Manda tu stesso un messaggio al bot  
(username: @tuonome\_assistente\_bot) e mostrami se arriva.

## Prompt 192

Aggiungi al bot questi comandi speciali:

- /report → esegue la skill report-settimanale e invia il risultato via Telegram
- /organizza → esegue la skill organizza-downloads e manda il riassunto
- /cerca [testo] → cerca nelle note in ~/Note/ e restituisce i risultati
- /aiuto → mostra la lista dei comandi disponibili

Per qualsiasi altro messaggio: trattalo come un prompt libero per Claude.

## Prompt 193

Aggiungi un hook Stop al subagent telegram-bot:  
quando il bot ha completato un task lungo (più di 30 secondi),  
invia automaticamente una notifica Telegram con il messaggio:  
"Task completato: [nome del task]"

## Prompt 194

/loop 10s /telegram-bot

Avvia il bot in polling continuo. Il loop riattiverà il bot  
ogni 10 secondi per raccogliere i nuovi messaggi.

Dimmi quando è in esecuzione.

## Prompt 195

Messaggi da inviare al bot su Telegram per testarlo:

Test 1: "Ciao! Funzioni?"

→ deve rispondere con una presentazione breve

Test 2: "Quanti file ho in ~/Documenti?"

→ deve accedere al MCP filesystem e rispondere con un numero

Test 3: "/cerca lean startup"

→ deve cercare nelle note e rispondere con i risultati

Test 4: "/aiuto"

→ deve listare tutti i comandi disponibili

## Prompt 196

Aggiungi una verifica di autorizzazione al bot Telegram:

- Leggi il mio user ID Telegram dalla variabile TELEGRAM\_ALLOWED\_USER\_ID
  - Prima di rispondere a qualsiasi messaggio, verifica che l'ID del mittente corrisponda
  - Se il mittente non è autorizzato: rispondi solo con "Non autorizzato" e logga il tentativo
- Come trovo il mio user ID Telegram? Aiutami a configurare questa variabile.

## Prompt 197

Aggiungi la variabile d'ambiente TELEGRAM\_ALLOWED\_USER\_ID al mio sistema.

Il valore è: [inserisci il tuo user ID]

Mostrami come aggiungerla in modo permanente su [Linux/Mac/Windows].

## Prompt 198

Aggiorna il subagent telegram-bot con queste restrizioni:

- Può LEGGERE file solo da: ~/Documenti, ~/Note, ~/Report, ~/Downloads
- Può SCRIVERE/MODIFICARE file solo da: ~/Note, ~/Downloads/organizzato
- Non può eseguire comandi Bash (solo operazioni file e letture)
- Non può accedere a file fuori dalle cartelle permesse
- Se richiede un'operazione non permessa: risponde "Operazione non consentita"

## Prompt 199

Crea questa struttura di cartelle in ~/Note/:

- Lavoro/ (per note professionali, riunioni, decisioni)
- Personale/ (per pensieri, riflessioni, obiettivi personali)
- Idee/ (per idee, brainstorming, progetti futuri)
- Riferimenti/ (per link, citazioni, informazioni da conservare)
- Riassunti/ (generati automaticamente)

Crea anche il file ~/Note/indice.md con la struttura e le istruzioni d'uso.

## Prompt 200

Usando il MCP memory, crea queste entità iniziali per il sistema di note:

- Utente: [il tuo nome], usa questo sistema per [scopo principale]
- Struttura cartelle: ~/Note/ con Lavoro/, Personale/, Idee/, Riferimenti/, Riassunti/
- Regole: ogni nota ha titolo, data, tag, e contenuto
- Categorie principali: [elenca i tuoi argomenti frequenti]

## Prompt 201

Crea la skill "nota" in ~/.claude/skills/nota/SKILL.md

La skill riceve come argomento il contenuto della nota (tutto quello che scrivo dopo /nota).

Deve:

1. Determinare la categoria giusta tra Lavoro, Personale, Idee, Riferimenti (basandosi sul contenuto)
2. Suggestire 2-3 tag rilevanti
3. Salvare la nota in ~/Note/[categoria]/[YYYY-MM-DD]-[titolo-breve].md con questo formato:  
---  
data: [data]  
tag: [tag1, tag2]  
---  
[contenuto]
4. Aggiornare l'indice in ~/Note/indice.md
5. Salvare anche una voce nel MCP memory per ricerche future

## Prompt 202

Crea la skill "cerca-note" in ~/.claude/skills/cerca-note/SKILL.md

La skill riceve come argomento il tema di ricerca (es. "/cerca-note react hooks").

Deve:

1. Cercare nei metadati MCP memory note rilevanti
  2. Cercare anche nel testo dei file in ~/Note/ per occorrenze del tema
  3. Presentare i risultati con: titolo, data, cartella, e un estratto di 2-3 righe
  4. Ordinare per rilevanza (prima le note più recenti con più occorrenze)
- Massimo 10 risultati.

## Prompt 203

/nota

Riunione con il team oggi (lunedì). Decisione: usiamo React 19 per il nuovo progetto.

Motivazione: migliore supporto per Suspense e streaming SSR.  
Azioni: fare spike tecnico entro venerdì, documento di decisione da condividere.

## Prompt 204

Crea la skill "riassunto-note" in ~/.claude/skills/riassunto-note/SKILL.md  
La skill accetta come argomento opzionale un periodo ("settimana", "mese", o una data).  
Deve:  
1. Leggere tutte le note create nel periodo indicato (default: ultima settimana)  
2. Identificare i temi ricorrenti e i concetti chiave  
3. Generare un riassunto strutturato con: temi principali, decisioni prese, idee da sviluppare  
4. Salvare il riassunto in ~/Note/Riassunti/[YYYY]-W[WW]-riassunto.md

## Prompt 205

Connetti il sistema note al bot Telegram (se hai completato 14.3):  
Aggiungi il comando /nota al bot Telegram così che possa accettare note via telefono.  
Quando scrivo al bot "/nota [contenuto]", deve:  
1. Invocare la skill nota con il contenuto  
2. Rispondere con "Nota salvata in [categoria]: [titolo]"

## Prompt 206

Connetti il sistema note al report settimanale (se hai completato 14.2):  
Modifica la skill report-settimanale per includere nella sezione "Note e osservazioni"  
le ultime 5 note della settimana dalla cartella ~/Note/Idee/  
In questo modo il report settimanale cattura automaticamente le idee che hai avuto durante la settimana.

## Prompt 207

Aggiorna la skill /nota con queste regole di categorizzazione prioritarie:  
- Se il testo contiene "riunione", "meeting", "cliente", "progetto" → Lavoro/  
- Se il testo contiene "idea", "concept", "sarebbe interessante", "e se" → Idee/  
- Se il testo contiene "link", "http", "fonte:", "citazione:" → Riferimenti/  
- In tutti gli altri casi → usa il giudizio contestuale  
Mostrami il SKILL.md aggiornato.

## Prompt 208

Cerca il file contenente "[parola che ricordo dalla nota]" in ~/Note/ usando grep ricorsivo.  
Se lo trovi, dimmi in quale file è e perché la skill /cerca-note non l'ha trovato.

## Prompt 209

Cerca in TUTTI i file .md di ~/Note/ (incluse sottocartelle) il testo "[termine da trovare]".  
Usa grep -r per la ricerca ricorsiva.  
Ordina i risultati dal più recente al meno recente.

## Prompt 210

Crea un indice delle note in ~/Note/.indice-cache.json:  
Per ogni nota: nome file, data, categoria, tag, prime 100 parole.  
L'indice si aggiorna ogni volta che si aggiunge una nuova nota via /nota.  
La skill /cerca-note deve cercare prima nell'indice (veloce) e poi nel testo completo  
solo per le note che l'indice considera candidate.

## Prompt 211

Verifica che questi strumenti siano installati e funzionanti:  
1. Node.js: node --version (richiesto 18 o superiore)  
2. FFmpeg: ffmpeg -version (qualsiasi versione recente)  
3. Skill remotion: esiste il file ~/.claude/skills/remotion-powered-by-truant/SKILL.md?  
4. Skill video-use: esiste il file ~/.claude/skills/video-use/SKILL.md?  
Dimmi l'esito di ogni verifica e cosa fare se qualcosa manca.

## Prompt 212

/remotion-powered-by-truant  
Tipo: Reel verticale 9:16  
Durata: 15 secondi  
Testo principale: "La semplicità è la sofisticazione suprema."  
Attribuito a: Leonardo da Vinci  
Stile: dark mode, sfondo quasi-nero (#0a0a0a), testo bianco puro

Animazione: reveal verticale, lettera per lettera, timing 0.8s  
Tema: ispirazione e saggezza  
Output: ~/Video/bozze/video-01-base.mp4

## Prompt 213

/video-use aggiungi sottotitoli italiani al file ~/Video/bozze/video-01-base.mp4  
Testo del sottotitolo: "La semplicità è la sofisticazione suprema. – Leonardo da Vinci"  
Stile: Arial Black 48px, testo bianco, contorno nero 3px  
Posizione: 82% dall'alto (zona sicura Instagram, sopra i controlli UI)  
Allineamento: centrato  
Salva come ~/Video/finiti/video-01-con-sub.mp4

## Prompt 214

Crea la skill "video-sociale" in ~/.claude/skills/video-sociale/SKILL.md  
La skill riceve questi parametri:

- TESTO: la citazione o frase da mettere nel video (obbligatorio)
- AUTORE: l'attribuzione (opzionale, default "")
- FORMATO: reel o post (default: reel)
- STILE: dark o light (default: dark)

La skill deve:

1. Invocare remotion-powered-by-truant con i parametri appropriati
2. Attendere il rendering completo
3. Invocare video-use per aggiungere sottotitoli
4. Salvare il video finale in ~/Video/finiti/ con nome [YYYY-MM-DD]-[primeparole].mp4
5. Mostrare il percorso del file finale e la durata dell'elaborazione

## Prompt 215

/video-sociale  
TESTO: "Il futuro appartiene a chi crede nella bellezza dei propri sogni."  
AUTORE: Eleanor Roosevelt  
FORMATO: reel  
STILE: dark

## Prompt 216

/video-sociale  
TESTO: "Fai oggi ciò che gli altri non vogliono fare."  
AUTORE: ""  
FORMATO: reel  
STILE: light

## Prompt 217

/video-sociale  
TESTO: "Il successo è la somma di piccoli sforzi ripetuti ogni giorno."  
AUTORE: Robert Collier  
FORMATO: post  
STILE: dark

## Prompt 218

Crea il file ~/Video/programma-settimanale.md con questo formato:

```
---  
Lunedì: "Testo per lunedì" – Autore  
Martedì: "Testo per martedì" – Autore  
[...]  
---
```

/schedule ogni domenica alle 20:00, leggi il file ~/Video/programma-settimanale.md  
e genera un video per ogni giorno della settimana entrante con la skill /video-sociale.  
Salva i video in ~/Video/finiti/ con il nome del giorno come prefisso.

## Prompt 219

Aggiorna la skill video-sociale per supportare un parametro PIATTAFORMA:

- instagram\_reel: 9:16, 1080x1920, max 60s, sottotitoli posizione 82% dall'alto
- tiktok: 9:16, 1080x1920, max 60s, sottotitoli 20% più grandi rispetto a Instagram
- youtube\_shorts: 9:16, 1080x1920, max 60s, sottotitoli centrati al 80%
- linkedin\_video: 16:9, 1920x1080, max 30s, sottotitoli in basso al 90%

Default: instagram\_reel

## Prompt 220

Leggi il file ~/Video/programma-settimanale.md e genera tutti i video per la settimana.  
Per ogni citazione: genera sia la versione Instagram che la versione LinkedIn.  
Nomina i file con il prefisso del giorno e la piattaforma:  
"lun-ig-video.mp4", "lun-li-video.mp4", ecc.  
Dimmi il totale dei file generati e il tempo totale di elaborazione.

## Prompt 221

Verifica l'installazione di Node.js e trovami il percorso completo dell'eseguibile.  
Poi aggiungi il percorso al PATH del mio sistema e verifica che funzioni.

## Prompt 222

Il video generato da remotion non ha audio.  
Verifica le dipendenze audio necessarie per remotion su [Linux/Mac/Windows]  
e installale. Poi ritenta la generazione del video.

## Prompt 223

Il file ~/Video/finiti/video-01-con-sub.mp4 ha i sottotitoli che escono dai bordi laterali.  
Riapplica i sottotitoli con:  
- Massimo 30 caratteri per riga  
- A capo automatico se la riga è più lunga  
- Margini laterali di 60px  
Salva come ~/Video/finiti/video-01-con-sub-corretto.mp4

## Prompt 224

Voglio una dashboard personale per la mattina con questi widget:  
1. Email urgenti: email non lette di oggi con etichetta o priorità alta (max 5)  
2. Agenda del giorno: eventi del calendario di oggi con orario e titolo  
3. Task aperti: file TODO.md in ~/Note/ – gli item con [ ] non ancora completati  
4. Citazione del giorno: una citazione ispirazionale diversa ogni giorno  
5. Meteo: temperatura attuale e previsione per oggi a [inserisci la tua città]

Per ogni widget: titolo, dati, e timestamp di ultimo aggiornamento.  
Layout: sezione verticale per mobile, 2 colonne per desktop.  
Colori: dark mode – sfondo #1a1a2e, card #16213e, accento #e94560, testo bianco.

## Prompt 225

Crea il file ~/Dashboard/dashboard.html con il design della dashboard.  
Usa HTML e CSS puro (no JavaScript framework, no dipendenze esterne).  
Per ora metti dati segnaposto in ogni widget – li sostituiremo con dati reali.  
Il file deve funzionare aprendolo direttamente nel browser (file://).

## Prompt 226

Modifica ~/Dashboard/dashboard.html per il widget Email:  
- Leggi le email non lette di oggi da Gmail usando MCP gmail  
- Filtra per mittenti importanti (non newsletter, non notifiche automatiche)  
- Per ogni email: mostra mittente, oggetto, ora di ricezione  
- Se ci sono più di 5 email: mostra solo le 5 più recenti  
- Se non ci sono email urgenti: mostra "Nessuna email urgente oggi [OK]"  
Aggiorna il timestamp "Aggiornato alle: [ora corrente]" nel widget.

## Prompt 227

Modifica ~/Dashboard/dashboard.html per questi due widget:

Widget Agenda:

- Leggi gli eventi di oggi da Google Calendar usando MCP google-calendar
- Per ogni evento: orario, titolo, e (se c'è) luogo
- Ordina cronologicamente
- Se non ci sono eventi: "Giornata libera!"

Widget Task aperti:

- Leggi il file ~/Note/TODO.md
- Mostra gli item con la checkbox vuota: "- [ ] testo"
- Massimo 5 task, ordina per ordine di apparizione nel file
- Link cliccabile per aprire il file TODO.md

## Prompt 228

Crea il file `~/claude/agents/dashboard-updater.md`  
Il subagent si chiama "dashboard-updater" e ha questo compito:

1. Legge il file `~/Dashboard/dashboard.html`
2. Aggiorna il contenuto di ogni widget con dati freschi (Gmail, Calendar, note)
3. Aggiorna tutti i timestamp "Aggiornato alle: [ora]"
4. Sovrascrive `~/Dashboard/dashboard.html` con la versione aggiornata
5. Non modifica nulla del design CSS – solo i dati

Permessi: Read + Edit su `~/Dashboard/`, Read su `~/Note/`,  
accesso MCP gmail e google-calendar.

## Prompt 229

`/schedule` ogni giorno alle 07:30, esegui il subagent dashboard-updater  
per aggiornare la mia dashboard mattutina.

## Prompt 230

Aggiungi un widget "Meteo" alla dashboard:

- Usa l'API OpenWeatherMap (gratuita, richiede chiave API da [openweathermap.org](https://openweathermap.org))
- Mostra: temperatura attuale, condizioni (sole/nuvole/pioggia), previsione del pomeriggio
- Città: [inserisci la tua città]

Chiedi la chiave API, salvala come variabile d'ambiente `OPENWEATHER_API_KEY`,  
poi aggiorna il widget nella dashboard.

## Prompt 231

Aggiungi un widget "Citazione del giorno":

- Ogni mattina una citazione diversa (usa una lista di 365 citazioni predefinite in `~/Dashboard/citazioni.txt`)
- La citazione del giorno è determinata dal numero del giorno nell'anno (`dayOfYear % 365`)
- Stile: testo in corsivo, più piccolo, in fondo alla dashboard

Crea il file `citazioni.txt` con 20 citazioni di partenza.

## Prompt 232

`/schedule` ogni giorno alle 07:30, esegui il subagent dashboard-updater  
per aggiornare `~/Dashboard/dashboard.html` con dati freschi.

## Prompt 233

Il widget email nella dashboard mostra email di ieri invece di oggi.  
Aggiorna il codice che legge Gmail per filtrare solo le email  
arrivate dopo la mezzanotte di oggi (midnight locale, non UTC).

## Prompt 234

Controlla i permessi del file `~/Dashboard/dashboard.html`.  
Deve essere leggibile da tutti (`chmod 644`).  
Se necessario, correggi i permessi e verifica che il file sia valido.

## Prompt 235

Crea una skill "sistema-personale" che esegue questa sequenza ogni mattina:

1. dashboard-updater → aggiorna la dashboard con dati freschi
2. Se è venerdì: esegui report-settimanale → genera il report della settimana
3. Se è domenica: esegui organizza-downloads → organizza la cartella Downloads
4. Al termine: invia una notifica Telegram con il riassunto delle operazioni eseguite

Questa skill deve girare automaticamente alle 07:30 tramite `/schedule`.

## Prompt 236

Esegui questi test di verifica per il mio sistema integrato:

1. `/organizza-downloads` → funziona? (testa su `~/Downloads/test/`)
2. `/nota test-verifica-sistema` → salva correttamente?
3. `/cerca-note test` → trova la nota appena creata?
4. dashboard-updater → aggiorna `~/Dashboard/dashboard.html` senza errori?
5. Bot Telegram → risponde ai messaggi? (manda un messaggio di test)

Dimmi quali componenti passano e quali falliscono.

### Prompt 237

Crea la skill "sistema-personale" in ~/.claude/skills/sistema-personale/SKILL.md  
La skill esegue questa sequenza ogni mattina:

1. Esegui il subagent dashboard-updater per aggiornare ~/Dashboard/dashboard.html
2. Controlla il giorno della settimana:
  - Se venerdì: esegui anche la skill report-settimanale
  - Se domenica: esegui anche la skill organizza-downloads
3. Invia una notifica Telegram al numero TELEGRAM\_ALLOWED\_USER\_ID con:  
"Sistema mattutino [data]:"
  - Dashboard aggiornata
  - [se venerdì] Report settimanale generato: ~/Report/[YYYY]-W[WW]-report.md
  - [se domenica] Downloads organizzati: [N] file spostati"
4. Scrivi una voce nel file ~/sistema-log.txt con timestamp e operazioni eseguite

### Prompt 238

/schedule ogni giorno alle 07:30, esegui la skill /sistema-personale

### Prompt 239

Il sistema-personale ha fallito stamattina. Controlla il file ~/sistema-log.txt e dimmi quale componente ha generato un errore e per quale motivo.

### Prompt 240

Leggi il file ~/sistema-log.txt degli ultimi 30 giorni e dimmi:

- Quante volte ha girato il sistema mattutino
- Quante operazioni ha eseguito in totale
- Qual è l'operazione più frequente
- Ci sono giorni in cui non ha girato? Perché?

Poi dimmi una stima del tempo che avrei impiegato a fare le stesse operazioni manualmente.

### Prompt 241

Leggi il mio sistema-log.txt e il report-settimanale più recente.  
Basandoti su quello che vedi, suggerisci tre miglioramenti al sistema:

1. Un'operazione che non sta funzionando bene e come risolverla
2. Un'automazione che manca e che sarebbe utile aggiungere
3. Un'ottimizzazione per ridurre il tempo di esecuzione totale

*Totale prompt Capitolo 14 — Sei progetti end-to-end: 78*

## Capitolo 15 — Piano d'azione — 30 giorni per un sistema che funziona

### Prompt 242

Crea un file ~/Note/piano-30-giorni.md con questa struttura:

```
# Piano 30 giorni — [data inizio]
```

```
## Settimana 1
- [ ] Giorno 1: [obiettivo]
- [ ] Giorno 2: [obiettivo]
...
```

```
## Note giornaliera
### Giorno 1 — [data]
- Cosa ho fatto:
- Cosa ha funzionato:
- Problemi incontrati:
- Per domani:
```

### Prompt 243

Ieri ho lavorato su [quello che ho fatto].  
Oggi voglio: [obiettivo del giorno].  
Aiutami a iniziare il task più importante della mia lista.

### Prompt 244

Non so bene da dove iniziare oggi. Leggi ~/Note/piano-30-giorni.md e suggerisci il task più importante per il giorno N del piano.

### Prompt 245

Dimmi qual è il tuo piano per oggi. Non devo fare nulla – voglio capire come pensi prima di darti task.

### Prompt 246

Sono [nome], lavoro come [ruolo/attività] e ho appena installato Claude Code.  
Il problema principale che voglio risolvere è: [descrivi un problema reale].  
Senza fare nulla ancora: dimmi come mi aiuteresti e cosa mi servirebbe configurare.

### Prompt 247

Analizza questo file [scegli qualsiasi file di testo sul tuo computer]  
e dimmi: cosa fa, come potrebbe essere migliorato, e cosa devo sapere prima di modificarlo.

### Prompt 248

Prima di modificare il file, mostrami un piano delle modifiche che vorresti fare.  
Aspetta la mia approvazione prima di procedere.

### Prompt 249

Configura il MCP filesystem con accesso a:  
- ~/Documenti (o equivalente sul mio sistema)  
- ~/Downloads

Verifica che l'accesso funzioni leggendo un file di test.

### Prompt 250

Mostrami le skill bundled disponibili in Claude Code e descrivi le tre che ti sembrano più utili per il mio caso d'uso  
(lavoro su [descrivi il tuo settore/attività]).

### Prompt 251

/batch  
Elabora tutti i file .txt nella cartella ~/Documenti/note-vecchie/  
e genera un riassunto di ogni file. Salva i riassunti in ~/Documenti/riassunti/

### Prompt 252

Installa e configura il MCP [scelto].  
Verifica l'accesso con un task di test semplice:  
- Gmail: leggi le ultime 3 email  
- Calendar: mostra gli eventi di questa settimana  
Dimmi se funziona.

### Prompt 253

Crea un riassunto della mia settimana basandoti su:  
- I file modificati negli ultimi 7 giorni in ~/Documenti  
- [se hai MCP gmail] Le email importanti di questa settimana  
Salva il riassunto in ~/Documenti/riassunto-settimana.md

### Prompt 254

Oggi è il giorno [N] del mio piano Claude Code.  
Ho fatto: [descrivi brevemente cosa hai fatto]  
Problemi incontrati: [se ci sono]  
Domani voglio: [obiettivo del giorno N+1]  
Aggiungi questa nota al file ~/Note/piano-30-giorni.md nella sezione giusta.

### Prompt 255

Leggi il mio file ~/Note/piano-30-giorni.md  
e aiutami a fare il bilancio della settimana 1:  
- Cosa ho configurato e cosa funziona  
- Dove ho incontrato difficoltà  
- Cosa voglio fare diversamente nella settimana 2  
- Sono pronto per iniziare la settimana 2?

## Prompt 256

Sfogliami le skill disponibili su agentskills.so (o nel registro delle skill installate).  
Filtra per quelle utili a [descrivi il tuo tipo di lavoro].  
Suggeriscimi le 3 più adatte e dimmi come installarle.

## Prompt 257

Configura il MCP [nome].  
Dopo l'installazione, dimmi:  
- Cosa posso fare con questo MCP che prima non potevo  
- Quali sono le operazioni più utili  
- Ci sono limitazioni da sapere?

## Prompt 258

Crea una tabella in ~/Note/mcp-comparison.md con queste colonne:  
MCP nome | Installato il | Usato per | Utilità (1-5) | Note  
Riempila con i MCP che ho già configurato e aggiornala ogni volta che ne installo uno nuovo.

## Prompt 259

Crea un subagent "assistente-[nome-del-task]" per questo task specifico:  
[descrivi il task che vuoi delegare: es. "review dei commit di ogni giorno",  
"aggiornamento del file README quando i test cambiano",  
"generazione di summary delle note della settimana"]

Il subagent deve:  
- Avere un nome che descrive esattamente cosa fa  
- Avere accesso solo agli strumenti necessari (non tutti)  
- Produrre un output specifico (non generico)  
- Loggare le operazioni eseguite

## Prompt 260

Aiutami a fare il bilancio della settimana 2.  
Leggi ~/Note/piano-30-giorni.md e dimmi:  
- Quali skill ho trovato più utili e perché  
- Quali MCP aggiunto hanno cambiato il mio flusso di lavoro  
- Come sta andando il subagent creato il giorno 13?  
- Cosa voglio prioritizzare nella settimana 3?

## Prompt 261

Iniziamo il Progetto [N] del Capitolo 14.  
Ho letto il capitolo e so che il prerequisito principale è [prerequisito].  
Confermami che è configurato correttamente, poi guida il primo step.

## Prompt 262

Stiamo lavorando al Progetto 14.[N].  
Aggiorna ~/Note/piano-30-giorni.md con queste informazioni:  
- Quanto tempo ha richiesto ogni step  
- Dove ho incontrato difficoltà e come le ho risolte  
- Cosa avrei fatto diversamente sapendo quello che so ora

## Prompt 263

/loop 30m  
Ogni 30 minuti, controlla se ci sono file in ~/Downloads/ che non appartengono a nessuna categoria  
e segnalami la lista. Fermati quando la lista è vuota.

## Prompt 264

Prova queste situazioni problematiche per la mia automazione [nome]:  
1. Cosa succede se la cartella di input è vuota?  
2. Cosa succede se un file è corrotto o non leggibile?  
3. Cosa succede se il MCP non risponde?  
4. Cosa succede se l'API (Whisper o altro) restituisce un errore?

Per ogni scenario: descrivimi il comportamento attuale e suggerisci come migliorare la gestione degli errori.

## Prompt 265

Settimana 3 completata. Aiutami a valutare lo stato del mio sistema:

1. Quali automazioni sono operative e funzionano senza supervisione?
2. Cosa mi ha sorpreso (in positivo o in negativo) nel lavorare con i loop?
3. I progetti del Capitolo 14 sono stati più facili o più difficili del previsto?
4. Cosa voglio costruire nella settimana 4?

Leggi ~/Note/piano-30-giorni.md per il contesto completo.

## Prompt 266

Voglio creare una skill personale. Il problema che voglio risolvere è:

[descrivi il tuo problema: qualcosa che fai spesso, che è ripetitivo, e che Claude potrebbe fare per te]

Prima di creare la skill, aiutami a definire:

1. Input: cosa passerò alla skill ogni volta
2. Output: cosa voglio ottenere
3. Processo: i passi che Claude deve eseguire
4. Condizioni: quando deve fermarsi o chiedere conferma

## Prompt 267

/skill-creator

Crea una skill per [il problema definito sopra].

Segui il processo completo: ideazione, SKILL.md, test cases, grading.

## Prompt 268

La mia skill personale [nome] non funziona bene in questi casi:

[descrivi i casi problematici]

Senza riscrivere la skill da zero, suggerisci 3 modifiche incrementalmente al SKILL.md per migliorare il comportamento in questi casi specifici.

## Prompt 269

Crea la skill sistema-personale che orchestra tutti i componenti del mio sistema:

[elenca le skill e i subagent che hai creato nelle ultime 4 settimane]

L'orchestrazione deve avvenire ogni mattina alle 07:30.

Testa il sistema manualmente prima di schedarlo.

## Prompt 270

Crea un file ~/Note/toolkit-personale.md con l'inventario del mio sistema Claude Code:

## Skill installate

[elenca con una riga di descrizione per ognuna]

## Skill create personalmente

[elenca con: problema che risolvono + quando le uso]

## MCP configurati

[elenca con: a cosa servono nel mio flusso di lavoro]

## Subagent personalizzati

[elenca con: task delegato + frequenza d'uso]

## Automazioni attive

[elenca con: orario di esecuzione + cosa fanno]

## Progetti completati (dal Cap 14)

[elenca i progetti completati]

## Prompt 271

Leggi ~/Note/piano-30-giorni.md dall'inizio alla fine.

Poi aiutami a rispondere a queste domande:

1. Cosa ho costruito nei 30 giorni? (lista dei componenti del mio sistema)
2. Quale di questi ha avuto il maggiore impatto pratico?
3. Cosa non ho completato e perché?
4. Cosa voglio costruire nel mese 2?
5. Qual è una cosa che ho imparato su Claude Code che mi ha sorpreso?

## Prompt 272

Ho solo 15 minuti al giorno per il piano Claude Code.  
Suggerisci una versione ridotta del piano 30 giorni che rimanga significativa ma che si adatti a questo vincolo di tempo.  
Mantieni l'ordine delle settimane ma riduci il numero di task per giorno.

## Prompt 273

Posso dedicare sabato e domenica al piano Claude Code – circa 3 ore a weekend.  
Come dovrei redistribuire i task delle settimane per ottimizzare il weekend intensivo?

## Prompt 274

Leggi ~/Note/piano-30-giorni.md e identifica il mio anti-pattern più frequente nelle ultime due settimane. Basati su:

- Frequenza con cui non ho letto l'output
- Numero di volte in cui ho ripetuto gli stessi prompt senza creare skill
- Se ho mai avviato un loop senza condizione di stop
- Se ho mai schedulato qualcosa senza testarlo prima

Suggerisci come correggere il comportamento specifico che emerge.

*Totale prompt Capitolo 15 — Piano d'azione — 30 giorni per un sistema che funziona: 33*

## Capitolo 16 — La mappa dell'ecosistema

### Prompt 275

```
curl -fsSL https://claude.ai/install.sh | bash
```

Su macOS, Linux e WSL, questo comando installa Claude Code con aggiornamento automatico in background. Su Windows, il comando PowerShell equivalente è:

### Prompt 276

```
irm https://claude.ai/install.ps1 | iex
```

Se preferisci Homebrew (macOS) o WinGet (Windows), le varianti sono disponibili – ma nota che queste installazioni non si aggiornano automaticamente: dovrai eseguire periodicamente `brew upgrade claude-code` o `winget upgrade Anthropic.ClaudeCode`.

### I punti di forza del CLI

**\*\*Composabilità Unix.\*\*** Il CLI segue la filosofia Unix: fa una cosa bene e si integra con tutto il resto. Puoi fare pipe con qualsiasi output di sistema:

### Prompt 277

```
git diff main --name-only | claude -p "Revisionami questi file per problemi di sicurezza"
```

Questa composabilità è esclusiva del CLI. Nessun'altra superficie di Claude Code ti permette di fare pipe di dati da terminale direttamente nel modello.

**\*\*Headless e CI/CD.\*\*** Il flag `-p` (print mode) avvia Claude in modalità non interattiva: riceve il prompt, produce output, termina. Perfetto per GitHub Actions, GitLab CI, cron job, script automatizzati. La Desktop app e VS Code richiedono un'interfaccia utente attiva; il CLI no.

**\*\*Accesso completo ai comandi.\*\*** Tutti i comandi slash sono disponibili: /loop, /schedule, /worktree, /plan, /compact, /mcp – la lista completa. VS Code offre un sottoinsieme, e il Desktop app ha un'interfaccia grafica per molti di questi ma non per tutti.

**\*\*Configurabilità totale.\*\*** Il file settings.json offre granularità completa: permessi per tool specifici, hook PreToolUse e PostToolUse, configurazione modello, timeout, variabili d'ambiente. Tutto configurabile da riga di comando o file.

### Quando preferire il CLI

Scegli il CLI quando:

- Stai scrivendo codice o automatizzando task tecnici
- Hai bisogno di composabilità con altri strumenti Unix
- Vuoi girare Claude in CI/CD o script
- Lavori in ambiente Linux (l'unica opzione: Desktop app non supporta Linux)
- Hai bisogno del controllo granulare su permessi e configurazione

### Quando NON è necessario il CLI

Se stai lavorando su un documento Word, preparando una presentazione, o facendo ricerca generale, claude.ai o Claude Cowork sono probabilmente più adatti. Il CLI brilla con file tecnici, codice, e task automatizzabili – non è ottimizzato per il knowledge work generico.

[PRATICA] Identifica i tuoi 3 use case principali

Prima di leggere oltre, annota tre attività che fai regolarmente e che potresti delegare a Claude. Per ognuna, chiediti: ho bisogno di accesso a file locali? Ho bisogno di eseguire comandi? La risposta determina quale strumento scegliere. Questo esercizio torna utile nella sezione 16.7.

---

## 16.3 claude.ai: il fratello maggiore

claude.ai è la chat AI di Anthropic – il prodotto che la maggior parte delle persone conosce prima di scoprire Claude Code. Capire le differenze tra claude.ai e Claude Code CLI chiarisce perché hai bisogno di entrambi, non di uno solo.

[VER 2026-05-03 – <<https://claude.com/pricing>>]

### Cosa fa che Claude Code CLI non fa

**\*\*Interfaccia conversazionale visuale.\*\*** claude.ai offre rendering di Markdown, visualizzazione di immagini, caricamento di PDF. La chat è ottimizzata per la lettura umana – vuoi vedere un grafico nell'output? claude.ai lo renderizza. Nel terminale, vedi il testo grezzo.

**\*\*Accesso da mobile.\*\*** Le app Claude per iOS e Android sono interfacce di claude.ai. Non c'è un'app "Claude Code" standalone per mobile – ma puoi monitorare le sessioni cloud di Claude Code dall'app Claude su iOS (nel tab Code, se hai un piano che lo include).

**\*\*Integrazione con Office.\*\*** Il piano Pro include integrazione con Excel, PowerPoint e Word: puoi caricare un file Excel e chiedergli di analizzarlo. Nel CLI, dovresti prima convertirlo in un formato che Claude può leggere.

**\*\*Research.\*\*** Il piano Pro include la funzionalità Research: Claude conduce ricerche multi-step su argomenti complessi, simile a un assistente di ricerca. Non è disponibile nel CLI.

**\*\*Free tier.\*\*** claude.ai ha un piano gratuito (\$0). Claude Code richiede almeno un piano Pro.

### Cosa claude.ai non fa

claude.ai non accede ai tuoi file locali senza che tu li carichi manualmente. Non esegue comandi sul tuo computer. Non gestisce sessioni di lunga durata con contesto persistente del progetto (il CLAUDE.md non esiste in claude.ai). Non può fare pipe con i dati del terminale.

In breve: claude.ai è un cervello brillante con cui chatti; Claude Code CLI è un agente che lavora con te nel tuo ambiente.

### Quando claude.ai sorprende positivamente

Una cosa che spesso sorprende chi viene dal CLI: claude.ai è notevolmente più bravo nella generazione di testo lungo e strutturato – saggistica, narrativa, reportage. Il motivo non è il modello (è lo stesso), ma il contesto d'uso: claude.ai è ottimizzato per conversazioni dove il testo è il prodotto finale, non uno step intermedio verso un'azione su un file. Se devi scrivere un piano strategico, una email complessa, o un articolo, claude.ai è spesso più efficace del CLI per quel tipo di output.

Viceversa, se vuoi che Claude legga un file Excel, estraiga dati, e scriva il risultato in un JSON – azione su file, non solo testo – stai fuori dalla zona ottimale di claude.ai e hai bisogno del CLI o di Cowork.

### Come usarli insieme

Il pattern più efficace è complementare:

## Prompt 278

Strumenti Desktop che non esistono nel CLI puro:

- Diff viewer visuale con commenti inline
- Sessioni parallele con sidebar
- Preview app integrata (dev server nel pannello)
- Monitoring automatico PR + CI
- Gestione grafica scheduled tasks locali
- Plugin marketplace con interfaccia grafica

## Prompt 279

Installare VS Code Extension:

1. Apri VS Code

2. Cmd+Shift+X (Mac) o Ctrl+Shift+X (Windows/Linux)
3. Cerca "Claude Code"
4. Clicca Install
5. Clicca sull'icona Spark nell'editor toolbar (richiede un file aperto)

## Prompt 280

Come restare aggiornato sull'ecosistema Claude:

1. Iscriviti al changelog di Claude Code: [code.claude.com/changelog](https://code.claude.com/changelog)
2. Segui @AnthropicAI su Twitter/X per annunci di prodotto
3. Leggi le release notes di Claude Code: [github.com/anthropics/claude-code/releases](https://github.com/anthropics/claude-code/releases)
4. Controlla la pagina news Anthropic: [anthropic.com/news](https://anthropic.com/news)
5. Ogni 30 giorni, fai:  
    `claude --version`  
    (e verifica se è uscita una versione più recente con nuove funzionalità)

*Totale prompt Capitolo 16 — La mappa dell'ecosistema: 6*

---

## Totale prompt nel pacchetto: 280

Versione: bonus pack v1.0 Generato da: LIBRO-FINALE-v2.md (maggio 2026) Per aggiornamenti: [brunotruant.com/claudecodepertutti/](https://brunotruant.com/claudecodepertutti/)