

SiAPI

Definizione di progetto

Contents

1.	INTRODUZIONE	3
2.	ARCHITETTURA API	3
3.	TECNOLOGIE	3
4.	PROCEDURA DI AUTENTICAZIONE	3
5.	CAMBIO PASSWORD	5
6.	DESCRIZIONE DEI SERVIZI	6
7.	SERVIZI AUTENTICATI	6
8.	SERVIZI NON AUTENTICATI	7
9.	CONFIGURAZIONE DEL SERVIZIO	8
10.	LOG	8
11.	SWAGGER	8
12.	PAGINA HTML DESCRITTIVA DEL SERVIZIO	9
13.	GESTIONE CREDENZIALI	9
14.	INSTALLAZIONE	9
15.	COSTRUZIONE DELLE QUERY	10
16.	REFERENZE	10

Versione	Data	Autore	Nota
1.0	21-03-2024	Alessandro Gambaro	Prima versione
2.0	27-03-2024	Alessandro Gambaro	WinForm per conversione SHA512
3.0	28-03-2024	Alessandro Gambaro	Aggiunto cambio password utente
4.0	03-04-2024	Alessandro Gambaro	Api descrizione dei servizi per utente autenticato
5.0	05-04-2024	Alessandro Gambaro	Api query senza autenticazione richiesta e con parametri
6.0	16-04-2024	Alessandro Gambaro	Aggiunti proprietà di help e nome del servizio in url nelle query post
7.0	30-05-2024	Alessandro Gambaro	Aggiuntà proprietà per differenziare visibilità servizi, pagina html dinamicamente generata descrittiva
8.0	20-06-2024	Alessandro Gambaro	Base url impostata in file di configurazione
9.0	20-07-2024	Alessandro Gambaro	Datahub con lables in appsetting.json, aggiunto filtro testuale sui servizi

1. INTRODUZIONE

Lo scopo del progetto è quello di sviluppare delle API che permettano l'esposizione dei dati stored sui database ARPAL.

2. ARCHITETTURA API

L'architettura prevede un server Web basato su IIS con autenticazione basata su JWT che esponga delle API agli utenti che risultano abilitati all'accesso, alcuni servizi possono essere invocati dagli utenti anche se non autenticati.

Le API avranno permesso di accedere al DB Oracle dal quale preleveranno le query scritte da personale ARPAL ed esporranno il risultato in formato Json.

3. TECNOLOGIE

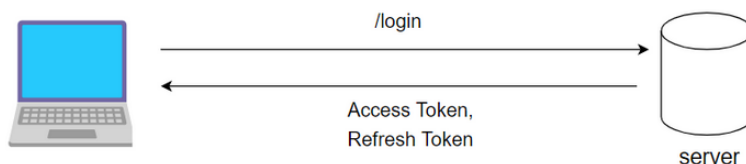
Le tecnologie utilizzate sono le seguenti:

- Microsoft .NET 8.0
- IIS
- JWT
- Json

4. PROCEDURA DI AUTENTICAZIONE

La procedura di autenticazione è basata su **"JWT Authentication with Refresh Token"**, Il JSON Web Token è uno standard aperto per il trasferimento sicuro di dati tra parti utilizzando un oggetto JSON. JWT viene utilizzato per meccanismi di autenticazione senza stato per utenti e fornitori, il che significa mantenere le sessioni sul lato client invece di memorizzarle sul server.

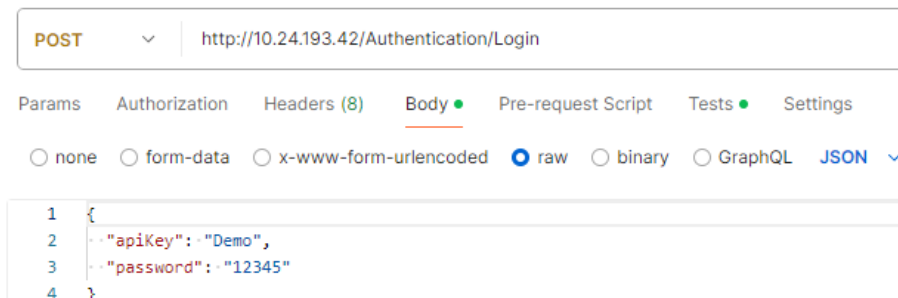
Dovrà essere effettuata una chiamata di autenticazione dalla quale si riceveranno due token



Access Token: con breve validità, contengono i dettagli dell'utente, mentre i token di aggiornamento, memorizzati come cookie HTTP-only, consentono una ri-autenticazione prolungata senza esporre informazioni sensibili al JavaScript lato client.

Refresh Token: un token unico utilizzato per ottenere ulteriori token di accesso. Ciò consente di avere token di accesso a breve durata senza dover raccogliere le credenziali ogni volta che uno scade.

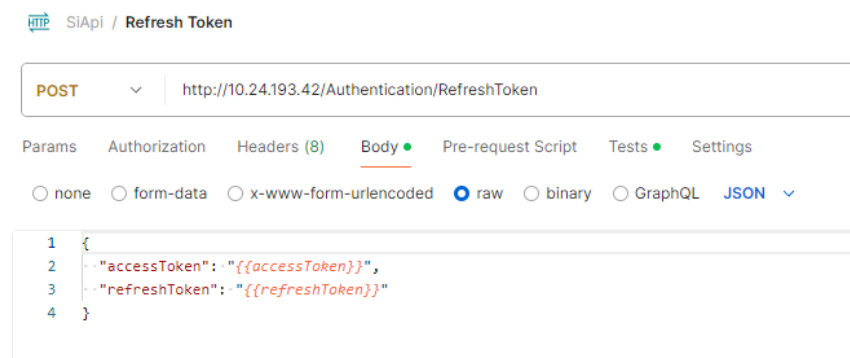
API di Login:



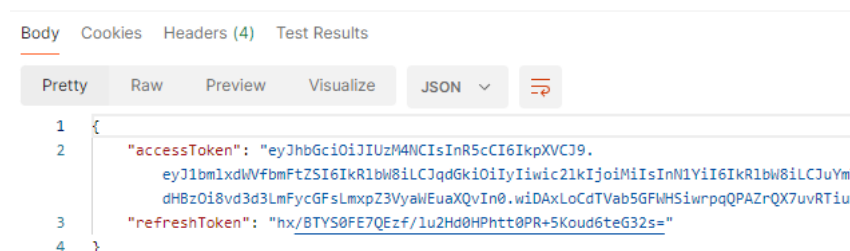
In risposta verranno ritornati (se il login avrà successo) i seguenti due dati

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bm1xdWVfbmFtZSI6Ikp1bm1xdWVfbmFtZSI6IkpXVCJ9.eyJ1bm1xdWVfbmFtZSI6Ikp1bm1xdWVfbmFtZSI6IkpXVCJ9",
  "refreshToken": "6DnW+wS9GmlcRYTeir+5xj3gQUyCGCJ9gWxSy74YU1s="
}
```

API RefreshToken:



La risposta è identica a quella di login:



Ricevuti i nuovi dati di autenticazione, potranno essere utilizzate le API di interrogazione.

L'Access token dovrà essere messo in Header di chiamata in questo formato:

```
--header 'Content-Type: application/json' \  
--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJ1bmRldGF0eWVmbmFtZSI6IHRlbnw8IlClqdGkiOiIyIiwic2lkIjoiaMiIsInN1YiI6IHRlbnw8  
MjIwMCwiawF0IjojoxNzExMDAwLCljc3MiOiJodHRwciovL3d3dy5hcncBbC5saWdIdcm:
```

API RefreshTokenRevoke:

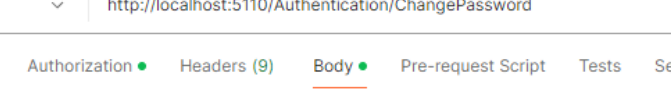
Questa funzionalità serve a revocare il refresh token, per essere richiamata ha bisogno dell'access token ed ha la seguente forma:

The screenshot shows the REST Client interface with the following details:

- Method:** POST
- URL:** http://10.24.193.42/Authentication/RefreshTokenRevoke
- Tab:** Authorization (selected)
- Type:** Bearer Token
- Warning:** Heads up! These parameters hold sensitive data. To keep this data secure, use variables.
- Token:** {{accessToken}}

5. CAMBIO PASSWORD

Un utente può decidere di cambiare la sua password e per farlo viene messa a disposizione la seguente api che vuole in post sia la vecchia che la nuova password, essendo questa una api chiamata da software e non interfaccia grafica non avrebbe senso mettere due volte in un json la conferma password:



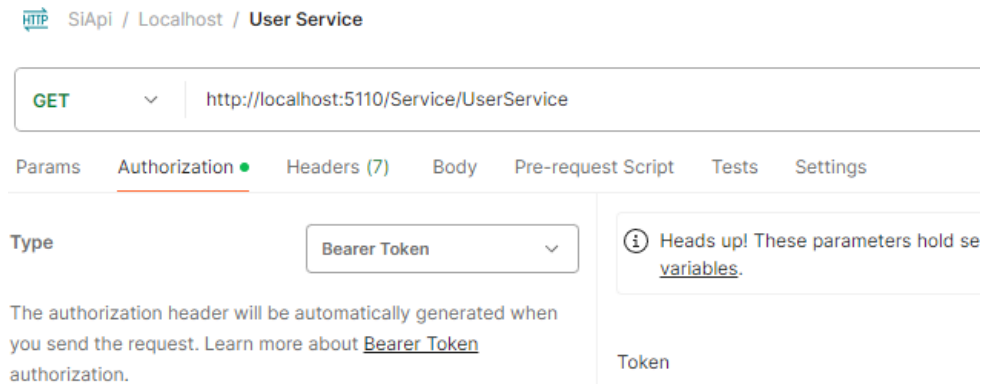
The screenshot shows the REST Client interface. The top bar displays the method **POST** and the URL `http://localhost:5110/Authentication/ChangePassword`. Below this, there are tabs for **Params**, **Authorization**, **Headers (9)**, **Body** (which is selected and underlined), **Pre-request Script**, **Tests**, and **Settings**. Under the **Body** tab, there are radio buttons for **none**, **form-data**, **x-www-form-urlencoded**, **raw** (which is selected), **binary**, and **GraphQL**. To the right of these is a dropdown menu set to **JSON**. The main area shows a JSON body with the following content:

```
1 {  
2   .... "oldPassword": "12345",  
3   .... "newPassword": "12345678m!$"  
4 }
```

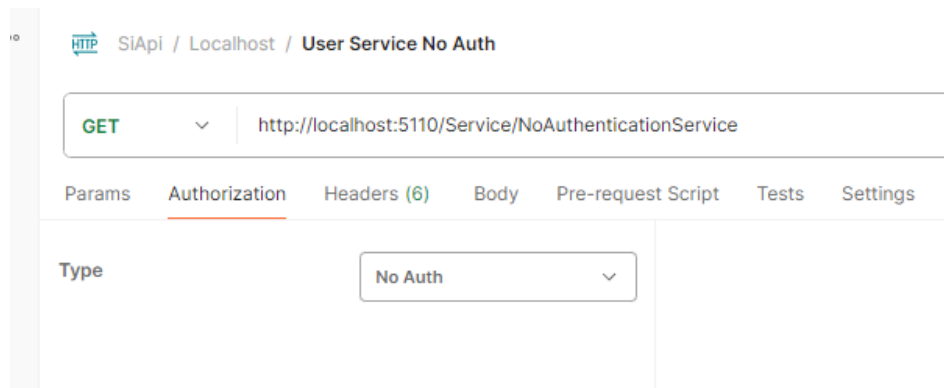
6. DESCRIZIONE DEI SERVIZI

Sono state realizzate due api distinte per non creare confusione, e che permettono ad utenti autenticati e non di vedere la lista dei servizi disponibili complessiva.

Per vedere la lista dei servizi per utente autenticati è necessario passare il token



Servizi per utenti non autenticati, non è necessario passare alcun token



7. SERVIZI AUTENTICATI

Come detto precedentemente, per poter interrogare queste API sarà necessario essere in possesso di un `accessToken` valido e non scaduto, dopo di che basterà effettuare una chiamata a questa API.

POST ▼ http://10.24.193.42/Service/Query/DISNEYTEST_DATE

Params Authorization ● Headers (9) **Body ●** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```

1 {
2   .."parametri":..[
3     ....{
4       ...."alias":.."DATA",
5       ...."value":.."2024-03-15"
6     ....}
7   ..]
8 }

```

Una attenzione particolare va data ai parametri che devono rispettare le regole di API, e quindi avere un alias ed un valore.

Il valore può assumere diversi formati a seconda del tipo che si vuole filtrare:

- **DATA:** ISO 8601 combined date and time format (e.g. 2024-10-20T23:45:34).
- **INTEGER:** un numeri interi
- **TEXT:** un testo a piacere
- **DECIMAL:** un numero decimale in formato inglese con separatore decimale il . e non la virgola (e.g. 123.45)

8. SERVIZI NON AUTENTICATI

Alcuni servizi possono essere utilizzati senza bisogno di autenticazione o ApiKey, questo dipende da come il servizio è stato configurato.

La chiamata al servizio sarà in GET, non ha bisogno di autenticazione, tutti i parametri sono messi in URL, ed ha una forma di questo tipo:

GET ▼ http://localhost:5110/Service/Query/DISNEYTEST_TEXT?apiKey=Demo&PERSONAGGIO=minnie

Params ● Authorization Headers (6) **Body** Pre-request Script Tests Settings

Query Params

<input checked="" type="checkbox"/> Key	Value
<input checked="" type="checkbox"/> apiKey	Demo
<input checked="" type="checkbox"/> PERSONAGGIO	minnie
<input type="checkbox"/> Key	Value

I parametri hanno gli stessi nome e forme di dato supportato di quelli delle query autenticate (si rimanda al precedente paragrafo per vedere).

9. CONFIGURAZIONE DEL SERVIZIO

Il servizio è dotato di file di configurazione, dove poter impostare diversi valori, si trova in questo file depositato dentro la cartella del servizio:

- appsettings.json

Contiene:

- **BaseUrl** → bisogna indicare l'url di base da utilizzare nella pagina base per comporre gli indirizzi, questo perché il redirect che attualmente viene fatto non fa arrivare al portale l'indirizzo.
- **ConnectionStrings** → **SiApi** Credenziali di accesso al database
- **FiltroVisibilitaServizi**: di valore testuale, filtra i servizi che contengono la parola desiderata dentro la colonna filtro, il filtro è case insensitive e se non valorizzato non imposta filtro.
- **Jwt** → **ExpiresInMinutes**: durata in minuti del Access Token
- **Jwt** → **RefreshTokenExpiresInMinutes**: durata in minuti del Refresh token
- **Labels**: contiene i testi della pagina dataHub.
- **Swagger** → **Enabled**: se abilitare swagger

10. LOG

L'applicativo prevede l'utilizzo di file di log dove poter verificare e controllare utilizzo dell'API, la configurazione del file di log si trova in questo file depositato dentro la cartella del servizio:

- log4net.config

La libreria base utilizzata è Log4Net agganciato con injection a dotNet Core.

11. SWAGGER

C'è la possibilità di abilitare la funzionalità Swagger, che mostra un dettaglio delle api stesse in diversi formati:

- **Web Grafico**: <http://10.24.193.42/swagger/index.html>
- **Json**: <http://10.24.193.42/swagger/v1/swagger.json>
- **Yaml**: <http://10.24.193.42/swagger/v1/swagger.yaml>

Per abilitare questa funzionalità è necessario impostare a true il parametro Swagger → Enabled in appsettings.json.

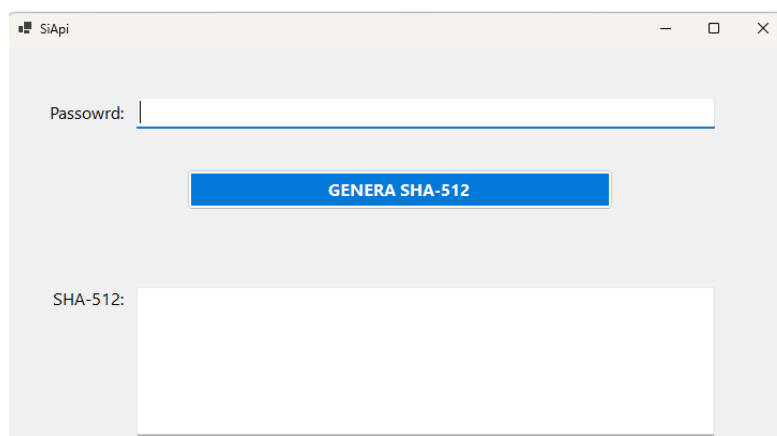
12. PAGINA HTML DESCRITTIVA DEL SERVIZIO

Oltre ad avere Swagger, è stata realizzata una pagina HTML dinamicamente generata che mostra e descrive tutti i servizi pubblici, il link da utilizzare ha la seguente forma:

- **Indirizzo:** <http://10.24.193.42/DataHub>

13. GESTIONE CREDENZIALI

Le credenziali sono memorizzate nella tabella “**SIAPI_APIKEYS**”, le password non vengono storate in chiaro ma sottoforma di SHA512, per questo motivo è stato sviluppato un applicativo Windows Form che permette di convertire una stringa in SHA512.



L'algoritmo non è reversibile e quindi non è possibile risalire alla password partendo dallo SHA512. Quando si vuole impostare una password è quindi necessario scriverla in password e cliccare genera, dopo di che nel database dovrà essere messo lo SHA512 generato e non la password stessa.

14. INSTALLAZIONE

L'installazione della WebApi è semplice ma richiede alcuni passi fondamentali; vedremo adesso l'installazione su macchina windows.

Requisiti:

- Windows Server compatibile con DotNet Core 8
- IIS installato
- [Dotnet Core 8 hosting](#)
- [Dotnet Core 8 Runtime \(dove si vuole utilizzare la WinForm per le credenziali\)](#)

Una volta installati i requisiti, è necessario creare un sito Web dentro IIS e farlo puntare alla folder dove sono stati messi i file compilati della applicazione Arpal.SiApi.WebApplication.

In conclusione non resta che configurare i due file appsettings.json e log4net.config.

15. COSTRUZIONE DELLE QUERY

Le query devono essere costruite in modo coerente con la definizione del servizio sul database, quindi se la query è parametrica, tutti i parametri devono essere configurati nella tabella dei parametri del servizio in modo coerente.

Se si vuol far sì che un parametro sia non mandatorio è necessario impostare a N nella colonna MANDATORY di SI-API_SERVIZI_PARAMETRI e poi nella query utilizzarlo in or con la condizione di null in questo modo:

```
( :COGNOME IS NULL OR COGNOME = :COGNOME )
```

Questo perchè lato applicativo le query non vengono editate ma vengono passati i soli parametri in modo dinamico, senza sapere come sia fatta la query stessa che potrebbe avere una complessità alta a piacere di chi la ha scritta.

16. REFERENZE

- <https://jwt.io/>
- <https://dotnet.microsoft.com/en-us/download>
- <https://en.wikipedia.org/wiki/JSON>