



Umberto Micillo N86003237

Biagio Cirocco N86003344

Indice

1-Documento dei Requisiti Software

a.	Analisi dei requisiti	2-26
i.	Modellazione di tutti i casi d'uso individuati.	3-6
ii.	Individuazione del target degli utenti.	7-14
iii.	Descrizioni testuali strutturate dei casi d'uso.	15-17
iv.	Prototipazione visuale via Mock-up dell'interfaccia utente.	18-22
v.	Valutazione dell'usabilità a priori.	23-25
vi.	Glossario.	26
b.	Specifica dei Requisiti	27-43
i.	Classi, oggetti e relazioni di analisi... ..	27-40
ii.	Diagrammi di sequenza di analisi dei casi d'uso.	41
iii.	Prototipazione funzionale e progettazione degli event-based statechart dell'interfaccia grafica per i casi d'uso.	42-43

2-Documento di Design del sistema.

a.	Descrizione dell'architettura proposta	44
b.	Descrizione/ motivazione delle scelte tecnologiche adottate	45-46
i.	Il server.	45
ii.	Il client.	45
iii.	La persistenza dei dati.	46
c.	Diagramma delle classi di design	46-48
i.	0 vendor lock-in.	48
d.	Diagrammi di sequenza di design per i casi d'uso	48

3- Codice Sorgente sviluppato con Dockerfile49-50

a.	File di build automatico	49
b.	Uso di strumenti di versioning	49
c.	Report di qualità del codice, generati da SonarQube per il back-end	50

4-Testing e valutazione sul campo dell'usabilità51-72

a.	Codice xUnit per unit testing	51-66
b.	Valutazione dell'usabilità sul campo	67-72
c.	Production-ready product	72

1 Documento dei Requisiti Software

a. Analisi dei requisiti

Requisiti di sistema di interesse¹ che si evincono dal documento fornito per lo sviluppo del prodotto software:

1. Un utente può registrarsi al sistema con un account che può essere di tipo: compratore, venditore o entrambi. La stessa e-mail può essere utilizzata per al più un account venditore e un account compratore.
2. Il sistema deve permettere ad un utente di personalizzare il profilo con una short bio, link al proprio sito web / ai propri social e area geografica.
3. Un utente può effettuare una ricerca tra le aste per categoria o per parola chiave.
4. Un utente può visualizzare i dettagli dell'asta.
5. Un utente può visualizzare il profilo del proprietario dell'asta.
6. Un venditore può creare un'asta silenziosa e il compratore può inviarvi offerte segrete che il venditore sceglie se accettare o meno. Il venditore può accettare al più una sola offerta segreta per asta.
7. Un compratore può creare un'asta inversa specificando un prezzo iniziale. I venditori in grado di fornire il prodotto in questione partecipano all'asta facendo offerte al ribasso fino alla scadenza dell'asta. Dopodiché, il venditore con l'offerta più bassa si aggiudica la fornitura del prodotto.
8. Oltre ai dettagli dello specifico tipo di asta vanno aggiunte un titolo, una descrizione, una scadenza, una categoria e opzionalmente una fotografia per ogni asta creata.

1- Le funzionalità a noi assegnate: [1,2,3,7,8]

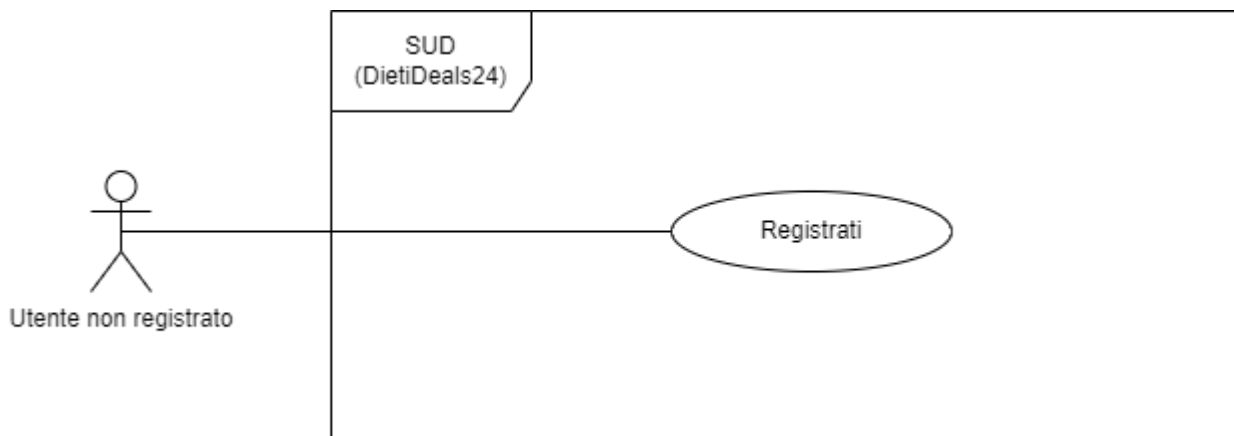
i. Modellazione di tutti i casi d'uso individuati

CASE tool utilizzato: drawio

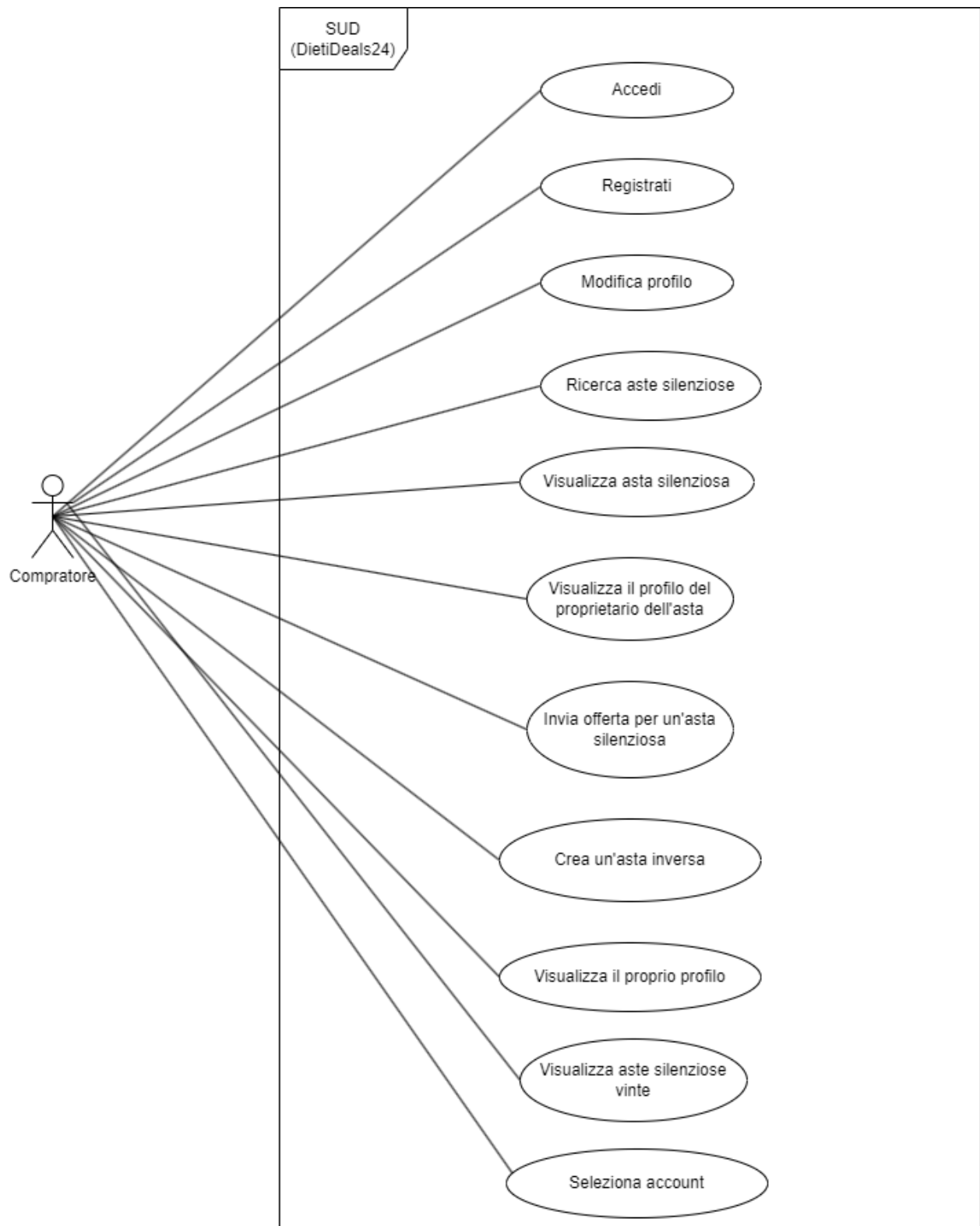
I seguenti use case illustrano i casi d'uso che sono stati identificati in seguito [all'analisi dei requisiti di sistema](#). Inoltre, includiamo alcuni casi d'uso proposti da noi, **compatibili con i requisiti esistenti**.

Per soddisfare al meglio il primo punto dell'analisi dei requisiti di sistema che abbiamo condotto, abbiamo deciso di implementare una funzionalità che permette all'utente di scegliere il tipo di account dopo aver completato la registrazione o l'accesso al sistema.

Utente non registrato:



Compratore:



Venditore:



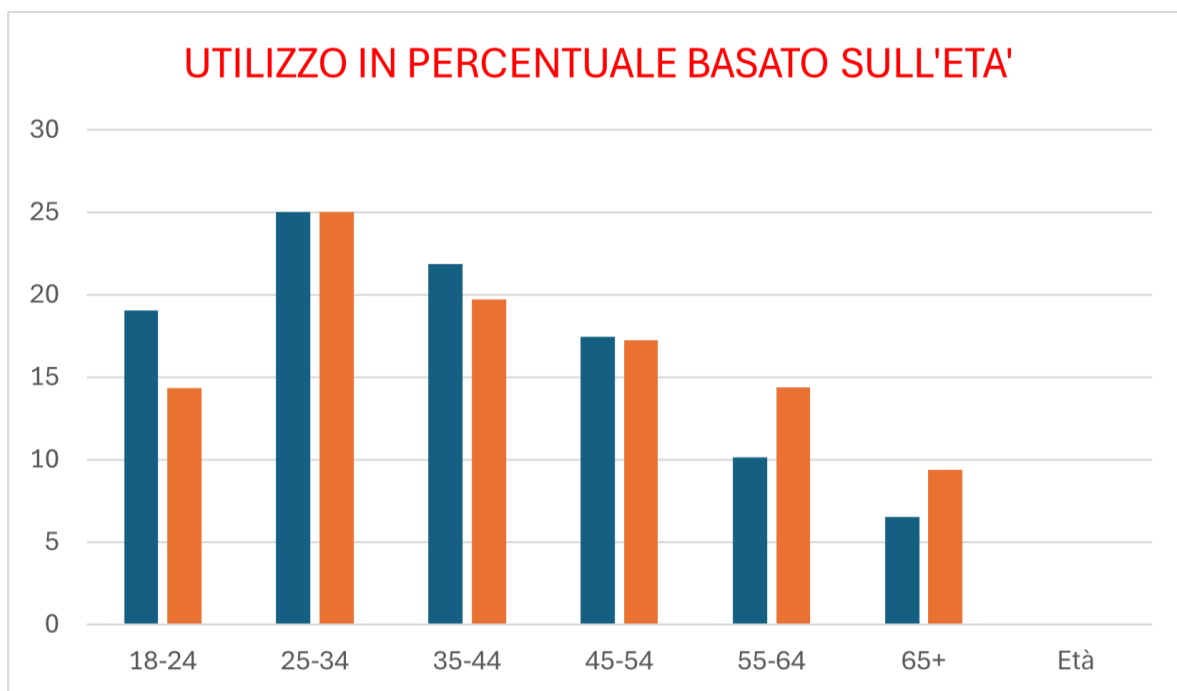
Unito:



ii. Individuazione del target degli utenti

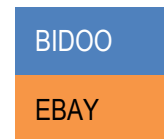
Per l'individuazione del target degli utenti abbiamo effettuato una ricerca su diversi siti e forum comprendendo:

- Chi sono le altre realtà che offrono la stessa tipologia di servizio.
- Chi è interessato al nostro prodotto.
- Quali sono le necessità/desiderata dei clienti per un'applicazione di questo tipo.
- Quali sono le categorie di tendenza riconducibili ad una certa fascia di utenti.



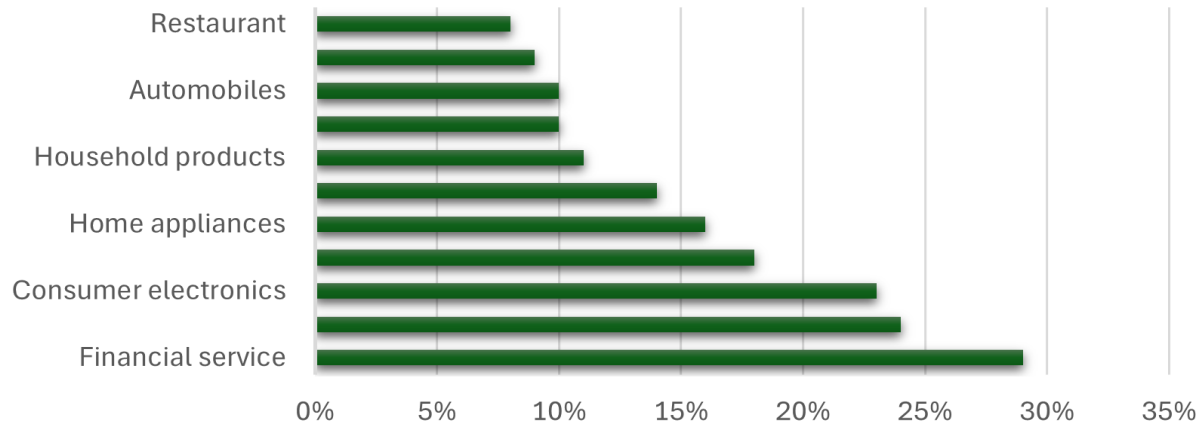
DATI RACCOLTI DAL SITO: <http://www.similarweb.com/>

LEGENDA



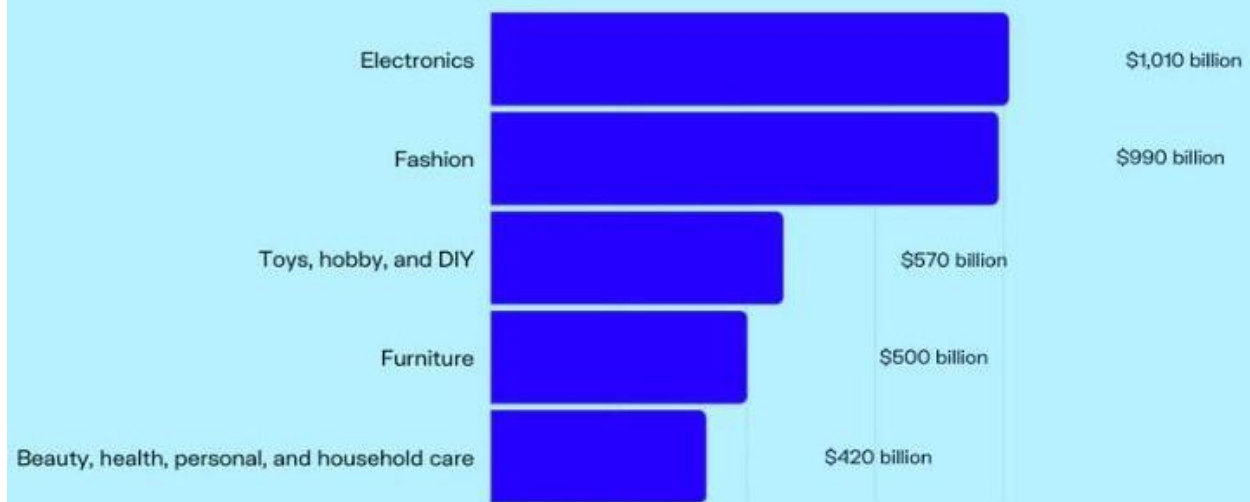
I dati raccolti in questo grafico rappresentano la percentuale di utilizzo di applicazioni web che gestiscono aste online per una determinata fascia di età degli utenti. Bidoo e Ebay sono molto utilizzati in tutto il mondo e permettono di partecipare a delle aste per un'infinità di risorse o servizi.

THE MOST POPULAR OMNI-SHOPPERS ONLINE SHOPPING BACK IN 2017



I dati raccolti in questo grafico rappresentano in percentuale quali erano le categorie di oggetti o servizi maggiormente acquistati online nel 2017.

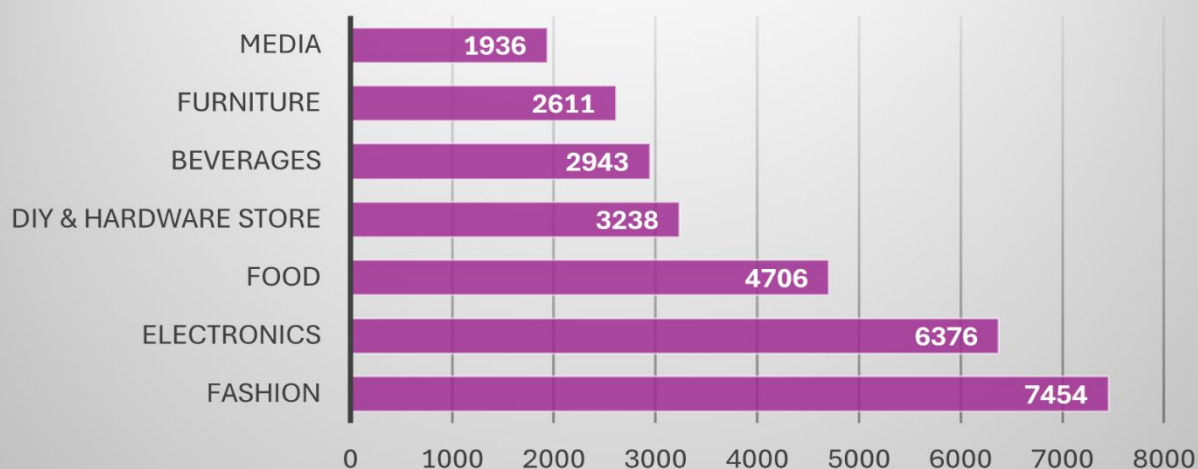
Top Online Shopping Categories Worldwide in 2022



Mentre qui possiamo vedere quali sono le categorie più redditizie nel 2022 per quanto riguarda lo shopping online.

DATI RACCOLTI DAL SITO: www.digitalmarketingcommunity.com

Top Online Shopping Categories Worldwide in 2024









Come il grafico precedente, anche qui abbiamo le categorie più redditizie nel 2024 per quanto riguarda lo shopping online. Siamo partiti dal 2017 per comprendere meglio come tale realtà si è evoluta nel tempo. Come si evince dai grafici le categorie più redditizie e gettonate sono moda, elettronica e immobili.

A valle di quest'analisi abbiamo annotato le categorie e le fasce di età di interesse.







DATI RACCOLTI DAL SITO: <http://www.shopify.com/>







PERSONAS


Per la creazione delle personas è stato utilizzato il sito: <http://www.hubspot.com/>

	Preferred Method of Communication <ul style="list-style-type: none"> • Phone • Email • Face-To-face 	Tools They Need to Do Their Job <ul style="list-style-type: none"> • Assisted design software • Email • Planning and layout tools • Catalogs of materials and furnishings • Measuring instruments • Online resources
Name CHIARA	Job Responsibilities Communication with customers and suppliers, development of technical drawings, development of the design concept	Their Job Is Measured By Customer satisfaction, compliance with budget and respecting deadlines, quality of execution
Job Title Interior designer	Reports to Private customers and project leader or chief designer	Goals or Objectives Satisfy customer needs, create functional and aesthetically pleasing spaces,
Age 55 to 64 years	They Gain Information By Consultations with clients, search for inspiration and trends, search for inspiration and trends, study of previous projects, practical experience	Biggest Challenges <ul style="list-style-type: none"> • Effective communication with customers • Management of budget and time constraints • Reconciling functional and aesthetic needs • Integrate sustainability • Managing customer expectations
Highest Level of Education Doctorate (e.g. PhD, EdD)		
Social Networks     		
Industry Real Estate		
Organization Size 1-10 employees		

	Preferred Method of Communication <ul style="list-style-type: none"> • Phone • Email • Face-To-face 	Tools They Need to Do Their Job <ul style="list-style-type: none"> • Magnifying glasses • Email • Diagnostic tools • Tools for gentle removal • Restoration materials • Documentation and data management software
Name GENNARO	Job Responsibilities Analysis and evaluation, carrying out the restoration, conservation and preservation, communication with the client or employer	Their Job Is Measured By Quality of the restoration, customer satisfaction, recognition and reputation in the industry
Job Title Artist, restorer	Reports to Private customer or owners of works of art	Goals or Objectives Restoration of the original appearance, customer satisfaction, conservation of cultural heritage
Age 25 to 34 years	They Gain Information By Study and training, practical experience, direct examination of works of art, collaboration with colleagues and experts	Biggest Challenges <ul style="list-style-type: none"> • Damage and deterioration of works of art • Resources • Collaboration & Creativity • Technology and equipment • Criticisms and evaluations • Conservation of original materials
Highest Level of Education Master's degree (e.g. MA, MFA)		
Social Networks     		
Industry Manufacturing		
Organization Size Self-employed		

	Preferred Method of Communication <ul style="list-style-type: none"> • Phone • Email • Face-To-face • Text Messaging • Social Media 	Tools They Need to Do Their Job <ul style="list-style-type: none"> • Email • Books and teaching materials
Name ROBERTA		
Job Title Undergraduate		
Age 18 to 24 years		
Highest Level of Education High school degree or equi		Job Responsibilities Study
Social Networks <div>      </div>	Reports to University professors	
They Gain Information By Going to class or contacting the professor		

	Preferred Method of Communication <ul style="list-style-type: none"> • Phone • Email • Face-To-face 	Tools They Need to Do Their Job <ul style="list-style-type: none"> • Email • Cloud-Based Storage & File Sharing Applications • Reporting Software
Name MARCO	Job Responsibilities Organize and interpret data in order to facilitate the work of his subordinates	Their Job Is Measured By Collect, refine and interpret data sets, so that we can solve various types of problems and assist the organization of the company.
Job Title Data analyst	Reports to Your superiors (for example, head of the company)	Goals or Objectives Earn money with his work and rise to the top of the company
Age 45 to 54 years	They Gain Information By Analysis of existing data, collection of new data and data analysis tools	Biggest Challenges <ul style="list-style-type: none"> • Resources • Communication • Collaboration & Creativity • Project Management & Disorganization
Highest Level of Education Master's degree (e.g. MA, M		
Social Networks <div>      </div>		
Industry Technology		
Organization Size 201-500 employees		

	<h3>Preferred Method of Communication</h3> <ul style="list-style-type: none"> • Phone • Email • Text Messaging • Face-To-face 	<h3>Tools They Need to Do Their Job</h3> <ul style="list-style-type: none"> • Email • Cloud-Based Storage & File Sharing Applications • Database or cataloging software • Filing and preservation tools • Tools for assessment and authentication
<h4>Name</h4> <p>SOFIA</p>		
<h4>Job Title</h4> <p>Collector</p>		
<h4>Age</h4> <p>35 to 44 years</p>		
<h4>Highest Level of Education</h4> <p>Bachelor's degree (e.g. BA,</p>	<h3>Job Responsibilities</h3> <p>Balance between passion and investment, search and identification, conservation and care of objects, cataloging and organization, insurance and risk management</p>	<h3>Their Job Is Measured By</h3> <p>Rarity and value of objects in the collection, financial investment</p>
<h3>Social Networks</h3> 		
<h4>Industry</h4> <p>Manufacturing</p>		
<h4>Organization Size</h4> <p>Self-employed</p>		
<h3>Reports to</h3> <p>Enter text here</p>	<h3>Goals or Objectives</h3> <p>Honors and awards</p>	
<h3>They Gain Information By</h3> <p>Online search, books and specialized publications, participation in collecting events, specialized catalogs and databases, personal experience</p>	<h3>Biggest Challenges</h3> <ul style="list-style-type: none"> • Management of financial resources • Resources • Conservation and maintenance of objects • Authenticity and counterfeiting 	

iii. Descrizioni testuali strutturate dei casi d'uso

Al fine di offrire una descrizione testuale strutturata dei casi d'uso abbiamo scelto di utilizzare il template di Alistair Cockburn.

I casi d'uso da noi scelti sono:

- Un compratore che vuole visualizzare il profilo del proprietario dell'asta.
- Un venditore che vuole creare un'asta silenziosa.

USE CASE #1	<i>VISUALIZZA IL PROFILO DEL PROPRIETARIO DELL'ASTA</i>		
Goal	Il compratore vuole visualizzare il profilo del proprietario dell'asta.		
Preconditions	L'utente deve aver effettuato correttamente il login selezionando l'account compratore.		
Success End Condition	Il compratore visualizza correttamente il profilo del proprietario dell'asta.		
Primary Actor	Compratore		
DESCRIPTION	Step n°	Attore Compratore	Sistema
	1	Preme sull'asta di interesse sul Mock-up home_dal_compratore(19)	
	2		Mostra Mock-up visualizza_asta_silenziosa_dal_compratore (8)
	3	Preme sul nome del proprietario dell'asta	
	4		Mostra il Mock-up profilo_creatore_asta (18) e termina use case
EXTENSIONS	Step n°	Attore Compratore	Sistema

Preme la freccia per tornare indietro sul Mock-up visualizza_asta_silenziosa_dal compratore (8)	3.1	Preme la freccia per tornare indietro	
	4.1		Mostra il Mock-up home_dal_compratore (19) e continua dallo step 1 del main scenario

USE CASE #2	CREA UN'ASTA SILENZIOSA		
Goal	Il venditore vuole creare un'asta silenziosa.		
Preconditions	L'utente deve aver effettuato correttamente il login selezionando l'account venditore.		
Success End Condition	Il venditore crea correttamente un'asta silenziosa.		
Primary Actor	Venditore		
DESCRIPTION	Step n°	Attore Venditore	Sistema
	1	Preme il pulsante "+" del navbar sottostante, presente sul Mock-up home_dal_venditore (5)	
	2		Mostra il Mock-up aggiungi_nuova_asta_silenziosa (10) con data di scadenza maggiore del giorno odierno e categorie caricate.
	3	Inserisce un'eventuale immagine	
	4	Inserisce il titolo dell'asta	

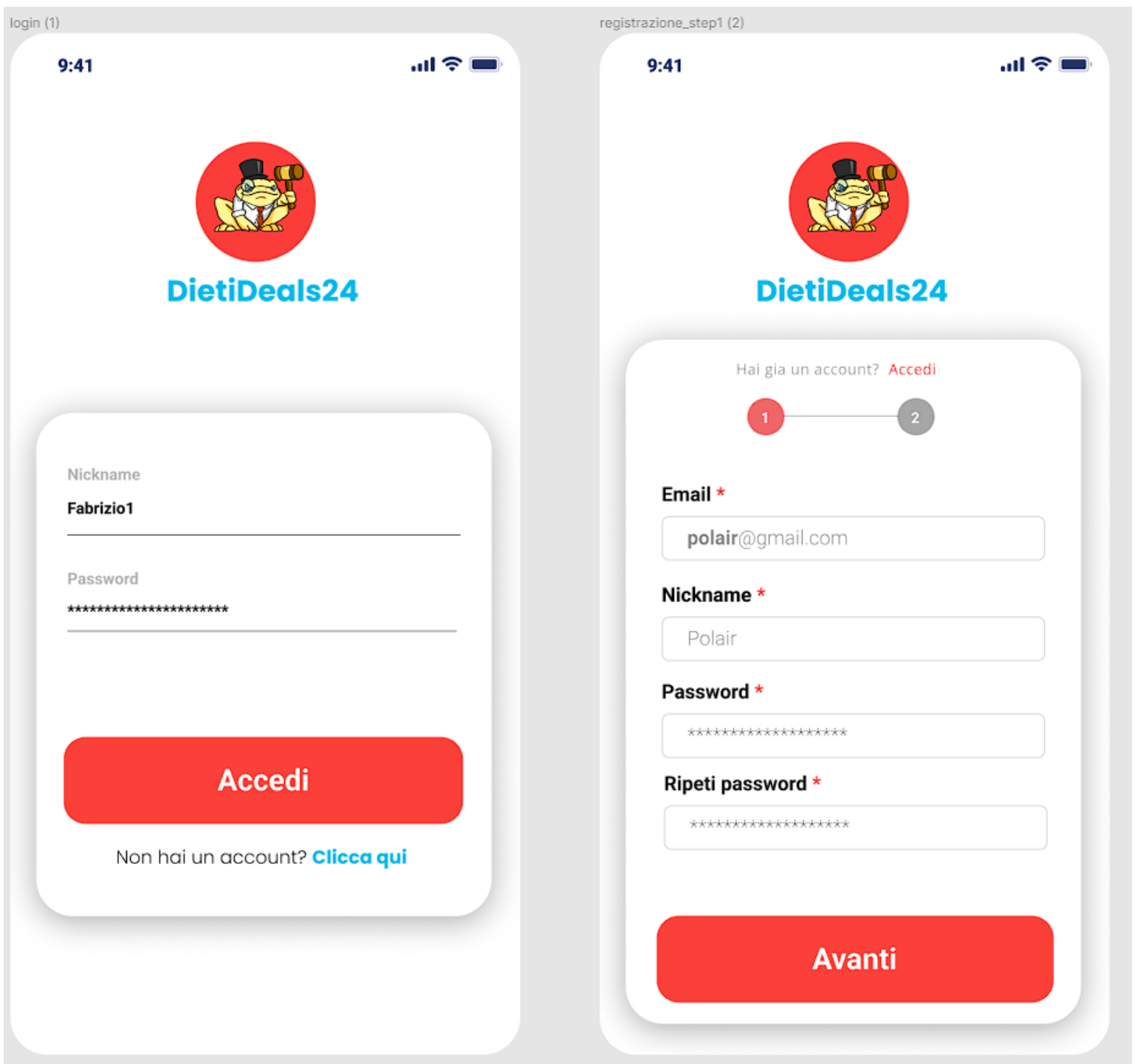
	5	Seleziona la categoria di appartenenza della risorsa in questione	
	6	Inserisce la descrizione	
	7	Inserisce la data di scadenza dell'asta	
	8	Preme il bottone "Crea"	
	9		Mostra il Mock-up visualizza_asta_silenziosa_dal_venditore (13) e termina use case.
EXTENSIONS	Step n°	Attore Venditore	Sistema
Preme la freccia per tornare indietro nel Mock-up aggiungi_nuova_asta_silenziosa (10)	In uno step da 3 a 8	Preme la freccia per tornare indietro	
			Mostra Mock-up home_dal_venditore (5) e continua dallo step 1 del main scenario
Il venditore non compila uno o più campi obbligatori	In uno step da 4 a 8		Mostra messaggio di errore live al venditore e riprendi dallo step 4 del main scenario

iv. Prototipazione visuale via Mock-up dell'interfaccia utente


CASE tool utilizzato: Figma

Abbiamo scelto Figma per realizzare i mock-up data la sua semplicità nell'utilizzo ma soprattutto per il suo supporto alla **collaborazione in tempo reale** via cloud che ha permesso di velocizzare il processo di prototipazione.

[Qui](#) il link del progetto.




9:41


**DietiDeals24**

Hai già un account? [Accedi](#)

1 — 2

Data di nascita *
 


Sesso *
☒ **Maschio** ☐ **Femmina**

Area geografica *
 


Biografia


Registrati

9:41

**DietiDeals24**



Seleziona account



Venditore


Compratore


[→ Esci](#)


9:41

 **24 Aste attive!** **DietiDeals24**


 [Filtra ▼](#)


Elettronica







MacBook 15Spro, 15GB RAM....
🕒 22/02/2024
🏷️ partenza 25,32 EUR


Android Galaxy S21 Ultra...
🕒 20/02/2024
🏷️ partenza 134,32 EUR

Per casa


Letto 2 piazze, comodo e
🕒 01/01/2024
🏷️ partenza 67,00 EUR


Set tavolo + sedie...
🕒 20/02/2024
🏷️ partenza 22,32 EUR



    


filtro (16)

Categoria


- ☒ Per Casa
- ☒ Elettronica
- ☐ Sport


9:41

 **24 Aste attive!** **DietiDeals24**


 [Filtra ▼](#)


Elettronica







Quadro Tulerean SIA in tela...
🕒 22/02/2024
👤 Dinaya6


Android Galaxy S21 Ultra...
🕒 20/02/2024
👤 Minniva88

Per casa


Letto 2 piazze, comodo e
🕒 01/01/2024
👤 Salemmè

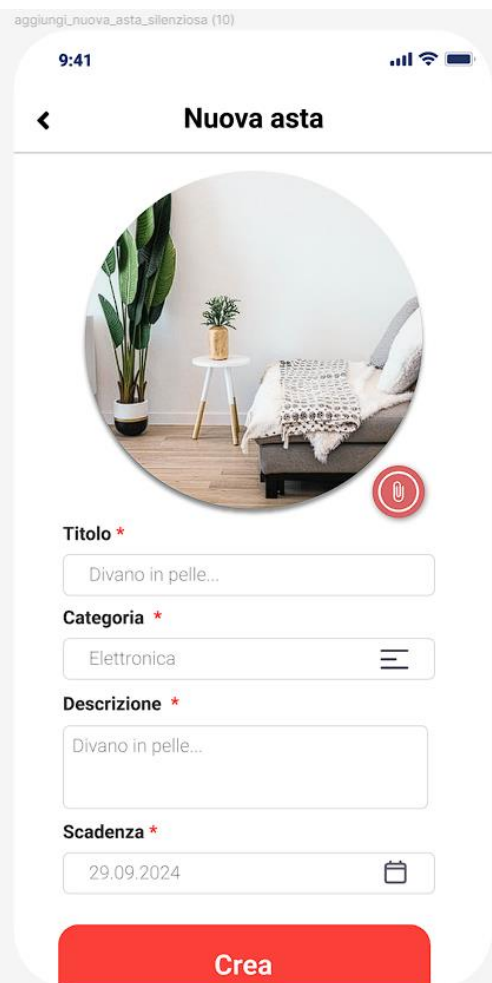
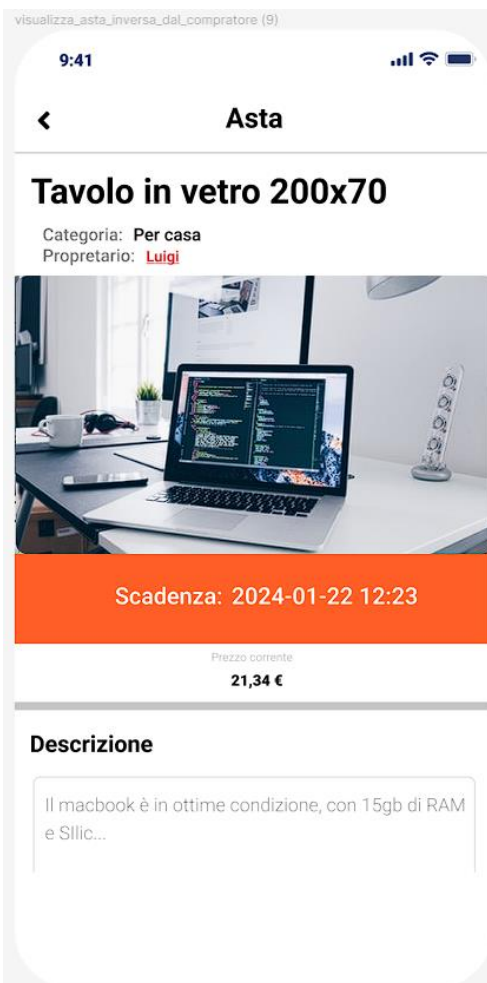
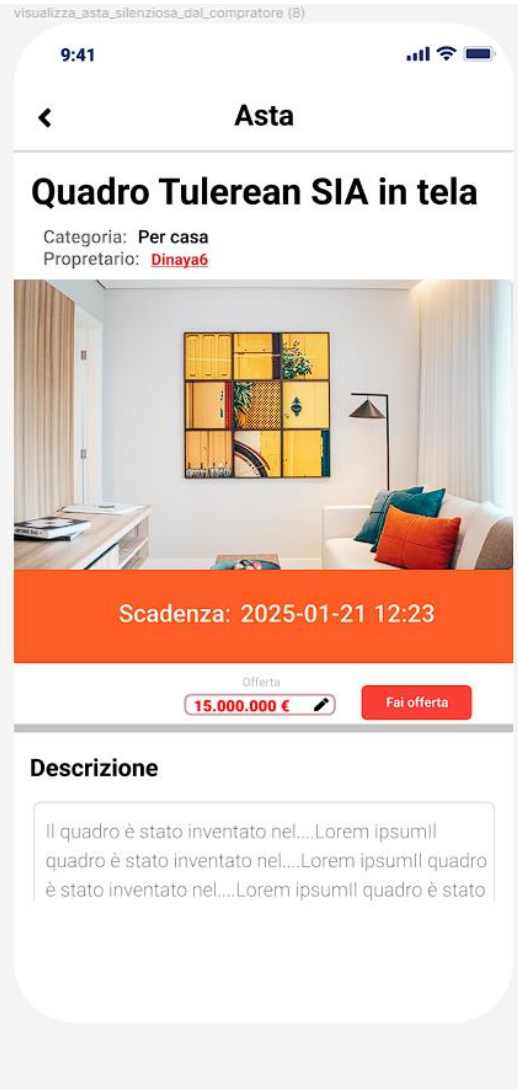

Set tavolo + sedie...
🕒 20/02/2024
👤 Giustino1

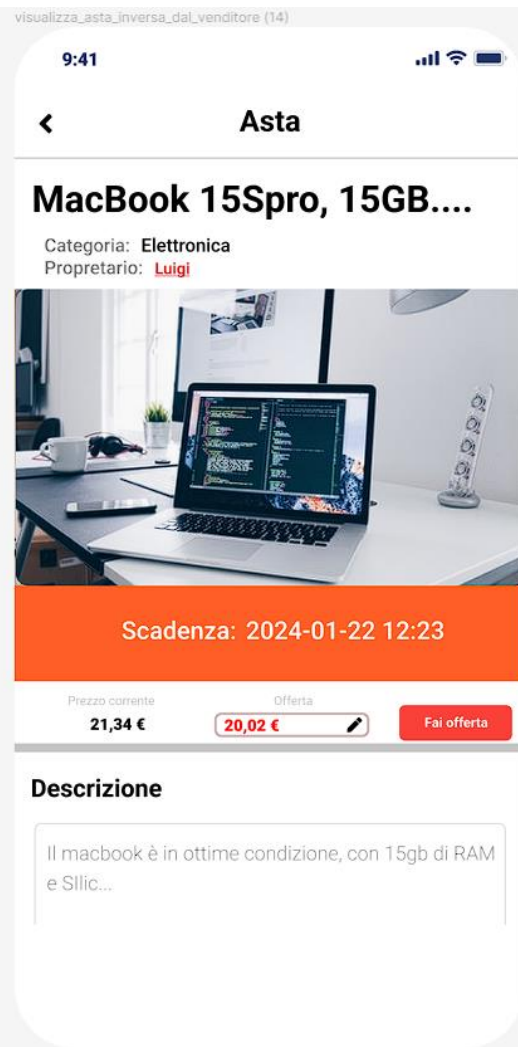
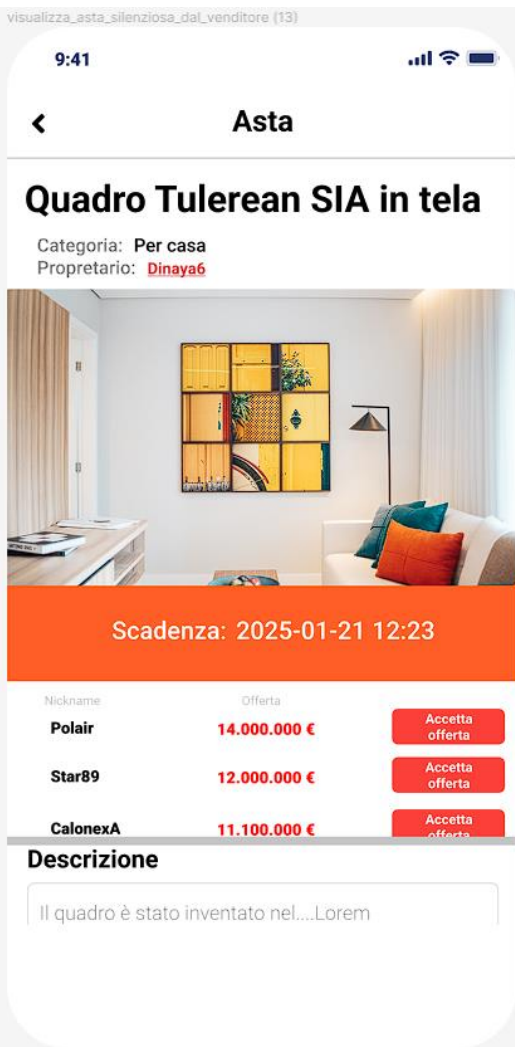
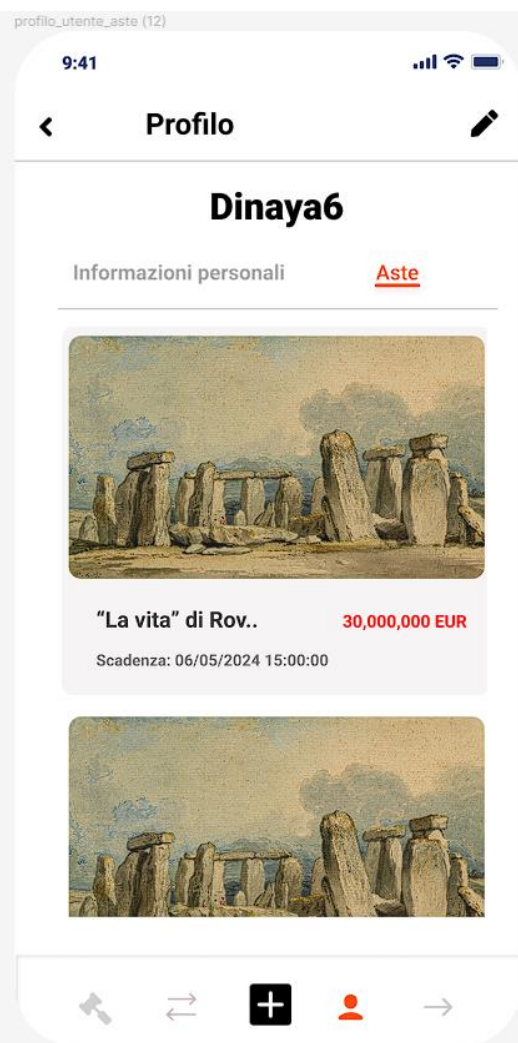
    

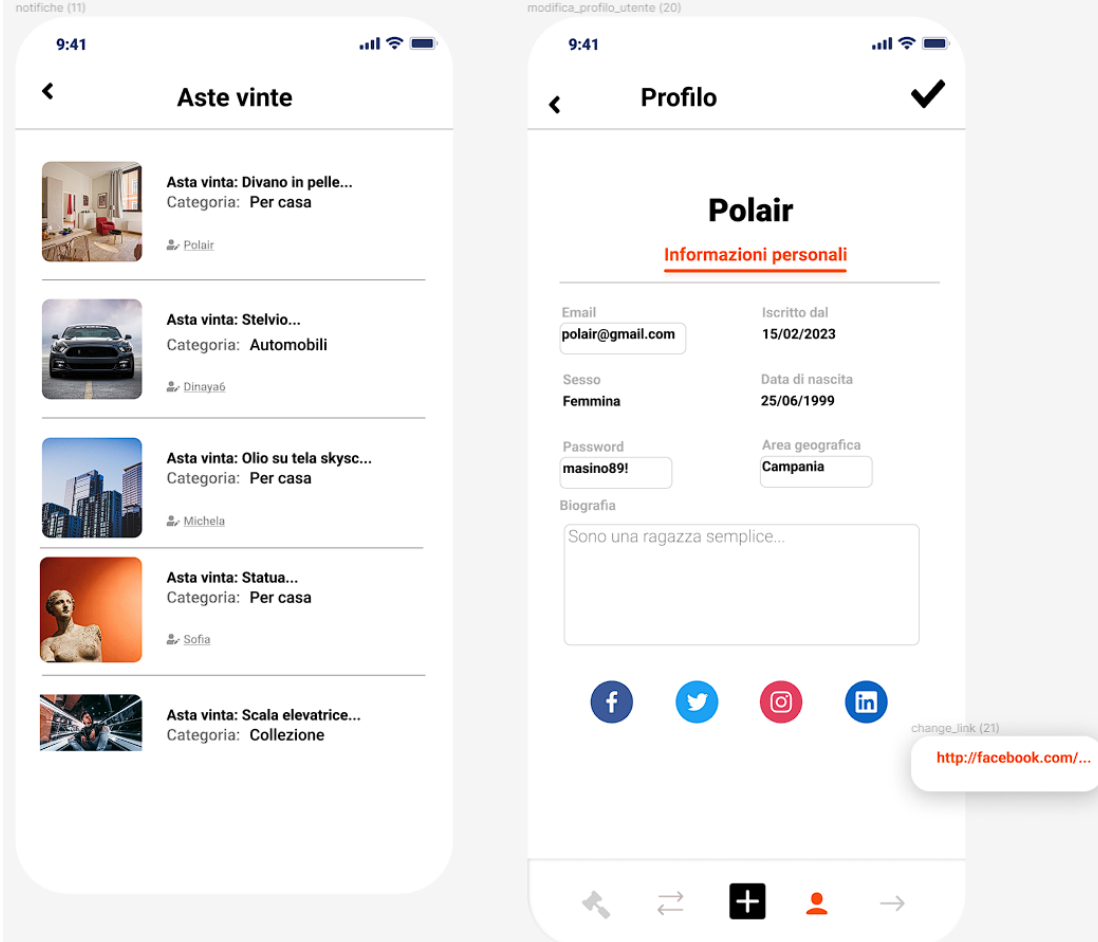
filtro (22)

Categoria

- ☒ Per Casa
- ☒ Elettronica
- ☐ Sport








aggiungi_nuova_asta_inversa (15)

9:41

Nuova asta



Titolo *

Divano in pelle...

Categoria *

Elettronica

Descrizione *

Divano in pelle...

Scadenza *

29.09.2024

Prezzo di partenza *

156,55 €

Crea

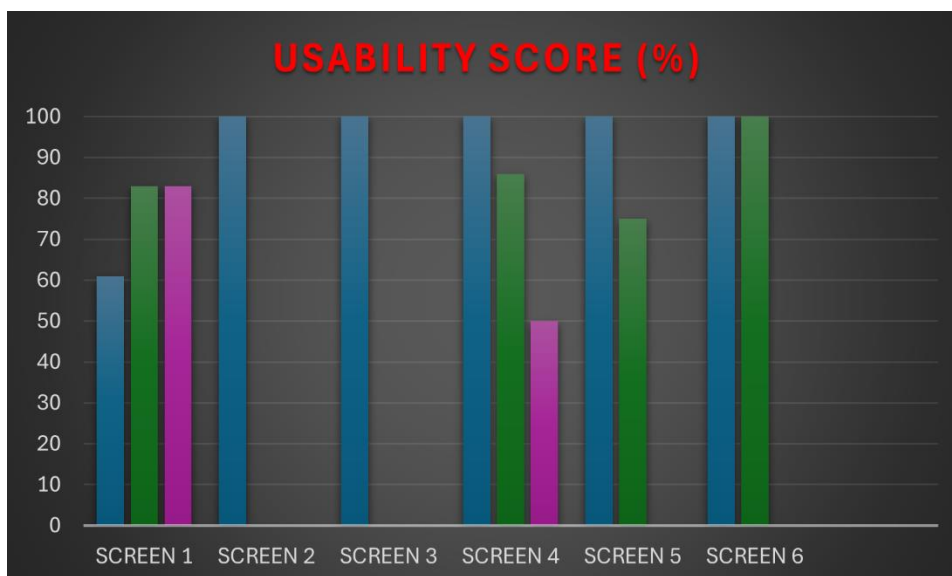
v. Valutazione dell'usabilità a priori

Per completare questa fase, abbiamo coinvolto alcuni amici e conoscenti, invitandoli a diventare tester. Questi hanno avuto l'opportunità di provare il prototipo della nostra app, che è stato importato da FIGMA in Maze, una piattaforma specializzata nel testing di usabilità e user experience. Il nostro obiettivo era che i tester eseguissero due specifici use case, fornendo loro indicazioni su come navigare all'interno del prototipo.

- [Visualizza il profilo del proprietario dell'asta](#)
- [Crea un'asta silenziosa](#)

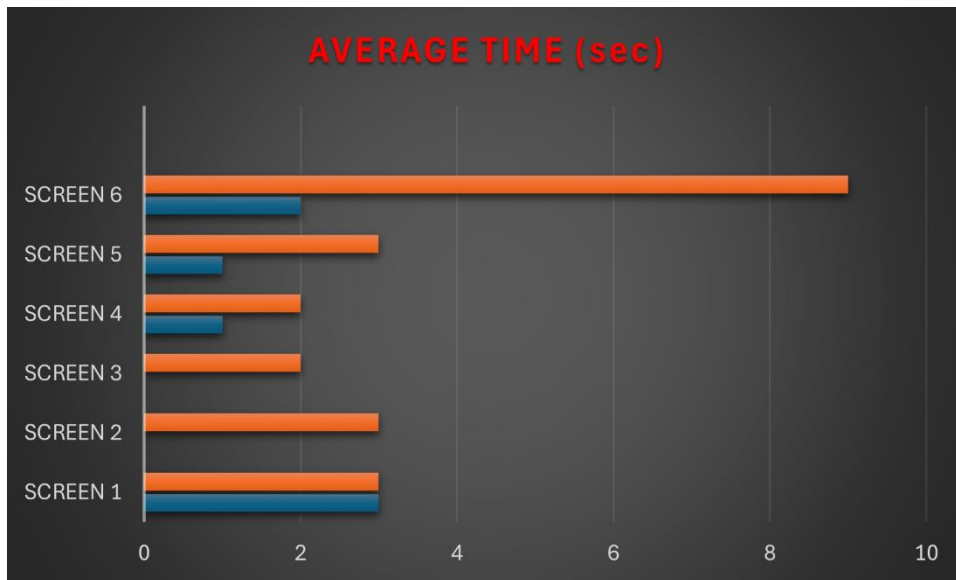
VISUALIZZA IL PROFILO DEL PROPRIETARIO DELL'ASTA

Per visualizzare i path e la mappatura tra screen e mock-up cliccare [qui](#)



CREA UN'ASTA SILENZIOSA

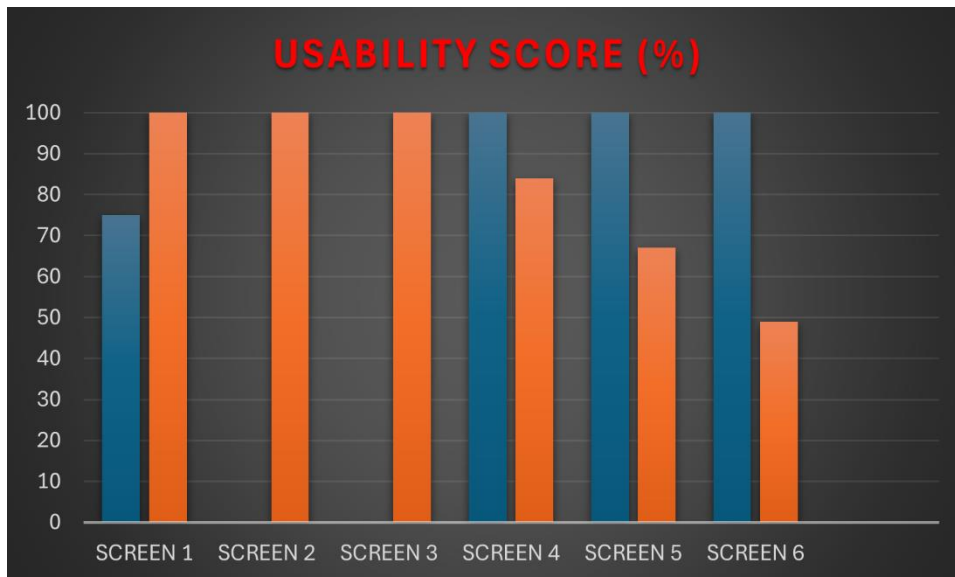
Per visualizzare i path e la mappatura tra screen e mock-up cliccare [qui](#)



LEGENDA

PATH 1

PATH 2



I risultati dei test sono molto promettenti, ma ci sono alcune schermate che presentano dati negativi. Questi risultati sono principalmente dovuti a errori di click in schermate che richiedevano la compilazione di campi, nonostante avessimo specificato chiaramente che i campi non erano compilabili. Alcuni tester hanno comunque provato a cliccare su tali campi, compromettendo leggermente la media del tempo e lo score di usabilità per alcune schermate.

Le due heatmap seguenti, infatti, mostrano le schermate peggiori:

[VISUALIZZA PROFILO CREATORE ASTA]: HEATMAP SCREEN 1



[CREA UN'ASTA SILENZIOSA]: HEATMAP SCREEN 6



vi. Glossario

PAROLE	SIGNIFICATO
VENDITORE	Utente del sistema che può mettere all'asta risorse/servizi per i compratori creando delle aste silenziose. Inoltre, può fare offerte al ribasso nelle aste inverse per aggiudicarsi la fornitura del bene/servizio offerto dall'asta.
COMPRATORE	Utente del sistema che può fare offerte segrete per risorse/servizi nelle aste silenziose. Inoltre, può creare aste inverse dove specifica un bene/servizio che sta cercando e riceverà offerte al ribasso dai venditori che possono procurarglielo.
ASTA SILENZIOSA	Tipo di asta creata dal venditore dove i compratori possono inviare offerte segrete. Una sola offerta segreta per asta silenziosa può essere accettata dal venditore.
ASTA INVERSA	Tipo di asta in cui il compratore specifica il bene/servizio richiesto con un prezzo iniziale. I venditori interessati possono fare offerte al ribasso. Al momento della scadenza il venditore con l'offerta più bassa si aggiudica l'asta.
OFFERTA AL RIBASSO	Si riferisce a un tipo di offerta proposta durante un'asta inversa, caratterizzata da un valore che è inferiore rispetto all'ultima offerta che è stata presentata.
OFFERTA SEGRETA	Tipo di offerta per un'asta silenziosa. Questa viene effettuata senza conoscere il valore corrente dell'asta. Il venditore può decidere se accettarla o meno.

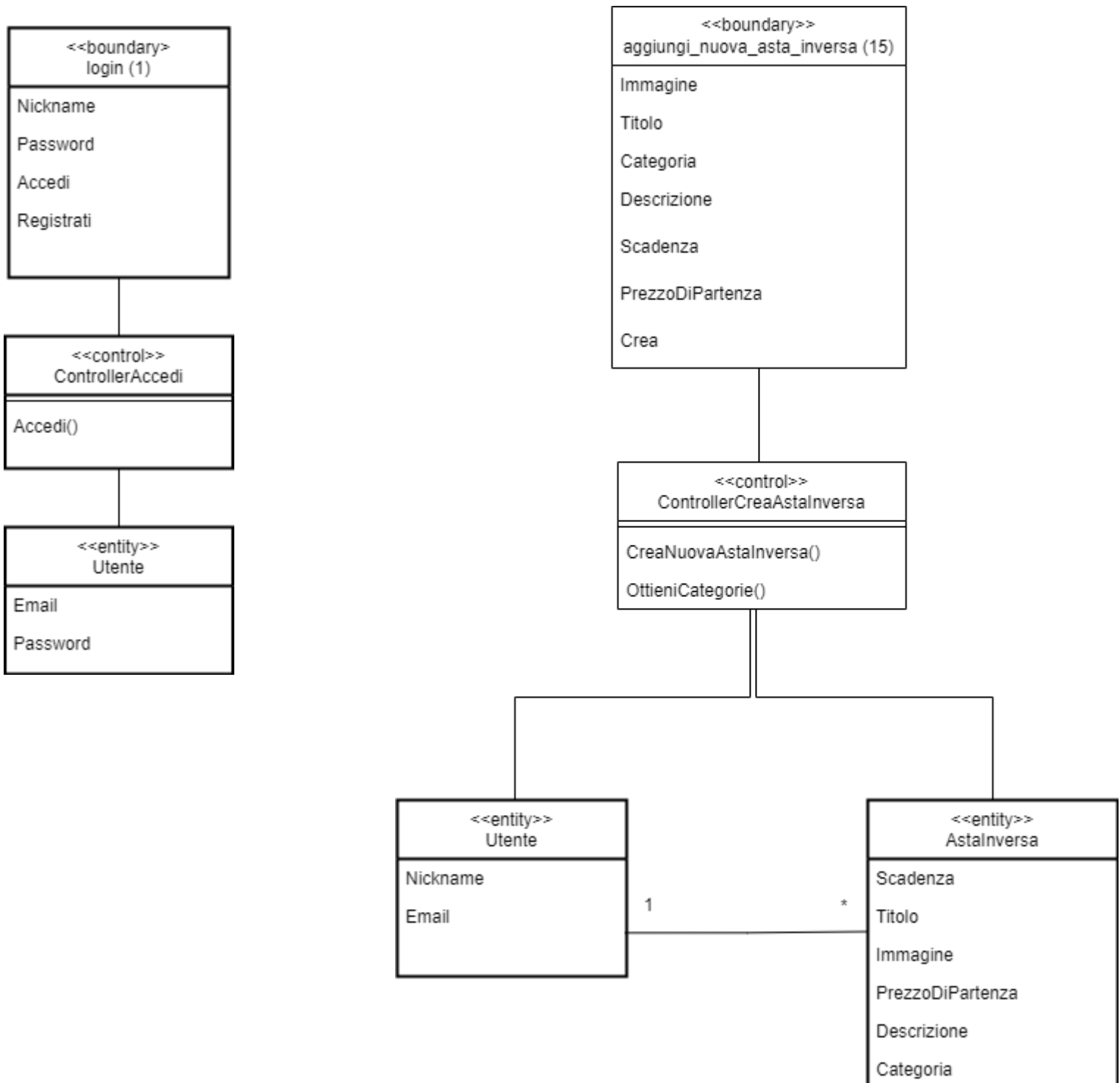
b. Specifica dei requisiti

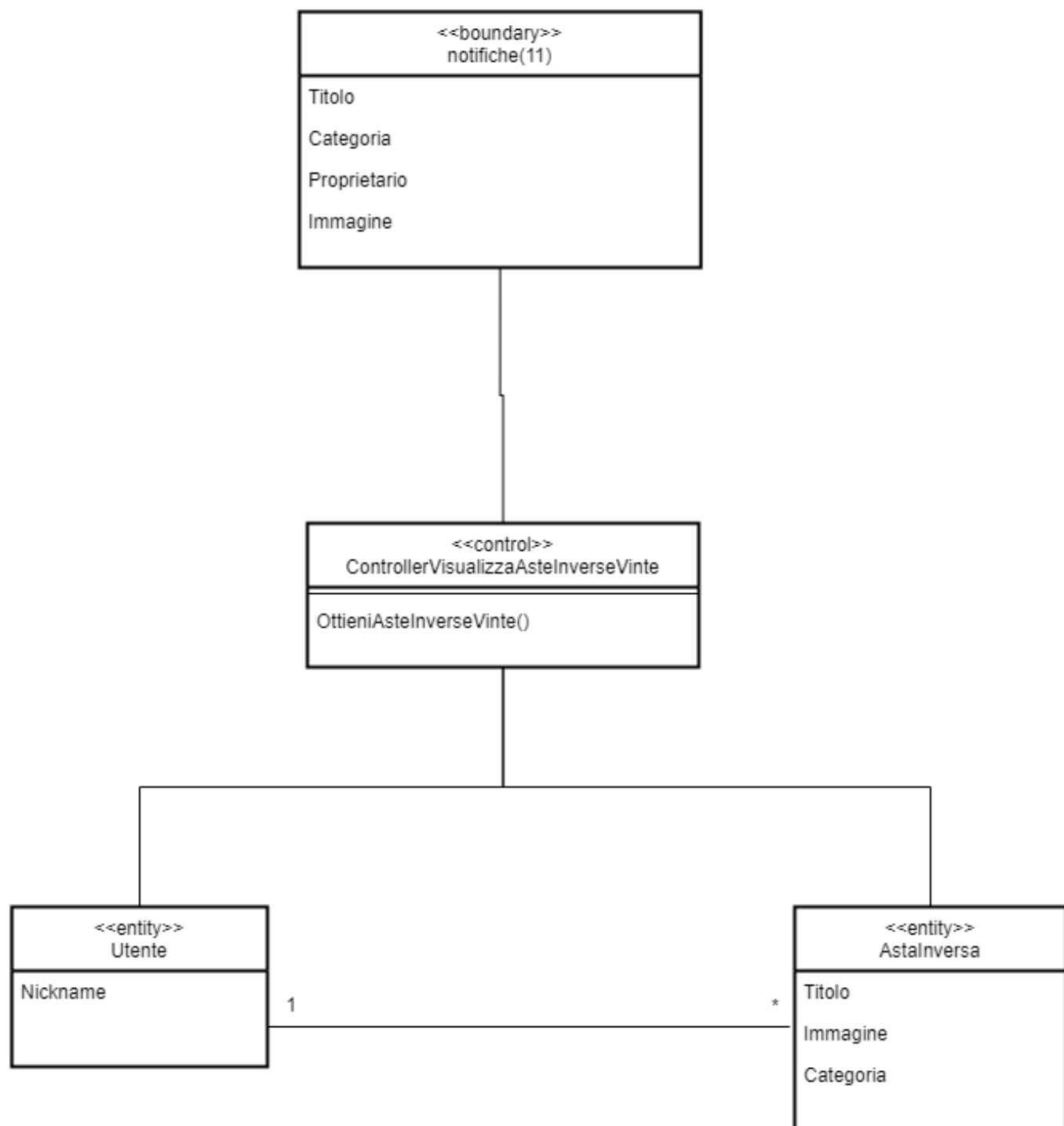
i. Classi oggetti e relazioni di analisi

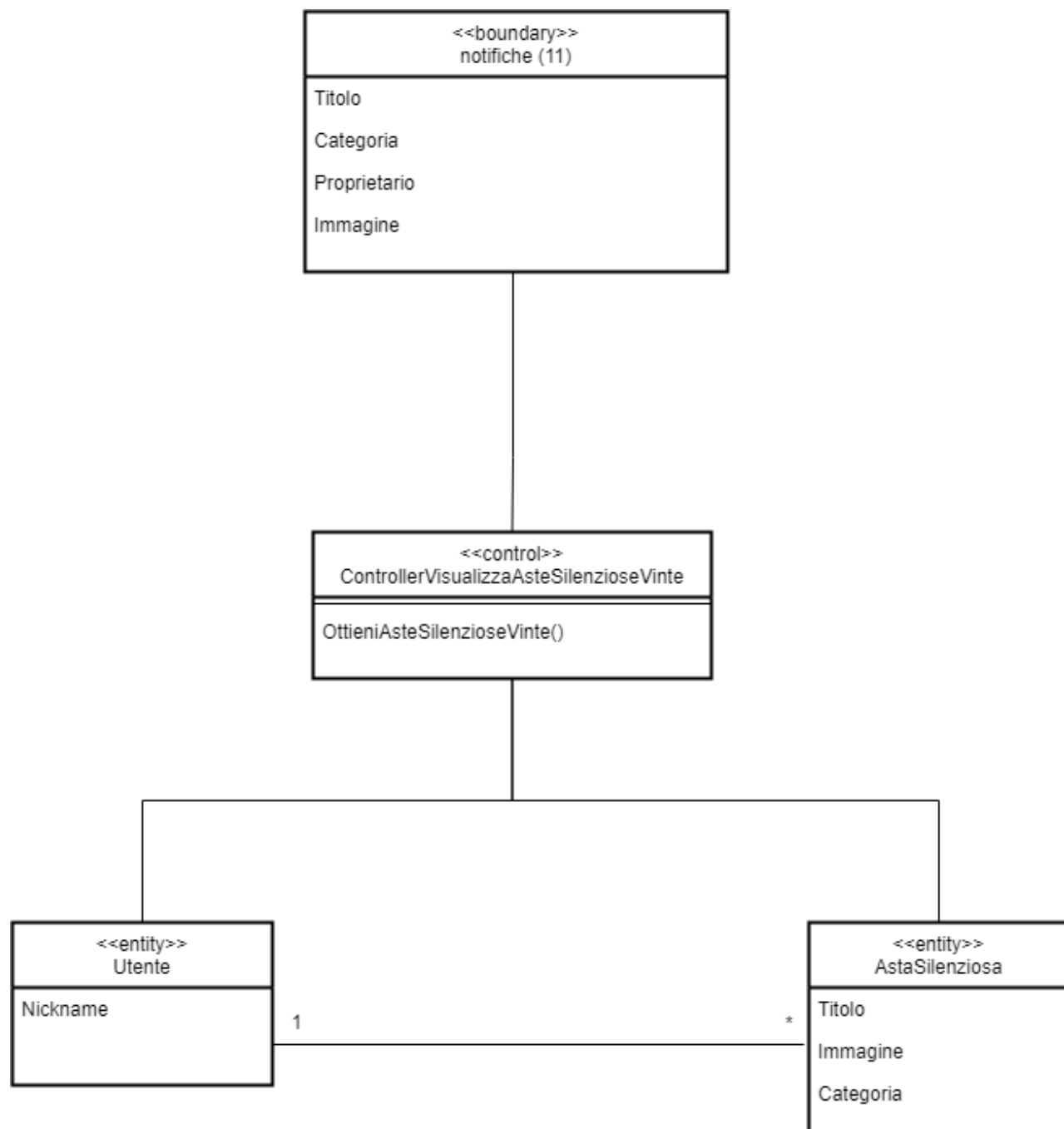
CASE tool utilizzato: drawio

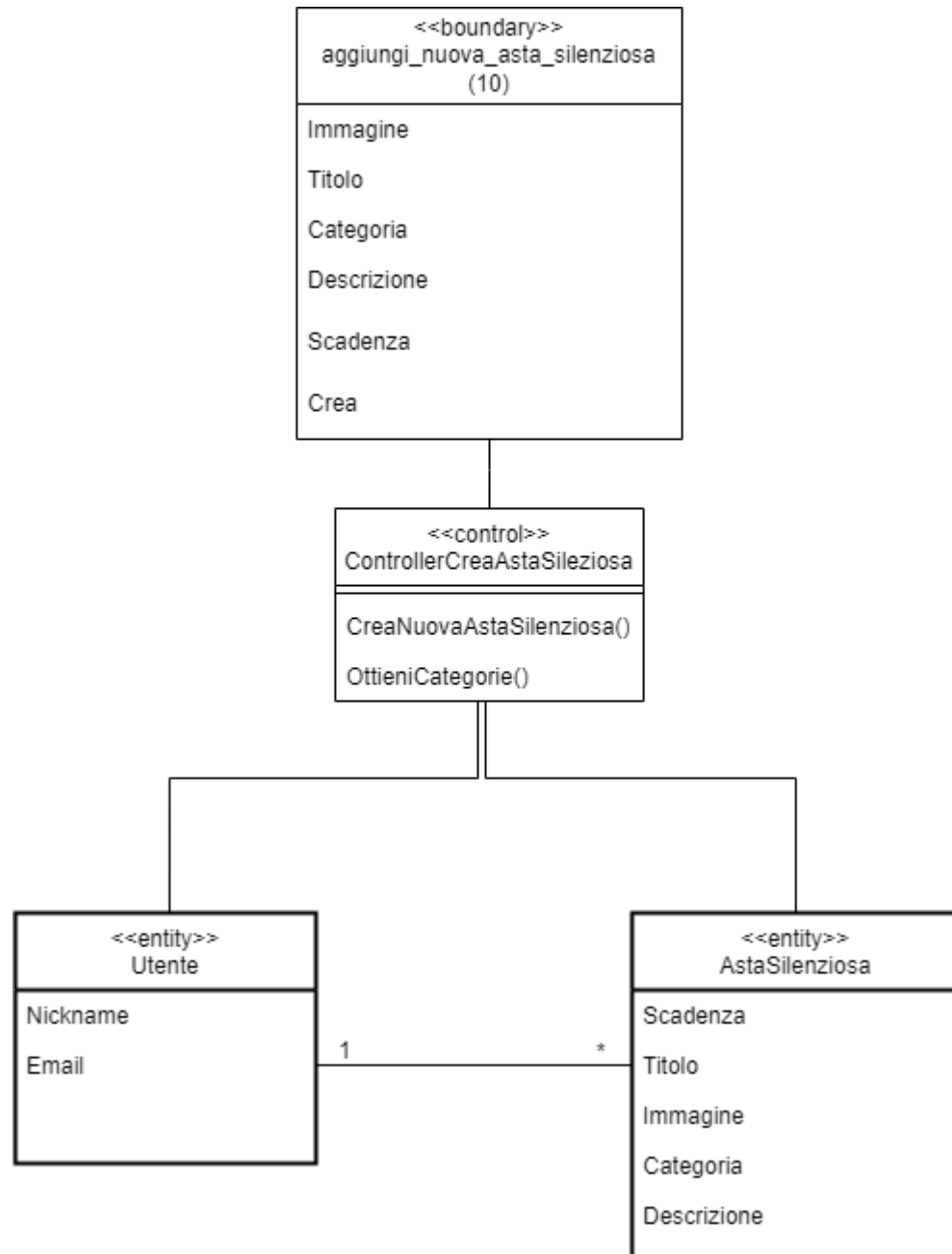
Data la grandezza, il class diagram di analisi unito è disponibile **solo** attraverso questo [link](#).

Gli altri possono essere consultati di seguito senza la necessita di seguire link esterni:

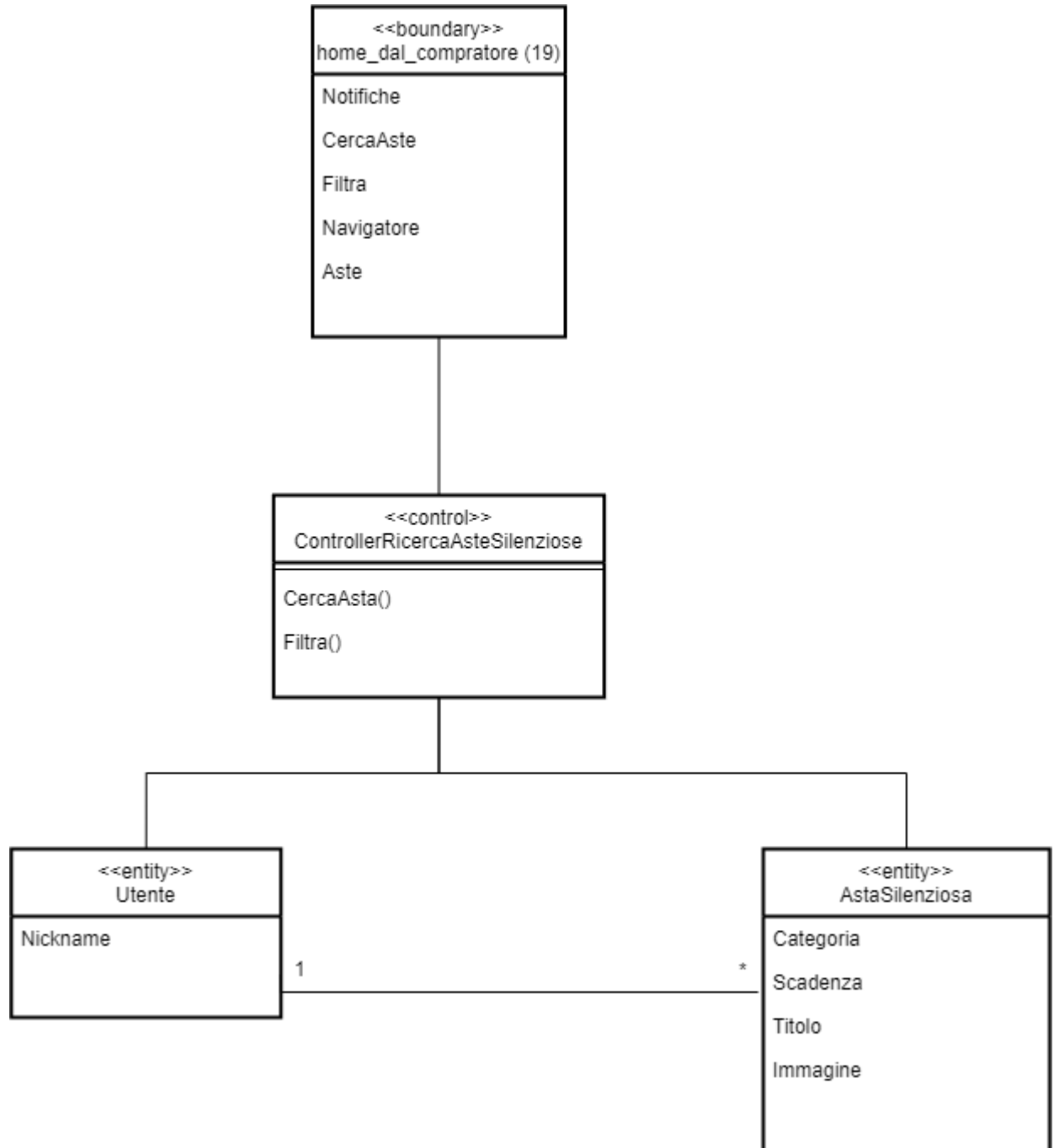


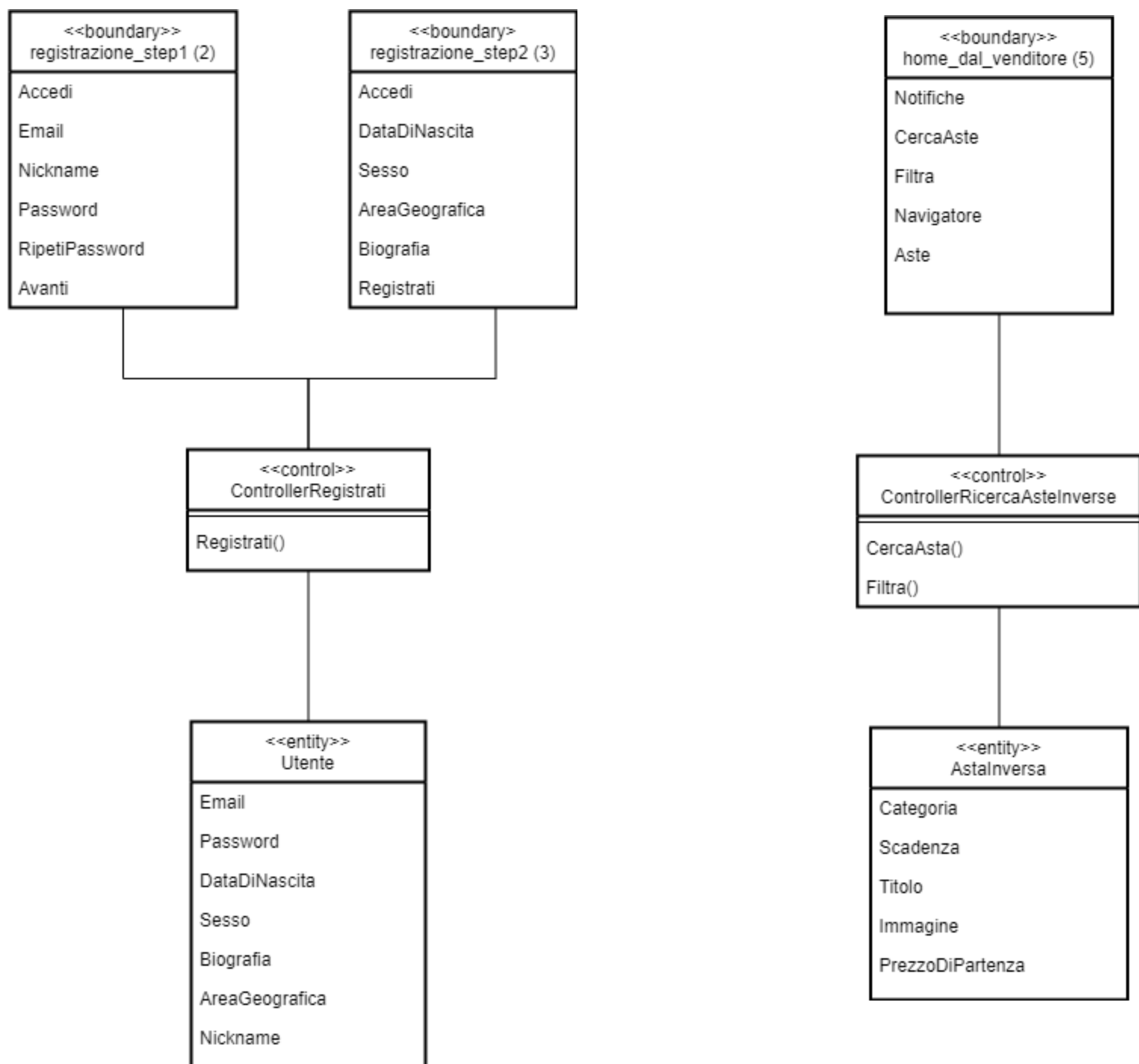


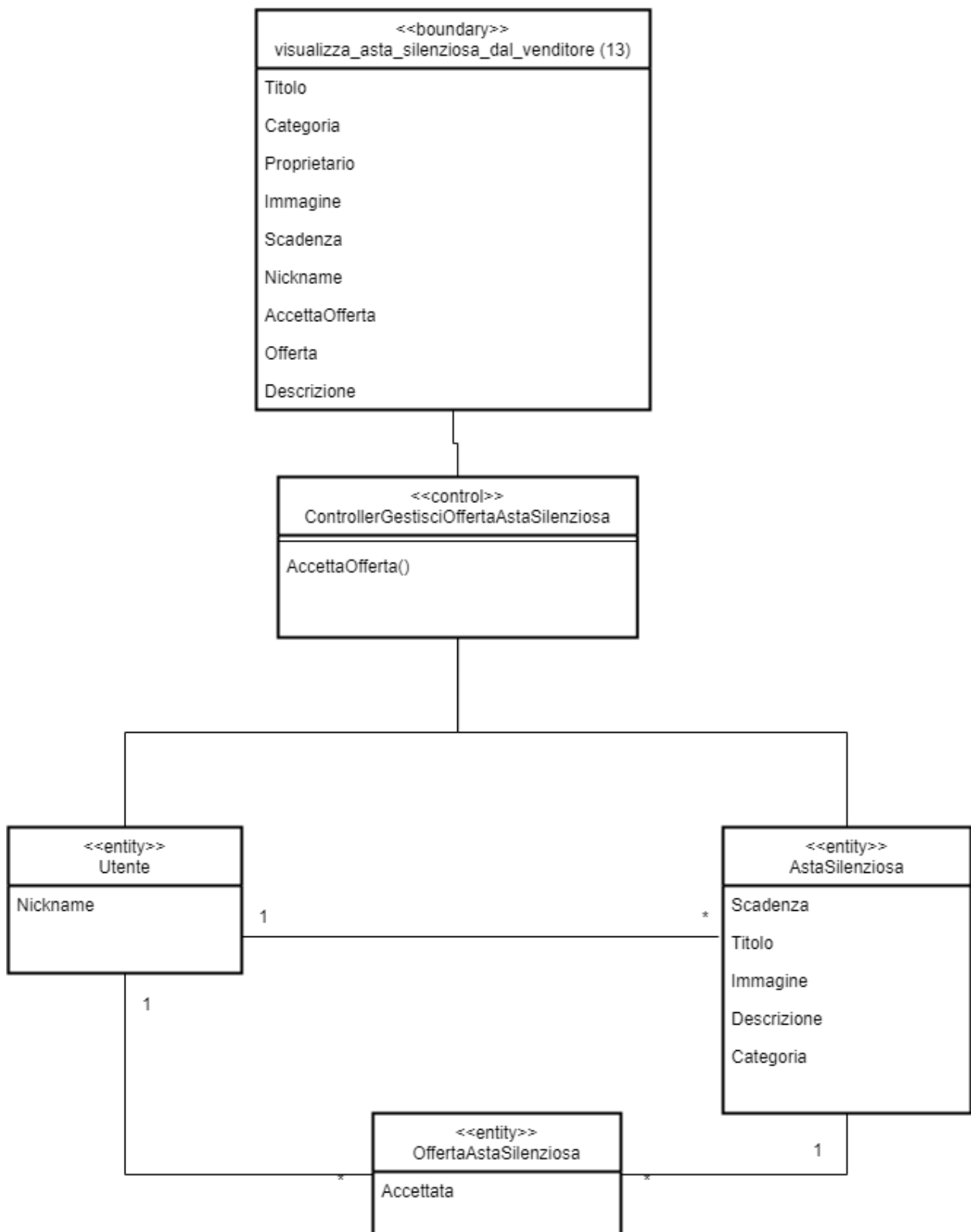


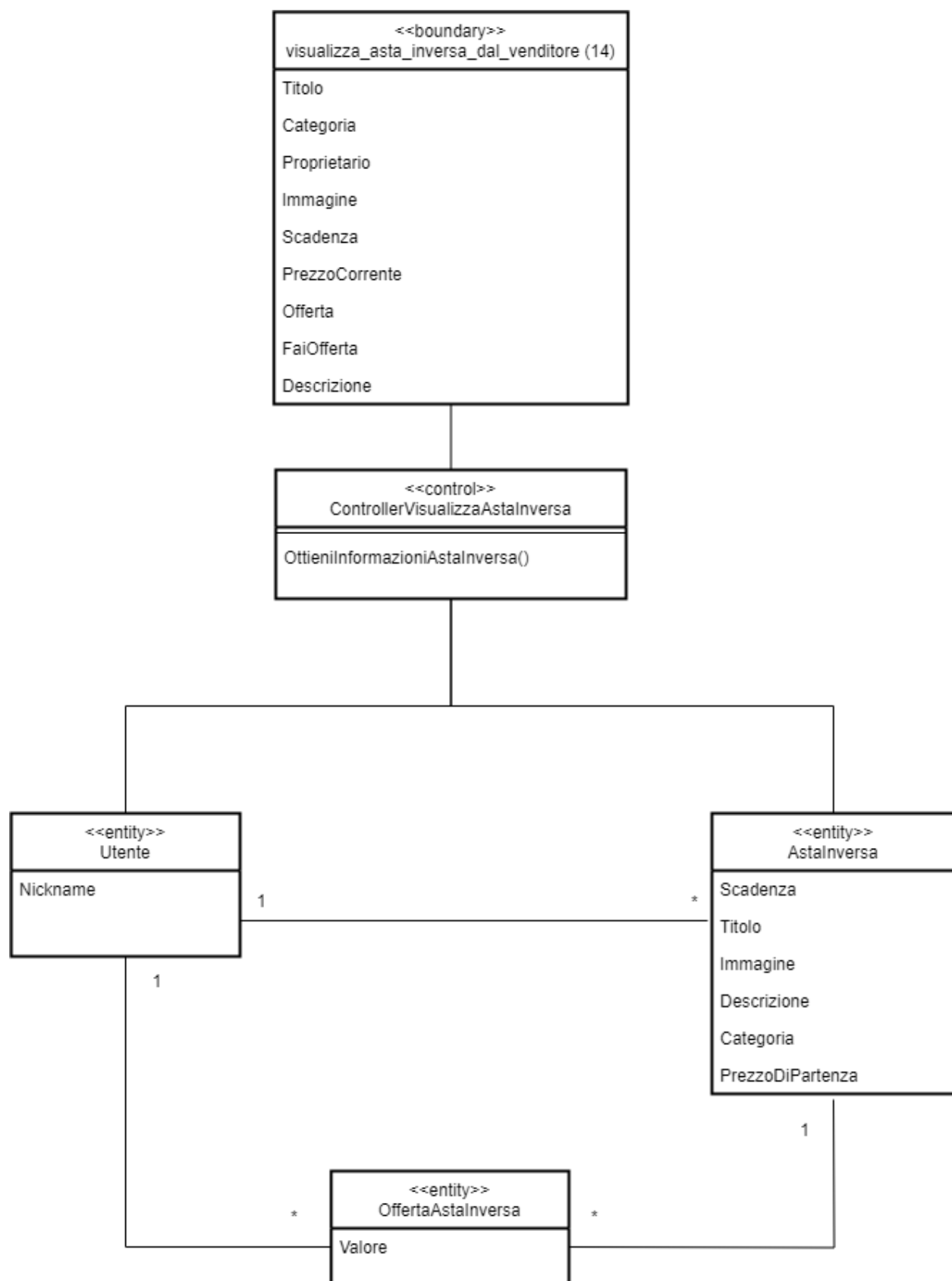


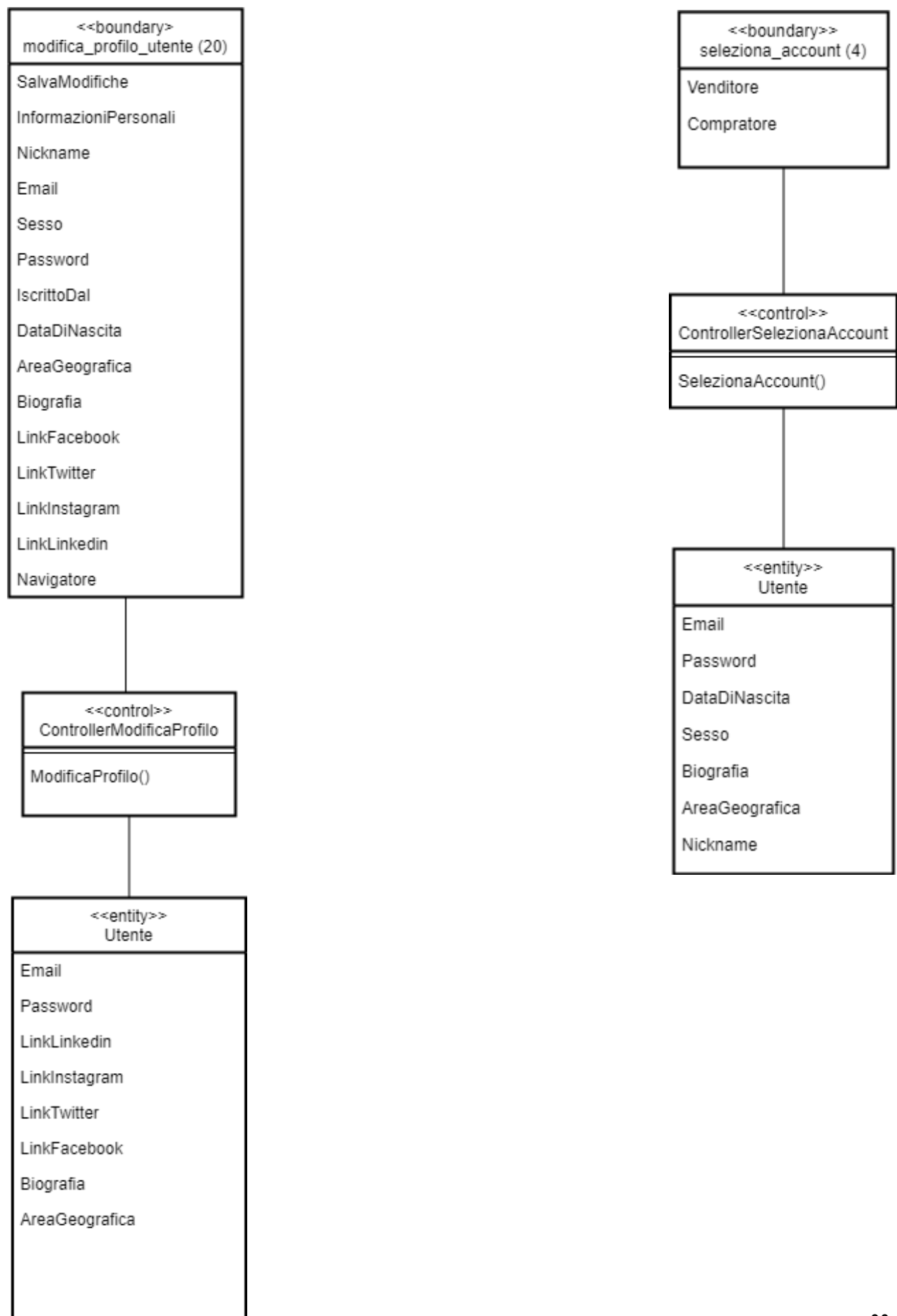


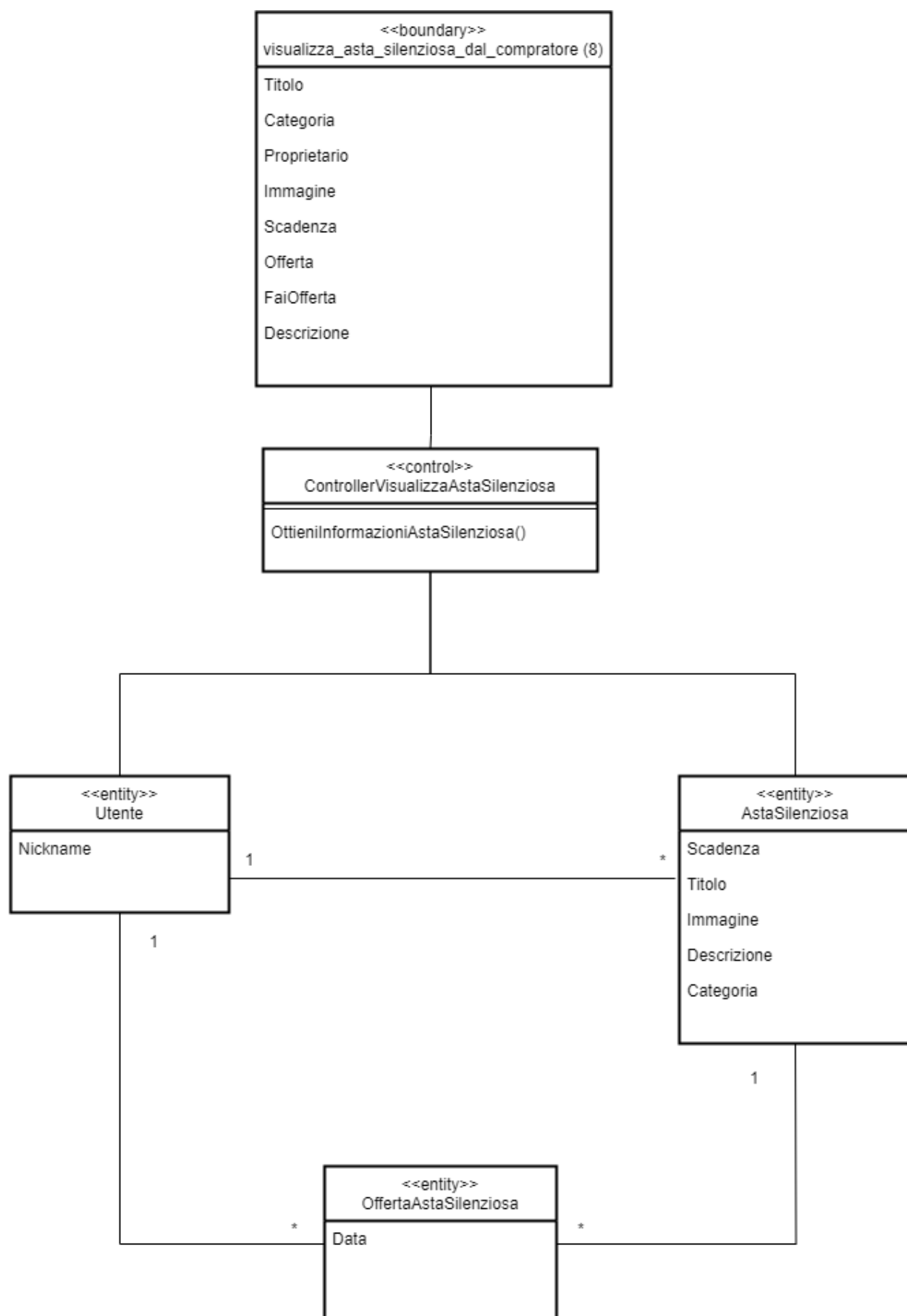


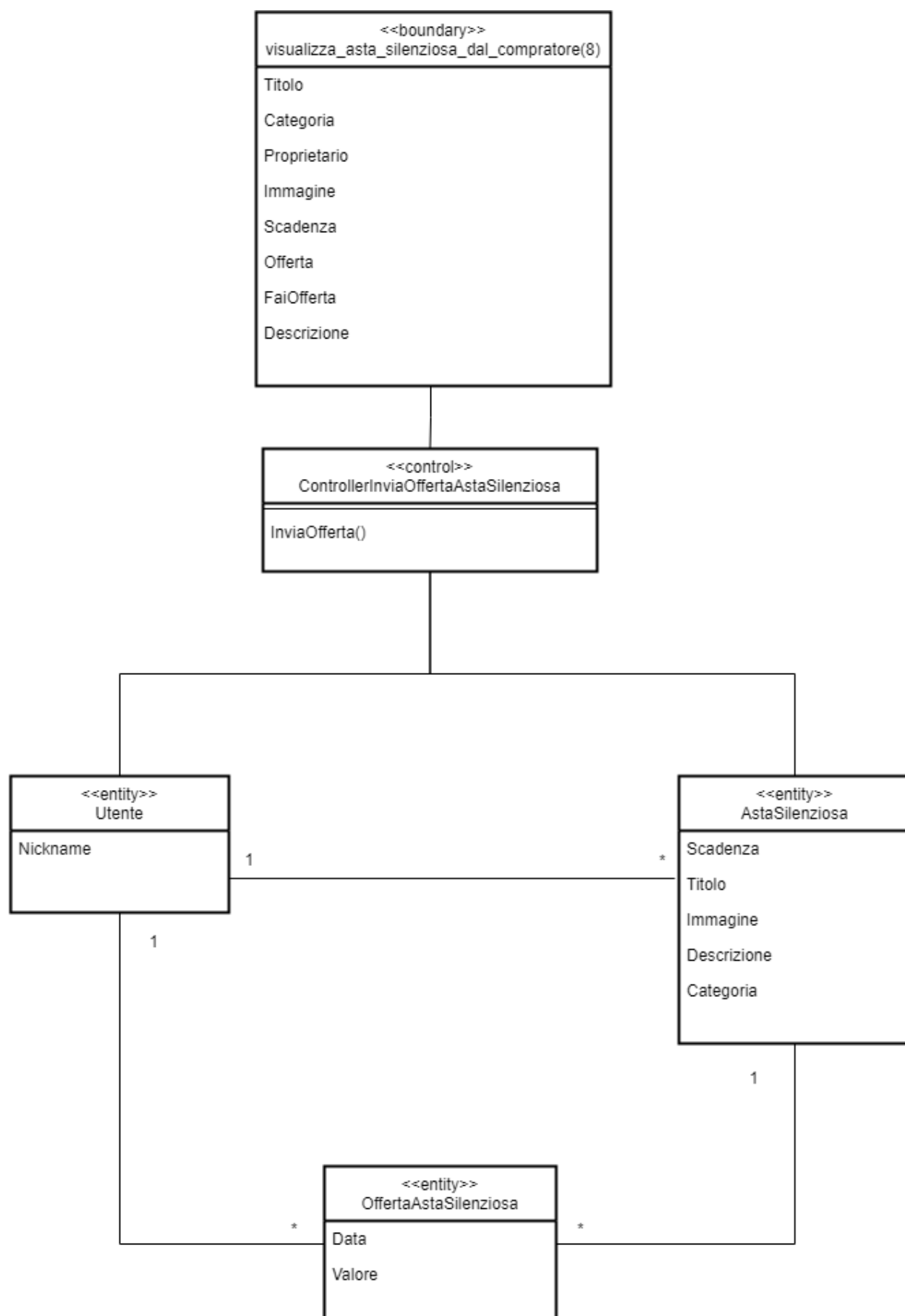


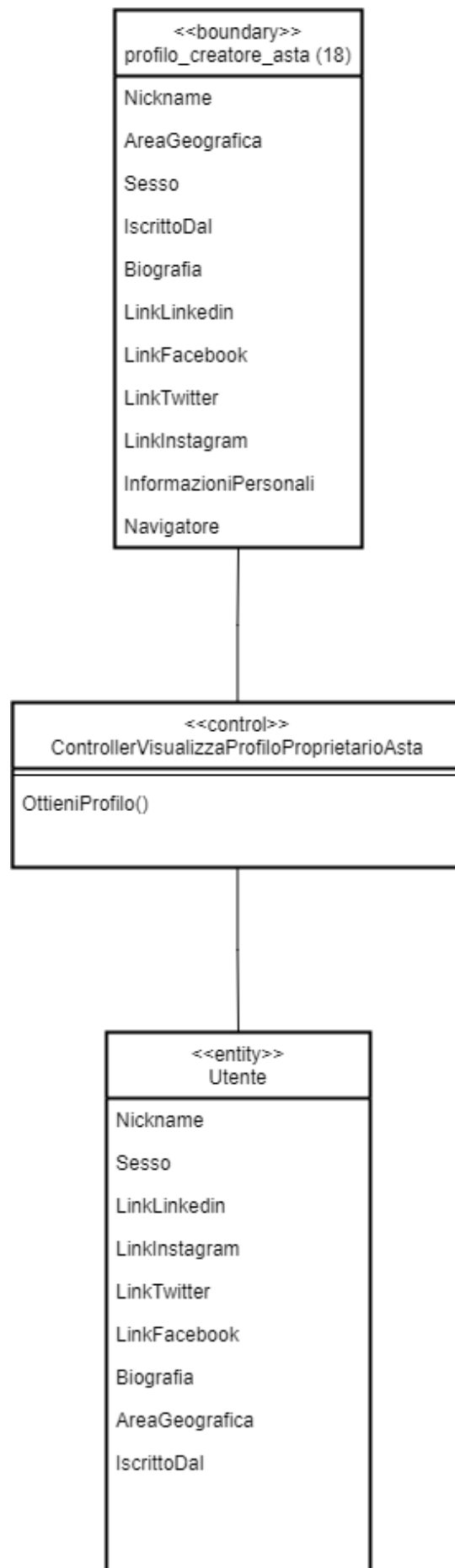


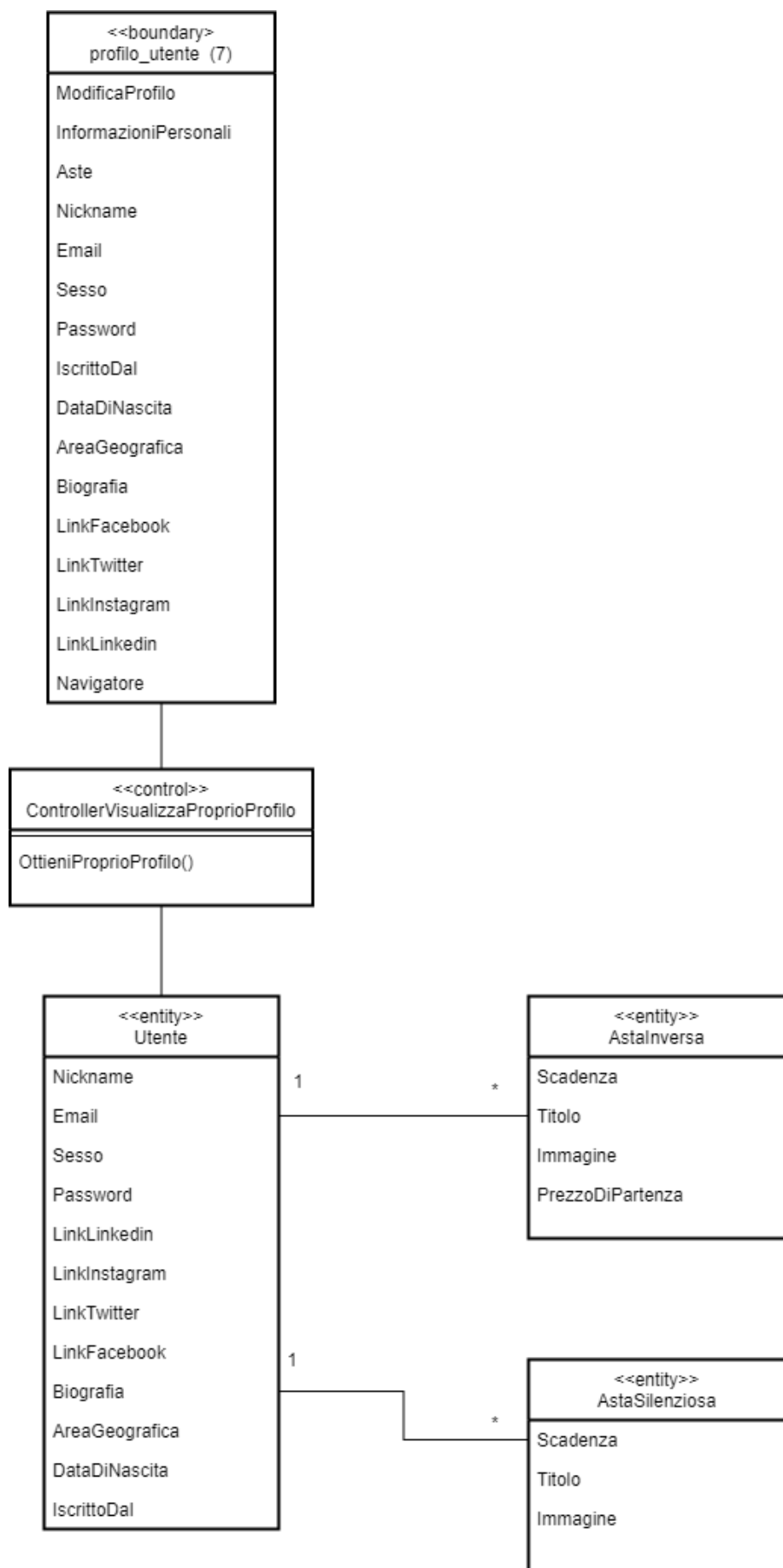












ii. Diagrammi di sequenza di analisi dei casi d'uso

CASE tool utilizzato: drawio

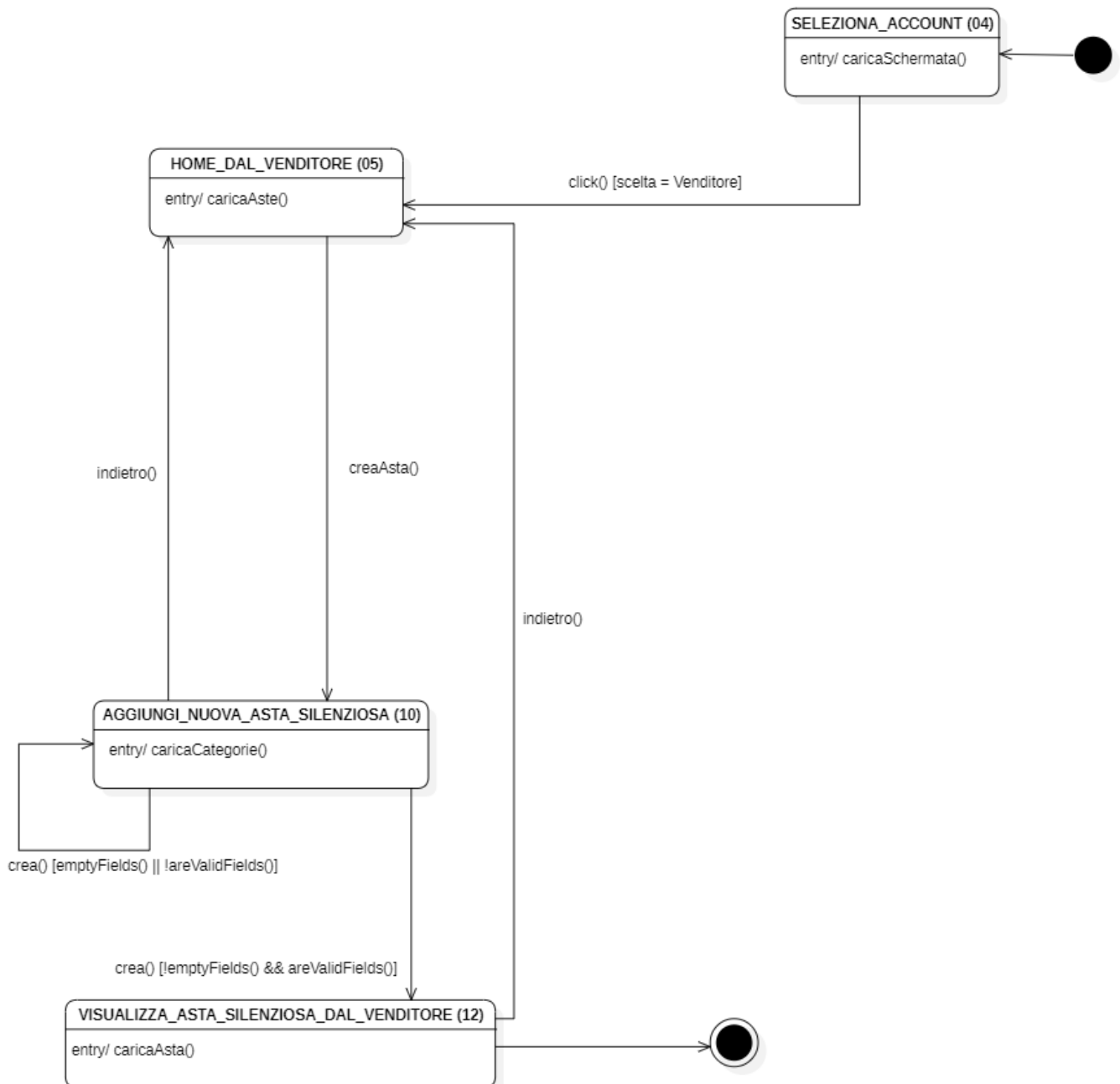
Il diagramma di sequenza di analisi per la creazione di un'asta silenziosa è disponibile presso questo [link](#).

Quello per la visualizzazione del profilo del proprietario di un'asta [qui](#).

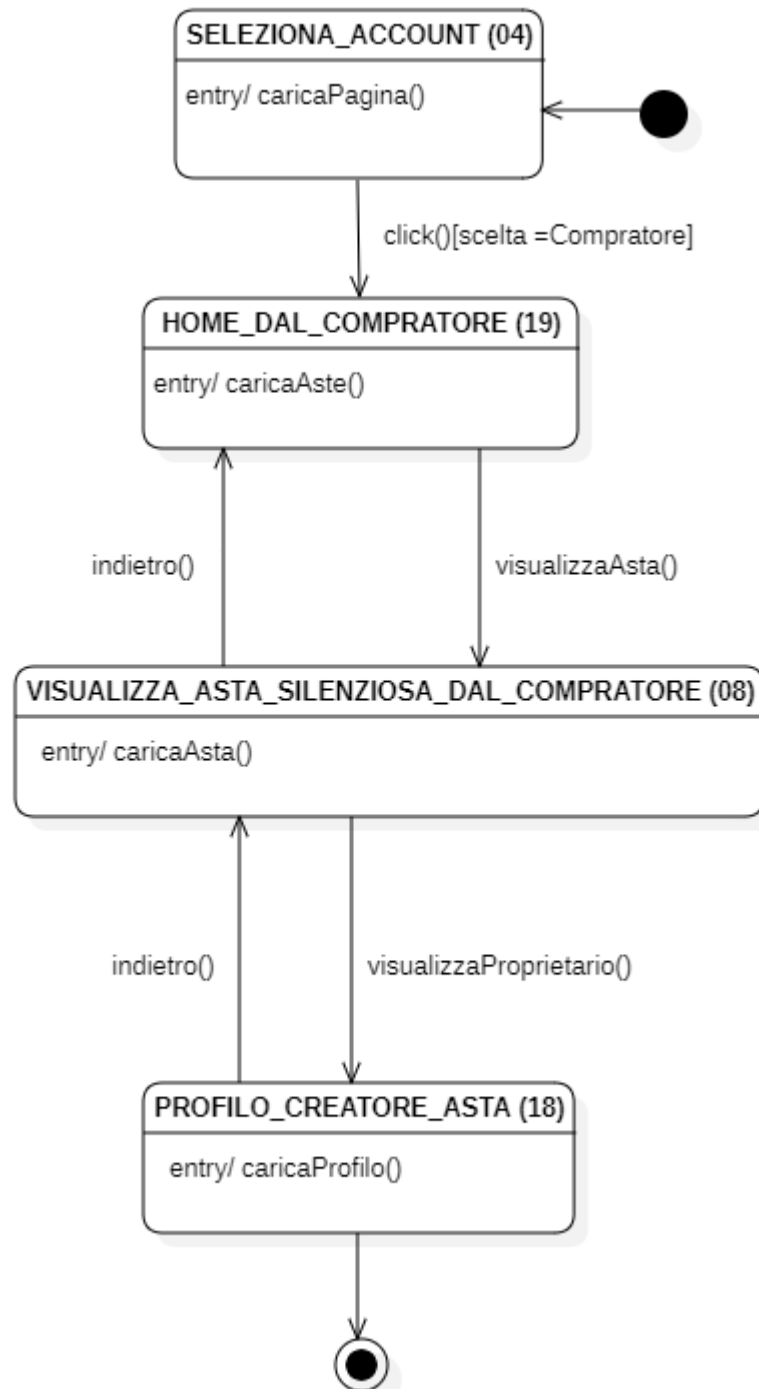
iii. Prototipazione funzionale e progettazione degli event-based statechart dell'interfaccia grafica per i casi d'uso

CASE tool utilizzato: StarUML

Statechart per lo use case: Crea un'asta silenziosa



Statechart per lo use case: Visualizza il profilo del proprietario dell'asta



2 Documento di Design del sistema.

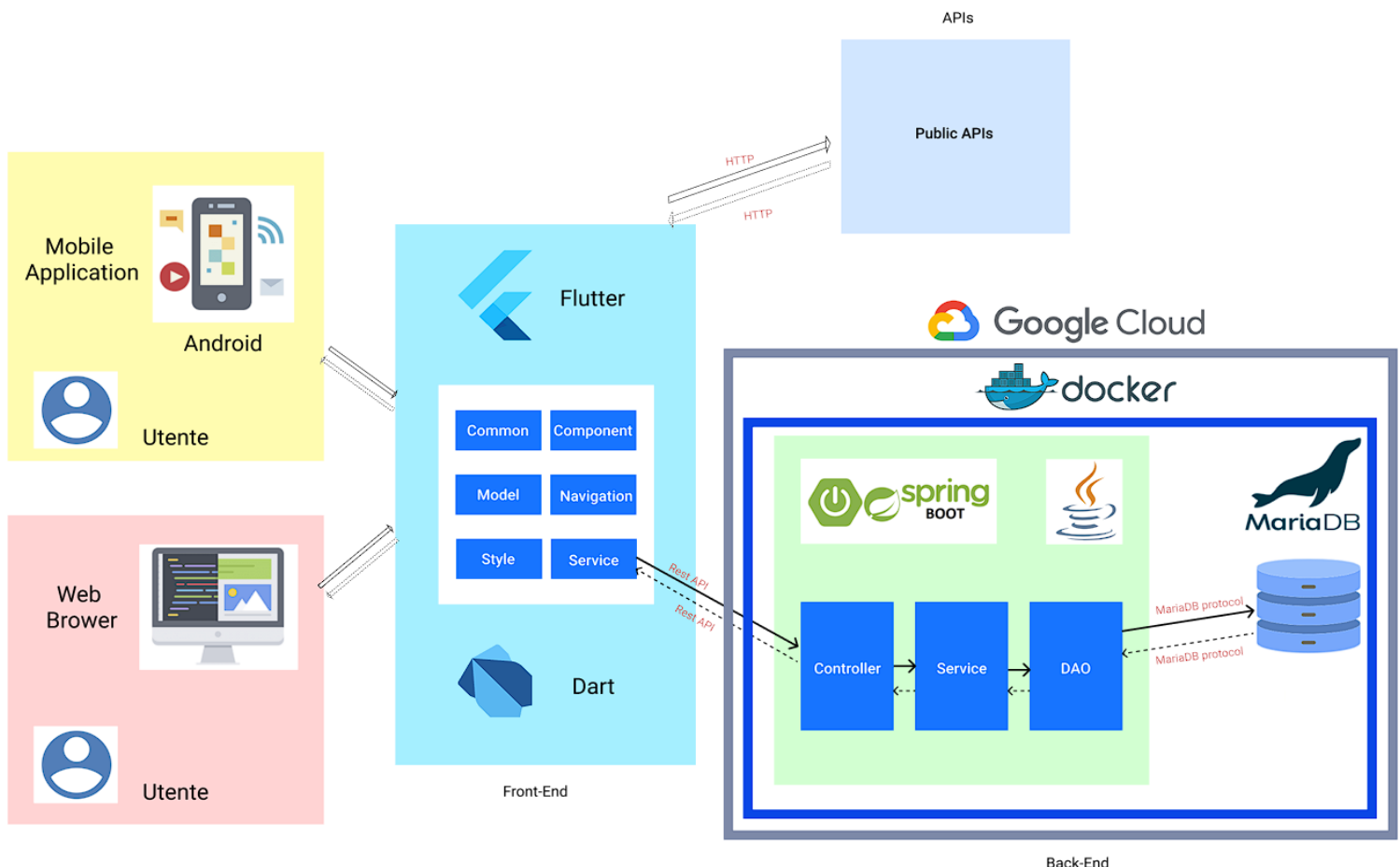
a. Descrizione dell'architettura proposta

La nostra proposta è un'architettura client-server progettata con tempi di sviluppo efficienti, garantendo al contempo flessibilità per il cliente. Questo è stato reso possibile grazie all'uso di tecnologie efficienti e all'avanguardia.

Le entità coinvolte nella comunicazione sono il client, il server e il sistema di persistenza dei dati.

La comunicazione tra il client e il server si svolge in maniera bidirezionale utilizzando il protocollo HTTP. Il formato utilizzato per la rappresentazione dei dati è JSON, che è considerato lo standard de facto per questo scopo, grazie alla sua flessibilità e semplicità. Il server mette a disposizione degli endpoint che rappresentano le REST API, consentendo al client di interagire con i dati. Il server, a sua volta, comunica con il sistema di persistenza dei dati attraverso un protocollo nativo.

Infine, per migliorare la portabilità dell'applicazione e facilitare lo sviluppo condiviso, abbiamo deciso di utilizzare Docker. Questa scelta ci permette di creare un ambiente di sviluppo uniforme e riproducibile, riducendo così i problemi legati alla configurazione dell'ambiente e aumentando l'efficienza del processo di sviluppo. Come servizio di VPS abbiamo scelto GCP che a valle di una ricerca si è mostrato quello più affidabile in termini di availability.



b. Descrizione/motivazione delle scelte tecnologiche adottate

i. Il server

Il server, nonché backend, è stato sviluppato con Spring boot, un framework in java che offre la possibilità di mettere a disposizione delle REST API. La scelta di un linguaggio di programmazione orientato agli oggetti come Java, rispetto ad un linguaggio general purpose come Python, è giustificata dalla nostra necessità di sfruttare ampiamente concetti come l'ereditarietà, interfacce e classi astratte.

Abbiamo adottato un'architettura n-tier chiusa, in cui ciascun livello interagisce solo con quello immediatamente adiacente. Il Model layer include le Entity e i DAO. Nel Business layer, invece, troviamo i Controller e i Service. L'introduzione dei Service nel livello Business è di fondamentale importanza: essi, infatti, racchiudono l'intera logica di business del software, alleggerendo così i Controller. Questo contribuisce a una più netta separazione delle responsabilità tra i vari livelli e favorisce la riutilizzabilità degli stessi.

ii. Il client

Il client, nonché frontend, è stato sviluppato interamente con Flutter, un framework che segue la filosofia “write once and deploy everywhere”. Abbiamo utilizzato il pattern architetturale MVC. Infine, Flutterflow che ha incrementato i tempi di sviluppo grazie al GUI builder e molteplici funzionalità per l'implementazione del codice. I motivi dietro la scelta di tale framework sono molteplici:

1. **Sviluppo multiplatforma:** Unico codice sorgente che può essere eseguito su diverse piattaforme come iOS, Android, web e desktop. Questo riduce la necessità di sviluppare e mantenere codice separato per ogni piattaforma.
2. **Performance:** Dart, il linguaggio utilizzato da Flutter, converte il codice sorgente in codice nativo, migliorando ulteriormente le prestazioni dell'app.
3. **Widget personalizzati e design flessibile:** Ampia gamma di widget predefiniti e la possibilità di creare widget personalizzati. Questo ci ha consentito di realizzare interfacce utente altamente personalizzate e di implementare facilmente design complessi.
4. **Integrazione con Backend:** Supporto per l'integrazione con vari servizi backend e API, consentendoci di rendere l'applicazione completamente funzionale con funzionalità di rete, autenticazione e gestione degli stati.

iii. La persistenza dei dati

La persistenza dei dati nel nostro sistema è gestita attraverso un database relazionale, specificamente MariaDB. Questa scelta è motivata dalla natura delle entità coinvolte nel nostro problema, come ad esempio, l'utente, l'asta e l'offerta. Queste entità possono essere efficacemente rappresentate attraverso una struttura dati ben definita, che tende a rimanere stabile nel tempo.

Inoltre, le relazioni intrinseche tra queste entità si adattano perfettamente all'uso dei vincoli di integrità dei dati offerti dai sistemi di gestione dei database relazionali (RDBMS). Questi vincoli ci permettono di mantenere la coerenza e l'integrità dei dati, assicurando che le relazioni tra le entità siano rispettate e che i dati siano accurati e affidabili. In questo modo, possiamo garantire che il nostro sistema funzioni in modo affidabile.

c. Diagramma delle classi di Design

CASE tool utilizzato: drawio

[Qui](#) è possibile vedere interamente il class diagram di design.

Al fine di favorire lo sviluppo futuro del codice, la riusabilità e manutenibilità sono state effettuate scelte importanti in fase di object design:

1. Largo utilizzo di interfacce e classi astratte anche a favore del Dependency inversion principle (DIP).
2. Design pattern Singleton per evitare più istanze di una classe.
3. Design pattern DAO.
4. Design pattern Factory.
5. Utilizzo dei generics per evitare cast pericolosi.

Un'ulteriore analisi può essere fatta rispondendo a ipotetiche domande:

D: Quanto è facile cambiare tecnologia per il front-end? Se volessi utilizzare Swing per costruire una desktop app?

R: A livello più alto abbiamo dei controller che sono interfacce, per cui basta implementarle per ottenere ControllerSwing anziché ControllerREST. Il resto non va cambiato, i ControllerSwing possono riutilizzare i Service sottostanti in quanto questi sono agnostici dal tipo di Controller sovrastante.

D: Se volessi aggiungere un'altra tipologia di asta alla mia applicazione?

R: Dopo aver esaminato tutte le varietà di aste nel documento, comprese quelle non pertinenti al nostro ambito, abbiamo identificato tutti gli aspetti (metodi, attributi...) condivisi da queste ultime e creato Service, Controller, DAO e Entity appositi. Basta quindi creare un ControllerNuovaTipologiaAsta che implementi ControllerAsta e aggiungervi i metodi specifici per il tipo di asta. Discorso analogo per i Service e per i DAO. Al livello di entità si aggiunge il nuovo tipo di asta che dovrà estendere la classe astratta Asta.

D: Se volessi cambiare database e usare Postgres o un altro meccanismo per la persistenza?

R: Basta implementare le diverse interfacce DAO per creare i vari PostgresDAO. Successivamente, aggiungi il caso "Postgres" al DAOFactory. Infine, sostituisci "Mysql" con "Postgres" nei Service corrispondenti.

D: Riutilizzare i Service o qualche altra classe per altri progetti è possibile?

R: Abbiamo cercato di tenere quanto più alta la coesione e quanto più basso l'accoppiamento per i metodi e le classi proprio per favorire questo. Metodi e classi, infatti, hanno una responsabilità unica e precisa che risulta comprensibile già dal loro nome. Inoltre, l'utilizzo di interfacce ci ha permesso di spezzare le dipendenze dalle classi di più alto livello alle implementazioni sottostanti (DIP), un service infatti non ha dipendenze con "Mysql...DAO", bensì solo con l'interfaccia DAO.

i. 0 vendor lock-in

Un aspetto cruciale che abbiamo considerato sin dalle prime fasi di progettazione è stato l'obiettivo di raggiungere il "zero vendor lock-in". Questo significa evitare di essere dipendenti dalle tecnologie di un fornitore specifico. Abbiamo adottato due strategie principali per avvicinarci a questo obiettivo:

- Per l'hosting delle immagini delle aste, utilizziamo principalmente imgBB. Tuttavia, abbiamo previsto un piano di riserva nel caso in cui il servizio fosse in manutenzione. L'upload delle immagini è gestito dal Service di appartenenza, dove abbiamo implementato la logica applicativa necessaria per utilizzare imgBB quando è disponibile, e ricorrere al file system sottostante come alternativa.
- Per il login non ci siamo affidati a fornitori di terze parti.

d. Diagrammi di sequenza di design per i casi d'uso

CASE tool utilizzato: drawio

[Qui](#) è possibile vedere il sequence diagram di design per la creazione di una nuova asta silenziosa.

[Qui](#) quello per la visualizzazione del profilo del proprietario dell'asta.

3 Codice Sorgente sviluppato con Dockerfile

Il codice sorgente, sia per il backend che per il frontend è disponibile oltre che nell'archivio inviato per email, anche nella repository.

Per accedere alla repository, si prega di fare riferimento al [punto b](#).

a. File di build automatico

La modalità di consegna attuale non permette l'invio di file compilati. Pertanto, il Dockerfile, che dipende dal file .jar dell'applicazione Spring (e quindi dai file .class, che sono file compilati), sarà disponibile **unicamente** nella cartella root della repository.

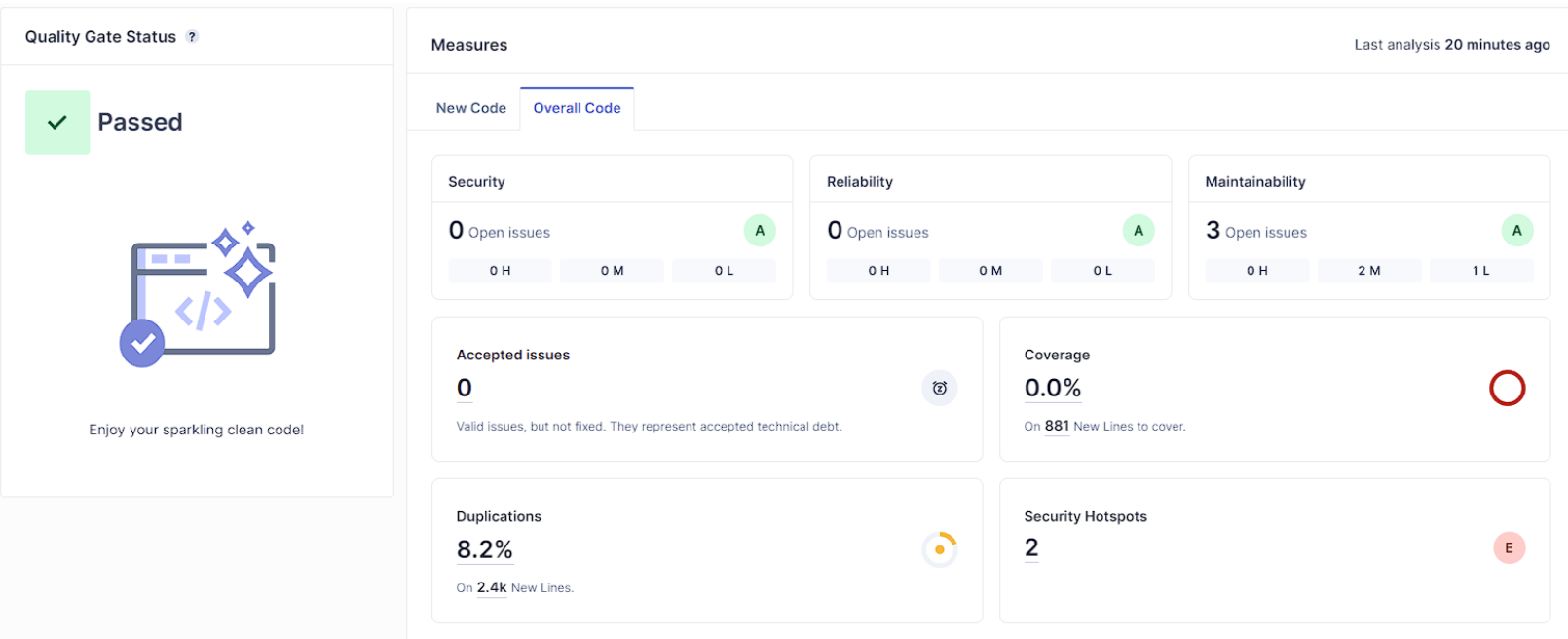
Inoltre, è disponibile il file docker-compose.yml. Questo file permette di avviare l'intera infrastruttura con un singolo comando, **vanificando** la necessità di avere un file di build separato: docker-compose up

b. Uso di strumenti di versioning

Github è stato utilizzato come strumento di versioning. È possibile accedere alla repository in modalità di sola lettura. Questo è reso possibile grazie a un token che abbiamo generato appositamente. Utilizzare il seguente comando per accedere alla repository:

```
git clone  
https://Okitask:github\_pat\_11ASI5K0Y0AiVTgXvnbWs3\_ldgFTaEqpM9BrwvtfOyYDgZ5HbYFvx8CblnyH5rfutYS3CTIYF5T85xp4fx@github.com/Okitask/dietiAuction.git
```

c. Report di qualità del codice, generati da SonarQube per il back-end



Come si evince dall'immagine, tramite un processo di raffinazione siamo riusciti ad ottenere "A" nei punti di interesse e ad abbassare le duplicazioni di codice a favore del principio DRY (Don't repeat yourself).

L'unica "E" ottenuta è nel dominio della security che esula dallo scopo di questo corso. Tale votazione è stata influenzata da valori hard-coded come la password utilizzata per effettuare la connessione al database:

The image shows the SonarQube Security Hotspots detail view. It displays 2 Security Hotspots. The first hotspot is titled 'Authentication' and has a review priority of 'High'. The description states: 'PASSWORD' detected in this expression, review this potentially hard-coded password. The second hotspot is titled 'Others' and has a review priority of 'Low'. The description states: Make sure using this hardcoded IP address is safe here. The code snippet shown is from src/main/java/com/db/mysql/MySQLConnection.java. The code includes imports for java.sql.SQLException and java.util.logging.Logger, and a public class MySQLConnection with private static final fields for instance, ADDRESS, PORT, DATABASE, USER, and PASSWORD. The PASSWORD field is set to 'dieti'. A red box highlights the PASSWORD field with the message: 'PASSWORD' detected in this expression, review this potentially hard-coded password.

4 Testing e valutazione sul campo dell'usabilità

a. Codice xUnit per unit testing

La fase di unit testing è stata completata utilizzando i test JUnit per il backend.

L'architettura software è stata ben stratificata, rendendo i metodi molto snelli. Di conseguenza, i metodi più complessi sono quelli che effettuano le manipolazioni sui model, ovvero i metodi dei DAO. Tuttavia, questi metodi, a causa della loro natura, non accettano più di un parametro come richiesto dal documento, ma solo l'oggetto che rientra nella responsabilità del metodo stesso.

Pertanto, abbiamo deciso di testare i metodi delle classi che incorporano l'intera logica applicativa, poiché sono anche quelli più soggetti a cambiamenti e quindi a errori. Questi metodi appartengono alle classi Service.

Le classi Service necessitano esclusivamente dei corrispondenti DAO per operare. Per simulare il comportamento di questi oggetti DAO durante i test, abbiamo utilizzato degli stub. Gli stub sono oggetti che simulano il comportamento degli oggetti reali in un contesto di test.

Abbiamo creato questi stub utilizzando Mockito, un framework di testing per Java. Mockito è particolarmente adatto a questo scopo perché permette di creare facilmente oggetti mock, che possono essere utilizzati per simulare il comportamento di oggetti reali in un contesto di test. In questo modo, siamo stati in grado di testare le classi Service in modo isolato, senza la necessità di interagire con i veri oggetti DAO.

Tutti i metodi sono stati testati con un approccio black-box con strategia R-WECT (WeakRobust Equivalence Class Testing), testando tutte le classi di equivalenze valide e quelle non valide prese singolarmente. La scelta di R-WECT è stata intrapresa per evitare un numero di test esponenziali sulle classi di equivalenza.

Di seguito i metodi testati:

- Ricezione di aste inverse.
- Accesso al sistema.
- Invio di un'offerta segreta per un'asta silenziosa.
- Creazione di un'asta silenziosa.

Ricezione di aste inverse

Il metodo ottieniAste(String categoria, String titolo) prende in input due parametri, una categoria ed un titolo. Le classi di equivalenza individuate sono:

CEC1→ Parametro categoria "Tutte" (valido)
CEC2→ Parametro categoria null (non valido)
CEC3→ Parametro categoria stringa vuota (non valido)
CEC4→ Parametro categoria "Elettronica" (valido)

CET1→ Parametro titolo stringa vuota (valido)
CET2→ Parametro titolo "Prova" (valido)
CET3→ Parametro titolo null (non valido)

CEC1,CET1

```
public class ServiceAstaInversaTest {

    @Mock
    AstaInversaDAO astadao;

    @InjectMocks
    ServiceAstaInversa service;

    @Test
    public void testOttieniAste() {

        //FASE DI ARRANGE

        // Inizializzo gli oggetti astadao e service (quelli con @Mock @InjectMocks)
        MockitoAnnotations.openMocks(this);

        // Creo dei dati fasulli di test
        Asta asta1 = new AstaInversa();
        Asta asta2 = new AstaInversa();
        List<Asta> astaList = Arrays.asList(asta1, asta2);

        // Configuro il mock del DAO per restituire i dati di test
        when(astadao.ottdieniAste()).thenReturn(astaList);

        //FASE DI ACT
        List<Asta> result = service.ottdieniAste(categoria:"Tutte", titolo:"");

        //FASE DI ASSERT
        assertEquals(astaList, result);
    }
}
```

CEC1,CET2

```
@Test
public void testOttieniAsteConTitolo() {

    //ARRANGE
    MockitoAnnotations.openMocks(this);

    Asta asta1 = new AstaInversa();
    Asta asta2 = new AstaInversa();
    List<Asta> astaList = Arrays.asList(asta1, asta2);

    when(astadao.ottdieniAsteTitolo(titolo:"Prova")).thenReturn(astaList);

    //ACT
    List<Asta> result = service.ottdieniAste(categoria:"Tutte", titolo:"Prova");

    //ASSERT
    assertEquals(astaList, result);
}
```

CEC2,CET1

```
@Test
public void testOttieniAsteConCategoriaNulla() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.ottieniAsteCategorie(categoria:null)).thenReturn(value:null);

    // ACT
    List<Asta> result = service.ottieniAste(categoria:null, titolo:"");

    //ASSERT
    assertNull( result);
}
```

CEC3,CET1

```
@Test
public void testOttieniAsteConCategoriaVuota() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.ottieniAsteCategorie(categoria:"")).thenReturn(value:null);

    // ACT
    List<Asta> result = service.ottieniAste(categoria:"", titolo:"");

    //ASSERT
    assertNull( result);
}
```

CEC4,CET3

```
@Test
public void testOttieniAsteConTitoloNull() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.ottieniAsteCategorie(categoria:"Elettronica")).thenReturn(value:null);

    // ACT
    List<Asta> result = service.ottieniAste(categoria:"Elettronica", titolo:null);

    //ASSERT
    assertNull( result);
}
```

CEC4,CET2

```
@Test
public void testOttieniAsteConCategoriaETitolo() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    Asta asta1 = new AstaInversa();
    Asta asta2 = new AstaInversa();
    List<Asta> astaList = Arrays.asList(asta1, asta2);

    when(astadao.ottieniAsteCategoriaTitolo(categoria:"Elettronica",titolo:"Prova")).thenReturn(astaList);

    // ACT
    List<Asta> result = service.ottieniAste(categoria:"Elettronica", titolo:"Prova");

    //ASSERT
    assertEquals(astaList, result);
}
```

Accesso al sistema

Il metodo `effettuaAccesso` (`String nickname`, `String password`) prende in input due parametri, il nickname e la password dell'utente. Le classi di equivalenza individuate sono:

- CEN1 → Parametro nickname "Fabrizio1" (valido)
- CEN2 → Parametro nickname null (non valido)
- CEP1 → Parametro password "password" (valido)
- CEP2 → Parametro password null (non valido)

CEN1, CEP1

```
public class ServiceUtenteTest {  
  
    @Mock  
    UtenteDAO utenteDAO;  
  
    @InjectMocks  
    ServiceAccesso serviceAccesso;  
  
    @Test  
    public void testEffettuaAccesso() {  
        // ARRANGE  
        MockitoAnnotations.openMocks(this);  
  
        when(utenteDAO.accedi(any(type:Utente.class))).thenReturn(value:true);  
  
        // ACT  
        Boolean result = serviceAccesso.effettuaAccesso(nickname:"Fabrizio1", password:"password");  
  
        // ASSERT  
        assertTrue(result);  
    }  
}
```


CEN2,CEP1

```
@Test
public void testEffettuaAccessoConNicknameNullo() {
    //ARRANGE
    MockitoAnnotations.openMocks(this);

    when(utenteDAO.accedi(any(type:Utente.class))).thenReturn(value:false);

    // ACT
    Boolean result = serviceAccesso.effettuaAccesso(nickname:null, password:"password");

    //ASSERT
    assertFalse(result);
}
```

CEN1,CEP2

```
@Test
public void testEffettuaAccessoConPasswordNulla() {
    //ARRANGE
    MockitoAnnotations.openMocks(this);

    when(utenteDAO.accedi(any(type:Utente.class))).thenReturn(value:false);

    // ACT
    Boolean result = serviceAccesso.effettuaAccesso(nickname:"Fabrizio1", password:null);

    //ASSERT
    assertFalse(result);
}
```

Invio di un'offerta segreta per un'asta silenziosa.

Il metodo `inviaOffertaSilenziosa` (`String nickname`, `Integer idAsta`, `Float offerta`) prende in input 3 parametri, il nickname dell'utente che vuole inviare l'offerta, l'id dell'asta di interesse e il valore dell'offerta. Le classi di equivalenza individuate sono:

- CEN1 → Parametro nickname "test" (valido)
- CEN2 → Parametro nickname null (non valido)
- CEI1 → Parametro idAsta positivo (valido)
- CEI2 → Parametro idAsta negativo (non valido)
- CEI3 → Parametro idAsta null (non valido)
- CEO1 → Parametro offerta positiva (valido)
- CEO2 → Parametro offerta negativa (non valido)
- CEO3 → Parametro offerta null (non valido)

CEN1,CEI1,CEO1

```
@Test
public void testInviaOffertaSilenziosa() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.inviaOffertaSilenziosa(any(type:AstaSilenziosa.class))).thenReturn(value:true);

    // ACT
    Boolean result = service.inviaOffertaSilenziosa(nickname:"test", idAsta:1, offerta:100.0f);

    // ASSERT
    assertTrue(result);
}
```

CEN2,CEI1,CEO1

```
@Test
public void testInviaOffertaSilenziosaConNicknameNullo() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.inviaOffertaSilenziosa(any(type:AstaSilenziosa.class))).thenReturn(value:false);

    // ACT
    Boolean result = service.inviaOffertaSilenziosa(nickname:null, idAsta:1, offerta:100.0f);

    // ASSERT
    assertFalse(result);
}
```

CEN1,CEI2,CEO1

```
@Test
public void testInviaOffertaSilenziosaConIdNegativo() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.inviaOffertaSilenziosa(any(type:AstaSilenziosa.class))).thenReturn(value:false);

    // ACT
    Boolean result = service.inviaOffertaSilenziosa(nickname:"test", -1, offerta:100.0f);

    // ASSERT
    assertFalse(result);
}
```

CEN1,CEI3,CEO1

```
@Test
public void testInviaOffertaSilenziosaConIdNullo() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.inviaOffertaSilenziosa(any(type:AstaSilenziosa.class))).thenReturn(value:false);

    // ACT
    Boolean result = service.inviaOffertaSilenziosa(nickname:"test", idAsta:null, offerta:100.0f);

    // ASSERT
    assertFalse(result);
}
```

CEN1,CEI1,CEO2

```
@Test
public void testInviaOffertaSilenziosaConOffertaNegativa() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.inviaOffertaSilenziosa(any(type:AstaSilenziosa.class))).thenReturn(value:false);

    // ACT
    Boolean result = service.inviaOffertaSilenziosa(nickname:"test", idAsta:1, -1f);

    // ASSERT
    assertFalse(result);
}
```

CEN1,CEI1,CEO3

```
@Test
public void testInviaOffertaSilenziosaConOffertaNulla() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.inviaOffertaSilenziosa(any(type:AstaSilenziosa.class))).thenReturn(value:false);

    // ACT
    Boolean result = service.inviaOffertaSilenziosa(nickname:"test", idAsta:1, offerta:null);

    // ASSERT
    assertFalse(result);
}
```

Creazione di un'asta silenziosa.

Il metodo `creaNuovaAstaDaUtente` (`Map<String,String> params`) prende in input un dizionario che contiene parametri quali `nickname`, `titolo`, `categoria`, `descrizione`, `scadenza` ed eventualmente un immagine. Le classi di equivalenza individuate sono:

- CEN1→ Parametro `nickname` “test” (valido)
- CEN2→ Parametro `nickname` null (non valido)
- CET1→ Parametro `titolo` “test” (valido)
- CET2→ Parametro `titolo` null (non valido)
- CEC1→ Parametro `categoria` “Elettronica” (valido)
- CEO2→ Parametro `categoria` null (non valido)
- CED1→ Parametro `descrizione` “test” (valido)
- CED2→ Parametro `descrizione` null (non valido)
- CES1→ Parametro `scadenza` superiore al timestamp attuale (valido)
- CES2→ Parametro `scadenza` inferiore al timestamp attuale (non valido)
- CES3→ Parametro `scadenza` nullo (non valido)
- CEI1→ Parametro `immagine` stringa vuota (valido)
- CEI2→ Parametro `immagine` stringa non vuota (valido)
- CEI3→ Parametro `immagine` null (non valido)

CEN1,CET1,CEC1,CED1,CES1,CEI1

```
public class ServiceAstaSilenziosaTest {

    @Mock
    AstaSilenziosaDAO astadao;

    @InjectMocks
    ServiceAstaSilenziosa service;

    @Test
    public void testCreaAsta() {
        // ARRANGE
        MockitoAnnotations.openMocks(this);

        when(astadao.creaNuovaAstaDaUtente(any(type:Utente.class))).thenReturn(value:1);

        Map<String, String> params = new HashMap<>();
        params.put(key:"nickname", value:"test");
        params.put(key:"titolo", value:"test");
        params.put(key:"categoria", value:"Elettronica");
        params.put(key:"descrizione", value:"test");
        params.put(key:"scadenza", value:"2025-12-31 11:22:00");
        params.put(key:"immagine", value:"");

        // ACT
        Integer result = service.creaAsta(params);

        // ASSERT
        assertEquals(Integer.valueOf(1), result);
    }
}
```

CEN1,CET1,CEC1,CED1,CES1,CEI2

```
@Test
public void testCreaAstaConImmagine() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.creaNuovaAstaConImmagine(any(type:Utente.class))).thenReturn(value:1);

    Map<String, String> params = new HashMap<>();
    params.put(key:"nickname", value:"test");
    params.put(key:"titolo", value:"test");
    params.put(key:"categoria", value:"Elettronica");
    params.put(key:"descrizione", value:"test");
    params.put(key:"scadenza", value:"2025-12-31 11:22:00");
    params.put(key:"immagine", value:"mybase64image");

    // ACT
    Integer result = service.creaAsta(params);

    // ASSERT
    assertEquals(Integer.valueOf(1), result);
}
```

CEN2,CET1,CEC1,CED1,CES1,CEI1

```
@Test
public void testCreaAstaConNicknameNullo() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.creaNuovaAstaDaUtente(any(type:Utente.class))).thenReturn(value:null);

    Map<String, String> params = new HashMap<>();
    params.put(key:"nickname", value:null);
    params.put(key:"titolo", value:"test");
    params.put(key:"categoria", value:"Elettronica");
    params.put(key:"descrizione", value:"test");
    params.put(key:"scadenza", value:"2025-12-31 11:22:00");
    params.put(key:"immagine", value:"");

    // ACT
    Integer result = service.creaAsta(params);

    // ASSERT
    assertNull(result);
}
```

CEN1,CET2,CEC1,CED1,CES1,CEI1

```
@Test
public void testCreaAstaConTitoloNullo() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.creaNuovaAstaDaUtente(any(type:Utente.class))).thenReturn(value:null);

    Map<String, String> params = new HashMap<>();
    params.put(key:"nickname", value:"test");
    params.put(key:"titolo", value:null);
    params.put(key:"categoria", value:"Elettronica");
    params.put(key:"descrizione", value:"test");
    params.put(key:"scadenza", value:"2025-12-31 11:22:00");
    params.put(key:"immagine", value:"");

    // ACT
    Integer result = service.creaAsta(params);

    // ASSERT
    assertNull(result);
}
```

CEN1,CET1,CEC2,CED1,CES1,CEI1

```
@Test
public void testCreaAstaConCategoriaNulla() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.creaNuovaAstaDaUtente(any(type:Utente.class))).thenReturn(value:null);

    Map<String, String> params = new HashMap<>();
    params.put(key:"nickname", value:"test");
    params.put(key:"titolo", value:"test");
    params.put(key:"categoria", value:null);
    params.put(key:"descrizione", value:"test");
    params.put(key:"scadenza", value:"2025-12-31 11:22:00");
    params.put(key:"immagine", value:"");

    // ACT
    Integer result = service.creaAsta(params);

    // ASSERT
    assertNull(result);
}
```

CEN1,CET1,CEC1,CED2,CES1,CEI1

```
@Test
public void testCreaAstaConDescrizioneNulla() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.creaNuovaAstaDaUtente(any(type:Utente.class))).thenReturn(value:null);

    Map<String, String> params = new HashMap<>();
    params.put(key:"nickname", value:"test");
    params.put(key:"titolo", value:"test");
    params.put(key:"categoria", value:"Elettronica");
    params.put(key:"descrizione", value:null);
    params.put(key:"scadenza", value:"2025-12-31 11:22:00");
    params.put(key:"immagine", value:"");

    // ACT
    Integer result = service.creaAsta(params);

    // ASSERT
    assertNull(result);
}
```


CEN1,CET1,CEC1,CED1,CES3,CEI1

```
@Test
public void testCreaAstaConScadenzaNulla() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.creaNuovaAstaDaUtente(any(type:Utente.class))).thenReturn(value:null);

    Map<String, String> params = new HashMap<>();
    params.put(key:"nickname", value:"test");
    params.put(key:"titolo", value:"test");
    params.put(key:"categoria", value:"Elettronica");
    params.put(key:"descrizione", value:"test");
    params.put(key:"scadenza", value:null);
    params.put(key:"immagine", value:"");

    // ACT
    Integer result = service.creaAsta(params);

    // ASSERT
    assertNull(result);
}
```

CEN1,CET1,CEC1,CED1,CES1,CEI3

```
@Test
public void testCreaAstaConImmagineNulla() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.creaNuovaAstaDaUtente(any(type:Utente.class))).thenReturn(value:null);

    Map<String, String> params = new HashMap<>();
    params.put(key:"nickname", value:"test");
    params.put(key:"titolo", value:"test");
    params.put(key:"categoria", value:"Elettronica");
    params.put(key:"descrizione", value:"test");
    params.put(key:"scadenza", value:"2025-12-31 11:22:00");
    params.put(key:"immagine", value:null);

    // ACT
    Integer result = service.creaAsta(params);

    // ASSERT
    assertNull(result);
}
```

CEN1,CET1,CEC1,CED1,CES2,CEI1

```
@Test
public void testCreaAstaConScadenzaInferioreAQuellaCorrente() {
    // ARRANGE
    MockitoAnnotations.openMocks(this);

    when(astadao.creaNuovaAstaDaUtente(any(type:Utente.class))).thenReturn(value:null);

    Instant timestamp = Instant.now().minusSeconds(secondsToSubtract:1);
    Map<String, String> params = new HashMap<>();
    params.put(key:"nickname", value:"test");
    params.put(key:"titolo", value:"test");
    params.put(key:"categoria", value:"Elettronica");
    params.put(key:"descrizione", value:"test");
    params.put(key:"scadenza", timestamp.toString());
    params.put(key:"immagine", value:null);

    // ACT
    Integer result = service.creaAsta(params);

    // ASSERT
    assertNull(result);
}
```

b. Valutazione dell'usabilità sul campo

La valutazione dell'usabilità sul campo consiste nel far testare alcune funzionalità del sistema a un campione di utenti, osservando come interagiscono con l'applicativo. In questo modo il progettista può ricavare indicazioni concrete per il miglioramento sia del design che del sistema in sé.

Metodologia utilizzata

L'interazione tra l'utente e il sistema può essere effettuata attraverso diverse metodologie. Abbiamo optato per l'approccio noto come "Mago di Oz". In questo scenario, gli utenti vengono incaricati di eseguire specifici compiti legati a determinate funzionalità del sistema, il tutto sotto la supervisione di uno o più osservatori. Nel nostro caso, uno di noi svolgerà il ruolo dell'osservatore.

Per quanto riguarda la simulazione del sistema, esistono diverse opzioni. La nostra scelta è stata quella di fornire l'applicativo direttamente ai tester. Questo significa che non avremo bisogno di un terzo individuo per simulare l'applicazione.

Inoltre, abbiamo previsto la presenza di un facilitatore. Questa figura avrà il compito di assistere l'utente nel caso in cui si trovi in difficoltà durante l'esecuzione dei task. Anche in questo caso, uno di noi assumerà il ruolo del facilitatore. Questo approccio ci permette di mantenere un controllo diretto sul processo di testing e di intervenire prontamente in caso di problemi.

Tester e Task

Non avendo conoscenze con persone esperte del campo o che abbiano già avuto esperienza con l'usabilità del software, abbiamo chiesto ad alcuni amici di darci una mano per testare la nostra applicazione finita.

Familiarità con la tecnologia	Alta	Tester: 1,5,6,7,8	Tester: 2,3
	Bassa	Tester: 4	
		Bassa	Alta
		Conoscenza del dominio	

In particolare, abbiamo chiesto loro di svolgere gli stessi task utilizzati per l'usabilità a priori, verificando così anche le differenze ([1-a. v](#)):

1. VISUALIZZARE PROFILO CREATORE DELL'ASTA;
2. CREARE UN'ASTA SILENZIOSA.

A differenza però dell'usabilità a priori, dove non era possibile autenticarsi e quindi quella fase veniva sorvolata, in questo caso, prima di entrare nel vivo del primo task, viene richiesto di creare un profilo utente e proseguire poi con il compito da eseguire. Nel secondo task abbiamo invece lasciato libera scelta ai valutatori, che potevano scegliere se accedere con lo stesso account o crearne un altro. Di seguito riportiamo la descrizione dei task così come sono stati presentati ai nostri tester:

1. Non avendo ancora un account, cliccare su registrati in alto o in basso. Compilare i campi come meglio si preferisce per personalizzare il proprio account. Dopo aver compilato i vari campi, e aver eseguito con successo la creazione dell'account, selezionare l'account compratore in modo da visualizzare la schermata home. Qui scegliere un'asta tra quelle presenti nella home e una volta visualizzati i dettagli dell'asta cliccare sul nome del proprietario per visualizzarne il profilo. A questo punto il goal è stato raggiunto.

2. Libera scelta sull'autenticazione (accedere all'account creato nel task precedente oppure creare un nuovo account). Una volta autenticato selezionare l'account venditore e nella schermata home cliccare in basso sul più per visualizzare la schermata di creazione di una nuova asta. Qui è facoltativo inserire un'immagine. Una volta compilati i vari campi cliccare su crea. A questo punto il goal è stato raggiunto.

Analisi dei risultati

In totale questi task sono stati svolti da 8 tester. Per 4 di loro, che sono stati visionati in prima persona con la tecnica riportata [sopra](#), sono riusciti tutti a completare facilmente i task, lasciando feedback solo su aspetti di design dell'applicativo. Abbiamo preso nota di questi suggerimenti e abbiamo apportato modifiche al design dell'applicazione, incorporando le loro idee e mantenendo fedeltà ai nostri ideali. Per gli altri 4 che hanno svolto i task da remoto (quindi senza essere osservati da nessuno di noi), alcuni hanno impiegato un paio di minuti in più nel completarli, ma purtroppo non hanno lasciato alcun feedback riguardante il motivo di tale ritardo o riguardante migliorie per l'applicativo. Nonostante ciò, hanno comunque completato i task a loro assegnati, quindi per coloro che hanno impiegato un tempo maggiore abbiamo pensato di attribuire un successo parziale del task (P). Di seguito riportiamo il tasso di successo dei task:

LEGENDA:

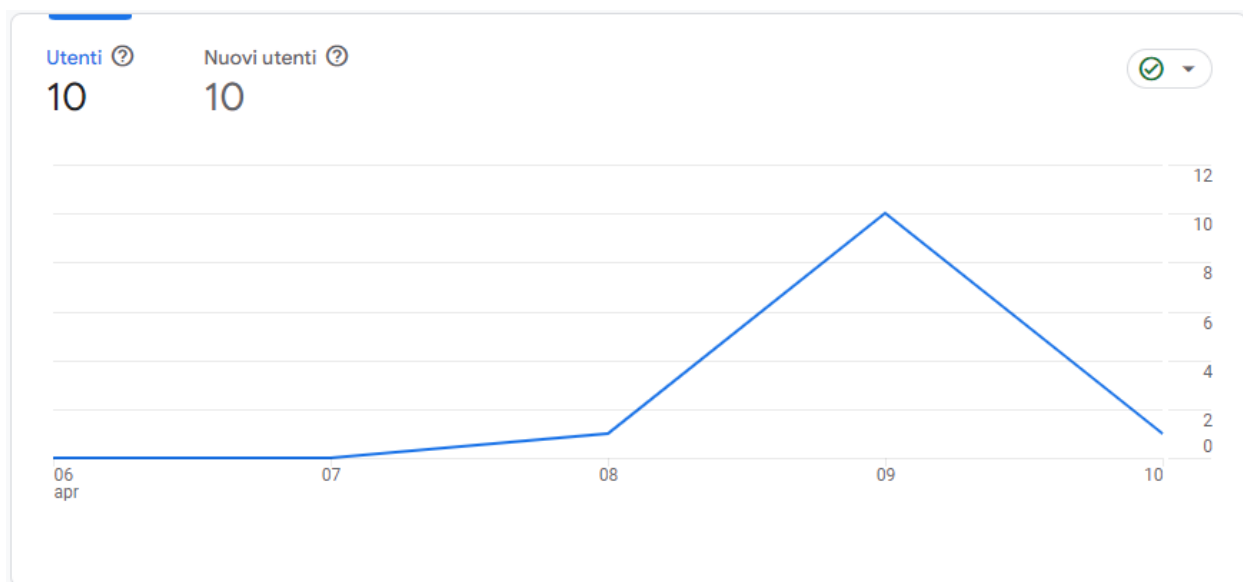
S = success; P = partial success; F = failed.

TASK	1 TESTER	2 TESTER	3 TESTER	4 TESTER	5 TESTER	6 TESTER	7 TESTER	8 TESTER
1 (VPCA)	S	S	S	S	P	S	S	S
2 (CAS)	S	S	S	S	P	S	S	P

Il tasso di successo si calcola facendo il rapporto tra la somma del numero di successi moltiplicati per 1 e del numero dei successi parziali moltiplicati per 0,5 e il numero totale dei task, quindi nel nostro caso:

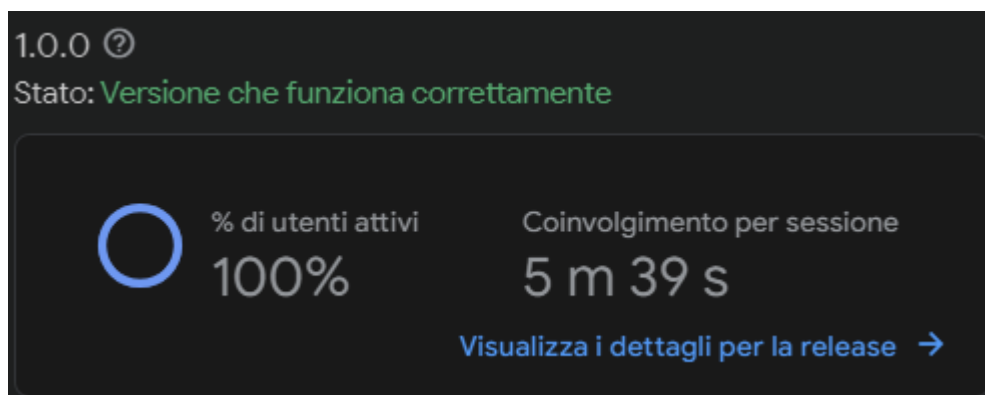
$$\text{Tasso di successo} = \frac{(13 * 1 + 3 * 0,5)}{16} = 0,90625 \cong 90\%$$

Per esaminare il comportamento degli utenti e i vari accessi all'applicativo abbiamo sfruttato Google Analytics, che fornisce informazioni dettagliate sull'utilizzo del prodotto software da parte degli utenti. Di seguito riportiamo alcune statistiche di utilizzo disponibili in Firebase:

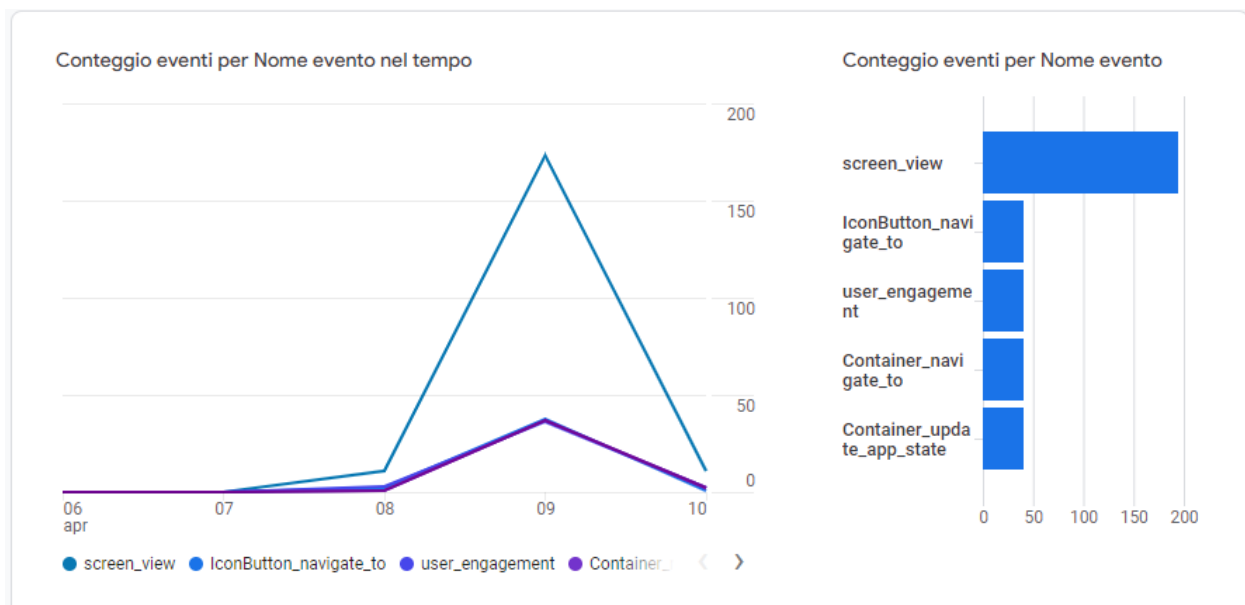


Come si evince dal grafico, gli accessi all'app sono 10 perché, oltre agli 8 tester che hanno svolto i task, ci sono anche gli accessi effettuati da noi. Inoltre, tutti hanno effettuato l'accesso da dispositivi Android:

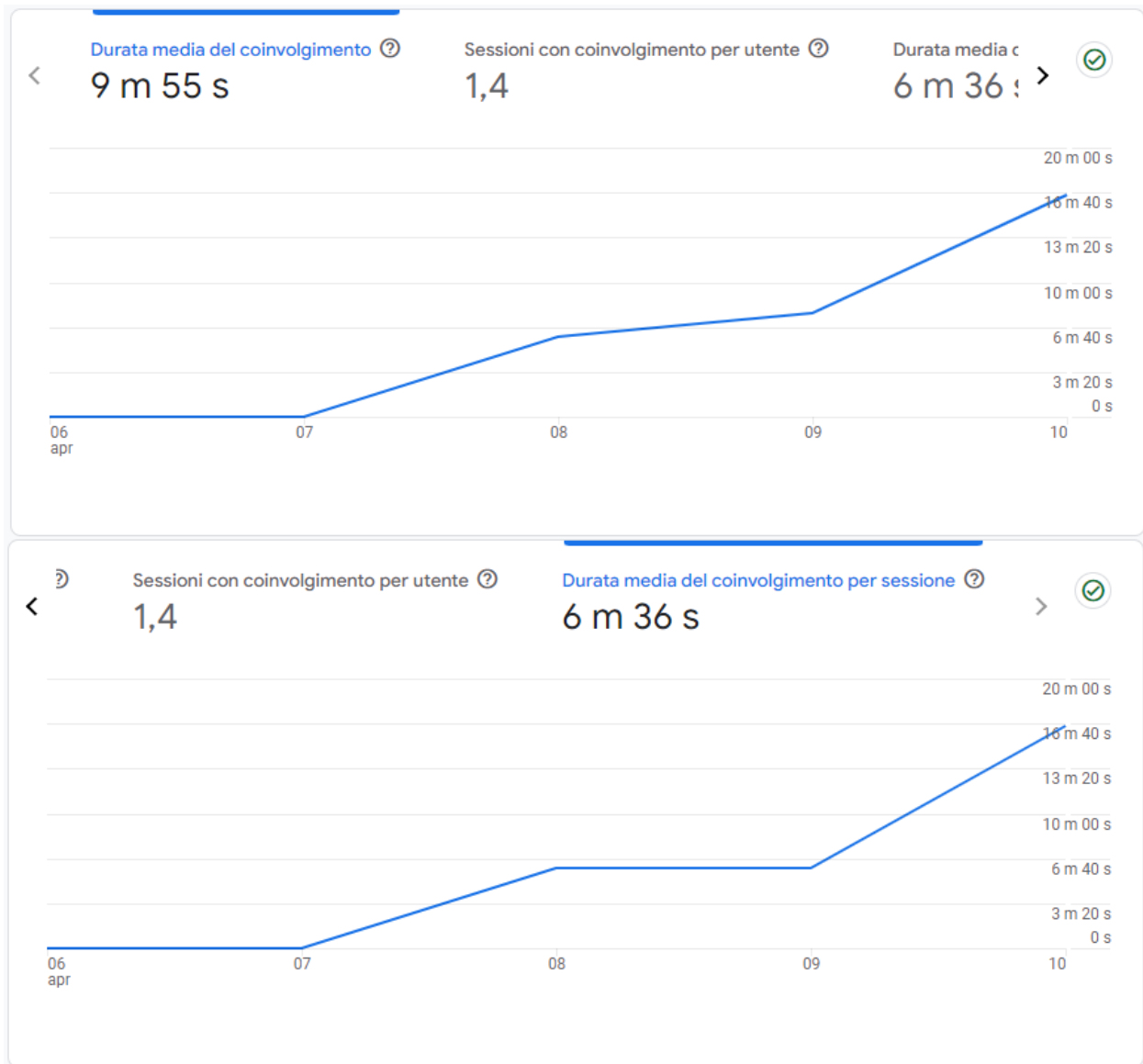
La versione scaricata dai tester non ha riscontrato alcun arresto anomalo:



Di seguito invece riportiamo un conteggio degli eventi fatti dagli utenti:



Per quanto riguarda invece le tempistiche:



c. Production-ready product

[Qui](#) è disponibile un video che mostra le principali funzionalità dell'applicazione pronta per l'utilizzo.