DOCUMENTAZIONE ESAME DI 4° SESSIONE

LARAVEL / API REST BY GABRIEL PERINI

Middleware AuthLogin

Questo middleware controlla che l'utente sia autenticato tramite un **token di autenticazione**. Se il token è valido, l'utente viene autenticato nel sistema. In caso contrario, il middleware blocca l'accesso e restituisce un messaggio di errore.

1. Recupero del token
2. Verifica del token
3. Validazione utente
4. Vai avanti/passa al prossimo middleware

Tipi di response

1) 403 FORBIDDEN
   - **"Token non presente"**: Quando non viene fornito alcun token nella richiesta.
   - **"Token non valido"**: Quando il token fornito non supera i controlli di validità.
   - **"Utente non attivo"**: Quando il contatto è stato trovato ma il suo stato non consente l'accesso.
2) 200 OK
   - Quando il token è valido e l'utente è attivo, il middleware consente l'accesso e prosegue con l'elaborazione della richiesta.

Middleware contattoRuolo

Questo middleware si occupa di controllare se l'utente autenticato possiede almeno uno dei ruoli richiesti per accedere a una determinata risorsa o eseguire una specifica operazione. Se l'utente non ha uno dei ruoli richiesti, la richiesta viene bloccata.

1. Quando il middleware viene attivato, riceve:
   - I ruoli richiesti (esempio: contattoRuolo:Amministratore,Utente).
   - I ruoli dell'utente autenticato dal middleware precedente ($request->contattiruoli).

2. Controlla se almeno uno dei ruoli richiesti corrisponde a quelli dell'utente.

Tipi di response

1. 403 FORBIDDEN
    - "Utente respinto": Quando il contatto è stato trovato ma il suo stato non consente l'accesso.

Se c'è almeno una corrispondenza, il middleware lascia passare la richiesta.

API REST

//FILM

Route::get(_VERS . '/film', [FilmController::class, 'film']);

Questa route, associata al metodo film del FilmController, permette di ottenere una lista di film. L'accesso alla risorsa è regolato da permessi basati sui ruoli e controllati tramite il sistema di autorizzazione di Laravel (Gate).

Il metodo verifica che l'utente abbia i permessi per accedere alla risorsa e restituisce una lista di film in base al ruolo dell'utente:

1. **Controllo del permesso di lettura:**
    - Viene verificato se l'utente ha il permesso di "leggere" (autorizzazione generale).
    - Se il permesso non è presente, l'accesso viene negato con un errore 403 FORBIDDEN e un messaggio:**"Non hai i permessi per accedere a questa risorsa".**
2. **Controllo dei ruoli specifici:**
    - Ruolo Amministratore:
      Gli utenti con il ruolo "Amministratore" ricevono una collezione (FilmAmministratoreCollection) contenente tutti i film non eliminati (dove deleted_at è NULL).
    - Ruolo Utente:
      Gli utenti con il ruolo "Utente" ricevono una collezione (FilmUtenteCollection) contenente solo i film non eliminati.

3. **Restituzione della collezione dei film:**
   Il metodo restituisce la collezione di film appropriata in base al ruolo dell'utente.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha il permesso "leggere"**
- **Quando l'utente non ha fatto l'autenticazione(login)**

200 OK

- **Per utenti con permessi validi, restituisce la lista di film in formato JSON.**

Route::get(_VERS . '/film/{film}', [FilmController::class, 'filmSingolo']);

Questa route è associata al metodo filmSingolo nel FilmController e consente di recuperare i dettagli di un singolo film, identificato dal suo ID. L'accesso è controllato tramite i permessi e i ruoli definiti nel sistema di autorizzazione di Laravel, utilizzando la funzionalità Gate per verificare i diritti dell'utente.

1. **Controllo permessi di lettura:**
   - Il metodo inizia verificando se l'utente ha il permesso di "leggere" la risorsa. Questo controllo viene fatto tramite il sistema Gate di Laravel.
   - Se l'utente non ha il permesso di lettura, la richiesta viene bloccata e restituito un errore **403 FORBIDDEN** con il messaggio:"Non hai i permessi per accedere a questa risorsa".
2. **Ricerca del film per ID:**
   - Se il permesso di lettura è concesso, il metodo procede cercando il film nel database utilizzando l'ID fornito nella route. Vengono considerati solo i film che non sono stati eliminati (cioè, il campo deleted_at è NULL).
   - Se il film con l'ID specificato non viene trovato, viene restituito un errore **404 NOT FOUND** con il messaggio:"Film non presente nel nostro catalogo".

3. **Controllo dei permessi di ruolo:**
    - Una volta trovato il film, il metodo verifica il ruolo dell'utente utilizzando Gate. In base al ruolo dell'utente, vengono restituiti dettagli differenti:
        - **Ruolo Amministratore:**
          Se l'utente è un amministratore, vengono restituite tutte le informazioni dettagliate sul film tramite la risorsa FilmAmministratoreResource.
        - **Ruolo Utente:**
          Se l'utente è un normale utente, vengono restituiti solo il nome e la descrizione del film tramite la risorsa FilmUtenteResource.
4. **Restituzione della risorsa del film:**
    - In base al controllo sui ruoli, il metodo restituisce una risorsa JSON contenente i dettagli del film, formattata in base al tipo di utente.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha il permesso "leggere"**

404 NOT FOUND

2. **Quando il film con l'ID specificato non viene trovato nel database**

200 OK

- **Quando l'utente ha il permesso di leggere la risorsa e il film è stato trovato. La risposta JSON conterrà i dettagli del film in formato appropriato in base al ruolo dell'utente.**

Route::get(_VERS . '/categoria/{categoria}/film', [FilmController::class, 'filmCategoria']);

Questa route è associata al metodo filmCategoria nel FilmController e consente di recuperare i film appartenenti a una determinata categoria, identificata dall'ID della categoria ($idCategoria). L'accesso ai film è controllato tramite i permessi e i ruoli

definiti nel sistema di autorizzazione di Laravel, utilizzando la funzionalità Gate per verificare i diritti dell'utente.

1)Controllo permessi di lettura:

- Il metodo inizia verificando se l'utente ha il permesso di "leggere" la risorsa. Questo controllo viene fatto tramite il sistema Gate di Laravel.
- Se l'utente non ha il permesso di lettura, la richiesta viene bloccata e restituito un errore **403 FORBIDDEN** con il messaggio: "Non hai i permessi per accedere a questa risorsa".

2)**Ricerca della categoria per ID:**

- Se il permesso di lettura è concesso, il metodo procede cercando la categoria nel database utilizzando l'ID della categoria fornito nella route. Se la categoria non esiste, viene restituito un errore **404 NOT FOUND** con il messaggio:"Categoria non trovata".

3)**Recupero dei film della categoria:**

- Se la categoria è trovata, vengono recuperati tutti i film associati a quella categoria, escludendo quelli eliminati (cioè, il campo deleted_at è NULL).

4)**Controllo dei permessi di ruolo:**

- Una volta ottenuti i film della categoria, il metodo verifica il ruolo dell'utente utilizzando Gate. In base al ruolo dell'utente, vengono restituiti dettagli differenti:
    - **Ruolo Amministratore:**
      Se l'utente è un amministratore, vengono restituiti tutti i film nella categoria tramite la collezione FilmAmministratoreCollection.
    - **Ruolo Utente:**
      Se l'utente è un normale utente, vengono restituiti solo i film in formato ridotto tramite la collezione FilmUtenteCollection.

5)**Restituzione dei film:**

- In base al controllo sui ruoli, il metodo restituisce una collezione di film, formattata in base al tipo di utente.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha il permesso "leggere"**

404 NOT FOUND

- **Quando la categoria con l'ID specificato non viene trovata nel database.**

200 OK

- **Quando l'utente ha il permesso di leggere la risorsa e la categoria e i relativi film sono stati trovati. La risposta JSON conterrà i dettagli dei film nella categoria, formattati in base al ruolo dell'utente.**

//SerieTv

Route::get(_VERS . '/serietv', [SerieController::class, 'serieTv']);

Questa route è associata al metodo serieTv nel SerieController e consente di recuperare l'elenco delle serie TV, esclusi quelle eliminate (basato sul campo deleted_at). L'accesso alle serie TV è controllato tramite i permessi definiti nel sistema di autorizzazione di Laravel, utilizzando il sistema Gate per verificare i diritti dell'utente.

1. **Controllo permessi di lettura:**
    - Il metodo inizia verificando se l'utente ha il permesso di "leggere" la risorsa. Questo controllo avviene tramite il sistema Gate di Laravel.
    - Se l'utente non ha il permesso di lettura, la richiesta viene bloccata e restituito un errore **403 FORBIDDEN** con il messaggio: "Non hai i permessi per accedere a questa risorsa".
2. **Recupero delle serie TV:**
    - Se il permesso di lettura è concesso, il metodo recupera tutte le serie TV dal database, escludendo quelle eliminate (cioè, il campo deleted_at è NULL).
3. **Controllo dei permessi di ruolo:**
    - Una volta ottenute le serie TV, il metodo verifica il ruolo dell'utente utilizzando Gate. In base al ruolo dell'utente, vengono restituiti dettagli differenti:
        - **Ruolo Amministratore:**
          Se l'utente è un amministratore, vengono restituiti tutti i dettagli

delle serie TV tramite la collezione SerieAmministratoreCollection.

- **Ruolo Utente:**
  Se l'utente è un normale utente, vengono restituiti solo i dettagli basilari delle serie TV tramite la collezione SerieUtenteCollection.

4. **Restituzione delle serie TV:**
   - In base al controllo sui ruoli, il metodo restituisce una collezione di serie TV, formattata in base al tipo di utente.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha il permesso "leggere"**

200 OK

- **Quando l'utente ha il permesso di leggere la risorsa e le serie TV sono state recuperate con successo. La risposta JSON conterrà i dettagli delle serie TV, formattati in base al ruolo dell'utente.**

```
Route::get(_VERS . '/serietv/{idSerietv}', [SerieController::class, 'serieTvSingola']);
```

Questa route è associata al metodo serieTvSingola nel SerieController e consente di recuperare i dettagli di una singola serie TV, in base al suo idSerietv. L'accesso ai dettagli della serie TV è controllato tramite i permessi di lettura definiti nel sistema di autorizzazione di Laravel e la visibilità delle informazioni dipende dal ruolo dell'utente.

1. **Controllo permessi di lettura:**
   - Inizialmente, il metodo verifica se l'utente ha il permesso di "leggere" la risorsa tramite il sistema di autorizzazione Gate. Se l'utente non ha il permesso di lettura, la richiesta viene bloccata con un errore **403 FORBIDDEN** e il messaggio:"Non hai i permessi per accedere a questa risorsa".
2. **Recupero della serie TV per idSerietv:**
   - Se l'utente ha il permesso di lettura, il metodo recupera la serie TV con l'idSerietv fornito. La ricerca esclude le serie TV che sono state soft

deleted (utilizzando whereNull('deleted_at') per escludere quelle con una data di eliminazione).

3. **Verifica esistenza della serie TV:**
    - Se la serie TV non viene trovata nel database (ossia, se il risultato è null), il metodo restituisce un errore **404 NOT FOUND** con il messaggio:"Serie TV non presente nel nostro catalogo".
4. **Controllo dei permessi di ruolo:**
    - Dopo aver recuperato la serie TV, il metodo verifica i permessi in base al ruolo dell'utente:
        - **Ruolo Amministratore:**
          Se l'utente è un amministratore, vengono restituiti tutti i dettagli della serie TV tramite la risorsa SerieAmministratoreResource.
        - **Ruolo Utente:**
          Se l'utente è un normale utente, vengono restituiti solo i dettagli basilari tramite la risorsa SerieUtenteResource.
5. **Restituzione della serie TV:**
    - Il metodo restituisce la risposta formattata in base al ruolo dell'utente, includendo i dettagli appropriati per ciascun caso.
    -

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha il permesso "leggere"**

404 NOT FOUND

- **Quando la serie TV con l'idSerietv specificato non esiste o è stata eliminata (soft deleted).**

200 OK

- **Quando la serie TV esiste ed è stata trovata correttamente nel database, e l'utente ha il permesso di accedere alla risorsa. La risposta JSON restituirà i dettagli della serie TV, formattati in base al ruolo dell'utente**

Route::get(_VERS . '/categoria/{idCategoria}/serietv', [SerieController::class, 'serieTvCategoria']);

Questa route è associata al metodo serieTvCategoria nel SerieController, e consente di recuperare tutte le serie TV appartenenti a una specifica categoria, in base all'idCategoria fornito. L'accesso alle serie TV è controllato tramite i permessi di lettura e dipende dal ruolo dell'utente, che determina i dettagli visibili.

1. **Controllo permessi di lettura:**
    - Il metodo verifica se l'utente ha il permesso di "leggere" la risorsa tramite il sistema di autorizzazione Gate. Se l'utente non ha il permesso, la richiesta viene bloccata con un errore **403 FORBIDDEN** e il messaggio: "Non hai i permessi per accedere a questa risorsa".
2. **Recupero della categoria per idCategoria:**
    - Se l'utente ha il permesso di lettura, il metodo cerca la categoria tramite l'idCategoria fornito. Se la categoria non esiste nel database, la richiesta restituirà un errore **404 NOT FOUND** con il messaggio: "Categoria non trovata".
3. **Recupero delle serie TV associate alla categoria:**
    - Una volta trovata la categoria, il metodo recupera tutte le serie TV ad essa associate che non sono state soft deleted, utilizzando il metodo whereNull('deleted_at') per escludere quelle eliminate.
4. **Controllo dei permessi di ruolo:**
    - Dopo aver recuperato le serie TV, il metodo verifica i permessi in base al ruolo dell'utente:
        - **Ruolo Amministratore:**
          Se l'utente è un amministratore, vengono restituiti tutti i dettagli delle serie TV tramite la risorsa SerieAmministratoreCollection.
        - **Ruolo Utente:**
          Se l'utente è un normale utente, vengono restituiti solo i dettagli basilari tramite la risorsa SerieUtenteCollection.
5. **Restituzione delle serie TV**
    - Il metodo restituirà una collezione di serie TV, formattata in base al ruolo dell'utente, includendo i dettagli appropriati per ciascun caso.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha il permesso "leggere"**

404 NOT FOUND

- **Quando la categoria con l'idCategoria specificato non esiste**

200 OK

- **Quando la categoria esiste e contiene delle serie TV che non sono soft deleted. La risposta JSON restituirà una collezione di serie TV, formattata in base al ruolo dell'utente**

//Episodi

Route::get(_VERS . '/serietv/{serietv}/episodi', [episodiController::class, 'listaEpisodi']);

Questa route è associata al metodo listaEpisodi nel EpisodiController, e consente di recuperare la lista degli episodi associati a una specifica serie TV, identificata dal suo idSerietv. L'accesso agli episodi è controllato tramite i permessi di lettura e dipende dal ruolo dell'utente, che determina i dettagli visibili.

1. **Controllo permessi di lettura:**
    - Il metodo verifica se l'utente ha il permesso di "leggere" la risorsa tramite il sistema di autorizzazione Gate. Se l'utente non ha il permesso, la richiesta viene bloccata con un errore **403 FORBIDDEN** e il messaggio: "Non hai i permessi per accedere a questa risorsa".
2. **Recupero della serie TV per idSerietv:**
    - Se l'utente ha il permesso di lettura, il metodo cerca la serie TV tramite l'idSerietv fornito. Viene utilizzato il metodo with per caricare anche gli episodi associati, applicando un filtro per escludere gli episodi soft deleted (whereNull('deleted_at')).
3. **Verifica dell'esistenza della serie TV:**
    - Se la serie TV con l'idSerietv fornito non esiste, la richiesta restituirà un errore **404 NOT FOUND** con il messaggio: "Serie TV non trovata".
4. **Recupero degli episodi associati alla serie TV**

- Una volta trovata la serie TV, il metodo recupera tutti gli episodi associati. Se non ci sono episodi associati (ad esempio, se la lista degli episodi è vuota), viene restituito un errore **404 NOT FOUND** con il messaggio: "Nessun episodio trovato per questa serie TV".

5. **Controllo dei permessi di ruolo:**
   - Dopo aver recuperato gli episodi, il metodo verifica i permessi in base al ruolo dell'utente:
     - **Ruolo Amministratore:**
       Se l'utente è un amministratore, vengono restituiti tutti i dettagli degli episodi tramite la risorsa EpisodiAmministratoreCollection.
     - **Ruolo Utente:**
       Se l'utente è un normale utente, vengono restituiti solo i dettagli basilari tramite la risorsa EpisodiUtenteCollection.
6. **Restituzione degli episodi:**
   - Il metodo restituirà una collezione di episodi, formattata in base al ruolo dell'utente, includendo i dettagli appropriati per ciascun caso.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha il permesso "leggere"**

404 NOT FOUND

- **Quando la serie TV con l'idSerietv specificato non esiste**
- **Quando non ci sono episodi associati alla serie TV**

200 OK

- **Quando la serie TV esiste e contiene episodi che non sono soft deleted. La risposta JSON restituirà una collezione di episodi, formattata in base al ruolo dell'utente.**

```
Route::get(_VERS . '/serietv/{serietv}/episodio/{idEpisodio}',[episodiController::class, 'singoloEpisodio']);
```

Questa route è associata al metodo singoloEpisodio nel EpisodiController, e consente di recuperare un episodio specifico di una serie TV, identificato dal suo idEpisodio, all'interno di una determinata serie TV (serietv).

1. **Controllo permessi di lettura:**
    - Il metodo verifica se l'utente ha il permesso di "leggere" la risorsa tramite il sistema di autorizzazione Gate. Se l'utente non ha il permesso, la richiesta viene bloccata con un errore **403 FORBIDDEN** e il messaggio: "Non hai i permessi per accedere a questa risorsa".
2. **Recupero dell'episodio per idEpisodio e idSerieTv:**
    - Se l'utente ha il permesso di lettura, il metodo cerca l'episodio associato alla serie TV tramite i parametri serietv e idEpisodio forniti nella richiesta. Viene utilizzato il filtro whereNull('deleted_at') per escludere gli episodi soft deleted.
3. **Verifica dell'esistenza dell'episodio:**
    - Se l'episodio con l'idEpisodio specificato nella serie TV non esiste o è stato eliminato, la richiesta restituirà un errore **404 NOT FOUND** con il messaggio: "Episodio non trovato o eliminato".
4. **Controllo dei permessi di ruolo:**
    - Dopo aver recuperato l'episodio, il metodo verifica i permessi in base al ruolo dell'utente:
        - **Ruolo Amministratore:**
          Se l'utente è un amministratore, vengono restituiti tutti i dettagli dell'episodio tramite la risorsa EpisodiAmministratoreResource.
        - **Ruolo Utente:**
          Se l'utente è un normale utente, vengono restituiti solo i dettagli basilari tramite la risorsa EpisodiUtenteResource.
5. **Restituzione dell'episodio:**
    - Il metodo restituirà un singolo episodio, formattato in base al ruolo dell'utente, includendo i dettagli appropriati per ciascun caso.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha il permesso "leggere"**

404 NOT FOUND

- **Quando l'episodio non viene trovato per la serie TV con l'idEpisodio specificato o se è stato eliminatoQuando non ci sono episodi associati alla serie TV**

200 OK

- **Quando l'episodio esiste e non è stato eliminato. La risposta JSON restituirà i dettagli dell'episodio formattati in base al ruolo dell'utente.**

//-----GESTIONE PROFILO-----//

//UTENTE

Route::put(_VERS . '/contatto/edit-profile/{id}', [ContattoController::class, 'update']);

Questa route è associata al metodo update nel ContattoController, che consente di aggiornare i dati anagrafici di un contatto.

1. **Controllo dei permessi di aggiornamento:**
    - Il metodo inizia verificando tramite il sistema di autorizzazione Gate se l'utente ha il permesso di "aggiornare" la risorsa. Se l'utente non ha il permesso, viene restituito un messaggio di errore con codice **403 FORBIDDEN**:
2. **Recupero del contatto e verifica dell'esistenza:**
    - Se l'utente ha il permesso di aggiornare, si verifica anche se l'utente ha il ruolo di **Amministratore** o **Utente**. Se il controllo passa, il contatto viene recuperato dal database utilizzando il suo id tramite il metodo findOrFail(). Se il contatto non viene trovato, la richiesta restituirà un errore **404 NOT FOUND** con il messaggio:

3. Validazione dei dati in arrivo:

- I dati ricevuti nella richiesta vengono validati con le seguenti regole:
    - nome: obbligatorio, massimo 255 caratteri.
    - cognome: obbligatorio, massimo 255 caratteri.

- sesso: obbligatorio.
- idNazione: obbligatorio, deve essere un intero.
- dataNascita: obbligatorio, deve essere una data valida.

- Se i dati non sono validi, Laravel restituirà automaticamente un errore di validazione.

4. **Aggiornamento dei dati:**

- Se la validazione è corretta, il metodo usa il metodo fill() per aggiornare il contatto con i nuovi dati provenienti dalla richiesta. Dopodiché, il contatto viene salvato nel database con il metodo save().

5. **Restituzione della risposta di successo:**

- Se tutto va a buon fine, viene restituita una risposta di successo con il codice **200 OK**

6. **Gestione delle eccezioni:**

- Se durante l'esecuzione del metodo si verifica un'eccezione (ad esempio, un errore nel salvataggio o nella validazione), viene catturata e restituita una risposta con codice **500 INTERNAL SERVER ERROR** e un messaggio di errore dettagliato

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha i permessi per aggiornare i dati**

404 NOT FOUND

- **Quando il contatto con l'id specificato non viene trovato nel database**

422 UNPROCESSABLE ENTITY (Errore di validazione)

- **Se i dati inviati non sono validi (ad esempio, nome è troppo lungo o dataNascita non è una data valida), Laravel restituirà un errore di validazione. La risposta JSON conterrà i dettagli degli errori di validazione.**

500 INTERNAL SERVER ERROR

- **Se si verifica un errore imprevisto durante l'esecuzione del metodo.**

200 OK

- **Quando i dati sono stati aggiornati con successo.**

Route::put(_VERS . '/contatto/edit-profile/credenziali/{idContatto}', [ContattoController::class, 'updateCredential']);

Questa route è associata al metodo updateCredential nel ContattoController, che consente di aggiornare le credenziali di un contatto (utente).

1. **Controllo dei permessi di aggiornamento:**
    - Il metodo inizia verificando tramite il sistema di autorizzazione Gate se l'utente ha il permesso di "aggiornare" le credenziali. Se l'utente non ha il permesso, viene restituito un errore di accesso negato.
2. **Verifica dei ruoli utente:**
    - Se il permesso di aggiornamento è presente, il metodo verifica se l'utente ha il ruolo di **Amministratore** o **Utente**. Se nessuno di questi ruoli è presente, l'accesso viene negato.
3. **Recupero del contatto e validazione dei dati:**
    - Il contatto viene recuperato dal database tramite l'idContatto. Se il contatto non viene trovato, viene restituito un errore di "utente non trovato".
    - I dati ricevuti nella richiesta vengono validati, in particolare per l'email e la password (se presente).
4. **Aggiornamento delle credenziali:**
    - Se i dati sono validi:
        - L'email viene hashata e aggiornata nel modello auth.

        - Se è stata fornita una nuova password, questa viene hashata e aggiornata nel modello password.
    - Le modifiche vengono salvate nel database.
5. **Restituzione della risposta di successo:**
    - Se tutto va a buon fine, viene restituita una risposta di successo indicando che le credenziali sono state aggiornate correttamente.
6. **Gestione delle eccezioni:**
    - Se si verifica un'eccezione durante l'esecuzione del metodo (ad esempio, un errore nel salvataggio delle credenziali), l'errore viene catturato e restituito con un messaggio di errore generico.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha i permessi necessari per aggiornare le credenziali o non appartiene ai ruoli autorizzati (Amministratore, Utente).**

404 NOT FOUND

- **Quando il contatto con l'idContatto specificato non viene trovato nel database.**

422 UNPROCESSABLE ENTITY (Errore di validazione)

- **Se i dati inviati non sono validi (ad esempio, l'email non è corretta o la password non soddisfa i requisiti), Laravel fornirà automaticamente un errore di validazione.**

500 INTERNAL SERVER ERROR

- **Se si verifica un errore imprevisto (ad esempio, un errore nel salvataggio dei dati nel database).**

200 OK

- **Quando le credenziali sono state aggiornate con successo.**

Route::put(_VERS . '/contatto/{idContatto}/edit-profile/crediti', [ContattoController::class, 'addCredits'])->middleware('contattoRuolo:Utente');

Questa route è associata al metodo addCredits nel ContattoController. Consente agli utenti con il ruolo di **Utente** di aggiungere crediti al proprio profilo.

1. **Middleware per il controllo dei ruoli:**
    - La route utilizza il middleware contattoRuolo:Utente, assicurandosi che solo gli utenti con il ruolo di **Utente** possano accedere a questa funzionalità.
2. **Controllo dei permessi di aggiornamento:**
    - Il metodo verifica tramite il sistema Gate se l'utente ha il permesso di "aggiornare".
3. **Recupero dell'ID del contatto:**
    - L'ID del contatto è recuperato dall'utente autenticato tramite $request->user()->idContatto.
4. **Verifica dell'esistenza del contatto:**
    - Il metodo cerca l'utente nel modello Auth utilizzando idContatto. Se il contatto non esiste, restituisce un errore 404.
5. **Aggiunta dei crediti:**
    - L'importo specificato in $request->crediti viene aggiunto al totale dei crediti del contatto ($contatto->crediti).
    - Le modifiche vengono salvate nel database.
6. **Risposta JSON:**
    - Se l'operazione ha successo, il metodo restituisce una risposta JSON con un messaggio di successo e il totale aggiornato dei crediti.
7. **Gestione degli errori:**
    - Se l'utente non ha i permessi o non è un **Utente**, viene restituito un errore 403.
    - Se il contatto non viene trovato, viene restituito un errore 404.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha i permessi necessari o non appartiene al ruolo Utente.**

404 NOT FOUND

- **Quando il contatto specificato non viene trovato nel database.**

200 OK

- Quando i crediti vengono aggiunti con successo, viene restituita la quantità totale aggiornata dei crediti.

//AMMINISTRATORE

Route::put(_VERS . '/settings/accounts/manage-credits/{idContatto}', [ContattoController::class, 'manageCredits']);

Questa route è associata al metodo manageCredits nel ContattoController. Consente agli Amministratori di gestire i crediti di un contatto specificato, aggiungendoli o sottraendoli.

1. **Controllo dei permessi di aggiornamento:**
    - **Il metodo verifica tramite il sistema Gate se l'utente ha il permesso di "aggiornare".**
2. **Verifica del ruolo di Amministratore:**
    - **Solo gli utenti con il ruolo di Amministratore possono eseguire questa operazione.**
3. **Recupero del contatto:**
    - **Il contatto è recuperato dal modello Auth tramite idContatto.**
    - **Se il contatto non esiste, restituisce un errore 404.**
4. **Validazione dell'input:**
    - **crediti: Deve essere un numero maggiore di 0.**

- **action: Deve essere specificato e valido (es. 'aggiungi' o 'sottrai').**

5. **Gestione dell'azione:**
   - **aggiungi: Aumenta i crediti del contatto del valore specificato.**
   - **sottrai: Riduce i crediti del contatto solo se i crediti attuali sono sufficienti.**
   - **Se l'azione è invalida o i crediti sono insufficienti, restituisce un errore 400.**
6. **Salvataggio e risposta:**
   - **Salva i crediti aggiornati nel database.**
   - **Restituisce una risposta JSON con un messaggio di successo.**

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha i permessi necessari o non è un Amministratore.**

404 NOT FOUND

- **Quando il contatto specificato non è presente nel database.**

400 BAD REQUEST - Errore nella richiesta

- **Quando i crediti sono <= 0.**
- **Quando l'azione è invalida.**
- **Quando si tenta di sottrarre crediti insufficienti.**

200 OK

- **Quando l'operazione è completata con successo.**

Route::put(_VERS . '/contatto/{idAdmin}/status-User', [ContattoController::class, 'StatusUtente']);

Questa route consente agli Amministratori di aggiornare lo stato (idStatus) di un contatto specifico tramite il metodo StatusUtente del ContattoController.

1. **Verifica dei permessi:**
    - Utilizza il sistema Gate per verificare se l'utente corrente ha i permessi per aggiornare e se è un Amministratore.
2. **Recupero dei dati dalla richiesta:**
    - idContatto: ID del contatto il cui stato deve essere aggiornato.
    - idStatus: Nuovo valore di stato da assegnare al contatto.
3. **Recupero del contatto da aggiornare:**
    - Utilizza il modello Contatto per cercare il contatto con l'ID specificato.
    - Se il contatto non viene trovato, restituisce un errore 404 con il messaggio "contatto non trovato".
4. **Aggiornamento dello stato:**
    - Modifica l'attributo idStatus del contatto con il nuovo valore fornito.
    - Salva le modifiche nel database.
5. **Risposta JSON di successo:**
    - In caso di aggiornamento completato con successo, restituisce un messaggio JSON con conferma.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha i permessi necessari o non è un Amministratore.**

404 NOT FOUND

- **Quando l'ID del contatto specificato non esiste nel database.**

200 OK

- Quando lo stato del contatto è stato aggiornato con successo.

//INSERIMENTO

Route::post(_VERS . '/insert/film', [FilmController::class, 'InsertFilm']);

Questa route consente agli **Amministratori** di aggiungere un nuovo film al database, associandolo a una categoria specifica.

1. **Autorizzazione:**
    - Verifica che l'utente abbia i permessi per creare risorse utilizzando il sistema Gate.
    - Consente l'accesso solo agli **Amministratori**.
2. **Validazione dei dati:**
    - I campi richiesti sono:
        - **nome**: Nome del film, massimo 255 caratteri.
        - **descrizione**: Descrizione del film.
        - **idCategoria**: ID della categoria a cui associare il film, deve essere un intero.
    - Se la validazione fallisce, restituisce un messaggio di errore con status **400**.
3. **Inserimento del film:**
    - Il film viene aggiunto alla tabella **film** con i dati validati.
4. **Associazione alla categoria:**
    - Viene creata una relazione tra il film e la categoria nella tabella **film_categoria**, utilizzando l'ID del film appena creato e l'ID della categoria fornito.
5. **Gestione degli errori:**
    - Eventuali eccezioni durante il processo di inserimento vengono catturate e viene restituito un messaggio di errore con status **500**.
6. **Risposta di successo:**

- In caso di inserimento completato con successo, restituisce un messaggio di conferma con status **200**.

Tipi di Response

403 FORBIDDEN

- **Quando l'utente non ha i permessi necessari o non è un Amministratore.**

500 INTERNAL SERVER ERROR - Errore durante l'inserimento

- **Quando si verifica un errore imprevisto durante l'inserimento.**

400 BAD REQUEST - Errore nella richiesta

- **Quando i dati forniti non rispettano i criteri di validazione.**

200 OK

- **Quando il film è stato inserito e associato correttamente a una categoria**

Route::post(_VERS . '/insert/serietv', [SerieController::class, 'InsertSerieTv']);

Questa route consente agli **Amministratori** di aggiungere una nuova Serie TV al database, associandola a una categoria esistente. Include una gestione robusta di validazione, transazioni, e controllo degli errori.

1. **Autorizzazione:**
   - Verifica che l'utente abbia i permessi per creare risorse utilizzando il sistema Gate.
   - Consente l'accesso solo agli **Amministratori**.
2. **Validazione dei dati:**
   - I campi richiesti sono:
     - **nome**: Nome della Serie TV, massimo 255 caratteri.
     - **descrizione**: Descrizione della Serie TV.
     - **idCategoria**: ID della categoria a cui associare la Serie TV, deve essere un intero.
3. **Verifica della categoria:**
   - Controlla che la categoria fornita esista nel database.

    - Se non trovata, restituisce un errore **404**.
4. **Inserimento della Serie TV:**
    - Crea una nuova Serie TV nella tabella **serietv**.
5. **Associazione della Serie TV alla categoria:**
    - Registra la relazione nella tabella pivot **serietv_categoria**.
6. **Transazione:**
    - Utilizza una transazione database per garantire che l'operazione sia atomica:
        - Se una delle operazioni fallisce, annulla tutte le modifiche eseguite.
7. **Gestione degli errori:**
    - Logga eventuali eccezioni e annulla la transazione con DB::rollBack().
    - Restituisce un errore generico con status **500**.
8. **Risposta di successo:**
    - Se l'operazione è completata con successo, restituisce i dettagli della Serie TV creata e il suo ID con status **200**.

Tipi di Response

403 FORBIDDEN

- **L'utente non ha i permessi necessari per creare risorse o non è un Amministratore.**

400 BAD REQUEST -Dati non validi

- **I dati forniti non rispettano i criteri di validazione.**

404 NOT FOUND - Categoria non trovata

- **L'ID della categoria fornito non corrisponde a una categoria esistente.**

500 INTERNAL SERVER ERROR - Errore durante l'inserimento

- **Si è verificato un errore durante l'operazione e la transazione è stata annullata.**

200 OK

- **La Serie TV è stata inserita correttamente e associata alla categoria.**

Route::post(_VERS . '/insert/serietv/episodio', [episodiController::class, 'InsertEpisodio']);

Questa route consente agli **Amministratori** di aggiungere un nuovo episodio a una Serie TV esistente. L'operazione è soggetta a controllo dei permessi e validazione dei dati.

1. **Autorizzazione:**
    - Controlla i permessi dell'utente:
        - Deve avere il permesso di **creare risorse**.
        - Deve essere un **Amministratore**.
2. **Validazione dei dati:**
    - I campi richiesti sono:
        - **idSerieTv**: ID della Serie TV a cui associare l'episodio (deve essere un intero).
        - **titolo**: Titolo dell'episodio (massimo 255 caratteri).
        - **descrizione**: Descrizione dell'episodio (massimo 255 caratteri).
3. **Creazione dell'episodio:**
    - Utilizza il metodo serieTv_Episodi::create() per inserire i dati nella tabella.
4. **Gestione degli errori:**
    - In caso di eccezione, restituisce un errore **500** con il messaggio dettagliato per il debug.
5. **Risposta di successo:**
    - Restituisce una risposta JSON con lo stato **201** e i dettagli dell'episodio creato.

Tipi di Response

403 FORBIDDEN Permessi insufficienti

- **⍰ L'utente non ha i permessi per creare risorse o non è un Amministratore.**

422 UNPROCESSABLE ENTITY - Dati non validi

- **I dati forniti non rispettano i criteri di validazione.**

500 INTERNAL SERVER ERROR - Errore durante l'inserimento

- **Si è verificato un errore durante l'operazione.**

201 CREATED

- **Restituisce i dettagli dell'episodio appena creato.**

//MODIFICA

route::put(_VERS . '/update/film/{idFilm}', [FilmController::class, 'UpdateFilm']);

Questa route permette agli **Amministratori** di aggiornare le informazioni di un film esistente, inclusa la sua relazione con una categoria. L'operazione è protetta da controlli sui permessi e validazione.

1. **Autorizzazione:**
    - Verifica che l'utente abbia il permesso di aggiornare risorse.
    - Controlla che l'utente sia un Amministratore.
2. **Identificazione del film:**
    - Cerca il film nel database tramite l'ID passato come parametro.
    - Se il film non viene trovato, restituisce un errore **404**.
3. **Validazione della categoria:**
    - Controlla se l'ID della categoria (idCategoria) esiste nella tabella film_Categoria.
    - Se la categoria non viene trovata, restituisce un errore **404**.
4. **Aggiornamento dei dati:**
    - Modifica la relazione tra il film e la categoria nella tabella film_Categoria.
    - Aggiorna i campi del film, come nome e descrizione, nella tabella principale.
5. **Gestione degli errori:**

- In caso di problemi durante il processo di aggiornamento, restituisce un messaggio di errore dettagliato.

6. **Risposta di successo:**
   - Conferma l'aggiornamento del film con un messaggio di successo.

Tipi di Response

403 FORBIDDEN Permessi insufficienti

- **L'utente non ha i permessi per aggiornare risorse o non è un Amministratore.**

404 NOT FOUND

- **L'ID fornito non corrisponde a nessun film nel database.**
- **L'ID della categoria fornito non esiste nella tabella film_Categoria.**

500 INTERNAL SERVER ERROR - Errore durante l'inserimento

- **Si è verificato un errore durante l'operazione.**

201 CREATED

- **Il film è stato aggiornato correttamente, inclusa la relazione con la categoria.**

route::put(_VERS . '/update/serietv/{idSerietv}', [SerieController::class, 'updateSerieTv']);

Questa route permette agli **Amministratori** di aggiornare le informazioni di una serie TV esistente, inclusa la sua relazione con una categoria. L'operazione è protetta da controlli sui permessi e validazione.

1. Autorizzazione:

- **Verifica permessi**: Verifica che l'utente abbia il permesso di aggiornare risorse.
- **Controllo ruolo**: Controlla che l'utente sia un **Amministratore**.

2. Identificazione della serie TV:

- **Ricerca nel database**: Cerca la serie TV nel database utilizzando l'ID passato come parametro.

- **Serie TV non trovata**: Se la serie TV non viene trovata, restituisce un errore **404 Not Found**.

3. Validazione della categoria:

- **Verifica idCategoria**: Controlla se l'ID della categoria (idCategoria) esiste nella tabella serietv_categoria.
- **Categoria non trovata**: Se la categoria non viene trovata, restituisce un errore **404 Not Found**.

4. Aggiornamento dei dati:

- **Modifica relazione serie TV-categoria**: Aggiorna la relazione tra la serie TV e la categoria nella tabella serieTv_categoria.
- **Aggiornamento serie TV**: Modifica i dati della serie TV, come il nome e la descrizione, nella tabella principale serietv.

5. Gestione degli errori:

- Se si verificano problemi durante l'operazione di aggiornamento, restituisce un errore **500 Internal Server Error** con un messaggio di errore dettagliato.

6. Risposta di successo:

- Se l'operazione ha successo, restituisce una risposta con il messaggio di conferma e il codice **201 Created**.

Tipi di Response:

403 FORBIDDEN - Permessi insufficienti

- **Descrizione**: L'utente non ha i permessi per aggiornare risorse o non è un Amministratore.

404 NOT FOUND

- **Descrizione**:
    - L'ID della serie TV fornito non corrisponde a nessuna serie TV nel database.
    - L'ID della categoria fornito non esiste nella tabella serietv_categoria.

500 INTERNAL SERVER ERROR - Errore durante l'inserimento

- **Descrizione**: Si è verificato un errore durante l'operazione.

201 CREATED - Successo

- **Descrizione**: La serie TV è stata aggiornata correttamente, inclusa la relazione con la categoria.

route::put(_VERS . '/update/serietv/{idSerietv}/episodio/{idEpisodio}', [episodiController::class, 'UpdateEpisodi']);

Questa route permette agli **Amministratori** di aggiornare le informazioni di un episodio di una serie TV esistente. L'operazione è protetta da controlli sui permessi e validazione.

1. Autorizzazione:

- **Verifica permessi**: Verifica che l'utente abbia il permesso di aggiornare risorse.
- **Controllo ruolo**: Controlla che l'utente sia un **Amministratore**.

2. Identificazione dell'episodio:

- **Ricerca nel database**: Cerca l'episodio nel database utilizzando l'ID dell'episodio passato come parametro.
- **Episodio non trovato**: Se l'episodio non viene trovato, restituisce un errore **404 Not Found**.

3. Verifica della serie TV:

- **Ricerca della serie TV**: Controlla se la serie TV associata all'episodio esiste nel database tramite l'ID della serie TV passato come parametro.
- **Serie TV non trovata**: Se la serie TV non viene trovata, restituisce un errore **404 Not Found**.

4. Aggiornamento dei dati:

- **Modifica dei dati dell'episodio**: Aggiorna i dati dell'episodio (titolo e descrizione) nella tabella serieTv_Episodi.

5. Gestione degli errori:

- Se si verificano problemi durante l'operazione di aggiornamento, restituisce un errore **500 Internal Server Error** con un messaggio di errore dettagliato.

6. Risposta di successo:

- Se l'operazione ha successo, restituisce una risposta con il messaggio di conferma e il codice **200 OK**.

Tipi di Response:

403 FORBIDDEN - Permessi insufficienti

- L'utente non ha i permessi per aggiornare risorse o non è un Amministratore.

404 NOT FOUND

- L'ID dell'episodio fornito non corrisponde a nessun episodio nel database.
- L'ID della serie TV fornito non esiste nel database.

500 INTERNAL SERVER ERROR - Errore durante l'aggiornamento

- Si è verificato un errore durante l'operazione.

200 OK - Successo

- L'episodio è stato aggiornato correttamente.

//ELIMINAZIONE

```
Route::delete(_VERS . '/delete/film/{idFilm}', [FilmController::class, 'DeleteFilm']);
```

Questa route permette agli **Amministratori** di eliminare un film dal database. L'operazione è protetta da controlli sui permessi e validazione.

1. Autorizzazione:

- **Verifica permessi**: Verifica che l'utente abbia il permesso di eliminare risorse.
- **Controllo ruolo**: Controlla che l'utente sia un **Amministratore**.

2. Identificazione del film:

- **Ricerca nel database**: Cerca il film nel database utilizzando l'ID del film passato come parametro.
- **Film non trovato**: Se il film non viene trovato, restituisce un errore **404 Not Found**.

3. Eliminazione del film:

- **Rimozione dal database**: Se il film viene trovato, viene eliminato dal database.

4. Gestione degli errori:

- **Permesso negato**: Se l'utente non ha il permesso di eliminare, restituisce un errore **403 Forbidden**.

- **Errore durante la cancellazione**: Se si verifica un errore durante l'eliminazione, restituisce un errore **500 Internal Server Error**.

5. Risposta di successo:

- Se l'operazione di eliminazione è riuscita, restituisce una risposta **204 No Content**.

Tipi di Response:

403 FORBIDDEN - Permessi insufficienti

- L'utente non ha i permessi per eliminare risorse o non è un Amministratore.

404 NOT FOUND

- L'ID del film fornito non corrisponde a nessun film nel database.

500 INTERNAL SERVER ERROR - Errore durante l'eliminazione

- Si è verificato un errore durante l'operazione di eliminazione.

204 NO CONTENT - Successo

- Il film è stato eliminato correttamente.

```
Route::delete(_VERS . '/delete/serietv/{idSerietv}', [SerieController::class, 'DeleteSerieTv']);
```

Questa route consente agli **Amministratori** di eliminare una serie TV dal database in modo sicuro. Sono previsti controlli di autorizzazione e validazione per garantire che solo utenti abilitati possano eseguire l'operazione.

1. Autorizzazione

- Verifica che l'utente abbia il permesso di eseguire l'operazione di eliminazione.
- Controlla che l'utente sia un **Amministratore**.
  In caso contrario, restituisce un errore **403 FORBIDDEN**.

2. Identificazione della Serie TV

- Ricerca la serie TV tramite l'idSerietv fornito come parametro.
- Se la serie TV non esiste:

- Restituisce un errore **404 NOT FOUND** con un messaggio: *"Serie TV non trovata"*.

3. Eliminazione

- Se la serie TV viene trovata:
  - La rimuove dal database.
  - L'operazione include la cancellazione di eventuali relazioni collegate.

4. Gestione degli errori

- In caso di permesso negato:
  Errore 403 con un messaggio: *"Non hai i permessi per accedere a questa risorsa"*.
- Se si verifica un problema tecnico durante l'eliminazione:
  Errore 500 con un messaggio: *"Errore durante l'eliminazione"*.

5. Risposta di Successo

- Se l'eliminazione avviene correttamente:
  - Restituisce una risposta **204 NO CONTENT** senza corpo.

Tipi di Response

403 FORBIDDEN - Permessi insufficienti

- L'utente non ha i permessi per eliminare risorse o non è un Amministratore.

404 NOT FOUND - Risorsa non trovata

- L'ID della serie TV fornito non corrisponde a nessuna serie TV nel database.

500 INTERNAL SERVER ERROR - Errore durante l'eliminazione

- Si è verificato un errore durante l'operazione di eliminazione.

204 NO CONTENT - Successo

- La serie TV è stata eliminata correttamente.

```
Route::delete(_VERS . '/delete/serietv/{idSerietv}/episodio/{idEpisodio}', [episodiController::class, 'DeleteEpisodio']);
```

Questa route permette agli Amministratori di eliminare un episodio specifico di una serie TV dal database. L'operazione è protetta da controlli sui permessi e validazione.

1. **Autorizzazione**:
    - **Verifica permessi**: Controlla che l'utente abbia il permesso di eliminare risorse.
    - **Controllo ruolo**: Assicura che l'utente sia un Amministratore.
2. **Identificazione della Serie TV e dell'Episodio**:
    - **Ricerca della Serie TV**: Cerca la serie TV nel database utilizzando l'ID della serie TV passato come parametro.
    - **Serie TV non trovata**: Se la serie TV non viene trovata, restituisce un errore 404 Not Found.
    - **Ricerca dell'Episodio**: Cerca l'episodio associato alla serie TV nel database utilizzando l'ID episodio e l'ID serie TV.
    - **Episodio non trovato**: Se l'episodio non viene trovato, restituisce un errore 404 Not Found.
3. **Eliminazione dell'Episodio**:
    - **Rimozione dal database**: Se l'episodio viene trovato, viene eliminato dal database.
4. **Gestione degli errori**:
    - **Permesso negato**: Se l'utente non ha il permesso di eliminare, restituisce un errore 403 Forbidden.
    - **Errore durante la cancellazione**: Se si verifica un errore durante l'eliminazione, restituisce un errore 500 Internal Server Error.
5. **Risposta di successo**:
    - **Eliminazione completata**: Se l'operazione di eliminazione è riuscita, restituisce una risposta 204 No Content.

Tipi di Response

403 FORBIDDEN - Permessi insufficienti

- L'utente non ha i permessi per eliminare risorse o non è un Amministratore.

404 NOT FOUND - Risorsa non trovata

- L'ID della serie TV fornito non corrisponde a nessuna serie TV nel database, oppure l'episodio specificato non è associato alla serie TV indicata.

500 INTERNAL SERVER ERROR - Errore durante l'eliminazione

- Si è verificato un errore durante l'operazione di eliminazione.

204 NO CONTENT - Successo

- L'episodio è stato eliminato correttamente.

//REGISTRAZIONE

```
Route::post(_VERS . '/registrazione', [ContattoController::class, 'store']);
```

Questa route consente di registrare un nuovo utente nel sistema. I dati forniti vengono salvati in più tabelle per separare i dettagli anagrafici, le credenziali di accesso e altre informazioni. L'operazione è eseguita in una transazione per garantire l'integrità dei dati.

1.Creazione del contatto

- Inserisce i dettagli personali dell'utente (nome, cognome, nazione, data di nascita, sesso) nella tabella contatto.

2. Verifica dell'email

- Controlla se l'email fornita è già presente nella tabella auth.
- Se l'email è già utilizzata, restituisce un errore 400 Bad Request.

3.Salvataggio dell'email

- Registra l'email nella tabella auth collegandola al record del contatto creato. L'email è salvata in forma hashata con l'algoritmo SHA-512.

4.Salvataggio della password

- Registra la password nella tabella password collegandola al record del contatto creato. La password è salvata in forma hashata con l'algoritmo SHA-512.

5.Gestione della transazione

- Inizia una transazione per garantire che tutte le operazioni vengano completate correttamente.
- Se si verifica un errore, la transazione viene annullata (rollback).

6.**Risposta di successo**

- Se tutte le operazioni vengono completate correttamente, restituisce una risposta 200 OK.

Tipi di Response

200 OK - Registrazione completata

- L'utente è stato registrato correttamente.

400 BAD REQUEST - Email già utilizzata

- L'email fornita è già presente nel database.

500 INTERNAL SERVER ERROR - Errore durante la registrazione

- Si è verificato un errore durante l'operazione. Il messaggio di errore specifico è incluso nella risposta.

//LOGIN

```
Route::get(_VERS . '/login/{email}/{hash?}', [AuthController::class, 'show']);
```

Questa route gestisce il login dell'utente. Se l'hash della password è presente, verrà confrontato con quella memorizzata nel database. Se l'hash è omesso, verrà avviata una "sfida" per verificare l'utente.

1. **Verifica dell'utente (CheckUtente)**
    - **Controlla se l'email esiste nel database**: La funzione CheckUtente verifica se l'email fornita è presente nella tabella auth.
    - **Generazione del sale**: Se l'email esiste, viene generato un "sale" (hash casuale) per la password.
    - **Generazione di una secretJWT**: Se l'email esiste, viene generato un secretJWT per l'utente, che viene salvato nel database.
    - **Salvataggio del sale e della secretJWT**: Dopo aver creato il sale e la secretJWT, queste informazioni vengono salvate nel database. Inoltre, viene registrato l'inizio della sfida (inizioSfida).
    - **Risposta**: La funzione restituisce il sale in formato JSON.
2. **Verifica della password (CheckPassword)**

- **Controllo dell'esistenza dell'utente**: La funzione CheckPassword verifica se l'email esiste nel database. Se l'utente non esiste, viene restituito un errore.
- **Controllo della scadenza della sfida**: Viene verificato se il tempo della "sfida" è scaduto. Se la sfida è scaduta, viene restituito un errore.
- **Controllo dei tentativi di login**: Se i tentativi di login falliti sono inferiori al limite (maxLoginErrati), si procede al controllo della password.
- **Verifica della password**: Viene confrontato l'hash della password inviata dal client con quella memorizzata nel database. Se i valori corrispondono, il login è considerato valido.
- **Generazione del token JWT**: Se il login è valido, viene creato un token di sessione JWT per l'utente.
- **Gestione dei tentativi falliti**: Se la password è errata, il tentativo fallito viene registrato.
- **Risposta**: Se il login è avvenuto con successo, viene restituito il token in formato JSON. Se il login fallisce o sono stati superati i tentativi, viene restituito un errore.

Tipi di Response

1. **200 OK - Login riuscito**
   - Il login è stato completato con successo.
2. **403 Forbidden - Errore nei dati**

- I dati di login non sono corretti, il numero di tentativi è stato superato, o la sfida è scaduta.
- **Contenuto della risposta**:
  - Se la password è errata: I dati non coincidono
  - Se il numero di tentativi è stato superato: Limite di tentativi raggiunto
  - Se la sfida è scaduta: esempio_errore_B

3. **500 INTERNAL SERVER ERROR - Errore durante il login**

- Si è verificato un errore interno durante il login, ad esempio un errore nel database o nella generazione del token.

- **Contenuto della risposta**:
    - Il messaggio di errore specifico è incluso nel corpo della risposta.