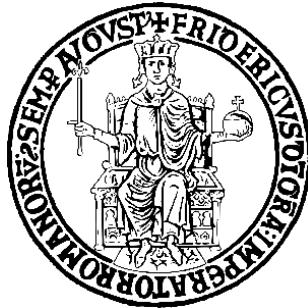


Università degli Studi di Napoli Federico II Scuola
Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione



Corso di Laurea in Informatica - Insegnamento di Ingegneria del Software

Anno Accademico 2023/2024

Progettazione di “DietiDeals24”, una
Piattaforma per la Gestione di Aste Online

Antonio Abbatiello - N86003037

Vincenzo Marotta - N86004151

Leonardo Colamarino - N86003586

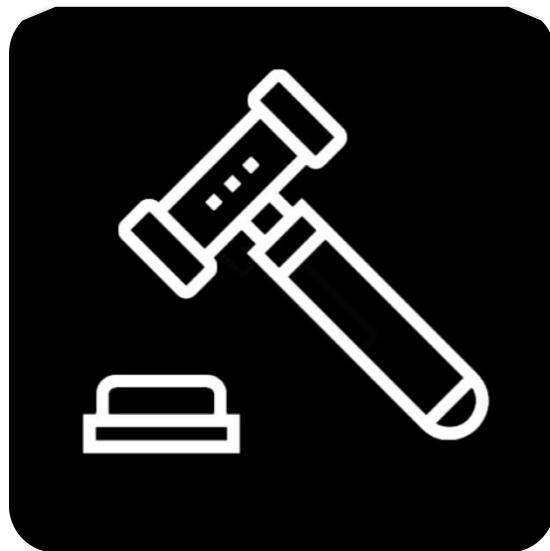
Sommario

1. Introduzione al Progetto.....	5
1.1 DietiDeals24.....	5
2. Documento dei Requisiti Software	7
2.1 Analisi dei Requisiti	7
2.1.1 Lista dei Requisiti Funzionali	7
2.1.2 Modellazione dei Casi d’Uso.....	11
2.1.3 Tabelle Cockburn.....	14
2.1.4 Prototipi Mock-Up.....	19
2.1.5 Individuazione del Target degli Utenti	30
2.1.6 Valutazione dell’Usabilità in Fase di Analisi	33
2.1.7 Glossario.....	37
2.2 Specifica dei Requisiti.....	40
2.2.1 Diagrammi di Classi di Analisi.....	40
2.2.2 Diagrammi di Sequenza di Analisi	47
2.2.3 Statechart dell’Interfaccia Grafica	50
3. Documento del Design di Sistema.....	53
3.1 Descrizione dell’architettura proposta	53
3.2 Sviluppo Front-end	53
3.3 Sviluppo Back-end	55
3.4 Diagrammi di Classi di Design	56
3.5 Diagrammi di Sequenza di Design.....	60
4. Versioning e Report della Qualità	63
4.1 Uso di Strumenti di Versioning.....	63
4.2 Report della Qualità del Codice	64
5. Testing e Valutazione dell’Usabilità.....	65
5.1 Unit Testing.....	65
5.2 Valutazione dell’Usabilità.....	75
5.2.1 Fase di Valutazione.....	75
5.2.2 Analisi File di Log	81

Capitolo 1

1. Introduzione al Progetto

1.1 DietiDeals24



Il progetto Dietideals24 consiste nella realizzazione di una piattaforma completa e intuitiva per la **gestione di aste online**, che permette agli utenti di partecipare e organizzare aste di diverso tipo, effettuare offerte per beni/servizi e monitorare lo stato delle proprie transazioni, il tutto tramite un'interfaccia user-friendly accessibile tramite dispositivi mobile iOS.

La piattaforma permette ad utenti di creare account per la compravendita, inserendo i dati personali nella schermata di autenticazione. In base alle proprie esigenze, un utente potrà decidere se l'utilizzo della piattaforma Dietideals24 sarà esclusivamente per **l'acquisto di beni e la partecipazione alle aste**, oppure se dovrà includere anche funzionalità specifiche ai venditori di beni.

Un utente compratore o venditore sarà in grado di controllare e modificare il proprio profilo, visualizzare le proprie aste attive, consultare aste correntemente attive di altri venditori e avere la possibilità di presentare offerte in denaro (nei margini delle soglie definite dal tipo dell'asta). Inoltre, gli utenti partecipanti ad un'asta saranno avvisati al termine di tale asta, tramite un sistema di notifiche integrato nell'applicazione mobile.

Il sistema è composto da **un'applicazione client front-end** disponibile come interfaccia utente, e da **un'applicazione server back-end** per la gestione del database sottostante. La comunicazione tra back-end e front-end sarà resa possibile dalle interfacce di REST API fornite dal sistema server.

Capitolo 2

2. Documento dei Requisiti Software

2.1 Analisi dei Requisiti

2.1.1 Lista dei Requisiti Funzionali

Di seguito sono elencati e categorizzati i requisiti funzionali, ottenuti tramite i colloqui con gli stakeholder e un successivo rifinimento di dettagli da parte del team di sviluppo di DietiDeals24.

Categoria	Requisiti
1. Registrazione/Accesso	<ul style="list-style-type: none">• Un utente nuovo deve poter registrare un account di tipo “compratore”, ed usare tale account per accedere al sistema.• Un account “compratore” dovrà fornire solo le funzionalità di un account per compratori.• L’indirizzo e-mail associato ad un account è unico nel sistema.• Un’utente già registrato con un account “compratore” avrà la possibilità di promuovere tale account ad un account “venditore”, fornendo informazioni aggiuntive al sistema(Partita IVA, Ragione sociale, ...).• Un account “venditore” dovrà fornire sia le funzionalità di un account per compratori che per venditori.• La registrazione e l’accesso potranno essere effettuati tramite metodologia classica o utilizzando servizi esterni (Google, Facebook, GitHub, etc...).

	<ul style="list-style-type: none"> • Un utente avrà a disposizione una sezione per personalizzare il proprio profilo (Biografia, Area geografica, Link a siti web esterni...).
2. Specifica Aste	<ul style="list-style-type: none"> • Il sistema deve permettere ai compratori di creare delle aste per l'acquisto di beni/servizi. • Il sistema deve permettere ai compratori di effettuare delle offerte per le aste correntemente attive di altri venditori. • Il sistema deve permettere ai venditori di creare delle aste per la vendita di beni/servizi. • Il sistema deve permettere ai venditori di effettuare delle offerte per le aste correntemente attive di altri compratori. • Ogni asta dev'essere caratterizzata da una categoria di appartenenza, una descrizione del bene/servizio e da fotografia opzionale.
3. Ricerca e Filtraggio Aste	<ul style="list-style-type: none"> • Il sistema deve presentare all'utente un sistema di ricerca delle aste disponibili tramite filtraggio di categoria e/o parole chiave. • A ricerca effettuata dovrà essere possibile visualizzare i dettagli di ciascuna asta e del profilo dell'utente.
4. Aste a Tempo Fisso	<ul style="list-style-type: none"> • Un utente deve poter creare un asta a tempo fisso con data di scadenza e una soglia minima segreta (visualizzabile solo dal venditore) a cui poter vendere il bene/servizio. • Un utente può visualizzare il valore in € dell'ultima offerta presentata, se presente, e può effettuare delle offerte in €, con l'importo offerto necessariamente maggiore rispetto all'ultima offerta (coincide con la migliore offerta corrente).

	<ul style="list-style-type: none"> • Se l'asta va a buon fine, l'offerente con l'offerta più alta si aggiudica il bene/servizio se e solo se raggiunge la soglia minima segreta. • Se la soglia non viene raggiunta entro il termine della data di scadenza, l'asta fallisce. • In conclusione di un'asta a tempo fisso sara' mandata una notifica all'organizzatore dell'asta e a tutti gli offerenti (se presenti).
5. Aste all'Inglese	<ul style="list-style-type: none"> • Un utente deve poter creare un'asta all'inglese dove va indicato: <ul style="list-style-type: none"> ◦ un prezzo di partenza (in €) ◦ il tempo massimo T tra un'offerta e l'altra (default 1 ora) ◦ una soglia di rialzo minima S (default 10€) • Un offerente puo' effettuare un offerta in € rispetto all'offerta corrente, a condizione che sia passato T tempo dall'ultima offerta e che l'offerta sia un valore multiplo di S. • Nel momento in cui è presentata la prima offerta per l'asta, e' avviato un timer. Per ogni offerta presentata prima della scadenza del tempo, il timer si resetta e l'offerta corrente si aggiorna. • Quando il timer raggiunge lo 0 l'asta si chiude, il bene/servizio viene aggiudicato all'ultimo offerente e arriva una notifica all'organizzatore dell'asta e a tutti gli offerenti (se presenti).
6. Aste al Ribasso	<ul style="list-style-type: none"> • Un utente può creare un'asta al ribasso dove va indicato: <ul style="list-style-type: none"> ◦ un prezzo di partenza elevato (in €) ◦ un timer di decremento del prezzo (default 1 ora) ◦ un importo (in €) per ciascun decremento P ◦ un prezzo minimo segreto per l'asta M

- | | |
|--|---|
| | <ul style="list-style-type: none">• Il prodotto dovrà essere inizialmente piazzato al prezzo di partenza elevato.• Alla creazione dell'asta, sarà avviato il timer. Ogni volta che scade, il prezzo corrente sarà decrementato di P euro.• Il bene/servizio verrà aggiudicato al primo offerente.• Se il prezzo del prodotto raggiunge l'importo minimo M, l'asta fallirà.• In conclusione sarà mandata una notifica all'organizzatore dell'asta e all'offerente (se presente). |
|--|---|

2.1.2 Modellazione dei Casi d'Uso

Per ricavare un diagramma di casi d'uso partendo dai requisiti funzionali del sistema, abbiamo modellato due schemi:

- Uno schema per l'accesso/registrazione di un utente al sistema
- Uno schema per l'utilizzo delle funzionalità dell'applicazione di un utente registrato

Tali schemi sono stati progettati tramite l'utilizzo di StarUML, uno strumento apposito per la progettazione di modelli e schemi UML.

Diagramma Use-Case per il sistema di autenticazione di DietiDeals24

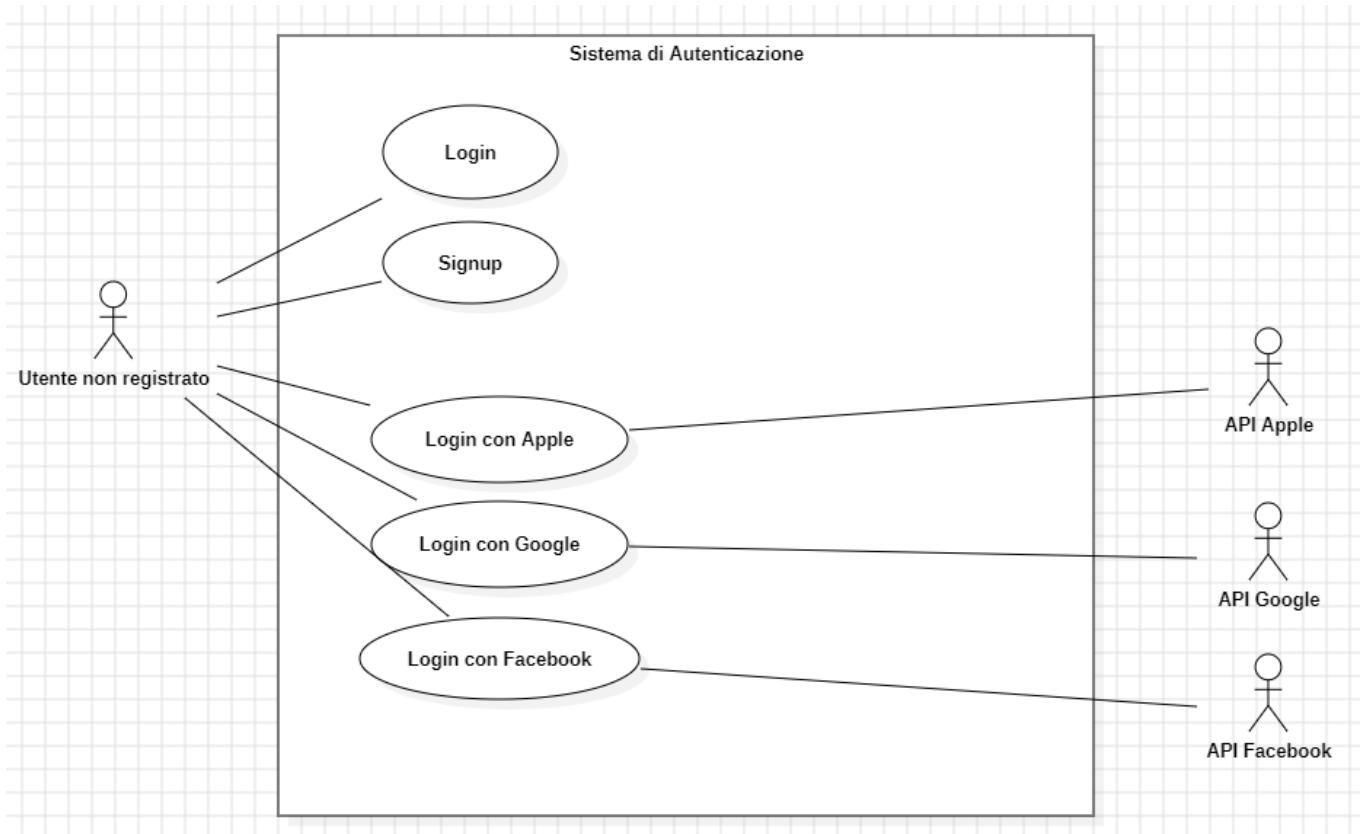
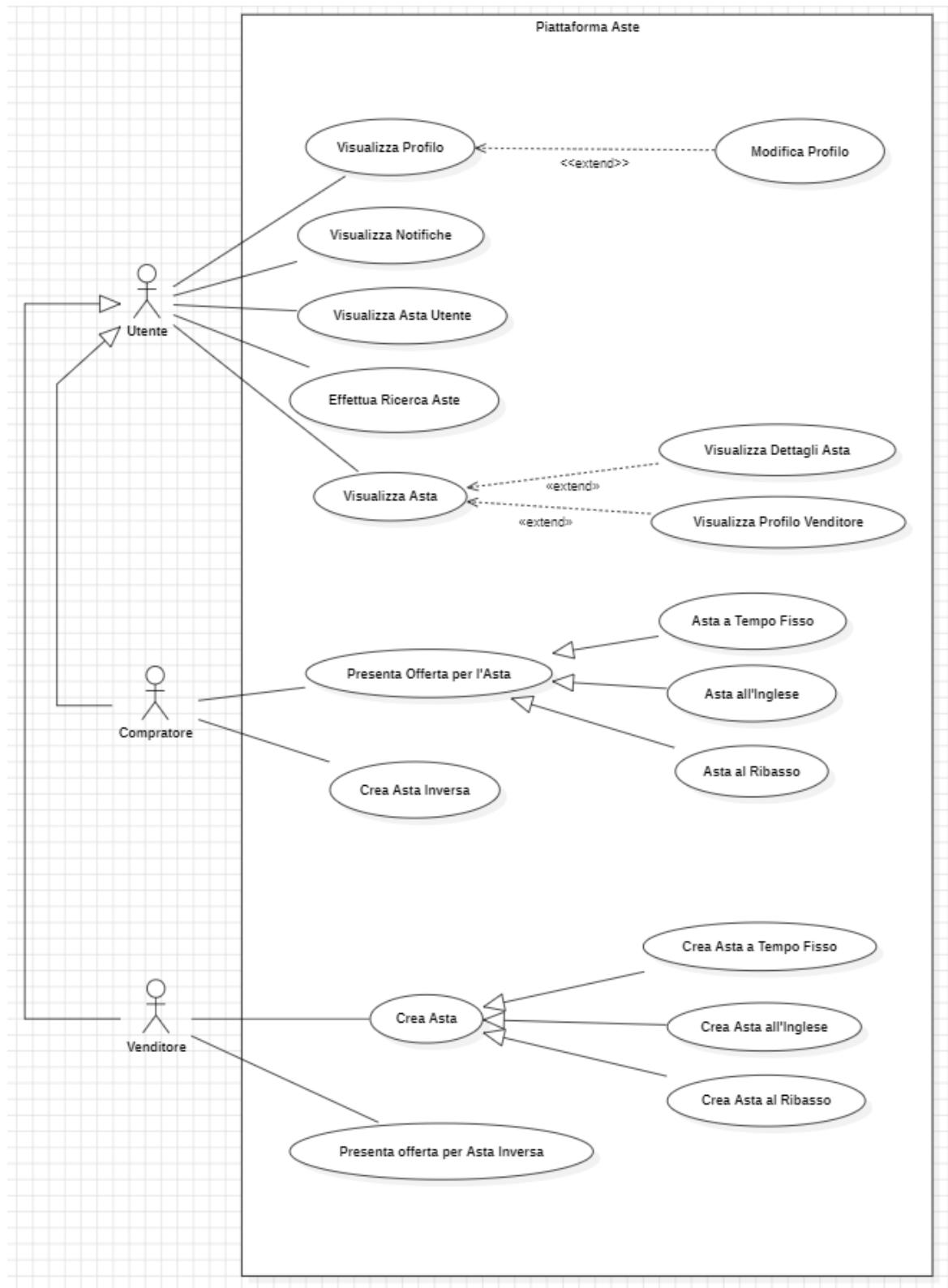


Diagramma Use-Case per l'interazione con la piattaforma aste di DietiDeals24



2.1.3 Tabelle Cockburn

Sono disponibili inoltre descrizioni testuali dettagliate, in forma di tabelle di Cockburn, per due casi d'uso dell'applicazione. Le tabelle di Cockburn, introdotte da Alistair Cockburn, forniscono una struttura formale e organizzata per esprimere nei minimi dettagli le entità, le relazioni e i passi necessari per un caso d'uso in particolare.

I casi d'uso scelti sono:

1. **Use Case #1:** Creazione di un'asta a tempo fisso
2. **Use Case #2:** Presentazione di un'offerta per asta a tempo fisso

Tabella Cockburn per Use Case N. 1

USE CASE #1	NOME UC: Crea Asta a Tempo Fisso		
Goal in Context	Il venditore vuole creare una nuova asta a tempo fisso per un prodotto/servizio, allo scopo di vendere il prodotto/servizio al miglior offerente.		
Preconditions	Il venditore deve aver effettuato la registrazione di un account per vendori. Inoltre, deve effettuare il login con tale account.		
Success End Condition	L'asta a tempo fisso viene creata con successo ed è resa visibile a tutti gli utenti, compreso il venditore stesso.		
DESCRIPTION	Step n°	Attore venditore	Sistema
	1	Preme il pulsante “My Auctions” nella barra di navigazione	
	2		Mostra la schermata “My Auctions”
	3	Preme il pulsante “+” in alto a destra	
	4		Mostra la schermata di selezione tipo di asta
	5	Preme il pulsante “Fixed Time Auction”	
	6		Mostra la schermata di creazione asta a tempo fisso
	7	Compila i campi	
	8	Preme il pulsante “Next”	
	9		Mostra una pagina che mostra un riassunto dell'asta e dettagli associati a essa
	10	Preme il pulsante “Publish”	
	11		Il sistema pubblica l'asta e la rende visibile a tutti gli utenti della piattaforma
EXTENSIONS	Step	Attore venditore	Sistema
Il venditore vuole inserire una foto	7.1	Inserisce una foto (facoltativa)	

SUBVARIATIONS	Step	Attore venditore	Sistema
Il venditore vuole tornare indietro	7A.1	Preme il pulsante "Back"	
	7A.2		<i>Mostra la pagina precedente</i>
Il venditore vuole tornare indietro	8A.1	Preme il pulsante "Back"	
	87A.2		<i>Mostra la pagina precedente</i>
Il venditore vuole tornare indietro	10A.1	Preme il pulsante "Back"	
	10A.2		<i>Mostra la pagina precedente</i>

Tabella Cockburn per Use Case N. 2

USE CASE #2	NOME UC: Presenta Offerta per l'Asta a Tempo Fisso		
Goal in Context	Il compratore vuole proporre un'offerta per un prodotto/servizio venduto in un'asta a tempo fisso.		
Preconditions	Il compratore deve aver effettuato la registrazione di un account per compratori. Inoltre, deve effettuare il login con tale account.		
Success End Condition	L'offerta per l'asta a tempo fisso viene accettata con successo e l'asta viene aggiornata in modo appropriato, assegnando all'ultima offerta l'offerta del compratore.		
DESCRIPTION	Step n°	Attore venditore	Sistema
	1	Preme il pulsante "Auctions" nella barra di navigazione	
	2		Mostra la schermata "Auctions"
	3	Preme il pulsante "Fixed Time" per filtrare le aste per tipo	
	4		Aggiorna la pagina mostrando solo aste a tempo fisso
	5	Seleziona l'asta di interesse	
	6		Mostra la schermata di visualizzazione dell'asta
	7	Preme il pulsante "Offer"	
	8	Inserire l'importo desiderato in €	
	9	Preme il pulsante "Make Offer" per effettuare l'offerta	
	10		Il sistema accetta l'offerta e aggiorna i campi di ultima offerta in modo appropriato
EXTENSIONS	Step	Attore venditore	Sistema
Il compratore vuole filtrare le aste per titolo o categoria	3.1	Inserisce una keyword nel campo di ricerca	
	3.2		Aggiorna la pagina mostrando solo aste a tempo fisso, che rispettano i criteri definiti dalle keyword inserite dall'utente

Il compratore vuole visualizzare il profilo del venditore	5.1	Preme sul pulsante di profilo utente	
	5.2		Mostra la pagina profilo del venditore
SUBVARIATIONS	Step	Attore venditore	Sistema
Il venditore vuole tornare indietro	5A.1	Preme il pulsante "Back"	
	5A.2		<i>Mostra la pagina precedente</i>
Il venditore vuole tornare indietro	7A.1	Preme fuori dalla schermata	
	7A.2		<i>Mostra la pagina precedente</i>

2.1.4 Prototipi Mock-Up

Di seguito sono mostrati dei prototipi mock-up di schermate dell'applicazione, creati usando l'applicazione Figma.

Tali prototipi sono utili per ottenere una visualizzazione dell'interfaccia utente con il sistema, quando il prodotto software non è ancora disponibile. Questo permette a stakeholder di avere un modello dimostrativo del prodotto finale e di formulare delle aspettative, dalle quali si passa ad un rifinimento di requisiti.

I casi d'uso scelti sono:

1. **Use Case #1:** Creazione di un'asta a tempo fisso
2. **Use Case #2:** Presentazione di un'offerta per asta a tempo fisso

Prototipo Mock-Up per Use Case N. 1

Sono riportate le seguenti schermate:

- Schermata “Profilo”
- Schermata “Le Mie Aste”
- Selezione di asta da creare
- Creazione di un’asta a tempo fisso
- Inserimento di dati per la creazione dell’asta
- Visualizzazione di un sommario dell’asta da pubblicare
- Popup di pubblicazione con successo dell’asta
- Schermata “Le Mie Aste” (ora contenente l’asta appena pubblicata)
- Visualizzazione asta a tempo fisso

05:39

05:39

+

Profile



Modify

Prova Prova

prova@email.it

Sign Out

Auctions Notifications My Auctions Profile

My Auctions

 **Fiume**
31/03/2024, 0:29:13 **0,00 €**

 **Erba**
31/03/2024, 1:34:16 **0,00 €**

 **Foglia**
31/03/2024, 1:39:17 **0,00 €**

 **Foglia**
31/03/2024, 1:39:17 **0,00 €**

 **Acqua**
31/03/2024, 1:42:40 **0,00 €**

Auctions Notifications **My Auctions** Profile

05:39

05:39

Back

Back

Auction Type

Fixed Time Auction

 Fixed Time Auction



Add picture

 English Auction

Title*

 Descending Price Auction

Description

0/1000

Category: [Automotive](#)

 Inverse Auction

Expiry Date: [31 Mar 2024](#)

06:39

Minimum Price (default 0)

Next

05:40

Back

Fixed Time Auction



Add picture

cascate

Cascade private

15/1000

Category: Beauty

Expiry Date: 31 Mar 2024 08:39

1.000,00 €

Next

05:40

Back

Summary



cascate

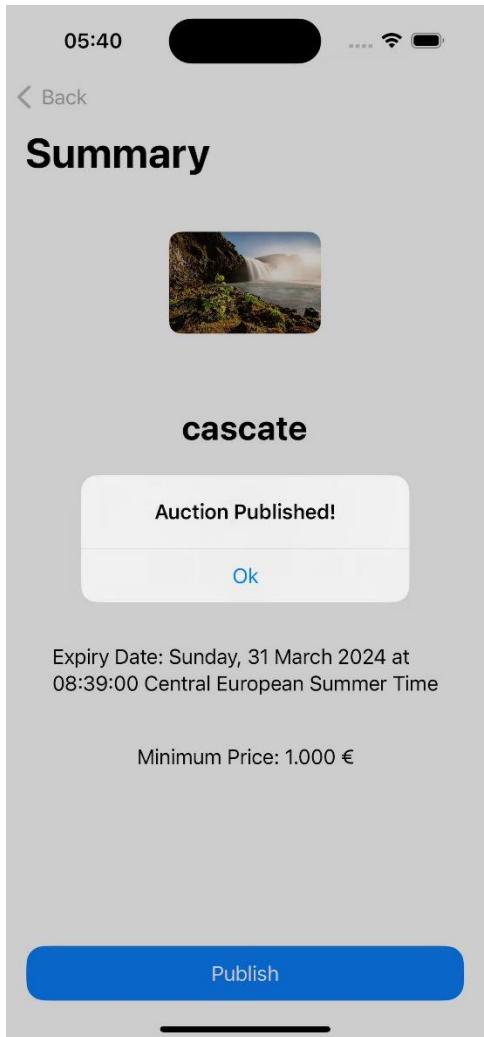
Cascade private

Beauty

Expiry Date: Sunday, 31 March 2024 at 08:39:00 Central European Summer Time

Minimum Price: 1.000 €

Publish



05:40



...



Back



cascade

Beauty

Username

Cascade private

0,00 €

Delete

Prototipo Mock-Up per Use Case N. 2

Sono riportate le seguenti schermate:

- Schermata “Profilo”
- Schermata “Aste”
- Schermata di visualizzazione asta e di presentazione offerta
- Popup di presentazione offerta
- Popup di offerta avvenuta con successo
- Schermata di visualizzazione asta (ora con ultima offerta aggiornata)

05:39

Profile

Stroking

[Modify](#)

Prova Prova

prova@email.it

05:40

Auctions

12 hours, 57 minutes

Fiore **34,00 €**
17 hours, 22 minutes

Fiore **0,00 €**
17 hours, 22 minutes

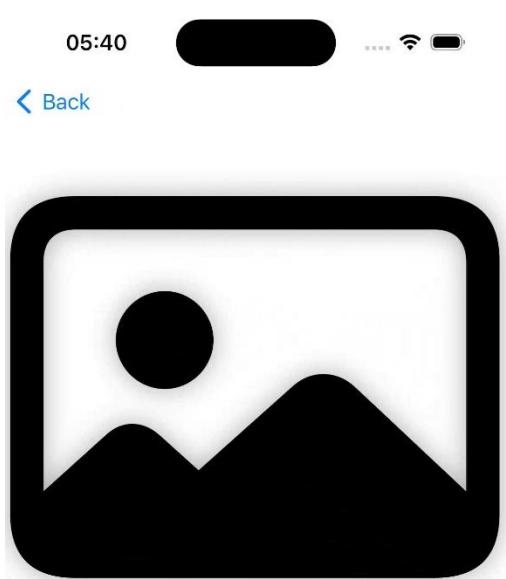
Fiore **0,00 €**
17 hours, 22 minutes

Cascata **0,00 €**
1 day, 20 hours

Sign Out

Auctions Notifications My Auctions Profile

Auctions Notifications My Auctions Profile



Fiore

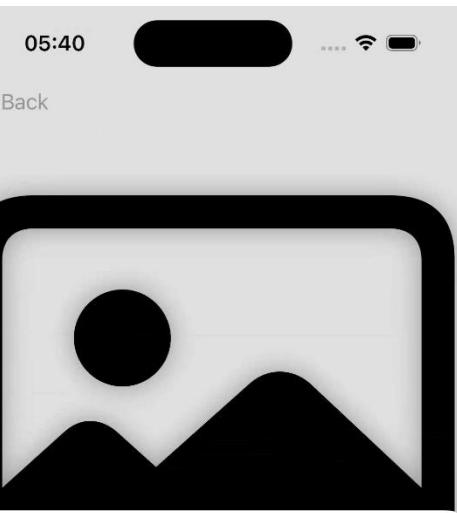
Groceries

Username

Fiorw e bianco

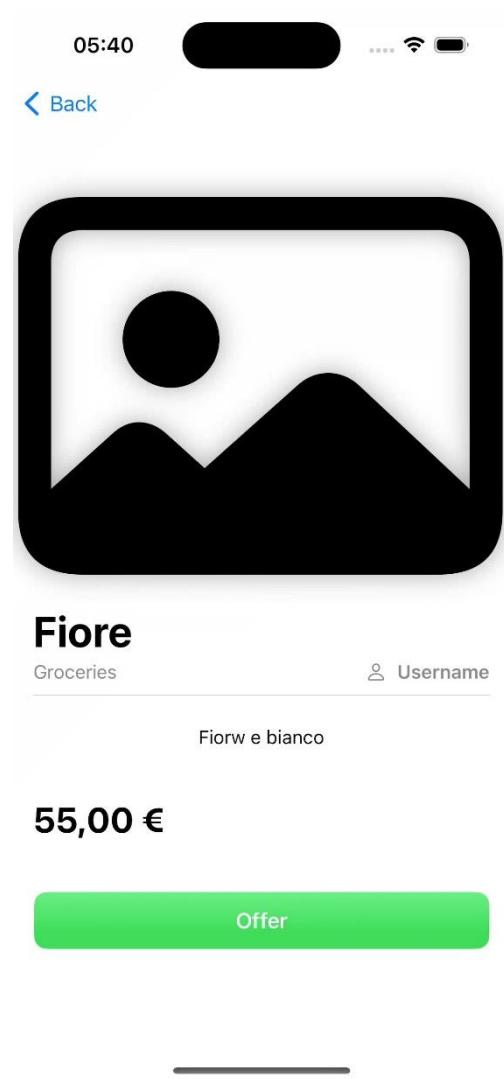
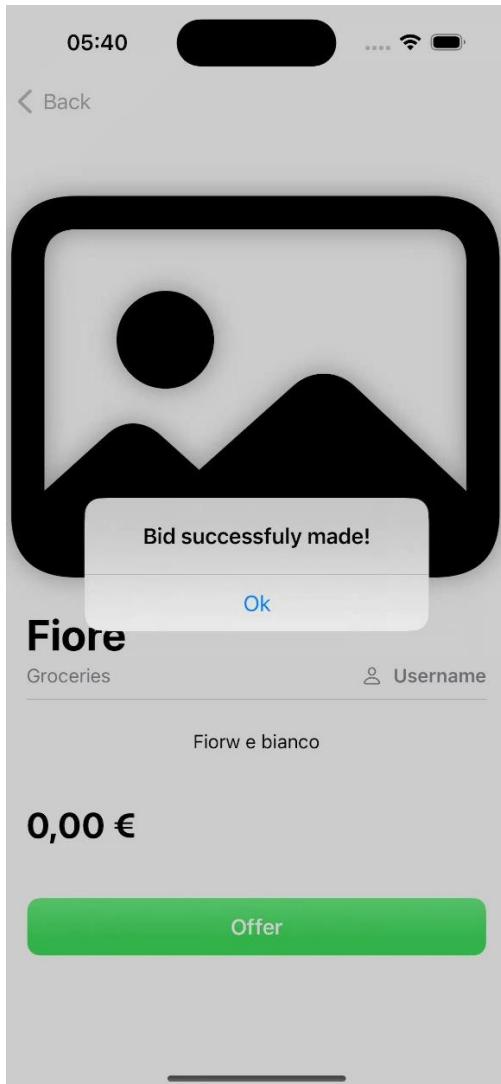
0,00 €

Offer



55,00 €

Make Offer



2.1.5 Individuazione del Target degli Utenti

L'individuazione del target di utenti è un passo fondamentale nello sviluppo di qualsiasi prodotto software, in quanto consente di definire le caratteristiche, le esigenze e i comportamenti degli utenti a cui ci si rivolge. Nel contesto di DietiDeals24, l'individuazione del target di utenti è cruciale per garantire che la piattaforma sia progettata in modo da soddisfare le esigenze specifiche del pubblico di destinazione.

Ci siamo confrontati con persone interessate alla compravendita online di prodotti di seconda mano a prezzi convenienti, e dal conforto della propria residenza. Tra gli interessati, abbiamo incontrato utenti con una certa dose di esperienza in piattaforme di gestione aste già esistenti, come Ebay, Subito ed Etsy, mentre altri utenti non avevano familiarità con app del genere, ma erano comunque interessati all'idea.

Questo ha permesso di **determinare in modo più coesivo e personale le funzionalità e le caratteristiche che la piattaforma di aste dovrebbe implementare**, in modo tale da assicurare un utilizzo semplice e facile da apprendere sia per inesperti che per venditori e compratori provenienti da altre piattaforme.

I colloqui con gli interlocutori interessati hanno portato allo sviluppo di un sistema di interazione con utente ispirata in grosso modo all'interfaccia mobile della piattaforma e-commerce di Ebay.

Una volta definito il target utenti, abbiamo creato 3 modelli **“Personas”** che si avvicinano alle tipologie di utenti medi che userebbero una piattaforma come DietiDeals24.



Carla Cassiano

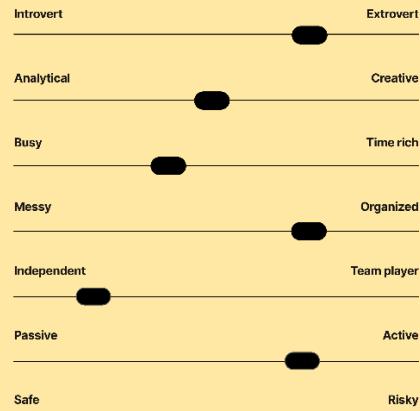
"Imprenditrice esperta"

- Età: 36 anni
- Occupazione: Imprenditrice nel settore dell'e-commerce
- Luogo: Torino, Italia

Bio

Carla Cassiano è un imprenditrice dinamica e appassionata, sempre alla ricerca di nuove opportunità per espandere il suo business e accrescere la sua collezione personale. Con una vasta esperienza nel settore dell'e-commerce, Carla comprende l'importanza di utilizzare le piattaforme online per scoprire e acquisire articoli unici e di valore.

Personality



Martina Bianchi

"Trova bellezza nelle piccole cose"

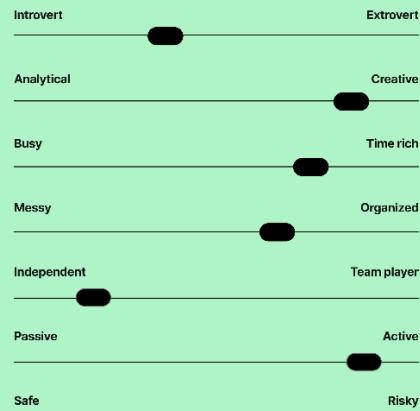
- Età: 28 anni
- Occupazione: Graphic Designer freelance
- Luogo: Berlino, Germania

Bio

Martina è una giovane professionista nel campo del design grafico, appassionata di arte e creatività. Essendo sempre alla ricerca di ispirazione per i suoi progetti e affari interessanti, Martina utilizza regolarmente app come eBay per partecipare a aste online e trovare articoli unici e di valore.

Martina è attratta soprattutto da articoli vintage e d'epoca, che spesso utilizza come fonte di ispirazione per i suoi progetti artistici.

Personality





Alessandro De Luca

"Manager presso una società di consulenza"

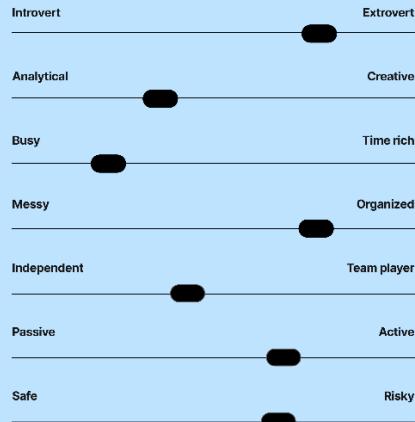
- Età: 45 anni
- Occupazione: Manager presso una società di consulenza
- Luogo: Milano, Italia

Bio

Alessandro è un uomo di mezza età con una carriera consolidata e una passione per l'arte e gli oggetti di valore. Ha una vita professionale impegnativa e spesso stressante, ma trova sollievo e gratificazione nel suo tempo libero attraverso il collezionismo e la partecipazione a aste online.

Essendo un appassionato collezionista, Alessandro è sempre alla ricerca di nuove opportunità per investire in oggetti di valore e rarezze. Utilizza servizi di aste online per scoprire e acquistare oggetti d'antiquariato, sfruttando la comodità e l'accessibilità offerta dalle piattaforme digitali.

Personality



2.1.6 Valutazione dell’Usabilità in Fase di Analisi

Per effettuare una valutazione completa dell’usabilità di DietiDeals24, è necessario dividere la valutazione in due fasi:

1. Una fase di valutazione euristica o di **valutazione dell’usabilità in fase di analisi**.
2. Una fase detta test dell’usabilità o di **valutazione dell’usabilità sul campo**.

Questa sezione si baserà sulla prima fase, e saranno mostrati i test di **valutazione dell’usabilità a priori**.

La fase di valutazione euristica è utile per identificare problemi in fase di prototipazione del modello software. In questa fase verranno eseguite valutazioni da parte di esperti di usabilità, appartenenti al team di sviluppo DietiDeals24, senza la presenza o la partecipazione di enti esterni come stakeholders o utenti.

La prima fase viene effettuata dopo la progettazione dei prototipi per l’interfaccia utente. Con un prototipo completo, è possibile effettuare test di usabilità per vari motivi:

- **Identificare problemi precoci** nell’interfaccia, che possono essere risolti senza dover modificare il codice del software sottostante (che non è ancora presente siccome siamo in fase di prototipazione)
- **Ottimizzare il design** dell’interfaccia e migliorare l’esperienza dell’utente

1. Feedback generali sull'applicazione

Inizialmente abbiamo deciso di collezionare feedback generali sull'app:

- In particolare abbiamo notato un problema di visibilità con la schermata che mostra il proprio profilo utente.
Il report fornito sulla schermata è “*Non è abbastanza chiara*”, e ciò ci ha portato quindi a modificarla leggermente nel prodotto finale favorendone la chiarezza.
- Un altro feedback è il supporto alla modalità scura (dark mode) siccome ci siamo accorti che in fase di test la nostra app veniva utilizzata con i colori di sistema scuri ma continuava a mostrare sfondi molto luminosi e di colore bianco.

2. Test di usabilità

Una volta raccolti i feedback siamo passati alla successiva fase di test di usabilità a priori, che consiste nell'assegnare delle task da completare tramite il nostro prototipo di Figma e osservare i risultati entro un tempo limite.

In particolare valuteremo se il task è andato a buon fine o no. L'esito per ogni task assegnato sarà:

- **Success**, se il task è andato a buon fine
- **Partial**, se il task è terminato entro il tempo limite ma con dei problemi riscontrati durante l'uso
- **Failure**, se il task è andato oltre il tempo limite oppure è impossibile terminarlo

Task 1: Registrazione account nuovo (5 min)

Esperto	Esito Task
Esperto 1	Success (2m 30s)
Esperto 2	Success (2m 12s)
Esperto 3	Success (3m 0s)

Task 2: Creazione Asta a Tempo Fisso (5 min)

Esperto	Esito Task
Esperto 1	Success (3m 1s)
Esperto 2	Success (2m 55s)
Esperto 3	Success (2m 25s)

Task 3: Ricerca e Presentazione Offerta per Asta all’Inglese (3 min)

Esperto	Esito Task
Esperto 1	Success (1m 33)
Esperto 2	Success (1m 12s)
Esperto 3	Success (1m 55s)

Assegnando il valore 1 per le attività svolte con successo (S) e il valore 0.5 per le attività svolte parzialmente o completate oltre il tempo previsto (P), il tasso di successo può essere calcolato come il rapporto tra la somma del valore assegnato alle attività S e il valore assegnato alle attività P, e il numero totale di attività eseguite, cioè: Tasso di successo = $\frac{(9*1)}{9} = 100\%$

2.1.7 Glossario

- **Piattaforma di gestione aste:** Una piattaforma di gestione delle aste è un'applicazione software progettata per facilitare e automatizzare il processo di organizzazione, gestione e conduzione di aste online.
- **Asta all'Inglese:** L'Asta all'inglese, nota anche come asta ascendente o asta all'asta, è una modalità di asta in cui il prezzo di un oggetto in vendita parte da un valore minimo e aumenta periodicamente in base al timer impostato dal venditore.
- **Asta al Ribasso:** L'asta al ribasso, anche nota asta al minimo offerente, è una modalità di asta in cui il prezzo di un oggetto inizia da un valore elevato e diminuisce gradualmente di prezzo fino a quando viene accettata un'offerta da parte di un acquirente.
- **Asta Inversa:** L'asta inversa, è una modalità di asta in cui è il compratore a iniziare l'asta e imposta un prezzo iniziale che è disposto a pagare e una data di scadenza.
- **Back-end:** Il back-end è la parte di un'applicazione software che si occupa della logica di elaborazione e dell'accesso ai dati. È responsabile per la gestione dei dati, il calcolo delle risposte e la logica di business dell'applicazione.
- **Front-end:** Il front-end è la parte di un'applicazione software con cui gli utenti interagiscono direttamente. È responsabile della presentazione dell'interfaccia utente (UI) e dell'interazione con gli utenti finali attraverso elementi come pagine web, schermate grafiche e interfacce utente mobili.
- **Framework:** Un framework di programmazione è un'infrastruttura software che fornisce un insieme di librerie, strumenti, modelli e linee guida predefinite per facilitare lo sviluppo di applicazioni software.

- **API:** Un'API (Application Programming Interface), è un insieme di definizioni e protocolli che consentono a software e applicazioni di comunicare tra di loro. Le API Web in particolare consentono alle applicazioni di accedere e manipolare le risorse su Internet utilizzando protocolli standard come HTTP e HTTPS.
- **RESTful API:** REST (Representational State Transfer) è un'architettura software che definisce un insieme di principi e vincoli per la creazione di servizi web leggeri, scalabili e interoperabili. Le API RESTful sono interfacce di programmazione che seguono questi principi e permettono alle applicazioni di comunicare tra di loro attraverso il web in modo semplice e standardizzato.
- **UML (Unified Modeling Language):** è un linguaggio di modellazione standard utilizzato principalmente nell'ingegneria del software per descrivere, progettare e documentare sistemi software complessi. Fornisce una serie di notazioni grafiche e regole semantiche per rappresentare aspetti di un sistema software, tra cui la struttura, il comportamento e le interazioni tra i componenti del sistema.
- **Use Case Diagram:** è una delle notazioni grafiche utilizzate nell'UML per modellare e rappresentare le funzionalità e le interazioni di un sistema software dal punto di vista degli attori esterni.
- **Sequence Diagram:** è uno dei diagrammi di interazione definiti nell'UML e viene utilizzato per visualizzare in modo grafico l'interazione tra gli oggetti o le componenti all'interno di un sistema software in un determinato scenario di esecuzione.
- **Statechart:** è uno dei diagrammi definiti nell'UML utilizzato per descrivere il comportamento di un sistema tramite la rappresentazione dei vari stati in cui può trovarsi un oggetto o una classe durante il suo ciclo di vita e le transizioni tra questi stati.
- **Personas:** sono rappresentazioni sintetiche e dettagliate di utenti tipici di un sistema o prodotto. Questo strumento aiuta i progettisti e gli sviluppatori a creare

soluzioni che si adattano meglio alle esigenze degli utenti, migliorando così l'esperienza complessiva dell'utente e aumentando le probabilità di successo del prodotto o servizio.

- **Usabilità:** si riferisce alla facilità con cui gli utenti possono utilizzare un sistema o un prodotto per raggiungere i propri obiettivi in modo efficace ed efficiente, con soddisfazione.

2.2 Specifica dei Requisiti

2.2.1 Diagrammi di Classi di Analisi

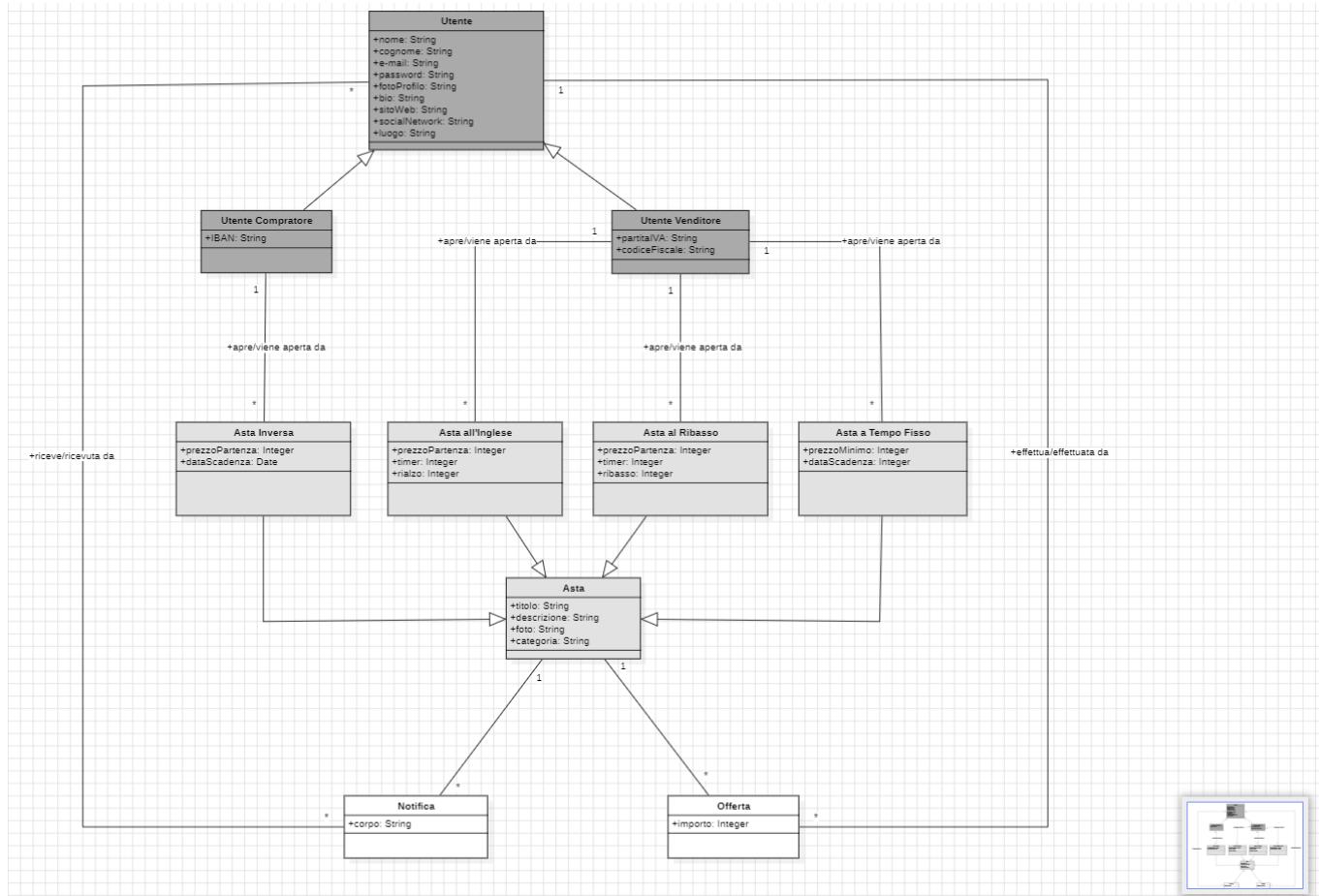
I class diagram sono strumenti essenziali per la modellazione dei concetti e delle relazioni all'interno del sistema, e forniscono una visione chiara delle entità e delle loro interazioni.

Per lo sviluppo del progetto sono stati ideati dei class diagram UML, creati con StarUML. Abbiamo scelto di implementare un'approccio “*three-object-type*”, il quale raggruppa gli oggetti di analisi in 3 tipi:

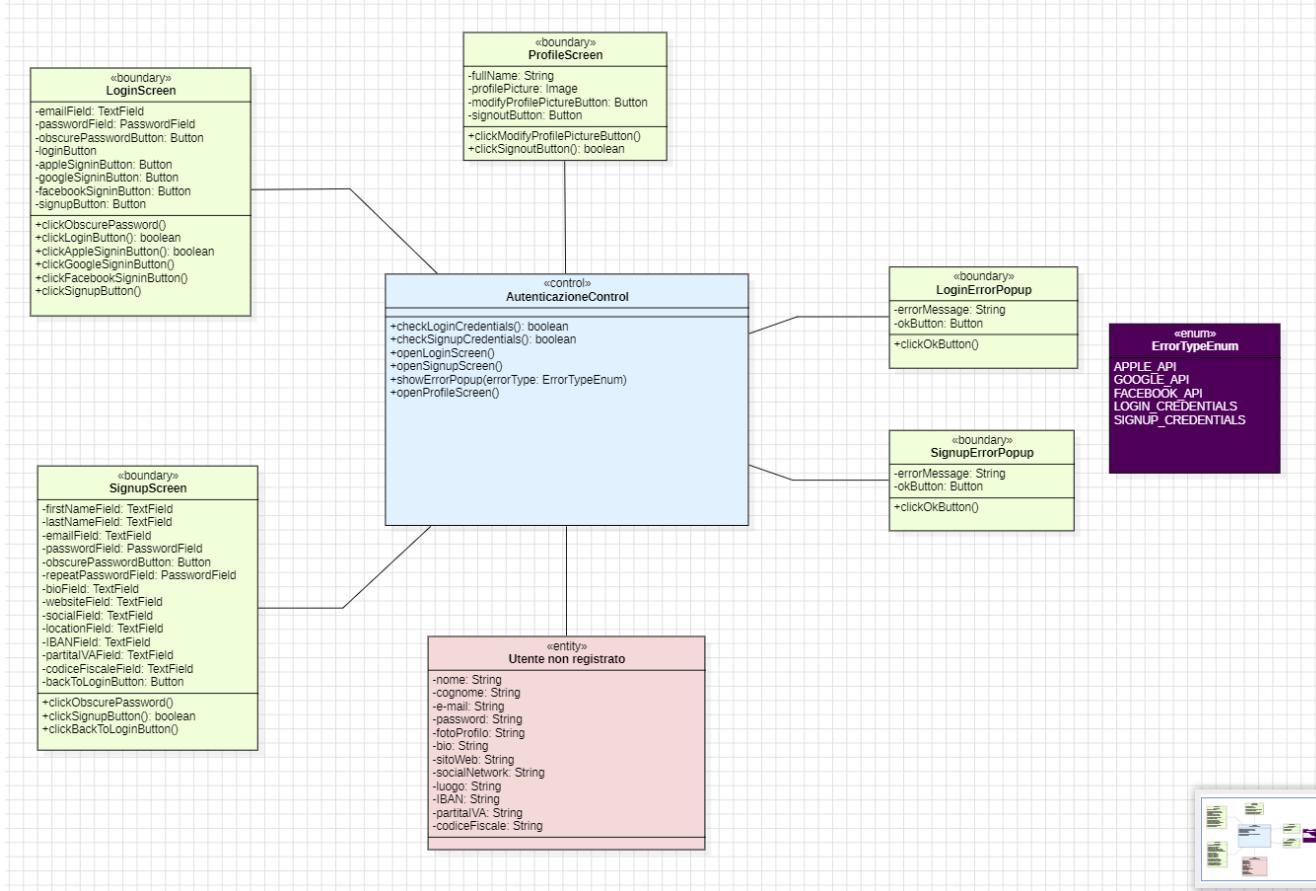
- **Entity:** Rappresentano l'informazione persistente, che deve essere memorizzata e gestita nel sistema a lungo termine
- **Boundary:** Rappresentano le interazioni tra attori e il sistema, e quindi le interfacce con il mondo esterno
- **Control:** Rappresentano la logica di controllo e si occupano di realizzare gli use case

Di seguito sono mostrati i class diagram sviluppati durante la fase di analisi dei requisiti del progetto.

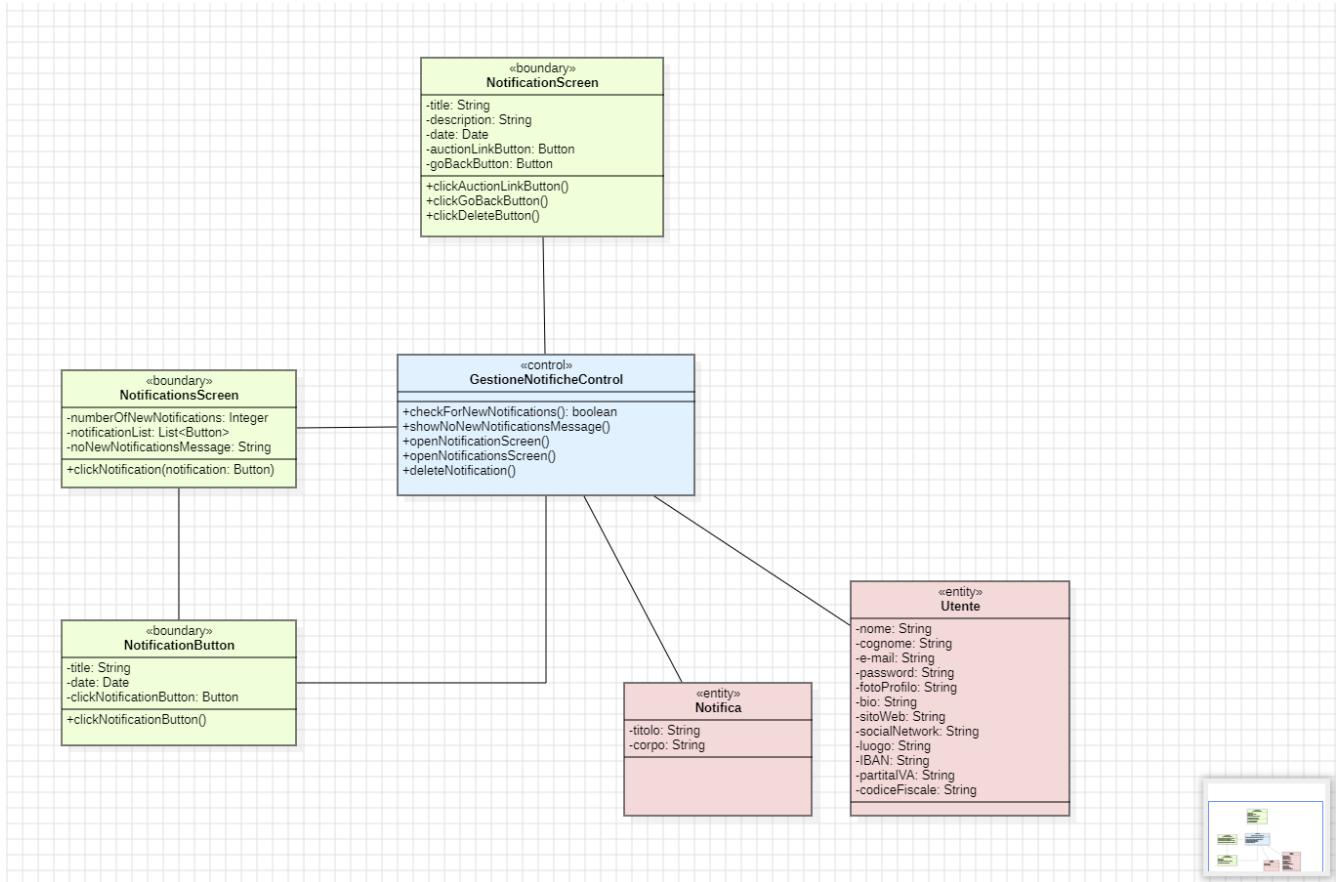
Class Diagram per le Entità



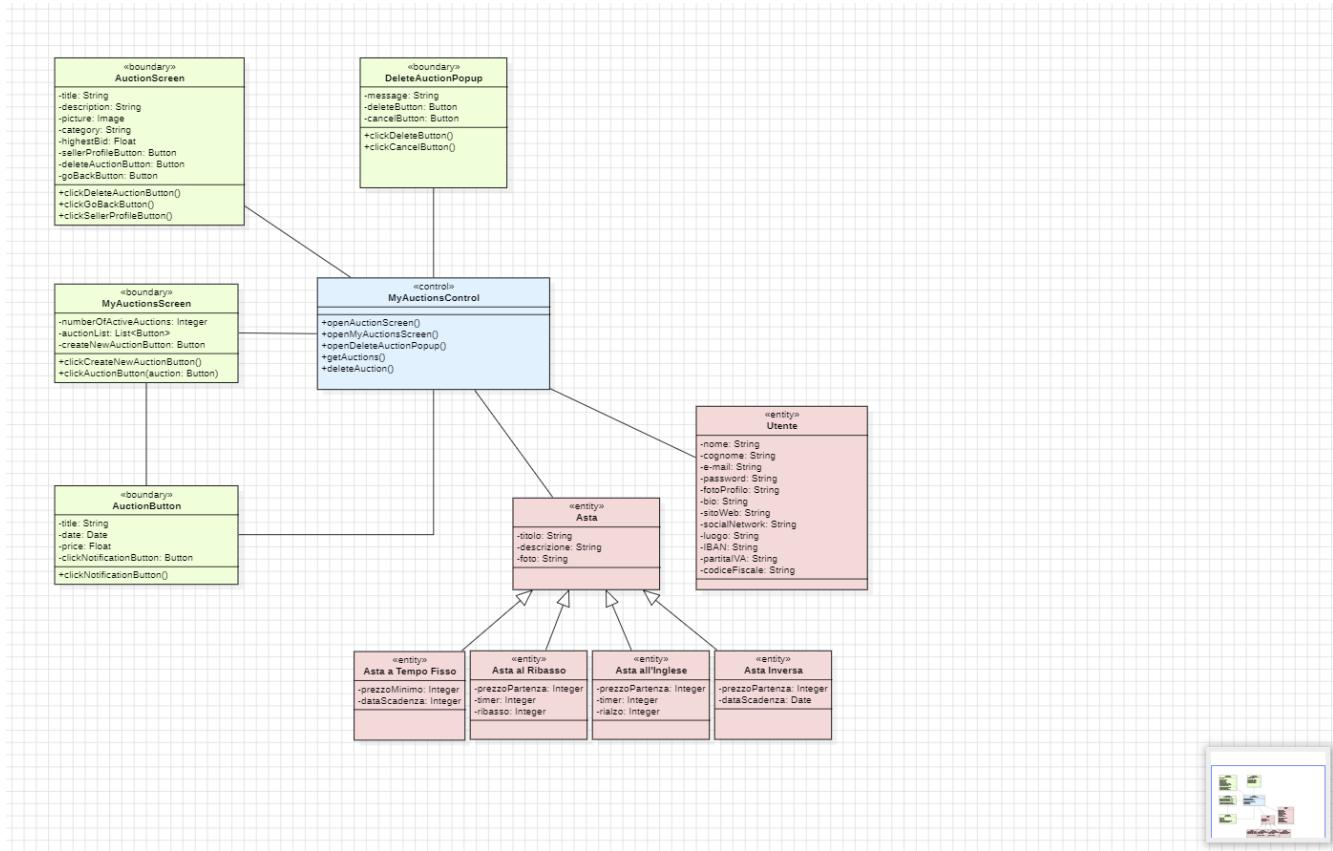
Class Diagram per l'Autenticazione di Utenti non registrati



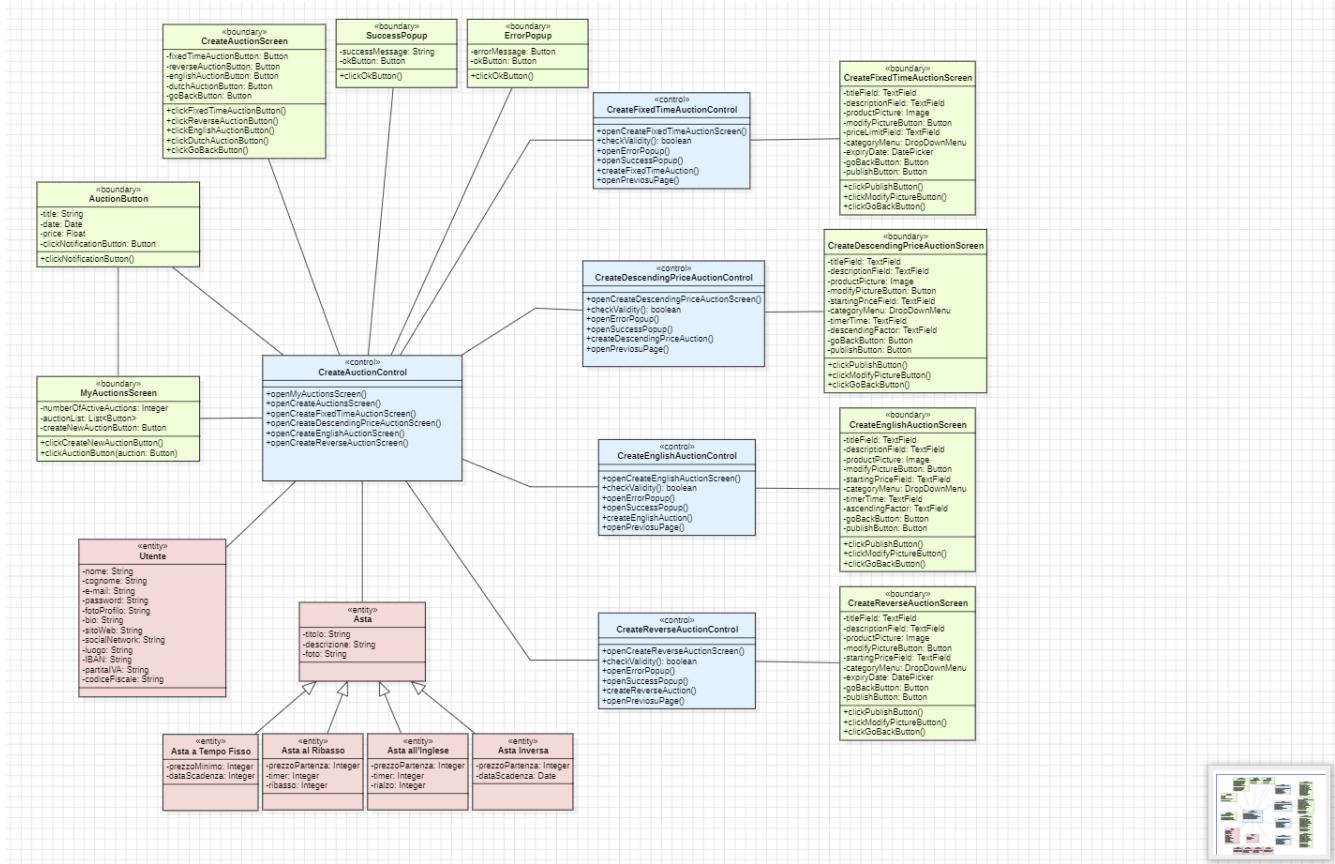
Class Diagram per la Gestione delle Notifiche



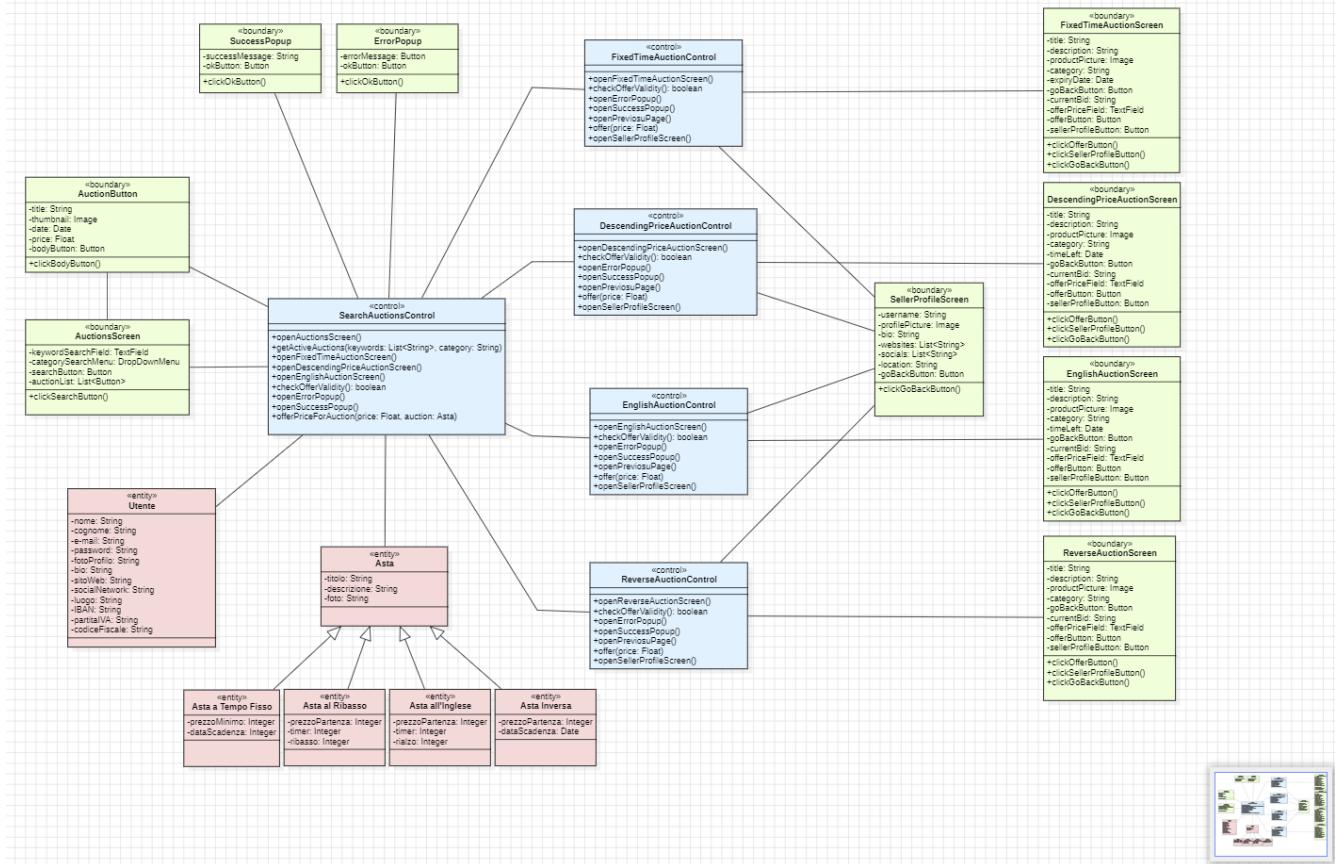
Class Diagram per la Visualizzazione di Aste Proprie



Class Diagram per la Creazione di Aste



Class Diagram per la Ricerca di Aste e Presentazione di Offerte



2.2.2 Diagrammi di Sequenza di Analisi

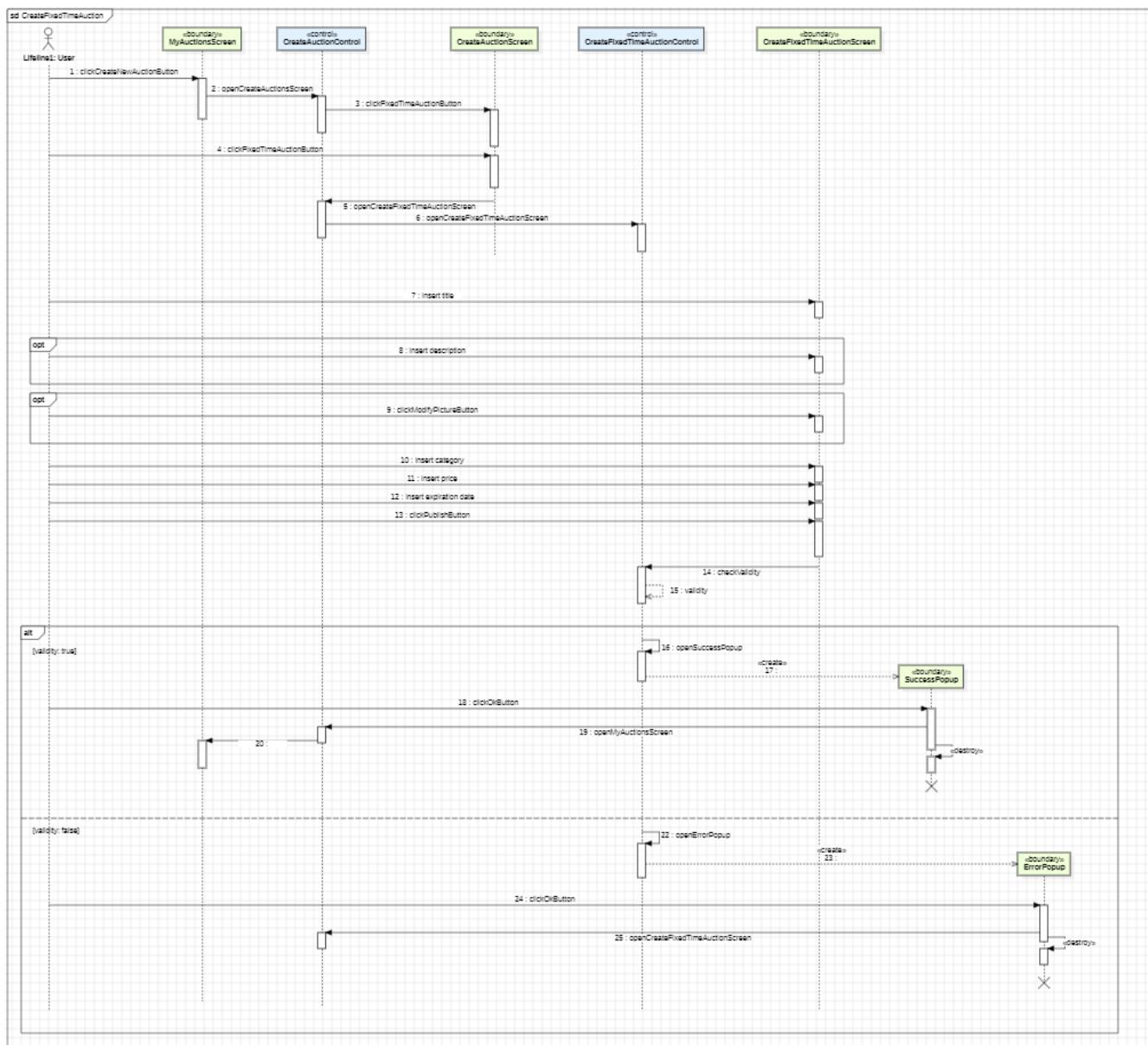
Un diagramma di sequenza UML ha l'utilità di fornire una rappresentazione dinamica di un sistema e le mostrarne le interazioni tra i componenti. Questo aiuta a seguire una timeline di eventi causati dalle interazioni tra le parti del sistema e fornire una rappresentazione dettagliata per un caso d'uso.

In questa sezione sono rappresentati i sequence diagram progettati in fase di analisi per due casi d'uso in particolare:

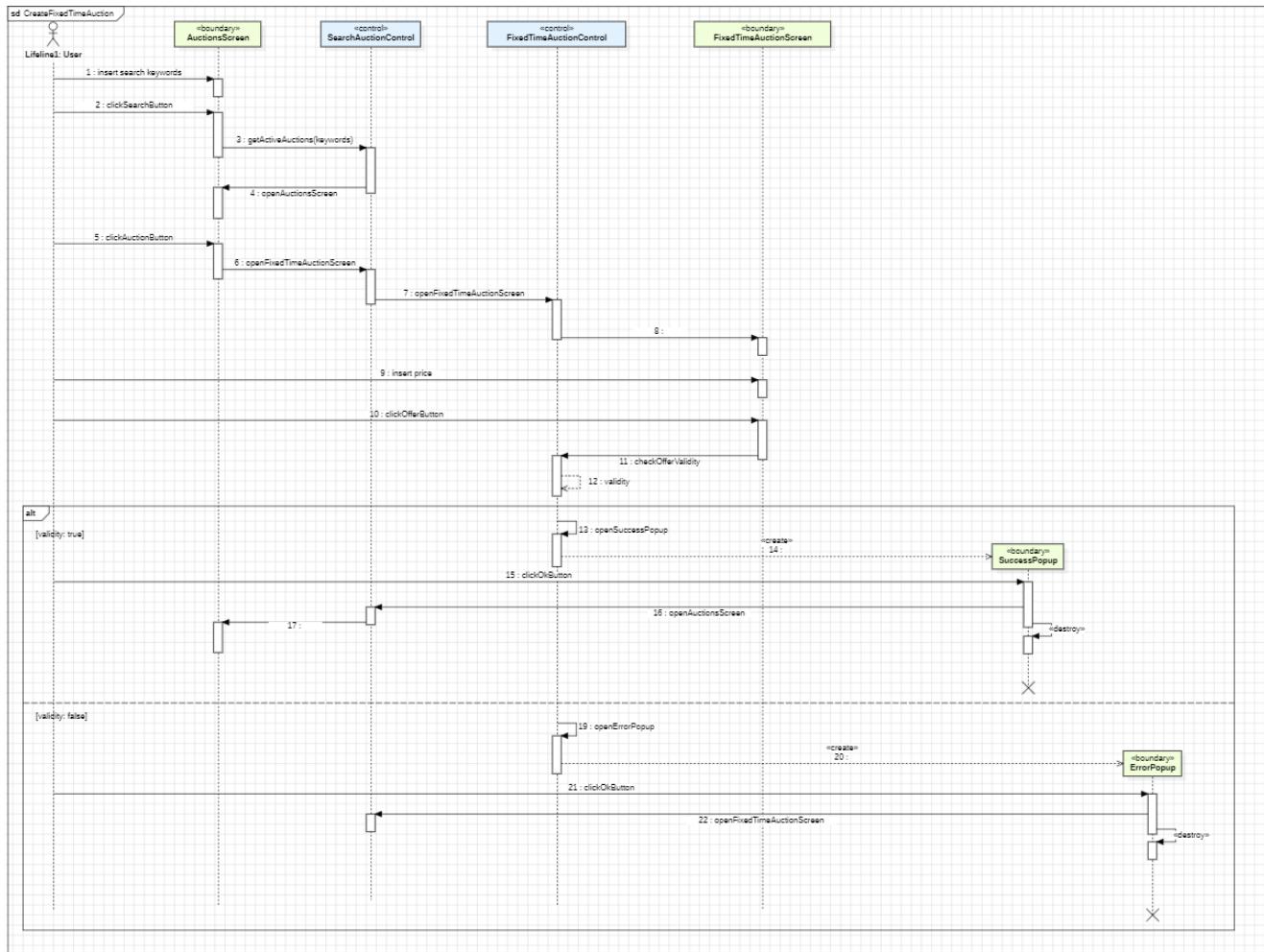
I casi d'uso scelti sono:

1. **Use Case #1:** Creazione di un'asta a tempo fisso
2. **Use Case #2:** Presentazione di un'offerta per asta a tempo fisso

Sequence Diagram per Caso d'Uso N. 1



Sequence Diagram per Caso d'Uso N. 2



2.2.3 Statechart dell'Interfaccia Grafica

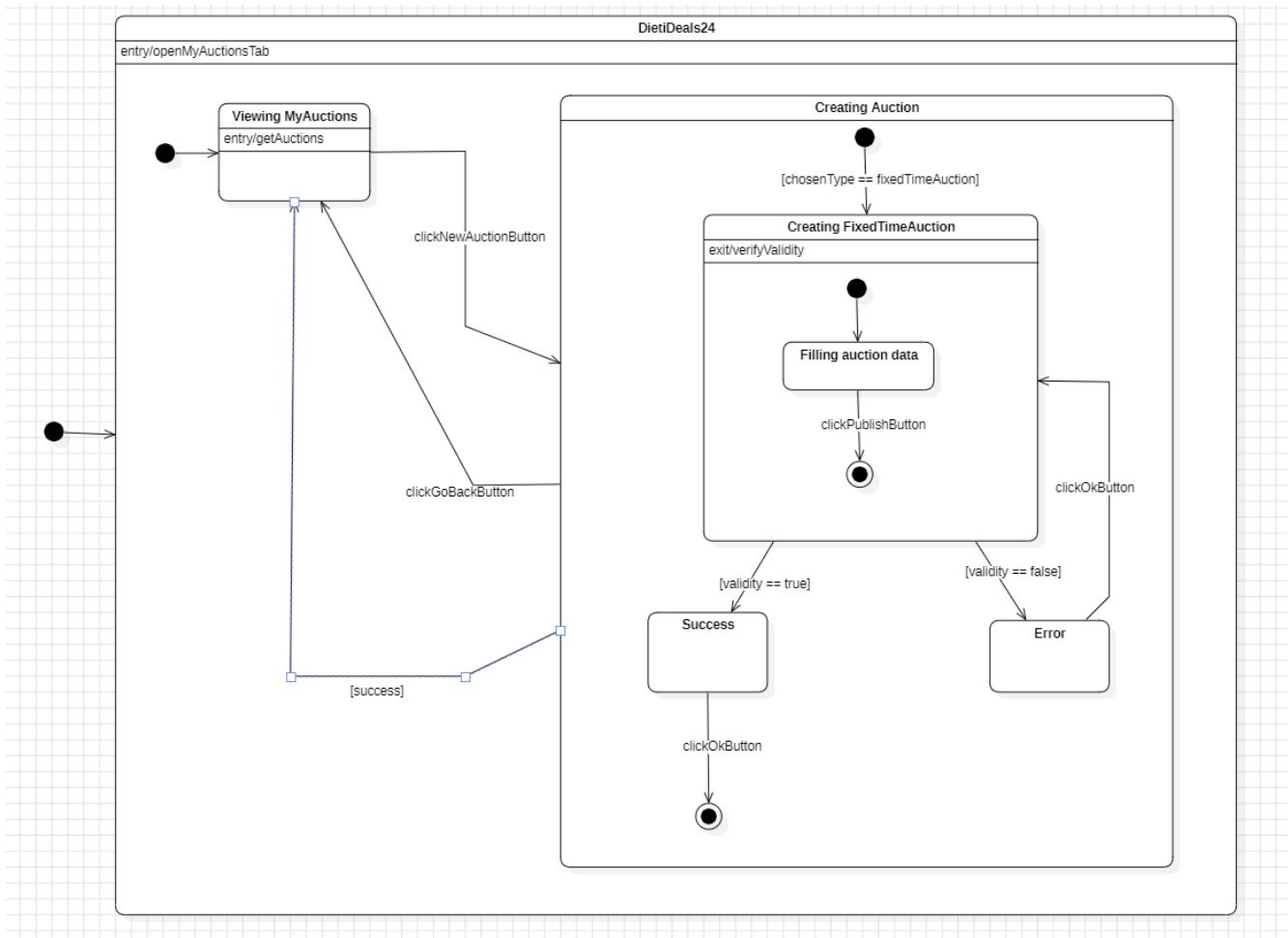
Uno Statechart, o diagramma di stato, è uno dei diagrammi definiti nell'UML utilizzato per modellare il comportamento dinamico di un sistema. Un diagramma di stato descrive il comportamento di un oggetto o di una classe attraverso la rappresentazione dei vari stati in cui può trovarsi e le transizioni tra questi stati in risposta a determinati eventi o condizioni.

In questa sezione sono rappresentati gli statechart progettati in fase di analisi per due casi d'uso in particolare:

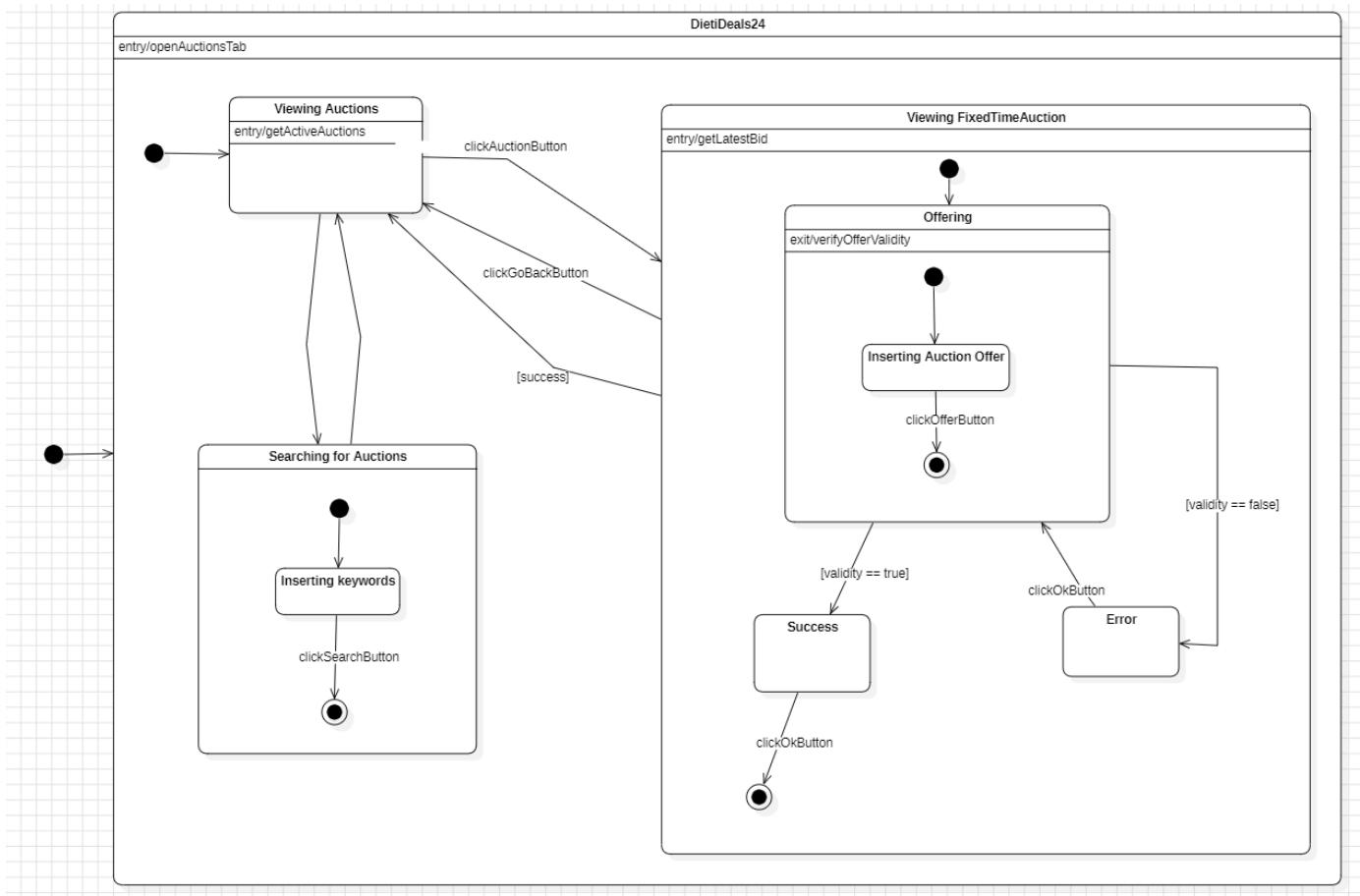
I casi d'uso scelti sono:

1. **Use Case #1:** Creazione di un'asta a tempo fisso
2. **Use Case #2:** Presentazione di un'offerta per asta a tempo fisso

Statechart per Caso d'Uso N. 1



Statechart per Caso d'Uso N. 2



Capitolo 3

3. Documento del Design di Sistema

3.1 Descrizione dell'architettura proposta

Per l'applicativo iOS utilizziamo **Amazon Web Service S3** che è un servizio che ci permette di caricare/scaricare immagini dal cloud, questo permette agli utenti nell'app di aggiungere immagini alle loro aste e/o al loro profilo.

Per quanto riguarda il back-end utilizziamo il servizio **EC2** fornito da Amazon che ci permette di hostare il back-end rendendolo pubblicamente raggiungibile tramite un URL pubblico.

Per lo scambio di informazioni tra front-end e back-end facciamo uso di richieste HTTP e di oggetti JSON, codificati/decodificati all'interno delle richieste.

Inoltre, per il back-end, implementiamo un layer di sicurezza basato su **Spring Security** in combinazione con **JWT**, che ci permettono di eliminare la necessità di sessioni sul server e di rifiutare richieste HTTP provenienti da host non autenticati.

3.2 Sviluppo Front-end

L'applicazione front-end è stata sviluppata utilizzando il linguaggio di programmazione **Swift**, garantendo una fluida esperienza utente su dispositivi Apple iOS. L'interfaccia

intuitiva permette agli utenti di navigare facilmente tra le aste in corso, visualizzare dettagli su prodotti in vendita e inoltre di effettuare offerte in tempo reale.

Swift è un linguaggio di programmazione sviluppato da Apple ed è ideato per la creazione di applicativi destinati a dispositivi mobili iOS. L'adozione di Swift come linguaggio di programmazione per lo sviluppo dell'interfaccia utente offre una serie di vantaggi significativi, sia per lo sviluppo dell'applicazione che per l'usabilità del prodotto finale.

L'utilizzo di Swift come linguaggio di programmazione per lo sviluppo del front-end offre una serie di vantaggi, tra i quali:

- Offre un'eccellente varietà di servizi ed un **alto livello di mantenimento e portabilità**. Esso offre una zona di lavoro sicura per gli sviluppatori, ed a sua volta garantisce un'esperienza ottimale per gli utenti.
- Swift ha inoltre una **sintassi chiara e intuitiva**, che lo rende più accessibile anche per sviluppatori principianti e alle prime armi nel campo dello sviluppo di applicazioni front-end. Ciò ha reso possibile lo sviluppo dell'applicazione in tempi relativamente ristretti.

In sintesi, l'uso di Swift come linguaggio per lo sviluppo dell'interfaccia utente offre facilità di utilizzo e integrazione con l'ecosistema iOS, che lo rendono una scelta ideale per la creazione di app iOS moderne.

3.3 Sviluppo Back-end

Parallelamente, il sistema back-end è stato implementato utilizzando Java e il framework Spring Boot, fornendo una base per la gestione delle logiche di business e la comunicazione con il database. Questo sistema è distribuito come container Docker, garantendo scalabilità, flessibilità e facilità di gestione.

L'utilizzo di Spring Boot come framework per lo sviluppo del back-end offre una serie di vantaggi, tra i quali:

- I meccanismi di configurazione sono per la maggior parte automatizzati, e ciò permette agli sviluppatori di concentrarsi sulla scrittura del codice, dei test e sulla gestione degli errori, senza doversi preoccupare di configurazioni complesse.
- Spring Boot fa inoltre parte dell'ecosistema Spring, che offre un'ampia gamma di librerie e moduli per affrontare una varietà di requisiti nel mondo dello sviluppo di applicazioni enterprise.
- È offerto un supporto completo per la creazione di API RESTful, che sono essenziali per la comunicazione tra il front-end e il back-end. Tramite Spring Boot è relativamente semplice definire i controller REST per gestire le richieste HTTP in ingresso, implementando operazioni CRUD (Create, Read, Update, Delete) su risorse come aste, utenti e offerte.
- La gestione delle dipendenze Maven all'interno di un'applicazione è semplificata dallo strumento Spring Initializr, che permette agli sviluppatori di avviare nuovi progetti con una configurazione predefinita e selezionare le dipendenze desiderate con pochi clic, semplificando così il processo di configurazione del progetto.

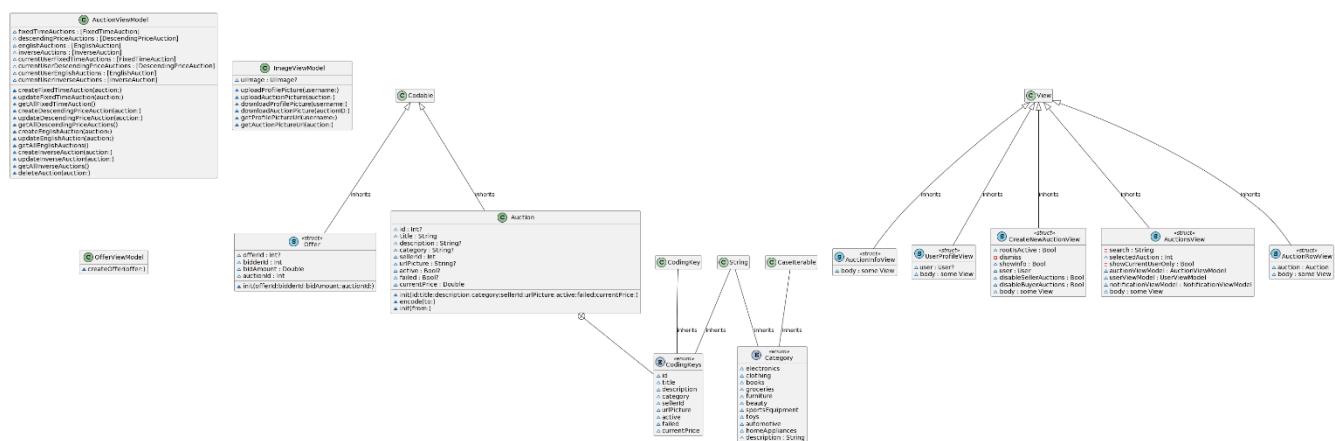
3.4 Diagrammi di Classi di Design

Di seguito sono mostrati i diagrammi di classi UML progettati durante la fase di implementazione dei requisiti software nel prodotto software vero e proprio. Ogni class diagram UML si riferisce ad uno o ad una serie di requisiti software.

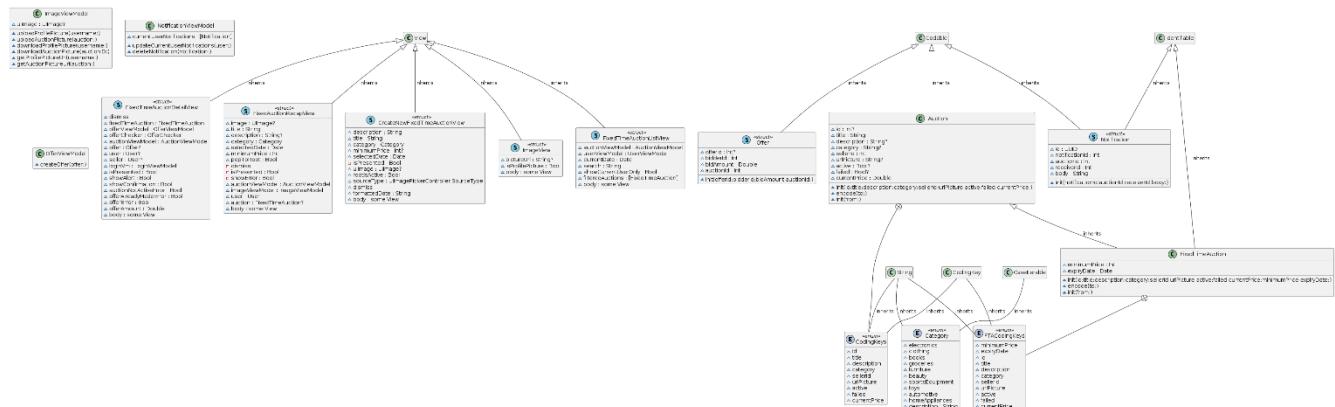
Class Diagram per il Requisito Funzionale N. 1



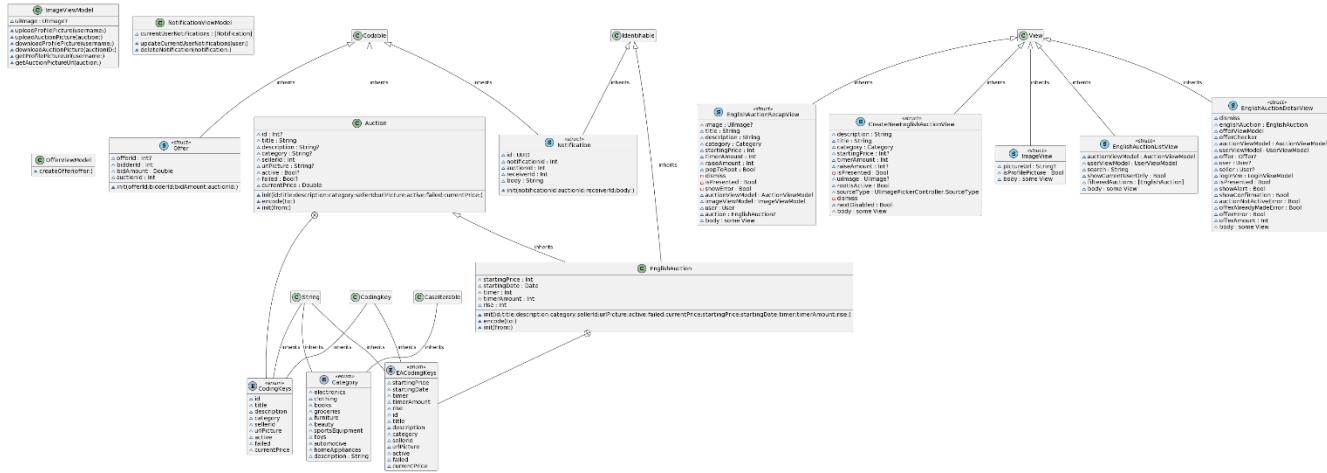
Class Diagram per i Requisiti Funzionali N. 2-3



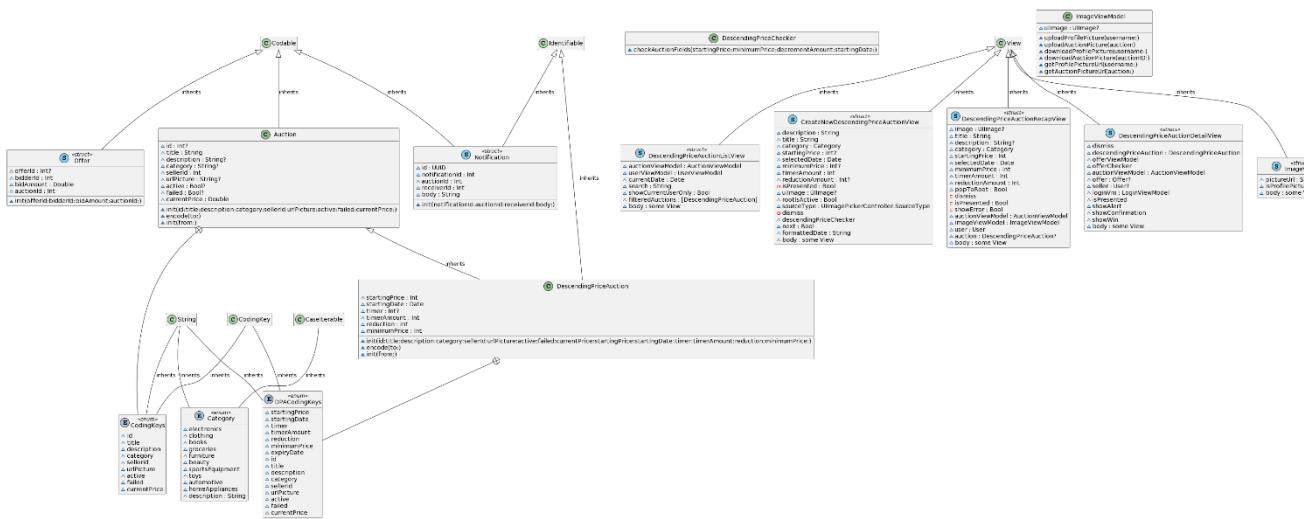
Class Diagram per il Requisito Funzionale N. 4



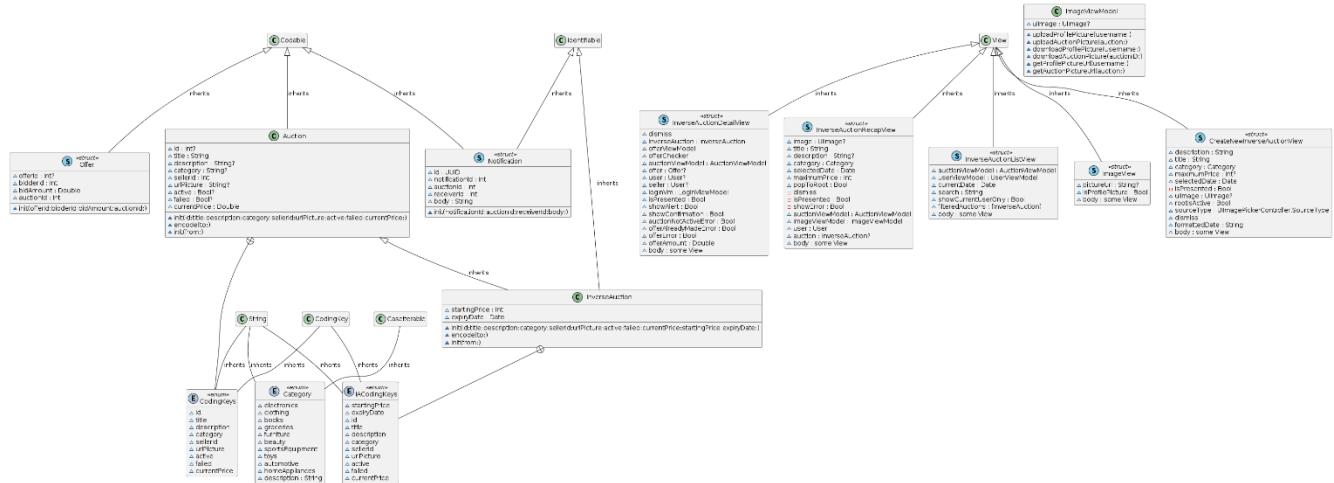
Class Diagram per il Requisito Funzionale N. 5



Class Diagram per il Requisito Funzionale N. 6



Class Diagram per il Requisito Funzionale N. 8



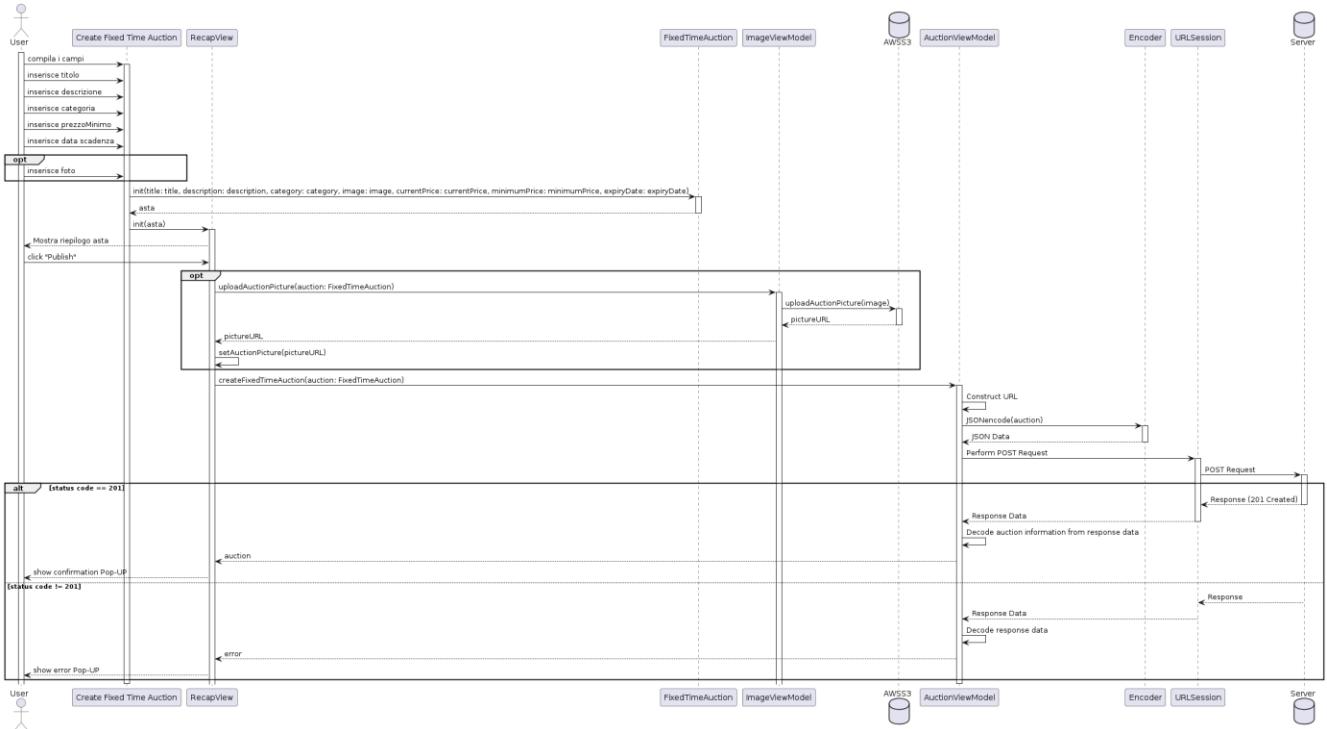
3.5 Diagrammi di Sequenza di Design

Di seguito sono mostrati i sequence diagram UML progettati durante la fase di implementazione dei requisiti software nel prodotto software vero e proprio. I due sequence diagram rappresentano l'interazione utente-sistema in due casi d'uso in particolare.

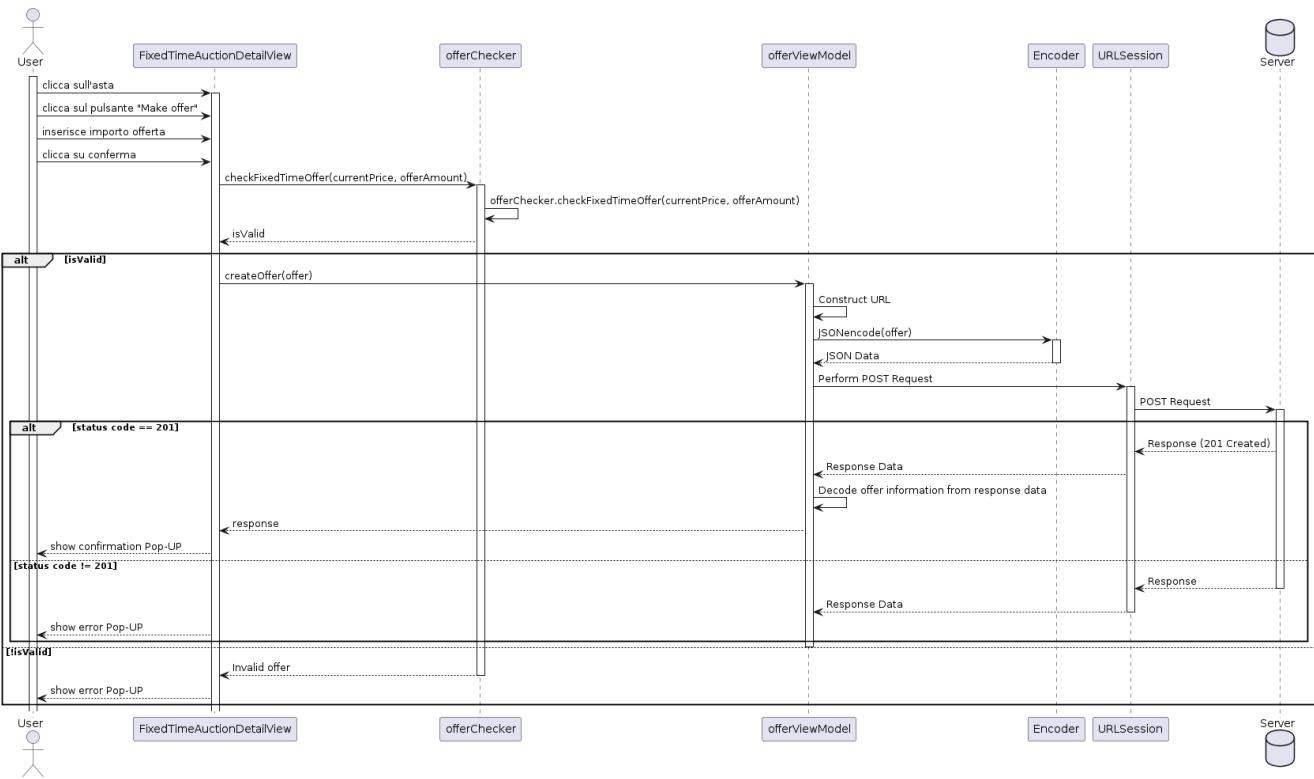
I casi d'uso scelti sono:

1. **Use Case #1:** Creazione di un'asta a tempo fisso
2. **Use Case #2:** Presentazione di un'offerta per asta a tempo fisso

Sequence Diagram per Caso d'Uso N. 1



Sequence Diagram per Caso d'Uso N. 2



Capitolo 4

4. Versioning e Report della Qualità

4.1 Uso di Strumenti di Versioning

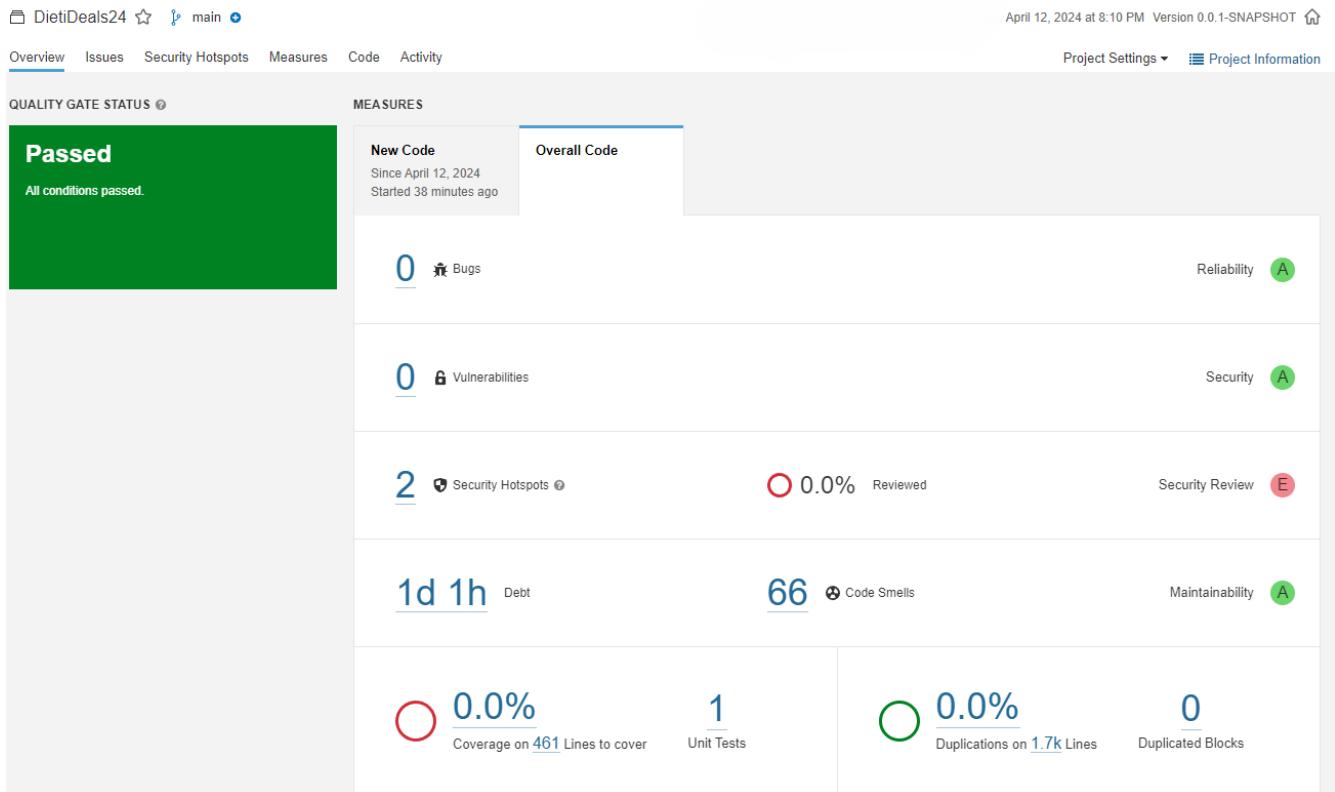
Lo sviluppo della piattaforma DietiDeals24 è stato possibile tramite l'uso di strumenti di versioning come GitHub. E' possibile visualizzare la repository del progetto al seguente link:

<https://github.com/T0nyAbb/DietiDeals24>

4.2 Report della Qualità del Codice

SonarQube è uno strumento di analisi statica del codice che identifica e segnala una vasta gamma di problemi di qualità del codice, tra cui bug, vulnerabilità, codice duplicato, complessità del codice e altro ancora.

Di seguito è mostrato in formato immagine il report per la qualità del codice generato da SonarQube.



Capitolo 5

5. Testing e Valutazione dell'Usabilità

5.1 Unit Testing

Per assicurare che la nostra app rispetti degli standard di qualità, è necessario fare del testing, onde evitare il presentarsi di comportamenti inattesi o errori.

È essenziale quindi eseguire **unit test** per garantire la stabilità e la robustezza del codice. I test unitari ci consentono di individuare e risolvere errori o comportamenti inaspettati in modo efficiente, migliorando la manutenibilità del nostro software. Inoltre, ci aiutano con l'implementazione e/o integrazione di nuove funzionalità.

Di conseguenza la scelta del framework per effettuare unit testing è ovviamente ricaduta su **XCTest** ovvero il framework di testing integrato fornito da Apple per lo sviluppo di applicazioni iOS, macOS, watchOS e tvOS. Esso offre una serie di classi e metodi per la scrittura e l'esecuzione di test unitari, di integrazione e di prestazioni all'interno dell'ambiente di sviluppo Xcode.

Si è scelto di testare 2 funzionalità appartenenti al lato front-end.

I metodi scelti da testare sono:

1. Il metodo **checkFields**, utilizzato per controllare che un utente inserisca dati veritieri e coerenti in fase di registrazione.

2. Il metodo **checkAuctionField**, che controlla che un venditore inserisca una asta al ribasso correttamente, verificando quindi la validità dei campi.

Per la progettazione della batteria di test si è scelto di utilizzare la metodologia **Black-Box** con criterio di copertura WECT. Si è preferito utilizzare WECT a discapito di altre metodologie come SECT o WCT per non avere un numero troppo elevato di casi di testing da scrivere.

```
1 //  
2 // DietiDeals24Tests.swift  
3 // DietiDeals24Tests  
4 //  
5 // Created by Antonio Abbatiello on 12/01/24.  
6 //  
7  
8 import XCTest  
9 @testable import DietiDeals24  
10  
11 final class DietiDeals24Tests: XCTestCase {  
12  
13     var fieldChecker: FieldChecker!  
14     var descendingPriceChecker: DescendingPriceChecker!  
15  
16     override func setUpWithError() throws {  
17         // Put setup code here. This method is called before the invocation of each test method in the class.  
18         super.setUp()  
19         fieldChecker = FieldChecker()  
20         descendingPriceChecker = DescendingPriceChecker()  
21     }  
22  
23     override func tearDownWithError() throws {  
24         // Put teardown code here. This method is called after the invocation of each test method in the class.  
25         fieldChecker = nil  
26         descendingPriceChecker = nil  
27         super.tearDown()  
28     }  
29 }
```

Test Setup

Unit Test N. 1 - checkFields

Il metodo prende come input 4 parametri:

- *username*
- *password*
- *confirmPassword*
- *Iban*

tutti di tipo Stringa.

Le classi di equivalenza individuate sono le seguenti:

EC1 -> Parametro username valido;

EC2 -> Parametro username vuoto;

EC3 -> Parametro username null;

EC4 -> Parametro username che non ha il carattere "@"

EC5 -> Parametro username che non ha il dominio (.com, .it)

EC6 -> Parametro password valida;

EC7 -> Parametro password vuoto;

EC8 -> Parametro password null;

EC9 -> Parametro password più corto di 5 caratteri;

EC10 -> Parametro password che non contiene almeno un numero;

EC11 -> Parametro confirmPassword che è uguale al parametro password;

EC12 -> Parametro confirmPassword che non è uguale al parametro password;

EC13 -> Parametro confirmPassword vuoto;

EC14 -> Parametro confirmPassword null;

EC15 -> Parametro iban valido;

EC16 -> Parametro iban vuoto;

EC17 -> Parametro iban null;

EC18 -> Parametro iban più corto di 15 caratteri;

EC19 -> Parametro iban più lungo di 30 caratteri;

EC20 -> Parametro iban che non inizia con due lettere;

I casi di test individuati sono i seguenti:

TC1 -> testValidFields() che copre le classi: EC1, EC6, EC11, EC15

TC2 -> testEmptyUsername() che copre le classi: EC2, EC6, EC11, EC15

TC3 -> testNilUsername() che copre le classi: EC3, EC6, EC11, EC15

TC4 -> testNoAtUsername() che copre le classi: EC4, EC6, EC11, EC15

TC5 -> testNoDomainUsername() che copre le classi: EC5, EC6, EC11, EC15

TC6 -> testEmptyPassword() che copre le classi: EC1, EC7, EC11, EC15

TC7 -> testNilPassword() che copre le classi: EC1, EC8, EC11, EC15

TC8 -> testPasswordTooShort() che copre le classi: EC1, EC9, EC11, EC15

TC9 -> testPasswordWithoutNumber() che copre le classi: EC1, EC10, EC11, EC15

TC10 -> testPasswordsNotMatching() che copre le classi: EC1, EC6, EC12, EC15

TC11 -> testEmptyConfirmPassword() che copre le classi: EC1, EC6, EC13, EC15

TC12 -> testNilConfirmPassword() che copre le classi: EC1, EC6, EC14, EC15

TC13 -> testEmptyIban() che copre le classi: EC1, EC6, EC11, EC16

TC14 -> testNilIban() che copre le classi: EC1, EC6, EC11, EC17

TC15 -> testIbanTooShort() che copre le classi: EC1, EC6, EC11, EC18

TC16 -> testIbanTooLong() che copre le classi: EC1, EC6, EC11, EC19

TC17 -> testInvalidIbanFormat() che copre le classi: EC1, EC6, EC11, EC20

```

30    // MARK: - Registration tests
31
32
33    //Test valid fields
34    func testValidFields() {
35        // Arrange
36        let username = "test@example.com"
37        let password = "Password123"
38        let confirmPassword = "Password123"
39        let iban = "AB1234567890123"
40
41        // Act & Assert
42        XCTAssertTrue(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
43    }
44
45    //Test empty username
46    func testEmptyUsername() {
47        // Arrange
48        let username = ""
49        let password = "Password123"
50        let confirmPassword = "Password123"
51        let iban = "AB1234567890123"
52
53        // Act & Assert
54        XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
55    }
56
57    //Test nil username
58    func testNilUsername() {
59        // Arrange
60        let username: String? = nil
61        let password = "Password123"
62        let confirmPassword = "Password123"
63        let iban = "AB1234567890123"
64
65        // Act & Assert
66        XCTAssertThrowsError(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
67    }
68
69    //Test no @ username
70    func testNoAtUsername() {
71        // Arrange
72        let username = "invalidemail.com"
73        let password = "Password123"
74        let confirmPassword = "Password123"
75        let iban = "AB1234567890123"
76
77        // Act & Assert
78        XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
79    }
80
81    //Test no domain username
82    func testNoDomainUsername() {
83        // Arrange
84        let username = "invalidemail@nodomain"
85        let password = "Password123"
86        let confirmPassword = "Password123"
87        let iban = "AB1234567890123"
88
89        // Act & Assert
90        XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
91    }
92
93    //Test empty password
94    func testEmptyPassword() {
95        // Arrange
96        let username = "test@example.com"
97        let password = ""
98        let confirmPassword = "Password123"
99        let iban = "AB1234567890123"
100
101        // Act & Assert
102        XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
103    }
104

```

```

105 //Test nil password
106 func testNilPassword() {
107     // Arrange
108     let username = "test@example.com"
109     let password: String? = nil
110     let confirmPassword = "Password123"
111     let iban = "AB1234567890123"
112
113     // Act & Assert
114     XCTAssertThrowsError(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
115 }
116
117 //Test password too short
118 func testPasswordTooShort() {
119     // Arrange
120     let username = "test@example.com"
121     let password = "1234"
122     let confirmPassword = "1234"
123     let iban = "AB1234567890123"
124
125     // Act & Assert
126     XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
127 }
128
129 //Test password not containing a number
130 func testPasswordWithoutNumber() {
131     // Arrange
132     let username = "test@example.com"
133     let password = "Password"
134     let confirmPassword = "Password"
135     let iban = "AB1234567890123"
136
137     // Act & Assert
138     XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
139 }
140
141 //Test passwords not matching
142 func testPasswordsNotMatching() {
143     // Arrange
144     let username = "test@example.com"
145     let password = "Password123"
146     let confirmPassword = "DifferentPassword123"
147     let iban = "AB1234567890123"
148
149     // Act & Assert
150     XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
151 }
152
153 //Test empty confirm password
154 func testEmptyConfirmPassword() {
155     // Arrange
156     let username = "test@example.com"
157     let password = "Password123"
158     let confirmPassword = ""
159     let iban = "AB1234567890123"
160
161     // Act & Assert
162     XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
163 }
164
165 //Test nil confirm password
166 func testNilConfirmPassword() {
167     // Arrange
168     let username = "test@example.com"
169     let password = "Password123"
170     let confirmPassword : String? = nil
171     let iban = "AB1234567890123"
172
173     // Act & Assert
174     XCTAssertThrowsError(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
175 }
176

```

```

177 //Test empty iban
178 func testEmptyIban() {
179     // Arrange
180     let username = "test@example.com"
181     let password = "Password123"
182     let confirmPassword = "Password123"
183     let iban = ""
184
185     // Act & Assert
186     XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
187 }
188
189 //Test nil iban
190 func testNilIban() {
191     // Arrange
192     let username = "test@example.com"
193     let password = "Password123"
194     let confirmPassword = "Password123"
195     let iban: String? = nil
196
197     // Act & Assert
198     XCTAssertThrowsError(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
199 }
200
201 //Test IBAN length less than 15 characters
202 func testIbanTooShort() {
203     // Arrange
204     let username = "test@example.com"
205     let password = "Password123"
206     let confirmPassword = "Password123"
207     let iban = "AB123456789012"
208
209     // Act & Assert
210     XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
211 }
212
213 //Test IBAN length more than 30 characters
214 func testIBANTooLong() {
215     // Arrange
216     let username = "test@example.com"
217     let password = "Password123"
218     let confirmPassword = "Password123"
219     let iban = "AB1234567890123456789012345678901"
220
221     // Act & Assert
222     XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
223 }
224
225 //Test IBAN format incorrect
226 func testInvalidIBANFormat() {
227     // Arrange
228     let username = "test@example.com"
229     let password = "Password123"
230     let confirmPassword = "Password123"
231     let iban = "123456789012345"
232
233     // Act & Assert
234     XCTAssertFalse(try! fieldChecker.checkFields(username: username, password: password, confirmPassword: confirmPassword, iban: iban))
235 }
236

```

Unit Test N. 2 – checkAuctionFields

Il metodo prende come input 4 parametri:

- *startingPrice*
- *minimumPrice*
- *decrementAmount* di tipo Int
- *startDate* di tipo Date

Le classi di equivalenza individuate sono le seguenti:

EC1 -> Parametro startingPrice maggiore di 0;

EC2 -> Parametro startingPrice minore o uguale a 0;

EC3 -> Parametro startingPrice null;

EC4 -> Parametro minimumPrice maggiore di 0;

EC5 -> Parametro minimumPrice minore o uguale a 0;

EC6 -> Parametro minimumPrice null;

EC7 -> Parametro decrementAmount maggiore di 0;

EC8 -> Parametro decrementAmount minore o uguale a 0;

EC9 -> Parametro decrementAmount null;

EC10 -> Parametro startingDate maggiore della data attuale;

EC11 -> Parametro startingDate minore o uguale alla data attuale;

EC12 -> Parametro startingDate null;

I casi di test individuati sono i seguenti:

TC1 -> `testValidAuctionFields()` che copre le classi: EC1, EC4, EC7, EC10

TC2 -> `testStartingPriceLessThanOrEqualToZero()` che copre le classi: EC2, EC4, EC7, EC10

TC3 -> `testNilStartingPrice()` che copre le classi: EC3, EC4, EC7, EC10

TC4 -> `testMinimumPriceLessThanOrEqualToZero()` che copre le classi: EC1, EC5, EC7, EC10

TC5 -> `testNilMinimumPrice()` che copre le classi: EC1, EC6, EC7, EC10

TC6 -> `testDecrementAmountLessThanOrEqualToZero()` che copre le classi: EC1, EC4, EC8, EC10

TC7 -> `testNilDecrementAmount()` che copre le classi: EC1, EC4, EC9, EC10

TC8 -> `testStartingDateInPast()` che copre le classi: EC1, EC4, EC7, EC11

TC9 -> `testNilStartingDate()` che copre le classi: EC1, EC4, EC7, EC12

Nota:

Il tipo Date in Swift non può avere direttamente errori di formattazione, eccezione fatta nel caso in cui si usa un DateFormatter che modifica il tipo Date, che non rientra nelle casistiche prese in esame, pertanto non è stata considerata una classe di equivalenza che tiene conto di un errore di formattazione.

Eseguendo la batteria di test si ottiene il seguente output:

```
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilConfirmPassword]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilConfirmPassword]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilDecrementAmount]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilDecrementAmount]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilIban]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilIban]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilMinimumPrice]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilMinimumPrice]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilPassword]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilPassword]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilStartingDate]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilStartingDate]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilStartingPrice]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilStartingPrice]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilUsername]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNilUsername]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNoAtUsername]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNoAtUsername]' passed (0.001 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNoDomainUsername]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testNoDomainUsername]' passed (0.001 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testPasswordsNotMatching]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testPasswordsNotMatching]' passed (0.001 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testPasswordTooShort]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testPasswordTooShort]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testPasswordWithoutNumber]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testPasswordWithoutNumber]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testStartingDateInPast]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testStartingDateInPast]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testStartingPriceLessThanOrEqualToZero]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testStartingPriceLessThanOrEqualToZero]' passed (0.001 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testValidAuctionFields]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testValidAuctionFields]' passed (0.000 seconds).
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testValidFields]' started.
Test Case '-[DietiDeals24Tests.DietiDeals24Tests testValidFields]' passed (0.000 seconds).
Test Suite 'DietiDeals24Tests' passed at 2024-04-14 02:09:52.943.
    Executed 26 tests, with 0 failures (0 unexpected) in 0.039 (0.101) seconds
```

Test Results

5.2 Valutazione dell’Usabilità

5.2.1 Fase di Valutazione

Questa sezione sarà dedicata alla seconda fase di valutazione, discussa già nel capitolo 2.1.6, basata sulla valutazione di usabilità a prodotto finito, o **valutazione dell’usabilità sul campo**.

Ad effettuare i test sono stati 4 utenti inesperti, a cui abbiamo fornito l’ambiente di gestione finale e delle “task” da effettuare. Al termine di ogni task, ogni esperienza di utente sarà valutata e saranno raccolti i feedback verbali di ogni utente.

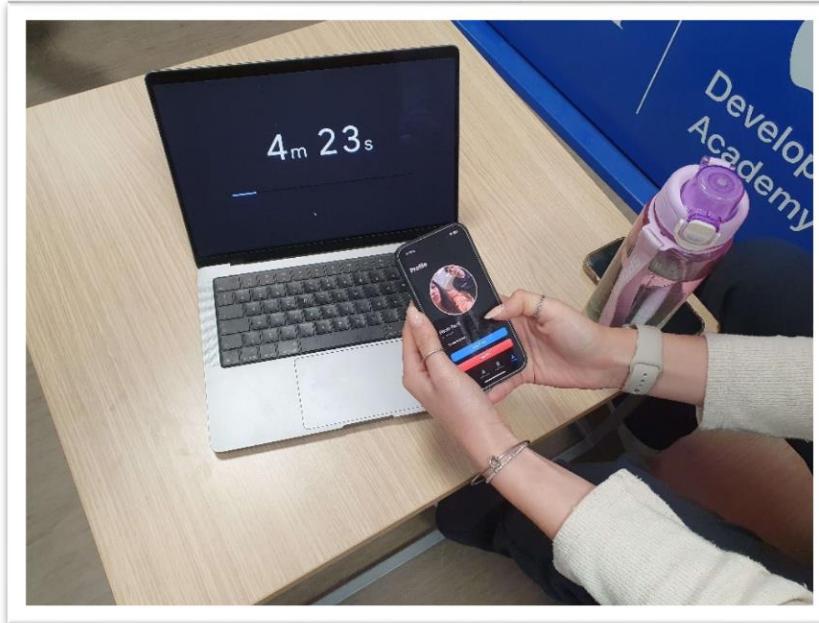
Il tempo limite per ogni task assegnata è di **X minuti** per utente.

L’esito per ogni task assegnato sarà:

- **Success**, se il task è andato a buon fine
- **Partial**, se il task è terminato entro il tempo limite ma con dei problemi riscontrati durante l’uso
- **Failure**, se il task è andato oltre il tempo limite oppure è impossibile terminarlo

TASK 1: Creare un'Asta al Ribasso

La prima task sarà creare una descending price auction tempo [5 minuti]

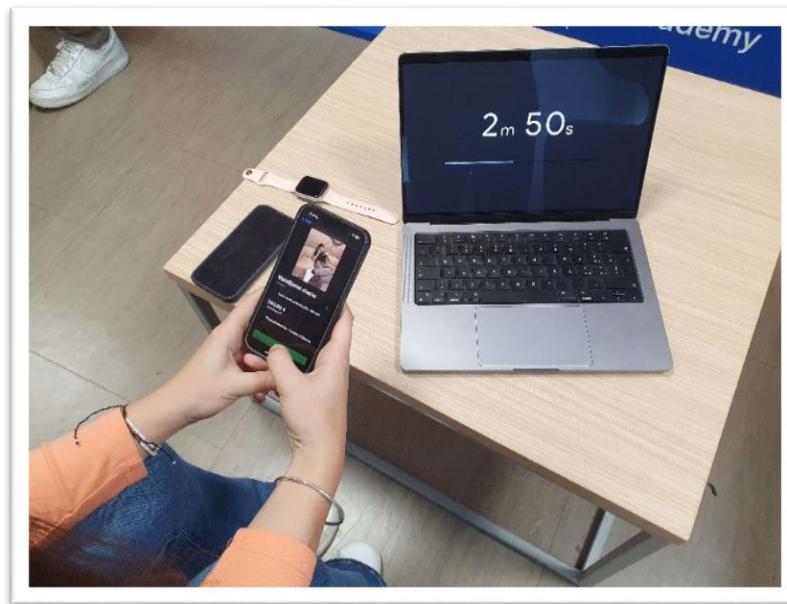


Utente	Esito Task
Utente 1	Success (4m 35s)
Utente 2	Success (4m 10s)
Utente 3	Partial* (4m 12s)
Utente 4	Success (4m 23s)

* L'utente ha completato la task in tempo, ma non è riuscito a distinguere tra la schermata "Le mie aste" e "Aste". È stato poi chiarito all'utente che la schermata "le mie aste" mostra tutte le aste create dal account autenticato al momento, mentre la schermata "Aste" mostra le aste correntemente attive create da altri utenti.

TASK 2: Presentare un'Offerta per un'Asta a Tempo Fisso

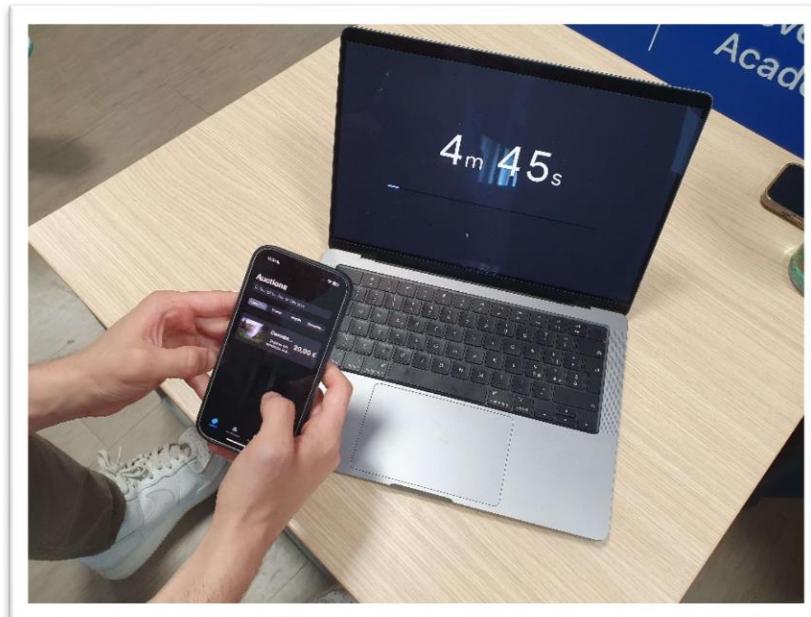
La seconda task sarà fare un'offerta per una fixed time auction [5 minuti]



Utente	Esito Task
Utente 1	Success (3m 15s)
Utente 2	Success (2m 50s)
Utente 3	Success (3m 2s)
Utente 4	Success (2m 56s)

TASK 3: Cercare un'Asta al Ribasso e Aggiudicarsi l'Asta

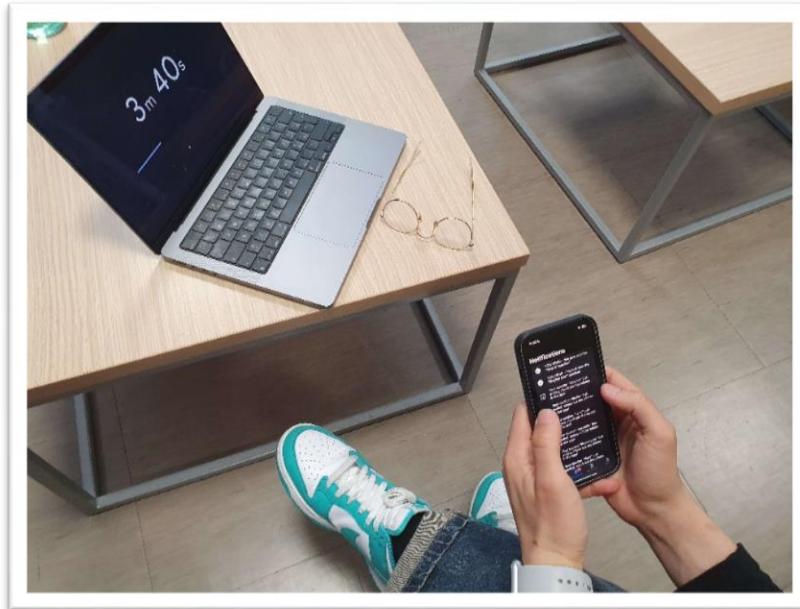
La terza task sarà cercare una descending price auction e aggiudicarsi l'asta [5 minuti]



Utente	Esito Task
Utente 1	Success (4m 45s)
Utente 2	Success (4m 14s)
Utente 3	Success (3m 53s)
Utente 4	Success (4m 29s)

TASK 4: Verificare la Ricevuta della Notifica e Cancellarla

La quarta ed ultima task sarà verificare la notifica ricevuta e cancellarla [2 minuti]



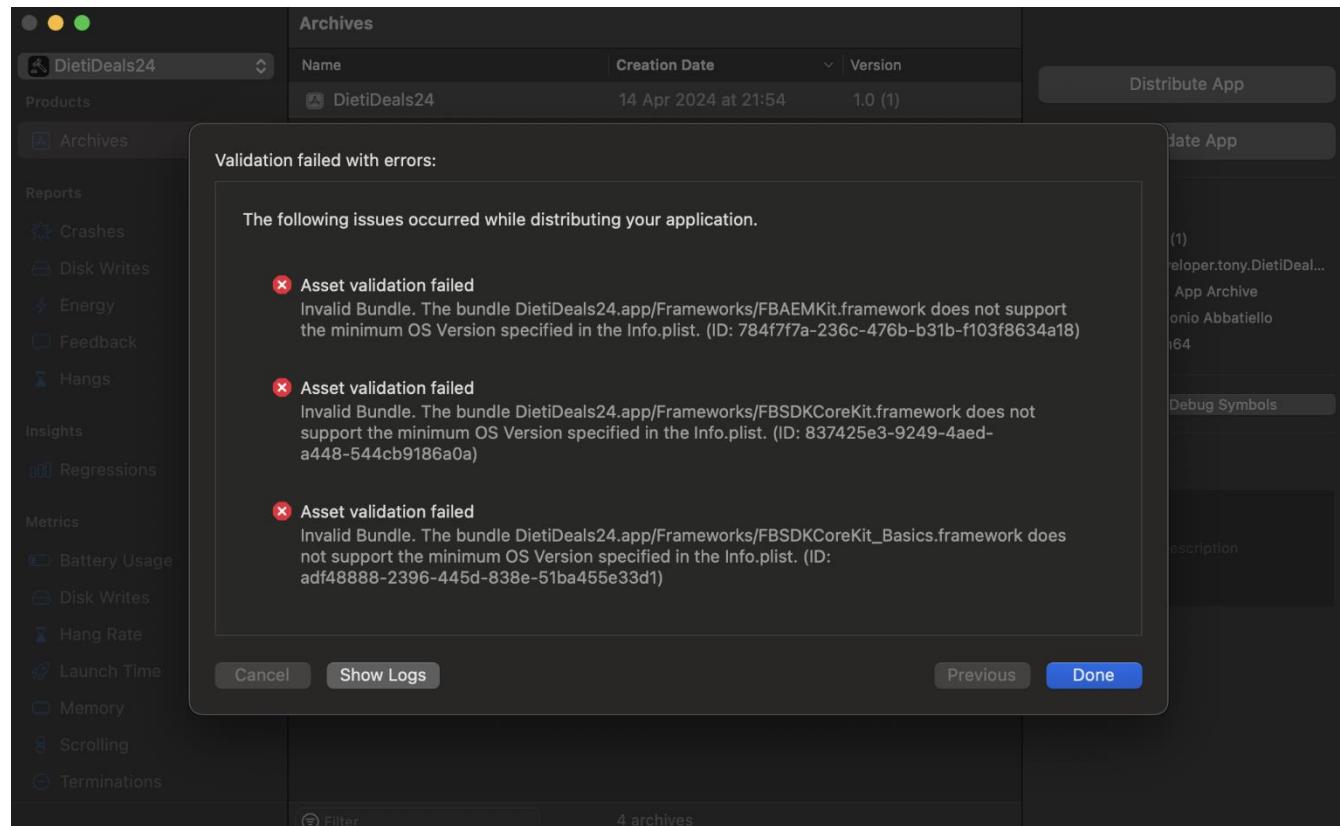
Utente	Esito Task
Utente 1	Success (1m 12s)
Utente 2	Success (0m 56 s)
Utente 3	Success (0m 50s)
Utente 4	Success (1m 4s)

Assegnando il valore 1 per le attività svolte con successo (S) e il valore 0.5 per le attività svolte parzialmente o completate oltre il tempo previsto (P), il tasso di successo può essere calcolato come il rapporto tra la somma del valore assegnato alle attività S e il valore assegnato alle attività P, e il numero totale di attività eseguite, cioè: Tasso di successo = $\frac{(15*1+1*0.5)}{16} \cong 97\%$

5.2.2 Analisi File di Log

Per l'analisi di file dei log si è scelto di utilizzare **AppStoreConnect** in combinazione con **TestFlight** che consentono di rilasciare app per Alpha-Beta testing e ci permettono di accedere ai log registrati sui dispositivi che sceglieranno di testare l'app, purtroppo al momento della stesura del documento non ci è stato possibile caricare l'app sulle sopracitate piattaforme a causa di un'errore della libreria FacebookSDK e di un bug presente nell'ultima versione di XCode, problematica riscontrata anche da altri utenti e documentata sui forum Apple.

(<https://forums.developer.apple.com/forums/thread/705297>), pertanto i file di log non saranno inclusi.



Dimostrazione della problematica riscontrata

