UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

CORSO DI LAUREA IN INFORMATICA INSEGNAMENTO DI INGEGNERIA DEL SOFTWARE ANNO ACCADEMICO 2022/2023

Specifica, progettazione, implementazione e validazione del Sistema Informativo "Ratatouille"

Autori
Mario De Luca N86/3911
Alessandro Bonomo N86/3852

Indice

Abstract

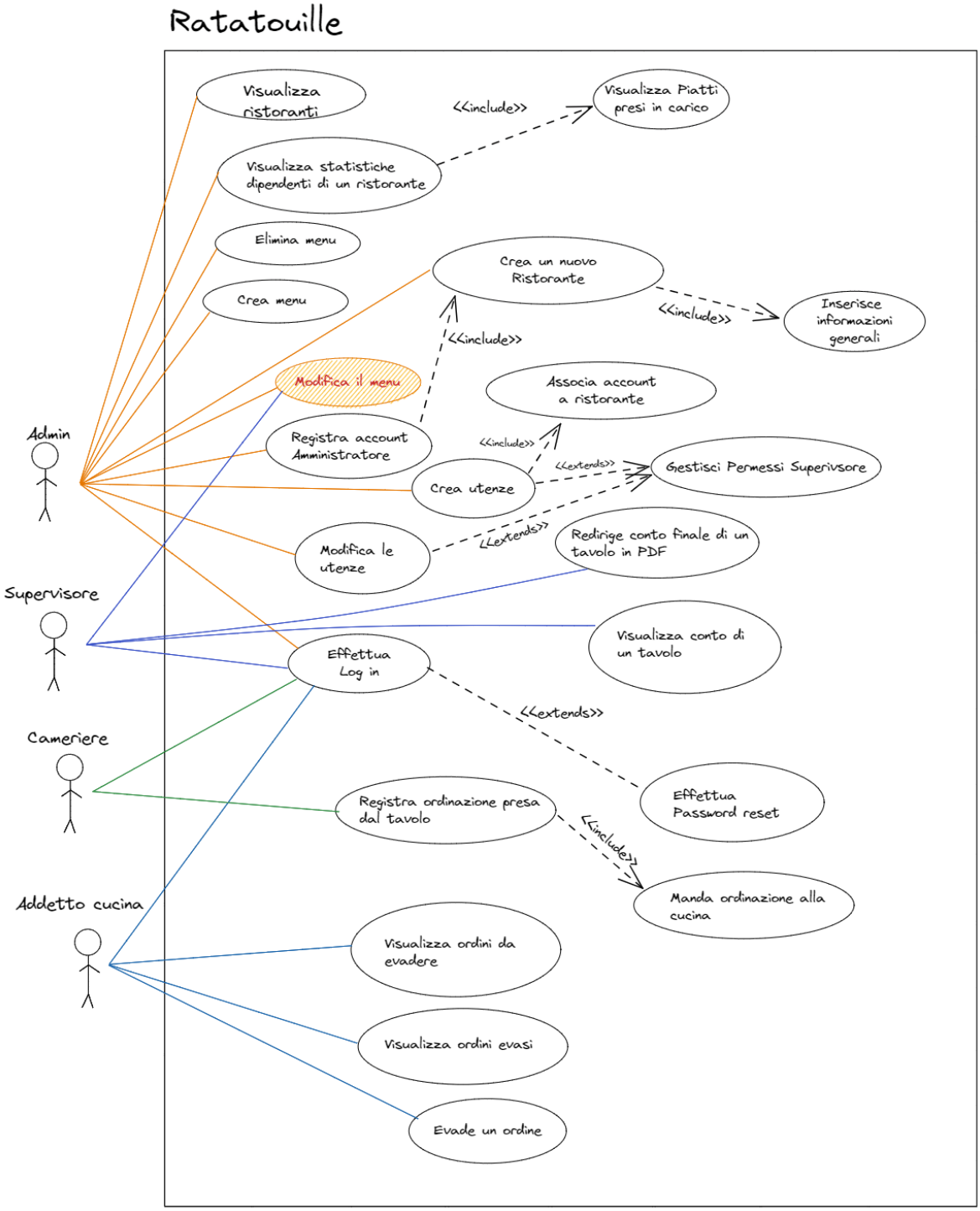
Descrizione del progetto

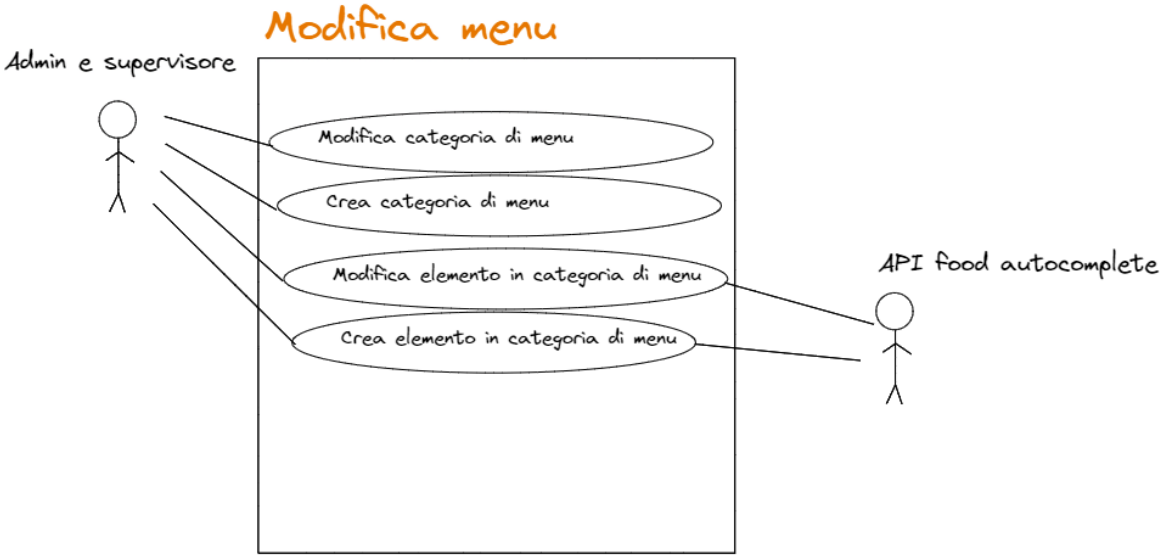
I Documento dei Requisiti Software

1 Analisi dei requisiti

1.1 Modellazione dei casi d'uso richiesti

Ecco il diagramma dei casi d'uso:

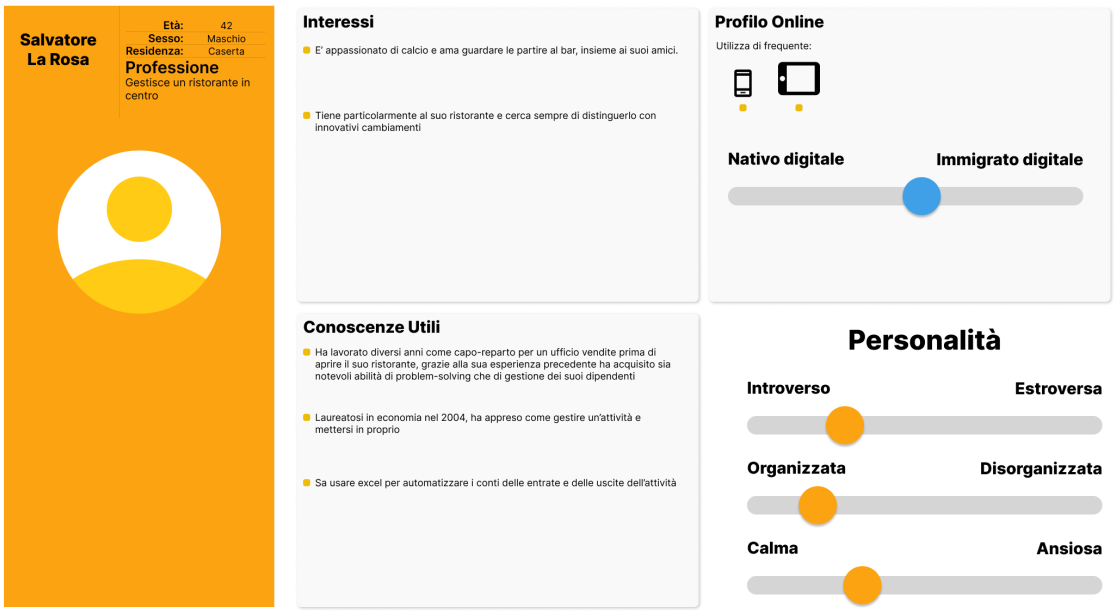




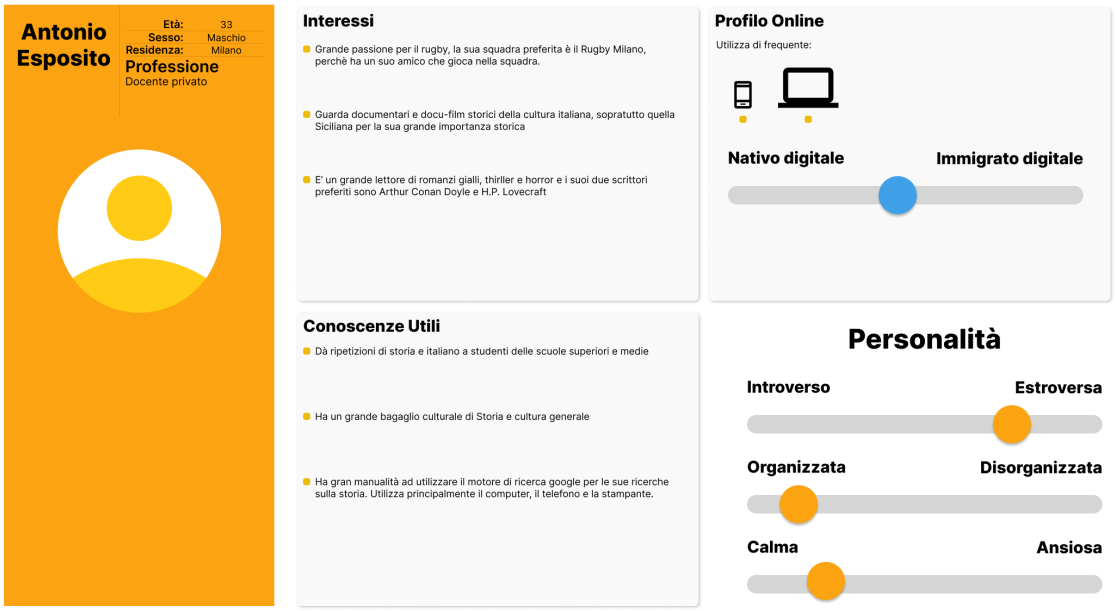
1.2 Individuazione del target degli utenti

La nostra applicazione è stata creata su misura per tutti gli utenti che lavorano nel ambiente della ristorazione. Facilita la gestione del ristorante all'amministratore, aiuta i camerieri a prendere le ordinazioni in maniera più agevole e ordinata. Infine, facilita le interazioni tra gli addetti alla cucina e il personale di sala. Abbiamo individuato 4 categorie di utenti:

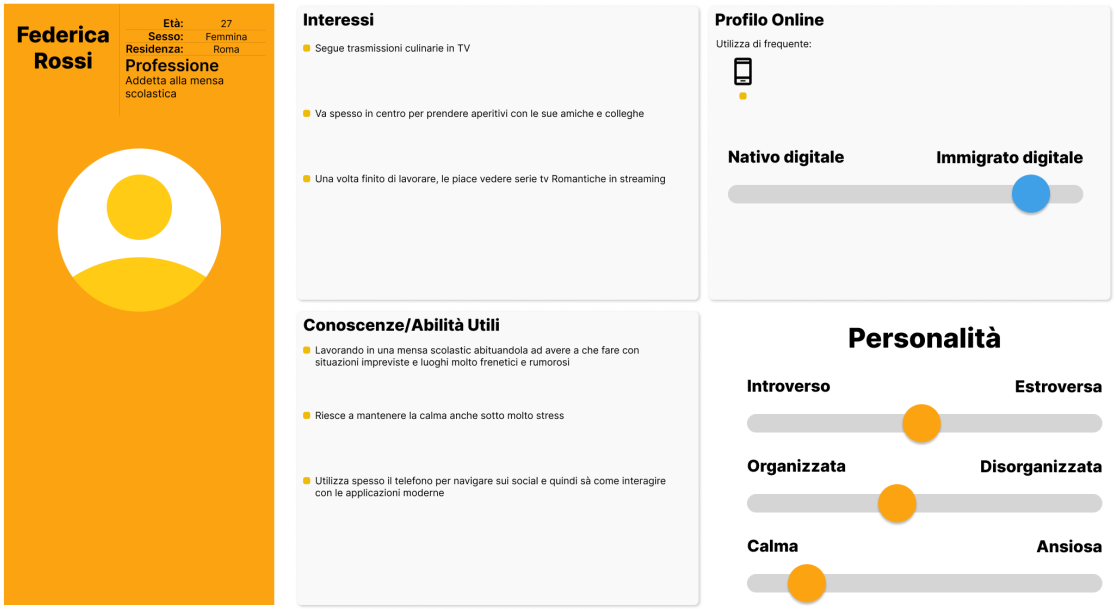
Amministratore Colui che gestisce i ristoranti e i suoi dipendenti, può gestire il menù di ogni singolo ristorante e avere un report sulle vendite di ogni locale.



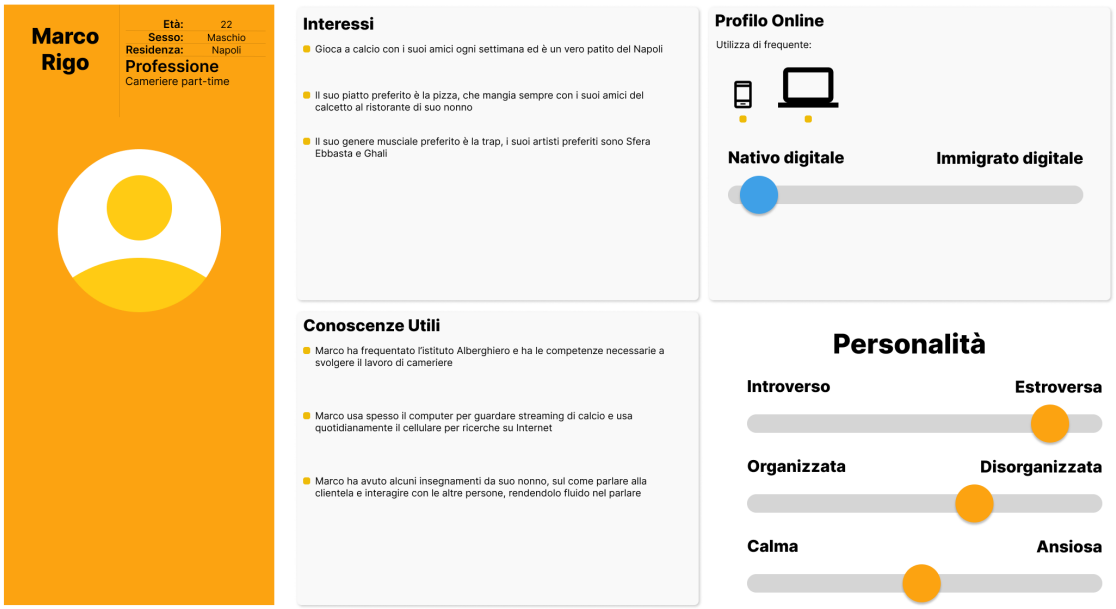
Supervisore E' un addetto alla cucina/sala, scelto dall'amministratore per supervisionare i suoi colleghi. Il supervisore può modficare il menu e gestire i conti dei tavoli.



Addetto alla cucina E' la persona che si occupa di leggere ed evadere gli ordini ricevuti dal cameriere.

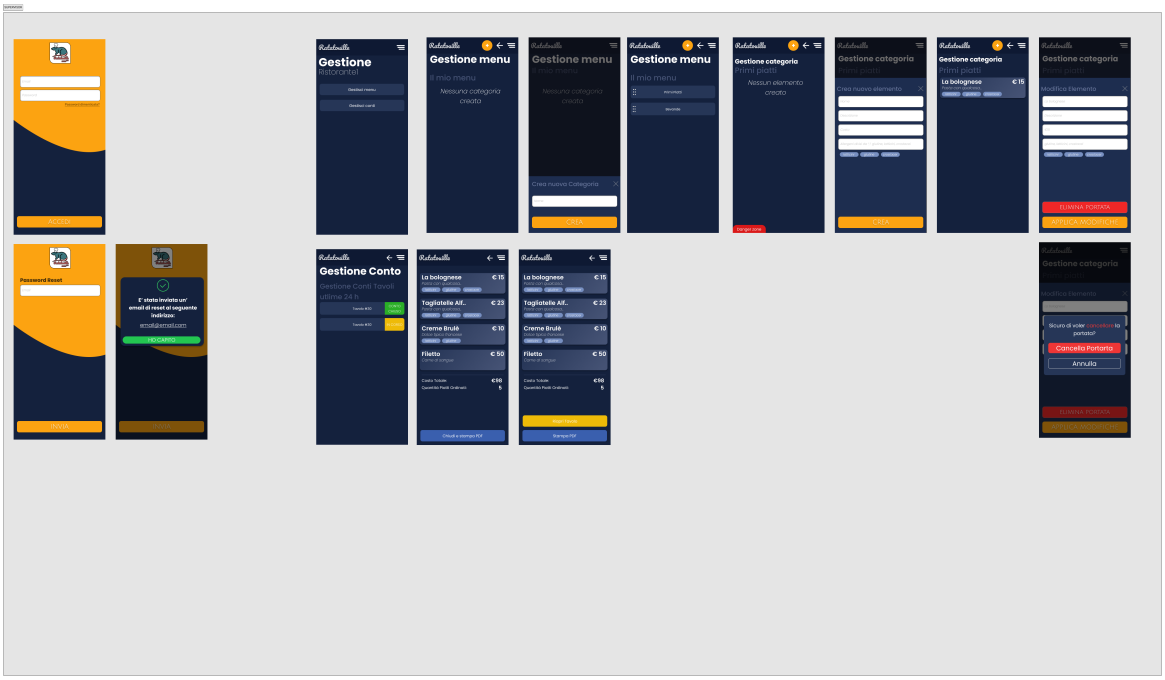


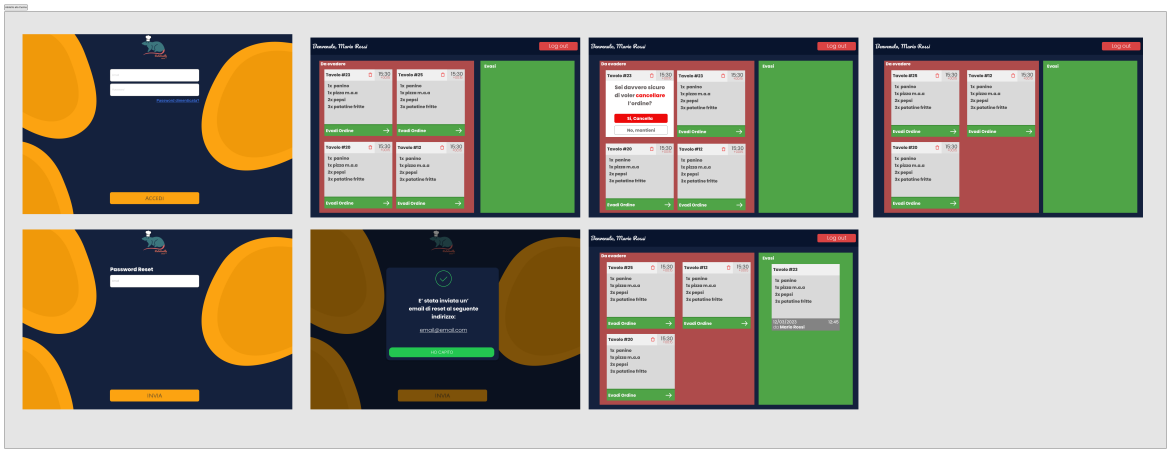
Addetto alla sala (Cameriere) E' colui che prende le ordinazioni e le invia alla cucina.



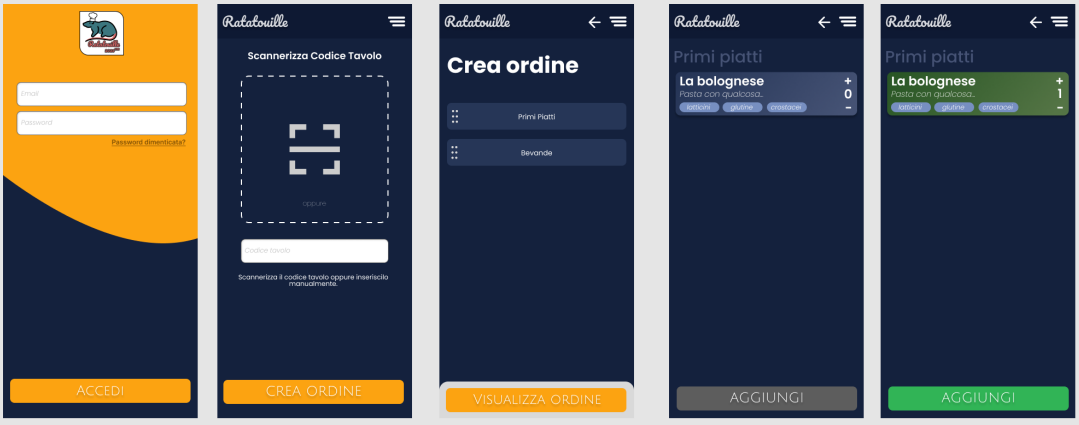
1.3 Prototipazione visuale via Mock-up dell'interfaccia utente per tutti i casi d'uso

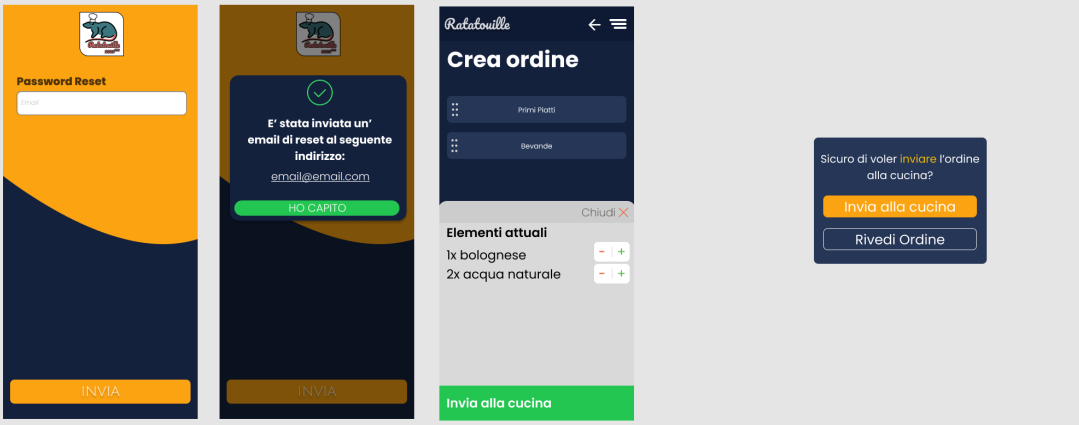
Riportiamo di seguito tutti i mock-up dell'applicazione. I mock up sono stati realizzati con Figma. E' possibile visualizzare i mock up in dettaglio cliccando qui (Link al progetto Figma in cloud).





Cameriere


Email
Password
Password dimenticata?
ACCEDI
Ratatouille
Scannerizza Codice Tavolo
oppure
Codice tavolo
Scannerizza il codice tavolo oppure inseriscilo manualmente.
CREA ORDINE
Crea ordine
Primi Piatti
Bevande
VISUALIZZA ORDINE
Primi piatti
La bolognese
Pasta con qualcosa..
0
AGGIUNGI
1


Password Reset
Email
INVIA
E' stata inviata un' email di reset al seguente indirizzo:
email@email.com
HO CAPITO
Crea ordine
Primi Piatti
Bevande
Chiudi
Elementi attuali
1x bolognese
2x acqua naturale
Invia alla cucina
Sicuro di voler inviare l'ordine alla cucina?
Invia alla cucina
Rivedi Ordine

1.4 Tabelle di Cockburn

Di seguito riportiamo le tabelle di cockburn associate ai relativi casi d'uso:

- Crea nuovo ristorante
- Crea nuova categoria

1.4.1 Crea nuovo ristorante

Use Case #1	Crea nuovo ristorante		
Goal in Context	Creazione di un nuovo ristorante con le relative informazioni.		
Preconditions	L'utente deve essere autenticato come amministratore e trovarsi nella schermata Dashboard admin.		
Success End Conditions	Il ristorante viene aggiunto al sistema e viene mostrato nell'elenco dei ristoranti registrati.		
Failed End Conditions	L utente clicca sull'icona di annullamento e ritorna alla schermata Dashboard admin.		
Primary Actor	Utente amministratore autenticato		
Trigger	L'utente clicca sul pulsante '+' nella schermata Dashboard admin		
Description	Step	User Action	System
	1	Clicca sul pulsante '+' nella schermata Dashboard admin	
	2		Mostra schermata Crea Ristorante
	3	Inserisce nome ristorante	
	4	Inserisce locazione ristorante	
	5	Inserisce numero di telefono ristorante	
	6	Clicca crea	
	7		Torna a Dashboard admin

Extension #1	Step	User Action	System
	1.1	Clicca sulla 'X'	
	1.2		Ritorna alla schermata Dashboard admin
Extension #2	Step	User Action	System
	6.1	Clicca su 'CREA' senza aver inserito il nome del ristorante	
	6.2	Mostra messaggio di errore	
Extension #3	Step	User Action	System
	6.1	Clicca su 'CREA' senza aver inserito la locazione del ristorante	
	6.2	Mostra messaggio di errore	
Extension #4	Step	User Action	System
	6.1	Clicca su 'CREA' senza aver inserito il numero di telefono del ristorante	
	6.2	Mostra messaggio di errore	
Subvariation #1	Step	User Action	System
	3.1	Inserisce l'url del sito web	
	3.2		Vai al punto 4
Subvariation #2	Step	User Action	System
	5.1	Seleziona immagine ristorante	
	5.2		Apre il browser del filesystem
	5.3	Seleziona l'immagine desiderata e conferma	
	5.4		Mostra preview immagine
	5.4		Vai al punto 6
Notes			

1.4.2 Crea nuova categoria

Use Case #2	Crea nuova categoria		
Goal in Context	Creazione di una nuova categoria del menu.		
Preconditions	L'utente deve essere autenticato come amministratore o supervisore e trovarsi nella schermata Gestione Menu.		
Success End Conditions	La categoria viene aggiunta al sistema e viene mostrata nel menu nella schermata Gestione Menu.		
Failed End Conditions	L utente clicca sull'icona di annullamento e ritorna alla schermata Gestione Menu.		
Primary Actor	Utente amministratore o supervisore autenticato		
Trigger	L'utente clicca sul pulsante '+' nella schermata Gestione Menu		
Description	Step	User Action	System
	1	Clicca sul pulsante '+' nella schermata Gestione Menu	
	2		Mostra schermata Crea categoria
	3	Inserisce nome della categoria	
	4	Clicca crea	
	5		Torna a schermata Gestione Menu
Extension #1	Step	User Action	System
	1.1	Clicca sulla 'X'	
	1.2		Ritorna alla schermata Gestione Menu
Extension #2	Step	User Action	System
	4.1	Clicca su 'CREA' senza aver inserito il nome della categoria	
	4.2		Mostra messaggio di errore
Notes			

1.5 Mock-up dei casi d'uso descritti nelle tabelle di Cockburn

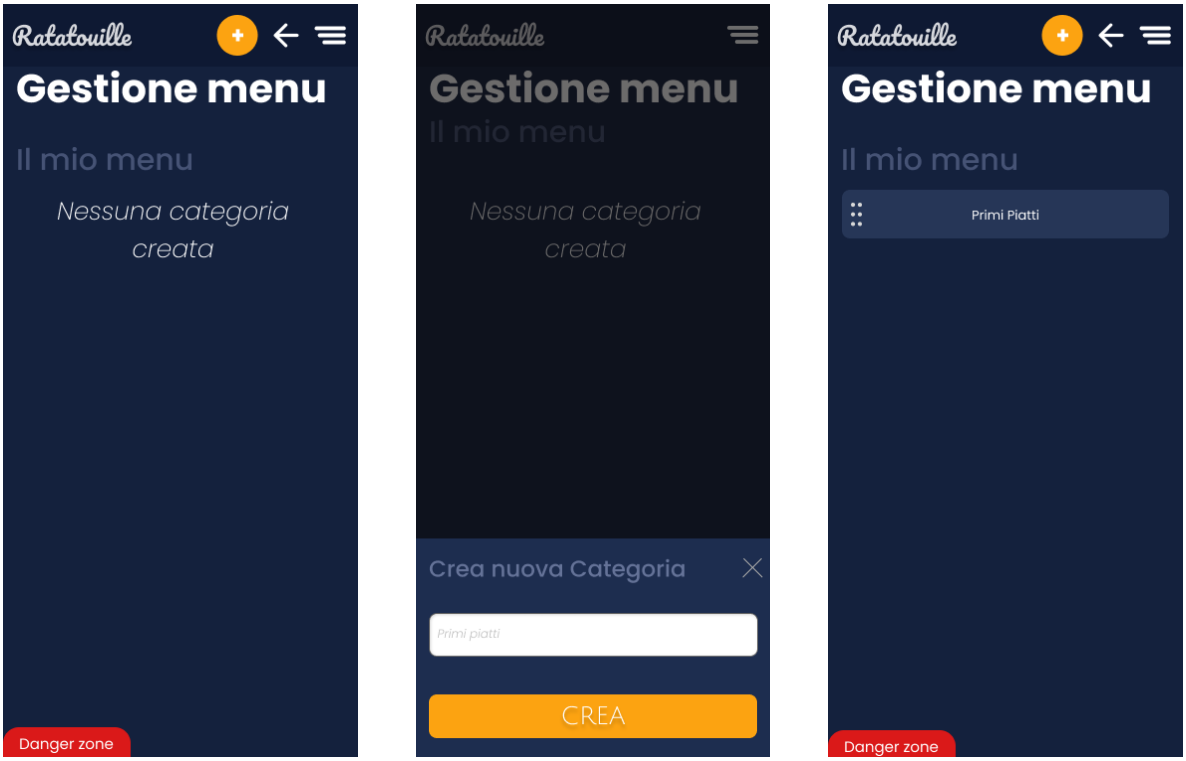


Figure 1: Mock-ups per Crea categoria

1.6 Valutazione dell'usabilità a priori

Prima di proseguire con lo sviluppo dell'applicazione, si è deciso di far effettuare ad un gruppo di utenti (tester) una valutazione del prototipo realizzato su Figma.

1.6.1 Testing del prototipo con Figma

Abbiamo dato ai tester una serie di Task da completare e abbiamo osservato le loro interazioni e difficoltà nello svolgere i compiti assegnati. Abbiamo poi compilato una tabella con gli esiti dei Task dati ai tester. Infine con una semplice formula siamo in grado di misurare la facilità di utilizzo della nostra applicazione.
Task analizzati:

1. Login
2. Registrazione
3. Crea ristorante
4. Crea ordine e invia alla cucina
5. Completa ed evadi ordine
6. Creazione di un menu con una portata
7. Stampa conto

1.6.2 Analisi dei risultati

	1	2	3	4	5	6	7
Tester 1	✔	✔	✔	✘	✔	🚚	✔
Tester 2	✔	✔	✔	🚚	✔	✔	✔
Tester 3	✔	✔	🚚	🚚	✔	✔	✔
Tester 4	✔	✔	✔	🚚	✔	✔	✔

Leggenda:
✔ Successo +1
✘ Fallimento -1
🚚 Successi Parziali +0.5

Ogni tester avrà il suo punteggio associato, che indica la sua facilità di esecuzione di quello specifico task.

$tasks = 7$
$testers = 4$
$punteggioMassimo = (testers * tasks) = 28$
$test_{i,n}$ = esito del test effettuato dal tester "i" nel task "n". Valori possibili: 0.5, 1 ,-1
$punteggioSingoloTester_i = \sum_{n=1}^{tasks} test_{i,n}$
$punteggioTotale = \sum_{i=1}^{testers} punteggioSingoloTester_i$

Facilità d' uso [1,-1] $= punteggioTotale/punteggioMassimo$

La facilità d'uso è un numero compreso tra 1 e -1. Se è negativa allora vuol dire che i fallimenti sono maggiori dei successi. Se è positiva, allora i successi sono maggiori dei fallimenti. Se è uguale a 0 allora i successi sono uguali ai fallimenti. Più il punteggio finale si avvicina ad 1, meglio è.

Il punteggio calcolato PRIMA delle correzioni è:
punteggioTotale $= 4.5 + 6.5 + 6 + 6.5$
Facilità d' uso $= 23.5/28 = 0.83$

Il punteggio calcolato DOPO le correzioni è:
punteggioTotale $= 6.5 + 6.5 + 7 + 6.5$
Facilità d' uso $= 26.5/28 = 0.94$

	1	2	3	4	5	6	7
Tester 1	✔	✔	🚚	✔	✔	✔	✔
Tester 2	✔	✔	✔	🚚	✔	✔	✔
Tester 3	✔	✔	✔	✔	✔	✔	✔
Tester 4	✔	✔	✔	✔	✔	🚚	✔

Figure 2: Testing dopo le correzioni

1.6.3 Correzioni in base al feedback dei tester

Grazie al feedback dei nostri testers siamo riusciti ad individuare 2 problematiche che impattavano sull'usabilità del prodotto.

Correzione logo Grazie al suggerimento di uno dei nostri tester abbiamo deciso di modificare leggermente il logo dell'applicazione per renderlo più affine al contesto culinario.

Figure 3: Correzione del logo. A sinistra la vecchia versione, a destra, la nuova

Correzione Task N°4 (Crea ordine) Una difficoltà comune a tutti i tester era riuscire a trovare l'interazione per riuscire a visualizzare e inviare l'ordine alla cucina. Abbiamo così deciso di modificare i Mock up per aumentare l'usabilità dell'applicazione, rendendo più visibile e intuitivo il pulsante per visualizzare l'ordine in corso.

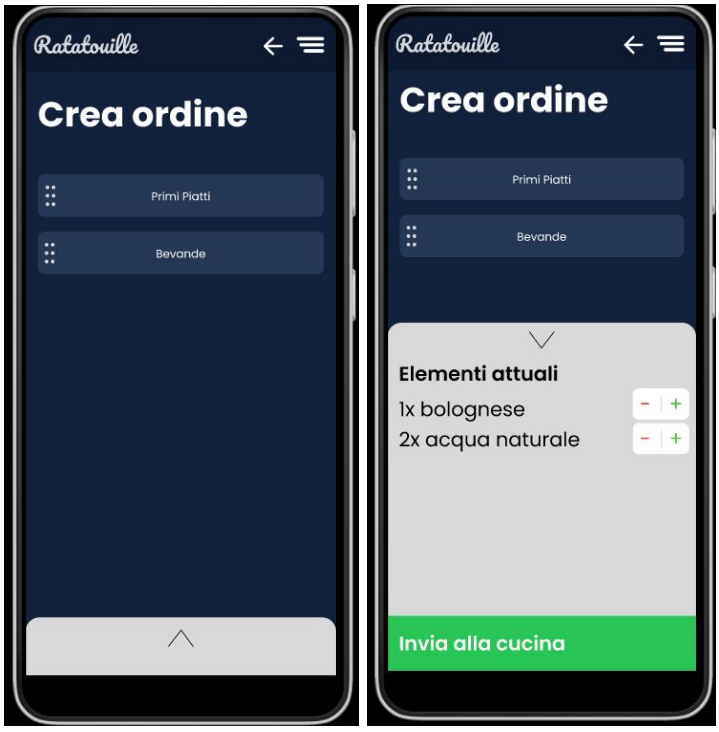


Figure 4: Prima della correzione

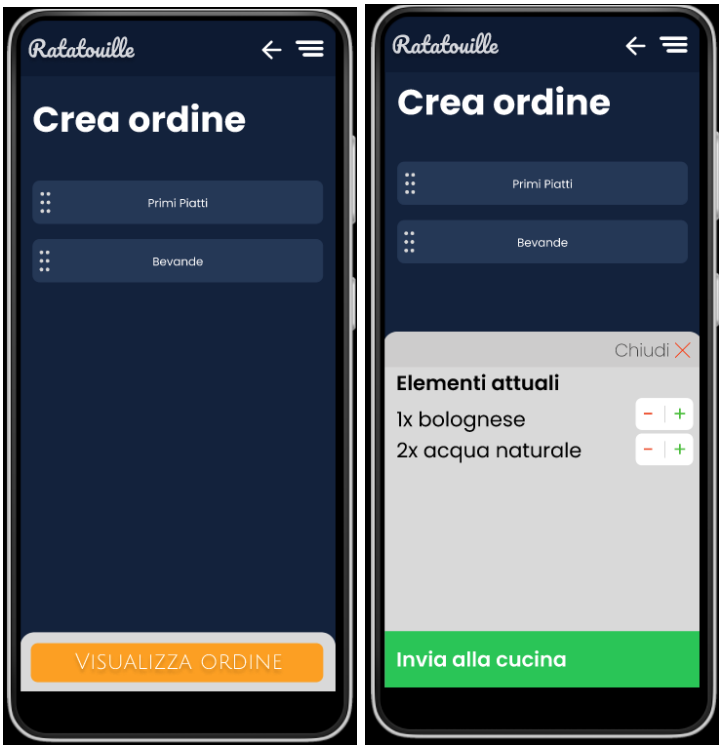


Figure 5: Dopo la correzione

1.7 Glossario

Raccolta dei termini utilizzati all'interno della documentazione:

Termine	Descrizione
Admin / Amministratore	Desc1
Supervisore	Desc1
Figma	Desc1
Immigrato digitale	Desc1

2 Specifica dei Requisiti

2.1 Classi, oggetti e relazioni di analisi

2.2 Diagrammi di sequenza di analisi

2.3 Prototipazione funzionale via statechart

II Documento di Design del Sistema

3 Analisi dell'architettura

3.1 Architettura 3 Tier

spiegazione di cos'è il 3 tier, spiegazione docker + immagine esplicativa

3.1.1 Client - Tier 1

per realizzare il client abbiamo usato react. E' molto leggero, dialoga con le api del backend e con api esterne per l'autocompletamento del testo.

3.1.2 Server - Tier 2

Abbiamo realizzato una restAPI in node js.(spiega restapi) Per garantire la sicurezza abbiamo usato un jwt (spiega jwt). Il server è un layer di controllo tra il client e il db. (spiega meglio)

Operazioni CRUD cosa sono le operazioni CRUD, quanti tipi di crud esistono (post,get,update...)

Prefisso routes Il prefisso di tutte le routes del backend è /**api**

Nome route	Descrizione	Tipo	Autorizzazione	Richiesta	Risposta
/	Messaggio di hello world del server	GET	No Auth	Nessun parametro	JSON message:string
/login	Se ha successo, restituisce il token in data	POST	No Auth	JSON username:string password:string	JSON success:bool data:string
/register	Registra un utente admin	POST	No Auth	JSON username:string password:string	JSON data:string
/resturant	Crea un ristorante	POST	Bearer Token, ADMIN	JSON nome:string indirizzo:string telefono:string [sitoWeb:string]	JSON success:bool data:string
/resturant/:id	Ottiene un ristorante dall'id	GET	Bearer Token	id	JSON success:bool data:string
/resturants	Ottiene i ristoranti dell'utente loggato	GET	Bearer Token	Nessun parametro	JSON success:bool data:string
/utente/:email	permette all'admin di modificare nome, cognome, telefono di un suo impiegato	PUT	Bearer Token, ADMIN	email + JSON nome:string cognome:string telefono:string	JSON success:bool data:string
/utente/:email	permette all'admin di eliminare un suo impiegato	DELETE	Bearer Token, ADMIN	email	JSON success:bool data:string
/pw-changed	Ritorna se la password è stata mai cambiata	GET	Bearer Token	Nessun parametro	Boolean
/pw-change	Permette di cambiare la password	POST	Bearer Token	JSON password:string	JSON success:bool data:string

Nome route	Descrizione	Tipo	Autorizzazione	Richiesta	Risposta
/utenza	Crea una nuova utenza	POST	Bearer Token, ADMIN	JSON nome:string cognome:string email:string password:string ruolo:string telefono:string supervisore:bool	JSON success:bool data:string
/categorie/: `id_ristorante`	Visualizza categorie del menu del ristorante	GET	Bearer Token	`id_ristorante`	*Categoria*[]
/categoria	Crea una nuova categoria	POST	Bearer Token, SUP	JSON nome:string `id_ristorante`	JSON success:bool data:string
/categoria/: `id_categoria`	Elminina categoria	DELETE	Bearer Token, SUP	`id_categoria`	JSON success:bool data:string
/categoria/: `id_categoria`	Modifica categoria	PUT	Bearer Token, SUP	`id_categoria`	JSON success:bool data:string
/utenti/: `id_ristorante`	Visualizza utenze del ristorante	GET	Bearer Token, ADMIN	`id_ristorante`	*Utente*[]
/elementi/: `id_categoria`	Visualizza elementi del menu della categoria	GET	Bearer Token	`id_categoria`	*Elemento*[]
/elemento/: `id_elemento`	Visualizza elemento	GET	Bearer Token	`id_elemento`	JSON success:bool data:string
/elemento/: `id_elemento`	Cancella elemento	DELETE	Bearer Token, SUP	`id_elemento`	JSON success:bool data:string
/elemento/: `id_elemento`	Modifica elemento	PUT	Bearer Token, SUP	`id_elemento`	JSON success:bool data:string
/elemento	Crea elemento	POST	Bearer Token, SUP	JSON nome:string `id_categoria` prezzo:string descrizione:string ingredienti: str list	JSON success:bool data:string

3.1.3 Database - Tier 3

immagine di Postgres db con file di inizializzazione, check constraint sulle tabelle, constraints di chiave esterna, chiavi primarie, validazione su db (spiega meglio)

3.2 Cloud Hosting

spiegazione del cloud hosting che abbiamo usato

4 Motivazione delle scelte adottate

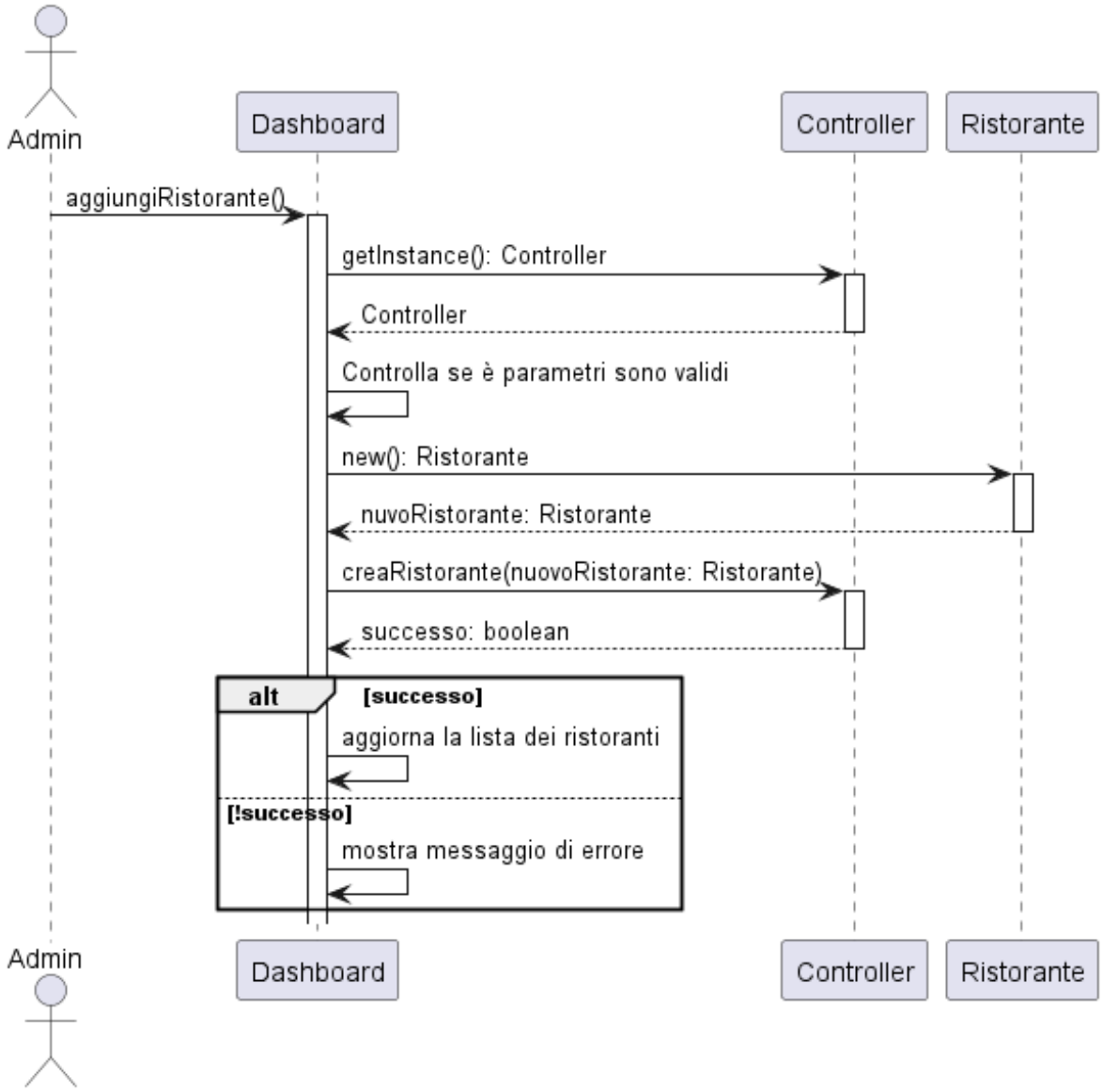
(spiega meglio) abbiamo usato la 3 tier perchè organizza meglio il progetto e lo rende più versatile, abbiamo scelto il cloud x perchè ci fornisce tot accessi al mese, costa tot, banda alta, spazio a sufficienza, abbiamo scelto di usare docker per avere un'applicazione robusta in modo che i servizi vengano avviati in un certo ordine, inoltre abbiamo il controllo delle porte esposte di docker.

5 Diagramma delle classi di design

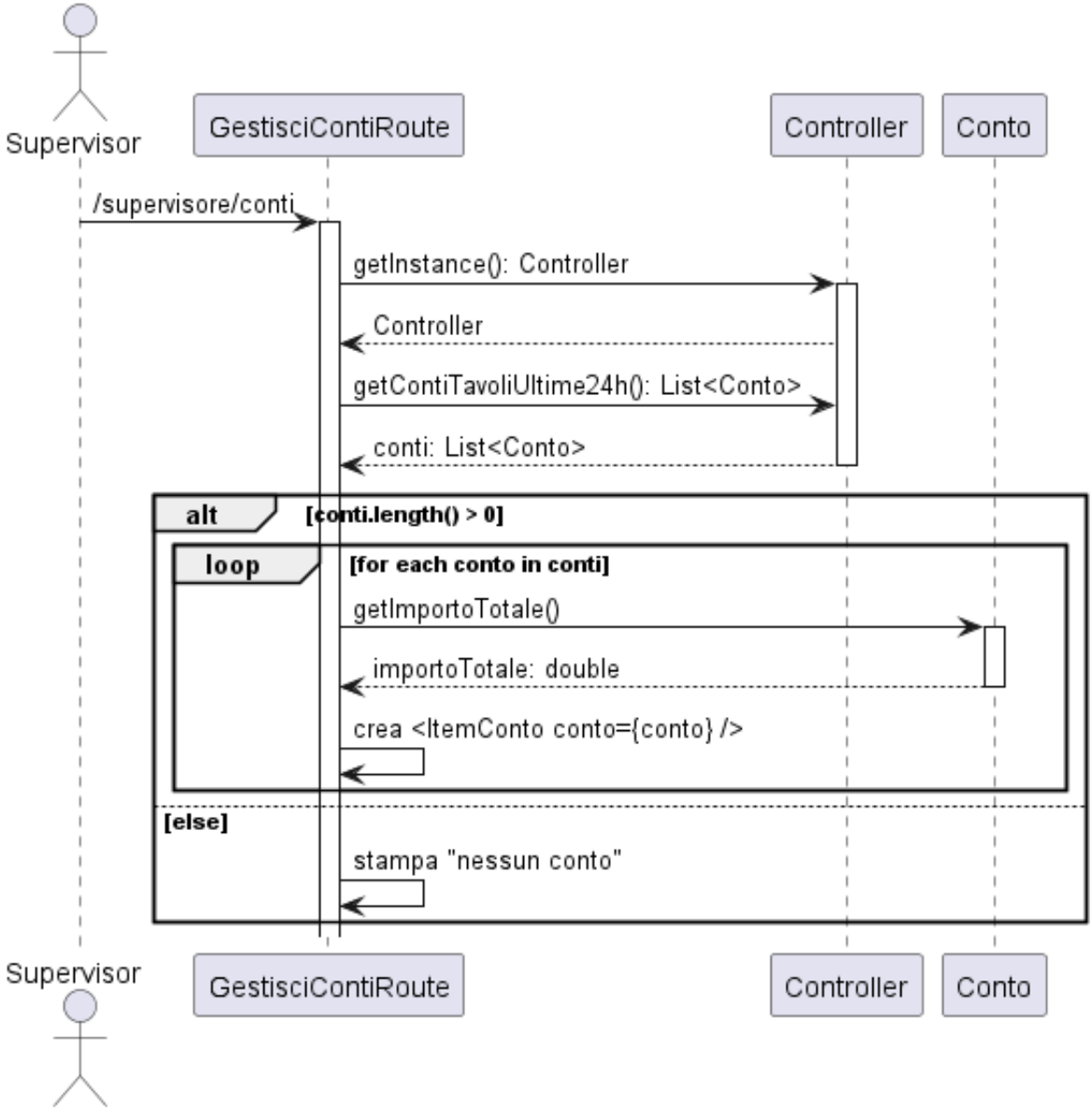
classi del frontend

6 Diagrammi di sequenza di design

6.1 aggiungiRistorante



6.2 getContiUltime24h



III Testing e valutazione sul campo dell'usabilità

7 Codice xUnit per unit testing

8 Valutazione dell'usabilità sul campo