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;
    - le regole di comunicazione tra client e server
  + 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, Enzo Calissi | 02/04/2024 | 0 | Prima stesura del documento |

# Analisi

## Utente



## Admin



# Progettazione

## Software

### Diagramma dei package



Figura 1 - Diagramma dei package

### Diagrammi delle classi



Figura 2 - Diagramma delle classi package aste

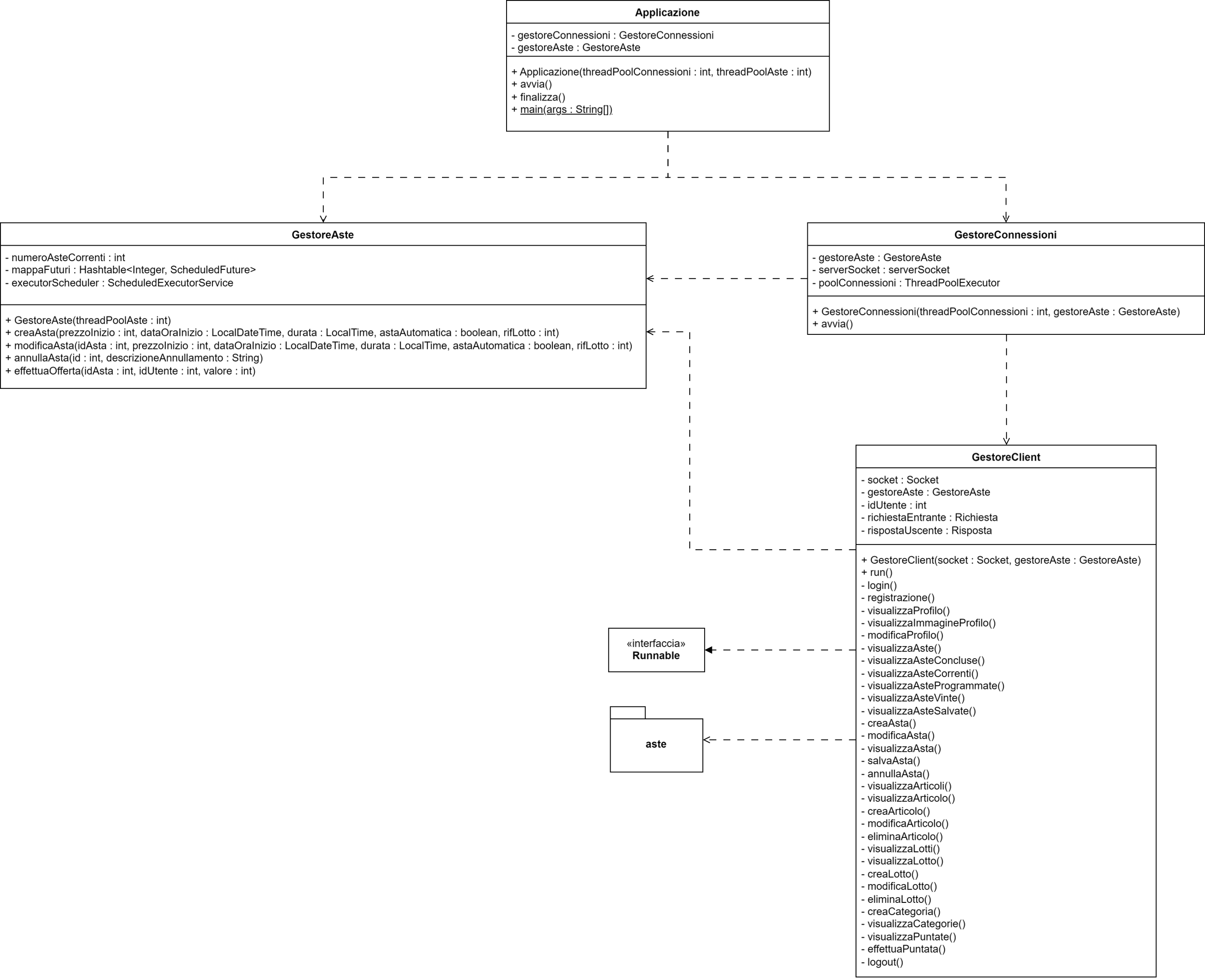


Figura 3 - Diagramma delle classi package server

### Diagrammi di sequenza



Figura 4 - Diagramma di sequenza per Login



Figura 5 - Diagramma di sequenza per Modifica Profilo

### Comunicazione client-server

#### 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** contiente:
  + idUtente : Integer
  + admin : Boolean
* **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 : LocalDate
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 uno o più campi hanno 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. idUtente : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: l’id fornito è coretto. In questo caso il **payload** conterrà:
  + Se **idUtente** è null i dati del suo profilo:

1. nome : String
2. cognome : String
3. dataNascita : LocalDate
4. cittaResidenza : String
5. cap : Integer
6. indirizzo : String
7. email : String
8. iban : String
   * altrimenti il profilo dell’**idUtente** di un altro utente:
9. nome : String
10. cognome : String
11. email : String

* **TipoRisposta.ERRORE**: se la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se l’**id\_Utente** fornito è malformato.
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se l’**Id\_Utente** non viene trovato.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Immagine Profilo

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

1. idUtente : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: l’id fornito è coretto. In questo caso il **payload** conterrà:
  + immagineProfilo : byte[]
* **TipoRisposta.ERRORE**: se la registrazione viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se l’**idUtente** è invalido.
  + **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. nome : String
2. cognome : String
3. dataNascita : LocalDate
4. cittaResidenza : String
5. cap : Integer
6. indirizzo : String
7. email : String
8. iban : String
9. immagineProfilo : byte[]

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

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: la modifica è andata a buon fine. In questo caso il **payload** sarà vuoto.
* **TipoRisposta.ERRORE**: la modifica viene rifiutata e il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono malformati.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Aste / Concluse / Correnti / Programmate / Vinte / Salvate

Il client manda al server un oggetto Richiesta con **tipoRichiesta** tra uno di questi:

* **TipoRichiesta.VISUALIZZA\_ASTE**
* **TipoRichiesta.VISUALIZZA\_ASTE\_CONCLUSE**
* **TipoRichiesta.VISUALIZZA\_ASTE\_CORRENTI**
* **TipoRichiesta.VISUALIZZA\_ASTE\_PROGRAMMATE**
* **TipoRichiesta.VISUALIZZA\_ASTE\_VINTE**
* **TipoRichiesta.VISUALIZZA\_ASTE\_SALVATE**

e **payload** così costruito:

1. numeroAste : Integer
2. numeroPagina : Integer
3. stringaRicerca : String
4. 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:
2. idAsta : Integer
3. durata : LocalTime
4. prezzoAttuale : Float
5. nomeLotto : String
6. immaginePrincipale : byte[]

Se non viene trovata nessuna asta l’array sarà vuoto.

* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se il numero di pagina è invalido per il numero di aste per pagina fornito, uno o più id delle categorie sono invalidi o uno o più campi sono malformati.
  + **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. dataOraInizio : LocalDateTime
2. durata : LocalTime
3. prezzoInizio : Float
4. astaAutomatica : Boolean
5. idLotto : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**:
  + Il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se si prova a mettere all’asta un lotto già venduto.
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono malformati.
  + **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. dataOraInizio : LocalDateTime
3. durata : LocalTime
4. prezzoInizio : Float
5. astaAutomatica : Boolean
6. idLotto : Integer

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

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se si sta provando a modificare un’asta già iniziata.
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono invalidi.
  + **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. dataOraInizio : LocalDateTime
2. durata : LocalTime
3. prezzoInizio : Float
4. prezzoAttuale : Float
5. ipMulticast : InetAddress
6. descrizioneAnnullamento : String
7. idLotto : Integer
8. nomeLotto : String
9. immaginiArticoli : byte[][]

Se l’asta non è ancora iniziata ipMulticast sarà null. Se l’asta non è stata annullata descrizioneAnnullamento sarà null.

* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se l’idAsta è invalido.
  + **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. idAsta : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se l’idAsta è invalido.
  + **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. descrizioneAnnullamento : String

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se l’idAsta appartiene ad un’asta che non è dell’utente, l’utente non è un admin, o l’idAsta è di un’asta già conclusa.
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono malformati.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Articoli

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

1. numeroArticoli : Integer
2. numeroPagina : Integer
3. stringaRicerca : String
4. idCategorie : int[]

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro nel **payload**:

1. articoli : Object[], un array di oggetti in cui si ripetono per ogni articolo:
2. idArticolo : Integer
3. nome : String
4. condizione : String
5. immaginePrincipale : byte[]

* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se il numero di pagina è invalido per il numero di articoli per pagina fornito, uno o più id delle categorie sono invalidi o uno o più campi sono malformati.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Articolo

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

1. idArticolo : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro nel **payload**:

1. nome : String
2. condizione : String
3. descrizione : String
4. idLotto : Integer
5. immaginiArticolo: byte[][]
6. idUtente : Integer
7. emailUtente : String

* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.ERRORE\_CARICAMENTO**: se il server non riesce a trovare gli articoli.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Crea Articolo

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

1. nome : String
2. condizione : String
3. descrizione : String
4. idLotto : Integer
5. immaginiArticolo: byte[][]
6. immaginePrincipale : Integer
7. categorie : int[]

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono malformati.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Modifica Articolo

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

1. idArticolo : Integer
2. descrizione : String
3. nome : String
4. condizione : String
5. idLotto : Integer
6. immaginiArticolo: byte[][]
7. immaginePrincipale : Integer
8. categorie : int[]

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

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se si sta provando a modificare un articolo attualmente all’asta, di un’asta vinta, o non proprio.
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono invalidi.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Elimina Articolo

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

1. idArticolo : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se si sta provando ad eliminare un articolo appartenente ad un lotto, e inoltre il **payload** contiene:
    1. idLotto : Integer
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono invalidi.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Lotti

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

1. numeroLotti : Integer
2. numeroPagina : Integer
3. stringaRicerca : Integer
4. idCategorie : int[]

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro nel **payload**:

1. lotti : Object[], un array di oggetti in cui si ripetono per ogni lotto:
2. idLotto : Integer
3. nome : String
4. immaginePrincipale : byte[]

* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.ERRORE\_CARICAMENTO**: se il server non riesce a trovare i lotti.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Lotto

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

1. idLotto : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro nel **payload**:

1. idLotto: Integer
2. nome: String
3. immaginiArticoli : byte[][]
4. articoli : Object[], che per ogni articolo contiene:
5. idArticolo : Integer
6. nomeArticolo : String

* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se il numero di pagina è invalido per il numero di articoli per pagina fornito, uno o più id delle categorie sono invalidi o uno o più campi sono malformati.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Crea Lotto

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

1. nomeLotto : String

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono malformati.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Modifica Lotto

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

1. idLotto : Integer
2. nome : String

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se si sta provando a modificare un articolo attualmente all’asta o di un’asta vinta.
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono invalidi.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Elimina Lotto

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

1. idLotto: Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se si sta provando ad eliminare un lotto già venduto in un asta, e inoltre il **payload** contiene:
    1. idAsta : Integer
  + **TipoErrore.CAMPI\_INVALIDI**: se uno o più campi sono invalidi.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Crea Categoria

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

1. nome : String

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** che contiene all’indice 0 l’idCategoria della categoria appena creata.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se il nome data è null.
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se il nome dato non è univoco tra tutte le categorie.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Categorie

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

1. stringaRicerca : String

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** che contiene per ogni categoria:

1. idCategoria : Integer
2. nome : String
3. articoliAssociati : Integer

* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.CAMPI\_INVALIDI**: se la stringaRicerca data è null.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Visualizza Puntate

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

1. IdAsta : Integer

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** che contiene per ogni offerta:

1. idPuntata : Integer
2. valore : Float
3. dataOraEffettuazione : LocalDateTime
4. idUtente : Integer
5. nomeUtente : String

* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se l’id asta non appartiene ad un’asta esistente.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Effettua Puntata

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

1. IdAsta : Integer
2. valore : Float

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se il server riceve una puntata con valore più basso di quello più alto attuale, per un’asta non in corso, o per un’asta dell’utente stesso.
  + **TipoErrore.GENERICO**: per un errore del server.

#### Logout

Il client manda al server un oggetto Richiesta con **tipoRichiesta** = **TipoRichiesta.LOGOUT** e **payload** vuoto.

Se il server risponde con **tipoRisposta**:

* **TipoRisposta.OK**: il server manda indietro il **payload** vuoto.
* **TipoRisposta.ERRORE**: il **payload** contiene nell’indice 0:
  + **TipoErrore.OPERAZIONE\_INVALIDA**: se non è stato effettuato un login preventivamente.
  + **TipoErrore.GENERICO**: per un errore del server.

## Database



Figura 6 - 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. |