

UNIVERSITÀ DEGLI STUDI DI TORINO

Dipartimento di Informatica

Corso di Laurea in Informatica

Anno Accademico 2022/2023

DotLine: concezione e sviluppo di una web app per la messagistica istantanea con supporto alla crittografia end-to-end

Relatore:
Dott. Marino Segnan

Candidato:
Enrico Sola

Sommario

1. Introduzione 5

2. Contesto del progetto 7

2.1. Obiettivo generale 7

2.2. Obiettivi derivati 7

2.2.2. Gestione delle credenziali e controllo degli accessi 8

2.2.1 Garanzia di riservatezza delle comunicazioni 9

2.2.3. Sincronizzazione dei messaggi 10

3. Descrizione del progetto 12

3.1. Requisiti 12

3.1.1. Scenario generale di utilizzo 12

3.1.2. Modalità di fruizione del servizio 13

3.1.3. funzionalità offerte 13

3.1.4. Scelta del nome e del logo 14

3.1.5. Glossario dei termini 15

3.1.6. Attori e casi d'uso 16

3.2. Casi d'uso principali 17

3.2.1. Creazione di un nuovo account / accesso all'account 17

3.2.2. Invio di un nuovo messaggio 21

3.2.3. Modifica / eliminazione di un messaggio 24

3.3. Architettura 26

3.3.1 Front-end 26

3.3.2 Back-end 27

3.3.3. Persistenza dei dati 28

3.3.4. Considerazioni architetturali del software 29

3.3.5. Considerazioni architetturali dei dati 30

3.3.6. Deployment 32

3.3.7. Monitoraggio 33

3.4. Implementazione 34

3.4.1. Modalità di organizzazione del lavoro 34

3.4.2. Ambiente di sviluppo 35

3.5. Scelta dell'interfaccia grafica 36

3.5.1. Galleria schermate e mockup 37

4. Considerazioni finali 43

4.1. Sfide e difficoltà affrontate 43

4.2. Conclusioni e sviluppi futuri 43

5. Bibliografia e sitografia 45

6. Ringraziamenti 47

1. Introduzione

Con l'evolvere delle tecnologie nel tempo e il susseguirsi delle innovazioni nel campo delle telecomunicazioni l'umanità ha dovuto affrontare numerosi cambiamenti nella propria vita quotidiana: quello che prima richiedeva giorni ora richiede pochi secondi, ciò che prima era fantascienza ora spesso è all'ordine del giorno; basti pensare al semplice invio di una lettera, fino a poche decadi fa erano necessari giorni, se non addirittura intere settimane per poter recapitare un messaggio a qualcuno, oggi invece lo si fa in pochi secondi.

Questo potere comunicativo ha aperto al mondo innumerevoli porte rendendo tutto più veloce, tuttavia ci ha portato innanzi a nuove problematiche e pericoli prima d'ora pressoché inesistenti o sconosciuti. Se da una parte abbiamo la possibilità di comunicare con le persone in modo istantaneo e con una comodità disarmante dall'altra parte dobbiamo fare i conti con un nuovo concetto di sicurezza personale, un concetto che oggi passa per un ambiente che a molti risulta del tutto oscuro e il cui funzionamento viene ritenuto dalle masse come criptico e difficilmente comprensibile, ossia la rete.

La rete ci mette a contatto con delle potenzialità mai viste prima dal genere umano, il tutto quasi in tempo reale, ma porta con sé nuovi rischi e problematiche per la nostra sicurezza personale, in particolar modo per i nostri dati personali.
Oggigiorno i dati personali hanno assunto un valore incalcolabile, spesso sottovalutato dagli internauti, eppure così alto da permettere a intere aziende di sopravvivere e addirittura a ergersi a colossi multinazionali. Si basti pensare ai social network o ai motori di ricerca che tutti i giorni vengono usati da centinaia di milioni di persone nel mondo, ebbene dietro a questi servizi offerti all'apparenza gratuitamente si nasconde un meccanismo di revenue basato appunto sui dati personali che forniamo noi stessi durante l'uso di tali piattaforme inconsciamente nonché in forma completamente gratuita.

In una società in cui il dato personale ha assunto un valore così alto questo non solo gioca un ruolo rilevante nel contesto del business di molte realtà imprenditoriali ma bensì è diventato il centro e il bersaglio di molte attività criminali che proprio come il resto del mondo si sono evolute di conseguenza. Alla luce di ciò diventa chiaro come la protezione e la salvaguardia dei propri dati personali, e in generale della propria vita digitale, sia una tematica di centrale importanza al giorno d'oggi.

Il progetto DotLine nasce proprio in questo contesto e ha come obiettivo quello di fornire gli strumenti necessari al fine di proteggere i propri dati personali in uno degli aspetti forse più comune del mondo digitale: le comunicazioni.
Spesso infatti ci si chiede che fine facciano i messaggi che noi tutti inviamo quotidianamente attraverso le varie piattaforme di messaggistica istantanea; la verità è che spesso tali messaggi vengono conservati sui server di queste piattaforme rimanendo alla mercè delle aziende stesse, per esempio per varie finalità di marketing o comunque con lo scopo ultimo di essere monetizzati. Tuttavia capita talvolta che i dati conservati da queste piattaforme vengano trafugati a seguito di intrusioni non autorizzate attuate con lo scopo di ottenere informazioni da rivendere o da utilizzare come merce di scambio per eventuali estorsioni di denaro ai danni delle aziende o peggio degli stessi utenti.

Risulta quindi fondamentale ormai sapersi orientare su piattaforme che applichino politiche di conservazione dei dati personali volte a salvaguardare gli stessi da questo tipo di accessi. Uno dei punti centrali del progetto è proprio quello di proteggere le informazioni degli utenti e garantirne la segretezza; per fare ciò DotLine mette a disposizione i più moderni standard di crittografia al fine di garantire che i messaggi scambiati dai fruitori del servizio risultino leggibili solamente dai partecipanti della conversazione e quindi protetti da eventuali sguardi indiscreti.

2. Contesto del progetto

Al fine di comprendere più a fondo la natura del software, le varie soluzioni adottate in fase di implementazione e le funzionalità proposte all'utente finale dell'applicativo è necessario fare prima una panoramica sul contesto in cui esso si posiziona.

2.1. Obiettivo generale

Nell'introduzione si è discusso di quanto i dati personali siano diventati importanti nell'era dell'informazione e, a conseguenza di ciò, quanto la protezione di tali dati sia di fondamentale importanza oggigiorno. L'obiettivo che ci si è posti nella realizzazione di DotLine è quello di fornire all'utente finale uno strumento semplice ma al tempo stesso sufficientemente efficace per salvaguardare i propri dati, più nello specifico le proprie conversazioni e i propri messaggi. Uno dei punti salienti del progetto stesso è appunto quello di essere in grado di raggiungere l'utilizzatore mettendo nelle sue mani uno strumento facilmente accessibile e utilizzabile il quale non sconvolga le abitudini dell'utente e dunque risulti naturale il suo conseguente uso. Da notare, sempre su questo tema, la scelta di rilasciare il software sottoforma di applicazione web, approccio che non costringe l'utente a dotarsi di programmi aggiuntivi per usufruire del servizio ma piuttosto permette all'utente di usare l'applicazione con strumenti al quale è già abituato e di cui è già in possesso.

Un altro punto tenuto in considerazione nella progettazione del software è quello di essere il meno invasivi possibile nella privacy del fruitore, questo si traduce nella riduzione al minimo delle informazioni personali che sono richieste sia al momento della creazione dell'account sia nell'utilizzo vero e proprio dell'applicativo. A tal fine si è deciso che in fase di creazione del profilo personale non debba essere necessario fornire un indirizzo e-mail o nessun altro tipo di dato che permetta l'identificazione dell'utente al di fuori del dominio dell'applicazione, questo per salvaguardare la privacy degli utilizzatori del servizio.

Da considerare che gli strumenti forniti da DotLine per la salvaguardia della riservatezza delle conversazioni nonché l'approccio poco invasivo nei confronti delle informazioni di contatto dell'utente sono strumenti meramente tecnici, infatti sta poi all'utente finale utilizzarli senza compromettere volontariamente la sicurezza dei suoi stessi dati personali prevenendo quindi spiacevoli fughe involontarie di informazioni private che possono essere evitate attraverso l'adozione di una condotta consapevole e responsabile quando si svolgono le proprie attività online.

2.2. Obiettivi derivati

Per raggiungere il grado di funzionalità tale da poter offrire gli obiettivi generali presentati in precedenza è stato quindi necessario assicurare una serie di obiettivi derivati, o possiamo dire secondari, che è stato dunque necessario prima individuare e quindi soddisfare affinché fosse possibile offrire un software coerente con quella che è l'offerta presentata e le conseguenti aspettative degli utenti finali del servizio.

Segue dunque un elenco di obiettivi secondari individuati e le relative soluzioni proposte per la conseguente implementazione:

2.2.2. Gestione delle credenziali e controllo degli accessi

Le credenziali d'accesso sono la porta per iniziare a utilizzare il servizio, pertanto, essendo la più importante protezione per il proprio account, devono essere trattate con un buon livello di sicurezza e protezione. Nonostante lo username dell'utente venga archiviato in chiaro sui server di DotLine questo non è altrettanto corretto per quanto concerne la password: infatti questa viene inviata al server sottoforma di hash e viene in seguito trattata per poter essere archiviata sotto forma di "cocktail", ovvero una quadrupla in cui sono contenuti due sali[1] (chiamati sale e pepe) di 32 caratteri, un numero intero casuale che determina quante volte il processo di hashing deve essere ripetuto e infine la stringa hash generata partendo dalla combinazione sale + password + pepe a cui è stato applicato l'algoritmo di hashing tante volte quanto definito dal numero intero descritto in precedenza.
Come verrà approfondito a breve, ogni utente ha a lui associata una coppia di chiavi RSA[2]: una pubblica e una privata. Una delle peculiarità di DotLine è quella di delegare tutti i vari processi crittografici necessari ai client in modo da fornire al server solo dati cifrati e dunque protetti da occhi indiscreti. Una domanda che sorge spontanea a questo punto è la seguente: ma se tutte le manipolazioni crittografiche avvengono lato client proprio per impedire che il server venga a conoscenza dei dati in chiaro, dove e come vengono archiviate le due chiavi RSA degli utenti? La risposta è piuttosto semplice, le due chiavi RSA vengono generate lato client al momento della creazione dell'account e vengono archiviate lato server così da poter essere fornite ai client al termine del processo di autenticazione, tuttavia la chiave RSA privata viene cifrata usando l'algoritmo AES[3] a 256 bit, la chiave pubblica invece viene mantenuta in chiaro in quanto per sua natura non rappresenta un pericolo per la sicurezza. La chiave AES utilizzata per cifrare la chiave RSA privata viene infatti derivata direttamente dalla password dell'utente utilizzando una funzione crittografica di derivazione chiamata PBKDF2 (Password-Based Key Derivation Function 2).

Un altro punto sul quale è necessario fare una riflessione è quello relativo al controllo delle sessioni aperte. È infratti piuttosto utile avere la possibilità di monitorare ed eventualmente intervenire interrompendo l'accesso a determinati client sul quale ci si era autenticati in precedenza. A tal fine DotLine permette di ottenere un elenco di tutte le sessioni attualmente aperte con le relative informazioni di tracciamento, come ad esempio il nome del browser utilizzato, la posizione geografica e l'indirizzo IP del client, nonché la data di primo ed ultimo accesso. Per ognuna di queste sessioni aperte è dunque possibile intervenire interrompendo

[1] Con sale si intende una stringa alfanumerica pseudo-casuale crittograficamente sicura, ovvero generata da una fonte casuale non facilmente previsibile; queste stringhe vengono concatenate alla password per rendere più complessi da attuare gli attacchi di tipo brute-force atti a ricavare la stringa originale dato un certo hash.
[2] La crittografia RSA (Rivest–Shamir–Adleman) è un tipo di crittografia, o meglio un algoritmo crittografico, asimmetrico che richiede una coppia di chiavi per funzionare: una pubblica, utilizzata per cifrare i dati, e una privata, usata invece per decifrare i dati. In DotLine entrambe queste chiavi hanno una lunghezza di 4096 bit.
[3] L'algoritmo AES (Advanced Encryption Standard) è una variante dell'algoritmo crittografico Rijndael, il quale è un algoritmo per la cifratura simmetrica a blocchi noto per la sua estrema sicurezza ed efficienza; DotLine usa questo algoritmo con chiavi a 256 bit in modalità GCM (Galois/Counter Mode).

l'accesso così che venga immediatamente negata la possibilità di effettuare qualsiasi ulteriore operazione senza effettuare nuovamente l'accesso all'account.
È anche possibile interrompere l'accesso a tutti i client al di fuori di quello attualmente in uso se lo si desidera; da notare come questo comportamento si verifichi in maniera totalmente automatica qualora la password dell'account venga cambiata o ripristinata, questo al fine di fornire un ulteriore modalità di protezione del proprio account qualora si sospetti un accesso indesiderato o un furto di credenziali.

2.2.1 Garanzia di riservatezza delle comunicazioni

Il fulcro di tutto il progetto risiede proprio in questo punto: garantire che i messaggi scambiati siano protetti da occhi indiscreti. A tal proposito, DotLine implementa i più moderni standard di sicurezza in materia crittografica; lo scenario generale è il seguente:

- Quando si desidera mettersi in contatto con qualcuno si cerca tale utente utilizzando l'apposita barra di ricerca, in questo modo è possibile aprire una conversazione con un utente e ottenere la sua chiave RSA pubblica.
- Quando viene aperta una nuova conversazione si genera una chiave crittografica a 256 bit per l'algoritmo AES propria della conversazione. Tale chiave viene resa disponibile a tutti i membri della conversazione e per ognuno di questi viene cifrata utilizzando la rispettiva chiave RSA pubblica a 4096 bit precedentemente ottenuta.
- Ogni qual volta che un messaggio viene inviato all'interno di una conversazione questo sarà cifrato usando l'algoritmo AES a 256 bit; per garantire inoltre l'integrità del dato ricevuto viene anche computata la sua firma mediante l'utilizzo di HMAC[4], il quale si appoggia a sua volta sull'algoritmo di hashing SHA[5] a 512 bit.
- Quando un messaggio viene ricevuto questo viene in automatico decifrato localmente utilizzando la chiave AES a 256 bit della conversazione precedentemente caricata e viene inoltre verificata la sua integrità e correttezza attraverso l'uso di HMAC e la firma ricevuta con il messaggio stesso.

Da notare come tutti i vari processi crittografici che ruotano attorno alla manipolazione dei messaggi avvengono localmente; questo in modo tale da far pervenire al server solo dati già cifrati e pertanto oscurati proprio al fine di prevenire fughe di dati in chiaro sia lungo il tragitto sia dall'interno del server stesso.
A tal punto è interessante aprire una parentesi concernente il lato implementativo in quanto è utile specificare come tutte le varie operazioni crittografiche svolte localmente all'interno del browser avvengono usando gli strumenti offerti nativamente dai browser stessi i quali implementano le varie funzioni di cifratura e di hashing senza l'appoggio di unità software di terze parti; questo in quanto esse sono implementate a basso livello e tendenzialmente usufruiscono di meccanismi di accelerazione hardware atti a rendere la loro esecuzione particolarmente efficiente su hardware moderno, in particolar modo sui dispositivi mobili.

[4] HMAC (Hash-based Message Authentication Code) è una modalità di autenticazione di messaggi che si basa su un algoritmo di hashing. Viene utilizzato da DotLine per verificare l'integrità dei messaggi scambiati e in questo contesto utilizza l'algoritmo di hashing SHA (Secure Hash Algorithm) a 512 bit.
[5] Un algoritmo di hashing è una funzione che prende in input un dato arbitrario, in questo caso specifico un messaggio, e restituisce una stringa unica, al netto di collisioni, da cui non è possibile risalire al dato originale.

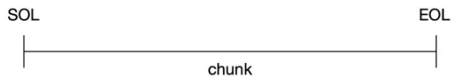
2.2.3. Sincronizzazione dei messaggi

Un altro dei punti chiave dell'applicativo è quello di mantenere i messaggi scaricati e archiviati localmente aggiornati: infatti non è sufficiente limitarsi a scaricare la lista dei messaggi delle varie conversazioni già presenti in locale, questo in quanto i messaggi precedenti, che così facendo non verrebbero considerati, potrebbero essere stati modificati o eliminati. Pertanto si procedere a scaricare una lista di commit, ovvero azioni fatte su un singolo messaggio, cronologicamente ordinati a partire dall'ultima data di sincronizzazione, o da zero se nessuna sincronizzazione è mai stata fatta; ognuno di questi commit rappresenta un nuovo messaggio o un cambiamento, come eliminazione o modifica, di un messaggio esistente e inviato in precedenza. Fatta questa introduzione si può passare ora a vedere meglio nel dettaglio come l'algoritmo di sincronizzazione dei messaggi operi; prima di cominciare è doveroso introdurre due nozioni, quella di SOL (Start Of Line) e quella di EOL (End Of Line) le quali rappresentano due tipologie di checkpoint all'interno della linea temporale dei messaggi. Data dunque una conversazione, possiamo riassumere l'algoritmo con il seguente pseudo-codice:

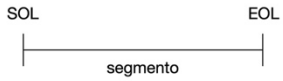
```
SMCS_2(C, U)
    # PRE: C è una data conversazione da sincronizzaree e U l'utente attualmente autenticato.
    # POST: tutta la lista di commit è stata scaricata e vi sono un SOL a simboleggiare
    # l'inizio della lista e un EOL a guardia della fine della stessa.

    D <- nil, L <- nil, F_SOL <- nil
    while L = nil or L not empty do
        # Carico un segmento di 250 commit dal server in ordine decrescente con data < D
        L <- fetchCommitList(C, U, D)
        if !isEmpty(L) then
            # Creo un checkpoint di tipo SOL usando il primo elemento di L.
            L_SOL <- createCheckpoint(SOL, L[0])
            # Creo un checkpoint di tipo EOL usando l'ultimo elemento di L.
            L_EOL <- createCheckpoint(EOL, L[-1])
            # Estraggo il primo checkpoint precedente alla data dell'EOL creato da L.
            checkpoint <- getFirstByDate(getDateOf(L_EOL))
            if checkpoint = nil or getTypeOf(checkpoint) != EOL then
                # Il checkpoint precedente nell'indice è SOL o non esiste, proseguo.
                addCheckpoint(EOL, L_EOL)
                D <- getDateOf(L_EOL)
            else
                # Il checkpoint è un EOL quindi è stato intersecato un segmento, salta alla fine di esso.
                D <- getDateOf(checkpoint)
            if F_SOL = nil then
                # Questa è la prima iterazione, aggiungo il checkpoint SOL a simboleggare l'inizio del segmento.
                addCheckpoint(SOL, L_SOL)
                F_SOL <- L_SOL
            # Estraggo tutti i checkpoint successivi alla data dell'EOL creato da L.
            checkpointList <- getCheckpointListByRange(getDateOf(L_EOL))
            foreach CH in checkpointList do
                isCurrentEol <- getTypeOf(CH) = EOL and equals(CH, L_EOL)
                isFirstSOL <- F_SOL != nil and equals(CH, F_SOL)
                if !isCurrentEol and !isFirstSOL then
                    # Il checkpoint CH è diverso dal primo SOL creato o dall'EOL aggiunto, lo rimuovo.
                    removeCheckpoint(CH)
    return
```

Per semplificare la lettura e la relativa comprensione dell'algoritmo appena illustrato si può considerare la seguente descrizione in prosa:
L'algoritmo inizia con il caricare la lista degli ultimi n commit, questa lista caricata prende il nome di chunk e rappresenta un sottoinsieme della lista completa di tutti i commit per una determinata conversazione e per l'utente attualmente autenticato, il quale è membro di tale conversazione. Dal chunk ricevuto si estraggono i checkpoint SOL ed EOL che determinano rispettivamente l'inizio e la fine del chunk, da notare che questi checkpoint non sono ancora

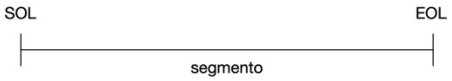
stati scritti nell'indice ma verranno archiviati in seguito se necessario. Prima di procedere con la scrittura nell'indice dei checkpoint creati è opportuno verificare se esiste un checkpoint di tipo EOL avente data minore a quella del checkpoint EOL estratto dal chunk appena caricato; qualora questo esista si procede con l'estrazione della data, la quale sarà la data da cui partire all'iterazione successiva quando si andrà a richiedere un nuovo chunk di commit. Qualora non venga trovato un checkpoint di tipo EOL allora si procederà quindi con l'aggiunta nell'indice del checkpoint EOL generato dal chunk il quale indicherà la fine del segmento sul quale si sta lavorando. Se l'attuale iterazione è la prima, e pertanto il chunk caricato rappresenta il primo frammento della lista completa di commit, allora verrà aggiunto all'indice un checkpoint di tipo SOL a indicare l'inizio del segmento attualmente in lavorazione.
L'ultimo passo è quello di ripulire l'indice da tutti i checkpoint aventi data precedente a quella del checkpoint EOL derivato dal chunk e che non sono stati creati durante l'attuale esecuzione dell'algoritmo. Una volta terminato anche questo passo l'algoritmo itera, le varie iterazioni proseguono fino a quando non si riceve un chunk vuoto.
Il risultato finale dell'algoritmo è la lista dei commit completamente caricata e un segmento, i cui riferimenti si trovano nell'indice dei checkpoint, avente agli estremi un checkpoint SOL e uno EOL, i quali indicano rispettivamente l'inizio e la fine dell'intera lista di commit caricata.
In questo tipo di gestione della lista dei commit si possono trovare due situazioni di interesse, la prima è quella in cui il chunk caricato include un segmento precedentemente caricato: in questo caso l'algoritmo rimuoverà i relativi checkpoint a simboleggiare un inglobamento del segmento in quanto sottoinsieme del chunk appena caricato. L'altra situazione d'interesse è invece rappresentata dall'intersezione, ovvero il caso in cui si trovi un checkpoint SOL avente data maggiore di quella dell'EOL del chunk ma relativo EOL con data minore di quest'ultimo. In questo caso l'algoritmo, dopo aver rimosso il checkpoint SOL del segmento, provvederà a cercare il checkpoint più vicino, se ne trova uno e questo è di tipo EOL allora significa che esso ha intersecato un segmento precedentemente caricato e quindi effettua un salto in modo da evitare di caricare nuovamente gli stessi dati. A seguito una rappresentazione grafica dei due casi di interesse appena descritti:



Il chunk ricevuto dal server.



a) **inclusione**: un segmento è incluso nel chunk caricato.



b) **intersezione**: un segmento inizia all'interno del chunk ma termina al di fuori di quest'ultimo.

Piccola nota: un commit è unico per un messaggio ed esso rappresenta l'ultima modifica fatta su quest'ultimo, ossia una creazione, in principio, l'ultima eventuale modifica, se ve ne sono state fatte, oppure l'eliminazione, qualora il messaggio sia stato eliminato.
Un ulteriore piccola nota di interesse: l'algoritmo è stato denominato SMCS-2, questo non è altro che l'acronimo di Segmented Message Commit Synchronize, il numero due indica che si tratta della seconda versione elaborata.

3. Descrizione del progetto

A questo punto è opportuno fare una panoramica sui vari aspetti del progetto e sulle scelte prese in fase di progettazione che hanno successivamente portato allo sviluppo di tali aspetti nonché sulle motivazioni dietro a queste scelte.

3.1. Requisiti

3.1.1. Scenario generale di utilizzo

Uno dei punti chiave dell'intero progetto è la semplicità e la naturalezza d'uso: iniziare a usare DotLine infatti risulta essere piuttosto semplice, tutto ciò che occorre è un account personale che può essere creato direttamente nella schermata principale dell'applicazione.
La creazione di un profilo personale richiede l'inserimento di due sole informazioni: un nome utente, il quale deve essere univoco e composto da caratteri alfanumerici latini nonché non più lungo di sedici caratteri, e una password con una lunghezza minima di otto caratteri. Come anticipato in precedenza non vengono richiesti altri dati personali, come per esempio indirizzo e-mail o numero di telefono, questo proprio per essere meno invasivi possibile sulla privacy dell'utente e impedirne il tracciamento. Qualora l'utente sia già in possesso di un account sarà dunque necessario effettuare l'accesso con l'username e password.

Una volta eseguito l'accesso l'utente troverà la lista delle conversazioni aperte, per avviare una nuova conversazione invece sarà sufficiente cercare la persona con la quale si desidera interloquire utilizzando la barra di ricerca in alto, sempre accessibile, e cercare l'username del destinatario. Attraverso la ricerca è possibile trovare e contattare qualsiasi utente iscritto al servizio e dal momento in cui si invia il primo messaggio verrà creata una nuova conversazione con il destinatario selezionato; da quel momento tutti i messaggi scambiati all'interno della conversazione saranno accessibili ai soli membri, senza la necessità di azioni aggiuntive, in modo completamente automatico.

DotLine permette di scambiarsi diversi tipi di messaggi: infatti oltre ai classici messaggi di testo è possibile inviare allegati di qualsiasi tipo, i quali beneficiano della crittografia end-to-end alla pari dei messaggi testuali. Gli allegati inviati, grazie a questo livello crittografico applicato, risultano quindi accessibili solo al mittente e al destinatario, pertanto, nonostante essi siano archiviati sul cloud esattamente come avviene per i messaggi tradizionali, questi risulteranno inutilizzabili se estratti direttamente dal cloud stesso. In aggiunta a questo è possibile inviare anche altri tipi di contenuti quali posizioni geografiche, sticker, statici o animati, e persino messaggi vocali; tutto protetto dalla crittografia end-to-end.

A completamento delle funzionalità di messaggistica offerte vi è il completo controllo delle conversazioni e dei messaggi in esse contenuti: è pertanto supportata sia la modifica che l'eliminazione dei messaggi inviati, in quest'ultimo caso per se stessi oppure per tutti i membri della conversazione; le modifiche apportate ai messaggi saranno sincronizzate in tempo reale fra tutti i membri della conversazione, compreso l'autore stesso delle modifiche, nel caso in

cui questo abbia precedentemente effettuato l'accesso su più dispositivi in contemporanea. L'eliminazione è inoltre supportata anche per un'intera conversazione, con le stesse modalità supportate per i messaggi singoli, ovvero sia per se stessi oppure per tutti i membri della stessa.

3.1.2. Modalità di fruizione del servizio

DotLine è stato pensato per raggiungere un vasto numero di utenti senza la necessità di imporre particolari requisiti in modo tale che utilizzare il servizio possa rivelarsi un'esperienza rapida che non richiede necessariamente l'installazione o la configurazione di alcun ulteriore componente aggiuntivo di supporto.
A tal fine l'applicativo è disponibile sottoforma di applicazione web, la quale richiede soltanto l'uso di un semplicissimo browser, purché sufficientemente moderno e aggiornato. A garanzia che i requisiti di utilizzo siano rispettati, il client web di DotLine esegue in autonomia una verifica sia sul client, ovvero il browser, verificando che tutte le API (Application programming interface) HTML5 necessarie per il suo corretto funzionamento siano quindi supportate, sia sul server al quale andrà successivamente a connettersi, così da garantire che il client supporti la versione specificata dal server in uso. Dunque è pertanto possibile accedere e utilizzare DotLine attraverso un software che ormai chiunque ogni giorno usa e con cui ha acquisito una dimestichezza almeno basilare.

Un discorso analogo vale anche per il dispositivo fisico utilizzabile: DotLine infatti è realizzato utilizzando un tipo di interfaccia grafica totalmente in grado di adattarsi alle dimensioni del dispositivo utilizzato, che questo sia un normale computer oppure sia un tablet o piuttosto uno smartphone. L'interfaccia responsiva è un aspetto su cui ci si è particolarmente focalizzati in modo tale da offrire all'utente un'esperienza d'utilizzo il più omogenea possibile al fine di ridurre eventuali barriere fra l'uso dell'applicativo tramite dispositivi desktop tradizionali e l'utilizzo attraverso dispositivi mobili.

3.1.3. funzionalità offerte

L'idea che ruota attorno alla base del progetto è quella di mettere nelle mani degli utilizzatori uno strumento sì semplice ma al contempo sufficientemente completo in modo che possa supportare le esigenze comunicative degli utenti che ne fanno uso. Per questo motivo sono state pensate e dunque implementate una serie di funzionalità, in particolar modo riferite ai messaggi stessi, in grado di rendere disponibili una moltitudine di supporti comunicativi; a tal scopo DotLine supporta le seguenti tipologie di messaggi:

- Messaggi testuali semplici, ai quali è anche possibile opzionalmente allegare uno o più file; i membri della conversazione potranno vedere un'anteprima di immagini, video e file audio, tuttavia per qualsiasi tipo di file vi è la possibilità di download.
- Invio di posizioni geografiche: con un semplice click è possibile condividere la propria posizione, i membri della conversazione saranno dunque in grado di visualizzarla sotto forma di mappa interattiva.

- Stickers: DotLine supporta diversi sticker raggruppati per sticker pack tra cui scegliere; gli sticker possono essere sia statici che animati.
- Messaggi vocali: è possibile comunicare in maniera più diretta registrando e inviando un messaggio vocale al posto di un classico messaggio di testo.

Da notare come il supporto alla crittografia end-to-end offerto copre sia i messaggi testuali che qualsiasi altro tipo di messaggio, file allegati compresi, tuttavia, a differenza dei messaggi testuali che vengono sincronizzati e archiviati in locale, i file allegati, messaggi vocali compresi, vengono scaricati e decifrati solo al momento del bisogno e vengono archiviati nel contesto della sessione del browser, ossia essi vengono rimossi una volta che la pagina viene chiusa o ricaricata. Se i file non supportano la visualizzazione in anteprima, disponibile per immagini, video e audio, essi verranno scaricati e decifrati qualora vengano scaricati dall'utente stesso.
In aggiunta alle funzionalità elencate relative ai messaggi possiamo aggiungere le funzionalità relative alla gestione dei messaggi, come ad esempio la modifica, attualmente supportata per i messaggi testuali soltanto, l'eliminazione, per tutti o solo per sé stessi.
Per quanto riguarda le conversazioni troviamo il supporto all'eliminazione e il supporto alla ricerca fra i messaggi, i quali sono indicizzati in locale quando il processo di sincronizzazione è concluso.

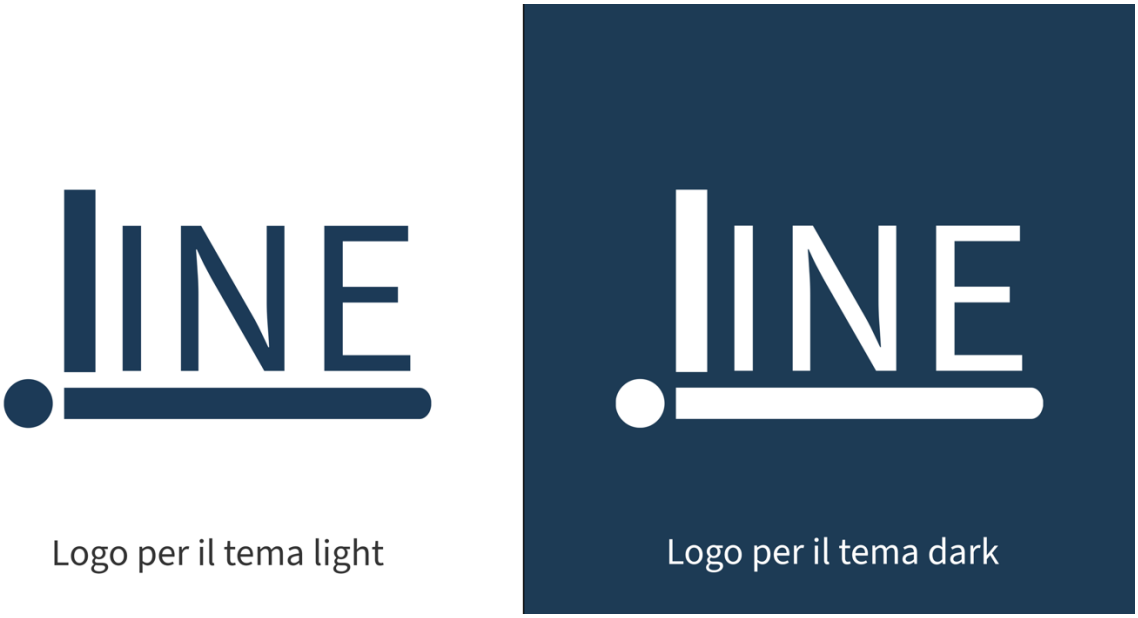
3.1.4. Scelta del nome e del logo

Il nome "DotLine" non è altro che la traduzione inglese del nome "punto linea", il quale è un riferimento a uno dei più primitivi meccanismi utilizzati nelle telecomunicazioni, ovvero il codice Morse e il telegrafo. La scelta di questa analogia nasce da due fonti, la prima, un'ironia legata proprio alla telegrafia al fine di sottolineare l'evoluzione delle telecomunicazioni, da un approccio complesso, poco affidabile, di natura volatile nonché facilmente intercettabile a un approccio semplice, affidabile, non volatile e sicuro. La seconda fonte di ispirazione del nome è la stessa che ha funto in primo luogo da fonte di ispirazione per l'intero progetto, ovvero la celebre applicazione di messaggistica istantanea Telegram.
Piccola curiosità relativa al nome: precedentemente il rilascio della prima versione pubblica il progetto è stato temporaneamente denominato "Project Pigeon", solo in seguito questo ha poi preso il nome definitivo "DotLine".

Per la realizzazione del logo si è preferito procedere con un approccio minimale che esprime in modo chiaro il nome dell'applicazione ma che al contempo lascia trasparire quella che è la metafora dietro alla scelta del nome. Anche per quanto riguarda la palette di colori ci si è orientati verso una scelta minimale, monocromatica, così da non affaticare l'occhio con giochi di colori; sempre su questa linea di pensiero si è selezionato un font di facile lettura, ovvero il Source Sans Pro, font open source rilasciato da Adobe e facente parte della famiglia Sans Serif, una famiglia che raggruppa font particolarmente indicati per il web design.
Il logo è realizzato in grafica vettoriale SVG[6] (Scalable Vector Graphics) in modo tale da poter essere facilmente scalato a varie dimensioni senza perdere di qualità, esso è disponibile in

[6] Il formato SVG permette di creare immagini utilizzando un linguaggio di markup testuale e comprensibile dalle persone; tali immagini vengono generalmente create usando software di grafica vettoriale come Adobe Illustrator o Inkscape e sono visualizzabili direttamente del browser senza dover essere convertite.

due varianti, una light con colore scuro e una dark dal colore più chiaro usate rispettivamente all'interno del tema light e di quello dark, tema che viene gestito automaticamente in base alle preferenze di sistema dell'utente oppure in alternativa può essere esplicitamente scelto da quest'ultimo qualora lo preferisse. A seguire i loghi realizzati per DotLine sia per la modalità light che per quella dark:



Piccola nota relativa al design di DotLine: l'intero applicativo usa una palette di colori flat e minimale, coerente con la scelta cromatica fatta per il logo; lo stesso si può dire anche per quanto riguarda il font: infatti viene utilizzato lo stesso font sia per il logo che per il software, relativamente al client però il font viene servito attraverso la piattaforma online Google Fonts. Per quanto concerne invece le icone utilizzate all'interno dell'applicativo esse sono fornite dalla piattaforma Font Awesome e anche in questo caso si è optato per uno stile minimale in modo tale da fornire indicazioni chiare e senza troppi fronzoli all'utilizzatore.

3.1.5. Glossario dei termini

Per meglio comprendere il progetto nella sua interezza è opportuno introdurre alcuni termini ricorrenti in questo contesto.

Termine	Descrizione	Sinonimi
Messaggio	Unità elementare di informazione all'interno di una conversazione: composto da un testo e/o una lista di allegati o altri contenuti multimediali.	
Conversazione	Una collezione cronologicamente ordinata di messaggi scambiati tra due o più utenti.	
Direct Message	Un messaggio inviato in una conversazione fra due soli utenti o, per estensione, una conversazione fra due soli utenti.	DM

Allegato	Un file caricato e inviato insieme ad un messaggio eventualmente privo di testo.	
Utente	Fruitore del servizio che, per mezzo di un account personale, usufruisce delle funzionalità offerte dall'applicativo.	
Accesso	Collegamento di un utente alla piattaforma previa autorizzazione con successo con conseguente rilascio di un token di autorizzazione.	Login, autenticazione
Sessione aperta	Accesso fatto in precedenza il cui token rilasciato è attualmente usato o usabile e per il quale si è a conoscenza delle informazioni di tracciamento del client (nome del browser, nome del sistema operativo, posizione geografica e indirizzo IP).	Sessione
Message commit	Una qualsiasi azione su un messaggio, esso comprende la creazione, la modifica e l'eliminazione di un particolare messaggio.	Commit
Sticker pack	Una collezione di sticker che possono essere selezionati e inviati.	
Checkpoint	Rappresenta un riferimento nell'indice dei checkpoint in merito all'avanzamento della sincronizzazione dei commit di una conversazione.	Checkpoint EOL Checkpoint SOL
Segmento	Nel contesto della sincronizzazione dei messaggi si tratta di una porzione dell'insieme cronologico dei commit importati e delimitato da un checkpoint SOL all'inizio e uno EOL alla fine.	
Chunk	Simile al concetto di segmento ma a differenza di quest'ultimo esso indica una porzione di lista dei commit ricevuta dal server e non ancora archiviata localmente.	
Segmento finale	Indica un segmento che copre l'intera lista dei commit disponibile per una conversazione.	

3.1.6. Attori e casi d'uso

Attore	Azioni
Utente	Creazione di un nuovo account / accesso all'account Ricerca di un utente nel sistema Invio di un nuovo messaggio Modifica / eliminazione di un messaggio inviato in precedenza Consultazione dei messaggi in una conversazione Modifica delle impostazioni dell'account Modifica della password associata all'account Modifica delle informazioni personali Consultazione delle sessioni aperte Chiusura di tutte le altre sessioni aperte

	Chiusura di una precisa sessione aperta Ripristino dell'account e della relativa password

3.2. Casi d'uso principali

3.2.1. Creazione di un nuovo account / accesso all'account

Nome caso d'uso: Creazione di un nuovo account / accesso all'account
Portata: Sistema
Livello: Obiettivo utente
Attore primario: Utente
Parti interessate: -
Pre-condizioni: -
Garanzie di successo o post-condizioni: L'utente è stato autenticato nel sistema ed eventualmente il suo account utente è stato creato.

Scenario principale di successo

#	Attore	Sistema
1	Accede al form di creazione dell'account	Mostra il form di registrazione
2	Inserisce lo username desiderato	-
3	Inserisce la password scelta	-
4	Re-inserisce la password scelta per conferma	-
5	Termina la creazione dell'account	Crea un nuovo account utente e autentica l'utente usando l'account creato e reindirizza l'utente alla schermata principale dell'applicazione.
	Termina il caso d'uso	

Estensioni ed eccezioni

#	Attore	Sistema
1a.1	Accede al form di autenticazione	Mostra il form di autenticazione
1a.2	Inserisce il proprio username e la propria password	-
1a.3	Opzionalmente attiva la persistenza della sessione di autenticazione	-
1a.4	Completa l'autenticazione	Autentica l'utente e lo reindirizza alla schermata principale dell'applicazione
	Termina il caso d'uso	

#	Attore	Sistema
2a.1	Inserisce lo username desiderato	Lo username inserito non rispetta il formato accettato (solo caratteri alfanumerici, punti, underscore e trattini per un massimo di 16 caratteri)
	Torna al passo 2	

#	Attore	Sistema
2b.1	Inserisce lo username desiderato	Lo username inserito è già stato assegnato ad un altro utente
	Torna al passo 2	

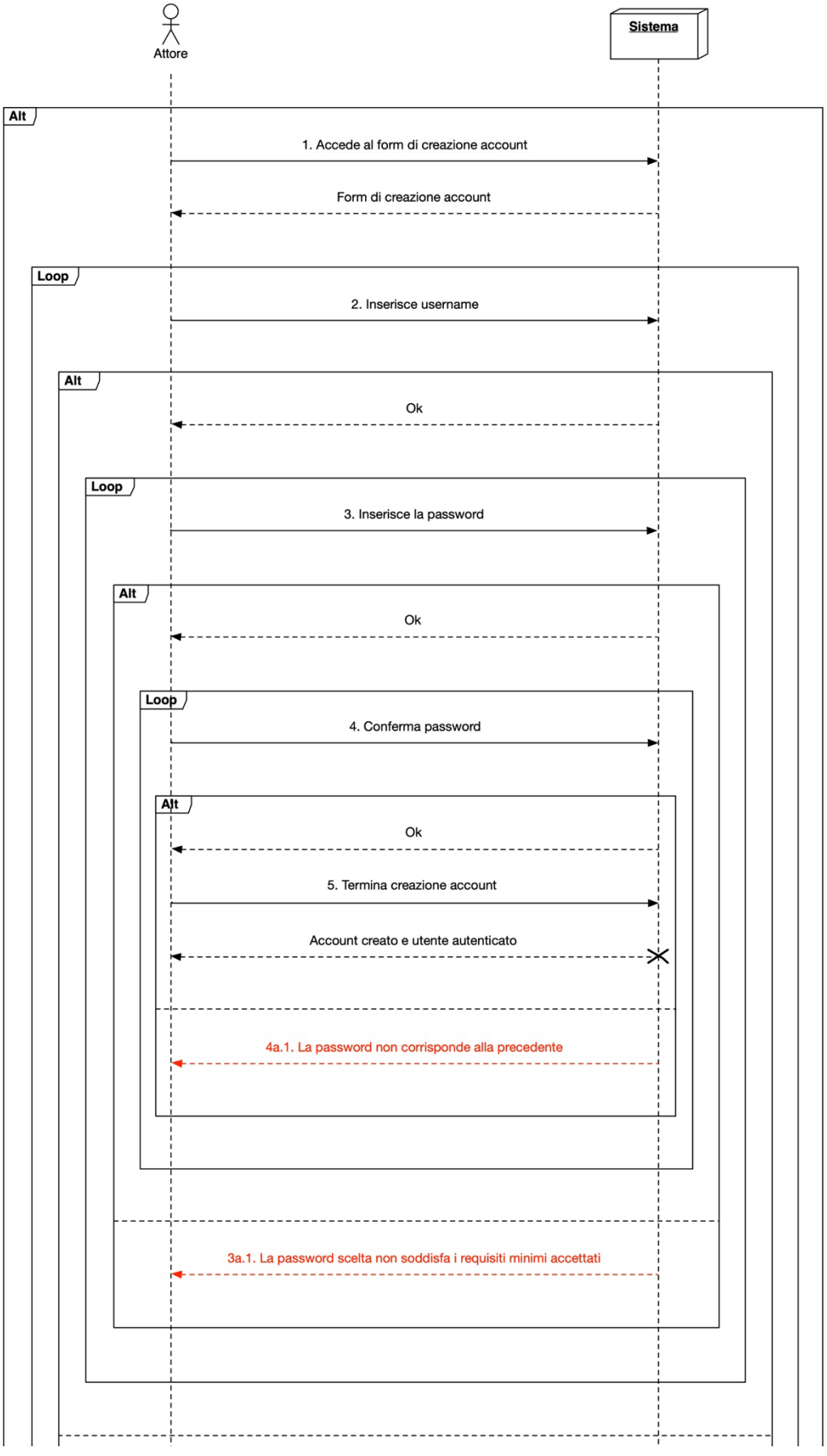
#	Attore	Sistema
3a.1	Inserisce la password scelta	La password scelta non soddisfa i requisiti minimi accettati (lunghezza minima di 8 caratteri)
	Torna al passo 3	

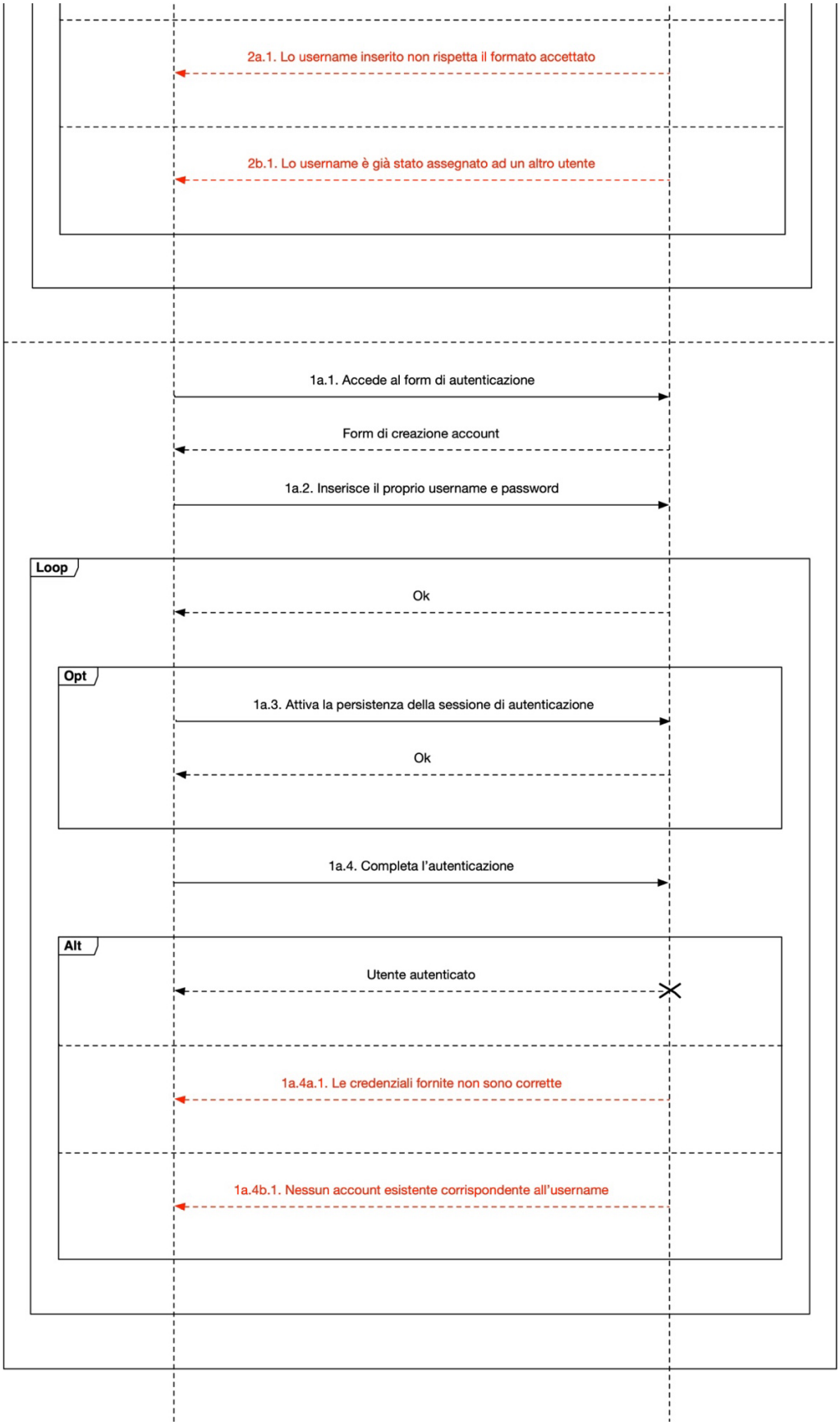
#	Attore	Sistema
4a.1	Re-inserisce la password scelta per conferma	La password inserita non corrisponde alla precedente
	Torna al passo 4	

#	Attore	Sistema
1a.4a.1	Si autentica usando le credenziali fornite	Le credenziali fornite non sono corrette
	Torna al passo 1a.2	

#	Attore	Sistema
1a.4b.1	Si autentica usando le credenziali fornite	Non esiste nessun account che corrisponda allo username dato
	Torna al passo 1a.2	

Diagrammi di sequenza di sistema




2a.1. Lo username inserito non rispetta il formato accettato
2b.1. Lo username è già stato assegnato ad un altro utente
1a.1. Accede al form di autenticazione
Form di creazione account
1a.2. Inserisce il proprio username e password
Loop
Ok
Opt
1a.3. Attiva la persistenza della sessione di autenticazione
Ok
1a.4. Completa l'autenticazione
Alt
Utente autenticato
1a.4a.1. Le credenziali fornite non sono corrette
1a.4b.1. Nessun account esistente corrispondente all'username

3.2.2. Invio di un nuovo messaggio

Nome caso d'uso: Invio di un nuovo messaggio
Portata: Sistema
Livello: Obiettivo utente
Attore primario: Utente
Parti interessate: -
Pre-condizioni: -
Garanzie di successo o post-condizioni: Il messaggio scritto dall'utente è stato inviato, se la conversazione di destinazione non esiste essa viene creata.

Scenario principale di successo

#	Attore	Sistema
1	Seleziona una conversazione	Mostra la conversazione selezionata
2	Scrive il messaggio	-
3	Opzionalmente aggiunge un allegato	-
	Ripete il passo 3 finché non è soddisfatto	
4	Invia il messaggio	Registra e invia il messaggio, se la conversazione è nuova essa viene prima creata.
	Termina il caso d'uso	

Estensioni ed eccezioni

#	Attore	Sistema
1a.1	Seleziona un utente fra i risultati di ricerca.	Mostra la conversazione relativa all'utente selezionato
	Continua con il passo 2	

#	Attore	Sistema
2a.1	Seleziona uno sticker pack	Mostra gli sticker contenuti nello sticker pack selezionato
2a.2	Seleziona uno sticker	-
	Continua con il passo 4	

#	Attore	Sistema
2b.1	Registra un messaggio vocale	-
	Continua con il passo 4	

#	Attore	Sistema
2c.1	Invia la propria posizione geografica	-
	Continua con il passo 4	

#	Attore	Sistema

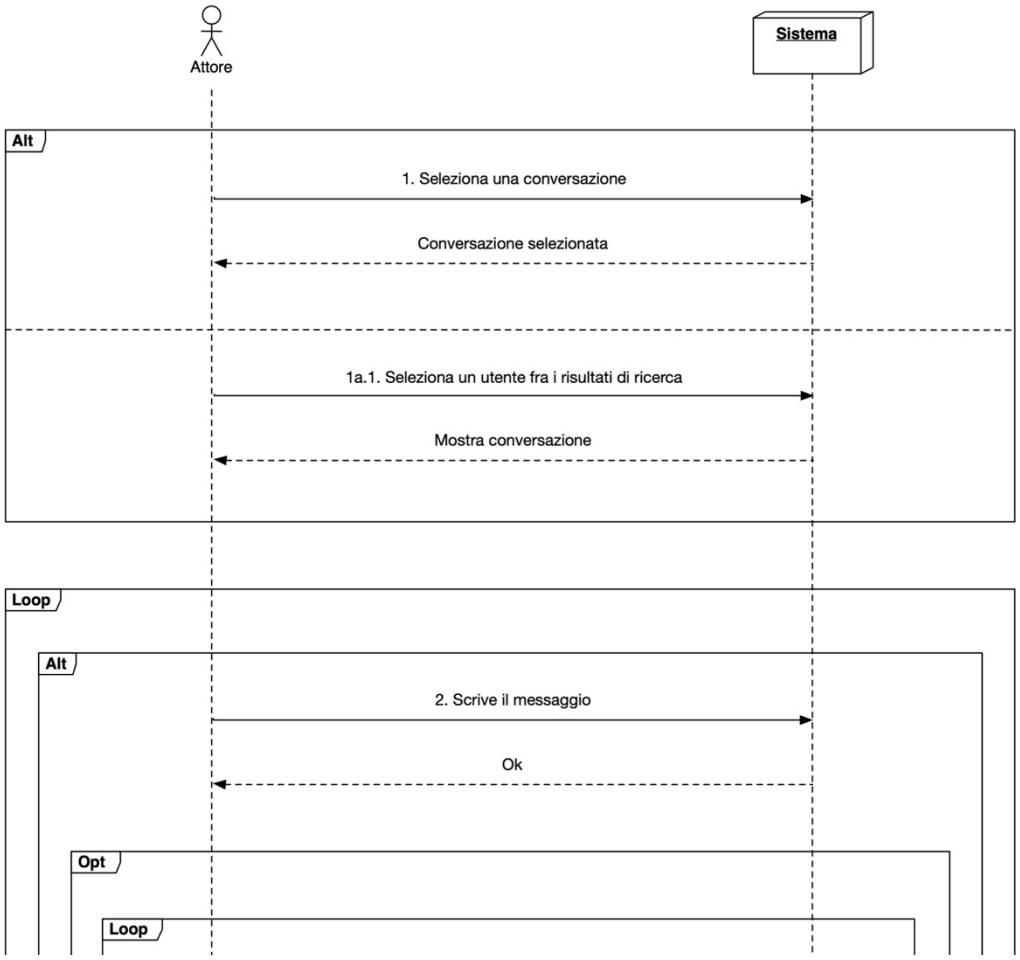
3a.1	Rimuove un allegato precedentemente aggiunto	-
	Ripete il passo 3a.1 finché non è soddisfatto e torna allo step 3	

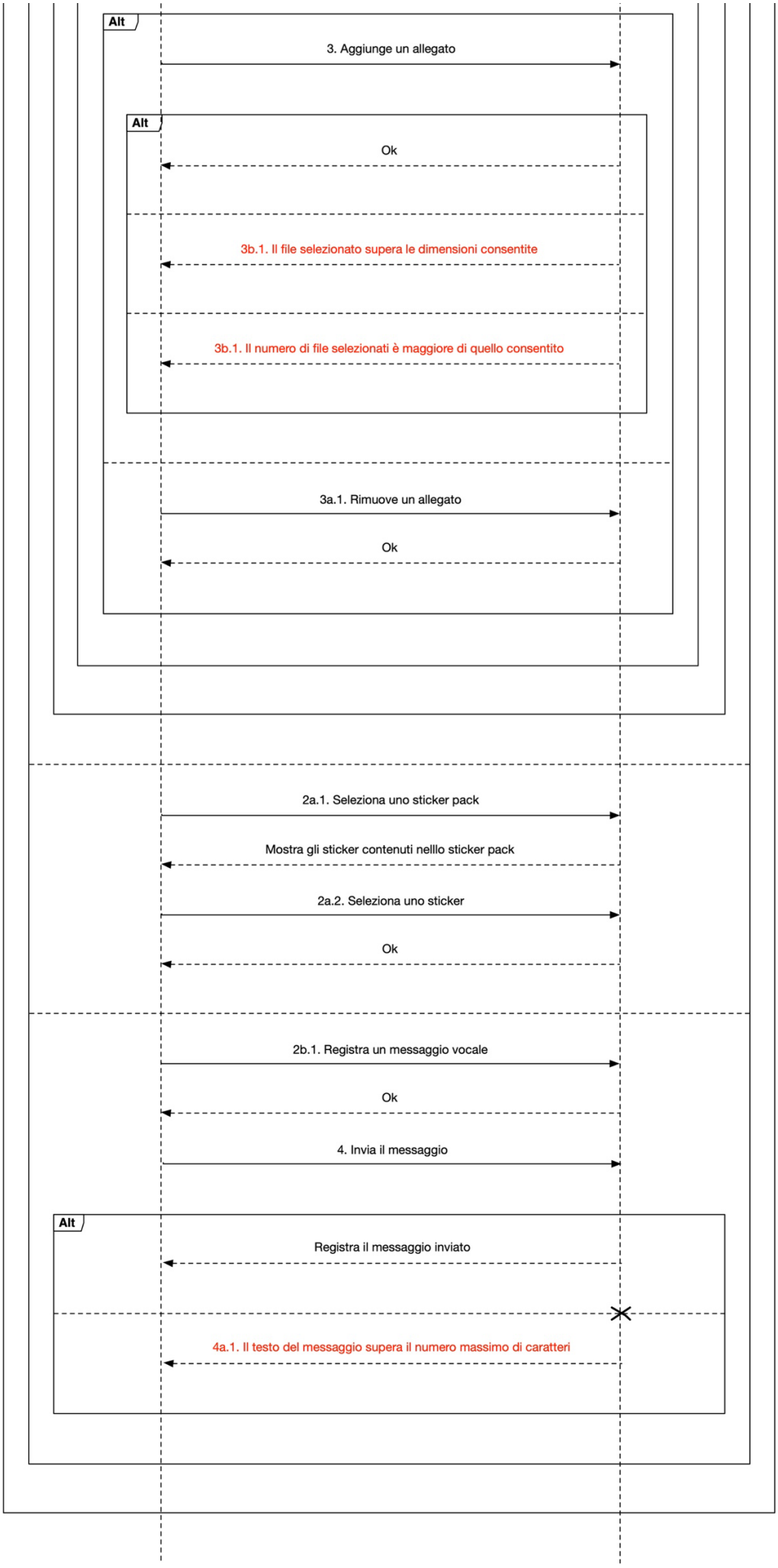
#	Attore	Sistema
3b.1	Opzionalmente aggiunge un allegato	Il file selezionato supera le dimensioni massime consentite
	Torna al passo 3	

#	Attore	Sistema
3c.1	Opzionalmente aggiunge un allegato	Il numero di file selezionati è maggiore a quello massimo di file consentiti
	Torna al passo 3	

#	Attore	Sistema
4a.1	Invia il messaggio	Il testo del messaggio supera il numero di caratteri massimo consentiti
	Torna al passo 2	

Diagrammi di sequenza di sistema




Alt
3. Aggiunge un allegato
Alt
Ok
3b.1. Il file selezionato supera le dimensioni consentite
3b.1. Il numero di file selezionati è maggiore di quello consentito
3a.1. Rimuove un allegato
Ok
2a.1. Seleziona uno sticker pack
Mostra gli sticker contenuti nelllo sticker pack
2a.2. Seleziona uno sticker
Ok
2b.1. Registra un messaggio vocale
Ok
4. Invia il messaggio
Alt
Registra il messaggio inviato
4a.1. Il testo del messaggio supera il numero massimo di caratteri

3.2.3. Modifica / eliminazione di un messaggio

Nome caso d'uso: Modifica / eliminazione di un messaggio
Portata: Sistema
Livello: Obiettivo utente
Attore primario: Utente
Parti interessate: -
Pre-condizioni: -
Garanzie di successo o post-condizioni: Il messaggio desiderato è stato modificato o eliminato

Scenario principale di successo

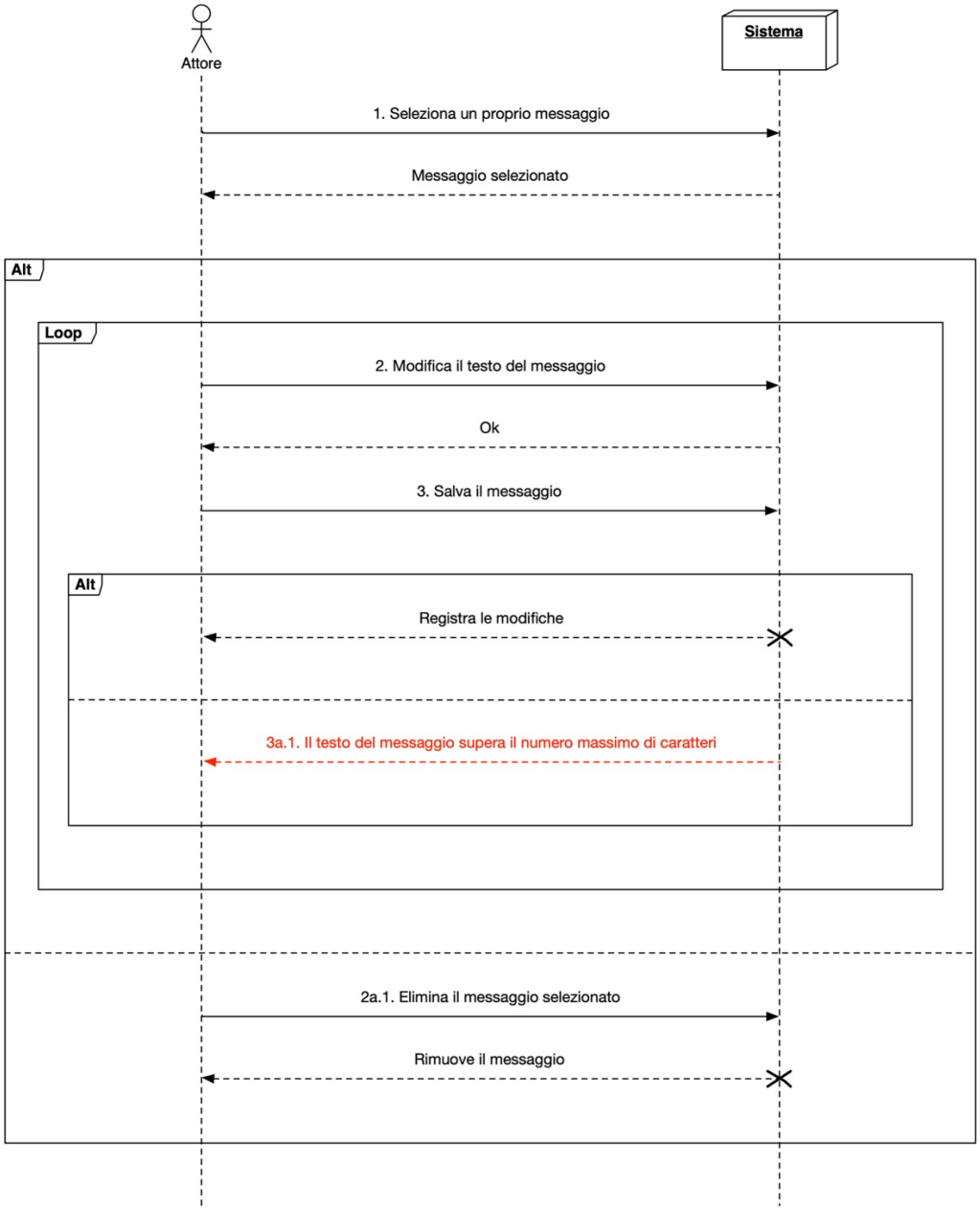
#	Attore	Sistema
1	Seleziona un proprio messaggio	Mostra il messaggio selezionato
2	Modifica il testo del messaggio	-
3	Salva il messaggio modificato	Registra le modifiche al messaggio
	Termina il caso d'uso	

Estensioni ed eccezioni

#	Attore	Sistema
2a.1	Elimina il messaggio selezionato	Rimuove il messaggio
	Termina il caso d'uso	

#	Attore	Sistema
3a.1	Salva il messaggio modificato	Il testo del messaggio supera il numero di caratteri massimo consentiti
	Torna al passo 2	

Diagrammi di sequenza di sistema



3.3. Architettura

Segue dunque una retrospettiva sui principali aspetti legati all'implementazione del software in senso prevalentemente tecnico e alle conseguenti scelte adottate in questo contesto.

3.3.1 Front-end

Il focus principale nella scelta delle tecnologie da utilizzare lato front-end è stato quello di offrire all'utente la migliore esperienza possibile mettendolo a contatto con un'interfaccia interattiva nonché di facile comprensione e utilizzo. Un altro aspetto fondamentale su cui ci si è concentrati è stato quello relativo alla scalabilità e manutenibilità del codice prodotto.
Per tutti questi motivi la scelta finale è ricaduta su uno stack tecnologico che include la libreria React, una libreria sviluppata e mantenuta da Facebook con il preciso scopo di semplificare la realizzazione di interfacce grafiche interattive e moderne.

In aggiunta a React completano lo stack di sviluppo il linguaggio JavaScript (versione ES2022) a cui si aggiunge un suo dialetto, JSX (JavaScript XML), una sintassi propria di React che fonde JavaScript e HTML insieme, e infine il linguaggio SCSS (Sassy Cascading Style Sheets), ovvero una variante del linguaggio CSS che permette una maggiore flessibilità nello sviluppo nonché un migliore grado di riutilizzabilità del codice scritto.
La persistenza dei dati lato browser è invece assicurata da una tecnologia offerta all'interno dello standard HTML5, ovvero IndexedDB[7]. Tuttavia, per semplificare l'accesso a questa API si è deciso di appoggiarsi al package jsStore il quale permette l'utilizzo di una sintassi più ad alto livello rispetto a quella offerta nativamente da IndexedDB.

Per rendere l'applicativo maggiormente accessibile agli utenti, indipendentemente dalla loro provenienza, è stato inoltre implementato il supporto alla localizzazione, ovvero è possibile usufruire del software in diverse lingue in base alle proprie preferenze.
L'internazionalizzazione di DotLine è stata implementata attraverso il pacchetto i18next e i file linguistici sono gestiti usando il formato JSON (JavaScript Object Notation); al momento la localizzazione è stata fatta interamente da agenti umani così da garantire una maggiore correttezza linguistica, tuttavia non si esclude in futuro l'utilizzo di traduttori elettronici per lingue meno diffuse in modo da poter estendere la copertura offerta.

Tutto questo pacchetto tecnologico è gestito attraverso alcune risorse software di supporto, nello specifico troviamo Babel, una nota libreria che permette di estendere la compatibilità del codice JavaScript convertendo codice scritto con uno standard ECMA[8] più recente in una versione dello stesso utilizzando una versione di ECMA più diffusa ma meno recente, in questo caso specifico ECMA2015. Tutte le risorse prodotte per la parte front-end vengono processate da un software denominato WebPack il cui compito specifico è quello di eseguire una serie di

[7] IndexedDB è una tecnologia offerta nativamente dai browser la quale permette l'archiviazione di dati strutturati in modo simile a quanto avviene in un database e offre una serie di strumenti per la loro gestione, scrittura e lettura in modo efficiente e ottimizzato in base alla piattaforma su cui ci si trova ad operare.
[8] ECMA, o meglio, ECMAScript è lo standard che descrive le specifiche del linguaggio JavaScript che vengono successivamente implementate dai vari compilatori JIT (Just-In-Time compiler) o interpreti del linguaggio.

ottimizzazioni e manipolazioni, per esempio la conversione del codice JSX in codice JavaScript, la conversione di codice SCSS in codice CSS, la minimizzazione delle risorse CSS e JavaScript e così via; questo permette di ottenere un risultato comprensibile da parte del browser nonché ottimizzato per quanto riguarda il download degli artefatti generati.

Tutte le dipendenze di terze parti richieste dal client di DotLine, ovvero la parte front-end del progetto, sono gestite attraverso l'uso del package manager NPM (Node Package Manager); la successiva compilazione del client richiede, oltre a quest'ultimo, la presenza del runtime JavaScript Node.js installato all'interno dell'ambiente di sviluppo, requisito che, come verrà illustrato meglio in seguito, risulta essere condiviso fra la parte front-end appena vista e quella back-end che verrà discussa a seguire.

3.3.2 Back-end

Per la realizzazione del server alla base del funzionamento di DotLine si è scelto un linguaggio di programmazione leggero, che gode di una larga diffusione nonché di un ottimo sostegno da parte della community di sviluppatori, un linguaggio ad alto livello e con inoltre un ottimo supporto all'asincronia e alla potenziale scalabilità del software: JavaScript. La parte back-end è infatti stata realizzata utilizzando il linguaggio JavaScript nella sua versione ES2022 la cui successiva esecuzione è affidata al runtime Node.js, versione 18 o superiore, già necessario come visto in precedenza per lo sviluppo della parte front-end.
La scelta di JavaScript come linguaggio di programmazione per la realizzazione del server di DotLine ha un ulteriore vantaggio intrinseco, ovvero permette un buon riutilizzo di parte del codice realizzato per il server da parte del client e vice versa nonché un grado pressoché totale di riciclo delle competenze richieste per la scrittura del client in quanto la sintassi e le basi del linguaggio rimangono tutto sommato invariate fra i due ambienti.

Il server è stato concepito seguendo l'architettura RESTful[9], ossia in modo tale da offrire una serie di endpoint accessibili attraverso il protocollo HTTP (Hypertext Transfer Protocol) con i quali è possibile interagire con le API di DotLine. Alcune funzionalità invece, per la loro natura, sono offerte attraverso l'uso di un altro protocollo, denominato WebSocket, anch'esso basato su HTTP; questo protocollo permette di interagire con il server in tempo reale e in modo bidirezionale e viene prevalentemente utilizzato per la propagazione dei vari eventi, per esempio la ricezione di un messaggio o la scrittura di un messaggio da parte di uno dei membri della conversazione attualmente aperta.
Gli endpoint sono implementati con il supporto del framework Express.js[10] e l'autenticazione avviene tramite l'ottenimento di un token Bearer in fase di registrazione o di autenticazione che viene poi successivamente iniettato all'interno delle richieste fatte dal client verso il server mediante l'uso di un header HTTP. Gli endpoint e i messaggi scambiati all'interno delle

[9] L'architettura RESTful, o semplicemente REST (Representational State Transfer), prevede che le funzionalità offerte da un'applicazione web siano suddivise in risorse web accessibili e controllabili attraverso specifici URL e verbi che ne definiscono le operazioni (per esempio GET per ottenere una risorsa, POST per crearla, PATCH per modificarla, PUT per sostituirla e infine il verbo DELETE per eliminarla).
[10] Il framework Express.js, noto anche come Express, è un framework open-source progettato per la realizzazione di applicazioni web basate sull'architettura RESTful; ad oggi è considerato il framework standard de-facto per lo sviluppo di questo tipo di applicazioni web su piattaforma Node.js.

connessioni WebSocket utilizzano il noto formato JSON per l'interscambio delle informazioni, formato particolarmente leggero nonché minimale e che gode inoltre di un ottimo supporto da parte del linguaggio JavaScript utilizzato su entrambi i fronti.

A livello architetturale il server è organizzato seguendo il celebre pattern MVC (Model–View–Controller) che prevede una netta separazione fra modelli, ovvero le unità software che si occupano di rappresentare e mappare le entità del livello di persistenza, e i controller, i quali si occupano invece di validare e preparare i parametri ricevuti affinché questi possano essere processati dai livelli sottostanti; inoltre essi si occupano anche di formattare la risposta ricevuta da tali livelli di modo che essa possa essere successivamente inviata in risposta al client che ha originariamente effettuato la richiesta.

3.3.3. Persistenza dei dati

Lo stoccaggio delle informazioni avviene su due fronti distinti. Il primo fronte è lato front-end: infatti i dati dell'utente vengono sincronizzati con il server dal client e poi successivamente archiviati in chiaro su quest'ultimo. Questo permette da un lato di avere accesso ai messaggi e alle conversazioni in modo piuttosto rapido mentre dall'altro lato permette di implementare funzionalità come per esempio la ricerca fra i messaggi, funzionalità che sarebbe impossibile da implementare lato server poiché i messaggi archiviati sul cloud sono cifrati pertanto non accessibili e di conseguenza non indicizzabili da un motore di ricerca.

Il secondo fronte su cui avviene lo stoccaggio dei dati è il server: infatti qui è dove vengono memorizzati tutti i dati degli utenti. Per il livello di persistenza lato server la scelta è ricaduta su Apache Cassandra, un database di tipo NoSQL[11], non relazionale e distribuito. Sviluppato originariamente da Facebook per la gestione dei dati relativi alla messaggistica questo DBMS (DataBase Management System) garantisce ottime performance anche qualora sia necessario maneggiare moli di dati enormi, anche dell'ordine dei petabyte se necessario, e permette inoltre di scalare l'infrastruttura orizzontalmente, ovvero connettendo più nodi insieme, in modo particolarmente efficace.

Lato server vi è anche la necessità di archiviare alcuni dati non particolarmente volatili ma generalmente molto consultati in modo tale che la loro lettura risulti estremamente rapida al fine di ottimizzarne l'accesso. Per implementare questo meccanismo di caching dei dati si è deciso di utilizzare Redis, un database NoSQL non relazionale con persistenza facoltativa, ossia i dati vengono di default memorizzati nella memoria volatile (RAM) in quanto anche se persi possono essere facilmente rimpiazzati o comunque rigenerati e ricaricati.

Per semplificare l'aspetto architetturale si è deciso di gestire queste dipendenze software di terze parti attraverso dei servizi cloud, nello specifico viene utilizzata la piattaforma DataStax Astra, un provider DBaaS (Database-as-a-Service) che si appoggia a diversi fornitori di servizi cloud, nel caso di DotLine vengono usati nodi Apache Cassandra gestiti da DataStax Astra e localizzati su AWS (Amazon Web Services). Uno scenario simile avviene anche per quanto

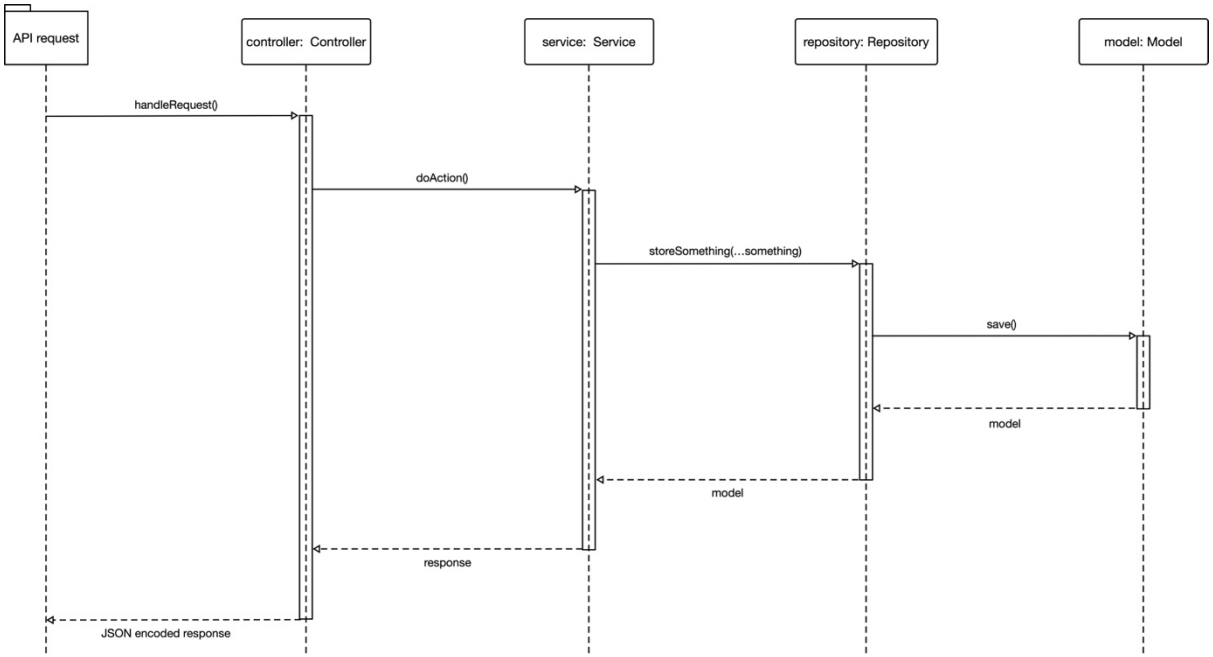
[11] Con NoSQL si intende l'insieme di tutti quei database che non seguono le indicazioni imposte dal linguaggio SQL e più generalmente il modello relazionale.

riguarda Redis, in questo caso il provider scelto è Redis Enterprise Cloud e anche in questo caso i nodi sono localizzati su server cloud forniti e gestiti da AWS.

Da notare come l'uso di AWS si possa rivelare un vantaggio notevole in termini di scalabilità orizzontale, questo poiché AWS ha a disposizione un'enorme flotta di server dispiegati in una moltitudine di paesi nel mondo permettendo non solo di distribuire i carichi su più nodi ma anche di geo-distribuire i dati così da accorciare le distanze fra gli utenti e i vari server.
Relativamente la sicurezza dei collegamenti con questi servizi esterni di terze parti vengono adottate misure di protezione delle comunicazioni affinché i canali siano protetti e l'accesso a tali sistemi sia ristretto: a tal proposito entrambi i fornitori di servizi cloud ammettono la connessione ai loro sistemi attraverso l'uso di certificati SSL e chiavi di sicurezza in modo da garantire un alto grado di protezione degli accessi nonché la cifratura delle informazioni che vengono scambiate fra il server di DotLine e i servizi in cloud.

3.3.4. Considerazioni architetturali del software

Per comprendere meglio come sono strutturati al loro interno il client e il server di DotLine è opportuno fare qualche considerazione relativamente alle scelte architetturali. Partendo dal server, si è anticipato in precedenza che per lo sviluppo ci si è basati sul pattern MVC, tuttavia questo non copre l'intera struttura di entità software; ebbene la struttura delle varie classi è dunque composta da molteplici livelli interconnessi fra loro. Prendiamo l'esempio più comune di un endpoint API in modo da comprendere meglio l'implementazione: come si evince dal diagramma a seguire, ogni richiesta comporta un conseguente avanzamento in profondità nella struttura del software in cui ogni livello rappresenta un diverso grado si specializzazione delle entità coinvolte.



Per avere un'idea più chiara di quelli che sono i vari livelli gerarchici, possiamo riassumere la gerarchia delle entità software presentate e delle relative responsabilità attraverso l'elenco numerato a seguire:

1. **Controller**: si occupa di validare ed eventualmente di preparare i dati ricevuti nonché inizializzare il processamento della richiesta da parte dei livelli sottostanti e in seguito di preparare la risposta in modo che questa possa essere quindi inviata al chiamante.
2. **Service**: implementa la così detta logica di business, da notare però che all'interno del service possono esserci chiamate anche ad altri servizi nonché a uno o più livelli della gerarchia sottostanti.
3. **Repository**: si occupa dell'interazione con le entità relative al livello di persistenza dei dati; tale layer intermedio permette una maggiore flessibilità a livello implementativo, si noti come questo tipo di entità non vengano inizializzate direttamente dai servizi ma bensì vengano "iniettate" nei servizi attraverso il pattern "dependency injection".
4. **Model**: si tratta di un componente software il quale rappresenta l'entità del livello di persistenza dei dati e ne permette quindi le eventuali manipolazioni da parte dei livelli sovrastanti.

In aggiunta a quanto discusso bisogna precisare che esistono anche altre entità software di supporto nonché altre classi, dette provider, le quali implementano tutte le varie operazioni di inizializzazione dell'applicativo e intervengono quando questo viene avviato. Inoltre vi sono alcuni componenti software, detti middleware, che intervengono invece ad ogni richiesta ricevuta dal server e svolgono alcune operazioni prima che la richiesta giunga al controller, un esempio tipico è rappresentato dal middleware che si occupa dell'autenticazione dell'utente. Da notare come alcune interazioni fra entità software avvengono in modo sconnesso, in alcuni casi persino in maniera asincrona: è il caso questo degli eventi e dei relativi "event listeners" che si occupano di agire al verificarsi di un determinato evento, come ad esempio l'invio di un messaggio da parte di un utente o la creazione di una nuova conversazione.

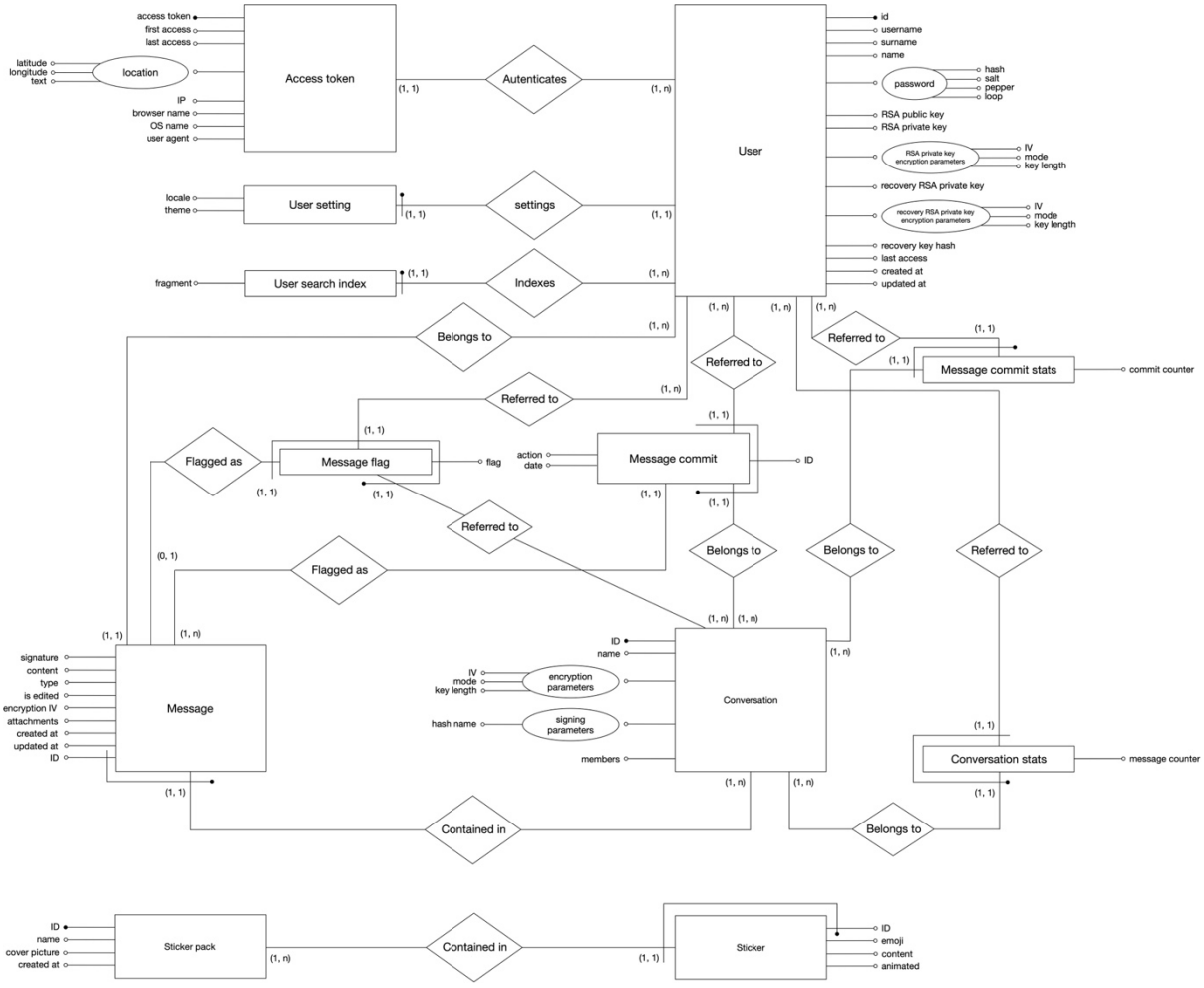
Per quanto concerne il lato client la struttura dei livelli è pressoché identica, l'unica differenza sostanziale risiede nel fatto che non vi sono richieste API ricevute e relativi controller ma bensì vi sono componenti React che implementano gli elementi dell'interfaccia grafica. Le richieste sono rappresentate dalle azioni degli utenti, come ad esempio il click su un bottone, le quali inizializzano la relativa gestione da parte della gerarchia dei livelli sottostanti.
Un'altra differenza importante è che i servizi del client non sempre vanno a manipolare i dati del livello di persistenza ma piuttosto interagiscono con il server attraverso le chiamate REST alle API messe a disposizione del server.

3.3.5. Considerazioni architetturali dei dati

Lavorare con database non relazionali NoSQL come Apache Cassandra implica il dover tenere a mente in fase di progettazione dell'architettura dei dati una serie di vincoli e limiti imposti di fatto dal DBMS stesso. Le limitazioni di Cassandra e del linguaggio CQL[12] utilizzato per le

[12] Il linguaggio CQL (Cassandra Query Language) è un linguaggio simile a SQL (Structured Query Language) ma sviluppato ad-hoc per l'interazione con Apache Cassandra e come tale non ha alcuna relazione con le specifiche alla base del linguaggio SQL.

interrogazioni includono vincoli in fase di filtraggio delle righe sulla quale si desidera operare, in particolare è possibile filtrare i dati usando solo i campi dichiarati come chiave primaria e seguendo l'ordine con il quale essi sono stati dichiarati, pertanto non si può filtrare prima per una certa chiave e poi eventualmente per un'altra se sono state dichiarate in ordine invertito. Esistono essenzialmente due tipi di chiavi primarie in Cassandra: quella di partizione e quelle invece di clustering, le quali definiscono anche l'ordinamento dei dati, che dunque non può essere arbitrario ma deve bensì essere dichiarato insieme alla definizione della tabella stessa. Anche le operazioni disponibili sono limitate: le operazioni su chiavi di partizione sono limitate al solo operatore di uguaglianza mentre invece gli operatori di disuguaglianza possono essere utilizzati solo se non vi sono altre chiavi filtrate dopo di essi e solo su chiavi di clustering.
Per ovviare a queste limitazioni tendenzialmente si fa ricorso a indici secondari oppure a viste, tuttavia su DataStax Astra le viste non sono disponibili, pertanto si è fatto ricorso all'uso degli indici in modo tale da poter implementare tutte le operazioni necessarie alla manipolazione dei dati. Per illustrare più nel dettaglio la struttura della base di dati in cui le informazioni degli utenti sono archiviate si faccia riferimento al diagramma ER (Entity-Relationship) a seguire:



Da notare come l'entità "User search index" sia stata aggiunta in modo da implementare un meccanismo di ricerca full-text applicato al campo username dell'entità User in modo tale da fornire le funzionalità di ricerca degli utenti; in questa entità viene archiviato l'username di un determinato utente in modo "incrementale" così da poter applicare successivamente una selezione attraverso l'operatore di uguaglianza (ad esempio l'indicizzazione della parola "hey"

comporterebbe l'archiviazione di tre righe, ossia "h", "he" e infine "hey"). Questo si è reso necessario in quanto gli indici secondari supportati da DataStax Astra non implementano le funzionalità di ricerca full-text a differenza degli indici SASI di Cassandra; essendo il caso d'uso specifico piuttosto semplice e limitato si è preferita questa soluzione come alternativa all'uso di un indice di ricerca esterno come Lucene oppure Elastic Search.

3.3.6. Deployment

Al fine di semplificare la distribuzione del software si è optato per l'utilizzo della tecnologia di containerizzazione, nello specifico viene utilizzata la piattaforma Docker[13] per la realizzazione dei container in cui vi si trova installato l'applicativo e tutte le sue varie dipendenze software. Questa decisione permette da una parte di ovviare all'installazione e alla conseguente configurazione dell'ambiente di deployment, cosa che richiede inesorabilmente del tempo e può portare a inconvenienti e conflitti, e dall'altro lato permette di distribuire l'applicazione per più architetture in modo molto semplificato; infatti le immagini Docker realizzate, una per il client e una per il server, sono archiviate e distribuite sul cloud attraverso la piattaforma online Docker Hub.

L'ambiente di produzione di DotLine è composto da server VPS (Virtual Private Server) basati su Linux, nello specifico equipaggiati con il sistema operativo Red Hat Enterprise Linux 9.
Per quanto concerne la scelta dei server, in particolare a livello di hardware, è stato fatto un ragionamento che tiene conto anche della sostenibilità ambientale: infatti i server designati montano processore Ampere Altra, i quali sono basati sull'architettura ARM (Advanced RISC Machine) v8 a 64 bit, architettura nota per la sua estrema efficienza in termini energetici nonché termici e che pertanto necessita di un minore quantitativo di energia per il suo funzionamento e il conseguente raffreddamento.

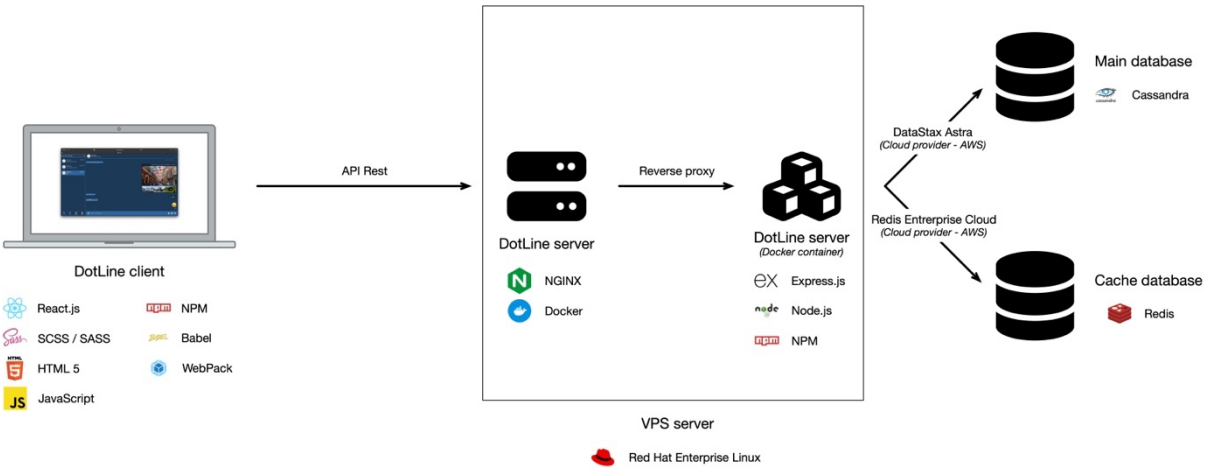
L'ambiente di produzione è inoltre configurato in modo tale che le richieste HTTP ricevute non vengano gestite direttamente dall'applicativo in sé, installato sottoforma di container Docker, ma piuttosto vengano gestite da un web server separato, nello specifico NGINX[14], il cui scopo è quello di permettere una più mirata gestione degli aspetti legati alle richieste HTTP, come per esempio la compressione dei dati, implementata attraverso l'algoritmo Brotli[15] (GZip come eventuale fallback), HTTPS e relativi certificati SSL (Secure Sockets Layer), nonché il supporto al protocollo HTTP/3[16] per migliori performance di caricamento dei dati richiesti.

[13] Docker è una tecnologia open-source che permette di eseguire applicazioni in ambienti isolati chiamati container su ambiente Linux. A differenza dei meccanismi di virtualizzazione esso si appoggia su funzionalità fornite dal kernel Linux stesso che permettono dunque di isolare processi e limitate le risorse risultando quindi molto più efficienti rispetto alle tradizionali macchine virtuali ottenendo tuttavia risultati molto simili.
[14] NGINX è uno dei più diffusi webserver particolarmente adatto ad applicativi che necessitano di poter scalare; la sua architettura ad eventi lo rende piuttosto efficiente in termini di risorse richieste e le sue funzionalità di reverse proxy lo rendono ottimo da usare di fronte a un applicativo scritto con Node.js.
[15] L'algoritmo Brotli è un algoritmo di compressione sviluppato da Google che si rivela piuttosto efficiente in termini di risultati prodotti a livello di percentuale di compressione in particolar modo su dati testuali.
[16] La terza versione del protocollo HTTP si appoggia al protocollo di trasporto QUIC, concepito da Google esso combina la comunicazione via TCP con quella via UDP per massimizzare le performance, specialmente su reti mobile.

Un altro dei motivi che hanno portato alla scelta del webserver NGINX come alternativa alla gestione diretta delle richieste da parte dei container Docker sottostanti è sicuramente quello di permettere una migliore gestione e relativa distribuzione delle richieste nell'ottica della scalabilità del sistema e del relativo load balacing.

Relativamente all'ambiente di staging e test di DotLine, questo si basa sulla piattaforma cloud Heroku e sul sistema operativo Ubuntu Linux. Per isolare l'ambiente di staging dall'ambiente di produzione si è deciso di utilizzare un database diverso ma offerto e gestito dallo stesso provider cloud su cui già si appoggia il database di produzione. L'ambiente di staging viene amministrato da remoto attraverso l'uso dell'interfaccia CLI (Command-Line Interface) offerta da Heroku ed è accessibile attraverso un sottodominio dedicato nonché protetto da un livello di autenticazione aggiuntivo atto a impedire accessi accidentali e non autorizzati.

Lo schema a seguire illustra la struttura complessiva del progetto e delle relazioni fra le sue componenti, sia interne piuttosto che di terze parti, nonché il comparto tecnologico coinvolto da ciascuna di esse:



3.3.7. Monitoraggio

È giusto spendere un paio di parole anche sulla fase successiva all'implementazione, ovvero l'esecuzione quotidiana del software. Infatti capita talvolta che si possano verificare errori o altri tipi di malfunzionamenti dovuti a bug o ad azioni errate o non gestite da parte degli utenti. È opportuno dunque essere in grado di rilevare e tracciare tali problematiche ogniqualvolta si verificano. A tal fine si è deciso di equipaggiare sia il client che il server di DotLine con Sentry, un servizio che permette di rilevare eventuali errori ed eccezioni e successivamente notificare in tempo reale via e-mail il team di sviluppo di un progetto fornendo tutti i dettagli relativi al problema nonché il punto preciso del codice in cui esso si è verificato così da poter intervenire tempestivamente per risolverlo.

Un'altra problematica è quella di verificare e garantire che il servizio non vada offline. Al fine di prevenire una caduta del servizio, o perlomeno di rilevarla e dunque poter agire il prima possibile, si è deciso di appoggiarsi al servizio StatusCake il quale permette di monitorare un servizio web, in questo caso il client e il server di DotLine, e quindi avvisare istantaneamente,

via e-mail o tramite SMS o telefonata, qualora il servizio in questione vada offline o si verifichi un altro tipo di disservizio che ne pregiudichi l'accesso, come per esempio la scadenza di un certificato SSL o un problema lato DNS (Domain Name System).

3.4. Implementazione

Seguono ora alcune note ed eventuali considerazioni in merito al come si sono svolti i lavori per la realizzazione del progetto e alle varie misure adottate per organizzare le attività e gli artefatti prodotti.

3.4.1. Modalità di organizzazione del lavoro

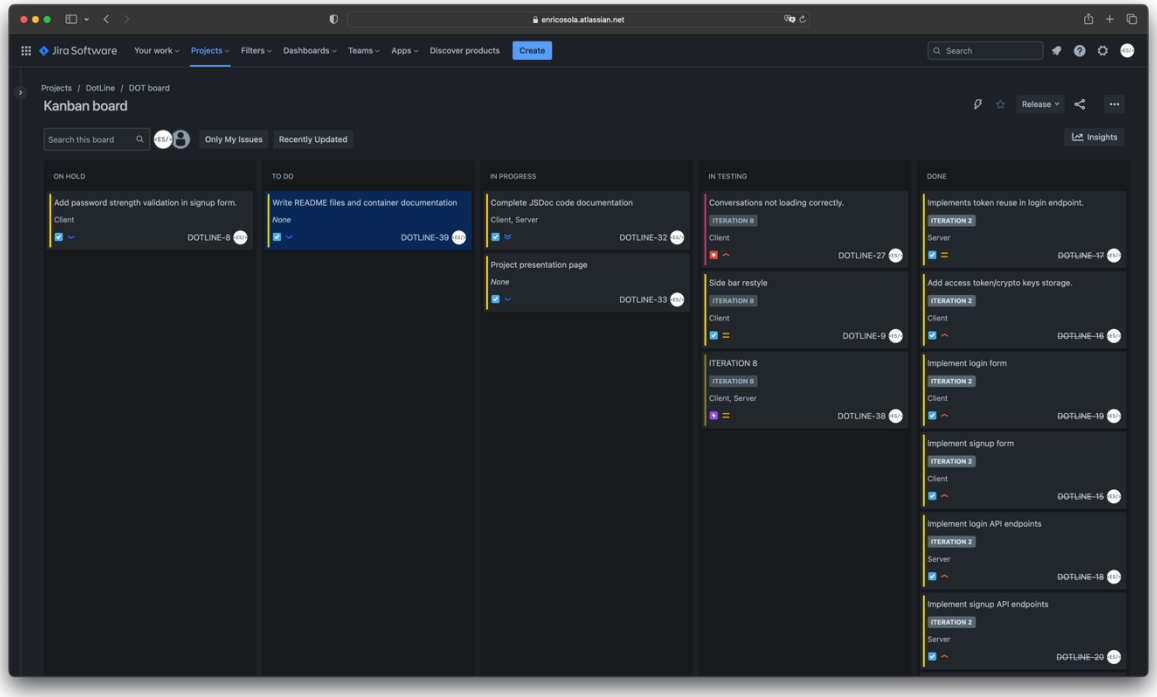
Lo sviluppo di DotLine è stato suddiviso in due tronconi paralleli, uno per il server e uno per il client, i quali seguono il modello di sviluppo iterativo. Gli sviluppi vengono suddivisi in task che a loro volta vengono pianificati e organizzati attraverso il modello Kanban e assegnati alle varie iterazioni al fine di avere, al termine di ogni iterazione, un applicativo funzionante nel quale sono implementate un sotto insieme di tutte le funzionalità previste per il progetto finito e completo. Il lavoro di gestione del progetto, comprensivo dell'organizzazione dei task all'interno della Kanban, avviene attraverso l'uso del software online Atlassian Jira, il quale permette di tracciare, gestire e pianificare i task, le iterazioni e le eventuali problematiche che possono venire riscontrate nel software durante le fasi di sviluppo e di testing; quest'ultimo si è rivelato uno strumento di fondamentale importanza per tenere traccia dei progressi fatti ma anche di eventuali problematiche riscontrate così come possibili idee e suggerimenti da approfondire e nel caso implementare.

Anche la gestione del codice prodotto è un aspetto che deve essere valutato e gestito in maniera opportuna per evitare perdite di materiale o conflitti nell'attività organizzativa. A tal fine si è deciso di appoggiarsi al software git, il software di versioning più diffuso al mondo tanto da essere quasi considerato uno standard de-facto nel settore. Il codice in sé viene invece stoccato su cloud in modo tale da ricevere una maggiore sicurezza contro eventuali perdite accidentali di dati e permettere inoltre la condivisione del software fra eventuali altri collaboratori. A tal proposito si è scelto di utilizzare la piattaforma online GitHub, anch'essa ormai considerata un punto di riferimento per lo sviluppo software moderno.

Per agevolare la comprensione e l'analisi del codice sorgente prodotto, anche da parte di eventuali agenti automatici utilizzati durante la scrittura del codice in sé, ad esempio integrati all'interno dell'IDE (Integrated Development Environment) utilizzato, il progetto è stato interamente documentato seguendo lo standard JSDoc[17]; questo permette a chi ha la necessità di leggere il codice di capire facilmente il funzionamento delle varie unità software che lo compongono. Da notare che tutta la documentazione del codice, così come anche la

[17] Lo standard per la documentazione del codice JavaScript JSDoc permette di documentare attraverso l'uso dei commenti i vari aspetti delle unità software, come ad esempio parametri, eccezioni e valori di ritorno, in modo tale che risultino leggibili sia da umani che dai software che li utilizzano per fornire suggerimenti e controlli sulla correttezza del codice.

documentazione esterna al codice in sé, come ad esempio la presentazione del progetto e la relativa descrizione del suo utilizzo e configurazione, è stata scritta in lingua inglese proprio per agevolare la comprensione del progetto da parte di tutti.



Visuale della Kanban utilizzata per organizzare le attività di sviluppo di DotLine su Atlassian Jira.

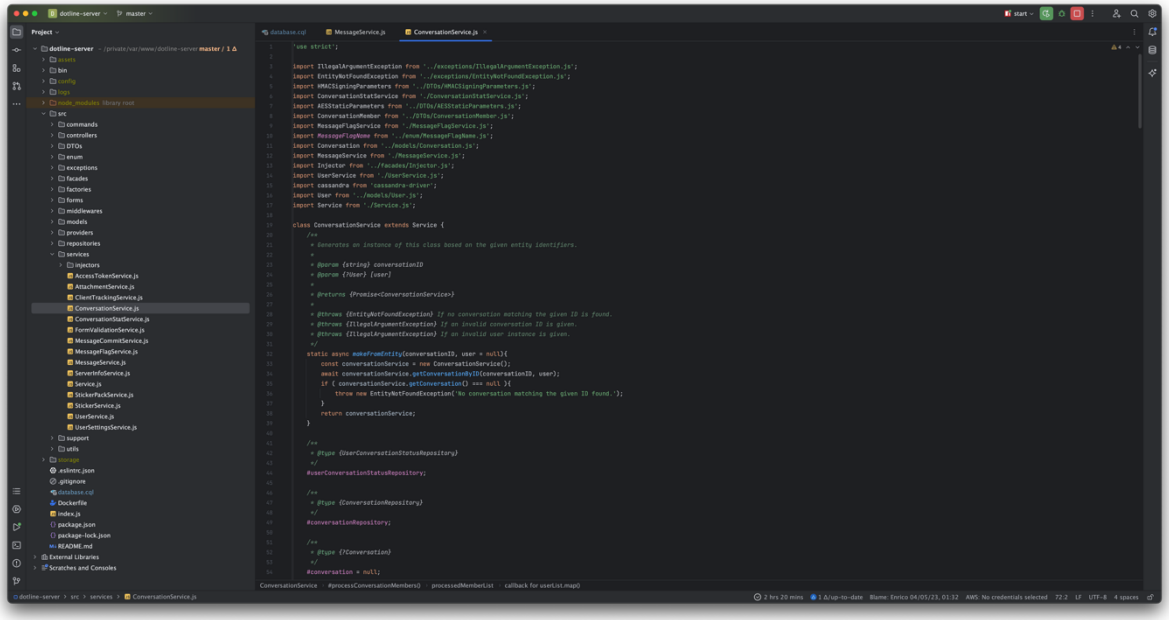
3.4.2. Ambiente di sviluppo

Le attività di sviluppo del progetto si sono svolte su un ambiente parzialmente simile a quello di produzione, nello specifico si è optato per un ambiente basato su macOS, anche in questo caso su piattaforma ARM v8 (Apple Silicon M1 Max). Per quanto riguarda invece la scelta dell'IDE, quest'ultima è ricaduta su WebStorm di Jetbrains, il quale è dotato di una serie di strumenti integrati particolarmente utili per lo sviluppo basato su JavaScript fra i quali una potente interfaccia per il debugger integrato all'interno di Node.js, un client per git piuttosto comodo con il supporto alle note relative alle modifiche per ogni riga del codice nonché una serie di integrazioni con i principali software di supporto per lo sviluppo basato su JavaScript come NPM, WebPack ed ESLint. Sempre rimanendo su questo tema, si è deciso di adottare l'uso di ESLint, un pacchetto concepito per Node.js e installato attraverso il package manager NPM introdotto in precedenza il cui scopo è quello di uniformare lo sviluppo con JavaScript imponendo una serie di regole sintattiche completamente personalizzabili volte a segnalare eventuali problematiche relative alla formattazione o altri aspetti del codice sorgente scritto. Per quanto concerne invece lo sviluppo del client di DotLine si è optato per l'uso del browser Firefox Developer Edition il quale integra diversi strumenti dedicati allo sviluppo e al debug degli aspetti legati sia al codice JavaScript sia allo styling particolarmente efficaci e potenti.

Il setup dell'ambiente locale richiede anche la presenza delle varie dipendenze software di terze parti, come per esempio i database. Per semplificare il setup dell'ambiente di sviluppo

è possibile appoggiarsi a Docker, oppure, come nel caso di questo progetto, si può optare per l'installazione delle dipendenze direttamente sulla macchina in uso; nello specifico è stato quindi necessario munirsi di Apache Cassandra e Redis i quali sono stati installati localmente attraverso l'uso di Homebrew, un gestore di pacchetti software di terze parti con interfaccia CLI particolarmente diffuso e utilizzato in ambiente Apple.

Oltre allo sviluppo è spesso necessario testare i risultati prodotti ed eventualmente svolgere operazioni di debug; talvolta serve compiere queste mansioni su altri dispositivi, ad esempio smartphone o tablet. Per potersi interfacciare con l'ambiente di sviluppo al di fuori di esso è stato utilizzato un servizio chiamato Ngrok al fine di creare dei tunnel in grado di collegare il browser dall'esterno dell'ambiente di sviluppo a quest'ultimo. Questa soluzione si è rivelata particolarmente utile per semplificare le fasi di test dell'applicativo sui simulatori in locale, per Android e iOS, nonché per i test su dispositivi fisici, nello specifico su iPhone e iPad.



Visuale del codice sorgente JavaScript relativo al server di DotLine su WebStorm.

3.5. Scelta dell'interfaccia grafica

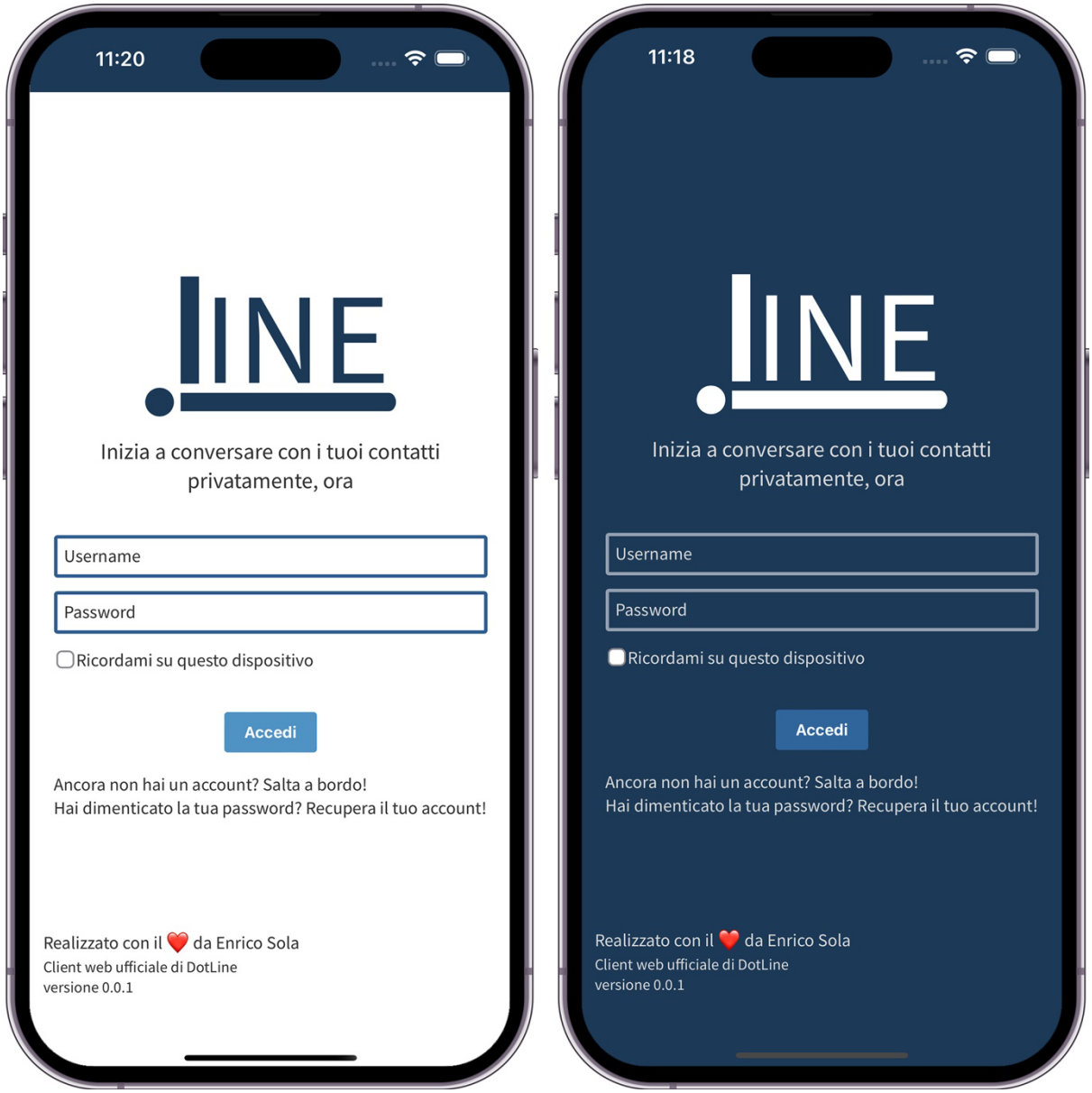
L'idea alla base della concezione dell'interfaccia grafica è quella di mantenere le linee del software il più semplici e minimali possibile al fine di non risultare ambigue e fuorvianti agli occhi dell'utilizzatore, il quale si troverà innanzi a un'interfaccia intuitiva e di facile utilizzo.
Nella schermata principale dell'applicativo troveremo dunque una barra laterale che fornisce l'accesso diretto alla lista di tutte le conversazioni aperte, selezionando la conversazione sarà possibile navigare attraverso i messaggi, i quali verranno caricati man mano che si naviga all'interno della conversazione aperta. Nella parte inferiore della barra laterale si trova invece il menu che permette di navigare le ulteriori sezioni disponibili fra cui la sezione dedicata alle impostazioni dell'account e quella dedicata invece al monitoraggio degli accessi effettuati. La barra di ricerca è sempre disponibile e visibile indipendentemente dalla sezione in uso nella barra laterale così da fornire una porta sempre visibile all'utente verso le sue informazioni.

L’utente ha la possibilità di personalizzare sia la lingua che il tema dell’interfaccia grafica, il quale verrà impostato automaticamente di default in base alle preferenze rilevate dal sistema operativo in uso.

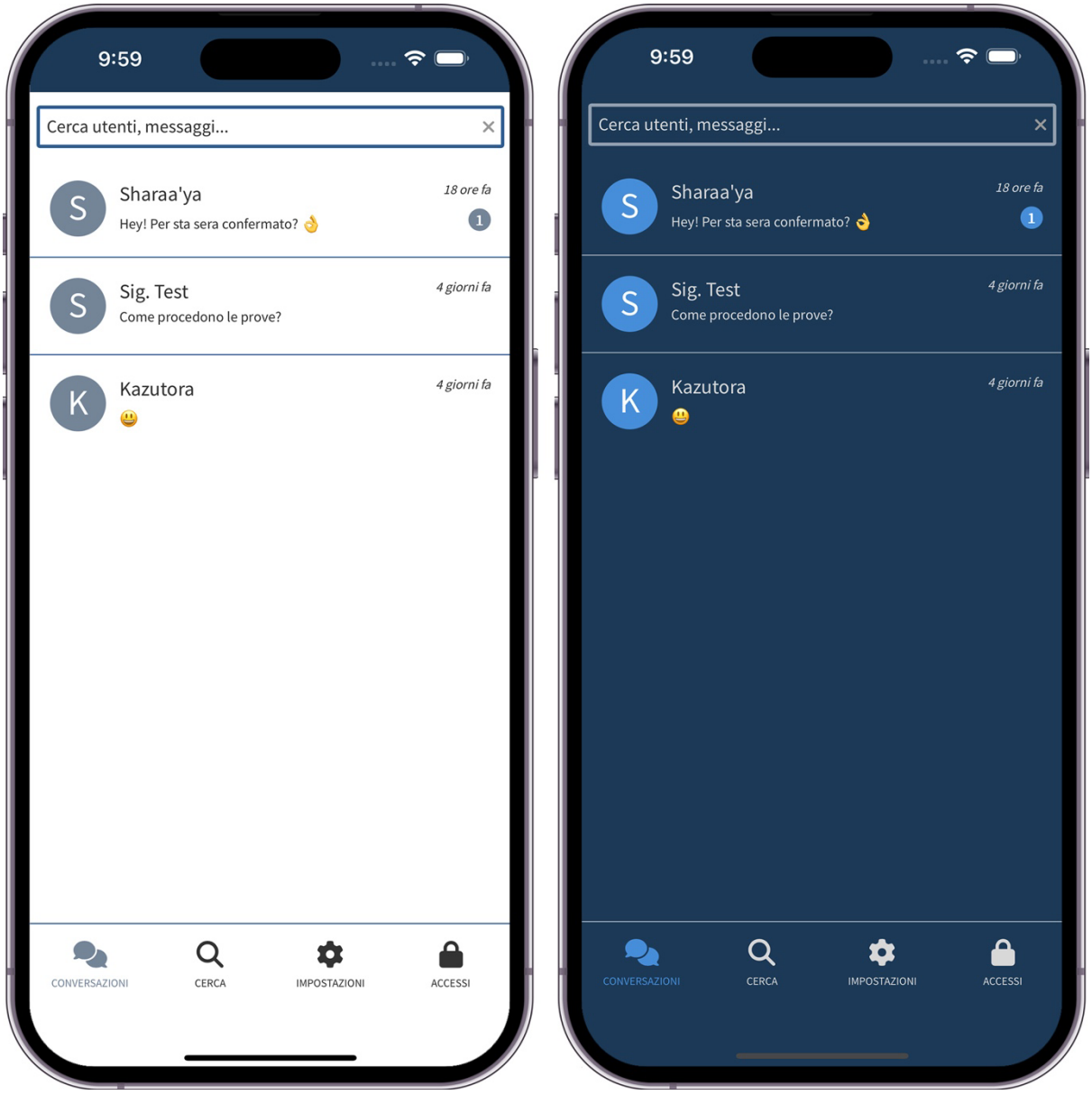
Da notare come l’interfaccia si adatti alle dimensioni del dispositivo in uso in modo tale da risultare perfettamente utilizzabile e accessibile anche nel caso in cui si decida di usufruire del software attraverso dispositivi di piccole dimensioni, come ad esempio uno smartphone. Questa scelta è stata pressoché naturale vista la tendenza al giorno d’oggi di accedere alla rete sempre più di frequente attraverso i dispositivi mobili, i quali hanno ormai superato i dispositivi desktop nello scenario collettivo.

3.5.1. Galleria schermate e mockup

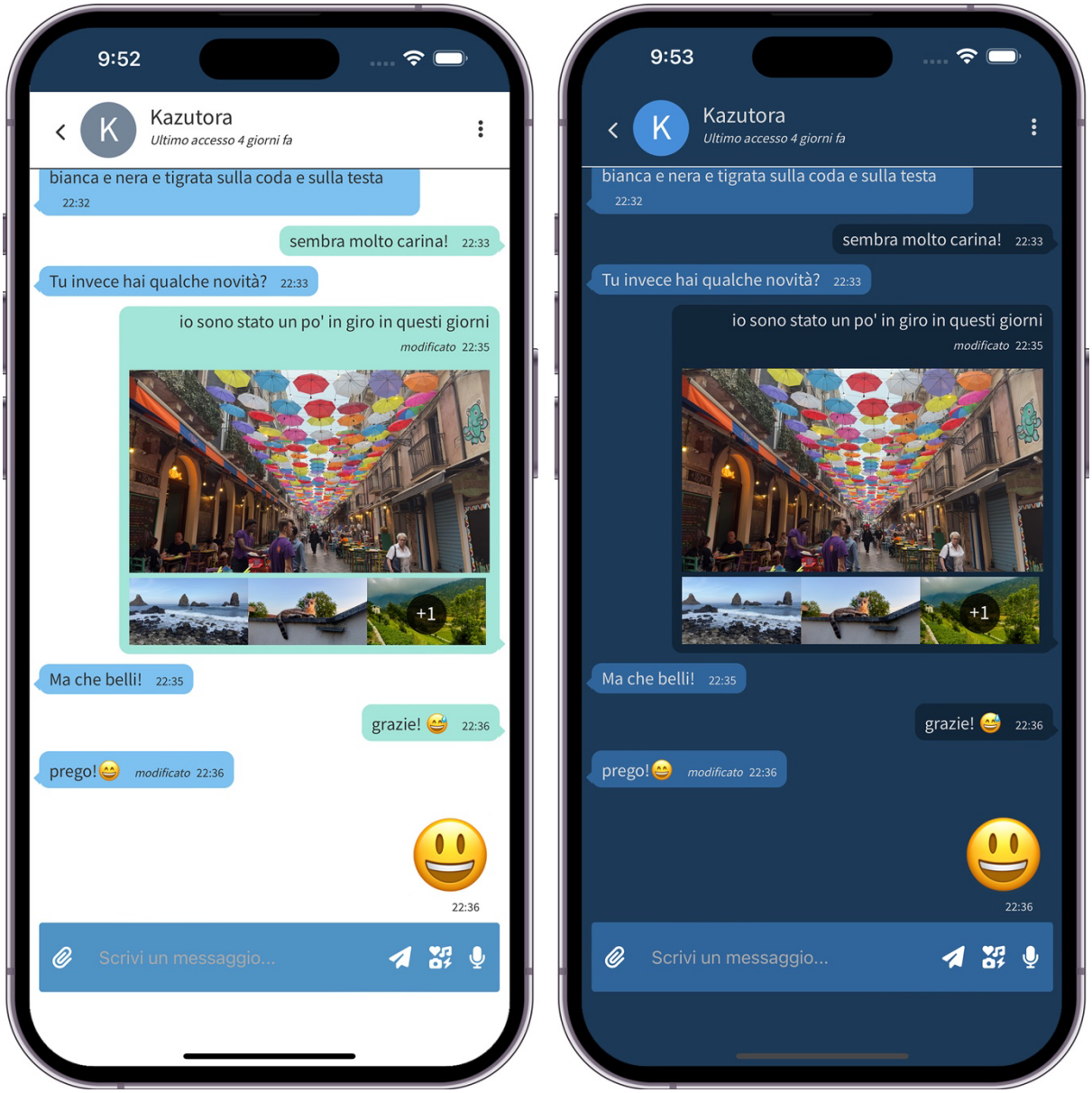
A partire dalla pagina successiva segue una serie di schermate e mockup vari che illustrano la grafica adottata su DotLine sia nel caso in cui vengano utilizzati dispositivi mobili sia nel caso di dispositivi desktop; tutte le schermate sono corredate sia della visuale in modalità light che della visuale in modalità dark.



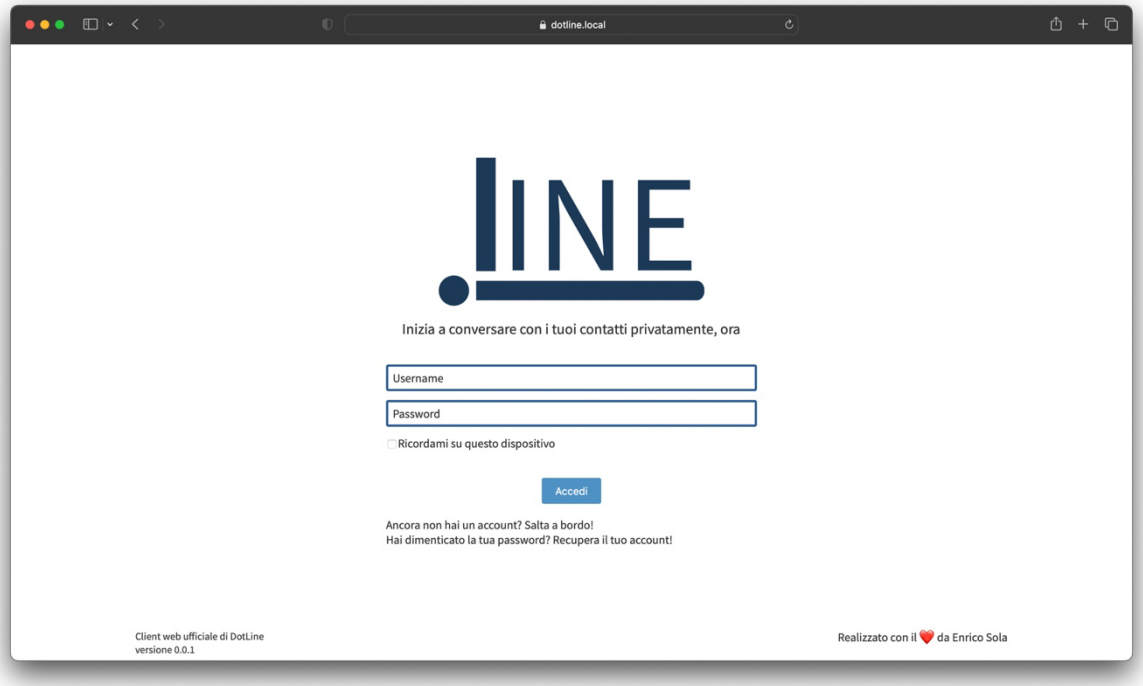
Schermata di autenticazione, ossia la schermata iniziale che accoglie gli utenti prima di accedere al servizio.

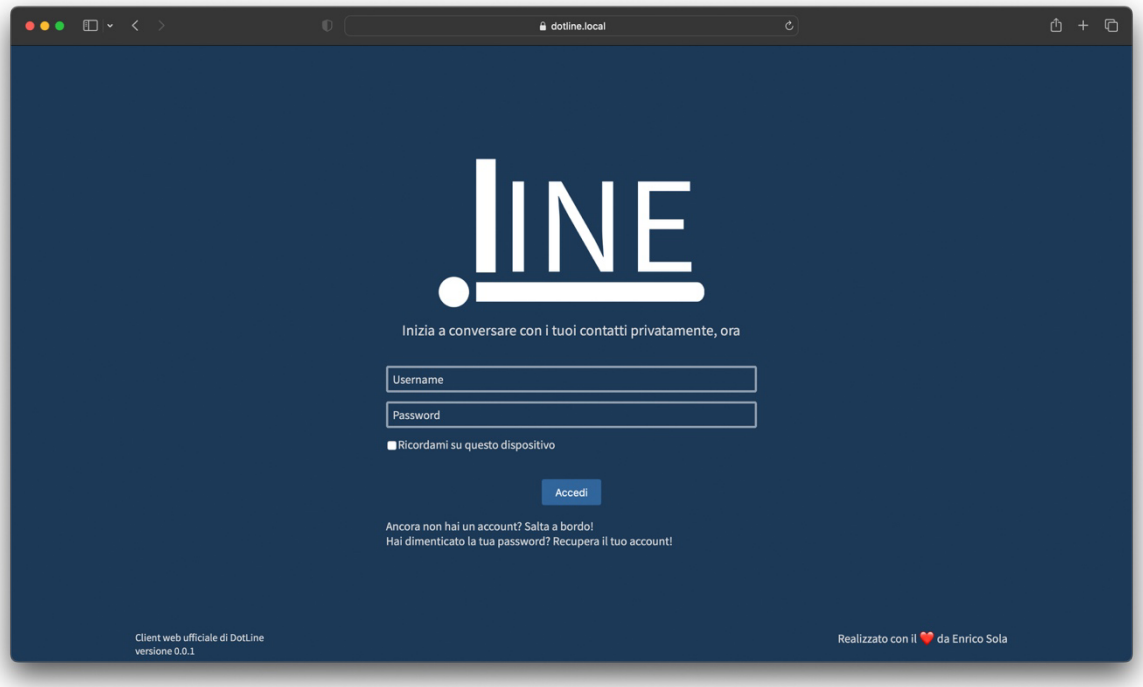


Lista delle conversazioni, da qui è anche possibile accedere alla ricerca e alle altre funzionalità dal menù in basso.

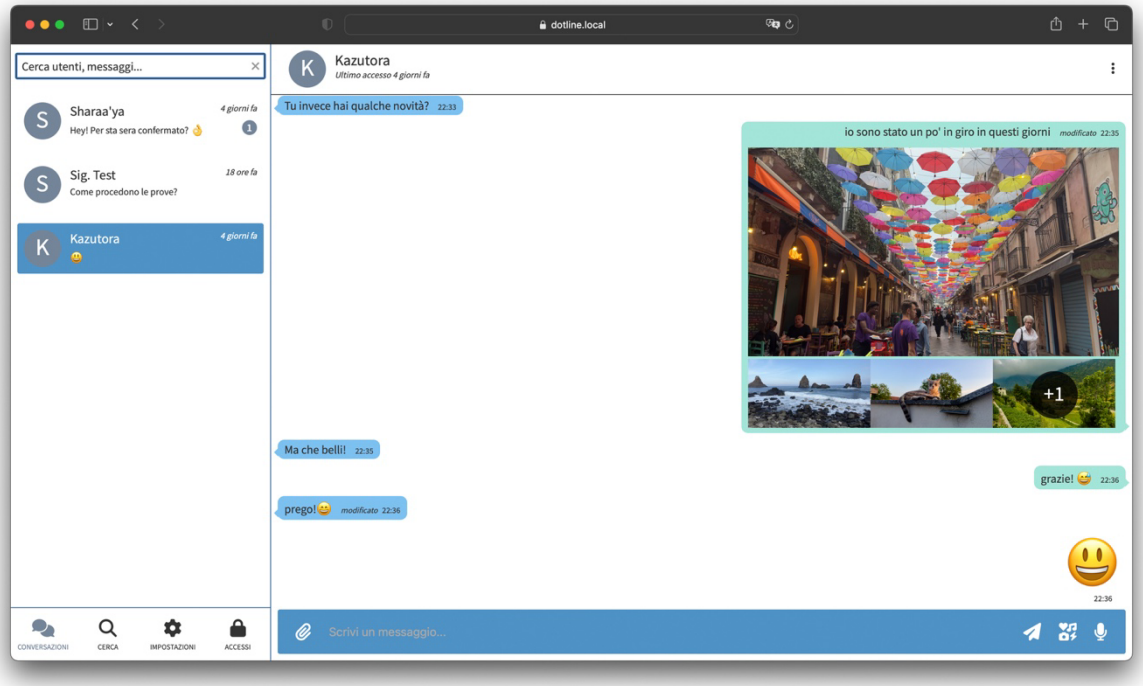


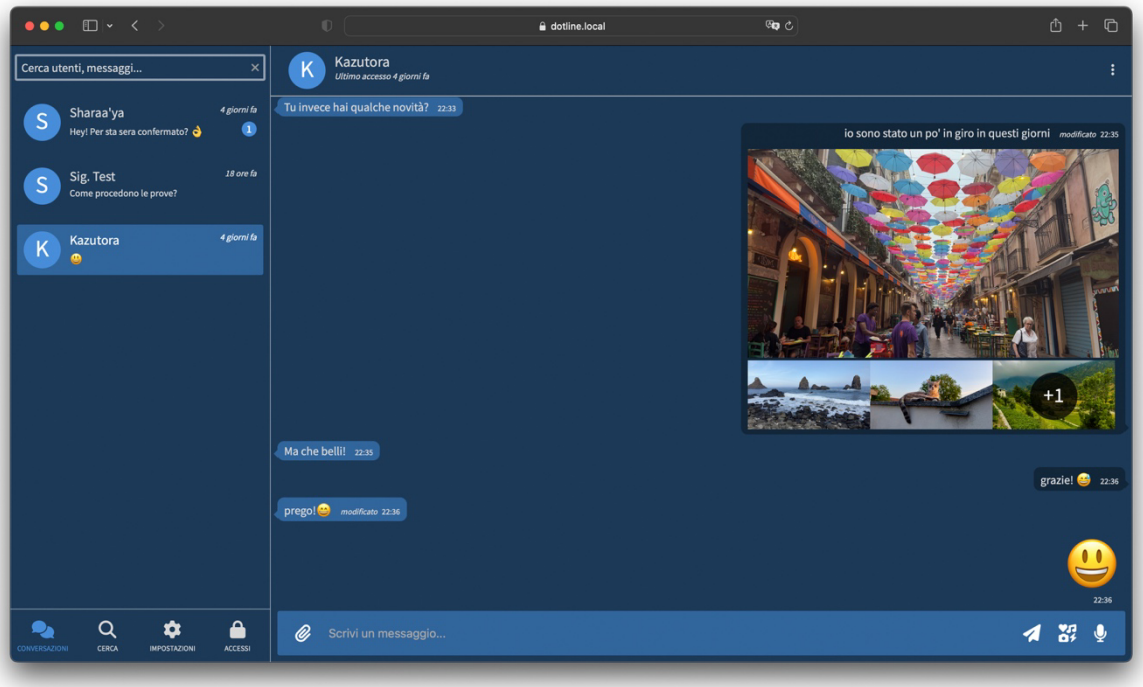
Conversazione aperta vista da un dispositivo mobile (light mode + dark mode).





Schermata di autenticazione in visuale desktop.





La schermata principale dalla quale si può accedere sia alla lista delle conversazioni che ad una precisa conversazione.

4. Considerazioni finali

La realizzazione di DotLine è stato un percorso che ha lanciato sia sfide per il presente ma al contempo anche porte aperte per il futuro del progetto. In questo capitolo si vuole un po' riassumere quella che è stata l'esperienza di sviluppo di un tale progetto nel suo complesso, le sfide affrontate ma anche le aspettative per il futuro.

4.1. Sfide e difficoltà affrontate

Lo sviluppo di questo progetto ha portato con sé senza ombra di dubbio una considerevole serie di problematiche e di questioni che si sono rivelate un'interessante sfida da affrontare, tra le principali si possono menzionare le questioni tecniche relative allo sviluppo, in particolar modo tutta quella che è la gestione dei vari flussi crittografici, dalla cifratura del dato alla decifratura una volta giunto alla sua destinazione mantenendolo integro e dunque privo di fenomeni di corruzione dovuti a eventuali errori indipendentemente dalla sua natura, testuale oppure binaria che sia. Certamente degna di nota anche la difficoltà nel trovare un meccanismo efficiente per poter sincronizzare non solo i messaggi in sé ma anche le relative modifiche attraverso tutti i dispositivi coinvolti; questo si è quindi tradotto sia nel trovare un algoritmo efficace per sincronizzare questo tipo di informazioni ma anche trovare un mezzo altrettanto efficiente per propagare questi dati in tempo reale. Infine, ma non per importanza, vi è l'aspetto relativo alla gestione dei dati in locale, aspetto necessario al fine di rendere il sistema realmente coerente con quella che è la nozione di crittografia end-to-end, ovvero il fulcro di tutto il software, ma al contempo fornire all'utente alcune delle funzionalità che egli si aspetta ma che non sarebbe possibile implementare lato server senza infrangere quello che è lo scopo ultimo di questo meccanismo crittografico. Ovviamente in questo tipo di progetti non vi sono solo difficoltà e sfide di carattere tecnico, il grado di privacy e protezione offerto da DotLine è sì un bene per l'utente ma apre anche un importante dibattito di carattere etico e morale in quanto l'assenza di controllo favorisce il proliferare di eventuali attività illecite che in questo contesto sono estremamente difficili da intercettare ed eradicare.

Nel complesso dunque DotLine si è rivelato un progetto particolarmente interessante su tutti i fronti nonché decisamente stimolante nella sua concezione e realizzazione, un progetto che necessita di un ragguardevole livello di competenze ma dal quale si può comunque imparare sicuramente molto.

4.2. Conclusioni e sviluppi futuri

Uno degli scopi che sin da subito è stato fisso e chiaro è certamente quello di voler realizzare un sistema di messaggistica istantanea completamente funzionante nonché aperto e fruibile dal pubblico. Sebbene il progetto sia nato in un contesto accademico il risultato finale è un software completo e funzionale sin dalla sua prima versione nonché in grado di adempiere a una buona parte delle esigenze comunicative che gli utenti ricercano e dunque si aspettano da un'applicazione di messaggistica online oggigiorno.

Il prodotto ottenuto getta le basi per una serie di aggiornamenti futuri che puntano a rendere DotLine un'applicazione ancora più versatile e potente da usare. Fra le funzionalità pianificate per le future versioni vi sono:

- Il supporto alle conversazioni fra più utenti, come ad esempio gruppi e canali, e in un secondo momento l'apertura ai bot.
- L'introduzione delle chiamate audio alle chiamate audio-video, anch'esse protette dalla crittografia end-to-end.
- L'introduzione di nuove funzionalità riguardanti i singoli messaggi, come ad esempio la condivisione in tempo reale della posizione geografica e le reazioni ai messaggi.
- L'aggiunta del supporto all'autenticazione "password-less", ovvero senza l'uso di una password ma bensì di un meccanismo di riconoscimento biometrico (esempio TouchID o FaceID) oppure mediante una chiave di sicurezza hardware Fido2[18].

Un altro punto sicuramente degno di nota è quello relativo alla distribuzione del software: al fine di alimentare il flusso di idee e l'innovazione si è deciso che DotLine non dovrebbe soltanto essere libero da usare come prodotto finito, ma anche come servizio standalone nonché dovrebbe essere modificabile da chiunque lo desideri. A tal scopo si è concordato di rendere l'intero progetto open source: chiunque può quindi creare il proprio servizio di messaggistica in modo rapido usufruendo delle immagini Docker del client e del server di DotLine, le stesse utilizzate in fase di deployment, le quali sono rese pubblicamente utilizzabili direttamente da Docker Hub. Inoltre, qualora lo si desideri, è possibile scaricare l'intero codice sorgente e modificarlo a proprio piacimento; infatti l'intero codice sorgente del client e del server è pubblicamente accessibile e scaricabile dalla piattaforma GitHub.

[18] Fido2 è un protocollo aperto per la realizzazione di uno standard di autenticazione per il web. Al giorno d'oggi i principali browser desktop e mobile implementano il supporto a questo standard e sono disponibili una moltitudine di dispositivi hardware che supportano il protocollo e permettono quindi l'autenticazione attraverso il loro collegamento al browser via USB o wireless (NCF oppure Bluetooth).

5. Bibliografia e sitografia

Comparto teorico

https://it.wikipedia.org/wiki/Derivazione_di_una_chiave_crittografica
https://it.wikipedia.org/wiki/Advanced_Encryption_Standard
https://it.wikipedia.org/wiki/Secure_Hash_Algorithm
https://it.wikipedia.org/wiki/Crittografia_end-to-end
https://it.wikipedia.org/wiki/RSA_(crittografia)
https://it.wikipedia.org/wiki/Kanban
https://it.wikipedia.org/wiki/HMAC

Comparto tecnico e tecnologico

https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux
https://it.wikipedia.org/wiki/Representational_state_transfer
https://www.datastax.com/products/datastax-astra
https://it.wikipedia.org/wiki/Model-view-controller
https://redis.com/redis-enterprise-cloud/overview
https://it.wikipedia.org/wiki/Cloud_database
https://it.wikipedia.org/wiki/NoSQL
https://cassandra.apache.org
https://www.heroku.com
https://www.docker.com
https://www.npmjs.com
https://webpack.js.org
https://sass-lang.com
https://nginx.org/en/
https://nodejs.org/it
http://expressjs.com
https://jsstore.net
https://babeljs.io
https://react.dev
https://redis.io

Software utilizzati per lo sviluppo

https://www.atlassian.com/it/software/jira
https://www.jetbrains.com/webstorm
https://www.statuscake.com
https://git-scm.com
https://ngrok.com
https://eslint.org
https://sentry.io
https://brew.sh

API HTML5 utilizzate dal client di DotLine

https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia
https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API
https://developer.mozilla.org/en-US/docs/Web/API/Intersection_Observer_API
https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API
https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API
https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API
https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API
https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API
https://developer.mozilla.org/en-US/docs/Web/API/Clipboard_API
https://developer.mozilla.org/en-US/docs/Web/API/Encoding_API
https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
https://developer.mozilla.org/en-US/docs/Web/API/URL_API
https://developer.mozilla.org/en-US/docs/Web/API/File_API

Collegamenti al progetto

https://hub.docker.com/r/enricosola/dotline-server
https://hub.docker.com/r/enricosola/dotline-client
https://github.com/RyanJ93/dotline-server
https://github.com/RyanJ93/dotline-client
https://dotline.enricosola.dev

6. Ringraziamenti

La realizzazione di questo progetto non sarebbe stata possibile senza la collaborazione e il supporto di diverse altre persone: con la presente desidero ringraziare tutti coloro che hanno prestato il loro tempo e che dunque hanno reso possibile questo traguardo; in particolar modo desidero esprimere la mia sincera gratitudine a tutti i beta tester che hanno contribuito con i loro feedback a rendere DotLine un prodotto stabile e completo nonché a tutti coloro che hanno contribuito con i loro suggerimenti in materia di design e di user experience.

Desidero inoltre ringraziare il prof. Marino Segnan per la disponibilità, l'attiva collaborazione nonché il supporto offertomi nel concepire e perfezionare questo elaborato.

Un particolare ringraziamento va anche a tutti coloro che mi hanno espresso la loro vicinanza e che mi hanno sostenuto non solo durante la realizzazione di questo progetto e dell'annessa relazione ma anche durante l'intero percorso accademico. Prendersi esclusivamente il merito di questo lungo cammino sarebbe superficiale ed egoista da parte mia, pertanto desidero ringraziare coloro che hanno reso possibile questa grande impresa, dal primo giorno fino ad oggi, e ad essi vorrei dedicare questi risultati, grazie davvero!

DICHIARAZIONE DI ORIGINALITÀ

Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata.