Università degli Studi di Napoli "Parthenope"
Dipartimento di Scienze e Tecnologie
Corso di laurea in Informatica

Elaborato finale di Laurea

Sviluppo di un ambiente virtuale per ridurre il senso di solitudine nella popolazione anziana

Integrazione di agenti virtuali per il sostegno alla socialità

Relatore
Prof.ssa Paola Barra

Candidato
Luca Tartaglia
Matr. 0124002294

Anno Accademico 2024/2025

ABSTRACT

In questa tesi presento **SOULFRAME**, un sistema conversazionale con avatar 3D pensato per rendere l'interazione con l'AI più semplice, naturale e accessibile. L'obiettivo principale è costruire un'esperienza d'uso intuitiva: l'utente deve poter parlare con l'avatar senza complessità tecniche, usando un'interfaccia adattabile sia a desktop (controlli da tastiera) sia a mobile/touch (push-to-talk e navigazione semplificata).

Il progetto è stato sviluppato con un'architettura modulare client-server. Sul lato frontend, **Unity WebGL** gestisce flusso UI, avatar e acquisizione audio. Sul lato backend, microservizi **FastAPI** si occupano di trascrizione vocale con **Whisper (STT)**, risposta contestuale con **LLM + RAG** (Ollama e ChromaDB), sintesi vocale con **Coqui XTTS v2 (TTS)** e gestione/caching degli asset avatar. La memoria è persistente per singolo avatar e può essere arricchita con note, documenti PDF e immagini, anche tramite OCR.

Una parte importante del lavoro riguarda l'automazione operativa: sono stati introdotti script di setup e gestione servizi per installazione e deploy in modo rapido su Windows e Ubuntu, riducendo errori manuali e tempi di configurazione. Durante lo sviluppo sono state affrontate criticità di integrazione tra servizi, latenza end-to-end e compatibilità tra ambienti. I risultati ottenuti mostrano un sistema funzionante, estendibile e sufficientemente robusto per conversazioni vocali contestuali in scenari realistici.

INDICE

1. INTRODUZIONE

1.1 Motivazione e contesto applicativo

La realtà virtuale (VR) si distingue dagli altri media interattivi per la capacità di far sentire l'utente presente in un luogo diverso da quello fisico. Slater e Sanchez-Vives descrivono questa presenza come il risultato di due fattori principali: la *place illusion*, legata alla coerenza tra movimenti e scena virtuale, e la plausibilità degli eventi, cioè quanto l'ambiente reagisce in modo credibile alle azioni dell'utente [1]. Per questo, non basta una buona qualità visiva. Conta anche come il sistema risponde e quanto le interazioni risultano significative.

In questo quadro, la VR viene usata da tempo per training, formazione e simulazioni in ambito medico e professionale, soprattutto quando serve riprodurre situazioni costose, rischiose o difficili da standardizzare. Diversi studi segnalano anche l'interesse per contesti con forte componente sociale, dove realismo e naturalezza dell'interazione influenzano direttamente l'esperienza.

Il limite emerge quando l'ambiente resta statico o gli attori virtuali seguono script rigidi. In questi casi l'interazione si riduce a scelte predefinite e perde adattività. In uno studio sul training in VR, Kan, Rumpelnik e Kaufmann confrontano agenti conversazionali con agenti a audio preregistrato. I risultati mostrano che una pipeline vocale con riconoscimento del parlato e sintesi vocale aumenta in modo significativo la co-presenza percepita [2]. Questo indica che il dialogo non è un elemento accessorio, ma una parte centrale dell'efficacia del sistema.

Gli Embodied Conversational Agents (ECA), cioè agenti conversazionali con corpo virtuale e segnali multimodali, nascono proprio da questa esigenza. La revisione sistematica di Yang e coautori mostra che la maggior parte degli studi in XR si concentra sulla VR, spesso con HMD e con Unity come piattaforma principale [3]. La stessa revisione evidenzia che molte interazioni restano *task-oriented*, ma cresce l'interesse per scambi più dinamici e adattivi grazie a modelli neurali

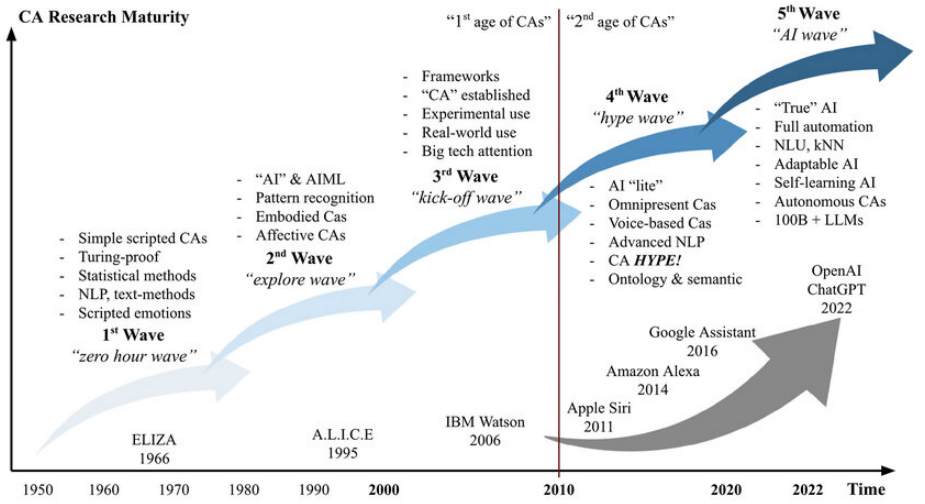

Fig. 1.1: Evoluzione della maturità della ricerca sui Conversational Agents (CA), dalla "zero hour wave" alla "AI wave", con riferimento ai principali passaggi tecnologici. [1]

e, più recentemente, ai *Large Language Models* (LLM). In SOULFRAME, questo filone viene adattato a una configurazione senza HMD, centrata sulla generazione e animazione di avatar a partire da segnali visivi e vocali dell'utente.

Dal punto di vista dei canali, la voce è ormai una scelta frequente sia in input sia in output. Resta però aperta la sfida di integrare dialogo libero e contestuale con un embodiment coerente.

Figura 1.1 sintetizza l'evoluzione dei Conversational Agents (CA) dalla fase iniziale di sistemi scriptati fino all'attuale ondata guidata da modelli linguistici avanzati. In questa traiettoria, l'integrazione di linguaggio naturale, adattività e autonomia rende più rilevante il tema dell'*embodiment* in ambienti immersivi. In questo quadro, SOULFRAME non nasce come semplice demo grafica: punta a collegare presenza sociale e interazione credibile in un ambiente 3D familiare, con la conversazione vocale come asse centrale.

SOULFRAME si inserisce in questo contesto come sistema immersivo con ECA vocale embodied. L'obiettivo è testare una pipeline completa che integra voce e un profilo avatar creato in fase iniziale tramite acquisizione visiva dell'utente, così da replicarne aspetto e stile comportamentale durante il dialogo. Un

[1] Fonte: adattamento da una figura pubblicata su ResearchGate, disponibile a questo link.

punto centrale è la memoria dell'agente: il sistema combina contesto conversazionale e recupero di conoscenza (*RAG*) per generare risposte coerenti con quanto detto e acquisito durante l'interazione. L'ipotesi di fondo è semplice: in ambienti immersivi e familiari, la qualità percepita dipende anche dalla capacità dell'agente di sostenere scambi plausibili, rapidi e contestualmente informati.

1.2 Perimetro e requisiti di progetto

SOULFRAME è un prototipo di interazione immersiva con un ECA vocale embodied in ambiente 3D. Lo scopo è valutare fattibilità tecnica e qualità percepita dell'interazione conversazionale in scenari con componente sociale. Il progetto non vuole essere una piattaforma generale per creare mondi virtuali e non è una soluzione clinica certificata. Questi obiettivi richiedono validazioni regolatorie e studi longitudinali fuori dal perimetro di una tesi triennale.

Il perimetro si concentra quindi sull'integrazione *end-to-end* della pipeline vocale, visiva e dialogica dentro una scena immersiva, con coerenza tra risposta verbale e comportamenti *embodied* dell'agente.

Sul piano tecnologico, il sistema integrato di SOULFRAME usa la fotocamera soprattutto nella fase di configurazione iniziale dell'utente, per creare e salvare il profilo avatar; durante l'interazione ordinaria il sistema si basa principalmente su microfono, memoria di contesto e stato conversazionale. La resa della scena avviene con moduli di generazione/animazione avatar e con un motore real-time 3D. Questa scelta favorisce replicabilità, facilita sostituzioni tra componenti e sostiene un'interazione più familiare, orientata alla replica visiva e comportamentale della persona.

I requisiti funzionali principali derivano dalla necessità di supportare un dialogo naturale in tempo quasi reale. L'input vocale è gestito con logica *push-to-talk*: l'utente tiene premuto un tasto per parlare e rilascia per chiudere il turno. A quel punto il sistema trascrive l'audio con Speech-to-Text (STT) e genera la risposta tramite LLM supportato dalla memoria RAG. La comprensione dell'input emerge dalla combinazione tra modello linguistico, contesto conversazionale e semplici regole applicative. Il testo prodotto viene poi convertito in audio tramite Text-to-Speech (TTS) e restituito attraverso la voce dell'agente. Nella fase di onboarding, il sistema usa la fotocamera per costruire il profilo visivo dell'avatar, che viene

poi riutilizzato durante la conversazione. A questa catena si aggiungono tre funzioni chiave: memoria contestuale con *Retrieval-Augmented Generation* (RAG), descrizione semantica delle immagini per estrarre informazioni dall'ambiente, e acquisizione testuale da documenti tramite OCR, così che l'avatar possa usare anche contenuti visivi e documentali nella conversazione.

I requisiti non funzionali riguardano soprattutto latenza, robustezza e modularità. Per mantenere plausibilità conversazionale, la catena STT–RAG/LLM–TTS deve restare reattiva, anche quando entrano in gioco recupero in memoria, analisi delle immagini e OCR. L'architettura deve anche tollerare guasti parziali senza interrompere l'esperienza immersiva. La modularità è perseguita soprattutto a livello di servizi e interfacce HTTP: i componenti backend possono essere sostituiti in modo relativamente rapido, purché restino compatibili con gli endpoint e i formati attesi dal client.

Per aspetti come la latenza percepita e i criteri di accettabilità, non c'è una soglia unica valida per tutti i contesti applicativi. In questo lavoro i vincoli sono quindi definiti in modo operativo sul prototipo e verificati nei capitoli successivi.

1.3 Obiettivi e contributi di SOULFRAME

Gli obiettivi di SOULFRAME seguono il filone degli studi sugli ECA in XR. Gli studi mostrano una forte presenza di implementazioni VR in Unity, interazioni spesso orientate al compito e una notevole eterogeneità in input, output e metodi di valutazione. Nello stesso tempo, i canali vocali sono molto usati, soprattutto con TTS in uscita [3]. In questo scenario, il progetto punta a costruire un prototipo che renda verificabile l'integrazione *end-to-end* della pipeline vocale con un agente *embodied*, documentando in modo chiaro le scelte tecniche e sperimentali.

Il primo obiettivo (RQ1) riguarda la fattibilità tecnica di un'interazione vocale in tempo reale con un agente embodied in ambiente immersivo 3D. In termini operativi, bisogna dimostrare che la catena STT $\rightarrow$ RAG/LLM $\rightarrow$ TTS $\rightarrow$ output audiovisivo dell'avatar funzioni in modo continuo e stabile, includendo anche descrizione immagini e OCR documentale. La valutazione considera tempi di elaborazione per blocco, completamento dei turni senza errori bloccanti e gestione corretta delle transizioni tra ascolto, elaborazione e risposta. La fattibilità inclu-

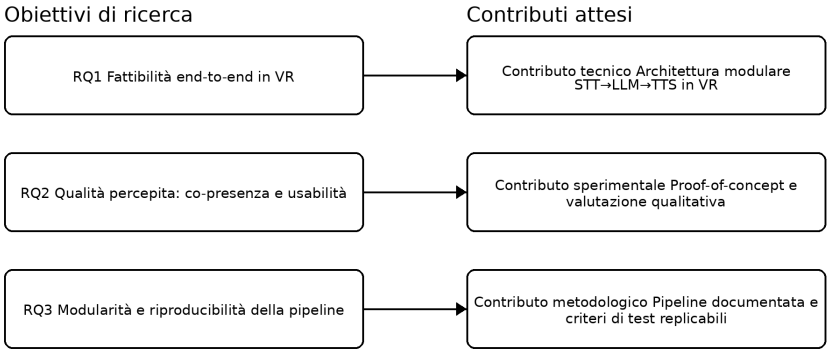

Fig. 1.2: Corrispondenza tra obiettivi di ricerca e contributi del progetto SOULFRAME.

de anche la coerenza dell'embodiment, cioè un avatar che allinei voce, aspetto visivo e segnali comportamentali.

Il secondo obiettivo (RQ2) riguarda la qualità percepita dell'interazione, in particolare co-presenza e naturalezza dialogica. SOULFRAME prevede almeno una valutazione qualitativa guidata, affiancata quando possibile da metriche soggettive usate negli studi VR con ECA: presenza, co-presenza, qualità dell'interazione e qualità della presentazione delle informazioni, oltre a misure di processo come durata dei compiti e performance percepita. Studi comparativi tra agenti conversazionali e audio pre-scriptato mostrano differenze osservabili, soprattutto sulla co-presenza [2].

Il terzo obiettivo (RQ3) riguarda modularità e riproducibilità della pipeline. In pratica, il sistema deve permettere la sostituzione dei componenti principali senza riscrivere l'intera architettura e deve rendere tracciabili configurazioni, tempi e risultati delle prove. Questa sostituzione avviene mantenendo stabili i contratti API tra client e servizi. Questo obiettivo completa i primi due, perché rende il prototipo confrontabile nel tempo e riutilizzabile in test successivi. Figura 1.2 sintetizza la corrispondenza tra i tre obiettivi e i contributi attesi del progetto.

I contributi si distribuiscono su tre livelli. Il contributo tecnico è un'architettura modulare *end-to-end* basata su microservizi (STT, RAG/LLM, TTS e servizi di memoria) con interfacce API esplicite e configurabili. Questo consente di sostituire i componenti con impatto limitato sul resto del sistema, a condizione di mantenere compatibilità dei contratti di scambio. Il contributo sperimentale è un

proof-of-concept verificabile con un set minimo di misure soggettive e osservazioni qualitative, utile a stimare comprensibilità, fluidità e credibilità dell'interazione. Il contributo metodologico è la documentazione riproducibile delle scelte progettuali, con criteri di logging, tracciamento dei tempi e selezione motivata delle misure di *user experience*. Questa impostazione è coerente con i principali lavori sul *dialogue management*, che distinguono approcci *finite-state*, *frame-based* e *agent-based* e raccomandano maggiore standardizzazione nelle metriche tecniche e nella valutazione dell'esperienza utente [4].

1.4 Panoramica del sistema e flusso end-to-end

SOULFRAME adotta un'architettura client–server pensata per supportare dialogo vocale naturale in un ambiente immersivo 3D. Il client gestisce rendering della scena, rappresentazione dell'agente embodied tramite avatar animato e acquisizione dei segnali utente. La cattura audio non è continua: parte quando l'utente tiene premuto il comando *push-to-talk* e termina al rilascio. La cattura video da fotocamera è invece usata in fase iniziale per configurare e salvare il profilo avatar. Il backend mantiene lo stato conversazionale e orchestra la pipeline linguistica e cognitiva: gestione del contesto, recupero di informazioni con *RAG*, generazione con LLM e integrazione di descrizioni di immagini e testi estratti via OCR. Questa separazione permette di tenere sul client le funzioni sensibili al frame-rate e sul backend i moduli più onerosi e soggetti a evoluzione.

Il flusso end-to-end parte dalla voce dell'utente e ritorna alla risposta audiovisiva dell'avatar. Dopo il rilascio del tasto *push-to-talk*, l'audio viene trascritto dal modulo STT. Il testo viene poi inviato al servizio di chat del backend, che funge da orchestratore RAG/LLM. Quando la memoria dell'avatar è presente, il servizio prova a recuperare contesto rilevante con ricerca ibrida; quando non ci sono ricordi utili, la risposta viene generata senza supporto documentale aggiuntivo. Il contesto può essere arricchito con descrizioni delle immagini e con testo proveniente da documenti acquisiti via OCR. Su questa base, il LLM produce la risposta, che viene sintetizzata dal TTS. In parallelo, il client anima l'avatar usando il profilo visivo creato in onboarding e regole di comportamento coerenti con il contesto dialogico, così da mantenere allineamento tra contenuto verbale e resa visiva.

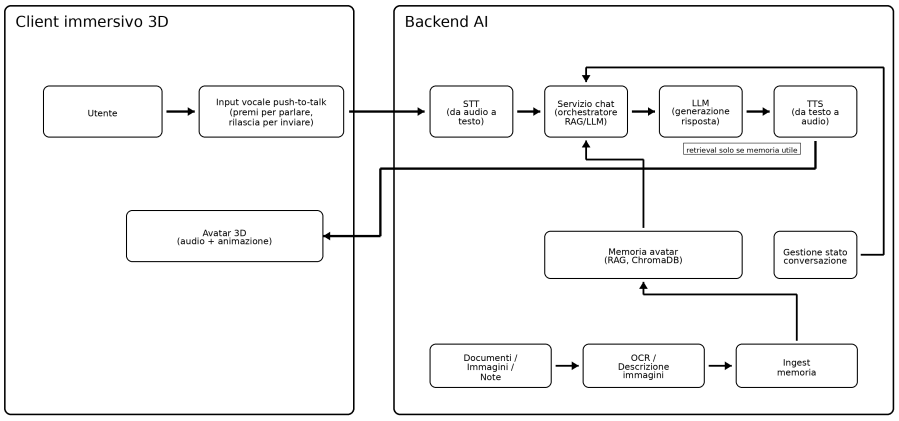

Fig. 1.3: Flusso end-to-end del sistema SOULFRAME: dall'input vocale dell'utente alla risposta dell'agente embodied.

La latenza totale dipende dalla somma delle latenze dei singoli moduli e dalla gestione a turni del *push-to-talk*. L'obiettivo non è eliminare ogni variabilità, ma mantenere continuità tra fine enunciato e risposta dell'agente con tempi percepiti stabili. Il backend coordina gli stati operativi e gestisce interruzioni o riformulazioni dell'utente senza perdere coerenza.

La modularità è un principio guida, ma nel senso operativo del progetto: i servizi sono separati, indirizzabili e configurabili, e possono essere sostituiti mantenendo coerenti endpoint e payload. In questo modo si possono confrontare varianti senza riprogettare tutto il sistema client. Figura 1.3 riassume la catena di elaborazione e la separazione di responsabilità tra client e backend.

1.5 Metodo di lavoro e struttura della tesi

Lo sviluppo di SOULFRAME segue un approccio iterativo basato su prototipazione incrementale. La scelta serve a ridurre il rischio tecnico tipico dei sistemi che combinano vincoli di reattività in interazione a turni (*push-to-talk*), elaborazione linguistica e interazione immersiva. Il lavoro procede per integrazioni successive: prima si valida ogni modulo in isolamento, poi si integra la pipeline completa e si verifica il comportamento in scenari via via più complessi. Questo metodo rende più facile individuare colli di bottiglia e problemi di sincronizzazione

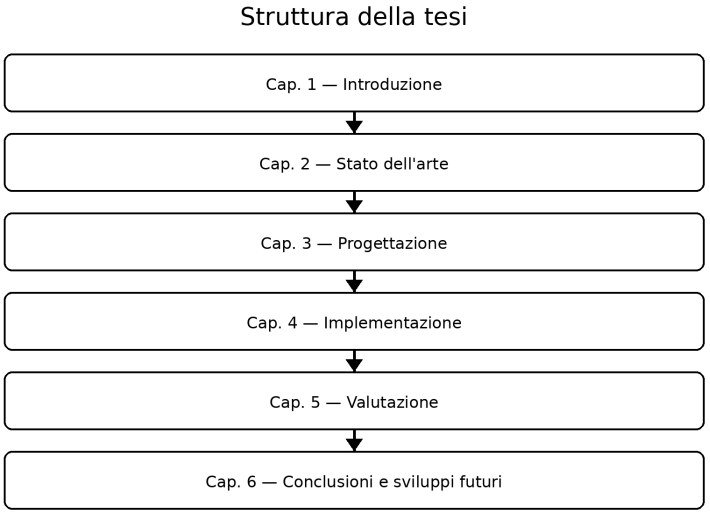

Fig. 1.4: Struttura della tesi e organizzazione dei capitoli.

in fase precoce. Durante tutto il processo vengono tracciate scelte, compromessi e dipendenze per mantenere l'evoluzione del sistema leggibile e replicabile.

La tesi è organizzata in sei capitoli. Il Capitolo 1 introduce problema, perimetro e obiettivi e fornisce la visione d'insieme del sistema. Il Capitolo 2 presenta fondamenti e stato dell'arte su VR, ECA vocali e approcci di memoria conversazionale come RAG, includendo OCR e descrizione immagini come fonti di contesto. Il Capitolo 3 descrive l'architettura di SOULFRAME, i macro-componenti e i criteri progettuali. Il Capitolo 4 entra nelle scelte implementative e tecnologiche, inclusa l'integrazione a microservizi tramite API, la gestione dello stato conversazionale e la resa embodied dell'agente. Il Capitolo 5 riporta test e valutazione, con verifica funzionale della pipeline e stima della qualità percepita. Il Capitolo 6 conclude il lavoro, discute limiti del prototipo e propone sviluppi futuri.

Figura 1.4 offre una vista sintetica della struttura e della progressione logica tra capitoli.

2. FONDAMENTI E STATO DELL'ARTE

2.1 Speech-to-Text (STT)

2.1.1 Pipeline ASR: acquisizione, preprocessing, decoding

Contenuto in preparazione.

2.1.2 Metriche e limiti (es. WER, rumore, accenti)

Contenuto in preparazione.

2.2 Modelli conversazionali (LLM)

2.2.1 Funzionamento generale e gestione del contesto

Contenuto in preparazione.

2.2.2 Prompting, parametri, allucinazioni

Contenuto in preparazione.

2.3 Embedding e Retrieval

2.3.1 Rappresentazione vettoriale del testo

Contenuto in preparazione.

2.3.2 Similarity search e top-k retrieval

Contenuto in preparazione.

2.4 Retrieval-Augmented Generation (RAG)

2.4.1 Ingestion, chunking, indexing

Contenuto in preparazione.

2.4.2 Grounding della risposta e riduzione allucinazioni

Contenuto in preparazione.

2.4.3 Limiti e trade-off

Contenuto in preparazione.

2.5 Text-to-Speech (TTS) e voice cloning

2.5.1 Pipeline di sintesi

Contenuto in preparazione.

2.5.2 Qualità percepita e latenza

Contenuto in preparazione.

2.5.3 Panoramica dei modelli TTS moderni

Contenuto in preparazione.

2.5.4 Criteri di valutazione dei modelli TTS

Contenuto in preparazione.

2.6 ASGI, FastAPI e server applicativi Python

2.6.1 Dallo standard WSGI ad ASGI

Contenuto in preparazione.

2.6.2 Ruolo di Uvicorn nella catena richiesta-risposta

Contenuto in preparazione.

2.6.3 Perché FastAPI richiede un server ASGI

Contenuto in preparazione.

2.6.4 Confronto ad alto livello tra Uvicorn, Gunicorn e Waitress

Contenuto in preparazione.

2.7 Architetture di pubblicazione API

2.7.1 Binding su loopback e isolamento dei micro-servizi

Contenuto in preparazione.

2.7.2 Reverse proxy, terminazione TLS e routing per path

Contenuto in preparazione.

2.7.3 Separazione tra frontend statico e backend dinamico

Contenuto in preparazione.

2.8 CORS e sicurezza nel browser

2.8.1 Same-Origin Policy e richieste cross-origin

Contenuto in preparazione.

2.8.2 Preflight OPTIONS: quando parte e cosa verifica

Contenuto in preparazione.

2.8.3 Significato operativo di allow_origins e allow_credentials

Contenuto in preparazione.

2.8.4 Differenza tra CORS e CSRF

Contenuto in preparazione.

2.9 Sistemi vocali conversazionali end-to-end

2.9.1 Architetture tipiche client-server

Contenuto in preparazione.

2.9.2 Gap che il progetto affronta

Contenuto in preparazione.

3. ARCHITETTURA E TECNOLOGIE UTILIZZATE

3.1 Requisiti del sistema

3.1.1 Requisiti funzionali

Contenuto in preparazione.

3.1.2 Requisiti non funzionali

Contenuto in preparazione.

3.2 Architettura generale di SOULFRAME

3.2.1 Vista d'insieme componenti frontend/backend

Contenuto in preparazione.

3.2.2 Flusso end-to-end audio → testo → risposta → audio

Contenuto in preparazione.

3.2.3 Modalità locale e modalità produzione

Contenuto in preparazione.

3.3 Frontend Unity WebGL

3.3.1 Gestione stati UI

Contenuto in preparazione.

3.3.2 Gestione avatar

Contenuto in preparazione.

3.3.3 Acquisizione audio lato client

Contenuto in preparazione.

3.3.4 Adattabilità desktop e mobile/touch

Contenuto in preparazione.

3.4 Backend AI a micro-servizi (FastAPI)

3.4.1 Servizio Whisper

Contenuto in preparazione.

3.4.2 Servizio RAG (Ollama + ChromaDB)

Contenuto in preparazione.

3.4.3 Servizio Coqui TTS

Contenuto in preparazione.

3.4.4 Avatar Asset Server

Contenuto in preparazione.

3.4.5 Porte, contract API e confini tra servizi

Contenuto in preparazione.

3.5 Modelli e librerie scelte

3.5.1 Whisper

Contenuto in preparazione.

3.5.2 LLM (es. Llama via Ollama)

Contenuto in preparazione.

3.5.3 Embedding model

Contenuto in preparazione.

3.5.4 Vector store

Contenuto in preparazione.

3.5.5 Confronto tra modelli TTS considerati

Contenuto in preparazione.

3.5.6 XTTS v2

Contenuto in preparazione.

3.5.7 Motivazioni della scelta finale (XTTS v2)

Contenuto in preparazione.

3.6 Orchestrazione servizi e deploy

3.6.1 Setup locale Windows

Contenuto in preparazione.

3.6.1.1 Ruolo di setup_soulframe_windows.bat

Contenuto in preparazione.

3.6.1.2 Ruolo di ai_services.cmd e gestione processi

Contenuto in preparazione.

3.6.2 Setup server Ubuntu

Contenuto in preparazione.

3.6.2.1 Fase di bootstrap e fase di provisioning

Contenuto in preparazione.

3.6.2.2 Layout filesystem: /opt, /etc, /usr/local/bin

Contenuto in preparazione.

3.6.2.3 Virtual environment e dipendenze Python

Contenuto in preparazione.

3.6.3 Systemd nel progetto

Contenuto in preparazione.

3.6.3.1 Unit file dei micro-servizi e mapping porte

Contenuto in preparazione.

3.6.3.2 soulframe.target e wrapper soulframe-ollama.service

Contenuto in preparazione.

3.6.3.3 Policy di restart, boot automatico e journald

Contenuto in preparazione.

3.6.4 Caddy come reverse proxy e static server

Contenuto in preparazione.

3.6.4.1 Serving della build WebGL

Contenuto in preparazione.

3.6.4.2 Routing /api/* verso servizi locali

Contenuto in preparazione.

3.6.4.3 Validazione configurazione e gestione log

Contenuto in preparazione.

3.6.5 Auto-shutdown per inattività

Contenuto in preparazione.

3.6.5.1 Timer soulframe-idle-shutdown.timer

Contenuto in preparazione.

3.6.5.2 Logica di inattività, SSH tracking e DRY_RUN

Contenuto in preparazione.

3.6.6 Helper operativi

Contenuto in preparazione.

3.6.6.1 sfctl

Contenuto in preparazione.

3.6.6.2 sfadmin

Contenuto in preparazione.

3.6.6.3 sfurl

Contenuto in preparazione.

3.6.7 Update, backup e idempotenza

Contenuto in preparazione.

3.6.7.1 Workflow aggiornamento Build e backend

Contenuto in preparazione.

3.6.7.2 Backup e pulizia sicura della update dir

Contenuto in preparazione.

3.6.7.3 Estensione del sistema con nuovi micro-servizi

Contenuto in preparazione.

4. SVILUPPO DEL PROGETTO: IMPLEMENTAZIONE E CRITICITA'

4.1 Implementazione frontend

4.1.1 UIFlowController e navigazione

Contenuto in preparazione.

4.1.2 Setup voce

Contenuto in preparazione.

4.1.3 Setup memoria

Contenuto in preparazione.

4.1.4 MainMode conversazionale

Contenuto in preparazione.

4.1.5 Gestione input desktop e input touch

Contenuto in preparazione.

4.2 Implementazione backend

4.2.1 Endpoint STT

Contenuto in preparazione.

4.2.2 Endpoint RAG e chat

Contenuto in preparazione.

4.2.3 Endpoint TTS e streaming

Contenuto in preparazione.

4.2.4 Integrazione TTS nel flusso applicativo

Contenuto in preparazione.

4.2.5 Endpoint avatar import/cache/list

Contenuto in preparazione.

4.3 Flussi di richiesta end-to-end

4.3.1 Flusso completo di una chiamata /api/rag/recall

Contenuto in preparazione.

4.3.2 Flusso CORS con preflight OPTIONS

Contenuto in preparazione.

4.3.3 Flusso /api/tts e gestione risposta audio

Contenuto in preparazione.

4.3.4 Flusso ingest_file con OCR

Contenuto in preparazione.

4.4 Integrazione tra moduli

4.4.1 Orchestrazione chiamate tra frontend, proxy e micro-servizi

Contenuto in preparazione.

4.4.2 Normalizzazione endpoint locale vs produzione

Contenuto in preparazione.

4.4.3 Gestione errori, retry e fallback

Contenuto in preparazione.

4.4.4 Persistenza per avatar

Contenuto in preparazione.

4.5 Problemi affrontati e soluzioni adottate

4.5.1 Latenza e timeout

Contenuto in preparazione.

4.5.2 CORS e routing API

Contenuto in preparazione.

4.5.3 OCR e qualità dell'ingestione

Contenuto in preparazione.

4.5.4 Compatibilità dipendenze/modelli

Contenuto in preparazione.

4.5.5 Robustezza e self-healing

Contenuto in preparazione.

4.5.6 Criticità specifiche del TTS e mitigazioni

Contenuto in preparazione.

4.5.7 Differenze operative tra Windows e Ubuntu

Contenuto in preparazione.

4.6 Runbook operativo e manutenzione

4.6.1 Comandi principali per start/stop/status/logs

Contenuto in preparazione.

4.6.2 Diagnostica guasti e lettura log

Contenuto in preparazione.

4.6.3 Procedura di update e rollback

Contenuto in preparazione.

4.7 Considerazioni su sicurezza e affidabilità

4.7.1 Validazione input

Contenuto in preparazione.

4.7.2 Isolamento memoria per avatar

Contenuto in preparazione.

4.7.3 CORS permissivo: vantaggi, limiti e hardening

Contenuto in preparazione.

4.7.4 Differenza pratica tra CORS e CSRF nel progetto

Contenuto in preparazione.

4.7.5 Logging e monitoring

Contenuto in preparazione.

5. RISULTATI E VALUTAZIONE

5.1 Metodologia di valutazione

5.1.1 Scenari e setup sperimentale

Contenuto in preparazione.

5.1.2 Scenari desktop locale e server Ubuntu

Contenuto in preparazione.

5.1.3 Metriche adottate

Contenuto in preparazione.

5.2 Risultati quantitativi

5.2.1 Prestazioni STT

Contenuto in preparazione.

5.2.2 Prestazioni RAG/LLM

Contenuto in preparazione.

5.2.3 Prestazioni TTS

Contenuto in preparazione.

5.2.4 Latenza end-to-end

Contenuto in preparazione.

5.2.5 Disponibilità dei servizi e tempi di recovery

Contenuto in preparazione.

5.3 Risultati qualitativi

5.3.1 Qualità percepita delle risposte

Contenuto in preparazione.

5.3.2 Usabilità dell'interfaccia desktop e touch

Contenuto in preparazione.

5.3.3 Analisi di casi e failure cases

Contenuto in preparazione.

5.3.4 Manutenibilità operativa (setup, update, troubleshooting)

Contenuto in preparazione.

5.4 Discussione dei risultati

5.4.1 Punti di forza

Contenuto in preparazione.

5.4.2 Limiti emersi

Contenuto in preparazione.

5.4.3 Trade-off tra semplicità d'uso e complessità infrastrutturale

Contenuto in preparazione.

6. CONCLUSIONI E SVILUPPI FUTURI

6.1 Sintesi del lavoro svolto

Contenuto in preparazione.

6.2 Contributi raggiunti

Contenuto in preparazione.

6.3 Limiti del sistema

Contenuto in preparazione.

6.4 Sviluppi futuri

6.4.1 Miglioramenti architetturali

Contenuto in preparazione.

6.4.2 Miglioramenti modelli AI

Contenuto in preparazione.

6.4.3 Scalabilità e deploy

Contenuto in preparazione.

6.5 Considerazioni finali

Contenuto in preparazione.

RINGRAZIAMENTI

BIBLIOGRAFIA

[1] Mel Slater e Maria V. Sanchez-Vives. «Enhancing Our Lives with Immersive Virtual Reality». In: *Frontiers in Robotics and AI* 3 (2016), p. 74. DOI: 10.3389/frobt.2016.00074.

[2] Peter Kán, Martin Rumpelnik e Hannes Kaufmann. «Embodied Conversational Agents with Situation Awareness for Training in Virtual Reality». In: *ICAT-EGVE 2023 – International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments*. Eurographics Association, 2023, pp. 27–36. DOI: 10.2312/egve.20231310.

[3] Fang-Chen Yang et al. «Embodied Conversational Agents in Extended Reality: A Systematic Review». In: *IEEE Access* 13 (2025). DOI: 10.1109/ACCESS.2025.3566244.

[4] Liliana Laranjo et al. «Conversational agents in healthcare: a systematic review». In: *Journal of the American Medical Informatics Association* 25.9 (2018), pp. 1248–1258. DOI: 10.1093/jamia/ocy072.