UNIVERSITÀ DEGLI STUDI DI TRENTO

Dipartimento di ingegneria e scienza dell'informazione

Corso di Laurea in Informatica

PROGETTO INGEGNERIA DEL SOFTWARE

AULE STUDIO UNITN

GRUPPO T43

Sarcletti Davide - Rizzi Mattia - Cattoni Alex

Anno accademico 2022/2023

INDICE

Obiettivi del documento

Il documento D4 ha l'obiettivo di mostrare le informazioni e le caratteristiche richieste per realizzare alcune tra le principali funzionalità del sito web Aule Studio Unitn.
Verranno mostrati i requisiti richiesti dall'utente e tutte le funzioni proposte dal sito per poter essere utilizzato correttamente.

Successivamente verranno mostrate le APIs utilizzate per poter implementare queste funzionalità, ovvero la gestione di una prenotazione, la gestione delle aule studio assieme alla creazione di una prenotazione.

Verrà inoltre mostrata la strumentazione che permette al sito di funzionare, come il database MongoDB utilizzato e il diagramma User Flow.

1. User Stories

Attraverso le User Stories si descrivono le richieste principali che effettua l'utente utilizzando il sito web.
Questa descrizione permette di mostrare come una richiesta dell'utente restituirà una determinata funzione, oltre che collegarla ai requisiti funzionali e non funzionali di cui abbiamo discusso in precedenza.

Le User Stories riportate in seguito si focalizzano sulle principali funzionalità dell'attore "Utente", ovvero colui che dovrà loggarsi nel sito, visualizzare gli orari e i posti disponibili per effettuare una prenotazione in un'aula studio, gestire una prenotazione, visualizzare la sua area personale ed effettuare il logout.

Nella tabella vengono riportate le User Stories attraverso una descrizione in linguaggio naturale delle principali funzionalità specificate nel paragrafo precedente.
La colonne della tabella rappresentano rispettivamente:

- User Stories: elenco delle user stories
- Descrizione User Story: descrizione richieste dell'utente
- Requisiti Funzionali: RF descritti precedentemente
- Requisiti non funzionali : RNF descritti precedentemente

Tabella delle **User Stories** e riferimenti ai RF e RNF nella tabella sottostante:

US	Descrizione User Story	RF	RNF
US2	Come utente, voglio poter visualizzare i dati personali in una pagina apposita	RF5	RNF3, RNF4
US3	Come utente, voglio poter interagire con le mie prenotazioni attive in una pagina apposita	RF3	RNF6, RNF4
US4	Come utente, voglio poter effettuare una prenotazione nella pagina delle aule studio il giorno desiderato	RF6	RNF5, RNF4
US5	Come utente, voglio poter visualizzare i dati delle aule studio con i relativi posti disponibili	RF6, RF3	RNF5, RNF4
US6	Come utente amministratore, voglio poter gestire le aule studio presenti nel database	RF7	RNF5, RNF4

2. Features

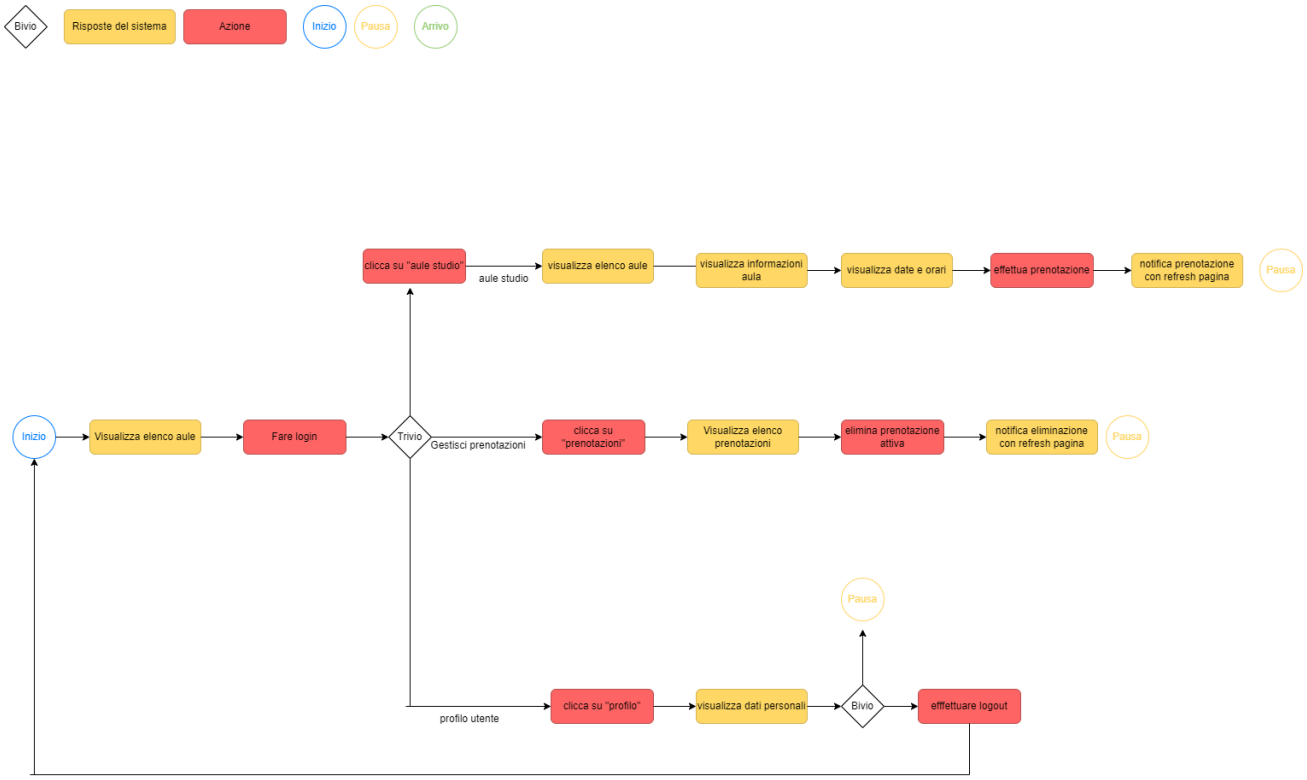
In seguito vengono elencate tutte le features che offre il sito web.

US	Feature Index	Feature
US1	F1	Visualizzare le aule studio con le loro informazioni
US5	F3	Visualizzare le date per prenotare
US5	F4	Visualizzare i posti disponibili per ogni aula studio
US1	F5	Estrarre le informazioni relative ad ogni aula studio tramite API locale
US2	F6	Loggarsi nel sito tramite API di controllo delle credenziali UniTN
US4	F7	Effettuare una prenotazione
US3	F8	Gestire una prenotazione tramite l'apposita pagina
US2	F9	Visualizzare le informazioni personali dello studente
US2	F10	Estrarre i dati dello studente tramite un API locale
US6	F11	Gestire le aule studio tramite API locale
US3	F12	Visualizzare i dati delle prenotazioni attive dello studente
US4	F13	Salvare i dati della prenotazione nel database MongoDB
US2	F14	Effettuare il logout

3. User Flow

Nel seguente capitolo si riportano gli User Flow che riguardano le principali funzionalità per il ruolo dell'utente nell'applicazione web ASU.
L'utente ha accesso a tre funzioni principali:la Prenotazione, il Profilo Utente e la Visualizzazione delle Prenotazioni.

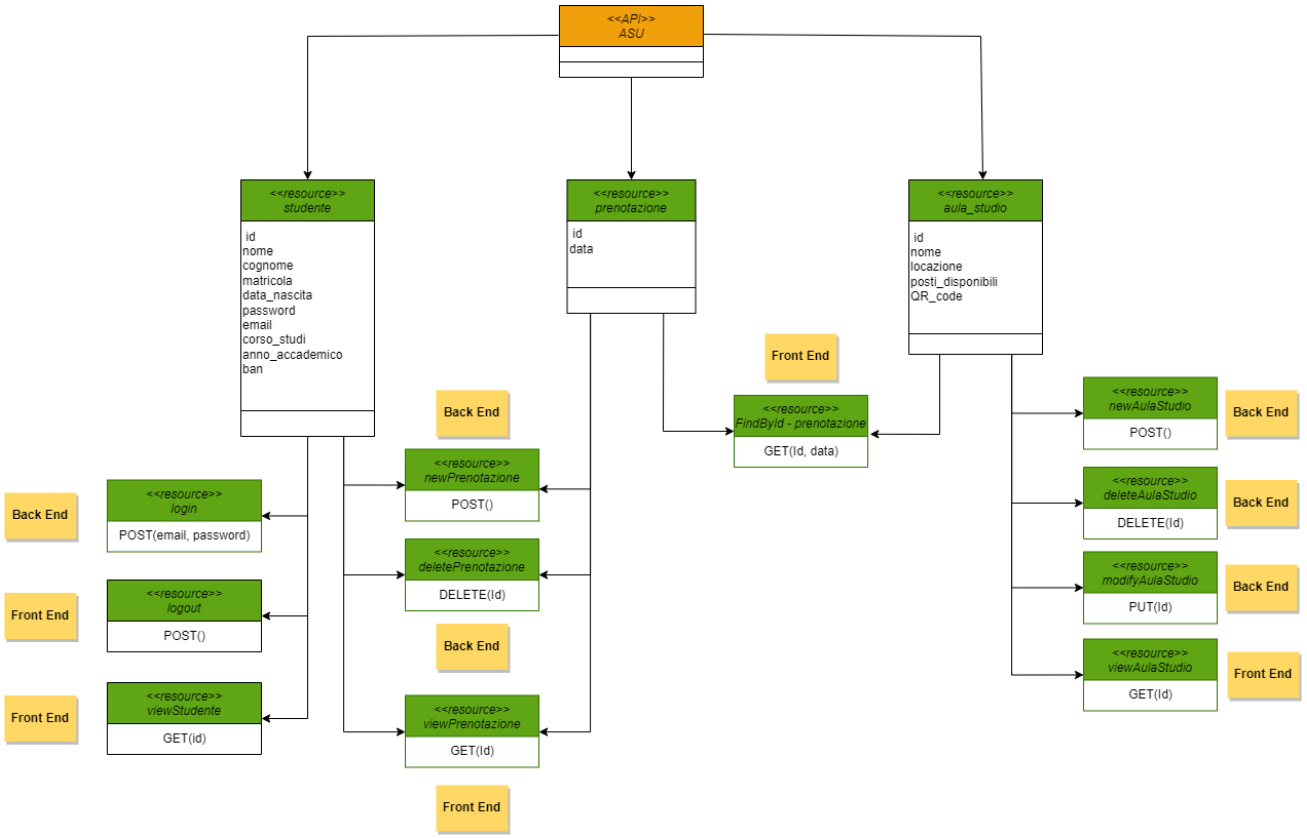
Attraverso la funzione di "***Prenotazione***" è possibile andare nella sezione delle aule studio dove vogliamo andare ad effettuare una prenotazione.
Dopo aver trovato l'aula di nostro interesse, dobbiamo andare a selezionare la data che ci interessa, infine, se ci sono posti disponibili, la prenotazione verrà effettuata.
Con la funzione del "***Profilo Utente***", è possibile visualizzare i propri dati personali che saranno estrapolati dal database dell'università e proiettati nella pagina del sito.
Da questa sezione è anche possibile effettuare il logout.
La funzionalità di "***Visualizza Prenotazioni***" permette di visualizzare le prenotazioni attualmente attive ed eventualmente di eliminarle.



4. DIAGRAMS

4.1 RESOURCES EXTRACTION

Dal class diagram, ricaviamo 3 risorse: ‘sudenti’, aula_studio’ e ‘prenotazione’.
Ogni risorsa ha i propri parametri che definiscono nello specifico la risorsa all’interno della realtà.
Da queste risorse otteniamo anche le funzionalità collegate ad esse, che vengono definite come risorse.

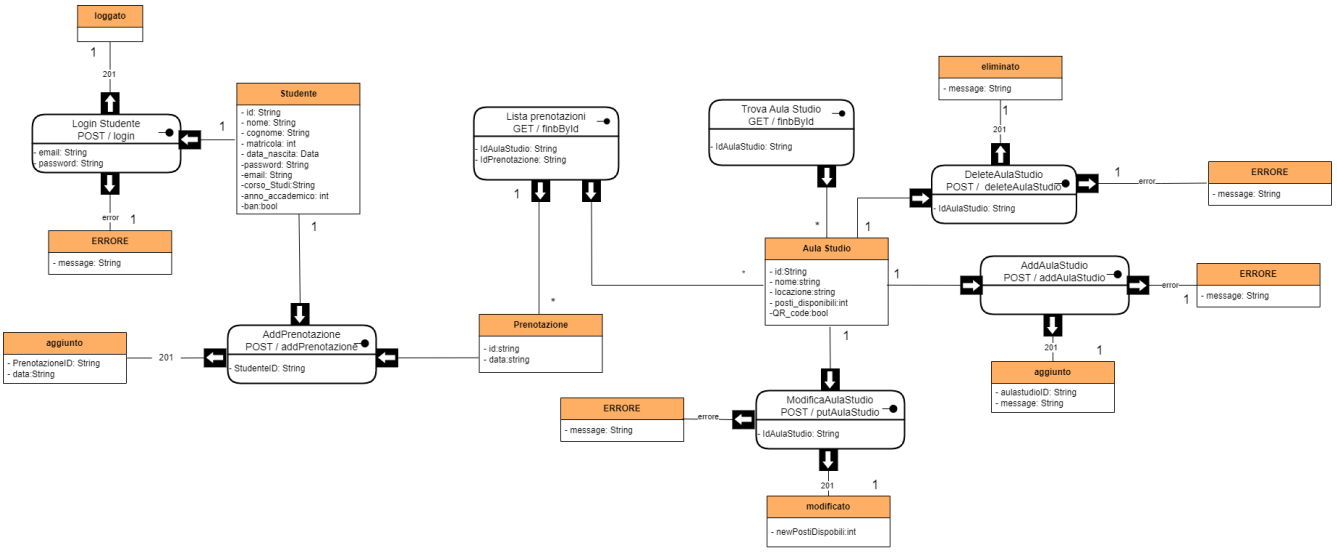


4.2 RESOURCE MODELS

Una volta estratte le risorse dal diagramma soprastante, andiamo a costruire il *Resources Model*.
Dalle 3 risorse principali estratte (‘studente’, ’prenotazione’, ’aula_studio’) creiamo le API collegate ad esse.
Ogni API rappresenta una funzionalità collegata alla risorsa.
Possiamo notare inoltre la presenza di una risorsa specifica: **ERRORE**.
Questa risorsa associata alle API rappresenta le varie situazioni di errore che potrebbero conseguirsi durante l’esecuzione dei vari metodi e funzioni.

Nel diagramma vengono rappresentati i seguenti API:

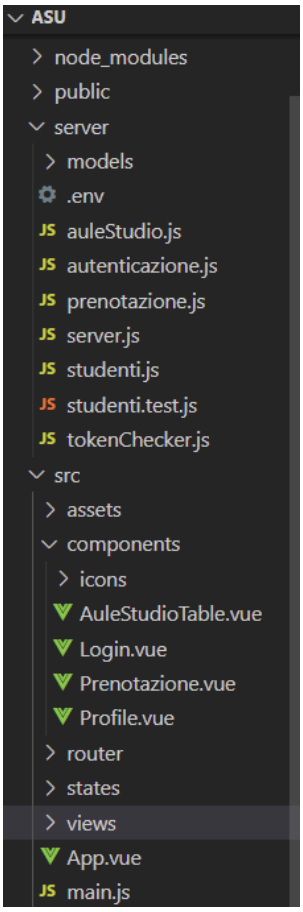
- **LOGIN STUDENTE**
- **ADD PRENOTAZIONE**
- **LISTA PRENOTAZIONI**
- **MODIFICA AULA STUDIO**
- **TROVA AULA STUDIO**
- **AGGIUNGI AULA STUDIO**
- **ELIMINA AULA STUDIO**



5. Application Implementation and Documentation

Le features definite nei precedenti capitoli verranno implementate come le principali funzionalità dal punto di vista dell'utente generale e che possono essere eseguite mentre utilizza l'applicazione web ASU.
Queste funzionalità seguono un preciso flusso definito dallo **User Flow** e comporta diversi comportamenti secondo il quale il sistema deve adattarsi a rispondere in base all'azione eseguita.
Per rispondere a queste richieste sono stati utilizzati **NodeJS** per definire le API locali con la relativa documentazione e il database MongoDB per la gestione dei dati.

4.1 Project Structure



La struttura del progetto è suddivisa in due directory principali: nella directory "**Server**" sono state definite le API del sito web e tutti i moduli utilizzati per l'implementazione, oltre ai file di test contrassegnati in arancione.
Nella directory "**src"** è stata definita l'interfaccia lato utente dell'applicazione, oltre al main e l'index.

4.2 Project Dependencies

I seguenti moduli Node sono stati utilizzati e aggiunti al file Package.Json:

- **Body-parser**
- **Cors**
- **Express**
- **MongoDB**
- **Swagger-jsdoc**
- **swagger-ui-express**
- **supertest**
- **tape**

4.3 Project Data or DB

Per la gestione dei dati necessari al funzionamento dell'applicazione abbiamo utilizzato il database MongoDB e definito le seguenti tre principali strutture dati: una collezione di "aula_studio", una collezione di "prenotazione" e infine una collezione di "studente" (che comprende anche l'account amministratore).

asu

LOGICAL DATA SIZE: 2.09KB STORAGE SIZE: 108KB INDEX SIZE: 108KB TOTAL COLLECTIONS: 3

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size
aula_studio	7	787B	113B	36KB	1	36KB
prenotazione	1	136B	136B	36KB	1	36KB
studente	6	1.19KB	203B	36KB	1	36KB

Di seguito sono riportati alcuni esempi delle istanze presenti all'interno del database:

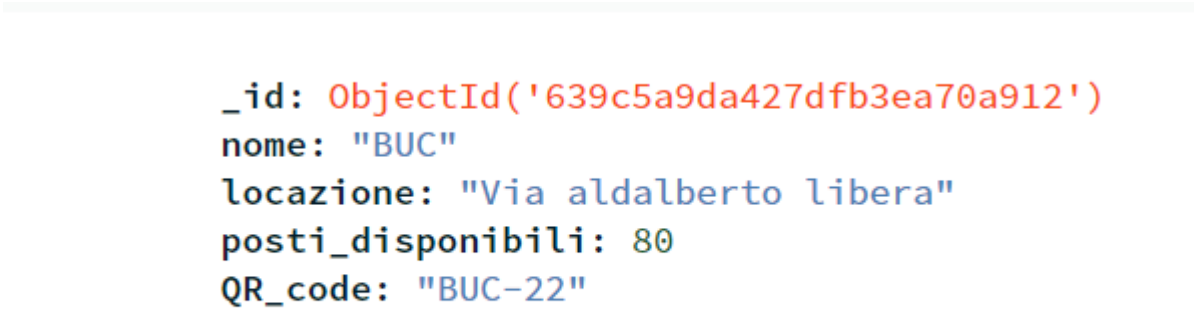


Figura 4.3.1: Istanza dell'aula studio BUC.

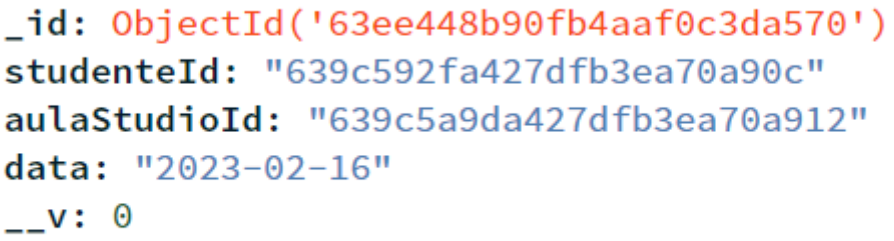


Figura 4.3.2: Istanza di una prenotazione di uno studente del 16 febbraio.

```
_id: ObjectId('639c59cfa427dfb3ea70a90e')
nome: "mattia"
cognome: "rizzoli"
matricola: 333111
data_nascita: 1999-12-31T23:00:00.000+00:00
password: "matrizz"
email: "mattia.rizzoli@studenti.unitn.it"
corso_studi: "Informatica L31"
anno_accademico: 3
ban: false
```


Figura 4.3.3: Istanza di uno studente (tipo utente).


```
_id: ObjectId('63aca60e5dc1ba567ad7602d')
email: "amm@gmail.com"
password: "1234"
amministratore: true
```


Figura 4.3.4: Istanza di un account amministratore.

4.4 Project API

Vengono elencate le varie API implementate per soddisfare i vari requisiti dell'applicazione dal punto di vista dell'utente.

4.4.1 Elenco dati utente

Questa API permette al sito di restituire i dati dell'utente in base all'*id* proposto, l'applicazione restituirà quindi tutti i campi con i dati specifici del singolo utente presenti nel database MongoDB.

```
router.get('/:id', async (req, res) => {
    let studenti = await Studenti.findById(req.params.id);
    res.send({
        self: 'studentis/' + studenti.id,
        nome: studenti.nome,
        cognome : studenti.cognome,
        email : studenti.email,
        matricola : studenti.matricola,
        data_nascita: studenti.data_nascita,
        corso_studi: studenti.corso_studi,
        password : studenti.password,
        anno_accademico: studenti.anno_accademico,
        ban : studenti.ban
    });
});
```

4.4.2 Elenco aule studio

Questa API invece restituisce semplicemente le informazioni di tutte le aule studio presenti nel database, senza stamparne una specifica. All'interno delle repository del codice è presente anche un ulteriore metodo get per poter restituire le informazioni di un'aula studio specifica tramite l'id.

```
router.get('', async (req, res) => {
    let auleStudios = await AulaStudio.find({});
    auleStudios = auleStudios.map( (auleStudios) => {
        return {
            self: 'auleStudios/' + auleStudios.id,
            nome: auleStudios.nome,
            id: auleStudios.id,
            posti_disponibili: auleStudios.posti_disponibili
        };
    });
    res.status(200).json(auleStudios);
});
```

4.4.3 Modifica aula studio

Questa API permette di andare a modificare i dati del database relativi ad un'aula studio tramite id.
Una volta ottenuto l'id, ci sarà un controllo che verificherà se l'aula studio richiesta esiste nel database, in caso di risposta positiva, si andrà ad effettuare la modifica dei parametri richiesti, altrimenti il sito restituirà un messaggio di errore.

```
router.put('/:id', async (req,res)=> {
    let aulaStudio = await AulaStudio.findById(req.params.id).exec();
    if (!aulaStudio) {
        res.status(404).send()
        console.log('AulaStudio non trovata')
        return;
    }else{
        let modifica=await AulaStudio.findByIdAndUpdate(req.params.id,{$set:{posti_disponibili:req.body.newPosti}}
        res.status(201).send()
    }
})
```

4.4.4 Crea prenotazione

Questa API permette di creare una prenotazione, tramite vari controlli, la prenotazione finale dovrà contenere una data, l'id dell'aula studio e l'id dell'utente.

Preleviamo l'id dello studente e l'id dell'aula studio, successivamente faremo un controllo per controllare se esistono nel database(anche tramite link). Se sono presenti, creeremo la prenotazione richiesta dall'utente aggiungendo anche la data.

```
router.post('', async (req, res) => {
    let studenteUrl = req.body.studente;
    let aulaStudioUrl = req.body.aulaStudio;

    if (!studenteUrl) {
        res.status(400).json({ error: 'Studente not specificato' });
        return;
    };

    if (!aulaStudioUrl) {
        res.status(400).json({ error: 'Aula studio non specificata' });
        return;
    };

    let studenteId = studenteUrl.substring(studenteUrl.lastIndexOf('/') + 1);
    let studente = null;
    try {
        studente = await Studente.findById(studenteId);
    } catch (error) {}

    if (studente == null) {
        res.status(400).json({ error: 'Studente non esistente' });
        return;
    };
```

```
    let aulaStudioId = aulaStudioUrl.substring(aulaStudioUrl.lastIndexOf('/') + 1);
    let aulaStudio = null;
    try {
        aulaStudio = await AulaStudio.findById(aulaStudioId).exec();
    } catch (error) {}

    if (aulaStudio == null) {
        res.status(400).json({ error: 'Aula studio non esistente' });
        return;
    };

    let prenotazione = new Prenotazione({
        data : req.body.data,
        studenteId: studenteId,
        aulaStudioId: aulaStudioId,
    });

    prenotazione = await prenotazione.save();

    let prenotazioneId = prenotazione.id;

    res.location("prenotazione/" + prenotazioneId).status(201).send();
});
```

4.4.5 Elimina prenotazione

Questa API permette di eliminare una prenotazione tramite id, quindi, dopo un iniziale controllo per verificare se la prenotazione esiste nel database, il sistema andrà a rimuovere l'istanza scelta.

```
router.delete('/:id', async (req, res) => {
    let prenotazione = await Prenotazione.findById(req.params.id).exec();
    if (!prenotazione) {
        res.status(404).send()
        console.log('Prenotazione non trovata')
        return;
    }
    await prenotazione.deleteOne()
    console.log('Prenotazione rimossa')
    res.status(204).send()
});
```

4.4.6 Elenco prenotazioni

Questa API permette di restituire un elenco di prenotazioni di una stessa aula e di una stessa data. Questa funzione viene utilizzata per calcolare i posti rimanenti per prenotare.
Se il numero di posti prenotati supera quello dei posti disponibili, allora la prenotazione non potrà essere effettuata.

```
router.get('/:id/:data', async (req, res) => {
    let prenotazioni = await Prenotazione.find({aulaStudioId : req.params.id, data : req.params.data});
    prenotazioni = prenotazioni.map((dbEntry) => {
        return {
            self: 'prenotazione/' + dbEntry.id,
            data : dbEntry.data,
            studente: 'studentis/' + dbEntry.studenteId,
            aulaStudio: 'auleStudios/' + dbEntry.aulaStudioId
        };
    });
    let posti2=Object.keys(prenotazioni).length
    res.send({posti:posti2});
});
```

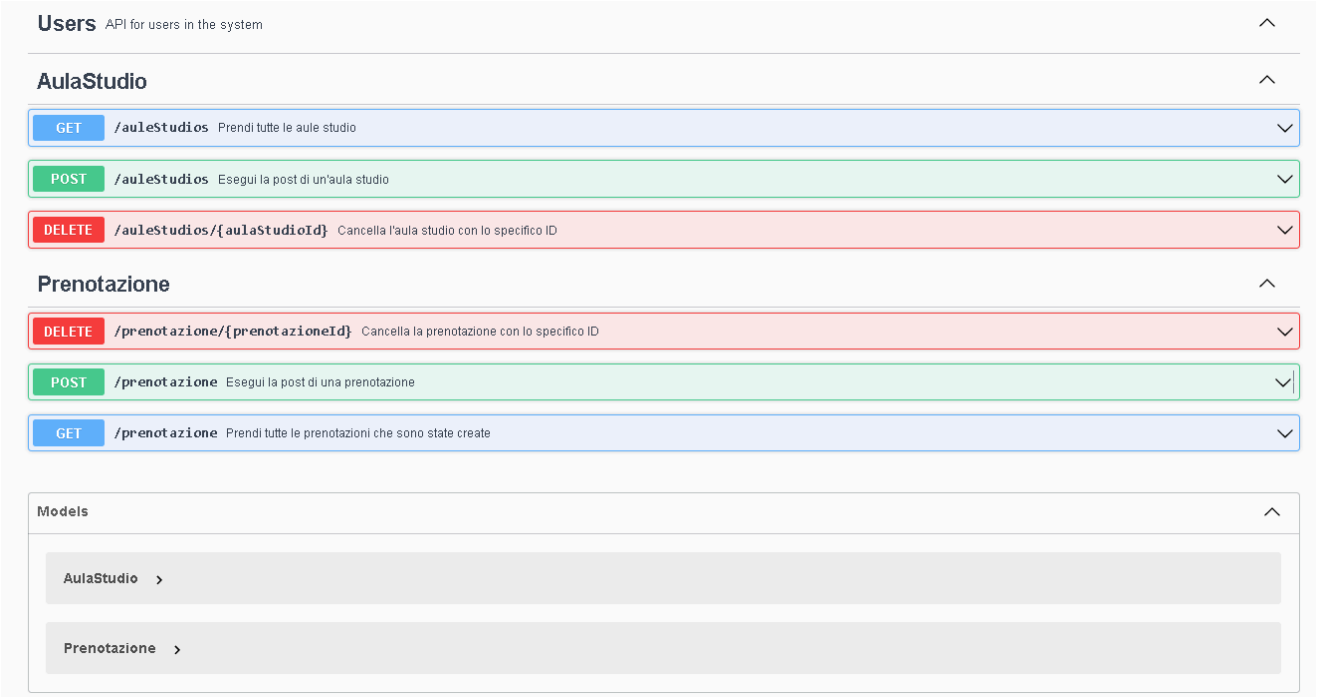
5. API Documentation

Le API locali sviluppate per l'applicazione web ASU sono state documentate e commentate attraverso una libreria di NodeJS chiamata **Swagger-UI-Express**. Attraverso lo strumento di JSDoc è stato generato un documento a partire dal codice sorgente delle API stesse.

È possibile consultare la documentazione delle API attraverso un URL specifico, al quale troviamo la pagina HTML generata da Swagger.
In questa pagina è possibile consultare le API di GET, POST e DELETE per la gestione delle richieste, degli invii e delle eliminazioni dei dati.

L'Endpoint da invocare per raggiungere la seguente documentazione è:

http://localhost:8080/api-docs/



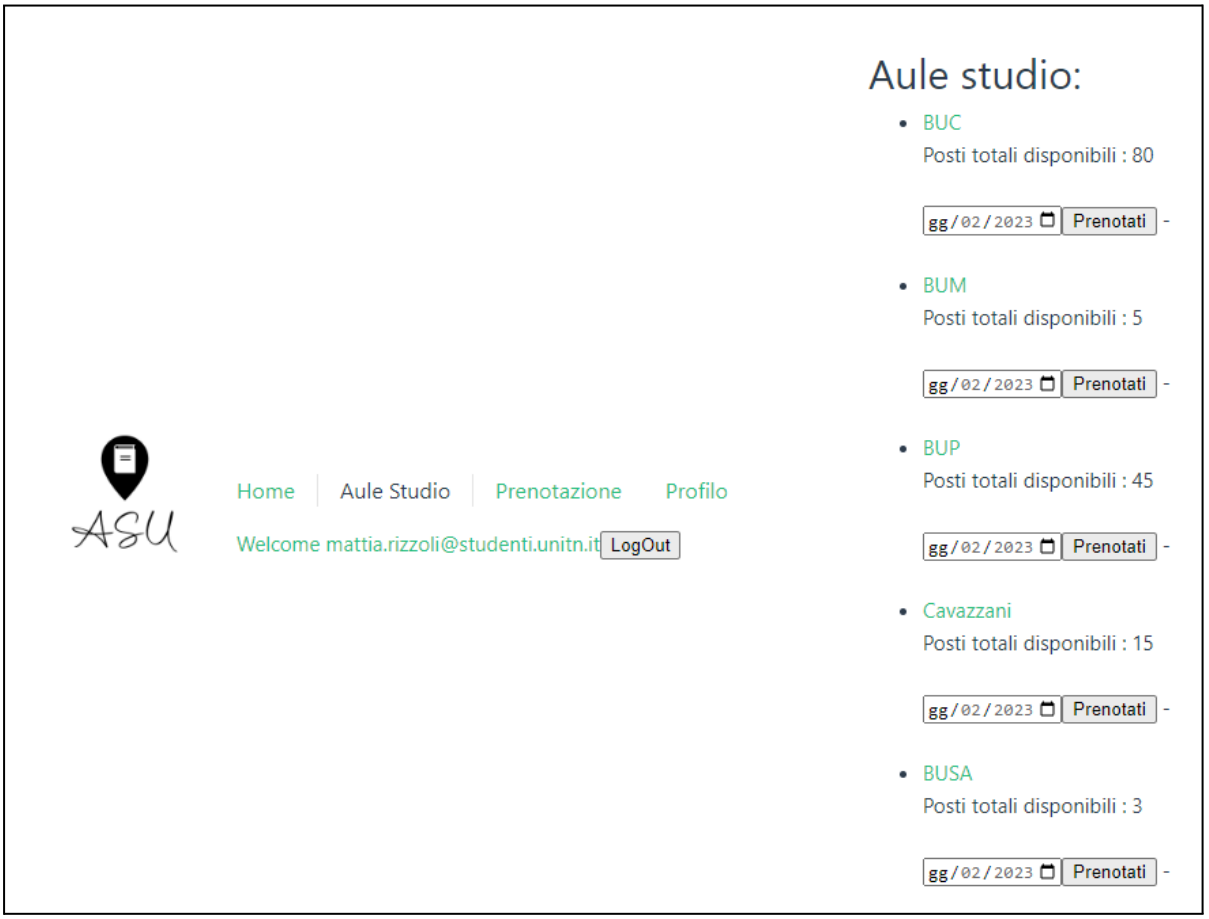
6. FrontEnd Implementation

Attraverso il Front-End è possibile visualizzare e avere un esempio dell'utilizzo delle features implementate per l'utente generale che utilizzerà l'applicazione web ASU.
Questo permette di avere un feedback anche dal cliente stesso, determinando se il prodotto sviluppato rispetta i vincoli definiti nei documenti precedenti.

Le pagine di Front-End forniscono le funzionalità di visualizzazione delle aule studio (con prenotazione), di visualizzazione dei dati del profilo utente ed infine di gestione delle prenotazioni attive.

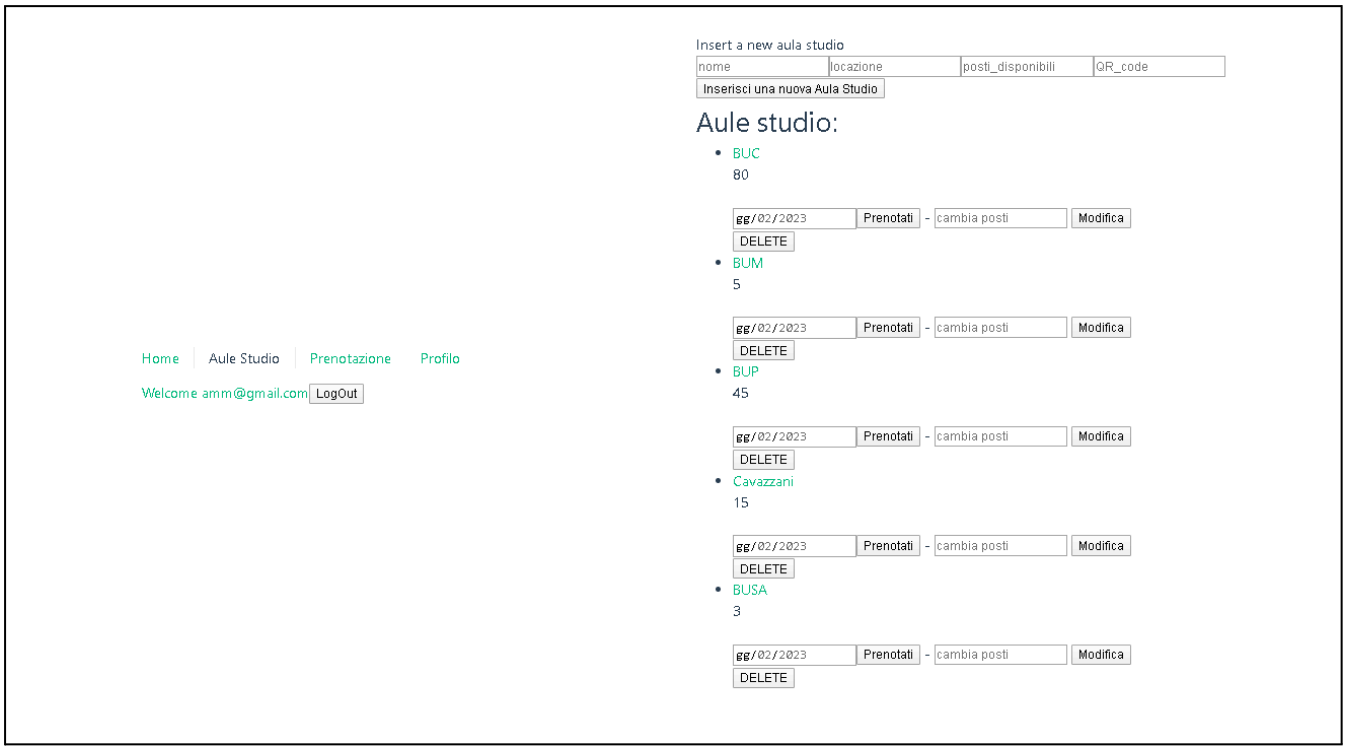
6.1 Visualizzazioni aule studio (anche amministratore)

La pagina di visualizzazione delle aule studio permette di visualizzare tutte le aule studio che aderiscono al progetto con tutte le loro relative informazioni.
Per gli utenti loggati sarà possibile effettuare una prenotazione (dopo aver scelto una data dal calendario) tramite il bottone "prenota".



Se clicchiamo su una specifica aula studio, ci apparirà un'altra pagina dove saranno riportate tutte le informazioni relative alla singola aula studio.

Se siamo loggati con l'account amministratore,, la pagina delle aule studio sarà diversa, ci saranno infatti dei campi e bottoni aggiuntivi che ci permetteranno rispettivamente di creare, modificare ed eliminare un aula studio.



6.2 Visualizzazioni profilo utente

Cliccando su “profilo” da loggati, si aprirà una pagina dove sarà possibile visualizzare tutte le informazioni relative all’account universitario dello studente.

6.2 Gestione prenotazioni attive

Cliccando su “prenotazioni” si aprirà una pagina con l’elenco di tutte le prenotazioni attive dello studente con le relative date e aula scelte.
Sarà inoltre possibile decidere di eliminare delle prenotazioni tramite l’apposito bottone “elimina” collocato alla destra di ogni prenotazione.

7. GitHub Repository Info

Il progetto ASU è disponibile al seguente link:

https://github.com/ASU-2022

Per eseguire il progetto è sufficiente lanciare dal terminale il comando “node server.js” nella cartella “**SERVER**”.
Successivamente, per quanto riguarda il lato client bisognerà inserire il comando “npm run dev” dal terminale dell’**utente**.

8. Testing

Per eseguire il testing è stato definito il file “**studenti.test.js**” nel quale è stato implementato lo script “*index.js*” necessario per effettuare il testing.

Vengono eseguiti i test di correttezza di due APIs:

- **Post** che permette allo studente di loggarsi
- **Get** che restituisce le informazioni di uno studente

Seguono i frammenti di codice dedicati al testing:

POST:

Segue il codice per l’autenticazione dell’utente nel sito, abbiamo utilizzato vari controlli per poter verificare innanzitutto che l’utente esiste nel database, per poi verificare che la password inserita sia coerente a quella nel database.
Infine, se tutto è corretto, si creerà un token con salvate le informazioni dello studente loggato.

```
router.post('', async function(req, res) {

    // find the user
    let studente = await Studenti.findOne({
        email: req.body.email
    }).exec();

    // user not found
    if (!studente) {
        res.json({ success: false, message: 'studente non trovato' });
        console.log("studente non trovato");
    }
    else if (studente.password != req.body.password) {
        res.json({ success: false, message: 'Password Sbagliata'});
        console.log("passoword sbagliata");
    }
    else{
        var payload = {
            email: studente.email,
            id: studente.id
        }
        var options = {
            expiresIn: 86400 // expires in 24 hours
        }
        var token = jwt.sign(payload, process.env.SUPER_SECRET, options);

        res.json({
            success : true,
            message: 'Token!',
            token: token,
            email: studente.email,
            id: studente.id,
            amministratore: studente.amministratore,
            self: "/"+ studente.id
        });
```

GET:

Questo codice, già mostrato in precedenza, rappresenta la restituzione delle informazioni di uno studente in base all'id.

```
router.get('/:id', async (req, res) => {
    let studenti = await Studenti.findById(req.params.id);
    res.send({
        self: 'studentis/' + studenti.id,
        nome: studenti.nome,
        cognome : studenti.cognome,
        email : studenti.email,
        matricola : studenti.matricola,
        data_nascita: studenti.data_nascita,
        corso_studi: studenti.corso_studi,
        password : studenti.password,
        anno_accademico: studenti.anno_accademico,
        ban : studenti.ban
    });
});
```

È possibile lanciare il testing posizionandosi con il terminale nella cartella "**ASU**" e lanciare il comando "**npm run test**".

Tale azione è possibile in quanto è stata applicata un'estensione al file di configurazione package.json.

Dopo aver eseguito il comando, se tutti i test andranno a buon fine, ci ritroveremo con una situazione finale di questo tipo.

```
 PASS  server/studenti.test.js
  GET /studentis/me
    √ GET studentis/me with no token should return 401 (21 ms)
    √ GET studentis/me?token=<invalid> should return 403 (8 ms)
    √ GET studentis/me?token=<valid> should return 200 (4 ms)
    √ GET studentis/639c592fa427dfb3ea70a90c should respond with "informazioni studente"  (4 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        1.161 s
Ran all test suites.
```