

E-Fit
Applicazioni e Servizi Web

Leo Marzoli - 0001102830 {leo.marzoli@studio.unibo.it},
Federico Pirazzoli - 0001079378 {federico.pirazzoli@studio.unibo.it},
Lorenzo Massone - 0001075612 {lorenzo.massone@studio.unibo.it}

03 Settembre 2023

0.1 Introduzione

Si vuole realizzare una soluzione software web-based basata su E-Commerce per una palestra, denominata da noi "E-Fit" il cui scopo è quello di erogare videocorsi online. E-Fit opera come una normale palestra, ma i suoi clienti si allenano direttamente da casa tramite la visione di corsi pubblicati dal suo staff, ovvero i suoi personal trainer. La palestra lucra sull'acquisto di videocorsi da parte degli utenti tramite una valuta personalizzata di "tokens". Il cliente che visita l'applicazione web spende del denaro per comprare i cosiddetti tokens che poi veranno utilizzati nell'acquisto di uno o più videocorsi di suo interesse.

0.2 Requisiti

Nello sviluppo delle funzionalità e delle caratteristiche del sistema si è scelto di preferire coloro che avrebbero portato un maggior interesse nell'utilizzo dell'applicazione web per l'utente finale, immaginando uno scenario quanto più realistico possibile. Dunque, si è scelto come funzionalità più rilevante delle altre la gestione di una piattaforma per la distribuzione di videocorsi che a loro volta dovranno rispettare le seguenti caratteristiche:

- La loro comparsa e aggiornamento risulta essere in tempo reale sulla piattaforma.
- Essi sono acquistabili con dei token (forma di denaro virtuale appositamente simulata).
- Ciascuno di essi è prodotto da un personal trainer e può essere usufruito da N utenti.

Si è scelto invece di considerare fra le funzionalità meno rilevanti, ma molto utili ai fini dello sviluppo, le seguenti:

- Gestione dei permessi sulla base del ruolo ricoperto al momento di login (Staff/Utente).
- Inserimento di una "Profile page" per ogni utente con relativi dati associati (come la visualizzazione dei tokens) e corsi acquistati.
- Possibilità di vedere quante persone hanno acquistato un videocorso e quante altre lo stanno visionando.

Fra le ultime funzionalità da inserire, ma con una importanza non particolarmente rilevante ai fini del progetto, pertanto facoltative si è sviluppato:

- Possibilità di filtraggio dei corsi sulla base di uno specifico personal trainer (Staff).
- Creazione di una pagina dedicata all'acquisto di nuovi token, con relativo aggiornamento della valuta associata ad ogni utente.

- Possibilità per l'utente di recensire un videocorso (like/dislike o commento).
- Possibilità per un personal trainer di effettuare una live (con partecipazione a pagamento).

0.3 Design

Riguardo la scelta di design del sistema questa segue uno stile User Centered Design (UCD). Gli utenti utilizzati durante la fase di sviluppo, ma anche per la successiva fase di test sono tutti simulati. In questo scenario è più facile caratterizzare gli utenti secondo le cosidette "personas" in quanto tutti gli utenti che decidono di approcciarsi ad E-Fit hanno in comune un interesse per il fitness. Questo viene fortemente tenuto in considerazione durante lo sviluppo delle funzionalità.

Inoltre, si è addottata una metodologia a "Waterfall", dove inizialmente si sono stabiliti quelli che erano i requirements e le principali funzionalità da integrare. Da qui, un primo design delle interfacce attraverso l'utilizzo di sketch e una analisi più dettagliata riguardo alla tipologia di utenti che E-Fit voleva tenere in considerazione, facendo quanto più possibile targeting degli stakeholders. Poi la fase di implementazione, verifica di ciò è stato introdotto tramite test e soprattutto debug delle funzionalità, per poi finire con il deployment. Si è scelto questa metodologia di sviluppo software rispetto ad una più dinamica come Agile: per un maggiore controllo nella suddivisione dei compiti fra i membri del team garantendo una certa "indipendenza" nel lavoro da svolgere, vista comunque la distanza e perché avendo requirements statici e ben definiti ed una complessità progettuale non particolarmente lunga in termini di tempo è risultato molto più efficiente procedere in questo modo.

Per quanto riguarda il design delle interfacce, si è seguita la regola KISS e "Less is More" per cercare di rendere quanto più semplici e intuitive all'utente. Tutte le interfacce implementate sono responsive in modo da migliorare la user experience.

Di seguito vengono riportati gli sketch delle interfacce pensate per E-Fit:

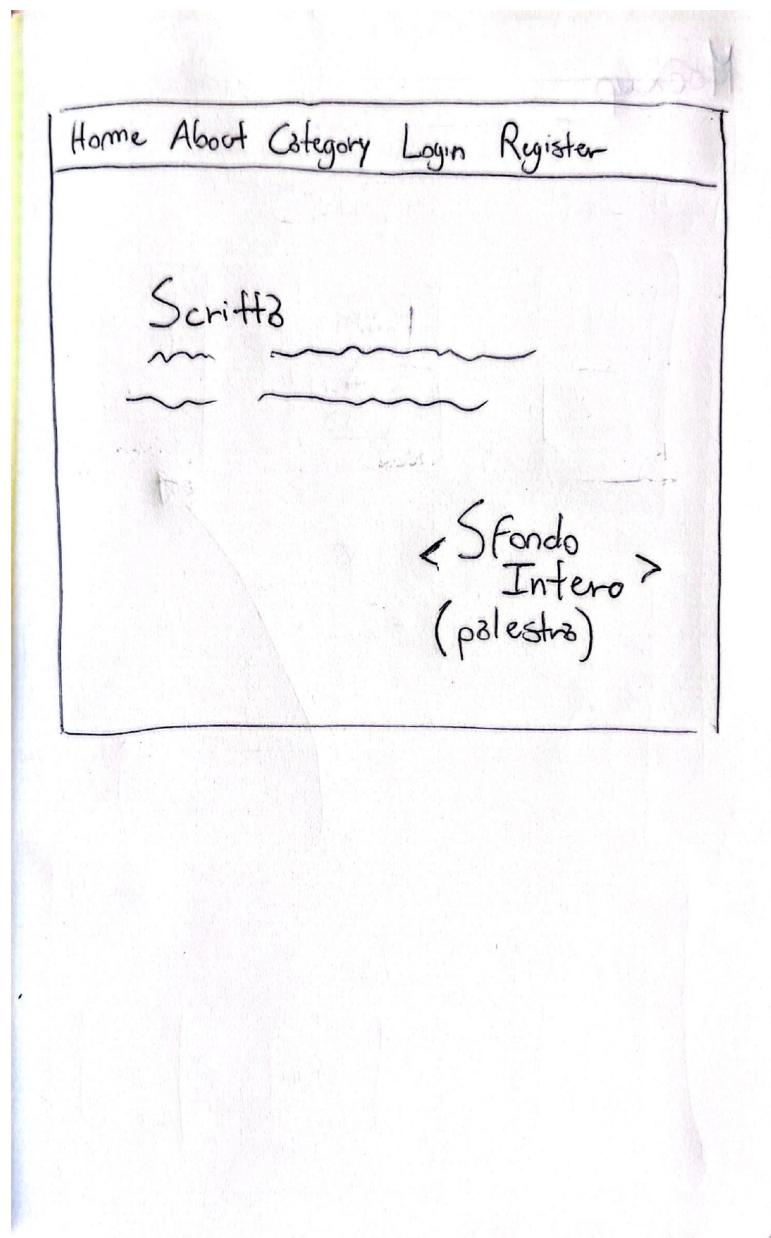


Figure 1: Home Page

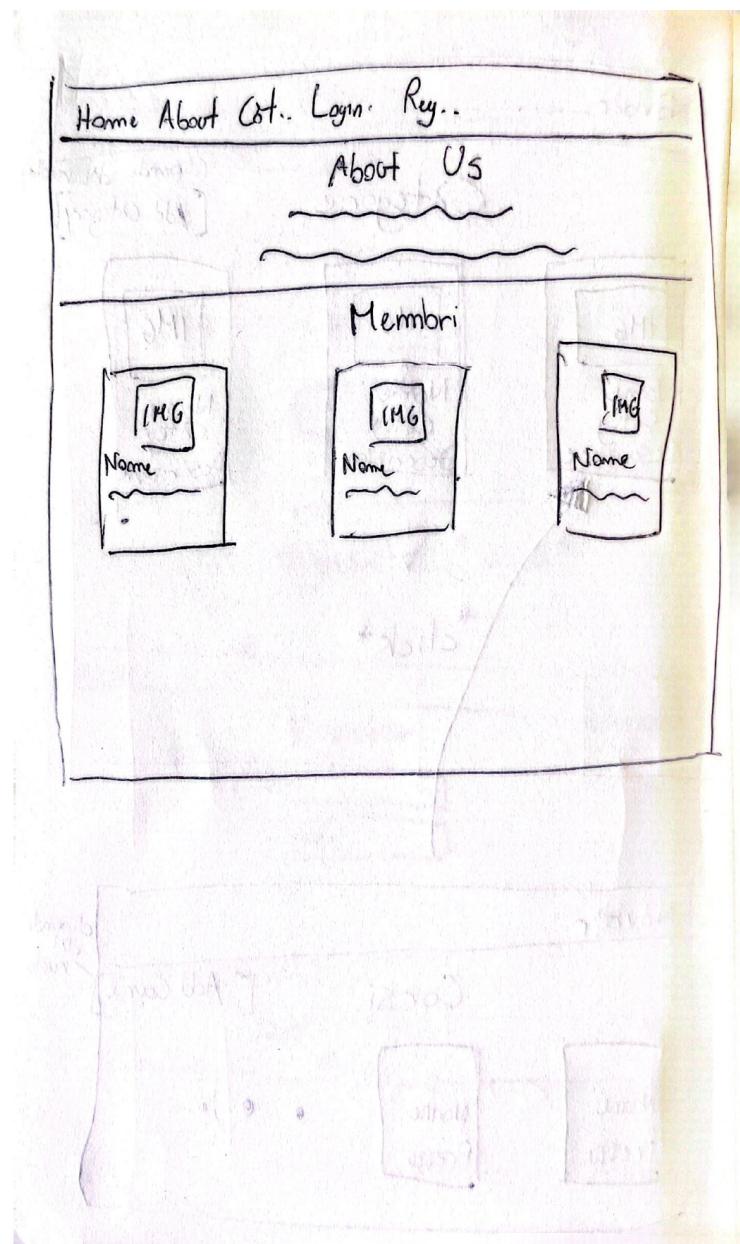


Figure 2: About Us Page

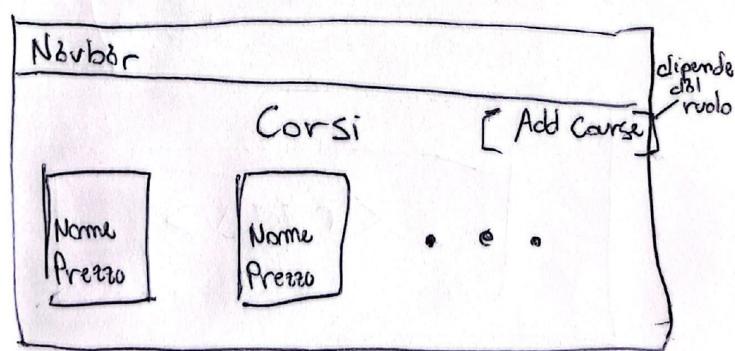
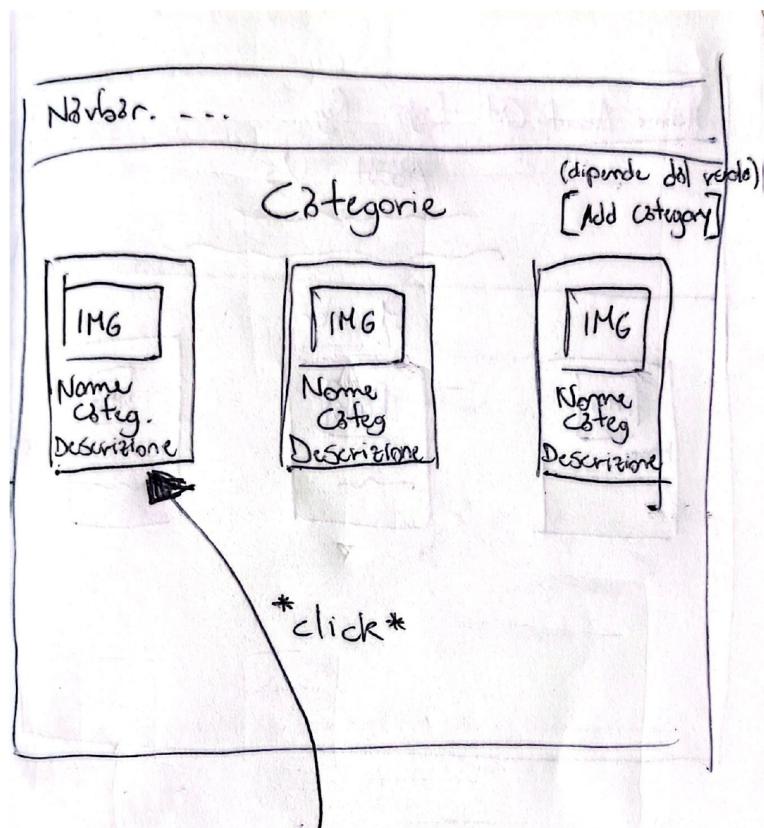


Figure 3: Categorie e Corsi Pages

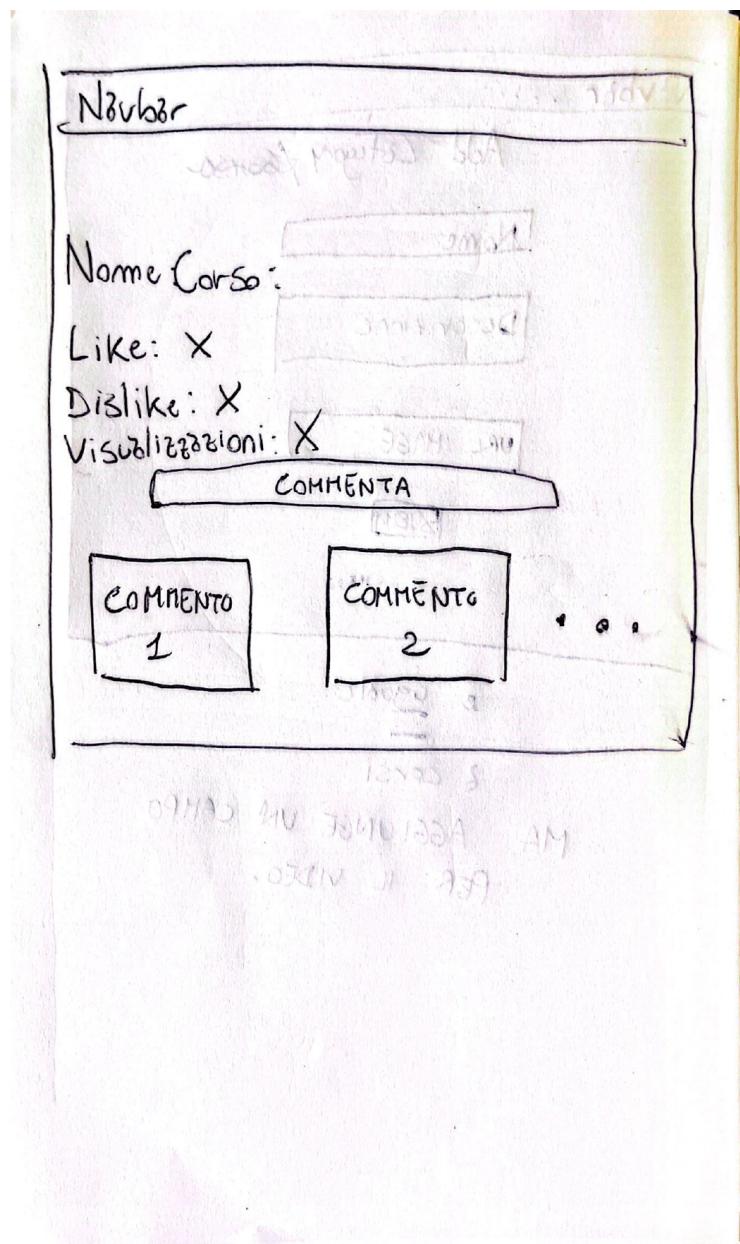


Figure 4: Contenuto di un corso Page

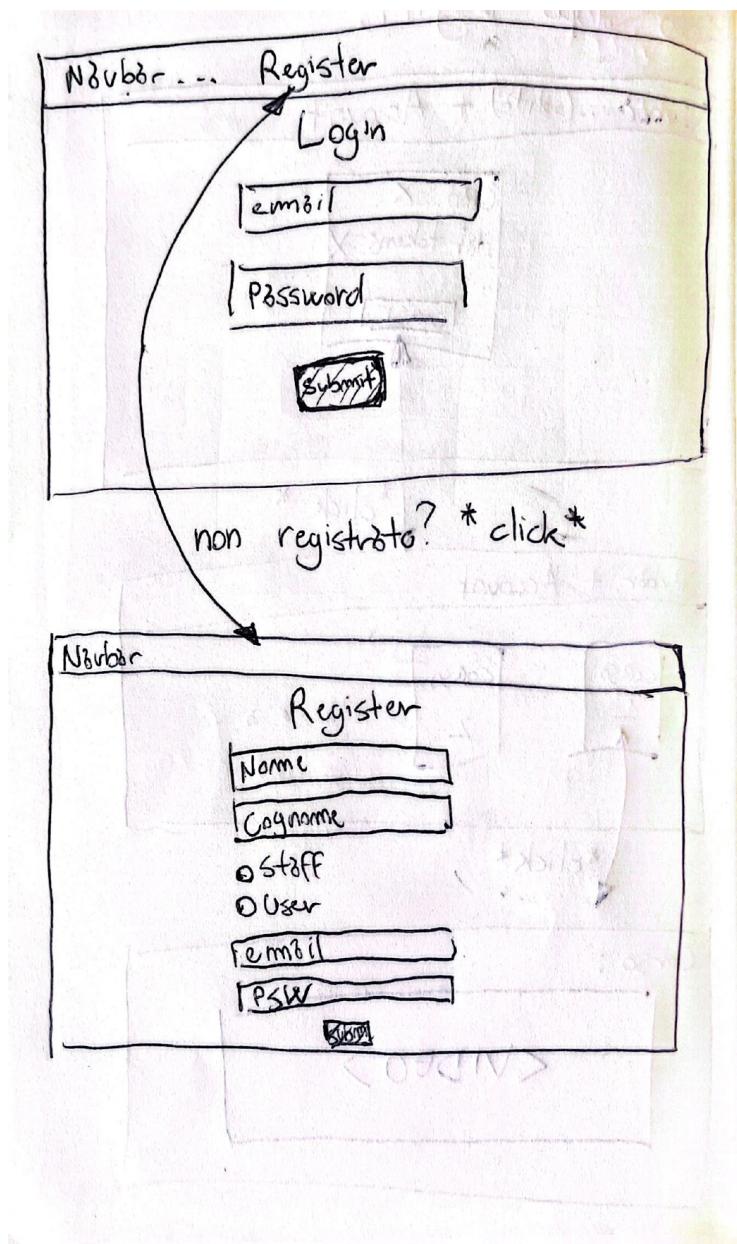


Figure 5: Login e Register Pages

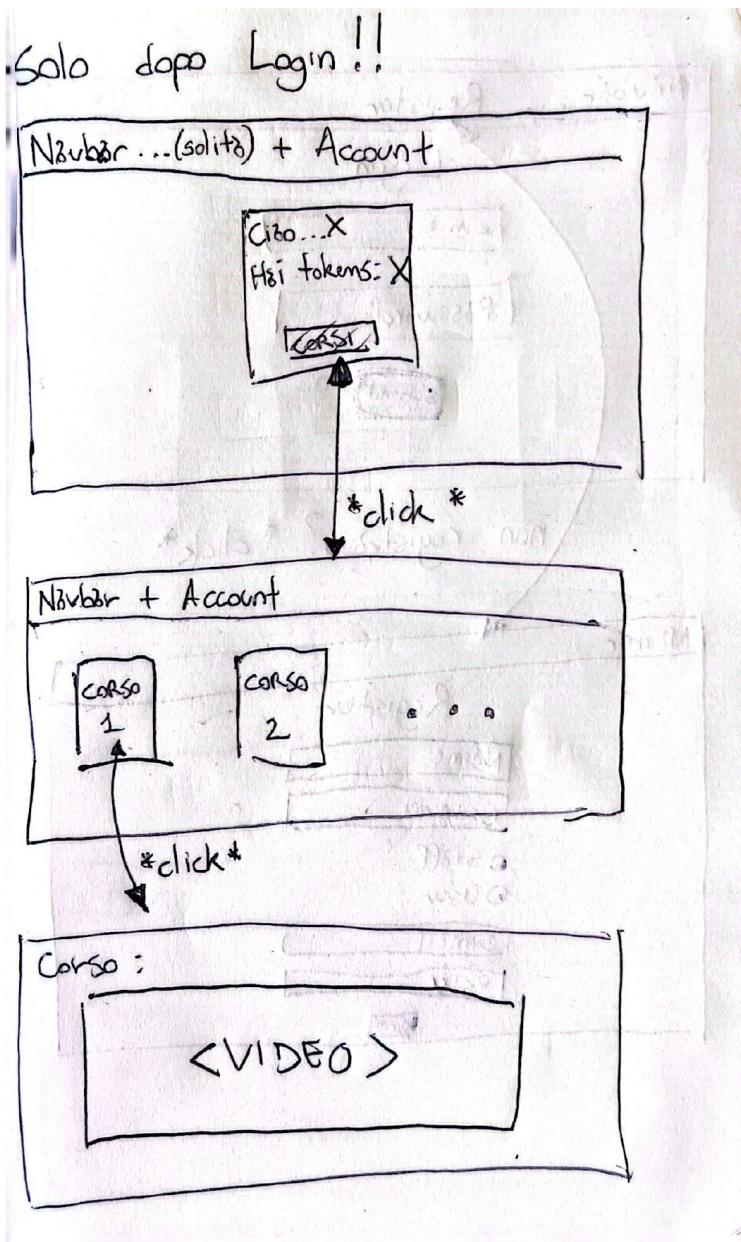


Figure 6: Account e Corsi personali Pages

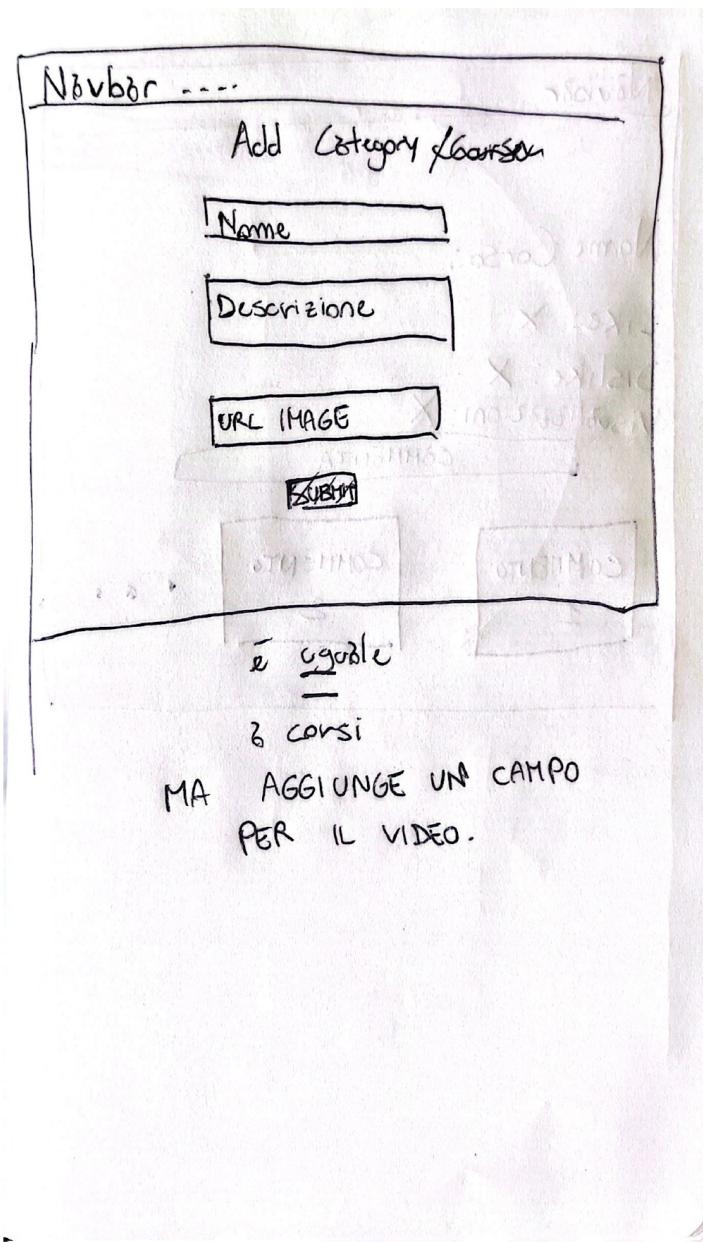


Figure 7: Add Category o Course Page

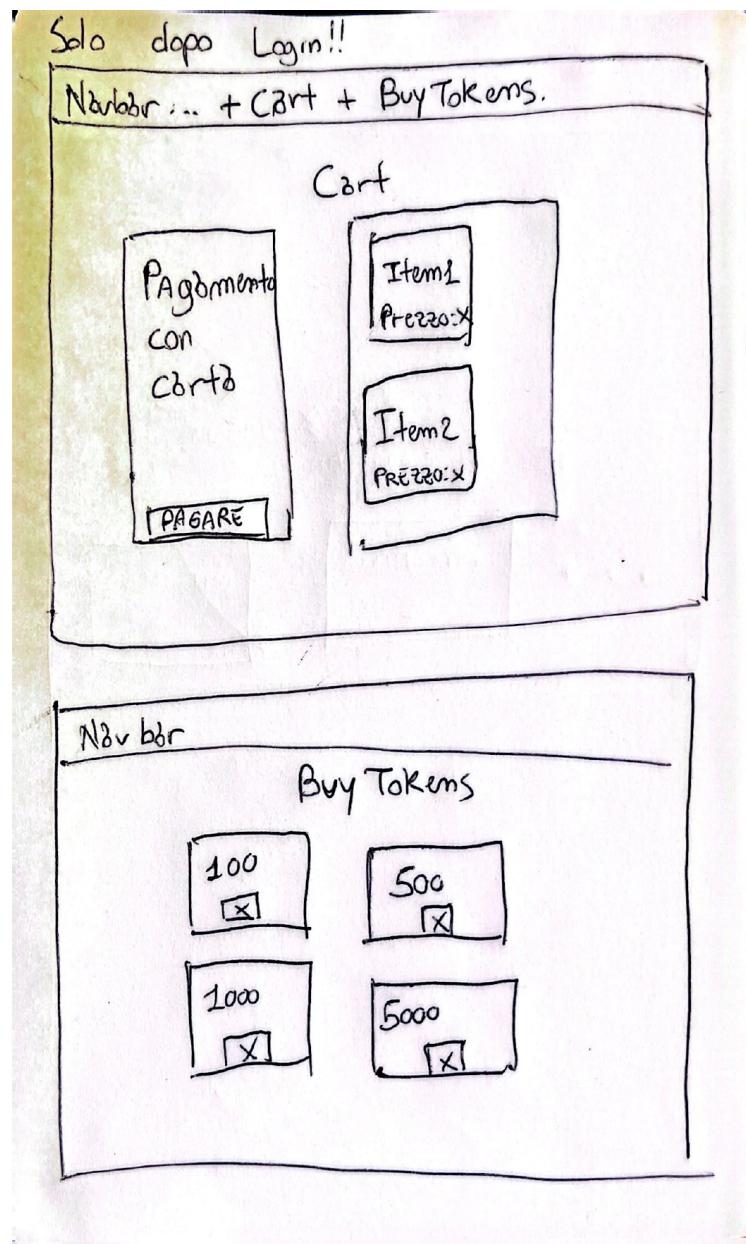


Figure 8: Cart e BuyTokens Pages

0.4 Tecnologie

Si è scelto di utilizzare una architettura basata su solution stack MEVN. Dunque, il sistema è costituito da un servizio di backend che utilizza Node.js come linguaggio di programmazione e Express.js come framework per la gestione delle richieste, supportato da un database non relazionale MongoDB. L'accesso al sistema è fornito agli utenti attraverso un'interfaccia sviluppata con Vue.js. Si è scelto di utilizzare come variante Vue, rispetto ad Angular, visto il suo notevole incremento di popolarità e personale interesse da parte del gruppo nel verificare l'efficienza di questa tecnologia.

0.4.1 Front-end

Fra le tecnologie utilizzate lato frontend oltre a quelle precedentemente citate possiamo trovare:

- VueCLI: semplifica notevolmente l'esecuzione del codice front-end, fornendo aggiornamenti automatici del rendering mentre le pagine vengono costruite e offrendo messaggi di errore chiari in caso di problemi di compilazione o esecuzione durante il processo di sviluppo.
- Bootstrap: uno fra i framework più conosciuti e utilizzati che offre un insieme completo di componenti e stili predefiniti, consentendo agli sviluppatori di creare rapidamente e facilmente interfacce utente responsive e moderne. Fornisce una griglia flessibile per la disposizione dei contenuti, componenti di navigazione, moduli, buttoni e molto altro. È altamente personalizzabile e compatibile con la maggior parte dei browser, rendendolo uno strumento essenziale per l'estetica di siti web. In particolare è stato utilizzato per la disposizione delle pagine web principali e per la creazione di "cards" sia per le categorie che per i videocorsi.
- CSS (Cascading Style Sheets): consente di definire la presentazione e la formattazione degli elementi HTML, inclusi layout, colori, tipi di carattere e dimensioni. Gli stili CSS possono essere applicati direttamente agli elementi HTML o collegati separatamente come fogli di stile esterni, offrendo un alto grado di flessibilità nell'aspetto e nella disposizione dei contenuti web. In questo caso specifico non sono stati utilizzati riferimenti esterni a file di stile CSS, ma ogni componente Vue.js ha integrato (se necessario) un tag di stile che definisce come deve apparire.
- Sweetalert: una semplice libreria utilizzata per migliorare la user experience, che permette di creare notifiche e alert personalizzati in modo da fornire tempestivamente un riscontro visivo ad alcune funzionalità.

Per rispettare le funzionalità elencate nei requisiti è stato necessario introdurre due nuove tecnologie lato front-end:

- Axios: Questa libreria permette di effettuare chiamate di rete al server, mantenendo una netta separazione tra l'utilizzo delle API e l'interfaccia

grafica. E' ampiamente riconosciuta come uno standard de facto nel campo, ci ha agevolato nella creazione di un metodo di accesso ai dati.

- Vuex: rappresenta un'estensione di Vue dedicata alla gestione dello stato, offrendo un modello per applicazioni a stato centralizzato, in questo caso da uno storage. Questo approccio agevola la gestione delle informazioni tra differenti componenti all'interno di un'applicazione a pagina singola (Single Page Application), garantendo un rigore e una protezione estremi nella modifica e nell'accesso ai dati, al fine di evitare situazioni di incostanza tra gli utenti. Essendo di vasto utilizzo abbiamo deciso di limitarne l'uso alla memorizzazione dei dati di un utente a Login sull'applicativo web e di ricordare con esso i dati associati ad ogni utente come il numero di tokens disponibili per ciascuno di esso o il contenuto del cart associato ad ogni utente durante l'acquisto di uno o più videocorsi.

0.4.2 Back-end

Come già evidenziato dal solution stack utilizzato le principali tecnologie utilizzate per la creazione del back-end sono Node.js ed Express per quanto riguarda la business logic e un database di tipo documentale (NoSql) a supporto del sistema creato MongoDB. Le tecnologie coinvolte in questa fase sono:

- Moongoose: questa libreria fornisce un'astrazione più comoda e orientata agli oggetti per interagire con MongoDB permettendo agli sviluppatori di definire in modo dichiarativo la struttura dei dati, denominata schema, e offre una serie di funzionalità per eseguire operazioni di lettura, scrittura, aggiornamento e cancellazione dei dati nel database. Inoltre, è molto importante far notare la sua caratteristica di poter definire middleware personalizzati per il reindirizzamento dei dati.
- Socket.io: questa libreria semplifica notevolmente lo sviluppo di applicazioni in tempo reale, consentendo ai programmatori di stabilire connessioni bidirezionali e mantenere una comunicazione continua tra il server e il client. Nel nostro caso specifico trova utilizzo nella visualizzazione real-time dei videocorsi per ciascuno utente che effettua una connessione ad E-Fit, ma anche per le categorie che raggruppano ogni videocorso e la funzionalità di like/dislike e la sezione commenti.

0.5 Codice

Nel codice sviluppato per il progetto il front-end è reso il più possibile modulare e scalabile, favorendo la sua riusabilità. Per questo si suddivide per ogni feature una componente Vue.js. Inoltre, per quanto riguarda la gestione dello stato globale questa è stata affidata allo standard de facto di Vuex il cui compito è quello di memorizzare attraverso l'utilizzo di storage per tutte le componenti le informazioni necessarie, nel nostro caso specifico si parla dei dati di un utente

(email e tokens) e il contenuto del suo carrello associato per l'acquisto di video-corsi.

Nella scrittura del codice per il backend un aspetto importante è l'utilizzo di middleware per effettuare il routing delle richieste da parte dei client e l'utilizzo della libreria Socket.io per l'aggiornamento real-time di categorie, videocorsi, like, dislike e commenti. Quest'ultima tecnologia ha integrato tramite le istruzioni "socket.on(...)" e "socket.emit(...)" rispettivamente l'ascolto e la registrazione di eventi. Per l'impostazione del lato server dell'applicazione web si è scelto di usare un modello MVC (Model-View-Controller), nonostante Vue utilizzi un modello MVVM (Model - View - ViewModel). La scelta di tornare ad una business logic gestita da un Controller e non da un ViewModel è dovuta alla scelta progettuale di avere una netta distinzione tra la view e la manipolazione dei dati.

0.6 Test

La funzione dei test è stata quella di verificare la responsività dei componenti che dovevano riadattarsi sulla base del dispositivo in cui venivano visualizzati, garantendo la portabilità del progetto. Per la verifica delle API si è utilizzato Postman che consente agli utenti di progettare, simulare, eseguire il debug, testare, documentare, monitorare e pubblicare API, tutto da un'unica posizione.

0.6.1 Euristiche di Nielsen

Il sistema è stato sottoposto alle heuristiche di Nielsen per testare l'usabilità delle sue interfacce utente:

1. Visibilità dello stato del sistema: sono state integrate librerie come "sweet-alert" per favorire la gestione di quest'aspetto in modo tale che l'utente che si collega ad E-Fit sappia sempre cosa stia succendendo, tramite notifiche o alert.
2. Corrispondenza tra sistema e mondo reale: sono state utilizzate icone e componenti grafiche sicuramente già conosciute dall'utente favorendo l'utilizzo delle interfacce, come l'icona del carrello per il "cart".
3. Controllo e libertà: l'utente può liberamente navigare fra le interfacce web tramite l'utilizzo di una componente "navbar", sia effettuato il login che non.
4. Consistenza e standard: l'utente effettuando una connessione avrà la sensazione di rimanere all'interno del contesto di E-Fit.
5. Prevenzione dell'errore: un utente potrà sempre tornare indietro e/o ricaricare il contenuto di una pagina per correggere errori.

6. Riconoscimento anziché ricordo: il layout delle interfacce web è quanto più minimale possibile, dato ciò l'utente non fa fatica a ricordare la posizione di elementi all'interno di E-fit.
7. Flessibilità ed efficienza d'uso: se un utente esperto decide di effettuare direttamente il login (poiché già registrato) avrà direttamente accesso alla sezione delle categorie per i videocorsi, mentre un utente meno esperto (che deve anche effettuare la registrazione) può comunque navigare su E-Fit, ma svolgerà un percorso più lungo.
8. Design e estetica minimalista: il layout delle interfacce segue uno stile minimalista come già evidenziato nella sezione di design.
9. Aiuto all'utente: sono stati implementati alert capaci di fornire all'utente informazioni riguardo l'errore che lo riguarda, in modo tale da potersi correggere.

0.6.2 Usability Test

I test sono stati effettuati soprattutto dai membri del team che hanno lavorato a componenti diverse del progetto in momenti diversi, pertanto lo scenario ricreato era molto realistico e tramite l'osservazione di uno o più membri del team, si controllava la lista di azioni compiuta da colui che aveva il compito di testare le componenti per ottenere feedback utili poi alla valutazione dell'interfaccia.

0.7 Deployment

Per il corretto funzionamento e deploy del progetto bisogna seguire i seguenti passi:

1. Recuperare i file sorgenti del progetto al seguente repository Github:
https://github.com/MarzoliLeo/ASW_Ecommerce
2. Assicurarsi di aver correttamente installato Node.js e MongoDB.
3. Aprire il progetto con l'IDE che si preferisce (consigliato fortemente: VisualStudio Code).
4. Installare come plugin al suo interno Volar (e disabilita Vetur) + TypeScript Vue Plugin (Volar).
5. Inoltre, aggiungere i plugin ESLint e Prettier (facoltativo, ma fortemente suggerito).
6. Aprire il terminale dentro l'IDE e digitare i seguenti comandi:
 - Setup del progetto.

```
npm install
```

- Compilazione e minimizzazione dei sorgenti per la compilazione.

```
npm run build
```

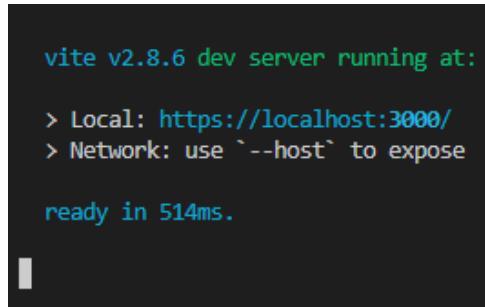
- Esecuzione dell'ambiente di sviluppo.

```
npm run dev
```

- Nota: per eventuali problemi con ESLint usare questo ulteriore comando.

```
npm run lint
```

7. A questo punto nel terminale dovrebbe partire il tool per l'esecuzione di Vue3, chiamato "VITE v4.3.5" e dovrebbe apparire una cosa simile: da



```
vite v2.8.6 dev server running at:
> Local: https://localhost:3000/
> Network: use `--host` to expose
ready in 514ms.
```

Figure 9: Inizializzazione Frontend con Vue.js

questo punto premere il pulsante "o" (shortcut) della tastiera avendo selezionato il terminale per aprire il progetto Vue3 nel browser e visualizzare il frontend.

8. Come step finale da seguire per mettere in comunicazione il front-end con il back-end e la base di dati gestita da MongoDB, si utilizza il comando

```
node .\src\backend\server.js
```

che andrà eseguita a sua volta dentro una shell diversa da quella dedicata a Vue, ma interna al root di progetto (nel nostro caso utilizziamo direttamente il terminale di Visual Studio Code).

Eseguiti questi passaggi si avrà accesso ad E-Fit e ai suoi contenuti.

0.8 Conclusioni

Concludiamo questo progetto soddisfatti. Ci ha permesso di arricchire le nostre competenze personali in ambito web, ma non solo. E' stato molto interessante anche analizzare aspetti come la HCI che spesso vengono messi da parte nello sviluppo di progetti informatici. Le tecnologie viste rispecchiavano le aspettative di efficienza che ci eravamo posti nei loro confronti, in particolare l'utilizzo di Socket.io è ciò che ci ha interessato di più, sia per la sua implementazione, ma anche per la vastità di funzionalità in cui può essere integrato per migliorare i contenuti di un servizio web. Il progetto rispecchia i requirements preposti, ma può sicuramente migliorare in termini di funzionalità. E' un progetto espandibile che opera come base per un potenziale contesto realistico futuro.