Esempio di documentazione

Titolo del progetto: QR Code generator
Alunno/a: Riley Bianchi
Classe: Info I3AC
Anno scolastico: 2025/2026
Docente responsabile: Ingrid Cereda

Introduzione

1.1 Informazioni sul progetto

Riley Bianchi I3AC
SAMT Trevano Informatica M306
12.09.2025 – 19.12.2025

1.2 Scopo

Lo scopo del progetto e di creare una pagina web che permette di visualizzare e creare codici QR

2 Analisi

2.1 Analisi del dominio

Il progetto nasce dall'esigenza di creare uno strumento semplice e accessibile per la creazione e la condivisione di codici QR. Attualmente esistono diversi generatori online, ma spesso risultano limitati, poco intuitivi o privi di funzionalità per la gestione dei contenuti e della loro visibilità.

Il prodotto sarà un sito web, accessibile sia da desktop che da mobile, che permetterà agli utenti registrati di generare codici QR a partire da testi, link, immagini e altri contenuti, scegliendo se renderli pubblici o privati. I QR pubblici saranno raccolti in una griglia visiva nella home, con indicazione del nome e dell'autore.

Gli utenti attesi hanno competenze informatiche di base e necessitano principalmente di:

- una generazione rapida e intuitiva di QR;
- possibilità di organizzarne la visibilità;
- funzioni di visualizzazione e stampa.

Il sistema si appoggia a standard come l'**ISO/IEC 18004** per i QR Code e segue convenzioni tipiche delle web app moderne (registrazione, login, interfaccia responsive). Per lo sviluppo sono richieste conoscenze in tecnologie web, gestione database e librerie per la generazione dei QR.

2.2 Analisi e specifica dei requisiti

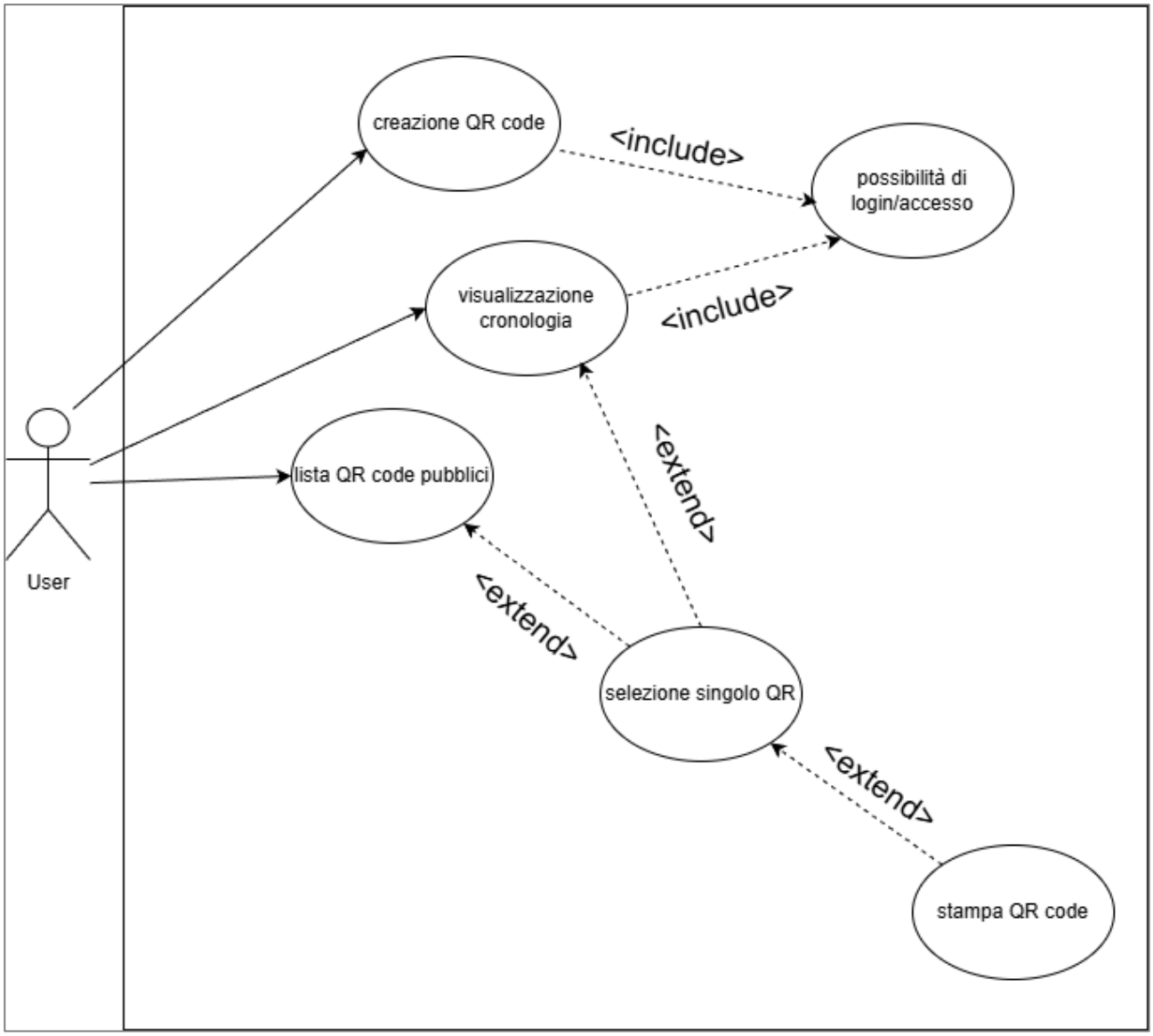
ID: REQ-01	
Nome	Creazione home
Priorità	1
Versione	1.0
Note	Pagina iniziale che visualizza i codici QR generati dagli utenti / richiede un database
ID: REQ-02	
Nome	Selezione singolo QR code
Priorità	1
Versione	1.0
Note	Possibilità di selezionare un QR code singolarmente
ID: REQ-03	
Nome	Stampa QR code
Priorità	2
Versione	1.0

Note	Possibilità di stampare il QR code senza stampare tutta la pagina
ID: REQ-04	
Nome	Login/accesso al sito
Priorità	1
Versione	1.0
Note	Possibilità di eseguire l'accesso al sito per sbloccare delle nuove funzionalità
ID: REQ-05	
Nome	Creazione pagina creazione QR code
Priorità	1
Versione	1.0
Note	Pagina per creare i QR code
Sotto requisiti	
Sreq1	Avere eseguito login / accesso
ID: REQ-06	
Nome	Creazione cronologia utente
Priorità	1
Versione	1.0
Note	Pagina che permette all'utente di visualizzare la cronologia dei suoi QR code creati / deve esserci la selezione singola come per la home
Sotto requisiti	
Sreq1	Avere eseguito login / accesso

2.2.1 Spiegazione elementi tabella dei requisiti:

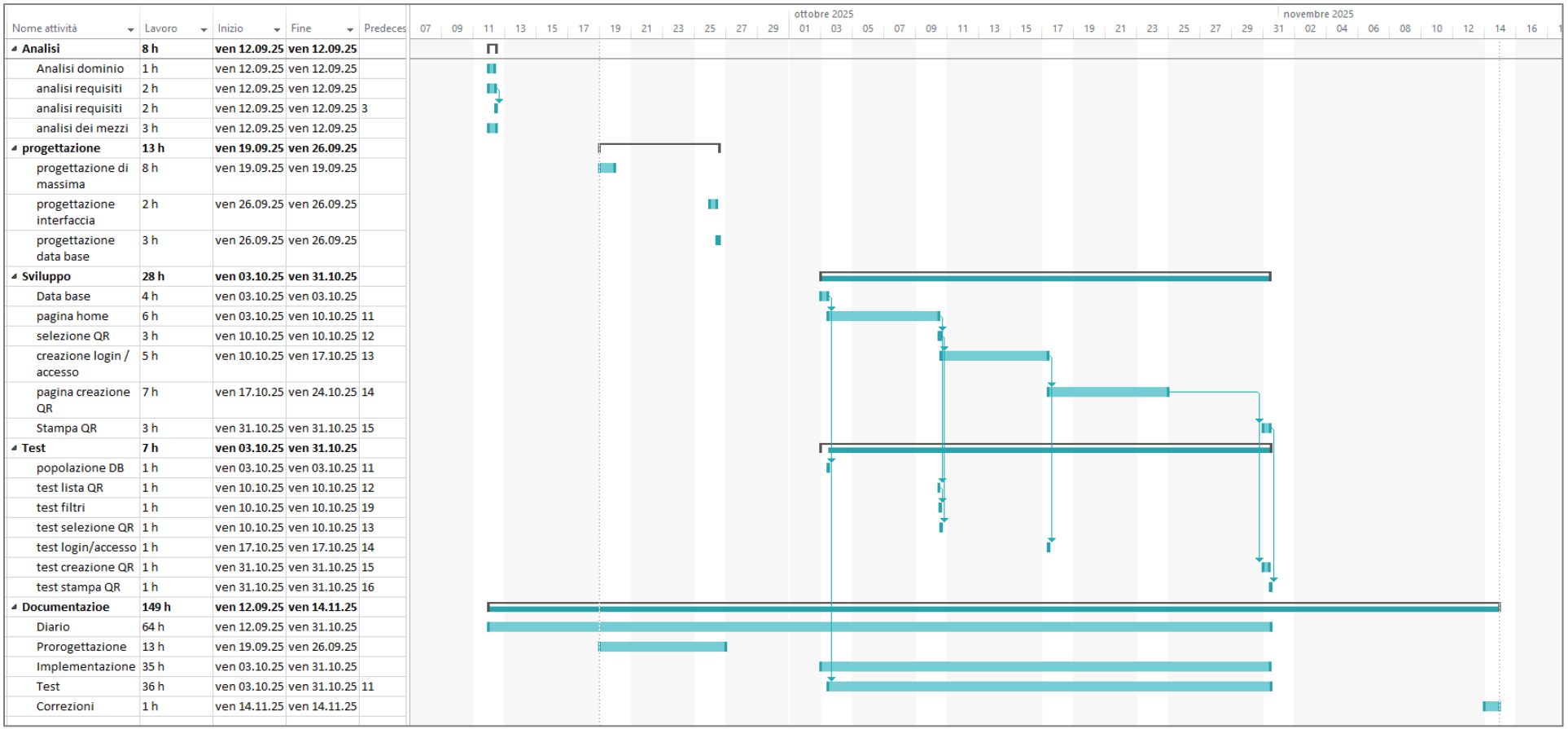
ID: identificativo univoco del requisito
Nome: breve descrizione del requisito
Priorità: indica l'importanza di un requisito nell'insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.
Versione: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata. Sulla documentazione apparirà solamente l'ultima versione, mentre le vecchie dovranno essere inserite nei diari.
Note: eventuali osservazioni importanti o riferimenti ad altri requisiti.
Sotto requisiti: elementi che compongono il requisito.

2.3 Use case


creazione QR code
<include>
possibilità di login/accesso
visualizzazione cronologia
<include>
<extend>
lista QR code pubblici
User
<extend>
selezione singolo QR
<extend>
stampa QR code

2.4 Pianificazione

Nome attività	Lavoro	Inizio	Fine	Predeces
Analisi	**8 h**	**ven 12.09.25**	**ven 12.09.25**	
Analisi dominio	1 h	ven 12.09.25	ven 12.09.25	
analisi requisiti	2 h	ven 12.09.25	ven 12.09.25	
analisi requisiti	2 h	ven 12.09.25	ven 12.09.25	3
analisi dei mezzi	3 h	ven 12.09.25	ven 12.09.25	
progettazione	**13 h**	**ven 19.09.25**	**ven 26.09.25**	
progettazione di massima	8 h	ven 19.09.25	ven 19.09.25	
progettazione interfaccia	2 h	ven 26.09.25	ven 26.09.25	
progettazione data base	3 h	ven 26.09.25	ven 26.09.25	
Sviluppo	**28 h**	**ven 03.10.25**	**ven 31.10.25**	
Data base	4 h	ven 03.10.25	ven 03.10.25	
pagina home	6 h	ven 03.10.25	ven 10.10.25	11
selezione QR	3 h	ven 10.10.25	ven 10.10.25	12
creazione login / accesso	5 h	ven 10.10.25	ven 17.10.25	13
pagina creazione QR	7 h	ven 17.10.25	ven 24.10.25	14
Stampa QR	3 h	ven 31.10.25	ven 31.10.25	15
Test	**7 h**	**ven 03.10.25**	**ven 31.10.25**	
popolazione DB	1 h	ven 03.10.25	ven 03.10.25	11
test lista QR	1 h	ven 10.10.25	ven 10.10.25	12
test filtri	1 h	ven 10.10.25	ven 10.10.25	19
test selezione QR	1 h	ven 10.10.25	ven 10.10.25	13
test login/accesso	1 h	ven 17.10.25	ven 17.10.25	14
test creazione QR	1 h	ven 31.10.25	ven 31.10.25	15
test stampa QR	1 h	ven 31.10.25	ven 31.10.25	16
Documentazioe	**149 h**	**ven 12.09.25**	**ven 14.11.25**	
Diario	64 h	ven 12.09.25	ven 31.10.25	
Prorogettazione	13 h	ven 19.09.25	ven 26.09.25	
Implementazione	35 h	ven 03.10.25	ven 31.10.25	
Test	36 h	ven 03.10.25	ven 31.10.25	11
Correzioni	1 h	ven 14.11.25	ven 14.11.25	



Titolo del progetto: QR Code generator
Alunno/a: Riley Bianchi
Classe: Info I3AC
Anno scolastico: 2025/2026
Docente responsabile: Ingrid Cereda

2.5 Analisi dei mezzi

Per realizzare il progetto è stato utilizzato un PC scolastico con Visual Studio Code usando Node JS, per quello che riguarda il database lavorando con Node ho dovuto utilizzare MongoDB

2.5.1 Software

Visual Studio Code
Node JS
Mongo DB Community

2.5.2 Hardware

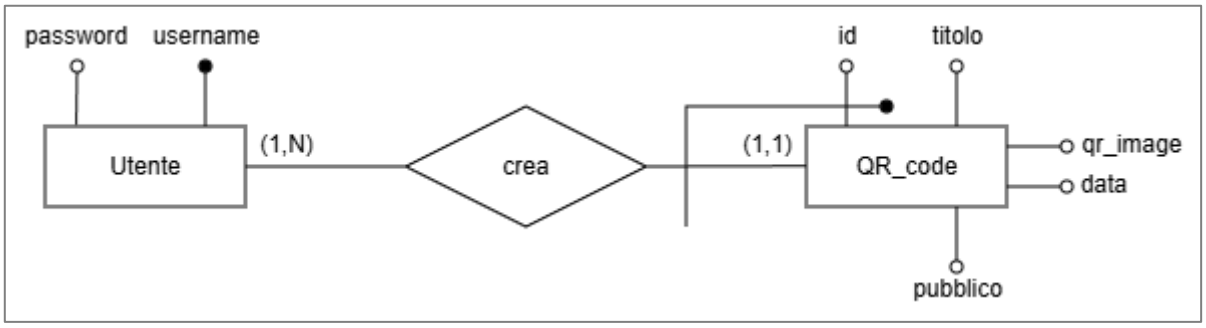
Il progetto dovrà essere eseguito su vari browser, di conseguenza gli hardware utilizzabili variano dal pc fino allo smartphone, l'importante e che possieda una parte grafica con annesso browser.
Per lo sviluppo utilizzerò solo un pc.

HW PC:

- Marca: DELL
- OS: Windows 11
- CPU: 13th Gen Intel® Core™ i7-13700
- GPU: NVIDIA T400 4GB
- Memoria: 32GB RAM DDR4

3 Progettazione

3.1 Design dei dati e database



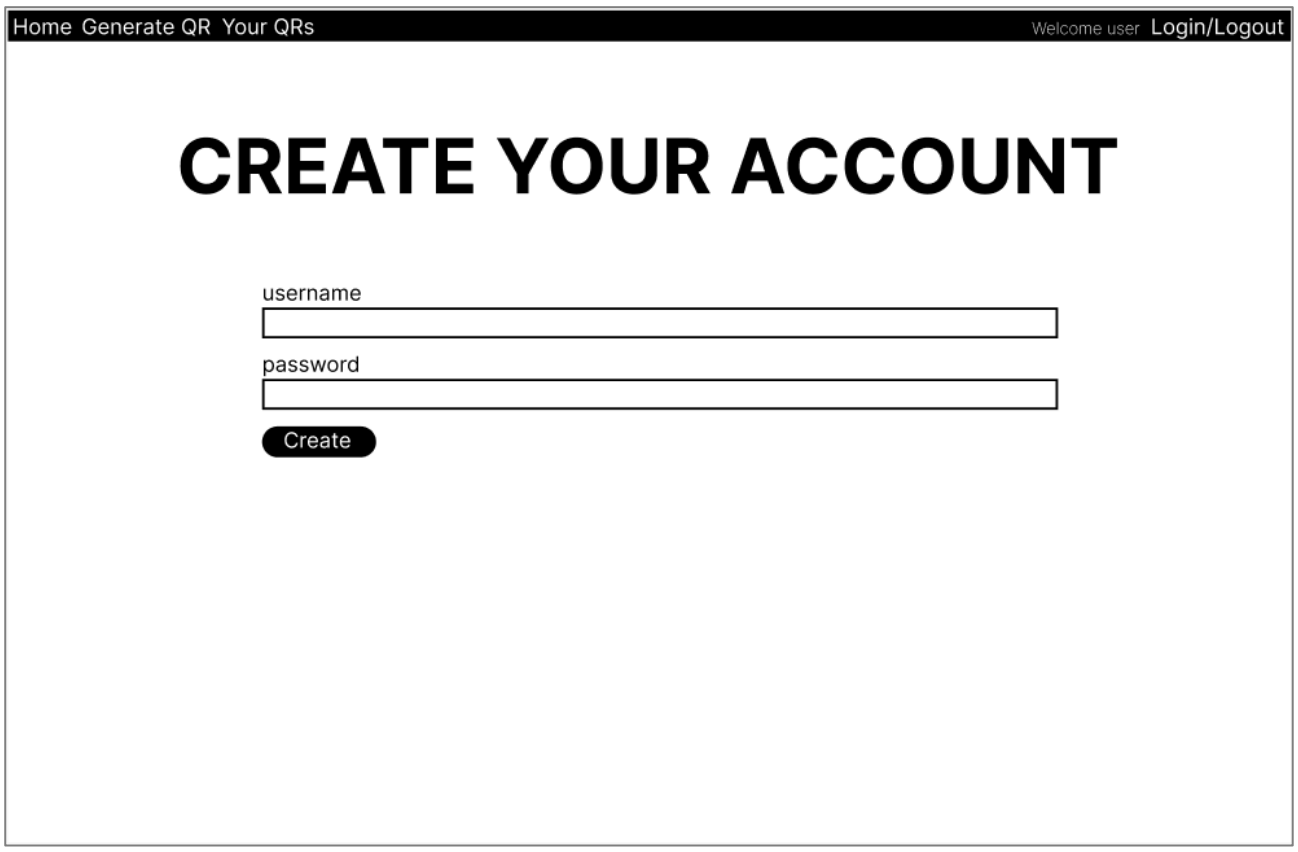
3.2 Design delle interfacce

3.2.1 Home

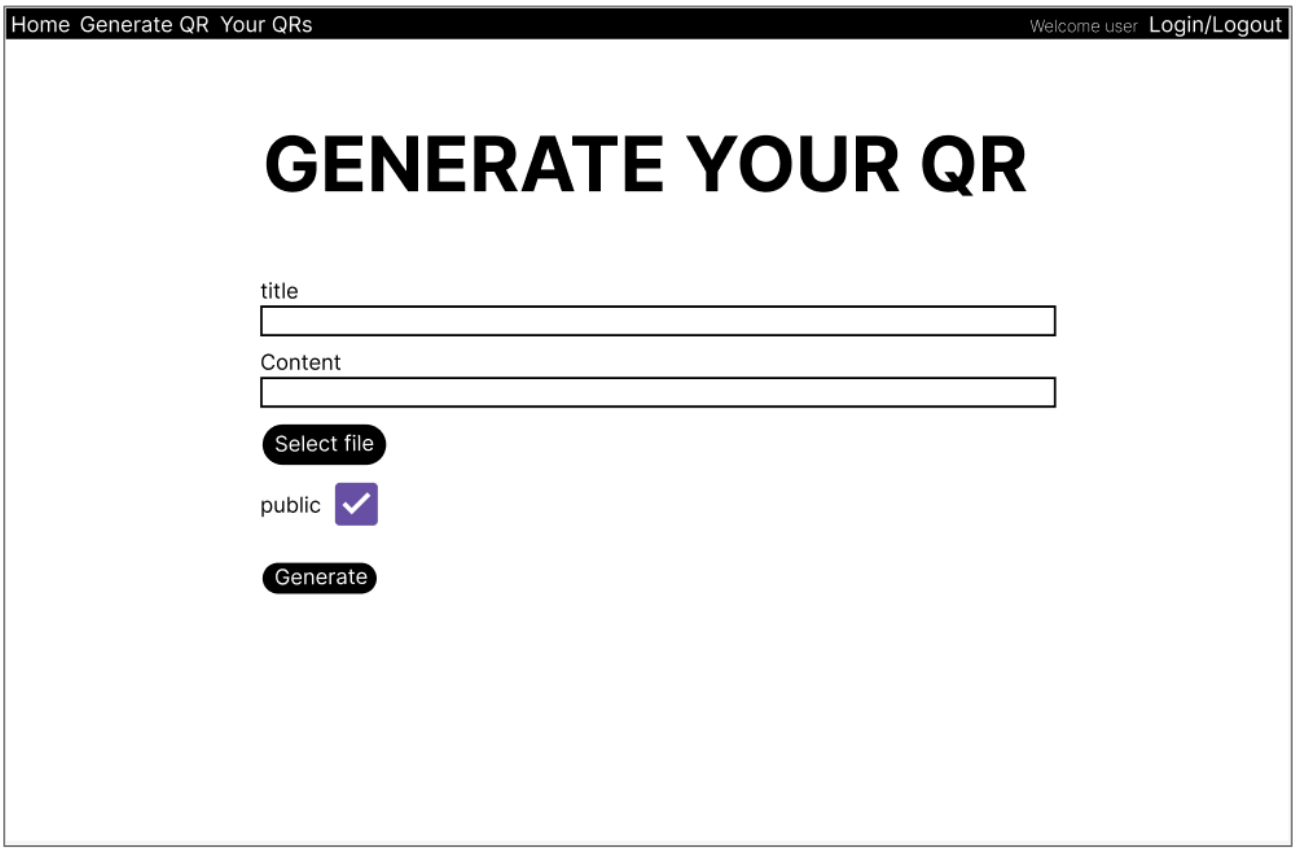
3.2.2 Login


Home Generate QR Your QRs
Welcome user Login/Logout
LOGIN
username
password
Login
You don't have an account, create one

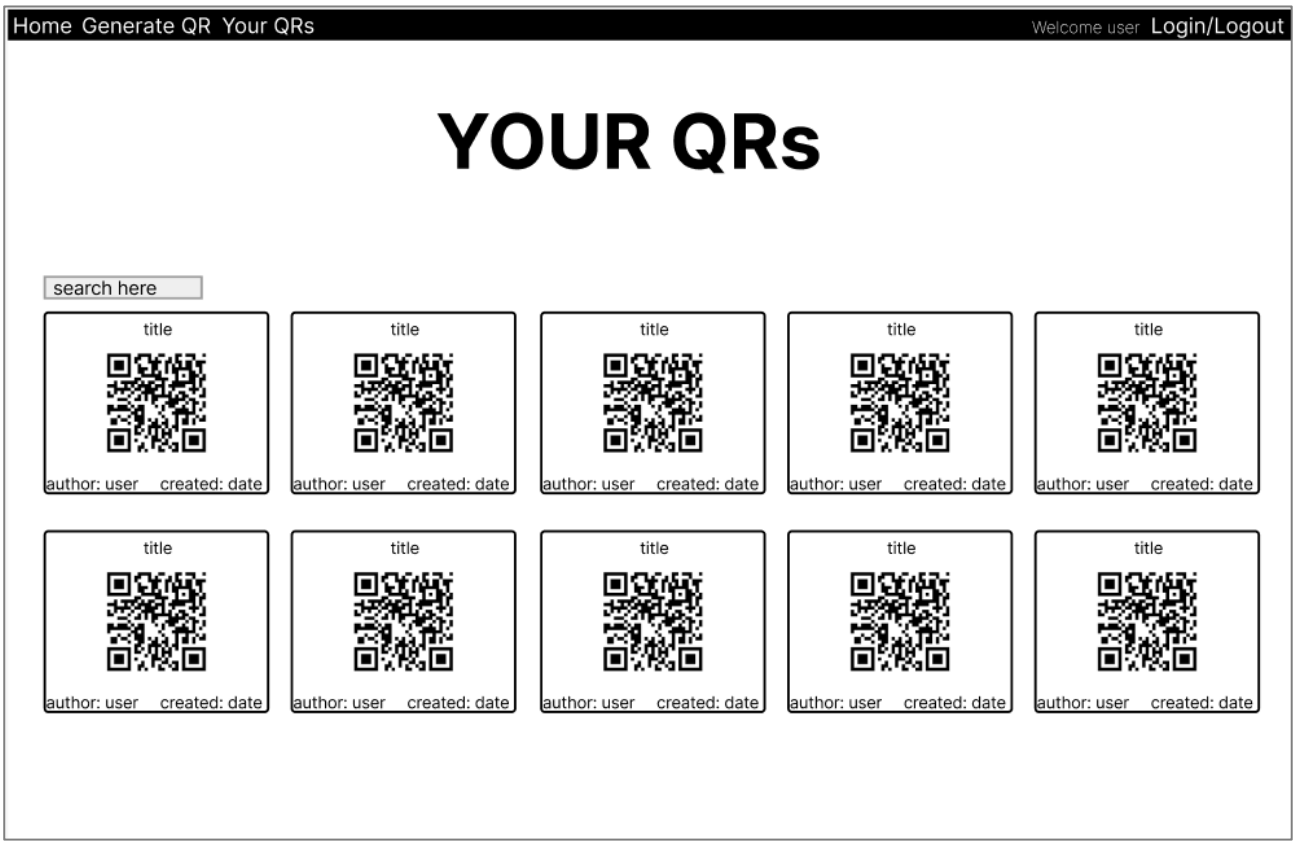
3.2.3 Create account



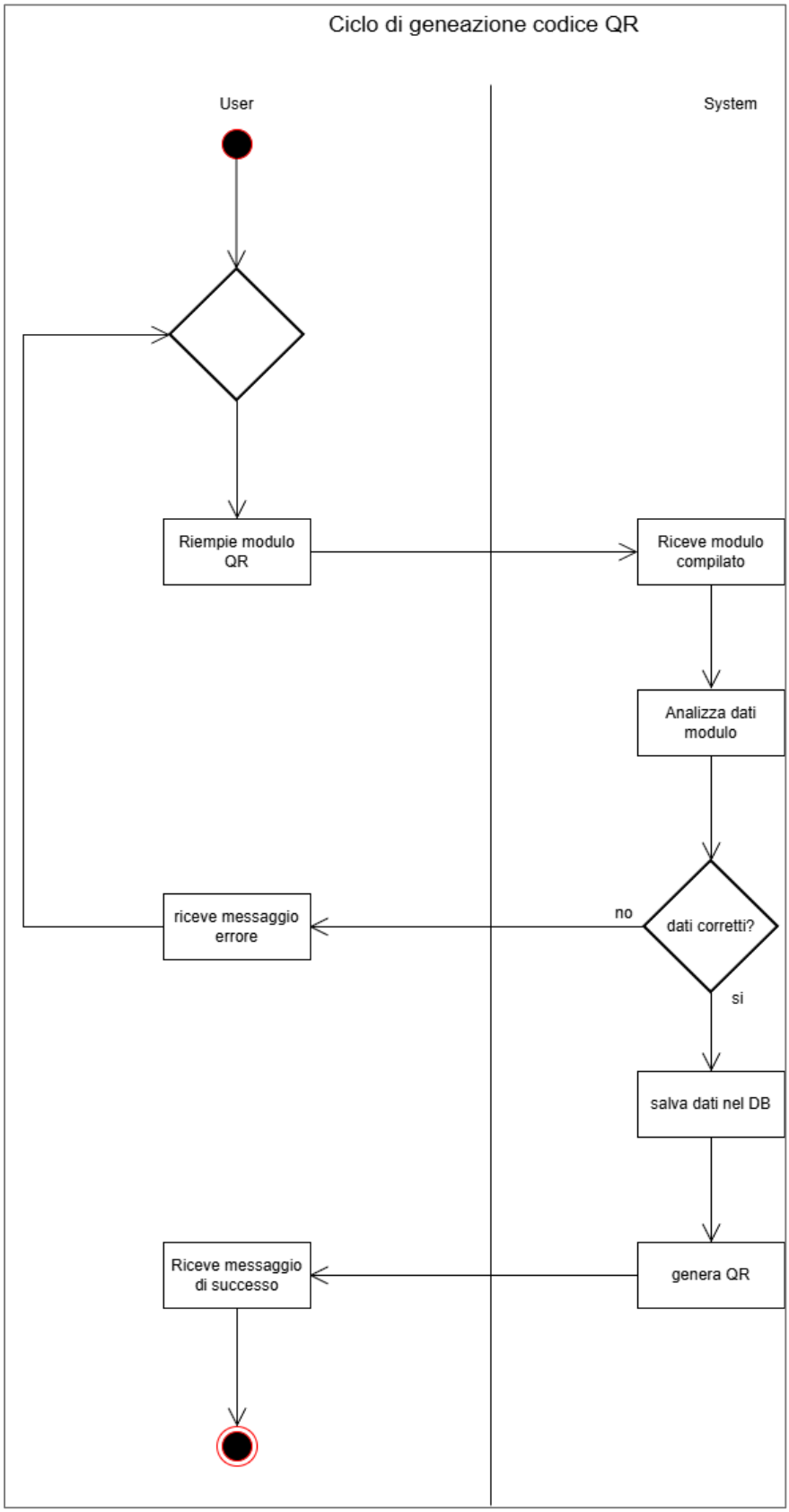
3.2.4 Generate QR



3.2.5 Your QRs


Home Generate QR Your QRs
Welcome user Login/Logout
YOUR QRs
search here
title
author: user created: date

3.3 Design procedurale



4 Implementazione

4.1 Creazione applicazione

Per iniziare la creazione dell'applicazione, ho creato il file app.js. Questo file si occupa della gestione generale del progetto, avviando il server, definendo le varie route in cui vengono gestite le richieste GET e POST, creando le sessioni, fondamentali per alcune funzionalità del progetto, e definendo le cartelle o i file che devono essere sempre considerati dall'applicazione, come quelli relativi allo stile. Inoltre, si occupa dell'aspetto più importante, ovvero l'avvio della connessione al database.

4.2 Model

4.2.1 User model

Il modello User consente di salvare all'interno del database i vari utenti che creano un account. Questo modello utilizza gli attributi username e password, entrambi obbligatori; inoltre, l'attributo username deve essere unico, in modo da evitare la presenza di più utenti con lo stesso nome.

```
1   const mongoose = require("mongoose");
2
3   const userSchema = new mongoose.Schema({
4       username: {
5           type: String,
6           required: true,
7           unique: true
8       },
9       password: {
10          type: String,
11          required: true
12      }
13  });
14
15  module.exports = mongoose.model("User", userSchema);
```

4.2.2 QR_code model

Il modello dei QR consente di salvare nel database i QR generati dagli utenti. Il modello richiede i seguenti attributi: un titolo, una stringa in Base64 dell'immagine del QR (la cui generazione verrà spiegata più avanti), una data con valore di default al momento della creazione, l'utente che viene recuperato tramite le sessioni e un attributo pubblico di tipo booleano, che indica se l'utente desidera rendere il QR visibile ad altri utenti facendolo apparire nella home. Tutti questi attributi sono obbligatori, ad eccezione della data.

```
const mongoose = require("mongoose");

const qrCodeSchema = new mongoose.Schema({
    titolo: {
      type: String,
      required: true
    },
    qr_image: {
      type: String,
      required: true
    },
    data: {
      type: Date,
      default: Date.now
    },
    utente: {
      type: String,
      required: true
    },
    pubblico: {
      type: Boolean,
      require: true
    }
  });

  module.exports = mongoose.model("QR_Code", qrCodeSchema);
```

4.3 Prtials views

Le partial views che utilizzo sono head.hbs e nav.hbs. Queste sono delle viste parziali che vengono inserite in ogni view principale, poiché contengono informazioni che devono essere sempre visibili.

```
<!DOCTYPE html>
<html lang="it">
    <head>
        {{> _head}}
    </head>
    <body>
        {{> _nav}}
        <div class="introduction">
```

La view head.hbs contiene semplicemente i classici elementi di un header HTML, come ad esempio il collegamento al file CSS e il titolo della pagina.

```
1   <meta charset="utf-8">
2   <meta name="viewport" content="width=device-width, initial-scale=1">
3   <title>{{#if title}}{{title}}{{else}}Template engine e Session{{/if}}</title>
4   <link rel="stylesheet" href="/styles.css">
```

La view nav.hbs mostra la barra di navigazione del sito. Finché l'utente non effettua il login, sono visibili tre opzioni di navigazione: Home, Generate QR e Login. Una volta effettuato l'accesso, apparirà l'opzione Your QRs, seguito da un breve messaggio con scritto "Welcome nome_utente" e l'opzione Login verrà sostituita da Log out.

4.3.1 Codice

L'unica parte di codice presente esclusivamente in nav.hbs riguarda l'utilizzo di if per nascondere alcune opzioni di navigazione. Questo è reso possibile grazie alla funzione currentUser presente nel middleware auth, che consente di verificare se l'utente ha effettuato l'accesso controllando se session e session.userId contengono dei dati. In tal caso, l'utente viene recuperato e inserito in res.locals.currentUser, che viene poi utilizzato come condizione nel blocco if.

```
<li><a href="/generate_QR">Generate QR</a></li>
{{#if currentUser}}
  <li><a href="/your_QRs">Your QRs</a></li>
{{/if}}
</ul>
<ul class="nav-right">
{{#if currentUser}}
  <li><p class="welcome">welcome {{currentUser.username}}</p></li>
  <li>
    <button onclick="Confirm()">Logout</button>
  </li>
{{else}}
  <li><a href="/login">Login</a></li>
{{/if}}
```

```
async function currentUserUser(req, res, next) {
    try{
        if (req.session && req.session.userId) {
            const user = await User.findById(req.session.userId).lean();
            res.locals.currentUser = user || null;
        } else {
            res.locals.currentUser = null;
        }
        next()
    } catch (err) {
        next(err);
    }
}
```

4.3.2 Logout

Oltre a quanto spiegato, è presente un piccolo script JavaScript che richiede una conferma prima di eseguire il logout. Se l'utente conferma, viene inviata una richiesta POST a una route che imposta la sessione a null e successivamente esegue un reindirizzamento alla home.

```
<script>
  //Alert di conferma del logout
  function Confirm(){
    if(confirm("Are u sure u want to log out") == true){
      return fetch("logout", { method: "post"})
      .then(response => {
        if (response.redirected) {
          window.location.href = response.url;
        }
      })
    }
  }
</script>
```

```
router.post("/logout", requireAuth, (req, res) => {
    req.session = null;
    res.redirect("/?msg=logged%20out");
})
```

Come si può notare dalle immagini, il reindirizzamento non viene eseguito direttamente dal POST, come accade normalmente, ma essendo la richiesta inviata tramite JavaScript, il redirect non avviene automaticamente. Il logout, invece, viene comunque eseguito; per questo motivo utilizzo un return per poi eseguire il reindirizzamento tramite JavaScript sfruttando la risposta.
Inoltre, nel router.post per il logout è presente un'altra funzione del middleware auth, chiamata requireAuth. Questa funzione verifica, in modo simile a currentUser, se l'utente è loggato. Nel caso in cui la verifica risulti negativa, l'utente viene reindirizzato alla pagina di login. Questo meccanismo serve a evitare che un utente possa accedere a pagine a cui non dovrebbe avere accesso, anche conoscendone il percorso, impedendo così l'esecuzione del router stesso. La funzione requireAuth è molto ricorrente all'interno del progetto.

4.4 Home

La pagina home è la pagina iniziale che tutti gli utenti visualizzano quando accedono al sito. Al suo interno sono presenti delle card disposte a griglia, che mostrano i vari QR creati e resi pubblici da altri utenti. Ogni card contiene il titolo, il QR da scansionare, l'autore e la data di creazione.

È inoltre possibile eseguire una ricerca per filtrare i QR presenti grazie a una piccola barra di ricerca, che permette di cercare per titolo o per autore.

Oltre a questo, l'utente può cliccare su una delle card per aprirla in primo piano; questa funzionalità consente di stampare il QR selezionato.

4.4.1 Codice

4.4.1.1 Recupero e visualizzazione QR

La pagina viene presentata grazie a un router chiamato all'avvio. All'interno vengono recuperati i QR pubblici in ordine inverso, poiché i QR più recenti vengono inseriti come ultima riga, ma si desidera che vengano mostrati dal più recente al più vecchio. Successivamente, la pagina viene reindirizzata passando la lista dei QR inseriti, che viene poi richiamata nel file home.hbs, inserendo i campi nei punti corretti. Nel caso in cui non venga trovato alcun QR, appare il messaggio "No QR codes has been found".

```
router.get("/", async(req, res) => {
    try {
        const { msg } = req.query;

        //Ricerca QR pubblici
        const listQR = (await QR_Code.find({pubblico: true})).reverse();//metodc

        res.render("home", { title: "Home", msg, listQR});
    }catch (err){
        console.error("An error as occured while searching pubblic QRs");
        res.status(500).send("An error as occured while loading the home page")
    }
});
```

```
{{#if listQR.length}}
<div class="qr-container">
    {{#each listQR}}
    <div class="qr-card" onclick="apriPopup('{{titolo}}', '{{qr_image}}', '{{utente}}', '{{formatDate data}}')">
        <strong>{{titolo}}</strong><br>
        <img src="{{qr_image}}" alt="qr-code" class="qr"><br>
        <div class="qr-card-footer">
            <span>author: {{utente}}</span>
            <span>created: {{formatDate data}}</span>
        </div>
    </div>
    {{/each}}
</div>
{{else}}
<p class="not-found">No QR codes has been found</p>
{{/if}}
```

4.4.1.2 Ricerca QR

Quando l'utente effettua una ricerca di un QR, viene chiamato un router denominato search_home, che si occupa di cercare i QR pubblici in base ai criteri inseriti dall'utente. La ricerca viene eseguita tramite due espressioni regolari (regex): una ricerca che verifica se il pattern definito è presente in una stringa e, in caso affermativo, restituisce true.
Queste ricerche vengono inserite in un operatore OR, poiché si desidera effettuare la ricerca sia per titolo sia per autore. In questo modo, se l'utente cerca la lettera "a", verranno mostrati tutti i QR in cui la lettera "a" è presente nel titolo oppure nel nome dell'autore. Inoltre, la ricerca viene eseguita in modalità case insensitive grazie all'opzione options: "i".

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

        const { msg } = req.query;

        const searched = req.body.search;

        //Query con regex per cercare per titolo o untente
        const listQR = (await QR_Code.find({$or: [
            { titolo: { $regex: searched, $options: "i" } },
            { utente: { $regex: searched, $options: "i" } }
          ], pubblico: true })).reverse();

        res.render("home", { title: "Home", msg, listQR });

    } catch (err) {
        console.error("An error as occured while searching QRs")
        res.status(500).send("An error as occured while loading the home page")
    }
});
```

4.4.1.3 Selezione QR

Quando l'utente clicca su una delle card, viene avviato un codice JavaScript che apre in primo piano la card sotto forma di popup. Il codice consiste nel recuperare i campi del QR presenti nella card, passati tramite argomento, e inserirli in una card nascosta con lo stesso formato di quelle standard. In questa card vengono però aggiunti due span, utilizzati rispettivamente per stampare e per chiudere il popup. È inoltre possibile chiudere il popup cliccando al di fuori di esso. Infine, il popup viene mostrato all'utente.

```html
<div id="popup" class="popup">
    <div class="popup-content">
        <span class="print" onclick="printQR()">&#128424;</span>
        <span class="chiudi" onclick="chiudiPopup()">&times;</span>
        <h3 id="popup-titolo"></h3>
        <img src="" id="popup-qr" class="qr"></img>
        <div class="popup-footer">
            <span id="popup-autore"></span>
            <span id="popup-data"></span>
        </div>
    </div>
</div>
```

```javascript
function apriPopup(titolo, qr_image, utente,  data){
    document.getElementById("popup-titolo").textContent = titolo;
    document.getElementById("popup-qr").src = qr_image;
    document.getElementById("popup-autore").textContent = "author: " + utente;
    document.getElementById("popup-data").textContent = "created the: " + data;

    document.getElementById("popup").style.display = "flex";
}

function chiudiPopup() {
    document.getElementById("popup").style.display = "none";
}

window.onclick = function(e) {
    if (e.target.id === "popup") chiudiPopup();
}
```

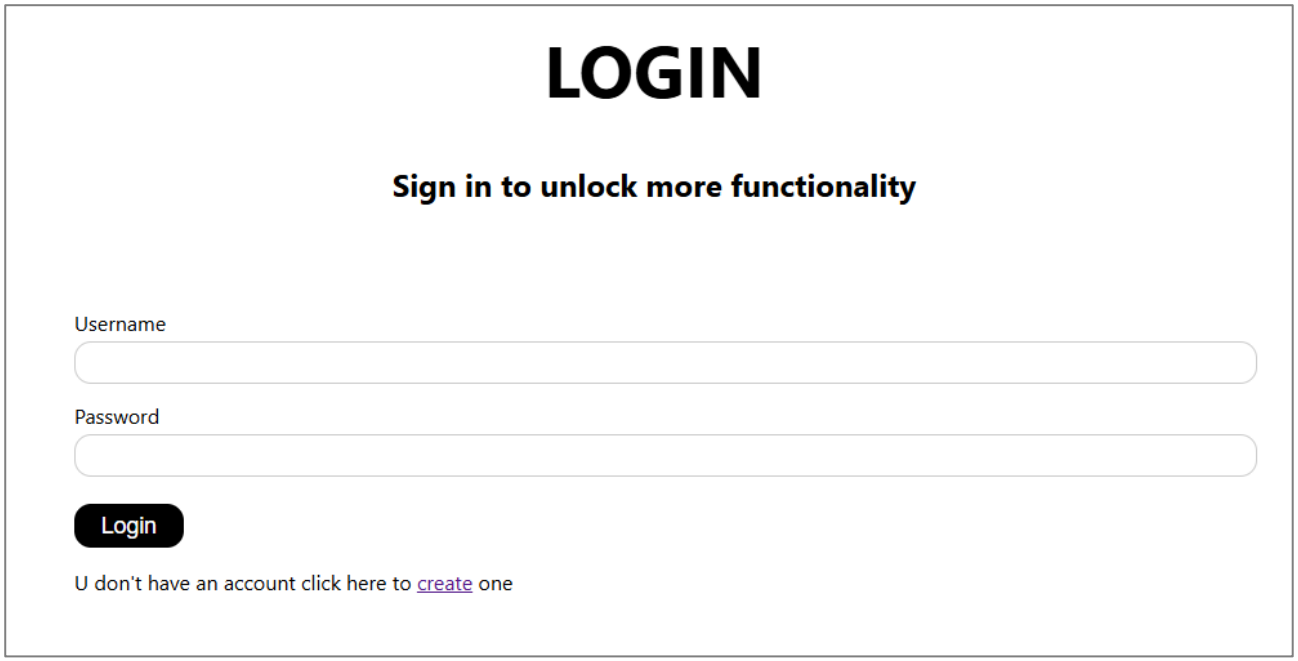
4.4.1.4 Stampa QR

Quando l'utente clicca sullo span dedicato alla stampa, viene avviata una funzione JavaScript che recupera dal popup esclusivamente il QR tramite l'attributo src. Successivamente viene creata una finestra temporanea contenente unicamente il QR selezionato, visualizzato in dimensioni maggiori. A questo punto viene immediatamente avviata la funzione di stampa di JavaScript, che apre la classica finestra di stampa. Indipendentemente dalla scelta dell'utente, la pagina temporanea viene poi chiusa automaticamente. In questo modo l'utente non visualizza mai realmente la pagina temporanea, ma soltanto la finestra di stampa.

```
w.document.write(`
    <html>
    <head>
        <title>QR Code</title>
        <style>
            /* Impostazioni di stampa */
            @page {
                size: A4 landscape;     /* A4 orizzontale */
                margin: 0;              /* senza margini */
            }

            body {
                margin: 0;
                width: 100vw;
                height: 100vh;

                display: flex;
                justify-content: center;
                align-items: center;

                background: white;
            }

            img {
                width: 70vh;           /* QR ingrandito */
                height: auto;
            }
        </style>
    </head>
    <body>
        <img src="${src}">
    </body>
    </html>
`);
```

4.5 Login

La pagina di login viene mostrata tramite un router di tipo GET quando viene selezionata dal menu di navigazione oppure quando si tenta di accedere a una pagina che richiede l'autenticazione, come già spiegato in precedenza. All'interno della pagina è possibile effettuare il login inserendo il nome utente e la password; nel caso in cui non si possieda un account, è possibile accedere alla pagina di creazione dell'account tramite un collegamento presente sotto il form da compilare.



4.5.1 Codice

4.5.1.1 Controllo login

Quando l'utente invia i dati cliccando sul bottone "Login", viene chiamato un router di tipo POST che recupera i dati inseriti dall'utente e li assegna a variabili apposite tramite req.body. Successivamente, l'utente viene cercato nel database in base al nome utente inserito e il risultato viene salvato in una variabile. Nel caso in cui il risultato sia null, viene restituito l'errore "Invalid username".

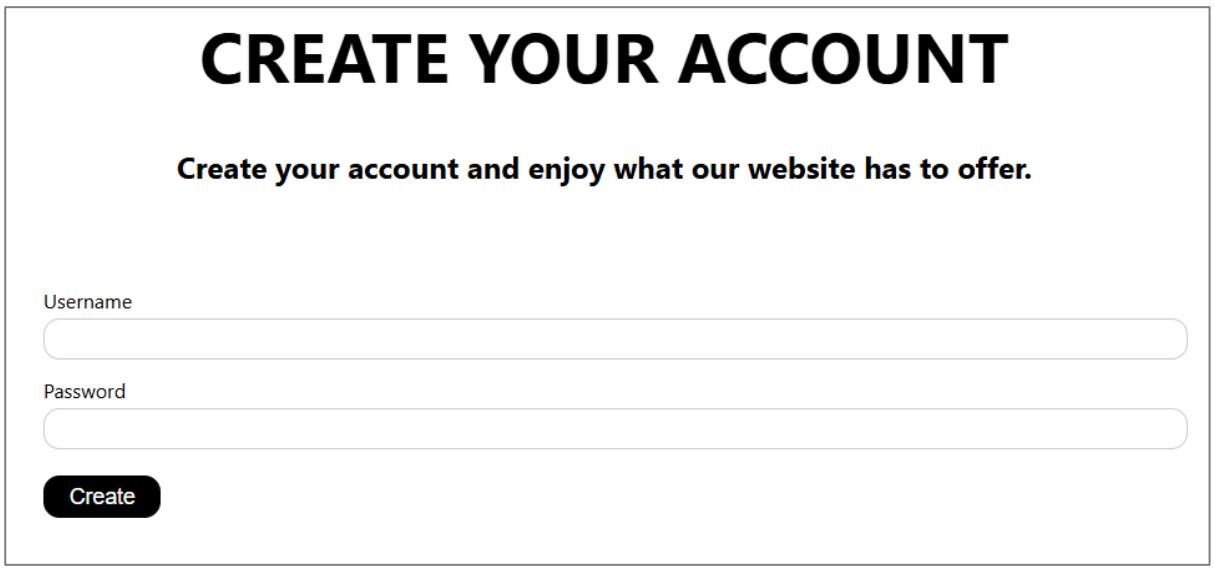
Se invece l'utente viene trovato, la password inserita viene confrontata con quella salvata nel database utilizzando bcrypt.compare(password, user.password), poiché le password vengono criptate al momento dell'inserimento nel database; il funzionamento di questo processo verrà spiegato nella sezione dedicata alla creazione dell'account. Se le password non coincidono, viene restituito il messaggio di errore "Invalid password".

Infine, se non vengono riscontrati errori, vengono impostate le sessioni per indicare che l'utente ha effettuato correttamente l'accesso e viene eseguito il reindirizzamento alla home, mostrando il messaggio "Login successful".

```
router.post("/login", async(req, res) => {
    const { username, password } = req.body;

    //Cerca l'utente nel database
    const user = await User.findOne({ username });
    if (!user) {
      return res.status(400).render("login", {title: "Login", errorU: "Invalid username"});
    }

    //Confronta la password inserita con quella hashata
    const isMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) {
      return res.status(400).render("login", {title: "Login", errorP: "Invalid password"});
    }

    req.session.userId = user.id;
    req.session.username = user.username;

    res.redirect("/?msg=Login%20succesfull");
})
```

4.6 Create account

La pagina di creazione account viene mostrata quando si clicca sul link presente nella pagina di login, precedentemente citato. Questa pagina permette di creare un account inserendo un nome utente, che non deve già esistere, e una password di almeno 8 caratteri.



4.6.1 Codice

4.6.1.1 Controllo Accesso

Quando l'utente prova a creare un account cliccando su Create, i dati del form vengono recuperati e viene eseguito un controllo nel database per verificare se l'utente esiste già. In caso positivo, viene restituito il messaggio di errore "Username not available". Altrimenti, viene verificato se la password contiene almeno 8 caratteri; se non rispetta questo requisito, viene restituito il messaggio "Password must contain at least 8 characters".

```
router.post("/create_account", async(req, res) => {
    const { username, password} = req.body;

    //Controllo se l'utente eseste già
    const userexist = await User.findOne({ username });
    if(userexist){
        return res.status(400).render("create_account",
        {title: "Create Account",
        errorU: "username not available"});
    }

    //Controllo se la password è almeno di 8 cartteri
    if(password.length < 8){
        return res.status(400).render("create_account",
        {title: "Create Account",
        errorP: "password must contain at least 8 characters"});
    }
```

4.6.1.2 Criptazione password ed inserimento utente nel DB

Una volta superati i controlli precedenti, si procede a criptare la password dell'utente. Questo avviene generando un **salt**, cioè una stringa casuale tipicamente lunga 16 byte, e definendo un **cost factor**, che determina il numero di iterazioni da eseguire durante l'hashing. Ad esempio, impostando il cost factor a 10, verranno eseguite 2^{10} = 1024 iterazioni.

Durante l'hashing, la password viene combinata con il salt e su questa combinazione viene applicato un algoritmo crittografico. Il processo viene ripetuto per tutte le iterazioni definite, prendendo ogni volta l'output generato come input per la successiva iterazione.

Al termine, la password criptata viene salvata nel database e l'utente viene automaticamente loggato.

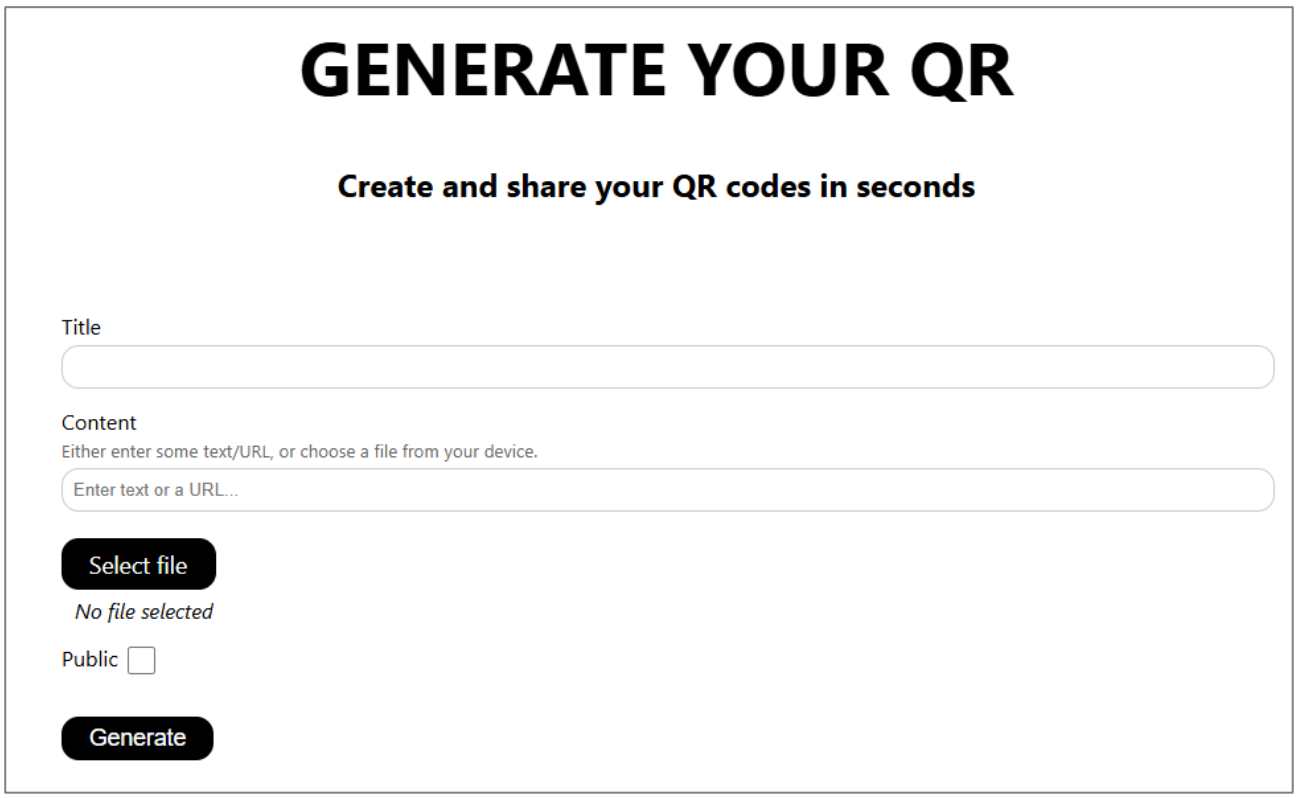
```
//Criptazione password
const salt = await bcrypt.genSalt(10);
const hashedPassword = await bcrypt.hash(password, salt);

//Salvataggio utente nel DB con password criptata
const newUser = new User({ username, password: hashedPassword });
await newUser.save();

//login automatico dopo la creazione
const user = await User.findOne({ username });

req.session.userId = user.id;
req.session.username = user.username;

res.redirect("/?msg=Creation%20succesfull");
```

Ricollegandosi al controllo della password durante il login, questo avviene estraendo la password dal database, recuperando il salt contenuto nell'hash salvato e applicando il processo di criptazione precedentemente descritto alla password inserita nel form di login. Se, al termine del processo, gli hash coincidono, il metodo restituisce true.

L'algoritmo utilizzato per la criptazione è Blowfish + Key Stretching. In pratica, Blowfish è un algoritmo di crittografia simmetrica progettato per cifrare dati; risulta molto veloce e sicuro ed è la base dell'algoritmo di hashing bcrypt. Key Stretching, invece, è una tecnica che aumenta il tempo necessario per calcolare un hash, rendendo più difficile un attacco brute-force. In pratica, è la parte che sfrutta il numero di iterazioni, ricalcolando più volte l'hash.

4.7 Generate QR

Una volta effettuato l'accesso, sarà possibile accedere alla pagina per generare i codici QR. In questa pagina è possibile creare QR per testi, URL o file. Sarà necessario inserire un titolo, quindi il testo o l'URL, oppure selezionare un file, e decidere se rendere il QR pubblico o privato.



4.7.1 Codice

4.7.1.1 Controllo eccezioni

Quando l'utente prova a generare un QR, viene prima verificato se non è stato inserito alcun contenuto oppure se sono stati inseriti contemporaneamente entrambi i tipi di contenuto, ovvero testo e file. In caso affermativo, viene restituito un errore specifico in base alla situazione.

```
router.post("/generate", requireAuth, upload.single("file"), async(req, res) =>{
    try{
        const { msg } = req.query;
        const {title, content} = req.body;
        const username = req.session.username;
        const public = req.body.checkboxP === "on";

        // Controllo: niente input
        if (!content && !req.file) {
            return res.status(400).render("generate_QR", {
                title: "Generate_QR",
                errorC: "You have not entered any text or file. Please choose one of the two."
            });
        }

        // Controllo: entrambi presenti
        if (content && req.file) {
            return res.status(400).render("generate_QR", {
                title: "Generate_QR",
                errorC: "You've entered both text and file. Choose only one."
            });
        }
```

4.7.1.2 Gestione contenuto tipo file

Una volta confermato che non ci sono errori, viene verificato il tipo di contenuto inserito. Nel caso si tratti di un file, questo viene caricato nella cartella uploads con un nome univoco, ottenuto unendo il nome originale del file con la data del momento della generazione. Successivamente, come contenuto effettivo del QR viene salvato il link al file nella cartella uploads. Questo procedimento è necessario perché non è possibile generare direttamente un QR a partire da un file.

```
//Insrimento nella cartella uploads se file
if (req.file) {
    //Generazione di un nome univoco pe evitare duplicati
    const newFileName =  Date.now() + "-" + req.file.originalname
    const uploadPath = path.join("uploads", newFileName); // cartella uploads
    fs.writeFile(uploadPath, req.file.buffer, (err) => {
        if (err) {
            return res.status(500).send("Error saving file");
        }
    });

    //Contenuto finale come percorso al file nel sito
    finalContent = `${req.protocol}://${req.get("host")}/uploads/${newFileName}`;
}
```

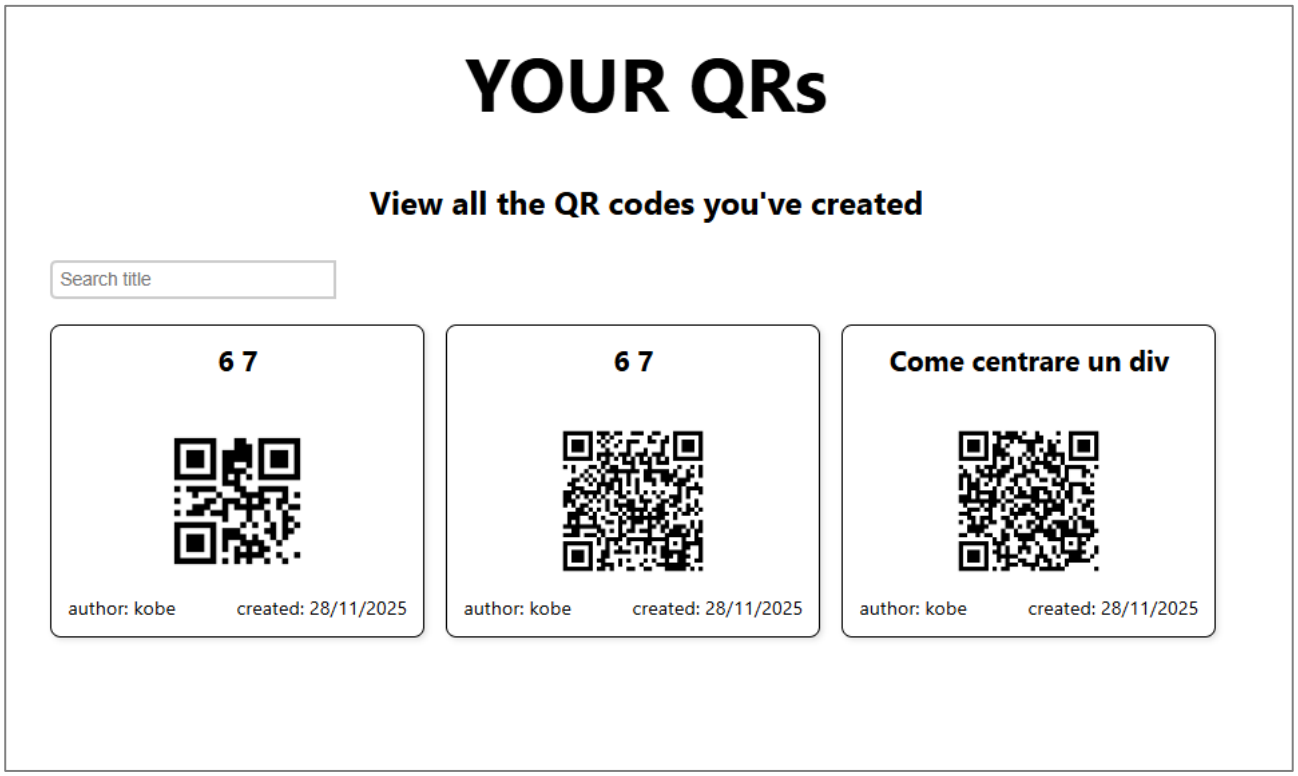
4.7.1.3 Generazione QR

Dopo aver definito e gestito il contenuto inserito, viene generato il QR code, che viene poi salvato nel database. Questo avviene utilizzando il metodo Gen_QR.toDataURL(), che crea una stringa in Base64 rappresentante l'immagine del QR. In questo modo è possibile inserire direttamente l'immagine nel browser semplicemente recuperandola dal database, senza doverla salvare su disco.

```
//Generazione immagine QR in base 64 e salvataggio nel DB
const qrBase64 = await Gen_QR.toDataURL(finalContent);

const newQR = new QR_Code({titolo: title,
    qr_image: qrBase64,
    utente: username,
    pubblico: public});

await newQR.save();

res.redirect("/generate_QR?msg=Generation%20succesfull");
```

4.8 Your QRs

Questa pagina permette di visualizzare la cronologia di tutti i QR creati dall'utente. La struttura è identica a quella della home, quindi le funzionalità disponibili sono le stesse. Le uniche differenze sono che la ricerca può essere eseguita solo per titolo, poiché l'utente è già conosciuto, e che vengono visualizzati anche i QR privati.



5 Test

5.1 Protocollo di test

Test Case: **Riferimento**:	TC-001 REQ-01	**Nome:**	Caricamento QR
Descrizione:	Controllare se i QR presenti nel database vengono caricati correttamente nella home		
Prerequisiti:	Avere un database con all'interno i QR e l'interfaccia della home		
Procedura:	Avviare l'applicazione ed entrare nel sito		
Risultati attesi:	vengono visualizzati i QR in delle card disposte a griglia		

Test Case: **Riferimento**:	TC-002 REQ-01	**Nome:**	Ricerca QR
Descrizione:	L'utente deve poter cercare un QR		
Prerequisiti:	Aver verificato il TC-001		
Procedura:	Avviare l'applicazione ed entrare nel sito Inserire un termine di ricerca nella barra di ricerca		
Risultati attesi:	Vengono visualizzati i QR in base alla ricerca, se non esiste nessun QR con i termini definiti deve apparire il messaggio nessun QR pubblico trovato		

Test Case: **Riferimento**:	TC-003 REQ-02	**Nome:**	Selezione QR
Descrizione:	L'utente deve poter cliccare su un QR per selezionarlo singolarmente		
Prerequisiti:	Aver verificato il TC-001		
Procedura:	Avviare l'applicazione ed entrare nel sito Cliccare su un QR		
Risultati attesi:	Il QR selezionato deve apparire in primo piano al centro dello schermo		

Test Case:	TC-004	**Nome:**	Stampa QR
Riferimento:	REQ-03		
Descrizione:	L'utente deve poter stampare il QR selezionato		
Prerequisiti:	Aver verificato il TC-003		
Procedura:	Avviare l'applicazione ed entrare nel sito Selezionare un QR Cliccare sull'icona della stampa		
Risultati attesi:	Deve aprirsi la finestra di stampa per poter stampare il QR		

Test Case:	TC-005	**Nome:**	Creazione account
Riferimento:	REQ-04		
Descrizione:	L'utente deve poter creare un account		
Prerequisiti:	Avere una pagina di creazione account		
Procedura:	Avviare l'applicazione ed entrare nel sito Entrare nella pagina di login Non avendo un account cliccare sul link per crearne uno Inserire nome utente e password rispettando i criteri		
Risultati attesi:	Dovrebbe reindirizzare l'utente alla home con il messaggio "creation succesfull" e sbloccare le nuove funzionalità		

Test Case:	TC-006	**Nome:**	Login
Riferimento:	REQ-04		
Descrizione:	L'utente deve poter eseguire il login		
Prerequisiti:	Aver verificato il TC-005 Avere una pagina di login		
Procedura:	Avviare l'applicazione ed entrare nel sito Se già connessi cliccare su logout Entrare nella pagina di login Inserire nome utente e password		
Risultati attesi:	Dovrebbe reindirizzare l'utente alla home con il messaggio "login succesfull" e sbloccare le nuove funzionalità		

Test Case:	TC-007	**Nome:**	Creazione QR
Riferimento:	REQ-05		
Descrizione:	L'utente deve poter creare un QR di tipo testo o file		
Prerequisiti:	Aver eseguito il login Avere una pagina di creazione QR		
Procedura:	Avviare l'applicazione ed entrare nel sito Eseguire il login se non si è già collegati Entrare nella pagina di creazione QR Riempire i campi seguendo le regole		
Risultati attesi:	Dovrebbe reindirizzare l'utente alla home con il messaggio "generation succesfull" ed il QR dovrebbe essere visualizzato nella home se reso pubblico		

Test Case:	TC-008	**Nome:**	Visualizzazione cronologia QR creati
Riferimento:	REQ-06		
Descrizione:	L'utente deve poter visualizzare la sua cronologia di QR creati		
Prerequisiti:	Aver eseguito il login Avere una pagina per la cronologia		
Procedura:	Avviare l'applicazione ed entrare nel sito Eseguire il login se non si è già collegati Entrare nella pagina di cronologia (Your QRs)		
Risultati attesi:	La pagina dovrebbe mostrare solo i QR creati dall'utente sia pubblici che privati		

5.2 Risultati test

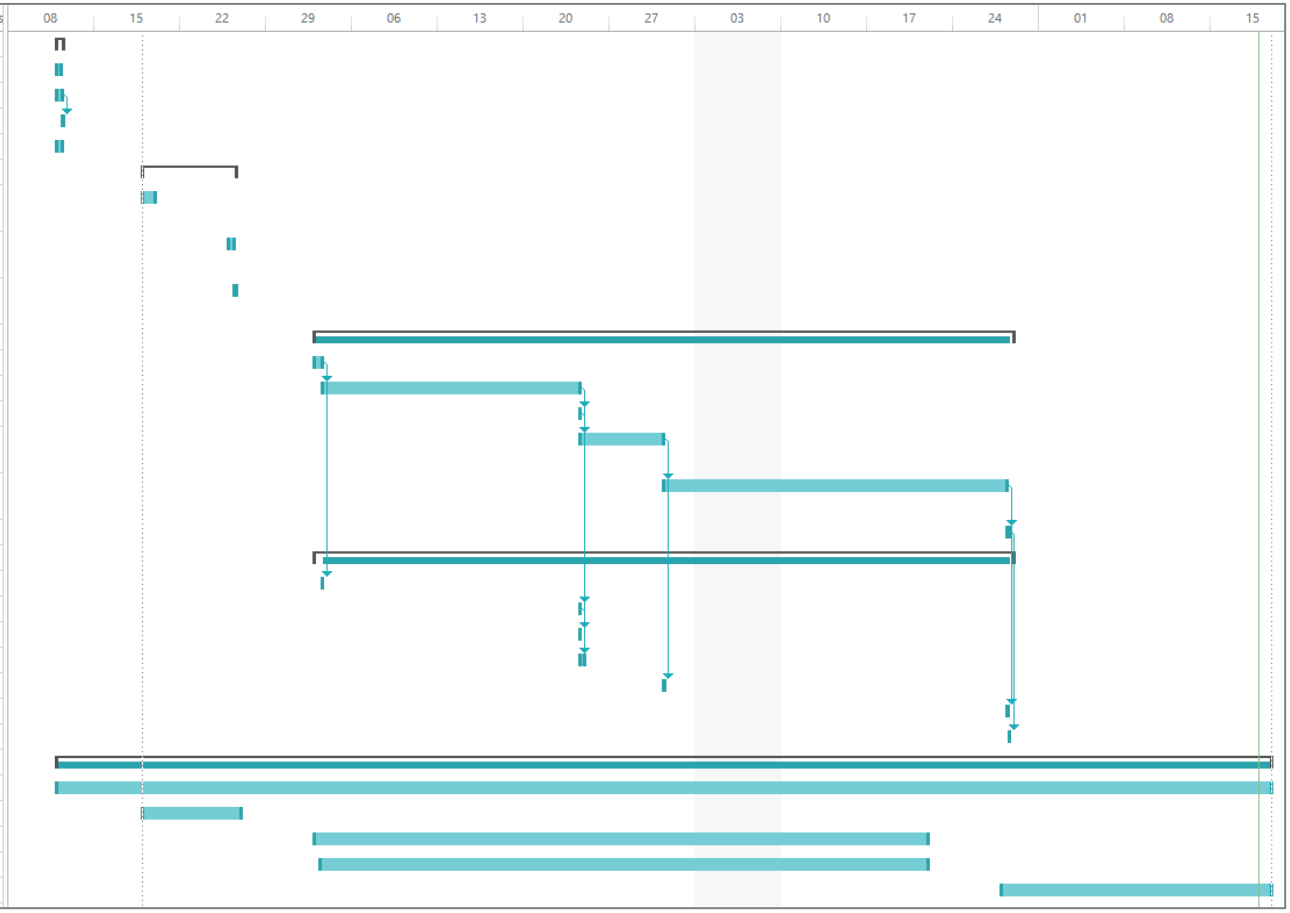
Test Case	Data	Risultato
TC-001	24.10.2025	Passato
TC-002	24.10.2025	Passato
TC-003	24.10.2025	Passato
TC-004	28.11.2025	Passato
TC-005	31.10.2025	Passato
TC-006	31.10.2025	Passato
TC-007	28.11.2025	Passato
TC-008	28.11.2025	Passato

5.3 Mancanze/limitazioni conosciute

Rispetto ai requisiti inseriti non ci sono delle mancanze però il fatto di dover salvare su una cartella ogni volta che un utente genera il QR per un file risulta in una limitazione in un caso in qui venga reso disponibile per il pubblico perché potrebbe dover richiedere troppo spazio per contenere tutti i file degli utenti.

6 Consuntivo

Nome attività	Lavoro	Inizio	Fine	Predeces
Analisi	**8 h**	**ven 12.09.25**	**ven 12.09.25**	
Analisi dominio	1 h	ven 12.09.25	ven 12.09.25	
analisi requisiti	2 h	ven 12.09.25	ven 12.09.25	
analisi requisiti	2 h	ven 12.09.25	ven 12.09.25	3
analisi dei mezzi	3 h	ven 12.09.25	ven 12.09.25	
progettazione	**13 h**	**ven 19.09.25**	**ven 26.09.25**	
progettazione di massima	8 h	ven 19.09.25	ven 19.09.25	
progettazione interfaccia	2 h	ven 26.09.25	ven 26.09.25	
progettazione data base	3 h	ven 26.09.25	ven 26.09.25	
Sviluppo	**49 h**	**ven 03.10.25**	**ven 28.11.25**	
Data base	5 h	ven 03.10.25	ven 03.10.25	
pagina home	14 h	ven 03.10.25	ven 24.10.25	11
selezione QR	2 h	ven 24.10.25	ven 24.10.25	12
creazione login / accesso	9 h	ven 24.10.25	ven 31.10.25	13
pagina creazione QR	17 h	ven 31.10.25	ven 28.11.25	14
Stampa QR	2 h	ven 28.11.25	ven 28.11.25	15
Test	**7 h**	**ven 03.10.25**	**ven 28.11.25**	
popolazione DB	1 h	ven 03.10.25	ven 03.10.25	11
test lista QR	1 h	ven 24.10.25	ven 24.10.25	12
test filtri	1 h	ven 24.10.25	ven 24.10.25	19
test selezione QR	1 h	ven 24.10.25	ven 24.10.25	13
test login/accesso	1 h	ven 31.10.25	ven 31.10.25	14
test creazione QR	1 h	ven 28.11.25	ven 28.11.25	15
test stampa QR	1 h	ven 28.11.25	ven 28.11.25	16
Documentazioe	**297 h**	**ven 12.09.25**	**ven 19.12.25**	
Diario	120 h	ven 12.09.25	ven 19.12.25	
Prorogettazione	13 h	ven 19.09.25	ven 26.09.25	
Implementazione	64 h	ven 03.10.25	ven 21.11.25	
Test	64 h	ven 03.10.25	ven 21.11.25	
Correzioni	36 h	ven 28.11.25	ven 19.12.25	



Titolo del progetto: QR Code generator
Alunno/a: Riley Bianchi
Classe: Info I3AC
Anno scolastico: 2025/2026
Docente responsabile: Ingrid Cereda

6.1 Analisi consuntivo

Come si può notare, l'intero periodo dedicato allo sviluppo è variato notevolmente rispetto alla previsione iniziale, comportando l'aggiunta di numerose ore di lavoro. Ciò è dovuto in gran parte al fatto che questa è stata la mia prima esperienza con Node.js; di conseguenza, ho dedicato molto tempo all'analisi della logica e della struttura del progetto, rallentando inizialmente il ritmo di lavoro. Una volta compresi questi aspetti, è stato però più semplice applicarli al resto del codice.

Inoltre, ho riscontrato alcune difficoltà nella gestione della logica legata a specifici requisiti del progetto, come la creazione dei QR e la crittografia delle password, attività che hanno richiesto ulteriore tempo. Infine, non avevo inizialmente considerato le ore necessarie per lo studio teorico e per i vari test; pertanto, oltre alle difficoltà incontrate durante lo sviluppo, sono state sottratte ulteriori ore al lavoro previsto.

7 Conclusioni

Ritengo che il progetto realizzato non rappresenti un'innovazione particolarmente rilevante, in quanto esistono già numerosi siti in grado di generare QR code. Tuttavia, posso affermare che il modo in cui è stata impostata la condivisione dei QR risulta interessante e rende il progetto potenzialmente apprezzabile dal pubblico.

7.1 Sviluppi futuri

Una possibile miglioria è sicuramente quella citata in precedenza, ovvero il caricamento dei file nella cartella uploads. In particolare, sarebbe utile trovare un metodo che, in caso di pubblicazione, non comporti rischi legati allo spazio di archiviazione.

Un possibile sviluppo futuro, invece, potrebbe essere l'implementazione di un sistema di analisi dei file caricati, al fine di prevenire comportamenti malevoli, come l'inserimento di virus, link a siti sospetti o contenuti espliciti o violenti. Questo permetterebbe di aumentare ulteriormente il livello di sicurezza dell'applicazione.

7.2 Considerazioni personali

Questo progetto mi ha permesso di scoprire Node.js e il suo funzionamento, aspetto che mi è piaciuto molto. Mi ha aiutato a comprendere in modo abbastanza chiaro come viene gestita un'applicazione di questo tipo, portandomi a riflettere sulla struttura dei file e su come collegarli tra loro. Questo mi è stato particolarmente utile anche per PHP, materia che abbiamo studiato in parallelo a questo progetto, soprattutto quando è stata introdotta la struttura MVC, che risulta abbastanza simile a quella utilizzata in Node.js.

A mio parere, Node.js è uno strumento molto utile e performante, e non mi dispiacerebbe riutilizzarlo in futuro.

8 Glossario

Inserite una semplice tabella con due colonne che spieghi i termini specifici del progetto (lista dei termini in ordine alfabetico A-Z)

Esempio:

Termine	Descrizione
QR Code	**Quick Response Code,** indica un codice a barre bidimensionale che memorizza informazioni (URL, testo, numeri di telefono, ecc.) e le rende accessibili istantaneamente quando scansionato con uno smartphone o lettore ottico, collegando il mondo fisico a quello digitale in modo veloce ed efficiente**.**

9 Bibliografia

9.1 Sitografia

- https://nodejs.org/en Node.js
- https://www.mongodb.com/ mongodb
- https://stackoverflow.com/ stack overflow
- https://www.w3schools.com/ w3schools
- https://chatgpt.com/ chatGPT

Esempio:

- http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.