RIA

ESERCIZIO 1

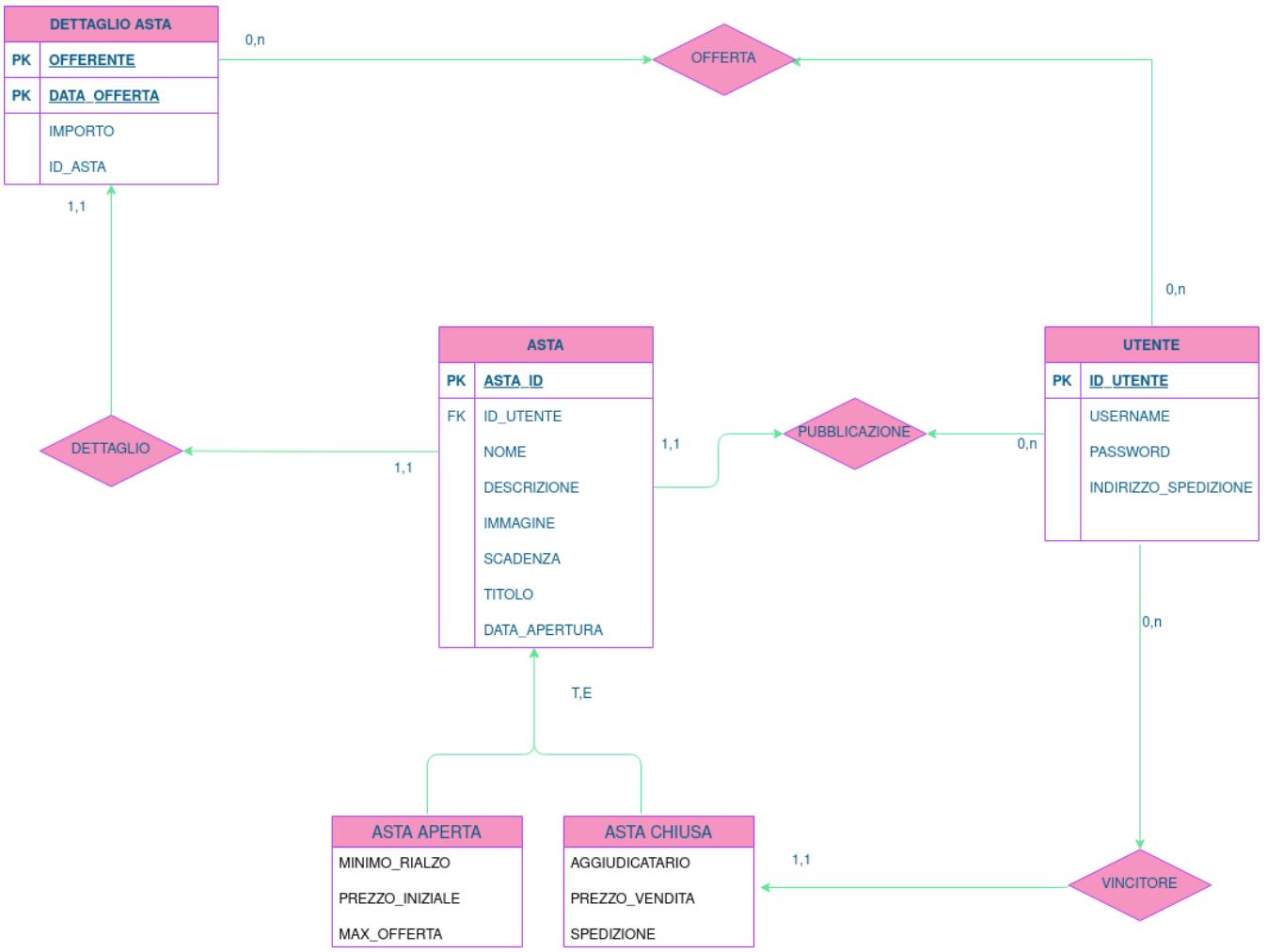
Armillotta Domenico - Borghetti Matteo

Data requirements analysis:

Un'applicazione web consente la gestione di aste online. Gli utenti accedono tramite login e possono vendere e acquistare all'asta. La HOME page contiene due link, uno per accedere alla pagina VENDO e uno per accedere alla pagina ACQUISTO. La pagina VENDO mostra una lista delle aste create dall'utente e non ancora chiuse, una lista delle aste da lui create e chiuse e una form per creare un nuovo articolo e una nuova asta per venderlo. L'asta comprende l'articolo da mettere in vendita (codice, nome, descrizione, immagine), prezzo iniziale, rialzo minimo di ogni offerta e una scadenza (data e ora, es 19-04-2021 alle 23:00). La lista delle aste è ordinata per data+ora crescente e riporta: codice e nome dell'articolo, offerta massima, tempo mancante (numero di giorni e ore) tra il momento del login e la data e ora di chiusura dell'asta. Cliccando su un'asta compare una pagina DETTAGLIO ASTA che riporta per un'asta aperta i dati dell'asta e la lista delle offerte (nome utente, prezzo offerto, data e ora dell'offerta) ordinata per data+ora decrescente. Un bottone CHIUDI permette all'utente di chiudere l'asta se è giunta l'ora della scadenza (si ignori il caso di aste scadute ma non chiuse). Se l'asta è chiusa, la pagina riporta i dati dell'asta, il nome dell'aggiudicatario, il prezzo finale e l'indirizzo di spedizione dell'utente. La pagina ACQUISTO contiene una form di ricerca per parola chiave. Quando l'acquirente invia una parola chiave la pagina ACQUISTO si ricarica e mostra un elenco di aste aperte (la cui scadenza è posteriore all'ora dell'invio) il cui articolo contiene la parola chiave nel nome o nella descrizione. La lista è ordinata in modo decrescente in base al tempo (numero di giorni. ore e minuti) mancante alla chiusura. Cliccando su un'asta aperta compare la pagina OFFERTA che mostra i dati dell'articolo, l'elenco delle offerte pervenute in ordine di data+ora decrescente e un campo di input per inserire la propria offerta, che deve essere superiore all'offerta massima corrente. Dopo l'invio dell'offerta la pagina OFFERTA mostra l'elenco delle offerte aggiornate. La pagina ACQUISTO contiene anche un elenco delle offerte aggiudicate all'utente con i dati dell'articolo e il prezzo finale.

Entities, attributes, relationships

Database Design



Application requirements analysis:

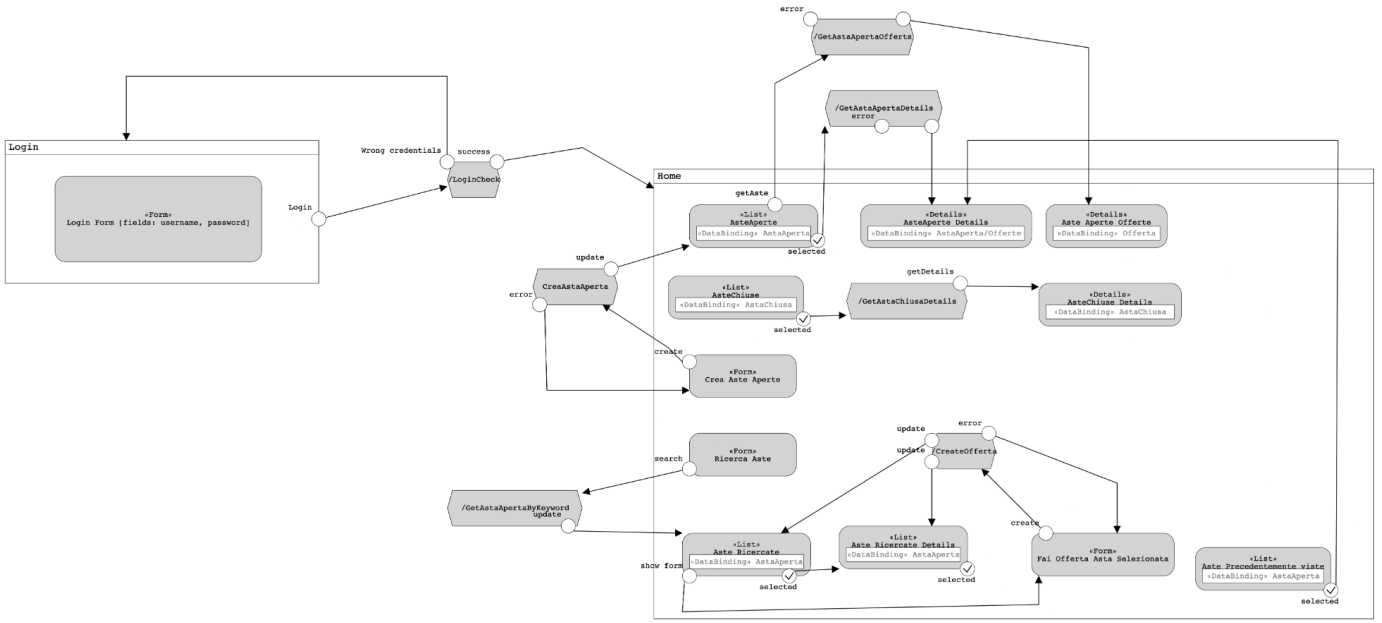
Un'applicazione web consente la gestione di aste online. Gli utenti accedono tramite login e possono vendere e acquistare all'asta. La HOME page contiene due link, uno per accedere alla pagina VENDO e uno per accedere alla pagina ACQUISTO. La pagina VENDO mostra una lista delle aste create dall'utente e non ancora chiuse, una lista delle aste da lui create e chiuse e una form per creare un nuovo articolo e una nuova asta per venderlo. L'asta comprende l'articolo da mettere in vendita (codice, nome, descrizione, immagine), prezzo iniziale, rialzo minimo di ogni offerta e una scadenza (data e ora, es 19-04-2021 alle 23:00). La lista delle aste è ordinata per data+ora crescente e riporta: codice e nome dell'articolo, offerta massima, tempo mancante (numero di giorni e ore) tra il momento del login e la data e ora di chiusura dell'asta. Cliccando su un'asta compare una pagina DETTAGLIO ASTA che riporta per un'asta aperta i dati dell'asta e la lista delle offerte (nome utente, prezzo offerto, data e ora dell'offerta) ordinata per data+ora decrescente. Un bottone CHIUDI permette all'utente di chiudere l'asta se è giunta l'ora della scadenza (si ignori il caso di aste scadute ma non chiuse). Se l'asta è chiusa, la pagina riporta i dati dell'asta, il nome dell'aggiudicatario, il prezzo finale e l'indirizzo di spedizione dell'utente. La pagina ACQUISTO contiene una form di ricerca per parola chiave. Quando l'acquirente invia una parola chiave la pagina ACQUISTO si ricarica e mostra un elenco di aste aperte (la cui scadenza è posteriore all'ora dell'invio) il cui articolo contiene la parola chiave nel nome o nella descrizione. La lista è ordinata in modo decrescente in base al tempo (numero di giorni. ore e minuti) mancante alla chiusura. Cliccando su un'asta aperta compare la pagina OFFERTA che mostra i dati dell'articolo, l'elenco delle offerte pervenute in ordine di data+ora decrescente e un campo di input per inserire la propria offerta, che deve essere superiore all'offerta massima corrente. Dopo l'invio dell'offerta la pagina OFFERTA mostra l'elenco delle offerte aggiornate. La pagina ACQUISTO contiene anche un elenco delle offerte aggiudicate all'utente con i dati dell'articolo e il prezzo finale.

Si realizzi un'applicazione client server web che modifica le specifiche precedenti come segue:

- Dopo il login, l'intera applicazione è realizzata con un'unica pagina.
- Se l'utente accede per la prima volta l'applicazione mostra il contenuto della pagina ACQUISTO. Se l'utente ha già usato l'applicazione, questa mostra il contenuto della pagina VENDO se l'ultima azione dell'utente è stata la creazione di un'asta; altrimenti mostra il contenuto della pagina ACQUISTO con l'elenco (eventualmente vuoto) delle aste su cui l'utente ha cliccato in precedenza e che sono ancora aperte. L'informazione dell'ultima azione compiuta e delle aste visitate è memorizzata a lato client per la durata di un mese.
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica solo del contenuto da aggiornare a seguito dell'evento.

  pages(views),view components,events,actions

Application Design (IFML)



EVENT & Actions

Client Side		Server Side	
Evento	Azione	Evento	Azione
login.html➔login form➔submit	Data Check	POST(username,password)	Data check
home.html➔load	based on the cookies it decides to show “vendita” or “Acquisto”page	-	-
home.html➔asta aperta list ➔ asta selection	Update views and display AstaAperta Details and Offerts	2 GET(asta_id)	Retrieval of data from db (AstaAperta/OfferteAsta) and Json conversion
home.html➔asta chiusa list ➔ asta selection	Update views and display AstaChiusa Details	GET(asta_id)	Retrieval of data from db (AstaChiusa) and Json conversion
Home.html➔asta Aperta Details ➔ close	Close Asta Aperta and update list Asta Aperta and list AstaChiusa	POST(asta_id)	Close AstaAperta, delete selected asta in db AstaAperta and add selected Asta in db AstaChiusa
Home.html➔href acquisito	show components of the page “Acquisto”	-	-
Home.html➔href vendita	show components of the page “Vendita”	-	-
Home.html➔create asta form➔submit	Create AstaAperta on database and update list AstaAperta	POST(asta details)	Data check & insertion
Home.html➔search asta da keyword➔submit	Do a query by keyword and update list of Searched	GET(asta_id)	Retrieval of data from db (AstaAperta) and Json conversion

	Auction		
Home.html➔createOfferts➔submit	Update database and list AsteRicercateDetails,list AsteRicercate	POST(offerts details)	Data check & insertion
Home.html➔asta cercata➔show	Show AstaRicercataDetails	2 GET(asta_id)	Retrieval of data from db (AstaAperta/OfferteAsta) and Json conversion
Home.html➔list “aste precedentemente ricercate”➔Show	Show details of auction clicked	2 GET(asta_id)	Retrieval of data from db (AstaAperta/OfferteAsta) and Json conversion

Controller & Event Handlers

Client Side		Server Side	
Evento	Controllore	Evento	Controllore
login.html➔login form➔submit	Function makeCall	POST(username,password)	LoginCheck(servlet)
home.html➔load	Function Page Orchestrator ➔AstaAperta.show ➔AstaChiusa.show	GET(no parameters) GET(no parameters)	GetAstaAperta(servlet) GetAstaChiusa (servlet)
Home.html➔asta Aperta Details ➔ close	Function AstaAperta.close	POST(asta_id)	CloseAstaAperta(servlet)
Home.html➔create asta form➔submit	Function Wizard.registerEvents	POST(info asta)	CreaAstaAperta(servlet)
Home.html➔search	Function	Get(keyword)	GetAstaApertaByKe

asta da keyword➔submit	AstebykeywordForm.registerEvents		yword(servlet)
Home.html➔createOfferts➔submit	Function FromForOfferta.ProcessForm	Post(asta_id,Offerts)	CreateOfferta(servlet)
home.html➔asta aperta list ➔ asta selection	Function AstaApertaDetails	GET(Asta_id)	GetAstaAperta(servlet)
home.html➔asta chiusa list ➔ asta selection	Function AstaChiusaDetails	GET(Asta_id)	GetAstaChiusa (servlet)
Home.html➔asta cercata➔show	Function AstaApertaDetailsForOfferta	GET(Asta_id)	GetAstaAperta(servlet)
Home.html➔list “aste precedentemente ricercate”➔Show	Function astaCookie.show	GET(Asta_id)	GetAstaApertaFromCookies(servlet)

Server Side Components:

- **Model Object (Beans)**
  - Asta
  - AstaAperta
  - AstaChiusa
  - Offerta
  - Utente

- **Data Access Object (Classes)**
  - AstaApertaDao
  - AstaChiusaDao
  - loginDao
  - OffertaDao
  - UtenteDao
- **Controllers (servlets)**
  - CloseAstaAperta
  - CreaAstaAperta
  - CreateOfferta
  - GetAstaApertaByKeyword
  - GetAstaApertaDetails
  - GetAstaChiusaDetails
  - GetAsteAperteFromCookies
  - GetListaAsteAperte
  - GetListaAsteChiuse
  - LoginCheck
  - Logout
- **Views(Template)**
  - Home
  - index

- **Filters**
  - Checker

Client Side Components:

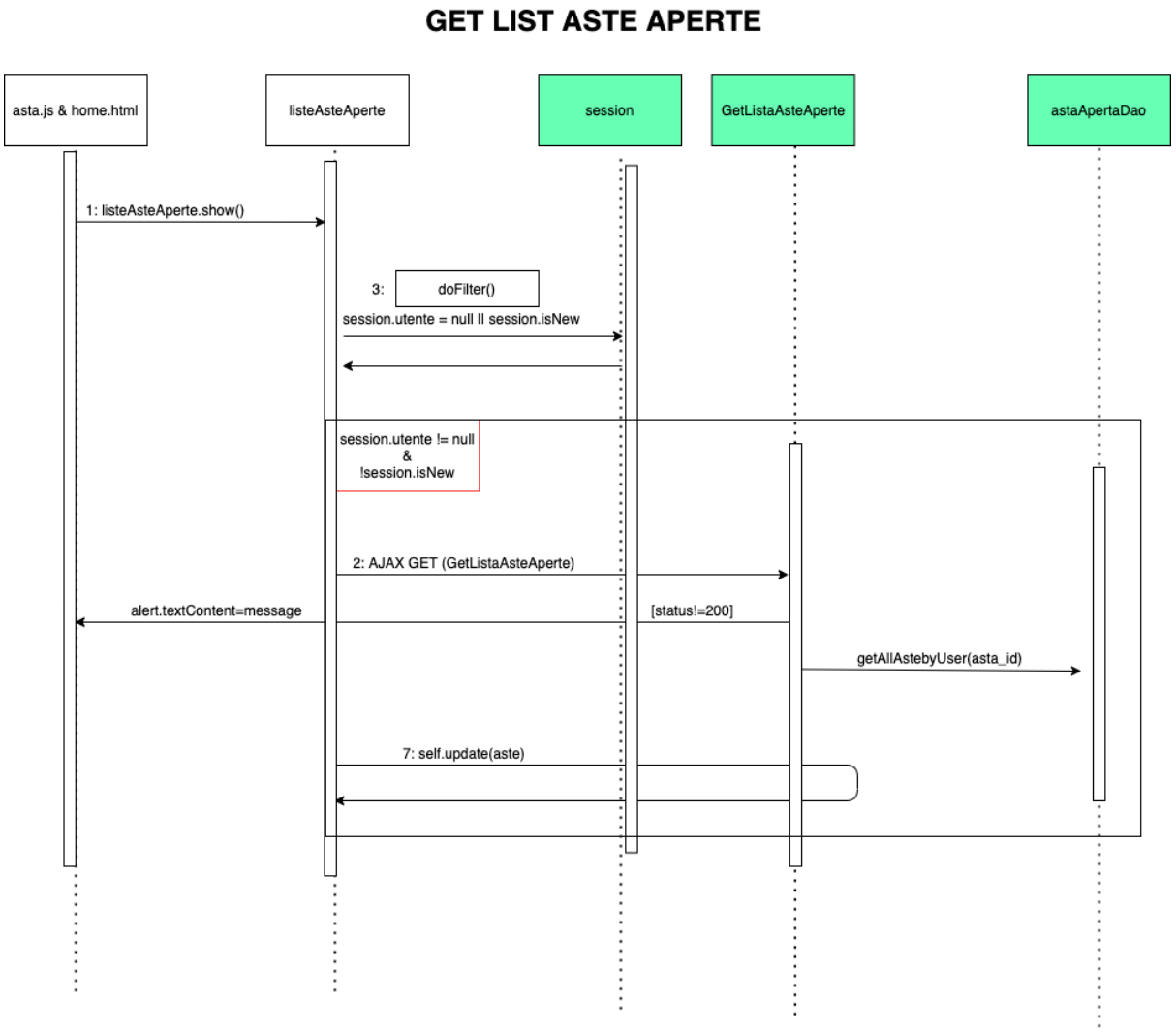
- **Login (index)**
  - Login Form
- **Home**
  - PageOrchestrator
    - refresh(): loads page components
    - start(): initialized page component
  - ListeAsteAperte
    - show(): show list and do makeCall to get data from database “AstaAperta”
    - reset(): hide list
    - update(): update and create the list html with data
  - ListeAsteChiuse
    - show():show list and do makeCall to get data from database “AstaChiusa”
    - reset(): hide list
    - update():update and create the list html with data
  - ListaOfferteAsteAperte
    - show(asta_id):show list and do makeCall to get data from the database “Offerte”
    - reset():hide list
    - update():update and create the list html with data
  - Manager
    - showAcquisto(): show the components of the page “Acquisto”
    - showVendita(): show the components of the page “Vendita”
    - showdettagliPage(): show the components of the page “Dettagli”, this page is for making offers
    - resetAcquisto():hide the components of the page “Acquisto”

        - resetVendita(): hide the components of the page “Vendita”
        - resetdettagliPage(): hide the components of the page “Dettagli”, this page is for making offers

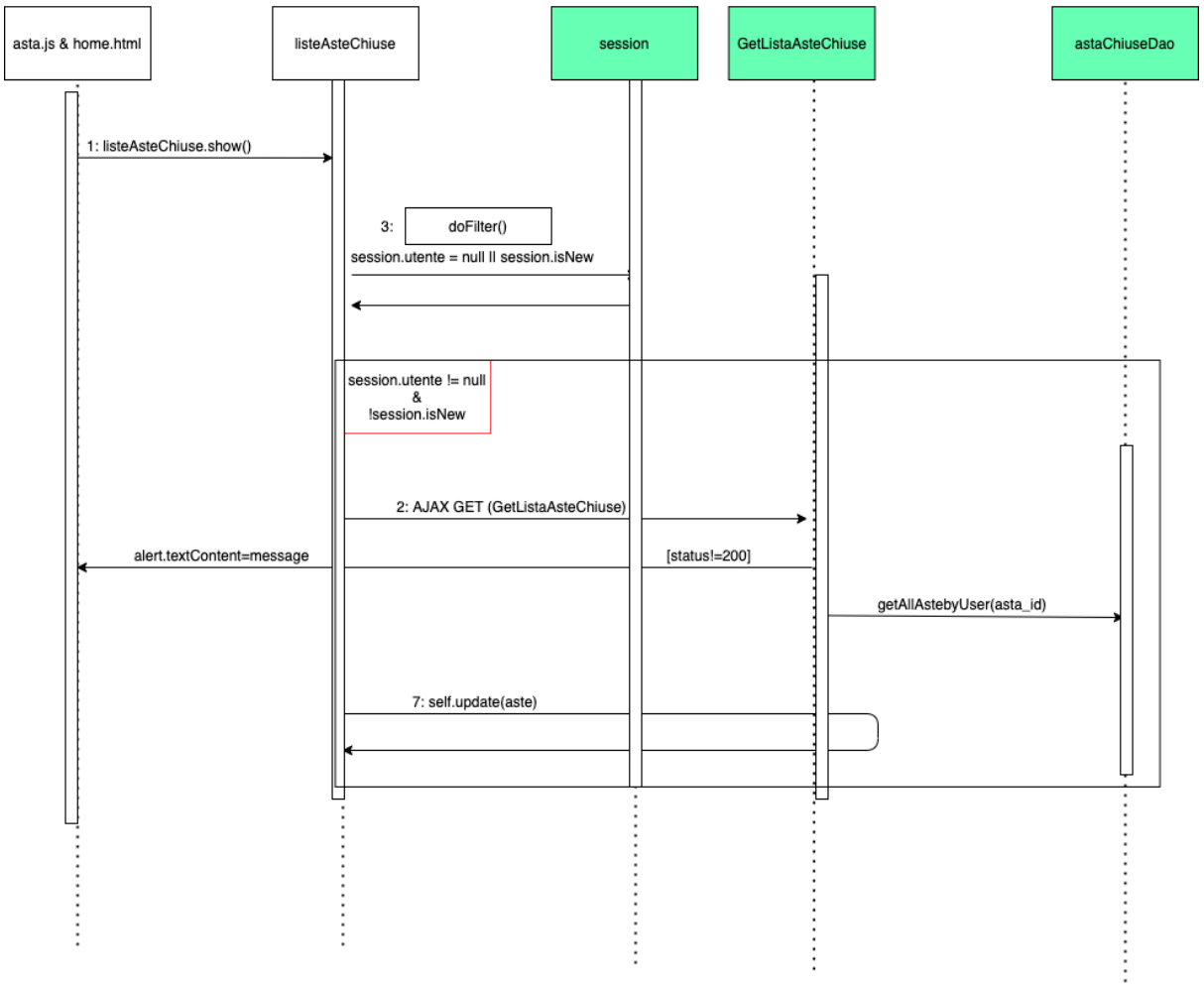
    - AstaChiusaDetails:
        - show(asta_id): show list and do makeCall to get data from database “AstaChiusa”
        - reset(): hide list
        - update(): update and create the list html with data
    - AstaApertaDetails:
        - show(asta_id):show list and do makeCall to get data from database “AstaAperta”
        - reset(): hide list
        - update():update and create the list html with data
    - Wizard
        - registerEvents(): takes the data from the form to create the auction, and then calls a post to insert it into the database, the does an update of ListaAsteAperte
    - ListaByKeyword
        - update(asteKeywordArray):display asteKeywordArray(an array of asteAperte with the keyword in their title or description).
        - reset():hide list of aste
    - AsteKeywordForm
        - registerEvents(orchestrator):manage submit of a keyword for searching a particular AstaAperta and handle any potential error.
        - updateOfferteAfterOffertaCorrect():if a user made a valid new offert , reload data of asta Aperte with the same keyword.
    - FormForOfferta

        - processForm(asta_id):manage submit of a new offert and handles any potential error
        - show(): show the form for offerta
        - reset(): hide form
    - AsteAperteDetailsForOfferta
        - show(): load details for a specific auction during offert’s phase
        - update(): reload details for a specific auction
        - reset(): hide asteAperte details
    - AstaCookie
        - show() : display previously clicked auctions
        - update(): refresh list of auctions previously clicked
        - reset(): hide asteAperte

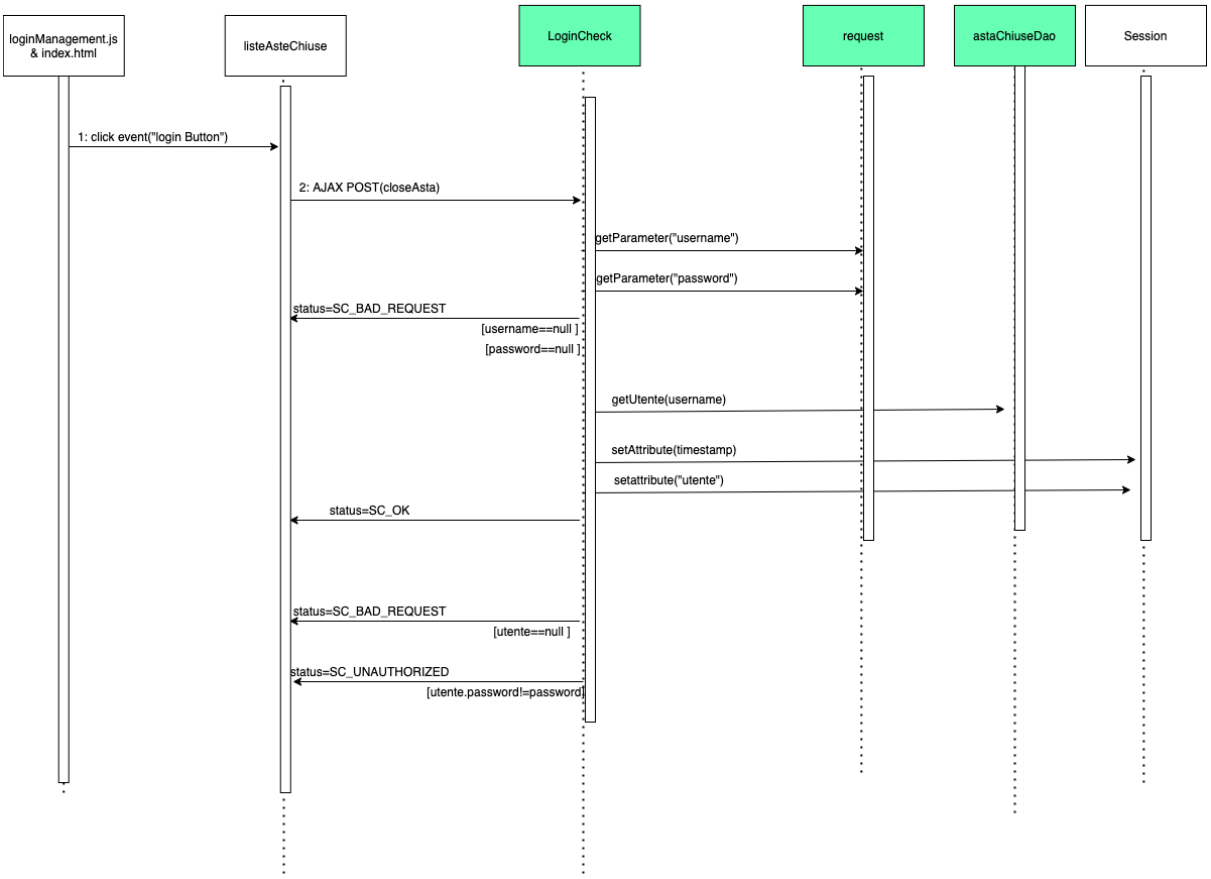
Sequence diagrams

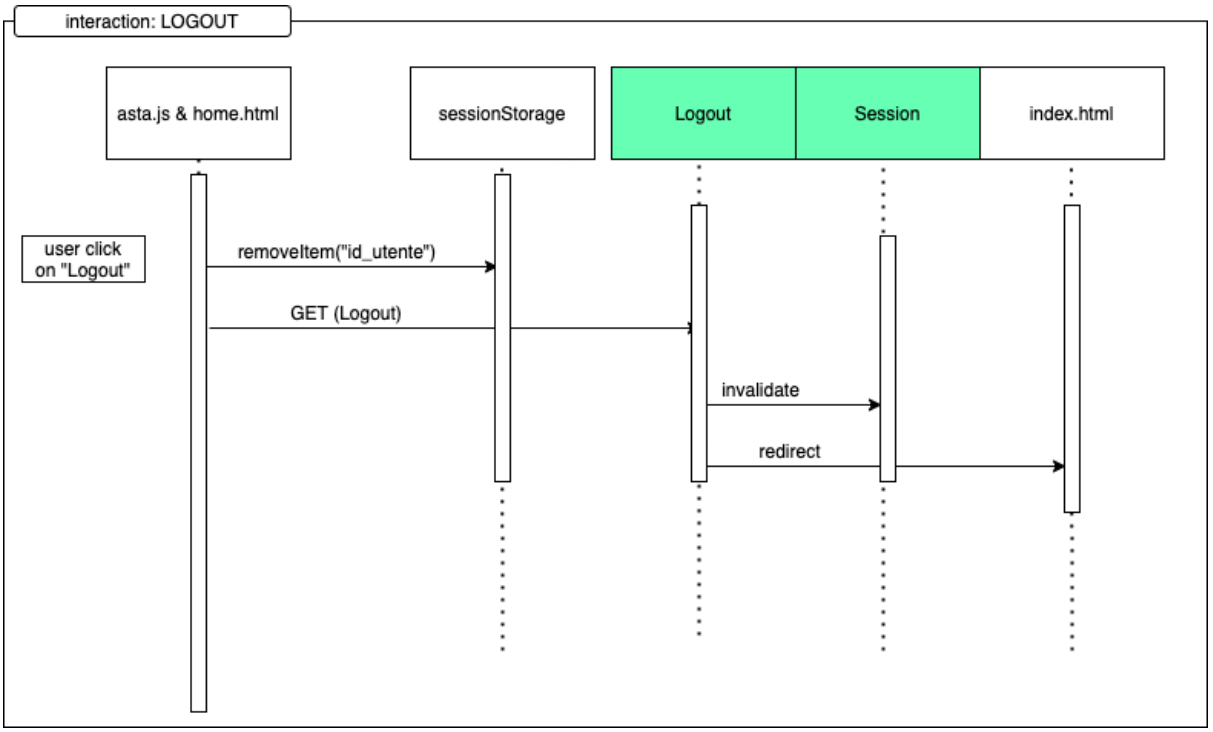


GET LIST ASTE CHIUSE

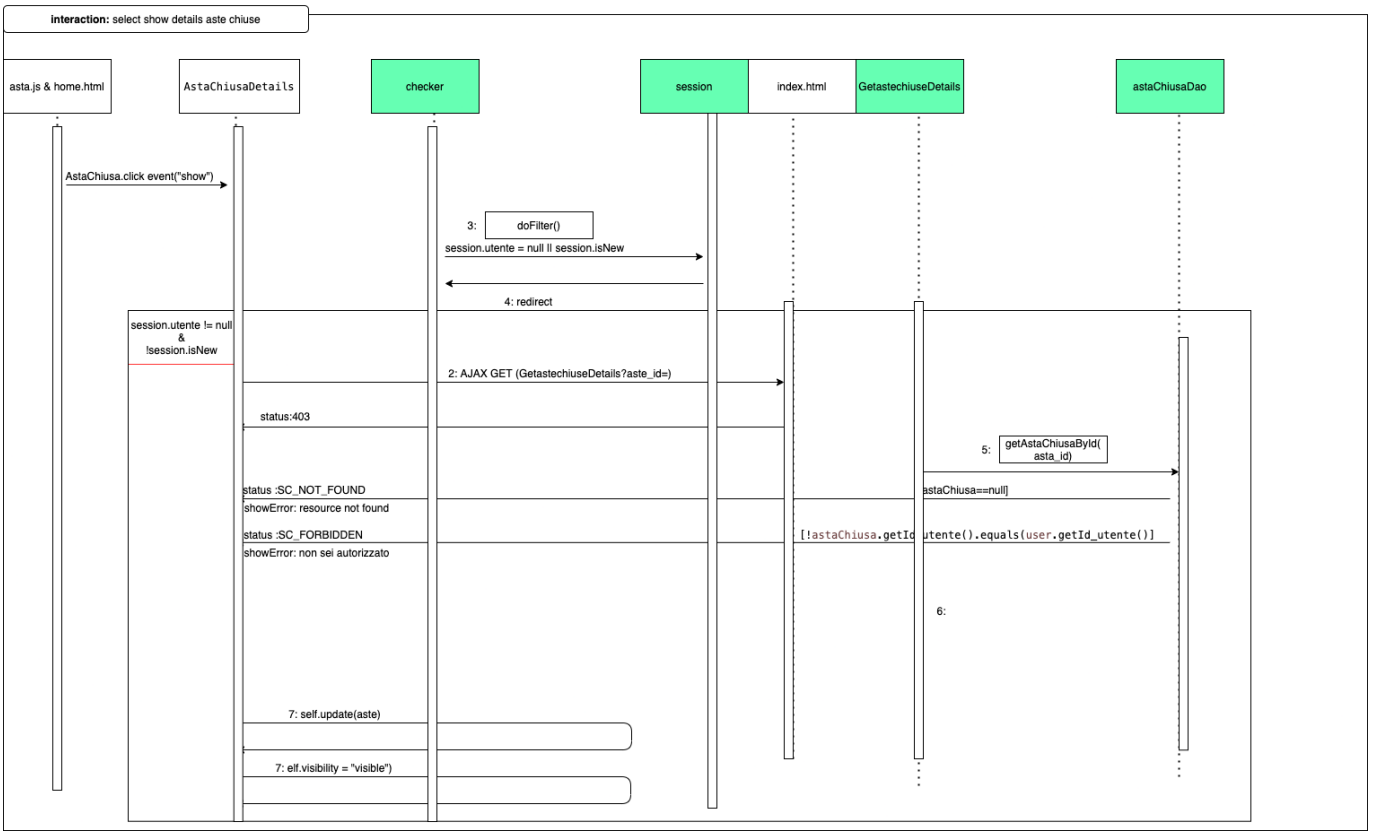


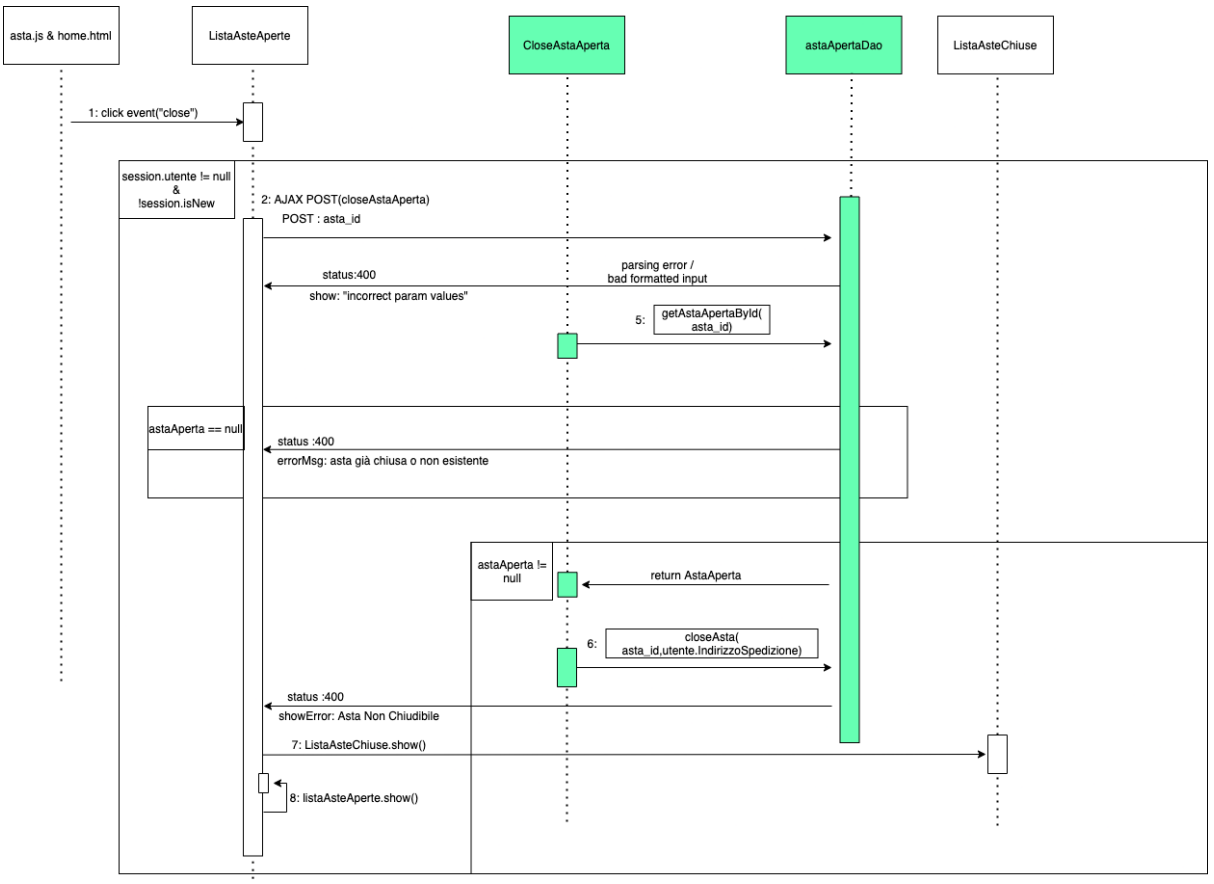
LOGIN CHECK

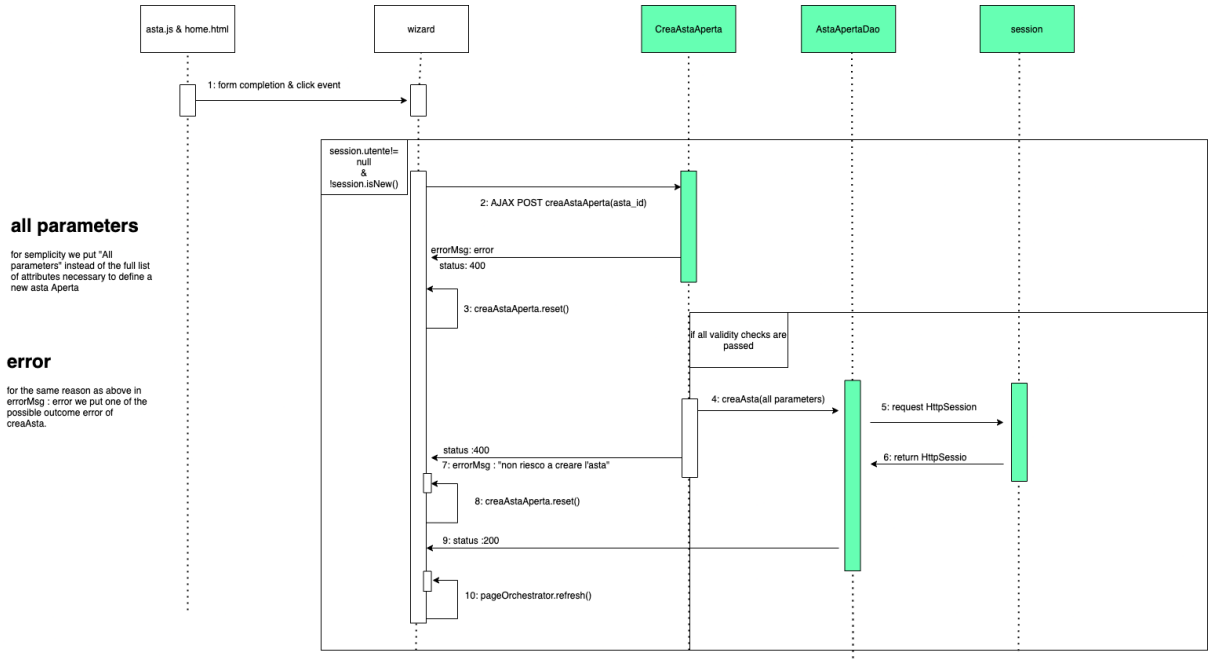

loginManagement.js & index.html
listeAsteChiuse
LoginCheck
request
astaChiuseDao
Session
1: click event("login Button")
2: AJAX POST(closeAsta)
getParameter("username")
getParameter("password")
status=SC_BAD_REQUEST
[username==null]
[password==null]
getUtente(username)
setAttribute(timestamp)
setattribute("utente")
status=SC_OK
status=SC_BAD_REQUEST
[utente==null]
status=SC_UNAUTHORIZED
[utente.password!=password]


interaction: LOGOUT
asta.js & home.html
sessionStorage
Logout
Session
index.html
user click on "Logout"
removeItem("id_utente")
GET (Logout)
invalidate
redirect

GET ASTA CHIUSA DETAILS





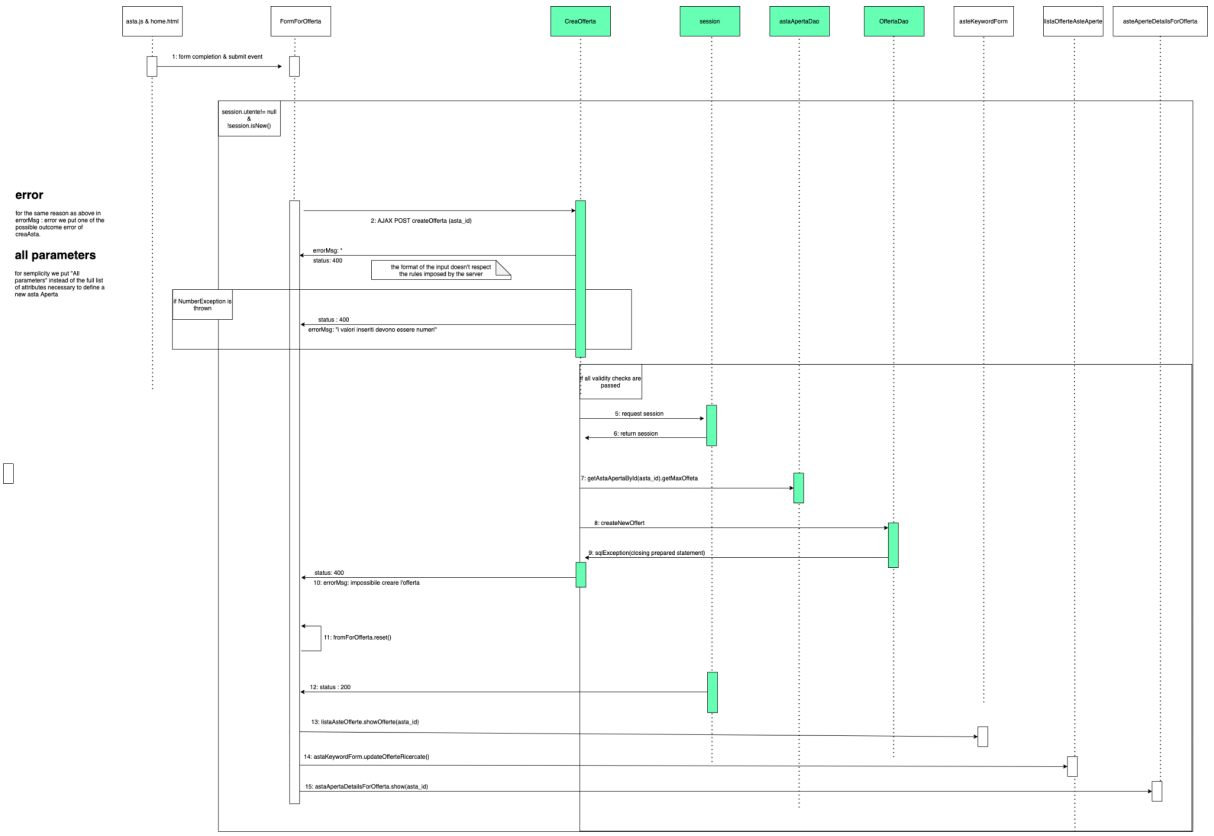


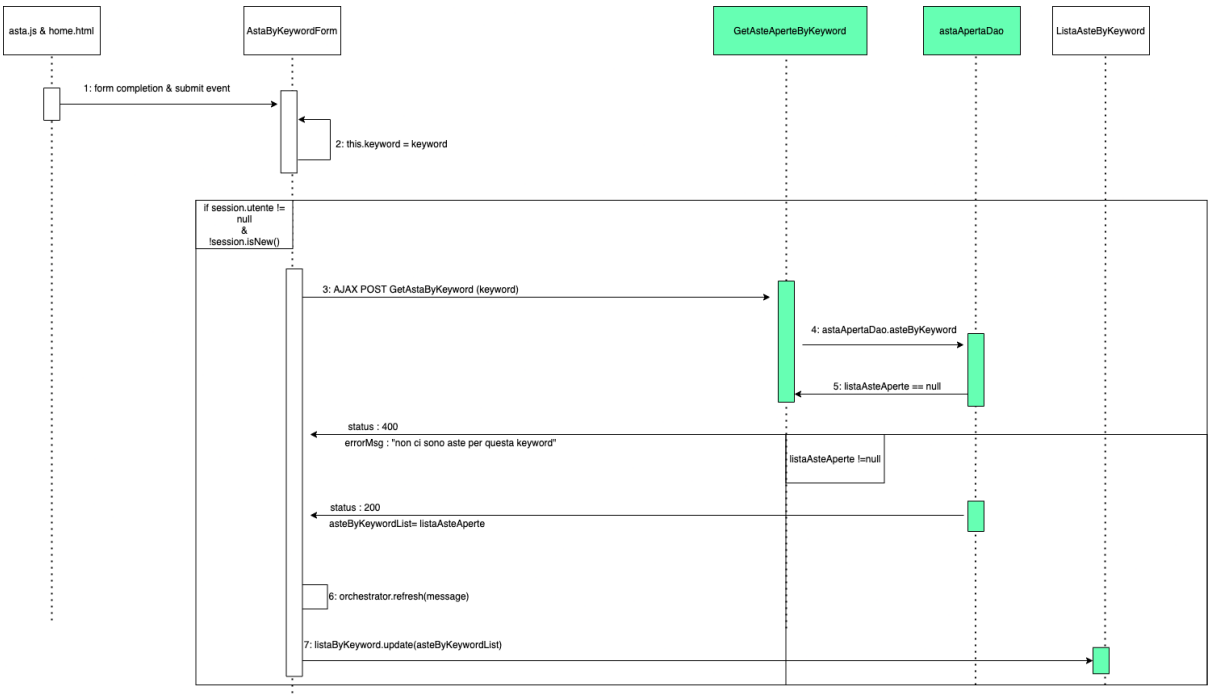
all parameters

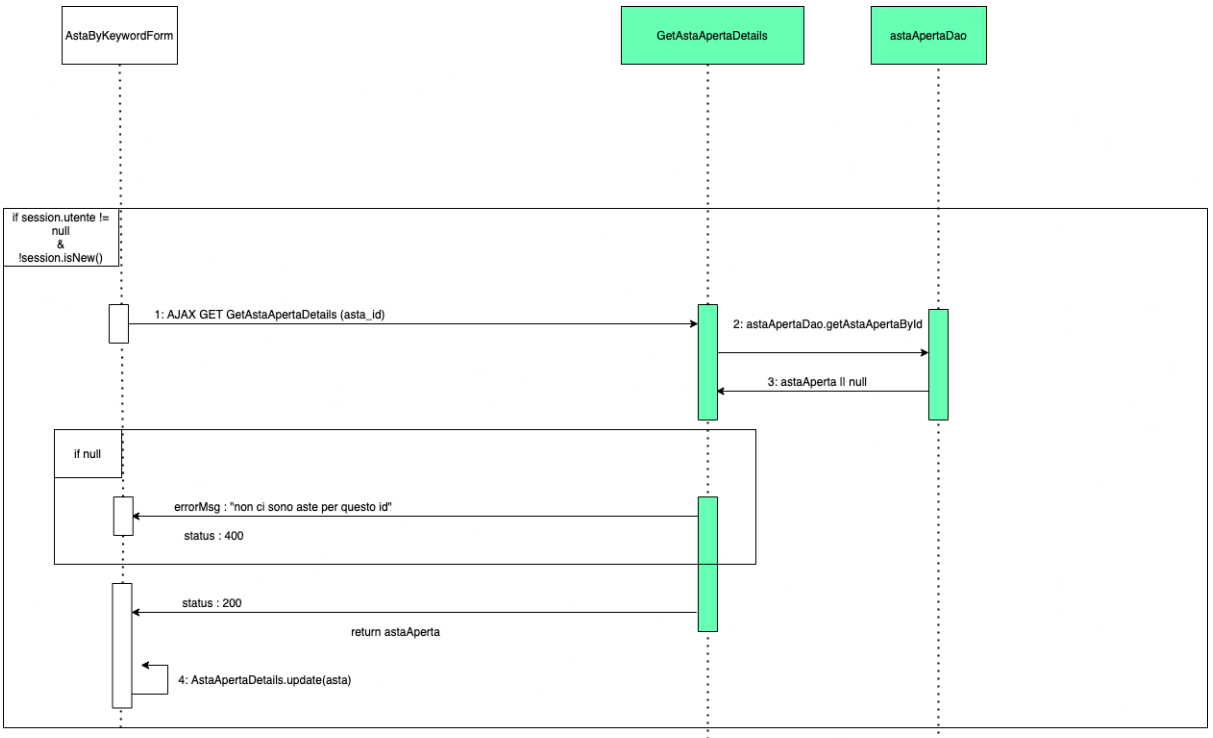
for semplicity we put "All parameters" instead of the full list of attributes necessary to define a new asta Aperta

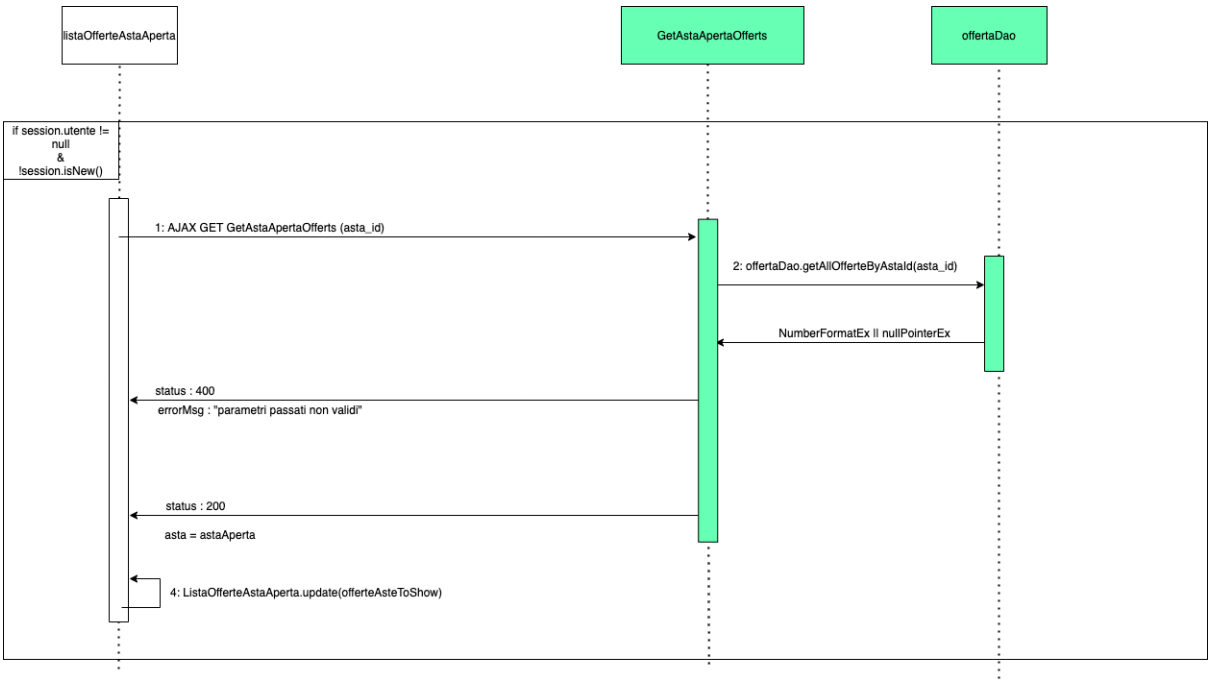
error

for the same reason as above in errorMsg : error we put one of the possible outcome error of creaAsta.

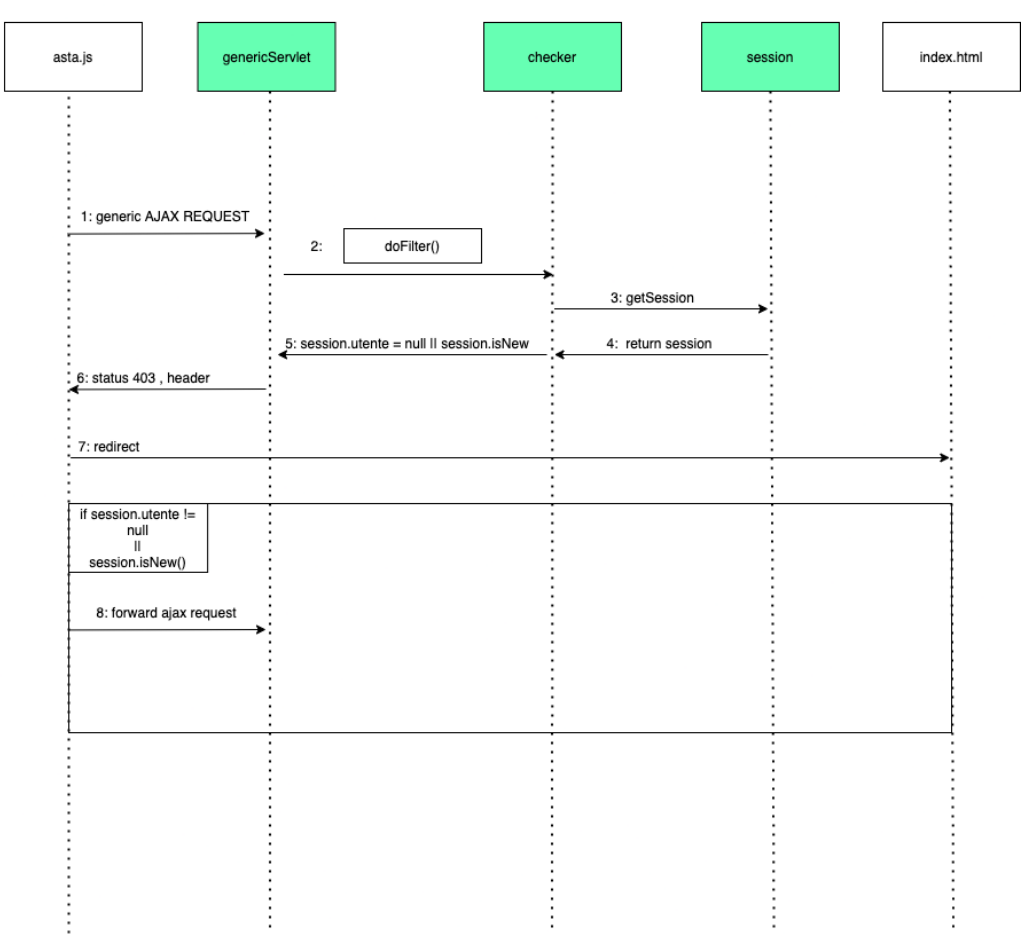

asta.js & home.html
FormForOfferta
CreaOfferta
session
astaApertaDao
OffertaDao
asteKeywordForm
listaOfferteAsteAperte
asteAperteDetailsForOfferta
1: form completion & submit event
session.utente!= null
&
!session.isNew()
error
for the same reason as above in errorMsg : error we put one of the possible outcome error of creaAsta.
all parameters
for semplicity we put "All parameters" instead of the full list of attributes necessary to define a new asta Aperta
2: AJAX POST createOfferta (asta_id)
errorMsg: "
status: 400
the format of the input doesn't respect the rules imposed by the server
if NumberException is thrown
status : 400
errorMsg: "I valori inseriti devono essere numeri"
if all validity checks are passed
5: request session
6: return session
7: getAstaApertaById(asta_id).getMaxOfeta
8: createNewOffert
9: sqlException(closing prepared statement)
status: 400
10: errorMsg: impossibile creare l'offerta
11: fromForOfferta.reset()
12: status : 200
13: listaAsteOfferte.showOfferte(asta_id)
14: astaKeywordForm.updateOfferteRicercate()
15: astaApertaDetailsForOfferta.show(asta_id)


asta.js & home.html
AstaByKeywordForm
GetAsteAperteByKeyword
astaApertaDao
ListaAsteByKeyword
1: form completion & submit event
2: this.keyword = keyword
if session.utente != null
&
!session.isNew()
3: AJAX POST GetAstaByKeyword (keyword)
4: astaApertaDao.asteByKeyword
5: listaAsteAperte == null
status : 400
errorMsg : "non ci sono aste per questa keyword"
listaAsteAperte !=null
status : 200
asteByKeywordList= listaAsteAperte
6: orchestrator.refresh(message)
7: listaByKeyword.update(asteByKeywordList)

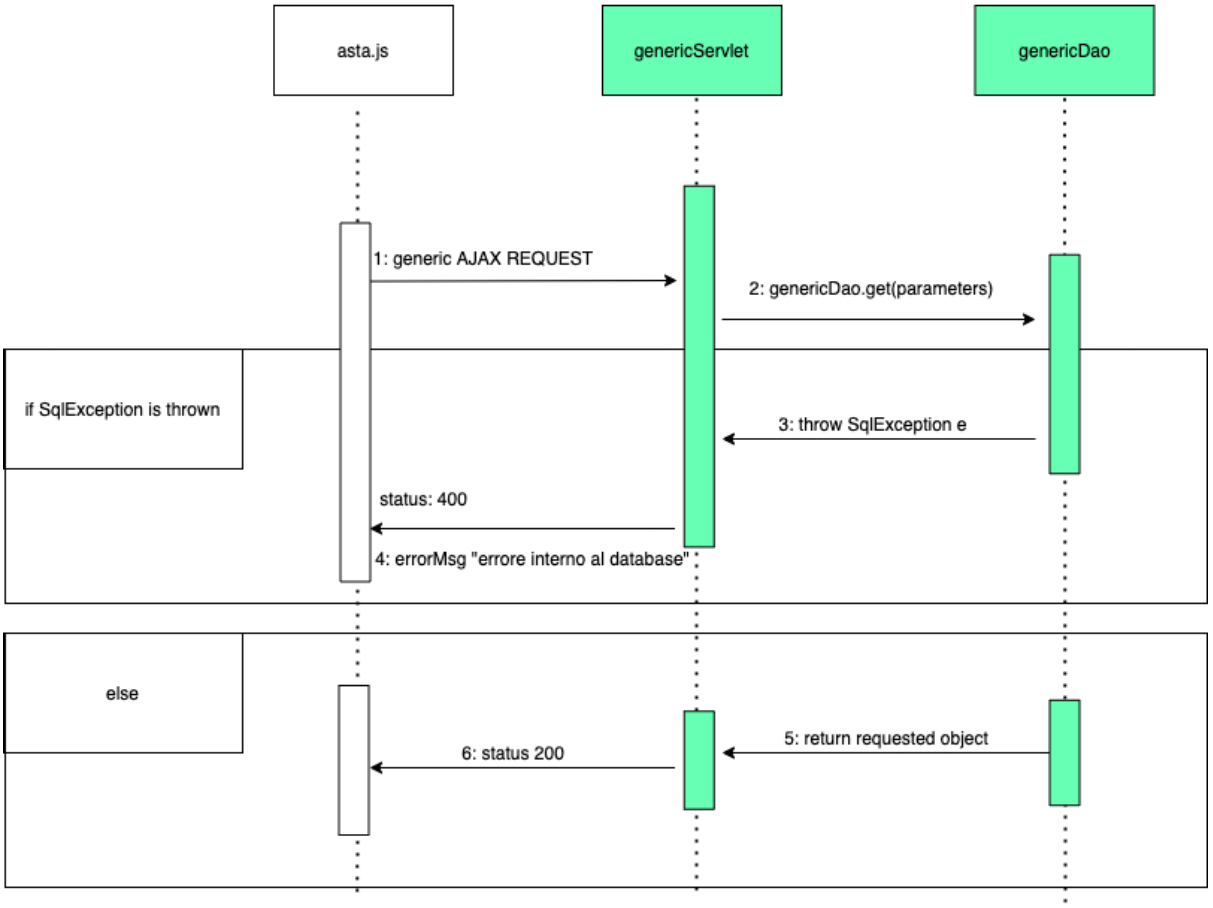

AstaByKeywordForm
GetAstaApertaDetails
astaApertaDao
if session.utente != null & !session.isNew()
1: AJAX GET GetAstaApertaDetails (asta_id)
2: astaApertaDao.getAstaApertaById
3: astaAperta || null
if null
errorMsg : "non ci sono aste per questo id"
status : 400
status : 200
return astaAperta
4: AstaApertaDetails.update(asta)


listaOfferteAstaAperta
GetAstaApertaOfferts
offertaDao
if session.utente != null & !session.isNew()
1: AJAX GET GetAstaApertaOfferts (asta_id)
2: offertaDao.getAllOfferteByAstaId(asta_id)
NumberFormatEx || nullPointerEx
status : 400
errorMsg : "parametri passati non validi"
status : 200
asta = astaAperta
4: ListaOfferteAstaAperta.update(offerteAsteToShow)

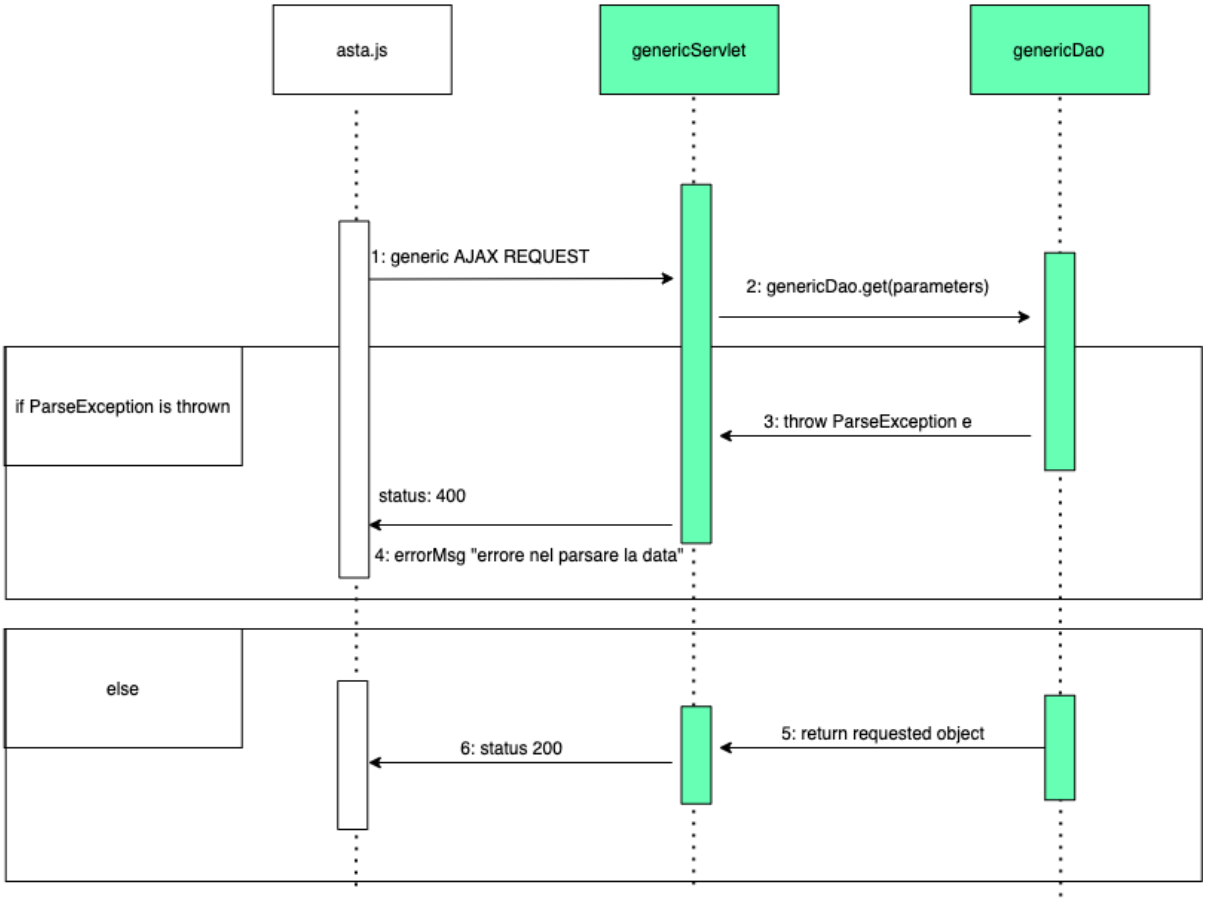
session checker



Handling SqlException



Handling parse exception



Strategic choices:

- "hidden" was not used to hide the invisible forms, but ".display = block / none" was used, to avoid spaces on the page
- all data entered in the forms are also checked on the server side, to avoid errors
- “closeAsta" query, which contains two database operations, has been made atomic
- “creaOfferta” calls offertaDao.creaOfferta() which contains two database operations back-to-back, has been made atomic.
- Cookies are stored with key = username and value = list of ids of clicked auctions or, if the user pressed “vendo”, the string “vendo”.
- For simplicity we have decided to store images in the database as base64 encoded Strings, this became useful because we could pass directly the string to the html tag without any conversion back to png from base64.
- The main Html page is divided in three blocks (id_acquisto and id_vendita,id_offerta), this way was easier to navigate the dom tree while writing the javascript client side.