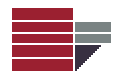
**Università della Calabria**

**Dipartimento di Matematica e Informatica**



**Corso di Laurea Triennale/Magistrale in Informatica**

Tesi di Laurea

Estensioni avanzate di un IDE web per formalismi logici

|  |  |  |
| --- | --- | --- |
|  | Relatrici/Relatori | Candidata/o |
|  | Prof.ssa/Prof. Calimeri Francesco | Carmine Blefari |
|  | Prof.ssa/Prof. Nome Cognome | Matricola 213535 |

Anno Accademico 2019/2020

INDICE

**Introduzione** …………………………………………………………………………. 3

**Capitolo 1** First Time Tutorial …………………………………………………….4

**Capitolo 2** Examples ………………………………………………………………

**Capitolo 3** Editor Collaborativo …………………………………………………..

**Conclusioni** …………………………………………………………………………..

**Ringraziamenti** ………………………………………………………………………

**Bibliografia** …………………………………………………………………………..

INTRODUZIONE

Il lavoro di tesi svolto presso il dipartimento di Informatica dell’Università della Calabria nasce con l’idea  di apportare modifiche al progetto LoIde , un web-based IDEE per la programmazione logica.

Il lavoro centrale è stato svolto sull’Editor che mette a disposizione LoIde ed il linguaggio principalmente utilizzato è TypeScript.

Gli obiettivi principali della tesi fin dall’inizio sono stati quelli di imparare a contribuire su un medio/grosso progetto e relazionarsi con altri contributori e colleghi attraverso il servizio GitHub e Trello.

Quindi è stato necessario imparare a lavorare su nuovi software con l’intento di interfacciarsi al mondo del lavoro aggiungendo al proprio curriculum quelle soft skills che si apprendono solo con l’esperienza e la costanza.

La necessità di apportare modifiche al progetto si sviluppa dopo uno studio delle caratteristiche principali dell’IDE volto non ad aggiungere futili funzionalità per il solo intento di riempire l’Editor, ma a causa di alcune caratteristiche che possano soddisfare qualsiasi tipologia di utente e andare di pari passo con gli Editor più utilizzati.

Fin da subito ci si è accorti che il primo approccio all’Editor risulta molto banale e disorganizzato, e che non erano presenti modalità di collaborazione che ormai risultano essere necessarie tra le più svariate tipologie di utenti.

Le modifiche principali apportate al progetto LoIde riguardano :

* Un tutorial iniziale che rendesse più interattiva l’esperienza di un nuovo utente interfacciandosi la prima volta con l’editor , e che implementasse una sorta di guida conoscitiva di quest’ultimo.
* La funzionalità degli esempi , in modo tale da mettere a disposizione dell’utente nell’Editor principale codice già pronto da cui prendere esempio e con il quale avere un primo impatto sulle funzionalità principali dell’Editor.
* Una funzionalità collaborativa , che permettesse ad un utente di lavorare sullo stesso progetto in contemporanea con una o più persone.

In questo lavoro di tesi andremo a vedere nel dettaglio i seguenti capitoli :

* Capitolo 1 :  First Time Tutorial
* Capitolo 2 :  Examples
* Capitolo 3 : Editor Collaborativo

Concludo l’introduzione affermando l’utilità di una tesi al fine di immedesimarsi in un sistema lavorativo già con una struttura non banale , poiché dalle difficoltà iniziali a causa della scarsa conoscenza di quelle soft skills sopra citate si arriva ad un punto finale in cui si è pronti per relazionarsi al mondo del lavoro.

1) FIRST TIME TUTORIAL

Il First Time Tutorial nasce con l’idea di spiegare agli utenti come funziona LoIde , attraverso un metodo di interazione e un linguaggio semplice e comprensibile.

Un Editor come LoIde è un programma usato per creare file di testo e modificarne il contenuto , di conseguenze le funzionalità offerte sono molteplici e quindi è importante considerare l’eventualità che gli utenti possano interpretare queste funzionalità in maniera scorretta e differente , o utilizzarle per scopi che non sono conformi ai motivi che hanno generato la necessità di importare una determinata funzionalità nel programma.

Diventa necessario offrire agli utenti le motivazioni per le quali un team ha aggiunto una funzionalità e indicare all’utente l’intenzione generale di quest’ultima.

Quindi, dopo un lavoro di ricerca su altri Editor già esistenti e dopo una serie di condivisioni di idee con gli altri contributori al progetto , si è deciso che un First Time Tutorial era un qualcosa di necessario al fine di accogliere qualcuno sull’Editor nella miglior maniera possibile.

In questo periodo di ricerca sono stati considerati Editor che offrissero molte funzionalità aggiuntive di quelle offerte da LoIde , o nel caso contrario , Editor molto più semplici e intuitivi.

Ma nel complesso, immedesimandosi nei panni di un utente che affronta per la prima volta un Editor , sono sempre diversi i dubbi riguardo alcune caratteristiche.

Di conseguenza si cerca di andare per intuizione, o di fare mente locale su Editor visti in precedenza.

L’intento di LoIde è quello di evitare di creare confusione in un utente.

Durante la fase di analisi di questa funzionalità sono state considerate le più svariate possibilità di implementazione.

Infatti è stato possibile constatare durante questo lavoro di tesi che l’idea generale fissata inizialmente può cambiare in qualsiasi momento, in seguito al dialogo e al confronto con i membri del team di sviluppo, ma anche grazie a constatazioni palesi dopo tentativi di approccio falliti.

Il problema principale riscontrato durante l’implementazione di questa funzionalità non è stato riscontrato nello sviluppo di quest’ultima, bensì nella fase iniziale.

Il First Time Tutorial inizialmente è stato immagino come inseribile all’interno di LoIde attraverso un video tutorial iniziale, che avrebbe spiegato nella pratica di pochi secondi a video quale è il ruolo delle funzionalità offerte dall’editor.

Tuttavia questo risultava essere troppo arretrato con l’idea portata avanti dal progetto LoIde e tuttavia un pò pesante per un utente.

Si è cercato quindi di pensare ad un metodo che coinvolgesse l’utente nell’utilizzo di questa funzionalità, anche attraverso semplici passaggi, ma che almeno non gli designasse un ruolo marginale.

Alla fine tutte le esigenze e i dubbi sorti sono stati annullati attraverso un Tool particolare che soddisfava tutti.

La funzionalità è stata implementata con l’intento di offrire uno step-by-step tutorial che fosse il più interattivo possibile.

Da qui nasce l’esigenza di introdurre nel progetto il Tool Intro.js per creare un vero e proprio tour step-by-step all’utente.

La scelta di Intro.js non è stata banale , ma dettata anche dalla sua facilità di utilizzo , dalla possibilità di customizzare il prodotto finale in base alle esigenze e infine poiché conforme alla politica di utilizzo e commercializzazione di LoIde.

Per ogni step è stato possibile mettere a disposizione una descrizione delle singole funzionalità e abilitare il prosieguo di questo tutorial potendo tornare indietro anche in caso fosse sfuggito qualcosa.

Alla fine si dà la scelta di non mostrare più questa funzionalità e proseguire nel programming.

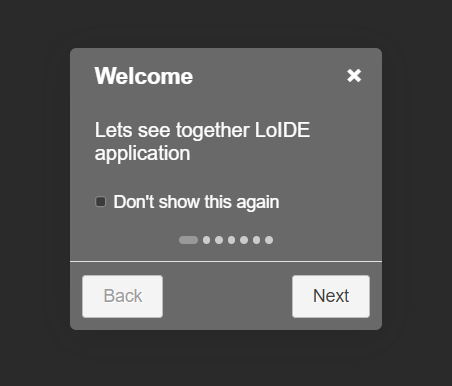
Per l’implementazione del First Time Tutorial è stato necessario introdurre il package Intro.js nel progetto.

Una volta importato il tutto si è passati a modellare il le funzionalità di Intro.js in risposta alle nostre necessità  :

* Lingua Inglese , scelta per allargare l’utilizzo di LoIde ad un numero più alto possibile di utenti.
* Accogliere l’utente , mostrando un primo riquadro di benvenuto e spiegando quali saranno i passi successivi.
* Identificare gli oggetti compresi nel tutorial , associando così nuovi id ai diretti interessati per richiamarli nell’implementazione del Tool.
* Inserire un cookie per evitare che questo tutorial venga mostrato ad ogni accesso a LoIde e dare la possibilità all’utente di non ripetere più il tutorial.

Lo stile del tutorial è stato modificato al fine di offrire un'interfaccia leggera e piacevole al cliente : per questo è stato necessario modificare il foglio di stile offerto dal Tool in base alle nostre esigenze.

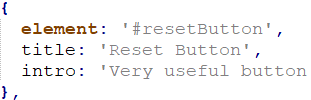
Alla fine il risultato ottenuto è il seguente :



Come si può notare nell’immagine è presente una check box con la voce ‘Don’t show this again’.

Questo ci permette di offrire all’utente la possibilità di abilitare questa voce una volta ci si senta sicuri di aver capito le funzionalità dell’Editor, così facendo si evita che il tutorial implementato venga presentato ogni volta che si acceda a quest’ultimo.

Per assegnare tutti gli attributi necessari ad ogni step del tutorial si è passati all’implementazione della logica del Tool nel seguente modo:



Ovvero richiamando l’oggetto a cui riferisce lo step corrente del tutorial , e assegnandogli un titolo e un corpo di testo.

Per richiedere l’eventuale implementazione del tutorial implementato , il parere del resto del team di sviluppo e la correzione di eventuali errori , è stata creata su GitHub una pull request a riguardo.



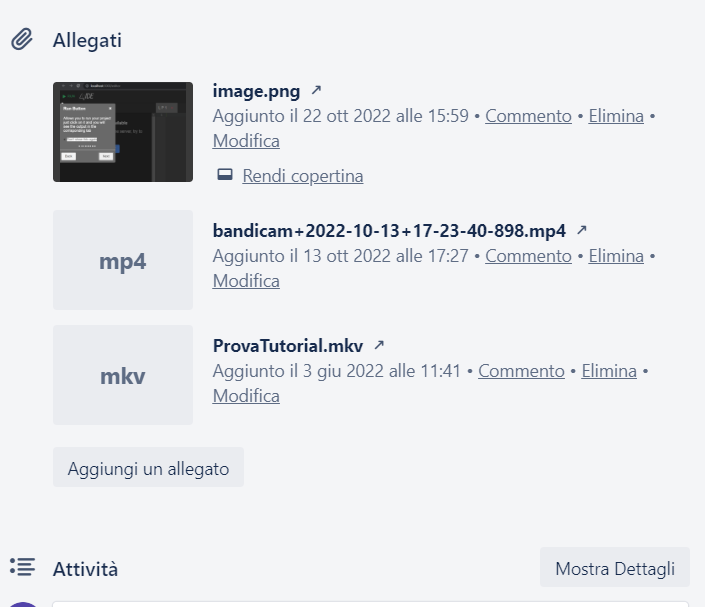
Una skill chiave appresa durante lo sviluppo di questa funzionalità la si è raggiunti attraverso la pulizia del codice.

Grazie all’esperienza e alla conoscenza dei relatori è stato importante capire che bisogna evitare di appesantire il codice di una singola componente inserendo tutti gli elementi fondamentali che contribuiscono allo sviluppo di una funzionalità, bensì organizzare il lavoro in cartelle, con nomi che mandassero all’utilità della corrispondente, al fine di avere un progetto ben organizzato, pulito e semplice da intraprendere da un occhio esterno.

Un esempio palese è stato quello della definizione degli steps del First Time Tutorial, che prevede una funzione che richiama gli oggetti su da attribuire ad ogni step del tutorial, e che quindi risulta essere ripetitiva e lunga.

Lasciare che questa parte contribuisca a comporre il codice di una componente solo perché la funzionalità introdotta appartiene ad esso è ingombrante, quindi è stato creato un file .js (JavaScript) a parte all’interno del quale è presente questa funzione.

Così facendo si sfruttano al meglio le potenzialità del TypeScript e una volta importata la funzione la si può utilizzare come se facesse parte della componente stessa.



Come si può vedere da questo screenshot di Trello, anche la cura del design del First Time Tutorial non è stata casuale.

Importante è stato il confronto con il team di sviluppo affinché la funzionalità fosse implementata seguendo lo stile già applicato all’Editor.

Anche per questo è diventato indispensabile il supporto offerto da Trello, un software gestionale che ha consentito di avere una propria area personale dedicata al proprio contributo offerto al progetto LoIde e che ha facilitato la comunicazione e lo scambio di idee con tutti i membri del team.

L’area personale di Trello offre la possibilità di creare nuove liste, all’interno di possono aggiungere delle card relative alle funzionalità da implementare.

Per il lavoro di tesi svolto le liste create sono: ToDo, Doing, Done.

Questo ha reso più facile lato tesista avere un lavoro organizzato nel migliore dei modi, e lato relatore comprendere più facilmente i progressi.

Un ulteriore elemento di supporto che ha agevolato lo sviluppo di questa funzionalità è stato GitHub,un servizio di hosting per progetti software che ha facilitato a differenza di Trello le discussioni e la risoluzione di problemi relativi al codice.



In questa fase infatti sono state effettuate delle pull request come si può vedere nell’immagine sopra.

Queste erano indirizzate al repository GitHub relativo al progetto ufficiale di LoIde, in seguito alle quali i membri del team, in particolare i miei relatori, hanno potuto constatare e analizzare il codice da me scritto.

All’interno di GitHub sono state aperte anche discussioni relative ai problemi presenti nel lavoro presentato e alle possibili operazioni di modifica e correzione.

2) EXAMPLES

Una delle operazioni più diffuse tra gli utenti durante le loro prime esperienze su un Editor è quella di eseguire del codice semplice e intuitivo, spesso associato a esperienze della vita reale oppure a esempi comprensibili facilmente.

L’utente cerca di comprendere l’Editor o un nuovo linguaggio che sta utilizzando non approfondendo la documentazione ad esso associata, ma avendo un responso immediato ad una idea da lui concepita o, come nel nostro caso, a esempi già pronti da eseguire.

Studiando gli Editor già diffusi e analizzando soprattutto quelli che più rispecchiano LoIde, le sue funzionalità e il suo scopo, è subito sorto il dubbio se una funzionalità di esempi già presenti e pronti all’esecuzione fosse necessaria e sufficiente.

La risposta ottenuta dalle diverse implementazioni di questa idea e da un’immersione lato utente nell’Editor LoIde è stata positiva.

Nasce così l’idea di Examples, una semplice funzionalità posta sull’Editor che permette all’utente di scegliere uno tra gli esempi proposti, visualizzarne il contenuto ed eseguirlo correttamente.

Alla base della scelta implementativa sussiste l’idea di rispecchiare lo stile già presente su LoIde : ogni scelta di design e di funzionalità associata ad essa si sposa perfettamente con le altre funzionalità già presenti al fine di non alterare la familiarità del progetto e sfruttare al meglio ciò che si ha a disposizione.

Per implementare la funzionalità Examples ovviamente non si è partiti dall’idea finale ben impostata, ma è stato necessario sciogliere alcuni dubbi ed effettuare prove che poi si sono rivelate insufficienti a risolvere l’intero problema.

L’idea iniziale era quella di creare una nuova Tab che si presentasse allo stesso modo del risultato finale poi raggiunto e successivamente sfruttare dei file già presenti nel nostro progetto per copiarli su un posto di comodo, in questo caso la Navigator.clipboard e poi permettere all’utente tramite un pulsante presente nella pagina principale dell’Editor di richiamare la funzione associata e visualizzare l’esempio scelto.

Come si può ben notare però lo sviluppo di questa opzione risulta essere molto complicato, non lineare con le idee già presenti su LoIde e forse troppo confusionaria per un utente che magari potrebbe decidere di non affidarsi ad essa proprio per queste motivazioni.

Diventa quindi fondamentale capire come risparmiare operazioni al fine di ottenere un risultato anche più pulito dal punto di vista implementativo, e anche come rendere la funzionalità più accessibile ad un utente e più semplice da utilizzare.

Si decide così di basarsi sulle idee già presenti nell'Editor e di proseguire utilizzando il modello applicato nelle operazioni già presenti.

Dal punto di vista del design della funzionalità si abbatte la creazione di un’altra Tab e si passa a crea un nuovo popup nella pagina principale, mentre dal punto di vista di implementazione codice si evita di utilizzare un semplice copia e incolla ma si utilizza la funzionalità dell’Editor che permette di cambiare il valore della Tab corrente passando l’indice che riconosce su quale Tab stiamo lavorando e il nuovo valore da visualizzare su quest’ultima.

Per quanto riguarda il primo punto quindi si è passati a creare un nuovo modello, ovvero un nuovo componente che funziona come un popup che non va ad aumentare il carico finale del progetto e non va ad appesantire l’interfaccia con una nuova pagina.

Dai risultati inoltre risulta più soddisfacente e comodo per un utente utilizzare questa modalità piuttosto che recarsi su un’altra pagina ed eseguire le stesse operazioni.

Per aprire questo popup si utilizzano gli State di react, ovvero attraverso lo stato di alcune variabili si può decidere se eseguire l’operazione di apertura e chiusura.

Inoltre per essere coerenti con l’intero progetto è stato necessario abilitare anche l’operazione di utilizzo associata ad Examples tramite comandi di tastiera.

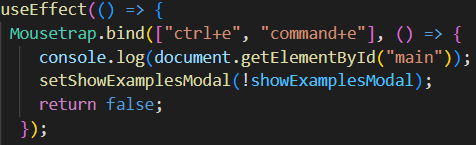
Un ulteriore incentivo che ha spinto all'incorporata di questa funzionalità nell’Editor principale è rappresentato dal secondo punto espresso sopra.

Con una nuova pagina non si poteva accedere direttamente alle funzioni dell’Editor ma si dovevano o implementare nuove funzioni, o duplicare funzioni già presenti.

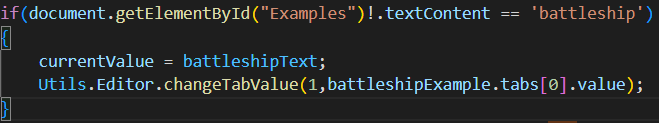
In questa maniera invece si agisce direttamente sul diretto interessato.

Il valore lo si ottiene importando i file .json (JSON) che sono già presenti nel progetto.

Ecco le scelte implementative adottate.



Di seguito mostro un esempio della funzione utilizzata.



In questo piccolo esempio dimostrativo si può vedere come quando la scelta dell’utente ricade sul gioco Battaglia Navale, si procede a cambiare il valore dell’area dedicata alla descrizione e il valore dell’Editor 1 in questo caso con il contenuto del file .json battleshipExample.

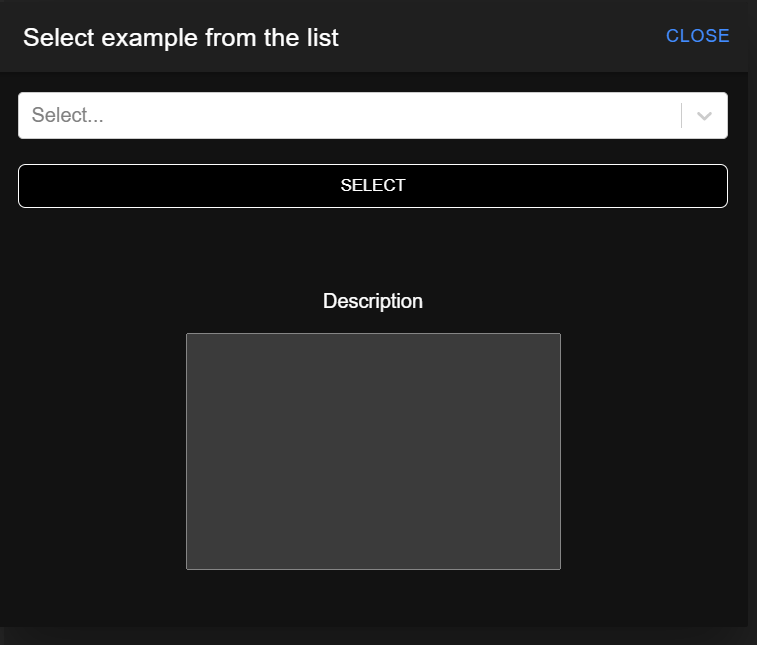
Anche la scelta degli esempi da proporre agli utenti non è stata casuale, ma si basa sulle esperienze portate sul campo dalle personali esperienze universitarie.

Presentare esempi di grandi dimensioni, con complessità elevate o che prevedono l’uso di funzioni di linguaggio avanzante non è lo scopo che vuole raggiungere la funzionalità: l’obiettivo rimane offrire all’utente una marcia in più nei primi momenti di approccio all’Editor, un piccolo reminder su semplici esempi di codice che in un momento qualsiasi può sfuggire.

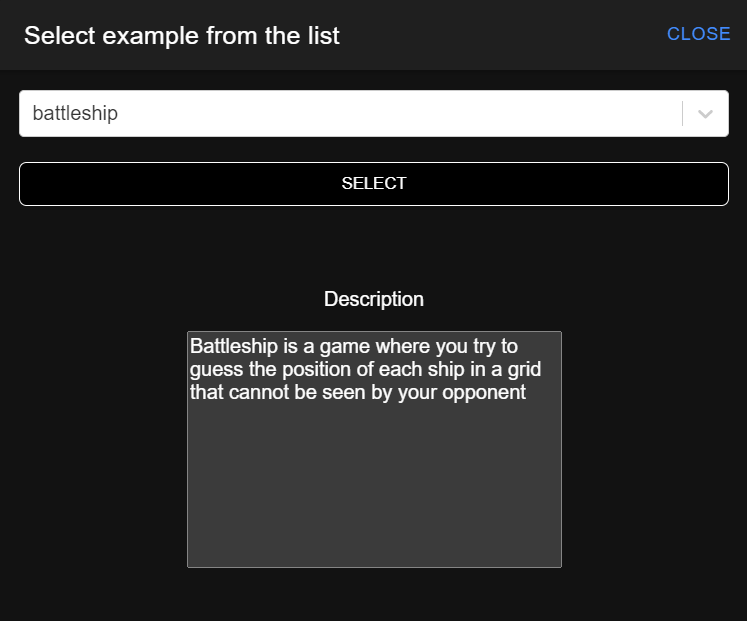
Ecco perché gli esempi sono molto semplici da comprendere e replicare con altre problematiche di codice che possono presentarsi durante il lavoro su progetti

La funzionalità si presenta su un popup che si apre cliccando sul pulsante aggiunto alla pagina principale dell’Editor.

Ecco un’anteprima di ciò che appare all’utente.



Successivamente a seguito della scelta.



Per implementare questa funzionalità si è sfruttati un componente di React risultato essere molto utili : Select.

Grazie al Select si offre la possibilità all’utente di scegliere uno tra i diversi esempi di codice già presenti in fase di accesso all’Editor.

Una volta selezionato un esempio viene caricato un file .js tramite un funzione e mostrato all’interno del riquadro “Description” : questo permette di offrire una piccola descrizione dell’esempio scelto prima ancora di eseguirlo.

Successivamente nell’Editor viene caricato il codice associato all’esempio; tuttavia è sempre possibile scegliere in questa fase un altro esempio.

Nel caso in cui si scelga un altro esempio da mostrare o nel caso in cui sull’Editor era presente già del codice questo viene sostituito con il nuovo esempio scelto, tutto attraverso la funzione utilizzata.

3) EDITOR COLLABORATIVO

La necessità di implementare un funzionalità collaborativa nel progetto LoIde nasce dopo un’attenta disamina delle modalità di studio tutt’oggi utilizzate ma anche dall’esperienza personale portata sul campo.

Utilizzando l’editor di LoIde sia in veste di studente che da implementatore si può notare come le sedute di coding siano spesso accompagnate da momenti di confronto e condivisione con altri colleghi, amici etc..

Anche per questo motivo quindi diventa utile avere un mezzo di collaborazione diretta che non va cercato al di fuori del contesto LoIde, ma è esso stesso che lo diventa.

Le forme di collaborazione sul codice possono essere le più varie, ma quando si ha a disposizione una funzionalità che permette a due o più utilizzatori dell’editor di lavorare in simbiosi e contemporanea sullo stesso progetto, tutto diventa più produttivo e soddisfacente.

Si può pensare anche a casi di utenti che non siano in grado di risolvere momentaneamente pezzi di codici, e in preda al dubbio e alla disperazione cercano aiuto online anche con scarsi risultati.

Diventa quindi compito e responsabilità di LoIde evitare di stimolare queste reazioni nell’utente e di presentare una risposta , in parte , al problema.

Certo, un editor collaborativo non rappresenta l’innovazione maggiore per ogni difficoltà di scrittura di codice, ma anche dall’esperienza portata in campo da altri editor si percepisce l’esigenza sviluppata dagli utenti di collaborare per ottenere risultati migliori e precoci.

Infine un tassello importante che ha spinto alla scelta di un editor collaborativo è stato la crescita esponenziale del lavoro da casa.

La combinazione di lavoro in loco e lavoro da remoto necessità di un mezzo che permetta ad un team di poter continuare a lavorare insieme.

Lo sviluppo di una funzionalità collaborativa all’interno di un editor può essere supportata dalla presenza di framework già esistenti, o in caso contrario di una modalità di collaborazione basata su un sistema interno.

Nel progetto LoIde è stato implementato un editor collaborativo attraverso l’uso del framework Fluid Framework.

La scelta tra le due nell’implementazione è stata dettata dalle risposte più che positive che Fluid Framework ha saputo dare nel presentare i suoi prodotti.

In primis si può notare che la scelta da parte di Microsoft di rilasciare il framework in maniera open source è stata sicuramente vincente e invitante.

Gli sviluppatori infatti hanno adottato subito questo metodo per creare contenuti interattivi e multi utenti.

Successivamente, un vantaggio principale è che il Fluid Framework permette di lavorare su documenti indipendenti : creare nuovi progetti e iniziare la collaborazione su ognuno di essi.

Con Fluid Framework i documenti non devono più essere passati avanti e indietro tra colleghi, ma si sfrutta la componente create e qualsiasi modifica si apporta verrà visualizzata.

Il primo passaggio per l’implementazione di questa funzionalità è stato creare una nuova pagina, denominata Collaborative Editor, che mette a disposizione dell’utente un nuovo Editor indipendente da quello principale utilizzato nella Main Tab ma che nonostante ciò si basa sullo stesso componente.

Necessario è stata la rivisitazione delle modalità di utilizzo dell’Editor Collaborativo e dei suoi valori iniziali pre impostati in fase di dichiarazione.

Il primo ostacolo incontrato nell’implementazione dell’Editor Collaborativo è rappresentato dalla tipologia di modello adottata da Fluid Framework, ovvero quello client server.

Il modello corrisponde perfettamente a ciò che il progetto LoIde aspirava, ma si basa su un client che viene eseguito localmente sull’Editor in utilizzo, e su un server invece, come esplicitato nella documentazione associata, che va eseguito tramite un’altra shell.

Questo presentava un problema per LoIde che invece vuole offrire all’utente un prodotto già finito, pronto all’utilizzo e senza necessità di coinvolgere l’utente in parti esterne a LoIde.

La soluzione alla base di questo intoppo è stata trovata nell’utilizzo di un nuovo comando per l’avvio del server associato all’applicazione React.

Un’applicazione base React infatti tramite il file package.json effettua l’avvio server dei propri scripts tramite il seguente comando.

https://lh4.googleusercontent.com/d-8CDvXvEr1CJcWyqwBhhOk81HPZKVJQL-rkdjgdo9ibMhDLnDSGLnb3N42svIZI6S4N7aZnEmm7HzGTOstg9Vzt-uvt5ns-CqEeRi9VcpSX3i_jyzmFWxMg-vy0lYIb1X6Q5uyPCx_P

Quello che si è rilevato invece molto utile nel prodotto finale raggiunto è stato l’utilizzo di un nuovo comando, concurrently, che è possibile utilizzare installandolo correttamente all’interno del progetto, e che prevede di poter avviare diversi server in concorrenza quando viene avviato il server principale.

Il risultato che ne segue, modificando l’opportuno file package.json è il seguente.

https://lh3.googleusercontent.com/_QMtwW5L-8aZEvHOkhG-y8akIatSR0VOKp1vPPDjsORIYVN0Ll1aKahxOcBGw7eZvQz1iGwZYpmBUyxM8WioeHdA4oiyJh9InHYjHnBCWXPlmv_CoQ8E0QTcBNoEMrfMOwZYfJpwVJJl

Così facendo, all’avvio dell’applicazione viene chiamato il comando concurrently che fa partire sia il server associato agli scripts della nostra applicazione, ma anche il server messo a disposizione da Fluid Framework.

Risolto il problema lato server, rimane quello di associare correttamente il client all’Editor Collaborativo utilizzato dall’utente.

Per prima cosa è stato dichiarato il client, importando la componente corretta da quelle installate del pacchetto Fluid Framework.

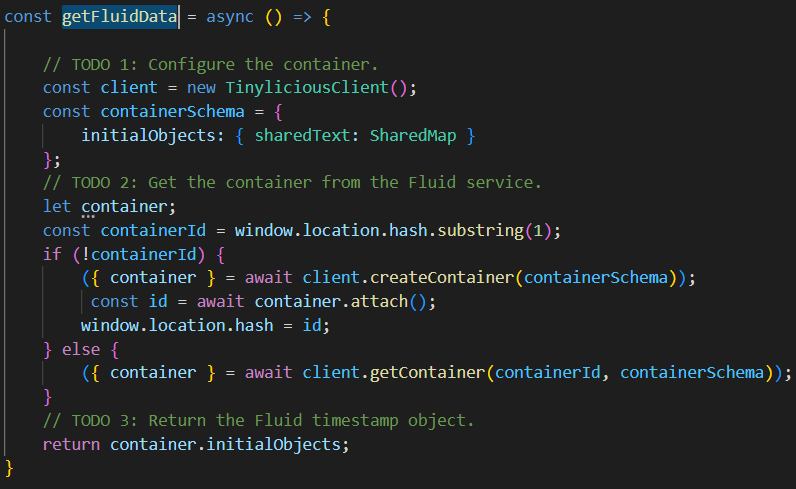
Un'opzione facoltativa è quella di dichiarare la porta dove dialogano i client connessi, ma omettendo questa dichiarazione si utilizza quella già presente all’avvio di LoIde.

Successivamente è stato dichiarato uno schema, contenente gli oggetti su cui si lavorerà attraverso il client.

Il terzo passaggio ha previsto la dichiarazione di un container, con annesso id, in modo tale da riconoscere l’oggetto che dialoga con il client.

Infine si ritorna lo il container dichiarato e lo schema ad esso associato.

Tutto ciò è stato implementato all’interno di una funzione che si presenta nella seguente immagine.



Questa funzione viene successivamente richiamata all’interno di uno useEffect, ovvero una modalità di react per effettuare operazioni a seguito di aggiornamenti che avvengono nella pagine, per andare a settare i dati associati al container che viene restituito.

Un ulteriore problema nell’implementazione dell’Editor Collaborativo è stato riscontrato sulla configurazione dell’Editor stesso utile per il framework utilizzato.

All’inizio si era pensato di utilizzare una semplice textarea modellata appositamente per permettere agli utenti di utilizzarla come un'area di lavoro il più simile possibile all’Editor della pagina principale.

Questo però avrebbe sminuito l’intento dell’implementazione, la sua importanza e anche la sua efficienza per gli utilizzatori.

 Analizzando però la componente utilizzata dall’Editor principale si è capiti che importando quest’ultima nella nuova pagina dedicata all’area collaborativa, poteva essere modellata esattamente allo stesso modo e renderla operativa e disponibile per affiancarle le funzionalità richieste dal Fluid Framework.

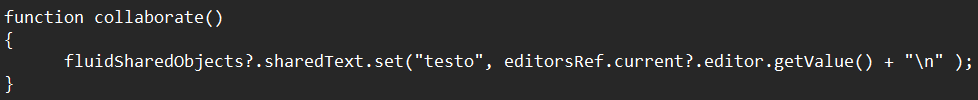
A differenza dell’Editor utilizzato nella pagina principale, che è rappresentato da una componente che utilizza la componente Ace Editor per creare un elemento unico con tutte le opzioni necessarie, nell'Editor Collaborativo si è lavorato invece proprio sull'Ace Editor, in modo tale da non conferire a terzi le caratteristiche richieste da Fluid Framework che risulterebbe in tal caso inutilizzabile.

Questo diventa un passo fondamentale, poiché cambiare il valore dell’Editor durante la collaborazione è possibile agendo direttamente sul valore di quest’ultimo, cosa non possibile utilizzando la componente presente nell’Editor principale.

L’Editor risultante risulta essere pronto a comporre il puzzle finale: l’unico tassello che rimane da inserire è la modalità di collaborazione.

Al fine di rendere la funzionalità più sicura e meno conflittuale possibile è stato introdotto un pulsante nella Toolbar dedicata all’Editor Collaborativo che ha la funzionalità di richiamare il Fluid Framework e mettere in contatto client e server e abilitare la collaborazione.

Una volta cliccato il pulsante viene richiamata la seguente funzione:



All’interno dell’Editor Collaborativo è stata inserita una voce indispensabile, quella della referenza, in modo tale da evitare il circolo inutile e dispersivo del contenuto dell’Editor e passando all’interno del metodo set della funzione del Fluid Framework il valore ottenuto dalla referenza dell’Editor corrente ad esso associato.

La collaborazione su cui si basa L’Editor Collaborativo di LoIde è possibile tra due o più utenti attraverso l’utilizzo di questa funzionalità sullo stesso Url: infatti basta semplicemente che essi abbiano aperto L’Editor Collaborativo sullo stesso Url e possono iniziare a sfruttare il tutto perfettamente.

Per permettere agli utenti di condividere l’Url della pagina su cui loro sono connessi è stato messo a disposizione un input field read only con accanto un pulsante per rendere l’Url visibile all’interno dell’input field, e poi successivamente di può proseguire a pulire l’area di quest’ultimo.

Il motivo principale alla base di questa scelta è il fatto che non bisogna dare per scontato che un utente debba intuire obbligatoriamente la concezione dello sviluppatore di rendere questa funzionalità disponibile semplicemente accedendo allo stesso Url e rendere veicolante il passaggio copiarlo dalla barra di ricerca e condividerlo successivamente.

Si prosegue quindi tramite il semplice utilizzo della funzionalità sottostante a prenderlo e mostrarlo di conseguenza.

https://lh6.googleusercontent.com/LHOjsFajTxHpMfHVj8vaspukdKON8P2G_Bh6SbJJFC8KDts1b3MQGqEew0hc827bcn8LKRrU8NsiURrvB-x8wr28kGCaKntMUnXZB7OVkCp-Rog1Sx9L4Ik8WIudpaaFNr0VSFfUj2gm

CONCLUSIONI

Sulla base dei risultati ottenuti e del lavoro svolto durante il periodo di tirocinio e svolgimento della tesi è arrivato il momento di tirare le somme.

Interfacciarsi su un progetto di medie/grandi dimensioni, attraverso canali di comunicazione tipici e usuali nel mondo del lavoro quali GitHub e Trello nel mio caso, è stata un’esperienza formativa a 360 gradi.

Risulta a volte troppo semplice immaginarsi come sarà la prima esperienza lavorativa, cosa aspettarsi e come relazionarsi, ma credo che una formazione come quella offerta dal Dipartimento di Informatica sia stata sufficiente a rispondere a qualsiasi dubbio possa avere mai avuto e rendermi più sicuro sulle scelte future, sulle modalità e sulle competenze da portare sul campo.

Posso affermare però di essere soddisfatto dei risultati ottenuti.

Sono partito da una conoscenza mediocre sugli strumenti utilizzati nel lavoro di tesi, e da una conoscenza molto scarsa relativa al modo in cui si lavora in un progetto, al modo in cui ci si interfaccia con un team di sviluppo, e alle pratiche da rafforzare ogni giorno sempre di più per eccellere in ciò che si fa.

Certamente ad oggi non mi credo di essere uno sviluppatore provetto, ma fortunatamente grazie a questa bellissima esperienza ho trovato ciò che realmente amo fare, e ciò che realmente voglio migliorare.

La base ottenuta dal mio percorso universitario è solida, ma va rafforzata sempre di più, e il lavoro di tesi mi ha confermato che in nessun altro campo come l’Informatica ci sono elementi che quotidianamente vanno aggiunti al proprio bagaglio personale e migliorati.

Ci sono elementi che a volte vengono considerati troppo banali, quali il modo di comunicare e le tipologie di aggiornamento collettivo sul progetto in corso, ma durante questo periodo, proprio sulla base di errori commessi, sono entusiasta di aver appresso nozioni indispensabili per essere considerati validi.

Ogni difficoltà è stata motivo di confronto e discussione, e sono felice di aver trovato sempre risposta ai miei problemi.

Certo il lavoro non finisce qua, perché sicuramente in qualsiasi percorso a maggior ragione quello intrapreso da noi Informatici l’aggiornamento non finisce mai, e ho imparato anche grazie all’esperienza fatta che non bisogna mai accontentarsi di ciò che si è raggiunto ma guardare sempre a cos’altro può essere fatto per migliorare.

I risultati ottenuti diventano quindi la base di una valutazione personale per aspirare ad ottenere sempre di più ed un punto di partenza fondamentale per il futuro.

BIBLIOGRAFIA

1. TypeScript :

Linguaggio di programmazione open source sviluppato da Microsoft

URL : <https://www.typescriptlang.org/>

1. GitHub :

Servizio che aiuta gli sviluppatori ad archiviare e gestire il codice e tracciare le modifiche

URL : <https://github.com/>

1. Trello :

Strumento che consente a gruppi di gestire visivamente qualsiasi tipo di progetto

URL : <https://trello.com/>

1. Intro.js :

Tool utile per la creazione step by step di un tutorial sui propri prodotti.

URL : <https://introjs.com/>

1. Fluid Framework :

Framework per creare applicazioni Fluid e abilitare l’uso dell’Editor Collaborativo

URL : <https://fluidframework.com/docs/start/quick-start/>

6. React:

React è una libreria open-source, front-end, JavaScript per la creazione di interfacce utente.

URL: <https://it.reactjs.org/>

7. Concurrently:

Tool di React che pemette di eseguire più comandi contemporaneamente.

URL: <https://www.npmjs.com/package/concurrently>

8. Shell:

La shell è la componente fondamentale che permette ad un utente di avere il più alto livello di interazione con un sistema operativo.

9. Cookie:

File di testo utile al server per ottenere informazioni sulle attività compiute da un utente su una pagina.

10. JavaScript:

Linguaggio di programmazione multi paradigma orientato agli eventi, utilizzato sia nella programmazione lato client web che lato server per la creazione di rest API.

URL: <https://www.javascript.com/>

11. JSON:

Acronimo di Javascript Object Notation, è un formato adatto all’interscambio di dati fra applicazioni client/server.

URL: <https://www.json.org/json-it.html>

12. Navigato.clipboard:

API che può essere utilizata per implementare le funzionalità di taglia, copia e incolla all’interno di un’applicazione web.