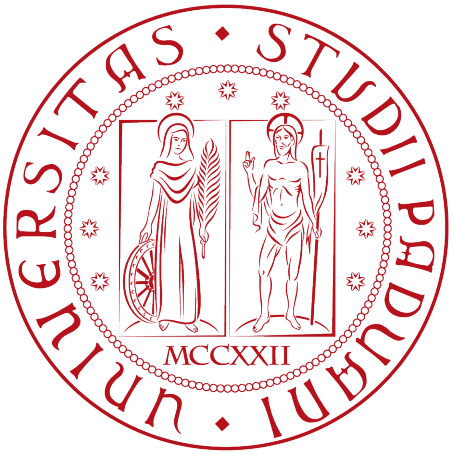
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Sviluppo di un'applicazione mobile per la gestione delle spese dei pranzi dell'azienda con Flutter

Tesi di laurea

Relatore

Prof. Ombretta Gaggi

Laureando

Andrea Crocco
Matricola 1226135

Anno Accademico 2022-2023

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di trecento ore, dal laureando Andrea Crocco presso l'azienda RiskApp S.r.l..
L'obiettivo del progetto di stage è stato quello di sviluppare un'applicazione mobile per la gestione interna delle spese relative ai pranzi effettuati dallo staff dell'azienda.
Durante lo svolgimento dello stage, innanzitutto si è svolta l'analisi dei requisiti, si è passati poi alla progettazione, e infine allo sviluppo e alla verifica delle funzionalità implementate. Tali fasi vengono analizzate nel dettaglio nei capitoli che compongono la presente tesi.

Convenzioni tipografiche

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*$^{[g]}$;
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

"Life is really simple, but we insist on making it complicated"

— Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Ombretta Gaggi, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Settembre 2023 Andrea Crocco
Matricola 1226135

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 L'azienda

La società RiskApp S.r.l. logo in Figura 1.1, con sede a Conselve (Padova), è stata fondata nel 2016 ed è specializzata nello sviluppo di software. Il loro prodotto principale è la piattaforma RiskApp, che rappresenta il fulcro delle attività aziendali, occupandosi della vendita e della manutenzione.
Questo software è stato concepito con l'obiettivo di fornire una visione completa dei rischi che le compagnie assicurative potrebbero affrontare. Grazie a un algoritmo proprietario basato sull'intelligenza artificiale, RiskApp raccoglie e analizza dati provenienti da diverse fonti, consentendo di stimare le possibili perdite finanziarie per le aziende clienti.
La distribuzione del prodotto avviene tramite il modello *Software as a Service (SaaS)*[g] [27] e ad oggi l'azienda intrattiene relazioni commerciali con importanti operatori del settore assicurativo. Inoltre, RiskApp S.r.l. collabora attivamente con l'Università degli Studi di Padova.

Figura 1.1: Logo dell'azienda.

1.1.1 Processi aziendali

Dato che il gruppo di lavoro è costituito da un numero limitato di individui, l'azienda si avvicina allo sviluppo del software adottando alcuni concetti chiave della metodologia Agile[g] [1]:

- Coinvolgimento del cliente: vengono frequentemente programmati incontri con i clienti, solitamente virtuali ma talvolta anche in presenza, per identificare e rivedere i requisiti.
- Comunicazione istantanea: le comunicazioni all'interno del team avvengono in modo rapido. L'ambiente di lavoro ridotto consente una comunicazione veloce

tra i membri del team, mentre per la comunicazione a distanza si utilizzano strumenti dedicati.

- Sviluppo iterativo: le diverse componenti del software vengono costantemente sviluppate e migliorate nel tempo. In questo modo, il prodotto può essere distribuito ai clienti in uno stato funzionante in tempi relativamente brevi, per poi essere migliorato ed esteso in base ai feedback ricevuti dal cliente.
- Versionamento: la *Codebase*[g] [4] è organizzata in diversi repository ospitati su GitHub per il controllo delle versioni.

1.2 Lo Stage

1.2.1 Scopo

L'azienda ha messo a disposizione ai propri dipendenti una cucina interna presso la sede lavorativa, che consente loro di preparare il pranzo giornaliero in modo autonomo e condiviso. I dipendenti si organizzano direttamente per fare la spesa e procurarsi gli ingredienti necessari alla preparazione delle pietanze. Durante la pausa pranzo, poi, è possibile consumare i pasti tutti insieme nella cucina aziendale, in un'ottica di socializzazione e collaborazione tra colleghi. Le spese sostenute per gli acquisti alimentari vengono poi equamente suddivise tra i membri dello staff che decidono di aderire a questa modalità di pausa pranzo autogestita e partecipata. Per gestire in modo trasparente ed efficiente le spese sostenute, l'azienda si avvale di un sistema di tracciamento delle stesse mediante l'utilizzo di un documento Excel condiviso. In questo foglio elettronico vengono registrate per ogni membro dello staff:

- Il numero dei pasti effettuati in quella settimana;
- La quota dei pasti totali mensili;
- Le spese sostenute e i soldi ricevuti da parte degli altri membri dello staff;
- La quota stornata (calcolata come differenza tra la quota totale dei pasti meno le spese sostenute e ricevute);

Per ottimizzare il tutto si è desiderata l'idea di un'applicazione multipiattaforma per la gestione interna delle spese relative ai pranzi effettuati dallo staff dell'azienda con le seguenti funzionalità:

- Accesso tramite autenticazione degli utenti (autenticazione e database gestiti tramite il servizio *Firestore*) [11];
- Monitoraggio delle spese dei pranzi, consentendo agli utenti di inserire le transazioni con i relativi dettagli;
- Monitoraggio della cassa comune, mostrando il saldo attuale e le spese effettuate;
- Consigliare su base giornaliera alcune ricette, tramite l'utilizzo di *Chat Generative Pre-trained Transformer (ChatGPT)*[g] [3];
- Permettere la scansione automatica dei dati di base degli scontrini delle spese sostenute, tramite fotocamera del telefono.

1.2.2 Obiettivi

Nella Tabella 1.1 vengono riportati gli obiettivi dello stage. Essi sono identificati dalla seguente sigla:

- **O** per gli obiettivi obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- **D** per gli obiettivi desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- **F** per gli obiettivi facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, identificativo dell'obiettivo.

Tabella 1.1: Tabella degli obiettivi dello stage.

Codice	Descrizione
O1	Accesso tramite credenziali
O2	Monitoraggio spese della cassa comune
O3	Aggiornamento delle spese
D1	Integrazione di un algoritmo per l'integrazione automatica degli scontrini delle spese effettuate per comprare il cibo per il pranzo
D2	Integrazione di *ChatGPT* [3] per ricevere consigli sulle ricette giornaliere, in quanto predispongono di cucina interna
F1	Stesura dei test
F2	Deploy WebApp

1.3 Organizzazione del testo

Il secondo capitolo descrive l'analisi dei requisiti, delineando gli attori partecipanti nell'interazione con le funzionalità implementate e delineando i casi d'uso identificati. Inoltre, vengono categorizzati i requisiti in base alla loro tipologia e rilevanza.

Il terzo capitolo descrive la fase di progettazione e le tecnologie utilizzate.

Il quarto capitolo approfondisce la fase di codifica descrivendo nel dettaglio il funzionamento dell'applicazione.

Il quinto capitolo descrive gli obiettivi e le competenze raggiunte accompagnato dalla valutazione personale.

Capitolo 2

Analisi dei requisiti

Il presente capitolo descrive l'analisi dei requisiti. Presenta gli attori coinvolti nell'interazione con le funzionalità implementate e descrive i casi d'uso individuati. Vengono riportati i requisiti i quali sono classificati in base alla tipologia e all'importanza.

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo Unified Modeling Language (UML) [29] dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Essendo il progetto finalizzato alla creazione di un tool per l'automazione di un processo, le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

2.1 Attori

Come si può vedere nella figura 2.1 ci sono 4 tipi di attori: l'**utente** che rappresenta un membro dello staff dell'azienda, il quale può:

- Monitorare le spese dei pranzi, consentendogli di inserire le transazioni con i relativi dettagli;
- Monitoraggio della cassa comune, mostrandogli il saldo attuale e le spese effettuate;
- Consigliargli su base giornaliera alcune ricette, tramite l'utilizzo di *ChatGPT* [3];
- Permettergli la scansione automatica dei dati di base degli scontrini delle spese sostenute, tramite fotocamera del telefono;

Dopodiché c'è l'**amministratore** che, oltre a poter svolgere le funzioni dell'utente normale, ha il compito inoltre di:

- Cambiare il costo del singolo pasto;
- Registrare il pasto giornaliero per gli stagisti;

In seguito abbiamo il **sistema di autenticazione**, con il quale è possibile autenticarsi tramite *Single sign-on*[g] [24] di *Microsoft*.

Infine abbiamo ***ChatGPT*** che viene usato per proporre nuove ricette sulla base degli ingredienti disponibili nella dispensa.

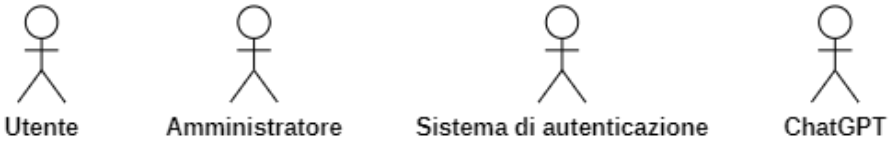


Figura 2.1: Attori.

2.2 Casi d'uso

Nel caso d'uso principale (Figura 2.2), l'utente inizia accedendo al sistema attraverso le credenziali o il *SSO* [24]. Una volta autenticato, può interagire con i pulsanti per navigare tra le diverse schermate dell'applicazione. Inizialmente, nella schermata principale (*home*), l'utente può visualizzare un resoconto del numero di pasti consumati nel mese corrente, l'ammontare attuale e la cassa comune con le quote stornate di tutti i membri dello staff. Inoltre, tramite *ChatGPT* [3], è possibile ottenere un'anteprima di una ricetta basata sugli ingredienti attualmente presenti nella dispensa.
La seconda schermata è dedicata alla *dispensa*, dove sono presenti due tab contenenti rispettivamente tutti gli ingredienti disponibili e la lista della spesa. Successivamente, c'è la sezione *aggiunta pasti*, dove l'utente può indicare i giorni in cui intende pranzare in azienda.
Nella penultima schermata, l'utente ha la possibilità di consultare un resoconto cronologico di tutte le transazioni o spese effettuate dagli utenti in un determinato periodo, oltre a visualizzare gli utenti a cui deve saldare debiti.
Infine, nella sezione *profilo*, l'utente può modificare diversi parametri, i quali saranno descritti successivamente.

2.2.1 Scenario principale

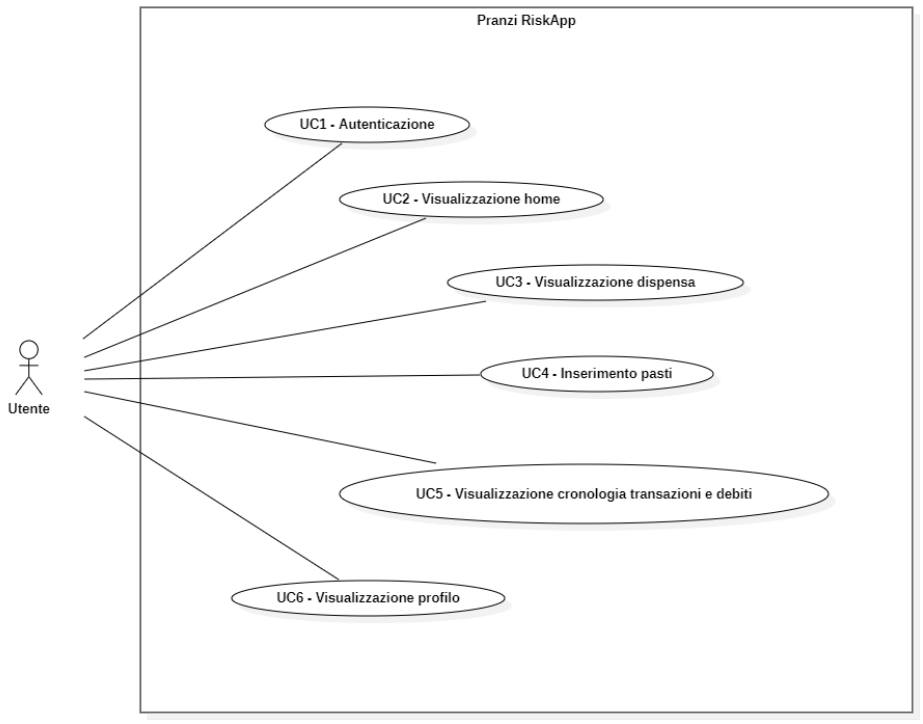


Figura 2.2: UC0: Scenario principale.

UC1: Autenticazione

Attori Principali: Utente.

Precondizioni: L'utente inserisce le proprie credenziali d'accesso (email e password).

Descrizione: Pagina di autenticazione dei membri dello staff per accedere all'applicazione.

Postcondizioni: Il sistema riconosce l'utente e lo lascia entrare.

2.2.2 Use Case 1: Autenticazione

La Figura 2.3 riporta i sotto casi d'uso relativi all'autenticazione.

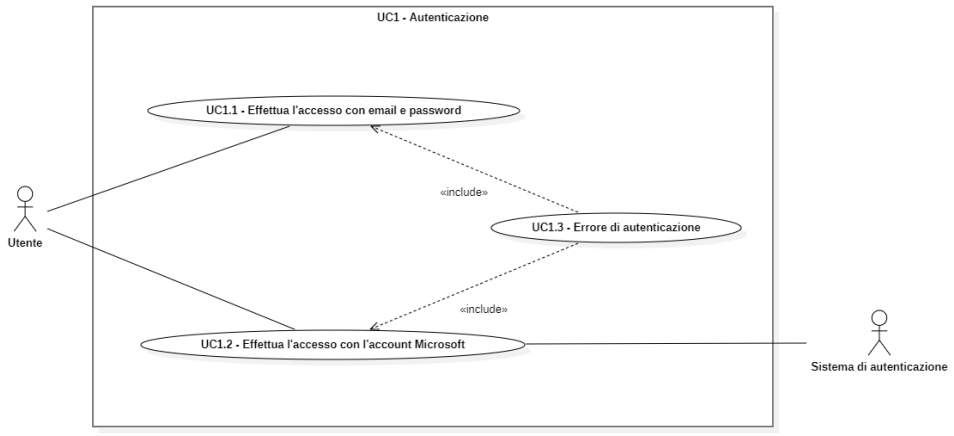


Figura 2.3: UC1: Autenticazione.

UC1.1: L'utente effettua l'accesso con email e password

Attori Principali: Utente.

Precondizioni: L'utente inserisce email e password.

Descrizione: Inserimento credenziali.

Postcondizioni: L'utente ha inserito le credenziali ed è pronto ad accedere all'applicazione.

UC1.2: L'utente effettua l'accesso con l'account *Microsoft*

Attori Principali: Utente, sistema di autenticazione.

Precondizioni: L'utente clicca il pulsante "Login con *Microsoft*".

Descrizione: Accesso alla piattaforma usando *SSO* [24] di *Microsoft*.

Postcondizioni: L'utente ha ricevuto un riscontro da *Microsoft*.

UC1.3: Errore di autenticazione

Attori Principali: Utente.

Precondizioni: L'utente ha inserito le proprie credenziali o ha provato ad accedere usando il *SSO* di *Microsoft*.

Descrizione: Errore mostrato quando l'utente o non è registrato o ha sbagliato a inserire le credenziali.

Postcondizioni: L'utente ha ricevuto un riscontro negativo da parte dell'applicazione negandogli l'accesso alla piattaforma.

2.2.3 Use Case 2: Visualizzazione home

La Figura 2.4 riporta i sotto casi d'uso relativi alla visualizzazione della schermata home.

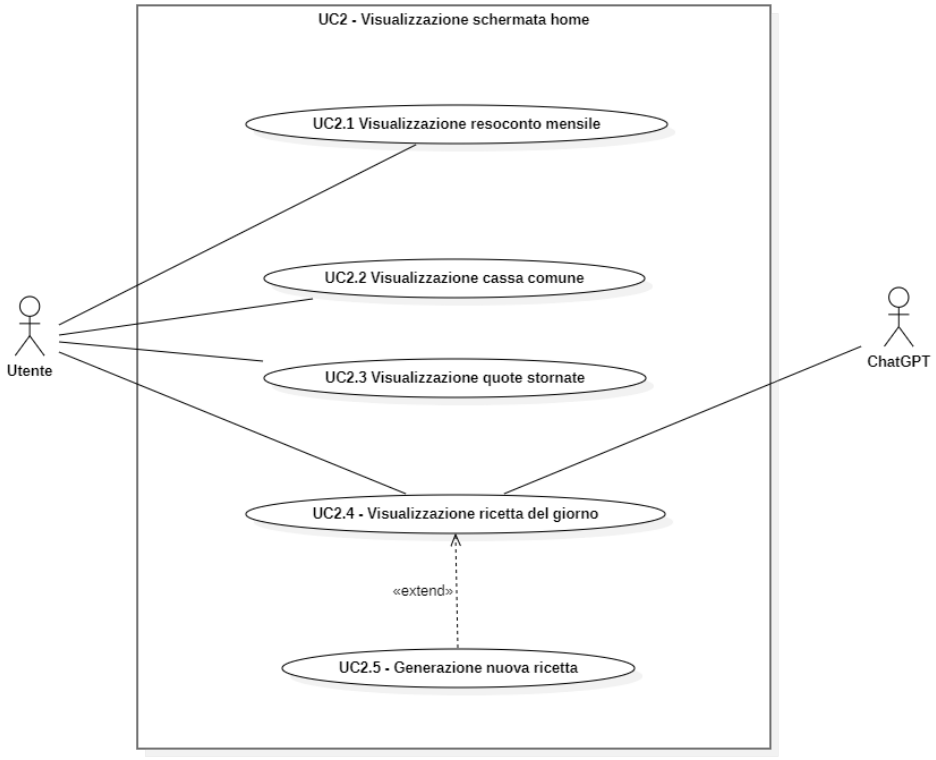


Figura 2.4: UC2: Visualizzazione schermata home.

UC2.1: Visualizzazione resoconto mensile

Attori Principali: Utente.

Precondizioni: L'utente preme il pulsante per vedere il suo resoconto mensile.

Descrizione: Serve per reindirizzare l'utente alla pagina del resoconto mensile di spese e numero di pasti.

Postcondizioni: L'utente viene reindirizzato alla pagina del resoconto mensile.

UC2.2: Visualizzazione cassa comune

Attori Principali: Utente.

Precondizioni: L'utente fa tap su "cassa comune".

Descrizione: L'utente può vedere quanto denaro è stato risparmiato nella cassa comune.

Postcondizioni: L'utente vede il risparmio attuale.

UC2.3: Visualizzazione quote stornate

Attori Principali: Utente.

Precondizioni: L'utente fa tap su "quote stornate".

Descrizione: L'utente può vedere una lista di tutte le persone che sono in credito o in debito.

Postcondizioni: L'utente vede le quote stornate di tutti gli utenti.

UC2.4: Visualizzazione ricetta del giorno

Attori Principali: Utente, *ChatGPT*.

Precondizioni: L'utente entra nella sezione home dell'applicazione.

Descrizione: Visualizzazione della ricetta del giorno sulla base degli ingredienti disponibili nella dispensa.

Postcondizioni: L'utente viene reindirizzato alla ricetta completa generata da *ChatGPT* [3].

UC2.5: Generazione nuova ricetta

Attori Principali: Utente, *ChatGPT*.

Precondizioni: L'utente clicca il pulsante "Genera nuova ricetta".

Descrizione: L'utente può scegliere di generare una nuova ricetta se quella precedente non gli soddisfa.

Postcondizioni: Viene mostrato all'utente una nuova ricetta generata da *ChatGPT* [3].

2.2.4 Use Case 3: Visualizzazione dispensa

La Figura 2.5 riporta i sotto casi d'uso relativi alla visualizzazione della schermata dispensa.

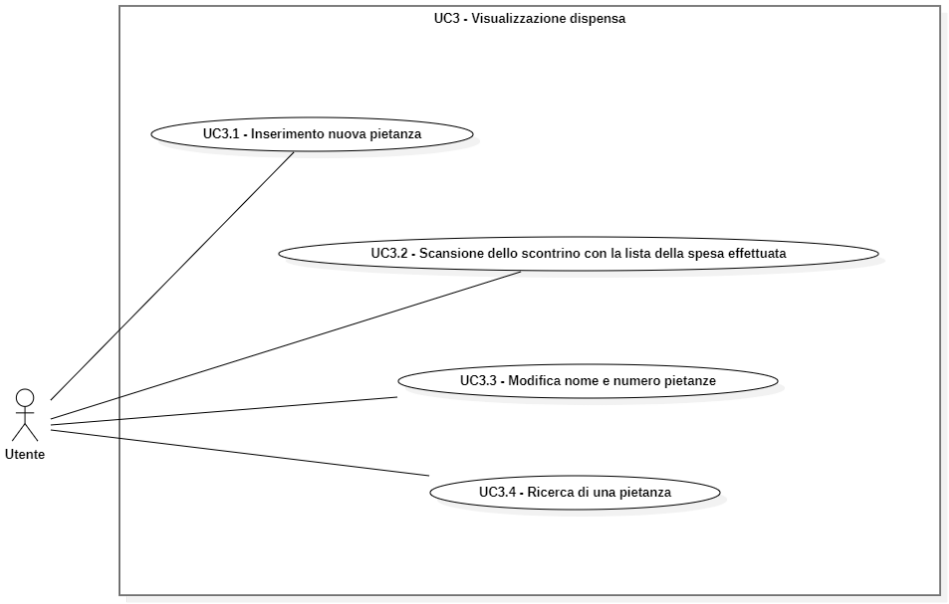


Figura 2.5: UC3: Visualizzazione schermata dispensa.

UC3.1: Inserimento nuovo ingrediente

Attori Principali: Utente.

Precondizioni: L'utente preme il pulsante per inserire un nuovo ingrediente.

Descrizione: L'utente può inserire un nuovo ingrediente nella dispensa.

Postcondizioni: L'ingrediente viene inserito nella dispensa aggiornando la quota stornata dell'utente e la cassa comune.

UC3.2: Scansione dello scontrino con la lista della spesa effettuata

Attori Principali: Utente.

Precondizioni: L'utente preme il pulsante per inserire la foto dello scontrino.

Descrizione: L'utente può inserire nuovi ingredienti semplicemente scansionando col telefono lo scontrino della spesa effettuata.

Postcondizioni: Gli ingredienti vengono inserite nella dispensa aggiornando la quota stornata dell'utente e la cassa comune.

UC3.3: Modifica nome e numero ingredienti

Attori Principali: Utente.

Precondizioni: L'utente preme l'icona edita in parte all'ingrediente per modificarne il nome e la sua quantità.

Descrizione: L'utente può modificare il nome e la quantità dell'ingrediente.

Postcondizioni: L'ingrediente viene aggiornato con i nuovi dati nella dispensa.

UC3.4: Ricerca di un ingrediente

Attori Principali: Utente.

Precondizioni: L'utente preme la casella di ricerca per cercare un ingrediente.

Descrizione: L'utente può cercare un ingrediente nella dispensa.

Postcondizioni: Gli ingredienti corrispondenti alla *query* di ricerca vengono mostrate all'utente.

2.2.5 Use Case 4: Registrazione pasto

La Figura 2.6 riporta i sotto casi d'uso relativi alla visualizzazione della schermata registrazione pasto.

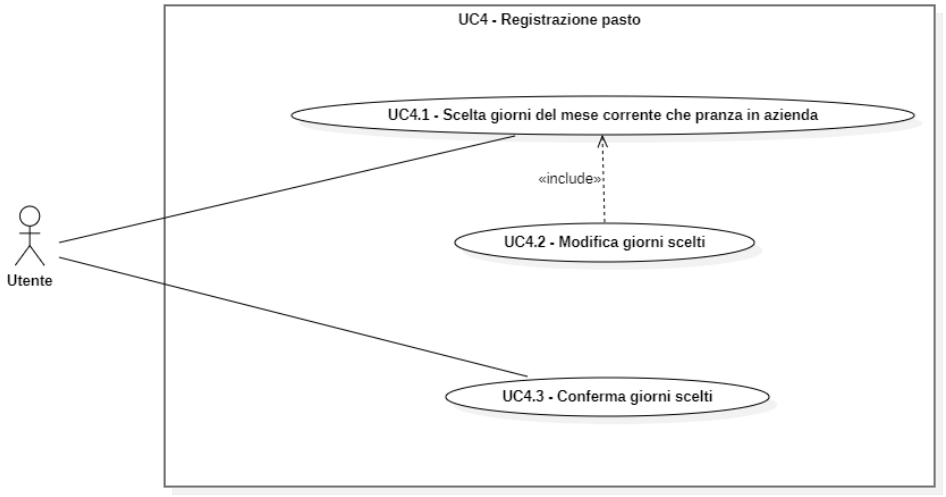


Figura 2.6: UC4: Visualizzazione schermata registrazione pasto.

UC4.1: Scelta giorni del mese corrente che pranza in azienda

Attori Principali: Utente.

Precondizioni: L'utente sceglie dal calendario i giorni che desidera pranzare in azienda nel mese corrente.

Descrizione: L'utente può scegliere dal calendario i giorni che desidera pranzare in azienda nel mese corrente.

Postcondizioni: Viene mostrata una lista di riepilogo di tutti i giorni selezionati.

UC4.2: Modifica giorni scelti

Attori Principali: Utente.

Precondizioni: L'utente deseleziona i giorni nel quale pranzare in azienda nel mese corrente.

Descrizione: L'utente può modificare i giorni selezionati per pranzare in azienda nel mese corrente.

Postcondizioni: Viene aggiornata la lista di riepilogo di tutti i giorni selezionati.

UC4.3: Conferma giorni scelti

Attori Principali: Utente.

Precondizioni: L'utente clicca il pulsante "Conferma" per confermare il tutto.

Descrizione: L'utente invia al sistema i giorni che desidera pranzare in azienda.

Postcondizioni: Viene mostrato un modale dove dice quanti soldi gli viene a costare il tutto.

2.2.6 Use Case 5: Visualizzazione transazioni

La Figura 2.7 riporta i sotto casi d'uso relativi alla visualizzazione della schermata transazioni.

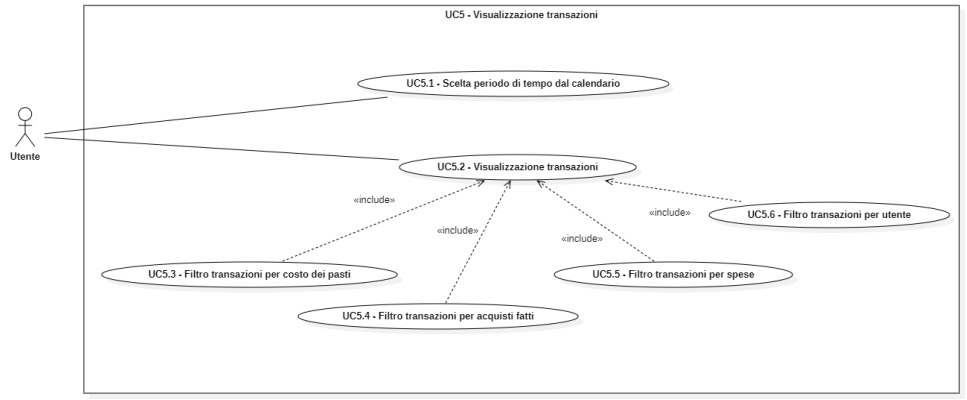


Figura 2.7: UC5: Visualizzazione schermata transazioni.

UC5.1: Scelta periodo di tempo dal calendario

Attori Principali: Utente.

Precondizioni: L'utente seleziona un periodo dal calendario.

Descrizione: L'utente seleziona un periodo dal calendario per visualizzare le transazioni effettuate.

Postcondizioni: Viene mostrato all'utente il periodo da lui selezionato.

UC5.2: Visualizzazione transazioni

Attori Principali: Utente.

Precondizioni: L'utente ha selezionato un periodo dal calendario.

Descrizione: All'utente vengono mostrate tutte le transazioni effettuate in quel periodo.

Postcondizioni: Viene mostrata una lista di riepilogo di tutte le transazioni effettuate in quel periodo.

UC5.3: Filtro transazioni per costo dei pasti

Attori Principali: Utente.

Precondizioni: L'utente ha selezionato il filtro "pasti".

Descrizione: Nella schermata l'utente può scegliere se filtrare le transazioni effettuate

nel periodo selezionato precedentemente per "pasti".

Postcondizioni: Viene mostrata una lista di riepilogo di tutte le transazioni effettuate per i pasti in quel periodo. Inoltre all'utente gli viene mostrato il numero di pasti sostenuti attualmente in quel mese e l'ammontare in euro.

UC5.4: Filtro transazioni per acquisti fatti

Attori Principali: Utente.

Precondizioni: L'utente ha selezionato il filtro "acquisti".

Descrizione: Nella schermata l'utente può scegliere se filtrare le transazioni effettuate nel periodo selezionato precedentemente per "acquisti".

Postcondizioni: Viene mostrata una lista di riepilogo di tutti gli acquisti effettuati in quel periodo.

UC5.5: Filtro transazioni per spese

Attori Principali: Utente.

Precondizioni: L'utente ha selezionato il filtro "spese".

Descrizione: Nella schermata l'utente può scegliere se filtrare le transazioni effettuate nel periodo selezionato precedentemente per "spese".

Postcondizioni: Viene mostrata una lista di riepilogo di tutte le transazioni relative alle spese effettuate nel periodo considerato, intendendo per spese i trasferimenti di denaro tra i vari membri dello staff.

UC5.6: Filtro transazioni per utente

Attori Principali: Utente.

Precondizioni: L'utente ha selezionato il filtro "utente".

Descrizione: Nella schermata l'utente può scegliere se filtrare le transazioni effettuate nel periodo selezionato precedentemente per "utente".

Postcondizioni: Viene mostrata una lista di riepilogo di tutte le transazioni effettuate dagli utenti selezionati in quel periodo.

2.2.7 Use Case 6: Debiti

La Figura 2.8 riporta i sotto casi d'uso relativi alla visualizzazione della schermata debiti.

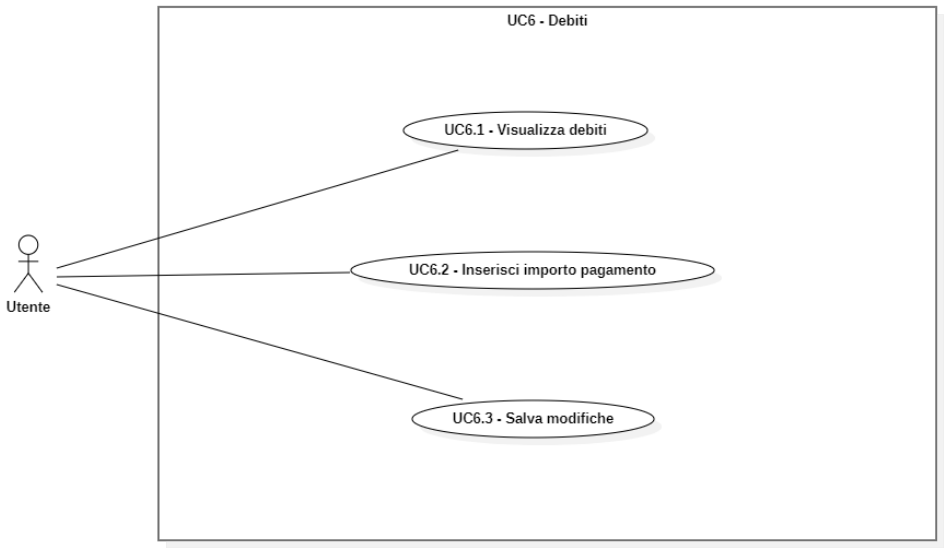


Figura 2.8: UC6: Visualizzazione schermata debiti.

UC6.1: Visualizza debiti

Attori Principali: Utente.

Precondizioni: L'utente apre la tab debiti.

Descrizione: L'utente in questa sezione può vedere tutti i debiti di tutti gli utenti che bisogna saldare entro il mese corrente.

Postcondizioni: Viene mostrato all'utente una lista di tutti gli utenti in debito.

UC6.2: Inserisci importo pagamento

Attori Principali: Utente.

Precondizioni: L'utente decide a quale utente sdebitarsi.

Descrizione: L'utente decide una somma da dare a quel determinato utente per sdebitarsi.

Postcondizioni: L'utente sceglie l'importo da dare a quello specifico utente.

UC6.3: Salva modifiche

Attori Principali: Utente.

Precondizioni: L'utente ha finito di sdebitare gli utenti.

Descrizione: L'utente salva le modifiche aggiornando i debiti rimanenti di tutti gli

utenti.

Postcondizioni: Il sistema ha registrato le modifiche.

2.2.8 Use Case 7: Profilo

La Figura 2.9 riporta i sotto casi d'uso relativi alla visualizzazione della schermata profilo.

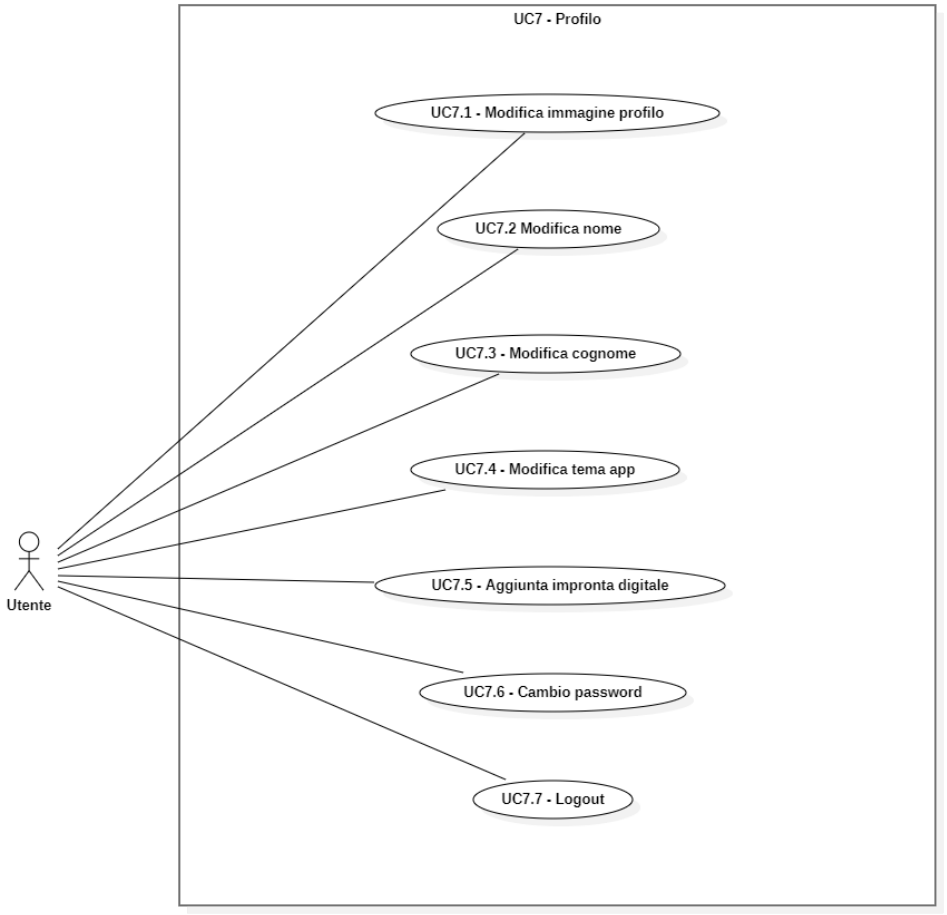


Figura 2.9: UC7: Visualizzazione schermata profilo.

UC7.1: Modifica immagine profilo

Attori Principali: Utente.

Precondizioni: L'utente fa tap sull'icona "modifica" posta in parte alla foto profilo.

Descrizione: L'utente può cambiare la propria foto profilo.

Postcondizioni: L'utente sceglie la foto che più gli piace per poi confermarla al sistema.

UC7.2: Modifica nome

Attori Principali: Utente.

Precondizioni: L'utente fa tap su "nome".

Descrizione: L'utente può cambiare il proprio nome che verrà visualizzato dagli altri utenti del sistema.

Postcondizioni: L'utente cambia il proprio nome e lo conferma al sistema.

UC7.3: Modifica cognome

Attori Principali: Utente.

Precondizioni: L'utente fa tap su "cognome".

Descrizione: L'utente può cambiare il proprio cognome che verrà visualizzato dagli altri utenti del sistema.

Postcondizioni: L'utente cambia il proprio cognome e lo conferma al sistema.

UC7.4: Modifica tema app

Attori Principali: Utente.

Precondizioni: L'utente fa tap su "tema".

Descrizione: L'utente può cambiare il tema dell'applicazione da chiaro a scuro e viceversa.

Postcondizioni: L'utente sceglie il tema che più gli piace.

UC7.5: Aggiunta impronta digitale

Attori Principali: Utente.

Precondizioni: L'utente fa tap su "impronta digitale".

Descrizione: L'utente può aggiungere la propria impronta digitale per accedere nell'applicazione.

Postcondizioni: L'utente aggiunge la propria impronta digitale nel sistema.

UC7.6: Cambio password

Attori Principali: Utente.

Precondizioni: L'utente fa tap su "cambio password".

Descrizione: L'utente può modificare la propria password con una nuova scelta da

lui.

Postcondizioni: L'utente inserisce e conferma la nuova password.

UC7.7: Logout

Attori Principali: Utente.

Precondizioni: L'utente fa tap su "*logout*".

Descrizione: L'utente può eseguire il *logout* dall'applicazione.

Postcondizioni: L'utente viene reindirizzato nella schermata di login.

2.2.9 Use Case 8: Lista della Spesa

La Figura 2.10 riporta i sotto casi d'uso relativi alla visualizzazione della schermata Lista della Spesa.

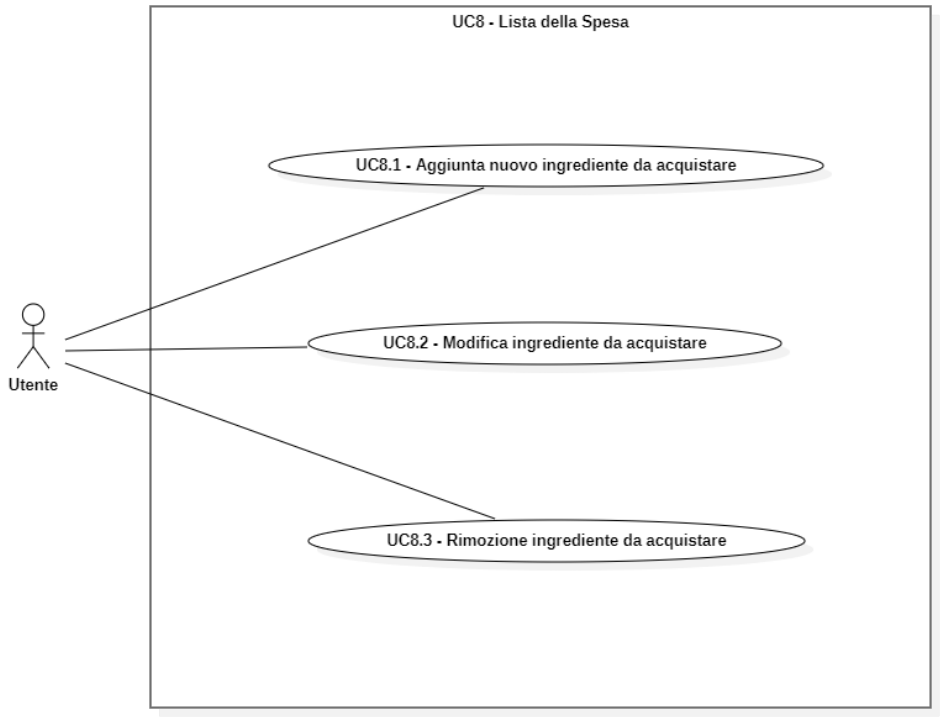


Figura 2.10: UC8: Visualizzazione schermata Lista della Spesa.

UC8.1: Aggiunta nuovo ingrediente da acquistare

Attori Principali: Utente.

Precondizioni: L'utente inserisce il nome e la quantità dell'ingrediente da acquistare.

Descrizione: L'utente può aggiungere nuovi ingredienti da acquistare al supermercato.

Postcondizioni: Il sistema salva l'inserimento effettuato dall'utente.

UC8.2: Modifica ingrediente da acquistare

Attori Principali: Utente.

Precondizioni: L'utente seleziona l'ingrediente che vuole modificare.

Descrizione: L'utente può decidere di modificare o il nome o la quantità dell'ingrediente da acquistare.

Postcondizioni: Il sistema salva le modifiche effettuate dall'utente.

UC8.3: Rimozione ingrediente da acquistare

Attori Principali: Utente.

Precondizioni: L'utente seleziona l'ingrediente che vuole rimuovere.

Descrizione: L'utente può decidere di rimuovere l'ingrediente da acquistare.

Postcondizioni: Il sistema rimuove l'ingrediente.

Nello scenario dell'amministratore del sistema le cose cambiano leggermente sulla schermata del profilo. Nella figura 2.11 riporta i casi d'uso aggiuntivi relativi alla visualizzazione della schermata profilo da parte dell'amministratore.

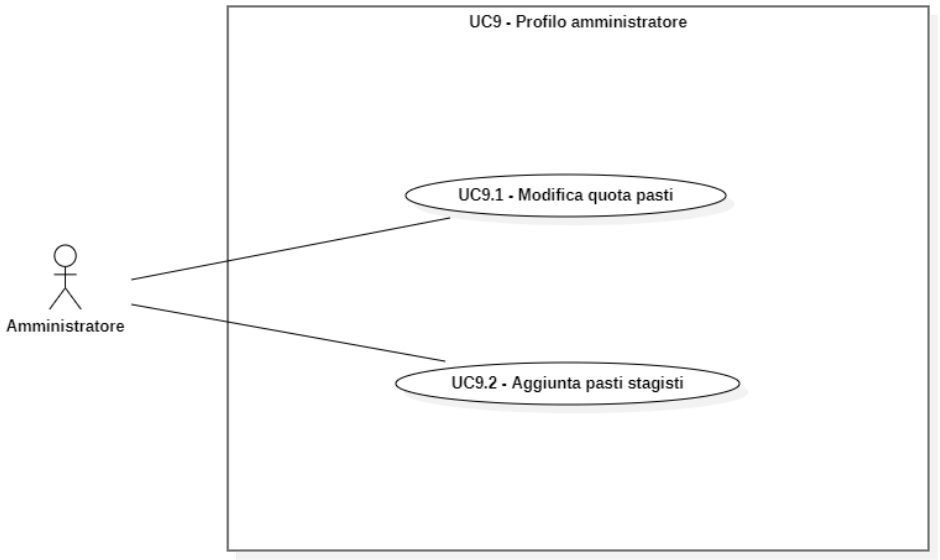


Figura 2.11: UC9: Visualizzazione schermata profilo da parte dell'amministratore.

UC9.1: Modifica quota pasti

Attori Principali: Amministratore.

Precondizioni: L'amministratore fa tap su "quota pasti".

Descrizione: L'amministratore può cambiare la quota dei pasti che si tengono in azienda.

Postcondizioni: L'amministratore sceglie la quota.

UC9.2: Aggiunta pasti stagisti

Attori Principali: Amministratore.

Precondizioni: L'amministratore fa tap su "pasti stagisti".

Descrizione: L'amministratore può aggiungere il numero di pasti che gli stagisti consumeranno in azienda nel mese corrente.

Postcondizioni: L'amministratore sceglie i giorni e la quantità di pasti che verranno consumati come descritto nello Use Case 4.

2.3 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto, è stata stilata la tabella che traccia i requisiti in rapporto agli use case.
Alcuni requisiti sotto elencati sono stati implementati anche da un'altra stagista in quanto dovevamo realizzare entrambi la stessa applicazione ma con alcune differenze nei requisiti.
Il codice dei requisiti è così strutturato **R(F/Q/V)(N/D/Z)(E)** dove:

- **R:** requisito
- **F:** funzionale
- **Q:** qualitativo
- **V:** di vincolo
- **N:** obbligatorio (necessario)
- **D:** desiderabile
- **Z:** opzionale
- **E:** requisito in comune con la stagista

2.3.1 Requisiti Funzionali

Nella Tabella 2.1 sono elencati i requisiti funzionali individuati durante l'attività di analisi a partire dai casi d'uso.

Tabella 2.1: Tabella del tracciamento dei requisti funzionali

Requisito	Descrizione	Fonte
RFNE-1	L'utente deve poter accedere all'applicazione per poter usufruire del servizio	UC1.1
RFNE-2	L'utente deve poter vedere il resoconto mensile	UC2.1
RFNE-3	L'utente deve poter visualizzare la cassa comune per vedere quanto denaro è stato risparmiato	UC2.2
RFNE-4	L'utente deve poter visualizzare le quote stornate di tutti gli utenti	UC2.3
RFN-5	L'utente deve poter vedere la ricetta del giorno generata da *ChatGPT* [3]	UC2.4
RFNE-6	L'utente deve poter inserire nella dispensa uno o più ingredienti	UC3.1
RFN-7	L'utente deve poter modificare il nome e la quantità degli ingredienti nella dispensa	UC3.3
RFNE-8	L'utente deve poter scegliere i giorni che desidera pranzare in azienda	UC4.1
RFNE-9	L'utente deve poter visualizzare tutte le transazioni eseguite nel mese corrente	UC5.2
RFNE-10	L'utente deve poter visualizzare tutti gli utenti che sono in debito	UC6.1
RFN-11	L'utente deve poter cambiare la propria password per accedere al servizio	UC7.6
RFNE-12	L'utente deve poter eseguire il *logout* dal sistema	UC7.7
RFNE-13	L'amministratore deve poter modificare il costo dei pasti in azienda	UC9.1
RFNE-14	L'amministratore deve poter aggiungere i pasti degli stagisti che desiderano pranzare in azienda	UC9.2

2.3.2 Requisiti qualitativi

Nella Tabella 2.2 sono elencati i requisiti qualitativi individuati durante l'attività di analisi a partire dalle discussioni con il proponente.

Tabella 2.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Fonte
RQD-1	L'utente può eseguire l'accesso con usando il proprio account *Microsoft*	UC1.2
RQD-2	L'utente può scansionare lo scontrino con la lista della spesa effettuata	UC3.2
RQD-3	L'utente può modificare la propria immagine del profilo	UC7.1
RQD-4	L'utente può modificare il proprio nome	UC7.2
RQD-5	L'utente può modificare il proprio cognome	UC7.3
RQZ-1	L'utente può effettuare la ricerca di un ingrediente nella dispensa	UC3.4
RQZ-2	L'utente può filtrare le transazioni per costo dei pasti	UC5.3
RQZ-3	L'utente può filtrare le transazioni per acquisti fatti	UC5.4
RQZ-4	L'utente può filtrare le transazioni per spese	UC5.5
RQZ-5	L'utente può filtrare le transazioni per utente	UC5.6
RQZ-6	L'utente può modificare il tema dell'app da chiaro a scuro e viceversa	UC7.4
RQZ-7	L'utente può registrare la propria impronta digitale per accedere al servizio	UC7.5
RQZ-8	L'utente può aggiungere nella lista della spesa un nuovo ingrediente da acquistare	UC8.1
RQZ-9	L'utente può modificare dalla lista della spesa l'ingrediente da acquistare	UC8.2
RQZ-10	L'utente può rimuovere dalla lista della spesa un ingrediente	UC8.3
RQZ-11	L'applicazione deve funzionare anche usando il Browser da PC	Proponente

2.3.3 Requisiti di vincolo

Nella Tabella 2.3 sono elencati i requisiti di vincolo individuati durante l'attività di analisi.

Tabella 2.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Fonte
RVNE-1	L'applicazione deve funzionare sui dispositivi mobili (Android e iOS)	Proponente

Capitolo 3

Progettazione

Il presente capitolo descrive la fase di progettazione analizzando in modo approfondito le varie tecnologie usate

3.1 Tecnologie e strumenti

Qui di seguito viene fornita un'anteprima delle tecnologie e degli strumenti impiegati. È importante notare che, a meno che non sia specificato diversamente, la decisione di adottare una particolare tecnologia o strumento è stata influenzata da requisiti o pratiche aziendali.

3.1.1 Tecnologie utilizzate

Dart

Dart (Figura 3.1) è un linguaggio di programmazione orientato agli oggetti per il web, completamente *open source*[g] [22] e sviluppato da Google.
Lo scopo di Dart è quello di aiutare lo sviluppatore a costruire moderne applicazioni web, server e mobile *cross-platform*[g] [5] con Flutter. Per fare ciò, Dart offre tutti gli attributi necessari per un linguaggio moderno e generico, una *Virtual Machine* per la compilazione del codice Dart, librerie di base e repository di gestione dei pacchetti per l'aggiunta di funzionalità.
Questo linguaggio offre inoltre i seguenti benefici:

- *Produttività*: Sintassi chiara e concisa e cicli di sviluppo quasi istantanei;
- *Velocità*: Tempi di avvio ed esecuzione eccellenti e prevedibili anche su dispositivi con poche risorse hardware;
- *Portabilità*: Dart è un linguaggio flessibile e può essere eseguito su molteplici sistemi operativi desktop (Windows, macOS, Linux) e mobile (Android e iOS) senza alcuna limitazione;
- *Reactive Programming*: La natura reattiva di Dart è dovuta alla rapida assegnazione degli oggetti e al supporto della programmazione asincrona. Ciò rende Dart un buon candidato per creare un'applicazione interattiva o ad alta intensità di dati.

Proprio per questi benefici, Dart viene utilizzato sempre di più da diverse aziende e progetti di grande calibro [6].

Figura 3.1: Logo del linguaggio Dart.

Flutter

Flutter (Figura 3.3) è un framework open-source ideato direttamente da Google per rispondere alla continua evoluzione dei dispositivi mobile. L'obiettivo è quello di utilizzare una singola *codebase* per la programmazione informatica di app *cross-platform* che girano in maniera nativa su dispositivi Android e sui dispositivi iOS e, inoltre, con le ultime versioni anche Web e Desktop.
Le prestazioni native di Flutter derivano da uno substrato scritto in C/C++, o meglio chiamato *Flutter Engine*, e uno strato scritto in Dart.
La sua architettura è descritta nella Figura 3.2 :

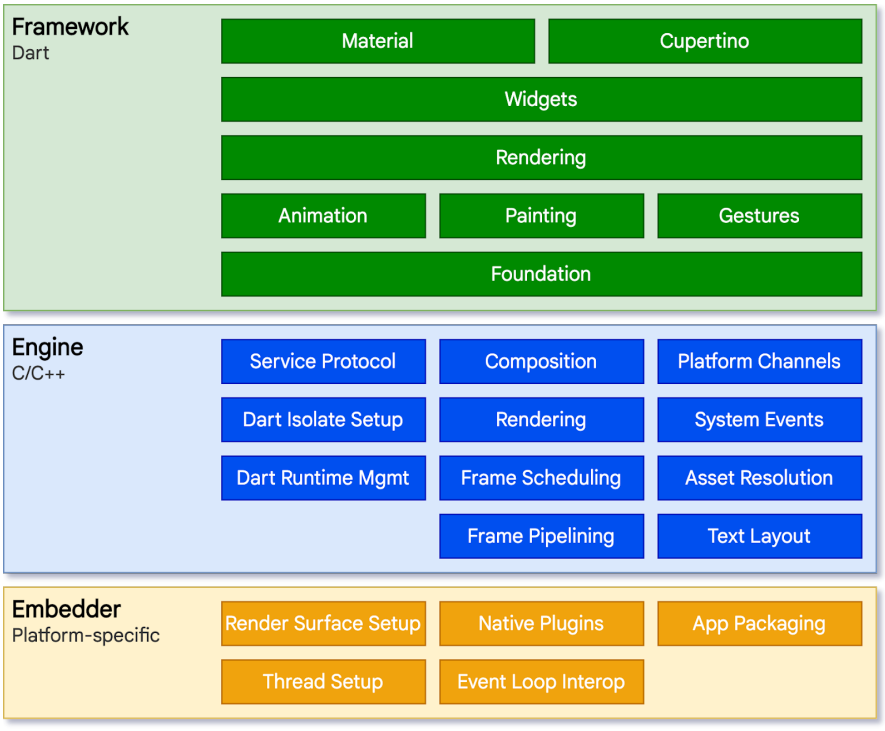


Figura 3.2: Architettura del linguaggio Flutter [2].

Di seguito vengono spiegati i 3 livelli:

- *Embedder*: fornisce un punto di ingresso al sistema operativo sottostante. Questo livello è scritto nel linguaggio di programmazione appropriato per la piattaforma su cui gira, attualmente Java e C++ per Android, *Swift* per iOS e macOS, e C++ per Windows e Linux; in altre parole consente alle app scritte in Flutter di poter girare su qualsiasi sistema operativo;

- *Flutter Engine*: è il cuore di Flutter, scritto interamente in C/C++, si occupa di fornire supporto a basso livello per il rendering dei widget dell'app Flutter. L'*Engine* è esposto al primo livello attraverso la libreria dart:iu che espone le funzioni primitive di basso livello come quelle per il *text-rendering* e la gestione I/O;

- *Dart*: lavora a stretto contatto con il *Flutter Engine* con il quale è in grado di fornire la funzione di "*Hot-Reload*", cioè ogni modifica del codice viene mandata immediatamente nell'applicazione in modo da visualizzare all'istante i cambiamenti senza un riavvio completo.

I vantaggi nell'utilizzo di Flutter rispetto ai vecchi tools sono molteplici:

- *Single Codebase*: Flutter garantisce lo sviluppo di un'applicazione sfruttando una singola base di codice. In parole povere, con lo stesso codice con cui si crea un'applicazione iOS si può generare un applicazione Android, Web o Desktop;

- *Hot-Reload e velocità di sviluppo*: Questa sua caratteristica permette ai web developer di poter osservare, istantaneamente, le modifiche effettuate sul codice, garantendo maggiori performance di sviluppo e tempi più snelli anche di debugging;

- *Ecosistema ricco di Widget*: Possiamo descrivere Flutter come un intero widget.

  Tutto al suo interno è un widget e i widget possono essere sia classi che funzioni: questo permette, infatti, all'applicazione dei principi della programmazione ad oggetti nello sviluppo della UI.

  Ogni widget è modificabile a proprio piacimento, ad esempio si possono includere dei widget dentro altri in modo da garantire specifiche funzioni su diverse risoluzioni dello schermo.

  Flutter offre una miriade di *UI* widget che rispettano tutti gli attuali princìpi chiave del design di applicativi sia web che mobile, offrendo sempre ottime performance.

  Il suo SDK è in grado di garantire tutto questo in quanto è l'unico che non utilizza alcun Bridge col linguaggio javascript per la creazione dei componenti, ma lavora facendo comunicare direttamente il core con il dispositivo aumentandone drasticamente le prestazioni.

  I widget in Flutter sono organizzati attraverso un albero, in cui ognuno di loro può contenerne altri e così via. Sarà il programmatore informatico a dover fare attenzione all'organizzazione dei widget nella propria App, cercando di ingegnerizzare al meglio lo sviluppo;

- *Community*: Negli ultimi anni Flutter è cresciuto tantissimo di popolarità. Questo ha favorito la crescita di una sempre più forte community attorno ad esso che

contribuisce costantemente al miglioramento del *framework*, rendendo molto semplice la risoluzione di problematiche comuni;

- *MVP*: Flutter può essere un fattore chiave per tutte quelle startup che devono creare un MVP.

  Un MVP creato con Flutter è compatibile con diversi dispositivi da iOS ad Android, per finire a Desktop e Web. Sicuramente, questo è un fattore attrattivo per molti investitori, potendo facilitarli nella ricerca e prova del progetto di business creato [12].

Figura 3.3: Logo del linguaggio Flutter.

Firestore

Firestore (Figura 3.4) è un database *NoSQL*[g] [21] documentale che offre un'archiviazione dati altamente scalabile e flessibile, progettato per essere utilizzato in applicazioni web e mobili. Alcune delle sue caratteristiche principali includono:

- *Struttura dei dati*: archivia i dati in forma di documenti *JSON* all'interno di collezioni. Ogni documento contiene campi chiave-valore, e le collezioni sono simili a tabelle in un database relazionale;
- *Scalabilità*: è altamente scalabile, il che significa che può gestire grandi quantità di dati e traffico senza problemi. Puoi facilmente scalare il tuo database al ritmo delle esigenze della tua applicazione;
- *Real-time synchronization*: supporta la sincronizzazione in tempo reale dei dati. Questo significa che le modifiche apportate ai dati da una parte dell'applicazione vengono immediatamente riflesse in tutte le altre parti dell'applicazione che stanno visualizzando quei dati;
- *Accesso sicuro*: offre funzionalità avanzate di controllo degli accessi e di autenticazione, che consentono di gestire chi può accedere ai dati e quali azioni possono essere eseguite su di essi;
- *Indicizzazione*: fornisce indicizzazione automatica dei dati per query efficienti. Puoi eseguire query complesse sui dati senza dover creare manualmente indici;
- *Integrazione con altre tecnologie Google*: è facilmente integrabile con altre soluzioni come *Firebase*, *Cloud Functions* e *Cloud Storage* [11].

Figura 3.4: Logo del database Firestore.

Firebase Storage

Firebase Storage (Figura 3.5) consente agli sviluppatori di memorizzare e gestire facilmente file multimediali come immagini, video, documenti e altri tipi di dati non strutturati all'interno dell'infrastruttura *cloud* di Google. Alcune delle sue caratteristiche principali sono la facilità di utilizzo, opzioni di autorizzazione e autenticazione avanzate, scalabilità e integrazione con altre funzionalità Firebase [10].

Figura 3.5: Logo di Firebase Storage.

Firebase Authentication

Firebase Authentication (Figura 3.6) è una funzionalità di autenticazione utente fornita da Firebase come servizio *backend*. Si tratta di un sistema di autenticazione basato su token che offre un'integrazione semplice con la maggior parte delle piattaforme.
Uno dei principali vantaggi è la possibilità di creare ruoli differenziati per gli utenti, assegnando diversi permessi in base al ruolo. Altre caratteristiche interessanti includono un'agevole integrazione con le *API* di *Firebase*, che rende questa funzionalità di autenticazione facilmente implementabile in brevissimo tempo.
Firebase Authentication sfrutta SDK e librerie dell'interfaccia utente pronte all'uso per autenticare gli utenti e creare esperienze personalizzate su tutti i dispositivi.
Nel dettaglio, si basa sui *JSON Web Token (JWT)*[g] [19] per autenticare gli utenti, supportando metodi come email/password ed email/link. Consente inoltre l'autenticazione via numero di telefono e l'accesso tramite provider di identità popolari come Google e Github [9].

Figura 3.6: Logo di Firebase Authentication.

OCR

Tecnicamente parlando, l'*Optical Character Recognition* è il processo di conversione di un testo stampato in un formato che può essere facilmente modificato e conservato da un computer o dispositivo. La tecnologia *OCR* richiede l'utilizzo combinato di *hardware* (scanner, fotocamera digitale o smartphone) e software (programmi ad hoc o dotati di funzionalità *OCR*). Un'*OCR* funziona nel seguente modo:

- *Acquisizione dell'Immagine*: Il processo inizia con l'acquisizione di un'immagine contenente testo. Questa immagine può provenire da scanner, fotocamere digitali o altri dispositivi di acquisizione di immagini;
- *Pre-elaborazione dell'Immagine*: L'immagine acquisita può essere soggetta a pre-elaborazione per migliorare la qualità. Questa fase può includere operazioni come la riduzione del rumore, la normalizzazione dell'illuminazione e la correzione della distorsione;
- *Segmentazione*: In questa fase, l'immagine viene suddivisa in regioni contenenti singoli caratteri o linee di testo. La segmentazione è un passo critico per identificare correttamente ciascun carattere nell'immagine;
- *Riconoscimento dei Caratteri*: Ogni segmento contenente un carattere viene analizzato per determinare quale carattere rappresenta. Ciò può essere fatto attraverso l'uso di algoritmi di riconoscimento dei caratteri, che possono essere basati su modelli di machine learning, reti neurali o altre tecniche avanzate;
- *Post-elaborazione*: Dopo il riconoscimento dei caratteri, potrebbe essere eseguita una fase di post-elaborazione per migliorare la precisione. Questo può includere la correzione degli errori rilevati e il miglioramento della formattazione;
- *Produzione del Testo Riconosciuto*: Infine, il testo riconosciuto viene prodotto in un formato digitale, come un documento di testo o un file PDF cercabile.

Nel nostro caso facciamo affidamento a delle librerie già precompilate che usano tale tecnologia. La logica che ci sta dietro a queste librerie è: dato un blocco di frasi è possibile estrarre le singole linee e a sua volta estrarre le singole parole. Tale logica viene riassunta nella Figura 3.7 [23]

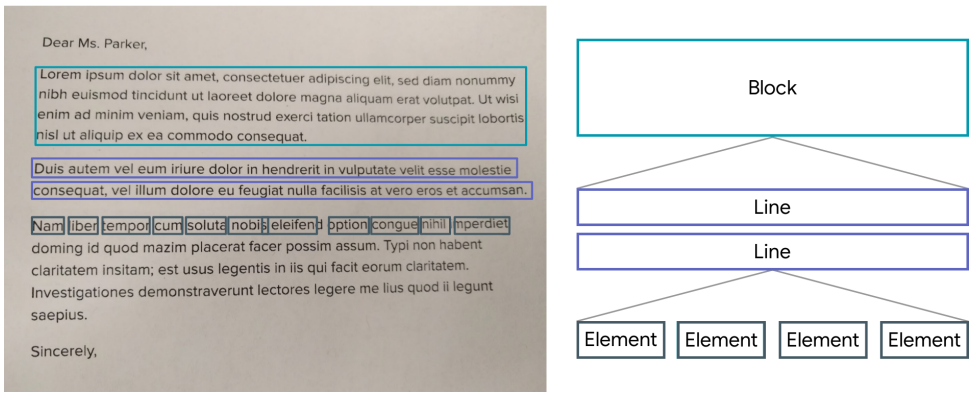


Figura 3.7: Funzionamento librerie OCR.

SSO

Il *Single Sign-On (SSO)* è un processo di autenticazione che consente ad un utente di accedere a più applicazioni con un solo set di credenziali di accesso. Con il *Single Sign-On* l'utente effettua il login una sola volta e ottiene l'accesso a diverse applicazioni senza la necessità di immettere nuovamente le credenziali di accesso in ogni applicazione. L'autenticazione *SSO*, quindi, facilita l'utilizzo delle risorse di rete senza soluzione di continuità. Idealmente, il *Single Sign-On* viene utilizzato con altre tecniche di autenticazione, ad esempio *smart card* e *token* con password monouso. A livello tecnico funziona nel seguente modo:

- Un'applicazione centrale, tipicamente un *Identity Provider (IdP)*[g] [18], si occupa di autenticare gli utenti. L'*IdP* memorizza le credenziali degli utenti (nome utente e password) in un database centrale;
- Le altre applicazioni, chiamate *Service Provider (SP)*[g] [18], si affidano all'*IdP* per l'autenticazione. Non hanno bisogno di memorizzare proprie credenziali utente;
- Quando un utente accede a uno *SP*, quest'ultimo reindirizza l'utente all'IdP per l'autenticazione;
- L'utente inserisce i propri dati di login nella pagina di autenticazione dell'*IdP*;
- Se l'autenticazione ha esito positivo, l'*IdP* genera un token che certifica l'identità dell'utente;
- Il token viene ritrasmesso allo *SP*, che può quindi riconoscere l'utente e fornirgli l'accesso ai propri servizi;
- Nel frattempo l'*IdP* mantiene traccia delle sessioni dell'utente nei vari *SP*, in modo che l'utente non debba ripetere l'accesso.

Nelle due immagini (Figura 3.8 e Figura 3.9) viene esplicitata questa procedura.

Figura 3.8: L'utente richiede l'accesso [14].

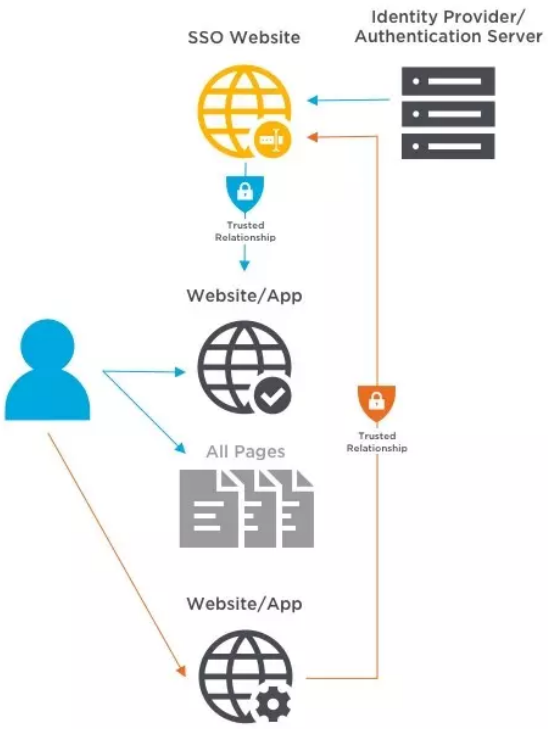


Figura 3.9: All'utente viene concesso l'accesso e quindi fa richiesta di accesso ad un nuovo sito [14].

Nel nostro caso abbiamo fatto affidamento all'SSO di *Microsoft* in quanto gli account aziendali dell'azienda sono stati creati con tale *SP* [24].

Riconoscimento biometrico

Il riconoscimento dell'identità di una persona tramite il suo volto o impronta digitale come avviene con *Touch ID* / *Face ID* su iOS o le *API* biometriche su Android fornisce una certezza sulla sicurezza che metodi tradizionali quali username e password non erano in grado di dare. Mentre infatti con nome utente e password si ha solo la consapevolezza che qualcuno è in possesso di quelle informazioni, con i dati biometrici come l'impronta digitale o la conformazione del viso si ottiene una verifica molto più solida dell'identità.
Il riconoscimento biometrico si basa su sistemi *hardware* per l'acquisizione dei dati integrati da componenti software in grado, mediante algoritmi matematici, di analizzare i dati acquisiti e ricostruire l'identità di un individuo e riconoscerlo.
Nell'app che si doveva sviluppare era necessario proteggere l'accesso all'applicazione e la condivisione dei contenuti al suo interno. Ci si è quindi affidati al package *local-auth* che consente di gestire l'autenticazione biometrica tramite volto o impronta sul dispositivo interfacciandosi direttamente con le *API* nativamente implementate nel sistema operativo (*API* biometriche su Android e *Local Authentication* su iOS) garantendo in tal modo un livello di sicurezza elevato.

3.1.2 Ambiente di lavoro e strumenti utilizzati

Mac OS Ventura

Mac OS Ventura (Figura 3.10) è l'ultima versione del sistema operativo di Apple per MacBook. Poichè il PC fornito dall'azienda a dipendenti e stagisti è un MacBook si è deciso di utilizzare questo sistema operativo [20].

Figura 3.10: Logo di Mac OS Ventura.

Visual Studio Code

Visual Studio Code (Figura 3.11) è un editor di codice cross-platform che permette di:

- Evidenziare la sintassi per ciascun linguaggio di programmazione;
- Avere un supporto integrato per il debugging;
- Avere un supporto integrato a Git;
- Effettuare refactoring del codice;

- Scaricare ed installare estensioni per ampliare le funzionalità del programma [30].

Figura 3.11: Logo di Visual Studio Code.

Hackolade

Hackolade (Figura 3.12) è uno strumento di modellazione dei dati per database *NoSQL*, formati di storage, *API REST* e *JSON* in RDBMS. Questo software fornisce un ambiente visivo per creare, esplorare e documentare i modelli di dati. *Hackolade* offre una serie di funzionalità utili per i suoi utenti. Alcuni esempi includono:

- Visualizzazione gerarchica dello schema delle entità (collezioni/tabelle/nodi);
- Funzionalità di copia e incolla degli attributi;
- Generazione dinamica di un documento di dati *JSON* di esempio quando si inseriscono dati nel campo Campione del Pannello Proprietà;
- Capacità di eseguire il reverse engineering dei dati esistenti [17].

Figura 3.12: Logo di Hackolade.

3.1.3 Piattaforme di Sviluppo e Collaborazione

Git

Git (Figura 3.13) è un sistema di controllo delle versioni che consente a più persone di collaborare efficacemente su un progetto, tenendo traccia delle modifiche apportate ai file nel corso del tempo. Il concetto chiave dietro Git è la registrazione delle modifiche (commit) e la creazione di un registro completo della storia del progetto. Ogni volta che si apporta una modifica a un file, si crea un nuovo commit che contiene una descrizione delle modifiche e un riferimento al commit precedente. Questo consente di tenere traccia di chi ha fatto cosa e quando.
Di seguito vengono elencati i vantaggi dell'utilizzo di *Git*:

- *Controllo delle Versioni*: consente di tenere traccia di tutte le modifiche apportate ai file di un progetto nel tempo. Questo è utile per ripristinare versioni precedenti, confrontare le modifiche tra diverse versioni e risolvere i conflitti;
- *Collaborazione*: rende la collaborazione tra team di sviluppo molto più semplice. Diversi sviluppatori possono lavorare simultaneamente su diverse funzionalità o correzioni di bug e poi unire le loro modifiche in un unico ramo principale (branch) del progetto;
- *Rami (Branches)*: consente di creare rami separati del progetto per sviluppare nuove funzionalità o risolvere problemi senza influire sul ramo principale. Questo facilita lo sviluppo parallelo e la sperimentazione;
- *Staging Area*: ha una "staging area" in cui è possibile preparare e revisionare le modifiche prima di effettuare un commit. Questo consente di selezionare esattamente quali modifiche includere in un commit;
- *Distribuzione Distribuita*: è un sistema di controllo versione distribuito, il che significa che ogni copia del repository contiene l'intera storia del progetto. Ciò rende più facile lavorare offline e distribuire copie del repository in più posizioni;
- *Velocità*: è noto per essere molto veloce nell'eseguire operazioni come il commit, il merge e il recupero di versioni precedenti dei file;
- *Community e Supporto*: è estremamente popolare e ha una vasta community di sviluppatori. Ciò significa che c'è un sacco di supporto, documentazione e strumenti disponibili per aiutarti a utilizzarlo efficacemente;
- *Open Source*: è un software open source, il che significa che è gratuito da utilizzare, può essere personalizzato e si può contribuire al suo sviluppo se lo si desidera.

Nello sviluppo di RiskAPP viene adottata la tecnica *Git Feature Branch Workflow*, che favorisce la creazione di un nuovo branch ogni volta che si deve aggiungere una nuova funzionalità all'app. All'interno del nuovo *branch* appena creato, verrà sviluppata la funzionalità desiderata. Una volta completata e funzionante, verrà eseguito il *Merge* nel *branch* principale (main) [15].

Figura 3.13: Logo di Git.

GitHub

GitHub (Figura 3.14) è una piattaforma web basata su *Git* che offre un servizio di *hosting* per *repository* di codice sorgente e collaborazione nello sviluppo di software. Di seguito vengono elencate alcune delle caratteristiche principali di GitHub:

- *Controllo di versione*: utilizza il sistema di controllo di versione distribuito *Git* per tenere traccia delle modifiche apportate al codice sorgente. Questo consente agli sviluppatori di gestire e registrare le variazioni nel codice nel tempo;
- *Repository*: i progetti su GitHub sono ospitati in repository, che sono come contenitori digitali per il codice sorgente. Ogni *repository* può includere file, cartelle e documentazione;
- *Collaborazione* facilita la collaborazione tra sviluppatori permettendo loro di lavorare contemporaneamente su progetti, tenere traccia delle modifiche apportate da altri utenti e risolvere conflitti di *merge*;
- *Issue Tracking*: gli utenti possono aprire "*issue*" per segnalare bug, proporre nuove funzionalità o discutere questioni relative al progetto. Queste *issue* possono essere assegnate agli sviluppatori, discusse e monitorate nel tempo;
- *Pull Requests*: i pull request consentono agli sviluppatori di contribuire alle modifiche del codice sorgente di un repository. Questi possono essere revisionati da altri collaboratori prima di essere incorporati nel progetto principale;
- *Integrazione continua*: offre strumenti di integrazione continua che consentono di automatizzare i processi di build, test e distribuzione del software;
- *Github Actions*: è una funzionalità che consente di automatizzare il flusso di lavoro di sviluppo, come la compilazione e i test del codice, e di eseguire attività personalizzate in risposta a eventi specifici nel *repository*;
- *GitHub Pages*: offre la possibilità di creare siti web statici direttamente dai repository, rendendo più facile la condivisione di documentazione e progetti online;
- *Community*: è una piattaforma sociale in cui gli sviluppatori possono seguire altri utenti, "stellare" repository interessanti e partecipare a discussioni nella comunità *open-source* [16].

Figura 3.14: Logo di GitHub.

Slack

Slack (Figura 3.15) è un'app di messaggistica per le aziende che collega le persone alle informazioni di cui hanno bisogno. Riunendo le persone e permettendo loro di lavorare come un unico team consolidato. Di seguito vengono elencate le caratteristiche di *Slack*:

- *Connesso*: semplifica il contatto con i colleghi, permettendo di inviare messaggi a chiunque all'interno o all'esterno dell'organizzazione. Gli utenti possono lavorare in aree dedicate chiamate canali che riuniscono le persone e le informazioni necessarie;
- *Flessibile*: consente di lavorare in modo asincrono. Quando il lavoro è organizzato in canali, si può accedere alle informazioni di cui si ha bisogno quando si desidera;
- *Inclusivo*: tutti i membri di un'organizzazione hanno accesso alle stesse informazioni ricercabili e condivise. Quando i team lavorano insieme nei canali, le informazioni possono essere condivise con tutti i membri contemporaneamente, consentendo ai team di restare allineati e prendere decisioni più rapidamente.

Questo software è stato utilizzato per comunicare con il tutor aziendale da remoto [26].

Figura 3.15: Logo di Slack.

Figma

Figma (Figura 3.16) è un'applicazione di progettazione e prototipazione basata su *cloud* utilizzata principalmente da designer e team di progettazione per creare, collaborare e condividere progetti di interfaccia utente (*UI*) ed esperienza utente (*UX*). È ampiamente utilizzato per la progettazione di siti web, applicazioni mobili, software e altri prodotti digitali. Questo strumento è servito a mostrare al tutor aziendale e di concordare con lui l'interfaccia da realizzare [7].

Figura 3.16: Logo di Figma.

3.2 Schema del Database

Nella figura 3.17 viene riportato lo schema del database utilizzato dall'applicazione.

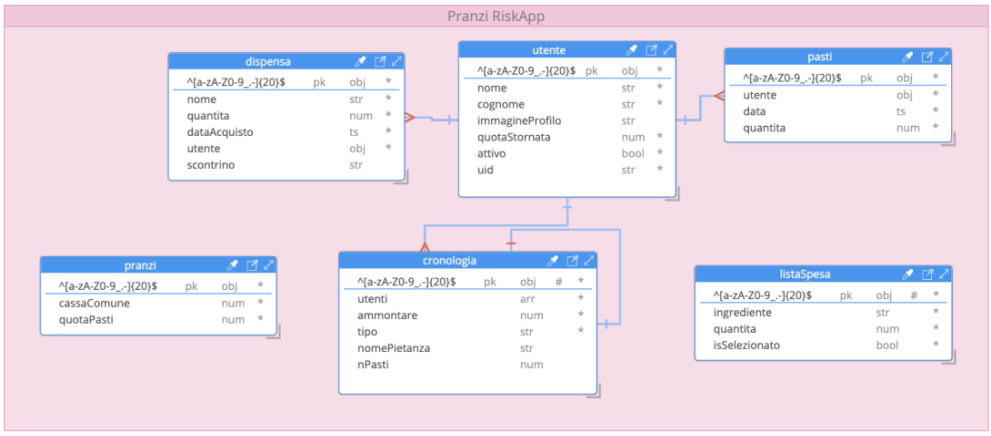


Figura 3.17: Schema database.

Abbiamo per prima cosa la collezione **utente** con i seguenti campi:

- *nome*: è un campo stringa obbligatorio e rappresenta il nome dell'utente;
- *cognome*: è un campo stringa obbligatorio e rappresenta il cognome dell'utente;
- *immagineProfilo*: è un campo stringa e rappresenta l'immagine del profilo dell'utente;
- *quotaStornata*: è un campo numerico obbligatorio e rappresenta la quota stornata dell'utente;
- *attivo*: è un campo booleano obbligatorio e serve all'amministratore dell'app per confermare o meno la registrazione dell'utente nel sistema;
- *uid*: è un campo stringa obbligatorio che rappresenta l'identificativo univoco delle credenziali di accesso (email e password) presenti in *Firebase Authentication* [9].

A seguire abbiamo la collezione **dispensa** con i seguenti campi:

- *nome*: è un campo stringa obbligatorio e rappresenta il nome della pietanza nella dispensa;
- *quantita*: è un campo numerico obbligatorio e rappresenta la quantità della pietanza disponibile nella dispensa;
- *dataAcquisto*: è un campo timestamp obbligatorio e rappresenta la data di acquisto della singola pietanza;
- *utente*: è un campo object obbligatorio e rappresenta l'utente che ha acquistato quella determinata pietanza;
- *scontrino*: è un campo stringa e viene utilizzato per salvare il percorso dell'immagine dello scontrino (se utilizzata la scansione OCR [23]).

Abbiamo poi la collezione **pasti** con i seguenti campi:

- *utente*: è un campo object obbligatorio e rappresenta l'utente che ha deciso di pranzare in azienda in quello specifico giorno;
- *data*: è un campo timestamp obbligatorio e rappresenta la data di registrazione del pasto effettutato dall'utente;
- *quantita*: è un campo numerico obbligatorio e rappresenta la quantità di pasti che consuma l'utente in quel determinato giorno.

Abbiamo poi la collezione **pranzi** con i seguenti campi:

- *cassaComune*: è un campo numerico obbligatorio e rappresenta la cassa comune come somma di tutte le quote stornate degli utenti;
- *quotaPasti*: è un campo numerico obbligatorio e rappresenta la quota di ogni singolo pasto.

Abbiamo successivamente la collezione **listaSpesa** con i seguenti campi:

- *ingrediente*: è un campo stringa obbligatorio e rappresenta il nome dell'ingrediente da acquistare;
- *quantita*: è un campo numerico obbligatorio e rappresenta la quantita dell'ingrediente da acquistare.
- *isSelezionato*: è un campo booleano obbligatorio e serve per far capire all'utente se un ingrediente è stato acquistato o meno.

E per finire abbiamo la collezione **cronologia** dove vengono descritte tutte le transazioni effettuate nell'applicazione. Di seguito la descrizione dei campi:

- *utenti*: è un campo array obbligatorio e rappresentano tutti gli utenti che hanno partecipato a quella specifica transazione;
- *ammontare*: è un campo numerico obbligatorio e rappresenta l'ammontare di quella specifica transazione;
- *tipo*: è un campo stringa obbligatorio e rappresenta la tipologia di transazione che può essere: pasti, spese e acquisti;
- *nomePietanza*: è un campo stringa e rappresenta il nome della pietanza acquistata;
- *nPasti*: è un campo numerico e rappresenta il numero di pasti effettutati dall'utente principale.

3.3 Architettura dell'Applicazione

3.3.1 Design Pattern utilizzati

MVC

L'*MVC* divide un'applicazione in tre componenti principali, ciascuno con responsabilità specifiche:

- *Model*: rappresenta la logica di business e i dati dell'applicazione
- *View*: rappresenta l'interfaccia utente e la presentazione dei dati.
- *Controller*: contiene la logica di coordinamento tra la Vista e il Modello.

É stato deciso di applicare tale pattern perchè offre i seguenti vantaggi:

- **Separazione delle Responsabilità**: L'*MVC* consente di separare chiaramente la logica di presentazione (Vista) dalla logica di business (Modello) e dalla gestione degli input (Controller). Ciò facilita la manutenzione del codice e permette la riusabilità di componenti;
- **Modularità**: Ogni componente (Modello, Vista, Controller) può essere sviluppato e testato indipendentemente dagli altri. Ciò favorisce la modularità del codice, semplificando lo sviluppo e il testing;
- **Facilità di Manutenzione**: La separazione delle responsabilità semplifica l'individuazione e la correzione degli errori. Consentendo modifiche isolate in una parte del sistema, si riduce il rischio di introdurre bug in altre parti;
- **Scalabilità**: L'*MVC* fornisce una struttura scalabile che facilita l'aggiunta di nuove funzionalità o modifiche senza dover riscrivere l'intero codice.

L'adozione dell'*MVC*, sebbene richieda uno sforzo iniziale più significativo per separare completamente modello, vista e controller, ha offerto notevoli vantaggi nella mia esperienza nello sviluppo dell'applicazione. La chiara separazione delle responsabilità tra queste tre componenti ha contribuito in modo significativo alla creazione di un codice ben strutturato e facilmente gestibile.
La modularità dell'architettura *MVC* ha reso più agevole la manutenzione dell'applicazione nel tempo, consentendo interventi mirati su singoli componenti senza influire pesantemente sul resto del sistema. Questo approccio ha dimostrato di essere particolarmente utile durante l'implementazione di modifiche o l'aggiunta di nuove funzionalità, in quanto ha permesso di concentrarsi su specifiche aree senza causare effetti indesiderati.
La struttura chiara e organizzata derivante dall'utilizzo di *MVC* ha facilitato la comprensione del codice da parte del tutor aziendale. Inoltre, la suddivisione logica tra modello, vista e controller ha fornito un quadro concettuale chiaro, semplificando il processo di debug e di individuazione delle cause di eventuali problemi.
Nel suo complesso, l'adozione dell'architettura *MVC* si è rivelata una scelta importantissima nonostante i maggiori sforzi iniziali, apportando benefici significativi alla qualità del software. Grazie ad essa il sistema ha potuto godere nel tempo di una migliore coerenza, flessibilità e capacità di crescere ed espandersi, oltre ad una più solida robustezza e adattabilità ai cambiamenti e alle crescenti esigenze dell'applicazione.

Singleton

Il *pattern Singleton* [25] garantisce che una classe abbia una sola istanza e fornisce un punto globale di accesso a tale istanza. Questo significa che una volta che l'istanza è stata creata, tutte le successive richieste di ottenere un'istanza restituiranno la stessa istanza originale, senza crearne una nuova. Il *Singleton* è spesso utilizzato quando è necessario un punto di controllo centralizzato per coordinare azioni in un sistema o quando si desidera condividere risorse tra più parti del programma.
Tale *pattern* viene utilizzato nel codice per creare una singola istanza della classe che accede a *Firebase* [8] e a *Flutter Secure Storage*[g] [13]. In questo modo ci assicuriamo che ci sia un unico punto di accesso a tali risorse, evitando così che vengano create più istanze che potrebbero causare problemi di gestione delle risorse o concorrenza nell'accesso ai dati. Il pattern Singleton è perciò adatto a questo scopo poiché garantisce l'esistenza di una sola istanza della classe in tutta l'applicazione.

Builder

Affidarsi ad un *builder* può semplificare di molto la costruzione di oggetti complessi. Questo pattern creazionale permette la produzione di diverse rappresentazioni di un oggetto per mezzo dello stesso codice di costruzione.
Il vantaggio principale dell'uso del *pattern Builder* è che permette di costruire oggetti complessi in modo *step-by-step*, separando la costruzione dalla rappresentazione del prodotto finale. Ciò rende il codice più flessibile, testabile e manutenibile rispetto ad approcci con costruzione incorporata nella classe prodotto.
In Flutter (e nel progetto in questione) è possibile incontrarlo sia sotto forma di widget vero e proprio (prevede una funzione di build la quale restituisce un widget), sia come costruttore particolare di widget standard come *ListView*, *StreamBuilder* e *FutureBuilder*.

3.3.2 Struttura dell'applicazione

Un ottimo modo per descrivere la struttura è partire dall'albero delle cartelle, esso infatti rispecchia in buona parte la suddivisione dei compiti fra le varie classi.
Nello sviluppo Flutter tutto il codice Dart è contenuto nella cartella `\lib`, la quale ha il seguente contenuto:

- *routing.dart*: classe principale nella quale viene definito il *routing*;
- *main.dart*: funzione che lancia l'esecuzione del prodotto;
- *firebase-options.dart*: è un file di configurazione che contiene le opzioni e le impostazioni necessarie per connettersi e utilizzare *Firebase* nell'applicazione.

Il resto di `\lib` è così organizzato:

Models

In questa cartella possiamo trovare i modelli MVC dell'applicazione. Essa contiene i seguenti file:

- *cronologia.dart*: rappresenta la logica di business della sezione **Transazioni** dell'applicazione;

- *dispensa.dart*: rappresenta la logica di business della sezione **Dispensa** dell'applicazione;
- *listaSpesa.dart*: rappresenta la logica di business della sezione **Lista della Spesa** dell'applicazione;
- *pasti.dart*: rappresenta la logica di business della sezione **Pasti** dell'applicazione;
- *pranzi.dart*: rappresenta la logica di business dei **Pranzi**, quali la cassa comune e la quota dei pasti;
- *utente.dart*: rappresenta la logica di business del singolo **Utente**.

Controllers

In questa cartella possiamo trovare i controller MVC dell'applicazione. Essa contiene i seguenti file:

- *cronologia-controller.dart*: questo controller mette in comunicazione la vista delle **Transazioni** con *Firebase* [8]. É responsabile dell'inserimento, dell'aggiornamento e della raccolta dati delle transazioni effettuate;
- *dispensa-controller.dart*: questo controller mette in comunicazione la vista della **Dispensa** con *Firebase*. É responsabile dell'inserimento, dell'aggiornamento e della raccolta dati dei singoli ingredienti;
- *listaSpesa-controller.dart*: questo controller mette in comunicazione la vista della **Lista della Spesa** con *Firebase*. É responsabile dell'inserimento, dell'aggiornamento e della raccolta dati degli ingredienti da acquistare;
- *pasti-controller.dart*: questo controller mette in comunicazione la vista dei **Pasti** con *Firebase*. É responsabile dell'inserimento, dell'aggiornamento e della raccolta dati dei singoli pasti sia dei membri dello staff che degli stagisti;
- *pranzi-controller.dart*: questo controller viene utilizzato per ottenere le info sui **Pranzi** (quindi le informazioni sulla cassa comune e sulla quota attuale dei pasti), calcolare e aggiornare la cassa comune e cambiare la quota dei pasti;
- *utente-controller.dart*: questo controller viene utilizzato per ottenere le info sui singoli **Utenti**, settare la quota stornata e settare le varie informazioni dell'utente quali il nome, il cognome, ecc.

Views

In questa cartella possiamo trovare le viste MVC dell'applicazione. Essa contiene i seguenti file:

- *cronologia-debiti.dart*: questa vista mostra due tab contenenti rispettivamente le transazioni degli utenti e i debiti ancora da saldare;
- *dispensa.dart*: questa vista mostra due tab contenenti rispettivamente gli ingredienti disponibili e la lista della spesa;
- *homepage.dart*: questa vista mostra il resoconto mensile, le quote stornate, la cassa comune e infine la ricetta giornaliera generata da *ChatGPT* [3];

- *login.dart*: questa vista rappresenta la schermata di autenticazione dove l'utente può loggarsi tramite email e password o con l'*SSO* [24] di *Microsoft*;
- *pasti.dart*: questa vista mostra due tab contenenti rispettivamente i pasti già registrati e la possibilità di inserire i giorni desiderati per pranzare in azienda;
- *pietanze-scontrino.dart*: questa vista mostra i pasti interpretati dall'*OCR* [23]. È possibile rimuoverli o modificarli;
- *profilo.dart*: questa vista mostra le informazioni dell'utente, quali: immagine del profilo, email, nome e cognome. Per gli amministratori è possibile modificare inoltre il costo del singolo pasto e registrare i giorni in cui gli stagisti pranzano in azienda;
- *registrazione.dart*: questa vista mostra la schermata di registrazione di un nuovo utente;
- *ricetta.dart*: in questa vista l'utente può consultare: il tempo medio di preparazione del pasto generato da *ChatGPT* [3], il tempo di cottura, la difficoltà, un elenco della lista degli ingredienti necessari e gli step per preparare la pietanza;
- *vista-scontrino.dart*: questa vista visualizza l'immagine dello scontrino captata dalla fotocamera del telefono. É possibile ritagliarla e darla poi in input all'*OCR*.

Abbiamo poi una cartella `\utilities` contenente le classi *Singleton* [25] di *Firebase* [8] e quelle per la gestione di *ChatGPT*, dell'*OCR* e del riconoscimento biometrico.

Capitolo 4

Codifica

Nel seguente capitolo vengono illustrate alcune delle schermate principali del sistema, approfondendone le funzionalità specifiche.

Il codice dell'applicazione è stato sviluppato principalmente per raggiungere tre obiettivi:

- Gestire la presentazione delle interfacce utente dichiarando la struttura delle pagine e l'assemblaggio dei vari componenti, a volte in base allo stato attuale dell'interfaccia;
- Mantenere e modificare lo stato di sessione durante la navigazione tra le diverse schermate;
- Consentire la comunicazione con il database tramite *Firebase* e il pattern *MVC*, permettendo di recuperare dati dal *backend* o inviarne di nuovi.

Nonostante sia consistente il numero di pagine visualizzabili dall'utente, queste sono state sviluppate applicando un approccio standard, in quanto generalmente richiedono funzionalità simili adattate al contesto specifico.
Solitamente ci si avvale della classe screen per definire il *layout*, delle classi *MVC* per la gestione dello stato tramite *Firebase* e di eventuali classi dati per l'interfacciamento al database.

Vediamo ora alcune delle pagine sviluppate precedute da una semplice descrizione.

4.1 Schermata di Login/Registrazione

Quando l'utente installa l'applicazione, la prima schermata che viene visualizzata è quella di login (Figura 4.1). Se l'utente non è ancora registrato, può farlo cliccando sul pulsante "Registrati".
Verrà quindi visualizzata la schermata di registrazione (Figura 4.2) in cui l'utente dovrà inserire tutti i dati richiesti (nome, cognome, email e password).
Una volta completata correttamente la procedura di registrazione, comparirà un modale che informerà l'utente che la registrazione è andata a buon fine e che la sua richiesta dovrà essere approvata dall'amministratore (Figura 4.3). L'amministratore potrà

approvare la richiesta accedendo direttamente a *Firebase* [8] e modificare da falso a vero un *flag* relativo alla registrazione dell'utente.
Nella schermata di login è inoltre presente un *checkbox* attraverso cui l'utente può decidere se salvare o meno le sue informazioni sul dispositivo.
Queste informazioni verranno salvate localmente sul dispositivo attraverso l'utilizzo del pacchetto *Flutter Secure Storage* [13], in modo da poter recuperare e riutilizzare le preferenze dell'utente, come ad esempio il tema selezionato o la possibilità di effettuare l'accesso tramite riconoscimento biometrico.
Per la gestione completa di queste operazioni, sono state impiegate due classi principali: *firebase-auth.dart* e *secure-storage.dart*. Entrambe sono implementate come classi *Singleton* [25] per avere un accesso controllato all'unica istanza della classe. La prima gestisce l'accesso al database *Firebase*, la registrazione degli utenti e l'acquisizione dei dati dell'utente, mentre la seconda è responsabile di salvare le preferenze dell'utente, inclusi l'indirizzo email e la password, in modo sicuro e criptato sul dispositivo, come precedentemente menzionato.
Una volta che l'utente ha effettuato correttamente l'accesso all'applicazione, il sistema lo riconosce e gli assegna una sessione creata automaticamente da *Firebase* con una durata limitata. Questa servirà a evitare che persone non autorizzate utilizzino le funzioni dell'applicazione.

Figura 4.1: Schermata di Login.

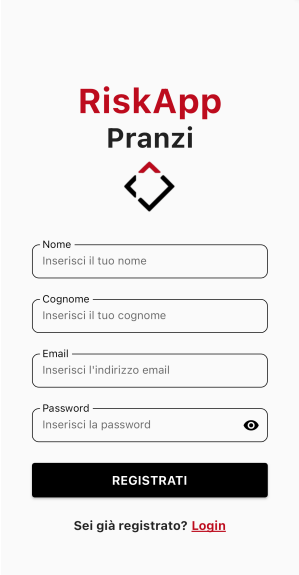


Figura 4.2: Schermata di Registrazione.

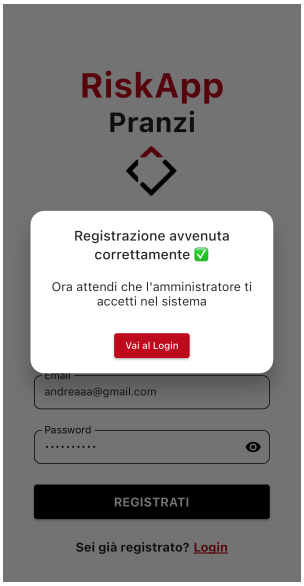


Figura 4.3: Modale di conferma.

4.2 Schermata Home

Una volta che l'utente viene riconosciuto dal sistema, compare un resoconto mensile dei pasti effettuati, contenente il numero e il costo totale dei pasti. Successivamente, compare la cassa comune con le relative quote stornate di tutti gli utenti.
Se viene cliccata l'icona a forma di freccia posta di fianco alla Cassa Comune, l'utente viene reindirizzato in una schermata di riepilogo di tutti i pasti registrati in quel mese come si può vedere in Figura 4.5.
In seguito vengono associati colori differenti per le quote stornate degli utenti: il colore verde indica che si è in credito (quota stornata minore di 0), rosso se si è in debito (quota stornata maggiore di 0) e nero se la quota vale 0.
Tale scelta di impostazione è stata pensata per mostrare all'utente, una volta aperta l'app, le informazioni più rilevanti che sono appunto il resoconto mensile e le quote stornate di tutti gli utenti.
Nella stessa schermata, è possibile visualizzare la ricetta del giorno generata da *ChatGPT* [3] sulla base dei prodotti disponibili nella dispensa, ed è quindi possibile dare origine a una nuova ricetta qualora quella proposta risultasse troppo difficile da preparare. Cliccando il pulsante "Vedi ricetta" si può visualizzare una descrizione dettagliata per la preparazione della pietanza, la sua difficoltà, il tempo di preparazione, il tempo di cottura e gli ingredienti necessari.
Qualora la dispensa risultasse vuota o contenente prodotti inesistenti, la ricetta risulta non disponibile (Figura 4.6).
Per quanto riguarda la gestione e l'utilizzo di *ChatGPT* nell'applicazione, si è fatto uso di una classe *Singleton* per avere un accesso controllato all'unica istanza della classe. In tale classe possiamo trovare diversi metodi utili per ottenere le risposte necessarie ai fini dell'applicazione quali la generazione del pasto del giorno dati gli ingredienti presenti nella dispensa e la generazione della ricetta stessa con le caratteristiche descritte precedentemente. Si vuole inoltre precisare che la chiave privata di *OpenAI* è stata memmorizzata dentro un file **.env** per questioni di sicurezza e portabilità.
Per permettere all'utente di passare da una pagina all'altra è stata implementata una *navbar* posizionata nella parte bassa dello schermo, che ha subìto diverse modifiche nel corso di sviluppo sia a livello di grafica che di logica sottostante, in quanto si voleva agevolare l'utente sia nella navigazione sia nell'interazione con il sistema.
Nella Figura 4.4 è possibile vedere graficamente come viene gestito il tutto.

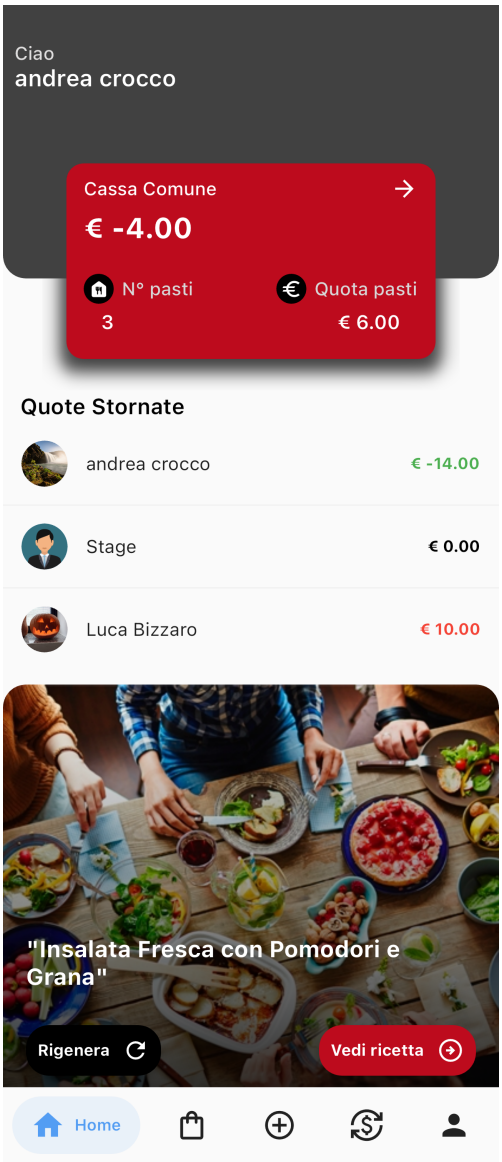


Figura 4.4: Schermata Home.

Figura 4.5: Resoconto mensile dei pasti registrati dall'utente.

Figura 4.6: Caso in cui la dispensa è vuota o sono presenti prodotti con nomi inesistenti.

4.3 Schermata Dispensa

4.3.1 Tab Dispensa

In questa scheda (Figura 4.7) l'utente può eseguire diverse operazioni: effettuare la scansione dello scontrino utilizzando la tecnologia *OCR* [23] per leggere i prodotti acquistati con i relativi prezzi, registrare il costo della spesa che comporterà l'aggiornamento della quota stornata dell'utente e della cassa comune (Figura 4.9), infine, aggiungere un prodotto manualmente inserendo il nome del prodotto, la quantità e a scelta il suo costo (Figura 4.10).
Si è voluto dare la possibilità all'utente di scegliere tra queste tre operazioni per agevolare l'inserimento dei prodotti nella dispensa, dove il pulsante "Registrazione

Spesa", è stato introdotto per poter inserire l'ammontare totale della spesa effettuata senza aggiungere manualmente tutti i nomi dei prodotti acquistati che richiederebbe un tempo maggiore.
Successivamente l'utente può inserire o meno il relativo costo del prodotto. Se quest'ultimo verrà inserito, si aggiornerà automaticamente la sua quota stornata e la cassa comune, lo stesso vale se l'utente decide di cambiare la quantità del singolo prodotto.
Per quanto riguarda l'aggiunta dei prodotti tramite scansione *OCR* [23], l'utente, una volta scelta o scattata una foto dalla galleria dello scontrino in questione, ha la possibilità di ritagliare l'immagine nel caso in cui lo scontrino non risultasse centrato o risultasse poco visibile.
Se ci sono errori nella lettura dello scontrino verrà mostrato all'utente un messaggio con le indicazioni necessarie affinchè l'immagine venga riconosciuta e interpretata correttamente dall'*OCR* (Figura 4.8).
Accanto a ogni prodotto nella dispensa sono presenti due pulsanti: cliccando il primo, viene mostrato all'utente un modale dove viene chiesto se si desidera rimuovere il prodotto dalla dispensa o anche la relativa transazione; con il secondo pulsante si possono ottenere maggiori informazioni sul prodotto e, se è stata utilizzata la scansione *OCR* visualizzandone lo scontrino relativo.
Per quanto riguarda la gestione dell'*OCR* ci si è affidati ad una libreria chiamata *MlKitOcr*, la quale ne semplifica il suo utilizzo. Si è pensato anche qui di utilizzare il pattern *Singleton* [25] per ospitare tale libreria e i metodi necessari per scansionare e raccogliere i dati dagli scontrini.
Oltre a ciò, in tale classe è presente anche un'altra libreria chiamata *EdgeDetection* che viene utilizzata per agevolare l'utente con la modifica e il ritaglio dell'immagine dello scontrino appena fotografato.
É presente inoltre una barra di ricerca dove è possibile cercare rapidamente un prodotto nella dispensa e si è voluto utilizzare lo *StreamBuilder*[g] [28] per ricevere in tempo reale gli inserimenti, le modifiche o la rimozione dei prodotti dalla dispensa da parte degli altri utenti.

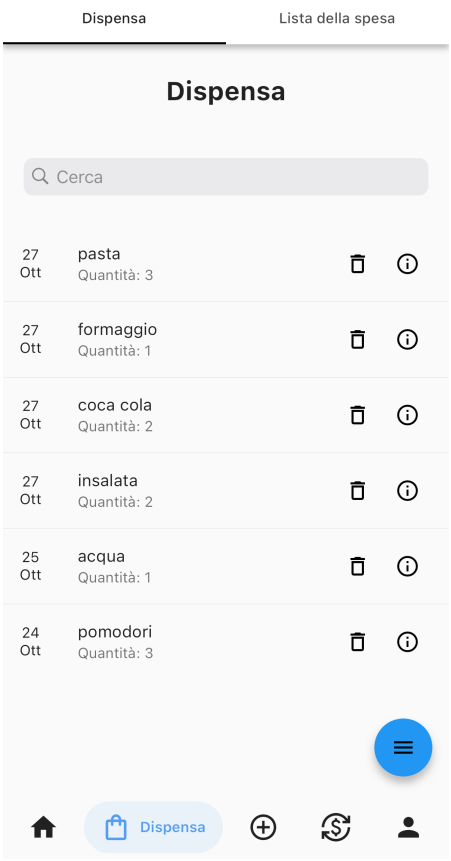


Figura 4.7: Tab Dispensa.

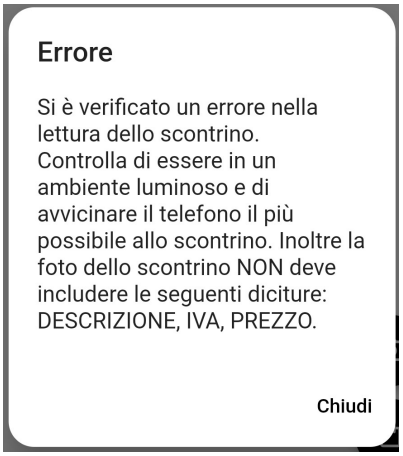


Figura 4.8: Modale di errore della lettura dello scontrino da parte dell'OCR.

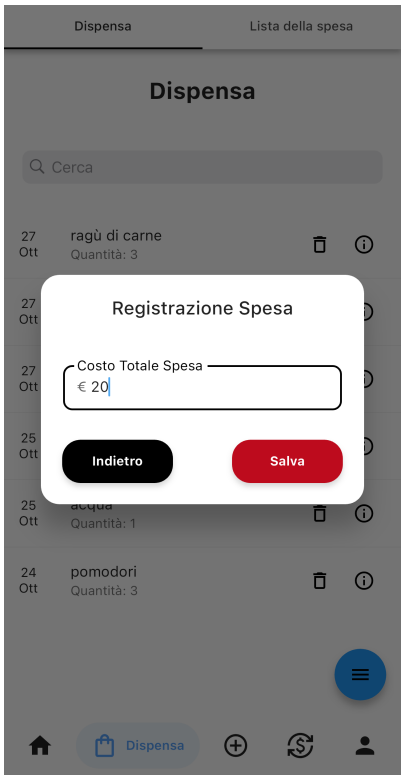


Figura 4.9: Modale registrazione spesa.

Figura 4.10: Modale aggiunta nuovo prodotto.

Per gestire graficamente queste operazioni si è fatto uso di un tasto fluttuante posto in basso a destra dove l'utente, se lo clicca, può vedere una serie di icone corrispondenti alle operazioni descritte precedentemente (Figura 4.11).

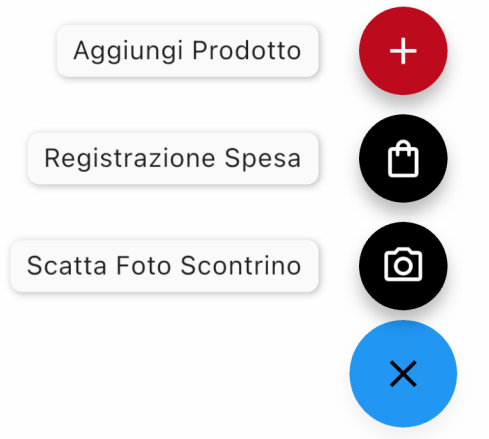


Figura 4.11: Menu fluttuante.

4.3.2 Tab Lista della Spesa

In questa scheda (Figura 4.12) è possibile eseguire diverse operazioni per la lista della spesa, dove ogni utente può inserire nuovi prodotti da acquistare, indicandone il nome e la quantità (Figura 4.13), modificarli, rimuoverli singolarmente o completamente dalla lista della spesa.
Per gestire graficamente queste operazioni si è fatto uso di un tasto fluttuante posto in basso a destra dove l'utente cliccandolo, può vedere una serie di icone corrispondenti alle operazioni descritte precedentemente (Figura 4.15).
L'utente che va a fare la spesa ha la possibilità di selezionare i prodotti aggiunti al carrello facendo tap sulla spunta di fianco al prodotto e, una volta acquistati, inserire i prodotti selezionati direttamente nella dispensa (Figura 4.14), aggiornando automaticamente la sua quota stornata e la cassa comune. In questo modo non c'è bisogno di inserire manualmente i prodotti nella dispensa.
Anche qui è presente una barra di ricerca per poter cercare rapidamente un prodotto nella lista della spesa e si è voluto utilizzare lo *StreamBuilder* [28] per ricevere in tempo reale gli inserimenti, le modifiche o la rimozione dei prodotti da acquistare da parte degli altri utenti.

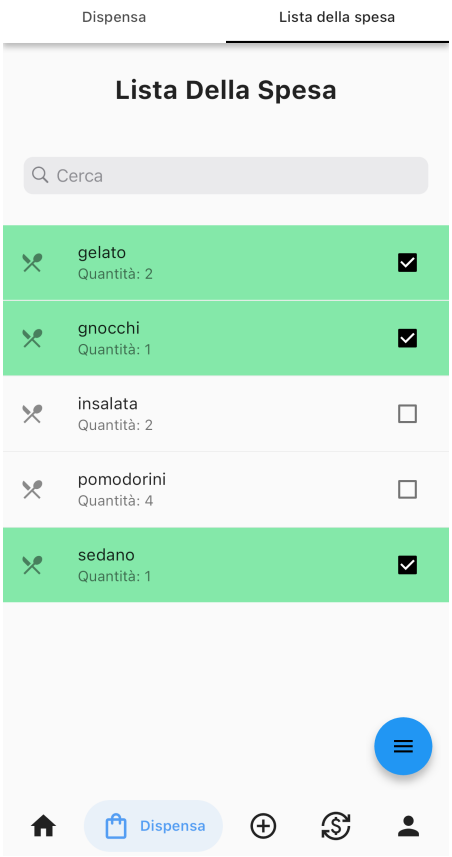


Figura 4.12: Tab Lista della Spesa.

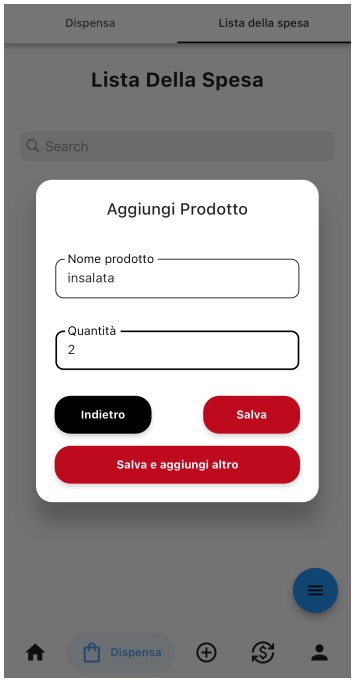


Figura 4.13: Modale aggiunta nuovo prodotto.

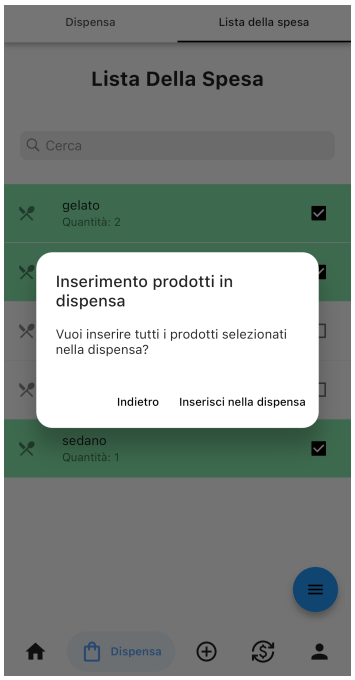


Figura 4.14: Modale inserimento prodotti in dispensa.

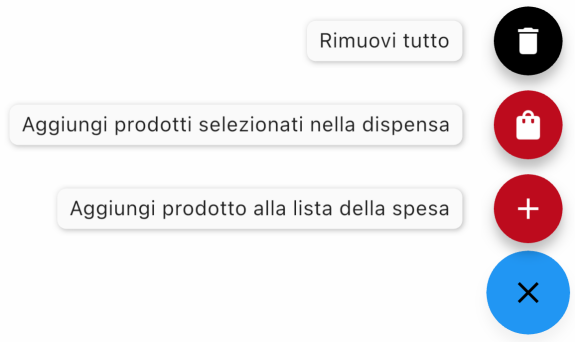


Figura 4.15: Menu fluttuante.

4.4 Schermata Pasti

4.4.1 Tab Pasti Registrati

In questa scheda (Figura 4.16) selezionando un mese, vengono elencati tutti i pasti registrati nel sistema (di default è il mese corrente). Per poter visualizzare i pasti registrati dei mesi scorsi, è possibile utilizzare il pulsante "Scegli data", dove si può selezionare il mese desiderato (Figura 4.17). In questo modo l'utente può vedere tutti i pasti registrati nel corso del tempo.

Oltre a ciò è possibile visualizzare una sintesi del costo del singolo pasto.

Figura 4.16: Tab Pasti Registrati.

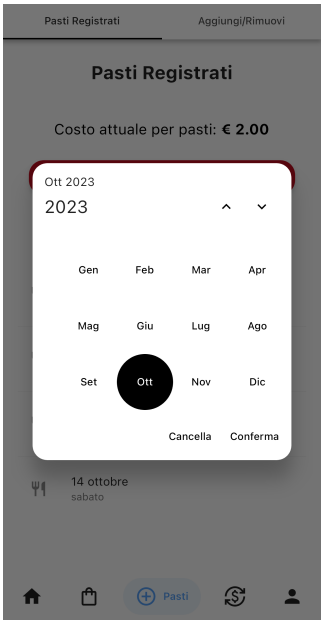


Figura 4.17: Modale per scegliere il mese.

4.4.2 Tab Aggiungi/Rimuovi Pasti

In questa scheda si può aggiungere, modificare o rimuovere uno o più giorni nei quali si desidera pranzare in azienda (Figura 4.18). La logica implementata è risultata particolarmente complessa e ha richiesto molto tempo per lo sviluppo. In sintesi, quando l'utente seleziona delle date dal calendario (accessibile cliccando il pulsante "Mostra calendario"), queste vengono memorizzate per poi essere visualizzate come "Pasti da aggiungere". Naturalmente, per non confondere l'utente tra le date appena inserite e quelle preesistenti, è stato deciso di mostrare o meno il pulsante "Salva" sul calendario a seconda del caso specifico. Per agevolare ulteriormente l'utente, vengono mostrati anche i "Pasti da rimuovere" per comprendere quali pasti verranno rimossi e quali aggiunti al sistema una volta cliccato il pulsante di conferma, posto in basso a destra. Confermate le modifiche, vengono aggiornati la quota stornata dell'utente, la cassa comune e le modifiche ai pasti nella scheda "Pasti Registrati".
Si è voluto utilizzare la libreria *TableCalendar* come base per registrare o meno i pasti dei singoli utenti in quanto aveva una ricca documentazione ed era ben conosciuta dalla community di *Flutter* [12].
Nella Figura 4.19 si può vedere in dettaglio come viene gestita l'aggiunta e la rimozione dei giorni sul calendario.
Su richiesta del committente, è stata disabilitata la possibilità di registrare i pasti per le date future rispetto al giorno corrente, consentendo di inserire solo pasti relativi alla giornata in corso o precedenti e in questo modo l'utente può aggiungere nel sistema solo i pasti effettivamente consumati, senza dover prenotare quelli per i giorni successivi. Tale scelta è stata dettata dall'esigenza di mantenere veritieri i dati registrati sulla piattaforma.

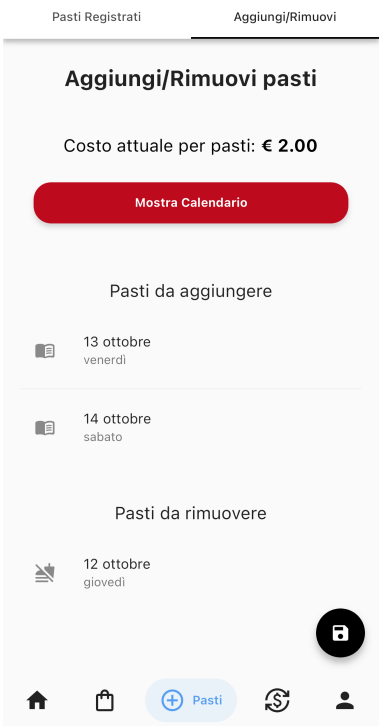


Figura 4.18: Tab Aggiunta/Rimozione Pasti.

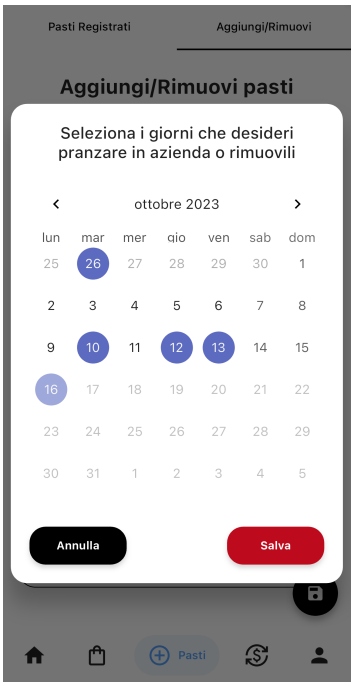


Figura 4.19: Calendario per l'aggiunta o la rimozione dei giorni.

4.5 Schermata Transazioni

4.5.1 Tab Cronologia

In questa scheda (Figura 4.20) si può visualizzare un riepilogo di tutte le transazioni effettuate nel mese selezionato (di default è il mese corrente). Queste transazioni includono gli acquisti effettuati, l'importo dei pasti consumati e i soldi dati ad altri utenti per sdebitarsi (spese). La *UI* è stata progettata per essere semplice da comprendere e facile da utilizzare, infatti si è pensato di includere una serie di filtri denominati rispettivamente: "utente", "spese", "pasti" e "pagamenti" per semplificare la navigazione. Se viene premuto il pulsante "spese", è possibile visualizzare solo gli acquisti effettuati nel mese selezionato (Figura 4.21). Premendo il pulsante "pasti", si visualizza solo il numero di pasti con il relativo costo del mese selezionato (Figura 4.22); mentre premendo il pulsante "pagamenti", è possibile visualizzare solo le spese sostenute tra i membri dello staff per sdebitarsi (Figura 4.23). Infine, abbiamo il pulsante "utente", dove si possono selezionare utenti specifici (Figura 4.24) e visualizzare tutte le transazioni a cui sono stati coinvolti (Figura 4.25).
Per agevolare l'utente con la lettura delle transazioni si è voluto associare a ciascun tipo di transazione un'icona specifica che richiami quel tipo di transazione, ed è stato usato lo *StreamBuilder* [28] per ricevere in tempo reale gli aggiornamenti sulle transazioni.

Figura 4.20: Tab Cronologia.

Figura 4.21: Filtro per gli Acquisti.

Figura 4.22: Filtro per i Pasti.

Figura 4.23: Filtro per le Spese.

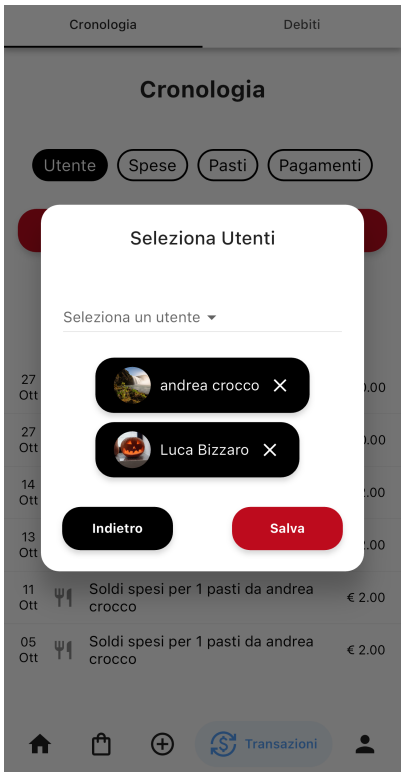


Figura 4.24: Filtro Utenti: selezione degli utenti.

Figura 4.25: Visualizzazione transazioni filtro Utenti.

Le singole transazioni si possono eliminare semplicemente facendo tap sulla transazione da rimuovere. Una volta "toccata", è possibile visualizzare un riepilogo completo della transazione e un pulsante "Rimuovi transazione" per eliminarla. Naturalmente, vengono rimosse solo le proprie transazioni e non quelle degli altri (Figura 4.26).
Una volta che l'utente clicca il pulsante "Rimuovi Transazione" il sistema aggiorna autonomamente la sua quota stornata e la cassa comune.
Si è deciso di agevolare l'utente rimuovendo le transazioni direttamente da questa schermata oltre a poterla rimuovere dalle altre schede, come ad esempio nella sezione della Dispensa o nella sezione dei Pasti in cui sono presenti le operazioni di rimozione.

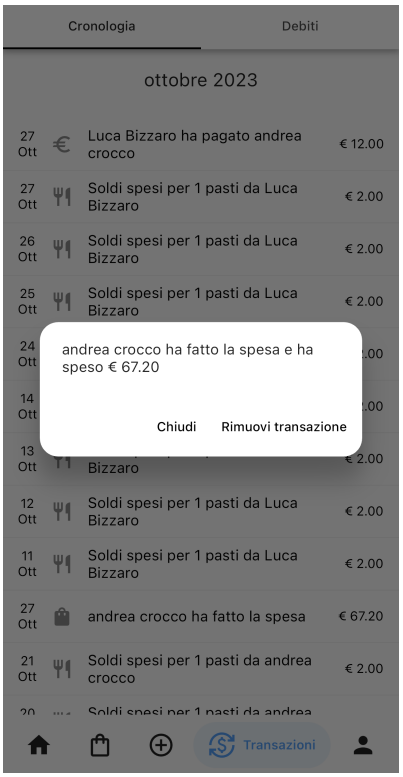


Figura 4.26: Riepilogo della transazione con possibilità di rimozione.

4.5.2 Tab Debiti

In questa scheda è possibile vedere se si è in debito o meno (quindi con quota stornata maggiore di 0). Nella Figura 4.27 viene visualizzato se si è in debito, quanto ammonta e gli utenti con il quale è possibile sdebitarsi. Successivamente viene visualizzata una lista di tutti gli utenti con quota stornata minore di 0, con nome e cognome dell'utente e il relativo credito.
In questo modo l'utente può decidere con quale utente sdebitarsi senza dover pagare una persona specifica.
Per saldare un debito con un utente, l'utente principale deve fare tap sull'utente in questione e inserire l'importo desiderato nel modale che compare. Una volta inserito l'importo, si preme il pulsante "Salva" per inviare le modifiche al sistema (Figura 4.28).
Se la quota da restituire all'utente è pari a 0, verrà visualizzato un messaggio che conferma che tutto è a posto. Tuttavia, se si è in credito (quindi con quota stornata minore di 0), verrà mostrato un messaggio che indica l'ammontare del credito. In caso contrario, come precedentemente descritto, verrà presentata una lista degli utenti con cui l'utente può saldare il proprio debito.
Come in ogni altra schermata dell'applicazione anche qui si fa uso di uno *StreamBuilder* [28]. In questo caso viene usato per ricevere aggiornamenti in tempo reale sulla situazione attuale dell'utente.

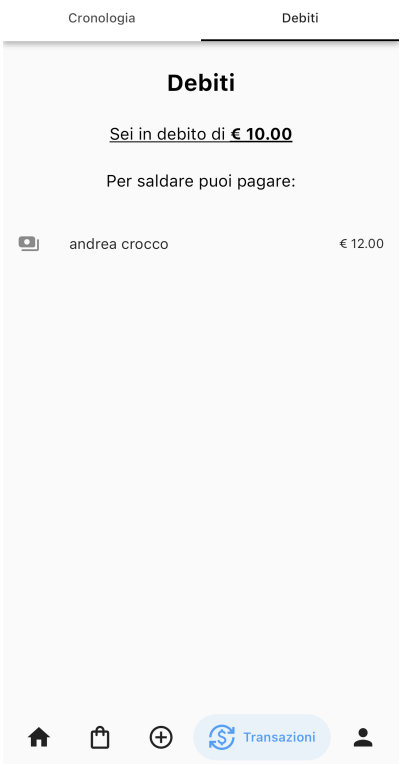


Figura 4.27: Tab Debiti.

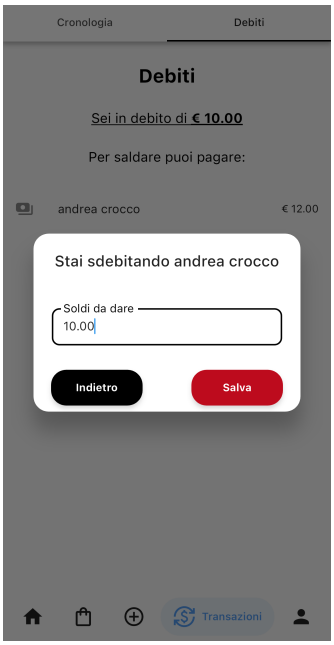


Figura 4.28: Modale per sdebitamento.

4.6 Schermata Profilo

In questa schermata (Figura 4.30), l'utente può cambiare la foto del profilo cliccando sull'icona a forma di matita situata accanto all'immagine del profilo. Le immagini dei profili dei singoli utenti vengono memorizzati dentro *Firebase Storage* [10] usando i metodi offerti dalla documentazione ufficiale. L'utente può, inoltre, modificare il nome, cognome e il tema, passando da *chiaro* a *scuro* o viceversa a seconda delle proprie preferenze.
Nella sezione dell'account, l'utente ha la possibilità di cambiare la password attuale con una nuova (la nuova password deve contenere almeno 8 caratteri alfanumerici) (Figura 4.31) e può decidere se utilizzare o meno il riconoscimento biometrico, come FaceID o l'impronta digitale (se disponibili), per l'accesso futuro all'app. Si è voluto aggiungere il riconoscimento biometrico come possibile metodo di autenticazione per accedere più velocemente alla piattaforma senza dover inserire email e password.
Per ragioni di sicurezza, si è deciso di non rendere modificabile l'indirizzo email per evitare l'inserimento di email spam.
Come già accennato nella Schermata di Login/Registrazione, la possibilità di utilizzare il tema e l'accesso tramite la biometria è disponibile solo se si è accettata l'opzione di salvare le informazioni sul telefono. In caso contrario, verrà visualizzato un messaggio che spiega la situazione all'utente, offrendo la possibilità di effettuare il logout e ritornare nella schermata precedente (Figura 4.32).
Sempre in questa schermata è presente in fondo alla pagina un pulsante "Esegui il Logout" dove, se cliccato, l'utente viene disconnesso sia dall'applicazione (tornando nella schermata di login), sia da *Firebase* [8].
Se l'accesso all'applicazione viene eseguito da un'amministratore, nella schermata del profilo viene aggiunta una sezione denominata "Pranzi", dove è possibile modificare il costo dei pasti e registrare i pasti degli stagisti (Figura 4.29).
L'amministratore è l'unico utente autorizzato ad inserire i pasti perchè si vuole sempre tenere un profilo veritiero su tutta la piattaforma.

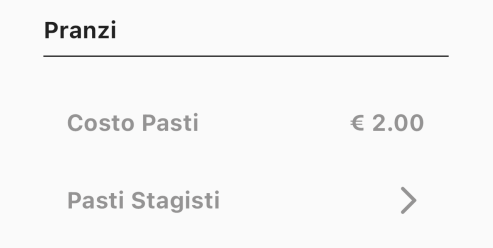


Figura 4.29: Sezione Pranzi.

Figura 4.30: Schermata Profilo.

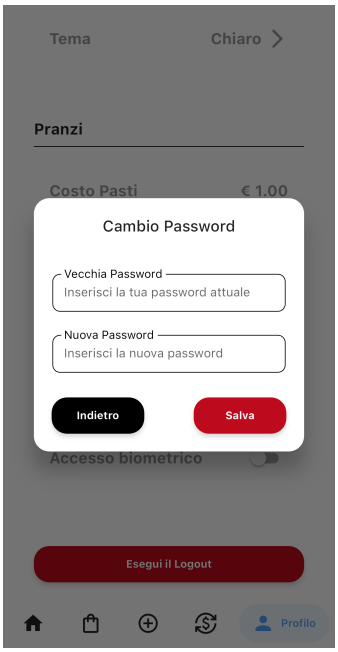


Figura 4.31: Modale per cambiare la password di accesso alla piattaforma.

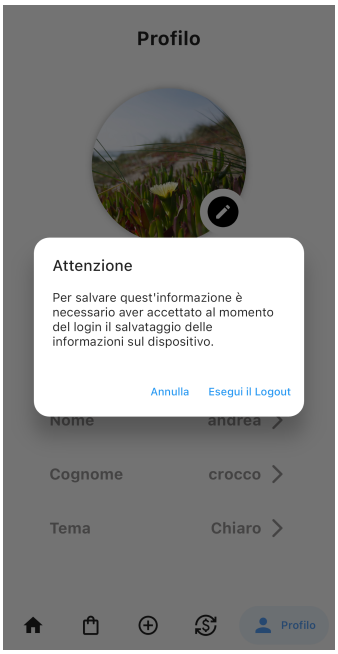


Figura 4.32: Messaggio di avvertimento che l'utente non ha accettato le condizioni al momento del login.

Riguardo la registrazione dei giorni in cui gli stagisti hanno pranzato in azienda, la schermata è pressochè simile a quella descritta nella sezione Schermata Pasti con

alcune differenze: nella scheda "Pasti Registrati" (Figura 4.33) sono presenti accanto a ciascuna data il numero di stagisti che hanno pranzato, a seguire ci sono due pulsanti che servono a modificare il numero dei pasti consumati in quel giorno. In presenza di tali modifiche rilevate dalla piattaforma, viene visualizzato un pulsante in basso a destra per inviare al sistema i cambiamenti apportati. Il sistema poi aggiornerà autonomamente la quota stornata e la cassa comune degli stagisti. La scelta di poter togliere la quantità dei pasti è stata inserita direttamente in questa tab, affinchè l'amministratore potesse modificare rapidamente la quantità dei pasti consumati in quel giorno. Nella scheda "Aggiungi/Rimuovi" il discorso non cambia: anche qui, accanto a ciascuna data, sono presenti due pulsanti con il numero di pasti consumati durante il giorno. In questa schermata è possibile rimuovere i pasti già registrati sul sistema oppure aggiungerne di nuovi (Figura 4.34).

Figura 4.33: Scheda Pasti Registrati per gli stagisti.

Figura 4.34: Scheda Aggiungi/Rimuovi per gli stagisti.

Capitolo 5

Conclusioni

Il presente capitolo riporta il consuntivo delle attività svolte, gli obiettivi raggiunti, i requisiti soddisfatti e una breve valutazione personale del lavoro complessivo.

Il mio periodo di stage, della durata di trecento ore, si è tenuto presso l'azienda RiskApp S.r.l. L'obiettivo del progetto di stage è stato quello di sviluppare un'applicazione mobile per la gestione interna delle spese relative ai pranzi effettuati dallo staff dell'azienda. Nell'applicazione è possibile inserire, modificare e rimuovere prodotti dalla dispensa; effettuare la scansione *OCR* dello scontrino; fare la lista della spesa; inserire o rimuovere i giorni in cui si vuole pranzare in azienda e visualizzare le transazioni e i debiti dei vari utenti.
Quasi tutti gli obiettivi (ad eccezione della stesura dei test) e sono stati raggiunti tutti i requisiti richiesti dal proponente. Sono molto soddisfatto dello stage, poiché ho avuto l'opportunità di creare per la prima volta un'app mobile completa per un'azienda, con un personale molto disponibile ad affrontare i miei dubbi durante lo sviluppo dell'applicazione.

5.0.1 Consuntivo finale

La Tabella 5.1 riporta il consuntivo delle attività svolte durante lo stage. La fase di analisi e pianificazione si è prolungata oltre le previsioni a causa della necessità di impiegare un tempo maggiore per individuare i casi d'uso e per esaminare il vecchio sistema di gestione dei pasti tramite fogli Excel. L'attività di scrittura dei test non è stata eseguita poiché le fasi di analisi, pianificazione e sviluppo hanno richiesto un quantitativo di tempo superiore rispetto a quanto inizialmente programmato.

5.1 Valutazione del Progetto

5.1.1 Completamento degli obiettivi

Nella Tabella 5.2 si presenta l'andamento del conseguimento degli obiettivi. La non realizzazione dell'obiettivo F1, originariamente considerato come opzionale, è dovuta alla necessità di dedicare un tempo maggiore del previsto al completamento degli obiettivi obbligatori e desiderabili.

Tabella 5.1: Ripartizione delle ore

Ore previste	Ore svolte	Descrizione dell'attività
72	**17**	**Formazione sulle tecnologie**
62	**75**	**Analisi**
16	*29*	*Analisi del problema e del dominio applicativo*
46	*46*	*Inizio progettazione dell'app mobile*
150	**170**	**Mock-up e Sviluppo**
30	*30*	*Creazione mock-up dell'applicazione*
40	*40*	*Progettazione*
90	*100*	*Sviluppo*
50	**-**	**Test**
38	**38**	**Collaudo finale**
28	*28*	*Collaudo*
5	*5*	*Verifica documentazione finale*
2	*2*	*Preparazione presentazione e demo*
1	*1*	*Incontro di presentazione della piattaforma con gli stakeholders*
2	*2*	*Live demo di tutto il lavoro di stage*
300	**300**	**Totale ore**

5.1.2 Soddisfacimento dei requisiti

Osservando la tabella 5.3 visualizziamo i requisiti funzionali con il loro stato, segue la tabella 5.4 dove sono descritti i requisiti qualitativi con lo stato relativo. Infine nell'ultima tabella 5.5 troviamo lo stato dei requisiti di vincolo.

Tabella 5.2: Completamento degli obiettivi.

Codice	Descrizione	Stato
O1	Accesso tramite credenziali	Completato
O2	Monitoraggio spese della cassa comune	Completato
O3	Aggiornamento delle spese	Completato
D1	Integrazione di un algoritmo per l'integrazione automatica degli scontrini delle spese effettuate per comprare il cibo per il pranzo	Completato
D2	Integrazione di *ChatGPT* [3] per ricevere consigli sulle ricette giornaliere, in quanto predispongono di cucina interna	Completato
F1	Stesura dei test	Non Completato
F2	Deploy WebApp	Completato

Tabella 5.3: Soddisfacimento dei requisti funzionali.

Requisito	Descrizione	Stato
RFNE-1	L'utente deve poter accedere all'applicazione per poter usufruire del servizio	Completato
RFNE-2	L'utente deve poter vedere il resoconto mensile	Completato
RFNE-3	L'utente deve poter visualizzare la cassa comune per vedere quanto denaro è stato risparmiato	Completato
RFNE-4	L'utente deve poter visualizzare le quote stornate di tutti gli utenti	Completato
RFN-5	L'utente deve poter vedere la ricetta del giorno generata da *ChatGPT* [3]	Completato
RFNE-6	L'utente deve poter inserire nella dispensa uno o più ingredienti	Completato
RFN-7	L'utente deve poter modificare il nome e la quantità degli ingredienti nella dispensa	Completato
RFNE-8	L'utente deve poter scegliere i giorni che desidera pranzare in azienda	Completato
RFNE-9	L'utente deve poter visualizzare tutte le transazioni eseguite nel mese corrente	Completato
RFNE-10	L'utente deve poter visualizzare tutti gli utenti che sono in debito	Completato
RFN-11	L'utente deve poter cambiare la propria password per accedere al servizio	Completato
RFNE-12	L'utente deve poter eseguire il *logout* dal sistema	Completato
RFNE-13	L'amministratore deve poter modificare il costo dei pasti in azienda	Completato
RFNE-14	L'amministratore deve poter aggiungere i pasti degli stagisti che desiderano pranzare in azienda	Completato

Tabella 5.4: Soddisfacimento dei requisiti qualitativi.

Requisito	Descrizione	Stato
RQD-1	L'utente può eseguire l'accesso usando il proprio account *Microsoft*	Completato
RQD-2	L'utente può scansionare lo scontrino con la lista della spesa effettuata	Completato
RQD-3	L'utente può modificare la propria immagine del profilo	Completato
RQD-4	L'utente può modificare il proprio nome	Completato
RQD-5	L'utente può modificare il proprio cognome	Completato
RQZ-1	L'utente può effettuare la ricerca di un ingrediente nella dispensa	Completato
RQZ-2	L'utente può filtrare le transazioni per costo dei pasti	Completato
RQZ-3	L'utente può filtrare le transazioni per acquisti fatti	Completato
RQZ-4	L'utente può filtrare le transazioni per spese	Completato
RQZ-5	L'utente può filtrare le transazioni per utente	Completato
RQZ-6	L'utente può modificare il tema dell'app da chiaro a scuro e viceversa	Completato
RQZ-7	L'utente può registrare la propria impronta digitale per accedere al servizio	Completato
RQZ-8	L'utente può aggiungere nella lista della spesa un nuovo ingrediente da acquistare	Completato
RQZ-9	L'utente può modificare dalla lista della spesa l'ingrediente da acquistare	Completato
RQZ-10	L'utente può rimuovere dalla lista della spesa un ingrediente	Completato
RQZ-11	L'applicazione deve funzionare anche usando il Browser da PC	Completato

Tabella 5.5: Soddisfacimento dei requisiti di vincolo.

Requisito	Descrizione	Stato
RVNE-1	L'applicazione deve funzionare sui dispositivi mobili (Android e iOS)	Completato

Tutti i requisiti sono stati soddisfatti e l'azienda è stata molto soddisfatta del lavoro svolto e della documentazione fornita.

5.1.3 Conoscenze acquisite

Lo stage ha permesso di ampliare le mie conoscenze del linguaggio *Dart* [6] e del framework *Flutter* [12] per lo sviluppo di app mobile, inoltre ho avuto modo di conoscere tutti i servizi offerti da *Firebase* [8] con il loro funzionamento applicandolo ad un contesto reale. Oltre a ciò, grazie allo stage ho avuto modo di capire come creare da zero un'app mobile partendo dal mock-up, passando poi alla progettazione fino ad arrivare alla sua codifica. Infine ho imparato a creare *UI* accattivanti usando il software *Figma* [7] con l'aiuto del mio tutor.
Durante il mio stage, ho avuto l'opportunità di immergermi nel mondo del lavoro reale e di fare nuove conoscenze. Questa esperienza mi ha profondamente affascinato, poiché

mi ha permesso di esplorare il campo delle applicazioni mobile in dettaglio, ispirandomi persino a creare una mia app. Avendo raggiunto tutti gli obiettivi mi ritengo molto soddisfatto delle mie capacità e delle mie conoscenze apprese durante questa bellissima esperienza.
Il team di RiskApp è stato eccezionalmente disponibile nel risolvere i miei dubbi durante lo sviluppo dell'applicazione e ha fornito preziosi consigli sui programmi da utilizzare e la configurazione dell'ambiente di lavoro.
Le 300 ore di stage hanno rappresentato un'opportunità fondamentale per perfezionare le mie competenze informatiche e organizzative, consentendomi di mettere in pratica le conoscenze acquisite nel corso del mio percorso universitario.
Nel complesso, ritengo di aver fatto una scelta eccellente sia per il progetto che per l'azienda stessa.

Acronimi e abbreviazioni

ChatGPT Chat Generative Pre-trained Transformer. 2

IdP Identity Provider. 29

JWT JSON Web Token. 27

SaaS Software as a Service. 1

SP Service Provider. 29

SSO Single sign-on. 69

UML Unified Modeling Language. 4

Glossario

Agile in ingegneria del software, la metodologia agile è un approccio allo sviluppo del software basato sulla distribuzione continua di software efficienti creati in modo rapido e iterativo [1]. 1

ChatGPT è un *chatbot* basato su intelligenza artificiale e apprendimento automatico sviluppato da OpenAI[g] specializzato nella conversazione con un utente umano [3]. 68

Codebase è l'intera collezione di codice sorgente usata per costruire una particolare applicazione o un particolare componente [4]. 2

cross-platform si intende che un software, sia esso un gioco o un'app di qualunque tipo, è stato sviluppato per più piattaforme diverse. Idealmente per tutte le piattaforme disponibili, sulle quali deve offrire le stesse caratteristiche e funzionalità [5]. 23

Flutter Secure Storage è un pacchetto Flutter progettato per fornire un metodo sicuro per archiviare dati sensibili in un'app Flutter. Questo può includere informazioni come token di accesso, password o altre informazioni riservate. L'obiettivo principale è proteggere queste informazioni da accessi non autorizzati e garantire la sicurezza dei dati sensibili dell'utente [13]. 39

IdP è un sistema che crea, archivia e gestisce le identità digitali [18]. 68

JWT è un token di accesso standardizzato e consente lo scambio sicuro di dati tra due parti. Contiene tutte le informazioni importanti su un'entità, in modo che non sia necessaria alcuna interrogazione del database e che la sessione non debba essere memorizzata sul server [19]. 68

NoSQL fa riferimento a tipi di database non relazionali e questi database archiviano i dati in un formato diverso dalle tabelle relazionali. I database NoSQL sono ampiamente utilizzati nelle applicazioni Web in tempo reale e nei big data, poiché i loro principali vantaggi sono l'elevata scalabilità e l'elevata disponibilità [21]. 26

open source è un codice progettato per essere accessibile pubblicamente. Chiunque può vederlo, modificarlo e distribuirlo secondo le proprie necessità [22]. 23

SaaS in informatica è un modello di servizio del software applicativo realizzato da un produttore che mette a disposizione un programma, direttamente o tramite terze parti, con modalità telematiche come ad esempio un'applicazione web [27]. 68

SP è un'applicazione o un servizio che si affida ad un Identity Provider[g] esterno per l'autenticazione degli utenti [18]. 68

SSO in informatica il Single Sign On è dunque un sistema specializzato che permette ad un utente di autenticarsi una sola volta e di accedere a tutte le risorse informatiche alle quali è abilitato [24]. 4

StreamBuilder è un widget che ricostruisce l'interfaccia utente in base ai nuovi valori passati tramite lo stream in ascolto [28]. 47

UML in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico [29]. 68

Bibliografia

Siti web consultati

[1] *Agile*. URL: https://www.bitil.com/best-practices/agile (cit. alle pp. 1, 69).

[2] *Architettura Flutter*. URL: https://docs.flutter.dev/resources/architectural-overview (cit. a p. 24).

[3] *ChatGPT*. URL: https://it.wikipedia.org/wiki/ChatGPT (cit. alle pp. 2–5, 9, 10, 21, 40, 41, 44, 65, 69).

[4] *Codebase*. URL: https://it.wikipedia.org/wiki/Codebase (cit. alle pp. 2, 69).

[5] *Cross-Platform*. URL: https://www.fastweb.it/fastweb-plus/digital-magazine/cross-platform-cosa-significa-e-perche-e-importante-nei-videogiochi/ (cit. alle pp. 23, 69).

[6] *Dart*. URL: https://dart.dev/ (cit. alle pp. 24, 66).

[7] *Figma*. URL: https://www.figma.com/ (cit. alle pp. 35, 66).

[8] *Firebase*. URL: https://firebase.google.com/ (cit. alle pp. 39–41, 43, 59, 66).

[9] *Firebase Authentication*. URL: https://firebase.google.com/docs/auth?hl=it (cit. alle pp. 27, 36).

[10] *Firebase Storage*. URL: https://firebase.google.com/docs/storage?hl=it (cit. alle pp. 27, 59).

[11] *Firestore*. URL: https://firebase.google.com/docs/firestore?hl=it (cit. alle pp. 2, 26).

[12] *Flutter*. URL: https://flutter.dev/ (cit. alle pp. 26, 53, 66).

[13] *Flutter Secure Storage*. URL: https://stackoverflow.com/questions/60840704/what-is-flutter-secure-storage-exactly-and-how-it-works (cit. alle pp. 39, 43, 69).

[14] *Funzionamento SSO*. URL: https://www.cybersecurity360.it/nuove-minacce/single-sign-on-accesso-facilitato-alle-risorse-di-rete-ecco-come-funziona/ (cit. a p. 30).

[15] *Git*. URL: https://git-scm.com/ (cit. a p. 33).

[16] *Github*. URL: https://github.com/ (cit. a p. 34).

[17] *Hackolade*. URL: https://hackolade.com/ (cit. a p. 32).

[18] *Identity Provider e Service Provider*. URL: https://www.entrust.com/it/resources/faq/what-is-an-identity-provider (cit. alle pp. 29, 69, 70).

[19] *Json Web Token*. URL: https://www.ionos.it/digitalguide/siti-web/programmazione-del-sito-web/json-web-token-jwt/ (cit. alle pp. 27, 69).

[20] *Mac OS Ventura*. URL: https://it.wikipedia.org/wiki/MacOS_Ventura (cit. a p. 31).

[21] *NoSQL*. URL: https://www.oracle.com/it/database/nosql/what-is-nosql/ (cit. alle pp. 26, 69).

[22] *Open Source*. URL: https://www.redhat.com/it/topics/open-source/what-is-open-source (cit. alle pp. 23, 69).

[23] *Optical Character Recognition*. URL: https://it.wikipedia.org/wiki/Riconoscimento_ottico_dei_caratteri (cit. alle pp. 28, 36, 41, 46, 47).

[24] *Single Sign On*. URL: https://vitolavecchia.altervista.org/che-cosa-e-e-come-funziona-il-single-sign-on-sso-in-informatica/ (cit. alle pp. 4, 5, 7, 30, 41, 70).

[25] *Singleton*. URL: https://it.wikipedia.org/wiki/Singleton_(informatica) (cit. alle pp. 39, 41, 43, 47).

[26] *Slack*. URL: https://slack.com/intl/it-it (cit. a p. 35).

[27] *Software as a Service*. URL: https://it.wikipedia.org/wiki/Software_as_a_service (cit. alle pp. 1, 69).

[28] *StreamBuilder*. URL: https://www.scaler.com/topics/streambuilder-flutter/ (cit. alle pp. 47, 50, 54, 57, 70).

[29] *Unified Modeling Language*. URL: https://systemscue.it/uml-un-passo-avanti-per-lingegneria-del-sofware/11729/ (cit. alle pp. 4, 70).

[30] *Visual Studio Code*. URL: https://code.visualstudio.com/ (cit. a p. 32).