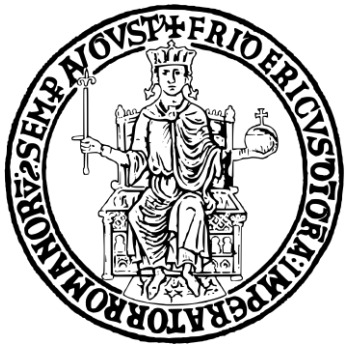


Ingegneria del software
2022 – 2023

Ratauille23

Cristian Carrella & Alfredo D’Andrea

Indice

1 Gestione della documentazione

1.1 Contributors

Ruolo	Nome e Cognome	Matricola
Co-autore	Cristian Carrella	NA86003677
Co-autore	Alfredo D'Andrea	NA86003615

1.2 Version Control

Data	Versione	Autore	Commento
31/11/2022	1.0	Cristian Carrella Alfredo D'Andrea	- Divisione in capitoli e paragrafi del documento. - Scelta del font da utilizzare
02/12/2023	1.1	Cristian Carrella Alfredo D'Andrea	Aggiunta di: - Individuazione target di utenti - Personas
05/12/2023 – 10/12/2023	1.2	Cristian Carrella Alfredo D'Andrea	Aggiunta di: - Analisi dei requisiti - Use Case Diagram - Template Cockburn - Class Diagram - Sequence Diagram - Activity Diagram
27/12/2022 – 08/01/2023	1.3	Cristian Carrella Alfredo D'Andrea	Aggiunta di: - Statechart - Usabilità - System design
27/01/2023 – 30/01/2023	1.4	Cristian Carrella Alfredo D'Andrea	Aggiunta di: - Testing
02/02/2023 – 09/02/2023	1.8	Cristian Carrella Alfredo D'Andrea	Aggiunta di: - Usabilità sul campo
10/02/2023 – 20/02/2023	1.9	Cristian Carrella Alfredo D'Andrea	- Aggiunta del glossario - Controllo errori grammaticali - Controlli finali

2 GLOSSARIO

Use Case Diagram	Sono diagrammi che descrivono come un attore interagisce con le varie funzionalità o servizi messi a disposizione dal sistema
Class Diagram	I diagrammi delle classi (class diagram) sono uno dei tipi di diagrammi che possono comparire in un modello UML. In termini generali, consentono di descrivere tipi di entità, con le loro caratteristiche e le eventuali relazioni fra questi tipi
Sequence Diagram	Un Sequence Diagram è un diagramma previsto dall'UML utilizzato per descrivere uno scenario. Uno scenario è una determinata sequenza di azioni in cui tutte le scelte sono state già effettuate; in pratica nel diagramma non compaiono scelte, né usi alternativi
Activity Diagram	L'Activity Diagram è un tipo di diagramma che permette di descrivere un processo attraverso dei grafi in cui i nodi rappresentano le attività e gli archi l'ordine con cui vengono eseguite
Statechart	Uno State Chart è un tipo di diagramma usato per descrivere il comportamento dei sistemi, il quale viene analizzato e rappresentato tramite una serie di eventi che potrebbero accadere per ciascun stato
MockUp	Un mockup è una realizzazione a scopo illustrativo di un oggetto o un sistema, senza le complete funzioni dell'originale; un mockup può rappresentare la totalità o solo una parte dell'originale di riferimento
API	API è l'abbreviazione di (application programming interface), ovvero un insieme di definizioni e protocolli per la creazione e l'integrazione di software applicativi
Test Case (TC)	In ingegneria del software, un caso di test è un insieme di condizioni o variabili sotto le quali un tester determina se una applicazione o sistema software risponde correttamente o meno
Classe di Equivalenza (CE)	Una relazione di equivalenza è un concetto matematico che esprime in termini formali oggetti che condividono una certa proprietà
MININT	È il valore minimo che un intero può assumere
MAXINT	È il valore massimo che un intero può assumere

Database (DB)	È un archivio di dati strutturato in modo da razionalizzare la gestione e l'aggiornamento delle informazioni e da permettere lo svolgimento di ricerche complesse
Null	È un valore nullo, ovvero un puntatore che non contiene un indirizzo di memoria valido
Node Path	Percorso dei nodi eseguiti per percorrere una determinata strada
GUI	L'interfaccia grafica, nota anche come GUI, in informatica è un tipo di interfaccia utente che consente l'interazione uomo-macchina in modo visuale utilizzando rappresentazioni grafiche
Affordance	Con affordance si definisce la qualità fisica di un oggetto che suggerisce a un essere umano le azioni appropriate per manipolarlo
Label	Tradotto con "etichetta" è un controllo grafico che mostra informazioni testuali
ComboBox	È un controllo grafico che permette all'utente di effettuare una scelta scrivendola in una casella di testo o selezionandola da un elenco
Shortcut	Una scorciatoia da tastiera è la pressione di due o più tasti contemporaneamente che richiamano una certa operazione
Backend	Lo sviluppo back-end è tutto ciò che opera dietro le quinte di una pagina come le interazioni con il database, in questo caso anche con l'uso di rest API
Frontend	Lo sviluppo front-end è lo sviluppo dell'interfaccia utente grafica del software in modo che gli utenti possano visualizzare e interagire con le sue funzionalità

3 INTRODUZIONE AL SISTEMA

"Ratatuille23" è un sistema informativo che si occupa della gestione di attività nell'ambito della ristorazione offrendo funzionalità intuitive per raggiungere al meglio tale scopo. Il sistema è distribuito, infatti, dovrà comprendere due applicazioni client (Android / Windows) e una gestione server.

Requisiti assegnati:

1. Un amministratore può creare utenze per i propri dipendenti (sia addetti alla sala, che addetti alla cucina, che supervisori). Al primo accesso, ogni utente deve re-impostare la password inserita dall'amministratore, scegliendo una password diversa.

2. Un amministratore può personalizzare i dettagli della propria attività di ristorazione, specificando nome, numero di telefono, indirizzo, e opzionalmente caricando un logo.

3. Un amministratore (o un supervisore) può personalizzare il menù dell'attività di ristorazione. In particolare, l'utente può creare e/o eliminare elementi dal menu. Ciascun elemento è caratterizzato da un nome, un costo, una descrizione, e un elenco di allergeni comuni. Inoltre, è possibile organizzare gli elementi del menù in categorie personalizzabili (e.g.: primi, dessert, prima di pesce, bibite, etc.), e definire l'ordine con cui gli elementi compaiono nel menù. In fase di inserimento, è richiesto l'autocompletamento di alcuni prodotti (e.g.: bibite o preconfezionati) utilizzando open data come quelli disponibili in https://it.openfoodfacts.org/data.

5. È possibile, per ristoranti che operano in località turistiche, specificare il nome e la descrizione di ciascun elemento del menù in una seconda lingua

9. Un addetto alla cucina (o un supervisore) può inoltre tenere traccia dell'inventario della dispensa. In particolare, l'utente può inserire/rimuovere prodotti dalla dispensa. Ciascun prodotto è caratterizzato da un costo di acquisto, da un nome, da una descrizione, e da una quantità (in Kg oppure in Litri, a seconda della tipologia di prodotto). In fase di inserimento, è apprezzato l'autocompletamento di alcuni prodotti (e.g.: bibite o altri preconfezionati) utilizzando open data come quelli disponibili in https://it.openfoodfacts.org/data.

13. Un supervisore o un amministratore può inserire nel sistema degli avvisi, che vengono visualizzati da tutti i dipendenti. Ciascun dipendente può marcare un avviso come "visualizzato" e nasconderlo.

4 INDIVIDUAZIONE TARGET UTENTI

Il bacino di utenza del Sistema comprende:

- Impiegati
- Supervisori
- Admin

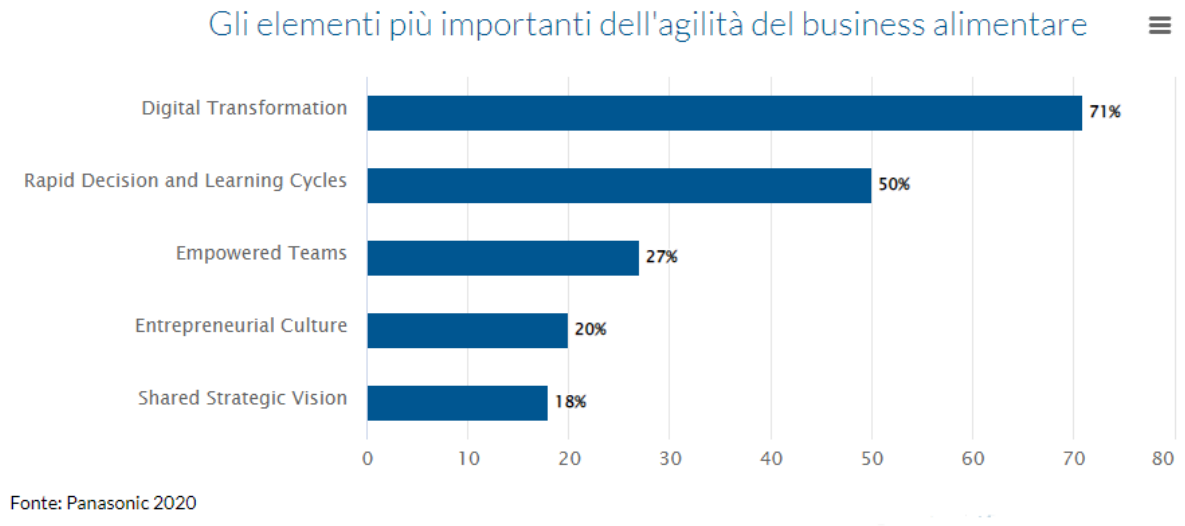
Secondo una ricerca del 2018 condotta da Oracle e NetSuite, il 73% dei ristoranti in Europa ha adottato almeno un software di gestione del ristorante. Inoltre, il 60% ha utilizzato un software per la gestione delle scorte, il 55% ha utilizzato un software per la gestione delle prenotazioni e il 44% ha utilizzato un software per la gestione del personale.

Perché utilizzare un applicazione per la gestione dei ristoranti?
Secondo financesonline.com:
- Oltre il 50% dei ristoranti ha aggiunto nuovi servizi a causa della pandemia (Square, 2021)

- Il 71% delle aziende alimentari ritiene che la trasformazione digitale sia un fattore cruciale per l'agilità aziendale, superiore alla quota di rapidi cicli decisionali e di apprendimento (50%), team responsabili (27%) e cultura imprenditoriale (20%) (Panasonic, 2020).

- il 90% dei ristoranti ritiene che una maggiore automazione aiuterebbe il proprio personale a concentrarsi maggiormente su compiti importanti (Square, 2021).

- Sfruttare le schede dei menu digitali porta potenzialmente a un aumento annuale delle vendite in negozio di $16.000 per unità (Presto, 2021).



Fonte: Panasonic 2020

PERSONAS

MARCO

"Con la giusta organizzazione si può tutto."

Età: 37
Lavoro: **Amministratore**
Situazione familiare: **Sposato, due figli**
Posizione: **Napoli, Italia**

Personalità

Introverso	Estroverso
Pensieroso	Sereno
Sensibile	Freddo
Seguece	Lea

Goals

- Aumentare il fatturato del ristorante
- Migliorare la qualità del servizio e del prodotto offerto
- Diffondere l'attività

Scoraggiamenti

- Non riuscire a mantenere la propria attività competitiva nel mercato
- Gestire il carico di lavoro e le scadenze
- Gestire il personale dell'attività

Bio

Sono Marco, l'amministratore delegato al ristorante di cucina tradizionale italiana dove lavoro. Ho molta esperienza nel settore della ristorazione e mi piace gestire al meglio il nostro locale. Sono sempre attento alle esigenze dei nostri clienti e faccio sempre il possibile per soddisfare ogni loro richiesta. Inoltre, sono sempre pronto a dare una mano dove servire e non ho mai paura di mettermi in gioco. Fuori dal lavoro, sono un appassionato di sport e trascorro molto tempo all'aria aperta. Amo anche viaggiare e scoprire nuove culture e cucine. Inoltre amo la tecnologia e soprattutto la sua applicazione pratica nei settori della ristorazione.

Motivazioni

Volontà
Paura
Crescita
Potere
Sociale

Necessità sul lavoro

- Organizzazione
- Precisione
- Professionalità
- Assistenza tecnico-finanziaria

LUCA

"A tavola perdonerei chiunque"

Età: **29**
Lavoro: **Chef**
Situazione Familiare: **Sposato, 2 figli.**
Posizione: **Napoli, Italia**

Personalità

Introverso	Estroverso
Pensieroso	Sereno
Sensibile	Freddo
Seguace	Leader

Goals

- Aprire un ristorante proprio
- Viaggiare con la sua famiglia
- Comprare una casa

Scoraggiamenti

- Lavorare con collaboratori poco efficienti
- Paura di essere troppo dipendente dal capo

Bio

Luca ha iniziato a lavorare nell'ambito della ristorazione all'età di 18 anni dopo aver conseguito il diploma " Enogastronomia e Ospitalità Alberghiera".
Ha lavorato già per altri ristoranti, anche a livelli importanti, non utilizzando mai tecnologie.

Motivazioni

Volontà
Paura
Crescita
Potere
Sociale

Necessità sul lavoro

- Ambiente tranquillo
- Lavoro efficiente

Marina

"I rapporti umani sono alla base di qualsiasi buona relazione, tra lavoratori e con i clienti."

Età: **25**
Lavoro: **Cameriera**
Situazione familiare: **Fidanzata, nessun figlio**
Posizione: **Napoli, Italia**

Personalità

Introversa	Estroversa
Pensierosa	Serena
Sensibile	Freddo
Seguace	Leader

Goals

- Aumentare la responsabilità lavorativa del proprietario
- Sviluppare le proprie abilità di comunicazione e di lavoro di squadra

Scoraggiamenti

- Lavorare in orari irregolari e/o a turni, che possono essere difficili da gestire
- Non avere un supporto adeguato dai superiori o dai colleghi
- Non ricevere una retribuzione adeguata al proprio lavoro e alle proprie responsabilità

Bio

Mi chiamo Marina e sono un'addetta alla sala in un ristorante di cucina tradizionale italiana. Amo il mio lavoro perché mi permette di interagire con i clienti e di aiutarli a scegliere il miglior piatto in base alle loro preferenze. Sono sempre attenta ai dettagli e mi sforzo di rendere ogni esperienza di cena indimenticabile. Inoltre, sono molto organizzato e mi piace mantenere la sala in ordine e pulita. Fuori dal lavoro, mi piace trascorrere del tempo con la mia famiglia e i miei amici, cucinare e viaggiare.

Motivazioni

Volontà
Paura
Crescita
Potere
Sociale

Necessità sul lavoro

- Ottima coordinazione
- Buon rapporto con i sottoposti

Giuseppe

"La qualità prima di tutto"

Età: **29**
Lavoro: **Supervisore**
Situazione familiare: Sposato
Posizione: **Napoli, Italia**

Personalità

Introverso	Estroverso
Pensieroso	Sereno
Sensibile	Freddo
Seguace	Leader

Goals

- Fare carriera
- Imparare una nuova lingua
- Migliorare le esperienze lavorative dei dipendenti

Scoraggiamenti

- Paura del cambiamento a causa della nuova era digitale
- Paura di sentirsi costantemente sotto pressione a causa delle sue responsabilità
- Conflitti con colleghi o con il manager

Bio

Sono Giuseppe, il supervisore del ristorante di cucina tradizionale italiana dove lavoro. Ho molta esperienza nel settore della ristorazione e mi piace gestire il team di lavoro per garantire che il nostro locale funzioni al meglio. Sono sempre attento alle esigenze dei nostri clienti e faccio del mio meglio per assicurarmi che ogni esperienza di cena sia indimenticabile. Inoltre, mi piace lavorare a stretto contatto con il ristorante del team per risolvere eventuali problemi e garantito che tutte le funzioni alla perfezione. Fuori dal lavoro, mi piace trascorrere del tempo con la mia famiglia e i miei amici, praticando sport e viaggiare.

Motivazioni

Volontà
Paura
Crescita
Potere
Sociale

Necessità sul lavoro

- Ambiente di lavoro positivo
- Comunicazione efficace ed efficiente
- Eventuale supporto e mentoring

5 ANALISI DEI REQUISITI

5.1 Requisiti funzionali

APP MOBILE

• Il sistema deve permettere il login all'utente, in particolare: l'utente deve avere la possibilità di accedere utilizzando indirizzo e-mail e password (resettata dopo il primo accesso, ad eccezione dell'amministratore).

• Il sistema deve permettere all'utente di visualizzare gli avvisi pubblicati dall'amministratore dell'attività, consentendo di marcare un avviso come "read/unread" e/o di nasconderlo dalla pagina.

• Il sistema deve permettere di tenere traccia dell'inventario della dispensa. In particolare, l'utente può inserire/rimuovere prodotti dalla dispensa.

• Il sistema deve permettere in fase di aggiunta l'autocompletamento di prodotti.

• Il sistema deve permettere di visualizzare le informazioni dell'account e di modificare (se serve) nome, cognome e data di nascita.

• Il sistema deve dare la possibilità di effettuare il log out.

SOFTWARE DESKTOP

• Il sistema deve permettere all'utente amministratore di registrarsi utilizzando il proprio indirizzo e-mail; inoltre, durante la registrazione, l'utente deve inserire dei campi obbligatori quali nome, password e data di nascita; in caso di previa avvenuta registrazione, l'utente amministratore potrà accedere al suo account mediante login con e-mail e password.

• Il sistema deve permette all'utente di pubblicare nuovi avvisi, prenderne visione e/o eliminarne di già presenti.

• Il sistema deve permettere all'utente di visualizzare il menù del ristorante compreso di nomi, descrizioni, prezzi e allergeni comuni. Sarà inoltre possibile aggiungere un nuovo piatto aggiungendogli un nome, una descrizione, una lista di allergeni comuni, un prezzo e una categoria (e.g. primo, secondo...). Sarà anche possibile creare una nuova categoria durante la creazione di un piatto (e.g. primo di terra, primo di mare...). Inoltre, ogni dettaglio dei piatti potrà essere aggiunto (a scelta dell'utente) in una seconda lingua se il ristorante è situato in una località turistica.

• Il software deve offrire la possibilità di riordinare a piacimento i piatti presenti nelle varie categorie oppure di cambiare l'ordine delle categorie stesse all'interno del menù.

• In caso in cui si voglia cercare un piatto specifico all'interno del menù, si potrà usufruire della barra di ricerca per immettere parti del nome o della descrizione del piatto oppure si può ricorrere all'uso del filtro presente. Si potrà inoltre massimizzare o minimizzare le categorie per scegliere se visualizzare tutti i suoi piatti oppure se visualizzare un elenco delle categorie minimizzate.

• Il sistema deve consentire di aggiungere nuove persone alle risorse umane inserendo la loro nome, cognome, mail e data di nascita oltre alla possibilità di assegnare uno e un solo compito tra: supervisore, addetto alla sala e addetto alla cucina. Si potrà inoltre sollevare un dipendente dal suo incarico, promuoverlo o declassarlo.

• Il sistema deve offrire la possibilità di cambiare il nome, numero di telefono e indirizzo del ristorante e opzionalmente può far caricare un logo dall'amministratore.

•Il sistema dovrà inoltre offrire la possibilità di entrare nelle impostazioni del proprio account e di effettuare un logout.

5.2 Requisiti non funzionali

• Il sistema deve garantire tempi di risposta inferiori a tre secondi per qualunque operazione compiuta sul menù.

• Il sistema deve utilizzare colori coerenti che suscitino senso di fiducia nell'utente.

• Le informazioni inserite dall'utente devono rispettare determinati vincoli:
- Nome: lunghezza compresa tra 3 caratteri e 15 caratteri;
- Cognome: lunghezza compresa tra 3 caratteri e 15 caratteri;
- E-mail: formato coerente con la standard signature;
- Password: lunghezza minima otto caratteri con compreso almeno una lettera maiuscola, un carattere speciale e un numero;

• Il sistema deve garantire la portabilità su tutti i sistemi Android e windows.

• Il sistema si obbliga a garantire il salvataggio delle password utente in modo crittografato.

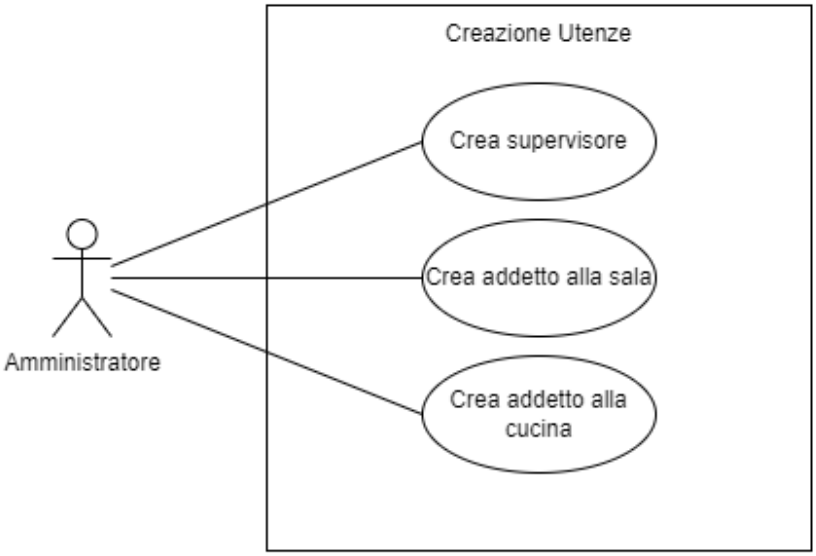
• Il sistema si deve occupare di garantire un'alta usabilità mediante l'utilizzo di appositi criteri di design attenendosi alle leggi della Gestalt.

5.3 Requisiti di dominio

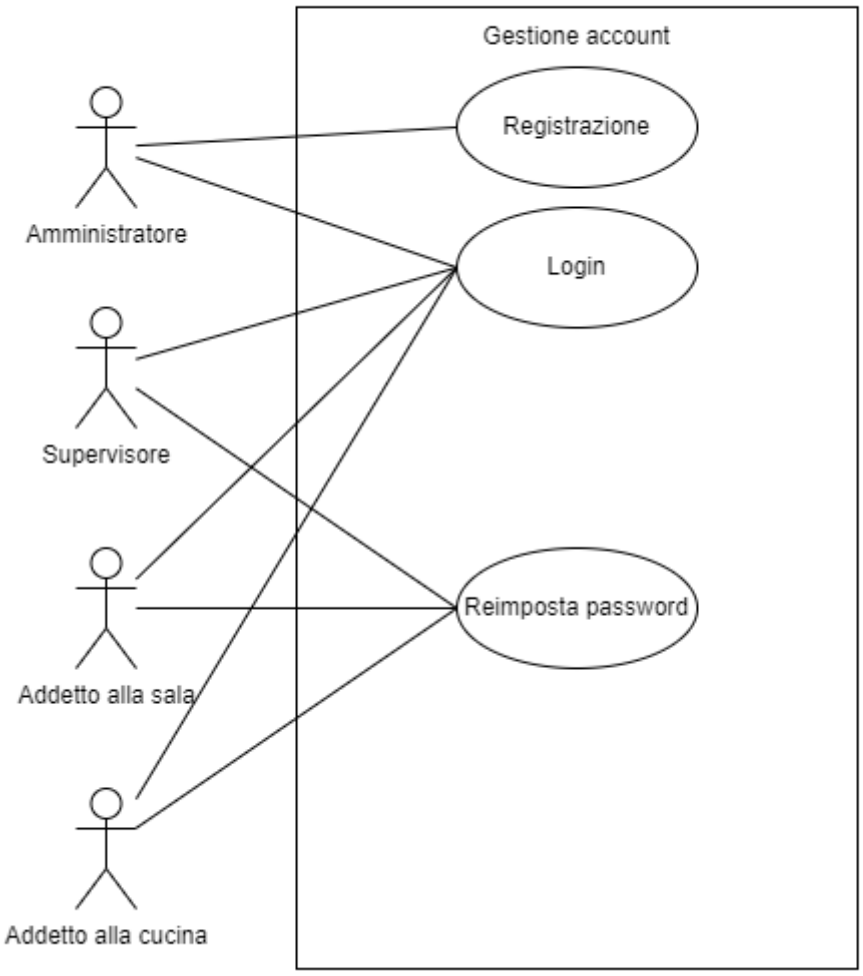
• Il software deve essere sviluppato secondo lo standard ISO 9126.

5.4 Use Case Diagram

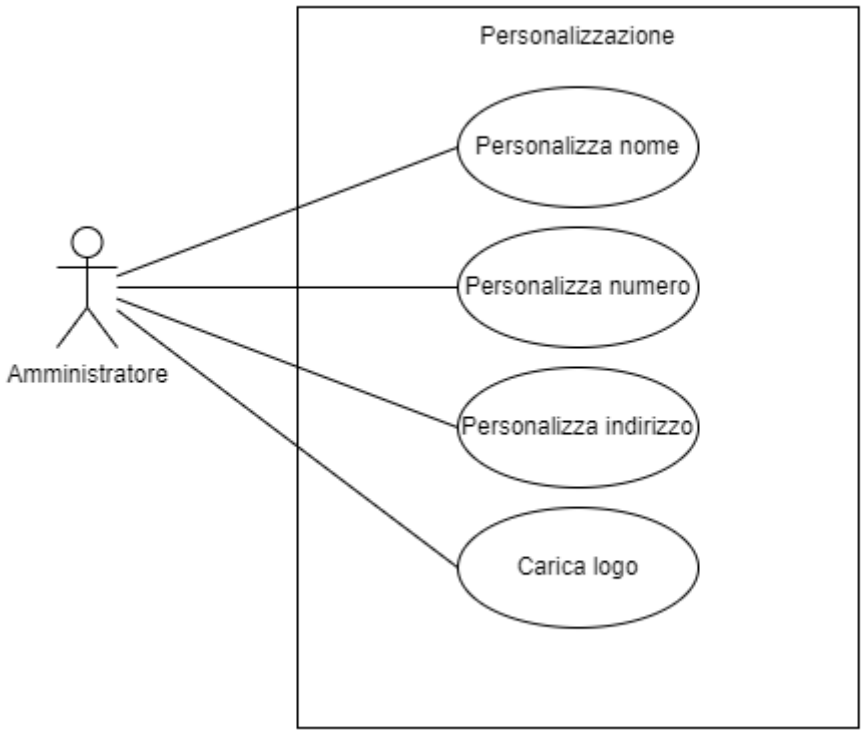
Use Case 1: Creazione Utenze



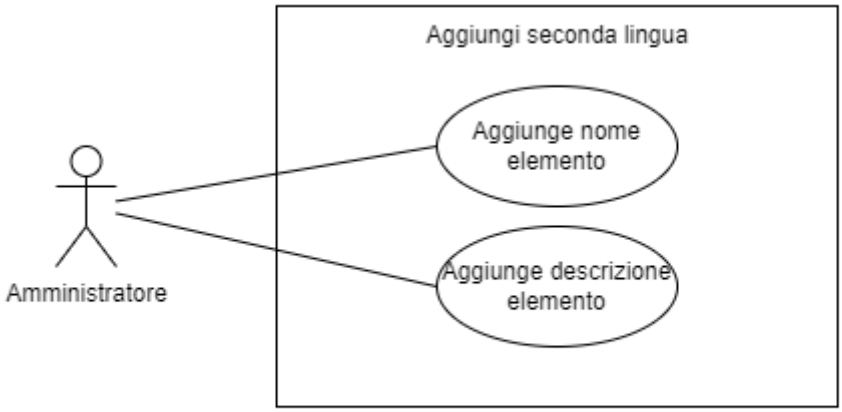
Use Case 2: Gestione Account



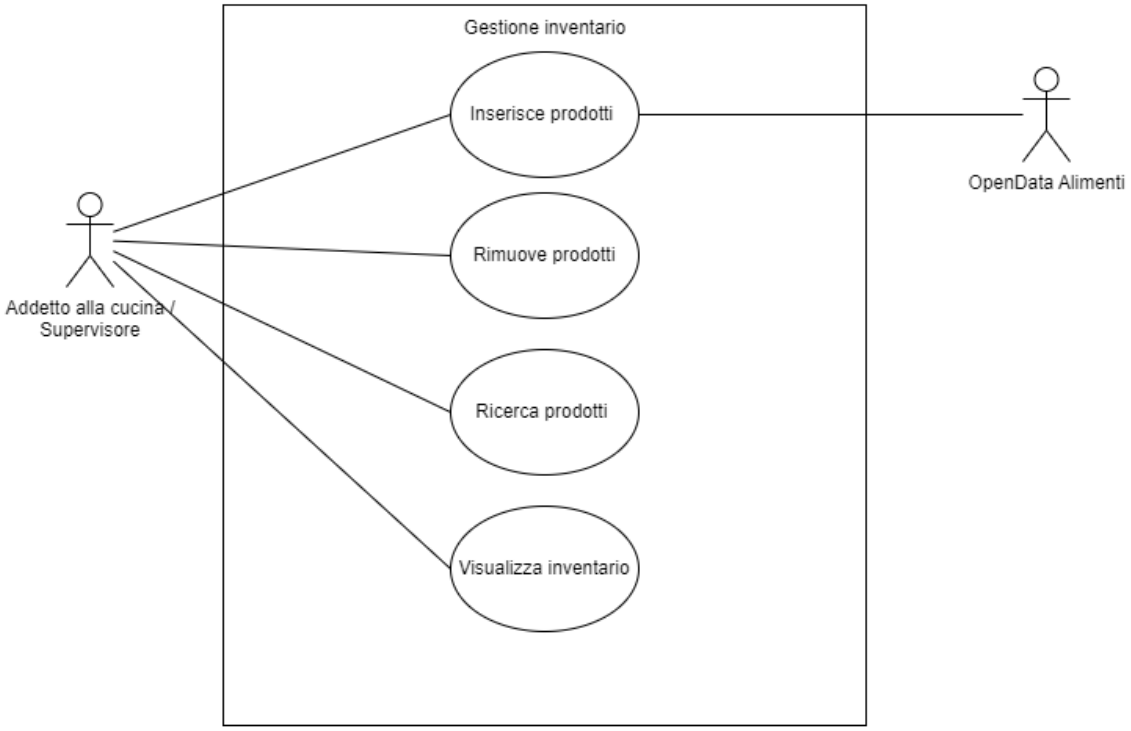
Use Case 3: Personalizzazione attività



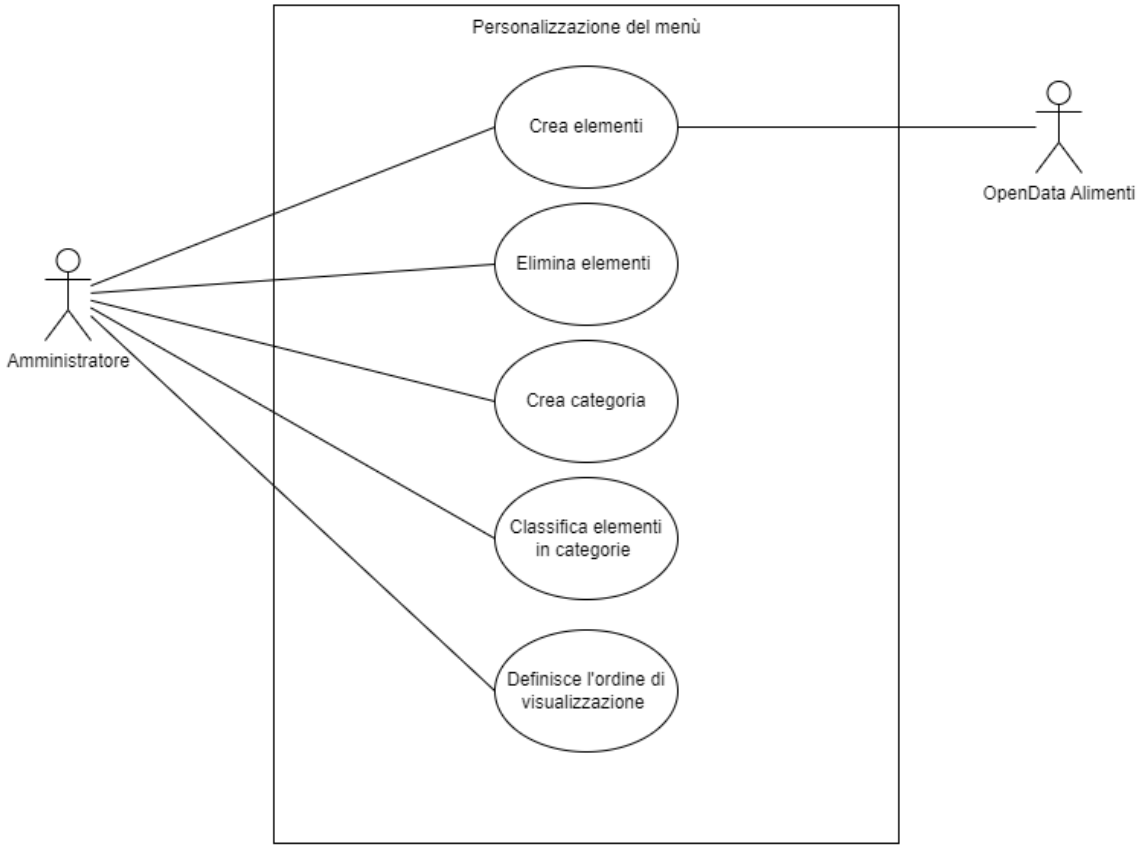
Use Case 4: Aggiungi seconda lingua



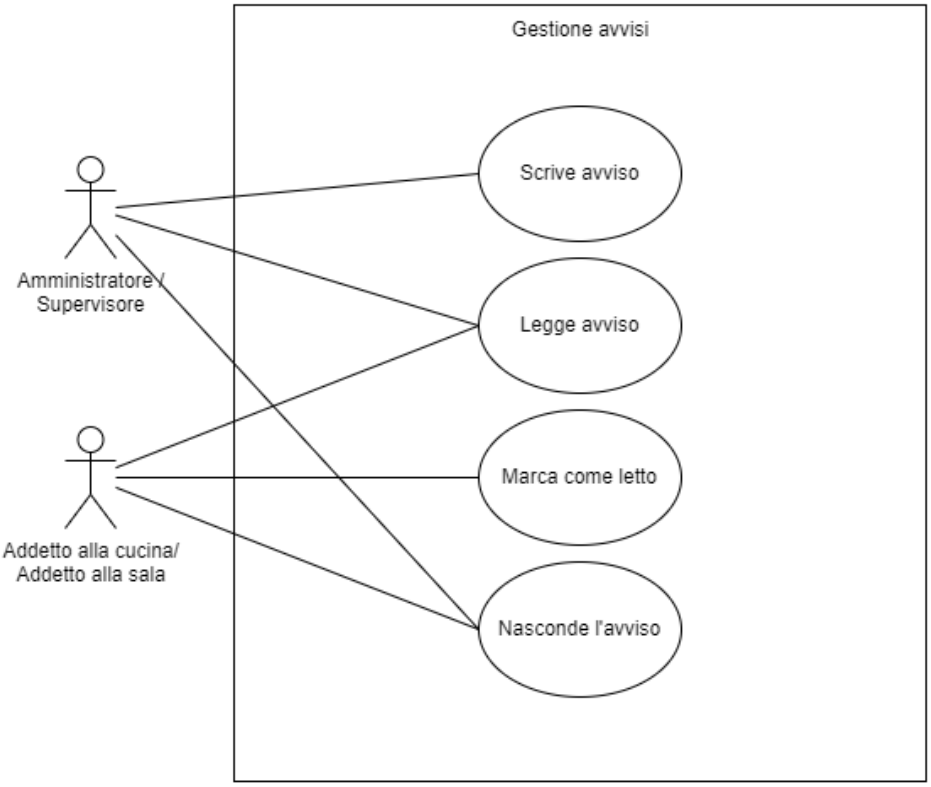
Use Case 5: Gestione Inventario



Use Case 6: Personalizzazione menù



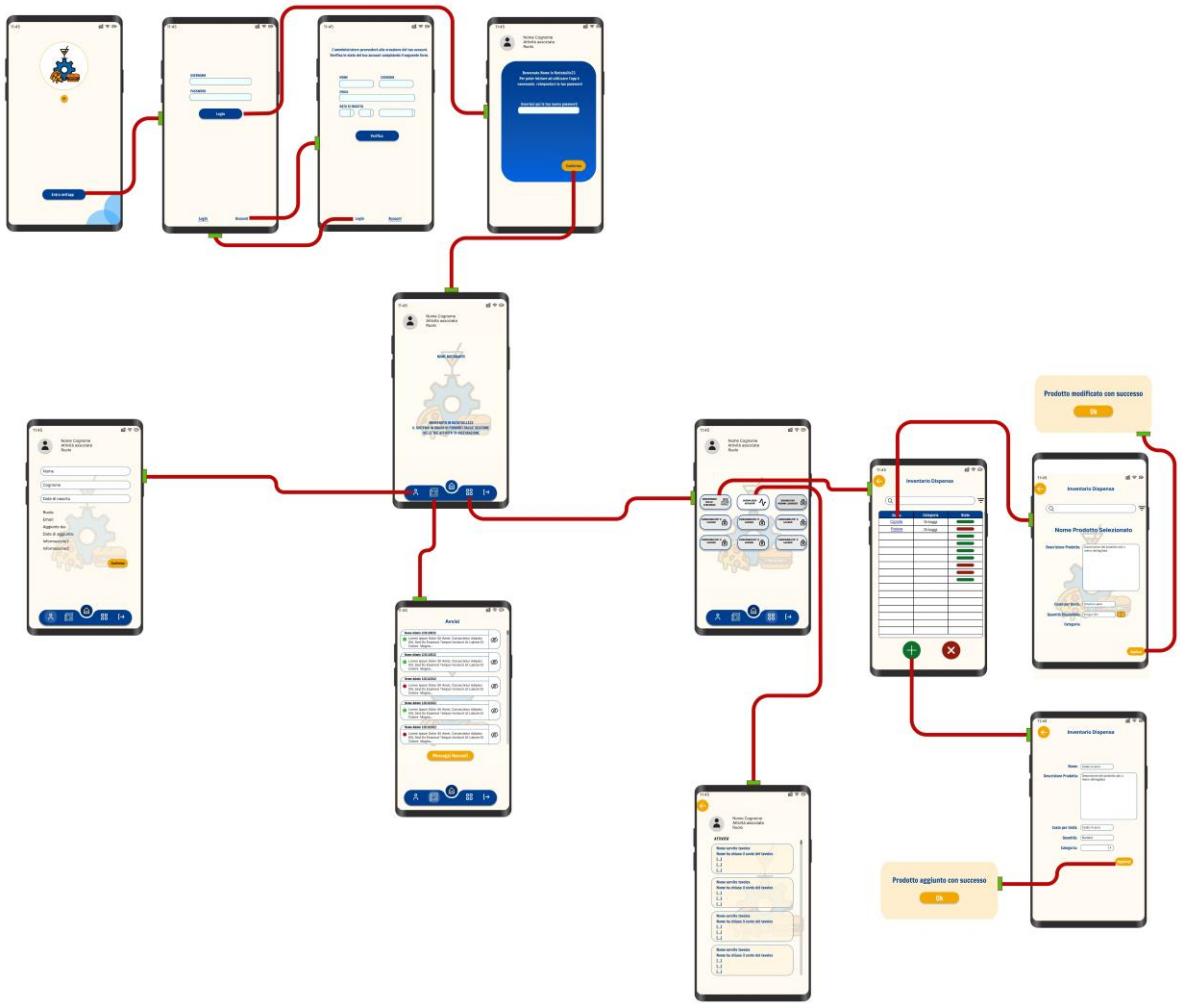
Use Case 7: Gestione Avvisi



5.5 MockUp

È possibile utilizzare il prototipo realizzato grazie a Figma.

MockUp app mobile



L'UI dell'app mobile è caratterizzata da:

- Una palette ben scelta di colori: #003F91, #F2A900, #FFFFFF
- Elementi dai bordi arrotondati
- Bottom-Navigation-Bar
- Organizzazione ad albero

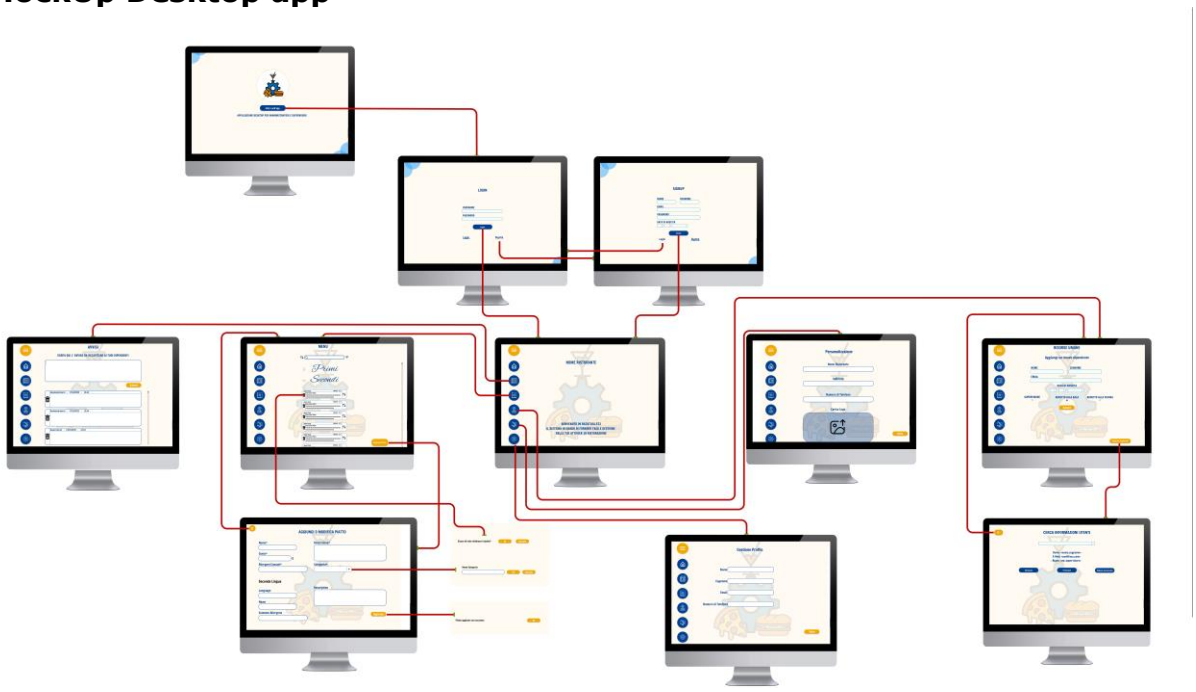
È stata pensata cercando di:

1. Offrire quanti più feedback possibile
2. Utilizzare quanto meno la memoria a breve termine, grazie a combobox, suggerimenti, pulsanti che facilitano interi compiti…
3. Garantire un'ottima usabilità il che comprende alla base un'efficacia consistente per le necessità richieste, un'alta dose di efficienza per minimizzare i costi, gli sforzi e il tempo dell'utente
4. Mostrare affordance per ogni funzionalità rendendo il tutto quanto più intuitivo possibile all'interno dell'applicazione così da creare quanti meno disagi, fraintendimenti e sprechi di tempo nel suo utilizzo
5. Mantenere un'alta coerenza come da convenzioni
6. Offrire in ogni momento all'utente di effettuare un "ritorno indietro" / "undo" sulle operazioni che sta compiendo in modo che in caso di errore,

fraintendimenti o variazioni dell'idea originale, l'utente possa annullare l'operazione in corso e tentarne una nuova

7. Utilizzare leggi della Gestalt

MockUp Desktop app



5.6 Templates di Cockburn (2 casi d'uso)

Use Case #1	**Aggiungi un nuovo alimento in dispensa e modifica la sua quantità**		
Goal in context	Un addetto alla cucina vuole aggiungere un alimento nuovo in dispensa e successivamente ne modifica le quantità.		
Precondition	L'utente deve possedere un account di tipo "addetto alla cucina/sala" e deve essere autenticato nell'app.		
Success End Condition	L'utente riesce ad aggiungere l'alimento da lui scelto e a modificarne la quantità.		
Failed End Condition	Il nome del prodotto è già presente nella dispensa al momento della sua creazione. Non tutti i campi sono compilati. Alcuni dei campi non rispettano i vincoli di lunghezza.		
Primary Actor	Addetto alla cucina/sala		
Trigger	L'addetto preme il pulsante "Lista di funzionalità" sul menu di navigazione		
Description	**Step #**	**Addetto alla cucina/sala**	**Sistema**
	1	Preme il tasto "Menu funzionalità" dal menu di navigazione	
	2		Mostra "M_Menu_Funzionalità"
	3	Preme "Inventario della Dispensa"	
	4		Mostra "M_Dispensa"
	5	Preme sul tasto "+"	
	6		Mostra "M_Aggiungi_Prodotto"
	7	Inserisce "Nome", "Descrizione", "Costo Unitario", "Quantità Disponibile" e seleziona "Categoria" e Preme "Aggiungi"	
	8		Mostra "Popup_Aggiunta"
	9	Preme "OK"	
	10		Mostra "M_Dispensa"
	11	Preme il link sul nome del prodotto precedentemente creato	
	12		Mostra "M_Prodotto"
	13	Modifica la quantità inserendo un altro numero al posto di quello presente e preme "Applica"	
	14		Mostra "Popup_Modifica"
	15	Preme "OK"	
	16		Mostra "M_Dispensa"
Extension #1 Durante la creazione	**Step #**	**Addetto alla cucina /sala**	**Sistema**
	7.1	Compila i campi e Preme	

di un nuovo prodotto non vengono riempiti tutti i campi		"Aggiungi"	
	8.1		Mostra etichetta di errore "Non tutti I campi sono stati compilati" e torna allo step 7 dello scenario principale
#2 Durante la modifica di un prodotto non vengono riempiti tutti I campi	**13.2**	Modifica la quantità inserendo un altro numero al posto di quello presente e preme "Applica"	
	14.2		Mostra etichetta di errore "Non tutti I campi sono stati compilati" e torna allo step 13 dello scenario principale
#3 La creazione di un prodotto non può essere valida se ne esiste già uno con lo stesso nome in dispensa	**7.3**	Inserisce "Nome", "Descrizione", "Costo Unitario", "Quantità Disponibile" e seleziona "Categoria" e Preme "Aggiungi"	
	8.3		Mostra etichetta di errore "Prodotto già presente in dispensa" e torna allo step 7 dello scenario principale
#4 Vincoli di lunghezza non rispettati	**7.4**	Durante la compilazione dei campi si raggiunge il limite massimo di lunghezza	
	8.4		Mostra etichetta "Limite caratteri raggiunto" e torna allo step 7 dello scenario principale
#5 Preme il tasto indietro	**7.5**	Preme il tasto "Indietro"	
	8.5		Torna allo step 5 dello scenario principale
#6 Preme il tasto indietro	**13.6**	Preme il tasto "Indietro"	
	14.6		Torna allo step 5 dello scenario principale
Subvariation	**Step #**	Addetto alla cucina /sala	Sistema
	13.1	Modifica la quantità premendo il tasto "+" o il tasto "-" tante volte quanto si desidera e preme "Applica"	
	14.1		Vai allo step 14 dello scenario principale

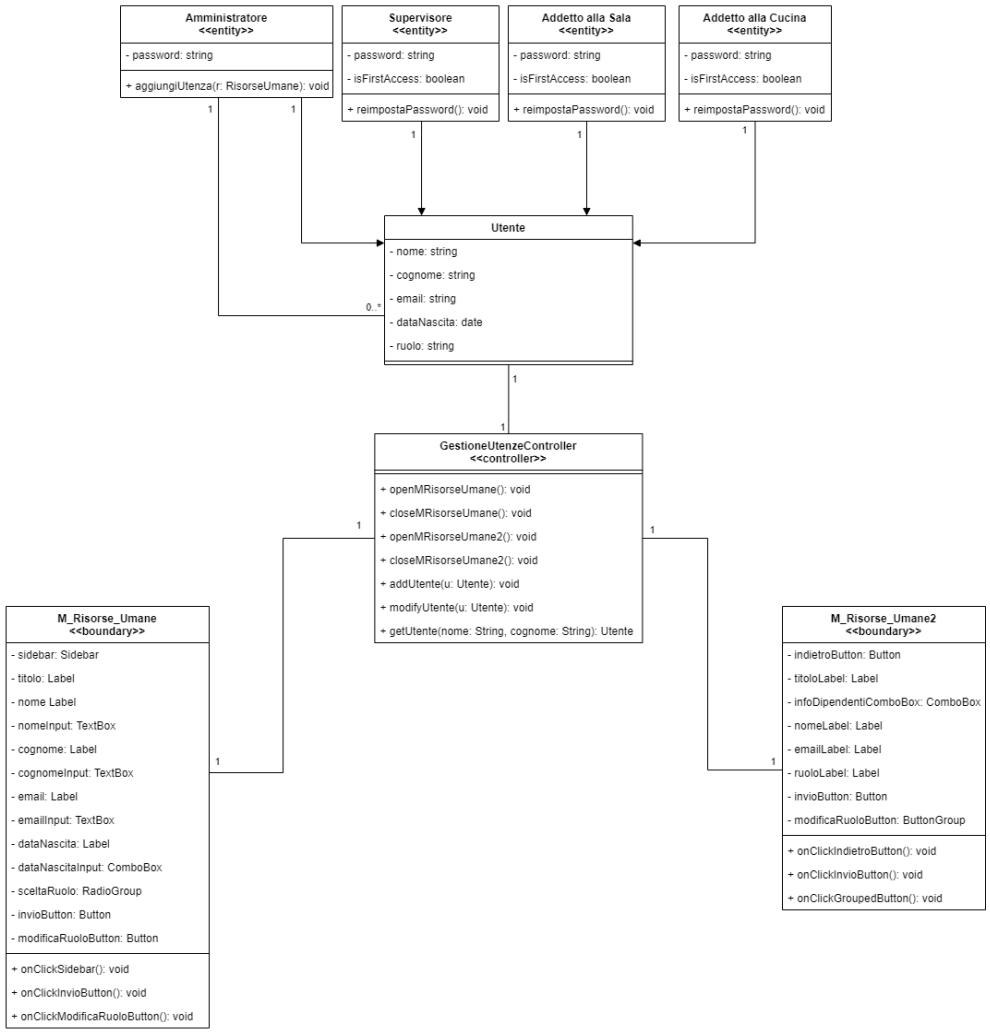
Use Case #2	**Crea e visualizza piatto creato in una nuova categoria con aggiunta di una seconda lingua**		
Goal in context	Un supervisore vuole creare un piatto da attribuire ad una categoria non ancora presente nel sistema e visualizzarlo nel menù sia in italiano che nella seconda lingua.		
Precondition	L'utente deve possedere un account di tipo "Supervisore" o "Amministratore" e deve essere autenticato nel software desktop.		
Success End Condition	L'utente riesce a creare e visualizzare il piatto che desidera nella categoria da lui creata e eventualmente anche nella seconda lingua inserita.		
Failed End Condition	Il piatto è già presente nel menu. Non tutti i campi sono compilati. Alcuni dei campi non rispettano i vincoli di lunghezza.		
Primary Actor	Amministratore/Supervisore		
Trigger	L'utente preme il tasto "Menu" sul menu di navigazione		
Description	Step #	Amministratore/Supervisore	Sistema
	1	Preme tasto "Menu" sul menu di navigazione	
	2		Mostra "M_Menu"
	3	Preme il tasto "Aggiungi Piatto"	
	4		Mostra "M_Aggiungi_A_Menu"
	5	Inserisce nome, descrizione, costo, allergeni	
	6	Click su comboBox "Categoria" e seleziona "Nuova Categoria"	
	7		Apre pop-up "Popup_Menu"
	8	Inserisce il nome della nuova categoria e preme "OK"	
	9		Mostra "M_Aggiungi_A_Menu"
	10	Seleziona la categoria creata dalla comboBox "Categoria"	
	11	Compila i campi "Language" con la lingua che si desidera inserire, "Name", "Description", "Common Allergens" avendo cura che combacino con la parte in italiano e preme "Aggiungi"	
	12		Mostra "M_Menu" con sovraimpresso "Popup_Aggiunta"
	13	Preme "Ok"	
	14		Chiudi "Popup_Aggiunta"
Extension	Step #	Amministratore/Supervisore	Sistema
#1 Durante la creazione del piatto preme "X"	5.1	Preme "X"	
	6.1		Mostra "M_Menu" e torna allo step 3 dello scenario principale
#2	8.2	Preme "Annulla"	

Preme “Annulla” durante la creazione della nuova categoria	9.2		Mostra "M_Aggiungi_A_Menu” con la categoria deselezionata
#3 Il piatto è già presente nel menu	11.3	Preme “Aggiungi”	
	12.3		Il nome del piatto coincide con uno già presente nel menu, quindi, mostra etichetta “Piatto già esistente”
#4 Non tutti I campi sono compilati	11.4	Preme “Aggiungi”	
	12.4		Non ha compilato tutti i campi, quindi, mostra etichetta “Compilare tutti i campi”
#5 Vincoli di lunghezza non rispettati	5.5	Durante la compilazione dei campi si raggiunge il limite massimo di lunghezza	
	6.5		Mostra etichetta “Limite caratteri raggiunto”

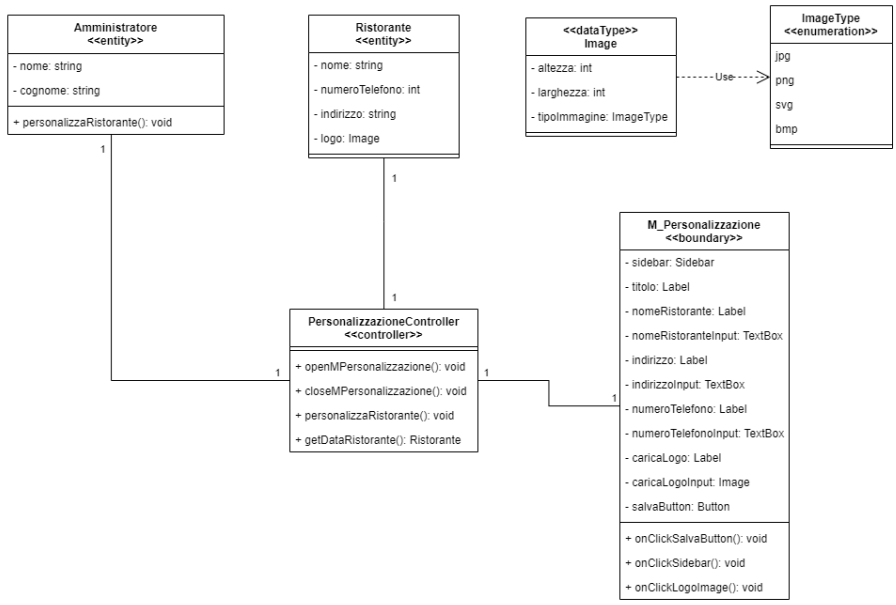
6 SPECIFICA DEI REQUISITI (MODELLO DI DOMINIO)

6.1 Class Diagram

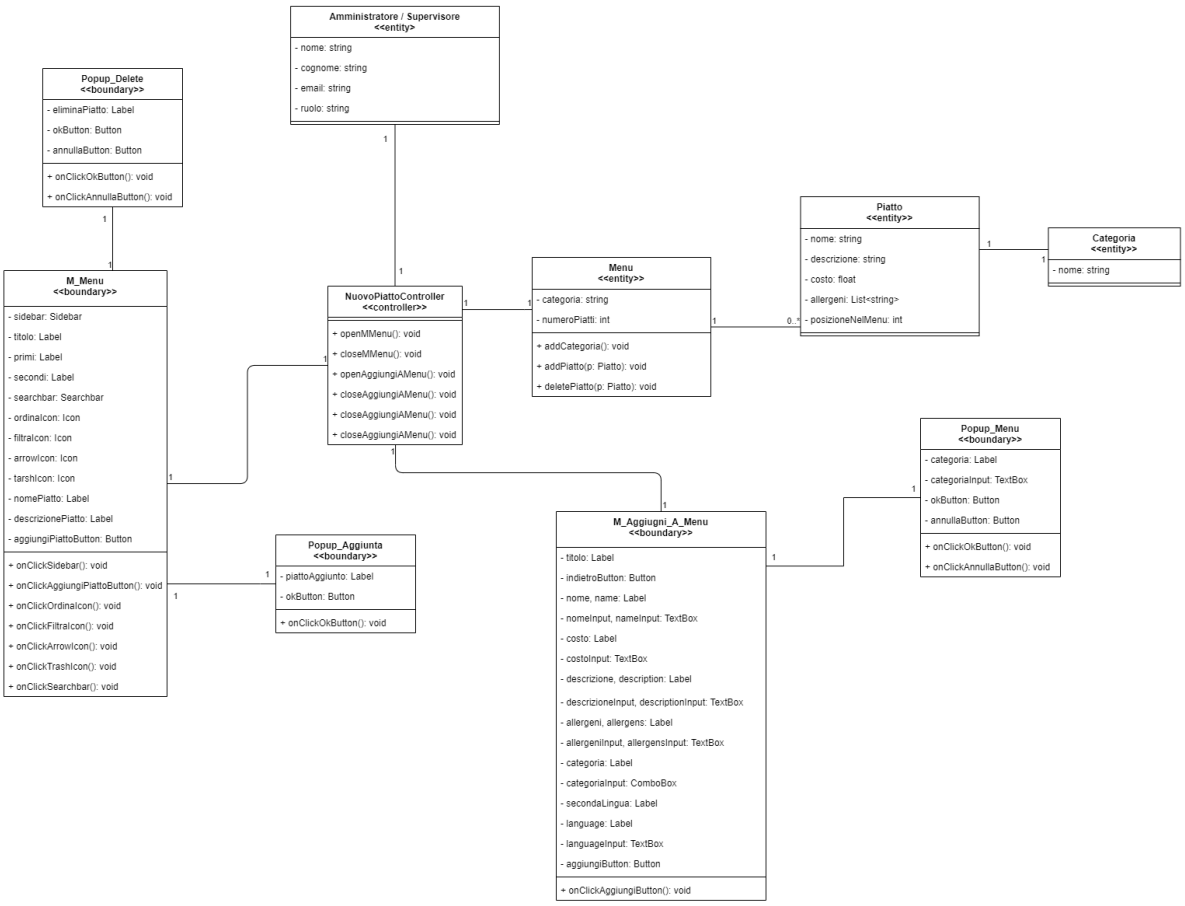
Gestione Utenze



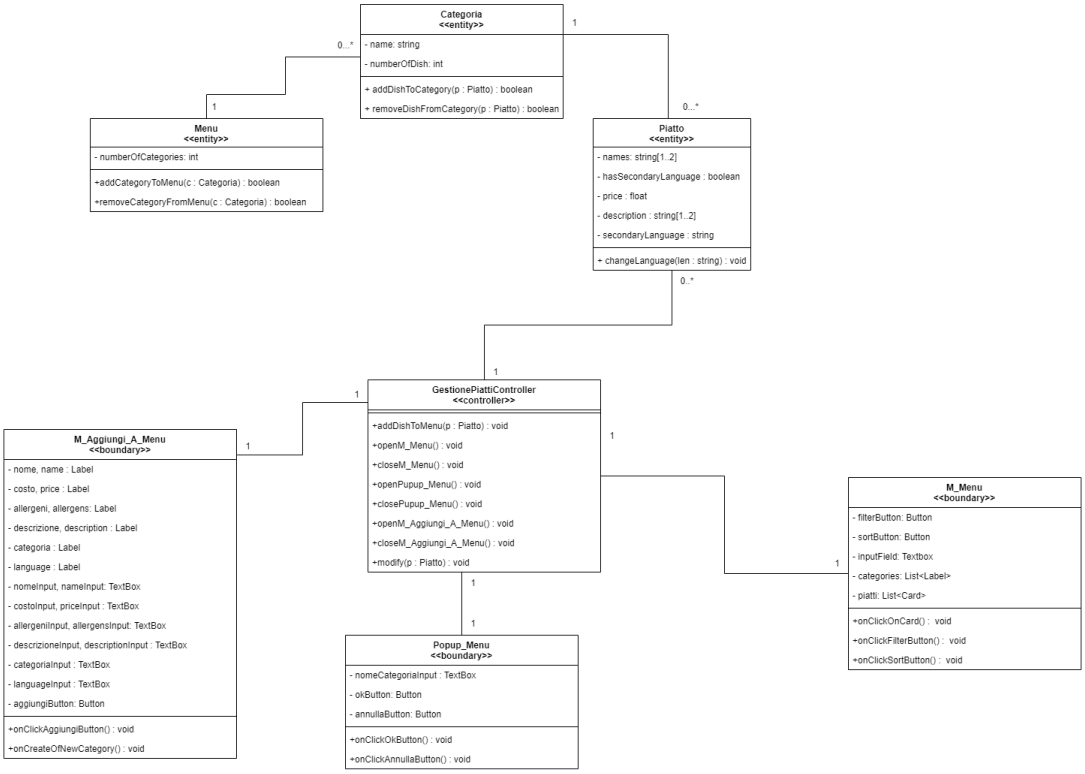
Personalizzazione attività



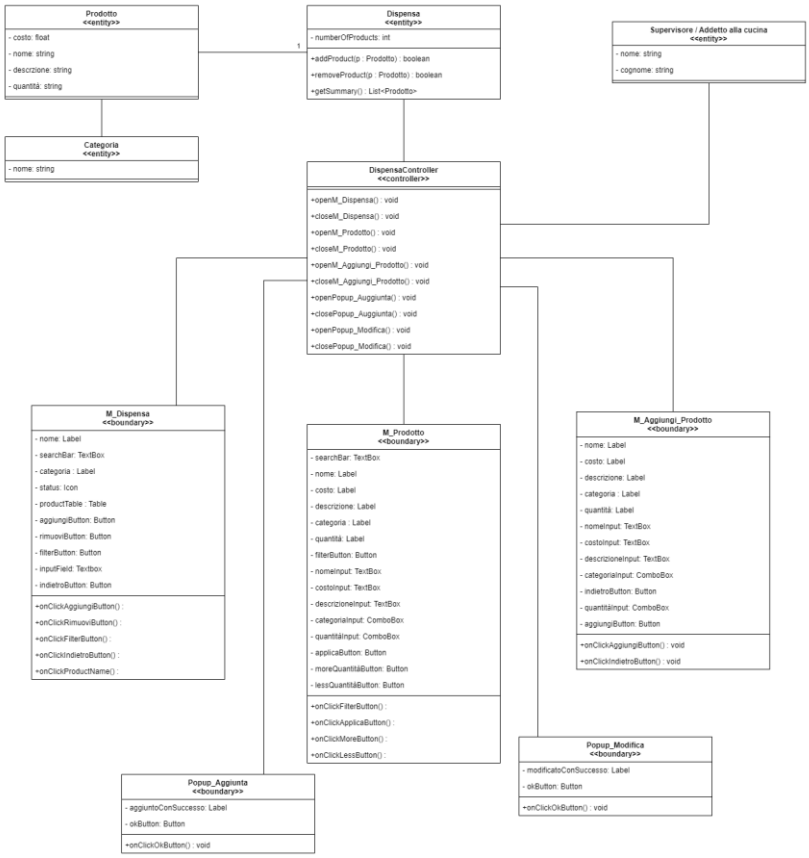
Gestione Menù



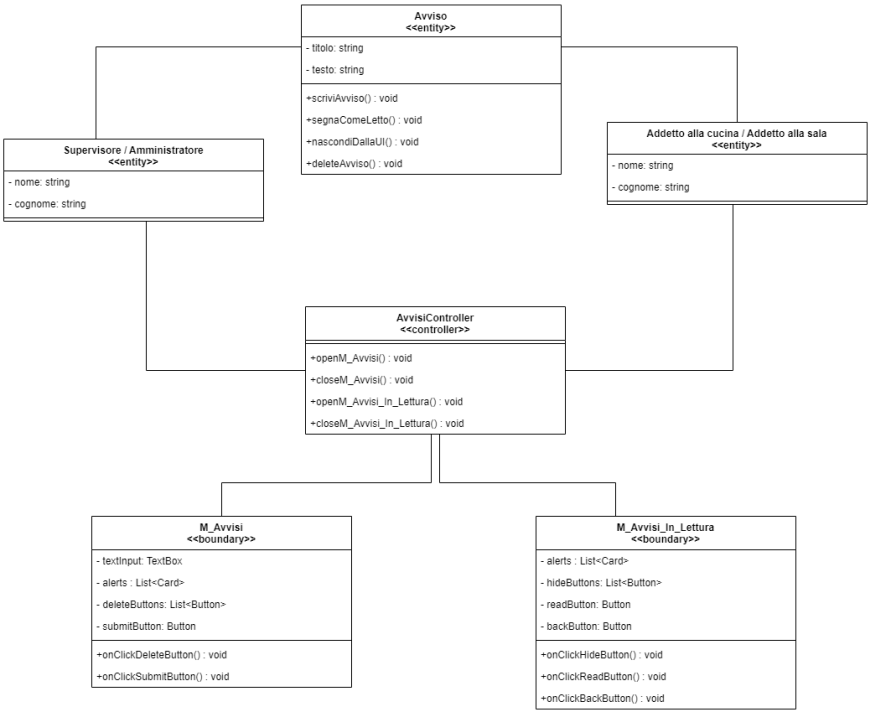
Scelta seconda lingua



Gestione inventario della dispensa

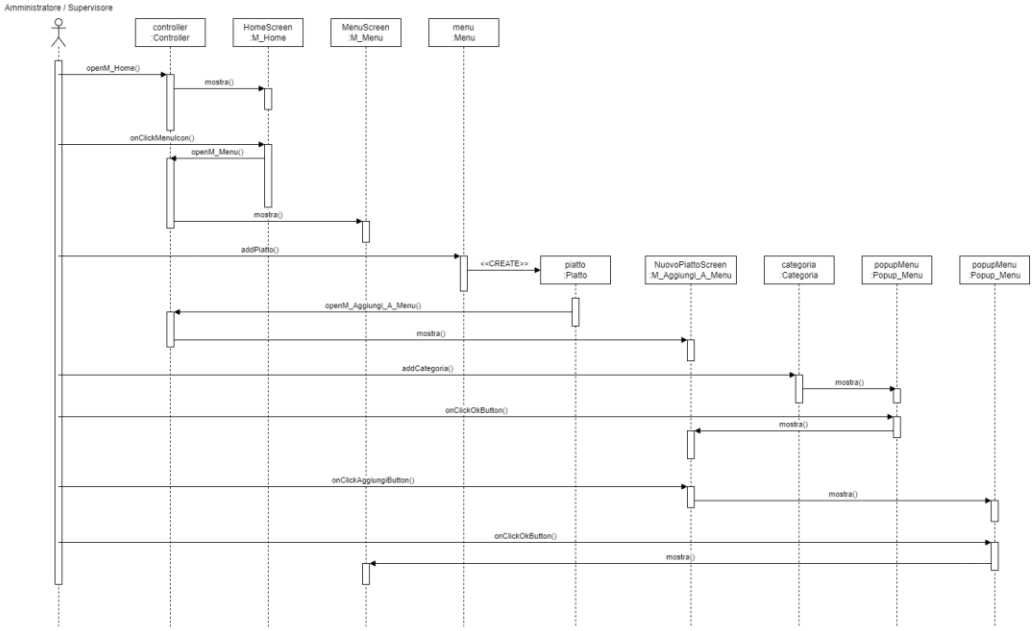


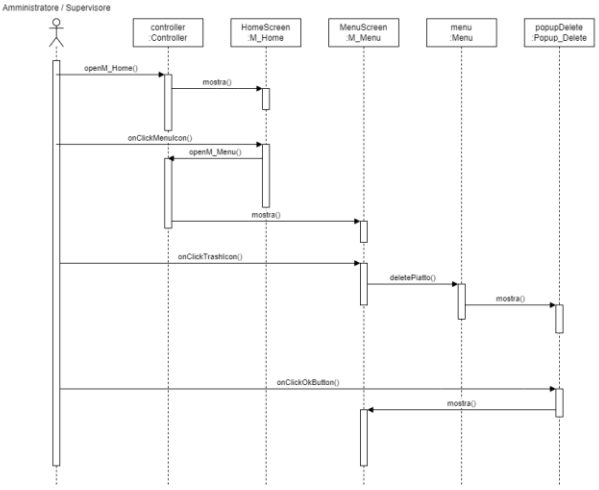
Gestione avvisi

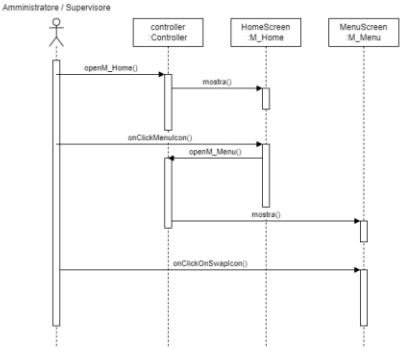


6.2 Sequence Diagram

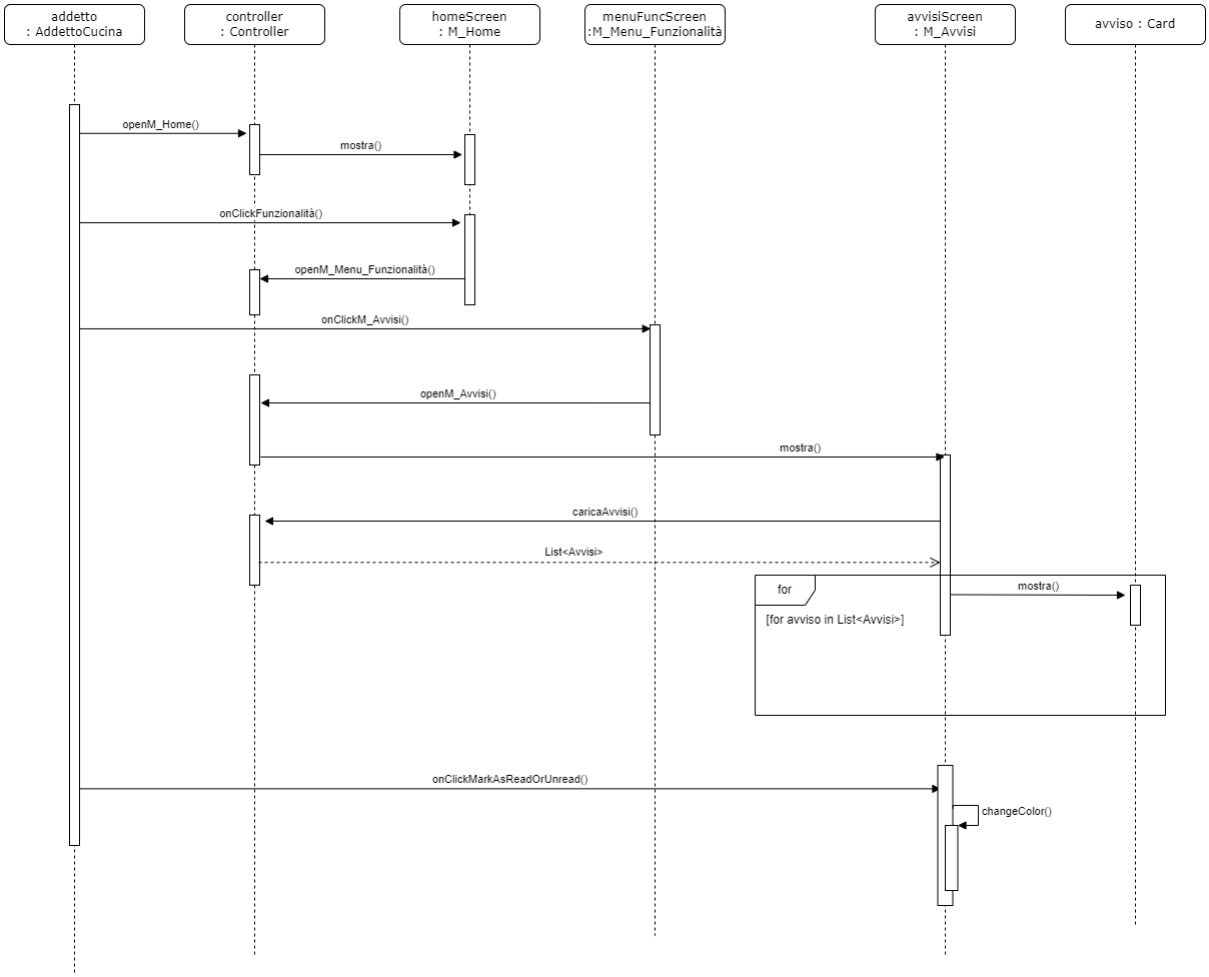
Punto 3 della traccia:

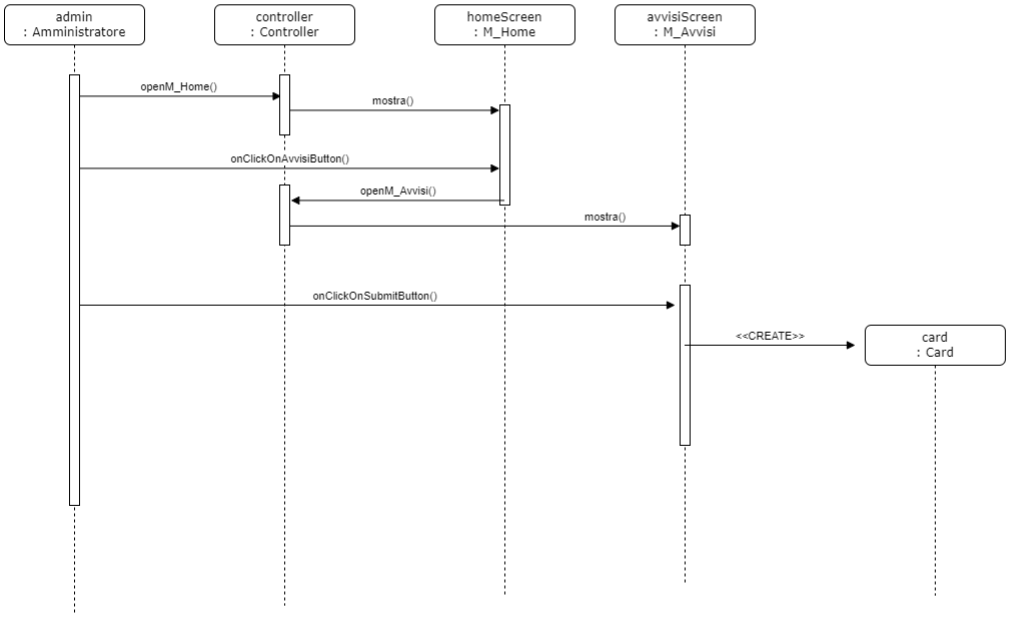






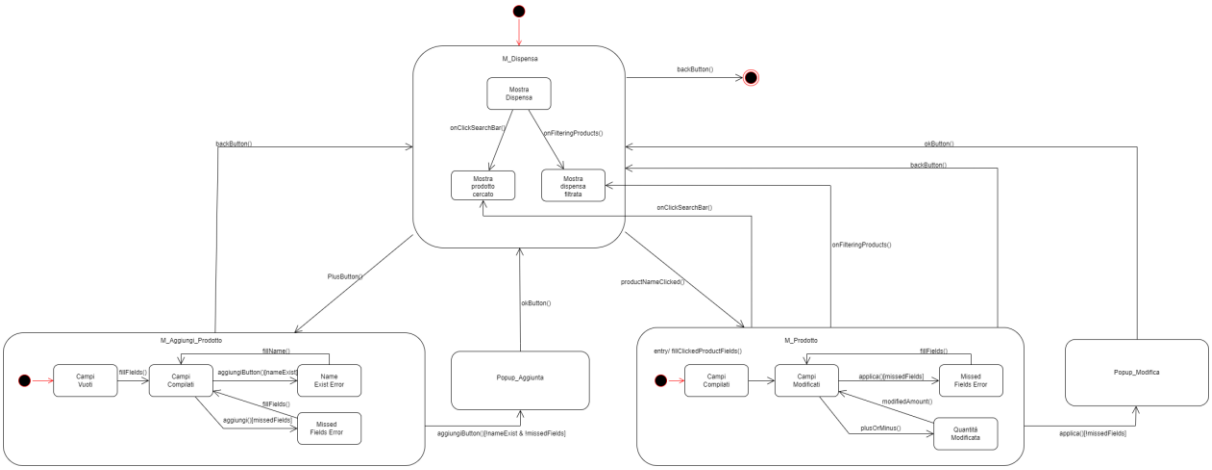
Punto 13 della traccia



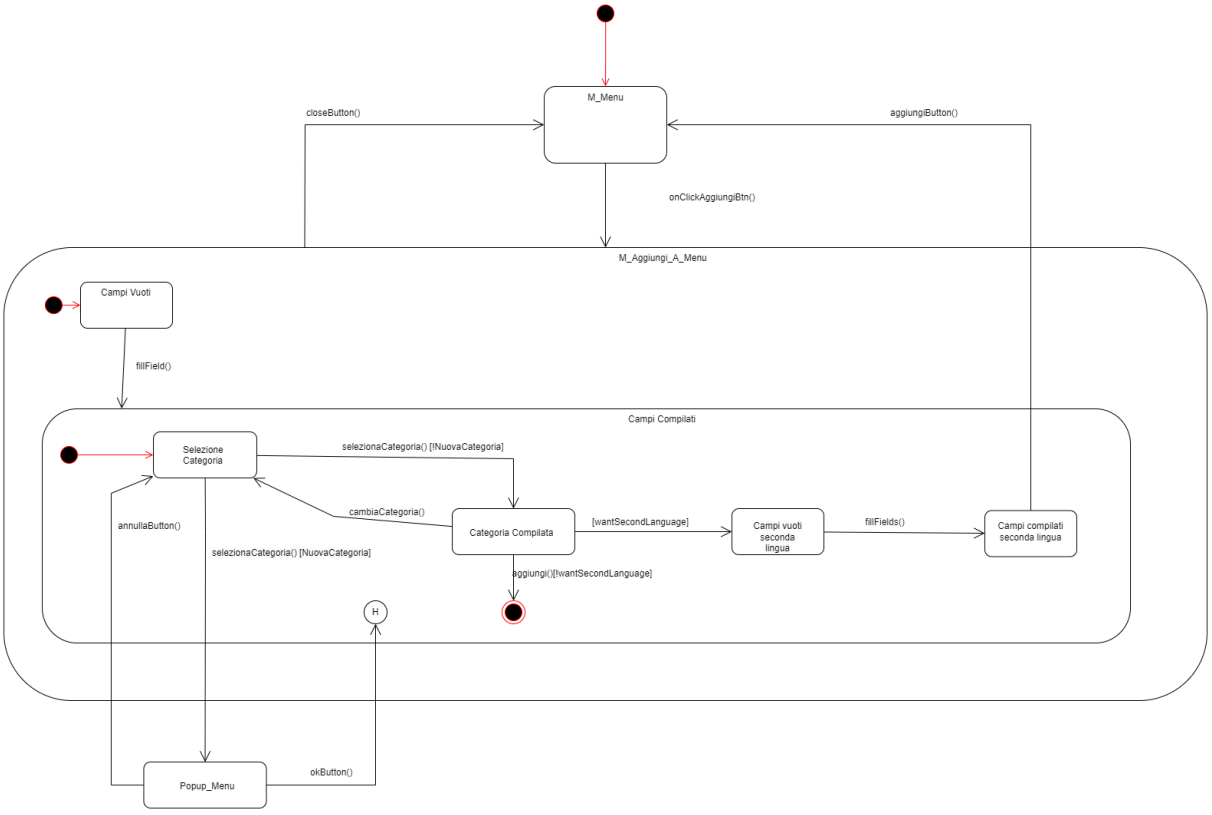


6.3 Statechart (2 casi d'uso) (prototipazione funzionale)

Gestione inventario dispensa



Gestione del menù



7 STUDIO USABILITA' A PRIORI

L'usabilità è valutata grazie alle funzionalità messe a disposizione dal prototipo realizzato mediante Figma.
Figma permette di rendere dinamici i mockup in modo tale che il sistema sembri effettivamente realizzato.
Le valutazioni in questa fase valgono solo per il prototipo, in quanto non è possibile simulare tutti i feedback e/o funzionalità aggiuntive.

VALUTAZIONE EURISTICA MEDIANTE ISO 9241

REGOLA	COMMENTO	PERTINENZA
Auto descrizione	Il sistema informa per ogni azione il cambiamento di stato del sistema. (e.g. hovering su bottoni, tocco e cambio schermata…). *Non è ancora possibile valutare al 100% tutti i feedback.*	7/10
Conformità alle aspettative	Il linguaggio utilizzato nelle frasi è molto vicino al linguaggio naturale, con frasi brevi e parole facenti parte del vocabolario comune. Il sistema garantisce almeno un modo per compiere una qualunque azione, talvolta (e.g. nel caso si voglia aggiungere e/o sottrarre prodotti nel magazzino) esistono più possibilità per lo stesso fine ma pensato per il solo scopo di velocizzare operazioni.	9/10
Controllabilità	Gli utenti possono cambiare stato del sistema in ogni modo con la possibilità di tornare indietro qualora l'azione compiuta non è quella desiderata e/o risulti involontaria, ci sono pochi vincoli stringenti dove l'utente effettivamente è obbligato a compiere un'azione.	8/10
Tolleranza verso gli errori	Non è possibile in questa fase visualizzare messaggi di errore. L'interfaccia è pensata per poter ridurre al minimo la possibilità di sbagliare non limitando però la libertà dell'utente.	Non valutabile
Adeguatezza al compito	Il grande numero di combobox presenti permette all'utente di minimizzare lo sforzo di memoria. Non esistono shortcut per utenti esperti.	9/10

Design minimalista ed estetico	Il design è pensato per essere sì minimalista ma riconoscibile, la palette di colori fa capire la funzionalità gestionale del sistema accostandola ai colori del settore della ristorazione. I colori principali suscitano nell'utente emozioni di affidabilità (blu) e ottimismo (giallo)	8/10
Adeguatezza all'apprendimento	Non presente	0/10

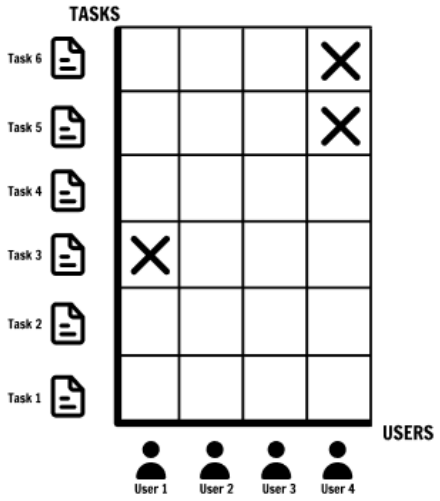
7.1 TEST DI USABILITA' (1)

Tabella dei compiti

Compito	Click Stimati	Tempo Stimato	Peso Compito (1 – 10)	Equivoci tollerato
Aumenta di un chilo la quantità di patate nella dispensa	5	31''	8	2
Crea un piatto nel menu in una categoria ancora non esistente	6	25''	9	2
Visualizza un messaggio dell'amministratore	3	25''	5	1
Scrivi un messaggio in qualità di amministratore	2	20''	7	1
Promuovi un dipendente	4	35''	7	2
Aggiungi un nuovo dipendente alle risorse umane	2	40''	7	2

	Compito 1				Compito 2				Compito 3				Compito 4				Compito 5				Compito 6			
Utente: Exp	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**
Utente 1: 7	5	0	25''	1	7	1	35''	1	5	2	23''	0	2	0	9''	1	8	4	45''	1	4	1	12''	1
Utente 2: 5	7	2	30''	1	6	0	21''	1	5	2	25''	1	7	6	27''	1	6	2	25''	1	2	0	5''	1
Utente 3: 7	5	0	32''	1	6	0	22''	1	6	3	26''	1	2	0	11''	1	7	3	26''	1	2	0	6''	1
Utente 4: 3	6	1	34''	1	7	1	30''	1	3	0	20''	1	2	0	12''	1	11	12	60''	0	9	11	30''	0

C = Click E = Equivoci T = Tempo F = Compito finito



Durante il test di usabilità si sono notate, grazie ai feedback dei tester, alcune possibilità di miglioramento e alcuni difetti del prototipo che hanno portato ad un cambiamento della user interface.

Problemi rilevati e fix:

- La freccia che prima era presente al posto del tasto "Gestisci personale" nella pagina M_Risorse_Umane non faceva capire che c'era la possibilità di fare altre cose in quella pagina.
- Label mancante "Aggiungi un nuovo dipendente" non faceva capire subito agli utenti che era la pagina giusta
- Per visualizzare un messaggio dell'admin erano necessari troppi passaggi ed essendo un'operazione presumibilmente frequente si è deciso di spostarla.

Note positive:

- Le icone hanno riscontrato una buona affordance da parte degli user tester.

CALCOLO USABILITÀ

1. Tasso di completamento:
(Numero di task completati / Numero ti task totali) * 100 = (21 / 24) * 100 = 87,5%
2. Usabilità aritmetica, ponderata:

C = Click	E = Equivoci	T = Tempo	F = Compito finito
C' = Click stimati	È = Equivoci tollerati	T' = Tempo stimato	exp = esperienza

$$f(C,\mathrm{E},\mathrm{T},\mathrm{F},\mathrm{C}',\mathrm{E}',\mathrm{T}') = \begin{cases} E \le E' \begin{cases} T < T' & \frac{F}{1 \times \left(\frac{C'}{C}\right) \times 1 \times \left(\frac{10}{\exp}\right)} \\ T \ge T' & \frac{F}{\left(\frac{T'}{T}\right) \times \left(\frac{C'}{C}\right) \times 1 \times \left(\frac{10}{\exp}\right)} \end{cases} \\ E > E' \begin{cases} T < T' & \frac{F}{1 \times \left(\frac{C'}{C}\right) \times \left(\frac{E'}{E}\right) \times \left(\frac{10}{\exp}\right)} \\ T \ge T' & \frac{F}{\left(\frac{T'}{T}\right) \times \left(\frac{C'}{C}\right) \times \left(\frac{E'}{E}\right) \times \left(\frac{10}{\exp}\right)} \end{cases} \end{cases}$$

In presenza di minori equivoci rispetto a quelli tollerati si va a verificare se il tempo impiegato per svolgere il compito risulta minore di quello prestabilito. In caso di verità, il valore di tempo ed equivoco saranno costanti (1) per mantenere bilanciato il resto dell'operazione in scala 1:1. Nel caso in cui gli equivoci sono superiori rispetto a quelli tollerati, si misura il tempo nello stesso modo. Tutte le frazioni (T'/T, C'/C, E'/E) servono per stabilire in percentuale la misura di usabilità del sistema, rapportata all'esperienza dell'utente (in base 10). Come numeratore è stato deciso di inserire la riuscita da parte dell'utente così che in caso di successo il risultato sia ben bilanciato e in caso di fallimento il risultato sia 0 a prescindere da qualsiasi altro dato.

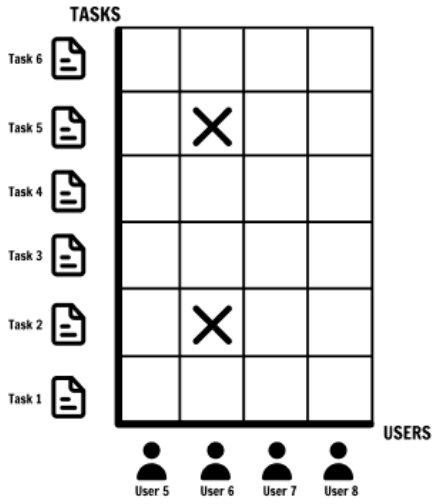
Usabilità aritmetica	Compito 1	Compito 2	Compito 3	Compito 4	Compito 5	Compito 6	
Utente 1	1,00	0,61	0,00	1,00	0,19	0,50	
Utente 2	0,71	1,00	0,30	0,04	0,67	1,00	
Utente 3	0,97	1,00	0,16	1,00	0,38	1,00	
Utente 4	0,76	0,71	1,00	1,00	0,00	0,00	**Medie**
Media aritmetica	**0,86**	**0,83**	**0,37**	**0,76**	**0,31**	**0,63**	**0,62**
Media ponderata	**0,69**	**0,75**	**0,18**	**0,53**	**0,22**	**0,44**	**0,47**

7.2 TEST DI USABILITA' (2) – MIGLIORAMENTO UI

Il secondo test di usabilità è stato eseguito su un gruppo di 4 persone che non erano presenti nel primo test.

	Compito 1				Compito 2				Compito 3				Compito 4				Compito 5				Compito 6			
Utente: Exp	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**
Utente 5: 6	5	0	18''	1	7	2	36''	1	2	0	5''	1	2	0	12''	1	7	3	43''	1	4	1	15''	1
Utente 6: 5	7	2	42''	1	10	2	30''	0	2	0	8''	1	4	3	29''	1	10	10	55''	0	2	0	5''	1
Utente 7: 8	5	0	32''	1	7	1	25''	1	2	0	5''	1	2	0	15''	1	4	0	10''	1	2	0	6''	1
Utente 8: 6	5	0	20''	1	8	3	40''	1	2	0	7''	1	2	0	15''	1	7	3	36''	1	2	0	8''	1

A fronte dei problemi riscontrati nello studio fatto nel paragrafo precedente, sono state apportate migliorie alla gui, come lo spostamento della pagina per la visualizzazione degli avvisi in modo da renderla più facilmente raggiungibile e l'aggiunta di etichette o modifiche delle icone per favorire l'ergonomia del sistema continuando a rispettare la legge della coerenza.



CALCOLO USABILITÀ

1. Tasso di completamento:
(Numero di task completati / Numero ti task totali) * 100 = (22 / 24) * 100 = 91,6%
2. Usabilità aritmetica, ponderata:

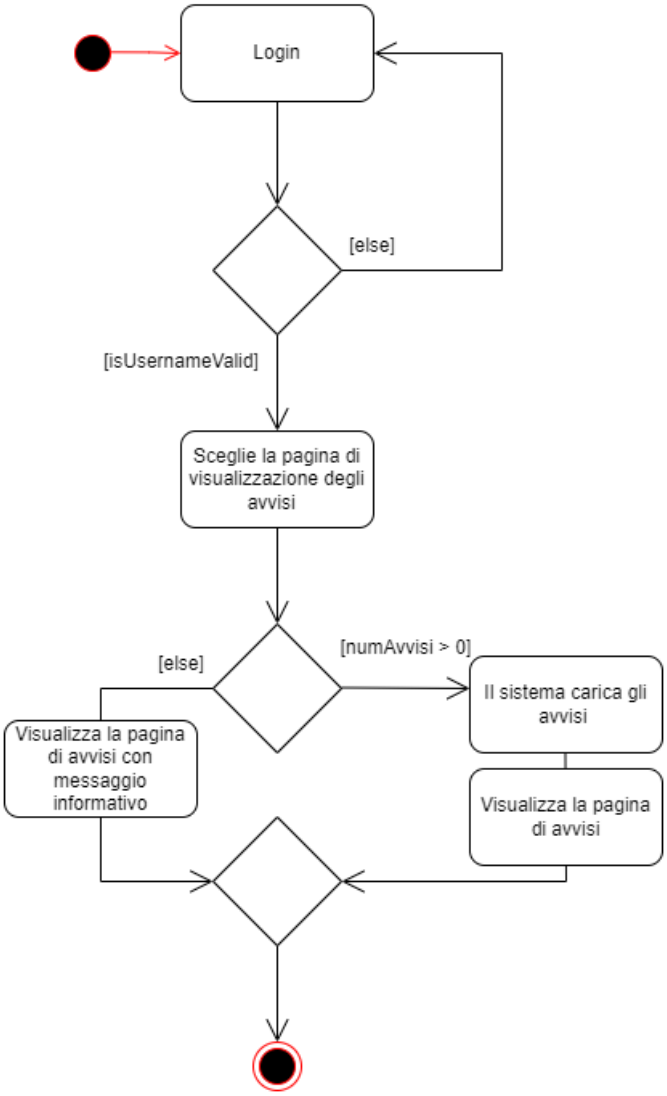
Usabilità aritmetica	**Compito 1**	**Compito 2**	**Compito 3**	**Compito 4**	**Compito 5**	**Compito 6**	
Utente 5	1,00	0,60	0,67	1,00	0,31	0,50	
Utente 6	0,53	0,00	0,67	0,11	0,00	1,00	
Utente 7	0,97	0,86	0,67	1,00	1,00	1,00	
Utente 8	1,00	0,31	0,67	1,00	0,37	1,00	**Medie**
Media aritmetica	**0,87**	**0,44**	**0,67**	**0,78**	**0,42**	**0,88**	**0,68**
Media ponderata	**0,70**	**0,40**	**0,53**	**0,62**	**0,34**	**0,70**	**0,55**

CONCLUSIONI

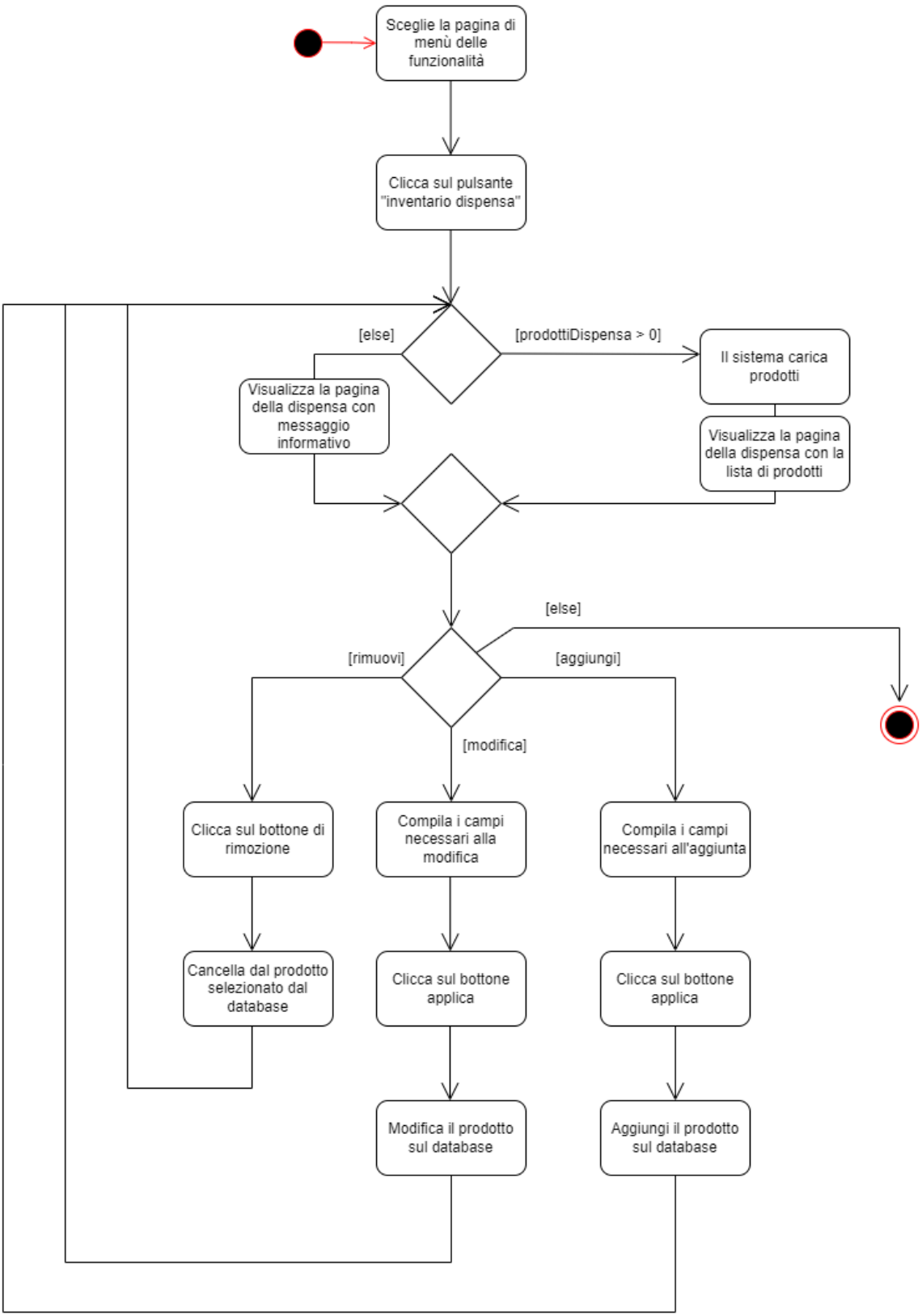
Valutando la crescita della media ponderata dell'usabilità (che era del 47%), nonostante possa sembrare a primo impatto bassa (55%) siamo comunque soddisfatti dei risultati ottenuti. Ciò è dovuto dal fatto che la funzione da noi definita prende in considerazione molti parametri, che rendono difficile raggiungere una soglia più alta in situazioni medie (e.g. Utente con 6 punti esperienza che compie un compito di peso 7…).
Si potrebbe pensare di aver sbagliato la scelta dei compiti da assegnare ma in realtà più o meno tutti i compiti avrebbero avuto una valutazione di peso simile. La scelta degli utenti avrebbe potuto invece essere migliore ma purtroppo non si è trovata disponibilità di persone con specifiche che cercavamo.
Pertanto, riteniamo che la soglia del 50% della media ponderata dell'usabilità sia sufficiente per procedere con la progettazione del sistema.

8 Activity Diagram

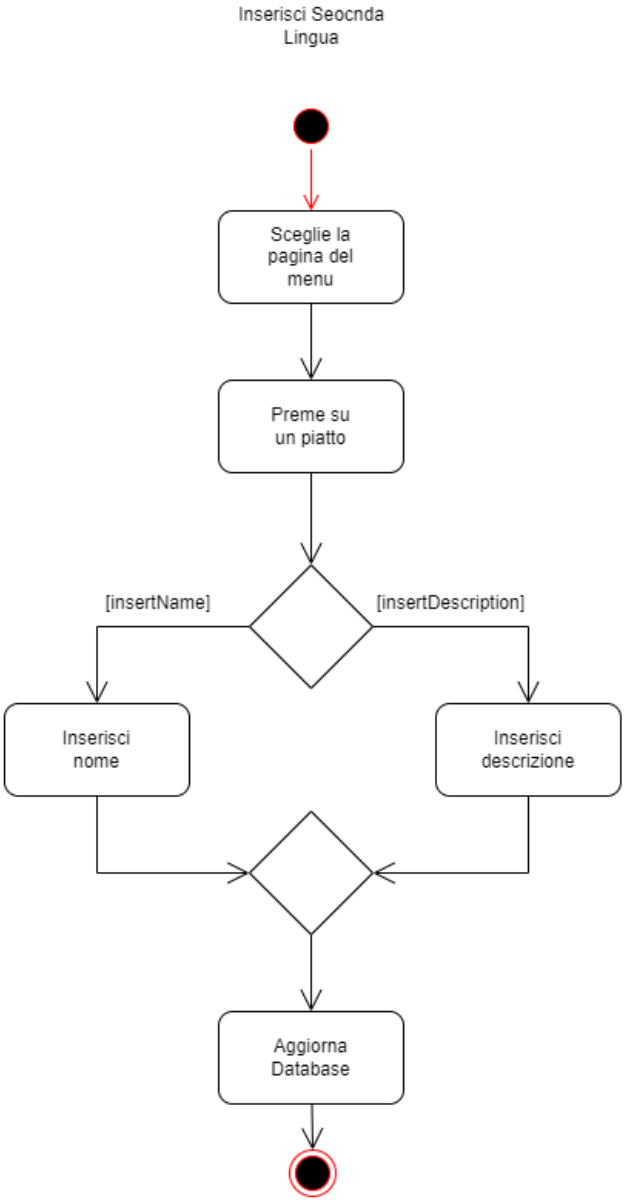
VISUALIZZA AVVISI



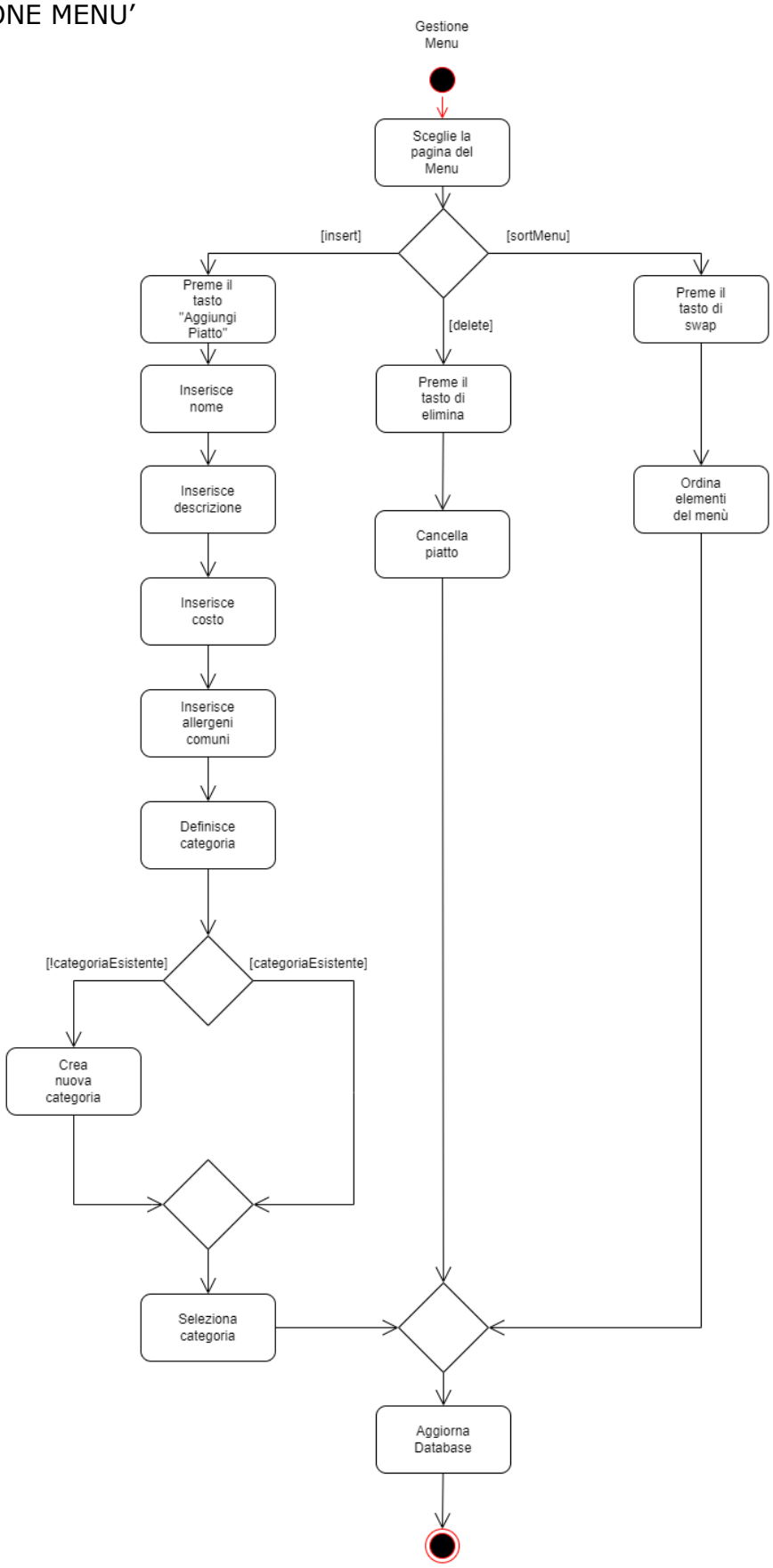
GESTIONE DISPENSA



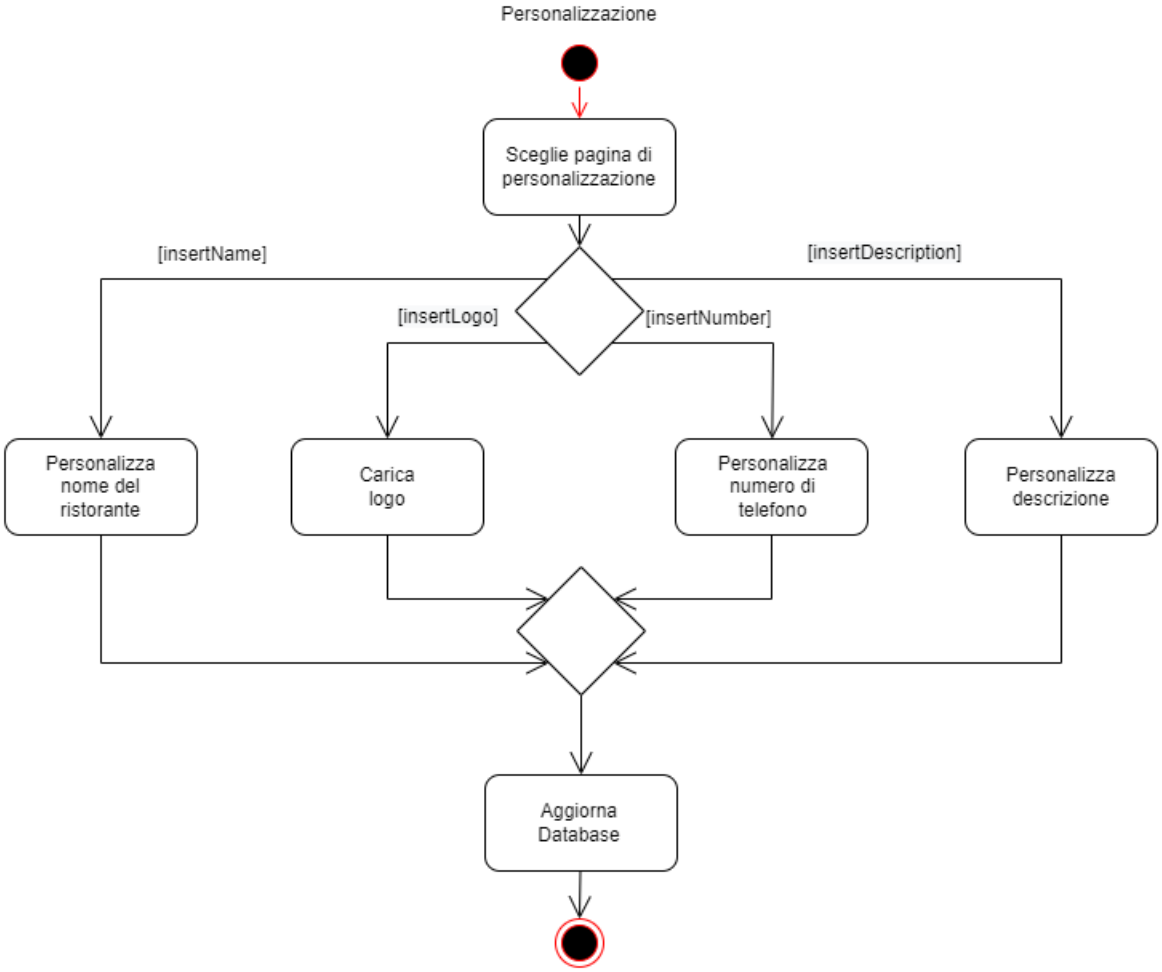
GESTIONE SECONDA LINGUA



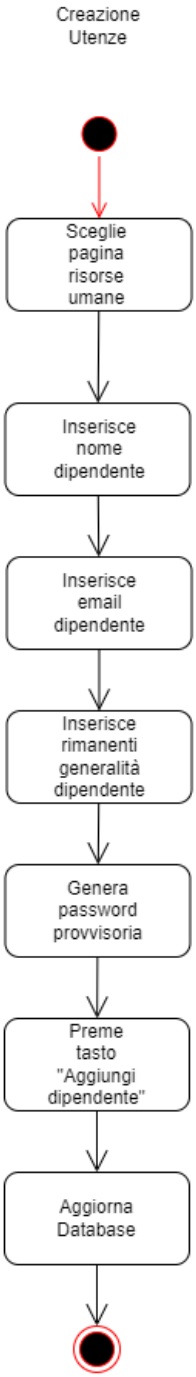
GESTIONE MENU’



PERSONALIZZAZIONE ATTIVITÁ



CREAZIONE UTENZE



9 DOCUMENTO DI DESIGN DEL SISTEMA (PROGETTAZIONE)

9.1 System Design

9.1.1 Analisi dell'architettura con esplicita definizione dei criteri di design, descrizione motivazione delle scelte tecnologiche adottate

CRITERI DI DESIGN
Sono stati scelti dei compromessi tra i seguenti 5 fondamentali criteri di design:
Performance: essendo un software gestionale di un ristorante non è prettamente improntato ad un'ottima performance poiché non è tra i requisiti fondamentali.
Affidabilità: il sistema deve risultare solido e sicuro per qualsiasi richiesta e operazione utilizzando framework che ne facilitano la gestione mediante l'utilizzo di token e diversi controlli sia front-end che back-end.
Costi: è un progetto universitario ma supponendo non lo fosse avremo mantenuto dei costi non troppo elevati.
Mantenimento: investimento di un tempo maggiore di progettazione e sviluppo per avere una maggiore estendibilità e scalabilità. E' necessario però trovare un trade-off che equilibri la qualità del software e il tempo a disposizione per realizzarlo.
Usabilità: lo studio di usabilità (precedentemente eseguito) garantisce all'utente una buona esperienza.
Nel complesso quindi si sacrifica del tempo iniziale, denaro e performance per favorire maggiore affidabilità, mantenimento e una buona usabilità.

ARCHITETTURA UTILIZZATA

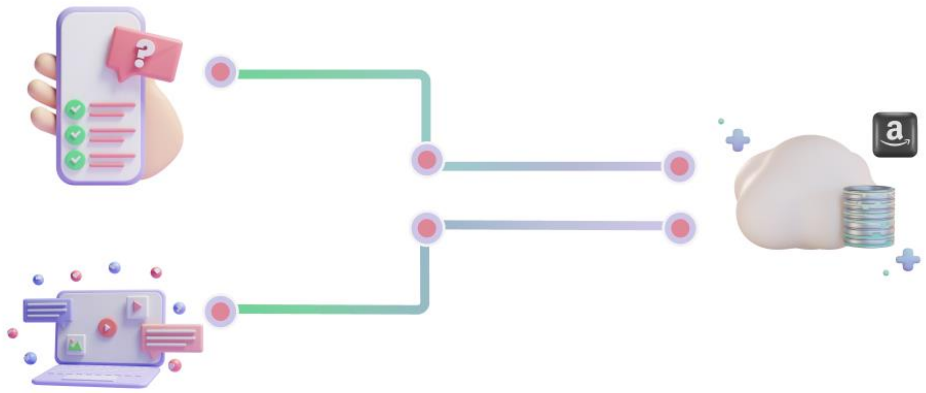
Architettura Three-Tier

Il tipo di architettura scelta risulta quella a tre livelli:

Livello di presentazione: Frontend Android e desktop
Livello applicazione (business logic): Backend e Controller dei Frontend
Livello dati: Database

AWS EC2

Per la realizzazione del software si è pensato, dovendo essere distribuito, di utilizzare il servizio cloud AWS di Amazon con un'istanza server virtuale gratuita Amazon Elastic Compute Cloud (EC2) sul quale offrire servizio di hosting per le rest API.
Il motivo per il quale utilizzare il servizio di cloud AWS è che garantisce un'alta disponibilità del servizio, riducendo il rischio di interruzioni, riduce i costi per l'hosting, la manutenzione e la gestione dell'infrastruttura, consentendo di concentrarsi sulle funzionalità dell'applicazione.
EC2 è uno **IaaS** (**Infrastructure as a Service**) ovvero non offre alcun servizio se non l'infrastruttura, tutto il resto è gestito da noi.



Il back-end svolge un ruolo importante nella gestione dei dati di un'applicazione. Quando un client invia una richiesta, il back-end la riceve e la elabora, quindi la propaga al database. Questo processo è fondamentale per garantire la sicurezza dei dati, poiché il back-end impedisce l'accesso diretto al database, fornendo una barriera protettiva.

L'architettura del back-end consente di mantenere la flessibilità dell'applicazione, poiché permette di modificare il modulo di memorizzazione dei dati senza dover rivedere l'intera struttura. Ad esempio, se si decide di passare a un modulo di memorizzazione dei dati più recente o più avanzato, è possibile sostituire il vecchio modulo con il nuovo senza dover riedificare l'intera struttura.

In questo modo, il back-end offre un modo sicuro e flessibile per archiviare e accedere ai dati, garantendo che l'applicazione funzioni in modo ottimale. Inoltre, fornisce un sistema di gestione dei dati affidabile e robusto che supporta le esigenze dell'applicazione e garantisce che i dati siano sempre accessibili e sicuri.

In sintesi, il back-end è un componente essenziale per la gestione dei dati in un'applicazione, che fornisce sicurezza, flessibilità e affidabilità nella gestione dei dati.

DECOMPOSIZIONE IN SOTTOSISTEMI

1. Server REST API:

Utilizzare una REST API consente una facile integrazione con altre applicazioni e servizi, rendendo possibile l'utilizzo di dati e funzionalità da altre fonti.
Il server verrà realizzato con il supporto del framework Spring Boot.
Per l'autenticazione e l'autorizzazione abbiamo valutato la possibilità di utilizzare framework come Spring Security oppure moduli di AWS adibiti a ciò come API Gateway ma abbiamo optato per la realizzazione di un token personalizzato.
Si utilizzano design pattern:

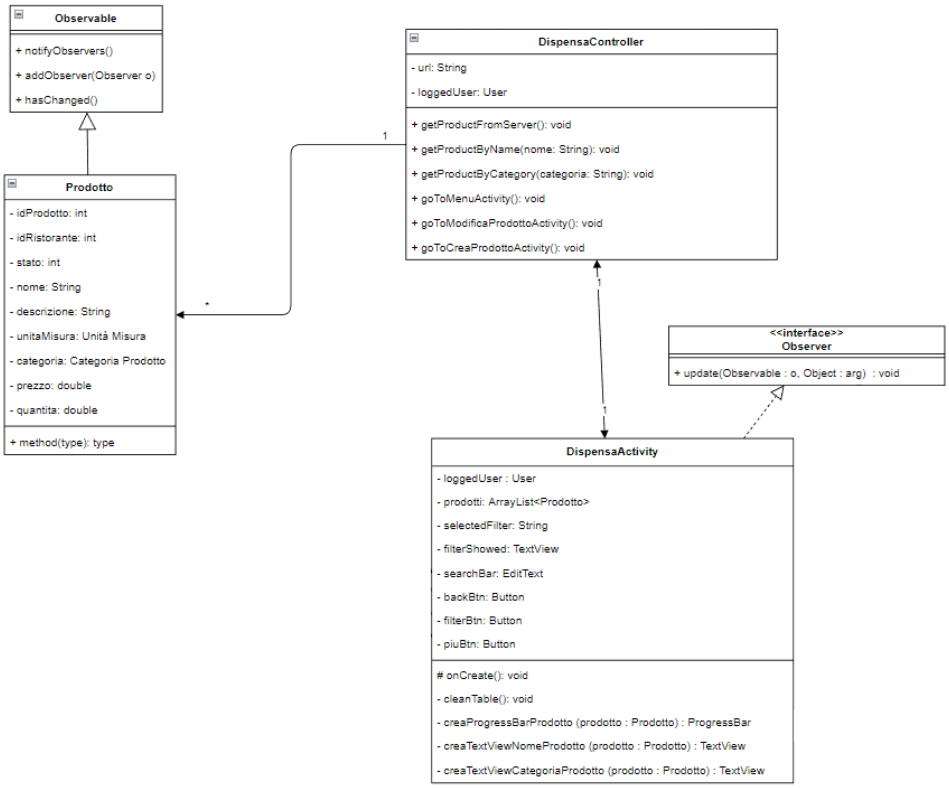
- DAO per garantire:
    - la separazione della logica di accesso ai dati: separa la logica di accesso ai dati dalla logica delle applicazioni, rendendo il codice più leggibile e manutenibile.
    - Riduzione della duplicazione di codice: rende più facile la riutilizzabilità del codice, riducendo la duplicazione di codice.
    - Migliore organizzazione del codice: aiuta a mantenere un'organizzazione ordinata del codice, rendendo più semplice la manutenzione e l'espansione del progetto.
- Singleton:
    - Riduzione della complessità: rende più semplice la gestione di risorse condivise del database, riducendo la complessità del codice.
    - Maggiore flessibilità: permette di modificare facilmente le proprietà e il comportamento della classe senza influire sulle altre parti dell'applicazione.
    - Un'unica istanza: Garantisce che ci sia una sola istanza della classe per l'intera applicazione, evitando la creazione di più istanze inutili.

ENDPOINTS:
Abbiamo impiegato le operazioni CRUD per soddisfare le nostre richieste API. CRUD è un acronimo che sta per Create (creare), Read (leggere), Update (aggiornare) e Delete (eliminare). Grazie alle operazioni CRUD, abbiamo potuto offrire una maggiore flessibilità e funzionalità nel nostro sistema di API, consentendo ai front-end di interagire con le nostre risorse in modi più efficaci e intuitivi.

@GetMapping("/user")	mostra un elenco di tutti gli utenti iscritti
@PutMapping("/user/account")	modifica account di un utente Android
@PutMapping("/user/accountDesktop")	modifica account di un utente desktop
@PutMapping("/user/upgradeRole")	promuove un dipendente
@PutMapping("/user/downgradeRole")	declassa un dipendente
@PostMapping("/user/fire")	licenzia un dipendente
@PostMapping("/signup-admin")	crea un nuovo admin e un nuovo ristorante associato
@PostMapping("/signup/newEmployee")	crea/assumi un nuovo dipendete
@PostMapping("/verify")	verifica se la mail inserita è di un dipendente
@PutMapping("/user/first-access")	controlla se è il primo accesso di un dipendente per eventualmente aggiornare la password
@GetMapping("/user/{id}")	cerca un dipendente in base al suo identificativo
@PostMapping("/login")	effettua il login
@GetMapping("/avvisi")	mostra tutti gli avvisi
@GetMapping("/avvisi-hidden/{id_user}")	mostra gli avvisi nascosti da un utente da specificare
@GetMapping("/avvisi-viewed/{id_user}")	mostra gli avvisi letti da un utente da specificare
@PostMapping("/avviso/segna-come-letto/{id_avviso}")	segnare un avviso come letto
@PostMapping("/avviso/segna-come-non-letto/{id_avviso}")	segnare un avviso come non letto
@PostMapping("/avviso/segna-come-nascosto/{id_avviso}")	segnare un avviso come nascosto
@PostMapping("/avviso/segna-come-non-nascosto/{id_avviso}")	segnare un avviso come non nascosto
@GetMapping("/avviso/numero-di-avvisi-da-leggere")	mostrare quanti avvisi ci sono da leggere (per il badge sull'icona)
@PostMapping("/avviso/crea")	scrivere un nuovo avviso
@PutMapping("/avviso/cancella/{id_avviso}")	eliminare un avviso
@GetMapping("/menu")	mostrare tutti gli elementi dei menu
@GetMapping("/menu/categoria")	mostrare tutte le categorie dei menu
@GetMapping("/menu/categoria/piatti")	mostrare tutti i piatti di un menu
@PutMapping("/menu/piatto-update-posizione")	aggiorna la posizione di un piatto nel menu
@DeleteMapping("/menu/delete-ordine-menu-precedente/{id_ristorante}")	sovrascrive l'elenco della posizione dei piatti
@PutMapping("/categoria-update-posizione")	aggiorna l'ordine delle categorie nel menu
@GetMapping("/menu/ristorante/piatto")	cerca un piatto in un ristorante
@GetMapping("/menu/ristorante/piatto/{id_piatto}")	cerca un piatto con il suo identificativo
@GetMapping("/menu/piatto")	cerca un piatto nei ristoranti
@PostMapping("/menu/newCategoria")	crea una nuova categoria nel menu
@PostMapping("/menu/newPlate")	crea un nuovo piatto

@PutMapping("/menu/updatePlate/{id_piatto}")	modifica le informazioni di un piatto
@DeleteMapping("/menu/deletePlate/{id_piatto}")	elimina un piatto
@PostMapping("/menu/piatto/secondaLingua")	aggiunge una seconda lingua per le informazioni di un piatto
@GetMapping("/dispensa")	mostra i prodotti nelle dispense dei ristoranti
@GetMapping("/dispensa/categoria")	mostra le categorie dei prodotti nelle dispense di un ristorante
@GetMapping("/dispensa/prodotto")	mostra un prodotto nella dispensa di un ristorante
@GetMapping("/dispensa/prodotto/disponibili")	mostra i prodotti nella dispensa di un ristorante che hanno una quantità maggiore di zero
@PostMapping("/dispensa/newProduct")	crea un nuovo prodotto in dispensa
@PutMapping("/dispensa/modifyProduct")	modifica un prodotto in dispensa
@PostMapping("/dispensa/deleteProduct")	elimina un prodotto dalla dispensa
@GetMapping("/error")	mostra un errore
@PostMapping("/logout")	effettua il logout in Android
@GetMapping("/business/nomeAttivita")	mostra il nome di un'attività dato il suo identificativo
@PutMapping("/business")	modifica le informazioni di un'attività
@PostMapping("/business/image")	modifica il logo del proprietario di un ristorante (png)
@GetMapping("/business/getImage")	ritorna una stringa codificata (base64) del logo del proprietario di un ristorante

2. Client Desktop:
Per lo sviluppo dell'interfaccia grafica del client desktop, verrà utilizzato JavaFX. Questo framework fornisce una vasta gamma di strumenti e risorse per creare interfacce utente attraenti e interattive. In caso di modifiche alle classi modello, sarà necessario ricorrere all'utilizzo del design pattern Model-View-Controller, che permette di aggiornare l'interfaccia grafica in tempo reale.



[Estratto dal class diagram di analisi]

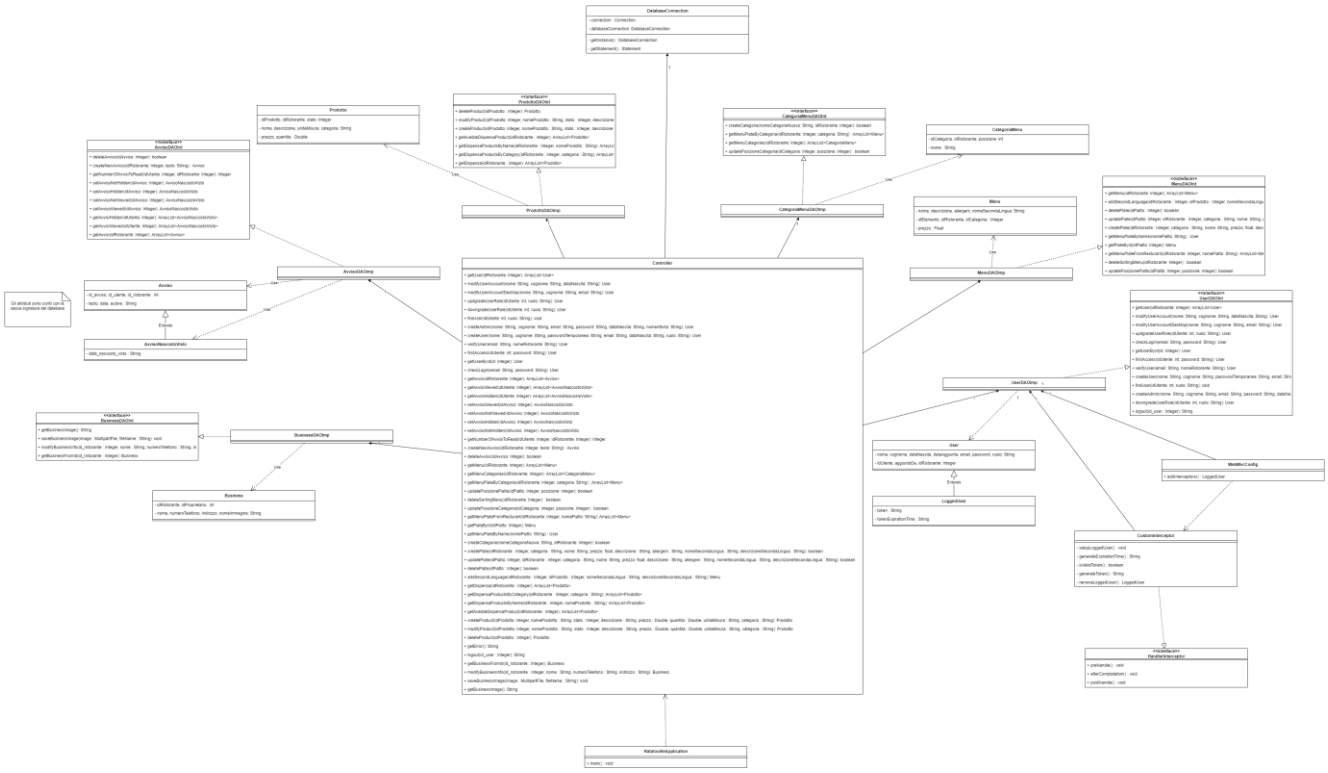
Per garantire una gestione efficiente delle richieste, sarà utile suddividere le stesse in macrocategorie e utilizzare più controller per gestirle. Le informazioni che verranno mostrate all'utente saranno ottenute mediante chiamate al server rest API descritto in precedenza. Questo garantirà che l'interfaccia sia sempre sincronizzata con i dati presenti sul server e che l'utente possa accedere alle informazioni di cui ha bisogno in modo rapido e intuitivo.

3. Client Android
L'applicazione Android deve seguire una struttura simile al client desktop, utilizzando il modello MVC e chiamate alle API REST del server. Questo garantirà una gestione efficiente e organizzata del codice, tuttavia, la parte grafica utilizzerà XML. Ogni schermata dell'app sarà rappresentata da un'attività, a eccezione della schermata iniziale e dei relativi collegamenti. Questi elementi, come la barra di navigazione e la parte superiore della pagina, saranno fissi e presenti in una sola schermata riutilizzabile tramite fragment interscambiabili. Questo permetterà una maggiore flessibilità nella navigazione all'interno dell'app e una migliore esperienza utente. In sintesi, l'obiettivo è quello di mantenere una struttura simile al client desktop, ma con una parte grafica personalizzata per la piattaforma Android.

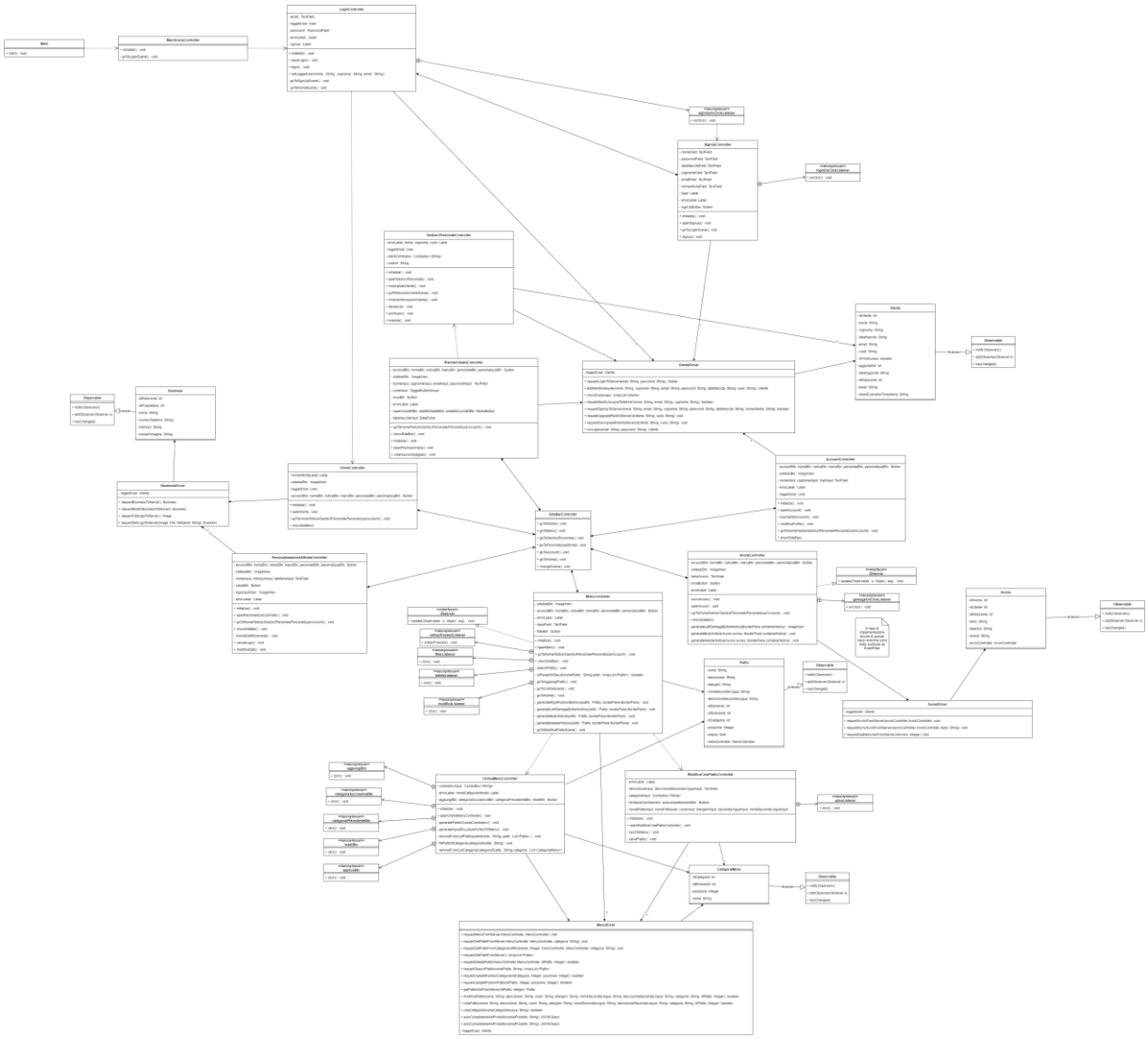
Entrambi i client, sia quello Android che quello desktop, sono progettati per essere responsive. Ciò significa che si adattano automaticamente alla risoluzione dello schermo e alle dimensioni dello stesso, offrendo una visualizzazione ottimale su qualsiasi dispositivo.

9.1.2 Class Diagram delle classi di Design

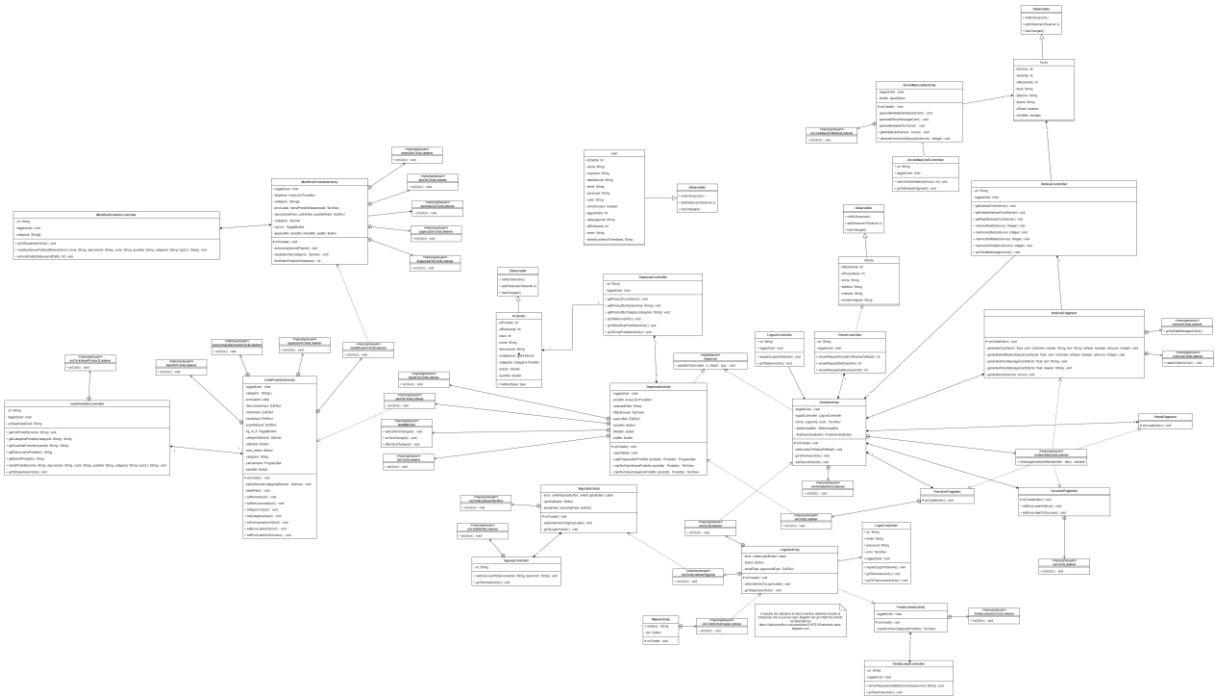
9.1.2.1 Class diagram backend



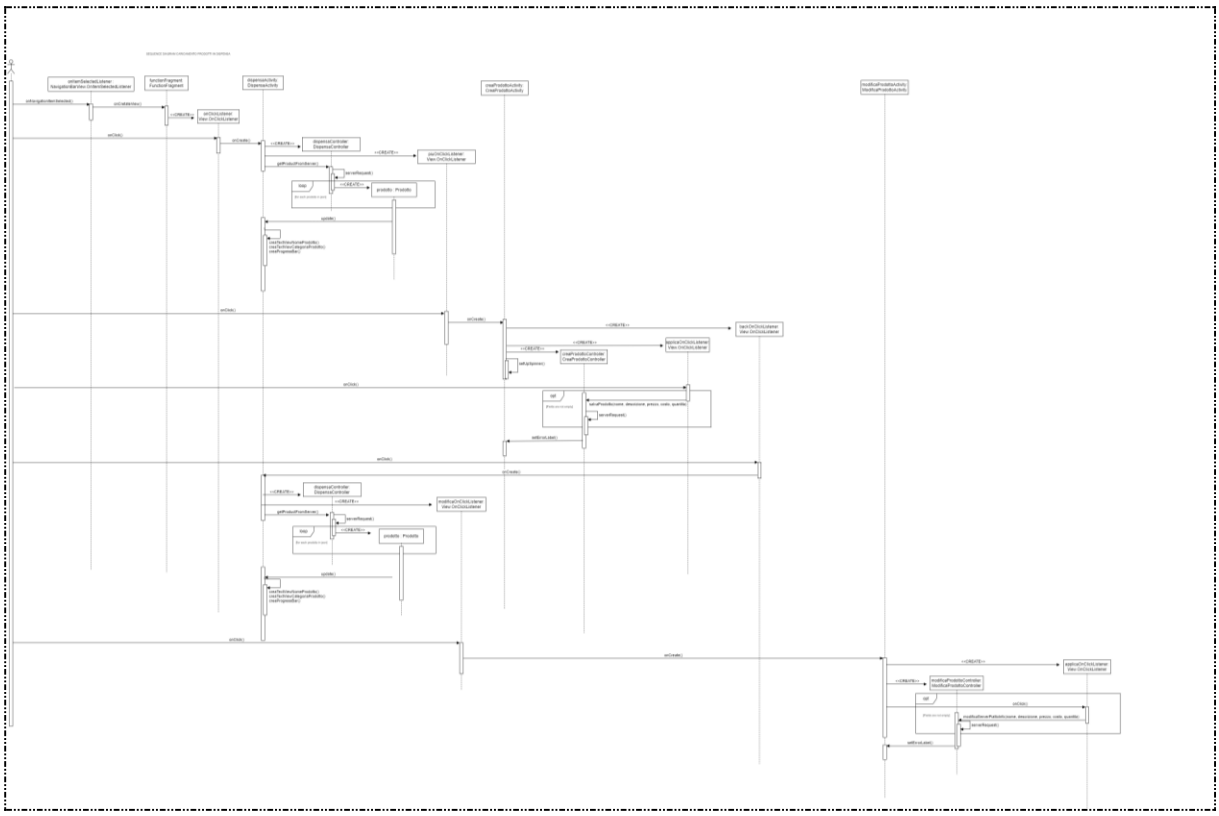
9.1.2.2 Class diagram desktop

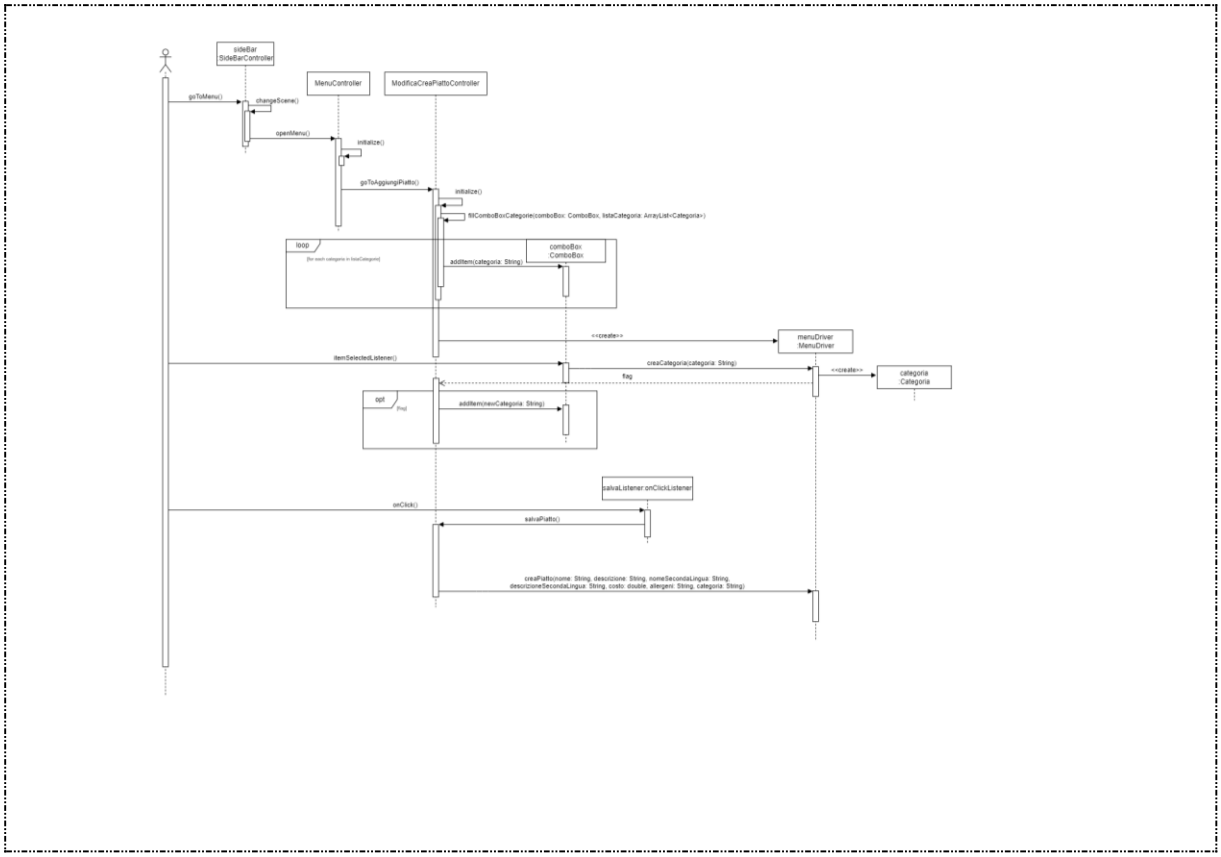


9.1.2.3 Class Diagram Android



9.1.3 Sequence Diagram di Design (2 casi d'uso descritti dai template di Cockburn)





TESTING

9.2 INDIVIDUAZIONE CLASSI DI EQUIVALENZA (1)

```java
public CategoriaMenu removeFromListCategoria(String categoriaScelta, List<CategoriaMenu> categorie) {
    if(categorie != null) {
        for (Iterator iterator = categorie.iterator(); iterator.hasNext();) {
            CategoriaMenu categoria = (CategoriaMenu) iterator.next();
            if(categoria != null) {
                if(categoria.getNome().equals(categoriaScelta)) {
                    queueCategoria.add(indexOfCategoria, categoria);
                    iterator.remove();
                    return categoria;
                }
            }
        }
    }
    return null;
}
```

CLASSI DI EQUIVALENZA categoriaScelta:
CE1: [Parola presente nella lista categoria] ---------- VALIDO
CE2: [null] ----------NON VALIDA
CE3: [Parola non presente nella lista categoria] ---------- NON VALIDA

CLASSI DI EQUIVALENZA categorie:
CE1: [lista piena senza duplicati] ---------- VALIDO
CE2: [null] ---------- NON VALIDA
CE3: [lista con elementi duplicati] ---------- NON VALIDA
CE4: [lista con elementi null] ---------- NON VALIDA
CE5: [lista vuota] ---------- NON VALIDA

9.3 STRATEGIA ADOTTATA PER LA PROGETTAZIONE (1)

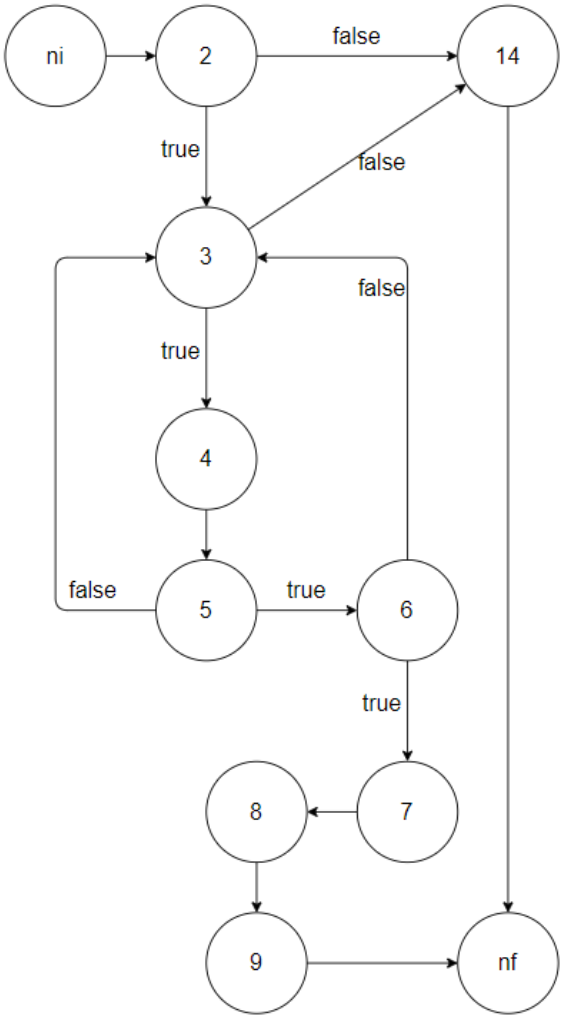
TESTING BLACK BOX

STRATEGIA UTILIZZATA: WECT

Definita una lista1 = {“Primi”, “Secondi”, “Contorni”, “Bevande”}
Definita una lista2 = null
Definita una lista3 = {“Primi”, “Secondi”, “Primi”, “Bevande”}
Definita una lista4 = {“Primi”, null, “Contorni”, “Bevande”}
Definita una lista5 = {}

TC1: (“Primi”, lista1)	⇔	obj.nome () ➔ “Primi”
TC2: (null, lista2)	⇔	null
TC3: (“Primi”, lista3)	⇔	obj.nome() ➔ “Primi”
TC4: (“Dessert”, lista4)	⇔	null
TC5: (null, lista1)	⇔	null
TC6: (“Dessert”, lista1)	⇔	null
TC7: (“Dessert”, lista5)	⇔	null

TESTING WHITE BOXING



NODE COVERAGE

Indice	Test Case	Oracolo	Path
1	(“Primi”, null)	null	2 – 14
2	(“Primi”, lista1)	obj.nome () ➔ “Primi”	2 – 3 – 4 – 5 – 6 – 7 – 8 – 9

Test Effectiveness Ratio = # 100 / # 100

BRANCH COVERAGE

Indice	Test Case	Oracolo	Path
1	(“Primi”, null)	null	2 – 14
2	(“Primi”, lista5)	null	2 – 3 – 14
3	(“Primi”, lista1)	obj.nome () ➔ “Primi”	2 – 3 – 4 – 5 – 6 – 7 – 8 – 9
4	(“Secondi”, lista1)	obj.nome () ➔ “Secondi”	2 – 3 – 4 – 5 – 6 – 3 – 4 – 5 – 6 – 7 – 8 – 9
5	(“Primi”, [null, “Primi”])	obj.nome () ➔ “Primi”	2 – 3 – 4 – 5 – 3 – 4 – 5 – 6 – 7 – 8 – 9
6	(“Antipasto”, list1)	null	2 – 3 – 4 – 5 – 6 – 3 – 14
7	(“Primi”, [null, null])	null	2 – 3 – 4 – 5 – 3 – 14
8	(“Secondi”, [“Primi”, null, “Secondi”])	obj.nome () ➔ “Secondi”	2 – 3 – 4 – 5 – 6 – 3 – 4 – 5 – 3 – 4 – 5 – 6 – 7 – 8 – 9

MODIFIED CONDITION COVERAGE

Non è necessaria copertura di tutte le possibili combinazioni di valori delle condizioni all'interno delle strutture di controllo poiché non si presentano in molteplicità.

9.4 INDIVIDUAZIONE CLASSI DI EQUIVALENZA (2)

```java
private Avviso createAvviso(Integer id_ristorante, String testo, Integer idUtente) throws SQLException {
    boolean isSupervisoreOrAdmin = false;
    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    LocalDateTime now = LocalDateTime.now();
    ResultSet rs;
    String query, autore = "";
    query = "SELECT ruolo, nome FROM utente WHERE id_utente = " + idUtente + " AND id_ristorante = " + id_ristorante;
    rs = db.getStatement().executeQuery(query);
    while(rs.next()) {
        String ruolo = rs.getString("ruolo");
        if(ruolo.equals("admin") || ruolo.equals("supervisore")) {
            isSupervisoreOrAdmin = true;
            autore = rs.getString("nome");
        }

    }

    if(isSupervisoreOrAdmin) {
        if(testo != null) {
            query = "INSERT INTO avviso (id_avviso, id_utente, id_ristorante, testo, data_ora) VALUES (default, " + idUtente + ", " +
                                                                                    id_ristorante + ", '" + testo + "', '" + now + "') ";
            db.getStatement().executeUpdate(query);

            query = "SELECT id_avviso FROM avviso WHERE id_utente = " + idUtente + " AND id_ristorante = " + id_ristorante + " AND testo
                                                                                        '" + testo + "' AND  data_ora = '" + now + "'";
            rs = db.getStatement().executeQuery(query);
            while(rs.next()) {
                return new Avviso(rs.getInt("id_avviso"), idUtente, id_ristorante, testo, now.toString(), autore);
            }
        }
    }
    return null;
}

```

Trattandosi di un metodo che interagisce con il database è necessario creare degli utenti di testing :

idUtente	Nome	E-mail	Ruolo	idRistorante
2	“Teka”	“teka.freitas@example.com”	“admin”	9
7	“Ece”	“ece.hakyemez@example.com”	“supervisore”	2
10	“Cornelio”	“cornelio.fuentes@example.com”	“addetto_sala”	7

Id_ristorante
CE1: [idRistorante a cui appartiene idUtente] ---------- VALIDA
CE2: [idRistoranti a cui non appartiene idUtente] ---------- VALIDA
CE3: [MININT, -1] ---------- NON VALIDA
Testo
CE1: [String] ---------- VALIDA
CE2: [null] ---------- NON VALIDA
IdUtente
CE1: [idUtente presente nel db] ---------- VALIDA
CE2: [idUtente non presente nel db] ---------- VALIDA
CE3: [MININT, -1] ---------- NON VALIDA

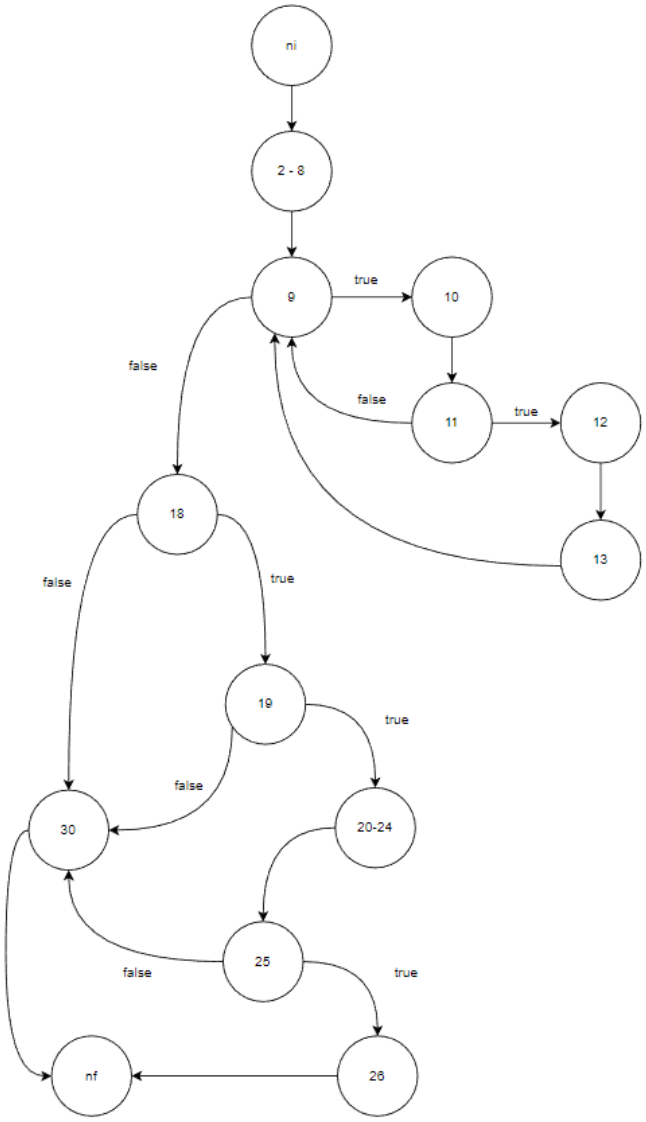
9.5 STRATEGIA ADOTTATA PER LA PROGETTAZIONE (2)

BLACK BOX

SECT - 3 * 2 * 3 = 18

TC1: (9, "testo", 2) ⇔Avviso (MAX (id_avviso), 2, 9, "testo", orario di aggiunta, "Teka")	(CE1, CE1, CE1)
TC2: (9, "testo", idUtente non presente nel db) ⇔ null	(CE1, CE1, CE2)
TC3: (9, "testo", -10) ⇔ null	(CE1, CE1, CE3)
TC4: (9, null, 2) ⇔ null	(CE1, CE2, CE1)
TC5: (9, null, idUtente non presente nel db) ⇔ null	(CE1, CE2, CE2)
TC6: (9, null, -10) ⇔ null	(CE1, CE2, CE3)
TC7: (10, "testo", 2) ⇔ null	(CE2, CE1, CE1)
TC8: (10, "testo", idUtente non presente nel db) ⇔ null	(CE2, CE1, CE2)
TC9: (10, "testo", -10) ⇔ null	(CE2, CE1, CE3)
TC10: (10, null, 2) ⇔ null	(CE2, CE2, CE1)
TC11: (10, null, idUtente non presente nel db) ⇔ null	(CE2, CE2, CE2)
TC12: (10, null, -10) ⇔ null	(CE2, CE2, CE3)
TC13: (-10, "testo", 2) ⇔ null	(CE3, CE1, CE1)
TC14: (-10, "testo", idUtente non presente nel db) ⇔ null	(CE3, CE1, CE2)
TC15: (-10, "testo", -10) ⇔ null	(CE3, CE1, CE3)
TC16: (-10, null, 2) ⇔ null	(CE3, CE2, CE1)
TC17: (-10, null, idUtente non presente nel db) ⇔ null	(CE3, CE2, CE2)
TC18: (-10, null, -10) ⇔ null	(CE3, CE2, CE3)

TESTING WHITE BOX



NODE COVERAGE

Indice	Test Case	Oracolo	Path
1	(9, “testo”, 2)	Avviso (MAX (id_avviso), 2, 9, “testo”, orario di aggiunta, “Teka”)	2/8 – 9 – 10 – 11 – 12 – 13 – 9 – 18 – 19 – 20/24 – 24 – 25 – 26
2	(idUtente non esistente, null, 2)	null	2/8 – 9 – 18 – 30

BRANCH COVERAGE

Possibili branch: 13
A causa dei seguenti motivi non è stato possibile ottenere il 100% del branch covering:

- Se si percorre il nodo 12 allora si deve percorrere il nodo 19 sennò la branch non è testabile e viceversa.
- Non è possibile che il result set restituisca più di un risultato quindi i percorsi che ritornano al nodo 9 due volte non sono percorribili.
- L'arco 25 – 30 è impossibile da verificare poiché il result set restituirà sempre l'avviso appena inserito.

Branch non testabili:
2/8 - 9 - 10 - 11 - 12 - 13 - 9 - 18 - 30
2/8 - 9 - 19 - 30
2/8 - 9 - 19 - 20/24 - 25 - 30
2/8 - 9 - 10 - 11 - 9 - 18 - 19 - 20/24 - 25 - 30
2/8 - 9 - 19 - 20/24 - 25 - 26
2/8 - 9 - 10 - 11 - 9 - 18 - 19 - 20/24 - 25 - 26
2/8 - 9 - 10 - 11 - 9 - 10 - 11 - 12 - 13 - 9 - 18 - 19 - 30
2/8 - 9 - 10 - 11 - 9 - 10 - 11 - 12 - 13 - 9 - 18 - 19 - 20/24 - 25 - 30
2/8 - 9 - 10 - 11 - 9 - 10 - 11 - 12 - 13 - 9 - 18 - 19 - 20/24 - 25 – 26

Indice	Test Case	Oracolo	Path
1	(2, null, 9)	null	2/8 – 9 – 10 – 11 – 12 – 13 – 9 – 18 – 19 – 30
2	(9, "testo", 2)	Avviso (MAX (id_avviso), 2, 9, "testo", orario di aggiunta, "Teka")	2/8 – 9 – 10 – 11 – 12 – 13 – 9 – 18 – 19 – 20/24 – 25 – 26
3	(idUtente non esistente, null, 2)	null	2/8 – 9 – 18 – 30
4	(7, "testo", 10)	null	8 - 9 - 10 - 11 - 9 - 18 – 30

Branch coverage: 4 / 13 = 31%

CONDITION MODIFIED

TC1.1: (2, null, 7) ⇔ null
TC1.2: (9, null, 2) ⇔ null

TC2.1: (2, "testo", 7) ⇔ Avviso (MAX(id_avviso), 7, 2, "testo", orario di aggiunta, "Ece")

TC2.2: (9, "testo", 2) ⇔ Avviso(MAX(id_avviso), 2, 9, "testo", orario di aggiunta, "Teka")

STUDIO USABILITÀ SUL CAMPO

Per condurre uno studio di usabilità dell'app e del software desktop a prodotto finito, abbiamo coinvolto alcuni dei nostri colleghi come tester. Abbiamo utilizzato sia le tecniche dello studio di usabilità a priori che gli strumenti di logging (Firebase) integrati nell'app Android per raccogliere dati sul comportamento degli utenti. Dalle analisi sono emersi i seguenti risultati:

9.6 TEST DI USABILITA' (1)

Tabella dei compiti

Compito	Click Stimati (dalle rispettive HOME)	Tempo Stimato	Peso Compito (1 – 10)	Equivoci tollerato
Aumenta di un chilo la quantità di patate nella dispensa	5	31''	8	2
Crea un piatto nel menu in una categoria ancora non esistente	7	35''	9	2
Visualizza un messaggio dell'amministratore	1	18''	5	1
Scrivi un messaggio in qualità di amministratore	3	20''	7	1
Promuovi un dipendente	5	35''	7	2
Aggiungi un nuovo dipendente alle risorse umane	3	40''	7	2

	Compito 1				Compito 2				Compito 3				Compito 4				Compito 5				Compito 6			
Utente: Exp	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**	**C**	**E**	**T**	**F**
Utente 1: 3	5	0	28''	1	7	0	32''	1	1	0	10''	1	5	1	22''	1	5	0	26''	1	3	0	30''	1
Utente 2: 7	5	0	19''	1	7	0	25''	1	1	0	5''	1	3	0	10''	1	5	0	27''	1	3	0	25''	1
Utente 3: 7	5	0	18''	1	7	0	35''	1	1	0	8''	1	3	0	10''	1	5	0	29''	1	3	0	31''	1
Utente 4: 5	5	0	20''	1	7	0	30''	1	1	0	4''	1	3	0	11''	1	7	1	37''	1	3	0	35''	1

C = Click E = Equivoci T = Tempo F = Compito finito

CALCOLO USABILITÀ

1. Tasso di completamento:
(Numero di task completati / Numero ti task totali) * 100 = (24 / 24) * 100 = 100%
2. Usabilità aritmetica, ponderata:

C = Click	E = Equivoci	T = Tempo	F = Compito finito
C' = Click stimati	E' = Equivoci tollerati	T' = Tempo stimato	exp = esperienza

$$f(C, \mathrm{E}, \mathrm{T}, \mathrm{F}, \mathrm{C}', \mathrm{E}', \mathrm{T}') = \begin{cases} E \leq E' \begin{cases} T < T' & \frac{F}{1 \times \left(\frac{C'}{C}\right) \times 1 \times \left(\frac{10}{\exp}\right)} \\ T \geq T' & \frac{F}{\left(\frac{T'}{T}\right) \times \left(\frac{C'}{C}\right) \times 1 \times \left(\frac{10}{\exp}\right)} \end{cases} \\ E > E' \begin{cases} T < T' & \frac{F}{1 \times \left(\frac{C'}{C}\right) \times \left(\frac{E'}{E}\right) \times \left(\frac{10}{\exp}\right)} \\ T \geq T' & \frac{F}{\left(\frac{T'}{T}\right) \times \left(\frac{C'}{C}\right) \times \left(\frac{E'}{E}\right) \times \left(\frac{10}{\exp}\right)} \end{cases} \end{cases}$$

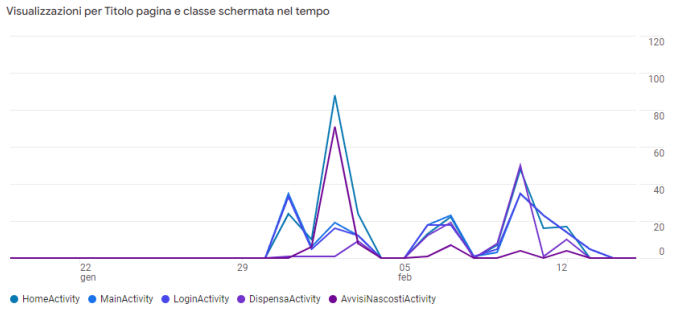
In presenza di minori equivoci rispetto a quelli tollerati si va a verificare se il tempo impiegato per svolgere il compito risulta minore di quello prestabilito. In caso di verità, il valore di tempo ed equivoco saranno costanti (1) per mantenere bilanciato il resto dell'operazione in scala 1:1. Nel caso in cui gli equivoci sono superiori rispetto a quelli tollerati, si misura il tempo nello stesso modo. Tutte le frazioni (T'/T, C'/C, E'/E) servono per stabilire in percentuale la misura di usabilità del sistema, rapportata all'esperienza dell'utente (in base 10). Come numeratore è stato deciso di inserire la riuscita da parte dell'utente così che in caso di successo il risultato sia ben bilanciato e in caso di fallimento il risultato sia 0 a prescindere da qualsiasi altro dato.

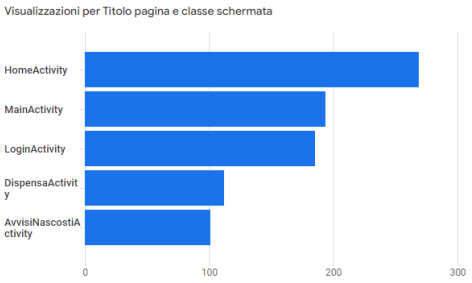
Usabilità aritmetica	Compito 1	Compito 2	Compito 3	Compito 4	Compito 5	Compito 6	
Utente 1	1,00	1,00	1,00	0,55	1,00	1,00	
Utente 2	1,00	1,00	1,00	1,00	1,00	1,00	
Utente 3	1,00	1,00	1,00	1,00	1,00	1,00	
Utente 4	1,00	1,00	1,00	1,00	0,68	1,00	**Medie**
Media aritmetica	1,00	1,00	1,00	0,89	0,92	1,00	**0,97**
Media ponderata	0,80	0,90	0,50	0,62	0,64	0,70	**0,69**

I risultati ottenuti mostrano un'elevata percentuale di peso nella media aritmetica, indicando un alto livello di performance complessiva. Tuttavia, la media ponderata presenta un risultato inferiore, principalmente a causa della semplicità di alcuni dei compiti assegnati. Nonostante ciò, il risultato della media ponderata è comunque soddisfacente e fornisce una visione più bilanciata delle prestazioni complessive.

10 FIREBASE

Firebase è stato il nostro strumento di logging, catturando automaticamente i cambi di activity e gli eventi di "user_engagement" (quando un utente interagisce con l'app per una durata minima) e "screen view" (numero di cambi di schermate).





	Titolo pagina e…asse schermata	Visualizzazioni	Utenti	Visualizzazioni per utente	Durata media del coinvolgimento	Conteggio eventi Tutti gli eventi	Conversioni Tutti gli eventi
		970 100% del totale	16 100% del totale	60,63 Uguale alla media	15 m 30 s Uguale alla media	2.112 100% del totale	16,00 100% del totale
1	HomeActivity	269	14	19,21	7 m 02 s	718	0,00
2	MainActivity	194	16	12,13	2 m 17 s	322	0,00
3	LoginActivity	185	16	11,56	3 m 52 s	261	0,00
4	DispensaActivity	112	13	8,62	1 m 12 s	173	0,00
5	AvvisiNascostiActivity	101	7	14,43	0 m 45 s	103	0,00
6	CreaProdottoActivity	37	9	4,11	1 m 17 s	47	0,00
7	ModificaProdottoActivity	34	11	3,09	0 m 36 s	57	0,00
8	FirstAccessActivity	22	11	2,00	0 m 12 s	32	0,00
9	SignUpActivity	16	9	1,78	0 m 08 s	19	0,00

Il tutto ha reso possibile un'analisi più accurata dei dati di utilizzo, consentendoci di realizzare il seguente studio di usabilità:

I valori attesi sono ricavati da misurazioni medie di utenti considerati quanto più vicini all’utente ideale.

Usabilità: $\begin{cases} T < T' & 1 \\ T \geq T' & \frac{T'}{T} \end{cases}$

NOME ACTIVITY	VALORE ATTESO MEDIO	USABILITÀ
DISPENSA	1’ 50’’	1
AVVISINASCOSTI	0’ 35’’	35/45 = 0,78
CREAPRODOTTO	2’ 00’’	1
MODIFICAPRODOTTO	0’ 40’’	1
		0,78

Per personalizzare il monitoraggio, abbiamo creato degli eventi su misura per le azioni più importanti dell'app, tra cui la modifica del profilo, la creazione e modifica dei prodotti e la possibilità di nascondere gli avvisi.

	Nome evento +	Conteggio eventi	Totale utenti	↑ Conteggio eventi per utente
		2.342 100% del totale	18 100% del totale	130,11 Uguale alla media
1	InCreaProdotto	1	1	1,00
2	first_open	18	18	1,00
3	app_remove	9	9	1,80
4	accountModified	18	9	2,00
5	select_content	2	1	2,00
6	creaProdotto	3	1	3,00
7	session_start	54	18	3,00
8	InPaginaAvvisiNascosti	31	9	3,44
9	ProdottoModificato	21	6	3,50
10	ProdottoCreato	51	9	5,67
11	InHomePage	114	15	7,60
12	user_engagement	145	18	8,06
13	InFunzionalità	114	14	8,14
14	InAccountFragment	125	15	8,33
15	dispensa	9	1	9,00
16	funzionalità	9	1	9,00
17	InDispensa	133	14	9,50
18	noticeSettedHidden	150	15	10,00
19	InAvvisiFragment	187	15	12,47
20	login	241	18	13,39
21	screen_view	1.063	18	59,06

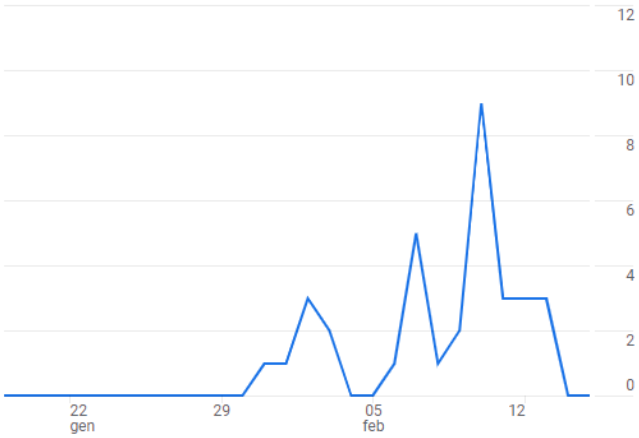
Questi eventi personalizzati ci hanno permesso di tracciare in modo più dettagliato il comportamento degli utenti nell'app e di capire meglio come interagiscono con essa.

Ad esempio, possiamo tracciare per ogni evento definito in quale activity si è verificato.
Come possiamo vedere dalla seguente foto, 11 utenti hanno visualizzato la pagina ModificaProdottoActivity e delle 37 visualizzazioni ci sono stato 21 click sul pulsante di modifica del prodotto. (Questa operazione è ripetibile per tutti gli eventi definiti nella foto precedente).

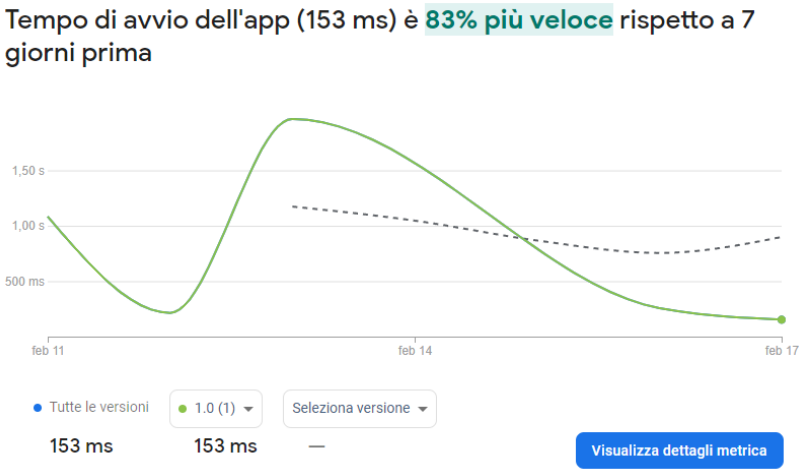
	Titolo pagina e...asse schermata	Visualizzazioni	Utenti	Visualizzazioni per utente	Durata media del coinvolgimento	Conteggio eventi ProdottoModificato
		1.063 100% del totale	18 100% del totale	59,06 Uguale alla media	14 m 37 s Uguale alla media	21 0,9% del totale
1	HomeActivity	287	16	17,94	6 m 17 s	0
2	MainActivity	208	18	11,56	2 m 08 s	0
3	LoginActivity	197	18	10,94	3 m 34 s	0
4	DispensaActivity	133	14	9,50	1 m 11 s	0
5	AvvisiNascostiActivity	107	9	11,89	0 m 36 s	0
6	CreaProdottoActivity	56	10	5,60	2 m 42 s	0
7	ModificaProdottoActivity	37	11	3,36	0 m 38 s	21

Non possiamo vedere il numero di click totali degli utenti per ogni activity, né tantomeno aggiungere log per vedere quando delle operazioni falliscono in quanto non si riuscirebbe a raccogliere dati sufficienti a derivarne un nuovo studio di usabilità quindi non è possibile trarre ulteriori conclusioni.

Firebase non solo fornisce informazioni sul comportamento degli utenti, ma offre anche la possibilità di monitorare le prestazioni dell'applicazione, come il tempo di avvio e i crash. Inoltre, consente di visualizzare il numero e i tipi di dispositivi connessi all'app.



[Utenti per versione]



[Tempo di avvio]