

KronosDB

Applicazioni e Servizi Web

Andrea Zammarchi - 000725342 {andrea.zammarchi3@studio.unibo.it}

04 Settembre 2023

1 Introduzione

La presente relazione documenta il progetto sviluppato in collaborazione con Kronos S.R.L. di Forlì, un'azienda che da anni opera nel settore dell'assistenza e delle riparazioni di dispositivi elettronici, principalmente Apple. Questo progetto si concentra sulla creazione di una Web App con l'obiettivo principale di fornire uno strumento di gestione degli interventi tecnici che semplifichi e ottimizzi il lavoro dei dipendenti dell'azienda.

La Web App sviluppata consente ai tecnici di Kronos S.R.L di registrare dettagliati report degli interventi effettuati presso i clienti distribuiti sul territorio. Ogni tecnico, in base alle autorizzazioni assegnate in base al proprio ruolo, può accedere al portale, gestire l'elenco dei clienti e aprire o modificare ticket di assistenza. Questi ticket rappresentano una traccia completa e accurata degli interventi, facilitando la comunicazione interna, il monitoraggio delle attività e il miglioramento della qualità del servizio erogato.

2 Requisiti

2.1 Requisiti utente

Requisiti per tecnico "Base"

- **Apertura Ticket:** L'utente Base deve poter aprire nuovi ticket di assistenza per registrare gli interventi tecnici.
- **Consultazione Ticket:** L'utente Base può consultare l'elenco dei ticket, ma non può modificarli o cancellarli, in quanto è un *receptionist* e non esegue fisicamente gli interventi tecnici.
- **Gestione Clienti:** Deve essere in grado di aggiungere nuovi clienti al sistema e di modificarne le informazioni.
- **Consultazione Tecnici:** L'utente Base può visualizzare l'elenco dei tecnici disponibili.
- **Chat Broadcast:** Deve essere in grado di partecipare a una chat broadcast con tutti i tecnici per comunicazioni generali.

Requisiti per tecnico "Junior" (include tutti i requisiti del tecnico "Base")

- **Modifica Ticket:** L'utente Junior ha il permesso di modificare i ticket esistenti, ad esempio per aggiornare lo stato o i dettagli dell'intervento.
- **Costo Orario:** L'utente Junior ha un costo orario, in quanto esegue effettivamente gli interventi tecnici.

Requisiti per tecnico "Senior" (include tutti i requisiti del tecnico "Junior")

- **Costo Orario Maggiore:** L'utente Senior ha un costo orario maggiore rispetto al Junior, in quanto più esperto ed esegue interventi tecnici generalmente più complessi.

Requisiti per tecnico "Admin" (include tutti i requisiti del tecnico "Senior")

- **Costo Orario Maggiore:** L'utente Admin ha un costo orario maggiore rispetto al Senior.
- **Modifica Info Tecnici:** L'utente Admin ha il permesso di modificare le informazioni degli altri tecnici, inclusi i costi orari.

2.2 Requisiti funzionali

Requisiti funzionali per la gestione dei Ticket

- **Campi Obbligatori:** Gli utenti devono completare campi obbligatori quando aprono un ticket, come la descrizione del problema, il cliente coinvolto e la data di apertura.
- **Assegnazione tecnico:** Gli utenti devono poter assegnare facoltativamente un tecnico ad un ticket aperto per l'intervento.
- **Gestione dell'Elenco Ticket:** Gli utenti devono poter cercare, visualizzare e ordinare l'elenco dei ticket in base a diversi criteri (ad esempio nome cliente, data apertura, o filtraggio per ticket già chiusi o non).

Requisiti funzionali per la gestione dei Clienti

- **Gestione dell'Elenco Clienti:** Gli utenti devono poter cercare, visualizzare e ordinare l'elenco dei clienti in base a diversi criteri (ad esempio nome).
- **Tessere:** Gli utenti devono poter aggiungere, modificare o rimuovere delle Tessere che i clienti possono acquistare. Queste tessere contengono un numero di ore totali di intervento prepagate. I clienti al momento del pagamento possono decidere se pagare un ticket tramite un saldo oppure scalando le ore di intervento da una tessera posseduta.

Requisiti funzionali per la gestione dei Tecnici

- **Assegnazione Ticket:** Gli utenti autorizzati devono poter assegnare i ticket a specifici tecnici in base alla loro disponibilità e competenza.

- **Compilazione Ticket:** Gli utenti autorizzati devono poter compilare i ticket aperti una volta effettuati i vari interventi, con una serie di campi specifici, inclusa la firma del cliente per confermarne la presa visione.
- **Gestione Ruoli Utente:** Gli amministratori devono poter creare, modificare e cancellare account utente e definire i loro ruoli e le relative autorizzazioni.

Requisiti funzionali per la Chat Broadcast

- **Notifiche:** Gli utenti devono poter ricevere una notifica ogni qualvolta viene inviato un messaggio da un tecnico sulla chat broadcast.
- **Topics:** La chat broadcast deve poter essere suddivisa su diversi topic (argomenti di discussione o comunicazioni importanti).

2.3 Requisiti non funzionali

- **Reattività:** Il sistema deve potersi aggiornare automaticamente senza la necessità di aggiornare manualmente la pagina da parte dell'utente.
- **Usabilità:** Interfaccia utente intuitiva e user-friendly per facilitare l'utilizzo dell'applicazione da parte degli utenti.
- **Compatibilità:** Compatibilità con vari browser e dispositivi per garantire un'esperienza uniforme per tutti gli utenti.
- **Performance:** Tempo di risposta accettabile per le richieste dell'utente, ad esempio per l'apertura dei ticket o la ricerca dei clienti.
- **Scalabilità:** Scalabilità per gestire un numero crescente di utenti e dati.

3 Design

3.1 Metodologie di sviluppo

Per quanto riguarda l'implementazione dell'applicazione, si è posto l'accento sul principio **Mobile First**, il che significa concentrarsi principalmente sulla progettazione e lo sviluppo per dispositivi mobili. Questa scelta è stata guidata a seguito di una specifica richiesta da parte dell'azienda in quanto i tecnici utilizzeranno principalmente degli iPad per compilare i ticket presso i clienti.

Durante la realizzazione del progetto, si è deciso di adottare la metodologia di design denominata **UCD** (*User Centered Design*). L'obiettivo principale è stato mettere al centro le esigenze dell'azienda e garantire un'usabilità ottimale per migliorare l'esperienza complessiva da parte dei tecnici. Gli Utenti a cui si farà riferimento successivamente sono rappresentati attraverso una serie di profili chiamati *personas*, che riflettono il profilo dei tecnici di riferimento dell'applicazione.

È stato deciso di adottare il principio **KISS** (*Keep It Simple, Stupid*) per creare interfacce che contengono solo gli elementi essenziali e che sono facilmente accessibili all'utente. L'obiettivo di questa Web App non è orientato alla mera estetica visiva, ma piuttosto alla semplicità e all'immediatezza d'uso.

3.2 Target User Analysis

Poiché il pubblico di utenti destinato all'applicazione web è decisamente limitato rispetto a un'applicazione web comune, è essenziale che la piattaforma sia progettata con attenzione alle funzionalità necessarie per soddisfare appieno i requisiti specificati dall'azienda.

Il Target User è diviso in tre profili, uno per ogni tipo di tecnico: Base, Junior/Senior e Admin (il tecnico Senior in questo momento viene considerato al livello del tecnico Junior, in quanto si differenzia solo per il prezzo orario ma ha gli stessi privilegi):

- **Base:** si fa riferimento ai dipendenti dell'azienda che stanno fisicamente in sede a fare accoglienza ai clienti fisici e a prestare assistenza telefonica. I loro compiti sono quindi ristretti: devono poter aprire un ticket quando un cliente chiede assistenza. Se il cliente è nuovo per l'azienda devono poterlo registrare.
- **Junior/Senior:** si fa riferimento ai dipendenti dell'azienda che eseguono effettivamente gli interventi tecnici, sia in sede che presso i clienti. Anch'essi possono ricevere chiamate quindi devono aver gli stessi privilegi dei tecnici Base per aprire ticket, inoltre devono poter compilare i ticket aperti quando eseguono un intervento.
- **Admin:** si fa riferimento ai dipendenti dell'azienda di livello superiore agli altri dipendenti. Anch'essi possono eseguire interventi tecnici, inoltre devono poter modificare i prezzi orari degli altri tecnici e i relativi profili.

Una volta definito lo User Target a cui è destinato il servizio, vengono presentati vari *personas*, che rappresentano ipoteticamente gli utenti dell'applicazione e le loro specifiche necessità nel compiere determinate attività.

Personas: Andrea, Alice Andrea è un tecnico Base di Kronos. Alice è una cliente dell'azienda da tempo.

Scenario d'uso: Andrea è in sede a fare accoglienza, Alice entra in sede e chiede aiuto ad Andrea in quanto si è rotto lo schermo del suo iPhone. Andrea vuole aprire un nuovo ticket ma attualmente non sa che tecnico assegnargli, così non lo specifica in modo che appena un tecnico si libera, se lo prenderà a carico.

Sequenza azioni:

1. Andrea accede al sito effettuando il login con le sue credenziali
2. Dalla home vuole aprire direttamente un nuovo ticket

3. Seleziona Alice come cliente del ticket
4. Inserisce la richiesta "Vetro superiore iPhone 14 rotto.", non specifica un tecnico da assegnare e salva il ticket

Personas: Barbara, Bruno Barbara è un tecnico Junior di Kronos. Bruno è un nuovo cliente dell'azienda.

Scenario d'uso: Barbara è in sede nel suo ufficio e riceve una chiamata da Bruno, un nuovo cliente, il quale si presenta e chiede aiuto a Barbara in quanto il suo Macbook ormai è vecchio e vorrebbe acquistarne uno nuovo. Barbara allora inizialmente vuole registrare Bruno come nuovo cliente, successivamente vuole aprire un ticket e assegnargli se stessa in quanto è libera in quel momento.

Sequenza azioni:

1. Barbara accede al sito effettuando il login con le sue credenziali
2. Dalla home vuole aprire registrare direttamente un nuovo cliente
3. Apre un nuovo ticket selezionando Bruno come cliente
4. Inserisce la richiesta "Desiderio acquisto nuovo Macbook"
5. Si assegna lei stessa al ticket e lo salva

Personas: Carlo, Claudia Claudio è un tecnico Admin di Kronos. Claudia è una cliente dell'azienda da tempo.

Scenario d'uso: Carlo si è diretto presso casa di Claudia perchè lei vorrebbe installare un nuovo antivirus sul suo computer ma non ci riesce. Carlo esegue l'intervento installando con successo il nuovo antivirus e vuole compilare il ticket al quale era stato assegnato precedentemente da un tecnico Base. Claudia vuole pagare l'intervento scalando le ore da una tessera precedentemente acquistata da loro. Dopodichè Carlo scrive sulla chat broadcast, sul topic *Interventi completati*, che ha chiuso questo ticket.

Sequenza azioni:

1. Una volta completato l'intervento tecnico Carlo accede al sito effettuando il login con le sue credenziali
2. Consulta l'elenco dei ticket e cerca quello inerente a Claudia.
3. Compila il ticket con le informazioni necessarie, seleziona la tessera come metodo di pagamento e fa firmare il ticket a Claudia
4. Claudio apre la chat broadcast, sul topic desiderato, e scrive che ha chiuso il ticket di Claudia.

3.3 Mockups

Nella fase iniziale di progettazione, stono stati realizzati dei mockup per creare una prima versione grafica dell'applicazione e per presentare un insieme di funzionalità che dovevano esserer incluse. Attraverso queste rappresentazioni visive, sono stati raccolti diversi feedback da parte di un dipendente di riferimento dell'azienda. I mockups sono stati realizzati tramite *Balsamiq Wireframes*[1].

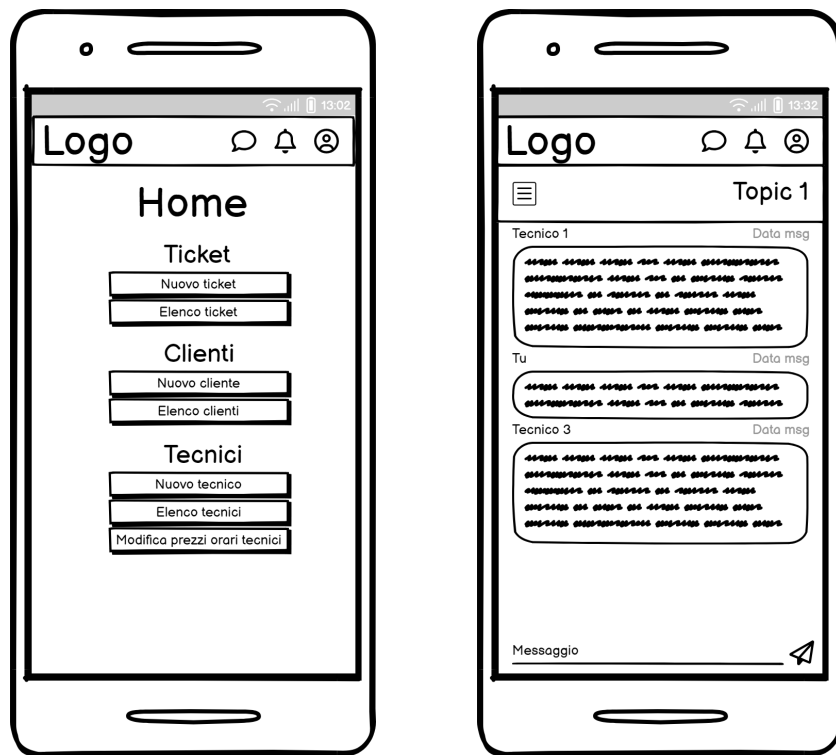


Figure 1: Home e Chat page



Figure 2: Modali per la creazione di nuovi ticket, clienti e tecnici

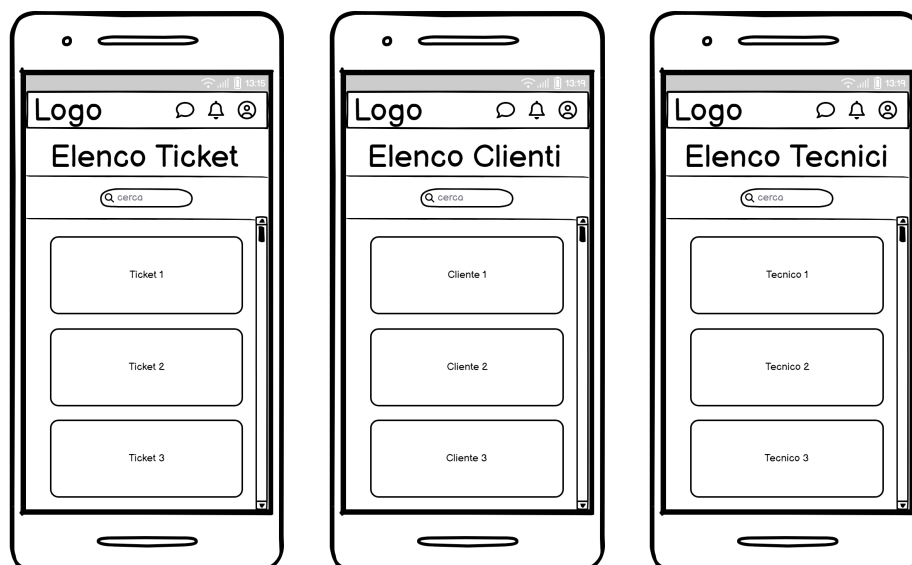


Figure 3: Pagine con elenchi di ticket, clienti e tecnici

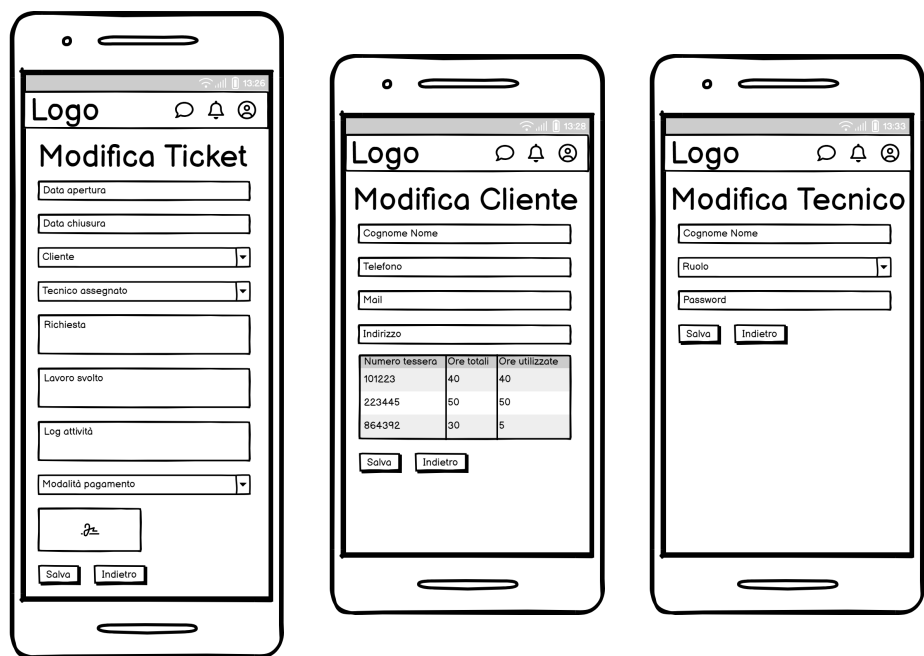


Figure 4: Pagine per la modifica di ticket, clienti e tecnici

3.4 Design architetturale

L'architettura del sistema è stata progettata seguendo il paradigma MEVN, che è un acronimo che comprende le seguenti tecnologie fondamentali:

- **MongoDB**[5]: database
- **Node.js**[3] e **Express**[2]: backend
- **Vue**[4]: frontend

Backend + database Ogni categoria di dati sotto-elencata è stata scorporata in *model*, *controller* e *routes*:

- **Chats**: insieme di chat, ognuna composta da un **topic** identificativo e da un insieme di messaggi.
- **Clients**: insieme di clienti dell'azienda, composti da un **idClient** identificativo, un nome, un numero di telefono, una mail e un indirizzo.
- **Technicians**: insieme di tecnici dell'azienda, composti da un **idTechnician** identificativo, un nome, un ruolo e una password criptata con *SHA256*.
- **Tickets**: insieme di ticket, composti da un **idTicket** identificativo, una data di apertura, una di chiusura, il cliente, la richiesta del cliente, il tecnico assegnato, il lavoro svolto, un log delle attività/note dell'intervento, le ore totali di intervento (ore di trasferimento + ore di lavoro), il metodo di pagamento e la firma del cliente.

Frontend Ogni pagina dell'applicazione è rappresentata da un *Vue Component*, la quale può contenere al suo interno ulteriori components. Questa suddivisione modulare favorisce la manutenibilità ed estendibilità del codice. L'utilizzo di Vue.js permette la reattività dell'applicazione, mantenendo i dati aggiornati automaticamente.

3.5 Schermate finali

L'applicazione finale è stata sviluppata seguendo da vicino i mockup iniziali del progetto. Alcune modifiche sono state apportate in seguito ai feedback forniti dal tecnico di riferimento dell'azienda per migliorarne l'usabilità e la funzionalità.

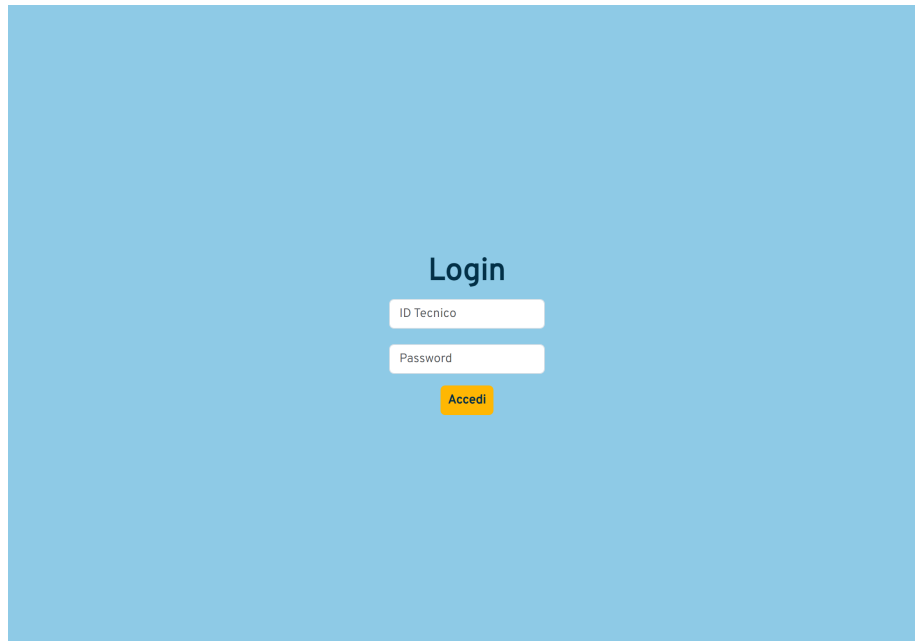


Figure 5: Login page

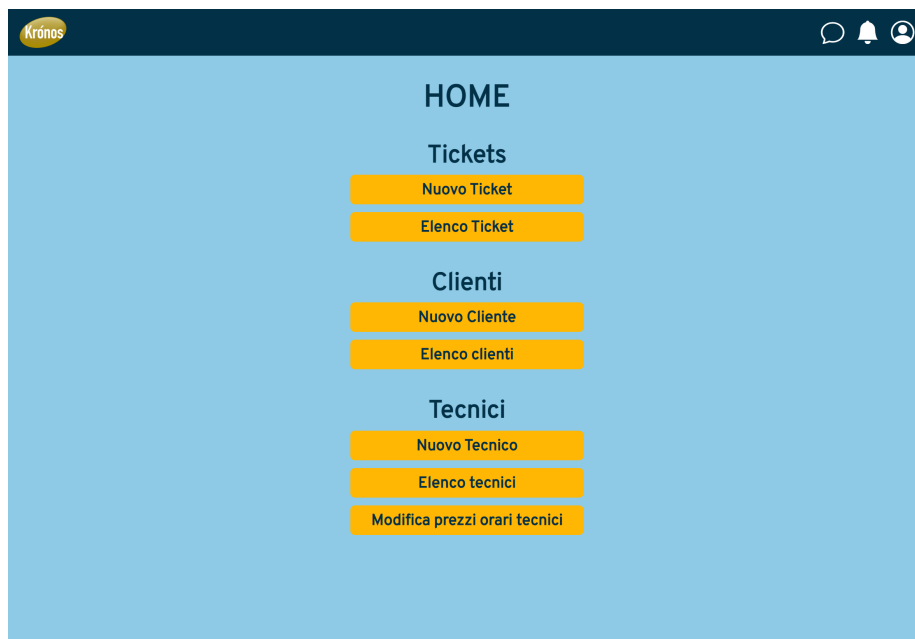


Figure 6: Home page

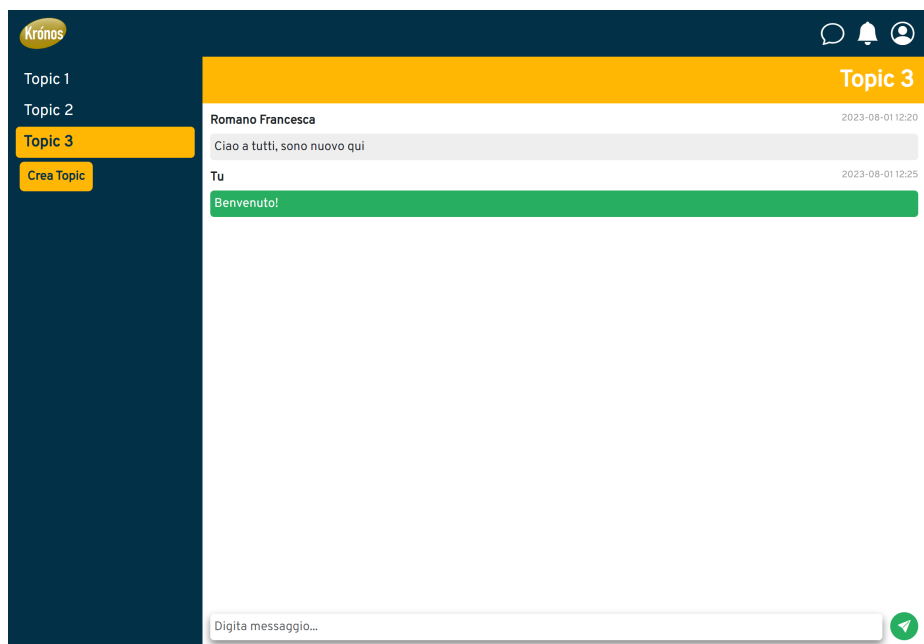


Figure 7: Chat page

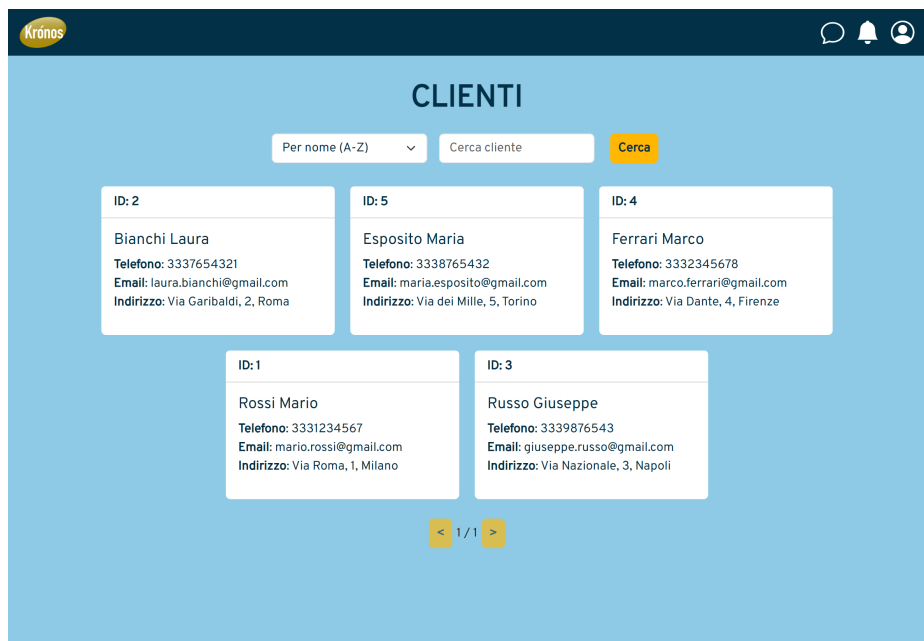


Figure 8: Elenco clienti

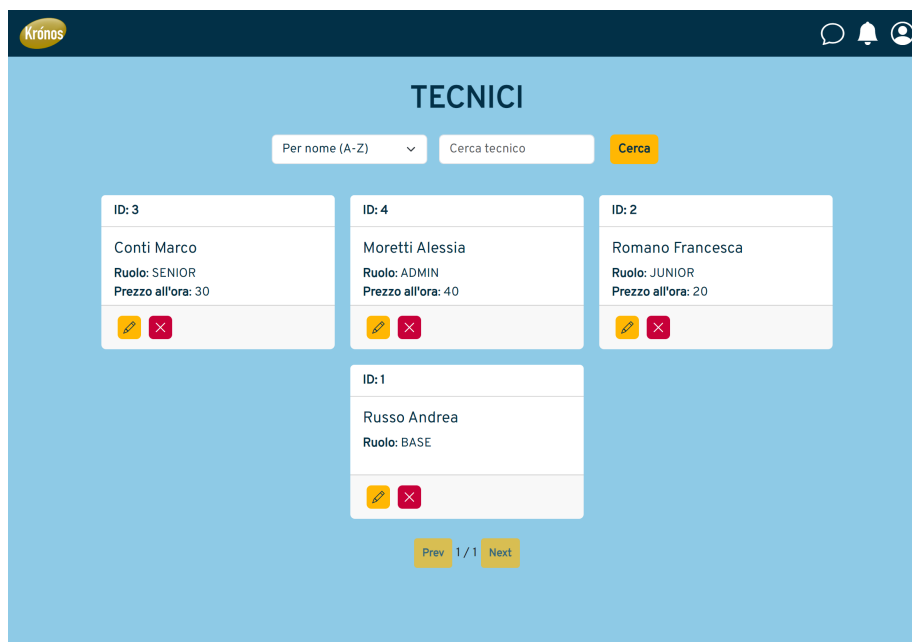


Figure 9: Elenco tecnici

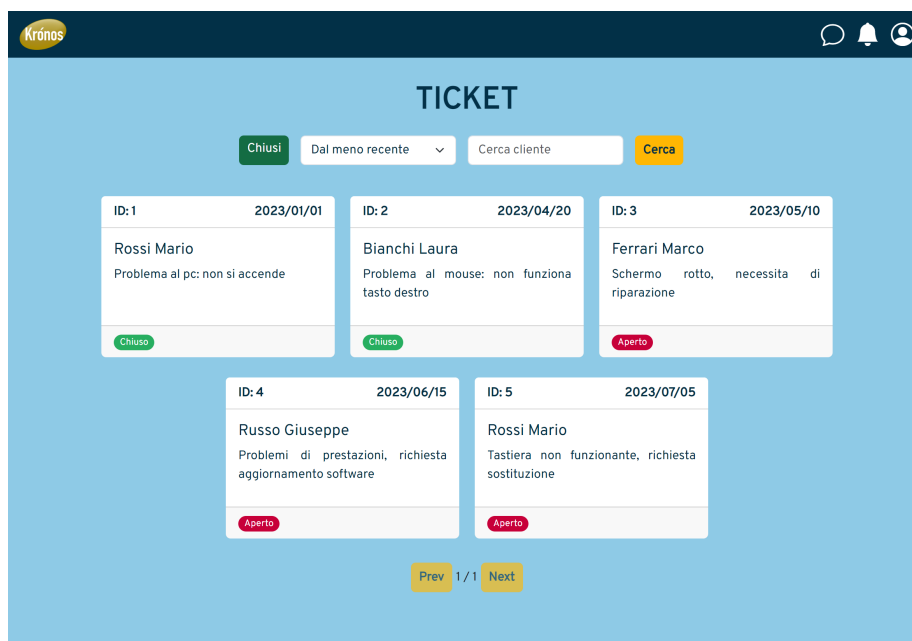


Figure 10: Elenco ticket

The screenshot shows the 'Nuovo Cliente' (New Client) form within the Krónos application. The form is centered on a dark blue background with the word 'HOME' at the top. The form has a title bar with a close button (X). It contains four input fields: 'Nome' (with placeholder 'Cognome Nome'), 'Telefono' (with placeholder '3333333333'), 'Mail' (with placeholder 'example@example.com'), and 'Indirizzo' (with placeholder 'Via, Numero Civico, Città'). At the bottom right of the form are two buttons: 'Aggiungi Cliente' (orange) and 'Chiudi' (blue).

Krónos

HOME

Nuovo Cliente

Nome

Cognome Nome

Telefono

3333333333

Mail

example@example.com

Indirizzo

Via, Numero Civico, Città

Aggiungi Cliente Chiudi

Figure 11: Aggiunta cliente

The screenshot shows the 'Nuovo Tecnico' (New Technician) form within the Krónos application. The form is centered on a dark blue background with the word 'HOME' at the top. The form has a title bar with a close button (X). It contains four input fields: 'Nome' (with placeholder 'Cognome Nome'), 'Ruolo' (a dropdown menu with 'ADMIN' selected), 'Prezzo all'ora' (with placeholder '40'), and 'Password' (with placeholder 'Password'). At the bottom right of the form are two buttons: 'Aggiungi Tecnico' (orange) and 'Chiudi' (blue).

Krónos

HOME

Nuovo Tecnico

Nome

Cognome Nome

Ruolo

ADMIN

Prezzo all'ora

40

Password

Password

Aggiungi Tecnico Chiudi

Figure 12: Aggiunta tecnico

The image shows a web application interface for Kronos. At the top, there is a dark blue header with the Kronos logo on the left and three icons (chat, notifications, user profile) on the right. Below the header, the word "HOME" is centered in a large, bold, white font. A modal form titled "Nuovo Ticket" is displayed in the center. The form has a close button (X) in the top right corner. It contains three sections: "Cliente" with a dropdown menu showing "1 - Doe John"; "Richiesta/segnalazione" with a large text input area; and "Tecnico assegnato (facoltativo)" with a dropdown menu showing "Nessuno". At the bottom of the form are two buttons: "Apri Ticket" (orange) and "Chiudi" (blue). Below the modal, there is a dark blue button labeled "Modifica prezzi orari tecnici".

Kronos

HOME

Nuovo Ticket

Cliente

1 - Doe John

Richiesta/segnalazione

Tecnico assegnato (facoltativo)

Nessuno

Apri Ticket Chiudi

Modifica prezzi orari tecnici

Figure 13: Aggiunta ticket

Krónos

MODIFICA CLIENTE

Dettagli

ID Cliente

4

Nome

Ferrari Marco

Telefono

3332345678

Email

marco.ferrari@gmail.com

Indirizzo

Via Dante, 4, Firenze

Tessere

☒ Mostra tessere completate




Numero tessera	Ore totali	Ore utilizzate		
5	5	3		
6	10	7		
7	20	15		

Salva modifiche

Indietro

Figure 14: Aggiunta cliente

Krónos



MODIFICA TECNICO

Dettagli

ID Tecnico

Nome




Ruolo

Salva modifiche

Indietro

Figure 15: Aggiunta tecnico

Krónos



MODIFICA TICKET

ID Ticket

1

Data apertura

2023/01/01

Data chiusura

2023/01/02

Cliente

1 - Rossi Mario

Tecnico assegnato

1 - Russo Andrea (attuale) ▾

Richiesta del cliente

Problema al pc: non si accende

Lavoro svolto

Sostituito pulsante accensione

Log attività

1. Pulsante accensione guasto, da sostituire

Ore intervento

1

Ore trasferimento

2

Metodo di pagamento

SALDO ▾

Saldo (€)

60

Firma del cliente

John

Cancel

Salva modifiche

Indietro

Figure 16: Aggiunta ticket

4 Tecnologie

In questo capitolo, saranno approfondite le tecnologie aggiuntive utilizzate nell'implementazione dell'applicazione, non facenti parte dello stack tecnologico di base ma rivelatesi essenziali per il completamento del progetto.

4.1 Axios

Axios è una libreria JavaScript basata su *promise* per la gestione delle richieste HTTP. Questa libreria svolge un ruolo fondamentale nella comunicazione tra il client e il server all'interno di un'applicazione reattiva. Consente di eseguire diversi tipi di richieste HTTP, come **GET**, **POST**, **PUT**, **DELETE**, restituendo una *promise* per ciascuna di esse. Questo approccio consente di sfruttare il paradigma delle computazioni asincrone, permettendo di definire delle callback da eseguire solo dopo il completamento della richiesta.

In pratica, utilizzando Axios, è possibile inviare richieste al server e specificare una funzione da eseguire una volta che la risposta dalla richiesta HTTP è disponibile. Questo rende possibile gestire molte interazioni tra client e server in modo reattivo e coerente all'interno dell'applicazione web, garantendo che l'applicazione rimanga responsiva anche durante operazioni di comunicazione con il server.

4.2 Mongoose

Mongoose è stato utilizzato per definire uno schema strutturato per i dati dell'applicazione (chat, clienti, tecnici e ticket). Questo approccio semplifica notevolmente l'implementazione della logica dell'applicazione e fornisce linee guida chiare per interagire con un database documentale come *MongoDB* in modo robusto e coeso.

Mongoose offre la possibilità di definire schemi per i dati e fornisce un set completo di operazioni *CRUD* (*Create*, *Read*, *Update*, *Delete*) che consentono di creare, leggere e modificare i dati memorizzati nel database. Questo permette di gestire le operazioni di accesso ai dati in modo strutturato e intuitivo, garantendo al contempo una coerenza nei dati memorizzati nel database.

4.3 Socket.IO

Socket.IO è una libreria JavaScript che offre un'API per l'implementazione di applicazioni web in tempo reale, consentendo la comunicazione bidirezionale in tempo reale tra client e server. Solitamente, Socket.IO è composta da due parti: una parte gestita lato server e l'altra lato client. Entrambe queste parti creano socket che comunicano utilizzando diversi *topic* o canali per gestire varie interazioni.

Nel contesto di KronosDB, Socket.IO è diventata una componente fondamentale per l'aggiornamento automatico dei ticket, dei clienti, dei tecnici e per la ricezione delle notifiche inerenti alle chat broadcast. Per i tecnici, è cruciale

semplificare il processo di creazione e modifica dei ticket, e Socket.IO consente di vedere immediatamente gli updates. Questo avviene senza la necessità di ricaricare manualmente la pagina per richiedere aggiornamenti, poiché gli aggiornamenti vengono ricevuti automaticamente. Questo ha reso il sistema più efficiente ed efficace.

5 Codice

Di rilevante importanza sono sicuramente le istruzioni dove si è fatto uso di Socket.IO per inviare e ricevere gli updates su chat, clienti, tecnici e ticket. Per semplicità si riporta di seguito solo la parte riguardante la chat.

```
1 exports.sendUpdatedChat = (data) => {  
2   io.emit('CHAT', data);  
3 }
```

Listing 1: Socket.IO server side

```
1 mounted() {  
2   this.socket.on('CHAT', (data) => {  
3     this.chats = data;  
4   });  
5 }
```

Listing 2: Socket.IO client side

6 Test

Progressivamente, durante lo sviluppo dell'applicazione, con l'implementazione di ogni nuova funzionalità, è stata fornita una demo dell'applicazione al tecnico di riferimento dell'azienda. Questo processo ha permesso al tecnico di fornire feedback sia sull'aspetto grafico che sulle funzionalità effettive dell'applicazione. Questa pratica ha garantito un processo di sviluppo iterativo e una comunicazione efficace tra il sottoscritto e il tecnico. Ciò ha contribuito a migliorare continuamente l'applicazione, adottando una prospettiva di miglioramento costante basata sui contributi del tecnico.

7 Deployment

7.1 Setup DB

La creazione del database risulta facilitata utilizzando MongoDB Compass:

1. Effettuare la connessione tramite MongoDB Compass sulla porta 27017
2. Creare un nuovo database con nome "kronosDB"
3. Creare una nuova collezione con nome "chats", una "clients", una "technicians" e una "tickets"

4. Entrare nella collezione "chats", selezionare "ADD DATA" e selezionare il file json a partire dalla root del progetto: `/db/chats.json`
5. Ripetere il procedimento analogo per "clients", "users", "technicians" e "tickets"

7.2 Setup backend + frontend

1. Clonare il repository al seguente indirizzo: <https://github.com/andreazammarchi3/KronosDB>
2. Tramite il terminale spostarsi all'interno della cartella del progetto
3. Spostarsi poi nella cartella *backend* e lanciare il comando `npm install` e ripetere l'operazione nella cartella *frontend*

7.3 Esecuzione backend + frontend

1. spostarsi nella cartella *backend* e lanciare il comando `npm start`
2. spostarsi nella cartella *frontend* e lanciare il comando `npm run dev`
3. aprire il browser e collegarsi su <https://localhost:5173/>
4. per accedere con account Admin: `id = "4"`, `password = "password"`

8 Conclusioni

Lo sviluppo di questo progetto ha rappresentato un'opportunità straordinaria e altamente significativa, in quanto ho avuto la possibilità di entrare in contatto diretto con il mondo aziendale. Senza dubbio, questo progetto si distingue da molti altri per la sua prospettiva di crescita e sviluppo continuo. Ho l'intenzione di mantenere stretti contatti con l'azienda al fine di contribuire alla realizzazione di un prodotto sempre più completo e soddisfacente.

Inoltre, l'adozione dello stack tecnologico MEVN mi ha consentito di interagire con tecnologie ampiamente utilizzate nell'ambito professionale e applicabili in una vasta gamma di progetti. Seguire le lezioni mi ha sicuramente aiutato molto, in quanto come base di partenza del progetto ho utilizzato l'app sui movies realizzata durante i laboratori. Questo approccio mi ha consentito di sviluppare un'applicazione robusta con un'architettura altamente scalabile.

In conclusione, questo progetto ha notevolmente potenziato le mie competenze tecniche, in particolare per quanto riguarda lo sviluppo di applicazioni web moderne.

Bibliography

- [1] <https://balsamiq.com/wireframes/>. Balsamiq wireframes.
- [2] <https://expressjs.com/>. Express.
- [3] <https://nodejs.org/>. Nodejs.
- [4] <https://vuejs.org/>. Vue.
- [5] <https://www.mongodb.com/>. Mongodb.