Collegamento FrontEnd a progetto Ticket

L'architettura dell'applicazione si divide in due componenti principali: il frontend e il backend. Il frontend rappresenta la parte dell'applicazione che viene eseguita direttamente nel browser dell'utente. Questa sezione è composta da tre tecnologie fondamentali: HTML per la struttura delle pagine, CSS per la gestione dello stile e della presentazione visiva, e JavaScript che funge da controller lato client per controllare la pagina e gestire la logica di interazione.

Il controller JavaScript ha il compito di comunicare con le API del backend. Questa comunicazione avviene attraverso l'invio di richieste HTTP verso il server e la successiva ricezione delle risposte. Nel caso specifico del diagramma, quando l'utente interagisce con la pagina per creare un nuovo ticket, il controller JavaScript nel file newTicket.js gestisce questa operazione inviando una richiesta verso il backend.

Il backend rappresenta la parte server dell'applicazione ed è implementato in Java. Espone delle API che ricevono le richieste provenienti dal frontend. Nel caso della creazione di un nuovo ticket, l'API Java attraverso il metodo newTicket() della classe TicketApi processa la richiesta ricevuta. L'operazione di inserimento dei dati viene poi gestita dal backend che si interfaccia con il database MySQL per la persistenza delle informazioni.

Una volta completata l'operazione sul database, il backend genera una risposta che viene inviata al frontend. Il controller JavaScript riceve questa risposta e aggiorna di conseguenza la pagina HTML visualizzata dall'utente, completando così il ciclo di comunicazione tra frontend e backend.
Il processo di sviluppo inizia con la creazione del file denominato newTicket.html, il quale rappresenta l'interfaccia dedicata all'ospite di una stanza per richiedere un nuovo intervento attraverso l'apertura di un ticket. Questa operazione consiste tecnicamente nell'inserimento di un record all'interno del sistema e viene strutturata attraverso il codice HTML.

Per quanto riguarda l'organizzazione dei contenuti statici, all'interno della cartella principale della web application viene definita una directory denominata ticket

service. Al suo interno è prevista una sottocartella chiamata css, destinata a ospitare il file style.css. Quest'ultimo funge da foglio di stile esterno globale per l'intero sito web, garantendo un'estetica coerente a tutte le pagine, pur lasciando la possibilità a ogni singola sezione di integrare ulteriori fogli di stile specifici. Nello specifico, vengono definite le regole grafiche per elementi fondamentali quali body, div, h1 e h2.

La fase successiva prevede la mappatura di questo file verso un controller tradizionale e l'integrazione di una risorsa specifica. Per la gestione della grafica dei campi di inserimento, vengono applicate delle classi di stile denominate standard input. Tale categorizzazione permette di identificare e formattare in modo uniforme tutti gli input appartenenti a questa categoria. In questo contesto, è importante ricordare il concetto di cascading, secondo il quale, in presenza di regole CSS in conflitto tra loro, l'ultima dichiarata prevale sulla precedente.

Infine, per integrare la logica di funzionamento della pagina, viene effettuata l'importazione del file JavaScript tramite il percorso dedicato alla directory js. Tale collegamento avviene richiamando lo script denominato newticket.js, completando così l'architettura che separa struttura, stile e comportamento dinamico.

Il collegamento del foglio di stile avviene mediante l'utilizzo del tag link. Questo elemento deve specificare la relazione come foglio di stile e indicare il percorso corretto verso il file style.css situato nella directory dedicata. Questo permette di applicare le regole grafiche globali e le classi specifiche, come quelle definite per gli input standard, garantendo l'uniformità visiva della pagina.

Per quanto riguarda la logica di interazione, il collegamento del file JavaScript si effettua tramite il tag script. L'attributo sorgente deve puntare al percorso del file newticket.js all'interno della cartella dei contenuti statici. Questo passaggio è fondamentale per consentire al controller lato client di gestire gli eventi della pagina e di inoltrare le richieste verso le API Java del backend.

Nel contesto di un'applicazione Spring Boot, i percorsi devono riflettere la struttura delle cartelle definite sotto la directory delle risorse statiche, affinché il server possa servire correttamente i file al browser al momento del caricamento della pagina.

L'integrazione della logica dinamica all'interno della pagina avviene attraverso il file JavaScript, il quale assume il ruolo di controller per la gestione dell'interfaccia.

Inizialmente, gli elementi grafici come i bottoni non possiedono un comportamento predefinito; è il controller a governare la pagina e a stabilirne le funzionalità.

La creazione di questo controller in JavaScript non richiede necessariamente la definizione di classi formali. È infatti possibile utilizzare un oggetto definito "usa e getta", dichiarato semplicemente attraverso l'apertura delle parentesi graffe. Questo approccio permette di popolare l'oggetto direttamente con le proprietà e i metodi necessari. Un esempio fondamentale di tale implementazione è il metodo denominato checkForm(), inserito all'interno dell'oggetto controller per gestire la validazione o l'elaborazione dei dati inseriti dall'utente prima dell'invio al backend.

Attraverso questa struttura, il file newticket.js è in grado di intercettare le interazioni dell'utente, come il clic sul pulsante di apertura ticket, eseguendo la logica definita nei suoi metodi e coordinando la comunicazione con le API Java.
Il processo di gestione della logica lato client prosegue con l'implementazione del metodo per il controllo dei dati inseriti. JavaScript ha pieno accesso alla pagina attraverso il Document Object Model, che rappresenta la struttura gerarchica del documento HTML e permette di interagire con ogni suo elemento.

Il primo passaggio consiste nel recuperare il riferimento al pulsante di invio utilizzando il metodo dedicato per la ricerca tramite identificativo univoco. Successivamente, viene definita la logica di validazione per determinare se il pulsante debba essere disabilitato o meno. Questa verifica avviene controllando il contenuto dei campi del modulo; se il valore relativo all'apertura o quello relativo alla stanza risultano vuoti, viene assegnato uno stato di disabilitazione. Tale stato viene poi applicato direttamente alla proprietà del pulsante per aggiornare l'interfaccia in tempo reale.

In JavaScript, le funzioni presentano una flessibilità strutturale elevata rispetto ad altri linguaggi: non è necessario dichiarare un tipo di ritorno e una funzione può restituire o meno un valore a seconda della necessità logica. Questa caratteristica deriva dal fatto che JavaScript non è un linguaggio tipizzato, il che permette una gestione dei dati e delle variabili estremamente dinamica durante l'esecuzione dello script.

L'integrazione tra l'interfaccia utente e la logica applicativa avviene attraverso il richiamo dei metodi del controller JavaScript direttamente dal codice HTML. Per garantire un'organizzazione del codice ordinata e professionale, queste funzioni

non vengono dichiarate in modo isolato, ma devono essere raggruppate all'interno di un unico oggetto denominato controller.

L'adozione di un oggetto per contenere la logica della pagina permette di centralizzare il governo dell'interfaccia, facilitando la gestione delle interazioni e delle comunicazioni con il backend. In JavaScript, questa struttura si realizza definendo un oggetto attraverso le parentesi graffe, all'interno del quale vengono inseriti i metodi necessari, come quelli dedicati alla validazione dei moduli o alla gestione degli eventi dei pulsanti.

Questo approccio metodologico assicura che l'HTML funga esclusivamente da struttura, mentre l'oggetto controller JavaScript agisca come l'entità incaricata di gestire il comportamento dinamico e il flusso dei dati della pagina newTicket.html.

Il linguaggio JavaScript interpreta i tag presenti nella pagina HTML come oggetti manipolabili all'interno del Document Object Model. In questa architettura, gli attributi definiti nei tag HTML vengono mappati direttamente come proprietà degli oggetti corrispondenti lato script.

Questa corrispondenza permette al controller JavaScript di interagire con l'interfaccia utente modificando dinamicamente lo stato degli elementi. Ad esempio, l'attributo relativo alla disabilitazione di un pulsante viene gestito come una proprietà booleana dell'oggetto, consentendo di attivare o disattivare l'interazione in base alla logica di validazione definita. Poiché JavaScript non è un linguaggio tipizzato, queste proprietà possono essere lette e aggiornate senza la necessità di definire tipi di dato rigidi, facilitando la sincronizzazione tra la struttura della pagina e il suo comportamento dinamico.

L'architettura del controller JavaScript viene ulteriormente definita attraverso l'integrazione di proprietà e metodi dedicati alla comunicazione con il backend. All'interno dell'oggetto controller viene inserita la proprietà URL, che contiene l'indirizzo dell'endpoint relativo al servizio ticket. Essendo il controller un oggetto, esso può ospitare contemporaneamente variabili di stato e logica funzionale.

Il metodo callAPI ha il compito di gestire il trasferimento dei dati. Inizialmente, viene creato un oggetto temporaneo composto dalle proprietà opening e room, i cui valori sono estratti direttamente dai campi del modulo HTML. Questo oggetto rappresenta il corpo della richiesta, che deve essere inoltrata utilizzando il metodo HTTP POST. Prima dell'invio, l'oggetto viene convertito in una stringa tramite

l'istruzione JSON.stringify, trasformandolo nel formato compatibile per la ricezione lato server, dove l'API Java lo interpreterà come un oggetto di tipo ticket.

La trasmissione effettiva avviene mediante l'istruzione fetch, la quale esegue una richiesta HTTP verso l'indirizzo specificato nella proprietà URL dell'API. Questa funzione agisce come il ponte tecnologico che permette al frontend di inviare i dati strutturati al server per la persistenza finale nel database.