



UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA

DIMES

Corso di laurea Magistrale in ingegneria Informatica -
Cybersecurity

SISTEMI DISTRIBUITI E CLOUD COMPUTING

RELAZIONE PROGETTO

Implementare un sistema distribuito online per gestire gli
appuntamenti di uno studio legale utilizzando servizi Amazon
AWS

Professore

Prof. Domenico Talia

Ing. Loris Belcastro

Candidato

Vincenzo Silva 257197

Sommario

Indice delle figure	2
Descrizione progetto	2
Traccia	3
Struttura del progetto	4
Installazione ed esecuzione del progetto	5
Descrizione dettaglio	5
Descrizione funzionale della web application:	9
Requisiti funzionali	13
Requisiti non funzionali	16
Mappatura con i servizi AWS	17
Link progetto	18

Indice delle figure

Figura 1- composizione Backend	4
Figura 2 - Composizione frontend	5
Figura 3 - Database	5
Figura 4 - composizione Middleware	6
Figura 5 - Composizione routes	6
Figura 6 - Composizione utils.....	6
Figura 7 - tabella appuntamenti	7
Figura 8 - tabella utenti dettaglio ruolo	7
Figura 9 - implementazione Amazon RDS	8
Figura 10 - Implementazione Bucket S3	8
Figura 11 - Configurazione nginx	9
Figura 12 - Homepage	9
Figura 13 - Chi Siamo	10
Figura 14 - login.....	10
Figura 15 - Register.....	11
Figura 16 - Cliente	11
Figura 17 - dettaglio appuntamenti	12
Figura 18 - Profilo	13
Figura 19 - Admin	13
Figura 20 - dettaglio verificaToken	14
Figura 21- Dettaglio verificaRuolo	14
Figura 22 - esempio bucket S3	15
Figura 23 - Configurazione Bucket S3	15
Figura 24 - esempio e-mail 1	16
Figura 25 - esempio e-mail 2	16

Descrizione progetto

Il progetto “Studio Legale” è una piattaforma web che consente la gestione di appuntamenti

legali tra clienti e lo studio legale. L'applicazione prevede un frontend in HTML e Javascript, un backend in Node.js/Express e l'integrazione con i servizi cloud AWS per garantire scalabilità, sicurezza e disponibilità dei dati. Il sistema utilizza Amazon RDS (MySQL) per la gestione delle informazioni strutturate.

Traccia

Studente: _____ Vincenzo Silva _____ Matricola: _____ 257197 _____

Data di assegnazione del progetto: _____ 20/05/2024 _____

Progettare e implementare un sistema distribuito online per gestire gli appuntamenti degli avvocati di uno studio legale. Il sistema dovrà consentire all'amministratore di definire diversi tipi di appuntamenti (ad esempio consulenza fiscale, procedimento civile, procedimento penale, ecc.). Per ciascun tipo di appuntamento, sarà possibile definire gli orari disponibili in una specifica giornata (ad esempio, il 10 marzo 2025 saranno disponibili 4 per consulenza fiscale, 2 per procedimenti penali, ecc.). Ai clienti sarà consentito:

- Collegarsi al sistema, selezionare e prenotare un appuntamento che desidera avere. All'atto della prenotazione, il cliente dovrà poter allegare anche dei documenti (denunce, atti legali, ecc.).
- Cancellare un appuntamento già prenotato.
- Mettersi in lista d'attesa per essere prenotati nel caso in cui si libera uno slot di appuntamento. Si chiede allo studente di studiare soluzioni adeguate per la gestione di tale funzionalità (es. notifiche via email, meccanismi di conferma, ecc.).

Per la realizzazione dell'applicazione, lo studente dovrà includere un'analisi dei requisiti funzionali e non funzionali del sistema da realizzare. Lo studente non ha vincoli sul linguaggio di programmazione da utilizzare. Il sistema realizzato, tuttavia, dovrà utilizzare le soluzioni di calcolo, storage e virtualizzazione messe a disposizione da:

- Amazon AWS

Per il completamento del progetto, lo studente dovrà presentare una demo funzionante dell'applicazione sopra descritta.

Consegna e valutazione del progetto

La consegna del progetto consiste in:

- consegna mediante CD-ROM o link a spazio di Cloud storage contenente tutti i sorgenti necessari per il funzionamento del sistema e una relazione in formato pdf;
- copia cartacea della relazione da presentare il giorno dell'orale.

I principali criteri di valutazione del progetto saranno:

1. rispondenza ai requisiti;
2. originalità;
3. organizzazione del codice (es., leggibilità e modularità);
4. organizzazione e completezza della relazione.

Il progetto deve essere consegnato due giorni prima della data dell'esame.

Struttura del progetto

Il progetto è strutturato nel seguente modo:

La parte di backend è stata costruita utilizzando principalmente Javascript con l'integrazione di framework specifici nel caso d'uso, tra questi troviamo Node.js con Express.js per la comunicazione. Questa parte si occupa della ricezione e della gestione dei messaggi ricevuti lato frontend con annessa gestione logica applicativa con restituzione di dati in formato standard in JSON, dell'interazione con il Database (MySQL), della sicurezza, dell'autenticazione e autorizzazione degli accessi degli utenti (imponendo una restrizione fra l'utente e il root del server).

Nome	Ultima modifica	Tipo	Dimensione
middleware	05/09/2025 16:40	Cartella di file	
node_modules	03/09/2025 20:45	Cartella di file	
routes	29/08/2025 20:50	Cartella di file	
utils	06/09/2025 15:31	Cartella di file	
.env	06/09/2025 15:35	File ENV	1 KB
db.js	06/09/2025 15:40	JSFile	1 KB
package	06/09/2025 12:01	File di origine JSON	1 KB
package-lock	06/09/2025 12:01	File di origine JSON	195 KB
s3.js	06/09/2025 15:35	JSFile	1 KB
server.js	06/09/2025 15:36	JSFile	1 KB

Figura 1- composizione Backend

La parte di frontend invece si occupa principalmente dell'interazione con l'utente e della gestione della UI della web application. Qui è stato adottato principalmente HTML per la struttura degli elementi integrando CSS per lo stile e il design, javascript invece per l'interattività e la struttura

logica degli script.

img	05/09/2025 17:15	Cartella di file	
admin	06/09/2025 15:38	Chrome HTML Do...	11 KB
chiamo	06/09/2025 15:38	Chrome HTML Do...	3 KB
cliente	06/09/2025 15:38	Chrome HTML Do...	14 KB
homepage	06/09/2025 15:38	Chrome HTML Do...	5 KB
login	06/09/2025 15:39	Chrome HTML Do...	5 KB
profilo	06/09/2025 15:39	Chrome HTML Do...	11 KB
register	12/08/2025 14:50	Chrome HTML Do...	4 KB
style	06/09/2025 15:39	File di origine CSS	3 KB

Figura 2 - Composizione frontend

Il database è di tipo relazionale ed è stato strutturato con MySQL. Al tuo interno sono presenti le seguenti tabelle.

Nome ^	Righe	Dimensi...	Creato	Aggiornato	Motore	Commento	Tipo
appuntamenti	2	16,0 KiB	2025-08-30 08:34:47		InnoDB		Table
lista_attesa	0	48,0 KiB	2025-08-30 08:34:48	2025-09-07 10:17:39	InnoDB		Table
prenotazioni	3	32,0 KiB	2025-09-06 12:58:07	2025-09-07 10:17:39	InnoDB		Table
slot_appuntam...	15	32,0 KiB	2025-08-30 08:34:49	2025-09-07 10:17:39	InnoDB		Table
utenti	4	32,0 KiB	2025-08-30 08:34:49		InnoDB		Table

Figura 3 - Database

Sono stati utilizzati per il progetto diversi servizi di Amazon AWS, tra questi troviamo Amazon RDS per la gestione del database relazionale configurato in MySQL, Amazon EC2 come server virtuale in cloud, Amazon Bucket S3 per archiviare e recuperare i documenti caricati dai clienti nella web App, Amazon SES (Simple Email Service) per l'invio e ricezione di e-mail ed Amazon IAM (identity and Access Management) per la gestione in modo sicuro degli utenti, gruppi, ruoli e permessi all'interno della gestione Amazon AWS.

Altre tecnologie utilizzate per l'esecuzione del progetto sono Nginx per l'esecuzione web sulla porta 80 ed utilizzato come reverse proxy, JWT per la gestione dei token in modo sicuro.

Installazione ed esecuzione del progetto

Per installare il progetto su un nuovo dispositivo è necessario installare i "node modules" attraverso il comando "npm install" ed installare/configurare Nginx.

Per avviare il progetto è necessario eseguire l'accesso in ssh con le credenziali di EC2 ed eseguire nella cartella di backend il comando "npm start"

Descrizione dettaglio

Backend:

Nel backend sono presenti i seguenti file e cartelle:

- Middleware: sono presenti i file `auth.js` e `verificareRuolo.js`. Questi file si occupano di verificare il token e il ruolo dell'utente loggato.


 <code>auth.js</code>	06/09/2025 15:24	JSFile	1 KB
 <code>verificaRuolo.js</code>	22/04/2025 13:58	JSFile	1 KB

Figura 4 - composizione Middleware

- Node_modules: all'interno di questa cartella sono presenti le librerie ed i moduli generati automaticamente attraverso "npm install" di node.js e di altre tecnologie utilizzate.
- Routes: sono presenti due file, `appointments.js` e `auth.js`. Nel primo file sono presenti gli endpoint delle chiamate API gestite dal router Express.js. Qui avvengono le prenotazioni degli utenti, la gestione degli account e della comunicazione attraverso l'invio di e-mail con il servizio Amazon SES. Nel secondo file invece sono presenti le chiamate API inerenti alla gestione del login e della registrazione dell'utente.



 <code>appointments.js</code>	06/09/2025 15:31	JSFile	19 KB
 <code>auth.js</code>	06/09/2025 15:26	JSFile	3 KB

Figura 5 - Composizione routes

- Utils: qui è presente il file `mailer.js`. Al suo interno viene configurato il client SES e viene definita la funzione per l'invio delle e-mail.

 <code>mailer.js</code>	07/09/2025 11:12	JSFile	1 KB
--	------------------	--------	------

Figura 6 - Composizione utils

- .env: è un file di testo usato per memorizzare le variabili d'ambiente, in modo da separare configurazioni sensibili e variabili di ambiente dal codice sorgente.
- Db.js: Al suo interno è configurato il client RDS e la gestione/connessione al database
- Package.json: file di configurazione del progetto di Node.js. contiene informazioni sul progetto e sulle dipendenze.
- Package-lock.json: file generato automaticamente da npm.
- S3.js: All'interno del seguente file è presente la gestione e configurazione del servizio Amazon Bucket S3.
- Server.js: è il file principale del server express.js, si occupa della configurazione del backend e della connessione tra frontend e API.

Frontend:

Nel Frontend sono presenti i seguenti file e cartelle:

- Img: le immagini utilizzate nella UI della web application

- Admin.html: questo file HTML rappresenta la pagina di amministrazione per la gestione degli appuntamenti. Al suo interno è presente un calendario interattivo.
- Chisiamo.html: pagina HTML con una breve descrizione immaginaria sull'attività.
- Client.html: in questo file è rappresentata la pagina di gestione degli appuntamenti da parte del client. Anche qui è presente un calendario interattivo per potersi prenotare/mettersi in attesa per gli appuntamenti.
- Homepage.html: pagina HTML che rappresenta la homepage principale.
- Login.html: questo file HTML si occupa di rappresentare la pagina di login.
- Register.html: questo file HTML si occupa di rappresentare la pagina di registrazione. Al suo interno è presente un form da compilare con le informazioni del client.
- Style.css: foglio di stile con le implementazioni grafiche

Database:

Nel database sono presenti le seguenti tabelle:

- Appuntamenti: qui sono presenti la tipologia di appuntamenti disponibili


#	1 id 	2 nome
1	1	consulenza fiscale
2	2	procedimento penale

Figura 7 - tabella appuntamenti

- Lista_attesa: le informazioni inerenti ai clienti in lista d'attesa in base all'appuntamento
- Prenotazioni: le prenotazioni effettuate dagli utenti
- Slot_appuntamenti: gli appuntamenti disponibili creati dell'utente admin
- Utenti: gli utenti registrati. Qui è presente il campo per definire il ruolo dell'utente.

7 ruolo
admin
cliente
cliente
cliente
cliente

Figura 8 - tabella utenti dettaglio ruolo

Servizi Amazon AWS:

- EC2: è stato configurato con il sistema operativo amazon linux utilizzando come tipo istanza il "t3.nano", questa scelta è stata applicata per l'efficienza e il basso consumo economico. Per potersi collegare alla web application è stata applicata la regola in entrata sulla porta TCP 80. Attraverso "Elastic IP" è stato settato un indirizzo statico pubblico, in modo tale che allo spegnimento del sistema operativo e al successivo avvio dell'istanza non venisse settato un nuovo indirizzo IP.
- RDS: è stato configurato con il motore MySQL community con le dimensioni "db.t4g.micro". Per potersi collegare attraverso HeidiSQL è stata applicata la regola in entrata sul protocollo TCP sulla porta 3306.

```
import mysql from 'mysql2';
import dotenv from "dotenv";

dotenv.config();

export const db = mysql.createConnection({
  host: process.env.DB_HOST,
  port: process.env.DB_PORT,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
  timezone: 'Z',
  dateStrings: true
});

db.connect(err => {
  if (err) throw err;
  console.log('Connessione al database RDS riuscita!');
});
```

Figura 9 - implementazione Amazon RDS

- Bucket S3: l'upload dei documenti degli utenti viene gestito con la creazione di una folder specifica per ogni singolo client. Al suo interno è possibile visualizzare i file caricati dagli utenti. Il bucket è settato come private ma se l'utente volesse visualizzare i file contenuti solo all'interno della propria cartella, attraverso la web application è possibile generare un Url temporaneo per la visualizzazione del file.

```
import dotenv from "dotenv";
import { S3Client } from "@aws-sdk/client-s3";

dotenv.config();

const s3 = new S3Client({
  region: process.env.AWS_REGION,
  credentials: {
    accessKeyId: process.env.AWS_ACCESS_KEY_ID,
    secretAccessKey: process.env.AWS_SECRET_ACCESS_KEY,
  },
});

export default s3;
```

Figura 10 - Implementazione Bucket S3

- SES: è stato configurato utilizzando come "key" solo l'e-mail dell'utente. Essendo un progetto universitario e per abbattere i costi è stato scelto di mantenere la configurazione in sandbox. Questo comporta che nonostante il servizio SES funzioni è necessario che l'utente venga inserito preventivamente come identità nel pool di configurazione.
- IAM: Qui sono stati configurati gli utenti per la gestione dei servizi attivi e dei permessi con accesso programmatico

Nginx

- È stato configurato per comunicare sulla porta 80 e da reverse proxy sulla porta 3000 (dove è abilitata la configurazione di node.js), qui per risolvere il problema delle chiamate API è stata impostata la regola per l'esecuzione degli endpoint.

```
server {
    listen 80;
    server_name 13.49.80.226;

    # Frontend static files
    location / {
        root /home/ec2-user/progetto/legal-appointments/frontend;
        index homepage.html;
        try_files $uri /homepage.html;
    }

    # API proxy
    location /api/ {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Figura 11 - Configurazione nginx

Descrizione funzionale della web application:

Collegandosi all'indirizzo del server EC2 è possibile visualizzare la pagina principale della web application. Qui è possibile selezionare tre scelte: “chi siamo”, “Prenota un appuntamento” e “Profilo” in base alla scelta saremo indirizzati verso una pagina specifica.

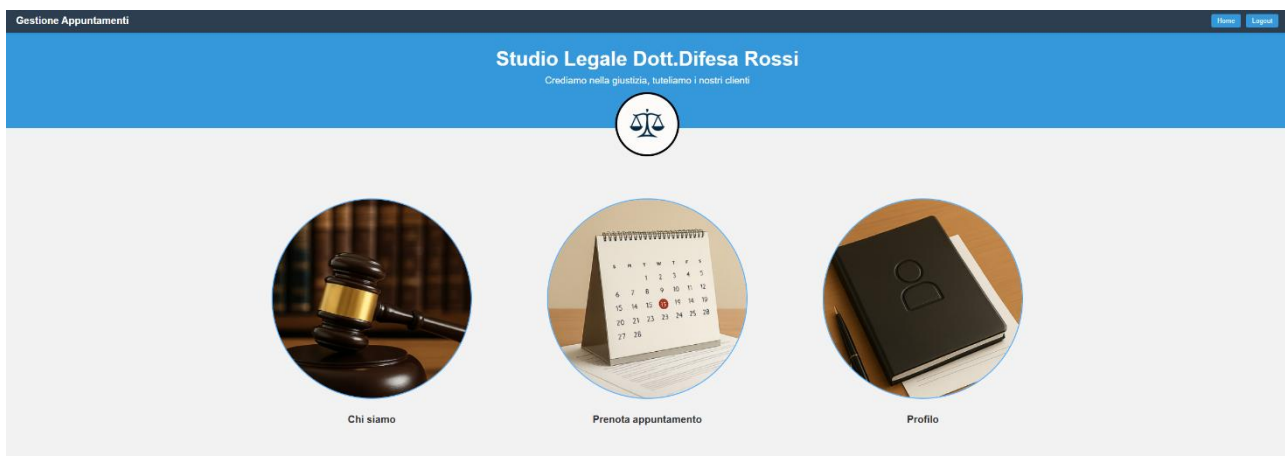


Figura 12 - Homepage

Se dovessimo premere “Chi siamo” verrà visualizzata una pagina descrittiva dell’attività:

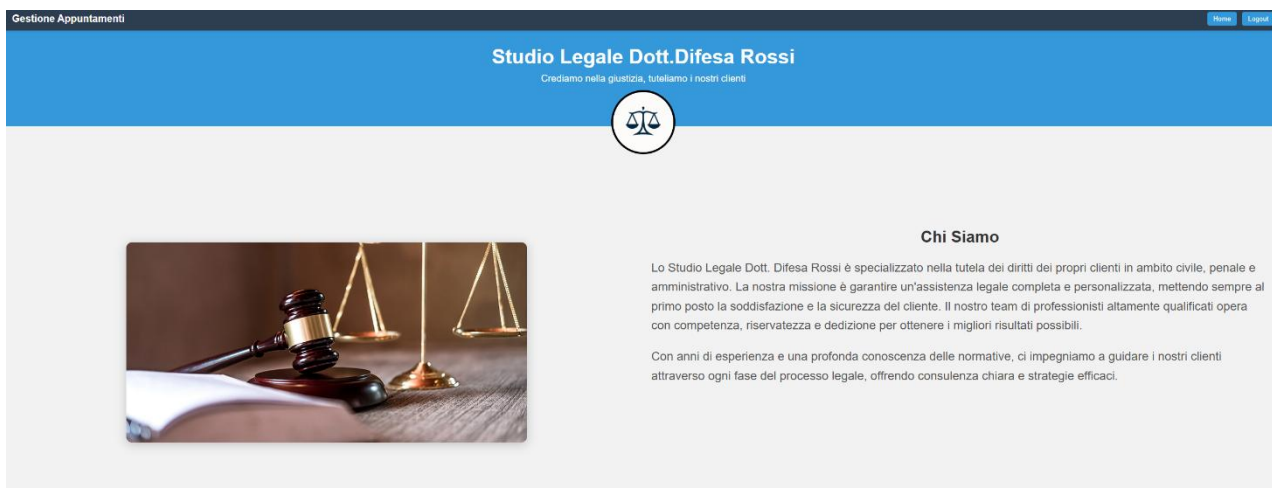


Figura 13 - Chi Siamo

Invece attraverso il tasto “Profilo” ci saranno due percorsi da intraprendere. Nel caso non fossimo loggati verremo indirizzati nella pagina di “login”. Qui è possibile accedere con il proprio account (se registrato precedentemente)

Accedi

Email

Password

Accedi

Non hai un account? [Registrati](#)

Figura 14 - login

Altrimenti se non in possesso delle credenziali è possibile premere sul tasto “Register” dove sarà possibile registrare l’utente.

Registrazione

Nome

Cognome

Numero di telefono

Email

Password

Conferma Password

[Registrati](#)

Hai già un account? [Accedi](#)

Figura 15 - Register

Al termine della procedura sarà possibile visualizzare la pagina “Cliente”. Nel header sarà visualizzato il proprio nome ed i tasti home e logout, il primo serve a ritornare sulla pagina principale (Homepage) il secondo a disconnettere la sessione con l’account collegato.

Gestione Appuntamenti Vincenzo Home Logout

Studio Legale Dott. Difesa Rossi
 Crediamo nella giustizia, tuteliamo i nostri clienti

August 2025 today < >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
	3 Con Fisc. - 09:00:00 - (1) Con Fisc. - 09:00:00 - (1) Con Fisc. - 09:00:00 - (1) Fisc. Pen. - 14:00:00 - (1)	4 Con Fisc. - 10:12:00 - (2)	5 Con Fisc. - 06:50:00 - (2)	6 Con Fisc. - 06:57:00 - (1)	7	8 Con Fisc. - 06:50:00 - (1) Con Fisc. - 06:50:00 - (1)
10	11	12	13	14	15	16

Appuntamenti del giorno
 Seleziona un giorno per visualizzare gli appuntamenti.

Mie Prenotazioni
 Nessuna prenotazione trovata

Figura 16 - Cliente

Nel “body” invece è presente il calendario con gli appuntamenti disponibili (in verde quelli disponibili i rosso non disponibili ma prenotabili). Gli appuntamenti sono visualizzati come “tipo di

appuntamento” – “orario” e “disponibilità”. Cliccando sulla giornata specifica verranno visualizzati nel dettaglio gli appuntamenti e le proprie prenotazioni. Qui è possibile fare l’upload dei file.

The screenshot displays a user interface for managing appointments and bookings. It is divided into two main sections: 'Appuntamenti del giorno' and 'Mie Prenotazioni'.

Appuntamenti del giorno

- Appointment 1:** Data: 04/08/2025, Orario: 09:00:00, Tipo: consulenza fiscale, Disponibili: 1. A green 'Prenota' button is present.
- Appointment 2:** Data: 04/08/2025, Orario: 09:00:00, Tipo: consulenza fiscale, Disponibili: 1. A green 'Prenota' button is present.
- Appointment 3:** Data: 04/08/2025, Orario: 09:40:00, Tipo: consulenza fiscale, Disponibili: 1. A green 'Prenota' button is present.
- Appointment 4:** Data: 04/08/2025, Orario: 14:00:00, Tipo: procedimento penale, Disponibili: 0. A red button reads 'Pieno - Mi metto in attesa'.

Mie Prenotazioni

- Booking 1:** 04/08/2025 - procedimento penale. Below the title is a trash icon, a 'Scegli file' button, the text 'Nessun file selezionato', and a blue 'Carica documenti' button.

Figura 17 - dettaglio appuntamenti

Mentre nella pagina “Profilo” sono presenti “informazioni Utente” con i campi inseriti in fase di registrazione e la possibilità di cambiare e-mail e password (si verrà avvisati tramite il servizio Amazon SES), il campo “I miei documenti” dove è possibile visionare i documenti caricati sul bucket S3, infine “Le mie prenotazioni” dove è presente un recap delle prenotazioni effettuate.

Figura 18 - Profilo

Nel caso fossimo in modalità “Admin”, invece di visionare la pagina “Cliente” verremo indirizzati nella pagina “Admin” dove sarà possibile visualizzare e gestire gli appuntamenti.

Figura 19 - Admin

Requisiti funzionali

L'applicazione per uno studio legale deve consentire:

Gestione utenti

- Registrazione e autenticazione sicura: attraverso l'utilizzo dei token JWT

```
import jwt from 'jsonwebtoken';
import dotenv from 'dotenv';

dotenv.config();

const JWT_SECRET = process.env.JWT_SECRET;

function verificaToken(req, res, next) {
  const token = req.headers.authorization?.split(' ')[1];
  if (!token) return res.status(401).json({ error: 'Token mancante' });

  try {
    const decoded = jwt.verify(token, JWT_SECRET);
    req.user = decoded;
    next();
  } catch {
    return res.status(401).json({ error: 'Token non valido' });
  }
}

//module.exports = verificaToken;
export default verificaToken;
```

Figura 20 - dettaglio verificaToken

- Differenziazione tra clienti e avvocato (admin): Le routes sono impostate in base al tipo di utente collegato, per poter accedere come admin è necessario impostare un campo “ruolo” che non è possibile settare attraverso la web application
- Ogni utente può accedere solo alle proprie informazioni: Sono state implementati dei controlli sui token e sui ruoli. Non è possibile accedere alle informazioni degli altri utenti

```
export default function verificaRuolo(ruoloRichiesto) {
  return (req, res, next) => {
    if (req.user?.ruolo !== ruoloRichiesto) {
      return res.status(403).json({ error: 'Accesso negato: ruolo insufficiente' });
    }
    next();
  };
}
```

Figura 21- Dettaglio verificaRuolo

Gestione prenotazioni

- Il cliente può prenotare un appuntamento scegliendo giorno e orario: attraverso il calendario è possibile selezionare un determinato giorno e prenotare/mettersi in attesa di un appuntamento
- L'avvocato (admin) può visualizzare le prenotazioni ricevute: attraverso il calendario l'admin può visualizzare gli appuntamenti calendarizzati
- Possibilità di modificare o cancellare una prenotazione: Sia l'utente che l'admin possono gestire gli appuntamenti (l'utente il proprio l'admin tutti). In aggiunta l'admin può cancellare anche un appuntamento prenotato da un utente.

Gestione documenti

- Il cliente può caricare uno o più documenti relativi a una prenotazione: i documenti saranno caricati sul bucket S3 e catalogati in base all'utente collegato (ogni utente ha il proprio folder)


<input type="checkbox"/>	Nome	▲	Tipo
<input type="checkbox"/>	 utente_4/		Cartella

Figura 22 - esempio bucket S3

- L'avvocato può scaricare i documenti caricati dai clienti: attraverso il bucket S3 l'admin può visualizzare e scaricare i documenti caricati dagli utenti.
- I documenti devono essere conservati in cloud (AWS S3) e devono essere visualizzabili dagli utenti: attraverso "profilo" è possibile visualizzare i documenti caricati in base al tipo di appuntamento, qui verrà generato un link temporaneo privato per quella sessione per visualizzare il contenuto.

```
import { SESClient, SendEmailCommand } from "@aws-sdk/client-ses";
import dotenv from "dotenv";

dotenv.config();

const ses = new SESClient({
  region: process.env.AWS_REGION,
  credentials: {
    accessKeyId: process.env.AWS_ACCESS_KEY_ID_SES,
    secretAccessKey: process.env.AWS_SECRET_ACCESS_KEY_SES
  }
});

export async function inviaEmail(destinatario, oggetto, testo) {
  const params = {
    Source: process.env.EMAIL_SES,
    Destination: { ToAddresses: [destinatario] },
    Message: {
      Subject: { Data: oggetto, Charset: "UTF-8" },
      Body: { Text: { Data: testo, Charset: "UTF-8" } }
    }
  };

  try {
    const result = await ses.send(new SendEmailCommand(params));
    //console.log("Email inviata con SES:", result.MessageId);
    return result;
  } catch (err) {
    console.error(" Errore invio email:", err);
    throw err;
  }
}
```

Figura 23 - Configurazione Bucket S3

Notifiche

- Conferma all'utente quando un documento o una prenotazione è stata caricata/modificata: attraverso degli alert l'utente viene notificato dell'esito dell'esecuzione del codice.

- Possibilità di inviare notifiche e-mail (AWS SES): l'utente viene notificato via e-mail attraverso il servizio di Amazon SES. Viene notificato del cambio password, nel caso fosse avvenuta la prenotazione dopo essersi messo in coda, della cancellazione del proprio appuntamento nel caso l'admin dovesse annullarlo.

Ciao! Uno slot si è liberato per il giorno 10/08/2025 e ti è stato automaticamente assegnato.



Figura 24 - esempio e-mail 1

Ciao Vincenzo,

La tua password è stata modificata correttamente.

Se non sei stato tu, contatta subito il nostro supporto.



Figura 25 - esempio e-mail 2

Pannello di amministrazione

- Visualizzazione complessiva delle prenotazioni: dal proprio pannello l'admin può visualizzare nel dettaglio tutte le prenotazioni e l'utente prenotato alla sessione disponibile.
- Gestione nuovi appuntamenti: l'admin attraverso il pannello dedicato può aggiungere nuovi appuntamenti

Requisiti non funzionali

Scalabilità

- Il sistema deve poter gestire un aumento del numero di utenti senza ridurre le performance.
- AWS RDS e S3 supportano la scalabilità automatica in base al carico.
- Possibilità di implementazione: è possibile implementare ulteriori servizi che per scelte economiche non sono stati implementati

Sicurezza

- Possibilità di Comunicazioni cifrate con HTTPS: attraverso Nginx è possibile configurare la porta 443 per le comunicazioni in HTTPS.
- Crittografia automatica dei file su S3 (SSE-S3).

- Accesso controllato tramite IAM per i servizi: sono stati definiti più utenti come identità per gestire i diversi servizi disponibili
- Autenticazione basata su token JWT lato backend.

Disponibilità

- Il sistema può essere disponibile 24/7: attraverso la configurazione su EC2 è possibile configurare l'esecuzione del sistema 24/7
- RDS replica i dati su più zone di disponibilità (alta affidabilità).
- I documenti caricati su S3 sono protetti e disponibili attraverso URL temporanei.

Performance

- Tempi di risposta ridotti (< 1/2 secondi medi): sono stati effettuati dei test sulla velocità e di carico dei servizi disponibili (ad esempio upload contemporaneo di più documenti da più utenti).
- Bucket S3 garantisce bassa latenza nel recupero dei documenti e nella distribuzione.

Usabilità

- Interfaccia web semplice e intuitiva: sono state sviluppate poche pagine per permettere una visualizzazione abbastanza semplificata e immediata.
- Possibilità di accedere da PC e dispositivi mobili: effettuati test sulla visualizzazione mobile dell'applicazione web ed effettuati controlli sulla ricezione dell'indirizzo IP con l'istanza del servizio in esecuzione

Portabilità

- L'applicazione è sviluppata in Node.js + MySQL, quindi facilmente distribuibile anche su altri ambienti

Costo

- Consumi bassi o quasi nulli per i servizi adoperati
- Possibilità di espandere con ulteriori servizi più specifici

Mappatura con i servizi AWS

Calcolo e Cloud:

- Backend Node.js e front end distribuito su Amazon EC2 con Elastic IP

Database (Storage strutturato):

- Amazon RDS con MySQL per la gestione delle prenotazioni e degli utenti.

Archiviazione (Storage oggetti):

- Amazon S3 per i documenti caricati dai clienti.

Sicurezza:

- IAM per gestione permessi e accesso programmatico.
- Crittografia lato server (SSE-S3).

Networking:

- Accesso sicuro tramite endpoint pubblici e restrizioni di sicurezza (Security Groups).

Link progetto

Il codice sorgente del progetto è visualizzabile:

<https://github.com/vincenzosilva96/legal-appointments>

Indirizzo pubblico della web application è:

http://56.228.51.212