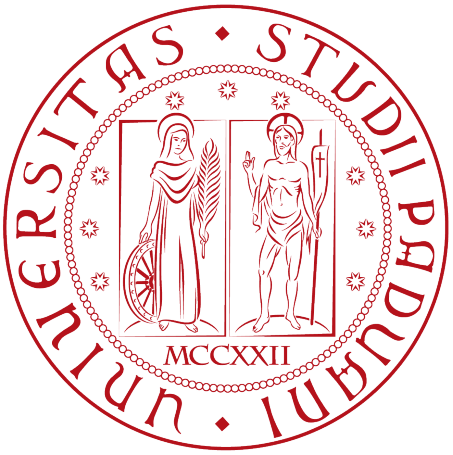
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Estensione delle funzionalità di videoispezione per il miglioramento dell'esperienza utente di un software aziendale

Tesi di laurea triennale

Relatore

Prof. Luigi De Giovanni

Laureando

Michele Masetto

Numero matricola

1224457

Anno Accademico 2021-2022

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di trecentoventi ore, dal laureando Michele Masetto presso l'azienda RiskApp S.r.l..

L'obiettivo del progetto di stage è stato quello di sviluppare alcune funzionalità per il miglioramento dell'esperienza utente durante l'utilizzo dell'applicativo aziendale che permette la videoispezione di siti da remoto da parte degli utenti, e di inserire alcune funzionalità mancanti.

Durante lo svolgimento dello stage, innanzitutto si è svolta l'analisi dei requisiti, si è passati poi alla progettazione, e infine allo sviluppo e alla verifica delle funzionalità implementate. Tali fasi vengono analizzate nel dettaglio nei capitoli che compongono la presente tesi.

Convenzioni tipografiche

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

* gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

"Life is really simple, but we insist on making it complicated"

— Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Luigi De Giovanni, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Dicembre 2022 Michele Masetto

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 L'azienda

RiskApp S.r.l. (logo in Figura 1.1) è un azienda di sviluppo software fondata nel 2016 e con sede a Conselve, in provincia di Padova. Il suo prodotto di punta è la cosiddetta piattaforma RiskApp, la cui vendita e mantenimento è il suo *core business*.

Si tratta di un software pensato per fornire ai suoi clienti, cioè figure del mondo assicurativo come agenti e *broker*, un quadro più completo possibile di tutti i rischi a cui possono andare incontro le compagnie assicurative. Infatti, grazie ad un algoritmo proprietario e sfruttando l'intelligenza artificiale, vengono raccolti dati provenienti da diverse fonti, dati che permettono di stimare le possibili perdite economiche di un'impresa.

Il prodotto viene distribuito come Software as a Service (SaaS)[g] [32] e ad oggi l'azienda intraprende relazioni commerciali con importanti *player* del mondo assicurativo, inoltre intrattiene rapporti di collaborazione con l'Università degli Studi di Padova.

Figura 1.1: Logo dell'azienda RiskApp

1.1.1 Processi aziendali

Essendo il gruppo di lavoro composto da un numero esiguo di individui, l'azienda si approccia allo sviluppo del software riprendendo alcuni concetti chiave della metodologia agile[g] [2]:

- coinvolgimento del cliente: vengono effettuati frequentemente degli incontri con i clienti, di solito virtuali ma talvolta anche di persona, per l'individuazione e la revisione dei requisiti;
- comunicazione istantanea: le comunicazioni all'interno del team avvengono in modo rapido. Infatti l'ambiente di lavoro piccolo permette di comunicare velocemente con gli altri membri, mentre per la comunicazione da remoto si utilizzano strumenti appositi;

* sviluppo iterativo: le varie componenti del software vengono continuamente sviluppate e migliorate nel tempo. In questo modo il prodotto può essere distribuito ai clienti in uno stato funzionante in tempi relativamente brevi, per poi migliorarlo ed estenderlo in base ai *feedback* ricevuti dal cliente;

* versionamento: la codebase[g] [5] è organizzata su più repository depositate su GitHub per il controllo della versione.

1.1.2 Strumenti utilizzati in azienda

Per l'esecuzione dei suoi processi interni, l'azienda si avvale dell'utilizzo dei seguenti strumenti:

* Slack: è un applicazione multipiattaforma di messaggistica istantanea [34];

* Gmail: è un servizio di posta elettronica [15];

* Microsoft Teams: è una piattaforma di comunicazione utilizzata per effettuare videochiamate [35];

* Jira: è un software utilizzato per la gestione dei progetti secondo la metodologia agile, inoltre è un issue tracking system[g] [17] [18];

* GitHub: è un servizio di hosting di repository Git [14];

* Circle CI: è una piattaforma di Continuous Integration (CI)[g] [7] e Continuous Delivery (CD)[g] [6] [4];

* Figma: è uno strumento di prototipazione utilizzato per creare *mockup*, ossia prototipi di interfacce [11].

1.2 Il progetto di stage

1.2.1 Dominio applicativo

Il progetto di stage proposto riguarda lo sviluppo lato *front-end* di alcune funzionalità per il miglioramento dell'esperienza utente durante l'utilizzo della videoispezione da parte degli utenti, e di inserire alcune funzionalità mancanti. Le funzionalità mancanti da implementare non erano state totalmente chiarite inizialmente e sono state rivalutate successivamente con il tutor aziendale, senza comunque comportare modifiche sostanziali al progetto di stage iniziale.

La videoispezione consiste in un'ispezione da remoto di uno stabile da parte di un agente assicurativo, in modo da poterlo valutare senza doversi recare fisicamente sul luogo e fornire al suo proprietario il preventivo per una polizza assicurativa.

La piattaforma RiskApp offre ai suoi clienti la possibilità di avviare una videoispezione direttamente dalla web app[g] [40] e di effettuarla condividendo con chi di dovere, tra cui il proprietario dello stabile, un link a una pagina apposita per collegarsi alla videochiamata come ospite.

1.2.2 Piano di lavoro

Obiettivi

In Tabella 1.1 vengono riportati gli obiettivi da raggiungere. Gli obiettivi previsti dal piano di lavoro iniziale sono stati integrati, durante il corso dello stage, con due ulteriori obiettivi, O05 e O06.

Tabella 1.1: Obiettivi

Codice	Descrizione
O01	Visualizzazione guidata della registrazione dei video durante le videoispezioni
O02	Visualizzazione guidata della registrazione delle immagini durante le videoispezioni
O03	Invio email live agli utenti guest durante le videoispezioni
O04	Visualizzazione guidata della creazione delle registrazioni per la consultazione
O05	Inserimento di azioni mancanti e miglioramento di azioni durante le videoispezioni
O06	Stesura documento manuale dello sviluppatore
D01	Inserimento di azioni mancanti nella visualizzazione di immagini e video delle videoispezioni
F01	Devono essere presenti dei test automatici a livello di *front-end*

Piano delle attività

La pianificazione, in termini di quantità di ore di lavoro, è distribuita come riportato in Tabella 1.2. La prima colonna fa riferimento alla quantità di ore per attività previste dal piano di lavoro iniziale, mentre la seconda fa riferimento alla quantità di ore aggiornata in corso di stage, in seguito all'aggiunta, nelle fasi preliminari dello stage, dei due ulteriori obiettivi riportati precedentemente.

Tabella 1.2: Ripartizione ore lavorative

Durata	Durata rivista	Descrizione dell'attività
72	**72**	**Formazione sulle tecnologie**
40	40	Formazione su ReactJS
32	32	Ricerca e studio di eventuali librerie utili
40	**64**	**Analisi e pianificazione**
152	**120**	**Sviluppo**
40	24	Implementazione della visualizzazione guidata della registrazione dei video durante le videoispezioni
40	16	Implementazione della visualizzazione guidata della registrazione delle immagini durante le videoispezioni
16	8	Implementazione dell'invio di email d'invito agli utenti guest durante le videoispezioni
24	16	Implementazione della visualizzazione guidata della creazione delle registrazioni per la consultazione

Tabella 1.2: Ripartizione ore lavorative

Durata	Durata rivista	Descrizione dell'attività
32	16	Implementazione dell'inserimento di azioni mancanti nella visualizzazione di immagini e video delle videoispezioni
-	40	Implementazione dell'inserimento di azioni mancanti e miglioramento di azioni durante le videoispezioni
56	**64**	**Collaudo finale**
32	32	Stesura dei test
16	16	Collaudo e verifica
8	16	Stesura documentazione finale
320	**320**	**Totale ore**

1.3 Organizzazione del testo

Il secondo capitolo descrive l'analisi dei requisiti. Presenta gli attori coinvolti nell'interazione con le funzionalità implementate e descrive i casi d'uso individuati. Vengono riportati i requisiti i quali sono classificati in base alla tipologia e all'importanza.

Il terzo capitolo descrive la fase di progettazione, l'architettura della piattaforma RiskApp e l'organizzazione dei file.

Il quarto capitolo descrive la fase di codifica, vengono mostrate alcune schermate del prodotto e sono riportate le funzioni più rilevanti, inoltre descrive la fase di verifica e di validazione.

Il quinto capitolo riporta il consuntivo delle attività svolte, gli obiettivi raggiunti, i requisiti soddisfatti e una breve valutazione personale del lavoro complessivo.

Capitolo 2

Analisi dei requisiti

Il presente capitolo descrive l'analisi dei requisiti. Presenta gli attori coinvolti nell'interazione con le funzionalità implementate e descrive i casi d'uso individuati. Vengono riportati i requisiti i quali sono classificati in base alla tipologia e all'importanza.

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo Unified Modeling Language (UML)[g] [38] dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.

2.1 Attori

Come si può osservare in Figura 2.1, l'unico attore che partecipa ai casi d'uso descritti nella Sezione 2.2 è l'utente, ossia colui che utilizza la piattaforma RiskApp e dispone dei suoi servizi.

Figura 2.1: Attori

2.2 Casi d'uso

La Figura 2.2 mostra il caso d'uso principale utilizzando la notazione UML.

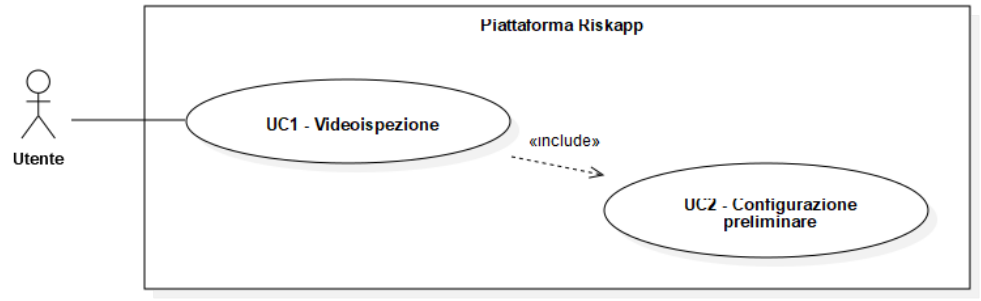


Figura 2.2: Use Case: UC0 - Scenario principale

UC1 - Videoispezione

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp.

Descrizione: L'utente si collega a una videoispezione.

Postcondizioni: L'utente si è collegato a una videoispezione.

La Figura 2.3 mostra il caso d'uso descritto utilizzando la notazione UML.

UC1.1 - Attivazione modalità schermo intero

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente cliccando su un pulsante apposito visualizza la schermata di videoispezione a schermo intero.

Postcondizioni: L'utente ha visualizzato la schermata di videoispezione a schermo intero.

UC1.2 - Visualizzazione messaggi

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente visualizza i messaggi inviati e ricevuti.

Postcondizioni: L'utente ha visualizzato i messaggi che ha inviato e i messaggi che ha ricevuto.

La Figura 2.4 mostra il caso d'uso descritto utilizzando la notazione UML.

UC1.2.1 - Visualizzazione singolo messaggio

Attori Principali: Utente

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

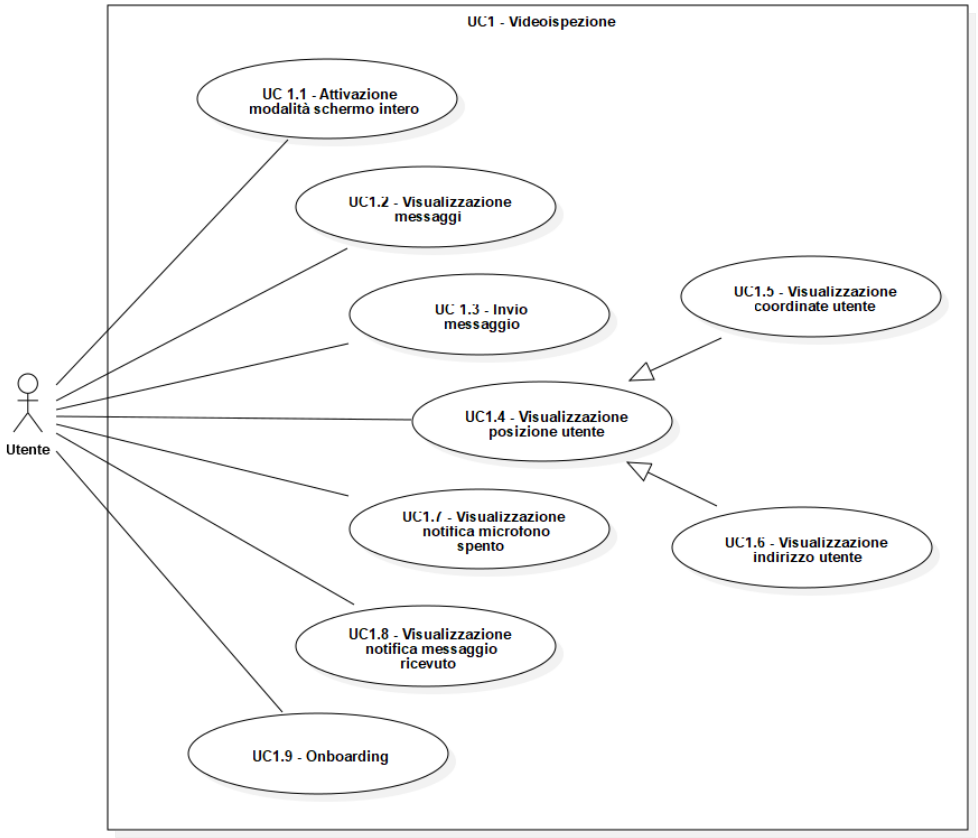


Figura 2.3: Use case: UC1 - Videoispezione

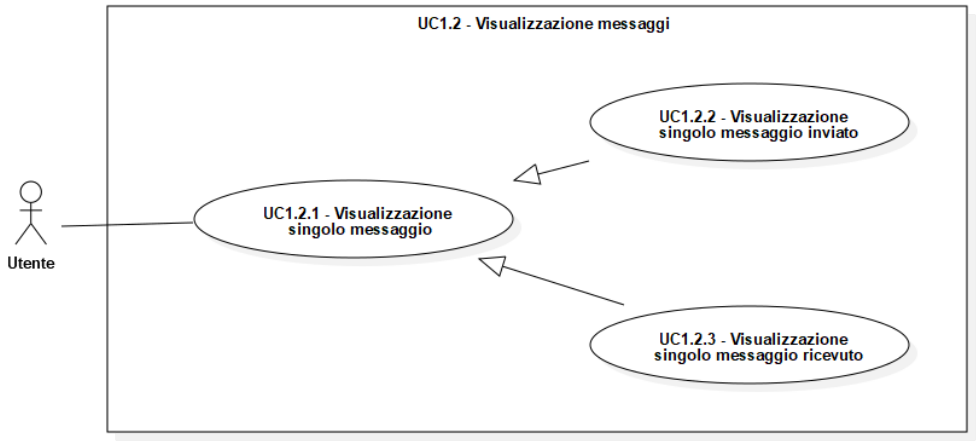


Figura 2.4: Use Case: UC1.2 - Visualizzazione messaggi

Descrizione: L'utente visualizza un singolo messaggio.

Postcondizioni: L'utente ha visualizzato un singolo messaggio.

La Figura 2.5 mostra il caso d'uso descritto utilizzando la notazione UML.

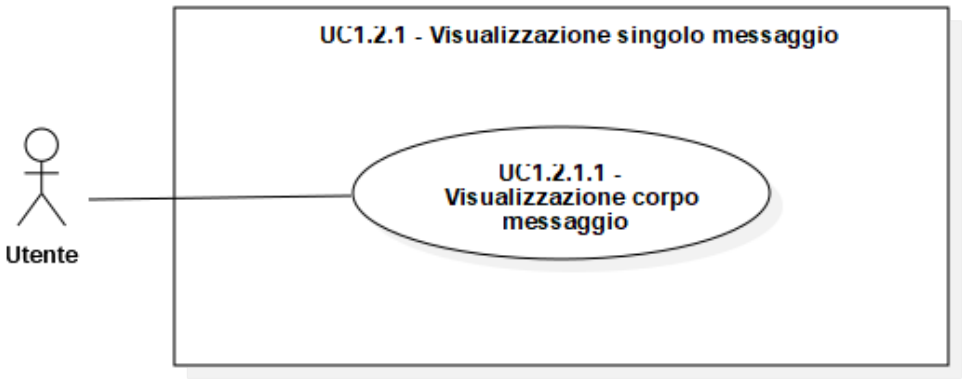


Figura 2.5: Use Case: UC1.2.1 - Visualizzazione singolo messaggio

UC1.2.1.1 - Visualizzazione corpo messaggio

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione. L'utente ha scelto di visualizzare un singolo messaggio.

Descrizione: L'utente visualizza il corpo del messaggio.

Postcondizioni: L'utente ha visualizzato il corpo del messaggio.

UC1.2.2 - Visualizzazione singolo messaggio inviato

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente visualizza un singolo messaggio inviato.

Postcondizioni: L'utente ha visualizzato un singolo messaggio inviato.

La Figura 2.6 mostra il caso d'uso descritto utilizzando la notazione UML.

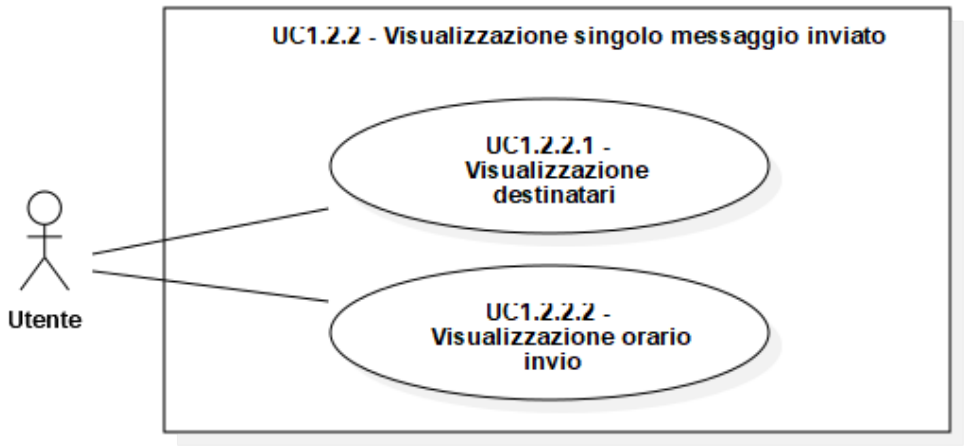


Figura 2.6: Use Case: UC1.2.2 - Visualizzazione singolo messaggio inviato

UC1.2.2.1 - Visualizzazione destinatari

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione. L'utente ha scelto di visualizzare un singolo messaggio inviato.

Descrizione: L'utente visualizza i destinatari del messaggio inviato.

Postcondizioni: L'utente ha visualizzato i destinatari del messaggio inviato.

UC1.2.2.2 - Visualizzazione orario invio

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione. L'utente ha scelto di visualizzare un singolo messaggio inviato.

Descrizione: L'utente visualizza l'orario di invio del messaggio ricevuto.

Postcondizioni: L'utente ha visualizzato l'orario di invio del messaggio ricevuto.

UC1.2.3 - Visualizzazione singolo messaggio ricevuto

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente visualizza un singolo messaggio ricevuto.

Postcondizioni: L'utente ha visualizzato un singolo messaggio ricevuto.

La Figura 2.7 mostra il caso d'uso descritto utilizzando la notazione UML.

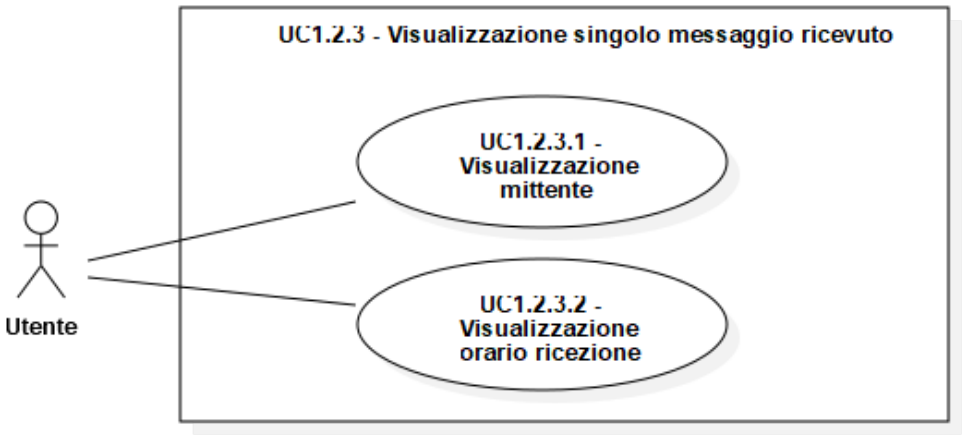


Figura 2.7: Use Case: UC1.2.3 - Visualizzazione singolo messaggio ricevuto

UC1.2.3.1 - Visualizzazione mittente

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione. L'utente ha scelto di visualizzare un singolo messaggio ricevuto.

Descrizione: L'utente visualizza il mittente del messaggio ricevuto.

Postcondizioni: L'utente ha visualizzato il mittente del messaggio ricevuto.

UC1.2.3.2 - Visualizzazione orario ricezione

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione. L'utente ha scelto di visualizzare un singolo messaggio ricevuto.

Descrizione: L'utente visualizza l'orario di ricezione del messaggio ricevuto.

Postcondizioni: L'utente ha visualizzato l'orario di ricezione del messaggio ricevuto.

UC1.3 - Invio messaggio

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente:

* sceglie uno o più destinatari tra gli altri utenti collegati alla videoispezione;
* inserisce il corpo del messaggio;
* conferma l'invio.

Postcondizioni: L'utente ha inviato un messaggio a uno o più destinatari scelti tra gli altri utenti collegati alla videoispezione. I destinatari hanno ricevuto il messaggio inviato.

La Figura 2.8 mostra il caso d'uso descritto utilizzando la notazione UML.

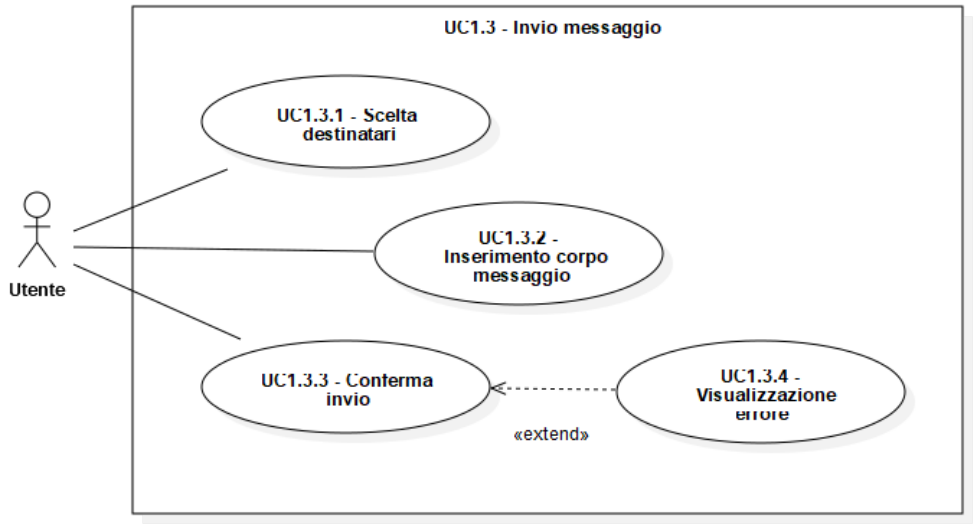


Figura 2.8: Use case: UC1.3 - Invio messaggio

UC1.3.1 - Scelta destinatari

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp e si è collegato a una videoispezione; inoltre ha iniziato la procedura per l'invio di un messaggio.

Descrizione: L'utente sceglie uno o più destinatari tra gli altri utenti collegati alla videoispezione.

Postcondizioni: L'utente ha scelto uno o più destinatari del messaggio da inviare.

UC1.3.2 - Inserimento corpo messaggio

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp e si è collegato a una videoispezione; inoltre ha iniziato la procedura per l'invio di un messaggio.

Descrizione: L'utente inserisce il corpo del messaggio.

Postcondizioni: L'utente ha inserito il corpo del messaggio da inviare.

UC1.3.3 - Conferma invio

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp e si è collegato a una videoispezione; inoltre ha iniziato la procedura per l'invio di un messaggio.

Descrizione: L'utente conferma l'invio del messaggio.

Postcondizioni: L'utente ha confermato l'invio del messaggio. Il messaggio è stato inviato ai relativi destinatari.

Scenario Alternativo: Nel caso in cui l'utente non abbia scelto un destinatario o non abbia inserito il corpo del messaggio, viene visualizzato un messaggio di errore (UC1.3.4).

UC1.3.4 - Visualizzazione errore

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp e si è collegato a una videoispezione; inoltre ha iniziato la procedura per l'invio di un messaggio.

Descrizione: L'utente visualizza un errore.

Postcondizioni: L'utente ha visualizzato un errore e l'invio del messaggio è stato annullato.

UC1.4 - Visualizzazione posizione utente

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente:

* seleziona un utente tra gli altri collegati alla videoispezione;
* visualizza la posizione dell'utente selezionato.

Postcondizioni: L'utente ha visualizzato la posizione dell'utente selezionato.

UC1.5 - Visualizzazione coordinate utente

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente:

* seleziona un utente tra gli altri collegati alla videoispezione;
* visualizza su una mappa la posizione, espressa in coordinate geografiche cartesiane, dell'utente selezionato.

Postcondizioni: L'utente ha visualizzato la posizione, espressa in coordinate geografiche cartesiane, dell'utente selezionato.

UC1.6 - Visualizzazione indirizzo utente

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente:

* seleziona un utente tra gli altri collegati alla videoispezione;
* visualizza la posizione, espressa come indirizzo, dell'utente selezionato.

Postcondizioni: L'utente ha visualizzato la posizione, espressa come indirizzo, dell'utente selezionato.

UC1.7 - Visualizzazione notifica microfono spento

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente:

* sta parlando ma ha il microfono spento;
* visualizza una notifica che lo informa del fatto che sta parlando ma ha il microfono spento.

Postcondizioni: L'utente ha visualizzato una notifica. L'utente è stato informato del fatto che sta parlando ma ha il microfono spento.

UC1.8 - Visualizzazione notifica messaggio ricevuto

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente:

* riceve un messaggio;
* visualizza una notifica che lo informa del fatto che ha ricevuto un messaggio.

Postcondizioni: L'utente ha visualizzato una notifica. L'utente è stato informato del fatto che ha ricevuto un messaggio.

UC1.9 - Onboarding

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente si è collegato a una videoispezione.

Descrizione: L'utente viene guidato all'utilizzo delle funzionalità fruibili durante la videoispezione mediante una procedura di *onboarding*, cioè un processo che gli permetta

di comprendere il vero valore del servizio che sta utilizzando.

Postcondizioni: L'utente è stato guidato all'utilizzo delle funzionalità fruibili durante la videoispezione mediante una procedura di *onboarding*. L'utente ha appreso come utilizzare le funzionalità fruibili durante la videoispezione.

UC2 - Configurazione preliminare

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp.

Descrizione: L'utente configura i parametri necessari per collegarsi a una videoispezione.

Postcondizioni: L'utente ha configurato i parametri necessari per collegarsi a una videoispezione.

La Figura 2.9 mostra il caso d'uso descritto utilizzando la notazione UML.

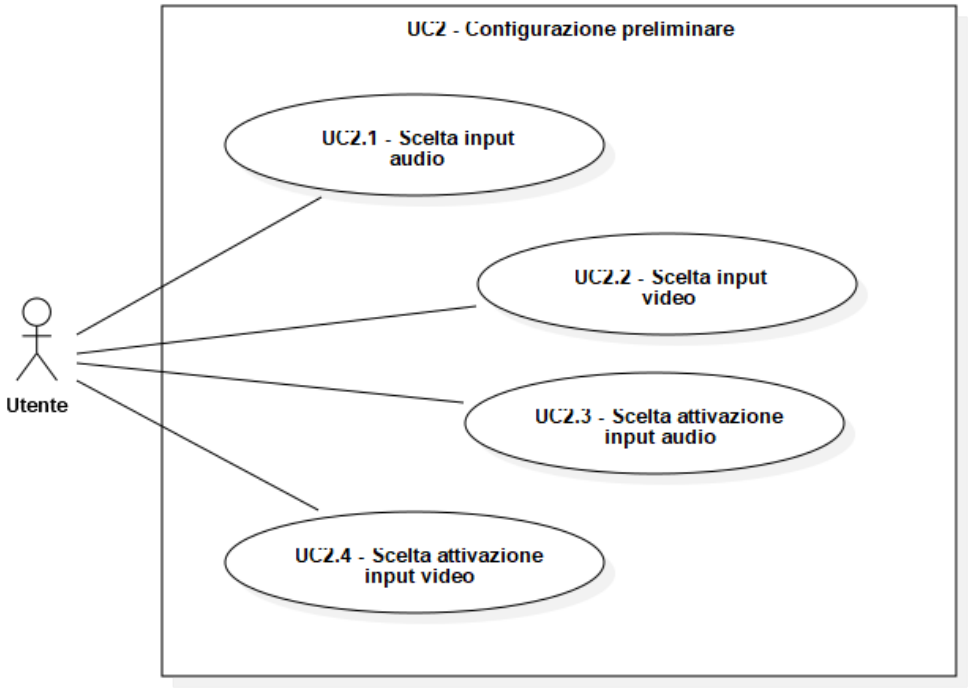


Figura 2.9: Use Case: UC2 - Configurazione preliminare

UC2.1 - Scelta input audio

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente ha iniziato la procedura di configurazione dei parametri necessari per collegarsi a una videoispezione.

Descrizione: L'utente sceglie un dispositivo di input audio tra quelli disponibili da

utilizzare durante una videoispezione.

Postcondizioni: L'utente ha scelto un dispositivo di input audio.

UC2.2 - Scelta input video

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente ha iniziato la procedura di configurazione dei parametri necessari per collegarsi a una videoispezione.

Descrizione: L'utente sceglie un dispositivo di input video tra quelli disponibili da utilizzare durante una videoispezione.

Postcondizioni: L'utente ha scelto un dispositivo di input video.

UC2.3 - Scelta attivazione input audio

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente ha iniziato la procedura di configurazione dei parametri necessari per collegarsi a una videoispezione.

Descrizione: L'utente sceglie se rendere attivo o meno il dispositivo di input audio nel momento in cui si collega a una videoispezione.

Postcondizioni: L'utente ha scelto se rendere attivo o meno il dispositivo di input audio.

UC2.3 - Scelta attivazione input video

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso alla piattaforma RiskApp. L'utente ha iniziato la procedura di configurazione dei parametri necessari per collegarsi a una videoispezione.

Descrizione: L'utente sceglie se rendere attivo o meno il dispositivo di input video nel momento in cui si collega a una videoispezione.

Postcondizioni: L'utente ha scelto se rendere attivo o meno il dispositivo di input video.

2.3 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e dei casi d'uso, è stata stilata la tabella che traccia i requisiti in rapporto ai casi d'uso.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.
Il codice dei requisiti è così strutturato:

[Tipologia][Importanza]-[Codice]

dove:

- **Tipologia**: può essere:
  - **F**: indica un requisito funzionale;
  - **Q**: indica un requisito qualitativo;
  - **V**: indica un requisito di vincolo;
- **Importanza**: può essere:
  - **N**: indica un requisito obbligatorio;
  - **D**: indica un requisito desiderabile;
  - **Z**: indica un requisito opzionale;
- **Codice**: codice identificativo univoco per ogni requisito.

2.3.1 Requisiti funzionali

Nella Tabella 2.1 sono elencati i requisiti funzionali individuati durante l'attività di analisi a partire dai casi d'uso.

Tabella 2.1: Tracciamento dei requisti funzionali

Requisito	Descrizione	Fonte
RFN-1	L'utente deve poter visualizzare la schermata a schermo intero	UC1.1
RFN-2	L'utente deve poter visualizzare i messaggi che ha inviato e che ha ricevuto	UC1.2
RFN-3	L'utente deve poter inviare un messaggio a un altro utente collegato alla videoispezione	UC1.3
RFN-4	L'utente deve essere avvisato se sta parlando ma ha il microfono spento	UC1.7
RFN-5	L'utente deve essere avvisato se riceve un messaggio	UC1.8
RFN-6	L'utente deve essere guidato all'utilizzo della videoispezione attraverso una procedura di *onboarding*	UC1.9
RFN-7	L'utente deve scegliere il dispositivo di input audio da utilizzare durante la videoispezione prima di collegarsi alla stessa	UC2.1
RFN-8	L'utente deve scegliere il dispositivo di input video da utilizzare durante la videoispezione prima di collegarsi alla stessa	UC2.2
RFN-9	L'utente deve scegliere se rendere attivo o meno il dispositivo di input audio prima di collegarsi alla videoispezione	UC2.3
RFN-10	L'utente deve scegliere se rendere attivo o meno il dispositivo di input audio prima di collegarsi alla videoispezione	UC2.3

2.3.2 Requisiti qualitativi

Nella Tabella 2.2 sono elencati i requisiti qualitativi individuati durante l'attività di analisi a partire dalle discussioni con il proponente.

Tabella 2.2: Ttracciamento dei requisiti qualitativi

Requisito	Descrizione	Fonte
RQN-1	Le funzionalità implementate devono essere tradotte sia in italiano che in inglese	Proponente
RQN-2	Stesura documento di analisi dei requisiti	Proponente
RQN-3	Stesura manuale dello sviluppatore	Proponente
RQZ-1	Copertura test dell'80% nelle funzionalità implementate	Proponente

2.3.3 Requisiti di vincolo

Nella Tabella 2.3 sono elencati i requisiti di vincolo individuati durante l'attività di analisi.

Tabella 2.3: Tracciamento dei requisiti di vincolo

Requisito	Descrizione	Fonte
RVN-1	Le funzionalità implementate devono funzionare con i browser: Google Chrome (v. 97.0.4692), Mozilla Firefox (v. 95.0) e Microsoft Edge (v. 97.0.1072.69)	ECMAScript[g] [10]

Capitolo 3

Progettazione

Il presente capitolo descrive la fase di progettazione, l'architettura della piattaforma RiskApp e l'organizzazione dei file.

3.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati. Si precisa che, dove non indicato diversamente, la scelta di utilizzare una certa tecnologia o un certo strumento è stata dettata da vincoli o consuetudini aziendali.

3.1.1 Tecnologie utilizzate

JavaScript

JavaScript (si veda la Figura 3.1) è un linguaggio di programmazione comunemente utilizzato per lo sviluppo Web lato client. Allo stato attuale risulta essere uno dei linguaggi più popolari a livello globale. L'enorme diffusione di JavaScript è dovuta principalmente al fiorire di numerose librerie nate allo scopo di semplificare la programmazione sul browser, ma anche alla nascita di *framework* lato server e nel mondo mobile che lo supportano come linguaggio principale [19].

Si è reso necessario utilizzare questa tecnologia in quanto React è un *framework* JavaScript, e il suo utilizzo era mandatorio. Non si è utilizzato TypeScript, un'estensione di JavaScript, perché non era necessaria la tipizzazione stretta che il linguaggio offre, e perché il *front-end* della piattaforma RiskApp è interamente scritto in JavaScript.

Figura 3.1: Logo del linguaggio JavaScript

CSS3

CSS3 (si veda la Figura 3.2) è la più recente versione di CSS, un linguaggio dichiarativo per la formattazione di pagine Web [8].

Figura 3.2: Logo del linguaggio CSS3

React

React (si veda la Figura 3.3) è un *framework* JavaScript per la creazione di interfacce utente. Si basa su un approccio *component-based*, dove un componente è una particolare funzione JavaScript che descrive la logica e il comportamento di una parte dell'interfaccia grafica, un componente appunto [25].

L'approccio allo sviluppo di componenti è dichiarativo, e per cui semplice e veloce. Si possono adottare due tecniche per descrivere un componente [26]:

* sviluppo a componenti: descrive i componenti come classi;
* sviluppo funzionale: la versione 16.7.0 ha introdotto la possibilità di utilizzare componenti funzionali, ossia componenti che sono descritti come funzioni JavaScript. Ad ogni modo lo sviluppo funzionale si offre esclusivamente come alternativa allo sviluppo a componenti, il quale al momento non è destinato a diventare deprecato.

Il *front-end* di RiskApp è interamente sviluppato a componenti.

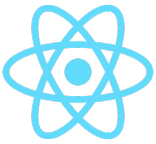


Figura 3.3: Logo del framework React

Node.js

Node.js (si veda la Figura 3.4) è un *framework open-source* che permette di eseguire codice JavaScript lato server in maniera asincrona [20].

Figura 3.4: Logo del framework Node.js

AntD

AntD (si veda la Figura 3.5) è una libreria che fornisce componenti dell'interfaccia grafica già preparati, personalizzabili e facili da utilizzare. È integrabile all'interno di un progetto React installando le librerie antd e @ant-design/icons tramite npm [3].

Un competitor di AntD più popolare è sicuramente Bootstrap, tuttavia si è scelto di utilizzare AntD in quanto è molto utilizzato per la modellazione dell'interfaccia grafica della piattaforma RiskApp. Un altro motivo è che i suoi componenti sono di altissima qualità, sia a livello grafico che di accessibilità, e inoltre, avendo avuto già esperienza con Bootstrap, si è deciso di confrontarsi e formarsi su una tecnologia precedentemente sconosciuta.

Figura 3.5: Logo del framework AntD

3.1.2 Strumenti utilizzati

WebStorm

WebStorm (si veda la Figura 3.6) è un Integrated Development Enviroment (IDE)[g] per lo sviluppo in ambiente JavaScript. È ricco di funzioni che aiutano lo sviluppatore durante la codifica, come l'auto-completamento, l'auto-correzione, controllo della sintassi e adeguamento alle varie versioni del linguaggio, integrazione con Node.js, refactoring e integrazione con sistemi di versionamento del codice [42].

La scelta di utilizzare questo IDE è legata al fatto di aver avuto precedentemente esperienza con questo strumento, ed è stata meramente personale, infatti l'azienda non ha imposto vincoli sullo strumento da utilizzare per la codifica, il debugging e il testing.

Figura 3.6: Logo del software WebStorm

Slack

Slack (si veda la Figura 3.7) è un applicazione multipiattaforma per la messaggistica istantanea tra membri di un gruppo di lavoro. Una delle funzioni di Slack è la possibilità di organizzare la comunicazione del team attraverso canali specifici, canali che possono essere accessibili a tutto il team o solo ad alcuni membri. È possibile inoltre comunicare con il team anche attraverso chat individuali private o chat con due o più membri [34].

Questo software è stato utilizzato sia per scambiare velocemente documenti, che per comunicare con il tutor aziendale da remoto.

Figura 3.7: Logo del software Slack

Figma

Figma (si veda la Figura 3.8) è un software per la progettazione di *User Interface* (UI). Permette infatti di realizzare prototipi delle interfacce, altresì detti *mockup*, che permettono di illustrare il risultato finale che si desidera ottenere [11].

Questo strumento è servito a mostrare al tutor aziendale e di concordare con lui l'interfaccia da realizzare.

Figura 3.8: Logo del software Figma

Git

Git (si veda la Figura 3.9) è un sistema di controllo del versionamento gratuito e a licenza libera [13].

Nello sviluppo della piattaforma RiskApp viene utilizzata la tecnica *Git Feature Branch Workflow*, la quale predilige la creazione di un nuovo *branch* ogniqualvolta si debba inserire una nuova funzionalità nella web app. Nel nuovo *branch* si sviluppa la funzionalità e al termine si effettua il merge[g] nel *branch* principale.

Figura 3.9: Logo del sistema di controllo di versione Git

GitHub

GitHub (si veda la Figura 3.10) è una piattaforma di hosting per repository git. Fornisce agli sviluppatori strumenti per migliorare e mantenere il codice come [14]:

* features utilizzabili da linea di comando;
* gestione delle pull request e code review;
* strumenti per l'issue tracking.

La codebase della piattaforma RiskApp è suddivisa in varie repository su GitHub.

Figura 3.10: Logo della piattaforma GitHub

3.2 Architettura preesistente

L'organizzazione del codice *front-end* è modulare, cioè è suddiviso per funzionalità, inoltre è distribuito su più repository GitHub. Per ciascuna funzionalità generalmente si distinguono cinque sottocartelle:

* `actions`: contiene un file JavaScript `index.js` in cui sono definite le *action* di Redux (si veda la Sottosezione 3.2.1);
* `assets`: contiene tutto ciò che non è codice, come ad esempio le immagini;
* `components`: contiene i file JavaScript, in formato `.jsx`, che definiscono i componenti React;
* `reducers`: contiene uno o più file JavaScript con i *reducer* di Redux (si veda la Sottosezione 3.2.1);
* `style`: contiene i fogli di stile per la formattazione dei componenti.

Per quanto riguarda il repository relativo alla piattaforma RiskApp, e quindi alla funzionalità di videoispezione, i file sono organizzati come mostrato nello Snippet di codice 3.1:

```
-src
|-...
|-locale
||-translations.json
|-...
|-inspection
||-index.js
||-actions
|||-index.js
||-assets
|||-...
||-components
|||-twilio
||||-JoinCall.jsx
||||-Participant.jsx
||||-VideocallInspectionContainer.jsx
||||-VideoCallMap.jsx
||||-VideoRoom.jsx
||-reducers
|||-index.js
|||-inspection_reducer.js
||-style
|||-inspector.css
|||-inspector-page.css
|-...
```

Snippet di codice 3.1: Organizzazione dei file relativi alla videoispezione

I file preesistenti che sono stati modificati, nel corso dello stage, durante la fase di codifica sono i seguenti:

* `translations.json`: è un file `.json` che contiene le traduzioni in italiano e in inglese del testo presente nella web app;
* `JoinCall.jsx`: è il componente per la scelta degli input audio e video prima di avviare o partecipare alla videochiamata;
* `VideocallInspectionContainer.jsx`: è il componente di livello superiore;
* `VideoCallMap.jsx`: è il componente che mostra la mappa con la posizione del partecipante remoto;
* `VideoRoom.jsx`: è il componente che fornisce lo streaming audio e video e i vari controlli per la gestione della videoispezione;
* `inspector.css`: è il foglio di stile per la formattazione dei componenti relativi alla videoispezione.

I file aggiunti sono invece:

* `VideoCallChat.jsx`: è il componente che mostra i messaggi inviati e ricevuti e permette l'invio di un messaggio.

3.2.1 Architettura di un'applicazione React con Redux

Redux è un contenitore di stato prevedibile per app JavaScript [28]. Fornisce un modo semplice per centralizzare lo stato e la logica di una web app: per questo è ideale nella programmazione di single-page application[g] [33].

Redux si basa sul concetto di avere un unico stato, rappresentato da un oggetto JSON e conservato in uno *store*. Questo stato può mutare solo in seguito ad azioni, ma non direttamente; la modifica avviene tramite l'invocazione di una funzione pura denominata *reducer*. Redux si basa su tre principi fondamentali [30]:

* *single source of truth*: lo stato è l'unica fonte di verità a cui fa riferimento l'interfaccia utente;
* *state is read-only*: lo stato è in sola lettura e può mutare solo con un intento ben definito, in pratica a partire da una azione e attraverso un *reducer*;
* *changes are made with pure functions*: i cambiamenti allo stato avvengono solo attraverso funzioni pure, cioè una funzione che dato un certo input restituisce sempre lo stesso output senza effetti collaterali. Questo rende le funzioni facilmente prevedibili e testabili, e vengono utilizzate per gestire i *reducer*.

Questi tre principi servono a garantire un corretto rispetto dei principi di coesione e accoppiamento.

In Redux si celano diversi termini e concetti importanti [29]:

* *action*: è un evento che descrive qualcosa che è accaduto nell'applicazione. Più precisamente è un oggetto JavaScript che possiede un campo `type`, e che può avere altri campi con informazioni aggiuntive, relative a cos'è successo. Per convenzione queste informazioni vengono memorizzate in un campo chiamato `payload`;

- *state*: è un oggetto che contiene valori utilizzabili durante il rendering, e modificabili all'occorrenza attraverso funzioni specifiche per l'aggiornamento degli elementi del Document Object Model (DOM)[g] [9] legati al nuovo stato;
- *reducer*: è una funzione che, ricevuto lo stato corrente e una *action*, aggiorna lo stato in maniera opportuna e lo ritorna. La firma di un *reducer* è quindi `(state, action) => newState`;
- *store*: è l'oggetto JavaScript in cui risiede lo stato corrente dell'applicazione. Viene costruito passando come parametro un oggetto contenente i *reducer*, e possiede un metodo chiamato `getState()` che ritorna lo stato corrente.

La Figura 3.11 mostra come questi concetti siano correlati tra loro.

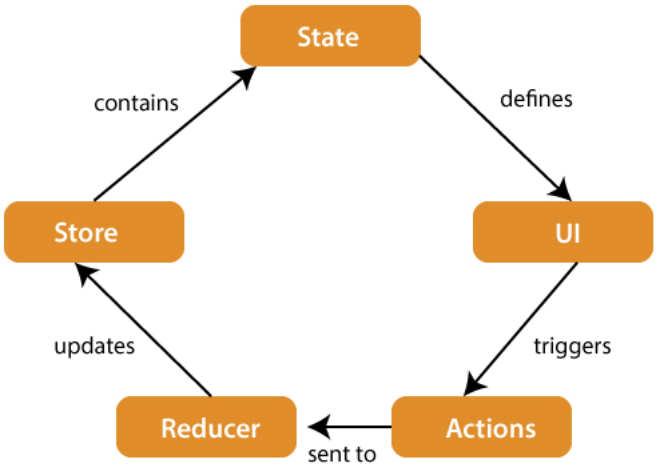


Figura 3.11: Architettura di Redux [27]

L'utente, attraverso l'interfaccia grafica, o un evento interno dell'applicazione, scatenano una *action* che memorizza il tipo di evento (ad esempio il cambiamento di valore di un *checkbox*) e il carico pagante relativo a quell'evento (ad esempio il valore corrente del *checkbox*). La *action* viene successivamente passata a un *reducer* che, attingendo allo *state* corrente dell'applicazione memorizzato nello *store*, modifica lo stato. Il cambiamento dello stato ha conseguenze anche sull'interfaccia grafica, che viene aggiornata.

3.2.2 Protocollo per la comunicazione audio e video real-time

Per gestire la comunicazione audio e video in tempo reale su connessioni *peer-to-peer*, ovvero connessioni in cui ogni nodo può fungere sia da client che da server, si utilizza WebRTC [41].

Acronimo di Web Real-Time Communication, è un insieme di protocolli e Application Program Interface (API)[g], originariamente sviluppati da Google, che consentono ai browser di richiedere informazioni in tempo reale dai browser di altri utenti, consentendo comunicazioni *peer-to-peer* e di gruppo in tempo reale tra cui voce, video, chat, trasferimento di file e condivisione dello schermo [41].

A questo proposito, Twilio Video è una piattaforma di comunicazione in tempo reale basata su WebRTC che fornisce agli sviluppatori la possibilità di essere subito operativi nell'implementazione della propria applicazione di videoconferenza, senza preoccuparsi dell'infrastruttura server necessaria e della loro gestione. Inoltre offre funzionalità di alto livello relative alla segnalazione, gestione dell'accesso degli utenti, elaborazione dei contenuti multimediali e distribuzione dei contenuti multimediali per consentire comunicazioni in tempo reale [37].

Un'applicazione Twilio Video richiede sia un componente *front-end* che un componente *back-end*:

* il lato server è necessario per generare i token di accesso per i partecipanti e per gestire le impostazioni della Room o le Recordings;
* il lato client è l'applicazione che si connette a Twilio Cloud e con cui gli utenti interagiscono.

La Figura 3.12 rappresenta visivamente quanto appena descritto.

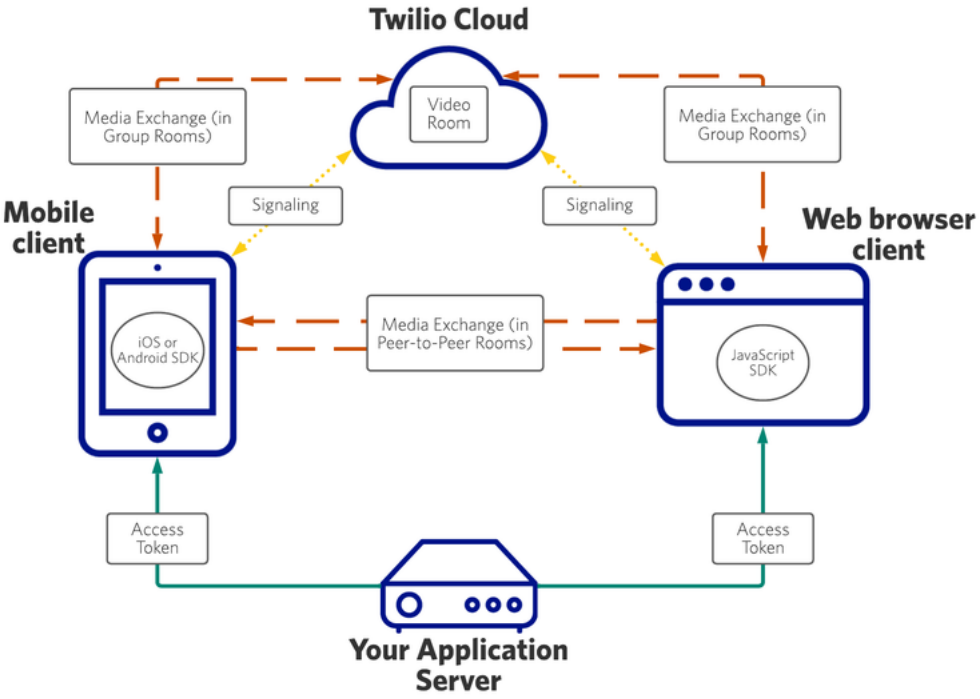


Figura 3.12: Architettura di Twilio Video [37]

In particolare le API lato client permettono la gestione delle seguenti risorse:

* Room: rappresenta lo spazio virtuale a cui gli utenti sono collegati dal punto di vista del client;
* Participant: rappresenta un utente collegato dal punto di vista del client. Esistono due tipi di Participant:

  - LocalParticipant: è il Participant in "questo" client, cioè il client in cui è in esecuzione il codice;
  - RemoteParticipant: è un Participant del "resto" dei client remoti;

* Track: è un flusso di byte che contiene i dati generati da una sorgente multimediale, visto dal punto di vista del client. Esistono tre tipi di Track:

  - AudioTrack: rappresenta un flusso di byte generato da una sorgente audio come un microfono;
  - VideoTrack: rappresenta un flusso di byte generato da una sorgente video come una webcam o uno schermo;
  - DataTrack: rappresenta un flusso di byte (o caratteri) che vengono generati dall'applicazione. In altre parole, gli sviluppatori possono utilizzare un DataTrack per comunicare dati arbitrari tra i Participant con una latenza molto bassa.

  Inoltre si possono distinguere in:

  - Local Tracks: LocalAudioTrack, LocalVideoTrack e LocalDataTrack sono i flussi di byte generati dal client locale;
  - Remote Tracks: RemoteAudioTrack, RemoteVideoTrack e RemoteDataTrack sono i flussi di byte provenienti da client remoti;

* Track Publication: rappresenta la pubblicazione di una Track nella Room in modo da renderla disponibile a tutti i Participant collegati. Le Track Publication sono classificate in modo speculare rispetto alle Track. [36]

La Figura 3.13 rappresenta visivamente le risorse e le loro dipendenze.

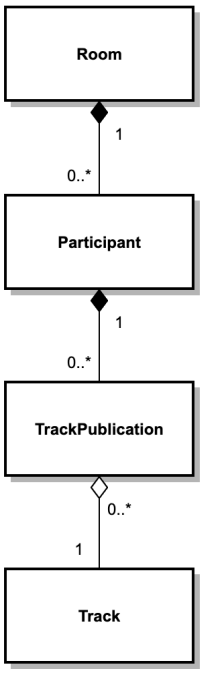


Figura 3.13: Risorse e dipendenze delle API client di Twilio Video [36]

3.3 Scelta dell'onboarding UX pattern

Con il termine *user onboarding* si intende il processo attraverso il quale il cosiddetto *first-time user* non solo viene guidato all'utilizzo del prodotto, ma anche comprende il suo vero valore. Questo è fondamentale sul lungo termine per fidelizzare i clienti e incentivarli a utilizzare il prodotto.

Inizialmente è stata valutata la possibilità di utilizzare il pattern *product tour* il quale consiste, come suggerisce il nome, in una spiegazione step-by-step del prodotto che sia chiara, completa e definitiva [24]. Questo approccio tuttavia ha lo svantaggio di essere una procedura che nella maggior parte dei casi l'utente trova invasiva e inopportuna [1]. Questo comportamento può essere spiegato con il cosiddetto paradosso dell'utente attivo, che si riferisce alla situazione in cui l'utente preferisce utilizzare il software immediatamente piuttosto che informarsi prima su come utilizzarlo. La situazione è paradossale perché si risparmierebbe del tempo nel lungo termine se si dedicasse del tempo inizialmente per imparare [1]. Inoltre si é ritenuto che alcuni comandi siano già di per sé semplici e autoesplicativi, e quindi nella maggior parte dei casi l'utente sa come utilizzarli.

Di conseguenza si è optato per una soluzione che non fosse invasiva. Un altro pattern piuttosto comune sono gli *hotspot* che si presentano a video come dei punti lampeggianti in modo da attirare l'attenzione dell'utente. Passando il cursore sopra gli stessi (o cliccandoli se si utilizza un dispositivo touch screen) compare un pop-up che riporta delle informazioni utili. Si è scelto di utilizzare questo pattern perché, come riportato innanzi, non é invasivo ma é opzionale.

3.4 Scelta del servizio di reverse geocoding

Per ricavare l'indirizzo di un utente a partire dalle sue coordinate geografiche è necessario utilizzare un servizio di reverse geocoding[g].

Per implementare la funzionalità di visualizzazione della posizione dell'utente remoto espressa come indirizzo, si è utilizzato *Nominatim*, il quale è uno strumento, gratuito e a licenza libera, per trovare dati di OpenStreetMap[g] [23] per nome e indirizzo e per generare indirizzi sintetici di punti OpenStreetMap [21].

Per interfacciarsi all'utilizzo di *Nominatim*, esistono delle API apposite che sono invocabili attraverso richieste HyperText Transfer Protocol (HTTP)[g] [16]. Il formato base per effettuare una richiesta di reverse geocoding è mostrato nello Snippet di codice 3.2:

```
https://nominatim.openstreetmap.org/reverse?lat=<value>&lon=<value>&<
    params>
```

Snippet di codice 3.2: Formato base per richieste di reverse geocoding

dove `lat` e `lon` sono parametri della query string[g], e definiscono, rispettivamente, la latitudine e la longitudine. Si accettano inoltre ulteriori parametri opzionali, come ad esempio `format` che permette di scegliere il formato di output della risposta. Nello Snippet di codice 3.3 è riportata, a titolo di esempio, la risposta dell'API alla richiesta `https://nominatim.openstreetmap.org/reverse?format=xml&lat=52.5487429714954&lon=-1.81602098644987&zoom=18&addressdetails=1` [31].

```
<reversegeocode timestamp="Fri, 06 Nov 09 16:33:54 +0000" querystring="
    ...">
<result place_id="1620612" osm_type="node" osm_id="452010817">
```

```
  135, Pilkington Avenue, Wylde Green, City of Birmingham, West Midlands
       (county), B72, United Kingdom
</result>
<addressparts>
  <house_number>135</house_number>
  <road>Pilkington Avenue</road>
  <village>Wylde Green</village>
  <town>Sutton Coldfield</town>
  <city>City of Birmingham</city>
  <county>West Midlands (county)</county>
  <postcode>B72</postcode>
  <country>United Kingdom</country>
  <country_code>gb</country_code>
</addressparts>
</reversegeocode>
```

Snippet di codice 3.3: Esempio di risposta

È bene sottolineare però che il fatto che sia un servizio gratuito e a licenza libera porta con sé anche dei limiti di utilizzo. Infatti è necessario il rispetto dei seguenti requisiti:

* non è possibile utilizzare il servizio in modo massivo (massimo 1 richiesta per secondo);
* si deve fornire un valido HTTP Referer o User-Agent che identifichi l'applicazione;
* si deve chiaramente riportare nell'interfaccia l'attribuzione di utilizzo [22].

Il requisito più stringente è il primo, che impone un limite circa il numero di richieste al secondo. Però si è valutato che fosse un limite ugualmente accettabile, considerato che il recupero delle coordinate, e conseguentemente dell'indirizzo, viene effettuato ogni 30 secondi.

Capitolo 4

Codifica

Il presente capitolo descrive la fase di codifica, vengono mostrate alcune schermate del prodotto e sono riportate le funzioni più rilevanti, inoltre descrive la fase di verifica e di validazione.

4.1 Codifica dei componenti React

La UI di un'applicazione sviluppata in React è composta da componenti, ossia pezzi di codice riutilizzabili e indipendenti. Per lo sviluppo di un componente, si possono seguire due approcci diversi [26]:

- sviluppo a classi;
- sviluppo funzionale.

A titolo di esempio sono riportati due snippet di codice che definiscono due componenti equivalenti dal punto di vista dell'output prodotto ma che differiscono per l'approccio utilizzato. Lo Snippet di codice 4.1 presenta un componente definito secondo lo sviluppo a classi, si tratta cioè di una classe che eredita da `React.Component` i suoi attributi e i suoi metodi, tra cui il metodo `render()` di cui viene fatto l'*override* e che ritorna un oggetto di tipo `ReactNode`. Lo Snippet di codice 4.2, invece, presenta un componente funzionale, ossia una funzione JavaScript che accetta come parametro un oggetto contenente delle *properties* e ritorna sempre un oggetto di tipo `ReactNode`. In entrambi i casi, la struttura dei componenti è definita utilizzando la sintassi JSX[g].

```
1 class Welcome extends React.Component {
2     render() {
3         return <h1>Hello, {this.props.name}</h1>;
4     }
5 }
```

Snippet di codice 4.1: Componente a classi

```
1 function Welcome(props) {
2     return <h1>Hello, {props.name}</h1>;
3 }
```

Snippet di codice 4.2: Componente funzionale

Su richiesta del tutor aziendale, la codifica delle funzionalità è stata fatta utilizzando l'approccio a classi. Il motivo alla base di questa richiesta è che il codice *front-end* della web app è interamente scritto utilizzando componenti a classi.

4.2 Convenzioni seguite

Per quanto concerne la codifica in React e JSX, al fine di rendere il codice prodotto più leggibile, più comprensibile e facilmente manutenibile nel futuro, sono state seguite le seguenti convenzioni:

* convenzioni sui nomi:
  - utilizzo dell'estensione `.jsx` per i file dei componenti React;
  - corrispondenza tra il nome del file e il relativo componente React;
  - utilizzo della notazione PascalCase[g] per i nomi dei file e dei componenti React;
  - utilizzo della notazione camelCase[g] per le istanze di componenti React;
  - utilizzo della notazione camelCase per i nomi delle *properties* di un componente;
* utilizzo delle virgolette doppie per gli attributi dei tag JSX, e delle virgolette singole per il resto del codice JavaScript;
* allineamento dei tag JSX come mostrato nello Snippet di codice 4.3;
* aggiunta di uno spazio prima della chisura di un *self-closing tag*;
* utilizzo di un *self-closing tag* quando lo stesso non presenta nodi figli;
* chiusura di un tag in una nuova linea di codice se presenta più *multi-line properties*.

```
1  // good
2  <Foo
3    superLongParam="bar"
4    anotherSuperLongParam="baz"
5  />
6
7  // if props fit in one line then keep it on the same line
8  <Foo bar="bar" />
9
10 // children get indented normally
11 <Foo
12   superLongParam="bar"
13   anotherSuperLongParam="baz"
14 >
15   <Quux />
16 </Foo>
```

Snippet di codice 4.3: Allineamento dei tag JSX

4.3 Schermata per la visualizzazione della posizione

La Figura 4.1 mostra la schermata per la visualizzazione della posizione "virtuale" di un altro utente collegato alla videoispezione. Per JavaScript, esiste una libreria apposita chiamata Leaflet che permette di importare una mappa interattiva all'interno del proprio progetto, inoltre tramite `npm` è possibile importare facilmente il package `react-leaflet` che fornisce il componente `Map` per la renderizzazione di una mappa. Sulla parte inferiore

a destra della schermata viene inoltre riportata, per rispettare correttamente la *usage policy* [22], l'attribuzione di utilizzo dei servizi di OpenStreetMap.

Oltre alla visualizzazione su mappa, premendo la freccia posta in alto a sinistra è possibile aprire/chiudere una tendina che mostra l'indirizzo. Lo Snippet di codice 4.4 mostra come viene effettuato il suo recupero.

```
1  // If coordinates are set performs reverse geocoding using Nominatim API
       .
2  axios
3      .get(
4          `https://nominatim.openstreetmap.org/reverse?` +
5          `format=json&` +
6          `lat=${this.props.coordinates.lat}&` +
7          `lon=${this.props.coordinates.lng}&` +
8          `accept-language=${this.props.p.t('inspection.map_2') === "Map"
               ? "en" : "it"}`,
9      )
10     .then((response) => {
11       this.setState({
12         address: response.data['display_name']
13       });
14     })
15     .catch(() => {
16       this.setState({
17         address: 'Loading...'
18       });
19     });
```

Snippet di codice 4.4: Recupero dell'indirizzo tramite Nominatim API

Come riportato precedentemente nella Sezione 3.4, si sono utilizzate le API di *Nominatim* per effettuare reverse geocoding. Per recuperare l'indirizzo viene eseguita una richiesta HTTP in cui si specifica nella query string il formato della risposta, la latitudine, la longitudine e la lingua con cui si deve ritornare la risposta. Quest'ultimo parametro varia a seconda delle preferenze impostate dall'utente per quanto riguarda la lingua. Lo stato corrente dell'applicazione relativo alla lingua impostata è salvato nello *store* di Redux (si veda la Sottosezione 3.2.1). Una volta che la *promise*, che in JavaScript corrisponde a un'operazione asincrona, relativa alla richiesta si è risolta con successo, una funzione di *callback*, che prende come parametro la risposta, aggiorna lo stato del componente e imposta l'attributo `address`. L'aggiornamento dello stato del componente provoca conseguentemente il *re-rendering* dello stesso, il quale mostra a video il nuovo indirizzo calcolato.

Figura 4.1: Schermata per la visualizzazione della posizione

4.4 Schermata per la visualizzazione e invio messaggi

Per gestire la visualizzazione dei messaggi inviati e ricevuti e l'invio di un nuovo messaggio, si è deciso di creare un componente trascinabile e ridimensionabile chiamato `VideoCallChat`, mostrato in Figura 4.2.

I messaggi inviati e ricevuti vengono mostrati come una lista, i cui elementi sono disposti in ordine cronologico. Oltre al contenuto, i messaggi inviati sono caratterizzati da uno o più destinatari e dall'orario di invio, mentre quelli ricevuti sono caratterizzati dal mittente e dall'orario di ricezione.

Sulla parte inferiore della finestra, invece, è presente un piccolo form per l'invio di un nuovo messaggio. Ogni *checkbox* fa riferimento a un partecipante remoto, e selezionandolo si decide di renderlo uno dei destinatari del messaggio da mandare. Una volta scelti i destinatari e inserito del testo nell'apposito campo, premendo il tasto di invio, il messaggio sarà inviato. Nel caso in cui nessun destinatario è stato selezionato, o il corpo del messaggio è vuoto, viene mostrato un messaggio di errore esplicativo.

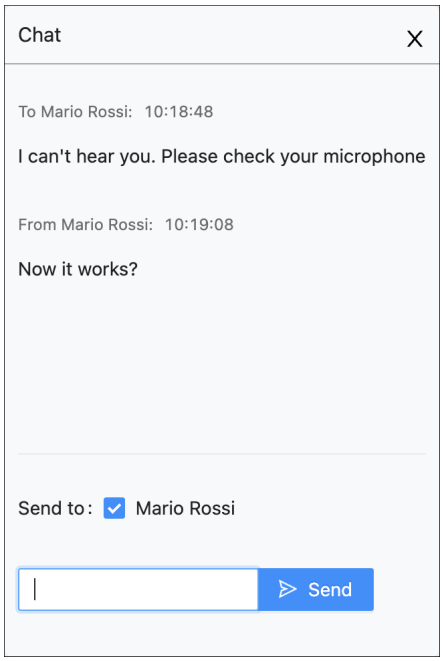


Figura 4.2: Schermata per la visualizzazione e invio di messaggi

4.5 Invio messaggio

I dati relativi ad un messaggio sono codificati mediante un oggetto JSON (vedasi Snippet di codice 4.5) composto dai seguenti campi:

- `from`: è una stringa che rappresenta l'identificativo univoco del partecipante alla videoispezione che ha inviato il messaggio;
- `to`: è un array di stringhe ciascuna rappresentante l'identificativo univoco di un partecipante a cui è destinato il messaggio;
- `datetime`: è una stringa rappresentante l'orario di invio di invio del messaggio espresso in formato HH:MM:SS;
- `text`: è una stringa rappresentante il corpo del messaggio.

```
{
    from: string
    to: Array,
    datetime: string,
    text: string
}
```

Snippet di codice 4.5: Oggetto JSON rappresentante un messaggio

Utilizzando la LocalDataTrack del client (si veda la Sottosezione 3.2.2) è possibile ricavare la relativa LocalDataTrackPublication, e utilizzarla per mandare l'oggetto, convertito in formato JSON, agli altri client collegati alla Room eseguendo il metodo riportato nello Snippet di codice 4.6.

```
sendMessage(msg) {
    let _localDataTrack;
    Array.from(this.props.room.localParticipant.dataTracks.values()).
        forEach((publication) => {
      _localDataTrack = publication.track;
    });
    if (_localDataTrack) {
      _localDataTrack.send(JSON.stringify(msg));
    }
  }
```

Snippet di codice 4.6: Metodo per l'invio di un messaggio

4.6 Notifica di microfono spento

Per implementare questa funzionalità si è fatto ricorso a *Web Audio* API, il quale appartiene ad un pacchetto più ampio di API per JavaScript chiamato *Web* API. La scelta è ricaduta su di loro perché, oltre a non richiedere installazioni ulteriori per il loro utilizzo, sono compatibili con tutti i browser più popolari e utilizzati al giorno d'oggi.

```
getVolume() {
  if (this.analyser) {
    const pcmData = new Float32Array(this.analyser.fftSize);
    this.analyser.getFloatTimeDomainData(pcmData);
    let sumSquares = 0.0;
    for (const amplitude of pcmData) {
      sumSquares += amplitude * amplitude;
    }
    return Math.sqrt(sumSquares / pcmData.length);
  }
  return 0;
}
```

Snippet di codice 4.7: Metodo per il rilevamento del volume audio registrato dal microfono.

Il metodo riportato nello Snippet di codice 4.7 ritorna un valore compreso tra 0 e 1. Fissata una soglia al valore 0.1, periodicamente ogni 2 secondi viene eseguita la rilevazione, e lo stato del componente relativo al volume viene aggiornato. Questo scatena l'invocazione del metodo `componentDidUpdate()` in cui si controlla se:

* l'utente ha il microfono spento;
* in precedenza l'utente non aveva il microfono spento, oppure il valore precedentemente rilevato era inferiore alla soglia;
* il valore attualmente rilevato supera la soglia e di conseguenza probabilmente l'utente sta parlando.

Se tutte e tre le affermazioni sono vere, allora l'utente viene avvisato con una notifica di allerta a scomparsa, visibile in Figura 4.3, che rimane a schermo per due secondi.

```
// If participant is muted, it's apparently talking, and it wasn't
    talking before
// or it wasn't muted before, emits a warning.
if (this.checkVolume) {
  if (
      this.state.muted &&
      (prevState.volume <= 0.1 || !prevState.muted) &&
```

```
7         this.state.volume > 0.1
8       ) {
9         message.warning(this.props.p.t("inspection.are_you_talking"), 2);
10      }
11      this.checkVolume = false;
12    }
```

Snippet di codice 4.8: Output della notifica di microfono spento

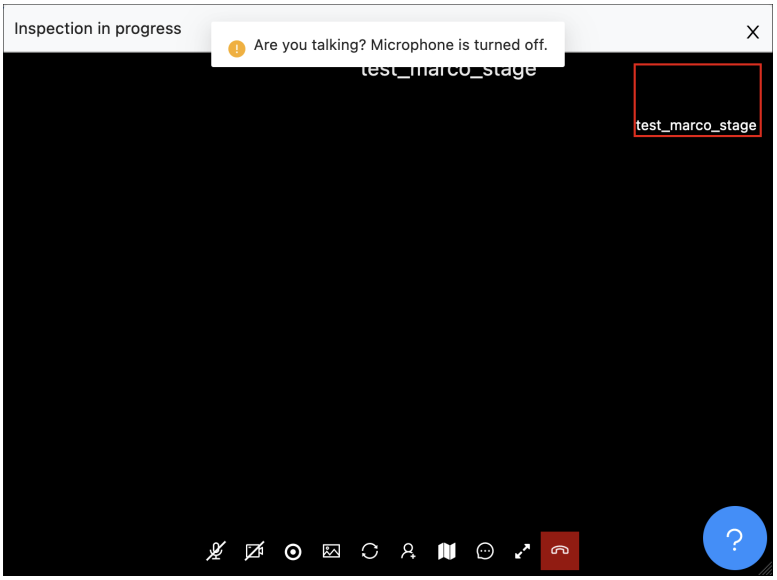


Figura 4.3: Notifica di microfono spento

4.7 Notifica di messaggio ricevuto

Come descritto precedentemente nella Sottosezione 3.2.2, un client utilizza le Track pubblicate dai suoi RemoteParticipant collegati alla Room, cioè le sue RemoteTrack, per ricevere i flussi audio, video, e dati provenienti da remoto.

Quando il client locale intercetta dei dati provenienti da una sorgente remota, viene invocato un opportuno *handler*. Questa funzione si occupa innanzitutto di controllare la natura dei dati ricevuti. Se questi ultimi sono relativi a un nuovo messaggio ricevuto, lo stato del componente `VideoRoom` viene aggiornato appendendo in coda all'array dei messaggi l'ultimo ricevuto, inoltre viene mostrata una notifica informativa a scomparsa, visibile in Figura 4.4, che rimane a schermo per due secondi.

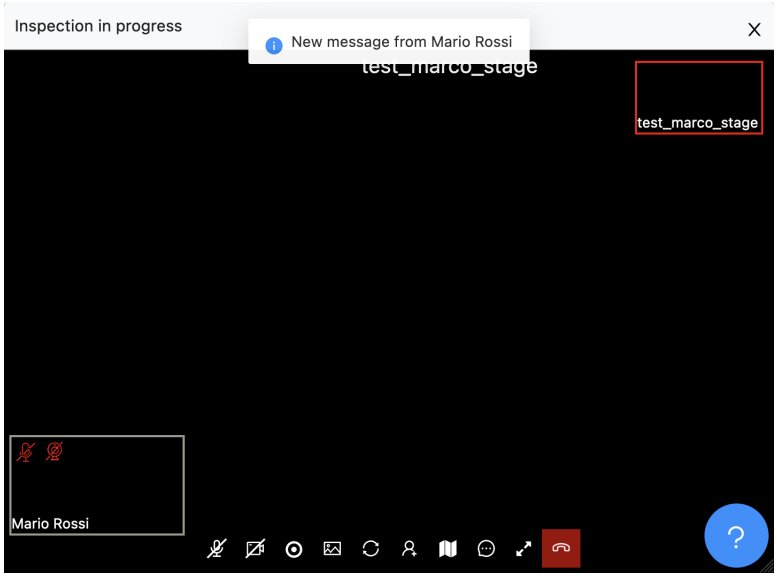


Figura 4.4: Notifica di messaggio ricevuto

4.8 Visualizzazione a schermo intero

Per la visualizzazione a schermo intero della schermata di videoispezione, ci si è affidati a *Fullscreen* API, il quale permette di presentare l'elemento desiderato usando tutto lo schermo disponibile, e rimuovendo quindi tutti gli elementi dell'interfaccia del browser e le altre applicazioni fino a quando la modalità non viene disattivata [12].

4.9 Schermata di configurazione preliminare

La schermata mostrata in Figura 4.5 viene presentata all'utente prima di collegarsi alla videoispezione. Egli può scegliere il dispositivo di input audio e video selezionando una tra le *option* dei *select*, mentre attraverso i due *checkbox* può scegliere se entrare in videochiamata con il microfono e/o la videocamera spenti.

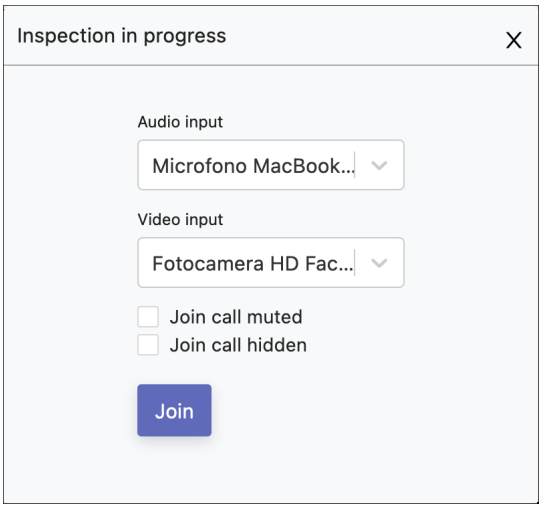


Figura 4.5: Schermata di configurazione preliminare

4.10 Hotspots

Come riportato nella Sezione 3.3, si è scelto di utilizzare un *onboarding pattern* per implementare, attraverso React e CSS, la visualizzazione guidata della videoispezione, cioè gli *hotspot*. Questi appaiono subito evidenti all'occhio dell'utente, come si può vedere in Figura 4.6, dove corrispondono ai pallini blu lampeggianti. Muovendo il cursore sopra gli stessi, o selezionandoli tramite il tasto tab, compare un pop-up che illustra all'utente come utilizzare la videoispezione. Sebbene siano immediatamente riconoscibili e attirino l'attenzione, non distraggono l'utente durante il normale utilizzo della videoispezione, ma l'utente può comunque scegliere, se lo desidera, di disattivarli premendo il bottone posto in basso a destra.

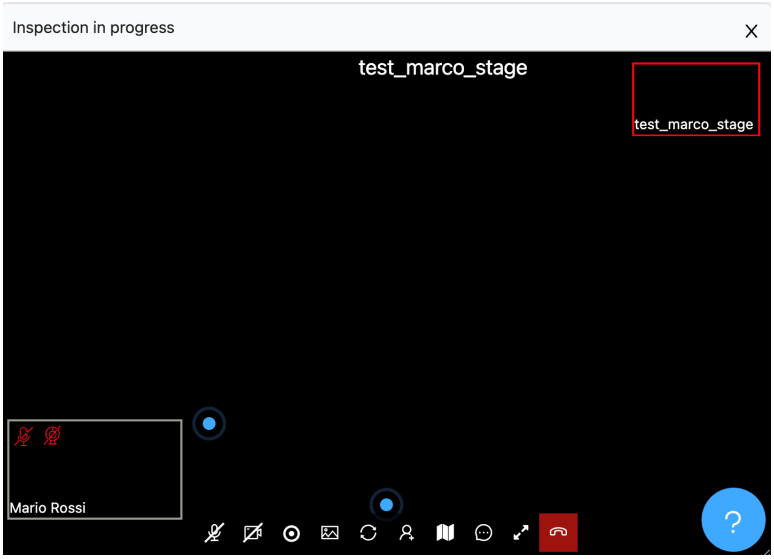


Figura 4.6: Hotspots mostrati a video

4.11 Verifica e validazione

Al termine della fase di codifica, si è svolta la verifica e la validazione del lavoro compiuto. Nella presente sezione si spiega nel dettaglio in che cosa consistono, e le attività svolte.

4.11.1 Verifica

La verifica è servita ad accertare che l'esecuzione delle attività svolte nel periodo di stage, in particolare la codifica, non abbia introdotto errori nel sistema [39].

Analisi statica

L'analisi statica è un metodo per effettuare il debug del software, e viene effettuato esaminando il codice non a *run-time*, e quindi senza che il software venga eseguito. L'analisi statica si rivela efficace per trovare problemi di codifica come:

* errori di programmazione;
* qualità del codice insufficiente;
* valori indefiniti;
* errori di sintassi;
* falle nel sistema che potrebbero comprometterne la sua sicurezza.

Durante tutta la fase di codifica è stata eseguita l'attività di analisi statica nei seguenti modi:

* lettura ripetuta e accurata del codice scritto in modo da trovare errori di programmazione, valori indefiniti o errori di sintassi;
* mantenimento di una qualità del codice adeguata effettuando *code refactoring*.

Analisi dinamica

Al contrario dell'analisi statica, l'analisi dinamica viene effettuata a *run-time*, cioè mentre il software è in esecuzione.

I test sono stati svolti seguendo un approccio manuale. Per ogni caso d'uso individuato, si è controllato se, in seguito a una serie di operazioni, l'output prodotto fosse quello atteso.

4.11.2 Validazione

La validazione accerta che il prodotto finale sia pienamente conforme alle aspettative [39]. La validazione è stata svolta con la partecipazione del proponente, in questo caso il tutor aziendale, il quale ha controllato e confermato che le funzionalità implementate fossero quelle attese.

Sono stati redatti e consegnati i seguenti documenti:

* analisi dei requisiti: descrive i casi d'uso e i requisiti individuati;
* manuale dello sviluppatore: descrive e spiega in maniera approfondita, utilizzando un linguaggio tecnico e in lingua inglese, il codice prodotto, inoltre presenta delle schermate che mostrano il corrispondente output.

Il codice prodotto, inoltre, è corredato da commenti in lingua inglese, che ne permettono una comprensione più semplice e immediata.

Capitolo 5

Conclusioni

Il presente capitolo riporta il consuntivo delle attività svolte, gli obiettivi raggiunti, i requisiti soddisfatti e una breve valutazione personale del lavoro complessivo.

5.1 Consuntivo finale

La Tabella 5.1 riporta il consuntivo delle attività svolte durante lo stage. In generale, il piano iniziale è stato rispettato, seppur con qualche lieve differenza. La fase di analisi e pianificazione è durata più di quanto preventivato perché è stato necessario spendere più tempo per individuare i casi d'uso, e per lo studio del codice preesistente. Non si è svolta l'attività di stesura di test, in quanto le fasi di analisi, pianificazione e sviluppo hanno richiesto più ore rispetto a quanto pianificato.

Tabella 5.1: Ripartizione ore lavorative

Ore previste	**Ore svolte**	**Descrizione dell'attività**
72	**72**	**Formazione sulle tecnologie**
40	40	Formazione su ReactJS
32	32	Ricerca e studio di eventuali librerie utili
64	**76**	**Analisi e pianificazione**
120	**124**	**Sviluppo**
24	16	Implementazione della visualizzazione guidata della registrazione dei video durante le videoispezioni
16	12	Implementazione della visualizzazione guidata della registrazione delle immagini durante le videoispezioni
8	8	Implementazione dell'invio di email d'invito agli utenti guest durante le videoispezioni
16	16	Implementazione della visualizzazione guidata della creazione delle registrazioni per la consultazione
16	16	Implementazione dell'inserimento di azioni mancanti nella visualizzazione di immagini e video delle videoispezioni

Tabella 5.1: Ripartizione ore lavorative

Ore previste	**Ore svolte**	**Descrizione dell'attività**
40	56	Implementazione dell'inserimento di azioni mancanti e miglioramento di azioni durante le videoispezioni
64	**48**	**Collaudo finale**
32	-	Stesura dei test
16	32	Collaudo e verifica
16	16	Stesura documentazione finale
320	**320**	**Totale ore**

5.2 Raggiungimento degli obiettivi

5.2.1 Completamento degli obiettivi

Nella Tabella 5.2 viene riportato il completamento degli obiettivi da raggiungere. Il mancato completamento dell'obiettivo F01, il quale era considerato facoltativo, è legato al fatto che si è reso necessario investire più tempo del previsto nel portare a termine gli obiettivi obbligatori e desiderabili.

Tabella 5.2: Completamento degli obiettivi

Codice	**Descrizione**	Stato
O01	Visualizzazione guidata della registrazione dei video durante le videoispezioni	Completato
O02	Visualizzazione guidata della registrazione delle immagini durante le videoispezioni	Completato
O03	Invio email live agli utenti guest durante le videoispezioni	Completato
O04	Visualizzazione guidata della creazione delle registrazioni per la consultazione	Completato
O05	Inserimento di azioni mancanti e miglioramento di azioni durante le videoispezioni	Completato
O06	Stesura documento manuale dello sviluppatore	Completato
D01	Inserimento di azioni mancanti nella visualizzazione di immagini e video delle videoispezioni	Completato
F01	Devono essere presenti dei test automatici a livello di *front-end*	Non completato

5.2.2 Soddisfacimento dei requisiti

Con riferimento alla Tabella 2.1, nella quale sono elencati i requisiti funzionali, alla Tabella 2.2, nella quale sono elencati i requisiti qualitativi e alla Tabella 2.3, nella quale sono riportati i requisiti di vincolo, in Tabella 5.3, Tabella 5.4 e Tabella 5.5 vengono riportati i requisiti con il loro stato di soddisfacimento.

Tabella 5.3: Soddisfacimento dei requisti funzionali

Requisito	Descrizione	Stato
RFN-1	L'utente deve poter visualizzare la schermata a schermo intero	Soddisfatto
RFN-2	L'utente deve poter visualizzare i messaggi che ha inviato e che ha ricevuto	Soddisfatto
RFN-3	L'utente deve poter inviare un messaggio a un altro utente collegato alla videoispezione	Soddisfatto
RFN-4	L'utente deve essere avvisato se sta parlando ma ha il microfono spento	Soddisfatto
RFN-5	L'utente deve essere avvisato se riceve un messaggio	Soddisfatto
RFN-6	L'utente deve essere guidato all'utilizzo della videoispezione attraverso una procedura di *onboarding*	Soddisfatto
RFN-7	L'utente deve scegliere il dispositivo di input audio da utilizzare durante la videoispezione prima di collegarsi alla stessa	Soddisfatto
RFN-8	L'utente deve scegliere il dispositivo di input video da utilizzare durante la videoispezione prima di collegarsi alla stessa	Soddisfatto
RFN-9	L'utente deve scegliere se rendere attivo o meno il dispositivo di input audio prima di collegarsi alla videoispezione	Soddisfatto
RFN-10	L'utente deve scegliere se rendere attivo o meno il dispositivo di input audio prima di collegarsi alla videoispezione	Soddisfatto

Tabella 5.4: Soddisfacimento dei requisiti qualitativi

Requisito	Descrizione	Stato
RQN-1	Le funzionalità implementate devono essere tradotte sia in italiano che in inglese	Soddisfatto
RQN-2	Stesura documento di analisi dei requisiti	Soddisfatto
RQN-3	Stesura manuale dello sviluppatore	Soddisfatto
RQZ-1	Copertura test dell'80% nelle funzionalità implementate	Non soddisfatto

Tabella 5.5: Soddisfacimento dei requisiti di vincolo

Requisito	Descrizione	Stato
RVN-1	Le funzionalità implementate devono funzionare con i browser: Google Chrome (v. 97.0.4692), Mozilla Firefox (v. 95.0) e Microsoft Edge (v. 97.0.1072.69)	Soddisfatto

Su un totale di 15 requisiti, 14 sono stati soddisfatti. Il requisito RQF-1 non è stato soddisfatto per ragioni legate alla tempistiche dello stage, infatti il suo compimento avrebbe richiesto un dispendio stimato in circa 32 ore lavorative.

Nonostante ciò, l'azienda si è comunque detta soddisfatta del lavoro svolto e della documentazione fornita.

5.3 Valutazione personale

Ritengo innanzitutto che questa esperienza sia stata importante dal punto di vista formativo. Infatti mi ha permesso di approfondire le conoscenze in mio possesso relative al *framework* React, il quale è una delle tecnologie più popolari e più utilizzate al giorno d'oggi per quanto riguarda lo sviluppo web lato *front-end*. A prova di ciò, si può affermare che è utilizzato da grandi compagnie del settore informatico e non solo.

In generale ho avuto modo di gestire tutto il percorso per lo sviluppo delle funzionalità implementate: partendo dallo svolgimento dell'analisi dei requisiti con il proponente, passando alla presentazione di un *mockup* grafico, svolgendo la fase di progettazione e codifica, e concludendo con la redazione della documentazione e la fase di validazione. Le 320 ore di stage svolte mi hanno permesso di affinare le mie conoscenze informatiche e organizzative, dandomi la possibilità di applicare le nozioni apprese durante il mio percorso di studi universitario.

Acronimi e abbreviazioni

API Application Program Interface. 25, 26, 28, 33, 36, 38, 49

CD Continuous Delivery. 2, 49

CI Continuous Integration. 2, 49

DOM Document Object Model. 25, 49

HTTP HyperText Transfer Protocol. 28, 29, 33, 50

IDE Integrated Development Enviroment. 21, 50

SaaS Software as a Service. 1, 51

UI *User Interface*. 22, 31

UML Unified Modeling Language. 5, 6, 8, 9, 11, 14, 51

Glossario

agile in ingegneria del software, la metodologia *agile* è un insieme di pratiche per lo sviluppo iterativo e incrementale del software. Si contrappone al modello a cascata e ad altri modelli di sviluppo tradizionali, proponendo un approccio meno strutturato e focalizzato sull'obiettivo di consegnare al cliente, in tempi brevi e frequentemente, software funzionante e di qualità [2]. 1, 2

API in informatica, con il termine *API, Application Programming Interface* (in italiano "interfaccia di programmazione di un'applicazione") si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 47

camelCase è una convenzione sui nomi, in cui la prima lettera di ogni parola, eccetto la prima, che forma il nome è scritta in maiuscolo. 32

CD in ingegneria del software, *CD, Continuous Delivery* (in italiano "distribuzione continua") è un approccio di ingegneria del software in cui i team producono software in cicli brevi, garantendo che il software possa essere rilasciato in modo affidabile in qualsiasi momento e, al momento del rilascio del software, farlo manualmente. Mira a creare, testare e rilasciare software con maggiore velocità e frequenza. L'approccio aiuta a ridurre i costi, i tempi e i rischi legati alla fornitura delle modifiche consentendo aggiornamenti più incrementali alle applicazioni in produzione [6]. 47

CI in ingegneria del software, *CI, Continuous Integration* (in italiano "integrazione continua") è una pratica che si applica in contesti in cui lo sviluppo del software avviene attraverso un sistema di versioning. Consiste nell'allineamento frequente (ovvero "molte volte al giorno") dagli ambienti di lavoro degli sviluppatori verso l'ambiente condiviso (*mainline*). Il concetto è stato originariamente proposto nel contesto dell'*extreme programming* (XP), come contromisura preventiva per il problema dell' *"integration hell"* (le difficoltà dell'integrazione di porzioni di software sviluppati in modo indipendente su lunghi periodi di tempo e che di conseguenza potrebbero essere significativamente divergenti) [7]. 47

codebase è l'intera collezione di codice sorgente usata per costruire una particolare applicazione o un particolare componente [5]. 2, 22

DOM è una forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti. È lo standard ufficiale del W3C (World Wide Web

Consortium) per la rappresentazione di documenti strutturati in maniera da essere neutrali sia per la lingua che per la piattaforma. È inoltre la base per una vasta gamma di interfacce di programmazione delle applicazioni, alcune di esse standardizzate dal W3C [9]. 47

ECMAScript è un'organizzazione no-profit per la creazione di standard nell'ambito delle tecnologie informatiche, tra cui JavaScript, e l'elettronica di consumo [10]. 17

HTTP in telecomunicazioni e informatica, *HTTP, HyperText Transfer Protocol* (in italiano "protocollo di trasferimento ipertesto") è un protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni sul web ovvero in un'architettura tipica client-server. Le specifiche del protocollo sono gestite dal World Wide Web Consortium (W3C) [16]. 47

IDE in informatica, *IDE, Integrated Development Enviroment* (tradotto ambiente di sviluppo integrato) è un software che offre una serie di funzioni per lo sviluppo, il testing e il debugging del codice di un programma. 47

issue tracking system è un software che consente agli utenti di registrare e seguire l'avanzamento di ogni ticket del cliente o "issue" nella loro casella di posta fino alla risoluzione del problema. I problemi possono essere monitorati nel contesto di applicazioni software, servizi informatici e altro ancora. Il processo di tracciamento dei problemi inizia con i clienti che informano i team di supporto che esiste un potenziale problema che stanno riscontrando. Il sistema di tracciamento dei problemi lo registra e lo memorizza insieme ad altri problemi sollevati dal cliente in passato. Un agente dei team di supporto viene quindi informato che esiste un potenziale problema che il cliente sta affrontando. Esaminando il problema, insieme ai problemi passati che il cliente ha registrato, gli agenti avranno una comprensione molto maggiore di come risolvere al meglio il problema del cliente. Questo approccio personalizzato aumenta la soddisfazione degli utenti e può potenzialmente aiutare a fidelizzare i clienti [17]. 2

JSX è un'estensione React del linguaggio JavaScript che fornisce un modo per strutturare il rendering dei componenti utilizzando una sintassi familiare a molti sviluppatori. A livello sintattico, è simile a HTML. 31, 32

merge nell'ambito dei sistemi di controllo di versione, il *merge* è un'operazione con cui si integrano i cambiamenti in un *branch* all'interno del *branch* principale chiamato HEAD oppure "master". 22

OpenStreetMap è un progetto collaborativo finalizzato a creare mappe del mondo a contenuto libero. Il progetto punta ad una raccolta mondiale di dati geografici, con scopo principale la creazione di mappe e cartografie [23]. 28, 33

PascalCase è una convenzione sui nomi, in cui la prima lettera di ogni parola che forma il nome è scritta in maiuscolo. 32

query string è la parte di un indirizzo web che contiene dei dati, codificati in forma di stringa, da passare in input ad un programma. 28, 33

reverse geocoding è il processo inverso del *geocoding*, ossia date delle coordinate geografiche viene restituito un indirizzo. 28, 33

SaaS in informatica *SaaS, Software as a Service* (in italiano "software come servizio") è un modello di servizio del software applicativo, realizzato da un produttore che mette a disposizione un programma, direttamente o tramite terze parti, con modalità telematiche come ad esempio un'applicazione web [32]. 47

single-page application è un'applicazione web o un sito web che può essere usato o consultato su una singola pagina web con l'obiettivo di fornire una esperienza utente più fluida e simile alle applicazioni desktop dei sistemi operativi tradizionali [33]. 24

UML in ingegneria del software, *UML, Unified Modeling Language* (in italiano "linguaggio di modellazione unificato") è un linguaggio di modellazione e specifica basato sul paradigma *object-oriented*. *UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 47

web app in informatica ed in particolare nella programmazione web, una web app (in italiano "applicazione web") indica genericamente tutte le applicazioni distribuite, ovvero applicazioni accessibili/fruibili via web per mezzo di un network, come ad esempio una Intranet all'interno di un sistema informatico o attraverso internet, cioè in una architettura tipica di tipo client-server, offrendo determinati servizi all'utente client [40]. 2, 22, 24, 31

Bibliografia

Riferimenti bibliografici

[1] J.M. Carroll e M.B. Rosson. *Paradox of the active user*. The MIT Press, 1987 (cit. a p. 28).

Siti web consultati

[2] *Metodologia agile - Wikipedia*. URL: https://it.wikipedia.org/wiki/Metodologia_agile (cit. alle pp. 1, 49).

[3] *Ant Design - The world's second most popular React UI framework*. URL: https://ant.design/ (cit. a p. 20).

[4] *Continuous Integration and Delivery - CircleCI*. URL: https://circleci.com/ (cit. a p. 2).

[5] *Codebase - Wikipedia*. URL: https://it.wikipedia.org/wiki/Codebase (cit. alle pp. 2, 49).

[6] *Continuous Delivery - Wikipedia*. URL: https://en.wikipedia.org/wiki/Continuous_delivery (cit. alle pp. 2, 49).

[7] *Integrazione continua - Wikipedia*. URL: https://it.wikipedia.org/wiki/Integrazione_continua (cit. alle pp. 2, 49).

[8] *CSS - Wikipedia*. URL: https://it.wikipedia.org/wiki/CSS (cit. a p. 19).

[9] *Document Object Model - Wikipedia*. URL: https://it.wikipedia.org/wiki/Document_Object_Model (cit. alle pp. 25, 50).

[10] *Mission - ECMAScript*. URL: https://www.ecma-international.org/mission/ (cit. alle pp. 17, 50).

[11] *Figma: the collaborative interface design tool*. URL: https://www.figma.com/ (cit. alle pp. 2, 22).

[12] *Fullscreen API - Web APIs | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fullscreen_API (cit. a p. 38).

[13] *Git*. URL: https://git-scm.com/ (cit. a p. 22).

[14] *GitHub*. URL: https://github.com/ (cit. alle pp. 2, 22).

[15] *Gmail: email private e sicure | Google Workspace*. URL: https://www.google.com/intl/it/gmail/about/ (cit. a p. 2).

[16] *Hypertext Transfer Protocol - Wikipedia*. URL: https://it.wikipedia.org/wiki/Hypertext_Transfer_Protocol (cit. alle pp. 28, 50).

[17] *Issue Tracking | Technology Glossary Definitions | G2*. URL: https://www.g2.com/glossary/issue-tracking-definition (cit. alle pp. 2, 50).

[18] *Jira | Software per il monitoraggio di ticket e progetti*. URL: https://www.atlassian.com/it/software/jira (cit. a p. 2).

[19] *Introduzione a JavaScript | Guida JavaScript di base | JavaScript HTML.it*. URL: https://www.html.it/pag/45343/introduzione-a-javascript/ (cit. a p. 19).

[20] *About | Node.js*. URL: https://nodejs.org/en/about/ (cit. a p. 20).

[21] *Nominatim 4.1.1*. URL: https://nominatim.org/release-docs/latest (cit. a p. 28).

[22] *Nominatim Usage Policy (aka Geocoding Policy)*. URL: https://operations.osmfoundation.org/policies/nominatim (cit. alle pp. 29, 33).

[23] *OpenStreetMap - Wikipedia*. URL: https://it.wikipedia.org/wiki/OpenStreetMap (cit. alle pp. 28, 50).

[24] *Introducing: Product Tours - The Definitive Guide*. URL: https://userguiding.com/blog/product-tour/ (cit. a p. 28).

[25] *React (web framework) - Wikipedia*. URL: https://it.wikipedia.org/wiki/React_(web_framework) (cit. a p. 20).

[26] *Components and Props - React*. URL: https://reactjs.org/docs/components-and-props.html (cit. alle pp. 20, 31).

[27] *React Redux - javapoint*. URL: https://www.javatpoint.com/react-redux (cit. a p. 25).

[28] *Redux - A predictable state container for JavaScript apps. | Redux*. URL: https://redux.js.org/ (cit. a p. 24).

[29] *Redux Essentials, Part 1: Redux Overview and Concepts | Redux*. URL: https://redux.js.org/tutorials/essentials/part-1-overview-concepts#terminology (cit. a p. 24).

[30] *Three Principles | Redux*. URL: https://redux.js.org/understanding/thinking-in-redux/three-principles (cit. a p. 24).

[31] *Reverse - Nominatim 4.1.1*. URL: https://nominatim.org/release-docs/latest/api/Reverse (cit. a p. 28).

[32] *Software as a service - Wikipedia*. URL: https://it.wikipedia.org/wiki/Software_as_a_service (cit. alle pp. 1, 51).

[33] *Single-page application - Wikipedia*. URL: https://it.wikipedia.org/wiki/Single-page_application (cit. alle pp. 24, 51).

[34] *Slack è il tuo ambiente digitale | Slack*. URL: https://slack.com/intl/it-it (cit. alle pp. 2, 21).

[35] *Videoconferenze, riunioni, chiamate | Microsoft Teams*. URL: https://www.microsoft.com/it-it/microsoft-teams/group-chat-software (cit. a p. 2).

[36] *Understanding Video Rooms APIs | Twilio*. URL: https://www.twilio.com/docs/video/tutorials/understanding-video-rooms-apis#the-video-client-sdk-api (cit. a p. 27).

[37] *Twilio Video Overview | Twilio*. URL: https://www.twilio.com/docs/video/overview (cit. a p. 26).

[38] *Diagrammi Use Case - Diagrammi Use Case.pdf*. URL: https://www.math.unipd.it/~rcardin/swea/2022/Diagrammi%20Use%20Case.pdf (cit. a p. 5).

[39] *Microsoft PowerPoint - A15 - Verifica e validazione.pptx - T14.pdf*. URL: https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/T14.pdf (cit. a p. 40).

[40] *Applicazione web - Wikipedia*. URL: https://it.wikipedia.org/wiki/Applicazione_web (cit. alle pp. 2, 51).

[41] *What is WebRTC? | Twilio | www.twilio.com*. URL: https://www.twilio.com/docs/glossary/what-is-webrtc (cit. a p. 25).

[42] *WebStorm: ottimo per imparare a scrivere del codice Javascript*. URL: https://www.laramind.com/blog/webstorm-ottimo-per-scrivere-codice-js/ (cit. a p. 21).