



Laurea Triennale in Informatica - Università di Salerno -
Sistema di autenticazione sulla rete Blockchain Ethereum



KRYPTOAUTH

Object Design Document KryptoAuth

Riferimento	
Versione	1.0
Data	29/07/2022
Presentato da	Montefusco Alberto



Sommario

Sommario	2
1. Introduzione	3
1.1 Object Design Goals	3
1.2 Componenti off-the-shelf	5
2. Packages	6
2.1 Interface	7
2.2 Application Logic	8
2.3 Storage	9
3. Class Interfaces	10

1. Introduzione

L'Object Design Document illustra i diversi dettagli legati alla fase implementativa del sistema KryptoAuth; in particolare, esso descrive gli object design goals, i trade-off di progettazione definiti dagli sviluppatori e, infine, la decomposizione dei sottosistemi in packages e classi.

1.1 Object Design Goals

Gli obiettivi di object design posti per il sistema sono:

- **Astrazione:** le interfacce devono essere intuitive e di un alto livello, così da garantire un'implementazione corretta e comprensibile;
- **Modularità:** le unità del Sistema devono essere organizzate in moduli facilmente collegati;
- **Riusabilità:** il riuso del codice deve essere prioritario e verrà fornito attraverso l'ereditarietà e i design pattern.

1.2 Componenti off-the-shelf

Il Sistema utilizzerà i seguenti componenti off-the-shelf:

- **Spring Boot:** framework per sviluppare applicazioni Java in modo produttivo ed efficiente;
- **Solidity:** linguaggio di alto livello orientato agli oggetti per l'implementazione di Smart Contracts.
- **Ganache:** Blockchain di test basata su Ethereum;
- **Metamask:** estensione browser per l'emulazione di un wallet (portafoglio) cifrato e un gateway per le app Blockchain;
- **Web3j:** libreria Java e Android altamente modulare, reattiva e sicura per lavorare con Smart Contracts e integrare con i client (nodi) sulla rete Ethereum;
- **Web3js:** una raccolta di librerie che consentono di interagire con un nodo Ethereum locale o remoto utilizzando HTTP, IPC o WebSocket;
- **Truffle:** ambiente di sviluppo per lavorare con gli Smart Contracts.

2. Packages

In questa sezione viene mostrata la suddivisione del Sistema in package, in base a quanto definito nel documento di System Design. Tale suddivisione è motivata dalle scelte architetturali prese e sottolinea la struttura di directory standard definita da Maven.

2.1 Interface

Questo package contiene i seguenti sub-package e le seguenti classi:

- Package **templates**:
 - Sub-package **error**:
 - **Error404**: viene visualizzata in caso di errore 404;
 - **Error401**: viene visualizzata in caso di errore 401;
 - **Error500**: viene visualizzata in caso di errore 500.
 - Sub-package **page**:
 - **admin**: area personale dell'amministratore;
 - **blog-single**: pagina di informazioni su un topic selezionato precedentemente;
 - **goals**: pagina relativa agli obiettivi che il sistema vuole soddisfare.
 - **index**: homepage del sito;
 - **login**: pagina di login per l'utente e l'amministratore;
 - **register**: pagina di registrazione per l'utente;
 - **roadmap**: pagina contenente le informazioni del sito;
 - **technologies**: pagina relativa alle tecnologie utilizzate.
 - Sub-package **partials**:
 - **Footer**: sezione informativa in fondo alla pagina;
 - **Header**: area contenente il menu di navigazione;
 - **Head**: contiene meta-informazioni che riguardano la pagina in cui tale partial viene richiamato;
 - **popupError**: popup di errore;
 - **popupPrivateKey**: popup per l'inserimento della chiave privata;
 - **popupSuccess**: popup di successo;
 - **popupRevokeRole**: popup per informare che l'account è disattivato.

2.2 Application Logic

Questo package contiene i seguenti sub-package e le seguenti classi:

- Package **contracts**:
 - **Authentication**: classe Java che rappresenta la traduzione dello Smart Contract scritto in Solidity.
- Package **controller**:
 - **BlockchainController**: servlet che si occupa di svolgere e gestire tutte le operazioni riguardanti le transazioni effettuate verso la Blockchain Ganache;
 - **KryptoController**: servlet che si occupa di gestire tutte le chiamate GET alle pagine HTML;
 - **ErrorPageController**: servlet che si occupa di svolgere e gestire parte degli errori http richiamando delle pagine personalizzate.
- Package **service**:
 - **BlockchainServiceImpl**: classe che implementa l'interfaccia BlockchainService contenente i servizi offerti dal sistema, in particolare, tutte le operazioni da effettuare sulla Blockchain;
 - **BlockchainService**: interfaccia che definisce i metodi dei servizi dell'applicazione.
- Sub-package **validator**:
 - **PasswordEqualsValidator**: implementa un validator personalizzato, in particolare, vengono confrontate due password e si verifica se sono uguali o diverse;
 - **PasswordEquals**: qualifier personalizzato.
- Package **smart contract**:
 - Sub-package **contracts**:
 - **Authentication**: smart contract in cui è definita tutta la logica dell'applicazione;
 - **Migration**: smart contract per l'operazione di migrazione sulla Blockchain;
 - Sub-package **migrations**:
 - **1_initial_migration**: file javascript che avvia lo Smart Contract Migration;
 - **2_deploy_contracts**: file javascript che effettua il deploy dello Smart Contract Authentication;
 - **truffle_config**: configurazione della rete Ganache.



2.3 Storage

Questo package contiene i seguenti sub-package e le seguenti classi:

- Package **model**:
 - **User**: classe Java che contiene le informazioni di un particolare utente (“User”, “Admin” oppure un utente che ancora non ha un ruolo definito);
 - **AjaxResponse**: classe Java che raccoglie i messaggi di errore da mostrare nel frontend.

3. Class Interfaces

Di seguito, vengono elencate le interfacce delle classi previste dal Sistema, che si trovano nel package “Application Logic”. Per motivi di leggibilità e chiarezza del documento non verranno riportati le interfacce delle classi degli altri package.

Javadoc di KryptoAuth

Per motivi di leggibilità si è scelto di creare un sito, messo in host tramite GitHub pages, contenente la Javadoc di KryptoAuth. In questo modo, chiunque può consultare la documentazione aggiornata dell'intero Sistema. Di seguito, il link al sito in questione: [Documentazione](#)



Classi nel Package ApplicationLogic

Nome Classe	BlockchainController
Descrizione	Questa classe contiene tutti i metodi POST e le transazioni alla Blockchain Ethereum.
Metodi	<p>+ loginPost(@Valid @ModelAttribute("user") User user, Errors errors, @RequestParam (value = "userAddress") String address, @RequestParam (value= "privateKey") String privateKey, HttpServletRequest request) throws Exception : AjaxResponse</p> <p>+ registerPost(@Valid @ModelAttribute("user") User user, Errors errors, @RequestParam (value = "userAddress") String address, @RequestParam (value= "privateKey") String privateKey, HttpServletRequest request) throws Exception : AjaxResponse</p> <p>+ activeAddress(@RequestParam (value = "address") String address, @RequestParam (value= "role") String role, @RequestParam (value= "status") String status, @RequestParam (value= "privateKey") String privateKey, HttpServletRequest request) throws Exception : AjaxResponse</p> <p>+ disactiveAddress(@RequestParam (value = "address") String address, @RequestParam (value= "role") String role, @RequestParam (value= "status") String status, @RequestParam (value= "privateKey") String privateKey, HttpServletRequest request) throws Exception : AjaxResponse</p> <p>+ renounceAdmin(@RequestParam (value = "address") String address, HttpServletRequest request : AjaxResponse</p> <p>+ renounceRoles(@RequestParam (value = "address") String address, HttpServletRequest request : AjaxResponse</p>
Invariante di classe	/



Nome Classe	KryptoController
Descrizione	Questa classe contiene tutti i metodi GET per la visualizzazione delle pagine e la funzione di logout dell'applicazione.
Metodi	<div>+ home() : String</div> <div>+ roadmap() : String</div> <div>+ goals() : String</div> <div>+ technologies() : String</div> <div>+ blog() : String</div> <div>+ logout(HttpServletRequest request) : String</div> <div>+ register(Model model, HttpServletRequest request) : String</div> <div>+ login(Model model, HttpServletRequest request) : String</div>
Invariante di classe	/

Nome Classe	ErrorPageController
Descrizione	Questa classe contiene permette di gestire gli errori HTTP.
Metodi	<div>+ handleError(Model model, HttpServletRequest request) : String</div>
Invariante di classe	/



Nome Classe	BlockchainServiceImpl
Descrizione	Questa classe contiene tutti i servizi offerti dall'applicazione.
Metodi	<ul style="list-style-type: none">+ loadContract(String account) throws Exception : Authentication+ deploy(String account) throws Exception : Authentication+ isContractLoaded(String address) throws Exception : boolean+ addressEquals(String address) throws Exception : boolean+ isAdmin(String address) throws Exception : boolean+ isUser(String address) throws Exception : boolean+ registerUser(String address, String name, String password) throws Exception : boolean+ loginUser(String address, String name, String password) throws Exception : boolean+ loginAdmin(String address, String name, String password) throws Exception : boolean+ addUser(String address) throws Exception : boolean+ addAdmin(String address) throws Exception : boolean+ removeUser(String address) throws Exception : boolean+ removeAdmin(String address) throws Exception : boolean
Invariante di classe	/



Nome Classe	PasswordEqualsValidator
Descrizione	Questa classe contiene tutti i metodi POST e le transazioni alla Blockchain Ethereum.
Metodi	+ initialize(PassqwordEquals arg0) : void + isValid(Object candidate, ConstraintValidatorContext arg1) : boolean
Invariante di classe	/