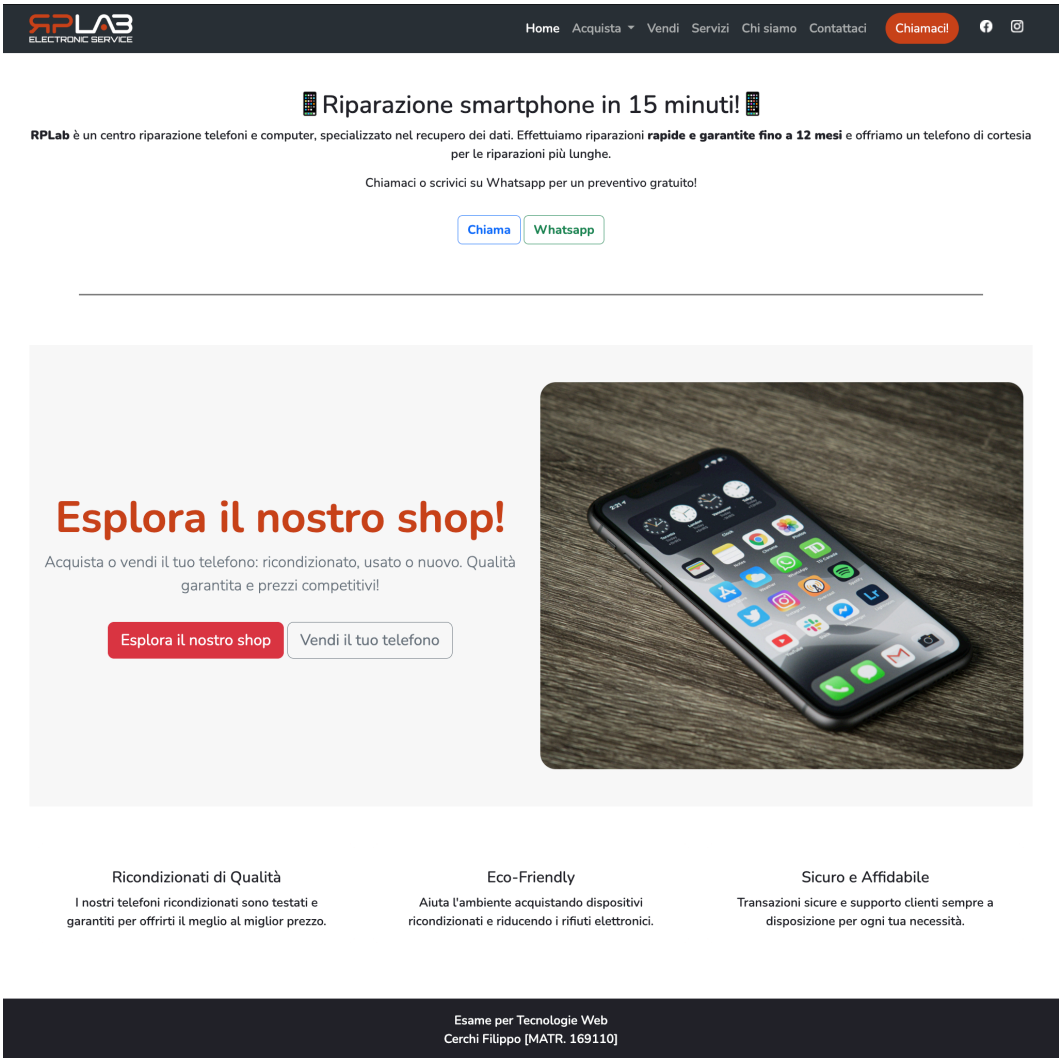
Esame per Tecnologie Web - Cerchi Filippo [MATR. 169110]

# RPLab Electronic Service

## Traccia

Il progetto consiste nello sviluppo di un sito web dinamico per un negozio fisico che offre servizi di riparazione, vendita e acquisto di dispositivi elettronici come smartphone, tablet e PC. Questo sito, basato su Django, mira a fornire un'esperienza utente completa, combinando funzionalità di eCommerce con un sistema per la valutazione automatica dei dispositivi da vendere.



## Descrizione del progetto

Il progetto in questione è un applicazione web per il negozio fisico “RPLab Electronic Service” situato a Mirandola (MO).

La **homepage** del sito è stata progettata per essere intuitiva e user-friendly, compresa di diverse sezioni per accumulare fiducia da parte del cliente. La struttura della pagina include sezioni mirate ad acquisire la fiducia del cliente: ‘servizi’, ‘chi siamo’, ‘contattaci’.

L’app di **eCommerce** si distingue per la sua chiarezza e semplicità d’uso. Tra le principali funzionalità si ha:

- Ricerca del prodotto
- Vetrine con tutti i prodotti e i brand disponibili
- Prodotti filtrati in base ai brand

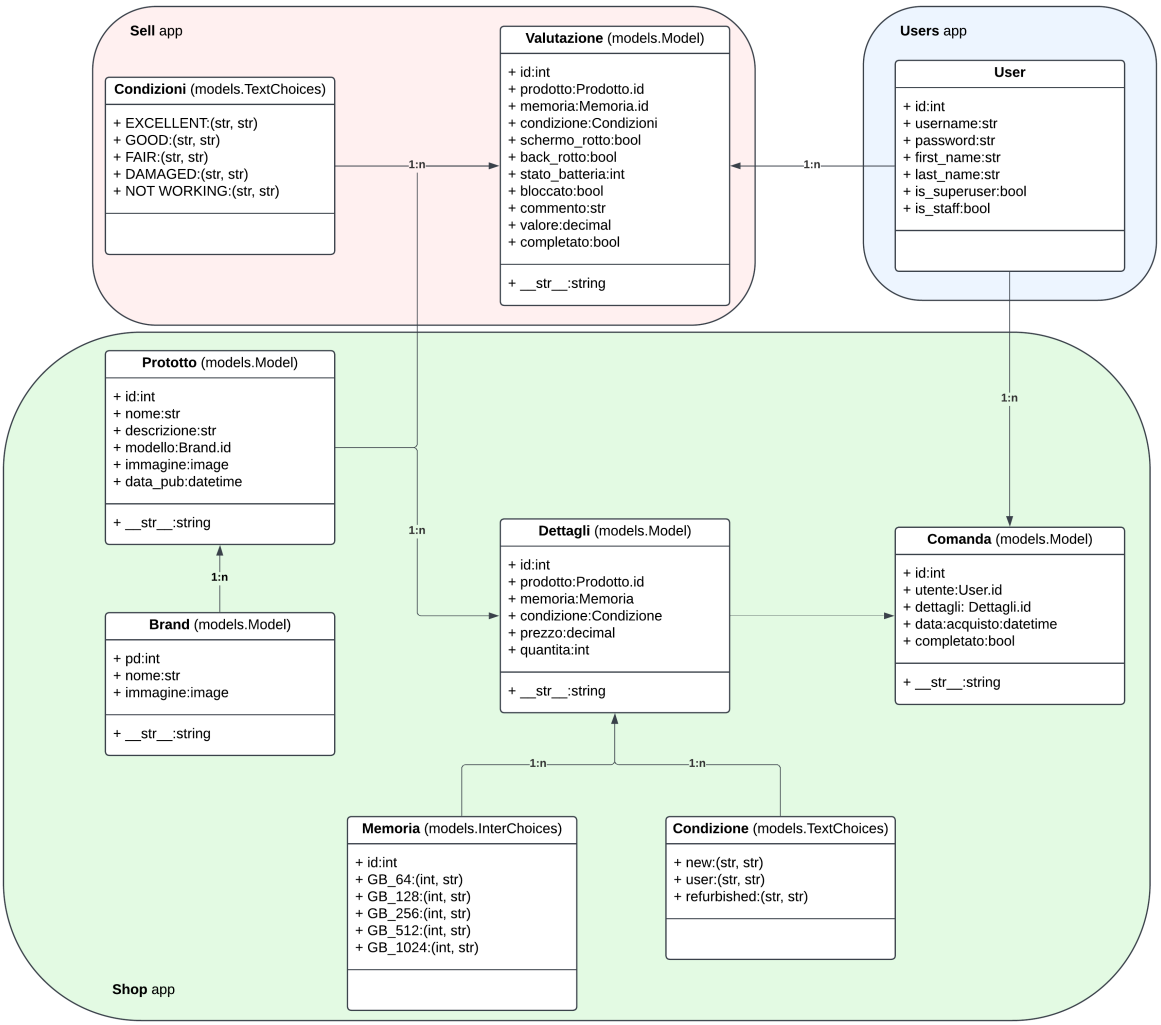
Per poter procedere con l'acquisto, è richiesto che l'utente sia registrato e autenticato all'interno dello shop. Al momento della selezione, l'utente può personalizzare il prodotto scegliendo sia la memoria desiderata sia la condizione (nuovo, usato o ricondizionato).

L’app di **valutazione** di un dispositivo da vendere è altrettanto intuitiva grazie alla presenza di un form guidato, semplice e veloce da compilare. Una volta compilato il form, il prezzo stimato viene immediatamente esposto nella pagina dedicata alla valutazione effettuata, fornendo all'utente un riscontro chiaro e immediato.

La sezione **Dashboard** rappresenta il vero punto di forza di questo progetto, fungendo da centro di controllo sia per gli utenti che per lo staff:

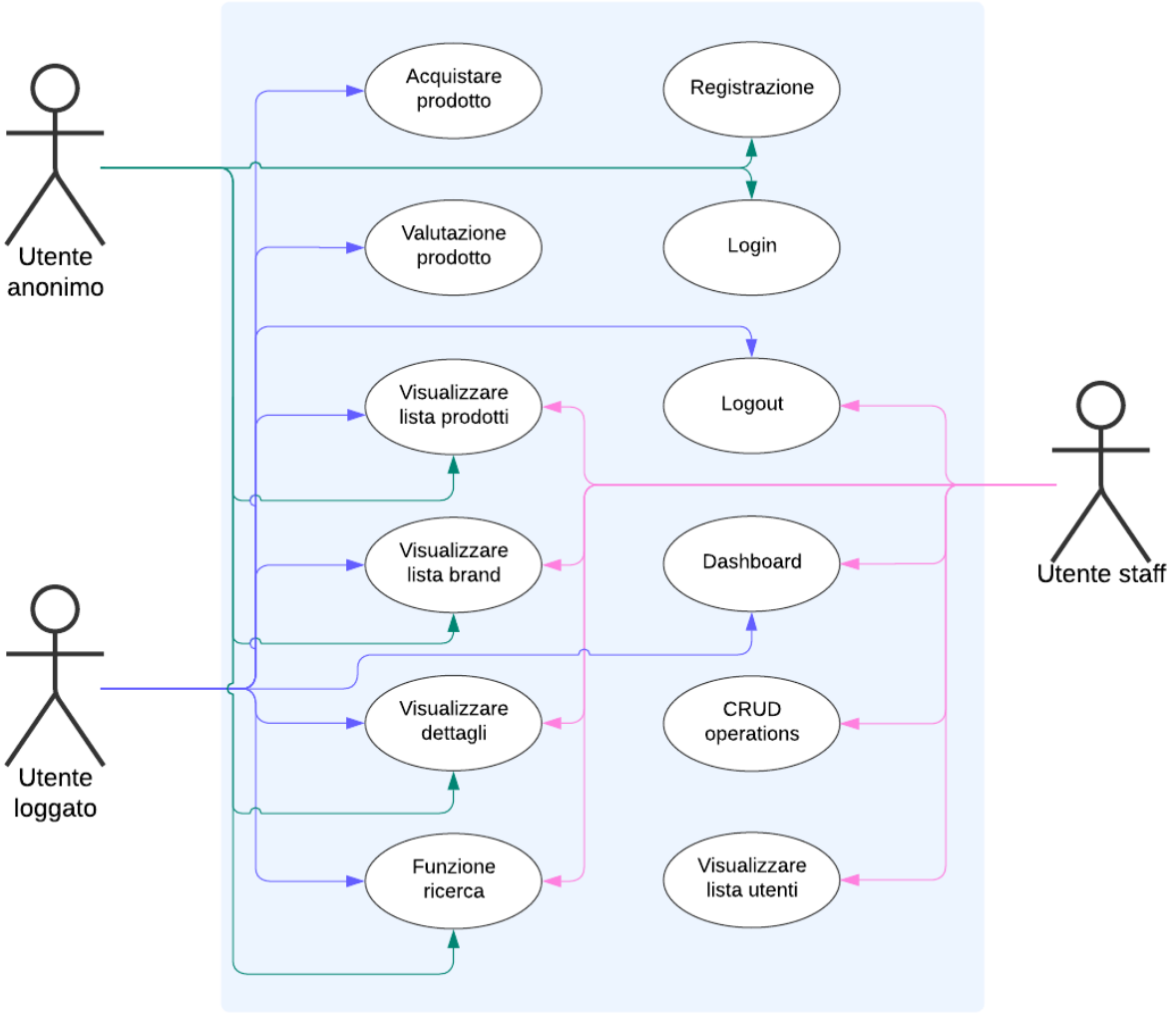
- Per gli utenti: la Dashboard consente di monitorare tutte le valutazioni effettuate e i prodotti acquistati in modo ordinato e accessibile.
- Per lo staff: questa sezione si rivela essenziale per la gestione delle comande e degli utenti registrati, offrendo strumenti avanzati per il controllo e l'ottimizzazione delle operazioni all'interno del sistema.

# Diagramma delle classi



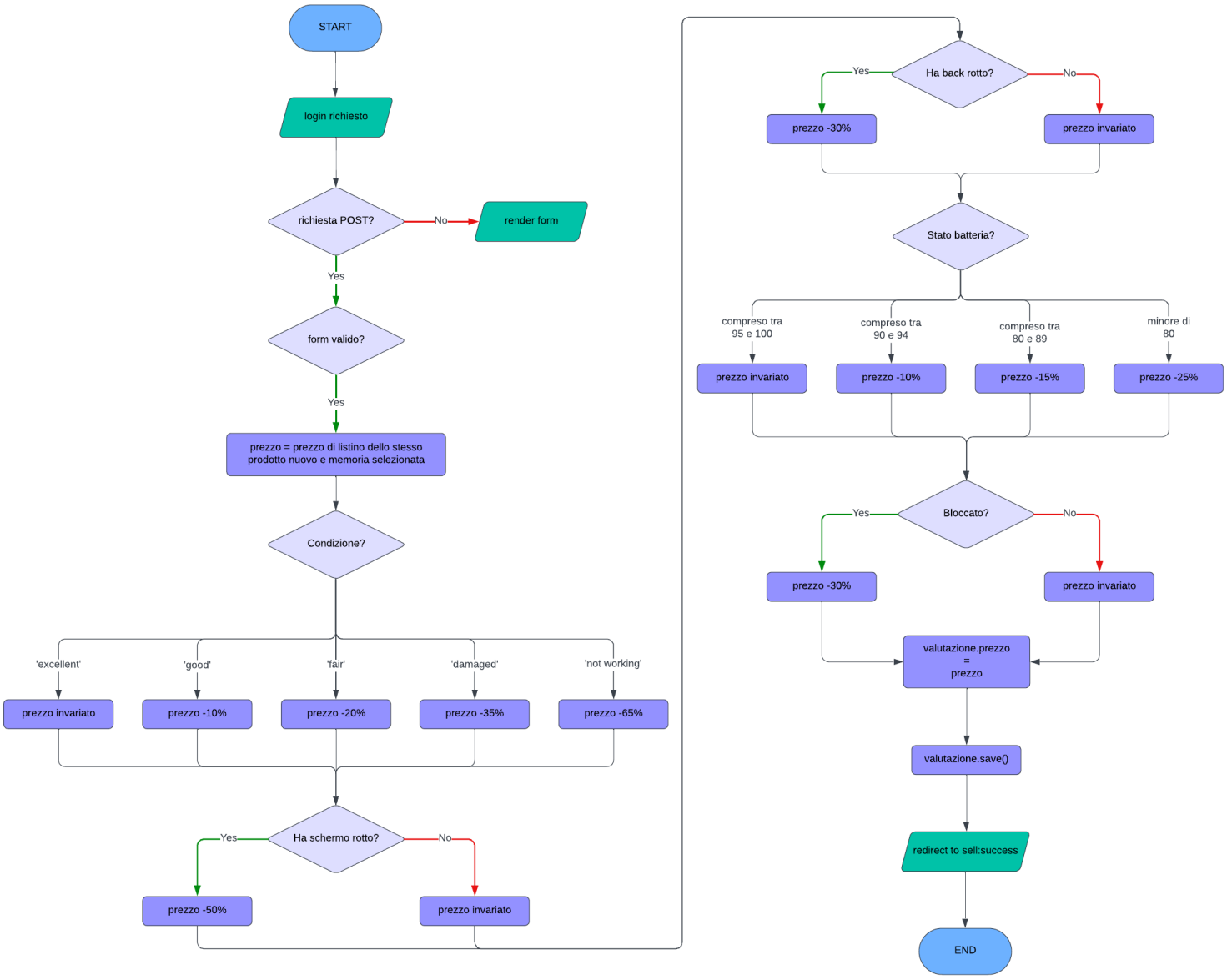

Questi sono i modelli definiti per garantire il corretto funzionamento dell'applicazione web. Essi sono organizzati e suddivisi in base alle applicazioni del progetto.

## Use Case UML




Il diagramma di sopra, evidenzia come i vari utenti possano interagire con diverse funzionalità dell'applicazione, con un percorso di interazione che parte dall'autenticazione (registrazione/login) e arriva alle funzioni più avanzate disponibili solo per gli utenti con ruoli specifici (come lo staff).

# Activity Diagram




Il diagramma in questione, descrive il flusso di valutazione di un prodotto in base a diverse variabili, come lo stato del prodotto, la batteria, la presenza di danni e altre condizioni, con l'obiettivo di determinare un prezzo finale.

## Tecnologie utilizzate

Per garantire il corretto funzionamento e la qualità dell'applicazione web, sono state integrate le seguenti tecnologie:

- **Django**: questo progetto si basa esclusivamente su questo framework, utilizzando Python come linguaggio principale.
- **HTML/CSS**: HTML è stato utilizzato per costruire la struttura semantica delle pagine web, mentre CSS ha permesso di definire lo stile visivo.
- **Bootstrap5**: è stato utilizzato per creare un'interfaccia utente responsiva e professionale. Questa libreria ha accelerato lo sviluppo grazie a griglie predefinite, componenti interattivi e stili CSS personalizzabili.
- **Google Fonts**: è stato integrato per migliorare la tipografia del sito, garantendo un aspetto grafico accattivante e una leggibilità ottimale su diversi dispositivi.
- **Javascript**: è stato impiegato per aggiungere interattività alle pagine web, entrando nel dettaglio, è stato usato per ridurre la lunghezza delle descrizioni dei prodotti.
- **AnimatedCSS**: Questa libreria è stata inclusa per aggiungere animazioni fluide e accattivanti agli elementi delle pagine, migliorando il lato UI/UX.
- **SQLite**: è stato scelto come database per la sua semplicità e integrazione nativa con Django.
- **Visual Studio Code**: è stato l'editor di codice principale utilizzato per lo sviluppo del progetto. Grazie alla sua vasta gamma di estensioni e strumenti di debug, ha facilitato l'implementazione e il testing delle funzionalità del sito.
- **Pillow**: è una libreria di Python che fornisce strumenti per lavorare con immagini.

## Organizzazione logica a livello di codice

Per garantire una struttura modulare e mantenibile, l'applicazione web è stata suddivisa in tre app principali:

### Shop

L'applicazione "Shop" si occupa della gestione delle operazioni legate ai brand, prodotti, comande e alla loro visualizzazione. Possiamo definire l'app come un eCommerce, dove l'utente è in grado di acquistare prodotti in base alle loro esigenze (condizione e memoria del dispositivo).

L'applicazione presenta diversi modelli:

- Brand: nome e immagine di un brand (es. Apple)
- Prodotto: nome, immagine, descrizione, brand associato e data pubblicazione
- Dettaglio: astrazione dettagliata di un prodotto che contiene alcune informazioni tra cui il prodotto associato, condizione, memoria, prezzo e quantità.
- Comanda: astrazione di un acquisto eseguito da un utente, contenente il prodotto nel dettaglio associato, la data di acquisto e se l'ordine è stato completato o meno.

La homepage dell'app è stata strutturata in modo tale da poter esplorare i prodotti e brand disposti in "vetrine". Questa funzionalità è stata resa possibile utilizzando elementi come ListView e DetailView.

Dalla parte di strutturazione delle pagine HTML, è stata applicata la filosofia "DRY", includendo ed estendendo pagine HTML precedentemente preparate e studiate nel dettaglio.

Anche il lato DTL è stato messo in pratica, utilizzando cicli e istruzioni condizionali, oltre che al contenuto (context) reso disponibile dalla vista associata.

È inoltre possibile ricercare un prodotto tramite nome, utilizzando la barra di ricerca situata nella navbar.

## Sell

L'applicazione "Sell" è progettata per gestire la valutazione dei prodotti usati che gli utenti desiderano vendere, in stile swappie.com. Essa è integrata con l'app “Shop", in quanto il prodotto da vendere deve essere già presente nel database dell'eCommerce.

L’applicazione si basa su un unico modello:

- Valutazione: associata all'utente e al prodotto, include informazioni sulla memoria, la condizione del prodotto, la presenza di danni (schermo rotto, retro danneggiato), lo stato della batteria, se il dispositivo è bloccato, eventuali commenti dell'utente, il valore stimato e lo stato di completamento della valutazione.

La valutazione del prodotto viene resa possibile attraverso un form, che l’utente, necessariamente registrato, deve compilare. Una volta che tutti i campi obbligatori sono stati correttamente compilati, un algoritmo di valutazione calcola il valore del dispositivo e, al termine del processo, l'utente viene indirizzato a una pagina di conferma contenente il valore stimato del prodotto.

## Users

L'applicazione "Users" si occupa della gestione degli utenti e delle relative associazioni, come ordini, valutazioni e altre attività correlate. Sebbene non presenti modelli particolarmente complessi, se non quello dell'utente, svolge un ruolo cruciale nell'intero sistema.

Una delle principali caratteristiche di questa applicazione è la sezione **"**Dashboard**"**, che consente di rilevare il ruolo di ciascun utente (superuser, staff o utente normale). In base al ruolo, la dashboard fornisce funzionalità e informazioni specifiche. Per gli utenti appartenenti al gruppo staff, la dashboard visualizza le comande e le valutazioni attive (non completate), permettendo loro di modificarle o eliminarle secondo necessità.

Inoltre, l'applicazione Users consente di eseguire operazioni CRUD (Create, Read, Update, Delete) su tutti i modelli del progetto. Queste operazioni vengono gestite tramite le relative View, che forniscono un'interfaccia intuitiva e sicura per la gestione dei dati associati agli utenti e agli altri modelli del sistema.

Queste operazioni sono state organizzate in cartelle dedicate al fine di mantenere una struttura di progettazione chiara e ben definita, evitando confusione.

## Decisioni adottate

I. Ho deciso di implementare tutte le operazioni CRUD da parte dello staff all'interno dell'app '`users`' per mantenere coerente la parte amministrativa del progetto e centralizzare la gestione di utenti, comande e valutazioni. Questo approccio semplifica l'organizzazione del codice e facilita la manutenzione, in quanto tutte le funzionalità relative all'amministrazione degli utenti e dei dati correlati si trovano in un'unica applicazione

II. Per garantire un'esperienza utente ottimale, ho utilizzato delle class-based views come '`ListView`', '`DetailView`' e '`UpdateView`' per implementare funzionalità standard come la visualizzazione e la modifica di modelli.

III. Ho protetto le operazioni amministrative con decoratori come '`@staff_member_required`' e '`@login_required`' per assicurare che solo utenti autenticati o membri dello staff possano accedere alle relative funzionalità.

IV. È stata una scelta strategica implementare un sistema gestionale per le comande e le valutazioni da parte dello staff, con la possibilità di filtrare gli elementi non completati. Questo approccio consente di ottenere una lista organizzata delle attività, ordinata in maniera crescente per ID, ottimizzando così la pianificazione e il monitoraggio delle operazioni in corso.

V. Aver utilizzato JavaScript in alcuni contesti, come la riduzione del testo delle descrizioni dei prodotti esposti in vetrina e la disabilitazione del pulsante 'Cerca' quando la barra di ricerca è vuota, mi ha aiutato a migliorare l'usabilità dell'interfaccia utente e a prevenire comportamenti indesiderati.

VI. L'utilizzo di un unico template, '`base.html`', come struttura comune per tutte le pagine dell'applicazione web mi ha permesso di mantenere la barra di navigazione e l'header in ogni pagina. Questa scelta progettuale non solo ha migliorato la coerenza visiva e funzionale del sito, ma ha anche aderito alla filosofia DRY (Don't Repeat Yourself) promossa da Django.

VII. Nel file '`models.py`' all'interno dell'app '`shop`', si possono notare due funzioni che permettono alla variabile '`immagine`' di essere salvata in modo coerente con gli altri parametri del modello, utilizzando come base il nome dell'istanza per generare un file univoco e organizzandole in percorsi specifici della struttura del progetto. Questo approccio garantisce il recupero delle immagini associate ai modelli, migliorando la gestione delle risorse statiche.

## Shop

1. **Test della Creazione di Brand, Prodotto e Dettaglio**: Verifica che i modelli ‘`Brand`’, ‘`Prodotto`‘ e ‘`Dettagli`’ vengano creati correttamente.
2. **Test del Form di Condizione del Prodotto**: Testa la validità del form per l'acquisto di un prodotto, sia con dati validi, sia con dati invalidi.
3. **Test delle Viste (Home, Prodotti, Dettaglio Prodotto, Cerca)**: Verifica il corretto funzionamento delle viste principali.
4. **Test della Vista Acquista**: Testa il comportamento della vista per l'acquisto di un prodotto, sia per utenti autenticati che non autenticati.

## Sell

1. **Test della Vista Home Sell**: Verifica il corretto funzionamento della homepage.
2. **Test del Form di Valutazione con Dati Validi**: Testa il form di valutazione del prodotto con dati validi e verifica che venga creato un record di valutazione con il valore corretto.
3. **Test del Form di Valutazione con Dati Invalidi**: Verifica il comportamento del form con dati non validi (ad esempio, stato della batteria maggiore di 100).
4. **Test della Vista di Successo**: Verifica la corretta visualizzazione della pagina di successo dopo che una valutazione è stata completata.

## Users

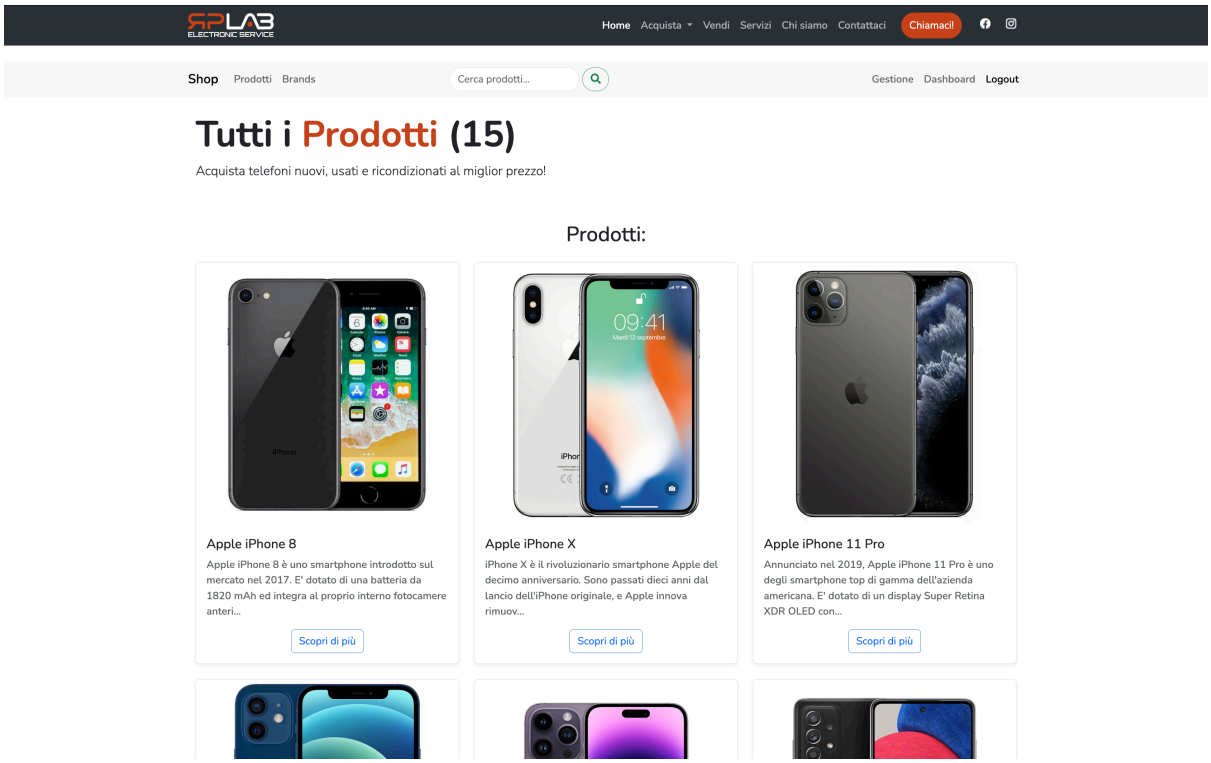
1. **Test di Registrazione Utente**: Verifica che il processo di registrazione funzioni correttamente, creando un nuovo utente e reindirizzando l'utente alla pagina di login.
2. **Test di Login**: Controlla che l'utente possa accedere correttamente al sistema, con indicizzazione alla home del negozio.
3. **Test di Logout**: Verifica che l'utente possa effettuare il logout e venga reindirizzato alla pagina di login.

4. **Test della Dashboard Utente**: Testa la visualizzazione della dashboard per un utente loggato, assicurandosi che vengano mostrati i dati corretti, come il brand e il prodotto.
5. **Test CRUD Utente (staff)**: Testa la visualizzazione della lista utenti, i dettagli di un singolo utente e la modifica dei dati utente per un account con permessi di staff.
6. **Test di Aggiornamento Utente**: Verifica che un utente possa aggiornare correttamente le proprie informazioni e che i cambiamenti siano salvati nel database.

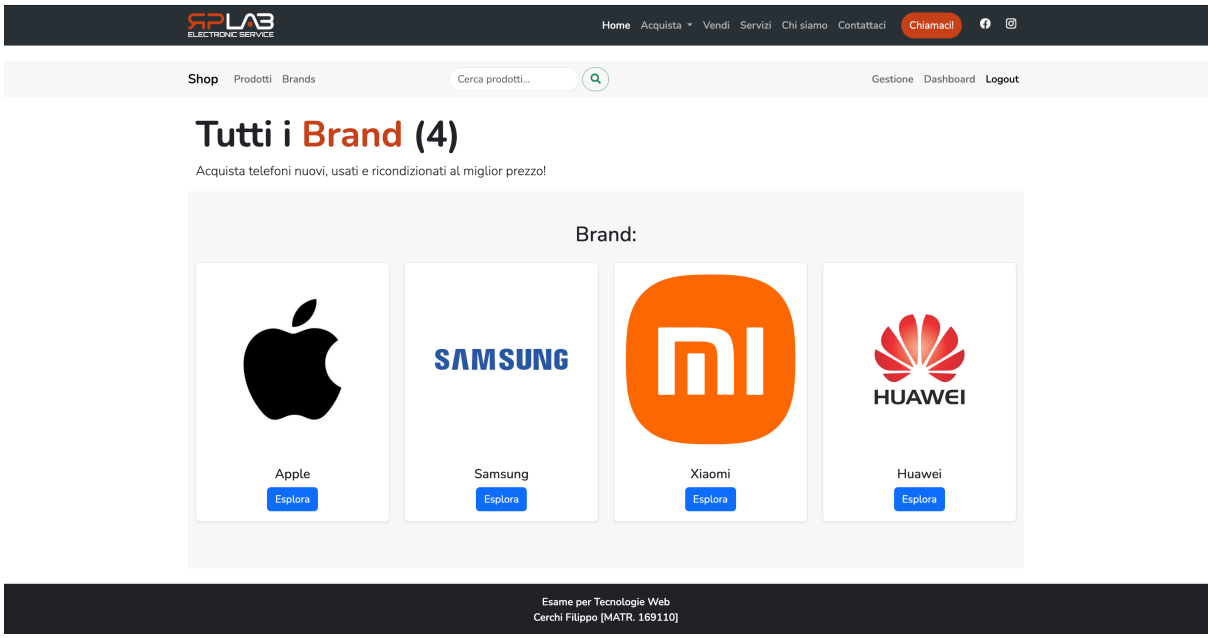
---

## Screenshot interessanti

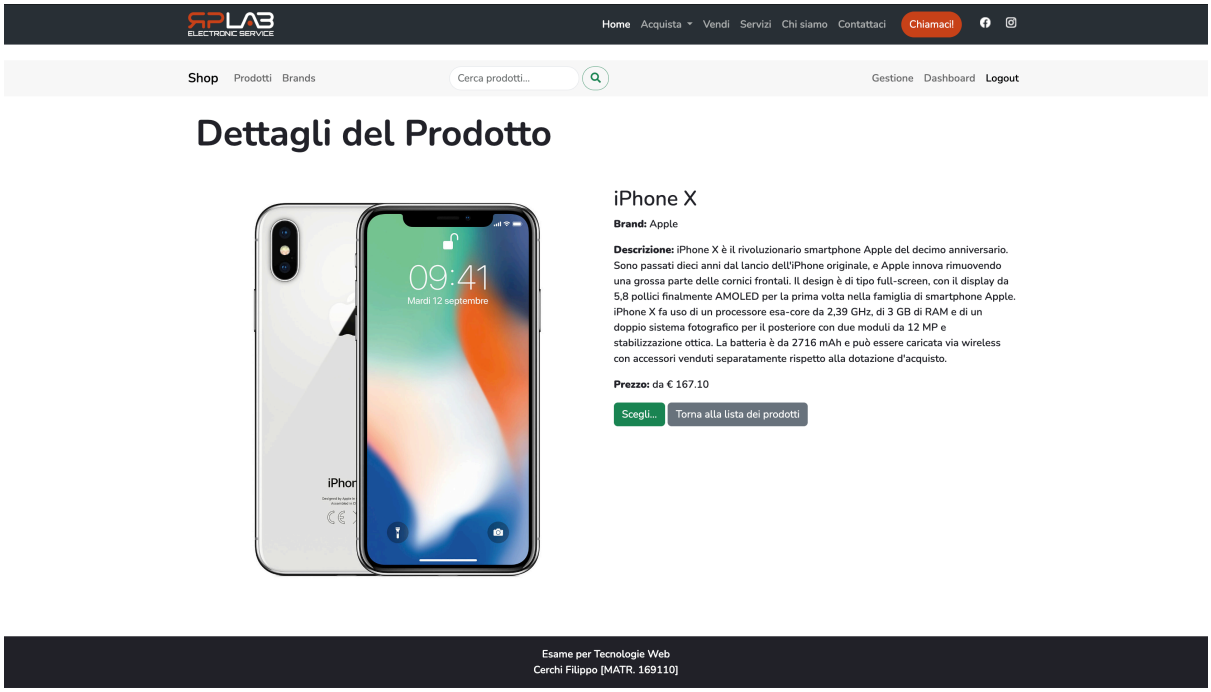
### Tutti i prodotti



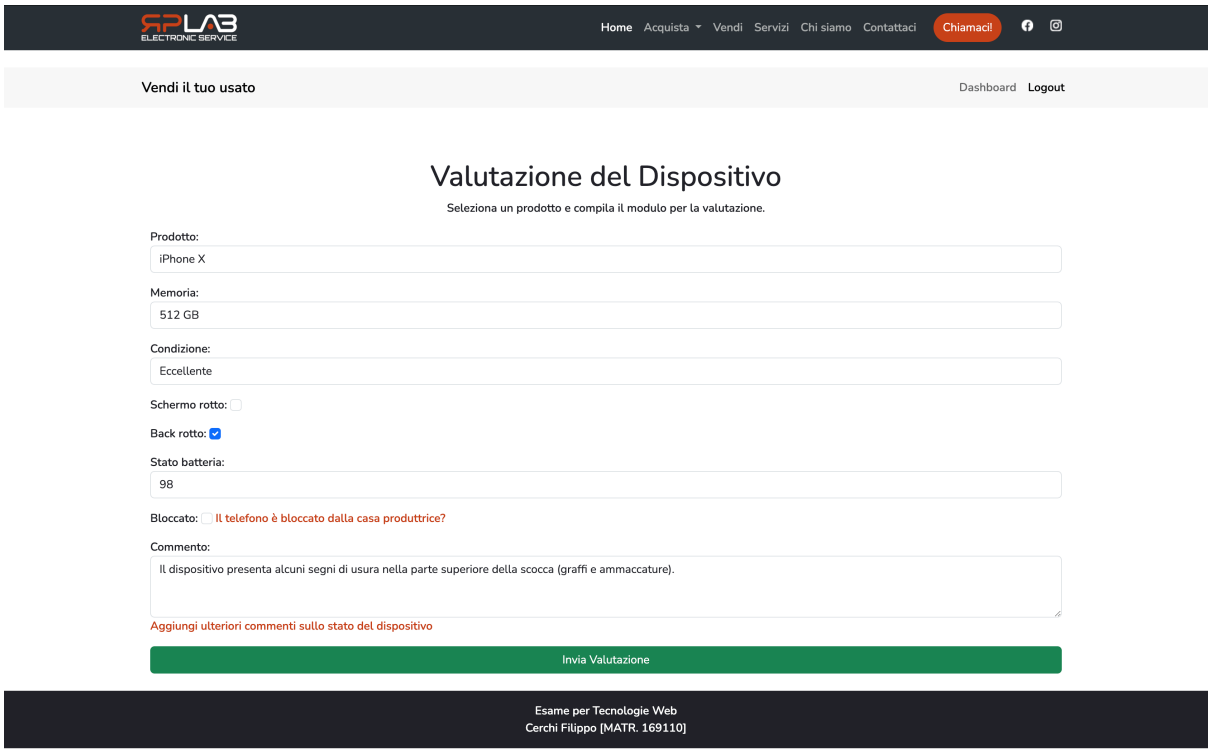
# Tutti i brand



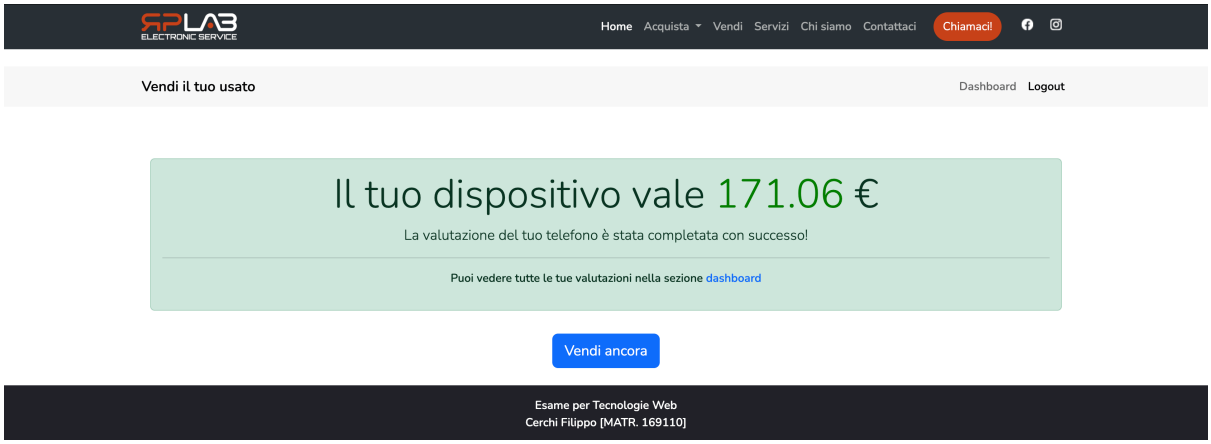

# Dettagli del prodotto



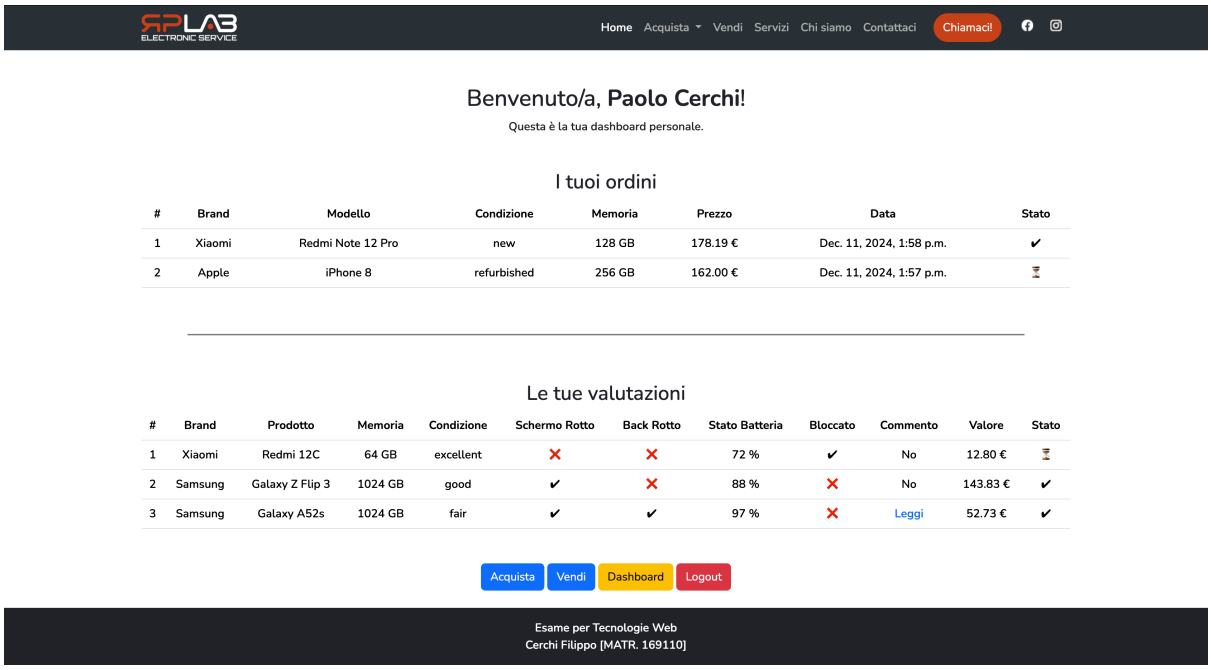
# Form per la valutazione del dispositivo



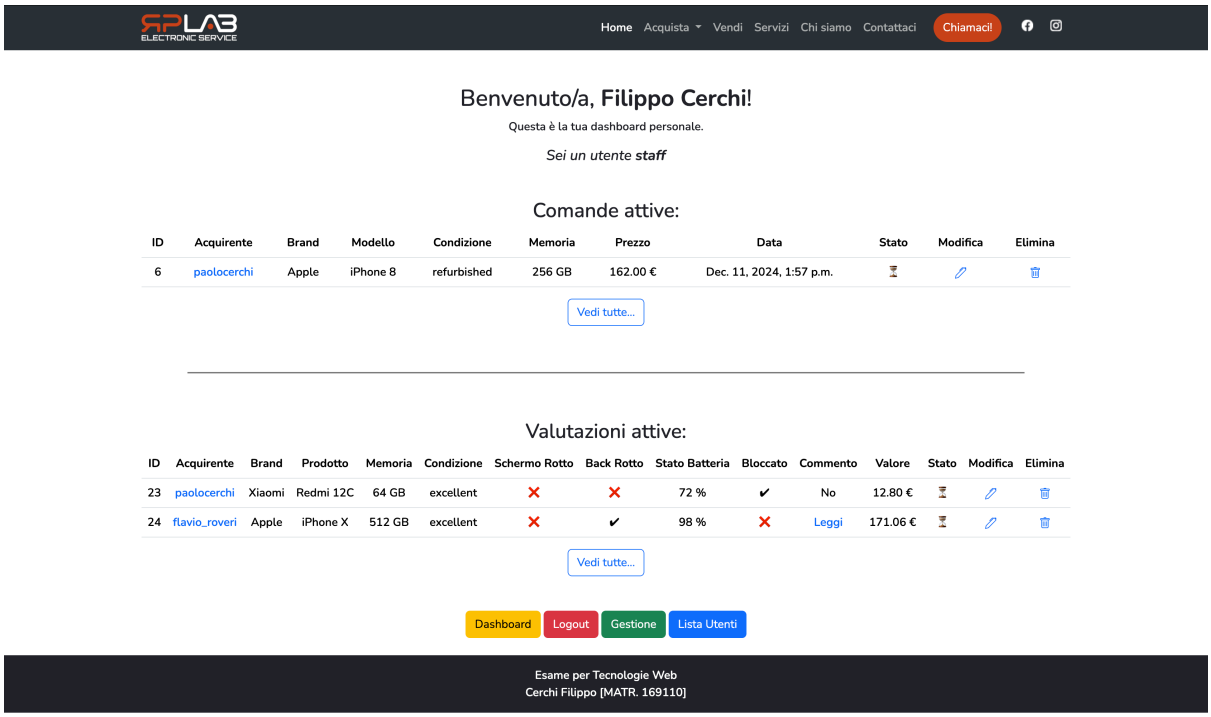

# Valutazione effettuata



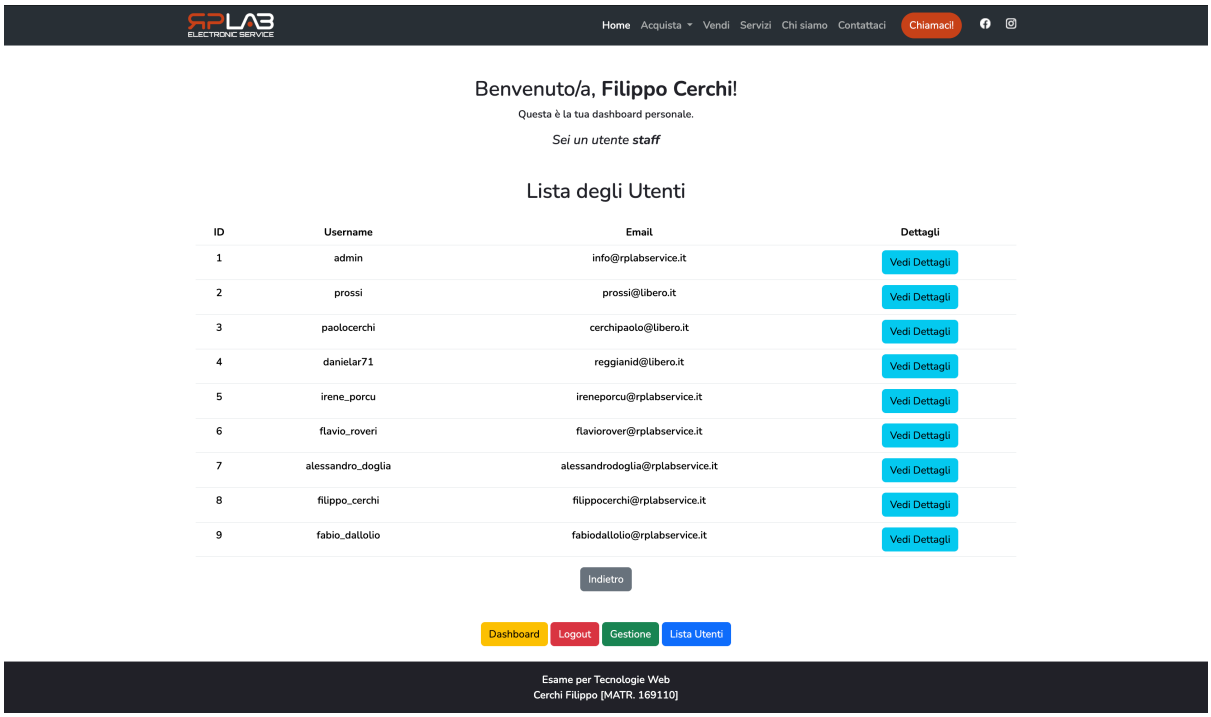
# Dashboard utente



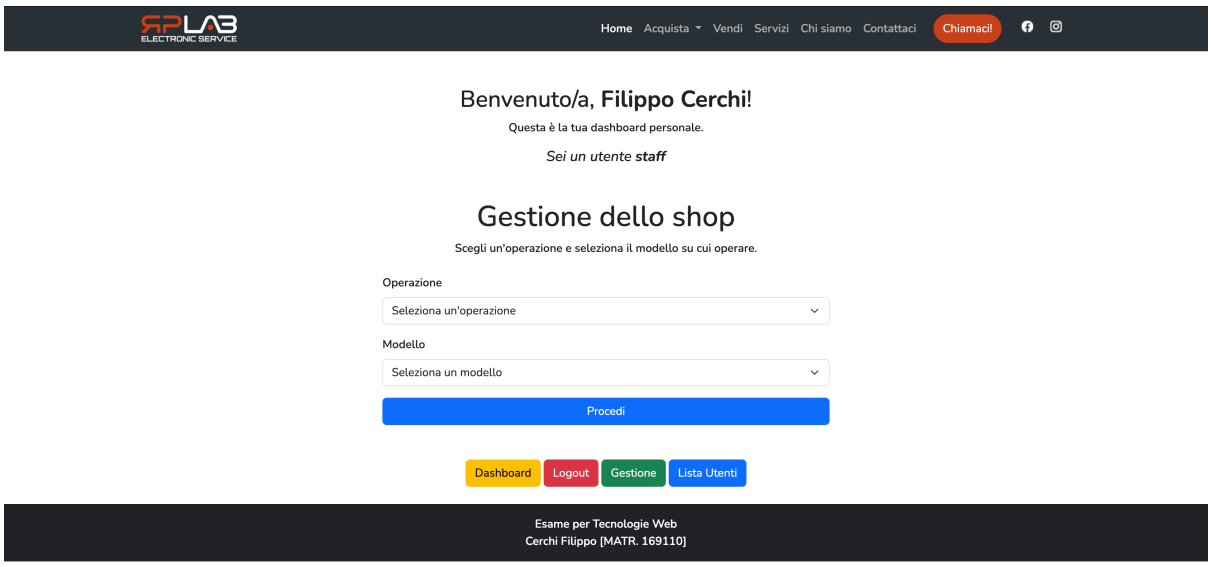

# Dashboard utente staff



# Gestione utenti




# Home operazioni CRUD



## Problemi riscontrati

1. **Gestione delle immagini**: inizialmente, la gestione delle immagini per i prodotti e i brand ha causato problemi con i percorsi di salvataggio. Il campo destinato alle immagini era definito come '`CharField`', che memorizzava manualmente il percorso dell'immagine. Tuttavia, a metà progetto, ho deciso di sostituire questa implementazione con un '`ImageField`', al fine di ottimizzare la gestione e il salvataggio delle immagini.

## Miglioramenti futuri

1. **Aggiunta del colore nel dettaglio prodotto**: implementazione della possibilità di selezionare il colore del dispositivo desiderato, con relativa immagine
2. **Miglioramento dell'algoritmo di valutazione**: sebbene l'algoritmo di valutazione sia già funzionante, l'obiettivo è perfezionarlo per ottenere valutazioni più precise e affidabili, migliorando l'accuratezza delle stime
3. **Gestione delle riparazioni**: introduzione di una nuova applicazione denominata '`Fix`' che consentirà agli utenti di aprire ticket per la riparazione di dispositivi. Lo staff avrà la possibilità salvare tutte le informazioni necessarie al fine di poter riparare il dispositivo e aggiornare l'utente in tempo reale sullo stato della riparazione mediante la dashboard (stile gestionale in cloud).
4. **Aggiunta del tipo di prodotto**: funzionalità che permetterà all'utente di scegliere il tipo di dispositivo da acquistare/valutare (tablet, PC, console, etc…)
5. **Filtraggio dei prodotti**: sviluppo di un sistema in grado di filtrare i prodotti in base alle loro proprietà (memoria, condizione, etc…)