

*Polo territoriale di Como*

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL’INFORMAZIONE

INGEGNERIA INFORMATICA

**Corso di Ingegneria del Software**

****

**AuthOK**

**PROGETTO DEL CORSO DI INGEGNERIA DEL SOFTWARE**

**Parte III – Java**

Ferrario Stefano

Gumus Tayfun

Isella Paolo

Martinese Federico

Indice

[Introduzione 3](#_Toc501630026)

[Authorizer 4](#_Toc501630027)

[Gestore Autorizzazioni 4](#_Toc501630028)

[**Gestore autorizzazioni** 4](#_Toc501630029)

[**Autorizzazione** 4](#_Toc501630030)

[Gestore Token 4](#_Toc501630031)

[**Gestore token** 4](#_Toc501630032)

[Token 5](#_Toc501630033)

[Gestore Risorse 5](#_Toc501630034)

[**Gestore risorse** 5](#_Toc501630035)

[**Risorsa** [abstract] 5](#_Toc501630036)

[Jsonrpc 6](#_Toc501630037)

[Class 6](#_Toc501630038)

[Zeromq 7](#_Toc501630039)

[Class 7](#_Toc501630040)

[User 8](#_Toc501630041)

[Class 8](#_Toc501630042)

**Introduzione**

L’implementazione del codice Java di AuthOK è la parte finale del processo di realizzazione del software. Essa ha il compito di dar vita al programma reale che dovrà rispettare tutti i modelli precedentemente realizzati, a partire dal Goal Diagram fino all’ultimo dei diagrammi UML. Uno alla volta, tutti i metodi sono stati implementati e integrati tra di loro al fine di realizzare la logica applicativa del programma. L’implementazione prevede la creazione delle diverse classi suddivise nei package “Authorizer”, “User”, “Jsonrpc” e “Zeromq”.

**Authorizer**

## **Gestore Autorizzazioni**

Il package GestoreAutorizzazioni contiene le classi GestoreAutorizzazioni e Autorizzazione. Include inoltre una classe AuthorizationException che estende la classe Exception per la gestione delle eccezioni.

### **Gestore autorizzazioni**

La classe Gestore Autorizzazioni implementa tutti i metodi necessari per le operazioni che vengono eseguite sulle autorizzazioni. Il costruttore di questa classe utilizza il design pattern singleton per garantirne l’univocità. Durante l’instanziazione di un nuovo gestore viene creata una nuova HashMap autorizzazioni che conterrà le nuove autorizzazioni create per consentirne il salvataggio e le successive verifiche su di essi. I principali metodi implementati in questa classe sono:

* genera\_chiave\_unica
* creaAutorizzazione
* revocaAutorizzazione
* verificaEisitenzaAutorizzazione
* verificaValiditàAutorizzazione

### **Autorizzazione**

La classe Autorizzazione ha al suo interno il costruttore necessario per la creazione di un nuovo token e i metodi per recuperarne le sue informazioni. Ogni autorizzazione è costituita da un nome utente, un livello di accesso e una scadenza.

## **Gestore Token**

Come nel precedente caso anche nel package GestoreToken sono presenti una classe “gestore” e una classe Token. Ancora una volta estendiamo la classe Exceprion per la generazione di TokenExceptions

### **Gestore token**

La classe GestoreToken implementa tutti i metodi necessari per le operazioni che vengono eseguite sui token. Il costruttore di questa classe utilizza il design pattern singleton per garantirne l’univocità. Durante l’istanziazione di un nuovo gestore viene creata una nuova HashMap che conterrà i token creati per consentirne il salvataggio e le successive verifiche su di essi. I metodi implementati in questa classe sono:

* creaToken
* verificaToken
* cancellaTokenScaduti
* cancellaTokenChiave

## **Token**

La classe Token ha al suo interno il costruttore necessario per la creazione di un nuovo token e i metodi per recuperarne le sue informazioni.

## **Gestore Risorse**

Per la creazione delle risorse si ricorre a un design pattern factory method. Pertanto per ogni tipo di risorsa vi sarà una classe “risorsa” e una classe “factory” dedicata. Per facilitare la gestione dei tipi di risorsa disponibili abbiamo introdotto un “ENUM”. Come nel caso precedente sono presenti anche qui una classe “gestore”, contenente i diversi metodi per la creazione e per le operazioni sulle risorse, e una classe ResourceException che estende la classe Exception.

### **Gestore risorse**

La classe GestoreRisorse presenta un costruttore di tipo singleton che all’instanziazione crea un’HashMap databaseRisorse la quale conterrà i dati relative alle risorse create. Il GestoreRisorse si occupa della creazione, modifica e cancellazione di una risorsa e della verifica dell’esistenza di una risorsa dato l’ID.

### **Risorsa** [abstract]

La classe Risorsa è di tipo abstract e viene estesa, nel nostro caso specifico dalle classi RisorsaFibonacci, RisorsaLanciaDado, RisorsaLink. Oltre al costruttore tale classe contiene anche i metodi necessari per recuperare e modificare le informazioni che costituiscono una risorsa ovvero il livello, l’ID.

**Jsonrpc**

## **Class**

testo

**Zeromq**

## **Class**

testo

**User**

## **Class**

testo

**Identificativo gruppo**: 10

**Progetto**: Tema A

**Componenti**:

845386 Ferrario Stefano [stefano6.ferrario@mail.polimi.it](mailto:stefano6.ferrario@mail.polimi.it)

843994 Gumus Tayfun [tayfun.gumus@mail.polimi.it](mailto:tayfun.gumus@mail.polimi.it)

854412 Isella Paolo [paolo.isella@mail.polimi.it](mailto:paolo.isella@mail.polimi.it)

845326 Martinese Federico [federico.martinese@mail.polimi.it](mailto:federico.martinese@mail.polimi.it)