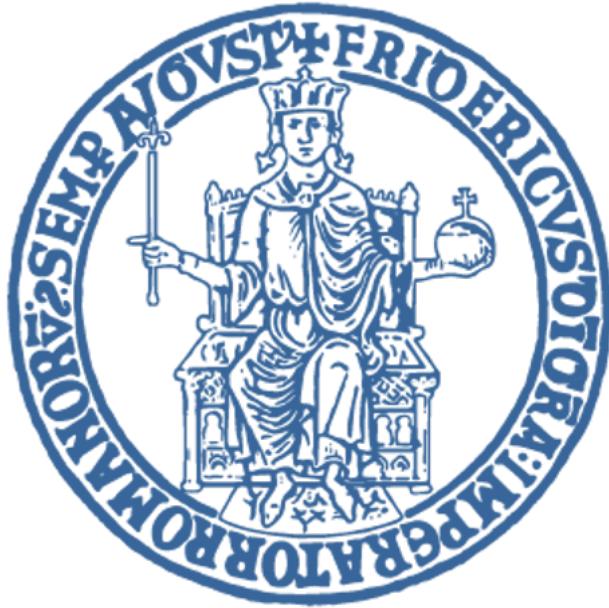


# DietiDeals24

Corso di laurea in Informatica  
Progetto di Ingegneria del software 2023/24

Giulio Borriello N86003381  
Emanuele Donnarumma N86003617  
Jonathan Borrelli N86004049

Aprile 2024



UNIVERSITÀ DEGLI STUDI DI NAPOLI  
**FEDERICO II**

# Indice

<b>1 Introduzione.</b>	<b>3</b>
<b>2 Requisiti software.</b>	<b>3</b>
2.1 Analisi dei requisiti . . . . .	3
2.1.1 Modellazione dei casi d'uso . . . . .	4
2.1.2 Individuazione del target degli utenti . . . . .	4
2.1.3 Descrizioni testuali strutturate . . . . .	8
2.1.4 Prototipazione visuale dell'interfaccia utente . . . . .	15
2.1.5 Valutazione dell'usabilità priori . . . . .	24
2.1.6 Glossario . . . . .	31
2.2 Specifica dei requisiti . . . . .	32
2.2.1 Class Diagram di analisi . . . . .	33
2.2.2 Sequence diagram di analisi . . . . .	47
2.2.3 Statechart funzionali . . . . .	50
<b>3 Design del sistema.</b>	<b>52</b>
3.1 Architettura del sistema e scelte tecnologiche adottate . . . . .	52
3.1.1 Client . . . . .	53
3.1.2 Server . . . . .	53
3.1.3 Comunicazione Client-Server . . . . .	55
3.1.4 Fase di deployment . . . . .	55
3.2 Class Diagram di design . . . . .	55
3.2.1 Backend . . . . .	56
3.2.2 Frontend . . . . .	65
3.3 Sequence diagram di design . . . . .	82
<b>4 Codice sorgente sviluppato.</b>	<b>87</b>
4.1 File di build automatica . . . . .	87
4.2 Versioning . . . . .	88
4.3 Code quality . . . . .	88
<b>5 Testing e valutazione sul campo dell'usabilità.</b>	<b>89</b>
5.1 JUnit testing . . . . .	89
5.1.1 Classi di equivalenza individuate . . . . .	89
5.1.2 Codice JUnit - getSilentAuctionsByTitleContainingAndCategory . . . . .	91
5.1.3 Codice JUnit - doesAccountExist . . . . .	93
5.1.4 Codice JUnit - updateRegion . . . . .	94
5.1.5 Codice JUnit - updateBio . . . . .	95
5.1.6 Esito dei test . . . . .	96
5.2 Valutazione dell'usabilità sul campo . . . . .	97
5.2.1 Test sul prodotto finito mediante testers umani . . . . .	97
5.2.2 Analisi dei file di log di Google Analytics . . . . .	98

## 1 Introduzione.

DietiDeals24 è una piattaforma per la creazione e gestione di aste online. Il sistema consiste in un'applicazione attraverso cui un utente può fruire delle seguenti funzionalità:

- Registrare un nuovo account di tipo [compratore](#), ed utilizzarlo per accedere al sistema. Successivamente l'utente potrà sbloccare la modalità [venditore](#). Suddetti account possono essere personalizzati aggiungendo dati personali, come una breve [bio](#), link ai propri siti web e/o social network e [regione](#).
- Presentare offerte alle aste degli altri utenti.
- Cercare e visualizzare i dettagli delle aste altrui mediante parole chiave e/o categorie. Inoltre, si possono anche visualizzare i dati dell'utente proprietario dell'asta.
- Creare [aste silenziose](#).
- Creare [aste al ribasso](#).
- Creare [aste inverse](#).

Il suddetto applicativo, oltre ai requisiti funzionali precedentemente elencati, dovrà tenere conto anche dei requisiti non funzionali, seguendo le linee guida del modello **FURPS**:

- **Functionality**: capacità di fornire le funzioni richieste.
- **Usability**: facilità d'uso dell'utenza finale.
- **Reliability**: gestione degli errori e dei crash.
- **Performance**: prestazioni dell'applicazione.
- **Supportability**: capacità di garantire assistenza e mantenimento.

In questo documento andremo ad analizzare tutto il ciclo di vita del software, caratterizzato da:

- Analisi e specifica dei requisiti software.
- Sviluppo del design di sistema.
- Testing.

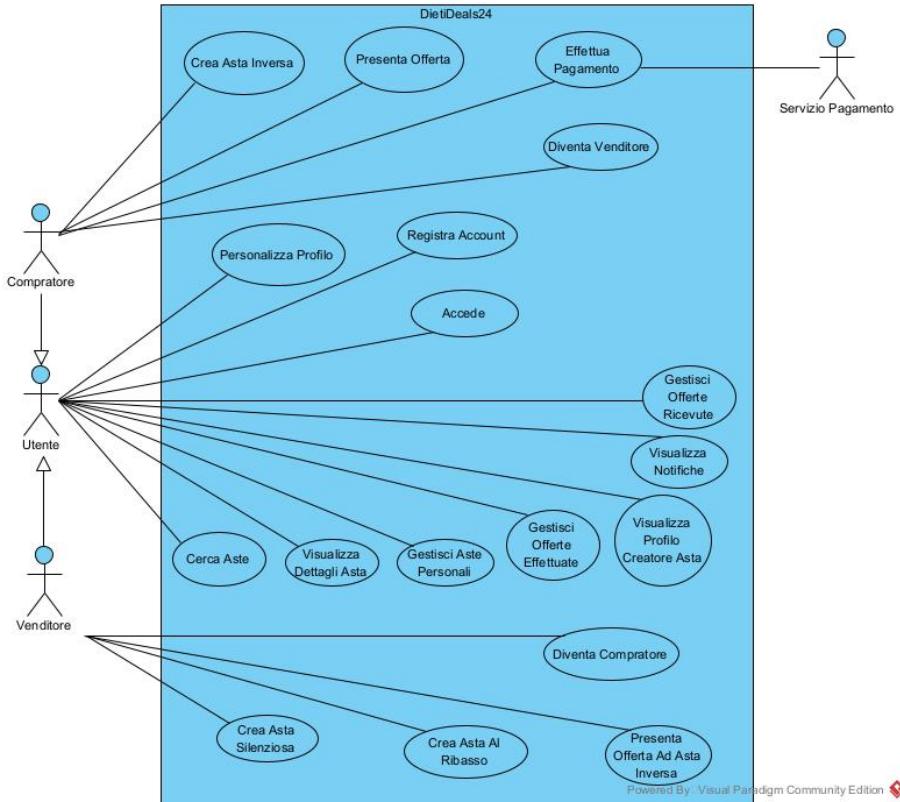
## 2 Requisiti software.

### 2.1 Analisi dei requisiti.

Di seguito analizzeremo i requisiti software descritti antecedentemente. Per la suddetta analisi abbiamo utilizzato lo strumento di modellazione UML [Visual Paradigm](#).

### 2.1.1 Modellazione dei casi d'uso.

Di seguito riportiamo la modellazione dei **casi d'uso** rappresentati mediante **Use Case Diagram**.



### 2.1.2 Individuazione del target degli utenti.

Il **target degli utenti** di DietiDeals24 da noi individuato è il seguente:

- Collezionisti.
- Commercianti.
- Utenti casual in cerca di buoni affari.

Suddetto target è rappresentato tramite i **Personas**, ovvero profili fintizi che identificano l'utente medio di quel target. Lo strumento utilizzato per generare i già menzionati **Personas** è **Xtensio**.

# Alessandro Valenti



*"Ho bisogno di un'app per ampliare la mia collezione"*

Età: 50-70

Lavoro: Business Man

Famiglia: Sposato con figli

Città: Milano, Italia

Carattere: Appassionato

## Personalità



## Obiettivi

- Comprare beni di collezionismo a prezzi convenienti
- Avere una piattaforma apposita per farlo.
- Completare le sue collezioni

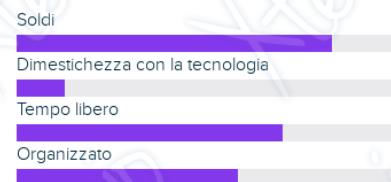
## Frustrazioni

- Non pratico con i dispositivi elettronici
- Nei negozi fisici non trova sempre quello che cerca
- Non conosce molti siti di aste

## Bio

Alessandro Valenti, nato e cresciuto in un'atmosfera intrisa d'arte e cultura, ha coltivato fin da giovane una passione profonda per la collezione di oggetti unici e significativi. Alessandro ha sviluppato nel tempo una collezione eclettica che spazia dalle opere d'arte più raffinate ai tesori nascosti di piccole comunità. Il suo patrimonio economico gli consente di partecipare a qualsiasi asta, non badando a spese.

## Attributi



## Motivazioni



# Gabriele Mercante



*"Voglio espandere il mio mercato su un altro canale"*

Età: 30-55

Lavoro: Proprietario di negozio

Famiglia: Padre di famiglia

Città: Napoli, Italia

Carattere: Lungimirante

## Personalità



## Obiettivi

- Vendere la merce invenduta nel negozio
- Pubblicizzare il suo negozio
- Comprare servizi relativi al suo negozio

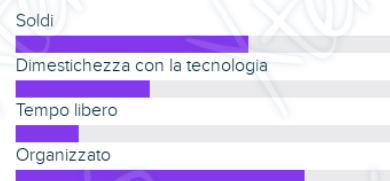
## Frustrazioni

- Non riesce a vendere parte della merce in magazzino
- Non conosce app per compravendita sia di beni che di servizi

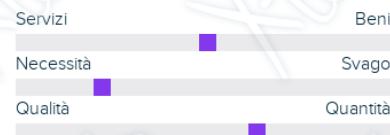
## Bio

Gabriele Mercante, l'anima calorosa e imprenditoriale di un accogliente negoziotto nel cuore della città, è una presenza familiare e amata nella comunità. Tuttavia, consapevole dell'importanza di adattarsi ai cambiamenti e alle nuove opportunità, Gabriele ha deciso di espandere il mercato del suo negozio, raggiungendo un pubblico più vasto e sfruttando al meglio ogni risorsa disponibile. Gabriele non vede l'applicazione solo come un mezzo per vendere la merce invenduta nel suo negozio, ma anche come una piattaforma per acquistare servizi sporadici cruciali per il suo business. Dalla ristrutturazione del suo spazio commerciale alla creazione di un sito online che riflette l'autenticità del suo negozio, Gabriele è pronto a investire in opportunità che porteranno il suo negozio ad un nuovo livello di successo.

## Attributi



## Motivazioni



# Marco Rando



*"Ho bisogno di comprare cose a buon prezzo e di vendere la mia roba usata"*

Età: **16-26**

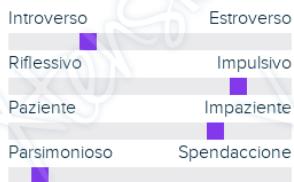
Lavoro: **Studente**

Famiglia: **Single con genitori**

Città: **Roma, Italia**

Carattere: **Intraprendente**

## Personalità



## Obiettivi

- Risparmiare
- Vendere la sua roba inutilizzata
- Cercare di fare lavori per guadagnare qualche soldo

## Frustrazioni

- Non si può permettere le cose nei negozi tradizionali
- Non è bravo a contrattare dal vivo

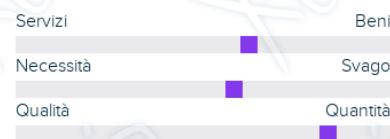
## Bio

Marco Rando incarna lo spirito moderno di chi cerca opportunità intelligenti. Attualmente studente, Marco è sempre alla ricerca di affari vantaggiosi per soddisfare le sue esigenze di studente e le sue passioni personali e trova vantaggiose l'asta inversa e l'asta al ribasso perché gli permettono di arrivare a trovare oggetti a prezzi più bassi rispetto a quelli di listino. Ma Marco non è solo un acquirente astuto. Con uno sguardo critico sulla sua stanza, ha riconosciuto il potenziale di liberarsi di oggetti inutilizzati. Così, si impegna anche nel lato venditore dell'app, dando nuova vita ai suoi articoli usati e guadagnando qualche soldo extra nel processo.

## Attributi



## Motivazioni



### **2.1.3 Descrizioni testuali strutturate.**

Ecco due [casi d'uso](#) che ci sono stati assegnati, descritti tramite il cosiddetto [template di Cockburn](#).

1. Personalizzazione del profilo.

<b>USE CASE #1</b>	<i>Personalizza profilo</i>		
<b>Goal in Context</b>	Modificare i dati del profilo		
<b>Preconditions</b>	L'utente ha effettuato l'accesso		
<b>Success End Condition</b>	Le modifiche sono state apportate con successo		
<b>Primary Actor</b>	Utente (o esclusivamente Venditore nel caso modifichi l'IBAN)		
<b>Trigger</b>	L'utente ha cliccato sull'icona del profilo		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>
	1	Preme il tasto "Modifica profilo"	
	2		Mostra la schermata "Modifica profilo"
	3	Preme il tasto per tornare indietro	
<b>EXTENSION A</b> L'utente inserisce un link in un formato invalido	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>
	7.d.a	Inserisce un URL in un formato non valido	
	8.d.a		<i>Mostra il messaggio di errore: L'URL inserito non è valido</i>
	9.d.a	Va allo step 7.d	
<b>EXTENSION B</b> L'utente inserisce un IBAN in un formato invalido	<b>Step n°</b>	<b>Venditore</b>	<b>Sistema</b>
	5.c.b	Inserisce un IBAN in un formato non valido	
	6.c.b		<i>Mostra il messaggio di errore: L'IBAN inserito non è valido</i>
	7.c.b	Va allo step 5.c	
<b>SUBVARIATION A</b> "Modifica la	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>
	3.a	Preme il tasto "Modifica"	

regione"		regione"	
	4.a		<i>Mostra la schermata "Modifica regione"</i>
	5.a	Seleziona la nuova regione per il profilo	
	6.a	Preme il tasto per tornare indietro	
	7.a		<i>Ritorna allo step 2</i>
SUBVARIATION B "Modifica la bio"	Step n°	Utente	Sistema
	3.b	Preme il tasto "Modifica bio"	
	4.b		<i>Mostra la schermata "Modifica bio"</i>
	5.b	Scrive la nuova bio per il profilo	
	6.b	Preme il tasto per confermare	
	7.b		<i>Torna allo step 2</i>
	Step n°	Venditore	Sistema
SUBVARIATION C "Modifica dati bancari"	3.c	Preme il tasto "Modifica IBAN"	
	4.c		<i>Mostra la schermata "Modifica conto bancario"</i>
	5.c	Inserisce il nuovo IBAN	
	6.c	Inserisce la nuova partita IVA	
	7.c	Preme il tasto per confermare	
	8.c		<i>Mostra la schermata col messaggio "IBAN modificato con successo"</i>
	9.c	Preme il tasto "Ok"	
	10.c		<i>Torna allo step 2</i>
SUBVARIATION D	Step n°	Utente	Sistema

<b>'Modifica i link esterni'</b>	3.d	Preme il tasto "Modifica link esterni"	
	4.d		Mostra la schermata "Modifica link"
	5.d	Preme il tasto "Aggiungi link esterno"	
	6.d		Mostra la schermata "Aggiungi link esterno"
	7.d	Inserisce l'URL	
	8.d	Inserisce il titolo	
	9.d	Preme il tasto per confermare	
	10.d		Mostra la schermata col messaggio "Link esterno aggiunto con successo"
	11.d	Preme il tasto "Ok"	
	12.d		Mostra la schermata "Modifica link"
	13.d	Preme il tasto per tornare indietro	
	14.d		Mostra la schermata "Modifica profilo"
	15.d	Preme il tasto per tornare indietro	
	16.d		Torna allo step 2
<b>SUBVARIATION D.1</b> <b>"Modifica un link esistente"</b>	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>
	5.d.1	Preme il tasto di un link esistente	
	6.d.1		Mostra la schermata "Modifica link esterno"
	7.d.1	Modifica l'URL	
	8.d.1	Modifica il titolo	
	9.d.1	Preme il tasto per confermare	

	10.d.1		Mostra la schermata "Modifica link"
	11.d.1	Preme il tasto per tornare indietro	
	12.d.1		Mostra la schermata "Modifica profilo"
	13.d.1	Preme il tasto per tornare indietro	
	14.d.1		Torna allo step 2
<b>SUBVARIATION D.2</b>  "Elimina un link esistente"	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>
	5.d.2	Preme il tasto di un link esistente	
	6.d.2		Mostra la schermata "Modifica link esterno"
	7.d.2	Preme il tasto "Rimuovi link"	
	8.d.2		Mostra la schermata "Modifica link"
	9.d.2	Preme il tasto per tornare indietro	
	10.d.2		Mostra la schermata "Modifica profilo"
	11.d.2	Preme il tasto per tornare indietro	
	12.d.2		Torna allo step 2

2. Creazione di un'asta silenziosa.

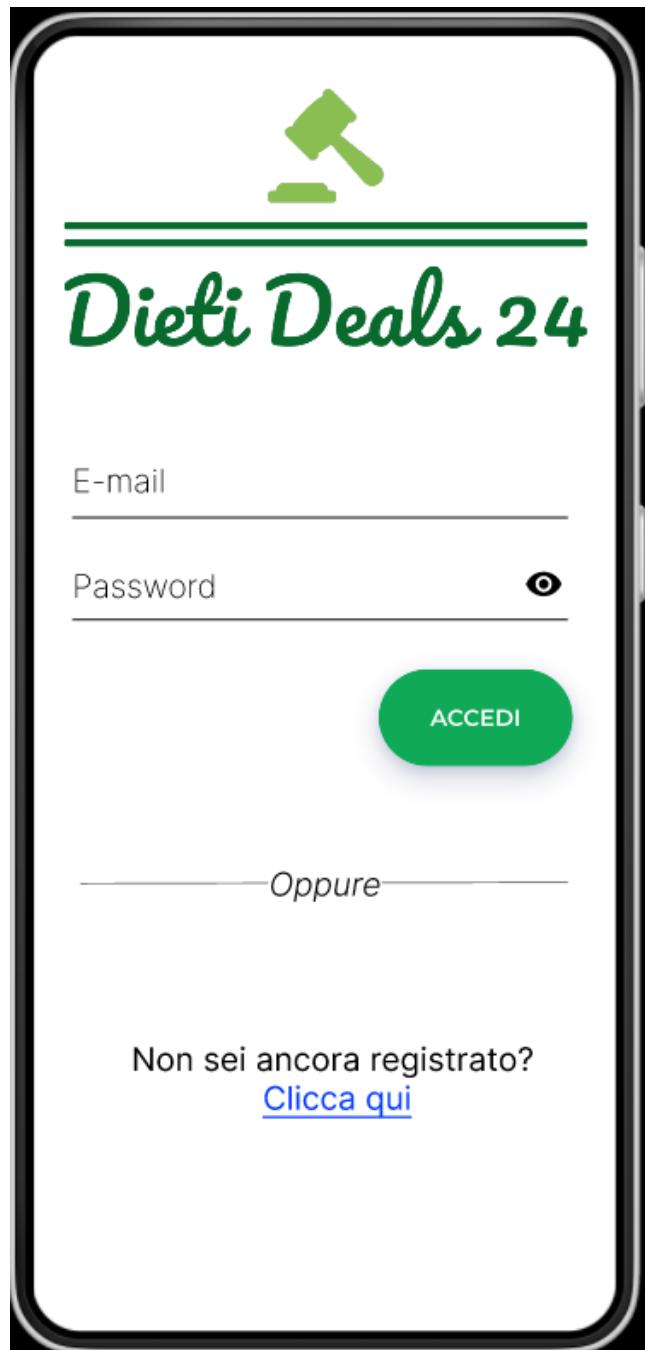
<b>USE CASE #2</b>	<i>Crea asta silenziosa</i>		
<b>Goal in Context</b>	Creare un'asta silenziosa		
<b>Preconditions</b>	L'utente ha effettuato l'accesso con un account da venditore		
<b>Success End Condition</b>	L'utente crea una nuova asta silenziosa con successo		
<b>Primary Actor</b>	Venditore		
<b>Trigger</b>	Il venditore ha cliccato sull'icona “+”		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Venditore</b>	<b>Sistema</b>
	1	Inserisce il titolo dell'asta	
	2	Inserisce la descrizione dell'asta	
	3	Seleziona il grado di usura	
	4	Seleziona la categoria	
	5	Preme il tasto per andare avanti	
	6		<i>Mostra la schermata “Scegli tipo asta (venditore)”</i>
	7	Preme il tasto “Silenziosa”	
	8		<i>Mostra la schermata “Crea asta silenziosa”</i>
	9	Seleziona la data di scadenza dell'asta	
	10	Seleziona il tempo massimo per ritirare l'offerta	
	11	Preme il tasto di conferma	
	12		<i>Mostra popup con il messaggio “Asta creata con successo”</i>
	13	Preme il tasto “Ok”	
	14		<i>Mostra la schermata “Aste”</i>

		<i>personali"</i>
SUBVARIATION A "Aggiunge una foto all'asta"	Step n°	Venditore
	5.a	Preme il tasto per aggiungere un'immagine
	6.a	
	7.a	Selezione una delle due modalità
	8.a	Inserisce l'immagine
	9.a	
	10.a	Va allo step 5
		<i>Mostra un popup dove chiede al venditore se scegliere un'immagine dalla galleria o scattare una foto</i>

#### **2.1.4 Prototipazione visuale dell'interfaccia utente.**

La prototipazione visuale è stata effettuata tramite lo strumento [Figma](#). Per visualizzare il [Mockup](#) nella sua interezza basta visitare il seguente [link](#). Di seguito mostriamo alcune delle schermate principali della nostra applicazione.

1. Schermata di login.



2. Schermata di ricerca delle aste.



3. Schermata di visualizzazione delle aste personali.



4. Schermata di visualizzazione delle proprie offerte.



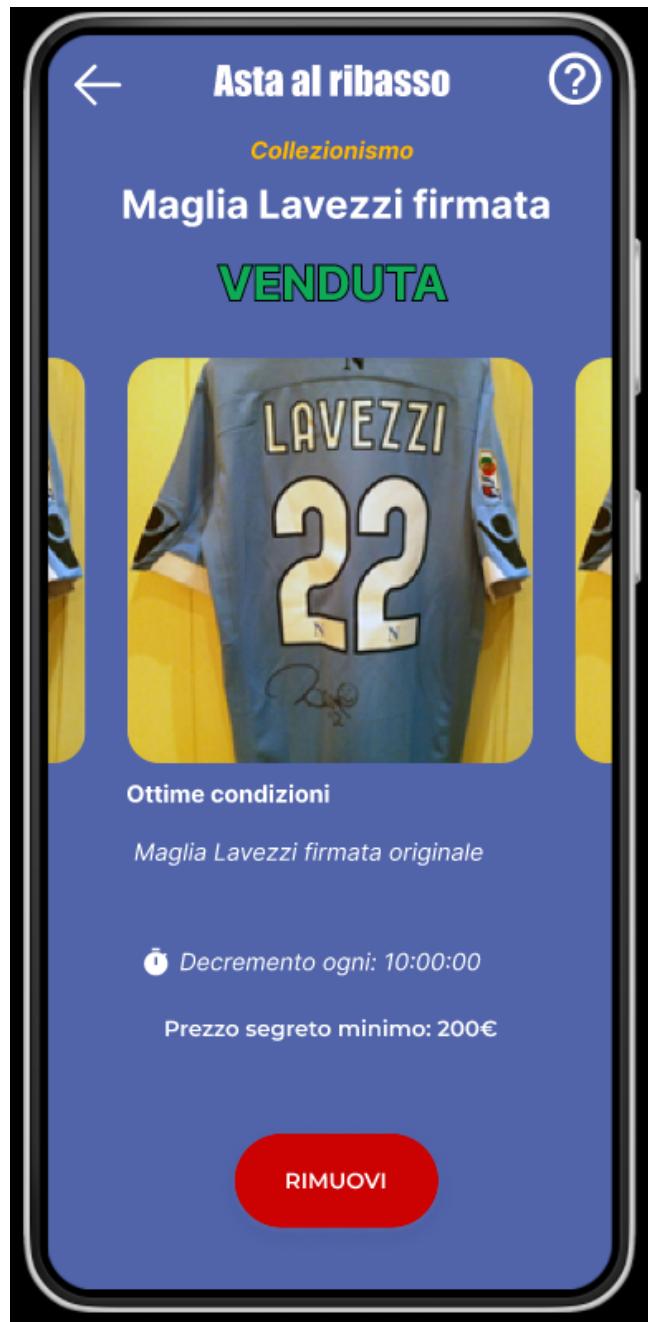
5. Schermata di visualizzazione del proprio profilo.



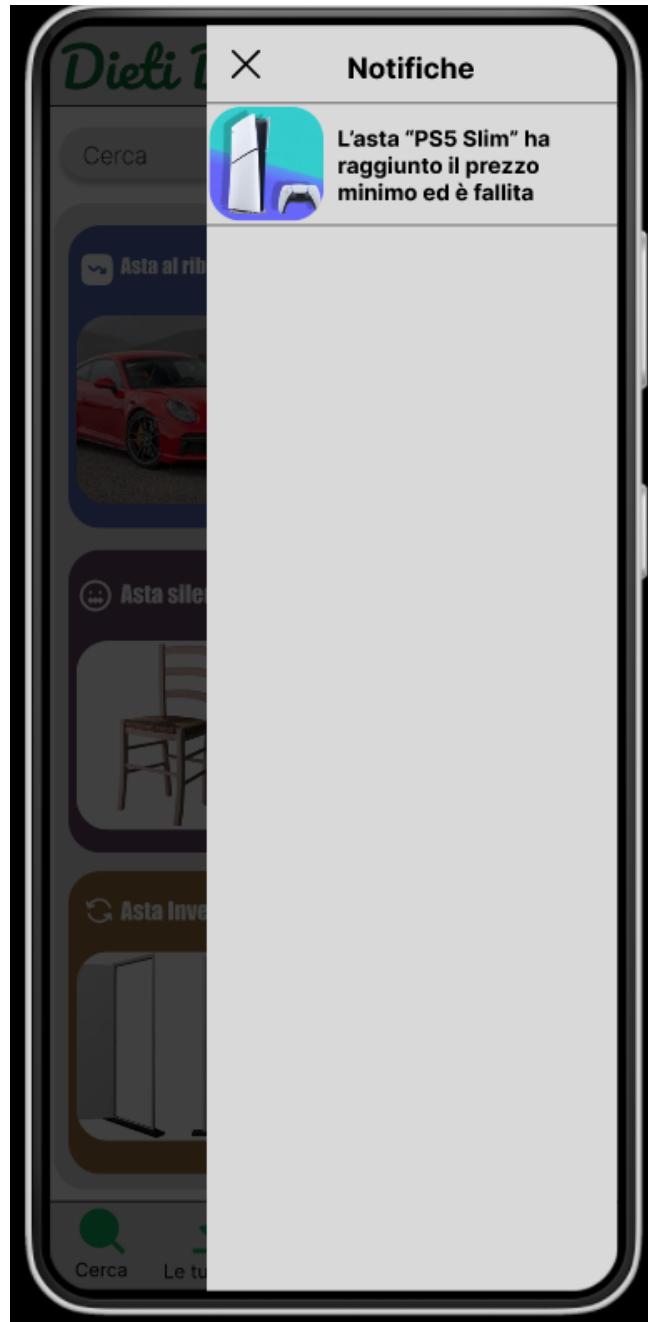
6. Una delle schermate di creazione delle aste.



7. Schermata di visualizzazione dei dettagli dell'asta.



8. Tendina di visualizzazione delle notifiche.



### 2.1.5 Valutazione dell'usabilità priori.

**Usabilità prima delle modifiche.** Per poter assegnare all'**usabilità** un valore numerico, abbiamo selezionato 4 persone in fasce demografiche diverse e gli abbiamo assegnato 10 compiti diversi. La tabella riporta i risultati riscontrati.

**S:** Successo = 1

**P:** Parziale Successo = 0.5

**F:** Fallimento = 0

	TASK 1	TASK 2	TASK 3	TASK 4	TASK 5	TASK 6	TASK 7	TASK 8	TASK 9	TASK 10
TESTER 1 59 anni	S	P	S	P	S	P	S	P	P	P
TESTER 2 62 anni	P	S	S	P	P	P	S	P	S	S
TESTER 3 21 anni	S	S	P	S	S	S	S	S	S	S
TESTER 4 20 anni	S	S	P	S	S	S	S	S	S	S

#### TASKS:

1. Registrati come un venditore
2. Crea un'asta al ribasso di un bene
3. Accetta un'offerta a una tua asta silenziosa
4. Effettua il pagamento di una tua offerta
5. Modifica il profilo aggiungendo un link esterno
6. Visualizza le offerte effettuate
7. Visualizza notifiche
8. Compra un'asta al ribasso
9. Fai un'offerta ad un'asta silenziosa
10. Visualizza il profilo di un creatore di un'asta

#### TESTERS:

1. Maurizio
2. Giuseppe
3. Lorenzo
4. Tester Anonimo

$$\text{Usabilità} = \frac{27 + (0.5 \times 13)}{40} = 0,8375 = 83,75\% \text{ di usabilità}$$

**Usabilità dopo le modifiche.** Dopo aver ascoltato i feedback ricevuti dai tester, abbiamo apportato alcune modifiche alle interfacce grafiche e successivamente abbiamo fatto testare nuovamente i 10 compiti ad altre persone, avendo cura che appartenessero alle stesse fasce demografiche dei precedenti.

**S:** Successo = 1

**P:** Parziale Successo = 0.5

**F:** Fallimento = 0

	TASK 1	TASK 2	TASK 3	TASK 4	TASK 5	TASK 6	TASK 7	TASK 8	TASK 9	TASK 10
TESTE R 1  60 anni	S	S	S	S	S	S	S	S	S	S
TESTE R 2  56 anni	S	S	P	P	P	P	P	F	S	P
TESTE R 3  22 anni	S	S	S	S	S	S	S	S	S	S
TESTE R 4  22 anni	S	S	S	S	S	S	S	S	S	S

**TASKS:**

1. Registrati come un venditore
2. Crea un'asta al ribasso di un bene
3. Accetta un'offerta a una tua asta silenziosa
4. Effettua il pagamento di una tua offerta
5. Modifica il profilo aggiungendo un link esterno
6. Visualizza le offerte effettuate
7. Visualizza notifiche
8. Compra un'asta al ribasso
9. Fai un'offerta ad un'asta silenziosa
10. Visualizza il profilo di un creatore di un'asta

**TESTERS:**

1. Giorgio
2. Emma
3. Alessandro
4. Chiara

$$\text{Usabilità} = \frac{33 + (0.5 \times 6)}{40} = 0.9 = 90\% \text{ di usabilità}$$

## Modifiche significative effettuate

### 1. Modifiche del menù

- Cambiata l'icona delle mie offerte per darle maggiore intuitività.
- Aggiunte le scritte alle varie opzioni del menù, per rendere più immediato il significato di ogni icona.
- Aggiunto un cerchio attorno al tasto per creare un'asta, per renderlo più visibile a colpo d'occhio.
- L'icona corrispondente alla scherma che l'utente sta visualizzando viene evidenziata, per aiutarlo nella navigazione.
- L'icona della schermata di ricerca è stata cambiata in una lente di ingrandimento.



Figura 1: Menù di navigazione prima delle modifiche.



Figura 2: Menù di navigazione dopo le modifiche.

2. Modifica del logo.

- Modificato il font e il colore del logo dell'applicazione, per renderlo più coerente con gli altri font e colori utilizzati

*Dieti Deals 24*

Figura 3: Font e colore del logo prima della modifica.

**Dieti Deals 24**

Figura 4: Font e colore del logo dopo la modifica.

3. Aggiunta di un evidenziatore dei passaggi di registrazione.

- Aggiunto un componente grafico chiamato stepper alle schermate di registrazione, per rendere più visibili i passaggi di registrazione all'utente.

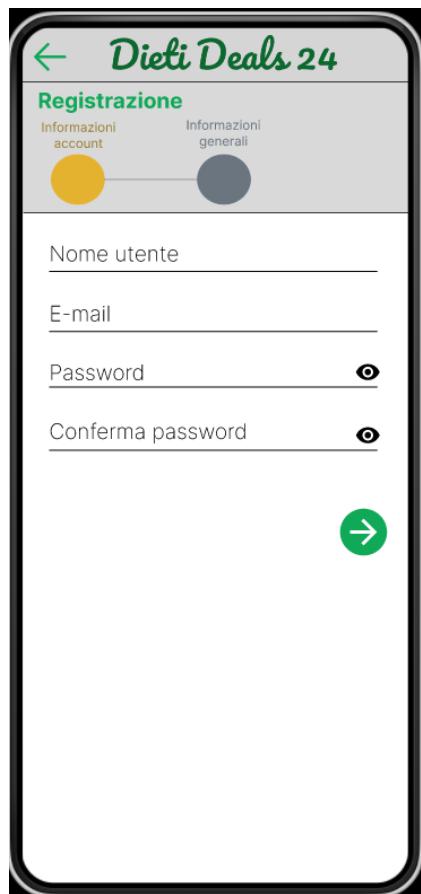


Figura 5: Prima schermata di registrazione.



Figura 6: Seconda schermata di registrazione.

4. Modificati i colori.

- Modificati alcuni colori per rendere più visibili le scritte all'interno dell'applicazione.

6. Modificate le schermate di pagamento.

- Aggiunta una schermata di selezione del metodo di pagamento, nonostante al momento l'utente possa scegliere solo il pagamento con carta di credito.



Figura 7: Schermata di selezione del metodo di pagamento.

- Aggiunta una schermata di riepilogo al momento del pagamento.

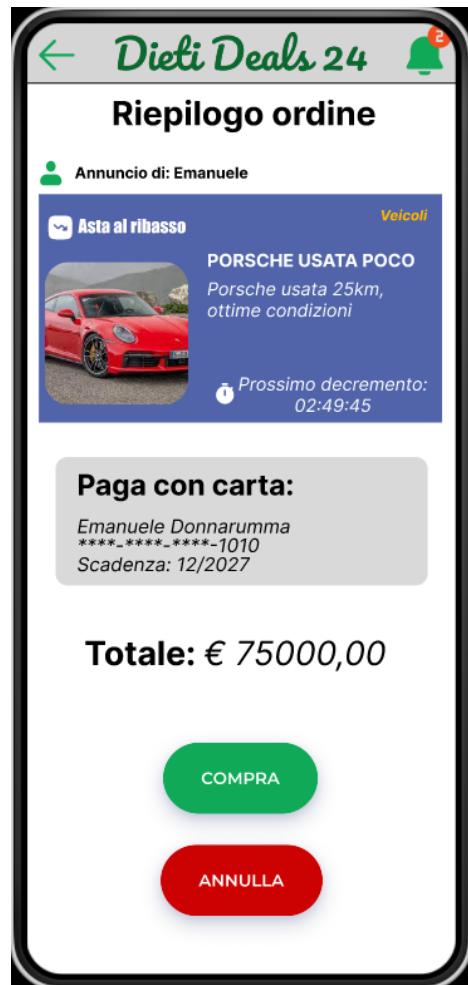


Figura 8: Schermata di riepilogo del pagamento.

## 2.1.6 Glossario.

<b>Asta inversa</b>	In questo tipo di asta, il compratore specifica il prodotto/servizio richiesto, eventualmente inserendo un'immagine dello stesso, un prezzo iniziale che è disposto a pagare, e una data di scadenza. I venditori in grado di fornire quel particolare prodotto/servizio possono quindi partecipare all'asta competendo abbassando il prezzo. In particolare, fino al momento della scadenza dell'asta, i venditori possono presentare offerte al ribasso. Al momento della scadenza dell'asta, il venditore con l'offerta più bassa si aggiudica la fornitura del prodotto/servizio.
<b>Asta al ribasso</b>	Un'asta al ribasso è caratterizzata da un prezzo iniziale elevato specificato dal venditore, da un timer per il decremento del prezzo (di default 1 ora), da un importo, in €, per ciascun decremento, e da un prezzo minimo (segreto) a cui vendere il prodotto/servizio. Il prodotto/servizio sarà in vendita al prezzo iniziale stabilito dal venditore. Al raggiungimento del timer, il prezzo verrà decrementato dell'importo previsto. Il primo compratore a presentare un'offerta si aggiudica il prodotto/servizio. Se il prezzo viene decrementato fino a raggiungere il prezzo minimo segreto, l'asta viene considerata fallita e il venditore visualizza una notifica.
<b>Asta silenziosa</b>	In questo tipo di asta, il venditore specifica una data di scadenza e i compratori possono inviare offerte segrete al venditore. Il venditore può scegliere se accettare o rifiutare le offerte ricevute. Una sola offerta può essere accettata per ogni asta.
<b>Caso d'uso</b>	Un caso d'uso specifica un'operazione che l'utente può compiere interagendo con l'applicazione. Corrisponde ai requisiti funzionali del sistema.
<b>Target degli utenti</b>	Un insieme di persone a cui è destinata l'applicazione accomunate da delle caratteristiche comuni.
<b>Compratore</b>	Un utente che è in grado di fare offerte per aste silenziose e al ribasso e che è in grado di creare delle aste inverse.

<b>Venditore</b>	Un utente che è in grado di fare offerte per aste inverse e che è in grado di creare aste silenziose e al ribasso.
<b>Bio</b>	La descrizione del profilo di un utente.
<b>Regione</b>	Si intende una delle regioni italiane.
<b>Usabilità</b>	La capacità di un applicativo di essere autoesplicativo nelle sue funzionalità e quindi dare all'utente un'interfaccia intuitiva e semplice.
<b>Class Diagram</b>	Un diagramma UML che mostra le classi dell'applicativo e i collegamenti fra di esse.
<b>Sequence Diagram</b>	Un diagramma UML che mostra il percorso che esegue il sistema (le classi e i metodi in gioco) nel compiere uno specifico caso d'uso.
<b>Use Case Diagram</b>	Un diagramma UML che mostra tutti i casi d'uso dell'applicativo e gli utenti che interagiscono in essi.
<b>State Chart</b>	Un diagramma UML che mostra gli stati di un'entità (schermate, classi ecc.) e le transizioni che avvengono tra di loro.
<b>Template di Cockburn</b>	Un diagramma che mostra i passaggi che gli utenti devono compiere per portare a termine uno specifico caso d'uso e come il sistema risponde a questi passaggi.
<b>Personas</b>	Una descrizione di un utente fittizio creato a partire da caratteristiche ottenute tramite una media delle stesse in una determinata fascia di utenza.
<b>Mockup</b>	Prototipo dell'interfaccia utente.

## 2.2 Specifica dei requisiti.

Adesso descriviamo la specifica dei requisiti utilizzando i seguenti diagrammi UML:

- [Class Diagram](#).
- [Sequence Diagram](#).
- [Statechart Funzionali](#).

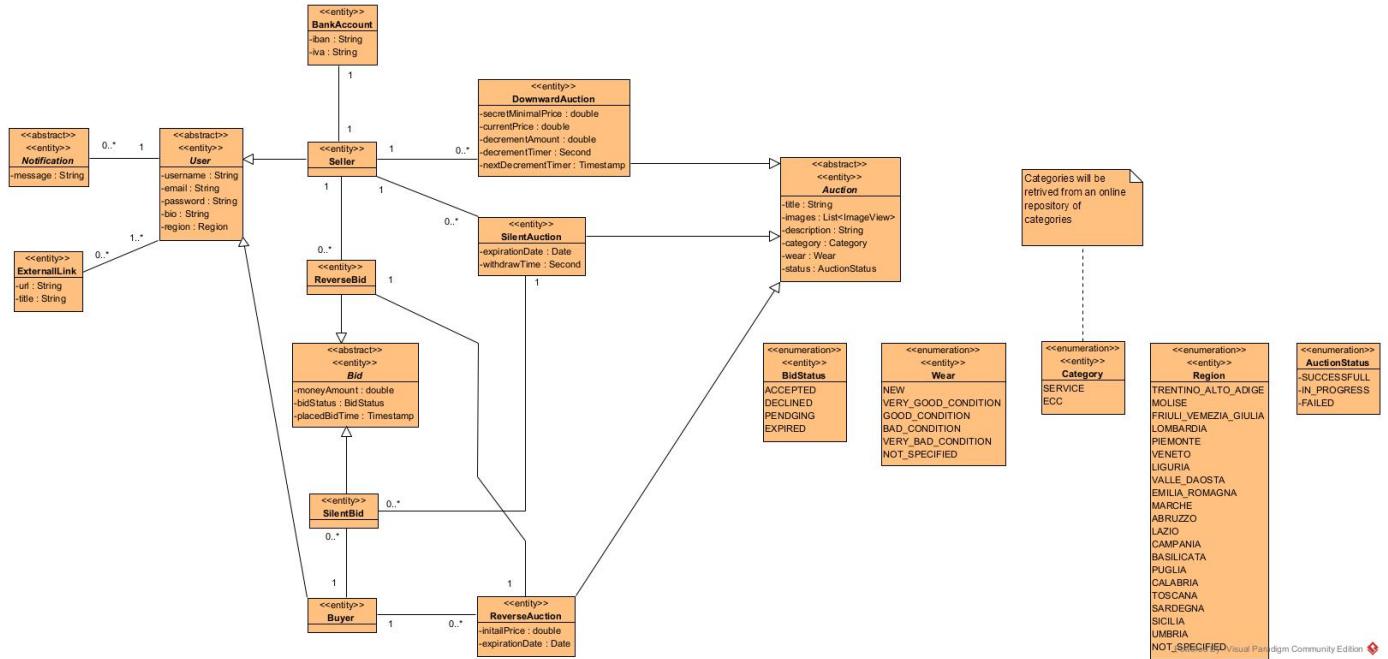
Per la fase di progettazione abbiamo sfruttato l'euristica **EBC**.

- **Entity**: Modella le informazioni.
- **Boundary**: Modella il modo in cui l'utente si interfaccia al sistema.
- **Control**: Modella la logica dell'applicativo e la comunicazione con il server.

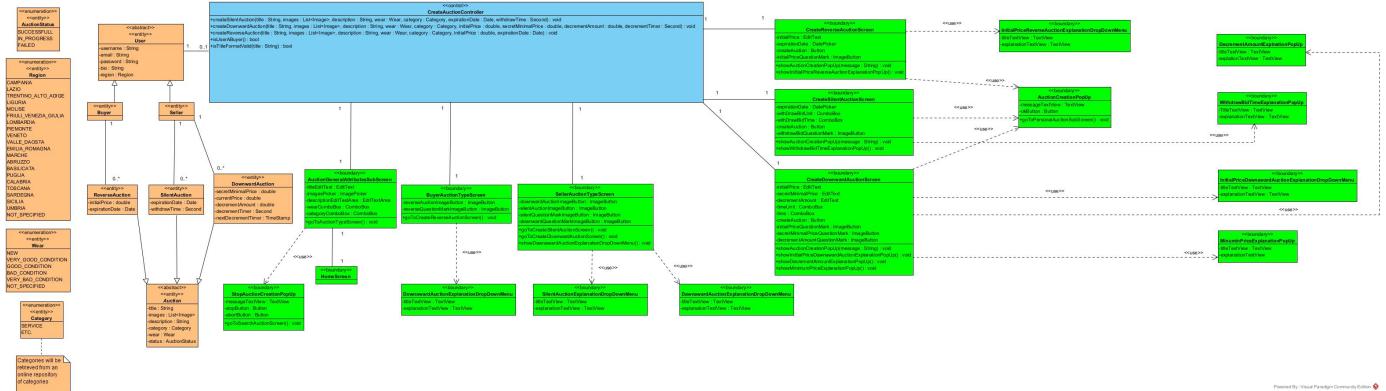
### 2.2.1 Class Diagram di analisi.

Di seguito mostriamo i **Class Diagram**. Per una maggiore leggibilità, sono state escluse alcune ridondanze, come metodi getter e setter e dipendenze ridondanti. Inoltre ogni diagramma mantiene coerenza cromatica, associando al colore arancione le classi Entity, al colore verde le classi Boundary ed al colore blu le classi Control.

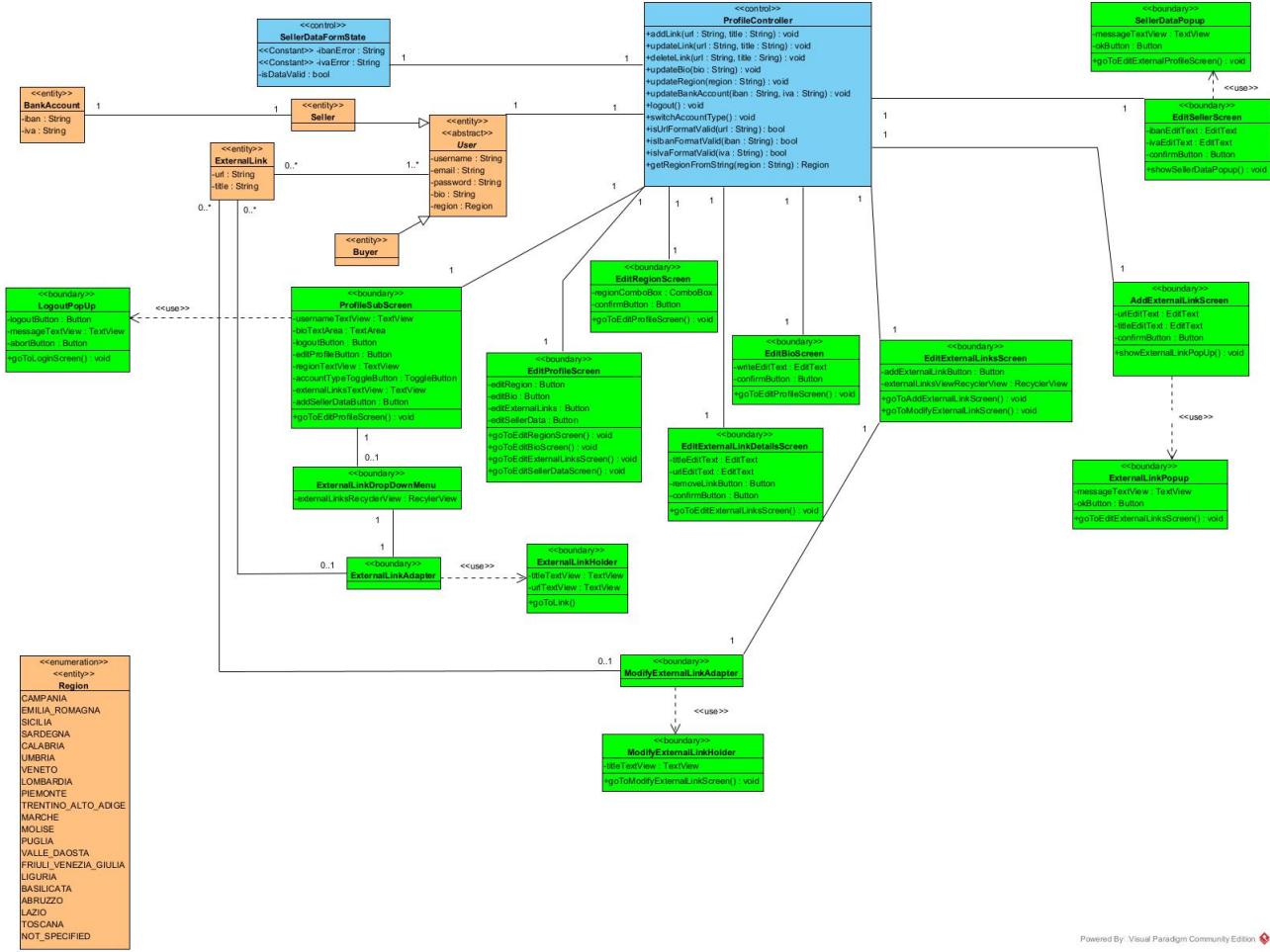
#### 1. Class Diagram delle entità.



## 2. Creazione di un'asta.

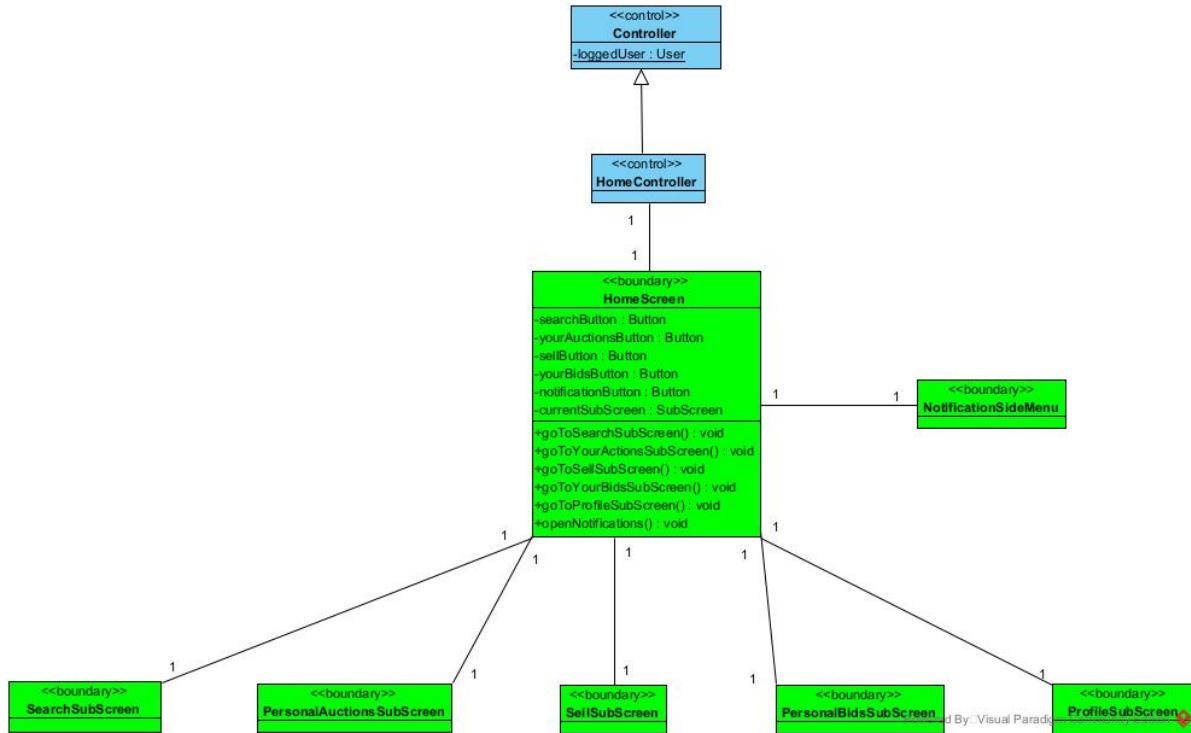


### 3. Modifica del proprio profilo.

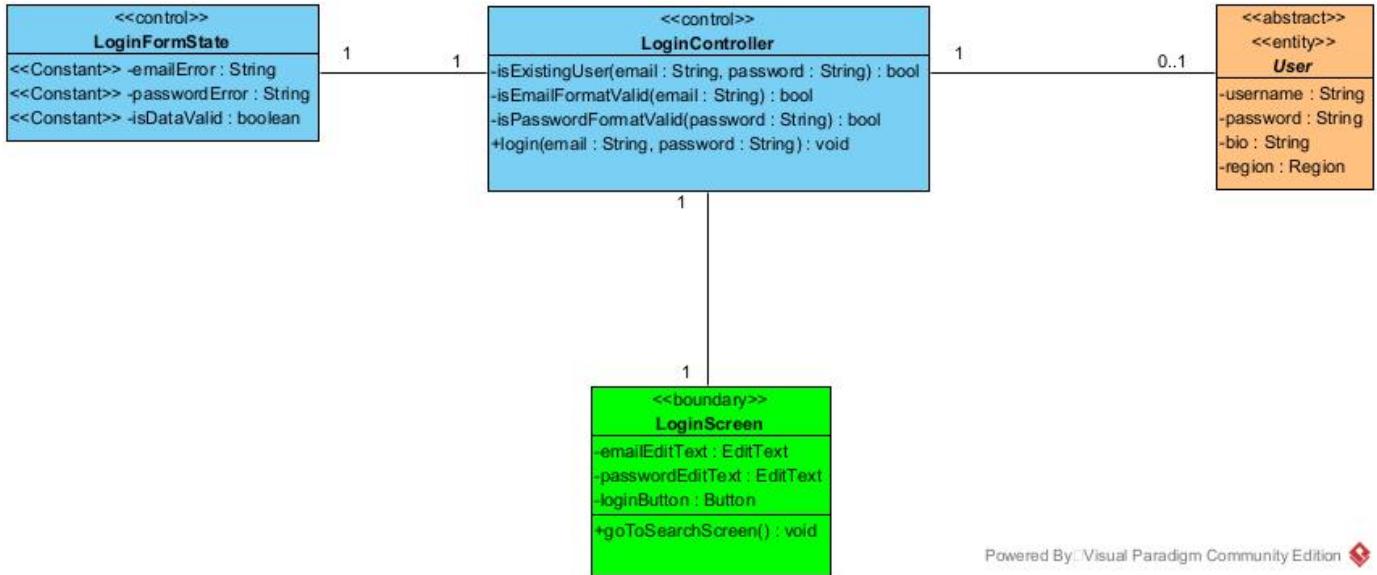


Powered By: Visual Paradigm Community Edition

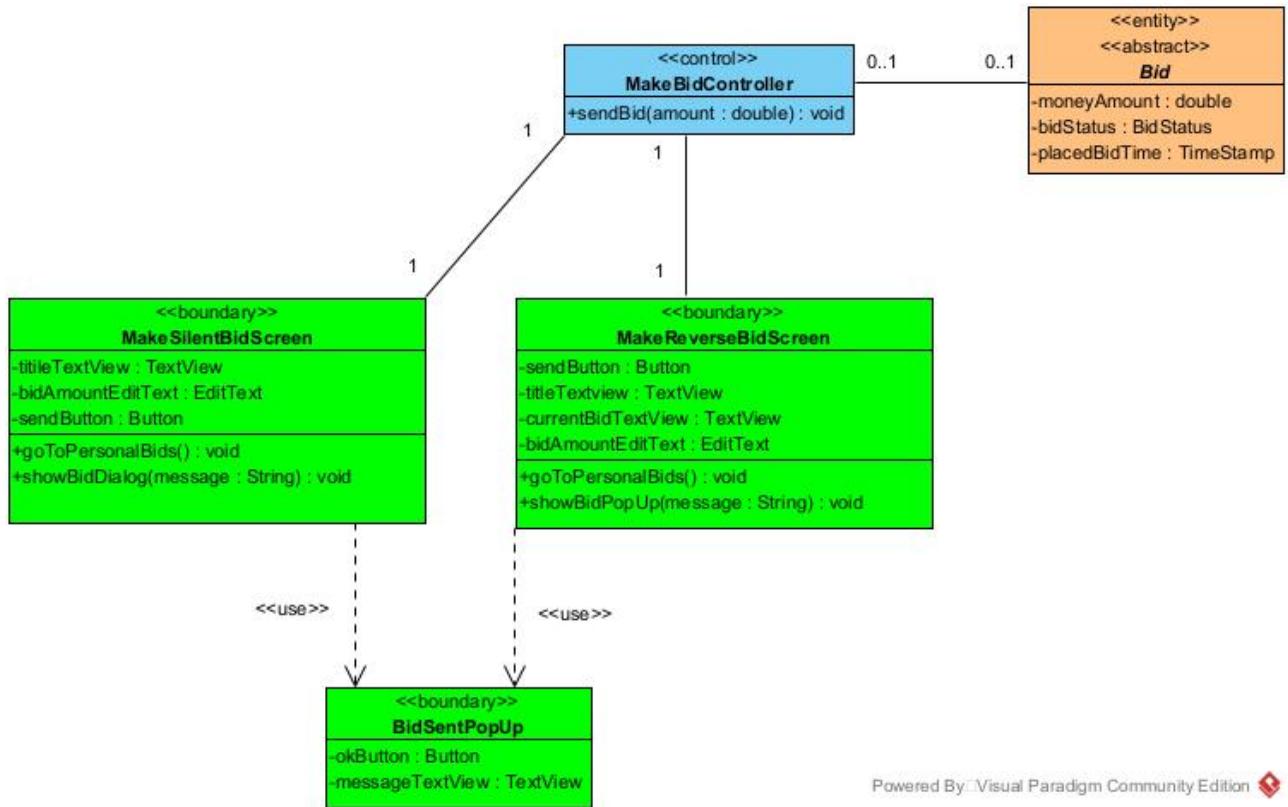
4. Class Diagram della schermata di home (Per evitare ridondanze sono stati omessi gli attributi e i metodi dei fragment: SearchAuctionsFragment, MyAuctionsFragment, CreateAuctionFragment, MyBidsFragment e ProfileFragment. Per vederli nel dettagli andare nei loro casi d'uso rispettivi).



5. Accesso.

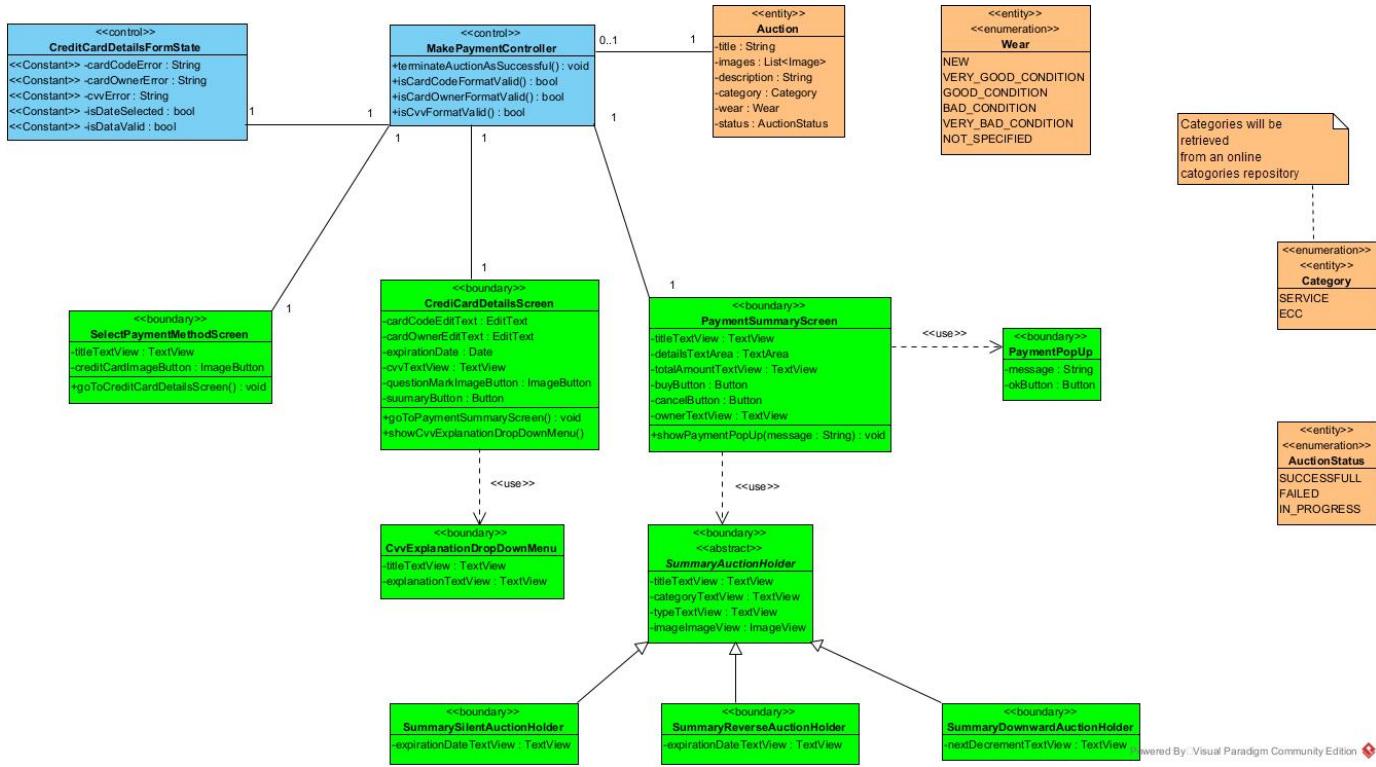


6. Invio di un'offerta ad un'asta.

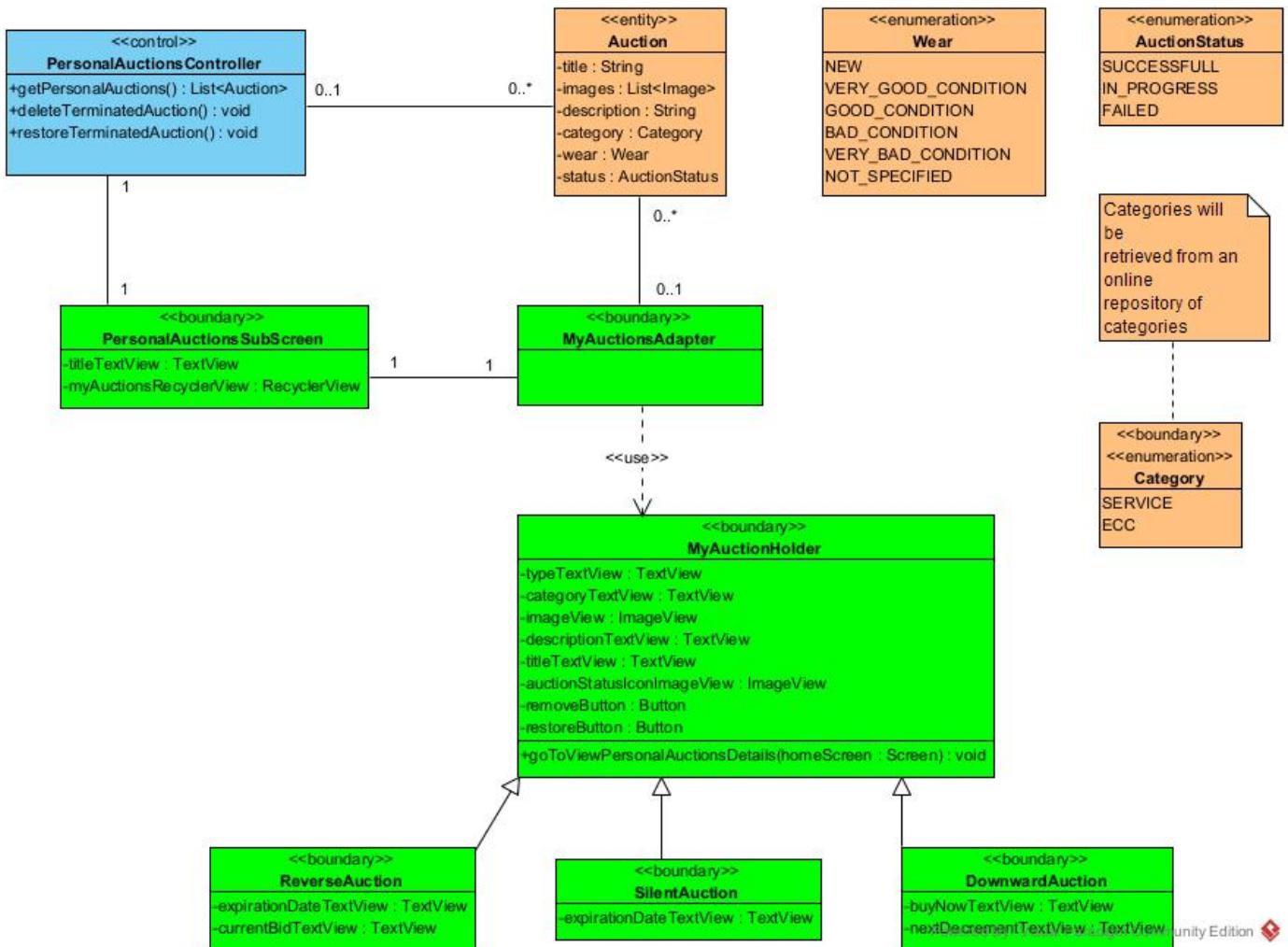


Powered By: Visual Paradigm Community Edition

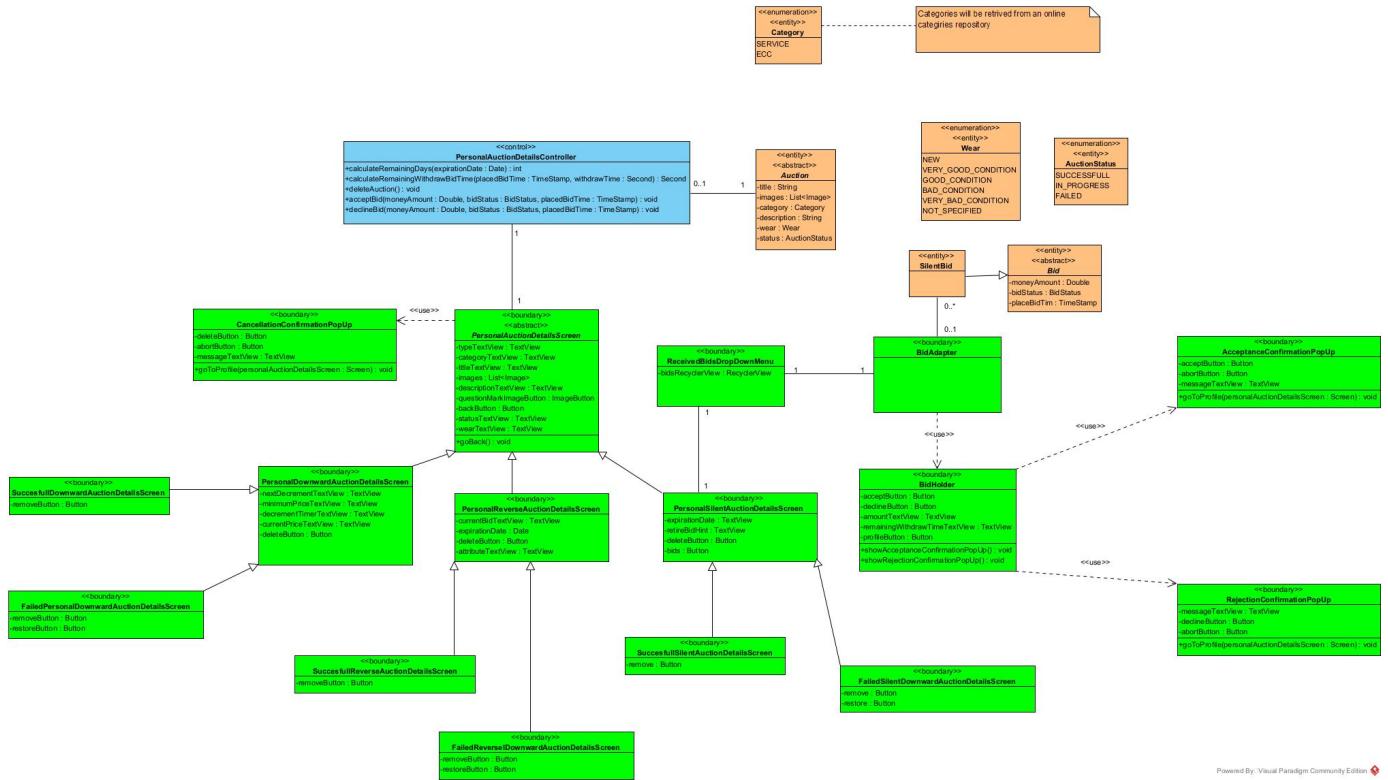
## 7. Effettuazione del pagamento.



8. Visualizzazione delle proprie aste.

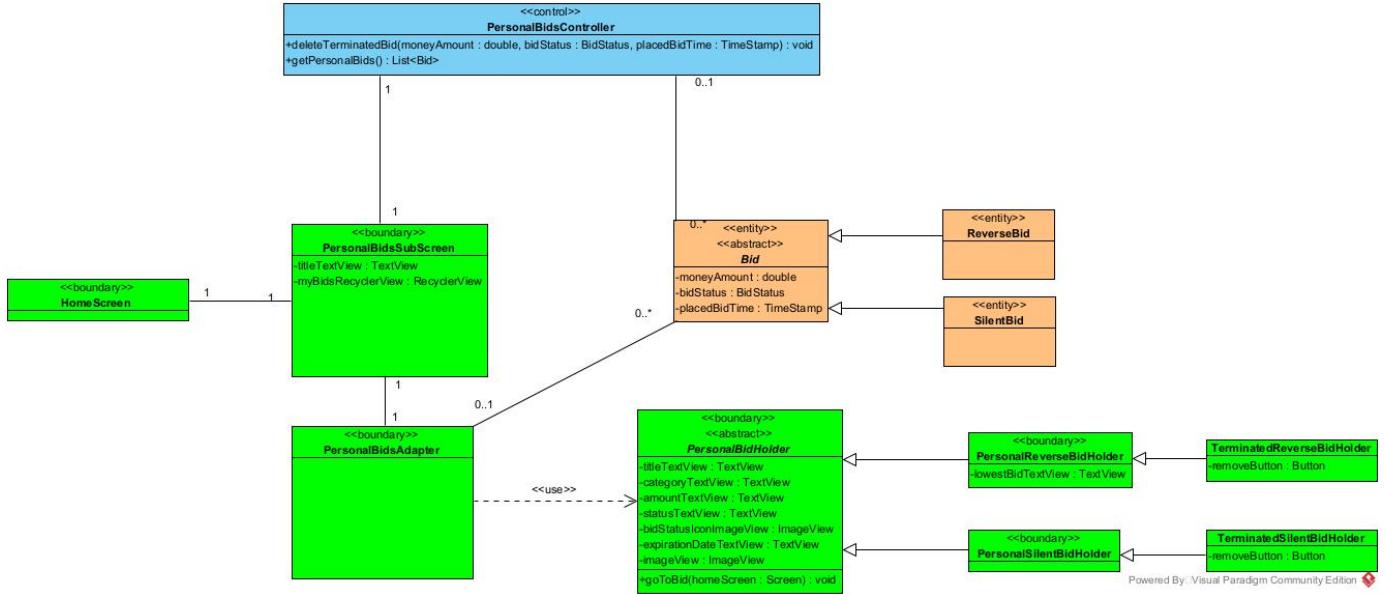


9. Visualizzazione dei dettagli di una propria asta.

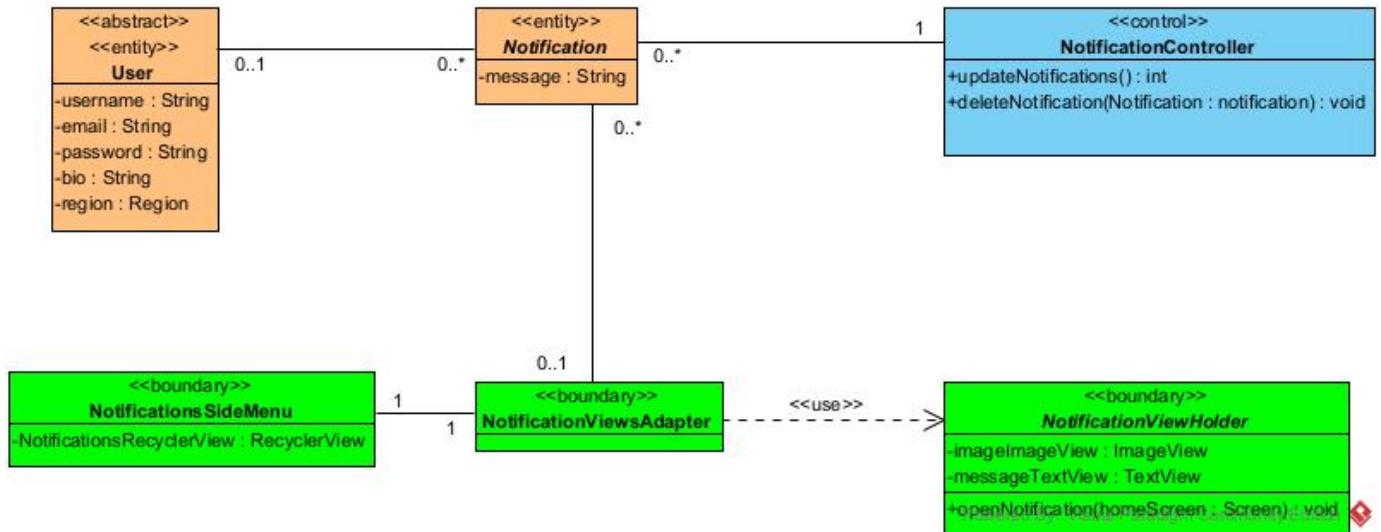


Powered By: Visual Paradigm Community Edition

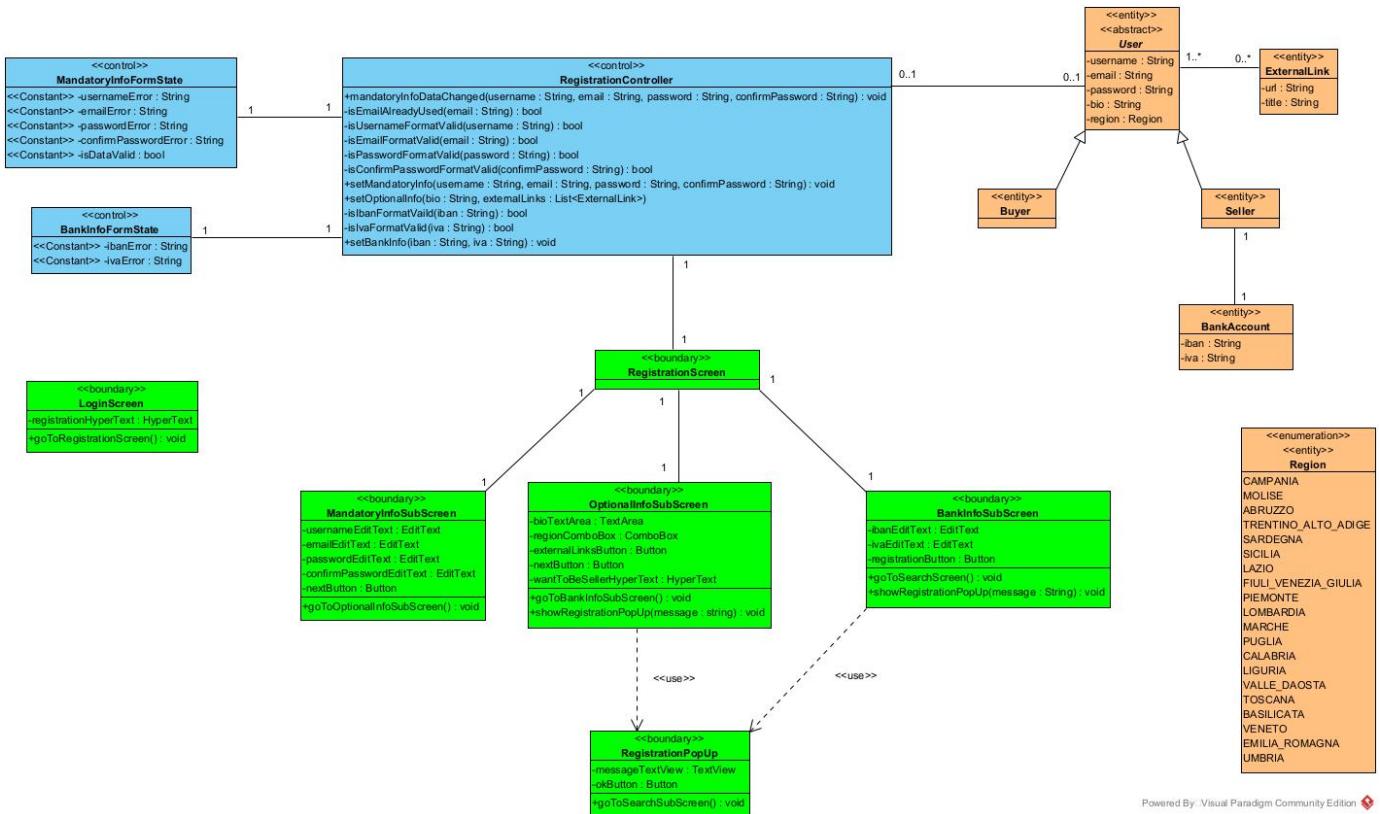
10. Visualizzazione delle offerte inviate alle aste altrui.



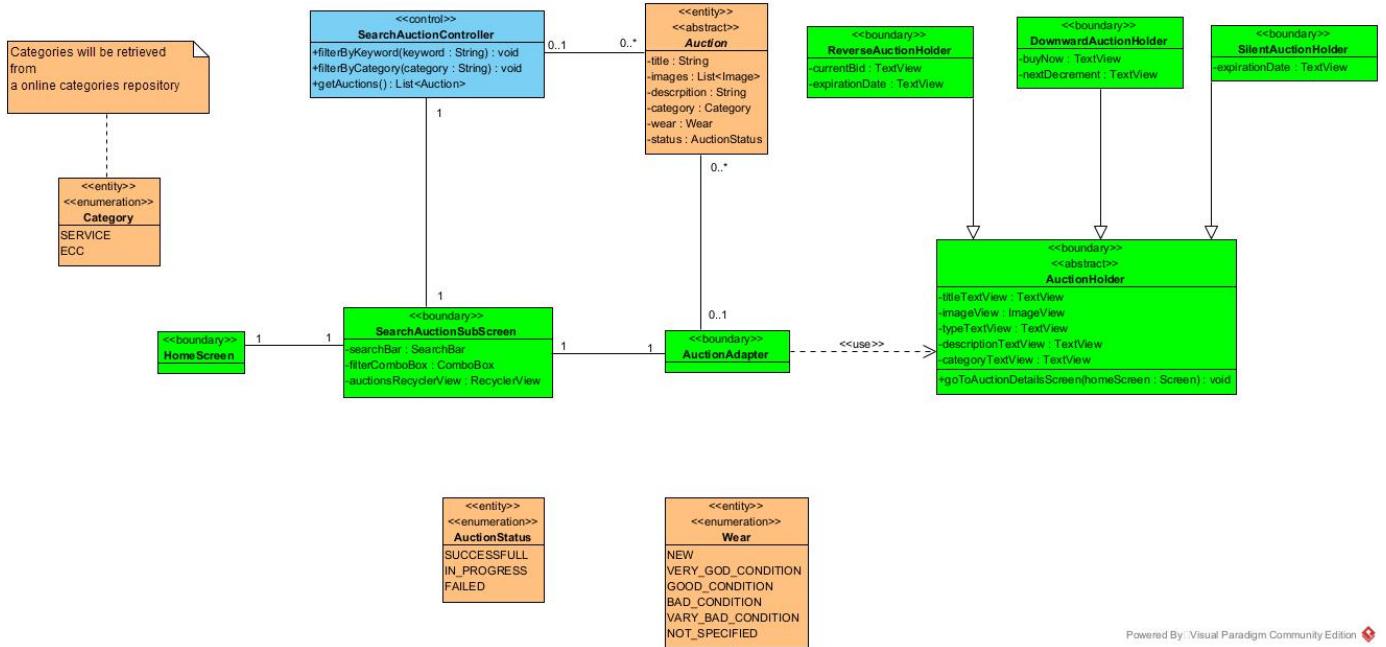
11. Visualizzazione delle notifiche.



## 12. Registrazione.

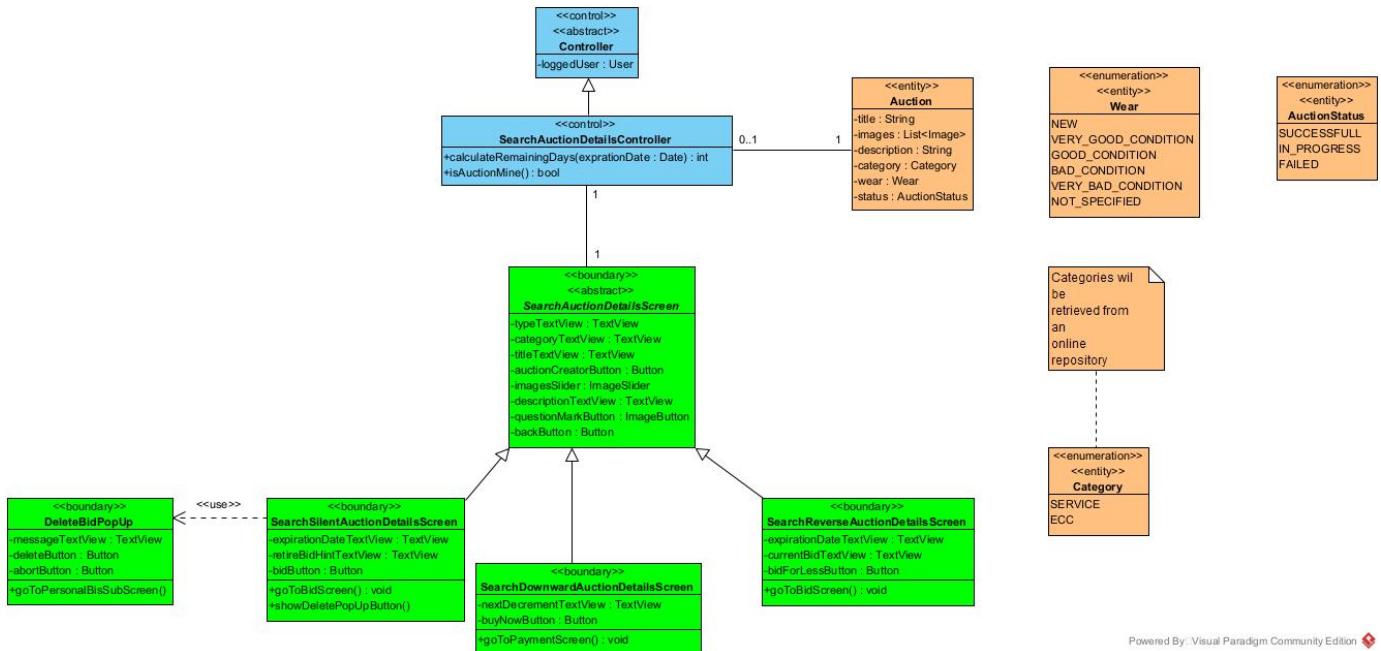


13. Ricerca di un'asta.



Powered By: Visual Paradigm Community Edition

14. Visualizzazione dei dettagli di un'asta altrui.

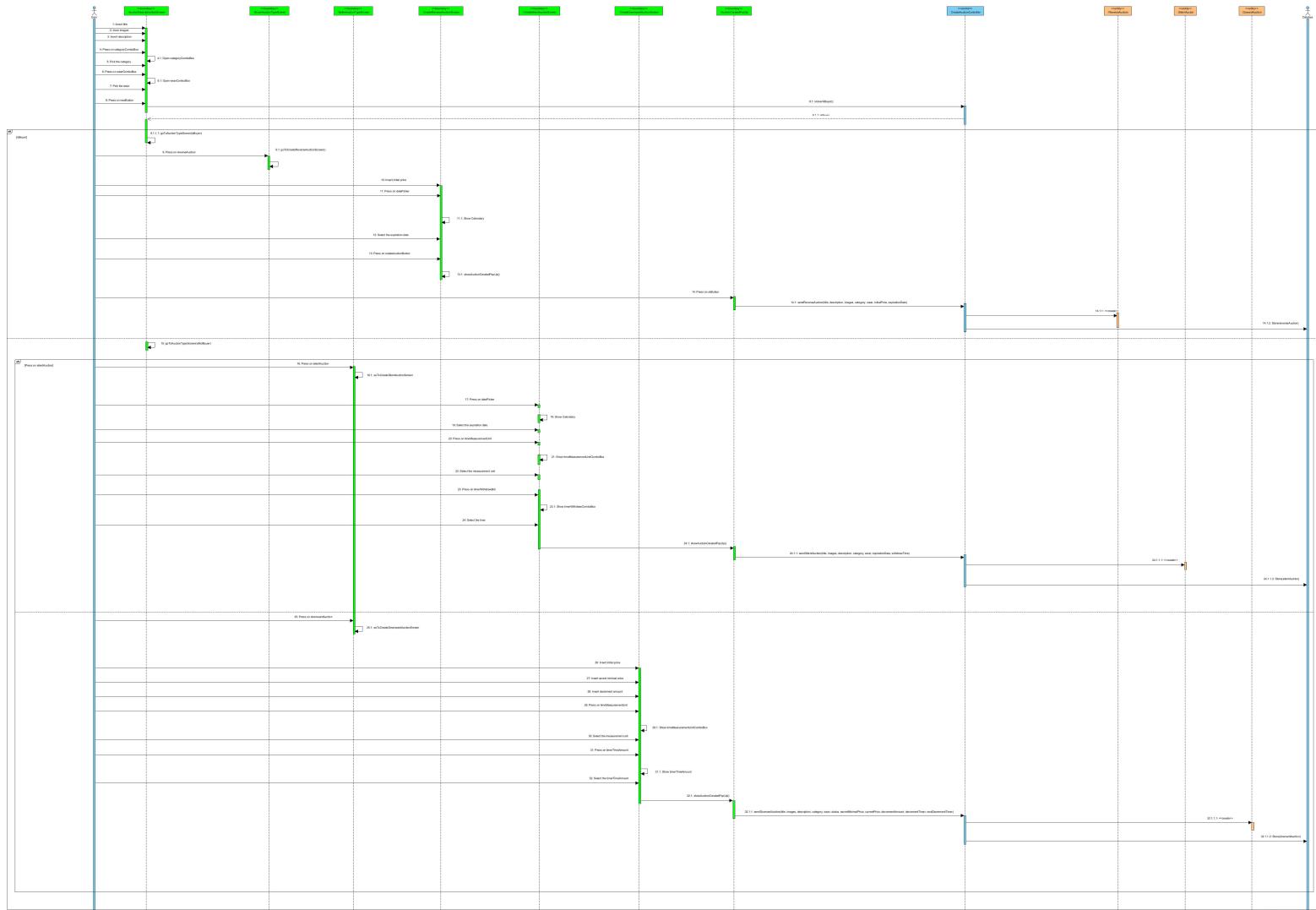


Powered By: Visual Paradigm Community Edition

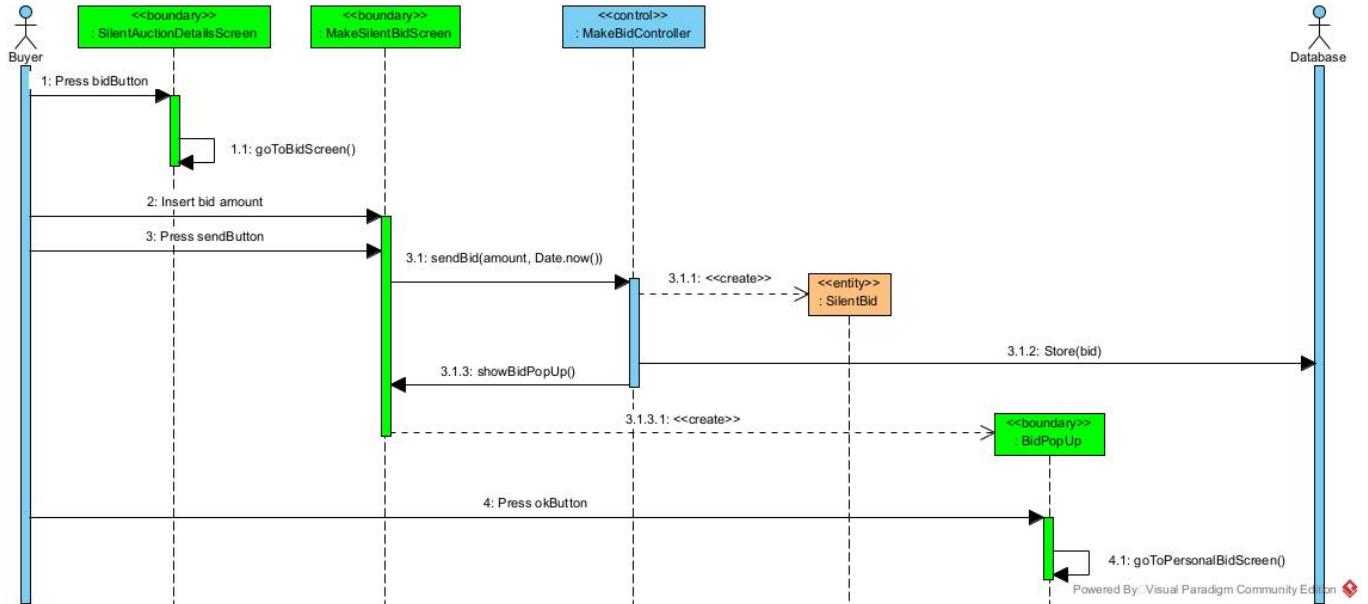
### **2.2.2 Sequence diagram di analisi.**

Di seguito le due funzionalità da noi scelte descritte tramite [Sequence Diagram](#).

## 1. Creazione di un'asta.



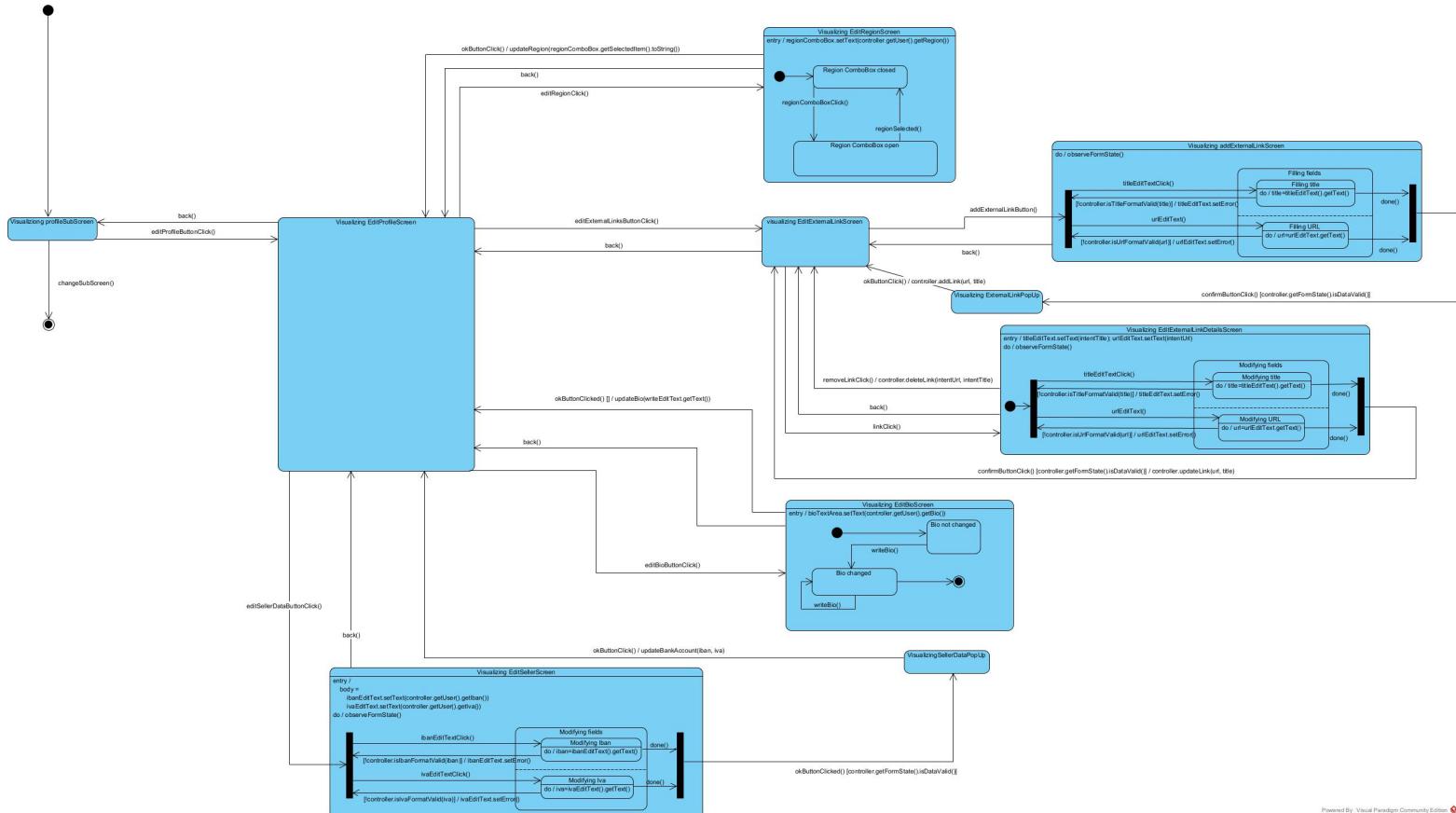
2. Invio di un'offerta ad un'asta silenziosa.



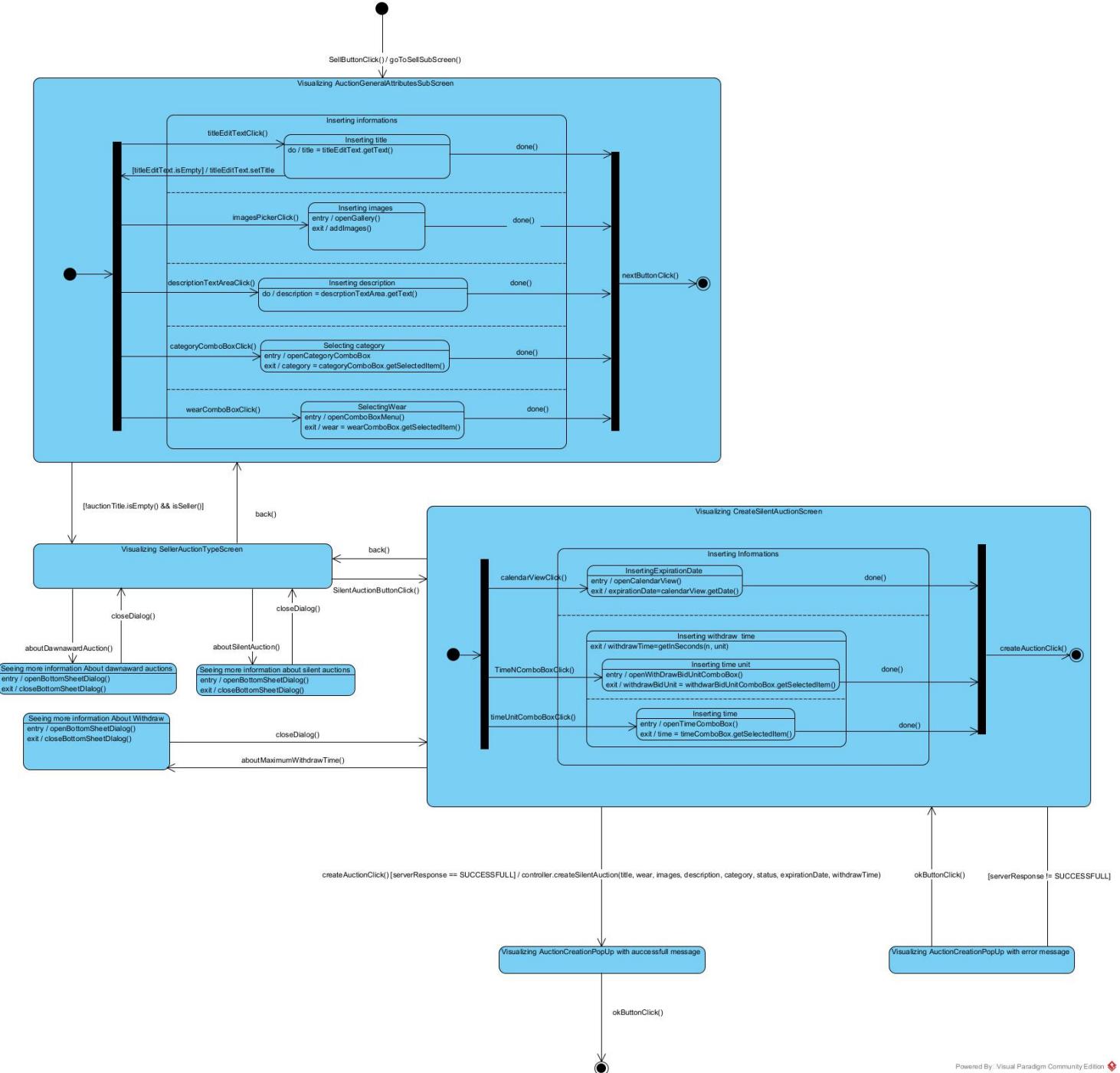
### 2.2.3 Statechart funzionali.

Di seguito gli Statechart funzionali dei casi d'uso identificati [qui](#).

#### 1. Personalizzazione del profilo.



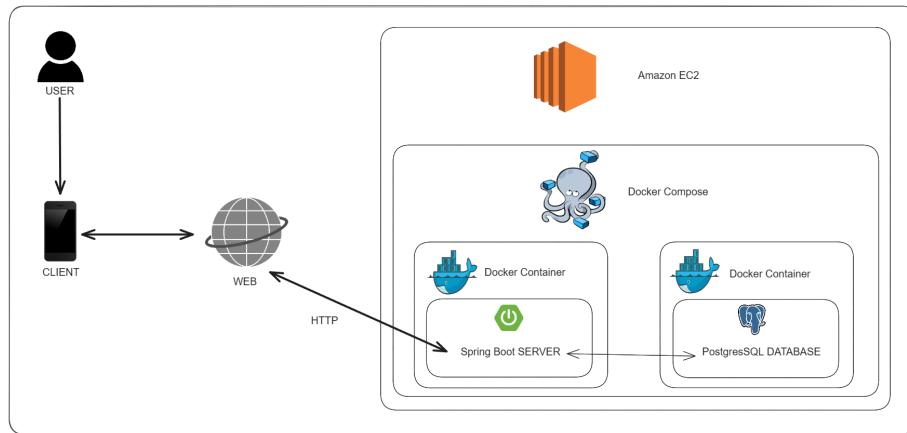
## 2. Creazione di un'asta silenziosa.



### 3 Design del sistema.

#### 3.1 Architettura del sistema e scelte tecnologiche adottate.

La nostra architettura è basata su tre livelli.



1. **Client (livello di presentazione):** questo livello rappresenta l'interfaccia utente. Nel nostro caso è stata realizzata come applicazione per dispositivi [Android](#).
2. **Server (livello di applicazione):** questo livello rappresenta la gestione della logica di business. Nel nostro caso il livello è stato realizzato utilizzando il framework [Spring Boot](#).
3. **Database (livello dei dati):** questo livello rappresenta la gestione della persistenza dei dati. Nel nostro caso il livello è stato realizzato utilizzando il database relazionale [Postgres](#).

Il nostro software per la realizzazione di una compravendita di aste punta sulle seguenti qualità:

- Usabilità: per aumentare questa qualità il dispositivo scelto sul quale girerà il software è lo smartphone Android. Programmare direttamente con API Android ci consentirà di avere accesso alle numerose feature specifiche di Android.
- Sicurezza: per aumentare la sicurezza abbiamo scelto di far autenticare l'utente mediante password che viene criptata nel database.
- Manutenibilità: l'applicazione è stata strutturata con l'idea che possa essere continuata in futuro aggiungendo nuove funzionalità.

Il nostro applicativo non punta alla performance dunque abbiamo preferito utilizzare il linguaggio di programmazione Java piuttosto che C++ (API native Android).

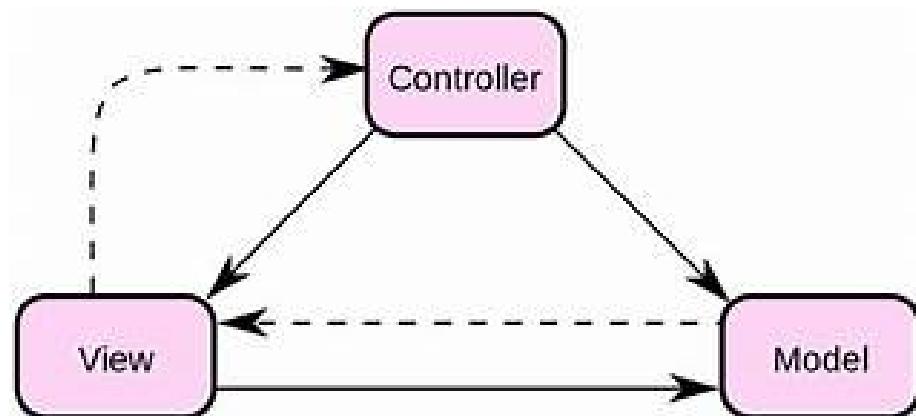
### 3.1.1 Client.

L'applicazione Android è stata sviluppata utilizzando le seguenti tecnologie:

- **Java**: linguaggio di programmazione imperativo orientato agli oggetti. Java dispone di un API per programmare con il sistema operativo Android. Il motivo per cui abbiamo scelto questo linguaggio piuttosto che un altro ( [Kotlin](#) o [C++](#) ) è che Java esistendo da più tempo ha un maggiore supporto sui forum di programmazione.
- **XML**: un linguaggio di Markup utilizzato per definire il layout delle schermate, indispensabile nella programmazione di app Android.

Il codice è stato strutturato con il pattern **MVC**:

- **Model**: classi che rappresentano le entità coinvolte nella logica di business.
- **View**: classi che rappresentano l'interfaccia utente e sono responsabili della comunicazione con quest'ultimo.
- **Controller**: classi che si occupano di gestire la logica di business e di comunicare con il database mediante le classi **DAO** (**Data Access Object**).

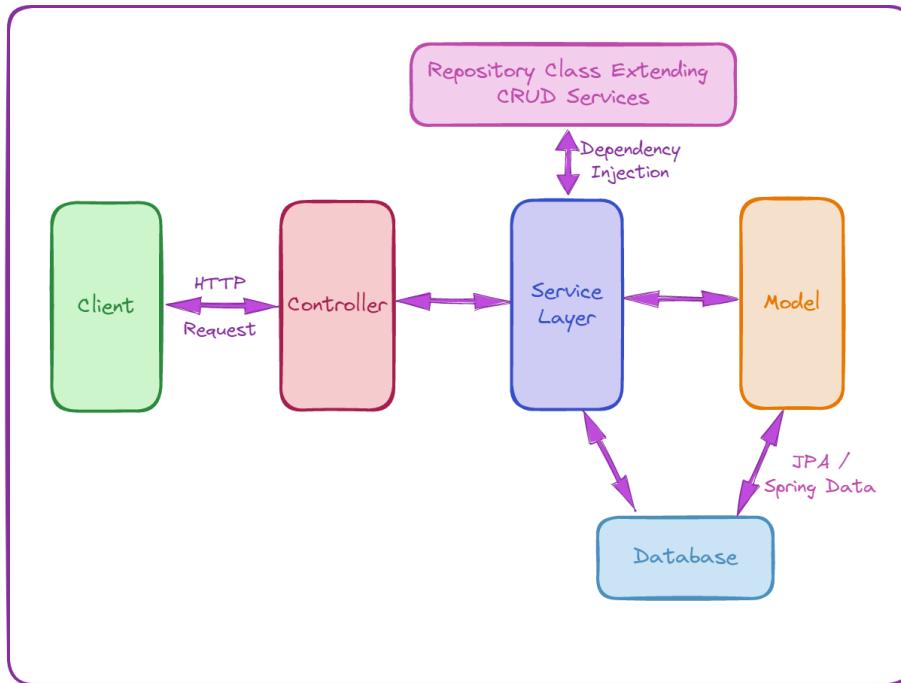


### 3.1.2 Server.

Il server è stato sviluppato utilizzando le seguenti tecnologie:

- **Spring Boot**: framework per facilitare lo sviluppo di un micro servizio, nel nostro caso un server che espone delle **REST API**, ovvero delle API accessibili mediante il protocollo **HTTP** (GET, POST, PUT, DELETE).
- **Spring JPA**: tool per automatizzare il mapping tra le classi java e le tabelle del database, e la creazione di metodi di accesso al database, ovvero operazioni **CRUD**.
- **JWT (JSON Web Token)**: standard web per firmare scambi di dati e criptare dati sensibili. Nel nostro caso, prima di qualsiasi operazione che modifica il database, è necessario autenticarsi mediante email e password. Segue la generazione di un token da inserire nell'Authorization Header di qualsiasi futura richiesta HTTP. Inoltre, le password nel database sono criptate, facendo sì che qualsiasi accesso al database non possa far trapelare questo dato particolarmente sensibile.

Il codice è stato realizzato seguendo il design pattern tipico delle applicazione Spring Boot.



- **REST Controller**: Classi incaricate della gestione delle richieste HTTP. Esse invocheranno i metodi delle classi **Service**.
- **Service**: Classi intermedie che richiameranno i metodi delle classi **Repository**
- **Repository**: Classi che si occupano delle operazioni **CRUD** sul database.

### **3.1.3 Comunicazione Client-Server.**

Nella fase di sviluppo del Server, per testare la bontà dei metodi che gestiscono le richieste HTTP abbiamo utilizzato l'applicativo [Postman](#), tool per l'invio e la ricezione di richieste e risposte HTTP.

Invece, l'applicazione Android finale per comunicare con il server utilizza [Retrofit](#), libreria che consente di mandare richieste HTTP ad un indirizzo specificato.

### **3.1.4 Fase di deployment.**

La fase di deployment consiste nella generazione del file APK dell'applicazione Android e nella migrazione al cloud del server. Per effettuare tale operazione ci siamo avvalsi delle seguenti tecnologie:

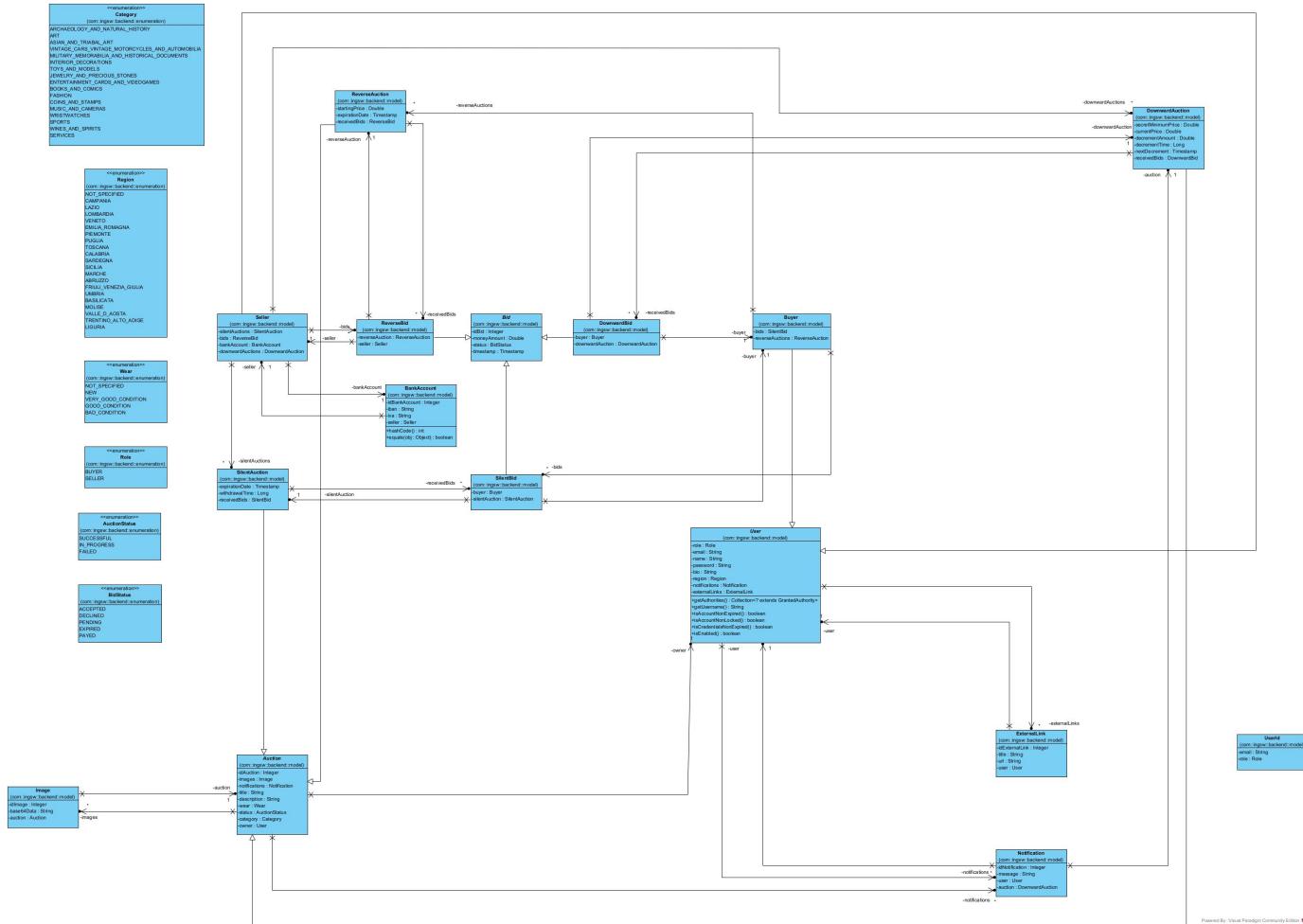
- [Docker](#): strumento per la gestione di container utilizzato nel nostro caso per la conteinerizzazione del server e del database in ambienti virtuali indipendenti dall'architettura fisica della macchina sottostante.
- [EC2](#): servizio offerto da [AWS \(Amazon Web Services\)](#), per il noleggio di macchine virtuali sulle quali vengono eseguite applicazioni. Tale servizio serve a rendere l'applicazione scalabile in modo da spendere la giusta cifra in base alle necessità.

## **3.2 Class Diagram di design.**

Di seguito mostriamo i [Class Diagram](#) di design.

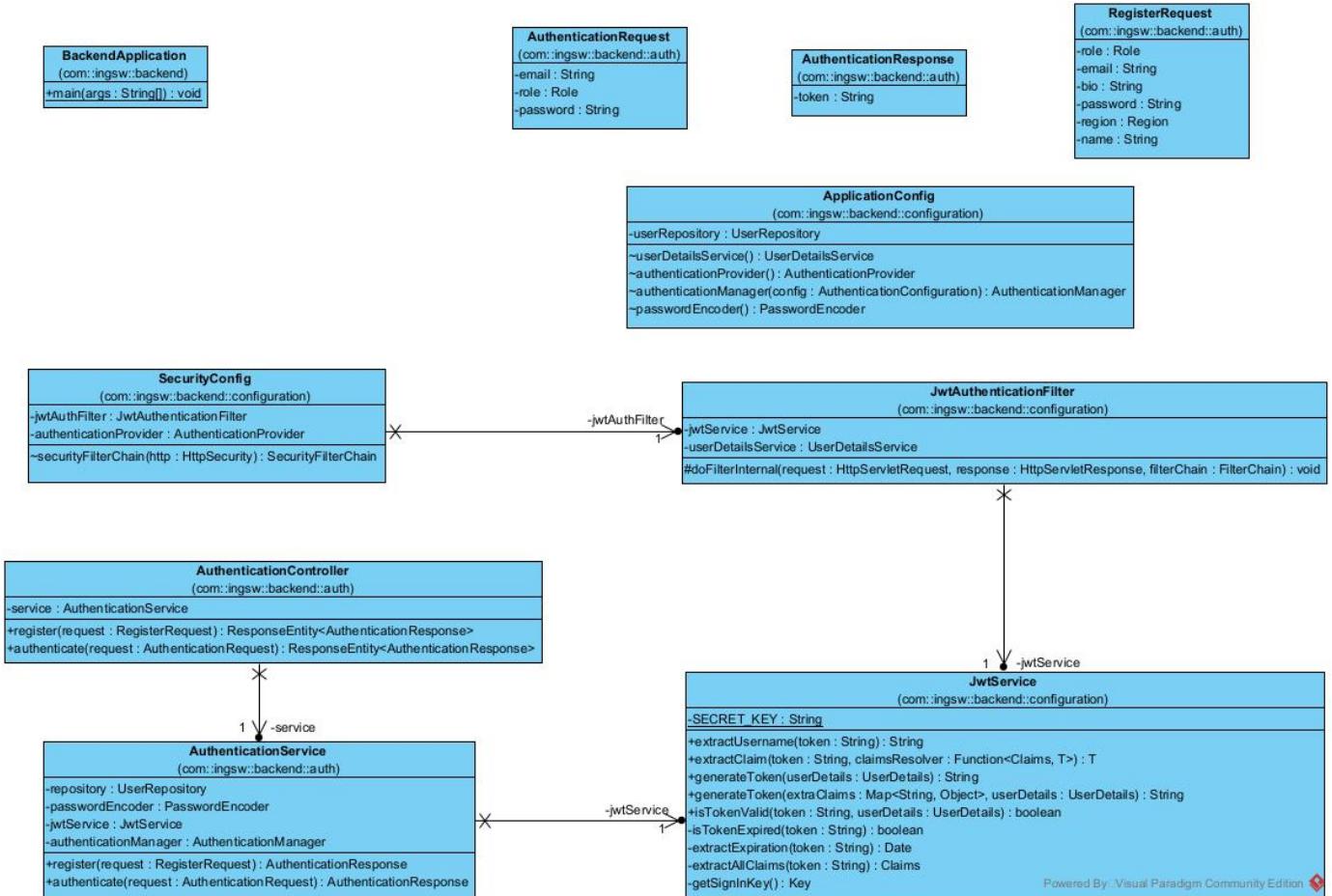
### 3.2.1 Backend.

#### 1. Entità.

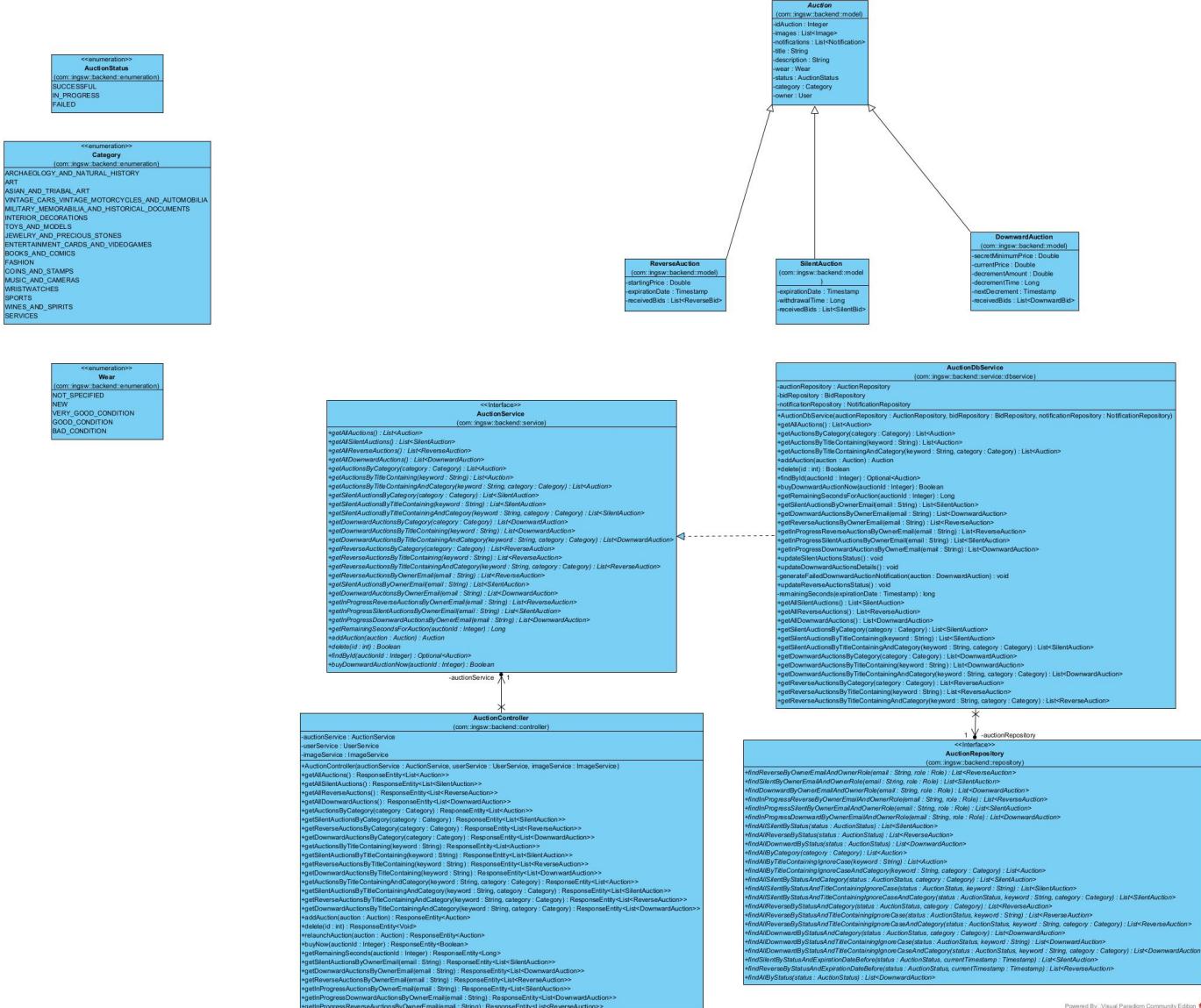


Powered By: Visual Paradigm Community Edition.

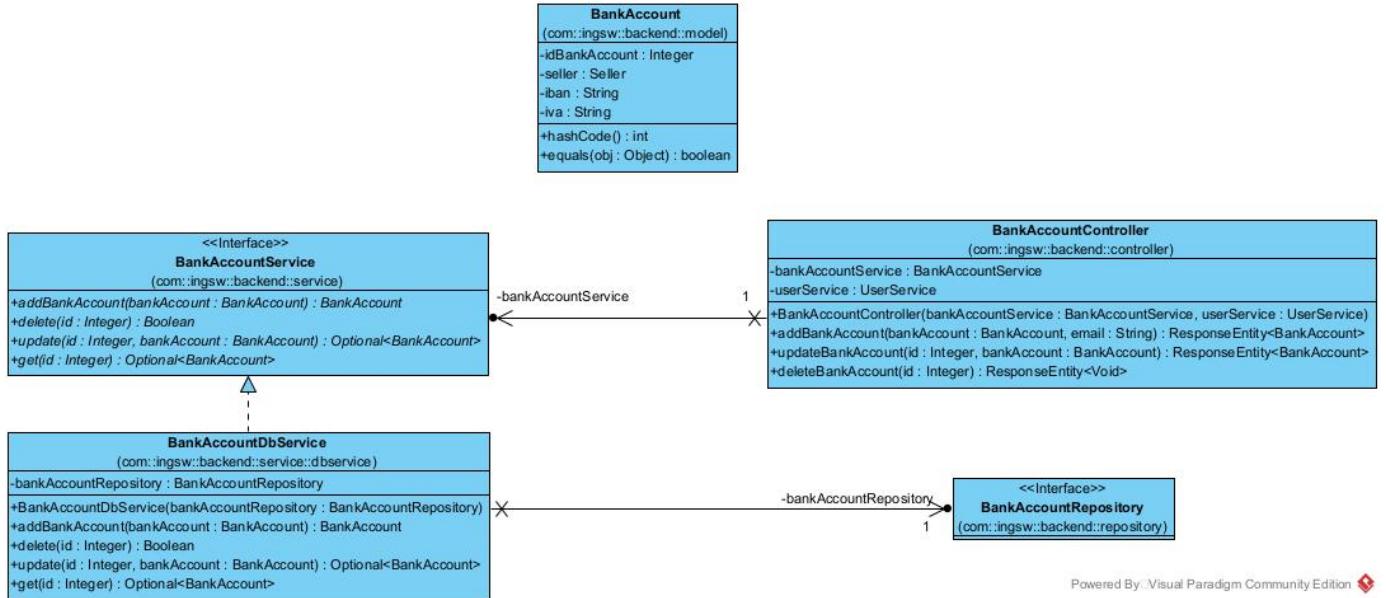
## 2. Sicurezza.



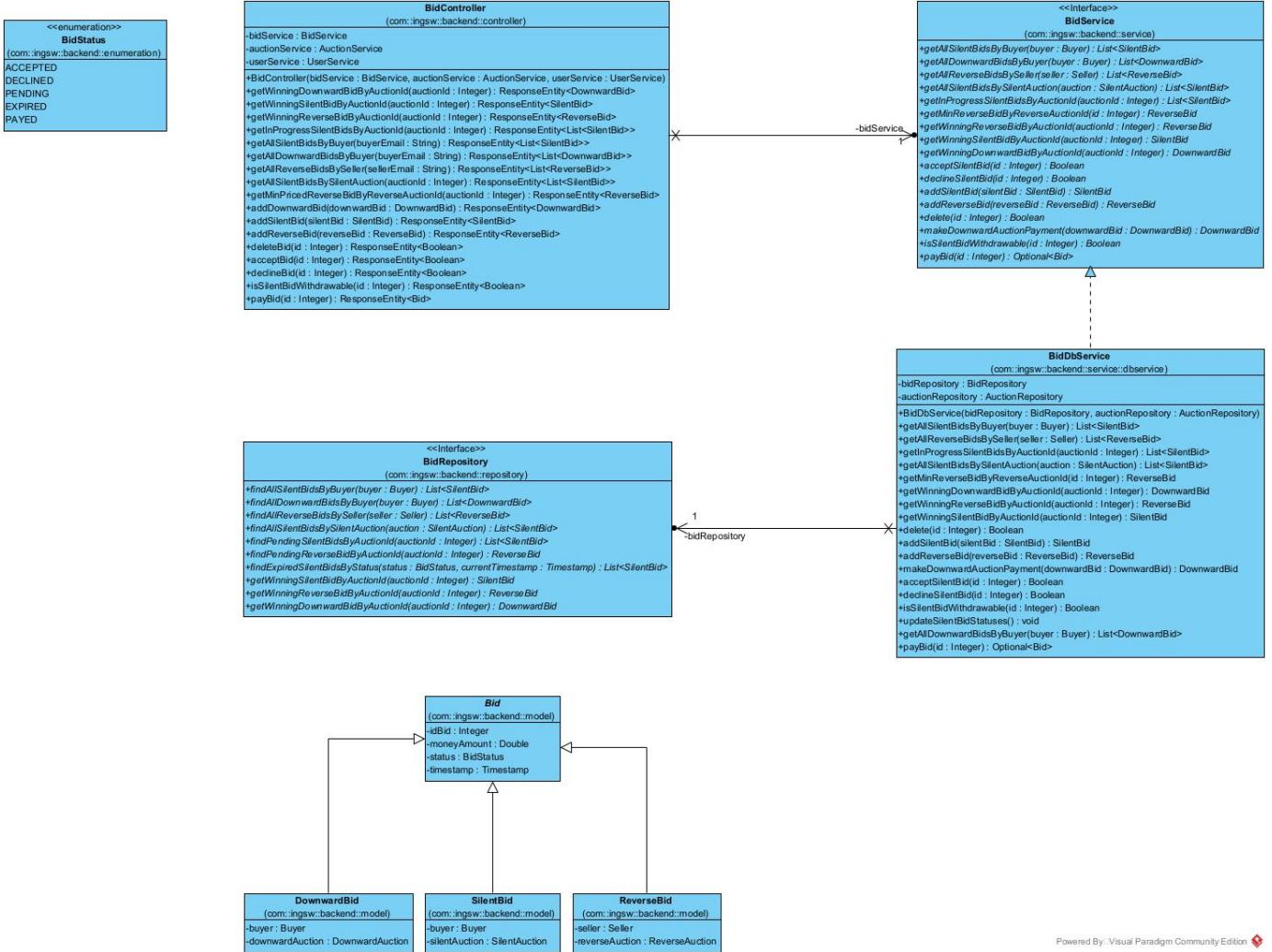
### 3. Asta.



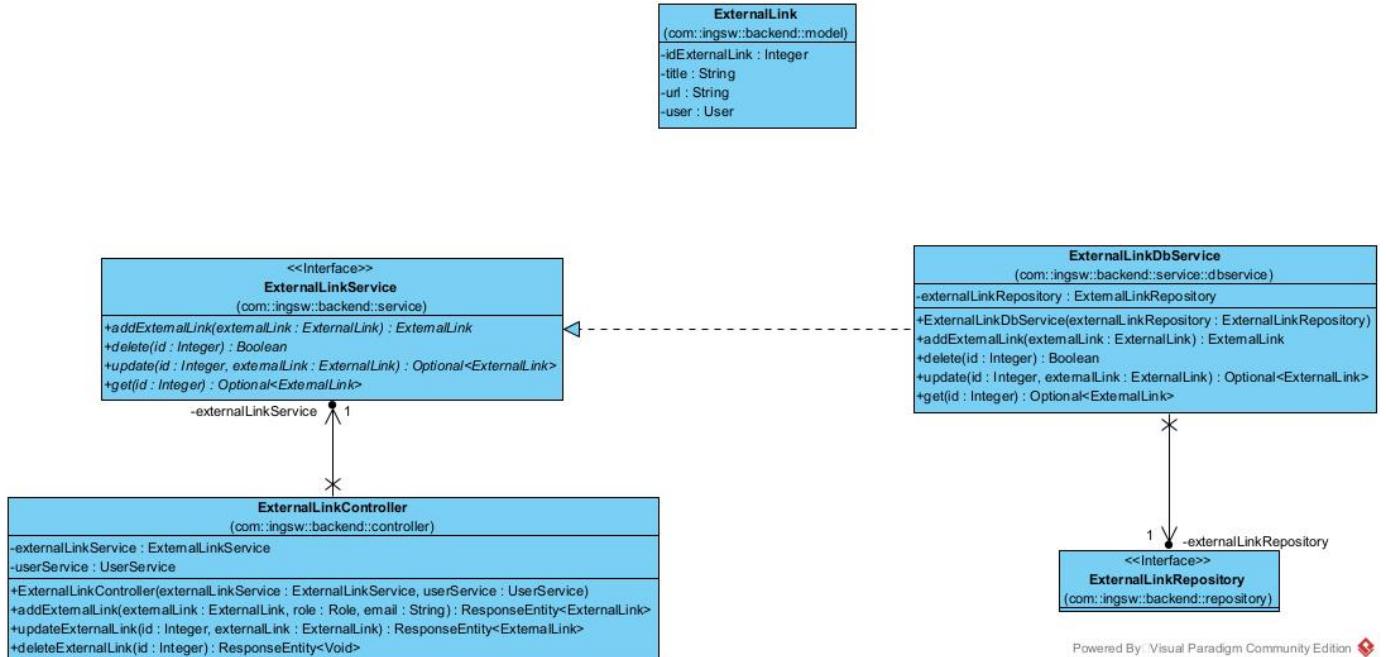
#### 4. Account bancario.



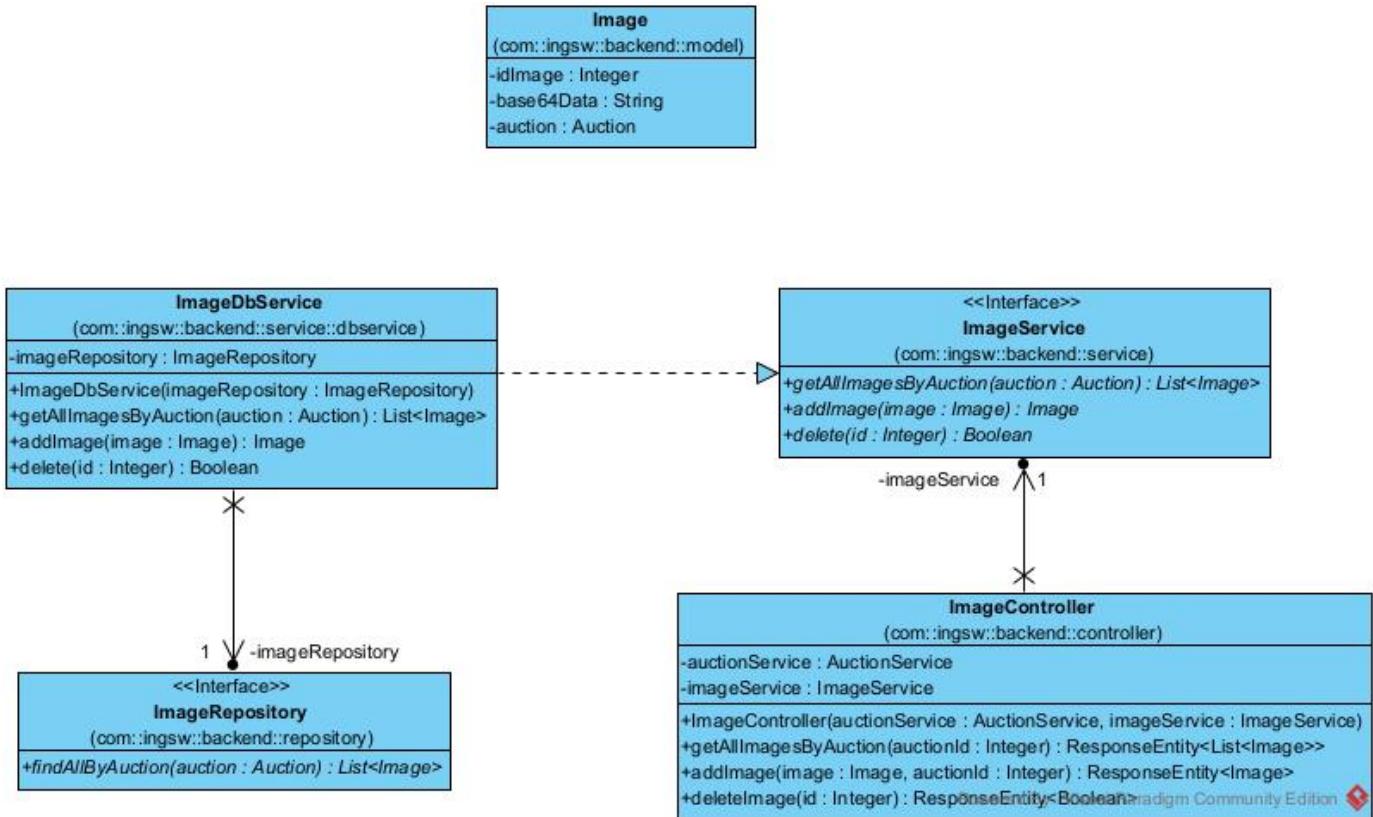
## 5. Offerta.



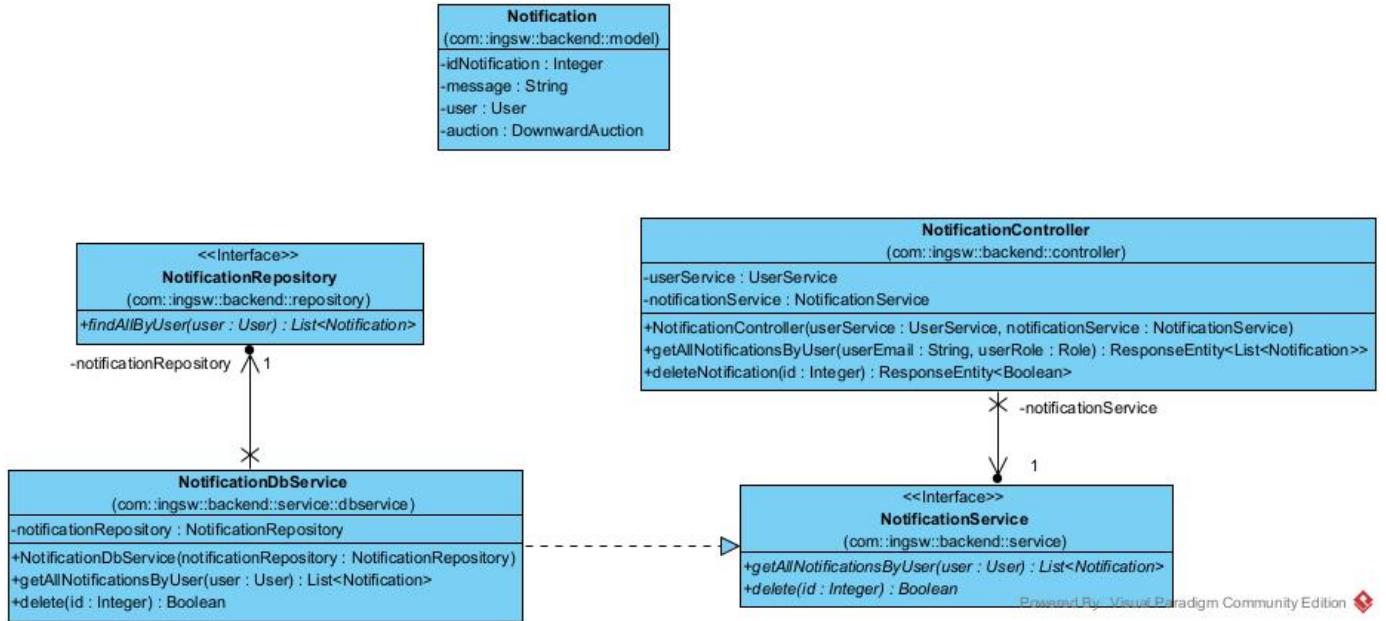
## 6. Link esterno.



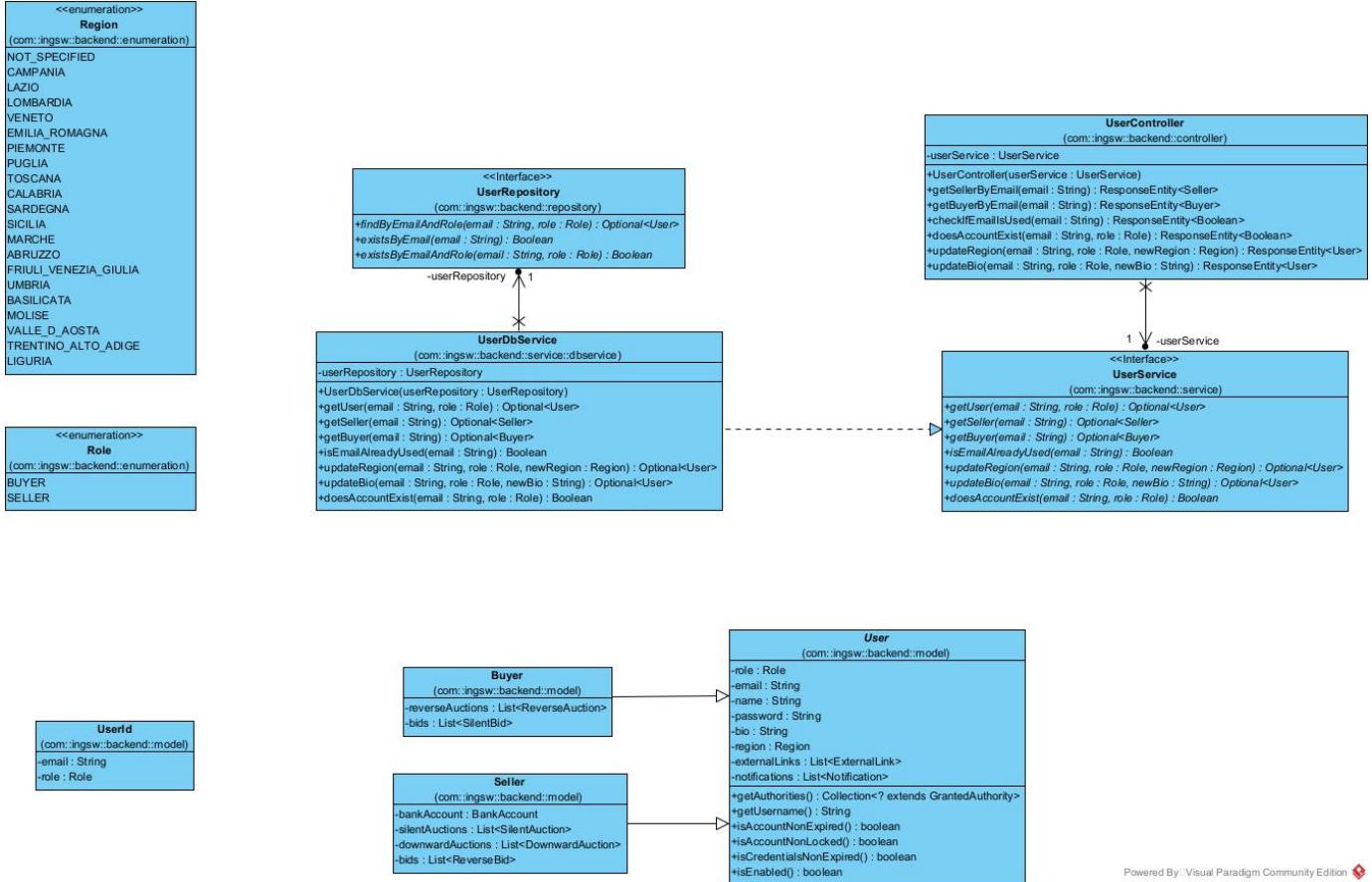
7. Immagine.



8. Notifica.



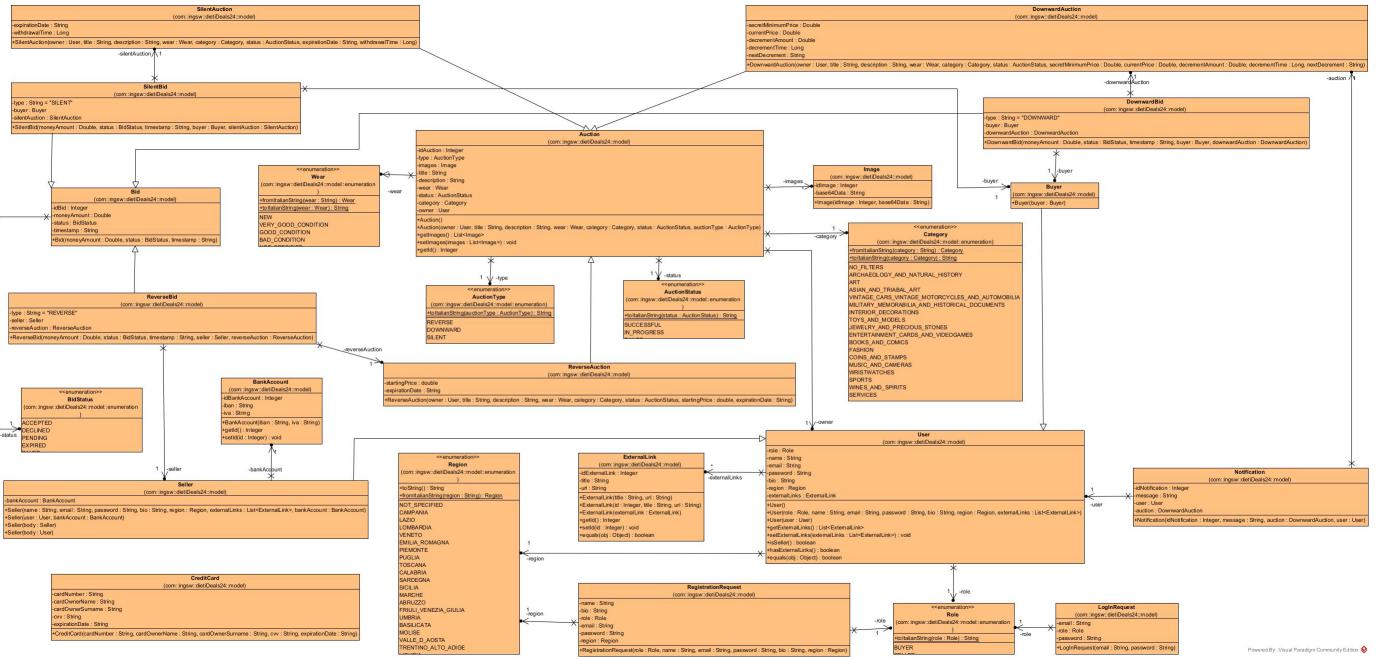
## 9. Utente.



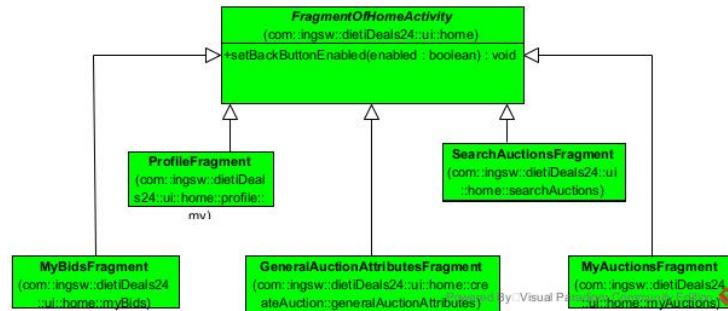
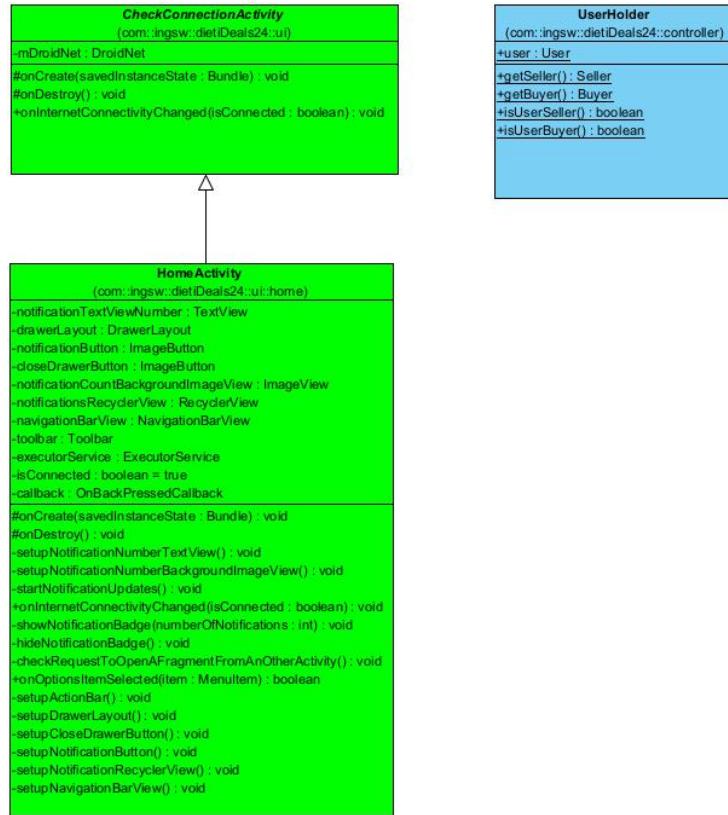
### 3.2.2 Frontend.

In tutti i Class Diagram la classe PopUpGenerator, per evitare ridondanze, contiene solo i metodi che interessano quel caso d'uso.

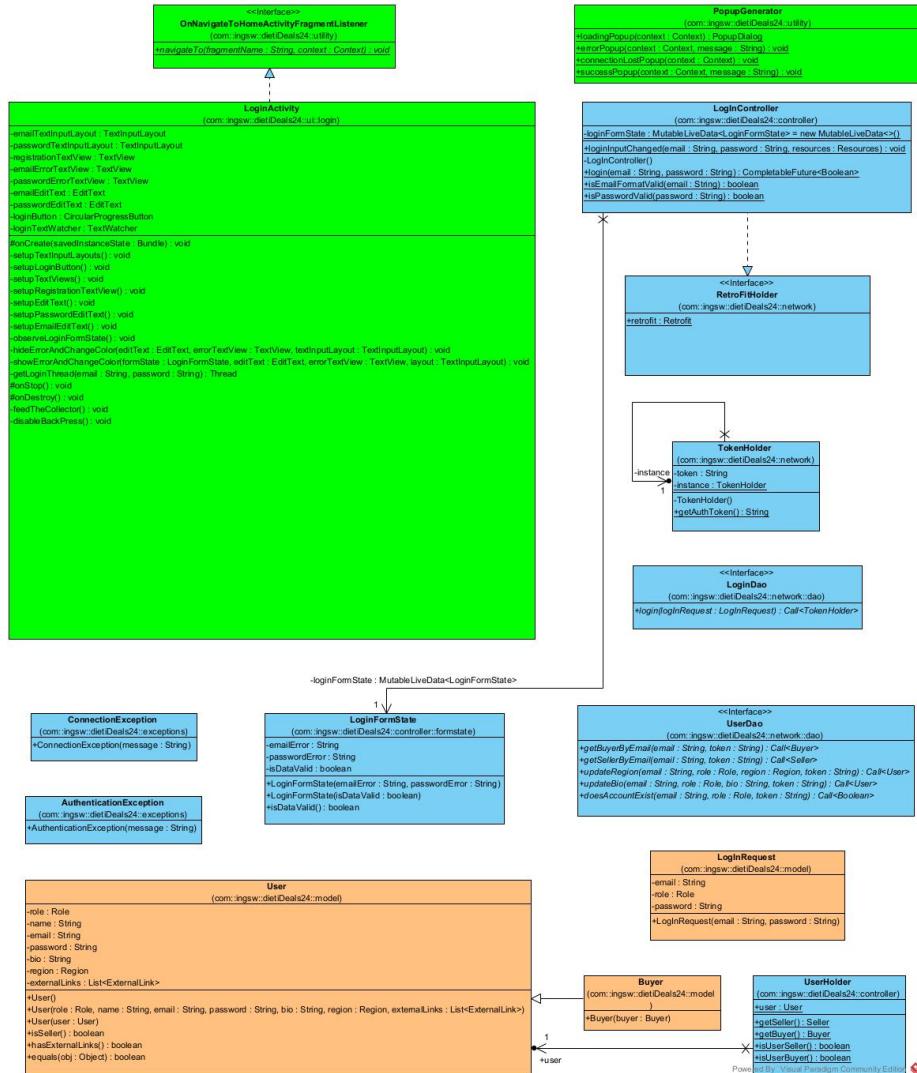
#### 1. Entità.



2. Home (Per evitare ridondanze sono stati omessi gli attributi e i metodi dei fragment: SearchAuctionsFragment, MyAuctionsFragment, CreateAuctionFragment, MyBidsFragment e ProfileFragment. Per vederli nel dettagli andare nei loro casi d'uso rispettivi).



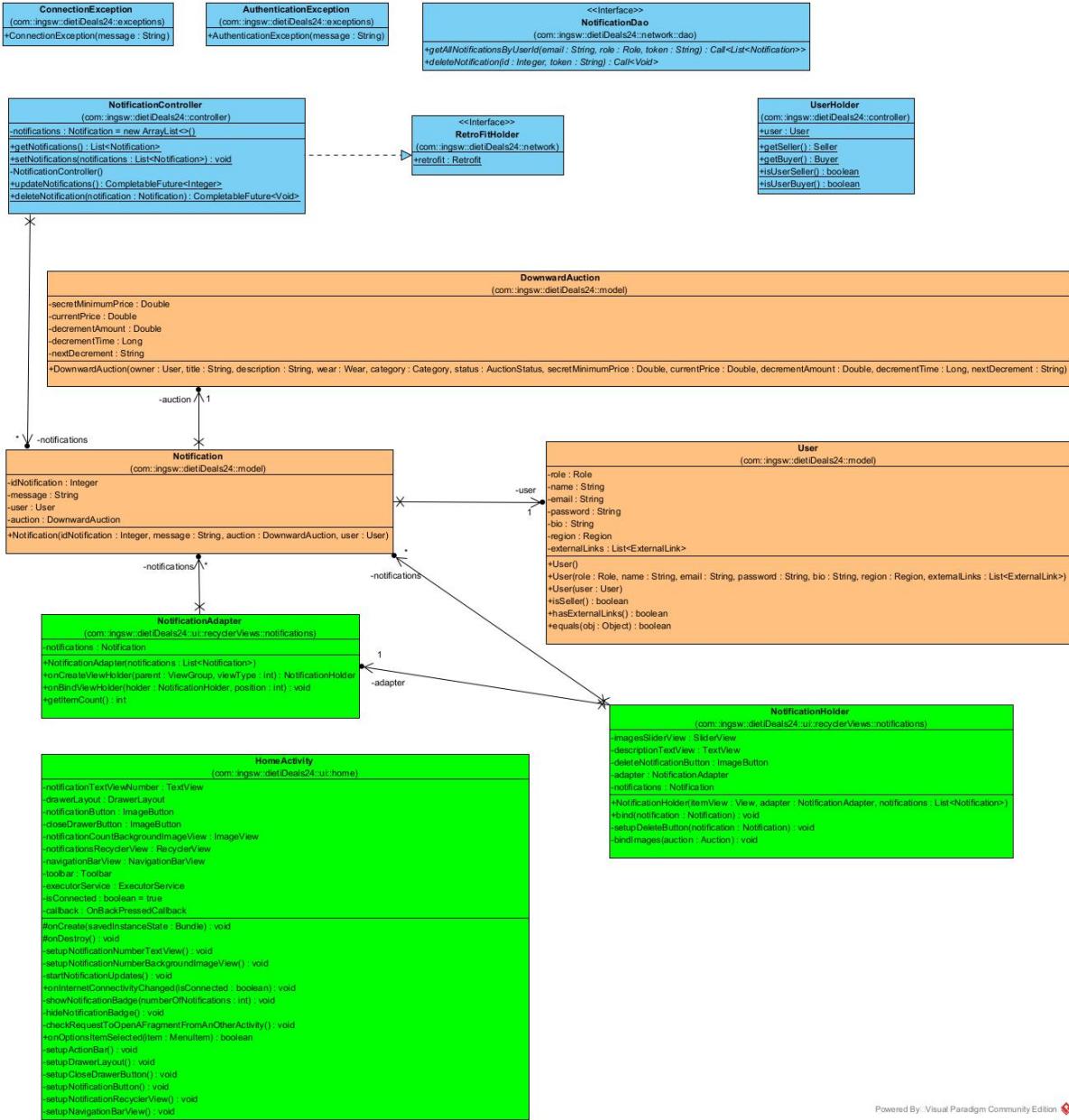
### 3. Accesso.



## 4. Registrazione.

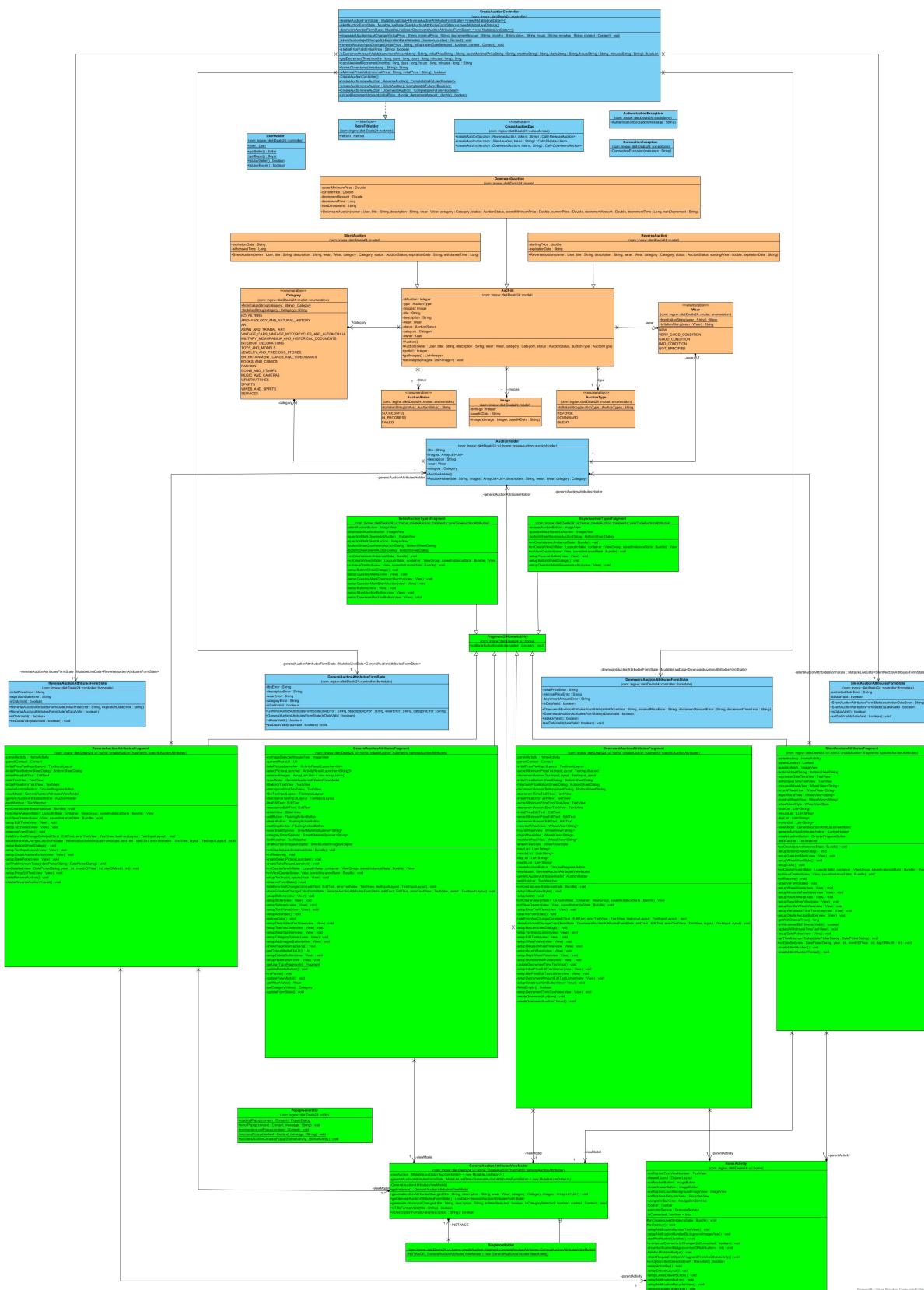


## 5. Visualizzazione delle notifiche.



Powered By: Visual Paradigm Community Edition

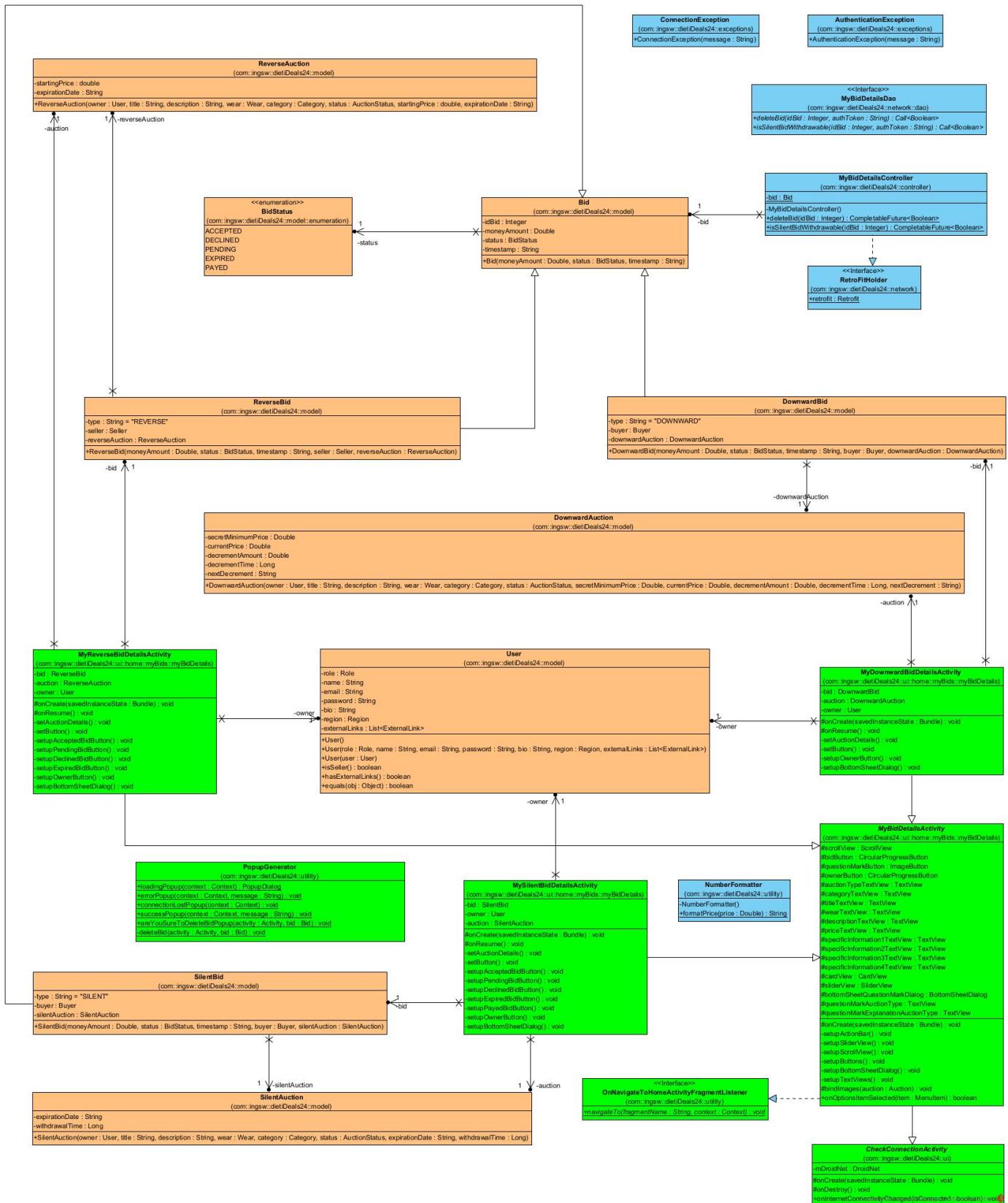
## 6. Creazione dell'asta.



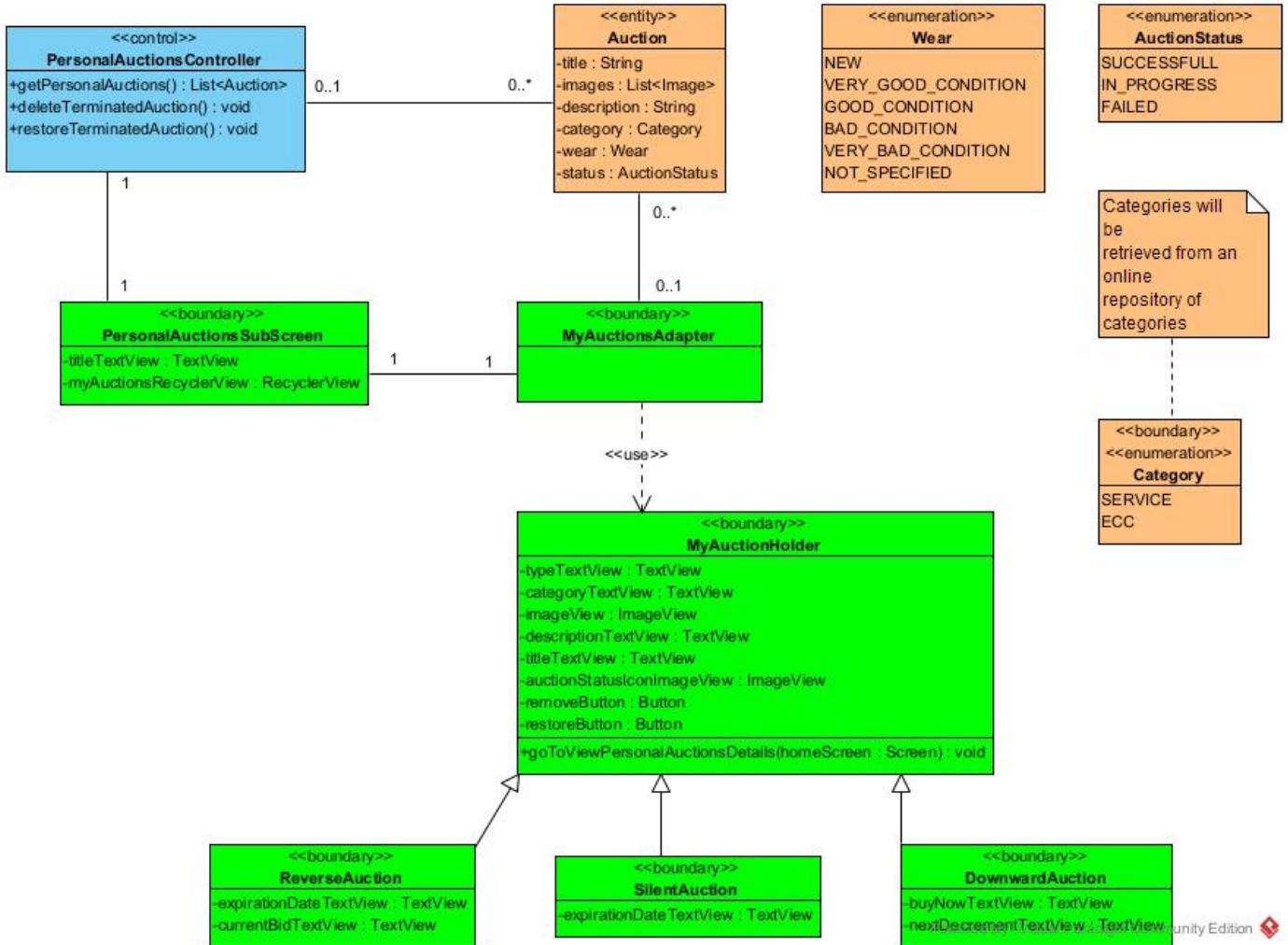
## 7. Le mie offerte.



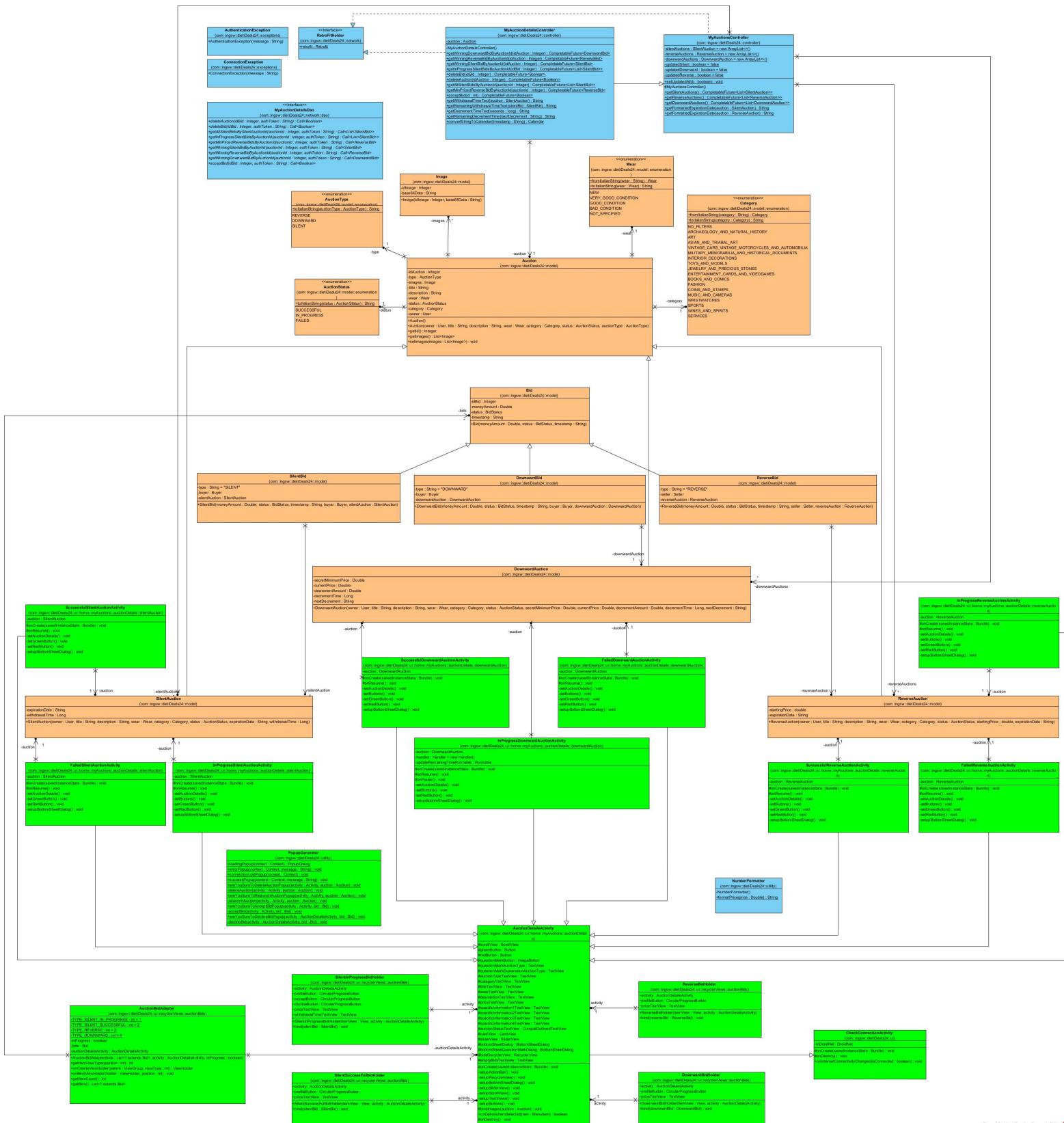
## 8. Visualizza i dettagli della mia offerta.



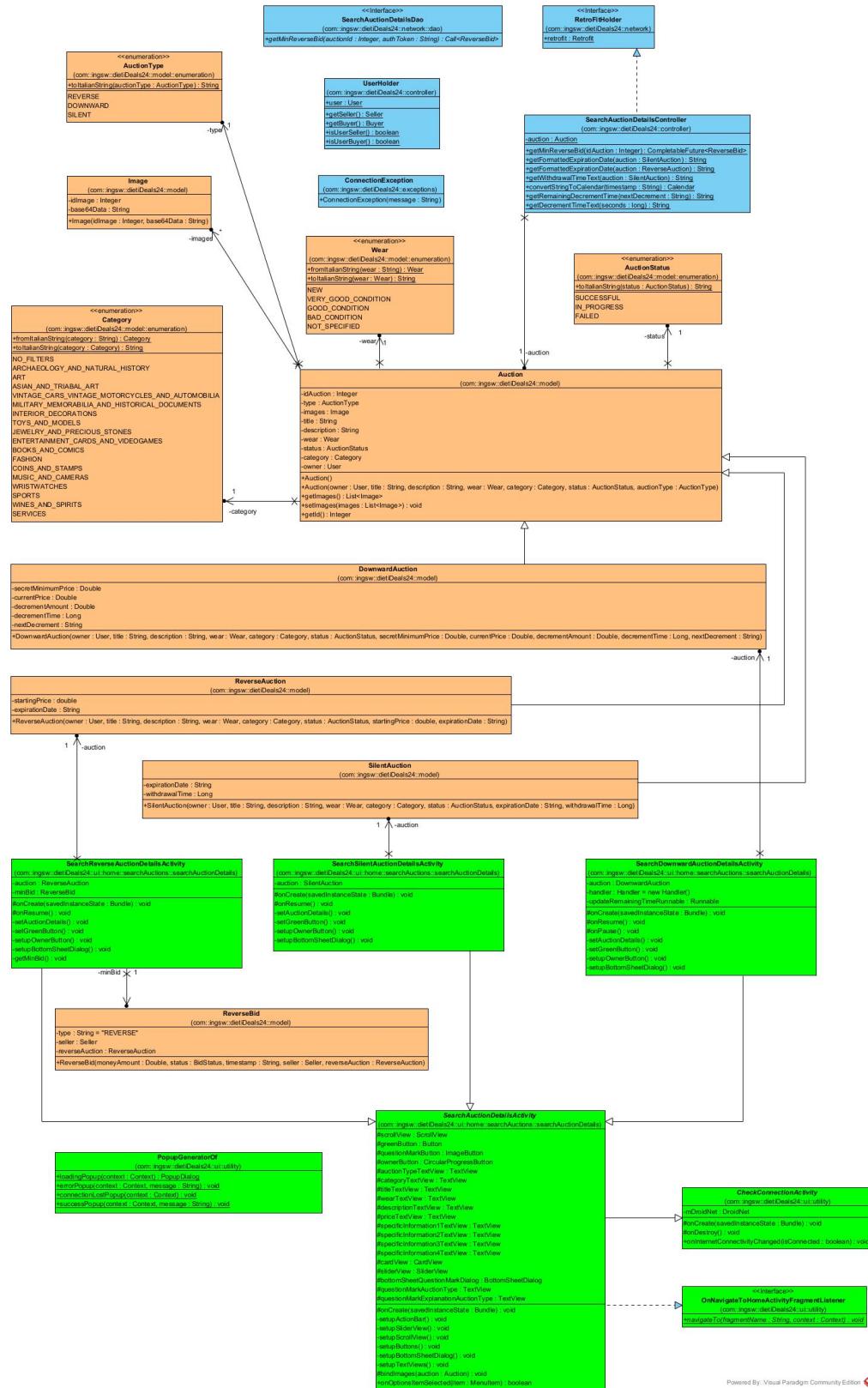
9. Le mie asta.



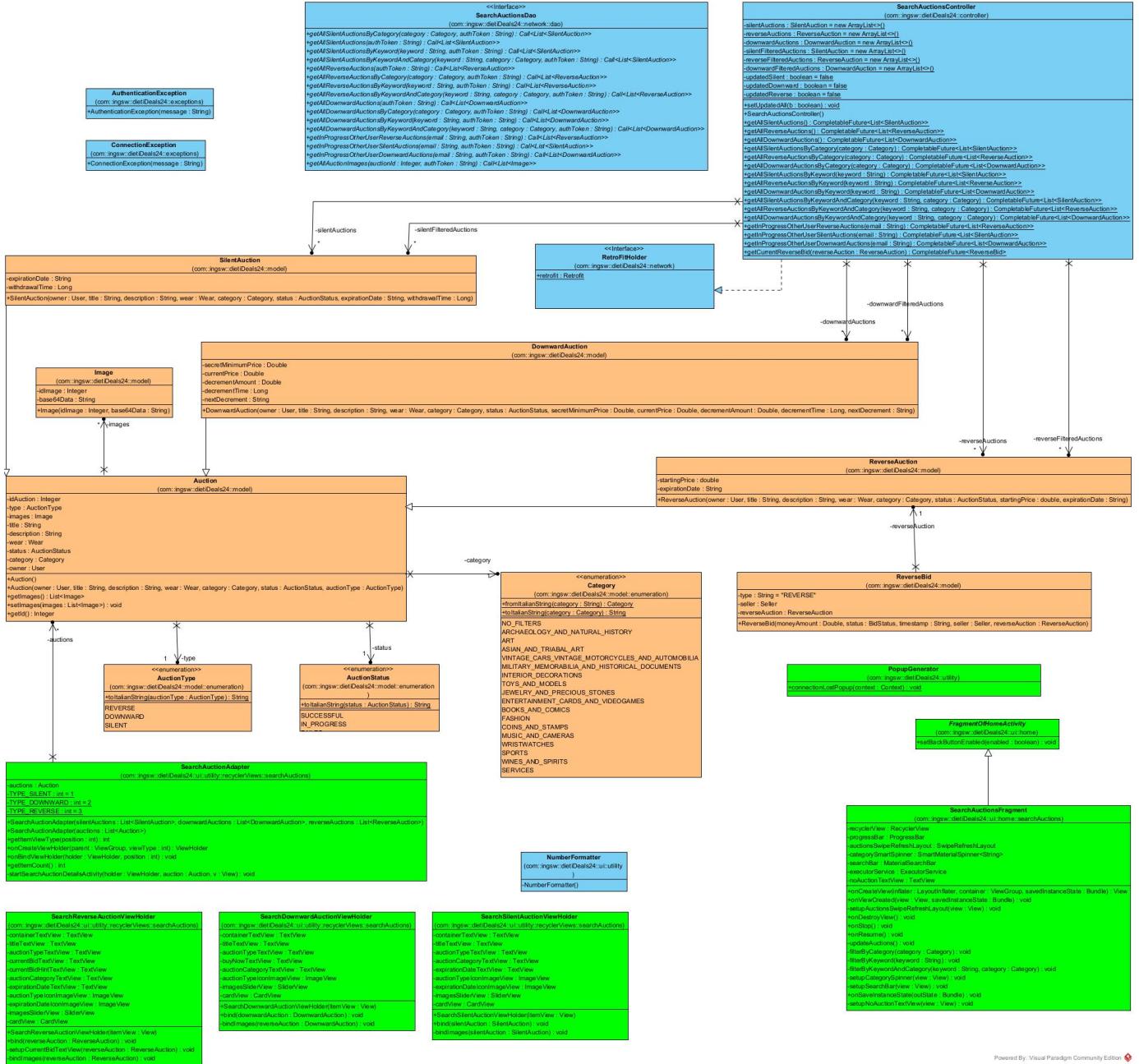
10. Visualizza i dettagli della mia asta



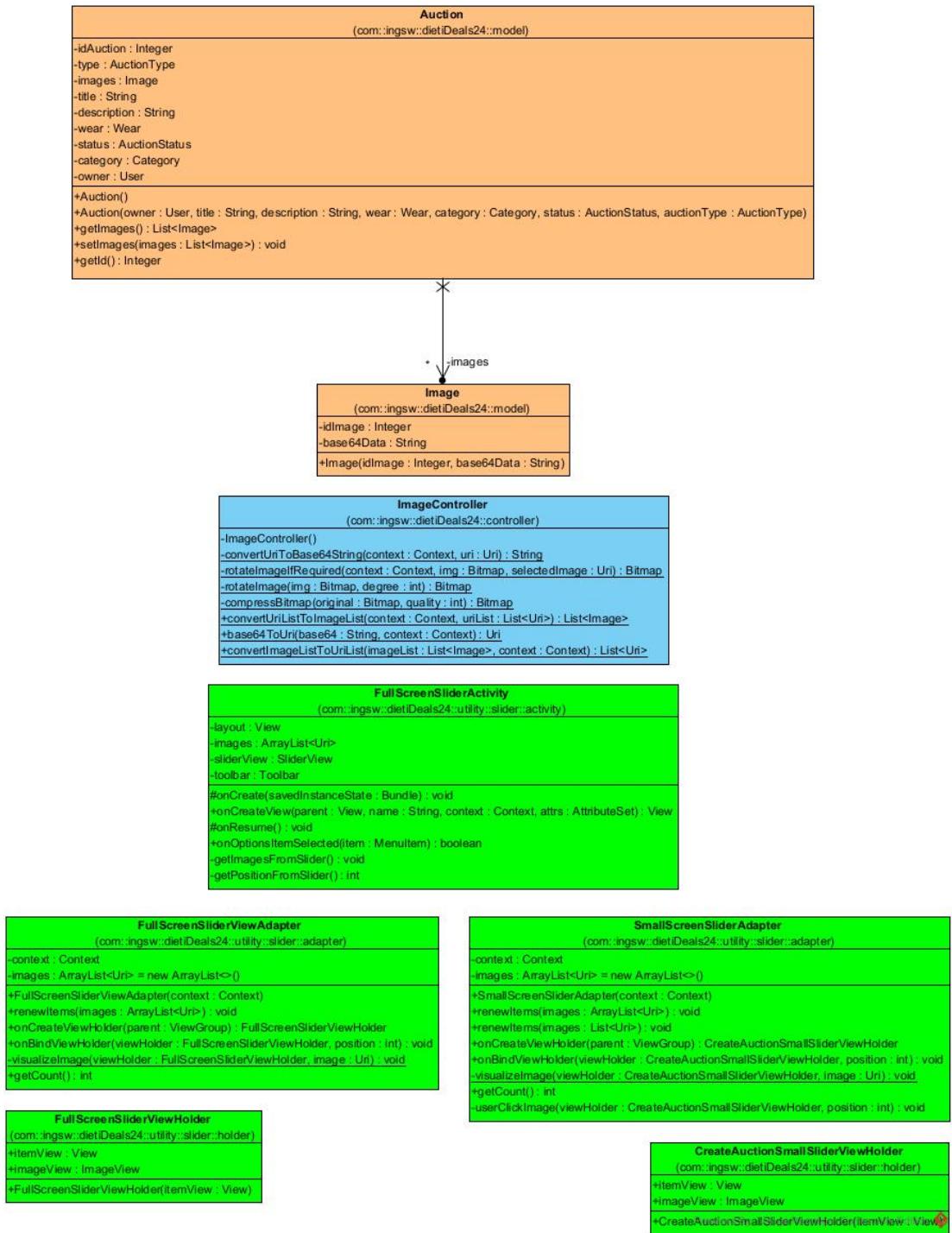
## 11. Visualizza i dettagli dell'asta cercata.



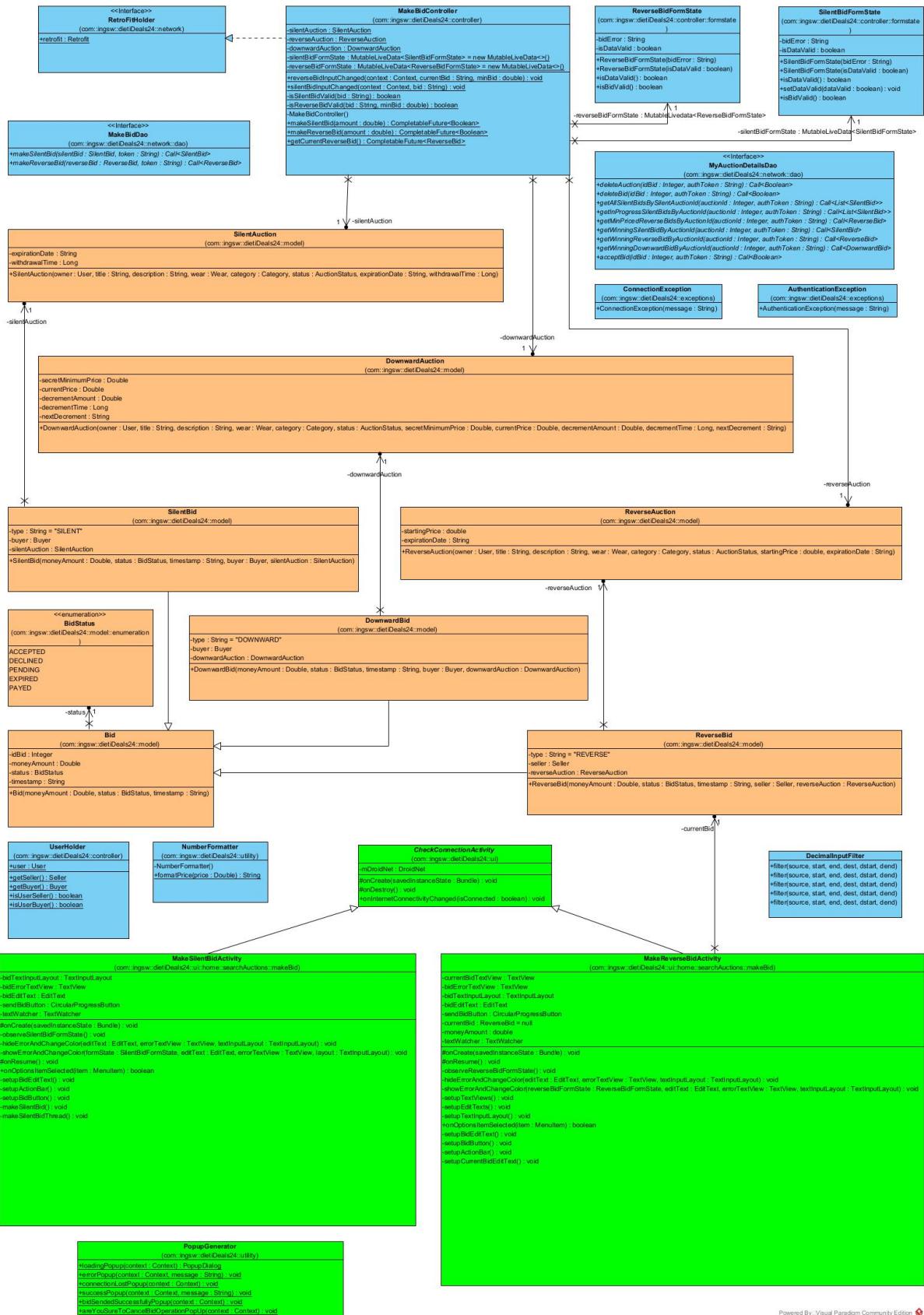
## 12. Ricerca delle aste.



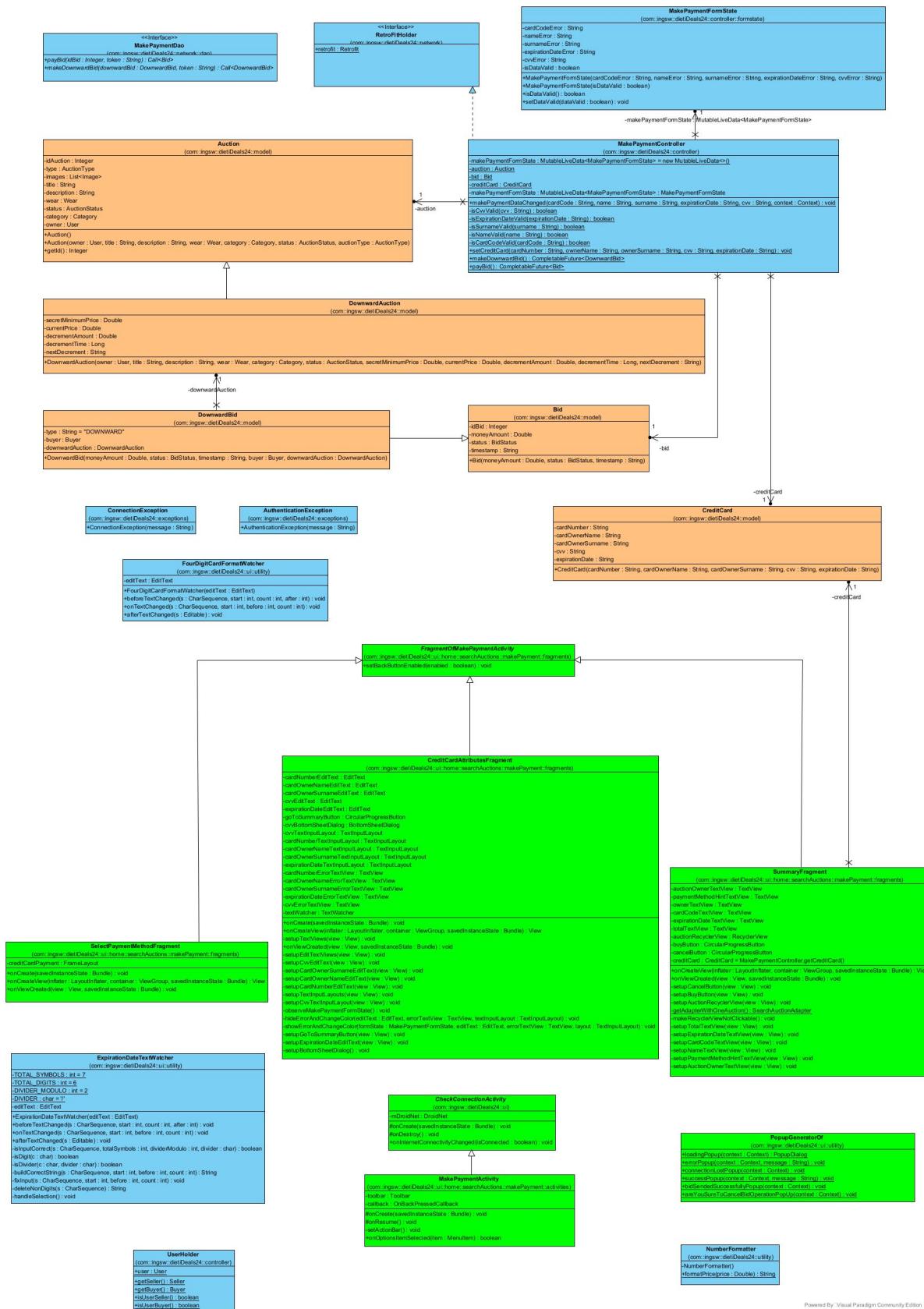
13. Visualizza le immagini dell'asta.



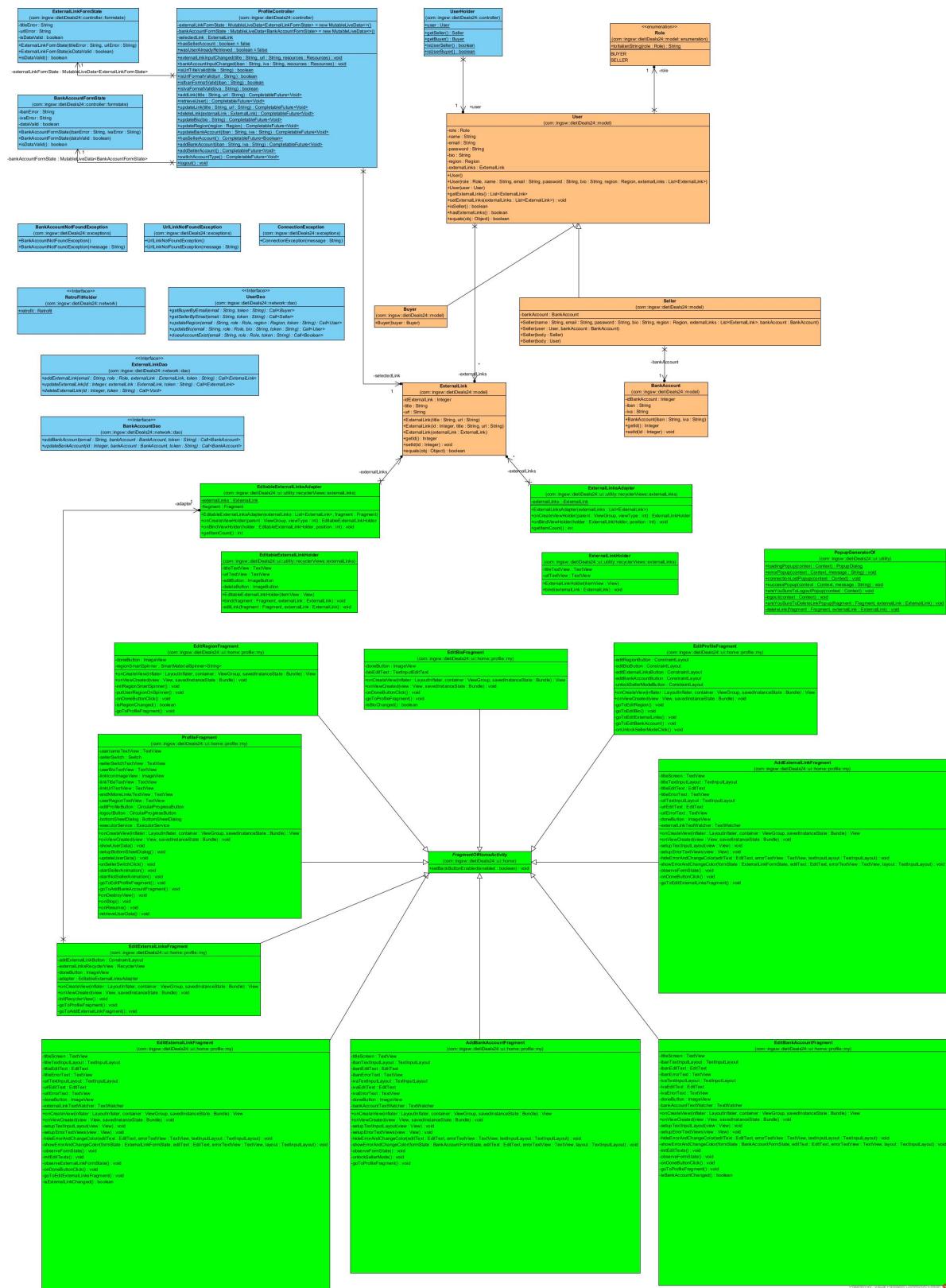
## 14. Invio dell'offerta.



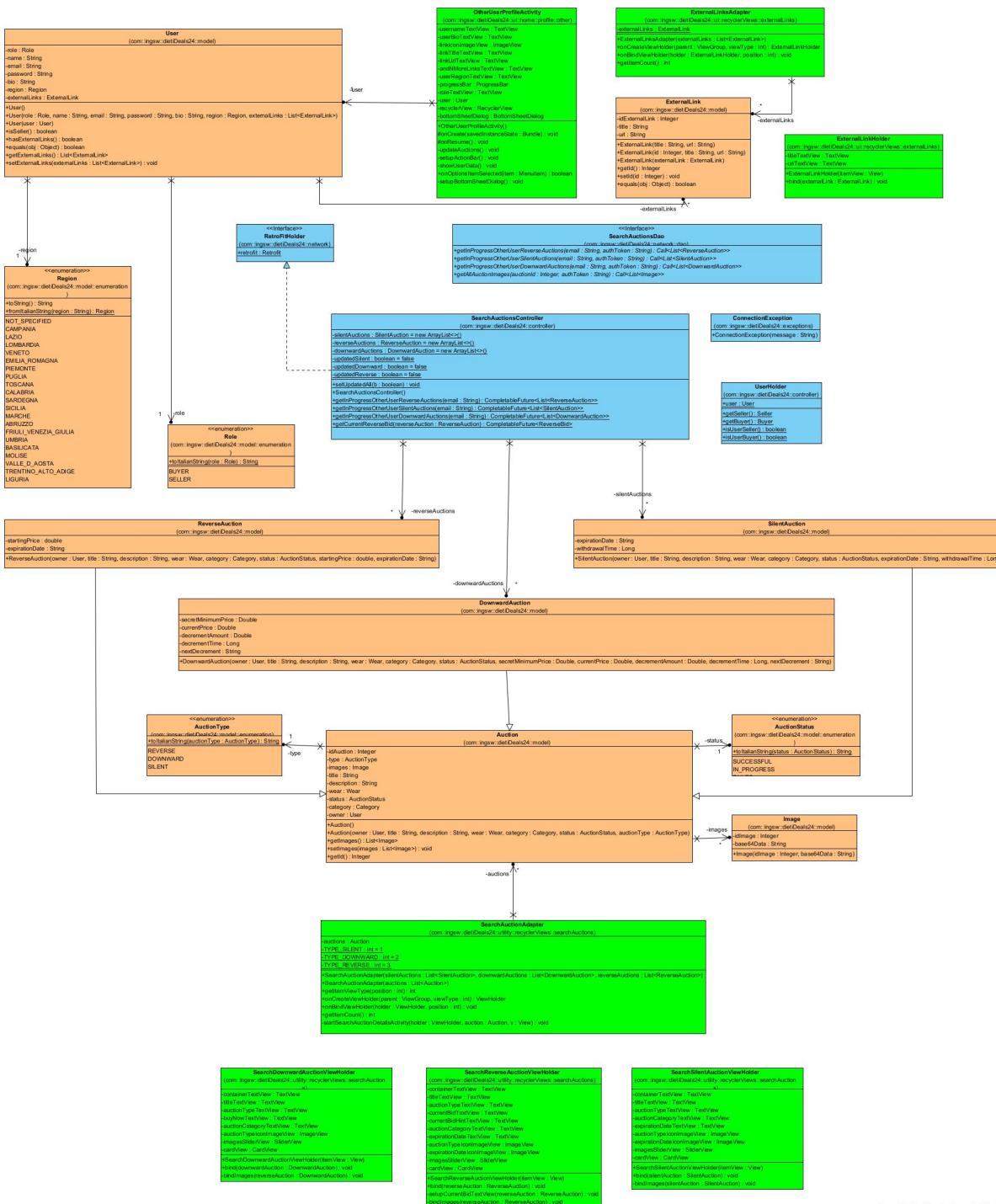
## 15. Pagamento.



## 16. Modifica del profilo.



## 17. Visualizza il profilo altrui.



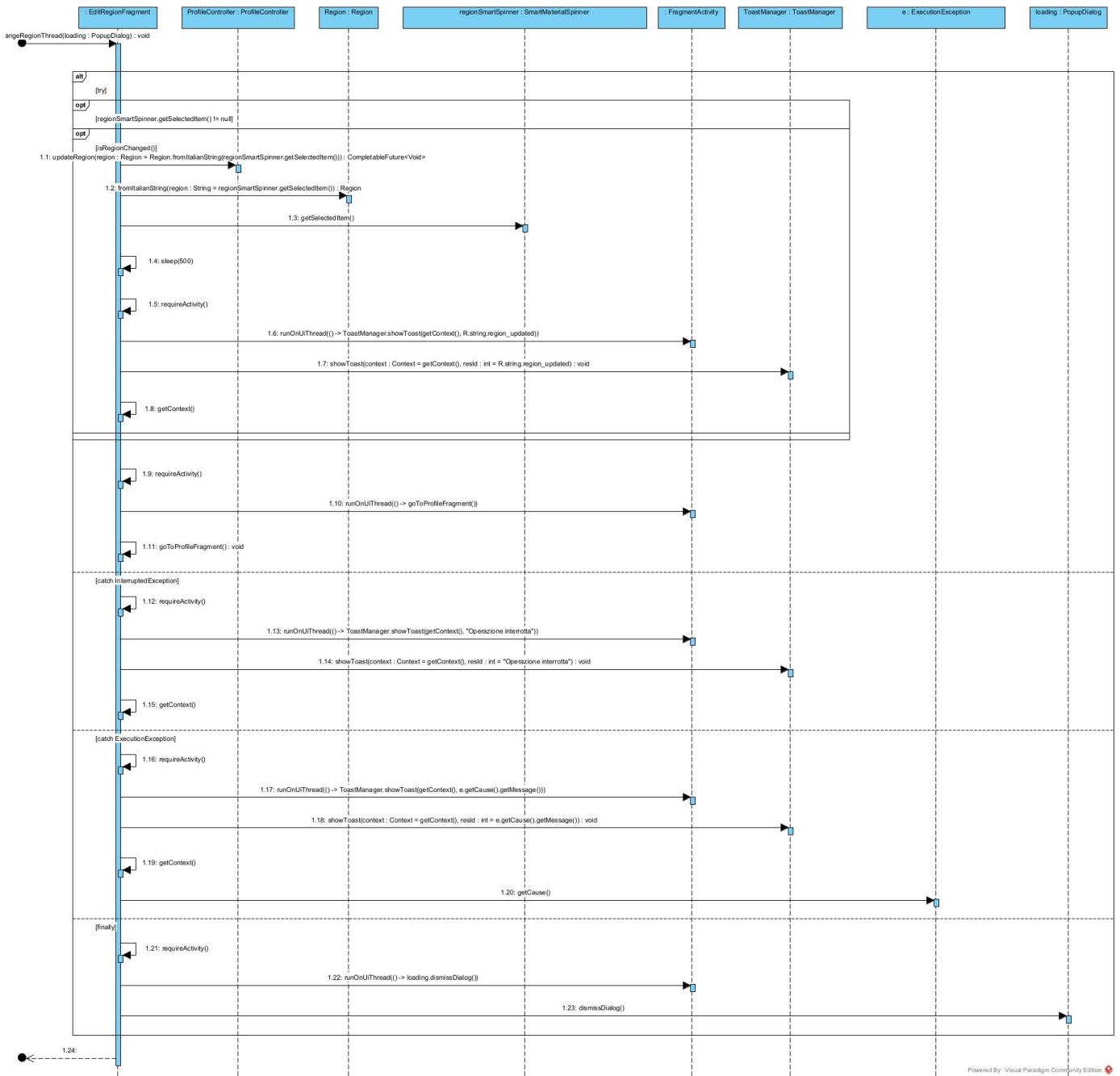
Powered By: Visual Paradigm Community Edition

### 3.3 Sequence diagram di design.

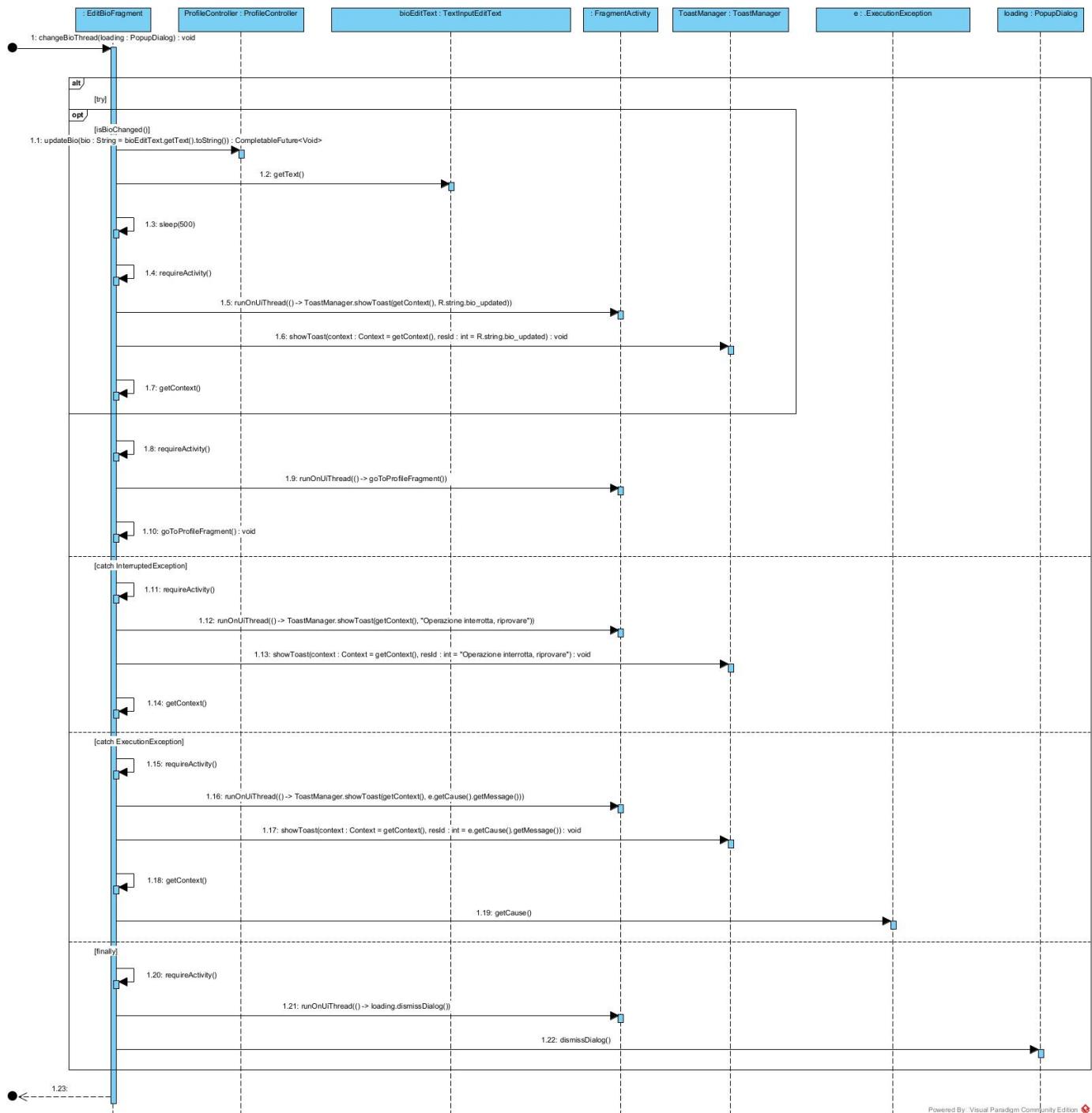
Di seguito mostriamo i [Sequence Diagram](#) di design per i casi d'uso identificati [qui](#).

#### 1. Personalizzazione del profilo.

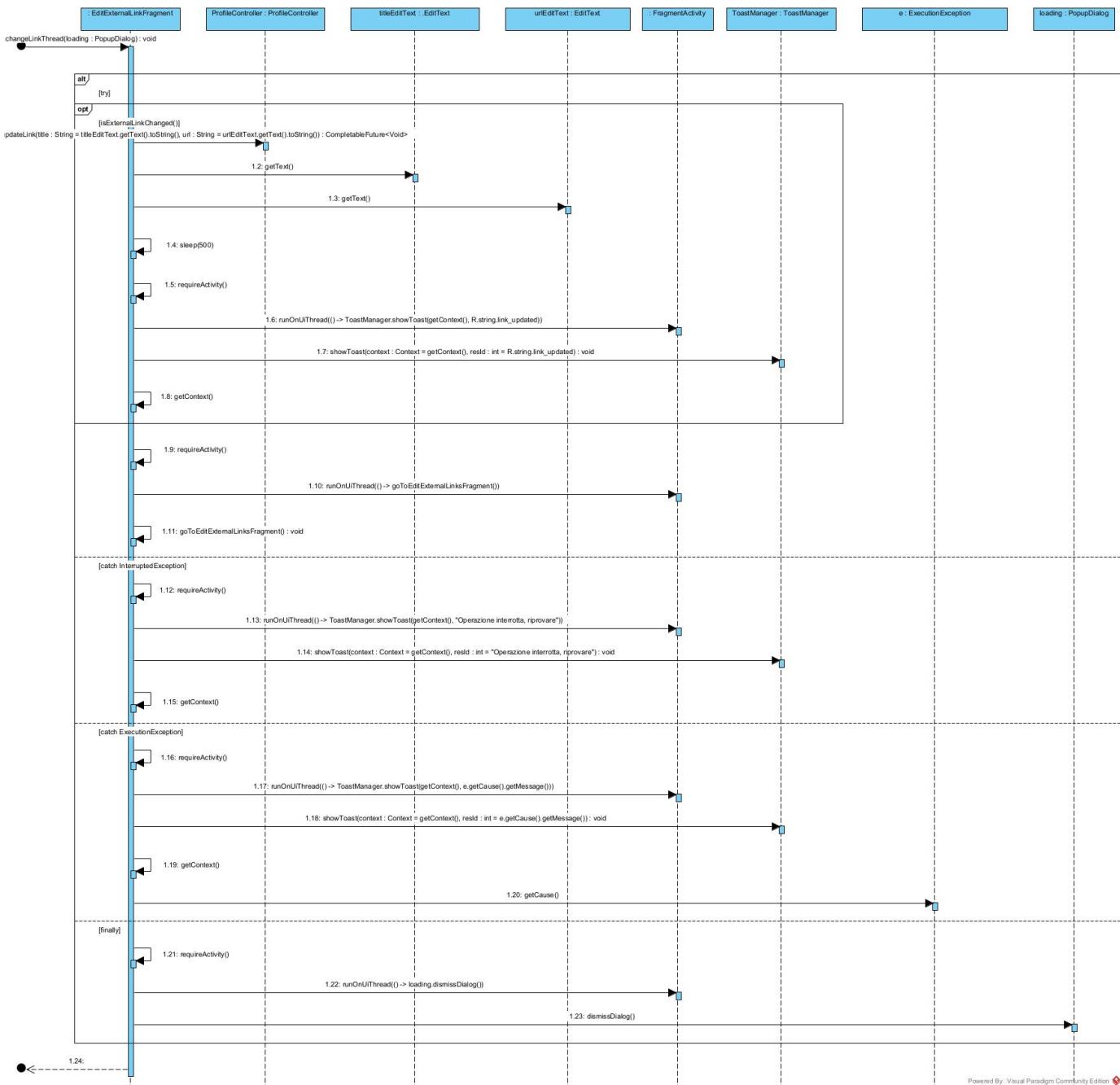
- Personalizza regione.



● Personalizza bio.

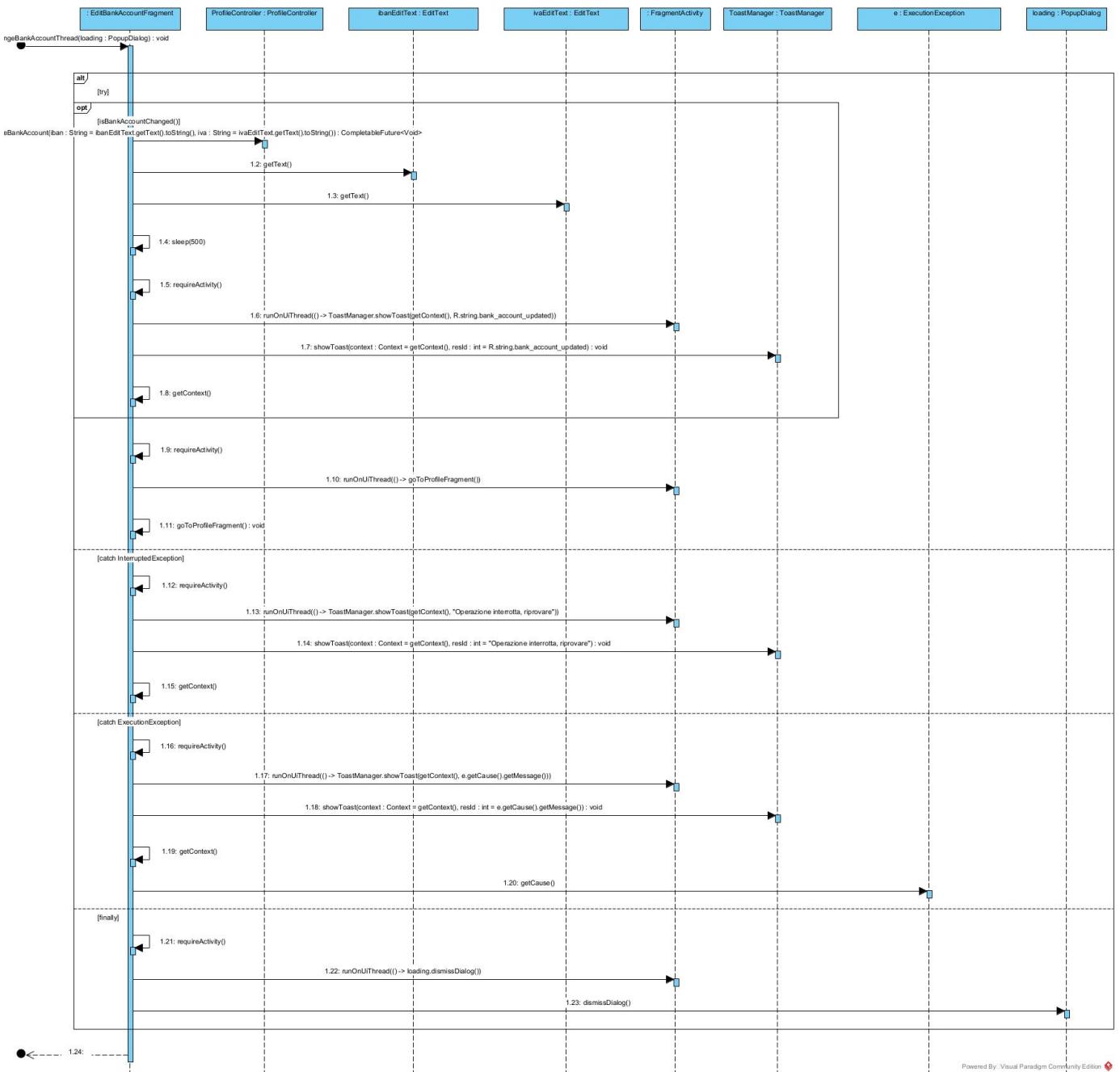


- Personalizza link esterni.



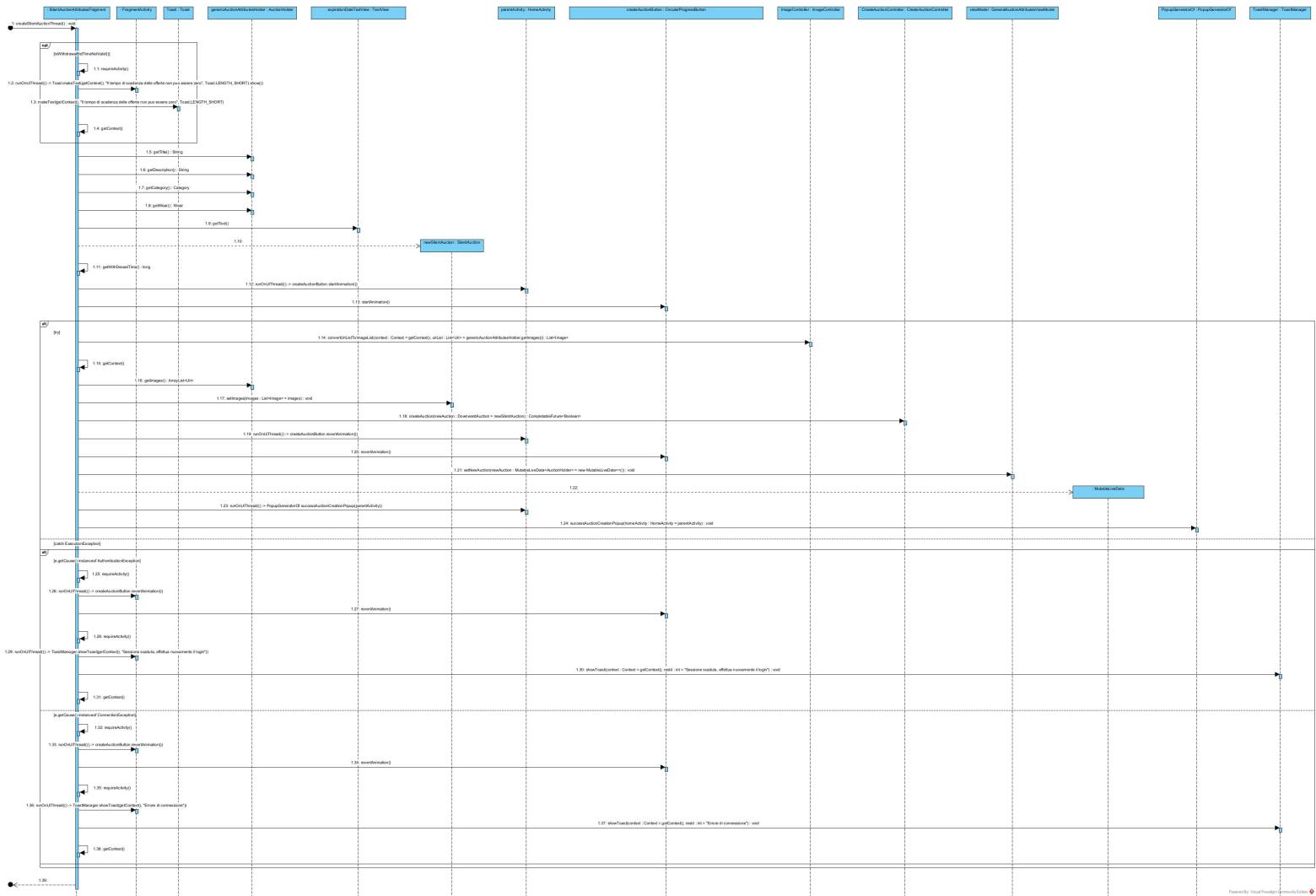
Powered By: Visual Paradigm Community Edition

● Personalizza account bancario.



Powered By: Visual Paradigm Community Edition

## 2. Creazione di un'asta silenziosa.



## 4 Codice sorgente sviluppato.

### 4.1 File di build automatica.

Il Server è stato containerizzato usando Docker. Il Dockerfile crea un'immagine del Server facendo partire l'eseguibile **DietiDeals-server.jar** (file che fa partire il nostro server Spring Boot) all'interno di un ambiente **JDK**.

---

```
# Step 1: Specify the base image
FROM openjdk:17-jdk-alpine

# Step 2: A volume point to /tmp because it is where spring boot creates
#          working directories for Tomcat by default
VOLUME /server

# Step 3: Copy the jar file into the image
COPY dietideals-server.jar /server/dietideals-server.jar

# Step 4 Expose the port for the database
EXPOSE 5432

# Step 5: Expose the port which the server will listen
EXPOSE 8080

# Step 6: Run the jar file
ENTRYPOINT ["java","-jar","/server/dietideals-server.jar"]
```

---

È possibile far partire l'immagine del Server (container di nome dietideals-server) insieme all'immagine del suo database (container di nome dietidealsdb) tramite il docker-compose.yaml. Esso riavvierà il Server automaticamente ogni volta che fallisce.

---

```
version: '3.1'

services:
  db:
    container_name: dietidealsdb
    image: postgres
    restart: always
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: dietidealsdb
    ports:
      - 5432:5432
    networks:
      - dietideals-network
  dietideals-server:
```

```

container_name: dietideals-server
build: .
restart: on-failure
environment:
  - SPRING_PROFILES_ACTIVE=production
ports:
  - 8080:8080
depends_on:
  - db
networks:
  - dietideals-network

networks:
  dietideals-network:
    driver: bridge

```

---

## 4.2 Versioning.

Gli strumenti di versioning sono molto importanti perché consentono di facilitare la collaborazione tra programmatori. Per sviluppare **DietiDeals24** noi abbiamo utilizzato [Git](#) e il servizio di hosting di repository [GitHub](#).

Per accedere al nostro repository fare click [qui](#).

## 4.3 Code quality.

Per avere un resoconto oggettivo sulla qualità del codice scritto per il backend abbiamo utilizzato [SonarQube](#), uno strumento di revisione automatica del codice che aiuta a scrivere codice pulito.



Il backend, come si evince dall'immagine, ha passato il [quality gate](#), lasciando dei problemi. Quello di sicurezza, infatti, è relativo ad un token utilizzato per la generazione del report di SonarQube, mentre quelli di manutenibilità sono relativi a degli attributi di alcune classi, dove era consigliato rendere suddetti attributi transient oppure serializable. Non avendo piena dimestichezza di questi due concetti, abbiamo preferito rimandare le correzioni da apportare, per evitare di incappare in problematiche inattese.

## 5 Testing e valutazione sul campo dell'usabilità.

### 5.1 JUnit testing.

I test sono stati scelti adottando la strategia **Black Box** di tipo **WECT** (Weak Equivalence Class Testing).

Sono stati testati i seguenti metodi:

- **getSilentAuctionsByTitleContainingAndCategory (String keyword, Category category)**  
Metodo della classe **AuctionController** che recupera dal database un insieme di aste silenziose specificando la categoria e la parola chiave che deve essere presente nel titolo.
- **doesAccountExist(String email, Role role)**  
Metodo della classe **UserController** che restituisce un valore booleano se il profilo dell'utente specificato tramite l'email e il ruolo esiste.
- **updateRegion(String email, Role role, Region newRegion)**  
Metodo della classe **UserController** che modifica la regione di un account specificato tramite l'email e il ruolo.
- **updateBio(String email, Role role, String newBio)**  
Metodo della classe **UserController** che modifica la bio di un account specificato tramite l'email e il ruolo.

#### 5.1.1 Classi di equivalenza individuate

Una volta scelti i metodi da testare, abbiamo individuato le seguenti classi d'equivalenza per ogni parametro dei metodi:

##### 1. **getSilentAuctionsByTitleContainingAndCategory**

Le classi d'equivalenza individuate sono le seguenti:

- **String: keyword**
  - "keyword presente nel db"
  - "keyword non presente nel db"
  - ""
  - null
- **Category: category**
  - categoria valida
  - null

##### 2. **doesAccountExist**

Le classi d'equivalenza individuate sono le seguenti:

- **String: email**
  - "email presente nel db"

- "email non presente nel db"
- ""
- null

- **Role: role**

- ruolo valido
- null

**3. updateRegion**

Le classi d'equivalenza individuate sono le seguenti:

- **String: email**

- "email presente nel db"
- "email non presente nel db"
- null
- ""

- **Role: ruolo**

- ruolo valido
- null

- **Region: region**

- regione valida
- null

**4. updateBio**

Le classi d'equivalenza individuate sono le seguenti:

- **String: email**

- "email presente nel db"
- "email non presente nel db"
- ""
- null

- **Role: role**

- ruolo valido
- null

- **String: bio**

- "bio valida"
- ""
- null

### 5.1.2 Codice JUnit - getSilentAuctionsByTitleContainingAndCategory

---

```
@Test  
@Transactional  
void testGetSilentAuctionsByTitleContainingAndCategory_NotEmptyResult() {  
    List<SilentAuction> result =  
        auctionController.getSilentAuctionsByTitleContainingAndCategory("Test",  
            Category.SERVICES).getBody();  
    assertEquals(result, testAuctions);  
}
```

---

```


---

@Test
@Transactional
void testGetSilentAuctionsByTitleContainingAndCategory_EmptyResult() {
    List<SilentAuction> result =
        auctionController.getSilentAuctionsByTitleContainingAndCategory("Nothing",
            Category.ART).getBody();
    assertEquals(result, new ArrayList<>());
}



---

@Test
@Transactional
void testGetSilentAuctionsByTitleContainingAndCategory_NullCategory() {
    List<SilentAuction> result =
        auctionController.getSilentAuctionsByTitleContainingAndCategory("Test",
            null).getBody();
    assertEquals(result, new ArrayList<>());
}



---

@Test
@Transactional
void testGetSilentAuctionsByTitleContainingAndCategory_NullKeyword() {
    List<SilentAuction> result =
        auctionController.getSilentAuctionsByTitleContainingAndCategory(null,
            Category.SERVICES).getBody();
    assertEquals(result, new ArrayList<>());
}



---

@Test
@Transactional
void testGetSilentAuctionsByTitleContainingAndCategory_EmptyKeyword() {
    List<SilentAuction> result =
        auctionController.getSilentAuctionsByTitleContainingAndCategory("",",
            Category.SERVICES).getBody();
    assertEquals(result, testAuctions);
}
```

---

### 5.1.3 Codice JUnit - doesAccountExist

```
@Test
@Transactional
void testDoesAccountExist_ExistingUser() {
    Boolean accountExists =
        userController.doesAccountExist(testBuyer.getEmail(),
            testBuyer.getRole().getBody());
    assertEquals(Boolean.TRUE, accountExists);
}

@Test
@Transactional
void testDoesAccountExist_NotExistingUser() {
    Boolean accountExists =
        userController.doesAccountExist("notExisting@example.com",
            Role.SELLER).getBody();
    assertEquals(Boolean.FALSE, accountExists);
}

@Test
@Transactional
void testDoesAccountExist_NullEmail() {
    Boolean accountExists = userController.doesAccountExist(null,
        Role.SELLER).getBody();
    assertEquals(Boolean.FALSE, accountExists);
}

@Test
@Transactional
void testDoesAccountExist_EmptyEmail() {
    Boolean accountExists = userController.doesAccountExist("", 
        Role.BUYER).getBody();
    assertEquals(Boolean.FALSE, accountExists);
}

@Test
@Transactional
void testDoesAccountExist_NullRole() {
    Boolean accountExists =
        userController.doesAccountExist(testSeller.getEmail(),
            null).getBody();
    assertEquals(Boolean.FALSE, accountExists);
}
```

#### 5.1.4 Codice JUnit - updateRegion

---

```
@Test
@Transactional
void testUpdateRegion_ValidEmail_ValidRole_ValidRegion() {
    Region newRegion = Region.CAMPANIA;
    User updatedUser =
        userController.updateRegion(testSeller.getEmail(),
            testSeller.getRole(), newRegion).getBody();
    assertEquals(newRegion, updatedUser.getRegion());
}
```

---

```
@Test
@Transactional
void testUpdateRegion_ValidEmail_ValidRole_NullRegion() {
    assertNull(userController.updateRegion(testSeller.getEmail(),
        testSeller.getRole(), null).getBody().getRegion());
}
```

---

```
@Test
@Transactional
void testUpdateRegion_ValidEmail_NullRole_ValidRegion() {
    assertNull(userController.updateRegion(testSeller.getEmail(), null,
        Region.CAMPANIA).getBody());
}
```

---

```
@Test
@Transactional
void testUpdateRegion_NullEmail_ValidRole_ValidRegion() {
    assertNull(userController.updateRegion(null, testSeller.getRole(),
        Region.CAMPANIA).getBody());
}
```

---

```
@Test
@Transactional
void testUpdateRegion_EmptyEmail_ValidRole_ValidRegion() {
    assertNull(userController.updateRegion("", testSeller.getRole(),
        Region.CAMPANIA).getBody());
}
```

---

---

```
@Test
@Transactional
void testUpdateRegion_InvalidEmail_ValidRole_ValidRegion() {
    assertNull(userController.updateRegion("invalid@example.com",
        testSeller.getRole(), Region.CAMPANIA).getBody());
}
```

---

### 5.1.5 Codice JUnit - updateBio

---

```
@Test
@Transactional
void testUpdateBio_ValidEmail_ValidRole_ValidBio() {
    String newBio = "test_seller_new_bio";
    User updatedUser = userController.updateBio(testSeller.getEmail(),
        testSeller.getRole(), newBio).getBody();
    assertEquals(newBio, updatedUser.getBio());
}
```

---

---

```
@Test
@Transactional
void testUpdateBio_ValidEmail_ValidRole_EmptyBio() {
    User updatedUser = userController.updateBio(testSeller.getEmail(),
        testSeller.getRole(), "").getBody();
    assertEquals("", updatedUser.getBio());
}
```

---

---

```
@Test
@Transactional
void testUpdateBio_ValidEmail_ValidRole_NullBio() {
    User updatedUser =
        userController.updateBio(testSeller.getEmail(),
            testSeller.getRole(), null).getBody();
    assertEquals("", updatedUser.getBio());
}
```

---

---

```
@Test
@Transactional
void testUpdateBio_ValidEmail_NullRole_ValidBio() {
    String newBio = "test_seller_new_bio";
    User updatedUser = userController.updateBio(testSeller.getEmail(),
        null, newBio).getBody();
    assertNull(updatedUser);
}
```

---

---

```

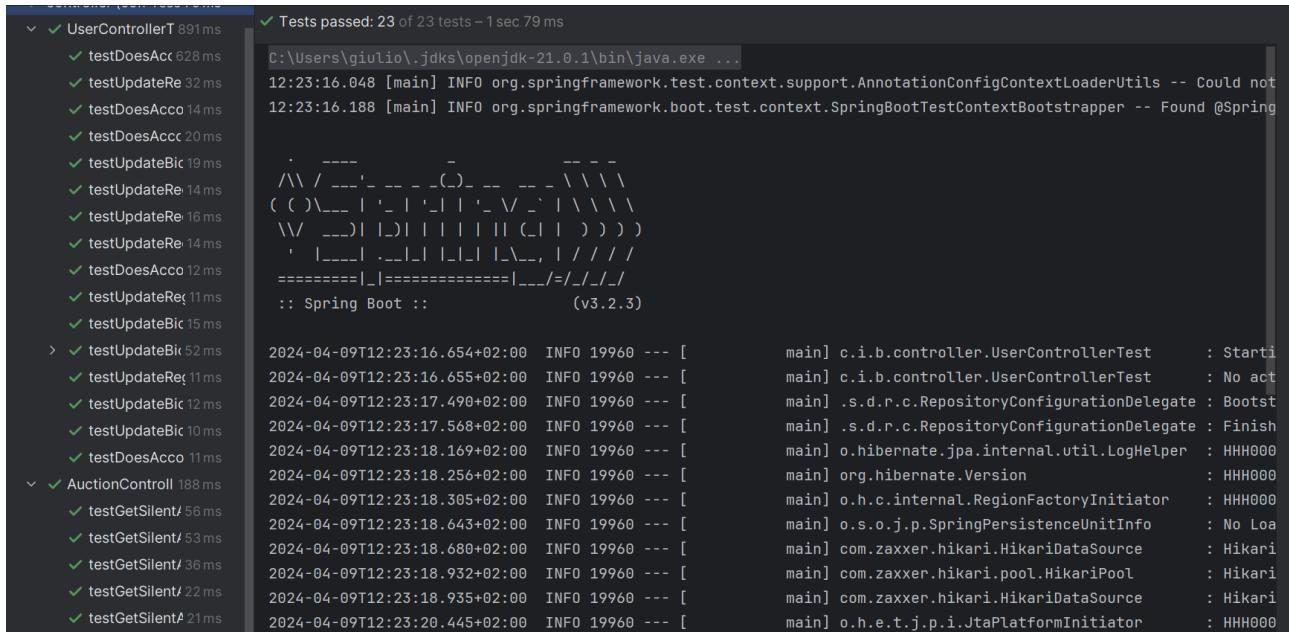
@ParameterizedTest
@Transactional
@NullAndEmptySource
@ValueSource(strings = {"invalid@example.com"})
void testUpdateBio_InvalidEmail_ValidRole_ValidBio(String email) {
    String newBio = "test_seller_new_bio";
    User updatedUser = userController.updateBio(email,
        testSeller.getRole(), newBio).getBody();
    assertNull(updatedUser);
}

```

---

### 5.1.6 Esito dei test

I test precedentemente riportati hanno tutti avuto esito positivo:



```

Tests passed: 23 of 23 tests - 1 sec 79 ms
C:\Users\giulio\.jdks\openjdk-21.0.1\bin\java.exe ...
12:23:16.048 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not
12:23:16.188 [main] INFO org.springframework.boot.test.context.SpringBootTestBootstrapper -- Found @Spring
          .   ---      -      --  -
 / \ / _ _' _ _ _ _ _ _ _ _ _ _ _ _ \ \ \
( ( )\ _ _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ | ' _ |
 \ \ / _ _ )| ( _ )| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
          ' | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ |
          ======|_|=====|_|=====|_|=====|_|=====|_|=====|_|=====|_|=====|_|=====|_|=====|_|=====|_|=====|_|=====|_
          :: Spring Boot ::                               (v3.2.3)

2024-04-09T12:23:16.654+02:00  INFO 19960 --- [           main] c.i.b.controller.UserControllerTest      : Starti
2024-04-09T12:23:16.655+02:00  INFO 19960 --- [           main] c.i.b.controller.UserControllerTest      : No act
2024-04-09T12:23:17.490+02:00  INFO 19960 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootst
2024-04-09T12:23:17.568+02:00  INFO 19960 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finish
2024-04-09T12:23:18.169+02:00  INFO 19960 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHHH000
2024-04-09T12:23:18.256+02:00  INFO 19960 --- [           main] org.hibernate.Version                   : HHHH000
2024-04-09T12:23:18.305+02:00  INFO 19960 --- [           main] o.h.c.internal.RegionFactoryInitiator   : HHHH000
2024-04-09T12:23:18.643+02:00  INFO 19960 --- [           main] o.s.o.j.p.SpringPersistenceUnitInfo     : No Loa
2024-04-09T12:23:18.680+02:00  INFO 19960 --- [           main] com.zaxxer.hikari.HikariDataSource       : Hikari
2024-04-09T12:23:18.932+02:00  INFO 19960 --- [           main] com.zaxxer.hikari.pool.HikariPool         : Hikari
2024-04-09T12:23:18.935+02:00  INFO 19960 --- [           main] com.zaxxer.hikari.HikariDataSource       : Hikari
2024-04-09T12:23:20.445+02:00  INFO 19960 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator      : HHHH000

```

## 5.2 Valutazione dell'usabilità sul campo.

### 5.2.1 Test sul prodotto finito mediante testers umani.

I testers scelti per testare il prodotto finito appartengono alle stesse fasce demografiche dei tester che abbiamo selezionato per valutare i prototipi. La tabella riporta i risultati riscontrati.

**S:** Successo = 1

**P:** Parziale Successo = 0.5

**F:** Fallimento = 0

	TASK 1	TASK 2	TASK 3	TASK 4	TASK 5	TASK 6	TASK 7	TASK 8	TASK 9	TASK 10
TESTER 1 55 anni	P	S	P	S	P	S	S	S	S	S
TESTER 2 64 anni	F	S	F	S	P	S	S	S	S	P
TESTER 3 23 anni	S	S	S	S	S	S	S	S	S	S
TESTER 4 22 anni	S	S	S	S	S	S	S	S	S	S

#### TASKS:

1. Registrati come un venditore
2. Crea un'asta al ribasso di un bene
3. Accetta un'offerta a una tua asta silenziosa
4. Effettua il pagamento di una tua offerta
5. Modifica il profilo aggiungendo un link esterno
6. Visualizza le offerte effettuate
7. Visualizza notifiche
8. Compra un'asta al ribasso
9. Fai un'offerta ad un'asta silenziosa
10. Visualizza il profilo di un creatore di un'asta

#### TESTERS

1. Dario
2. Flavia
5. Eduardo
6. Elisabetta

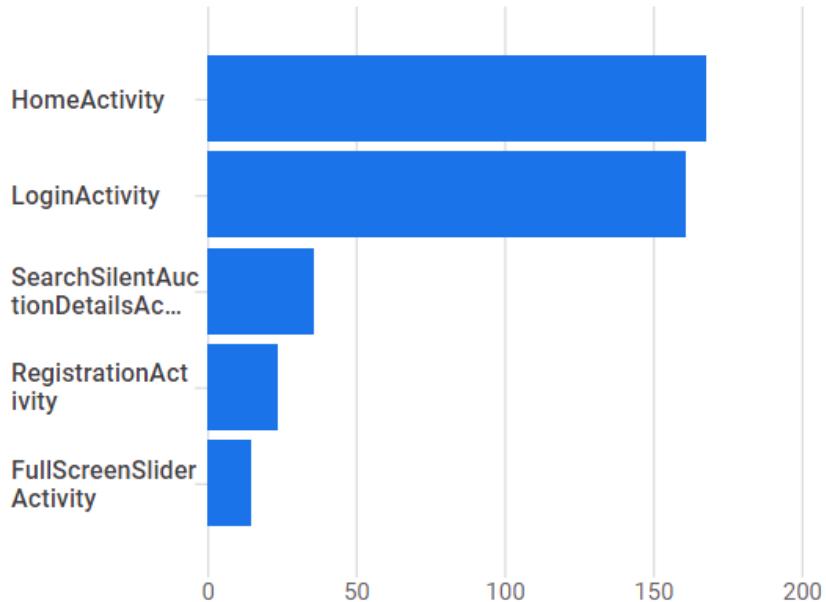
$$\text{Usabilità} = \frac{33 + (0.5 \times 5)}{40} = 0.89 = 89\% \text{ di usabilità}$$

### 5.2.2 Analisi dei file di log di Google Analytics

I log sono stati ottenuti utilizzando lo strumento di analisi [Google Analytics](#). Le seguenti immagini mostrano alcuni dei risultati ottenuti (quelli più interessanti).



### Visualizzazioni per Percorso pagina e classe schermata



Percorso pagina e classe schermata	Visualizzazioni	Utenti	Visualizzazioni per utente	Durata media del coinvolgimento	Conteggio eventi
	100% del totale	100% del totale	Uguale alla media	Uguale alla media	100% del total
1 HomeActivity	463 100% del totale	19 100% del totale	24,37 Uguale alla media	11 m 48 s Uguale alla media	74 100% del total
2 LoginActivity	168	16	10,50	9 m 51 s	29
3 SearchSilentAuctionDetailsActivity	161	19	8,47	2 m 15 s	22
4 RegistrationActivity	36	9	4,00	16 s	3
5 FullScreenSliderActivity	24	11	2,18	1 m 23 s	2
6 SearchReverseAuctionDetailsActivity	15	7	2,14	2 s	1
7 OtherUserProfileActivity	14	5	2,80	10 s	1
8 MakeSilentBidActivity	11	5	2,20	7 s	1
9 InProgressSilentAuctionActivity	10	7	1,43	22 s	1
9 InProgressDownwardAuctionActivity	7	2	3,50	12 s	
10 InProgressDownwardAuctionActivity	4	1	4,00	12 s	

Dai report si possono osservare alcuni dati interessanti:

- La durata media di utilizzo dell'applicazione in una singola sessione (pari a 11 minuti e 48 secondi).
- Il tempo medio di visualizzazione delle singole schermate, dove la schermata di home prende gran parte del tempo di utilizzo totale, dato che contiene i principali fragment. Mentre invece il tempo di utilizzo della schermata di login risulta essere falsato, dato che essa era la schermata che veniva visualizzata da ogni tester mentre gli si spiegava i compiti che doveva tentare di portare a termine.

Per visualizzare il report completo fare click [qui](#).