Profice

AI ACADEMY

Applicare l’Intelligenza Artificiale nello sviluppo software

AI ACADEMY

Hackaton: Rapid prototyping
27/06/2025

INTRODUZIONE DELL'ISTRUTTORE

Tamas Szakacs

Formazione

- Laureato come programmatore matematico
- MBA in management

Principali esperienze di lavoro

- Amministratore di sistemi UNIX
- Oracle DBA
- Sviluppatore di Java, Python e di Oracle PL/SQL
- Architetto (solution, enterprise, security, data)
- Ricercatore tecnologico e interdisciplinare di IA

Dedicato alla formazione continua

- Teorie, modelli, framework IA
- Ricerche IA
- Strategie aziendali
- Trasformazione digitale
- Formazione professionale

email: tamas.szakacs@proficegroup.it

MOTIVI E RIASSUNTO DEL CORSO

L'**Intelligenza Artificiale (AI)** è oggi il motore dell'innovazione in ogni settore, grazie alla sua capacità di analizzare dati, automatizzare processi e generare nuove soluzioni. Questo corso offre una panoramica completa e pratica sullo sviluppo di applicazioni AI moderne, guidando i partecipanti dall'ideazione al rilascio in produzione.

Attraverso una **combinazione di teoria chiara ed esercitazioni pratiche**, saranno affrontate le tecniche e gli strumenti più attuali: **machine learning, deep learning, reti neurali, Large Language Models (LLM), Transformers, Retrieval Augmented Generation (RAG)** e progettazione di agenti AI.
Le competenze acquisite saranno applicate in progetti concreti, dallo sviluppo di chatbot all'integrazione di modelli generativi, fino al deploy di soluzioni AI in ambienti reali e collaborativi.

Il percorso è pensato per chi vuole imparare a progettare, valutare e integrare sistemi AI di nuova generazione, con particolare attenzione alle best practice di programmazione, collaborazione in team, sicurezza, valutazione delle performance ed etica dell'AI.

DURATA: 17 GIORNI

OBIETTIVI

Il percorso formativo è progettato per **giovani consulenti junior**, con una conoscenza base di programmazione, che stanno iniziando un percorso professionale nel settore AI.

L’obiettivo centrale è fornire una panoramica pratica, completa e operativa sull’intelligenza artificiale moderna, guidando ogni partecipante attraverso tutte le fasi fondamentali.

OBIETTIVI

- Allineare conoscenze AI, ML, DL di tutti i partecipanti
- Saper usare e orchestrare modelli LLM (closed e open-weight)
- Costruire pipeline RAG complete (retrieval-augmented generation)
- Progettare agenti AI semplici con strumenti moderni (LangChain, tool calling)
- Capire principi di valutazione, robustezza e sicurezza dei sistemi GenA
- Migliorare la produttività come sviluppatori usando tool GenAI-driven
- Padroneggiare best practice di sviluppo, versioning e deploy AI
- Introdurre i fondamenti di Graph Data Science e Knowledge Graph
- Ottenere capacità di valutazione dei modelli e metriche
- Comprensione dell'etica e dei bias nei modelli di intelligenza artificiale
- Approfondire le normative di riferimento: AI Act, compliance e governance AI

Il corso è **estremamente pratico** (circa il 40% del tempo in esercitazioni hands-on, notebook, challenge e hackathon), con l'utilizzo di Google Colab, GitHub, e tutti gli strumenti necessari per lavorare su progetti reali e simulati.

STRUTTURA DELLE GIORNATE – PROGRAMMA BREVE

Tutte le giornate sono di 8 ore (9:00-17:00), con 1 ora di pausa suddivisa (mezz'ora pranzo, due pause da 15 min durante la mattina e il pomeriggio).

La progettazione sintetica delle giornate:

Giorno	Tema	Breve descrizione
1	Git & Python clean-code	Collaborazione su progetti reali, versionamento, codice pulito e testato
2	Machine Learning Supervised	Modelli supervisionati per predizione e classificazione
3	Machine Learning Unsupervised	Clustering, riduzione dimensionale, scoperta di pattern
4	Prompt Engineering avanzato	Scrivere e valutare prompt efficaci per modelli generativi
5	LLM via API (multi-vendor)	Uso pratico di modelli LLM via API, autenticazione, deployment
6	Come costruire un RAG	Pipeline end-to-end per Retrieval-Augmented Generation
7	Tool-calling & Agent design	Progettare agenti AI che usano strumenti esterni
8	Hackathon: Agentic RAG	Challenge pratica: chatbot agentico RAG in team

STRUTTURA DELLE GIORNATE – PROGRAMMA BREVE

Tutte le giornate sono di 8 ore (9:00-17:00), con 1 ora di pausa suddivisa (mezz'ora pranzo, due pause da 15 min durante la mattina e il pomeriggio).

La progettazione sintetica delle giornate:

Giorno	Tema	Breve descrizione
9	Hackathon: Rapid Prototyping	Da prototipo a web-app con Streamlit e GitHub
10	AI Productivity Tools	Workflow con IDE AI-powered, automazione e refactoring assistito
11	Docker & HF Spaces Deploy	Deployment di app GenAI containerizzate o su HuggingFace Spaces
12	AI Act & ISO 42001 Compliance	Fondamenti di compliance e governance AI
13	Knowledge Base & Graph Data Science	Introduzione a Knowledge Graph e query con Neo4j
14	Model evaluation & osservabilità	Metriche avanzate, explainability, strumenti di valutazione
15	AI bias, fairness ed etica applicata	Analisi dei rischi, metriche e mitigazione dei bias
16-17	Project Work & Challenge finale	Lavoro a gruppi, POC/POD, presentazione e votazione progetti

METODOLOGIA DEL CORSO

1. Approccio introduttivo ma avanzato

Il corso è introduttivo nei concetti base dell'AI applicata allo sviluppo, ma affronta anche tecnologie, modelli e soluzioni avanzate per garantire un apprendimento completo.

2. Linguaggio adattato

Il linguaggio utilizzato è chiaro e adattato agli studenti, con spiegazioni dettagliate dei termini tecnici per favorirne la comprensione e l'apprendimento graduale.

3. Esercizi pratici

Gli esercizi pratici sono interamente svolti online tramite piattaforme come Google Colab o notebook Python, eliminando la necessità di installare software sul proprio computer.

4. Supporto interattivo

È possibile porre domande in qualsiasi momento durante le lezioni o successivamente via email per garantire una piena comprensione del materiale trattato.

NOTA

Il corso segue un **approccio laboratoriale**: ogni giornata combina sessioni teoriche chiare e concrete con molte attività pratiche supervisionate, per sviluppare *competenze reali* immediatamente applicabili.

I partecipanti lavoreranno spesso in gruppo, useranno notebook in Colab e versioneranno codice su GitHub, vivendo una vera simulazione del lavoro in azienda AI.

Nessun prerequisito avanzato richiesto: si partirà dagli strumenti e flussi fondamentali, con una crescita graduale verso le tecniche più attuali e richieste dal mercato.

ORARIO TIPICO DELLE GIORNATE

Orario	Attività	Dettaglio
09:00 – 09:30	Teoria introduttiva	Concetti chiave, schema della giornata
09:30 – 10:30	Live coding + esercizio guidato	Esempio pratico, notebook Colab
10:30 – 10:45	*Pausa breve*	
10:45 – 11:30	Approfondimento teorico	Tecniche, best practice
11:30 – 12:30	Esercizio hands-on individuale	Sviluppo o completamento di codice
12:30 – 13:00	Discussione soluzioni + Q&A	Condivisione e correzione
13:00 – 14:00	*Pausa pranzo*	
13:30 – 14:15	Teoria avanzata / nuovi tools	Nuovi strumenti, pattern, demo
14:15 – 15:30	Esercizio a gruppi / challenge	Lavoro di squadra su task reale
15:30 – 15:45	*Pausa breve*	
15:45 – 16:30	Sommario teorico e pratico	
16:30 – 17:00	Discussioni, feedback	Riepilogo, best practice, domande aperte

DOMANDE?

Cominciamo!

OBIETTIVI DELLA GIORNATA

Obiettivi della giornata

- Comprendere i principi del rapid prototyping e il loro valore nello sviluppo di soluzioni AI.
- Scoprire cos’è Streamlit e quali sono i suoi punti di forza per creare app prototipali con Python.
- Imparare a trasformare un progetto Python (es. chatbot RAG-agentico) in un’applicazione web interattiva.
- Conoscere gli elementi base di Streamlit (layout, componenti di input/output, visualizzazione dati).
- Integrare modelli AI e pipeline in Streamlit per test rapidi e demo accessibili.
- Realizzare una semplice demo funzionante e pronta da condividere.
- Analizzare vantaggi e limiti di Streamlit rispetto ad altri strumenti di prototipazione.

ANALISI DOCUMENTALE E PROTEZIONE DATI

SmartDocs Srl – Analisi documentale e protezione dati

Scenario:
SmartDocs Srl, media azienda europea, deve gestire email e documenti contenenti dati sensibili di clienti (es: IBAN, codice fiscale, indirizzi, nomi, numeri di telefono).

L’azienda vuole automatizzare:

- Estrazione di entità e dati sensibili (NER, pattern matching)
- Riepilogo automatico e risposta alle richieste clienti
- Tutelare la privacy (alcuni dati NON devono mai lasciare il server locale)
- Minimizzare i costi cloud e garantire risposte rapide

ANALISI DOCUMENTALE E PROTEZIONE DATI

Useremo due modelli LLM per la soluzione aziendale

Architettura semplificata

1. **Modello locale open source (es: TinyLLaMA)**
    1. Viene eseguito localmente, direttamente sul vostro computer o su server aziendali.
    2. Si occupa delle operazioni più “sensibili”, come l’estrazione di dati personali e la protezione della privacy (Named Entity Recognition e masking dei dati).
    3. È veloce, economico, e mantiene i dati riservati all’interno dell’azienda.

2. **Modello cloud avanzato (es: Azure OpenAI GPT-3.5/4)**
    1. Viene utilizzato tramite API esterne, in cloud.
    2. Si occupa di attività più complesse come il riepilogo automatico, l’analisi semantica e la generazione di risposte ai clienti.
    3. Permette di gestire testi lunghi e offre maggiore potenza di calcolo e qualità delle risposte.

ANALISI DOCUMENTALE E PROTEZIONE DATI

L’obiettivo:

Sfruttare **i punti di forza di entrambi i modelli**:

- La privacy e la velocità del modello locale
- La potenza e la flessibilità del modello cloud

Garantire che **i dati più delicati non escano dall’azienda**, mentre sfruttiamo le migliori tecnologie disponibili per la produttività e l’automazione.

Distribuzione dei lavori

- Tutti lavorano sullo **stesso problema reale** ma con strumenti diversi.

Lavoro locale autonomo

- privacy, sicurezza, regex/NER, masking.

Lavoro cloud con l’aiuto dell’istruttore

- prompt, parametri, API, gestione delle risposte.

Altri componenti futuri per i giorni successivi

- RAG, contestualizzazione, configurazione, deployment ecc.
- La **collaborazione** tra i modelli con una architettura AI.

RUOLI DEI DUE MODELLI

1. Modello locale (TinyLLaMA o simili) — “Difensore/controllore”
Viene **eseguito localmente.**
Obiettivi:

- **NER (Named Entity Recognition)**: trova nomi, indirizzi, IBAN, codice fiscale, numeri, ecc.
- **Pattern Detection**: segnala se sono presenti dati sensibili o “red flag”.
- **RAG** (Retrieval Augmented Generation) opzionale: usa una knowledge base aziendale locale per arricchire le risposte.
- **Controlla l’output** prima che venga inviato al modello cloud, mascherando o anonimizzando i dati sensibili (es. sostituendo “Mario Rossi” con “[NOME]”).

2. Modello cloud (Azure GPT-3.5, GPT-4, ecc) — “Analista/colloquio”
Viene usato tramite API cloud, guidato passo-passo.
Obiettivi:

- Riceve documenti **già “ripuliti”** o parzialmente anonimizzati dal modello locale.
- Esegue:
  - Riepilogo (summary)
  - Analisi semantica
  - Generazione di risposte per i clienti
- Gestisce solo dati che non violano la privacy.

ESEMPIO DI PIPELINE COLLABORATIVA

Input:
L'utente carica un documento/email.

Passo 1 (locale):
Il modello locale fa NER, segnala dati sensibili e li anonimizza (es: "Mario Rossi" → "[NOME]", "IT60X0542811101000000123456" → "[IBAN]").

Passo 2 (controllo):
L'output ripulito viene controllato/validato (gli esperti possono anche vedere che i dati sono davvero rimossi).

Passo 3 (cloud):
Il testo anonimizzato viene inviato all'API Azure che fa il riepilogo, classifica la richiesta e prepara una risposta.

Passo 4 (output):
L'output finale può essere ricomposto localmente, reinserendo alcune entità dove permesso, oppure consegnato così.

RUOLI DI PROGETTO NEL TEAM AGENTIC RAG CHATBOT

Introduzione
Per lavorare in modo efficace su un progetto complesso come lo sviluppo di un chatbot agentico basato su RAG, è fondamentale definire chiaramente i ruoli all'interno del team. Ogni ruolo contribuisce con competenze specifiche per garantire qualità, sicurezza, organizzazione e successo della soluzione.

Principali ruoli di progetto:

Project Manager (PM)
Coordina le attività, pianifica tempi e risorse, facilita la comunicazione e monitora l'avanzamento del progetto.

Business Analyst (BA)
Raccoglie i requisiti del cliente, definisce il perimetro funzionale, traduce i bisogni in specifiche tecniche e funzionali.

Solution Architect
Progetta l'architettura generale del sistema, definisce le scelte tecnologiche e l'integrazione tra le componenti (LLM, RAG, agenti, API, interfacce).

RUOLI DI PROGETTO NEL TEAM AGENTIC RAG CHATBOT

Introduzione
Per lavorare in modo efficace su un progetto complesso come lo sviluppo di un chatbot agentico basato su RAG, è fondamentale definire chiaramente i ruoli all'interno del team. Ogni ruolo contribuisce con competenze specifiche per garantire qualità, sicurezza, organizzazione e successo della soluzione.

Principali ruoli di progetto:

Developer (Frontend/Backend)
Implementa le funzionalità del chatbot, gestisce l'integrazione tra moduli, scrive codice e risolve bug.

Prompt Engineer
Progetta, ottimizza e testa i prompt per LLM e agenti, curando l'efficacia e la sicurezza delle interazioni AI.

Security Specialist
Analizza rischi, implementa misure di sicurezza (es. controllo accessi, protezione dati, guardrail su input/output).

Tester / QA
Esegue test funzionali, di integrazione e sicurezza, valida i risultati, individua errori e propone correzioni.

DELIVERABLES E DOCUMENTI MINIMI PER IL PROGETTO

Per consegnare i lavori bisogna preparare questi deliverable:

- **Risposta progettuale**
  Breve descrizione di come il team ha risolto il problema proposto, con riferimento alle scelte principali.
- **Schema architetturale**
  Schema essenziale (testuale o diagramma semplice) dell'architettura della soluzione e dei principali componenti (modello, agenti, pipeline, API, ecc.).
- **Codice sorgente**
  Tutto il codice sviluppato, con commenti chiari e autoesplicativi (documentazione del codice generata direttamente dai commenti).
- **Dataset di test**
  File di dati usati per le prove e le demo, rappresentativi dei casi d'uso.
- **Metodo di test**
  Breve descrizione del metodo di test applicato: quali casi, quali dati, come è stato valutato il funzionamento.
- **(Opzionale – da aggiungere dopo lezione su etica/EU AI Act)**
  Eventuali note su conformità etica, privacy e regole AI.

INTRODUZIONE A RAPID PROTOTYPING

Il rapid prototyping (“prototipazione rapida”) è un approccio che mira a realizzare rapidamente versioni funzionanti e semplificate di un’applicazione o servizio, così da testare idee, flussi di lavoro e funzionalità prima di un vero sviluppo su larga scala.
Permette di individuare punti di forza, problemi e bisogni reali degli utenti, accelerando il ciclo feedback e riducendo tempi e costi di sviluppo.

Modi di fare Rapid Prototyping

- **Script interattivi**
  Uso di notebook (Jupyter, Colab) o script Python con input/output semplici per testare algoritmi o modelli.
- **App prototipo con Streamlit**
  Trasformare rapidamente codice in app web con interfaccia minimale e controlli interattivi, senza occuparsi di frontend complessi.
- **Wireframe e mockup**
  Creazione di interfacce grafiche statiche o semplificate (con strumenti come Figma, ecc.) per mostrare la logica e il layout senza codice.
- **Tool low-code/no-code**
  Utilizzo di piattaforme (es. Bubble, PowerApps) per costruire prototipi funzionanti trascinando elementi e logiche, con pochissimo codice.
- **Demo video o walkthrough**
  Simulazione del flusso utente con video o sequenze di immagini, utile per spiegare concept e ricevere feedback.

INTRODUZIONE A STREAMLIT

Streamlit è un framework open-source per Python che permette di trasformare facilmente script e prototipi in semplici applicazioni web interattive, senza richiedere esperienza in sviluppo frontend.
Il suo obiettivo è accelerare la fase di rapid prototyping, ovvero la creazione veloce di versioni funzionanti di nuove idee o strumenti, per testare, condividere e ricevere feedback in tempi rapidi.

Piattaforma e modalità d'uso

Streamlit si esegue direttamente su un computer con Python installato (Windows, macOS, Linux) e permette di lanciare app locali che possono essere condivise su browser tramite link (localhost o in cloud, con Streamlit Cloud).
Non è necessario installare server web separati: basta salvare uno `script.py` e lanciare `streamlit run script.py` dal terminale per vedere l'app funzionante.

Trasformare uno script Python in un'app Streamlit

Il rapid prototyping in Streamlit significa aggiungere poche righe di codice a uno script Python per creare pulsanti, caselle di testo, upload di file e grafici interattivi.
Elementi di input e visualizzazione si inseriscono facilmente tramite funzioni come `st.button`, `st.text_input`, `st.file_uploader`, `st.write`, ecc.
Il risultato è un'applicazione pronta per essere usata dagli utenti o dal team, senza la complessità di web framework tradizionali.

INTRODUZIONE A STREAMLIT

Widget in Streamlit?

I **widget** sono gli elementi interattivi che trasformano un semplice script Python in un'applicazione dinamica e user-friendly.
Attraverso i widget, gli utenti possono inserire dati, scegliere opzioni, caricare file, avviare operazioni e visualizzare risultati in tempo reale.
Ogni widget si aggiunge con una sola riga di codice, rendendo possibile la creazione di interfacce ricche senza dover scrivere codice HTML, CSS o JavaScript.

Esempi di widget:

- Caselle di testo per inserire input
- Pulsanti per avviare azioni
- Menu a tendina, slider, check-box, sidebar
- Upload e download di file
- Selettori di data, immagini, progress bar

I widget sono la base per il **rapid prototyping**: permettono di testare idee e funzioni con pochi click, raccogliendo subito feedback dagli utenti.

INTRODUZIONE A STREAMLIT

Elementi principali di Streamlit

- **Casella di testo (`st.text_input`)**
  Permette all'utente di inserire dati o domande; utile per parametri o prompt.
- **Pulsanti (`st.button, st.checkbox, st.radio`)**
  Consentono interazione immediata e controllo del flusso dell'app.
- **Sidebar (`st.sidebar`)**
  Area laterale dedicata per raccogliere controlli, menu o opzioni aggiuntive, mantenendo l'interfaccia pulita.
- **Upload e gestione file (`st.file_uploader`)**
  Facilita il caricamento di dati o documenti, integrando direttamente input utente e analisi.
- **File Download (`st.download_button`)**
  Permette agli utenti di scaricare file generati dall'app, come risultati, report o dati elaborati.
- **Visualizzazione dati (`st.write, st.dataframe, st.chart, st.map`)**
  Mostra tabelle, grafici, risultati di modelli AI o mappe geografiche in modo chiaro e dinamico.
- **Slider (`st.slider`)**
  Consente di regolare valori numerici (es. intervallo, soglie) in modo intuitivo, muovendo un cursore. Ideale per impostare parametri dinamici.

INTRODUZIONE A STREAMLIT

Elementi principali di Streamlit

- **Selectbox (`st.selectbox`)**
  Permette di creare un menu a tendina da cui l'utente può scegliere una tra diverse opzioni predefinite. Utile per selezionare categorie, modelli o parametri.
- **Number Input (`st.number_input`)**
  Campo per inserire numeri precisi, utile per valori quantitativi come quantità, prezzi, limiti o altri parametri numerici.
- **Date Input (`st.date_input`)**
  Widget per selezionare rapidamente una data dal calendario, adatto per pianificazioni o filtri temporali.
- **Image (`st.image`)**
  Visualizza immagini statiche o dinamiche direttamente nell'applicazione, utile per esempi visivi, grafici o output di modelli AI.
- **Progress Bar (`st.progress`)**
  Mostra l'avanzamento di un'operazione lunga (es. caricamento dati, inferenza modello), rendendo l'esperienza più trasparente.

SCOPO DEL PROGETTO

Creare il chatbot con agentic RAG usando rapid prototyping con Streamlit

Capacità o caratteristiche richieste

- ☐ Anonimizzazione dei dati sensitivi (min. nomi e IBAN) usando NER e regex
- ☐ Gestione di documenti con tecniche RAG nei prompt
- ☐ Uso di embedding, chunking, vettorizzazione dei documenti e prompt (similarity search su documenti)
- ☐ Autonomia / agente per gestire files e risposte (con creazione di files, per esempio risposta mail in un file)
- ☐ Chat e gestione di una singola sessione con la sequenza dei messaggi con Langchain

Visualizzazioni richieste con Streamlit

- ☐ Chat input e output in formato history (tipo ChatGPT)
- ☐ Bottone per caricare documenti (da gestire con autonomia dal modello)
- ☐ Elenco dei file caricati
- ☐ (Opzionale) Gestione di diverse sessioni
- ☐ (Opzionale) Gestione di memoria, a disposizione di ogni sessione

DOMANDE?

Lavoro in gruppi

DOMANDE?

PAUSA

GRAZIE PER L'ATTENZIONE