



Aruba Authentication System

04/07/2023

—

Marco Dalai

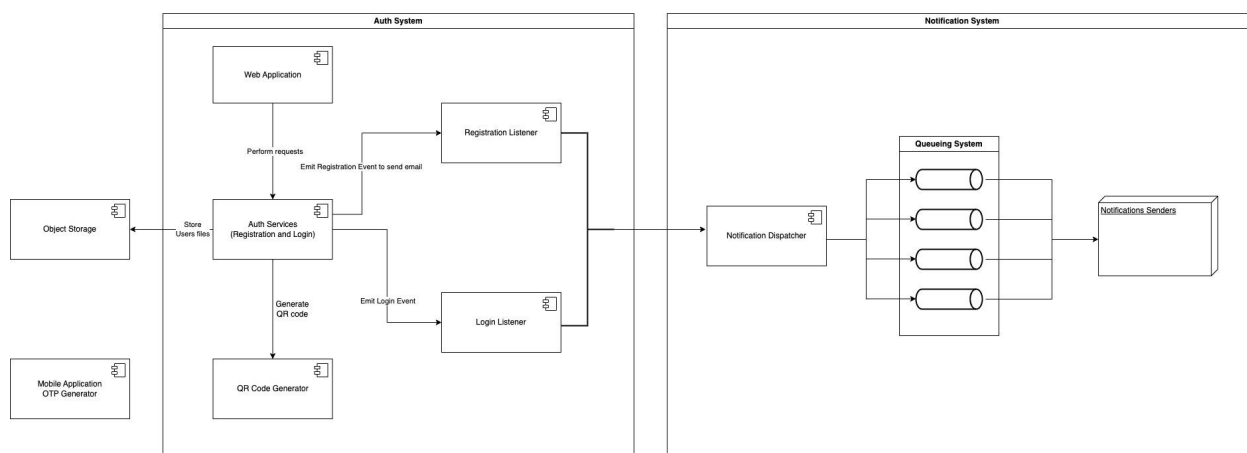
Panoramica

Questo documento ha lo scopo di presentare la soluzione per il sistema di autenticazione richiesto per la prova pratica richiesta da Aruba.

Obiettivi

1. Presentare la soluzione proposta.
2. Mostrare le interconnessioni tra gli elementi del sistema.

Specifiche




Il sistema si compone di diversi elementi che comunicano tra loro per raggiungere l'obiettivo finale. Di seguito vengono riportati gli elementi fondamentali e come comunicano tra loro.

Web Application

La **Web Application** ha il compito di fornire all'utente le interfacce attraverso cui può svolgere le operazioni richieste.

Auth Services

Questo servizio è responsabile di ricevere le richieste dalla **Web Application** ed eseguire le operazioni necessarie per completare le richieste che arrivano dall'utente. Questo servizio comunica con l'OTP Generator per ottenere il codice OTP da inviare all'utente quando viene richiesta l'autenticazione a due fattori. Il servizio ha anche il compito di emettere eventi al



verificarsi di determinati eventi: in particolare, emette eventi in caso di registrazione utente, richiesta di OTP e login. Compito di questo servizio è anche di assicurare che i dati siano persistiti nel sistema di storage. L'**Auth Services** genera anche il token JWT per completare la registrazione. Al momento della registrazione, viene inviata una mail con il token presente nell'url. Il token contiene le informazioni essenziali dell'utente in modo da fare il match con il codice fiscale fornito dall'utente in fase di conferma.

Registration Listener

Quando viene completato il processo di registrazione, l'**Auth Services** emette un evento che indica l'avvenuta registrazione dell'utente. A questo punto è necessario inviare una mail contenente le istruzioni per validare la registrazione dell'utente. L'evento contiene le informazioni necessarie ad individuare l'utente nel database e creare il messaggio da inviare nella mail. La richiesta di invio viene inserita nel **Notification System**.

Login Listener

Il **Login Listener** è simile al **Registration Listener**, ma recepisce gli eventi emessi dal login. Anche lui inserisce gli eventi nel **Notification System**.

QR Code Generator

Il **QR Code Generator** ha il compito di generare il codice QR che contiene il **secret** dell'utente per generare il codice OTP. Questo deve essere scannerizzato sull'app mobile per memorizzarlo e generare i codici **TOTP**.

Object Storage

L'**Object Storage** è il componente che si occupa di conservare i documenti caricati dall'utente in fase di registrazione. Sarà organizzato in un bucket contenente una cartella per utente. Visto che il codice fiscale rappresenta un utente in modo univoco, le cartelle saranno nominate con il codice fiscale. Il sistema sarà accessibile attraverso interfaccia web.

OTP Generator

L'**OTP Generator** ha il compito di generare gli OTP per permettere l'accesso. Questi vengono inseriti nel form di login quando l'utente ha selezionato l'opzione **2FA**. Viene installato su un dispositivo mobile dell'utente ed entra in funzione dopo aver scannerizzato il QR code che contiene il secret dell'utente.

Notification Dispatcher

Il **Notification Dispatcher** si occupa di ricevere le richieste di invio di notifica ed inviarle alle rispettive code per essere consumate dai Notifications Sender. Le code sono gestite dal **Queueing System**.

Queueing System

Il **Queueing System** è il sistema che riceve le richieste di invio delle notifiche. Ogni tipologia di invio notifica rappresenta una coda nel sistema. Mettendo le richieste di invio di notifiche in un sistema a coda e deferendo il loro invio, è possibile scaricare il sistema principale da operazioni gravose, permettendo la loro esecuzione out-of-bound.

Notifications Senders

I **Notification Senders** sono gli elementi che si occupano di ricevere i messaggi dalle code delle notifiche e di inviarli al destinatario secondo il metodo selezionato.

Stack Tecnologico

Ogni **System** rappresenta un servizio composto dagli elementi presenti all'interno del perimetro.

Lo stack tecnologico scelto per implementare la soluzione è il seguente:

- **Auth System:** Spring Boot versione 3, Java 17 e PostgreSQL. Spring Boot è stato scelto come framework di riferimento grazie ai suoi molteplici vantaggi: Dependency Injection, Inversion of Control, ecc... L'applicativo espone interfacce web a reinderizzazione server side attraverso il template engine **Thymeleaf**. Si è deciso di utilizzare questa soluzione perché le pagine di registrazione e login non richiedono un'interazione ed uno scambio di informazioni tali da giustificare l'utilizzo di un framework JavaScript. Come DBMS si è deciso di utilizzare PostgreSQL. L'interfacciamento tra l'applicativo ed il DBMS verrà realizzato utilizzando il JPA, in particolar modo l'estensione Spring Data JPA.
- **Object Storage:** MinIO Object Storage. MinIO è stato scelto come Object Storage per salvare i documenti caricati dall'utente in fase di registrazione. MinIO fornisce un'interfaccia compatibile con AWS S3, questo significa che, nel caso in cui fosse necessario utilizzare lo storage S3, questo può essere utilizzato senza dover intervenire sul codice dell'applicativo. Inoltre fornisce alte performance ed è cloud native.

- **Notification System:** Spring Boot 3, Java 17. Il Notification System sarà realizzato utilizzando Spring Boot 3 e Java 17. Spring Boot dispone di molte integrazioni che lo rendono ideale per ricevere i messaggi da Apache Kafka ed inoltrarli all'utente attraverso il canale selezionato.
- **Queueing System:** Apache Kafka. Per implementare il queueing system si è deciso di utilizzare Apache Kafka. Ogni tipologia di notifica avrà il proprio topic dedicato. Il numero di partizioni per topic verrà valutato ed aggiustato a seconda delle necessità e del volume di richieste che dovranno essere gestite. Ogni tipologia di notifica avrà i propri consumer dedicati raggruppati in consumer group. Il numero di consumer verrà regolato in base al numero di partizioni, ricordando che il parallelismo fornito da Kafka non dipende dal numero di consumer nel consumer group, ma dal numero di partizioni.
- **OTP:** TOTP (Time-based One Time Password). La tipologia di OTP selezionata per implementare il **2FA** è la TOTP, che permette di avere token di breve durata e, quindi, più sicuri.
- **OTP Generator:** App mobile. L'app mobile viene installata su un dispositivo mobile dell'utente e viene utilizzata per produrre **TOTP** (Time-based One Time Password). Compito dell'app è memorizzare il secret dell'utente scansando il QR generato quando viene selezionata la modalità **2FA** per effettuare il login.

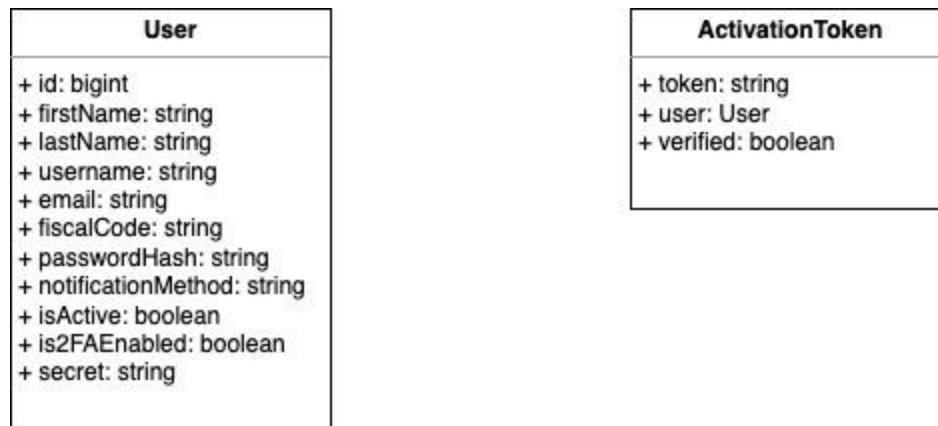
La comunicazione tra **Auth System** e **Notification System** rappresenta un elemento cruciale per il funzionamento dell'applicativo. Per ottimizzarla e renderla efficiente si è deciso di utilizzare il protocollo **gRPC**, che garantisce questo attraverso l'utilizzo dei **Protocol Buffers** e del protocollo **HTTP 2.0**.

Per generare i token di attivazione, si è deciso di utilizzare la tecnologia **JWT** con token che hanno 24 ore di scadenza.

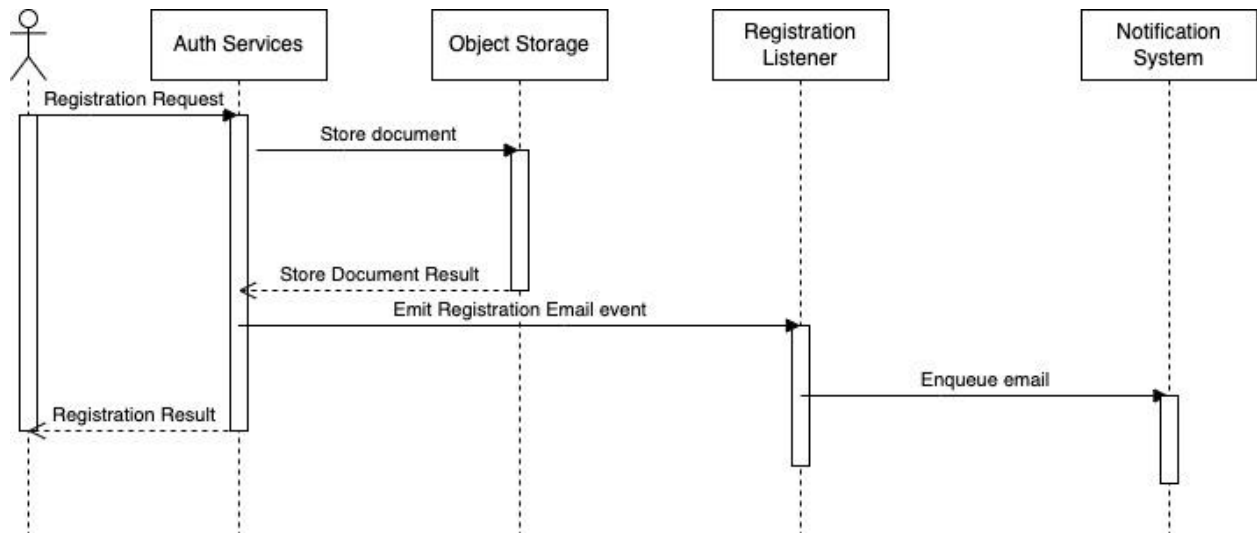
Logback è stato scelto come framework per gestire il logging, esposto attraverso l'interfaccia **SLF4J**. Nelle righe di log potrebbero comparire informazioni sensibili dell'utente: per evitare questo problema e rispettare le direttive del GDPR, verrà implementata un'estensione di **Logback** che permetterà, attraverso l'utilizzo di regular expressions, di trovare le informazioni dell'utente ed offuscarle nelle righe di log.

Diagrammi

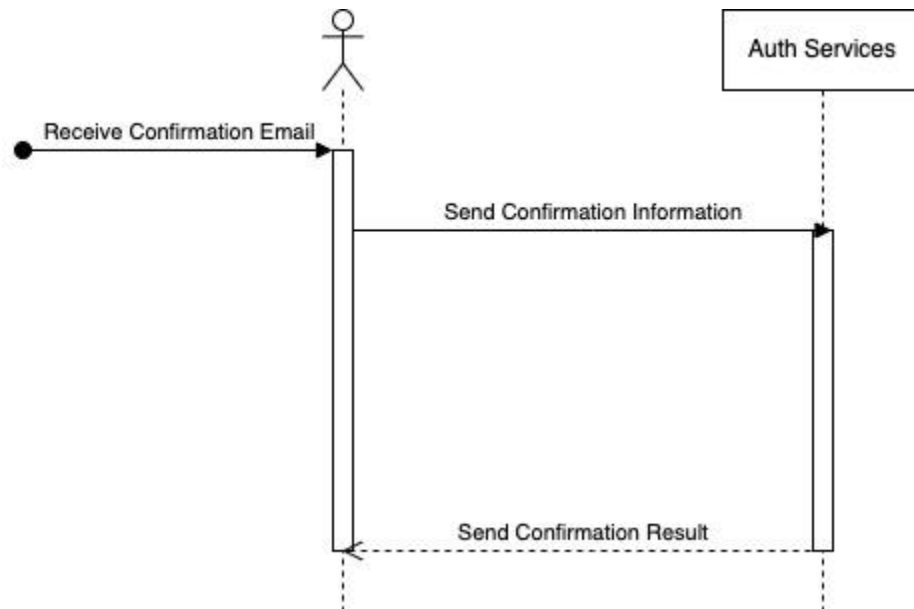
UML Class Diagram



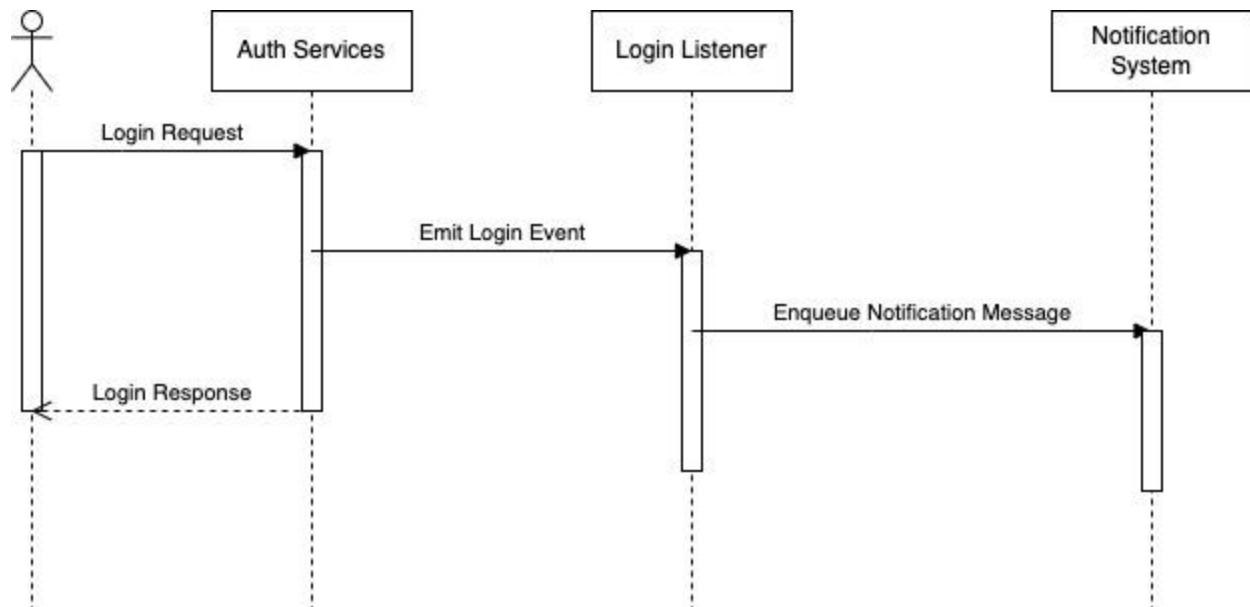
UML Sequence Diagram: Registration



UML Sequence Diagram: Confirm Registration



UML Sequence Diagram: Login



Essendo stata utilizzata la tecnologia **TOTP**, questo diagramma descrive sia il login a singolo fattore che a doppio fattore siccome il token viene generato dall'app mobile e viene aggiunto ai parametri della richiesta di login.