

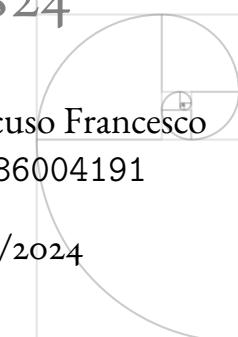
Documentazione del Software

DIETI DEALS24

Tortora Salvatore
N86004033

Terrecuso Francesco
N86004191

Anno accademico 2023/2024



UNIVERSITÀ DEGLI STUDI DI
NAPOLI FEDERICO II

INDICE

1	Introduzione	6
1.1	Struttura della Documentazione	6
2	Documento dei Requisiti	7
2.1	Modello dei Casi d'uso	7
2.1.1	Tabella dei casi d'uso:	8
2.1.2	Descrizione dettagliata dei casi d'uso:	8
2.2	Descrizioni Testuali Strutturate :	12
2.2.1	Tabella Cockburn del primo caso d'uso : Creazione asta incrementale	13
2.2.2	Tabella Cockburn del secondo caso d'uso : Cambio password	14
2.2.3	Tabella Cockburn del terzo caso d'uso : Puntare su un'asta	15
2.2.4	Tabella Cockburn del quarto caso d'uso : Filtraggio delle aste	16
2.3	Prototipazione visuale attraverso Mock-up :	17
2.3.1	Scelte Progettuali :	17
2.3.2	Osservazioni fatte da stakeholder	18
2.3.3	Elenco dei Mock-up :	18
2.4	Target Utenti :	27
2.4.1	Personas Individuate :	27
2.4.2	Analisi dei tratti caratteristici:	29
2.5	Valutazione dell'usabilità a priori :	30
2.5.1	Euristiche di Nielsen :	30
2.5.2	Model Human Processor:	31
2.5.3	Apprendimento degli utenti:	31
2.5.4	Curva di Apprendimento :	32
2.6	Glossario :	33
3	Specifiche dei requisiti	34
3.1	Modello di Dominio	34
3.2	Scelte progettuali	34
3.3	Pensando al Futuro: Progettare per Facilitare le Modifiche del Codice	34
3.3.1	Registrazione al sistema tramite mail, Facebook, Google, Apple.	35

3.3.2	Creazione Asta a tempo fisso.	36
3.3.3	Creazione Asta Incrementale.	37
3.3.4	Mostra storico delle aste create.	38
3.3.5	Mostra aste correttamente attive nel sistema con ricerca e filtro.	39
3.3.6	Personalizzazione del profilo.	40
3.3.7	Cambio password.	41
3.3.8	Login al sistema.	42
3.3.9	Visualizzazione delle notifiche.	43
3.3.10	Mostra profilo del venditore di un'asta specifica.	44
3.3.11	Mostra aste a cui si sta partecipando ed aste vinte.	45
3.3.12	Puntata su Asta a tempo fisso.	46
3.3.13	Puntata su Asta Incrementale.	47
3.4	Sequence Diagram di due casi d'uso	48
3.4.1	Caso d'uso : Personalizzazione del profilo	48
3.4.2	Caso d'uso : Puntare su un'asta a tempo fisso	50
3.5	Statechart	51
3.5.1	Statechart I : Cambio Password	51
3.5.2	Statechart II : Creazione Asta Incrementale	52
3.5.3	Statechart III : Puntare su un'Asta Incrementale	53
3.5.4	Statechart IV : Filtraggio Aste	54
4	Documento di design	55
4.1	Analisi dell'architettura	55
4.1.1	Client	55
4.1.2	Server	56
4.2	Scelte tecnologiche adottate :	57
4.2.1	Servizi Cloud usati:	57
4.2.2	Altri servizi esterni usati:	57
4.3	Diagrammi UML delle classi di Design	58
4.3.1	Registrazione al sistema tramite mail, Facebook, Google, Apple e Login al sistema .	58
4.3.2	Creazione Asta a Tempo fisso	59
4.3.3	Creazione Asta Incrementale	60
4.3.4	Mostra storico Aste Create	61
4.3.5	Mostra aste correttamente attive nel sistema con ricerca e filtro	62

4.3.6	Personalizzazione del Profilo e cambio password	63
4.3.7	Mostra profilo venditore di un'asta specifica	64
4.3.8	Visualizzazione delle notifiche	65
4.3.9	Punta su un'Asta (Incrementale/Tempo fisso)	66
4.4	Diagrammi UML del Server	68
4.4.1	Model	68
4.5	Diagrammi UML di Sequenza	69
4.5.1	Caso d'uso: Personalizzazione del profilo	69
4.5.2	Caso d'uso : Puntare su un'asta a tempo fisso	70
5	DockerFile e file di build Automatica	71
5.1	Dockerfile	71
5.2	Docker Compose	71
6	Qualità del Codice	73
6.1	SonarQube	73
6.1.1	Indice di Duplicazione:	74
6.1.2	Indice degli Errori:	74
6.1.3	Report di qualità dell'ultima modifica	75
6.1.4	SonarLint	75
7	Strumento di Versioning	76
7.0.1	Repository Front-End:	76
7.0.2	Repository Back-End:	77
8	Testing	78
8.1	Utilizzo di XCTest	78
8.1.1	checkEmailAndPassword	78
8.1.2	SignUp	81
8.1.3	isOfferValid	85
8.1.4	changePassword	87
9	Studio dell'usabilità sul campo	90
9.1	Test di compito	91
9.2	Test di scenario	91

9.3	Questionario di valutazione	93
10	Monitoraggio del software	94
10.1	Caratteristiche principali di Firebase Analytics	94
10.2	Implementazione e utilizzo	94
10.3	Analisi dei dati	94
10.3.1	Grafico Coinvolgimento Utenti:	95
10.3.2	Grafico Nazionalità Utenti:	95
10.3.3	Grafico Attività Utente nel tempo:	96
10.4	Monitoraggio del server	97
10.4.1	Rete in Ingresso e Rete in Uscita (byte):	97
10.4.2	Utilizzo Medio Della CPU:	97
10.5	Monitoraggio dell'applicazione	98
10.5.1	Grafico utilizzo CPU:	98
10.5.2	Grafico dell'utilizzo di RAM:	99

Questa pagina è stata volutamente lasciata bianca

INTRODUZIONE

DietiDeals24 è un sofisticato sistema software progettato per la gestione di aste online, con l'obiettivo di semplificare il processo di vendita e acquisto di beni o servizi. L'applicazione offre un'interfaccia user-friendly che facilita l'interazione tra gli utenti, migliorando complessivamente l'esperienza di chi partecipa alle aste.

I.I STRUTTURA DELLA DOCUMENTAZIONE

Il seguente documento è suddiviso in tre sezioni:

- **Documento dei requisiti del Software** : in questa prima fase si studia a fondo la richiesta del cliente elaborando con esatta precisione tutti i requisiti richiesti per la realizzazione del software; Difatti rappresenta la parte più importante del processo.
- **Documento di Design del Sistema** : questa sezione descrive e giustifica tutte le scelte tecnologiche prese per la realizzazione del sistema.
- **Testing e valutazione dell'usabilità** : spiega in che modo il sistema finito è stato testato e ne fa uno studio dal punto di vista dell'usabilità.

DOCUMENTO DEI REQUISITI

Per garantire il successo di DietiDeals24, è fondamentale comprendere a fondo i requisiti del sistema. L'analisi dei requisiti si concentrerà su:

- Identificazione chiara degli attori principali del sistema, compresi venditori e acquirenti.
- Definizione dettagliata delle funzionalità chiave degli attori, come la creazione, partecipazione e organizzazione di aste
- Definizione del target degli utenti
- Prototipazione dell'applicazione via Mock-up dell'interfaccia utente
- Studio dell'usabilità a priori

Questo processo di analisi approfondita ci consentirà di sviluppare un sistema robusto e adatto alle esigenze specifiche degli utenti di DietiDeals24.

2.I MODELLO DEI CASI D'USO

Lo scopo del modello dei casi d'uso è quello di delineare in modo preciso ma semplice e schematico l'insieme di funzionalità e gli attori che entrano in gioco. Attraverso UML, la visualizzazione dei diagrammi sarà di facile comprensione.

Ulteriore compito di questi modelli sta nello stabilire il confine del sistema da sviluppare, descrivendo tutte quelle funzionalità che il sistema *deve* e *non deve* implementare. (*System Under Development*).

Nella fase iniziale, è fondamentale identificare gli attori coinvolti nel sistema e tutte le funzionalità che il sistema deve garantire. Durante questa fase preliminare, si è proceduto a schematizzare in una tabella tutti gli attori e i rispettivi casi d'uso.

Nella pagina seguente è illustrata la tabella dei casi d'uso e i relativi attori.

2.1.1 Tabella dei casi d'uso:

Attore	Casi d'uso
Utente non registrato	<ul style="list-style-type: none"> – registrazione al sistema tramite mail – registrazione al sistema tramite Google, Apple o Facebook
Venditore	<ul style="list-style-type: none"> – Crea Asta a tempo fisso – Crea Asta all'inglese – Visualizza aste create (storico)
Compratore	<ul style="list-style-type: none"> – Presenta offerta per asta a tempo fisso. – Presenta offerta per asta all'inglese. – Visualizza aste a cui sta partecipando e aste vinte.
compratore e venditore	<ul style="list-style-type: none"> – Visualizzare notifica asta conclusa. – Personalizza profilo personale. – Visualizza aste e dettagli asta – Filtra aste per categoria e/o parola chiave e prezzo – Visualizza profilo venditore di un'asta – Cambio di password

2.1.2 Descrizione dettagliata dei casi d'uso:

Le seguenti descrizioni sono volte a estrarre tutte le richieste dell'utente molto generali in requisiti di sistema più dettagliati e precisi.

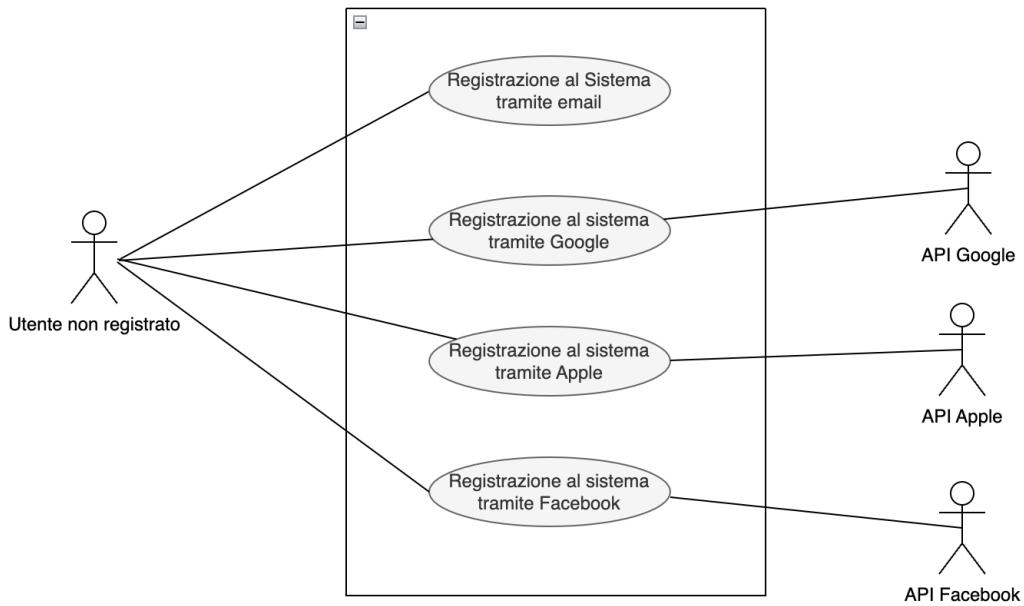
- **Registrazione al sistema tramite email :** un utente non registrato ha la possibilità di registrarsi al sistema inserendo alcuni dati quali email, password e numero di telefono.
- **Crea Asta a tempo fisso :** il sistema permette ad un utente di tipo venditore la creazione di un'asta a tempo fisso: l'utente deve inserire una data di scadenza dell'asta, e optionalmente inserire una soglia minima segreta non visibile ai compratori. Quando termina la data di scadenza abbiamo due scenari: la soglia non è stata raggiunta quindi l'asta fallisce. La soglia è stata superata, dunque ha successo. In entrambi i casi chi ha partecipato all'asta riceve una notifica.
- **Crea Asta all'Inglese :** il sistema permette ad un utente di tipo venditore di creare un asta all'inglese. In particolare l'utente deve inserire i dettagli dell'asta quindi foto, descrizione base d'asta pubblica.

Opzionalmente può modificare il tempo fissato per la presentazione di nuove offerte che di default viene settato a un'ora ed una base di rialzo (di default di 10 euro); il timer viene resettato ogni qualvolta un utente presenta un'offerta. Quando questo tempo scade (arriva a 0 secondi) vi sono due scenari : nessuno ha presentato un'offerta, dunque l'asta fallisce; l'asta a seguito dell'ultima offerta presentata ha esito positivo. Al termine tutti quelli che hanno partecipato all'asta, ricevono una notifica.

- **Visualizza storico delle Aste Create :** il sistema permette ad un utente di tipo venditore di visualizzare la lista e i dettagli delle aste che ha creato.
- **Punta su asta a tempo fisso :** un utente di tipo compratore ha la possibilità di presentare un'offerta migliorativa rispetto al prezzo corrente di un'asta a tempo fisso.
- **Punta su asta all'inglese :** un utente di tipo compratore ha la possibilità di presentare un'offerta migliorativa rispetto al prezzo corrente di un'asta all'inglese in base alla soglia di rialzo che il venditore dell'asta ha stabilito a monte.
- **Visualizza aste a cui sta partecipando e visualizza aste vinte :** il sistema permette ad un account di tipo compratore di visualizzare le aste a cui sta partecipando correttamente attive e tutte le aste in cui ha partecipato in passato ed ha vinto.
- **Visualizza notifica asta conclusa :** nel momento in cui un'asta termina, a tutti gli utenti coinvolti, ovvero tutti coloro che hanno presentato un'offerta e l'organizzatore dell'asta, devono poter visualizzare una notifica che li avvisa del termine dell'asta, conclusa in modo positivo o meno.
- **Personalizza profilo personale :** tutti gli utenti che hanno accesso al sistema hanno un profilo personale personalizzabile con foto, short-bio, links ai propri siti web, ai propri social, area geografica, numero di telefono e la propria mail.
- **Visualizza aste e dettagli asta :** il sistema permette agli utenti registrati di visualizzare le aste attive e i dettagli di queste ultime come ad esempio la foto, la descrizione, il luogo, il tipo d'asta e l'offerta migliore fino a quel momento.
- **Filtra aste per categoria e/o parola chiave e prezzo :** il sistema permette all'utente attraverso una barra di ricerca, di trovare delle aste specifiche partendo da parole chiave oppure anche filtrando per prezzo; permette inoltre di accedere alle categorie di aste presenti nel sistema, così da filtrare e fare una ricerca su un set di dati più ristretto.
- **Visualizza profilo venditore di un'asta :** tutti gli utenti presenti nel sistema possono, dopo aver visualizzato dettagli dell'asta cliccare sul nome dell'utente che l'ha organizzata. In questo modo visualizzerà la foto, la bio, i link associati a quell'utente. Ciò è di vitale importanza per il software in quanto il suo scopo è, oltre alla gestione dello sviluppo di aste, anche quello di mettere in contatto il venditore dell'asta con il vincitore.

- **Cambia password per accedere al sistema :** Il sistema deve garantire agli utenti il cambio della password con cui si sono registrati. Quest'ultimo viene fatto attraverso l'inserimento di una nuova password diversa da quella impostata precedentemente.





2.2 DESCRIZIONI TESTUALI STRUTTURATE :

In questa sezione analizzeremo alcuni casi d'uso in maniera dettagliata, esprimendo in che modo vengono gestiti dal sistema e dall'interazione con l'utente;

Lo stile deciso per la stesura di tali documenti è attraverso le tabelle di *Alistair Cockburn* descritte ampiamente in un suo noto libro *Writing Effective Use Cases*. Tali tabelle sono state però semplificate eliminando alcune sezioni dal template originale.

La traccia ha richiesto esplicitamente la descrizione di quattro casi d'uso non banali escludendo dunque l'autenticazione al sistema. Durante questo processo per aiutarci idealmente su come costruire un modello di interazione fra utente e il sistema, in un primo momento ci siamo aiutati ideando delle interfacce tramite mock-up fatti su carta.

2.2.1 Tabella Cockburn del primo caso d'uso : Creazione asta incrementale

USE CASE #01 CREA ASTA ALL'INGLESE			
DESCRIPTION	Step n°	Venditore	Sistema
Goal L'utente venditore vuole creare un asta incrementale Preconditions L'utente venditore ha effettuato l'accesso Success End Condition L'utente venditore crea correttamente un asta accessibile agli altri utenti registrati	1	Preme il pulsante "+" nel mokup "home"	
	2		Mostra il mokup "banner" con 2 possibili scelte: English e Fixed-time
	3	Preme il pulsante "English"	
	4		Mostra il mokup "create english auction"
	5	Inserisce tutti i dati e preme il pulsante "create auction"	
	6		Torna alla home page e mostra il banner che dice "auction created successfully"
	7	Preme il pulsante "ok!"	
EXTENSIONS	Step n°	Venditore	Sistema
Preme il pulsante "fixed-time" nel mokup "banner"	3.a	Preme pulsante Fixed-time	
	3.b		Mostra il mokup "create fixed-time auction"
	3.c	Preme il pulsante "Home" in alto a sinistra	
	3.d		Mostra il mokup "home" e riprende dallo step n°1
Non inserisce tutti i dati nel mokup "create english auction"	5.a	Legge l'header del pulsante che dice "Enter all missing data to continue"	
	5.b	riprende dallo step 5	

2.2.2 Tabella Cockburn del secondo caso d'uso : Cambio password

USE CASE #02	NOME UC		
DESCRIPTION	Step n°	Attore 1	Sistema
	1	Preme il pulsante "settings" in un qualsiasi mokup della sezione "Home"	
	2		Mostra il mokup "settings"
	3	Preme il pulsante "change password"	
	4		Mostra il mokup "changePassword"
	5	Inserisce correttamente la vecchia password, la nuova password e preme il pulsante "save"	
	6		<i>Torna al mokup Login</i>
EXTENSIONS	Step n°	Attore 1	Sistema
Preme un altro pulsante del menù in basso	1.a	Preme un pulsante vicino al pulsante "settings"	
	1.b		<i>Mostra il mokup corrispondente</i>
	1.c	Riparte dallo Step n'1	
Preme un pulsante vicino a "change password"	3.a	Preme il pulsante "info auctions"	
	3.b		<i>Mostra il mokup corrispondente</i>
	3.c	Preme il pulsante "close"	
	3.d		<i>Ritorna al mokup "settings"</i>
	3.e	Riparte dallo Step n'3	
Inserisce la nuova password in modo errato	5.1.a	Inserisce la nuova password ma non è uguale alla conferma password	
	5.1.b	Legge l'header del pulsante che dice "If you cannot save, check that the password is the same as the confirmed one"	
	5.1.c	Continua dallo Step n°5	
Inserisce la vecchia password in modo errato	5.2.a	Inserisce la vecchia password in modo errato	
	5.2.b		<i>Mostra un banner che dice "the old password is incorrect"</i>

	5.2.c	Preme "ok" e continua dallo step n°5	
La nuova password è uguale alla vecchia password	5.3.a		<i>Mostra un banner che dice "the new password is the same as the old one"</i>
	5.3.b	Preme il pulsante "ok"	
	5.3.c		<i>Continua dallo step n°7</i>

2.2.3 Tabella Cockburn del terzo caso d'uso : Puntare su un'asta

USE CASE #03 PUNTA SU ASTA			
DESCRIPTION	Step n°	Utente	Sistema
	1	Preme il pulsante "Home" se non si trova già lì	
	2		<i>Mostra tutte le aste correntemente attive</i>
	3	Clicca l'asta su cui vuole fare una puntata	
	4		<i>Mostra la vista di dettaglio dell'asta</i>
	5	Preme su "offer more"	
	6		<i>Mostra una vista con una casella di testo in cui inserire l'importo</i>
	7	Se necessario inserisce un valore della puntata corretto e poi preme il bottone "offer"	
EXTENSIONS	Step n°	Utente	Sistema
Preme il pulsante sbagliato	1.a	Preme il pulsante "my auction" che si trova vicino a "home"	
	1.b		<i>Mostra la vista "my auction"</i>
	1.c	Preme il pulsante "home" e continua dallo step n° 2	
Inserisce un importo non valido	7.a	Inserisce un importo che è minore della puntata attuale	
	7.b		<i>Rende il bottone "continue" non cliccabile</i>
	7.c	Continua dallo step n° 7	

2.2.4 Tabella Cockburn del quarto caso d'uso : Filtraggio delle aste

USE CASE #04 FILTRA ASTE PER PAROLA CHIAVE, CATEGORIA E PREZZO			
Goal in Context	L'utente filtra le aste per nome, categoria e prezzo		
Preconditions	L'utente ha effettuato il login		
Success End Condition	L'utente riesce a filtrare le aste in modo da ottenere i risultati sperati		
DESCRIPTION	Step n°	Utente	Sistema
	1	Clicca sul bottone "home" se non si trova già lì	
	2		Mostra la schermata "home"
	3	Clicca sul bottone filtro in alto a sinistra	
	4		Mostra la vista di filtraggio
	5	Seleziona categoria, prezzo, scrive il nome dell'oggetto/servizio a cui è interessato e preme invio	
	6		<i>Mostra a schermo tutte le aste correntemente attive correlate ai dati di ricerca nella home.</i>
EXTENSIONS	Step n°	Utente	Sistema
Clicca sul bottone sbagliato	1.a	Clicca sul bottone "my auction" che si trova vicino a "home"	
	1.b		<i>Mostra la schermata home</i>
	1.c	Clicca sul bottone "home" e riparte dallo step n°1	

2.3 PROTOTIPAZIONE VISUALE ATTRAVERSO MOCK-UP :

In questa sezione, esploreremo i Mock-up creati per illustrare il flusso delle principali funzionalità di DietiDeals24. Questi mockup non solo forniscono una panoramica visiva delle schermate dell'applicazione, ma fungono anche da strumento per raccogliere feedback e riflessioni iniziali sugli aspetti visivi e funzionali dell'interfaccia.

2.3.1 Scelte Progettuali :

Proprio come la spirale aurea che si trova nel frontespizio, ideare e progettare un sistema software vuol dire studiare nei minimi dettagli l'impostazione della GUI, al fine di renderla armoniosa e funzionale in quanto essa rappresenta l'unica interazione *utente-sistema*. La scelta di progettare per dispositivi iOS segue principi di design coerenti con l'ecosistema Apple, garantendo una familiarità all'utente e sfruttando le linee guida dell'interfaccia utente di iOS. Ciò assicura che l'applicazione si integri perfettamente nell'ambiente operativo degli utenti Apple, fornendo un'esperienza fluida e coerente.

- **Le scelte cromatiche:** sono state attentamente selezionate per garantire un'armonia visiva e una leggibilità ottimale. L'uso di colori accattivanti e intuitivi contribuisce a evidenziare le diverse sezioni dell'applicazione e a guidare gli utenti attraverso il flusso delle attività.
- **l'Allineamento :** l'organizzazione degli elementi all'interno di ciascuna schermata è stata progettata per massimizzare l'usabilità. L'accesso rapido alle funzionalità principali, la disposizione logica degli elementi e la chiarezza delle informazioni sono stati prioritari durante il processo di progettazione.
- **Eyetracking :** Abbiamo scelto di posizionare i pulsanti principali nella parte inferiore del frame, prendendo in considerazione le analisi di *Jakob Nielsen e Kara Pernice* riguardo all'uso dell'eyetracking . Questa decisione è stata guidata dalla necessità di attirare l'attenzione dell'utente, utilizzando un colore dei pulsanti che si differenzia dallo sfondo. Questa disposizione mira a evitare che l'utente trascuri la parte inferiore dell'interfaccia, assicurando una maggiore visibilità e facilitando l'accesso alle funzionalità chiave dell'applicazione.

2.3.2 Osservazioni fatte da stakeholder

Durante la fase di sviluppo, abbiamo interagito con numerosi utenti che frequentemente utilizzano applicazioni o siti web per partecipare a aste online. Da queste interazioni, abbiamo notato che la **soddisfazione** degli utenti in questo contesto sono strettamente legate a diversi fattori chiave. Alcune delle osservazioni più significative includono:

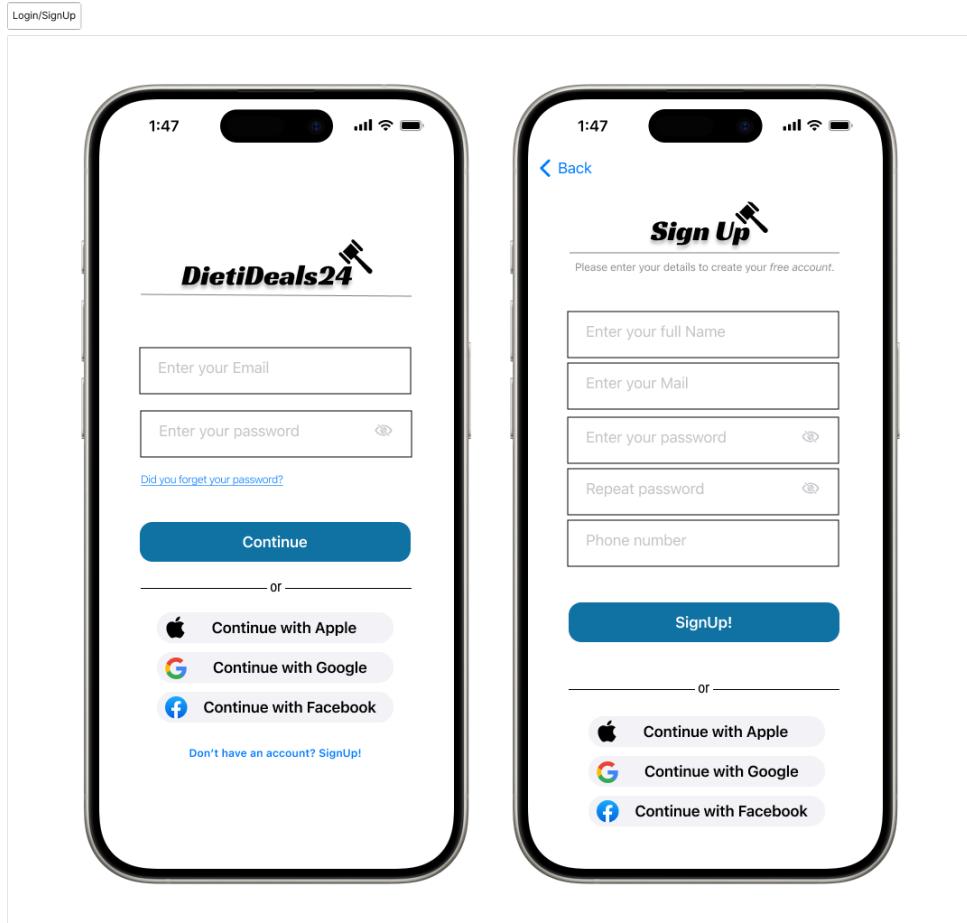
1. **Intuitività dell'Interfaccia Utente:** Gli utenti preferiscono un'interfaccia utente intuitiva e user-friendly. Elementi come menu ben organizzati, pulsanti chiari e una navigazione fluida sono stati identificati come fattori cruciali per garantire una esperienza d'uso senza intoppi.
2. **Visibilità delle Informazioni Cruciali:** La visibilità delle informazioni cruciali, come il prezzo corrente di un'asta, il tempo rimanente e le azioni disponibili, è stata considerata fondamentale. Gli utenti desiderano avere accesso immediato a queste informazioni senza dover cercare in modo eccessivo.
3. **Risposta Rapida:** La prontezza e la tempestività delle risposte dell'applicazione durante la presentazione di offerte o la partecipazione a aste sono state valutate positivamente. Gli utenti apprezzano un'esperienza reattiva che riduce al minimo i ritardi e le attese.
4. **Gestione Intuitiva delle Aste:** La possibilità di gestire facilmente le aste, compresa la creazione da parte dei vendori e la partecipazione da parte degli acquirenti, è stata ritenuta importante. Un flusso di lavoro chiaro e intuitivo per tutte le fasi di un'asta è essenziale per coinvolgere gli utenti in modo efficace.
5. **Personalizzazione del Profilo:** Gli utenti desiderano avere la possibilità di personalizzare il proprio profilo in modo da renderlo unico e rappresentativo delle loro preferenze e attività nell'applicazione.
6. **Notifiche Efficaci:** Le notifiche, specialmente quelle riguardanti l'esito delle aste o le offerte, devono essere chiare e tempestive. Gli utenti apprezzano essere prontamente informati delle attività rilevanti.

2.3.3 Elenco dei Mock-up :

I Mock-up sono stati realizzati attraverso Figma che è un editor di grafica vettoriale e uno strumento di prototipazione. Di seguito sono elencati tutti i prototipi di interfaccia sviluppati:

Sezione logIn/SignUp :

Questi mock-up descrivono il flow di registrazione e/o autenticazione al sistema; Quando si preme il pulsante *continue* o *sign up*, si accede alle schermate nelle prossime pagine.



In questo flow, si noti bene:

1. Nel caso in cui l'utente prova a registrare un ulteriore account con la stessa mail, il sistema avvisa che ciò non è possibile seguendo il requisito espresso nel punto 1 della traccia; In altre parole l'applicazione avvisa con un banner l'errore.

SEZIONE WORK-FLOW VENDITORE :

Tale work-flow delinea i comportamenti del prototipo per account di tipo venditore.

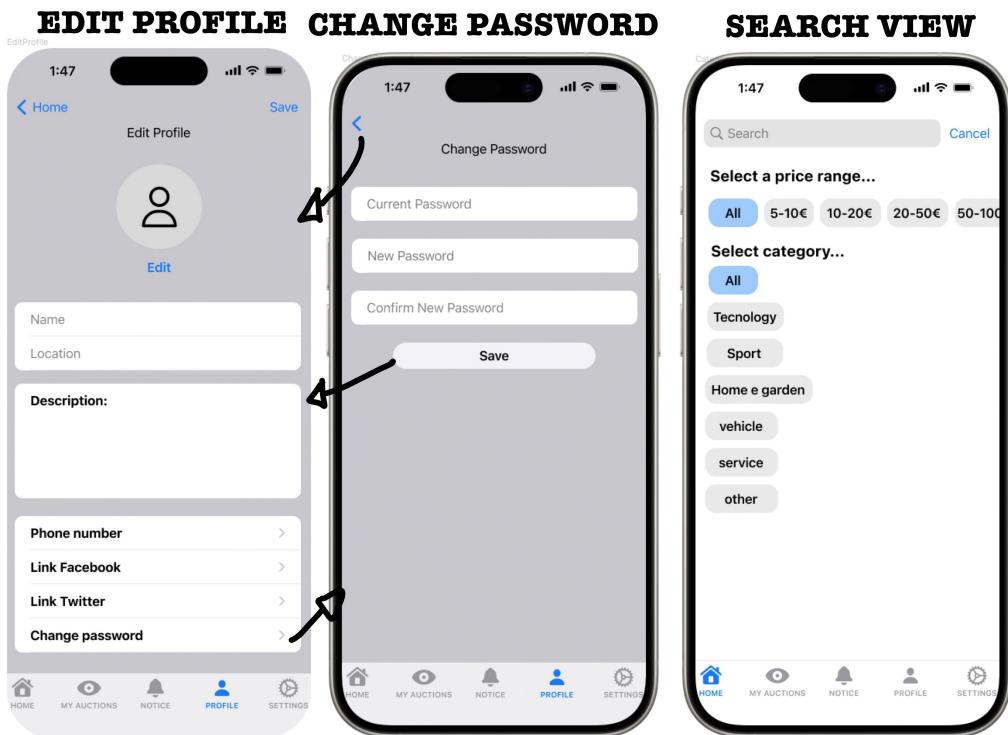
Home Venditore :



Alcune Osservazioni :

1. Nella schermata **home** è possibile sposarsi in cinque viste diverse con la pressione di un tasto
2. È possibile cliccare una qualsiasi asta per metterla a fuoco, vedendone le caratteristiche e il pulsante per fare un offerta (vedi vista Aste)
3. Il menù **my auctions**, siccome si tratta di un account di tipo *venditore*, mostra seguendo i casi d'uso descritti in precedenza, tutte le aste in corso e concluse create del venditore.
4. Ogni volta che si clicca su una notifica, viene visualizzato il profilo del compratore che ha vinto l'asta. Se l'asta è fallita, allora il sistema non mostra niente
5. Il pulsante in alto a destra è quello che permette la creazione di aste ed è presente solo se l'account è di tipo venditore.
6. Il pulsante **info aste** presente nel menu **settings**, anche se non prototipato, rimanda ad una schermata testuale in cui vengono spiegate le varie tipologie di aste ed il loro funzionamento.
7. il **toggle** in Settings serve a passare se disattivato al corrispondente account buyer associato.

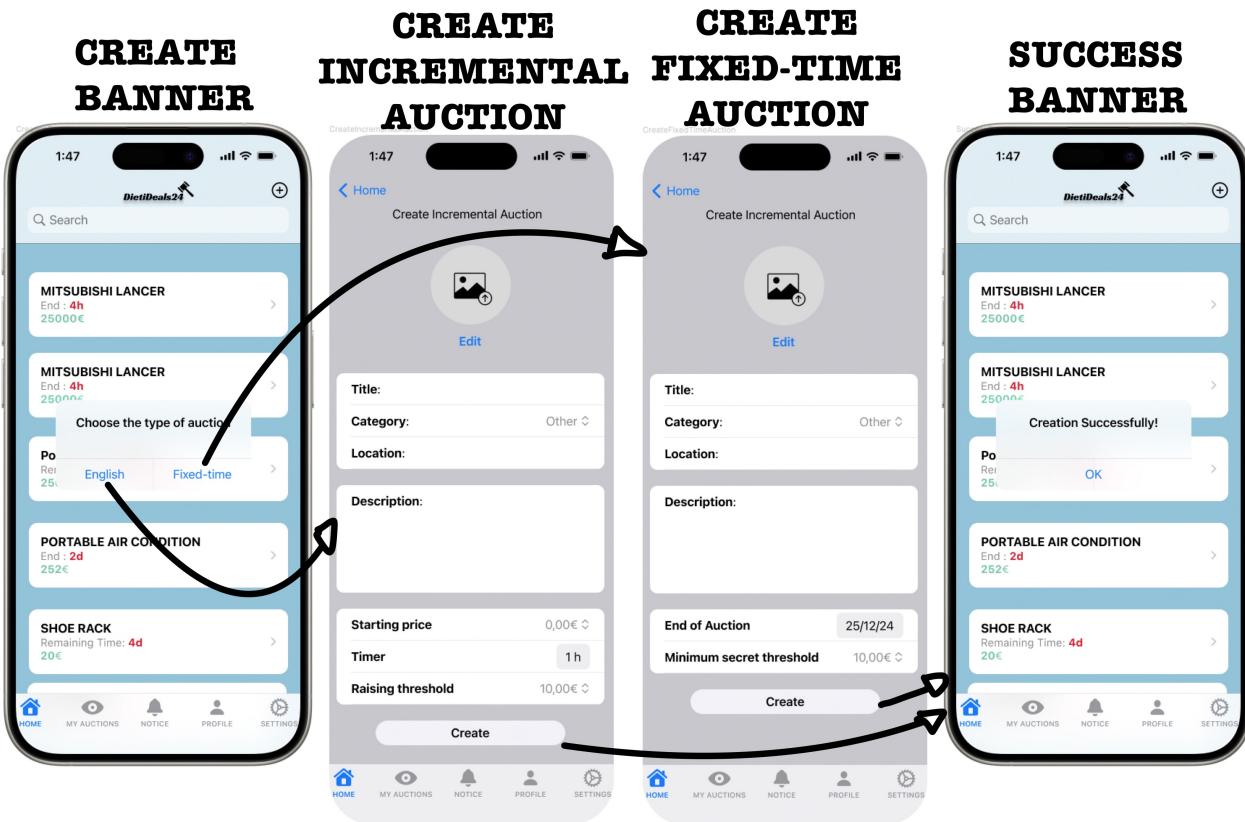
Profilo e Search View del Venditore :



Alcune Osservazioni:

1. Si noti che il pulsante di **modifica profilo** è accessibile sia dal menù settings che nel menu profilo, in quanto analizzando l'uso di questo prototipo da parte di stakeholders, si è notato che è un'operazione che viene spesso effettuata dagli utenti. A valle di ciò si è pensato di inserire un modo alternativo di svolgere queste operazioni in modo da semplificare la *user-experience*.
2. Quando il cambio di password avviene con successo, allora il sistema sconnette l'utente per sicurezza e lo rimanda nella schermata di login; In questo modo si accede solamente con la nuova password creata per quel utente.
3. La ricerca per categorie è stata una vittoria dopo tanti fallimenti, in quanto si poteva fare veramente in tanti modi, ma abbiamo deciso di includerla nella barra di ricerca, di modo che quando l'utente vuole cercare ha già davanti a se tutto quello che gli può essere utile per filtrare i risultati

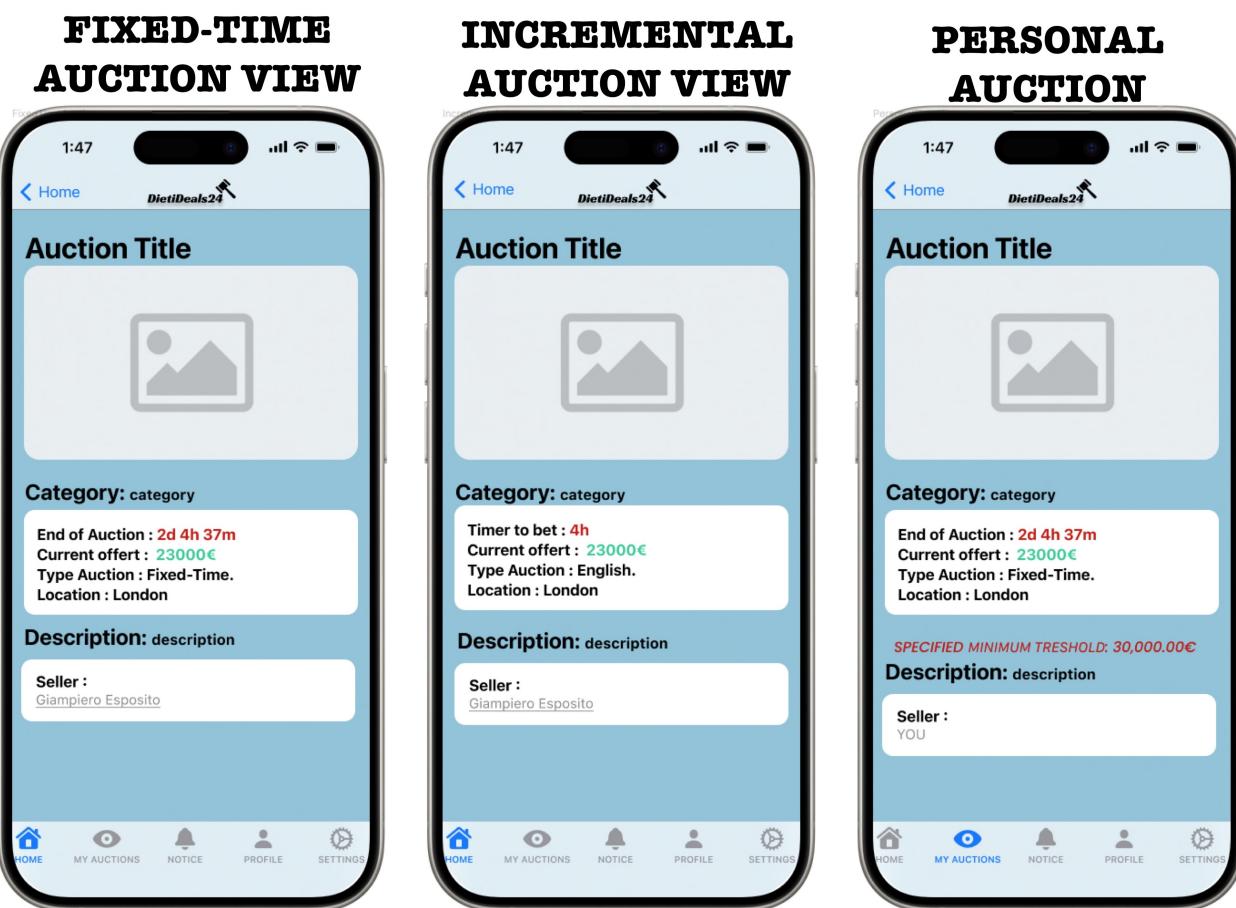
Creazione Aste Venditore :



Alcune Osservazioni:

1. Per entrambi i tipi di aste è obbligatorio in fase di creazione aggiungere il titolo, la categoria e la locazione. Questo è implicato dal fatto che un qualsiasi utente deve poter cercare l'asta o filtrarla per categoria e sapere dove si svolge. In altre parole l'asta dev'essere categorizzata all'interno del sistema in modo tale da renderla più accessibile.
2. Quando l'asta viene creata, allora l'utente viene rimandato alla schermata home e si riceve un **feedback** sotto forma di banner che annuncia il successo della creazione.

Vista delle Aste in dettaglio :



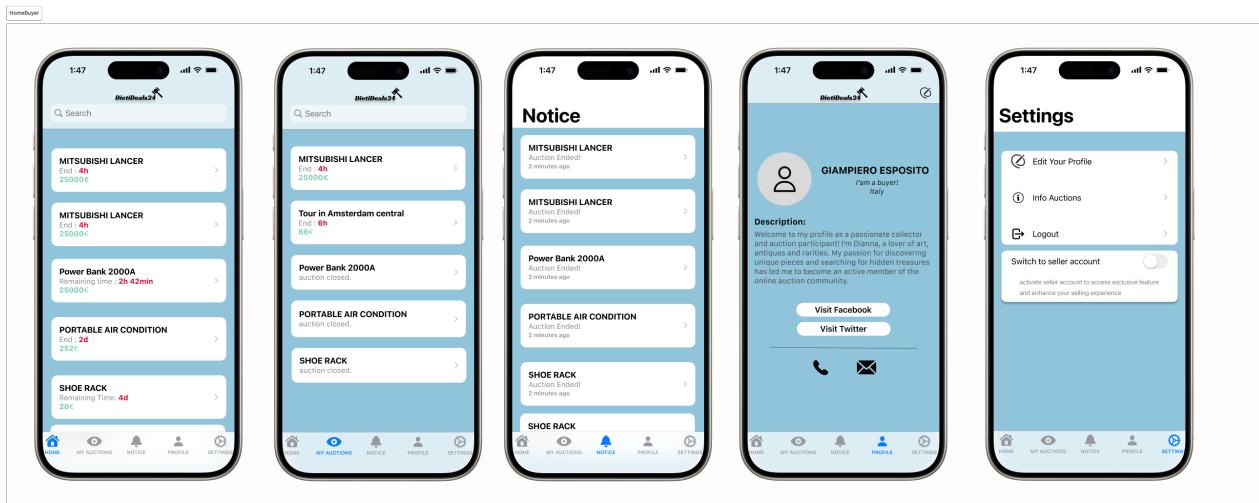
Alcune Osservazioni:

1. Essendo un profilo venditore, non è possibile puntare sulle aste di alcun tipo.
2. Cliccando sul nome del venditore che ha creato l'asta, sarai indirizzato al profilo corrispondente senza la possibilità di apportare modifiche, mentre se il profilo è il tuo, comparirà "YOU" al posto del nome del venditore

SEZIONE WORK-FLOW COMPRATORE :

Tale work-flow delinea i comportamenti del prototipo per account di tipo compratore.

Home Compratore :



Alcune Osservazioni :

1. Sulla schermata principale (**Home**), non sarà presente alcun pulsante per la creazione di asta, in quanto il compratore non è abilitato a effettuare tale funzionalità
2. Nella schermata principale (**Home**), puoi selezionare qualsiasi asta per visualizzarne dettagli e accedere al pulsante per effettuare un'offerta (vedi vista delle Aste).
3. Nella schermata **profile**, è possibile accedere al proprio profilo personale e eventualmente modificarlo
4. Nel caso non sia chiaro, è possibile ottenere informazioni dettagliate sulle differenze tra le aste a tempo fisso e le aste all'inglese tramite il pulsante dedicato **Info Auctions** nella schermata **settings**.
5. La schermata **my Auctions** mostra le aste attualmente in corso in cui hai effettuato almeno un'offerta, insieme a uno storico delle aste vinte.
6. All'interno della schermata **Settings** è possibile passare all'account seller associato attraverso il toggle.

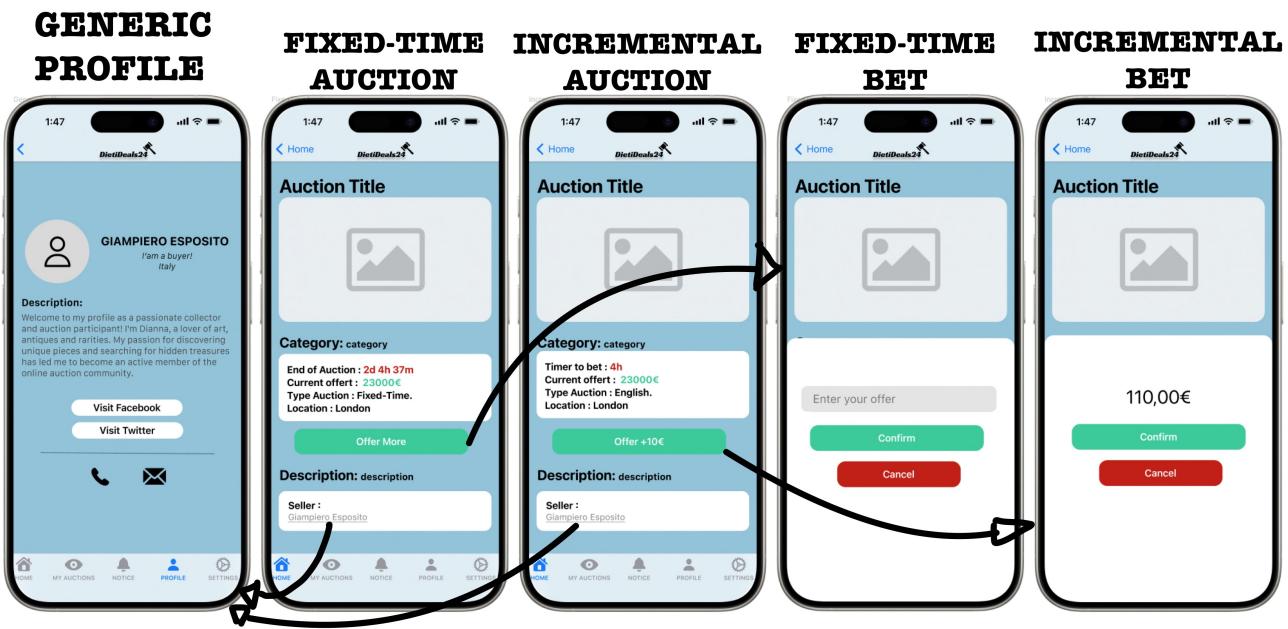
Profilo e Search View del Compratore :



Alcune Osservazioni:

1. Le viste sono praticamente le stesse di un venditore e funzioneranno nello stesso modo

Vista Aste Compratore :



Alcune Osservazioni:

1. Cliccando sul nome del venditore che ha creato l'asta, sarai indirizzato al profilo corrispondente senza la possibilità di apportare modifiche
2. Nella schermata delle **Aste All'Inglese**(incremental), cliccando sul pulsante "Offri" con l'importo prefissato dal venditore, è possibile effettuare una puntata.
3. Nella schermata delle **Aste a Tempo Fisso**(fixed-time), basta premere il pulsante "Offri" per effettuare una puntata, poiché il sistema mostra un banner in cui è possibile inserire l'importo desiderato.
4. Grazie alla metodologia di inserimento che avviene con comparsa del **tastierino numerico e conferma rapida**, si risolvono problemi riscontrati durante le visite con stakeholders, quali puntate effettuate per sbaglio e difficoltà nell'inserire l'importo, e il tutto senza notare nessun significativo peggioramento nella performance del sistema e del suo utilizzo

Di seguito è riportato il link di Figma : *link*

2.4 TARGET UTENTI :

Durante la progettazione di un nuovo software, è fondamentale individuare e definire il target di utenti coinvolti nell'uso dell'applicazione.

Personas :

Nello sviluppo del software, è essenziale avere un'idea concreta dell'utente che utilizza l'applicazione. Questo aiuta il team di sviluppo a concentrarsi su determinati tipi di utenti, categorie ben specifiche che riassumono idee, obiettivi, ruoli e ambizioni personali. La sintesi di un cluster ben preciso di utenti è chiamata *personas*, ottenute attraverso ricerche etnografiche. Ogni persona sintetizza comportamenti, obiettivi e caratteristiche rilevanti di utenti reali che utilizzeranno il software da progettare.

Obiettivo :

Nello specifico, le personas permettono ai progettisti di concentrarsi sulle esigenze reali dei futuri utenti, evitando generalizzazioni troppo ampie. In altre parole, permettono di focalizzare l'attenzione solamente su un gruppo ristretto di persone con determinate caratteristiche, evitando la creazione di un prodotto che cerchi di soddisfare un pubblico troppo variegato, finendo inevitabilmente all'insuccesso.

Prima di passare alle personas generate, bisogna seguire dei passi ben specifici: innanzitutto, stabilire le **variabili comportamentali**, ovvero tutte le caratteristiche osservabili e misurabili dell'utente come, ad esempio, la tipologia di attività svolte e la loro frequenza, attitudini, motivazioni e capacità. Dopo l'individuazione di tali variabili, è necessario far emergere quelle più importanti ai fini dello sviluppo software. In conclusione, una volta riassunti i dati e lo studio fatto sugli utenti, è possibile sviluppare queste personas partendo dalle informazioni raccolte.

2.4.1 Personas Individuate :

Per la creazione delle Personas abbiamo consultato varie fonti da cui sono emersi i dati rilevanti per la costruzione di questi modelli : <https://similarweb.com> da cui abbiamo estratto molti dati relativi agli utenti che usano le aste online su siti quali <https://bidoo.com> oppure <https://www.catawiki.com>. Un'altra fonte importante è stata le statistiche del sito <https://www.ebay.com>.

Di seguito riportiamo le Personas che sono emerse facendo inferenza sui dati raccolti dalle fonti sopra citate.



Chiara Formicola

About

25 Graphic Designer
Italia Fidanzata
Curiosa Innovativa Solare
Insicura Curiosa

Descrizione

Chiara è una giovane graphic designer che si avvicina per la prima volta al mondo delle aste online. Non ha molta esperienza in questo settore ma è interessata a scoprire nuove opportunità per fare acquisti unici e interessanti. Ha una mente creativa e apprezza l'arte in molte forme.

Obiettivi

- Imparare il Processo: Chiara vuole imparare il processo delle aste online senza sentirsi sopraffatta. Cerca un'applicazione che fornisca una guida passo dopo passo e risorse informative per i principianti.
- Scoprire Nuovi Prodotti: Desidera scoprire nuovi prodotti e artisti attraverso le aste online.

Criticità

- La principale criticità per Chiara potrebbe derivare dalla sua inesperienza nel partecipare alle aste online. Potrebbe sentirsi insicura riguardo a come funzionano le aste, le dinamiche di offerta, e su come evitare possibili truffe o errori. La sua priorità sarà quindi un'esperienza utente facile e chiara.

Personalità

Introverso	Estroverso
Pensatore	Sensazionalista
Giudice	In Prospettiva
Calmo	Turbolento
Intuitivo	Osservatore

Skills tecniche

Internet: 5/10
Social Networks: 7/10
Online Shopping: 4/10

Brands

Google, TikTok, Instagram, Apple, Spotify, Twitter



Marco Esposito

About

24 Studente Economia
Italia Single
Meticoloso Osservatore Energico
Creativo

Descrizione

Marco è uno studente universitario appassionato di economia e finanza. Con un budget limitato, è costantemente alla ricerca di affari e opportunità per ottenere prodotti di valore senza spendere troppo. Le aste online sono diventate il suo modo preferito per fare shopping, poiché gli offrono la possibilità di trovare oggetti unici a prezzi convenienti.

Obiettivi

- Riuscire a trovare affari unici a prezzi molto convenienti.

Criticità

- Eccezionale focalizzazione sul prezzo in quanto possiede un budget molto limitato.

Personalità

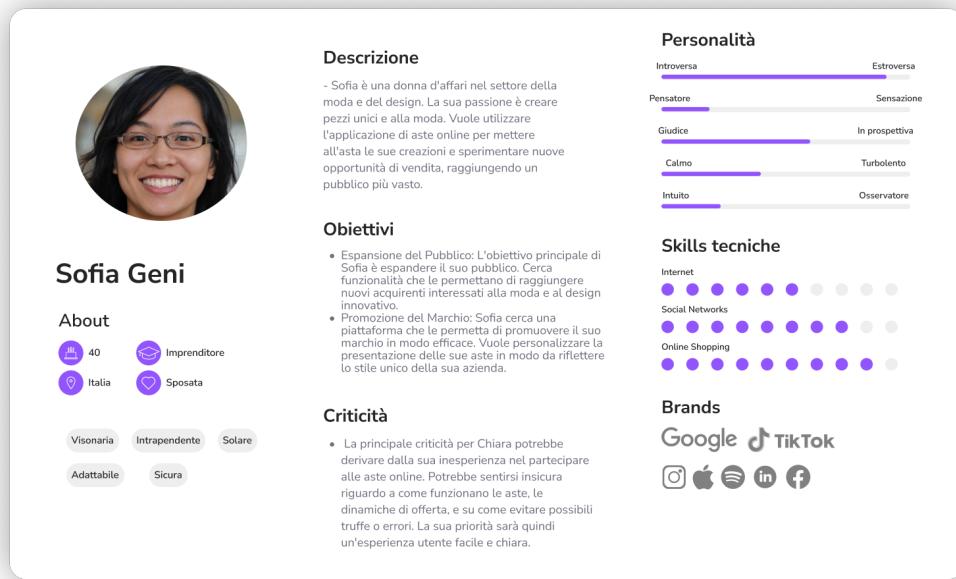
Introverso	Estroverso
Pensatore	Sensazione
Giudice	In prospettiva
Calmo	Turbolento
Intuitivo	Osservatore

Skills tecniche

Internet: 7/10
Social Networks: 6/10
Online Shopping: 5/10

Brands

Google, MARVEL, TikTok, Instagram, ZARA, Apple, Spotify, Amazon



2.4.2 Analisi dei tratti caratteristici:

Nel dettagliato esame delle personas, emergono distinti tratti che intendiamo identificare in modo univoco attraverso attributi chiave:

- **Il Primo Utente - Chiara Formicola:** Questa categoria è incarnata dalla persona di "Chiara Formicola". L'attributo distintivo di questa tipologia di utente è la sua novità nel mondo delle aste. Pertanto, si rivelerebbe essenziale implementare nel sistema funzionalità didattiche per facilitare il suo approccio, cercando di rendere la curva di apprendimento più progressiva, evitando bruschi aumenti. Un approccio proposto consiste nell'introduzione di un pulsante "Info Aste," il quale, se attivato, fornirebbe informazioni chiare e semplici sulle diverse tipologie di aste presenti nel sistema.
 - **Il Cacciatore d'Affari - Marco Esposito:** Questa categoria di utenti, personificata da "Marco Esposito," rappresenta coloro con un budget limitato alla ricerca dell'affare ideale. Al fine di agevolare questo specifico target, si è pensato di introdurre un filtro basato sul prezzo, consentendo loro di individuare facilmente le opportunità economiche che rientrano nei loro parametri finanziari.
 - **L'Imprenditore Innovativo - Sofia Geni:** Questa tipologia di utenti è rappresentata da "Sofia Geni" e comprende coloro che utilizzano la piattaforma per promuovere la propria azienda. Si tratta di individui interessati a fare pubblicità attraverso aste. Da questa osservazione, emerge la necessità di implementare una funzionalità che consenta ai venditori di visualizzare lo storico delle proprie aste. Questo aspetto mira a migliorare la gestione delle aste create, fornendo una panoramica dettagliata delle performance passate.

2.5 VALUTAZIONE DELL'USABILITÀ A PRIORI :

"Per un utente l'interfaccia è l'applicazione"

Lo studio dell'usabilità rappresenta un importante passo nello sviluppo della interfaccia; Con questa frase di *Alan Kay* si esprime come l'applicazione rappresenti l'unico e tangibile contatto *uomo-macchina*, soprattutto per le persone che d'informatica se ne intendono ben poco.

cosa s'intende per usabilità? L'usabilità di un'interfaccia è la misura *dell'efficacia, efficienza e soddisfazione* con cui determinati utenti possono compiere determinati compiti in un determinato contesto utilizzando tale interfaccia. Per la valutazione dell'usabilità, esistono tuttavia diversi approcci nelle diverse fasi di progettazione. Come specifica **lo standard ISO 13407** : *"la valutazione è un passo essenziale in una progettazione human-centred."*, difatti l'usabilità è in generale studiata sia a priori dello sviluppo sia a prodotto finito.

2.5.1 Euristiche di Nielsen :

Le euristiche di Nielsen sono un insieme di principi guida proposti da Jakob Nielsen, un esperto di usabilità, per valutare l'usabilità di un'interfaccia utente. Queste euristiche forniscono un quadro generale per identificare problemi comuni di usabilità durante l'ispezione di un'interfaccia. Di seguito sono riportate brevi spiegazioni delle euristiche di Nielsen:

1. **Visibilità dello stato del sistema:** L'interfaccia dovrebbe fornire feedback chiaro e comprensibile sull'esito delle azioni dell'utente e sullo stato corrente del sistema.
2. **Corrispondenza tra il sistema e il mondo reale:** Il linguaggio, i concetti e le azioni all'interno dell'interfaccia dovrebbero riflettere il mondo reale in modo che gli utenti possano comprendere facilmente.
3. **Controllo utente e libertà:** Gli utenti devono avere la possibilità di navigare liberamente all'interno del sistema, e in caso di errori, devono poter annullare o correggere le azioni.
4. **Consistenza e standard:** L'interfaccia dovrebbe seguire convenzioni e standard del settore per garantire una familiarità e coerenza nella user experience.
5. **Prevenzione degli errori:** L'interfaccia dovrebbe essere progettata in modo da evitare errori comuni da parte degli utenti, o fornire un modo semplice per recuperare da essi.
6. **Riconoscimento anziché ricordo:** Gli utenti dovrebbero essere in grado di riconoscere facilmente le opzioni e le azioni anziché dover ricordare informazioni specifiche.
7. **Flessibilità ed efficienza d'uso:** L'interfaccia dovrebbe adattarsi alle diverse esigenze degli utenti, consentendo sia agli utenti principianti che a quelli esperti di utilizzare il sistema in modo efficiente.

8. **Estetica e design minimalista:** Un'interfaccia pulita e ben progettata migliora l'usabilità fornendo un ambiente visivo piacevole e senza distrazioni.

9. **Aiuto agli utenti a riconoscere, diagnosticare e recuperare gli errori:** L'interfaccia dovrebbe fornire messaggi di errore chiari e indicare agli utenti come correggere eventuali problemi.

10. **Aiuto e documentazione:** L'interfaccia dovrebbe includere documentazione utile e accessibile per aiutare gli utenti a comprendere il sistema e risolvere eventuali problemi.

Questi punti sono stati ampiamente discussi dal team in fase di prototipazione dell'interfaccia, considerandoli come punti cardine a cui fare riferimento nelle impostazioni di ogni schermata.

2.5.2 Model Human Processor:

Per garantire alcuni di questi punti importanti dell'euristica di Neilsen, sono stati seguiti studi anche di *psicologia sperimentale*; Nello specifico si è analizzato i vari processi di **comprendione** di un utente, attraverso lo studio della *memoria a breve termine* e quella a *lungo termine*.

Memoria e Attenzione dell'utente:

La memoria umana può essere concettualizzata attraverso il noto **modello modale**, che illustra come le informazioni provenienti dalle interfacce vengano memorizzate nella mente dell'utente. Questo processo di compressione e memorizzazione segue diverse fasi: inizialmente, le informazioni sono temporaneamente immagazzinate nella memoria sensoriale, che, se paragonata a una macchina, rappresenta una sorta di buffer in cui le informazioni non ancora elaborate sono conservate. Successivamente, entrano in gioco la memoria a breve termine e quella a lungo termine.

Gli studi psicologici sulla memoria a breve termine hanno evidenziato che gli esseri umani hanno la capacità di ricordare fino a **sette "chunk" di informazioni**. In base a questa comprensione, le interfacce progettate sono state sviluppate evitando di sovraccaricare l'utente con un numero di "chunk" prossimo a sette, al fine di rendere più agevole il ricordo di tutte le informazioni pertinenti.

2.5.3 Apprendimento degli utenti:

Un altro aspetto di fondamentale importanza è lo studio in funzione del tempo del processo di apprendimento degli utenti. In altre parole, si studia la curva di apprendimento dell'utente per stabilire quanto un'interfaccia possa essere intuitiva e facilmente apprendibile nel corso del tempo. Questo processo di valutazione è essenziale per garantire che gli utenti possano sfruttare appieno le funzionalità del sistema senza incontrare difficoltà e frustrazioni e per massimizzare l'efficacia complessiva dell'interfaccia utente.

Per condurre uno studio sull'apprendimento degli utenti, è possibile utilizzare diverse metodologie :

1. Test di Usabilità Iniziale: Questo coinvolge un gruppo di utenti rappresentativi che interagiscono con l'interfaccia per la prima volta. Questo fornisce informazioni sulla facilità d'uso iniziale e identifica eventuali ostacoli immediati.

2. Osservazione del Comportamento: Monitorare il modo in cui gli utenti interagiscono con l'interfaccia nel tempo può rivelare i progressi nel loro apprendimento. L'osservazione diretta o l'analisi dei dati di utilizzo possono essere utilizzate per valutare i cambiamenti nel comportamento degli utenti.

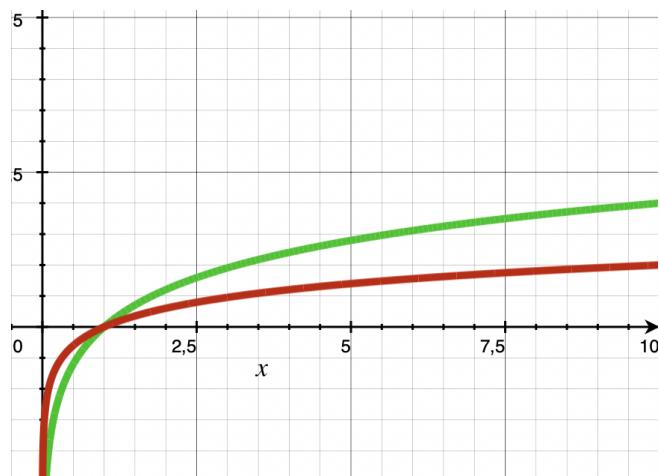
4. Analisi dei Dati di Utilizzo: Monitorare le metriche di utilizzo nel tempo, come il tempo speso su determinate funzioni o il numero di errori commessi, può fornire indicazioni sulla curva di apprendimento degli utenti.

2.5.4 Curva di Apprendimento :

Studiare la curva di apprendimento degli utenti è cruciale per il successo a lungo termine di un sistema. Un'interfaccia che risulta difficile da imparare potrebbe scoraggiare gli utenti, riducendo l'adozione del sistema e compromettendo l'efficienza complessiva. Pertanto, l'ottimizzazione dell'apprendimento degli utenti dovrebbe essere parte integrante del processo di progettazione e miglioramento dell'interfaccia utente.

Dopo questa fase, è stato chiesto ai colleghi del corso d'informatica, amici e persone che non sanno nulla di design e usabilità di valutare il prototipo nei suoi due diversi flow di lavoro : **compratore e venditore**.

Da questi dati ricavati è stato formato il seguente grafico dell'usabilità in funzione del tempo :



Di rosso è raffigurata la curva di apprendimento del compratore, di verde quella dell'utente venditore. Lo strumento software usato per la creazione di questo grafico è *Grapher*. Lo studio dimostra che un venditore impiega più tempo a comprendere il funzionamento del software in quanto ha una funzionalità importante in più, ovvero quello di creare aste di diverso tipo.

2.6 GLOSSARIO :

Qui sono riportati i vari termini utilizzati nel documento dei requisiti, con relative definizioni.

- **UML** : In ingegneria del software, UML (Unified Modeling Language, "linguaggio di modellizzazione unificato") è un linguaggio di modellazione e di specifica basato sul paradigma orientato agli oggetti. Nel nostro contesto lo usiamo per modellare il dominio del problema attraverso dei diagrammi (come ad esempio use case diagram).
- **System Under Development** : Rappresenta in sintesi i limiti del software che bisogna sviluppare; In altre parole circoscrive il dominio del problema, il quale può essere immerso in una realtà ben più ampia come può essere un modulo software o comunque un ambiente esterno che interagisce con esso ed è già dunque definita la sua implementazione.
- **Eye tracking** : L'Eye tracking è il monitoraggio oculare che consente di rilevare gli aspetti su cui si focalizza l'attenzione di un individuo, analizzando il movimento dello sguardo.
- **Mock-up** : Il termine “ mockup ” deriva dall'inglese e significa letteralmente “modello” e si può definire come una simulazione della realtà, che permette di avere un'idea più chiara della resa finale di un progetto in sviluppo.
- **Stakeholder** : Ciascuno dei soggetti direttamente o indirettamente coinvolti in un progetto o nell'attività di un'azienda.
- **Prototipo** : Un prototipo è, dunque, un modello approssimato o parziale del sistema che vogliamo sviluppare, realizzato allo scopo di valutarne determinate caratteristiche.
- **Chunck** : Il chunk in psicologia cognitiva è un'unità di informazione.
- **Feedback** : Un feedback da parte di un software si riferisce alle informazioni o alle risposte fornite dal software in risposta a determinate azioni o input dell'utente

SPECIFICA DEI REQUISITI

3.1 MODELLO DI DOMINIO

Il modello di dominio è una componente fondamentale nell'analisi orientata agli oggetti. Si tratta di una rappresentazione visuale delle classi concettuali o degli oggetti presenti nel mondo reale, insieme alle relazioni che intercorrono tra di essi, all'interno di un particolare dominio di interesse.

L'utilizzo di *UML*, un linguaggio visuale, si giustifica in quanto facilita la comprensione dei concetti e delle relazioni, sfruttando la capacità del cervello umano di elaborare informazioni visive e connessioni tra di esse.

3.2 SCELTE PROGETTUALI

Nel contesto delle scelte progettuali e del processo di sviluppo, si procede con la descrizione dei diagrammi delle classi di analisi, supportati dai mockup dei casi d'uso. Per identificare le classi concettuali nel dominio di un caso d'uso, si adotta la tecnica dell'analisi linguistica o euristica di *Abbot*. Inoltre, il processo segue l'euristica **Three-Object Type**, che guida il raggruppamento degli oggetti in categorie come Entity, Boundary e Control.

3.3 PENSANDO AL FUTURO: PROGETTARE PER FACILITARE LE MODIFICHE DEL CODICE

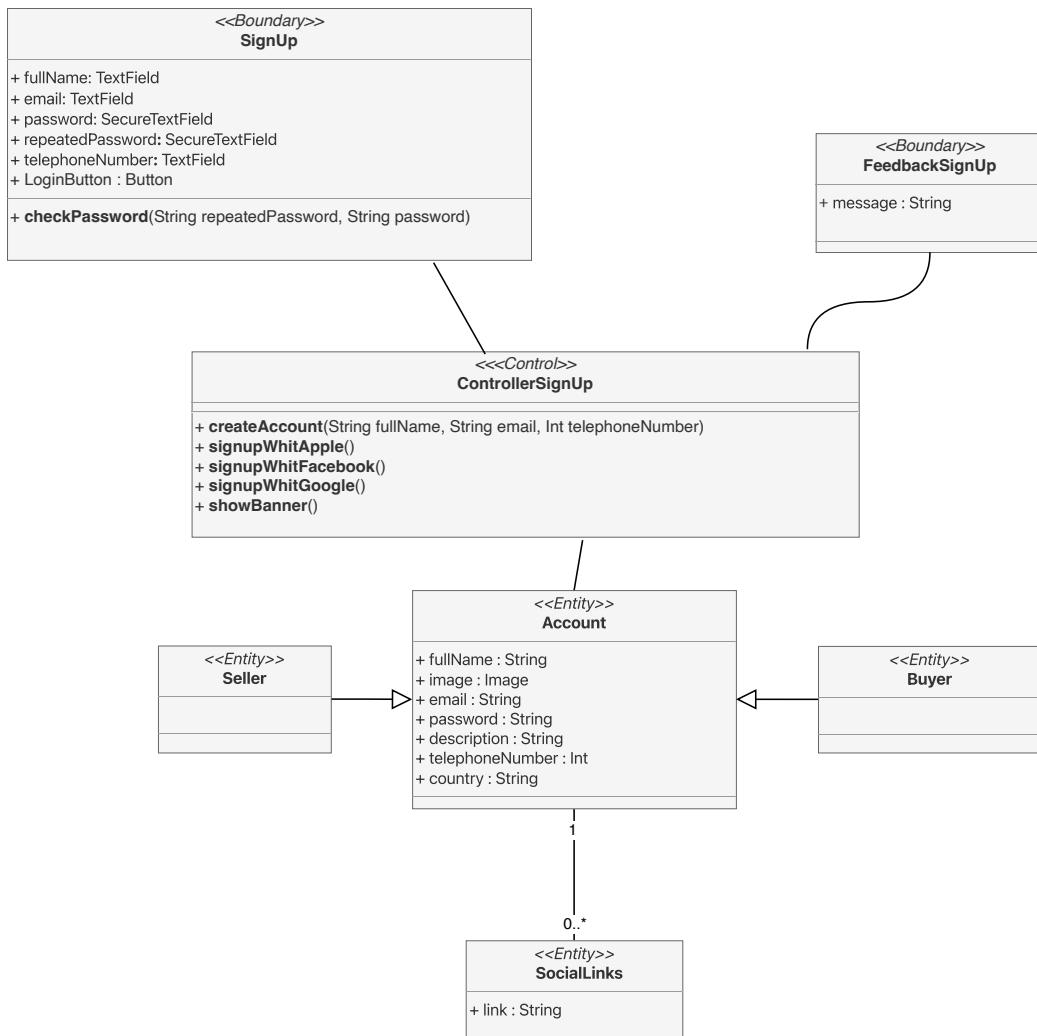
"Program to an interface not an implementation"

Questo è uno dei principi che è stato seguito durante la costruzione dei diagrammi. Esso riprende quello espresso nel libro *Design Pattern* scritto da Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides. L'obiettivo principale è quello di scrivere codice che dipenda dalle astrazioni, piuttosto che dalle implementazioni concrete. Questo rende il sistema più flessibile e resistente ai cambiamenti, in quanto le modifiche all'implementazione di una particolare classe non influenzano il codice che dipende dall'interfaccia. Quando si segue questo principio, si crea un'architettura più modulare e manutenibile. Inoltre, favorisce il riutilizzo del codice, poiché le implementazioni possono essere sostituite senza dover modificare il codice che fa uso delle astrazioni.

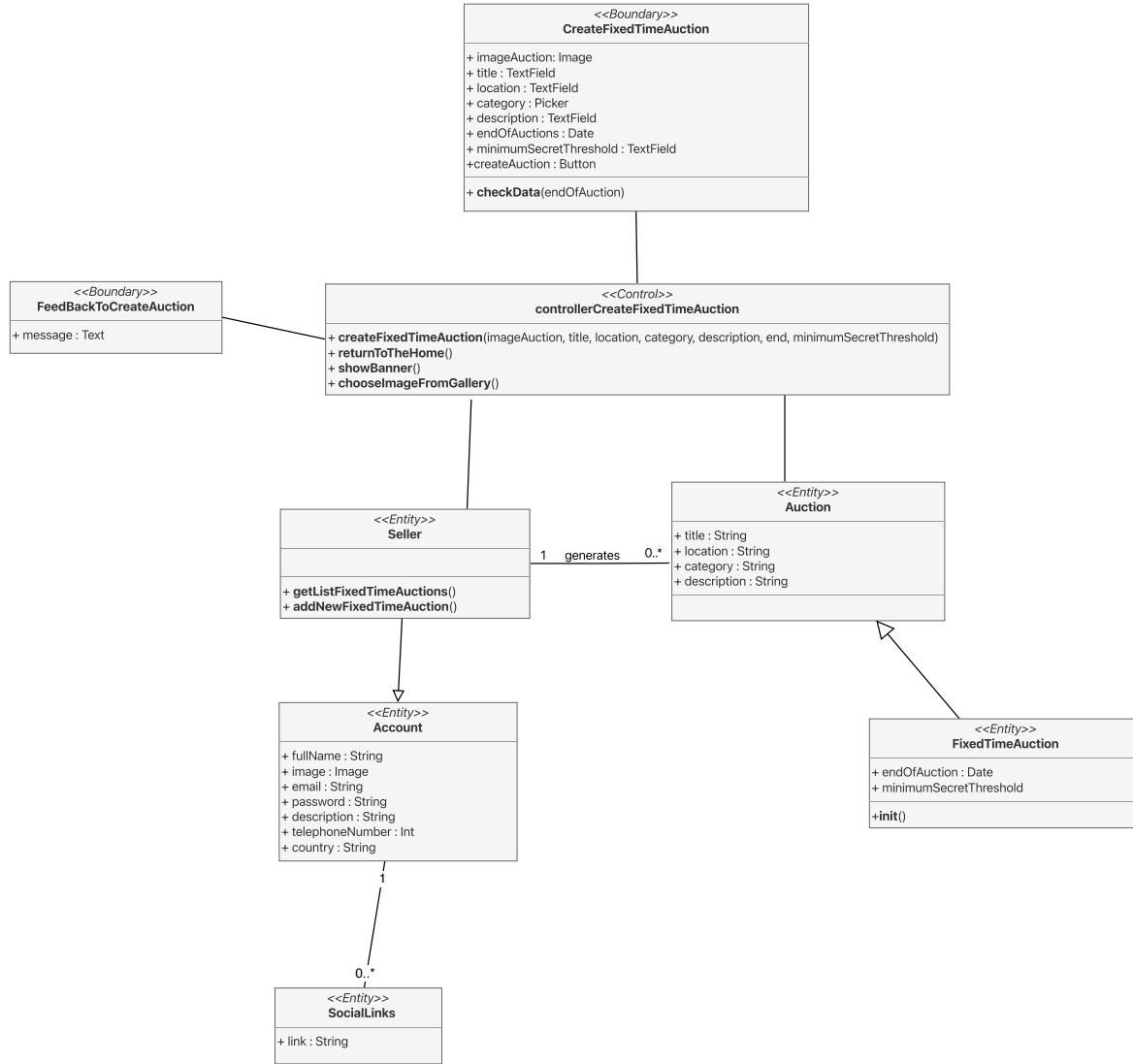
Nelle sezioni successive, verranno presentati i diagrammi delle classi per ciascun caso d'uso del sistema descritti nella sezione precedente, delineando così le scelte progettuali adottate.

3.3.1 Registrazione al sistema tramite mail, Facebook, Google, Apple.

I seguenti casi d'uso rappresentati sono legati ad un utente non ancora registrato.

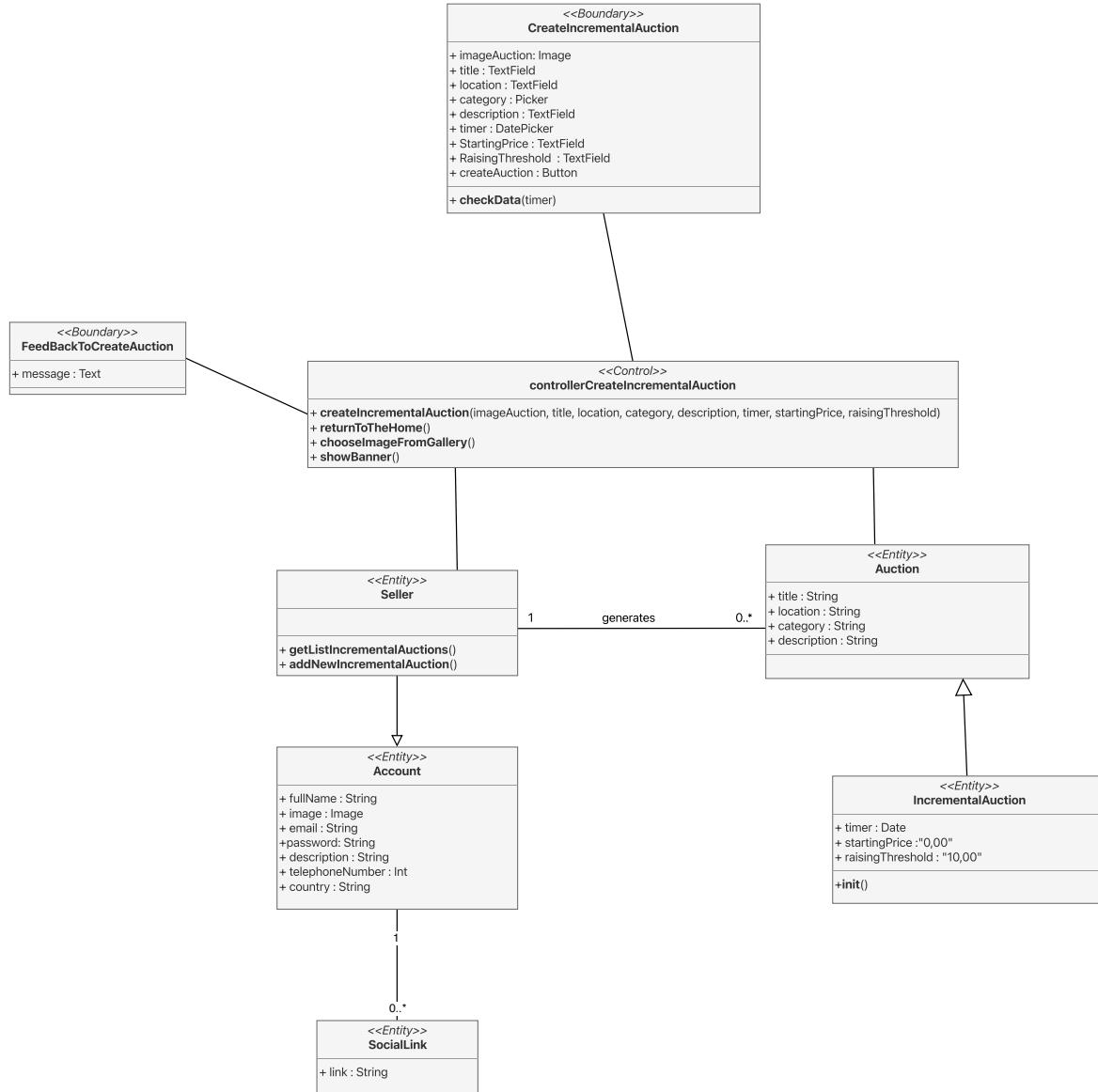


3.3.2 Creazione Asta a tempo fisso.



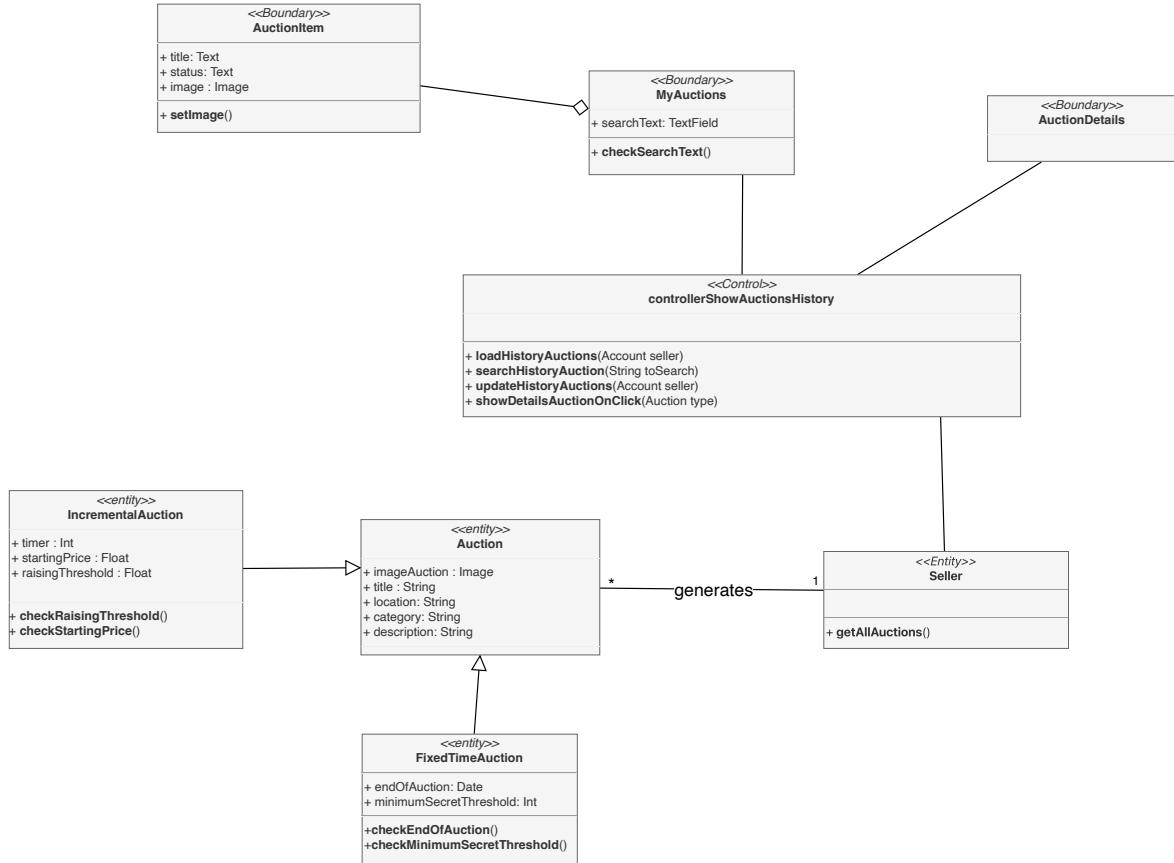
Questo caso d'uso è specifico per un utente di tipo venditore

3.3.3 Creazione Asta Incrementale.



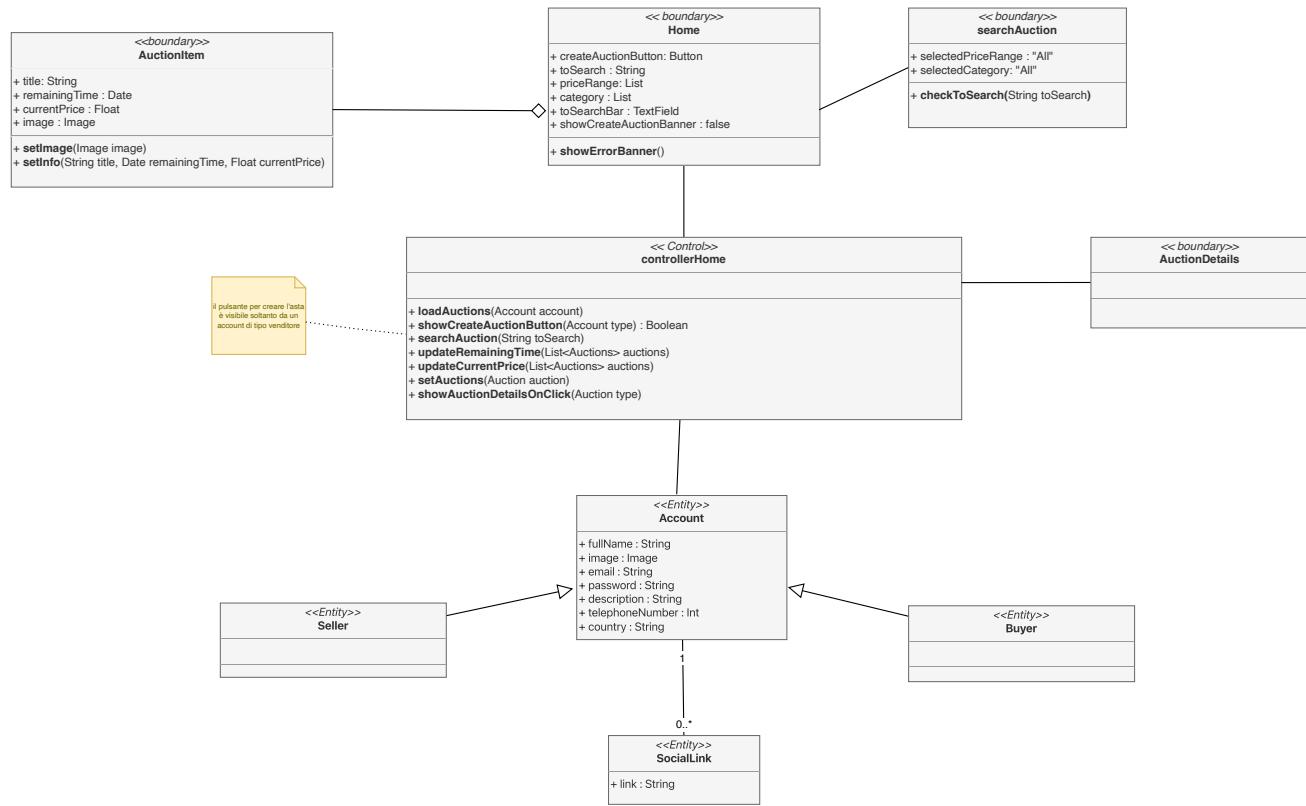
Questo caso d'uso è specifico per un utente di tipo venditore

3.3.4 Mostra storico delle aste create.

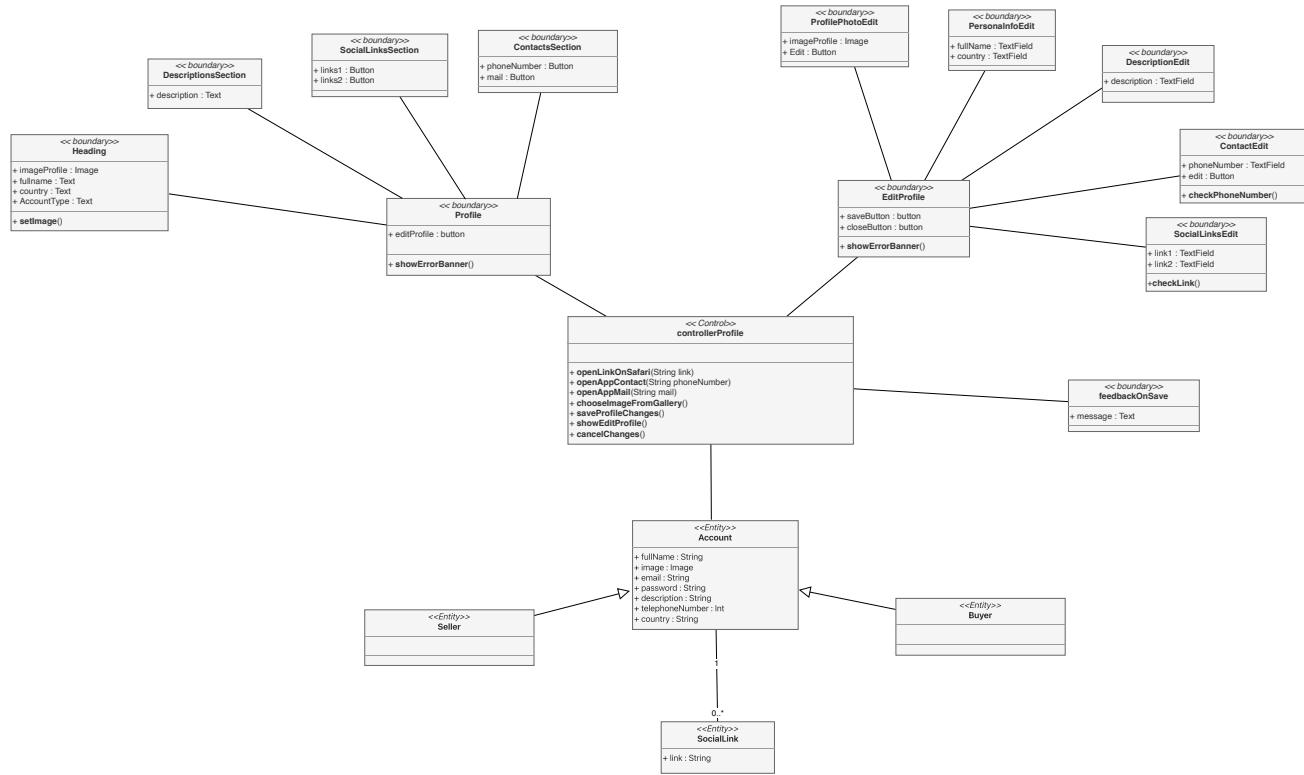


Questo caso d'uso è specifico per un utente di tipo venditore

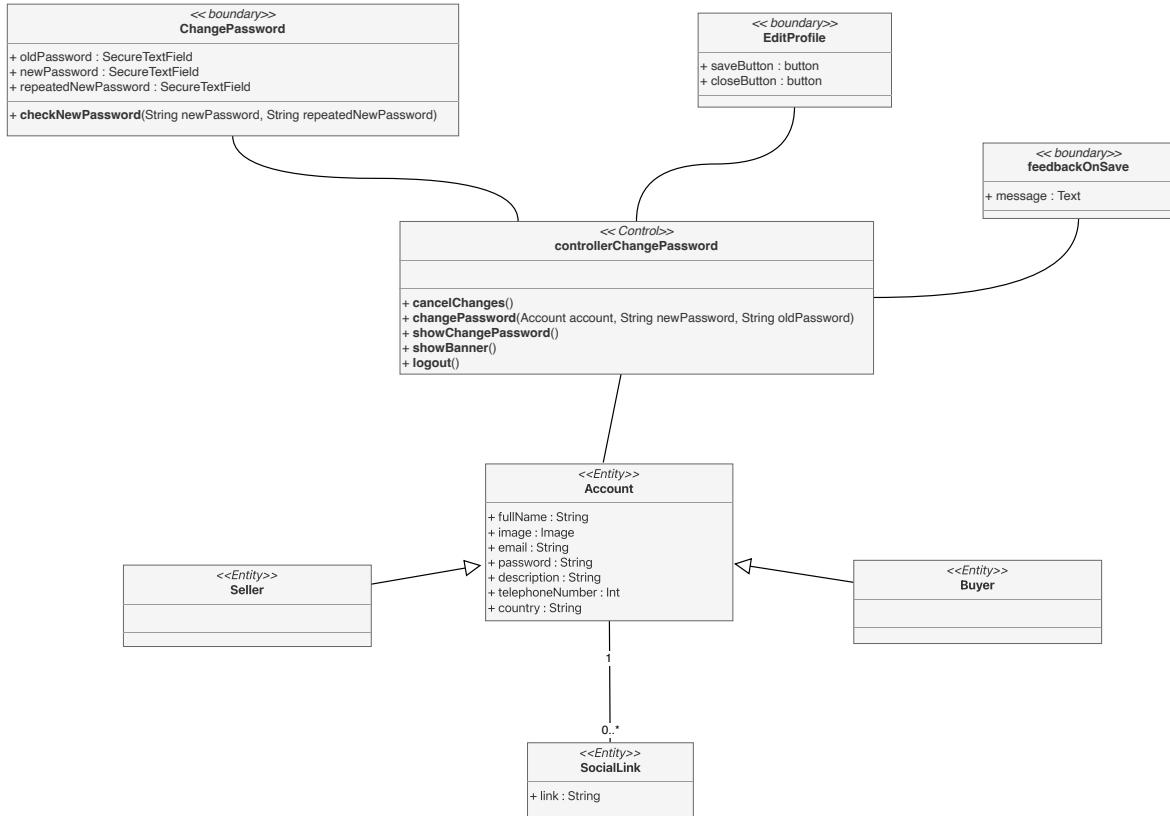
3.3.5 Mostra aste correttamente attive nel sistema con ricerca e filtro.



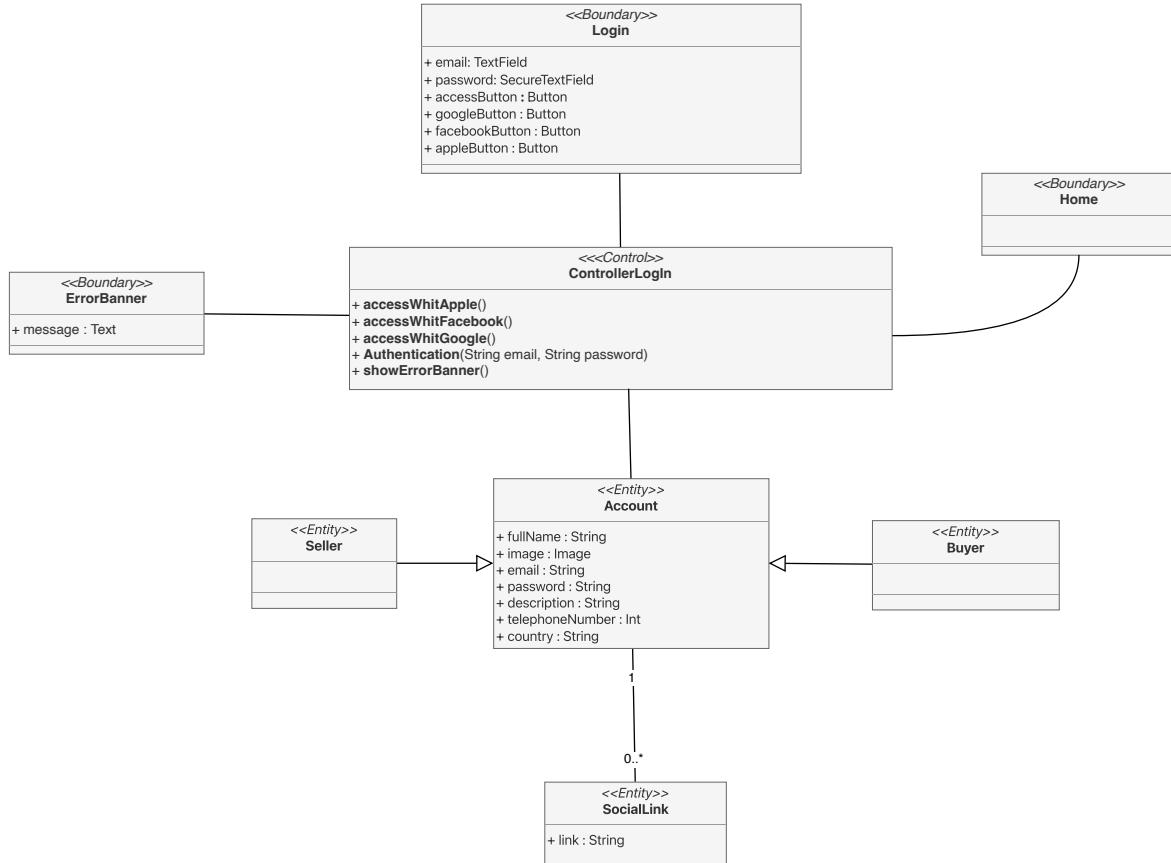
3.3.6 Personalizzazione del profilo.



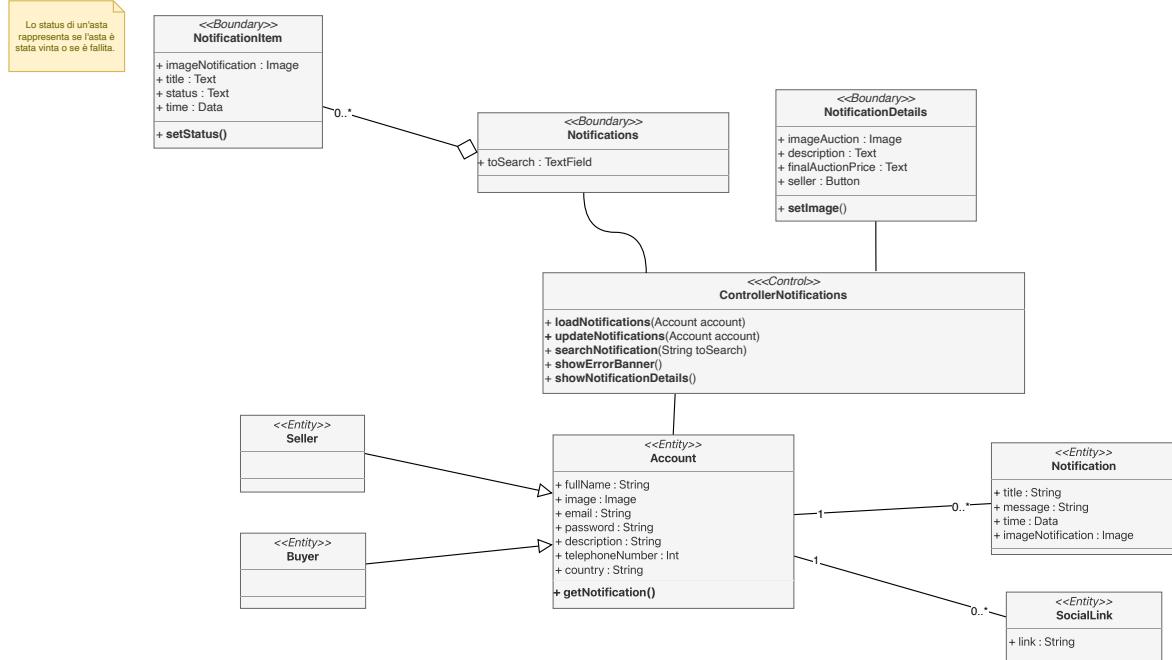
3.3.7 Cambio password.



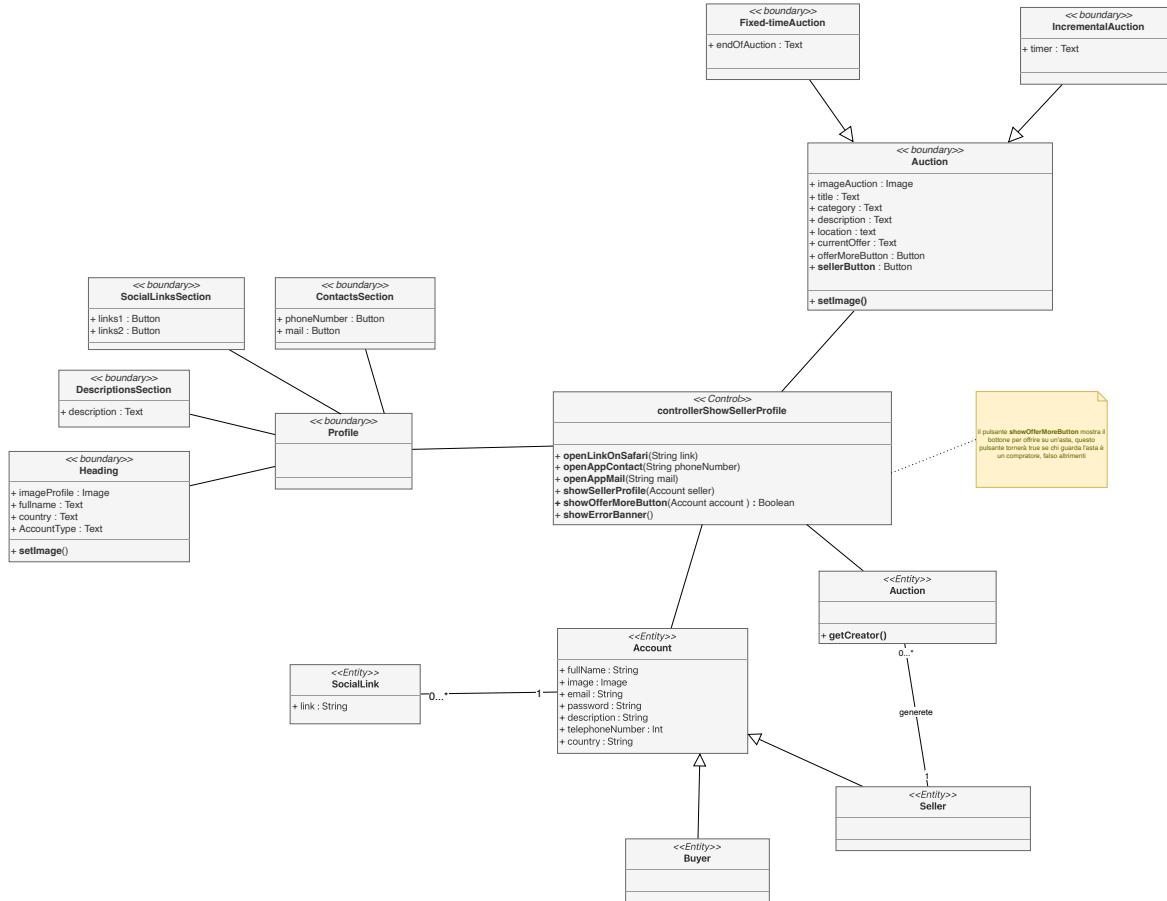
3.3.8 Login al sistema.



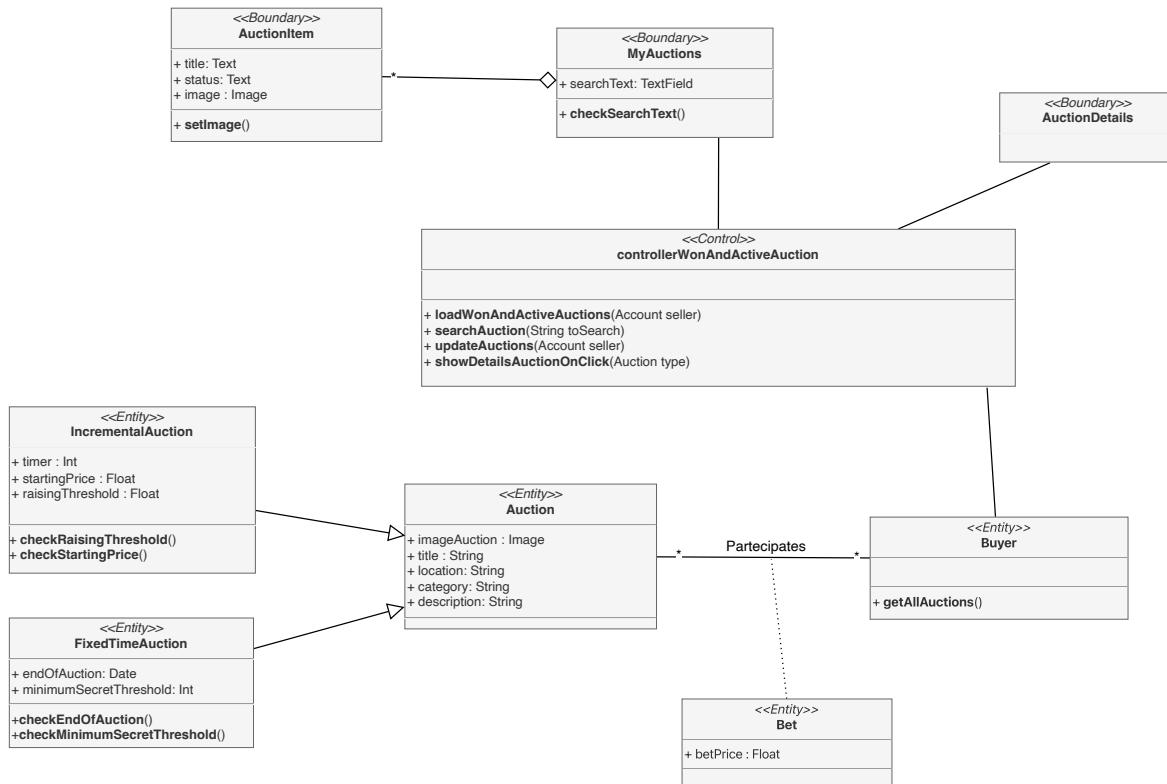
3.3.9 Visualizzazione delle notifiche.



3.3.10 Mostra profilo del venditore di un'asta specifica.

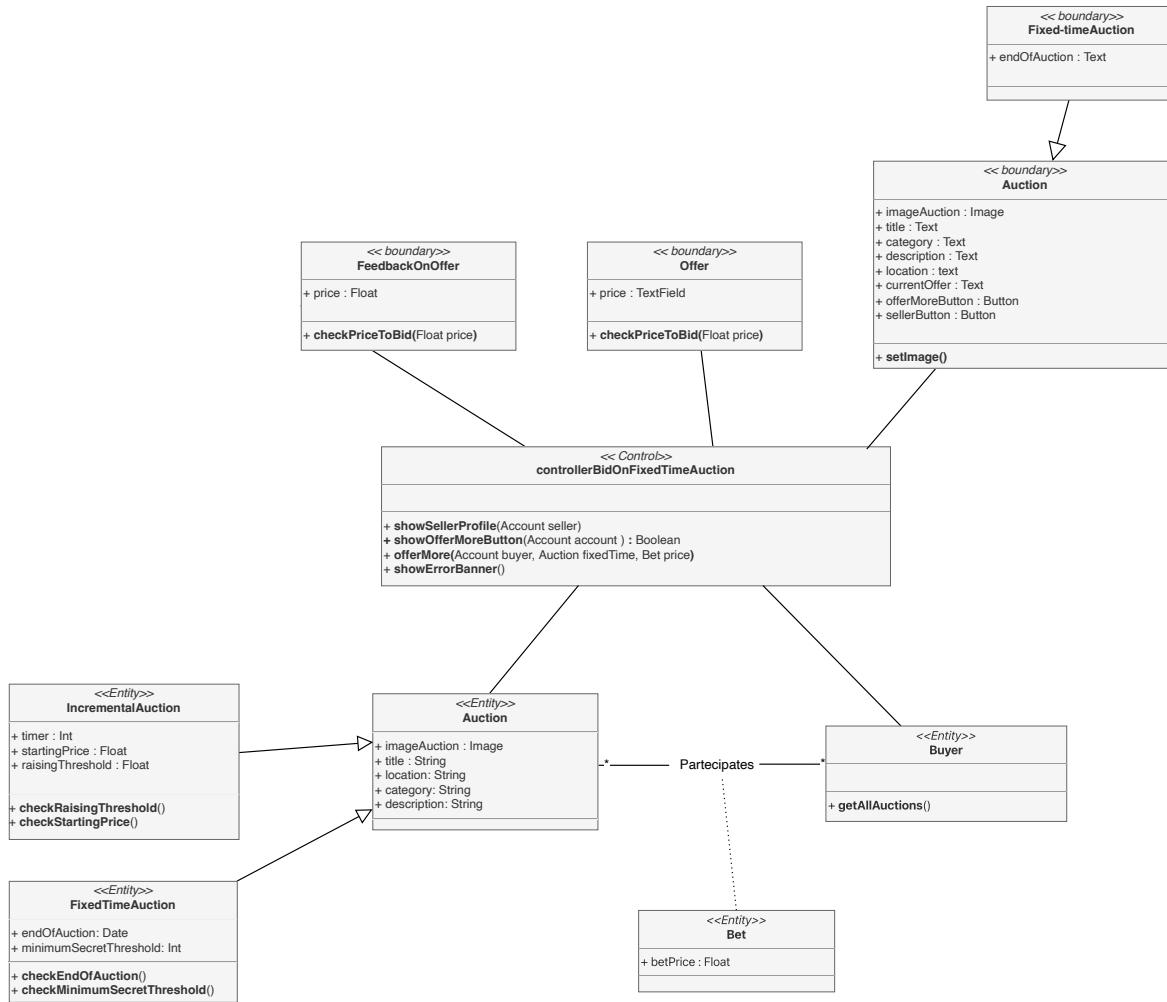


3.3.11 Mostra aste a cui si sta partecipando ed aste vinte.



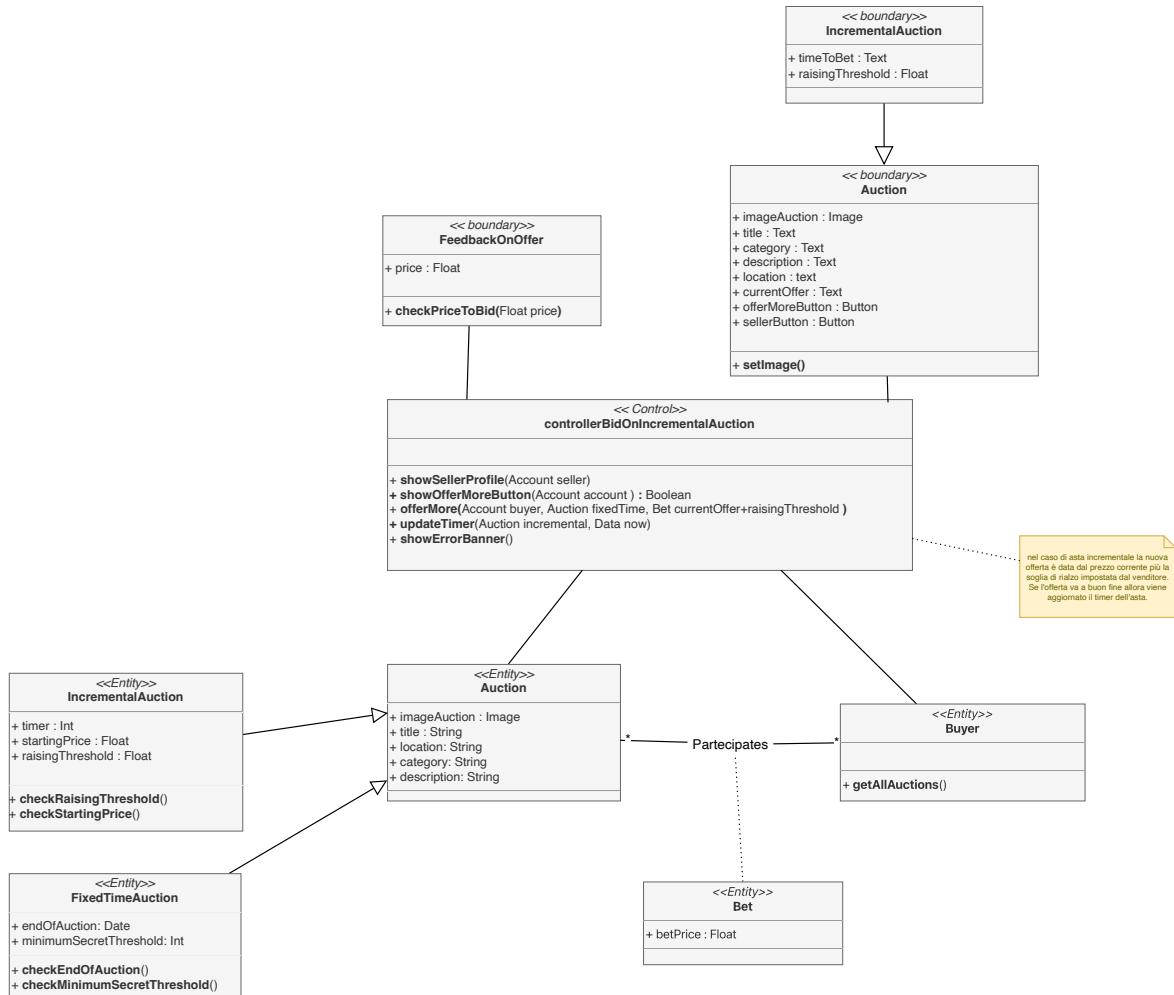
Questo caso d'uso è specifico per un utente di tipo compratore

3.3.12 Puntata su Asta a tempo fisso.



Questo caso d'uso è specifico per un utente di tipo compratore

3.3.13 Puntata su Asta Incrementale.



Questo caso d'uso è specifico per un utente di tipo compratore

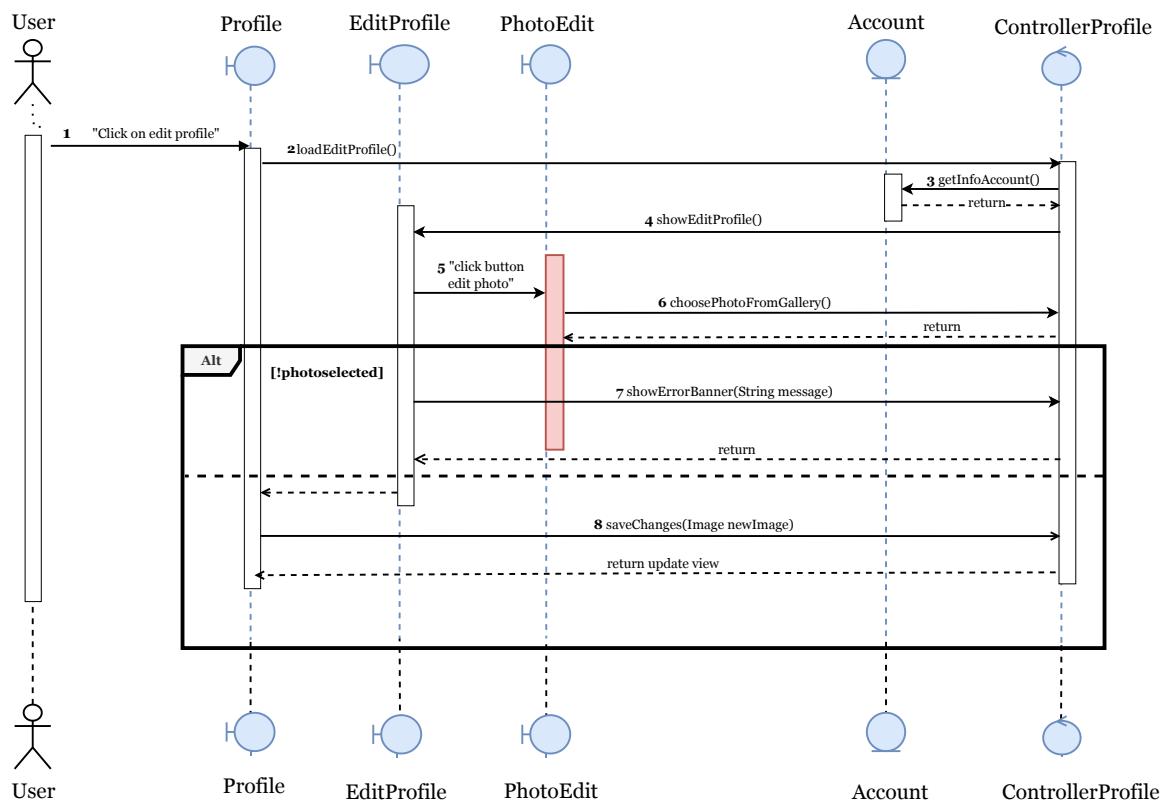
3.4 SEQUENCE DIAGRAM DI DUE CASI D'USO

I diagrammi di sequenza sono uno strumento fondamentale nella fase di specifica dei requisiti poiché forniscono una rappresentazione visuale delle **interazioni dinamiche** tra gli oggetti all'interno di un sistema. Essi enfatizzano il flusso di controllo e la collaborazione tra gli oggetti durante l'esecuzione di un particolare scenario di utilizzo. Di seguito sono riportati i sequence diagram che il team ha scelto di voler modellare.

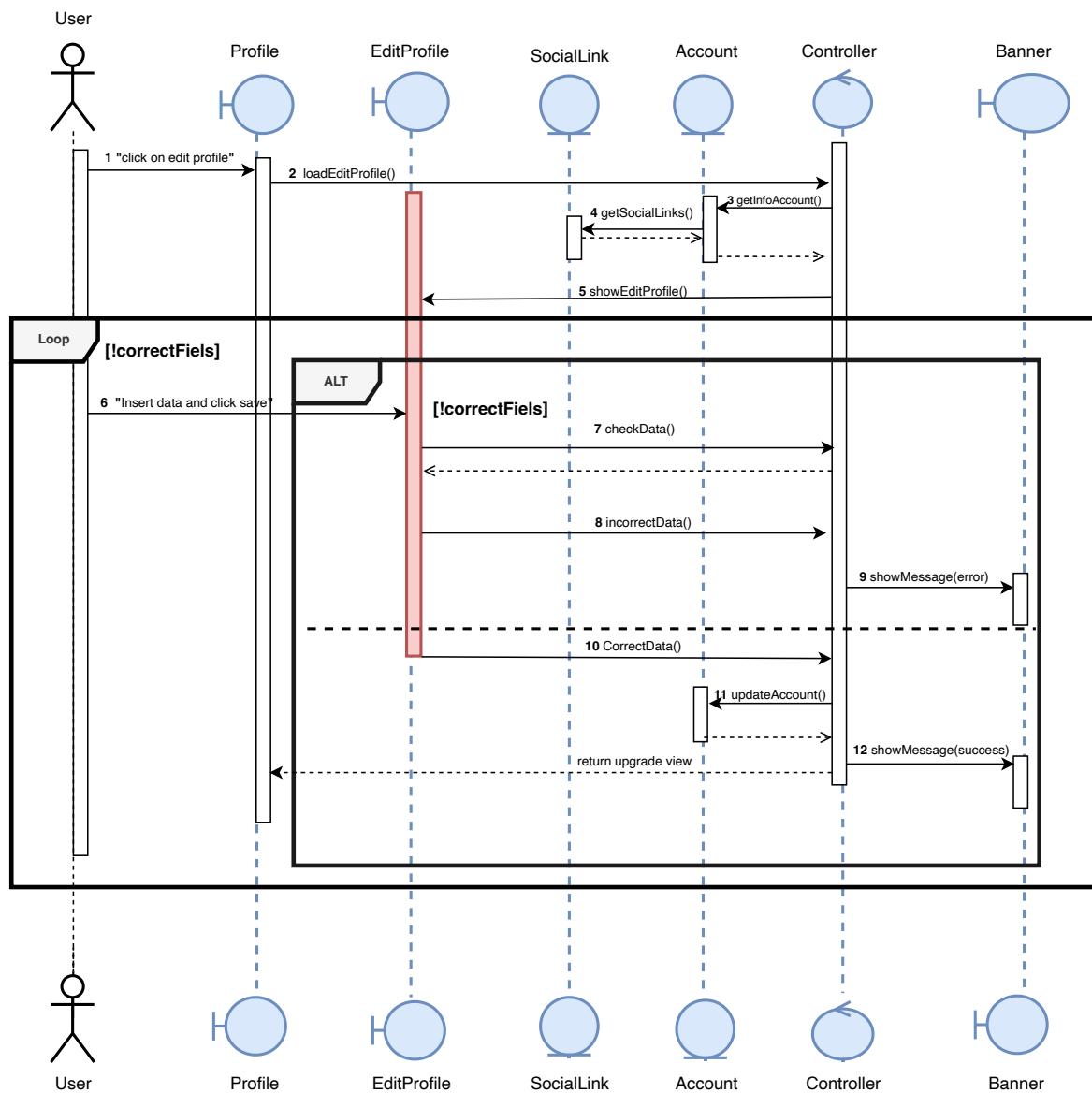
3.4.1 Caso d'uso : Personalizzazione del profilo

Siccome questo caso d'uso è complesso abbiamo deciso di rappresentare prima in dettaglio la modifica di un campo, in questo caso il cambio della foto profilo e di seguito invece descriviamo in maniera generale la modifica del profilo.

Modifica della foto profilo:

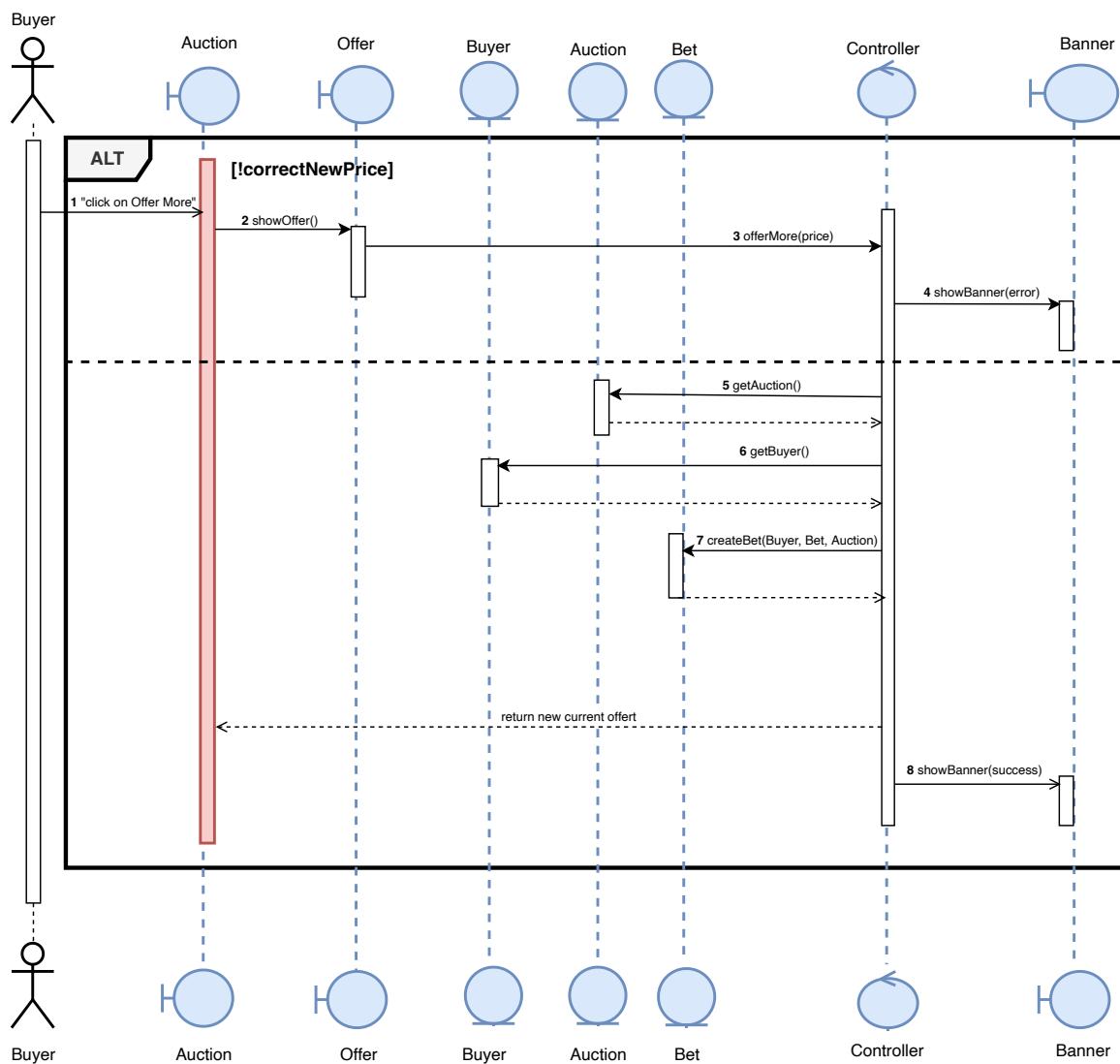


Personalizzazione del profilo:



La seguente immagine rappresenta dunque il processo iterativo di modifica di tutti i campi presenti nel proprio profilo personale.

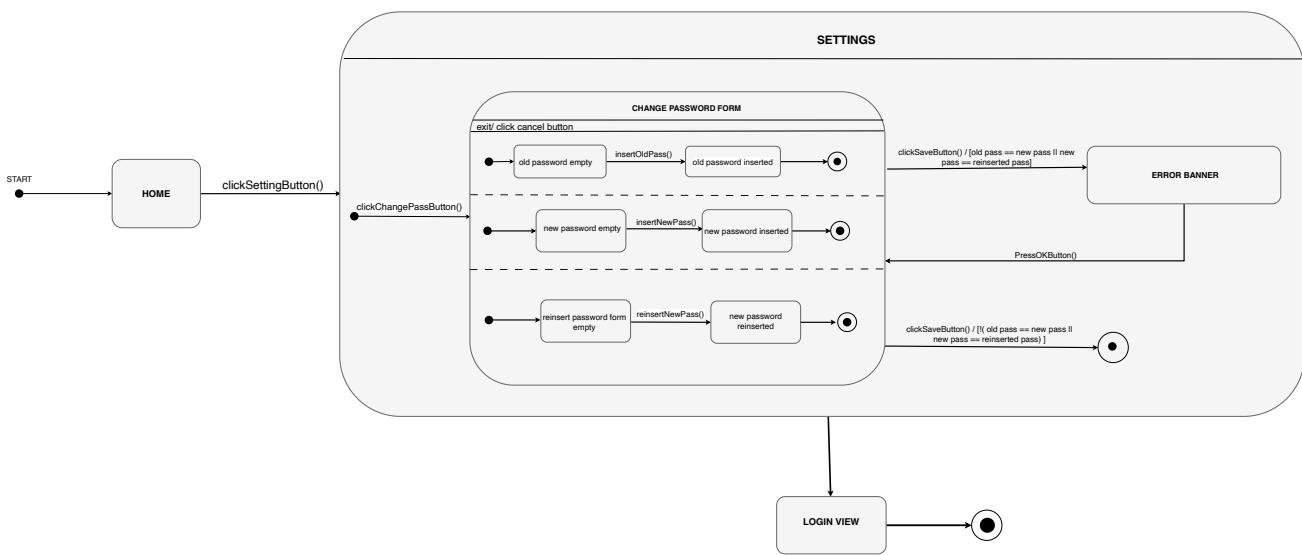
3.4.2 Caso d'uso : Puntare su un'asta a tempo fisso



3.5 STATECHART

I diagrammi Statechart sono uno strumento di modellazione visiva. Questi diagrammi rappresentano il comportamento dinamico di un sistema attraverso **stati**, **transizioni** e **azioni**. Ogni stato rappresenta una condizione del sistema, mentre le transizioni indicano il passaggio da uno stato all'altro in risposta a determinati eventi o condizioni. Le azioni possono essere associate sia agli stati che alle transizioni e rappresentano le attività eseguite quando il sistema si trova in uno stato specifico o durante il passaggio tra stati.

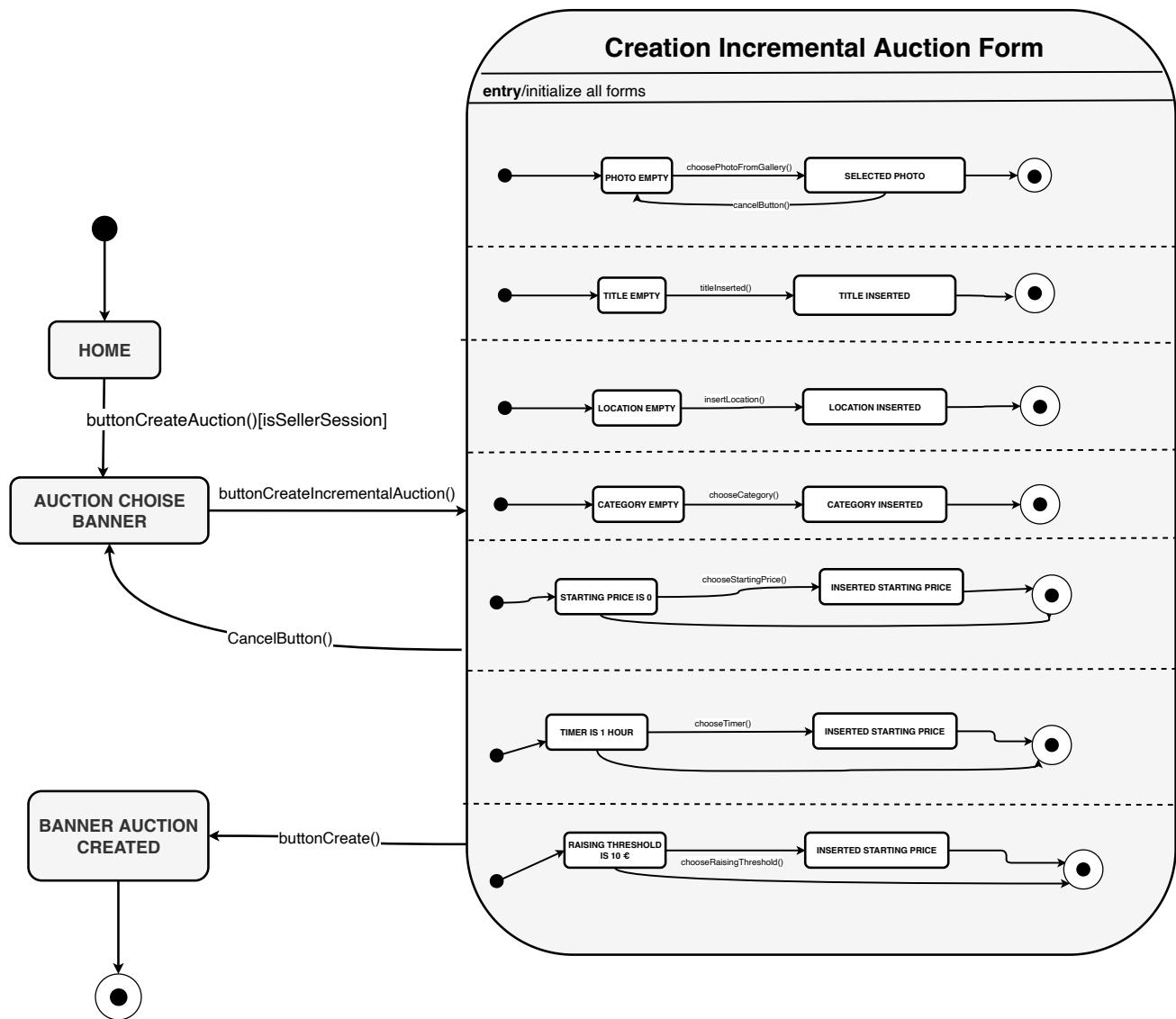
3.5.1 Statechart I : Cambio Password



Questo Statechart descrive il cambio di password di un utente. In particolare, delinea il comportamento dell'interfaccia grafica e gli errori possibili da parte dell'utente. Il processo inizia quando l'utente decide di cambiare la propria password e interagisce con l'interfaccia grafica. Dalla schermata principale, l'utente naviga alla sezione delle impostazioni e seleziona l'opzione per cambiare la password. Una volta all'interno del modulo di cambio password, l'utente è tenuto a inserire la vecchia password, la nuova password desiderata e confermare la nuova password. Il Statechart gestisce efficacemente questo processo, garantendo che l'utente fornisca tutte le informazioni necessarie. Durante questa fase, possono verificarsi diversi scenari. Se l'utente inserisce correttamente tutti i campi richiesti e la nuova password rispetta i criteri di sicurezza, il sistema procederà con il cambio della password e riporterà l'utente alla schermata di accesso (Login).

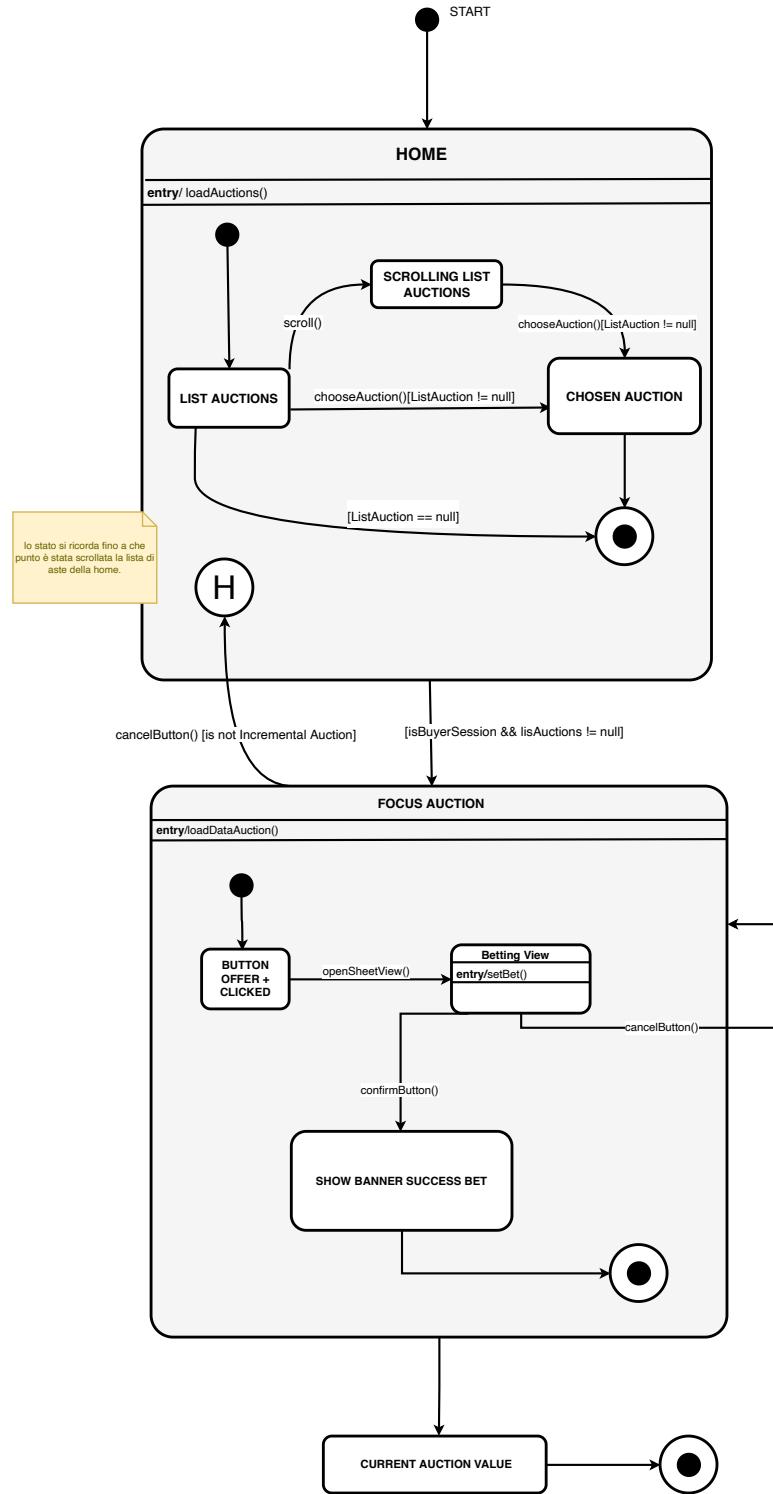
Tuttavia, se l'utente inserisce una nuova password che non corrisponde alla conferma della password o se la vecchia password fornita non è corretta, il Statechart gestisce questi casi come errori. In tal caso, viene visualizzato un messaggio di errore appropriato e l'utente viene invitato a correggere i campi necessari e a riprovare il processo di cambio password.

3.5.2 Statechart II : Creazione Asta Incrementale



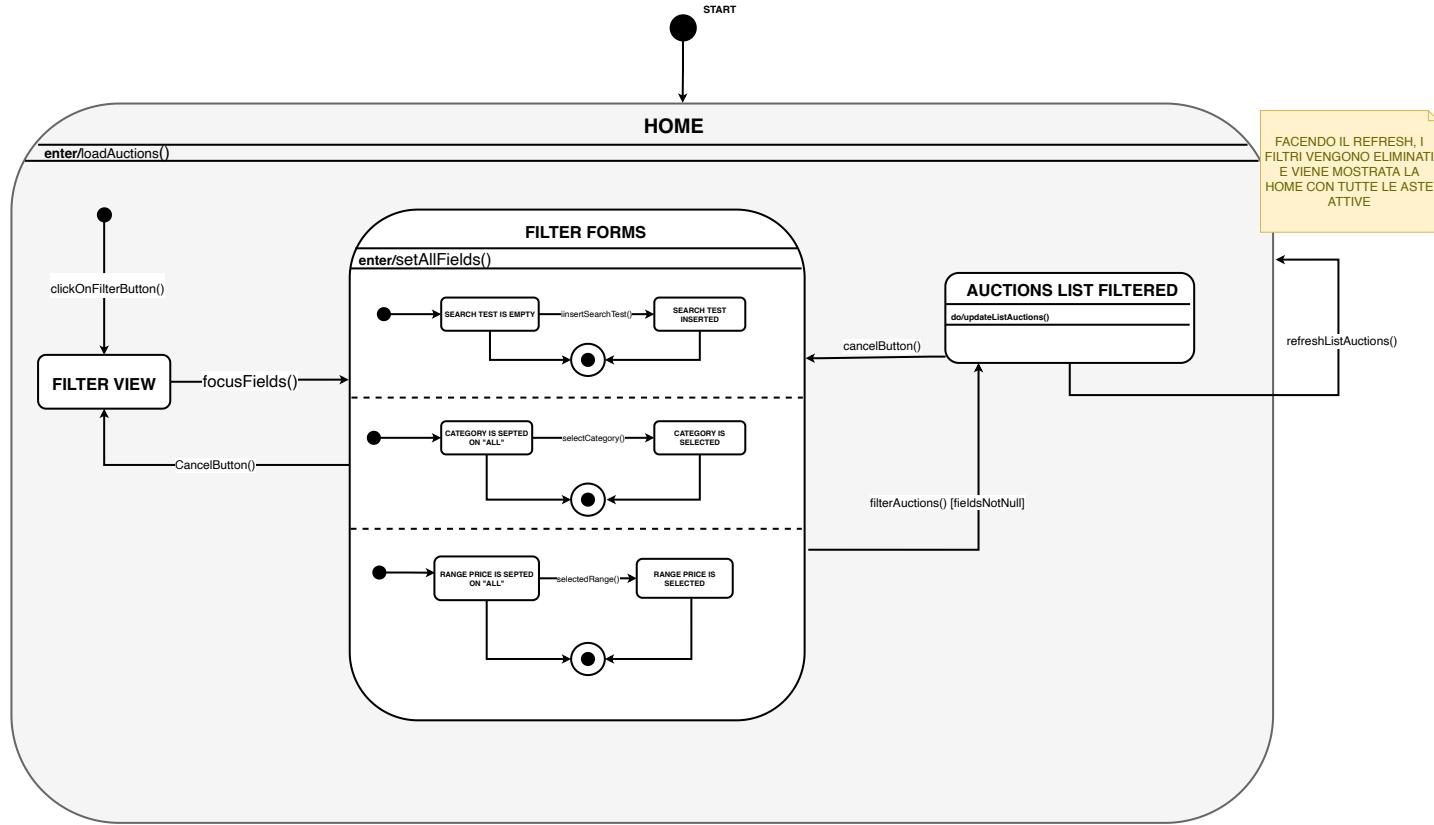
Questo Statechart descrive la creazione di un'asta di tipo incrementale. Una volta che l'utente ha compilato correttamente tutti i campi richiesti e ha confermato la creazione dell'asta, il sistema procederà con l'inizializzazione dell'asta incrementale, con relativo banner di successo della creazione.

3.5.3 Statechart III : Puntare su un'Asta Incrementale



Tale Statechart descrive come fare una puntata su un'asta di tipo Incrementale.

3.5.4 Statechart IV : Filtraggio Aste



Questo Statechart invece descrive come il sistema, attraverso l'interazione con l'utente elabora il filtraggio delle aste della Home. Non è stato messo uno stato finale per far intendere che l'operazione può essere ripetuta un numero indefinito di volte.

4 DOCUMENTO DI DESIGN

4.I ANALISI DELL'ARCHITETTURA

Il software sviluppato si presenta come un'architettura ***client-server***. Tale decisione è comunemente adottata per le applicazioni che richiedono una gestione centralizzata dei dati. I componenti in gioco sono il Client che mette a disposizione un'interfaccia semplice per gli utenti, mentre la logica e i dati sono elaborati dal Server. Per aumentare la **scalabilità** dell'applicazione, abbiamo utilizzato diversi servizi forniti da Amazon AWS, ad esempio *Amplify* e *Cognito* per garantire l'autenticazione degli utenti.

4.I.I Client

Il client è stato sviluppato utilizzando **Swift** in sincronia con il nuovo framework per sviluppo iOS **SwiftUI** nato nel 2019. Tale decisione nasce dalla volontà di acquisire competenze in un nuovo linguaggio di programmazione e dalla determinazione nel dimostrare ai colleghi che hanno svolto l'Apple Academy che non è indispensabile seguire corsi specifici per padroneggiare nuove tecnologie.

E' stato scelto il pattern **MVVM** (Model-View-ViewModel). La scelta ricade su questo design in quanto il suo utilizzo presenta vantaggi in diversi aspetti:

- **Separazione delle responsabilità:** MVVM permette una chiara separazione tra la logica di business (Model), la presentazione dell'interfaccia utente (View) e la gestione degli input utente (ViewModel).
- **Riutilizzo del codice:** La separazione dei componenti consente il riutilizzo del ViewModel in più View o in contesti diversi, facilitando la manutenibilità e la scalabilità del codice.
- **Testabilità:** La logica di business contenuta nel ViewModel può essere testata in modo indipendente dalla View, semplificando la scrittura di test unitari.
- **Binding e reattività:** SwiftUI offre un meccanismo di binding potente che si integra bene con MVVM, semplificando la sincronizzazione dei dati tra ViewModel e View.

Ogni ViewModel, oltre a gestire la logica di presentazione e interazione dell'interfaccia utente, è progettato con una dipendenza da un'interfaccia **API** per facilitare la comunicazione con il server. Questo approccio consente una *maggiore modularità e separazione delle responsabilità nel codice*, in quanto il ViewModel si concentra esclusivamente sulla manipolazione dei dati e sullo stato dell'applicazione, delegando le operazioni di rete e comunicazione al livello dell'interfaccia API.

Nota: Durante il processo di sviluppo, è stato fatto riferimento al lavoro di un collega, Alessandro Zechini, più in particolare la sua tesi di laurea magistrale per l'utilizzo di questo pattern in Swift. riferimento

4.1.2 Server

Per la creazione del backend, abbiamo utilizzato Jakarta RESTful Web Services, è un framework Java che fornisce un'API per la creazione di servizi Web RESTful. REST (Representational State Transfer) è uno stile architettonico per la progettazione di servizi Web che si basa sui principi fondamentali della trasmissione dei dati tramite HTTP.

Il server comunica con un database Postgresql attraverso l'implementazione di interfacce **DAO**

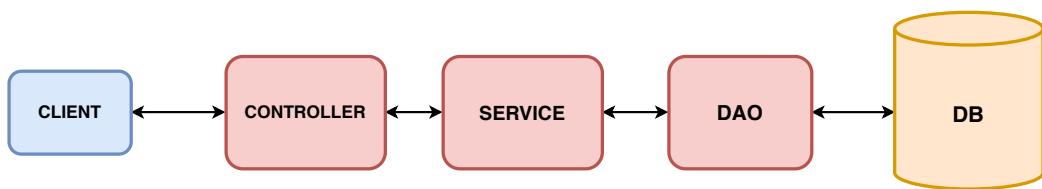
Architettura backend:

Si è scelto di suddividere l'architettura del Server in 3 layer : Controller, Service e DAO.

- **Controller:** questo package si occupa di fare il routing delle richieste HTTP e di gestire le richieste da inviare e ricevute dal Client.
- **Service:** Contiene la logica di business dell'applicazione. I servizi offerti dal backend sono implementati all'interno di questa componente.
- **DAO (Data Access Object):** Responsabile dell'accesso ai dati nel database. Contiene metodi per l'interazione con il database, inclusi inserimenti, aggiornamenti, cancellazioni e letture. La sua implementazione è stata fatta attraverso *JDBC*.

Ogni layer ha una specifica responsabilità all'interno dell'architettura, garantendo una separazione dei compiti e una maggiore **manutenibilità** del codice.

Struttura schematizzata :



Con questo schema viene riassunto il modo di comunicare fra i vari componenti del software, rendendo la visione del suo funzionamento più semplice e chiara.

4.2 SCELTE TECNOLOGICHE ADOTTATE :

In questa sezione verranno analizzate le *scelte tecnologiche* adottate per la realizzazione del sistema. In particolare, verranno esaminate le tecnologie utilizzate per lo sviluppo del client e del server, insieme alle motivazioni che hanno guidato tali scelte e alle potenziali alternative considerate durante il processo decisionale.

4.2.1 Servizi Cloud usati:

- **Amazon EC2:** *Amazon Elastic Compute Cloud* (EC2) è un servizio di cloud computing fornito da Amazon Web Services (AWS) che consente di lanciare e gestire istanze di macchine virtuali. Abbiamo scelto questo servizio perché possiamo facilmente scalare le risorse computazionali in base alle esigenze dell'applicazione. Inoltre, EC2 offre una vasta gamma di tipi di istanze con diverse configurazioni hardware, permettendoci di scegliere quella più adatta alle nostre esigenze di prestazioni e costo.
- **Amazon S3:** *Amazon Simple Storage Service* (S3) è un servizio di storage di oggetti che consente di archiviare e recuperare dati di qualsiasi dimensione in modo sicuro ed efficiente. Abbiamo utilizzato Amazon S3 per memorizzare file statici come immagini, video e altri asset dell'applicazione, poiché offre una durabilità elevata, scalabilità e facilità di integrazione con altri servizi AWS.
- **Amplify:** *AWS Amplify* è un insieme di strumenti e servizi che semplificano lo sviluppo di app per dispositivi mobili e web. Offre funzionalità per autenticazione degli utenti, gestione degli aggiornamenti dell'app e altro ancora. Abbiamo utilizzato Amplify per semplificare lo sviluppo e la gestione delle nostre applicazioni, garantendo al contempo elevate prestazioni e scalabilità.
- **Amazon Cognito:** Amazon Cognito è un servizio di AWS che gestisce *l'autenticazione e l'autorizzazione degli utenti* per le applicazioni web e mobili. Fornisce funzionalità di registrazione, accesso e recupero password, insieme alla gestione delle autorizzazioni per gli utenti autenticati. Abbiamo integrato Amazon Cognito per garantire un'autenticazione sicura e affidabile degli utenti nelle nostre applicazioni.

4.2.2 Altri servizi esterni usati:

- **Alamofire:** Alamofire è una libreria HTTP per Swift, che semplifica le richieste di rete su iOS, macOS, watchOS e tvOS. Abbiamo utilizzato Alamofire per gestire le richieste di rete nel nostro progetto, garantendo una comunicazione efficiente e affidabile con i servizi backend.
- **Swift Algorithm Club:** Swift Algorithm Club è una raccolta di algoritmi e strutture dati implementati in Swift. Abbiamo consultato Swift Algorithm Club per ottenere implementazioni efficienti di algoritmi fondamentali, contribuendo così alla nostra logica di business e alla risoluzione di problemi nel nostro progetto.

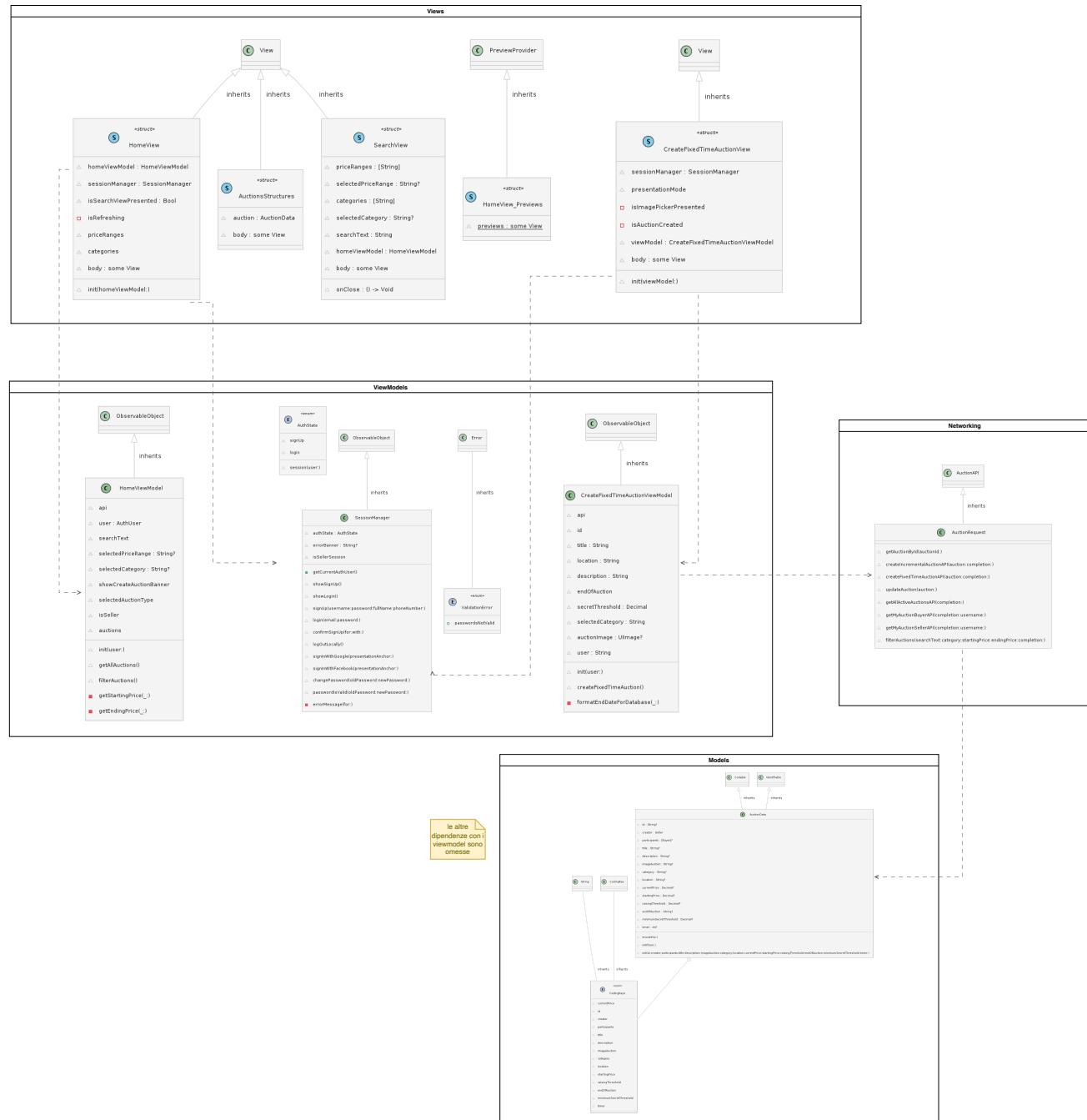
4.3 DIAGRAMMI UML DELLE CLASSI DI DESIGN

In questa sezione analizziamo tutti i requisiti funzionali individuati nella prima fase, esplicitandoli attraverso i diagrammi UML per rappresentare le soluzioni adottate per ognuno di essi. Come ideato nella fase della specifica e analisi dei requisiti, siccome si è pensati di seguire EBC, lo sviluppo è stato fatto suddividendo i vari componenti su tre layer differenti, seguendo i principi del pattern *MVVM*

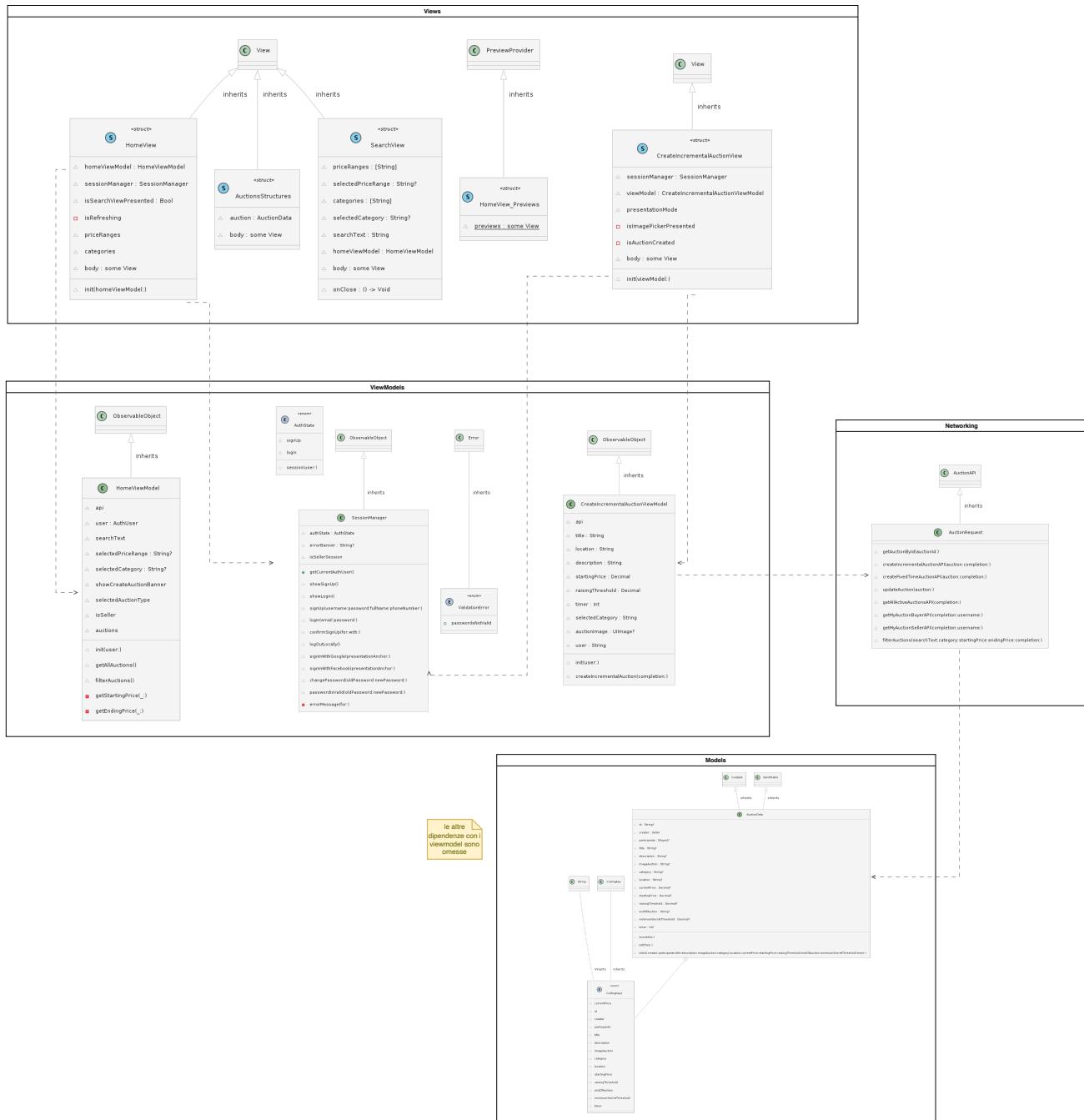
4.3.1 Registrazione al sistema tramite mail, Facebook, Google, Apple e Login al sistema



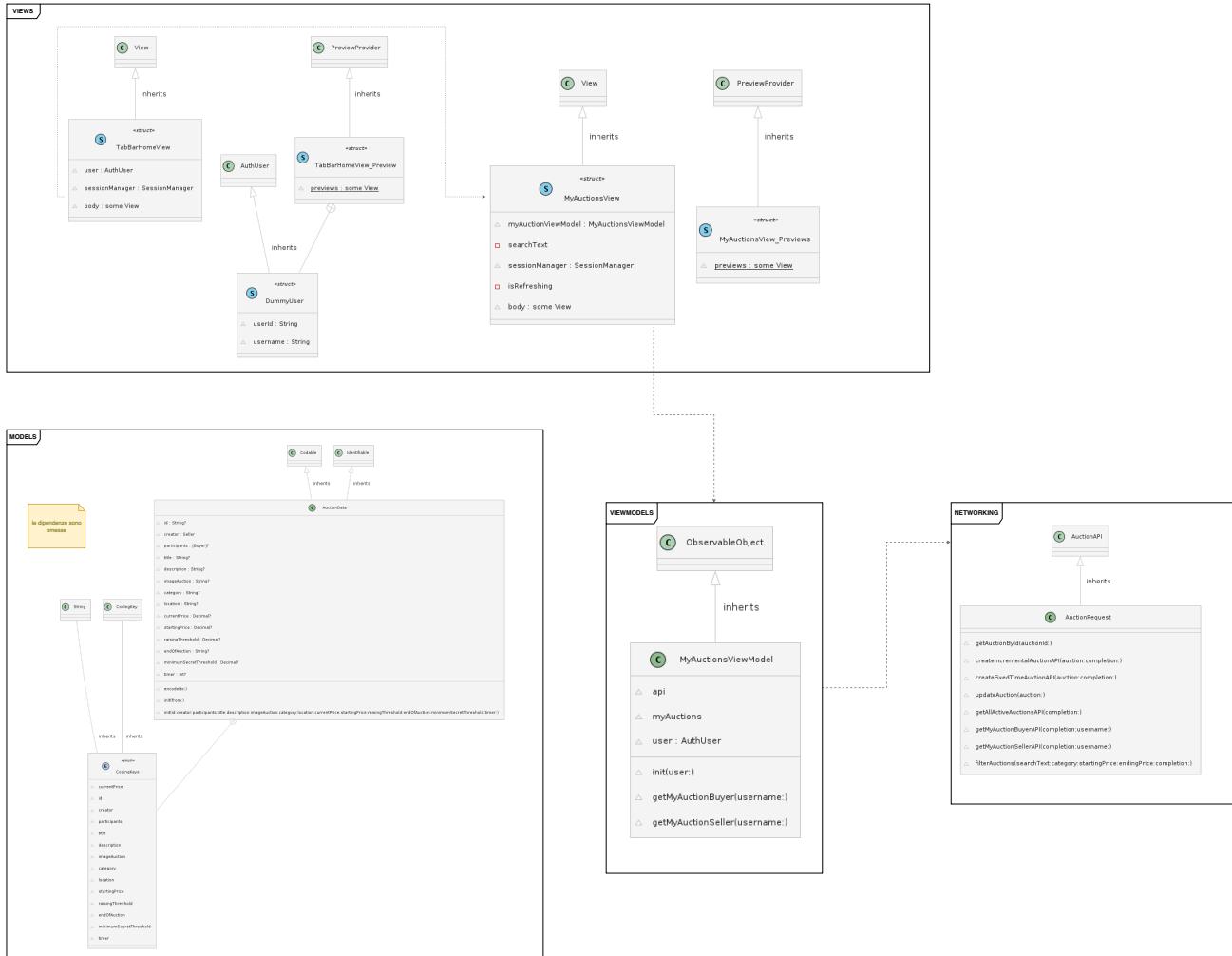
4.3.2 Creazione Asta a Tempo fisso



4.3.3 Creazione Asta Incrementale



4.3.4 Mostra storico Aste Create



Nota: il diagramma per mostrare le aste a cui si partecipa e le aste vinte del compratore ha come rappresentazione lo stesso diagramma, ciò che cambia è la gestione che ne fa *SessionManager*.

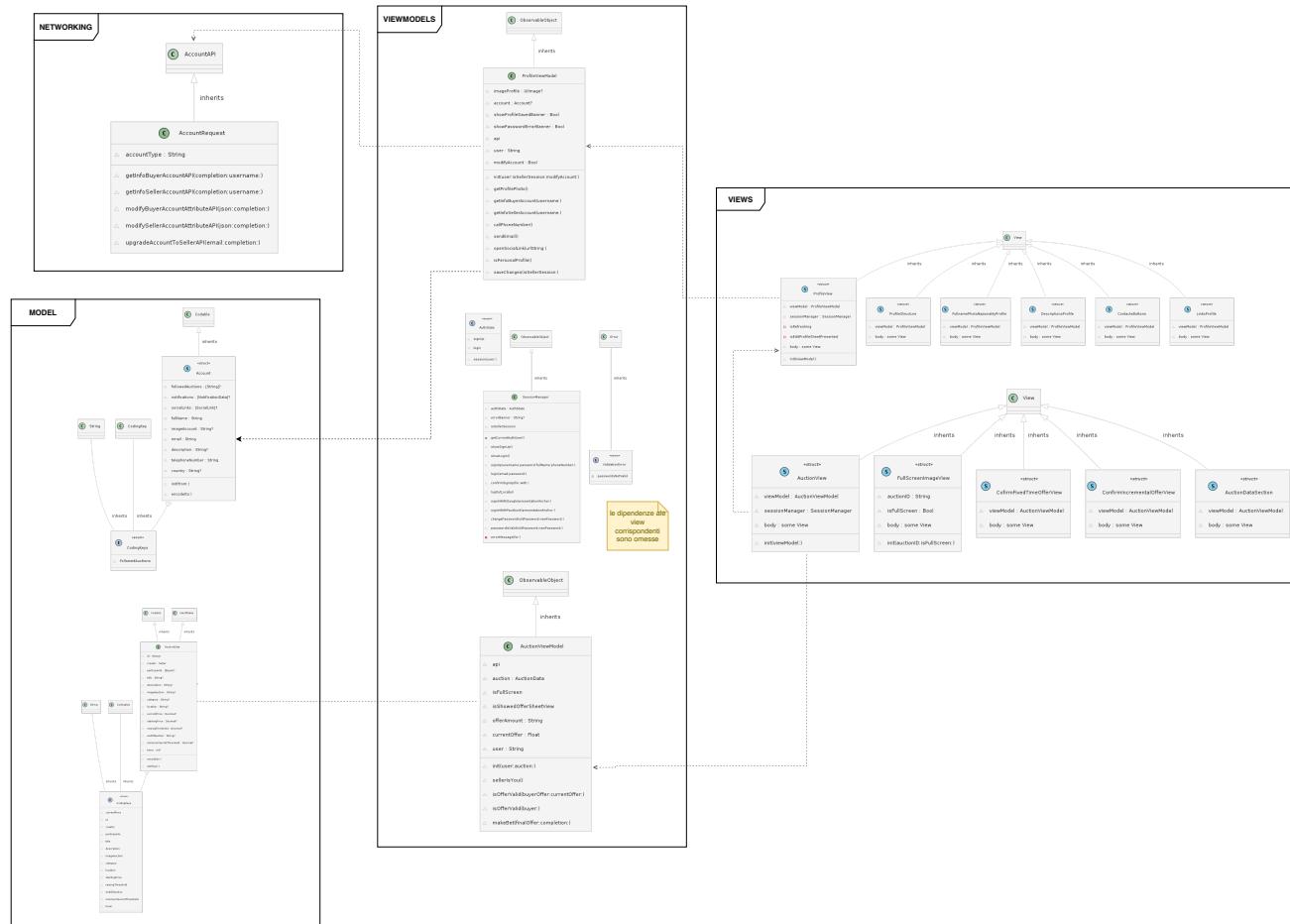
4.3.5 Mostra aste correttamente attive nel sistema con ricerca e filtro



4.3.6 Personalizzazione del Profilo e cambio password

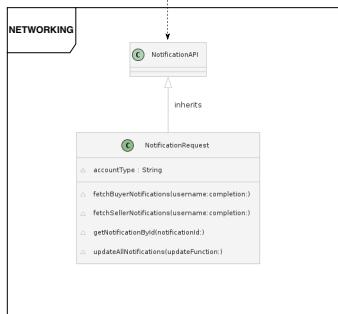
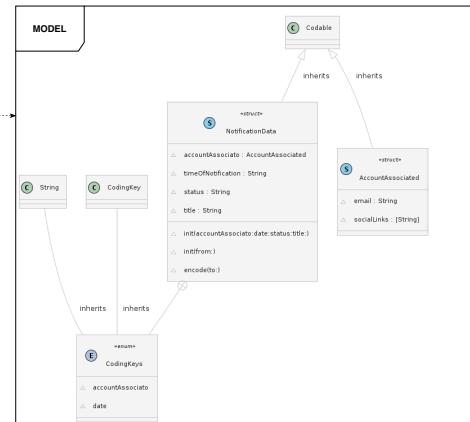
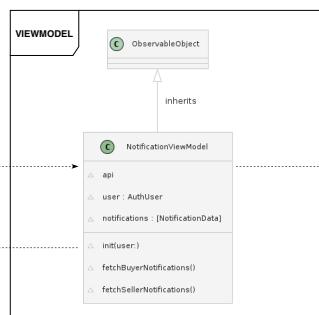
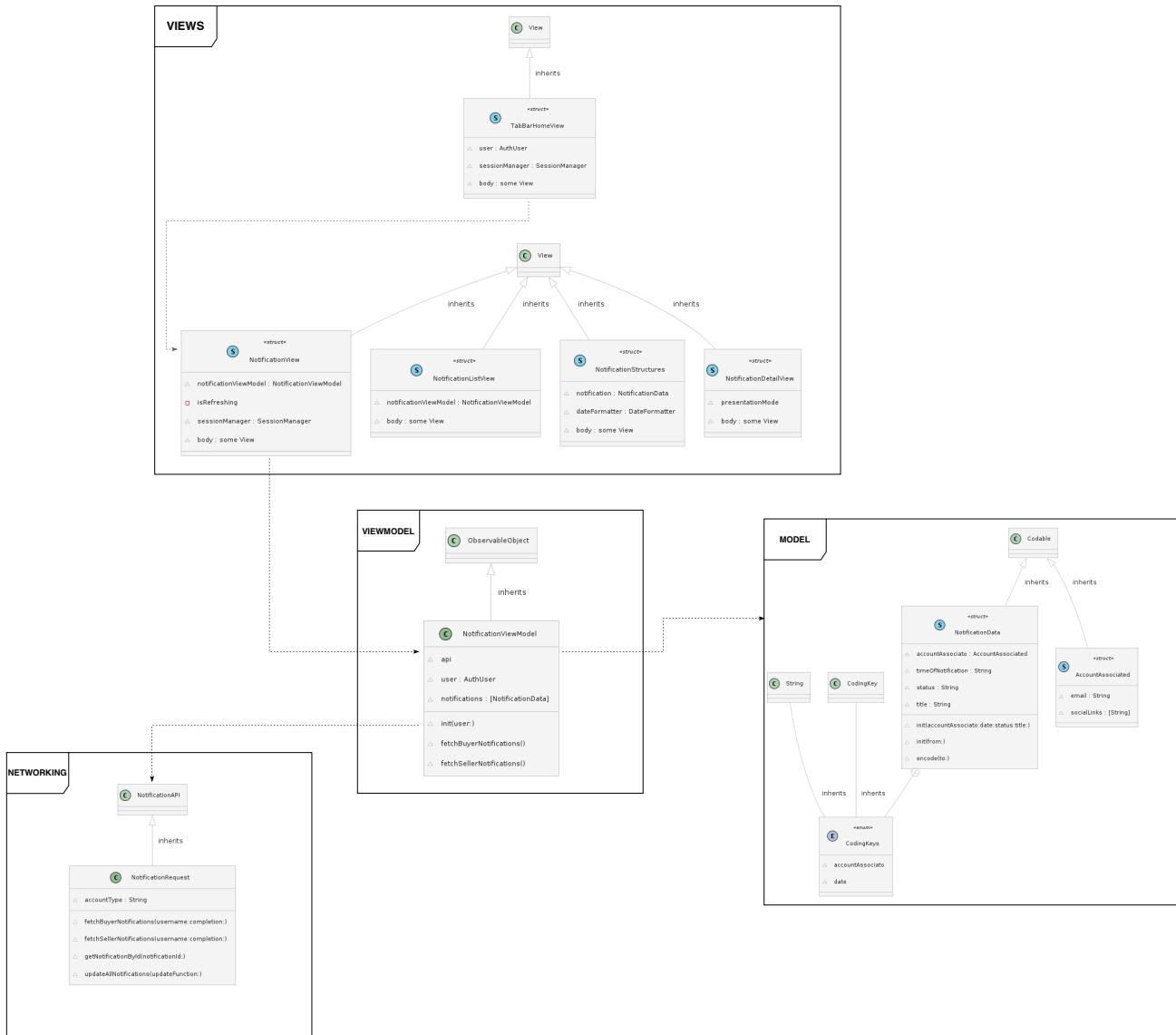


4.3.7 Mostra profilo venditore di un'asta specifica



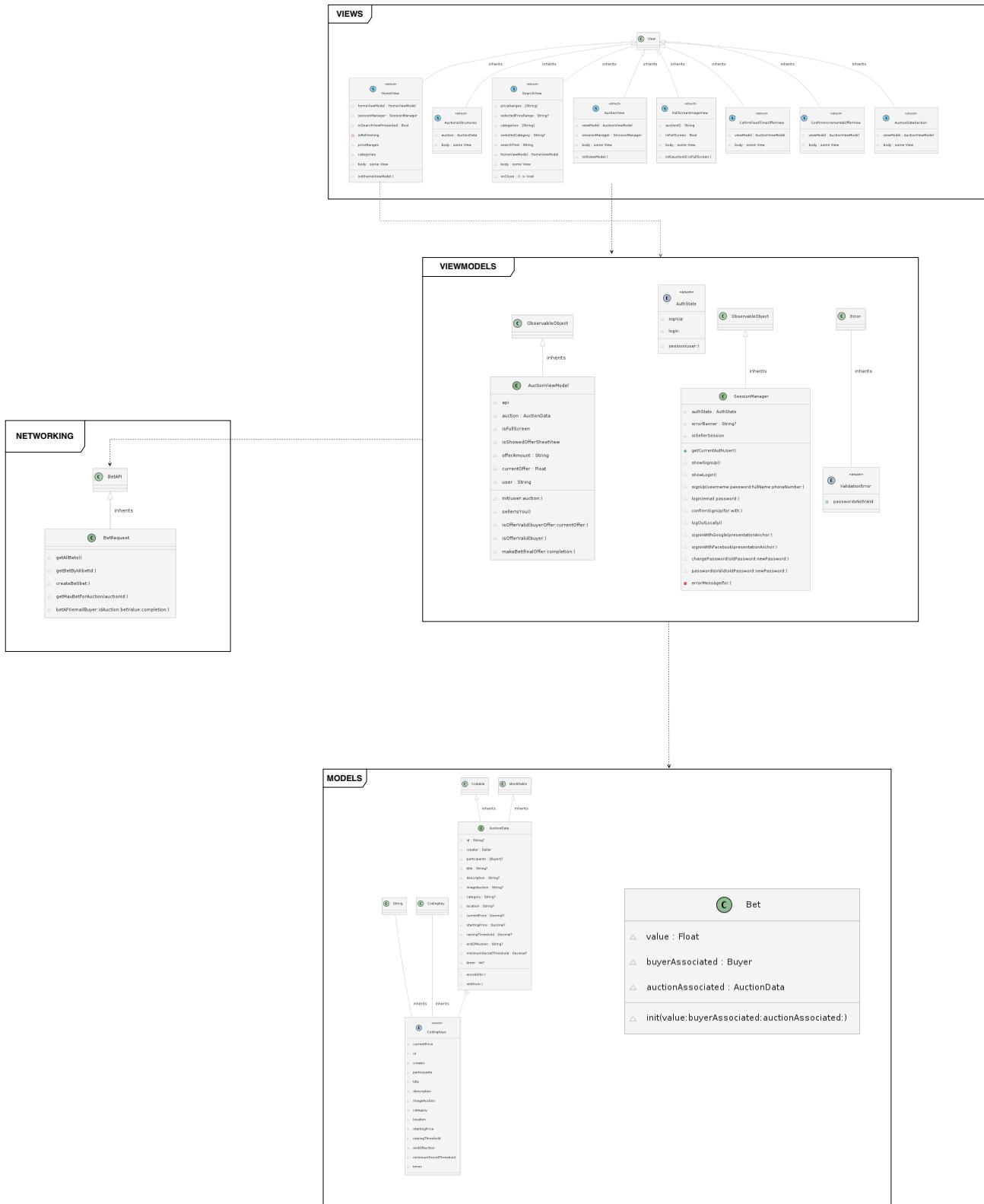
Nota: l'utente che vuole vedere il profilo venditore di un'asta, in primis mette a fuoco l'asta desiderata poi clicca sul profilo del venditore, ciò scaturisce la chiamata http al server che recupera i dati che vengono inviati al *ProfileViewModel* che automaticamente aggiorna la view.

4.3.8 Visualizzazione delle notifiche



Nota: la visualizzazione delle notifiche è uguale sia per compratore che per venditore

4.3.9 Punta su un'Asta (Incrementale/Tempo fisso)



Vorrei esprimere la mia più sincera gratitudine all'autore della repository su GitHub che mi ha permesso di generare i diagrammi da Xcode utilizzando un plug-in che sfrutta PlantUML. Questo strumento ci ha permesso non solo di velocizzare lo sviluppo della documentazione in maniera gratuita ma anche quello di avere una visione d'insieme e ragionare man mano sui vari componenti del front-end;

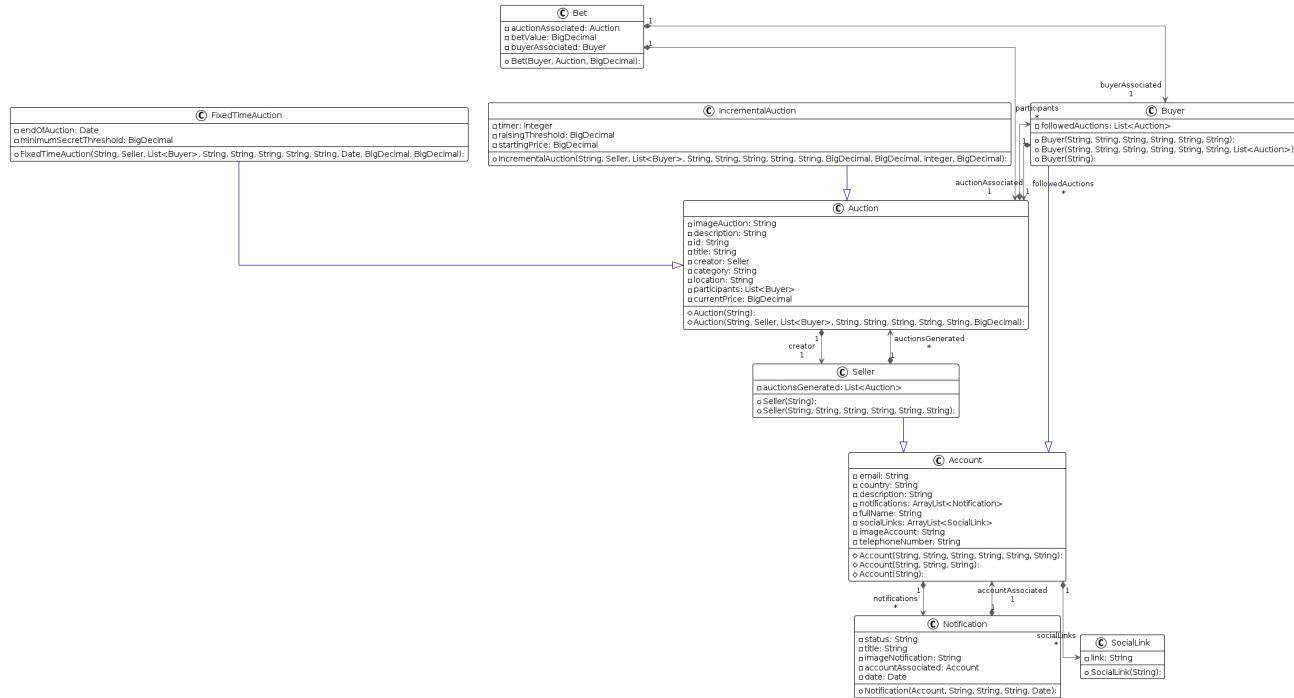
Il link della repository è il seguente: <https://github.com/MarcoEidinger/SwiftPlantUML-Xcode-Extension>

4.4 DIAGRAMMI UML DEL SERVER

In questa sezione analizziamo i diagrammi relativi al back-end, in particolare descriviamo la struttura del Server.

4.4.1 Model

Per completezza e avere una visione più chiara di come vengono gestiti i dati e la logica di business del software, si è deciso di inserire anche il diagramma UML del *model*. Questo diagramma offre una panoramica dettagliata delle strutture dati e delle loro relazioni, fornendo un quadro completo del sistema backend.

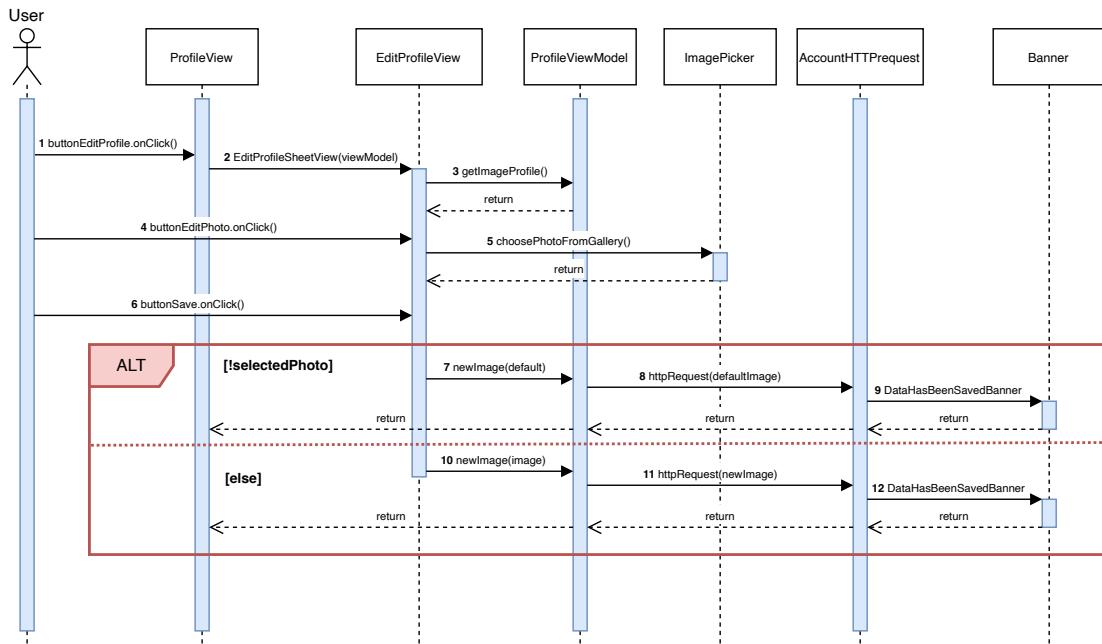


Nota: Il model del backend e frontend non corrispondono in quanto il backend gestisce la logica di business e l'accesso ai dati, mentre il frontend si concentra sulla presentazione e l'interazione con l'utente. Di conseguenza, le strutture dei dati e le loro rappresentazioni possono differire tra i due livelli per rispondere alle rispettive esigenze specifiche. In particolare, il model del backend è progettato per ottimizzare l'efficienza delle operazioni di lettura e scrittura nei database, garantendo coerenza e integrità dei dati, mentre il frontend può avere strutture più orientate alla facilità di manipolazione e visualizzazione dei dati.

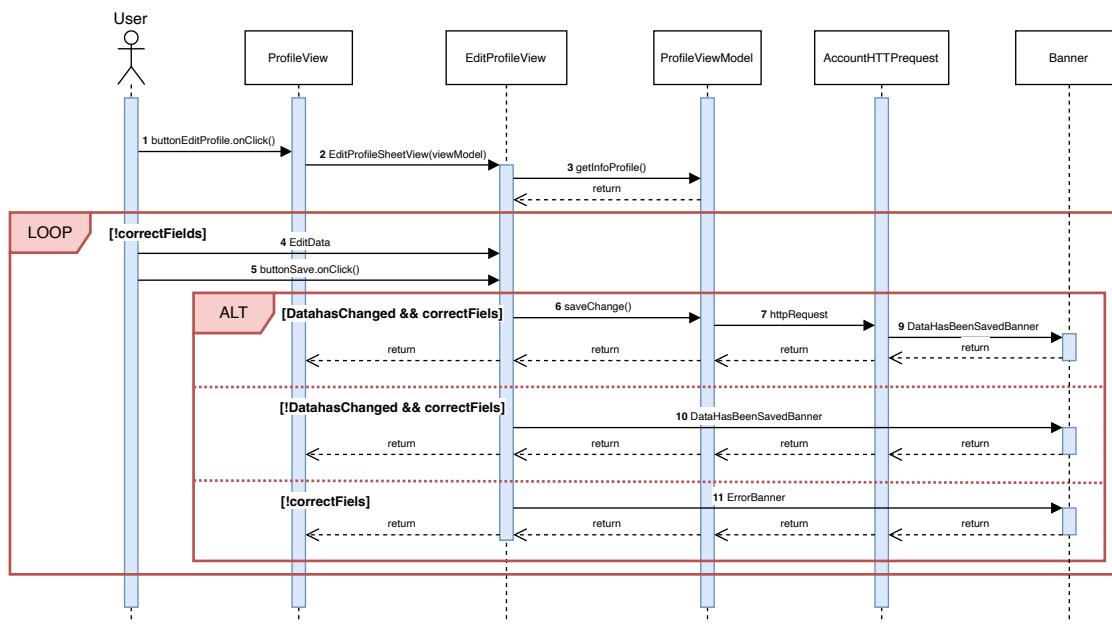
4.5 DIAGRAMMI UML DI SEQUENZA

4.5.1 Caso d'uso: Personalizzazione del profilo

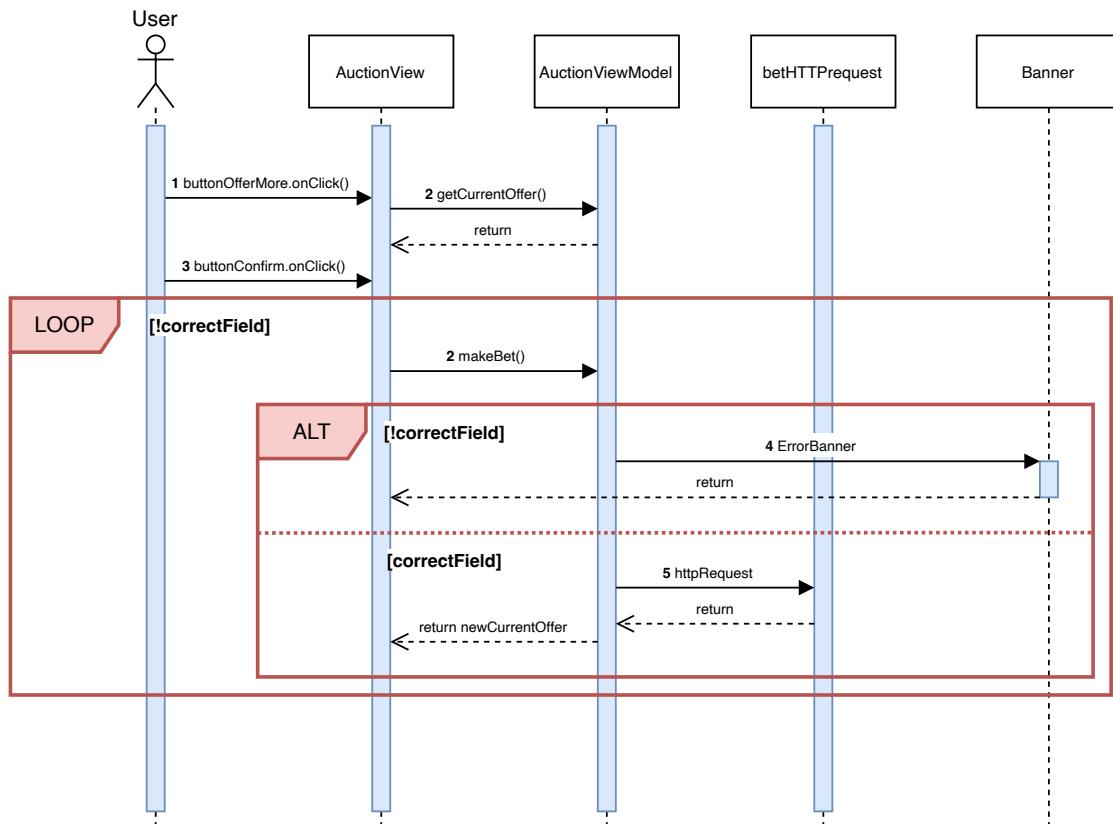
Modifica della foto profilo:



Personalizzazione del profilo:



4.5.2 Caso d'uso : Puntare su un'asta a tempo fisso



DOCKERFILE E FILE DI BUILD AUTOMATICA

In questa sezione analizziamo il Dockerfile e il processo di build automatica per configurare e distribuire l'applicazione backend in un ambiente Dockerizzato.

5.1 DOCKERFILE

Il seguente Dockerfile definisce i passaggi per creare un'immagine Docker basata su Ubuntu, configurare l'ambiente e avviare un'applicazione Java backend:

```

1 FROM ubuntu:latest
2 RUN apt update && apt-get update
3 RUN apt-get install -y maven openjdk-8-jdk
4 COPY . /usr/src/myapp
5 WORKDIR /usr/src/myapp
6 RUN mvn install
7 EXPOSE 8080
8 CMD ["java", "-jar", "target/backend.jar"]

```

5.2 DOCKER COMPOSE

```

1 version: '3.8'
2
3 services:
4   postgres:
5     image: postgres:latest
6     restart: always
7     environment:
8       POSTGRES_DB: postgres
9       POSTGRES_USER: postgres
10      POSTGRES_PASSWORD: indovinato
11     volumes:
12       - ./init.sql:/docker-entrypoint-initdb.d/init.sql # Monta il
13         file init.sql nella directory di inizializzazione
14     ports:
15       - "5432:5432" # Mappa la porta del container PostgreSQL sulla
16         porta della macchina host
17
18   backend:
19     build:
20       context: ./ # Percorso del Dockerfile del backend
21     restart: always
22     ports:
23       - "80:8080" # Mappa la porta del container del backend sulla
24         porta della macchina host

```

```

22  depends_on:
23    - postgres # Dipendenza dal servizio PostgreSQL
24
25  environment:
26    POSTGRES_DB: postgres
27    POSTGRES_USER: postgres
28    POSTGRES_PASSWORD: indovinato
29    POSTGRES_HOST: postgres
30
31  init-script:
32    image: postgres:latest
33    volumes:
34      - ./init.sql:/scripts/init.sql # Monta il file init.sql nella
35        directory di esecuzione
36    entrypoint: /bin/bash
37    command: >
38      -c "
39        until pg_isready -h postgres -U postgres; do
40          echo 'Waiting for postgres...';
41          sleep 5;
42        done;
43        PGPASSWORD=indovinato psql -h postgres -U postgres -d postgres -f
44        /scripts/init.sql
45      "
46
47  depends_on:
48    - postgres

```

Questo file *Docker Compose* configura tre servizi principali:

- **postgres**: utilizza l'immagine di PostgreSQL per fornire un database persistente. Viene configurato con un database, un utente e una password specifici, oltre a montare uno script di inizializzazione SQL. (Questo servizio gestisce la persistenza dei dati)
- **backend**: definisce il servizio per l'applicazione backend, costruito utilizzando il Dockerfile specificato nel contesto corrente ('./'). Espone la porta **8080** per consentire l'accesso all'applicazione e dipende dal servizio PostgreSQL per il corretto funzionamento. (Questo servizio gestisce la logica di business dell'applicazione)
- **init-script**: utilizza l'immagine di PostgreSQL per eseguire uno script di inizializzazione. Attende il completamento del servizio PostgreSQL prima di eseguire lo script SQL. (Questo servizio esegue operazioni iniziali di setup)

Questo Docker Compose semplifica la gestione di più container Docker, consentendo un **deployment** efficiente e scalabile dell'applicazione.

6 QUALITÀ DEL CODICE

La qualità del codice è un aspetto fondamentale nello sviluppo software. Un codice di alta qualità non solo facilita la manutenzione e l'evoluzione del sistema, ma contribuisce anche alla sua affidabilità, efficienza e sicurezza. L'adozione di pratiche di codifica di alta qualità riduce il rischio di bug e vulnerabilità, rendendo il software più robusto e meno incline a fallimenti imprevisti. Inoltre, un codice ben scritto e documentato è più comprensibile e modificabile, migliorando la collaborazione tra i membri del team e facilitando il lavoro dei nuovi sviluppatori che si uniscono al progetto.

Mantenere un alto standard di qualità del codice richiede strumenti e metodologie efficaci per la verifica e l'analisi continua del codice. Uno di questi strumenti è **SonarQube**, che abbiamo scelto per analizzare il server scritto in JAX-RS per diverse ragioni chiave.

6.I SONARQUBE

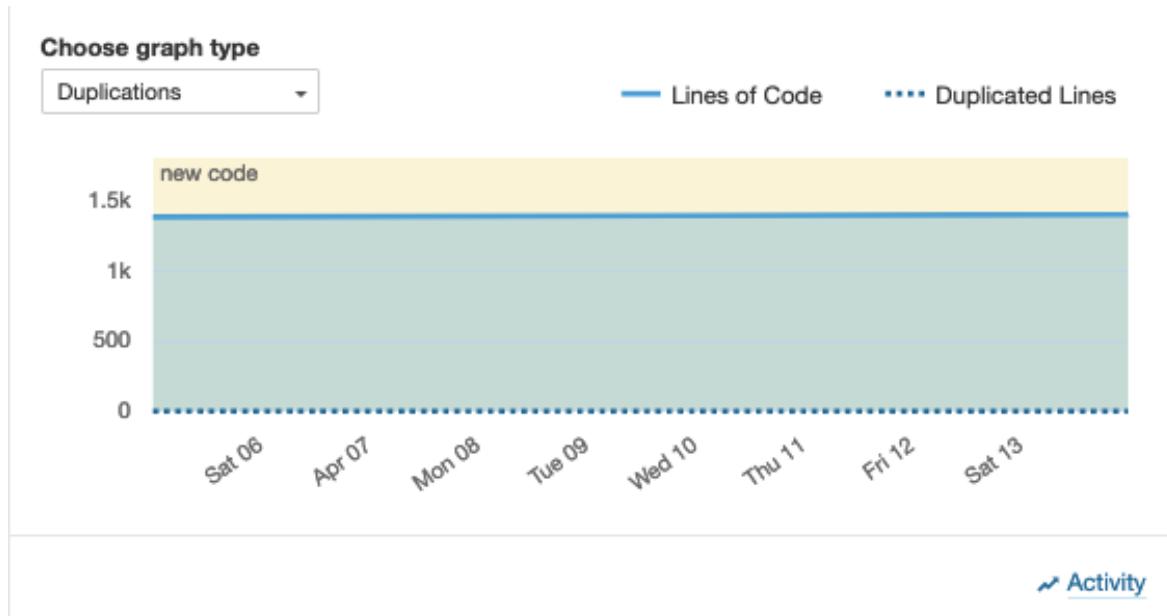
SonarQube è una piattaforma di analisi della qualità del codice *open-source* ampiamente utilizzata, che offre una gamma completa di funzionalità per monitorare e migliorare continuamente la qualità del codice. Ecco alcune delle principali ragioni per cui abbiamo scelto SonarQube:

- **Copertura Multilingua:** SonarQube supporta oltre 25 linguaggi di programmazione, tra cui Java, il linguaggio principale utilizzato per il nostro server JAX-RS. Questo ci permette di analizzare il nostro codice in modo approfondito e di ricevere feedback su una vasta gamma di problematiche.
- **Analisi Statica del Codice:** SonarQube esegue un'analisi statica del codice, identificando bug, vulnerabilità e problemi di manutenzione prima che possano causare problemi in fase di produzione. Questo è essenziale per prevenire errori e migliorare la sicurezza del software.
- **Indicatori di Qualità:** La piattaforma offre una serie di indicatori di qualità, come il code smell, la copertura dei test, la duplicazione del codice e la complessità ciclomatica. Questi indicatori aiutano a mantenere il codice pulito e manutenibile, migliorando la leggibilità e riducendo i costi di manutenzione.

Utilizzando SonarQube per l'analisi del nostro server JAX-RS, siamo in grado di mantenere elevati standard di qualità del codice, riducendo i rischi associati a bug e vulnerabilità, e assicurando che il nostro software sia robusto, efficiente e sicuro.

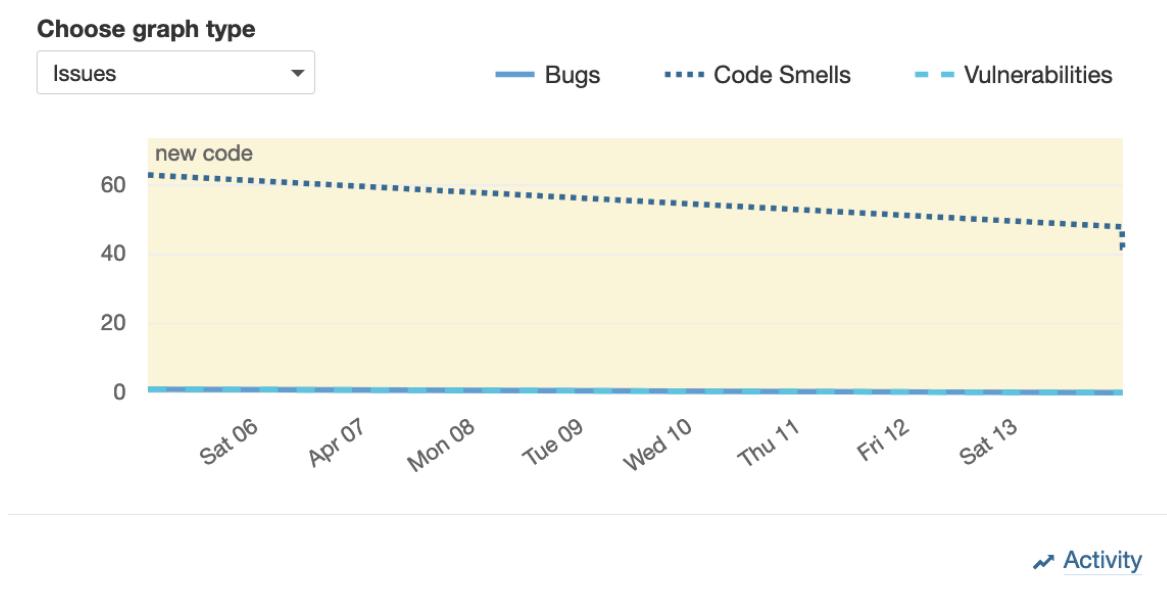
Di seguito sono riportati i dati dell'ultima modifica effettuata sul Server.

6.1.1 Indice di Duplicazione:



Come riportato dal grafico, non vi sono linee duplicate nel progetto. La mancanza di duplicazione indica che il codice è stato scritto in modo efficiente e non ci sono segmenti di codice ripetuti che potrebbero portare a problemi di manutenibilità o a una complessità non necessaria.

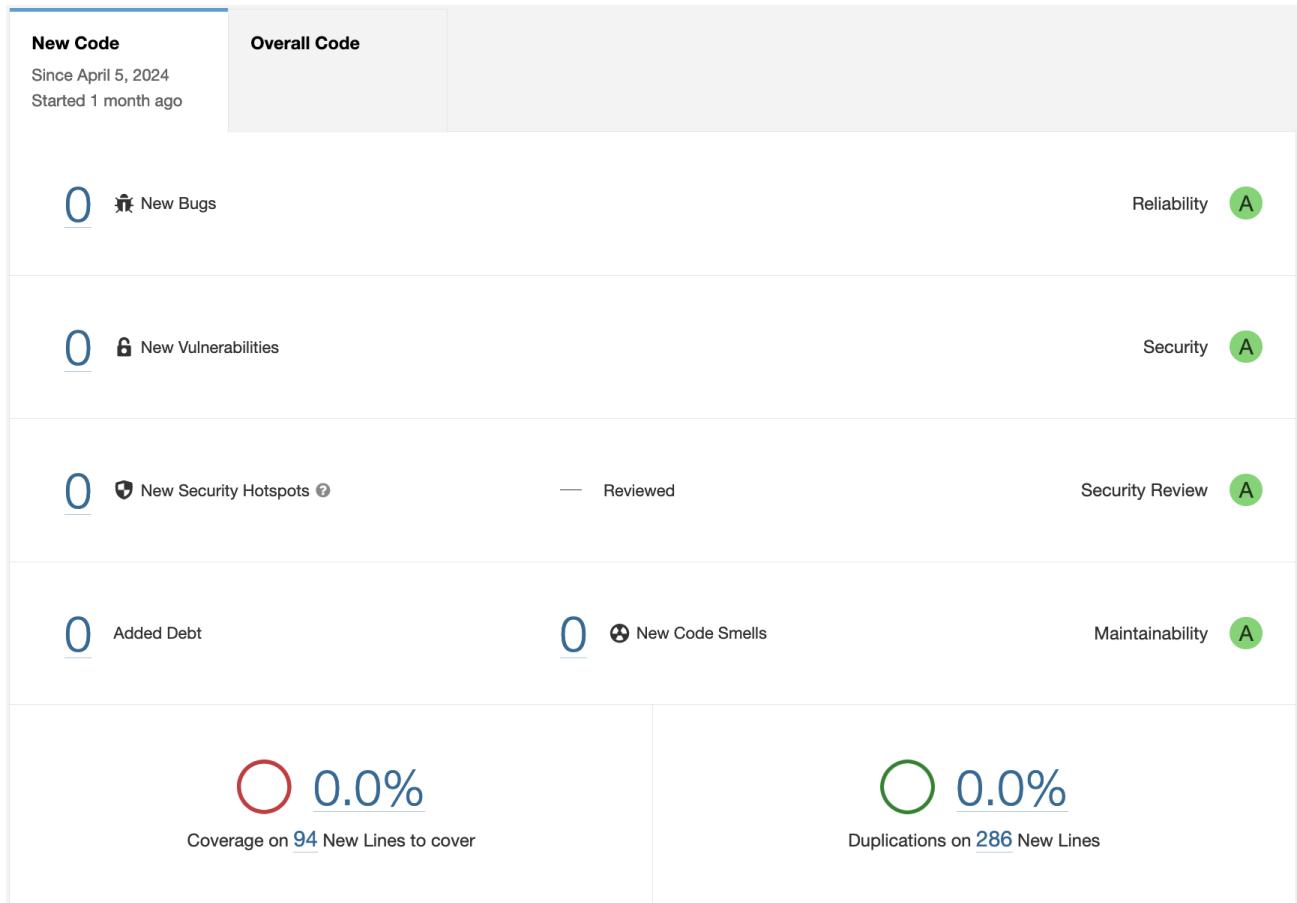
6.1.2 Indice degli Errori:



Come riportato dal grafico, inizialmente sono stati identificati circa 60 problemi nel codice. Tuttavia, nel corso del tempo, questi problemi sono stati risolti, portando il numero di errori riportati a zero. Ciò indica un

processo di miglioramento continuo della qualità del codice, con un'attenta risoluzione dei problemi rilevati durante l'analisi statica. La correzione tempestiva degli errori contribuisce a garantire che il codice sia robusto, affidabile e conforme agli standard di qualità definiti.

6.1.3 Report di qualità dell'ultima modifica



Da notare che segnala anche il *Added Debt* che rappresenta il tempo di correzione di eventuali *code smell* che vengono inseriti all'interno del codice.

6.1.4 SonarLint

SonarLint è uno strumento di analisi del codice integrato nell'IDE IntelliJ IDEA che offre un'analisi continua e in tempo reale del codice mentre viene scritto. Simile a SonarQube, SonarLint fornisce feedback immediato sulle potenziali problematiche del codice, consentendo agli sviluppatori di correggerle prima ancora di salvare o eseguire il codice.

Infatti, grazie all'uso di questo strumento, quando il codice è stato fatto analizzare da sonarQube, gli indici risultavano già ottimali.

STRUMENTO DI VERSIONING

In questo progetto, abbiamo adottato un approccio di versioning basato su **GitHub**, utilizzando due repository separati per gestire il front-end e il back-end dell'applicazione. Questa scelta è motivata da diversi fattori chiave:

- **Separazione delle Responsabilità:** La separazione dei repository consente a ciascun team (frontend e backend) di concentrarsi esclusivamente sul proprio codice e sulle proprie risorse. Ciò promuove una migliore gestione del codice, riduce i conflitti di merge e facilita il lavoro parallelo tra i due gruppi di sviluppatori.
- **Facilità di Gestione del Codice:** Con repository separati, è più semplice gestire le modifiche, le revisioni e le versioni del codice per ciascuna parte dell'applicazione. Ogni repository può essere gestito con le proprie regole di branching, pull request e cicli di rilascio, ottimizzando così il flusso di lavoro dello sviluppo.
- **Chiarezza e Tracciabilità:** La suddivisione del codice in repository dedicati aumenta la chiarezza e la tracciabilità delle modifiche. È più facile identificare le cause dei problemi, monitorare lo stato del codice e applicare correttamente le correzioni necessarie, migliorando così la qualità complessiva del software.

7.0.1 Repository Front-End:

terrecruis aggiunta statechart		
	a6ce0ee · last week	55 Commits
📁 DietiDeals24	aggiunto il quarto testing della funzionalita isOfferValid	last week
📁 analisi dei requisiti	aggiunta statechart	last week
📁 burocrazia	Sistemazione di tutti i viewModel corrispondenti alle viste	2 months ago
📁 specifica dei requisiti/UML-PROGETTO	aggiunta dei progressi piu recenti reperibili	5 months ago
📄 .DS_Store	aggiunta statechart	last week
📄 .gitignore	prova 2	4 months ago
📄 README.md	modifica del readme	5 months ago

7.0.2 Repository Back-End:

 terrecruis	aggiornamento di alcuni bug del database e in piu anche risolti alcun...	8749d0b · 2 months ago	22 Commits
 backend	aggiornamento di alcuni bug del database e in piu anche ...	2 months ago	
 database	aggiornamento di alcuni bug del database e in piu anche ...	2 months ago	
 .gitignore	aggiunta del Model, aggiunta del file properties per il Dat...	5 months ago	
 README.md	Update README.md	4 months ago	

In sintesi, l'approccio con due repository distinti su GitHub per il front-end e il back-end rappresenta una scelta strategica finalizzata a migliorare l'efficienza dello sviluppo, la gestione del codice e la qualità complessiva dell'applicazione.

TESTING

8.I UTILIZZO DI XCTEST

XCTest è il framework di testing di Apple, integrato nell'ambiente di sviluppo di Apple (Xcode), che permette di scrivere e eseguire oltre a unit test anche test di prestazioni e interfaccia utente, in questo progetto ci limiteremo a scrivere unit test per quattro funzionalità

8.I.I checkEmailAndPassword

Questo metodo si occupa di verificare che l'email e la password siano in un formato corretto prima di procedere alla registrazione nel sistema, i parametri utilizzati sono quindi:

- L'email che deve rispettare un'espressione regolare
- La password che deve contenere da 8 a 16 caratteri

il metodo viene testato con metodologia **Black Box**, di seguito le classi di equivalenza analizzate:

EMAIL:

- **CE 1** l'email è un'espressione regolare (valida)
- **CE 2** l'email non è un'espressione regolare (non valida)

PASSWORD:

- **CE 3** password = 8...16 (valida)
- **CE 4** password = MinInt...7 (non valida)
- **CE 5** password = 17...MaxInt (non valida)

Utilizziamo il criterio di copertura **N-SECT** (Normal Strong equivalence class test), che prevede la scrittura di $2^3 = 6$ test case, di seguito elencati:

- CT 1** checkEmailAndPassword(email: "prova@ciao.it", password: "12345678")
- CT 2** checkEmailAndPassword(email: "prova@ciao.it", password: "1234567")
- CT 3** checkEmailAndPassword(email: "prova@ciao.it", password: "123456789123456789")
- CT 4** checkEmailAndPassword(email: "prova@ciao", password: "12345678")
- CT 5** checkEmailAndPassword(email: "prova@ciao", password: "1234567")
- CT 6** checkEmailAndPassword(email: "prova@ciao", password: "123456789123456789")

Elenco dei metodi XCTest:

```
func testCT1_CheckEmailAndPassword_ValidEmailAndPassword() {
    // Arrange
    let viewModel = SignUpViewModel()
    viewModel.email = "prova@ciao.it"
    viewModel.password = "12345678"

    // Act
    let isValid = viewModel.checkEmailAndPassword()

    // Assert
    XCTAssertTrue(isValid, "CT1: Email e password valide devono tornare true")
}

func testCT2_CheckEmailAndPassword_TooShortPassword() {
    // Arrange
    let viewModel = SignUpViewModel()
    viewModel.email = "prova@ciao.it"
    viewModel.password = "1234567"

    // Act
    let isValid = viewModel.checkEmailAndPassword()

    // Assert
    XCTAssertFalse(isValid, "CT2: Email valida e password non valida deve tornare false")
}

func testCT3_CheckEmailAndPassword_TooLongPassword() {
    // Arrange
    let viewModel = SignUpViewModel()
    viewModel.email = "prova@ciao.it"
    viewModel.password = "123456789123456789"

    // Act
    let isValid = viewModel.checkEmailAndPassword()

    // Assert
    XCTAssertFalse(isValid, "CT3: Email valida e password troppo lunga deve tornare false")
}

func testCT4_CheckEmailAndPassword_InvalidEmail() {
    // Arrange
    let viewModel = SignUpViewModel()
    viewModel.email = "prova@ciao"
    viewModel.password = "12345678"

    // Act
    let isValid = viewModel.checkEmailAndPassword()

    // Assert
    XCTAssertFalse(isValid, "CT4: Email non valida e password valida deve tornare false")
}
```

```
func testCT5_CheckEmailAndPassword_InvalidEmailAndPassword() {
    // Arrange
    let viewModel = SignUpViewModel()
    viewModel.email = "prova@ciao"
    viewModel.password = "1234567"

    // Act
    let isValid = viewModel.checkEmailAndPassword()

    // Assert
    XCTAssertFalse(isValid, "CT5: Email e password non valide deve tornare false")
}

func testCT6_CheckEmailAndPassword_InvalidEmail_TooLongPassword() {
    // Arrange
    let viewModel = SignUpViewModel()
    viewModel.email = "prova@ciao"
    viewModel.password = "123456789123456789"

    // Act
    let isValid = viewModel.checkEmailAndPassword()

    // Assert
    XCTAssertFalse(isValid, "CT6: Email non valida e password troppo lunga deve tornare false")
}
```

8.1.2 SignUp

Questo metodo verifica la validità dell'input prima della registrazione nel sistema, i parametri in input sono quattro stringhe:

- il campo username che va riempito con la mail che deve essere conforme a un'espressione regolare
- la password da 8 a 16 caratteri
- fullName da almeno 3 caratteri
- phoneNumber sono precisamente 10 cifre

Il metodo viene testato con metodologia **Black-Box**, di seguito le classi di equivalenza analizzate:

EMAIL:

- **CE 1** email è un'espressione regolare (valida)
- **CE 2** email non è un'espressione regolare (non valido)

PASSWORD:

- **CE 3** password = 8...16 (valido)
- **CE 4** password = 17...MaxInt (non valido)
- **CE 5** password = MinInt...7 (non valido)

FULL-NAME:

- **CE 6** fullname = 3...MaxInt (valido)
- **CE 7** fullname = 0...3 (non valida)

PHONE-NUMBER:

- **CE 8** phoneNumber = 10 (valido)
- **CE 9** phoneNumber = MinInt...9 (non valida)
- **CE 10** phoneNumber = 11...MaxInt (non valida)

utilizziamo il criterio di copertura **R-WECT** (Robust Weak Equivalence Class Testing) con la scrittura dei seguenti test case:

-**CT 1** signUp(username: goodMail@prova.it, password: 12345678, fullName: goodName secName, phoneNumber: 3456543567)

-**CT 2** signUp(username: badMail@prova, password: 12345678, fullName: goodName secName, phoneNumber: 3456543567)

-**CT 3** signUp(username: goodMail@prova.it, password: passwordIsTooLong, fullName: goodName secName, phoneNumber: 3456543567)

-**CT 4** signUp(username: goodMail@prova.it, password: shortp, fullName: goodName secName, phoneNumber: 3456543567)

- CT 5** signUp(username: goodMail@prova.it, password: 12345678, fullName: "", phoneNumber: 3456543567)
- CT 6** signUp(username: goodMail@prova.it, password: 12345678, fullName: goodName secName, phoneNumber: 75776)
- CT 7** signUp(username: goodMail@prova.it, password: 12345678, fullName: goodName secName, phoneNumber: 34565435635227)

elenco dei metodi XCTest:

```

func testCT1_SignUp_ValidParameter() {
    //Arrange
    let sessionManager = SessionManager()
    let username = "goodPassword"
    let password = "12345678"
    let fullName = "GoodName"
    let phoneNumber = "1234567899"

    //Act + Assert
    XCTAssertNoThrow(Task {try await sessionManager.signUp(username:username,password:password,fullName:fullName,phoneNumber:phoneNumber)})
}

func testCT2_SignUp_InvalidEmail() {
    //Arrange
    let sessionManager = SessionManager()
    let username = "goodPassword"
    let password = "12345678"
    let fullName = "GoodName"
    let phoneNumber = "1234567899"

    //Act
    let task = Task {
        do {
            try await sessionManager.signUp(username:username,password:password,fullName:fullName,phoneNumber:phoneNumber)
            XCTFail("Expected AuthError error to be thrown.")
        } catch {
            XCTAssertTrue(error is AuthError)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

func testCT3_SignUp_PasswordTooLong() {
    //Arrange
    let sessionManager = SessionManager()
    let username = "goodMail@prova.it"
    let password = "passwordIsTooLong"
    let fullName = "goodName secName"
    let phoneNumber = "3456543567"

    //Act
    let task = Task {
        do {
            try await sessionManager.signUp(username:username,password:password,fullName:fullName,phoneNumber:phoneNumber)
            XCTFail("Expected AuthError error to be thrown.")
        } catch {
            XCTAssertTrue(error is AuthError)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

```

```

func testCT4_SignUp_passwordTooShort() {
    //Arrange
    let sessionManager = SessionManager()
    let username = "goodMail@prova.it"
    let password = "shortp"
    let fullName = "goodName secName"
    let phoneNumber = "3456543567"

    //Act
    let task = Task {
        do {
            try await sessionManager.signUp(username:username,password:password,fullName:fullName,phoneNumber:phoneNumber)
            XCTFail("Expected AuthError error to be thrown.")
        } catch {
            XCTAssertTrue(error is AuthError)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

func testCT5_SignUp_EmptyFullname() {
    //Arrange
    let sessionManager = SessionManager()
    let username = "goodPassword"
    let password = "12345678"
    let fullName = ""
    let phoneNumber = "1234567899"

    //Act
    let task = Task {
        do {
            try await sessionManager.signUp(username:username,password:password,fullName:fullName,phoneNumber:phoneNumber)
            XCTFail("Expected AuthError error to be thrown.")
        } catch {
            XCTAssertTrue(error is AuthError)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

func testCT6_SignUp_TooShortPhoneNumber() {
    //Arrange
    let sessionManager = SessionManager()
    let username = "goodPassword"
    let password = "12345678"
    let fullName = "goodName secName"
    let phoneNumber = "123"

    //Act
    let task = Task {
        do {
            try await sessionManager.signUp(username:username,password:password,fullName:fullName,phoneNumber:phoneNumber)
            XCTFail("Expected AuthError error to be thrown.")
        } catch {
            XCTAssertTrue(error is AuthError)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

```

```
func testCT7_SignUp_TooLongLongPhoneNumber() {
    //Arrange
    let sessionManager = SessionManager()
    let username = "goodPassword"
    let password = "12345678"
    let fullName = "goodName secName"
    let phoneNumber = "12345678999999"

    //Act
    let task = Task {
        do {
            try await sessionManager.signUp(username:username,password:password,fullName:fullName,phoneNumber:phoneNumber)
            XCTFail("Expected AuthError error to be thrown.")
        } catch {
            XCTAssertTrue(error is AuthError)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}
```

8.1.3 isOfferValid

Questo metodo verifica la validità dell'input prima della puntata su una qualsiasi asta nel sistema, i parametri in input sono:

- currentOffer di tipo Float che non può essere maggiore di buyerOffer
- buyerOffer di tipo Decimal che non può essere minore di currentOffer

Il metodo viene testato con metodologia **Black-Box**, di seguito le classi di equivalenza analizzate:

BUYER-OFFER:

- **CE 1** buyerOffer =]currentOffer...MaxDecimal] (valida)
- **CE 2** buyerOffer = [MinDecimal...currentOffer] (non valido)

CURRENT-OFFER

- **CE 3** currentOffer = [0...buyerOffer[(valido)
- **CE 4** currentOffer = [buyerOffer...MaxFloat] (non valido)

utilizziamo il criterio di copertura **N-WECT** (Normal Weak Equivalence Class Testing) aggiungendo anche un test per un caso limite, di seguito elencati i test case:

- CT 1** isOfferValid(buyerOffer: 89.45, currentOffer: 64.6)
- CT 2** isOfferValid(buyerOffer: 50.22, currentOffer: 100.3)
- CT 3** isOfferValid(buyerOffer: 50.3, currentOffer: 50.3)

elenco dei metodi XCTest:

```
func testCT1_isOfferValid_ValidInput() {
    //Arrange
    let auctionViewModel = AuctionViewModel(user: "", auction: AuctionData())
    let buyerOffer: Decimal = 89.45
    let currentOffer: Float = 64.6

    //Act
    let result = auctionViewModel.isOfferValid(buyerOffer: buyerOffer, currentOffer: currentOffer)

    //Assert
    XCTAssertEqual(result)
}

func testCT2_isOfferValid_InvalidInput() {
    //Arrange
    let auctionViewModel = AuctionViewModel(user: "", auction: AuctionData())
    let buyerOffer: Decimal = 50.22
    let currentOffer: Float = 100.3

    //Act
    let result = auctionViewModel.isOfferValid(buyerOffer: buyerOffer, currentOffer: currentOffer)

    //Assert
    XCTAssertFalse(result)
}

func testCT4_isOfferValid_LimitCase() {
    //Arrange
    let auctionViewModel = AuctionViewModel(user: "", auction: AuctionData())
    let buyerOffer: Decimal = 50
    let currentOffer: Float = 50

    //Act
    let result = auctionViewModel.isOfferValid(buyerOffer: buyerOffer, currentOffer: currentOffer)

    //Assert
    XCTAssertFalse(result)
}
```

8.1.4 changePassword

Questo metodo verifica la validità dell'input all'inserimento di una nuova password, i parametri in input sono due stringhe:

- la vecchia password da 8 a 16 caratteri
- la nuova password da 8 a 16 caratteri

Deve essere rispettata un ulteriore condizione, cioè che le due stringhe non possono essere uguali. Il metodo viene testato con metodologia **Black-Box**, di seguito le classi di equivalenza analizzate:

VECCIA PASSWORD:

- **CE 1** oldPassword = 8...16 (valida)
- **CE 2** oldPassword = 17...MaxInt (non valido)
- **CE 3** oldPassword = MinInt...7 (non valido)

NUOVA PASSWORD:

- **CE 4** newPassword = 8...16 (valido)
- **CE 5** newPassword = 17...MaxInt (non valido)
- **CE 6** newPassword = MinInt...7 (non valido)

DISUGUAGLIANZA:

- **CE 7** oldPassword != newPassword (valido)
- **CE 8** oldPassword = newPassword (non valida)

utilizziamo il criterio di copertura **R-WECT** (Robust Weak Equivalence Class Testing) con la scrittura dei seguenti test case:

- CT 1** changePassword(oldPassword: password1, newPassword: password2)
- CT 2** changePassword(oldPassword: password1, newPassword: password1)
- CT 3** changePassword(oldPassword: passwordIsTooLong293, newPassword: goodPassword)
- CT 4** changePassword(oldPassword: shortp , newPassword: goodPassword)
- CT 5** changePassword(oldPassword: goodPassword , newPassword: passwordIsTooLong293)
- CT 6** changePassword(oldPassword: goodPassword , newPassword: shortp)

elenco dei metodi XCTest:

```

func testCT1_ChangePassword_DifferentPasswords() {
    //Arrange
    let sessionManager = SessionManager()
    let oldPassword = "password1"
    let newPassword = "password2"

    //Act + Assert
    XCTAssertNoThrow(Task {try await sessionManager.changePassword(oldPassword: oldPassword, newPassword: newPassword)})
}

func testCT2_ChangePassword_SamePasswords() {
    //Arrange
    let sessionManager = SessionManager()
    let oldPassword = "password1"
    let newPassword = "password1"

    //Act
    let task = Task {
        do {
            try await sessionManager.changePassword(oldPassword: oldPassword, newPassword: newPassword)
            XCTFail("Expected ValidationError.passwordsNotValid error to be thrown.")
        } catch {
            XCTAssertTrue(error is ValidationError)
            XCTAssertEqual(error as? ValidationError, ValidationError.passwordsNotValid)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

func testCT3_ChangePassword_OldPasswordTooLong() {
    //Arrange
    let sessionManager = SessionManager()
    let oldPassword = "passwordIsTooLong293"
    let newPassword = "goodPassword"

    //Act
    let task = Task {
        do {
            try await sessionManager.changePassword(oldPassword: oldPassword, newPassword: newPassword)
            XCTFail("Expected ValidationError.passwordsNotValid error to be thrown.")
        } catch {
            XCTAssertTrue(error is ValidationError)
            XCTAssertEqual(error as? ValidationError, ValidationError.passwordsNotValid)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

```

```

func testCT4_ChangePassword_OldPasswordTooShort() {
    //Arrange
    let sessionManager = SessionManager()
    let oldPassword = "shortp"
    let newPassword = "goodPassword"

    //Act
    let task = Task {
        do {
            try await sessionManager.changePassword(oldPassword: oldPassword, newPassword: newPassword)
            XCTFail("Expected ValidationError.passwordsNotValid error to be thrown.")
        } catch {
            XCTAssertTrue(error is ValidationError)
            XCTAssertEqual(error as? ValidationError, ValidationError.passwordsNotValid)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

func testCT5_ChangePassword_NewPasswordTooLong() {
    //Arrange
    let sessionManager = SessionManager()
    let oldPassword = "goodPassword"
    let newPassword = "passwordIsTooLong293"

    //Act
    let task = Task {
        do {
            try await sessionManager.changePassword(oldPassword: oldPassword, newPassword: newPassword)
            XCTFail("Expected ValidationError.passwordsNotValid error to be thrown.")
        } catch {
            XCTAssertTrue(error is ValidationError)
            XCTAssertEqual(error as? ValidationError, ValidationError.passwordsNotValid)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

func testCT6_ChangePassword_NewPasswordTooShort() {
    //Arrange
    let sessionManager = SessionManager()
    let oldPassword = "goodPassword"
    let newPassword = "shortp"

    //Act
    let task = Task {
        do {
            try await sessionManager.changePassword(oldPassword: oldPassword, newPassword: newPassword)
            XCTFail("Expected ValidationError.passwordsNotValid error to be thrown.")
        } catch {
            XCTAssertTrue(error is ValidationError)
            XCTAssertEqual(error as? ValidationError, ValidationError.passwordsNotValid)
        }
    }

    // Assert
    Task { @MainActor in
        await task.value
    }
}

```

STUDIO DELL'USABILITÀ SUL CAMPO

Philip Virzi, ricercatore e praticante nel campo dell'Usabilità e dell'Interazione Uomo-Computer, sostiene che con un gruppo di 5 utenti nei test di usabilità si riesce a identificare circa l'85% delle problematiche che impattano sull'usabilità del prodotto finito. Per questo motivo, è necessario svolgere test di usabilità sul campo anche dopo aver effettuato valutazioni euristiche. La valutazione euristica, pur essendo utile, non sostituisce i test con gli utenti. Le due tecniche sono complementari e, se utilizzate insieme, offrono una valutazione più completa dell'usabilità del sistema.

La scelta dei soggetti che hanno provato il software non è stata immediata, poiché il target di possibili utilizzatori copre una vasta fascia di soggetti. Tuttavia, bisogna avere l'accortezza di selezionare utenti con caratteristiche diverse, in modo da ottenere dati vari e rappresentativi su cui lavorare:

- Federica, 20 anni, lavora in un negozio di abbigliamento e ha esperienza con la compravendita e le spedizioni online.
- Valentina, 21 anni, è una studentessa che cerca di risparmiare su vari prodotti, con poca esperienza tecnologica.
- Noemi, 22 anni, acquista frequentemente online e ha familiarità con software di e-commerce.
- Nicola, 22 anni, viaggia spesso per lavoro e cerca buone offerte per la compravendita di oggetti e servizi, ma non è molto esperto di tecnologia.
- Giampiero, 27 anni, ha l'hobby del fai-da-te e a volte cerca offerte online.

Di seguito riporteremo il risultato dei test sommativi, svolti a valle della progettazione del software, in particolare abbiamo effettuato un test di compito e due di scenario.

9.1 TEST DI COMPITO

I test di compito sono progettati per valutare l'usabilità di funzionalità specifiche o compiti definiti all'interno del sistema, gli utenti partecipanti ai test sono incaricati di completare compiti specifici, seguendo istruzioni dettagliate fornite dagli osservatori, le istruzioni devono essere chiare e mirate. Il test di compito che abbiamo scelto è "puntare su un asta", fornendo loro tutti i passaggi per farlo, di seguito i risultati:

	Federica	Valentina	Noemi	Nicola	Giampiero
Tempo di Completamento	1 min	1.5 min	0.5 min	1.5 min	1 min
Errori commessi	0	0	0	1	0

In base al test emerge un tempo medio di 1.1 minuti per il completamento del task assegnato e meno di un errore medio per utente, tuttavia abbiamo notato che alcuni utenti hanno mostrato incomprensione nel funzionamento delle aste, per cui abbiamo deciso di introdurre una sezione aggiuntiva nel menù "impostazioni" che spiega precisamente le differenze tra i due tipi di aste presenti, questo semplice metodo permette la divulgazione di informazioni senza uno sforzo eccessivo, riducendo l'indice di errore medio.

9.2 TEST DI SCENARIO

I test di scenario si differenziano dai test di compito perché non forniscono istruzioni dettagliate su come procedere passo dopo passo. Gli utenti vengono posti di fronte a situazioni d'uso che riflettono scenari reali o tipici per il sistema e sono liberi di completare il compito come ritengono opportuno.

1°TEST

Nel primo test viene chiesto agli utenti di cambiare la password del loro profilo:

	Federica	Valentina	Noemi	Nicola	Giampiero
Tempo di Completamento	1 min	1.5 min	1.5 min	2 min	1 min
Errori commessi	0	1	0	1	0

In base al test emerge un tempo medio di 1.4 minuti per il completamento del task assegnato e meno di un errore per utente in media, inoltre è emerso che 2 su 5 utenti cercavano il tasto di cambio password nel menu "impostazioni", quindi abbiamo deciso di aggiungerlo anche lì, avendo un utile ridondanza nel sistema che migliora sicuramente il tempo medio di svolgimento della funzionalità.

2° TEST

Nel secondo test viene chiesto agli utenti di personalizzare il profilo:

	Federica	Valentina	Noemi	Nicola	Giampiero
<i>Tempo di Completamento</i>	2 min	2.5 min	1.5 min	2.5 min	1.5 min
<i>Errori commessi</i>	0	0	0	0	0

In base al test emerge un tempo medio di 2 minuti per il completamento del task assegnato e non ci sono stati errori, l'idea di mettere un bottone in alto a destra per modificare il profilo è stata buona, in quanto gli utenti non hanno avuto difficoltà a trovarlo e a completare il processo di modifica.

9.3 QUESTIONARIO DI VALUTAZIONE

riportiamo infine il questionario presentato agli utenti che hanno utilizzato l'app, con una valutazione del gradimento pari al **90%**

Valutazione del software DietiDeals24

1. Quanto è stata di tuo gradimento l'app?

- Non lo è stato**
- non molto**
- abbastanza**
- molto**

2. Quanto hai gradito il design?

- Per niente**
- non molto**
- abbastanza**
- molto**

3. Quanto è stato facile utilizzare l'app?

- Non è stato facile**
- solo qualche difficoltà iniziale**
- abbastanza facile**
- molto facile**

4. Hai riscontrato bug nel sistema durante l'utilizzo?

- Molti**
- Pochi**
- Solo uno o due**
- Nessuno**

5. Riutilizzeresti ancora l'app?

- No**
- Non penso**
- Penso di sì**
- Si**

Grazie per il tuo feedback!

MONITORAGGIO DEL SOFTWARE

Per il monitoraggio del software sviluppato abbiamo utilizzato **Firebase Analytics**, una soluzione di analisi gratuita offerta da Google. Questa piattaforma consente di raccogliere e analizzare dati relativi all'utilizzo delle applicazioni mobili, permettendo agli sviluppatori di comprendere meglio il comportamento degli utenti e migliorare l'esperienza utente.

10.1 CARATTERISTICHE PRINCIPALI DI FIREBASE ANALYTICS

- **Tracciamento degli eventi:** Monitoraggio di eventi personalizzati come interazioni con l'app.
- **Report in tempo reale:** Report dettagliati per vedere l'interazione degli utenti in tempo reale, utile per identificare e risolvere rapidamente eventuali problemi.
- **Segmentazione del pubblico:** Creazione di segmenti di pubblico basati su criteri come demografia e comportamento degli utenti.

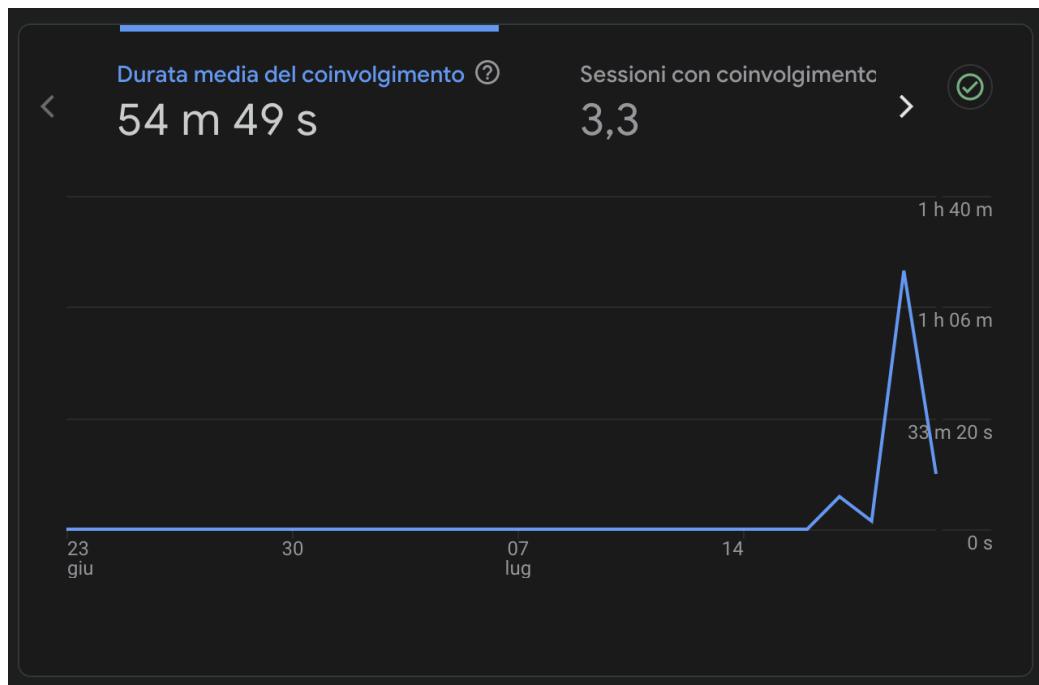
10.2 IMPLEMENTAZIONE E UTILIZZO

L'implementazione di Firebase Analytics è stata semplice e rapida. Dopo aver integrato il SDK, abbiamo iniziato a tracciare vari eventi significativi per analizzare il comportamento degli utenti, come login, modifica delle impostazioni del profilo, partecipazione alle aste, acquisti e visualizzazione delle pagine dei prodotti.

10.3 ANALISI DEI DATI

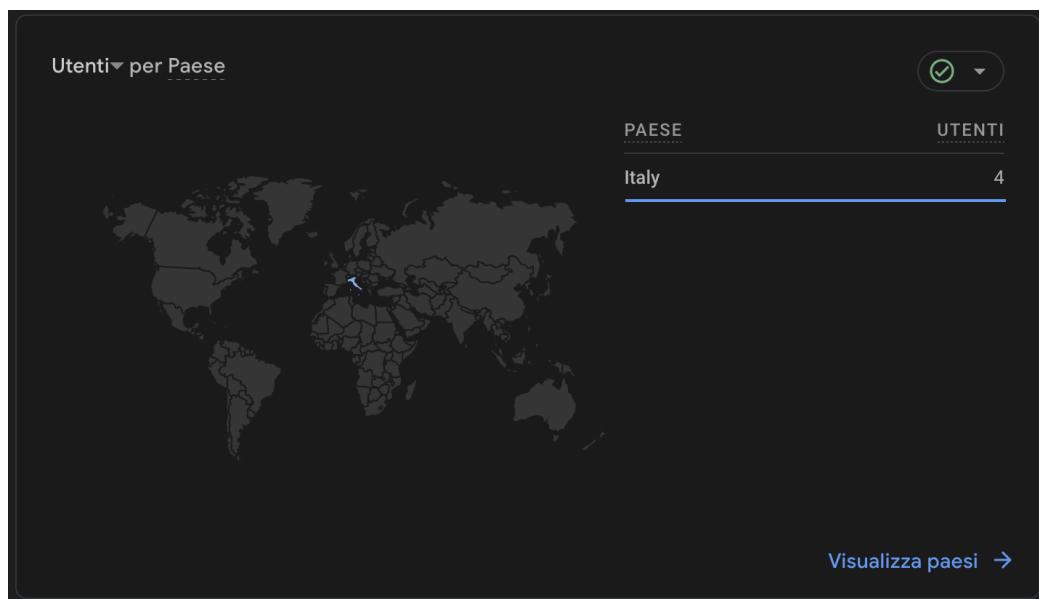
L'analisi dei dati raccolti ha fornito preziose informazioni sul comportamento degli utenti, aiutandoci a identificare problemi di usabilità e migliorare le funzionalità dell'applicazione. Di seguito sono riportati i grafici relativi ai dati ottenuti.

10.3.1 Grafico Coinvolgimento Utenti:



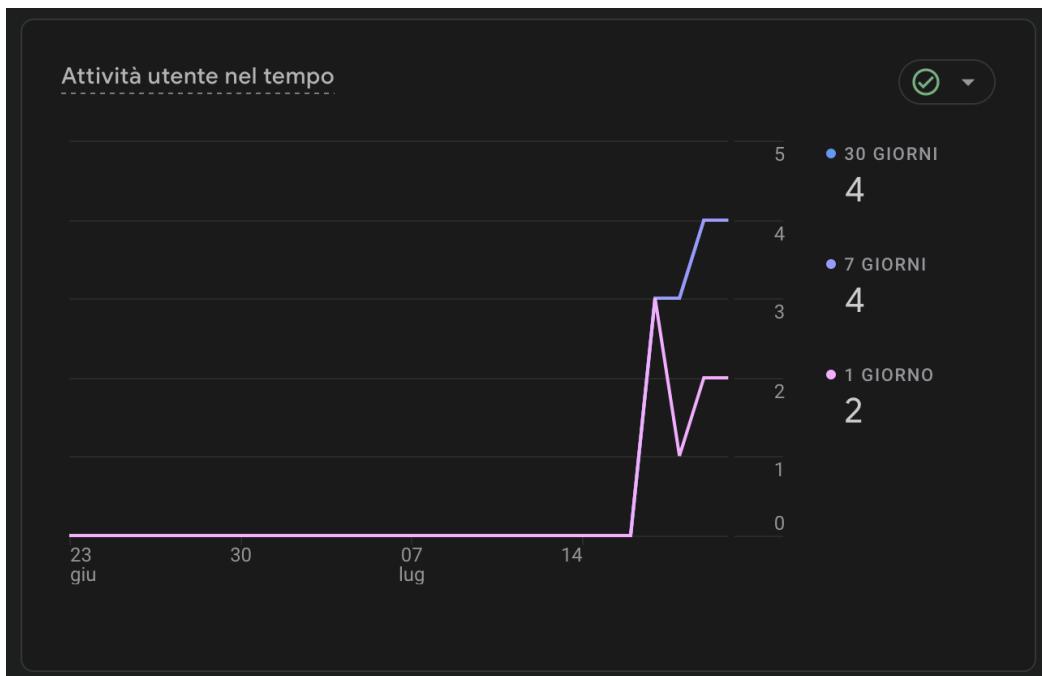
Questo grafico mostra il livello di coinvolgimento degli utenti, indicando la frequenza e la durata delle loro sessioni nell'app. Il picco di utilizzo durante le ore serali suggerisce quando gli utenti sono più attivi, aiutandoci a pianificare manutenzioni e aggiornamenti.

10.3.2 Grafico Nazionalità Utenti:



Questo grafico rappresenta la distribuzione geografica degli utenti dell'applicazione.

10.3.3 Grafico Attività Utente nel tempo:

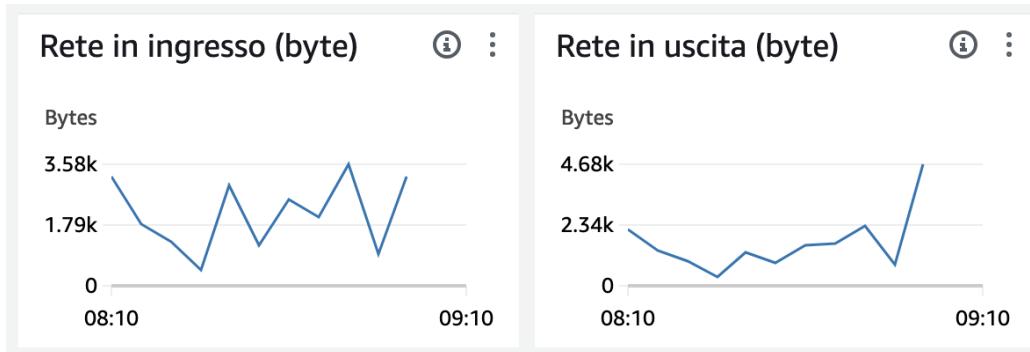


Questo grafico illustra l'attività degli utenti nel tempo, evidenziando trend e pattern di utilizzo. L'analisi di questi dati ci permette di identificare periodi di maggiore e minore attività, ottimizzando così le risorse del server e migliorando le prestazioni dell'app durante i picchi di utilizzo.

10.4 MONITORAGGIO DEL SERVER

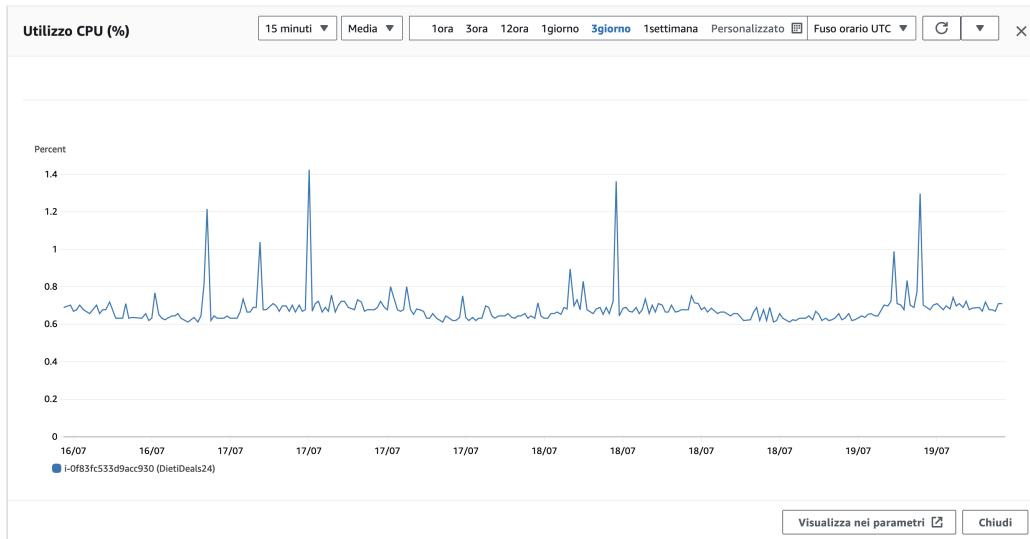
In questa sezione vengono analizzati i dati relativi alle prestazioni del server, ospitato su un'istanza EC2 di Amazon Web Services (AWS):

10.4.1 Rete in Ingresso e Rete in Uscita (byte):



Questo grafico mostra il traffico di rete in entrata e in uscita, evidenziando i picchi di utilizzo della banda.

10.4.2 Utilizzo Medio Della CPU:

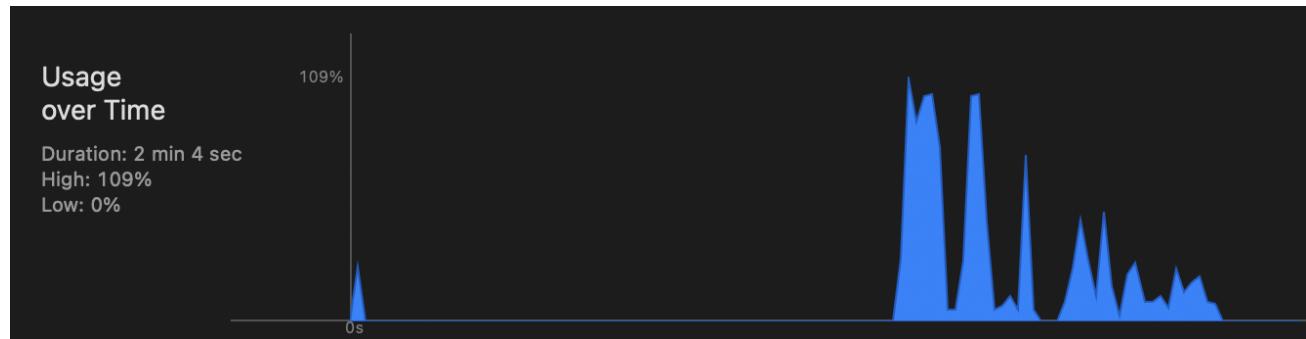


Questo grafico illustra l'utilizzo medio della CPU, aiutandoci a monitorare le prestazioni e identificare eventuali sovraccarichi. Se necessario, la potenza dell'istanza EC2 può essere aumentata per migliorare le prestazioni e gestire carichi di lavoro più elevati.

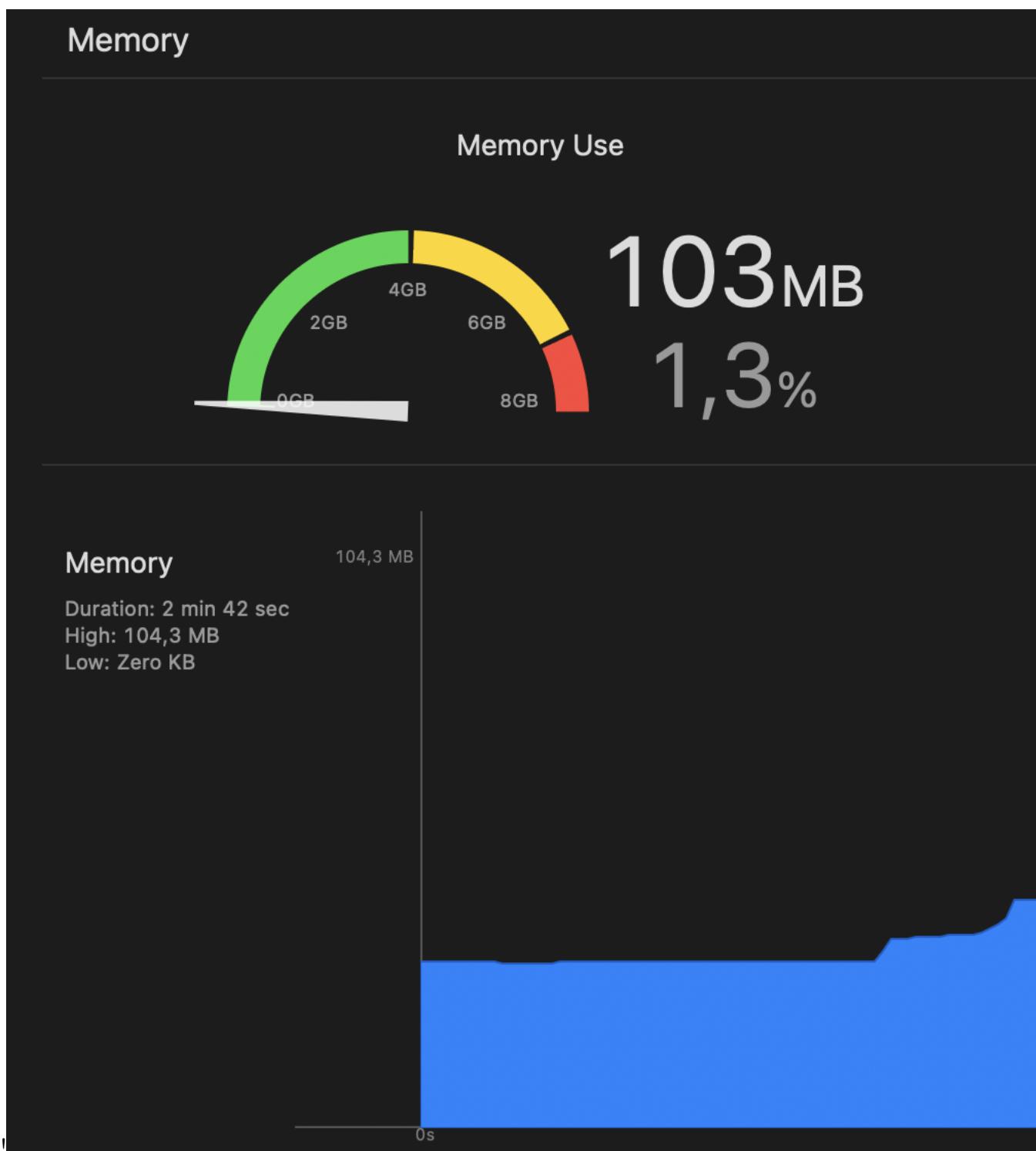
10.5 MONITORAGGIO DELL'APPLICAZIONE

In questa sezione vengono analizzati i dati relativi all'utilizzo dell'applicazione su dispositivi iOS. Monitorare le prestazioni dell'applicazione è fondamentale per garantire un'esperienza utente fluida e priva di problemi. I dati raccolti ci aiutano a identificare eventuali colli di bottiglia e a ottimizzare l'efficienza dell'app.

10.5.1 Grafico utilizzo CPU:



Questo grafico mostra l'utilizzo della CPU da parte dell'applicazione sui dispositivi iOS, aiutandoci a capire quanto l'app sfrutti le risorse del processore e identificare eventuali inefficienze.

10.5.2 Grafico dell'utilizzo di RAM:

Questo grafico illustra l'utilizzo della memoria RAM da parte dell'applicazione sui dispositivi iOS.

Monitorare l'utilizzo della RAM è essenziale per garantire che l'app funzioni senza rallentamenti o crash, specialmente su dispositivi con risorse limitate.