|  |
| --- |
| ***Relazione progetto “Aste Online”*** |
| *By Paini Mirko, Barsi Matteo Luigi,*  *Del Checcolo Matteo & Gori Lorenzo* |
|  |
|  |
|  |
|  |

***INDICE***

1. ***Scopo del progetto***
2. ***Ruoli del progetto***
3. ***Funzionamento del progetto***
4. ***Diagramma delle classi***
5. ***Diagramma di Sequenza***
6. ***Database***
   1. ***Entità***
   2. ***Analisi delle Relazioni***
   3. ***Schema ER***
   4. ***Schema Logico***
   5. ***SQL***
7. ***Strumenti***

***Scopo del progetto***

*Lo scopo del nostro progetto è quello di realizzare un software per gestire l'acquisto e la vendita di oggetti tramite aste online.*

*Quest’ultimo permetterà agli utenti di mettere in vendita i propri oggetti, partecipare alle aste per acquistarne di nuovi e gestire le transazioni in modo rapido e sicuro.*

***Ruoli del progetto***

* *Mirko Paini: Diagrammi di Sequenza , Database, Relazione;*
* *Lorenzo Gori: Diagramma delle Classi, Back-end;*
* *Luigi Matteo Barsi: Database, Back-end, Front-end;*
* *Matteo Del Checcolo: Diagrammi di Sequenza, Relazione.*

***Funzionamento del progetto***

*Il nostro progetto si basa sulla realizzazione di un software di gestione di aste online, che fa uso di Socket TCP e di Multicast. Questo permette agli utenti di partecipare alle aste dei vari lotti e formulare delle offerte e, eventualmente, di aggiudicarsi i relativi oggetti.*

*Il nostro progetto è strutturato in 9 classi, ciascuna svolgendo un ruolo specifico:*

* *La classe "Oggetto" rappresenta un oggetto appartenente ad’un lotto, ha come attributi una categoria, un nome ed’ un ID*
* *La classe "Lotto” rappresenta un insieme di oggetti a cui il client, successivamente, si potrà connettere per verificare lo stato dell’asta e il suo contenuto.*
* *La classe "Asta” rappresenta il nostro “contenitore” che va a raccogliere l’insieme di tutti i lotti presenti.*

* *La classe "GestoreAste” ha due funzionalità:*

1. *Permette agli admin di inserire, chiudere e visualizzare le aste;*
2. *Permette agli utenti di inserire nuovi lotti, effettuare dei rilanci e visualizzare la lista delle aste.*

* *La classe "Utente” rappresenta un utente (con Nome, Cognome, E-mail, Password e Numero di telefono) oppure un admin che può o meno essere connesso al server.*
* *La classe "Client” gestisce l’invio di richieste da parte sia degli utenti che degli admin al server.*

*Tra le richieste troviamo:*

*Quelle che possono effettuare sia i client che gli admin:*

* *Registrazione;*
* *Login;*
* *Logout;*
* *Richiesta lista aste;*

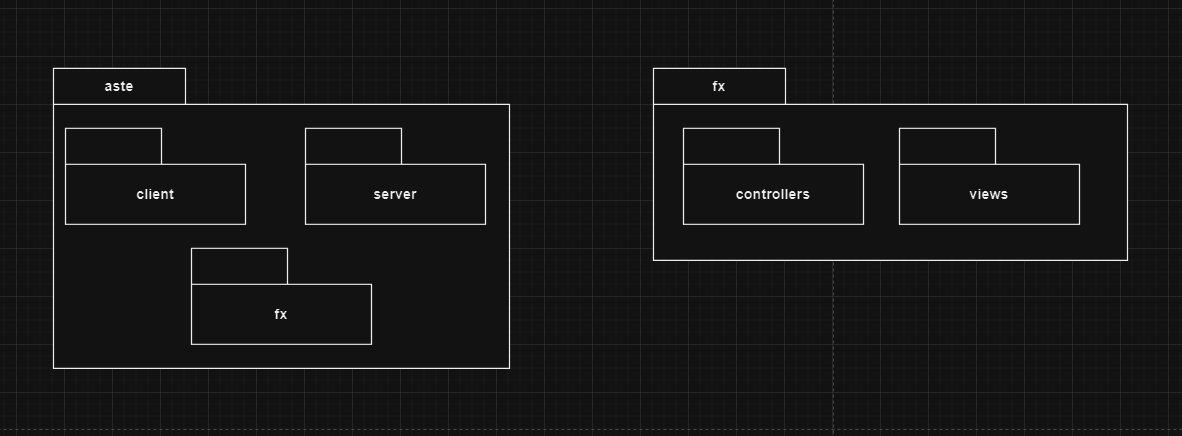
*Quelle che possono effettuare solo gli admin:*

* *Creazione aste;*
* *Chiusura aste;*

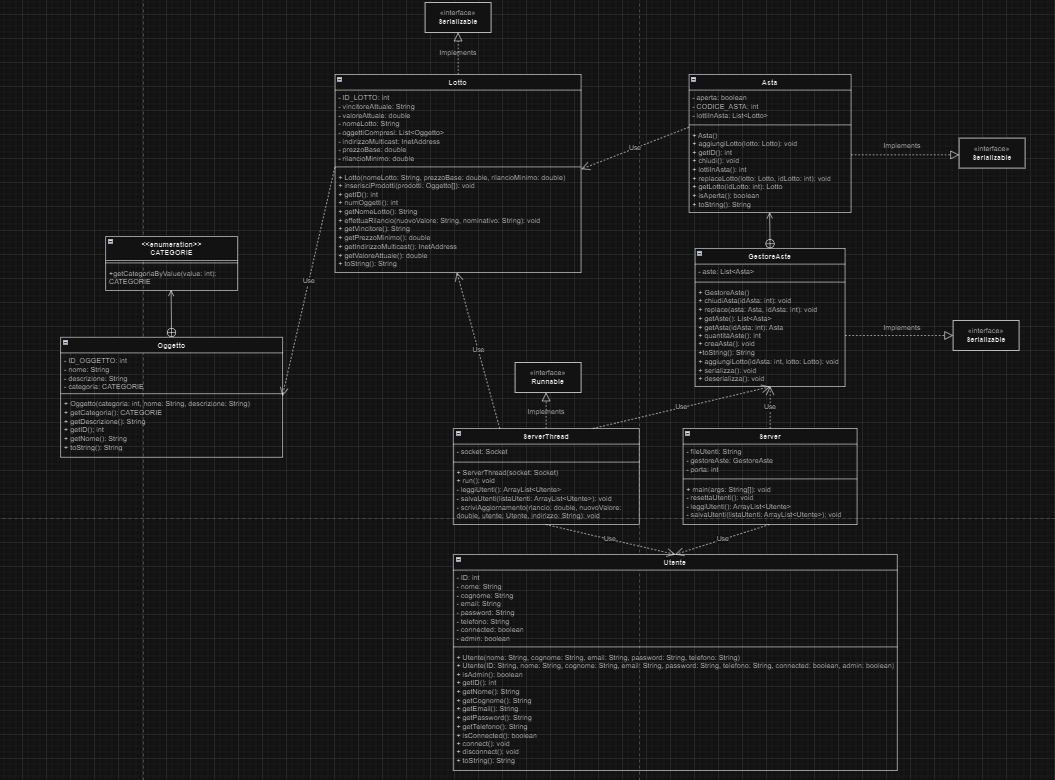
*Quelle che possono effettuare solo gli utenti:*

* *Inserimento lotto;*
* *Entrare in un gruppo (lotto);*
* *Uscire da un gruppo (lotto);*
* *Effettuare un rilancio.*
* *La classe "Server” si occupa di verificare e nel caso accettare le richieste di connessione da parte dei client.*
* *La classe “ServerThread” ha il compito di fare da tramite tra il client e il server, gestendo lo scambio di messaggi tra le due entità.*
* *La classe “ThreadMulticast” permette al client di entrare nei vari gruppi Multicast (Lotti ).*

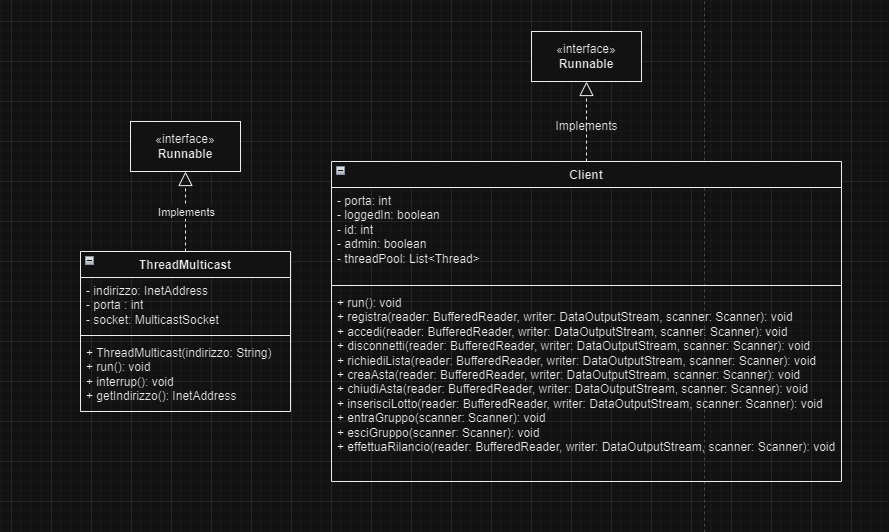
***Diagramma dei package***



***Diagrammi delle classi***

-Package server  


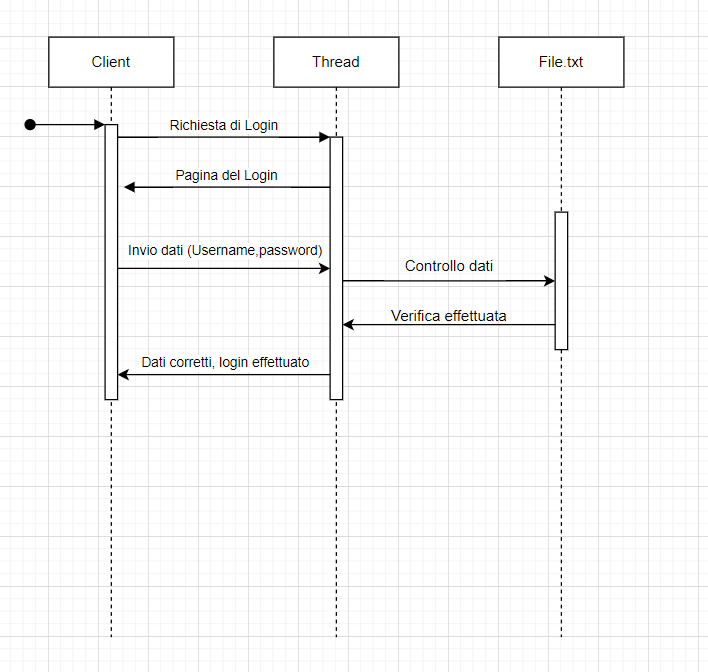
-Package client



***Diagramma di Sequenza***

***Richiesta di Login***

* *Il primo diagramma rappresenta la sequenza dei messaggi tra un client e un thread nel momento in cui andiamo a effettuare il login:*



*Client invia richiesta Login*

*Server invia pagina di Login*

*Client invia email, password*

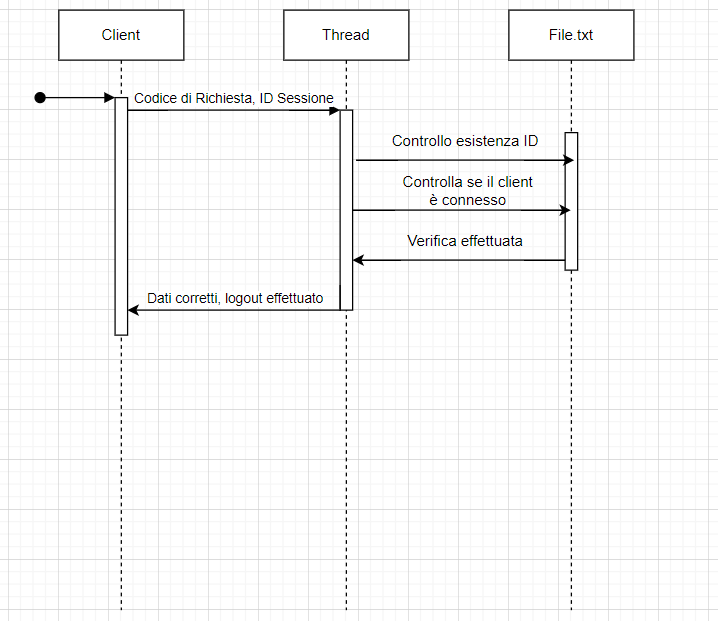
*Server controlla i dati, invia ([OK] + id di sessione) se i dati sono corretti,*

*Invia ([ER] + motivo dell’errore) altrimenti*

*Client salva id di sessione se ricevuto, stampa l’errore altrimenti*

***Richiesta di Logout***

* *Il secondo diagramma rappresenta la sequenza dei messaggi tra un client e un thread nel momento in cui andiamo a effettuare il logout:*



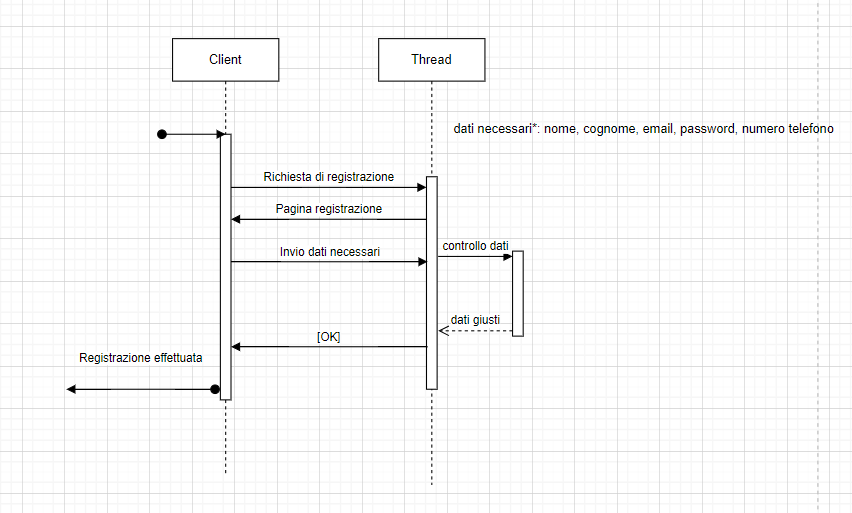
***Client invia richiesta di logout + id di sessione***

***Server controlla che il client con l’id sia connesso, invia [OK] se connesso***

***invia ([ER] + motivo dell’errore) se altrimenti***

***Client disconnette se ricevuto [OK], altrimenti stampa motivo dell’errore***

***Richiesta di Signin***



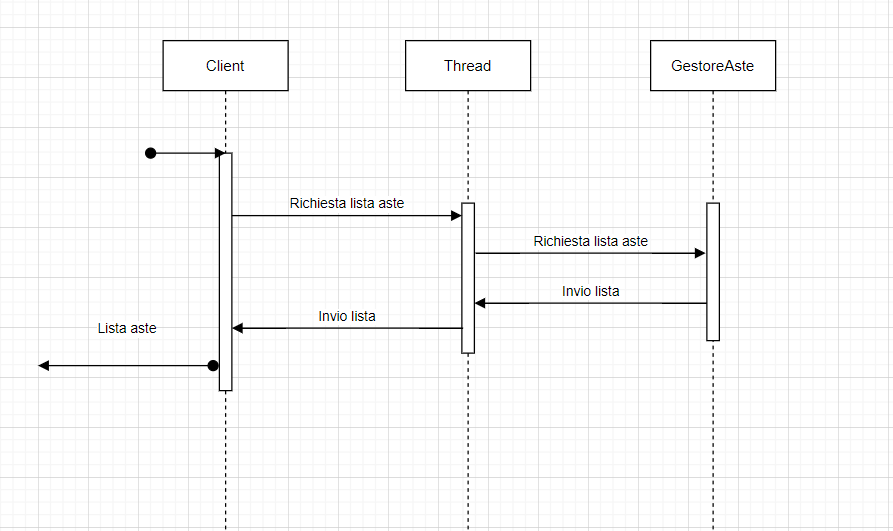
***Client invia richiesta di registrazione***

***Server invia pagina di registrazione***

***Client invia nome, cognome, email, password, numero di telefono***

***Server controlla i dati, invia [OK] se i dati sono corretti ed’effettua registrazione dell’utente***  
  ***invia ([ER] + motivo dell’errore) se utente già esistente o dati errati***

***Client visualizza “Registrazione avvenuta” se ricevuto [OK], motivo dell’errore altrimenti***

***Richiesta lista aste***  


***Client invia richiesta lista aste, id di sessione***

***Server richiede lista aste al gestore***

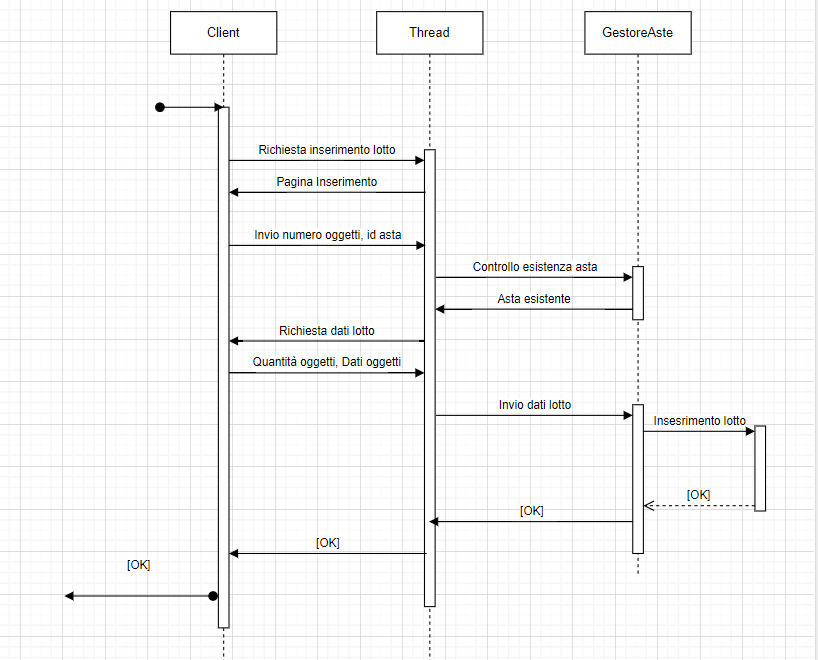
***GestoreAste invia lista aste***

***Server invia [OK] + lista aste se l’utente risulta connesso e autorizzato,***

***Invia ([ER] + motivo dell’errore) altrimenti***

***Client visualizza lista aste se ricevuto [OK], motivo dell’errore altrimenti***

***Richiesta inserimento lotto***



***Client invia richiesta di inserimento lotto + id di sessione***

***Server controlla che l’utente sia connesso e autorizzato ed invia ([OK] + pagina inserimento***  ***lotto),***

***invia ([ER] + motivo dell’errore) altrimenti***

***Client invia quantità oggetti del lotto, id asta in cui inserirlo, se ricevuto [OK]***

***Client visualizza motivo dell’errore altrimenti***

***Server controlla che l’asta indicata esista, invia [OK] se l’asta esiste***

***invia ([ER] + motivo dell’errore) altrimenti***

***Client invia Dati oggetti (categoria, nome, descrizione) se ricevuto [OK],***

***Client visualizza motivo dell’errore altrimenti***

***Server invia risposta [OK]***

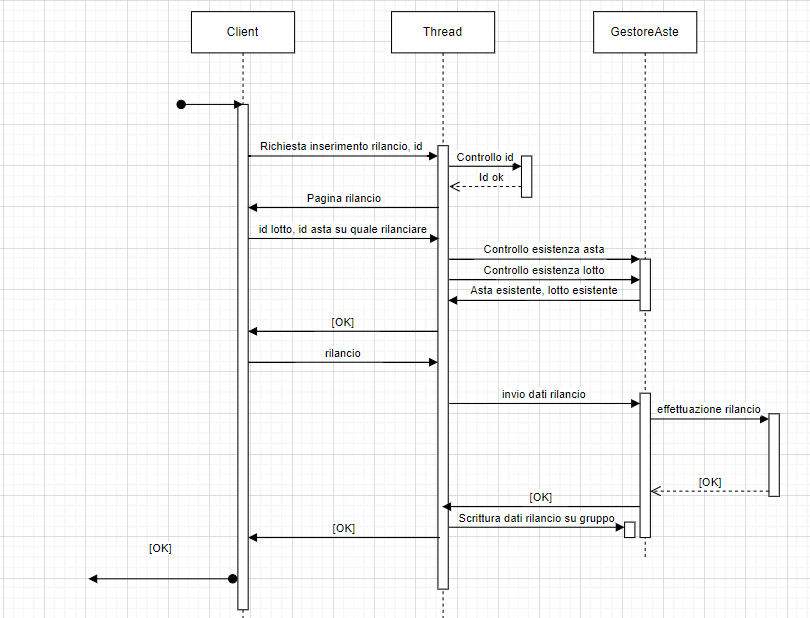
***Client invia Dati lotto (nome lotto, prezzo base, rilancio minimo)***

***Server inserisce lotto nell’asta, invia [OK] se inserito con successo***

***invia ([ER] + motivo dell’errore) altrimenti***

***Client visualizza “Lotto aggiunto con successo” se ricevuto [OK]***

***Client visualizza motivo dell’errore altrimenti***

***Richiesta invio rilancio***

***Client invia richiesta di rilancio al server, insieme al suo id di sessione***  
***Server controlla che l’utente sia connesso e invia ([OK] + pagina rilancio)***  
 ***(invia [ER] + motivo dell’errore) altrimenti***

***Client invia id lotto su quale effettuare il rilancio e id asta su cui si trova il lotto,***  
 ***se è stato ricevuto [OK]***  
 ***visualizza motivo dell’errore altrimenti***  
***Server controlla che lotto e asta esistano ed’ invia [OK],***   
 ***(invia [ER] + motivo dell’errore) altrimenti***

***Client invia il valore del rilancio se ricevuto [OK]***  
 ***visualizza motivo dell’errore altrimenti***

***Server effettua il rilancio sull’asta, scrive il dato sul gruppo Multicast del lotto ed’ invia [OK]***  
 ***invia ([ER] + motivo dell’errore) se non è stato possibile effettuare il rilancio***

***Client visualizza messaggio “Rilancio effettuato con successo”, se ricevuto [OK]***  
 ***visualizza motivo dell’errore altrimenti***

***Database***

***Entità***

* ***Utente:***
* *ID\_Utente  int*
* *Nome string*
* *Cognome string*
* *Email string*
* *Password string*
* *Telefono int*
* ***Articoli:***
* *ID\_Articoli int*
* *Nome string*
* *Descrizione string*
* ***Categoria:***
  + *ID\_Categoria int*
  + *Nome string*
* ***Lotto:***
  + *ID\_Lotto int*
  + *Nome string*
  + *Ip\_multicast string*
  + *Valore\_Iniziale float*
* ***Asta:***
  + *ID\_Asta int*
  + *Data\_ora\_inizio date*
  + *Data\_ora\_fine date*
  + *Durata int*
  + *Terminata  boolean*
* ***Puntata:***
  + *ID\_Puntata int*
  + *Valore int*
  + *Data\_ora\_effettuazione date*

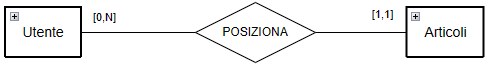
***Analisi delle relazioni***

1. *UTENTE->ARTICOLI*

*Un utente può posizionare più articoli all’asta.*

*ARTICOLI->UTENTE*

*un articolo deve essere posizionato da un solo utente.*

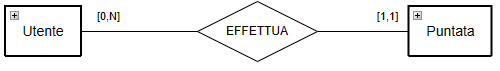


1. *UTENTE->PUNTATA*

*Un utente può effettuare zero o più puntate.*

*PUNTATA->UTENTE*

*Una puntata deve essere effettuata da un solo utente.*



1. *PUNTATA->ASTA*

*Una puntata deve riguardare un’asta.*

*ASTA->PUNTATA*

*Ad un’asta possono riguardare nessuna o più puntata.*

https://lh7-us.googleusercontent.com/k7HZhbG2ZAMbU7gaLJaFUsTTelVAsEs41kk1s7pbHxl2Tjcfp-g2x_nsYZh3_QelmWjsTl-VTS18V3bjFyjKcl_Dyjt8cuGuJ36r8t78ikUlsS4ILXWp6gXtBt0SwyYLiUP8VvbU2R3zpheI-rQgKBI

1. *ARTICOLI->CATEGORIA*

*Un articolo deve rientrare in una categoria.*

*CATEGORIA->ARTICOLI*

*In una categoria rientrano nessuno o più articoli.*

https://lh7-us.googleusercontent.com/hgDajYj6lizIB_xk7dT6-jASMXejDfexgmDHoQpIkacSKe20iCfl0_8ZFkt3KhJp1L0uP1bhDHagKk2WkBgCo9ANWv2mx3jTQycpC0TsFGjUwO4a3vvpftG4o6zQBNbtuijgwsJ-F8dhx9NwR-mfJJM

1. *ASTA->LOTTO*

*Un’asta deve essere riferita ad un lotto.*

*LOTTO->ASTA*

*Ad ‘un lotto possono essere riferite più aste.*

https://lh7-us.googleusercontent.com/LzKNrXt4ldwKHGJPjxmVE6KUw8F4-KNnQBzSbjsvQwX2tUGOs-LQn8E3YQTsXw6Qe2dJ7UelZsx8E4-aygXEcm3BHBcwGn68r1BfkfuC1acgaxOL3fD2JlTq2i1ZJoXlZN2bk_z5ic0W5qmiDsDm1LA

1. *ARTICOLI->LOTTO*

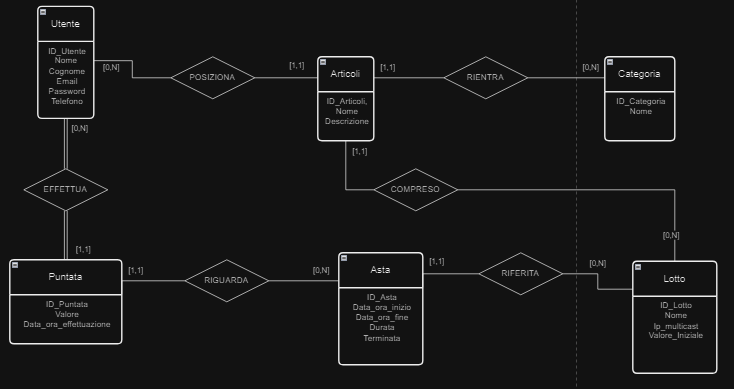
*Un articolo deve essere compreso in un lotto.*

*LOTTO->ARTICOLI*

*Un lotto può comprendere nessuno o più articoli.*

https://lh7-us.googleusercontent.com/ewkrh8h9S9Qor4uau3Om3XEHXbO27kDlYwJFFwOJHCiSDYoDAGoXasr1MrJgQJQ9lshXYk4VU7FY_HJsKTXA-mb803nip1HdnboV8lLPEPflazm9ID3ZeVLRU7ZnoV08vdlMs8d2_VO7k2lQxX2hWdk

***Schema ER***



***Schema logico***

* *ROSSO = chiave primaria.*
* *BLU = chiave esterna.*

*Utente (ID\_Utente, Nome, Cognome, Email, Password, Telefono)*

*Articoli( ID\_Articoli, Nome, Descrizione, RIF\_Lotto, RIF\_Utente )*

*Categoria( ID\_Categoria, Nome );*

*ArticoliCategoria( RIF\_Articoli, RIF\_Categoria, PRIMARY KEY (RIF\_Articoli,RIF\_Categoria))*

*Asta( ID\_Asta, Data\_ora\_inizio, Data\_ora\_fine, Durata, Terminata, RIF\_Lotto)*

*Lotto( ID\_Lotto, Nome, Ip\_multicast , Valore\_Iniziale )*

*Puntata( ID\_Puntata, Valore, Data\_ora\_effettuazione, RIF\_Utente, RIF\_Lotto)*

***SQL***

***Esempio di Creazione tabella SQL:***

*CREATE TABLE Utente(*

*ID\_Utente int NOT NULL AUTO\_INCREMENT,*

*Nome char(50),*

*Cognome char(50),*

*Email char(128),*

*Password char(128),*

*Telefono char(13),*

*PRIMARY KEY (ID\_Utente)*

*);*

*CREATE TABLE Articoli(*

*ID\_Articoli int NOT NULL AUTO\_INCREMENT,*

*Nome char(50),*

*Descrizione char(255),*

*RIF\_Lotto int NOT NULL,*

*RIF\_Utente int NOT NULL,*

*PRIMARY KEY (ID\_Articoli),*

*FOREIGN KEY (RIF\_Lotto) REFERENCES Lotto(ID\_Lotto),*

*FOREIGN KEY (RIF\_Utente) REFERENCES Utente(ID\_Utente)*

*);*

*CREATE TABLE ArticoliCategoria(*

*RIF\_Articoli int NOT NULL,*

*RIF\_Categoria int NOT NULL,*

*CONSTRAINT PK\_ArticoliCategoria PRIMARY KEY (RIF\_Articoli,RIF\_Categoria),*

*FOREIGN KEY (RIF\_Lotto) REFERENCES Lotto(ID\_Lotto),*

*FOREIGN KEY (RIF\_Utente) REFERENCES Utente(ID\_Utente)*

*);*

*CREATE TABLE Categoria(*

*ID\_Categoria int NOT NULL AUTO\_INCREMENT,*

*Nome char(50),*

*PRIMARY KEY (ID\_Categoria)*

*);*

*CREATE TABLE Lotto(*

*ID\_Lotto int NOT NULL AUTO\_INCREMENT,*

*Nome char(50),*

*Ip\_multicast char(15),*

*Valore\_Iniziale int,*

*PRIMARY KEY (ID\_Lotto)*

*);*

*CREATE TABLE Asta(*

*ID\_Asta int NOT NULL AUTO\_INCREMENT,*

*Data\_ora\_inizio date,*

*Data\_ora\_fine date,*

*Durata int,*

*Terminata boolean,*

*RIF\_Lotto int NOT NULL,*

*PRIMARY KEY (ID\_Asta),*

*FOREIGN KEY (RIF\_Lotto) REFERENCES Lotto(ID\_Lotto)*

*);*

*CREATE TABLE Puntata(*

*ID\_Puntata int NOT NULL AUTO\_INCREMENT,*

*Valore float,*

*Data\_ora\_effettuazione date,*

*RIF\_Utente int NOT NULL,*

*RIF\_Lotto int NOT NULL,*

*PRIMARY KEY (ID\_Puntata),*

*FOREIGN KEY (RIF\_Lotto) REFERENCES Lotto(ID\_Lotto),*

*FOREIGN KEY (RIF\_Utente) REFERENCES Utente(ID\_Utente)*

*);*

***Strumenti***

* ***Database:*** *draw.io,  Notepad++,  phpMyAdmin;*
* ***Back-end:*** *Visual Studio Code,  Stack Overflow;*
* ***Front-end:*** *Scene Builder;*
* ***Diagrammi:*** *draw.io.*