



4OURSQUARED

4oursquared.unipd@gmail.com

Lumos Minima
Imola Informatica

Specifica Tecnica

Informazioni

<i>Redattori</i>	Soldà Matteo
<i>Versione</i>	0.0.3
<i>Uso</i>	esterno

Descrizione

Questo documento descrive la specifica architettuale del progetto, illustrando gli aspetti progettuali che caratterizzino il design del prodotto.



Versione	Data	Redattore	Verificatore	Descrizione
0.0.3	22/09/2023	Soldà Matteo	Alberti Nicolas	Stesura dei dati relativi ai test
0.0.2	21/09/2023	Soldà Matteo	Alberti Nicolas	Stesura delle parti mancanti esclusi dati relativi ai test
0.0.1	20/09/2023	Soldà Matteo	Alberti Nicolas	Stesura delle prime sezioni.
0.0.0	05/09/2023	Soldà Matteo	Alberti Nicolas	Prima stesura.



Contents

1	Introduzione	1
1.1	Scopo del Documento	1
1.2	Riferimenti Normativi	1
1.3	Riferimenti Informativi	1
2	Tecnologie Utilizzate	2
2.1	Frontend	2
2.2	Backend	2
2.3	Database	2
2.4	Meccanismo di Comunicazione	2
3	Architettura Logica	3
3.1	Frontend	3
3.2	Backend	3
4	Design Pattern	4
5	Altri Aspetti Progettuali Rilevanti	5
5.1	Persistenza dei Dati	5
5.2	Interfacciamento tra Sensori e Lampioni	6
5.3	Comunicazione Automatica	6
5.4	Autenticazione e Autorizzazione tramite JWT	7
5.5	Client e Smart Components	10
5.6	Sicurezza	10
6	Requisiti e Soddisfacimento	11
6.1	Tabella dei Requisiti	11
6.2	Requisiti di Qualità	13
6.3	Code Coverage	14



List of Figures

1	Architettura del client.	3
2	Architettura del server.	3
3	Rappresentazione grafica dell'architettura MERN.	4
4	Struttura del database.	5
5	Struttura dei dati nel DB.	6
6	Diagramma di flusso del segnale relativo al movimento rilevato dai sensori.	7
7	Diagramma di flusso del meccanismo di autorizzazione tramite JWT.	8
8	Diagramma di flusso del meccanismo di autorizzazione tramite middleware.	9
9	Diagramma di flusso del meccanismo di re-render.	10
10	Code coverage del frontend.	14
11	Code coverage del backend.	14



1 Introduzione

1.1 Scopo del Documento

Questo documento ha lo scopo di descrivere nel dettaglio, soprattutto tramite diagrammi, le caratteristiche architetture del prodotto software sviluppato.

1.2 Riferimenti Normativi

- Capitolato C2 - Lumos Minima

1.3 Riferimenti Informativi

- Design Pattern Architetture
- Dependency Injection
- MVC e Derivati
- Pattern Creazionali
- Software Architecture
- Pattern Strutturali
- Pattern Comportamentali
- Programmazione SOLID



2 Tecnologie Utilizzate

2.1 Frontend

Per realizzare il frontend, ossia la parte di applicazione che viene eseguita sul browser dell'utente, sono state utilizzate le seguenti tecnologie:

- React: libreria JavaScript per la creazione di interfacce utente;
- Typescript: linguaggio di programmazione che estende JavaScript aggiungendo i tipi, permettendo una codifica più robusta e sicura;
- Bootstrap: framework CSS per la creazione di interfacce responsive e accattivanti;

2.2 Backend

Per realizzare il backend, ossia la parte di applicazione che viene eseguita sul server, sono state utilizzate le seguenti tecnologie:

- Node.js: runtime JavaScript che permette di eseguire codice JavaScript lato server;
- Typescript: linguaggio di programmazione che estende JavaScript aggiungendo i tipi, permettendo una codifica più robusta e sicura;
- Express: framework che permette di creare applicazioni web e API più facilmente e con una miglior gestione;
- Axios: client HTTP basato su promise per effettuare richieste HTTP basate su Promise;
- Mongoose: libreria che permette di gestire in modo più semplice e intuitivo i database MongoDB;
- Cors: middleware che permette di configurare in modo semplice e veloce le politiche CORS;
- JWT: libreria che permette di gestire in modo semplice e veloce i token JWT;
- Cron: libreria che permette di gestire in modo semplice e veloce i cronjob, ossia per la creazione di routine automatiche;

2.3 Database

Il database utilizzato è di tipo NoSQL, in particolare MongoDB, che permette di gestire documenti in formato BSON (Binary JSON).

2.4 Meccanismo di Comunicazione

Tutte le comunicazioni, sia esterne (da client a server) che interne (da server a server) sono state gestite tramite API REST.

3 Architettura Logica

3.1 Frontend

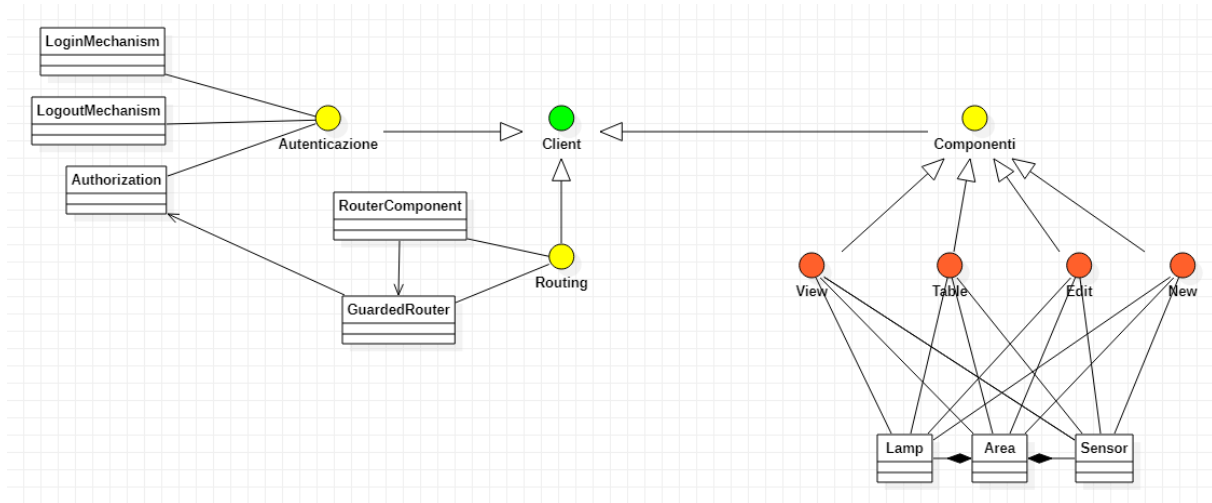


Figure 1: Architettura del client.

3.2 Backend

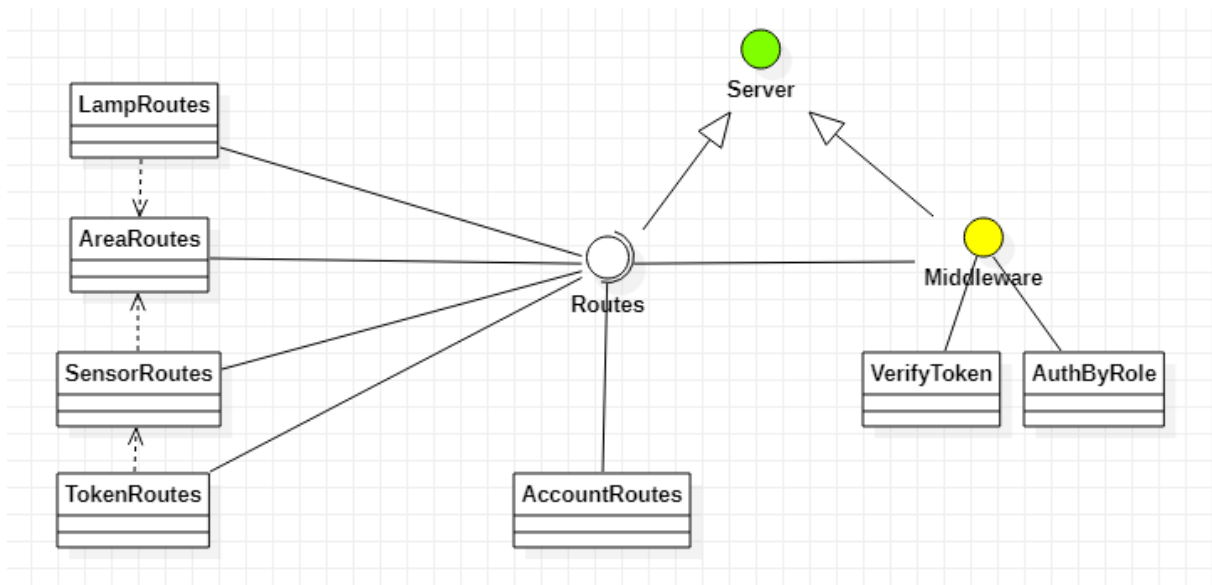


Figure 2: Architettura del server.

4 Design Pattern

Lumos Minima è un'applicazione web di tipo SPA (Single-page application) per la gestione di impianti luminosi e di sensori per la rilevazione del movimento associati ai primi, raggruppati in insiemi definiti aree illuminate.

L'applicazione si avvale del cosiddetto stack MERN (dove le iniziali stanno rispettivamente per MongoDB, Express.JS, React.JS, Node.JS) e presenta un design architetturale "3-tier" dotata di un comparto client (presentazione), un server ("business logic"), e un database No-SQL per ospitare i dati (persistence). Tuttavia, invece di JavaScript "puro" si è preferito TypeScript, che viene compilato in JavaScript e fornisce un supporto opzionale della tipizzazione stretta. Il diagramma mostra le tecnologie di cui si serve ciascun layer.

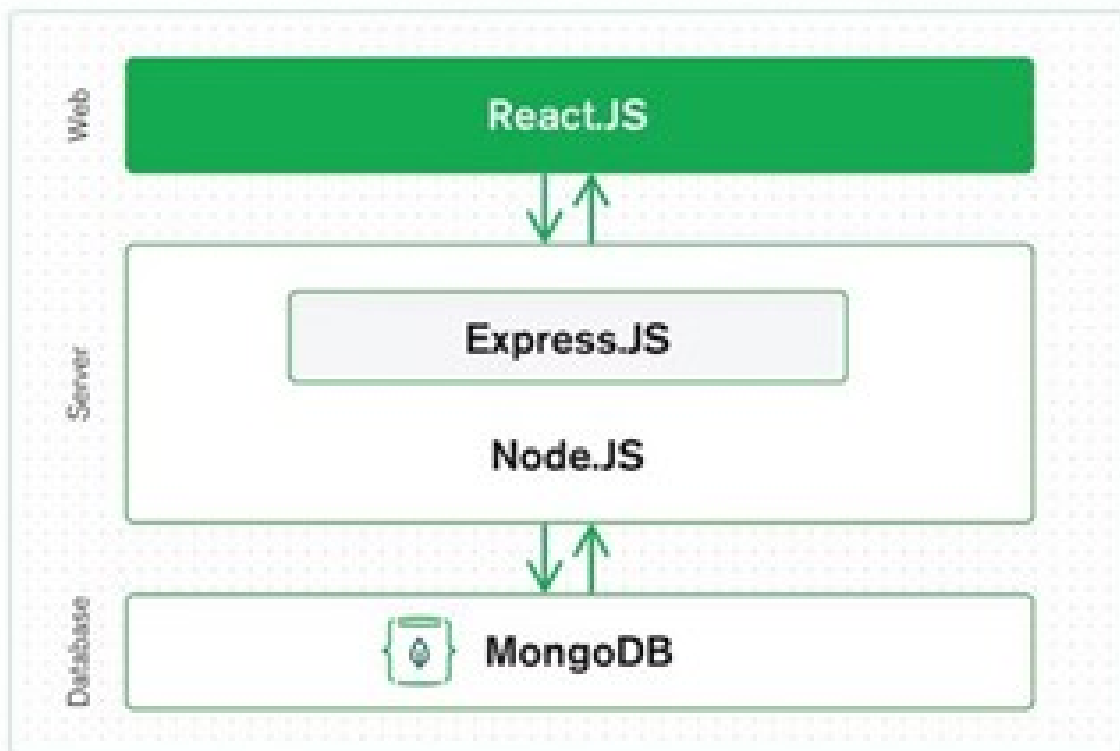


Figure 3: Rappresentazione grafica dell'architettura MERN.

Questo sistema "a livelli" è stato scelto poiché, oltre a trattarsi di un sistema consolidato (e quindi che si prestasse a uno sviluppo anche gravato da vincoli temporali stringenti), consente in modo agevole il mock del layer sottostante in occasione dei test di unità, vista la necessità di raggiungere la percentuale di code coverage inizialmente indicata nel capitolato.

5 Altri Aspetti Progettuali Rilevanti

5.1 Persistenza dei Dati

Per realizzare lo strato di persistenza dei dati, come precedentemente indicato, è stato utilizzato MongoDB, un database NoSQL che permette di gestire documenti in formato BSON (Binary JSON).

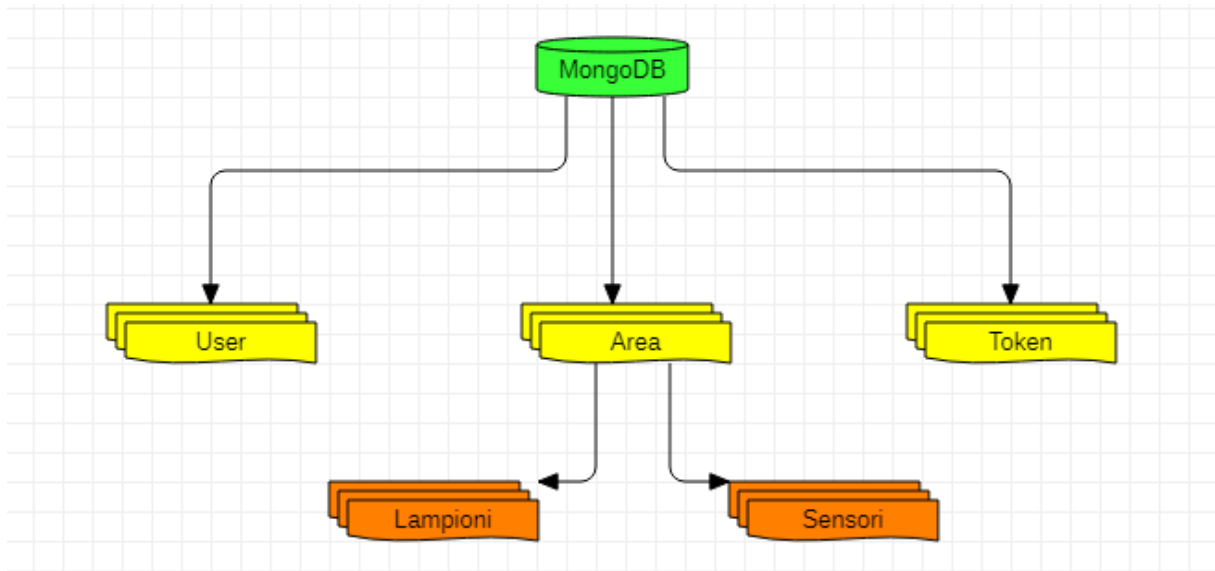


Figure 4: Struttura del database.

A differenza di un database di tipo SQL, in MongoDB, i dati sono raccolti in *collezioni*. Queste *collezioni* contengono uno, nessuno o una moltitudine di *documenti*.

Dato che nei database di tipo NoSQL non esiste il concetto di "relazione", ogni documento di tipo area conterrà un array di documenti di tipo sensore e un array di documenti di tipo lampione.

Per quanto riguarda invece la comunicazione tra il server e il database, per poter comunicare i dati tra una parte e l'altra, sono stati utilizzati gli schemi di *Mongoose* che permettevano di definire la struttura dei dati, i tipi di dati, i vincoli e le validazioni, oltre a fornire una interfaccia che permettesse di comunicare con il giusto tipo di dati.

Di seguito la struttura dei documenti (e conseguentemente degli schemi) utilizzati:

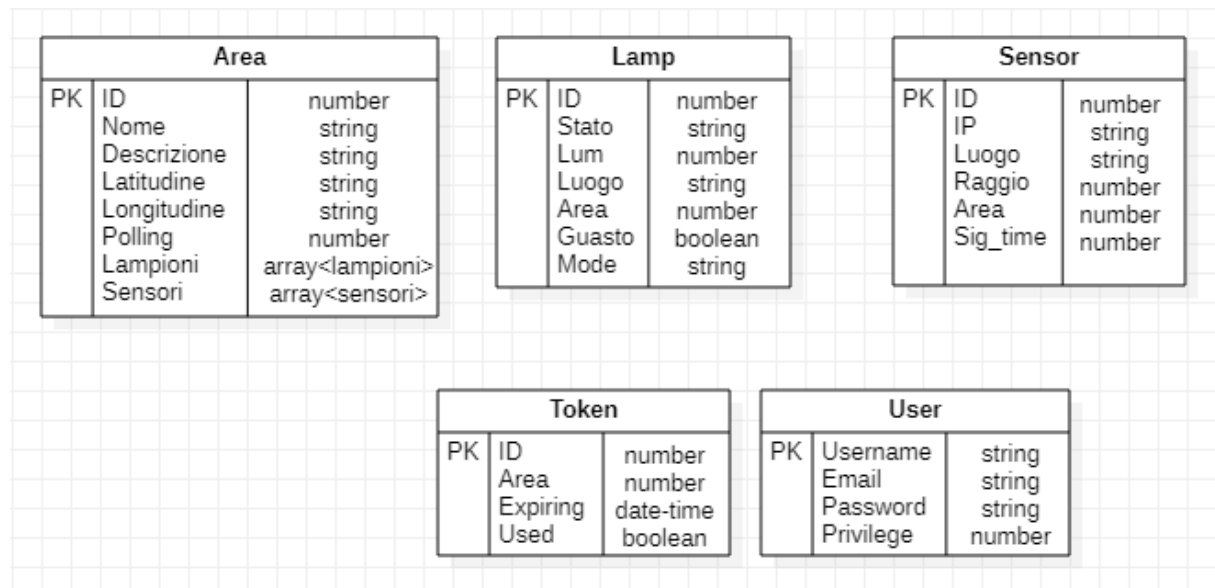


Figure 5: Struttura dei dati nel DB.

5.2 Interfacciamento tra Sensori e Lampioni

Per accedere ai dati riguardanti un determinato lampione e/o sensore, abbiamo utilizzato la lettura di parametri dalla URL.

Ovviamente gli endpoint relativi ai sensori e ai lampioni sono distinti, infatti:

- Endpoint sensori: `/api/aree/<IdArea>/sensori/<IdSensore>`;
- Endpoint lampioni: `/api/aree/<IdArea>/lampioni/<IdLampione>`;

5.3 Comunicazione Automatica

Per quanto riguarda il meccanismo che permette di aumentare e diminuire automaticamente la luminosità dei lampioni che si trovano dentro una determinata area illuminata, il soggetto principale sono i sensori: essi, qualora rilevassero il movimento di una persona, di un veicolo o di un animale, dovrebbero inviare una richiesta di tipo HTTP all'endpoint prestabilito. Alla ricezione della richiesta, il server si occuperà della gestione della stessa come descritto nel diagramma sottostante.

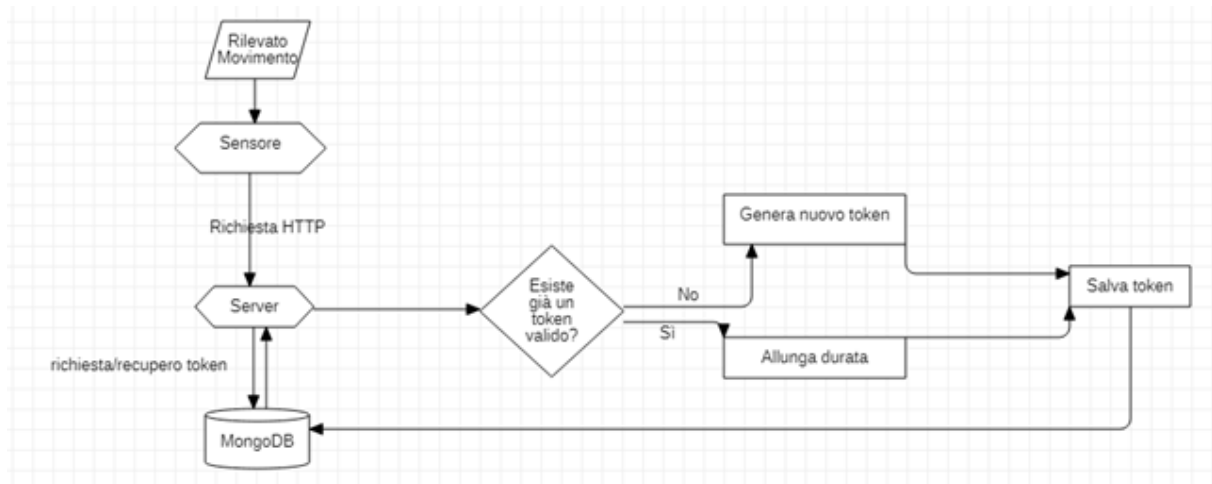


Figure 6: Diagramma di flusso del segnale relativo al movimento rilevato dai sensori.

Il token trova un utilizzo concreto solo per i lampioni dell'area che sono configurati in modalità PULL (manuale).

Per questo motivo, di seguito c'è una piccola spiegazione di come viene gestito il rilevamento di movimento da entrambi i tipi di lampione:

- PUSH: i lampioni si illuminano quando il sensore segnala un movimento, senza necessità di utilizzare il token
- PULL: la parte di server che gestisce l'area illuminata, con cadenza regolare e definita dall'attributo *polling time*, controlla nel database se c'è un token valido per l'area di riferimento:
 - Se il token è valido e inutilizzato: illumina tutti i lampioni impostati in modalità PULL;
 - Se il token non è valido ma è stato utilizzato: riduce la luminosità di tutti i lampioni a quella iniziale;
 - Negli altri casi, verrà restituito un codice stato consono, ma senza effettuare nessun'altra azione.

5.4 Autenticazione e Autorizzazione tramite JWT

Dopo aver inserito le proprie credenziali nella login mask, se la password (criptata con hashing SHA512) coincide con quella dell'utente viene restituito un token JWT firmato dal server, una stringa che contiene informazioni sul ruolo dell'utente. L'autorizzazione (necessaria per l'accesso alle pagine dal lato "client", e per utilizzare le API del server) sfrutta questo meccanismo:

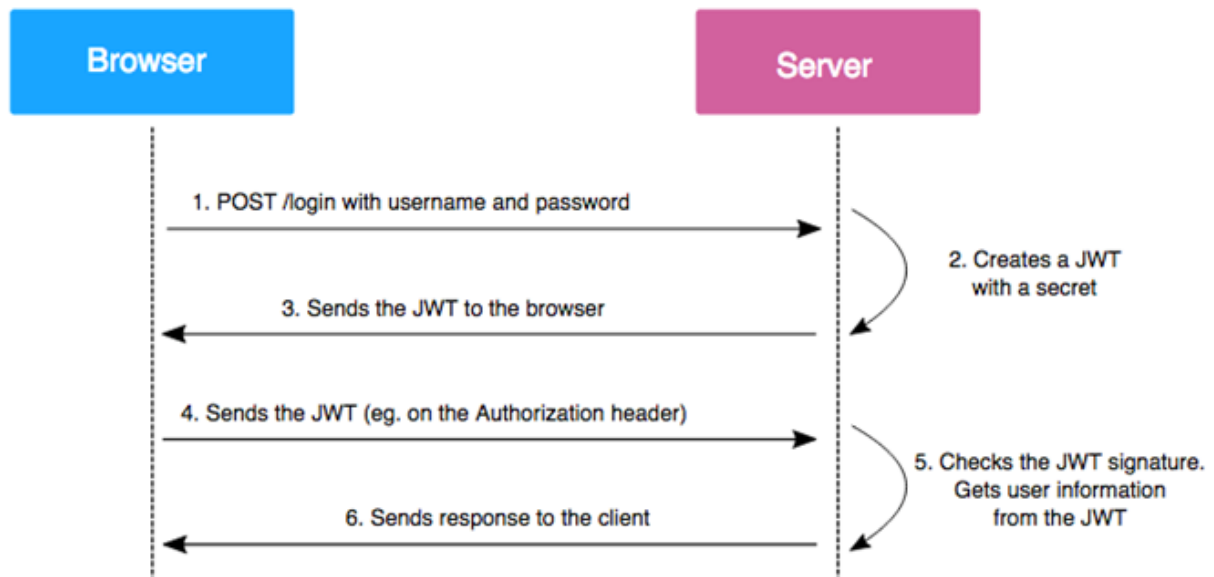


Figure 7: Diagramma di flusso del meccanismo di autorizzazione tramite JWT.

Tuttavia, il JWT è memorizzato (e inviato al server) all'interno di un cookie HTTP-Only: questo impedisce che il contenuto del cookie possa essere letto da codice JavaScript malevolo, vanificando così attacchi alla confidenzialità di tipo XSS (Cross-Site Scripting).

Le API sono protette da due livelli di middleware che prendono in carico la richiesta HTTP.

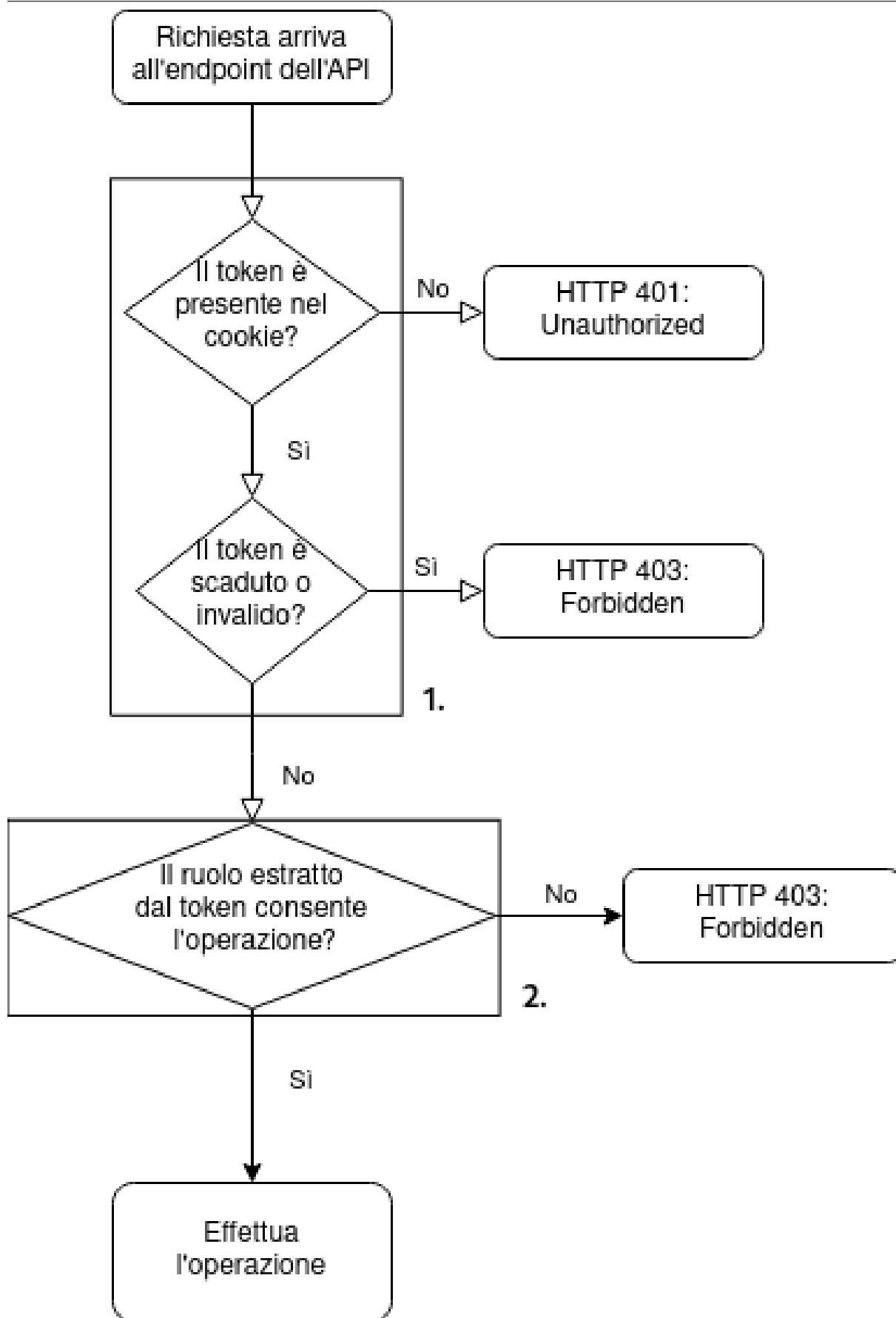


Figure 8: Diagramma di flusso del meccanismo di autorizzazione tramite middleware.

5.5 Client e Smart Components

Il layer di presentazione in React.JS adopera Smart, o “Stateful” Components, ovvero dei moduli - uno per ciascun macro-elemento dell'app - aventi come responsabilità, oltre al semplice rendering a schermo, la gestione di uno stato interno. Questa scelta si presta alla naive hierarchical architecture (S. Nelson):

- Quando richiesto dall'utente, avviene un render iniziale del componente;
- In maniera asincrona (non bloccante), i dati vengono quindi reperiti dal business layer tramite lo “hook” `useEffect`;
- Nello hook, i dati appena reperiti vengono immessi nello stato locale con la funzione `setState`: ciò comporta un re-render che mostrerà tali dati;
- In presenza di un input dell'utente viene trasmessa la modifica al business layer, e (qualora questa abbia successo) in concomitanza avviene l'aggiornamento “locale” dello stato del componente con la stessa funzione `setState`, che ne comporta così un re-render che rispecchia la modifica.

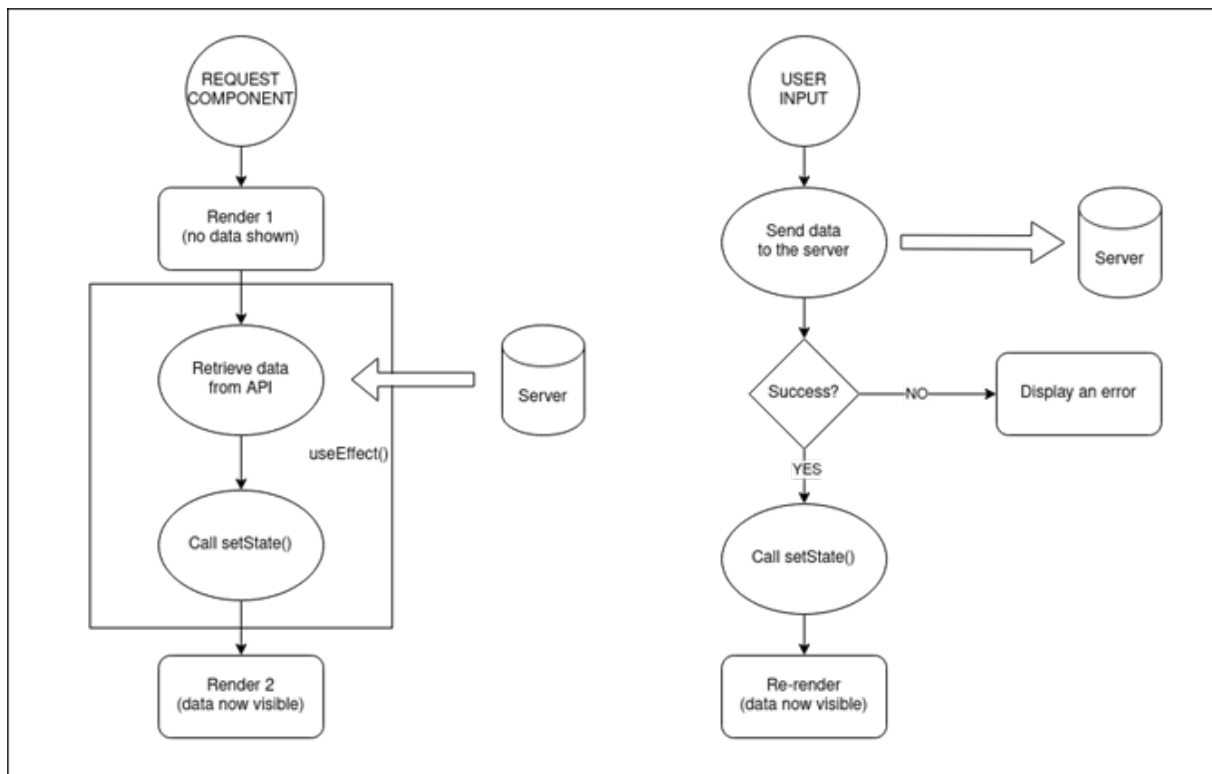


Figure 9: Diagramma di flusso del meccanismo di re-render.

5.6 Sicurezza

Per assicurarsi di evitare gravi problemi legati alla sicurezza, sono stati implementati dei moduli all'interno di *GitHub* che ad ogni *push* o *pull request* controllano che il codice sia conforme a determinati standard.

Gli strumenti utilizzati sono stati:

- Snyk
- SonarCloud
- GitHub CodScanning



Prima di poter effettuare il merge nei branch *dev* (relativo allo sviluppo) oppure nel branch *main* (relativo alle versioni stabili del prodotto), era necessario che tutti i controlli venissero superati, che non fosse presente una duplicazione del codice superiore al 3% e che non fossero presenti *code smells*.

6 Requisiti e Soddisfacimento

6.1 Tabella dei Requisiti

Requisiti funzionali			
Codice	Descrizione	Rilevanza	Soddisfatto
RF1-O	Rilevamento della presenza di individui in una delle aree illuminate.	Obbligatorio	SI
RF2-O	Acquisizione dell'intensità luminosa e successiva determinazione precisa del livello di luminosità.	Obbligatorio	SI
RF3-O	Una volta acquisito il livello di luminosità iniziale, l'operatore autenticato aumenta manualmente il livello di intensità luminosa.	Obbligatorio	SI
RF4-O	Una volta acquisito il livello di luminosità iniziale, l'operatore autenticato diminuisce manualmente il livello di intensità luminosa.	Obbligatorio	SI
RF5-O	Il gestore può effettuare l'accesso per gestire manualmente i sistemi di illuminazione.	Obbligatorio	SI
RF6-O	Il gestore può effettuare il logout dall'interno dell'area di gestione dei sistemi.	Obbligatorio	SI
RF7-O	Il gestore può consultare l'intero elenco delle aree illuminate.	Obbligatorio	SI
RF8-O	Il gestore può consultare l'intero elenco delle aree illuminate con guasti.	Obbligatorio	SI
RF9-O	Il gestore può aggiungere manualmente un guasto selezionando un impianto dalla lista di quelli attivi.	Obbligatorio	SI
RF10-O	Il gestore può creare una nuova area illuminata, inserendone la posizione geografica e i relativi dettagli.	Obbligatorio	SI
RF11-O	Il gestore può modificare i dettagli di un'area illuminata già esistente.	Obbligatorio	SI
RF12-O	Il gestore può rimuovere un'area illuminata già esistente.	Obbligatorio	SI
RF13-O	Il sistema di gestione dell'illuminazione aumenta l'intensità luminosa al passaggio di una o più persone.	Obbligatorio	SI



RF14-O	Il sistema di gestione dell'illuminazione diminuisce l'intensità luminosa al passaggio di una o più persone.	Obbligatorio	SI
RF15-O	Il gestore può impostare la modalità di funzionamento automatico per l'impianto selezionato.	Obbligatorio	SI
RF16-O	Il gestore può rimuovere un'area illuminata dall'elenco delle aree illuminate con guasti e ritorna nelle aree illuminate attive.	Obbligatorio	SI
RF17-O	Inserimento di un nuovo sensore in un'area per essere gestito dal sistema.	Obbligatorio	SI
RF18-O	Il gestore può rimuovere dal sistema uno dei sensori registrati.	Obbligatorio	SI
RF19-O	Il gestore può inserire nel sistema l'impianto di illuminazione da attivare con i relativi dettagli.	Obbligatorio	SI
RF20-O	Il gestore può rimuovere dal sistema uno degli impianti luminosi registrati.	Obbligatorio	SI
RF21-O	Rilevamento della presenza di individui in modalità automatica in una delle aree illuminate	Obbligatorio	SI
RF22-F	Rilevamento della presenza di individui su richiesta in una delle aree illuminate	Facoltativo	SI
RF23-O	Una volta acquisito il livello di luminosità iniziale, l'operatore autenticato aumenta manualmente il livello di intensità luminosa di un'area illuminata	Obbligatorio	SI
RF24-F	Una volta acquisito il livello di luminosità iniziale, l'operatore autenticato aumenta manualmente il livello di intensità luminosa di più aree illuminate	Facoltativo	NO
RF25-O	Una volta acquisito il livello di luminosità iniziale, l'operatore autenticato diminuisce manualmente il livello di intensità luminosa di un'area illuminata	Obbligatorio	SI
RF26-F	Una volta acquisito il livello di luminosità iniziale, l'operatore autenticato diminuisce manualmente il livello di intensità luminosa di più aree illuminate	Facoltativo	NO
RF27-O	Una volta acquisito il livello di luminosità iniziale, il sistema aumenta automaticamente il livello di intensità luminosa di un'area illuminata	Obbligatorio	SI
RF28-F	Una volta acquisito il livello di luminosità iniziale, il sistema aumenta automaticamente il livello di intensità luminosa di più aree illuminate	Facoltativo	NO



RF29-O	Una volta acquisito il livello di luminosità iniziale, il sistema diminuisce automaticamente il livello di intensità luminosa di un'area illuminata	Obbligatorio	SI
RF30-F	Una volta acquisito il livello di luminosità iniziale, il sistema diminuisce automaticamente il livello di intensità luminosa di più aree illuminate	Facoltativo	NO
RF31-F	Una volta rilevato un valore di intensità luminosa sotto soglia, il sistema provvede ad aumentare l'intensità luminosa.	Facoltativo	NO
RF32-F	Una volta rilevato un valore di intensità luminosa sopra soglia, il sistema provvede ad diminuire l'intensità luminosa.	Facoltativo	NO
RF33-F	Il sensore rileva un'intensità luminosa che è sopra un certo valore soglia.	Facoltativo	NO
RF34-F	Il sensore rileva un'intensità luminosa che è sotto un certo valore soglia.	Facoltativo	NO
RF35-F	Il sistema rileva la presenza di un guasto o un'anomalia riguardo una misurazione errata.	Facoltativo	NO
RF36-F	Il sistema provvede ad inserire nella lista di impianti guasti l'area in cui è presente l'anomalia.	Facoltativo	NO
RF37-F	Durante l'inserimento di un nuovo sensore si specifica il tipo di interazione automatico (push) con il sistema.	Facoltativo	SI
RF38-F	Durante l'inserimento di un nuovo sensore si specifica il tipo di interazione su richiesta (pull) con il sistema.	Facoltativo	SI
RF39-O	Durante l'inserimento di un nuovo sensore si specificano i dettagli relativi a tale sensore.	Obbligatorio	SI
RF40-O	Durante l'inserimento di un nuovo sensore si specifica la posizione geografica del sensore di riferimento.	Obbligatorio	SI
RF41-O	Durante l'inserimento di un nuovo sensore si specifica l'ampiezza del raggio d'azione del sensore che si sta aggiungendo.	Obbligatorio	SI
RF42-O	L'impianto guasto viene riparato dal Manutentore e viene segnalato come nuovamente funzionante.	Obbligatorio	SI

Tutti i requisiti funzionali obbligatori sono stati soddisfatti. I requisiti funzionali facoltativi sono stati soddisfatti in parte, in quanto non sono stati implementati i requisiti facoltativi che riguardano la gestione di più aree illuminate contemporaneamente a causa del poco tempo rimanente, anche se oggettivamente sarebbero di facile implementazione.

6.2 Requisiti di Qualità

Requisiti di Qualità			
Codice	Descrizione	Rilevanza	Soddisfatto
RQ1-O	Test che dimostrino il corretto funzionamento dei servizi e delle funzionalità previste, con una copertura dell'70% del codice.	Obbligatorio	SI
RQ2-O	Documenti su scelte implementative e progettuali e relative motivazioni, i problemi aperti e le eventuali soluzioni proposte da approfondire.	Obbligatorio	SI
RQ3-O	Web Application responsive per soddisfare i requisiti obbligatori nei casi d'uso.	Obbligatorio	SI

Tutti i requisiti di qualità, in quanto obbligatori, sono stati soddisfatti.

6.3 Code Coverage

I moduli utilizzati per la code coverage sono stati:

- **Jest:** framework per l'esecuzione di test JavaScript;
- **Supertest:** framework per l'esecuzione di test HTTP;

Inoltre, per verificare la duplicazione di codice è stato utilizzato **SonarCloud**.

Con il committente, inizialmente era stata richiesta una percentuale minima dell'80%, ma in seguito agli incontri di presentazione, riconoscendo che il codice è altamente modulare, è stata abbassata al 70% così da poter escludere costruttori, distruttori e funzioni di supporto raramente utilizzate ma comunque utili.

I test sono riproducibili tramite il comando `npm run coverage` dentro la directory `/client`, mentre con il comando `npm test` dentro la directory `/server` per i test relativi al frontend e al backend. Le percentuali riportate indicano la media, che viene riportata nella prima riga del report.

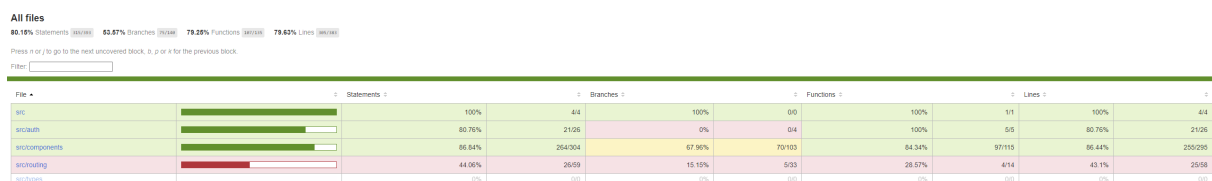


Figure 10: Code coverage del frontend.

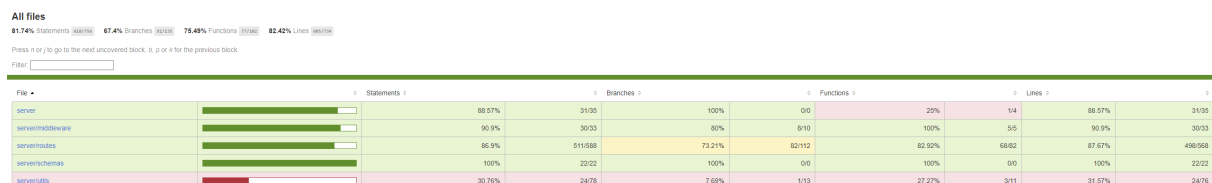


Figure 11: Code coverage del backend.