

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 TRIENNALE IN INFORMATICA

PROGETTO D'ESAME DI OBJECT ORIENTATION

**PROGETTAZIONE E SVILUPPO DI UN APPLICATIVO IN JAVA
PER UNA PIATTAFORMA DI COMPRACENDITA TRA
STUDENTI UNIVERSITARI**

Relatore

Professore Sergio Di Martino

Candidati

Buffardi Mario N86004704
ma.buffardi@studenti.unina.it

Cinquegrana Francesco N86004258
frances.cinquegrana@studenti.unina.it

Sommario

1	Presentazione del progetto	3
1.1	UninaSwap	3
1.2	Un primo sguardo	3
2	Progettazione	3
2.1	Architettura	3
2.2	Organizzazione dei Package	3
2.2.1	Model (it.unina.uninaswap.model)	3
2.2.2	DAO Layer (it.unina.uninaswap.dao)	3
2.2.3	View (it.unina.uninaswap.view)	3
2.2.4	Controller (it.unina.uninaswap.controller)	4
2.2.5	Utilities (it.unina.uninaswap.util)	4
2.3	Flusso di Esecuzione (Esempio Login)	4
2.4	Class Diagram	5
2.4.1	ClassDiagram Entità	5
2.4.2	ClassDiagram EntityDao	6
2.4.3	ClassDiagram Controller	7
3	Sequence Diagram	8
3.1	Sequence Diagram per la creazione di un annuncio	8
3.2	Sequence Diagram per fare un offerta di acquisto ad una vendita partendo dalla home	9
4	MockUp	10
4.1	Schermata di Accesso e Registrazione (LoginView)	10
4.2	Dashboard Principale (AnnunciMainView)	10
4.3	Dettaglio Annuncio (AnnuncioDetailView)	12
4.4	Area Personale (ProfileView)	13
4.5	Centro Notifiche (NotificationView)	14
4.6	Creazione Annuncio (AnnuncioCreateDialog)	14
4.7	Visualizzazione delle Statistiche e Report	15

1 *Presentazione del progetto*

1.1 *UninaSwap*

UninaSwap è una piattaforma progettata per la compravendita, lo scambio e il regalo di oggetti tra studenti universitari. L'applicativo permette agli studenti di registrarsi tramite le credenziali universitarie. Gli utenti possono pubblicare annunci relativi a oggetti personali, offrire o accettare offerte e completare transazioni che includono anche la gestione delle consegne e la possibilità di lasciare recensioni.

1.2 *Un primo sguardo*

Un utente, dopo la registrazione o l'accesso con le proprie credenziali, può visualizzare e cercare annunci, effettuare acquisti e mandare proposte. Può inoltre creare un proprio annuncio, specificando un titolo, la tipologia dell'annuncio, la categoria dell'oggetto e in caso di un annuncio di vendita, il prezzo. Può inoltre scrivere una descrizione e caricare foto dell'oggetto che si sta caricando sulla piattaforma selezionandone una come foto principale, che sarà poi visualizzata nella schermata degli annunci.

2 *Progettazione*

2.1 *Architettura*

Il sistema UninaSwap è un applicativo desktop sviluppato in Java e gestito tramite Maven. L'architettura è stata progettata per garantire modularità e manutenibilità, seguendo il pattern architetturale MVC (Model-View-Controller), che separa nettamente la logica di business, l'interfaccia utente e la gestione dei dati.

Il sistema si appoggia a un database relazionale PostgreSQL per la persistenza dei dati e utilizza la libreria Swing (arricchita dalla libreria FlatLaf) per la realizzazione dell'interfaccia grafica.

2.2 *Organizzazione dei Package*

Il codice è strutturato in package distinti in base alle responsabilità, facilitando lo sviluppo e la manutenzione:

2.2.1 *Model (it.unina.uninaswap.model)*

- Entity: Contiene le classi, che mappano le tabelle del database (es. Studente, Annuncio, Indirizzo, Offerta). Queste classi rappresentano il dominio del problema.
- Enums: Gestisce le enumerazioni per garantire la consistenza dei dati (es. SessoStudente, StatoOfferta, TipoAnnuncio).

2.2.2 *DAO Layer (it.unina.uninaswap.dao)*

Questo livello astrae l'accesso al database, nascondendo i dettagli delle query SQL al resto dell'applicazione.

- **Interfacce DAO:** Definiscono i metodi CRUD e di ricerca (es. StudenteDAO, AnnuncioDAO).
- **Implementazioni (impl):** Contengono la logica effettiva di accesso ai dati tramite JDBC (es. StudenteDAOImpl). Utilizzano PreparedStatement per sicurezza ed efficienza.
- **DBConfig:** La classe DBConnection gestisce la connessione centralizzata al database PostgreSQL tramite il pattern Singleton/Static factory.

2.2.3 *View (it.unina.uninaswap.view)*

Gestisce l'interfaccia grafica e l'interazione con l'utente.

- Le classi estendono componenti Swing (come JFrame o JPanel) e definiscono il layout e gli elementi visivi (es. LoginView, AnnunciMainView).
- La View non contiene logica di business, ma espone metodi per aggiungere *Listeners* che vengono gestiti dai Controller.

2.2.4 *Controller (it.unina.uninaswap.controller)*

Agisce da intermediario tra la View e il Model.

- Gestisce gli eventi generati dall'utente (click sui pulsanti, input).
- Invoca i metodi del DAO per recuperare o salvare dati.
- Aggiorna la View in base ai risultati.
- Esempio: LoginController ascolta il click sul tasto "Accedi", verifica le credenziali tramite StudenteDAO e, in caso di successo, istanzia la HomeController e apre la dashboard principale.

2.2.5 *Utilities (it.unina.uninaswap.util)*

Contiene classi di supporto trasversali all'applicazione, come UITheme per la gestione costante dei colori e dei font, o ImageUtil e ChartUtil (che sfrutta *JFreeChart*) per la manipolazione di immagini e grafici.

2.3 *Flusso di Esecuzione (Esempio Login)*

1. L'App avvia la LoginView e il LoginController.
2. L'Utente inserisce i dati e preme "Accedi".
3. Il Controller cattura l'evento, valida l'input e chiama `studenteDAO.findByEmailAndPassword(...)`.
4. Il DAO esegue la query SQL sul Database e restituisce un oggetto `Studente`.
5. Il Controller, ricevuto l'oggetto, chiude la finestra di login e apre la `AnnunciMainView`.

Questa struttura garantisce che ogni componente abbia una singola responsabilità ben definita, rendendo il codice pulito ed estensibile.

2.4.1 ClassDiagram Entità

2.4.2 ClassDiagram EntityDao

① ↻ OffertaDAO

```
Ⓜ ↻ countAcquistiAccettatiPerTipologia(String, TipoAnnuncio) int
Ⓜ ↻ countAcquistiAccettatiPerCategoria(String, TipoCategoria) int
Ⓜ ↻ findInviatePerOfferenteByStato(String, String) List<Offerta>
Ⓜ ↻ rifiutaAltreOfferteStessoAnnuncio(int, int) void
Ⓜ ↻ countOfferteArrivatePerCategoria(String, TipoCategoria) int
Ⓜ ↻ countOfferteInviatePerCategoria(String, TipoCategoria) int
Ⓜ ↻ countOfferteArrivatePerTipologia(String, TipoAnnuncio) int
Ⓜ ↻ insert(Offerta) void
Ⓜ ↻ countVenditeAccettatePerCategoria(String, TipoCategoria) int
Ⓜ ↻ countVenditeAccettatePerTipologia(String, TipoAnnuncio) int
Ⓜ ↻ updateContenuto(Offerta) void
Ⓜ ↻ updateStato(int, String) void
Ⓜ ↻ findRicevuteInAttesaPerVenditore(String) List<Offerta>
Ⓜ ↻ findInviateInAttesaPerOfferente(String) List<Offerta>
Ⓜ ↻ findRicevutePerVenditoreByStato(String, String) List<Offerta>
Ⓜ ↻ findById(int) Offerta
Ⓜ ↻ countOfferteInviatePerTipologia(String, TipoAnnuncio) int
```

① ↻ AnnuncioDAO

```
Ⓜ ↻ setConcluso(int, boolean) void
Ⓜ ↻ insert(Annuncio) void
Ⓜ ↻ findById(int) Annuncio
Ⓜ ↻ findUltimiByStudente(String, int) List<Annuncio>
Ⓜ ↻ update(Annuncio) void
Ⓜ ↻ delete(int, String) boolean
Ⓜ ↻ findAttivi(String, String, String, BigDecimal, BigDecimal, Boolean, Boolean) List<Annuncio>
Ⓜ ↻ findAttiviEsclusoStudente(String, String, String, String, BigDecimal, BigDecimal) List<Annuncio>
Ⓜ ↻ findByStudente(String) List<Annuncio>
```

① ↻ TransazioneDAO

```
Ⓜ ↻ getMediaImportoVenditaAccettataPerAcquirente(String) BigDecimal
Ⓜ ↻ findById(int) Transazione
Ⓜ ↻ findByOffertaAccettata(int) Transazione
Ⓜ ↻ getMinImportoVenditaAccettataPerAcquirente(String) BigDecimal
Ⓜ ↻ getMinImportoVenditaAccettataPerVenditore(String) BigDecimal
Ⓜ ↻ getMediaImportoVenditaAccettataPerVenditore(String) BigDecimal
Ⓜ ↻ getMaxImportoVenditaAccettataPerAcquirente(String) BigDecimal
Ⓜ ↻ getMaxImportoVenditaAccettataPerVenditore(String) BigDecimal
```

① ↻ StudenteDAO

```
Ⓜ ↻ findByEmailAndPassword(String, String) Studente
Ⓜ ↻ update(Studente) void
Ⓜ ↻ findByMatricola(String) Studente
Ⓜ ↻ insert(Studente) void
Ⓜ ↻ insert(Studente, Connection) void
Ⓜ ↻ existsByMatricola(String) boolean
Ⓜ ↻ existsByEmail(String) boolean
```

① ↻ ConsegnaDAO

```
Ⓜ ↻ findByTransazione(int) Consegna
Ⓜ ↻ insert(Consegna) void
Ⓜ ↻ creaConsegnaSpedizione(int, int, String) void
Ⓜ ↻ creaConsegnaInUni(int, LocalDate, String, String, String) void
```

① ↻ FotoDAO

```
Ⓜ ↻ findPrincipaleOrFirstByAnnuncio(int) Foto
Ⓜ ↻ findByAnnuncio(int) List<Foto>
Ⓜ ↻ deleteByAnnuncio(int) void
Ⓜ ↻ insert(Foto) void
```

① ↻ RecensioneDAO

```
Ⓜ ↻ insert(Recensione) void
Ⓜ ↻ findRicevuteByStudente(String) List<Recensione>
Ⓜ ↻ existsByTransazioneAndAutore(int, String) boolean
```

① ↻ IndirizzoDAO

```
Ⓜ ↻ insert(Indirizzo, Connection) void
Ⓜ ↻ insert(Indirizzo) void
Ⓜ ↻ findByStudente(String) List<Indirizzo>
```

2.4.3 ClassDiagram Controller

Ⓢ ↗ SettingsController
Ⓜ ↗ openSettingsDialog() void

Ⓢ ↗ LoginController
Ⓜ ↗ handleLogin() void
Ⓜ ↗ handleRegistration() void
Ⓜ ↗ validateRegistrationInput() boolean

Ⓢ ↗ ProfiloController
Ⓜ ↗ refreshProfileView() void
Ⓜ ↗ handleBackFromVenditoreProfile() void
Ⓜ ↗ refreshVenditoreProfileView(Studente) void

Ⓢ ↗ OfferteController
Ⓜ ↗ handleLasciaRecensione(Offerta) void
Ⓜ ↗ handleAcceptOffer(Offerta) void
Ⓜ ↗ buildNewOffertaForAnnuncio(Annuncio) Offerta
Ⓜ ↗ rifiutaOrRitiraOfferta(Offerta) void
Ⓜ ↗ handleEditOfferta(Offerta) void
Ⓜ ↗ handleAnnuncioOffertaAction(Annuncio, String) void

Ⓢ ↗ NotificationController
Ⓜ ↗ refreshNotificationView() void
Ⓜ ↗ openAnnuncioFromOfferta(Offerta) void
Ⓜ ↗ canLasciaRecensione(Offerta) boolean

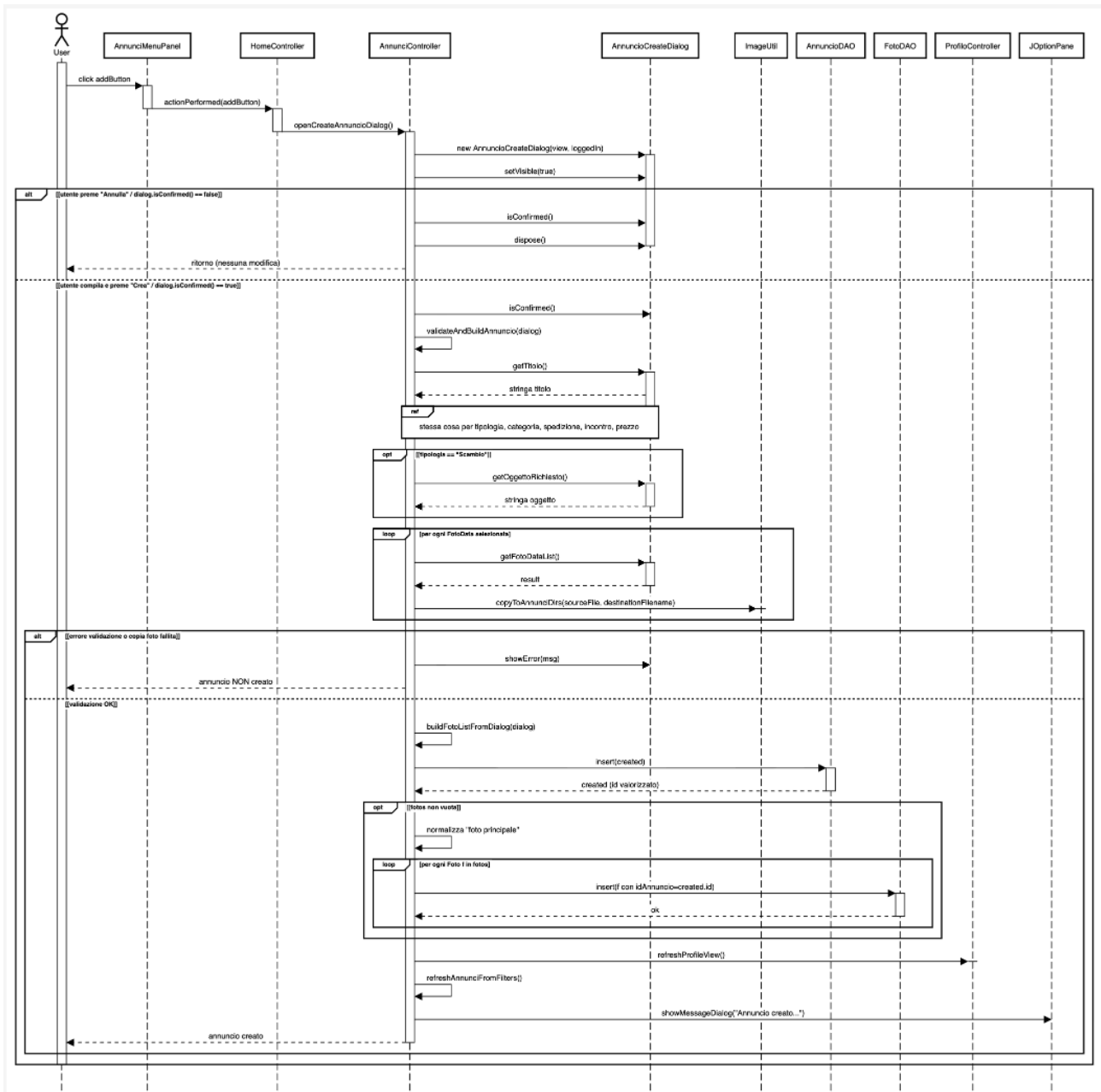
Ⓢ ↗ ReportController
Ⓜ ↗ refreshReportView() void

Ⓢ ↗ AnnunciController
Ⓜ ↗ openAnnuncioDetailFromProfile(Annuncio) void
Ⓜ ↗ openAnnuncioDetailFromHome(Annuncio) void
Ⓜ ↗ openAnnuncioDetailCommon(Annuncio) boolean
Ⓜ ↗ parseBigDecimalOrNull(String) BigDecimal
Ⓜ ↗ refreshAnnunciFromFilters() void
Ⓜ ↗ configureBackToProfile() void
Ⓜ ↗ configureBackToVenditoreProfile() void
Ⓜ ↗ deleteAnnuncio(Annuncio) void
Ⓜ ↗ configureBackToHome() void
Ⓜ ↗ openAnnuncioEdit(Annuncio) void
Ⓜ ↗ openAnnuncioDetailFromVenditoreProfile(Annuncio) void
Ⓜ ↗ configureBackToNotification() void
Ⓜ ↗ openCreateAnnuncioDialog() void

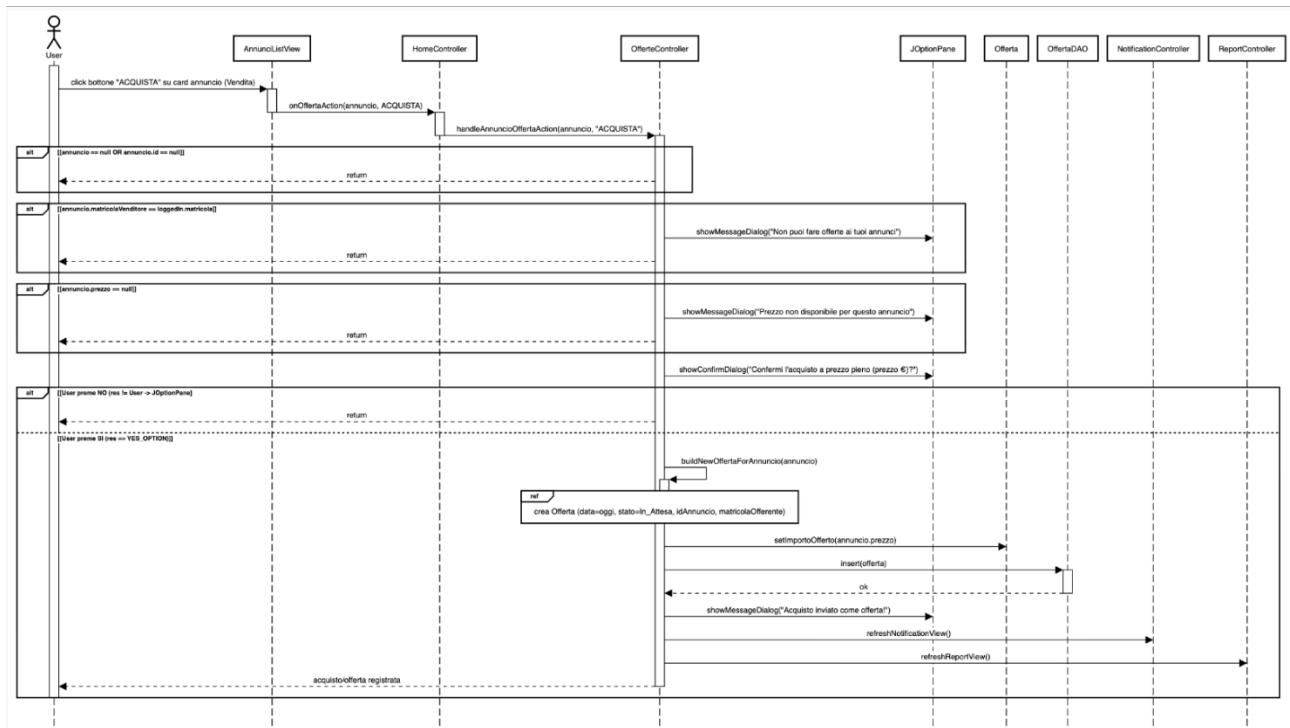
Ⓢ ↗ HomeController
Ⓜ ↗ initDefaultState() void
Ⓜ ↗ setLastOpenedAnnuncio(Annuncio) void
Ⓜ ↗ setLastOpenedVenditore(Studente) void
Ⓜ ↗ getOfferteController() OfferteController
Ⓜ ↗ getCurrentDetailOrigin() DetailOrigin
Ⓜ ↗ getAnnunciController() AnnunciController
Ⓜ ↗ getReportController() ReportController
Ⓜ ↗ setCurrentDetailOrigin(DetailOrigin) void
Ⓜ ↗ getLastOpenedVenditore() Studente
Ⓜ ↗ getAnnuncioIdBeforeVenditoreProfile() Integer
Ⓜ ↗ getDetailOriginBeforeVenditoreProfile() DetailOrigin
Ⓜ ↗ getProfiloController() ProfiloController
Ⓜ ↗ setAnnuncioIdBeforeVenditoreProfile(Integer) void
Ⓜ ↗ getNotificationController() NotificationController
Ⓜ ↗ getLastOpenedAnnuncio() Annuncio
Ⓜ ↗ setDetailOriginBeforeVenditoreProfile(DetailOrigin) void
Ⓜ ↗ initListeners() void
Ⓜ ↗ handleLogout() void

3 Sequence Diagram

3.1 Sequence Diagram per la creazione di un annuncio



3.2 Sequence Diagram per fare un offerta di acquisto ad una vendita partendo dalla home

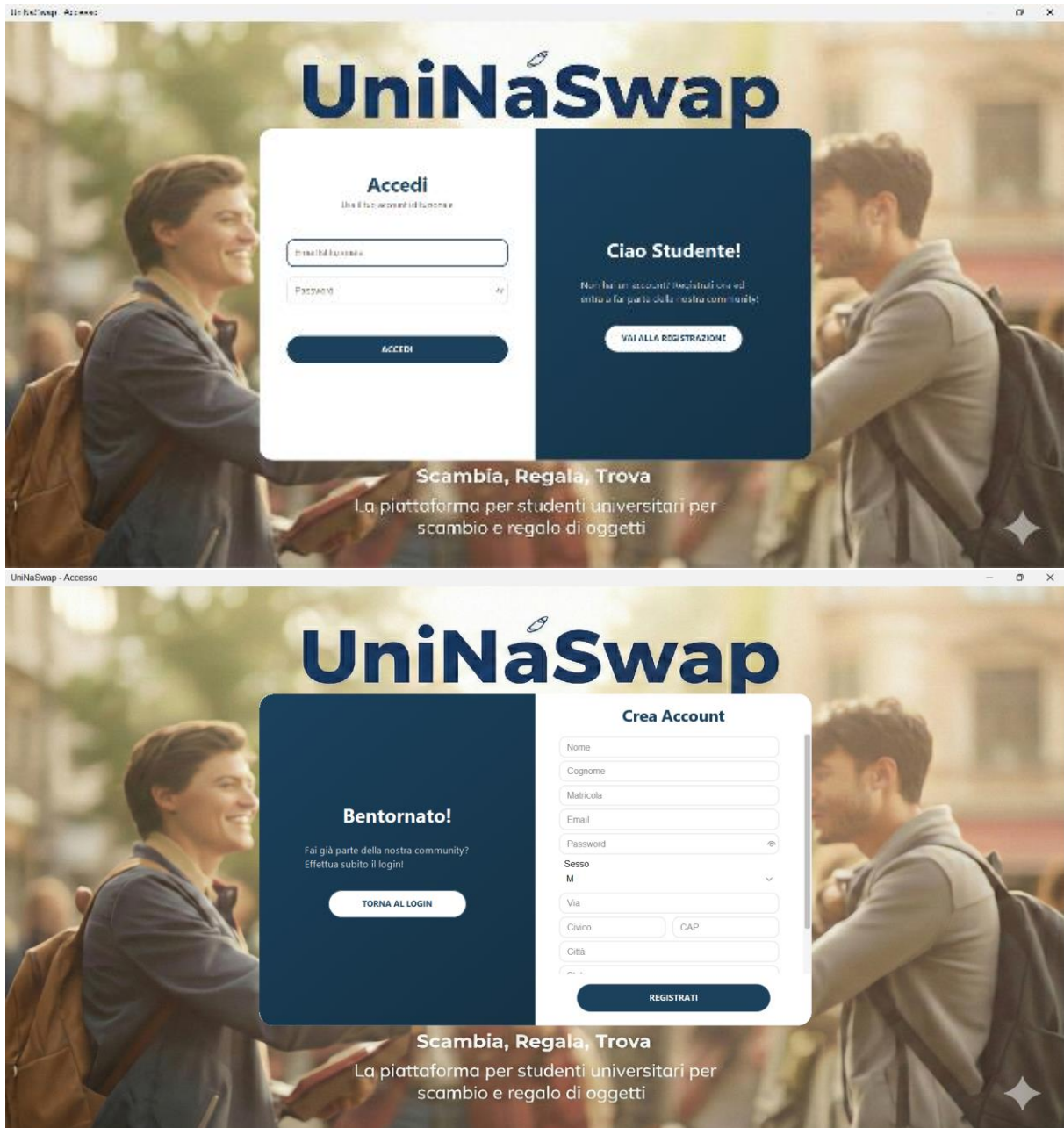


4 MockUp

Il flusso dell'applicazione è progettato per essere intuitivo, guidando lo studente dall'accesso alla compravendita di materiale didattico. Le schermate principali sono:

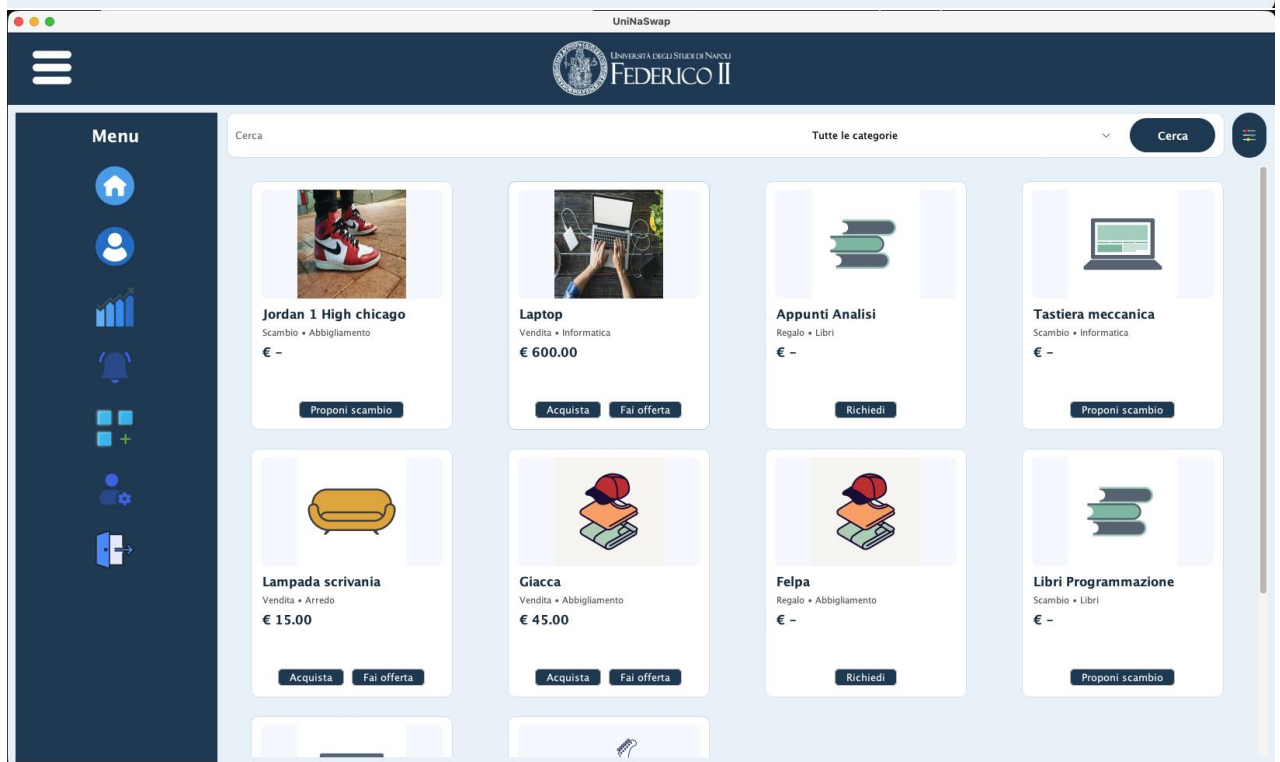
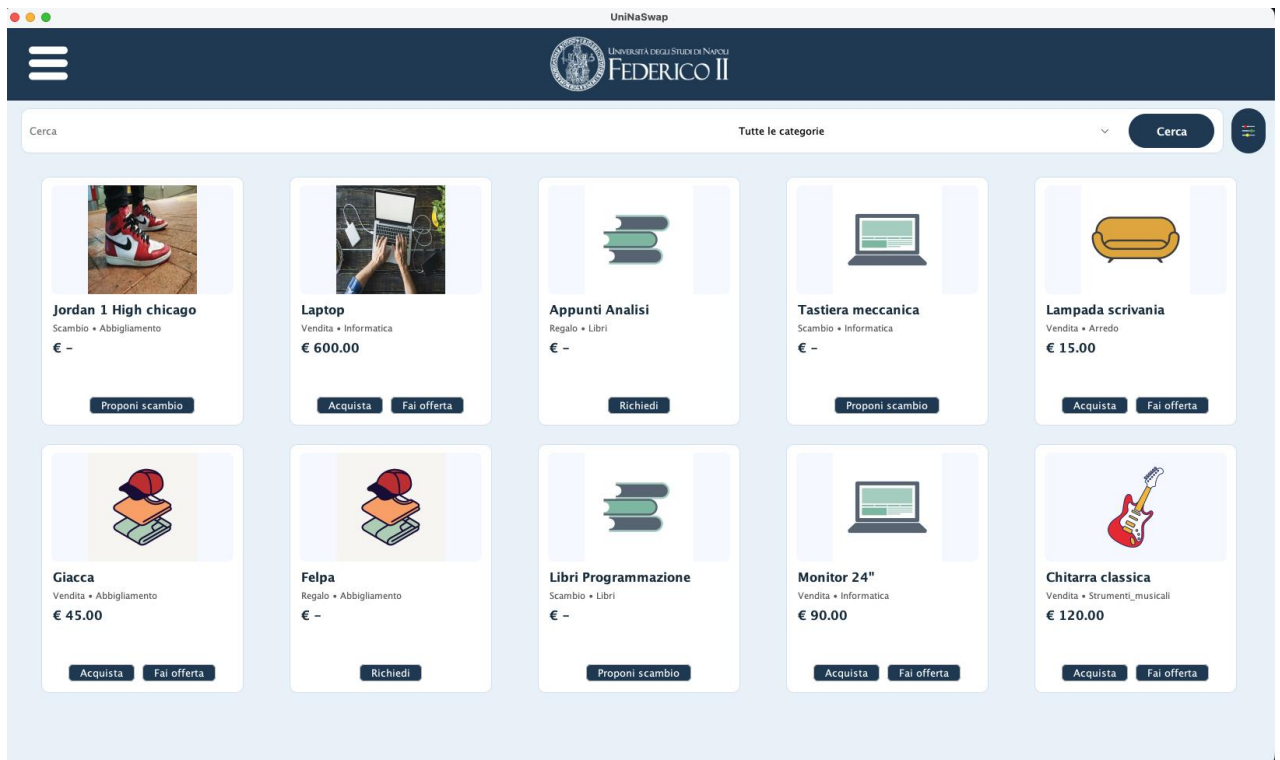
4.1 Schermata di Accesso e Registrazione (LoginView)

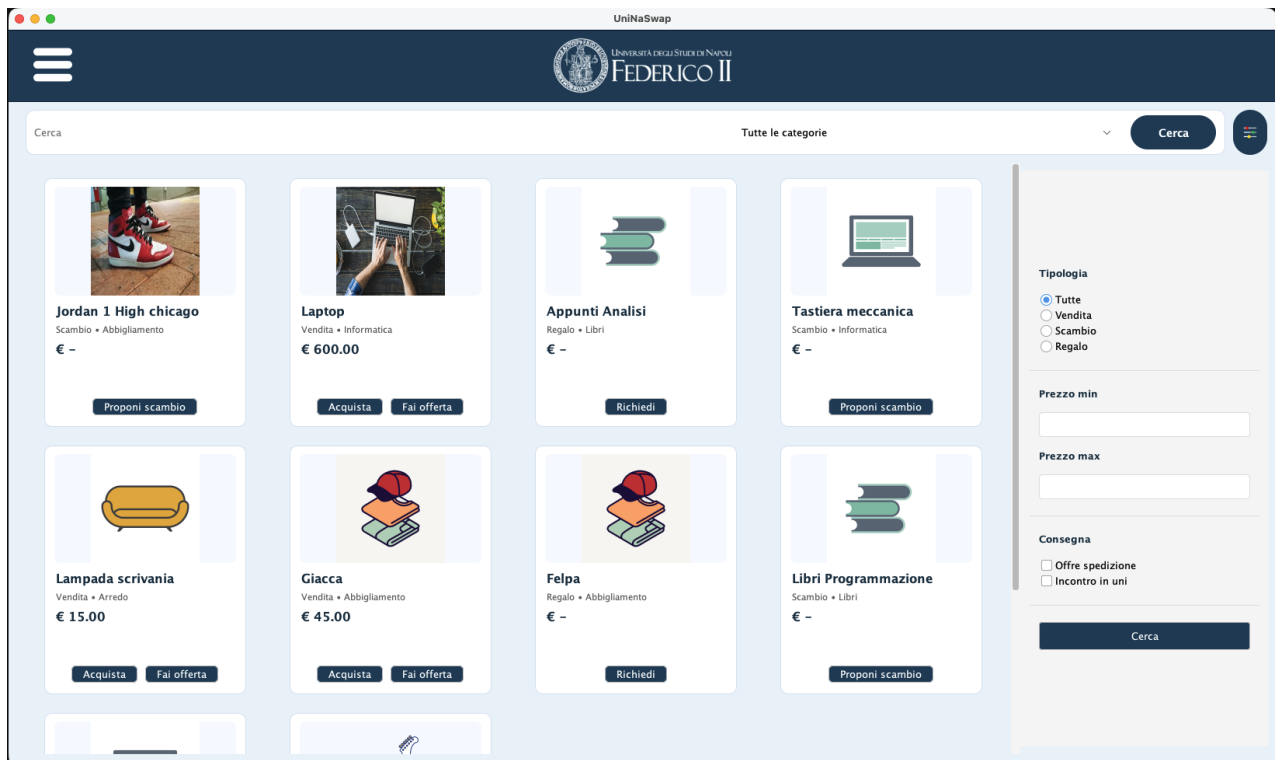
Questa è la finestra di ingresso all'applicativo. È caratterizzata da un design a pannelli scorrevoli (*Overlay Animation*) che permette di passare dalla fase di login a quella di registrazione senza cambiare finestra.



4.2 Dashboard Principale (AnnunciMainView)

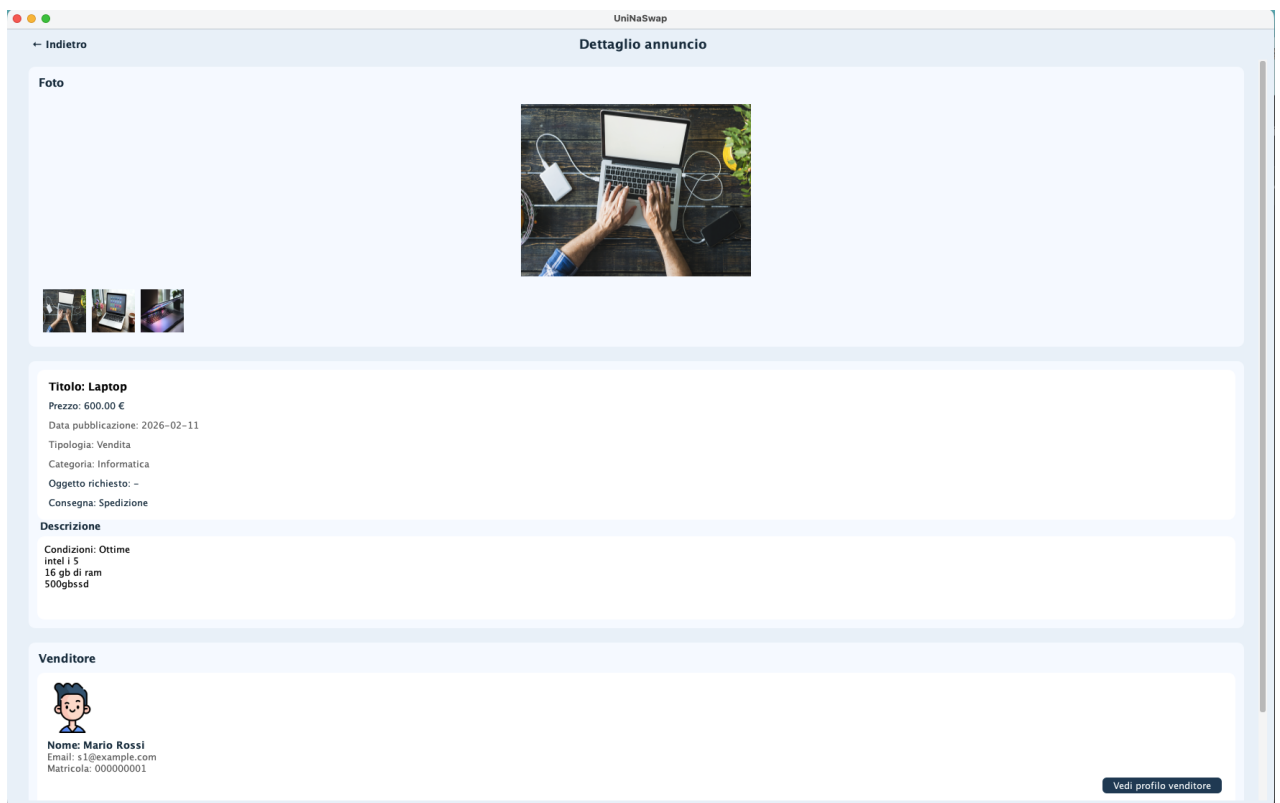
Una volta autenticato, l'utente accede alla Dashboard. Questa finestra funge da homepage. In homepage è possibile cercare annunci tramite l'apposita barra di ricerca, utilizzando il menu dei filtri. È possibile anche accedere al menù laterale per navigare tra le varie pagine.





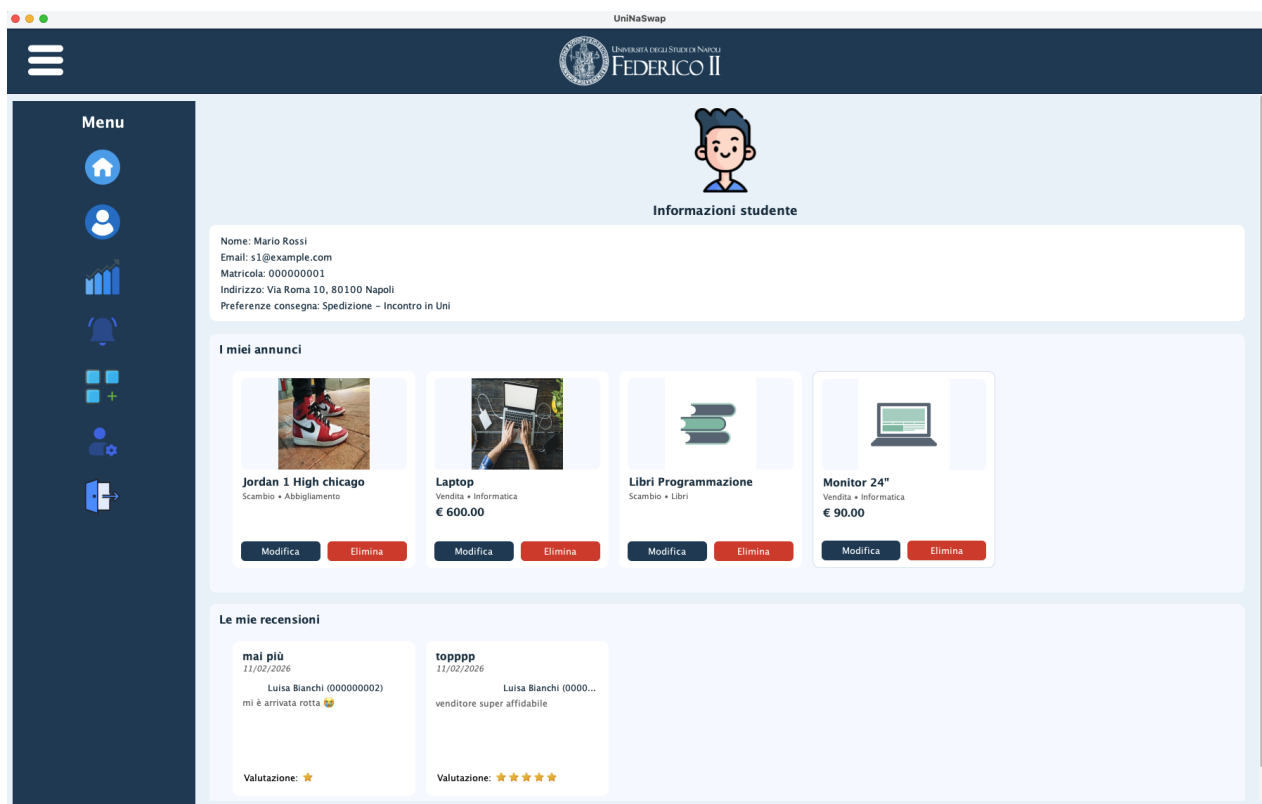
4.3 *Dettaglio Annuncio (AnnuncioDetailView)*

Cliccando su un annuncio nella bacheca, si apre la vista di dettaglio, dove è possibile controllare la descrizione e tutte le informazioni relative all'annuncio.



4.4 Area Personale (ProfileView)

Questa sezione è dedicata alla gestione dell'account studente, con la possibilità di modificare o eliminare gli annunci e controllare le recensioni ricevute.



Impostazioni profilo

Matricola:

00000001

Nome:

Mario

Cognome:

Rossi

Email:

s1@example.com

Password:

•••••

Sesso:

M

Preferenze consegna

☒ Spedizione ☒ Incontro in uni

Indirizzo (per spedizione)

Via:

Civico:

CAP:

Città:

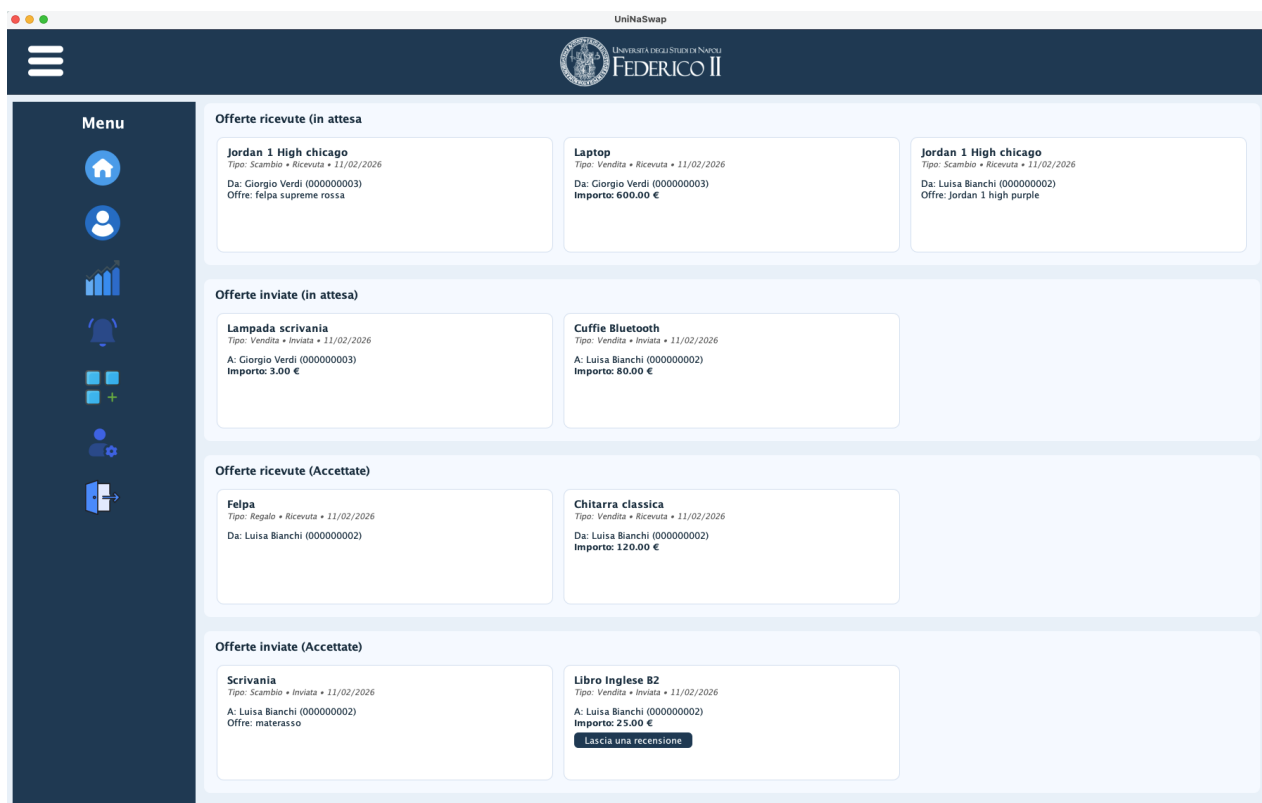
Stato:

Annulla

Salva

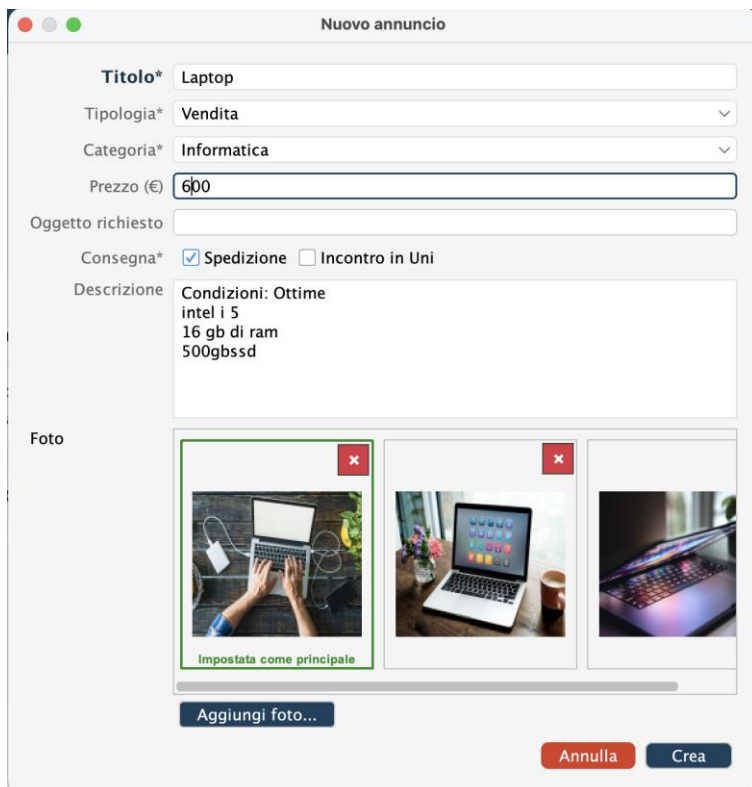
4.5 Centro Notifiche (Notification View)

Un'area dedicata dove l'utente riceve aggiornamenti in tempo reale sulle proprie attività



4.6 Creazione Annuncio (AnnuncioCreateDialog)

Una finestra dialog che guida l'utente nell'inserimento di un nuovo articolo.



4.7 Visualizzazione delle Statistiche e Report

Una area dedicata alle statistiche del proprio profilo, in cui è possibile vedere, tramite grafici, le vendite e gli acquisti effettuati e i guadagni.

