2025

PROGETTO 32
Contenitore di evento per i ricercatori

GRUPPO
JEREMIE FWAPA KORANGI
PARFAIT WESILAKENDA KUBIKULAVE
STERLING MOMO NGUIMASTA

Indice

Capitolo 1

Introduzione

Come introdotto nel corso di ingegneria del software, lo scopo principale di questo lavoro è di applicare tutti i concetti imparati a lezione per realizzare un software talmente grande e complesso che sarà realizzato dalla squadra Team PSJ. L'obiettivo principale è quindi di analizzare, di progettare, di implementare e di testare un software, di preciso un'applicazione web, avendo lavorato con questa tipologia di software.

Nel nostro caso, ci è richiesto di creare un contenitore di evento per i ricercatori, si vuole sviluppare una applicazione web per la diffusione di eventi legati alla ricerca. In particolare, si vuole creare un portale di eventi, tipo conferente o workshop, che i ricercatori o altri entri creano e vogliono diffondere. Agli eventi possono partecipare i ricercatori iscritti alla piattaforma. Si deve poter creare un evento con tutti i suoi dettagli: data, luogo, programma, etc... Nel caso in cui ci si iscrive ad un evento, il ricercatore riceve tutti gli aggiornamenti sul suo cruscotto.

Per migliorare l'interazione con l'app, abbiamo deciso di darle un nome, EventApp.

1.1 EventApp

Come anticipato nell'introduzione, abbiamo deciso di chiamarla EventApp, essendo un software, dovrebbe avere un nome per essere identificato nel mercato e per favorire principalmente questa fase di sviluppo per non richiamarla ogni volta con la sua descrizione "contenitore di evento per i ricercatori".

EventApp si tratta di un'applicazione molto complessa che sarà sviluppata con tutte le nuove tecnologie che abbiamo a disposizione al giorno di oggi e soprattutto grazie alla maggiore flessibilità di queste tecnologie che cercano sempre di migliorarsi e produrre delle versioni più robuste e sofisticate.

Capitolo 2

Analisi dei requisiti

L'analisi dei requisiti è quel processo di comprensione di ciò che è richiesto a un sistema. Sviluppare un'applicazione web non significa scrivere delle righe di codice, ma ha un altro significato che gli sviluppatori considerano come la base dello sviluppo. Si tratta di scrivere più o meno formale e rigorosa di una caratteristica del sistema.

La gestione dei requisiti (acquisizione, analisi, negoziazione, specifica, validazione) è il primo passo del processo di sviluppo e una delle fasi più critiche dello sviluppo software, perché influenza in modo diretto il successo o il fallimento dei progetti.

2.1 L'idea

In questo progetto, si vuole realizzare un'applicazione web based che permetta ai ricercatori ed altri enti di creare degli eventi che vogliono diffondere. Lo scenario tipico è quello di avere una pagina principale che mostra tutti gli eventi ai quali possono iscriversi delle tipologie di utenti, questi eventi vengono creati dagli utenti che si registrano nella piattaforma. Questi utenti sono principalmente ricercatori ma anche altri enti. Hanno a disposizione una pagina personale dove possono visualizzare, inserire, modificare e rimuovere degli eventi. In piu, devono ricevere degli aggiornamenti sugli eventi in cui sono iscritti. Insomma, l'applicazione deve consentire l'inserimento, la visualizzazione, la modifica e la rimozione degli eventi, fornendo anche aggiornamento in tempo reale, e permettendo a tutte le tipologie di utenti di iscriversi agli eventi che possono essere creati solo dai ricercatori ed altri enti.

Il nome della web-app sarà "EventApp".

Le funzionalità richieste sono:

- **Visualizza** eventi
- **Registrazione** dell'account (ricercatori o altri enti)
- **Login** dell'utente nel sistema
- **Iscrizione** ad un evento
- **Ricerca** di un evento tramite hashtag
- **Inserimento**, **modifica** e **rimozione** degli eventi
- **Notifica** sugli eventi iscritti
- **Logout** dell'utente dall'applicazione

Esempio. Un ricercatore sta organizzando un evento che vorrebbe diffondere, ma non ha conoscenza dell'applicazione. Entra in EventApp, e visualizza tutti gli eventi che gli altri ricercatori hanno già diffuso, ma si registra prima nella piattaforma e poi fa l'accesso e crea un evento che poi pubblica nell'applicazione. E poi potrebbe iscriversi ad un altro evento creato da un altro utente.

2.2 Glossario

TERMINE	DESCRIZIONE	SINONIMI
Home Page	Pagina principale dell'applicazione dove è visualizzata le tre azioni importanti da fare	Pagina principale
Ricercatore	Attore del sistema	Ricercatore, utente
Atri enti	Attore del sistema	Altri enti,
Login	Azione dell'utente per essere autenticato nel sistema	Accesso al sistema, connessione, autenticazione
Logout	Azione dell'utente per uscire dal sistema	Uscita dal sistema, disconnessione
Registrazione	Azione di creare un conto nel sistema	Registrarsi, Creazione conto
Visualizzazione eventi	Lista degli eventi sulla home page	Manage event
Hashtag	Filtro che consente di visualizzare gli eventi di un determinato tipo	filtro
Password	Sequenza di caratteri alfanumerici scelta dall'utente per accedere al sistema	Parola segreta

2.3 Requisiti funzionali

I requisiti funzionali descrivono le funzionalità ed i servizi forniti dal sistema (cosa deve essere fatto). Nel ciclo di sviluppo software i requisiti funzionali rappresentano i casi d'uso. Di seguito sono riportate nel dettaglio le funzionalità richieste.

Registra account

Rappresenta la pagina di benvenuto dell'applicazione. Ogni utente che intende utilizzare l'applicazione per diffondere gli eventi, che sia ricercatore o utenti, deve essere registrato nel database. Al primo accesso all'applicazione, per poter fare il Login nel sistema, l'utente deve inserire dei dati personali di registrazione:

- Nome utente
- Indirizzo e-mail
- Password
- Conferma password
- Tipologia utente (ricercatori o altri enti)

Una volta che l'utente ha inserito i suoi dati, il sistema li memorizza nel database per poterli utilizzare successivamente in fase di login. Per l'autenticazione nel sistema, l'utente dovrà inserire indirizzo e-mail e password.

Login

Questa funzionalità permette al sistema di autenticare l'utente tramite l'account creato precedentemente nel caso d'uso registra account.

In fase di login, il sistema mostra all'utente i campi in cui inserire indirizzo mail e password, verifica la correttezza dei dati inseriti confrontandoli con i dati presenti nel database.

Se i dati inseriti risultano corretti, viene autenticato nel sistema e può usare l'applicazione, se i dati non sono corretti, il sistema propone all'utente di modificarli.

Visualizza eventi

Questo requisito è uno dei più importanti perché permette a tutti gli utenti dell'applicazione, di iscriversi ad un evento anche senza essere registrato nella piattaforma. La visualizzazione degli eventi deve prevedere una vista:

1. Il sistema deve prevedere una sezione "hashtag" da cui è possibile visualizzare la tipologia degli eventi che scelta dall'utente

Iscrizione

L'utente che ha scelto l'evento per cui vuole iscriversi ha a disposizione dei dati che vengono generati dal sistema, tra i quali: il titolo, la data e l'ora, il luogo dell'evento, la descrizione.

Inserimento – Modifica -Rimozione

L'utente che si è registrato nel sistema, per diffondere un evento, ha la possibilità di aggiungerlo, di modificarlo e di rimuoverlo

Ricerca con l'hashtag

Un qualsiasi utente dell'applicazione deve essere capace di trovare un evento specifico attraverso l'uso della barra di ricerca proposta dal sistema con l'hashtag che corrisponde all'evento per il quale corrisponde in funzione dei dati inseriti dal creatore dell'evento stesso.

Notifica sugli eventi

Dopo essere iscritto ad un evento, i ricercatori, gli altri enti e gli utenti esterni devono ricevere dal sistema, una notifica di aggiornamento in tempo reale dell'evento per il quale sono iscritti.

Logout

Il sistema deve fornire la procedura di uscita dall'applicazione. Effettuando il logout, l'utente viene scollegato dall'applicazione.

2.4 Requisiti non funzionali

I requisiti non funzionali non sono collegati direttamente con le funzioni implementate dal sistema, ma piuttosto alle modalità operative, di gestione. Definiscono vincoli sullo sviluppo del sistema. In seguito, verranno descritti tali requisiti.

Responsive web design

Nello sviluppo dell'applicazione è richiesto l'utilizzo della tecnica di web design responsive, in modo che le pagine web adattino automaticamente il layout per fornire una visualizzazione

ottimale in funzione dell'ambiente nei quali vengono visualizzati: pc, tablet, smartphone sono i principali.

Durata sessione

Se l'utente non esegue alcuna azione nel sistema, la sessione deve interrompersi dopo 10minuti, quindi scollegare l'utente.

Visualizzazione degli eventi

Gli eventi vengono elencati in una lista in ordine random con tutte le informazioni sull'evento (data, programma, luogo...)

Linguaggio di markup

Il linguaggio di markup da utilizzare è HTML5

Database

Le informazioni principali da salvare nel database riguardano gli account degli utilizzatori del sistema e gli eventi.

Lingua

L'applicazione deve essere in lingua italiana.

Sicurezza

Il sistema deve prevedere delle norme di sicurezza:

- Le password salvate nel database devono essere criptate
- Controllare le variabili che arrivano dai form per evitare attacchi di SQL Injection, Form injection, Variable injection.

Concorrenza

L'applicazione deve prevedere l'utilizzo di più account contemporaneamente, e uno stesso account deve poter essere utilizzato da più utenti allo stesso momento.

2.5 Casi d’uso

Un caso d’uso specifica cosa ci aspetta da un sistema (“what?”) ma nasconde il suo comportamento (“how?”). È una sequenza di azioni, con varianti, che producono un risultato osservabile da un attore e rappresenta un requisito funzionale. Ogni sequenza (detta scenario) rappresenta l’interazione di entità esterne al sistema (attori) con il sistema stesso o sue componenti. Il flusso principale degli eventi viene separato dalle varianti alternative.

L’attore rappresenta un soggetto o un’entità che non fa parte del sistema, ma interagisce in qualche modo con esse. Individua un ruolo che l’utente ricopre nell’interagire con il sistema. Gli attori eseguono i casi d’uso.

Nel nostro caso, l’attore del sistema è l’utente (Ricercatori, Altri enti, utente esterno), i casi d’uso sono: Visualizzazione eventi, Login, Registra account, Iscrizione, Inserimento-modifica-rimozione, Ricerca con l’hashtag, Notifica, Logout.

2.5.1 Diagramma dei casi d’uso

Il diagramma dei casi d’uso è il modello che descrive i requisiti del sistema in termini di funzionalità (casi d’uso) e ambiente circonstante (attori). Mostra cosa deve fare il sistema.

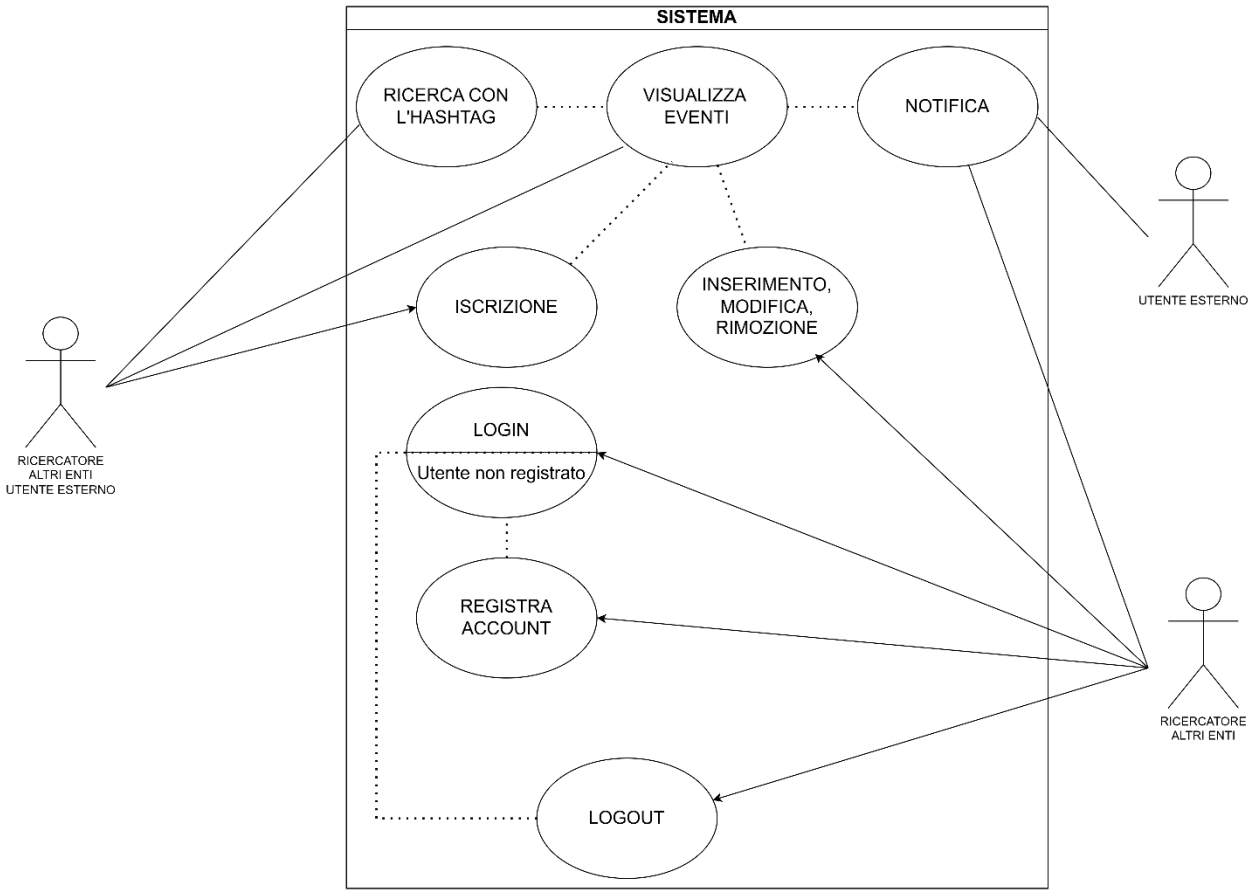


Figura 1: Diagramma dei casi d'uso

Di seguito vengono descritte le specifiche dei casi d'uso con scenari principali e alternativi. Uno scenario è una sequenza di passi che descrivono l'interazione tra un sistema e un attore (che dovrebbe trarre vantaggio dall'interazione).

2.5.2 Registra account

- **Nome caso d'uso**: Registra account
- **Id**: UC1
- **Attori**: Utente (ricercatore o altri enti o utente esterni)
- **Precondizioni**
  - L'utente non è registrato
- **Scenario principale**
  - L'utente accede per la prima volta al sistema
  - Il sistema visualizza la scelta tra Login e Registra account
  - L'utente sceglie Registra account

    - Il sistema visualizza i campi da compilare per registrare l'account utente
    - L'utente inserisce i dati di registrazione
    - Il sistema verifica la correttezza dei dati
        - Se i dati non sono corretti, il sistema evidenza i campi da modificare e ritorna al punto in cui visualizza i campi da compilare
    - Se i dati sono corretti, il sistema conclude la fase di registrazione
    - L'utente può effettuare il login
- **Postcondizioni:**
    - L'utente è registrato
- **Scenario secondario:**
    - L'utente accede al caso d'uso registra account
    - Il sistema visualizza i campi da compilare per la registrazione dell'account
    - L'utente inserisce i dati
    - Il sistema verifica che i dati sono già presenti nel database
    - Il sistema avvisa l'utente che i dati inseriti sono già stati utilizzati
- **Postcondizioni:**
    - Inizia il caso d'uso Login

2.5.3 Login

- **Nome caso d'uso**: Login
- **Id**: UC2
- **Attori**: Utente (ricercatore o altri enti)
- **Precondizioni**
    - L'utente ha avviato il sistema
- **Scenario principale:**
    - Il caso d'uso inizia quando l'utente avvia l'applicazione da un browser
    - Il sistema visualizza i campi dove inserire indirizzo mail e password dell'utente
    - L'utente inserisce indirizzo mail e password dell'utente
    - Il sistema verifica se indirizzo mail e password sono corretti
        - Se non sono corretti, si ritorna al punto dove visualizza i campi
    - Se indirizzo mail e password sono corretti, il sistema visualizza lo spazio utente
- **Postcondizioni:**
    - L'utente è registrato nel sistema
- **Scenario secondario:**
    - L'utente non è registrato nel sistema
    - Il sistema avvia il caso d'uso Registra account
- **Postcondizioni:**
    - Inizia il caso d'uso Registra account

2.5.4 Visualizza eventi

- **Nome caso d'uso**: Visualizza eventi
- **Id**: UC3
- **Attori**: Ricercatori, altri enti e utenti esterni
- **Precondizioni**:
    - L'utente ha fatto accesso all'applicazione
- **Scenario principale:**
    - Il sistema mostra la lista degli eventi in modo random disponibili per un certo periodo
    - L'utente visualizza tutti questi eventi
- **Postcondizioni**
    - L'utente può selezionare un evento specifico

2.5.5 Iscrizione

- **Nome caso d'uso**: Iscrizione
- **Id**: UC4
- **Attori**: Ricercatori, altri enti e utenti esterni
- **Precondizioni**
    - L'utente ha scelto l'evento per cui vuole fare l'iscrizione
- **Scenario principale**
    - Il caso d'uso inizia quando l'utente ha realmente scelto per quale evento vuole partecipare
    - Il sistema chiede all'utente se non è connesso di connettersi o di continuare compilando una sezione dei dati
    - Il sistema elabora la domanda ed invia una mail di avvenuta iscrizione all'evento
- **Postcondizioni**:
    - L'utente può visualizzare, modificare o cancellare l'iscrizione e ricevere aggiornamento

2.5.6 Inserimento – Modifica – Rimozione

- **Nome caso d'uso**: Inserimento, Modifica, Rimozione
- **Id**: UC5
- **Attori**: Ricercatori o altri enti
- **Precondizioni**:

    - Non esiste nessun evento
- **Scenario principale**
    - Il caso d'uso inizia quando l'utente si è connesso
    - Il sistema permette di aggiungere un evento
    - L'utente sceglie questa azione
    - Il sistema controlla la correttezza dell'operazione e procede
- **Postcondizioni**
    - Esiste un evento
- **Scenario secondario**
    - L'utente vuole modificare o rimuovere un evento esistente
    - Il sistema chiede che azione vuole fare
    - L'utente sceglie un'azione
    - Il sistema controlla la correttezza dell'operazione e procede

2.5.7 Ricerca con hashtag

- **Nome caso d'uso**: Ricerca con hashtag
- **Id**: UC6
- **Attori**: Utente
- **Precondizioni**
    - L'utente ha fatto accesso alla pagina di visualizzazione degli eventi
- **Scenario principale**
    - Il caso d'uso inizia quando l'utente vuole ha accesso alla pagina
    - Il sistema apre una barra di ricerca che permette di inserire l'hashtag
    - L'utente digita l'hashtag
    - Il sistema rilascia gli eventi che corrispondo all'hashtag richiesto
- **Postcondizioni**
    - Il sistema non trova l'evento ricercato

2.5.8 Notifica

- **Nome caso d'uso**: Notifica
- **Id**: UC7
- **Precondizioni**:
    - L'utente è già connesso al sistema
- **Scenario principale:**
    - Il caso d'uso inizia quando l'utente è connesso
    - Il sistema fornisce all'utente un aggiornamento dell'evento
- **Postcondizioni**:

    - L’utente non è connesso al sistema
- **Scenario secondario**
    - L’utente fa accesso al suo account
    - Il caso d’uso inizia
    - Il sistema fornisce all’utente un aggiornamento in tempo reale sull’evento

2.5.9 Logout

- **Nome caso d’uso**: Logout
- **Id**: UC8
- **Attori**: Utente (ricercatori o altri enti)
- **Precondizioni**
    - L’utente ha effettuato il login
- **Scenario principale**:
    - Il caso d’uso inizia quando l’utente vuole uscire dal sistema
    - L’utente chiede al sistema di disconnettersi
    - Il sistema disconnette l’utente
- **Postcondizioni**
    - L’utente è disconnesso dal sistema

2.6 Diagramma delle attività

Il **diagramma delle attività**, in UML, è un diagramma di flusso (con alcuni elementi aggiuntivi) che mostra una sequenza di attività. Viene utilizzato per rappresentare i passi (le transazioni) che compongono il flusso di un caso d’uso, descrivono quindi il comportamento dinamico di un sistema. La rappresentazione delle attività è comoda in quanto consente di rappresentare sinteticamente flusso principale e flussi alternativi. Il diagramma in figura2 mostra il flusso delle attività dell’utente.

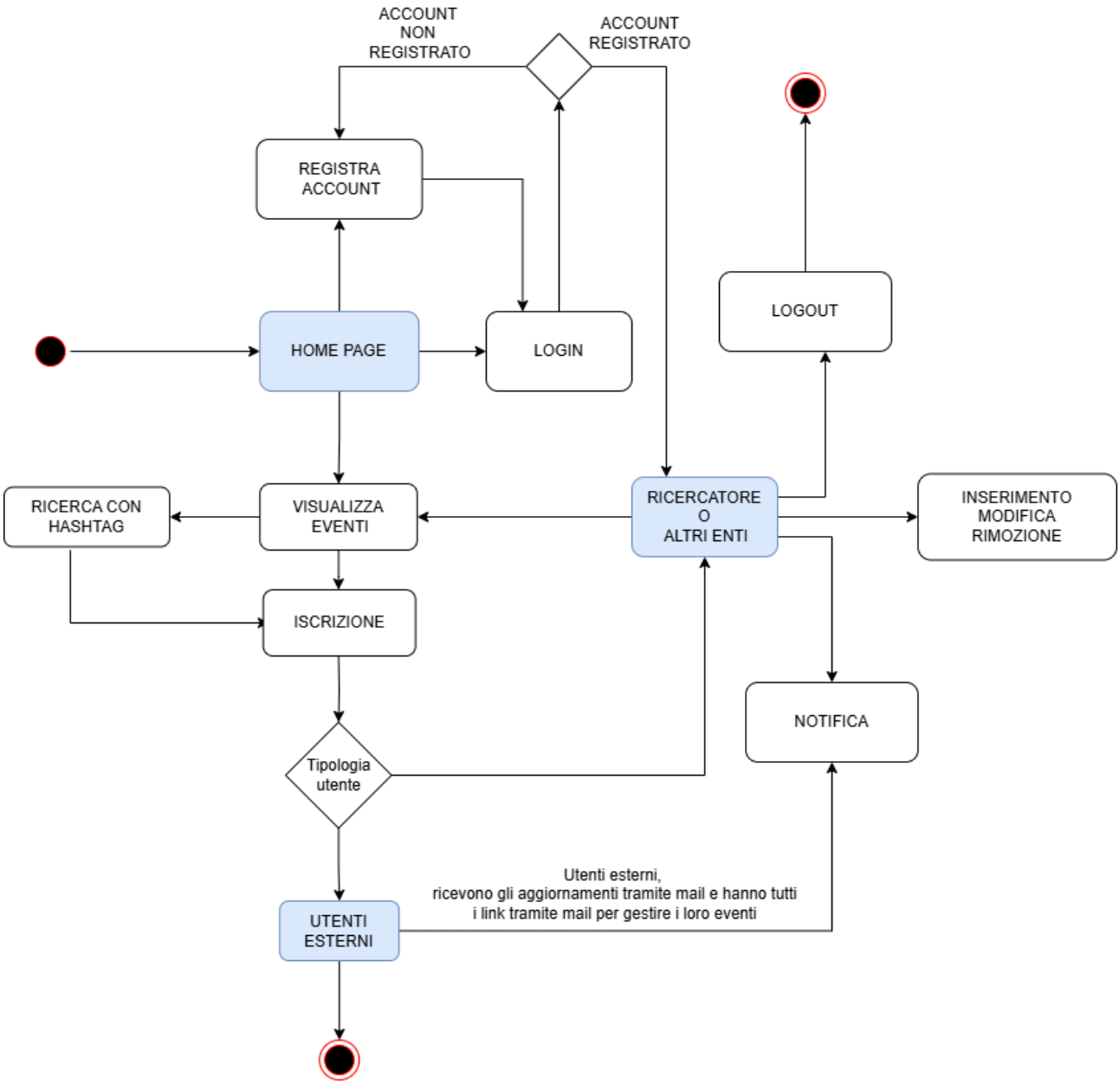


Figura 2 - Diagramma delle attività

2.7 Diagramma di stato

Il **diagramma di stato** mostra gli stati principali del sistema e come si passa da uno stato all'altro in

base alle azioni degli utenti.

Flusso degli stati:

2.7.1 Stato Iniziale: L'applicazione si avvia con un utente (può essere un ricercatore, un

utente esterno, o altri enti)

2.7.2 Registra account/Login: L'utente (ricercatori, altri enti) si autentica nel sistema. Se

avviene con successo, questo gli da accesso alla sua Dashboard.

2.7.3 Visualizza Eventi: Qualunque utente può visualizzare gli eventi pubblicati, ma solo i ricercatori o altri enti possono visualizzare eventi creati da loro e anche interagire con eventi iscritti (cancellarli) nella dashboard.

2.7.4 Iscriversi: L'utente può visualizzare il contenuto di un evento e iscriversi a quel evento.

2.7.5 Inserimento/Modifica/Rimozione: L'utente interagisce con il sistema per creare, modificare o cancellare un evento nella dashboard.

2.7.6 Notifica: Il sistema aggiorna l'utente con notifiche ogni volta che questo ultimo effettua un'azione.

2.7.7 Logout: L'utente può terminare la sessione.

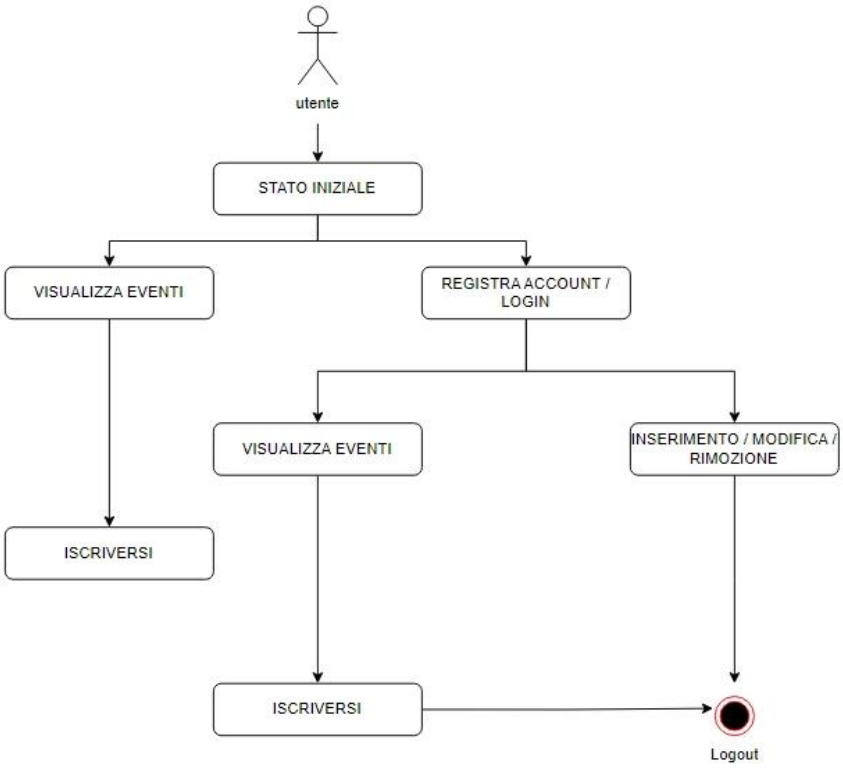


Figura 3 - Diagramma di stato

2.8 Diagramma di sequenza

Il diagramma di sequenza descrive nel dettaglio il flusso delle interazioni tra utenti e sistema,

indicando chi fa cosa e come il sistema risponde. illustra le interazioni tra gli attori (ricercatori, utenti esterni) e il sistema in uno specifico scenario, come la registrazione, la ricerca di un evento,

l'iscrizione a un evento o la gestione degli eventi.

Descrizioni:

2.8.1 Login o Registrazione:

- L'utente inserisce le credenziali.
- Il sistema verifica le informazioni nel database.
- Accesso confermato.

2.8.2 Ricerca Evento:

- L'utente digita un hashtag.
- Il sistema invia una query al database.
- Gli eventi vengono restituiti e visualizzati.

2.8.3 Iscrizione:

- L'utente seleziona un evento, lo visualizza poi si iscrive.
- Il sistema aggiorna il database con la nuova iscrizione.
- Notifica inviata all'utente.

2.8.4 Gestione eventi: Inserimento/modifica/rimozione di un evento

- L'utente (ricercatore o ente) accede alla sezione di gestione eventi
- L'utente sceglie di inserire, modificare o rimuovere un evento
- Se inserisce un evento, inserisce tutti i dettagli richiesti
- Se modifica, aggiorna i dettagli dell'evento
- Se rimuove un evento, il sistema lo rimuove dal database

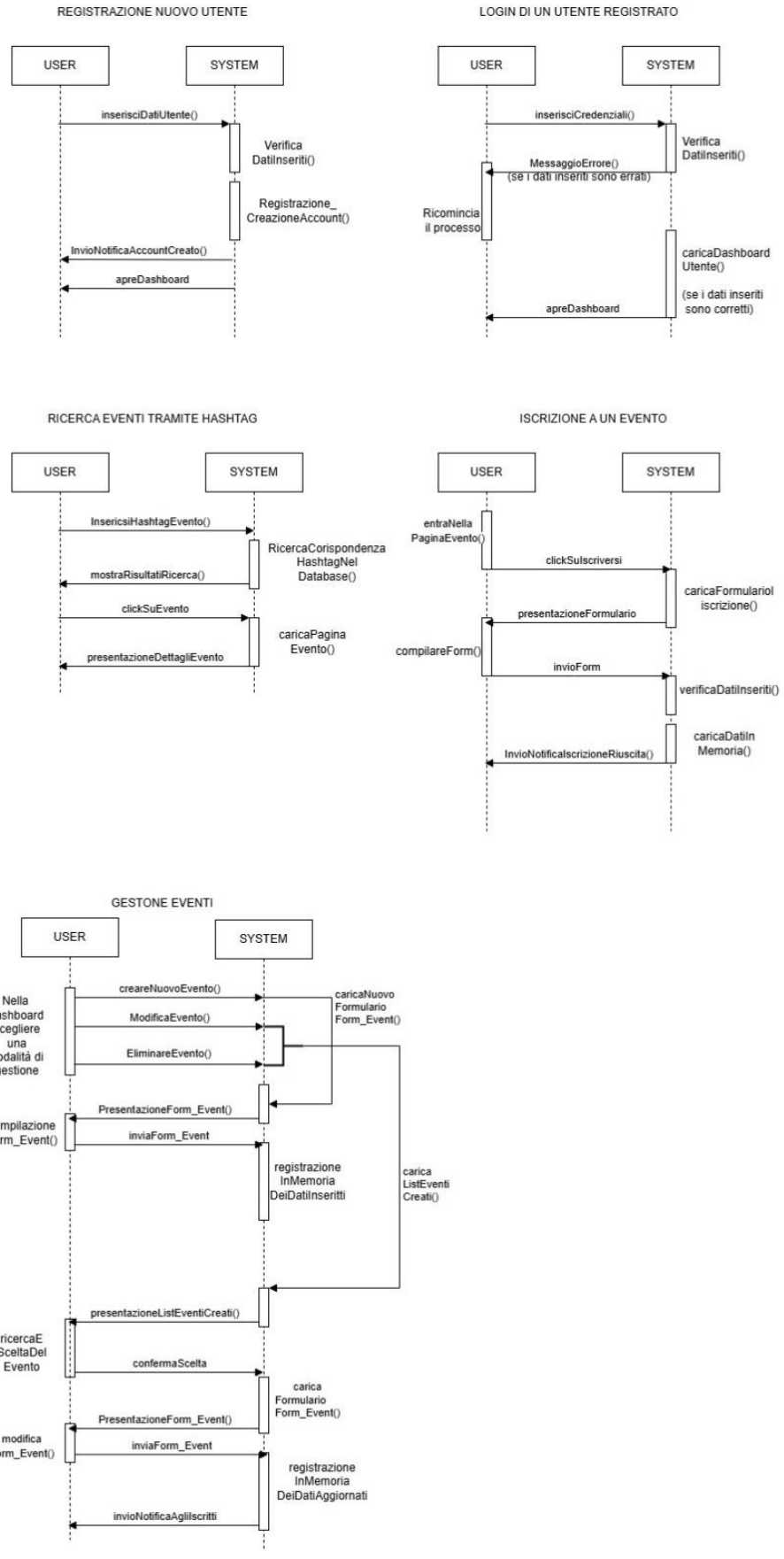


Figura 4 - Diagramma di sequenza

2.9 Diagramma di comunicazione

Il **diagramma di comunicazione** rappresenta l'interazione tra gli oggetti del sistema, mostrando i messaggi scambiati per realizzare le principali funzionalità del progetto "Contenitore di Eventi". Questo diagramma è utile per comprendere il flusso delle informazioni tra i componenti e garantire un'architettura chiara e modulare.

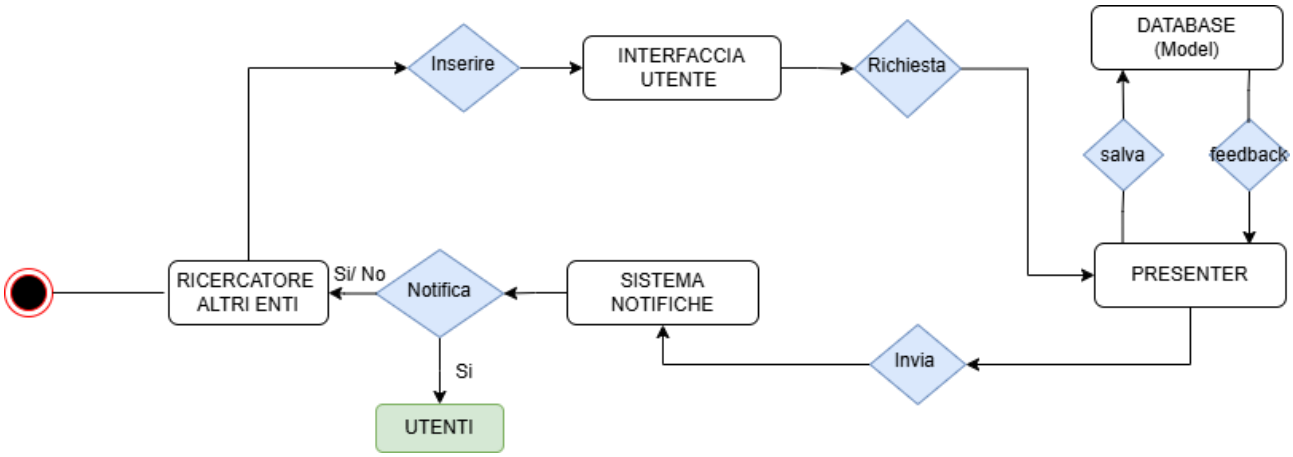


Figura 5 - Diagramma di comunicazione

2.10 Diagramma di classe

Mostra le classi del sistema, con attributi, metodi e relazioni tra loro.

- Aiuta a: Definire la struttura dati e le responsabilità delle classi.

Nel nostro progetto:

Definisce entità come Utente, Ricercatore, Evento, Hashtag, Notifica.

Specifica associazioni tra le classi (es. un Utente può partecipare a più Eventi).

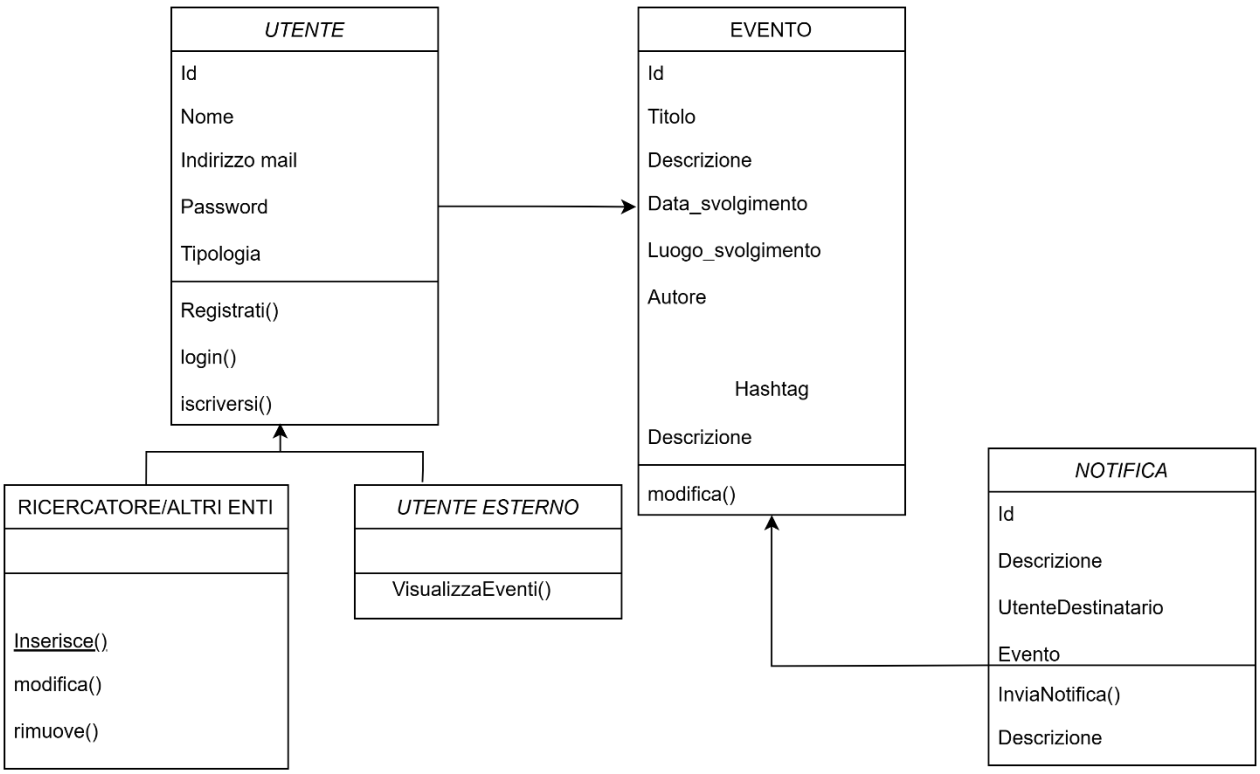


Figura 6 - Diagramma di classe

Capitolo 3

Progettazione

In questa fase si definisce l'architettura software su cui su baserà la realizzazione del prodotto EventApp. Abbiamo scelto una strutturazione tipica su tre livello logico-funzionali ma che possono essere distribuiti anche su più livelli.

L'applicazione web Three-Tier si sviluppa su 3 livelli:

- Presentazione (cliente): costituisce l'interfaccia utente dell'applicazione e corrisponde al browser web. Le tecnologie scelte da utilizzare sono HTML e CSS

- Applicazione (Servente Web): è il livello logico della web-app, la componente elaborativa. Dal lato server (Server-side) utilizza la tecnologia PHP, dal lato client (Client-side) utilizza il linguaggio di scripting Javascript.

- Dati (Servente RDBMS): consente di modellare e gestire il contenuto informativo dell'applicazione. La tecnologia usata a questo livello è il Database relazione MySQL.

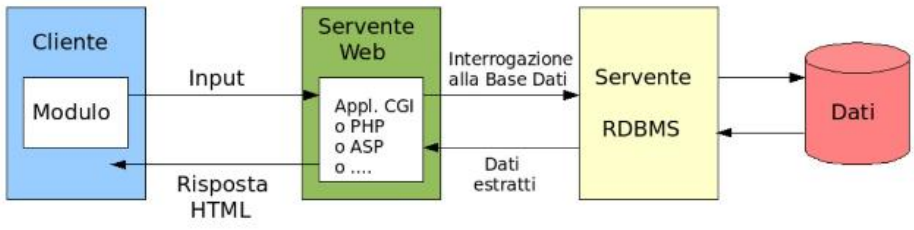


Figura 7: Struttura web application

L'applicazione ha una architettura di comunicazione Client-Server di tipo MVP REST-full API CRUD.

Gli strumenti che abbiamo utilizzato sono:

Docker è uno strumento per la creazione, la gestione e l'orchestrazione di più container di applicazioni. Il suo obiettivo è quello di ottimizzare il processo di sviluppo, test, consegne e distribuzione delle applicazioni, mediante la creazione di pacchetti di ogni componente dell'applicazione in un contenitore separato.

Nel nostro caso, l'abbiamo utilizzato per il lato server installando tre immagini:

- **Web**: php:8.2-apache, questa immagine ci consente di lanciare la nostra applicazione come se fosse un server reale
- **Database**: mysql:8.1.0, ci consente di creare e di gestire il database
- **Phpmyadmin**: phpmyadmin/phpmyadmin, consente di gestire php e di avviare lo strumento di gestione di MySQL

Si accede all'applicazione, collegandosi all'indirizzo http://localshot, per accedere invece a phpmyadmin, http://localhost:8080

3.1 Tecnologie utilizzate

In questa sezione sono descritte in dettaglio le tecnologie utilizzare per la realizzazione dell'applicazione. Lo studio di esse ha occupato una parte rilevante del nostro lavoro in quanto sono tecnologie di nuova generazione.

HTML

L'**HyperText Markup Language** (lett. "linguaggio di marcatura d'ipertesto"), comunemente noto con l'acronimo **HTML**, è il linguaggio di marcatura più usato per i documenti web. Nato per la formattazione e impaginazione di documenti ipertestuali disponibili nel web 1.0, oggi è utilizzato principalmente per il disaccoppiamento della struttura logica di una pagina web (definita appunto dal markup) e la sua rappresentazione, gestita tramite gli stili CSS per adattarsi alle nuove esigenze di comunicazione e pubblicazione all'interno di Internet.[1]

CSS

Cascading Style Sheets, meglio noto come **CSS** (in italiano **fogli di stile a cascata**), è un linguaggio usato per definire la formattazione di documenti HTML, XHTML e XML, ad esempio i siti web e relative pagine web. Le regole per comporre il CSS sono contenute in un insieme di direttive (*Recommendations*) emanate a partire dal 1996 dal W3C.

JAVASCRIPT

JavaScript è un linguaggio di programmazione multi paradigma orientato agli eventi, utilizzato sia nella programmazione lato client web che lato server (Node.js) per la creazione di RESTful API, applicazioni desktop e embedded, siti web e applicazioni web, di effetti dinamici interattivi tramite funzioni di script invocate da *eventi* innescati a loro volta in vari modi dall'utente sulla pagina web in uso (mouse, tastiera, caricamento della pagina ecc...).

PHP

PHP (acronimo ricorsivo di "PHP: HyperText Preprocessore", preprocessore di ipertesti; originariamente acronimo di "Personal Home Page") è un linguaggio di scripting interpretato, originariamente concepito per la programmazione di pagine web dinamiche. L'interprete PHP è un software libero distribuito sotto la licenza PHP.

DOCKER

Docker è un popolare software libero che utilizza la virtualizzazione a livello di sistema operativo per eseguire applicazioni in ambienti isolati chiamati container.

WEB SERVER APACHE

Apache è il nome di un server web libero sviluppato dalla Apache Software Foundation. È la piattaforma server Web modulare più diffusa, in grado di operare su una grande varietà di sistemi operativi, tra cui UNIX/Linux, Microsoft Windows e OpenVMS. È un software che realizza le funzioni di trasporto delle informazioni, di internet work e di collegamento, ed ha il vantaggio di offrire funzioni di controllo per la sicurezza come quelle effettuate da un proxy.

MYSQL

MySQL o **Oracle MySQL** è un *relational database management system* (RDBMS) composto da un client a riga di comando e un server. Ambo i costituenti sono multipiattaforma e sono disponibili ufficialmente su tutte le distribuzioni conosciute, quali Debian, Ubuntu e CentOS, sebbene lo abbiano sostanzialmente sostituito con MariaDB a partire dal 2012.

BOOTSTRAP

Bootstrap è una libreria di strumenti liberi per la creazione di siti e applicazioni per il Web. Essa contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia, come moduli, pulsanti e navigazione, così come alcune estensioni opzionali di JavaScript.

Software utilizzati

- Windows 11 Home x64: sistema operativo
- Docker: web server
- Visual studio: Ambiente di sviluppo
- DrawIO: per la realizzazione dei diagrammi UML
- Mozilla Firefox, Google Chrome, Opera, Internet Explorer: browser usati per ricerca, sviluppo e test.

3.2 Database

Quando i dati sono molti e salvarli su filesystem risulta inefficiente, bisogna usare il supporto di una base di dati, nel nostro caso l'abbiamo chiamata "contenitore".

L'analisi dei requisiti ha portato all'individuazione di quattro entità fondamentali: "Account", "Eventi", "Iscrizione" e "Notifica"". Le entità e le associazioni che intercorrono tra esse sono rappresentate nel seguente schema Entity/Relationship.

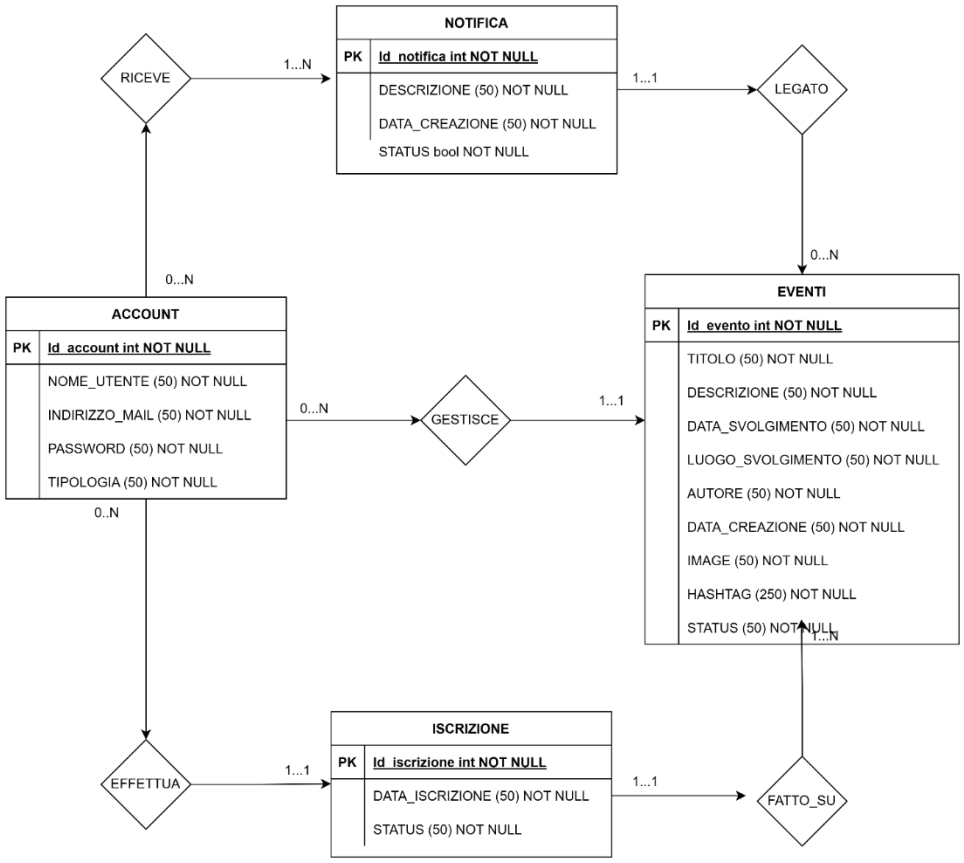


Figura 8: Schema Entity/Relationship

Lo schema logico ottenuto è il seguente:

ACCOUNT (id_account, nome_utente, indirizzo_mail, password, tipologia)

EVENTI (id_evento, titolo, descrizione, data_svolgimento, luogo_svolgimento, autore, data_creazione, image, status)

ISCRIZIONE (id_iscrizione, data_iscrizione, status, account, evento)

NOTIFICA (id_notifica, descrizione, data_creazione, status, account, evento)

Nelle sezioni seguenti verrà approfondita l'analisi delle entità, precisando il ruolo di ciascuna.

3.2.1 Entità "Account"

L'entità "Account" serve per tenere traccia di tutti gli utilizzatori del sistema (ricercatore, altri enti e utenti esterni). Il compito principale è la gestione delle credenziali per poter effettuare il login tramite indirizzo mail e password scelti dall'utente in fase di registrazione.

Gli attributi di Account sono i seguenti:

id_account serve a identificare univocamente un utente del sistema (chiave primaria)

nome_utente serve a conoscere il nome completo del proprietario del conto.

indirizzo_mail serve principalmente nella fase di login e nella fase di notifica tramite mail

password serve anche in fase di login per autenticare l'utente nel sistema. Questo campo

come detto nella fase di analisi, viene salvato nel database criptato per motivi di sicurezza

tipologia serve per conoscere di che tipologia è l'utente che si è registrato nel sistema

L'entità "Account" risulta la seguente:

ACCOUNT	
PK	Id_account int NOT NULL
	NOME_UTENTE (50) NOT NULL
	INDIRIZZO_MAIL (50) NOT NULL
	PASSWORD (50) NOT NULL
	TIPOLOGIA (50) NOT NULL

3.2.2 Entità "Eventi"

Questa entità ci permette di tenere traccia di tutti gli eventi che vengono aggiunti nell'applicazione.

Gli attributi di Eventi sono i seguenti:

id_evento serve per identificare in modo univoco l'evento

titolo serve per il titolo dell'evento

descrizione serve a descrivere l'evento

data_svolgimento serve a conoscere quando si terrà l'evento

luogo_svolgimento serve a conoscere invece l'ora dell'evento

autore serve a identificare l'id di chi ha creato l'evento

data_creazione tiene traccia dell'ora esatta della creazione dell'evento

image serve per aggiungere un'immagine legata all'evento

status ci da l'indicazione sulla validità dell'evento

L'entità risulta quindi la seguente:

EVENTI	
PK	**Id_evento int NOT NULL**
	TITOLO (50) NOT NULL
	DESCRIZIONE (50) NOT NULL
	DATA_SVOLGIMENTO (50) NOT NULL
	LUOGO_SVOLGIMENTO (50) NOT NULL
	AUTORE (50) NOT NULL
	DATA_CREAZIONE (50) NOT NULL
	IMAGE (50) NOT NULL
	STATUS (50) NOT NULL

3.2.3 Entità "Iscrizione"

L'entità 'Iscrizione' tiene traccia di tutte le iscrizioni effettuate dagli utenti.

Gli attributi di iscrizione sono i seguenti:

id_iscrizione serve a identificare univocamente le iscrizioni degli utenti

data_iscrizione serve a memorizzare la data dell'iscrizione ad un evento

status indica lo status dell'iscrizione, se è valido o no

L'entità risulta la seguente:

ISCRIZIONE	
PK	Id_iscrizione int NOT NULL
	DATA_ISCRIZIONE (50) NOT NULL
	STATUS (50) NOT NULL

3.2.4 Entità “Notifica”

Questa entità tiene traccia di tutte le notifiche che gli utenti del sistema ricevono in funzione dell’evento sul quale viene effettuate delle azioni.

Gli attributi di Notifica sono i seguenti:

id_notifica serve a identificare univocamente la notifica

descrizione fornisce un piccolo messaggio della notifica

data_creazione indica il momento della creazione della notifica

status indica la validità della notifica in funzione del fatto che l’utente lo apre o no.

L’entità risulta la seguente:

NOTIFICA	
PK	Id_notifica int NOT NULL
	DESCRIZIONE (50) NOT NULL
	DATA_CREAZIONE (50) NOT NULL

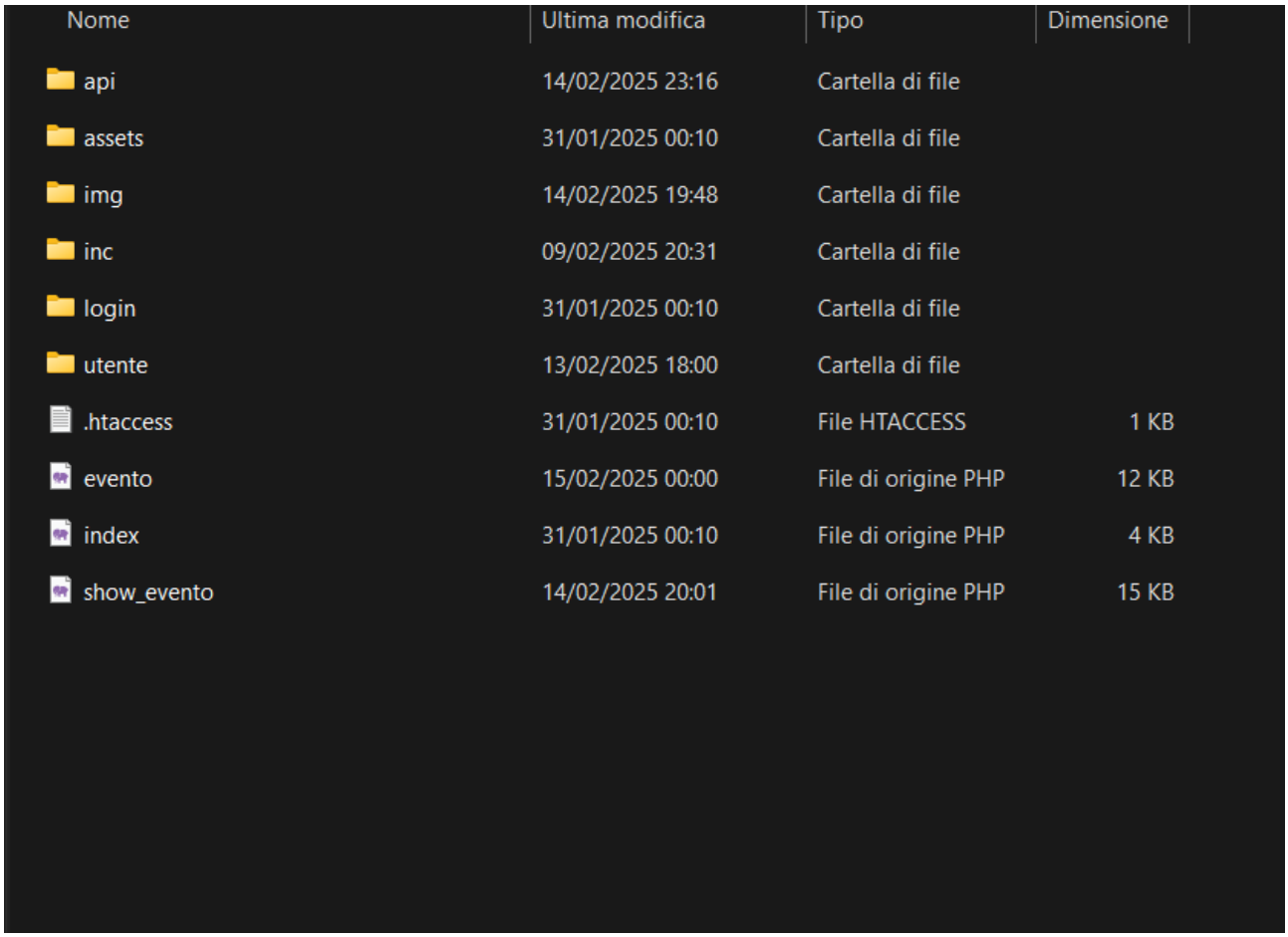
Capitolo 4

Implementazione e test

E la fase realizzativa. Ogni modulo, come unità indipendente, viene implementato e controllato per assicurarne la correttezza.

4.1 Struttura grafica dell'applicazione Event App

La directory principale dell'app si trova nella cartella html/ ed è costituita di dieci elementi (cartelle, file htaccess e file php). Il modello utilizzato per svilupparla è l'MVP (Model View Presenter). Per favorire la sua implementazione, è stato pensato la suddivisione in tre aspetti: il modello all'interno della cartella api/model, il presenter all'interno della cartella api/presenter e la view, ciò che vede l'utente.



Nome	Ultima modifica	Tipo	Dimensione
api	14/02/2025 23:16	Cartella di file	
assets	31/01/2025 00:10	Cartella di file	
img	14/02/2025 19:48	Cartella di file	
inc	09/02/2025 20:31	Cartella di file	
login	31/01/2025 00:10	Cartella di file	
utente	13/02/2025 18:00	Cartella di file	
.htaccess	31/01/2025 00:10	File HTACCESS	1 KB
evento	15/02/2025 00:00	File di origine PHP	12 KB
index	31/01/2025 00:10	File di origine PHP	4 KB
show_evento	14/02/2025 20:01	File di origine PHP	15 KB

Le pagine dell'applicazione sono principalmente file php. Ci sono quattro livelli in quest'applicazione, per favorire una buona lavorazione e scrittura del codice. Entrando nell'applicazione, l'attore che interagisce con l'app trova la home page (index.php), è il punto principale di ingresso che costituisce il primo livello. Il secondo livello è quello che permette all'attore di iniziare ad interagire con il sistema fornendo la possibilità di registrarsi e di fare login, costituiscono il secondo livello dell'applicazione e si trova nella cartella login. Il terzo livello è quello che permette all'attore di gestire personalmente i dati degli eventi inseriti e di vedere qualche statistiche.

A questo punto, forniamo un elenco di tutte le pagine fornite a tutti i livelli.

- **Index.php**: Home page dell'applicazione che chiede all'attore quale azione vuole compiere tra: Visualizzare gli eventi, fare login e registrarsi.
- **Evento.php**: Pagina che mostra tutti gli eventi disponibili.
- **Show_evento.php**: Pagina che permette all'attore di iscriversi all'evento selezionato;

All'interno della cartella login/

- **Index.php**: è la pagina principale che permette di fare login al proprio conto.
- **Registrati.php**: è la pagina che permette di registrarsi nell'applicazione

All'interno della cartella utente/

- **Index.php**: è la pagina di benvenuto agli utenti che hanno fatto login correttamente, che mostra direttamente la dashboard e il menu delle azioni.
- **Inserimento.php**: è la pagina che permette di inserire gli eventi.
- **Modifica.php**: mostra la lista degli eventi inseriti su cui si vuole effettuare delle modifiche
- **Rimozione.php**: mostra la lista degli eventi su cui si vuole effettuare delle rimozioni
- **Eventi_iscritti.php**: mostra la lista degli eventi su cui si è iscritto.
- **Eventi_inseriti.php**: mostra la lista degli eventi che l'utente ha inserito

E la cartella api/

- **Database.php**: contiene il file di configurazione dell'applicazione che collega l'app al database.
- **Evento.php**: è il route dell'applicazione, riceve le richieste e le reindirizza verso il presenter per elaborare
- **Model.php**: è responsabile del contatto con il database.
- **Presenter.php**: è il ponte tra model e view

Sono i file principali che girano sul server che costituiscono l'architettura dell'applicazione.

4.1.1 Funzioni rilevanti per la web-app

Quando si realizzano applicazione web è molto utile definire una volta per tutte alcune caratteristiche comuni a tutte le pagine dell'app oppure porzioni di codice da utilizzare in

punti diversi dell'app stessa. Alcuni di questi file sono contenuti all'interno delle cartelle inc/; I file sorgenti "head.php" al primo livello permette a tutte le pagine di disporre dell'head.

4.1.1.1 index.php

Il file "index.php" è la pagina di default richiamata quando viene digitata la URL o una specifica pagina del sito. Contiene delle funzioni Javascript fra cui: jquery e bootstrap che lo fanno girare. Fornisce principalmente tre pulsanti: visualizza eventi, login, registrati.

4.1.1.2 evento.php

In quest'applicazione, abbiamo utilizzato un'api restful CRUD, la funzione principale per comunicare con il server era la funzione fetch offerta da Javascript. Quest'applicazione comunica con il server tramite l'api per inviare i dati e riceverli in formato JSON. La chiamata di questa funzione si trova praticamente in tutte le pagine perché l'applicazione offre un'esperienza unica e dinamica e impedisce il caricamento abusivo delle pagine. Quindi è stato pensato all'elaborazione di un'api molto semplice che comprende un route per indirizzare le richieste effettuate dall'utente.

```
function getAllArticles() {
    if (isSearching) return; // Bloque la mise à jour si une recherche est en cours

    fetch('api/evento')
        .then(response => response.json())
        .then(articles => {
            const container = document.getElementById("getEvent");
            container.innerHTML = "";

            let numero = articles[0];
            document.getElementById("hours").textContent = numero.totale;

            articles.forEach(article => {
                const articleHTML = `
                    <div class="col-lg-4 col-md-6 col-sm-12">
                        <div class="event-card">
                            <div class="event-image">
                                <img src="img/${article.immagine}" alt="${article.titolo}">
                            </div>
                            <div class="event-body">
                                <h3 class="event-title">${article.titolo}</h3>
                                <div class="event-meta">
                                    <span><i class="lni lni-user"></i> ${article.nome_utente} | ${article.tipologia} </span>
                                    <span><i class="lni lni-map-marker"></i> ${article.luogo_svolgimento}</span>
                                    <span><i class="lni lni-calendar"></i> ${article.data_svolgimento}</span>
                                    <span class="event-hashtag">#${article.hashtag}</span>
                                </div>
                                <p class="event-description">${article.descrizione.substring(0, 100)}...</p>
                                <div class="event-footer">
                                    <a href="show_evento?id=${article.id_evento}">
                                        <button class="btn btn-primary btn-iscriversi" data-id="${article.id_evento}"> Iscriversi</button>
                                    </a>
                                </div>
                            </div>
                        </div>
                    </div>
                `;
                container.innerHTML += articleHTML;
            });
        })
        .catch(error => console.error("Erreur:", error));
}
```

In questa immagine, si vede la funzione getAllArticles contenuta all'interno di questa pagina, e di tutte le pagine che mostrano all'utente la lista degli eventi disponibili. Come funziona? Chiama l'api e invia una richiesta GET chiedendo al route di inviare una richiesta al presenter e di ritornarli sotto forma di file JSON che poi vengono trattati dalla view come si vede nell'immagine.

4.1.1.3 show_evento.php

Dopo aver visualizzato tutti gli eventi disponibili, l'utente ha la possibilità di iscriversi all'evento desiderato cliccando su "iscriversi" dalla pagina "evento", questo clic apre una pagina che riceve nell'url un id che identifica l'evento per il quale si vuole effettuare l'iscrizione. Una volta all'interno di questa pagina, si ha la possibilità di scegliere il tipo di utente che effettua l'iscrizione. Nel caso in cui l'utente non è connesso come ricercatore o altri enti, il sistema chiede se si vuole connettere alla piattaforma oppure iscriversi all'evento come utente esterno. Quando queste operazioni sono svolte, una richiesta di tipo POST viene inviata al server tramite route con i dati dell'utente che si iscrive mostrato sull'immagine

```
<script>
document.querySelector('.btn-register').addEventListener('click', function() {

    const userId = <?php echo $_SESSION['id_account']; ?>;
    console.log(eventId);

    if (userId && eventId) {
        fetch('../api/iscrizione', {
            method: 'POST',
            body: JSON.stringify({action: "iscrizione", id_account: userId, id_evento: eventId })
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                alert(data.message);
                window.location.href="evento";
            } else {
                alert(data.message);
            }
        })
        .catch(error => {
            alert('Errore nella richiesta: ' + error);
        });
    } else {
        alert('Errore: ID evento o ID utente non validi.');
    }
});
</script>
```

4.1.1.4 login/index.php

Questa pagina è quella mostrata quando l'utente decide di autenticarsi nella piattaforma, in altre parole, quando vuole registrarsi ad un conto che possiede. In funzione dei dati inseriti (indirizzo mail e password), il sistema verifica se l'utente esiste o no. La funzione che verifica questi dati

```
<script>
  document.addEventListener("DOMContentLoaded", function () {
  document.getElementById("signupForm").addEventListener("submit", async function (event) {
      event.preventDefault();

      const email = document.getElementById("indirizzo_mail").value.trim();
      const password = document.getElementById("password-field").value.trim();

      if (email === "" || password === "") {
          alert("Tutti i campi sono obbligatori !");
          return;
      }

      const response = await fetch("../api/utente", {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify({
              action: "login",
              indirizzo_mail: email,
              password: password,
          }),
      });

      const result = await response.json();

      if (result.success) {
          <?php if(!isset($_GET['id'])) : ?>
              alert("Bravissimo ! Verrai reindirizzato verso il dashboard.");
              window.location.href = "../utente/";
          <?php else: ?>
              alert("Bravissimo ! Puo procedere all'iscrizione all'evento desiderato !");
              window.location.href = "../show_evento?id=<?= $_GET['id']; ?>"
          <?php endif; ?>
      } else {
          alert("Errore : " + result.message);
      }
  });
);

/script>
```

La funzione mostra che il sistema recupera i campi al clic dell'utente, e invia una richiesta di tipo POST all'api con i dati digitati codificati in JSON. E poi l'api risponde, e i risultati vengono messi all'interno della variabile result. Reindirizza l'evento verso la dashboard se l'utente non proviene dalla pagina show_evento e reindirizza verso show_evento se proviene da li.

4.1.1.5 login/registrati.php

Una delle pagine importanti perché è quella che permette all'utente di entrare nel sistema e di prendere possessione di tutte le funzionalità offerte dal sistema. La fase principale è la verifica della correttezza dei dati inseriti, il nome utente deve essere valido, l'indirizzo mail

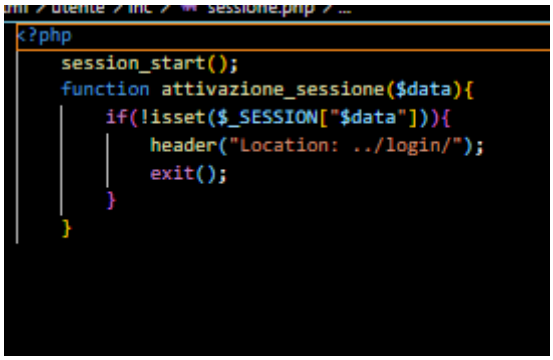
deve corrispondere a quello di un indirizzo standard con l'@ e la password non deve essere inferiore ai sei caratteri. Tutti questi controlli vengono effettuati da questa funzione

```
<script>
document.addEventListener("DOMContentLoaded", function () {
    document.getElementById("signupForm").addEventListener("submit", function (event) {
        event.preventDefault(); // Empêche l'envoi du formulaire

        let nomeUtente = document.getElementById("nome_utente").value.trim();
        let email = document.getElementById("indirizzo_mail").value.trim();
        let password = document.getElementById("password").value.trim();
        let confirmPassword = document.getElementById("password_confirm").value.trim();
        let errors = [];


        if (nomeUtente === "") {
            errors.push("Il nome utente deve essere inserito.");
        }


        let emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!emailRegex.test(email)) {
            errors.push("L'indirizzo mail non è valido.");
        }


        if (password.length < 6) {
            errors.push("La password deve contenere almeno 6 caratteri.");
        }


        if (password !== confirmPassword) {
            errors.push("Le password non corrispondono.");
        }


        if (errors.length > 0) {
            alert(errors.join("\n"));
        } else {

            const username = document.getElementById("nome_utente").value;
            const email = document.getElementById("indirizzo_mail").value;
            const password = document.getElementById("password").value;
            const typ=document.getElementById("tipologia").value;

            fetch("../api/utente", {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({
                    action: "register",
                    nome_utente: username,
                    indirizzo_mail: email,
                    password: password,
                    tipologia:typ
                })
            })
            .then(response => response.json())
            .then(data => {
                alert(data.message || data.error);
            })
            .catch(error => console.error("Errore:", error));


              setTimeout(() => {
                  window.location.href = "index";
                }, 3000);
        }
    });
```

Se i dati inseriti rispettano i requisiti del sistema, viene inviata una richiesta all'api di tipo POST con i dati inseriti codificati in JSON con una risposta dello stato di trattazione della domanda. Alla fine, viene definito un timer per redirigere l'utente verso la pagina di login per completare l'operazione di registrazione.

4.1.1.6 utente/index.php

Questa pagina è accessibile solo ai ricercatori o altri enti che si sono autenticati nel sistema, contiene quindi una funzione in testa del codice che impedisce l'accesso a tutti quelli che non hanno il diritto di accederci. Questa funzione è contenuta in tutte le pagine php della cartella utente



```php
<?php
    session_start();
    function attivazione_sessione($data){
        if(!isset($_SESSION["$data"])){
            header("Location: ../login/");
            exit();
        }
    }
```

Il sistema redirige automaticamente l'utente verso la pagina di login se non ha il diritto di accederci.
Il file index lui, è costituito da un menu laterale a sinistra e presenta la dashboard con delle statistiche sul proprio account, sui propri eventi iscritti e sulle iscrizioni. SI vede nell'immagine

```
    <script>
        function getStatistiche() {
    fetch("../api/backend?action=statistiche")
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                let disponibili = document.getElementById("eventi_disponibili");
                let inseriti = document.getElementById("eventi_inseriti");
                let iscritti = document.getElementById("eventi_iscritti");

                if (disponibili && inseriti && iscritti) {
                    disponibili.textContent = data.eventi.eventi_disponibili ?? 0;
                    inseriti.textContent = data.eventi.eventi_inseriti ?? 0;
                    iscritti.textContent = data.eventi.eventi_iscritti ?? 0;
                } else {
                    console.error("Errore");
                }
            } else {
                console.error("Erreur API :", data.message);
            }
        })
        .catch(error => console.error("Erreur durante la recuperazione delle statistiche :", error));
}

document.addEventListener("DOMContentLoaded", getStatistiche);

setInterval(getStatistiche, 5000);

    </script>
```

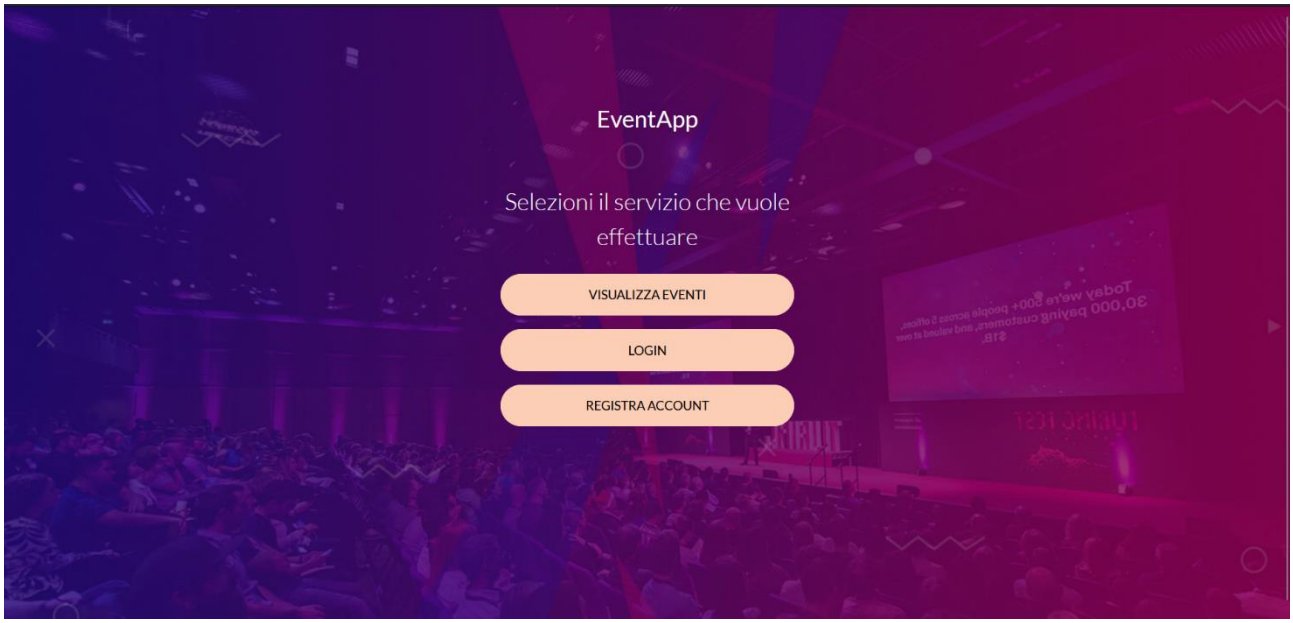
All’interno di questa cartella, troviamo tutte le pagine che hanno delle funzioni simili a quelle mostrate: inserimento, modifica, rimozione.

4.1.1.7 api

Nella cartella api, troviamo il file htaccess, il modello, il presenter, il database e il route. Il modello è composto da 17 funzioni che svolgono un ruolo importante per l’applicazione, perché permette di connettere il sistema con il database e di effettuare tutte le richieste SQL, ad esempio la recupera degli eventi. Il modello invia i dati al presenter che si occupa di inviarli alla view in funzione dello stato di elaborazione della richiesta. Il presenter è composto da due funzioni principali handleRequest e richiesta per motivi di modularità. All’interno della funzione troviamo uno switch per gestire il tipo di richiesta dell’utente che sia una GET, un POST … Il file del database crea un’istanza della classe pdo per gestire la connessione con il database utilizzato in quest’applicazione. E infine la route permette di recuperare i dati generati dalla view e di chiamare il presenter per passarglieli e poi ritrasmettere il risultato alla view.

4.2 Navigazione del sito

Come indicato nel diagramma di attività, l’utente apre l’applicazione e vede direttamente una pagina dove viene mostrata tre opzioni: visualizza eventi, login, registrati



In questa pagina, non viene fatta nessuna interazione direttamente con il sistema perché non ci sono dati da mostrare direttamente, si tratta semplicemente di codice html, css e

Javascript. Quando l'utente ha scelto l'opzione conveniente, il sistema apre la pagina richiesta e viene svolta una serie di connessione al sistema per elaborare i dati, controllare la correttezza dei dati inseriti o inserire un nuovo utente.

La pagina login reindirizza l'utente verso la cartella login/ costituita di due file principali (index.php e registrati.php). Login permette all'utente già iscritto alla piattaforma di autenticarsi e di poter approfittare di tutte le funzionalità proposte dal sistema: inserire gli eventi, modificarli, rimuoverli. Per farlo, bisogna aver creato un conto prima, nella pagina registrati dove troviamo cinque campi controllati direttamente dalla parte view per verificare che l'utente inserisca i dati giusti.

La pagina visualizza eventi mostra la lista di tutti gli eventi inseriti nel sistema da tutti gli utenti del sistema. Offre in più una barra di ricerca dove l'utente inserisce l'hashtag desiderato. E poi viene scelto l'evento per cui si vuole iscrivere cliccando sul pulsante iscriversi. Se l'utente non è connesso, viene chiesto la tipologia di connessione, ricercatore/altri enti o utente esterno. Una volta passato questo step, il sistema inserisce e conferma l'iscrizione all'evento scelto.