CHARLES: EVOLUZIONE E STATO DELL'ARTE DI UNO STRUMENTO DI COMUNICAZIONE

Edoardo Bontà

Università di Urbino Carlo Bo
edoardo.bonta@uniurb.it

— FULL PAPER —

***ARGOMENTO**: Sviluppo di plugin, temi e soluzioni tecniche.*

Abstract

L'applicazione web ChaRLeS, Chat Room Learning System, è una chat testuale sviluppata per favorire l'interazione sincrona tra studenti e docenti in ambito scolastico e universitario. Essendo basata sullo standard LTI, Learning Tools Interoperability, ChaRLeS è compatibile con diversi Learning Management Systems, tra cui Moodle. Rispetto ad altre applicazioni di chat, fra le caratteristiche distintive di ChaRLeS vi sono la gestione della comunicazione a più canali e la selezione in tempo reale degli interventi dei partecipanti in modo da produrre, al termine di ogni sessione, un documento testuale sintetico e ben strutturato per una consultazione efficace degli argomenti discussi. Questo articolo presenta l'evoluzione e lo stato dell'arte di ChaRLeS oltre agli aspetti che ne renderebbero interessante l'integrazione con le odierne tecnologie di Intelligenza Artificiale.

Keywords – Moodle, Chat, LTI, comunicazione multicanale, aula virtuale, testo.

1 INTRODUZIONE

ChaRLeS [8] è una chat testuale che ricrea virtualmente un'aula scolastica o universitaria, ovvero è una applicazione web real-time progettata per favorire la comunicazione sincrona tra i partecipanti e per agevolare la moderazione da parte dei docenti durante una sessione di chat. Essa è anche in grado di produrre una traccia facilmente leggibile della discussione avvenuta in chat, accessibile in modalità asincrona da chi non avesse partecipato alla sessione. Una delle principali motivazioni che negli anni 2015 e 2016 portò alla ideazione e alla realizzazione di ChaRLeS, fu la volontà di creare una alternativa più solida rispetto alla chat nativa della piattaforma *Moodle* [17]. In effetti, proprio il plugin ufficiale di attività *Chat* [18], già allora poco usabile, venne successivamente deprecato dalla comunità di sviluppo di Moodle [19] a partire dalla versione 4.4, fino alla rimozione definitiva nella versione 4.6 della piattaforma rilasciata a dicembre 2024.

L'intento del nostro progetto era quello di creare uno strumento molto leggero ma efficace, compatibile con Moodle e con altri Learning Management System, in breve LMS, che supportassero lo standard LTI [14]. Volevamo anche garantire un'ampia portabilità della applicazione sia lato server, su host con differenti sistemi operativi, sia lato client, su qualsiasi dispositivo dotato di browser in grado di supportare HTML5/*WebSocket* e senza alcuna necessità di software aggiuntivo installato. Inoltre, dal punto di vista funzionale e di usabilità, l'obiettivo era quello di creare uno strumento simile all'applicazione di chat *Land of Learning*, o *LoL*, *Classroom* [21], integrata su piattaforma proprietaria e basata su tecnologie già allora obsolete, ma con un modello di comunicazione multicanale molto apprezzato dagli studenti e dai docenti dei primi corsi di e-learning della Università di Urbino.

1.1 Il contesto storico

Dalla installazione della prima versione funzionante di ChaRLeS sui server del nostro Ateneo ad oggi sono passati quasi dieci anni e sono trascorse varie "ere" tecnologiche e culturali. Ad esempio, l'avvento di HTML5 e del protocollo *WebRTC* – che rese possibile l'esecuzione di videochiamate direttamente su

browser senza l'ausilio di client dedicati oppure di vecchie tecnologie come *Flash* o *Silverlight* – contribuì alla rapida diffusione delle applicazioni web di videoconferenza e delle chat audio/video, più ricche e immediate nelle interazioni di quelle puramente testuali. A partire dal 2016, strumenti come *Blackboard Collaborate* [7] e successivamente *BigBlueButton* [6] abbracciarono questa nuova tecnologia e ampliarono la propria diffusione nelle piattaforme LMS verso cui offrivano già una forte integrazione a livello di gestione delle aule virtuali e di trasferimento delle autorizzazioni dei partecipanti.

L'arrivo della pandemia COVID-19 costrinse alla reclusione forzata la popolazione mondiale fin dai primi mesi del 2020, spingendola ad un ricorso massivo allo smartworking e ai software di videoconferenza e di collaborazione remota. I software vennero impiegati per svolgere attività lavorative di vario genere, oltre che per attività educative scolastiche e universitarie come lezioni ed esami [3, 4]. In questo scenario, le tecnologie che ebbero la meglio furono quelle fortemente scalabili e in cloud, cioè quelle più resistenti ai sovraccarichi di traffico in rete, nonché quelle più convenienti economicamente quando applicate a grandi numeri di utenti. Ciò avvenne anche a discapito della funzionalità, ovvero molte organizzazioni educative adottarono strumenti di comunicazione e videoconferenza generalisti, non particolarmente mirati alla gestione di aule virtuali né alla moderazione degli studenti o alla integrazione con piattaforme LMS. Nacquero vari strumenti "artigianali", ossia plugin realizzati da volontari in diverse università italiane e mondiali, per mettere in comunicazione i propri LMS con i software di videoconferenza, normalmente forniti come *Software as a Service*, SaaS, da grandi compagnie. Allo stesso tempo, per rispondere a una domanda in rapida ascesa, fra le grandi compagnie ci furono quelle che costruirono [9] e quelle che rafforzarono [11] il proprio ecosistema didattico in cloud, integrando strumenti di videoconferenza con altri strumenti per attività educative sincrone e asincrone, a volte anche perdendo di vista gli standard e la priorità della integrazione con LMS esterni e non proprietari [5].

L'era successiva, che parte da fine 2022 con il rilascio di *ChatGPT* [20], è quella che viviamo al momento della scrittura di questo articolo, ovvero la grande diffusione della Intelligenza Artificiale, IA, e dei *Large Language Model*, LLM. Per quanto gli strumenti dominanti nelle interazioni sincrone remote siano ancora i software di videoconferenza in cloud, se si osservano alcune tendenze come le traduzioni simultanee tra diversi linguaggi parlati dai partecipanti, le trascrizioni o le sintesi delle sessioni di chat, l'uso crescente di assistenti virtuali e agenti IA che attingono informazioni da una base di conoscenza, appare ragionevole pensare che lo streaming testuale abbia riacquistato importanza, seppure si trovi spesso "nascosto tra le pieghe" della chat.

Va infine ricordato che nell'arco di questi anni, alcuni strumenti di comunicazione "debolmente sincrona" – ovvero nei quali non è necessariamente richiesta una risposta immediata dei partecipanti, come ad esempio *Whatsapp* o *Telegram*, oppure *Slack* in ambito aziendale – pur essendo basati prevalentemente su messaggi testuali, sono sempre rimasti in uso tra la gente, grazie anche a caratteristiche quali la persistenza [23] e la facilità di ricerca dei messaggi. Questo può essere visto come un indicatore del fatto che non è tanto il testo ad essere diventato obsoleto negli strumenti di comunicazione, quanto invece specifici strumenti e contesti di utilizzo della comunicazione testuale.

1.2 Panoramica

Questo articolo affronterà gli aspetti progettuali e le tecnologie impiegate per lo sviluppo della applicazione web ChaRLeS, mettendo in evidenza le trasformazioni avvenute nel corso di un decennio, dalla prima implementazione dello strumento di comunicazione ad oggi.

L'articolo è organizzato come segue. Nella sezione 2 viene esposta la struttura dello strumento di chat testuale ChaRLeS, descrivendone il modello di comunicazione, l'interfaccia utente e l'architettura software. Nella sezione 3 viene discussa l'evoluzione della tecnologia specifica utilizzata per lo sviluppo di ChaRLeS, riportandone gli aggiornamenti e le scelte progettuali. Nella sezione 4 sono presentate alcune considerazioni riguardanti lo stato dell'arte e le direzioni future in relazione alle potenzialità della comunicazione testuale, alla sostituzione di partecipanti con strumenti di intelligenza artificiale e allo sviluppo di codice sorgente. Infine, la sezione 5 conclude l'articolo evidenziando le peculiarità ed i possibili sviluppi e impieghi futuri della applicazione ChaRLeS.

2 CHARLES: STRUTTURA DELLA CHAT DI TESTO

Questa sezione descriverà, nell'ordine, la soluzione di inefficienze caratteristiche delle chat testuali e il modello di comunicazione di ChaRLeS, l'interfaccia utente e, infine, l'architettura del sistema software adottata per lo sviluppo della applicazione.

2.1 La soluzione di inefficienze in chat testuali e il modello di comunicazione

L'applicazione ChaRLeS è stata progettata cercando di prevenire le inefficienze tipiche dei tradizionali strumenti di chat testuale [22]. Le diverse inefficienze sono elencate sotto, assieme alle relative soluzioni che definiscono il modello di comunicazione della nostra chat.

3. **Mancanza di riconoscimento** causata dall'uso incontrollato di nickname da parte dei partecipanti. Questo problema è risolto attraverso l'uso dello **standard LTI**, importando nominativi e rispettivi ruoli (docenti o studenti) direttamente dalla piattaforma Moodle o dal LMS a cui ChaRLeS si interfaccia, ove di norma l'uso di nickname incomprensibili non è consentito.
4. **Mancanza di "indicatori di intenzione"**, cioè mancanza di chiarezza nel comprendere a chi è indirizzato un messaggio e quale sia il suo scopo. Questo problema è risolto grazie all'uso di **canali multipli** per differenti comunicazioni, ovvero i canali ***Question*** (o *Intervention*), ***Lecture***, ***Service***, oltre al canale ***Private*** per inviare messaggi a destinatari selezionati. La disponibilità dei canali è anche subordinata al ruolo dei partecipanti.
5. **Mancanza di controllo sull'ordine degli interventi**, che può provocare disorganizzazione, sovrapposizione e confusione nei contributi dei partecipanti durante la sessione di chat. Questo problema è risolto grazie al **meccanismo di gestione degli interventi**, o **domande**, che consente a tutti i partecipanti di preparare e pubblicare i propri contributi simultaneamente. Spetterà poi al docente/moderatore decidere quando pubblicare uno specifico contributo dal canale delle domande, creando in questo modo un discorso di chat coerente.
6. **Inefficienza della digitazione**, che limita la comunicazione in base alle capacità di digitazione degli utenti. Questo problema è risolto sia dal meccanismo di pubblicazione delle domande, che consente a tutti gli utenti di preparare i contributi al proprio ritmo e di pubblicarli simultaneamente senza il rischio di creare rumore comunicativo, sia dai sistemi di dettatura vocale ormai presenti in qualsiasi dispositivo – nativamente negli smartphone – in alternativa alla tastiera fisica o virtuale.
7. **Mancanza di contesto**, causata dal fatto che tutta la comunicazione, indipendentemente dal suo contesto semantico, viene scritta all'interno della stessa chat testuale. Questo problema viene evitato perché i contributi sono organizzati in modo coerente nel thread della chat, come anche specificato al punto 3.
8. **Basso rapporto "segnale/rumore**" derivante dal fatto che le informazioni didattiche importanti sono mescolate a informazioni di socializzazione o di servizio, molto utili per la creazione di una comunità di apprendimento in un contesto di interazione sincrona ma che non necessitano di essere registrate. Questo problema è risolto suddividendo rigorosamente i diversi flussi di comunicazione e filtrando dai canali *Lecture* e *Question* (che vengono registrati) il "rumore" legato ai canali *Service* e *Private*, ove i partecipanti conversano liberamente senza l'intervento del docente/moderatore della chat.
9. **Inutilità della traccia della sessione**, cioè scarsa leggibilità dei testi registrati a causa della mancanza di coerenza, della frammentazione dei contributi e delle interruzioni causate dai messaggi in arrivo. Questo problema è risolto in quanto l'usabilità della traccia della chat è garantita dalla produzione in tempo reale di un **file di log HTML** (o ***chatlog***) che include solo le interazioni rilevanti dei canali *Lecture* e *Question*, escludendo le comunicazioni irrilevanti dei canali *Service* e *Private*.

Una volta affrontate e risolte le limitazioni dei sistemi di interazione testuale, l'applicazione ChaRLeS può mostrare i propri punti di forza anche rispetto a soluzioni di videoconferenza più moderne e blasonate. In particolare: (i) il testo utilizza una larghezza di banda molto ridotta, quindi ChaRLeS può essere agevolmente utilizzata anche su reti particolarmente "lente" o congestionate; (ii) i testi risultanti dalle sessioni di chat sincrone sono automaticamente prodotti, archiviati e facilmente reperibili (e

auspicabilmente studiati) dalla piattaforma Moodle o dal LMS con cui ChaRLeS si integra; (iii) ChaRLeS può interfacciarsi in modo trasparente con la maggior parte dei dispositivi di input/output, anche su sistemi mobili (ad esempio screen reader, dispositivi di input vocale, ecc.), consentendo una buona accessibilità alle attività online ad un elevato numero di persone, incluse coloro che hanno disabilità di varia natura.

2.2 L'interfaccia utente

L'applicazione web ChaRLeS è il risultato del modello di comunicazione descritto nella precedente sottosezione e della architettura del sistema software che sarà sinteticamente descritta più avanti.

La Fig. 1 raffigura un tipico scenario di utilizzo di un'aula virtuale in ChaRLeS, all'interno della quale si svolge una conversazione tra due partecipanti. Uno di loro, Steve Smith, è uno studente, mentre John Doe è un docente. Il punto di vista della figura è quello dello studente, che in questo esempio utilizza solo i canali *Question* e *Private*. Il docente, invece, scrive un messaggio di servizio, nel canale *Service*, per tutti i partecipanti, poi pubblica una domanda dello studente e successivamente risponde alla stessa domanda attraverso il canale *Lecture* (che è disabilitato per lo studente e non appare nella sua list-box "*channel*"). In seguito, il docente e lo studente continuano la loro conversazione utilizzando il canale *Private* per comunicare tra loro.

Per quanto riguarda la cronologia dei messaggi della sessione di chat, solo le domande pubblicate dal canale *Question* e i messaggi del canale *Lecture* (ad esempio, le risposte del docente) verranno salvati nel registro della sessione di chat, o *chatlog* – a meno che le impostazioni non siano state modificate rispetto ai valori predefiniti. I messaggi dei canali *Service* e *Private*, invece, verranno conservati in una cache del server che sarà svuotata dopo un determinato periodo di inattività di tutti i partecipanti entrati nella stessa aula – ossia 20 minuti, per impostazione predefinita.

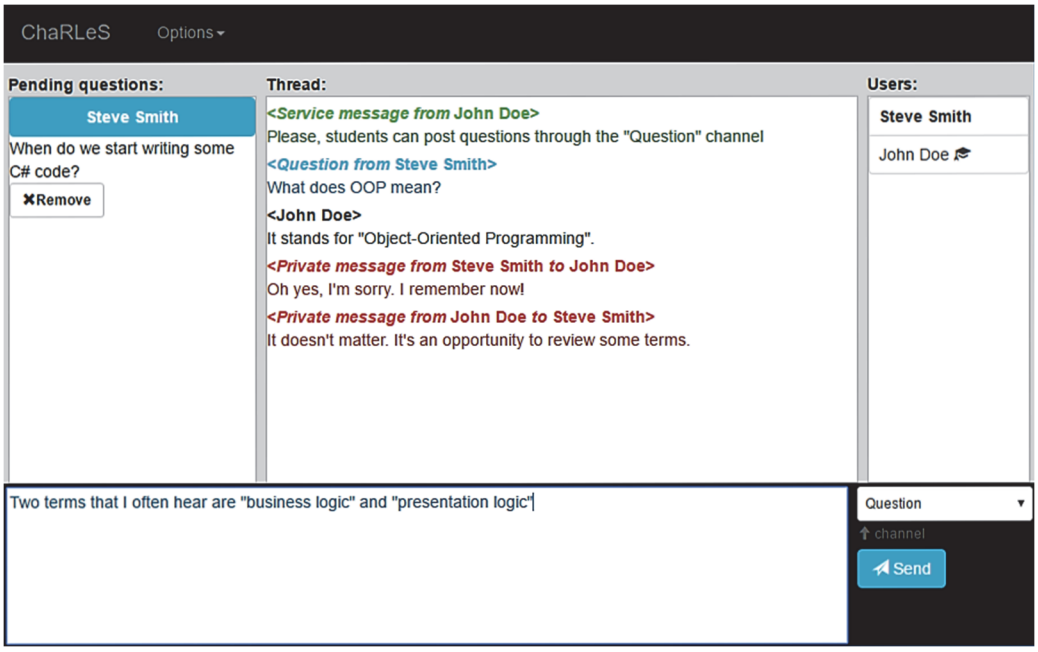

Figura 1 – Schermata di una sessione di chat con ChaRLeS

Ogni nuovo partecipante che entra nell'aula riceverà immediatamente gli aggiornamenti, ovvero tutte le domande in sospeso e tutti i messaggi del canale *Lecture*, assieme ai messaggi del canale *Service* memorizzati nella cache, ma non vedrà i messaggi del canale *Private* indirizzati ad altri partecipanti. Tuttavia, quando l'intera cronologia dei messaggi supererà una determinata quantità di memoria – cioè 2 MB, per impostazione predefinita – il partecipante riceverà solo gli ultimi messaggi (≤ 2 MB). In ogni caso, l'intera cronologia della sessione sarà disponibile a tutti gli utenti autorizzati nella pagina *viewsession*, dalla quale il *chatlog* potrà essere visualizzato e scaricato.

Un aspetto interessante della usabilità di ChaRLeS è il fatto che una domanda/intervento pendente può essere assimilato alla "mano alzata", usata per richiedere la parola nei comuni strumenti di

videoconferenza. In questo contesto, però, il contenuto è specificato in chiaro e risulta leggibile prima della pubblicazione. Ciò permetterà al docente di scegliere l'ordine di pubblicazione oppure di escludere un determinato intervento nel caso in cui questo abbia poco senso, sia offensivo o sia già stato affrontato in precedenza. Allo stesso tempo, lo studente sarà in grado di verificare se ci sono già interventi pendenti, di altri partecipanti, simili a quanto avrebbe intenzione di scrivere, potendo quindi riconsiderare o aggiustare il proprio contributo.

2.3 L'architettura del sistema

Come già menzionato, ChaRLeS è un'applicazione web real-time e, in particolare, utilizza il protocollo WebSocket su HTTP(S) per inviare messaggi dal server ai client in modalità broadcast o selettiva.

Essendo il backend sviluppato in C#, per gestire le comunicazioni real-time lato server abbiamo scelto la libreria *SignalR* [16]. Tale componente è supportata ufficialmente dalla piattaforma open-source .NET che permette anche di ospitare l'applicazione su diversi sistemi operativi come Windows, Linux e macOS.

Lato client, il frontend è sviluppato in HTML e JavaScript, ed essendo completamente disaccoppiato dal backend, non richiede conoscenze di sviluppo .NET/C# per la manutenzione dell'interfaccia utente. Nel frontend non sono stati usati framework, e le librerie impiegate nel codice JavaScript sono *jQuery*, per la definizione di funzioni generiche, e *Bootstrap* per gestire stile, layout e portabilità su dispositivi mobili.

Infine, è possibile usare opzionalmente il web server *Nginx* come *reverse proxy* per unificare diversi percorsi e applicazioni web in un'unica porta HTTP e per abilitare comunicazioni HTTPS tramite un singolo certificato SSL/TLS, in virtù dell'ottimo supporto che il server è in grado di offrire a WebSocket e connessioni HTTP(S) persistenti.

In Fig. 2 è riportata l'architettura complessiva della applicazione ChaRLeS in cui si può osservare il disaccoppiamento tra il servizio in backend e l'applicazione client, nonché il ruolo del reverse proxy nella unificazione dei vari percorsi.

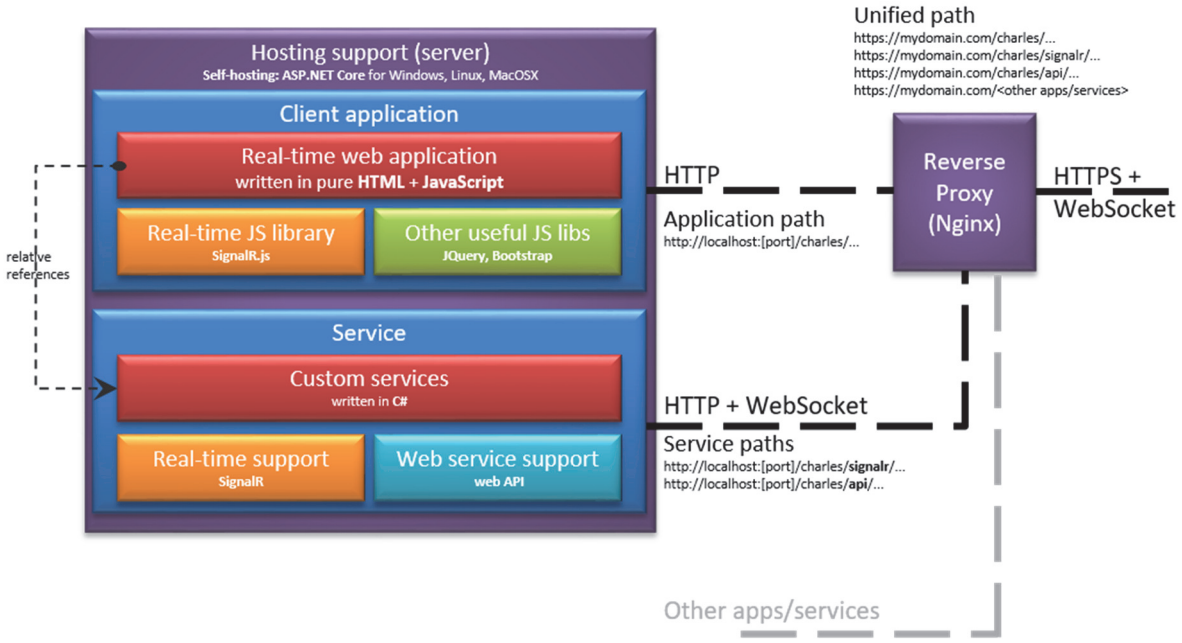

Figura 2 – Architettura della applicazione web ChaRLeS

3 EVOLUZIONE DELLE TECNOLOGIE

Per quanto ChaRLeS sia stato progettato a "prova di passato", garantendone al momento dello sviluppo la retrocompatibilità con vecchie tecnologie e browser già obsoleti, non è stato possibile renderlo altrettanto a "prova di futuro" rispetto alla evoluzione tecnologica e agli aggiornamenti di linguaggi e strumenti utilizzati. In realtà, alcune scelte si sono rivelate particolarmente azzeccate, mentre altre

hanno richiesto vari aggiornamenti per adeguarsi alle nuove funzionalità delle piattaforme e delle librerie ufficiali impiegate, portandoci anche ad accettare compromessi sulla retrocompatibilità e abbandonando quindi il supporto a tecnologie abbastanza datate.

Nelle prossime sottosezioni verranno riportati i principali aggiornamenti tecnologici da noi adottati per adeguare ChaRLeS alla evoluzione di linguaggi, piattaforme e librerie, nonché le scelte progettuali che si sono rivelate più o meno positive alla prova del tempo.

3.1 Aggiornamento dello stack tecnologico

A. Le tecnologie .NET (backend)

La tecnologia usata in origine per lo sviluppo della applicazione web ChaRLeS era *DNX*, ovvero un ambiente di esecuzione, o *runtime*, per ASP.NET 5.0 disponibile in Visual Studio 2015. Tale tecnologia fu gradualmente abbandonata da Microsoft dopo che ebbe acquisito Xamarin nel 2016. A metà del 2017 divenne per noi necessario eseguire la transizione verso l'ambiente *.NET Core*, cosa che richiese un notevole sforzo per l'adeguamento del codice sorgente e delle dipendenze dalle librerie native, ma che apportò vari giovamenti alla manutenibilità e alla più semplice portabilità dei file eseguibili su piattaforme differenti da Windows, in particolare su Linux. Contestualmente, venne anche adottata la specifica *.NET Standard* 1.3 per la definizione delle librerie sviluppate per il backend. Negli anni successivi, le versioni di NET Core e quelle di .NET Standard usate da ChaRLeS vennero aggiornate, passando quindi dal *Core* alla *piattaforma* .NET e stabilizzandosi, per le librerie, sulla specifica .NET Standard 2.0.

B. La libreria SignalR (backend+frontend)

La libreria SignalR, estremamente utile per gestire ad alto livello la comunicazione bidirezionale fra client e server web, è stata costantemente aggiornata assieme alle tecnologie .NET, dalle quali è ufficialmente supportata. La caratteristica a cui dovemmo rinunciare a malincuore furono le soluzioni di fallback rispetto all'uso del protocollo WebSocket, presenti solo nelle prime versioni della libreria. In realtà, tali soluzioni alternative avevano sempre meno senso con il passare del tempo, perché tutti i browser più moderni supportavano WebSoocket senza problemi.

C. Le librerie JavaScript (frontend)

Il frontend di ChaRLeS è attualmente realizzato in HTML, CSS e JavaScirpt puro, in gergo *Vanilla JS*, senza l'uso di framework. Le librerie principali, oltre SignalR lato client, sono *Bootstrap* e *jQuery*. La prima, Bootstrap, è tuttora molto utile per la gestione del layout di pagina su PC e dispositivi mobili, in particolare smartphone e tablet. La seconda, jQuery, che aveva in origine lo scopo di agevolare la portabilità del codice JavaScript su differenti browser, è diventata ormai superflua ed è destinata ad essere eliminata nei prossimi aggiornamenti di ChaRLeS.

D. Lo standard LTI (comunicazione con LMS)

Un cambiamento fondamentale è legato alla evoluzione dello standard LTI su cui è basata la comunicazione tra ChaRLeS e la piattaforma LMS con cui essa si integra. In passato, la piattaforma Moodle utilizzava la versione LTI 1.1 – e ne mantiene tuttora la compatibilità – ma attualmente supporta la nuova versione LTI 1.3 con l'estensione *Advantage*. È importante che, nei prossimi aggiornamenti, ChaRLeS garantisca anche la compatibilità con questa ultima versione di LTI.

3.2 Scelte progettuali e "lezioni apprese"

Fra le scelte progettuali positive, ovvero quelle fatte fin dalle prime fasi della progettazione ma che sono risultate particolarmente felici anche a distanza di anni, possiamo annoverare le seguenti.

- Scelta di raccogliere più codice possibile, lato backend, in librerie basate su .NET Standard 2.0. Questo permette di mantenere le librerie sempre compatibili con il codice che ne fa uso, indipendentemente dalla evoluzione del linguaggio (C#) e delle piattaforme .NET.
- Scelta di salvare i messaggi della chat in formato HTML, accodando ogni nuovo messaggio al chatlog dell'aula virtuale. Questo consente di recuperare agevolmente lo storico dei messaggi all'ingresso di un nuovo utente nella stessa aula, applicando immediatamente gli stili CSS per distinguere visivamente i differenti messaggi. Ancora più importante, permette di conservare

metadati nei tag di ogni messaggio – fra cui, ad esempio, l'orario di scrittura del messaggio – senza doverli nascondere o filtrare esplicitamente prima del rendering del chatlog HTML.

- Sincronizzazione degli accessi ai chatlog e ai percorsi delle aule virtuali mediante dizionari chiave-valore, e scrittura diretta dei chatlog su file. L'organizzazione dei dati su filesystem con accesso sincronizzato dai dizionari, permette una strutturazione NoSQL dei dati e la possibilità di sostituire facilmente il filesystem con un database non relazionale, scalabile, come *Redis* o *ValKey*.

Fra le lezioni apprese, invece, possiamo riportare scelte negative che hanno però indicato una direzione differente da seguire nei futuri sviluppi del software.

In questo contesto sicuramente ricadono le scelte fatte nella realizzazione di un plugin di attività per Moodle da usare come connettore con l'applicazione esterna ChaRLeS. Il plugin è dotato di un'interfaccia utente più evoluta e "amichevole" rispetto a quella del plugin nativo *External Tools*, seppure entrambi utilizzino lo standard LTI. I problemi nascono dal fatto che il plugin connettore ha vari parametri di configurazione gestiti localmente in Moodle, che devono però essere trasmessi all'applicazione esterna ChaRLeS. Lo standard LTI prevede tuttavia che la trasmissione all'applicazione esterna avvenga al momento dell'accesso di un utente, studente o docente che sia, e non al momento della configurazione locale in Moodle. Questa caratteristica crea diverse inconsistenze se la configurazione di un'aula viene modificata nel connettore in Moodle quando nella stessa aula su ChaRLeS è già in corso una sessione di chat. Abbiamo trovato degli accorgimenti per minimizzare le inconsistenze, ma alcune di queste ancora permangono. Soluzioni più stabili al problema potrebbero essere, in alternativa, le seguenti:

- evitare l'uso del plugin connettore e dei parametri locali di configurazione, ma usare direttamente External Tools, tenendo anche conto che la possibilità di personalizzazione di quest'ultimo plugin è evoluta rispetto a qualche anno fa;
- introdurre una strategia per fare in modo che al momento della configurazione dei parametri locali nel plugin connettore avvenga un accesso effettivo a ChaRLeS, possibilmente attraverso un utente fittizio creato dal connettore stesso, al fine di trasmettere immediatamente i parametri;
- basare il plugin connettore su interfacce di programmazione API anziché sul protocollo LTI, come del resto fanno comunemente molti fornitori di servizi esterni a Moodle.

4 STATO DELL'ARTE E DIREZIONI FUTURE

In questa sezione verranno riportate alcune considerazioni relative alle potenzialità legate al testo, alla sostituzione di partecipanti con strumenti di intelligenza artificiale, e allo sviluppo di codice sorgente, sempre mediante intelligenza artificiale.

4.1 Il potere del testo

Il testo è l'infrastruttura, spesso invisibile, delle applicazioni moderne. Mentre gli utenti interagiscono con interfacce grafiche, video e contenuti multimediali, dietro le quinte si svolge l'attività di elaborazione testuale che rende possibili funzionalità date spesso per scontate nelle applicazioni di tutti i giorni.

Negli attuali strumenti di videoconferenza, per esempio, la voce umana viene trasformata automaticamente in testo attraverso sistemi di riconoscimento vocale. Il processo di trascrizione in tempo reale permette così la traduzione istantanea in altre lingue, e quindi la comunicazione fluida tra partecipanti che non condividono una lingua comune. Il sistema, infatti, traduce nella lingua di destinazione il flusso testuale proveniente dalla fonte vocale di origine, poi lo riconverte in audio e lo trasmette al partecipante in ascolto. Parallelamente lo stesso sistema può generare trascrizioni complete di lezioni o riunioni, indicizzabili e ricercabili, trasformando le conversazioni in documenti permanenti consultabili. Questo offre la possibilità di cercare parole chiave all'interno di ore di registrazioni o di estrarre automaticamente i punti salienti di una discussione.

Le applicazioni di messaggistica istantanea e i social media rappresentano un altro ambito dove il testo ha un ruolo fondamentale. Al di là della comunicazione esplicita tra utenti, questi sistemi analizzano costantemente i contenuti testuali per capire l'umore generale e quali argomenti sono più discussi, o per moderare contenuti inappropriati e suggerire connessioni rilevanti. Anche il formato di trasmissione

del testo è rilevante. Messaggi codificati in JSON viaggiano attraverso protocolli WebSocket per garantire aggiornamenti in tempo reale, mentre HTML struttura la presentazione nei browser e nelle applicazioni native. Markdown, invece, permette di strutturare documenti con intestazioni, elenchi, codice e formattazione senza la complessità di markup più pesanti, garantendo leggibilità sia per gli esseri umani sia per i sistemi di elaborazione automatica.

Grazie alla recente diffusione degli strumenti di intelligenza artificiale basati su modelli linguistici di grandi dimensioni, LLM, in molti ambiti il testo supera il ruolo di semplice mezzo di trasporto o di annotazione dei contenuti. Esso diventa il fine stesso della elaborazione nonché l'operatore che la definisce, in quanto le trasformazioni linguistiche richieste e applicate sul testo possono essere espresse a loro volta in linguaggio naturale.

Una delle frontiere della elaborazione intelligente del testo è rappresentata dai sistemi RAG*, Retrieval-Augmented Generation*. Questi sistemi permettono l'accesso a basi di conoscenza specifiche da parte dei modelli linguistici LLM, affinandone la capacità generativa oltre i limiti della conoscenza che i modelli acquisiscono durante l'addestramento. Quando un utente formula una domanda, il sistema recupera i frammenti testuali più rilevanti dalla base di conoscenza e li fornisce come contesto al modello linguistico che genera risposte informate da queste evidenze specifiche. Questa architettura permette di costruire assistenti esperti in domini specializzati senza necessità di riaddestramento completo dei modelli. Lezioni ed esercitazioni tra docenti e studenti, letteratura scientifica e archivi normativi possono essere distillati in sistemi interrogabili in linguaggio naturale.

L'applicazione ChaRLeS può integrarsi perfettamente in questo contesto, gestendo quella che è la "nervatura" del sistema, ovvero la comunicazione testuale tra più parti. Ad esempio, connettendosi con le opportune tecnologie mediante interfacce di programmazione API, ChaRLeS potrebbe raccogliere basi di conoscenza raggruppandole per aule virtuali o per corsi di insegnamento, premettendo così l'uso di RAG, anche gerarchici (*HiRAG* [12]), per definire differenti livelli di conoscenza. L'applicazione avrebbe già un input informativo ben strutturato grazie al modello di comunicazione descritto in precedenza, cioè contestualizzato e privo di "rumore", e garantirebbe quindi un buon processo di raccolta e distillazione dell'informazione. Anche la traduzione simultanea, da testo a testo, risulterebbe relativamente semplice delegando l'elaborazione a strumenti esterni adeguati.

4.2 Sostituzione di un partecipante con IA

Una possibile direzione evolutiva di ChaRLeS è quella di permettere all'applicazione di sostituire opzionalmente il docente con un assistente virtuale, magari in modalità asincrona per rispondere a domande da parte di studenti riguardo a lezioni già svolte, sfruttando la base di conoscenza accumulata. Oppure si potrebbe prevedere anche una modalità sincrona, ove l'assistente è un agente IA che, oltre a rispondere alle richieste, funge anche da moderatore tra i partecipanti. Sarebbe un interessante esperimento di chat "uno-a-molti" (una macchina, molti umani), in contrapposizione al modello "uno-a-uno" con cui siamo abituati a interagire con strumenti come ChatGPT, *Claude* [2] e simili.

Un altro esperimento sociale potrebbe essere quello di introdurre l'assistente virtuale al posto di uno studente anziché del docente. L'assistente potrebbe intervenire saltuariamente nella discussione, ma senza svelare la propria identità artificiale. La chat si trasformerebbe così in un gioco per indovinare chi è il partecipante artificiale, cioè "l'impostore", creando uno scenario misto tra il test di Turing e il videogioco online *Among Us* [1].

4.3 Sviluppo mediante IA

Al di fuori del modello comunicativo e delle funzionalità, lo sviluppo di ChaRLeS dovrebbe ora giovare della disponibilità di numerosi strumenti per la generazione e la revisione del codice sorgente. Alcuni piccoli ritocchi al codice sono già stati fatti recentemente avvalendosi di *GitHub Copilot* [10]. Sarebbe comunque auspicabile esplorare le varie direzioni prospettate in questo articolo utilizzando anche strumenti complementari, come Claude *Code* ed eventualmente tool come *Lovable* [13] per la ridefinizione completa della interfaccia grafica della applicazione web.

5 CONCLUSIONI

In questo articolo sono stati affrontati l'evoluzione e lo stato dell'arte dello strumento di comunicazione ChaRLeS, una applicazione web real-time che ricrea aule virtuali in cui docenti e studenti interagiscono attraverso messaggi di testo. L'applicazione si integra con Moodle e altre piattaforme LMS attraverso il protocollo LTI, che permette di trasmettere ad ogni aula virtuale informazioni di contesto, tra cui anche le identità e i ruoli dei partecipanti. Il modello di comunicazione di ChaRLeS è costituito da quattro canali: *Lecture*, per la lezione del docente, *Question*, per gli interventi dello studente, che devono però essere moderati dal docente per essere pubblicati nel canale della lezione, *Service,* per le comunicazioni non riguardanti direttamente gli argomenti della lezione, *Private*, per la comunicazione diretta e privata tra partecipanti.

Essendo stato rilasciato circa dieci anni fa, il software è stato soggetto a vari interventi di manutenzione per contrastare l'obsolescenza delle tecnologie sulle quali era basato. Alcune scelte effettuate durante le prime fasi di progettazione e sviluppo si sono rivelate lungimiranti, ad esempio l'aderenza a .NET Standard 2.0, altre scelte hanno invece comportato alcuni problemi, ad esempio l'uso di eccessivi parametri di configurazione dell'aula nel connettore locale a Moodle, cioè il plugin basato su LTI.

È stato poi analizzato lo scenario tecnologico attuale, ove l'uso del testo nelle comunicazioni assume ancora più importanza rispetto al passato, grazie anche ai sistemi di intelligenza artificiale ed LLM che eseguono elaborazioni e trasformazioni su stream testuali. In questo contesto, ChaRLeS ha varie potenzialità di utilizzo e di espansione, tra cui i possibili casi d'uso:

- come strumento completo di interazione "semi-sincrona" o "debolmente sincrona", cioè come via di mezzo tra forum e chat, per sfruttare al meglio la persistenza dei messaggi, ma anche con l'ausilio di agenti IA a disposizione degli studenti a qualsiasi ora, e con la possibilità di interrogare la base di conoscenza e fornire riassunti o suggerimenti specifici;
- come sistema di trasporto in backend per lo smistamento e lo streaming di messaggi tra partecipanti remoti, eventualmente integrato in maniera trasparente con strumenti di comunicazione più evoluti.

In relazione al secondo punto, è interessante osservare l'analogia con il sistema *Matrix* [15], progettato per gestire comunicazioni distribuite e per interfacciarsi con differenti applicazioni. Sarà interessante, in futuro, analizzare e confrontare le peculiarità dei due sistemi, al fine di osservare quali casi d'uso possano essere più adeguati all'una o all'altra architettura.

Riferimenti bibliografici

[1] Among Us*, Amongus Play*, https://amongusplay.online/

[2] Anthropic, *Claude*, https://claude.ai

[3] Bernardo M., Bontà E., "Facing the COVID-19 Pandemic with Moodle, Collaborate, Smowl, Meet", in Proc. of the 2nd Int. Workshop on Higher Education Learning Methodologies and Technologies Online (HELMETO 2020), G. Casalino, R. Pecori editors, pp. 81-84, Bari (Italy), September 2020.

[4] Bernardo M., Bontà E., "Teaching and Learning Centers and Coordinated Technologies for an Effective Transition at COVID-19 Pandemic Time to Massive Distance Learning and Online Exams", in Journal of e-Learning and Knowledge Society 19:22-29, August 2023.

[5] Bertelli A., Carnevali F., Tebaldi L., "*Google Meet per Moodle usando le nuove API ufficiali*", in Proc. of the *20th Italian Conf. MoodleMoot Italia 2024*, P. de Waal, P. Gallo, R. Pinna, S. Rabellino editors, pp. 293-298, Viterbo (Italy), October 2024.

[6] BigBlueButton, *Virtual Classroom Software*, https://bigbluebutton.org/

[7] Blackboard, *About Blackboard Collaborate*, https://www.blackboard.com/en-eu/resources/about-blackboard-collaborate

[8] Bontà E., Torrisi G., Bernardo M., "ChaRLeS: An Open-Source Chat Room Learning System", in Proc. of the 2nd Italian Conf. on Education, Multimedia and Moodle (EMEMITALIA 2016), M. Rui editor, Genova University Press, pp. 1084-1093, Modena (Italy), September 2016.

[9] Class, *Class Collaborate is now Class for Web*, https://www.class.com/collaborate/

[10] GitHub, *GitHub Copilot - Your AI pair programmer*, https://github.com/features/copilot

[11] Google, *Classroom*, https://classroom.google.com/

[12] Huang H. et al., "Retrieval-Augmented Generation with Hierarchical Knowledge", in Findings of the Association for Computational Linguistics (EMNLP 2025), pp. 6044-6060, November 2025.

[13] Lovable, *Build something Lovable*, https://lovable.dev/

[14] LTI, *Learning Tools Interoperability (LTI)*, https://www.1edtech.org/standards/lti

[15] Matrix, *An Open Network for Secure Decentralized communication*, https://matrix.org/

[16] Microsoft, *SignalR*, https://learn.microsoft.com/en-us/aspnet/signalr/

[17] Moodle, *Welcome to the Moodle community*, https://moodle.org/

[18] MoodleDocs, *Chat Activity*, https://docs.moodle.org/405/en/Chat_activity

[19] Moodle Developers Forum, *Deprecation of the Chat Tool*, https://moodle.org/mod/forum/discuss.php?d=455659

[20] OpenAI, *ChatGPT*, https://chatgpt.com/

[21] Pigliapoco E., Torrisi G., Messina M., Bogliolo A., "LoL Classroom: A Virtual University Classroom Based on Enhanced Chats", in European Journal of Open, Distance and E-Learning (EURODL), pp. 1-9, January 2008.

[22] Vronay D., Smith M., Drucker S., "Alternative Interfaces for Chat" in Proc. Of The 12th Annual Acm Symposium on User Interface Software and Technology, pp. 19-26, Asheville, Nc, 1999.

[23] Wikipedia, *Computer-mediated communication*, https://en.wikipedia.org/wiki/Computer-mediated_communication