Kyudd-imi (Sito Web per il monitoraggio delle finestre del CPTT)

1 Introduzione 4

1.1 Informazioni sul progetto 4

1.2 Abstract 4

1.3 Scopo 5

1.4 Analisi del dominio 6

1.5 Analisi e specifica dei requisiti 6

1.6 Use case 9

1.7 Pianificazione 11

1.8 Analisi dei mezzi 12

1.8.1 Software 12

1.8.2 Hardware 12

1.8.3 Node.js 12

1.8.4 Express.js 12

1.8.5 EJS 13

1.8.6 Bcrypt 13

2 Progettazione 14

2.1 Route 14

2.1.1 Route / 14

2.1.2 Route /windowsPage 14

2.1.3 Route /windowPage/:classroom/:id 14

2.1.4 Route /windows/:floor? 14

2.1.5 Route /windowsOpen/:floor? 15

2.1.6 Route /window/:classroom/:id 15

2.1.7 Route /classroomsPage 15

2.1.8 Route /classroomPage/:id 15

2.1.9 Route /classrooms/:floor? 15

2.1.10 Route /classroomsOpen/:floor? 15

2.1.11 Route /classroom/exists/:id 15

2.1.12 Route /classroom/:id 15

2.1.13 Route /login (GET) 16

2.1.14 Route /login (POST) 16

2.1.15 Route /logout (GET) 16

2.1.16 Route /3DPage 16

2.2 Design dell’architettura del sistema 16

2.3 Design dei dati e database 18

2.3.1 Sensore 18

2.3.2 Finestra 18

2.3.3 Stato 19

2.3.4 Utente 19

2.3.5 Tipo 19

2.3.6 Log 19

2.4 Design procedurale 20

2.5 Design delle interfacce 21

2.5.1 Interfaccia home 21

2.5.2 Interfaccia lista finestre 22

2.5.3 Interfaccia finestra 23

2.5.4 Interfaccia lista aule 24

2.5.5 Interfaccia aula 25

2.5.6 Interfaccia login 26

2.5.7 Interfaccia modello 3D 27

3 Implementazione 28

3.1 File di configurazione 28

3.2 Gestione sessioni 28

3.3 Login / Logout 29

3.4 ThreeJS 31

3.5 Routes 32

3.5.1 Visualizzazione pagina 32

3.5.2 Ricevere le risposte JSON 32

3.6 EJS 33

3.6.1 Template 33

3.6.2 Passaggio di dati 33

3.7 Sicurezza 34

3.7.1 SQL Injection 34

3.7.2 Broken access control 35

3.7.3 Salvataggio password 35

3.8 LOG 36

3.9 Database 36

4 Test 38

4.1 Protocollo di test 38

4.2 Risultati test 43

4.3 Mancanze/limitazioni conosciute 44

5 Consuntivo 45

6 Conclusioni 46

6.1 Sviluppi futuri 46

6.2 Considerazioni personali 46

7 Bibliografia 47

7.1 Sitografia 47

7.2 Glossario 48

7.3 Indice delle figure 49

8 Allegati 50

# Introduzione

## Informazioni sul progetto

* Alunno: Gioele Cavallo
* Sezione: Informatica
* Classe: I4AC
* Supervisore: Pascal Poncini
* Perito: Ioulia Zintchenko
* Title: Kyudd-imi (Sito Web per il monitoraggio delle finestre del CPTT)
* Data Inizio: 2023-02-05
* Data Fine: 2023-05-16

## Abstract

**Initial situation**

*In the reality of the Trevano CPT, managing the windows is complicated. It is necessary to remember at the end of class to close the windows, and this task is the responsibility of janitors and cleaning operators as well.*

*In addition, in case of bad weather, it is necessary to be sure that all windows are closed to prevent water from ruining the classrooms.*

**Implementation**

*For this project, it was required to create a local website to display the open and closed windows of the building. In addition, classrooms with open windows and statistics on the opening and closing of individual windows and status history are also to be displayed.*

*Since the website is to be used by janitors to close open windows, it needs to be responsive so that it can be used from both mobile devices and computers.*

*Also wanted is separate users between admin and standard users who will be able to perform different operations.*

*The database will be hosted on a raspberry, as the website, and will be populated with minimal information such as: transaction number, time change status, information about the ESP32 device that collected the information, name and window/classroom number, the status of the window.*

**Results**

*All requirements were successfully implemented by creating the website and hosting it on the raspberry in the internal network.*

*The utilities were implemented but do not perform anything different from each other, this is because in discussion it was noted that it was not sure what to make the different users do, so it will be part of future development.*

## Scopo

Il progetto riguarda la creazione di un sito web utilizzabile per gestire le finestre della scuola CPT di Trevano. Bisogna poter visualizzare le finestre chiuse e le aule che contengono finestre aperte. Delle finestre si vogliono avere anche delle statistiche sul tempo che sono state aperte/chiuse.   
Inoltre ci devono essere delle utenze che permettano in un futuro di eseguire differenti azioni.

Analisi

## Analisi del dominio

Questo progetto inizia da zero. Mi è stato chiesto di creare un sito web, ospitato su un raspberry sulla rete nera, il quale permette di visualizzare l’elenco delle finestre aperte e chiuse. Oltre alle finestre si può visualizzare lo stato di un’aula, quindi se ci sono finestre aperte o meno, e di ogni aula/finestra si può avere lo storico degli stati passati.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Sito web locale |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Bisogna avere un sito web locale nella rete del CPT di Trevano. |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Visualizzazione finestre |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Il sito web deve mostrare le finestre aperte e chiuse. |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Visualizzazione classi |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Il sito web deve mostrare le classi e se hanno delle finestre aperte o chiuse. |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Statistiche finestre |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Bisogna far visualizzare le statistiche sul tempo di apertura / chiusura delle finestre. |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Informazioni minime |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | La visualizzazione delle informazioni deve essere al minimo di una lista delle classi con finestre aperte / chiuse (o 3D). |

|  |  |
| --- | --- |
| **ID: REQ-06** | |
| **Nome** | Accessi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Deve essere possibile eseguire il login per accedere alla visualizzazione dei dati. |

|  |  |
| --- | --- |
| **ID: REQ-07** | |
| **Nome** | Utenze |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Ci devono essere 2 tipi di utenti: standard e admin. Il primo può solo visualizzare i dati, il secondo invece può anche scaricare i dati. |

|  |  |
| --- | --- |
| **ID: REQ-08** | |
| **Nome** | Responsive |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il sito deve essere responsive. Quindi visualizzabile da pc, tablet e telefoni. |

|  |  |
| --- | --- |
| **ID: REQ-09** | |
| **Nome** | Database con info minime |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il database deve contenere al minimo:   * Numero della transazione * Ora cambiamento di stato * Nome del dispositivo (numero ESP32) * Numero finestra online * Stato della finestra (Aperta / Chiusa) |

|  |  |
| --- | --- |
| **ID: REQ-10** | |
| **Nome** | Host su Raspberry |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il sito web e il database devono essere ospitati su un Raspberry. |

## Use case

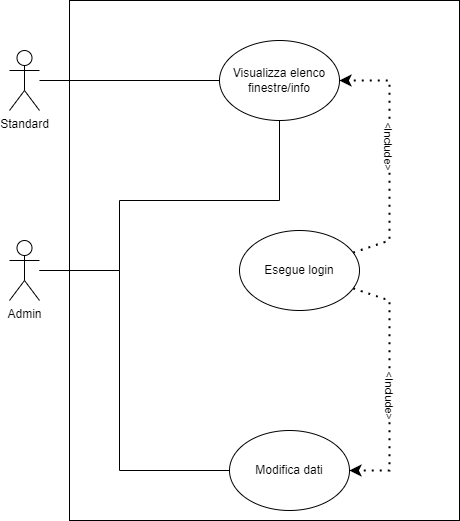


Figura 1 Use case

Per questo progetto sono stati identificati 2 attori:

* Standard: questo attore è un’utente con permessi di lettura dei dati solo nel caso in cui eseguisse il login. Nello specifico può visualizzare le pagine web che mostrano l’elenco delle finestre e le rappresentazioni di esse.
* Admin: questo attore è un’utente con permessi di lettura e scrittura dei dati nel caso in cui eseguisse il login. Questo utente oltre a vedere le liste delle finestre e le rispettive info, potrà in un futuro anche modificare i dati (nel caso ci fossero dei problemi).

## Pianificazione

Di seguito è rappresentata la pianificazione preventiva del progetto. Si può notare come la maggior parte del tempo sia dedicato all’implementazione. La metodologia utilizzata per la pianificazione è quella Waterfall, a motivo della facilità nell’implementarla.

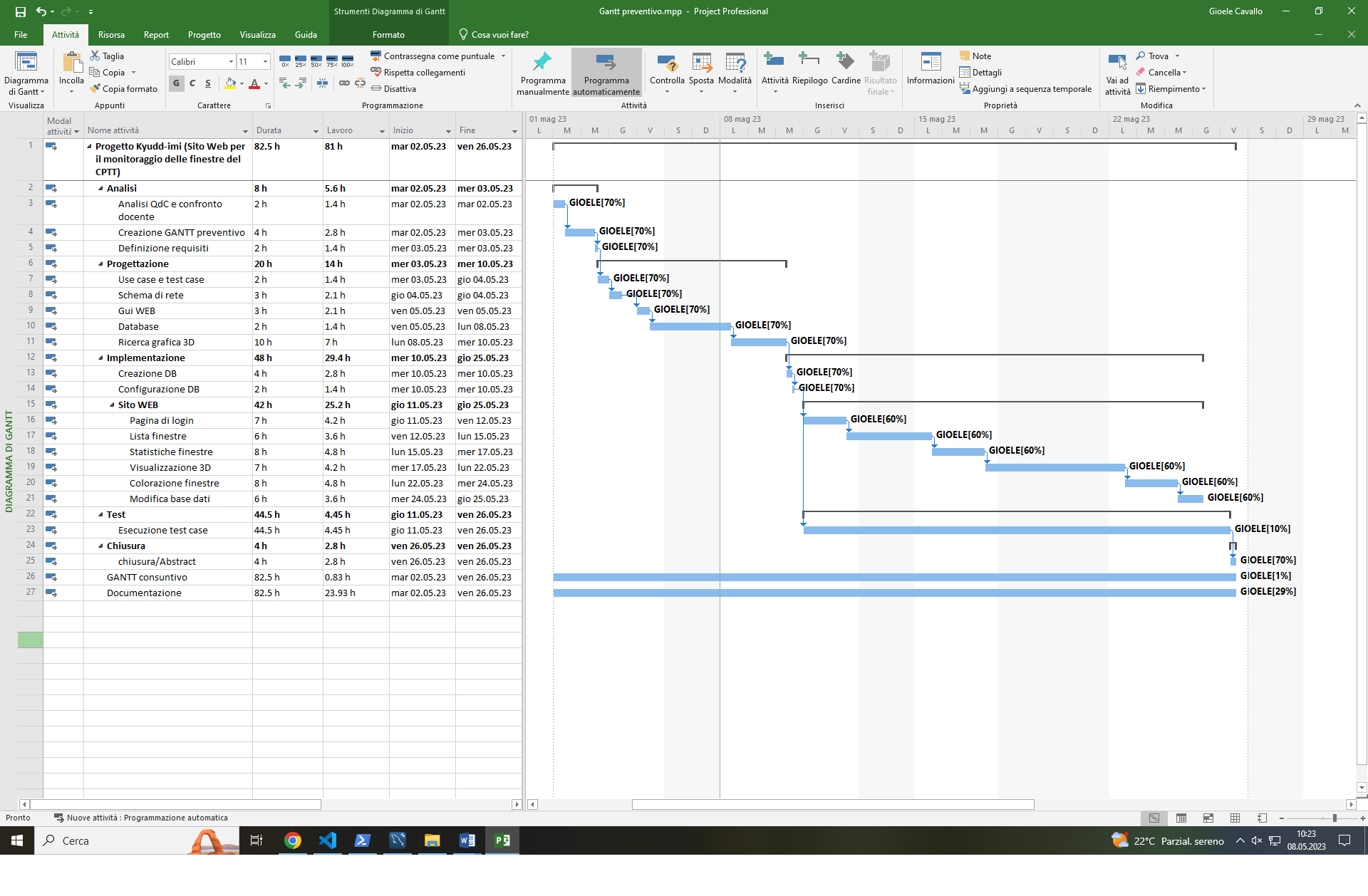


Figura 2 GANTT preventivo

## Analisi dei mezzi

### Software

Node.js 18.16.0, runtime system

Express 4.18.2, framework backend per Node.js

Ejs 3.1.9, linguaggio di modellihtml/javascript.

MySQL 8.0, sistema di gestione di database relazionale

Threejs 0.150.1, libreria JS per il render e animazione 3D.

Npm 9.5.1, gestore di pacchetti opensource.

Datatables 1.13.4, libreria per la visualizzazione dei dati.

bcrypt 5.1.0, funzione di hashing.

### Hardware

Per lo sviluppo è stato usato un computer fornito dalla scuola con:

* CPU: Intel(R) Xeon(R) CPU E3-1240 v5 3.50GHz
* RAM: 16 GB DDR4
* GPU: NVIDIA Quadro K620
* HDD: 250 GB

Per la fase di distribuzione è stato utilizzato un Raspberry con:

* CPU:
* RAM
* GPU
* HDD

### Node.js

Node è un runtime system scritto in C e C++. Questo runtime system utilizza un’architettura a *Single Thread Event Loop* che si differenzia dell’architettura *Multi Thread* in quanto non viene creata una thread per ogni richiesta. In poche parole c’è solo una *thread* che gestisce le richieste, ogni volta che arriva una richiesta essa viene messa in una *pool* e quando la *thread* è libera esegue l’operazione.   
Se la richiesta può essere eseguita nell’immediato e non è bloccante la *thread* principale se ne occupa restituendo una risposta, altrimenti si affida a dei worker (che sono delle thread limitate) che avvertiranno la thread principale alla fine dell’esecuzione del lavoro per restituire il risultato.

Questo permette a Node di risparmiare molte risorse e memoria che sfocia quindi in un’esecuzione più rapida delle *tasks*.

### Express.js

Express è un *framework backend* per Node.js che offre sistemi di *routing* separabili in più file, in questo modo si hanno le route aggregate in piccoli file più facili da gestire. Inoltre Express permette di implementare i *middleware* (sono dei codici eseguiti prima che la richiesta http arrivi al client).

Express diventa modulare grazie al suo CLI (strumento di interfaccia a riga di comando) NPM per aggiungere all’applicazione più funzionalità.

### EJS

EJS, Embedded JavaScript templates, è un *template engine* che permette di utilizzare dei pattern per rendere l’applicazione che si va a sviluppare più consistente nelle interfacce. Inoltre è utile per passare dei dati alle pagine web. In questo modo senza fare delle richieste ad un’API si possono ricevere dei valori utili da visualizzare nella pagina web e per la logica in essa.

### Bcrypt

Bcryt è una funzione di *hashing* basata sulla cifratura blowfish. Questa funzione permette di rendere nel tempo adattiva la sicurezza. Aumentando il conteggio di iterazioni eseguite, l’esecuzione delle operazioni diventa più lento e quindi diventa più resistente agli attacchi di forza bruta.

Inoltre Bcrypt permette il *salting* delle password, che consiste nell’inserimento di bit casuali in una password complicando gli attacchi “a dizionario” (rainbow table).

# Progettazione

## Route

In questo capitolo vengono descritte le varie routes ed il loro utilizzo.

|  |  |  |
| --- | --- | --- |
| Metodo | Url | Ritorno |
| GET | / | Pagina web |
| GET | /windowsPage | Pagina web |
| GET | /windowPage/:classroom/:id | Pagina web |
| GET | /windows/:floor? | application/json |
| GET | /windowsOpen/:floor? | application/json |
| GET | /window/:classroom/:id | application/json |
| GET | /classroomsPage | Pagina web |
| GET | /classroomPage/:id | Pagina web |
| GET | /classrooms/:floor? | application/json |
| GET | /classroomsOpen/:floor? | application/json |
| GET | /classroom/exists/:id | application/json |
| GET | /classroom/:id | application/json |
| GET | /login | Pagina web |
| POST | /login | Pagina web |
| GET | /logout | Pagina web |
| GET | /3DPage | Pagina web |

### Route /

Questa route ritorna la pagina web home.

### Route /windowsPage

Questa route ritorna la pagina web dell’elenco delle finestre.

### Route /windowPage/:classroom/:id

Questa route ritorna la pagina web delle informazioni di una singola finestra (:id) appartenente ad una classe (:classroom).

### Route /windows/:floor?

Questa route ritorna un file JSON con una la lista di oggetti che sarebbero le finestre ed i propri attributi con anche l’ultimo stato della finestra.

Il parametro *floor* facoltativo, se settato, diventa un filtro per il numero del piano delle finestre da ritornare.

### Route /windowsOpen/:floor?

Questa route ritorna un file JSON con una la lista di finestre aperte con i propri attributi e l’ultimo stato della stessa.

Il parametro *floor* facoltativo, se settato, diventa un filtro per il numero del piano delle finestre aperte da ritornare.

### Route /window/:classroom/:id

Questa route ritorna un file JSON con una lista di oggetti che sarebbero tutti gli stati della finestra (:id) appartenente alla classe(:classroom).

### Route /classroomsPage

Questa route ritorna la pagina web per la lista delle classi.

### Route /classroomPage/:id

Questa route ritorna la pagina web per classe *:id*.

### Route /classrooms/:floor?

Questa route ritorna un file JSON con la lista di classi con il relativo ultimo stato.

Il parametro *floor* facoltativo, se settato, diventa un filtro per il numero del piano delle classi da ritornare.

### Route /classroomsOpen/:floor?

Questa route ritorna un file JSON con la lista di classi con il relativo ultimo stato e solo delle classi aperte.

Il parametro *floor* facoltativo, se settato, diventa un filtro per il numero del piano delle classi aperte da ritornare.

### Route /classroom/exists/:id

Questa route ritorna un file JSON con gli attributi della classe *:id*.

### Route /classroom/:id

Questa route ritorna un file JSON con la lista degli stati della classe passata (:id).

### Route /login (GET)

Questa route ritorna la pagina web con il form per eseguire il login.

### Route /login (POST)

Questa route esegue i controlli dei campi inseriti nel form del login e ritorna la pagina web home, con relativo messaggio se il login è stato eseguito correttamente o meno.

### Route /logout (GET)

Questa route esegue il logout, se è stato effettuato prima il login, e poi reindirizza alla pagina home.

### Route /3DPage

Questa route ritorna la pagina web della visualizzazione del modello 3D.

## Design dell’architettura del sistema

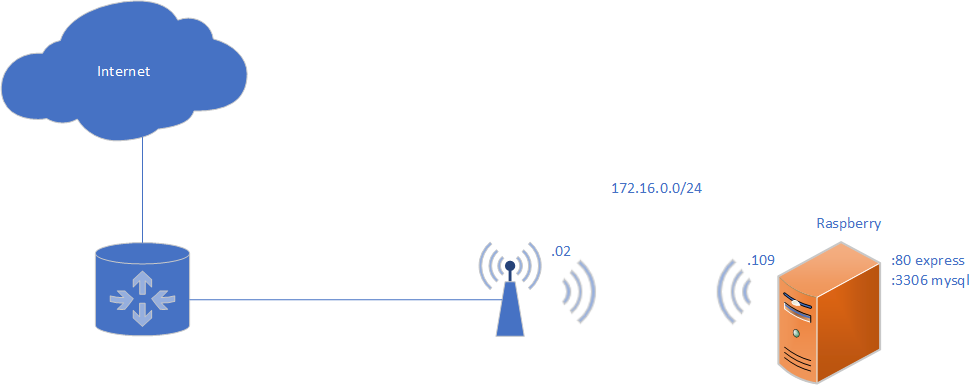


Figura 3 Schema di rete

Per questo progetto viene utilizzato un raspberry collegato alla rete nera (172.16.0.0/24), il quale fornisce l’host per il sito web e anche per il database mysql. Questa rete era già presente in quanto la rete nera è stata implementata nella nostra struttura dapprima che iniziassi il progetto.

Di seguito è rappresentata la struttura delle cartelle utilizzata per il progetto:

/

├── node\_modules/ -> dipendenze del progetto installate  
│

├── public/ -> cartella pubblica raggiungibile dall'esterno   
│ │

│ ├── icons/ -> cartella per le icone utilizzate nel sito

│ ├── models/ -> cartella per il modello 3D

│ ├── scripts/ -> cartella per gli scripts utilizzati nelle pagine web

│ └── template/ -> cartella per i templates utilizzati nelle pagine web  
│

├── routes/ -> cartella contenente i file con tutte le route del programma  
│

├── .env -> file per le variabili da tenere segrete

├── database.js -> file di utility utilizzato per connettersi al database ed   
│ inserire dati all'interno

├── npm-shrinwrap.json -> file con tutti i pacchetti installati dell'applicazione

├── package.json -> informazioni sull'applicazione

└── server.js -> file principale dove inizia l'applicazione

## Design dei dati e database

Qui di seguito è riportato il diagramma del database di questo progetto.

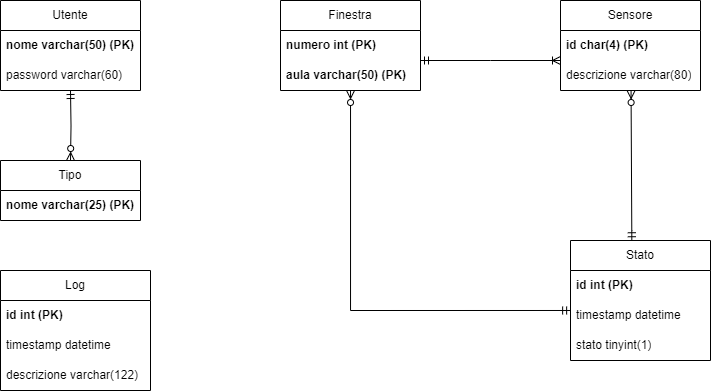


Figura 4 Diagramma database

### Sensore

Questa tabella possiede 2 attributi:

* Id: è la chiave primaria della tabella.
* Descrizione: contiene una descrizione del sensore.

### Finestra

Questa tabella possiede 4 attributi:

* numero: è il numero della finestra all’interno dell’aula. 0 significa che il dato rappresenta tutta l’aula.
* aula: è il nome dell’aula a cui appartiene la finestra.
* sensore\_id: è l’id del sensore che raccoglie i dati della finestra.

Aula e numero formano una chiave composta.

### Stato

Questa tabella possiede 4 attributi:

* id: è la chiave della tabella utile a identificare lo stato. Questa chiave viene generata in modo auto incrementale.
* timestamp: è il tempo in cui è cambiato lo stato.
* stato: è lo stato della finestra (1 se aperta, 0 se chiusa).
* sensore\_id: è l’id del sensore che ha mandato il dato dello stato.
* finestra\_numero: il numero della finestra dalla quale è stato preso lo stato.
* finestra\_aula: l’aula della finestra dalla quale è stato preso lo stato.

### Utente

Questa tabella possiede 3 attributi:

* nome: è la chiave della tabella (il nome dell’utente) utile a identificare l’utente.
* password: è la password dell’utente criptata.
* tipo\_nome: è il nome del tipo di utente.

### Tipo

Questa tabella possiede unicamente 1 attributo utile a differenziare i tipi di utenti:

* nome: questo attributo, e chiave della tabella, server a differenziare gli utenti in base ai privilegi.

### Log

Questa tabella verrà utilizzata per tenere traccia delle azioni degli utenti nel sito web. Possiede 3 attributi:

* id: è un numero auto incrementale usato come chiave primaria nella tabella.
* timestamp: è il tempo in cui è stato registrato l’evento.
* descrizione: è la descrizione dell’evento.

## Design procedurale

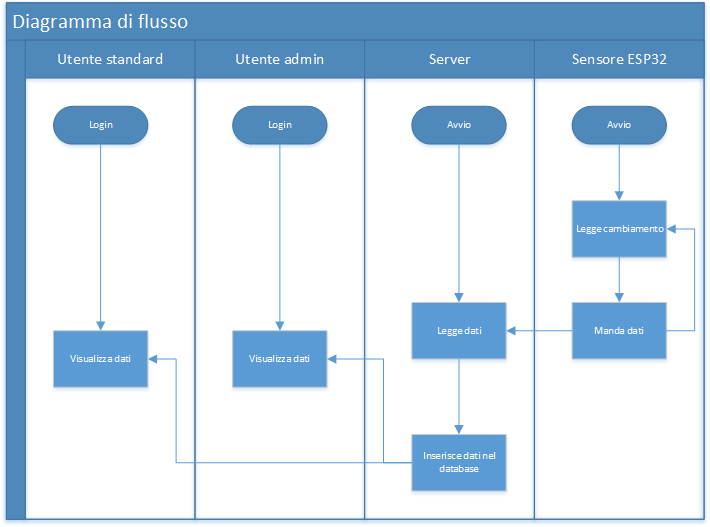


Figura 5 Diagramma di flusso

Questo è il diagramma di flusso *Swimlane* ideato per il progetto con 4 attori. Il diagramma è relativo al completo progetto, quindi anche con i sensori di cui non mi occupo direttamente io. Questo diagramma server per dare un’idea più chiara del giro che fanno i dati e da dove arrivano.

Ci sono 4 attori:

* Utente standard: questo utente può visualizzare i dati e di conseguenza le pagine web relative, solo dopo il suo login.
* Utente admin: questo utente può visualizzare i dati e di conseguenza le pagine web relative, solo dopo il suo login.
* Server: questo attore server per leggere i dati provenienti dai sensori e salvarli in un database.
* Sensore ESP32: questo attore è colui che legge i dati dalle finestre che poi manda al server, il quale gestisce queste informazioni. L’operazione di leggere degli eventuali cambiamenti, e nel caso mandare i dati al server, viene sempre eseguita ripetutamente.

## Design delle interfacce

### Interfaccia home

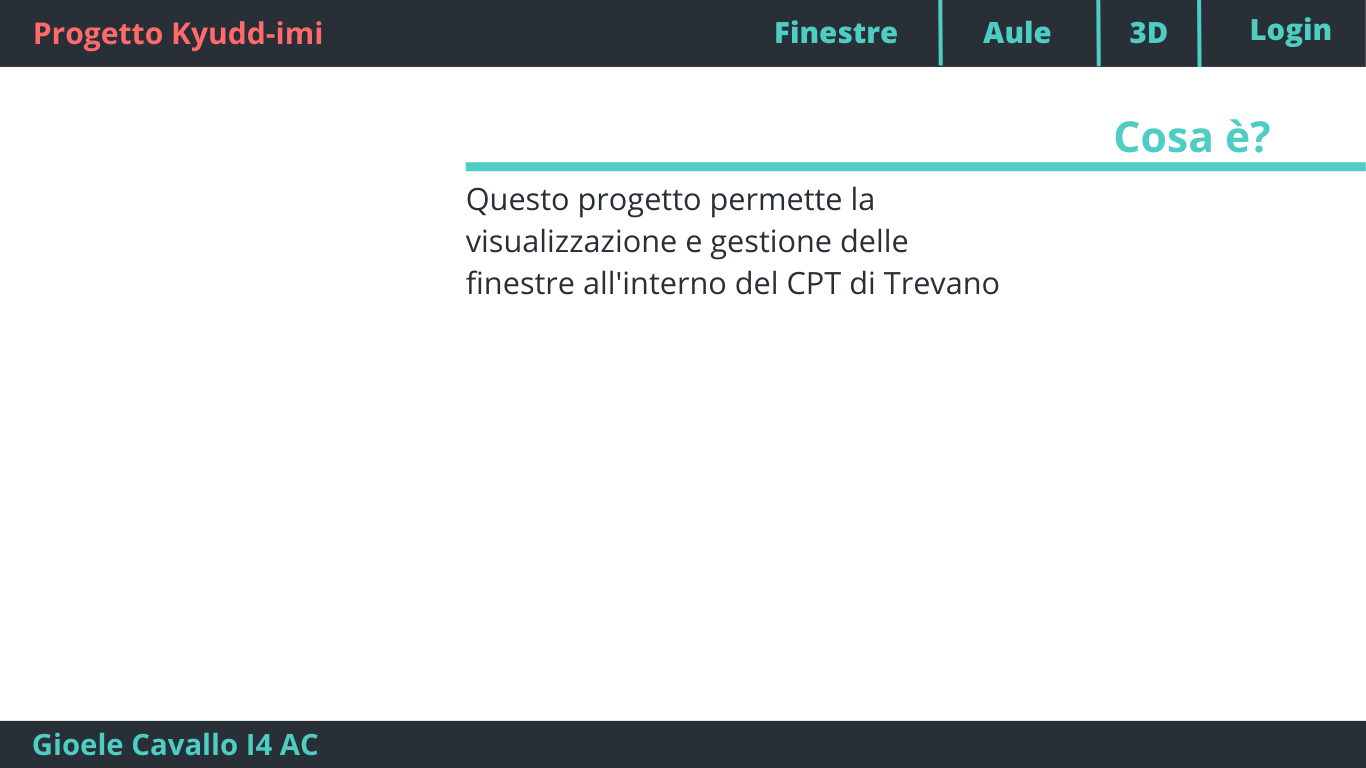


Figura 6 Interfaccia home

Questa interfaccia è la prima che viene visualizzata. Possiede una breve descrizione del progetto.

Inoltre ha un header, uguale per tutte le pagine, con 4 link che portano a:

* La pagina dove si visualizza l’elenco delle finestre.
* La pagina dove si visualizza l’elenco delle aule.
* La pagina per visualizzare il modello 3D della scuola.
* La pagina per il login / logout (a dipendenza se ci si è già autenticati).

### Interfaccia lista finestre

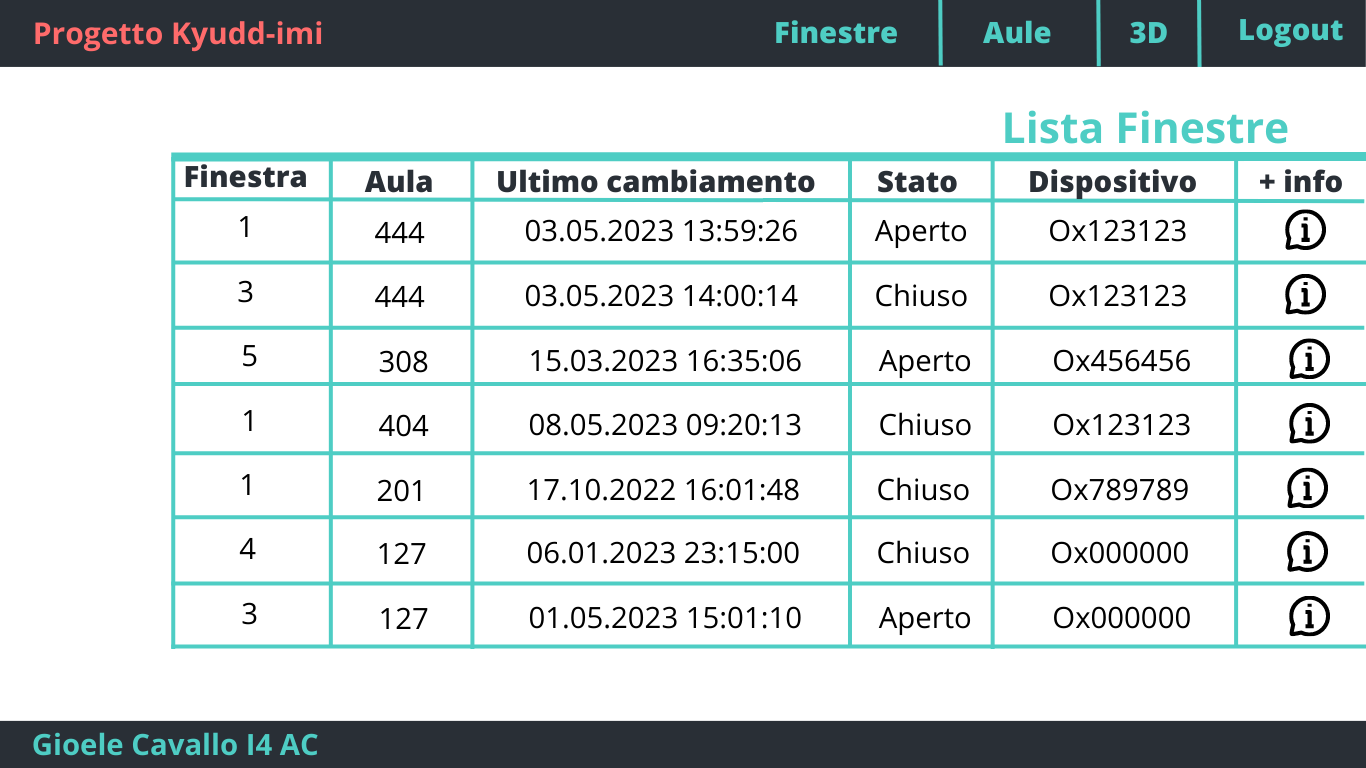


Figura 7 Interfaccia lista finestre

Questa interfaccia ha lo scopo di visualizzare la lista delle finestre con il loro nome, l’aula alla quale appartengono, l’ultimo cambiamento di stato, lo stato attuale e il dispositivo che ha registrato quel valore.

In fine la tabella ha una colonna con un link che porta alla pagina specifica della finestra dove si possono visualizzare tutti gli stati passati.

### Interfaccia finestra

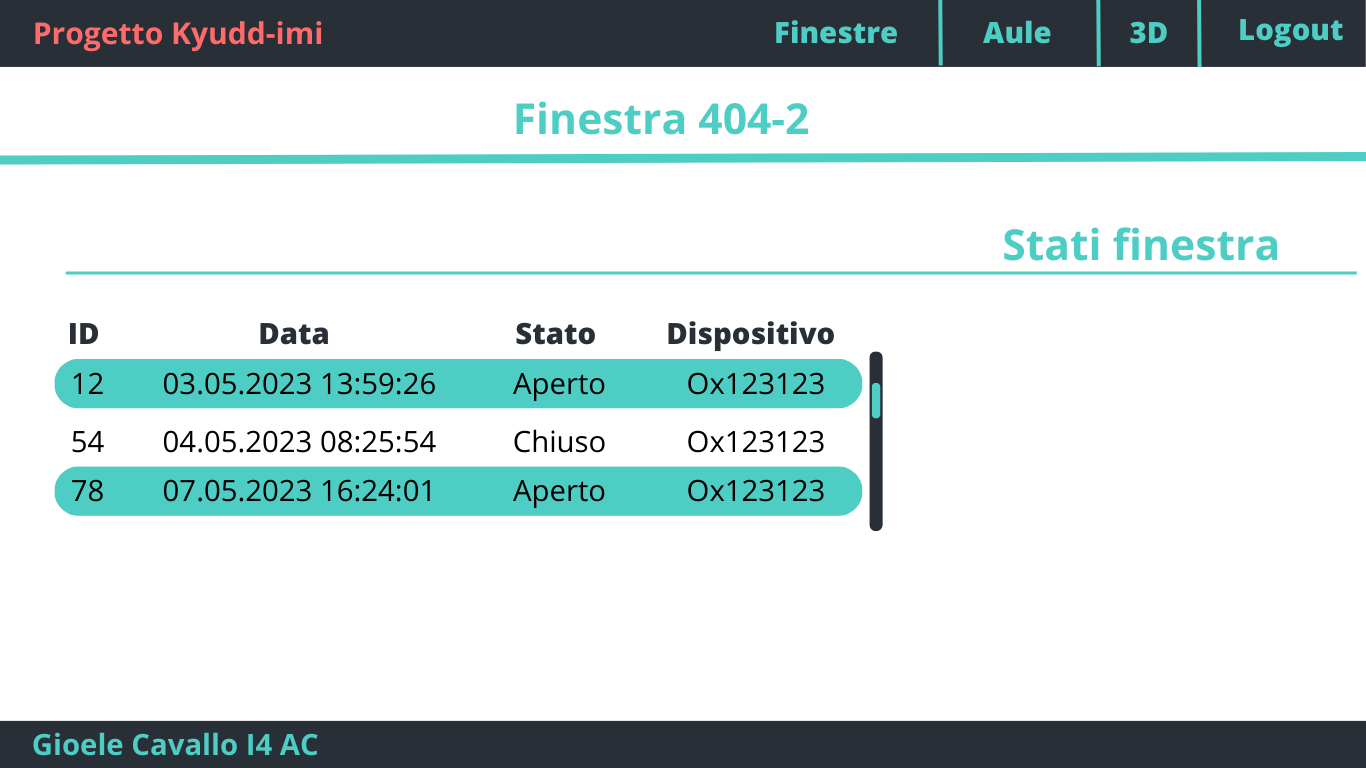


Figura 8 Interfaccia finestra

Questa interfaccia permette di visualizzare tutti gli stati passati di una finestra. Degli stati viene visualizzato l’id, la data, lo stato e il dispositivo che ha registrato il dato.

### Interfaccia lista aule

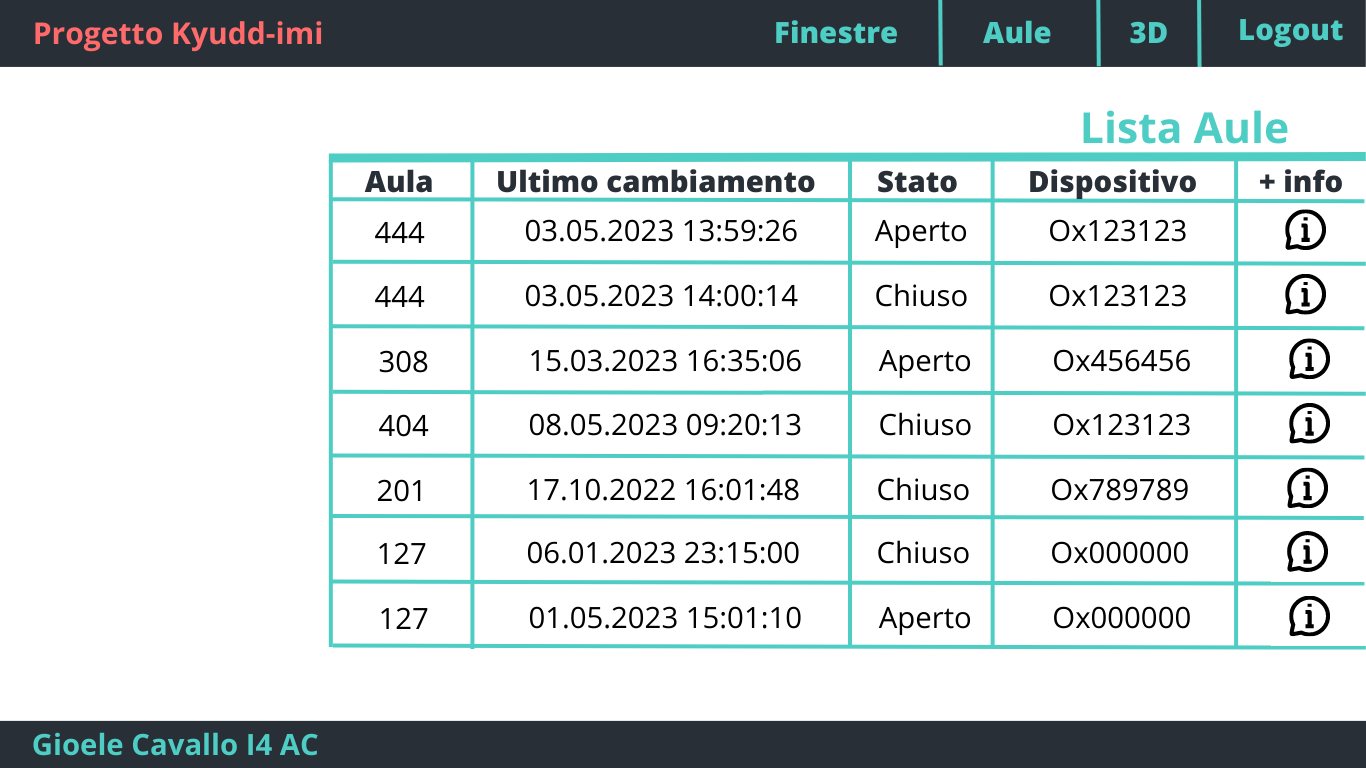


Figura 9 Interfaccia lista aule

Questa interfaccia fa visualizzare l’elenco delle aule con il rispettivo nome, data dell’ultimo cambiamento, lo stato e il dispositivo che ha registrato l’ultimo dato.

Inoltre l’ultima colonna possiede un link che permette di visualizzare più informazioni sull’aula.

### Interfaccia aula

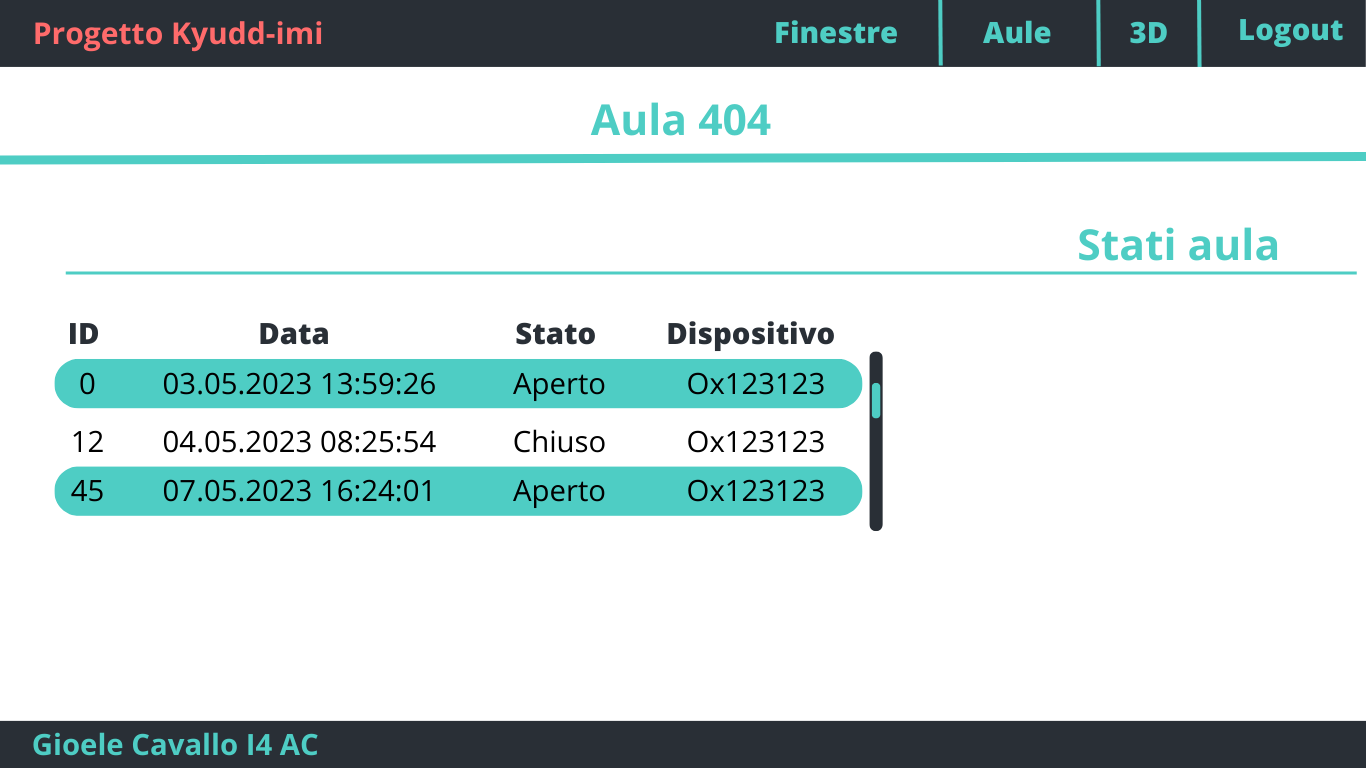


Figura 10 Interfaccia aula

Questa interfaccia permette di visualizzare tutti gli stati passati di un’aula. Degli stati viene visualizzato l’id, la data, lo stato e il dispositivo che ha registrato il dato.

### Interfaccia login

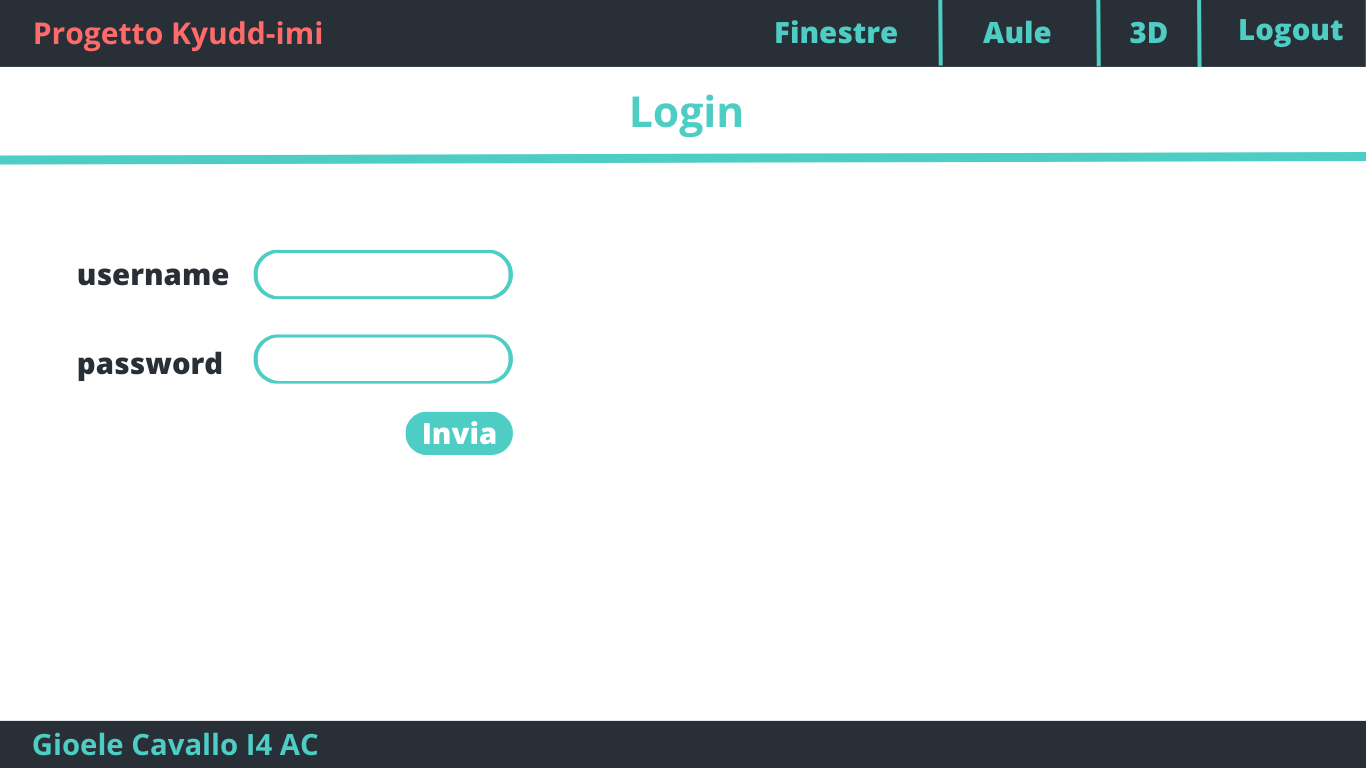


Figura 11 Interfaccia login

Questa interfaccia permette di eseguire il login così da poter accedere a tutte le funzionalità.

### Interfaccia modello 3D

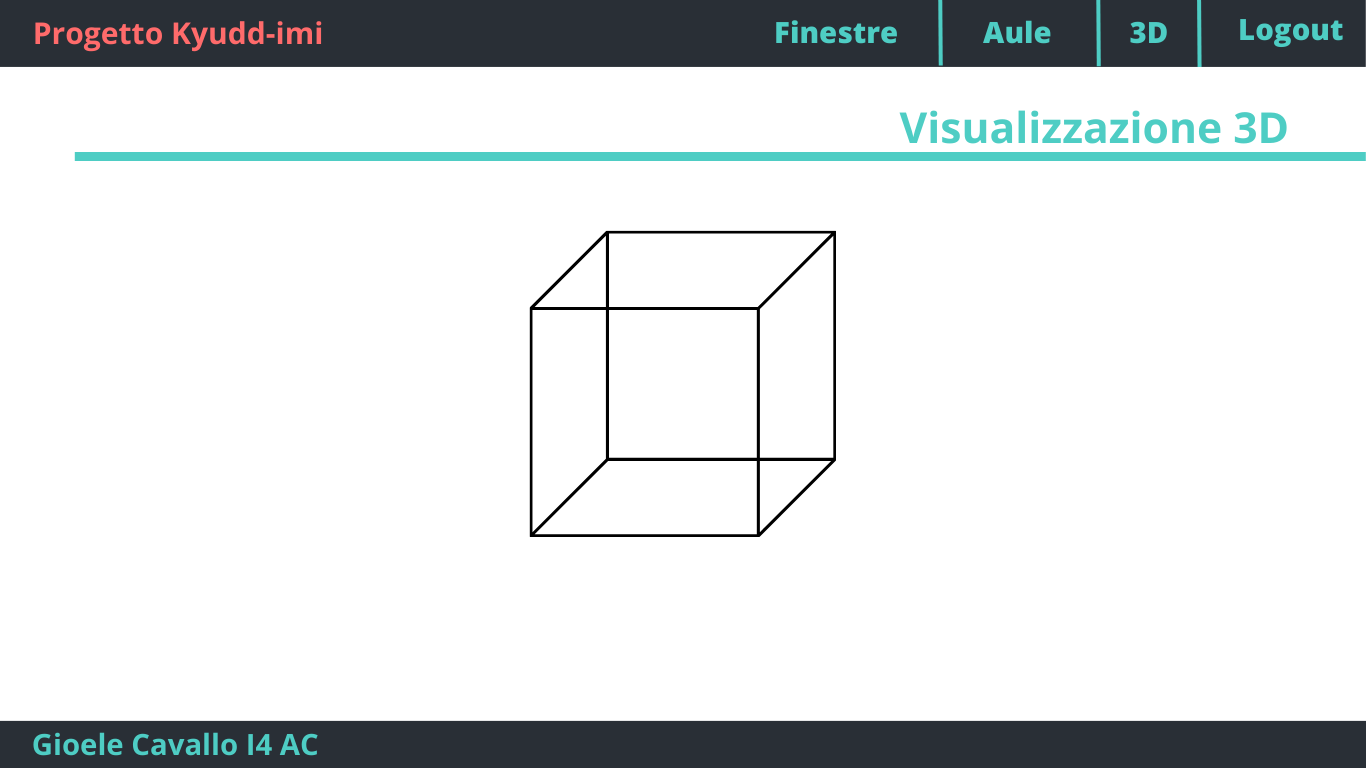


Figura 12 Interfaccia modello 3D

Questa interfaccia permette di visualizzare il modello 3D dell’istituto con le relative aule.

# Implementazione

## File di configurazione

Esiste un file di configurazione .env nel quale vengono salvate le variabili utilizzate nel codice e che devono rimanere segrete. In questo file salvo le variabili:

PUBLIC\_DIR 🡪 percorso della cartella pubblica del sito web.

DB\_USER 🡪 username con il quale ci si connette al database

DB\_HOST 🡪 host del database al quale ci si connette per il database.

DB\_PASSWORD 🡪 password con la quale ci si connette al database.

DB\_DATABASE 🡪 nome del database al quale ci si connette.

Per caricare le varabili presenti nel file di configurazione utilizzo la libreria *dotenv* che automaticamente prende le variabili presenti nel file .env e le carica all’interno del codice.

Questo è come viene utilizzata la libreria:

require('dotenv').config();

const PUBLIC = process.env.PUBLIC\_DIR;

Nella prima parte di codice richiamo il file di *enviroment* e carico nella memoria le variabili d’ambiente, nella seconda linea vado a leggere la variabile *PUBLIC\_DIR* e la memorizzo in una variabile.

## Gestione sessioni

Questo è il codice utilizzato per le sessioni e la loro inizializzazione:

const session = require('express-session');

app.use(session({

    secret: "<the secret id>",

    saveUninitialized: false,

    resave: false

}));

Per prima cosa viene creato un oggetto di sessione con il *require* dopodiché viene utilizzato come middleware.

Le opzioni utilizzate nell’oggetto di sessione sono :

* secret: questa stringa serve a firmare l’ID della sessione in modo da crittografare i dati nella comunicazione bidirezionale tra client e server.
* saveUninitialized: Questa opzione permette di decidere se salvare la sessione anche se non è stata modificata durante la richiesta. La ho impostata a *`false`* in modo da non salvare le sessioni in quanto non mi è utile averle. In ogni caso il parametro coockie.MaxAge di default è impostato a null, quindi una volta chiuso il browser i coockie e la sessione vengono cancellati.
* resave: Questo parametro permette di salvare una sessione solo nel caso in cui è stata modificata. Impostando il parametro su *`false`* la sessione viene salvata solo se è stata modificata durante la richiesta.

Le sessioni vengono utilizzate per immagazzinare lo *username* dell’utente che ha eseguito il login. In base a questo valore, se è nullo o meno, si può sapere se è stato effettuato il login.

La sessione quindi viene utilizzata per rendere determinate route disponibili solo in caso di login.

## Login / Logout

Questa è la route che gestisce il login.

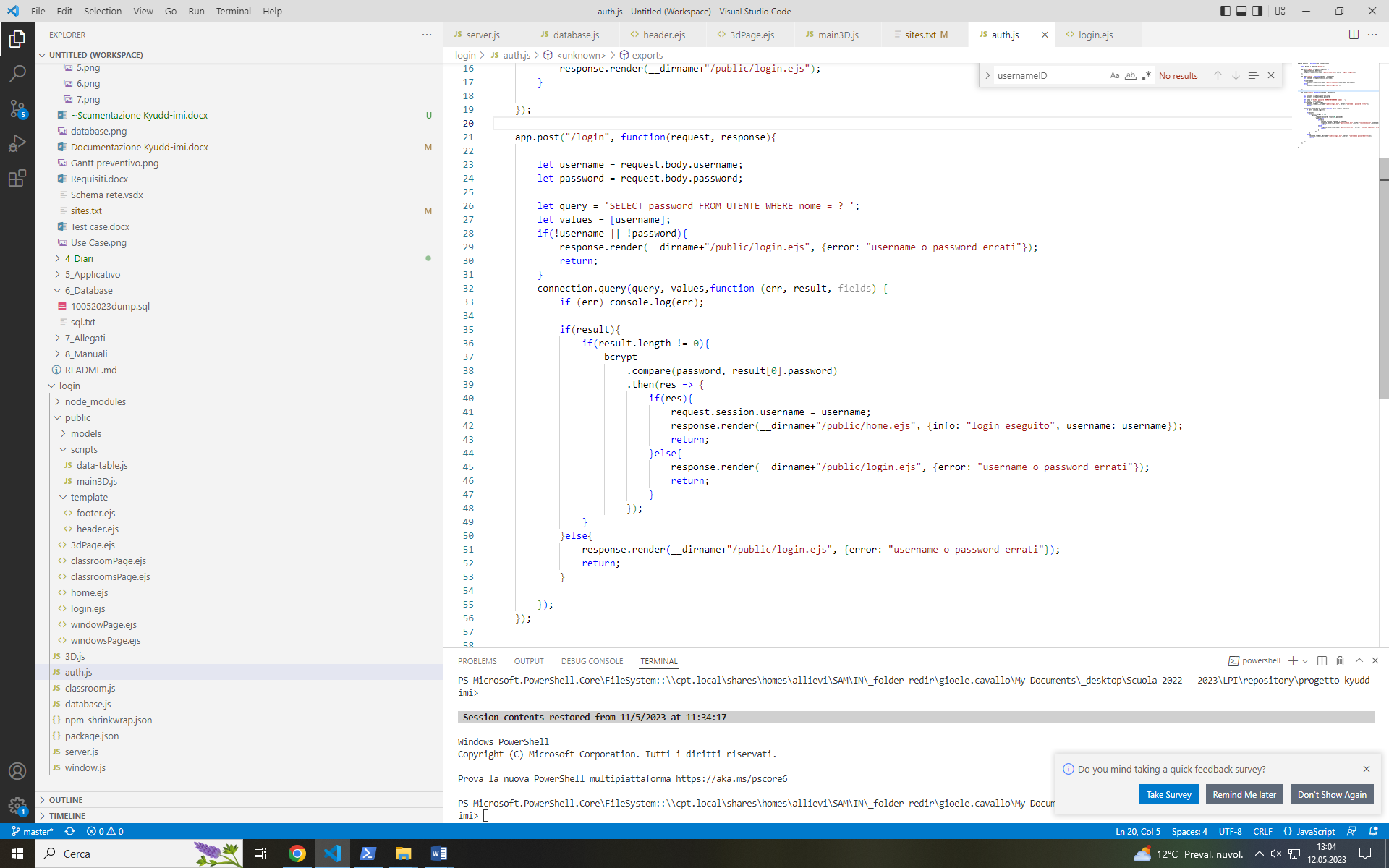


Figura 13 codice post login

Questo è il codice che risponde alla richiesta post per il login.

Per prima cosa vengono presi i dati dal form e viene controllato che non siano nulli.

Dopodiché viene eseguita una query al database dove viene ritornata la password dell’utente ricercato. Dopodiché utilizzando la libreria bycrypt viene controllata che la password passata sia uguale al’hash nel database. Se dovessero essere uguali nella sessione viene salvato il nome dell’utente, altrimenti viene ritornata la pagina di login con un errore.

Per il logout il codice è il seguente:

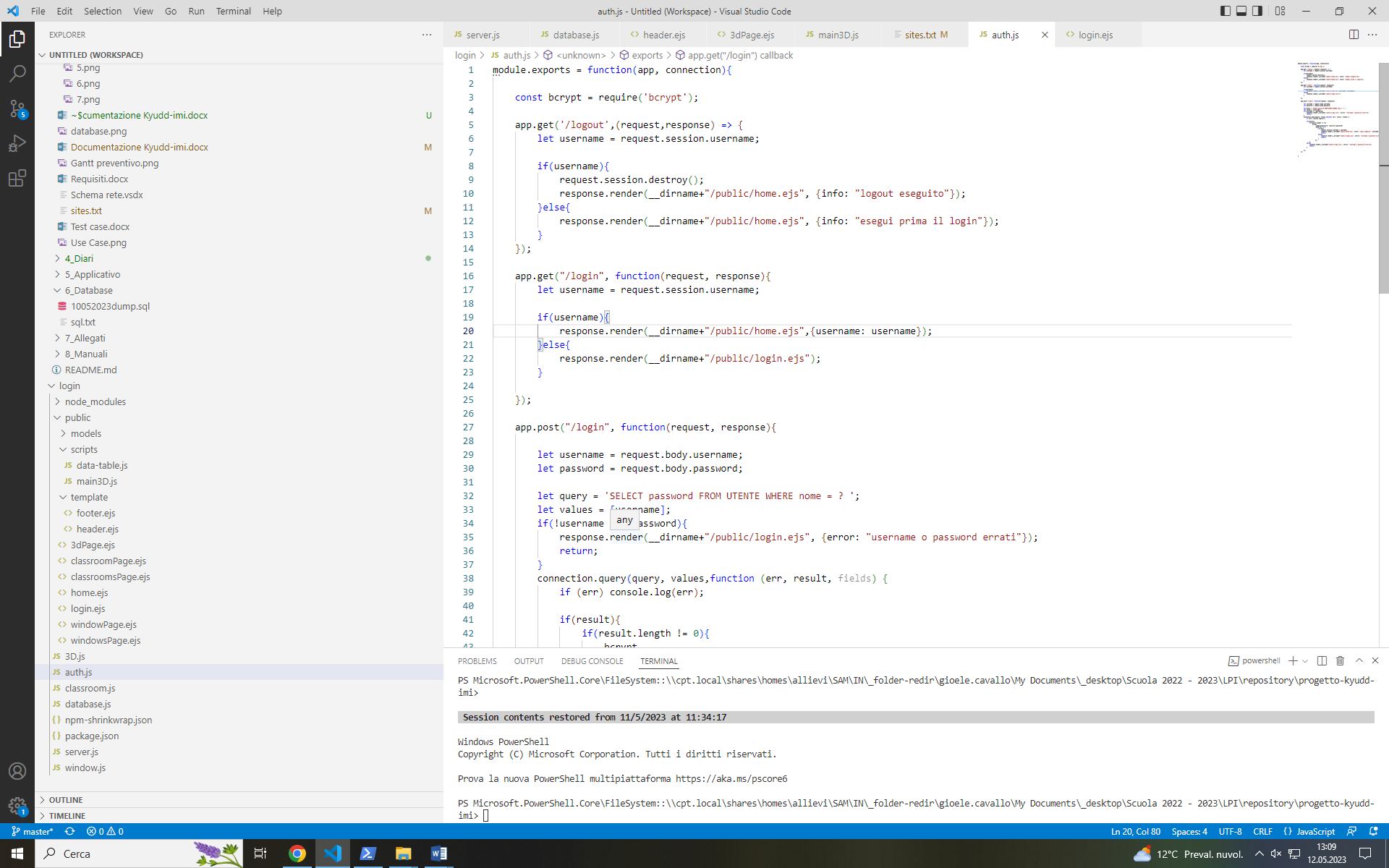


Figura 14 codice logout

In questa route si controlla che l’utente abbia prima eseguito il login e poi viene reindirizzato alla pagina in base al caso.

## ThreeJS

Qui di seguito è riportato la parte di codice principale usato per gestire l’oggetto 3D:

const scene = new THREE.Scene();

const loader = new GLTFLoader();

loader.load(

  ObjectPath,

  function (gltf) {

    object = gltf.scene;

    scene.add(object);

  }

);

function colorModel(){

   $.get("http://localhost:4000/classrooms", function(data, status){

      let json = data;

      for (const key in json){

          if(json.hasOwnProperty(key)){

            let window = json[key];

            let value = window.stato == 1 ? 0xff0000: 0x00ff00;

            let material = new THREE.MeshPhongMaterial(

{ color: value, vertexColors: THREE.VertexColors }

);

            object.getObjectByName(window.aula).material = material;

          }

      }

   });

}

Per prima cosa viene creata la scena e l’oggetto del modello 3D. Dopodiché grazie alla funzione *.load* dell’oggetto si può caricare tramite una path il modello ed aggiungerlo alla scena.

La funzione **colorModel** fa una richiesta GET delle classi per poi iterare le classi e cerca il loro id all’interno del modello 3D per colorare la porzione rappresentante la classe.

Three.js permette anche di muoversi all’interno della pagina web girando e ingrandendo la visuale sull’oggetto tramite l’ausilio di un *listener*. Per fare questo è necessario aggiungere una camera dandole una posizione ed un’angolazione verso l’oggetto.

Il formato usato per il modello 3D è il GL Transmission Format (gLTF) che viene utilizzato molto a motivo della sua riduzione delle dimensioni del file e quindi la potenza necessaria per il rendering dell’oggetto.

## Routes

In tutte le route vengono gestite le sessioni per controllare il login dell’utente, il codice essendo ripetitivo verrà tralasciato dove possibile. Da ricordarsi però che il controllo viene sempre eseguito.

### Visualizzazione pagina

app.get("/classroomsPage", function(request, response){

    let username = request.session.username;

    if(username){

        response.render(\_\_dirname+"/public/classroomsPage.ejs", {username: request.session.username});

    }else{

        response.render(\_\_dirname+"/public/home.ejs", {error: "login necessario"});

    }

});

### Ricevere le risposte JSON

Per ricevere le risposte in formato JSON utilizzo la funzione di express che esegue la conversione delle stringhe a JSON con *app.use(express.json())* .

Questo è un codice di esempio di una query con il risultato restituito sottoforma di JSON:

let query = 'SELECT F.aula, F.sensore\_id, S.stato, S.timestamp FROM FINESTRA F JOIN STATO S ON F.numero = S.finestra\_numero AND F.aula = S.finestra\_aula WHERE S.timestamp = ( SELECT MAX(timestamp) FROM STATO WHERE finestra\_numero = F.numero AND finestra\_aula = F.aula ) AND F.numero = ?; ';

let values = [request.params.numero];

connection.query(query, values, function (err, result, fields) {

response.status(200).send(result);

return;

});

In questo codice viene dichiarata la query e i valori da inserire all’interno vengono segnati con il carattere `?`. I valori poi vengono passati sottoforma di array alla funzione *connection.query* assieme alla query dichiarata. Grazie alla funzione di *callback* si possono ritornare i risultati.

Questo pattern nel codice viene seguito ogni qualvolta bisogna ritornare dei valori sottoforma di json e bisogna interfacciarsi al database.

## EJS

EJS può essere utilizzato sia per la funzionalità dei suoi template ed anche per la possibilità di passare all’interno del codice html delle variabili.

Nei capitoli seguenti spiegherò come ho utilizzato tutte e due le sue funzioni.

### Template

EJS viene utilizzato per avere un sito web consistente e quindi con un template uguale nelle pagine in cui il cliente può ritrovarsi ad utilizzare facilmente.

Per questo motivo ho creato due sezioni, *header* e *footer*, le quali chiamo all’interno delle pagine web attraverso il codice seguente:

<%- include('template/header'); %>

Questo codice, in poche parole, permette di prendere il contenuto del file header.ejs, posizionato nella cartella template, e stamparlo nella posizione in cui è stato chiamato. In questo modo si possono riutilizzare gli stessi *header* e *footer* per tutto il sito web mantenendo le stesse funzioni.

### Passaggio di dati

EJS è utile anche per passare dati dalle route alle pagine web. Inoltre permette di usare funzioni per iterare i dati passati o per controllarli.

Grazie a queste funzioni si possono mostrare relative sezioni solo se l’utente ha eseguito il login:

<% if(typeof username != "undefined" && username) { %>

<li class="nav-item">

           <a class="nav-link a-text" href="/logout">logout</a>

      </li>

<% } else { %>

      <li class="nav-item">

<a class="nav-link a-text" href="/login">login</a>

</li>

<% } %>

Oppure si possono usare le variabili passate per mostrare degli errori o delle avvertenze:

<% if(typeof info != "undefined" && info) { %>

<div class="alert alert-info" role="alert">

<%= info %>

</div>

<% } %>

<% if(typeof error != "undefined" && error) { %>

<div class="alert alert-danger" role="alert">

<%= error %>

</div>

<% } %>

Per poter passare i dati bisogna passarli quando si renderizza una pagina, in questo modo:

response.render(\_\_dirname+"/public/home.ejs", {username: "administrator"});

## Sicurezza

In questo capitolo spiego come presto attenzione alla sicurezza dell’applicativo e cosa ho fatto per gestire determinati attacchi.

### SQL Injection

SQL Injection è una tecnica utilizzata per immettere nel codice SQL del codice malevolo. Questo può avvenire quando si controlla la password di un utente:

Select \* from UTENTE where password = “psw” AND username = “username”

In un caso del genere i dati inseriti *psw* e *username* vengono presi da un input testuale, e se non vengono controllati il codice SQL verrà eseguito per come viene scritto, quindi con anche del codice malevolo al posto dello username o della password.

Per contrastare questo attacco utilizzo la libreria mysql di node che permette di usare la funzione apposita per le query. Questa funzione accetta una stringa (codice SQL), un array (dati da inserire nella stringa) ed una funzione di *callback*.

let query = 'INSERT INTO Log(timestamp,descrizione) VALUES(?,?) ';

let values = [formatDateTime(),msg];

connection.query(query, values, function (err, result, fields) {

if(err) console.log(err);

});

La stringa contenente la query viene passata con al posto dei valori inseriti dall’utente dei punti di domanda ‘?’. Questi punti di domanda verranno poi sostituiti dalla funzione *connection.query* con i dati passati all’interno dell’array non prima però di averli sanificati e controllati.

### Broken access control

Questa vulnerabilità consiste nel dare accesso alle risorse ad utenti non autorizzati. Nel mio caso per gestire questa vulnerabilità utilizzo le sessioni.

app.get("/classroomsPage", function(request, response){

let username = request.session.username;

if(username){

response.render(PUBLIC+"/classroomsPage.ejs", {username: username, title:"Lista classi"});

}else{

response.render(PUBLIC+"/home.ejs", {error: "login necessario", title:"home Progetto Kyudd-imi"});

}

});

Per prima cosa prendo dalla sessione lo *username* dell’utente, se inesistente (quindi non ha eseguito il login) lo reindirizzo alla schermata home con un errore, altrimenti lo faccio accedere alla risorsa.

### Salvataggio password

Per il salvataggio delle password utilizzo la libreria Bcrypt che esegue l’hash utilizzando blowfish, esegue il *salt* della password per renderla sicura dagli attacchi *rainbow table* e permette di avere più interazioni per renderla più sicura contro la pratica di *brute force*.

bcrypt

.genSalt(12)

.then(salt => {

return bcrypt.hash("secret", salt)

})

.then(hash => {

console.log('Hash: ', hash)

})

Questo è il codice utilizzato per generare la password.

Per prima cosa viene generato un *salt* casuale eseguito con 12 interazioni, dopodiché tramite questa sequenza di bit viene fatto l’hash della password “secret”. Questa funzione ritornerà poi il risultato dell’hash che per semplicità viene stampato a terminale.

*$2a$16$H/vJylVCGEKmjoERjJ5/l.00.rCkJjkp7pAFFPMeRHEKMyv7iSo8W*

Questo è un possibile hash creato dove segue una formattazione di questo genere: ***$\[algorithm]$[cost]$[salt\][hash].***

Dove l’algoritmo sarà Bcrypt se di valore *$2a* oppure *$2b.* Il costo è invece l’esponente utilizzato per il numero di iterazioni, *2^cost*. ed in fine si avrà il *salt* creato seguito dall’hash della password.

## LOG

I log in questo progetto sono stati pensati per essere salvati in una tabella del database.

Vengono salvati le operazioni di login e logout, in questo modo c’è una base per quando si applicherà la modifica dei dati così da avere una traccia di ciò che è stato fatto.

Il codice che si utilizza è

## Database

Il database è stato creato con il comando “CREATE DATABASE Kyudd\_imi”. Dopodichè, dopo aver utilizzato il database con il comando “use”, sono state create le tabelle.

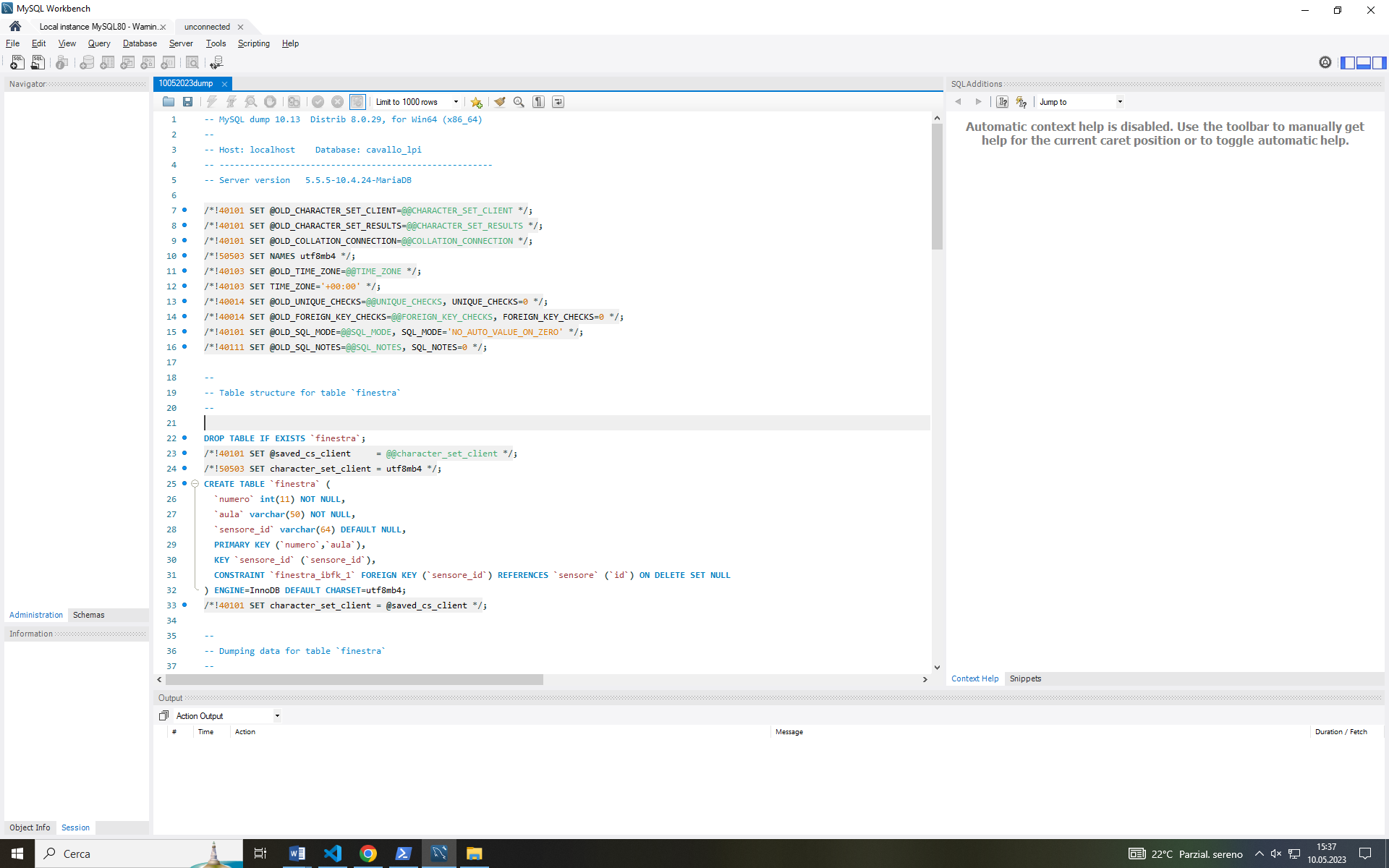


Figura 15 Creazione tabella *finestra*

In questa immagine è rappresentato il codice con cui è stata creata la tabella *finestra.*

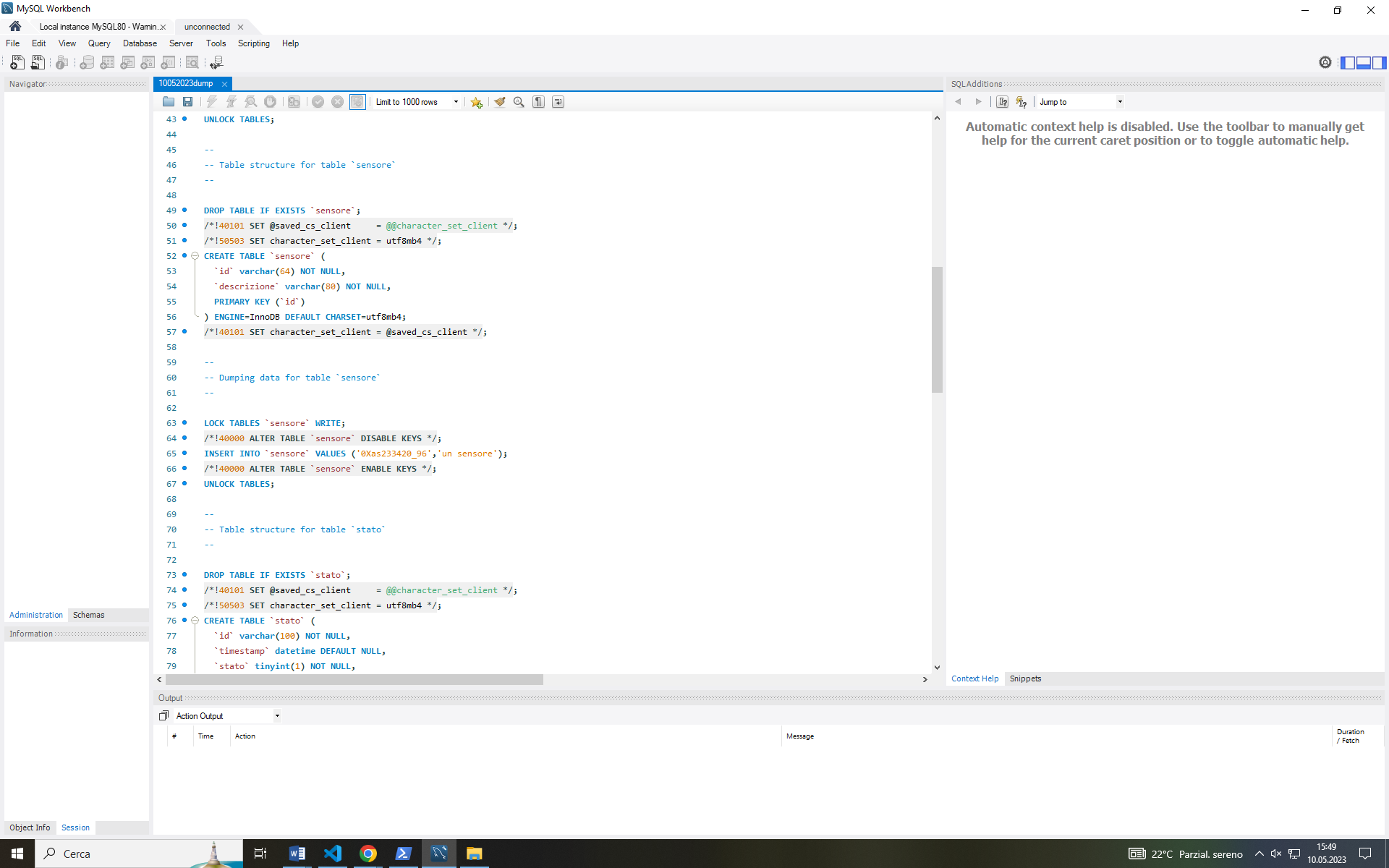


Figura 16 Creazione tabella *sensore*

La seguente immagine contiene la creazione della tabella *sensore.*

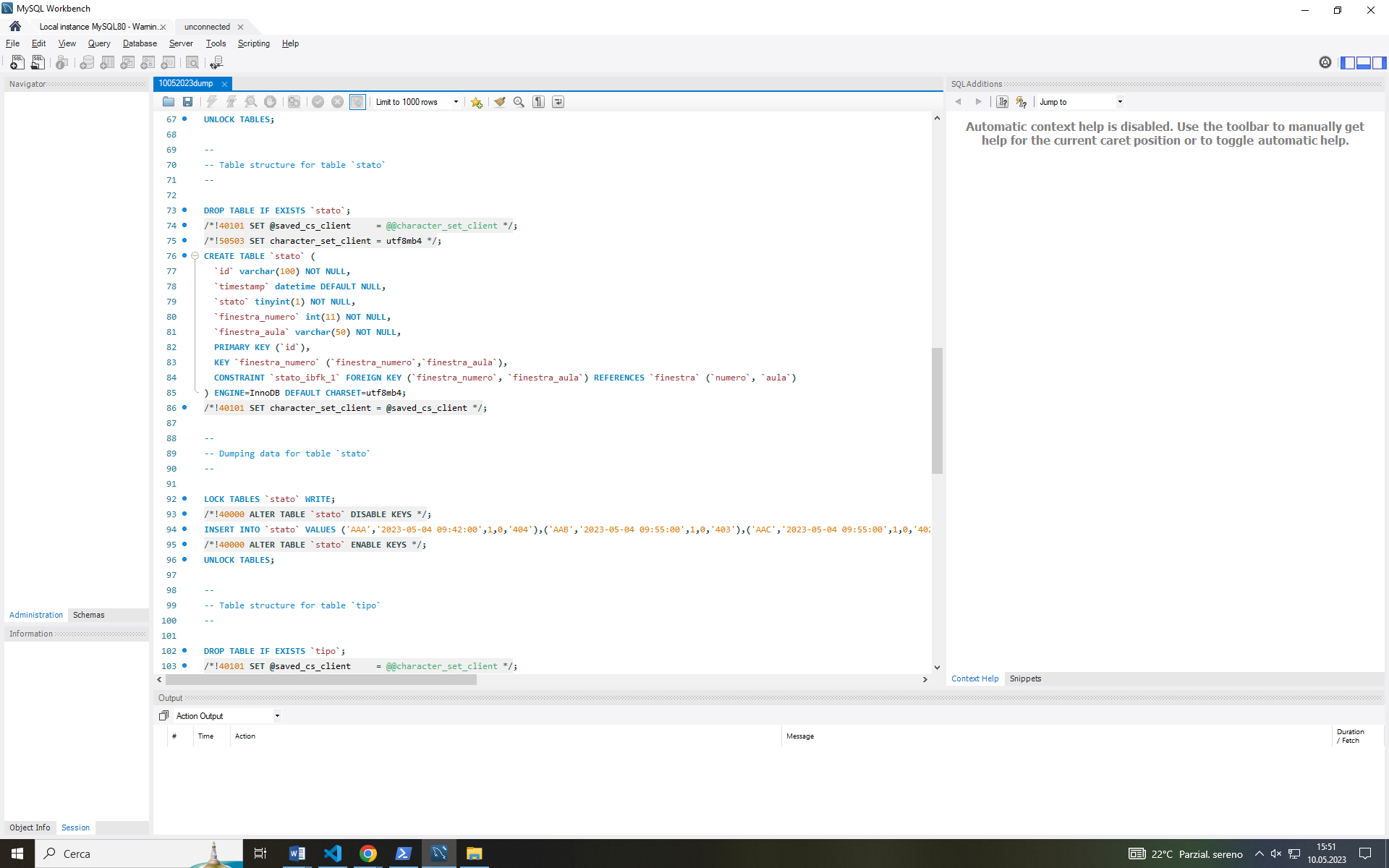


Figura 17 Creazione tabella *stato*

Questo è il codice utilizzato per creare la tabella *stato.*

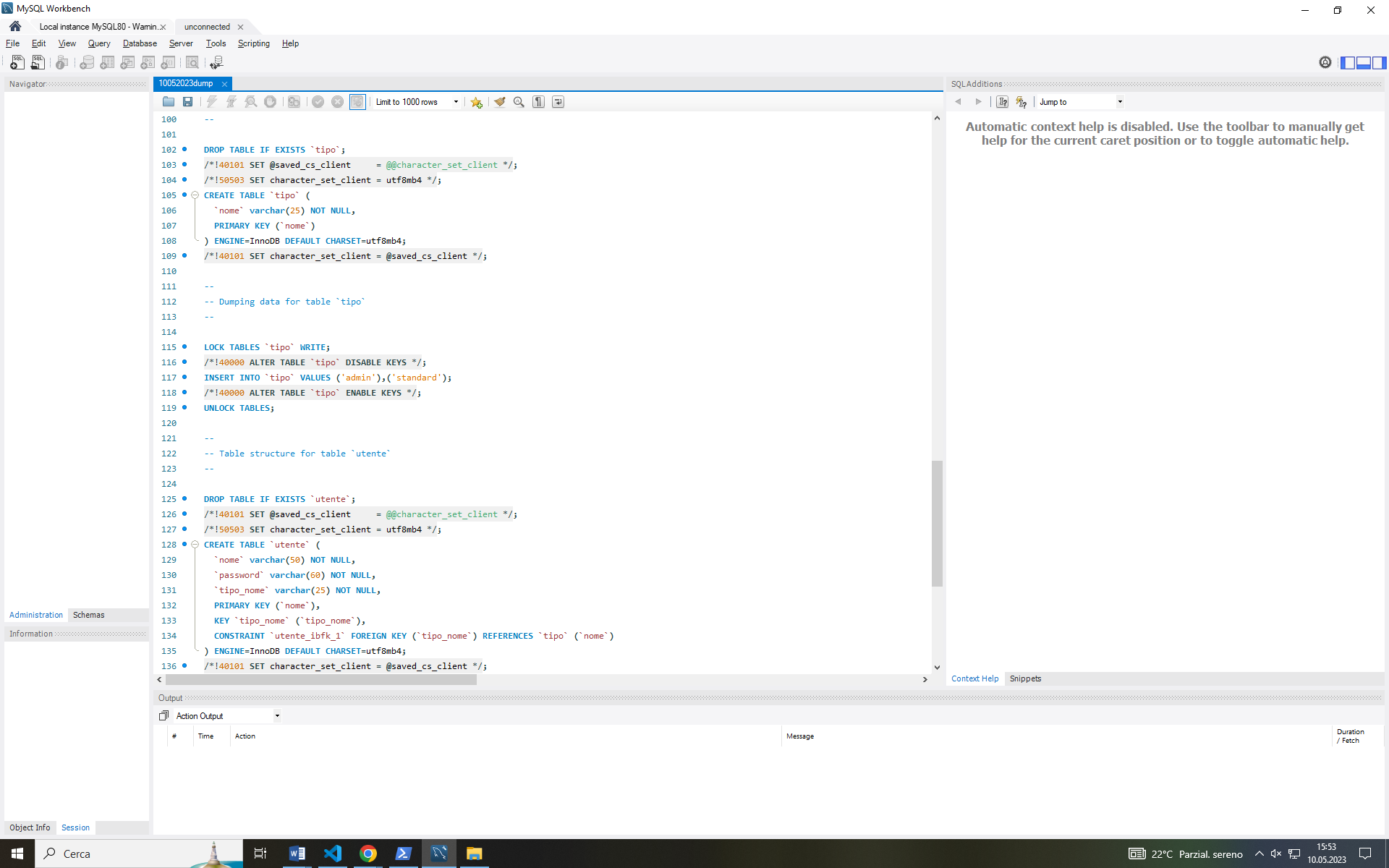


Figura 18 Creazione tabella *tipo*

Questo codice è stato utilizzato per creare la tabella *tipo.*

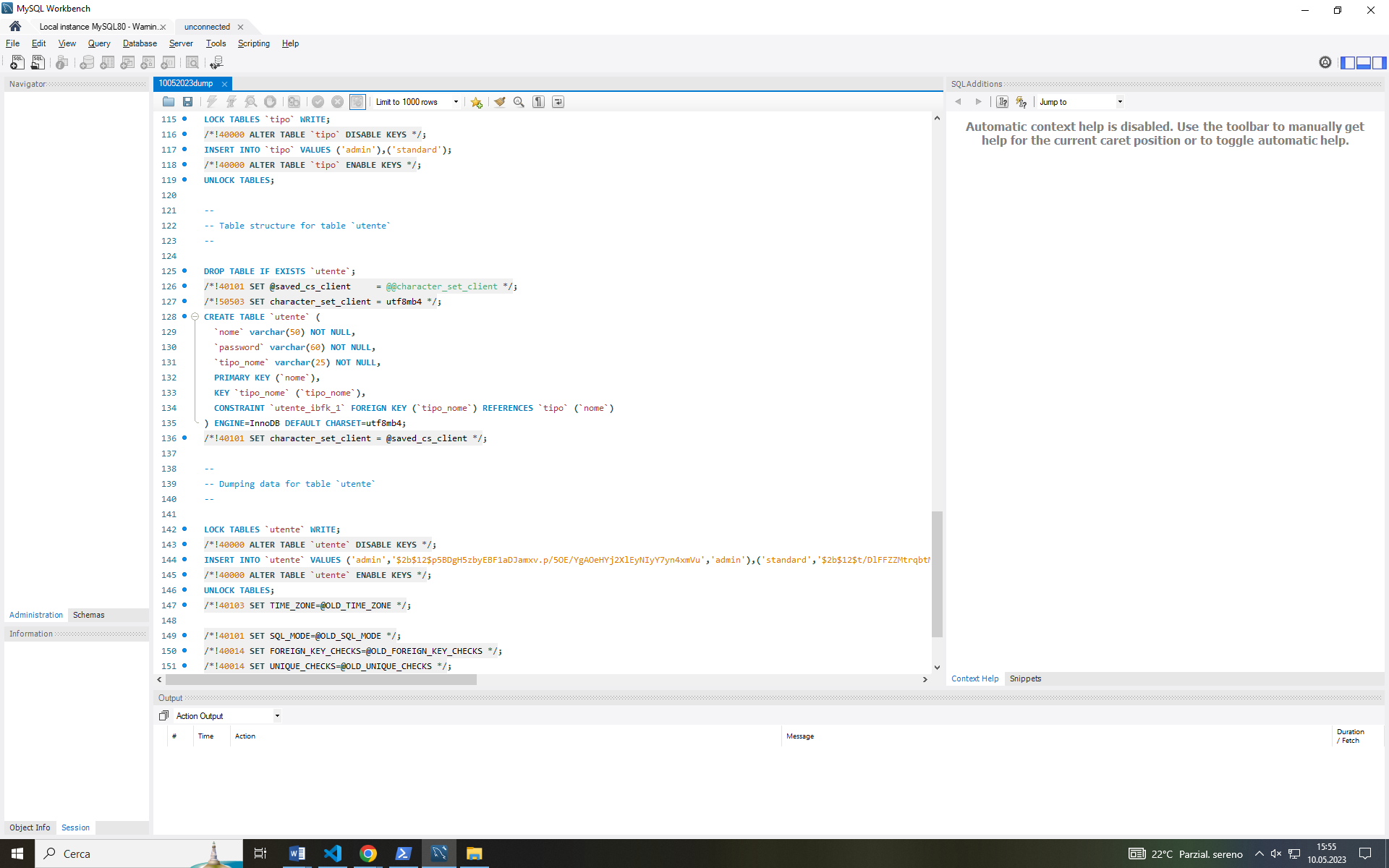


Figura 19 Creazione tabella *utente*

In questa immagine è rappresentato il codice per la creazione della tabella *utente.*

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Sito web locale |
| **Descrizione:** | Il sito web deve essere all’interno della rete del CPT di Trevano. | | |
| **Prerequisiti:** | * Bisogna avere il raspberry raggiungibile. | | |
| **Procedura:** | 1. Sul raspberry eseguire il comando **ip a**. | | |
| **Risultati attesi:** | Le interfacce *eth0* e *wlan0* siano sulla rete **172.16.0.0/24** | | |
| **Note:** | Avendo richiesto una riservazione sul router del MAC address del raspberry, abbiamo la sicurezza che il dispositivo sia sulla rete interna | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Visualizzazione finestre |
| **Descrizione:** | Il sito deve mostrare le finestre aperte o chiuse. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante con i login. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login come usr: *admin* psw: *admin*. 3. Accedere alla pagina delle finestre (/windowsPage). | | |
| **Risultati attesi:** | Bisogna vedere la lista delle finestre aperte e chiuse. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-002 | **Nome:** | Aggiornamento finestre |
| **Descrizione:** | Il sito deve aggiornare la visualizzazione delle finestre. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante con i login. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login come usr: *admin* psw: *admin*. 3. Accedere alla pagina delle finestre (/windowsPage). 4. Cambiare stato ad una finestra nel database. 5. Visualizzare il cambiamento nella pagina web ricaricandola. 6. Cambiare di nuovo lo stato della finestra. 7. Visualizzare il cambiamento nella pagina web ricaricandola. | | |
| **Risultati attesi:** | Ogni cambiamento allo stato delle finestre, deve essere visualizzato nella pagina web. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-003 | **Nome:** | Visualizzazione classi |
| **Descrizione:** | Il sito deve mostrare lo stato delle classi. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante con i login. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login come usr: *admin* psw: *admin*. 3. Accedere alla pagina delle classi (/classroomsPage). | | |
| **Risultati attesi:** | Bisogna vedere la lista delle classi con finestre le aperte o chiuse. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-003 | **Nome:** | Aggiornamento classi |
| **Descrizione:** | Il sito deve aggiornare la visualizzazione delle classo. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante con i login. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login come usr: *admin* psw: *admin*. 3. Accedere alla pagina delle classi (/classroomsPage). 4. Cambiare stato di una classe nel database. 5. Visualizzare il cambiamento nella pagina web ricaricandola. 6. Cambiare di nuovo lo stato di una classe. 7. Visualizzare il cambiamento nella pagina web ricaricandola. | | |
| **Risultati attesi:** | Ogni cambiamento allo stato delle classi deve essere visualizzato nella pagina web. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-004 | **Nome:** | Statistiche finestre |
| **Descrizione:** | Bisogna avere le statistiche sulle finestre. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante con i login. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login come usr: *admin* psw: *admin*. 3. Accedere alla pagina delle finestre. 4. Selezionare una finestra dall’elenco e visualizzarne le info. 5. Quindi accedere alla pagina specifica della finestra. | | |
| **Risultati attesi:** | Bisogna avere una sezione della pagina web con il rapporto del tempo in cui la finestra è stata aperta/chiusa. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-005 | **Nome:** | Visualizzazione minima |
| **Descrizione:** | Bisogna avere almeno una tabella contenente le finestre e relative info. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante con i login. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login come usr: *admin* psw: *admin*. 3. Accedere alla pagina delle finestre. 4. Accedere alla pagina delle classi. 5. Accedere alla pagina del modello 3D. | | |
| **Risultati attesi:** | Ogni le prime 2 pagine web devono avere una lista con le classi o finestre aperte/chiuse. La terza pagina invece deve far visualizzare il modello 3D della scuola con le classi e il loro stato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-006 | **Nome:** | Accessi |
| **Descrizione:** | Gli accessi con nome utente e password validi devono funzionare. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login come usr: *admin* psw: *admin*. | | |
| **Risultati attesi:** | Il login deve andare a buon fine. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-006 | **Nome:** | Accessi |
| **Descrizione:** | Gli accessi con nome utente e password non validi non devono funzionare. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login con un utente inesistente (usr: *paolo* psw: *admin*). 3. Rieseguire il login con un utente esistente ma sbagliando la password (usr: *admin* psw: *prova*). | | |
| **Risultati attesi:** | in tutti e tre i casi il login deve restituire un errore. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-007 | **Nome:** | Utenze |
| **Descrizione:** | Gli utenti devono essere divisi tra admin e standard. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login con un utente usr: *standard* psw: *standard*. 3. Accedere alla pagina home. | | |
| **Risultati attesi:** | la pagina home deve mostrare il nome dell’utente (standard) e il suo tipo (standard). | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-007 | **Nome:** | Utenze |
| **Descrizione:** | Gli utenti devono essere divisi tra admin e standard. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante. * Il database deve essere funzionante. | | |
| **Procedura:** | 1. Accedere al sito web. 2. Eseguire il login con un utente usr: *admin* psw: *admin*. 3. Accedere alla pagina home. | | |
| **Risultati attesi:** | la pagina home deve mostrare il nome dell’utente (*admin*) e il suo tipo (*admin*). | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-012  REQ-008 | **Nome:** | Responsive |
| **Descrizione:** | Il sito web deve essere responsive. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante. | | |
| **Procedura:** | 1. Accedere al sito web da un pc. 2. Eseguire il login come usr: *admin* psw: *admin*. 3. Accedere al sito web da un tablet. 4. Eseguire il login come usr: *admin* psw: *admin*. 5. Accedere al sito web da un telefono. 6. Eseguire il login come usr: *admin* psw: *admin*. | | |
| **Risultati attesi:** | Sia da computer, che dal tablet che dal telefono, le pagine web devono essere ben formattate e il contenuto leggibile. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-013  REQ-009 | **Nome:** | Database con info minime |
| **Descrizione:** | Il database deve avere delle informazioni minime. | | |
| **Prerequisiti:** | * Bisogna avere il database funzionante. | | |
| **Procedura:** | 1. Analizzare il codice presente in /6\_Database/sql.sql | | |
| **Risultati attesi:** | Nel codice devono essere presenti gli attributi che rappresentano:   * Numero della transazione * Data e ora cambiamento di stato * Nome del dispositivo (ESP32) * Numero della finestra o dell’aula * Stato della finestra o dell’aula | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-014  REQ-010 | **Nome:** | Host su Raspberry |
| **Descrizione:** | Il sito web deve essere sopra un Raspberry. | | |
| **Prerequisiti:** | * Bisogna avere il sito funzionante * Bisogna avere il database funzionante. | | |
| **Procedura:** | 1. Accedere tramite l’ip del Raspberry al sito web (172.16.0.109) | | |
| **Risultati attesi:** | Bisogna raggiungere il sito web. | | |

## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Esito | Risultati | Data |
| TC-001 | OK | Il sito web è in locale. | 23.05.2023 |
| TC-002 | OK | È possibile visualizzare l’elenco di tutte le finestre. | 10.05.2023 |
| TC-003 | Ok | La lista delle finestre viene aggiornata con la pagina. | 10.05.2023 |
| TC-004 | OK | È possibile visualizzare l’elenco di tutte le classi. | 12.05.2023 |
| TC-005 | OK | La lista delle classi viene aggiornata con la pagina. | 12.05.2023 |
| TC-006 | OK | È possibile visualizzare le statistiche sul tempo di una finestra in cui è rimasta aperta / chiusa. | 12.05.2023 |
| TC-007 | OK | È disponibile almeno una lista delle finestre con le loro informazioni. | 10.05.2023 |
| TC-008 | OK | Gli accessi validi funzionano. | 09.05.2023 |
| TC-009 | OK | Gli accessi non validi non funzionano. | 09.05.2023 |
| TC-010 | OK | Tra gli utenti ci sono quelli di tipo admin. | 09.05.2023 |
| TC-011 | OK | Tra gli utenti ci sono quelli di tipo standard. | 09.05.2023 |
| TC-012 | OK | Il sito web è responsive. | 16.05.2023 |
| TC-013 | OK | Il database contiene le informazioni richieste. | 17.05.2023 |
| TC-014 | OK | Il sito web è ostato su un raspberry. | 23.05.2023 |

## Mancanze/limitazioni conosciute

Per questo progetto non ci sono state limitazioni che non hanno permesso il completamento del progetto, difatti tutti i requisiti sono stati completati come anche alcuni sviluppi futuri sono stati implementati.

# Consuntivo

Di seguito è rappresentata la progettazione consuntiva del progetto.

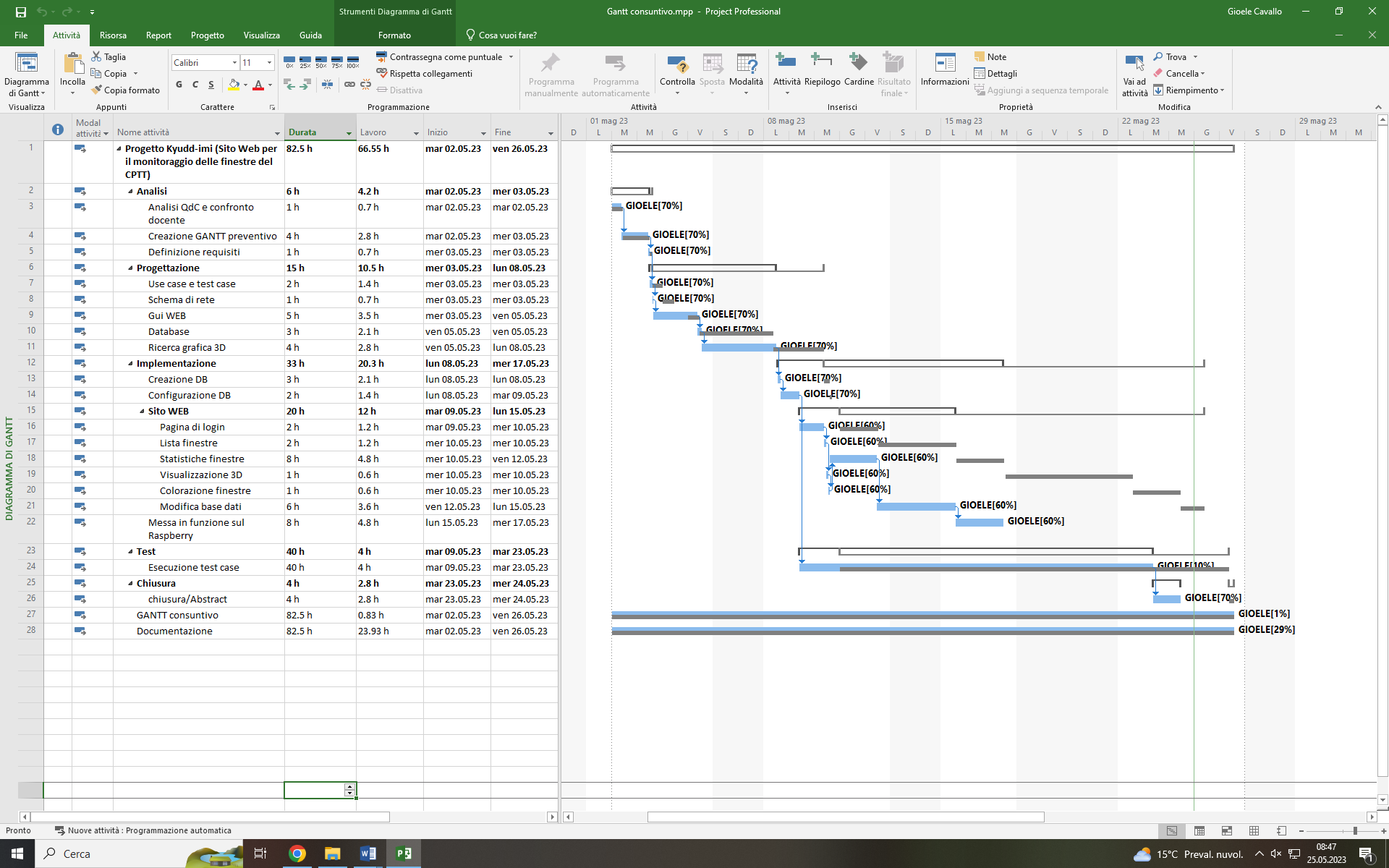


Figura 20 GANTT consuntivo

Si può notare come i tempi del GANTT consuntivo (azzurri) siano diversi dai tempi preventivati (grigio). Infatti iniziata la progettazione le attività si siano svolte con molta facilità, tranne per le GUI web che hanno richiesto un’attenzione maggiore di quella che pensavo all’inizio. Nell’ implementazione questo è ancora più evidente poiché la libreria *three.js*, essendo ben documentata e avendola utilizzata già in passato, è stata facile da implementare per questo caso d’uso. Anche la libreria *datatables* è stata molto utile in questo progetto, perché mia ha fatto risparmiare molto tempo nella creazione delle tabelle e la loro gestione.

# Conclusioni

## Sviluppi futuri

Come sviluppo futuro si può sicuramente migliorare il sito web, che da mobile non è ben proporzionato. Inoltre si potrebbero fare il fetch dei dati nelle tabelle periodicamente, così da non dover riaggiornare la pagina.

Nel futuro si può pensare di aggiungere un modello 3D rappresentante la scuola e le varie finestre / aule, così da avere una rappresentazione grafica dell’edificio e delle finestre aperte.

Una cosa da sviluppare in futuro è l’utilizzo delle tipologie degli utenti. Mi è stato chiesto di implementare questa differenziazione senza effettivamente cambiare

## Considerazioni personali

Personalmente sono molto soddisfatto di questo progetto in quanto ho utilizzato tecnologie che ho già utilizzato in passato ma provandole singolarmente, quindi integrarle è stato molto interessante.

Inoltre sono contento di essere riuscito ad implementare uno sviluppo futuro durante questo progetto.

# Bibliografia

## Sitografia

* <https://blender.stackexchange.com/questions/109/how-can-i-use-an-array-modifier-to-create-individually-manipulatable-objects>, *How can I use an array modifier to create individually manipulatable objects*, 03-05-2023
* <https://www.youtube.com/watch?v=CWdbrRNCJIA>, *Video tutorial on array modifiers in* Blender, 03-05-2023
* <https://datatables.net/>, *DataTables official website*, 08-05-2023
* <https://kinsta.com/it/knowledgebase/node-js/>, *Node.js knowledge base on Kinsta*, 12-05-2023
* <https://kinsta.com/it/knowledgebase/cos-e-express-js/>, *What is Express.js? - Kinsta knowledge base*, 12-05-2023
* <https://tree.nathanfriend.io/>, *A visual tree editor for 3D scenes*, 12-05-2023
* <http://expressjs.com/en/resources/middleware/session.html>, *Express.js middleware for session management*, 12-05-2023
* <https://glue-labs.com/articoli/three-js-il-framework-il-rendering-3d-sul-web>, *Three.js - Il framework per il rendering 3D sul web*, 12-05-2023
* <https://discourse.threejs.org/t/clientx-and-window-innerwidth-question/33271/14>, *Question about clientX and window.innerWidth in Three.js*, 15-05-2023
* <https://huobur.medium.com/how-to-setup-wifi-on-raspberry-pi-4-with-ubuntu-20-04-lts-64-bit-arm-server-ceb02303e49b>, *How to Setup Wi-Fi on Raspberry Pi 4 with Ubuntu 20.04 LTS (64-bit ARM Server),* 16-05-2023
* <https://askubuntu.com/questions/155424/changing-tty-keyboard-layout-on-a-server>, *Changing TTY keyboard layout on a server*, 16-05-2023
* <https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04>, *How to Install MySQL on Ubuntu 20.04*, 16-05-2023
* <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-ubuntu-22-04>, *How to Set Up a Node.js Application for Production on Ubuntu 22.04*, 17-05-2023
* <https://blog.logrocket.com/password-hashing-node-js-bcrypt/>, *Password Hashing in Node.js with bcrypt*, 17-05-2023
* [https://chat.openai.com](https://chat.openai.com/), *OpenAI Chat Platform*, 16-05-2023
* <https://it.wikipedia.org/wiki/Bcrypt>, *Bcrypt su Wikipedia*, 22-05-2023
* <https://it.wikipedia.org/wiki/SQL_injection>, *SQL Injection su Wikipedia*, 22-05-2023
* <https://www.tutorialspoint.com/run-a-script-on-startup-in-linux>, *How to Run a Script on Startup in Linux*, 24-05-2023

## Glossario

|  |  |
| --- | --- |
| Termine | Descrizione |
| Brute force | Tattica utilizzata per provare a indovinare tramite molteplici tentativi le credenziali d’accesso. |
| Chiave composta | Combinazione di più attributi che formano assieme formano un’ unicità nella tabella. |
| Chiave primaria | Attributo non replicabile all’interno della tabella. |
| Express | Framework opensource Node.js per la programmazione di API e applicazioni web. |
| fetch | Termine utilizzato per indicare la richiesta di dati da una base dati. |
| framework backend | Strumento software utilizzato per automatizzare e semplificare il processo web. |
| GET | Metodo http per richiedere al server un’informazione o risorsa. |
| gLTF | Formato file standard per scene e moelli 3D. |
| Hashing | Operazione per trasformare un testo in una sequenza di bit dal quale non si può ritornare al testo originale. |
| middleware | Applicazioni o funzioni che vengono eseguite dalla ricezione della richiesta del client fino all’invio della risposta. |
| Multi Thread | Supporto per eseguire più operazioni contemporaneamente utilizzando più thread. |
| Node.js | Runtime system open source multipiattaforma per l’esecuzione javascript. |
| NPM | Gestore di pacchetti per Node.js . |
| POST | Metodo http per inviare dei dati al server. |
| QdC | Quaderno dei Compiti: il documento consegnato dal docente responsabile che descrive le specifiche del progetto. |
| Query | Comando per ricavare uno o più risultati da una base dati. |
| Rainbow table | Tabella usata per avere una relazione tra un testo e il suo hash. |
| runtime system | Software che fornisce servizi necessari all’esecuzione di un programma. |
| salting | Operazione che si esegue nell’hashing. Consiste nell’inserimento di una sequenza casuale di bit all’interno della stringa da criptare. |
| Single Thread Event Loop | È l’utilizzo di una pool finita di thread a disposizione che servono a rispondere alle richieste dei client. |
| Swimmlane | Elemento grafico usato nei diagrammi di flusso che suddivide graficamente le varie operazioni e i loro operatori. |
| Thread | Una parte di un processo viene definito Thread. Queste thread vengono eseguite contemporaneamente. |
| Three.js | Libreria javascript per creare e visualizzare componenti 3D. |

## Indice delle figure

[Figura 1 Use case 9](#_Toc135985273)

[Figura 2 GANTT preventivo 11](#_Toc135985274)

[Figura 3 Schema di rete 16](#_Toc135985275)

[Figura 4 Diagramma database 18](#_Toc135985276)

[Figura 5 Diagramma di flusso 20](#_Toc135985277)

[Figura 6 Interfaccia home 21](#_Toc135985278)

[Figura 7 Interfaccia lista finestre 22](#_Toc135985279)

[Figura 8 Interfaccia finestra 23](#_Toc135985280)

[Figura 9 Interfaccia lista aule 24](#_Toc135985281)

[Figura 10 Interfaccia aula 25](#_Toc135985282)

[Figura 11 Interfaccia login 26](#_Toc135985283)

[Figura 12 Interfaccia modello 3D 27](#_Toc135985284)

[Figura 13 codice post login 29](#_Toc135985285)

[Figura 14 codice logout 30](#_Toc135985286)

[Figura 15 Creazione tabella *finestra* 36](#_Toc135985287)

[Figura 16 Creazione tabella *sensore* 36](#_Toc135985288)

[Figura 17 Creazione tabella *stato* 37](#_Toc135985289)

[Figura 18 Creazione tabella *tipo* 37](#_Toc135985290)

[Figura 19 Creazione tabella *utente* 37](#_Toc135985291)

[Figura 20 GANTT consuntivo 45](#_Toc135985292)

# Allegati

Elenco degli allegati:

* Installazione programma (README.md)
* Qdc (1\_Qdc)
* Abstract (2\_Abstract)
* Diari di lavoro (4\_Diari)
* Sito web (5\_Applicativo)
* Maunuali (8\_Manuali)