



Università degli studi di Napoli Parthenope

Progetto Ingegneria del Software e Interazione Uomo Macchina

Parthenope

System Design Document

Gruppo:

Riccardo Andrea Spinosa

0124002253

Alessandro massadoro

0124002450

Professori:

Mariacarla Staffa

Paola Barra

Simple Sell

IMPORTANTE!

L'indice che segue è basato sul documento word del template Modello Analisi pubblicato su elearning .

Premessa Importante:	3
1.0 Introduction	3
2.0 Sistema Proposto:	4
2.1 Panoramica	4
2.2 Requisiti Funzionali (FR)	6
2.3 Requisiti non funzionali (NFR)	7
2.4 Pseudo Requisiti	8
2.5.0 Modelli del sistema	8
2.5.1 Utenti del sistema	8
2.5.2 Scenari	8
2.5.2.1 Primo Scenario	8
2.5.2.2 Secondo Scenario	9
2.5.2.3 Terzo scenario	10
2.5.2.4 Quarto Scenario	11
2.5.3 Use case model	11
2.5.4 Use Case Description	12
2.5.5 Modello Ad Oggetti	22
2.5.5.1 Dizionario dei dati	22
2.5.5.2 Diagramma Delle Classi	23
2.5.6 Modelli dinamici	25
2.5.6.1 Diagrammi delle Sequenze	25
2.5.6.1.1 Registrazione	26
2.5.6.1.2 Login	26
2.5.6.1.3 Inserisci Annuncio	27
2.5.6.1.4 Modifica Annuncio	27
2.5.6.1.5 Elimina Annuncio	28
2.5.6.1.6 Visualizza Wallet	28
2.5.6.1.7 Scarica Fattura	29
2.5.7 Design Pattern	30
2.5.7.1 Singleton	31
2.5.7.2 Factory	32
2.5.7.3 Observer	34
2.5.8 Interfaccia utente	35
2.5.8.1 Realizzazione del prototipo:	36
2.5.8.2 Idea Interfaccia	37

2.5.8.3 Mock-ups:	40
2.5.9. Testing	54
2.5.9.1 Obiettivo dei test:	54
2.5.9.2 Metodologia usata	54
3.0 Analisi di mercato	58
4.0 Future Implementazioni	59
5.0 Glossario	59

Premessa Importante:

Per una migliore leggibilità dei diagrammi e del progetto, sono stati omessi alcuni casi d'uso o funzionalità che non riguardano direttamente l'idea del progetto. Ad esempio password dimenticata è una procedura scontata all'interno del login che non è stata appositamente aggiunta.

1.0 Introduction

L'obiettivo del sistema vuole essere la realizzazione di una dashboard che permetta di gestire e aggiungere annunci attraverso un'interfaccia intuitiva. L'utente non registrato che si interfaccia la prima volta al sistema dovrà eseguire obbligatoriamente la registrazione (email,password,partita iva, etc.) .

Si premette che Alcuni utenti hanno la partita iva, e possono eseguire operazioni aggiuntive su piattaforme specifiche che la richiedono (amazon, wish...) altri no e dunque potranno interfacciarsi solo a piattaforme che non la richiedono.

Gli annunci, ripartiti in piattaforme diverse, sono gestiti direttamente dalla nostra app. Collezionano tutti gli annunci inseriti nelle varie numerose piattaforme. Una volta registrato l'utente potrà effettuare il login e interfacciarsi al sistema. Nello specifico, la dashboard reindirizza l'utente alla home e mostra tutti gli annunci attivi delle varie piattaforme. La dashboard con apposita sezione permette di aggiungere nuovi annunci, composti da foto del prodotto selezionando le piattaforme di pubblicazione e altre info necessarie alla sua pubblicazione. In maniera intuitiva sarà possibile effettuare dall'applicazione eventuali modifiche dell'inserzione o la sua eliminazione. La sezione cerca, permetterà di trovare in maniera rapida un annuncio.

Ogni annuncio invia periodicamente in maniera real-time notifiche all'utente per segnalare eventi quali vendita del prodotto con specifica piattaforma, o messaggi con apposita sezione segnalando la specifica piattaforma sulla quale è stato ricevuto.

La dashboard offrirà la sezione profilo, dove potranno essere visualizzate varie informazioni e sarà possibile in tale sezione scaricare la fattura o visualizzare il wallet- Il wallet conterrà solo informazioni riguardanti le transazioni degli articoli venduti.(+50€ -> amazon)

Il tutto verrà gestito tramite APIs offerte dalle varie piattaforme.

esempio APIs offerte: <https://developer.ebay.com/develop/apis>

2.0 Sistema Proposto:

2.1 Panoramica

La dashboard consente la gestione e l'aggiunta di annunci attraverso un'interfaccia intuitiva. Gli utenti devono registrarsi e i registrati possono effettuare il login. La dashboard visualizza tutti gli annunci attivi delle varie piattaforme, permettendo di

aggiungere, modificare e eliminare annunci con foto del prodotto e altre info. La sezione cerca consente di trovare rapidamente un annuncio e la sezione profilo visualizza informazioni come la fattura e il wallet per le transazioni degli articoli venduti. La dashboard invia notifiche in tempo reale per eventi come la vendita di prodotto su specifiche piattaforme.

Gli obiettivi del sistema sono:

- Consentire la gestione e l'aggiunta di annunci attraverso un'interfaccia intuitiva
- Registrare gli utenti
- Consentire agli utenti registrati di effettuare il login
- Visualizzare tutti gli annunci attivi delle varie piattaforme
- Permettere agli utenti di aggiungere, modificare e eliminare annunci con foto del prodotto e altre informazioni
- Consentire di trovare rapidamente un annuncio
- Visualizzare informazioni come la fattura e il wallet per le transazioni degli articoli venduti
- Inviamo notifiche in tempo reale per eventi come la vendita di prodotto su specifiche piattaforme.

2.2 Requisiti Funzionali (FR)

Sia i requisiti funzionali che quelli non funzionali descrivono caratteristiche specifiche che un prodotto deve avere per soddisfare le esigenze degli stakeholder e del business stesso. Ma si concentrano su cose diverse.

FR1:

Visualizzare annunci attivi: l'utente deve poter visualizzare i propri annunci attivi inseriti nelle varie piattaforme, in maniera semplice ed intuitiva nell'apposita sezione Home.

FR2:

Notifica: La dashboard può inoltrare agli utenti delle notifiche:

1) Notifica di messaggio: il sistema segnala la presenza di un messaggio inviato da una specifica piattaforma su uno specifico annuncio.

2) Notifica vendita: il sistema segnala che l'annuncio è stato venduto e la piattaforma di appartenenza tutto nell'apposita sezione notifiche.

FR3:

L'utente deve poter modificare gli annunci delle diverse piattaforme nella sezione Home.

FR4:

L'utente deve poter aggiungere nuovi annunci permettendo di scegliere le varie piattaforme dove decide di inserire l'annuncio.

FR5:

L'utente deve poter eliminare gli annunci delle diverse piattaforme nella sezione Home.

FR6:

Wallet : la Dashboard permette, di tenere traccia di tutti i versamenti ricevuti inerenti alle vendite, tale portafoglio è solo virtuale ad esso non verrà associata nessuna carta fisica, ma al suo interno verranno salvate solo le informazioni associate mostrate dalle varie piattaforme del pagamento ricevuto. Esso sarà nell'apposita sezione profilo

FR7:

Ricerca : L'utente deve poter cercare in maniera veloce un annuncio scrivendo il nome dell'annuncio sopra la barra ricerca nella Sezione Home

FR8:

Registrazione: La dashboard permette a nuovi utenti di registrarsi con o senza partita iva, tale informazione è necessaria in quanto alcuni servizi di vendita quali amazon,wish,aliexpress sono dedicati esclusivamente ai clienti con partita iva.

FR9

Login: L'applicazione permette a questi ultimi di effettuare l'accesso tramite le credenziali inserite nella fase di registrazione, o per chi preferisce con le credenziali raccolte con accesso con autenticatori autorizzati(Google,Yahoo,Facebook...).

2.3 Requisiti non funzionali (NFR)

NFR1:

Usabilità: La dashboard deve avere un'interfaccia chiara e semplice da utilizzare.

NFR2:

Conformità alle linee guida: L'interfaccia deve seguire le linee guida del Interface Guidelines di Apple.

NFR 3:

Throughput: Il sistema deve essere in grado di immagazzinare e processare almeno 10.000 interazioni al minuto.

NFR 4:

Disponibilità: Il sistema si adatterà in base al tipo utente che effettua l'accesso.

NFR 5:

Sicurezza: Il sistema deve essere sicuro.

2.4 Pseudo Requisiti

Implementazione

Il sistema deve essere sviluppato in Swift(Native Ios App),

2.5.0 Modelli del sistema

2.5.1 Utenti del sistema

Gli utenti del sistema possono essere suddivisi in due categorie: quelli con partita iva e quelli senza. Il primo gruppo, avendo la partita iva, può eseguire operazioni aggiuntive su piattaforme specifiche che la richiedono (ad esempio Amazon, Wish). Il secondo gruppo, non avendo la partita iva, può interfacciarsi solo a piattaforme che non la richiedono. Questi utenti possono utilizzare la dashboard per gestire e aggiungere annunci attraverso un'interfaccia intuitiva, effettuare modifiche o eliminazioni e cercare annunci. Inoltre, ricevono notifiche periodiche in real-time su eventi come vendite e messaggi. Il contesto di utilizzo riguarda la gestione di annunci su diverse piattaforme.

2.5.2 Scenari

2.5.2.1 Primo Scenario

Nome Scenario	Scenario 1
Attori Partecipanti	Noa Festa: Senza partita iva
Flusso Eventi	<p>1. Noa Festa, utente senza partita iva, accede alla dashboard utilizzando la funzione di “Login”.</p> <p>2. Noa una volta effettuato l'accesso può usare la funzione “ Aggiungi Annuncio” , impostando che tale annuncio venga inserito su ebay e facebook marketplace</p> <p>3. Noa conseguentemente alla aggiunta dell'annuncio, visualizza nella sezione home come procede l'annuncio inserito.</p> <p>4. La dashboard segnala due notifiche. La prima notifica alle ore 13:30 un messaggio di richiesta informazioni di un cliente da ebay.</p> <p>5. La seconda notifica segnala che: alle ore 14:00 ha venduto il prodotto che precedentemente aveva inserito.</p>

2.5.2.2 Secondo Scenario

Nome Scenario	Scenario 2
Attori Partecipanti	Maria Rombo: Utente con partita iva:
Flusso Eventi	<p>1. Maria, utente con partita Iva e venditrice di software, effettua il Login per monitorare lo stato dei suoi annunci.</p> <p>2. Maria, una volta effettuato l'accesso, utilizzando la barra di ricerca, cerca l'annuncio precedentemente inserito.</p>

Nome Scenario	Scenario 2
	<p>3. Maria, resasi conto di aver inserito un annuncio sbagliato, effettua l'eliminazione di tale annuncio attraverso la funzionalità "Elimina Annuncio".</p>

2.5.2.3 Terzo scenario

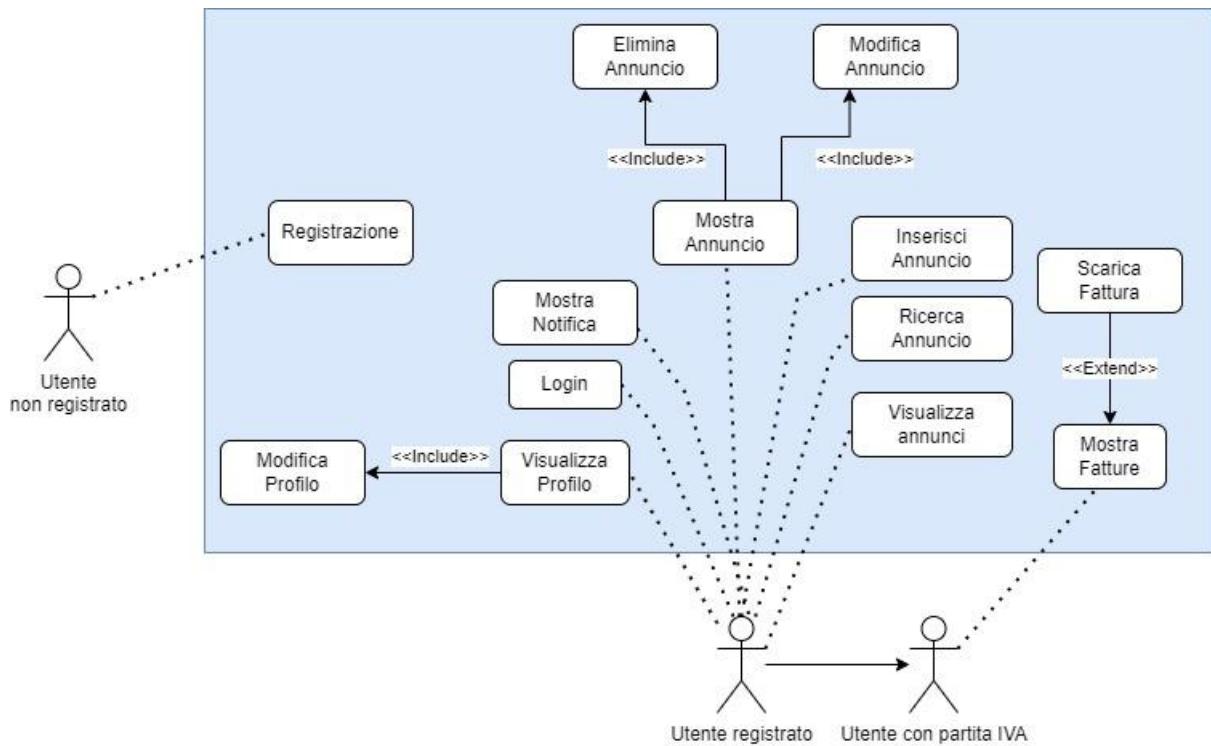
Nome Scenario	Scenario 3
Attori Partecipanti	Riccardo Andrea Spinosa: Utente non registrato:
Flusso Eventi	<p>1.Riccardo Andrea Spinosa,non registrato, si reca nell'applicazione, curioso di capire di cosa si tratta.</p> <p>2.Riccardo, aperta l'applicazione, nota che deve effettuare il login o registrarsi per potersi interfacciare.</p> <p>3. Riccardo è convinto dell'applicazione, si registra ed effettua il login.</p> <p>4 A questo punto Riccardo, utente senza partita iva crea il suo primo annuncio postato su Ebay.</p>

2.5.2.4 Quarto Scenario

Nome Scenario	Scenario 4
Attori Partecipanti	Alessandro Massadoro: Utente con partita iva:
Flusso Eventi	<p>1. Alessandro Massadoro, utente registrato con partita Iva, effettua il login e attraverso APIs, inserirà nuovamente un annuncio scaduto su ebay.</p> <p>2. Alessandro si è reso conto che l'annuncio scaduto non fosse efficace, così decide di modificarlo, cambiando titolo, prezzo e descrizione</p>

2.5.3 Use case model

Password dimenticata è una procedura scontata all'interno del login che non è stata appositamente aggiunta per una maggiore lettura del diagramma in quanto la corretta lettura e comprensione dei diagrammi non è un optional.



2.5.4 Use Case Description

2.5.4.1

Nome Scenario	Modifica Profilo
Attori Partecipanti	Utente
Condizioni d'ingresso:	I l'utente è loggato in app
Flusso di eventi:	<ol style="list-style-type: none"> L'utente avvia la funzione "Visualizza Profilo" L'app risponde visualizzando il profilo dell'utente con tutte le sue informazioni e una serie di operazioni tra cui "modifica profilo" L'utente sceglie quindi di modificare alcuni dati del suo profilo, inserisce quindi i nuovi dati. L'app aggiorna tutti i campi modificati dall'utente

Nome Scenario	Modifica Profilo
Condizioni di uscita:	L'utente termina la modifica.

2.5.4.2

Nome Scenario	Scarica Fattura
Attori Partecipanti	Utente
Condizioni d'ingresso:	I'utente è loggato in app come utente con partita iva
Flusso di eventi:	<p>1. L'utente avvia la funzione “Mostra Fattura”</p> <p>2. l'app risponde e mostrando la funzionalità “Scarica Fattura” permette il download delle fatture</p> <p>3. L'utente con partita iva preme sul pulsante “Scarica Fattura”</p> <p>4. L'app esegue il download delle fatture in modo tale che queste risultino essere presenti sul dispositivo dell'utente.</p>

Nome Scenario	Scarica Fattura
Condizioni di uscita:	La fattura viene scaricata

2.5.4.3

Nome Scenario	Inserisci annuncio
Attori Partecipanti	Utente
Condizioni d'ingresso:	l'utente è loggato in app
Flusso di eventi:	<ol style="list-style-type: none"> 1. L'utente avvia la funzione “Inserisci Annuncio”, icona del + 2. L'app risponde e l'utente può inserire tutti i dati necessari

Nome Scenario	Inserisci annuncio
	<p>3. L'utente riempie tutti i dati necessari per la creazione dell'annuncio, l'utente inserisce quindi il titolo, la foto del prodotto, il prezzo , la categoria a cui appartiene, la marca e una descrizione, seleziona infine su quali piattaforme vuole che l'annuncio venga inserito</p> <p>4. L'app riceve tutte le informazioni inserire e tramite APIs procede a postare l'annuncio sulle piattaforme indicate dall'utente</p>
Condizioni di uscita:	L'utente ha inserito tutti i campi necessari e posta l'annuncio

2.5.4.4

Nome Scenario	Modifica Annuncio
Attori Partecipanti	Utente
Condizioni d'ingresso:	l'utente è loggato in app

Nome Scenario	Modifica Annuncio
Flusso di eventi:	<p>1. L'utente avvia la funzione “Ricerca”</p> <p>2. L'app risponde visualizzando l'ordine trovato con il titolo indicato dall'utente, mostra quindi tutti i dati dell'annuncio reperibili.</p> <p>3. L'utente decide di voler modificare l'annuncio, esegue quindi la funzione “Modifica Annuncio”</p> <p>4. L'app mostrerà tutti i campi modificabili</p> <p>5. L'utente sceglie di modificare descrizione e il prezzo in quanto l'annuncio non è più competitivo con i prezzi del mercato, e pensa che la descrizione presenti degli errori.</p> <p>6. L'app aggiornerà i campi modificati dall'utente su tutte le piattaforme che contenevano quell'annuncio, tramite APIs.</p>
Condizioni di uscita:	L'utente termina la modifica dell'annuncio

2.5.4.5

Nome Scenario	Elimina Annuncio
Attori Partecipanti	Utente
Condizioni d'ingresso:	I'utente è loggato in app
Flusso di eventi:	<p>1. L'utente avvia la funzione "Ricerca"</p> <p>2. L'app risponde visualizzando l'ordine trovato con il titolo indicato dall'utente, mostra quindi tutti i dati dell'annuncio reperibili.</p> <p>3. L'utente decide di voler modificare l'annuncio, ma sbaglia ed esegue la funzione "Elimina Annuncio"</p> <p>4. L'app mostra un messaggio chiedendo se l'utente è sicuro di quell'operazione, informandolo che l'operazione una volta eseguita è irreversibile.</p> <p>5. L'utente sceglie quindi di annullare la sua scelta.</p>
Condizioni di uscita:	L'utente annulla l'operazione

2.5.4.6

Nome Scenario	Elimina Annuncio 2
Attori Partecipanti	Utente
Condizioni d'ingresso:	I'utente è loggato in app
Flusso di eventi:	<ol style="list-style-type: none">1. L'utente avvia la funzione "Ricerca"2. L'app risponde visualizzando l'ordine trovato con il titolo indicato dall'utente, mostra quindi tutti i dati dell'annuncio reperibili.3. L'utente decide di voler eliminare l'annuncio, esegue quindi la funzione "Elimina Annuncio"4. L'app mostra un messaggio chiedendo se l'utente è sicuro di quell'operazione, informandolo che l'operazione una volta eseguita è irreversibile.5. L'utente sceglie di continuare con la sua operazione6. L'app esegue l'eliminazione dell'annuncio da tutte le piattaforme su cui era stato inserito.
Condizioni di uscita:	L'utente elimina l'annuncio

2.5.4.7

Nome Scenario	Visualizza Annunci
Attori Partecipanti	Utente
Condizioni d'ingresso:	l'utente è loggato in app
Flusso di eventi:	<ol style="list-style-type: none">1. L'utente accede all'home2. L'app risponde visualizzando tutti gli annunci attivi3. L'utente scorre gli annunci
Condizioni di uscita:	Richiesta di una nuova operazione

2.5.4.8

Nome Scenario	Login
Attori Partecipanti	Utente
Condizioni d'ingresso:	l'utente non loggato in app
Flusso di eventi:	<ol style="list-style-type: none">1. L'utente inserisce i dati e avvia la funzione "Login"2. L'app risponde controllando i dati inseriti dall'utente che verrà fatto accedere se i dati sono corretti

Nome Scenario	Login
	3. L'app non fa ecedere al sistema
Condizioni di uscita:	Dati errati

2.5.4.9

Nome Scenario	Mostra Notifiche
Attori Partecipanti	Utente
Condizioni d'ingresso:	I'utente non loggato
Flusso di eventi:	<p>1. L'utente è interessato alle notifiche per cui esegue la funzionalità "Mostra Notifiche" cliccando sull'icona delle notifiche.</p> <p>2. L'app risponde mostrando tutte le notifiche provenienti dalle piattaforme in cui l'utente ha inserito annunci</p>
Condizioni di uscita:	Nuova operazione

2.5.4.10

Nome Scenario	Ricerca Annunci
Attori Partecipanti	Utente
Condizioni d'ingresso:	I'utente in app

Nome Scenario	Ricerca Annunci
Flusso di eventi:	<p>1. L'utente effettua la ricerca per un singolo annuncio esegue quindi la funzione "Ricerca"</p> <p>2. L'app risponde ricercando il titolo dell'annuncio che ha inserito e mostrando i risultati all'utente.</p>
Condizioni di uscita:	Annuncio restituito

2.5.4.11

Nome Scenario	Registrazione
Attori Partecipanti	Utente
Condizioni d'ingresso:	I'utente non registrato
Flusso di eventi:	<p>1. L'utente accede al sistema per la prima volta, esegue quindi la funzione di "Registrazione"</p> <p>2. L'app risponde mostrando il form da completare per potersi registrare all'app</p> <p>3. L'utente riempie tutti i campi necessari per registrarsi, inserisce quindi il proprio Nome, Cognome, Email ecc...</p> <p>3. L'app riporta l'utente in fase di login</p>

Nome Scenario	Registrazione
Condizioni di uscita:	Operazione annullata

2.5.5 Modello Ad Oggetti

2.5.5.1 Dizionario dei dati

Il dizionario dei dati è un registro che contiene informazioni dettagliate sui dati presenti in un sistema di gestione dei dati, come database o data warehouse. Il dizionario dei dati descrive le caratteristiche dei dati, come il tipo di dati, la lunghezza, le restrizioni, i valori ammissibili, la provenienza e così via.

Il dizionario dei dati è utilizzato per fornire informazioni a persone che lavorano con i dati, come sviluppatori di software, analisti di dati e altri utenti aziendali. Queste informazioni possono aiutare a comprendere meglio i dati e a utilizzarli in modo più efficace.

Il dizionario dei dati è anche utilizzato come fonte di informazioni per la progettazione di database, per assicurarsi che i dati siano archiviati in modo consistente e che sia possibile recuperare e utilizzare i dati in modo efficiente. Inoltre, il dizionario dei dati può essere utilizzato per monitorare i cambiamenti nei dati e garantire che la qualità dei dati sia mantenuta.

Nel nostro caso il nostro sistema di gestione dei dati è un filesystem per tanto verranno riportate tutte le informazioni necessarie per gestire i dati cruciali del nostro sistema.

Utente:

Email (chiave primaria, stringa, lunghezza massima 100 caratteri, univoca, non nulla)
 Password (stringa, lunghezza massima 100 caratteri, lunghezza minima 8 deve contenere caratteri maiuscoli ,caratteri speciali e numeri,non nulla)
 Partita IVA (stringa, lunghezza massima 11 caratteri, opzionale)

Nome(stringa, lunghezza massima 100 caratteri, non nulla)
Cognome(stringa, lunghezza massima 100 caratteri, non nulla)
Telefono(minimo 9 caratteri, massimo 11 caratteri,stringa)

Transazione

-ID (stringa, chiave primaria, non nulla,stringa)
-Totale(float, non nullo)

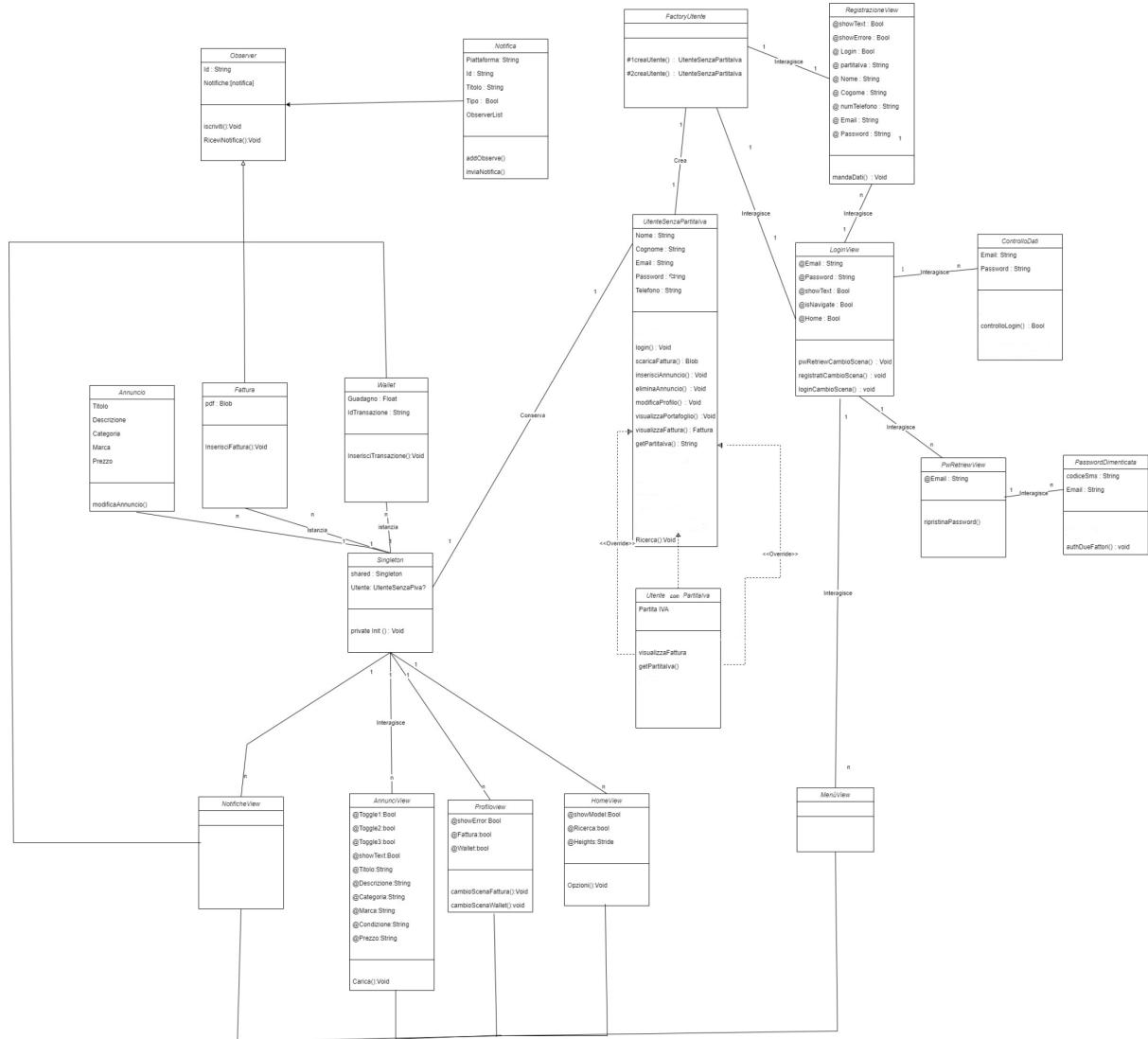
Fattura

-pdf(dato non strutturato non ha vincoli)

Piattaforme

dati necessari alla gestione delle APIs.

2.5.5.2 Diagramma Delle Classi



Alleghiamo lo jpeg dello schema

Note per una migliore comprensione del diagramma:

Linguaggio utilizzato: Swift.

Gli attributi con la @ sono state, variabili riferite ad elementi di Testo , TextField o booleani necessari per elementi dinamici e di spostamento tra le viste.

Alcuni @state, come “@Fattura, @Login, @IsPresented” sono utilizzato per permettere la visualizzazione di nuove viste

Le dipendenze(Fatture e Transazioni) sono inserite automaticamente nel database grazie all'observer delle notifiche, ogni volta che avviene una vendita la fattura viene scaricata nel filesystem e lo stesso avviene per la transazione del wallet.

Aggiungiamo il private al costruttore del singleton come meccanismo di protezione

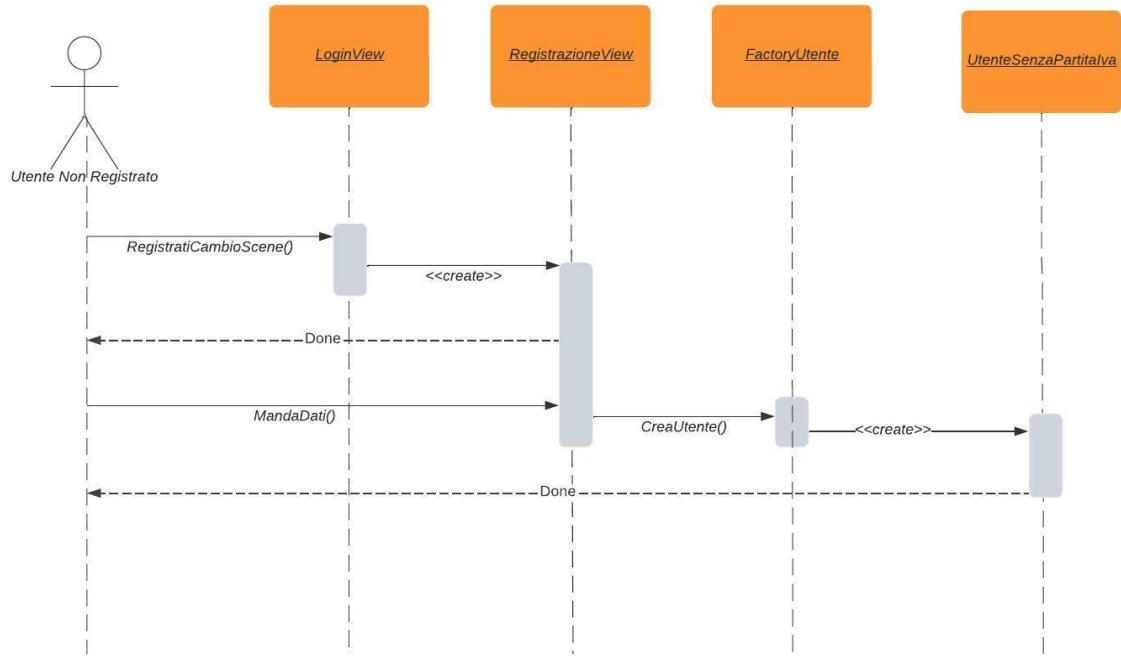
Le due view vuote sono riportate in quel modo in quanto, MenuView è messo a disposizione da swift per la rappresentazione delle viste, mentre NotificheView è la vista che contiene le notifiche ricevute

La classe Observer resta in osservazione delle informazioni ricevute dalle API

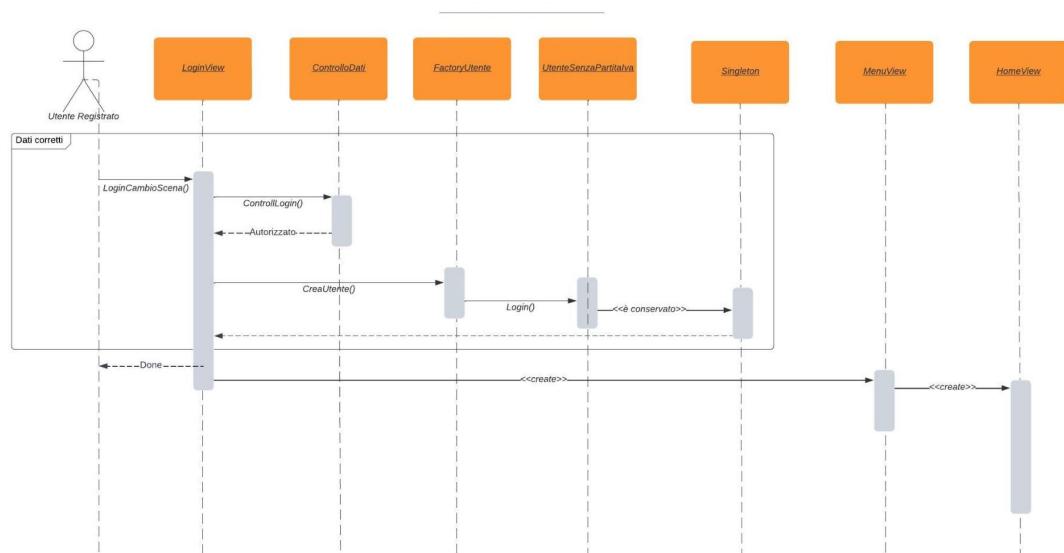
2.5.6 Modelli dinamici

2.5.6.1 Diagrammi delle Sequenze

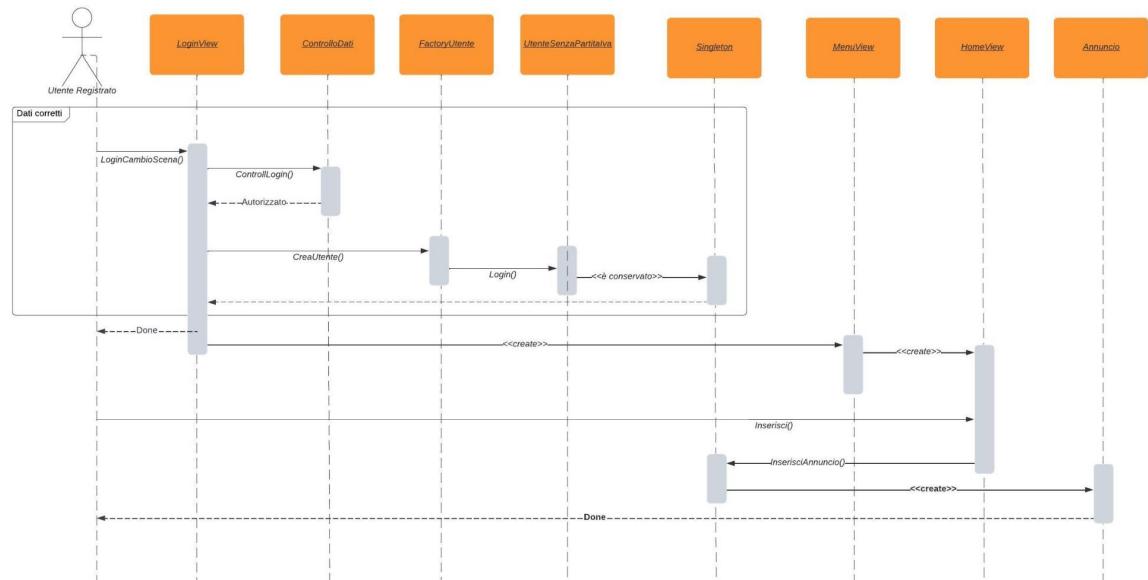
2.5.6.1.1 Registrazione



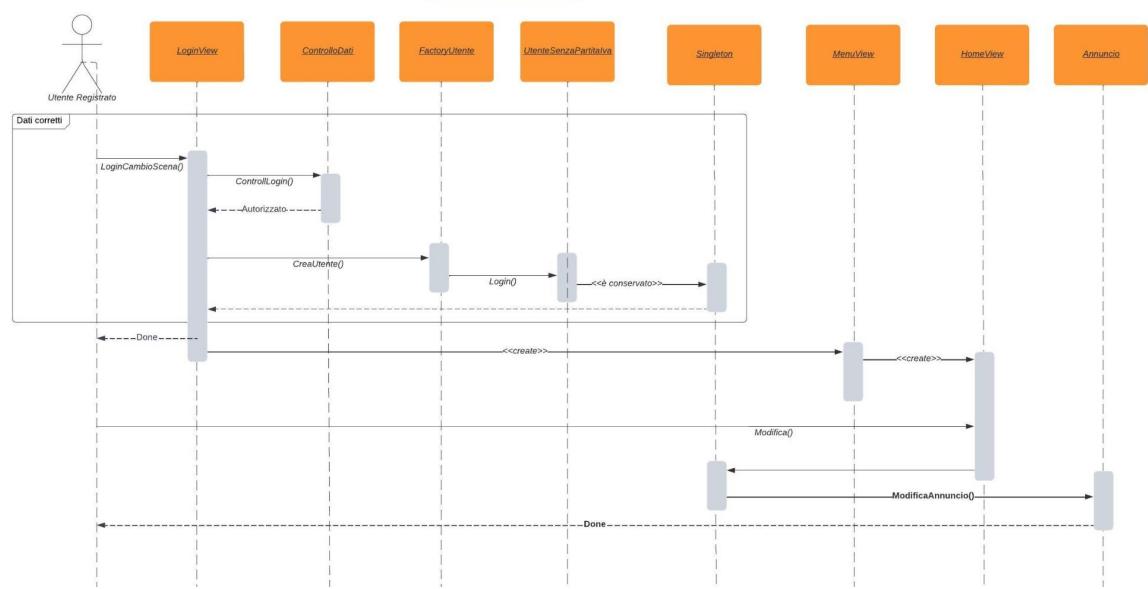
2.5.6.1.2 Login



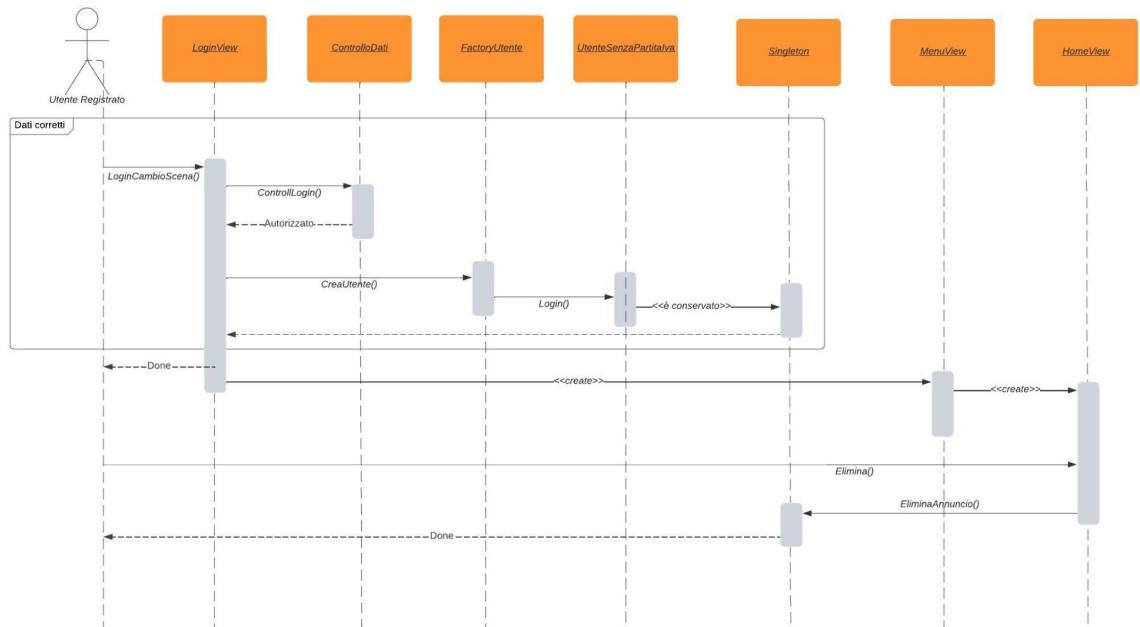
2.5.6.1.3 Inserisci Annuncio



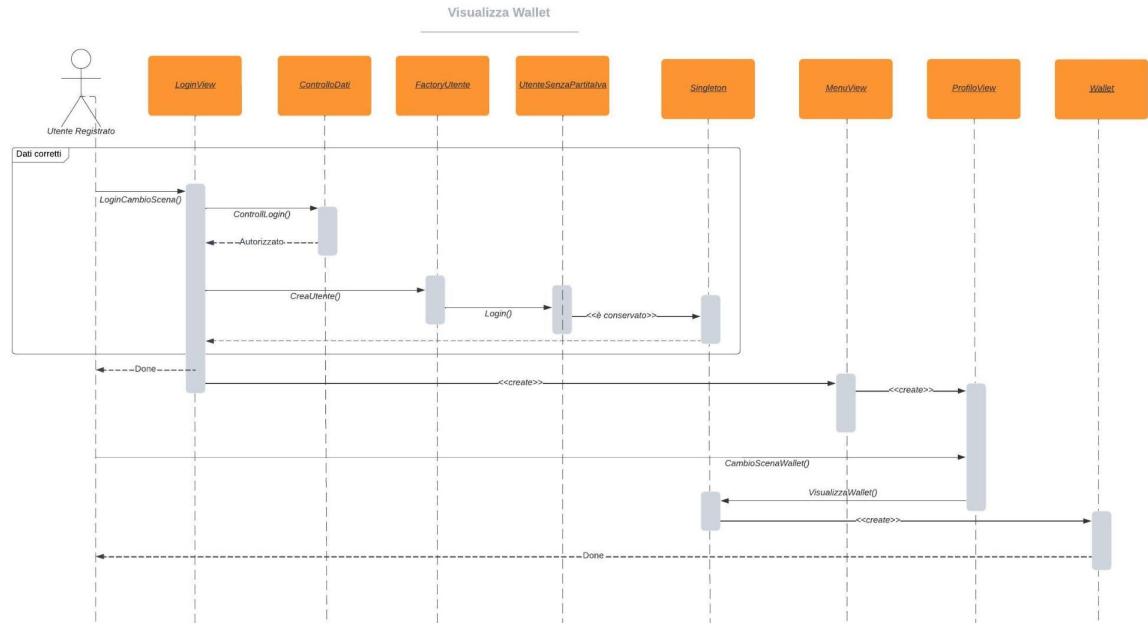
2.5.6.1.4 Modifica Annuncio



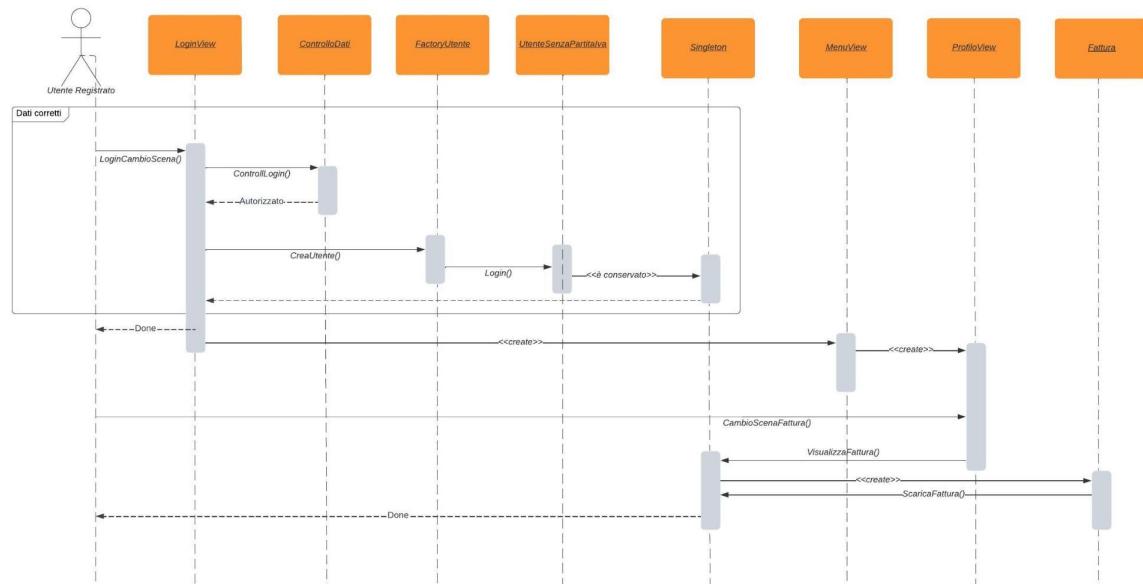
2.5.6.1.5 Elimina Annuncio



2.5.6.1.6 Visualizza Wallet



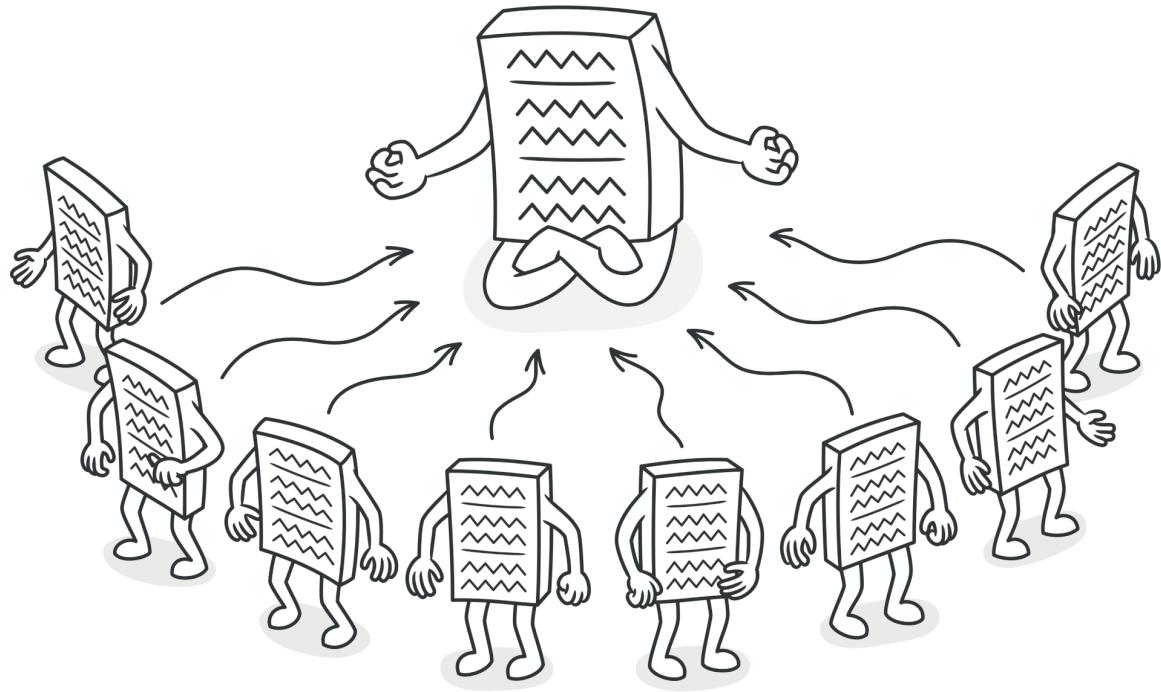
2.5.6.1.7 Scarica Fattura



2.5.7 Design Pattern

I design pattern sono delle soluzioni generali a problemi comuni che si presentano nella progettazione di software. Essi rappresentano un insieme di buone pratiche e linee guida che possono essere utilizzate per risolvere problemi specifici di progettazione in modo efficiente e mantenibile. La motivazione principale per utilizzare i design pattern durante la fase di progettazione è quella di fornire una soluzione collaudata a problemi comuni che si presentano nella progettazione di software. Ciò consente di ridurre i rischi di errori di progettazione e di aumentare la qualità e la manutenibilità del codice.

2.5.7.1 Singleton

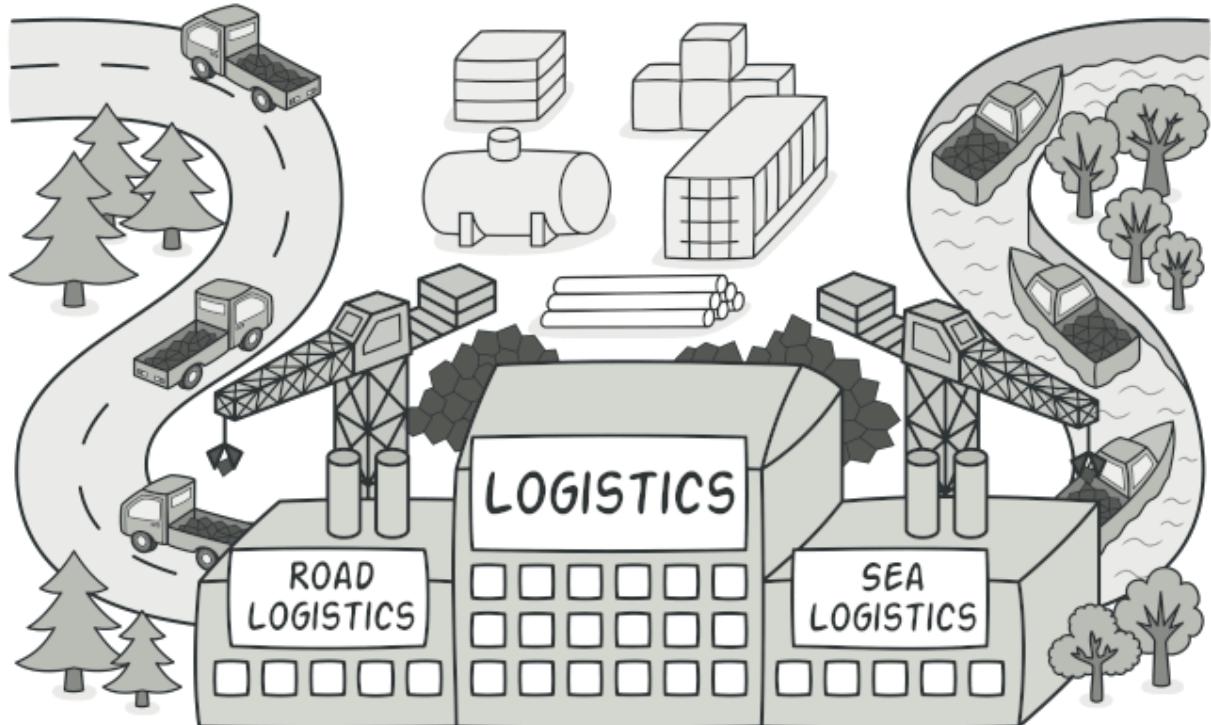


L'utilizzo del pattern Singleton è utile per la creazione di una classe per la gestione delle autorizzazioni o delle sessioni utente, in modo che solo una istanza di questa classe sia presente in tutta l'applicazione e che sia possibile accedere a questa istanza da qualsiasi punto dell'applicazione come ad esempio l'utente che accede al sistema. Inoltre il singleton può essere utilizzato in caso di utilizzo delle api, per gestire la connessione e la sessione con le api in modo da non dover creare più istanze o gestire la creazione di più sessioni con le api stesse.

Il singleton permette di garantire che solo un'istanza della classe sia presente in memoria, evitando di utilizzare risorse inutilmente o di avere più istanze in conflitto tra di loro.

```
8 import Foundation
9
10 class Singleton{
11     static let shared = Singleton()
12
13     private init(){}
14
15     var utente : UtenteSenzaPiva?
16
17
18
19 }
20
```

2.5.7.2 Factory

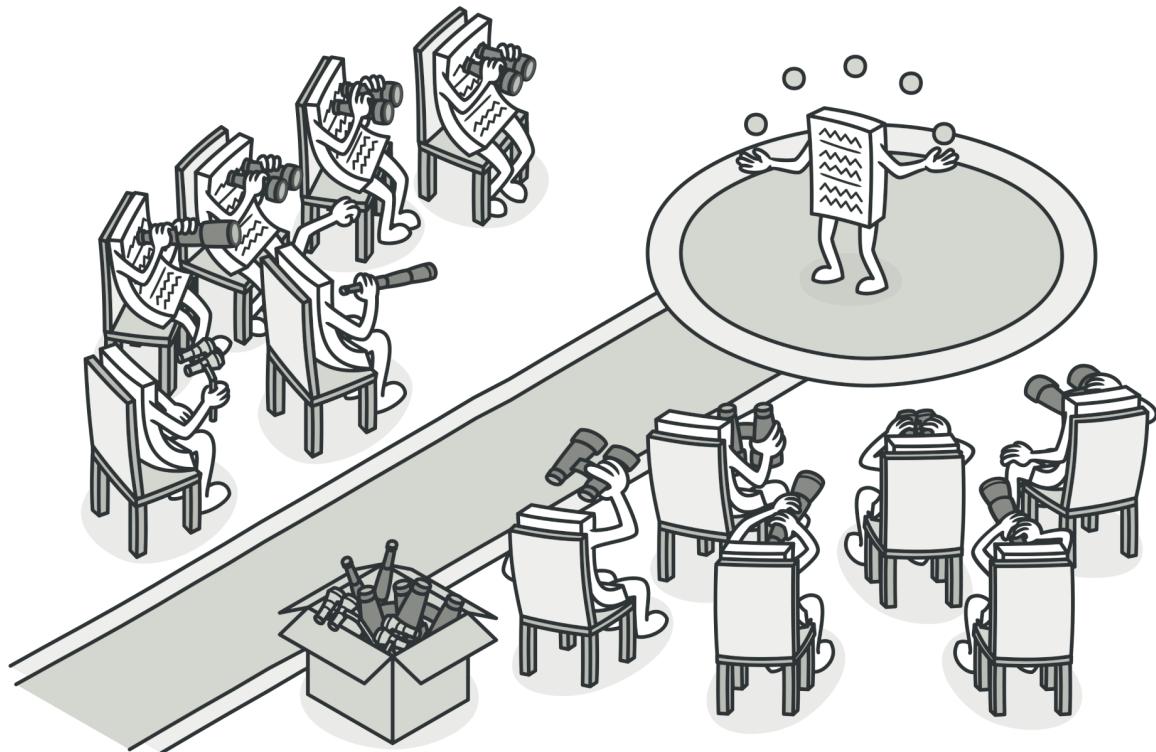


Il pattern Factory Method è un design pattern che fornisce un modo per creare oggetti in una sottoclasse senza specificare esplicitamente la classe di oggetti che vengono creati. Il pattern consiste nella definizione di un'interfaccia o di una classe astratta che specifica i metodi per creare oggetti, e poi nella creazione di sottoclassi che implementano questi metodi per creare oggetti specifici. Questo pattern viene utilizzato quando c'è la necessità di creare istanze di diverse sottoclassi che appartengono a una stessa classe astratta o interfaccia, ma non si vuole conoscere esattamente quale sottoclasse verrà istanziata. Il Factory Method offre una soluzione modulare per la creazione di oggetti, che rende il codice più flessibile e facile da mantenere.

```
class FactoryUtente
{
    //factory utente senza piva
    static func creaUtente(tel: String, email: String, pw: String, nome: String, cognome: String) -> UtenteSenzaPiva{
        var utente : UtenteSenzaPiva
        utente = UtenteSenzaPiva(tel:tel,email:email,pw:pw,nome:nome,cognome:cognome)
        return utente
    }

    //factory utente con piva
    static func creaUtente(tel: String, email: String, pw: String, nome: String, cognome: String, piva:String) -> UtenteSenzaPiva{
        var utente : UtenteSenzaPiva
        utente = UtentePiva(tel:tel,email:email,pw:pw,nome:nome,cognome:cognome,piva: piva)
        return utente
    }
}
```

2.5.7.3 Observer



Il pattern Observer è un pattern di progettazione che consente a un oggetto di essere avvisato automaticamente di eventuali cambiamenti di stato di un oggetto o di un insieme di oggetti a cui è interessato. In altre parole, è un modo per un oggetto di "osservare" gli stati degli altri oggetti e di essere notificato automaticamente quando questi cambiano. Il pattern Observer consente a più oggetti di essere notificati in modo asincrono quando un evento si verifica in un oggetto osservato. In questo caso, il pattern Observer è stato progettato per notificare gli utenti dell'app dei cambiamenti nello stato degli articoli in vendita, ad esempio quando un articolo viene venduto, ciò genererà una dovuta notifica che dovrà essere inviata in tempo reale all'utilizzatore del sistema così che tutte le dipendenze vengano aggiornate dopo l'avvenuta vendita.

In questo caso, serve un account veritiero per poter testare il funzionamento delle APIs per questo pattern, si necessita dunque obbligatoriamente di un account attivo, per tanto si è scelto di riportare solo un codice di esempio, al contrario dei precedenti portati che invece sono stati implementati e sono perfettamente funzionanti. L'esempio riportato è solo un esempio logico non realmente

implementato nella nostra applicazione ma è stato solo strutturato così che possa essere implementato come futura implementazione.

```
class Observer {
    |
    private var id = UUID().uuidString
    var notifiche = [Notifica]()

    func iscriviti(notifica: Notifica) {
        notifica.addObserver(observer: self)
    }

    func riceviNotifica(notifica: Notifica) {
        notifiche.append(notifica)
        print("Observer \(id) ha ricevuto la notifica: \(notifica.titolo)")
    }
}

extension Notifica {
    private var observerList = [Observer]()

    func addObserver(observer: Observer) {
        observerList.append(observer)
    }

    func inviaNotifica() {
        observerList.forEach { $0.riceviNotifica(notifica: self) }
    }
}
```

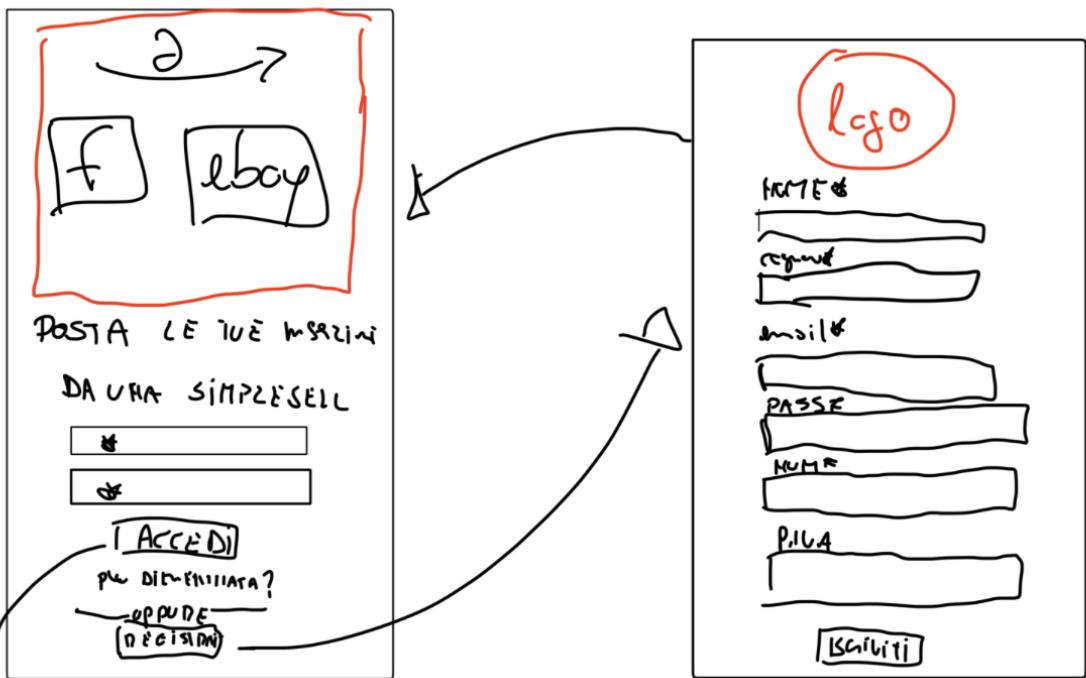
2.5.8 Interfaccia utente

2.5.8.1 Realizzazione del prototipo:

- **Scopo:** Il prototipo, all'interno dello spazio tridimensionale dello "scopo" dei prototipi, è posizionato in prossimità del "vertice" associato all'interfaccia. Il sistema, infatti, è rivolto ad un pubblico estremamente ampio e variegato, pertanto l'usabilità del sistema è di fondamentale importanza. Lo scopo del progetto è semplificare un processo per gli utenti e renderlo più intuitivo. Il prototipo, dunque, ha l'obiettivo di valutare l'usabilità dell'interfaccia; pertanto, viene implementata la reale esperienza utente, simulando la logica sottostante. Il prototipo, inoltre, è utile a verificare che ruolo ricoprirebbe il prodotto finale nella vita quotidiana di un utente.
 - **Modalità d'uso:** Dal momento che il prototipo è utile a verificare l'esperienza utente nell'utilizzo del sistema reale, la modalità d'uso, quindi, è *interattiva*. Vengono infatti verificate tutte le possibili azioni che l'utente può eseguire
 - **Fedeltà del prototipo:** "Alta fedeltà" perché bisogna valutare l'interfaccia effettiva che utilizzeranno gli utenti e verificare che le modalità di interazione siano chiare agli utenti.
 - **Completezza funzionale:** Il prototipo si focalizza sull'utilizzo delle funzionalità principali dell'applicazione da parte dell'utente. Non sono implementate, dunque, le funzioni che non sono necessarie all'obiettivo preposto in fase di progettazione ovvero testare l'efficacia dell'usabilità dell'interfaccia. Ad esempio è stato fornito un utente di prova per poter accedere al sistema per tanto la gestione del profilo non è implementata per altri utenti ed anche scarica fattura non è stato implementato in quanto scaricare effettivamente un pdf contenente la fattura non influiva e ne soddisfaceva i requisiti progettuali imposti fase di progettazione. Sono quindi implementate soltanto le funzioni chiave del sistema.
-
- **Durata:**
Nel nostro caso il prototipo è di tipo evolutivo in quanto viene costantemente migliorato e sviluppato nel tempo, con l'obiettivo di diventare un prodotto finale. Inoltre è interattivo in quanto il prototipo è progettato per essere utilizzato e testato con utenti reali, con l'obiettivo di raccogliere feedback e migliorare la soluzione.
 - **Tipo di prototipo:** applicazione IOs.

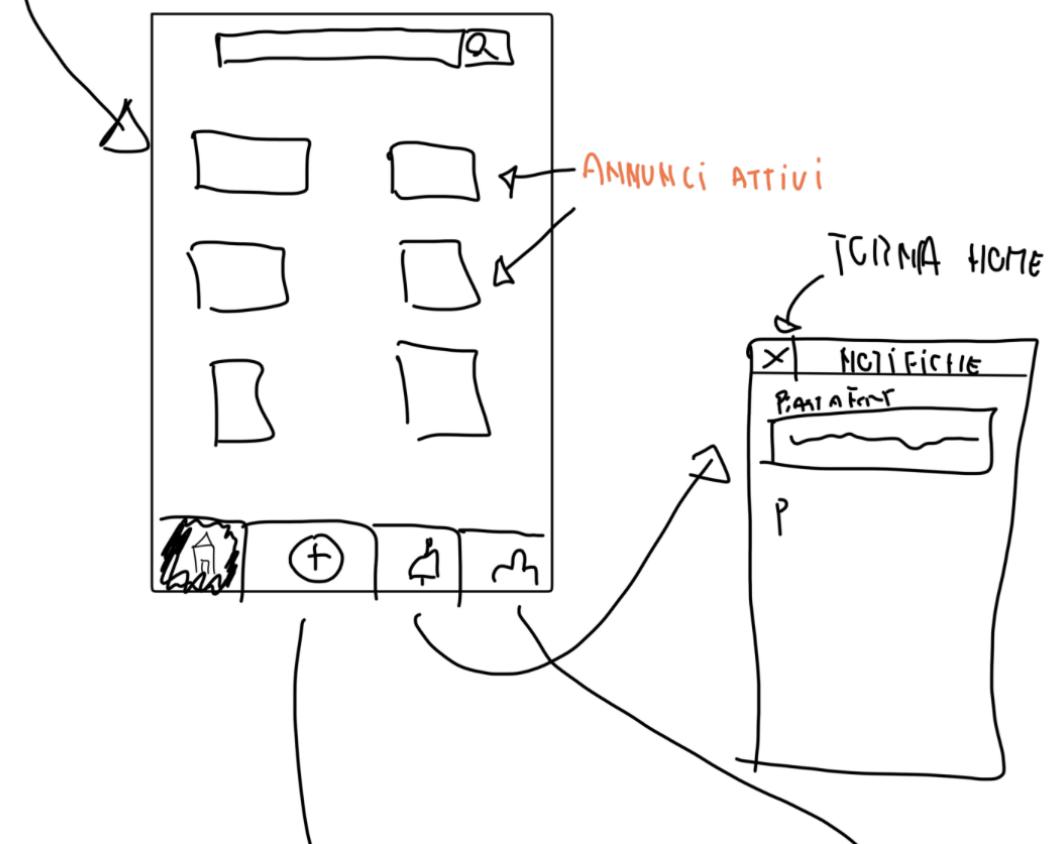
2.5.8.2 Idea Interfaccia

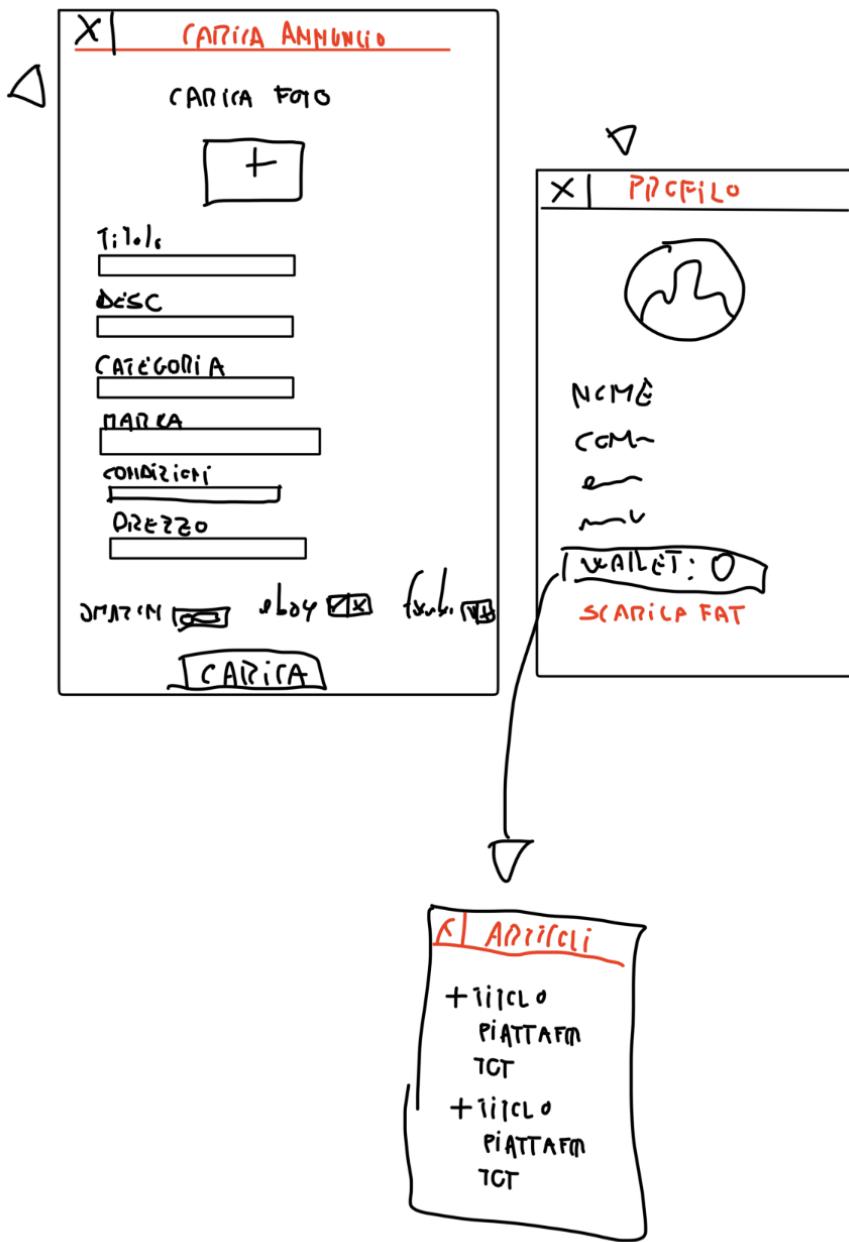
Per rendere chiaro il modo in cui abbiamo lavorato si riporta lo schizzo dell'interfaccia da realizzare.



Logo

NAME
cognome
email
PASS
NUM
PIVA
ISCRITTI

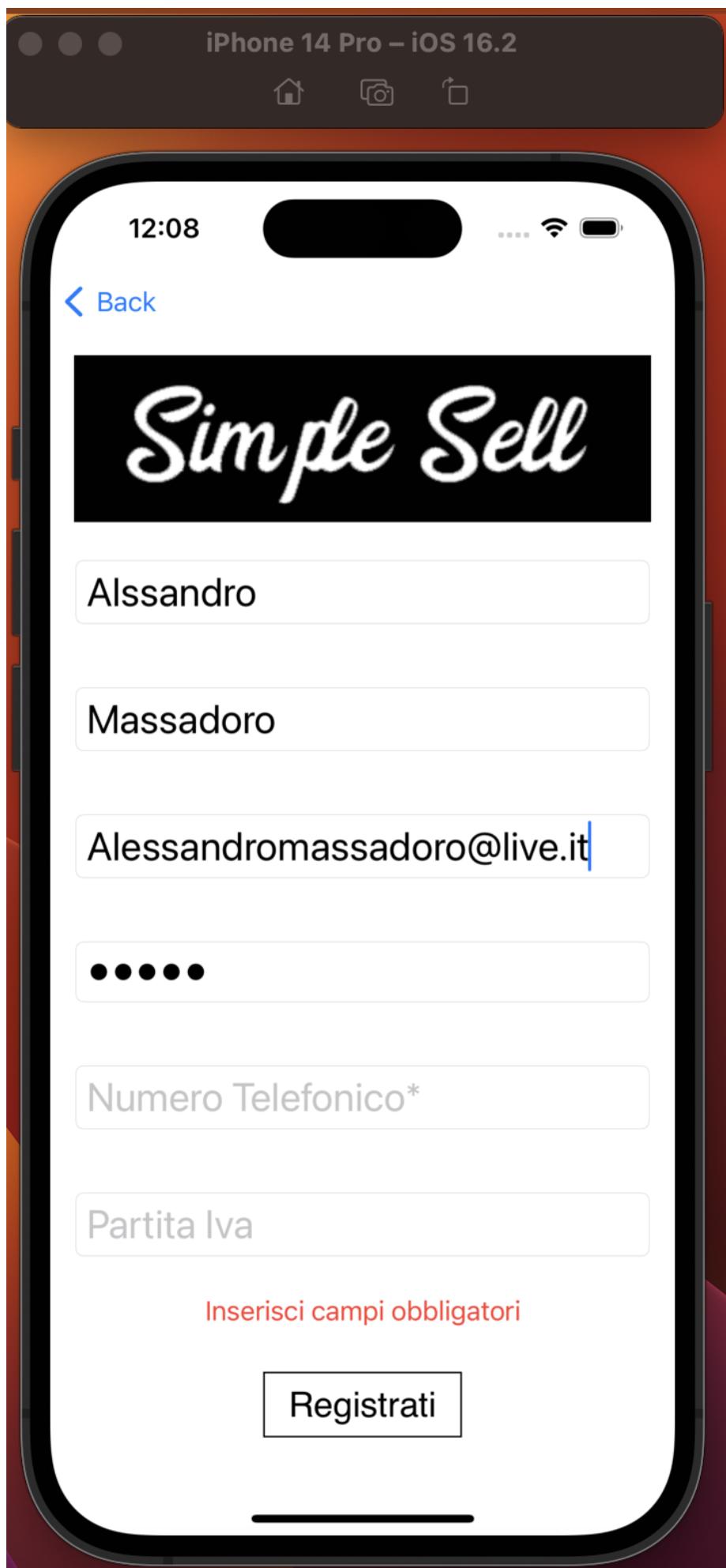


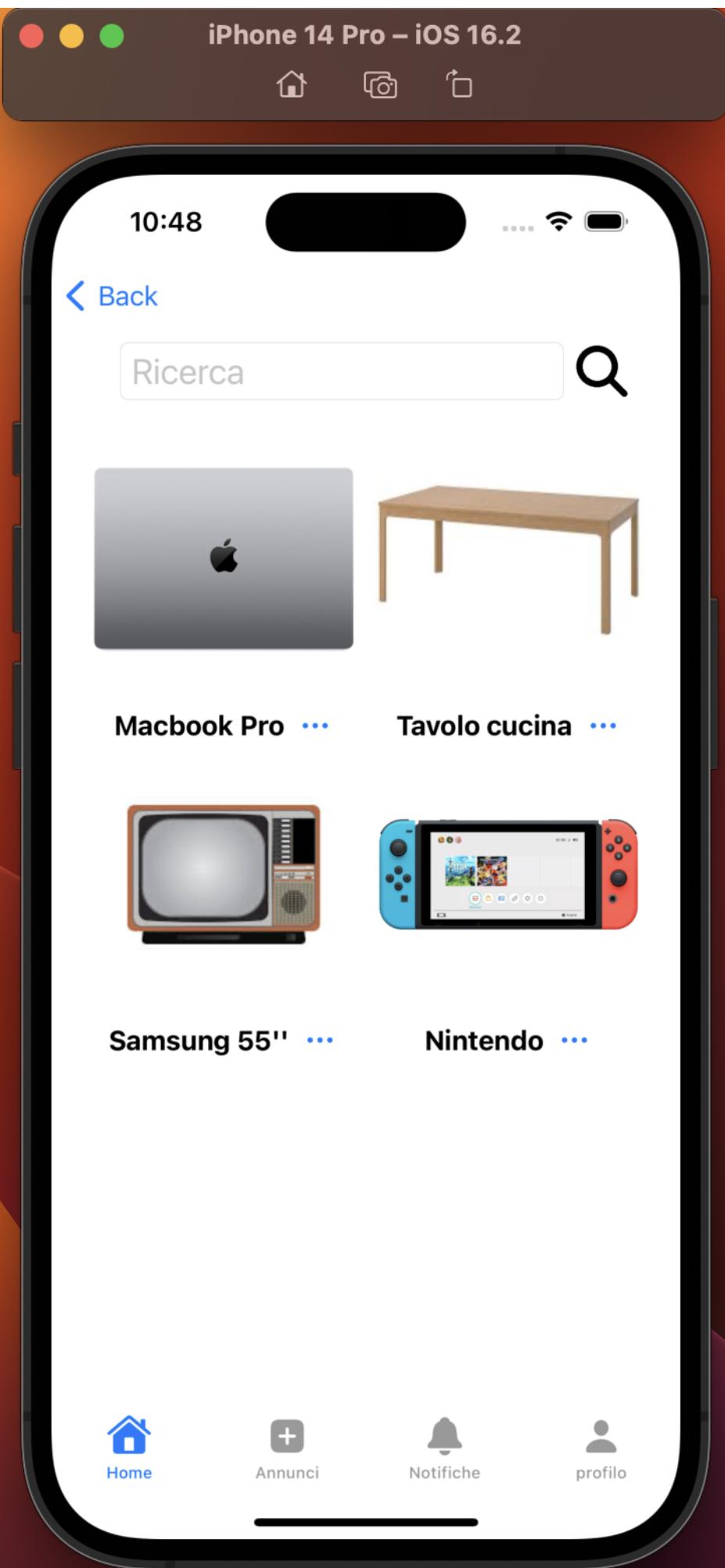


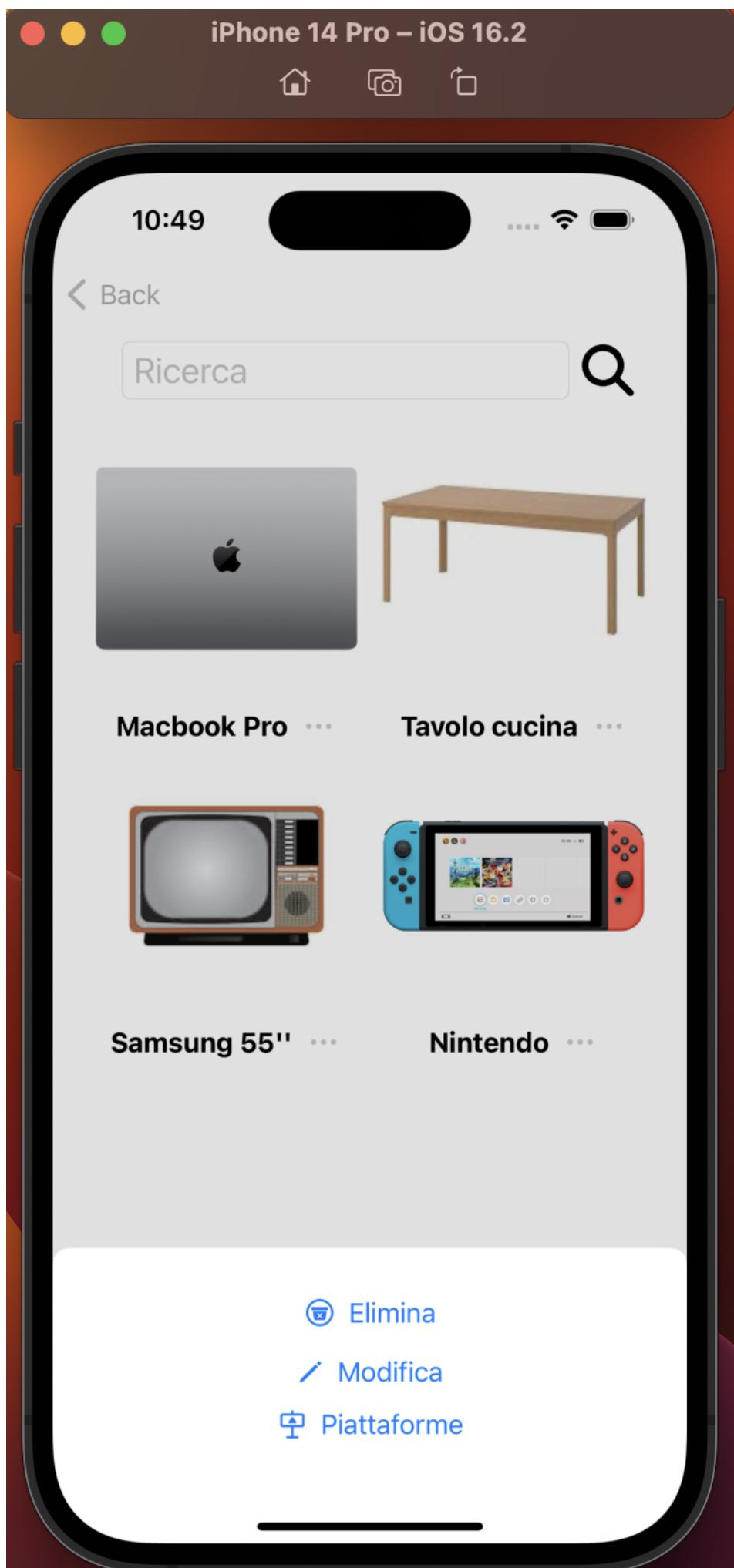
2.5.8.3 Mock-ups:









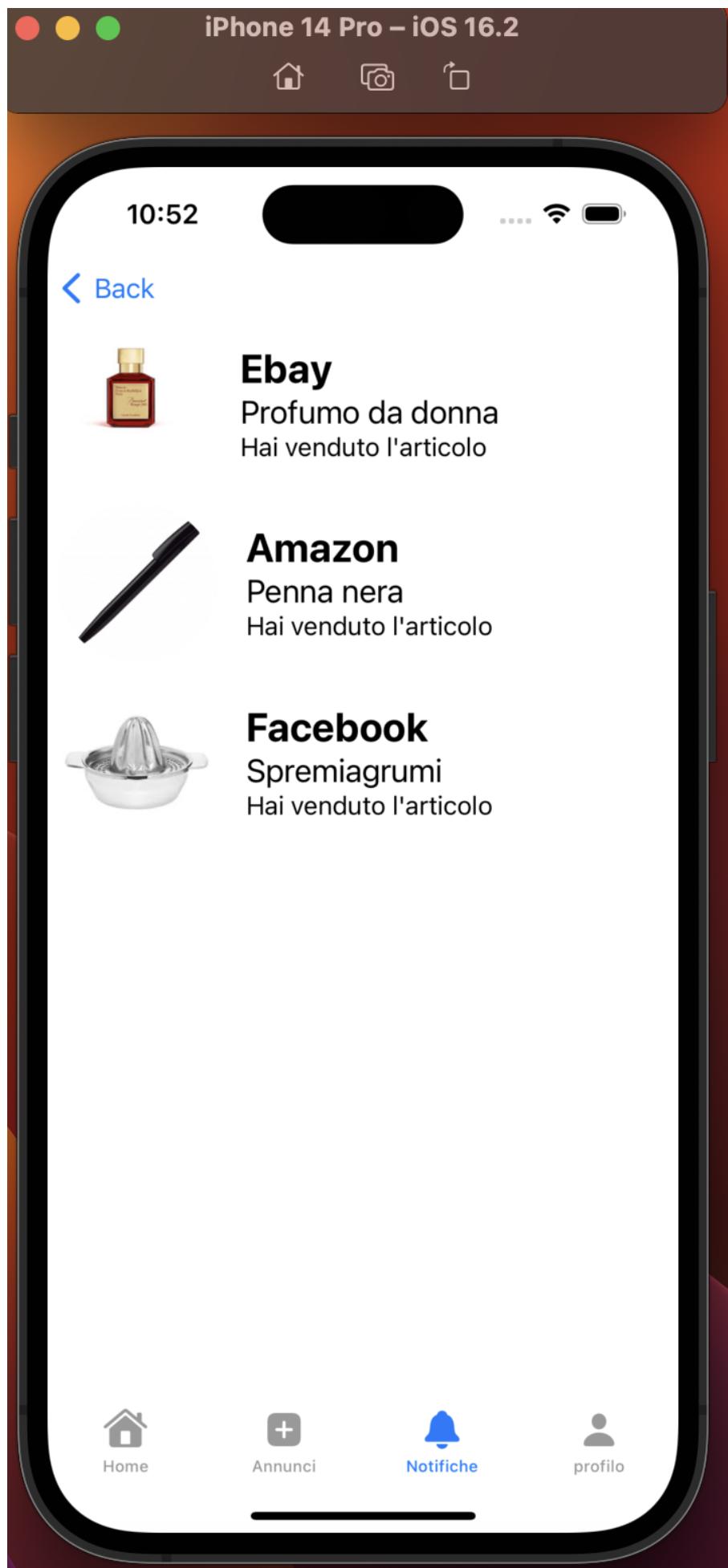


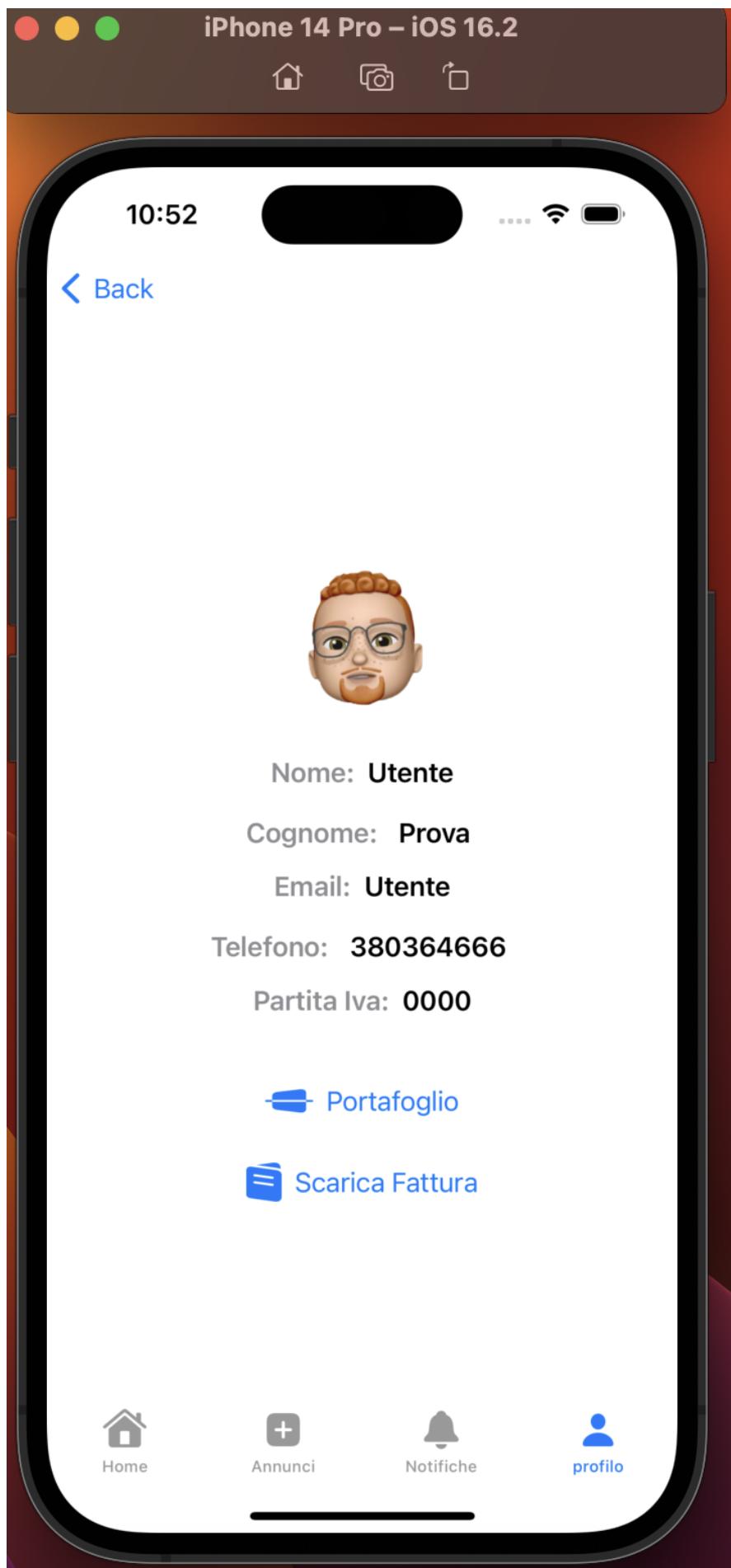


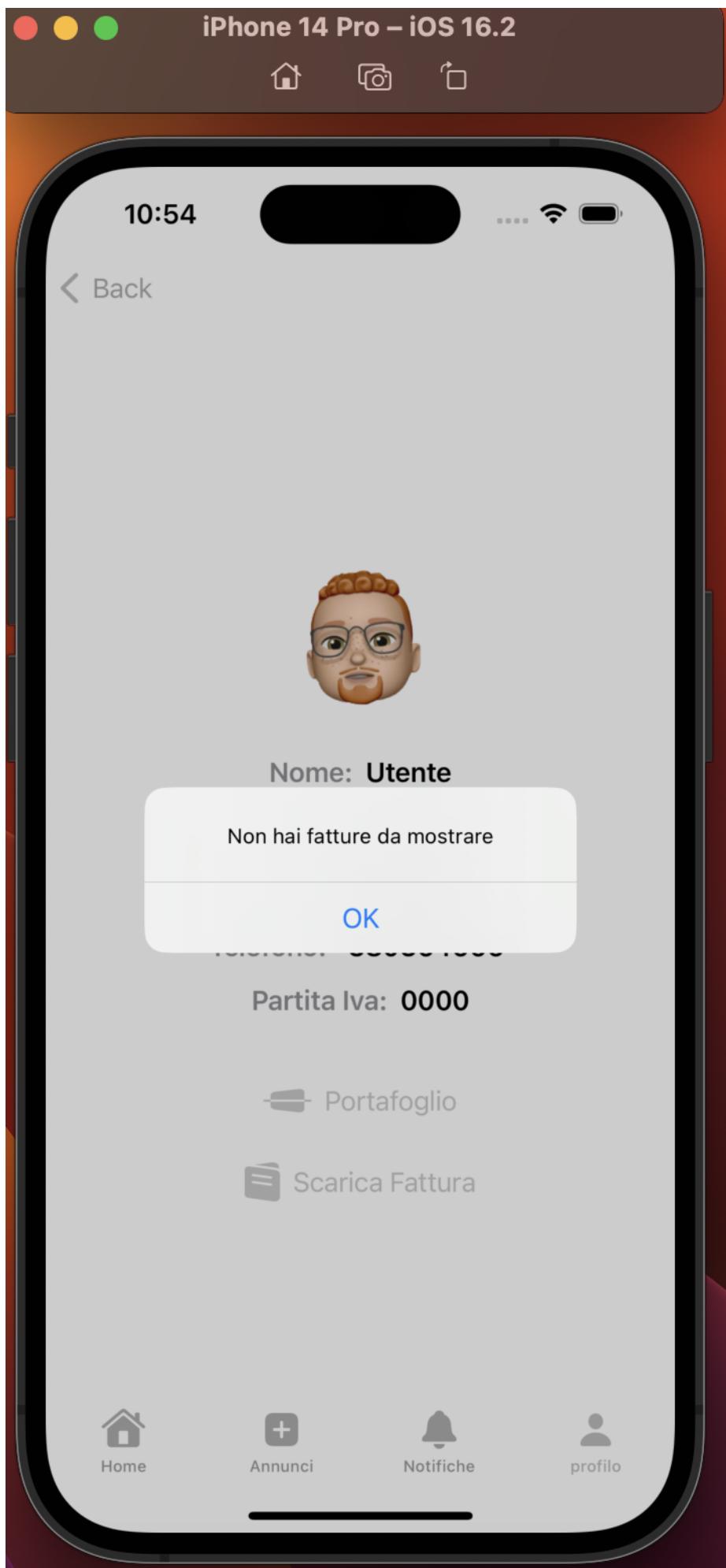












12.0

Valutazioni

Euristiche di usabilità

- **Visibilità dello stato del sistema:** L'home Page del sistema riporta la lista degli annunci presenti nel sistema e le operazioni che l'utente può compiere su di essi. Ogni operazione eseguita dall'utente gli restituisce sempre un feedback che lo informa degli effetti che questa ha avuto sul sistema. L'utente, inoltre, è sempre cosciente della sua posizione nell'app poiché questa viene segnalata nella barra di navigazione.
- **Corrispondenza fra il mondo reale e il sistema:** il sistema è rivolto ad un pubblico vario e ampio, non è stato quindi utilizzato un linguaggio specifico di un dominio.
- **Libertà e controllo da parte degli utenti:** l'utente può navigare nell'applicazione tramite un sistema di tab. Le azioni più importanti nel sistema richiedono la conferma dell'utente e quelle non reversibili sono segnalate.
- **Consistenza e standard:** l'interfaccia segue le linee guida per il design delle applicazioni mobile e presenta un'interfaccia simile a quelle delle principali applicazioni di compravendita, in modo da risultare familiare per gli utenti.
- **Prevenzione degli errori:** all'utente sono mostrate tutte e sole le azioni che è autorizzato a compiere. Per ogni campo di testo, inoltre, sono eseguiti gli opportuni controlli sul tipo di dati inseriti dall'utente e non è consentito inserire valori non idonei.
- **Riconoscere piuttosto che memorizzare:** l'utente non ha bisogno di memorizzare alcunché. Il sistema utilizza delle icone, piuttosto che delle scritte, al fine di indicare la funzionalità. Inoltre non ci sono opzioni "nascoste", ma sono tutte visibili all'utente.
- **Flessibilità ed efficienza d'uso:** data la semplicità del sistema non è stato valutato necessario inserire scorciatoie per l'utente esperto.
- Design minimalista ed estetico: le informazioni dei dialoghi non sono mai invadenti ed occupano lo spazio minimo necessario.
- **Aiutare gli utenti a riconoscere gli errori diagnosticarli e correggerli:** Anche se rari, i messaggi di errore nell'applicazione spiegano chiaramente quale sia il problema all'utente e indicano quali azioni eseguire per risolverlo
- **Guida e documentazione** In fase di progettazione è stato deciso che si punta tutto sull'usabilità del sistema, ciò significa che l'applicazione deve essere a tal punto intuitiva che non si necessita di una guida o documentazione di supporto all'utilizzo.

2.5.9. Testing

2.5.9.1 Obiettivo dei test:

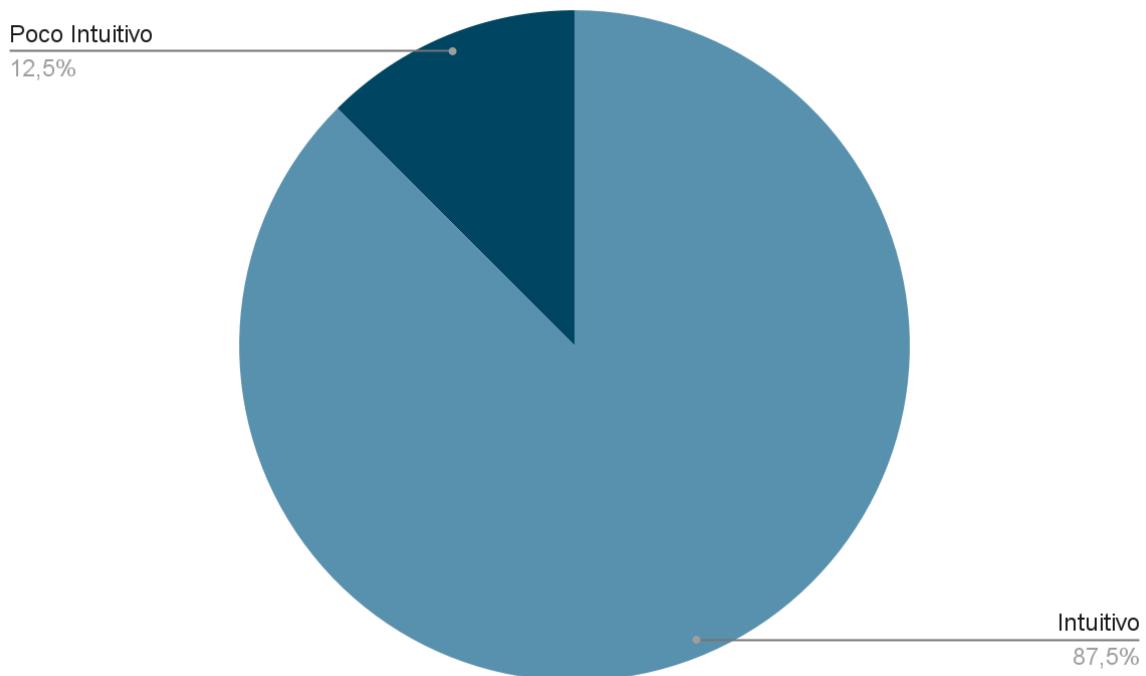
valutazione esperienza d'uso da parte dell'utente

2.5.9.2 Metodologia usata

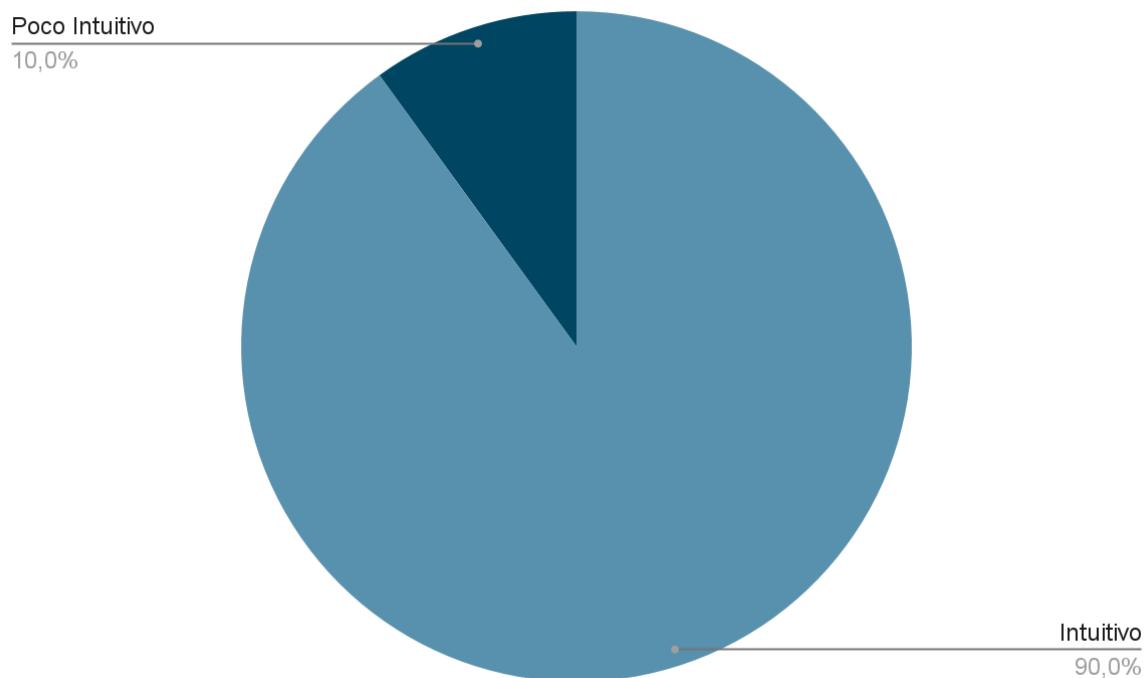
Per la metodologia di Testing abbiamo fatto testare la nostra applicazione a 5 persone, abbiamo assegnato ad ognuna di loro 5 task da eseguire e abbiamo chiesto di valutare la difficoltà di ognuna di esse. Ogni utente prima di procedere alle task è stato informato brevemente di cosa si occupasse l'applicazione.

- Sintesi delle misure:

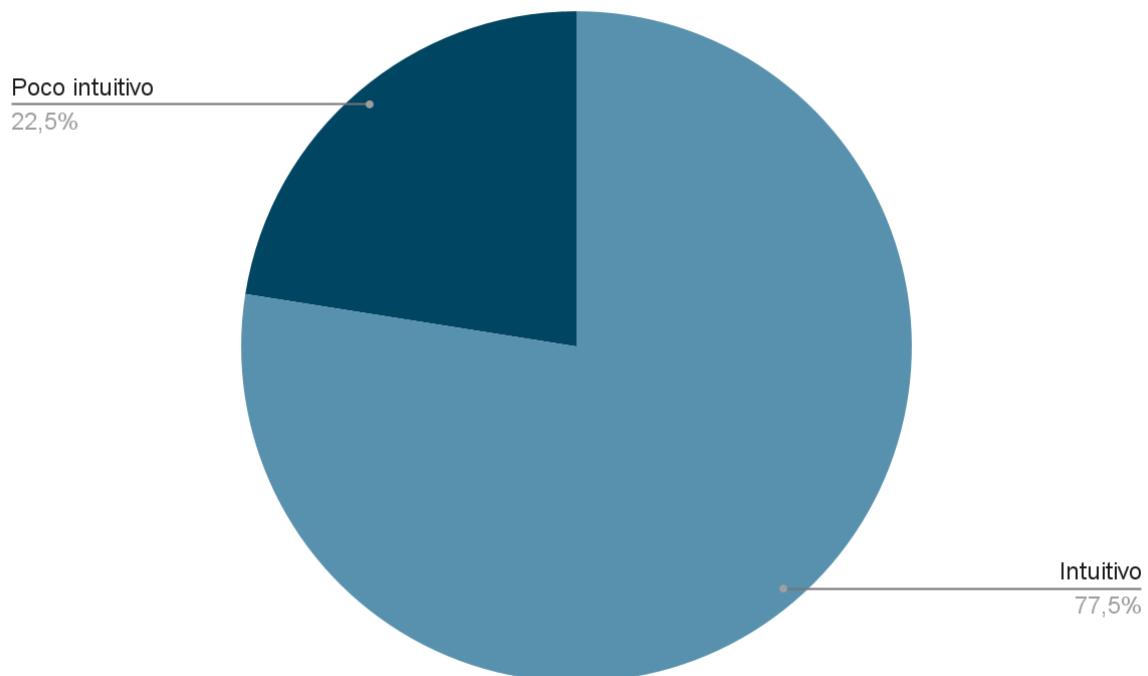
Prova a registrarti all'applicazione:



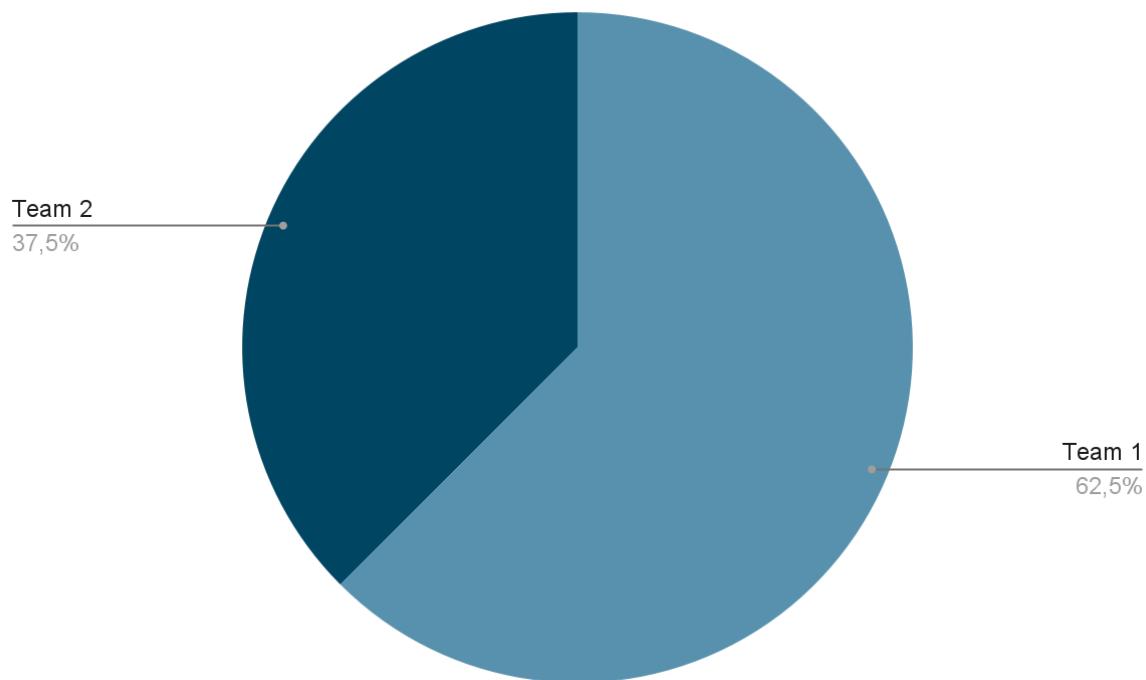
Inserisci un annuncio su ebay:



Prova a scaricare una fattura:



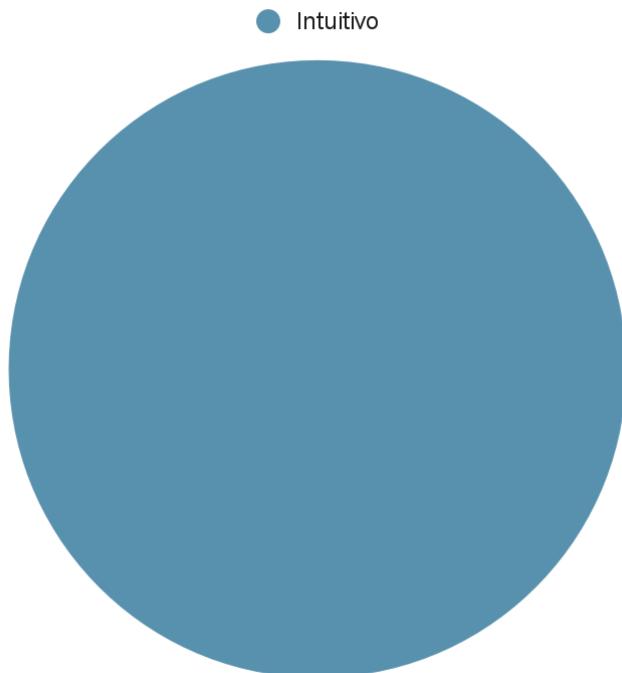
Elimina un annuncio:



L'eliminazione dell'annuncio nella fasi di testing è risultata poco intuitiva in confronto agli altri task assegnati, di fatti i tester si sarebbero aspettati di poter tener premuto sull'annuncio e dopo il tap poter eliminare.

Prendiamo quindi nota di questo feedback

Visualizza notifiche:



Al tester: Chiara Impagliazzo è stata chiesto una valutazione sulla grafica dell'interfaccia essendo quest'ultima in possesso di titoli di studio inerenti alla grafica design.

Chiara Impagliazzo:

Ciao, sono Chiara sono nata ad Ischia sono residente a Forio ho un diploma in Grafica Design conseguito presso l'Iliad di Napoli, mi è stato chiesto dal mio ragazzo Riccardo di esprimere pareri sulla grafica design dell'app in generale. Noto che l'interfaccia possiede una buona grafica design idonea ad una applicazione

mobile, il contrasto dei colori è buono e non da fastidio agli occhi, consiglio l'inserimento di una modalità notte.

2.5.9.3 Test di performance:

Tempi:

Apertura applicazione: 2 secondi

Registrazione al sistema: 0.5 secondi + TA

Login: 0.5 secondi + TA

Tempo restituzione dati errati: 0.5 secondi + TA

Tempo di accesso ad "Home": 0.5 secondi

Switch tra le varie viste: 0.2 secondi

Tempo di eliminazione: 0.3 secondi

Tempo d'inserimento: 0.3 secondi

TA: tempo di accesso al filesystem.

L'eliminazione e inserimento di un annuncio, vengono effettuati nelle varie piattaforme in maniera asincrona per tanto, il tempo di restituzione dati della nostra applicazione non corrispondono a quelli delle varie piattaforme. Quest'ultimo dipende dalle piattaforme stesse e non dalla nostra applicazione. I tempi sopra riportati dunque sono esclusivamente i tempi dei dati restituiti all'interno dell'applicazione e non contano quelli asincroni.

3.0 Analisi di mercato

Non risulta nessun competitor per tanto l'analisi di mercato non può essere approfondita l'unica affermazione è che tale prototipo risulta essere innovativo e non ancora sul mercato. Essendo l'applicazione non ancora sul mercato l'utenza potrebbe risultare alta, in quanto un'app intermediaria che consenta di vendere contemporaneamente su più piattaforme risulta essere richiesta, tale affermazione è

data dal fatto che l'idea nasce da un fabbisogno personale inoltre è stato chiesto attraverso un mini questionario ad amici che effettuano vendite tramite applicazioni se scaricherebbero la nostra app e la risposta è stata positiva.

4.0 Future Implementazioni

Nelle future implementazione oltre all'implementazione completa del prototipo si implemeterà un diverso approccio per l'eliminazione dell'annuncio oltre che consentire l'eliminazione dal menù come già implementato, permetteremo l'eliminazione dell'annuncio dopo una pressione prolungata sull'annuncio, tramite una (x) come emerso dalla fase di testing.

Inoltre è emerso che la scritta annunci sotto l'icona (+) va sostituita con la scritta vendi ciò aumenta l'usabilità ed evita qualsiasi incomprensione.

5.0 Glossario

@State: La variabile @state è una variabile utilizzata in swift per poter interagire con gli elementi di una view.

Api: (Application Programming Interfaces) sono interfacce di programmazione per le applicazioni che consentono a due sistemi software di comunicare tra loro. In pratica, le API forniscono un modo per accedere ai dati e alle funzionalità di un sistema da parte di altri sistemi o applicazioni. Questo permette agli sviluppatori di integrare funzionalità esterne nei loro prodotti, creare nuovi servizi e migliorare l'efficienza del lavoro.

Piattaforma: sistema software atto alla vendita online

Annunci: post inseriti dagli utenti che vogliono vendere i propri prodotti online

Fattura: documento commerciale che rappresenta una transazione tra due parti, in questo caso rilasciata agli utenti che effettuano vendite con partita iva

File system: utilizziamo un sistema di file e directory per memorizzare i dati critici del nostro sistema