Politecnico di Torino

Politecnico di Torino

Ingegneria Informatica

A.a. 2023/2024

Sessione di Laurea Dicembre 2024

Generazione di interfacce grafiche su misura tramite intelligenza artificiale

Relatori:

Luigi De Russis

Tommaso Calò

Candidato:

Andrea Sillano

Ringraziamenti

La stesura di questa tesi rappresenta la conclusione del mio percorso universitario, reso possibile anche grazie alle persone che mi sono state vicine e mi hanno costantemente supportato. A queste voglio dedicare i miei più sinceri ringraziamenti.

Ringrazio i miei relatori Luigi De Russis e Tommaso Calò per la grande disponibilità, i consigli e la loro preziosa guida durante lo svolgimento della tesi.

Un ringraziamento speciale va alla mia famiglia, in primis ai miei genitori che mi hanno accompagnato in questo difficile percorso, che con i loro sforzi non mi hanno mai fatto mancare nulla. Mi avete sempre sostenuto credendo sempre in me e senza di voi questo non sarebbe stato possibile. Vorrei ringraziare anche mio fratello, nonché mio coinquilino, che mi ha supportato e accompagnato nel mio percorso incoraggiandomi e sostenendomi.

Un grande grazie ai miei cugini e miei zii per essere mi stati vicino, avermi aiutato ed avermi strappato un sorriso anche nei momenti più difficili.

Vorrei ringraziare i miei nonni che anche se lontani mi sono stati vicini con il cuore e con grande affetto hanno celebrato ogni passo del mio percorso. Desidero dedicare un pensiero speciale anche ai miei nonni materni, siete stati per me fonte di ispirazione e forza. Anche se non siete qui con me fisicamente, vi sento ancora vicini, ed i vostri insegnamenti mi hanno guidato in questo percorso. So che sareste fieri di me.

Desidero ringraziare anche i miei amici di una vita Francesco, Alessio, Antonio che mi hanno sempre incoraggiato e appoggiato.

Vorrei inoltre ringraziare Lorenzo e Giancarlo con i quali ho vissuto fino ad ora le più belle avventure dalla mia vita, grazie per avermi sempre sostenuto, ascoltato ed aiutato.

Desidero ringraziare i miei colleghi Matteo, Davide ed in particolare Riccardo per aver arrichito il mio percorso e reso questi anni indimenticabili.

Ringrazio Sofia per aver reso questo percorso più spensierato rallegrandomi le giornate ed avermi incoraggiato in ogni momento.

Infine, voglio ringraziare tutte le persone incontrate lungo il mio cammino, chi di passaggio e chi no, con le quali ho condiviso questo viaggio rendendolo speciale.

A tutti voi grazie per il vostro contributo con il quale ho raggiunto questo importante traguardo.

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Contesto

L'evoluzione delle interfacce grafiche pone le sue radici negli anni 60'-70', dove troviamo i primi tentativi di permettere all'utente di interagire con un computer attraverso icone e sistemi di puntamento. Proprio in quell'epoca si sviluppa la disciplina che segna l'inizio dello studio sullo sviluppo delle interfacce utente ed esperienza utente. Human Computer Interaction si occupa di studiare come l'utente interagisce con i sistemi informatici gettando le basi e principi per il design.

Solo negli anni 90' però con l'esplosione dei desktop computer, divenuti alla portata di tutti, si è creata la necessità di sviluppare interfacce grafiche di facile utilizzo e accessibili. Con il progredire della tecnologia e delle capacità computazionali dei sistemi moderni la ricerca sulle interfacce grafiche si è dedicata, non solo alla sola semplicità di utilizzo, ma anche e soprattutto all'aspetto.

Con l'avvento degli smartphone e del Web l'attenzione si è concentrata ancor di più sul comprendere quali fossero i bisogni degli utenti per creare delle esperienze più piacevoli. Il numero di utenti che utlizzano servizi digitali è cresciuto enormemente quindi anche la tipologia e capacità degli utilizzatori si è ampliata.

Se infatti nel passato tali servizi erano destinati ad utenti esperti, adesso ciò non è più necessariamente vero, quasi chiunque ha infatti ha acesso al web e di conseguenza ad una vasta scelta di servizi differenti.

I moderni sistemi digitali offrono agli utenti un livello di personalizzazione che può variare notevolmente a seconda della tipologia di servizio utilizzato. Questi sistemi consentono agli utenti di modificare vari aspetti dell'interfaccia e delle funzionalità per adattarli meglio alle loro preferenze personali. Ad esempio, è possibile cambiare il colore di sfondo di un'app, selezionare il tipo di layout preferito o configurare il comportamento di determinati elementi dell'interfaccia.

Tuttavia, in molti casi, la personalizzazione offerta è limitata a una serie di opzioni predefinite: l'utente può scegliere tra una gamma di alternative già predisposte dal sistema, ma non ha la libertà di andare oltre quei limiti imposti. Questo tipo di personalizzazione, che si può definire "implicita" è quindi governata da regole stabilite dagli sviluppatori, piuttosto che dal diretto input dell'utente.

L'obiettivo principale della personalizzazione, in qualsiasi contesto, è quello di migliorare l'esperienza utente, rendendo i servizi più intuitivi, accessibili e veloci da utilizzare. Un'interfaccia che si adatta alle preferenze individuali può ridurre il carico cognitivo dell'utente, migliorare la soddisfazione complessiva e aumentare l'efficienza nell'esecuzione delle attività. Tuttavia, è importante notare che non tutte le forme di personalizzazione sono uguali, e si possono identificare tre principali tipologie di personalizzazione:

- Personalizzazione Individuale - La personalizzazione deriva dalle scelte dell'utente e può essere data-driven.
- Personalizzazione basata su ruolo - La personalizzazione è predetereminata e si basa sul ruolo dell'utente all'interno del servizio.
- Personalizzazione basata sulla posizione - La personalizzazione si basa sulla posizione dell'utente ad esempio lo stesso sito può apparire differente in base alla nazione dove ci connettiamo.

Si può quindi affermare che mentre la personalizzazione implicita offerta dalla maggior parte dei sistemi moderni ha senza dubbio migliorato l'esperienza dell'utente, esiste un potenziale ancora maggiore per sistemi che offrono livelli di personalizzazione su misura, permettendo agli utenti di modellare l'interfaccia e il comportamento delle applicazioni in modi più creativi e mirati.

1.2 Obiettivo

La personalizzazione di una interfaccia grafica può nascere da diverse esigenze da parte degli utenti, tuttavia non sempre tutte le richieste possono essere soddisfatte dalle scelte precostruite nel sistema. Solitamente la maggior parte dei sistemi offrono configurazioni sul tema come ad esempio chiaro o scuro, senza la possibilità di modificare ulteriormente i colori dell'interfaccia o creare combinazioni personali. Anche la personalizzazione dei caratteri spesso risulta limitata in quanto di solito, si prevede la selezione di un tipo di carattere ed una determinata dimensione per l'intera applicazione senza poter effettuare un preciso controllo su di essi.

La configurazione che però rimane quasi solitamente fissa è il layout, questo di solito è composto da elementi che solitamente non possono essere spostati o in caso contrario la loro posizione può essere modificata ma solo in un numero limitato di

posizioni, impedendo all'utente di adattare l'ordine o la posizione degli elementi nel modo più utile e comodo per il loro utilizzo.

Si può quindi affermare che nei sistemi tradizionali la personalizzazione dell'interfaccia ricade bene o male nel "One size fits all", includendo solo un numero limitato di configurazioni disponibili e di conseguenza non tenendo conto delle esigenze e richieste del singolo utente.

La tesi vuole quindi proporre una soluzione che sia in grado di superare gli attuali sistemi di personalizzazione, sfruttando la capacità dei Large Language Model per comprendere le necessità e preferenze degli utenti e successivamente rimodellare l'interfaccia basandosi su queste. La tesi, quindi, ha come obiettivo quello di sviluppare un'architettura di un'applicazione web che consenta all'utente di esprimere le proprie preferenze in modo più soddisfacente e approfondito e creare un'esperienza di utilizzo su misura. Tale architettura si basa sull'impiego di un modello di AI chiamato Large Language Model che permette all'utente di definire le proprie preferenze attraverso il linguaggio naturale andando quindi a innovare l'espressione delle preferenze.

In particolare l'architettura è stata realizzata scegliendo come caso d'uso l'applicazione di un calendario web, la scelta deriva principalmente dalla circostanza che trattatasi di un applicativo con cui la maggior parte delle persone ha esperienza ed è quindi in grado di avere un'idea di confronto con altre app già esistenti.

L'integrazione di questa nuova funzionalità avviene attraverso un menù a scomparsa dove l'utente può selezionare l'elemento del calendario che intende modificare e successivamente descrivere sia la modifica da effettuare sia il nuovo aspetto che esso dovrà assumere. Verrà quindi generata una nuova interfaccia dell'applicazione sulla base delle preferenze espresse dall'utente.

Più dettagliatamente si può suddividere lo sviluppo in tre macro fasi:

- Progettazione calendario web - In questa prima fase avviene lo sviluppo di un calendario web di base, vengono definite quali sono le funzionalità minime che rimarrano indipendenti dalla generazione dell'interfaccia.

- Integrazione AI - In questa seconda fase viene completata l'architettura integrando il LLM che si occuperà della realizzazione dell'interfaccia grafica. È stato quindi necessario rendere l'applicazione capace di modificare il proprio aspetto integralmente durante l'esecuzione.

- Aspetti Utente - In questa ultima fase si è realizzato ciò che concerne l'utente, sono stati per tanto definiti i componenti personalizzabili e le modalità esatte con cui l'utente avrebbe interagito con il menù di generazione.

1.3 Struttura della tesi

La tesi segue lo sviluppo in vari passi della predetta architettura, partendo dallo studio preliminare, seguendo poi con le scelte progettuali ed infine mostrando la soluzione proposta accompognata da uno studio utente per valutarne l'efficacia e usabilità. Più in dettaglio:

- Capitolo 2 "Background" - Background teorico dei vari aspetti che abbracciano vari domini di studio, partendo dalle interazioni uomo-computer, proseguendo all'analisi delle soluzioni già esistenti fino alla modellazione utente ed i LLMs.
- Capitolo 3 "Progettazione" - In questo capitolo vengono definiti quali sono i requisiti base per lo sviluppo del caso d'uso proposto e successivamente una possibile architettura che soddisfi questi ultimi.
- Capitolo 4 "Implementazione" - In questo capitolo viene definito come è stata implementata la soluzione proposta e quali tecnologie sono state utilizzate a tale scopo. Vengono analizzati i principali attori che compongono l'architettura ed il ruolo che essi svolgono all'interno di questa.
- Capitolo 5 "Studio Utente" - Viene descritta la metodologia per la costruzione dello studio utente con la conseguente analisi dei risultati ricavati dall'esperimento.
- Capitolo 6 "Conclusioni" - Considerazioni con punti di forza, eventuali limitazioni e possibili sviluppi futuri.

Capitolo 2

Background

In questo capitolo viene definito in modo più approfondito quale è il contesto di sviluppo della tesi, si approfondisce lo studio delle interazioni macchina-computer con particolare riguardo a ciò che coinvolge la personalizzazione e la realizzazione di un modello utente sfruttabile per tale scopo. Successivamente vengono presentate differenti soluzioni esistenti, infine, si definiscono i LLMs ed il loro funzionamento.

2.1 Interazione Uomo-Computer (HCI)

Lo studio dell'interazione uomo-computer/macchina risale all'inizio del XX secolo in ambito industriale. Solo dopo la seconda guerra mondiale però si sviluppò la Ergonomics Research Society, quest'ultima si occupava esclusivamente del design fisico per ottimizzare le prestazioni. In seguito vennero considerati anche fattori umani come il carico cognitivo che ampliarono notevolmente lo studio, oggi l'HCI si occupa delle interazioni e comunicazioni tra uomo e macchina sia a livello fisico che a livello piscologico. HCI coinvolge due principali attori, l'utente che è chiunque stia svolgendo un'attività con strumenti tecnologici ed il computer che è un qualsiasi elemento elettronico con il quale ci si può interfacciare. L'interazione viene definita come qualsivoglia mezzo di comunicazione tra utente e computer [1].

L'HCI è una materia fortemente multidisciplinare e copre più ambiti tra i quali psicologia, scienza cognitiva, ergonomia, sociologia e informatica, "it is not possible to design effective interactive systems from one discipline in isolation." [1], è possibile quindi affermare che per il design di sistemi efficienti è necessario il contributo di più discipline per garantire usabilità. Lo studio delle interazioni non si muove su di una teoria ben definita, tuttavia il capo saldo attorno al quale ruota il principio più importante è che gli utenti utilizzano i computer per svolgere delle attività quindi il design di tali sistemi deve essere consono all'utilizzo dell'utente.

In particolare come riportato nel libro *Human-Computer Interaction* [1] ci sono "use" words che devono essere vere per far in modo che un prodotto sia di successo:

- "useful" - Deve eseguire qualche operazione.
- "usable" - Deve essere facile da utilizzare.
- "used" - Deve attirare l'utente.

Con l'evolversi delle tecnologie anche i bisogni degli utenti sono mutati portando l'ambito dell'HCI da una visione prettamente di usabilità ed efficenza ad aspetti come l'esperienza utente ed estetica un tempo ritenuti secondari.

2.1.1 Interazione Uomo-AI (HMAI)

L'intelligenza artificiale sta acquisendo sempre più spazio all'interno della nostra società, influenzandone sempre più aspetti. L'AI sta quindi rivoluzionando il nostro modo di rapportarci con la tecnologia e l'HMAI è proprio quella branca dell'HCI che si occupa di come queste interazioni avvengono e di come renderle il più facili ed efficaci possibili.

L'obiettivo è quindi quello di permettere agli utenti di interagire con sistemi intelligenti garantendo però un'usabilità elevata, tuttavia ciò non sempre è di facile realizzazione.

Infatti durante l'utilizzo di tecnologie dotate di AI, a causa della loro natura a volte incerta, si possono produrre risultati anomali andando a diminuire la gradevolezza dell'esperienza utente.

Dato il contesto di questa tesi è necessario capire come approciarsi all'interazione tra uomo ed AI e analizzarne le caratteristiche.

Come riportato nell'articolo «Re-examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design» [2] la costruzione di interfacce ed esperienze utente che integrano sistemi intelligenti presenta non poche difficoltà tra le principali troviamo:

- Difficoltà nel comprendere le capacità dell'AI.
- Difficoltà nell'integrare nuovi metodi di interazioni.
- Difficoltà nell'integrare prototipazione e testing iterativi.

In aggiunta uno degli aspetti principali, affrontati anche all'interno di questa tesi, risiede nel fatto che le AI non sempre si comportano in modo coerente con le aspettative degli utenti. Tali disfunzioni potrebbero portare l'utente a non avere fiducia del sistema, soprattutto se il funzionamento dell'AI non è di facile comprensione. Si è cercato quindi di stilare delle linee guida in questo amibito per

rendere l'esperienza agevole, fluida e comprensibile, nonostante ciò l'attuazione di tali principi rimane complessa. Le linee guida descritte [3] coprono vari ambiti e temi, ma in particolare è opportuno mensionare:

- Chiarezza - Aiuta l'utente a comprendere cosa il sistema fa e come l'AI compie le sue decisioni.
- Gestione degli errori - Garantire che gli errori non compromettano l'utilizzo del sistema.
- Trasparenza - Il sistema deve spiegare per quanto possibile le decisioni dell'AI.
- Adattamento Controllato - Se il sistema si adatta all'utente questo deve avvenire gradualmente senza sorprenderlo o confonderlo.

L'applicazione delle linee guida può contribuire alla realizzazione di un sistema migliore che riesca a coinvolgere l'utente e ad ottenere la sua fiducia.

2.2 Interfacce dinamiche

Le interfacce dinamiche sono progettate per modificare il loro aspetto e comportamento in base al contesto e all'utente per migliorare l'esperienza di utilizzo. Tuttavia progettare tali sistemi non è privo di difficoltà ed ostacoli dovuto al numero di configurazioni e contesti di applicazione.

L'approccio più semplice è quello manuale che però diventa rapidamente insostenibile al crescere del numero di variazioni dell'interfaccia e dei bisogni dell'utente. Proprio a causa di queste problematiche nasce l'esigenza di sviluppare sistemi più flessibili e generici che possano unificare lo svliuppo di un'interfaccia e la personalizzazione su misura. È inoltre importante che questi strumenti offrano una grande flessibilità, permettendo l'adattamento a diversi casi d'uso e garantiscano allo stesso tempo uno sviluppo semplice e fluido.

2.2.1 Personalizzazione delle interfacce

La personalizzazione delle interfacce si sviluppa su due diversi livelli, il layout che riguarda la disposizione e le dimensioni dei singoli elementi all'interno dell'interfaccia e l'aspetto ovvero l'estetica dei componenti. Di conseguenza, verranno analizzate le tecnologie impiegate per entrambi gli aspetti, sia per la gestione del layout che per la personalizzazione dell'aspetto visivo dei componenti, al fine di comprendere come queste permettano la creazione di interfacce adattive e personalizzabili e allo stesso tempo di conoscerne i limiti.

Il design di un layout è un'attività dispendiosa sia in termini di tempo sia in termini di energia. In particolare è una sfida a causa della complessità combinatoria

e del numero di oggetti trattati [4]. Esistono diverse possibili soluzioni a tale problema ed in particolare le reti generative hanno acquisito sempre più rilevanza grazie alla loro abilità di apprendere distribuzioni complesse e multi-dimensionali. Seguono alcune delle architetture per la generazione di layout:

Attribute Conditioned Layout GAN [5]

Layout GAN è una particolare architettura generativa basata sulle Generative Adversarial Network GAN [6]. Questa prevede l'esistenza di due reti, una generativa ed una discriminatoria che lavorano in contrapposizione. In particolare l'architettura proposta è utilizzata per generare i parametri, ovvero le coordinate, degli elementi grafici presenti nell'interfaccia. La rete è stata appositamente modificata con l'aggiunta di nuovi parametri per rendere il layout generato più piacevole ed utilizzabile.

Variational Transformer Network Layout Generation [7]

Il VTN è un modello generativo che utilizza i *transformer* per comprendere relazioni complesse tra gli elementi di un layout, come margini e allineamenti, senza la necessità di supervisione esplicita. A differenza di approcci tradizionali basati su Convolutional Neural Networks (CNNs) o Recurrent Neural Networks (RNNs), il VTN sfrutta la *self-attention* per modellare le relazioni a lungo termine tra gli elementi, rendendolo particolarmente adattoalla generazione di layout di vario tipo. Il VTN sfrutta come architettura di base quella di un Variational Autoencoder (VAE), dove l'encoder-decoder permette al modello di apprendere rappresentazioni latenti che rispettano i vincoli del design dei dati di addestramento, per questo i layout generati mostrano una somiglianza con i dati originali.

Layout VQ-VAE [8]

Il modello VQ-VAE si basa sull'archiettura di un Variational Autoencoder (VAE) superando le soluzioni precedenti permettendo di apprendere e modellare le relazioni tra gli elementi di un layout senza l'utilizzo di euristiche. Questo sistema consente di generare layout diversi basati sugli elementi presenti e sullo scenario fornito. L'obiettivo principale è predirre con precisione la dimensione e le coordinate degli elementi in funzione del contesto, del numero e delle categorie di elementi coinvolti. L'architettura apprende le bounding box, categorie degli elementi e scenario per poter derivare la variabile latente. Questa variabile viene successivamente "interrogata" utilizzando un *transformer* monodirezionale che, in combinazione con un decoder, genera layout personalizzati in base a condizioni specifiche.

GRIDS: Interactive Layout Design with Integer Programming [4]

Introduce un metodo di ottimizzazione basato sulla programmazione lineare intera mista, che garantisce l'inserimento, l'allineamento, il raggruppamento e il posizionamento preferenziale degli elementi in un layout. Questo approccio consente la generazione in tempo reale di layout e, grazie alla sua formulazione matematica, produce soluzioni "esatte", preferibili rispetto a quelle ottenute tramite ricerche casuali. Al modello base, che genera il layout, vengono aggiunti nuovi obiettivi formulati, consentendo al sistema di cercare "iterativamente" layout che soddisfano queste condizioni. L'algoritmo, date le specifiche condizioni, è in grado di esplorare una vasta gamma di layout con efficienza, grazie alla natura polinomiale del problema, che dipende unicamente dal numero di elementi. Questo consente di ottimizzare rapidamente il design del layout, rendendo il processo di creazione interattivo e altamente personalizzabile.

Dopo aver definito ed analizzato alcune delle principali architetture che permettono la generazione di layout è possibile proseguire con modelli che ricadono nella sfera della personalizzazione dell'aspetto dell'interfaccia. Adattare l'interfaccia ad un utente è un'operazione delicata la quale deve tenere in considerazione l'ipotesi che invece di rendere l'utilizzo più semplice, vada ad impattare in modo negativo sull'esperienza e sull'utilizzo.

Adapting user interfaces with model-based reinforcement [9]

La soluzione si basa sull'adozione di un approccio di reinforcement learning (RL) model-based, che sfrutta modelli predittivi per comprendere il comportamento degli utenti e stimare gli effetti delle modifiche proposte sull'esperienza d'uso. Il sistema sviluppato si basa su un modello di RL con Monte Carlo Tree Search (MCTS), riducendo notevolmente i costi computazionali e integrando principi di Human-Computer Interaction (HCI) per migliorare le interazioni utente-sistema. I modelli predittivi utilizzati per determinare le modifiche possono essere specificamente adattati per affrontare diverse problematiche, come il tempo di accesso e il carico cognitivo andando quindi a personalizzare l'interfaccia seguendo diverse metriche.

Refining Graphic Designs by Exploring Design Principles and Human Preference [10]

Lo strumento proposto offre un'opportunità per accelerare il processo di miglioramento delle interfacce. Il tool opera principalmente in due fasi: generazione di candidati guidata dai principi di design ed anche generazione di candidati basata sui dati. Nella prima fase, viene fornito un design di partenza dal quale si producono candidati che rispettano specifiche linee guida di design. Questa generazione tiene conto di restrizioni che consentono ai nuovi candidati di non

discostarsi eccessivamente dal design originale, ma che al contempo apportino miglioramenti grafici apprezzabili. Successivamente, questi candidati vengono esaminati un approccio basato sui dati, al fine di garantire una valutazione il più oggettiva possibile. Viene impiegata una Siamese Network che utilizza due moduli di scoring con parametri condivisi. La rete analizza ogni design ed apprende come bilanciare le diverse caratteristiche ottimizzando ulteriormente il processo di raffinamento.

FrameKit: A Tool for Authoring Adaptive User Interfaces Using Keyframes [11]

FrameKit è un strumento progettato per facilitare la generazione di nuove interfacce utente (UI) a partire da design point, noti come *frames*. A differenza di altri modelli, FrameKit non è vincolato a un contesto di utilizzo specifico rendendolo uno strumento versatile. FrameKit consente agli utenti di creare interfacce composte da vari widgets, ciascuno con proprietà editabili. Gli utenti modificando i parametri dei widget definscono il comportamento di questi in specifiche situazioni e l'insieme delle condizioni e dei valori delle proprietà costituisce un frame. Una volta completata la realizzazione di diversi frame, è possibile generare ulteriori interfacce interpolando il comportamento di questi frame, una funzionalità conosciuta come UI Blending.

ReactGenie: A Development Framework for Complex Multimodal Interactions Using Large Language Models [12]

ReactGenie è un framework di programmazione dichiarativa per lo sviluppo di applicazioni multimodali che separa le interfacce di input/output dai modelli di sottostanti. Il framework integra la traduzione dei comandi degli utenti in chiamate a funzioni tramite modelli linguistici di grandi dimensioni (LLM) che permettono di compiere operazioni sull'interfaccia come la modifica di uno stato o modifica di un parametro.

2.2.2 Limiti delle architetture esistenti

Nonostante le architetture precedentemente citate abbiano raggiunto risultati notevoli, presentano delle limitazioni per lo sviluppo di un'architettura che genera interfacce su misura per l'utente.

Innanzitutto, molte di esse dipendono da euristiche o regole fisse, il che può portare a soluzioni rigide che non si adattano a scenari diversi e ad utenti con differenti richieste. Un altro aspetto critico sono i costi computazionali che crescono esponenzialmente quando si utilizzano approcci come apprendimento rinforzato o algoritmi di ottimizzazione. Un altro limite è dovuto al fatto che tali architetture

non sono progettate per apprendere dai dati di input variabili rendendole incapaci di apprendere dai feedback degli utenti.

In conclusione, si può affermare che questi limiti evidenziano l'importanza di sviluppare architetture più flessibili e intelligenti, capaci di apprendere, adattarsi a diversi scenari e soprattutto alle necessità degli utenti permettendo di costruire interfacce su misura.

2.2.3 Raccolta e modellazione delle preferenze utente

Per avere una personalizzazione su misura è indispensabile acquisire informazione sull'utente. In particolare si possono trovare due soluzioni per la modellazione dell'utente. La prima, *Rule-Based*, dove la costruzione del modello si basa su regole ben definite, mentre la seconda, *Data-driven*, dove l'utente viene definito in base alle informazioni ad esso inerenti [13]. Tuttavia entrambi queste tecniche di modellazione utente non sono esenti da problematiche che si possono così riassumere.

Prima fra tutte come osservato nell'articolo «Challenges in User Modeling and Personalization» [13] viene suggerito che il modello utente che viene prodotto attraverso un approccio basato su regole dovrebbe essere esplorabile. Ciò però può divenire intrattabile se il modello e le connessioni diventano eccessive. Tuttavia questo renderebbe il processo decisionale più comprensibile e deteremistico.

La seconda difficoltà riscontrata è la generalizzazione del modello utente. Infatti non sempre è possibile costruire modelli individuali per ogni utente perciò si realizzano versioni più generali sulla base di regole che raggruppano utenti simili in classi. Questa metodologia può risultare non ottimale poichè la suddivisione in gruppi può far perdere i dettagli dei singoli utenti.

Infine troviamo la selezione delle scelte nei modelli basati sui dati, fondate sul concetto "Saggezza della folla" ma occorre prestare attenzione poichè le scelte attuali possono essere influenzate da un bias proveniente dalle scelte iniziali effettuate dai primi utenti.

Una delle soluzioni che sfruttano un modello utente per la personalizzazione delle interfacce è ARNAULD [14], il quale costruisce una funzione di costo che migliora l'interfaccia sulla base dell'esigenze dell'utente. La raccolta di preferenze si suddivide in *Example Critiquing* dove, data l'interfaccia attuale, vengono registrate le modifiche che l'utente intende eseguire sui vari elementi ed infine viene chiesto all'utente se il risultato è soddisfacente. Sulla base delle scelte viene modificata la funzione di costo. Il secondo passo è la *Active Elicitation*, il sistema interroga l'utente su delle scelte binarie sull'aspetto di alcuni widget, la risposta ai quesiti permette l'aggiornamento dell'interfaccia. Le preferenze vengono poi trasformate in requisiti, trasformando a loro volta il problema che diviene nella ricerca dei pesi per i quali vengono soddisfatte il maggior numero di richieste.

La chiave per una personalizzazione efficace sembra quindi risiedere nella riduzione della complessità architetturale e l'uso di metodi che possano garantire la comprensione delle preferenze per una personalizzazione più accurata, senza trascurare la flessibilità e la trasparenza del processo.

2.3 AI Tradizionale e Modelli Generativi

L'intelligenza artificiale viene definita come la capacità di un sistema di simulare l'intelligenza umana attraverso l'ottimizzazione di funzioni matematiche [15]. Prima dell'avvento dei modelli generativi l'intelligenza artificiale si basava su due pilastri: classificazione e regressione.

La classificazione rientra nella tipologia dell'apprendimento supervisionato e si occupa di etichettare un determinato elemento all'interno di un set di classi predefinite. Durante la fase di addestramento, vengono appresi i pattern e le caratteristiche discriminanti che separano le classi. L'obiettivo è produrre una funzione decisionale che sia in grado di generalizzare su nuovi elementi non visti in precedenza e assegnare loro l'etichetta più probabile in base a quanto appreso.

La regressione invece di classificare un elemento in una classe discreta, apprende una funzione che mappa i dati di input ad un valore continuo, è quindi utilizzato in contesti dove l'obiettivo è stimare o prevedere un valore numerico.

L'AI generativa rapresenta la nuova frontiera tecnologica, se infatti quella tradizionale aveva come obiettivo l'analisi e la classificazione questa vuole produrre contenuti inediti ed originali come testo, foto e recentemente video.

2.3.1 Cosa sono e come funzionano gli LLMs

Come riportato in precedenza esistono diverse tipologie di AI generative, in particolare nell'ambito di questa tesi verranno utilizzate quelle che si occupano della produzione di testo. I modelli linguistici calcolano la probabilità che una determinata parola compaia all'interno di una sequenza in una determinata posizione in una specifica lingua e sulla base di questo costruiscono frasi e testi come rappresentato in figura 2.1.

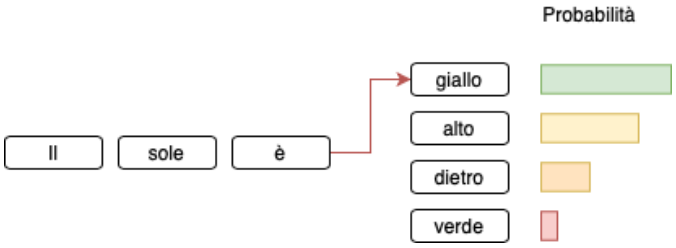

Figura 2.1: Esempio funzionamento LLMs

I primi modelli che iniziarono a riscuotere successo furono le Recurrent Neural Netwok (RNN), si tratta di architetture progettate per compiti da sequenza a sequenza, come traduzioni da una lingua ad un'altra ed inoltre consentivano la costruzione di un contesto inteso come l'insieme dei dettagli che possono influenzare le risposte. Acquisirono poi maggiore spazio con l'introduzione dei moduli Long Short Term Memory che risolsero il problema che causava la scomparsa del gradiente durante l'addestramento, migliorando l'apprendimento del contesto.

Tuttavia queste reti avevano dei problemi che le rendevano inadatte per l'utilizzo di massa, innanzitutto erano estremamente complesse e lente da addestrare ed in secondo luogo la comprensione del contesto era ancora troppo ridotta. La vera rivoluzione in questo ambito è avvenuta con la pubblicazione dell'articolo *Attention Is All You Need* [16] e la progettazione di un nuovo tipo di architettura denominato *transformer*, che ha permesso lo sviluppo dei Large Language Model. Gli LLMs sono modelli linguistici di enormi dimensioni estremamente flessibili e capaci. A differenza delle precedenti architetture le parole non sono rappresentate come una tabella numerica, questi modelli sfruttano dei vettori multidimensionali denominati *embeddings* per la rappresentazione. Questo permette di computare matematicamente le relazione tra le parole e le frasi, successivamente di eseguire in parallelo migliaia di sequenze.

2.3.2 I transformer

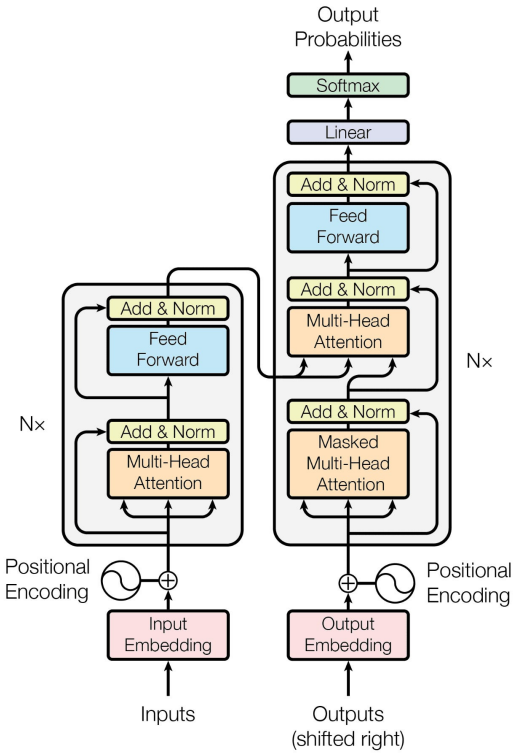

Figura 2.2: Architettura di un Transformer

L'architettura mostrata in figura 2.2 mostra la composizione di un transformer che insieme al meccanismo di *attention* sono il cuore dei nuovi modelli generativi. Essi si compongono principalmente di due elementi l'encoder visibile sulla sinistra ed il decoder rappresentato a destra. Partendo dall'encoder gli input vengono suddivisi in token e come detto in precedenza ne viene calcolato il valore numerico detto *embedding* e nello stesso tempo anche il *positional embeddings* che consente di non perdere l'informazione spaziale di quel token specifico all'interno del testo.

Come si nota dalla figura gli *embeddings* sono inviati al *Multi-Head Attention* e successivamente al passo *Feed-Forward* l'ouput dell'encoder è una rappresentazione dell'attenzione relativa di ogni token rispetto agli altri permettendo di produrre un contesto ben definito. Il decoder invece riceve in input l'output generato fino a quel momento e applica lo stesso trattamento mascherando però il prossimo token prima di passare per il primo *Multi-Head Attention*, questo è necessario poichè in questo modo la predizione non dipende dai valori futuri, ma solo da quelli attuali. Successivamente avviene il passaggio attraverso un secondo modulo di attenzione

che unisce la rapresentazione dell'encoder con quella del decoder. Infine vi sono un layer lineare che produce valori probabilistici per i token e l'ultimo livello applica una softmax per scegliere quale sarà il prossimo token.

Il *Multi-Head Attention* è fondamentale per l'ampliamento e la comprensione del contesto del testo in ingresso, si può immaginare il suo funzionamento come un insieme di lettori che leggono il medesimo testo ma carpiscono informazioni leggeremente diverse poichè prestano attenzione a diversi dettagli. Il principio utilizzato è il medesimo e questo rende i modelli dotati di questo meccanismo estremamente capaci di comprendere il contesto. Sulla base di questa nuova architettura si sono fondati la maggior parte dei LLMs.

2.3.3 GPT

Uno dei modelli che ha completamente rivoluzionato il mondo dell'AI e più nello specifico dei modelli generativi è quello presentato da OpenAI denominato *Generative Pre-Trained Transformer* (GPT). L'introduzione di questo modello è stata presentata nell'articolo «Improving language understanding by generative pre-training» [17], nel quale viene descritta la nuova architettura ed il procedimento utilizzato per il training della prima versione che ha spianato la strada per la realizzazione dei modelli successivi. L'architettura si basa su quella dei *transformer* descritta in precedenza opportunamente modificata, l'obiettivo era quello di predirre la prossima parola sulla base di quelle precedenti come mostrato in figura 2.1.

La vera rivoluzione risiedeva nel metodo a due fasi utilizzato per l'addestramento. La prima fase è denomintata *pre-training*, dove il modello viene addestrato su una grande quantità di testo per "imparare" a predirre le parole, pattern linguistici e sintassi. Successivamente avviene la seconda fase *fine-tuning* dove il modello viene addestrato nuovamente su un dataset più piccolo per specializzarlo in un compito specifico. Questa operazione non solo rende il modello capace di eseguire un compito specifico ma anche come effetto secondario migliora la capacità generale di comprensione del contesto.

Negli anni sono stati rilasciati diversi modelli che di volta in volta crescevano in termini di complessità raggiungendo oltre 1 trilione numero di parametri. Tuttavia questi modelli ancora non sono privi di problematiche, soffrono della cosidetta *allucinazione* che produce output falsati o che non rientrano nell'attuale contesto. Nonostante ciò i modelli GPT rimangono ottimi, possiedono grandi capacità che li rendono in grado di svolgere molteplici compiti che spaziano dalla produzione di testo sino alla generazione di codice.

Capitolo 3

Progettazione

In questo capitolo si delinea il contesto di progettazione, fornendo una descrizione del caso d'uso proposto e dei requisiti che deve soddisfare. Si parte dall'analisi dei requisiti funzionali di base, per poi approfondire le caratteristiche chiave sui quali si basa la tesi. Una volta definito il contesto, si passa all'analisi dell'architettura progettuale proposta, esaminando i vari blocchi dai quali è composta.

3.1 Caso d'uso: Calendario Web

Oggigiorno quasi tutti gli utilizzatori di dispositivi informatici sono entrati in contatto almeno una volta con un calendario, sia esso online oppure sotto forma di applicazione, di fatto quasi tutti i moderni sistemi ne possiedono uno preinstallato. Nonostante ne esistano di diverse tipologie e aspetto, le funzionalità basiche rimangono le medesime permettendo a chiunque abbia un minimo di esperienza di utilizzare in modo semplice qualunque versione. Tuttavia è comunque possibile distinguerli poichè ognuno di essi possiede un'interfaccia unica e offre un'esperienza utente diversa attraverso delle funzionalità specifiche di quella versione.

La maggior parte di essi offre delle possibilità di personalizzare la propria interfaccia per migliorare l'esperienza e per accogliere le esigenze dell'utente al meglio. Di seguito sono riportate alcuni esempi di calendari comunemente utilizzati con le loro offerte di personalizzazione:

- Google Calendar:
  - Impostazione di colori diversi per ogni calendario o evento.
  - Visualizzazioni personalizzabili (giorno, settimana, mese, anno, agenda).
  - Temi scuri e chiari per l'interfaccia.
- Microsoft Outlook:

  - Personalizzazione delle visualizzazioni (giorno, settimana, mese, pianificazione).
  - Temi grafici personalizzabili per l'interfaccia.
  - Opzioni di visualizzazione (a schede, elenco) e selezione layout.

- Microsoft Outlook:
  - Personalizzazione delle visualizzazioni (giorno, settimana, mese, pianificazione).
  - Temi grafici personalizzabili per l'interfaccia.
  - Opzioni di visualizzazione (a schede, elenco) e selezione layout.

- Apple Calendar:
  - Calendari multipli con codici colore personalizzabili.
  - Temi grafici personalizzabili per l'interfaccia.

- Fantastical:
  - Interfaccia personalizzabile con diverse viste (giornaliera, settimanale, mensile, annuale).
  - Temi grafici e stili visivi per personalizzare l'interfaccia.
  - Opzioni di layout.
  - Widget personalizzabili.

Osservando le offerte di personalizzazione dei calendari si può notare come la maggior parte di esse ricada in una selezione di differenti opzioni predetereminate, come la modifica della vista o la scelta tra tema chiaro o scuro etc. Di conseguenza ciò impone implicitamente un limite alle possibilità di modifica che l'utente può eseguire e per quanto queste possano risultare ampie, sarà comunque l'utente a doversi "adattare" piuttosto che viceversa.

Potrebbe infatti capitare che, per gli utilizzi che gli utenti fanno del calendario, queste opzioni non siano sufficienti. Ad esempio, alcuni utenti potrebbero desiderare visualizzazioni più avanzate, personalizzazioni grafiche più dettagliate o modificare posizioni e funzionalità di elementi già disponibili. Inoltre, le esigenze specifiche di ciascun utente possono variare notevolmente e di conseguenza la possibilità di personalizzare completamente l'interfaccia utente potrebbe essere una funzionalità estremamente utile.

La realizzazione di un sistema capace di realizzare tale scopo ad oggi è possibile grazie al progresso in ambito AI degli utlimi anni ed in particolare allo sviluppo di LLMs. OpenAI ha reso pubblico uno dei loro modelli più avanzati che nel nostro

contesto consente all'utente di esprimere in linguaggio naturale le proprie preferenze riguardo l'aspetto e funzionalità dell'interfaccia ed è in grado di generare una nuova interfaccia che sia conforme ai propri bisogni esposti.

3.1.1 Funzionalità

Sulla base di quanto discusso nella sezione 3.1, sono stati definiti i requisiti che l'architettura deve soddisfare. In questa fase ci concentreremo esclusivamente sui requisiti funzionali. Questi rappresentano requisiti tecnici focalizzati sulle funzionalità che l'architettura deve garantire per supportare le esigenze operative dell'applicazione.

Di seguito si trovano i requisiti principali:

- Operazioni di autenticazione - Il sistema deve essere in grado di supportare le operazioni di login e logout. Queste sono necessarie per il funzionamento in quanto identificano l'utente e permettono a questo di ottenere la propria configurazione personale.

- Funzionalità base calendario - Il sistema offre una versione semplice di un calendario web, che permette la navigazione settimanale, la visualizzazione degli eventi e la loro durata.

- Database - Il sistema dispone di un database che non solo memorizza le credenziali dell'utente, ma permette anche il salvataggio delle configurazioni personalizzate prodotte. Ogni volta che viene generata una nuova configurazione, il sistema archivia automaticamente sia la versione corrente che una copia della versione precedente, garantendo così un registro storico delle modifiche. Questo consente all'utente di ripristinare l'ultima configurazione in qualsiasi momento.

- Preferenze utente - Il sistema deve acquisire le preferenze dell'utente in modo agevole e strutturato. Le preferenze possono includere aspetti come il layout, i colori, le dimensioni dei caratteri, e altre impostazioni visive o funzionali.

- Generazione Interfaccia - Il sistema deve essere in grado di generare dinamicamente una nuova interfaccia basata sulle preferenze espresse dall'utente. Una volta acquisite, queste preferenze vengono elaborate dal sistema, che rigenera l'interfaccia in tempo reale per adattarla alle esigenze specifiche dell'utente. In questo modo, l'utente ottiene un'esperienza personalizzata e ottimizzata, basata sulle sue scelte e preferenze.

- Riprisitino Interfaccia - Il sistema mette a disposizione un meccanismo di ripristino già citato in precedenza che permette all'utente sia di ottenere

l'ultima interfaccia prima di quella attuale sia di effettuare un reset tornando alla versione base di questa.

- Feedbacks - Il sistema notifica l'utente riguardo alle sue attività in tempo reale. Durante il caricamento della versione base del calendario, l'utente viene informato sullo stato del processo. Analogamente, durante la fase di generazione di nuove interfacce, il sistema notifica l'utente sull'esecuzione, garantendo che sia sempre consapevole di ciò che sta avvenendo.

- Gestione degli errori - Il sistema deve essere in grado di gestire eventuali errori di generazioni, notificando all'utente ove questi avvengano.

- Logging - Il sistema deve essere in grado di effettuare logging periodici riguardo l'attività dell'utente per fini di ricerca ed analisi.

3.2 Architettura

Dopo aver delineato i requisiti necessari affinché il sistema raggiunga l'obiettivo descritto nella sezione 1.2, è possibile progettare un'architettura in grado di soddisfare tali scopo. Questa proposta è suddivisa in componenti che gestiscono aspetti chiave come la visualizzazione dinamica dell'interfaccia, la generazione del nuovo aspetto e l'interazione con il database per il salvataggio delle configurazioni.

3.2.1 Definizione moduli

L'architettura proposta si presenta suddivisa in moduli interconnessi che interagiscono tra loro. In questa sezione vengono descritti i componenti principali e le loro funzionalità, con l'intento di fornire una visione sulla struttura dell'architettura.

In particolare, il sistema comprende un modulo di input, che si interfaccia con il front-end del calendario web, il quale consente di raccogliere le preferenze. Poi vi è un modulo di elaborazione che gestisce la renderizzazione dell'interfaccia e la visualizzazione dinamica. Successivamente troviamo il server che orchestra il funzionamento dei restanti moduli, ovvero un modulo di salvataggio per la conservazione delle versioni generate e delle informazioni dell'utente; infine un modulo AI che si occupa della vera e propria generazione.

L'approccio modulare non solo rende la comprensione dell'architettura più semplice ma permette anche di semplificare l'evoluzione del sistema nel corso del tempo, infatti una volta definite le interazioni è possibile sostituire o migliorare blocchi singolarmente per aggiornare il sistema.

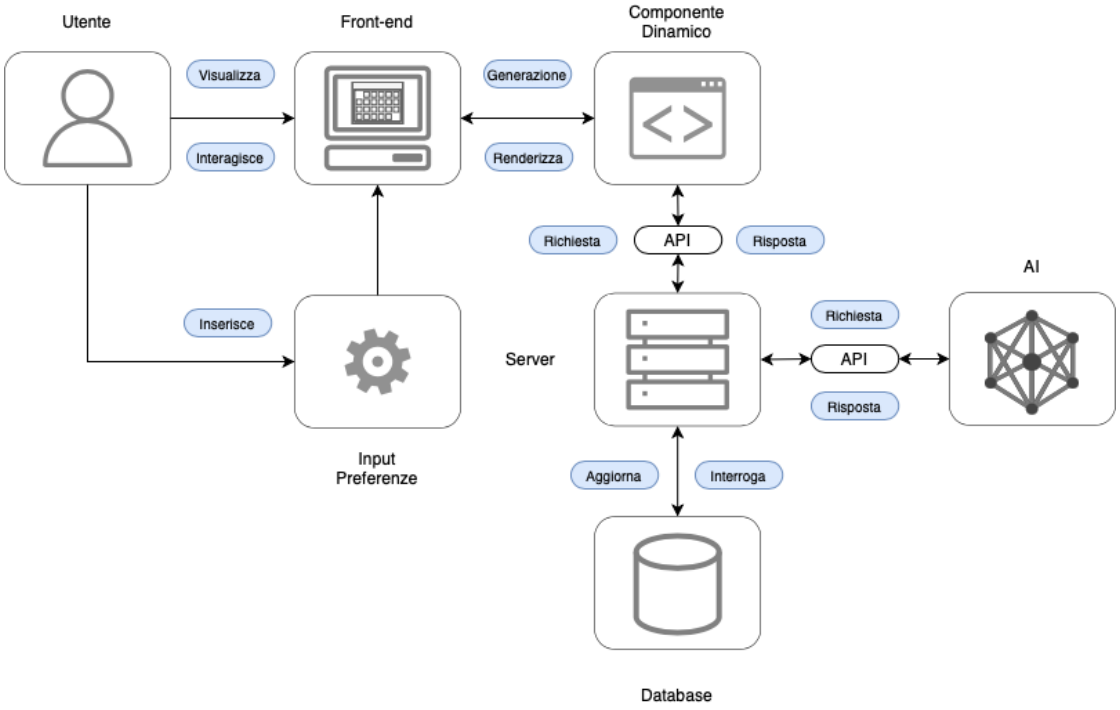

Figura 3.1: Architettura

3.3 Funzionamento

La figura 3.2 illustra un possibile modello di architettura per il sistema. In questo modello, l'utente oltre alle funzionalità di base del calendario ha a disposizione delle opzioni di configurazione avanzate. L'utente può esprimere le proprie preferenze in due modalità distinte. La prima modalità prevede una selezione di opzioni, come negli ordinari sistemi di configurazione per parametri, come la dimensione del testo e i colori di base, permettendo di personalizzare l'aspetto generale dell'interfaccia in modo diretto. La seconda modalità consente all'utente di utilizzare il linguaggio naturale per fornire indicazioni più specifiche su come desidera personalizzare l'estetica e comportamento dei singoli elementi dell'interfaccia.

Successivamente, queste informazioni vengono inviate a un modello di intelligenza artificiale. In una prima fase, questo analizza i dati ricevuti per comprendere le esigenze e le preferenze dell'utente. Dopodichè lo stesso si occupa della generazione di una nuova interfaccia, cercando di soddisfare il maggior numero possibile di richieste espresse, al fine di creare un'esperienza su misura.

Il nuovo design dell'applicazione viene quindi salvato nel database, dove sono archiviate sia l'ultima versione generata dell'interfaccia sia la versione originale. Infine, la nuova interfaccia viene estratta dal database e inviata al modulo di

visualizzazione dinamica. Questo modulo permette la visualizzazione dell'interfaccia generata, assicurando che l'utente possa immediatamente vedere e interagire con le modifiche apportate.

L'architettura consente all'utente di avere un'esperienza di utilizzo su misura.

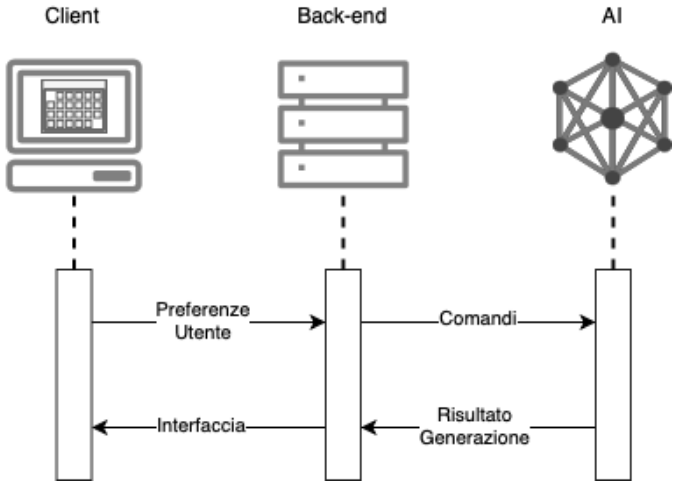

Figura 3.2: Visualizzazione funzionamento

Capitolo 4

Implementazione

In questo capitolo verrà presentata l'implementazione software dell'architettura proposta nella sezione 3.2, approfondendo gli aspetti tecnologici della sua realizzazione. Saranno analizzate le tecnologie utilizzate, i tool selezionati e le modalità di integrazione tra i vari componenti, al fine di fornire una visione completa e approfondita del processo di sviluppo. Verranno descritte le scelte tecnologiche alla base del progetto, le motivazioni per cui sono state adottate, e il loro contributo alla funzionalità complessiva dell'architettura proposta.

Nell'elenco che segue troviamo i principali componenti seguiti da un breve descrizione che verrà ampliata nei paragrafi successivi, precisamente:

- Calendario Web - Descritta nel paragrafo 4.3.1 è il Front-End del calendario web scritto in React. Possiede funzialità basiche come layout settimanale e visualizzazione di eventi e dei loro dettagli.
- Database - Analizzato nel paragrafo 4.6, il sistema consiste in un database relazionale strutturato in diverse tabelle, ognuna delle quali assolve specifiche funzioni.
- Componente Dinamico - È il componente principale dell'architettura 4.4, poiché consente la visualizzazione dinamica dell'interfaccia generata in tempo reale, senza interrompere l'esperienza d'uso. Questo elemento è fondamentale per garantire che le modifiche apportate all'interfaccia siano visualizzate in tempo reale.
- Server - Descritto nel paragrafo 4.7, il server si occupa della logica applicativa. Gestisce le richieste del client, comunica con il database e coordina la generazione dinamica dell'interfaccia.
- Tool di creazione - Analizzato nel paragrafo 4.7.2 è l'elemento responsabile per la generazione della nuova interfaccia, si occupa di formattare le preferenze

dell'utente e preparare il prompt necessario per la generazione, infine, genera l'interfaccia e la restituisce al server per poterla poi utilizzare.

- Tool di logging - È un strumento di logging 4.7.3 realizzato con lo scopo ti tracciare l'utente durante l'esecuzione, è stato necessaria la sua realizzazione per poter osservare i comportamenti di differenti utenti e studiare l'usabilità del sistema.

4.1 Tecnologie disponibili

In questa sezione verrano elencate e descritte le principali tecnologie ed principali tool necessari per lo sviluppo.

Javascript

È un linguaggio di programmazione interpretato molto leggero che supporta diversi paradigmi. Può essere impiegato sia lato "client" che "server" per la costruzione di RestfulAPI. Javascript possiede diverse capacità tra le quali la possibilità di produrre ed eseguire script dinamici, operazione fondamentale nel contesto di questa tesi. Per questo progetto è stato utilizzato sia per la realizzazione del back-end che per dei componenti del front-end. [18]

React

È una libreria open-source per la creazione di interfacce grafiche, realizzata da *Meta*, che utilizza la programmazione orientata a componenti per consentire il riutilizzo degli stessi a beneficio delle prestazioni. Inoltre i componenti sono renderizzati su di un DOM (Document Object Model) virtuale che concede di incrementare notevolmente l'efficenza durante l'utilizzo. [19]

Vite

Vite.js è un moderno tool di sviluppo per applicazioni web, progettato per essere estremamente veloce e leggero. Sfrutta l'esecuzione del codice in modalità nativa del browser durante lo sviluppo, utilizzando ES Modules, eliminando la necessità di bundling iniziale. Vit.js offre un sistema di Hot Module Replacement (HMR) che riduce significativamente i tempi di attesa durante lo sviluppo, aggiornando i moduli senza ricaricare l'intera pagina. [20]

Node

Node.js è un framework utilizzato per lo sviluppo di applicazioni web, basato sul motore V8 di Chrome, che permette l'esecuzione di JavaScript al di fuori del browser. Grazie al suo modello asincrono e event-driven, è utilizzato per sviluppare anche il lato server. In questo progetto, Node.js è stato impiegato per costruire il server, gestendo la logica backend e l'interazione con il database. [21]

OpenAI API

OpenAI è una società di ricerca e sviluppo di modelli AI, con particolare focus nei Large Language Model. Hanno sviluppato un API che consente di accedere ai loro modelli più potenti utilizzando un approccio RESTful API. In particolare la loro API consente di accedere ad un'interfaccia generica "text-in, text-out" che permette di integrare in modo semplice i loro modelli nelle appplicazioni. [22]

SQL

SQL (Structured Query Language) è un linguaggio standard utilizzato per la gestione e la manipolazione di database relazionali. SQL consente operazioni di creazione, modifica e interrogazioni del database ed è inoltre capace di eseguire query complesse per ottenere informazioni specifiche. In questo progetto, SQL è stato utilizzato per gestire il database relazionale, strutturato in più tabelle, per conservare e organizzare i dati relativi agli utenti. [23]

GitLab

GitLab è una piattaforma di gestione del codice e collaborazione basata su Git e fornisce strumenti per il controllo delle versioni, la revisione e la gestione dei progetti, facilitando il lavoro di sviluppo.[24]

4.2 Organizzazione del codice

Per gestire il codice dell'architettura basata su client e server, è stato opportuno organizzare lo sviluppo del codice in diverse cartelle e sottocartelle. Questa struttura gerarchica consente di separare chiaramente i componenti del front-end e del back-end, facilitando la navigazione e la manutenzione del codice. La struttura del progetto, quindi, si presenta suddivisa in due macro cartelle: client e server.

Client

All'interno di questo spazio sono collocati tutti i file relativi alla parte di front-end con i quali l'utente interagisce direttamente. Oltre ai file creati automaticamente come l'entry point, il CSS di default e gli assets, all'interno dellla cartella *src* è stata creata la sotto cartella Components che contiene i diversi componenti con cui l'utente interagisce. Tra questi troviamo *LoginScreen.jsx* che gestisce il processo di autenticazione degli utenti. Il *LoadingComponent.jsx* mostra un indicatore di caricamento durante le operazioni di caricamento, migliorando l'esperienza utente. Il *NavbarComponent.jsx* con *UserPreferences.jsx* consentono agli utenti di configurare le proprie preferenze.

Nella medesima cartella vi è *CalendarAI.jsx* che si occupa di definire le funzionalità del calendario fungendo da wrapper per il componente grafico del calendario che viene aggiornato dinamicamente al momento della generazione.

Infine tornando alla cartella src troviamo l'elemento più importante del lato client ovvero il *DynamicComponent.jsx*, questo è il componente che valuta l'output del modello e produce la nuova interfaccia.

Server

Nella cartella del back-end troviamo come primo elemento il *server.js* che è il coordinatore tra i vari processi di generazione e salvataggio delle informazioni ed è il cuore della logica dell'applicazione.

Nel medesima cartella vi è la sottocartella riferita al database; al suo interno troviamo il file *database.db* accompagnato da altri file che contengono codice SQL per svolgere operazioni rapide sulla base dati durante lo sviluppo.

Successivamente troviamo la sottocartella tools che al suo interno contiene il file *loggerTool.js* che viene utilizzato per loggare le operazioni dell'utente all'interno della cartella logs. Nella medesima cartella vi è contenuto l'elemento principale del server ovvero il *creatorTool.js*, che si occupa di generare le interfaccie.

4.3 Sviluppare in React

Negli ultimi anni si è assistito ad un cambiamento nello sviluppo web, si è passati dall'avere singole pagine statiche all'avvento di web app sempre più dinamiche ed interattive. Nonostante non esista una separazione netta fra queste, la peculiarità delle applicazioni web è la loro capacità di permettere all'utente di compiere delle azioni e ricevere in cambio una risposta.

In passato un sito web era principalmente costituito da una pagina HTML che veniva richiesta dal client e fornita dal server, lo svantaggio risiedeva nel fatto che per ogni cambiamento su essa era necessario richiedere nuovamente l'intera pagina

al server. Al contrario nelle moderne applicazioni web viene caricata la prima pagina e il client richiede di volta in volta gli elementi necessari per aggiornarla senza ricaricarla per intero. Appare quindi ovvio come questo tipo di paradigma forzi una separazione tra client e sever che va a beneficio dei tempi e della difficoltà di implementazione.

In particolare React [19] è un framework per lo sviluppo di moderne interfacce grafiche che ha modificato la visione originaria di sviluppo web sostituendola per un component-based, di seguito si trova un esempio di un componente che stampa "Hello World!" in una pagina web.

```
1    function ExampleComponent() {
2        return <h2>Hello World!</h2>;
3    }
```

Al termine dello sviluppo un'applicazione in React, possiede una struttura più complessa di quella di un ordinario sito web in quanto deve passare attraverso un processo di build prima di poter essere renderizzata nel browser.

4.3.1 Calendario Web

Dopo aver definito gli aspetti che coinvologno la realizzazione di un'applicazione web con React si può procedere con lo sviluppo della versione base del calendario web. A seguire l'immagine che l'utente visualizza al primo accesso.

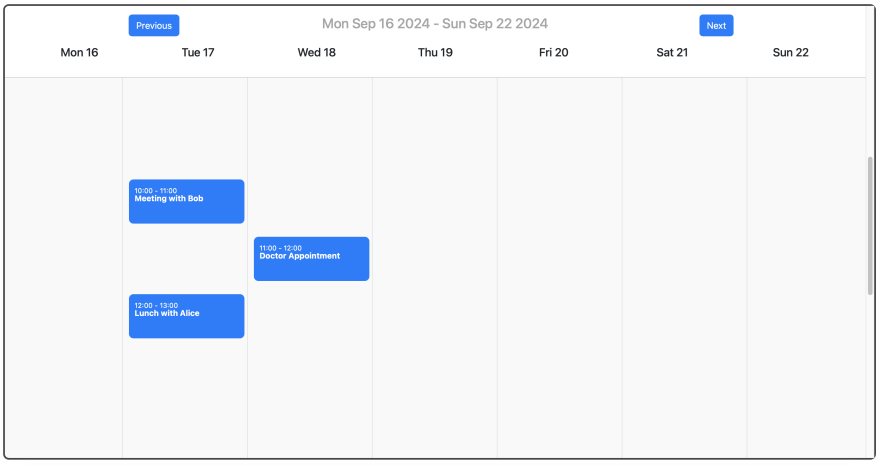

Figura 4.1: Versione base calendario web

Seguendo gli obiettivi funzionali delineati nel paragrafo 3.1.1 è stato implementato il calendario mediante l'utilizzo di React e React-Boostrap per la gestione

dell'interfaccia e del layout. Dalla libreria React-Boostrap vengono utilizzati diversi componenti come *Container*, *Row*, *Column*, *Button*, *Badge* e *Modal* per strutturare il layout, lo style invece viene definito inline attravreso la variabile *styles* che contiene un dizionario di stili associato ai vari componenti presenti. Questa soluzione tornerà utile per il processo di generazione.

La struttura principale del calendario, come mostra la Figura 4.1, si basa sulla visualizzazione settimanale degli eventi. Ogni giorno è suddiviso in fasce orarie all'interno delle quali vengono visualizzati gli eventi.

Gli eventi, passati come proprietà attraverso l'oggetto events, vengono filtrati e visualizzati dinamicamente all'interno dei rispettivi giorni e fasce orarie. La funzione *renderEvents* si occupa di gestire questo processo, assicurando che ogni evento venga collocato correttamente in base al suo orario di inizio e fine.

```
1  const renderEvents = (day) => {
2  const dayStart = new Date(day.setHours(0, 0, 0, 0));
3  return events?.filter(event => event.start >= dayStart && event.
       start <= new Date(day.setHours(23, 59, 59, 999)));
4  };
5
6  const renderDayView = (date, index) => {
7  const hours = Array.from({ length: 24 }, (_, i) => new Date(date.
       setHours(i, 0, 0, 0)));
8  const dayEvents = renderEvents(date);
9  const isCurrentDay = index === currentDate.getDay() - 1 &&
       currentDate.getDate() === new Date().getDate();
10 return (
11   <div key={index} style={styles.calendarDay}>
12     {hours.map((hour, idx) => renderDayEvents({ hourEvents:
       dayEvents?.filter(event => event.start >= hour && event.start <
        new Date(hour.getTime() + 60 * 60 * 1000)), idx, currentHour:
       hour.getHours(), currentTime, isCurrentDay }))}
13   </div>
14 );
15 };
```

La funzione *renderDayView* gestisce la visualizzazione di un singolo giorno suddiviso per fasce orarie. Riceve come parametri la data del giorno e un indice che indica il giorno nella settimana. All'interno della funzione viene creato un array di 24 elementi che rappresentano ciascuna ora del giorno, con oggetti *Date* che indicano l'ora esatta. Per ogni giorno, viene eseguito un filtro sugli eventi tramite la funzione *renderEvents*, in modo da mostrare solo quelli relativi al giorno. Infine, per ogni ora del giorno, vengono visualizzati gli eventi relativi, utilizzando la funzione *renderDayEvents*, che mostra un badge con titolo, orari di inizio e di fine dell'evento.

La funzione *renderWeekView* visualizza l'agenda dell'intera settimana. La funzione riceve in ingresso una data e calcolando l'inizio della settimana tramite la funzione *startOfWeek*, che restituisce il lunedì della settimana corrente. Successivamente, genera un array di sette giorni, a partire da quella data, incrementandolo progressivamente di un giorno. Per ciascun giorno della settimana, viene chiamata la funzione *renderDayView*, che visualizza le ore e gli eventi associati a quel giorno. Ogni giorno viene visualizzato come una colonna, creando così una disposizione settimanale ordinata, dove ciascun giorno contiene le sue fasce orarie e gli eventi corrispondenti.

```
const renderWeekView = (date) => {
const start = startOfWeek(date);
return (
  <Col xs={12} style={{ padding: "0" }}>
    <Row style={{ display: "flex" }}>
      {Array.from({ length: 7 }, (_, i) => new Date(start.getTime() + i * 24 * 60 * 60 * 1000)).map((day, index) => (
        <Col key={index} md={4} style={styles.calendarDayWrapper} onClick={() => handleDayClick(day)}>
          {renderDayView(day, index)}
        </Col>
      ))}
    </Row>
  </Col>
);
};
```

Si può quindi osservare che la funzione *renderDayView* si occupi della gestione del singolo giorno e degli eventi in agenda, mentre la *renderWeekView* integra tutte le singole visualizzazioni giornaliere di una settimana.

```
<Modal show={showModal} onHide={handleCloseModal}>
  <Modal.Header closeButton><Modal.Title>{selectedEvent.title}</Modal.Title></Modal.Header>
  <Modal.Body>
    <p><strong>Start:</strong> {selectedEvent.start.toLocaleString()}</p>
    <p><strong>End:</strong> {selectedEvent.end.toLocaleString()}</p>
    <p><strong>Description:</strong> {selectedEvent.description}</p>
  </Modal.Body>
</Modal>
```

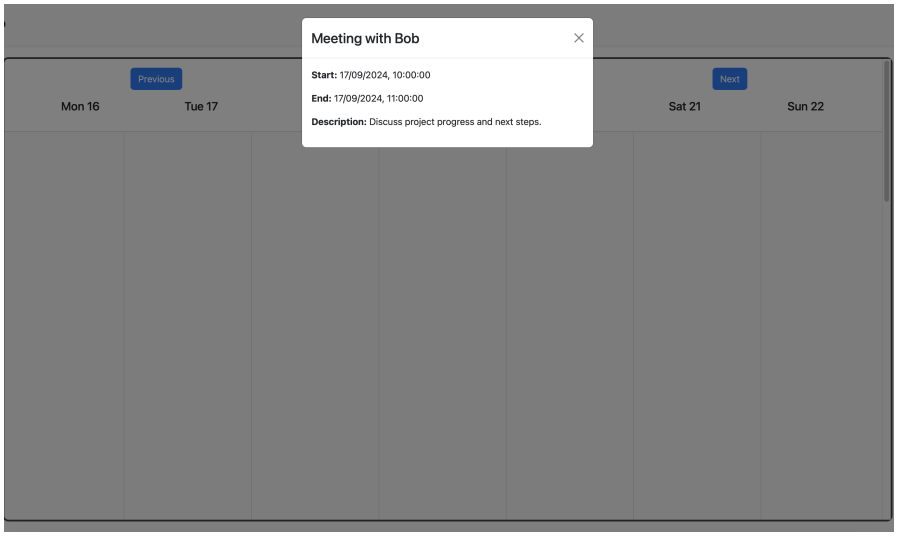

Figura 4.2: Visualizzazione dettagli evento

Il *modal* visualizza i dettagli di un evento selezionato come mostrato in figura 4.2. Quando un utente clicca su un evento, si attiva la funzione *handleEventClick*, che apre la finestra di dialogo passando l'evento selezionato come parametro. Il modale stesso è reso visibile quando la variabile showModal è impostata su true. All'interno del *modal*, la proprietà *selectedEvent* viene utilizzata per mostrare i dettagli dell'evento. Questi dettagli includono il titolo dell'evento, l'ora di inizio/fine e una descrizione più estesa.

L'utente ha la possibilità di chiudere il modal utilizzando il pulsante di chiusura, che invoca la funzione *handleCloseModal*, responsabile di nasconderlo.

4.4 Componente Dinamico

Il componente centrale dell'architettura, denominato *DynamicComponent*, svolge un ruolo fondamentale nel garantire la dinamicità dell'interfaccia del calendario. Si tratta di un componente React, progettato per ricevere codice sotto forma di stringa di testo, interpretarla, e produrre in modo dinamico un nuovo componente a partire dal codice ricevuto. Questo approccio consente di adattare l'interfaccia in tempo reale senza l'intervento diretto sul codice. L'utilizzo di questo componente è analogo a quello dei normali componenti React, come verrà illustrato a seguire. In questo esempio, vengono passati due parametri: il primo è l'oggetto user utilizzato per identificare l'utente, mentre il secondo, denominato props, contiene le funzioni di base che il calendario è in grado di eseguire. Questo approccio consente al

componente di generare unicamente l'interfaccia utente, mantenendo inalterate le funzionalità già presenti.

```
1  /*Esempio di utilizzo del DynamicCompnent*/
2
3  const handlePrevDay = ({day= 7}) => {
4      setCurrentDate(new Date(currentDate.setDate(currentDate.
       getDate() - days));
5      };
6  const handlePrevDay = ({days= 7}) => {
7      setCurrentDate(new Date(currentDate.setDate(currentDate.
       getDate() - days)));
8  };
9  const props = {
10     handlePrevDay,
11     handlePrevDay
12 }
13
14 return (
15     <DynamicComponent user={user} props={props}/>
16 )
```

Nella prossima sezione verrà illustrato dettagliatamente questo componente, analizzando le sue funzioni e il flusso di esecuzione. In particolare si osserveranno i meccanismi che consentono di caricare e compilare codice dinamicamente e come viene gestita la renderizzazione del nuovo componente generato.

La prima funzione eseguita è denominata *loadComponent* che viene invocata all'interno di un *useEffect* richiamata quando il componente viene montato, in questo modo il processo di caricamento e compilazione avviene una volta che il componente React è pronto. La *loadComponent* chiama a sua volta *loadAndCompileComponent*, che esegue la logica vera e propria di richiesta e compilazione del codice. Durante questo processo, il codice del componente viene recuperato da un'API remota utilizzando una chiamata *fetch* che utilizza l'ID utente per identificare il codice dall'interno del database. Il codice viene quindi pre-processato con *preprocessCode*, che ripulisce il codice dalle istruzioni di importazione.

Successivamente il codice viene compilato utilizzato babel, tale processo lo rende utilizzabile da React e quindi renderizzabile. Prima di poter essere utilizzato però il codice eseguibile viene inglobato in una funzione generata dinamicamente la quale riceve come parametri le dipendenze di cui il codice necessita. Infine il componente viene ritornato ed il suo valore salvato in uno stato denominato *RetrivedData*.

Se il caricamento o la compilazione falliscono, lo stato RetrievedData viene impostato su un componente di fallback che visualizza un messaggio di errore, garantendo un'esperienza utente fluida anche in caso di problemi.

L'implementazione del *DynamicComponent* è visibile nel codice sottostante:

```
1  import React, { useState, useEffect, useRef } from 'react';
2  import { Row, Col, Button, Badge, Modal, Card, Container,
       ListGroup, Table,  Carousel } from 'react-bootstrap';
3  import * as babel from '@babel/standalone';
4  import { useAuth } from './Components/AuthProvider';
5  import LoadingComponent from './Components/LoadingComponent';
6
7
8  const preprocessCode = (codeString) => {
9      /*Codice di pre-processing*/
10 };
11   const loadAndCompileComponent = async (user,existingDependencies
       ) => {
12     const userId = user.user_id
13     const dependencyNames = Object.keys(existingDependencies);
14     const dependencyValues = dependencyNames.map(name =>
       existingDependencies[name]);
15     try {
16       const response = await fetch(
17         `http://localhost:3001/api/component-code?userId=${userId
       }`
18       );
19       if (!response.ok) {
20           throw new Error(`Error: ${response.statusText}`);
21       }
22       const data = await response.json();
23
24       const scriptContent = data.code
25       const { imports, cleanedCode } = preprocessCode(
       scriptContent);
26
27       const compiledCode = babel.transform(cleanedCode, {
28         presets: ['react', ['es2017', { modules: false }]],
29       }).code;
30
31       const func = new Function('props', ...dependencyNames, `
32         return ${compiledCode}
33       `);
34
35       const EvaluatedComponent = (props) => {
36         const Component = func(props, ...dependencyValues);
37         return <Component {...props} />;
38       };
39
40       return EvaluatedComponent;
41
42     } catch (error) {
43       console.error(`Error loading component`, error);
```

```
44        return null;
45      }
46    };
47
48    const DynamicComponent = ({user, props }) => {
49      const [RetrievedData, setRetrievedData] = useState(null);
50
51      const dependencies = {/*React-Boostrap Dependencies*/};
52
53      const loadComponent = async () => {
54        try{
55          const mainComponent = await loadAndCompileComponent(user,
     dependencies);
56          if (mainComponent) {
57            setRetrievedData(() => mainComponent);
58          }
59
60        }catch{
61          setRetrievedData(()=> <><div>Fail!</div></>)
62        }
63
64      };
65
66      useEffect(() => {
67        loadComponent();
68      }, []);
69
70      return (
71        <>
72          {RetrievedData ? (
73            <RetrievedData {...props} />
74          ) : (
75            <div>Oh, Something went wrong!</div>
76          )}
77        </>w23
78      );
79    };
80
81  export default DynamicComponent;
```

4.5 Input preferenze

Durante la realizzazione della tesi uno degli aspetti più importanti di cui occuparsi, oltre alla realizzazione dell'infrastruttura a supporto, è stato la progettazione delle modalità di espressione delle preferenze. Essendo il modello scelto un LLM e

volendo sfruttare le sue caratteristiche di "comprensione" del linguaggio naturale si è quindi optato per un tipo di input basato sulla descrizione da parte dell'utente. Tuttavia, un input libero non è stato sufficiente poichè rimaneva eccessivamente generico e astratto rispetto all'utilizzo all'interno dell'interfaccia. La prima versione dell'interfaccia di personalizzazione si presentava come in figura 4.3.

Va detto che questo approccio risultava troppo libero e l'utente poteva trovare difficoltà nel gestire gli input, in quanto come si può esservare è presente un solo campo denominato *Additional Specification* per descrivere le modifiche da effettuare per personalizzare l'interfaccia.

Figura 4.3: Modalità espressione preferenze: prima versione

Per far fronte a questo problema è stato necessario strutturare l'input in modo più consistente, portando alla seconda versione rappresentata in figura 4.4. Si può quindi osservare come si sia trasformata la schermata. In questa versione l'input delle preferenze avviene direttamente all'interno della schermata del calendario attraverso un menù a scomparsa. In particolare i settaggi statici, non visibili nell'immagine perchè nascosti, sono rimasti i medesimi; al contrario per ciò che riguarda la configurazione dinamica dell'interfaccia sono stati aggiunti e modificati i campi *Goal*, *Structure*, *Style*. È inoltre presente un campo *AI imitation* che riceve l'url di un immagine e produce un'interfaccia simile sovrascrivendo le precedenti opzioni.

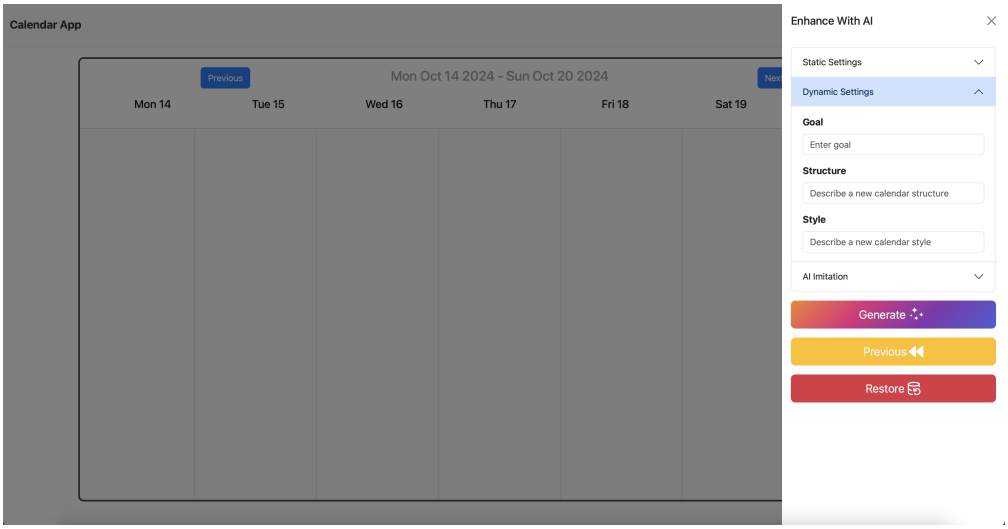

Figura 4.4: Modalità espressione preferenze: seconda versione

In virtù dei miglioramenti, avendo raggruppato nella medesima schermata la configurazione, la generazione ed il ripristino è l'applicazione stessa a rendendere l'utilizzo più semplice. Ciò nonstante, questa modalità rimaneva ancora eccessivamente astratta e non strutturata a sufficienza, infatti i tre campi necessari potevano rendere difficile all'utente produrre una descrizione dettagliata dell'aspetto desiderato di ogni componente e del funzionamento atteso e allo stesso rendere più complesso per il modello soddisfare le richieste.

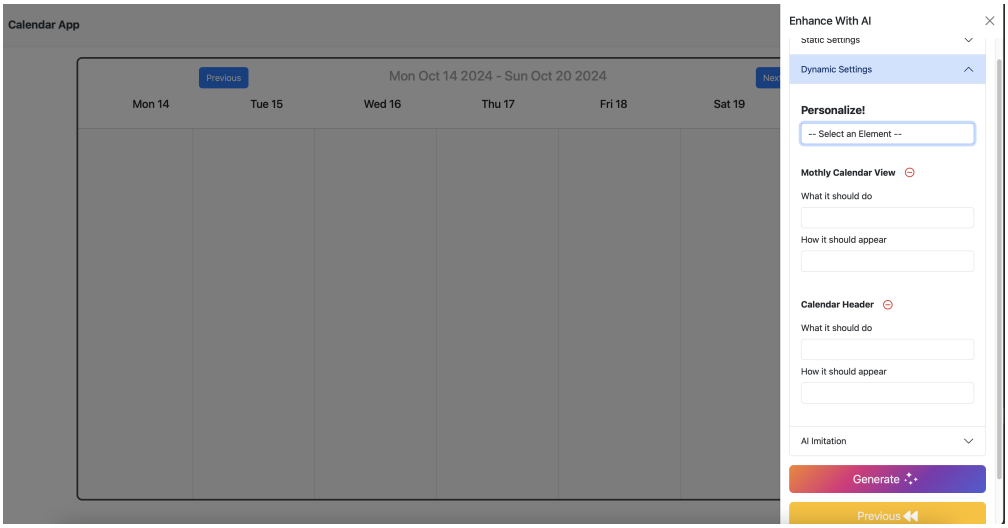

Figura 4.5: Modalità espressione preferenze: versione finale

Per queste ragioni è stato necessario rendere più definito l'input senza però limitare l'utente nell'esprimere le proprie preferenze. La soluzione è visibile in figura 4.5, nella quale sono state mantenute invariate le caretteristiche dei menù a scomparsa e della posizione dello stesso. I tre campi di inserimento sono stati rimossi ed infatti è stato implementato un selettore che permette di scegliere su quale componente attuare la modifica.

Dopo la selezione compaiono i campi relativi a quel componente che sono *What it should do* e *How it should appear* ovvero cosa vuoi che faccia e come vuoi che appaia. Questo tipo di input più strutturato aiuta l'utente a capire esattamente su quale componente sta eseguendo le modifiche e come vuole personalizzarlo e allo stesso tempo aiuta il modello avendo un input più schematico.

4.6 Database

Nella realizzazione dei sistemi moderni la scelta della tipologia di database svolge un ruolo importante, perchè definisce il modo in cui i dati vengono salvati, le tipolgie di accesso e a seconda del tipo di dato la scelta può impattare sulle prestazioni. Esistono diverse categorie di database come quelli relazionali MySQL o PostgreSQL e i database NoSQL, che gestiscono dati non strutturati o semi-strutturati, ideali per applicazioni con grandi quantità di dati dinamici.

Per lo sviluppo dell'architettura proposta, è stato necessario integrare un database per garantire la persistenza delle informazioni degli utenti e delle interfacce personalizzate. Questa scelta si è resa indispensabile per diversi motivi.

Innanzitutto, un database consente di memorizzare in modo strutturato le informazioni di ciascun utente offrendo la possibilità di conservare, ripristinare e richiamare rapidamente i vari codici delle interfacce personalizzate, migliorando così l'esperienza utente.

4.6.1 SQL Database

La scelta della tipologia del database per l'architettura proposta è ricaduta sulla categoria dei relazionali. Questi database utilizzano tabelle strutturate e relazioni tra dati, risultando ideali per gestire in maniera efficiente la tipologia di informazioni utilizzate. La struttura relazionale del database nonchè il ridotto numero di dati da salvare rendono questo tipo di base dati adatta allo scopo. Infatti attraverso l'utilizzo delle chiavi primarie le operazioni di aggiornamento, recupero e manipolazione dei dati risultano semplici.

Nonostante i database relazionali permettano l'uso di query più complesse, in questo caso non è stato necessario ricorrere a tali funzionalità. L'architettura proposta si concentra principalmente sulla gestione di dati strutturati e relativamente semplici da trattare.

4.6.2 Descrizione database

Il database all'interno dell'architettura è formato da due tabelle principali come mostrate in figura 4.6. La prima tabella prende il nome di *user*, è destinata a preservare le informazioni degli utenti quali email, password necessari per il login e lo *userId* un numero che identifica univocamente un utente pertanto funge anche da chiave primaria. Successivamente troviamo la seconda tabella, questa è composto da *comnponentId* che identifica univocamente il componente perciò è anche la chiave primaria della tabella. Poi vi è lo *userId* che è la chiave esterna che crea la relazione "One-To-One" con l'utente in modo che gli possa essere associato il proprio componente.

Il campo *code* contiene il codice interfaccia grafica sotto forma di testo, mentre il campo *prevCode* contiene il codice della precedente interfaccia garantendo quindi la possibilità di ripristino dell'ultima versione.

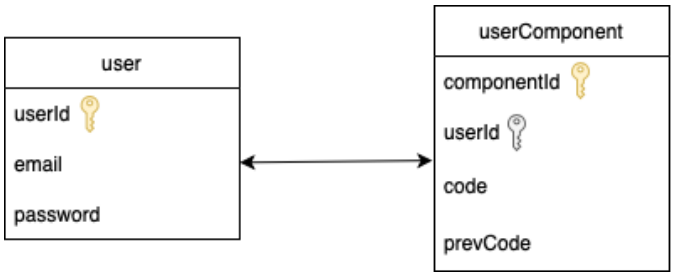

Figura 4.6: Schema tabelle database

4.7 Back-End

Dopo aver esaminato gli aspetti che riguardano la parte di interfaccia utente e la gestione dei dati è necessario affrontare l'implementazione della logica del back-end, ovvero la parte che si occupa di orchestrare i dati e le interazioni degli utenti. In particolare nei paragrafi successivi si analizzerà la struttura del server e alcuni dei principali elementi che lo compongono.

4.7.1 Server

Il server è il cuore del funzionamento, nel contesto di questa tesi è fondamentale per garantire coordinazione tra le richieste dell'utente e la realizzazione di queste occupandosi,quindi dell'integrazione delle API e delle integrazioni di queste nel front-end. Come accennato in precedenza il server presenta un'architettura basata su API RESTfull sviluppata attraverso node, ogni end point è stato progettato

per assolvere compiti provenienti dall'interfaccia come login, logout, generazione interfaccia, reset, torna all'ultima interfaccia e caricamento.

Come prima configurazione è stato abilitato CORS (Cross-Origin Resource Sharing) per la ricezione di richieste provenienti dal front-end in React, successivamente si è configurata la connessione al database ed infine sono state definite le politiche di autenticazione. Di seguito si trovano le descrizioni dei principali API e la loro implementazione.

Login - POST `/api/login`
Autentica l'utente utilizzando email e password tramite Passport, in caso positivo crea una sessione e ritorna i dettagli dell'utente autenticato.

Sessione Corrente - GET `/api/sessions/current`
Restitusice i dettagli dell'utente autenticato nella sessione corrente, in caso negativo ritorna un errore.

Logout API - DELETE `/api/sessions/current`
Disconnette l'utente dalla sessione corrente.

Componente Precedente - POST `/api/previous-component`
Ripristina all'interno del database il codice dell'interfaccia precedente per un utente specifico, utilizzando l'ID dell'utente fornito nella query. Se l'ID dell'utente non è fornito o se il componente non è trovato, restituisce un errore.

Reset Componente - POST `/api/reset-componente`
Ripristina nel database il codice dell'interfaccia attuale con quello di default per un utente specifico, utilizzando l'ID dell'utente fornito nella query. Se l'ID dell'utente non è fornito o se il componente non è trovato, restituisce un errore.

Codice Componente - GET `/api/component-code`
Recupera il codice dell'interfaccia associato a un utente specifico, utilizzando l'ID dell'utente fornito nella query. Se l'ID dell'utente non è fornito o se il componente non è trovato, restituisce un errore.

Aggiorna Personalizzazione - POST `update-customization`
Aggiorna il codice dell'interfaccia per un utente specifico, utilizzando l'ID dell'utente fornito nella query e i dati forniti nel corpo della richiesta. Se l'ID dell'utente non è fornito, restituisce un errore. In particolare questo endpoint chiama una funzione denominata *generateComponent* che si occupa produrre il nuovo codice dell'interfaccia. Recupera il codice dell'interfaccia corrente dall'utente dal database, genera un file JSX temporaneo e vi inserisce il nuovo codice utilizzando una funzione esterna *creatorTool*, infine aggiorna il codice dell'interfaccia dell'utente con il nuovo codice generato all'interno del file e salva il codice precedente.

4.7.2 Modulo di AI

Per semplificare la realizzazione di nuove interfacce è stato costruito uno strumento su misura che si interfacciasse con il server e il modello AI permettendo una comunicazione più agevole e che semplificasse l'utilizzo. Questo modulo formatta l'input utente, sfrutta l'API di OpenAI[22] per richiamare il modello AI per interpretare i requisiti espressi dall'utente e produrre il codice React della nuova interfaccia.

Il processo ha inizio dopo che l'utente ha avviato la generazione, le sue preferenze sono inviate al back-end tramite l'API */update-customization*, viene quindi invocata la funzione *creatorTool* che riceve come parametri il codice dell'attuale interfaccia le preferenze dell'utente. Successivamente viene invocata la *generateReactComponent* alla quale vengono passati i medesimi parametri, all'interno di questa avviene la chiamata all'API di OpenAI che permette di accedere a differenti modelli LLMs. Tale funzione prende in input quattro parametri: il nome del modello scelto, in questo caso l'ultimo disponibile in questo momento *gpt-4o*, seguentemente il prompt che viene costruito attraverso la funzione *getMessages*, il numero massimo di token ed infine la temperatura che regola la casualità dell'output.

Per migliorare la qualità della generazione e definirne le caratteristiche è stato necessario effettuare prompt engineering, questa è una tecnica che si occupa si progettare e ottimizzare i prompt da inviare al modello, per ottenere risposte più soddisfacenti. L'obiettivo è quello di formulare un testo di input che influenzi il modello per massimizzare la qualità generativa. In questa tesi si è reso necessario applicare due prompt: prompt di sistema e prompt generativo.

Il primo, di cui si riporta il contenuto di seguito, ha lo scopo di definire la tipologia delle interazioni e quindi gli input attesi e i limiti dell'output, utilizzando regole che il modello dovrà rispettare. Il secondo invece contiene nell'intestazione l'attuale interfaccia da modificare e successivamente le preferenze opportunamente formattate.

```
1  "Instructions for the components:
2  You are a React components creator who makes extremely good
3  looking React components inspired by the one provided to.
4  The code you will create will be used in Calendar web app
5  application
6
7  0- DO NOT Start with three backsticks javascript as comment
8  1- DO NOT OUTPUT THE CODE AS A COMMENT or USE 'javascript' or
9  'JSX' COMMENTS.
10 2- MUST USE React and React-boostrap libraries.
11 3- USE LESS TOKENS AS POSSIBLE.
12 4- COLOR PALETTE primary ${data.primaryColor} and secondary
13 ${data.secondaryColor}.
14 5- TEXT SIZE MUST BE CONSTRAINED BETWEEN
15 ${data.textSizeRange[0]} and secondary ${data.textSizeRange[1]}
```

```
6- TEXT FONT ${data.fontFamily}
```

La risposta contenente il codice dell'interfaccia, prima di essere inviata viene scritta all'interno di un file temporaneo per poter essere successivamente salvata nel database, passa attraverso due funzioni di formattazione: *removeTripleBackticks* per rimuovere i backsticks (` `) che il modello può aggiungere all'inizio del codice e *removeExportDefaultCalendar* per rimuovere l'export alla fine del codice. Di seguito si può osservare il codice del processo descritto precedentemente.

```
async function generateReactComponent(currentCode,data) {

  try {
   const response = await openai.chat.completions.create({
        messages: getMessages(currentCode,data),
        model: 'gpt-4o',
        max_tokens: 2100,
        temperature: 0.8
    });

    const code = removeTripleBackticks(response.choices[0].message
   .content);
    const codeWithoutExport = removeExportDefaultCalendar(code)

    return codeWithoutExport;
  } catch (error) {
    throw new Error(`Error generating code: ${error.message}`);
  }
}

function saveCodeToFile(filename, code) {
  fs.writeFileSync(`./temp/${filename}.jsx`, code, (err) => {
    if (err) {
      console.error(`Error saving file: ${err}`);
    } else {
      console.log(`Code saved to ${filename}`);
    }
  });
}

async function creatorTool(currentCode,data) {
    const newUUID = uuidv4();
    try {
        const code = await generateReactComponent(currentCode,data
   );
        saveCodeToFile(newUUID, code);
        return newUUID
    } catch (error) {
```

```
37          console.error(error.message);
38          return error
39      }
40  }
```

4.7.3 Strumento di logging

Al fine di ottenere una comprensione più approfondita e accurata dell'interazione tra gli utenti e l'applicazione, è stato necessario introdurre un sistema di logging personalizzato.

Questo sistema ha permesso di monitorare e tracciare in modo preciso il comportamento degli utenti durante l'utilizzo dell'app, raccogliendo dati fondamentali per l'analisi successiva. Sebbene esistano già numerosi tool e piattaforme di terze parti dedicati al monitoraggio delle interazioni utente, si è deciso di sviluppare una soluzione ad hoc. La scelta di una soluzione su misura ha permesso di ottenere log specifici, in grado di riflettere esattamente le azioni chiave che gli utenti compiono all'interno dell'applicazione. Questo permette di adattare il sistema di logging alle caratteristiche uniche dell'applicazione stessa, rendendolo uno strumento fondamentale per lo studio delle dinamiche utente.

La raccolta di dati specifici, infatti, ha consentito di concentrarsi su particolari aspetti del comportamento dell'utente, come le interazioni con specifici elementi dell'interfaccia o i flussi di navigazione, rendendo possibile un'analisi più mirata e approfondita.

L'implementazione del logger è la seguente.

```
1       const fs = require('fs')
2
3   function logToFile(user,message){
4       const logStream = fs.createWriteStream(`./logs/logs_user${user
        }.txt`, {flags: 'a'})
5       logStream.write(`${message}\n`)
6       logStream.end()
7   }
8   const logger = {
9       start: (user,message) => logToFile(user,`${new Date().
        toUTCString()} [START] ${message}`),
10      info: (user,message) => logToFile(user,`${new Date().
        toUTCString()} [INFO] ${message}`),
11      warn: (user,message) => logToFile(user,`${new Date().
        toUTCString()} [WARN] ${message}`),
12      error: (user,message) => logToFile(user,`${new Date().
        toUTCString()} [ERROR] ${message}`),
13      end: (user,message) => logToFile(user,`${new Date().
        toUTCString()} [END] ${message}`),
```

```
14 }
```

Capitolo 5

Studio utente

A completamento della tesi si è deciso di implementare un test di valutazione per verificare l'usabilità dell'architettura proposta e raccogliere informazioni utili sul suo funzionamento.

Effettuare dei test di usabilità permette di valutare qualsivoglia software, applicazione e sito attraverso l'esperienza diretta dell'utente. Per far ciò si realizzano degli esperimenti durante i quali vengono posti quesiti e dati compiti da svolgere ai candidati, nel corso dell'esecuzione possono essere registrate le azioni degli utenti, le loro difficoltà e reazioni. Gli esperimenti possono assumere diversi formati che variano da questionari, osservazioni dirette sino alla misurazione di metriche specifiche sull'utilizzo. Come riportato nel libro *Human-Computer Interaction* [1] possiamo distinguere diversi obiettivi oggetto di questa valutazione:

1. **Valutare la funzionalità del sistema**
   Il sistema deve essere allineato con le aspettative degli utenti, ovvero non solo deve fornire le funzionalità promesse ma deve anche renderle fruibili in modo semplice.

2. **Valutare l'esperienza utente**
   Oltre alla funzionalità è necessario concentrarsi sulla facilità di apprendimento, sull'usabilità e sul grado di soddisfazione.

3. **Individuare problemi di design**
   Infine l'obiettivo ultimo di uno studio utente è quello di trovare specifici difetti di progettazione che portano a risultati imprevisti o confusione per l'utente.

La costruzione dello studio è stata suddivisa in 3 fasi descritte in modo più approfondito nei paragrafi successivi. Il primo step è la pianificazione, in questa fase si definiscono la tipologia di utenti che prenderanno parte all'esperimento, l'obiettivo del test e le sue modalità. Nella seconda fase, ovvero l'esecuzione, si ha

l'effettiva attuazione dell'esperimento e si registrano i relativi dati. Infine vi è la terza ed ultima fase che consiste nell'analisi da cui si estraggono i dati e si valutano le prestazioni del sistema.

5.1 Pianificazione

In questa prima fase come detto in precedenza viene progettato lo studio utente, definendo il target di utenti e le modalità di esecuzione.

Partecipanti

Per la scelta dei partecipanti all'esperimento si è cercato di variare il più possibile le caratteristiche degli utilizzatori non essendo il sistema destinato ad una singola categoria. Il numero di partecipanti selezionato è 18, poichè in questo caso studio si è reso necessario disporre di un campione ampio per rendere l'esperimento rilevante statisticamente.

Script

Per semplificare l'esecuzione dello studio utente è stato necessario la realizzazione di uno *script*. Questo ha un duplice ruolo, permette di avere sempre a disposizione le istruzioni step-by-step e allo stesso tempo consente di rimanere costanti le varie interviste. Lo *script* suddiviso in sezioni, partendo da quella iniziale che introduce gli utenti all'esperimento ed al suo scopo. Viene quindi spiegato loro che l'oggetto della valutazione è il sistema e non le loro abilità perciò non esistono veri e propri errori che si possono commettere durante l'utilizzo.

Successivamente viene anticipato loro che avrebbero dovuto effettuare dei compiti, prima di ciò è stato richiesto di rispondere sia a domande a risposta aperta sia a domande a risposta chiusa. Questi quesiti sono consultabili nell'appendice A sono di carettere generico inizialmente e scendono progressivamente nel dettaglio del caso studio. Questo permette di conoscere parzialmente il background dell'intervistato consentendo di svolgere analisi più dettagliate sull'utilizzo del sistema.

Attività

L'esperimento pratico è articolato in 2 attività principali che l'utente deve svolgere, entrambe consistono nella replica, partendo dalla versione base in figura 5.3, l'immagine fornita utilizzando il prototipo dell'architettura. Questo consente di misurare sia come l'utente interagisce con l'interfaccia, sia come il sistema interpreta le preferenze e produce nuovi risultati, analizzando come un utente si comporta

durante il normale utilizzo. Sono state rese disponibili, allo scopo, 10 generazioni e 20 minuti per immagine al termine dei quali verrà salvato lo stato attuale e l'utente potrà loggarsi con un secondo account per la seconda attività. Durante l'utilizzo l'intervistato può ripristanare da zero o tornare indietro di uno step alla versione precedente un numero illimitato di volte.

Per la prima attività viene fornita l'immagine rappresentata in figura 5.1, che mostra una versione del calendario con una visione giornaliera compatta dove gli eventi sono rappresentati in una lista verticale. Inoltre anche i bottoni per navigare hanno cambiato forma, colore e posizione. Questa versione viene fornita per prima perchè ritenuta più semplice con lo scopo di far approcciare l'utente al programma.

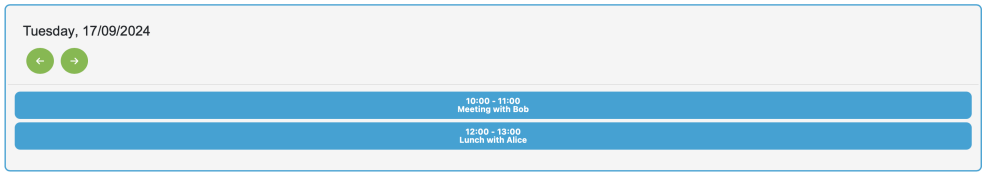

Figura 5.1: Calendario da replicare semplice

La seconda attività si basa sull'immagine in figura 5.2, come si può vedere in questo caso è presente un componente aggiuntivo inserito alla sinistra del calendario che visualizza gli eventi della settimana in modo sintetico, inoltre anche la data ed i bottoni per navigare tra le settimane sono stati modificati rendendoli circolari.

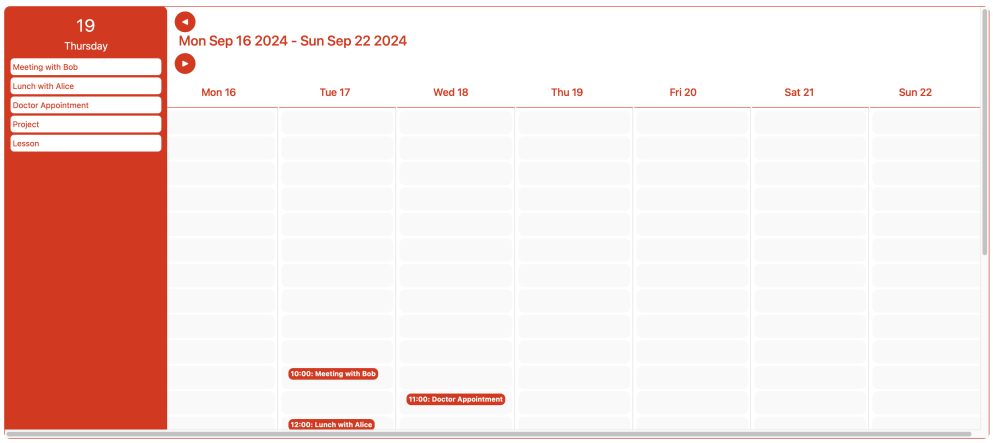

Figura 5.2: Calendario da replicare complesso

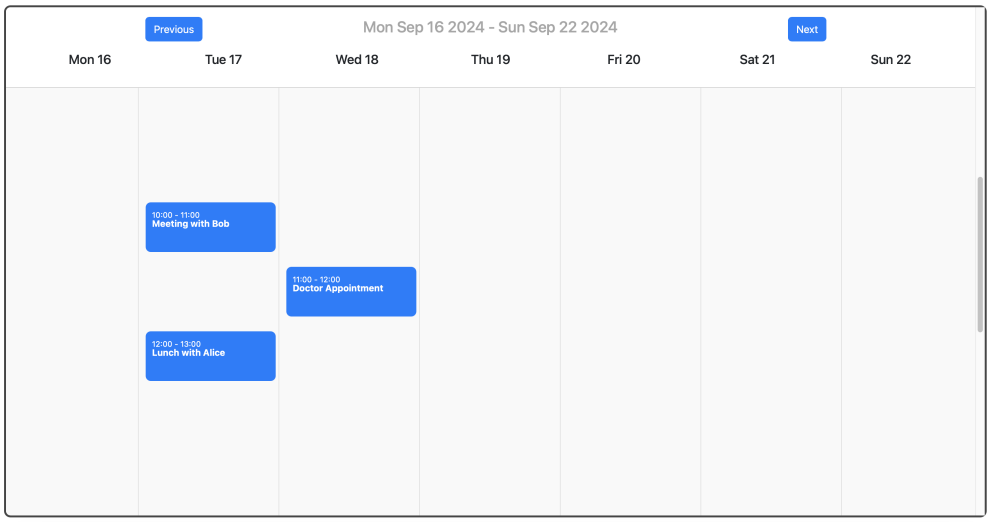

Figura 5.3: Calendario di default

Per valutare la similarità delle interfacce generate è stata sviluppata una scala Likert [25]. La scala Likert è uno strumento di misurazione utilizzato per raccogliere dati su opinioni, percezioni, atteggiamenti o preferenze degli individui. In particolare sono stati definiti i componenti e le caratteristiche di questi oggetto di valutazione, ad ognuno di essi viene assegnato un punteggio di somiglianza che va da 0 (diverso) a 5 (identico). Questo processo permette di produrre punteggi di similarità per le immagini che consentono di classificare le prestazione del modello e degli utenti. Ogni elemento ha 4 caratteristiche che concorrono al suo punteggio per un totale di 20 punti ciascuno, essendo la prima interfaccia composta da 4 componenti principali si ha un massimo di 80, mentre la seconda avendo 5 componenti si raggiunge un massimo di 100.

Poichè la scala Likert non prevede l'utilizzo di metriche quantitative, è stato necessario esprimere per ogni elemento osservato le caratterische che influenzano il punteggio permettendo di rimanere consistente nelle valutazioni. Di seguito troviamo l'elenco dei componenti:

Data

- Testo: La valutazione si basa sulle informazioni contenute nel testo, mese, giorno, anno e successivamente sulla formattazione. Più condizione sono rispechhiate più è alto il punteggio.
- Posizione: Il punteggio è in base alla posizione rispetto al target da raggiungere, maggiore è la distanza minore è il punteggio.
- Colore: Per ottenere il punteggio si valuta la gradazione di colore rispetto al target, più è vicina al target maggiore è il punteggio.

- Dimensione: Viene valutato il font e altri modificatori testuali come grassetto e corsivo, meno diversità sono riscontrate maggiore è il punteggio.

Bottoni

- Testo/Frecce: La valutazione avviene osservando il contenuto dei bottoni, se sono presenti frecce o altri elementi si valuta la loro similarità con il target, tuttavia se il target preenta triangoli ripeni e la generazine ne produce di vuoti il punteggio non sarà massimo.
- Posizione: Il punteggio è in base alla posizione rispetto al target da raggiungere, maggiore è la distanza minore è il punteggio
- Colore: Per ottenere il punteggio si valuta la gradazione di colore rispetto al target, più è vicina al target maggiore è il punteggio.
- Forma: Si valuta la forma del bottone, più caratterische ha in comune con il target maggiore è il punteggio, ad esempio se il target è circolare e la generazione produce bottoni quadrati con bordi curvi la valutazione sarà maggiore rispetto a bottoni con angoli a spigolo.

Vista

- Vista: Il punteggio è data da quanto la tipologia di visualizzazione del calendario è simile al target.
- Posizione: Il punteggio è in base alla posizione rispetto al target da raggiungere, maggiore è la distanza minore è il punteggio.
- Colore: Per ottenere il punteggio si valuta la gradazione di colore rispetto al target, più è vicina al target maggiore è il punteggio.
- Dimensione: Si valuta le dimensioni della vista rispetto all'orginale, più dimensioni combaciano più il punteggio è alto.

Eventi

- Visualizzazione: Il punteggio è dato dalla similarità tra la posizione degli eventi nel target e quella generata.
- Testo: Il punteggio è dato dalle informazioni riportate nel testo degli eventi e successivamente dalla disposizione del testo di queste.
- Colore: Per ottenere il punteggio si valuta la gradazione di colore rispetto al target, più è vicina al target maggiore è il punteggio.
- Dimensione: Si valuta le dimensioni dell'evento rispetto all'orginale, più dimensioni combaciano più il punteggio è alto.

Componente aggiuntivo - se presente

- Funzionalità: Si valutano quante funzionalità del componente target sono presenti nel componente generato.
- Forma: Si valuta la forma del bottone, più caratterische ha in comune con il target maggiore è il punteggio.
- Colore: Per ottenere il punteggio si valuta la gradazione di colore rispetto al target, più è vicina al target maggiore è il punteggio.
- Posizione: Il punteggio è in base alla posizione rispetto al target da raggiungere, maggiore è la distanza minore è il punteggio.

SUS (System Usability Scale)

Al termine dello svolgimento dei due compiti assegnati segue un questionario SUS (System Usability Scale) [26]. Il questionario SUS è utilizzato per misurare come gli utenti percepiscono l'usabilità del sistema, è composto da 10 domande ed ognuna a 5 risposte che vanno da *Strongly Disagree* a *Strongly Agree*. Il risultato del questionario è un punteggio da 0 a 100, calcolato seguendo le seguenti regole:

1. Ogni risposta va da 1 a 5.
2. Ogni domanda numerata dispari sottrarre 1 dal valore della risposta.
3. Ogni domanda numerata pari sottrare a 5 il valore della risposta.
4. Sommare i valori così ottenuti.
5. Moltiplicare per 2.5.

Un sistema viene definito "Usabile" con un punteggio di 68 che è sopra la media. Il SUS è molto versatile e può essere utilizzato in molti contesti, inoltre e molto rapido ed è ampiamente impiegato tuttora e per questo è stato scelto come questionario per lo studio utente di questo progetto. Tuttavia bisogna tenere in considerazione che il risultato è puramente soggettivo, poiche misura la percezione dell'usabilità di un sistema, di conseguenza non risulta possibile comparare due diversi sistemi utilizzando questo strumento. Inoltre non è in grado di fornire informazioni su cosa migliorare all'interno del sistema e quindi non può essere usato come strumento di diagnostica.

5.2 Esecuzione

Lo svolgimento dei test è avvenuto in presenza con l'utilizzo di un computer sul quale è stato eseguita la versione dell'architettura oggetto di test. Il candidato quindi, seguendo l'ordine dello script, risponde ai quesiti posti nella fase inizale.

Successivamente gli vengono fornite le credenziali di accesso per svolgere le attività e viene indirizzato nel browser dove potrà effetuare il login.

Poichè tutti gli esperimenti sono stati svolti da una singola persona, il ruolo di osservatore e facilitatore è stato unificato in un unico individuo. Il facilitatore si occupa di intervenire nel caso in cui il partecipante riscontri delle criticità o difficoltà prestando attenzione a non compromettere la validità dell'esperimento.

Durante lo svolgimento delle attività le principali azioni dell'intervistato sull'interfaccia sono registrate attraverso un *log* automatico che identifica l'orario ed il tipo di azione effettuata.

Al termine delle attività e della compilazione del SUS viene richiesto al cadidato di esprimere delle considerazioni aggiuntive riguardo la sua esperienza di utilizzo ed eventuali problematiche riscontrate durante le interazioni con il sistema.

5.3 Analisi

In questo paragrafo verrano analizzati i dati raccolti durante lo svolgimento dell'esperimento. Le informazioni includono aspetti demografici, conoscenza di LLM, soprattutto le metriche relative allo svolgimento delle attività proposte ed infine i risultati del SUS. Lo scopo è quello di verificare se il sistema proposto è in grado di essere utilizzato dagli utenti e che grado di accurettezza raggiunge.

5.3.1 Analisi Demografica

In questa sezione vengono analizzati i dati raccolti dai partecipanti all'esperimento. In particolare le informazioni visualizzate includono dati demografici e professionali, come l'età, la professione e il livello di familiarità con gli LLM. Questi dati consentono di comprendere meglio il profilo degli utenti che hanno partecipato ai test, valutando se caratteristiche come esperienza e background professionale possano influire sull'interazione con l'interfaccia e l'usabilità del sistema. La Figura 5.4 mostra la distribuzione dei partecipanti per fasce di età. Si può notare che i due terzi dei partecipanti all'esperimento rientrano nel gruppo di età compreso tra i 20 ed i 30 anni, mentre la restante parte è composta in maggioranza dal gruppo tra i 30 ed i 60 anni ed un solo individuo al di sotto dei venti. Queste informazioni verrano utilizzate in seguito per analizzare come diverse fasce di età interagiscono con il sistema.

La figura 5.5, invece, mostra la distribuzione delle occupazioni dei partecipanti all'esperimento. Si può notare che la metà delle persone coinvolte lavora o studia nel settore della tecnologia, che potrebbe indicare una predisposizione all'utilizzo di queste tecnologie. Le restanti persone sono distribuite in vari altri campi, questa diversità di background è necessaria per valutare il sistema anche per individui non esperti.

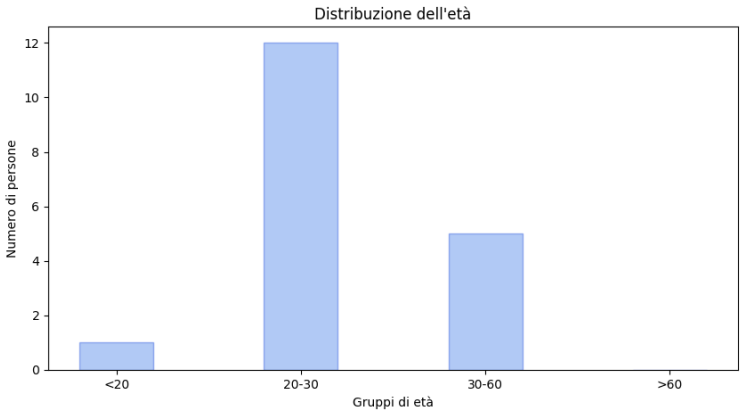

Figura 5.4: Distribuzione età partecipanti

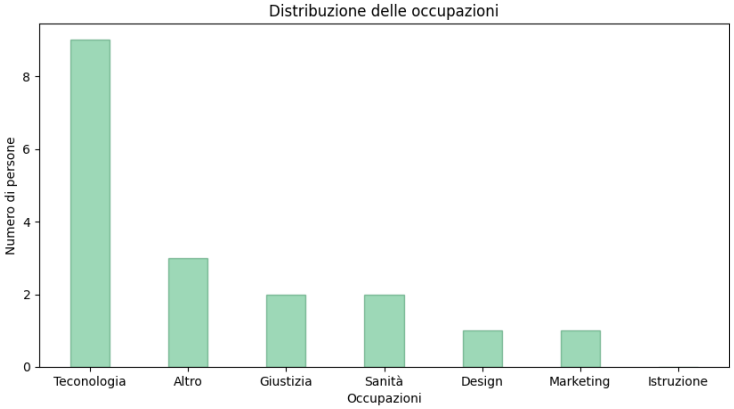

Figura 5.5: Distribuzione occupazioni partecipanti

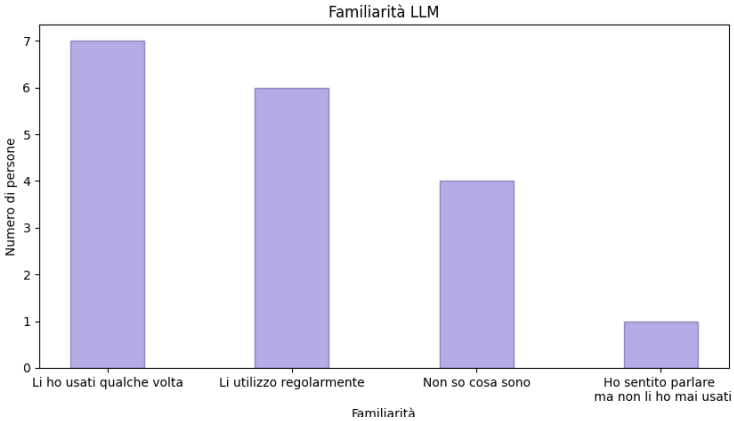

Figura 5.6: Distribuzione familiarità partecipanti con LLM

Nella figura 5.6, viene mostrata la familiarità che le persone hanno con l'utilizzo dei LLM. Questa metrica risulta particolarmente rilevante poiché permette di individuare non solo se chi conosce ed utilizza già questi modelli ha un vantaggio nell'eseguire le attività, ma anche se chi non ne ha mai fatto uso è in grado di adattarsi e utilizzare efficacemente il sistema. Osservando la figura si può notare come la maggior parte degli intervistati conosca e abbia usato gli LLM, tuttavia rimane un terzo degli utenti che non hanno mai avuto modo di usare questi modelli.

5.3.2 Analisi Attività Semplice

In questa sezione verranno presentati i dati e le metriche raccolte durante lo svolgimento della prima attività. Come si può vedere rapresentato nella figura 5.8 in media gli utenti hanno impiegato tra i 15 e di 20 minuti, compreso il tempo di generazione, per completare l'attività. Un elemento che condiziona la durata è da ricondurre al tempo che l'utente impiega per comprendere l'utilizzo corretto dell'interfaccia di generazione e la scrittura di una descrizione efficace. Successivamente i grafici mostreranno una riduzione nel tempo, dovuta all'esperienza acquisita.

I punteggi relativi alle interfacce generate sono riportati in figura 5.9. I valori sono stati suddivisi per gruppi di età, mostrando come i punteggi siano molto simili tra i vari gruppi. In particolare, si osserva che, in media, è stato raggiunto un punteggio di similarità del 77%, ciò sta a significare che gli utenti, indipendentemente dall'età, sono riusciti a riprodurre l'interfaccia proposta con un discreto grado di similarità, toccando in alcuni casi picchi del 93%. È interessante notare come non siano emerse differenze significative tra le fasce d'età, suggerendo che il modello è stato in grado di comprendere agevolmente le descrizioni del design degli utenti con esperienze e competenze potenzialmente diverse. A supporto di questa analisi, la figura 5.7 mostra un esempio concreto di un'interfaccia generata da un utente durante l'esperimento.

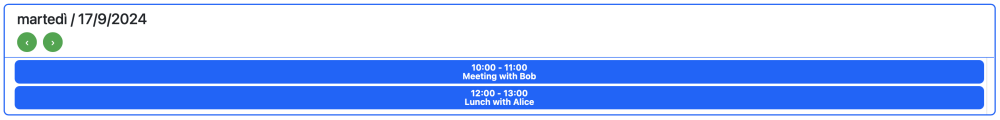

Figura 5.7: Esempio generazione utente

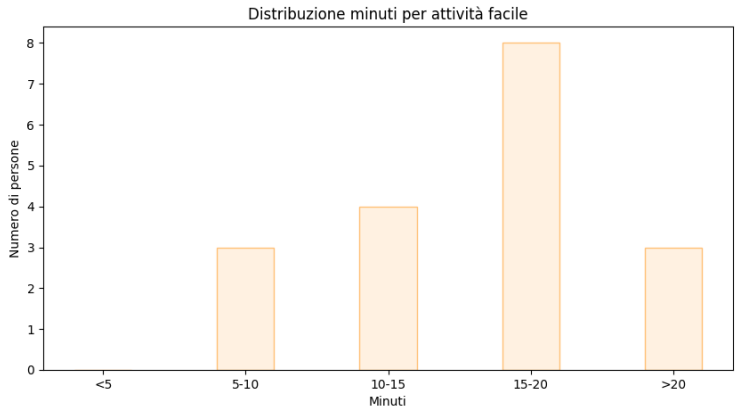

Figura 5.8: Distribuzione tempo di completamento attività

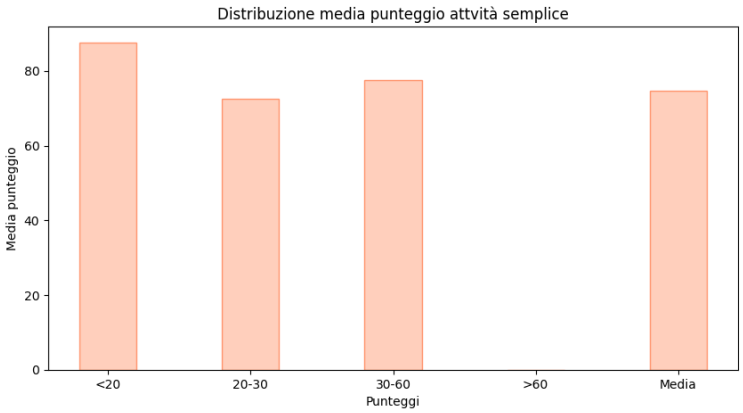

Figura 5.9: Distribuzione punteggi per età

Nella figura 5.10 viene mostrata la media delle operazioni svolte dall'utente per completare l'attività. Si può notare che il raggiungimento del risultato ottimale ha richiesto mediamente poco più della metà delle generazioni previste evidenziando un'elevata efficienza del modello, in quanto gli utenti raramente hanno scelto di tornare all'interfaccia precedente o di ricominciare da capo. Tale comportamento suggerisce che il modello è in grado di generare interfacce coerenti ed in linea con le richieste, dimostrando una notevole capacità di interpretare le preferenze e le scelte espresse. Solo in rari casi il sistema ha prodotto interfacce non utili o non coerenti con le descrizioni degli utenti, evidenziando margini di miglioramento nella gestione di richieste particolarmente complesse o ambigue. Tuttavia, la frequenza ridotta di questi eventi sottolinea l'affidabilità complessiva del sistema.

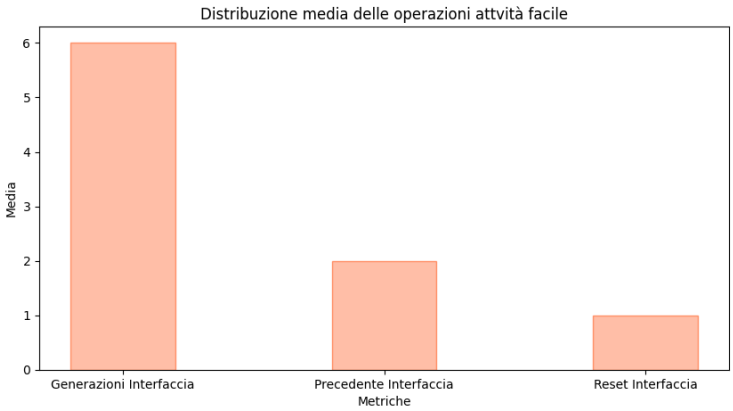

Figura 5.10: Distribuzione media delle operazioni

Un'analisi interessante riguarda la relazione tra la lunghezza della descrizione fornita dall'utente e il punteggio ottenuto, come mostrato in figura 5.11. Si osserva che la similarità e l'accuratezza dell'interfaccia generata aumentano proporzionalmente all'incremento del numero di parole fino a una lunghezza compresa tra 15 e 20 parole. Tuttavia, superata questa soglia, la qualità delle interfacce prodotte tende, verosimilmente, a diminuire.

Questo comportamento suggerisce come il modello riesca a gestire con maggiore efficacia descrizioni concise e focalizzate, mentre incontra difficoltà crescenti nel comprendere richieste particolarmente lunghe. Tali difficoltà possono essere causate da una "saturazione dell'attenzione" ovvero la capacità limitata del modello di distribuire la propria attenzione su descrizioni estese, portando a una perdita di coerenza o precisione nell'output generato. Un altro aspetto da considerare è che descrizioni più lunghe aumentano la probabilità di introdurre involontariamente incongruenze o ambiguità. Questi errori possono confondere il modello, spingendolo a generare interfacce meno coerenti rispetto alle aspettative dell'utente.

In conclusione si può affermare che i partecipanti sono stati in grado di completare la prima attività con successo, mostrando che per questa tipologia di compito il sistema risponde correttamente agli input degli utenti indipendentemente dalle loro abilità.

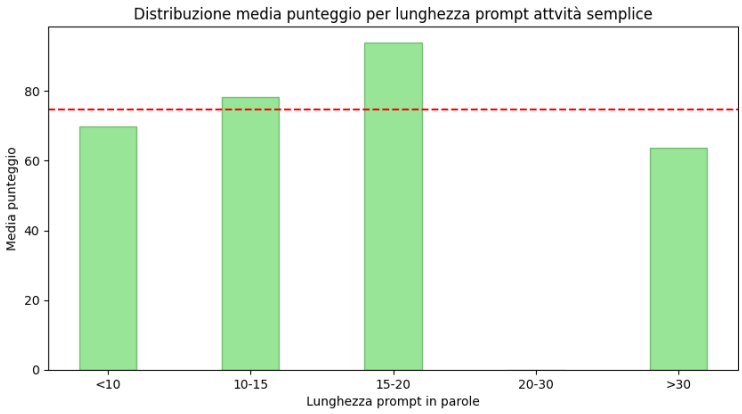

Figura 5.11: Distribuzione punteggi per lunghezza descrizione

5.3.3 Analisi Attività Difficile

La seconda attività prevede la generazione di un'interfaccia più complessa, tuttavia come emerge in figura 5.13 il tempo medio per lo svolgimento si è ridotto nel range tra i 10 ed i 15 minuti. La riduzione del tempo richiesto per completare il processo è attribuibile, come emerge anche dai commenti degli utenti nei paragrafi seguenti, all'aumento della familiarità con il sistema e con l'interfaccia. Questa maggiore confidenza ha consentito di accelerare le operazioni di generazione. Si può inoltre osservare una diminuzione dei punteggi ottenuti per questa specifica attività. Nella figura 5.14 vengono rappresentate le prestazioni degli intervistati nella seconda attività, evidenziando un calo rispetto ai risultati ottenuti rispetto alla prima (5.9). Sebbene gli utenti abbiano dimostrato una maggiore familiarità con il sistema e con la scrittura delle preferenze, la generazione di un'interfaccia più complessa ha rappresentato una sfida più significativa. I dati rivelano che la media dei punteggi di similarità delle interfacce generate è stata del 70%, con picchi che hanno raggiunto l'87%. Tuttavia, è emerso un divario tra gli utenti più esperti, capaci di sfruttare appieno le potenzialità del sistema, e quelli meno esperti con punteggi intorno al 55%. Questo suggerisce la necessità di migliorare ulteriormente il design dell' inserimento delle istruzioni o di potenziare il sistema con meccanismi di guida e feedback, tutto ciò per supportare l'utilizzatore nella costruzione di interfacce complesse in modo più efficace. Di seguito nella Figura 5.12 un esempio di generazione per questa attività svolto da un utente durante l'esperimento.

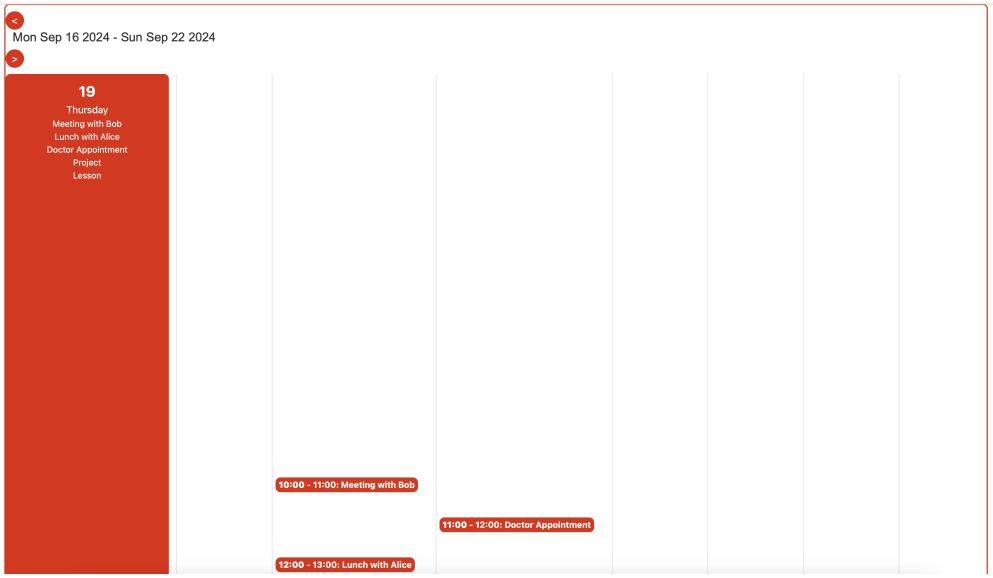

Figura 5.12: Esempio generazione utente

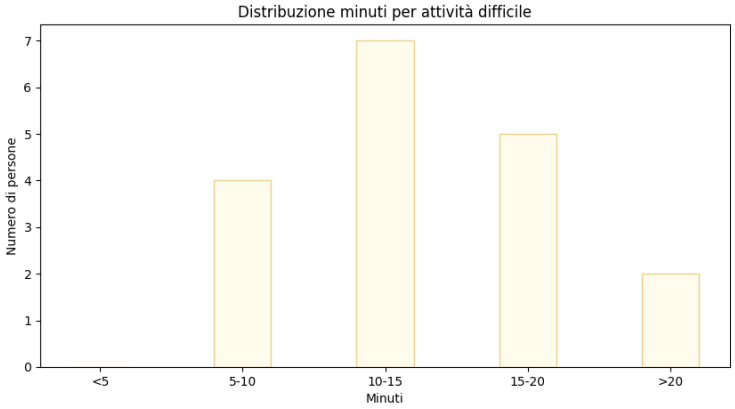

Figura 5.13: Distribuzione tempo di completamento attività

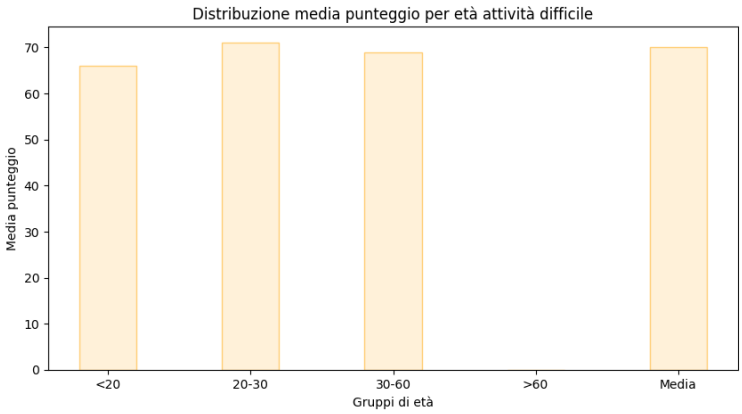

Figura 5.14: Distribuzione punteggi per età

Anche per l'attività più complessa è stato possibile mettere in relazione i punteggi ottenuti con la lunghezza media delle descrizioni, come mostrato in figura 5.15. Analizzando i dati, si osserva un andamento simile a quello rilevato nella prima attività: i punteggi crescono con l'aumento del numero di parole fino a una lunghezza ottimale di 15-20 parole per descrizione, oltre tale soglia, si verifica una graduale diminuzione delle prestazioni. Questa osservazione avvalora l'ipotesi avanzata in precedenza, secondo cui una descrizione concisa ma sufficientemente dettagliata rappresenta un elemento chiave per ottenere risultati di elevata qualità. Le descrizioni troppo brevi risultano insufficienti per generare un'interfaccia che rifletta accuratamente le preferenze dell'utente, mentre quelle troppo lunghe rischiano di introdurre ambiguità o complessità che compromettono l'efficacia del sistema.

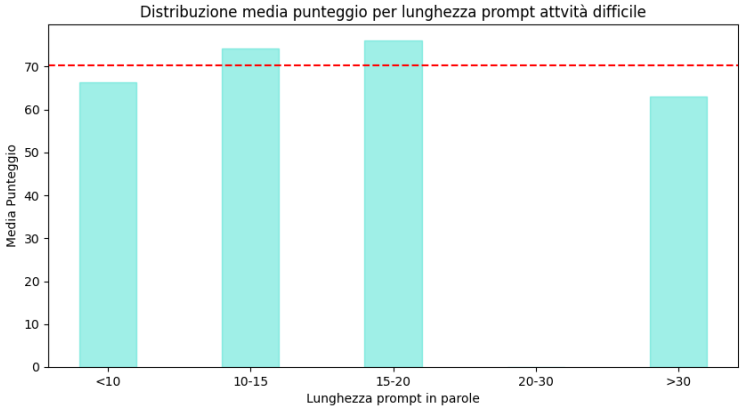

Figura 5.15: Distribuzione punteggi per età

5.3.4 Comparazione risultati attività

Dopo aver analizzato i risultati delle due attività singolarmente, si può procedere approfondendo lo studio mediante la comparazione degli esiti, in modo da ottenere un quadro più dettagliato ed evidenziare eventuali differenze e criticità. Questo confronto è utile per comprendere se determinate problematiche emergano solo in specifici scenari o se siano invece di carattere generale, influenzando evidentemente l'intera esperienza utente. Osservando la figura 5.16, si può notare che i risultati ottenuti dai partecipanti nelle due attività sono coerenti, sebbene appaia una differenza nei punteggi medi. L'attività più complessa registra valutazioni leggermente inferiori ma il divario non risulta significativo.

L'analisi mostra come i partecipanti al test sono agevolmente riusciti a interagire con il sistema grazie all'esperienza acquisita nella prima fase operativa, consentendogli di procedere nelle fasi successive con maggiore sicurezza. Queste circostanze confermano, ancora una volta, la bontà dell'interfaccia di generazione. Tuttavia, non può sottacersi che la complessità aggiuntiva potrebbe necessitare di un supporto per generazioni complesse, ad esempio integrando il processo generativo.

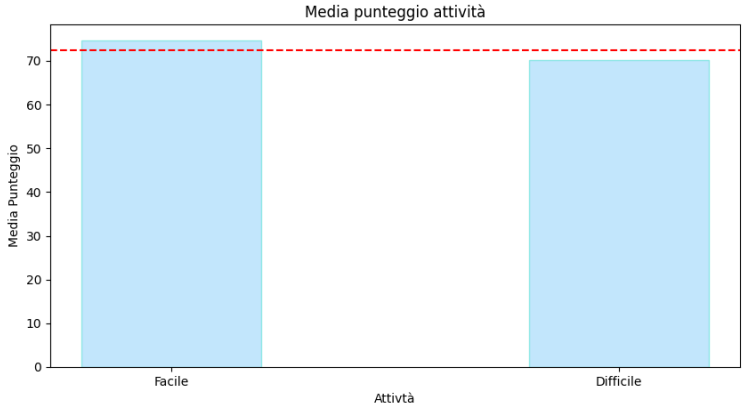

Figura 5.16: Punteggi per attività

Un'altra metrica rilevante emersa dall'analisi è la riduzione del tempo necessario per completare le attività proposte che viene mostrata nella figura 5.17. Questo dato è particolarmente significativo, in quanto esprime un miglioramento nella familiarità degli utenti con l'interfaccia e una maggiore efficienza nelle loro interazioni. Nonostante la seconda attività fosse più complessa rispetto alla prima, il tempo impiegato per portarla a termine è diminuito del 18%, dimostrando che gli utenti sono stati in grado di adattarsi dopo il primo utilizzo. Questo risultato può essere attribuito alla capacità degli utenti di comprendere il funzionamento dell'interfaccia durante l'esperienza precedente, tuttavia come sottolineato nei commenti la curva

di apprendimento non è così lineare, probabilmente con qualche variazione sarà possibile ridurre ancora i tempi e migliorare la performance.

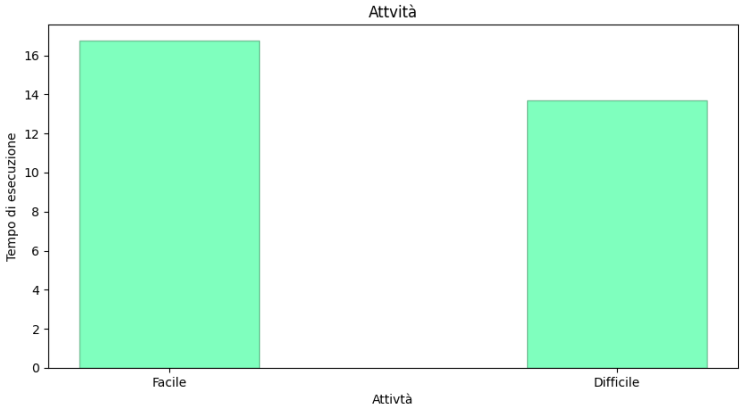

Figura 5.17: Tempo per attività

Infine, la figura 5.18 mostra il confronto della lunghezza media delle descrizioni generate per le due attività. Dal confronto emerge un aumento della lunghezza media delle descrizioni nella seconda attività. Come già detto in precedenza nella sezione 5.3.3, questo incremento del numero di parole potrebbe essere una delle cause delle prestazioni leggermente inferiori osservate nella seconda attività.

L'aumento della lunghezza delle descrizioni potrebbe indicare che gli utenti hanno cercato di fornire istruzioni più dettagliate o complesse per affrontare la maggiore difficoltà della seconda attività. Tuttavia, questa complessità aggiuntiva potrebbe aver introdotto ambiguità, riducendo l'efficacia complessiva.

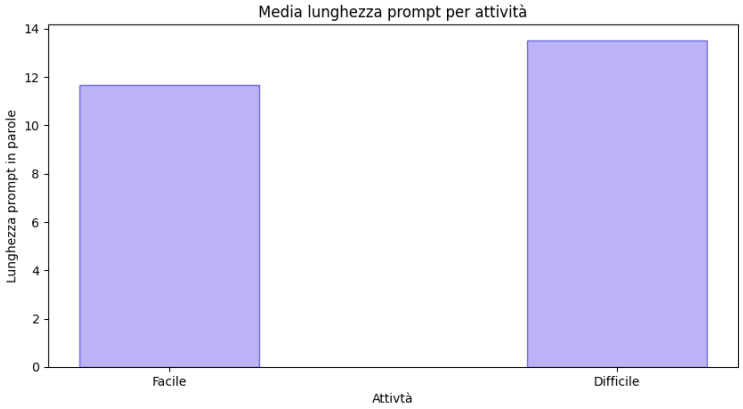

Figura 5.18: Lunghezza prompt per attività

5.4 Risultati questionario SUS

I risultati dei questionari SUS raccolti sono stati organizzati e analizzati per valutare l'usabilità complessiva del sistema. Per approfondire l'analisi e comprendere se sussistono differenze significative basate sull'età, i partecipanti sono stati suddivisi in gruppi (es. <20 anni, 20-30 anni, 30-60 anni, >60 anni).

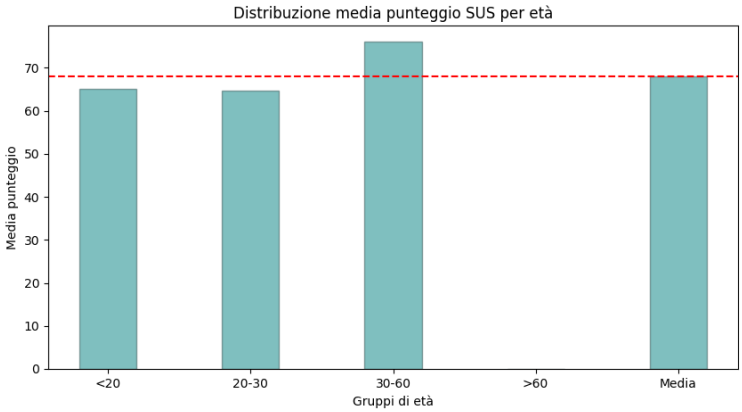

Figura 5.19: Distribuzione punteggi SUS per età

Come illustrato nella Figura 5.19, il sistema ha ottenuto un punteggio medio di 68/100, rappresentato dalla linea tratteggiata rossa. Questo risultato indica che l'interfaccia ha raggiunto la soglia minima necessaria per essere considerata usabile secondo lo standard del SUS. Successivamente, è stata calcolata la media aritmetica per ciascuna domanda del questionario, come mostrato in Figura 5.20. Dall'analisi si può notare che tutte le domande dispari hanno ottenuto un punteggio medio superiore a 3, mentre le domande pari si attestano su un punteggio medio di 2. Questo andamento riflette una tendenza comune nei questionari SUS promettenti, dove le domande dispari valutano aspetti positivi del sistema (ad esempio, facilità d'uso e fiducia nell'interfaccia), mentre le domande pari si concentrano su aspetti negativi (come la complessità e le difficoltà percepite). Un punteggio medio più alto per le domande dispari suggerisce che gli utenti hanno percepito in modo positivo diversi elementi del sistema. Tuttavia, i punteggi mostrano aree di miglioramento, come ad esempio la riduzione della complessità percepita o un affinamento delle funzionalità meno intuitive.

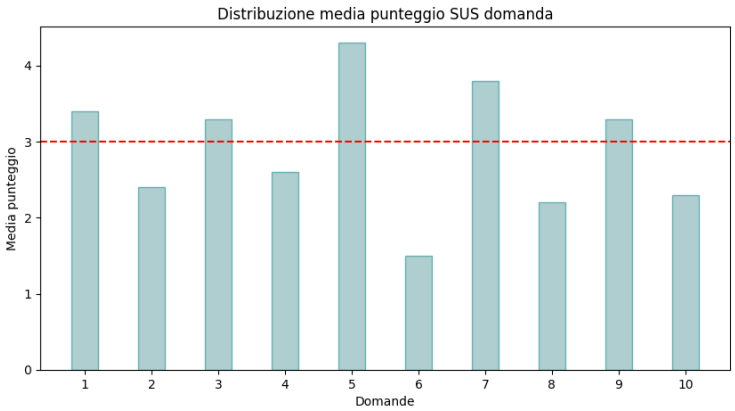

Figura 5.20: Media aritmetica domande SUS

5.4.1 Analisi commenti e possibili modifiche

Dall'analisi dei feedback espressi dagli utenti a fine esperimento è emerso che la maggior parte di essi ha percepito il sistema e la sua interfaccia come intuitiva e chiara da utilizzare, ma con alcune lievi criticità che potrebbero ostacolare l'esperienza.

> "*Ho trovato facile muovermi all'interno dell'interfaccia utente, e capire come selezionare e implementare i singoli comandi, è molto intuitivo e alla portata di chiunque. Al contrario ho trovato complesso riuscire ad esprimere a parole i comandi da eseguire...*"

Di seguito vengono mostrate le principali criticità segnalate e le possibili soluzioni per migliorare l'esperienza.

Difficoltà nella creazione dei prompt

Uno degli aspetti più ricorrenti nei commenti riguarda la complessità nel definire correttamente i comandi da eseguire. Come indicato dal seguente commento:

> "*La parte più complicata è comprendere come scrivere correttamente le istruzioni per la generazione delle varie parti, al contrario capire come utilizzare il sistema di inserimento è immediato.*"

Per migliorare questo aspetto, si potrebbero inserire degli esempi di prompt pre-impostati come suggerito anche in altri commenti, oppure fornire linee guida chiare e visibili su come formulare correttamente le richieste.

Feedback visivo ed evidenziazione degli elementi

Un altro elemento presente nei commenti è stata la necessità di migliorare il feedback visivo durante l'interazione.

> "*Evidenziare gli elementi che si stanno modificando per capire meglio come procedere. Integrare un sistema di evidenziazione degli elementi selezionati o modificati potrebbe aiutare gli utenti a orientarsi meglio e ridurre gli errori.*"

Alcuni utenti non riescono ad individuare il componente o l'insieme dei componenti sui quale stanno agendo, pertanto si potrebbe integrare un sistema per porre in risalto l'elemento selezionato permettendo una migliore comprensione delle proprie azioni sull'interfaccia.

Semplificazione dei nomi degli elementi e categorizzazione

Legato al punto precedente oltre all'evidenziare l'elemento sul quale si sta agendo, gli utenti hanno suggerito di semplificare i titoli permettendo di agire facilmente nell'interazione con il sistema.

> "*I nomi degli elementi andrebbero semplificati per una migliore identificazione.*"

Problemi di visualizzazione durante l'inserimento dei comandi

Un'altra segnalazione ricorrente è stata la difficoltà nel visualizzare l'intero testo dei prompt durante l'inserimento come riportato di seguito.

> "*L'unico difetto che ho riscontrato è stato che mentre si scrive il prompt si può vedere solo parte della frase scritta, rendendo difficile modificarla.*"

La soluzione consiste nel rendere la casella di input della descrizione disponibile su più righe ed implementare una funzione di scroll del testo.

Supporto iniziale per i nuovi utenti

Infine, nonostante il sistema risulti intuitivo dopo le prime attività, alcuni utenti hanno riportato difficoltà iniziali, tra queste si segnala la seguente:

> "*All'inizio mi ci è voluto un attimo e una spiegazione dettagliata del programma ma una volta capito mi sono trovata molto bene.*"

Tra i suggerimenti troviamo la possibilità di inserire un tutorial o guida iniziale come avviene nelle moderne applicazioni, che mostri le funzionalità e riduca la curva di apprendimento.

Capitolo 6

Conclusioni

L'obiettivo della tesi era quello di sviluppare un'architettura in grado di permettere la generazione di interfacce su misura in tempo reale che riescano a soddisfare le esigenze degli utenti in modo più soddisfacente dei sistemi tradizionali.

Lo sviluppo dell'architettura proposta è stato preceduto da uno studio dei principi delle discipline HCI (Human-Computer Interaction) e HMAI (Human-Machine AI Interaction), per comprendere come gli utenti interagiscono con le tecnologie e come i sistemi intelligenti possano migliorare tali interazioni, offrendo esperienze più intuitive e personalizzate.

Successivamente, è stata condotta un'analisi delle architetture esistenti per la personalizzazione delle interfacce, evidenziandone i limiti principali: personalizzazione vincolata a opzioni predefinite, complessità tecnica elevata e mancanza di adattamento dinamico alle esigenze dell'utente. Infine, è stato introdotto lo studio dei LLMs (Large Language Models), modelli generativi che rappresentano una soluzione innovativa per superare tali barriere, consentendo di progettare un sistema adattivo.

Dopo lo studio preliminare, è iniziata la fase di progettazione, durante la quale sono stati definiti i moduli principali dell'architettura e le loro interazioni per garantire un flusso coerente e flessibile. Il sistema è stato progettato in moduli chiave: un Modulo di Input Utente, per raccogliere preferenze in modo intuitivo; un Modulo di Generazione, basato su LLM, per interpretare gli input e creare soluzioni personalizzate; un Modulo di Rendering, per trasformare le soluzioni in interfacce interattive.

Successivamente, è stata avviata la fase di realizzazione attraverso un processo iterativo, che ha portato alla creazione della interfaccia di input, permettendo di testare e perfezionare i moduli definiti durante la sviluppo.

Infine per verificare l'usabilità del sistema, attestare le capacità di soddisfare i bisogni degli utenti e produrre interfacce valide è stato condotto uno studio utente. Sono stati scelti 18 partecipanti di diversi gruppi di età e background ai quali è

stato chiesto di eseguire due attività e successivamente di valutare l'architettura ed il suo funzionamento. L'analisi dello studio utente ha prodotto risultati positivi, con la maggior parte degli utenti che ha ampiamente completato le attività con successo, suggerendo tuttavia delle possibili modifiche. Infine ad ognuno di essi è stato somministrato un questionario SUS per valutare quantitivamente l'usabilità del sistema ottenendo un punteggio sufficente, sussistendo margini per migliorare l'esperienza di utilizzo.

Complessivamente l'architettura realizzata soddisfa i requisiti prefissati all'inzio della tesi, dimostrando di essere in grado di generare interfacce su misura basate sulle preferenze e descrizioni degli utenti. Questo risultato consente di introdurre un nuovo approccio alla personalizzazione e conferma come l'applicazione di nuove tecnologie possa trasformare l'interazione in un'esperienza su misura ponendo al centro l'utente e le sue necessità.

6.1 Sviluppi Futuri

Dall'esito dei test è emerso che l'architettura funziona correttamente pur essendoci possibili miglioramenti che renderebbero l'esperienza utente ancora più semplice. Come riportato nei commenti la criticità riscontrata è correlata all'apprendimento delle modalità di descrizione le proprie preferenze. Pertanto si annota la possibilità di interventi sull'interfaccia e nei metodi di input, inserendo anche la possibilità di migliorare automaticamente le descrizioni in uno step che precede la generazione, permettendo di rendere più determinanti le preferenze degli espresse dagli utenti.

Questa architettura apre la strada a nuovi metodi di configurazione e personalizzazione che sfruttando i modelli generativi estendono le attuali possibilità.

Appendice A

Questionario

Q. 1 Quanti anni hai?

- ○ <20
- ○ 20-30
- ○ 30-60
- ○ >60

Q. 2 Quale è il tuo genere?

- ○ Maschio
- ○ Femmina
- ○ Altro o preferisco non specificare

Q. 3 Quale è il più alto livello di istruzione completato?

- ○ Scuola superiore
- ○ Laurea Triennale
- ○ Laurea Magistrale
- ○ Dottorato
- ○ Altro

Q. 4 Quanto ti senti a tuo agio ad usare la tecnonlogia (ex. Computer, Smartphone etc..)

- ○ Per nulla a mio agio
- ○ Poco a mio agio
- ○ Neutro
- ○ Abbastanza a mio agio

- ○ Molto a mio agio

Q. 5 In quale settore lavori / studi?

- ○ Tecnologia
- ○ Design
- ○ Marketing
- ○ Istruzione
- ○ Sanità
- ○ Giustizia
- ○ Altro

Q. 6 Quanto spesso utilizzi applicazioni di calendario (es. Google Calendar, Apple Calendar)?

- ○ Mai
- ○ Raramente (una volta al mese)
- ○ Spesso (una volta a settimana)
- ○ Quotidianamente

Q. 7 Quale tipo di applicazione di calendario utilizzi di solito?

- ○ Google Calendar
- ○ Apple Calendar
- ○ Microsoft Outlook Calendar
- ○ Altro
- ○ Nessuna

Q. 8 Quanto è importante per te la possibilità di personalizzare l'aspetto del tuo calendario (es. colori, caratteri)?

- ○ Per nulla importante
- ○ Poco importante
- ○ Neutro
- ○ Abbastanza importante
- ○ Molto importante

Q. 9 Quanto sei familiare con i LLM (es. ChatGPT)?

- ○ Non so cosa sono
- ○ Ho sentito parlare, ma non li ho mai usati

- ○ Li ho usati qualche volta
- ○ Li utilizzo regolarmente

Q. 10 Hai mai scritto prompt per utilizzare un modello di intelligenza artificiale (es. ChatGPT, MidJourney)?

- ○ Sì, spesso
- ○ Sì, qualche volta
- ○ No, mai

Q. 11 Se hai scritto prompt in passato, per quale scopo li hai utilizzati?

- ○ Per generare testo
- ○ Per generare immagini
- ○ Per automatizzare attività
- ○ Altro
- ○ Non ho mai scritto prompt

Q. 12 Quanto ti senti a tuo agio nel creare prompt specifici per ottenere risultati desiderati da un'intelligenza artificiale?

- ○ Per nulla a mio agio
- ○ Poco a mio agio
- ○ Neutro
- ○ Abbastanza a mio agio
- ○ Molto a mio agio

Q. 13 Quanto ti fidi degli strumenti di intelligenza artificiale per personalizzare l'esperienza utente?

- ○ Per nulla
- ○ Poco
- ○ Neutro
- ○ Abbastanza
- ○ Molto

Q. 14 Hai esperienza nella progettazione di interfacce utente?

- ○ No, non ho esperienza
- ○ Sì, ho una conoscenza di base
- ○ Sì, ho esperienza intermedia

○ Sì, sono un esperto nella progettazione UI

Q. 15 Se hai avuto esperienze conl'uso di LLM, potresti descrivere brevemente come li hai utilizzati e in quali contesti applicativi?

Q. 16 Se potessi personalizzare l'interfaccia utente di un calendario, quali aspetti vorresti modificare? Perché queste modifiche sarebbero importanti?

Bibliografia

[1] Alan Dix, Janet E. Finlay, Gregory D. Abowd e Russell Beale. *Human-Computer Interaction*. USA: Prentice-Hall, Inc., 2003. ISBN: 0130461091 (cit. alle pp. 5, 6, 42).

[2] Qian Yang, Aaron Steinfeld, Carolyn Rosé e John Zimmerman. «Re-examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design». In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376301. URL: https://doi.org/10.1145/3313831.3376301 (cit. a p. 6).

[3] Saleema Amershi et al. «Guidelines for Human-AI Interaction». In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–13. ISBN: 9781450359702. DOI: 10.1145/3290605.3300233. URL: https://doi.org/10.1145/3290605.3300233 (cit. a p. 7).

[4] Niraj Ramesh Dayama, Kashyap Todi, Taru Saarelainen e Antti Oulasvirta. «GRIDS: Interactive Layout Design with Integer Programming». In: CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376553. URL: https://doi.org/10.1145/3313831.3376553 (cit. alle pp. 8, 9).

[5] Jianan Li, Jimei Yang, Jianming Zhang, Chang Liu, Christina Wang e Tingfa Xu. *Attribute-conditioned Layout GAN for Automatic Graphic Design*. 2020. arXiv: 2009.05284 [cs.CV]. URL: https://arxiv.org/abs/2009.05284 (cit. a p. 8).

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville e Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661 (cit. a p. 8).

[7] Diego Martin Arroyo, Janis Postels e Federico Tombari. *Variational Transformer Networks for Layout Generation*. 2021. arXiv: 2104.02416 [cs.CV]. URL: https://arxiv.org/abs/2104.02416 (cit. a p. 8).

[8] Qianzhi Jing, Tingting Zhou, Yixin Tsang, Liuqing Chen, Lingyun Sun, Yankun Zhen e Yichun Du. «Layout Generation for Various Scenarios in Mobile Shopping Applications». In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI '23. Hamburg, Germany: Association for Computing Machinery, 2023. ISBN: 9781450394215. DOI: 10.1145/3544548.3581446. URL: https://doi.org/10.1145/3544548.3581446 (cit. a p. 8).

[9] Kashyap Todi, Gilles Bailly, Luis Leiva e Antti Oulasvirta. «Adapting User Interfaces with Model-based Reinforcement Learning». In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. ACM, mag. 2021. DOI: 10.1145/3411764.3445497. URL: http://dx.doi.org/10.1145/3411764.3445497 (cit. a p. 9).

[10] Wenyuan Kong, Zhaoyun Jiang, Shizhao Sun, Zhuoning Guo, Weiwei Cui, Ting Liu, Jianguang Lou e Dongmei Zhang. «Aesthetics++: Refining Graphic Designs by Exploring Design Principles and Human Preference». In: *IEEE Transactions on Visualization and Computer Graphics* 29.6 (2023), pp. 3093–3104. DOI: 10.1109/TVCG.2022.3151617 (cit. a p. 9).

[11] Jason Wu, Kashyap Todi, Joannes Chan, Brad A Myers e Ben Lafreniere. «FrameKit: A Tool for Authoring Adaptive UIs Using Keyframes». In: *Proceedings of the 29th International Conference on Intelligent User Interfaces*. IUI '24. Greenville, SC, USA: Association for Computing Machinery, 2024, pp. 660–674. ISBN: 9798400705083. DOI: 10.1145/3640543.3645176. URL: https://doi.org/10.1145/3640543.3645176 (cit. a p. 10).

[12] Jackie (Junrui) Yang et al. «ReactGenie: A Development Framework for Complex Multimodal Interactions Using Large Language Models». In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Vol. 1. CHI '24. ACM, mag. 2024, pp. 1–23. DOI: 10.1145/3613904.3642517. URL: http://dx.doi.org/10.1145/3613904.3642517 (cit. a p. 10).

[13] Paul De Bra. «Challenges in User Modeling and Personalization». In: *IEEE Intelligent Systems* 32.5 (2017), pp. 76–80. DOI: 10.1109/MIS.2017.3711638 (cit. a p. 11).

[14] Krzysztof Gajos e Daniel S. Weld. «Preference elicitation for interface optimization». In: *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*. UIST '05. Seattle, WA, USA: Association for Computing Machinery, 2005, pp. 173–182. ISBN: 1595932712. DOI: 10.1145/1095034.1095063. URL: https://doi.org/10.1145/1095034.1095063 (cit. a p. 11).

[15] Wikypedia. *Intelligenza artificiale*. URL: https://it.wikipedia.org/wiki/Intelligenza_artificiale (cit. a p. 12).

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser e Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762 (cit. a p. 13).

[17] Alec Radford, Karthik Narasimhan, Tim Salimans e Ilya Sutskever. «Improving language understanding by generative pre-training». In: (2018) (cit. a p. 15).

[18] MDN. *Javascript*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (cit. a p. 23).

[19] Meta. *React*. URL: https://react.dev (cit. alle pp. 23, 26).

[20] *Vite*. URL: https://vite.dev (cit. a p. 23).

[21] *Node*. URL: https://nodejs.org/en (cit. a p. 24).

[22] *OpenAI*. URL: https://openai.com (cit. alle pp. 24, 38).

[23] Wikipedia. *SQL*. URL: https://it.wikipedia.org/wiki/Structured_Query_Language (cit. a p. 24).

[24] *GitLab*. URL: https://about.gitlab.com (cit. a p. 24).

[25] Rensis Likert. *A technique for the measurement of attitudes*. 1998 (cit. a p. 45).

[26] *SUS*. URL: https://hell.meiert.org/core/pdf/sus.pdf (cit. a p. 47).