

# 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

<b>I Documento dei Requisiti Software</b>	<b>2</b>
<b>1 Analisi dei requisiti</b>	<b>2</b>
1.1 Modellazione dei casi d'uso richiesti . . . . .	2
1.2 Individuazione del target degli utenti . . . . .	3
1.3 Prototipazione visuale via Mock-up dell'interfaccia utente per tutti i casi d'uso . . . . .	6
1.4 Tabelle di Cockburn . . . . .	8
1.4.1 Crea nuovo ristorante . . . . .	8
1.4.2 Crea nuova categoria . . . . .	10
1.5 Mock-up dei casi d'uso descritti nelle tabelle di Cockburn . . . . .	11
1.6 Valutazione dell'usabilità a priori . . . . .	12
1.6.1 Testing del prototipo con Figma . . . . .	12
1.6.2 Analisi dei risultati . . . . .	13
1.6.3 Correzioni in base al feedback dei tester . . . . .	14
1.7 Glossario . . . . .	16
<b>2 Specifica dei Requisiti</b>	<b>17</b>
2.1 Classi, oggetti e relazioni di analisi . . . . .	17
2.2 Diagrammi di sequenza di analisi . . . . .	17
2.3 Prototipazione funzionale via statechart . . . . .	17
2.3.1 Statechart - Crea un Ristorante . . . . .	17
2.3.2 Statechart - Crea una Categoria . . . . .	19
<b>II Documento di Design del Sistema</b>	<b>19</b>
<b>3 Analisi dell'architettura</b>	<b>19</b>
3.1 Architettura 3 Tier . . . . .	19
3.1.1 Client - Tier 1 . . . . .	19
3.1.2 Server - Tier 2 . . . . .	20
3.1.3 Database - Tier 3 . . . . .	20
3.1.4 Schema architettura . . . . .	20
3.1.5 Docker containers in azione . . . . .	20
3.2 Routes del Backend . . . . .	20
3.3 Cloud Hosting . . . . .	24
3.4 Pannello di controllo in cloud . . . . .	24
<b>4 Motivazione delle scelte adottate</b>	<b>24</b>
<b>5 Diagramma delle classi di design</b>	<b>24</b>
<b>6 Diagrammi di sequenza di design</b>	<b>25</b>
6.1 aggiungiRistorante . . . . .	25
6.2 getContiUltime24h . . . . .	26
<b>III Testing e valutazione sul campo dell'usabilità</b>	<b>26</b>
<b>7 Codice xUnit per unit testing</b>	<b>26</b>
<b>8 Valutazione dell'usabilità sul campo</b>	<b>26</b>

## Abstract

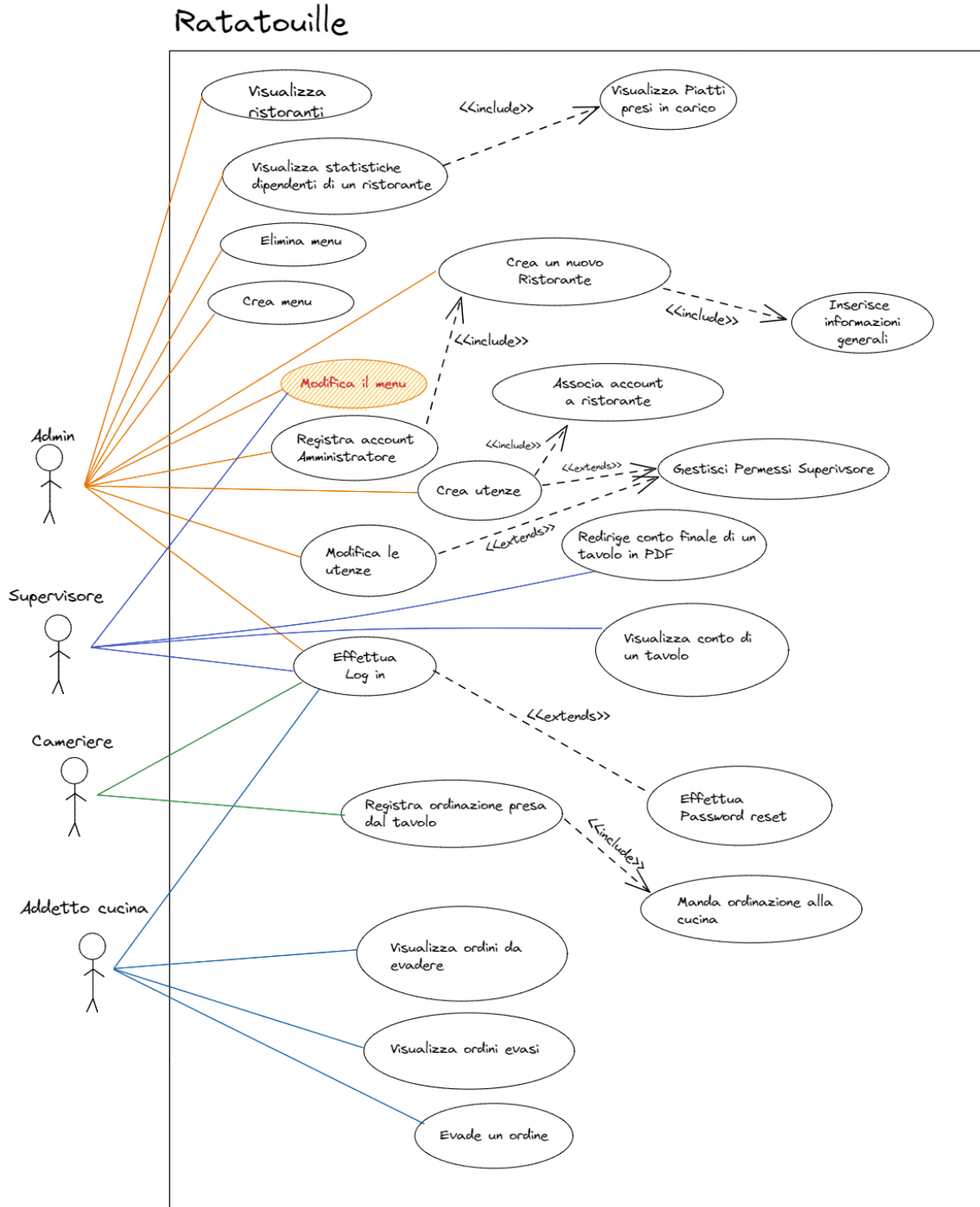
Descrizione del progetto

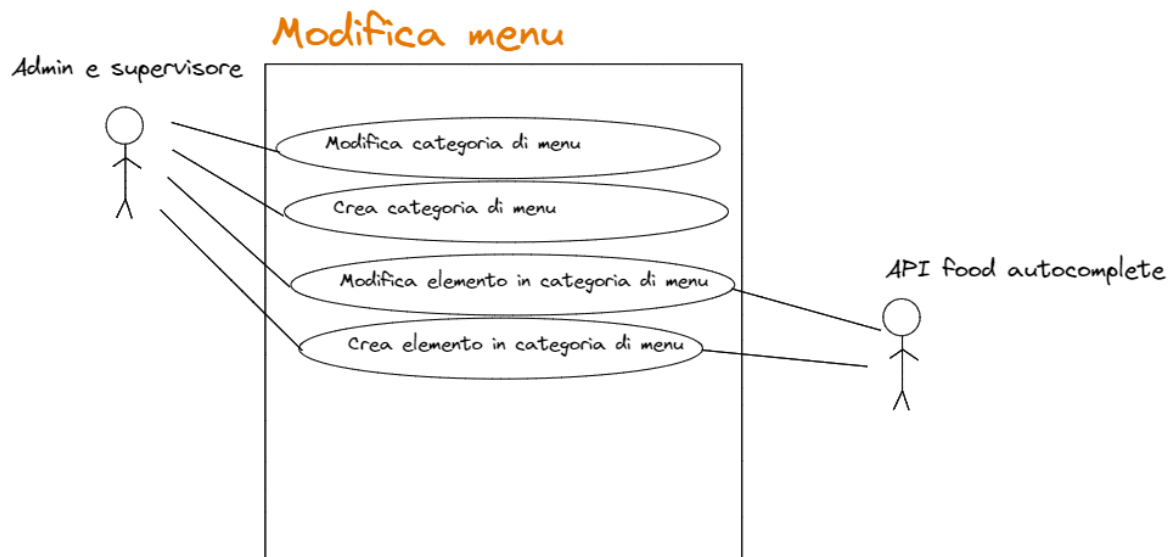
# I Documento dei Requisiti Software

## 1 Analisi dei requisiti

### 1.1 Modellazione dei casi d'uso richiesti

Ecco il diagramma dei casi d'uso:

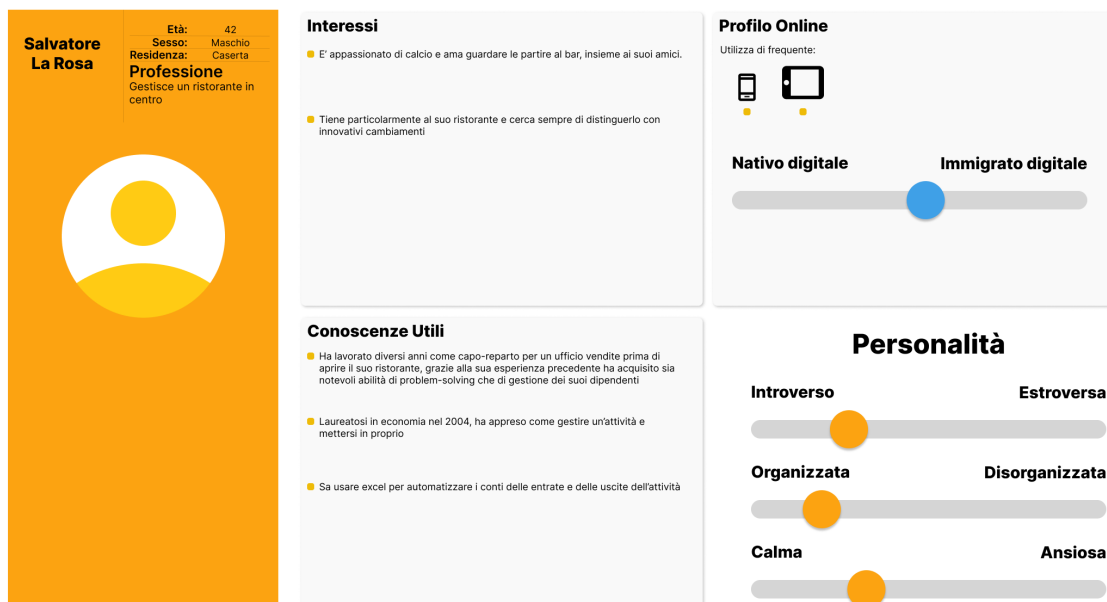




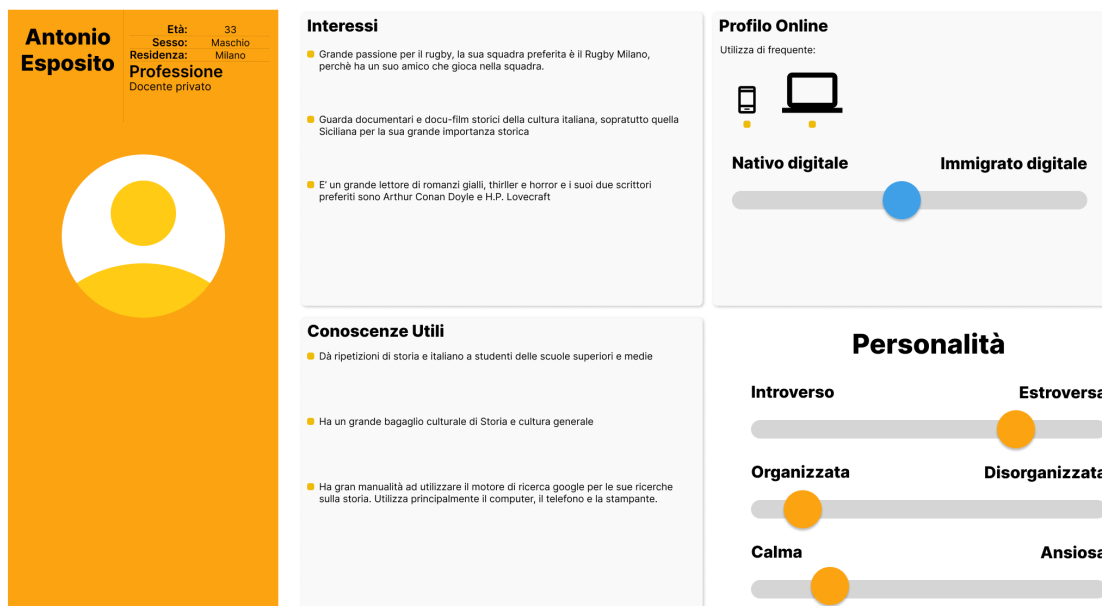
## 1.2 Individuazione del target degli utenti

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:

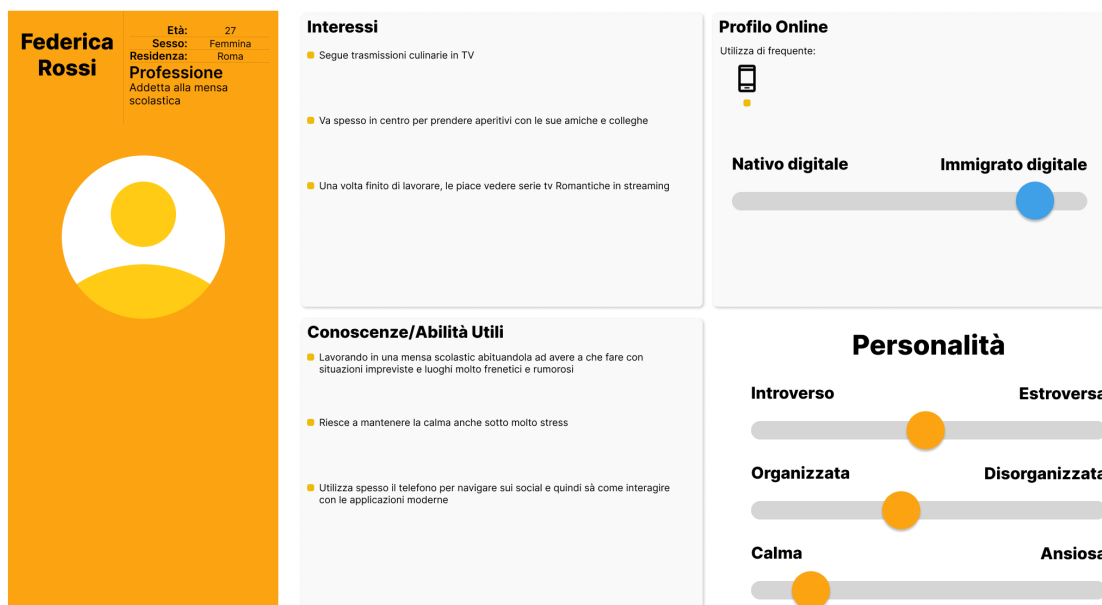
**Amministratore** Colui che gestisce i ristoranti e i suoi dipendenti, può gestire il menù di ogni singolo ristorante e avere un report sulle vendite di ogni locale.



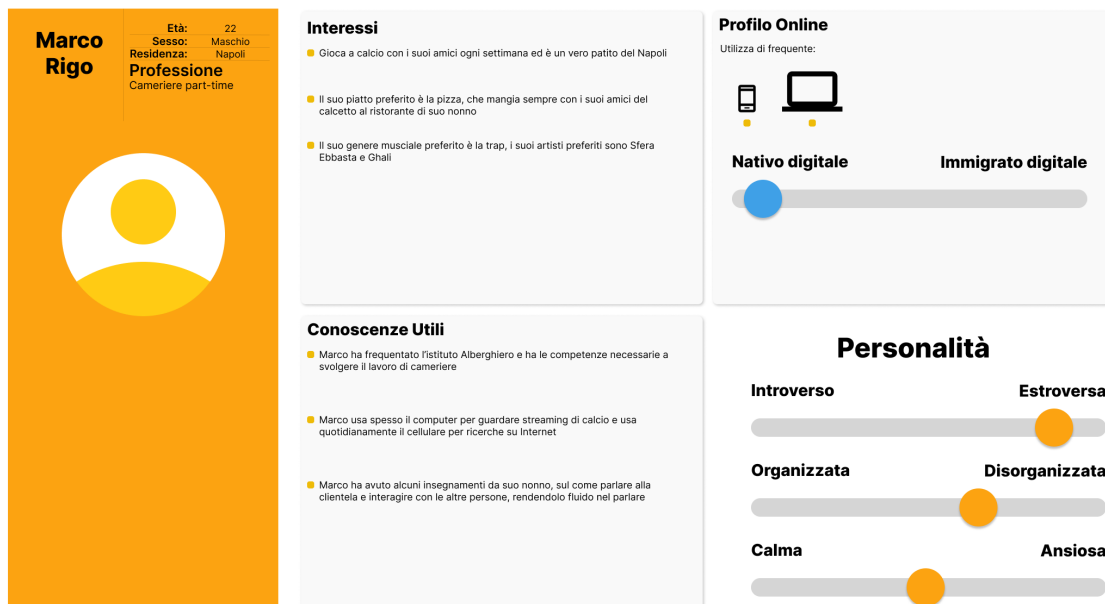
**Supervisore** E' un addetto alla cucina/sala, scelto dall'amministratore per supervisionare i suoi colleghi. Il supervisore può modificare il menu e gestire i conti dei tavoli.



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



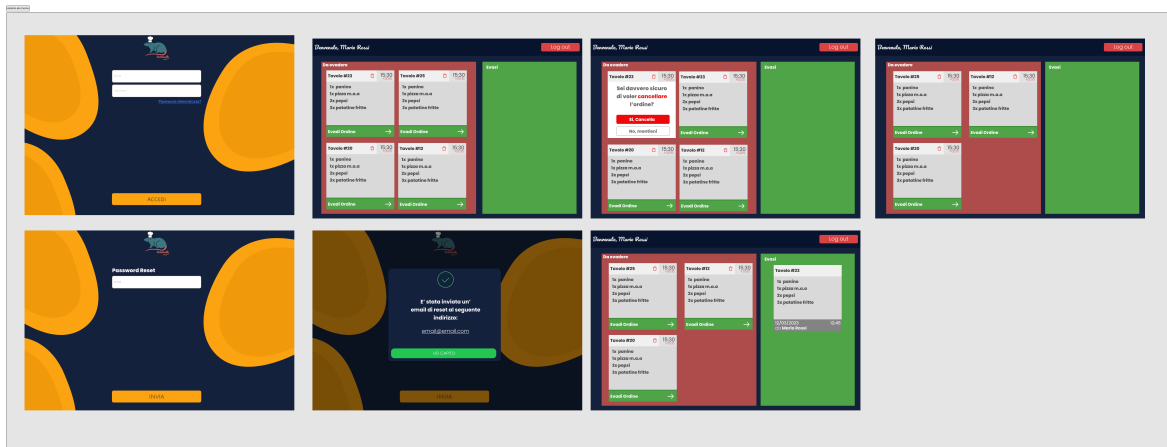
**Addetto alla sala (Cameriere)** E' colui che prende le ordinazioni e le invia alla cucina.



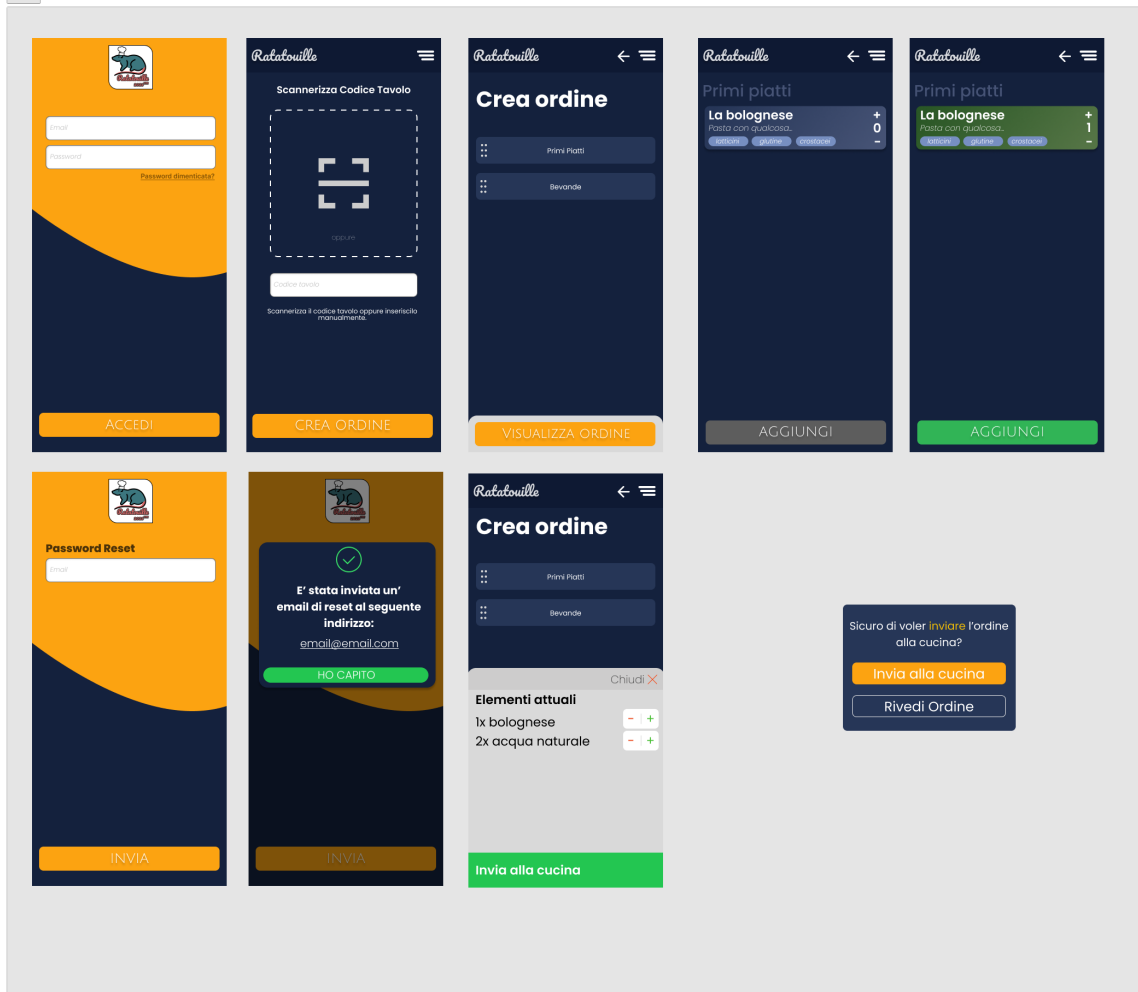
## 1.3 Prototipazione visuale via Mock-up dell'interfaccia utente per tutti i casi d'uso

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).





Caratteristiche





## 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

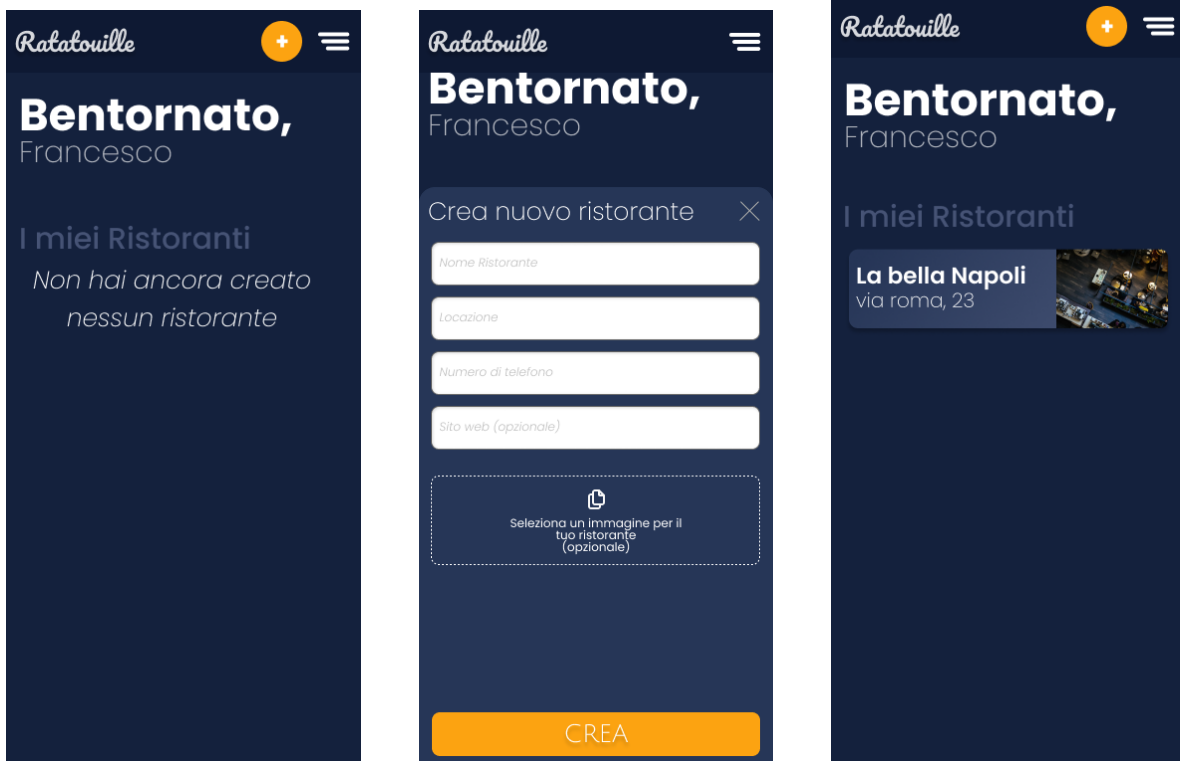
Use Case #1	Crea nuovo ristorante		
Goal in Context	Creazione di un nuovo ristorante con le relative informazioni.		
Preconditions	L'utente deve essere autenticato come amministratore e trovarsi nella schermata Dashboard admin.		
Success End Conditions	Il ristorante viene aggiunto al sistema e viene 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		Apri 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



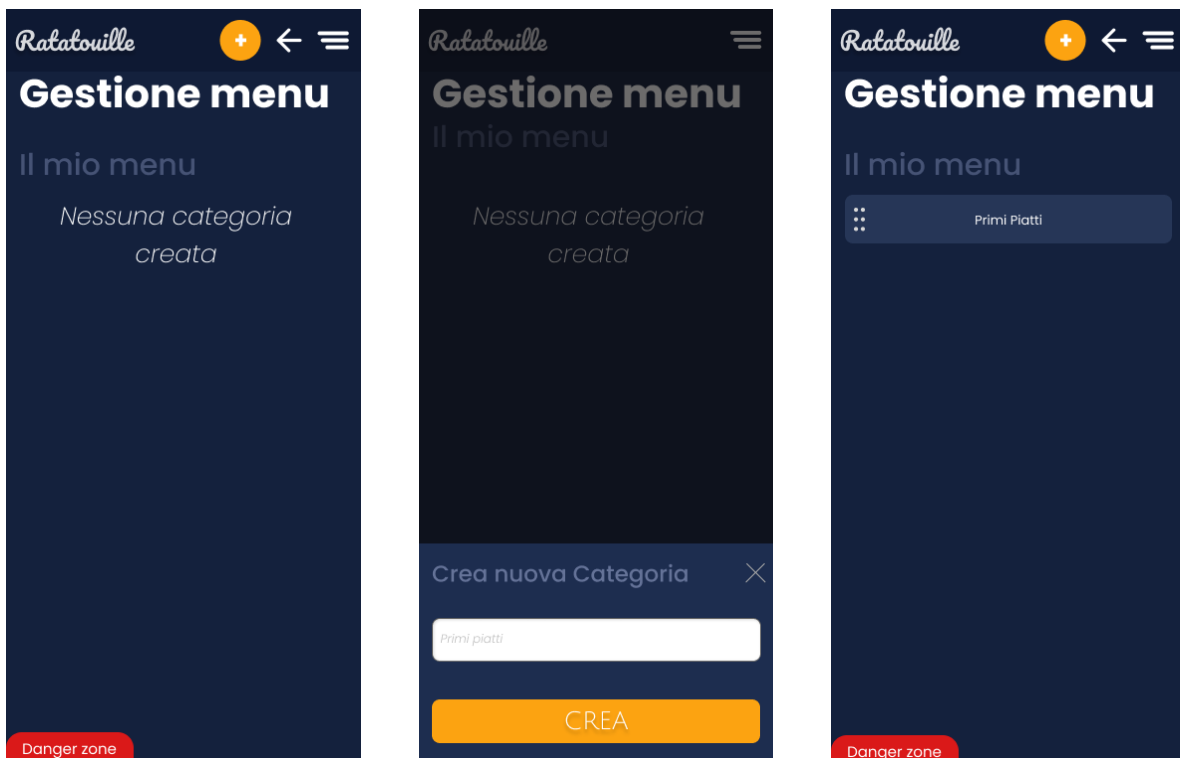


Figure 1: Mock-ups per Crea categoria

## 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.

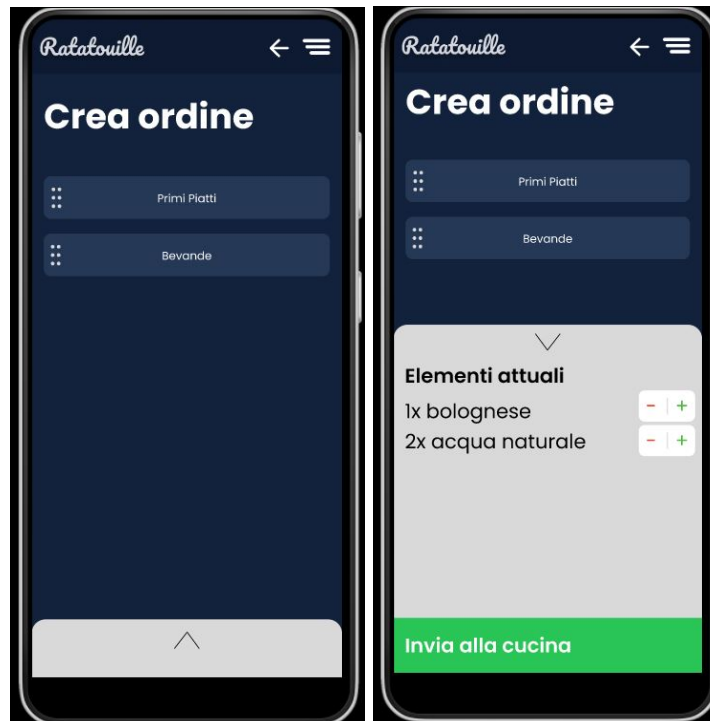


Figure 4: Prima della correzione

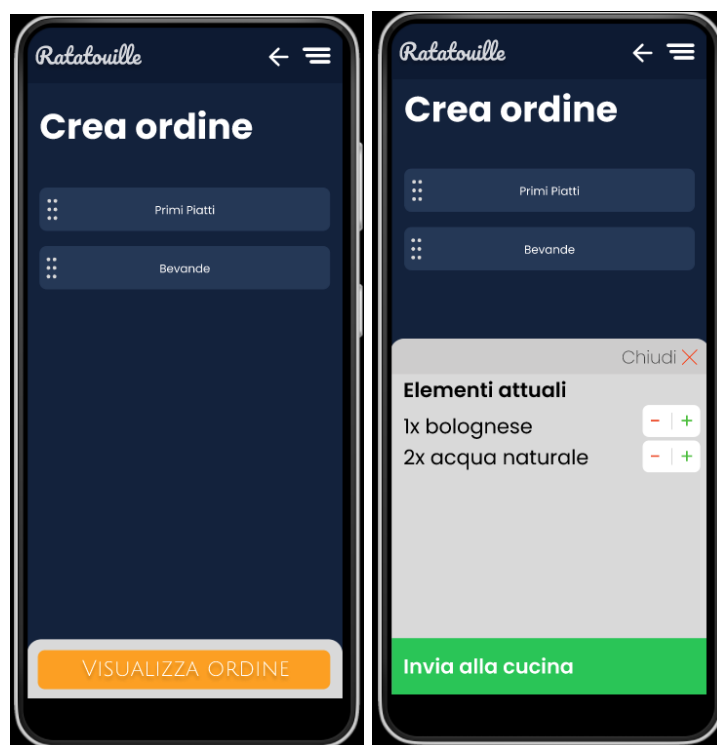


Figure 5: Dopo la correzione



## 1.7 Glossario

Raccolta dei termini utilizzati all'interno della documentazione:

Termine	Descrizione
Admin / Amministratore	Colui che crea le utenze, visiona le statistiche del personale, crea il menu e amministra i ristoranti
Supervisore	Utente con permessi superiori a quelli dell'utente base ma inferiori all'admin. Può modificare categorie e elementi del menù, stampare il conto e gestire i tavoli e le ordinazioni del locale
Figma	Figma è un software di progettazione grafica basato su cloud, utilizzato principalmente per la creazione di interfacce utente, web design e design di prodotto.
Immigrato digitale	Un immigrato digitale è una persona che ha difficoltà nell'utilizzo della tecnologia digitale e delle piattaforme online.

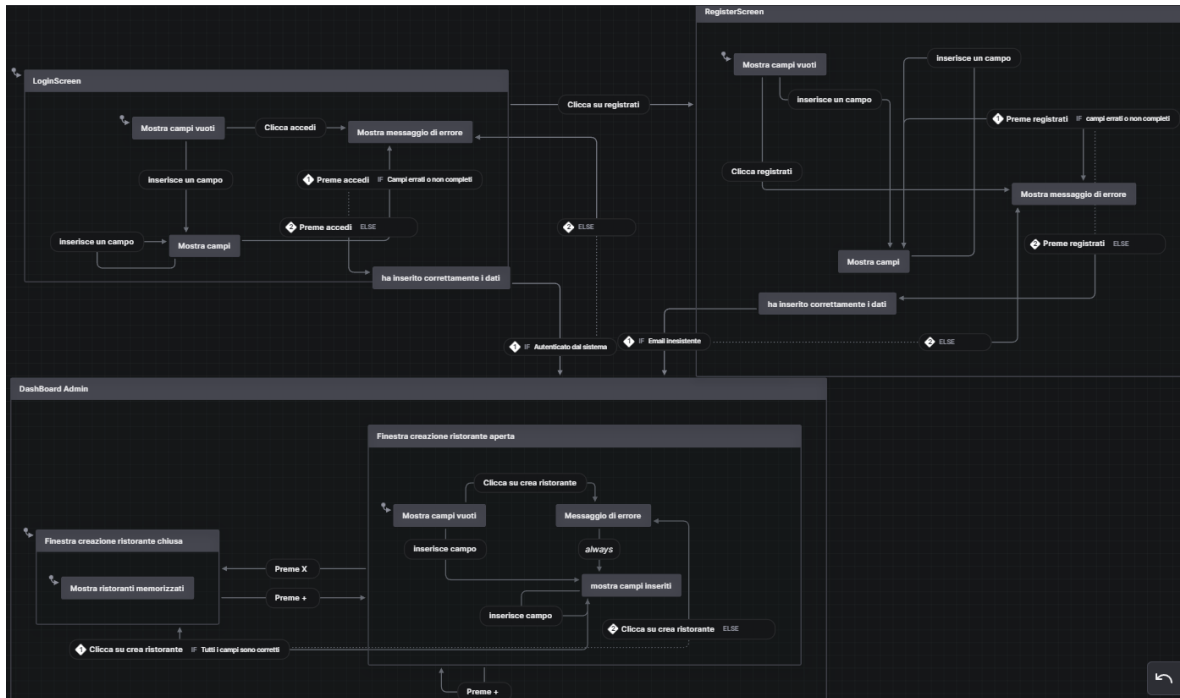
## 2 Specifica dei Requisiti

### 2.1 Classi, oggetti e relazioni di analisi

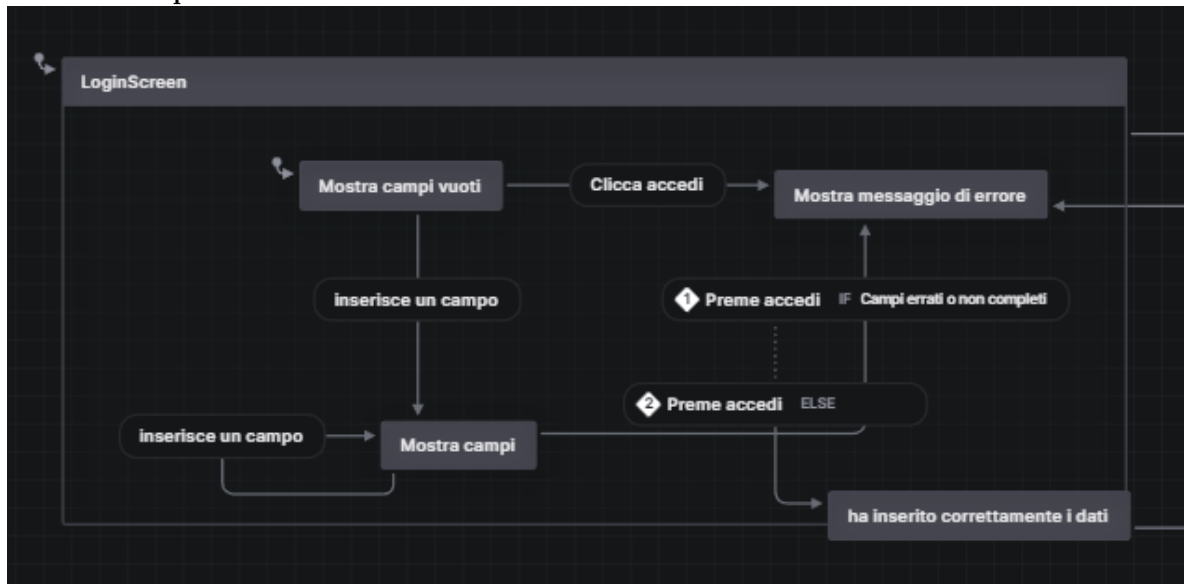
### 2.2 Diagrammi di sequenza di analisi

### 2.3 Prototipazione funzionale via statechart

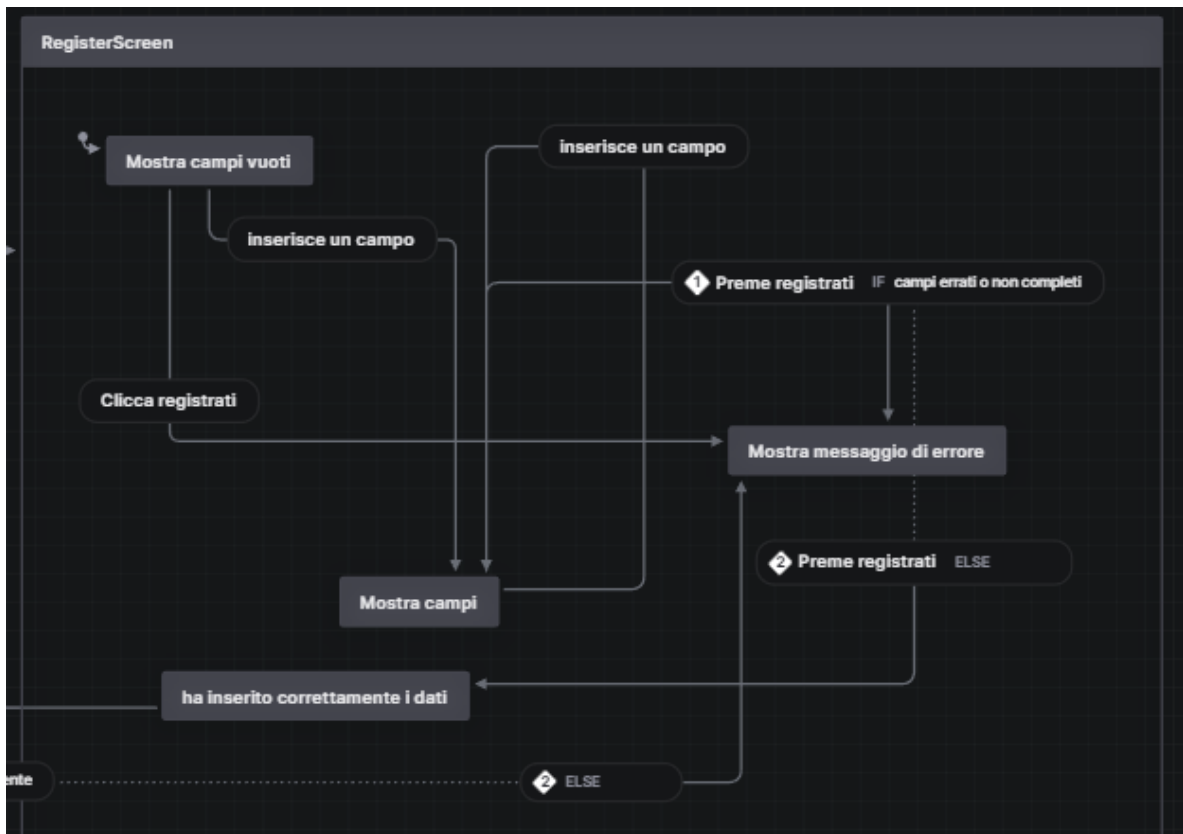
#### 2.3.1 Statechart - Crea un Ristorante



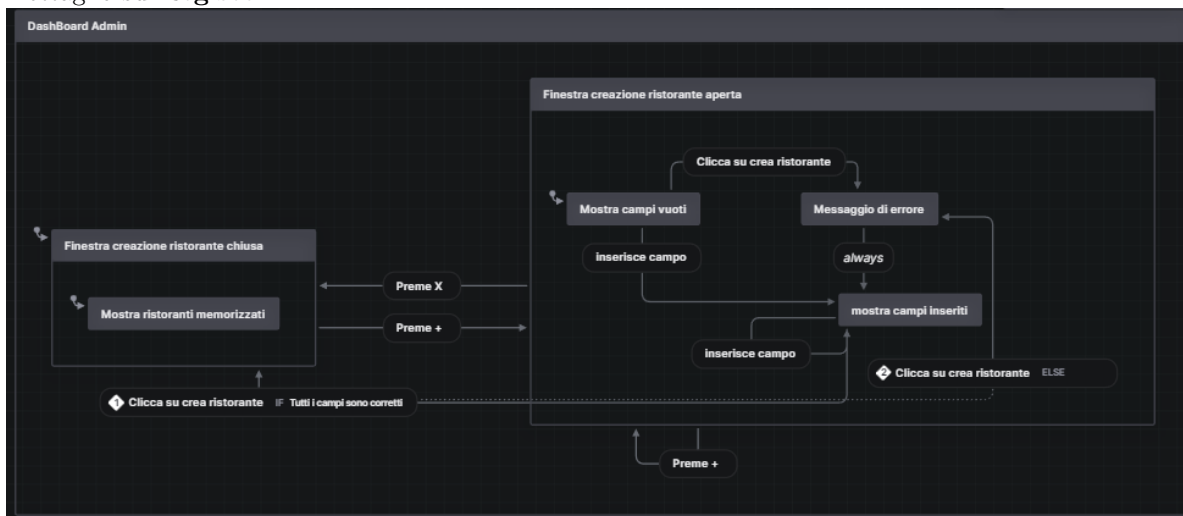
Schema complessivo



Dettaglio su Login



Dettaglio su **Register**

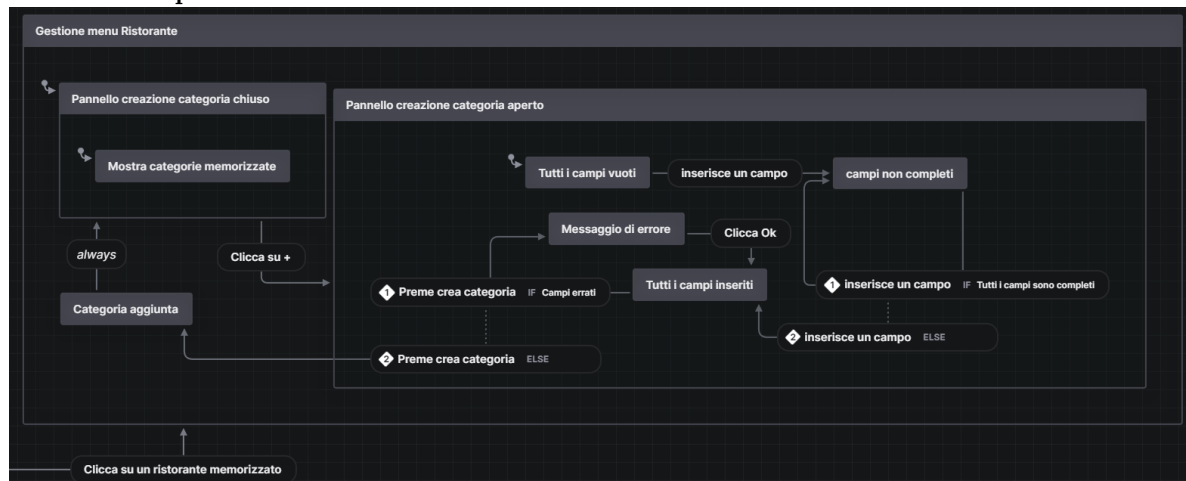


Dettaglio su **Dashboard Admin**

### 2.3.2 Statechart - Crea una Categoria



Schema complessivo



Dettaglio su Gestione menu Ristorante

## II Documento di Design del Sistema

### 3 Analisi dell'architettura

#### 3.1 Architettura 3 Tier

L'architettura a 3 tier (o a 3 livelli) è un'architettura software che prevede la suddivisione dell'applicazione in tre livelli distinti: un livello di presentazione (frontend), un livello di elaborazione (backend) e un livello di persistenza dei dati (database).

##### 3.1.1 Client - Tier 1

Il frontend, o livello di presentazione, ha lo scopo di creare l'interfaccia utente dell'applicazione, che l'utente finale vedrà e con cui interagirà. Nel caso specifico, il frontend è stato realizzato utilizzando React, una libreria JavaScript per la creazione di interfacce utente. Grazie alla sua leggerezza, React consente di creare interfacce utente reattive e performanti, migliorando l'esperienza dell'utente. Inoltre, il frontend dialoga con le API del backend, che forniscono i dati e le funzionalità necessarie per

l'applicazione, e con le API esterne per l'autocompletamento del testo. Questo significa che il frontend non si occupa di elaborare i dati o di effettuare operazioni complesse, ma si limita a presentare i dati e a interagire con gli altri livelli dell'applicazione. In definitiva, lo scopo del frontend è quello di fornire all'utente un'interfaccia intuitiva e funzionale per interagire con l'applicazione, senza dover preoccuparsi degli aspetti tecnici sottostanti.

### 3.1.2 Server - Tier 2

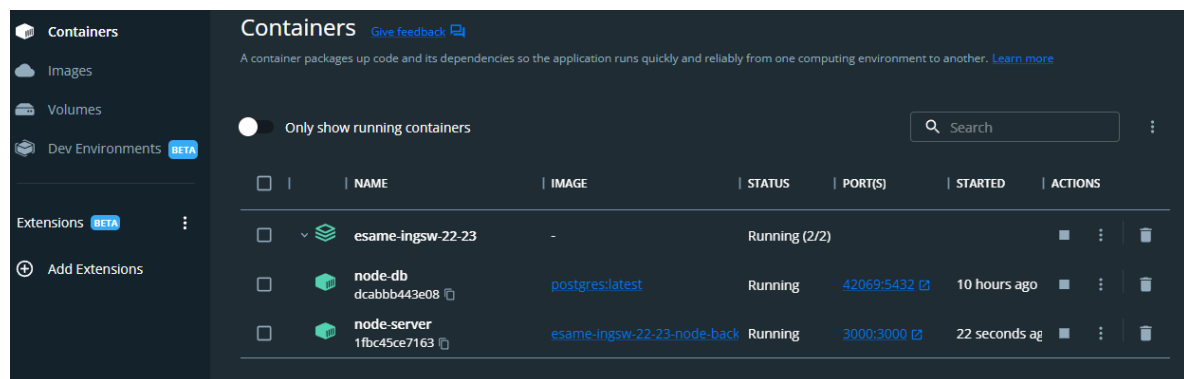
Il livello di backend, o livello di elaborazione, è responsabile dell'elaborazione dei dati e delle logiche di controllo dell'applicazione. Nel caso specifico, è stata realizzata una REST API in Node.js. Una REST API (API di tipo Representational State Transfer) è un'interfaccia che permette alle applicazioni di comunicare tra loro, scambiando dati in formato JSON o XML attraverso richieste HTTP. Essa è basata sul concetto di risorse, ovvero oggetti o dati che possono essere richiesti, creati, aggiornati o cancellati. Per garantire la sicurezza delle richieste e delle risposte, è stato utilizzato JSON Web Token (JWT). Un JWT è un token di sicurezza che viene utilizzato per autenticare e autorizzare gli utenti in modo sicuro, senza dover trasmettere le credenziali dell'utente ad ogni richiesta. Il server è un layer di controllo tra il client (ovvero il frontend) e il database. Esso riceve le richieste dal frontend, le elabora e le inoltra al database, poi riceve la risposta dal database, la elabora e la invia al frontend. In questo modo, il server offre un'interfaccia sicura e controllata per l'accesso ai dati e alle funzionalità dell'applicazione. In definitiva, il livello di backend è responsabile dell'elaborazione dei dati e delle logiche di controllo dell'applicazione, garantendo al contempo la sicurezza e il controllo dell'accesso ai dati e alle funzionalità dell'applicazione stessa.

### 3.1.3 Database - Tier 3

Il terzo tier dell'architettura a tre livelli è rappresentato dal database PostgreSQL. In questo caso, il database è stato creato utilizzando un file di creazione dello schema e delle istanze, che definisce la struttura del database e le tabelle in esso contenute. Inoltre, è stato deciso di utilizzare Docker per la gestione dell'intera applicazione, incluso il database. In particolare, il container di PostgreSQL viene avviato come primo, poiché il container del backend dipende da esso e richiede una connessione al database per funzionare correttamente.

### 3.1.4 Schema architettura

### 3.1.5 Docker containers in azione



## 3.2 Routes del Backend

**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`

**LEGGENDA della tabella** **SUP** = Accesso riservato a Superiore e Admin, **ADMIN** = Accesso riservato ad admin. Se non specifico il tipo di un parametro allora è di tipo **string**.

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

Nome route	Descrizione	Tipo	Autorizzazione	Richiesta	Risposta
/utenza/: id_ristorante	Crea una nuova utenza	POST	Bearer Token, ADMIN	id_ristorante JSON nome cognome email password ruolo telefono supervisore	JSON success:bool data
/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 id_ristorante	JSON success:bool data
/categoria/: id_categoria	Elimina categoria	DELETE	Bearer Token, SUP	id_categoria	JSON success:bool data
/categoria/: id_categoria	Modifica categoria	PUT	Bearer Token, SUP	id_categoria	JSON success:bool data
/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
/elemento/: id_elemento	Cancella elemento	DELETE	Bearer Token, SUP	id_elemento	JSON success:bool data
/elemento/: id_elemento	Modifica elemento	PUT	Bearer Token, SUP	id_elemento	JSON success:bool data
/elemento	Crea elemento	POST	Bearer Token, SUP	JSON nome id_categoria prezzo descrizione allergeni	JSON success:bool data

Nome route	Descrizione	Tipo	Autorizzazione	Richiesta	Risposta
/allergene	Crea una nuovo allergene	POST	Bearer Token, SUP	JSON nome <code>id_elemento</code>	JSON success:bool data
/allergeni/: <code>id_elemento</code>	Visualizza allergene	GET	Bearer Token	<code>id_elemento</code>	<i>Allergene</i> []
/allergene/: <code>id_allergene</code>	Cancella allergene	DELETE	Bearer Token	<code>id_allergene</code>	JSON success:bool data



### 3.3 Cloud Hosting

DigitalOcean è stata la nostra scelta per rendere accessibile il software su Internet, grazie alla sua capacità di fornire un'infrastruttura di hosting affidabile e facile da usare. DigitalOcean ci ha fornito un indirizzo IP statico, il che significa che il nostro sito web sarà sempre raggiungibile utilizzando lo stesso indirizzo IP. Questo è particolarmente importante per le applicazioni che richiedono un'accessibilità costante e affidabile. Per quanto riguarda la configurazione, Docker ha reso la distribuzione dell'applicazione estremamente semplice e portatile, consentendoci di caricare l'immagine del server in cloud. Infine, abbiamo caricato il frontend compilato e ha funzionato senza problemi.

### 3.4 Pannello di controllo in cloud

## 4 Motivazione delle scelte adottate

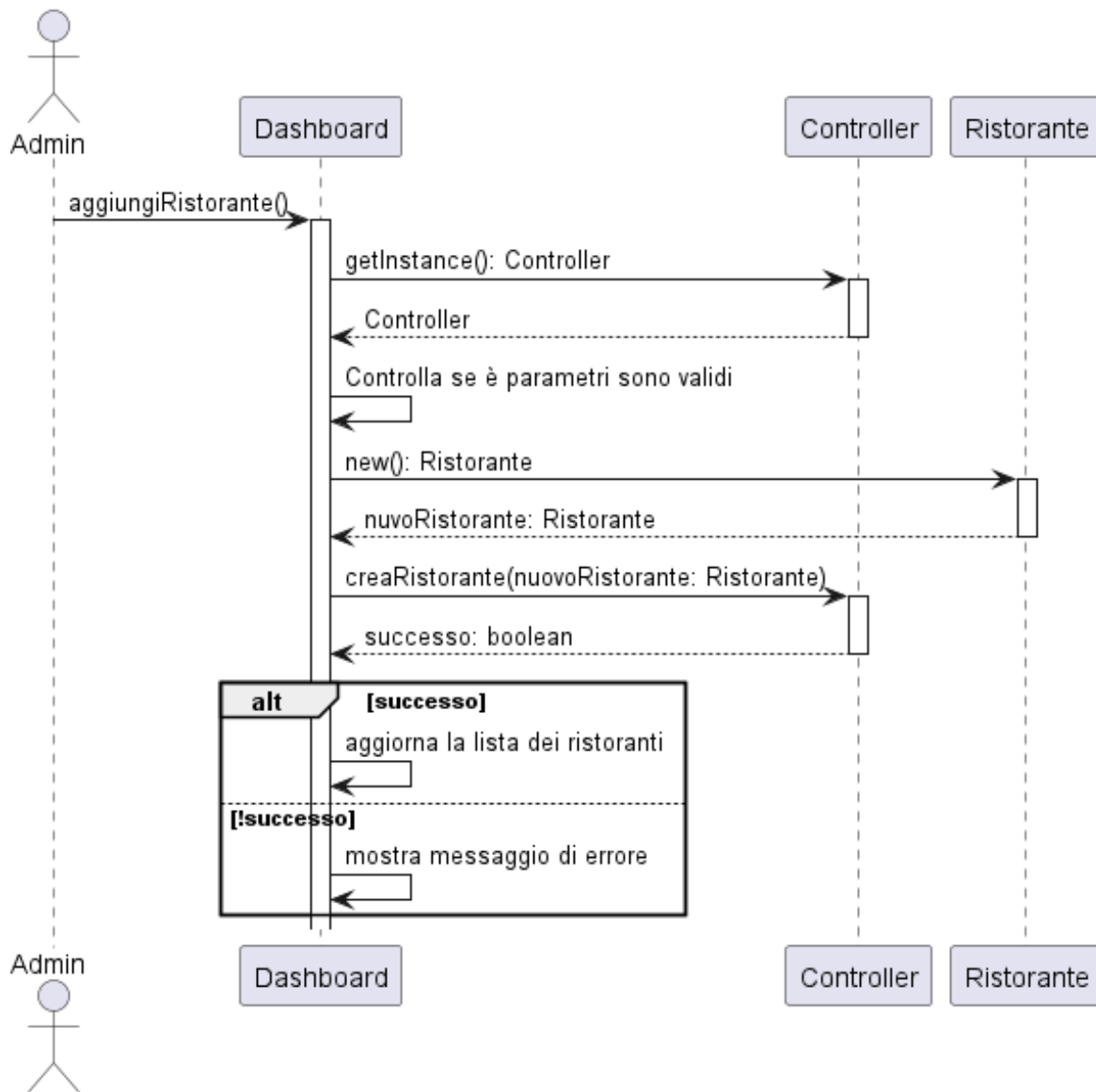
L'architettura a tre livelli è stata scelta per organizzare il progetto in modo più strutturato e versatile. In questo modo, ogni livello ha un compito specifico e ben definito, e le interazioni tra di essi sono chiare e gestibili. Inoltre, questo tipo di architettura consente di separare la logica di presentazione dall'elaborazione dei dati, semplificando lo sviluppo e la manutenzione dell'applicazione. Per quanto riguarda la scelta del cloud, è stata fatta considerando diversi fattori come il numero di accessi al mese, il costo, la banda e lo spazio a disposizione. Il cloud offre molte soluzioni di hosting scalabili e personalizzabili, che possono adattarsi alle esigenze dell'applicazione. L'utilizzo di Docker è stato scelto per ottenere un'applicazione robusta e facilmente replicabile in diversi ambienti. Docker permette di isolare i servizi in container, in modo da gestirli in maniera indipendente e controllata. Inoltre, Docker garantisce che i servizi vengano avviati nell'ordine corretto, semplificando la configurazione dell'applicazione. Infine, l'uso di Docker consente di avere il controllo completo delle porte esposte, che possono essere gestite in modo flessibile e personalizzato a seconda delle esigenze dell'applicazione.

## 5 Diagramma delle classi di design

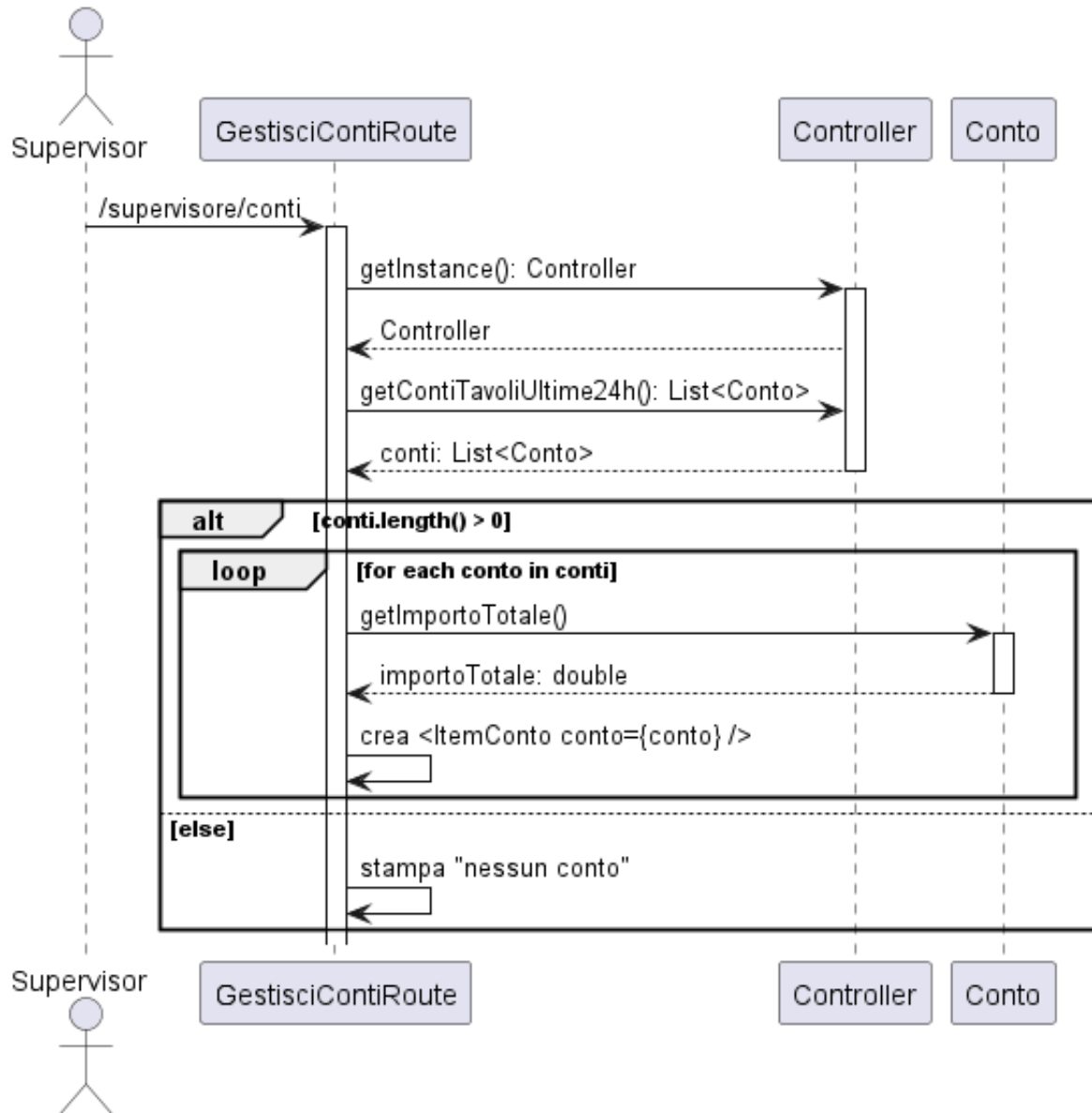
classi del frontend

## 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