Corso di Programmazione Web e Mobile
A.A. 2020-2021

Registrazione di accesso alle aule universitarie

◖ Luca, Forzano, 909398 ◗

Deploy app: https://lf-pwm-project.herokuapp.com
Github: https://github.com/Ciussino/PWM_Progetto.git

Autore: luca forzano

Registrazione di accesso alle aule universitarie

Programmazione Web e Mobile – Luca Forzano

1. Introduzione

Il progetto realizzato consiste in una semplice applicazione web che permette agli studenti dell'Università Statale di Milano di prenotare l'accesso alle aule universitarie.

Lo scopo di tale applicazione è quello di monitorare gli accessi alle aule al fine di rispettare le norme di distanziamento introdotte per contenere l'epidemia Covid-19.

Il backend dell'applicazione è stato sviluppato utilizzando NodeJS in combinazione con il framework Express. I dati relativi agli utenti ed alle prenotazioni sono salvati all'interno del database NoSQL MongoDB.

Il frontend è stato realizzato utilizzando HTML5, CSS3, JavaScript e EJS come template engine.

1.1. Breve analisi dei requisiti

1.1.1. Destinatari

Capacità e possibilità tecniche.

I destinatari dell'applicazione sono tutti gli studenti che hanno accesso alle aule universitarie, pertanto le capacità tecniche variano da esperti di informatica e nello specifico della programmazione a persone totalmente inesperte.

Per tale motivo l'applicazione è stata realizzata nella maniera più semplice e intuitiva possibile al fine di rendere agevole l'utilizzo del software a tutti gli studenti indipendentemente dalle proprie capacità tecniche.

Gli utenti che accedono a tale applicazione risultano già consapevoli dello scopo della stessa, pertanto non risulta necessaria una guida per lo studente. Inoltre, lo studente avendo già conoscenza delle varie applicazioni web presenti sui sistemi dell'Università Statale di Milano (Unimia, Unimi ecc…) non presenterà alcuna difficoltà nell'utilizzare l'applicazione in oggetto, infatti le pagine sono composte da semplici form (login e prenotazione) dove inserire le proprie informazioni, da una tabella di riepilogo per visualizzare le prenotazione attive e da elementi cliccabili (button) dove poter confermare / rimuovere i dati e muoversi all'interno dell'applicazione stessa.

Infine, è stato realizzato un responsive design per permettere all'utente di utilizzare l'applicazione su sistemi Personal Computer / Laptop ma anche su dispositivi mobili (es. Smartphone).

Linguaggio.

L'applicazione non fa uso di un particolare linguaggio tecnico o terminologia settoriale, in quanto deve essere accessibile a tutti gli studenti dell'università indipendentemente dalle capacità tecniche dell'utente.

Motivazione.

L'applicazione è stata realizzata per fini educational in quanto utilizzata dagli studenti dell'Università Statale di Milano per garantire il rispetto delle norme introdotte per contenere l'epidemia Covid19.

L'obiettivo è quindi quello di poter usufruire delle aule universitarie in maniera conforme alle norme di distanziamento sociale, tutto ciò è possibile effettuando l'accesso dal proprio laptop o smartphone prenotando l'aula e/o il laboratorio desiderato nella data scelta. L'applicazione permette all'utente di effettuare le seguenti operazione:

- L'utente può effettuare fino ad un massimo di cinque prenotazioni contemporanee
- L'utente non può prenotare più di un'aula nella stessa giornata
- Non è possibile prenotare un'aula che risulta completamente occupata alla data selezionata

Basandoci sul modello di Bates, possiamo identificare l'utente in un'azione di "Searching" in quanto il grado di consapevolezza dell'informazione è diretto perché l'utente accede a tale applicazione per un motivo ben preciso e conosce già la finalità del sito web. Inoltre, la volontarietà dell'utente è attiva in quanto l'accesso avviene solo nel caso in cui l'utente decida di usufruire delle aule / laboratori dell'università.

In sintesi, possiamo quindi definire che l'utente è consapevole delle funzionalità dell'applicazione e quindi recupera le informazioni in maniera rapide e precisa.

	Active	Passive
Directed	**Searching**	**Monitoring**
Undirected	Browsing	Being Aware

1.1.2. Modello di valore

Il valore principale dell'applicazione e quello di poter fornire un servizio a tutti gli studenti per usufruire delle aule / laboratori dell'università in questo particolare periodo, inoltre permette anche ai docenti di poter monitorare gli accessi ed il numero di utenti che accedono alle proprie aule per una maggiore sicurezza di tutto il personale universitario.

Tale applicazione potrà essere utilizzata anche a seguito alla situazione che stiamo vivendo in quanto potrà permettere all'università una gestione ed organizzazione più efficiente delle aule presenti all'interno dell'università evitando sovraffollamenti delle stesse.

Il bacino di utenza che può essere definito per tale applicazione è relativo al numero degli studenti dell'università che hanno l'intenzione di accedere alle strutture universitarie.

Tale applicazione potrà essere anche diffusa ad altre università italiane che potranno ricevere un vantaggio nella gestione degli accessi alle aule / laboratori.

1.1.3. Flusso dei dati

I contenuti / dati vengono generati direttamente dall'utente attraverso l'applicazione, vengono archiviati ed organizzati all'interno di un database MongoDB e le diverse iterazioni con il database stesso vengono gestite tramite API con l'utilizzo del paccheto npm Mongoose.

I dati vengono elaborati in formato JSON sia in fase di salvataggio all'interno del database sia in fase di estrazione delle informazioni per la visualizzazione in frontend.

I contenuti recuperati dal database vengono forniti all'utente attraverso delle chiamate dirette con Mongoose e fornite al client attaverso il template engine EJS.

1.1.4. Aspetti tecnologici

HTML5:

La struttura delle pagine web è stata realizzata attraverso il linguaggio di Markup HTML5 che permette in maniera semplice ed efficace di ottenere una struttura adeguata a qualsiasi pagina web e compatibile con tutti i dispositivi e sistemi operativi.

Template Engine EJS:

Il templating delle pagine è stato realizzato con EJS (Embedded JavaScript Templates) che fornisce in maniera semplice ed efficace la possibilità di recuperare le informazioni lato backend e dal database stesso.

La scelta di tale template engine si è basata sulla semplicità di utilizzo, di recupero delle informazioni in rete e della vasta presenza di tutorial in merito.

CSS3:

Il design delle diverse pagine web è stato realizzato utilizzando CSS3 che permette in maniera semplice ed efficace di fornire alle pagine dell'applicazione un aspetto e una presentazione delle informazioni più comprensibile sia all'utente finale sia durante la fase di programmazione.

JavaScript:

Il linguaggio JavaScript è stato utilizzato sia per la programmazione lato client (frontend) per la gestione di messaggi di errore, sia lato server (backend) per la creazione di API e delle diverse connessioni con gli elementi dell'applicazione.

NodeJS:

Per lo sviluppo lato server è stato utilizzato NodeJS una piattaforma che permette di utilizzare JavaScript non solo lato client ma anche serverside.

Express:

Il framework utilizzato è Express.js che in combinazione con NodeJS permette di creare applicazioni web e API in maniera semplice ed efficace.

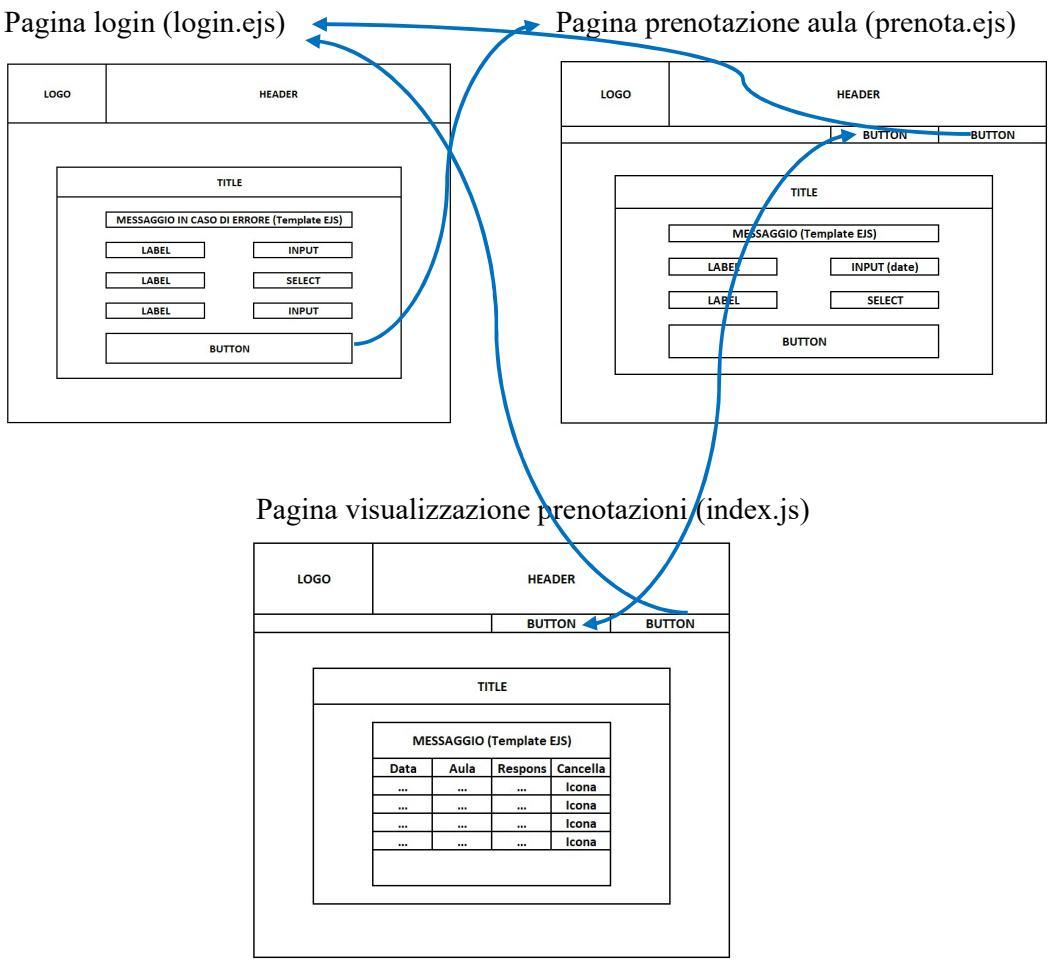
NPM packages:

A supporto dello sviluppo lato server sono stati utilizzati diversi npm packages riportati di seguito:

- Mongoose: utilizzato per la connessione e la creazione dei modelli del DB e per la gestione delle chiamate al database in modalità asincrona

- Bcryptjs: utilizzato per effettuare un hash delle password sia in fase di creazione utente sia in fase di login
- Express-Session: utilizzato per la creazione di una sessione ogni qualvolta venga effettuato un accesso all'applicazione in modo da poter garantire una sicurezza maggiore
- Body-Parser: utilizzato per gestire le informazioni provenienti dal body dell'applicazione e gestirle lato server
- Ejs-Mate: utilizzato per settare e gestire il template engine dell'applicazione
- Nodemon: utilizzato in fase di sviluppo per permettere un aggiornamento automatico dell'applicazione ad ogni salvataggio

2. Interfacce



Le interfacce utente dell'applicazione sono tre, la pagina relativa al login dell'applicazione, la pagina per effettuare la prenotazione dell'aule e la pagina per visualizzare le prenotazioni attive già effettuate con la possibilità di eliminarle.

Tutte le pagine hanno una struttura molto semplice in quanto sono formate da un header che contiene il logo ed il nome dell'applicazione, le pagine prenota.ejs e index.ejs contengono anche una barra di navigazione che permette di effettuare il logout oppure di passare da una pagina all'altra e viceversa.

All'interno della pagina login.ejs è presente un form che permette di inserire le proprie credenziali e viene poi effettuato un check lato server all'interno del database. Se le informazioni relative alle credenziali inserite non sono corrette viene visualizzato un messaggio di errore attraverso il template engine EJS, nel caso in cui le credenziali risultano corrette si viene reindirizzati direttamente alla pagina delle prenotazioni aula.

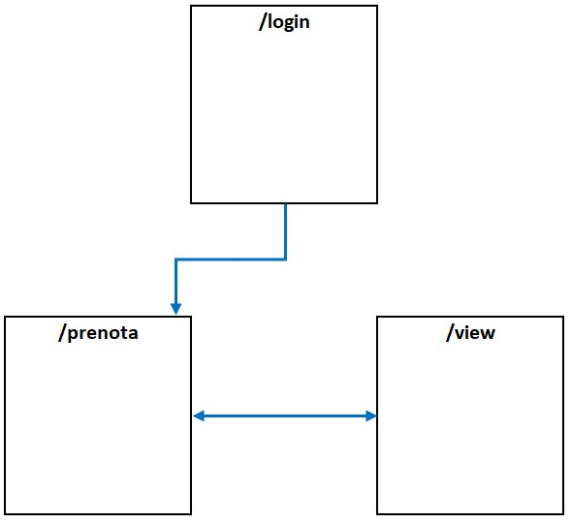
La pagina prenota.js contiene un semplice form che permette all'utente di selezionare la data e l'aula in cui desidera effettuare la prenotazione, nel caso di prenotazione corretta o se risulta un errore durante la prenotazione viene visualizzato il messaggio di riferimento tramite il template engine EJS.

Infine, alla pagina index.ejs è possibile visualizzare una tabella che contiene le prenotazioni attive effettuate dall'utente loggato, viene fornita anche la possibilità di eliminare le prenotazioni effettuate tramite una semplice icona, se l'operazione va' a buon fine viene visualizzato il messaggio di conferma tramite il template engine EJS.

3. Architettura

3.1. Diagramma dell'ordine gerarchico delle risorse

L'applicazione è formata da tre pagine con una gerarchia molto semplice, è infatti necessario accedere inizialmente alla pagina del login corrispondente all'URI **/login,** una volta effettuato il login correttamente si accede direttamente alla pagina relativa alle prenotazioni aula corrispondente all'URI **/prenota** infine dalla pagina prenota è possibile accedere all'URI **/view** che contiene le informazioni relative alle prenotazioni attive già effettuate.



3.2. Descrizione delle risorse

L'applicazione è formata da quattro risorse, create attraverso il routing di ExpressJS.

Le risorse dell'applicazione sono:

- **/login:**

All'interno della risorsa login viene effettuata una chiamata GET che restituisce la vista della pagina in cui effettuare il login (login.ejs) ed una chiamata POST che permette di confrontare le informazioni inserite dall'utente nel form con quelle presenti all'interno del database in modo da verificare la correttezza dei dati relativi all'autenticazione.

Viene poi passato un parametro chiamato utentecheck che permette di effettuare e visualizzare le prenotazioni relative al solo utente loggato.

- **/prenota:**

La risorsa prenota effettua una chiamata GET che restituisce la vista della pagina in cui effettuare la prenotazione delle aule (prenota.ejs) ed una chiamata POST dove vengono effettuate dei controlli sui dati inseriti dall'utente e se conformi a quanto richiesto viene salvata la prenotazione all'interno del database. Ad ogni prenotazione viene salvato all'interno del database anche l'utente loggato tramite il parametro utentecheck.

- **/view:**

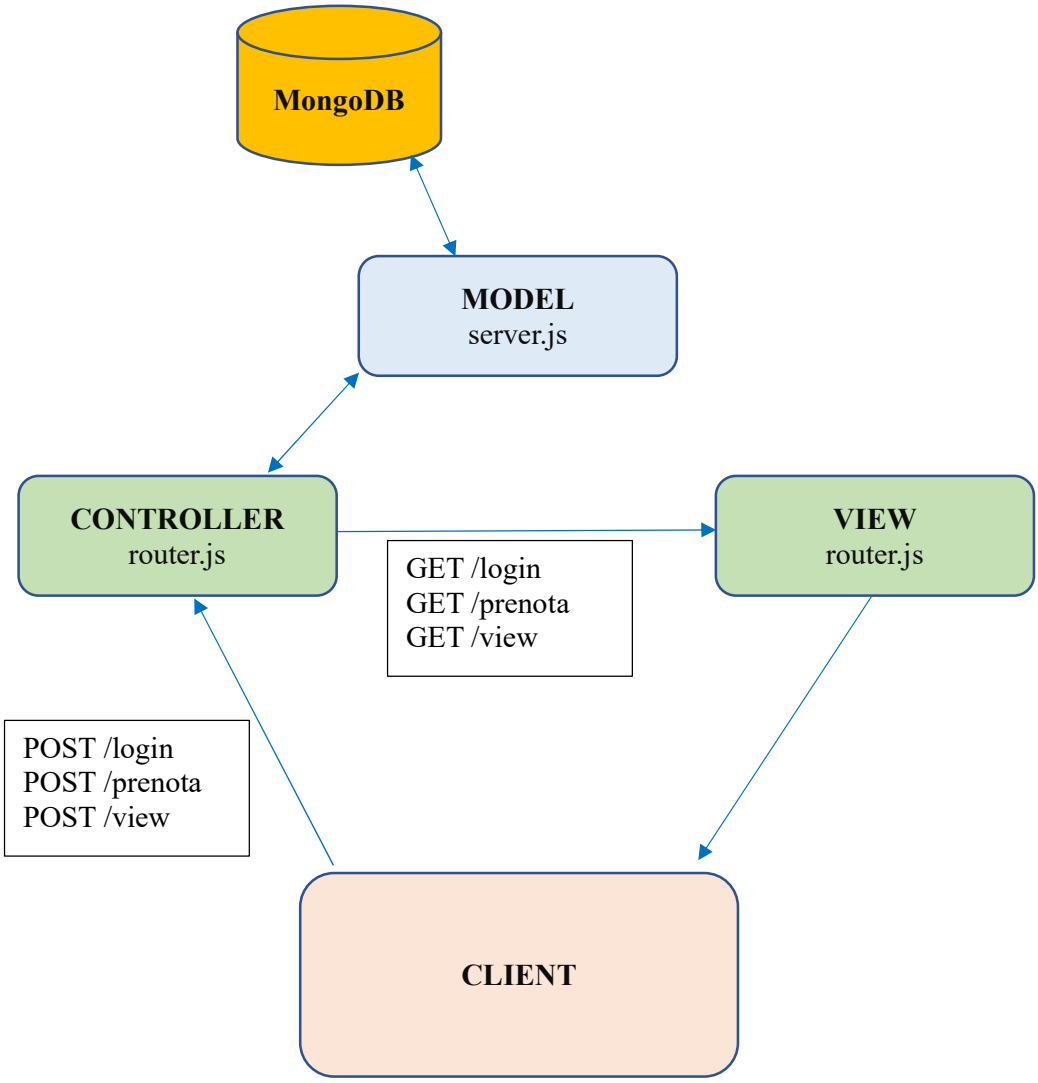
La risorsa view effettua una chiamata GET che restituisce attraverso la selezione dal database i dati relativi alle prenotazioni attive del singolo utente loggato che verranno poi visualizzate attraverso il template engine EJS.

Viene poi effettuata una chiamata POST che permette all'utente di eliminare la singola prenotazione selezionata.

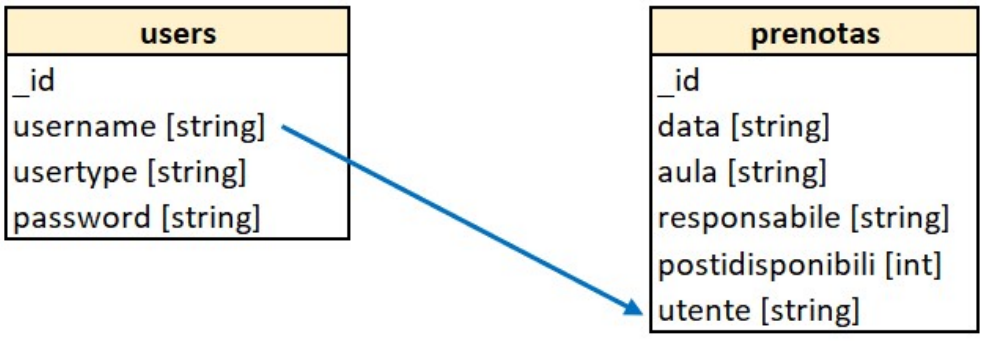
- **/logout:**

Tale risorsa ha il solo compito di eliminare la sessione relativa all'utente loggato e reindirizzare alla pagina /login.

Schema MVC relativo all'applicazione:



3.3. Diagramma database:



4. Codice

Creazione della sessione e della funzione di autenticazione:

```
app.use(session({
    secret:"key",
    resave:"false",
    saveUninitialized: false,
    cookie: {
        maxAge: 3600000 //durata 1 ora
    }
}));
```

```
router.post("/login", async (req,res) => {

    const {username, usertype, password} = req.body;

    var user = await User.findOne({username, usertype});

    if(!user){
        res.render("login/login", {message: "Nome utente e/o Password errati!"});
    }

    const checkPsw = await bcrypt.compare(password, user.password);
    if(!checkPsw){
        res.render("login/login", {message: "Nome utente e/o Password errati!"});
    }

    req.session.isAuth = true;
    utenteCheck = req.body.username;
    res.redirect("/prenota");
});
```

Verifica dei parametri inseriti dall'utente e salvataggio della prenotazione in assenza di errore:

```
Prenota.countDocuments({data: data, utente: utente}, (err, countPerData) => {
    if(err){
        console.log(err);
    }
    else if(countPerData > 0) {
        res.render("prenota/prenota", {message: "Prenotazione già esistente per la data selezionata!", success: ""});
    }
    else(Prenota.countDocuments({utente: utente, data: {$lt: oggi}}, (err, countScadute) => {
        if(err) {
            console.log(err);
        }
        else(Prenota.countDocuments({utente: utente}, (err, countPerUtente) => {
            if(err){
                console.log(err);
            }
            else if(countPerUtente - countScadute > 4) {
                res.render("prenota/prenota", {message: "Raggiunto il numero massimo di prenotazioni contemporanee!", success: ""});
            }
            else(Prenota.countDocuments({data: data, aula: aula}, async (err, countPostiAula) => {
                if(err){
                    console.log(err);
                }
                else if(countPostiAula == postiDisponibili) {
                    res.render("prenota/prenota", {message: "L'aula selezionata risulta al completo!", success: ""});
                }
                else {
                prenota = new Prenota({
                    data,
                    aula,
                    responsabile,
                    postiDisponibili,
                    utente
                })

                await prenota.save();
                res.render("prenota/prenota", {message: "", success: "Prenotazione avvenuta con successo!"});
                }
```

Visualizzazione dei dati recuperati dal database all'interno della tabella presente in /view

```
<div class="page">
    <div class="view">
        <div class="message">
            <h4><%= message %></h4>
        </div>
        <table>
            <thead>
                <tr>
                    <th>Data</th>
                    <th>Aula</th>
                    <th>Responsabile</th>
                    <th>Cancella</th>
                </tr>
            </thead>
            <tbody>
                <% prenota.forEach(function(prenotazione) { %>
                    <tr>
                        <td><%= prenotazione.data %></td>
                        <td><%= prenotazione.aula %></td>
                        <td><%= prenotazione.responsabile %></td>
                        <td>
                            <form action="/view/<%= prenotazione._id%>" method="POST">
                                <button class="cancella"><img src="../img/bin.svg" alt="IconNotFound"></button>
                            </form>
                        </td>
                    </tr>
                <% }) %>
            </tbody>
        </table>
```

5. Nota bibliografica e sitografica

(1) Don Gosselin, JavaScript, Apogeo Education, 2000

(2) https://stackoverflow.com/ Consultazione forum per problemi di sviluppo

(3) NPM Packages: https://www.npmjs.com/

(4) Immagini:

1. Sfondo /login: https://www.pexels.com/photo/gray-laptop-computer-showing-html-codes-in-shallow-focus-photography-160107/
2. Logo Unimi: https://it.wikipedia.org/wiki/File:Logo_Universit%C3%A0_degli_Studi_di_Milano.svg#/media/File:Logo_Universit%C3%A0_degli_Studi_di_Milano.svg
3. Immagini /prenota e /view: https://undraw.co/
4. Icona /view: https://icomoon.io/