Documento di Analisi e Progettazione

# Scopo del progetto

Il progetto del nostro gruppo ha lo scopo di sviluppare un software per la gestione della compravendita di oggetti all’asta online, il quale dovrà rispettare:

* le indicazioni definite nel documento Indicazioni committente;
* le regole definite nel documento Coding rules.

# Scopo del documento

Questo documento include i punti più importanti delle fasi analisi e progettazione del software includendo:

* per la fase di analisi:
  + i diagrammi UML dei casi d’uso del software;
* per la fase di progettazione:
  + software:
    - un diagramma dei package;
    - cinque diagrammi UML delle classi per ogni package;
    - due diagrammi UML di sequenza per l’interazione tra gli oggetti descritti nel diagramma delle classi;
  + database:
    - un diagramma E/R ristrutturato del database.

# Ruoli

Adnaan Juma: Backend

Mirko Colombo: Database / JDBC

Marwan Mohamed: Front-End

Enzo Calissi: Comunicazione client-server

# Cronologia versioni

|  |  |  |  |
| --- | --- | --- | --- |
| Autori | Data | Versione | Descrizione |
| Adnaan Juma, Colombo Mirko | 02/04/2024 | 0 | Prima stesura del documento |

# Analisi

## Utente



## Admin



# Progettazione

## Software

### Diagramma dei package



Figura 1 - Diagramma dei package

### Diagramma delle classi package aste

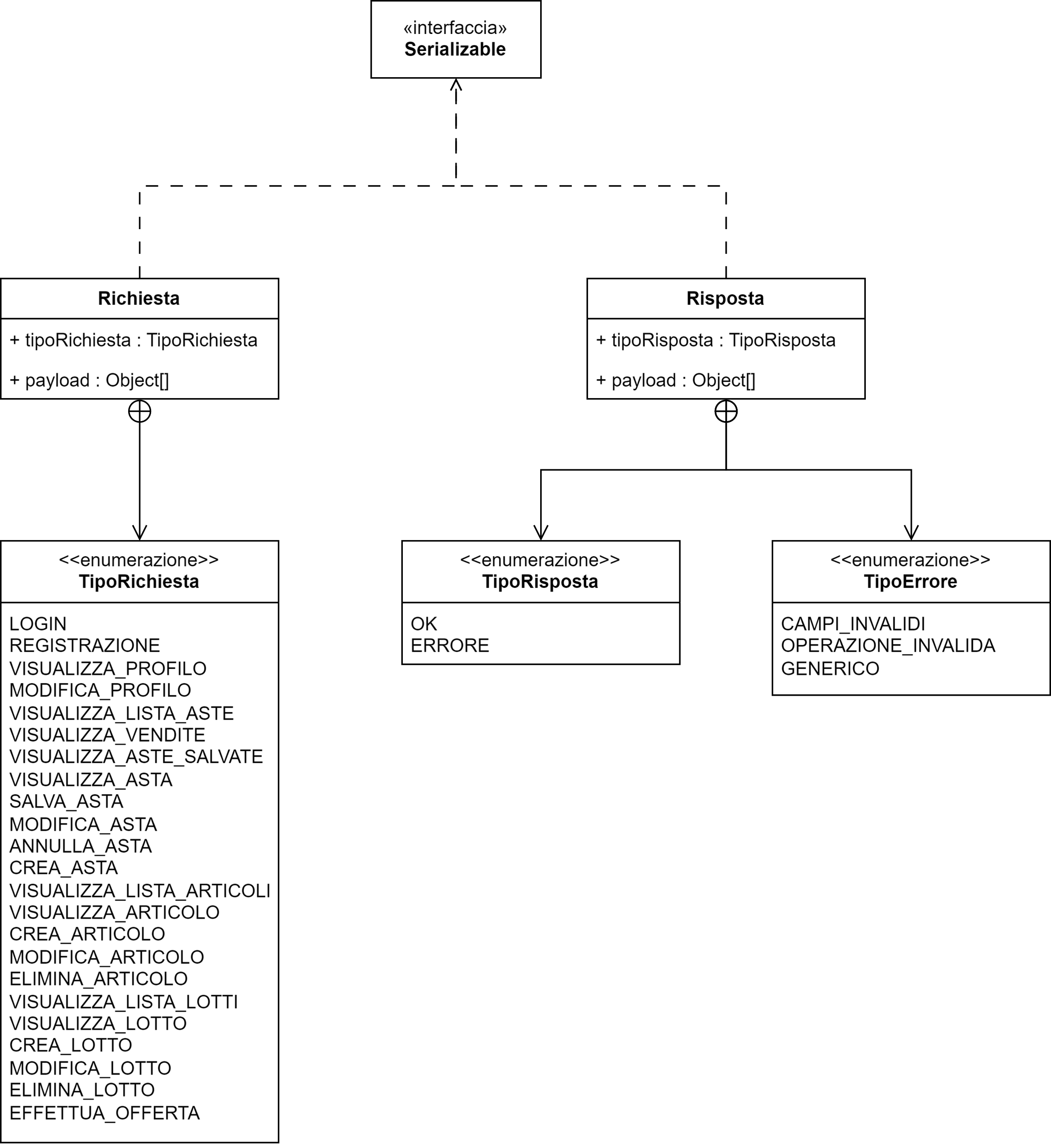


Figura 2 - Diagramma delle classi package aste

### Regole di comunicazione

#### Login

Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.LOGIN** e **payload** così costruito:

1. email : String
2. password : String

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: le credenziali di login sono corrette si può procedere con l’accesso. In questo caso il **payload** contiene l’id\_utente;
* **TipoRisposta.ERRORE**: l’accesso viene rifiutato e il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se la combinazione di email e password è errata.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Registrazione

Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.REGISTRAZIONE** e **payload** così costruito:

1. nome : String
2. cognome : String
3. Password : String
4. dataNascita : Date
5. cittaResidenza : String
6. cap : Integer
7. indirizzo : String
8. email : String
9. iban : String

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: le credenziali di registrazione sono corrette si può procedere con la registrazione. In questo caso il **payload** sarà vuoto;
* **TipoRisposta.ERRORE**: la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se l’email a una sintassi errata.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Profilo

Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.VISUALIZZA\_PROFILO** e **payload** così costruito:

1. ID\_Utente : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: l’id fornito è coretto. In questo caso il **payload** conterrà:
  + Se l’utente visualizza il suo **ID\_UTENTE**:
    1. nome : String
    2. cognome : String
    3. Password : String
    4. dataNascita : Date
    5. cittaResidenza : String
    6. cap : Integer
    7. indirizzo : String
    8. email : String
    9. iban : String
  + Se un utente visualizza l’**ID\_UTENTE** di un altro:
    1. nome : String
    2. cognome : String
    3. email : String
* **TipoRisposta.ERRORE**: la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se l’**ID\_UTENTE** non viene trovato.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Modifica Profilo

Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.MODIFICA\_PROFILO** e **payload** così costruito:

1. IdUtente : Integer
2. nome : String
3. cognome : String
4. dataNascita : Date
5. cittaResidenza : String
6. cap : Integer
7. indirizzo : String
8. email : String
9. saldo : Float
10. iban : String

Gli oggetti con valore null, sono da considerarsi da non modificare.

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: l’id fornito è corretto. In questo caso il **payload** sarà vuoto.
* **TipoRisposta.ERRORE**: la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se l’**ID\_UTENTE** non corrisponde con quello dell'utente.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Lista Aste / Vendite / Aste Salvate

La richiesta è uguale per tutte e tre le tipologie.

* Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.VISUALIZZA\_LISTA\_ASTE** e **payload** così costruito:
  1. numeroAste : Integer
  2. numeroPagina : Integer
  3. idCategorie : Int[]

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**:
  + Il server manda indietro nel **payload**:
    1. aste : Object[], un array di oggetti in cui si ripetono per ogni asta:
       1. idAsta : Integer
       2. durata : LocalDate
       3. prezzoAttuale : Float
       4. nomeLotto : String
* **TipoRisposta.ERRORE**: la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.ERRORE\_CARICAMENTO**: se il server non riesce a trovare le aste.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Asta

* Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.VISUALIZZA\_ASTA** e **payload** così costruito:
  1. IdAsta : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**:
  + Il server manda indietro nel **payload**:
    - 1. idAsta : Integer
      2. durata : LocalDate
      3. prezzoAttuale : Float
      4. nomeLotto : String
      5. articoli : Object[] :
         1. nomeArticolo : String
* **TipoRisposta.ERRORE**: la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.ERRORE\_CARICAMENTO**: se il server non riesce a trovare le aste.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Salva Asta

* Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.SALVA\_ASTA** e **payload** così costruito:
  + - 1. durata : LocalDate
      2. prezzoAttuale : Float
      3. nomeLotto : String
      4. IdLotto : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**:
  + Il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.ERRORE\_CARICAMENTO**: se il server non riesce a salvare le aste.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Modifica Asta

* Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.MODIFICA\_ASTA** e **payload** così costruito:
  + - 1. idAsta : Integer
      2. durata : LocalDate
      3. prezzoAttuale : Float
      4. nomeLotto : String
      5. IdLotto : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**:
  + Il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.ERRORE\_CARICAMENTO**: se il server non riesce a modificrare le aste.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Annulla Asta

* Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.ANNULLA\_ASTA** e **payload** così costruito:
  + - 1. idAsta : Integer
      2. durata : LocalDate
      3. prezzoAttuale : Float
      4. nomeLotto : String
      5. IdLotto : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**:
  + Il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.ERRORE\_CARICAMENTO**: se il server non riesce ad annullare le aste.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Crea Asta

* Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.CREA\_ASTA** e **payload** così costruito:
  + - 1. idAsta : Integer
      2. durata : LocalDate
      3. prezzoAttuale : Float
      4. nomeLotto : String
      5. IdLotto : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**:
  + Il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.ERRORE\_CARICAMENTO**: se il server non riesce a creare le aste.
  + **TipoErrore.GENERICO**: per un errore del server.

## Database



Figura 3 - Diagramma E/R ristrutturato

# Documenti di riferimento

|  |  |  |
| --- | --- | --- |
| Titolo | Nome file | Descrizione |
| Indicazioni committente | [Testo commitente.pdf](Testo%20committente.pdf) | Documento contenente le indicazioni date dal committente. |
| Coding rules | [Coding rules.docx](Coding%20rules.docx) | Documento contente le regole di creazione e stesura del codice e dei file nel repository. |
| Progettazione database | [DB Aste.docx](DB%20Aste.docx) | Documento contenente la progettazione del database, sia nella parte concettuale che logica. |
| Diagramma ER database | <Schema_ER.drawio> | Diagramma Entity-Relationship del database. |
| Use Cases utente | <UC-Utente.drawio> | Diagramma degli Use Cases per l’utente. |
| Use Cases admin | <UC-Admin.drawio> | Diagramma degli Use Cases per l’admin. |
| Diagramma dei package | [Diagramma Package.drawio](Diagramma%20Package.drawio) | Diagramma dei package dell’applicazione. |
| Diagramma delle classi package aste | <DC-Aste.drawio> | Diagramma delle classi per il package aste. |
| Diagramma delle classi package client | [DC- Client.drawio](DC-%20Client.drawio) | Diagramma delle classi per il package aste.client. |
| Diagramma delle classi package server | <DC-Server.drawio> | Diagramma delle classi per il package aste.server. |
| SQL database | <TabelleDB.sql> | Codice SQL per la creazione del database. |