



**POLITECNICO**  
MILANO 1863

# Tecnologie Informatiche per il Web

**Aste Online** – Documentazione e presentazione progetto

Studente: Jacopo Frasson – Sessione: Settembre 2021

Applicazione per gestire semplici aste online, realizzata in due versioni implementate all'interno di un unico package:

- una versione HTML pura, lato client funzionante interamente tramite HTML e senza scripting
- una versione RIA realizzata con Javascript, lato client orchestrato da uno script e realizzato su una singola pagina con refresh minimi
- Le due versioni condividono il sistema di sessioni e login, il server e il database.
- Al momento del login, l'utente può scegliere quale delle due versioni utilizzare.
- È possibile effettuare un logout e poi passare all'altra versione.
- Entrambe le versioni condividono lo stile CSS e sono realizzate con stile responsive per adattarsi alle dimensioni della finestra.

## Versione HTML pura:

- Home page con link alle sottosezioni (sempre disponibili anche nell'header della pagina)
- Pagina VENDO:
  - Lista delle aste aperte dell'utente
  - Lista delle aste concluse dell'utente
  - Form per la creazione di nuova asta
- Pagina ACQUISTO:
  - Funzionalità di ricerca aste
  - Lista delle aste acquistate in precedenza
- Pagina DETTAGLIO ASTA & OFFERTA:
  - Funzionalità gestite dalla stessa pagina
  - Riporta i dati dell'asta, la lista delle offerte e funzionalità contestuali a seconda dell'asta visualizzata, in particolare:
    - Fare offerte su aste aperte da altri utenti
    - Chiudere aste in corso se di competenza dell'utente
    - Consultare dettagli di aste già concluse

Funzionalità aggiuntive nella versione RIA:

- Link alla versione HTML pura in caso di javascript disattivato
- Tracking dell'utente tramite cookie per ricordarsi l'ultima operazione effettuata, per riconoscere nuovi utenti e per tenere traccia delle aste aperte visionate
- Aste visionate in precedenza e ancora aperte in evidenza nella pagina ACQUISTO

Il client è realizzato tramite l'utilizzo di codice Javascript nativo (nessuna libreria Javascript aggiuntiva). Il client è stata testato con il browser Chrome.

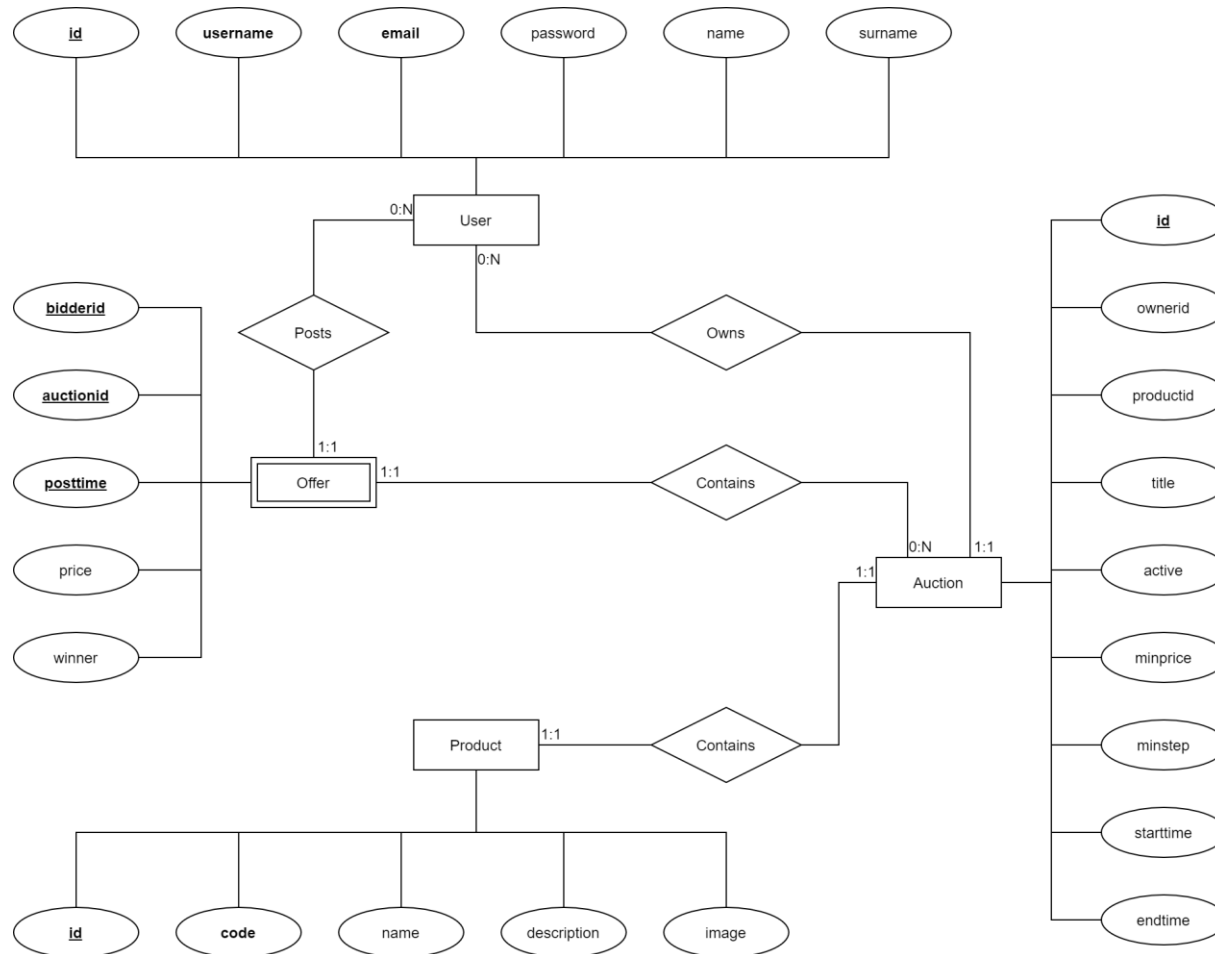
Di seguito sono riportati:

- Il diagramma ER del database alla base dei due applicativi
- Le sequenze DDL utilizzate per la descrizione delle tabelle

Nel diagramma vengono indicate in **grassetto sottolineato** le Primary Keys, e in **grassetto non sottolineato** le Unique Keys.

Nel codice DDL vengono indicate in **rosso** le Primary Keys, in **arancione** le Unique Keys e in **blu** le Foreign Keys.

# Database design – Diagramma ER





# Database design – Tabella Users

```
CREATE TABLE `users` (  
    `id` int NOT NULL AUTO_INCREMENT,  
    `username` varchar(45) NOT NULL,  
    `password` varchar(45) NOT NULL,  
    `name` varchar(45) NOT NULL,  
    `surname` varchar(45) NOT NULL,  
    `email` varchar(45) NOT NULL,  
    PRIMARY KEY (`id`),  
    UNIQUE KEY `username_UNIQUE` (`username`),  
    UNIQUE KEY `email_UNIQUE` (`email`)  
)
```

# Database design – Tabella Auctions

```
CREATE TABLE `auctions` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `ownerid` int NOT NULL,  
  `productid` int NOT NULL,  
  `title` varchar(45) NOT NULL,  
  `active` tinyint NOT NULL,  
  `minprice` decimal(18,2) NOT NULL,  
  `minstep` int NOT NULL,  
  `endtime` datetime NOT NULL,  
  `starttime` datetime NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `ownerid_idx` (`ownerid`), KEY `productid_idx` (`productid`),  
  CONSTRAINT `ownerid` FOREIGN KEY (`ownerid`) REFERENCES `users`  
    (`id`) ON DELETE CASCADE ON UPDATE CASCADE,  
  CONSTRAINT `productid` FOREIGN KEY (`productid`) REFERENCES  
    `products` (`id`) ON DELETE CASCADE ON UPDATE CASCADE  
)
```



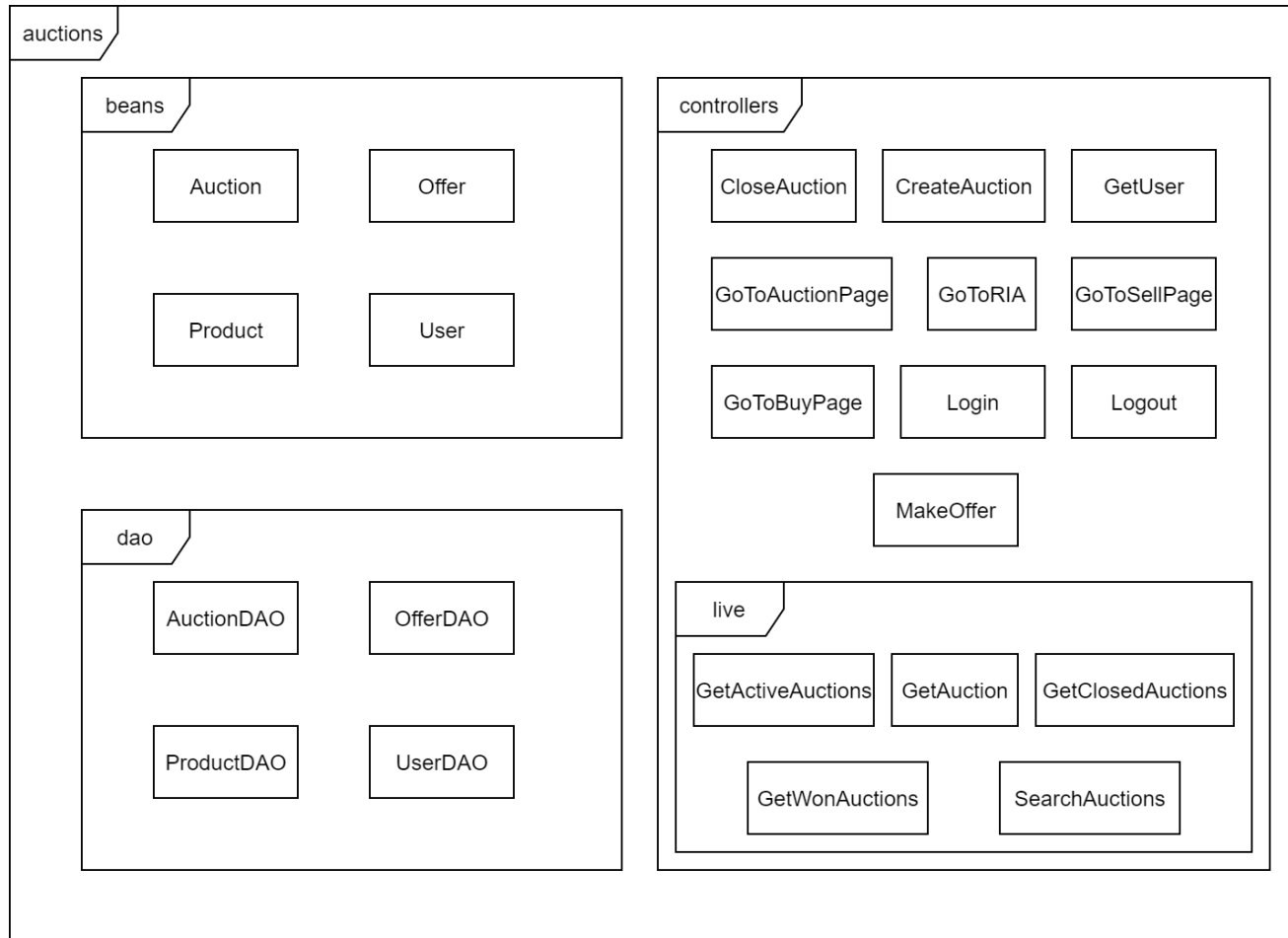
# Database design – Tabella Products

```
CREATE TABLE `products` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `code` varchar(10) NOT NULL,  
  `name` varchar(45) NOT NULL,  
  `description` varchar(1000) DEFAULT NULL,  
  `image` longblob,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `code_UNIQUE` (`code`),  
)
```

# Database design – Tabella Offers

```
CREATE TABLE `offers` (  
  `bidderid` int NOT NULL,  
  `auctionid` int NOT NULL,  
  `winner` tinyint NOT NULL,  
  `posttime` datetime NOT NULL,  
  `price` decimal(18,2) NOT NULL,  
  PRIMARY KEY (`bidderid`, `auctionid`, `posttime`),  
  KEY `auctionid_idx` (`auctionid`),  
  CONSTRAINT `auctionid` FOREIGN KEY (`auctionid`)  
    REFERENCES `auctions` (`id`) ON DELETE CASCADE ON  
    UPDATE CASCADE,  
  CONSTRAINT `bidderid` FOREIGN KEY (`bidderid`)  
    REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE  
    CASCADE  
)
```

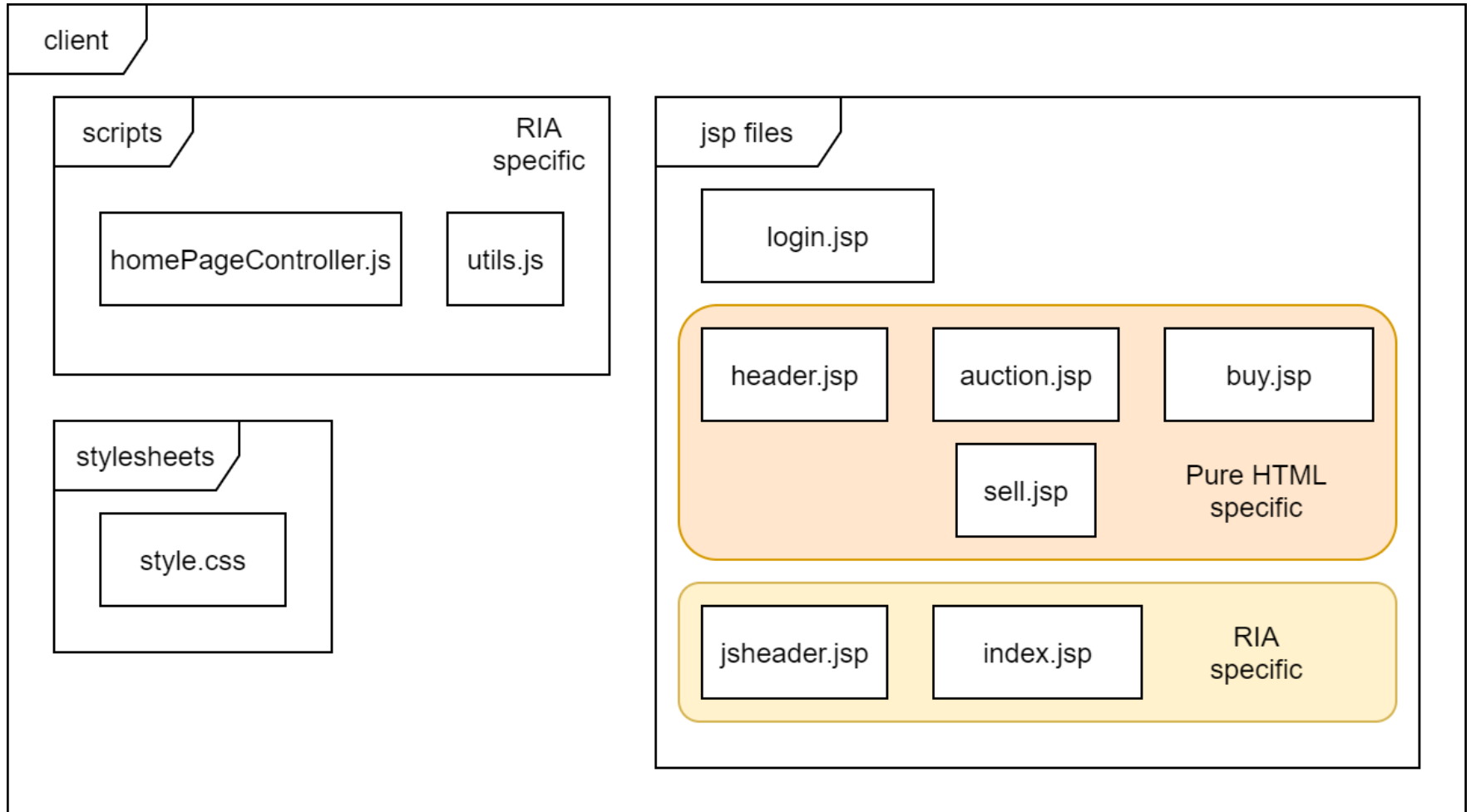
# Modello del server



I controller al di fuori del package live sono specifici del client non-RIA.

Il client HTML e RIA sono attivi e utilizzabili allo stesso tempo.

# Modello del server



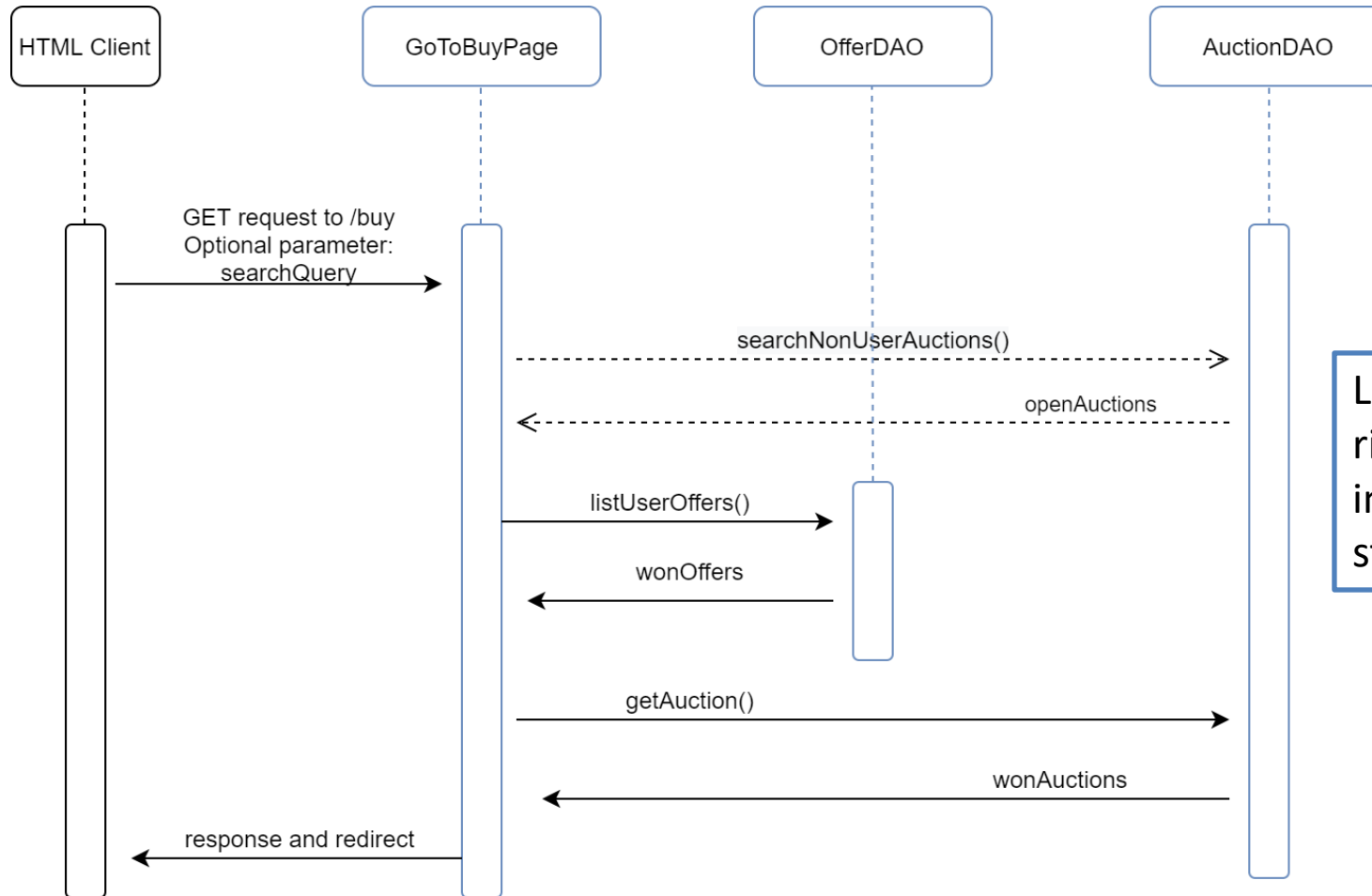
# Diagrammi di flusso degli eventi

Nelle pagine seguenti sono riportati i diagrammi di flusso degli eventi più rilevanti nel funzionamento dei due applicativi.

Il client HTML si avvale in parte di controller dedicati che processano i dati per essere inseriti nel DOM del documento direttamente e inviano risposte contenenti redirect; per questo motivo alcuni grafici fanno riferimento a un solo tipo di client dove specificato. La stessa funzionalità è raggiunta tramite chiamate multiple a endpoint mirati nel client RIA.

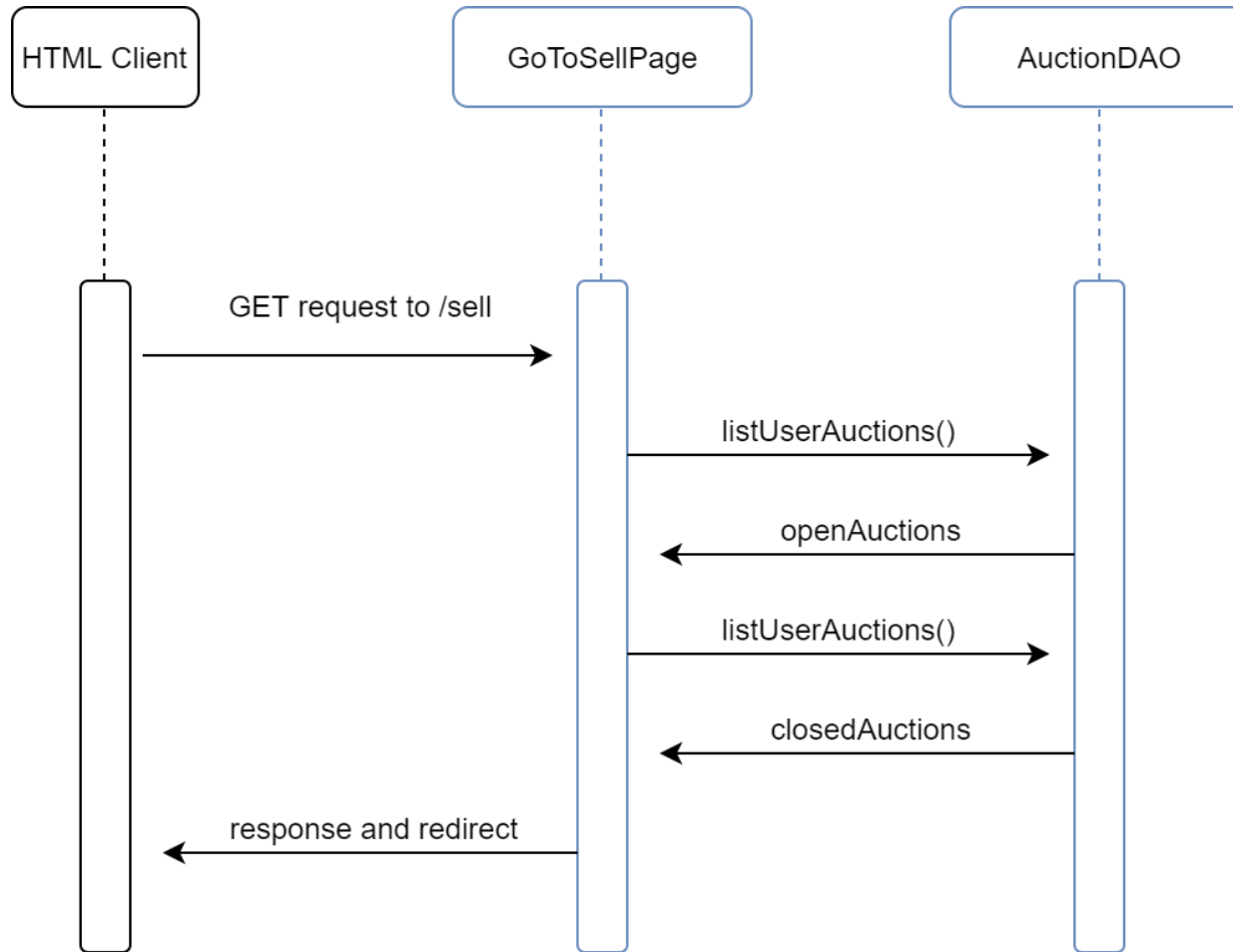
Il client RIA riceve risposte in formato JSON. Per semplificare il processo di encoding, si è scelto di utilizzare la libreria GSON di Google lato servlet. È stato necessario l'utilizzo di un'estensione di GSON per assicurare la compatibilità con le entità `java.time` di Java 8.

# Evento: client HTML accede a pagina ACQUISTO



La funzionalità di ricerca è gestita internamente allo stesso servlet.

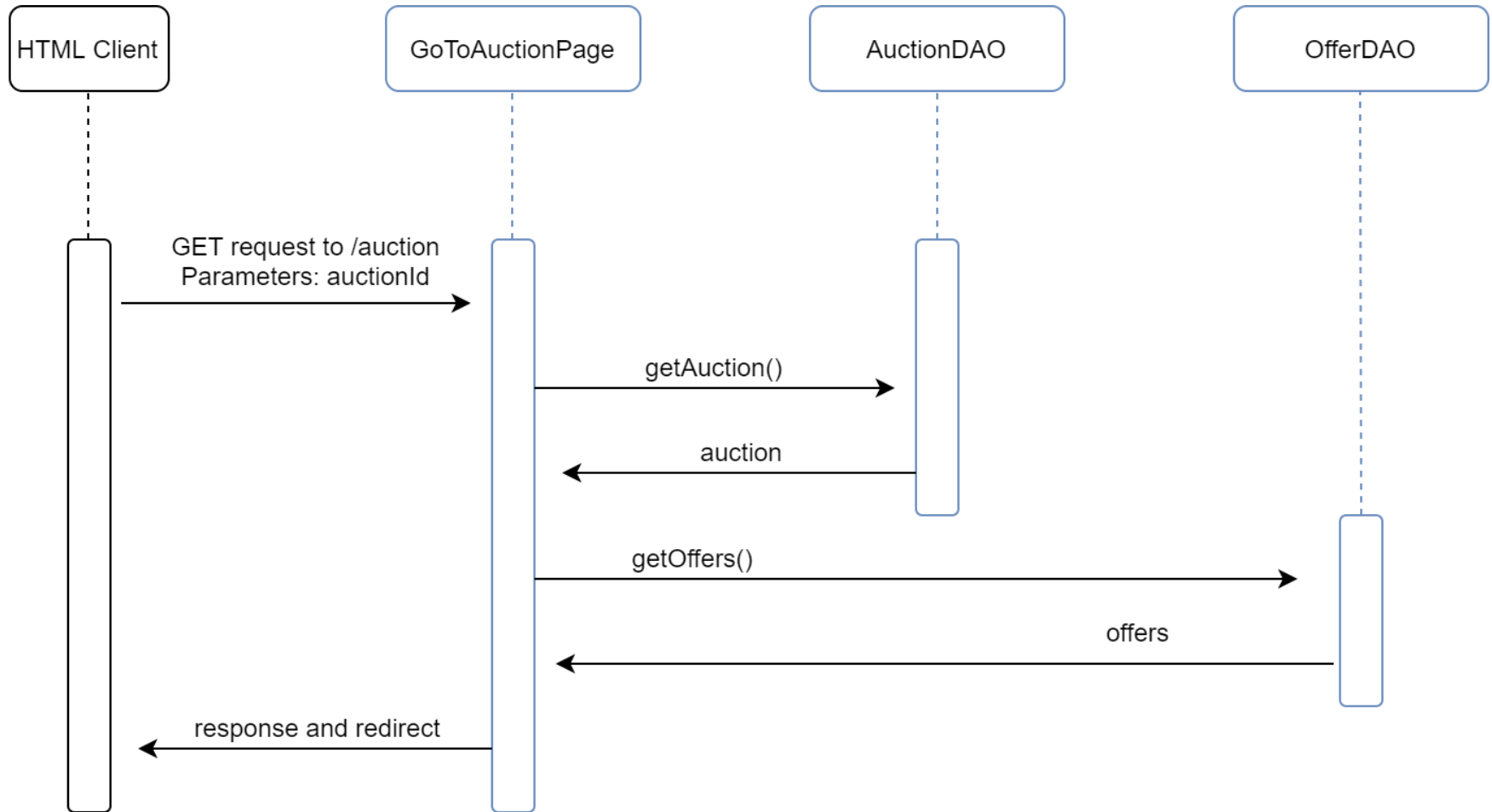
# Evento: client HTML accede a pagina VENDO



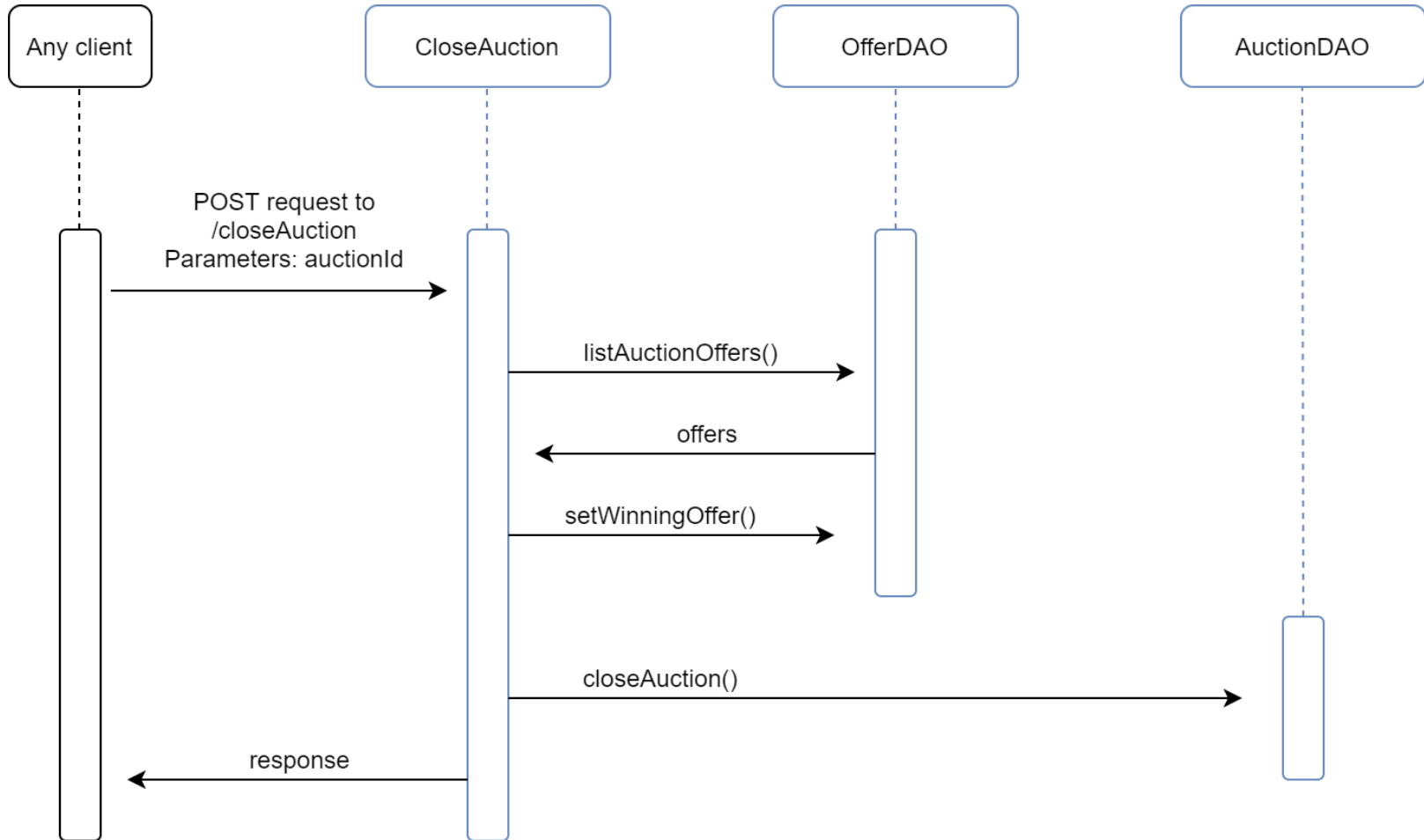
La struttura di questo evento riassume bene la funzionalità del client HTML puro.



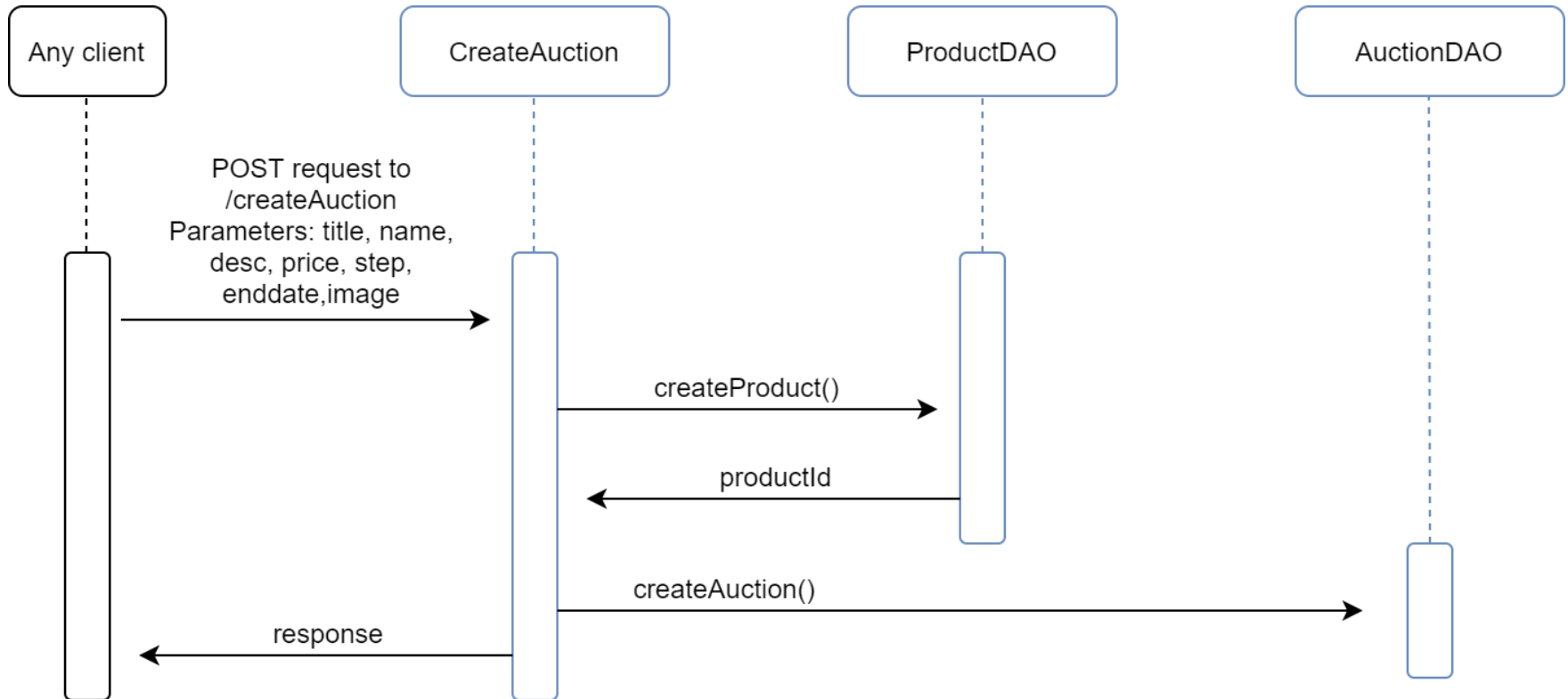
# Evento: client HTML accede a pagina OFFERTA o DETTAGLIO ASTA



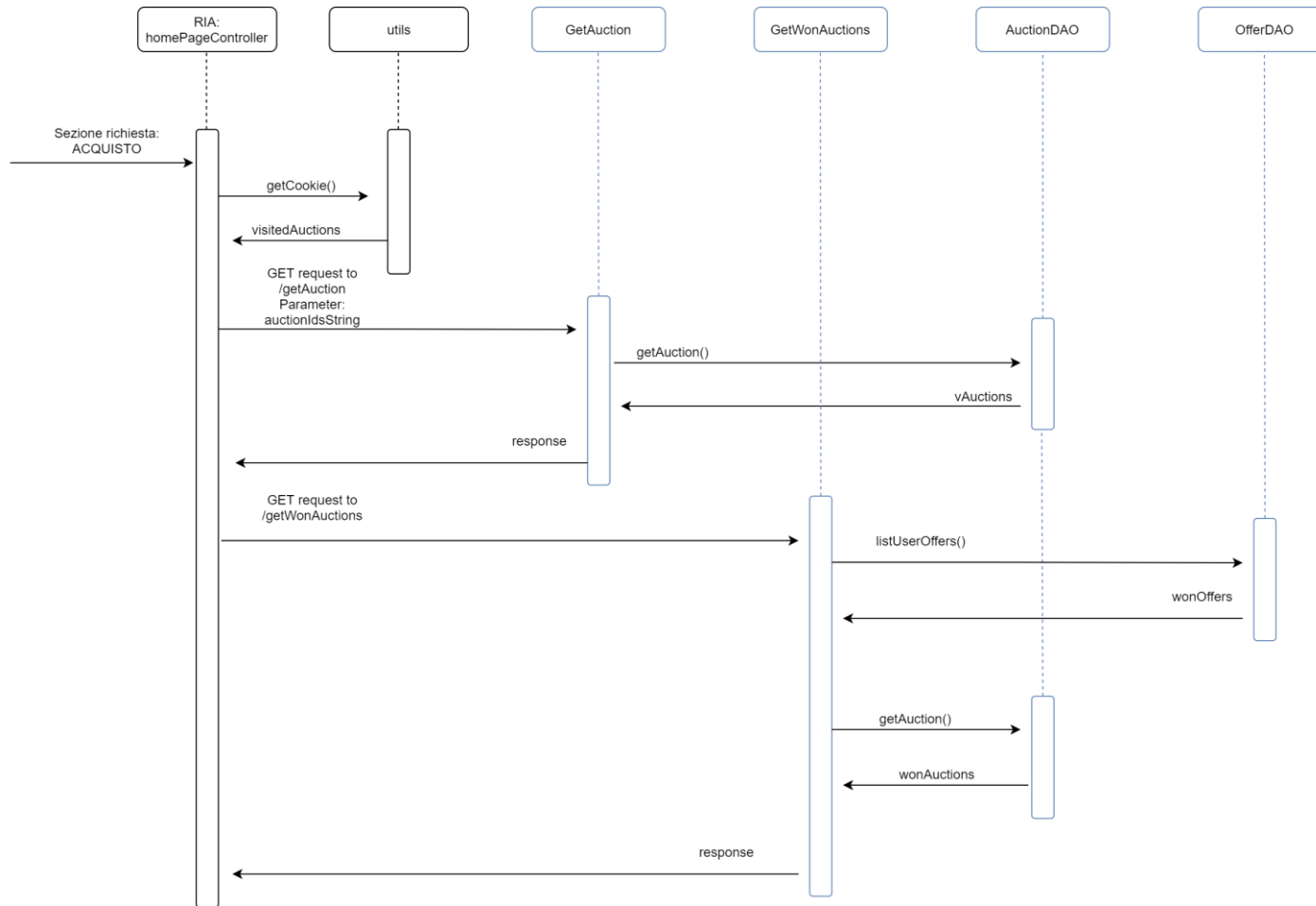
# Evento: client invia richiesta di chiusura di un'asta



# Evento: client invia richiesta di creazione di un'asta



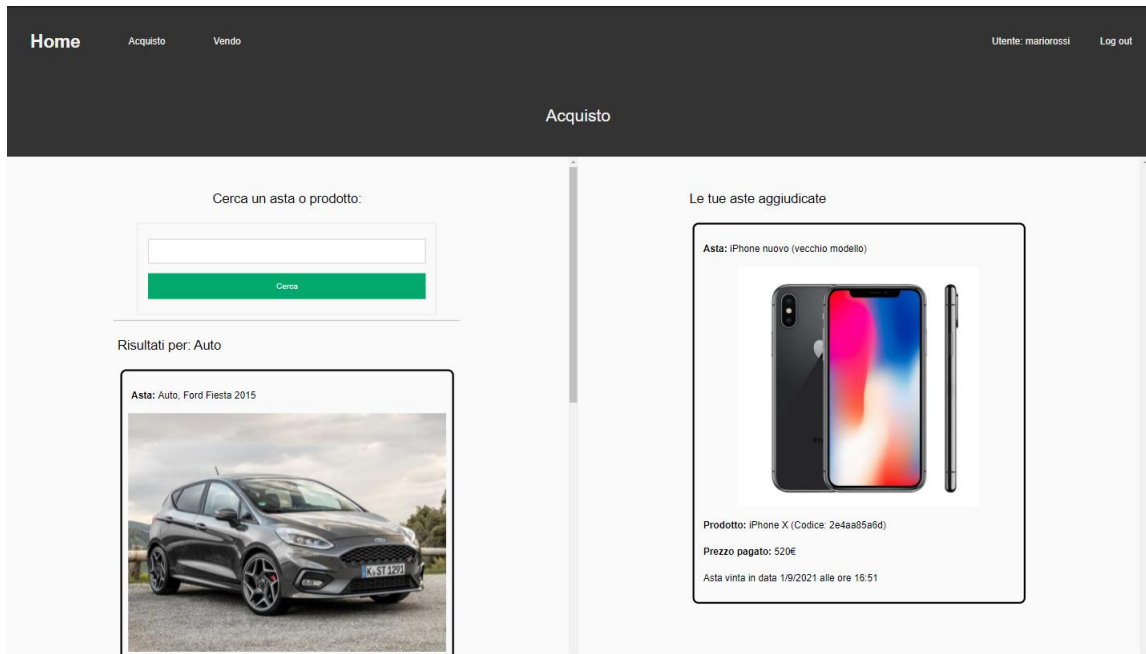
# Evento: client RIA richiede pagina ACQUISTO



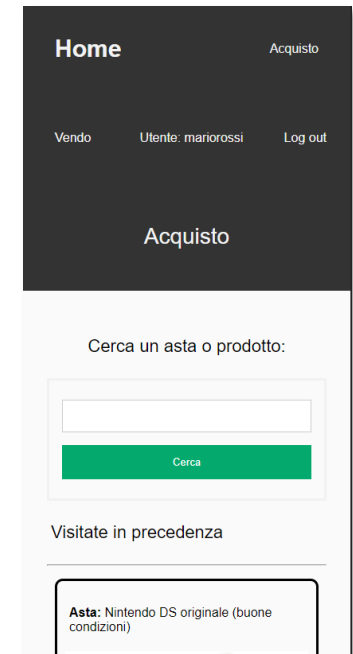
La struttura di questo evento riassume bene la funzionalità del client RIA.

# Breakdown Interfaccia

## Pagina Acquisto



## Versione mobile



# Breakdown Interfaccia


## Pagina Vendo

[Home](#) [Acquisto](#) [Vendo](#) Utente: shoponline Log out

Vendo

Le tue aste in corso:

**Asta:** Nintendo DS originale (buone condizioni)




Prodotto: Nintendo DS (2nd gen) (Codice: ffa963740f)  
Offerta massima corrente: 85€  
Asta creata in data 1/9/2021 alle ore 16:45  
Asta termina in data 28/9/2021 alle ore 16:42

**Asta:** Scrivania da ufficio nuova

Le tue aste concluse:

**Asta:** iPhone nuovo (vecchio modello)



Prodotto: iPhone X (Codice: 2e4aa85a5d)  
Offerta massima: 520€  
Asta conclusa in data 1/9/2021 alle ore 16:51

Crea nuova asta

**Titolo asta**

**Nome del prodotto**

**Descrizione del prodotto**

**Carica un'immagine del tuo prodotto**

**Prezzo di partenza (€)**

**Rialzo minimo**

**Termine asta**

Inserisci immagine

## Versione mobile

[Home](#) [Acquisto](#)

Vendo Utente: shoponline Log out

Vendo

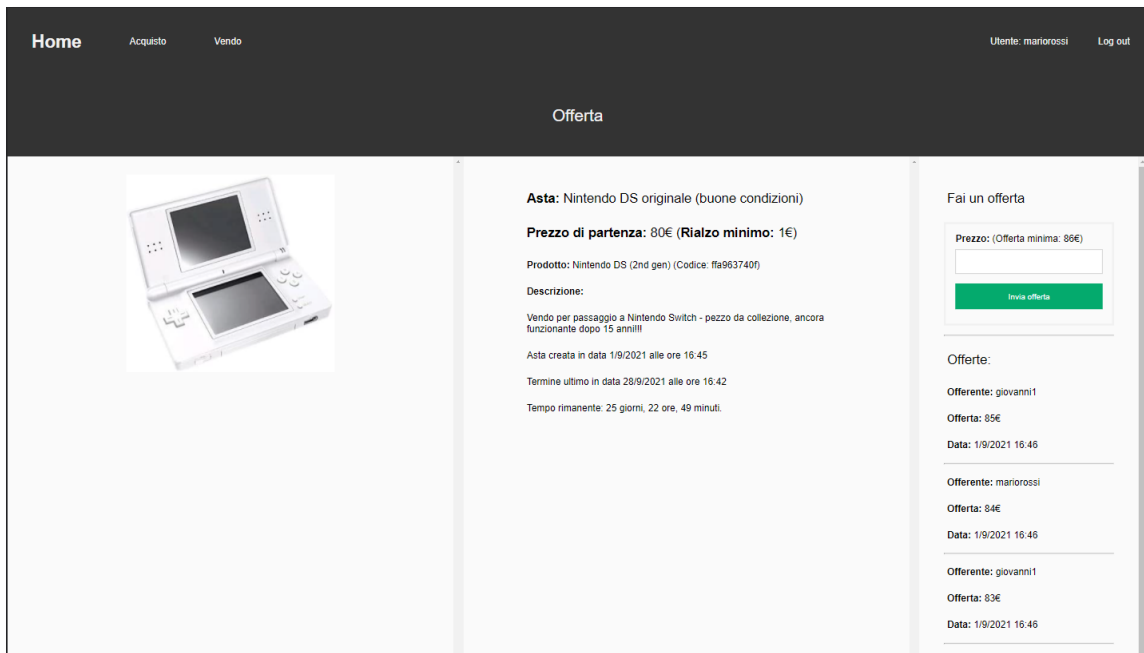
Le tue aste in corso:

**Asta:** Nintendo DS originale (buone condizioni)



# Breakdown Interfaccia

## Pagina Offerta



## Versione mobile



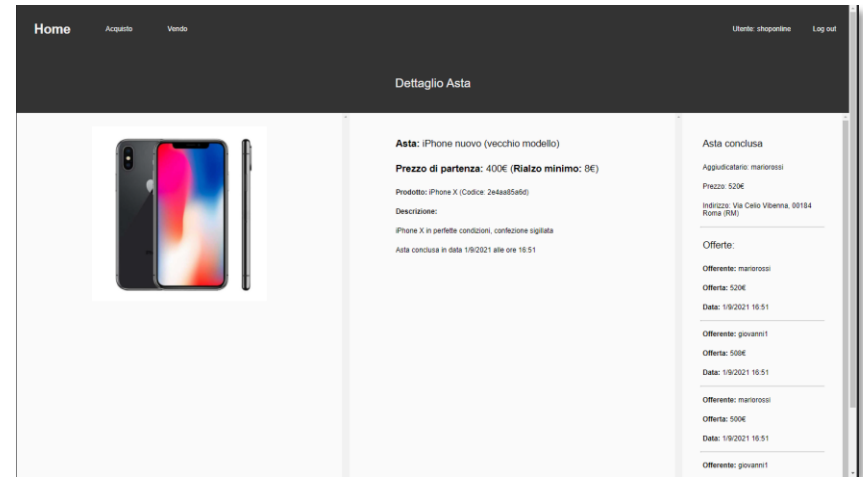
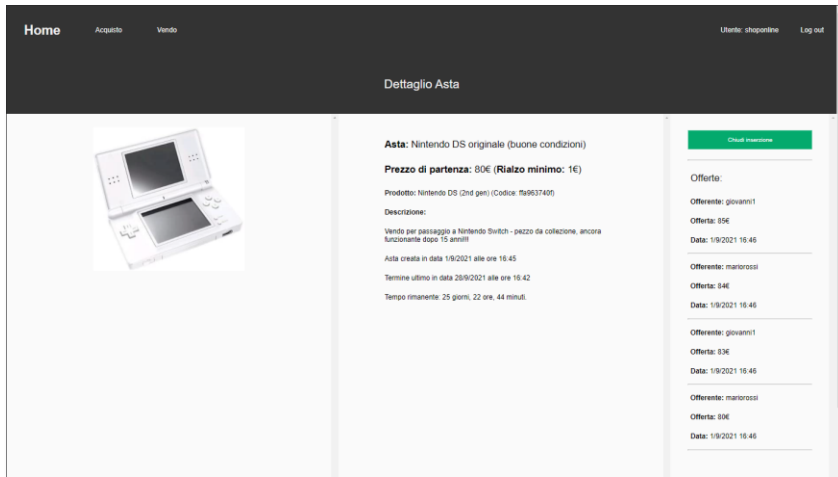


# Breakdown Interfaccia

## Pagina Dettaglio Asta

Aperta

Chiusa



# References

- <https://github.com/xevizero/AuctionsWebsiteProject>
- <https://github.com/gkopff/gson-javatime-serialisers>
- <https://github.com/google/gson>