



DietiDeals24

Università degli Studi di Napoli Federico II

Scuola Politecnica e delle Scienze di Base
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione
Corso di Laurea in Informatica

Anno Accademico 2023-2024

Scisciola Simone
N86004025

Lucchese Andrea
N86004112

Carandente Ilaria Giuseppina
N86004269

29 gennaio 2025

Indice

Traccia	3
DietiDeals24	3
1 Glossario	4
1.1 Obiettivo	4
1.2 Glossario	4
2 Requirement Elicitation	5
2.1 Obiettivo	5
2.2 Introduzione al sistema	5
2.2.1 Requisiti funzionali	5
2.2.2 Requisiti non funzionali	9
3 Requirement Analysis	10
3.1 Obiettivo	10
3.2 Use Case Diagram	10
3.3 Utenti target	11
3.4 Mock-up	13
3.4.1 Logo	13
3.4.2 Colori	13
3.5 Tabelle di Cockburn	15
3.5.1 I caso d'uso: Crea un'asta inversa	15
3.5.2 II caso d'uso: Accetta un'offerta ricevuta all'asta silenziosa	19
3.5.3 III caso d'uso: Visualizza i dettagli di un'asta	22
3.5.4 IV caso d'uso: Modificare il proprio profilo	24
3.6 Valutazione dell'usabilità a priori	28
3.6.1 Valutazioni con utenti	28
3.6.2 Modifiche a seguito delle valutazioni	30
4 Requirement Specification	32
4.1 Obiettivo	32
4.2 Class Diagram	32
4.2.1 Class Diagram del dominio del problema	32
4.2.2 Dizionario delle classi	33
4.2.3 Dizionario delle associazioni	34
4.3 Class Diagram per casi d'uso	36
4.4 Sequence Diagram	59
4.4.1 I caso d'uso: Crea un'asta inversa	59
4.4.2 II caso d'uso: Accetta un'offerta ricevuta all'asta silenziosa	60
4.5 Statechart Diagram	61
4.5.1 I caso d'uso: Crea un'asta inversa	61
4.5.2 II caso d'uso: Accetta un'offerta ricevuta all'asta silenziosa	62
4.5.3 III caso d'uso: Visualizza i dettagli di un'asta	63
4.5.4 IV caso d'uso: Modificare il proprio profilo	64

5 System Design	65
5.1 Architettura e criteri di progettazione	65
5.2 Tecnologie adottate	67
5.2.1 Android	67
5.2.2 AWS EC2 & AWS Cognito	67
5.2.3 Kotlin	69
5.2.4 Java	70
5.2.5 Docker (Compose)	70
5.2.6 PostgreSQL	71
5.2.7 REST API	72
5.2.8 Spring Framework & Spring Boot	72
5.2.9 SonarQube	74
6 Object Design	75
6.1 Design Pattern	75
6.1.1 Client	75
6.1.2 Resource Server	76
6.2 Class Diagram	78
6.2.1 Client	78
6.2.2 Server	78
6.3 Sequence Diagram	79
6.3.1 I caso d'uso: Crea un'asta inversa	79
6.3.2 II caso d'uso: Accetta un'offerta ricevuta all'asta silenziosa	81
6.3.3 III caso d'uso: Visualizza i dettagli di un'asta	83
6.3.4 IV caso d'uso: Modificare il proprio profilo	85
6.4 Endpoint	87
6.5 Database	91
6.5.1 Ristrutturazione della progettazione concettuale	91
6.5.2 Schema logico	91
6.5.3 Schema fisico	92
7 Verifica del software	99
7.1 Verifica statica	99
7.2 Verifica dinamica	101
7.2.1 I test: offertaService.checkFieldsValid (server)	101
7.2.2 II test: accountService.isEmailAndPasswordValid (server)	105
7.2.3 III test: ModelProfilo validate (client)	109
7.2.4 IV test: JWTTests getUserEmail (client)	111
8 Validazione del software	113
8.1 Alpha test	113
8.2 Beta test	114
8.3 Firebase Analytics	115
Bibliografia	117

Traccia

DietiDeals24

DietiDeals24 è una piattaforma per la gestione di aste online. Il sistema consiste in un'applicazione mobile, desktop o web-based, performante e affidabile, attraverso cui gli utenti possono fruire delle funzionalità del sistema in modo intuitivo, rapido e piacevole.

Le principali funzionalità offerte da DietiDeals24 sono indicate di seguito:

- 1 Un utente può registrare un nuovo account ed utilizzarlo per accedere al sistema. Un account può essere di due tipi: venditore o acquirente. La stessa e-mail può essere utilizzata per al più un account venditore e un account compratore. È apprezzata la possibilità di effettuare la registrazione utilizzando credenziali di terze parti (e.g.: Google, Facebook, GitHub, etc...). Gli utenti possono personalizzare il proprio profilo con una short bio, link al proprio sito web / ai propri social, area geografica, etc.
- 2 Il sistema permette ai venditori/compratori di creare aste di diverso tipo per la vendita/acquisto di beni/servizi e presentare offerte per le aste correntemente attive. Ciascuna asta è caratterizzata da una descrizione del bene/servizio in vendita e, optionalmente, da una fotografia dello stesso. Ciascuna asta inoltre è caratterizzata da una categoria (e.g.: elettronica, informatica, giocattoli, alimentari, servizi, etc...), introdotta per facilitare la navigazione tra le tante aste presenti nel sistema.
- 3 Venditori e compratori possono effettuare ricerche tra le aste correntemente attive, filtrando per categoria e/o per parole chiave, e visualizzare i dettagli di ciascuna asta. Inoltre, è possibile visualizzare anche il profilo utente del venditore che ha creato l'asta.
- 4 Un venditore può creare una nuova **Asta a tempo fisso**. Un'asta a tempo fisso è caratterizzata da una data di scadenza, scelta dal venditore. Inoltre, il venditore può specificare una soglia minima (segreta) di prezzo al quale vendere il prodotto. Gli acquirenti possono visualizzare i dettagli dell'asta, inclusa l'offerta più alta ricevuta finora, ma non la soglia minima di prezzo fissata dal venditore. Gli acquirenti possono, fino alla data di scadenza dell'asta, presentare delle offerte migliorative rispetto alla migliore offerta corrente, specificando l'importo desiderato (in €). Al momento della scadenza, il compratore con l'offerta più alta si aggiudica il bene/servizio. Se non si raggiunge la soglia minima segreta impostata dal venditore, l'asta viene considerata fallita. In entrambi i casi, il venditore e tutti gli acquirenti che hanno partecipato all'asta visualizzano una notifica.
- 7 Un venditore può creare una nuova **Asta silenziosa**. In questo tipo di asta, il venditore specifica una data di scadenza e i compratori possono inviare offerte segrete al venditore. Il venditore può scegliere se accettare o rifiutare le offerte ricevute. Una sola offerta può essere accettata per ogni asta.
- 8 Un compratore può creare una nuova **Asta inversa**. In questo tipo di asta, il compratore specifica il prodotto/servizio richiesto, eventualmente inserendo un'immagine dello stesso, un prezzo iniziale che è disposto a pagare, e una data di scadenza. I venditori in grado di fornire quel particolare prodotto/servizio possono quindi partecipare all'asta competendo abbassando il prezzo. In particolare, fino al momento della scadenza dell'asta, i venditori possono presentare offerte al ribasso. Al momento della scadenza dell'asta, il venditore con l'offerta più bassa si aggiudica la fornitura del prodotto/servizio.

CAPITOLO 1

Glossario

1.1 Obiettivo

In questa sezione sono definiti i vocaboli meno comuni utilizzati all'interno della documentazione.

1.2 Glossario

Vocabolo	Definizione
Stakeholders	Insieme dei soggetti che hanno un interesse nei confronti di un'organizzazione e che con il loro comportamento possono influenzarne l'attività. Nel caso dell'ingegneria del software, tutti gli individui interessati alla messa in opera del sistema.
Requirement Elicitation	Raccolta dei requisiti.
Requirement Analysis	Analisi dei requisiti.
Requirement Specification	Specificazione dei requisiti.
System Design	Progettazione del sistema.
Object Design	Progettazione orientata agli oggetti.
Utente loggato	Utente che ha effettuato il login, ovvero l'accesso al proprio account.
Utente target	Utente "tipico" che utilizzerà l'applicativo, individuato a seguito di uno studio demografico.
Use Case Diagram	Diagramma dei casi d'uso.
Class Diagram	Diagramma delle classi.
Sequence Diagram	Diagramma di sequenza.
Statechart Diagram	Diagramma degli stati.
NRU	Not Registered User (utente non registrato).
RU	Registered User (utente registrato).
USR	User (utente, non registrato o registrato).
BUY	Buyer (compratore).
SEL	Seller (venditore).
SYS	System (sistema).

CAPITOLO 2

Requirement Elicitation

2.1 Obiettivo

A seguito di una intervista agli stakeholders per carpire informazioni sul dominio del problema, nonché le funzionalità e le qualità che il software debba possedere per risultare conforme ai bisogni dell'utente, si elencano di seguito tutti i requisiti individuati:

2.2 Introduzione al sistema

2.2.1 Requisiti funzionali

Utente non registrato

REQ-NRU-01

L'utente che non ha effettuato l'accesso può registrarsi con email correttamente formattata e password.

REQ-NRU-02

L'utente che non ha effettuato l'accesso può registrarsi con un account Facebook già esistente.

REQ-NRU-03

L'utente che non ha effettuato l'accesso può registrarsi con un account Google già esistente.

REQ-NRU-04

L'utente che non ha effettuato l'accesso può selezionare se il suo nuovo account è di tipo compratore o venditore.

REQ-NRU-05

L'utente che non ha effettuato l'accesso può creare uno e un solo account di tipo compratore con la stessa email.

REQ-NRU-06

L'utente che non ha effettuato l'accesso può creare uno e un solo account di tipo venditore con la stessa email.

REQ-NRU-07

L'utente che non ha effettuato l'accesso può creare un profilo condiviso tra l'account venditore e l'account compratore con la stessa email. La creazione del profilo avviene subito dopo la fase di registrazione account.

REQ-NRU-08

L'utente che non ha effettuato l'accesso può accedere a un account con l'email e password utilizzati nella registrazione.

REQ-NRU-09

L'utente che non ha effettuato l'accesso può accedere con un account Facebook già esistente.

REQ-NRU-10

L'utente che non ha effettuato l'accesso può accedere con un account Google già esistente.

Utente registrato

REQ-RU-01

L'utente che ha effettuato l'accesso può visualizzare il suo profilo.

REQ-RU-02

L'utente che ha effettuato l'accesso può modificare i dati sul suo profilo, come biografia, link al proprio sito, link social e area geografica.

REQ-RU-03

L'utente che ha effettuato l'accesso può visualizzare le aste che ha creato.

REQ-RU-04

L'utente che ha effettuato l'accesso può eliminare le aste che ha creato.

REQ-RU-05

L'utente che ha effettuato l'accesso può visualizzare le aste a cui ha partecipato.

REQ-RU-06

L'utente che ha effettuato l'accesso può visualizzare l'elenco delle offerte proposte alle aste da lui create.

REQ-RU-07

L'utente che ha effettuato l'accesso può visualizzare le proprie notifiche.

Utente generico

REQ-USR-01

L'utente (sia che abbia effettuato l'accesso, sia che non l'abbia effettuato) può ricercare le aste attraverso parole chiave.

REQ-USR-02

L'utente (sia che abbia effettuato l'accesso, sia che non l'abbia effettuato) può ricercare le aste sulla base della categoria.

REQ-USR-03

L'utente (sia che abbia effettuato l'accesso, sia che non l'abbia effettuato) può ricercare le aste attraverso parole chiave e filtrando sulla base della categoria.

REQ-USR-04

L'utente (sia che abbia effettuato l'accesso, sia che non l'abbia effettuato) può visualizzare i dettagli di un'asta.

REQ-USR-05

L'utente (sia che abbia effettuato l'accesso, sia che non l'abbia effettuato) può visualizzare il profilo del proprietario dell'asta.

REQ-USR-06

L'utente (sia che abbia effettuato l'accesso, sia che non l'abbia effettuato) può visualizzare le aste attive.

REQ-USR-07

L'utente (sia che abbia effettuato l'accesso, sia che non l'abbia effettuato) può visualizzare l'attuale offerta più alta dell'asta di tipo "a tempo fisso".

REQ-USR-08

L'utente (sia che abbia effettuato l'accesso, sia che non l'abbia effettuato) può visualizzare l'attuale offerta più bassa dell'asta di tipo "inversa".

REQ-USR-09

L'utente (sia che abbia effettuato l'accesso, sia che non l'abbia effettuato) può accedere ad una sezione di aiuto per leggere istruzioni sull'utilizzo dell'applicativo.

Compratore**REQ-BUY-01**

Il compratore può creare aste di acquisto di un prodotto/servizio. L'asta risulterà attiva (cioè sarà possibile presentare offerte) fin da subito.

REQ-BUY-02

Il compratore può creare un'asta di tipo "inversa".

REQ-BUY-03

Il compratore di un'asta di tipo "inversa" può inserire, al momento della creazione dell'asta, una data e ora di scadenza, un titolo, una descrizione, una categoria, un prezzo di partenza che è disposto a pagare e opzionalmente una o più fotografie per l'asta da lui creata.

REQ-BUY-04

Il compratore di un'asta di tipo "inversa" può modificare la data e ora di scadenza, il titolo, la descrizione, la categoria e le fotografie (se aggiunte) per l'asta da lui creata.

REQ-BUY-05

Il compratore che vuole acquistare quel particolare prodotto/servizio può presentare offerte per le aste di tipo "a tempo fisso" o "silenziosa" entro la data e ora di scadenza dell'asta. Tale offerta è una somma in euro sempre maggiore di 0, e per l'asta di tipo "a tempo fisso" deve anche essere maggiore dell'attuale offerta più alta.

Venditore**REQ-SEL-01**

Il venditore può creare aste di vendita di un prodotto/servizio. L'asta risulterà attiva (cioè sarà possibile presentare offerte) fin da subito.

REQ-SEL-02

Il venditore può selezionare il tipo di asta da creare. L'asta può essere di tipo "a tempo fisso" o "silenziosa".

REQ-SEL-03

Il venditore di un'asta di qualsiasi tipo può inserire, al momento della creazione dell'asta, una data e ora di scadenza, un titolo, una descrizione, una categoria e opzionalmente una o più fotografie per l'asta da lui creata. Per l'asta di tipo "a tempo fisso", può anche specificare un prezzo minimo da raggiungere.

REQ-SEL-04

Il venditore di un'asta di qualsiasi tipo può modificare la data e ora di scadenza, il titolo, la descrizione, la categoria e le fotografie (se aggiunte) per l'asta da lui creata.

REQ-SEL-05

Il venditore in grado di fornire quel particolare prodotto/servizio può presentare offerte per le aste di tipo "inversa" entro la data e ora di scadenza dell'asta. Tale offerta è una somma in euro minore rispetto alla somma più bassa attualmente raggiunta (tale valore deve essere maggiore di 0).

Sistema

REQ-SYS-01
Il sistema assegna la vincita al compratore di un asta di tipo "a tempo fisso" che ha offerto la somma più alta entro la data e ora di scadenza dell'asta.
REQ-SYS-02
Il sistema dichiara l'asta fallita se nessun compratore di un asta di tipo "a tempo fisso" ha offerto una somma più alta della soglia minima entro la data e ora di scadenza dell'asta.
REQ-SYS-03
Il sistema dichiara l'asta fallita se non è stata avanzata alcuna offerta da parte di un compratore di un'asta di tipo "a tempo fisso" entro la data e ora di scadenza dell'asta.
REQ-SYS-04
Il sistema mostra l'offerta di un'asta di tipo "silenziosa" solo al venditore di tale asta e a colui che ha proposto l'offerta.
REQ-SYS-05
Il sistema assegna la vincita dell'asta al compratore di un'asta di tipo "silenziosa" la cui offerta è stata accettata dal venditore dell'asta di tipo "silenziosa".
REQ-SYS-06
Il sistema dichiara l'asta fallita se il venditore di un'asta di tipo "silenziosa" non ha accettato alcuna offerta entro la data e ora di scadenza dell'asta.
REQ-SYS-07
Il sistema dichiara l'asta fallita se non è stata avanzata alcuna offerta da parte di un compratore di un'asta di tipo "silenziosa" entro la data e ora di scadenza dell'asta.
REQ-SYS-08
Il sistema invia una notifica di proposta offerta al venditore dell'asta di tipo "silenziosa" che ha ricevuto una nuova offerta alla propria asta.
REQ-SYS-09
Il sistema invia una notifica di rifiuto dell'offerta al partecipante dell'asta di tipo "silenziosa" la cui offerta è stata rifiutata dal venditore dell'asta.
REQ-SYS-10
Il sistema assegna la vincita dell'asta di tipo "inversa" al venditore che ha offerto la somma più bassa al sopraggiungere della data e ora di scadenza dell'asta a cui ha partecipato.
REQ-SYS-11
Il sistema dichiara l'asta di tipo "inversa" fallita se non è stata avanzata alcuna offerta da parte di un venditore entro la data e ora di scadenza dell'asta.
REQ-SYS-12
Il sistema controlla che l'email abbia il formato corretto (ovvero sia del tipo x+@y+.z+) e che la password sia sicura (ovvero sia lunga almeno 8 caratteri e contenga, nel complesso, almeno una lettera maiuscola [A-Z], una lettere minuscola [a-z], un numero [0-9] e un carattere speciale [!@#\$%^&*]).
REQ-SYS-13
Il sistema invia una notifica di chiusura dell'asta a tutti i partecipanti dell'asta di tipo "a tempo fisso" (ovvero il venditore e tutti i compratori che hanno offerto almeno una volta una somma di denaro).

REQ-SYS-14

Il sistema invia una notifica di chiusura dell'asta a tutti i partecipanti dell'asta di tipo "silenziosa" (ovvero a tutti i compratori che hanno offerto almeno una volta una somma di denaro) entro 10 secondi dal momento in cui è stata accettata un'offerta. Tale notifica indicherà se la propria offerta è stata accettata o rifiutata.

REQ-SYS-15

Il sistema invia una notifica di chiusura dell'asta a tutti i partecipanti dell'asta di tipo "inversa" (ovvero il compratore e tutti i venditori che hanno offerto almeno una volta una somma di denaro).

2.2.2 Requisiti non funzionali

REQ-A

Il sistema deve far recapitare le notifiche agli utenti in meno di 10 secondi.

REQ-B

Il sistema deve elaborare quali aste hanno raggiunto la data di scadenza entro 5 secondi dalla loro conclusione.

REQ-C

Le operazioni di rete effettuate dal sistema non devono richiedere più di un secondo.

REQ-D

Il sistema non deve utilizzare più di 500 Mb della memoria della macchina che lo esegue.

CAPITOLO 3

Requirement Analysis

3.1 Obiettivo

Le informazioni ottenute attraverso la raccolta sono state organizzate attraverso l'uso di diagrammi ad alto livello, quali diagramma UML dei casi d'uso, tabelle di Cockburn e mock-up.
Sono stati anche individuati gli utenti target del sistema, nonché sono stati effettuati test di usabilità a priori sui mock-up dell'interfaccia utente.

3.2 Use Case Diagram

Il diagramma dei casi d'uso indica per ciascun attore (utente che utilizzerà il sistema) quali sono le funzionalità alle quali egli potrà accedere.

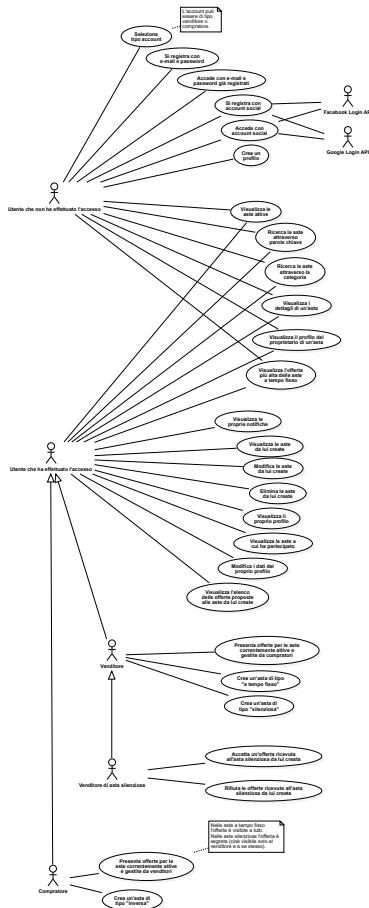


Figura 3.1: Use Case Diagram

3.3 Utenti target

L'individuazione di utenti target risulta cruciale per sviluppare un applicativo che possa essere indirizzato ad una specifica demografica.

In particolare, permette di comprendere quali sono i bisogni delle diverse tipologie di utenti, rendendo semplici le operazioni che hanno bisogno di effettuare più spesso ciascuno di loro e risolvere le frustrazioni che essi possono riscontrare nei confronti di applicativi già esistenti.

Lo studio avviene attraverso la creazione di mock-up definiti "personas". Abbiamo individuato in totale sei gruppi di utenti, dei quali i più prominenti sono quelli rappresentati da Maria Lombardo (persona primaria) e Luca Serra (persona secondaria).



Figura 3.2: Maria Lombardo

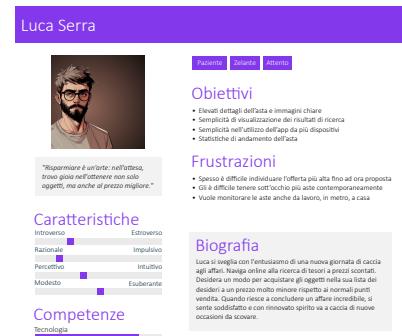


Figura 3.3: Luca Serra



Figura 3.4: Marco Rossi



Figura 3.5: Martina Silvestri

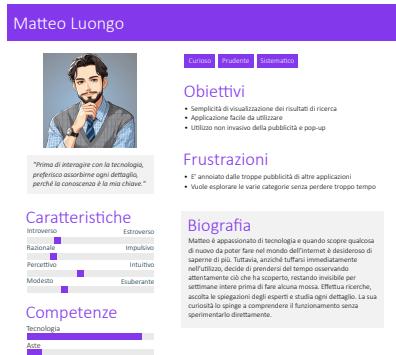


Figura 3.6: Matteo Luongo



Figura 3.7: Arturo Campobello

3.4 Mock-up

La realizzazione dell’interfaccia è stata eseguita attraverso il software di prototipazione Figma (il link al mock-up è visualizzabile [qui](#)).

Il software ha consentito di individuare l’assetto generale dell’interfaccia utente per l’esecuzione di ogni caso d’uso individuato, nonché indicare le trasformazioni dell’interfaccia stessa in caso di errori del sistema.

3.4.1 Logo

Il logo dell’applicazione è un’immagine che richiama i mondi dell’informatica e del mercato delle aste: un monitor simboleggia la natura informatizzata dei servizi offerti, mentre il martelletto che compare sullo schermo rimanda al mondo del mercato delle aste; sotto questi elementi, il nome dell’applicazione “DietiDeals24”.

3.4.2 Colori

Riferendosi alla teoria dei colori, si è scelto di impiegare una paletta cromatica ben specifica. I colori caldi sono utilizzati per riflettere passione, felicità, entusiasmo, energia; i colori freddi sono invece utilizzati per trasmettere calma e professionalità.

I colori principali dominanti nell’interfaccia sono l’arancione e il blu:

- L’arancione è spesso associato alla creatività, al cambiamento (dovuto al suo rimando ai colori autunnali), ma anche a salubrità e vitalità.

Esso permette di tenere viva l’attenzione senza allarmare quanto il rosso, spesso utilizzato per il pericolo e l’importanza.

In seno alla sua scelta risiede la volontà di associare la nostra applicazione a un’opera creativa e vivace, capace di catturare l’utente.

- Il blu non necessariamente deve implicare tristezza.

Spesso, viene utilizzato per trasmettere calma e responsabilità.

Principalmente, i toni più chiari sono utilizzati per instillare la tranquillità nell’osservatore, mentre quelli più scuri sono spesso scelti dai brand premium per conferire un tocco di forza ed affidabilità allo stesso.

La scelta del blu scuro risiede proprio nella volontà di far percepire la nostra applicazione come affidabile e professionale.

Un ruolo importante nella scelta di questi due colori ha giocato anche la loro complementarietà: essendo situati a due poli opposti dello spettro cromatico, il contrasto tra i due risulta notevolmente accentuato, evidenziando maggiormente gli elementi dell’interfaccia.

Lo sfondo invece impiega un colore grigio tendente al blu, risultando in una sfumatura molto chiara di quest’ultimo.

Tenendo conto che i due “colori” bianco e nero, se utilizzati puri possono affaticare troppo la vista dell’utente, questa scelta permette sia di ridurre il fastidio visivo, sia di provocare come tutti i blu più chiari un senso di calma.

Altri colori impiegati sono il rosso, principalmente per richiamare l’attenzione su elementi dell’interfaccia che comportano azioni importanti o irreversibili, ma anche per sottolineare messaggi di errore dell’applicazione.

Il verde, invece, adempie funzioni opposte al rosso, come ad esempio indicare bottoni di annullamento di un’operazione critica.

Il blu chiaro sottolineato è stato utilizzato per le parole (o frasi) cliccabili, essendo questa una convenzione ben radicata nell’ambito delle interfacce grafiche.

Il grigio ed il bianco mantengono una funzione conforme ai normali standard di realizzazione dell’interfaccia:

- Il grigio è utilizzato per quei campi con i quali non si può interagire, come ad esempio i pulsanti nella barra di navigazione che senza account non devono permettere azione da parte dell'utente.
- Il bianco, infine, viene sfruttato per evidenziare campi con i quali è possibile interagire, come ad esempio i campi di testo.

3.5 Tabelle di Cockburn

Basandosi sull'interfaccia realizzata, è stata effettuata la stesura delle tabelle di Cockburn di quattro casi d'uso significativi.

Lo scopo delle tabelle è quello di presentare in maniera sequenziale come giungere all'esecuzione di un caso d'uso interagendo con l'interfaccia utente.

Sono stati evidenziati anche possibili metodi alternativi per raggiungere lo stesso scopo o errori che potrebbero sorgere.

3.5.1 I casi d'uso: Crea un'asta inversa

USE CASE #1			
DESCRIPTION	Crea un'asta inversa		
	Goal in Context	L'utente di tipo "compratore" vuole creare un'asta inversa	
	Preconditions	L'utente ha effettuato l'accesso con un account di tipo "compratore"	
	Success End Condition	L'utente di tipo "compratore" ha correttamente creato un'asta "inversa"	
	Step n°	Compratore	Sistema
	1	Preme bottone "Crea asta" su mock-up C1 (Figura 3.8)	
	2		Mostra mock-up P0A (Figura 3.9) con testo "Seleziona tipo di asta" dove è possibile scegliere il tipo di asta da creare
	3	Seleziona "Inversa" e clicca "Conferma" sul mock-up P0A (Figura 3.9)	
	4		Mostra mock-up C4 (Figura 3.10) con tutte le informazioni dell'asta inversa da poter inserire
	5	Preme sul campo di input "Data di scadenza"	
	6		Mostra il pannello di selezione della data da calendario
	7	Seleziona la data	
	8	Clicca su OK	
	9		Scrive la data selezionata nel campo di input "Data di scadenza"
	10	Preme sul campo di input "Ora di scadenza"	
	11		Mostra il pannello di selezione dell'ora
	12	Seleziona l'ora	
	13	Clicca su OK	
	14		Scrive l'ora selezionata nel campo di input "Ora di scadenza"
	15	Preme sul campo di input "Prezzo di partenza"	
	16		Mostra il tastierino numerico
	17	Inserisce il prezzo di partenza	

	18	Preme sul campo di input "Nome prodotto"	
	19		Mostra la tastiera
	20	Inserisce il nome del prodotto oggetto dell'asta	
	21	Preme sul campo di input "Categoria"	
	22		Mostra il menù a tendina con le diverse categorie tra cui scegliere
	23	Seleziona la categoria	
	24		Scrive la categoria scelta nel campo di input "Categoria"
	25	Preme sul campo di input "Descrizione"	
	26		Mostra la tastiera
	27	Inserisce la descrizione del prodotto oggetto dell'asta	
	28	Preme sul pulsante "Crea"	
	29		Mostra mock-up P20 (Figura 3.11, Pop-up di successo)
	30	Preme il pulsante X sul mock-up P20 (Figura 3.11)	
	31		Torna al mock-up C1 (Figura 3.8, Finestra Home)
EXTENSIONS	Step n°	Compratore	Sistema
Seleziona asta diversa da quella inversa	3.a	Seleziona tipo di asta diversa da "Inversa" e clicca "Conferma" sul mock-up P0A (Figura 3.9)	
	4.a		Mostra mock-up di creazione asta relativo all'asta selezionata
Seleziona data antecedente a quella odierna	9.b		Mostra mock-up P19 (Figura 3.12) con testo "Non puoi inserire una data antecedente a quella odierna!"
	10.b	Clicca su X sul mock-up P19 (Figura 3.12)	
	11.b		Continua da step 3 di main scenario
Seleziona data odierna e ora antecedente a quella attuale	14.c		Mostra mock-up P14 (Figura 3.13) con testo "Non puoi selezionare un'ora antecedente a quella attuale in data odierna!"
	16.c	Clicca su X sul mock-up P14 (Figura 3.13)	
	17.c		Continua da step 8 di main scenario
Seleziona prezzo di partenza negativo	18.d		Mostra mock-up P10 (Figura 3.14) con testo "Non puoi inserire una somma di partenza minore di 0 €."
	19.d	Clicca su X sul mock-up P10 (Figura 3.14)	

	20.d		Continua da step 13 di main scenario
Alcuni campi obbligatori non compilati	29.e		Mostra mock-up E12 (Figura 3.15) dove vengono segnalati in rosso i campi obbligatori non compilati
	30.e	Riparte da step 3 di main scenario, saltando i campi di input già compilati	
Esce dalla pagina di creazione asta	<i>In qualche passo del main scenario</i>	Preme qualsiasi tasto di navigazione (della navbar in basso o tasto "indietro" del dispositivo)	
			Mostra mock-up P6 (Figura 3.16) con testo "Se uscirai da questa schermata, i dati inseriti nei campi saranno cancellati. Vuoi proseguire?"
		Clicca su Esci sul mock-up P6 (Figura 3.16)	
			Mostra mock-up corrispondente al tasto cliccato
SUBVARIATIONS	Step n°	Compratore	Sistema
Inserisce immagine	18.s1	Preme sul campo di input "Aggiungi immagine prodotto"	
	19.s1		Mostra il sistema per selezionare le immagini
	20.s1	Selezione una o più immagini	
	21.s1	Clicca su OK	
	22.s1		Continua da step 16 di main scenario



Figura 3.8: Home compratore



Figura 3.9: Popup scelta tipo di asta da creare



Figura 3.10: Crea asta compratore



Figura 3.11: Successo creazione asta

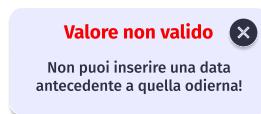


Figura 3.12: Errore data non valida



Figura 3.13: Errore ora non valida



Figura 3.14: Errore somma di partenza non valida

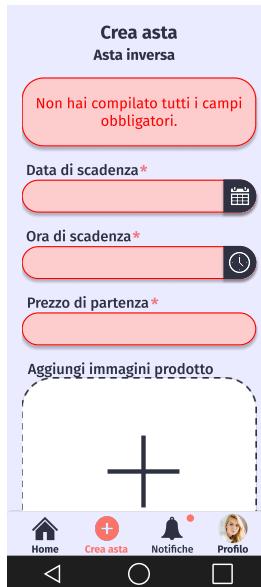


Figura 3.15: Errore campi obbligatori non compilati



Figura 3.16: Richiesta conferma uscita

3.5.2 II caso d'uso: Accetta un'offerta ricevuta all'asta silenziosa

USE CASE #2	Accetta un'offerta ricevuta all'asta silenziosa		
Goal in Context	L'utente di tipo "venditore" vuole accettare un'offerta ricevuta in un'asta "silenziosa" da lui creata		
Preconditions	L'utente ha effettuato l'accesso con un account di tipo "venditore" e ha creato un'asta "silenziosa"		
Success End Condition	L'utente di tipo "venditore" ha correttamente accettato un'offerta di un'asta "silenziosa" da lui creata		
DESCRIPTION	Step n°	Venditore di asta silenziosa	Sistema
	1	Preme bottone "Profilo" su mock-up V1 (Figura 3.17)	
	2		Mostra mock-up V7 (Figura 3.18) con tutte le opzioni per utenti loggati
	3	Preme sul bottone "Le mie asta create"	
	4		Mostra il mock-up V11 (Figura 3.19) con tutte le asta create dal venditore
	5	Clicca sull'icona a forma di elenco dell'asta di cui si vogliono visualizzare le offerte ricevute	
	6		Mostra mock-up V13 (Figura 3.20) con l'elenco di tutte le offerte ricevute
	7	Clicca sulla spunta verde dell'offerta che si vuole accettare	
	8		Mostra il mock-up P7 (Figura 3.21) con testo "Confermi di voler accettare questa offerta? Tutte le altre saranno automaticamente rifiutate."
	9	Clicca sul pulsante "Accetta"	
EXTENSIONS	10		Mostra mock-up V14 (Figura 3.22) dove vengono elencate tutte le offerte ricevute a questa asta "silenziosa". L'offerta accettata sarà l'unica in bianco, tutte le altre saranno in grigio. Viene inviata una notifica a tutti coloro che hanno partecipato almeno una volta all'asta per indicare la chiusura della stessa
	Step n°	Venditore di asta silenziosa	Sistema
Esce dalla pagina di gestione delle offerte ricevute	<i>In qualunque passo del main scenario</i>	Preme qualsiasi tasto di navigazione (della navbar in basso o tasto "indietro" del dispositivo)	

			Mostra mock-up corrispondente al tasto cliccato
SUBVARIATIONS	Step n°	Venditore di asta silenziosa	Sistema
Visualizza i dettagli dell'asta silenziosa selezionata	5.s1	Preme sulla parte bianca dell'anteprima dell'asta	
	6.s1		Mostra il mock-up V17 (Figura 3.23) con i dettagli dell'asta selezionata
	7.s1	Clicca sull'icona a forma di elenco dell'asta di cui si vogliono visualizzare le offerte ricevute	
	8.s1		Continua con step 6 di main scenario
Riceve notifica di proposta offerta per l'asta silenziosa	1.s2	Preme sul bottone "Notifiche"	
	2.s2		Mostra il mock-up V6 (Figura 3.24) con la notifica relativa all'offerta ricevuta
	3.s2	Clicca sulla notifica	
	4.s2		Continua con step 6 di subvariation s1



Figura 3.17: Home venditore



Figura 3.18: Menu profilo venditore



Figura 3.19: Aste create venditore

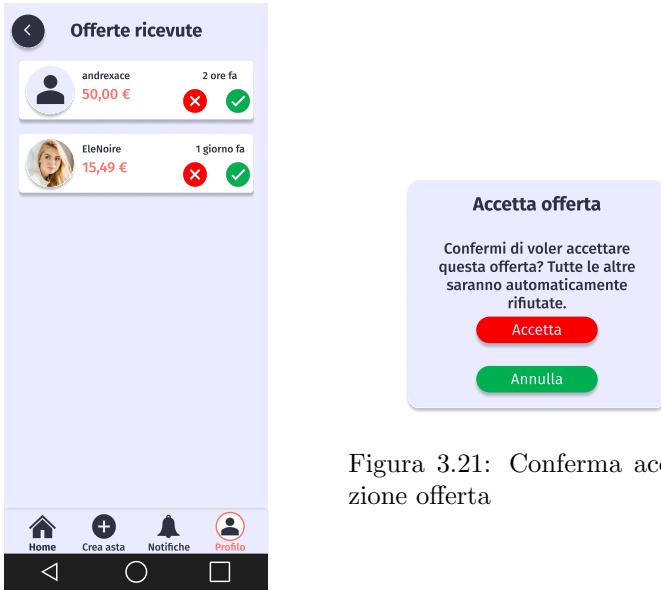


Figura 3.21: Conferma accettazione offerta

Figura 3.20: Offerte ricevute asta silenziosa

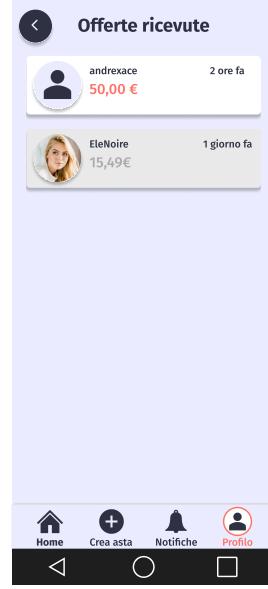


Figura 3.22: Offerte ricevute asta silenziosa dopo aver accettato un'offerta



Figura 3.23: Dettagli asta selezionata



Figura 3.24: Notifiche

3.5.3 III caso d'uso: Visualizza i dettagli di un'asta

USE CASE #3 Visualizza i dettagli di un'asta			
Goal in Context	L'utente di un qualsiasi tipo vuole visualizzare i dettagli di un'asta creata da un altro utente		
Preconditions	L'utente si trova nella home		
Success End Condition	L'utente visualizza correttamente la pagina con tutti i dettagli dell'asta		
DESCRIPTION	Step n°	Utente	Sistema
	1	Preme sull'area bianca della card di anteprima dell'asta su mock-up C1 (Figura 3.25)	
	2		Mostra mock-up C7 (Figura 3.26) con tutte le informazioni circa l'asta
EXTENSIONS	Step n°	Utente	Sistema
Esce dalla pagina di dettagli dell'asta	<i>In qualunque passo del main scenario</i>	Preme qualsiasi tasto di navigazione (della navbar in basso o tasto "indietro" del dispositivo), il tasto indietro dell'app, il nome o l'immagine del creatore dell'asta	
			Mostra mock-up corrispondente all'elemento cliccato
SUBVARIATIONS	Step n°	Utente loggato	Sistema
Riceve notifica circa un'offerta, la fine o la vittoria all'asta	1.s1	Preme sul bottone "Notifiche"	
	2.s1		Mostra il mock-up C5 (Figura 3.27) con la notifica relativa all'asta
	3.s1	Clicca sulla notifica	
	4.s1		Passa allo step 2 del main scenario
È nella pagina delle aste alle quali ha partecipato	1.s2	Preme sull'area bianca della card di anteprima dell'asta su mock-up C15 (Figura 3.28)	
	2.s2		Passa allo step 2 del main scenario
È nella pagina delle aste che ha creato	1.s2	Preme sull'area bianca della card di anteprima dell'asta su mock-up C14 (Figura 3.29)	
	2.s2		Passa allo step 2 del main scenario



Figura 3.25: Home utente



Figura 3.26: Dettagli asta

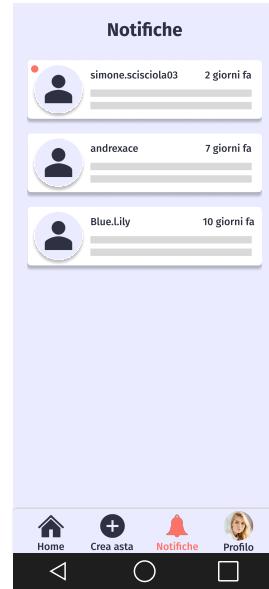


Figura 3.27: Notifiche



Figura 3.28: Aste partecipate



Figura 3.29: Aste create

3.5.4 IV caso d'uso: Modificare il proprio profilo

USE CASE #4	Modificare il proprio profilo		
Goal in Context	L'utente che ha effettuato l'accesso vuole modificare i dettagli salvati nel proprio profilo		
Preconditions	L'utente ha effettuato l'accesso con un account di tipo qualsiasi		
Success End Condition	L'utente ha modificato con successo le informazioni sul suo profilo personale		
DESCRIPTION	Step n°	Utente loggato	Sistema
	1	Preme bottone "Profilo" su mock-up C1 (Figura 3.30)	
	2		Mostra mock-up C6 (Figura 3.31) con tutte le opzioni per utenti loggati
	3	Preme sul bottone "Informazioni utente"	
	4		Mostra mock-up C11 (Figura 3.32) con tutte le informazioni del profilo
	5	Preme sul pulsante di modifica a forma di matita in basso a destra	
	6		Mostra mock-up C12 (Figura 3.33) con i campi aperti per la modifica
	7	Preme sul campo "Nome"	
	8		Mostra la tastiera
	9	Inserisce il proprio nome nel campo	
	10	Preme sul campo "Cognome"	
	11		Mostra la tastiera
	12	Inserisce il proprio cognome nel campo	
	13	Preme sul campo "Data di nascita"	
	14		Mostra il calendario per la selezione della data di nascita
	15	Seleziona la data di nascita e preme su OK	
	16	Preme sul pulsante di conferma in alto a destra	
	17		Mostra mock-up P22 (Figura 3.34)
	18	Preme su "X" del popup	
	19		Ritorna al mock-up C11 (Figura 3.32) con le informazioni aggiornate
EXTENSIONS	Step n°	Utente loggato	Sistema
Alcuni campi obbligatori non compilati	17.e		Mostra mock-up E14 (Figura 3.35) dove vengono segnalati in rosso i campi obbligatori non compilati

	18.e	Riparte da step 7 di main scenario, saltando i campi di input già compilati	
Esce dalla pagina di modifica del profilo	<i>In qualunque passo del main scenario</i>	Preme qualsiasi tasto di navigazione (della navbar in basso o tasto "indietro" del dispositivo)	
			Mostra mock-up P6 (Figura 3.36) con testo "Se uscirai da questa schermata, i dati inseriti nei campi saranno cancellati. Vuoi proseguire?"
		Clicca su Esci sul mock-up P6 (Figura 3.36)	
			Mostra mock-up corrispondente al tasto cliccato
SUBVARIATIONS	Step n°	Utente loggato	Sistema
Inserisce immagine	7.s1	Preme sul tasto "+" accanto alla sua immagine di profilo	
	8.s1		Mostra il sistema per selezionare le immagini
	9.s1	Seleziona un'immagine	
	10.s1	Clicca su OK	
	11.s1		Continua da step 7 di main scenario
Inserisce genere	16.s2	Preme sul campo di input del genere	
	17.s2		Mostra la tastiera
	18.s2	Inserisce il proprio genere	
	19.s2		Continua da step 16 di main scenario
Inserisce area geografica	16.s3	Preme sul campo di input dell'area geografica	
	17.s3		Mostra la tastiera
	18.s3	Inserisce la propria area geografica	
	19.s3		Continua da step 16 di main scenario
Inserisce biografia	16.s4	Preme sul campo di input della biografia	
	17.s4		Mostra la tastiera
	18.s4	Inserisce la propria biografia	
	19.s4		Continua da step 16 di main scenario
Inserisce link personale	16.s5	Preme sul campo di input del link personale	
	17.s5		Mostra la tastiera
	18.s5	Inserisce il proprio link personale	

	19.s5		Continua da step 16 di main scenario
Inserisce link social	16.s6	Preme sul pulsante di uno dei social	
	17.s6		Mostra il popup P4 (Figura 3.37)
	18.s6	Preme sul campo di testo del link social	
	19.s6		Mostra la tastiera
	20.s6	Inserisce il proprio link social	
	21.s6	Preme su "Conferma"	
	22.s6		Torna al mockup C12 (Figura 3.33)
	23.s6		Continua da step 16 del main scenario



Figura 3.30: Home utente



Figura 3.31: Menu profilo utente



Figura 3.32: Profilo personale



Figura 3.33: Modifica profilo personale

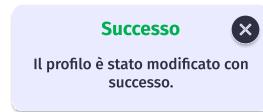


Figura 3.34: Pop-up di successo



Figura 3.35: Errore campi non compilati



Figura 3.36: Popup di conferma uscita



Figura 3.37: Popup di inserimento link social

3.6 Valutazione dell'usabilità a priori

La valutazione dell'usabilità consiste nel verificare l'aderenza dell'interfaccia alle 8 regole d'oro di Ben Shneiderman o alle 10 euristiche di Nielsen. In particolare, alcuni punti forti individuati sono stati:

- Consistenza: Lo stile dell'applicazione, sia nei colori che nelle dimensioni di font ed elementi dell'interfaccia rimane costante e coerente.
- Standard: L'interfaccia utilizza convenzioni dell'industria sia nel posizionamento degli elementi sull'interfaccia e sia nelle icone utilizzate.
- Scorciatoie: L'applicazione permette agli utenti di effettuare delle operazioni con un numero estremamente basso di tap e re-direzioni.
- Messaggi informativi: L'applicazione permette agli utenti di essere a conoscenza di esiti negativi delle loro operazioni grazie a dei pop-up informativi. Lo stesso vale per l'esito positivo di una operazione.
- Memoria a breve termine: L'applicazione ha un'interfaccia minimale senza troppi ingombri visivi che possono sovraccaricare la memoria a breve termine dell'utente; in tal modo non si sentirà disorientato dalla mole di informazioni mostrate su schermo.
- Corrispondenza con il mondo reale: L'applicazione utilizza un linguaggio semplice e non costringe gli utenti ad informarsi attraverso fonti esterne per comprendere ciò che gli viene mostrato.
- Controllo dell'utente e prevenzione errori: L'applicazione impedisce all'utente di effettuare delle operazioni importanti senza prima richiedere una conferma, così da consentire di cambiare idea oppure prevenire errori di tap.
- Aiuto e documentazione: L'applicazione fornisce una sezione di aiuto interna che permette all'utente di capire come effettuare passo per passo ogni operazione più importante.

3.6.1 Valutazioni con utenti

Sono stati effettuati una serie di valutazioni di usabilità attraverso i prototipi costruiti con Figma. In particolare, nelle tabelle seguenti sono riportati i casi d'uso testati e, per ogni utente, viene indicato con:

- S = Successo (ovvero la task è stata completata velocemente e senza intoppi)
- P = Successo parziale (ovvero è stato necessario un po' di tempo in più e piccoli suggerimenti)
- F = Fallimento (ovvero la task non è stata completata nonostante i piccoli suggerimenti)
- X = Non testato

Al fine di quantificare la bontà di realizzazione dell'interfaccia e la riposta degli utenti ad essa (in particolare il primo, utilizzeremo un sistema di punteggio per ogni caso d'uso).

Il punteggio è riferito al primo test di usabilità sulla prima versione del prototipo.

Il campione di utenti in esame distingue ogni individuo per sesso, età, livello di inclinazione al digitale e contesto familiare.

In particolare seguiranno le seguenti regole:

- Per ogni caso d'uso, il punteggio massimo attribuibile è da 0 ad N (dove N è il numero di utenti che hanno testato quel caso d'uso).
- Se l'utente ha testato il caso d'uso con successo (S), si somma al totale 1.

	Registrazione account	Accesso account	Ricerca aste	Filtra aste	Visualizza dettagli asta	Visualizza profilo proprietario asta	Visualizza le tue aste create	Modifica la tua asta	Elimina la tua asta
Utente 1	S	S	S	S	S	S	S	S	S
Utente 2	S	S	S	P	S	P	S	S	S
Utente 3	S	S	S	S	S	S	S	S	S
Utente 4	S	S	S	S	S	P	S	S	S
Utente 5	S	S	S	S	P	P	P	S	S
Utente 6	S	S	S	S	P	S	P	S	S
Utente 7	S	S	X	X	P	P	S	S	S
Utente 8	S	S	S	S	S	S	P	S	S
Utente 9	S	S	X	X	S	S	S	P	S

	Visualizza il tuo profilo	Visualizza aste a cui hai partecipato	Modifica il tuo profilo	Visualizza offerte per la tua asta	Fai un'offerta ad un'asta	Crea asta	Accetta offerte asta silenziosa	Rifiuta offerte asta silenziosa
Utente 1	S	S	S	S	S	S	X	X
Utente 2	S	S	P	S	S	S	X	X
Utente 3	S	S	P	S	S	S	S	S
Utente 4	S	S	P	S	S	S	S	S
Utente 5	S	F	P	S	S	S	S	S
Utente 6	S	P	F	S	X	S	X	X
Utente 7	P	S	P	S	X	S	X	X
Utente 8	S	P	S	S	S	S	X	X
Utente 9	S	S	S	S	S	S	X	X

- Se l'utente ha testato il caso d'uso con parziale successo (P), si somma al totale 0,5.
- Se l'utente ha testato il caso d'uso con insuccesso (F), si somma al totale 0.
- Si calcola infine per ogni caso d'uso la percentuale di successo così:

$$\frac{\sum_{utente=1}^N punteggio(utente)}{\sum_{utente=1}^N 1} = \frac{punteggioOttenuto}{punteggioTotale}.$$

Le percentuali di successo per ogni caso d'uso testato al sono:

- Registrazione account: $\frac{9}{9} = 1 = 100\%$
- Accesso account: $\frac{9}{9} = 1 = 100\%$
- Ricerca aste: $\frac{7}{7} = 1 = 100\%$
- Filtra aste: $\frac{6,5}{7} = 0,93 = 93\%$
- Visualizza dettagli asta: $\frac{7,5}{9} = 0,83 = 83\%$
- Visualizza profilo proprietario asta: $\frac{7}{9} = 0,78 = 78\%$
- Visualizza le tue aste create: $\frac{7,5}{9} = 0,83 = 83\%$
- Modifica la tua asta: $\frac{8,5}{9} = 0,94 = 94\%$
- Elimina la tua asta: $\frac{9}{9} = 1 = 100\%$

- Visualizza il tuo profilo: $\frac{8,5}{9} = 0,94 = 94\%$
- Visualizza asta a cui hai partecipato: $\frac{7}{9} = 0,78 = 78\%$
- Modifica il tuo profilo: $\frac{6,5}{9} = 0,72 = 72\%$
- Visualizza offerte per la tua asta: $\frac{9}{9} = 1 = 100\%$
- Fai un'offerta ad un'asta: $\frac{7}{7} = 1 = 100\%$
- Crea asta: $\frac{9}{9} = 1 = 100\%$
- Accetta offerte asta silenziosa: $\frac{3}{3} = 1 = 100\%$
- Rifiuta offerte asta silenziosa: $\frac{3}{3} = 1 = 100\%$

3.6.2 Modifiche a seguito delle valutazioni

Dalle valutazioni effettuate, sono state riscontrate alcune criticità che hanno comportato una rielaborazione di alcuni aspetti dell'interfaccia.

Dopo la risoluzione di tali problemi, il tasso di successo ha registrato un incremento.

Problematica riscontrata	Modifica apportata
La presenza dello sfondo bianco sotto le singole informazioni delle "informazioni utente" ha indotto alcuni soggetti a cliccare direttamente i campi per effettuare una modifica, ignorando la presenza del pulsantino apposito.	Abbiamo rimosso lo sfondo bianco da tutti gli elementi non cliccabili, concentrando l'attenzione del soggetto solo ed esclusivamente sul pulsante di modifica apposito.
All'interno della schermata "home", la presenza di un nome utente all'interno della scheda di anteprima subito al di sotto dell'offerta più elevata ha erroneamente indotto alcuni soggetti a pensare che quell'utente fosse l'offerente del prezzo, e non il creatore dell'asta come inteso.	Abbiamo rimosso il nome dell'utente nelle schede di anteprima delle aste specificandolo solo ed esclusivamente nei dettagli dell'asta, preceduto dalla dicitura "Creato da".
La presenza della propria asta creata all'interno della "home" ha confuso un soggetto e lo ha spinto a voler offrire una somma di denaro per suddetta asta, quando invece ciò non è possibile essendone lui stesso il creatore.	Abbiamo rimosso dalla "home" le aste a cui l'utente non può partecipare, ossia le aste create da altri account dello stesso tipo e le aste da lui stesso create.
La presenza del tasto "home" nella barra di navigazione dell'applicazione ha portato alcuni soggetti a cliccare su di essa per tornare alla schermata principale una volta effettuata una ricerca per parola chiave o per categoria, ignorando il tasto indietro apposito (del dispositivo).	Abbiamo reso il pulsante "home" utilizzabile anche per tornare alla schermata principale una volta effettuata una ricerca per parola chiave o per categoria.
La presenza di ingleismi (logout, guest) ha confuso i soggetti più anziani, ignari del significato della parola.	Abbiamo reso la nostra applicazione completamente in italiano.
Un utente ha provato ad effettuare lo zoom sulle immagini con un pinch out anziché toccando l'immagine stessa.	Tale azione sarà implementata nell'applicazione finale.

Problematica riscontrata	Modifica apportata
Nella sezione "Aiuto", alcuni utenti hanno cliccato sulla parte bianca del pulsante per accedere ai singoli pop-up informativi; quando era inteso cliccare solo sulla freccia blu.	Abbiamo reso cliccabili ogni singolo pulsante bianco nella sezione "Aiuto".
Un utente non esperto del dominio desiderava conoscere quali fossero le caratteristiche di un determinato tipo di asta direttamente dai dettagli dell'asta e non andando a cercare nella sezione "Aiuto".	Abbiamo aggiunto un "?" che rimanda alle informazioni sul tipo di asta visualizzata. Tale bottone si trova vicino al tipo di asta nella sezione "Dettagli asta", raggiungibile cliccando sull'anteprima di un'asta nella "Home".
Un utente ha trovato difficoltà nel modificare il profilo poichè non vedeva il bottone dedicato a tale funzionalità. In particolare, ha cercato il bottone nella parte bassa dello schermo.	Abbiamo spostato il bottone di "modifica profilo" rendendolo un bottone in sovrapposizione in basso a destra all'interno della sezione delle "informazioni utente".

CAPITOLO 4

Requirement Specification

4.1 Obiettivo

I requisiti che sono stati riorganizzati nella fase di analisi sono formalizzati in diagrammi (diagramma di classe, diagramma di sequenza, diagramma degli stati) attraverso l'uso di linguaggi di modellazione come UML.

Per individuare il modello ad oggetti per questo dominio, si è scelto di impiegare l'euristica "Three-Object-Type" perché permette di costruire un modello più flessibile e facile da modificare.

4.2 Class Diagram

Il Class Diagram individua quali sono le entità coinvolte all'interno del nostro dominio e le relazioni che tra esse sussistono.

In particolare, il modello concettuale basato sull'analisi dei requisiti permetterà di creare un modello efficace per la gestione delle aste.

4.2.1 Class Diagram del dominio del problema

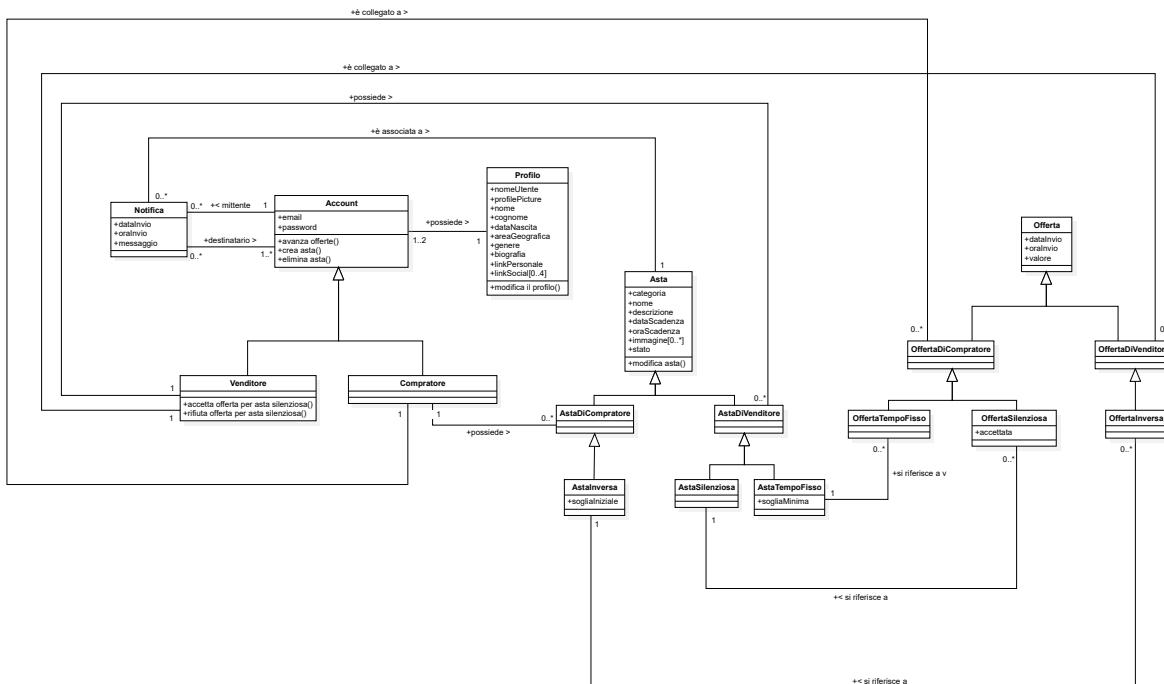


Figura 4.1: Class Diagram del dominio del problema

4.2.2 Dizionario delle classi

Classe	Descrizione
Account	L'account raccoglie le informazioni necessarie per permettere ad un utente dell'applicazione di effettuare l'accesso alla stessa. Esso può essere di due tipologie: compratore e venditore.
Venditore	Un account di tipo venditore è un account che rappresenta un utente che utilizza l'applicazione per vendere i propri beni o servizi ad altri utenti.
Compratore	Un account di tipo compratore è un account che rappresenta un utente che utilizza l'applicazione per acquistare beni o servizi da altri utenti.
Profilo	Il profilo concentra le informazioni di natura personale che riguardano l'utente registrato. Esso è condiviso tra l'account compratore e venditore di uno stesso utente.
Notifica	Una notifica è un messaggio che viene recepito da un utente; ciò avviene quando un'asta alla quale ha partecipato si è conclusa (ed eventualmente vinta), la sua offerta è stata accettata in un'asta silenziosa, o un'asta creata ha ricevuto un'offerta da un'utente.
Asta	Un'asta è un annuncio di compravendita di un bene o un servizio e che raccoglie tutte le informazioni che riguardano l'articolo e chi ha pubblicato questo annuncio. Anche questa può essere di tipo compratore o venditore.
Asta di compratore	Un'asta di tipo compratore è un'asta nella quale il compratore richiede a dei venditori di fornirgli un bene o un servizio. Essa è di un'unica tipologia, ossia un'asta inversa.
Asta di venditore	Un'asta di tipo venditore è un'asta nella quale il venditore fornisce a dei compratori un bene o un servizio da essi richiesto. Essa può essere di due tipologie, ossia un'asta a tempo fisso o un'asta silenziosa.
Asta a tempo fisso	Un'asta a tempo fisso è un'asta nella quale si stabilisce una data di fine, entro la quale i compratori possono fare delle offerte al rialzo, e una soglia minima da raggiungere. Quando sopraggiunge la data stabilita, la persona che ha offerto la cifra più alta si aggiudica l'asta. Se non c'è stata nessuna offerta o non si è raggiunta la soglia minima entro il tempo limite, allora l'asta è dichiarata fallita.
Asta silenziosa	Un'asta silenziosa è un'asta nella quale si stabilisce una data di fine, entro la quale i compratori possono fare offerte al rialzo. Gli offerenti non possono però vedere l'offerta attuale più alta, e quindi dovranno offrire alla cieca. Quando sopraggiunge la data stabilita, la persona della quale l'offerta è stata accettata si aggiudica l'asta. Se non c'è stata nessuna offerta o nessuna offerta è stata accettata entro il tempo limite, allora l'asta è dichiarata fallita.
Asta inversa	Un'asta inversa è un'asta nella quale si stabilisce una data di fine, entro la quale i venditori possono fare offerte al ribasso, e una soglia massima da cui partire. Quando sopraggiunge la data stabilita, la persona che ha offerto la cifra più bassa si aggiudica l'asta. Se non c'è stata nessuna offerta entro il tempo limite, allora l'asta è dichiarata fallita.
Offerta	Un'offerta è una somma di denaro che l'utente può inviare ad un'asta con lo scopo di aggiudicarsela.
Offerta di compratore	Un'offerta che viene inviata da un account compratore.
Offerta di venditore	Un'offerta che viene inviata da un account venditore.

Offerta a tempo fisso	Un'offerta che viene inviata da un account compratore ad un'asta a tempo fisso.
Offerta silenziosa	Un'offerta che viene inviata da un account compratore ad un'asta silenziosa.
Offerta inversa	Un'offerta che viene inviata da un account venditore ad un'asta inversa.

4.2.3 Dizionario delle associazioni

Associazione	Descrizione
Mittente	Associazione tra Account e Notifica. Quando viene inviata una notifica, essa conserva l'utente che ha effettuato l'azione che ha innescato l'invio. Una notifica ha un solo mittente, ma un mittente può essere responsabile di più notifiche.
Destinatario	Associazione tra Account e Notifica. Quando viene inviata una notifica, essa conserva l'utente al quale deve essere mandata per sapere a chi arrivare. Una notifica ha uno o più destinatari, e un destinatario può ricevere più notifiche.
Possiede	Associazione tra Account e Profilo. Quando viene creato un account, esso viene collegato ad un profilo. Un account è collegato ad un solo profilo, e un profilo è collegato al più a due account, ossia uno di tipo compratore e uno di tipo venditore.
È associata a	Associazione tra Notifica e Asta. Quando viene inviata una notifica, essa conserva l'asta relativa all'evento che ha causato l'invio, così che l'utente possa raggiungere velocemente l'asta attraverso la notifica. Una notifica è associata ad una sola asta, ma un'asta è associata a più notifiche.
Possiede	Associazione tra Venditore e Asta di venditore. Un venditore può creare un'asta per la vendita di un bene o servizio e ne diventa il possessore e gestore. Un'asta è quindi associata ad un solo venditore, ma un venditore può creare più aste.
È collegato a	Associazione tra Venditore e Offerta di venditore. Quando viene inviata un'offerta dal venditore, essa conserva il suo offerente così che si possa risalire ad esso nell'elenco delle offerte dell'asta. Un'offerta è collegata ad un solo venditore, ma un venditore può inviare più offerte.
Possiede	Associazione tra Compratore e Asta di compratore. Un compratore può creare un'asta per l'acquisto di un bene o servizio e ne diventa il possessore e gestore. Un'asta è quindi associata ad un solo compratore, ma un compratore può creare più aste.
È collegato a	Associazione tra Compratore e Offerta di compratore. Quando viene inviata un'offerta dal compratore, essa conserva il suo offerente così che si possa risalire ad esso nell'elenco delle offerte dell'asta. Un'offerta è collegata ad un solo compratore, ma un compratore può inviare più offerte.
Si riferisce a	Associazione tra Offerta a tempo fisso e Asta a tempo fisso. Ogni offerta è collegata alla relativa asta. In particolare, ogni offerta è relativa ad una sola asta e ogni asta è collegata a più offerte.
Si riferisce a	Associazione tra Offerta silenziosa e Asta silenziosa. Ogni offerta è collegata alla relativa asta. In particolare, ogni offerta è relativa ad una sola asta e ogni asta è collegata a più offerte.

Si riferisce a	Associazione tra Offerta inversa e Asta inversa. Ogni offerta è collegata alla relativa asta. In particolare, ogni offerta è relativa ad una sola asta e ogni asta è collegata a più offerte.
----------------	---

4.3 Class Diagram per casi d'uso

In questa sezione verranno mostrati i diagrammi secondo l'euristica "Three-Object-Type" per ogni caso d'uso. Le classi possono essere di tipo "Boundary", "Control" e "Entity".

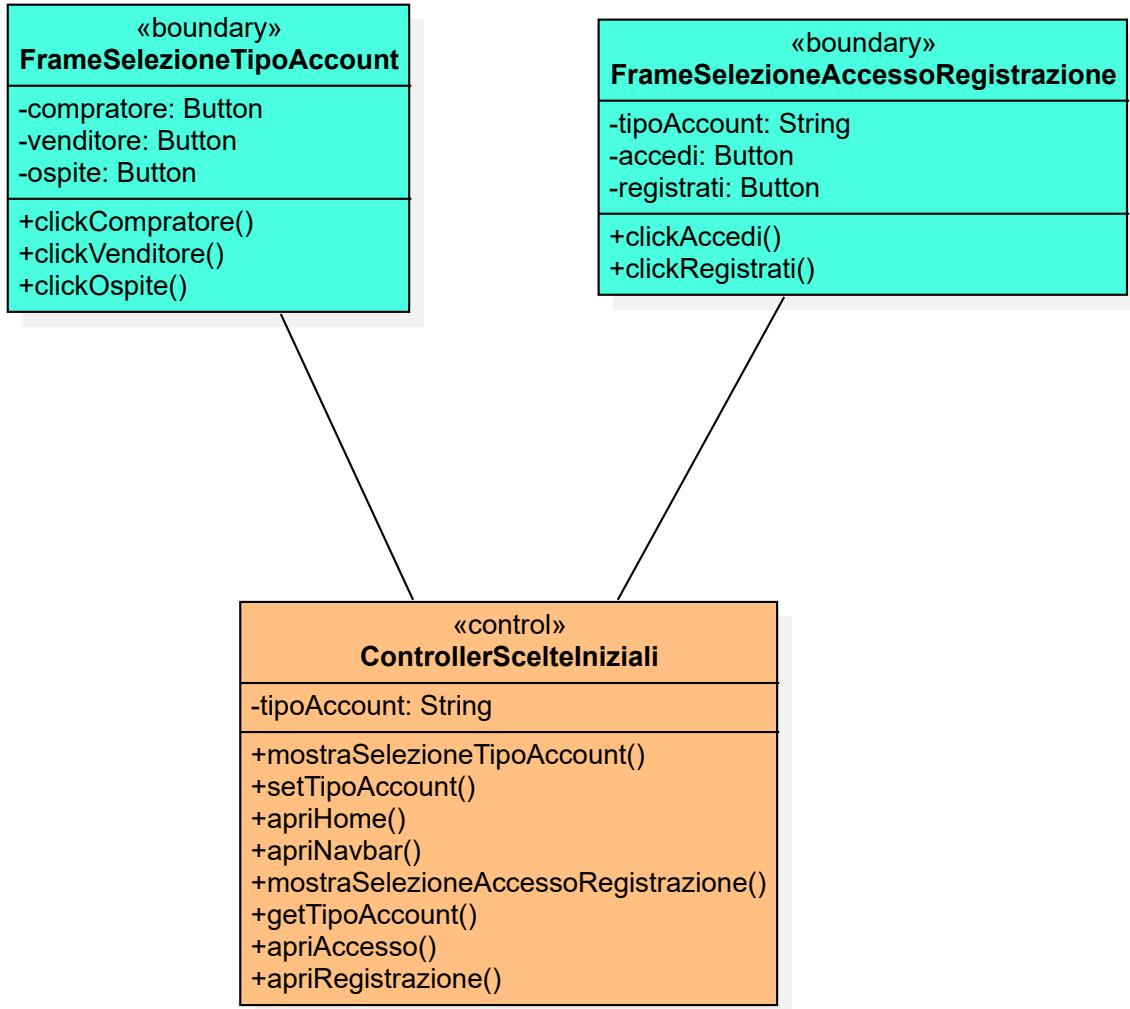


Figura 4.2: Scelta del tipo di account

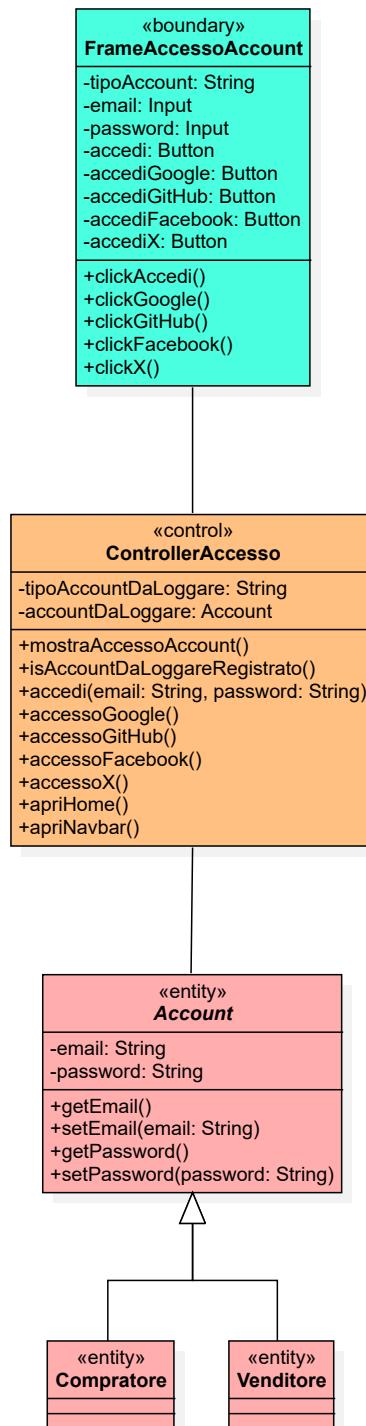


Figura 4.3: Accesso

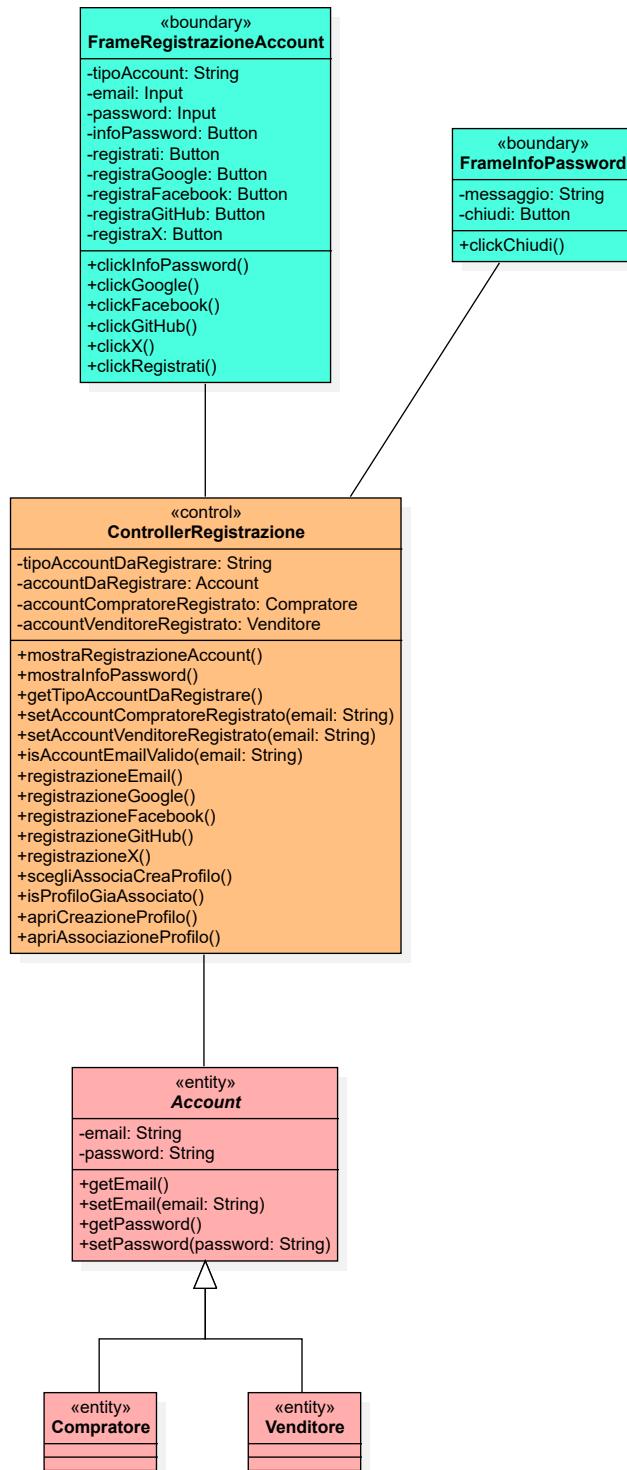


Figura 4.4: Registrazione

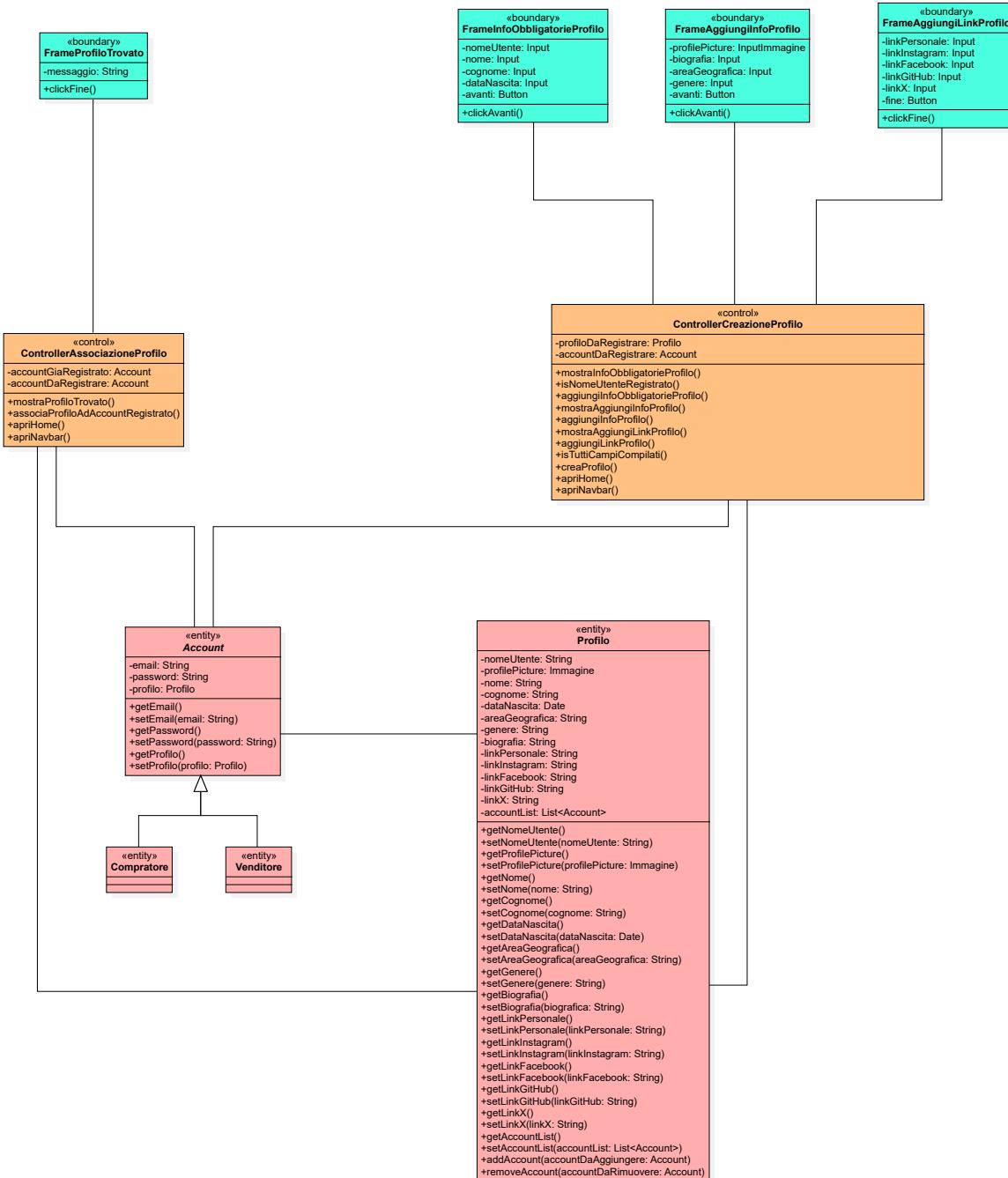


Figura 4.5: Creazione del profilo

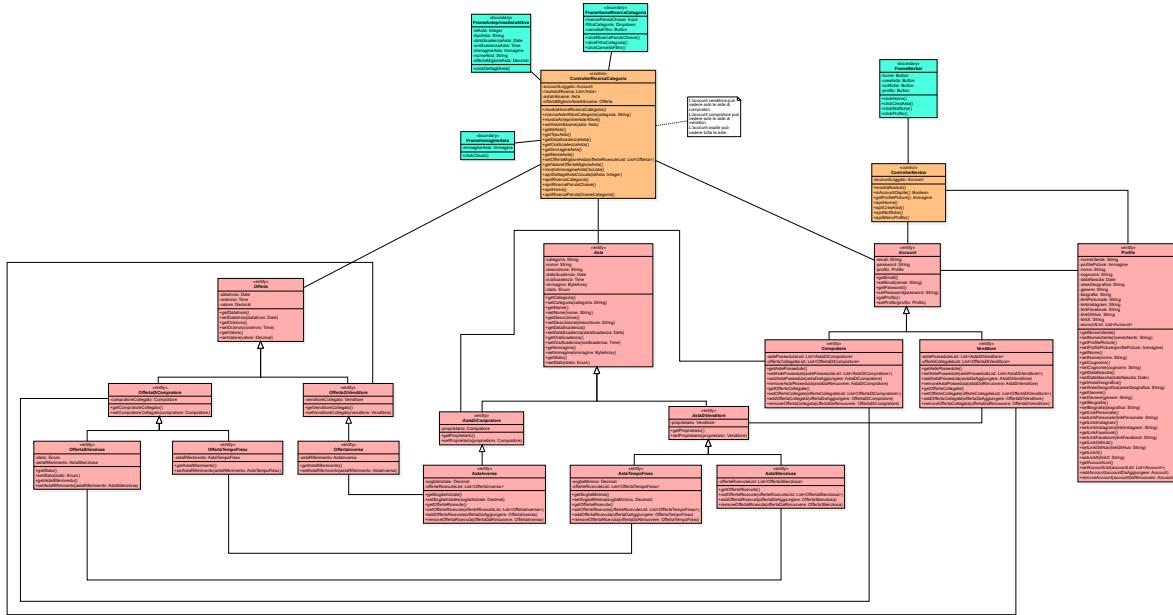


Figura 4.6: Ricerca per categoria

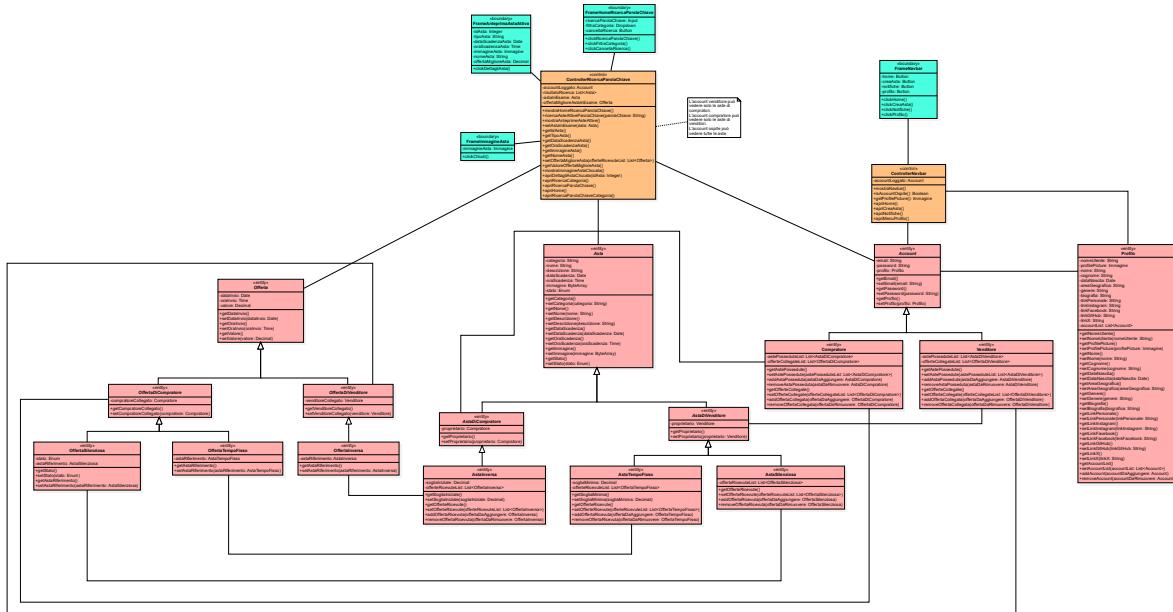


Figura 4.7: Ricerca per parola chiave

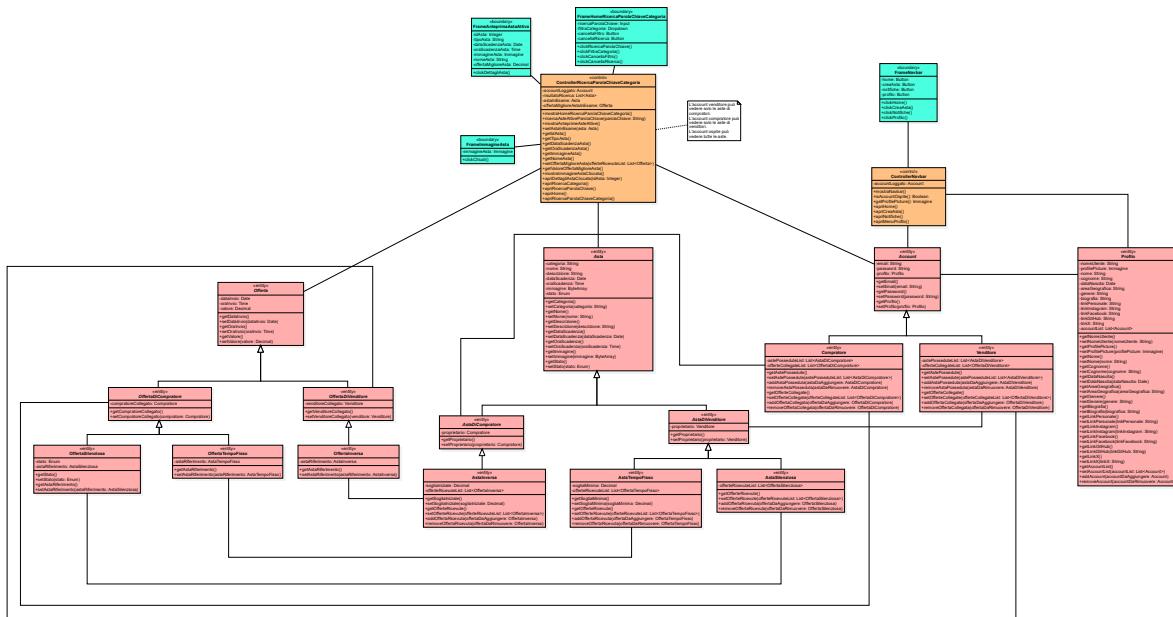


Figura 4.8: Ricerca per parola chiave e categoria

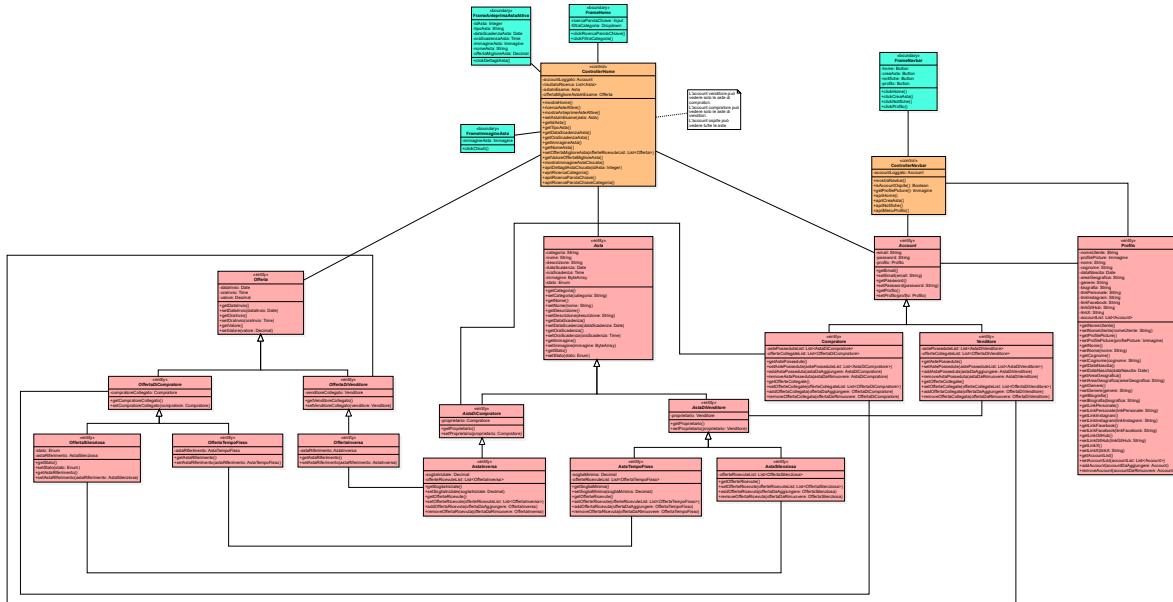


Figura 4.9: Visualizza aste attive

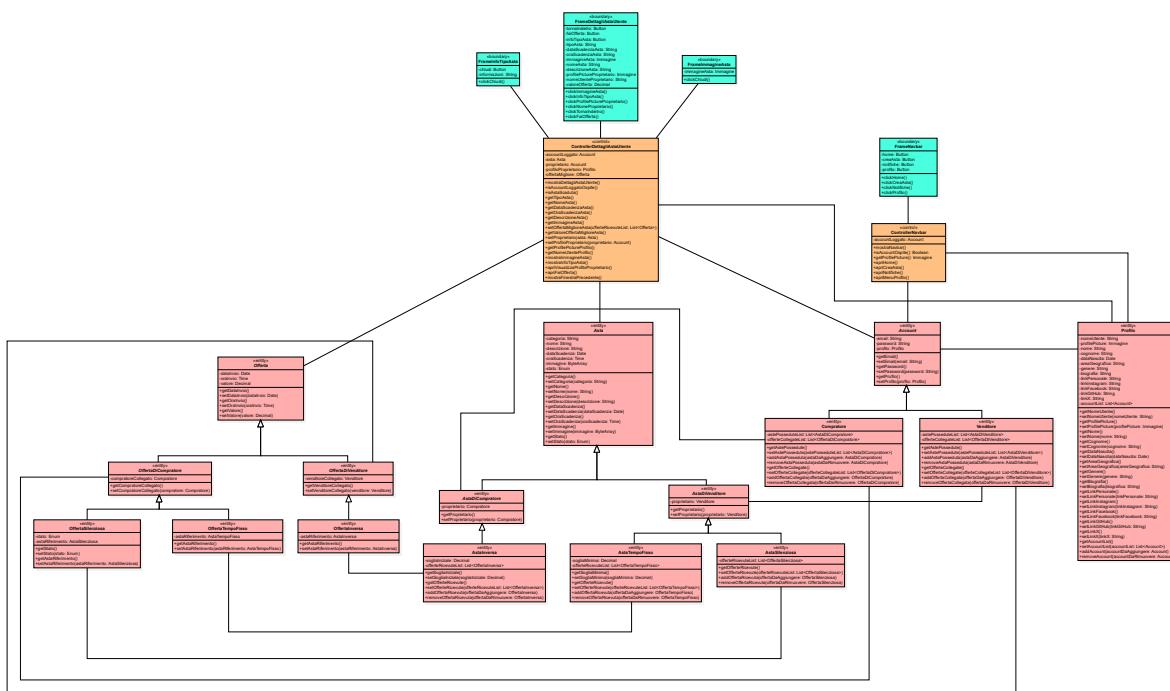


Figura 4.10: Visualizza dettagli asta di un altro utente

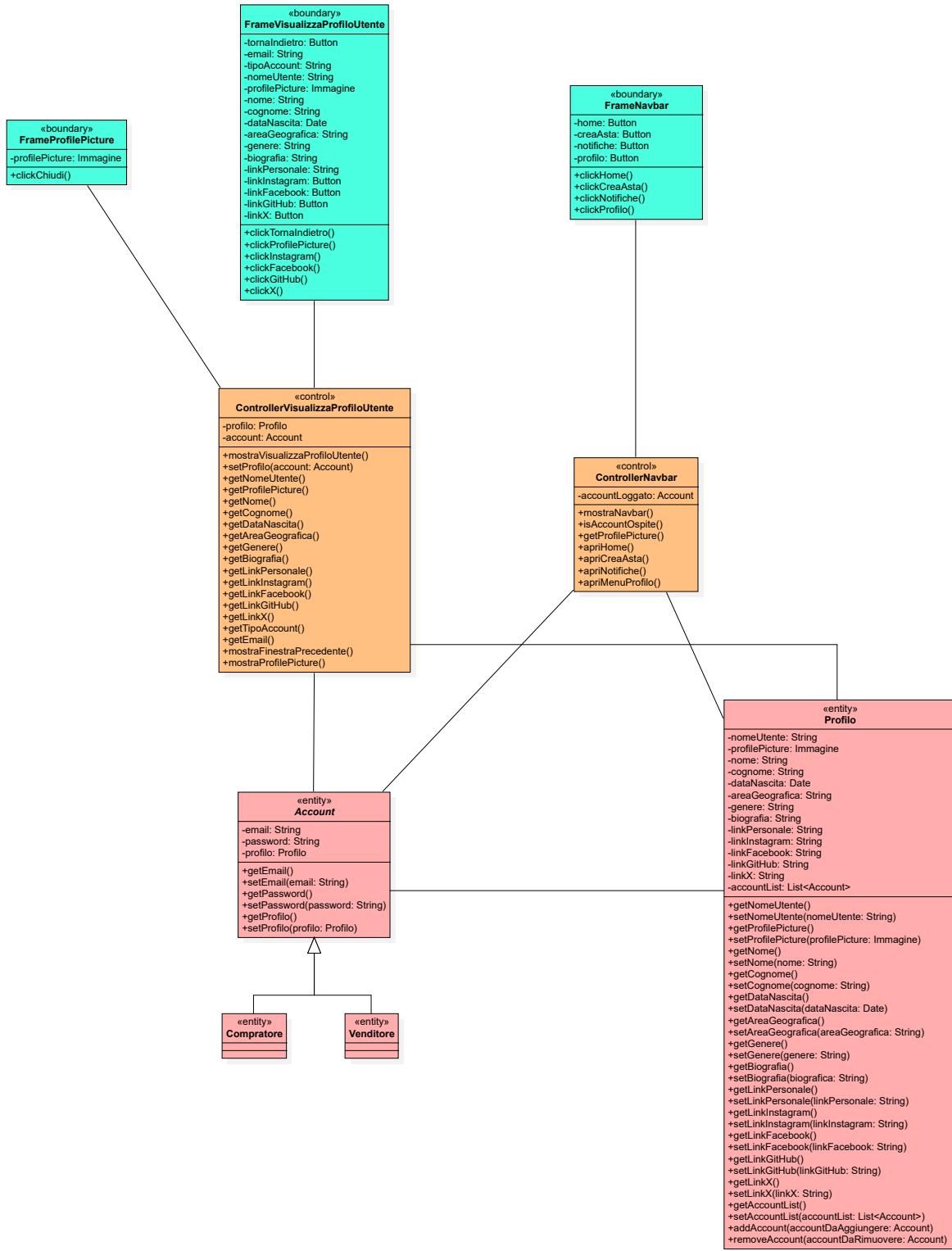


Figura 4.11: Visualizza profilo utente

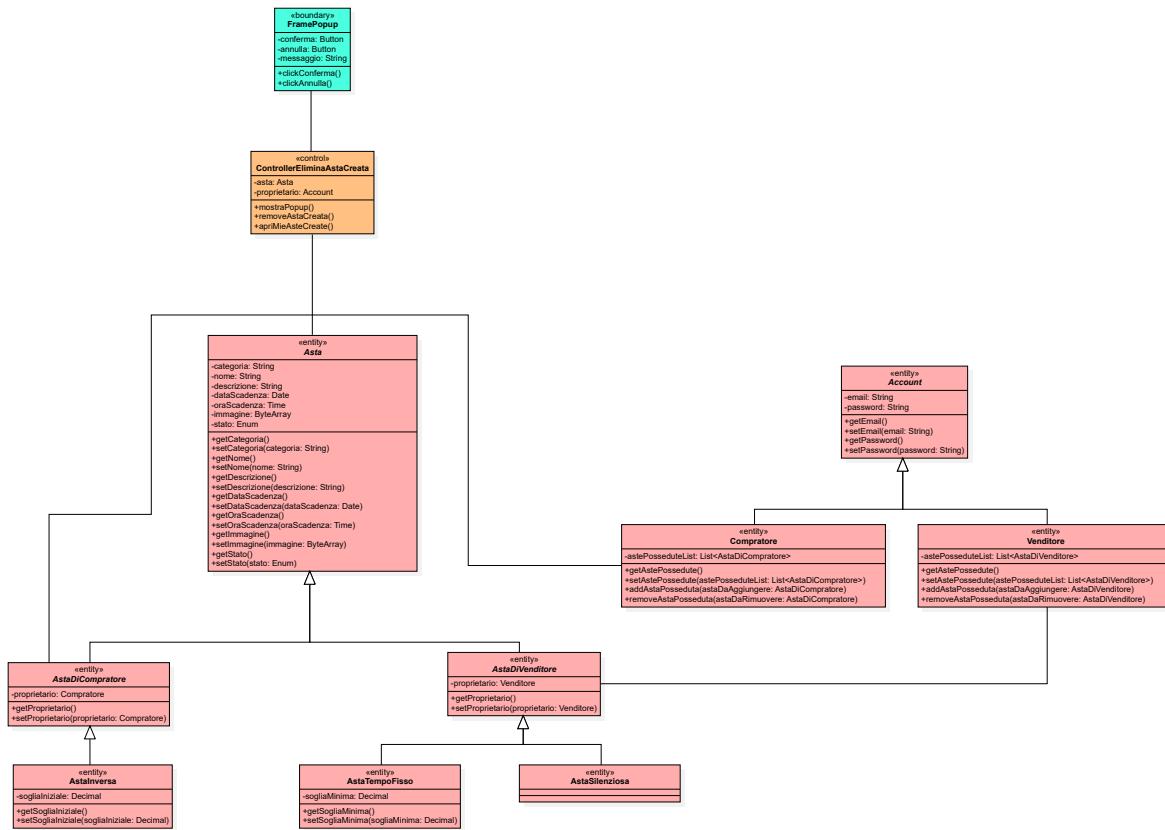


Figura 4.12: Elimina asta creata

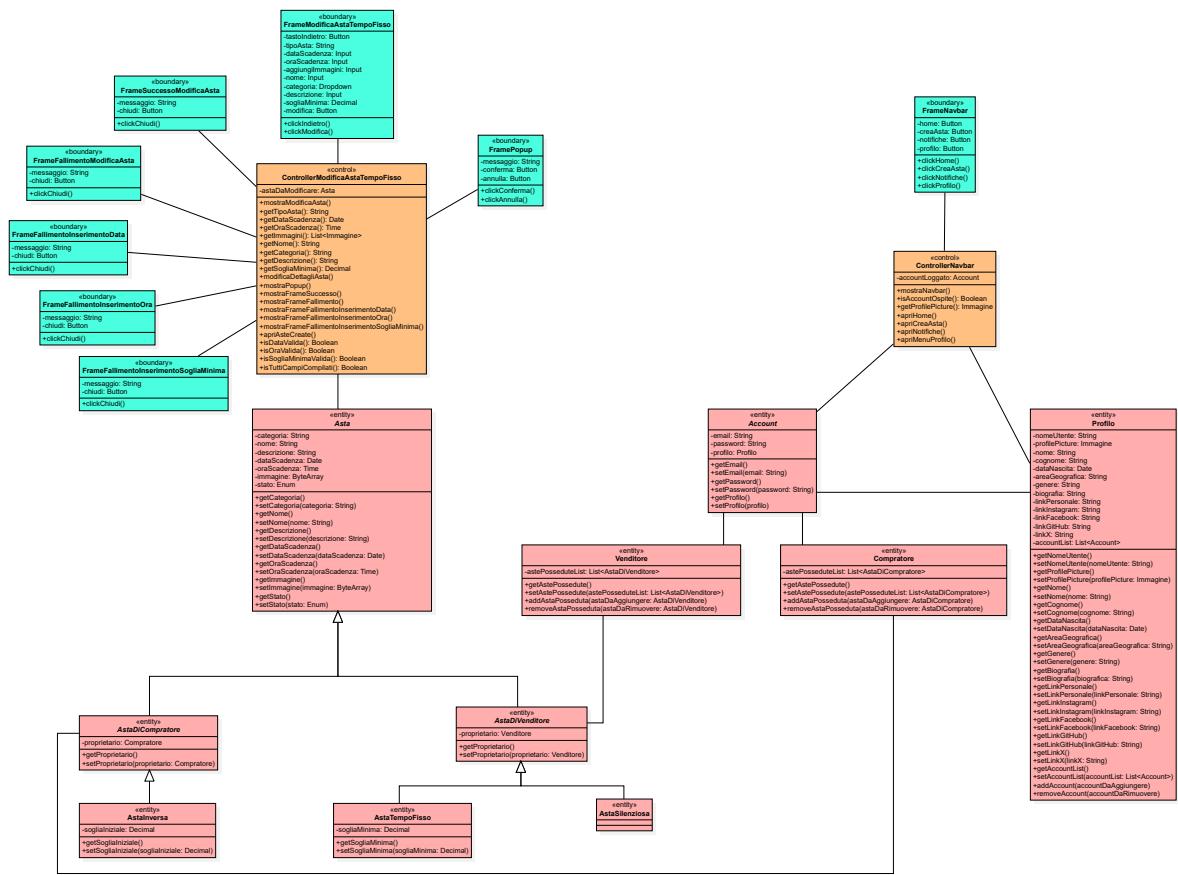


Figura 4.13: Modifica asta a tempo fisso

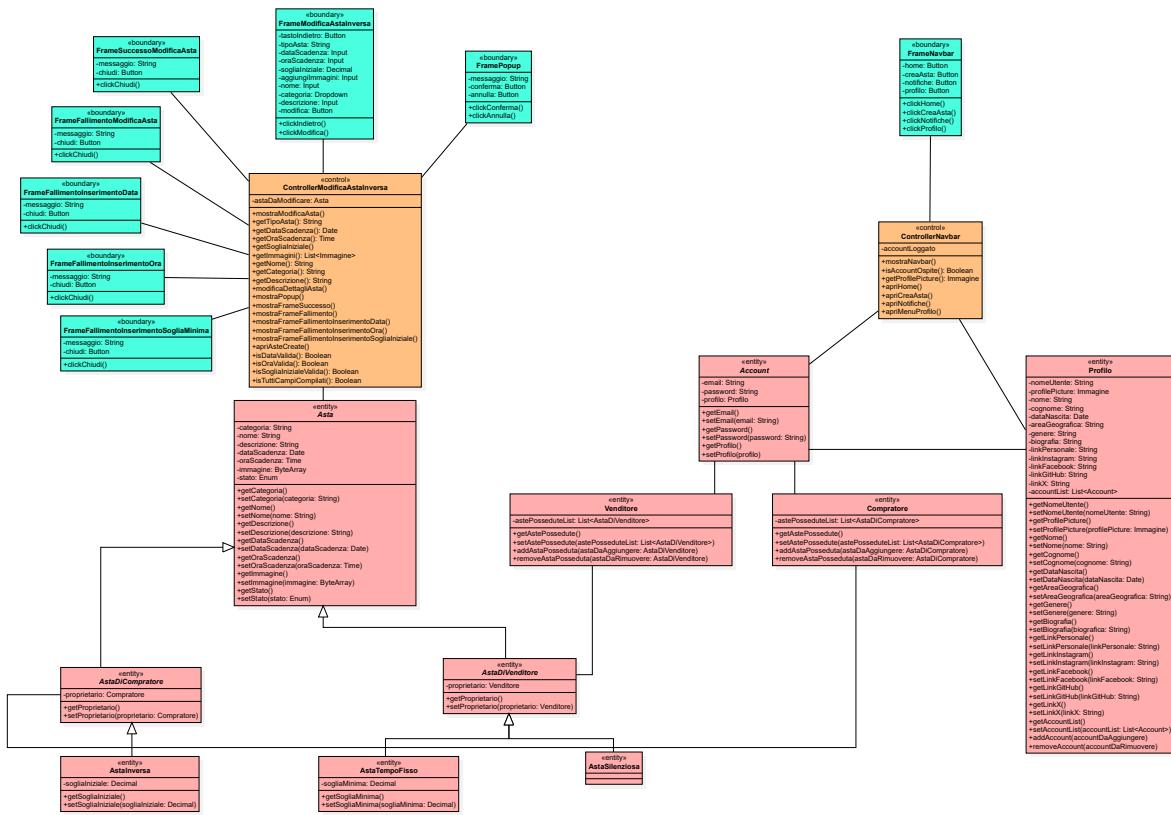


Figura 4.14: Modifica asta inversa

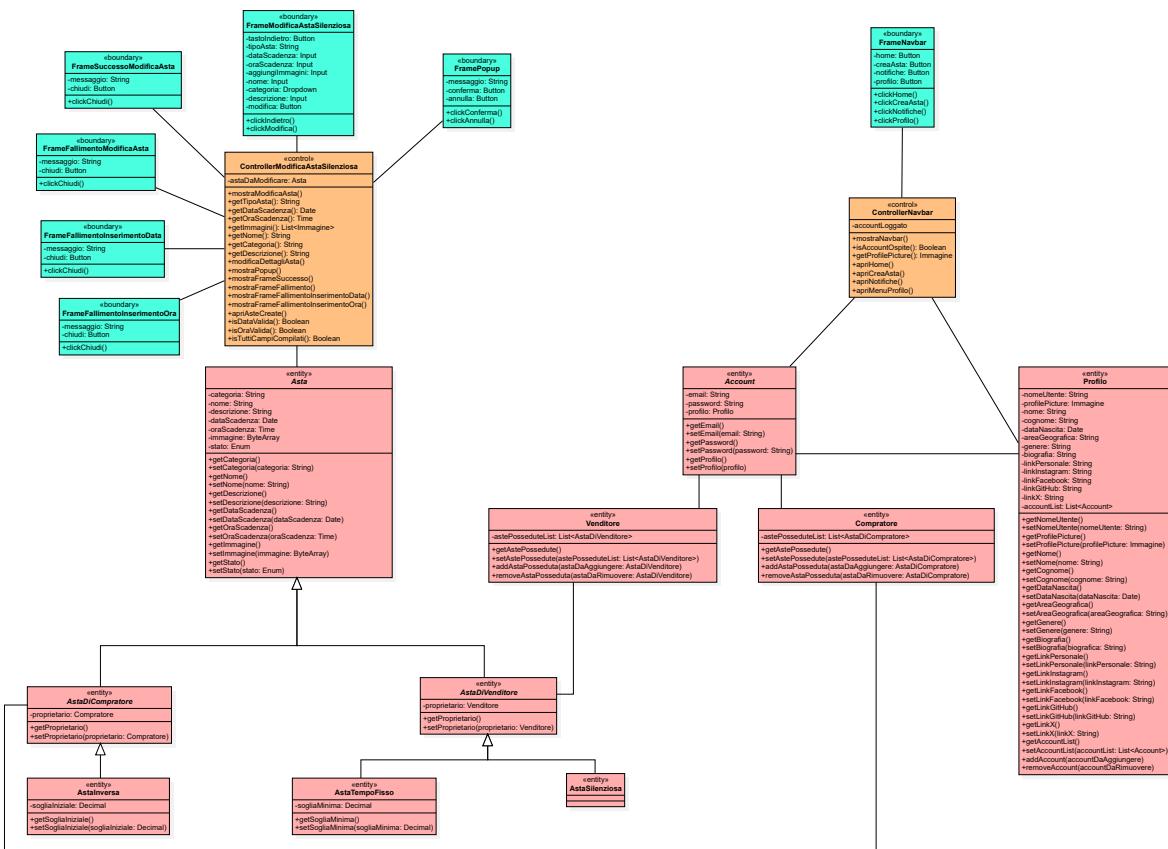


Figura 4.15: Modifica asta silenziosa

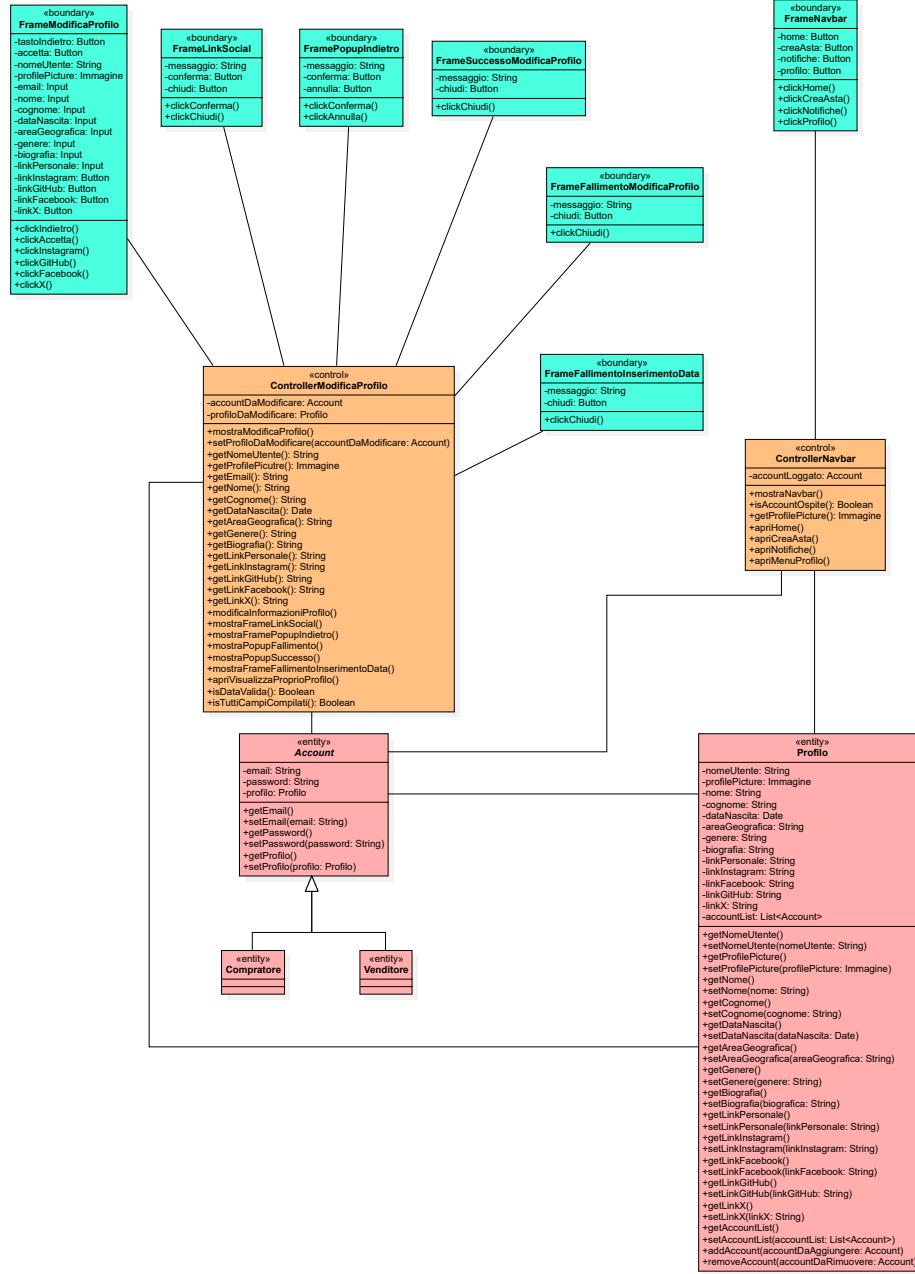


Figura 4.16: Modifica profilo

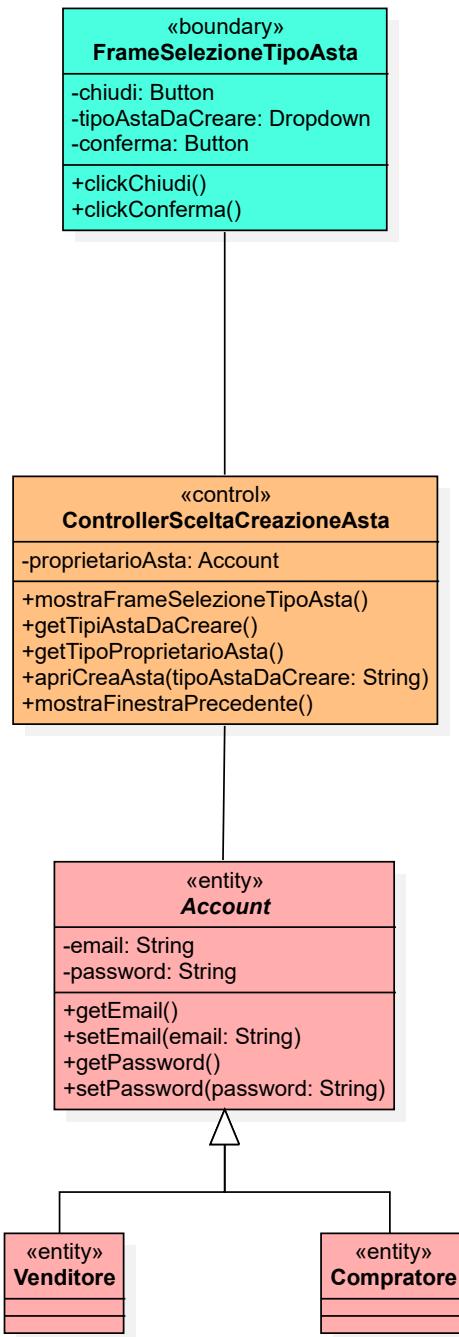


Figura 4.17: Scelta tipo di asta da creare

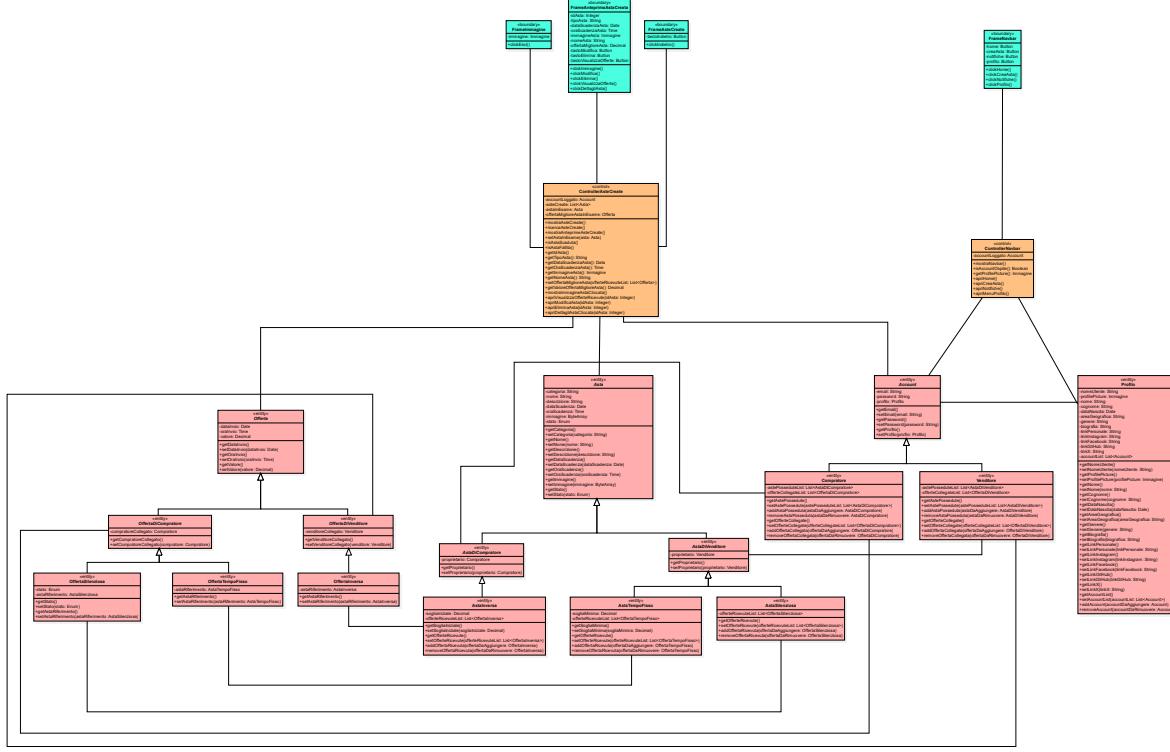


Figura 4.18: Visualizza aste create

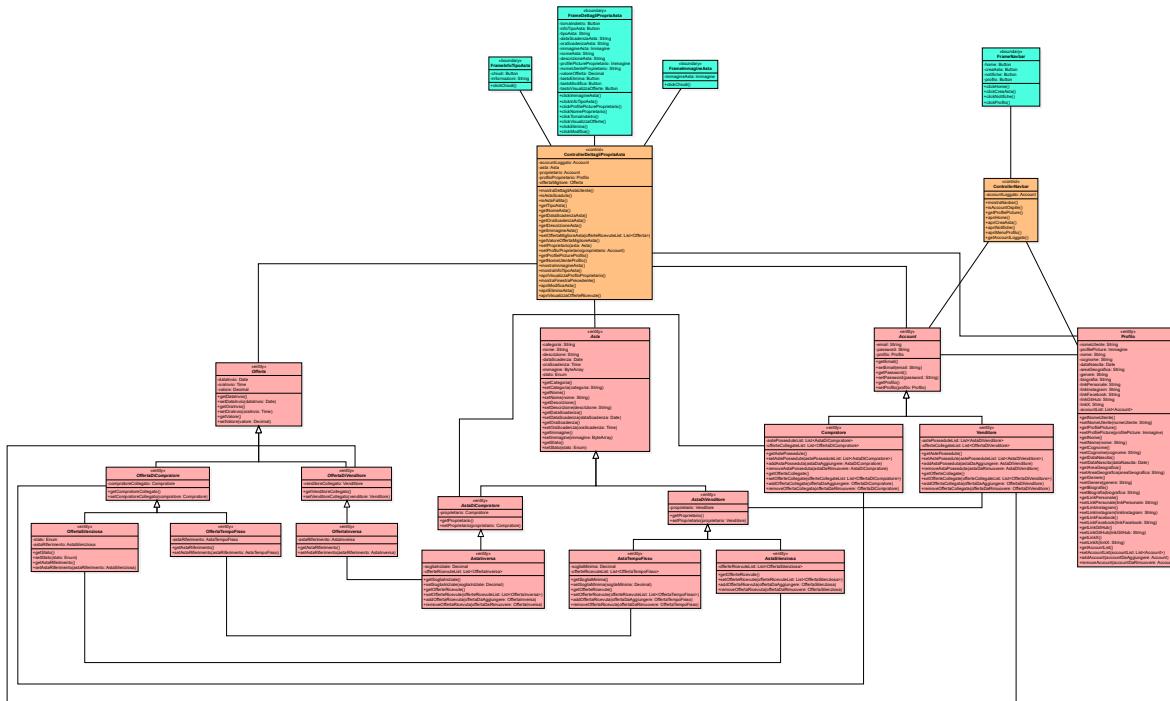


Figura 4.19: Visualizza dettagli della propria asta

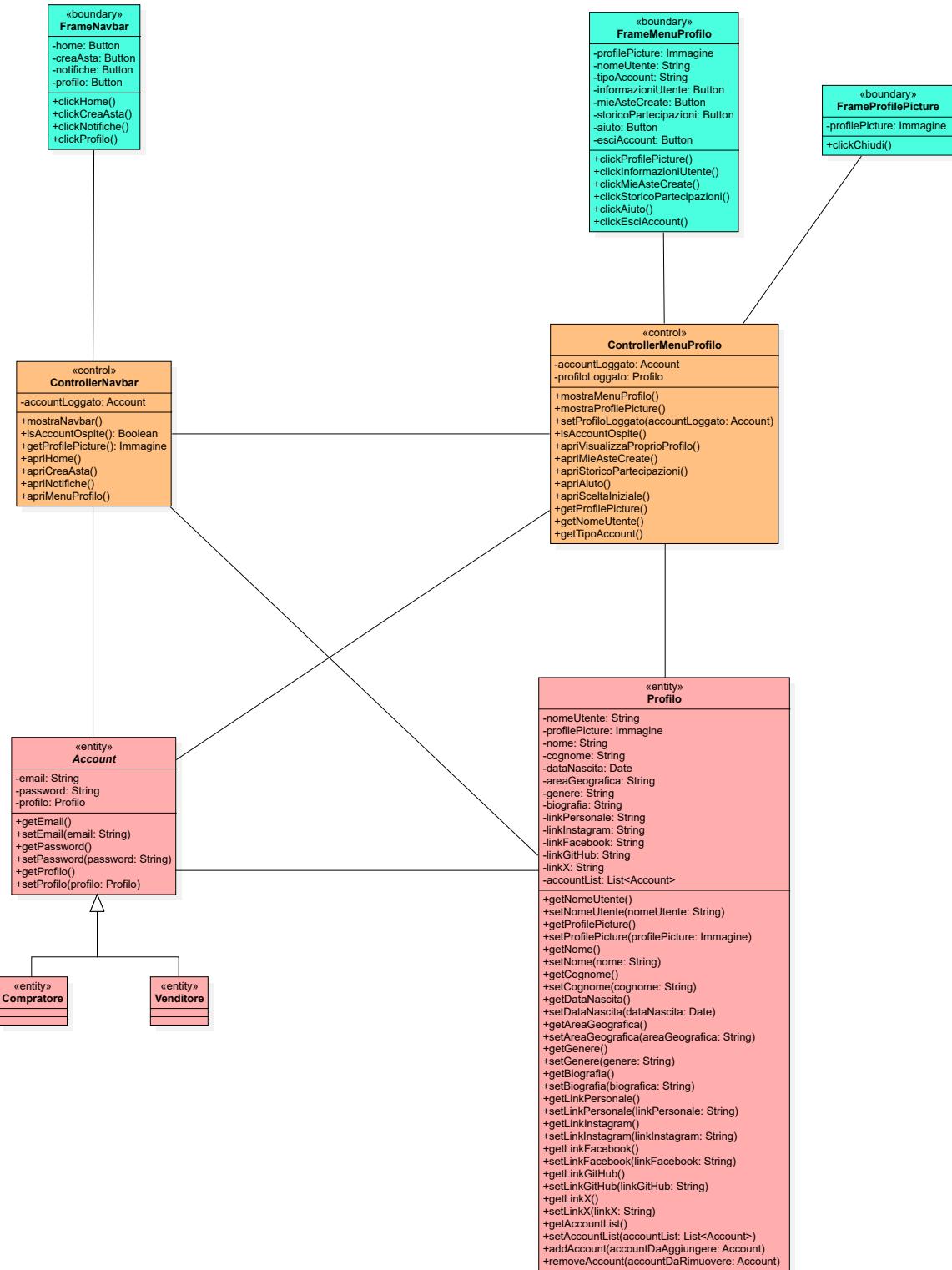


Figura 4.20: Visualizza menu profilo

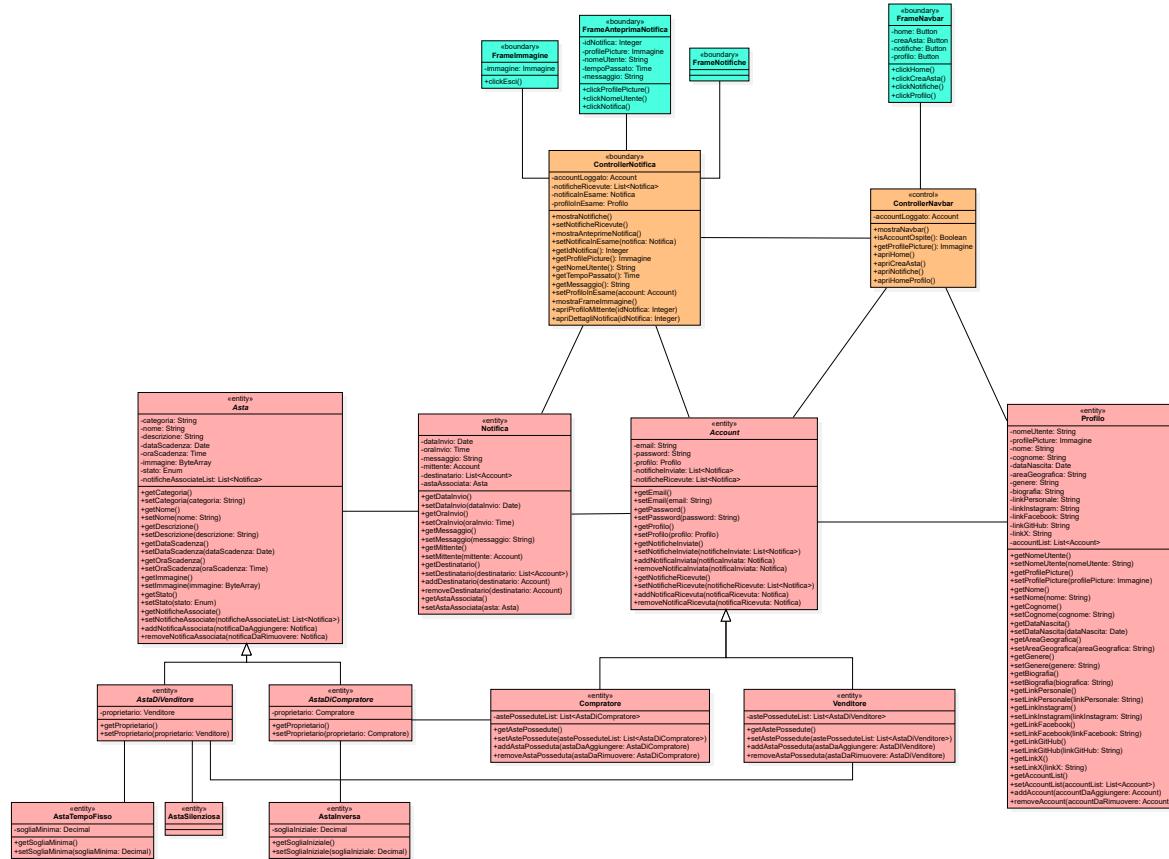


Figura 4.21: Visualizza notifiche

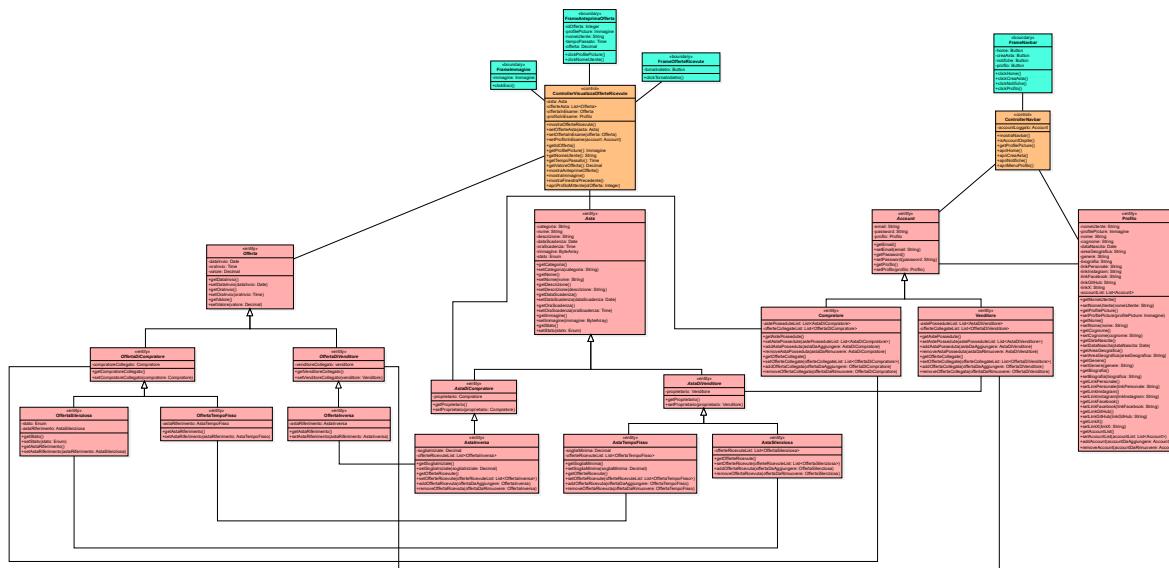


Figura 4.22: Visualizza offerte ricevute

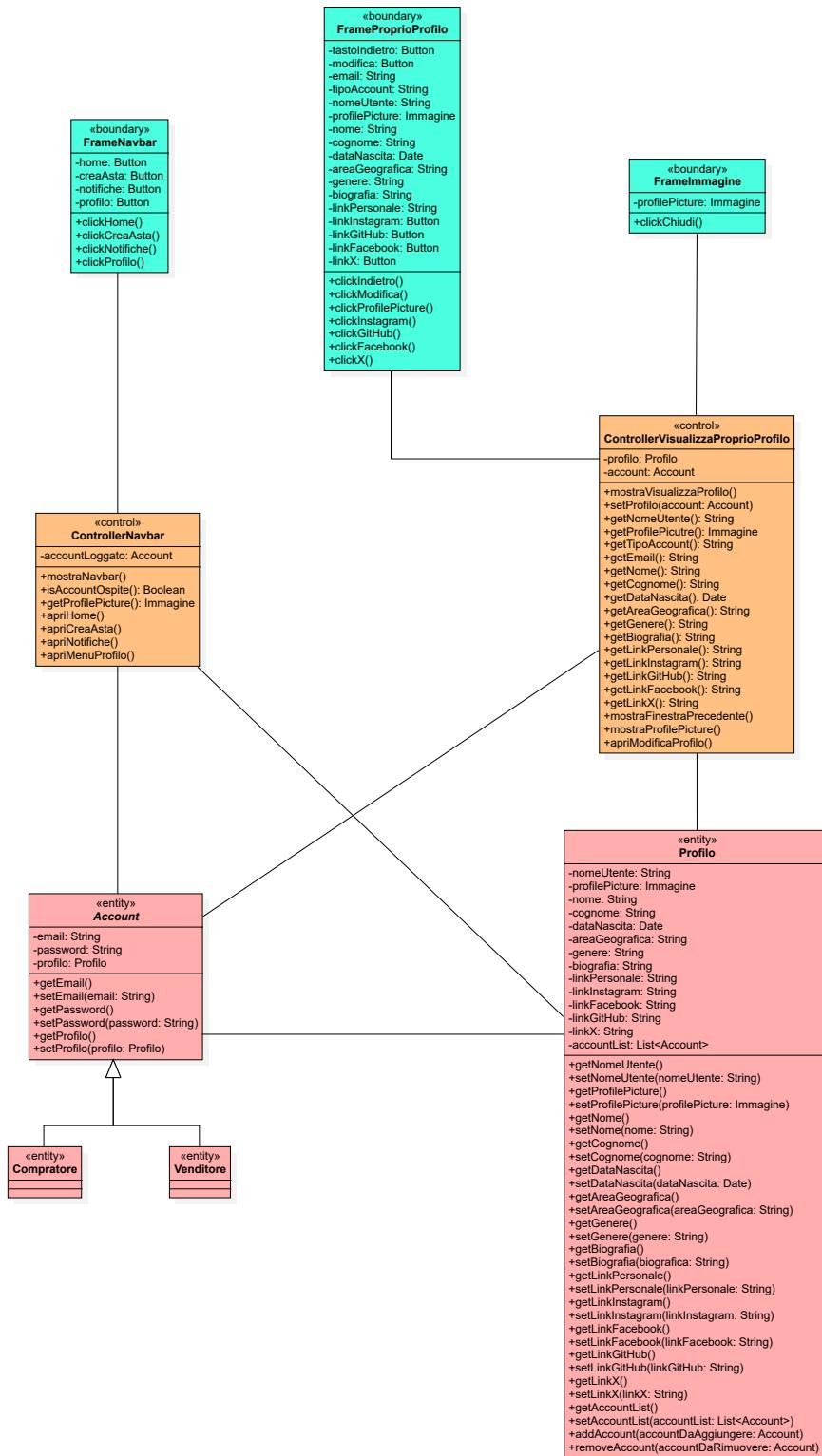


Figura 4.23: Visualizza profilo personale

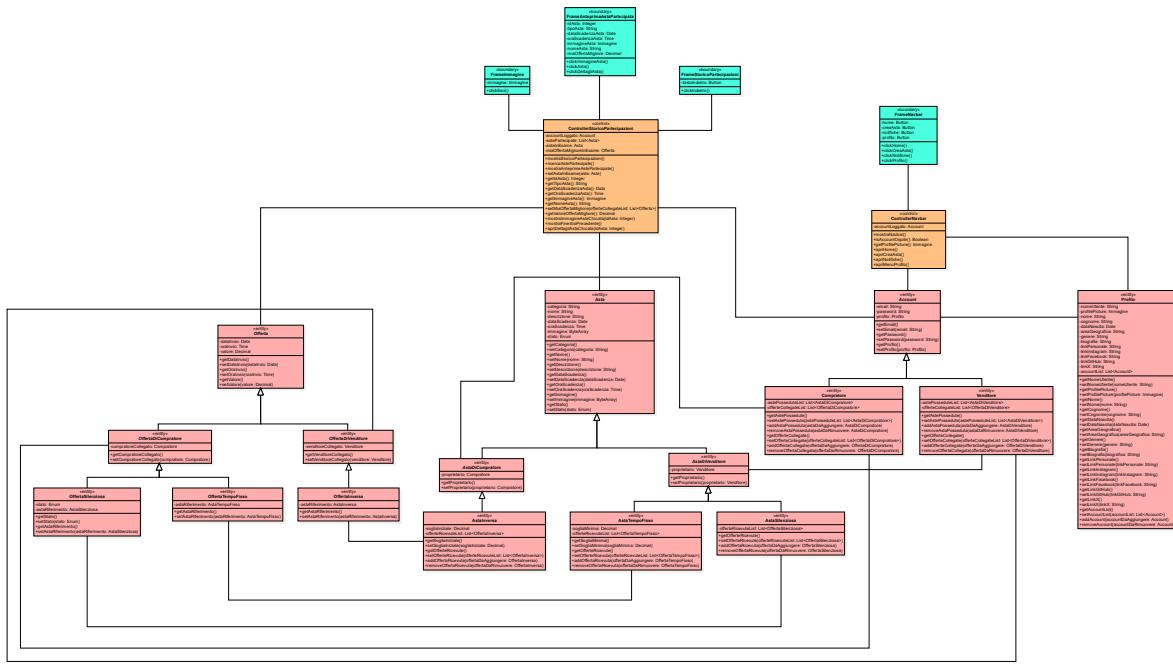


Figura 4.24: Visualizza aste partecipate

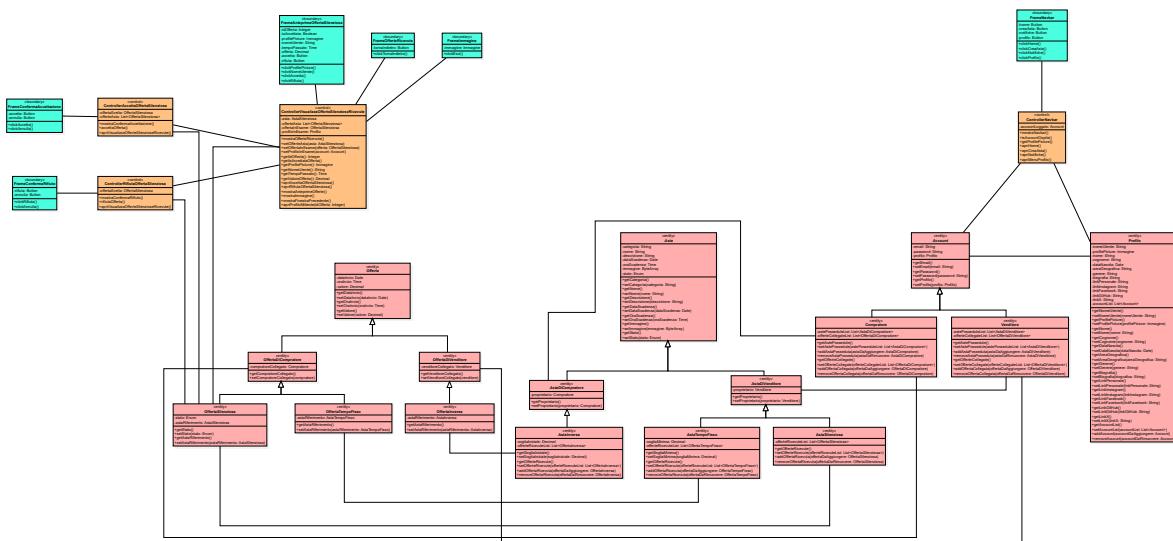


Figura 4.25: Accetta o rifiuta un'offerta per un'asta silenziosa

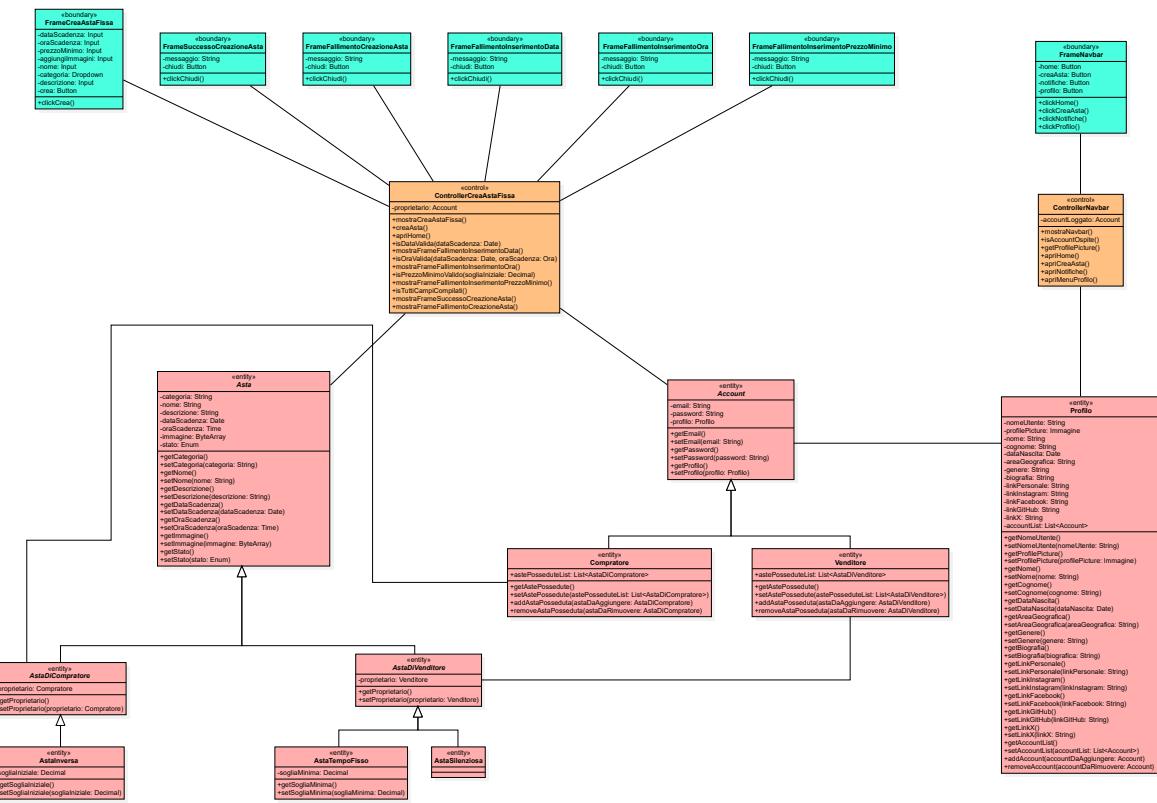


Figura 4.26: Crea un'asta a tempo fisso

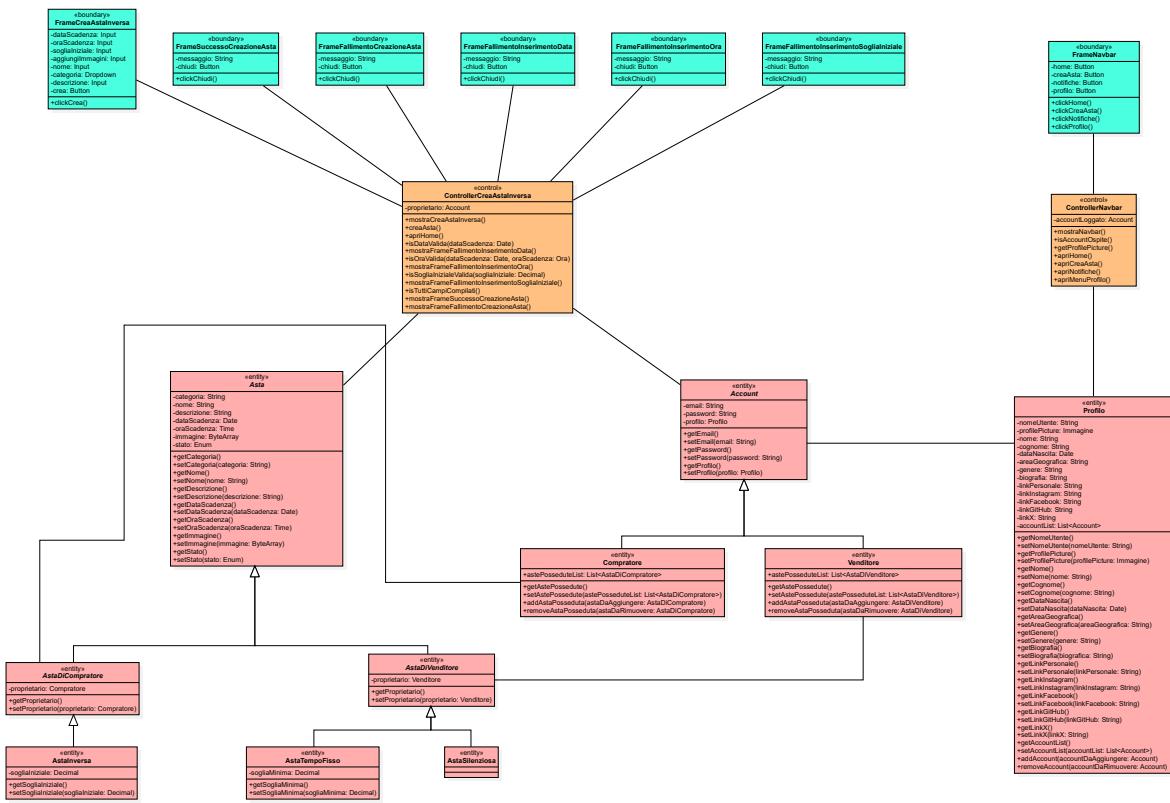


Figura 4.27: Crea un'asta inversa

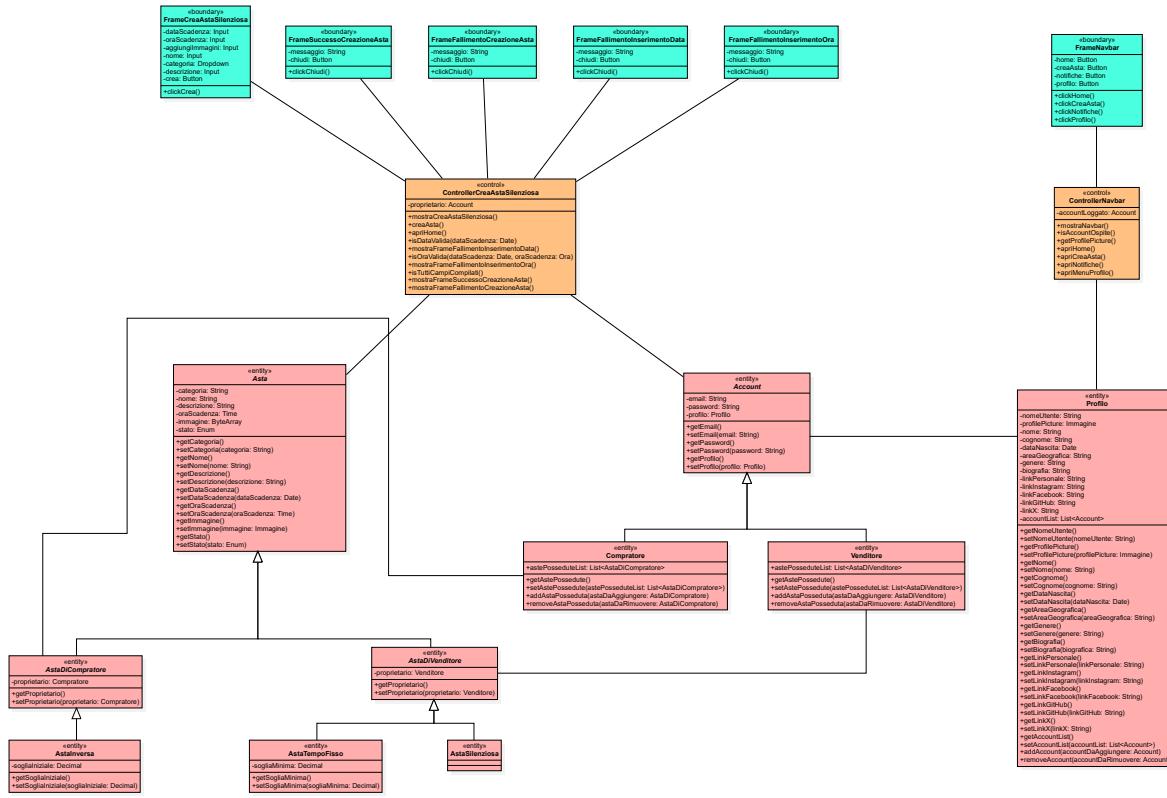


Figura 4.28: Crea un'asta silenziosa

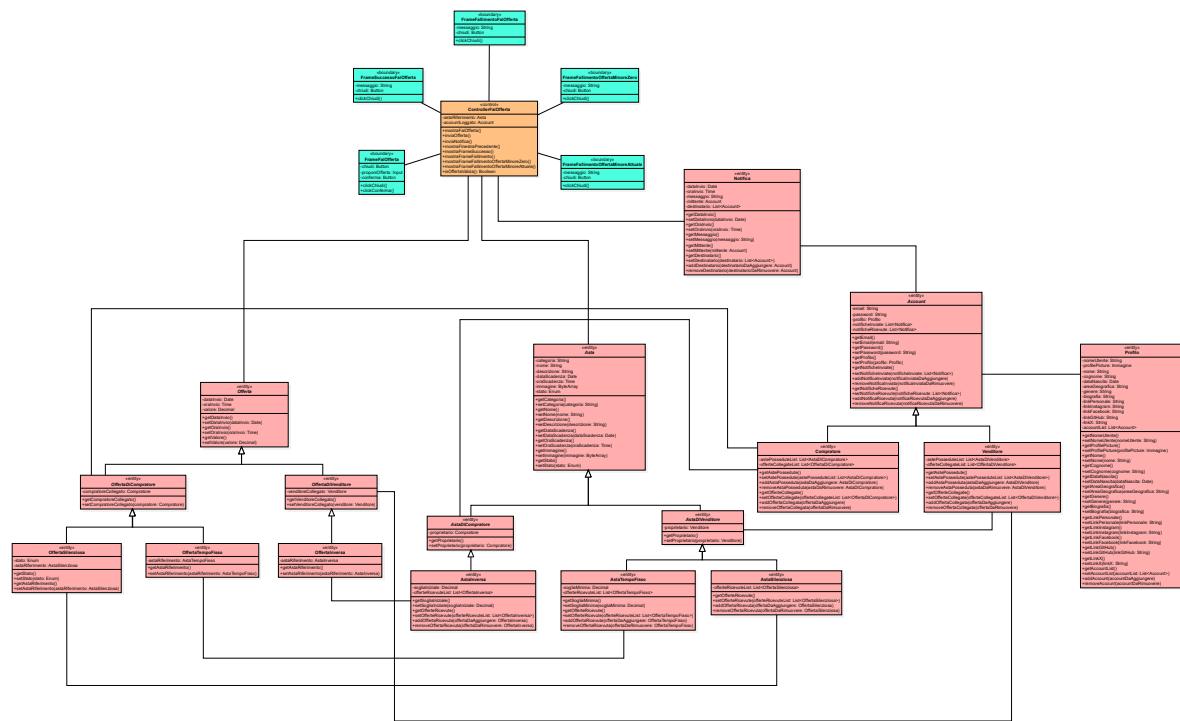


Figura 4.29: Fai un'offerta

4.4 Sequence Diagram

Il diagramma di sequenza rappresenta i processi e gli oggetti coinvolti e la sequenza dei messaggi scambiati necessari per adempiere ad una funzionalità.

Le interazioni sono disposte lungo un'asse temporale per dare un'idea dell'ordine di avvicendamento dei messaggi.

4.4.1 I caso d'uso: Crea un'asta inversa

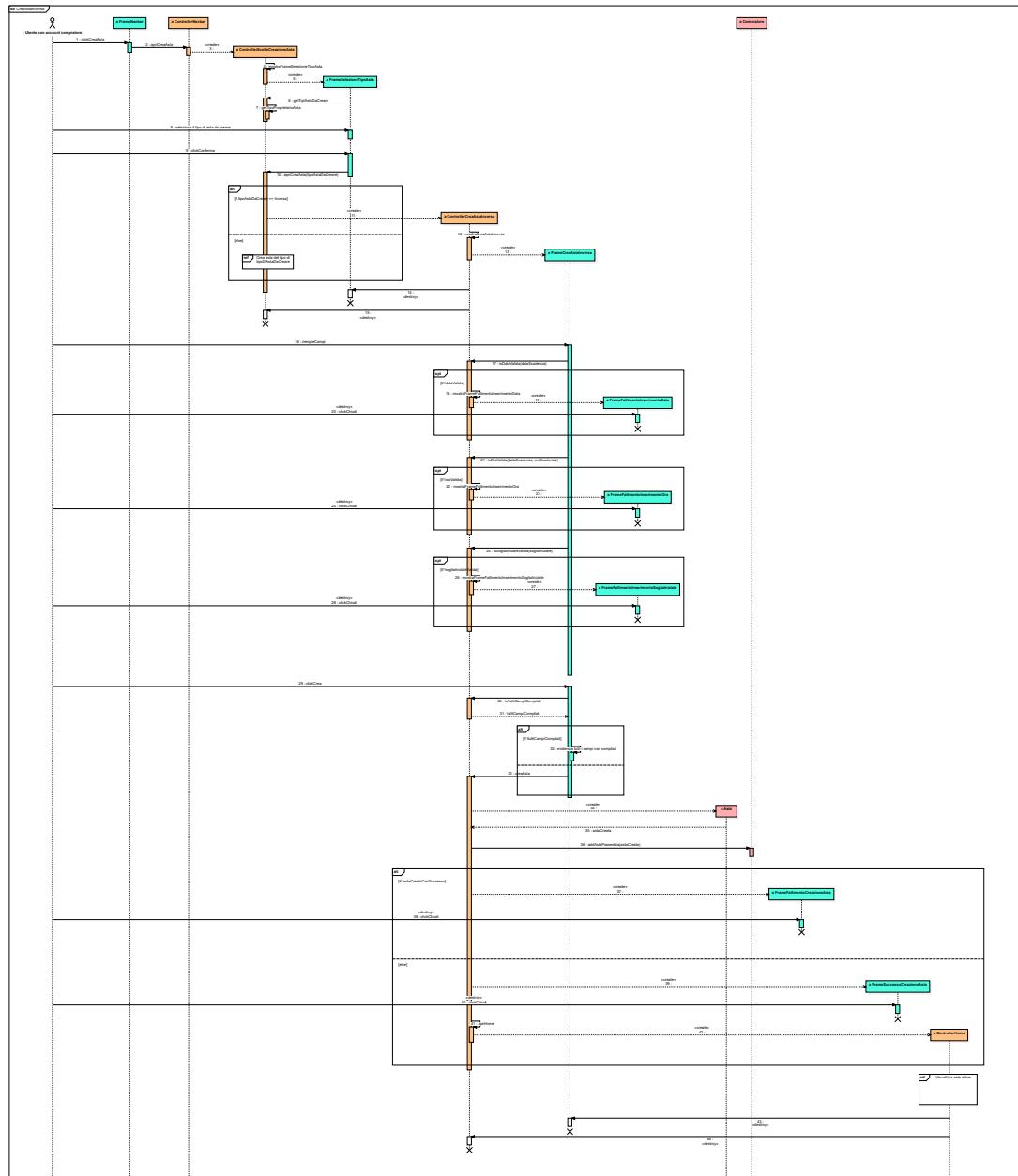


Figura 4.30: Sequence Diagram creazione asta inversa

4.4.2 II caso d'uso: Accetta un'offerta ricevuta all'asta silenziosa

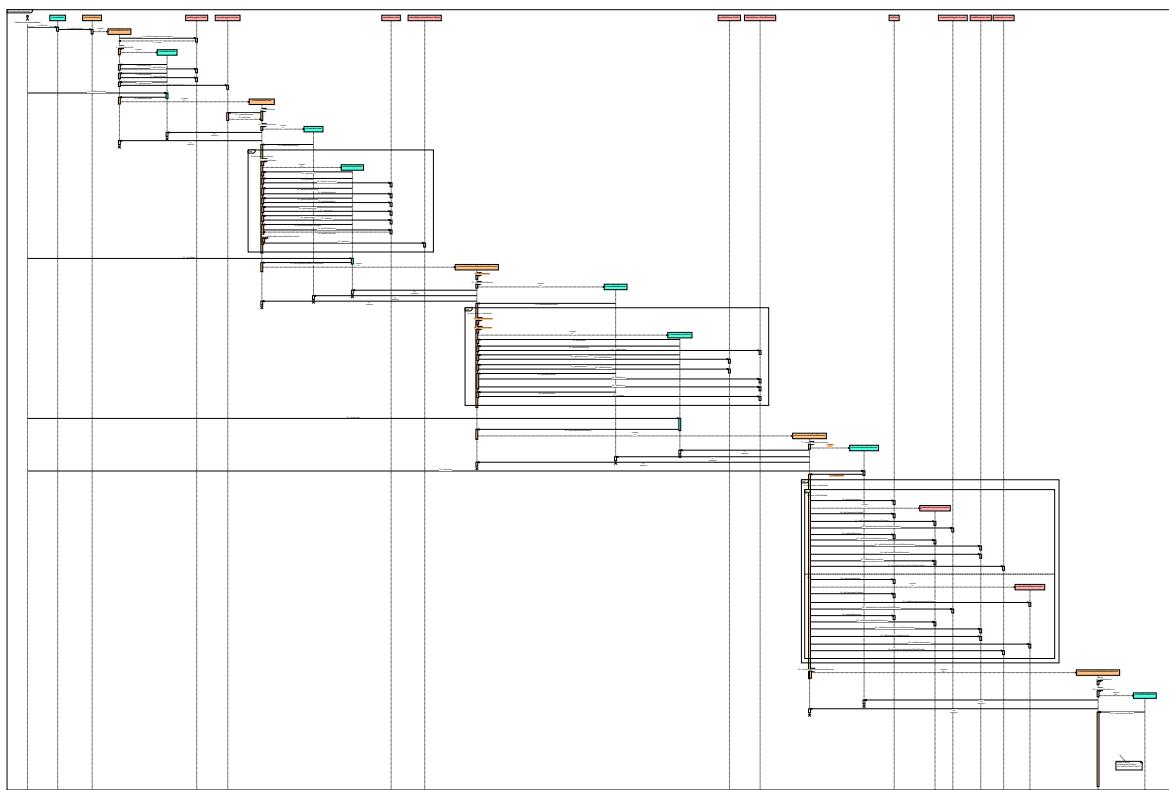


Figura 4.31: Sequence Diagram accetta offerta per un'asta silenziosa

4.5 Statechart Diagram

Il diagramma degli stati consente di rappresentare gli aspetti dinamici di un sistema, in particolare per l'interfaccia utente.

Presenta quindi una serie di stati che il sistema attraversa (come una macchina a stati finiti), le azioni che comportano una transizione da uno stato all'altro, e le condizioni da rispettare affinché la transizione possa avvenire.

4.5.1 I caso d'uso: Crea un'asta inversa

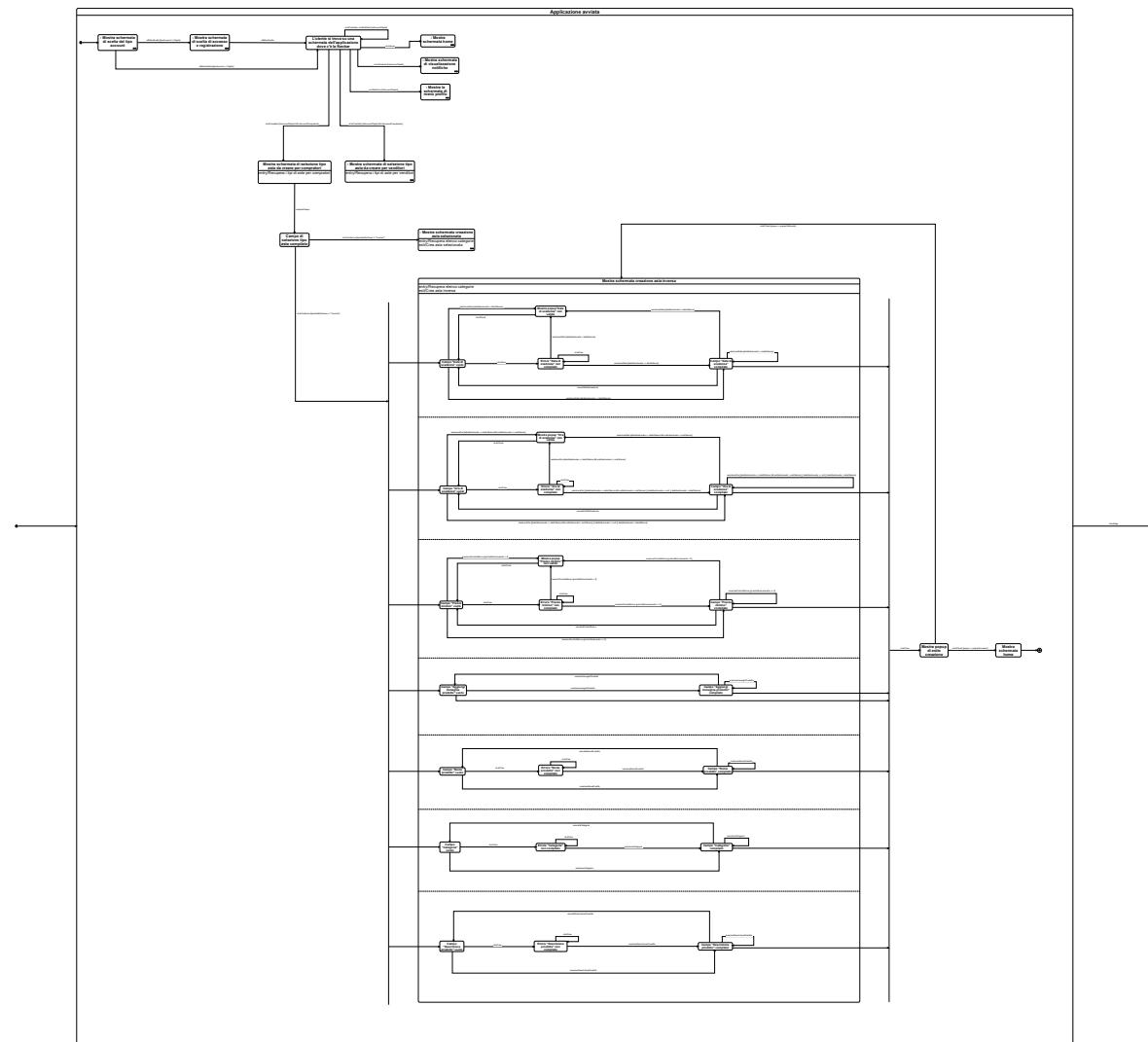


Figura 4.32: Statechart Diagram creazione asta inversa

4.5.2 II caso d'uso: Accetta un'offerta ricevuta all'asta silenziosa

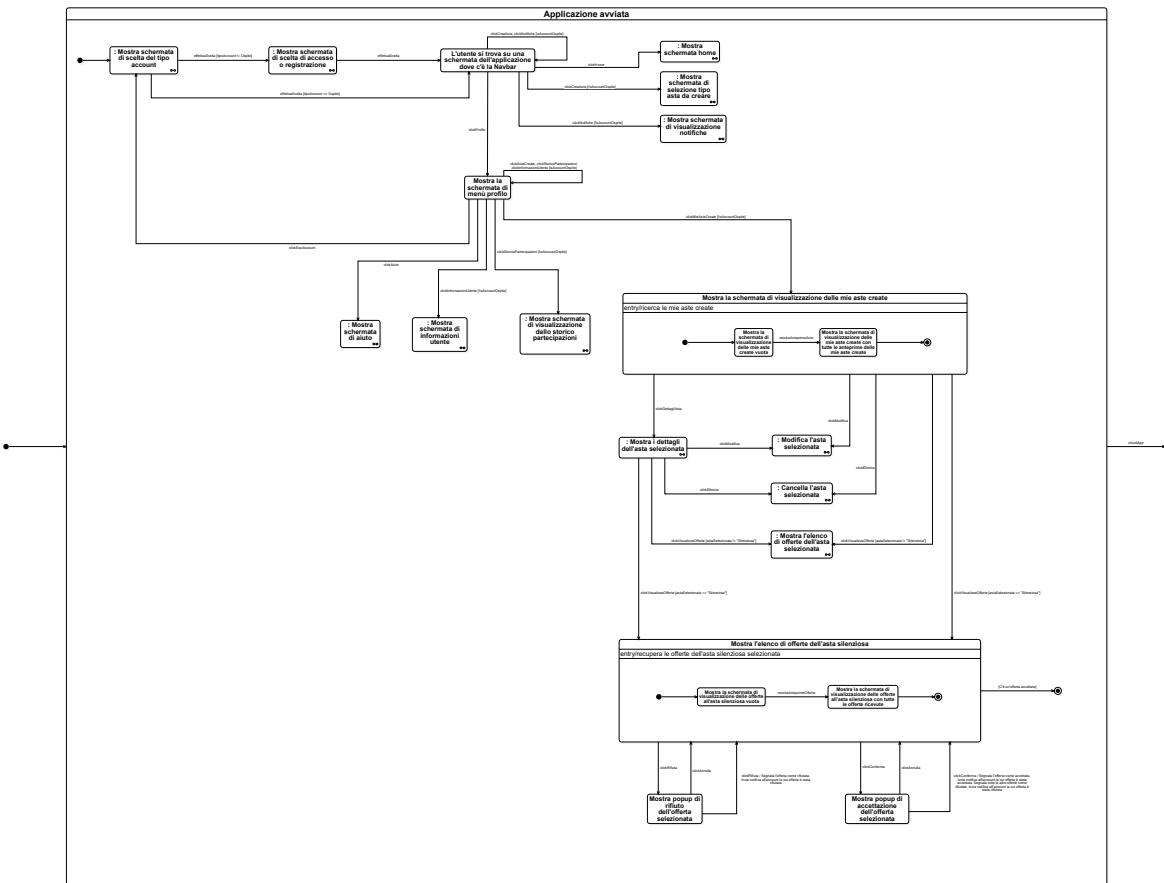


Figura 4.33: Statechart Diagram accetta offerta per un'asta silenziosa

4.5.3 III caso d'uso: Visualizza i dettagli di un'asta

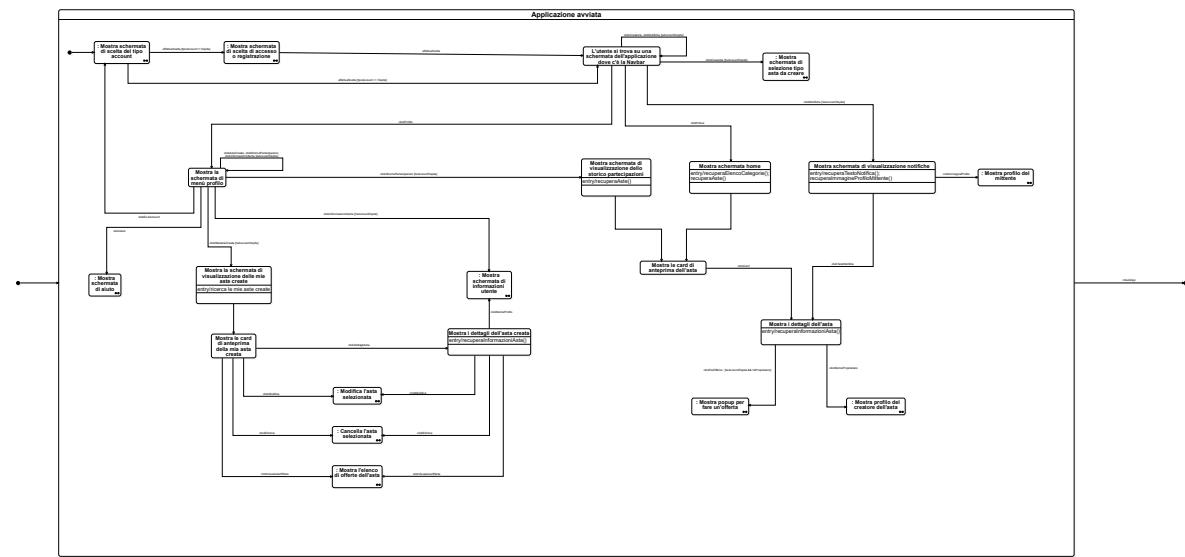


Figura 4.34: Statechart Diagram visualizza dettagli di asta

4.5.4 IV caso d'uso: Modificare il proprio profilo

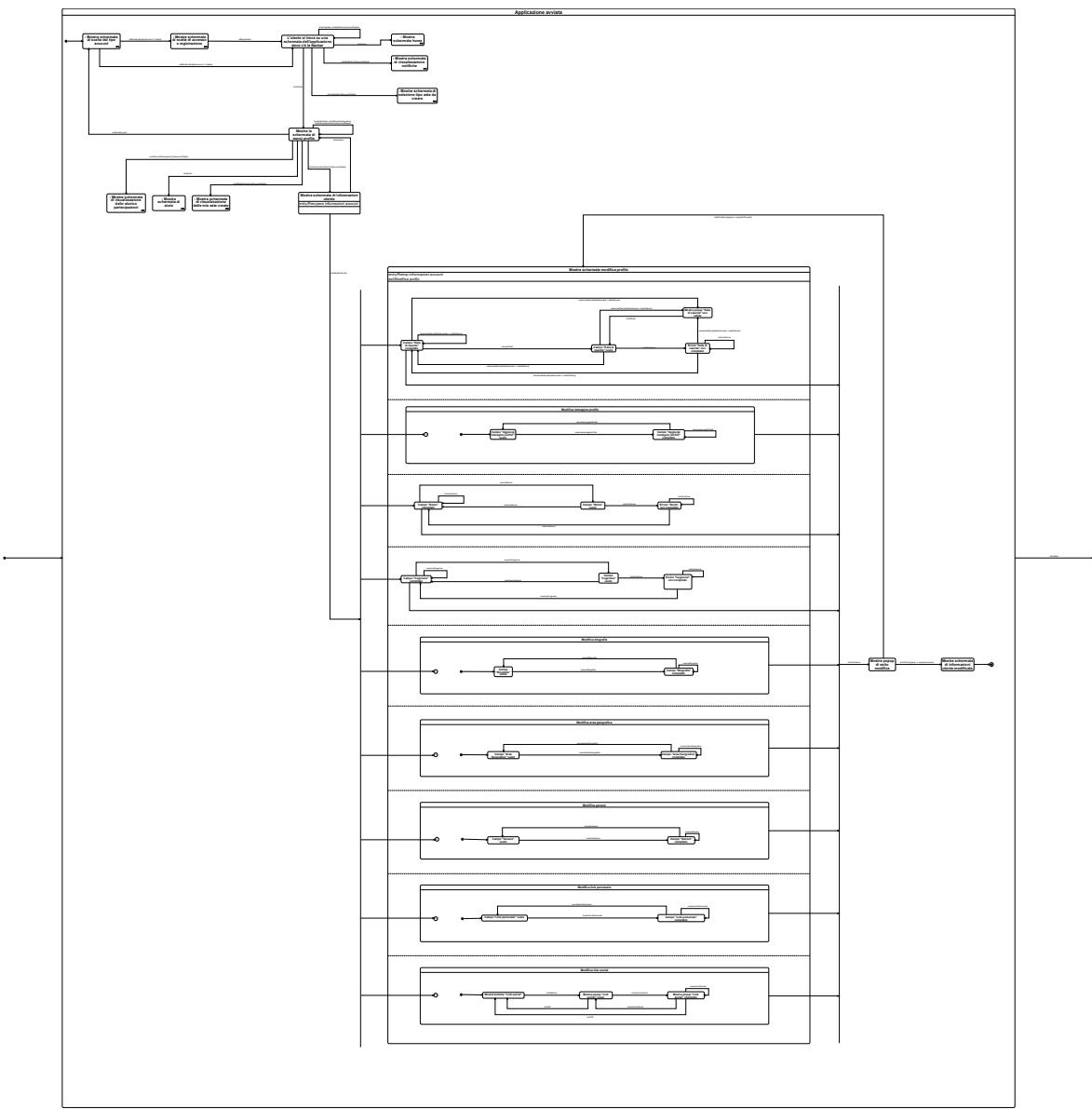


Figura 4.35: Statechart Diagram modifica proprio profilo

CAPITOLO 5

System Design

La progettazione del sistema è un insieme di scelte effettuate dall'ingegnere del software per la trasformazione del modello di analisi nel modello di design del sistema.

Vengono qui definiti gli obiettivi di progettazione del software, l'architettura del sistema (il quale viene decomposto in più sottosistemi realizzabili da differenti squadre di sviluppatori) e le strategie di creazione del sistema.

5.1 Architettura e criteri di progettazione

L'architettura di un software è la struttura del sistema, che comprende gli elementi del software, le proprietà visibili all'esterno di tali elementi e le relazioni tra di essi.

Sulla base delle richieste avanzate dagli utenti, ovvero di un'applicazione **portabile, facile da usare e rapida**, si ritiene che i criteri di progettazione sui quali bisogni focalizzarsi siano **bassi tempi di risposta (bassa latenza), basso consumo di memoria, alta portabilità e modificabilità**.

Per soddisfare tali requisiti qualitativi, il sistema deve essere suddiviso in diversi moduli tali che tra essi sussistano un'**alta coesione** e un **basso accoppiamento**.

Si è optato, in questo sistema, per un'architettura **client-server a due livelli con modello client leggero** e una **base di dati relazionale** per la persistenza dei dati.

Si desidera ora evidenziare i criteri di progettazione definiti che hanno portato a questa decisione:

- La coesione è una misura di quanto siano fortemente relate e mirate le responsabilità, cioè i servizi offerti da un modulo; ciò permetterà di ottenere un'alta manutenibilità del codice e focalizzare i cambiamenti in interventi mirati e precisi.

Per ottenere un'alta coesione, si è optato per una architettura client-server, nella quale ogni modulo adempie ad una funzione ben precisa e le cui classi collaborano per lo stesso fine.

L'architettura client-server è una architettura distribuita, ove un sottosistema "servitore" offre servizi e dati ad altri sottosistemi "cliente", i quali sono responsabili dell'interazione con l'utente. I client chiamano il server richiedendo un servizio e ad essi viene restituito il risultato; questa chiamata è possibile poiché i client conoscono le interfacce del server, ma non è vero il viceversa. In questo modo, tutti gli utilizzatori dell'applicazione possono accedere alla stessa fonte di dati, ossia il server centrale.

- L'accoppiamento è una misura su quanto fortemente un modulo sia dipendente da altri. Un basso accoppiamento permetterà il riutilizzo dei moduli nonché una facilitazione nella modifica dello stesso, senza che le modifiche possano avere conseguenze inaspettate su altri moduli. Per ottenere un basso accoppiamento, si è scelto di adottare i principi espressi dalla legge di Demetra, spingendo al massimo l'incapsulamento.

- Il modello client leggero è l'alternativa al modello client pesante in un'architettura client-server. In questo modello, il client si occupa solamente dell'interfaccia e l'interazione con l'utente, e quindi può essere eseguito su gran parte dei dispositivi, anche con modeste prestazioni.

Lo svantaggio di tale scelta è che pone maggior carico di elaborazione sia sul server, che si occupa della logica dell'applicazione e dell'immagazzinamento dei dati, sia sulla rete.

Questa decisione è anche frutto della piattaforma scelta per il client, ossia i dispositivi mobili;

seppur nel corso del tempo abbiano conosciuto un incremento in termini di capacità di elaborazione, memoria e durata della batteria, restano comunque dei dispositivi poco adatti a applicazioni con elaborazioni troppo pesanti.

- La scelta della base di dati relazionale invece deriva dalla consapevolezza di una struttura fissa dei dati che verranno conservati, e dalla garanzia delle quattro proprietà ACID (atomicità, coerenza, isolamento e durabilità) che essa fornisce.

Sulla base delle specifiche appena esposte si è deciso di organizzare il sistema in due sottosistemi principali:

- Il Client, che si occupa dell'interazione con l'utente, inviando richieste al Resource Server ed elaborandone le risposte;
- Il Resource Server, che si occupa di gestire le richieste pervenute dal Client, far rispettare i vincoli esposti dalla business logic, interagire con la base di dati relazionale per la persistenza dei dati e inviare le risposte al Client. Inoltre, si occupa anche di interagire con i servizi esterni che si sceglierà di usufruire (ad esempio, l'Authorization Server di AWS Cognito con cui interagisce attraverso API).

5.2 Tecnologie adottate

Il client è un'applicazione per sistema operativo **Android**, scritta in linguaggio **Kotlin**.

Il server si articola in:

- Un'applicazione **Java** salvata in un'immagine ed eseguita in un container **Docker**, installato su una macchina virtuale fornita da **AWS EC2**.
- Una base di dati relazionale **PostgreSQL**, al cui interno sono conservati tutti i dati dell'applicazione, eseguita anch'essa come immagine Docker sulla stessa macchina virtuale AWS EC2.

I due container comunicano grazie all'utilizzo di **Docker Compose**.

Per lo scambio di messaggi tra client e server (attraverso le **REST API**), il server adotta lo **Spring Framework**.

5.2.1 Android

Cos'è Android? [10]

Android è un sistema operativo gratuito ed open source, basato su una versione modificata del kernel Linux e di altri software open source. Creato originariamente per dispositivi touch come smartphone e tablet, è ora esteso a circa 2.5 miliardi di dispositivi [14], tra i quali anche smartwatch (Wear OS), smart TV (Android TV) e personal computer (questi ultimi utilizzano una versione modificata della distribuzione Gentoo Linux, chiamata Chrome OS, ma possono eseguire una macchina virtuale per utilizzare applicazioni Android). Il sistema operativo è correntemente sviluppato da Google.



Figura 5.1: Logo di Android

Perché Android?

Quasi tutti ormai posseggono uno smartphone. Ed essendo un sistema operativo così diffuso (conta circa il 69.94% del mercato dei sistemi operativi per dispositivi mobili [24]) e disponibile su una grande varietà di dispositivi, Android è la scelta giusta per rendere l'applicazione disponibile quasi ovunque ed accessibile a chiunque.

5.2.2 AWS EC2 & AWS Cognito

Cos'è AWS? [5]

Amazon Web Services è la soluzione cloud, fornita da Amazon, più ricca e ampiamente adottata al mondo: conta circa 200 servizi forniti dai numerosi centri di dati sparsi in tutto il globo. Attualmente il tasso di adozione dei servizi AWS globalmente è al 71% [16]. AWS offre anche un Free Tier, ossia permette di sfruttare alcuni dei suoi servizi gratuitamente (a tempo limitato, illimitato, a soglia). Inoltre, ogni servizio fornisce delle API per poter richiedere al server la dispensa del servizio affittato.



Figura 5.2: Logo di AWS

Perché AWS? [5]

L'utilizzo di AWS permette di tagliare i costi, seguire un approccio più agile, e innovare più in fretta. AWS offre nettamente più servizi rispetto ai concorrenti e le funzionalità all'interno dei singoli servizi sono più ricche. Inoltre, la maturità, l'esperienza, l'affidabilità e la sicurezza che offre sono senza eguali. Il Free Tier è anche un'ottima opportunità per piccole aziende che richiedono modesti strumenti per il rilascio della propria applicazione. Le API estremamente semplici permettono di richiedere il servizio direttamente nel codice astraendo tutte le fasi intermedie più complesse.

Cos'è AWS EC2?

Amazon Elastic Compute Cloud è uno dei servizi di AWS che consente l'affitto di computer virtuali in cloud sui quali eseguire le proprie applicazioni [9]. Offre la più ampia e profonda piattaforma di computazione, permettendo di scegliere oltre 750 istanze e personalizzare processore, archiviazione, rete, sistema operativo [2].

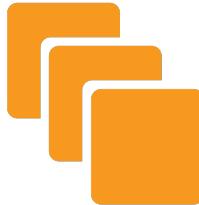


Figura 5.3: Logo di AWS EC2

Perché AWS EC2?

A causa della sua ampia personalizzazione sulla base delle necessità dell'applicazione e grazie alla possibilità di creare, lanciare, e fermare le istanze delle macchine virtuali sul server su richiesta, pagando solo il tempo di computazione sfruttato, l'elasticità del servizio è indubbia [9]. Il Free Tier di EC2 garantisce 750 ore di computazione in 12 mesi su una istanza di tipo "micro" [1].

Cos'è AWS Cognito? [4]

Amazon Cognito è una piattaforma di identità per app web e per dispositivi mobili. È una directory utente, un server di autenticazione e un servizio di autorizzazione per token di accesso OAuth 2.0 e credenziali AWS. Con Amazon Cognito, puoi autenticare e autorizzare gli utenti dalla directory utente integrata, dalla directory aziendale e dai provider di identità utente come Google e Facebook.

Perché AWS Cognito? [3]

- Offre una gestione dell'identità e degli accessi del cliente, grazie a un servizio conveniente e personalizzabile;



Figura 5.4: Logo di AWS Cognito

- Aggiunge funzionalità di sicurezza avanzate come l'autenticazione adattiva basata sul rischio, il monitoraggio delle credenziali compromesse e i parametri di sicurezza;
- Permette di ridimensionare la directory degli utenti fino a milioni di utenti, con alte prestazioni e affidabilità.
- Consente l'accesso federato con OIDC o SAML 2.0 e la connessione ad altri servizi e prodotti AWS.

5.2.3 Kotlin

Cos'è Kotlin? [12]

Kotlin è un linguaggio di programmazione multi-piattaforma, staticamente e fortemente tipizzato con inferenza di tipo, general purpose, multi-paradigma (orientato agli oggetti, imperativo, funzionale). Sviluppato da JetBrains, Kotlin è realizzato per essere interscambiabile con Java, poiché è interpretato da una macchina virtuale Java (JVM) che utilizza la libreria di classi Java standard. Ha pian piano soppiantato l'utilizzo di Java per la realizzazione di applicazioni Android ed è diventato nel 2019 il linguaggio primario e consigliato da Google per la realizzazione di esse.



Figura 5.5: Logo di Kotlin

Perché Kotlin? [20]

Circa il 50% degli sviluppatori Android utilizza Kotlin (Java ricopre il 30%), e il 70% di essi afferma che lo sviluppo in Kotlin li abbia resi più produttivi. Lo sviluppo con Kotlin consente di beneficiare di:

- Minore quantità di codice con conseguente maggiore leggibilità;
- Minore quantità di errori comuni che potrebbero comportare il crash delle applicazioni (ad esempio la gestione della nullità dei riferimenti [21]);
- Supporto per la libreria Jetpack Compose per la costruzione di interfaccia utente delle applicazioni;
- Supporto per lo sviluppo multi-piattaforma, poiché le librerie Kotlin possono essere utilizzabili anche per lo sviluppo di app iOS, desktop e web;
- Maturità del linguaggio e dell'ambiente di sviluppo, continuamente migliorati e supportati;

- Interoperabilità con Java, consentendo di iniettare codice Java in quello Kotlin e di utilizzare librerie di Java con Kotlin;
- Semplicità e facilità di apprendimento ed utilizzo.

5.2.4 Java

Cos'è Java?

Java è un linguaggio di programmazione staticamente e fortemente tipizzato, general purpose, multi-paradigma (orientato agli oggetti, imperativo, funzionale). Sviluppato da Oracle, in precedenza Sun Microsystems, è un linguaggio interpretato e compilato. Il codice sorgente è compilato in un linguaggio intermedio chiamato bytecode che viene interpretato da una macchina virtuale Java (JVM). Java è presente su miliardi di dispositivi in tutto il mondo.



Figura 5.6: Logo di Java

Perché Java?

Il fatto che Java sia un linguaggio eseguito su macchina virtuale comporta:

- Sicurezza, perché ogni programma è isolato e ogni comunicazione con l'esterno è controllata dalla JVM, impedendo l'esecuzione di codice malizioso;
- Portabilità, poiché il codice viene trasformato in un codice universalmente comprensibile da tutte le implementazioni della JVM sui vari dispositivi (PC, dispositivi mobili, dispositivi specializzati, server, workstation...);

Il linguaggio è anche open source e ampiamente utilizzato da milioni di sviluppatori nel mondo, e possiede una ricca libreria standard di funzioni per la realizzazione delle applicazioni. Infine, molti framework, tra i quali il selezionato Spring, si basano proprio su questo linguaggio.

5.2.5 Docker (Compose)

Cos'è Docker (Compose)?

Docker è un insieme di servizi Platform as a Service (PaaS) che utilizza la virtualizzazione a livello di sistema operativo del kernel per eseguire programmi in ambienti isolati e distribuibili chiamati container, i quali sono gestiti dal Docker Engine [11]. I container sono leggeri e al loro interno conterranno tutto il necessario per l'esecuzione del software, senza preoccuparsi di quale macchina sta eseguendo il Docker Engine. Questi container sono configurabili attraverso l'uso di file di configurazione appositi. Compose, inizialmente un add-on per la piattaforma Docker, è stato successivamente incluso al suo

interno, e consente attraverso un singolo file YAML la gestione e la comunicazione reciproca di più container. I container sono un’istanza di un’immagine Docker, ossia un template di sola lettura che istruisce su come il container debba essere creato [8].



Figura 5.7: Logo di Docker

Perché Docker (Compose)?

Il successo di Docker è dovuto al rivoluzionario approccio allo sviluppo che esso promuove. Grazie a Docker, si può impacchettare all’interno di un’immagine Docker l’applicazione e tutto l’ambiente necessario alla sua esecuzione. L’immagine può essere caricata sulla repository centrale di Docker affinché chiunque possa accedervi, o comunque essere distribuita. Non bisogna quindi preoccuparsi di nulla: ogni container è isolato e contiene tutto ciò che è necessario, e non c’è bisogno di installare applicazioni oltre al Docker Engine, né gestire le librerie o le versioni dei linguaggi installati. Aggiornamenti dei software possono essere sempre rilasciati attraverso le immagini, e basta scaricarne la nuova versione e metterla in esecuzione sull’Engine per poter disporre degli ultimi strumenti. Permette anche di avere sulla stessa macchina più istanze in esecuzione di una stessa applicazione o servizio per favorire il bilanciamento del carico, la tolleranza al guasto e l’isolamento in caso di compromissione.

5.2.6 PostgreSQL

Cos’è PostgreSQL? [15]

PostgreSQL è una potente base di dati relazionale gratuita ed open source che usa il linguaggio SQL standard ma non disdegnando delle aggiunte utili per conservare e scalare i carichi di dati più complessi (è gergalmente definito un ”dialetto”). PostgreSQL ha una forte reputazione di RDBMS robusto, affidabile, estensibile. Esso è eseguibile su tutti i principali sistemi operativi e garantisce le proprietà ACID (atomicità, consistenza, isolamento, durabilità) per dati e transazioni. Inoltre fornisce anche utili add-on per estendere le capacità della base di dati. È disponibile sia attraverso riga di comando che GUI.

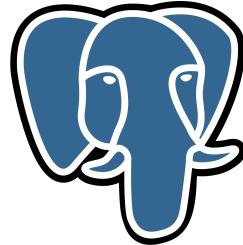


Figura 5.8: Logo di PostgreSQL

Perché PostgreSQL?

PostgreSQL fornisce numerose funzionalità utili per lo sviluppo delle applicazioni, la protezione dell’integrità dei dati e la costruzione di ambienti tolleranti al guasto. Permette anche di scrivere procedure

e funzioni procedurali in differenti linguaggi e per differenti linguaggi di programmazione (PL/pgSQL, PL/Tcl, PL/Perl, PL/Python). Tra le funzionalità notevoli, si possono annoverare:

- Supporto ad una vasta quantità di tipi di dato;
- Funzionalità per il controllo dell'integrità dei dati;
- Esecuzione concorrente e alte prestazioni;
- Affidabilità e ripresa dal disastro;
- Elevata sicurezza;
- Ampia estensibilità;
- Internazionalità e capacità di elaborazione testuale [15];
- Supporto sempre gratuito;
- Il linguaggio procedurale PL/pgSQL è pienamente compatibile con il linguaggio procedurale di altre basi di dati, facilitando eventuali future migrazioni [17].

5.2.7 REST API

Cosa sono le REST API? [19]

Le REST API (REpresentational State Transfer Application Programming Interface) sono un'interfaccia di programmazione dell'applicazione che segue i principi REST, utilizzata per connettere tra loro componenti e applicazioni in un'architettura di microservizi. Le REST API sfruttano l'impalcatura del protocollo HTTP per poter permettere di effettuare tutte le principali operazioni sui dati attraverso i metodi HTTP, e i messaggi sono scambiati attraverso file JSON (ma anche XML o HTML).

Perché le REST API?

I sei principi REST sono un'interfaccia uniforme, il disaccoppiamento client-server, servizio senza stato, possibilità di utilizzo del meccanismo di cache, organizzazione del sistema a più livelli, ed esecuzione di codice su richiesta [19]. La realizzazione di applicazioni che comunicano attraverso REST API permette di spingere al massimo l'indipendenza tra i vari servizi, non rinunciando però ad una semplicità di fondo nella comunicazione, essendo il protocollo HTTP un protocollo maturo, quindi testato e funzionante, e strutturalmente semplice. Inoltre, la comunicazione attraverso file JSON (un formato comodo nella lettura sia per esseri umani che per calcolatori e uno dei formati di scambio dei dati più diffuso al mondo), ne permette la lettura da parte di moltissimi linguaggi di programmazione, ciascuno dei quali in possesso di strumenti di parsing di tale formato.

5.2.8 Spring Framework & Spring Boot

Cos'è Spring Framework?

Spring è un framework gratuito ed open source per applicazioni web scritte in Java. Inizialmente nato come alternativa al modello Enterprise JavaBeans di Java Enterprise Edition (ora Jakarta Enterprise Edition), le sue funzioni possono essere utilizzate da qualsiasi applicazione Java, e fornisce anche numerosi servizi aggiuntivi, che prendono il nome di Spring Boot, Spring Security, Spring Data, eccetera [13]. Grazie a Spring, si possono realizzare delle REST API per la propria applicazione.



Figura 5.9: Logo di Spring Framework

Perché Spring Framework? [23]

I punti forti di Spring Framework sono:

- Alta modularità e basso livello di dipendenze grazie al potente meccanismo di dependency injection, che promuove un basso accoppiamento, facilitando manutenzione e riusabilità;
- Ricco ecosistema, fornendo molti strumenti per ricoprire le varie necessità di sviluppo dell'applicazione;
- Programmazione orientata all'aspetto, ossia permette di incapsulare le fasi più critiche dello sviluppo dell'applicazione, come la sicurezza e il tracciamento delle attività attraverso file di log;
- Una community molto estesa e documentazione molto ampia;
- Il supporto dello sviluppo orientato al test, assieme alla semplicità nella gestione delle dipendenze, contribuisce a migliori pratiche di test;
- Flessibilità nella scelta di specifici componenti e moduli in base alle necessità del progetto.

Cos'è Spring Boot? [18]

Java Spring Boot (Spring Boot) è uno strumento che semplifica e velocizza lo sviluppo di applicazioni web e microservizi con Spring Framework. E' un layer che si trova al di sopra dello Spring Framework e permette di configurarlo velocemente attraverso una serie di convenzioni.



Figura 5.10: Logo di Spring Boot

Perché Spring Boot?

Essendo altamente flessibile e configurabile, Spring Framework richiede un notevole dispendio di tempo e competenze significative per configurare, impostare e implementare applicazioni Spring. Spring Boot agevola questo lavoro tramite tre funzionalità principali:

- Configurazione automatica: le applicazioni vengono inizializzate con dipendenze preimpostate che non si ha bisogno di configurare manualmente, attenendosi alle best practices. Ad ogni modo, è sempre possibile personalizzare queste configurazioni;

- Un approccio categorico alla configurazione: Spring Boot sceglie i pacchetti da installare e i valori predefiniti da utilizzare. Tuttavia è possibile definire le esigenze del proprio progetto nel corso del processo di inizializzazione, durante il quale è possibile scegliere tra più dipendenze iniziali;
- La capacità di creare applicazioni autonome: Spring Boot consente di creare applicazioni autonome che vengono eseguite da server web già integrati nell'applicazione durante il processo di inizializzazione. In questo modo non è necessario fare affidamento su un server web esterno.

Queste funzionalità cooperano per fornire uno strumento che consente di configurare applicazioni basate su Spring che richiedono una configurazione e un'installazione minime.

5.2.9 SonarQube

Cos'è SonarQube?

SonarQube è una piattaforma open-source sviluppata da SonarSource per l'ispezione della qualità del codice. Esegue un'analisi statica automatica del codice per identificare bug, "code smells" e vulnerabilità di sicurezza nel codice. SonarQube supporta diversi linguaggi di programmazione e può integrarsi con vari strumenti di compilazione e pipeline di integrazione continua. La piattaforma fornisce report e visualizzazioni dettagliate per aiutare gli sviluppatori a migliorare la manutenibilità e l'affidabilità del codice.



Figura 5.11: Logo di SonarQube

Perché SonarQube?

I vantaggi dell'utilizzo di SonarQube sono evidenti:

- Migliora la qualità del codice, aiutando a mantenere alta la qualità dello stesso identificando potenziali bug e problemi.
- Identifica il codice che potrebbe funzionare ma che sembra scritto male e difficile da mantenere.
- Rileva le falte di sicurezza che potrebbero essere sfruttate da potenziali aggressori.
- Fornisce il supporto per un'ampia gamma di linguaggi di programmazione.
- Offre dashboard e report completi, ricchi di dettagli ed intuitivi per monitorare lo stato di salute dei progetti nel tempo.

CAPITOLO 6

Object Design

La progettazione orientata agli oggetti è la fase di sviluppo del software nella quale si definisce il dettaglio implementativo dei vari sottosistemi.

Tali sottosistemi devono soddisfare i requisiti non funzionali di performance, affidabilità e manutenibilità. Sono costruiti sulla base degli oggetti individuati nel dominio del problema, ai quali inevitabilmente saranno aggiunti nuovi oggetti per giungere alla soluzione.

Le scelte adoperate hanno l'obiettivo di favorire il basso accoppiamento tra i sottosistemi e la manutenibilità, in modo da mantenere un'alta qualità interna del software.

6.1 Design Pattern

6.1.1 Client

Il client possiede un'architettura con due strati: lo **strato della UI** e lo **strato dei dati**.

Nel complesso, è un ibrido tra il pattern **MVVM** (Model-View-ViewModel) e **MVC** (Model-View-Controller).

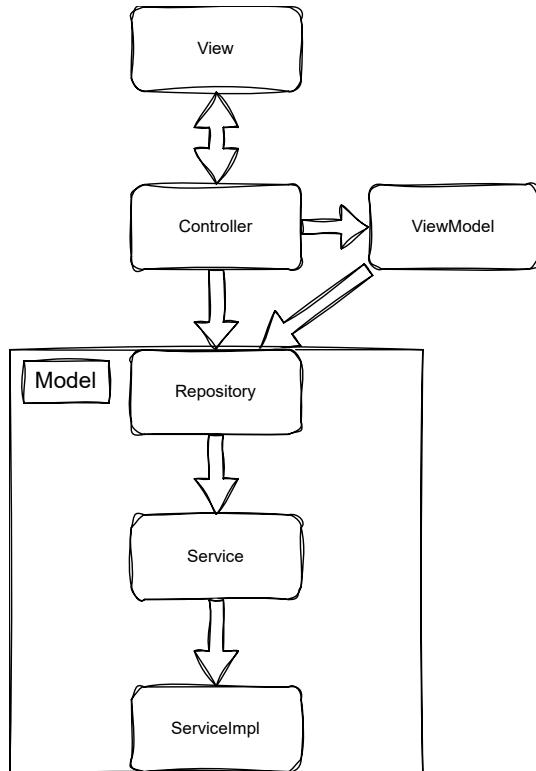


Figura 6.1: Schema dell'architettura adottata (client)

- Lo strato della UI comprende **View**, **StateHolder** (che mantengono i dati sullo stato della View, come i **ViewModel**) e **Controller** e si occupa di gestire l’interfaccia e l’interazione con l’utente.
 - Le *View* sono dei file .XML che utilizzano degli schemi definiti da Android per la creazione dell’interfaccia utente. Inoltre, grazie al View binding, per ogni vista viene generata una classe Java che contiene riferimenti a tutti gli elementi della vista per poter interagire con essa. Esse interagiscono con il Controller per chiamare i metodi che gestiscono l’interazione con l’utente.
 - I *ViewModel* sono delle classi che hanno il compito di mantenere i dati che devono essere mostrati sulla UI. L’utilizzo del ViewModel è essenziale per poter mantenere i dati anche in caso di cambiamenti dello stato dell’applicazione, come rotazioni dello schermo o riduzione ad attività in secondo piano. I ViewModel, che nella architettura MVVM interagiscono direttamente con il Model e si interpongono tra quest’ultimo e la View, eseguono anche operazioni sui dati, come il recupero dei dati paginati da mostrare sullo schermo e la validazione dei dati inseriti dall’utente.
 - L’aggiunta del *Controller* è ciò che rende ibrida questa architettura. Il Controller si interponde tra ViewModel e View per separare i compiti che riguardano il conservare e il convalidare i dati della vista (affidati al ViewModel) da altre operazioni, come appunto le risposte all’interazione con l’utente, la creazione di messaggi di errore, e l’invio di richieste non relative a dati da mostrare, come ad esempio l’autenticazione. Inoltre esso funge da osservatore ai dati che sono presenti nel ViewModel e, in caso di mutazione, provvede ad aggiornare la vista con tali nuovi dati.
- Lo strato dei dati è rappresentato dal **Model** e il suo ruolo è comunicare con le fonti di dati, siano esse esterne (come basi di dati) che locali (come cache o basi di dati interne). Il Model, a sua volta, è organizzato secondo il pattern **Repository**, con tre sottostrati: **Repository**, **Service** e **ServiceImpl**.
 - Le classi *Repository* rappresentano una enumerazione delle operazioni che la logica dell’applicazione può effettuare sui dati, chiamando poi attraverso le interfacce Service i metodi che svolgono l’effettivo lavoro sui dati.
 - Le classi *Service* rappresentano una interfaccia che astrae queste operazioni sui dati. In questo caso, esse rappresentano una interfaccia di metodi che verranno utilizzati per la creazione di richieste HTTP da inviare agli endpoint del backend.
 - Le classi *ServiceImpl* rappresentano dei metodi concreti - implementati dalla libreria Retrofit - che, sulla base delle annotazioni e i parametri inseriti nelle interfacce dei metodi di Service, creeranno la giusta richiesta HTTP da inviare al backend.

6.1.2 Resource Server

L’architettura adottata per il sottosistema **Resource Server** (da qui in avanti chiamato semplicemente **Server**) è un’architettura di tipo **esagonale**. L’obiettivo di tale architettura è di definire una **separazione netta** tra la business logic del dominio (anche detto *domain layer*) e tutto ciò che sono i fattori esterni, ovvero il layer di interfaccia con l’utente (anche detto *application layer*) e il layer di persistenza (anche detto *infrastructure layer*), attraverso una serie di interfacce ben definite.

L’obiettivo di tale scelta è quello di migliorare la manutenibilità, la scalabilità e la modularità del backend, definendo così interfacce ben definite che permettono a più team di sviluppo di lavorare parallelamente, conoscendo esattamente il contratto per le interazioni tra la logica del dominio e dei protocolli di comunicazione tra l’applicazione e i suoi adattatori. [6]

In particolare, il design pattern adottato per il domain layer è di tipo **Domain Driven**. Un approccio di tipo Domain Driven permette di avere un’**alta coesione** e **ridurre al minimo l’accompagnamento** tra le classi della business logic, dando a ciascuna classe del domain layer un solo motivo per cambiare: un cambiamento nell’entità del dominio corrispondente.

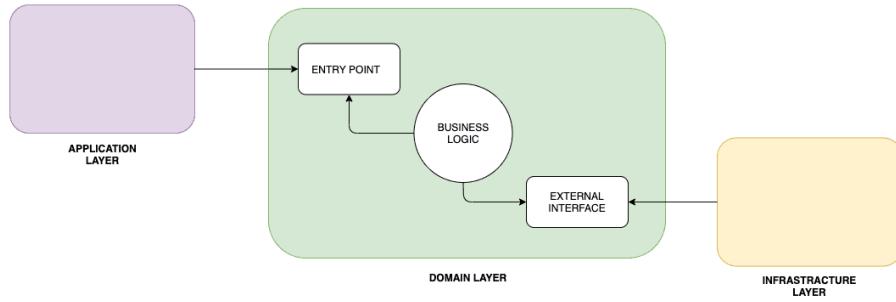


Figura 6.2: Schema dell'architettura adottata (backend)
[22]

In particolare, il backend è così organizzato:

- L'*application layer* contiene le classi responsabili di esporre le REST API che verranno chiamate dall'utente. E' dipendente solo e unicamente da classi DTO, che si occupano di raccogliere le informazioni ricevute sottoforma di HTTP Request e saranno tradotte in HTTP Response. [7]
- Il *domain layer* contiene le classi responsabili della business logic, che dipende solo e unicamente dalle classi del dominio, ovvero le entità. Al suo interno troviamo anche classi responsabili di tradurre i DTO in entità del dominio.
Inoltre, grazie all'interfaccia offerta dalla **Java Persistance API**, le classi necessarie per la comunicazione con il persistance layer sono gestite da Spring Boot attraverso una Spring dependency.
- Il *persistance layer* contiene i driver responsabili della comunicazione con il database.

6.2 Class Diagram

A seguito delle considerazioni fatte ai punti precedenti, in questa sezione verranno rappresentati i diagrammi delle classi di design del client e del server.

In particolare, il tipo di visibilità è espressa da questa tabella:

Icon for field	Icon for method	Visibility
□	■	private
◇	◆	protected
△	▲	package private
○	●	public

Figura 6.3: Legenda della visibilità

Inoltre, i costruttori della classe sono i metodi contrassegnati da <<Create>>.

6.2.1 Client

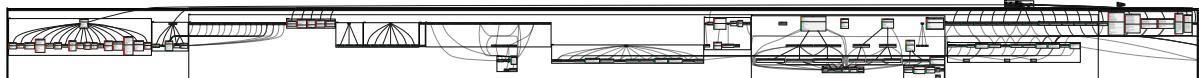


Figura 6.4: Class Diagram di design del Client

6.2.2 Server

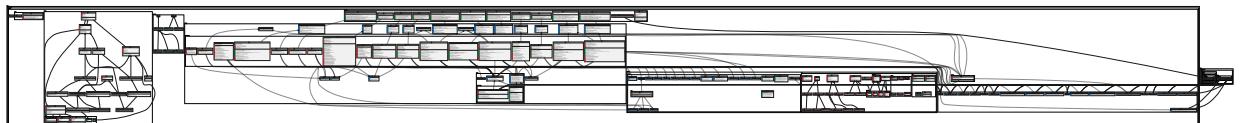


Figura 6.5: Class Diagram di design del Server

6.3 Sequence Diagram

In questa sezione saranno rappresentati i diagrammi di sequenza di design del client e del server per i casi d'uso individuati nella fase di Requirement Analysis.

6.3.1 I casi d'uso: Crea un'asta inversa

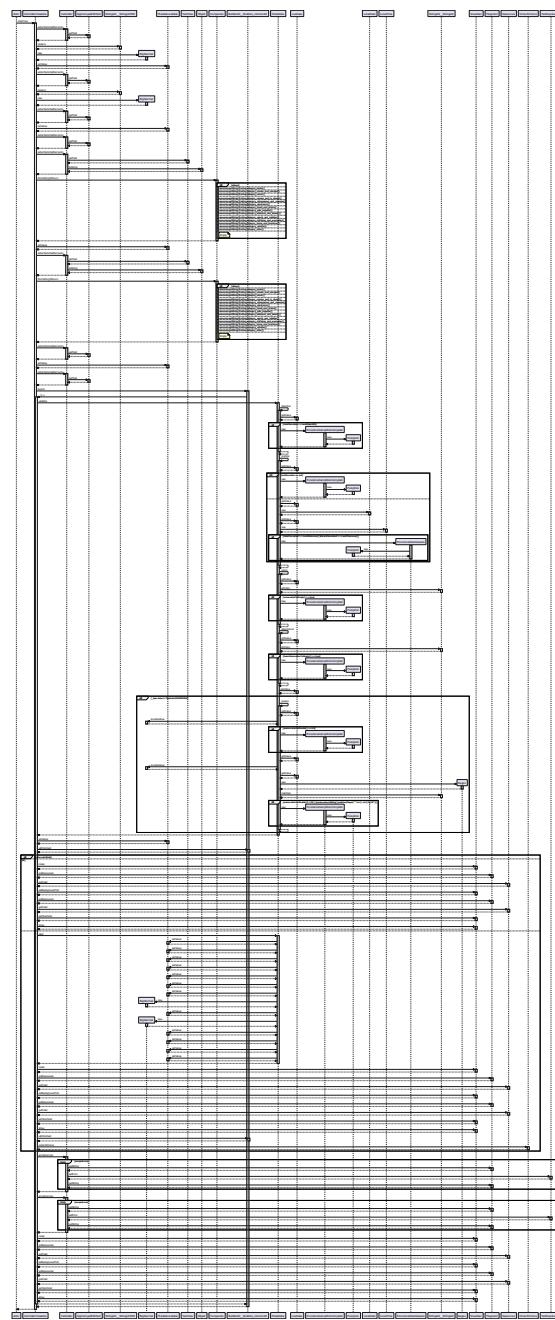


Figura 6.6: Sequence Diagram di design creazione asta inversa (client)

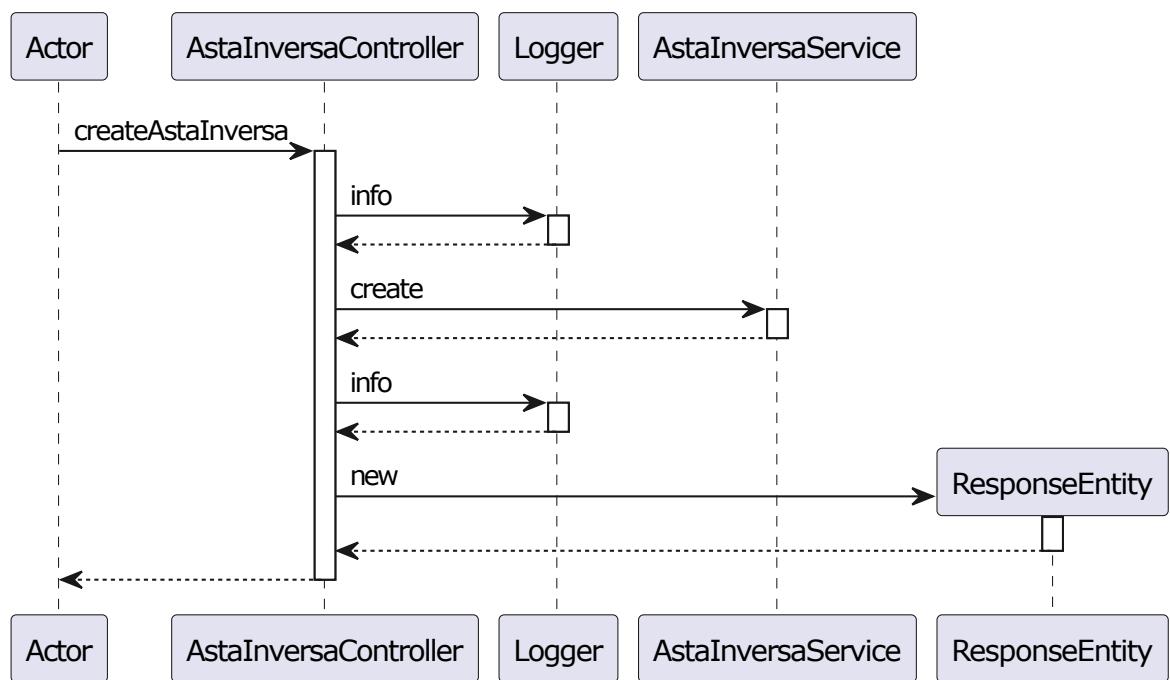


Figura 6.7: Sequence Diagram di design creazione asta inversa (server)

6.3.2 II caso d'uso: Accetta un'offerta ricevuta all'asta silenziosa

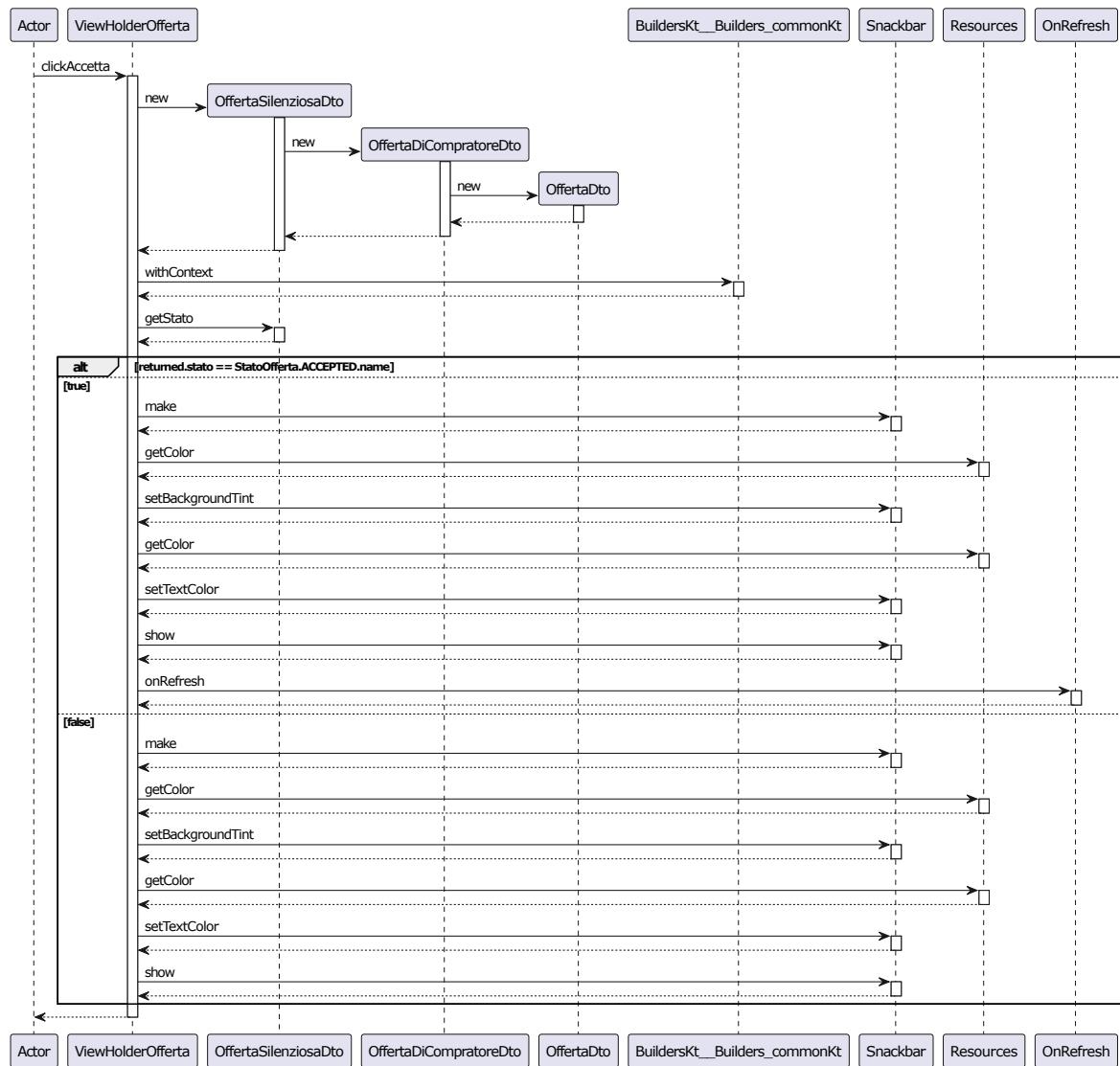


Figura 6.8: Sequence Diagram di design accetta offerta per un'asta silenziosa (client)

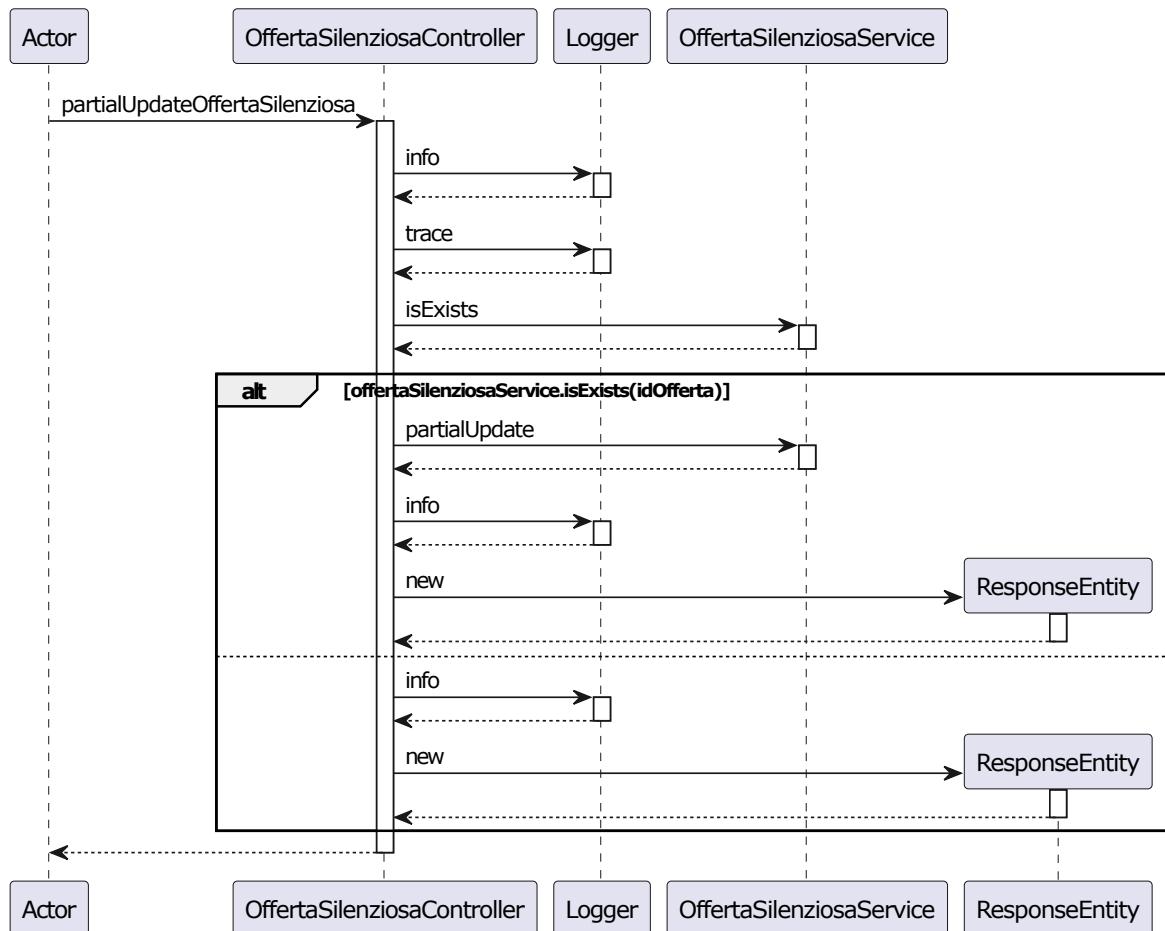


Figura 6.9: Sequence Diagram di design accetta offerta per un'asta silenziosa (server)

6.3.3 III caso d'uso: Visualizza i dettagli di un'asta

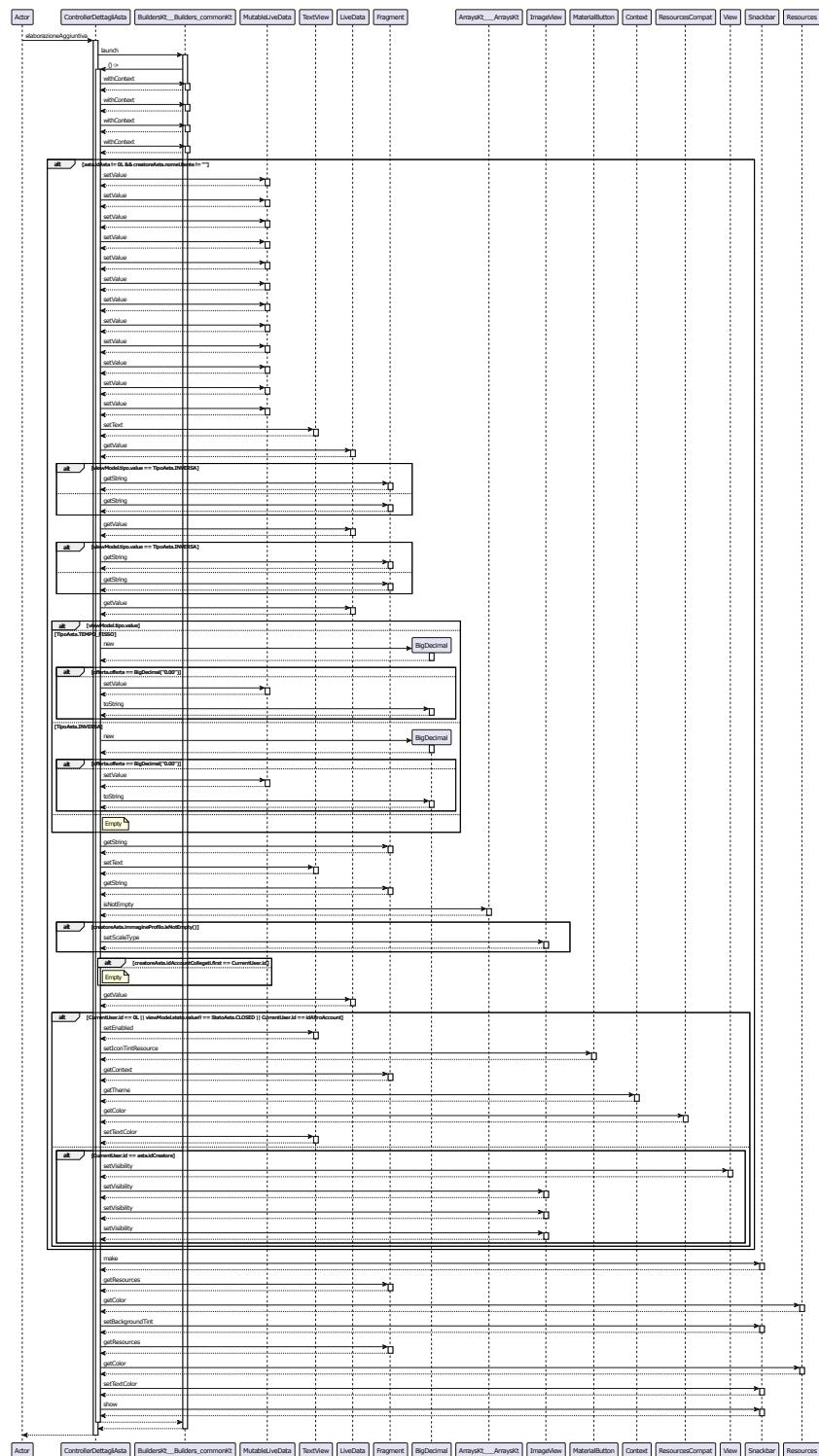


Figura 6.10: Sequence Diagram di design visualizza dettagli asta (client)

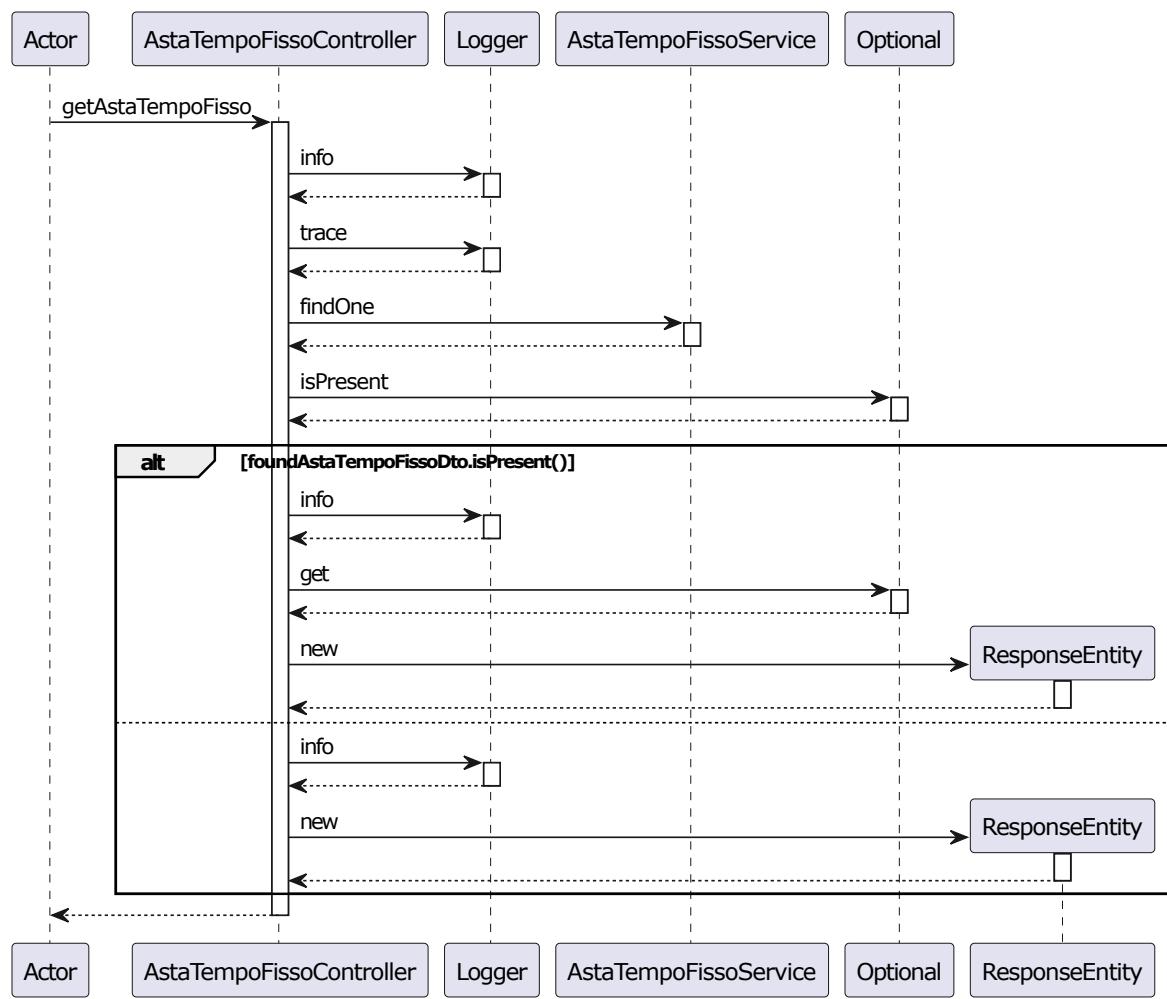


Figura 6.11: Sequence Diagram di design visualizza dettagli asta (server)

6.3.4 IV caso d'uso: Modificare il proprio profilo

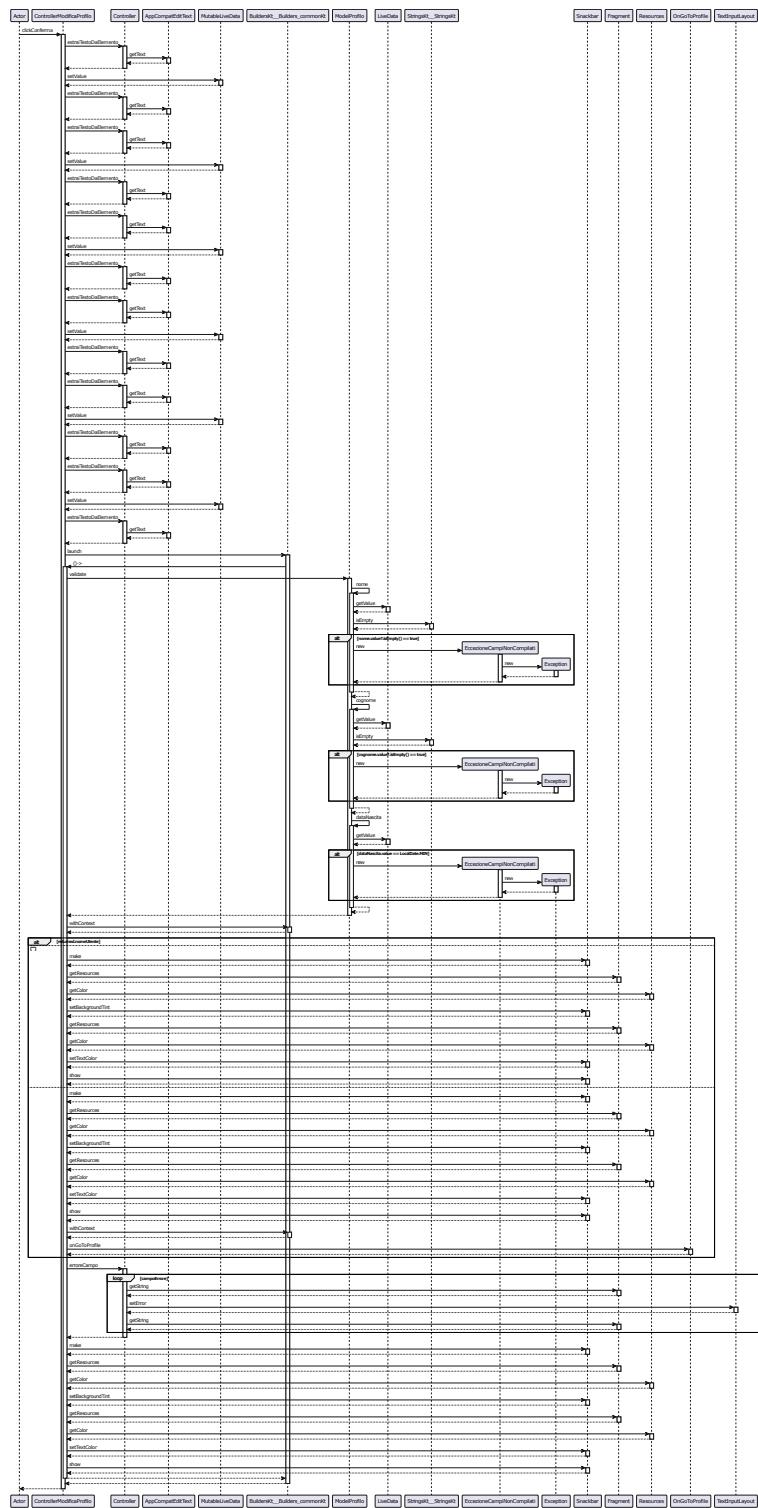


Figura 6.12: Sequence Diagram di design modifica dati profilo (client)

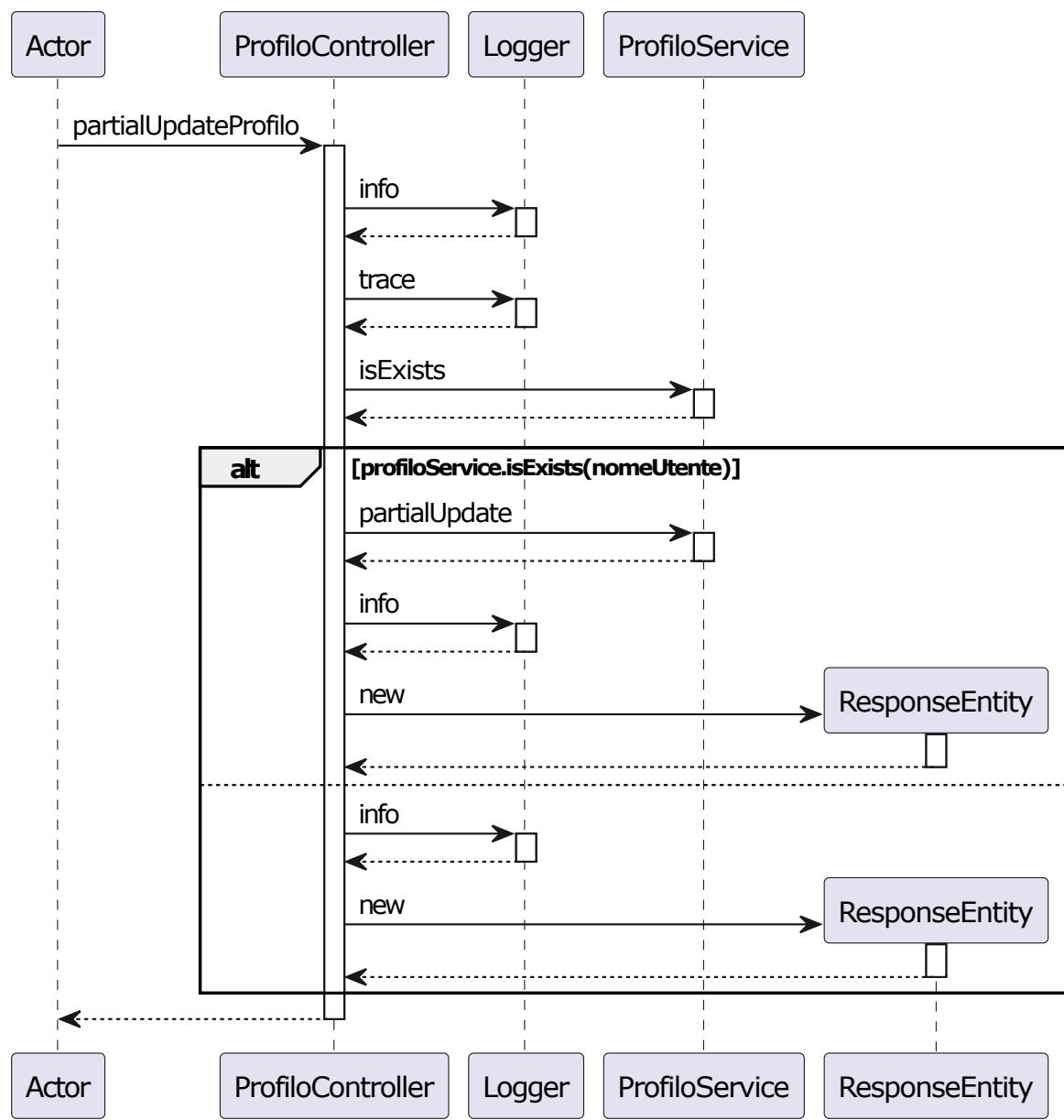


Figura 6.13: Sequence Diagram di design modifica dati profilo (server)

6.4 Endpoint

In questa sezione vengono descritti gli endpoints esposti dal backend. Per gli endpoints che accettano query parameters, le ulteriori "modalità" sono separate da un ";" e, rispettivamente, anche l'autorizzazione minima è separata da un ";".

Le autorizzazioni definite sono:

- ADMIN: E' l'autorizzazione più elevata, che garantisce completo accesso a tutti gli endpoint dell'applicazione;
- USER: E' l'autorizzazione base, che garantisce l'accesso ai soli endpoint che riguardano un utente loggato;
- NO AUTH: Non è necessaria alcuna autenticazione per accedere agli endpoint contrassegnati da questa autorizzazione.

Per una descrizione dettagliata dei query parameters, esempi di richieste e di risposte per ciascun endpoint, riferirsi alla [documentazione generata tramite Postman](#).

URI	ME-THOD	DESCRIZIONE	AUTORIZZA-ZIONE MINIMA
/accounts/compratori	POST	Permette di creare un account compratore	USER
	GET	Permette di recuperare tutti gli account compratori; filtrati per email; id	ADMIN; USER; NO AUTH
	PUT	Permette la sostituzione di un account compratore	ADMIN
	PATCH	Permette di modificare un campo "non associazione" di un account compratore	ADMIN
	DELETE	Permette di eliminare un account compratore	ADMIN
/accounts/venditori	POST	Permette di creare un account venditore	USER
	GET	Permette di recuperare tutti gli account venditori; filtrati per email; id	ADMIN; USER; NO AUTH
	PUT	Permette la sostituzione di un account venditore	ADMIN
	PATCH	Permette di modificare un campo "non associazione" di un account venditore	ADMIN
	DELETE	Permette di eliminare un account venditore	ADMIN
/aste/di-compratori/inverse	POST	Permette di creare un'asta inversa	USER
	GET	Permette di recuperare tutte le aste inverse; filtrate per nome e categoria; proprietario; offerente; id	NO AUTH; NO AUTH; USER; USER; NO AUTH

	PUT	Permette la sostituzione di un'asta inversa	ADMIN
	PATCH	Permette di modificare un campo "non associazione" di un'asta inversa	USER
	DELETE	Permette di eliminare un'asta inversa	USER
/aste/di-venditori/silenziose	POST	Permette di creare un'asta silenziosa	USER
	GET	Permette di recuperare tutte le aste silenziose; filtrate per nome e categoria; proprietario; offerente; id	NO AUTH; NO AUTH; USER; USER; NO AUTH
	PUT	Permette la sostituzione di un'asta silenziosa	ADMIN
	PATCH	Permette di modificare un campo "non associazione" di un'asta silenziosa	USER
	DELETE	Permette di eliminare un'asta silenziosa	USER
/aste/di-venditori/tempo-fisso	POST	Permette di creare un'asta a tempo fisso	USER
	GET	Permette di recuperare tutte le aste a tempo fisso; filtrate per nome e categoria; proprietario; offerente; id	NO AUTH; NO AUTH; USER; USER; NO AUTH
	PUT	Permette la sostituzione di un'asta a tempo fisso	ADMIN
	PATCH	Permette di modificare un campo "non associazione" di un'asta a tempo fisso	USER
	DELETE	Permette di eliminare un'asta a tempo fisso	USER
/offerte/di-compratori/tempo-fisso	POST	Permette di creare un'offerta a tempo fisso	USER
	GET	Permette di recuperare tutte le offerte a tempo fisso; filtrate per asta di riferimento; maggior valore e asta riferimento; maggior valore, asta riferimento e venditore; id	ADMIN; USER; NO AUTH; USER; ADMIN
	PUT	Permette la sostituzione di un'offerta a tempo fisso	ADMIN
	PATCH	Permette di modificare un campo "non associazione" di un'offerta a tempo fisso	ADMIN
	DELETE	Permette di eliminare un'offerta a tempo fisso	ADMIN
/offerte/di-compratori/silenziose	POST	Permette di creare un'offerta silenziosa	USER

	GET	Permette di recuperare tutte le offerte silenziose; filtrate per asta di riferimento; maggior valore e asta riferimento; maggior valore, asta riferimento e venditore; id	ADMIN; USER; NO AUTH; USER; ADMIN
	PUT	Permette la sostituzione di un'offerta silenziosa	ADMIN
	PATCH	Permette di modificare un campo "non associazione" di un'offerta silenziosa	USER
	DELETE	Permette di eliminare un'offerta silenziosa	ADMIN
/offerte/di-venditori/inverse	POST	Permette di creare un'offerta inversa	USER
	GET	Permette di recuperare tutte le offerte inverse; filtrate per asta di riferimento; maggior valore e asta riferimento; maggior valore, asta riferimento e venditore; id	ADMIN; USER; NO AUTH; USER; ADMIN
	PUT	Permette la sostituzione di un'offerta inversa	ADMIN
	PATCH	Permette di modificare un campo "non associazione" di un'offerta inversa	ADMIN
	DELETE	Permette di eliminare un'offerta inversa	ADMIN
/categorie-asta	PUT	Permette di creare (o sostituire, se già esistente) una categoria asta	ADMIN
	GET	Permette di recuperare tutte le categorie asta; filtrate per id	NO AUTH; NO AUTH
	PATCH	Permette di modificare un campo "non associazione" di una categoria asta	ADMIN
	DELETE	Permette di eliminare una categoria asta	ADMIN
/notifiche	POST	Permette di creare una notifica	ADMIN
	GET	Permette di recuperare tutte le notifiche; filtrate per destinatario; id	ADMIN; USER; ADMIN
	PUT	Permette la sostituzione di una notifica	ADMIN
	PATCH	Permette di modificare un campo "non associazione" di una notifica	ADMIN
	DELETE	Permette di eliminare una notifica	ADMIN
/profili	PUT	Permette di creare (o sostituire, se già esistente) un profilo. Viene automaticamente creato un account associato	USER
	GET	Permette di recuperare tutti i profili; filtrati per id	ADMIN; NO AUTH

	PATCH	Permette di modificare un campo "non associazione" di un profilo	USER
	DELETE	Permette di eliminare un profilo	ADMIN
/auth/url	GET	Permette di recuperare l'URL a cui collegarsi per fare l'autenticazione	NO AUTH
/auth/callback	GET	Permette di autenticare un account in base al codice restituito dal provider di autenticazione	NO AUTH
/auth/refresh	GET	Permette di aggiornare la sessione di autenticazione tramite il refresh_token restituito da auth/callback	NO AUTH
/auth/logout	GET	Permette di eliminare la sessione di autenticazione tramite il refresh_token restituito da auth/callback e restituisce l'URL a cui accedere per dimenticare le credenziali di login	USER

6.5 Database

L'analisi dei requisiti permette di individuare le entità, relazioni ed attributi che caratterizzano il dominio del problema. Tuttavia, per immagazzinare questi dati in una base di dati relazionale in modo efficace è necessario fare uno studio adeguato al fine di arrivare a uno schema di base di dati che permetta di rispettare i requisiti non funzionali di velocità ed efficienza.

6.5.1 Ristrutturazione della progettazione concettuale

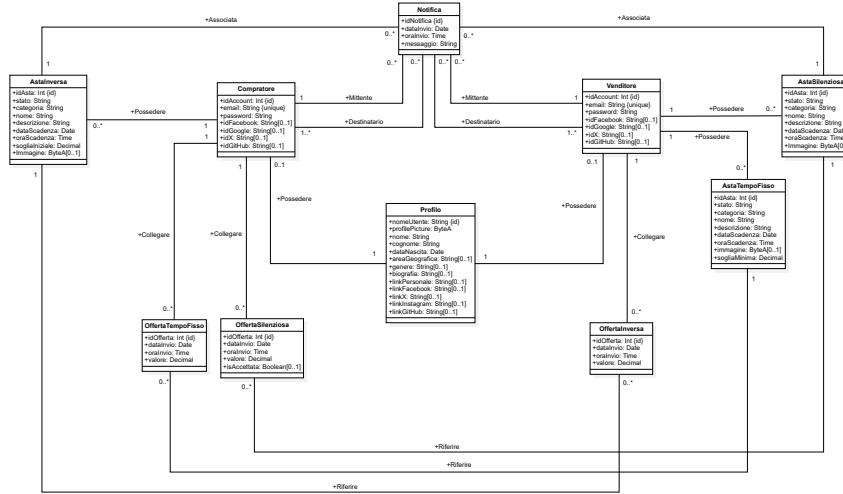


Figura 6.14: Schema concettuale ristrutturato del database

6.5.2 Schema logico

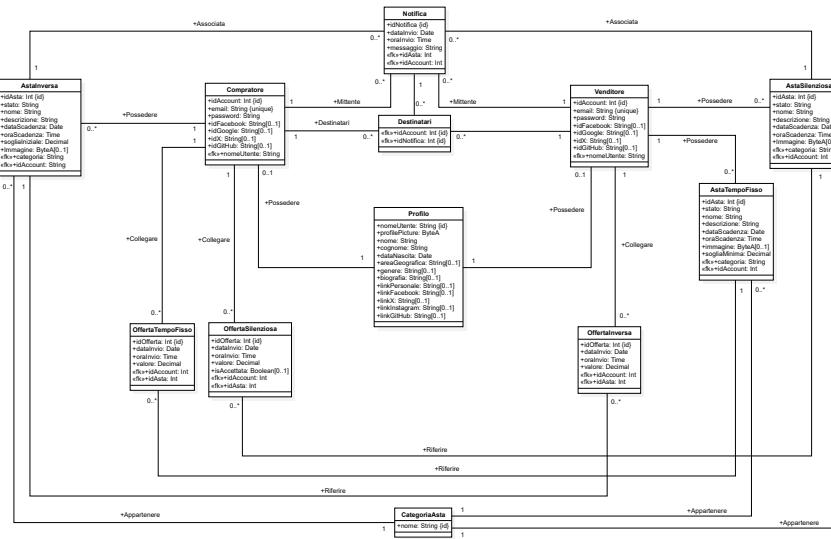


Figura 6.15: Schema logico del database

6.5.3 Schema fisico

```

1      DROP SCHEMA IF EXISTS dd24 CASCADE;
2      CREATE SCHEMA dd24;
3      SET search_path TO dd24;
4

```

Listing 6.1: Preparazione ambiente

```

1      CREATE TABLE categoria_asta
2      (
3          nome TEXT NOT NULL,
4          CONSTRAINT pk_categoria_asta PRIMARY KEY (nome)
5      );
6

```

Listing 6.2: Relazione categoria asta

```

1      CREATE TABLE profilo
2      (
3          nome_utente      TEXT NOT NULL,
4          CONSTRAINT pk_profilo PRIMARY KEY (nome_utente),
5          area_geografica TEXT,
6          biografia        TEXT,
7          cognome         TEXT NOT NULL,
8          data_nascita    DATE NOT NULL,
9          CONSTRAINT chk_data_nascita CHECK (data_nascita <= NOW()),
10         genere          TEXT NOT NULL,
11         nome            TEXT NOT NULL,
12         link_personale TEXT,
13         link_facebook   TEXT,
14         link_x          TEXT,
15         link_instagram  TEXT,
16         link_git_hub    TEXT,
17         profile_picture BYTEA
18     );
19

```

Listing 6.3: Relazione profilo

```

1      CREATE TABLE compratore
2      (
3          id_account      BIGSERIAL NOT NULL,
4          CONSTRAINT pk_compratore PRIMARY KEY (id_account),
5          email           TEXT      NOT NULL,
6          CONSTRAINT uk_compratore_email UNIQUE (email),
7          password        TEXT      NOT NULL,
8          id_facebook     TEXT,
9          id_git_hub      TEXT,
10         id_google       TEXT,
11         id_x            TEXT,
12         profilo_nome_utente TEXT      NOT NULL,
13         CONSTRAINT fk_profilo_nome_utente FOREIGN KEY (profilo_nome_utente)
14     ) REFERENCES profilo (nome_utente) ON UPDATE CASCADE ON DELETE CASCADE
15
16         CREATE FUNCTION cleanup_compratore()
17             RETURNS TRIGGER AS
18                 $$
19                 BEGIN
20                     DELETE
21                     FROM offerta_silenziosa
22                     WHERE compratore_id_account = OLD.id_account;
23                     DELETE
24                     FROM offerta_tempo_fisso

```

```

25      WHERE compratore_id_account = OLD.id_account;
26      DELETE
27      FROM asta_inversa
28      WHERE compratore_id_account = OLD.id_account;
29      DELETE
30      FROM destinatari
31      WHERE account_id_account = OLD.id_account;
32      DELETE
33      FROM notifica
34      WHERE account_id_account = OLD.id_account;
35  END
36  $$;
37      LANGUAGE PLPGSQL;
38
39      CREATE TRIGGER trg_compratore
40          BEFORE DELETE
41          ON compratore
42          FOR EACH ROW
43          EXECUTE FUNCTION cleanup_compratore();
44

```

Listing 6.4: Relazione compratore

```

1      CREATE TABLE venditore
2      (
3          id_account            BIGSERIAL NOT NULL,
4          CONSTRAINT pk_venditore PRIMARY KEY (id_account),
5          email                 TEXT      NOT NULL,
6          CONSTRAINT uk_venditore_email UNIQUE (email),
7          password              TEXT      NOT NULL,
8          id_facebook           TEXT,
9          id_git_hub            TEXT,
10         id_google              TEXT,
11         id_x                  TEXT,
12         profilo_nome_utente   TEXT      NOT NULL,
13         CONSTRAINT fk_profilo_nome_utente FOREIGN KEY (profilo_nome_utente)
14     ) REFERENCES profilo (nome_utente) ON UPDATE CASCADE ON DELETE CASCADE
15
16     CREATE FUNCTION cleanup_venditore()
17         RETURNS TRIGGER AS
18     $$
19     BEGIN
20         DELETE
21         FROM offerta_inversa
22         WHERE venditore_id_account = OLD.id_account;
23         DELETE
24         FROM asta_tempo_fisso
25         WHERE venditore_id_account = OLD.id_account;
26         DELETE
27         FROM asta_silenziosa
28         WHERE venditore_id_account = OLD.id_account;
29         DELETE
30         FROM destinatari
31         WHERE account_id_account = OLD.id_account;
32         DELETE
33         FROM notifica
34         WHERE account_id_account = OLD.id_account;
35     END
36  $$;
37      LANGUAGE PLPGSQL;
38
39      CREATE TRIGGER trg_venditore
40          BEFORE DELETE
41          ON venditore
42          FOR EACH ROW

```

```

43     EXECUTE FUNCTION cleanup_venditore();
44

```

Listing 6.5: Relazione venditore

```

1      CREATE TABLE asta_inversa
2      (
3          id_asta            BIGSERIAL      NOT NULL,
4          CONSTRAINT pk_asta_inversa PRIMARY KEY (id_asta),
5          data_scadenza     DATE          NOT NULL,
6          CONSTRAINT chk_data_scadenza CHECK (data_scadenza > NOW()),
7          descrizione       TEXT          NOT NULL,
8          immagine          BYTEA ,
9          nome               TEXT          NOT NULL,
10         ora_scadenza     TIME          NOT NULL,
11         categoria_asta_nome TEXT          NOT NULL,
12         CONSTRAINT fk_categoria_asta_nome FOREIGN KEY (categoria_asta_nome)
13        ) REFERENCES categoria_asta (nome) ON UPDATE CASCADE ON DELETE CASCADE,
14                  compratore_id_account BIGINT      NOT NULL,
15                  CONSTRAINT fk_compratore_id_account FOREIGN KEY (
16                  compratore_id_account) REFERENCES compratore (id_account) ON UPDATE CASCADE ON
17                  DELETE CASCADE,
18                  soglia_iniziale      DECIMAL(2, 10) NOT NULL,
19                  CONSTRAINT chk_soglia_iniziale CHECK (soglia_iniziale >= 0),
20                  stato              TEXT          NOT NULL
21      );
22
23      CREATE FUNCTION cleanup_asta_inversa()
24          RETURNS TRIGGER AS
25          $$$
26          BEGIN
27              DELETE
28              FROM notifica
29              WHERE asta_id_asta = OLD.id_asta;
30          END
31          $$$
32          LANGUAGE PLPGSQL;
33
34      CREATE TRIGGER trg_asta_inversa
35          BEFORE DELETE
36          ON asta_inversa
37          FOR EACH ROW
38          EXECUTE FUNCTION cleanup_asta_inversa();
39
40

```

Listing 6.6: Relazione asta inversa

```

1      CREATE TABLE asta_silenziosa
2      (
3          id_asta            BIGSERIAL NOT NULL,
4          CONSTRAINT pk_asta_silenziosa PRIMARY KEY (id_asta),
5          data_scadenza     DATE          NOT NULL,
6          CONSTRAINT chk_data_scadenza CHECK (data_scadenza > NOW()),
7          descrizione       TEXT          NOT NULL,
8          immagine          BYTEA ,
9          nome               TEXT          NOT NULL,
10         ora_scadenza     TIME          NOT NULL,
11         categoria_asta_nome TEXT          NOT NULL,
12         CONSTRAINT fk_categoria_asta_nome FOREIGN KEY (categoria_asta_nome)
13        ) REFERENCES categoria_asta (nome) ON UPDATE CASCADE ON DELETE CASCADE,
14                  venditore_id_account BIGINT      NOT NULL,
15                  CONSTRAINT fk_venditore_id_account FOREIGN KEY (venditore_email)
16                  REFERENCES venditore (id_account) ON UPDATE CASCADE ON DELETE CASCADE,
17                  stato              TEXT          NOT NULL
18      );
19

```

```

17      CREATE FUNCTION cleanup_asta_silenziosa()
18          RETURNS TRIGGER AS
19              $$
20          BEGIN
21              DELETE
22                  FROM notifica
23                  WHERE asta_id_asta = OLD.id_asta;
24          END
25          $$;
26          LANGUAGE PLPGSQL;
27
28
29      CREATE TRIGGER trg_asta_silenziosa
30          BEFORE DELETE
31          ON asta_silenziosa
32          FOR EACH ROW
33          EXECUTE FUNCTION cleanup_asta_silenziosa();
34

```

Listing 6.7: Relazione asta silenziosa

```

1      CREATE TABLE asta_tempo_fisso
2      (
3          id_asta            BIGSERIAL NOT NULL,
4          CONSTRAINT pk_asta_tempo_fisso PRIMARY KEY (id_asta),
5          data_scadenza      DATE        NOT NULL,
6          CONSTRAINT chk_data_scadenza CHECK (data_scadenza > NOW()),
7          descrizione        TEXT       NOT NULL,
8          immagine           BYTEA,
9          nome               TEXT       NOT NULL,
10         ora_scadenza      TIME       NOT NULL,
11         categoria_asta_nome TEXT       NOT NULL,
12         CONSTRAINT fk_categoria_asta_nome FOREIGN KEY (categoria_asta_nome)
13     ) REFERENCES categoria_asta (nome) ON UPDATE CASCADE ON DELETE CASCADE,
14             venditore_id_account BIGINT    NOT NULL,
15             CONSTRAINT fk_venditore_id_account FOREIGN KEY (
16                 venditore_id_account) REFERENCES venditore (id_account) ON UPDATE CASCADE ON DELETE
17             CASCADE,
18                 soglia_minima      DECIMAL(2, 10) NOT NULL,
19                 CONSTRAINT chk_soglia_minima CHECK (soglia_minima >= 0),
20                 stato             TEXT       NOT NULL
21     );
22
23
24      CREATE FUNCTION cleanup_asta_tempo_fisso()
25          RETURNS TRIGGER AS
26              $$
27          BEGIN
28              DELETE
29                  FROM notifica
30                  WHERE asta_id_asta = OLD.id_asta;
31          END
32          $$;
33          LANGUAGE PLPGSQL;
34
35
36      CREATE TRIGGER trg_asta_tempo_fisso
37          BEFORE DELETE
38          ON asta_tempo_fisso
39          FOR EACH ROW
40          EXECUTE FUNCTION cleanup_asta_tempo_fisso();

```

Listing 6.8: Relazione asta a tempo fisso

```

1      CREATE TABLE notifica
2      (

```

```

3      id_notifica      BIGSERIAL NOT NULL,
4      CONSTRAINT pk_notifica PRIMARY KEY (id_notifica),
5      data_invio        DATE      NOT NULL,
6      CONSTRAINT chk_data_invio CHECK (data_invio <= NOW()),
7      messaggio         TEXT     NOT NULL,
8      ora_invio         TIME      NOT NULL,
9      asta_id_asta     BIGINT    NOT NULL,
10     account_id_account BIGINT    NOT NULL
11 );
12
13     CREATE FUNCTION chk_asta_id_asta()
14     RETURNS TRIGGER AS
15     $$$
16     BEGIN
17         IF
18             NOT EXISTS (SELECT id_asta FROM asta_inversa WHERE id_asta =
19             NEW.asta_id_asta) AND
20                 NOT EXISTS (SELECT id_asta FROM asta_silenziosa WHERE id_asta
21             = NEW.asta_id_asta) AND
22                 NOT EXISTS (SELECT id_asta FROM asta_tempo_fisso WHERE id_asta
23             = NEW.asta_id_asta) THEN
24                     RAISE EXCEPTION 'L''identificativo dell''asta del record
25             inserito non riferenzia un''asta esistente';
26             ELSE
27                 RETURN NEW;
28             END IF;
29         END;
30         $$$
31         LANGUAGE PLPGSQL;
32
33     CREATE TRIGGER trg_asta_id_asta
34     BEFORE INSERT OR
35         UPDATE OF asta_id_asta
36     ON notifica
37     FOR EACH ROW
38     EXECUTE FUNCTION chk_asta_id_asta();
39
40     CREATE FUNCTION chk_account_id_account()
41     RETURNS TRIGGER AS
42     $$$
43     BEGIN
44         IF
45             NOT EXISTS (SELECT id_account FROM compratore WHERE id_account
46             = NEW.id_account) AND
47                 NOT EXISTS (SELECT id_account FROM venditore WHERE id_account
48             = NEW.id_account) THEN
49                     RAISE EXCEPTION 'L''identificativo dell''account del record
50             inserito non riferenzia un account esistente';
51             ELSE
52                 RETURN NEW;
53             END IF;
54         END;
55         $$$
56         LANGUAGE PLPGSQL;
57
58     CREATE TRIGGER trg_account_id_account
59     BEFORE INSERT OR
60         UPDATE OF account_id_account
61     ON notifica
62     FOR EACH ROW
63     EXECUTE FUNCTION chk_account_id_account();

```

Listing 6.9: Relazione notifica

```

1          CREATE TABLE destinatari
2          (
3              notifica_id_notifica BIGINT NOT NULL,
4              CONSTRAINT fk_notifica_id_notifica FOREIGN KEY (
5      notifica_id_notifica) REFERENCES notifica (id_notifica) ON UPDATE CASCADE ON DELETE
6      CASCADE,
7                  account_id_account    BIGINT NOT NULL,
8                  CONSTRAINT pk_destinatari PRIMARY KEY (notifica_id_notifica,
9      account_id_account)
10             );
11
12             CREATE TRIGGER trg_account_id_account
13                 BEFORE INSERT OR
14                     UPDATE OF account_id_account
15                 ON destinatari
16                 FOR EACH ROW
17                     EXECUTE FUNCTION chk_account_id_account();

```

Listing 6.10: Relazione destinatari

```

1          CREATE TABLE offerta_inversa
2          (
3              id_offerta            BIGSERIAL      NOT NULL,
4              CONSTRAINT pk_offerta_inversa PRIMARY KEY (id_offerta),
5              data_invio           DATE          NOT NULL,
6              CONSTRAINT chk_data_invio CHECK (data_invio <= NOW()),
7              ora_invio            TIME          NOT NULL,
8              valore               DECIMAL(2, 10) NOT NULL,
9              CONSTRAINT chk_valore CHECK (valore > 0),
10             venditore_id_account BIGINT        NOT NULL,
11             CONSTRAINT fk_venditore_id_account FOREIGN KEY (
12     venditore_id_account) REFERENCES venditore (id_account) ON UPDATE CASCADE ON DELETE
13     CASCADE,
14             asta_inversa_id_asta BIGINT        NOT NULL,
15             CONSTRAINT fk_asta_inversa_id_asta FOREIGN KEY (
16     asta_inversa_id_asta) REFERENCES asta_inversa (id_asta) ON UPDATE CASCADE ON DELETE
17     CASCADE
18         );

```

Listing 6.11: Relazione offerta inversa

```

1          CREATE TABLE offerta_silenziosa
2          (
3              id_offerta            BIGSERIAL      NOT NULL,
4              CONSTRAINT pk_offerta_silenziosa PRIMARY KEY (id_offerta),
5              data_invio           DATE          NOT NULL,
6              CONSTRAINT chk_data_invio CHECK (data_invio <= NOW()),
7              ora_invio            TIME          NOT NULL,
8              valore               DECIMAL(2, 10) NOT NULL,
9              CONSTRAINT chk_valore CHECK (valore > 0),
10             compratore_id_account BIGINT        NOT NULL,
11             CONSTRAINT fk_compratore_id_account FOREIGN KEY (
12     compratore_id_account) REFERENCES compratore (id_account) ON UPDATE CASCADE ON
13     DELETE CASCADE,
14             stato                TEXT          NOT NULL,
15             asta_silenziosa_id_asta BIGINT        NOT NULL,
16             CONSTRAINT fk_asta_silenziosa_id_asta FOREIGN KEY (
17     asta_silenziosa_id_asta) REFERENCES asta_silenziosa (id_asta) ON UPDATE CASCADE ON
18     DELETE CASCADE
19         );

```

Listing 6.12: Relazione offerta silenziosa

```

1   CREATE TABLE offerta_tempo_fisso
2   (
3       id_offerta          BIGSERIAL      NOT NULL,
4       CONSTRAINT pk_offerta_tempo_fisso PRIMARY KEY (id_offerta),
5       data_invio          DATE          NOT NULL,
6       CONSTRAINT chk_data_invio CHECK (data_invio <= NOW()),
7       ora_invio           TIME          NOT NULL,
8       valore              DECIMAL(2, 10) NOT NULL,
9       CONSTRAINT chk_valore CHECK (valore > 0),
10      compratore_id_account BIGINT        NOT NULL,
11      CONSTRAINT fk_compratore_id_account FOREIGN KEY (
12          compratore_id_account) REFERENCES compratore (id_account) ON UPDATE CASCADE ON
13          DELETE CASCADE,
14          asta_tempo_fisso_id_asta BIGINT        NOT NULL,
15          CONSTRAINT fk_asta_tempo_fisso_id_asta FOREIGN KEY (
16              asta_tempo_fisso_id_asta) REFERENCES asta_tempo_fisso (id_asta) ON UPDATE CASCADE
17          ON DELETE CASCADE
18      );

```

Listing 6.13: Relazione offerta a tempo fisso

CAPITOLO 7

Verifica del software

La fase di verifica del software fa parte dell'ultima fase del ciclo di vita del software. In particolare, l'obiettivo della *verifica del software* è quello di accertarsi che ciò che è stato implementato sia **corretto e funzionante**, così da essere coerente con quelle che sono le specifiche descritte nel documento dell'analisi dei requisiti. Si è attraversata sia una prima fase di verifica statica che, in seguito, di verifica dinamica.

7.1 Verifica statica

La verifica statica è basata su tecniche di analisi statica del software, senza ricorso all'esecuzione del codice. Questo è stato fatto prevalentemente per **ispezione del codice**, così da individuare in anticipo i possibili problemi e risolverli ancor prima che il software fosse pronto per l'esecuzione. In particolare, il processo di ispezione è stato semi-automatizzato attraverso il software **SonarQube**. SonarQube mette a disposizione degli sviluppatori un'ampia dashboard con numerosi grafici e statistiche che misurano la qualità del codice. Tra i più interessanti possiamo appuntare i seguenti:



Figura 7.1: Grafico dei code smells e dei bug

Il grafico sopra mostrato evidenzia i **bug** e i **code smells** nel codice, ossia delle abitudini o delle tecniche di scrittura dello stesso (definiti "pattern") che sfavoriscono la manutenibilità e/o la sicurezza. Sarebbe necessario mantenerne al minimo per cercare di ottenere un codice quanto più pulito possibile. Uno strumento che può correre in aiuto in tal senso è anche il plugin **SonarLint**, che si connette all'istanza in esecuzione di SonarQube sulla macchina per dare consigli on-the-fly per migliorare il codice, simili a quelli che fornirebbe un'analisi effettuata con SonarQube.

Interessante è anche esaminare il grafico che mostra il livello di **duplicazione** nel codice.

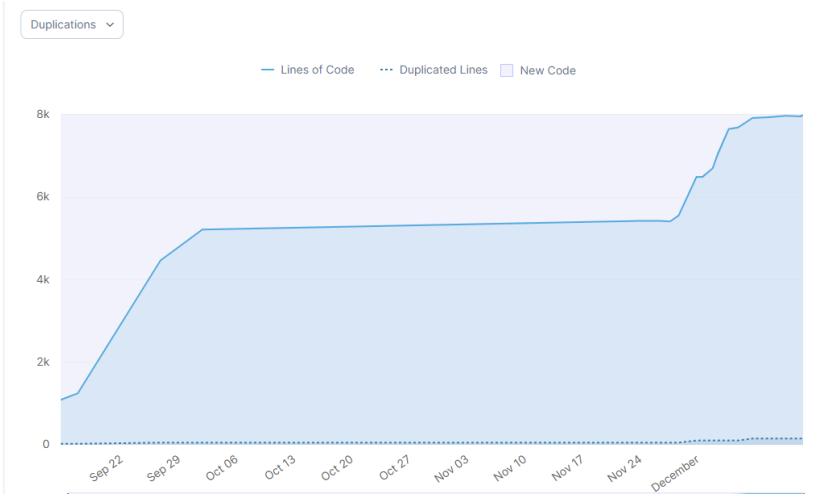


Figura 7.2: Grafico della duplicazione

La linea blu continua mostra l'aumento della quantità delle righe di codice (asse delle ordinate, attualmente circa 8000 righe) nel tempo (asse delle ascisse).

La linea tratteggiata, invece, mostra la quantità di righe che risultano duplicate all'interno del codice. Avere poche duplicazioni è essenziale considerando che sfavorisce la manutenibilità, in quanto in caso di modifica in un punto è necessaria una modifica anche negli altri punti, nonché la leggibilità, poiché allunga il corpo dei metodi nei quali viene inserito.

Un livello così basso di duplicazione è stato ottenuto grazie a un attento **refactoring** del codice, ossia un'organizzazione altamente modulare dello stesso per favorire la futura evoluzione del software.

TESTS				
Total	Success Rate	Skipped	Errors	Failures
91	100.0 %	0	0	0

Figura 7.3: Tabella dei test eseguiti

Osserviamo anche dal report di SonarQube le analisi dei test scritti per confermare il corretto funzionamento del software (sia Unit che Integration tests), appuntando come essi siano tutti risultati in un esito positivo.

Prendiamo infine in esame il cruscotto riassuntivo del progetto nella dashboard di SonarQube, ove possiamo controllare la valutazione che lo strumento affibbia ad altre quattro qualità del software:

- **Sicurezza**, ossia problemi che possono comportare un accesso o un utilizzo non autorizzato dell'backend dell'applicazione da parte di utenti malintenzionati;
- **Affidabilità**, ossia la capacità del codice di essere eseguito così come documentato gestendo i vari input e casi eccezionali senza causare crash o altri problemi;
- **Manutenibilità**, ossia la facilità di modifica del software nel caso di futuri sviluppi ed evoluzioni;
- **Hotspot di sicurezza**, ossia dei punti nei quali bisogna prestare particolare attenzione ai meccanismi di sicurezza per evitare utilizzo non consenso o non autorizzato del software.

La valutazione per questi aspetti è, in una scala letterale americana, una "A", mostrando gli standard di qualità ai quali si attiene il codice, e tutti i punti caldi di sicurezza risultano revisionati e messi al sicuro.



Figura 7.4: Valutazione generale

7.2 Verifica dinamica

La verifica dinamica è basata su tecniche di analisi dinamica del software, cioè per mezzo della sua esecuzione e osservandone il comportamento. Attraverso tecniche quali **testing** e **debugging** è possibile scovare i *failure* ed individuarne i *fault* che li hanno causati, così da correggerli. In particolare, il testing è stato effettuato attraverso la libreria **JUnit** sia su backend che su frontend (in quest'ultimo caso, un'implementazione apposita per Android).

7.2.1 I test: offertaService.checkFieldsValid (server)

Questo metodo verifica la validità dell'offerta inviata, verificando che i suoi attributi non violino le regole di business.

La firma del metodo è la seguente:

```
void checkFieldsValid(OffertaDto offertaDto)
```

Dunque il metodo accetta un parametro di input, che tuttavia ha i seguenti attributi:

- **idOfferta**, che è un numero intero. Può essere qualsiasi valore;
- **dataInvio**, che è una data. Deve essere precedente alla data attuale;
- **oraInvio**, che è un orario. Se la data è precedente a quella attuale, può essere qualsiasi valore. Se la data è la stessa di quella attuale, deve essere precedente all'ora attuale
- **valore**, che è un numero decimale. Deve essere positivo

Si individuino le classi di equivalenza secondo un approccio basato per funzionalità.

- idOfferta:
 - **C.E. 1:** idOfferta > 0;
 - **C.E. 2:** idOfferta = 0;
 - **C.E. 3:** idOfferta < 0;
 - **C.E. 4:** idOfferta = null.

- dataInvio:
 - **C.E. 1:** dataInvio > data attuale;
 - **C.E. 2:** dataInvio = data attuale;
 - **C.E. 3:** dataInvio < data attuale;
 - **C.E. 4:** dataInvio = null.
- oraInvio:
 - **C.E. 1:** oraInvio > ora attuale;
 - **C.E. 2:** oraInvio = ora attuale;
 - **C.E. 3:** oraInvio < ora attuale;
 - **C.E. 4:** oraInvio = null.
- valore:
 - **C.E. 1:** valore > 0;
 - **C.E. 2:** valore = 0;
 - **C.E. 3:** valore < 0.
 - **C.E. 4:** valore = null.

Sceglieremo di fare testing **Black-Box** con criterio di copertura per la combinazione di valori di tipo **R-WECT** (Robust - Weak Equivalence Class Testing).

R-WECT prevede un totale di $4 + 7 = 11$ tests, ovvero il più grande partizionamento (4 classi di equivalenza) più 7 casi non validi. Si scelga un valore da ciascuna classe di equivalenza.

Per semplicità, esprimo i valori degli attributi del parametro in questo modo:
`offertaDto{idOfferta; dataInvio; oraInvio; valore}`

I test case sono:

- `checkFieldsValid(offertaDto{5; LocalDate.now().minusDays(5); LocalTime.now().minusMinutes(5); BigDecimal.valueOf(1)})`
- `checkFieldsValid(offertaDto{0; LocalDate.now(); LocalTime.now(); BigDecimal.valueOf(2)})`
- `checkFieldsValid(offertaDto{-5; LocalDate.now().minusDays(5); LocalTime.now().plusMinutes(6); BigDecimal.valueOf(4)})`
- `checkFieldsValid(offertaDto{null; LocalDate.now(); LocalTime.now().minusMinutes(6); BigDecimal.valueOf(4)})`
- `checkFieldsValid(offertaDto{5; LocalDate.now(); LocalTime.now().plusMinutes(5); BigDecimal.valueOf(1)})`
- `checkFieldsValid(offertaDto{4; LocalDate.now().plusDays(3); LocalTime.now().minusMinutes(6); BigDecimal.valueOf(4)})`
- `checkFieldsValid(offertaDto{4; null; LocalTime.now().minusMinutes(6); BigDecimal.valueOf(4)})`
- `checkFieldsValid(offertaDto{4; LocalDate.now().minusDays(5); null; BigDecimal.valueOf(4)})`

- checkFieldsValid(offertaDto{5; LocalDate.now().minusDays(5); LocalTime.now().minusMinutes(5); BigDecimal.valueOf(0)})
- checkFieldsValid(offertaDto{5; LocalDate.now().minusDays(5); LocalTime.now().minusMinutes(5); BigDecimal.valueOf(-5)})
- checkFieldsValid(offertaDto{5; LocalDate.now().minusDays(5); LocalTime.now().minusMinutes(5); null})

Ossia:

```

1  class OffertaServiceTests {
2
3      private OffertaService offertaService;
4
5      @BeforeEach
6      void initUnderTest() {
7          offertaService = new OffertaServiceImpl();
8      }
9
10     @Test
11     void testCheckFieldsValid_idMaggioreDataMinoreOraMinoreValoreMaggiore() {
12         // Arrange
13         OffertaDto offertaDto = new OffertaDto();
14         offertaDto.setIdOfferta(5L);
15         offertaDto.setDataInvio(LocalDate.now().minusDays(5));
16         offertaDto.setOraInvio(LocalTime.now().minusMinutes(5));
17         offertaDto.setValore(BigDecimal.valueOf(1));
18
19         // Act & Assert
20         assertDoesNotThrow(() -> offertaService.checkFieldsValid(offertaDto));
21     }
22
23     @Test
24     void testCheckFieldsValid_idZeroDataAttualeOraAttualeValoreMaggiore() {
25         // Arrange
26         OffertaDto offertaDto = new OffertaDto();
27         offertaDto.setIdOfferta(0L);
28         offertaDto.setDataInvio(LocalDate.now());
29         offertaDto.setOraInvio(LocalTime.now());
30         offertaDto.setValore(BigDecimal.valueOf(2));
31
32         // Act & Assert
33         assertDoesNotThrow(() -> offertaService.checkFieldsValid(offertaDto));
34     }
35
36     @Test
37     void testCheckFieldsValid_idMinoreDataMinoreOraMaggioreValoreMaggiore() {
38         // Arrange
39         OffertaDto offertaDto = new OffertaDto();
40         offertaDto.setIdOfferta(-5L);
41         offertaDto.setDataInvio(LocalDate.now().minusDays(5));
42         offertaDto.setOraInvio(LocalTime.now().plusMinutes(6));
43         offertaDto.setValore(BigDecimal.valueOf(4));
44
45         // Act & Assert
46         assertDoesNotThrow(() -> offertaService.checkFieldsValid(offertaDto));
47     }
48
49     @Test
50     void testCheckFieldsValid_idNullDataAttualeOraMinoreValoreMaggiore() {
51         // Arrange
52         OffertaDto offertaDto = new OffertaDto();
53         offertaDto.setIdOfferta(null);
54         offertaDto.setDataInvio(LocalDate.now());
```

```

55     offertaDto.setOraInvio(LocalTime.now().minusMinutes(6));
56     offertaDto.setValore(BigDecimal.valueOf(4));
57
58     // Act & Assert
59     assertDoesNotThrow(() -> offertaService.checkFieldsValid(offertaDto));
60 }
61
62     @Test
63     void testCheckFieldsValid_idMaggioreDataAttualeOraMaggioreValoreMaggiore() {
64         // Arrange
65         OffertaDto offertaDto = new OffertaDto();
66         offertaDto.setIdOfferta(5L);
67         offertaDto.setDataInvio(LocalDate.now());
68         offertaDto.setOraInvio(LocalTime.now().plusMinutes(5));
69         offertaDto.setValore(BigDecimal.valueOf(1));
70
71         // Act & Assert
72         assertThrowsExactly(InvalidParameterException.class, () -> offertaService.
73             checkFieldsValid(offertaDto));
74     }
75
76     @Test
77     void testCheckFieldsValid_idMaggioreDataMaggioreOraMinoreValoreMaggiore() {
78         // Arrange
79         OffertaDto offertaDto = new OffertaDto();
80         offertaDto.setIdOfferta(4L);
81         offertaDto.setDataInvio(LocalDate.now().plusDays(3));
82         offertaDto.setOraInvio(LocalTime.now().minusMinutes(6));
83         offertaDto.setValore(BigDecimal.valueOf(4));
84
85         // Act & Assert
86         assertThrowsExactly(InvalidParameterException.class, () -> offertaService.
87             checkFieldsValid(offertaDto));
88     }
89
90     @Test
91     void testCheckFieldsValid_idMaggioreDataNullOraMinoreValoreMaggiore() {
92         // Arrange
93         OffertaDto offertaDto = new OffertaDto();
94         offertaDto.setIdOfferta(4L);
95         offertaDto.setDataInvio(null);
96         offertaDto.setOraInvio(LocalTime.now().minusMinutes(6));
97         offertaDto.setValore(BigDecimal.valueOf(4));
98
99         // Act & Assert
100        assertThrowsExactly(InvalidParameterException.class, () -> offertaService.
101            checkFieldsValid(offertaDto));
102    }
103
104    @Test
105    void testCheckFieldsValid_idMaggioreDataMinoreOraNullValoreMaggiore() {
106        // Arrange
107        OffertaDto offertaDto = new OffertaDto();
108        offertaDto.setIdOfferta(4L);
109        offertaDto.setDataInvio(LocalDate.now().minusDays(5));
110        offertaDto.setOraInvio(null);
111        offertaDto.setValore(BigDecimal.valueOf(1));
112
113        // Act & Assert
114        assertThrowsExactly(InvalidParameterException.class, () -> offertaService.
115            checkFieldsValid(offertaDto));
116    }
117
118    @Test
119    void testCheckFieldsValid_idMaggioreDataMinoreOraMinoreValoreZero() {

```

```

116     // Arrange
117     OffertaDto offertaDto = new OffertaDto();
118     offertaDto.setIdOfferta(5L);
119     offertaDto.setDataInvio(LocalDate.now().minusDays(5));
120     offertaDto.setOraInvio(LocalTime.now().minusMinutes(5));
121     offertaDto.setValore(BigDecimal.valueOf(0));
122
123     // Act & Assert
124     assertThrowsExactly(InvalidParameterException.class, () -> offertaService.
checkFieldsValid(offertaDto));
125   }
126
127   @Test
128   void testCheckFieldsValid_idMaggioreDataMinoreOraMinoreValoreMinore() {
129     // Arrange
130     OffertaDto offertaDto = new OffertaDto();
131     offertaDto.setIdOfferta(5L);
132     offertaDto.setDataInvio(LocalDate.now().minusDays(5));
133     offertaDto.setOraInvio(LocalTime.now().minusMinutes(5));
134     offertaDto.setValore(BigDecimal.valueOf(-5));
135
136     // Act & Assert
137     assertThrowsExactly(InvalidParameterException.class, () -> offertaService.
checkFieldsValid(offertaDto));
138   }
139
140   @Test
141   void testCheckFieldsValid_idMaggioreDataMinoreOraMinoreValoreNull() {
142     // Arrange
143     OffertaDto offertaDto = new OffertaDto();
144     offertaDto.setIdOfferta(5L);
145     offertaDto.setDataInvio(LocalDate.now().minusDays(5));
146     offertaDto.setOraInvio(LocalTime.now().minusMinutes(5));
147     offertaDto.setValore(null);
148
149     // Act & Assert
150     assertThrowsExactly(InvalidParameterException.class, () -> offertaService.
checkFieldsValid(offertaDto));
151   }
152 }

```

Listing 7.1: OffertaServiceTests.java

7.2.2 II test: accountService.isEmailAndPasswordValid (server)

Questo metodo verifica la validità dell'email e password passati per parametro.

La firma del metodo è la seguente:

```
boolean isEmailAndPasswordValid(String email, String password)
```

Dunque il metodo accetta due parametro di input:

- **email**, che è una stringa. Deve soddisfare la regex $\wedge[A-Za-z0-9+.-]+\@\.(.+)\$$;
- **password**, che è una stringa. Deve essere non vuota.

Si individuino le classi di equivalenza secondo un approccio basato per funzionalità.

- email:
 - **C.E. 1:** email = null;

- **C.E. 2:** email = "";
 - **C.E. 3:** email rispetta l'espressione regolare;
 - **C.E. 4:** email non rispetta l'espressione regolare.
- password:
 - **C.E. 1:** password = null;
 - **C.E. 2:** password = "";
 - **C.E. 3:** password è una qualsiasi stringa.

Sceglieremo di fare testing **Black-Box** con criterio di copertura per la combinazione di valori di tipo **N-WECT** (Normal - Weak Equivalence Class Testing).
 N-WECT prevede un totale di 4 tests, ovvero il più grande partizionamento (4 classi di equivalenza). Si scelga un valore da ciascuna classe di equivalenza.

I test case sono:

- `isEmailAndPasswordValid(null, null)`
- `isEmailAndPasswordValid("", "")`
- `isEmailAndPasswordValid("abc@def.it", "abc")`
- `isEmailAndPasswordValid("abc", "")`

Inoltre, in aggiunta al test Black-Box, ne facciamo anche uno di tipo **White-Box** con criterio di copertura per la combinazione di valori di tipo **branch coverage**. Prima di tutto, rappresentiamo il grafo del flusso di controllo della funzione:

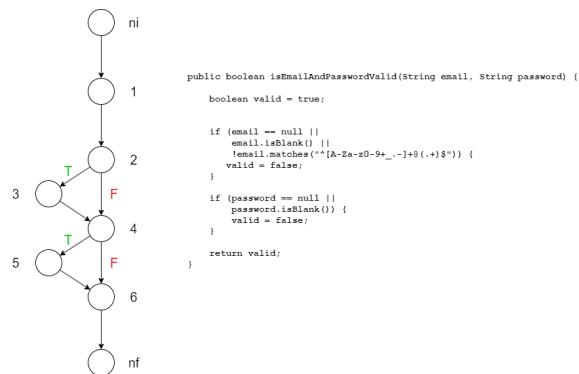


Figura 7.5: Grafo del flusso di controllo di `isEmailAndPasswordValid`

Si scelga un valore in modo da coprire ogni ramo.

I test case sono:

- `isEmailAndPasswordValid("", null)`
- `isEmailAndPasswordValid("abc", "abc")`
- `isEmailAndPasswordValid("abc@def.it", null)`
- `isEmailAndPasswordValid("ABc@CDe.A", "123")`

Ossia:

```
1  class AccountServiceTests {
2
3      @MockBean
4      private TokensAccountMapper tokensAccountMapper;
5
6      @MockBean
7      private RelationsConverter relationsConverter;
8
9      @MockBean
10     private AccountRepository accountRepository;
11
12     private AccountServiceImpl accountService;
13
14     @BeforeEach
15     void initUnderTest() {
16         this.accountService = new AccountServiceImpl(tokensAccountMapper,
17             relationsConverter, accountRepository);
18     }
19
20     // Test Black-Box
21     @Test
22     void testIsEmailAndPasswordValid_EmailNullPasswordNull() {
23         // Arrange
24         String email = null;
25         String password = null;
26
27         boolean result = accountService.isEmailAndPasswordValid(email, password);
28
29         // Act
30         boolean oracolo = false;
31
32         // Assert
33         assertEquals(oracolo, result);
34     }
35
36     @Test
37     void testIsEmailAndPasswordValid_EmailBlankPasswordBlank() {
38         // Arrange
39         String email = "";
40         String password = "";
41
42         boolean result = accountService.isEmailAndPasswordValid(email, password);
43
44         // Act
45         boolean oracolo = false;
46
47         // Assert
48         assertEquals(oracolo, result);
49     }
50
51     @Test
52     void testIsEmailAndPasswordValid_EmailValidPasswordValid_1() {
53         // Arrange
54         String email = "abc@def.it";
55         String password = "abc";
56
57         boolean result = accountService.isEmailAndPasswordValid(email, password);
58
59         // Act
60         boolean oracolo = true;
61
62         // Assert
63         assertEquals(oracolo, result);
```

```

64 }
65
66 @Test
67 void testIsEmailAndPasswordValid_EmailNotValidPasswordBlank() {
68     // Arrange
69     String email = "abc";
70     String password = "";
71
72     boolean result = accountService.isEmailAndPasswordValid(email, password);
73
74     // Act
75     boolean oracolo = false;
76
77     // Assert
78     assertEquals(oracolo, result);
79 }
80
81 // Test White-Box
82 @Test
83 void testIsEmailAndPasswordValid_EmailBlankPasswordNull() {
84     // Arrange
85     String email = "";
86     String password = null;
87
88     boolean result = accountService.isEmailAndPasswordValid(email, password);
89
90     // Act
91     boolean oracolo = false;
92
93     // Assert
94     assertEquals(oracolo, result);
95 }
96
97 @Test
98 void testIsEmailAndPasswordValid_EmailNotValidPasswordValid() {
99     // Arrange
100    String email = "abc";
101    String password = "abc";
102
103    boolean result = accountService.isEmailAndPasswordValid(email, password);
104
105    // Act
106    boolean oracolo = false;
107
108    // Assert
109    assertEquals(oracolo, result);
110 }
111
112 @Test
113 void testIsEmailAndPasswordValid_EmailValidPasswordNull() {
114     // Arrange
115     String email = "abc@def.it";
116     String password = null;
117
118     boolean result = accountService.isEmailAndPasswordValid(email, password);
119
120     // Act
121     boolean oracolo = false;
122
123     // Assert
124     assertEquals(oracolo, result);
125 }
126
127 @Test
128 void testIsEmailAndPasswordValid_EmailValidPasswordValid_2() {

```

```

129     // Arrange
130     String email = "ABc@CDe.A";
131     String password = "123";
132
133     boolean result = accountService.isEmailAndPasswordValid(email, password);
134
135     // Act
136     boolean oracolo = true;
137
138     // Assert
139     assertEquals(oracolo, result);
140   }
141 }
142

```

Listing 7.2: AccountServiceTests.java

7.2.3 III test: ModelProfilo validate (client)

Questo metodo verifica, quando si modifica il profilo, che i campi obbligatori non siano non validi.

La firma del metodo è la seguente:

```
fun validate(nome: String, cognome: String, dataNascita: LocalDate)
```

Dunque il metodo accetta tre parametri di input:

- **nome**, una stringa che rappresenta un nome;
- **cognome**, una stringa che rappresenta un cognome;
- **dataNascita**, che rappresenta una data.

Si individuino le classi di equivalenza secondo un approccio basato per funzionalità.

- nome:
 - **C.E. 1:** nome vuoto;
 - **C.E. 2:** nome compilato;
- cognome:
 - **C.E. 1:** cognome vuoto;
 - **C.E. 2:** cognome compilato;
- dataNascita:
 - **C.E. 1:** dataNascita = LocalDate.MIN;
 - **C.E. 2:** dataNascita != LocalDate.MIN;

Sceglieremo di fare testing **Black-Box** con criterio di copertura per la combinazione di valori di tipo **N-SECT** (Normal - Strong Equivalence Class Testing).

N-SECT prevede un totale di $2 * 2 * 2 = 8$ tests, ovvero il prodotto cartesiano tra tutte le classi di equivalenza. Si scelga un valore da ciascuna classe di equivalenza.

I test case sono:

- `validate("", "", LocalDate.MIN)`

- validate("Mario", "Rossi", LocalDate.of(1980,6,5))
- validate("", "Rossi", LocalDate.of(1980,6,5))
- validate("Mario", "", LocalDate.of(1980,6,5))
- validate("Mario", "Rossi", LocalDate.MIN)
- validate("", "", LocalDate.of(1980,6,5))
- validate("", "Rossi", LocalDate.MIN)
- validate("Mario", "", LocalDate.MIN)

Ossia:

```

1  @RunWith(AndroidJUnit4.class)
2  public class ModelProfiloTests {
3
4      private ModelProfilo viewModel;
5
6      @Before
7      public void setup() {
8          viewModel = new ModelProfilo();
9      }
10
11     @Test
12     public void test_TuttiMancanti() {
13         assertThrows(EccezioneCampiNonCompilati.class, () -> viewModel.validate("", ""));
14         // LocalDate.MIN));
15     }
16
17     @Test
18     public void test_TuttiCompilati() {
19         try {
20             viewModel.validate("Mario", "Rossi", LocalDate.of(1980, 6, 5));
21         } catch (EccezioneCampiNonCompilati e) {
22             fail();
23         }
24     }
25
26     @Test
27     public void test_NomeMancante() {
28         assertThrows(EccezioneCampiNonCompilati.class, () -> viewModel.validate("", "Rossi", LocalDate.of(1980, 6, 5)));
29     }
30
31     @Test
32     public void test_CognomeMancante() {
33         assertThrows(EccezioneCampiNonCompilati.class, () -> viewModel.validate("Mario", "", LocalDate.of(1980, 6, 5)));
34     }
35
36     @Test
37     public void test_DataNascitaMancante() {
38         assertThrows(EccezioneCampiNonCompilati.class, () -> viewModel.validate("Mario", "Rossi", LocalDate.MIN));
39     }
40
41     @Test
42     public void test_NomeECognomeMancanti() {
43         assertThrows(EccezioneCampiNonCompilati.class, () -> viewModel.validate("", "", LocalDate.of(1980, 6, 5)));
44     }

```

```

45     @Test
46     public void test_NomeEDataNascitaMancanti() {
47         assertThrows(EccezioneCampiNonCompilati.class, () -> viewModel.validate("",
48             "Rossi", LocalDate.MIN));
49     }
50
51     @Test
52     public void test_CognomeEDataNascitaMancanti() {
53         assertThrows(EccezioneCampiNonCompilati.class, () -> viewModel.validate("",
54             "Mario", "", LocalDate.MIN));
55     }
}

```

Listing 7.3: ModelProfiloTests.java

7.2.4 IV test: JWTTests getUserEmail (client)

Questo metodo estrae, quando è stato recuperato l'access token dall'authentication server di Cognito, l'email dell'utente che ha effettuato l'accesso o la registrazione.

Decodifica prima il JWT dalla base 64 e poi estrapola con l'aiuto di una espressione regolare e un gruppo di cattura il valore dell'email.

La firma del metodo è la seguente:

```
fun getUserEmail(jwt: String): String
```

Dunque il metodo restituisce l'email sotto forma di stringa e accetta un parametro di input:

- jwt, una stringa che contiene l'intero JWT recuperato da Cognito.

Si individuino le classi di equivalenza secondo un approccio basato per funzionalità.

- jwt:
 - **C.E. 1:** il JWT contiene l'email dell'utente;
 - **C.E. 2:** il JWT non contiene l'email dell'utente;
 - **C.E. 3:** il JWT è vuoto;

Consideriamo adesso tutti i valori che la stringa può assumere.

I test case sono:

- getUserEmail("eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InRlc3RAdGVzdC5jb20ifQ.sig")
- getUserEmail("eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIn0.sig")
- getUserEmail("")

Ossia:

```

1  @RunWith(AndroidJUnit4.class)
2  public class JWTTests {
3
4      @Test
5      public void test_emailPresente() {
6          String jwt = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
7          eyJlbWFpbCI6InRlc3RAdGVzdC5jb20ifQ.sig";

```

```
7     String email = JWT.Companion.getUserEmail(jwt);
8     assertEquals("test@test.com", email);
9 }
10
11 @Test
12 public void test_emailAssente() {
13     String jwt = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
14 eyJzdWIiOiIxMjM0NTY3ODkwIn0.sig";
15     String email = JWT.Companion.getUserEmail(jwt);
16     assertEquals("", email);
17 }
18
19 @Test
20 public void test_jwtVuoto() {
21     String jwt = "";
22     String email = JWT.Companion.getUserEmail(jwt);
23     assertEquals("", email);
24 }
25 }
```

Listing 7.4: JWTTests.java

CAPITOLO 8

Validazione del software

La fase di validazione del software fa parte dell'ultima fase del ciclo di vita del software. In particolare, l'obiettivo della *validazione del software* è quello di accertarsi che ciò che è stato implementato rispetti ciò che voleva il cliente, dunque che i requisiti esposti nell'analisi modellino esattamente ciò che il cliente desiderava. Per verificarlo, sono stati condotti studi di usabilità sul campo.

8.1 Alpha test

L'alpha test è la fase che consiste nella prova dell'applicazione da parte del team di sviluppo stesso. Attraverso la fase di alpha test, si sono scovati bug, piccoli problemi di progettazione e prestazioni dell'applicazione che non soddisfacevano le aspettative.

A seguito degli interventi, l'applicazione ha raggiunto i seguenti risultati: Come si può osservare dal

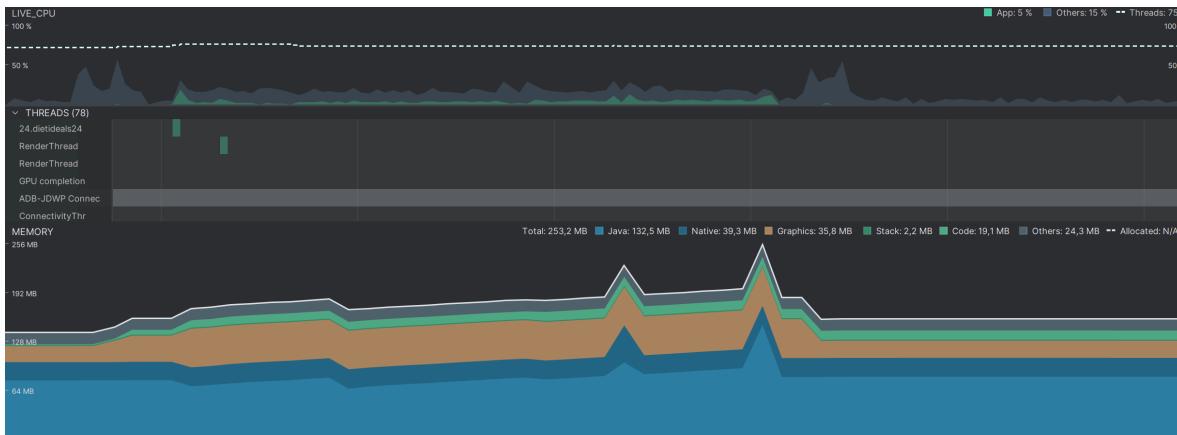


Figura 8.1: Utilizzo delle risorse dell'applicazione su client

seguente grafico, estratto dal Profiler di Android Studio, le prestazioni dell'applicazione sono modeste. In circa 30 secondi di prova, consistente nello spostamento consecutivo su più schermate con relative chiamate API da effettuare ed elaborare, l'applicazione ha occupato nel punto più concitato - ossia il recupero delle aste e delle notifiche - 253,2 MB della memoria del dispositivo, come osservabile dai picchi registrati nel grafico.

L'appiattimento del grafico, verso la fine, segue lo spegnimento dello schermo, confermando che l'applicazione "a riposo" consuma circa 150 MB.

Come si può osservare la maggioranza della memoria è occupata dall'allocazione di classi Java, mentre le classi Kotlin native, la UI e lo stack delle chiamate si dimostrano molto più modeste.

La CPU viene utilizzata appena al 5%, mostrando che le operazioni da effettuare siano poco pesanti - a dispetto della quantità di dati da elaborare - ed adatte al tipo di dispositivo target.

In quantità più elevata si possono osservare i thread generati dall'applicazione.

Occorre appuntare che l'applicazione fa ampio uso dei meccanismi di Coroutines di Kotlin per scaricare il lavoro di I/O dal thread dell'interfaccia e relegarla a thread specializzati chiamati Dispatchers, favorendo quindi la fluidità della UI e sfruttando a pieno le capacità di elaborazione concorrente dei

moderni processori mobili.

L'analisi non tiene conto del calo di utilizzo che si registrerebbe quando il sistema operativo Android inserisce l'app in modalità Doze - una modalità di sonno profondo dell'app attivata dopo un lungo periodo di inutilizzo dove le operazioni vengono interrotte e solo per brevi intervalli di tempo, distanziati tra loro, all'app viene permesso effettuare operazioni in background.

Si può, infine, osservare l'evoluzione dei tempi di avvio dell'applicazione Android nel corso dei precedenti 60 giorni ai test; attualmente, il valore si attesta a 789 millisecondi.



Figura 8.2: Grafico dei tempi di avvio dell'applicazione nel corso dello scorso mese

8.2 Beta test

Dopo aver raggiunto la maturità per una prova più estesa, l'app ha attraversato un periodo di beta test, dove è stata distribuita ad un campione ridotto di utilizzatori di sesso, età, occupazione e stato sociale differenti.

Il test si è svolto lasciando libera autonomia al campione, chiedendo di utilizzare l'applicazione come farebbe normalmente se utilizzasse un'applicazione per il proprio telefono e osservando le azioni effettuate.

Alla conclusione del test, è stato chiesto al campione di compilare un form per apprendere l'opinione generale e scoprire eventuali margini di miglioramento.

Il form, consultabile [qui](#) e realizzato con Google Forms, mira a conoscere le opinioni circa la sessione di utilizzo dell'applicazione nonché esperienze pregresse con applicazioni simili.

Dopo l'input delle risposte, i risultati sono stati inviati ad un file di Google Sheets collegato.

Essi sono i seguenti: I punteggi in percentuale di apprezzamento delle seguenti caratteristiche (che

	Soddisfazione /10	Feedback sistema/5	Intuitività interfaccia /5	Libertà di azione/5	Consistenza stile/5	Prevenzione errori/5	Facilità di memorizzazione /5	Scorciatoie/5
Utente 1	8	3	3	2	2	2	3	3
Utente 2	7	3	2	1	2	2	3	3
Utente 3	7	4	4	4	4	3	3	4
Utente 4	8	2	2	3	2	2	2	2
Utente 5	6	2	2	3	3	3	2	3
Utente 6	8	5	5	5	2	5	5	5

ricalcano le euristiche di Nielsen) scaturiti dal form sono:

- Soddisfazione: $\frac{44}{60} = 0,73... = 73\%$

	Piacevolezza visiva/5	Facilità di comprensione errori/5	Aiuti e documentazione/5	Familiarità con la tecnologia /10	Già usato app simili	App usate
Utente 1	3	1	3	10	Sì	eBay
Utente 2	2	2	3	4	No	
Utente 3	4	3	3	8	Sì	Amazon, Vinted, Temu
Utente 4	3	3	3	9	Sì	Amazon, eBay, Subito
Utente 5	2	2	2	7	No	
Utente 6	3	5	5	7	Sì	Vinted, eBay, Subito

- Feedback sistema: $\frac{19}{30} = 0,63 = 63\%$
- Intuitività interfaccia: $\frac{18}{30} = 0,6 = 60\%$
- Libertà di azione: $\frac{18}{30} = 0,6 = 60\%$
- Consistenza stile: $\frac{15}{30} = 0,5 = 50\%$
- Prevenzione errori: $\frac{17}{30} = 0,56... = 57\%$
- Facilità di memorizzazione: $\frac{18}{30} = 0,6 = 60\%$
- Scorciatoie: $\frac{20}{30} = 0,66... = 67\%$
- Piacevolezza visiva: $\frac{17}{30} = 0,56... = 57\%$
- Facilità di comprensione errori: $\frac{16}{30} = 0,53... = 53\%$
- Aiuti e documentazione: $\frac{19}{30} = 0,63 = 63\%$

8.3 Firebase Analytics

Grazie all'integrazione del frontend Android con Firebase Analytics, un servizio fornito da Google utilizzabile per ottenere delle metriche circa l'utilizzo e l'andamento dell'applicazione, si possono ricavare i seguenti dati:

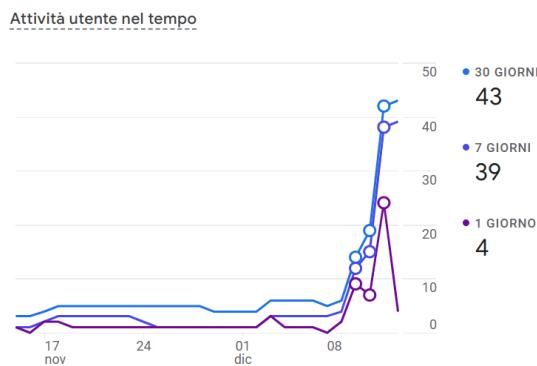


Figura 8.3: Grafico degli utenti attivi nel corso del tempo



Figura 8.4: Grafico del tempo di attività medio degli utenti



Figura 8.5: Grafico del numero di sessioni medio per ogni utente

Bibliografia

- [1] Amazon. *Amazon EC2 - Secure and resizable compute capacity*. URL: <https://aws.amazon.com/ec2/pricing>. (consultato: 26.02.2024).
- [2] Amazon. *Cloud Compute Capacity - Amazon EC2*. URL: <https://aws.amazon.com/ec2/>. (consultato: 26.02.2024).
- [3] Amazon. *Servizio di autenticazione - Customer IAM (CIAM) - Amazon Cognito - AWS*. URL: <https://aws.amazon.com/it/cognito>. (consultato: 15.10.2024).
- [4] Amazon. *What is Amazon Cognito? - Amazon Cognito*. URL: <https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html>. (consultato: 15.10.2024).
- [5] Amazon. *What is AWS? - Cloud Computing with AWS*. URL: <https://aws.amazon.com/what-is-aws/>. (consultato: 26.02.2024).
- [6] AppMaster. *Le strategie più efficaci per l'architettura esagonale in Java*. URL: <https://appmaster.io/it/blog/architettura-esagonale-java>. (consultato: 13.12.2024).
- [7] baeldung. *The DTO Pattern (Data Transfer Object)*. URL: <https://www.baeldung.com/java-dto-pattern>. (consultato: 13.12.2024).
- [8] Inc. Docker. *Docker overview*. URL: <https://docs.docker.com/get-started/overview/>. (consultato: 25.02.2024).
- [9] Wikimedia Foundation. *Amazon Elastic Compute Cloud*. URL: https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud. (consultato: 26.02.2024).
- [10] Wikimedia Foundation. *Android (operating system)*. URL: [https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system)). (consultato: 25.02.2024).
- [11] Wikimedia Foundation. *Docker (software)*. URL: [https://en.wikipedia.org/wiki/Docker_\(software\)](https://en.wikipedia.org/wiki/Docker_(software)). (consultato: 25.02.2024).
- [12] Wikimedia Foundation. *Kotlin (programming language)*. URL: [https://en.wikipedia.org/wiki/Kotlin_\(programming_language\)](https://en.wikipedia.org/wiki/Kotlin_(programming_language)). (consultato: 25.02.2024).
- [13] Wikimedia Foundation. *Spring Framework*. URL: https://en.wikipedia.org/wiki/Spring_Framework. (consultato: 25.02.2024).
- [14] Google. *Che cos'è Android*. URL: https://www.android.com/intl/it_it/what-is-android/. (consultato: 25.02.2024).
- [15] PostgreSQL Global Development Group. *PostgreSQL: About*. URL: <https://www.postgresql.org/about/>. (consultato: 25.02.2024).
- [16] Mark Haranas. *AWS Vs. Azure Vs. GCP: Flexera 2023 Customer Cloud Results*. URL: <https://www.crn.com/news/cloud/aws-vs-azure-vs-gcp-flexera-2023-customer-cloud-results>. (consultato: 26.02.2024).
- [17] Hevodata. *PostgreSQL vs Oracle: 6 Critical Differences*. URL: <https://hevodata.com/learn/postgresql-vs-oracle/>. (consultato: 25.02.2024).
- [18] IBM. *Cos'è Java Spring Boot?* URL: <https://www.ibm.com/it-it/topics/java-spring-boot>. (consultato: 13.12.2024).
- [19] IBM. *Cos'è un'API REST?* URL: <https://www.ibm.com/it-it/topics/rest-apis>. (consultato: 25.02.2024).
- [20] JetBrains. *Kotlin for Android*. URL: <https://kotlinlang.org/docs/android-overview.html>. (consultato: 25.02.2024).

- [21] JetBrains. *Null safety*. URL: <https://kotlinlang.org/docs/null-safety.html>. (consultato: 25.02.2024).
- [22] Lukasz Ryś. *Organizing Layers Using Hexagonal Architecture, DDD, and Spring*. URL: <https://www.baeldung.com/hexagonal-architecture-ddd-spring>. (consultato: 13.12.2024).
- [23] Spoclearn. *Java EE vs Spring*. URL: <https://www.spoclearn.com/blog/java-ee-vs-spring/>. (consultato: 25.02.2024).
- [24] Statcounter Global Stats. *Mobile Operating System Market Share Worldwide*. URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide>. (consultato: 25.02.2024).