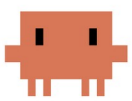
Corso Django con Claude Code

dr. Alessandro Dentella

slides: https://corsi.e-den.it



Programma

- Architettura Django:
  - Progetto/app
  - Comandi gestione
  - MVT
- Modelli (db, lazy loading)
- Template (ereditarietà)
- Admin (struttura, inlines…)
- Management commands
- API:
  - DRF (Django rest Framework)
  - Serializzatori
- App: panoramica più comuni (DRF, Allauth, Dj Toolbar, Dj Estension, Dj storages, Cors)
- Debug… testing... deploy...

Django

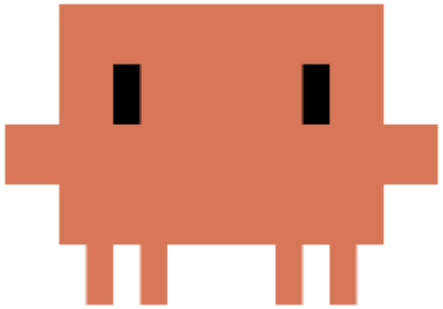
vuejs

- Struttura
- Componenti
- Routing
- Axios
- Persitent Storage (Pinia)
- Debug… testing... deploy….

& more...

- Http
- Html
- css

Alessandro Dentella
sandro.dentella@gmail.com



Claude Code è uno strumento da riga di comando che consente agli sviluppatori di **delegare compiti di programmazione a Claude** (l'intelligenza artificiale di Anthropic) direttamente dal proprio terminale.

In termini semplici:

Claude Code trasforma Claude in un assistente di programmazione autonomo che può:

- Leggere e modificare file nel progetto
- Eseguire comandi nel sistema
- Scrivere, testare e correggere codice
- Implementare funzionalità complete partendo da una descrizione testuale

Claude Code: Come avere un praticante esperto sempre disponibile
Immagina un collaboratore che:
🎓 Conosce perfettamente Django, Vue.js e tutte le best practice
⚡ Scrive codice 10x più veloce di te
🔍 Non dimentica mai la sintassi
🤝 Lavora esattamente come gli chiedi
❓ Ma ha bisogno che tu decida cosa costruire e come strutturarlo
La differenza cruciale:
Il praticante classico impara mentre lavora
Claude Code già sa programmare, ma serve tu sappia progettare

Alessandro Dentella
sandro.dentella@gmail.com

Claude Code: Il tuo collega che scrive codice

Prima (programmazione tradizionale)	Con Claude Code
Ricordi la sintassi	Descrivi l'obiettivo
Scrivi riga per riga	Supervisioni l'implementazione
Cerchi la documentazione	Claude la conosce già
Debug manuale	Debug assistito e iterativo
20 minuti per task semplice	2 minuti per task semplice

Risultato

Ti concentri su **cosa** costruire, Claude si occupa del **come**

Cosa cambia per noi?

Alessandro Dentella
sandro.dentella@gmail.com

Cambio ruolo

Sviluppatore Potenziato dall'IA

Direi che il ruolo non corrisponde esattamente a nessuna delle categorie tradizionali, ma è un ibrido evolutivo che combina elementi di tutti e tre:

Architetto Software (60%)

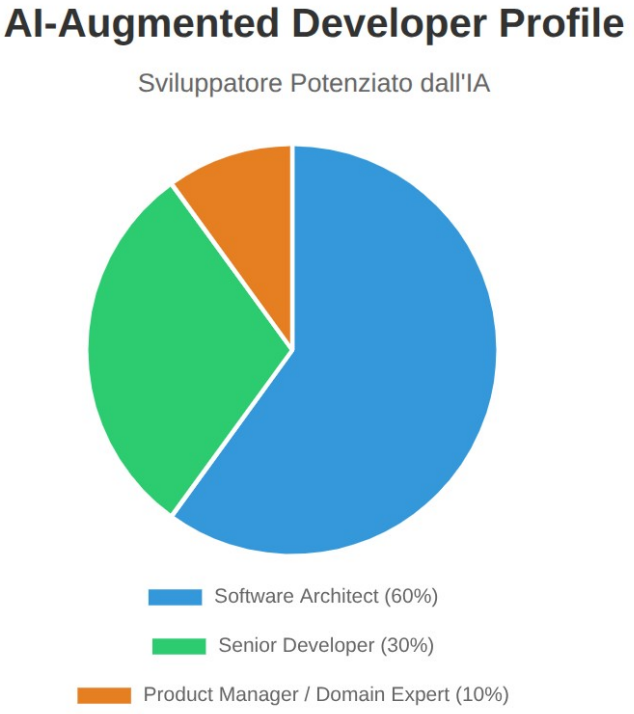
Competenze di progettazione: Deve sapere cosa è possibile costruire e come strutturarlo
Riconoscimento di schemi: Riconosce quali architetture sono appropriate per ogni contesto
Valutazione dei compromessi: Sa valutare vantaggi e svantaggi di diverse soluzioni proposte dall'IA
Requisiti non funzionali: Conosce le implicazioni di prestazioni, scalabilità, sicurezza

Sviluppatore Esperto (30%)

Capacità di revisione del codice: Deve saper leggere e valutare criticamente il codice generato
Approccio al debug: Capisce dove e come cercare problemi
Conoscenza delle buone pratiche: Riconosce codice problematico e schemi da evitare
Conoscenza dei framework: Conosce le capacità e i limiti dei framework

Responsabile di Prodotto/Esperto di Dominio (10%)

Formulazione dei requisiti: Sa formulare specifiche chiare e complete
Ragionamento per storie utente: Traduce bisogni in requisiti implementabili
Definizione delle priorità: Distingue l'essenziale dal desiderabile



Alessandro Dentella
sandro.dentella@gmail.com

Nuovo paradigma

La competenza critica: "Progettazione delle Richieste"

Quello che emerge come capacità centrale è una nuova competenza che potremmo chiamare "Progettazione delle Richieste" o "Progettazione di Sistemi Conversazionali":

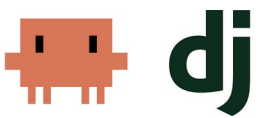
- ***Scomposizione dei problemi:*** Dividere un sistema complesso in richieste/attività gestibili
- ***Scrittura delle specifiche:*** Articolare requisiti in modo preciso ma comprensibile all'IA
- ***Criteri di validazione:*** Definire cosa rende una soluzione "corretta"
- ***Raffinamento iterativo:*** Guidare l'IA attraverso cicli di miglioramento

Cosa NON serve più (o serve meno):

- ❌ Memorizzazione della sintassi dettagliata
- ❌ Ricordare nomi esatti di funzioni e interfacce
- ❌ Implementazione meccanica di algoritmi standard
- ❌ Scrittura di codice ripetitivo

Cosa serve DI PIÙ:

- ✅ Comprensione concettuale profonda
- ✅ Capacità di valutazione critica
- ✅ Pensiero architetturale
- ✅ Conoscenza del dominio applicativo
- ✅ Capacità comunicative (con l'IA!)



Alessandro Dentella
sandro.dentella@gmail.com

In soldoni

Spenderemo più tempo a

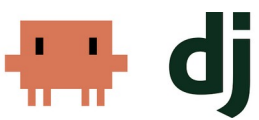
- Consultare l’IA
- Guidare l’IA
- Correggere l’IA
- … ricordaci che è solo uno strumento, non possiamo arrabbiarci se non ha capito/fatto quanto chiesto
- Escogitare un modo per metterle il paraocchi, perché faccia solo quello che ci serve

Alessandro Dentella
sandro.dentella@gmail.com

Conoscerla

Per questo dobbiamo conoscerla bene

- Come è fatta
- Come si comporta
- Che strumenti ci mette a disposizione



Alessandro Dentella
sandro.dentella@gmail.com

Competitor di Claude Code

- CLI
- WEB
- VS Plugin

- Modelli

- Mercato in rapidissima evoluzione

Nome	Tipo	Caratteristica Principale	Costo
GitHub Copilot	Plugin IDE	Completamento inline, ampia adozione	10€/mese
OpenAI Codex	API/Deprecato	Modello originale dietro Copilot (ora deprecato)	-
Claude Code	CLI	Autonomo, accesso file e comandi	API usage
Cursor	IDE completo	Fork VS Code con AI integrata	20$/mese
Kiro (AWS)	IDE completo	Spec-driven development, agent hooks	Gratis (preview)
Cline	Plugin VS Code	Autonomo, esegue comandi nel terminale	API usage
Aider	CLI	Pair programming, supporta molti LLM	API usage
Windsurf	IDE completo	Cascade mode per modifiche complesse	Freemium
Codeium	Plugin IDE	Alternativa gratuita a Copilot	Gratis
Augment	Plugin IDE	Context-aware, memoria progetto	20$/mese
Amazon CodeWhisperer	Plugin IDE	Ottimizzato per AWS	Gratis
Tabnine	Plugin IDE	Opzione self-hosted	Freemium
Continue.dev	Plugin VS Code	Open source, modelli locali	Gratis
Roo-cline	Plugin VS Code	Fork di Cline con migliorie UI	API usage
Qwen Coder	CLI/Locale	Modello locale di Alibaba	Gratis
Sourcegraph Cody	Plugin IDE	Code search e comprensione	Freemium
Replit Ghostwriter	Web/IDE Cloud	Ambiente cloud completo	10$/mese
Tabby	Self-hosted	Copilot self-hosted open source	Gratis + infra
v0.dev (Vercel)	Web	Genera componenti React/Vue	Freemium
Lovable (ex GPT Engineer)	Web	Genera app complete da prompt	Pagamento
Bolt.new (StackBlitz)	Web	Prototipazione rapida full-stack	Freemium
Claude.ai	Web Chat	Analisi codice, lungo contesto	20$/mese Pro
ChatGPT	Web Chat	Versatile, spiegazioni	20$/mese Plus
Devin (Cognition)	Web/Autonomo	"AI Software Engineer" (beta privata)	TBD
Gemini Code Assist	Plugin IDE	Integrato con Google Cloud	Pagamento
GPT Engineer	CLI	Genera progetti da zero	API usage

ella
om

Anticipo

Ci occuperemo principalmente dei concetti che influenzano il buon comportamento del nostro collaboratore

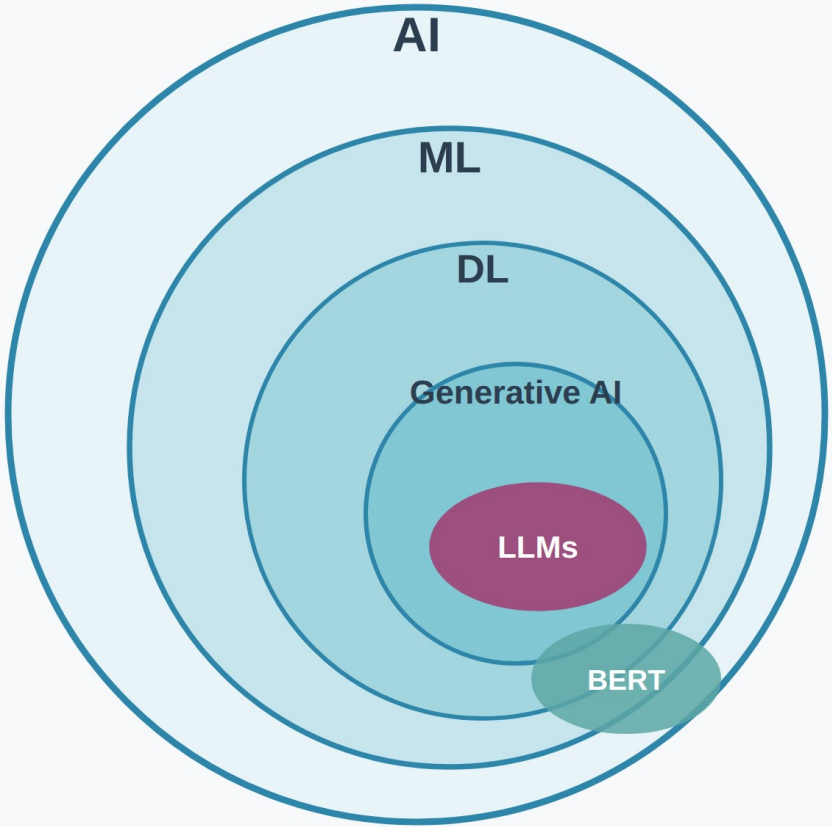
- Modello
- Agente
- Prompt
- Thinking
- Contesto
- Memorie

Alessandro Dentella
sandro.dentella@gmail.com

- Apriamo claude code e proviamo “Ciao sono sandro, chi sei”
- Poi chiediamogli chi siamo…
- /clear
- Chi siamo?
- Ora facciamogli fare una script che faccia un elenco di tutti i file in una cartella in una tabella con opzioni per decidere se ricorsivo o meno e quante estensioni max provare

Tassonomia dell'Intelligenza Artificiale

Relazioni tra AI, ML, DL, GenAI e LLM



🔷 Artificial Intelligence (AI)

Sistemi che simulano intelligenza umana

🔷 Machine Learning (ML)

Apprende dai dati senza programmazione esplicita

🔷 Deep Learning (DL)

Reti neurali profonde con molti layer

🔷 Generative AI

Crea nuovo contenuto (testo, immagini, audio)

🔸 Large Language Models (LLMs)

Generano testo nuovo token-by-token (GPT, Claude, Llama)

🔸 BERT-like Models

Deep Learning per linguaggio ma NON generativi (encoder-only)

Cos'è un Modello Linguistico

Concetti Fondamentali

Token: Come l'AI "vede" il testo (pezzi di parole)

Predizione probabilistica: Prevede il prossimo token più probabile

Context window: La memoria di lavoro (es. 200K token)

Parametri: La "dimensione del cervello" (Opus > Sonnet > Haiku)

Training vs Inference

Training = apprendimento dai dati. Inference = utilizzo del modello.

Alessandro Dentella
sandro.dentella@gmail.com

Dalle Parole ai Numeri: Gli Embeddings

Il Problema

I computer lavorano solo con numeri, ma noi comunichiamo con parole. Come facciamo a far capire il **significato** delle parole a una macchina?

In pratica: Parole con significato simile hanno numeri vicini nello spazio

La Soluzione: Embeddings

Vettorializzazione

Ogni parola diventa un vettore di numeri che ne cattura il **significato semantico**

Esempio classico:

```
re - uomo + donna ≈ regina
```

Visualizzazione: spazio vettoriale con re, regina, uomo, donna

Perché è importante: Gli embeddings permettono all'LLM di "comprendere" le relazioni tra concetti.

I Parametri: La Conoscenza Congelata

175B

Parametri (esempio GPT-3)

Cosa sono?

Miliardi di numeri che rappresentano la conoscenza acquisita durante il training.

Definiscono la **"dimensione"** e la **capacità** del modello.

La Metafora della Fotografia

Durante il Training

Il modello "osserva" miliardi di dati fino a una data specifica. Questa conoscenza viene codificata nei parametri.

Dopo il Training

I parametri sono **congelati**. Come una fotografia, catturano un momento ma non si aggiornano.

Knowledge Cutoff: La "data dello scatto". Il modello non ha conoscenza diretta degli eventi successivi — i parametri restano immutati.

Come "Pensa" un LLM

Meccanismo di Base

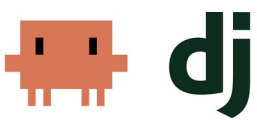
Pattern matching, non vera comprensione

Allucinazioni: Può generare informazioni plausibili ma false

Stateless: Non ricorda tra conversazioni diverse

Temperature: Controllo tra determinismo e creatività

> **Importante:** Il modello non "sa" - riconosce pattern nei dati di training.



Alessandro Dentella
sandro.dentella@gmail.com

Dai Numeri alle Parole: Predizione Probabilistica

Come Genera Testo

L'LLM non "capisce" nel senso umano. **Predice la parola più probabile** che dovrebbe seguire, basandosi su miliardi di esempi visti durante il training.

Esempio: "Il gatto è sul..."

- **tavolo**: 45%
- **tetto**: 30%
- **letto**: 15%
- altro: 10%

Il Ruolo della Temperatura

Temperatura Bassa

Sceglie sempre le opzioni più probabili

→ Preciso, ripetitivo

Temperatura Alta

Dà più chance alle opzioni meno probabili

→ Creativo, imprevedibile

Analogia: Come scegliere al ristorante - vai sul "solito" o provi qualcosa di nuovo?

Perché "sbaglia": L'LLM predice con pattern statistici, non comprensione reale. Pattern errati generano contenuti plausibili ma falsi.

Hai assolutamente ragione - è stata un'operazione COMPLETAMENTE INUTILE.

🤔 **Analisi del mio errore di logica:**

Il problema era: Policy troppo restrittiva (solo 7 bucket invece di 85)

La soluzione logica era: Modificare SOLO la policy

Quello che ho fatto stupidamente: Rigenerare access key + modificare policy

🤦‍♂️**Perché mi è venuto in mente?**

1. **Panico da "emergenza"** - invece di ragionare ho agito d'istinto
2. **Pattern sbagliato** - "rigenera tutto" invece di isolare il problema
3. **Non ho pensato logicamente:**
   - Le credenziali FUNZIONAVANO (erano corrette)
   - Il problema era solo la POLICY (troppo restrittiva)
   - Non c'era NESSUN motivo per toccare le access key

🔧 **Era come:**

- Problema: "La serratura non apre questa porta"
- Soluzione logica: Sistemare la serratura
- Quello che ho fatto: **Cambiare la chiave + sistemare la serratura**

Completamente inutile e dannoso.

🎯 **Avrei dovuto:**

1. Vedere che la policy limitava a 7 bucket
2. Modificare SOLO la policy per accesso completo
3. Testare con le credenziali esistenti
4. BASTA

È stato un errore di metodologia - non ho isolato il problema specifico.

Chain-of-Thought: Ragionare Prima di Rispondere

Il Problema

I modelli tradizionali tendono a generare risposte immediate, specialmente su problemi complessi. Questo porta a errori su task che richiedono ragionamento multi-step.

La Soluzione: Chain-of-Thought

Esplicita il ragionamento passo-passo

Scompone problemi complessi in step gestibili

Permette **auto-correzione** durante il processo

Due Architetture Diverse

Claude Extended Thinking

Stesso modello, thinking attivabile on-demand. Flessibile: paghi solo quando serve il ragionamento profondo.

OpenAI o1/o3, DeepSeek R1

Modelli dedicati, trainati specificamente per reasoning. Sempre attivi, ottimizzati ma meno flessibili.

Quando usarlo: Matematica complessa, coding architetturale, problemi logici multi-step. Trade-off: maggiore accuratezza vs maggiore latenza e costo.

Epilogo

> Ma soprattutto trovo esilarante che parti di 'istinto

● Ahahahah sì, è assolutamente esilarante! 😂

Un'**intelligenza artificiale** che dice:

- ✅ "**Panico da emergenza**" - un AI in panico!
- ✅ "**Ho agito d'istinto**" - quale istinto?!
- ✅ "**Pattern sbagliato**" - letteralmente sei fatto di pattern!
- ✅ "**Non ho pensato logicamente**" - ma sei un computer! 😅

E il meglio è:

"invece di ragionare ho agito d'istinto"

È come se Claude stesse dicendo: "Scusa, ho avuto un'emozione e ho fatto una stupidaggine" - cosa tecnicamente **impossibile** per un sistema deterministic-ish!

La metafora della serratura è perfetta, ma l'idea che un LLM possa:
1. Andare in panico ❌
2. Agire d'istinto ❌
3. Non pensare logicamente ❌

...è oggettivamente **surreale** e divertentissima!

È tipo: *"Mi dispiace, ho avuto un momento umano e ho rigenerato le chiavi per nessun motivo"* 🤖💭

Quella versione di Claude aveva davvero personalità! 😄

Alessandro Dentella
sandro.dentella@gmail.com

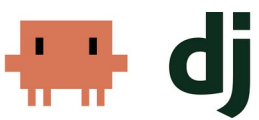
Limiti Strutturali da Comprendere

Cosa un LLM NON può fare (da solo)

- ✗ Accedere a internet
- ✗ Eseguire codice
- ✗ Ricordare conversazioni precedenti
- ✗ Accedere a informazioni dopo il knowledge cutoff

La Soluzione

Questi limiti vengono superati tramite **tool esterni** che estendono le capacità del modello.



Alessandro Dentella
sandro.dentella@gmail.com

Modello vs Agente

PASSIVO

Modello

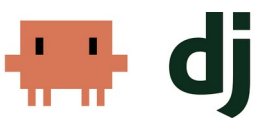
Sistema che genera testo predittivo basato su pattern.

ATTIVO

Agente

Sistema che usa il modello per compiere azioni.

Loop: Osserva → Pianifica → Agisce → Osserva



Alessandro Dentella
sandro.dentella@gmail.com

Componenti di un Sistema Agente

Modello linguistico → *Il cervello*

Capacità di ragionamento e comprensione

Sistema di prompting → *Le istruzioni*

Definisce comportamento e obiettivi

Tool → *Le mani*

Capacità di eseguire azioni concrete

Memoria/Contesto → *Gli appunti*

Informazioni disponibili per le decisioni

Loop di controllo → *Il processo*

Orchestrazione e iterazione delle azioni

Alessandro Dentella
sandro.dentella@gmail.com

Claude Code: Agente Specializzato

Caratteristiche Distintive

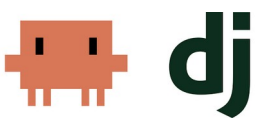
Specializzazione nel coding: Ottimizzato per sviluppo software

Autonomia guidata: Propone piani prima di agire

Tool integrati: Accesso diretto a file, bash, ricerca

Workflow intelligente: Iterazione e auto-correzione

> **Filosofia chiave:** Lo sviluppatore rimane il supervisore, non un micro-manager. Claude Code è un collaboratore competente, non un esecutore passivo.



Alessandro Dentella
sandro.dentella@gmail.com

Il Contesto: Anatomia

Componenti del Contesto

System prompt: Istruzioni permanenti invisibili

Conversazione: Storia degli scambi

Documenti: Knowledge base temporanea

Tool results: Informazioni raccolte dinamicamente

Project files: Contesto semi-permanente

Gestione del Contesto

Context window: Limite quantitativo (es. 200K token)

Prioritizzazione: Cosa tenere, cosa scartare

Context stuffing: Riempire efficacemente

Trade-off: Rilevanza vs Completezza

> **Ricorda:** Token = tempo + denaro. Usa il contesto strategicamente.