Lezione 7 · Online

Lezione 7 - Vibecoding: integrare l'AI nel proprio flusso di lavoro (esempio pratico).

Venerdì 17 aprile 2026 · 14:00-16:00 · Corso avanzato sull'Intelligenza Artificiale

Obiettivo: passare da compito a strumento operativo con un flusso verificabile.

Docente: Dott. Daniele Dragoni

Profilo docente e contatti.

Dott. Daniele Dragoni

CONTATTO

daniele.dragoni@uniroma3.it

PERCORSO DOTTORALE

Dottorato nazionale in Teaching and Learning Sciences

AFFILIAZIONE

Università Roma Tre e Università di Macerata

TEMA DI RICERCA

Educazione e AI

LAUREE

Scienze cognitive della comunicazione e dell'azione; E-learning e media education

Quanto usate già l'AI nel vostro lavoro?

Rispondete al sondaggio: ci servono 90 secondi.

- **Q1** Perché seguite questo corso?
- **Q2** Usate già strumenti di AI nel lavoro? (Sì/No)
- **Q3** Qual è lo scopo principale per cui usate AI oggi?
- **Q4** Conoscete già il vibe coding?

SONDAGGIO LIVE

Inquadra ora il QR e invia le risposte.

Apri sondaggio

Ecco cosa avete risposto.

Sessione 2026-05-12 · risposte raccolte: 100 · aggiornato alle 08:27

Uso strumenti AI

Si	78%
No	

Motivazione

- Velocizzare consegne e prototipi tecnici — **43%**
- Automatizzare task ripetitivi di sviluppo — **35%**
- Capire limiti, qualita e rischi dell'AI — **22%**

Familiarita con il vibecoding

- Si, lo applico gia in progetti reali — **24%**
- Ne ho sentito parlare ma non lo applico — **49%**
- No, e nuovo per me — **27%**

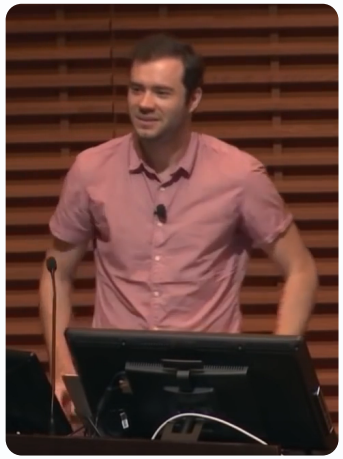
Scopi principali

1. Scrittura, correzione e rifattorizzazione codice

Vibecoding: descrivere invece di scrivere.

Tu descrivi → AI genera → Tu verifichi

- Da un'idea a un prototipo funzionante in pochi minuti.
- Con metodo e competenza tecnica, si va ben oltre le bozze: applicazioni, analisi dati, automazioni.
- Il risultato va sempre verificato: la responsabilita resta a chi lo usa.



Termine coniato da Andrej Karpathy, febbraio 2025. Fonte: Wikimedia Commons, Gladwin Analytics (CC BY 3.0).

Le 7 fasi della progettazione.

Fasi concettuali, non sequenziali rigide — in un progetto reale si sovrappongono. In questa lezione: §1=F1, §2=F2+F4, §3=F3+F5+F6, §4=F7.

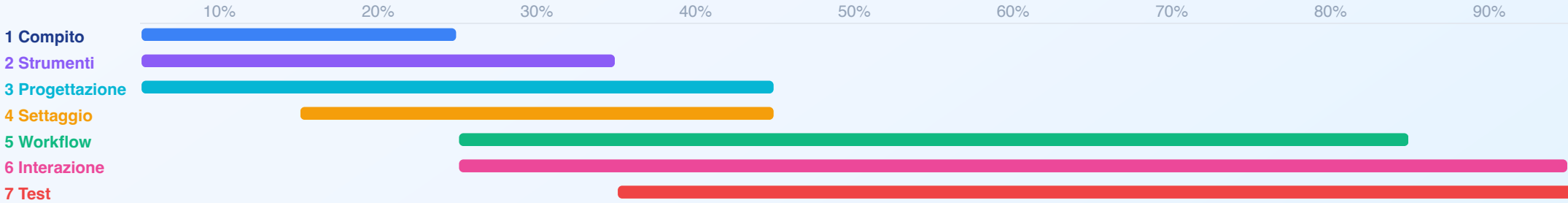
La mappa in 7 fasi

Avvia

1. Definire il compito
2. Scegliere strumenti e fonti
3. Progettare il flusso
4. Preparare l'ambiente
5. Gestire consegne e controlli
6. Costruire con l'AI
7. Testare e rifinire

Fase 1 — Definire il compito
Obiettivo, vincoli e risultato atteso.

Come si distribuiscono in un progetto reale:



1) Il compito: prima decidere, poi chiedere.

Che risultato voglio, per chi, con quale rischio e quale delega?

Obiettivo
Risultato concreto in una frase

Destinatario
Chi userà l'output

Revisore
Chi lo validerà

Rischio
Basso / medio / alto impatto

Delega
AI propone / esegue / umano decide

Done
Quando il compito è davvero c

Se non sai dire quando è finito, non hai ancora definito il compito.

1b) Obiettivi: osservabili, misurabili, monitorabili.

Osservabile

Si vede chiaramente nel risultato finale.

Misurabile

Ha una soglia o metrica esplicita.

Monitorabile

Può essere tracciato nel tem

Entro [tempo], ottenere [risultato osservabile], con [metrica + soglia], tracciato tramite [strumento].

1c) Specifica operativa: Input → Output → Vincoli.

Input

- Fonti consentite
- Dati disponibili
- Documenti ed esempi
- Policy di riferimento

Output

- Formato atteso
- Struttura e lunghezza
- Tono e registro
- Destinatario finale

Vincoli

- Normativi e compliance
- Tecnici e di formato
- Temporali e budget
- Privacy e riservatezza

*Se input e output non sono espliciti, il **prompt** resta ambiguo.*

1d) Esempio pratico: ticket ricambio urgente.

Esempio: ticket ricambio urgente con AI Avvia

- **Obiettivo** Ticket ricambio urgente per cuscinetto motore linea A
- **Destinatario** Manutenzione e magazzino
- **Osservabile** Codice ricambio, stato ticket e data consegna
- **Misurabile** Presa in carico < 15 min, chiusura < 24 ore
- **Monitorabile** Timeline stati ticket e ora di chiusura
- **Delega** AI classifica priorita; umano approva ordine e chiusura

Definition of Done: ricambio consegnato, montaggio confermato, ticket chiuso.

1e) Tre modi di progettare: quando entra l'AI?

La definizione del compito si fa sempre prima. Poi si sceglie il grado di delega.

Senza AI

Carta e penna. Massimo controllo e consapevolezza. L'AI entra solo dopo.

Ticket: compilo la tabella dei 6 campi da solo, poi passo all'AI.

Con AI

Io alla regia, AI come supporto: esplorare, comparare, scomporre.

Ticket: definisco obiettivo e rischio, chiedo all'AI di espandere i criteri misurabili.

Delegata all'AI

AI imposta il compito. Veloce, ma serve

Ticket: chiedo all'AI di generare l'i specifica, poi rivedo tutto.

Regola: più aumenta la delega progettuale, più deve aumentare la verifica umana.

2) Definizione degli strumenti.

Fasi 2 e 4 — scegliere strumenti e preparare l'ambiente prima di iniziare.

ChatGPT Projects

Progetti con istruzioni e file persistenti

Contesto stabile tra sessioni, istruzioni di progetto, file allegati. Buono per lavori continuativi senza IDE.

Perplexity Spaces

Spazio di ricerca con fonti e memoria

Risposte con fonti verificabili, documenti caricati nello Space, persistenza del contesto. Ideale per ricerca (codici, normative).

VS Code + AI

IDE con Copilot / Claude

Contesto progetto completo, file persiste integrato. Puoi costruire interfacce per g coinvolti.

Supporto: documenti fonte, checklist, repository.

Criterio: complessità, tracciabilità, delega.

Regola: prima metodo e flusso, poi s

2b) Perché VS Code + AI?

I primi due sono buoni, ma per il nostro compito serve più controllo.

ChatGPT Projects e Perplexity Spaces funzionano bene per:

- Brainstorming e esplorazione iniziale
- Ricerca con fonti verificabili (normative, codici ricambio)
- Lavori continuativi senza necessità di codice
- Contesto persistente tra sessioni

Ma il nostro ticket richiede:

- **Verificabilita** — ogni file è tracciato, ogni modifica è visibile
- **Controllo** — accetto, rifiuto o modifico ogni singola propost
- **Persistenza strutturata** — file, cartelle, versioni, non solo
- **Riproducibilita** — posso rieseguire, testare, iterare
- **Interfacce** — posso costruire UI per far interagire gli attori

ChatGPT Projects	Perplexity Spaces	**VS Code + AI**
→	→	✓
buono per esplorare	buono per ricercare	**per costruire e verificare**

2c) Pre-settaggio: preparare VS Code prima di iniziare.

Ambiente, contesto e regole vanno impostati prima di scrivere il primo prompt.

Virtual environment

Python venv isolato per il progetto. Dipendenze tracciabili, riproducibili, senza conflitti.

```
python -m venv .venv && pip install ...
```

Knowledge base

Cartella con documenti di riferimento: normative, cataloghi, procedure come fonti.

```
knowledge-base/ normative/ cataloghi/
```

Regole e skill Claude

CLAUDE.md per istruzioni globali, .claude/skills/ per competenze riusabili. L'AI sa come comportarsi.

```
CLAUDE.md .claude/skills/ .github/copilot-instructions.md
```

Struttura progetto

Cartelle per output, bozze, template, log decisioni. Tutto organizzato p partire.

```
output/ drafts/ templates/ decisions/
```

Regola: l'ambiente si prepara prima. Se parti senza struttura, l'AI amplifica il caos.

2d) Il primo prompt: specifica operativa.

Questo e il prompt che mandiamo all'AI dopo aver preparato l'ambiente.

```
Sono un ingegnere responsabile della manutenzione di una linea produttiva industriale. Sto progettando un sistema di ticket per gestire richieste di ricambi
con supporto AI.

Ho già definito questi elementi:

- Obiettivo: generare un ticket ricambio urgente per un cuscinetto motore della linea A
- Destinatario: reparto manutenzione e magazzino
- Rischio: medio-alto (fermo linea se il ricambio non arriva in tempo)
- Delega: l'AI classifica la priorità e propone il codice ricambio; l'umano approva l'ordine e la chiusura

Aiutami a completare la specifica operativa espandendo questi punti:

1. Criteri osservabili: cosa deve comparire nel ticket affinché sia completo e verificabile?
2. Criteri misurabili: soglie temporali e metriche concrete (presa in carico, chiusura, consegna).
3. Criteri monitorabili: come si traccia lo stato del ticket nel tempo?
4. Input necessari: quali dati servono per aprire il ticket (es. codice macchina, tipo guasto, foto)?
5. Output atteso: formato e struttura del ticket generato.
6. Vincoli: normativi, tecnici, temporali, di budget.
7. Definition of Done: quando il compito è davvero chiuso?
8. Struttura progetto VS Code: proponi l'alberatura completa delle cartelle del progetto. I nomi delle cartelle devono essere in italiano e comprensibili, t
che sono standard in inglese (src, tests, templates, output, drafts, .venv, .claude, .github). Includi: ambiente virtuale Python, base di conoscenza (catalo
normative, procedure), configurazione AI (CLAUDE.md, .claude/skills/, copilot-instructions.md), templates, output, bozze, registro-decisioni, sorgenti e tes

Rispondi in formato tabellare strutturato. Non inventare dati specifici dell'azienda: dove servono dati reali, metti un segnaposto [DA COMPILARE]. Proponi c
concrete che io possa accettare, modificare o scartare.
```

2e) Interazione dopo il prompt: costruire il sito.

Dalla specifica al prototipo funzionante — con due problemi risolti in tempo reale.

Tu

Crea un'interfaccia web Flask per gestire i ticket. Form inserimento dati macchina, lista ticket aperti con stato e priorità AI, pulsante **Approva ordine** per il responsabile.

AI

Genera `app.py` + `templates/index.html` : route `GET /` e `POST /ticket/nuovo`, integrazione SQLite, chiamata all'API AI per la priorità.

Problema 1

Il form invia ma risponde **405 Method Not Allowed**.

«Ho questo errore quando clicco Invia ticket.»

Fix 1

La route accettava solo GET. Aggiunto `methods=['GET','POST']`. Riavviare il serv

Problema 2

Il campo **Priorità AI** nella lista ticket appare sempre vuoto.

«La priorità proposta non compare nella pagina.»

Fix 2

Il template leggeva `ticket.priority` invece di `ticket.ai_priorita`. Aggiornat
Jinja2.

Prototipo funzionante: form → analisi AI → lista ticket con priorità → gate approvazione umana.

2f) Buone pratiche per lavorare con l'AI.

8 regole operative per risultati affidabili, controllo e qualità.

Spezzetta i compiti
Un compito per chat. Pulisci il contesto tra un task e l'altro.
«Apri una chat per ogni funzione: login, dashboard, report.»

Chiedi di spiegare prima di agire
L'AI deve dirti cosa intende fare prima di eseguire.
«Prima di scrivere codice, dimmi cosa intendi fare passo per passo.»

Richiedi sempre approvazione
Nessuna modifica senza il tuo via libera esplicito.
«Non modificare nessun file finché non ti dico OK.»

Fatti interrogare
Chiedi all'AI di farti domande per chiarire cosa vuoi.
«Fammi 5 domande per capire meglio cosa devo costruire.»

Usa la modalita progetto
CLAUDE.md, copilot-instructions.md, .env per chiavi e segreti.
«Regole in CLAUDE.md, API key e segreti nel .env.»

Documenta tutto
Registra decisioni, modifiche e motivazioni in un log.
«Ogni volta che fai una modifica, aggiorna il CHANGELOG.»

Usa il versioning
Git e GitHub per tracciare ogni cambiamento e tornare indietro.
«Fai un commit dopo ogni funzionalità completata.»

Chiedi alternative
Chiedi più soluzioni a confronto con pro e contro.
«Dammi 3 soluzioni diverse per gestire l'autenticazione.»

Il contesto operativo: cosa vede l'AI.

SCENARIO **"Riscrivi l'introduzione in tono formale"**

Il **prompt** è solo una parte: contano anche **regole**, **skill**, **RAG** e **tool call**.

Prompt | Regole | File | Cronologia | Skill | Tool call | RAG | Context | LLM | Output

← ▶ Avvia → Reset

1/10

Fonti → Context window → LLM → Output

Fonti che entrano nel contesto

7 sorgenti distinte

Prompt
richiesta esplicita
riscrivi in tono formale

Regole
sistema + progetto
ruolo · tono · vincoli · CLAUDE.md

File
allegati
introduzione-bozza.md

Cronologia
storia chat
correzioni · preferenze

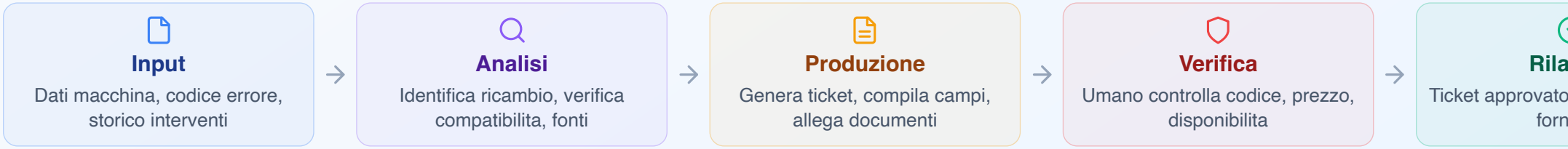
Skill
procedura riusabile
/guida-review

Tool call
schemi funzioni
read_file() · search()

RAG

3) Progettazione del flusso.

Fasi 3, 5 e 6 — flusso operativo, gestione consegne e costruzione con l'AI.



Anche in locale: questo flusso puo girare con un modello locale come **Gemma 4** (via Ollama/LM Studio) — nessun cloud, dati che restano sul PC, costo zero.

Checkpoint tra ogni fase
Criteri espliciti di avanzamento

Handoff chiaro
Chi fa cosa, quando passa al prossimo

Bozza ≠ versione finale
Separazione netta tra draft e rilasci

4) Test e raffinamento.

Fase 7 — verifica iterativa: testare, valutare, correggere fino a stabilità.

Test su casi reali
Tre livelli: un caso semplice, uno medio, uno critico.
«Testa il ticket con errore noto, errore ambiguo e guasto multiplo.»

Valutazione con rubrica
Accuratezza, chiarezza, verificabilità, utilità pratica.
«Per ogni output: e corretto? e chiaro? posso verificarlo? e utile?»

Errori ricorrenti
Identifica pattern di errore e correggi il workflow alla radice.
«Se sbaglia sempre il codice fornitore, aggiungi un vincolo al prompt.»

Raffinamento iterativo
Ripeti finche il risultato non e stabile e riproducibile.
«Rilancia 3 volte lo stesso caso: se esce uguale, e pronto.»

Output finale: strumento pronto all'uso con processo documentato.

5) Cosa abbiamo costruito nel progetto.

Simulazione end-to-end di un ticket urgente con AI assistiva e gate umano obbligatorio.

Workflow completo del ticket

Input Analisi Produzione Verifica Rilascio

Timeline fasi, stato runtime e handoff tra operatore, AI, responsabile umano e magazzino.

Passaggio tra fasi bloccato se mancano dati o approvazione.

Raccolta dati ticket guidata

Scenario precompilabile: cuscinetto, sensore, cinghia.

Campi tecnici e operativi: linea, macchina, guasto, sintomo, impatto, dati macchina, ambiente.

Proposta AI robusta

Prova chiamata a Ollama con endpoint e modello configurabili.

Fallback locale euristico se Ollama non risponde.

Output sempre completo: priorità, codice ricambio, confidenza, motivazione, dati mancanti.

Persistenza e audit

SQLite via API: creazione ticket con POST /api/tickets e aggiornamento con PATCH.

Eventi tracciati in ticket_events per audit completo dei pa

Approvazione umana obbligatoria

Coda ticket da approvare con checklist tecnica (codice, disponibilita).

Approvazione una sola volta, con possibilita di revoca.

Nessun passaggio a Rilascio senza approvazione valida completi.

Ricambi, KPI e strumenti demo

Catalogo ricambi, associazione per ticket, quantita e sub

KPI demo lato frontend sui tempi tra fasi + log eventi tick

Simulazione automatica, reset UI e reset UI + database reset-demo.

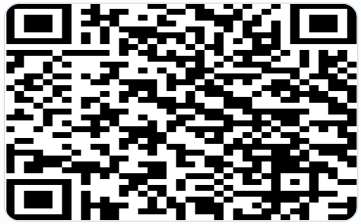
In sintesi: ciclo completo di ticket urgente con AI assistiva, controllo umano obbligatorio e tracciamento persistente su data

Questionario di gradimento.

8 dimensioni su scala 1–7 + suggerimenti liberi (60 secondi).

1. **Chiarezza — I contenuti sono stati esposti in modo chiaro?**
2. **Utilità — Quanto è applicabile al vostro lavoro quotidiano?**
3. **Novità — Quanto era nuovo per voi ciò che avete visto?**
4. **Ritmo — Il ritmo della lezione è stato adeguato?**
5. **Materiali — Slide ed esempi pratici efficaci?**
6. **Interazione — Sufficiente spazio per domande?**
7. **Motivazione — Quanto vi sentite motivati a sperimentare?**
8. **Raccomandazione — Lo suggerireste a un collega?**

QUESTIONARIO FINALE



Inquadra il QR e invia il tuo feedback.

Apri sondaggio

Ecco il vostro feedback.

In attesa delle risposte…

Grazie per l'attenzione.

Adesso tocca a voi: provate il flusso, sperimentate e migliorate un passaggio alla volta.

*Un buon **workflow** non nasce perfetto: si testa, si misura, si affina.*