

1 Prima di iniziare

1.1 L^AT_EX: vita morte e miracoli

Prima di inizia poniamo delle premesse, L^AT_EX è un linguaggio di marcatura il che vuol dire che nasce col fine di formattare testo. Nasce nel 1994 e si evolve in una miriade di sottolinguaggi come il K_AT_EX di notion (quello dei blocchi matematici) che verticalizza e isola l'uso di L^AT_EX rendendolo una comoda implementazione per scrivere equazioni. Un'altra evoluzione seguita da L^AT_EX è stata quella di evolvere i propri compilatori¹, li vedremo più nello specifico in futuro ma vi basti sapere che ho scelto di usare XeL^AT_EX che è un implementazione più nuova e ottimizzata del compilatore base denominato PdfL^AT_EX.

1.2 Ai

Una premessa anche sull'uso dell'intelligenza artificiale è opportuna. Gemini, in quanto probabilmente sarà la vostra scelta, nella sua modalità pro in particolare, è brava a capire come usare piccoli e singoli blocchi di L^AT_EX, tipo "come si crea una tabella su L^AT_EX?", "come creo una lista?", "come aggiungo un'immagine". Più informazioni passiamo a Gemini, più questa risposta sarà precisa. Un'accortezza che dovrete avere o quantomeno un punto da tener in considerazione nell'uso di gemini è lo specificare le librerie che volete usi o informarmi anche conversando con Gemini e prendendo le sue informazioni con le pinze di quali siano le librerie compatibili con la compilazione che abbiamo scelto. Attenzione però che Gemini usato senza

¹Un compilatore è un programma che si occupa di tradurre da codice a programma finito, nel nostro caso da .tex a pdf

1 Prima di iniziare

controllo sbaglia, sbaglia spesso e porta a peggiorare gli errori commessi consigliando di apportare diverse modifiche, magari giuste in generale ma non per il vostro progetto. Il mio consiglio è di dare due possibilità a Gemini, se due modifiche di Gemini non sistemano il codice allora ritornate indietro a prima che Gemini ci mettesse mano. Un'altra alternativa sarebbe usare Claude valgono pur sempre le avvertenze ma è molto migliore di Gemini. Claude però ha un piano base con limiti piuttosto stringenti quindi usatelo "in caso di emergenza". Un uso utile di un'intelligenza artificiale è quando il vostro lavoro non compila e come errore vi dà un log immenso da leggere, copia incollatelo interamente su un ai e cercate di vedere che spiegazione vi dà, valutatene la correttezza delle sue affermazioni e correggete.

1.3 Dove scrivere in L^AT_EX

L'ambiente in cui lavorare è alla fine una vostra scelta, posso al più darvi dei consigli e sconsigliarvi altri metodi.

1.3.1 Overleaf

Il tool web più usato si chiama Overleaf, è una piattaforma comoda da usare ma ha un piano base molto limitato che generalmente vieta di lavorare su grandi file o su file che contengono tante strutture più complesse del semplice testo. Altro punto negativo è che essendo un sito web si può accedere ai file solo se connessi a una rete wifi.

1.3.2 Overleaf Toolkit

È la versione selfhosted² di Overleaf. Permette di alzare i limiti ma essendo il mio server non

²Il termine selfhosted vuol dire che si può installare localmente su un server o computer, nel mio caso userò selfhosted per indicare i programmi che posso scaricare sul mio server e rendere a voi accessibili

particolarmente potente non so che carico potrà effettivamente sostenere. Il carico inoltre scala in funzione di quanti di voi decideranno di utilizzarlo.

1.3.3 Download locale

Il download locale di $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ necessita di 2 componenti:

- Ambiente di sviluppo
- Compilatore locale o remoto

Ambiente di sviluppo

Visual Studio Code L'ambiente di sviluppo locale che consiglio è **Visual Studio Code**. Mi rendo conto possa essere un troppo per quel poco di $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ che dobbiamo fare in quanto nasce come ambiente per la programmazione ma proprio per questo risulta essere il più completo editor di testo che puoi desiderare. Possiede una miriade di estensioni installabili per migliorare il suo uso e permette di visualizzare nella stessa applicazione sia il codice che il pdf finito. **Visual Studio Code** possiede una funzione denominata **IntelliSense** che suggerisce la completazione automatica del codice (non del testo) che stiamo scrivendo in funzione di ciò che abbiamo scritto. **IntelliSense** non è basata su intelligenza artificiale.

TeXStudio L'opzione più vecchia è altrimenti usare **TeXStudio** che nativamente supporta $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ e che forse possiede un compilatore integrato. Non l'ho mai usato. Sarà sicuramente comodo quanto vecchio. Lo penso perchè non sono nati molti altri programmi del genere.

Compilatore locale

Windows Per quanto riguarda **Windows** abbiamo due possibilità: **TeX Live** e **MiKTeX**.

1 Prima di iniziare

TeX Live **TeX Live** installa la libreria completa di **LaTeX** con tutte librerie e i compilatori ad esso associate . Si parla di 1.6GB ma il problema non è tanto il peso quanto il fatto di essere distribuito da un server vecchio e che il download non è centralizzato, ovvero ogni singolo file ha una posizione differente, che rallenta maggiormente il download di questi file. A me ha impiegato circa 6 ore. È un download che non dovrebbe essere interrotto.

MiKTeX **MiKTeX** è l'alternativa più veloce e più nuova, installa una minima parte delle librerie e dei compilatori necessari, necessità del download di una libreria in più denominata Pearl (un linguaggio di programmazione) per poter essere usata con l'estensione di **Visual Studio Code** per **LATEX**. **MiKTeX** scarica le ulteriori librerie solo quando compiliamo un progetto che le usa.

MacOsX L'opzione per **MacOsX** è una sola: **MacTeX**. Non l'ho mai usata.

Compilatore remoto

Questa è una soluzione che sto testando con i lavori degli ultimi giorni. Consiste nell'usare git insieme a Gitea (versione selfhosted del più famoso Github). Git gestisce il `push` (upload) e `pull` (download) dei progetti in una cartella remota definita come `repository`. Possiede una funzione detta `Action` che permette di eseguire in modo automatico su un `docker`³ remoto delle azioni di compilazione e aggiungere al nostro upload il risultato finito. Gitea essendo selfhosted è priva di code a meno di più richieste simultanee e generalmente risolve la compilazione entro i due minuti. È possibile scaricare il pdf in locale effettuando una `pull`. Altro vantaggio di git il quale lo rende estremamente usato sia dai programmatori che più in generale per qualsiasi progetto di gruppo è la sua comoda gestione del lavoro simultaneo di

³Un particolare programma di macchine virtuali

1.3 Dove scrivere in $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$

più persone contemporaneamente anche sullo stesso file.