SErvizio Rilevazione Presenze Aule

Applicazioni e Servizi Web

Enrico Fiumana - 0001103348
enrico.fiumana2@studio.unibo.it

24 ottobre 2023

Indice

1 Introduzione

Il progetto nasce dall'esigenza dell'Ateneo di conteggiare gli studenti che presenziano alle lezioni e di monitorare l'utilizzo delle aule per capire quanto queste vengano sfruttate. Il sistema è composto da diversi componenti, sia hardware che software. In particolare nelle aule sono presenti delle speciali fotocamere che rilevano la traccia termica delle persone che sono collegate a dei Raspberry Pi4[1] sul quale girano una serie di software che, ogni tot minuti, conteggiano quante persone sono presenti sui banchi e lo trasmettono, attraverso delle apposite API, ad un server centralizzato. Questo server ha inoltre una serie di API che permettono di interrogare lo stato dei device, il conteggio sia attuale che storico e di gestire sia gli edifici su cui sono installati i device sia la posizione degli apparati

2 Requisiti

Inizialmente ho analizzato i requisiti necessari all'applicazione, suddividendoli in requisiti di business (quindi dell'applicazione stessa) e di utente (riguardanti quindi l'interazione con l'utente finale) Requisiti di business

- Poter gestire l'aggiunta/modifica/rimozione di edifici
- Poter gestire l'aggiunta/modifica/rimozione di device all'interno di un edificio
- Poter gestire le richieste in real time del conteggio per device e/o per edificio
- Poter verificare lo stato dei device
- Poter gestire utenti di tipo diverso (amministratori e viewer)

Requisiti utente

- Visualizzare su una mappa i device e il loro stato
- Visualizzare su una mappa l'ultimo numero di presenze rilevato
- Poter richiedere il numero istantaneo delle presenze
- L'amministratore deve poter inserire nuovi edifici con la possibilità di caricare una mappa dove inserirei i device. Se non c'è la mappa, una lista
- L'amministratore può potere aggiungere, modificare o cancellare device sulla mappa/lista

3 Design

La metodologia utilizzata per lo sviluppo dell'applicazione, fa riferimento alla cosiddetta *"User Centered Design"* (UCD), cioè la pratica di mettere al centro dello sviluppo, le necessità dell'utente finale. Siccome può diventare difficile trovare utenti disponibili, per ottenere lo stesso risultatosi fa riferimento a quelle che vengono chiamate *"Personas"*, cioè degli utenti virtuali che rispecchiando diverse necessita da soddisfar attraverso l'applicazione.

Pur avendo sviluppato l'applicazione in solitaria, ho cercato di adottare Il metodo di sviluppo cosidetto "AGILE", permettendomi di sviluppare in maniera incrementale le funzionalità richieste.

Sono inoltre stati opportunamente installati e configurati in ogni aula, degli apparati che acquisiscono ed elaborano le immagini catturate da apposite telecamere e trasmettono poi i dati all'applicativo

3.1 Design dell'interfaccia

Per lo sviluppo dell'interfaccia utente invece ho fatto riferimento alla metodologia cosiddetta "Less is more" permettendomi così di sviluppare un'interfaccia minimale ma essenziale, comprendente cioè tutte le funzionalità richieste e permettendo contemporaneamente, la possibilità di ampliarne le funzionalità.

3.1.1 Mockup

mockup

3.2 Target User Analysys

I target user di questa applicazione possono essere principalmente suddivisi in 2 diverse tipologie che poi, a loro volta, possono esser suddivise ulteriormente a seconda dell'ambito in cui operano. Le principali categorie sono:

- L'amministratore di sistema che potrà gestire tutti gli oggetti presenti nel sistema.
- L'utente normale che è invece interessato all'interrogazione dei dati statistici.

In particolare ho pensato a 4 personas utilizzatori dell'applicativo.

- Enrico: È l'amministratore di sistema: deve poter gestire in maniera efficiente e facilmente gli edifici, le aule e gli utenti che possono accedere.
- Mary: fa parte del personale di portineria: le chiedono spesso se un'aula è disponibile e ha quindi bisogno di un sistema che le permetta in maniera facile e veloce di capire se nell'aula ci sono persone.

- Silvia: È una docente Che vuole dei report settimanale sulla frequenza delle sue lezioni per verificare che non si siano abbandoni ed eventualmente averne un immediato riscontro e quindi porre in atto le necessarie azioni.
- Vittorio: Anche Vittorio è un docente, ma è inoltre il responsabile del CdL: deve quindi fornire dei report settimanali sulla frequenza degli studenti del suo CdL

3.3 Design architetturale

L'architettura dell'applicativo è suddivisa in due entità: il backend che si occupa della gestione della persistenza dei dati e che gestisce la cosiddetta "business logic" e il frontend che invece si occupa della gestione dell'interfaccia con l'utente finale. La comunicazione fra queste due entità avviene attraverso il paradigma di API REST. L'utilizzo di questa tecnologia ci permette di:

- Indipendenza dalla tecnologia utilizzata dal client e dal server
- Avere una maggior possibilità di scalare
- Non essere dipendenti dalla tecnologia utilizzata per la memorizzazione dei dati persistenti
- Facilità di identificazione ed accesso ai dati

Oltre a queste due parti che forniscono l'accesso ai dati da parte dell'utente finale, abbiamo anche una terza entità, che sarebbero gli apparati presenti nelle aule che catturano le immagini e calcolano il numero di persone presenti.

3.3.1

4 Tecnologie

Ho deciso di adottare lo solution stack MEVN, ma vista la tipologia dei dati persistenti da gestire , si è deciso di non utilizzare il database documentale MongoDB[2], mail più classico database relazione MySql[3]. La parte di frontend invece è stata realizzata utilizzando il framework VueJS[4]. Il modulo di backend è stato realizzato appunto utilizzando Node.js[5] ed Express[6] come framework per la gestione delle chiamate API REST, utilizzando, come già detto, un classico DB relazione Mysql. Per quanto riguarda invece gli apparati presenti nelle aule invece, per quanto riguarda la parte di gestione delle telecamere e del conteggio sono stati usati software particolari opensource, mentre invece per quanto riguarda la comunicazione con il backend, anche qui è stati utilizzato Node.js ed Express.js

4.1 Frontend

Per quanto riguarda lo sviluppo del frontend, come specificato ho utilizzat Vue.js. In particolare ho utilizzato ViteJS[7] che permette una miglior gestione del codice. Inoltre, attraverso l'aggiornamento immediato delle pagine ad ogni modifica, attraverso il server integrato, mi ha permesso una maggior velocita nello sviluppo. Ho utilizzato anche l'estensione Vue DevTools per Firefox, che mi ha permesso di verificare Runtime lo stato dei componenti e di tutta l'applicazione. Ho suddiviso l'interfaccia in diversi componenti, in modo da modularizzare l'applicazione.

4.1.1 Stile

Per la parte riguardante invece il rendering dell'interfaccia utente, ho utilizzato Bottstrap, in paerticolare la versione per Vue, denominata BootstrapVue[8]. Questo, basandosi su Flexbox, ha permesso di avere una interfaccia con un layout fluido

4.1.2 Gestione dei dati e dello stato

Per quanto riguarda invece la gestione dei dati, ho utilizzato Axios[9] per realizzare le chiamate alle API di backend. Questo è ormai una libreria "standard" in quanto utilizzatissima per effettuare la comunicazione fra client e server ed è anche molto potente e facile da utilizzare . Per quanto riguarda invece il mantenimento dello stato interno dell'applicativo, in particolare per quanto riguarda sia l'autenticazione e l'autorizzazione, ho utilizzato Pinia[10]. Questo strumento permette di mantenere uno stato comune a tutti i vari componenti dell'applicazione, facilitando così l'accesso ai dati di comune interesse.

4.2 Backend

Come già detto per quanto riguarda la realizzazione del backend, mi sono affidato all'uso di Node.js ed Express.js, mentre invece per la memorizzazione dei

dati ad un più standard Mysql L'utilizzo di Node.js ed Express.js mi ha permesso uno sviluppo rapido, efficace ed abbastanza semplice della sezione server, anche grazie alla numerosissima disponibilità di componenti già preconfezionati presente in rete

4.2.1 Persistenza

Per quanto riguarda la persistenza dei dati, ho utilizzato un DB relazione Mysql mediante la libreria Mysql2[11]. Questo da un lato mi ha aiutato in quanto già estremamente abituato al suo utilizzo, ma mi ha anche costretto ad un overhead di lavoro dato dalla preparazione delle tabelle e degli indici necessari D'altro canto invece, la possibilità di utilizzare un DB documentale come MongoDB, mi avrebbe aiutato molto nella realizzazione del DB stesso, cosa che infatti sarebbe stata realizzata attraverso l'uso della libreria Mongoose che, una volta definito lo schema dei dati, realizza la collezione al momento dell'utilizzo

4.3 Apparati

Per gli apparati presenti nelle aule si è deciso di utilizzare dei Raspberry Pi4 sui quali è stato installato il sistema operativo Raspbian[12] (Cioè una distribuzione Debian Linux customizzata per girare su Raspberry ARM[13]). Questi sono collegati a delle telecamere speciali Intel RealSense[14] che permettono di avere immagini della sola profondità delle aule, visualizzando così solo i contorni della persona presente in aula e non il viso, rispettando così la privacy delle persone. Su questi apparati è stato installato diverso software che permette appunto, partendo da una immagine in profondità dell'aula, di conteggiare le persone presenti. Questo software, attraverso una serie di script appositamente creati da me, si interfaccia poi con le API presenti sul server di backend trasmettendo i dati acquisiti, questi dati sono poi trasferiti sul sistema di memorizzazione dei dati persistenti. Sugli apparati è stato inoltre installato un server HTTP realizzato attraverso Node.JS ed Express.JS. Questo fornisce delle ulteriori API che il backend sfrutta per eseguire particolari operazioni sul device. In definitiva su questi apparati è stato svolto un importante lavoro di studio, configurazione e testing.

5 Codice

Durante lo sviluppo del codice sia del backend che del frontend, ho posto particolare attenzione alla modularità del sistema e cercando di rendere il software facilmente scalabile. Nel frontend ho sviluppato una serie di componenti Vue, uno principale per ogni pagina che si occupa della logica e altri al suo interno specializzati invece nella renderizzazione dei dati. Ho inoltre utilizzato uno strumento per lo state management: pinia. Con questo strumento sono riuscito a memorizzare e condividere fra diversi componenti, diverse informazioni utili alla gestione dello stato, come ad esempio lo userid dell'utente loggato

6 Test

Il testing del frontend è stato fatto utilizzando i tre browser più utilizzati attualmente, cioè Firefox, Chrome ed Edge. Questo per poter garantire la maggior portabilità possibile. Tutti i test sono stati eseguiti con esito favorevole. Le API del backend invece sono state testate inizialmente attraverso l'uso di Postman, in modo da verificare che il comportamento fosse quello desiderato, prima di portarlo in produzione Per quanto riguarda invece la User Experience, purtroppo non sono riuscito, come invece mi sarebbe piaciuto, sottoporre dei test ad utenti reali e registrarne le reazioni. Mi sono dovuto accontentare di valutare e di cercare di applicare. Il decalogo delle 10 euristiche di Nielsen[15]. In particolare:

- **Corrispondenza tra sistema e mondo reale** – I termini e le icone sono quelli familiari a chi di solito usa questo tipo di applicativi
- **Controllo e libertà** – L'utente ha la possibilità di muoversi come meglio crede all'interno dell'interfaccia, senza nessuna o pochissime costrizioni
- **Consistenza e standard** – L'interfaccia è sempre la stesa in tutto l'applicativo
- **Riconoscimento anziché ricordo** – I layout sono semplici e di facile comprensione
- **Design e estetica minimalista** – L'interfaccia, come già detto, è basata sulle regole del KISS, quindi di facile comprensione per chiunque

7 Deployment

Il deployment dell'applicativo avviene utilizzando la tecnica dei container Docker[16], in modo da poter assicurarne la portabilità e la coerenza dell'installazione.
L'applicativo può essere avviato utilizzando i compose file presenti nella root del progetto. Tali file configurano ed avviano i 3 componenti principali, cioè il database, il backend e il frontend. Visto che l'applicativo sarà acceduto sia dal browser presente sul pc dell'utente che da quello presente sull'host di installazione, ho previsto due possibili configurazioni: una, di testing, che prevede l'installazione, la configurazione e l'utilizzo tutto sulla stessa macchina, ed una seconda configurazione che invece prevede, come detto, l'utilizzo da parte del browser personale dell'utente finale. In quest'ultimo caso ho dovuto preconfigurate nel file di configurazione, il nome host del backend, che poi, attraverso un alias (CNAME) all'interno del DNS, punterà all'host del container di backend, il quale espone la porta di accesso. I 3 file di configurazione sono quindi:

- docker-compose.yaml
- docker-compose-localhost.yml

- docker-compose-remote.yml

L'esecuzione avviene attraverso il comando docker-compose in questo modo: docker-compose -f docker-compose.yaml -f docker-compose-¡remote—localhot¿.yml –build up

8 Conclusioni

Visto la povertà del mio background tecnologico nell'ambito dei servizi web, e vista la delicatezza del progetto in sé, mi ritengo molto soddisfatto del lavoro svolto. Mancano ancora diversi tasselli al progetto, che però sono solo di secondaria importanza e che possono essere ritenuti delle rifiniture, come ad esempio:

- Codificare le password attraverso una funzione di hashing all'interno del frontend
- Creare una nuova API sul backend per poter cancella tuttii dati di un'aula quando questa viene eliminata
- Sui Raspberry inoltrare i log di sistema (rsyslog) su un syslog server e mettere il filesystem in modalità overlay (In sola lettura, le modifiche saranno solo in RAM) in modo da preservare la vita alla memoria SD
- Poter far cambiare la periodicità di rilevazione delle presenze. La relativa API è già presente sul backend.
- Permettere di "forzare" la rilevazione all'istante. La relativa API è già presente sul backend

Mi sembra che il prodotto finale sia di facile utilizzo ed intuitivo, presentando tutte le funzioni richieste dagli utenti, in modo semplice e chiaro Per quanto mi riguarda, forse perché è stato il mio primo progetto o forse perché l'applicazione sarà realmente utilizzata all'interno di UniBo, malgrado diversi momenti di sfiducia, sono molto contento di essere arrivato in fondo e di aver appreso, sia dal punto dio vista tecnico che da quello teorico, tutta una serie di nozioni a me sconosciute e che da ora in avanti faranno parte del mio bagaglio culturale. Insomma, sono stato veramente contento di aver affrontato questo progetto e di averlo concluso, a mio avviso, in maniera positiva.

GRAZIE!

Riferimenti bibliografici

[1] Raspberry. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/, .

[2] Mongodb. https://www.mongodb.com/.

[3] Mysql. https://www.mysql.com/.

[4] Vue.js. https://vuejs.org/.

[5] Node.js. https://vuejs.org/.

[6] Express. https://expressjs.com/.

[7] Vite. https://vitejs.dev/.

[8] Bootstrapvue. https://bootstrap-vue.org/.

[9] Axios. https://axios-http.com/.

[10] Pinia. https://pinia.vuejs.org/.

[11] Mysql2. https://github.com/sidorares/node-mysql2.

[12] Raspbian. https://www.raspbian.org/, .

[13] arm. https://www.arm.com/.

[14] Realsense. https://www.intelrealsense.com/.

[15] 10 heuristics nielsen. https://www.nngroup.com/articles/ten-usability-heuristics/.

[16] Docker container. https://www.docker.com/.