

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE
TECNOLOGIE DELL'INFORMAZIONE
CORSO DI LAUREA TRIENNALE IN INFORMATICA
INGEGNERIA DEL SOFTWARE

DOCUMENTAZIONE RELATIVA AL SISTEMA RATATOUILLE23

Studenti
Francesco Scarfato
Vincenzo Brancaccio

Matricole
N86003769
N86003944

2022-2023

<i>INDICE</i>	1
---------------	---

Indice

I Documento dei requisiti del software	3
1 Presentazione dell'idea progettuale	3
2 Individuazione del target degli utenti	5
2.1 Identikit dell'utente	5
2.1.1 Identikit amministratore/proprietario	5
2.1.2 Identikit dipendente	7
2.2 Personas	8
3 Use Case Diagram	10
3.1 Use Case Diagram Generale	10
3.2 Use Case Diagram per gestire il menù	11
3.3 Use Case Diagram per gestire le ordinazioni	12
3.4 Use Case Diagram per gestire la preparazione piatti	13
4 Tabelle di Cockburn	14
4.1 Inserimento di un elemento nel menù	14
4.2 Registrazione di un ordinazione	15
4.3 Aggiunta di un utente	16
4.4 Eliminazione di un ordinazione	17
5 Mock-up	18
6 Valutazione dell'usabilità a priori	21
6.1 Test di usabilità	22
7 Class diagram di analisi	25
8 Sequence Diagram	28
8.1 Inserimento di un elemento nel menù	28
8.2 Registrazione di un'ordinazione	29
9 Statechart diagram	30
9.1 Inserimento di un elemento nel menù	30
9.2 Registrazione di un'ordinazione	31
9.3 Aggiunta di un utente	31
9.4 Eliminazione di un ordinazione	32
II Documento di design del sistema	33

<i>INDICE</i>	2
10 Analisi dell'architettura	33
10.1 Architettura Hardware	33
10.2 Architettura software	33
11 Descrizione delle tecnologie	35
11.1 Architettura e tecnologie in fase di sviluppo	36
11.2 Architettura e tecnologie in fase di produzione	38
12 Class Diagram di design	40
12.1 Registra nuovo utente	40
12.2 Aggiunta di un nuovo elemento	41
12.3 Aggiunta di una nuova categoria	42
12.4 Eliminazione di un elemento	43
12.5 Eliminazione di una categoria	43
12.6 Modifica di un elemento	44
12.7 Registrazione di una nuova ordinazione	45
12.8 Visualizzazione delle ordinazioni	46
12.9 Modifica-Elimina ordinazione	47
12.10 Gestione delle ordinazioni	48
12.11 Visualizzazione delle statistiche	49
13 Sequence diagram di design	49
13.1 Inserimento di un elemento nel menù	49
13.2 Registrazione di un ordinazione	50
13.3 Registrazione di un utente	50
13.4 Eliminazione di un ordinazione	51
III Testing, Usabilità e Logging	52
14 Testing	52
15 Strategia di testing	52
15.1 Aggiungi utente	52
15.2 Trova categoria in una lista	54
15.3 Aggiungi categoria	55
15.4 Set quantità	57
16 Valutazione dell'usabilità finale	58
17 Logging	62

Parte I

Documento dei requisiti del software

1 Presentazione dell'idea progettuale

Ratatouille23 è un sistema veloce e interattivo per la gestione di servizi di ristorazione. Il sistema è diviso in due parti:

- un applicazione responsiva che supporta dispositivi mobile, desktop e tablet
- un backend scalabile con database annesso, il tutto salvato in cloud

per permettere agli utenti di utilizzarlo comodamente in ogni luogo del locale, sia dov'è disponibile l'uso di una postazione fissa (come ad esempio alla cassa dell'attività) sia in altri spazi che possono essere anche esterni. Inizialmente l'applicativo viene distribuito al cliente, amministratore del locale, con una password di default che al primo accesso dovrà cambiare. Le features del sistema gestionale comprendono:

- la creazione di nuovi utenti
- la gestione del menù
- la gestione delle ordinazioni
 - per il personale di sala
 - per il personale in cucina
- la visualizzazione di statistiche riguardanti l'andamento del locale

Il sistema è fruibile da:

- amministratori
- supervisori
- personale di sala
- addetti alla cucina

Ad ogni classe di utente sono associate delle funzioni (per le specifiche fare riferimento alla sezione 3).

La **creazione di nuovi utenti** permette, solamente all'amministratore, di registrare nuovi utenti e assegnargli un ruolo tra quelli disponibili.

La **gestione del menù** permette di: aggiungere nuove pietanze, categorizzarle, eliminare pietanze ed eliminare categorie. Ad ogni piatto è possibile attribuire varie caratteristiche:

- nome
- costo
- descrizione
- elenco di allergeni comuni

La **gestione delle ordinazioni** è una feature per il personale di sala e per gli addetti in cucina che permette di gestire efficientemente le ordinazioni dal momento in cui vengono registrate fino alla loro evasione.

Permette al personale di sala di prendere l'ordinazione in modo veloce selezionando gli elementi presenti nel menù, mentre consente agli addetti alla cucina di:

- tenere traccia delle ordinazioni prese dal personale di sala
- evadere le ordinazioni quando il piatto è pronto
- controllare le ordinazioni già evase

L'amministratore ha anche una funzione per **visualizzare statistiche** riguardanti l'andamento del suo locale. Le statistiche vengono mostrate tramite grafici interattivi.

2 Individuazione del target degli utenti

Il target principale di *Ratatouille23* sono coloro che possiedono un' impresa di servizi di ristorazione e i loro dipendenti. Analizzando i dati¹, gli esercizi commerciali per cui l'applicazione potrebbe essere utile sono risultati essere ristoranti e bar che rappresentano l'81,6% delle attività ristorative.

	val. assoluti	val. %	n. dipendenti per azienda
bar	205.952	25,9	3,5
discoteche	1.867	0,2	6,8
mense e catering	63.076	7,9	67,3
fornitura di pasti preparati	63.500	8,0	5,4
ristoranti	443.767	55,7	6,2
stabilimenti balneari	18.358	2,3	6,2
Totale	796.520	100	5,5

Fonte: elaborazione C.S. Fipe su dati Inps

Dalla tabella si può evincere anche il numero medio di dipendenti per ogni comparto ristorativo. La media tra tutti i comparti è 5,5.

2.1 Identikit dell'utente

Di seguito cerchiamo di identificare le caratteristiche dell'utente medio tramite l'analisi dei dati. Abbiamo raggrupato gli utenti in 2 categorie:

- Amministratori intesi come proprietari dell'azienda
- Dipendenti dell'azienda

2.1.1 Identikit amministratore/proprietario

Sesso: Sono 683.769 le imprese registrate gestite da uomini ovvero il 71,5%. Mentre sono 112.751 le imprese registrate nel settore ristorazione gestite da donne, pari al 28,5% del totale (52,2% ristoranti, 46,8% bar e appena 1% mense e catering).

Regione	Societa' di capitale	Societa' di persone	Ditte individuali	Altre forme	Totale*
Ristoranti ed attività di ristorazione mobile	23,6	16,6	33,8	28,0	25,9
Fornitura di pasti preparati	23,0	22,2	33,4	31,8	26,6
Bar e caffè	26,7	20,2	42,9	19,9	32,2
Totale servizi di ristorazione	24,5	18,3	37,9	25,5	28,5

Fonte: elaborazione C.S. Fipe su dati Infocamere

¹I dati sono presi dal report del 2021 della FIPE (Federazione Italiana Pubblici Esercizi) per il settore ristorativo. Il report è un *mosaico* di dati presi da ISTAT, INPS e Confcommercio

Età: Sono 50.952 le imprese registrate nel settore ristorazione gestite da under 35, pari al 12,8% del totale così distribuite: 58,0% ristoranti, 41,4% bar e 0,6% mense e catering. Mentre gli over 35 costituiscono l'87,2% quindi sono registrate 745.568 azienda da parte di persone over 35.

Regione	Societa' di capitale	Societa' di persone	Ditte individuali	Altre forme	Totale
Ristoranti ed attività di ristorazione mobile	13,4	5,8	17,9	7,3	13,1
Fornitura di pasti preparati	6,7	4,0	12,3	4,3	7,3
Bar e caffè	14,3	5,2	17,9	6,3	12,9
Totale* servizi di ristorazione	13,5	5,5	17,9	6,6	13,0

Fonte: elaborazione C.S. Fipe su dati Infocamere

Regione di appartenenza: La Lombardia si conferma la prima regione per numero di imprese del settore con una quota sul totale pari al 14,8%, seguita da Lazio (10,8%) e Campania (10,1%).

Regione	Valori assoluti	valori %
Piemonte	23.770	7,0
Valle d'Aosta	1.098	0,3
Lombardia	50.301	14,8
Trentino A.A.	5.645	1,7
Veneto	25.961	7,6
Friuli V. Giulia	7.057	2,1
Liguria	12.410	3,7
Emilia Romagna	25.522	7,5
Toscana	22.706	6,7
Umbria	4.757	1,4
Marche	8.504	2,5
Lazio	36.611	10,8
Abruzzo	8.941	2,6
Molise	1.939	0,6
Campania	34.283	10,1
Puglia	20.193	5,9
Basilicata	2.891	0,9
Calabria	11.290	3,3
Sicilia	24.353	7,2
Sardegna	11.540	3,4
Italia	339.772	100

Fonte: elaborazione C.S. Fipe su dati Infocamere

Nazionalità: L'87,4% sono imprese gestite da italiani, mentre sono poco meno di 50mila le imprese con titolari stranieri attive nel mercato della ristorazione, pari al 12,6% del totale. Incidenza alta al Nord di imprese a gestione a straniera, dove spicca la Lombardia con il 21,8%, e modesta al sud dove merita di essere citato il 3,9% della Campania.

Regione	Ristoranti	Fornitura di pasti preparati	Bar e caffè	Totale servizi di ristorazione
Piemonte	17,8	3,7	11,6	15,1
Valle d'Aosta	8,7	0,0	7,6	8,2
Lombardia	27,4	3,8	15,8	21,8
Trentino A.A.	18,0	4,2	15,2	16,5
Veneto	17,5	5,9	18,7	17,7
Friuli V. Giulia	19,2	15,6	16,0	17,5
Liguria	14,9	5,9	8,2	11,8
Emilia Romagna	18,1	6,8	18,2	18,0
Toscana	14,3	7,5	8,9	12,1
Umbria	12,2	5,2	10,6	11,4
Marche	12,0	2,1	8,9	10,7
Lazio	15,0	4,1	8,3	12,2
Abruzzo	10,6	1,9	9,8	10,1
Molise	8,3	0,0	8,0	8,1
Campania	4,2	2,4	3,5	3,9
Puglia	6,3	3,1	3,9	5,3
Basilicata	4,9	0,0	4,4	4,5
Calabria	5,5	1,3	4,7	5,1
Sicilia	6,3	3,8	3,3	4,9
Sardegna	6,1	2,5	4,2	5,4
Italia	14,3	4,1	10,5	12,6

Fonte: elaborazione C.S. Fipe su dati Infocamere

2.1.2 Identikit dipendente

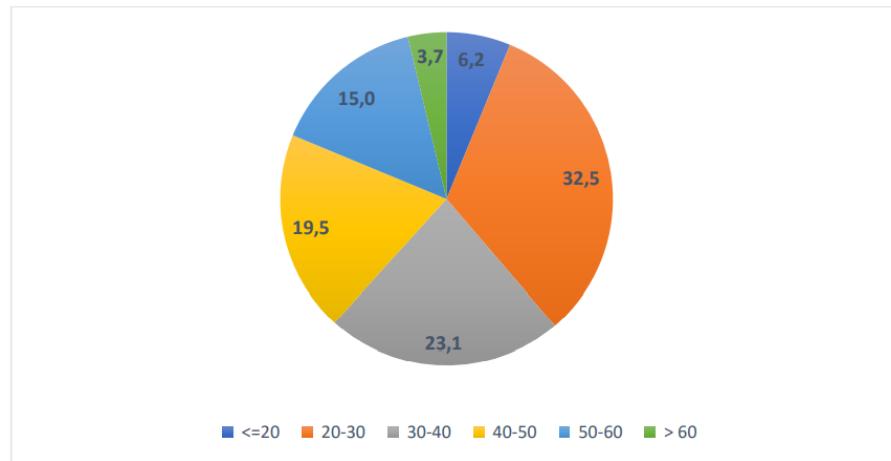
Sesso: Negli esercizi pubblici l'impiego di lavoratrici donne supera quello dei lavoratori maschi: oltre cinque dipendenti su dieci sono donne. Su 796.520 dipendenti il 51,7% sono donne.

Nazionalità: Nel corso degli anni anche la presenza degli stranieri è cresciuta non soltanto tra gli imprenditori, ma anche e soprattutto tra i lavoratori dipendenti la cui quota sul totale si attesta intorno al 24%

		val. assoluti	val. %
Nazionalità	Italiano	605.922	76,1
	Straniero	190.598	23,9
Sesso	Femmina	411.676	51,7
	Maschio	384.843	48,3
Totale		796.520	100

Fonte: elaborazione C.S. Fipe su dati Inps

Età: Il 38,7% ha meno di 30 anni e il 61,8% meno di 40



Fonte: elaborazione C.S. Fipe su dati Inps

2.2 Personas

Tramite l'analisi dettagliata dei dati statistici presentati sopra, sono state generate delle personas in modo da avvicinarci alle idee degli stakeholders per offrire un prodotto in linea con le loro esigenze.



Gregorio Chillari

Età: 47
Citt: Milano
Occupazione: Imprenditore
Salario: 37K / anno

Bio:
Gregorio ha un'attività ristorativa ereditata dal padre da circa 8 anni, è sposato con Grazia e ha 2 figli: un maschio e una femmina. Ha un diploma in ragioneria.

Descrizione generale:

- Preferisce qualità a quantità
- Parla 3 lingue
- Ama viaggiare e scoprire nuovi posti

Personalità

Estroverso → Introverso

Pensiero → Sereno

Vivace → Rilassato

Organizzato → Disorganizzato

Independent → Collaborative

Obiettivi

Vuole insegnare ai figli ad essere sempre coerenti con loro stessi

Vuole garantire un'ottima istruzione ai figli

Vorrebbe modernizzare il suo locale

Motivazioni

Coraggio → Paura

Paura → Crescita

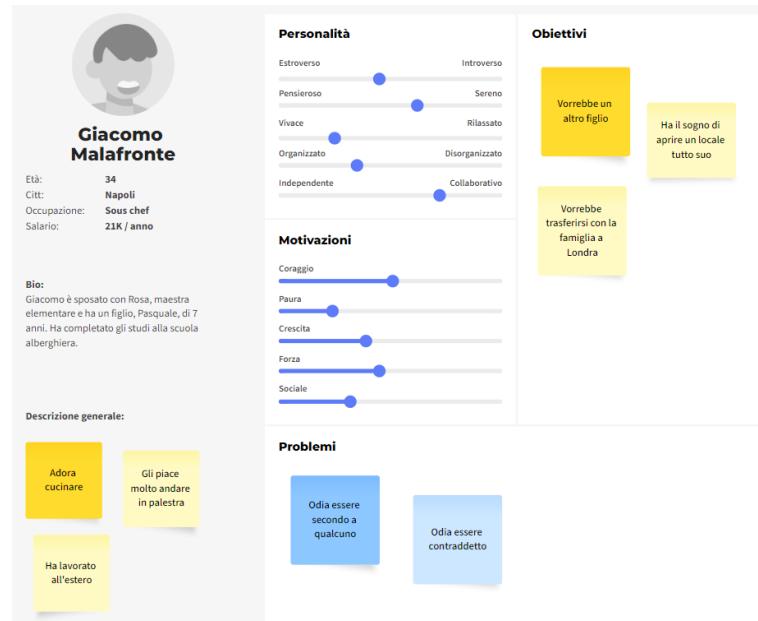
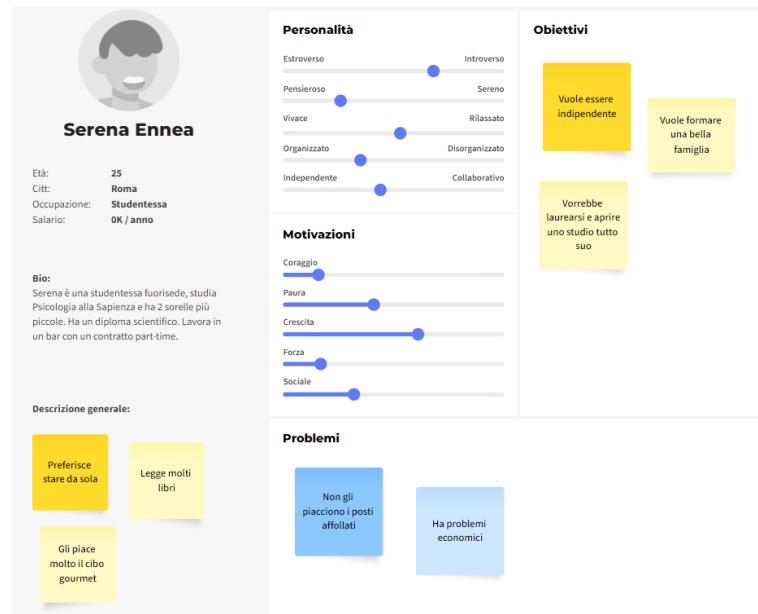
Crescita → Forza

Forza → Sociale

Problemi

Odia i resort e le SPA

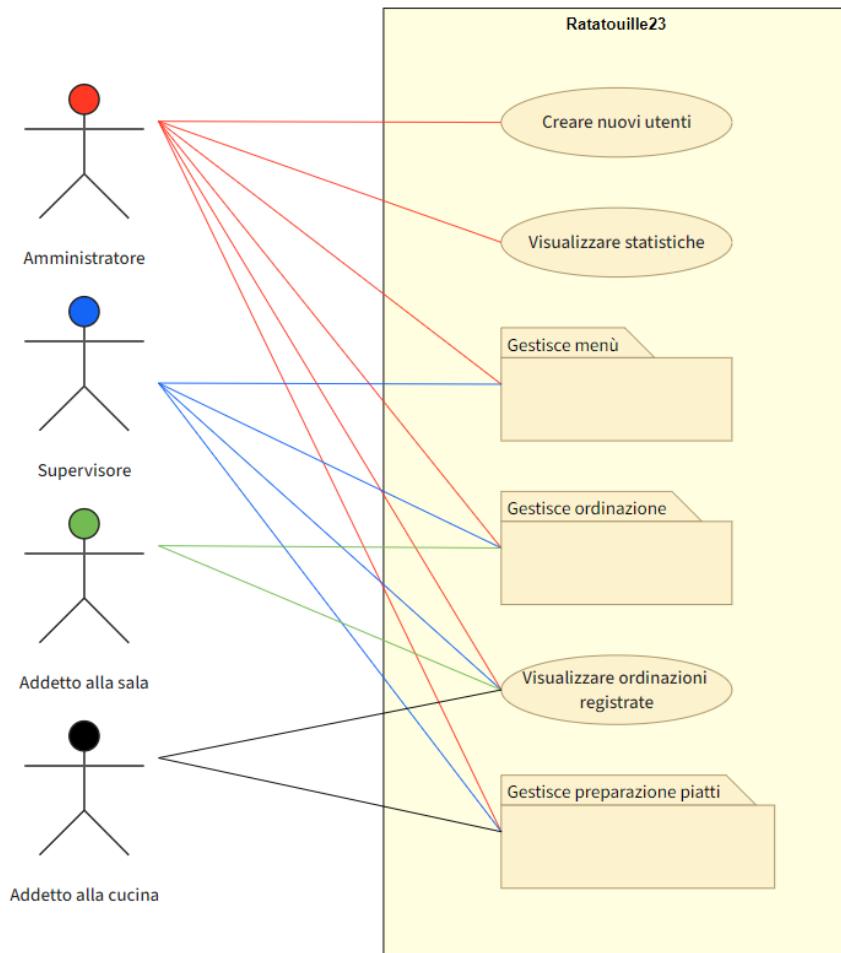
Non gli piace stare da solo



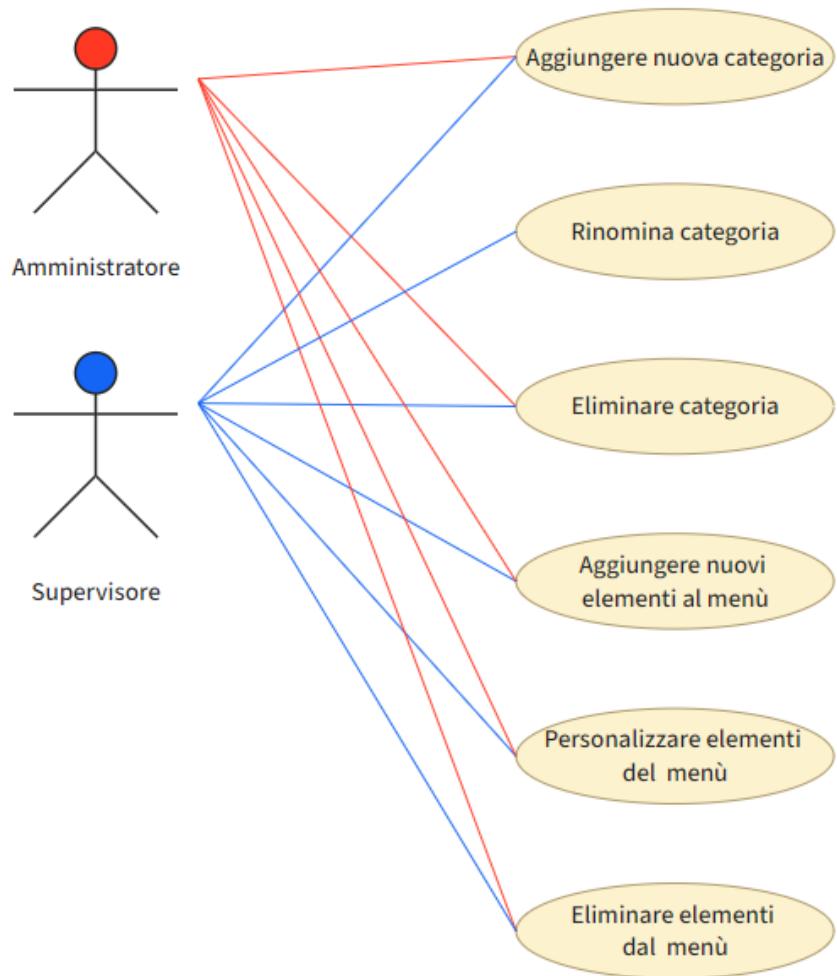
3 Use Case Diagram

In questa sezione, sono presentati gli Use Case Diagrams per esplicitare le funzioni di ogni utente. Per rappresentarli al meglio sono stati usati dei package che raggruppano più casi di uso. Inoltre abbiamo omesso il caso d'uso "Accedi", in quanto associato a tutti gli utenti.

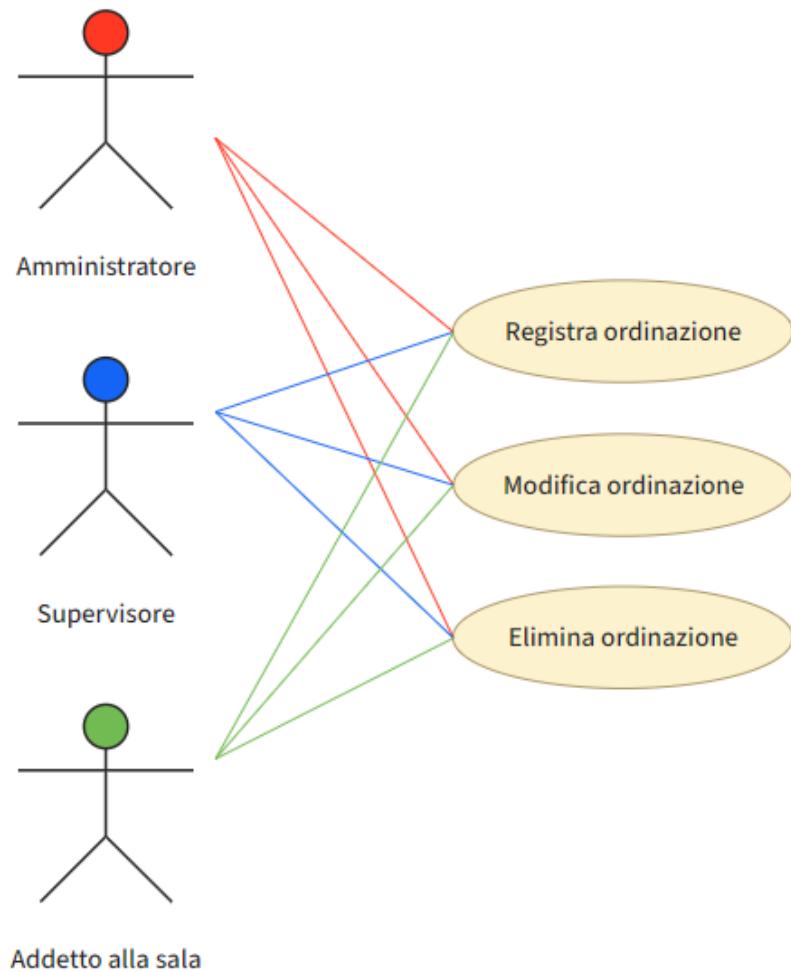
3.1 Use Case Diagram Generale



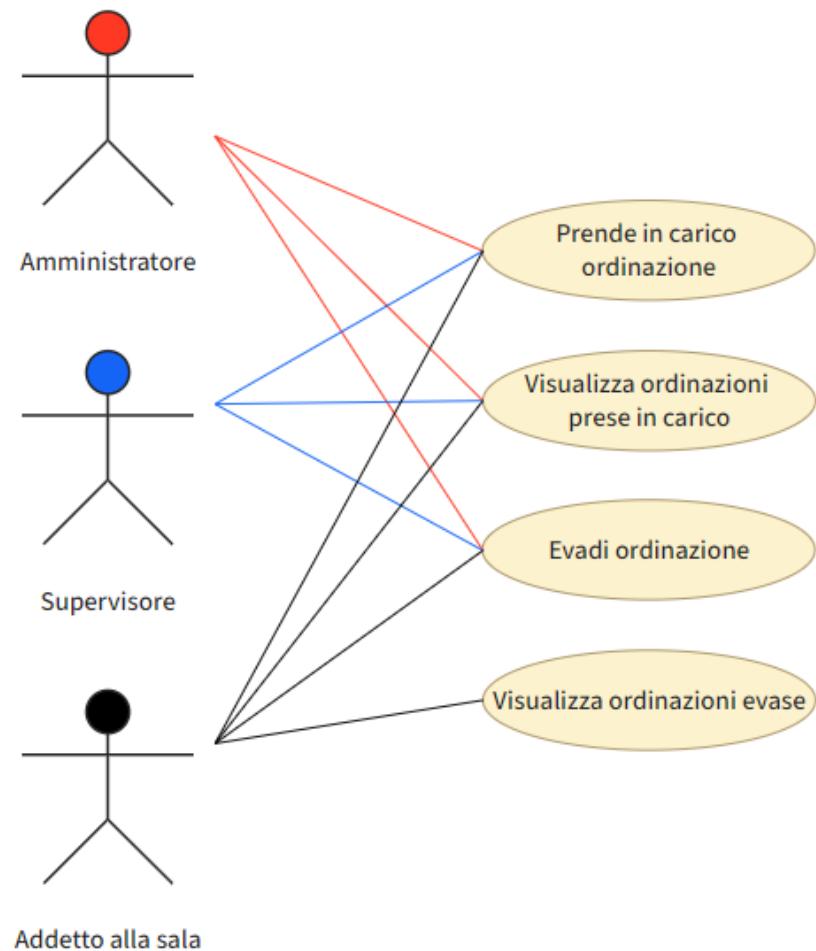
3.2 Use Case Diagram per gestire il menù



3.3 Use Case Diagram per gestire le ordinazioni



3.4 Use Case Diagram per gestire la preparazione piatti



4 Tabelle di Cockburn

In aggiunta agli Use Case, vengono esposte anche le tabelle di Cockburn in modo da visualizzare e specificare in maniera più chiara il flusso dei vari casi d'uso.

4.1 Inserimento di un elemento nel menù

Use case	Inserimento di un nuovo elemento nel menù	
Goal in context	Un amministratore (o un supervisore) desidera aggiungere un nuovo elemento al menù	
Precondition	L'utente deve aver fatto l'accesso e dev'essere creata almeno una categoria	
Success End Condition	Viene inserito il nuovo elemento nel menù	
Failed End Condition	Non viene aggiunto il nuovo elemento al menù	
Primary Actor	Amministratori o supervisori	
Trigger	Preme sull'icona "Aggiungi elemento"	
DESCRIPTION	Step n°	Amministratore o supervisore
	0	Mostra l'interfaccia di gestione del menù
	1	Preme sull'icona "Aggiungi elemento"
	2	Mostra l'interfaccia con i campi da compilare per aggiungere un nuovo elemento
	3	Inserisce il nome del elemento
	4	Inserisce il costo del elemento
	5	Inserisce la descrizione del elemento
	6	Inserisce l'elenco degli allergeni presenti nel elemento
	7	Sceglie la categoria a cui appartiene il elemento
	8	Abilita il bottone "Aggiungi elemento"
	9	Preme "Aggiungi elemento"
EXTENSION: tenta di inserire un elemento con lo stesso nome di un elemento già esistente	Step n°	Amministratore o supervisore
	8.1	Preme "Aggiungi elemento"
	8.2	Mostra un pop-up informativo per avvisare l'utente che un elemento con lo stesso nome è già presente nel menù
	Torna allo step n° 2	

4.2 Registrazione di un ordinazione

Use case	Registrazione di un ordine		
Goal in context	Un addetto alla sala / Amministratore/ supervisore effettua un ordinazione		
Precondition	L'utente ha effettuato l'accesso e sono presenti elementi nel menu		
Success End Condition	Viene effettuata un ordinazione		
Failed End Condition	Non viene effettuata un ordinazione		
Primary Actor	Addetto alla sala, amministratori o supervisori		
Trigger	Preme sull'icona "Registra nuova ordinazione"		
DESCRIPTION	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	0		Mostra interfaccia per la gestione delle ordinazioni
	1	Preme sull'icona "Registra nuova ordinazione"	
	2		Mostra pop-up per inserire il numero del tavolo
	3	Inserisce il numero del tavolo a cui si riferisce l'ordine	
	4	Preme il bottone "Proseguì"	
	5		Mostra interfaccia in cui sono presenti tutti gli elementi del menu
	6	Inserisce gli elementi da ordinare	
	7	Preme il bottone "Visualizza riepilogo"	
	8		Mostra interfaccia che riepiloga tutti gli elementi selezionati
	9	Preme "Conferma ordine"	
	10		Ritorna all'interfaccia di gestione delle ordinazioni
EXTENSIONS: Modifica l'ordine	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	7.1	Preme sull'icona "Modifica"	
	Torna allo step n° 5		

EXTENSIONS: Cancella l'ordine	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	7.2	Preme sull'icona "Cestino"	
Torna allo step n° 0			
EXTENSIONS: Aggiunge delle note all'ordine	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	7.3	Aggiunge delle note all'ordine	
Torna allo step n° 8			
EXTENSIONS: Ordine vuoto	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	10.1		Mostra popup per informare l'utente che non ha inserito nessun'elemento nell'ordinazione
Torna allo step n°8			

4.3 Aggiunta di un utente

Use case	Inserimento di un utente		
Goal in context	Ammministratore		
Precondition	L'utente ha effettuato l'accesso		
Success End Condition	Viene aggiunto un utente		
Failed End Condition	Non viene aggiunto un utente		
Primary Actor	Ammministratore		
Trigger	Preme sull'icona "Aggiungi nuovo utente"		
DESCRIPTION	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	0		Mostra interfaccia per aggiungere un nuovo utente con campi da compilare
	1	Compila il campo "Nome"	
	2	Compila il campo "Cognome"	
	3	Compila il campo "Email"	
	4	Compila il campo "Password"	
	5	Sceglie il ruolo del nuovo utente	
	6	Preme il bottone "Aggiungi Utente"	
EXTENSIONS: Utente già esistente	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	7.1		Mostra popup per informare che l'utente esiste già
	Torna allo step n°0		
EXTENSIONS: Preme il bottone "Indietro"	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	6.1		Torna alla schermata principale

4.4 Eliminazione di un ordinazione

Use case	Eliminazione di un ordine		
Goal in context	Un addetto alla sala / Amministratore/ supervisore elimina un ordinazione		
Precondition	L'utente ha effettuato l'accesso e sono presenti delle ordinazioni		
Success End Condition	Viene eliminata un ordinazione		
Failed End Condition	Non viene eliminata un ordinazione		
Primary Actor	Addetto alla sala, amministratori o supervisori		
Trigger	Preme sull'icona "Ordinazione"		
DESCRIPTION	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
EXTENSIONS: Non ci sono ordini	0		Mostra interfaccia per la gestione delle ordinazioni
	1	Preme sull'icona del cestino	
	2		Mostra pop-up per confermare la volontà di eliminare un'ordinazione
	3	Preme il bottone "Conferma"	
	4		Mostra l'interfaccia per la gestione delle ordinazioni senza l'ordinazione eliminata
EXTENSIONS: Ordinazione già presa in carico	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	1.1	Prova ad aggiornare la lista	
	Torna allo step n°0		
EXTENSIONS: Ordinazione già eliminata	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	2.1		Mostra popup in cui avvisa l'utente che l'ordinazione è già stata presa in carico da un addetto alla cucina
	Torna allo step n° 0		

EXTENSIONS: Annulla l'eliminazione dell'elemento	Step n°	Amministratore o supervisore o addetto alla sala	Sistema
	3.1	Preme il tasto "Annulla"	
	Torna allo step n° 0		

5 Mock-up

In questa sezione viene effettuata una prototipazione visuale con i mock-up per presentare una bozza delle interfacce grafiche dell'applicativo.



Figura 1: Mock up per l'accesso



Figura 2: Mock up per il menù principale

Nome

Cognome

Email

Password

Ruolo

AGGIUNGI UTENTE

Figura 3: Mock up per l'aggiunta di un utente

NON CI SONO ELEMENTI NEL TUO MENU
CREA UNA CATEGORIA E AGGIUNGI UN NUOVO ELEMENTO CLICCANDO IL BOTTOONE

Aggiungi categoria

Aggiungi elemento

Elimina categoria

Nome

Costo

Descrizione

Allergeni

Categoria

AGGIUNGI ELEMENTO

Aggiungi Nuova Categoria

Nome Categoria

Aggiungi

ANNULLA

GLIACCANDO IL BOTTOONE

Primi Secondi di mare Secondi di terra Antipasti Dolci Frutta

Carbonara scomposta Costo: 9,00\$

Spaghetti con le vongole Costo: 9,00\$

Spaghetti con le cozze Costo: 9,00\$

Figura 4: Mock up per la gestione del menù

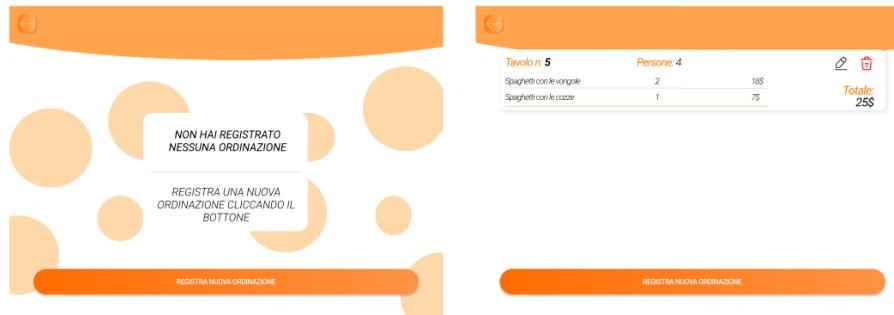


Figura 5: Mock up per la registrazione di nuove ordinazioni

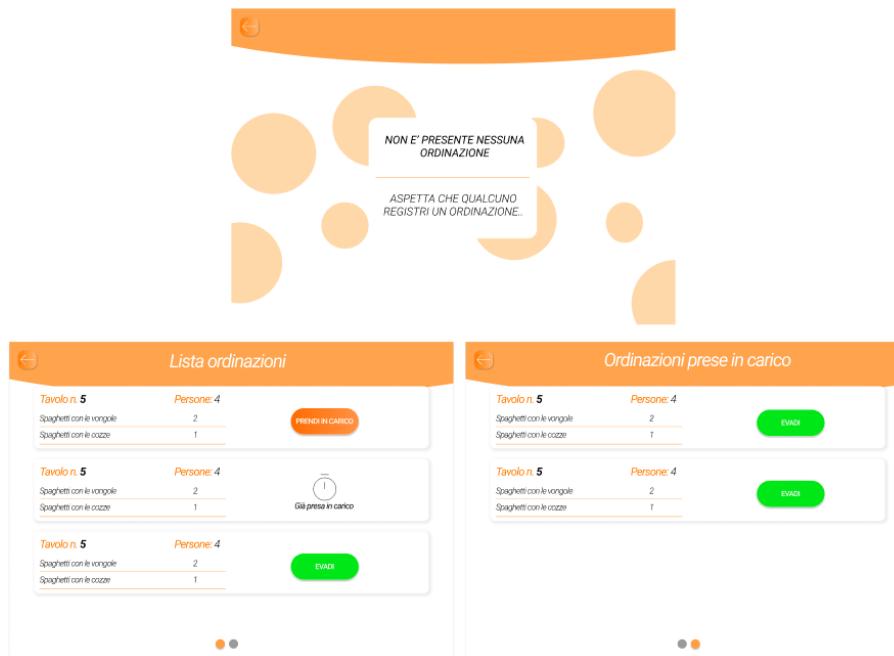
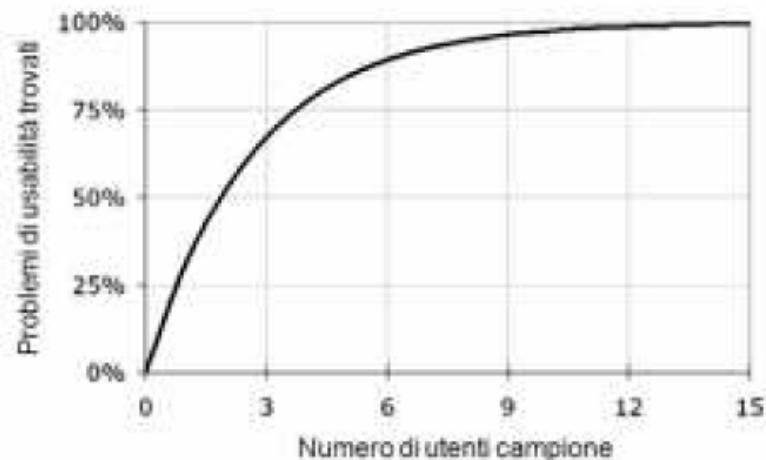


Figura 6: Mock up per la gestione delle ordinazioni

6 Valutazione dell'usabilità a priori

L'usabilità di un prodotto è il grado con cui esso può essere usato da specifici utenti, in un determinato contesto d'uso con efficacia, efficienza e soddisfazione. In accordo con Nielsen abbiamo tenuto conto nella valutazione anche dell'apprendibilità e deòla memorabilità del prodotto per capire la facilità con cui i nuovi utenti prendessero familiarità nell'uso dell'applicativo. Per valutare ciò abbiamo somministrato dei test a 5 diverse persone per ricevere dei feedback sull'applicazione, infatti per la regola di Nielsen, 5 utenti sono sufficienti per mettere in evidenza oltre il 75% dei problemi.



I test sono stati eseguiti su mock-up interattivi sviluppati con l'editor di prototipazione Figma e tramite la tecnica del mago di Oz. La scelta dei tester è stata fatta in modo che essi differissero tra loro per esperienza pregressa, sia per l'uso di tecnologie simili sia per la conoscenza del campo della ristorazione, questo per far sì che i risultati non fossero falsati da un campionamento dei tester troppo omogeneo che non evidenziassero eventuali problemi dovuti a differenze di età, provenienza, familiarità con la tecnologia dei possibili utenti dell'applicazione. Il test somministrato agli utenti prevedeva l'esecuzione senza nessun suggerimento delle seguenti tasks, effettuate sempre partendo dalla schermata iniziale:

- Aggiungere un nuovo utente
- Aggiungere un nuovo elemento
- Evadere un'ordinazione
- Eliminare un elemento

Alla fine del test abbiamo chiesto ai tester quanto ritenessero semplice ed "usabile" l'applicazione, quali sono state le difficoltà maggiori riscontrate durante l'esecuzione delle task e se ci fossero componenti da migliorare o da correggere.

6.1 Test di usabilità

La presentazione dei tester, i risultati e le loro opinioni sono riportati di seguito. Marco è un utente non esperto di ristorazione, ma ha fatto uso di applicativi simili. Inoltre, avendo 24 anni, è abituato all'utilizzo di app per smartphone/tablet.

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	27	Nessuno
Aggiungere un nuovo elemento	Successo	66	Ha cliccato sul pulsante "Cucina"
Evadere un'ordinazione	Successo	83	Ha cliccato sul pulsante "Ordinazioni" e per evadere ha cliccato sulla card dell'ordine
Eliminare un elemento	Successo	37	Nessuno

Feedback finali: Ha ritenuto facile l'uso dell'app e intuitiva la grafica, dando un voto di 8 su 10 per la facilità d'uso. Ha avuto difficoltà a capire le varie aree Ordinazione, Cucina e Menù, consigliandoci di scegliere nomi più chiari.

Nicola è un utente non esperto di ristorazione, ma ha fatto uso di applicativi simili. Inoltre, avendo 23 anni, è abituato all'utilizzo di app per smartphone/tablet.

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	30	Nessuno
Aggiungere un nuovo elemento	Successo	23	Nessuno
Evadere un'ordinazione	Successo	59	Ha cliccato sul pulsante "Ordinazioni"
Eliminare un elemento	Successo	44	Nessuno

Feedback finali: Ha ritenuto facile l'uso dell'app e intuitiva la grafica, dando un voto di 7 per la facilità d'uso. Ha suggerito di cambiare il testo del bottone "Cucina" in "Gestione ordinazioni". Ha ritenuto ingannevoli i bottoni "Menu" e "Cucina" in quanto molto simili.

Utente anonimo, è una donna che ha spesso lavorato nel settore della ristorazione ma non ha mai fatto uso di app simili. A differenza dei primi non è molto abituata ad usre applicazioni mobile il che si noterà soprattutto dal tempo impiegato per ciascuna task.

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	64	Nessuno
Aggiungere un nuovo elemento	Successo	85	Ha cliccato su "Cucina"
Evadere un'ordinazione	Fallito	103	Ha cliccato sul pulsante "Ordinazioni", non riusciva a capire come funzionava l'evasione
Eliminare un elemento	Successo	39	Nessuno

Feedback finali: Ha trovato le icone semplici ed intuitive. Anche i colori sono stati molto apprezzati.

Francesca non ha mai lavorato nel campo ristorativo e non ha mai utilizzato applicazioni simili. Essendo giovane, è abituata a usare applicazioni mobile.

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	55	Nessuno
Aggiungere un nuovo elemento	Successo	61	Ha cliccato su "Ordinazione"
Evadere un'ordinazione	Parziale	86	Ha cliccato sul pulsante "Ordinazioni" ed è riuscita solo a prendere in carico l'ordinazione
Eliminare un elemento	Successo	37	Nessuno

Feedback finali: Ha trovato l'applicazione abbastanza semplice ma non alla portata di tutti. Ha trovato poco intuitivo la gestione delle ordinazioni e ha consigliato di cambiare i colori dei bottoni nel menù principale poichè poco professionali.

Giacomo non ha mai usato app simili e non ha mai lavorato nella ristorazione. Essendo giovane, è abituato a usare applicazioni mobile.

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	18	Nessuno
Aggiungere un nuovo elemento	Successo	21	Nessuno
Evadere un'ordinazione	Successo	33	Ha cliccato sul pulsante "Ordinazioni"
Eliminare un elemento	Successo	11	Nessuno

Feedback finali: Nessuna considerazione finale

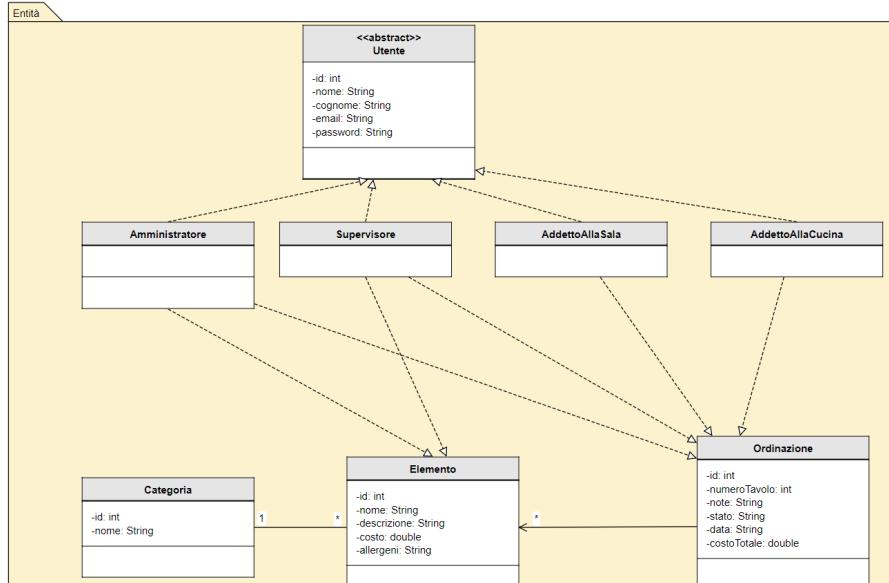
Nella seguente tabella sono evidenziati tutti gli errori commessi dagli utenti.

	Durante l'aggiunta di un elemento ha cliccato su "Cucina"	Durante l'evasione di un ordine ha cliccato sulla card dell'ordine anziché sul bottone	Durante l'evasione di un ordine ha cliccato su "Ordinazioni"	Durante l'aggiunta di un elemento ha cliccato su "Ordinazioni"	Numeri errori totali per utente
Marco	X	X	X		3
Nicola			X		1
Anonima	X		X		2
Francesca			X	X	2
Giacomo			X		1
Numeri errori totali per task	2	1	5	1	

Come evidenziato dalla tabella, l'errore più comune è stato premere sul pulsante che mostra le ordinazioni per l'addetto alla sala invece che premere sul pulsante per visualizzare gli ordini come addetto alla cucina, sebbene l'errore è stato effettuato da tutti abbiamo deciso di non intraprendere misure correttive poichè in realtà l'errore è possibile commetterlo solo dagli admin in quanto gli altri utenti hanno i bottoni relativi agli altri ruoli 'bloccati', inoltre gli admin sono meno della metà degli utenti e statisticamente gli amministratori (che sono i titolari del locale) operano molto poco in cucina. Considerando anche la bassa soglia di apprendimento e l'alta memorabilità dell'applicativo, si ritiene pertanto che quell'errore possa verificarsi raramente.

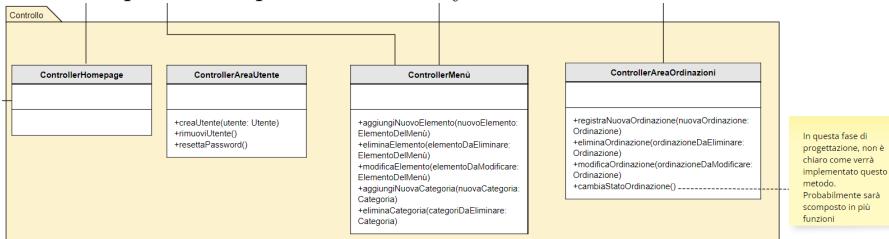
7 Class diagram di analisi

In questa sezione viene presentato il class diagram dell'applicazione *Ratatouille*. Abbiamo optato per un pattern architetturale ECB ovvero Entity-Boundary - Control per suddividere in compiti specifici le varie aree del software. In questa fase di sviluppo e progettazione la maggior parte dei metodi sono solo abbozzati e non è ancora perfettamente chiaro come verranno implementati, infatti sono presenti delle note che lo specificano.

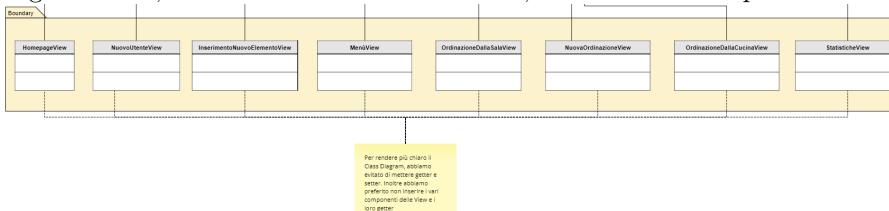


La maggior parte delle entità deriva dagli attori dei vari use cases. In generale rappresentano le informazioni importanti per gli stakeholders ed utili per l'uso e la gestione del software.

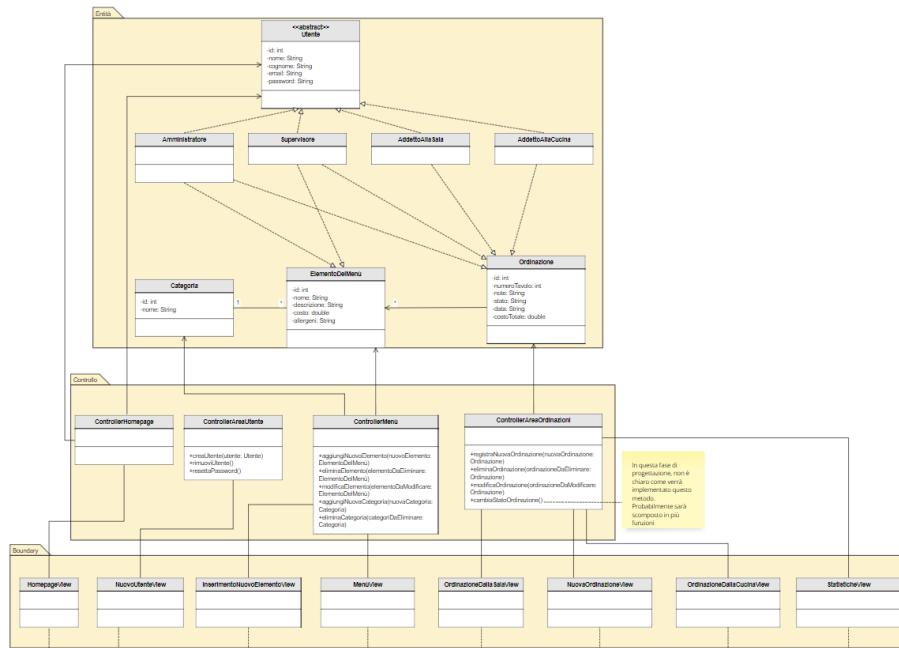
Le entità vengono istanziate e usate dalle classi di controllo. Queste ultime si occupano di come gestire le informazioni e le interazioni dell'utente con le classi comprese nella parte di boundary.



Le classi di boundary rappresentano le GUI (Graphical User Interface) cioè il collegamento tra l'utente esterno e il software. Inoltre catturano gli input degli utenti e, tramite una classe di controllo, forniscono un output.



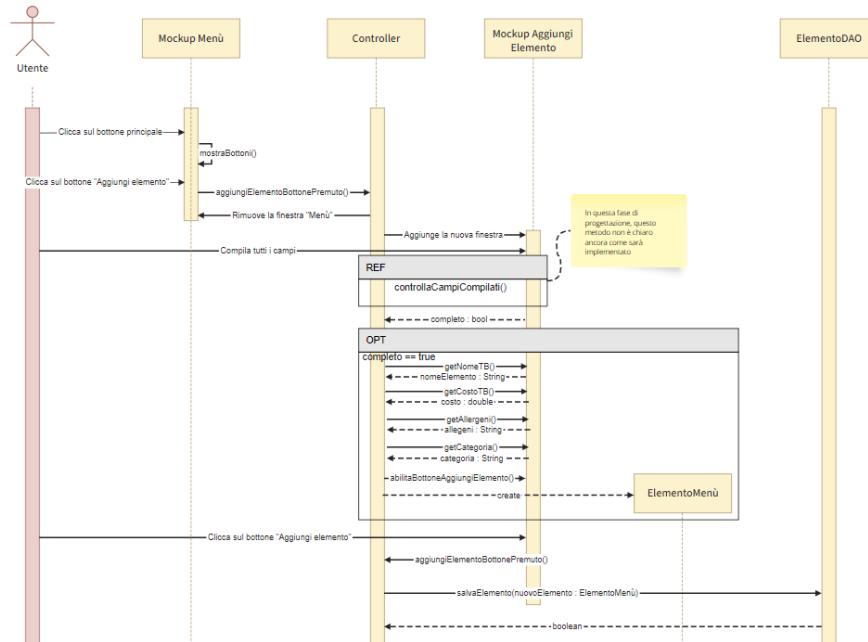
Di seguito viene fornito il class diagram che comprende tutti e 3 i livelli esposti precedentemente.



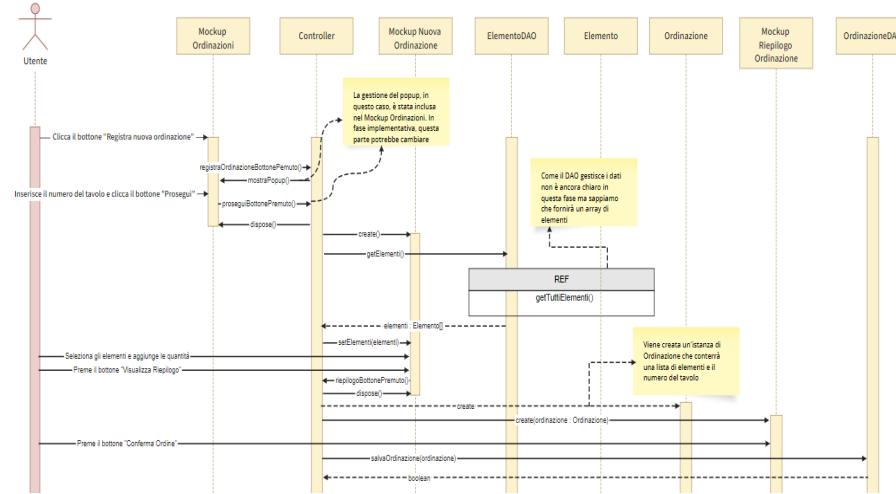
8 Sequence Diagram

Il sequence diagram è una rappresentazione visuale di come le varie classi interagiscono tra di loro, per la gestione di una task, tramite una sequenza di azioni deterministiche.

8.1 Inserimento di un elemento nel menù



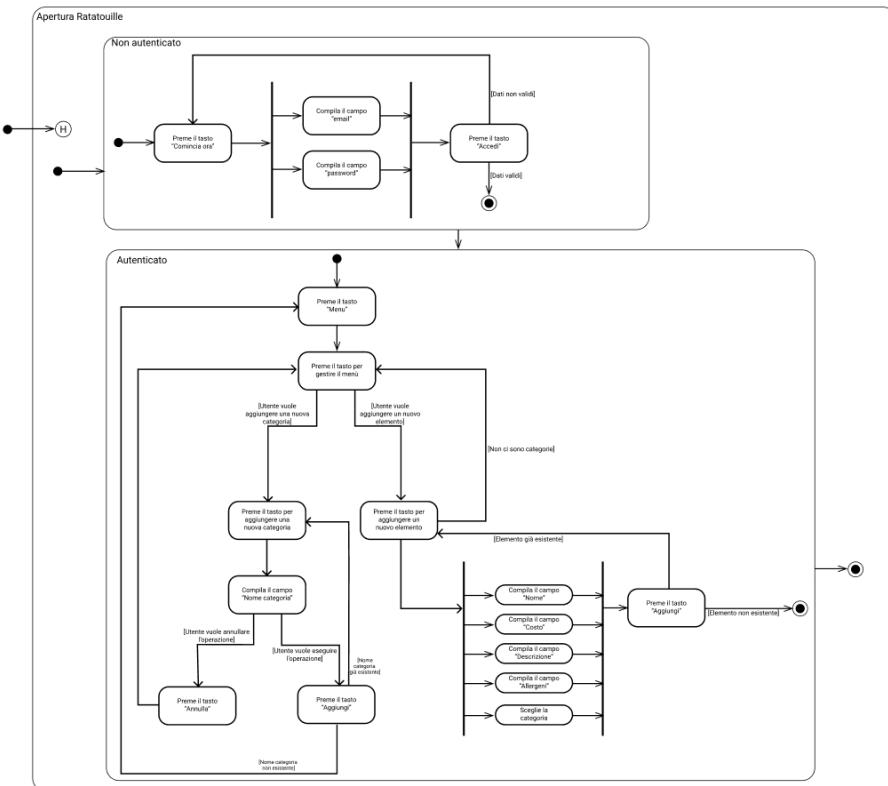
8.2 Registrazione di un'ordinazione



9 Statechart diagram

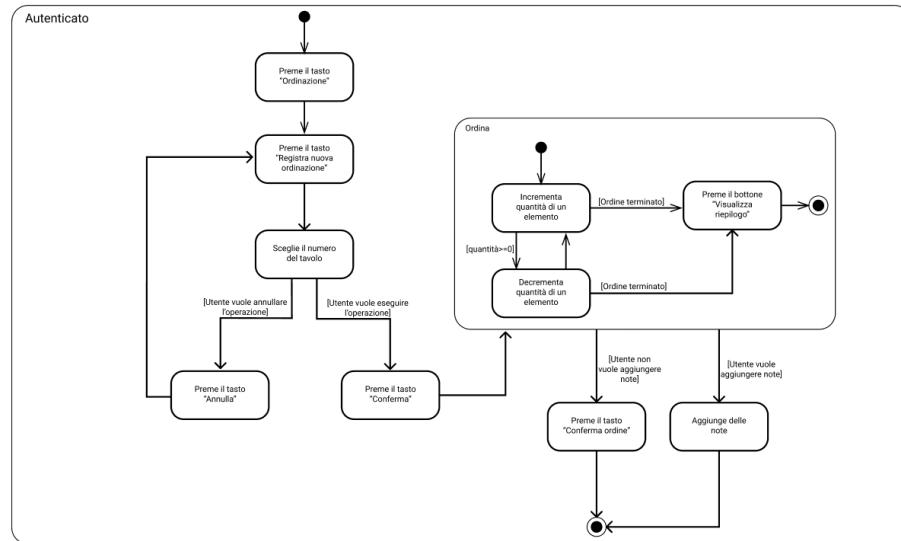
Lo Statechart diagram è molto simile al sequence diagram, in quanto rappresenta una sequenza di azioni per gestire un compito, ma supporta anche scelte diverse.

9.1 Inserimento di un elemento nel menù

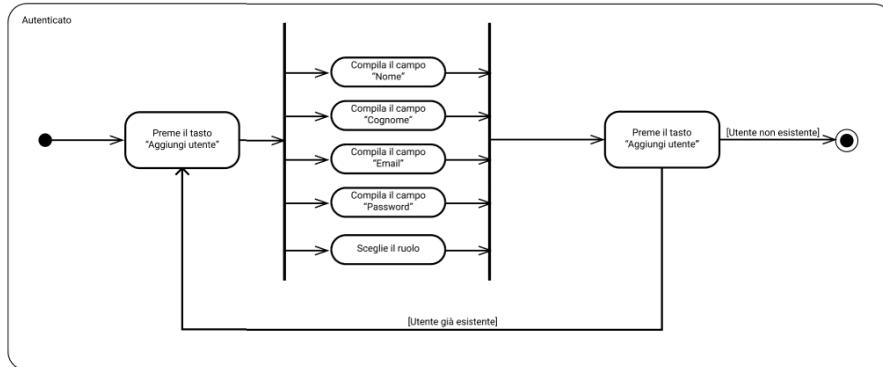


9.2 Registrazione di un'ordinazione

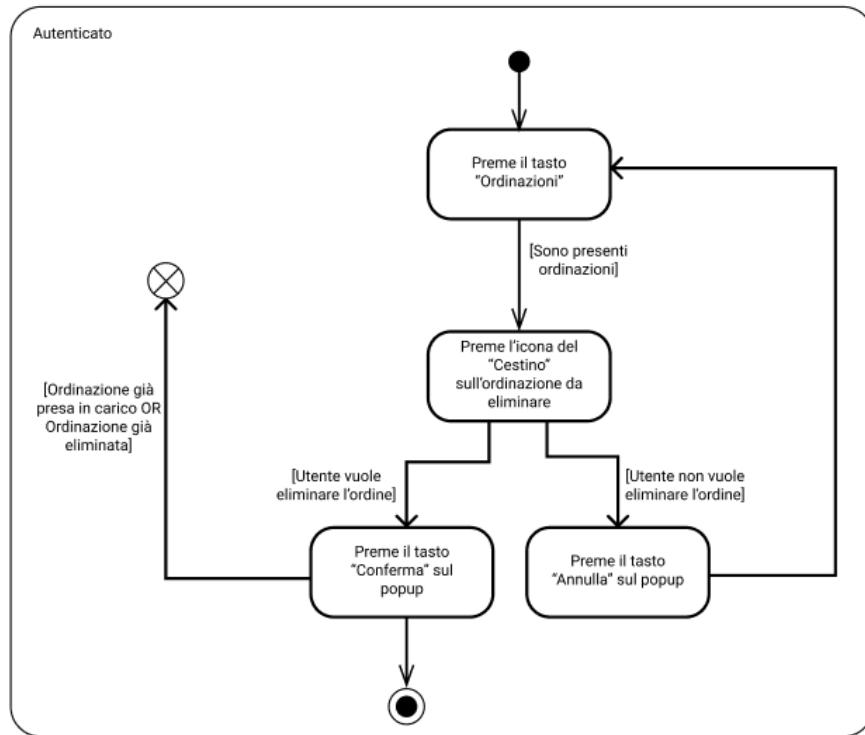
Per brevità è stato omesso lo stato di "Non autenticato", quindi l'utente ha già effettuato l'accesso.



9.3 Aggiunta di un utente



9.4 Eliminazione di un ordinazione



Parte II

Documento di design del sistema

10 Analisi dell'architettura

10.1 Architettura Hardware

Si è optato per un'architettura chiusa a strati. Ogni strato è una decomposizione gerarchica di un blocco software che a sua volta ha una sua organizzazione interna; essendo chiusa ogni livello comunica solo con quello adiacente.

Il nostro sistema ha 3 livelli: client, server e database.

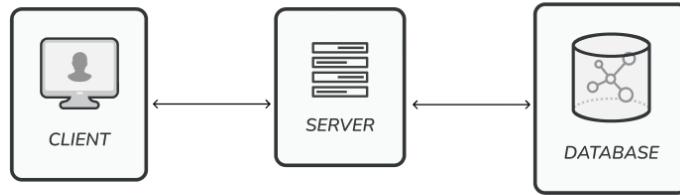


Figura 7: Architettura 3 layer

Da notare che l'architettura è stata descritta seguendo un alto livello di astrazione. Nella sezione 11 sarà esposta più nel dettaglio l'architettura, in fase di sviluppo e in fase di produzione, con le relative tecnologie utilizzate.

10.2 Architettura software

Come anticipato, ogni strato ha una propria architettura interna e in questa sezione andremo ad analizzare l'organizzazione del livello client e del livello server.

Il design pattern utilizzato per il client è Model View Controller Service (MVCS). I livelli sono 4:

- Model, sono i modelli usati per mappare gli oggetti del sistema
- View, presenta l'informazione all'utente, ne cattura le azioni e le gestisce richiamando un Controller
- Controller, gestisce le richieste dell'utente, richiamando i servizi che desidera

- Service, contiene la logica del sistema ed effettua tutte le connessioni a servizi esterni

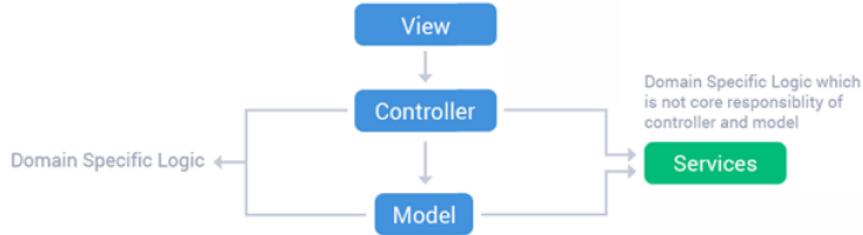


Figura 8: Model View Controller Service pattern

Il design pattern utilizzato per il server è il Controller-Service-Repository. I livelli sono 3:

- Controller, espone API REST e gestisce le richieste del client tramite i service
- Service, contiene la logica dell'applicativo server e ottiene dati tramite repository
- Repository, effettua le operazioni sul database tramite JDBC

Si potrebbe aggiungere un ulteriore livello: Model Layer. Come nel client, anche qui fornisce i modelli degli oggetti e li mappa nel database.

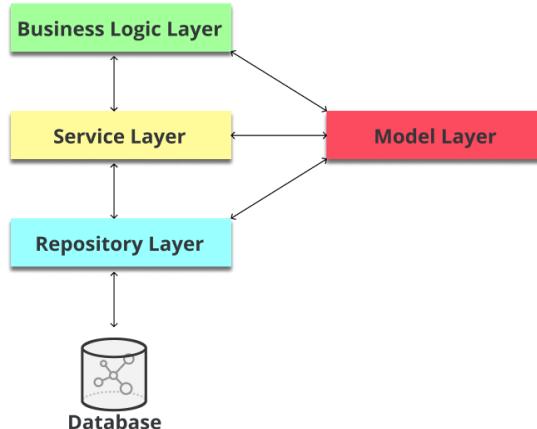


Figura 9: Controller Service Repository pattern

Altri design pattern utilizzati qui sono:

- dependency injection, usato per istanziare classi tramite injection da un componente esterno
- DTO, usato per incapsulare gli oggetti prima di mandarli se hanno bisogno di attributi diversi rispetto all'oggetto originale
- observable, per aggiornare le interfacce grafiche dinamicamente

11 Descrizione delle tecnologie

Come menzionato nella sezione 10.1, il sistema ha un'architettura a 3 strati in cui ognuno di essi utilizza una tecnologia e un linguaggio diverso.

Per il client si è adottato **Flutter**, framework open-source molto popolare di proprietà di Google, che si basa sul linguaggio Dart che è orientato agli oggetti. Le motivazioni alla base di questa scelta sono molteplici:



- la capacità di sviluppare codice multi-piattaforma tramite una singola base di codice che permette di implementare codice portabile
- la costruzione di interfacce grafiche tramite Flutter non richiede componenti native di piattaforme specifiche (a differenza di framework come React e Xamarin)
- semplicità di implementare UI funzionali e ben strutturate tramite i widget messi a disposizione che sono componenti di codice riusabili che permettono un miglioramento notevole di qualità interne del codice come la riusabilità, manutenibilità, riparabilità e portabilità.
- velocità nello sviluppo grazie alla funzione hot reload, in quanto permette allo sviluppatore di vedere in tempo reale i cambiamenti che apporta alla sua applicazione

Spring, invece, è il framework utilizzato per implementare il server. Utilizza come linguaggio di programmazione Java quindi usufruisce di tutti i suoi vantaggi come la Java Virtual Machine.



I principali vantaggi di usare un framework come Spring sono le innumerevoli quantità di estensioni che permettono di avere diverse funzionalità molto comode per lo sviluppo di un web server. Uno dei principali vantaggi è la configurazione automatica per web server come Apache Tomcat e la conseguente esposizione di REST API, la facilità di modellazione del database tramite beans e la semplicità di accesso ai dati tramite JDBC. Altra estensione interessante è Spring Security con cui garantire una comunicazione sicura con il database. Inoltre, come descritto

nella sezione [10.2](#), Spring è basato sul design pattern dependency injection che permette la diminuzione dell'accoppiamento tra classi.

11.1 Architettura e tecnologie in fase di sviluppo

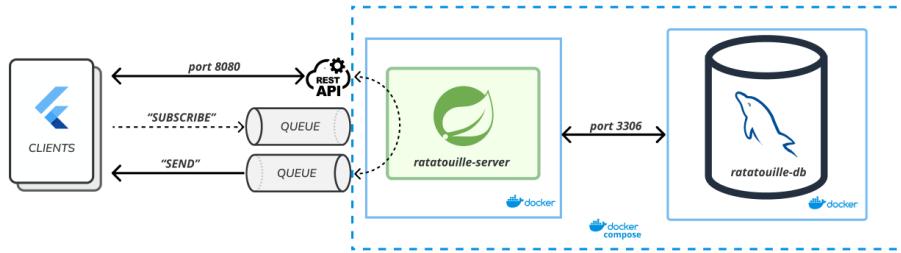


Figura 10: Architettura in fase di sviluppo

Questa è l'architettura a 3 livelli delineata più nel dettaglio. Le caratteristiche più intriganti da sottolineare sono la gestione delle code tramite WebSocket e l'utilizzo dei container tramite Docker. Il protocollo Websocket è sviluppato per superare le limitazioni di HTTP e, in questo caso, è essenziale per implementare la gestione in tempo reale delle ordinazioni. Infatti, offre un canale di comunicazione bidirezionale simultaneo, quindi permette un flusso continuo di dati. Questa tecnologia lavora sul livello di trasporto TCP ma non permette di specificare i tipi di messaggi che vengono mandati/ricevuti. Per questo motivo abbiamo fatto uso di un protocollo di livello più alto che lavora sulle WebSocket ovvero il protocollo STOMP (Streaming Text Oriented Message Protocol). Esso permette uno scambio di piccoli frame tra sorgente e destinazione tramite l'uso di semplici message brokers.

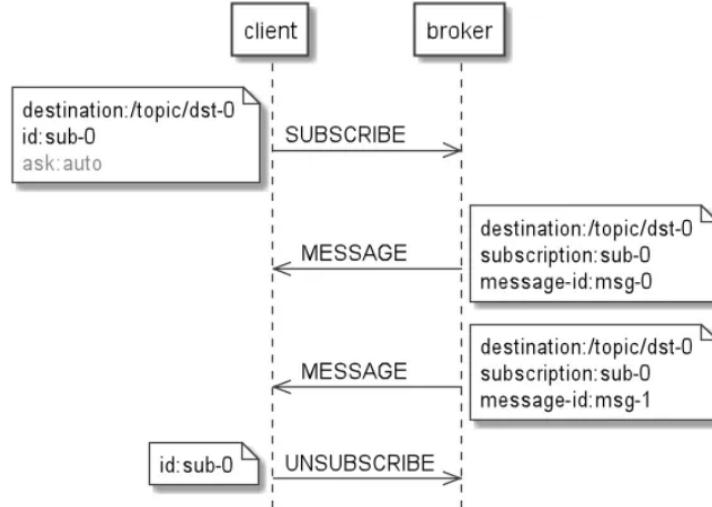


Figura 11: Gestione messaggi con STOMP

Analizziamo il caso in cui sfruttiamo le REST API e le WebSockets, ovvero per la gestione delle ordinazioni in tempo reale per gli addetti alla cucina.

- Il server configura un message broker, espone le REST API e uno STOMP endpoint.
- I clients degli addetti alla cucina mandano un messaggio di tipo *SUBSCRIBE* allo STOMP endpoint con l'id del ristorante a cui appartengono e restano in ascolto.
- Il client di un addetto alla sala registra/modifica/cancella un'ordinazione quindi manda un messaggio HTTP alla porta 8080.
- Il server cattura il messaggio tramite le REST API e invia, tramite JSON, l'ordinazione, con l'operazione da compiere, sulla coda del ristorante corrispondente.
- I clients in ascolto ricevono il messaggio, lo incapsulano in un oggetto e gestiscono l'UI.

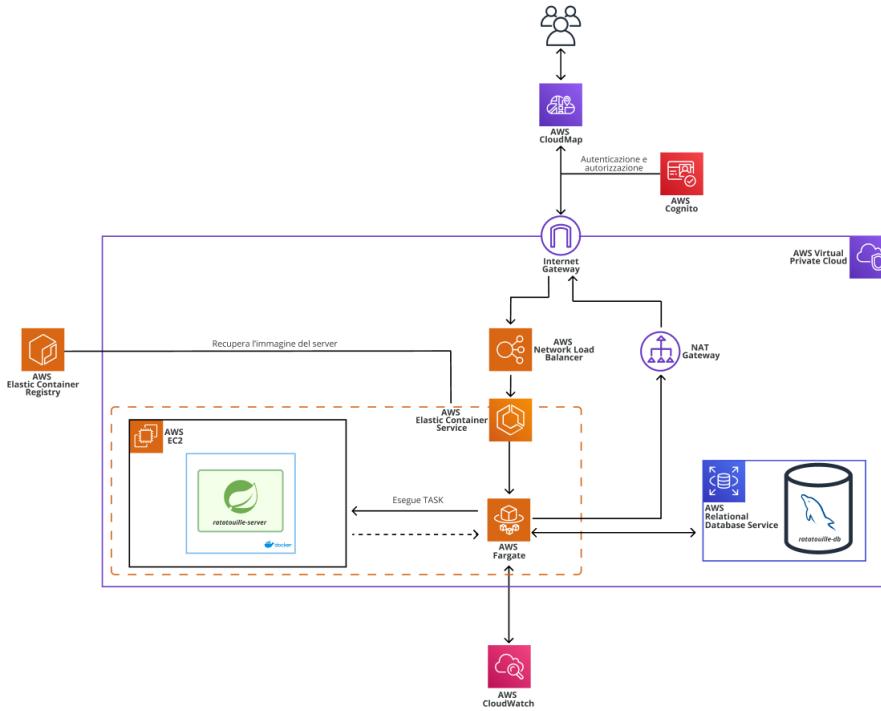
Per quanto riguarda i container, sono degli ambienti isolati e minimali che permettono la facile distribuzione del software containerizzato.

Docker è il software utilizzato per implementare container, creare delle immagini (file per creare container) dell'applicazione e distribuirle pubblicamente tramite Docker Hub. Per far comunicare diversi container, viene usata la funzione docker-compose, altrimenti, essendo i container isolati, non potrebbero comunicare.



In fase di sviluppo, abbiam deciso di containerizzare sia il server che il database. Essendo i container un concetto stateless, containerizzare il database non è vantaggioso poichè nel caso il container venisse eliminato per qualsiasi motivo, i dati andrebbero persi. Ma, in questa fase, è vantaggioso avere un database in un container poichè è possibile modificare la struttura interna del database velocemente e testarlo efficientemente.

11.2 Architettura e tecnologie in fase di produzione



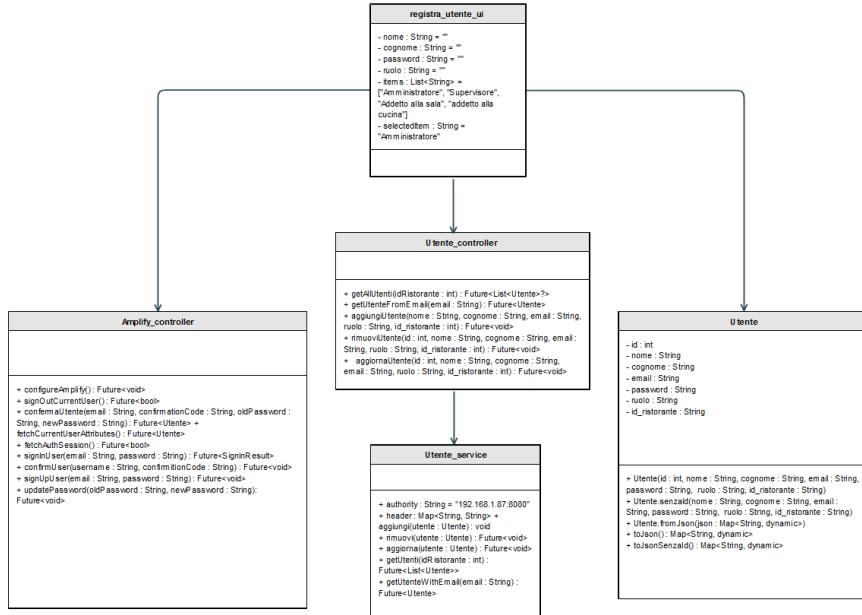
In fase di produzione, si fa largo uso dei servizi web forniti da Amazon. Ora forniamo una descrizione dei vari servizi usati:

- **AWS CloudMap**, è un servizio di rilevamento delle risorse cloud. Con Cloud Map è possibile definire i nomi personalizzati per le risorse dell'applicazione e gestisce la posizione aggiornata di queste risorse che cambiano in modo dinamico. Ciò aumenta la disponibilità dell'applicazione poiché il servizio Web rileva sempre le posizioni più aggiornate delle risorse.
- **AWS Cognito**, è un servizio di gestione dell'identità e dell'accesso dei clienti (CIAM) incentrato sugli sviluppatori e conveniente. Fornisce un archivio di identità sicuro e opzioni di federazione che possono essere dimensionate a milioni di utenti. In questo caso fornisce il token per fare richieste al server.

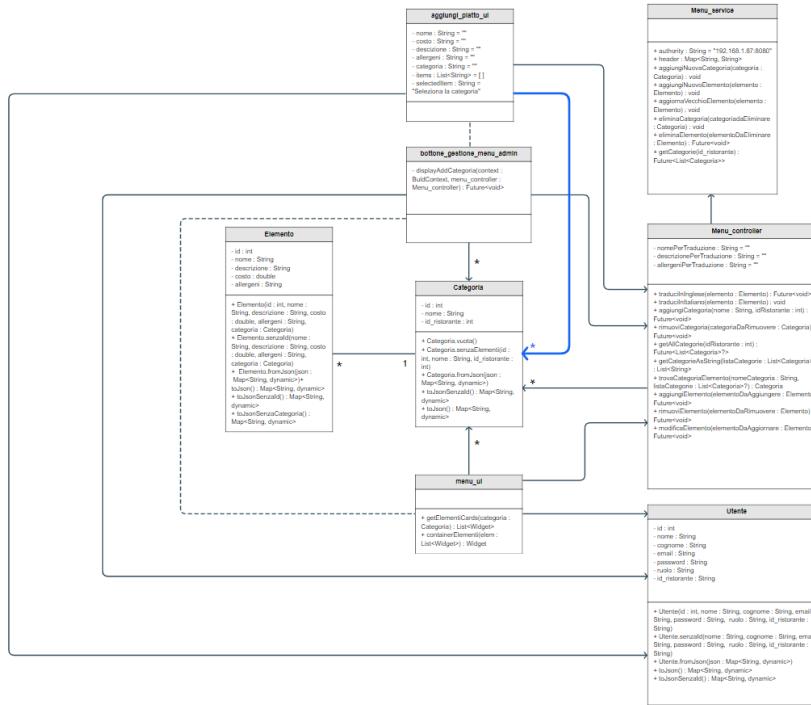
- **AWS Virtual Private Cloud**, è un servizio che permette di avviare risorse AWS in una rete virtuale isolata logicamente definita dall'utente. Gestisce il flusso di dati dall'esterno all'interno della rete tramite vari componenti come Internet Gateway e NAT Gateway. Fornisce all'utente svariate opzioni di sicurezza per filtrare le richieste tramite Security Group.
- **AWS Elastic Load Balancing**, distribuisce automaticamente il traffico in entrata delle applicazioni tra molteplici destinazioni ed appliance virtuali in una o più zone di disponibilità (AZs). In questo caso viene usato un Network Load Balancer che agisce sul livello 4 dello stack di rete.
- **AWS Elastic Container Service**, è un servizio di orchestrazione di container completamente gestito da AWS che aiuta a implementare, gestire e dimensionare facilmente applicazioni containerizzate. Le applicazioni vengono gestite tramite la definizione di tasks. Le immagini delle applicazioni da containerizzare possono essere ottenute tramite Docker Hub o Amazon Elastic Container Registry.
- **AWS Elastic Container Registry**, permette di archiviare ed eseguire facilmente le immagini del container per le applicazioni con uno degli strumenti di orchestrazione (in questo caso ECS).
- **AWS Fargate**, è incorporato con ECS, è un motore Serverless con pagamento in base al consumo che ti permette di concentrarti sulla creazione di applicazioni senza dover gestire i server. Gestirà il dimensionamento e l'infrastruttura necessari per eseguire i container. Istanzierà e gestirà le nostre risorse computazionali (Elastic Cloud Compute o EC2).
- **AWS Relational Database Service**, è una raccolta di servizi gestiti che rende semplice impostare, operare e scalare i database nel cloud. Permette di ricalibrare le risorse di calcolo e di memoria che sostengono la distribuzione. Esegue il backup automatico di database e registri delle transazioni e archivia questi dati per un periodo di conservazione definito dall'utente.
- **AWS CloudWatch**, raccoglie e visualizza log, parametri e dati sugli eventi in tempo reale, in pannelli di controllo automatizzati, per semplificare la manutenzione dell'infrastruttura e delle applicazioni.

12 Class Diagram di design

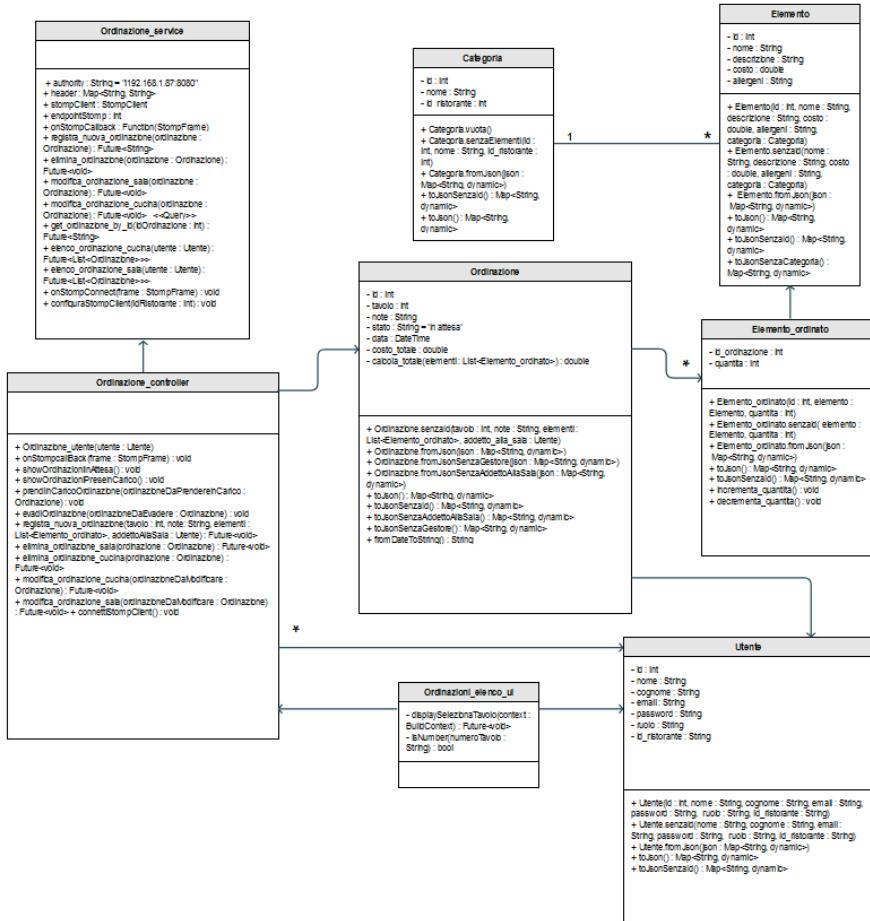
12.1 Registra nuovo utente



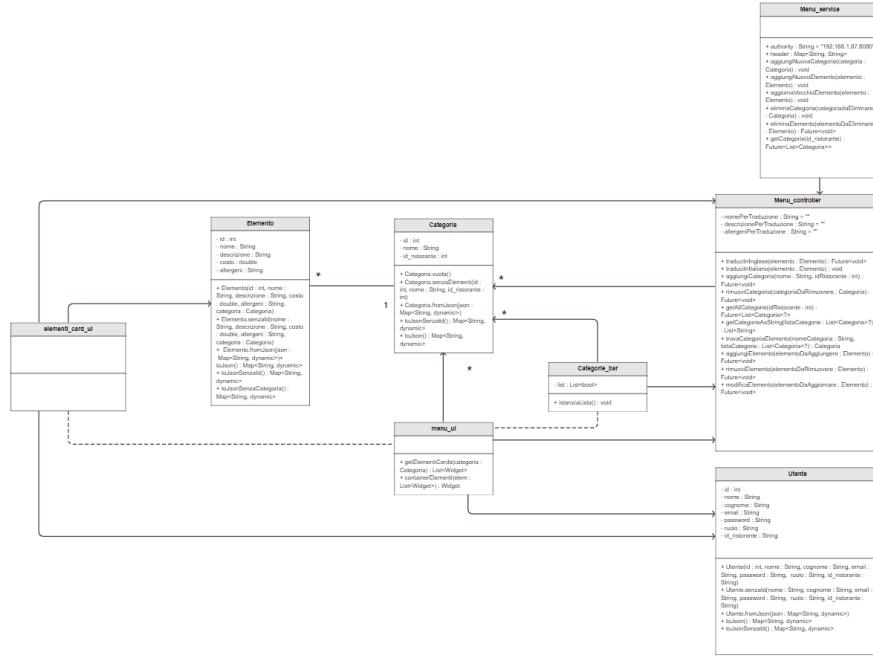
12.2 Aggiunta di un nuovo elemento



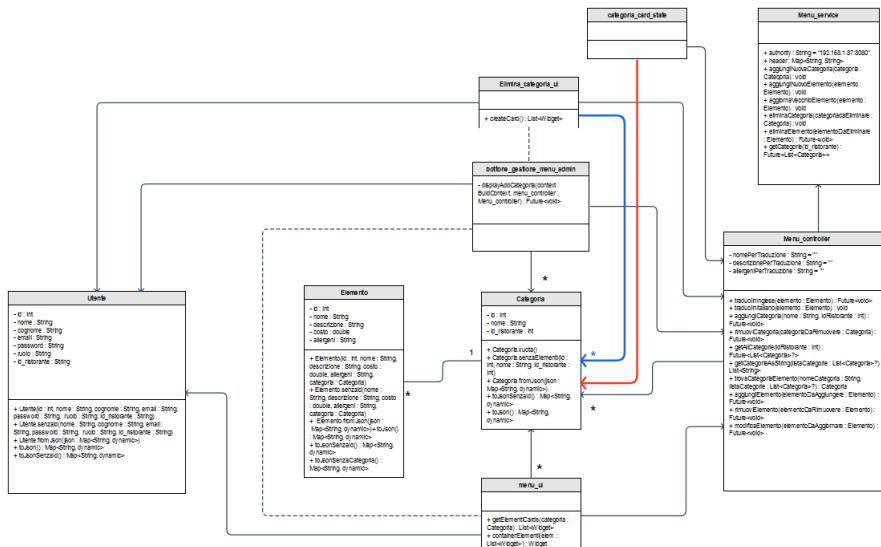
12.3 Aggiunta di una nuova categoria



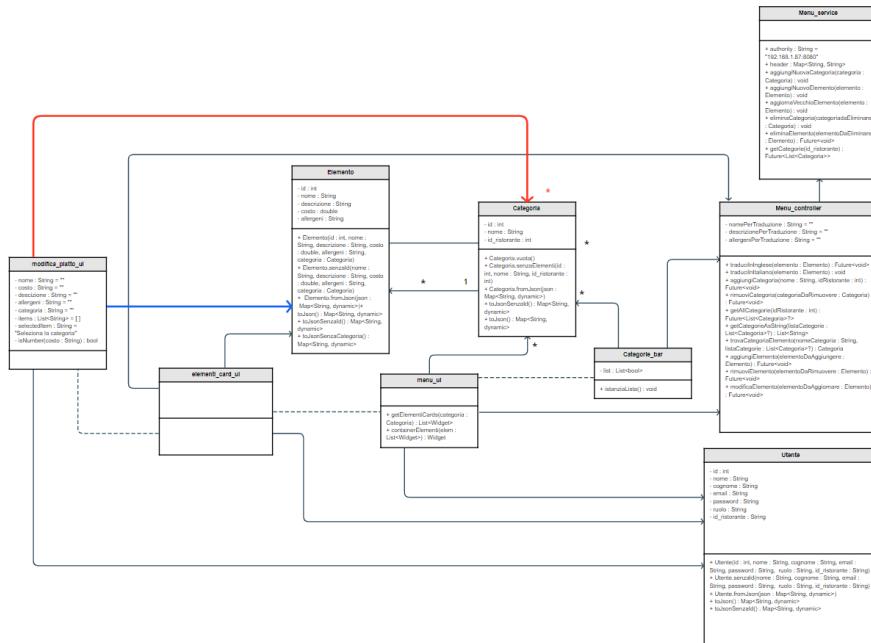
12.4 Eliminazione di un elemento



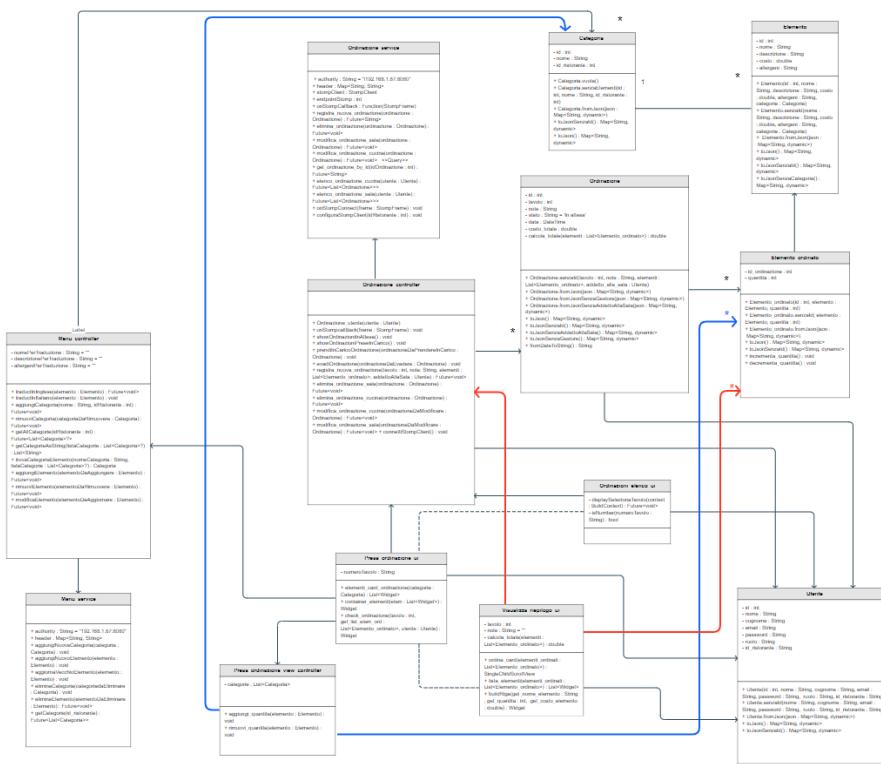
12.5 Eliminazione di una categoria



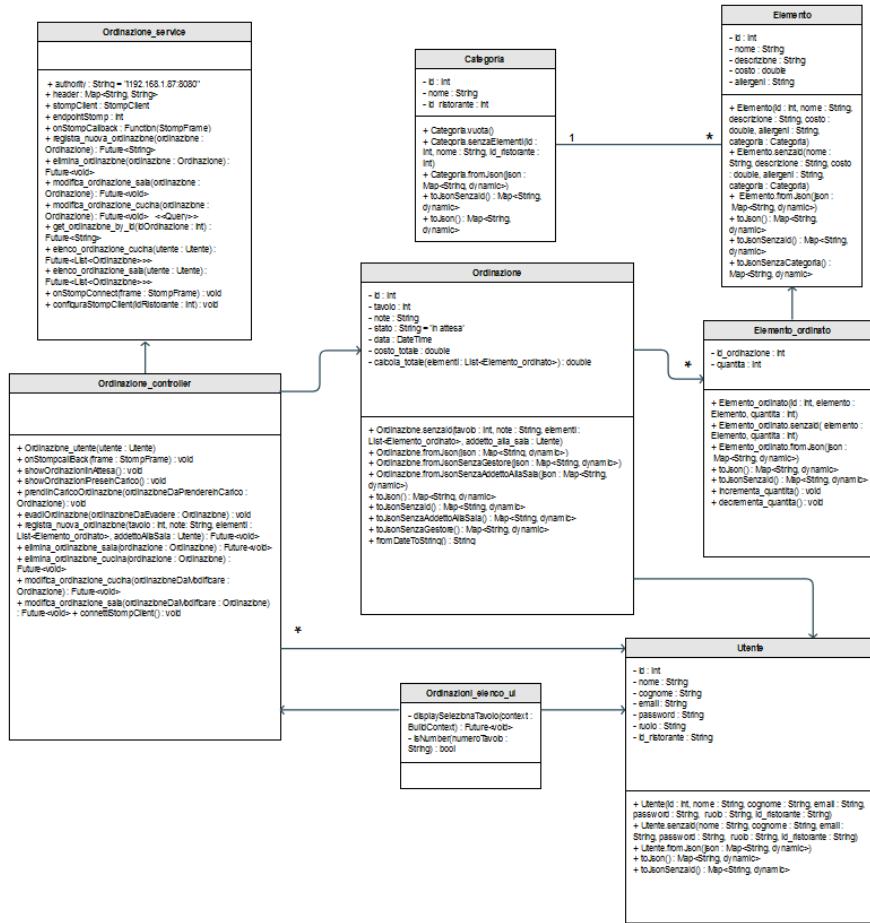
12.6 Modifica di un elemento



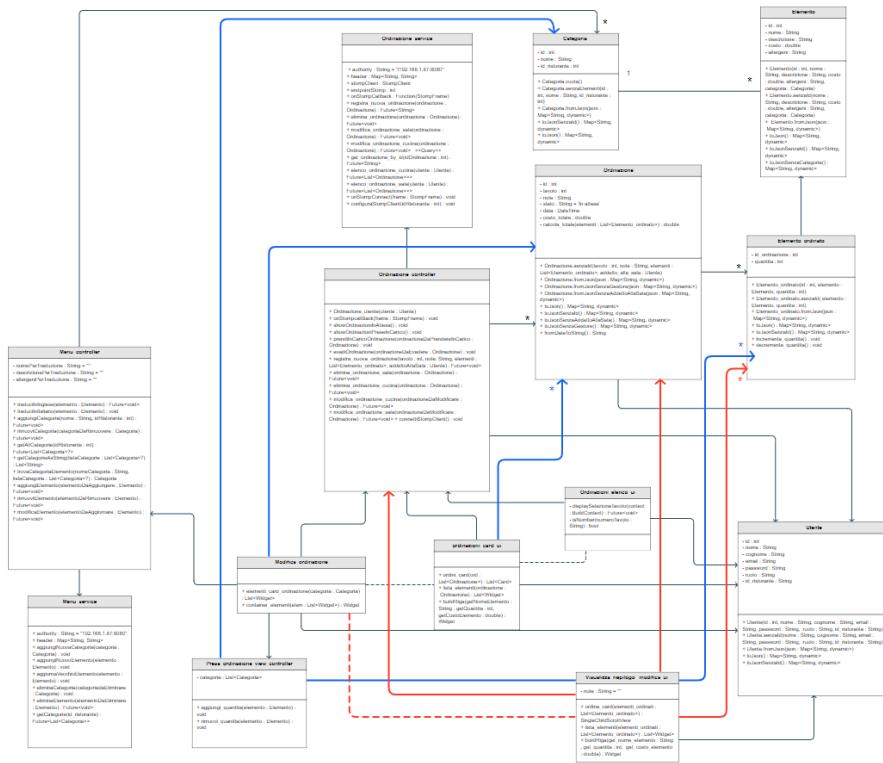
12.7 Registrazione di una nuova ordinazione



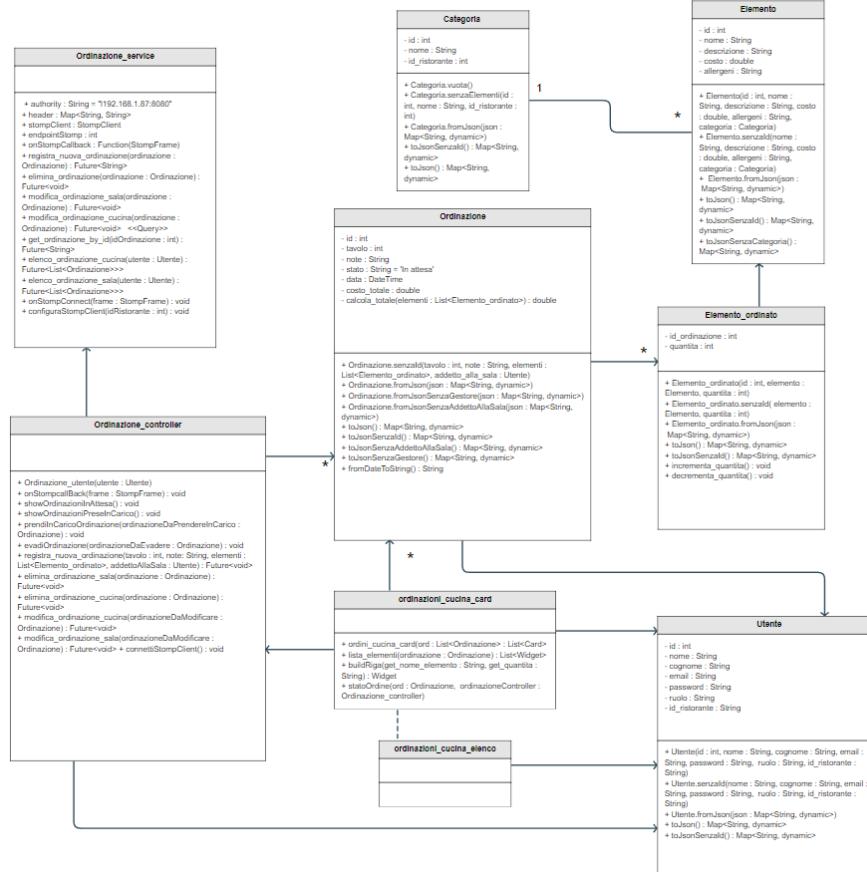
12.8 Visualizzazione delle ordinazioni



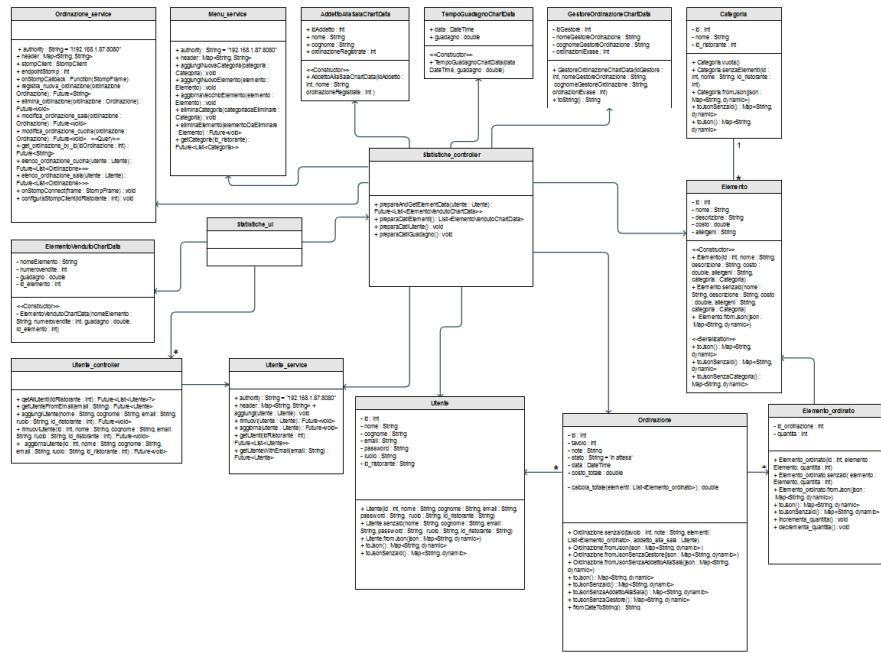
12.9 Modifica-Elimina ordinazione



12.10 Gestione delle ordinazioni

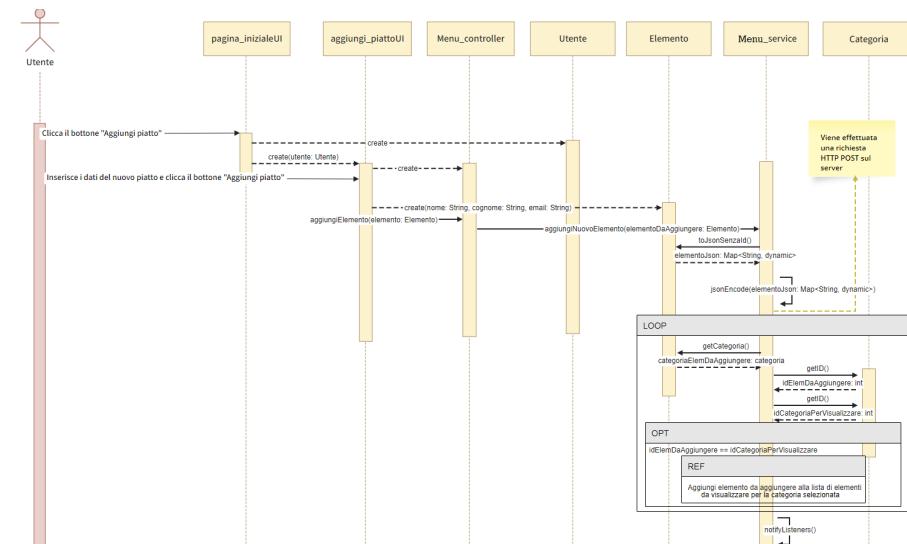


12.11 Visualizzazione delle statistiche

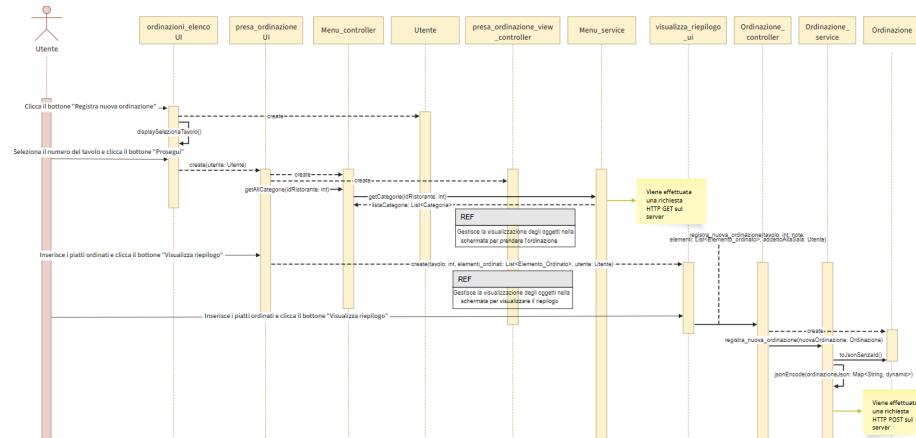


13 Sequence diagram di design

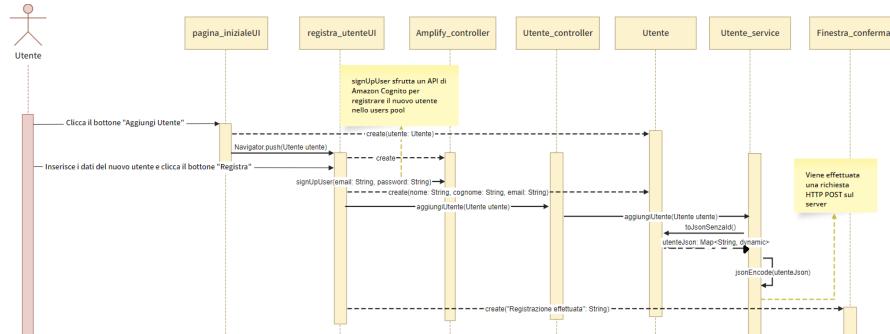
13.1 Inserimento di un elemento nel menù



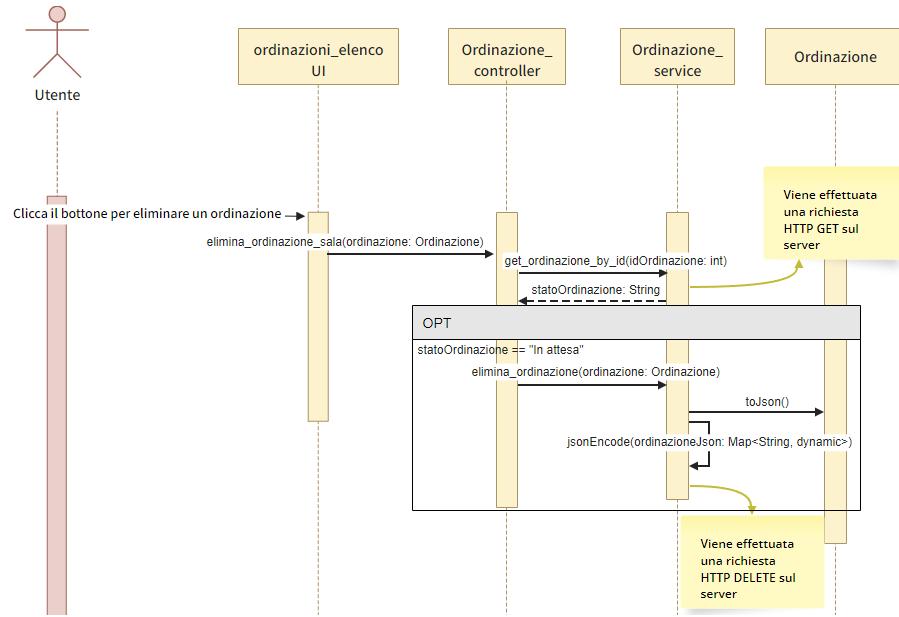
13.2 Registrazione di un ordinazione



13.3 Registrazione di un utente



13.4 Eliminazione di un ordinazione



Parte III

Testing, Usabilità e Logging

14 Testing

Il testing è una parte fondamentale nello sviluppo di un software e nel suo mantenimento. In questo specifico caso è stato effettuato uno Unit testing su 4 metodi. Uno Unit testing è una tecnica di verifica dinamica in quanto il codice viene eseguito. Questo è il livello più basso di testing poichè si concentra sui metodi di una classe. In tutti i metodi è stata usata la tecnica di Scaffolding e, di conseguenza, classi Mock/Stub che simulano delle classi da cui dipendono i metodi che stiamo testando.

15 Strategia di testing

La strategia di testing è White Box per testare tutte le righe di codice dei metodi. Per ogni metodo è stato creato un Control Flow Graph *CFG* in modo da visualizzare tutti i 'brench' del codice. Grazie alla semplicità dei metodi, siamo riusciti a coprire con pochi test case il 100% delle casistiche sia nel 'brench coverage' sia nello 'statement coverage'.

15.1 Aggiungi utente

```

34     Future<void> aggiungiUtente(String nome, String cognome, String email, String ruolo, int id_ristorante) async {
35         try{
36             var _utenteDaAggiungere = Utente.senzaId(nome, cognome, email, ruolo, id_ristorante);
37             await _utente_service.aggiungi(_utenteDaAggiungere);
38         } catch (error) {
39             rethrow;
40         }
41     }

```

Il metodo da testare è nella classe 'Utente controller' e serve per registrare un nuovo utente. Viene istanziata:

- una classe Mock/Stub per simulare la chiamata HTTP effettuata al server dal service
- una classe Mock/Stub per simulare la classe Utente

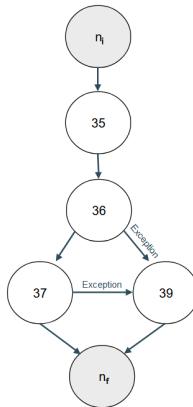


Figura 12: Control Flow Graph di aggiungiUtente

Con 3 test case si riescono a coprire tutti i 'banchi' del CFG.

```

test('testAggiungiUtente_Path_35_36_39', () async {
  try{
    await utente_controller.aggiungiUtente("12345", "Scarfato", "email", "ruolo", 1);
    fail("Test fallito");
  } on Exception {
    expect(true, true);
  }
});

test('testAggiungiUtente_Path_35_36_37', () async {
  try{
    when(() => utente_service_stub.aggiungi(any())).thenAnswer((_) => Future.value());
    await utente_controller.aggiungiUtente("Francesco", "Scarfato", "email", "ruolo", 1);
    expect(true, true);
  } on Exception {
    fail("Test fallito");
  }
});

test('testAggiungiUtente_Path_35_36_37_39', () async {
  try{
    //response.statusCode!=200
    when(() => utente_service_stub.aggiungi(any())).thenThrow(Exception());
    await utente_controller.aggiungiUtente("Francesco", "Scarfato", "email", "ruolo", 1);
  } on Exception {
    expect(true, true);
  }
});
  
```

Figura 13: Batteria di test per aggiungiUtente

15.2 Trova categoria in una lista

```

121     Categoria trovaCategoriaElemento(String nomeCategoria, List<Categoria>? listaCategorie){
122         int i=0;
123         while(listaCategorie?[i].get_nome() != nomeCategoria) {
124             i=i+1;
125         }
126         return listaCategorie![i];
127     }

```

Il metodo da testare è nella classe 'Menu controller' e serve per cercare una categoria in una lista tramite il nome. Per simulare la lista di categorie viene definita una classe Mock/Stub per 'Categoria', vengono istanziate diverse di queste classi e aggiunte a una lista.

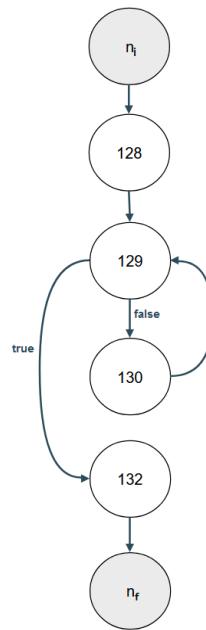


Figura 14: Control Flow Graph di trovaCategoria

Serve 1 solo test case in quanto, basta eseguire almeno una volta l'istruzione nel body del while e poi uscire per verificare tutti i vari rami del CFG.

```

group('trovaCategoriaElementoTEST', () {
    List<CategoriaSTUB> listaCategorie = <CategoriaSTUB>[];

    setUp(() {
        listaCategorie.add(CategoriaSTUB("Primi"));
        listaCategorie.add(CategoriaSTUB("Secondi"));
    });

    test('trovaCategoriaElemento_Path_128_129_130_132', () {
        expect(menu_controller.trovaCategoriaElemento("Secondi", listaCategorie), isA<CategoriaSTUB>());
    });
});

```

Figura 15: Test case per trovaCategoria

15.3 Aggiungi categoria

```

75     Future<void> aggiungiCategoria(String nome, int idRistorante) async {
76         try{
77             var categoriaDaAggiungere = Categoria.senzaIdAndElementi(nome, idRistorante);
78             _listaCategorie.add(categoriaDaAggiungere);
79             await _menu_service.aggiungiNuovaCategoria(categoriaDaAggiungere);
80             notifyListeners();
81         } on HttpException {
82             rethrow;
83         }
84     }

```

Il metodo da testare è nella classe 'Menu controller' e serve per aggiungere una nuova categoria. Viene istanziata:

- una classe Mock/Stub per simulare la chiamata HTTP effettuata al server dal service
- una classe Mock/Stub per simulare la classe Categoria

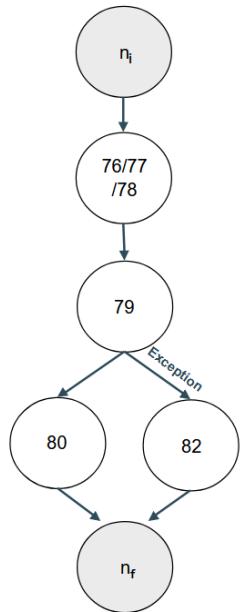


Figura 16: Control Flow Graph di aggiungiCategoria

Con 2 test case riesco a coprire tutti i 'banchi' del CFG.

```

group('aggiungiCategoria TEST', () {
  late MenuServiceSTUB menuServiceSTUB;

  setUp(() {
    menuServiceSTUB = MenuServiceSTUB();
    registerFallbackValue(CategoriaSTUB("Secondi"));
  });

  test('aggiungiCategoria_Path_76_77_78_79_80', () {
    try {
      when(() => menuServiceSTUB.aggiungiNuovaCategoria(any())).thenAnswer((_) => Future.value());
      menu_controller.aggiungiCategoria("Secondi", 1);
      expect(true, true);
    } on HttpException {
      fail("Test fallito");
    }
  });

  test('aggiungiCategoria_Path_76_77_78_79_82', () async {
    try {
      menu_controller = Menu_controller.service(menuServiceSTUB);
      when(() => menuServiceSTUB.aggiungiNuovaCategoria(any())).thenThrow(HttpException("statusCode"));
      await menu_controller.aggiungiCategoria("Secondi", 1);
    } on HttpException {
      expect(true, true);
    }
  });
});

```

Figura 17: Batteria di test per aggiungiCategoria

15.4 Set quantità

```

41   ↴ set_quantita(Elemento elemento, int quantita){
42     int i = 0;
43     ↴ while(_list_elem_ord[i].get_elemento().id != elemento.id){
44       i++;
45     }
46     ↴ _list_elem_ord[i].set_quantita(quantita);
47   }

```

Il metodo da testare è nella classe 'Presa ordine view controller' e serve per cercare un elemento in una lista e definirne la quantità in un ordinazione. Per simulare la lista di elementi viene definita

- una classe Mock/Stub per 'Elemento Ordinato', vengono istanziate diverse di queste classi e aggiunte a una lista
- una classe Mock/Stub per 'Elemento'

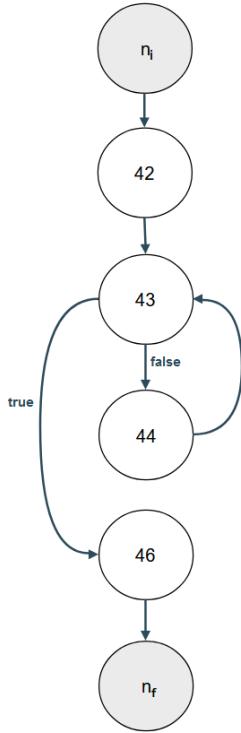


Figura 18: Control Flow Graph di set quantità

Serve 1 solo test case in quanto, basta eseguire almeno una volta l'istruzione nel body del while e poi uscire per verificare tutti i vari rami del CFG.

```

group('set_quantitaTEST', () {
    setUp() => presa_ordinazione_view_controller.set_list_elem_ora(<ElementoOrdinatoSTUB>[ElementoOrdinatoSTUB(0), ElementoOrdinatoSTUB(1)]);

    test('setQuantita_Path_42_43_44_46', () {
        presa_ordinazione_view_controller.set_quantita(ElementoSTUB(1), 2);
        expect(presa_ordinazione_view_controller.list_elem_ord[1].get_quantita(), 2);
    });
});
  
```

Figura 19: Test case per set quantità

16 Valutazione dell'usabilità finale

Al termine dell'implementazione, sono stati ripetuti test per valutare l'usabilità. I test scelti sono gli stessi effettuati sui prototipi con l'aggiunta del test avente come compito la presa di un ordinazione. I test sono stati somministrati agli stessi utenti con i risultati riportati di seguito.

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	23	Nessuno
Aggiungere un nuovo elemento	Successo	68	Ha cliccato sul pulsante "Cucina"
Evadere un'ordinazione	Successo	79	Nessuno
Eliminare un elemento	Successo	39	Nessuno
Effettuare un'ordinazione	Successo	28	Nessuno

Figura 20: Tabella di usabilità per Marco

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	32	Nessuno
Aggiungere un nuovo elemento	Successo	25	Nessuno
Evadere un'ordinazione	Successo	54	Ha cliccato sul pulsante "Ordinazioni"
Eliminare un elemento	Successo	40	Nessuno
Effettuare un'ordinazione	Successo	41	Nessuno

Figura 21: Tabella di usabilità per Nicola

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	63	Nessuno
Aggiungere un nuovo elemento	Successo	81	Ha cliccato su "Cucina"
Evadere un'ordinazione	Successo	27	Nessuno
Eliminare un elemento	Successo	31	Nessuno
Effettuare un'ordinazione	Successo	36	Nessuno

Figura 22: Tabella di usabilità per donna anonima

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	53	Nessuno
Aggiungere un nuovo elemento	Successo	60	Ha cliccato su "Ordinazione"
Evadere un'ordinazione	Successo	80	Ha cliccato sul pulsante "Ordinazioni"
Eliminare un elemento	Successo	32	Nessuno
Effettuare un'ordinazione	Successo	39	Nessuno

Figura 23: Tabella di usabilità per Francesca

Task	Risultato	Tempo impiegato (in secondi)	Errori commessi
Aggiungere un nuovo utente	Successo	18	Nessuno
Aggiungere un nuovo elemento	Successo	24	Nessuno
Evadere un'ordinazione	Successo	31	Nessuno
Eliminare un elemento	Successo	13	Nessuno
Effettuare un'ordinazione	Successo	24	Nessuno

Figura 24: Tabella di usabilità per Giacomo

	Durante l'aggiunta di un elemento ha cliccato su "Cucina"	Durante l'evasione di un ordine ha cliccato su "Ordinazioni"	Durante l'aggiunta di un elemento ha cliccato su "Ordinazioni"	Numeri errori totali per utente
Marco	X			1
Nicola		X		1
Anonima	X			1
Francesca		X	X	2
Giacomo				0
Numero errori totali per task	2	2	1	

Figura 25: Valutazione euristica

I risultati dei test sono migliorati rispetto a quelli sui prototipi e gli errori più comuni sono stati commessi da 2 tester su 5, quindi si ritiene il risultato

soddisfacente. Tra le metriche per valutare l'usabilità vi sono, come detto precedentemente, anche l'apprendibilità e la memorabilità dell'uso dell'applicativo. Per raggiungere un elevato livello di queste due caratteristiche è necessario rendere l'applicazione il più semplice possibile, intuitiva e che non affatichi molto la memoria degli utenti chiedendogli di tenere a mente troppe informazioni mentre esegue qualche compito. A tal scopo sono state prese decisioni mirate che sono riportate di seguito:

1. Sono stati ingranditi diversi pulsanti per essere più facilmente visibili e raggiungibili, in accordo con la legge di Fitts, permettendo così all'utente di eseguire determinati task, che richiedono la pressione dei tasti ripetutamente (come ad esempio la presa di un ordine da parte di un addetto alla sala), più velocemente in quanto il tempo medio di pressione è stato ridotto
2. sono stati scelti colori delle icone per rendere tutto più leggibile, distinguibile e cercare di indurre soddisfazione nell'utente

Per rendere l'app più semplice abbiamo ridotto al minimo il numero di passaggi che l'utente deve compiere per eseguire un determinato compito, in modo che possa eseguirli velocemente ma anche che non sia difficile ricordarli e impararli. Un esempio di come abbiamo semplificato l'app può essere trovato nella schermata dell'aggiunta di un nuovo piatto, anche se le informazioni del form riportate nella schermate non sono poche, abbiamo preferito non spezzarle in schermate differenti in modo che un eventuale dubbio da parte dell'utente su un eventuale campo inserito precedentemente, non lo costringesse a tornare indietro ma potesse semplicemente controllare rapidamente con un'occhiata, così come nell'autocompletamento dell'inserimento degli allergeni che supporta l'utente nel ricordarli. Per rendere bene l'idea della rapidità con cui l'utente può effettuare i task è riportata di seguito lo hierarchical tree dell'applicazione con radice nella pagina iniziale di un amministratore.

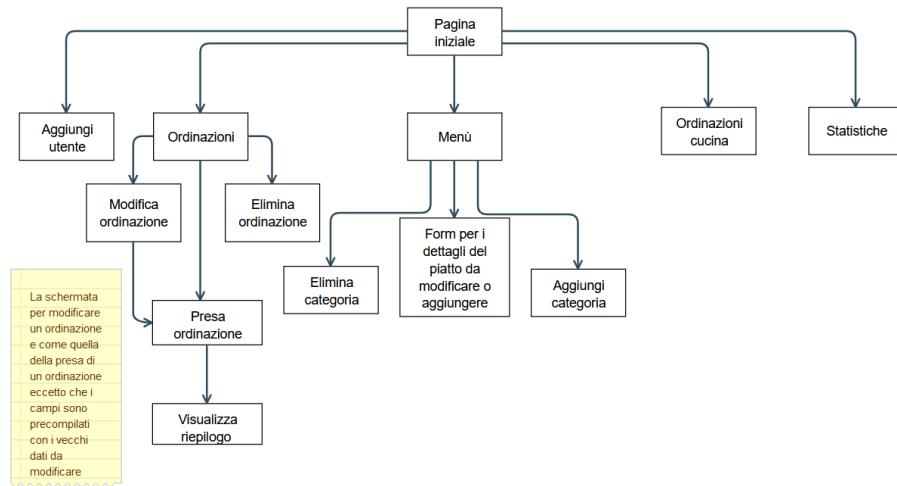


Figura 26: Hierarchical Tree

Come si può vedere ciascun compito è realizzabile in massimo 6 passaggi, garantendo velocità di esecuzione, apprendimento e memorabilità, quest'ultimo anche nel caso in cui l'utente avesse appena ricevuto le istruzioni su come eseguire un compito in quanto i chunk da tenere in mente sono meno di 7.

17 Logging

Al fine di poter monitorare e studiare il comportamento del sistema, le sue prestazioni e l'uso fatto dagli utenti, si sono raccolti e valutati diversi dati di log. Per farlo l'applicazione è stata distribuita ad un ristoratore che l'ha utilizzata come prova, permettendoci di raccogliere i dati di log forniti da AWS CloudWatch e da Google Analytics.

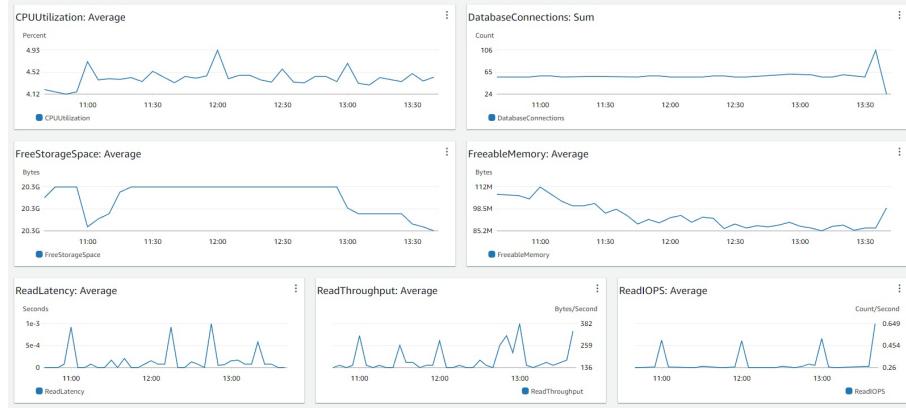


Figura 27: Metrics sul database 1

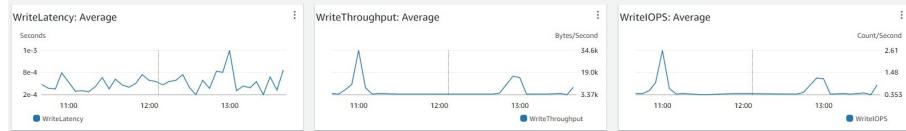


Figura 28: Metrics sul database 2

Come mostrato nei grafici sopra il tempo di latenza media (cioè il tempo che intercorre tra l'input e la ricezione dell'output conseguente) per operazioni di lettura e scrittura sul database sono entrambe di 3 ms nel peggio dei casi, le prestazioni sono, quindi, molto buone infatti si stima che il sistema riesca a rispondere a circa 333 richieste al secondo. Possiamo inoltre vedere come il throughput medio e l'IOPS (cioè il numero di celle non contigue lette) sono rispettivamente 382 byte/secondo e 0.649/secondo per la lettura, 34.6 byte/secondo e 2.61/secondo per la scrittura, questo indica che l'uso che gli utenti fanno dell'applicazione non è molto intenso, quindi il sistema non è mai eccessivamente stressato ed anche per questo si ritengono i risultati riguardo alla latenza ottimi. La bassa intensità dell'uso da parte degli utenti si rispecchia anche nell'uso della CPU che non supera mai il 4.93%, questo basso utilizzo delle risorse permette inoltre di utilizzare i servizi cloud a costi ridotti, permettendo quindi al committente di incrementare i ricavi. Anche l'utilizzo della CPU del server che si tiene su un massimo del 25% (figura 29) quindi ben al di sotto del 40% suggerito dal fornitore del servizio cloud utilizzato, AWS. Più usata è invece la memoria RAM, che arriva a picchi dell'82%.

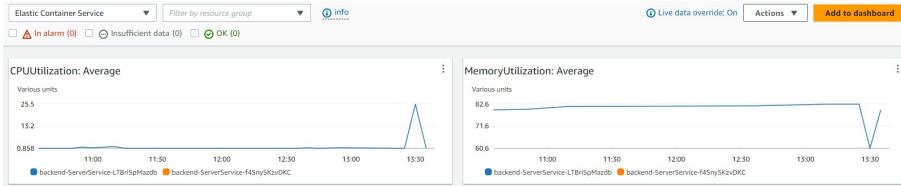


Figura 29: Metrics sul server

Di seguito sono riportati i grafici riguardo ai dati sull'interazione con la rete.

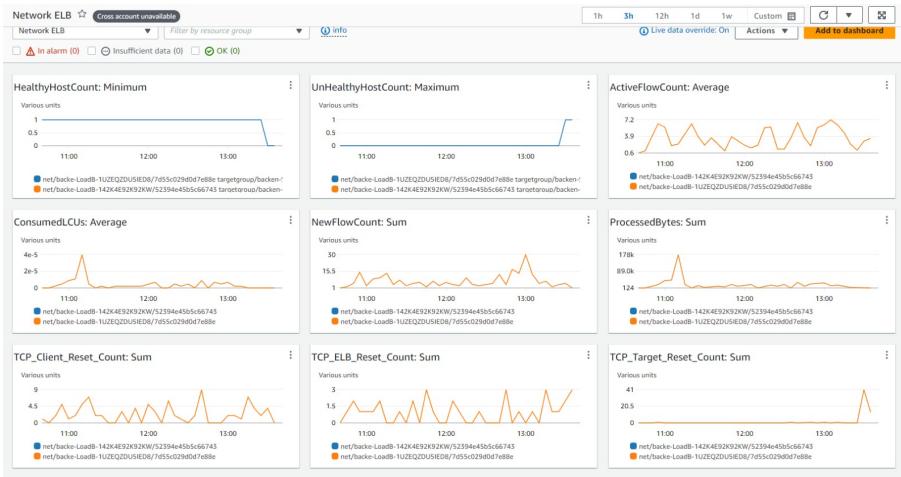


Figura 30: Metrics sul Network Load Balancer

Le statistiche sono di vario tipo, come per esempio:

- ActiveFlowCount Average, che è la media del numero di connessioni correnti al target
- NewFlowCount Sum, che è la somma delle nuove connessioni al target in un determinato periodo di tempo
- ProcessedBytes Sum, che è il totale del numero di bytes processati in un determinato periodo di tempo

I grafici seguenti mostrano invece alcuni dati sull'uso dell'applicazione da parte degli utenti.

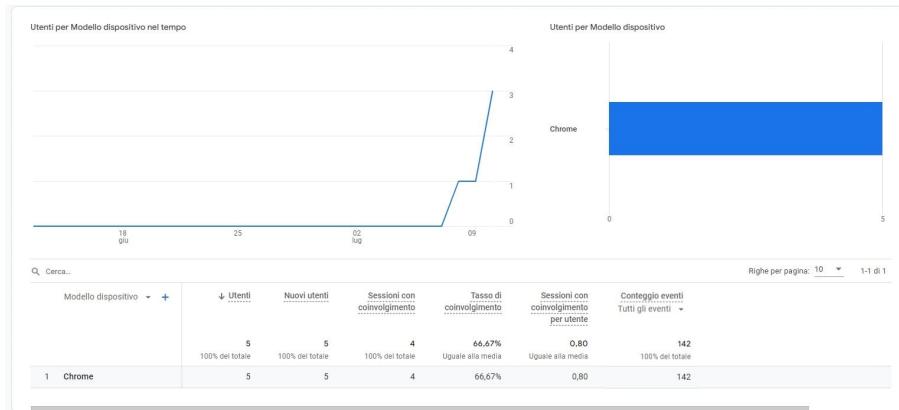


Figura 31: Analytics di Google sugli utenti 1

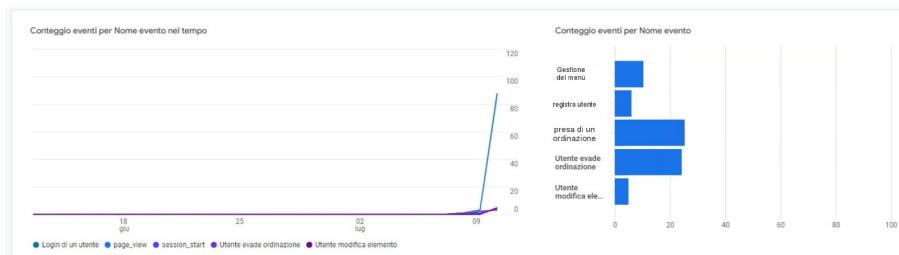


Figura 32: Analytics di Google sugli utenti 2

Come si può vedere gli utenti che hanno usato l'applicazione lo hanno fatto tutti tramite l'utilizzo del browser chrome, ed il tasso di coinvolgimento (cioè la percentuale di quante sessioni siano durate più di 10 secondi) è stato del 66.67% che hanno fatto registrare un totale di 142 eventi. L'uso principale che ne hanno fatto gli utenti era come prevedibile la presa e l'evasione degli ordini.