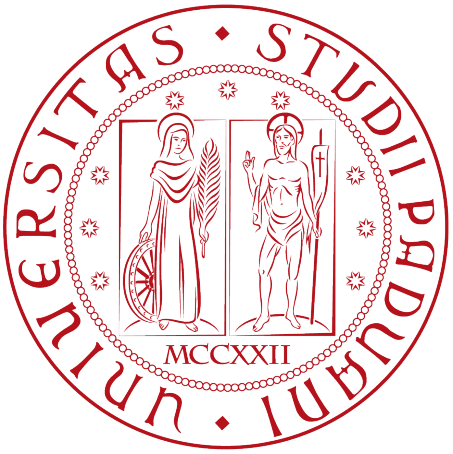
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Sviluppo del frontend di un'applicazione web in React js

Tesi di laurea

Relatore

Prof.Paolo Baldan

Laureando

Mattia Episcopo

Anno Accademico 2022-2023

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

— Oscar Wilde

Dedicato a ...

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Pinco Pallino presso l'azienda Azienda S.p.A. Gli obbiettivi da raggiungere erano molteplici.
In primo luogo era richiesto lo sviluppo di ... In secondo luogo era richiesta l'implementazione di un ... Tale framework permette di registrare gli eventi di un controllore programmabile, quali segnali applicati Terzo ed ultimo obbiettivo era l'integrazione ...

"I'm not crazy, my mother had me tested"

— Dott. Sheldon Lee Cooper

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. NomeDelProfessore, 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, Febbraio 2023 Mattia Episcopo

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Introduzione al contesto applicativo.

Esempio di utilizzo di un termine nel glossario
Application Program Interface (API).

Esempio di citazione in linea
site:agile-manifesto.

Esempio di citazione nel pie' di pagina
citazione[1]

1.1 L'azienda

Synthema Artificial Intelligence (S.AI) è una start-up innovativa che si occupa di ricerca, progettazione, sviluppo, commercializzazione e manutenzione di prodotti e servizi innovativi ad alto valore tecnologico, basati sull'Internet of Things, la blockchain e su tecniche di Intelligenza Artificiale (in particolare reti neurali profonde), per l'analisi integrata e la comprensione di dati multimodali da fonti eterogenee (linguaggio naturale sia scritto che parlato, audio, immagini, video, dati generati da sensori) e per la gestione dei workflow, sia nel settore pubblico che privato.

1.2 Introduzione al progetto

Lo stage svolto presso SAI aveva come scopo la creazione di un prototipo di piattaforma web/desktop per il caricamento delle registrazioni audio effettuate nelle aule di tribunale, con l'aiuto di alcuni metadata prodotti durante le medesime registrazioni. In particolare l'applicazione da sviluppare è divisa in due parti backend e frontend (spiegare meglio back e front) che comunicano tra loro mediante API rest (spiegare API). Lo stage si è focalizzato sullo sviluppo della parte frontend, avanzando di pari passo con lo sviluppo del backend realizzata da altri membri del team di sviluppo. Per la realizzazione del frontend si è scelto di utilizzare Javascript con i framework React e Redux.

[1] **womak:lean-thinking**.

[fare merge dei due - secondo quello scritto nuovo] [valutare se iniziare da qui a introdurre i concetti di Metadati: Marker, Procedimenti, Interventi; Jobs]

1.3 Introduzione al progetto

Il progetto nasce dalla necessità dell'azienda di creare un sistema semi-automatico per la creazione di particolari ticket. In particolare il sistema è rivolto ai lavoratori della aule di tribunale, nel nostro caso chiamati fonici, che dopo aver registrato la loro porzione di giornata nel tribunale di competenza, devono inserire tutti i dati relativi ai processi nel sistema di memorizzazione. I dati che vengono raccolti in aula sono registrati come tracce audio su dei CD, seguendo degli schemi abbastanza rigidiù. Queste tracce audio sono regolate da un file di testo, ove il fonico, seguendo gli stessi schemi delle registrazioni audio prende degli appunti su quanto sta accadendo in aula in quel momento. Un gruppo di dati che si riferisce ad un procedimento è quindi interpretato come ticket dai fonici, che ne devono inserire una grande quantità per ogni giornata lavorativa. Con questo progetto l'azienda intende risolvere il problema, dando la possibilità a questi lavoratori, di caricare i dati relativi ai procedimenti senza dover compilare nessun modulo per creare un ticket ma facendo in modo che questi, vengano automaticamente creati leggendo le tracce audio dei CD caricati e il file di testo che contiene gli appunti.

1.4 Principali problematiche

Per questo progetto sorgono diverse problematiche che elenco di seguito:

* **tecnologiche**: tutte le tecnologie da utilizzare necessitano di una fase di apprendimento perchè inizialmente risultano per lo più non conosciute
* **argomento del progetto**: l'argomento del progetto, ossia le registrazioni delle aule di tribunale, è conosciuto solo in parte
* **tipi di dati e metadati**: i dati e metadati contenuti nei vari file relativi alle registrazione sono molto eterogenei e una loro classificazione dettagliata risulta molto difficile

1.5 Soluzione scelta

Il progetto pone vari dubbi nella fase di analisi per scegliere la soluzione migliore, che sia allo stesso tempo veloce nel caricamento dei dati e nella creazione dei ticket ma allo stesso tempo che non perda informazioni importanti, visto la sensibilità dei dati trattati. Viene scelto per questo progetto di creare un'applicazione inizialmente web ma con la possibilità di renderla desktop in futuro. L'applicazione sarà formata da un backend che servirà per tenere traccia delle informazioni che realmente servono, e un frontend che servirà principalmente per la composizione delle informazioni da passare al backend. In questo modo ogni volta che un CD viene caricato non viene interamente mandato al server, ma rimane in gestione al frontend che ne elabora i dati e li gestisce seguendo le preferenze dell'utente, prima di passare solo i dati necessari al backend. Per la memorizzazione dei file invece la strada pensata in fase di analisi è l'utilizzo di un client (object-storage) esterno dove fare l'upload solo delle tracce audio

necessarie. L'oggetto del progetto di stage è proprio la parte di frontend di questo sistema pensato per agevolare il lavoro dei collaboratori delle aule di tribunale.

1.6 Descrizione del prodotto ottenuto

Il prodotto finale ottenuto si discosta leggermente da quello inizialmente pensato, infatti è stata soltanto abbozzata la costruzione del ticket vero e proprio per motivi legati alle tempistiche dello stage e per dare la precedenza agli obiettivi obbligatori e desiderabili dello stage stesso. L'applicazione finale risulta un buon punto di partenza per costruire la restante parte del prodotto. Il prodotto ottenuto ha come inizio l'autenticazione dell'utente con la maschera di login, in modo che l'accesso a tutte le operazioni sia consentito solo agli utenti autorizzati. L'applicazione nello stato finale presenta le seguenti funzionalità:

* **caricamento dei dati**: l'utente può caricare i dati sull'applicazione scegliendo se caricare interi CD di registrazioni o singoli file, questa seconda funzionalità risulta molto utile nel caso l'utente intenda aggiungere manualmente file che magari non sono presenti nel CD caricato in precedenza.

* **viste dati caricati**: l'utente può scegliere di vedere i dati caricati ordinandoli sia per file sia per procedimento, quest'ultima risulta molto importante per l'utente finale perchè può visualizzare le tracce audio caricate raggruppate per procedimento

* **compilazione automatica form**: al caricamento del CD contenente le registrazioni i form relativi ai procedimenti vengono automaticamente compilati in modo da facilitare lìutente che deve solo controllarli

* **riproduzione audio dei file**: nella pagina di visualizzazione dei file l'utente ha la possibilità di riprodurre ogni traccia audio caricata

* **visulizzazione metadati del file**: grazie all'interpretazione dei metadati, per ogni file caricato è possibile vedere tutte le informazioni che ha a disposizione, nello specifico, l'utente può visualizzare tutti gli interventi e le annotazioni che sono state fatte

* **caricamento dei file sul sistema**: dopo la revisione dei dati, quelli ritenuti validi possono essere facilmente caricati sul Backend

* **riepilogo dati salvati**: l'utente può visulizzare la lista dei dati dei quali è già stato fatto l'upload e andare nel dettaglio di un singolo file per visualizzare più informazioni

1.7 Strumenti utilizzati

L'azienda usa un'infrastruttura già collaudata della quale fanno parte sistemi di versionamento del codice, strumenti per l'organizzazione del lavoro e per la comunicazione all'interno del gruppo di lavoro. Di suguito una panoramica dettagliata degli strumenti utilizzati per lavorare nel team di sviluppatori dell'azienda.

slack

E lo strumento che viene usato per la collaborazione aziendale, permette di vedere lo stato degli utenti (sviluppatorei dell'azienda nel nostro caso) se disponibili o assenti al momento. Questa applicazione da la possibilità di comunicare singolarmente con gli altri partecipanti oppure in piccoli gruppi divisi per progetto. Permette in oltre la condivisione dei file, molto utile nelle comunicazioni veloci.

Jira

Jira è lo strumento che l'azienda usa per l'organizzazione del lavoro. Permette di creare vari progetti, e per ogni progetto consente la creazione dei relativi ticket. In base alla metodologia di lavoro che si sceglie di utilizzare lo strumento permette di adottarla in tutto. Nel nostro caso la scelta aziendale è il metodo SCRUM e questo strumento consente la creazione degli sprint, è fornito di un backlog, ha una bacheca personalizzata per ogni utente che consente di vedere lo stato di avanzamento dei ticket di interesse. Inoltre consente di creare report sull'andamento degli sprint oppure su alcuni periodi, cosa molto utile per tracciare la guida nel miglioramento aziendale.

Gitlab

Sistema di versionamento dei file di codice usato dall'azienda. Questo strumento si basa su git e consente tutte le operazioni di un repository git. I file vengono condivisi tra tutti gli svilupparori grazie a questo strumento, divisi in repository, uno per ogni progetto sul quale l'azienda lavora. La linea guida aziendale per l'iutilizzo del sistema GitLab si basa sul workflow denominato feature branching, questo sistema di lavoro prevede di lasciare il ramo principale (solitamente chiamato main) sempre pulito e con una versione funzionante e testata del prodotto, mentre invece quanto si vuole lavorare su una nuova feature si crea un nuova ramo partendo da quello principale e una volta che questa feature sarà pronta per essere revisionata e integrata con quella principale si chiederà una merge request di questo feature branch sul branch principale. Il mantenimento dei file di codice sul repository si basa sulla regola aziendale 1 modifica - 1 commit - 1 file, ovvero si predilige che ogni branch che lo sviluppatore crea per sviluppare la relativa feature sia composto da commit che riguardano soltato un file con la relativa modifica.

VS Code

E l'IDE () di lavoro utilizzato per sviluppare il locale. Questo strumento è completo di tutto quello che serve per lo sviluppo del prodotto. Esso infatti consente di sviluppare nei linguaggi che interessano il progetto, aiutando con vari plugin per il riconoscimento del codice, aiutando così lo sviluppatore a lavorare più velocemente e in modo più intuitivo. Inoltre è competamente integrato con i sistemi di versionamento, in particolare nel nostro caso con GitLab. Oltre a queste funzionalità consente di utilizzare plugin per la pulizia del codice, che settati in maniera corretta permettono di rimanere sempre fedeli alle regole aziendali automaticamente ad ogni salvataggio.

Jitsi

E lo strumento che l'azienda utilizza per le riunioni in videocall settimanali ma anche per quelle individuali che dovessero essere necessarie in qualsiasi momento.

SCRUM

SCRUM è il framework che l'azienda segue per il lavoro in gruppo sui vari progetti. Questo framework molto diffuso per lo sviluppo software in team si basa su sprint di durata breve che mirano al completamento dei task assegnati, il ripetersi temporale di questi sprint porta l'avanzamento del prodotto. Nel nostro caso particolare quella che segue l'azienda è una versione personalizzata di questo framework. Il funzionamento è spiegato di seguito.

* Durata degli sprint: normalmente una settimana;
* Meeting del lunedì:
  - ognuno da una rapida panoramica per tenere tutti aggiornati su quello che ha svolto nella settimana precendente;
  - si discutono e assegnano i task che si trovavano nel backlog per lo sprint seguente;
  - rapida retrospettiva per valutazioni critiche dello sprint appena terminato;
* Stati dei task da svolgere:
  - da completare: nel corrente sprint ma nessuno ci sta ancora lavorando;
  - in corso: qualcuno sta lavorando a questo task;
  - pronto per la revisione: il task è stato completato, si attende che l'incaricato revisioni il codice e faccia il merge nel ramo principale;
  - completato: il task è stato revisionato e integrato nel ramo principale;
  - bloccato: per qualche ragione il task è bloccato, si scrive il motivo per il quale non può essere proseguito e se è necessario parlarne con qualcuno;
  - da posticipare: il task deve essere riposto nel backlog e ripianificato per un altro sprint;

1.8 Organizzazione del testo

Il secondo capitolo descrive ...

Il terzo capitolo approfondisce ...

Il quarto capitolo approfondisce ...

Il quinto capitolo approfondisce ...

Il sesto capitolo approfondisce ...

Nel settimo capitolo descrive ...

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*.

Capitolo 2

Descrizione dello stage

In questo capitolo viene descritto come si è pianificato e svolto lo stage presso SAI, introducendo il progetto, considerando gli obiettivi e i possibili rischi.

2.1 Analisi preventiva dei rischi

Nella fase di analisi iniziali si sono individuati i principali rischi a cui si poteva andare incontro e si è proceduto a definire le possibili soluzione per farne fronte.

1. Uso di nuove tecnologie
Descrizione: le tecnologie proposte per la gestione e lo sviluppo del progetto erano per lo più nuove o scarsamente conosciute..
Soluzione: è stato previsto un periodo iniziale di studio e formazione su queste tecnologie..

2. Modalità di lavoro smart working
Descrizione: lo stage è stato fatto completamente da remoto, e poteva portare ad una possibile mancanza di comunicazione e ad un incertezza nelle attività da svolgere..
Soluzione: il tutor aziendale si è reso disponibile a vari meeting nelle prime fasi del progetto e nella formazione iniziale, e rimanendo a disposizione per altri meeting in caso di dubbi sulle attività da svolgere..

2.2 Requisiti e obiettivi

Si farà riferimento ai requisiti secondo le seguenti notazioni:

* O per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
* D per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
* F per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo. Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, identificativo del requisito.

Codice	Descrizione
O01	autenticazione mediante server remoto
O02	lettura dati da CD
O03	precompilazione di form con i dati caricati da CD
O04	editing dei dati del form
D01	upload dei dati verso i sistemi esterni
D02	test di unità esaustivi
F01	possibilità di ascoltare le registrazioni
F02	possibilità di modificare i dati mediante interazioni evolute (per es. drag-n-drop)
F03	realizzazione di un'applicazione desktop con Electron
F04	compilazione multipiattaforma dell'applicazione desktop

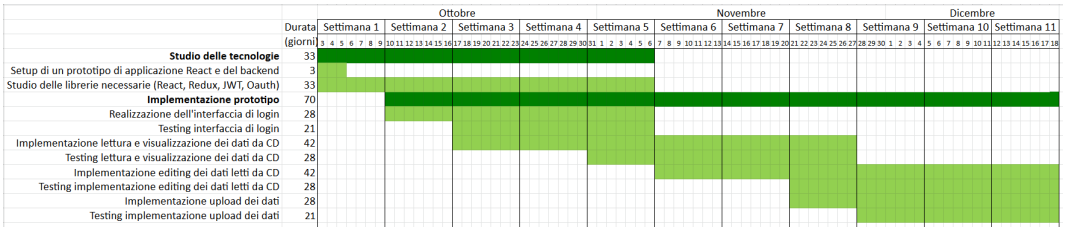
2.3 Pianificazione

La pianificazione, in termini di quantità di ore, sarà distribuita in attività di studio e attività implementative.

Lo stage è stato strutturato in due fasi principali, la prima dedicata alle attività di studio e la seconda alle attività implementative. Di seguito vengono elencate le attività pianificate con la relativa quantità di ore stimata per ogni attività.

* Studio delle tecnologie (80 ore)
  - Setup di un prototipo di applicazione React e del backend (16 ore);
  - Studio delle librerie necessarie (React, Redux, JWT, OAuth) (64 ore);
* Implementazione prototipo (240 ore)
  - Realizzazione dell'interfaccia di login (30 ore);
  - Implementazione lettura e visualizzazione dei dati da CD (70 ore);
  - Implementazione editing dei dati letti da CD (70 ore);
  - Implementazione upload dei dati (30 ore);
  - Testing dei prodotti realizzati (40ore);

Per ogni attività riguardante la fase di implementazione del prototipo è stata prevista anche la relativa attività di testing. Il tutto viene rappresentato dal seguente dal diagramma di Gantt.



2.4 Prodotto ottenuto

Capitolo 3

Analisi dei requisiti

Breve introduzione al capitolo

3.1 Casi d'uso

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) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Essendo il progetto finalizzato alla creazione di un tool per l'automazione di un processo, le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

3.1.1 UC1 - Autenticazione

* **Identificativo**: UC1
* **Nome**: autenticazione
* **Descrizione grafica**:
* **Attori**
  - *Primari*: utente non autorizzato
  - *Secondari*: Google o Faceebook (???)
* **Precondizione**: l'utente non autenticato si trova sulla pagina di autenticazione.
* **Postcondizione**: l'utente è autenticato.
* **Scenario principale**: l'utente vuole effettuare il login all'applicazione.
* **Scenario secondario**: l'utente non riesce ad autenticarsi a causa di un errore nella procedura. (**UC1.3**)

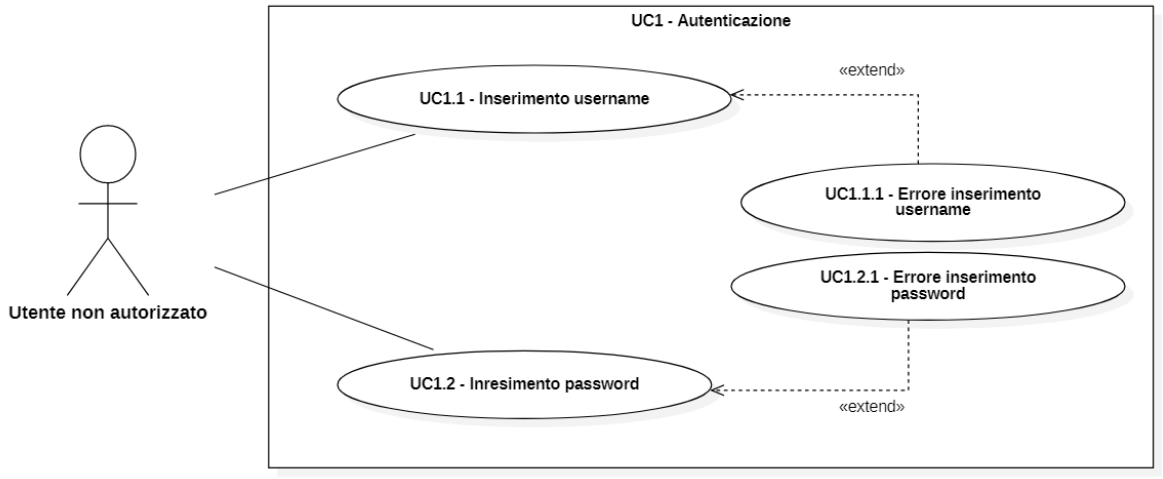


Figura 3.1: Descrizione grafica caso d'uso UC1

UC1.1 - Inserimento username

* **Identificativo**: UC1.1
* **Nome**: inserimento username
* **Descrizione grafica**: (approfondita in UC1)
* **Attori**
  - *Primari*: utente non autorizzato
* **Precondizione**: l'utente ha a disposizione una username
* **Postcondizione**: l'utente ha inserito la username.
* **Scenario principale**: l'utente inserisce la username nell'apposito campo di input.
* **Scenario secondario**: l'utente ha inserito una username non corretta che causa un errore. (**UC1.3**)

UC1.1.1 - Errore inserimento username

* **Identificativo**: UC1.1.1
* **Nome**: errore inserimento username
* **Descrizione grafica**: (approfondita in UC1)
* **Attori**
  - *Primari*: utente non autorizzato
* **Precondizione**: la username inserita dall'utente non è correttta.

* **Postcondizione**: l'errore viene mostrato all'utente.
* **Scenario principale**: l'utente inserisce una username non corretta, il sistema segnala l'errore all'utente e mostra nuovamente la maschera di login.

UC1.2 - Inserimento password

* **Identificativo**: UC1.1
* **Nome**: inserimento password
* **Descrizione grafica**: (approfondita in UC1)
* **Attori**
  - *Primari*: utente non autorizzato
* **Precondizione**: l'utente ha a disposizione una password
* **Postcondizione**: l'utente ha inserito la password.
* **Scenario principale**: l'utente inserisce la password nell'apposito campo di input.
* **Scenario secondario**: l'utente ha inserito una password non corretta che causa un errore. (**UC1.3**)

UC1.2.1 - Errore inserimento password

* **Identificativo**: UC1.2.1
* **Nome**: errore inserimento password
* **Descrizione grafica**: (approfondita in UC1)
* **Attori**
  - *Primari*: utente non autorizzato
* **Precondizione**: la password inserita dall'utente non è correttta.
* **Postcondizione**: l'errore viene mostrato all'utente.
* **Scenario principale**: l'utente inserisce una password non corretta, il sistema segnala l'errore all'utente e mostra nuovamente la maschera di login.

3.1.2 UC2 - Lettura dati da CD

* **Identificativo**: UC2
* **Nome**: lettura dati da CD
* **Descrizione grafica**:

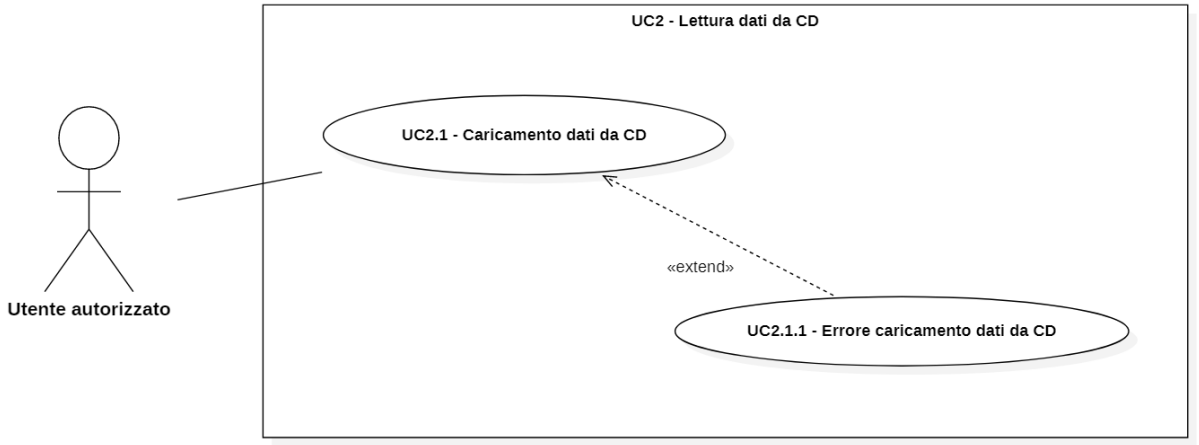


Figura 3.2: Descrizione grafica caso d'uso UC2

* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente autorizzato si trova nella pagina per il caricamento dei dati.
* **Postcondizione**: l'utente ha caricato i dati da CD.
* **Scenario principale**: l'utente premendo sull'apposito bottone può caricare i file contenuti nel CD di interesse e li visualizza (**UC3**).

UC2.1 - Caricamento dati da CD

* **Identificativo**: UC2.1
* **Nome**: caricamento dati da CD
* **Descrizione grafica**: (approfondita in UC2)
* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente ha richiesto il caricamento di una cartella contenente i file del CD.

* **Postcondizione**: i file sono stati caricati nell'applicazione.
* **Scenario principale**: l'utente richiede il caricamento di una cartella contenente i file del CD premendo sull'apposito bottone.
*
* **Scenario secondario**: il sistema riscontra un errore nella procedura di caricamento dei dati. (**UC2.1**)

UC2.1.1 - Errore caricamento dati da CD

* **Identificativo**: UC2.1
* **Nome**: errore caricamento dati da CD
* **Descrizione grafica**: (approfondita in UC2)
* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente ha tentato di caricare i dati.
* **Postcondizione**: l'errore viene mostrato all'utente.
* **Scenario principale**: il sistema non è riuscito a gestire la richiesta di caricamento dei dati da parte dell'utente.

3.1.3 UC3 - Visualizzazione file

* **Identificativo**: UC3
* **Nome**: visualizzazione file
* **Descrizione grafica**:

Figura 3.3: Descrizione grafica caso d'uso UC3

* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente si trova sulla pagina per il caricamento dati, che ha già effettuato correttamente.
* **Postcondizione**: l'utente visualizza i file e i relativi metadati caricati.
* **Scenario principale**: l'utente ha caricato correttamente i dati, questi vengono visualizzati con una lista di file ognuno con i relativi metadati.
* **Scenario secondario**: l'utente può visualizzare ed elaborare i metadati relativi ad ogni file **UC5**.

UC3.1 - Visualizzazione lista file

* **Identificativo**: UC3.1
* **Nome**: visualizzazione lista file
* **Descrizione grafica**: (approfondita in UC3)
* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente si trova sulla pagina per il caricamento dati, che ha già effettuato correttamente.
* **Postcondizione**: l'utente visualizza la lista dei file ognuno con i relativi metadati.
* **Scenario principale**: l'utente visualizza la lista dei file e può visualizzare la lista dei metadati associati.
* **Scenario secondario**: l'utente può riprodurre i file audio **UC3.1.1**.

UC3.1.1 - Riproduzione audio file

* **Identificativo**: UC3.1.1
* **Nome**: riproduzione audio file
* **Descrizione grafica**: (approfondita in UC3)
* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente si trova sulla pagina per il caricamento dati, che ha già effettuato correttamente.
* **Postcondizione**: il file audio viene riprodotto.
* **Scenario principale**: l'utente comanda il riproduttore audio con gli appositi comandi play/pause.

[Manca visualizzazione/modifica codici procedimenti da capire come gestire questo caso d'uso]

3.1.4 UC4 - Visualizzazione procedimenti

* **Identificativo**: UC4
* **Nome**: visualizzazione procedimenti
* **Descrizione grafica**:

Figura 3.4: Descrizione grafica caso d'uso UC4

* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente ha già effettuato correttamente il caricamento dati da CD.
* **Postcondizione**: l'utente visualizza i procedimenti relativi ai dati caricati.
* **Scenario principale**: l'utente da questa vista, può visualizzare la lista dei procedimenti e i file relativi ad ognuno.
* **Scenario principale**: l'utente può modificare i metadati di ogni procedimento. (**UC5.1**)

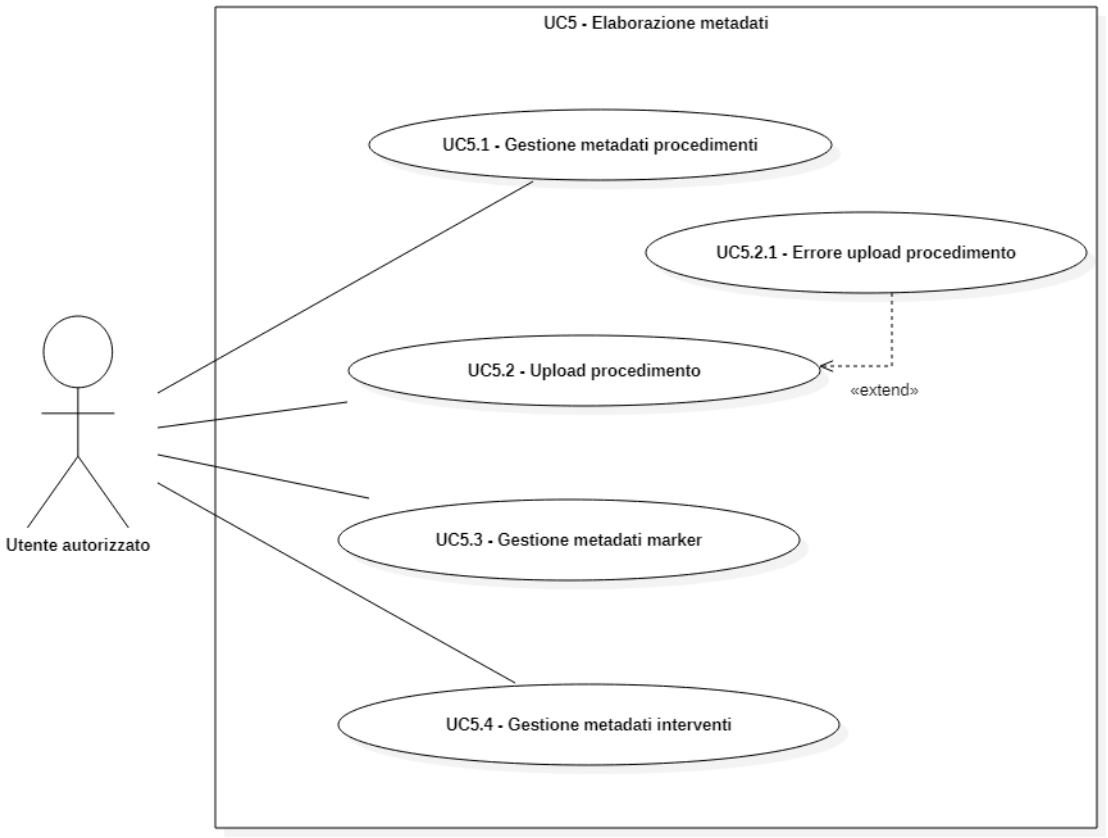


Figura 3.5: Descrizione grafica caso d'uso UC5

3.1.5 UC5 - Elaborazione metadati

* **Identificativo**: UC5
* **Nome**: elaborazione metadati
* **Descrizione grafica**:
* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente si trova sulla pagina di visualizzazione dei file con i relativi metadati.
* **Postcondizione**: l'utente ha visualizzato ed eventualmente elaborato i metadati desiderati.
* **Scenario principale**: l'utente può visualizzare i metadati relativi ad ogni file caricato.

* **Scenario secondario**: l'utente può gestire (modificare aggiungere eliminare) i procedimenti relativi ad ogni file. (**UC5.1**)

* **Scenario secondario**: l'utente può gestire (modificare aggiungere eliminare) i marker relativi ad ogni file. (**UC5.3**)

* **Scenario secondario**: l'utente può gestire (modificare aggiungere eliminare) gli interventi relativi ad ogni file. (**UC5.4**)

UC5.1 - Gestione metadati Procedimenti

* **Identificativo**: UC5.1

* **Nome**: getione metadati Procedimenti

* **Descrizione grafica**: (approfondita in UC5)

* **Attori**

  - *Primari*: utente autorizzato

* **Precondizione**: l'utente vuole elaborare procedimenti relativi ad un file.

* **Postcondizione**: l'utente ha elaborato i procedimenti.

* **Scenario principale**: l'utente può modificare, aggiungere o eliminare i procedimenti relativi ad un file.

UC5.2 - Upload procedimento

* **Identificativo**: UC5.2

* **Nome**: upload procedimento

* **Descrizione grafica**: (approfondita in UC5)

* **Attori**

  - *Primari*: utente autorizzato

* **Precondizione**: l'utente vuole fare l'upload dei metadati riguardanti un procedimento relativi ad un file.

* **Postcondizione**: l'utente ha fatto l'upload del procedimento desiderato.

* **Scenario principale**: l'utente può tramite un apposito bottone fare l'upload del file e i relativi metadati del procedimento che desidera.

* **Scenario secondario**: si è verificato un errore nella richiesta di upload del procedimento. (**UC5.2.1**)

UC5.2.1 - Errore upload procedimento

* **Identificativo**: UC5.2.1
* **Nome**: errore upload procedimento
* **Descrizione grafica**: (approfondita in UC5)
* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: il sistema non ha gestito correttamente la richiesta di upload procedimento.
* **Postcondizione**: l'errore viene visualizzato sull'applicazione.
* **Scenario principale**: la richiesta di upload procedimento non va a buon fine e l'errore viene mostrato all'utente.

UC5.3 - Gestione metadati Marker

* **Identificativo**: UC5.3
* **Nome**: getione metadati Marker
* **Descrizione grafica**: (approfondita in UC5)
* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente vuole elaborare i marker relativi ad un file.
* **Postcondizione**: l'utente ha elaborato i marker.
* **Scenario principale**: l'utente può modificare, aggiungere o eliminare i marker relativi ad un file.

UC5.4 - Gestione metadati Interventi

* **Identificativo**: UC5.4
* **Nome**: getione metadati Intervetni
* **Descrizione grafica**: (approfondita in UC5)
* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente vuole elaborare gli interventi relativi ad un file.
* **Postcondizione**: l'utente ha elaborato gli interventi.
* **Scenario principale**: l'utente può modificare, aggiungere o eliminare gli interventi relativi ad un file.

3.1.6 UC6 - Visualizzazione jobs

* **Identificativo**: UC6
* **Nome**: visualizzazione jobs
* **Descrizione grafica**:

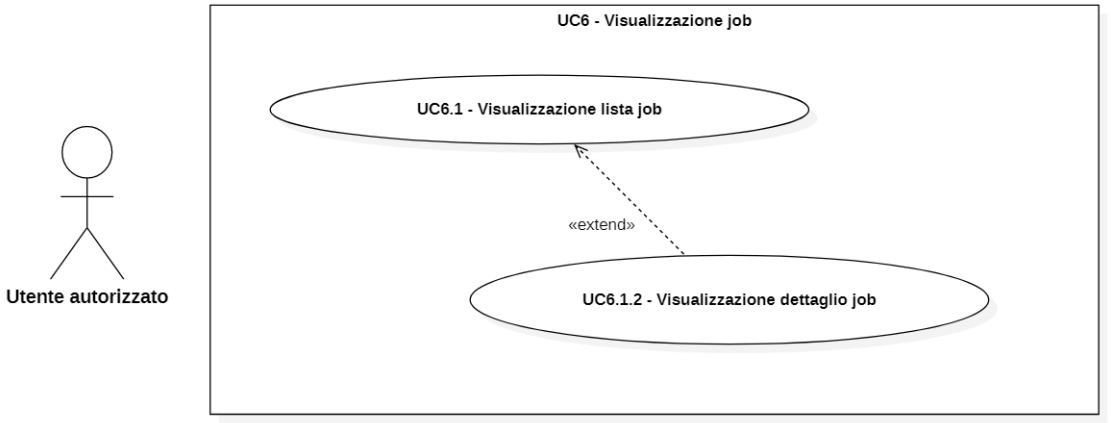


Figura 3.6: Descrizione grafica caso d'uso UC6

* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente si trova all'interno dell'applicazione e vuole visualizzare la lista dei jobs.
* **Postcondizione**: l'utente visualizza la lista dei jobs.
* **Scenario principale**: l'utente può visualizza la lista dei jobs (**UC6.1**) con un piccolo riassunto delle informazioni principali di ognuno.
* **Scenario secondario**: l'utente può visualizzare il dettaglio di un singolo job premendo sull'apposito link. (**UC6.2**)

UC6.1 - Visualizzazione lista dei job

* **Identificativo**: UC6.1
* **Nome**: visualizzazione lista dei job
* **Descrizione grafica**: (approfondita in UC6)
* **Attori**
  - *Primari*: utente autorizzato

* **Precondizione**: l'utente ha premuto sull'apposito link per visualizzare la lista dei job.
* **Postcondizione**: la lista dei job viene visualizzata.
* **Scenario principale**: l'utente visualizza la lista dei job paginata e ordinata per ultimo caricato.

UC6.1.2 - Visualizzazione dettaglio job

* **Identificativo**: UC6.2
* **Nome**: visualizzazione dettaglio job
* **Descrizione grafica**: (approfondita in UC6)
* **Attori**
  - *Primari*: utente autorizzato
* **Precondizione**: l'utente ha premuto sull'apposito link per visualizzare il dettaglio di un job.
* **Postcondizione**: il dettaglio del job viene visualizzato.
* **Scenario principale**: l'utente visualizza tutte le informazioni riguardanti il job desiderato.

3.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.
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 R(F/Q/V)(N/D/O) dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

F = facoltativo

Nelle tabelle **??**, **??** e **??** sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

Requisito	**Descrizione**	**Use Case**
RFO-1	autenticazione	UC1
RFO-2	lettura dati da CD	UC2
RFO-3	visualizzazione dati ordinati per file	UC3
RFO-4	visualizzazione dati ordinati per procedimenti	UC4
RFO-5	gestione dei Marker relativi al file	UC3
RFO-6	gestione degli Interventi relativi al file	UC3
RFD-1	upload procedimento	UC5
RFF-2	visualizzazione jobs	UC6
RQD-1	test di unità esaustivi	-
RFF-1	riproduzione audio file	UC3.1

Capitolo 4

Progettazione

Breve introduzione al capitolo

In questo capitolo vengono spiegate in modo dettagliate le tecnologie utilizzate nella realizzazione del progetto, inoltre viene data una spiegazione generale di come è stata progettata l'architettura del sistema e una più approfondita riguardante il frontend, che era oggetto del progetto di stage.

4.1 Tecnologie

Di seguito viene data una panoramica delle tecnologie utilizzate.
[[[forse posso dividere tra codice, testing, sistema, utility]]]

Javascript

JavaScript è un linguaggio di programmazione orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione applicazioni web. L'intera applicazione è stata scritta con questo linguaggio.

React

React è una libreria Javascript utilizzata per implementare interfacce utente (UI) lato frontend. React si basa sul concetto di component, idealmente è una libreria che permette di costruire i propri component come fossere degli elementi HTML del DOM per poi poterli riusare nell'intera applicazione.

Redux & Redux RTK

Redux è un contenitore dello stato per le applicazione Javascript. Viene usato per la gestione centralizzata dello stato delle applicazioni sviluppate in React Javascript. In particolare con la sua libreria Redux-Toolkit, permette una gestione dello stato semplice ed efficente.

API rest

Descrizione Tecnologia 2

JSON

Markup usato per la pubblicazione dei dati dell'API.

MUI

Material UI è una libreria React open-source che permette di implementare i Google's Material Design. Essa comprende una collezione di componenti React precostruiti che possono essere facilmente adattati e messi in uso nella UI dell'applicazione.

React Router

React Router è la libreria standard per il routing in React. Questa libreria permette la navigazione tra le varie viste dell'applicazione , permette di gestire le URL, e mantenere la sincronizzazione tra URL e viste.

Docker

Docker è un sistema che permette di facilitare il deployment delle applicazioni.

jest

(???) Descrizione Tecnologia 2

React testing library

(???) Descrizione Tecnologia 2

airbnb js guidelines

(???) Descrizione Tecnologia 2

eslint

(???)

4.2 Architettura dell'applicazione

L'architettura generale dell'applicazione era già stata progettata prima dell'inizio dello stage in oggetto, e in parte già esistente. Il sistema si divide principalmente in tre parti:

* **backend**:
* **frontend**:
* **object storage**:

La comunicazione tra le diverse parti del sistema invece avviene con l'utilizzo di **APIrest**.

[disegno generale di front end back end api minio ecc ecc]

Backend

Il backend è un'applicazione a se stante realizzata interamente in Python, con architettura MVC (Model-View-Controller). Questa architettura permette di separare completamente la logica del prodotto dal modello, le viste non sono state approfondite in quanto lo scopo principale del backend è gestire e memorizzarei dati che vengono passati dal frontend e non visualizzarli. Il backend espone delle particolari URL come endpoint per le chiamate REST del client frontend. Il backend utilizza il modello (Model) per rappresentare i dati di interesse, il controller per gestirli, e la vista (View) per rappresentarli. Nel nostro caso non dovendo rappresentare i dati, operazione che spetta al frontend, esso espone dell URL come endpoint per le chiamate REST proprio del frontend. In questo modo ogni volta che viene fatta una rechiesta su una corretta URL al backend, esso la interpreta con l'apposito controller, gestendo i dati strutturati come nel modello, e dopo aver completato la gestione della richiesta, ne restituisce la risposta al frontend.

Frontend

Il frontend dell'applicazione è generalmente la parte di interfaccia per l'utente, ossia quello che l'utente visualizza del nostro sistema e che gli da l'opportunità di interagire con il backend. Nel nostro caso il frontend non fa solo da interfaccia utente ma gestisce anche dei dati nel proprio stato fin tanto che questi rimangono in sessione, in modo da non fare continue richieste al backend che rallenterebbero molto il sistema a causa della mole dei dati da gestire e visto che non tutti i dati devono essere passati al backend. Il frontend è stato scritto in JavaScript e in particolare usando il framework React, gestendo però lo stato esternamente usando Redux, questo porta ad un'architettura leggermente più complessa dell'intero frontend ma da grandi benefici in quanto aiuta a mantenere per quanto possibile la separazione tra logica dell'applicazione e reppresentazione dei dati. [disegno frontend react + redux] In generale l'architettura è quella della single page application, cioè una pagina che non necessita di essere totalmente ricaricata ad ogni modifica ma che va a modificare soltanto la porzione interessata.

Object storage

L'object storage che si è deciso di utilizzare è minIO (scelta aziendale, legata anche ad altri progetti) e nel caso della nostra applicazione serve in particolare come stumento per memorizzare i file con delle semplici richieste tramite API. Questo ci permette essenzialmente due cose molto importanti non dover preoccuparci troppo dei dettagli implementativi della memeorizzazione e allo stesso tempo poter reperire agevolmente i file che ci servono. Per il caricamento di un file è necessario fare una semplice richiesta PUT, ad una apposita URL minIO passando il file come parametro il file che si desidera salvare. Questa URL viene concordata tra backend e minIO stesso in modo da essere sempre univoca per ogni nuovo file caricato, questo comporta che l'operazione finale di salvataggio del file comprenda tre richieste:

* **frontend -> backend**: con questa richiesta il frontend comunica quale file vuole salvare al backend che può così memorizzarne i dati per tenerne traccia qual'ora si voglia in futuro reperire;

* **backend -> minIO**: il backend richiede su un apposita URL a minIO di comunicargli la vera e propria URL univoca sulla quale fare la richiesta di salvataggio del file, dopo che l'ha ottenuta la da in risposta al frontend;
* **frontend -> minIO**: il frontend dopo aver ricevuto la corretta URL sulla quale effettuare la richiesta PUT può eseguirla passando il file come parametro

Per la versione finale del progetto non è escluso di rivedere questa parte che si prevede una delle più importanti ma allo stesso tempo difficile da eseguire in modo efficace ed efficiente, per lo stato che il prodotto deve avere alla fine dello stage invece questa gestione è più che sufficiente.

APIrest

Le API rest sono il tramite tra le due applicazioni, vengono gestite nella parte di frontend con redux-tolkit, e interrogano il backend sugli appositi endpoint, per poi lasciare al frontend l'interpretazione del risultato. Sono state utilizzate chiamate di tipo GET e POST. Inoltre una particolare chiamata di tipo PUT viene effettuata dal frontend per l'upload dei file su minIO.

4.3 Architettura frontend

Per capire nel dettaglio l'architettura del frontend ricorriamo al seguente schema di approfondimento. [disegno del frontend redux react come lavorano] L'applicazione frontend ha un architettura semplice...

Il frontend dell'applicazione è stato progettato come una single page application, che mantiene seprarazione tra lo stato e la user interface.

React

Redux

React router

Mui

4.4 Design Pattern utilizzati

(??????) tipo observer template se react lo usa... c'è una forma di MVVM in react+redux... posso citare MVC del back end o non frega niente a nessuno?

Capitolo 5

Realizzazione e testing

Breve introduzione al capitolo

5.1 Progettazione

viste implementate (jobs, proceeding view, file view), caricamento dati, upload dei dati, un po di come sono state pensate

In sede di progettazione del frontend si è deciso di realizzarlo in javascript, e in particolare usando la libreria react, sfruttando a pieno il riutilizzo del codice che questa libreria permette agevolmente costruendo componenti. Lo stato del frontend è stato gestito in modo indipendente utilizzando redux, e redux toolkit per sfruttare le API quando è necessario comunicare con il backend. Per il routing è stata utilizzata la libreria apposita react-router che permette con semplicità di gestire le url e i link all'interno dell'applicazione senza dover usare strumenti esterni al linguaggio. Graficamente invece, si è scelto di usare MUI per mantenere un interfaccia molto intuitiva, semplice da usare, responsive e con tanti elemente già pronti per essere integrati con React.

Redux permette di creare uno store, dove persistono tutti i dati che vogliamo gestire lato frontend, questo permette di non dover salvare i dati sul server e fare inutili e pesanti chimate ogni volta che si necessita di rappresentare questi dati. Lo store nel nostro caso particolare è stato usato per memorizzare lato frontend i "dati caricati dall'utente nella fase di caricamento dati da CD".

1) store 2) componenti 3) integrazione componenti 5) routing 4) viste principali

5.2 Codifica o Realizzazione???

Come ho agito in fase di codifica, cioè creazione dello store redux, creazione dell'api rtk, creazione dei vari componenti in react, integrazione dei componenti react, interazione tra componenti e stato/store test??? eslint-airbnb style guide utilizzate come standard aziendale versionamento in gitlab regola aziendale 1 modifica 1 file 1 commit (nella situazione ideale), new file and delete file tutti sullo stesso commit purchè tutti dello stesso tipo big commit iniziali o di modifiche più sostanziose 1 file 1 commit

Generalmente il flusso della codifica viene gestito in base allo sprint scrum, quindi lunedì retrospettiva su attività fatte, selezione delle attività da fare per lo scritp

settimanale e si parte. il ticket passa dallo stato "da completare" allo stato "in corso" parlando di react che induce all'uso di componenti riutilizzabili, si è cercato di sfruttare il più possibile questo principio. creazione di un nuovo componente integrazione del componente nella vista / nelle viste creazione dello store redux creazione delle api redux-toolkit

5.2.1 Store

Lo store è l'oggetto che contiene tutto lo stato dell'applicazione lato frontend, viene creato e gestito con Redux ed è completamente separato dalla user interface. Lo store nel nostro caso è molto complesso, perchè i dati possono diventare numericamente molto grandi in modo veloce e perchè la loro rappresentazione non è sempre serializzata come Redux prevede. Con le funzioni che Redux mette a disposizione la creazione e la manutenzione dello store risulta molto facile, una volta che è stato compreso come questo si comporta. Lo store rappresenta per la nostra applicazione lato frontend il punto dove vengono mantenuti tutti i dati e le operazioni che possono essere eseguite su questi. La caratteristica principale dello store è quella di essere diviso in **slice**, ovvero fette di dati, questo perchè più l'applicazione diventa grande più vinene mantenuta la divisione tra i dati. Lo stato dell'applicazione gestito con lo store può essere modificato soltanto chiamando dei particolari eventi chiamati **action** che permettono allo stato dell'applicazione di essere ricalcolato solo per le parti che convolgono le modifiche. Questo nuovo stato viene poi aggiornato automaticamente nei componenti React dopo il ricalcolo. Tutte le nostre action sono create all'interno di particolari funzioni della libreria Redux chiamate **reducer** che ricevono come parametri lo stato e l'azione e permettono il calcolo del nuovo stato. Infine per rendere disponibili i dati presenti nello store si possono creare dei **selector** che selezionano porzioni di dati dallo store e le rendono disponibili nei componenti React.

Creazione dello Store

progettato e creato come, intro slice e reducer

Lo store implementato con Redux, viene creato in un apposito file javascript all'esterno dell'applicazione React, in questo modo manteniamo la separazione, e viene passato all'applicazione in un unico punto, che per convenzione è il file index.jsx dove tutta l'applicazione risiede. Lo store viene creato con un particolare metodo di Redux che combina tutte le slice che abbiamo creato.

[[[[esempio store combine]]]] con immagine

Slice e Reducer

Una slice contiene una parte del nostro store, nel caso della nostra apllicazione si è deciso di dividere lo store in due slice una per la parte dove vengono salvati i dati da visualizzare e modificare all'interno dell'applicazione (procedimenti, registrazioi e tracce) e l'altra per la parte che gestisce le richieste tramite API al backend, per questa seconda slice è necessario l'utilizzo della libreria Redux-toolkit che ha una serie di funzioni che permette di facilitare le richieste, e di integrare le risposte nello store. Ogni slice ha i propri reducer, ovvero le funzioni che prendendo in input lo stato attuale e una action restituiscono il nuovo stato modificato secondo quello che l'azione prevede. Nel nostro caso per mantenere più ordinata possibile la gestione dello stato ogni slice presenta il proprio reducer che contiene le varie action.

[[[[esempio trialSlice]]]] con immagine

Action

Le action modificano lo stato dell'applicazione senza farlo direttamente ma demandando la vera e propria modifica dei dati a Redux, per capire meglio queste operazioni che sono il fulcro della gestione dello store ci serviamo del seguente esempio. Una delle action del nostro store è quella che prevede il caricamento dei dati, un .....

[[[[Esempio load procedimenti]]]] con immagine per ogni reducer le sue action, quali ho fatto importanti e perchè (loadcron)

Selector

Per reperire una parte di dati come per esempio un array di registrazioni o un singolo procedimento all'interno del nostro store abbiamo bisogno dei selector, questi sono delle piccole funzioni che fanno un'azione di filtro dell'intero store restituendo solo quello che ci serve.

[[[[Esempio getProcedimenti]]]] con immagine selettori perchè, i più efficaci e quali usare dove

5.2.2 Component

La scelta di react in fase di analisi del progettoci permette di utilizzare i component tipici di questa libreria javascript. Il concetto di component è molto intuitivo quando ci si approccia ad usare React, essi sono come delle funzioni javascript che accettano in input dei particolari dati chiamati **props** e ritornano elementi React che descrivono cosa si dovrà vedere sullo schermo. Usando i components possiamo facilmente dividere la user interface in piccole parti indipendenti tra loro e riusabili, in modo da doverci occupare di una parte alla volta pensando solo a quella in modo isolato prima di integrarla con il resto della UI. I component secondo React possono essere di due tipi class component o function component, nel nostro caso abbiamo scelto di costruire tutti function component principalmente per avere un codice più leggibile e perchè non c'è la necessità di avere i vantaggi dei class component (per esempio non ci serve nessun metodo costruttore all'interno dei component che abbiamo creato). I component possono quindi essere qualsiasi parte della user interface vogliamo realizzare, da un bottone a un form oppure un intera vista. Sviluppando la nostra applicazione abbiamo pensato inizialmente a i component più "esterni", ciè quelli che fanno da contenitore agli altri, andando poi a raffinare sempre di più fino a i component più "interni", ossia quelli riguardanti le parti più piccole dell'applicazione, applicando in questo modo una sorta di metodo top-down per lo sviluppo delle componenti della nostra applicazione. Una vola che abbiamo creato i vari component, utilizzando questo modello top-down, siamo passati all'integrazione tra i vari componenti che servivano per realizzare la completa user interface. React permette molto agevolmente, proprio come punto di forza nell'uso di questa libreria, l'integrazione tra i vari componenti. Basta importare nella pagina il componente che ci serve, richiamarlo nel codice come se fosse un normale tag HTML e passargli le props di cui ha bisogno, non dobbiamo preoccuparci di altro, React svolgerà per noi il corretto rendering dell'insieme dei componenti. [[[[fileView codice]]]] Vediamo nel dettaglio come viene reralizzato un component nella nostra applicazione, nello specifico il component fileView, questo realizza un intera vista che si occupa di visualizzare la lista di tutti i file dopo che sono stati caricati dall'utente. I vari import spiegati [righe da a] Questo componente non necessita di alcuna props ma si serve di uno dei selector Redux che abbiamo creato per reperire la lista dei file di cui abbiamo bisogno dallo store. [righe da a] La lista dei file recuperata grazie all'apposito

selector Redux viene snocciolata con la funzione nativa javascript map [riga boh], che permette di eseguire delle operazioni per ogni oggetto della lista, nel nostro caso per ogni file. In questo component per ogni file si desidera richiamare altri tre component che si occuperanno di alcune parti della user interface che riguarda ogni file, e sono:

* AudioPlayer: che provvede a fare il render di un player per riprodurre il file
* ProceedingTabs: tramite una tendina permette di visualizzare e gestire i metadati relativi al file (Marker e Interventi)
* ProceedingList: recupera e visualizza tutti i procedimenti relativi a questo file

[[[[fileView vista renderizzata]]]]

5.2.3 Hooks

Gli Hook sono delle funzioni javascript che permettono di entrare nel ciclo usato da React per renderizzare i componenti, e in questo modo descriverne il comportamento. Queste funzioni non possono essere usate nei componenti di tipo classe e hanno delle regole di invocazione abbastanza rigide, per proteggere lo stato dell'applicazione e il ciclo di render. Gli hook infatti non possono essere chiamati all'interno di cicli, istruzioni condizionali o funzioni che non siano componenti react. La libreria React mette a disposizione degli Hook generici che risultano molto utili per gestire il rendering dei componenti, ma se qusti non dovessero bastare possiamo anche crearne di personalizzati. Di seguito vediamo un hook appositamente creato per recuperare dal backend la lista dei jobs, è stato creato per essere riusato qual'ora si rendesse necessario da un altra parte dell'applicazione ma anche perchè sfruttando le proprietà del linguaggio e in particolare di queste speciali funzioni possiamo facilmente gestire il filtraggio e la paginazione della lista dei jobs.

[[[[useJobResult]]]]

5.2.4 Routing

[[[[Ha senso questa sezione???]]]]

5.2.5 UI style

Lo sviluppo della parte grafica, in particolare per quanto riguarda lo stile dell'applicazione, non aveva nessun vincolo quindi si è scelta una soluzione che possa essere intuitiva, veloce da implementare e generalmente riconosciuta. La scelta è ricaduta su **MUI - Material UI** che è una libreria open-source di componenti React già precostruiti seguendo le regolole di Google Material Design. Material è un sistema di design creato da Google per velocizzare lo sviluppo dei componenti. MUI permette di non doversi preoccupare di aspetti chiave quali responsive-design, integrazione grafica dei nuovi componenti, stile generale dell'applicazione, perchè tutti questi aspetti vengono automatizzati usando stili globali e componenti già preimpostati. [[[[configuarzione di un componente esempio grafico - ProceedingCard]]]] In questo esempio preso direttamente da uno dei componenti della nostra applicazione, possiamo vedere vari MUI component utilizzati e personalizzati appositamente per il nostro funzionamento. Tra questi c'è per esempio **<Stack>** che costruisce una pila di sottocomponenti, ma che possono essere direzionati in colonna o in righa (come nel nostro caso). Altri utilizzi sono **<FormControl>** e **<TextField>** che costruiscono una porzione di un form con un campo di tipo testo al suo interno. [[[[risultato del componente - ProceedingCard]]]]

5.2.6 Viste principali

file view, proceeding view w jobs
[[[[Ha senso questa sezione???]]]]

5.3 Test o Testing???

Parte importante del nostro processo di sviluppo, sono i test. Secondo regola aziendale ma anche seguendo il piano di progetto dello stage è stata pianificata una buona parte di ore di testing. Le ore di testing sono state ripartite in modo proporzionale alle ore di sviluppo di ogni funzionalità. Nella fase iniziale di studio delle tecnologie da utilizzare per il progetto sono state incluse anche vaerie librerie per i test su funzioni javascript e componenti React che sono la grande parte del prodotto da realizzare. Dopo un confronto con il tutor si è deciso di intraprendere la strada di **React-testing-library** per il testing dei componenti e appoggiargi a **jest** per la parte javascript dei test.

5.3.1 Test javascript function

Il framework jest è uno dei più diffusi per il test di applicazioni javascript, è focalizzato sulla semplicità nella scrittura dei test e risulta comunque in grado di testare tutte le parti di nostro interesse. All'interno dei nostri test è stato usato in supporto a React-testing-library per il testing dei componenti React e in prima battuta per il test di tutte le utility javascript che abbiamo creato, in particolare si è reso molto importante nel testing delle funzioni che leggono i metadati caricati dall'utente e li manipolano per renderli usabili per l'applicazione. Il funzionamento di jest è molto semplice per i nostri casi, ma può essere esteso a molti altri tipi di test. Per noi è semplice e molto intuitivo creare un test javascript con questo framework, basta creare le condizioni per l'utilizzo di una determinata funzione che vogliamo testare, chimarla come se dovessimo realmente utilizzarla e vedere se i risultati in usicta da questa chiamata sono quelli che ci aspettiamo. [[[esempio di test tipo cronologia o funzioni di ordinamento]]] Nella prima parte del test come si può vedere abbiamo creato i dati fittizzi che ci servono da passare alla funzion da testare (in questo caso [[[nome funzione]]]), in seguito abbiamo effettuato la chiamata di funzione e per concludere il test abbiamo scritto delle particolari funzioni (nomi reali tipo expect mi sembra) con le quali andare a vedere se i risultati ottenuti sono quelli che ci aspettiamo passando i nostri dati fittizzi.

5.3.2 Test React component

La libreria usata è React-testing-library, questa si appoggia completamente su un altra libreria di testing chiamata DOM-testing-library e fornisce una abbastanza completa suite di testing per i componenti React. [[[Prima devo dire che si basa sul testare il DOM]]] Il principio che guida questa libreria e anche i nostri test sui componenti è che i test di integrazione hanno maggior valore degli unit test nel caso di applicazioni React. In particolare con queste librerie (React-testing e jest) usate in modo combinato si creano test su parti di applicazione dopo il rendering, ovvero la logica dei nostri test è quella di andare a verificare funzionamenti e presenza di elementi nel DOM dopo che è stato fatto il rendering dell'applicazione. Questo tipo di test ovviamente è un test di integrazione in quanto non va a testare singolarmente ogni componente e ogni funzionalità del componente ma esegue un test della UI dal punto di vista

dell'utente, cioè controlliamo che l'applicazione in ogni momento si comporti come abbiamo progettato che lo faccia. La libreria prevede di descrivere un test passando ad un apposito metodo il componente di cui fare il render, e una volta che questo è stato completato con altri metodi specifici si va a controllare cosa è nel DOM. Altre situazioni possono esssere sottoposte a test in questo modo, come scatenare in modo controllato un evento (per esempio il click su un bottone) e vedere se il DOM si modifica come ci si aspetta. Ci sono molte altre possibilità per casi più particolari come per testare chiamate ad API, nella nostra applicazione però questo tipo di test non è stato implementato per motivi di tempo e in accordo con l'azienda che prevede di mettere appunto una specifica suite di test per questo ambito per vari prodotti che segue. [[[[esempio di test]]]] Nel nostro piccolo esempio che abbiamo riportato qui di test di un componente si può vedere come sia facile e veloce implementare un test di questo tipo, ma allo stesso tempo come verifichi esattamente quello che ci aspettiamo dalla nostra applicazione. In questo caso dopo il rendering del nostro componente testato [[nome componente]] andiamo a controllare se nel DOM sono presenti gli elementi che ci aspettiamo [[nome elementi oggetto del test]].

5.3.3 Redux store, ho qualcosa???

[[[[come femo???]]]]

Capitolo 6

Conclusioni

6.1 Consuntiovo finale

tabella cronologica dello stage e valutazione rispetto al gantt iniziale esempio subsection

Settimana		**Attività**
1	dal 17/10/2022 al 25/10/2022	autenticazione mediante server remoto
2	dal 17/10/2022 al 25/10/2022	lettura dati da CD
3	dal 17/10/2022 al 25/10/2022	precompilazione di form con i dati caricati da CD
4	dal 17/10/2022 al 25/10/2022	editing dei dati del form
5	dal 17/10/2022 al 25/10/2022	upload dei dati verso i sistemi esterni
6	dal 17/10/2022 al 25/10/2022	test di unità esaustivi
7	dal 17/10/2022 al 25/10/2022	possibilità di ascoltare le registrazioni
8	dal 17/10/2022 al 25/10/2022	possibilità di modificare i dati mediante interazioni evolute (per es. drag-n-drop)
9	dal 17/10/2022 al 25/10/2022	realizzazione di un'applicazione desktop con Electron
10	dal 17/10/2022 al 25/10/2022	compilazione multipiattaforma dell'applicazione desktop

Orario

Sviluppo

6.2 Raggiungimento degli obiettivi

tabella requisiti stage superati e non

Codice	Descrizione	Stato
O01	autenticazione mediante server remoto	soddisfatto
O02	lettura dati da CD	soddisfatto
O03	precompilazione di form con i dati caricati da CD	soddisfatto
O04	editing dei dati del form	soddisfatto
D01	upload dei dati verso i sistemi esterni	soddisfatto
D02	test di unità esaustivi	soddisfatto
F01	possibilità di ascoltare le registrazioni	soddisfatto
F02	possibilità di modificare i dati mediante interazioni evolute (per es. drag-n-drop)	non soddisfatto
F03	realizzazione di un'applicazione desktop con Electron	non soddisfatto
F04	compilazione multipiattaforma dell'applicazione desktop	non soddisfatto

6.3 Conoscenze acquisite

dove ho acquisito conoscenze s git, programmazione js, vari react ecc ecc, metodologie di lavoro

6.4 Valutazione finale

cazzate riguardo quanto è utile lo stage, come mi sono trovato con azienda tutor ecc ecc, valutazione critica delle conoscenze acquisite delle tempistiche

Bibliografia