# Indice dei contenuti

1. Descrizione del servizio
2. Sicurezza
3. Risorse disponibili
   1. /exams
   2. /instructors
   3. /students
   4. /users

# Descrizione del servizio

Student-RESTful-API è un servizio basato sul tipico paradigma client/server, che permette, attraverso il protocollo http, di manipolare dei dati relativi a Studenti/Insegnati/Esami.

Questo servizio è stato reso più pratico attraverso l’utilizzo del tool swagger (OpenAPI) che genera un file di specifica indipendente dal linguaggio utilizzato nell’ API.

Una volta eseguito il servizio, è possibile consultare l’interfaccia di swagger tramite il percorso **/swagger/v1/** .

La comunicazione del servizio è [stateless](https://www.redhat.com/it/topics/cloud-native-apps/stateful-vs-stateless), quindi non prevede la memorizzazione delle informazioni del client; ogni richiesta è distinta e non connessa. Le risorse possano essere manipolate dal client tramite un’interfaccia uniforme che prevede l’utilizzo dei 4 principali metodi supportati dal protocollo http.

Infine, la persistenza delle risorse è garantita tramite l’utilizzo di un database di tipo SQLite, dove sono salvate sotto forma di tuple.

# Sicurezza

Il servizio Student-RESTful-API soddisfa le proprietà di Confidenzialità, Integrità, Autenticazione e Autorizzazione, in particolar modo:

**Confidenzialità** ed **Integrità** sono garantite dall’utilizzo del protocollo https che è basato a sua volta sul protocollo crittografico TLS.

TLS permette una comunicazione sicura dalla sorgente al destinatario (end-to-end) su reti TCP/IP fornendo un’autenticazione unilaterale (solo il server si autentica), integrità dei dati e confidenzialità operando al di sopra del livello di trasporto.

**Autenticazione** è garantita tramite un controllo custom e l’utilizzo di unJWT (JSON Web Token).

La richiesta di autenticazione viene effettuata tramite una specifica chiamata **/users/authenticate** di tipo POST,la quale contiene, nel corpo del messaggio, un modello (*AuthenticateRequest.cs*) con le credenziali dell’utente in formato JSON.

Il core dell’autenticazione è presente nel *UserService.cs,* dove vengono confrontate le credenziali fornite con quelle presenti sul database. Una volta confrontato l’username e l’hash della password, se l’utente è registrato nel database viene generata una risposta (*AuthenticateResponse.cs*) con un nuovo JW, altrimenti viene ritornato un messaggio di errore.

Infine,l’**autorizzazione** viene gestita principalmente dal *JwtMiddleware.cs* che “intercetta” le richieste che necessitano una autorizzazione e ne verifica la compatibilità. Il *JwtMiddleware* autorizza o meno una richiesta tramite l’utilizzo dell’header *Authorization* che contiene il JWT token precedentemente calcolato. Nel caso in cui il token non e valido o è semplicemente scaduto, il servizio risponde con un errore del tipo 401 Unauthorized.