Laurea Magistrale in Sicurezza Informatica - Università di Salerno
Corso di *Sicurezza dei Dati* – Prof. C. Esposito, A. De Santis

KRYPTOAUTH

Object Design Document

KryptoAuth

Riferimento	
Versione	1.0
Data	12/02/2023
Destinatario	Prof. C. Esposito Prof. A. De Santis
Presentato da	Montefusco Alberto

Sommario

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 efficienze;
- **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;
- **Pinata**: è un gateway dedicato che permette di reperire i contenuti più velocemente dai nodi IPFS risparmiando larghezza di banda e tempo;
- **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;
    - **rgister**: pagina di registrazione per l'utente;
    - **roadmap**: pagina contenente le informazioni del sito;
    - **technologies**: pagina relativa alle tecnologie utilizzate.

    - Sub-package **page/NFT-marketplace**:
      - **info-nft**: pagina di informazioni di un NFT;
      - **marketplace**: pagina di visualizzazione degli NFT;
      - Sub-package **page/NFT-marketplace/admin**:
        - **create-nft**: pagina di creazione di un NFT;
      - Sub-package **page/NFT-marketplace/user**:
        - **user-profile**: pagina personale di un user.

- 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**:
  - **KryptoNFT**: 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 per la gestione dei ruoli degli utenti registrati nella Web DApp;
    - **KryptoNFT**: smart contract per la gestione del Marketplace di NFT;
  - Sub-package **migrations**:
    - **2_deploy_contracts**: file javascript che effettua il deploy dello Smart Contract KryptoNFT;
  - **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	+ loginPost(@Valid @ModelAttribute("user") User user, Errors errors, @RequestParam (value = "userAddress") String address, @RequestParam (value = "privateKey") String privateKey, HttpServletRequest request) throws Exception : AjaxResponse + registerPost(@Valid @ModelAttribute("user") User user, Errors errors, @RequestParam (value = "userAddress") String address, @RequestParam (value = "privateKey") String privateKey, HttpServletRequest request) throws Exception : AjaxResponse + activeAddress(@RequestParam (value = "address") String address, @RequestParam (value = "role") String role, @RequestParam (value = "status") String status, @RequestParam (value = "addressMetamask") String addressMetamask, HttpServletRequest request) throws Exception : AjaxResponse + disactiveAddress(@RequestParam (value = "address") String address, @RequestParam (value = "role") String role, @RequestParam (value = "status") String status, @RequestParam (value = "addressMetamask") String addressMetamask, HttpServletRequest request) throws Exception : AjaxResponse + adminToUser(@RequestParam (value = "address") String address, @RequestParam (value = "addressMetamask") String addressMetamask, HttpServletRequest request) : AjaxResponse + revokeAdmin(@RequestParam (value = "address") String address, @RequestParam (value = "addressMetamask") String addressMetamask, HttpServletRequest request) : AjaxResponse

+ revokeUser(@RequestParam (value = “address”) String address,
@RequestParam (value = “addressMetamask”) String addressMetamask, @RequestParam (value = “role”) String role, HttpServletRequest request) throws Exception : AjaxResponse

+ flipMarketplace(@RequestParam (value = “flag”) boolean flag,
@RequestParam (value = “addressMetamask”) String addressMetamask, HttpServletRequest request) throws Exception : AjaxResponse

+ adminViewMarketplace(@RequestParam (value = “addressMetamask”) String addressMetamask, HttpServletRequest request) throws Exception : AjaxResponse

+ adminViewNftsOnPinata(HttpServletRequest request) : AjaxResponse

+ createNft(@RequestParam(value = "name") String name,
@RequestParam(value = "category") String category,
@RequestParam(value = "price") String price,
@RequestParam(value = "validUntil") String validUntil,
@RequestParam(value = "sale") String sale,
@RequestParam(value = "addressMetamask") String addressMetamask,
@RequestParam(value = "description") String description,
HttpServletRequest request) throws Exception : AjaxResponse

+ saveNftOnBlockchain(@RequestParam(value = "name") String name,
@RequestParam(value = "category") String category,
@RequestParam(value = "url") String url,
@RequestParam(value = "description") String description,
@RequestParam(value = "price") String price,
@RequestParam(value = "validUntil") String validUntil,
@RequestParam(value = "sale") String sale,
@RequestParam(value = "address") String address,
HttpServletRequest request) throws Exception : AjaxResponse

+ deleteNft(@RequestParam(value = "id") String id,
@RequestParam(value = "address") String address,
HttpServletRequest request) throws Exception : AjaxResponse

	+ deleteNftExpired(@RequestParam(value = "addressMetamask") String address, HttpServletRequest request) throws Exception : AjaxResponse + assignNft(@RequestParam(value = "id") String id, @RequestParam(value = "address") String address, HttpServletRequest request) throws Exception : AjaxResponse + userViewMarketplace(@RequestParam(value = "addressMetamask") String addrMetamask, HttpServletRequest request) throws Exception : AjaxResponse + buyNft(@RequestParam(value = "id") String id, @RequestParam(value = "address") String address, HttpServletRequest request) throws Exception : AjaxResponse + useNft(@RequestParam(value = "id") String id, @RequestParam(value = "address") String address, @RequestParam(value = "choice") String choice, HttpServletRequest request) throws Exception : AjaxResponse + buyFt(@RequestParam(value = "token") String token, @RequestParam(value = "address") String address, HttpServletRequest request) throws Exception : AjaxResponse + sellFt(@RequestParam(value = "token") String token, @RequestParam(value = "address") String address, HttpServletRequest request) throws Exception : AjaxResponse + profileUser(@RequestParam(value = "address") String address, HttpServletRequest request) throws Exception : AjaxResponse
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	+ home() : String + roadmap() : String + goals() : String + technologies() : String + blog() : String + logout(HttpServletRequest request) : String + register(Model model, HttpServletRequest request) : String + login(Model model, HttpServletRequest request) : String + marketplace(Model model, HttpServletRequest request) throws Exception : String + infoNft(@RequestParam("id") String id, Model model, HttpServletRequest request) throws Exception : String + administration(Model model, HttpServletRequest request) throws Exception : String + createNft(HttpServletRequest request) throws Exception : String + userProfile(Model model, HttpServletRequest request) throws Exception : String
Invariante di classe	/

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

Nome Classe	BlockchainServiceImpl
Descrizione	Questa classe contiene tutti i servizi offerti dall'applicazione.
Metodi	+ 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 + getName(String address) throws Exception : String + mintNft(String name, String category, String description, String url, BigInteger price, BigInteger validUntil, BigInteger sale) throws Exception : Boolean + burnNft(BigInteger id) throws Exception : boolean + assignNft(BigInteger id, String address) throws Exception : boolean + buyFt(BigInteger amounts) throws Exception : boolean + sellFt(BigInteger amounts) throws Exception : boolean + buyNft(BigInteger id) throws Exception : boolean

	+ sellNftUser(BigInteger id) throws Exception : boolean + useNft(BigInteger id) throws Exception : boolean + burnNftUser(BigInteger id) throws Exception : boolean + isValidNft(BigInteger id) throws Exception : boolean + flipSaleState(boolean flag) throws Exception : boolean + isMarketplaceActive() throws Exception : boolean + setTokenPrice(BigInteger price) throws Exception : boolean + getTokensUser() throws Exception : BigInteger + getMyNfts_json() throws Exception : JSONArray + getMyNfts_string() throws Exception : String + getNftsAllAdmin() throws Exception : String + getNftById(BigInteger id) throws Exception : String + getNftsAddr(String address) throws Exception : JSONArray + balanceOf(String address, BigInteger id) throws Exception : long
Invariante di classe	/

Nome Classe	**PasswordEqualsValidator**
Descrizione	Questa classe contiene i metodi per verificare le password.
Metodi	+ initialize(PassqwordEquals arg0) : void + isValid(Object candidate, ConstraintValidatorContext arg1) : boolean
Invariante di classe	/