

UNIVERSITÀ DEGLI STUDI DI MESSINA

**PROGETTO DI
PROGRAMMAZIONE 3**

Vitaliy Lyaskovskiy

472981

INTRODUZIONE ED ANALISI DEGLI STRUMENTI UTILIZZATI

Il tema centrale di questo progetto è la creazione di un servizio web per la gestione di una mining farm, per raggiungere questo obiettivo ho sviluppato un **servizio web** con un'architettura **REST API** ed un sito internet correlato.

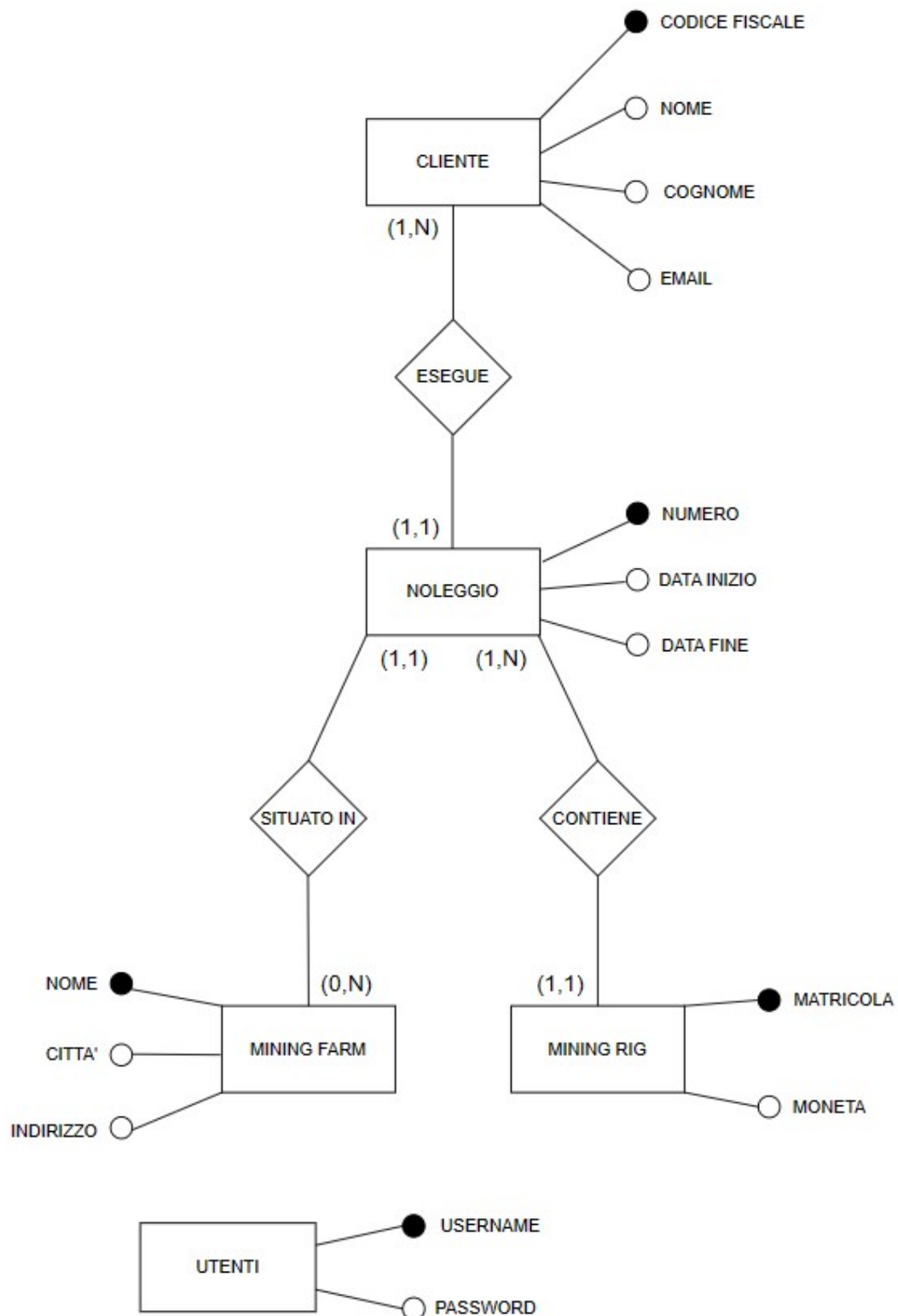
Il sito web, oltre a presentare diverse pagine informative, dispone di un'area riservata che consente agli utenti di accedere al servizio REST e di interagire con le sue API mediante l'utilizzo di interfacce grafiche.

Il sito web è stato realizzato utilizzando il linguaggio **HTML** per definirne la struttura, il linguaggio **Javascript** per rendere dinamico il contenuto e l'interazione con il server, ed infine sono stati utilizzati i fogli di stile **CSS** per creare effetti visivi, gestire il layout degli elementi e rendere l'interfaccia il più user friendly possibile.

Per quanto riguarda il lato server, è stato utilizzato il linguaggio **PHP** per implementare le API che rappresentano gli end points del mio servizio REST, infine la gestione dei dati è stata implementata utilizzando un database **MySQL**.

Per descrivere la realizzazione di questo progetto è stato scelto di utilizzare un approccio bottom-up.

SCHEMA ENTITÀ - RELAZIONE



Lo schema Entità - Relazione rappresenta le entità, le loro proprietà e le relazioni tra di esse all'interno del mio progetto.

MODELLO RELAZIONALE

- CLIENTI(CodiceFiscale, Nome, Cognome, Email)
 - o PK: CodiceFiscale

- NOLEGGI(Numero, DataInizio, DataFine, CodiceFiscaleCliente, NomeMiningFarm)
 - o FK: CodiceFiscaleCliente REFERENCES Clienti
 - o FK: NomeMiningFarm REFERENCES Mining Farm
 - o PK: Numero AUTO_INCREMENT

- MINING FARM(Nome, Città, Indirizzo)
 - o PK: Nome

- MINING RIG(Matricola, Hashrate, Moneta, NumeroNoleggio)
 - o FK: NumeroNoleggio REFERENCES Noleggi
 - o PK: Matricola AUTO_INCREMENT

- UTENTI(Username, Password)
 - o PK: Username

Il modello relazione, ricavato dallo schema entità – relazione, individua le chiavi primarie ed i vincoli sui dati.

IMPLEMENTAZIONE ED ANALISI DEL CODICE LATO SERVER

```
class Database
{
    1 reference
    private $host = "127.0.0.1";
    1 reference
    private $databaseName = "miningfarm";
    1 reference
    private $username = "root";
    1 reference
    private $password = "";
    4 references
    public $conn;

    5 references | 0 overrides
    public function getConnection()
    {
        $this->conn = null;
        try {
            $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" .
            $this->databaseName, $this->username, $this->password);
            $this->conn->exec("set names utf8");
            return $this->conn;
        } catch (PDOException $ex) {
            error_log("Errore durante la connessione con il database: "
            . $ex->getMessage() . "\n", 3, "C:/xampp/htdocs/logServer.log");
        }
    }
}
```

La classe Database viene utilizzata per creare un'istanza che rappresenta la connessione con il database, per fare ciò è stato utilizzata PDO (PHP Data Objects), un'interfaccia in PHP che fornisce un approccio orientato agli oggetti per lavorare con i database. Al fine di impedire attacchi di tipo SQL Injection, le query vengono eseguite mediante l'utilizzo delle prepared statement. È presente inoltre un sistema di log in tutti gli script che comunicano con il database con lo scopo di facilitare la gestione degli errori.

CLASSI

Ogni categoria di risorse viene gestita tramite una istanza della rispettiva classe, i metodi di ogni oggetto istanziato forniscono l'interfaccia per le operazioni CRUD con il database.

```
public function getRentals()
{
    try {
        $query = "SELECT * FROM " . $this->dbTable;
        $statement = $this->conn->prepare($query);
        $statement->execute();
        $result = $statement->fetchAll(PDO::FETCH_ASSOC);
        return $result;
    } catch (PDOException $ex) {
        error_log($this->dbTable . ": " . $ex->getMessage() . "\n", 3, "C:/xampp/htdocs/logServer.log");
        return false;
    }
}
```

Recupero di tutti i noleggi registrati

```
public function getSingleRental()
{
    try {
        $query = "SELECT dataInizio, dataFine, codiceFiscaleCliente, nomeMiningFarm
        FROM " . $this->dbTable . " WHERE numero = :id";
        $statement = $this->conn->prepare($query);
        $statement->bindParam(":id", $this->id);
        $statement->execute();
        $result = $statement->fetch(PDO::FETCH_ASSOC);
        return $result;
    } catch (PDOException $ex) {
        error_log($this->dbTable . ": " . $ex->getMessage() . "\n", 3, "C:/xampp/htdocs/logServer.log");
        return false;
    }
}
```

Recupero di un singolo noleggio

```

public function insertRental()
{
    try {
        $query = "INSERT INTO " . $this->dbTable . " SET dataInizio = :startDate, dataFine = :endDate,
        codiceFiscaleCliente = :clientTaxCode, nomeMiningFarm = :miningFarmName";
        $statement = $this->conn->prepare($query);
        $statement->bindParam(":startDate", $this->startDate);
        $statement->bindParam(":endDate", $this->endDate);
        $statement->bindParam(":clientTaxCode", $this->clientTaxCode);
        $statement->bindParam(":miningFarmName", $this->miningFarmName);
        $statement->execute();
        return;
    } catch (PDOException $ex) {
        error_log($this->dbTable . ": " . $ex->getMessage() . "\n", 3, "C:/xampp/htdocs/logServer.log");
        if (($ex->getCode() == 23000)) {
            return "violazioneChiaveEsterna";
        } else return false;
    }
}

```

Inserimento di un noleggio

```

public function updateRental()
{
    try {
        $query = "UPDATE " . $this->dbTable . " SET dataInizio = :startDate, dataFine = :endDate,
        codiceFiscaleCliente = :clientTaxCode, nomeMiningFarm = :miningFarmName WHERE numero = :id";
        $statement = $this->conn->prepare($query);
        $statement->bindParam(":id", $this->id);
        $statement->bindParam(":startDate", $this->startDate);
        $statement->bindParam(":endDate", $this->endDate);
        $statement->bindParam(":clientTaxCode", $this->clientTaxCode);
        $statement->bindParam(":miningFarmName", $this->miningFarmName);
        $statement->execute();
        $count = $statement->rowCount();
        if ($count == 0) {
            return "nessunRisultato";
        } else return;
    } catch (PDOException $ex) {
        error_log($this->dbTable . ": " . $ex->getMessage() . "\n", 3, "C:/xampp/htdocs/logServer.log");
        if (($ex->getCode() == 23000)) {
            return "violazioneChiaveEsterna";
        } else return false;
    }
}

```

Modifica di un noleggio

```

public function getValidRentals()
{
    try {
        $query = "SELECT * FROM " . $this->dbTable . " WHERE dataFine > CURDATE()";
        $statement = $this->conn->prepare($query);
        $statement->execute();
        $result = $statement->fetchAll(PDO::FETCH_ASSOC);
        return $result;
    } catch (PDOException $ex) {
        error_log($this->dbTable . ": " . $ex->getMessage() . "\n", 3, "C:/xampp/htdocs/logServer.log");
        return false;
    }
}

```

Recupero di tutti i noleggi in corso

```

public function deleteRental()
{
    try {
        $query = "DELETE FROM " . $this->dbTable . " WHERE numero = :id";
        $statement = $this->conn->prepare($query);
        $statement->bindParam(":id", $this->id);
        $statement->execute();
        //Restituisce il numero di record eliminati
        $count = $statement->rowCount();
        if ($count == 0) {
            return "nessunRisultato";
        } else return;
    } catch (PDOException $ex) {
        error_log($this->dbTable . ": " . $ex->getMessage() . "\n", 3, "C:/xampp/htdocs/logServer.log");
        return false;
    }
}

```

Cancellazione di un noleggio

AUTENTICAZIONE - TOKEN JWT

Per poter consumare le API è richiesta la presenza del token JWT nell'header della richiesta. Il token viene rilasciato al client solo dopo aver eseguito l'autenticazione inserendo username e password. Trattandosi di un servizio di tipo RESTful è il client che si occupa di salvare il token e di includerlo nell'apposto header in ogni richiesta.


```

public function checkUser()
{
    try {
        $query = "SELECT * FROM " . $this->dbTable . " WHERE username = :username";
        $statement = $this->conn->prepare($query);
        $statement->bindParam(":username", $this->username);
        $statement->execute();
        $result = $statement->fetch(PDO::FETCH_ASSOC);
        if (empty($result)) {
            return "username sbagliato";
        }
        if (password_verify($this->password, $result['password'])) {
            return true;
        } else {
            return "password sbagliata";
        }
    } catch (PDOException $ex) {
        error_log($this->dbTable . ": " . $ex->getMessage() . "\n", 3, "C:/xampp/htdocs/logServer.log");
        return false;
    }
}

```

Verifica delle credenziali

```

function createJWT($username)
{
    global $issuerClaim, $secretKey;
    $issuedatClaim = time();
    $notbeforeClaim = $issuedatClaim;
    $expireClaim = $issuedatClaim + 3600; // SCADENZA IN SECONDI
    $token = array(
        "iss" => $issuerClaim,
        "iat" => $issuedatClaim,
        "nbf" => $notbeforeClaim,
        "exp" => $expireClaim,
        "data" => array(
            "username" => $username
        )
    );
    $jwt = JWT::encode($token, $secretKey, "HS256");
    return $jwt;
}

```

Generazione del token prima di essere inviato

```

function checkJWT($token)
{
    global $secretKey;
    try {
        // Lancia un'eccezione se il token non è valido oppure scaduto
        JWT::decode($token, new Key($secretKey, "HS256"));
        return true;
    } catch (Exception $ex) {
        return false;
    }
}

```

Verifica del token quando arriva una richiesta

API – AUTENTICAZIONE

Il cuore dell'endpoint responsabile dell'autenticazione è rappresentato dal seguente codice:

```

$requestMethod = $_SERVER["REQUEST_METHOD"];

if ($requestMethod == "POST") {
    $postInput = json_decode(file_get_contents("php://input"), true);

    $user->setUsername(sanitizeInput($postInput["username"]));
    $user->setPassword(sanitizeInput($postInput["password"]));

    $result = $user->checkUser();
    if ($result === true){
        $jwt = createJWT($user->getUsername());
        http_response_code(200);
        echo json_encode(["jwt" => $jwt]);
    } else if ($result === "username sbagliato" or $result === "password sbagliata") {
        http_response_code(401);
        echo json_encode(["messaggio" => "Credenziali errate"]);
    }else{
        http_response_code(500);
        echo json_encode(["Messaggio" => "Errore interno del server"]);
    }
} else {
    http_response_code(405);
    echo json_encode(["Messaggio" => "Metodo non consentito"]);
}

```

Dopo aver eseguito la verifica sulle credenziali ricevute, il server informa il client sull'esito dell'autenticazione. In caso di successo viene inviato il token JWT appena generato, in caso di insuccesso viene comunicato l'esito al client con un appropriato codice di stato. I dati vengono inviati in formato Json.

API - RISORSE

Le API responsabili di fornire una rappresentazione dello stato delle risorse sono così implementate:

```
if (isset($_SERVER["HTTP_AUTHORIZATION"])) {  
    $authHeader = $_SERVER["HTTP_AUTHORIZATION"];  
    $array = explode(" ", $authHeader);  
    if (isset($array[1]) && is_string($array[1])) {  
        $token = sanitizeInput($array[1]);  
    } else {  
        http_response_code(400);  
        echo json_encode(["Messaggio" => "Token non riconosciuto"]);  
        exit();  
    }  
} else {  
    http_response_code(400);  
    echo json_encode(["Messaggio" => "Richiesta senza autorizzazione"]);  
    exit();  
}
```

Nella prima fase viene controllata la presenza del token nell'header, successivamente viene effettuato un controllo sulla validità:

```
if (!checkJWT($token)) {  
    http_response_code(400);  
    echo json_encode(["Messaggio" => "Token non valido"]);  
    exit();  
}
```

In fine in base al metodo della richiesta vengono svolte le operazioni che coinvolgono le classi precedentemente descritte:

```
switch ($requestMethod) {  
    case "GET":  
        if (isset($_GET["status"]) && $_GET["status"] === "valid") {  
            $result = $rental->getValidRentals();  
        } elseif (isset($_GET["id"])) {  
            $rental->setId(sanitizeInput($_GET["id"]));  
            $result = $rental->getSingleRental();  
        } else {  
            $result = $rental->getRentals();  
        }  
  
        if (empty($result)) {  
            http_response_code(404);  
            echo json_encode(["Messaggio" => "Nessun risultato"]);  
        } elseif ($result === false) {  
            http_response_code(500);  
            echo json_encode(["Messaggio" => "Errore interno del server"]);  
        } else {  
            http_response_code(200);  
            echo json_encode($result);  
        }  
        break;  
}
```

Nella richiesta effettuata col metodo GET, viene testato il parametro della query string del URL per stabilire se recuperare un singolo noleggio oppure i noleggi in corso. In caso di assenza di parametri vengono recuperate tutte le informazioni presenti nella tabella associata. È importante notare che i valori ricevuti tramite query string oppure tramite contenuto della richiesta, vengono prima sanitizzati prima di essere utilizzati:


```
function sanitizeInput($data)
{
    $data = strip_tags($data);
    $data = htmlspecialchars($data);
    $data = stripslashes($data);
    $data = trim($data);
    return $data;
}
```

La funzione `sanitizeInput($data)` rimuove tutti i tag HTML e PHP dall'input, converte i caratteri speciali in entità HTML, rimuove gli spazi bianchi e restituisce il dato pulito.

```
case "DELETE":
    if (isset($_GET["id"])) {
        $id = sanitizeInput($_GET["id"]);
        $rental->setId($id);
        $result = $rental->deleteRental();
        if ($result === false) {
            http_response_code(500);
            echo json_encode(["Messaggio" => "Errore interno del server"]);
        } elseif ($result == "nessunRisultato") {
            http_response_code(404);
            echo json_encode(["Messaggio" => "Noleggio non presente"]);
        } else {
            http_response_code(204);
        }
    } else {
        http_response_code(500);
        echo json_encode(["Messaggio" => "Richiesta non valida"]);
    }
    break;
```

Il metodo Delete rimuove dal database la risorsa identificata tramite id

```

case "POST":
    $postInput = json_decode(file_get_contents("php://input"), true);
    $rental->setStartDate(sanitizeInput($postInput["startDate"]));
    $rental->setEndDate(sanitizeInput($postInput["endDate"]));
    $rental->setClientTaxCode(sanitizeInput($postInput["clientTaxCode"]));
    $rental->setMiningFarmName(sanitizeInput($postInput["miningFarmName"]));
    $result = $rental->insertRental();

    if ($result === false) {
        http_response_code(500);
        echo json_encode(["Messaggio" => "Errore interno del server"]);
    } else if ($result == "violazioneChiaveEsterna") {
        http_response_code(409);
        echo json_encode(["Messaggio" => "Violazione del vincolo di chiave esterna"]);
    } else {
        http_response_code(201);
        echo json_encode(["Messaggio" => "Dati inseriti correttamente"]);
    }
    break;

```

Il metodo Post crea una nuova risorsa recuperando i dati sanitizzati dal corpo della richiesta.

```

case "PUT":
    $putInput = json_decode(file_get_contents("php://input"), true);
    $rental->setId(sanitizeInput($putInput["id"]));
    $rental->setStartDate(sanitizeInput($putInput["startDate"]));
    $rental->setEndDate(sanitizeInput($putInput["endDate"]));
    $rental->setClientTaxCode(sanitizeInput($putInput["clientTaxCode"]));
    $rental->setMiningFarmName(sanitizeInput($putInput["miningFarmName"]));
    $result = $rental->updateRental();

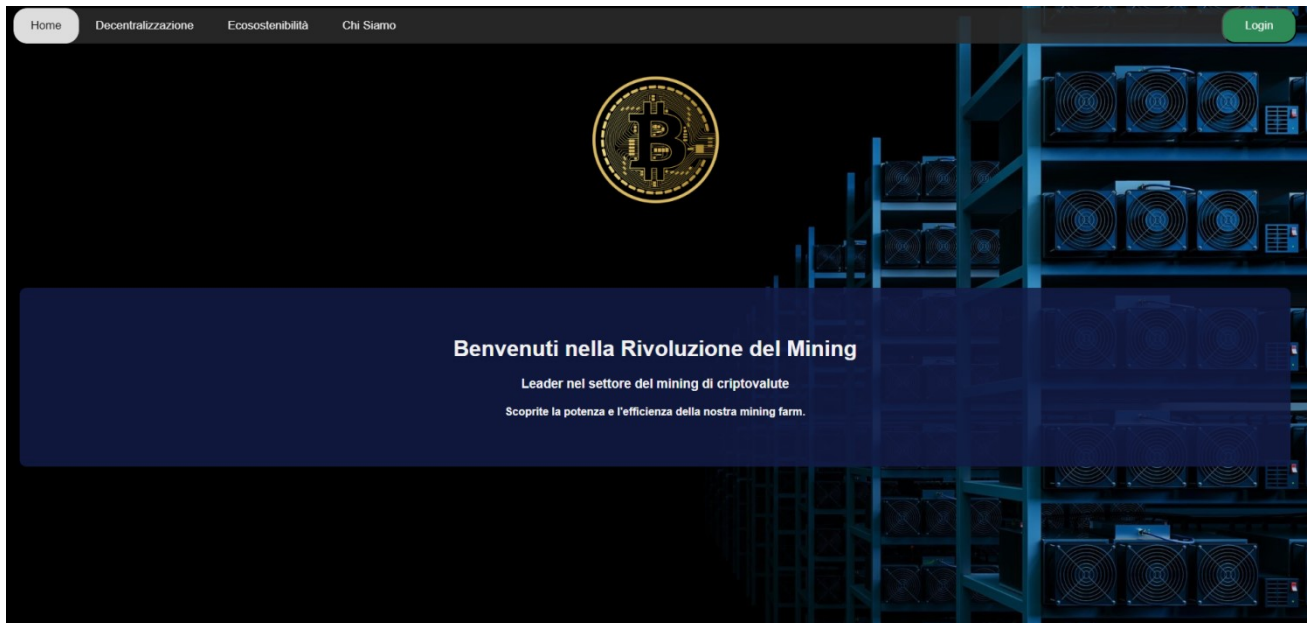
    if ($result === false) {
        http_response_code(500);
        echo json_encode(["Messaggio" => "Errore interno del server"]);
    } else if ($result == "violazioneChiaveEsterna") {
        http_response_code(409);
        echo json_encode(["Messaggio" => "Violazione del vincolo di chiave esterna"]);
    } elseif ($result == "nessunRisultato") {
        http_response_code(404);
        echo json_encode(["Messaggio" => "Noleggio non presente"]);
    } else {
        http_response_code(204);
    }
    break;

```

Infine il metodo put aggiorna i dati della risorsa identificata dall'id

IMPLEMENTAZIONE ED ANALISI DEL CODICE LATO CLIENT

Gli utenti autorizzati ad accedere al sistema possono inserire le loro credenziali mediante l'apposito modale di autenticazione richiamabile dalla barra di navigazione del sito utilizzando il button login:



Il modale è così implementato:

```
<div id="modalLogin">
  <div class="modalContent animation">
    <span class="modalClose" title="Chiudi">&times;</span>
    <div class="modalTopContainer">
      <i class="fa fa-bitcoin" style="font-size:45px"></i>
    </div>
    <form id="modalForm" onsubmit="loginUser(event);">
      <label for="username"><b>Username</b></label>
      <input type="text" id="username" placeholder="Inserisci il tuo Username" name="username" required>
      <label for="password"><b>Password</b></label>
      <input type="password" id="password" placeholder="Inserisci la tua Password" name="password" required>
      <button type="submit" class="modalButton">Login</button>
      <div id="responseMessage"></div>
    </form>
  </div>
</div>
```

Il comportamento normale del form in seguito all'evento submit causerebbe l'aggiornamento della pagina, tuttavia ho scelto di gestire la richiesta al server in modalita asincrona per evitare inutili

aggiornamenti della pagina nel caso in cui vengano inserite credenziali errate:

```
function loginUser(event) {
  event.preventDefault();
  const requestData = {
    username: document.querySelector('[name="username"]').value,
    password: document.querySelector('[name="password"]').value
  };
  fetch('/api/auth/login.php', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(requestData)
  })
  .then(response => {
    if (response.ok) {
      return response.json();
    }
    else if (response.status === 401) {
      throw new Error('Credenziali errate');
    } else {
      throw new Error(response.statusText);
    }
  })
  .then(data => {
    if (data.jwt) {
      window.location.href = 'dashboard.html';
      sessionStorage.setItem('token', JSON.stringify(data.jwt));
    } else {
      throw new Error('La risposta non contiene un token');
    }
  })
  .catch(err => {
    document.getElementById('responseMessage').innerHTML = '❌ ' + err.message;
  });
}
```

La funzione `loginUser(event)` implementata in Javascript riceve come parametro l'evento che ha causato la sua chiamata, il metodo `preventDefault()` impedisce il comportamento predefinito dell'evento che trattandosi di un form risulta essere il ricaricamento della pagina. In questo modo la funzione può gestire l'evento di login inviando una richiesta al server e gestendo la risposta in maniera asincrona.

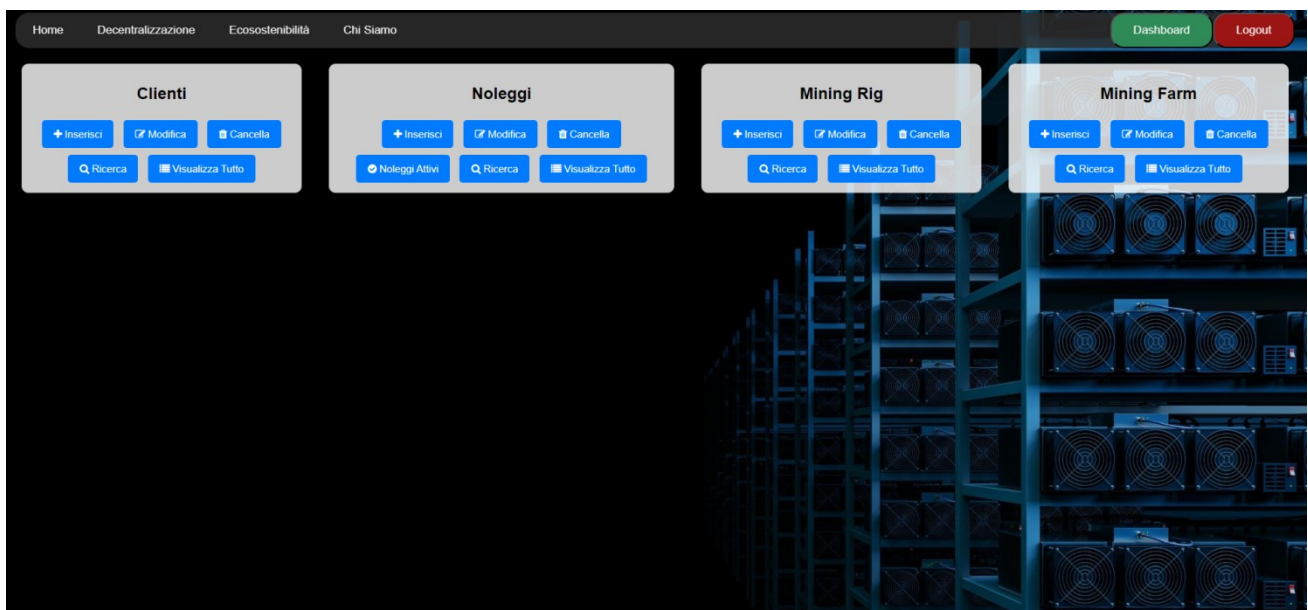
Dopo aver recuperato il valore dei campi input, viene eseguita la funzione `fetch()` per inviare la richiesta al server.

La funzione `fetch()` resituisce una promise che, nel caso in cui venga soddisfatta, consente di accedere ai dati tramite il metodo `.then()`.

Se la risposta ha un codice di stato 200, viene verificata la presenza del token e successivamente quest'ultimo viene salvato nel `sessionStorage` del browser, in caso contrario verrà lanciato un errore catturato dal blocco `catch` che provvederà ad informare l'utente mediante l'inserimento di un messaggio sotto il tasto di login.

AREA RISERVATA

L'utente correttamente autenticato viene reindirizzato all'area riservata:



L'utente ha la possibilità di navigare in qualsiasi parte del sito e poter ritornare nell'area riservata senza avere più il bisogno di eseguire nuovamente l'accesso. Infatti il seguente script:

```
document.addEventListener('DOMContentLoaded', function () {  
    if (sessionStorage.getItem('token') !== null) {  
        document.getElementById("loginButton").style.display = "none";  
        document.getElementById("logoutButton").style.display = "block";  
        document.getElementById("dashButton").style.display = "block";  
    }  
});
```

viene eseguito quando il contenuto del documento HTML è completamente caricato e analizzato, grazie all'evento DOMContentLoaded.

Lo script controlla se esiste un elemento token nel sessionStorage del browser. Se il token esiste nasconde il pulsante di login e mostra i pulsanti di logout e dashboard dell'area riservata. Questo approccio viene utilizzato per cambiare l'interfaccia utente in base allo stato di autenticazione.

Nell'area riservata l'utente dispone di molteplici interfacce che gli consentono di interagire con tutte le risorse messe a disposizione dal servizio web. In particolare l'utente per ogni categoria di risorsa può svolgere le seguenti azioni:



Per i noleggi è possibile anche visualizzare i noleggi attualmente in corso, come precedentemente descritto mostrato nel codice.



L'interazione con i button causerà un comportamento dinamico del browser che, come avvenuto per la fase di autenticazione, provvederà ad inviare una richiesta al server in modo asincrono, non appena i dati della risposta arriveranno al client, verranno processati dal seguente codice javascript:

```

function visualizzaNoleggi() {
  fetch('http://localhost/api/rentals/', {
    headers: {
      'Content-Type': 'application/json',
      'Authorization': 'Bearer ' + JSON.parse(sessionStorage.getItem('token'))
    }
  })
  .then(response => {
    if (response.ok) {
      return response.json();
    } else if (response.status == 404) {
      throw new Error('Nessun risultato');
    } else {
      throw new Error(response.statusText);
    }
  })
  .then(datiNoleggi => {
    var divNoleggi = document.getElementById('modaleDashboard');
    divNoleggi.style.display = 'block';
    var html = `
    <span class="modalClose" onclick="chiudiRisultati()" title="Chiudi">&times;</span>
    <h2>Noleggi Registrati</h2>
    <table>
    <tr><th>Numero</th><th>Data Inizio</th><th>Data Fine</th><th>Codice Fiscale Cliente
    </th><th>Mining Farm</th></tr>`;
    datiNoleggi.forEach(function (noleggio) {
      html += `
      <tr>
      <td>${noleggio.numero}</td>
      <td>${noleggio.dataInizio}</td>
      <td>${noleggio.dataFine}</td>
      <td>${noleggio.codiceFiscaleCliente}</td>
      <td>${noleggio.nomeMiningFarm}</td>
      </tr>
      `;
    });
    html += '</table>';
    divNoleggi.innerHTML = html;
  })
  .catch(err => {
    fallimento(err.message);
  });
}

```

I dati ricevuti verranno processati e visualizzati dinamicamente mediante l'inserimento del codice html in un elemento div inserito per l'apposito ruolo, questo elemento sarà reso visibile solo nel momento in cui è necessario inserire o visualizzare i dati. Ulteriori interfacce presenti:

Nuovo Noleggio

Data Inizio

gg/mm/aaaa

Data Fine

gg/mm/aaaa

Codice Fiscale Cliente

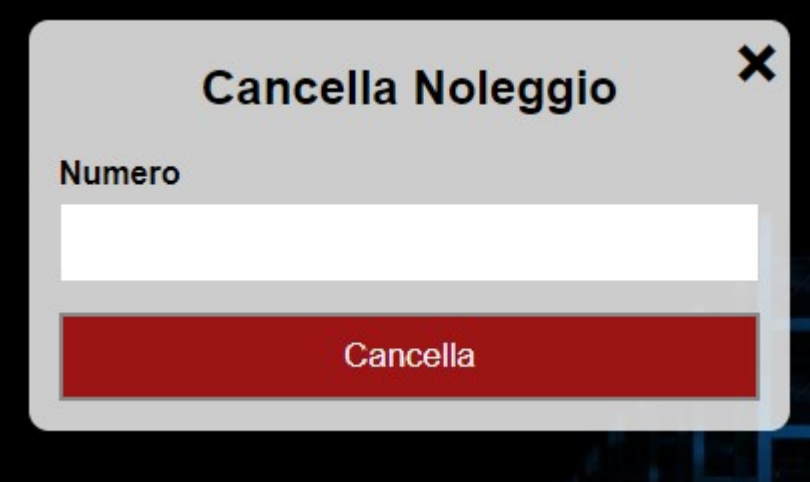
Nome Mining Farm

Inserisci

Inserimento di un nuovo noleggio

Noleggi Registrati				
Numero	Data Inizio	Data Fine	Codice Fiscale Cliente	Mining Farm
19	2024-05-04	2024-05-19	giulianericf0000	giove
20	2024-05-01	2024-05-31	lucaverdicf00000	plutone
21	2024-04-01	2024-05-01	mariorossicf0000	giove

Visualizzazione dei noleggi registrati

A modal dialog box with a light gray background and rounded corners. At the top, the title "Cancella Noleggio" is displayed in bold black text, followed by a close button (X) in the top right corner. Below the title, the label "Numero" is positioned above a white text input field. At the bottom of the dialog, there is a prominent red button with the white text "Cancella".

Cancella Noleggio ✕

Numero

Cancella

Cancellazione di un noleggio

A modal dialog box with a light gray background and rounded corners. At the top, the title "Ricerca Cliente" is displayed in bold black text, followed by a close button (X) in the top right corner. Below the title, the label "Codice Fiscale" is positioned above a white text input field. At the bottom of the dialog, there is a prominent green button with the white text "Ricerca".

Ricerca Cliente ✕

Codice Fiscale

Ricerca

Ricerca di un cliente

Le interfacce appena mostrate processano i dati in maniera dinamica ed asincrona. Infine l'utente che ha intenzione di abbandonare l'area riservata, può eseguire il logout attraverso l'apposito button, in questo modo verrà reindirizzato all'homepage ed il token salvato nel sessionStorage sarà cancellato impedendo quindi l'accesso all'area riservata a personale non autorizzato.