

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

---

SCUOLA DI SCIENZE  
Corso di Laurea in Informatica

**USO DI LLM  
PER LA REALIZZAZIONE DI SITI WEB:  
UN CONFRONTO**

**Relatore:**  
Chiar.mo Prof.  
FABIO VITALI

**Presentata da:**  
SIMONE BORTOLAI

**Correlatrice:**  
Chiar.ma Dott.ssa.  
ALESSANDRA  
STRAMIGLIO

**III Sessione  
Anno Accademico 2022/2023**



*“Buongiorno... e casomai non vi rivedessi, buon pomeriggio,  
buonasera e buonanotte!”*

Truman Burbank



# Abstract

Questo studio si concentra sull'analisi comparativa delle prestazioni di tre modelli di intelligenza artificiale avanzata, GPT-3.5, GPT-4 e Gemini, nell'ambito dello sviluppo web full stack.

Attraverso l'implementazione pratica di un'applicazione web per la gestione di note, comprensiva di frontend, backend e integrazione con database, abbiamo valutato la capacità di ciascun modello di generare codice funzionale, l'efficacia nell'implementazione di stili grafici e l'usabilità del prodotto finito.

L'obiettivo era determinare il livello di autonomia che questi Large Language Models (LLM) possono raggiungere nello sviluppo di applicazioni web e identificare le loro principali limitazioni.

La ricerca evidenzia che, nonostante i progressi nell'IA, persistono sfide significative legate alla coerenza del codice, alla gestione del contesto e all'integrazione di aspetti visivi e di usabilità, sottolineando la necessità di ulteriori miglioramenti per realizzare lo sviluppo web completamente automatizzato.



# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Contesto . . . . .	1
1.2	Obbiettivi . . . . .	2
1.3	Struttura del documento . . . . .	4
<b>2</b>	<b>Nozioni di base</b>	<b>7</b>
2.1	Reti Neurali . . . . .	7
2.2	Transformer . . . . .	8
2.3	Large Language Model (LLM) . . . . .	11
2.4	GPT (OpenAI) . . . . .	12
2.5	Gemini (Google) . . . . .	14
2.6	Prompt Engineering . . . . .	16
2.7	Letteratura Esistente . . . . .	17
<b>3</b>	<b>Definizione del progetto</b>	<b>19</b>
3.1	Descrizione dell'applicativo . . . . .	19
3.2	Descrizione dell'ambiente di deploy . . . . .	21
<b>4</b>	<b>Sviluppo dell'applicativo</b>	<b>23</b>
4.1	Prompt di partenza . . . . .	23
4.2	GPT-3.5 . . . . .	24
4.2.1	Prompting . . . . .	24
4.2.2	Commenti . . . . .	26
4.3	GPT-4 . . . . .	26
4.3.1	Prompting . . . . .	26

4.3.2	Nessun input visivo . . . . .	39
4.3.3	Input visivi . . . . .	41
4.3.4	Input visivi con modifiche umane . . . . .	43
4.3.5	Commenti . . . . .	45
4.4	Gemini . . . . .	45
4.4.1	Prompting . . . . .	45
4.4.2	Commenti . . . . .	47
<b>5</b>	<b>Valutazioni</b>	<b>53</b>
5.1	Correttezza . . . . .	53
5.2	Sofisticazione grafica . . . . .	54
5.3	Usabilità . . . . .	54
5.4	Tabella riassuntiva . . . . .	56
5.5	Eventuali limiti dei modelli . . . . .	56
<b>6</b>	<b>Conclusione</b>	<b>59</b>
	<b>Bibliografia</b>	<b>64</b>



# Elenco delle figure

2.1	Il modello Transformer . . . . .	9
4.1	Home page realizzata da GPT-4 . . . . .	39
4.2	Login page realizzata da GPT-4 . . . . .	40
4.3	Form di creazione della nota realizzato da GPT-4 . . . . .	40
4.4	Home page (con stile) realizzata da GPT-4 . . . . .	41
4.5	Login page (con stile) realizzata da GPT-4 . . . . .	42
4.6	Form di creazione della nota (con stile) realizzato da GPT-4 . . . . .	42
4.7	Layout modificato a mano . . . . .	43
4.8	Immagine generata da GPT-4 per rimpiazzare l'icona . . . . .	44
4.9	Layout modificato da GPT-4 su prompt . . . . .	44
4.10	Errore di Gemini nella generazione del codice . . . . .	48
4.11	Risposta di Gemini multilingua al prompt "E' necessario fornire un Dockerfile unico per backend e frontend" . . . . .	49
4.12	Gemini fornisce codice incompleto al prompt "Scrivi il codice per la pagina NotePage.jsx" . . . . .	50
4.13	Home page realizzata da Gemini . . . . .	51



# Capitolo 1

## Introduzione

### 1.1 Contesto

L'intelligenza artificiale si trova oggi in una fase di crescita e di innovazione senza precedenti, rappresentando un pilastro fondamentale nell'evoluzione tecnologica e nel modo in cui interagiamo con il mondo digitale. In particolare, i Large Language Models (LLM) come GPT e Gemini stanno ridefinendo i limiti dell'interazione uomo-macchina. Questi modelli avanzati hanno il potenziale di trasformare radicalmente una vasta gamma di settori, dalla produzione di contenuti digitali alla programmazione, dall'assistenza clienti all'analisi dei dati, e oltre.

La capacità degli LLM di comprendere e generare testo naturale con un alto grado di coerenza e rilevanza apre nuovi orizzonti per lo sviluppo di applicazioni web full stack. Lo sviluppo di tali applicazioni, tradizionalmente un processo che richiede tempo e competenze specifiche in diverse aree della programmazione, potrebbe essere notevolmente accelerato e semplificato attraverso l'uso di questi modelli linguistici. La possibilità di generare rapidamente prototipi di siti web, esplorare diverse soluzioni di design e implementare funzionalità complesse senza la necessità di scrivere manualmente ogni singola riga di codice permetterebbe di ottimizzare l'approccio allo sviluppo software.

## 1.2 Obiettivi

L'obiettivo è di valutare in che modo l'automazione del codice, mediata da questi modelli di intelligenza artificiale, possa incidere sulla velocità e sull'efficacia del processo di sviluppo software. La capacità degli LLM di generare un'ampia varietà di prototipi in tempi ridotti offre una nuova prospettiva sulla fase di progettazione, permettendo agli sviluppatori di testare e valutare diverse soluzioni con un impegno temporale significativamente inferiore rispetto ai metodi tradizionali. Questo approccio apre la strada a un'esplorazione più ampia delle potenzialità progettuali, con il potenziale di identificare configurazioni ottimali che meglio rispondono alle necessità degli utenti.

Parallelamente, ci si chiede come l'adozione di LLM influenzi le competenze richieste agli sviluppatori, in particolare riguardo alla capacità di interagire efficacemente con questi modelli. Questa competenza diventa essenziale per guidare gli LLM verso la produzione di risultati specifici e di alta qualità, sottolineando l'importanza di una formazione adeguata e di un aggiornamento continuo delle abilità tecniche. Conoscere le tecniche più efficienti non solo migliora l'interazione con gli LLM, ma contribuisce anche a una maggiore comprensione di come questi strumenti possano essere sfruttati per massimizzare l'efficienza dello sviluppo software.

Queste motivazioni delineano un ambito di ricerca focalizzato sull'impiego degli LLM come strumenti per innovare e ottimizzare il processo di sviluppo delle applicazioni web. Determinare se e come l'uso di questi modelli possa rappresentare un approccio valido e sostenibile, in grado di offrire vantaggi concreti in termini di velocità di sviluppo, qualità del codice generato, e capacità di rispondere efficacemente alle esigenze progettuali è essenziale. In questo contesto, questa ricerca mira a fornire un contributo iniziale alla comprensione delle potenzialità e delle sfide associate all'adozione degli LLM nello sviluppo di software, con un'attenzione particolare alle implicazioni per la pratica professionale e alla formazione degli sviluppatori.

Di conseguenza, è logico sollevare alcuni interrogativi critici che meritano un'analisi dettagliata per comprendere appieno il loro potenziale e le sfide che presentano.

**Frontend Visivamente Coerente:** Il design del frontend di un'applicazione web è cruciale per attirare e mantenere l'interesse degli utenti. Si deve dimostrare la capacità di generare interfacce utente che non solo siano esteticamente gradevoli ma anche coerenti con l'identità visiva dell'applicazione. Questo include la scelta di schemi di colori, tipografie, layout di pagina e componenti di interfaccia utente che siano armoniosi e navigabili. La sfida sta nel garantire che il modello possa adattarsi a varie esigenze di design senza sacrificare nessun aspetto.

**Backend Robusto e Funzionale:** La logica funzionale e la gestione dei dati sono al cuore di qualsiasi applicazione web. Il modello deve essere capace di progettare e implementare un backend che gestisca efficacemente le richieste degli utenti, l'elaborazione dei dati e l'interazione con il database. Ciò richiede la generazione di codice che sia non solo funzionale ma anche ottimizzato, sicuro e scalabile. Il modello deve dimostrare di poter affrontare questioni complesse come l'autenticazione degli utenti, la gestione delle sessioni e la crittografia dei dati.

**Integrazione con il Database:** La scelta del database e la progettazione dello schema sono elementi fondamentali che influenzano le prestazioni e la scalabilità dell'applicazione. Il modello deve essere in grado di integrare il backend con un database in modo che i dati siano memorizzati, recuperati, aggiornati ed eliminati efficacemente. La sfida include la progettazione di schemi di database che siano logici, efficienti e in grado di supportare le operazioni richieste dall'applicazione.

**Usabilità ed Esperienza Utente:** Un'applicazione deve essere intuitiva e facile da usare per gli utenti finali. Ciò comporta la creazione di percorsi utente chiari, la minimizzazione della complessità dell'interfaccia e l'assicurazione che l'applicazione sia reattiva e performante. Si deve dimostrare la capacità di prevedere le esigenze degli utenti e di progettare un'esperienza utente che soddisfi queste aspettative, migliorando l'engagement e la soddisfazione dell'utente.

**Accessibilità:** L'accessibilità è fondamentale per garantire che l'applicazione possa essere utilizzata da persone con diverse capacità. Questo include la conformità agli standard WCAG (Web Content Accessibility Guidelines [1]), l'implementazione di

funzionalità di navigazione tramite tastiera, il supporto per lettori di schermo e l'adattamento dei contenuti per coloro che hanno difficoltà visive o uditive. La sfida per il modello è quella di generare codice e design che rispettino questi principi senza compromettere la funzionalità o l'estetica.

**Deployment:** E' necessario poter deployare l'applicazione, pertanto un LLM deve essere in grado di produrre codice che sia deployabile e che possa essere aggiornato o modificato facilmente senza interruzioni significative del servizio.

Per indagare l'efficacia degli LLM nello sviluppo autonomo di applicazioni web full stack, la tesi adotta un approccio metodico centrato sul Prompt Engineering. Questo metodo si basa sulla creazione di prompt dettagliati, progettati per orientare gli LLM verso la produzione di componenti software che soddisfino i requisiti specifici di un progetto.

## 1.3 Struttura del documento

Nel Paragrafo 3.1, viene fornita una descrizione dettagliata dell'applicativo, definendo il contesto e gli obiettivi specifici del progetto. Questo capitolo funge da base per comprendere le motivazioni dietro la scelta del design e le funzionalità richieste per l'applicazione web.

Successivamente, nel Paragrafo 4.1, viene illustrata la formulazione del prompt iniziale, che stabilisce gli obiettivi del progetto, le specifiche tecnologiche, i vincoli dell'ambiente di deploy e le aspettative relative a funzionalità, usabilità e accessibilità.

La fase sperimentale vera e propria inizia con l'analisi dei tentativi di generazione del sito, seguendo i dettagli forniti dal prompt iniziale. I Paragrafi 4.2, 4.3 e 4.4 documentano, rispettivamente, gli esperimenti condotti utilizzando GPT-3.5, GPT-4 e Gemini. Ogni capitolo esplora la capacità di ciascun modello di affrontare le fasi di sviluppo, dalla preparazione dell'ambiente, inclusa la strutturazione delle directory di progetto e l'installazione delle dipendenze, fino all'implementazione del codice per il frontend scelto dal modello, il backend in Node.js e l'integrazione con MongoDB.

Particolare attenzione viene posta alla valutazione della coerenza e dell'efficacia con cui ogni modello gestisce i compiti assegnati, mettendo in luce come questi si conformano ai vincoli tecnici forniti e agli standard attuali. Attraverso un'analisi critica delle funzionalità, della conformità tecnica, dell'esperienza utente, dell'accessibilità e della scalabilità del sito web risultante, questi capitoli forniscono un quadro dettagliato della capacità degli LLM di navigare la complessità dello sviluppo di software.

Infine, viene esaminata l'efficienza del processo di deploy nell'ambiente di destinazione, valutando la fattibilità di mantenere e lanciare l'applicazione web con facilità.

L'obiettivo di questa serie di esperimenti è di offrire una panoramica comprensiva dell'applicabilità degli LLM nel contesto dello sviluppo di applicazioni web full stack, evidenziando potenzialità, limitazioni e aree di miglioramento per l'uso di questi modelli nelle pratiche di sviluppo software moderne.





# Capitolo 2

## Nozioni di base

Nel contesto di questa ricerca, è cruciale avere una chiara comprensione delle tecnologie che stanno alla base dei Large Language Models per poter valutare le loro potenzialità nell'ambito delineato nell'introduzione. Pertanto, si dedicherà particolare attenzione all'esplorazione dei principi delle Reti Neurali e dell'architettura Transformer, fondamentali per comprendere la natura degli LLM. Successivamente, verranno dettagliate le specifiche tecniche dei modelli oggetto di studio, seguite da un'analisi della letteratura esistente relativa ai nostri obiettivi di ricerca.

### 2.1 Reti Neurali

Le reti neurali, ispirate alla struttura e al funzionamento del cervello umano, rappresentano un pilastro nell'ambito dell'intelligenza artificiale. Questi modelli computazionali si avvalgono di neuroni artificiali, organizzati in layer, per elaborare informazioni: un layer di input accoglie i dati, i layer nascosti li elaborano, e il layer di output produce il risultato finale. Il cuore dell'apprendimento in queste reti risiede nell'aggiustamento dei pesi delle connessioni tra i neuroni, un processo che ottimizza la rete per svariati compiti, dalla classificazione alla regressione, fino al riconoscimento di pattern.

Una classe di reti neurali, le reti neurali profonde, o *deep learning* sono caratterizzate da una complessità architettonica che include molti layer nascosti tra l'input e

l'output. Questa profondità permette ai modelli di apprendere rappresentazioni dei dati a livelli di astrazione sempre più elevati, consentendo l'identificazione di pattern e caratteristiche complesse che non sarebbero altrimenti riconoscibili. La capacità di trasformare l'informazione attraverso successivi layer di elaborazione rende le reti neurali profonde estremamente potenti per una vasta gamma di applicazioni, che vanno dalla visione artificiale al riconoscimento vocale, dalla soluzione ai problemi in linguaggio naturale (NLP) alla generazione di contenuti.

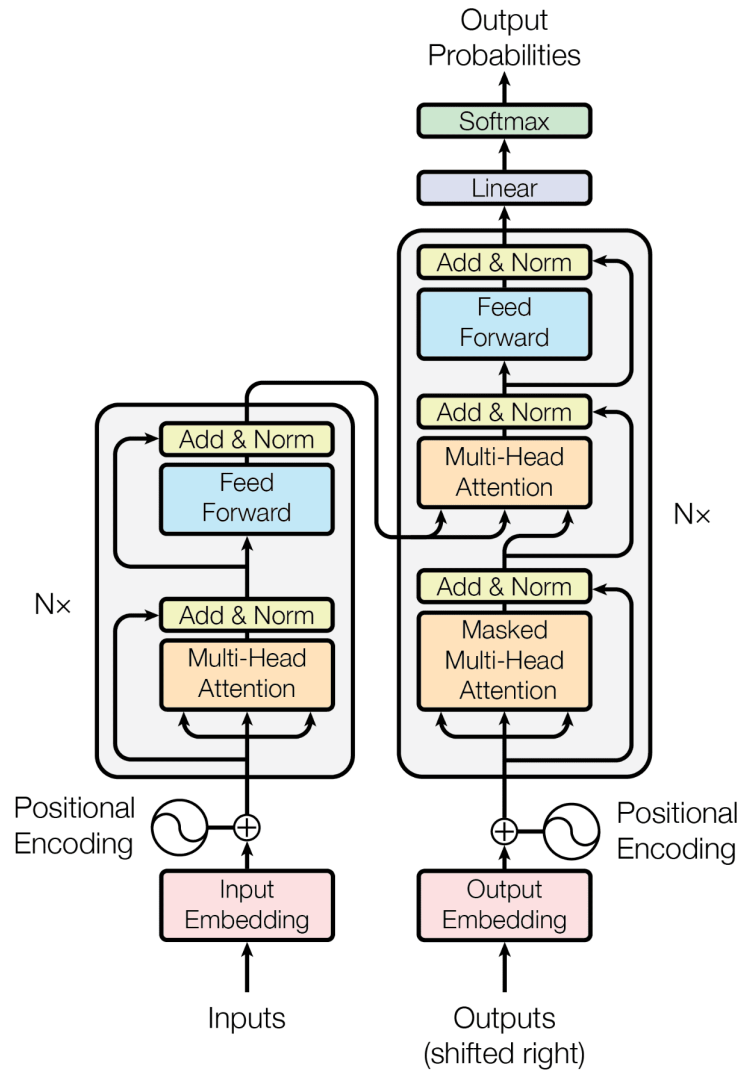
Il successo delle reti neurali profonde si basa sulla loro capacità di eseguire un apprendimento automatico di feature gerarchiche dai dati, eliminando la necessità di feature engineering manuale, un processo che in precedenza richiedeva un'intensa conoscenza di dominio e un significativo sforzo umano. Attraverso l'addestramento su grandi dataset, queste reti sono in grado di scoprire rappresentazioni interne che ottimizzano la performance del modello per il compito specifico a cui sono applicate.

Inoltre, il progresso nelle tecniche di ottimizzazione e nella potenza computazionale, come l'uso di GPU (Graphics Processing Unit) [10] e TPU (Tensor Processing Unit) [10], ha reso possibile l'addestramento di reti sempre più profonde, superando le sfide legate alla diffusione del gradiente e al sovraadattamento. Tecnologie come la normalizzazione batch, le funzioni di attivazione avanzate e i meccanismi di regolarizzazione hanno ulteriormente migliorato la stabilità e l'efficacia dell'addestramento di queste architetture complesse.

Le reti neurali sono diventate strumenti indispensabili nel campo del machine learning. La loro capacità di apprendere da volumi significativi di dati e di identificare pattern complessi ha aperto la strada a progressi significativi in diverse aree, evidenziando il loro ruolo centrale nello sviluppo di soluzioni intelligenti.

## 2.2 Transformer

Il modello Transformer [9], rappresenta un punto cardine nel campo del Natural Language Processing (NLP) grazie alla sua architettura basata esclusivamente sul meccanismo di attention. Questo modello si distingue nettamente dai predecessori, come le reti neurali ricorrenti (RNN) [7][8] e le Long Short-Term Memory



**Figura 2.1:** Il modello Transformer

(LSTM) [8], superando i limiti legati alla gestione delle dipendenze a lungo termine e migliorando l'efficienza computazionale attraverso la possibilità di apprendimento parallelo e una maggiore scalabilità. L'architettura del Transformer [Immagine 2.1] è articolata in due componenti fondamentali: l'encoder e il decoder.

L'encoder del Transformer è composto da una serie di blocchi identici sovrapposti, ognuno dei quali incorpora un meccanismo di attention multi-head seguito da reti feedforward posizionalmente indipendenti. Questa struttura consente al modello di processare l'intero input testuale simultaneamente, contrariamente agli approcci se-

quenziali delle RNN. Grazie a questo, il modello è in grado di catturare le relazioni complesse tra le parole, indipendentemente dalla loro distanza nel testo, migliorando significativamente la comprensione del contesto. Il meccanismo di attention multi-head analizza l'input da diverse prospettive, permettendo al modello di focalizzarsi su differenti aspetti del contesto linguistico per ogni parola, arricchendo la rappresentazione vettoriale dell'input con informazioni contestuali profonde.

Il decoder, simile nell'architettura all'encoder, utilizza l'output dell'encoder per generare il testo sequenzialmente, sfruttando un meccanismo di attention che gli permette di concentrarsi su parti specifiche dell'input mentre genera ogni parola dell'output. La combinazione di questi due componenti, insieme al meccanismo di attention multi-head, rende il Transformer eccezionalmente efficace nel catturare le dinamiche linguistiche complesse e nel generare testo coerente e contestualmente rilevante.

Questa innovativa architettura ha ispirato lo sviluppo di una nuova generazione di modelli di linguaggio, come Gemini e GPT nel nostro caso, che hanno ridefinito i benchmark di successo in svariate applicazioni NLP, dalla traduzione automatica alla sintesi testuale, dalla comprensione del testo al riconoscimento vocale. Il successo dei modelli basati su Transformer evidenzia l'importanza del contesto e dell'attenzione nel trattamento del linguaggio naturale, aprendo nuove frontiere nella comprensione e nella generazione del linguaggio umano.

## 2.3 Large Language Model (LLM)

I Large Language Models sono modelli di intelligenza artificiale basati su reti neurali profonde, specializzati nel trattamento del linguaggio naturale. Questi modelli sono progettati per comprendere, generare e manipolare il linguaggio umano, apprendendo da vasti dataset di testo. L'obiettivo principale degli LLM è di catturare le sottigliezze linguistiche, la coerenza del contesto, le strutture grammaticali e i vari livelli di significato presenti nel linguaggio naturale.

Il fondamento teorico degli LLM risiede nell'architettura dei modelli di rete neurale, in particolare il modello Transformer[9]. Questa architettura si distingue per la sua capacità di gestire sequenze di dati, come il testo, in maniera efficiente e flessibile attraverso il meccanismo di attention, che permette al modello di ponderare l'importanza relativa di diverse parti del testo durante la generazione o la comprensione di nuove sequenze.

Gli LLM sono addestrati attraverso un processo noto come apprendimento automatico supervisionato, nel quale il modello viene esposto a grandi quantità di testo. Durante l'addestramento, il modello apprende a predire la parola successiva in una sequenza di testo, affinando progressivamente la sua capacità di generare risposte coerenti e contestualmente appropriate. Questo processo di addestramento richiede risorse computazionali significative e grandi dataset di testo, spesso composti da una vasta gamma di fonti per assicurare una comprensione linguistica ampia e variegata.

Dal punto di vista teorico, gli LLM si basano su concetti chiave del Natural Language Processing (NLP) e del deep learning, integrando nozioni di linguistica computazionale, statistica e teoria dell'informazione. La loro capacità di modellare la lingua si estende oltre la semplice comprensione testuale, integrando la capacità di effettuare inferenze, comprendere il contesto, generare testo creativo e tradurre tra lingue diverse con un livello di fluidità e naturalezza che si avvicina a quello umano.

I modelli di questo tipo non sono privi di limiti e potenziali errori, che sono intrinseci agli LLM.

## 2.4 GPT (OpenAI)

<https://chat.openai.com/>

Il modello Generative Pre-trained Transformer 3.5 (GPT-3.5) è un modello sviluppato da OpenAI. Con 175 miliardi di parametri, GPT-3.5 ha segnato un'evoluzione significativa nell'ambito degli LLM, addestrato su un vasto dataset che riflette la varietà dei contenuti disponibili sul web fino al 2021 circa. Questo modello si basa sul concetto di Transformer [9] ed è noto per la sua capacità di contestualizzazione e comprensione del sottotesto, permettendo un'interpretazione del linguaggio umano con un certo livello di precisione.

Il modello riceve l'input testuale, lo converte in token che può elaborare, e attraverso i layer Transformer [9], cattura relazioni e contesti dei token. Le rappresentazioni intermedie create vengono poi decodificate in testo leggibile dall'uomo. GPT-3.5 dimostra un'apprendimento trasferibile, essendo capace di affrontare compiti non esplicitamente previsti durante la fase di addestramento, mostrando una flessibilità notevole. Può generare testo, rispondere a interrogativi, sintetizzare informazioni e produrre codice in diversi linguaggi di programmazione, sebbene la mancanza di una componente visiva nell'addestramento limiti la sua capacità di comprendere e integrare aspetti grafici essenziali per un'esperienza utente completa.

GPT-4 [4], il successore di GPT-3.5, mantiene le fondamenta del suo predecessore ma introduce evoluzioni significative. Con un numero di parametri superiore di diversi ordini di grandezza rispetto a GPT-3.5, GPT-4 estende notevolmente le capacità di comprensione e generazione del linguaggio naturale. Una delle novità più rilevanti è la capacità di elaborare input visivi oltre a quelli testuali, un'abilità non presente in GPT-3.5. Questa competenza permette a GPT-4 di eseguire compiti che richiedono la comprensione di elementi visivi, trasformando gli input visivi tramite un encoder in formati adatti per il modello di testo, e generando output testuali a partire da input che combinano testo e immagini.

La capacità di GPT-4 di integrare e interpretare input visivi apre nuove possibilità di applicazione, specialmente nello sviluppo web e nella progettazione di interfacce utente, dove la comprensione degli aspetti grafici è essenziale per offrire

un'esperienza utente completa e soddisfacente.

In sintesi, mentre GPT-3.5 ha posto le basi per l'elaborazione avanzata del linguaggio naturale e l'applicazione pratica nei vari settori, GPT-4 porta queste capacità un passo avanti, introducendo l'elaborazione di input visivi e superando alcune delle limitazioni precedenti. Entrambi i modelli condividono principi fondamentali come la contestualizzazione, la comprensione del sottotesto e l'apprendimento trasferibile, ma è l'innovazione di GPT-4 nell'integrazione di elementi visivi che ne estende il potenziale applicativo.

Nell'ambito dello sviluppo web, un modello addestrato esclusivamente su basi testuali è in grado di generare output codificati con coerenza logica e sintattica. Tuttavia, questo approccio potrebbe risultare carente nell'interpretazione e nella realizzazione di elementi grafici. La mancanza di una componente visiva nell'addestramento limita significativamente la capacità del modello di comprendere e integrare aspetti grafici essenziali per un'esperienza utente completa e soddisfacente. Di conseguenza, pur assicurando una solidità nel back-end e nelle funzionalità basate su testo, si potrebbero osservare lacune nella progettazione front-end e nell'interfaccia utente, che sono critici per l'usabilità complessiva del sito web.

Dimensione Modello	Descrizione Modello
GPT-3.5	Si distingue per la sua robustezza e versatilità in un'ampia gamma di compiti di NLP. È ottimizzato per offrire un bilanciamento tra performance di generazione di testo, comprensione contestuale e latenza, rendendolo adatto per applicazioni che richiedono interazioni conversazionali naturali e generazione di contenuto di alta qualità.
GPT-4	Date le sue dimensioni notevolmente maggiori, aumenta e migliora le funzionalità di GPT-3.5. Risulta essere più performante e robusto, inoltre integra la possibilità di analizzare input visivi dato l'Encoder interno che li trasforma in testo.

**Tabella 2.1:** Descrizione delle funzionalità del modello GPT.

## 2.5 Gemini (Google)

<https://gemini.google.com/app>

Il modello Gemini [3](si fa riferimento al modello “Pro”), frutto dello sviluppo di Google AI, si distacca dai tradizionali LLM come GPT-4, attraverso l’implementazione di un’architettura basata sull’apprendimento multimodale (ovvero apprendimento su immagini, testo, video e audio).

Il modello Gemini è stato addestrato utilizzando acceleratori TPUv5e e TPUv4 [5], scegliendo tra questi in base alle dimensioni specifiche del modello e alla configurazione richiesta. Inoltre, viene addestrato su un dataset che è contemporaneamente multimodale e multilingue, incorporando una vasta gamma di dati che provengono da documenti web, libri e codice, nonché includendo dati sotto forma di immagini, audio e video.

Per la tokenizzazione, viene utilizzato il tokenizer SentencePiece [6], il quale, addestrato su un ampio campione dell’intero corpus di addestramento, ha dimostrato di migliorare significativamente il vocabolario inferito e di conseguenza le prestazioni del modello. Ad esempio, si è osservato che il modello è in grado di effettuare la tokenizzazione in modo efficiente anche di script non latini, il che si traduce in benefici tangibili per la qualità del modello, così come per la velocità di addestramento e di inferenza. A differenza di GPT, allenato su un training set limitato al 2021, il modello di Google può fornire informazioni aggiornate alla data odierna.

Il modello esibisce la capacità di analizzare immagini e video, interpretando relazioni spaziali e identificando oggetti. Queste competenze estendono notevolmente la sua capacità di comprensione del contesto e gli consentono di fornire risposte che richiedono l’integrazione di elementi visivi e testuali.

Anche Gemini è sviluppato sulla base del modello Transformer [9] e vanta un apprendimento di tipo trasferibile che permette di affrontare con flessibilità compiti non esplicitamente previsti durante la fase di addestramento.

Nel contesto dello sviluppo web, l’impiego di un modello addestrato su contenuti multimediali dovrebbe essere capace di fornire risultati che potrebbero mantenere un equilibrio tra l’integrità del codice e la sofisticatezza grafica. Tale approccio di



formazione potrebbe conferire al modello una comprensione più completa dei requisiti necessari per lo sviluppo web. Di conseguenza, si faciliterebbe la progettazione di interfacce utente che sarebbero non solo tecnicamente solide ma anche esteticamente valide e coinvolgenti, migliorando così l'esperienza complessiva dell'utente.

Dimensione Modello	Descrizione Modello
Ultra	Il modello più capace, con alte prestazioni in una vasta serie di compiti di elevata complessità, come il ragionamento e le attività multimodali.
Pro	Il modello è ottimizzato per massimizzare le prestazioni rispetto al costo e alla latenza, dimostrandosi efficace in un'ampia gamma di attività e dotato di eccellenti capacità di ragionamento e funzionalità multimodali.
Nano	Il modello più efficiente per l'utilizzo on-device, è stato sviluppato in due varianti con 1.8 miliardi (Nano-1) e 3.25 miliardi (Nano-2) di parametri per dispositivi con memoria ridotta e avanzata.

**Tabella 2.2:** Descrizione delle funzionalità del modello Gemini.

## 2.6 Prompt Engineering

Il Prompt Engineering si riferisce alla possibilità di formulare domande o istruzioni (prompt) in modo che massimizzino l'efficacia e la precisione delle risposte fornite da un modello di intelligenza artificiale basato su linguaggio. Questa pratica è fondamentale per sfruttare appieno il potenziale degli LLM, in quanto la qualità e la precisione dell'input influenzano direttamente l'output generato.

La ricerca sul Prompt Engineering [2] ha messo in evidenza diverse strategie per ottimizzare l'interazione con gli LLM:

- **Role-prompting:** consiste nel dare al modello un ruolo specifico da interpretare, come quello di un assistente disponibile o di un esperto conoscitore. Questo approccio può essere particolarmente efficace nel guidare le risposte del modello e assicurare che queste siano in linea con l'output desiderato. Ad esempio, se al modello viene chiesto di comportarsi come uno storico, è più probabile che fornisca una risposta dettagliata e contestualmente accurata quando gli viene chiesto di un evento storico. Questa tecnica sfrutta la capacità del modello di adattarsi a diverse prospettive e modi di pensare, migliorando la pertinenza e la precisione delle sue risposte.
- **Zero-shot:** in questo approccio, il modello è sollecitato a rispondere a una richiesta senza alcun addestramento specifico sulla task in questione. Ciò implica la capacità del modello di generalizzare da ciò che ha imparato durante il training a nuove task non viste prima.
- **Few-shot:** questa tecnica prevede di fornire al modello esempi limitati che illustrano il tipo di risposta desiderata. Questi esempi servono a guidare il modello verso una migliore comprensione della task specifica, migliorando la qualità delle risposte generate.
- **Chain of Thought:** questa strategia coinvolge la fornitura di passaggi intermedi di ragionamento per guidare le risposte del modello. Può essere facilitata attraverso semplici prompt come "Pensiamo passo dopo passo" o attraverso una serie di dimostrazioni manuali, ognuna composta da una domanda e una catena di ragionamento che porta a una risposta. Il CoT prompting fornisce

una struttura chiara per il processo di ragionamento del modello, semplificando la comprensione di come il modello è giunto alle sue conclusioni.

- **Contextualized Knowledge Generation:** In questa strategia, gli LLM vengono incoraggiati a generare potenzialmente informazioni utili su una determinata domanda o richiesta prima di generare una risposta finale. Questo metodo è particolarmente efficace nelle attività che richiedono il ragionamento basato sul buon senso, poiché consente al modello di generare e incorporare contesto aggiuntivo che potrebbe non essere esplicitamente presente nella richiesta iniziale. Mediante la generazione preliminare di informazioni rilevanti e il loro utilizzo nella risposta, il modello migliora la coerenza logica e la completezza delle sue risposte, tenendo conto di fattori salienti che altrimenti potrebbero essere trascurati.
- **Retrieval Augmentation:** mirare a ridurre le allucinazioni, cioè la generazione di informazioni irreali o inaccurate. La tecnica coinvolge l'incorporazione di conoscenze esterne aggiornate nell'input del modello, consentendo al modello di migliorare la coerenza e l'accuratezza delle risposte utilizzando informazioni da fonti esterne.

Questi sono solo alcuni delle tecniche utilizzabili per migliorare la qualità e la correttezza degli output ottenuti.

## 2.7 Letteratura Esistente

La ricerca “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation” [11] esplora l'utilizzo di sistemi multi-agente che integrano diversi modelli per lo sviluppo di applicazioni autonome. Questo metodo multi-agente apre nuove possibilità nello sviluppo di software complesso, grazie alla cooperazione tra agenti che possiedono diverse competenze e capacità cognitive.

Sebbene l'approccio proposto dallo studio rappresenti un significativo progresso nel campo, la valutazione delle capacità di un singolo LLM nell'affrontare autonomamente progetti ben definiti resta rilevante. Questo aspetto diventa particolarmente pertinente in contesti in cui l'impiego di sistemi avanzati come AutoGen risulta limi-

tato da restrizioni di risorse, economiche o infrastrutturali. Pertanto, questa tesi si propone di indagare in maniera approfondita le prestazioni e le possibilità offerte da un approccio basato su un unico agente LLM, analizzando i vantaggi e le sfide legate alla gestione autonoma di progetti di sviluppo web, con l'obiettivo di evidenziare sia i limiti sia le potenzialità di tale metodologia nell'ambito della programmazione e della progettazione software.

# Capitolo 3

## Definizione del progetto

In questa sezione, delinearemo le specifiche e le funzionalità dell'applicazione da sviluppare, nonché le proprietà dell'ambiente di deploy finale. Questi elementi saranno cruciali per la valutazione finale del progetto, poiché la non conformità a tali specifiche potrebbe risultare nel fallimento del deploy dell'applicativo o nell'incompleta implementazione delle funzionalità previste.

### 3.1 Descrizione dell'applicativo

L'applicativo che si vuole realizzare è per la gestione di note e memo che includa un sistema di autenticazione per gli utenti.

Vogliamo valutare le scelte prese dai modelli per realizzare le seguenti parti:

- **Frontend:** Progettare un'interfaccia utente responsive e accessibile, che offra un'esperienza utente fluida e intuitiva. Le funzionalità quali aggiunta, modifica, eliminazione di note, modifica di nome utente e della password, cancellazione di account. Eventualmente verificare la possibilità di inserire immagini come input per poter replicare temi di stile.
- **Backend:** Sviluppare una logica applicativa robusta, che gestisca efficacemente le sessioni degli utenti e l'interazione con il database. È essenziale implementare un sistema di autenticazione sicuro per proteggere i dati degli utenti

e garantire l'accesso alle note solo a utenti autorizzati. La creazione di API efficienti per la gestione delle note è fondamentale per assicurare prestazioni elevate e la sicurezza dei dati.

- **Database:** Selezionare e configurare adeguatamente il database per supportare lo storage e la gestione delle note. È importante progettare uno schema di database che ottimizzi le operazioni di query per un recupero efficiente delle informazioni. La protezione delle informazioni sensibili degli utenti è un aspetto critico, che richiede l'adozione di misure di sicurezza adeguate.

Per quanto riguarda le funzionalità richieste:

- Gli utenti devono avere la possibilità di registrarsi e accedere al servizio, garantendo un sistema di autenticazione sicuro e affidabile.
- Gli utenti devono poter visualizzare un elenco delle proprie note, permettendo una facile navigazione e organizzazione delle informazioni salvate.
- Gli utenti devono essere in grado di creare nuove note, modificarle o eliminarle, assicurando un'interfaccia utente intuitiva per la gestione dei contenuti.
- Gli utenti devono avere l'opzione di visualizzare e modificare le informazioni del proprio account.

La verifica della capacità di un LLM di sviluppare un'applicazione web che affronti tutte le sfide inerenti alla realizzazione di un sito web - dalla progettazione dell'interfaccia utente, alla logica di backend, fino alla gestione del database - apre la prospettiva di estendere questo approccio a progetti di maggiore complessità.

Dimostrando che un LLM può efficacemente gestire l'intero ciclo di vita dello sviluppo di un'applicazione web, si potrebbe considerare la sua applicazione a scenari più ampi, ponendo le basi per uno sviluppo web completamente automatizzato dall'intelligenza artificiale. Questo non solo sottolineerebbe l'evoluzione delle capacità degli LLM, ma offrirebbe anche nuove metodologie per accelerare e ottimizzare la creazione di soluzioni software, ridefinendo il ruolo degli sviluppatori umani in un contesto sempre più integrato con l'intelligenza artificiale.

## 3.2 Descrizione dell'ambiente di deploy

Verificare la capacità di un LLM di gestire in autonomia il deploy di un'applicazione rappresenta un aspetto cruciale per comprendere il livello di autonomia che queste tecnologie possono raggiungere nel ciclo di sviluppo software. Un LLM in grado di orchestrare il deploy su infrastrutture estremamente specifiche, dimostrerebbe non solo una profonda comprensione dei requisiti tecnici specifici dell'applicazione ma anche la capacità di interagire con sistemi di gestione ambientale avanzati.

La struttura di deploy target per questo esperimento è fornita dal Dipartimento di Informatica, Scienza e Ingegneria (DISI), che utilizza Gocker (un'istanza di Docker) come fornitore di container software. Questi container, essenzialmente macchine virtuali minimali, sono configurati per eseguire singole applicazioni, riducendo la necessità di un'intera infrastruttura di sistema operativo.

Il DISI offre macchine virtuali Docker preinstallate con ambienti di sviluppo specifici, tra cui Node, Python, PHP e MongoDB, limitando l'accettazione di immagini con software preinstallato diverso da questi. Inoltre, i container Docker possono accedere a un file system condiviso, consentendo il loro avvio in qualsiasi directory.

I siti saranno dunque raggiungibili ai relativi URL:

<https://site222354.tw.cs.unibo.it> (GPT-4)

<https://site222355.tw.cs.unibo.it> (Gemini)

Di conseguenza, è fondamentale tenere in considerazione alcuni limiti specifici relativi alla nostra configurazione di Docker durante lo sviluppo e il deploy dell'applicazione. In particolare, la configurazione predefinita stabilisce che, per Node.js, l'unica porta disponibile e aperta sia la porta 8000. E' necessario che questa restrizione venga comunicata al modello per verificare se la soluzione fornita è coerente con le specifiche di deploy.

Inoltre, è richiesto l'uso di MongoDB come sistema di gestione del database, in quanto le immagini Docker fornite dal DISI sono preinstallate con questo specifico software. Questo requisito limita la scelta della tecnologia di database, imponendo al modello l'adozione di MongoDB per lo storage dei dati.

La verifica dell'autonomia di un LLM in questo contesto permetterebbe di effettuare una valutazione sulla sua capacità di adattarsi e operare entro i vincoli di un ambiente di deploy specifico.



# Capitolo 4

## Sviluppo dell'applicativo

In questo capitolo, condurremo una serie di esperimenti utilizzando un prompt iniziale sviluppato attraverso varie strategie di Prompt Engineering. Valuteremo gli esperimenti che dimostrano maggior potenziale per ciascun modello di linguaggio analizzato: GPT-3.5 (Paragrafo 4.2), GPT-4 (Paragrafo 4.3) e Gemini (Paragrafo 4.4). I risultati vengono poi discusse e vengono fornite osservazioni dettagliate sullo sviluppo legato a ogni LLM considerato.

### 4.1 Prompt di partenza

Abbiamo utilizzato un prompt **Zero-shot** con **Role-prompting** [2], per migliorare la coerenza dei risultati. Il prompt è il seguente:

```
Sei un web developer, realizza un applicazione per la gestione di
note con autenticazione degli utenti.
```

```
Deve essere presente un frontend, un backend e un DB.
```

```
Il deploy deve avvenire su un massimo di due container docker,
il DB deve essere Mongo. La porta esposta per il backend
Node.js deve essere la porta 8000.
```

```
Struttura il progetto e fornisci passo dopo passo come
sviluppare l'applicazione.
```

## 4.2 GPT-3.5

### 4.2.1 Prompting

GPT-3.5 si distingue per la sua velocità (probabilmente dovuta alla dimensione ridotta del modello) ma tende a fornire risultati meno dettagliati rispetto alle sue versioni più evolute.

Organizza in modo adeguato il progetto e struttura le directory **frontend** e **backend** correttamente. Il focus iniziale del modello è sul backend, dove si nota una mancanza di attenzione verso aspetti critici come la sicurezza delle password:

```
1 // backend/models/User.js
2
3 const mongoose = require('mongoose');
4
5 const userSchema = new mongoose.Schema({
6   username: String,
7   email: String,
8   password: String
9 });
10
11 module.exports = mongoose.model('User', userSchema);
```

L'approccio identificato nel codice potrebbe esporre a rischi di sicurezza per quanto riguarda la gestione delle credenziali, che risultano memorizzate in testo non criptato nel database.

Quando si è richiesto un Dockerfile per effettuare test all'interno di un container locale, la proposta del modello ha omesso le istruzioni necessarie per implementare adeguatamente la build statica del frontend. Questa mancanza limita la capacità di condurre test in un ambiente locale che simuli accuratamente l'ambiente di produzione.

```
1 FROM node:latest
2 WORKDIR /app/backend
3 COPY backend/package.json ./
4 RUN npm install
5 COPY backend .
```

```
6 EXPOSE 8000
7 CMD ["node", "server.js"]
```

Si riscontra inoltre una tendenza del modello GPT-3.5 a fornire codice incompleto, anche quando espressamente richiesto, lasciando all'utente il dovere di completare diverse sezioni del lavoro di programmazione.

```
1 // backend/routes/auth.js
2
3 const express = require('express');
4 const router = express.Router();
5
6 // Rotte per la gestione dell'autenticazione
7 // Esempio: router.post('/login', loginController);
8
9 module.exports = router;
```

Similmente a quanto osservato per il backend, anche il frontend è trattato in maniera superficiale. Le indicazioni fornite sono vaghe e talvolta non si allineano con le implementazioni già effettuate. Per quanto riguarda i componenti del frontend, non viene fornita alcuna logica funzionale e spesso si omette del tutto il contenuto interno dei componenti.

```
1 // frontend/src/components/Login.js
2
3 import React from 'react';
4
5 function Login() {
6   return (
7     <div>
8       <h2>Login</h2>
9       { /* Form di login */ }
10    </div>
11  );
12 }
13
14 export default Login;
```

Di conseguenza, si rende necessario l'uso di un numero consistente di prompt per ottenere risposte precise e significative, anche se non completamente coerenti con le scelte prese dal modello in precedenza.

## 4.2.2 Commenti

Il modello GPT-3.5 si distingue per la sua rapidità, attribuibile alle dimensioni ridotte, ma tende a fornire risultati meno dettagliati rispetto alle versioni successive. Nonostante sia in grado di organizzare e strutturare adeguatamente i progetti, spesso manca di approfondimento in aree critiche come la sicurezza e la logica applicativa, sia nel backend che nel frontend. Questa superficialità comporta la necessità di un elevato numero di interazioni per ottenere risposte complete e pertinenti.

In conclusione, GPT-3.5 può essere raccomandato come strumento ausiliario per la programmazione, offrendo un supporto di base e rapido, ma con un livello di autonomia molto basso. È particolarmente adatto in scenari dove una guida superficiale è sufficiente e dove l'utente possiede la competenza per completare e affinare il lavoro iniziato dal modello.

## 4.3 GPT-4

<https://site222354.tw.cs.unibo.it>

### 4.3.1 Prompting

Dando in input lo stesso prompt presentato in 4.1, il modello fornisce una struttura chiara per il setup del progetto e per gli strumenti utilizzati:

```
Creare un'applicazione per la gestione di note con autenticazione degli utenti, frontend, backend, e un database MongoDB, il tutto deployato su container Docker, richiede diversi passaggi. Ecco una guida passo passo su come strutturare e sviluppare l'applicazione:
```

```
1. Progettazione dell'Applicazione
```

```
Backend:
```

Linguaggio/Framework: Node.js con Express per gestire le richieste HTTP.

Autenticazione: Utilizzare JWT (JSON Web Tokens) per l'autenticazione degli utenti.

API REST: Creare endpoint per la gestione delle note (CRUD: Create, Read, Update, Delete) e per l'autenticazione/registrazione degli utenti.

Frontend:

Framework: React per costruire l'interfaccia utente.

Stato dell'Applicazione: Utilizzare Context API o Redux per gestire lo stato dell'applicazione, inclusa l'autenticazione.

Database:

MongoDB: Database NoSQL per memorizzare gli utenti e le note.

## 2. Configurazione del Progetto

Inizializzare i progetti:

Backend: `npm init` nella cartella del backend.

Frontend: `npx create-react-app frontend`.

Dipendenze:

Backend: Installare Express, mongoose (per MongoDB), bcryptjs (per hashing delle password), jsonwebtoken (per JWT), cors, dotenv.

Frontend: Installare axios (per le chiamate API), react-router-dom (per il routing).

Configurazione Docker:

Creare un file Dockerfile nel backend per costruire l'immagine del container.

Creare un `docker-compose.yml` per orchestrare il backend, frontend, e MongoDB.

## 3. Sviluppo del Backend

Struttura delle Cartelle:

Models (per i modelli di Mongoose).

Routes (per definire le rotte API).

Middleware (per funzioni di middleware, ad es. verifica del JWT).  
Controllers (per la logica di business).

Implementazione:

Configurare Express e connettersi a MongoDB.  
Definire i modelli di Mongoose per gli utenti e le note.  
Creare le rotte per l'autenticazione degli utenti e la gestione delle note.  
Implementare la logica per la registrazione, il login, e la verifica del JWT.

#### 4. Sviluppo del Frontend

Struttura delle Cartelle:

Components (per i componenti React).  
Context (per la gestione dello stato globale, se necessario).  
Services (per le funzioni che interagiscono con l'API backend).

Implementazione:

Configurare le rotte con React Router.  
Creare componenti per la registrazione, il login, e la visualizzazione delle note. Implementare la logica per inviare richieste al backend.

#### 5. Dockerizzazione

Backend Dockerfile:

Specificare l'immagine di base (ad es., node:alpine).  
Copiare i file del progetto e installare le dipendenze.  
Esporre la porta 8000.

docker-compose.yml:

Definire i servizi: backend, frontend (se si decide di containerizzare anche il frontend), e MongoDB.  
Utilizzare l'immagine ufficiale di MongoDB per il database.

Configurare le porte e i volumi.

Il modello risponde con coerenza alle richieste fatte, optando per un database MongoDB e un backend Node.js. Come richiesto, gestisce l'autenticazione con una libreria scelta in autonomia (JWT).

GPT-4 fornisce una struttura di progetto coerente e inizializza correttamente tutte le dipendenze necessarie. Le indicazioni fornite sulla dockerizzazione sono parzialmente incoerenti con la richiesta, poichè suggerisce un docker-compose con con tre container (backend, frontend, e MongoDB.) invece che con due come richiesto.

Procedendo con la scrittura del backend:

```
1 // backend/app.js
2
3 const express = require('express');
4 const path = require('path');
5 const mongoose = require('mongoose');
6 const cors = require('cors');
7 require('dotenv').config();
8 const authRoutes = require('./routes/authRoutes');
9 const noteRoutes = require('./routes/noteRoutes');
10 const userRoutes = require('./routes/userRoutes');
11
12 const app = express();
13 const PORT = process.env.PORT || 8000;
14
15 app.use(cors());
16 app.use(express.json());
17
18 mongoose.connect(process.env.MONGO_URI,
19 { useNewUrlParser: true, useUnifiedTopology: true })
20 .then(() => console.log('Connected to MongoDB'))
21 .catch(err => console.log(err));
22
23 app.listen(PORT, () => {
24   console.log('Server is running on port ${PORT}');
25 });
26
27 app.use('/api/user', userRoutes);
```

```
28 app.use('/api/auth', authRoutes);
29 app.use('/api/notes', noteRoutes);
```

Il file base per il backend viene scritto correttamente, utilizzando la cartella `/routes` per definire le chiamate alle API. Di seguito un esempio per la chiamata di POST per creare una nuova nota:

```
1 // backend/routes/noteRoutes.js
2
3 const express = require('express');
4 const Note = require('../models/Note');
5 const { verifyToken } = require('../middleware/authMiddleware');
6 const router = express.Router();
7
8 // Crea una nuova nota
9 router.post('/', verifyToken, async (req, res) => {
10   try {
11     const { title, content } = req.body;
12     const note = new Note({
13       user: req.user.userId,
14       title,
15       content
16     });
17     await note.save();
18     res.status(201).send(note);
19   } catch (error) {
20     res.status(400).send(error);
21   }
22 });
```

Viene scritto il middleware (`verifyToken`) per verificare lo stato di autenticazione di un utente quando viene richiesta una determinata route:

```
1 // backend/middleware/authMiddleware.js
2
3 const jwt = require('jsonwebtoken');
4
5 exports.verifyToken = (req, res, next) => {
6   const token = req.headers['authorization'];
7
```



```

8   if (!token) {
9     return res.status(403).send({ message: 'No token provided!' });
10  }
11
12  jwt.verify(token, process.env.JWT_SECRET, (err, decoded) => {
13    if (err) {
14      return res.status(401).send({ message: 'Unauthorized!' });
15    }
16    req.user = decoded;
17    next();
18  });
19 };

```

Infine vengono descritti gli schema per il database (utenti e note). Riportiamo il modello per gli utenti:

```

1  // backend/models/User.js
2
3  const mongoose = require('mongoose');
4  const bcrypt = require('bcryptjs');
5
6  const userSchema = new mongoose.Schema({
7    username: { type: String, required: true, unique: true },
8    password: { type: String, required: true }
9  });
10
11 // Middleware per criptare la password prima di salvare l'utente
12 userSchema.pre('save', async function(next) {
13   if (!this.isModified('password')) return next();
14
15   this.password = await bcrypt.hash(this.password, 8);
16   next();
17 });
18
19 // Metodo per verificare la password durante il login
20 userSchema.methods.comparePassword =
21 function(candidatePassword, callback) {
22   bcrypt.compare(candidatePassword, this.password,
23     (err, isMatch) => {
24     if (err) return callback(err);

```

```

25     callback(null, isMatch);
26   });
27 };
28
29 const User = mongoose.model('User', userSchema);
30
31 module.exports = User;

```

Nonostante l'LLM avesse fornito una cartella `/controllers` che non è stata utilizzata e non vi è stato scritto nessun file, il backend risulta completo di tutte le funzioni richieste. Il modello si è anche concentrato sull'aspetto di sicurezza, assicurandosi che le password non vengano confrontate o salvate in chiaro. Per il corretto funzionamento dell'applicazione è necessario un file di variabili ambientali `.env`, fornito anch'esso dal modello:

```

MONGODB_URI=tuo_mongodb_uri
JWT_SECRET=VnTi7g6DXqbrd6a2taXLBf17DSkL0zUyfK1TlGqVXiQ

```

Dato che l'URL per MongoDB verrà fornito in un secondo momento, durante le fasi di test e di deploy, il file `.env` è configurato correttamente. Terminato il backend, abbiamo iniziato la scrittura del frontend.

```

1 // frontend/src/App.js
2
3 import React from 'react';
4 import { BrowserRouter as Router, Route, Switch }
5   from 'react-router-dom';
6 import HomePage from './components/HomePage';
7 import LoginPage from './components/LoginPage';
8 import RegisterPage from './components/RegisterPage';
9 import NotePage from './components/NotePage';
10
11 function App() {
12   return (
13     <Router>
14       <Switch>
15         <Route path="/" exact component={HomePage} />
16         <Route path="/login" component={LoginPage} />
17         <Route path="/register" component={RegisterPage} />

```

```

18     <Route path="/notes" component={NotePage} />
19     </Switch>
20 </Router>
21 );
22 }
23
24 export default App;

```

Il codice fornito è sostanzialmente corretto per le versioni inferiori alla 6 di `react-router-dom`, lo standard corrente sarebbe di rimpiazzare `Switch` con `Routes` e la dichiarazione delle pagine in `<Route path="/" element={<Page />} />`. Viene fatto notare al modello che procede a correggerlo correttamente.

GPT-4 procede con la scrittura dei componenti necessari al funzionamento del sito, ne riportiamo successivamente un esempio:

```

1 // frontend/src/components/LoginPage.jsx
2
3 import React, { useState } from 'react';
4 import axios from 'axios';
5
6 function LoginPage() {
7     const [username, setUsername] = useState('');
8     const [password, setPassword] = useState('');
9
10    const handleLogin = async () => {
11        try {
12            const response = await axios.post(
13                'http://localhost:8000/api/auth/login', {
14                username,
15                password
16            });
17            localStorage.setItem('token', response.data.token);
18            // Redirigi l'utente alla pagina delle note
19        } catch (error) {
20            console.error(error);
21        }
22    };
23
24    return (

```

```

25 <div>
26   <input type="text" value={username}
27     onChange={e => setUsername(e.target.value)} />
28   <input type="password" value={password}
29     onChange={e => setPassword(e.target.value)} />
30   <button onClick={handleLogin}>Login</button>
31 </div>
32 );
33 }
34
35 export default LoginPage;

```

Una volta terminati i componenti viene scritto un context personalizzato per la gestione dello stato di autenticazione dell'utente:

```

1 // frontend/src/context/AuthContext.js
2 import React, { createContext, useContext,
3   useState, useEffect } from 'react';
4
5 const AuthContext = createContext();
6
7 export const useAuth = () => useContext(AuthContext);
8
9 export const AuthProvider = ({ children }) => {
10   const [isAuthenticated, setIsAuthenticated] = useState(false);
11
12   useEffect(() => {
13     const token = localStorage.getItem('token');
14     setIsAuthenticated(!!token);
15   }, []);
16
17   return (
18     <AuthContext.Provider
19       value={{ isAuthenticated, setIsAuthenticated }}>
20       {children}
21     </AuthContext.Provider>
22   );
23 };

```

Viene quindi scritto un ulteriore componente wrapper per proteggere le pagine riservate agli utenti con login effettuato e viene aggiornata la pagina principale per utilizzare il nuovo componente:

```
1 // frontend/src/components/ProtectedRoutes.jsx
2
3 import React from 'react';
4 import { Navigate, useLocation } from 'react-router-dom';
5 import { useAuth } from '../context/AuthContext';
6
7 function ProtectedRoute({ children }) {
8   let { isAuthenticated } = useAuth();
9   let location = useLocation();
10
11   if (!isAuthenticated) {
12     return <Navigate to="/login"
13       state={{ from: location }} replace />;
14   }
15
16   return children;
17 }
18
19 export default ProtectedRoute
```

```

1 // frontend/src/App.js
2
3 import React from 'react';
4 import { BrowserRouter as Router, Routes, Route }
5   from 'react-router-dom';
6 import HomePage from './components/HomePage';
7 import LoginPage from './components/LoginPage';
8 import RegisterPage from './components/RegisterPage';
9 import NotePage from './components/NotePage';
10 import CreateNote from './components/CreateNote';
11 import { AuthProvider } from './context/AuthContext';
12 import Navbar from './components/Navbar';
13 import ProtectedRoute from './components/ProtectedRoute';
14 import EditNote from './components/EditNote';
15 import AccountUpdateForm from './components/AccountUpdateForm';
16
17 function App() {
18   return (
19     <AuthProvider>
20       <Router>
21         <Navbar />
22         <Routes>
23           <Route path="/" element={<HomePage />} />
24           <Route path="/login" element={<LoginPage />} />
25           <Route path="/register" element={<RegisterPage />} />
26           <Route
27             path="/notes"
28             element={
29               <ProtectedRoute>
30                 <NotePage />
31               </ProtectedRoute>
32             }
33           />
34           <Route
35             path="/edit-note/:id"
36             element={
37               <ProtectedRoute>
38                 <EditNote />
39               </ProtectedRoute>

```

```

40     }
41   />
42   <Route
43     path="/create-note"
44     element={
45       <ProtectedRoute>
46         <CreateNote />
47       </ProtectedRoute>
48     }
49   />
50   <Route
51     path="/update-account"
52     element={
53       <ProtectedRoute>
54         <AccountUpdateForm />
55       </ProtectedRoute>
56     }
57   />
58 </Routes>
59 </Router>
60 </AuthProvider>
61 );
62 }
63
64 export default App;

```

Il frontend possiede quindi ogni funzionalità richiesta, tuttavia il sito non risulta navigabile per la mancanza di una Navbar. Viene chiesta esplicitamente al modello di realizzarne una in linea con il progetto appena creato, e con bottoni diversi a seconda del grado di autenticazione dell'utente:

```

1 // frontend/src/components/Navbar.js
2 import React from 'react';
3 import { Link, useNavigate, useLocation } from 'react-router-dom';
4 import { useAuth } from '../context/AuthContext';
5
6 function Navbar() {
7   const { isAuthenticated, setIsAuthenticated } = useAuth();
8   const navigate = useNavigate();

```

```

9   const location = useLocation();
10
11  const handleLogout = () => {
12    localStorage.removeItem('token');
13    setIsAuthenticated(false);
14    navigate('/'); // Redirect to home page after logout
15  };
16
17  const getNavLinkClass = (path) => {
18    return location.pathname === path ? 'active' : '';
19  }
20
21  return (
22    <nav>
23      <div className="nav-links">
24        {!isAuthenticated && (
25          <>
26            <Link to="/">Home</Link>
27            <Link to="/login">Login</Link>
28            <Link to="/register">Registrati</Link>
29          </>
30        )}
31
32        {isAuthenticated && (
33          <>
34            <Link to="/notes">Note</Link>
35            <Link to="/create-note">Crea Nuova Nota</Link>
36            <Link to="/update-account">Modifica Account</Link>
37            <button onClick={handleLogout}>Logout</button>
38          </>
39        )}
40      </div>
41    </nav>
42  );
43 }
44
45 export default Navbar;

```

Il frontend è ora ufficialmente completo. Nei paragrafi successivi, esamineremo l'ef-



ficacia di GPT-4 nel creare uno stile coerente per il sito appena creato. Valuteremo tre scenari: senza alcun input visivo, fornendo un input visivo dello stato attuale del sito, e inserendo un input visivo con modifiche apportate manualmente.

### 4.3.2 Nessun input visivo

Il modello è riuscito a sviluppare una applicazione funzionante, ma completamente priva di stile.

L'usabilità del sito è compromessa per vari motivi, tra cui l'assenza di placeholder nei campi di testo, dimensioni del font troppo piccole, e la mancanza di margini o padding per separare adeguatamente gli elementi, risultando in un sito visivamente inadeguato.

[Home](#)[Login](#)[Registrati](#)

**Welcome to the Note Management App**

[Register](#)[Login](#)

**Figura 4.1:** Home page realizzata da GPT-4

[Home](#)[Login](#)[Registrati](#)

## Login

**Figura 4.2:** Login page realizzata da GPT-4

[Note](#)[Crea Nuova Nota](#)[Modifica Account](#)

## Crea una nuova nota

Titolo:

Contenuto:

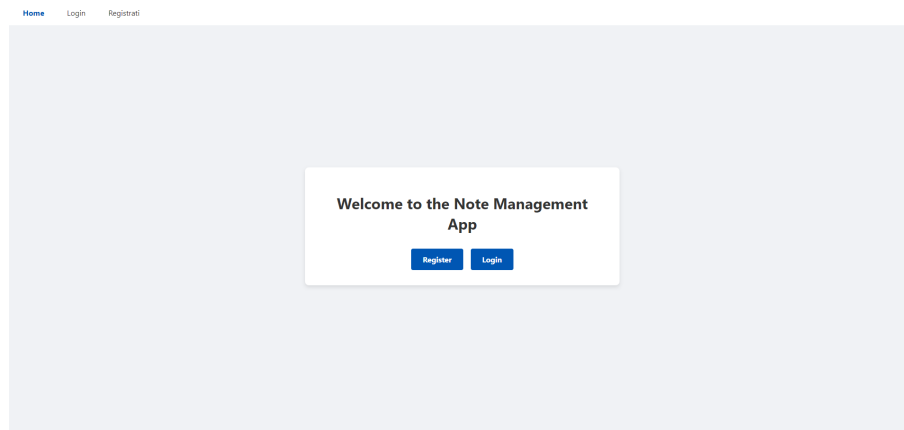
**Figura 4.3:** Form di creazione della nota realizzato da GPT-4

### 4.3.3 Input visivi

Per migliorare lo stile, viene fornito a GPT-4 uno screenshot dello stato attuale della pagina e un prompt molto vago:

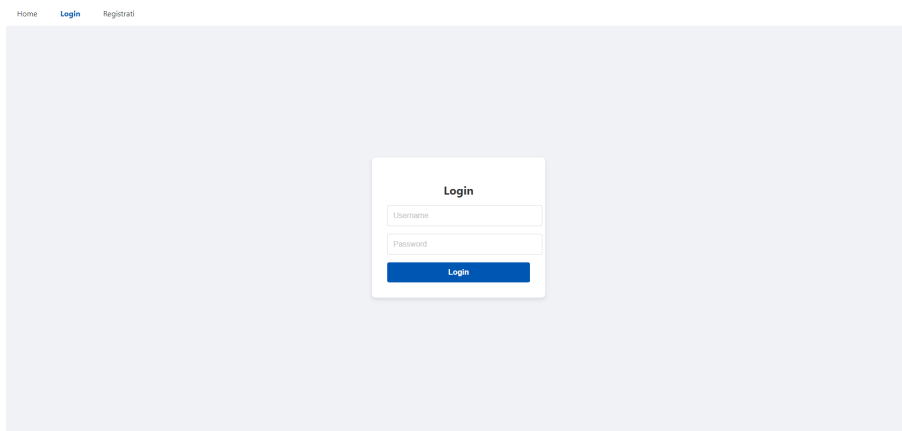
```
/* screenshot */  
  
Migliora lo stile della pagina.  
  
/* codice del frontend della pagina */
```

Il modello, operando in totale autonomia, genera codice CSS per un tema uniforme e procede alla modifica del codice frontend per inserire `className` e addattare il codice ai cambiamenti. Questo processo viene applicato a ciascun componente del sito web.

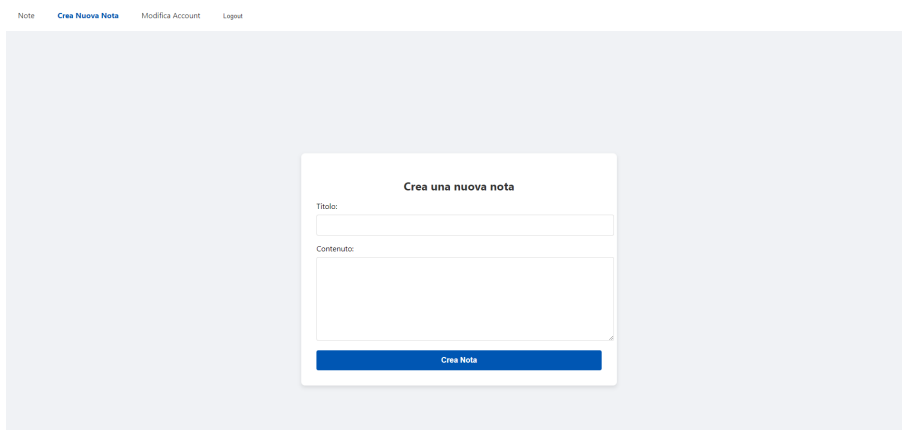


**Figura 4.4:** Home page (con stile) realizzata da GPT-4

Sebbene lo stile applicato abbia migliorato notevolmente l'aspetto del prodotto finale, permangono alcune imperfezioni legate al posizionamento (gli elementi non sono allineati perfettamente al centro) e alle dimensioni (i campi di testo presentano una larghezza disomogenea, essendo più estesi da un lato).



**Figura 4.5:** Login page (con stile) realizzata da GPT-4



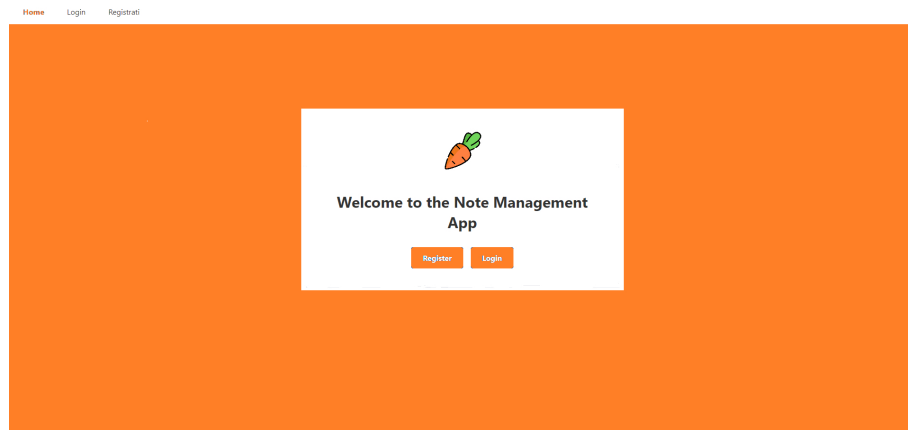
**Figura 4.6:** Form di creazione della nota (con stile) realizzato da GPT-4

### 4.3.4 Input visivi con modifiche umane

Nell'ultimo esperimento, abbiamo deciso di testare l'inserimento di un layout modificato come input, per verificare se il modello sarebbe stato in grado di generare un risultato adeguato. Abbiamo inoltre chiesto la creazione di un'immagine da utilizzare come logo, con l'intento di sostituire quella presente nel layout originale.

Tuttavia, GPT-4 non ha apportato le modifiche previste alla Navbar, nonostante le modifiche richieste fossero minime, e ha introdotto stili non richiesti e problematici per la navigabilità del sito, come ad esempio l'effetto hover dei bottoni che risultava dello stesso colore dello sfondo. Inoltre, ha eliminato alcuni elementi, come lo sfondo bianco al centro dello schermo.

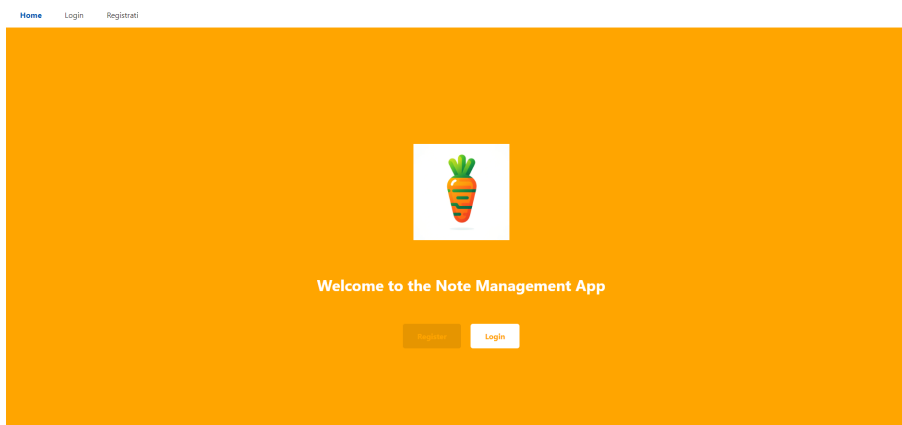
Nonostante numerosi tentativi e diversi prompt forniti, non siamo riusciti a ottenere il risultato desiderato in questo contesto.



**Figura 4.7:** Layout modificato a mano



**Figura 4.8:** Immagine generata da GPT-4 per rimpiazzare l'icona



**Figura 4.9:** Layout modificato da GPT-4 su prompt

### 4.3.5 Commenti

Nel tentativo di raggiungere il risultato desiderato, sono stati realizzati più di dieci esperimenti, dei quali solamente uno ha portato a conseguenze significative. Si sono verificate frequenti situazioni in cui i prompt risultavano inefficaci, mancando di coerenza con i tentativi precedenti.

Si sono rilevate lacune nell'implementazione di determinate funzionalità, e si sono incontrate difficoltà nella configurazione del Dockerfile. Per assicurare un deployment efficace, risulta fondamentale disporre di una versione statica del sito gestita dal backend e effettuare test in un ambiente locale. Ciò implica l'impiego di un Dockerfile che integri correttamente la directory di build con il codice del backend. I Dockerfile proposti dal modello, in diverse occasioni, non hanno rispettato le specifiche richieste, nonostante numerosi tentativi di adeguamento.

Sebbene l'esito di questo esperimento sia stato positivo, è stato necessario apportare alcune correzioni manuali, seppur in misura limitata, principalmente a causa del riferimento a standard non più attuali, dato che il database di GPT è aggiornato fino al 2021.

Si è osservata anche una carenza nella resa grafica, persino utilizzando input visivi, in linea con il fatto che il modello è stato addestrato esclusivamente su testo.

In conclusione, il modello esibisce un discreto grado di autonomia ma si dimostra più adeguato come strumento di supporto nella programmazione piuttosto che come sviluppatore completamente autonomo.

## 4.4 Gemini

<https://site222355.tw.cs.unibo.it>

### 4.4.1 Prompting

Gemini fornisce una struttura di progetto organizzata, tuttavia, mostra incoerenze rispetto ai requisiti specificati, in particolare l'intenzione di utilizzare due container

separati, uno per il frontend e uno per il backend, senza prevedere alcun container dedicato al database.

```
Progetto: Applicazione per la gestione di note

Struttura del progetto

Frontend:

Cartella frontend:
Libreria React
Libreria Material UI
File index.js per l'avvio dell'app
Componenti React per la visualizzazione e la modifica delle note
Componente per la login e la registrazione degli utenti

Backend:

Cartella backend:
File server.js con l'avvio del server Node.js
Moduli per la gestione delle note e degli utenti
Modulo per l'autenticazione con JWT
Integrazione con MongoDB
Database:

MongoDB: Archiviazione delle note e degli utenti
Docker:

File docker-compose.yml per la configurazione di due container:
Uno per il frontend (immagine nginx)
Uno per il backend (immagine node)
```

Dopo aver fatto notare l'errore, viene fornita di nuovo un architettura a tre container totali e in seguito ad ulteriori tentativi, l'LLM risponde in una lingua diversa da quella del prompt fornito, e non fornisce una soluzione.

Viene quindi richiesto di decidere se iniziare dal frontend o dal backend, ma il modello continua a fornire pro e contro di iniziare da uno o dall'altro senza decidere.

Il modello mostra una precisione teorica elevata, ma manifesta alcune incoeren-



ze, specialmente nella generazione del codice. Gli utenti spesso devono intervenire attivamente per correggere codice incompleto o errato, deducendo le intenzioni originarie dell'LLM. Questa necessità di correzione riguarda sia lo sviluppo backend che frontend, sebbene il codice backend tenda a essere simile a quello di versioni precedenti, come GPT-4, ma meno organizzato a causa della mancanza di istruzioni dettagliate per la gestione delle directory o dei file.

Per il frontend, il modello raccomanda l'uso di MaterialUI per lo styling e inizia correttamente il progetto, ma continua a fornire codice parziale o errato, facendo riferimento a componenti non definiti o implementati. Questo approccio porta a soluzioni che non si armonizzano con la struttura già definita, risultando in un prodotto finale non funzionante.

In tentativi successivi, si è fatto progresso sviluppando un backend e un frontend completi tuttavia, a causa delle discrepanze tra l'implementazione nel backend e quella nel frontend, il prodotto finito non è in grado di andare oltre la pagina di login.

#### **4.4.2 Commenti**

Dopo un numero considerevole di tentativi, superiore rispetto agli altri modelli testati, le ricorrenti imperfezioni riscontrate ostacolano la realizzazione di un progetto completamente funzionale senza interventi correttivi significativi da parte dell'utente.

Emergono quindi le limitazioni di Gemini, che non raggiunge un livello di autonomia sufficiente e necessita dell'intervento diretto di uno sviluppatore per risolvere le discrepanze e gli errori nel codice fornito.

Purtroppo la natura multimodale di Gemini non è stata sfruttata data l'incapacità di generare autonomamente codice corretto per un frontend funzionante.

L'impiego del modello come supporto alla programmazione è fattibile e può risultare efficace, ma non si è al riparo da occasionali errori di generazione o codice non corretto, come precedentemente menzionato.



## Rotte API nel server.js (Continuazione)

### 8. Rotta per l'eliminazione di una nota:

#### JavaScript

```
router.delete('/notes/:id', async (req, res) => {
  const noteId = req.params.id;

  // Recupero della nota con l'ID specificato
  const note = await Note.findById(noteId);

  // Se la nota non esiste
  if (!note) {
    return res.status(404).json({ error: 'Nota non trovata' });
  }

  // Eliminazione della nota dal database
  await note.delete();

  // Invio di una risposta di successo
  res.status(200).json({ message: 'Nota eliminata correttamente' });
});

**9. Middleware di autenticazione:**

```javascript
const auth = require('../middleware/auth');

router.use(auth);

// Le rotte successive a questo punto saranno protette da autenticazione
```

Figura 4.10: Errore di Gemini nella generazione del codice

✦ **Utilizzo di un singolo container per frontend e backend**

虽然我先前推荐使用两个容器架构，但也可以选择将前端和后端代码部署在一个容器中。

**Vantaggi:**

- **Semplicità:** Richiede meno configurazione e risorse poiché tutto è in un unico container.
- **Rapidità di sviluppo:** Può essere più veloce per l'impostazione iniziale e lo sviluppo rapido.

**Svantaggi:**

- **Complessità:** Mescola preoccupazioni del frontend e del backend, rendendo il container più complesso e difficile da gestire.
- **Scalabilità:** Scalare l'intera app significa scalare entrambi frontend e backend contemporaneamente, anche se uno ha bisogno maggiore di risorse.
- **Sicurezza:** I problemi di sicurezza in una parte potrebbero compromettere l'altra.

**Figura 4.11:** Risposta di Gemini multilingua al prompt “E’ necessario fornire un Dockerfile unico per backend e frontend”

```
JavaScript

import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import Button from '@material-ui/core/Button';
import NoteForm from './NoteForm';
import axios from 'axios';

const useStyles = makeStyles({
  root: {
    width: '50%',
    margin: 'auto',
  },
});

const NotePage = ({ match, history }) => {
  const classes = useStyles();
  const [note, setNote] = useState(null);

  useEffect(() => {
    axios.get(`/api/notes/${match.params.id}`)
      .then((res) => setNote(res.data))
      .catch((err) => console.log(err));
  }, [match.params.id]);

  const onDeleteNote = () => {
    axios.delete(`/api/notes/${match.params.id}`)
      .then(() => history.push('/'))
      .catch((err) => console.log(err));
  };

  const onEditNote = (updatedNote) => {
    setNote(updatedNote);
  };

  return (
    <div className={classes.root}>
      {note && (
        <>
          <Typography variant="h4">{note.title}</Typography>
          <Typography variant="body1">{note.content}</Typography>
          <Button
            variant="contained"
            color="secondary"
            margin="normal"
            onClick={onDeleteNote}
          >
            Elimina
          </Button>
          <NoteForm onEditNote={

```

Usa il codice [con cautela](#).

**Figura 4.12:** Gemini fornisce codice incompleto al prompt “Scrivi il codice per la pagina Note-Page.jsx”

- [Home](#)
- [Login](#)
- [Registrati](#)

## Home page

Benvenuto nella tua app per la gestione delle note!

**Figura 4.13:** Home page realizzata da Gemini



# Capitolo 5

## Valutazioni

Per analizzare le prestazioni dei tre modelli di linguaggio (GPT-3.5, GPT-4 e Gemini) nello sviluppo web full stack, prenderemo in considerazione vari fattori che indicano il livello di autonomia di questi Large Language Models (LLM) in tale ambito. Gli elementi chiave da valutare sono: l'accuratezza del codice prodotto, la sofisticatezza grafica dell'interfaccia utente del prodotto finito, e l'usabilità di tale prodotto.

Dopo aver confrontato questi modelli su tali parametri, procederemo con un'analisi dei limiti incontrati da ciascun modello durante il loro utilizzo.

### 5.1 Correttezza

La correttezza del codice, definita come la capacità di funzionare adeguatamente in base alle istruzioni fornite, rappresenta un criterio cruciale nella generazione di codice da parte degli LLM.

I tre modelli in esame hanno dimostrato di essere capaci di generare codice valido quando considerato singolarmente, al di fuori del contesto più ampio del progetto. Tuttavia, l'introduzione del criterio dei requisiti del progetto cambia la situazione. Nonostante i frammenti di codice generati fossero intrinsecamente corretti, spesso si rivelavano incompatibili quando combinati, portando alla creazione di un progetto complessivamente disfunzionale.

Tra i modelli analizzati, GPT-4 e GPT-3.5 hanno mostrato una maggiore coesione nel codice prodotto, necessitando di minori interventi manuali per garantire la coerenza complessiva. Al contrario, i tentativi con Gemini hanno frequentemente portato a codice frammentato e a volte erroneo, includendo errori significativi come riferimenti indefiniti e l'uso di funzioni inesistenti.

## 5.2 Sofisticazione grafica

La sofisticatezza grafica, che si riferisce all'attrattiva visiva di un sito web e alla coerenza del suo stile complessivo, è cruciale nello sviluppo web. È fondamentale che una pagina non solo sia funzionalmente precisa, ma che appaia anche esteticamente gradevole agli utenti.

Durante lo sviluppo del sito non è stato stabilito nessuno stile specifico da parte di nessuno dei tre modelli, ma su richiesta dell'utente hanno cercato di apportare miglioramenti estetici al design finale. Sebbene la maggior parte dei tentativi abbia prodotto risultati insoddisfacenti, con problemi quali accostamenti cromatici inadatti e disallineamenti orizzontali o verticali che compromettevano la fruibilità del sito, GPT-4 si è distinto offrendo un esito discreto. Nonostante ciò, anche questo stile ha mostrato limitazioni, tra cui la non uniformità delle dimensioni e problemi di allineamento.

Da sottolineare è che, nonostante i modelli avessero raccomandato l'adozione di framework per lo styling (ad esempio, Tailwind, MaterialUI, Bootstrap) al fine di migliorare l'estetica, l'implementazione effettiva di questi framework si è rivelata complessa da integrare con gli altri elementi del progetto, risultando più efficace un approccio di styling più basilare tramite CSS tradizionale.

## 5.3 Usabilità

L'usabilità, definita come la facilità con cui gli utenti navigano e interagiscono con un sito web, è di vitale importanza nello sviluppo web, al pari della sofisticatezza grafica. Per garantire una positiva esperienza utente, è essenziale che un sito sia non solo esteticamente accattivante ma anche intuitivo e facilmente accessibile.



Le funzionalità di base sono state implementate con successo dai modelli, con GPT-4 che ha dimostrato una maggiore coerenza tra i componenti rispetto agli altri due. Tuttavia, tutti i modelli hanno incontrato difficoltà nel realizzare funzionalità aggiuntive per potenziare l'usabilità complessiva dell'applicazione. Ad esempio, la possibilità di modificare username, email o password del proprio account è stata attuata solo in parte da GPT-4, e nonostante numerosi tentativi, non è stato possibile aggiungere alcuna funzionalità completamente operativa.

Inoltre, ogni notifica viene presentata agli utenti attraverso alert, una scelta non ottimale per l'esperienza utente.

Per quanto concerne la sicurezza, sia GPT-4 che Gemini hanno optato per l'uso di una libreria esterna per un gestione sicura delle credenziali, con GPT-4 che ha mostrato una migliore integrazione di questa soluzione nel contesto del progetto.

Sul fronte dell'accessibilità, ci sono stati tentativi da parte dei modelli di fornire codice per realizzare pagine accessibili, ma nessuno di questi tentativi ha pienamente rispettato le linee guida sull'accessibilità.

Infine, lo stile adottato non garantisce una piena responsività del sito: pur essendo navigabile su schermi di dimensioni ridotte, l'aspetto grafico risulta poco gradevole.

## 5.4 Tabella riassuntiva

	GPT-3.5	GPT-4	Gemini
Strutturazione del progetto	✓	✓	✓
Correttezza di codice isolato	✓	✓	Frequenti errori di generazione
Coerenza del codice con il progetto	Bassa	Discreta, con correzioni umane	Nulla, nella maggior parte dei casi
Realizzazione di stile	×	Discreto, con qualche incoerenza	×
Esperienza utente	×	Non ottimale	×
Accessibilità	×	×	×
Deploy	×	✓	Notevoli correzioni umane richieste

## 5.5 Eventuali limiti dei modelli

Nell'analisi dei modelli, GPT-3.5 si è rivelato particolarmente efficiente in compiti di piccola scala e specifici, una caratteristica che può derivare dalla sua disponibilità come modello gratuito offerto da OpenAI.

Per contro, GPT-4 si distingue per la sua completezza, pur mostrando una minore velocità di elaborazione, che potrebbe essere attribuita alla sua architettura più complessa e al maggior numero di parametri. Un aspetto critico di GPT-4 è la sua difficoltà nel mantenere il contesto di conversazioni lunghe e complesse, una limitazione che si presume sia legata alla gestione delle risorse da parte di OpenAI.

Gemini, invece, tende a mostrare meno efficacia nei compiti di dimensioni medie e grandi, trovandosi più a suo agio con incarichi limitati e ben definiti. Questo modello appare progettato per essere più complementare all'intervento umano che completamente autonomo. La sua propensione a focalizzarsi su attività di portata ridotta suggerisce una capacità moderata di portare avanti progetti di ampio respiro senza un considerevole supporto esterno.

Complessivamente, sebbene ciascun modello offra vantaggi distinti in termini di efficienza, accuratezza e flessibilità, essi condividono delle sfide comuni, quali la gestione della continuità del contesto nelle interazioni prolungate e la gestione efficiente di compiti complessi e vasti.



# Capitolo 6

## Conclusione

La presente tesi ha esplorato le capacità degli LLM (GPT-3.5, GPT-4 e Gemini) nell'ambito dello sviluppo di applicazioni web full stack, con l'obiettivo di valutare il loro impatto sulla velocità e sull'efficacia del processo di sviluppo software.

Come illustrato nei capitoli dedicati agli esperimenti con ciascun modello, è stato possibile osservare come l'impiego di questi modelli influenzi significativamente le fasi di progettazione e implementazione, accelerando la creazione di prototipi e facilitando la realizzazione di funzionalità complesse.

Nonostante le potenzialità mostrate, gli esperimenti hanno anche evidenziato limiti non trascurabili. La difficoltà nel mantenere coerenza e contesto nei progetti più complessi, unito alle sfide legate all'integrazione di stili grafici avanzati e alla garanzia di usabilità e accessibilità, sottolinea la necessità di ulteriori miglioramenti per questi LLM. In particolare, come discusso nei capitoli sperimentali, la creazione di interfacce utente coerenti e specifiche, l'integrazione efficace con il backend e la gestione del database rappresentano aspetti critici che richiedono attenzione e interventi manuali da parte degli sviluppatori.

Nel dettaglio, esploriamo i vantaggi e le sfide emerse nel corso di questi esperimenti, evidenziando i principali ostacoli incontrati e i benefici ottenuti nell'uso degli LLM per lo sviluppo:

**Capacità di Generazione Codice e Velocità di Sviluppo:** I modelli LLM hanno dimostrato una notevole efficacia nell'accelerare la creazione di prototipi e nella realizzazione di funzionalità complesse, grazie alla loro capacità di generare rapidamente codice funzionale per varie parti dell'applicazione, dal frontend al backend e alla gestione del database. Questo aspetto, come discusso nei Paragrafi 4.2, 4.3 e 4.4, evidenzia il potenziale degli LLM di ridurre i tempi di sviluppo e di permettere agli sviluppatori di concentrarsi su aspetti più innovativi e creativi dei loro progetti.

**Integrazione di Stili Grafici e Usabilità:** Nonostante la capacità degli LLM di produrre codice, la loro efficacia nel generare interfacce utente che siano non solo esteticamente piacevoli ma anche coerenti e funzionali risulta essere uno dei principali limiti. Le sfide nell'assicurare un'elevata usabilità e accessibilità, conformi agli standard WCAG [1] e alle migliori pratiche per l'esperienza utente, sottolineano la necessità di un'attenta supervisione e di interventi manuali da parte degli sviluppatori, come illustrato nelle Sezioni 4.3.2, 4.3.3, 4.3.4, 5.2 e 5.3.

**Comprensione e Gestione del Contesto nei Progetti Complessi:** Come spiegato al termine del Paragrafo 5.1, un'altra area critica riguarda la difficoltà degli LLM di mantenere il contesto in progetti ampi, dove è richiesta una comprensione profonda e integrata delle diverse parti dell'applicazione. Questa limitazione implica che, per il momento, gli LLM non possono garantire un'autonomia completa nello sviluppo di software, specialmente in fasi che richiedono decisioni progettuali complesse e una forte coerenza tra i vari componenti dell'applicazione.

**Implicazioni per le Competenze degli Sviluppatori:** L'utilizzo efficace degli LLM richiede una nuova competenza critica tra gli sviluppatori: la capacità di formulare prompt precisi e dettagliati (Paragrafo 4.1) che guidino i modelli verso la generazione di soluzioni specifiche e di alta qualità. Questo richiede non solo una solida conoscenza tecnica ma anche una comprensione approfondita delle potenzialità e dei limiti degli LLM, evidenziando l'importanza di un continuo aggiornamento professionale in questo settore in rapida evoluzione.

In conclusione, sebbene gli LLM rappresentino un avanzamento significativo nella semplificazione e nell'accelerazione dello sviluppo di applicazioni web, la loro implementazione pratica solleva questioni complesse che vanno oltre la generazione di codice. Il successo del loro utilizzo dipenderà dalla capacità degli sviluppatori di navigare queste sfide, sfruttando le competenze di Prompt Engineering per massimizzare l'efficienza e l'efficacia degli LLM. Future ricerche dovrebbero concentrarsi sul superamento delle attuali limitazioni tecniche e sulla formazione degli sviluppatori per sfruttare appieno le capacità di questi potenti strumenti.

Per il futuro, è opportuno che la ricerca si concentri sull'ulteriore miglioramento delle capacità di contestualizzazione e di gestione dei progetti complessi da parte degli LLM, sulla personalizzazione degli stili grafici e sull'integrazione più fluida con le tecnologie del backend e del database.

Un altro ambito di interesse riguarda l'evoluzione del Prompt Engineering e dello sviluppo di interfacce più intuitive per la comunicazione con questi modelli, in modo da rendere la loro potenza accessibile anche a sviluppatori non esperti in tecniche avanzate.

In conclusione, i risultati ottenuti aprono nuove prospettive sull'utilizzo degli LLM nello sviluppo autonomo di applicazioni web, ponendo le basi per future indagini e sviluppi che possano portare alla realizzazione di processi di sviluppo software sempre più efficienti e automatizzati, in grado di rispondere efficacemente alle esigenze dinamiche del settore.





# Bibliografia

- [1] Web content accessibility guidelines. URL <https://www.w3.org/WAI/standards-guidelines/wcag/>.
- [2] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering in large language models: a comprehensive review, 2023. URL <https://arxiv.org/abs/2310.14735>.
- [3] Gemini Team et al. Gemini: A family of highly capable multimodal models, 2023. URL <https://arxiv.org/abs/2312.11805>.
- [4] OpenAI et al. Gpt-4 technical report, 2023. URL <https://arxiv.org/abs/2303.08774>?
- [5] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A Patterson. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture, ISCA '23*, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700958. doi: 10.1145/3579371.3589350. URL <https://doi.org/10.1145/3579371.3589350>.
- [6] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71,

- Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL <https://aclanthology.org/D18-2012>.
- [7] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [8] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020. ISSN 0167-2789. doi: <https://doi.org/10.1016/j.physd.2019.132306>. URL <https://www.sciencedirect.com/science/article/pii/S0167278919305974>.
- [9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL <https://arxiv.org/abs/1706.03762>.
- [10] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.
- [11] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. URL <https://doi.org/10.48550/arXiv.2308.08155>.