SoundSphere

INGEGNERIA DEL SOFTWARE

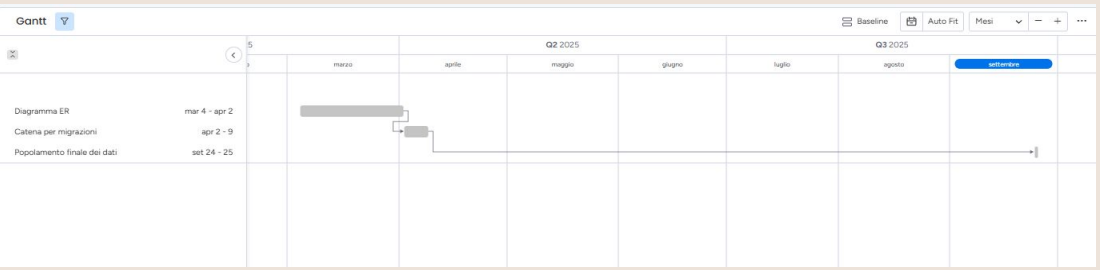
CodeCommanders:
Segala Riccardo
Petrini Jacopo
Raimondi Miriam

Cos’è SoundSphere?

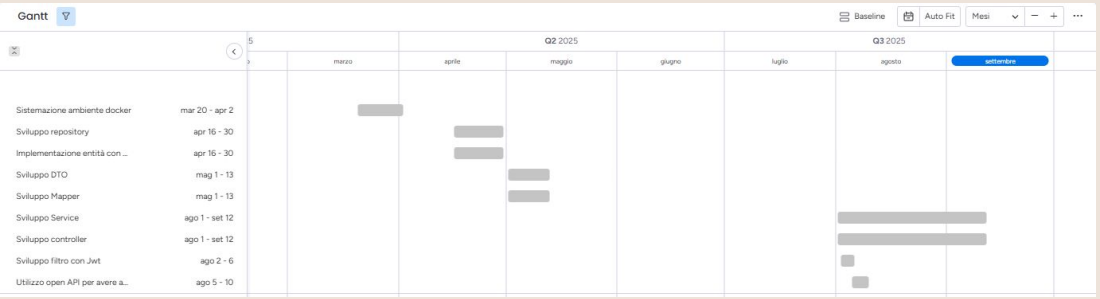
E’ un’app di shop online che si occupa di strumenti musicali e di audio ed è dedicata principalmente ad esperti nel campo ma anche ai principianti che si affacciano per la prima volta al mondo meraviglioso della musica.

Inoltre l’applicazione SoundSphere gestisce l’organizzazione di filiali fisiche con dipendenti e stock per dare la possibilità anche di vedere e comprare in loco gli accessori o gli strumenti richiesti.

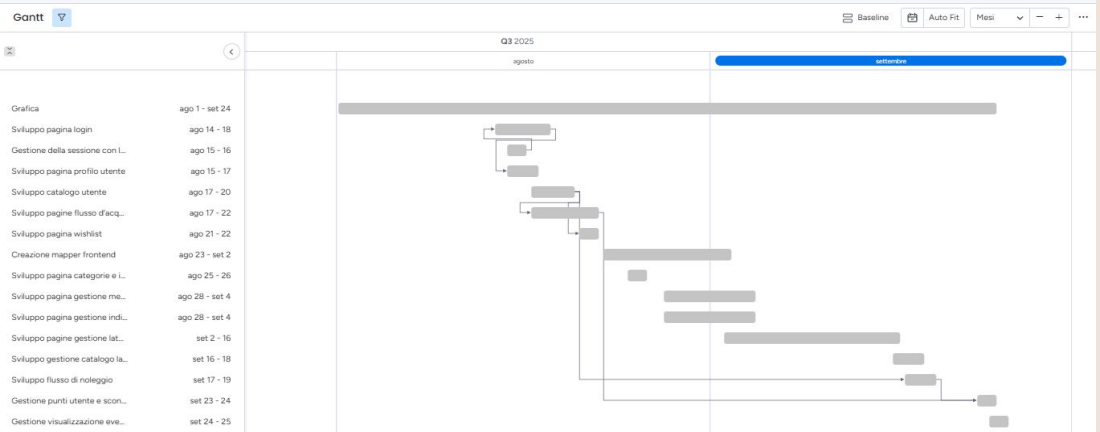
L’ultima categoria, ma non meno importante, che può usufruire di servizi a loro dedicati sono gli Organizzatori di Eventi che possono noleggiare strumenti musicali o di audio per eventi speciali con prezzi vantaggiosi, d’altronde come si dice sempre: Show Must Go On…



DATABASE



BACKEND



FRONTEND

GANTT

TEAM LEADER

Riccardo Segala

FRONTEND DEVELOPER

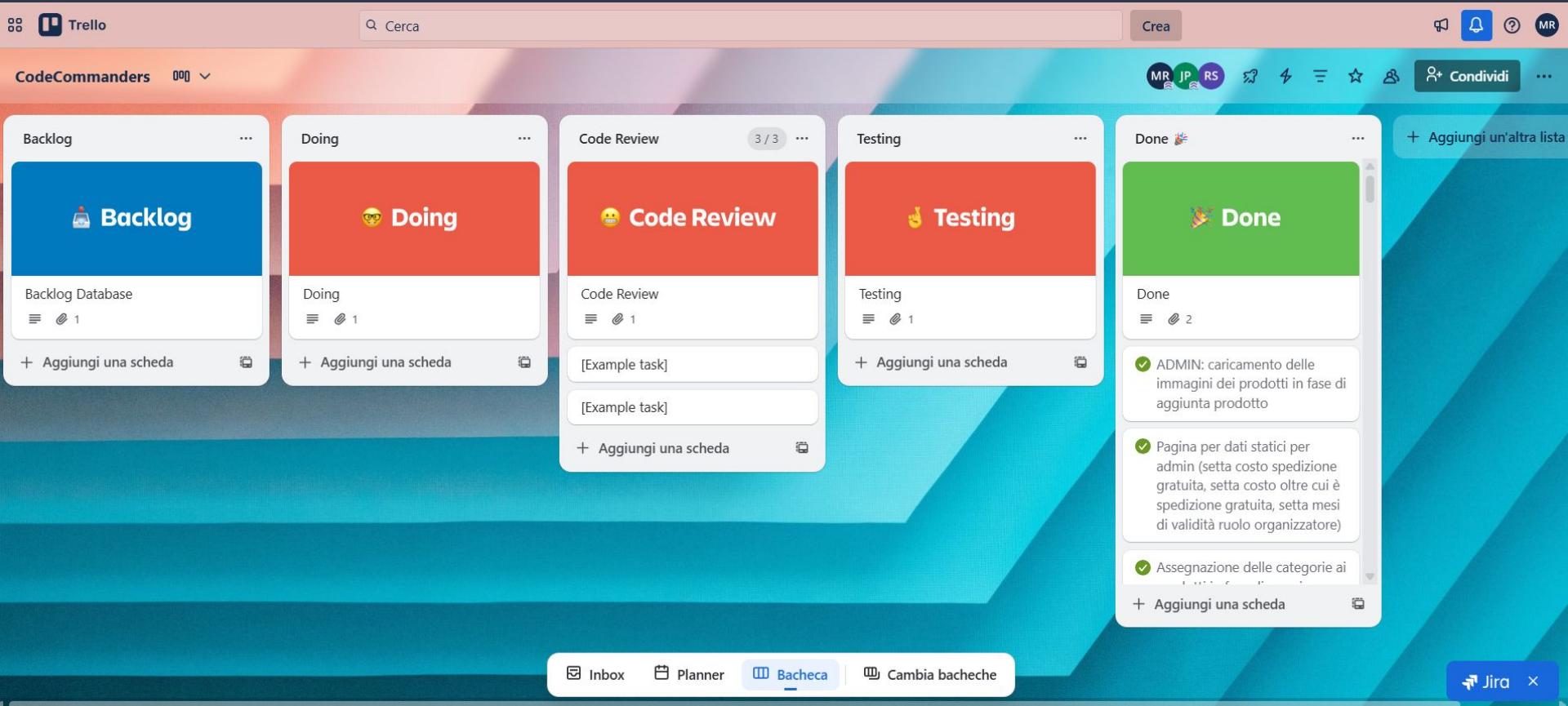
Jacopo Petrini

BACKEND DEVELOPER

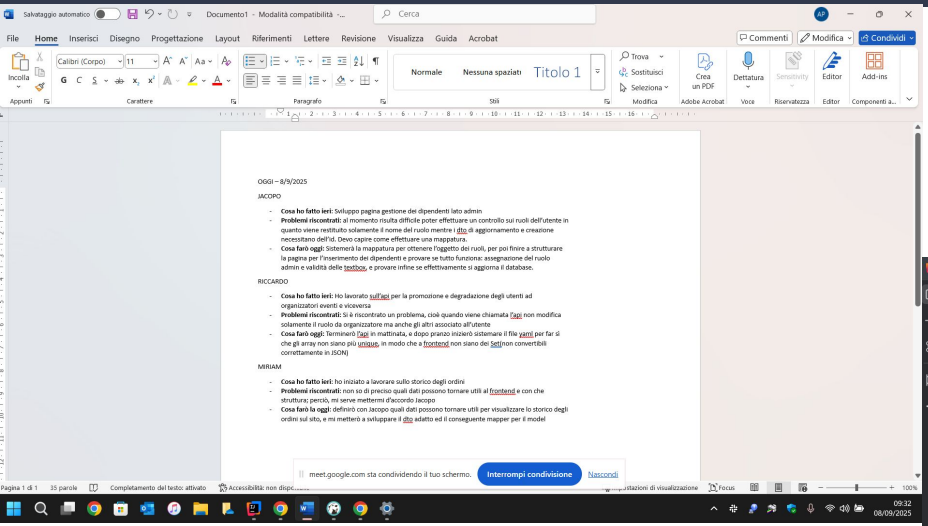
Miriam Raimondi

Distribuzione dei Ruoli nel Team

TRELLO


Trello
Cerca
Crea
CodeCommanders
MR
JP
RS
Condividi
Backlog
Backlog
Backlog Database
Aggiungi una scheda
Doing
Doing
Doing
Aggiungi una scheda
Code Review
3 / 3
Code Review
Code Review
[Example task]
[Example task]
Aggiungi una scheda
Testing
Testing
Testing
Aggiungi una scheda
Done
Done
Done
ADMIN: caricamento delle immagini dei prodotti in fase di aggiunta prodotto
Pagina per dati statici per admin (setta costo spedizione gratuita, setta costo oltre cui è spedizione gratuita, setta mesi di validità ruolo organizzatore)
Assegnazione delle categorie ai
Aggiungi una scheda
Aggiungi un'altra lista
Inbox
Planner
Bacheca
Cambia bacheche
Jira

Cerimonie



OGGI – 8/9/2025

JACOPO

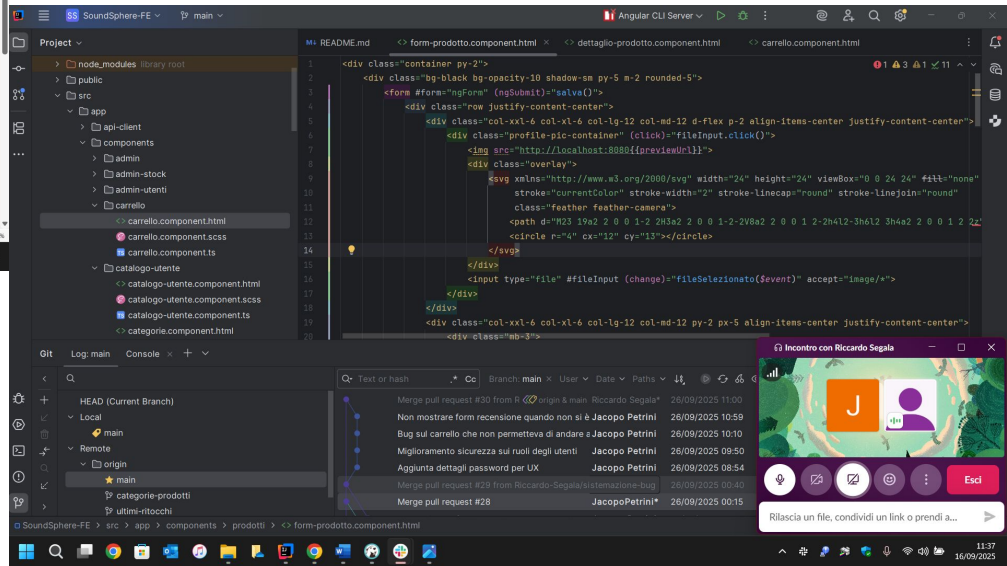
- **Cosa ho fatto ieri:** Sviluppo pagina gestione dei dipendenti lato admin
- **Problemi riscontrati:** al momento risulta difficile poter effettuare un controllo sui ruoli dell'utente in quanto viene restituito solamente il nome del ruolo mentre i dto di aggiornamento e creazione necessitano dell'id. Devo capire come effettuare una mappatura.
- **Cosa farò oggi:** Sistemerò la mappatura per ottenere l'oggetto dei ruoli, per poi finire a strutturare la pagina per l'inserimento dei dipendenti e provare se tutto funziona: assegnazione del ruolo admin e validità delle textbox, e provare infine se effettivamente si aggiorna il database.

RICCARDO

- **Cosa ho fatto ieri:** Ho lavorato sull'api per la promozione e degradazione degli utenti ad organizzatori eventi e viceversa
- **Problemi riscontrati:** Si è riscontrato un problema, cioè quando viene chiamata l'api non modifica solamente il ruolo da organizzatore ma anche gli altri associato all'utente
- **Cosa farò oggi:** Terminerò l'api in mattinata, e dopo pranzo inizierò sistemare il file yaml per far sì che gli array non siano più unique, in modo che a frontend non siano dei Set(non convertibili correttamente in JSON)

MIRIAM

- **Cosa ho fatto ieri:** ho iniziato a lavorare sullo storico degli ordini
- **Problemi riscontrati:** non so di preciso quali dati possono tornare utili al frontend e con che struttura; perciò, mi serve mettermi d'accordo Jacopo
- **Cosa farò la oggi:** definirò con Jacopo quali dati possono tornare utili per visualizzare lo storico degli ordini sul sito, e mi metterò a sviluppare il dto adatto ed il conseguente mapper per il model



Esempio Test cases

```
@Test

void calcolaTotaleParziale_conCarrelloPieno_restituisceSommaCorretta() {
        // Creo un utente e dei prodotti finti per il test
        UUID utenteId = UUID.randomUUID();
        Utente utenteFinto = new Utente();
        utenteFinto.setId(utenteId);

        UUID prodottoId1 = UUID.randomUUID();
        Prodotto chitarra = new Prodotto();
        chitarra.setId(prodottoId1);
        chitarra.setPrezzo(300.0);

        UUID prodottoId2 = UUID.randomUUID();
        Prodotto violino = new Prodotto();
        violino.setId(prodottoId2);
        violino.setPrezzo(500.0);

        Carrello riga1 = new Carrello(utenteFinto, chitarra, 1); // 1 chitarra
        Carrello riga2 = new Carrello(utenteFinto, violino, 2); // 2 violini
        List<Carrello> righeCarrelloSimulate = Arrays.asList(riga1, riga2);

        // c) Configuro il mock
        when(carrelloRepository.findCartByUserId(utenteId)).thenReturn(righeCarrelloSimulate);
        // Chiamo il metodo che voglio testare
        double totaleCalcolato = carrelloService.calcolaTotaleParziale(utenteId);

        // Controllo che il risultato sia quello che mi aspetto
        // Calcolo atteso: (300.0 * 1) + (500.0 * 2) = 300.0 + 1000.0 = 1300.0
        assertEquals(1300.0, totaleCalcolato, "Il totale del carrello non è stato calcolato correttamente.");
}
```

Questo codice testa un caso specifico, ossia quello favorevole, del metodo calcolaTotaleParziale nel CarrelloService che serve per l'appunto a calcolare il costo totale degli oggetti presenti nel carrello dell'utente. Il test si articola principalmente in 3 FASI distinte. Nella PREPARAZIONE vengono preparati degli oggetti finti che devono essere passati al metodo o al mock. Sempre in questa fase, vengono preparati i mock ovvero simulazioni di query al db tramite Repository o di metodi specifici chiamati dal metodo testato che quando verrà eseguito non eseguirà veramente ma gli verranno passati i nostri oggetti finti. La fase di AZIONE prevede l'esecuzione del metodo da testare. E infine la VERIFICA prevede di verificare i dati in uscita e se i metodi sono o non sono stati eseguiti in base al contesto da testare.

Elenco Funzionalità:

- Registrazione
  - Registrazione utenti standard
- Login
- UTENTI NON REGISTRATI (CLIENTE BASE)
  - Home: visualizzazione elenco macro categorie prodotti
  - Accesso al catalogo online
  - Carrello
- UTENTI LOGGATI:
  - Lista preferiti
  - Procedere all'acquisto
  - Vedere storico degli ordini
  - Recensire
  - Visualizzare il carrello
- DIPENDENTE:
  - Funzionalità base dell'utente
  - Gestisce quantità magazzino locale (per acquisti in loco)
- ADMIN:
  - Può vedere e modificare Utenti/Dipendenti/Organizzatori
  - Promuove gli Utenti a Organizzatori
  - Gestisce le filiali e i loro stock
  - Gestione Ruoli da assegnare agli utenti
  - Gestisce i prodotti
- ORGANIZZATORE DI EVENTI:
  - Noleggiare
  - Storico Noleggi
  - Funzionalità utente base

Dettaglio di una funzionalità:

Checkout

L’utente può accedere alla pagina di checkout del sito solamente se loggato nel sito e solamente dalla pagina del carrello, se esso contiene almeno un prodotto. Entrato verranno caricati i dati relativi al proprio carrello, ai propri indirizzi e ai propri metodi di pagamento registrati nel sito. In particolare, i primi verranno disposti sulla destra insieme al costo di spedizione (se carrello inferiore ai 150€) e l’eventuale sconto a cui si ha accesso grazie ai punti accumulati.

A questo punto l’utente potrà scegliere a quale indirizzo far spedire i prodotti (tramite menù a tendina, tag select) e con quale carta effettuare il pagamento. L’elemento settato come di default verrà selezionato come primo. Nel caso in cui si è un organizzatore di eventi e si accede al componente per effettuare il checkout per noleggio, verranno mostrati anche due calendari per selezionare la data di inizio e fine.

Premendo il pulsante “Conferma Acquisto” si invocherà il metodo paga(), il quale in base al modo in cui si è acceduti alla pagina (checkout per ordine o noleggio) preparerà il dto con i dati necessari da inviare al service corrispondente, e una volta pronti, effettua la chiamata http al backend.

A backend viene chiamato il metodo di checkout (Ordini o Noleggi) che verifica l’utente autenticato, l’indirizzo e il metodo di pagamento passato e prende gli oggetti nel carrello e crea rispettivamente un ordine o un noleggio con i relativi dettagli. In seguito viene calcolato il totale finale e per gli ordini vengono aggiunti i punti ottenuti dall’acquisto tramite funzioni implementate anche in altri service delegando così parte del lavoro e rendendo checkout un metodo gestionale.

Quando la chiamata asincrona viene completata si hanno due opzioni:
- andata a buon fine, allora si controlla quale è l’id dell’ordine/noleggio, si chiama un alert per comunicare all’utente il completamento dell’operazione e lo si inoltra alla pagina che contiene lo storico dei suoi ordini, non prima di aver aggiornato i suoi punti ed eventualmente il vantaggio nella variabile di sessione.

- non andata a buon fine, si manda un messaggio di errore nel log.

L’inizio non è stato semplice, mentre i miei compagni facevano cose io stavo preparando altro e mi sono trovato una quantità di informazioni che non conoscevo elevata. In particolare il dover gestire errori con docker, token jwt, migrazioni ecc… è stato davvero snervante e non nego che ad uno punto la stavo anche prendendo male, pensando che tutto questo non fosse necessario. Anche il dover collaborare seriamente per la prima volta è stato strano, dover magari aspettare o far aspettare altri da cui si dipende. Ma con il passare del tempo e le cose che finalmente si appiattivano, alla fine mi è tutto sommato piaciuto.

Jacopo

E’ stato molto interessante non mi era mai capitato di creare da zero un’applicazione web, sicuramente ho molto da migliorare ma mi ha dato anche molto, ho imparato tanti strumenti che non conoscevo!!

Miriam

In passato ho avuto esperienze con sviluppo angular con backend api rest, lo step aggiuntivo che però questa esperienza ha dato è stata la necessità di imparare ad avviare un progetto da zero, imparando nuove tecnologie e scontrandosi con i relativi limiti. Il lavoro di gruppo mi è piaciuto molto, la percezione del gruppo è stata di una bella unione anche di fronte a problematiche strutturali da risolvere. Nel complesso sono rimasto molto contento.

Riccardo

Feedback Esperienza

GRAZIE PER L'ATTENZIONE

CODECOMMANDERS:
Segala Riccardo
Petrini Jacopo
Raimondi Miriam