



# UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Corso di Laurea in Informatica

Anno accademico 2022/2023



## Progetto di Ingegneria del Software

Ratatouille23 – un sistema per la gestione e l'operatività di un'attività di ristorazione

**Commissionato da:**  
*SoftEngUniNA*

**Sviluppato da:**  
Pasquale Orlando - N86003266  
Alessia Verrazzo - N86003326



# Sommario

1 Introduzione.....	8
2 Analisi dei Requisiti .....	9
2.1 Requisiti.....	9
2.1.1 Requisiti funzionali .....	9
2.1.2 Requisiti non funzionali.....	10
2.1.3 Requisiti di dominio .....	11
2.2 Casi d'uso .....	11
2.2.1 Use Case Generico .....	12
2.2.2 Use Case Gestione piatti.....	13
2.2.3 Use Case Gestione categorie.....	13
2.2.4 Use Case Gestione menu .....	14
2.3 Tabelle di Cockburn .....	15
2.3.1 Organizzare menù .....	15
2.3.2 Prendere ordinazioni.....	19
2.3.3 Statistiche addetti alla sala.....	28
2.3.4 Visualizzazione ordinazioni (addetto alla cucina).....	32
2.4 Prototipazione visuale via Mock up .....	35
2.4.1 Login .....	35
2.4.2 Mock up Amministratore .....	36
2.4.3 Mock up Supervisore.....	44

2.4.3 Mock up Addetto alla sala.....	51
2.4.4 Mock up addetto alla cucina.....	55
2.4.5 Popup .....	57
2.5 Individuazione target utenti .....	61
2.6 Valutazione dell'usabilità a priori .....	66
2.6.1 Test di usabilità.....	67
2.7 Glossario.....	69
3 Specifica dei requisiti .....	70
3.1 Classi, oggetti e relazioni di analisi.....	70
3.1.1 Entità .....	70
3.1.2 Class Diagram Autenticazione .....	71
3.1.3 Class Diagram Dipendenti .....	71
3.1.4 Class Diagram Tavoli e Nuove ordinazioni .....	72
3.1.5 Class Diagram Gestione ordinazioni .....	73
3.1.6 Class Diagram Gestione profilo .....	74
3.1.7 Class Diagram Gestione piatti .....	75
3.1.8 Class Diagram Gestione categorie .....	76
3.1.9 Class Diagram Gestione menù.....	77
3.1.10 Class Diagram Statistiche addetti alla sala e cucina .....	78
3.1.11 Class Diagram Gestione attività.....	79
3.2 Sequence Diagram di analisi.....	79
3.2.1 Organizzare menù .....	80

3.2.2 Prendere ordinazioni.....	81
3.3 Statechart dell'interfaccia grafica.....	83
3.3.1 Organizzare menù .....	83
3.3.2 Effettuare nuova ordinazione .....	84
3.3.3 Statistiche addetti alla sala .....	85
3.3.4 Visualizzazione ordinazioni (addetti alla cucina) .....	86
4 Analisi dell'architettura.....	88
4.1 Architettura cloud.....	89
4.2 Il Server SpringBoot.....	90
4.3 Il WebServer ReactJS.....	92
5 Tecnologie adoperate .....	93
5.1 Motivazioni .....	93
6 Class Diagram di Design .....	95
6.1 Autenticazione .....	95
6.2 Dipendenti .....	96
6.3 Tavoli e Nuova ordinazione .....	97
6.4 Gestione ordinazioni.....	98
6.5 Gestione profilo .....	98
6.6 Gestione piatti .....	99
6.7 Gestione categorie.....	100
6.8 Gestione menù.....	101
6.9 Statistiche addetti alla sala e cucina.....	102

6.10 Gestione attività .....	102
<b>7 Sequence Diagram di Design .....</b>	<b>103</b>
7.1 Organizzare menù.....	103
7.2 Prendere ordinazioni .....	104
7.3 Statistiche addetti alla sala .....	106
7.4 Visualizzazione ordinazioni (addetti alla cucina).....	110
<b>8 Codice xUnit .....</b>	<b>112</b>
8.1 CalcolaNuovaSelezione.....	113
8.2 OttieniInfoPiatto.....	116
8.3 CalcolaPiattiEvasi.....	118
8.4 CalcolaOrdiniDipendente.....	123
<b>9 Valutazione dell’usabilità sul campo.....</b>	<b>129</b>

# **Documento dei Requisiti Software**

# 1 Introduzione

*Ratatouille23* è un sistema dedicato al supporto e alla gestione delle attività di ristorazione. Inoltre, è un'applicazione performante e affidabile progettata per offrire un'esperienza intuitiva, rapida e piacevole agli utenti.

L'obiettivo principale di *Ratatouille23* è semplificare le operazioni quotidiane all'interno di un ristorante, consentendo una gestione efficiente e ottimizzata. Il sistema fornisce diverse funzionalità chiave che coprono le diverse esigenze dei vari ruoli all'interno dell'attività di ristorazione. Per cominciare, *Ratatouille23* offre un modulo di gestione degli utenti che consente all'amministratore di creare account per il personale, inclusi addetti alla sala, addetti alla cucina e supervisori. Ogni utente, al primo accesso, sarà tenuto a reimpostare la password fornita dall'amministratore per garantire la sicurezza dei propri dati. Una delle caratteristiche più importanti del sistema è la possibilità di personalizzare il menù dell'attività di ristorazione. L'amministratore o un supervisore avrà il controllo completo sulla creazione e l'eliminazione di elementi dal menù. Sarà possibile definire il nome, il costo, la descrizione e gli allergeni comuni di ciascun elemento, nonché organizzare gli elementi in categorie personalizzabili. Inoltre, il sistema supporterà l'auto-completamento per semplificare l'inserimento dei prodotti, utilizzando talvolta dati disponibili da fonti come <https://it.openfoodfacts.org/data>.

*Ratatouille23* rende anche più efficiente il processo di registrazione degli ordini. Gli addetti alla sala potranno registrare le ordinazioni indicando l'identificativo del tavolo e gli elementi del menù desiderati, garantendo un servizio accurato e tempestivo. Per il personale addetto alla cucina, il sistema offre una visualizzazione in tempo reale delle ordinazioni, consentendo loro di procedere con la preparazione dei piatti e tenere traccia delle ordinazioni già evase. Ciò contribuirà a ottimizzare il flusso di lavoro e migliorare l'efficienza operativa. Inoltre, *Ratatouille23* mette a disposizione dell'amministratore statistiche dettagliate sulla produttività del personale sia in sala che in cucina. *Ratatouille23* è stato sviluppato con l'obiettivo di semplificare e ottimizzare la gestione delle attività di ristorazione, offrendo un sistema completo e intuitivo.

## 2 Analisi dei Requisiti

Questa sezione si occupa di descrivere in maniera astratta, anche attraverso specifici modelli UML (vedi voce [UML](#)), le funzionalità che il sistema offre, le categorie di utenti al quale il sistema è indirizzato e una prima valutazione dell'usabilità dell'applicativo.

### 2.1 Requisiti

#### 2.1.1 Requisiti funzionali

Da un'analisi dei requisiti utente, sono stati elaborati i seguenti requisiti di sistema:

- Il sistema fornisce un unico amministratore, il quale può creare utenze per i propri dipendenti.
- Il sistema permette ad ogni utente di autenticarsi.
- Il sistema permette ad ogni utente di re-impostare la propria password.
- Il sistema permette all'amministratore e ai supervisori di creare/eliminare/modificare piatti.
- Il sistema permette all'amministratore e ai supervisori di creare/eliminare/modificare categorie.
- Il sistema permette all'amministratore e ai supervisori di creare/eliminare/modificare/ordinare menu.
- Il sistema permette all'amministratore e agli addetti alla sala di prendere ordinazioni, indicando l'identificativo del tavolo e gli elementi del menù desiderati.
- Il sistema permette all'amministratore e agli addetti alla cucina di visualizzare le ordinazioni in tempo reale, procedere alla preparazione dei piatti, e tenere traccia delle ordinazioni già evase.
- Il sistema permette all'amministratore di visualizzare statistiche dettagliate sulla produttività del personale addetto alla sala in un lasso di tempo personalizzabile.
- Il sistema permette all'amministratore di visualizzare statistiche dettagliate sulla produttività del personale addetto alla cucina in un lasso di tempo personalizzabile.

## 2.1.2 Requisiti non funzionali

Quando si sviluppa un'applicazione è fondamentale definire e comprendere non solo le funzionalità richieste, ma anche i requisiti non funzionali. I requisiti non funzionali rappresentano gli aspetti critici che vanno oltre le funzionalità stesse e che influenzano l'esperienza degli utenti, le prestazioni, la sicurezza e altri attributi essenziali dell'applicazione. I requisiti non funzionali riguardanti *Ratatouille23* sono:

1. **Prestazioni:** ogni pagina si carica entro 3 secondi.
2. **Efficienza:** l'applicazione utilizza le risorse (ad esempio, CPU, memoria, spazio di archiviazione) in modo efficiente, minimizzando il consumo e ottimizzando le prestazioni.
3. **Interfaccia utente intuitiva:** l'interfaccia utente è intuitiva, facile da usare e orientata all'utente, riducendo la curva di apprendimento e migliorando l'esperienza complessiva.
4. **Compatibilità:** l'applicazione è compatibile con diversi browser e sistemi operativi, assicurando un'esperienza coerente per gli utenti.
5. **Manutenibilità:** l'applicazione è facilmente manutenibile, consentendo la correzione di bug, l'aggiunta di nuove funzionalità o l'aggiornamento del software in modo efficiente.
6. **Affidabilità:** l'applicazione è affidabile e prevedibile, evitando crash, malfunzionamenti o perdite di dati significative.
7. **Sicurezza:** l'applicazione garantisce la sicurezza dei dati sensibili degli utenti, proteggendo l'accesso non autorizzato, prevenendo attacchi informatici e adottando le pratiche di sicurezza consigliate.
8. **Scalabilità:** l'applicazione è in grado di gestire un aumento del carico senza compromettere le prestazioni, consentendo l'espansione del numero di utenti.
9. **Integrità dei dati:** l'applicazione garantisce l'integrità dei dati, evitando errori, corruzione o perdita di dati durante il processo di archiviazione o elaborazione.

## 2.1.3 Requisiti di dominio

Il sistema rispetta le seguenti direttive e norme:

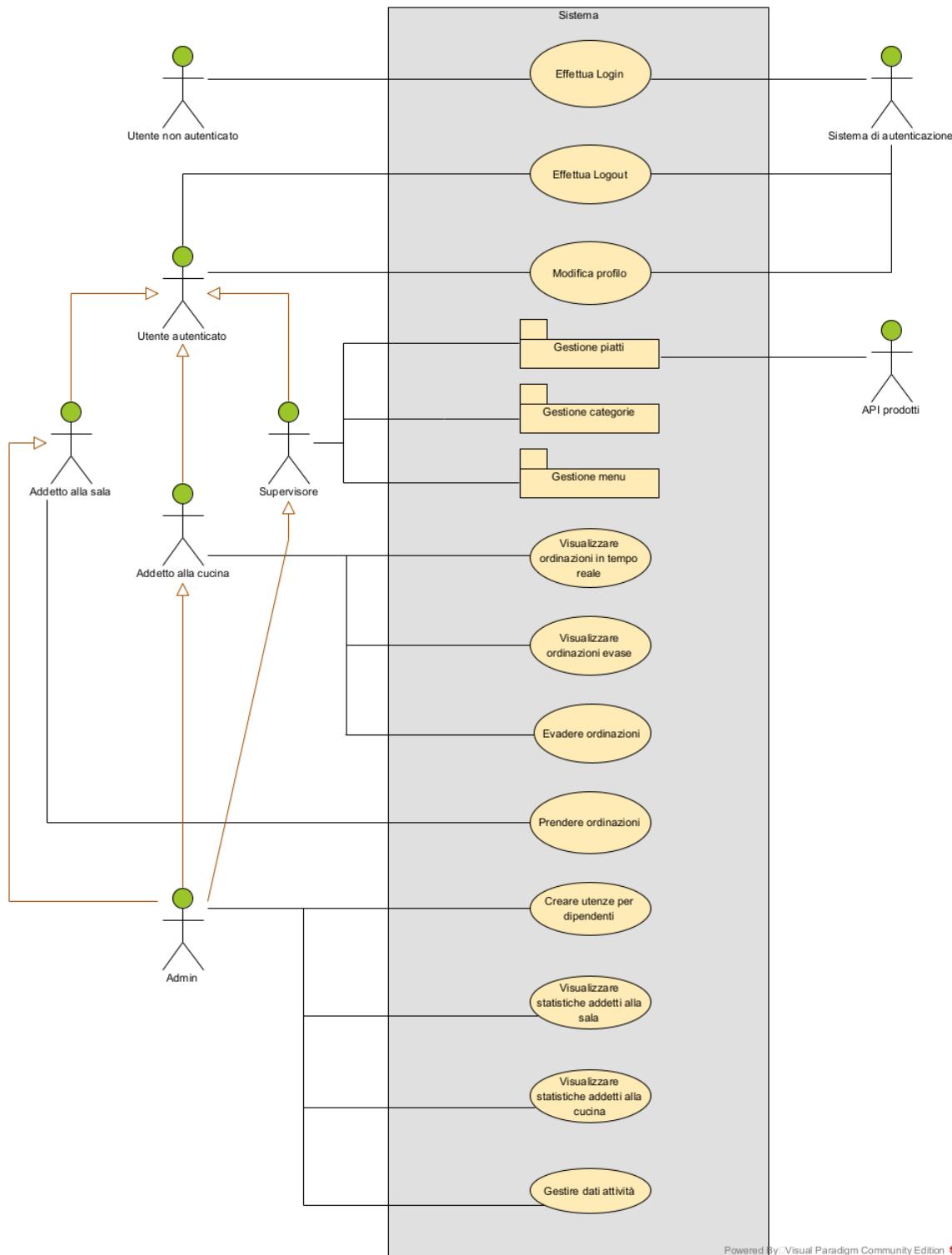
- **GDPR:** è un regolamento dell'UE in materia di trattamento dei dati personali e di privacy.
- **ISO/IEC 25010 (SQuaRE):** è una serie di requisiti della qualità di sistemi e valutazione dei prodotti.
- **ISO 13407 (Human-Centered Design Processes for Interactive Systems):** fornisce linee guida per l'applicazione di approcci centrati sull'utente nello sviluppo di sistemi interattivi.

## 2.2 Casi d'uso

Di seguito sono modellate le funzionalità del sistema, mediante l'uso di Use Cases Diagram (vedi voce [Use Case Diagram](#)).

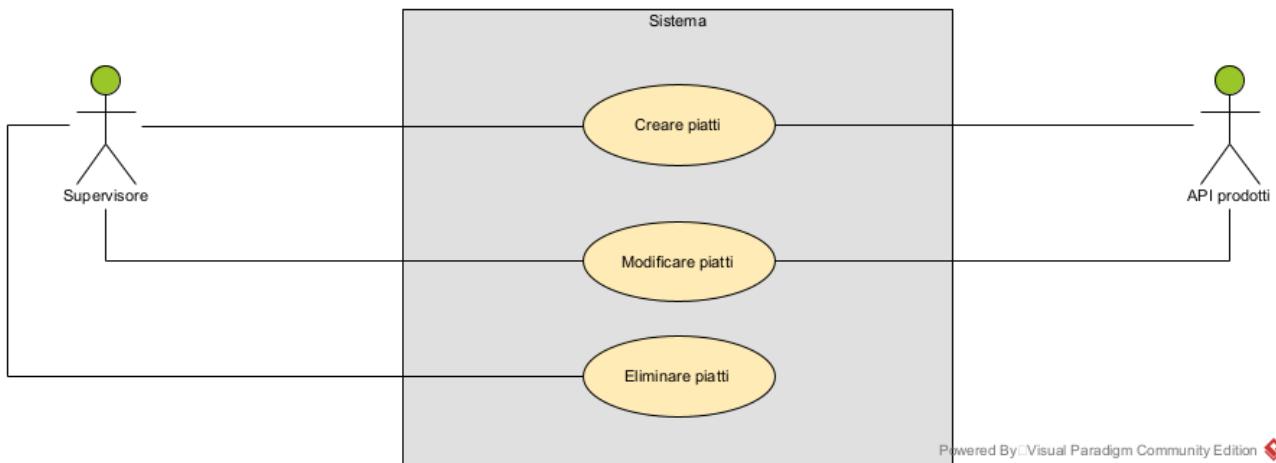
## 2.2.1 Use Case Generico

Nel seguente Use Case Diagram alcune funzionalità sono state racchiuse in package per questioni di spazio.

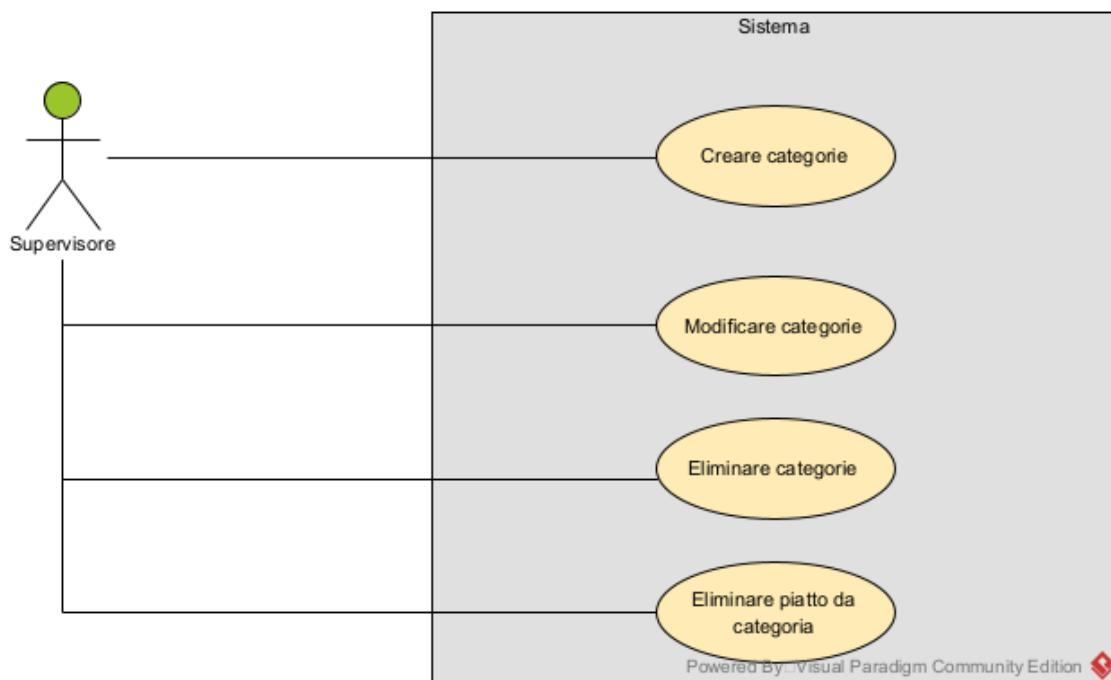


Powered by: Visual Paradigm Community Edition

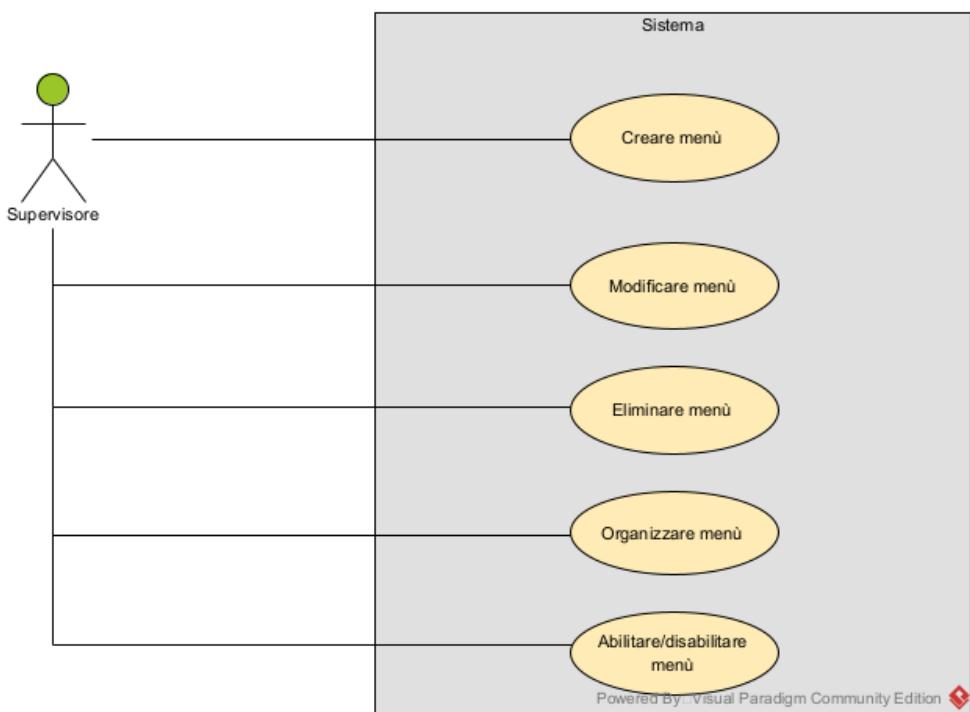
## 2.2.2 Use Case Gestione piatti



## 2.2.3 Use Case Gestione categorie



## 2.2.4 Use Case Gestione menu



## 2.3 Tabelle di Cockburn

Le tabelle di Cockburn, introdotte da Alistair Cockburn, sono una tecnica di documentazione che rappresenta i casi d'uso in formato tabellare. Queste tabelle forniscono una struttura organizzata per descrivere gli attori, i casi d'uso, le precondizioni, le post-condizioni e gli scenari tipici associati a un sistema, facilitando la comprensione e la comunicazione delle specifiche di un'applicazione.

### 2.3.1 Organizzare menù

The image shows a user interface mockup for a system named 'RATATOUILLE'. On the left is a dark green sidebar menu with a chef's hat icon at the top. The menu items include: Dashboard, Dipendenti, Tavoli, Menu (selected), Piatti, Categorie, Gestionе menù (highlighted in green), Statistiche, Addetti alla sala, Addetti alla cucina, Ordinazioni, Profilo, Gestione Attività, and Logout. To the right is a light gray main area titled 'Gestione menù'. It shows a list of menu items with selection status and enable/disable toggles:

Caso d'uso	Selezionato	Disabilita	Abilita
Menu generale	1 Selezionato	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Menu estivo		<input type="checkbox"/>	<input checked="" type="checkbox"/>
Menu pizze		<input type="checkbox"/>	<input checked="" type="checkbox"/>

Mockup M1

**Gestione menù**

Per aggiungere, eliminare o modificare una categoria clicca **QUI**.

**Menù 1**

Categoria	Disabilita	Abilita
Antipasti	<input checked="" type="checkbox"/>	Abilita
Frittura all'italiana crocch, arancini, zeppole, polenta	<input checked="" type="checkbox"/>	Abilita
Tris di bruschette bruschetta con pomodorini gialli del Pienno, bruschetta con pesto e pomodorini, bruschetta con pomodori rossi di san Marzano	<input checked="" type="checkbox"/>	Abilita
Primi piatti	<input checked="" type="checkbox"/>	Abilita
Primi piatti di pesce	<input checked="" type="checkbox"/>	Abilita
Contorni	<input checked="" type="checkbox"/>	Abilita
Dolci	<input checked="" type="checkbox"/>	Abilita

[Torna a Selezione menù](#) [Salva modifiche](#)

Mockup M2

**Gestione menù**

Per aggiungere, eliminare o modificare una categoria clicca **QUI**.

**Menù 1**

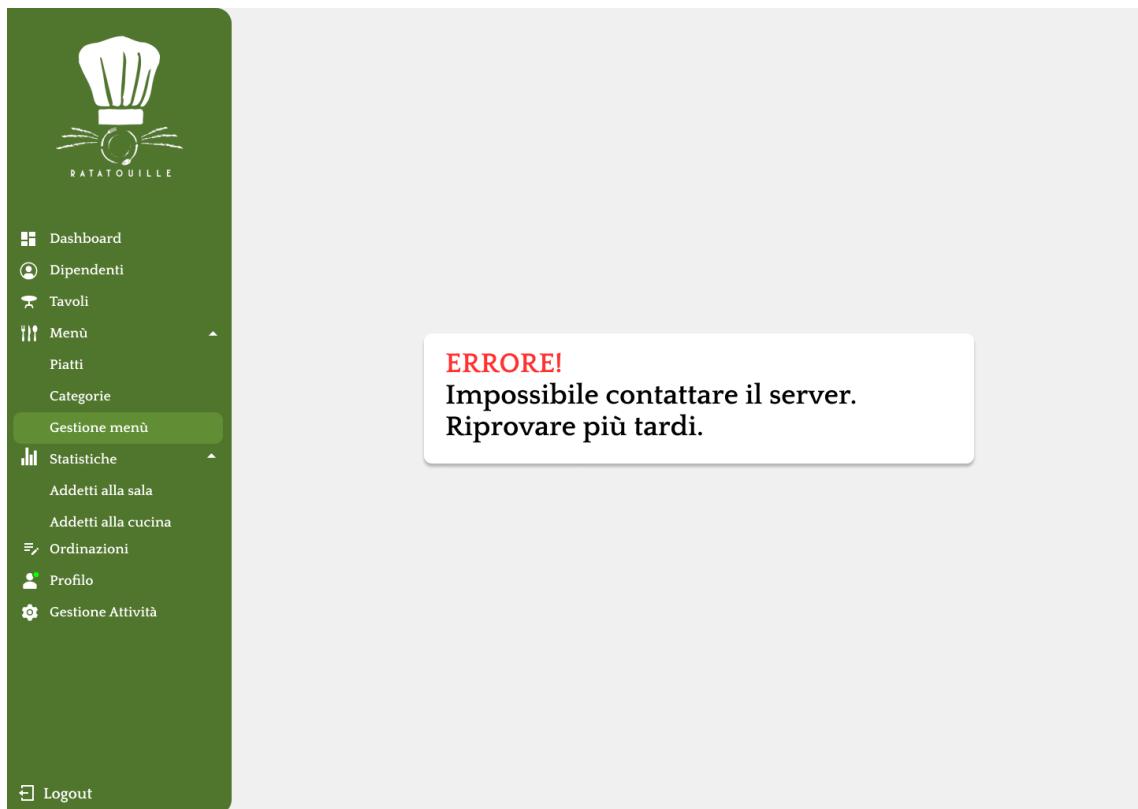
Categoria	Disabilita	Abilita
Antipasti	<input checked="" type="checkbox"/>	Abilita
Frittura all'italiana crocch, arancini, zeppole, polenta	<input checked="" type="checkbox"/>	Abilita
Tris di bruschette bruschetta con pomodorini gialli del Pienno, bruschetta con pesto e pomodorini, bruschetta con pomodori rossi di san Marzano	<input checked="" type="checkbox"/>	Abilita
Primi piatti	<input checked="" type="checkbox"/>	Abilita
Primi piatti di pesce	<input checked="" type="checkbox"/>	Abilita
Contorni	<input checked="" type="checkbox"/>	Abilita
Dolci	<input checked="" type="checkbox"/>	Abilita

**Modifiche effettuate**

Modifiche effettuate con successo!

[Torna a Selezione menù](#) [Salva modifiche](#)

Mockup M3



Mockup M4

Gestione menù

Per aggiungere, eliminare o modificare una categoria clicca QUI.

Menù 1

Categoria	Descrizione	Disabilita	Abilita
Antipasti	Frittura all'italiana crocchette, arancini, zeppole, polenta Tris di bruschette bruschetta con pomodorini gialli del Pienno, bruschetta con pesto e pomodorini, bruschetta con pomodori rossi di san Marzano	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Primi piatti		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Primi piatti di pesce		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Contorni		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Dolci		<input checked="" type="checkbox"/>	<input type="checkbox"/>

Sei sicuro di voler tornare indietro?

Se torni indietro, le modifiche non salvate non verranno applicate!

SI      NO

Torna a Selezione menù      Salva modifiche

Mockup M5

<b>Use Case #1</b>	Organizzare menù		
<b>Goal in Context</b>	L'utente vuole organizzare il menù, gestendo l'ordine delle categorie e dei piatti al loro interno.		
<b>Preconditions</b>	L'utente deve essere autenticato e deve essere l'amministratore o un supervisore.		
<b>Success End Condition</b>	L'utente riesce a organizzare il menù.		
<b>Failed End Condition</b>	L'utente non riesce a organizzare il menù.		
<b>Primary actor</b>	Supervisore, Amministratore		
<b>Trigger</b>	Da qualsiasi finestra, l'utente apre la tendina "Menù" nella SideBar e seleziona "Gestione Menù".		
<b>Description</b>	<b>Step</b>	<b>Supervisore, Admin</b>	<b>Sistema</b>
	1	Seleziona il menù da ordinare	
	2		Mostra M2
	3	Organizza le categorie e i piatti al loro interno, trascinandoli con il tasto "Drag&Drop"	
	4	Sceglie se abilitare o disabilitare una categoria	
	5	Preme su "Salva modifiche"	
	6		Salva le modifiche nel server
	7		Mostra Popup Successo (M3)
	8	Clicca "OK"	
<b>Extension #1</b>	<b>Step</b>	<b>Supervisore, Admin</b>	<b>Sistema</b>
Il sistema non riesce a connettersi al server.	1.a		Mostra Popup di Errore (M4)
<b>Extension #2</b>	<b>Step</b>	<b>Supervisore, Admin</b>	<b>Sistema</b>
L'utente vuole annullare le modifiche.	5	Preme "Torna a Selezione Menù"	
	6		Mostra M5
	7	Preme "Si"	

## 2.3.2 Prendere ordinazioni

**Tavoli**

- Libero
- Conto richiesto
- Occupato

1	2	3	4 Ospiti: 3	5 Ospiti: 3	6 Ospiti: 3	7 Ospiti: 3	8 Ospiti: 3
9 Ospiti: 3	10 Ospiti: 3						

Mockup M1

**Tavoli**

- Libero
- Conto richiesto
- Occupato

1	2	3	4 Ospiti: 3	5 Ospiti: 3	6 Ospiti: 3	7 Ospiti: 3	8 Ospiti: 3
9 Ospiti: 3	10 Ospiti: 3						

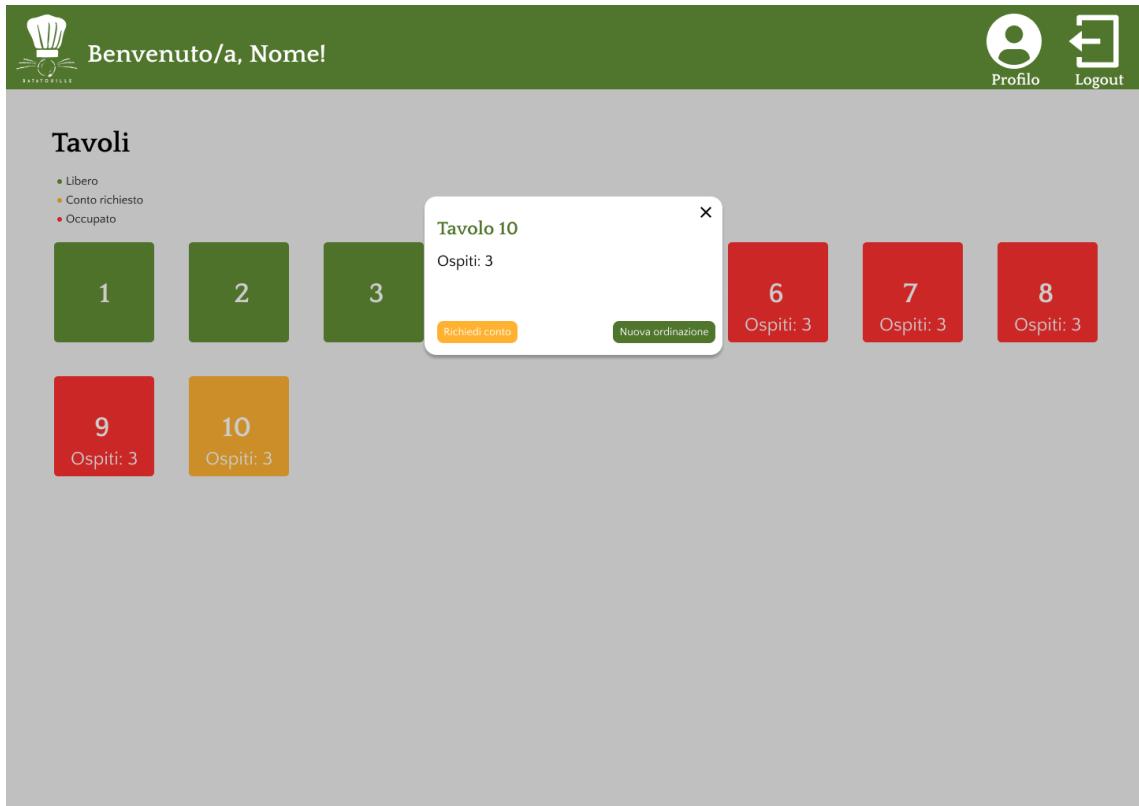
Nuovi ospiti

Hai selezionato il tavolo: 1

Ospiti:

[Annulla](#) [Occupava tavolo](#)

Mockup M2



Mockup M3

Indietro Benvenuto/a, Nome!

Menù

Menù 1

Cerca

Antipasti Primi Primi di pesce Secondi Contorni Dolci Bibite

Tris di bruschette ⓘ Frittura all'italiana ⓘ Fiori di zucca ⓘ Frittura di pesce ⓘ Antipasto vegetariano ⓘ

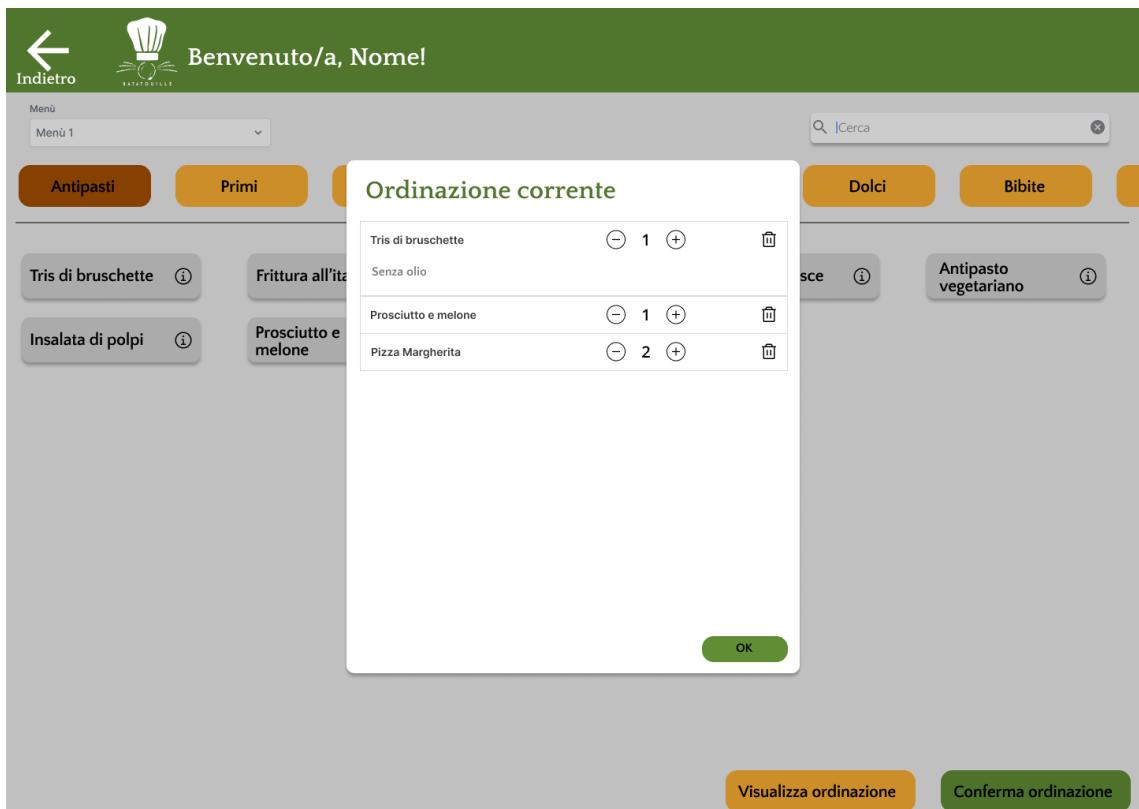
Insalata di polpi ⓘ Prosciutto e melone ⓘ

Visualizza ordinazione Conferma ordinazione

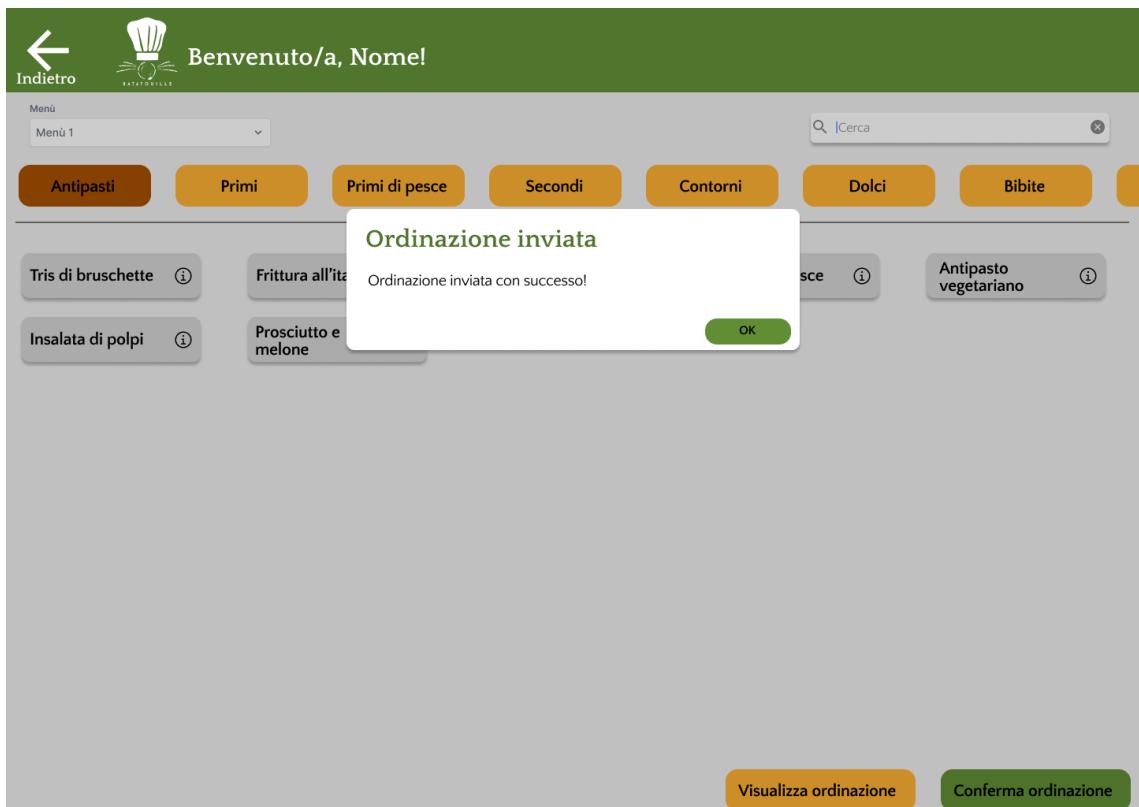
Mockup M4



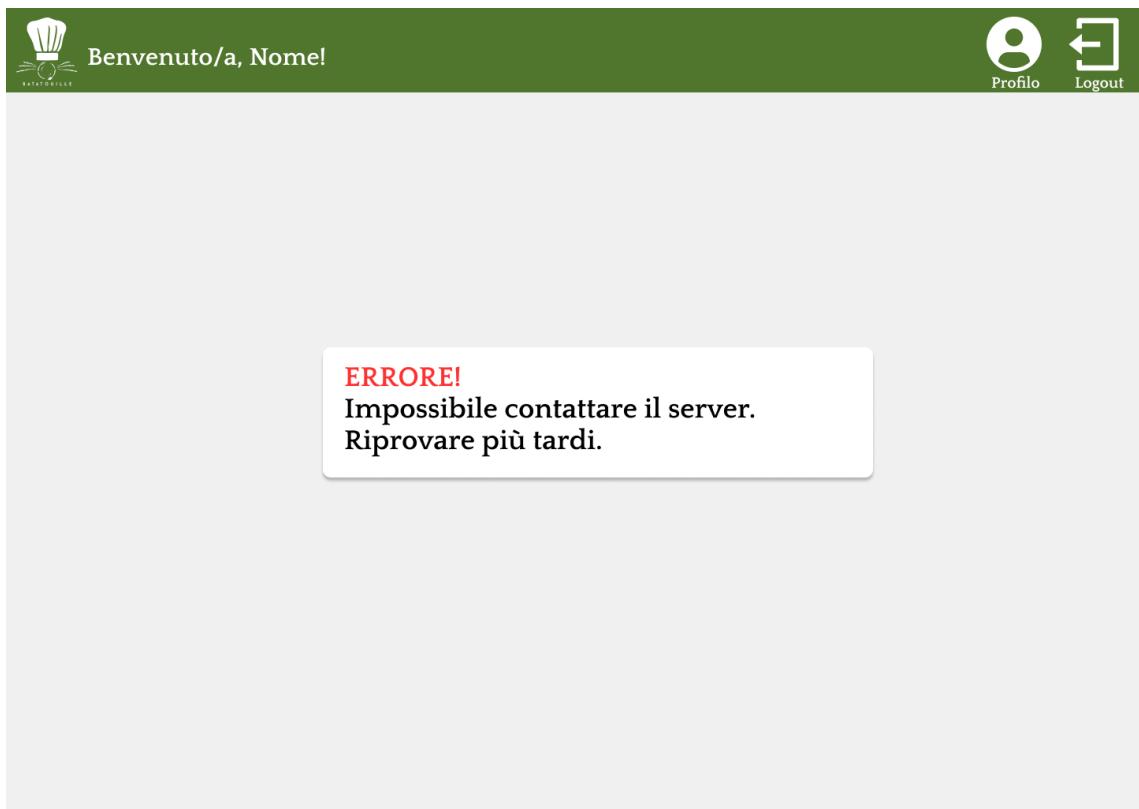
Mockup M5



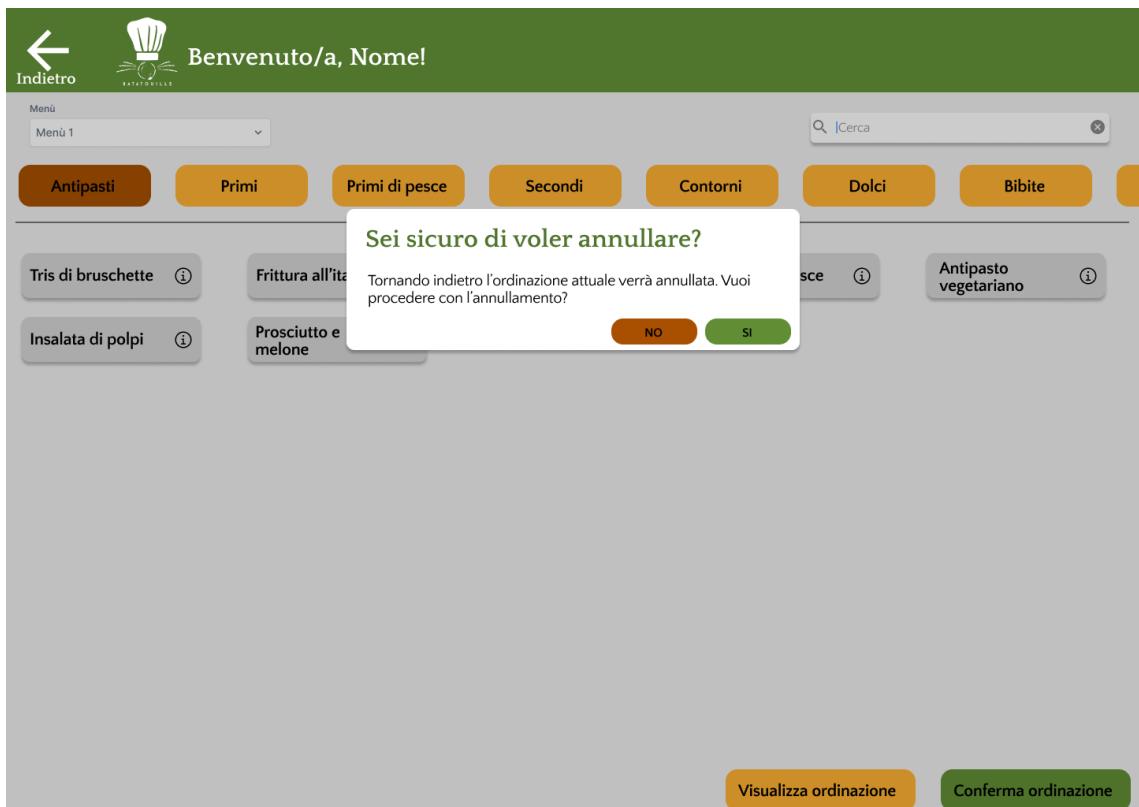
Mockup M6



Mockup M7



Mockup M8



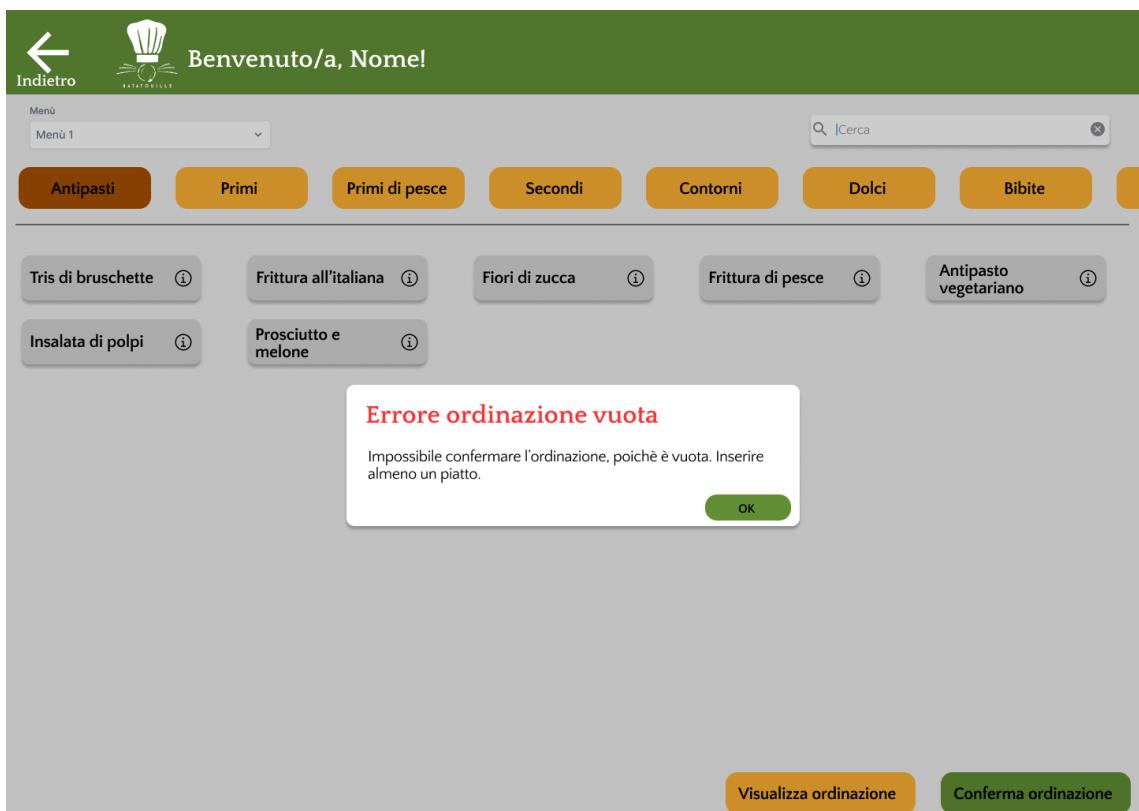
Mockup M9



Mockup M10



Mockup M11



Mockup M12

<b>Use Case #2</b>	Prendere ordinazioni		
<b>Goal in Context</b>	L'utente vuole prendere un'ordinazione.		
<b>Preconditions</b>	L'utente deve essere autenticato e deve essere l'amministratore o un addetto alla sala.		
<b>Success End Condition</b>	L'utente riesce a prendere l'ordinazione.		
<b>Failed End Condition</b>	L'utente non riesce a prendere l'ordinazione.		
<b>Primary actor</b>	Addetto alla sala, Amministratore		
<b>Trigger</b>	Dopo aver effettuato l'accesso, l'addetto visualizzerà in automatico la schermata dei tavoli. L'amministratore preme su "Tavoli" nella SideBar.		
<b>Description</b>	<b>Step</b>	<b>Addetto alla sala</b>	<b>Sistema</b>
	1		Mostra M1
	2	Preme un tavolo libero	
	3		Mostra M2
	4	Imposta gli ospiti e preme "Occupava Tavolo"	
	5		Aggiorna stato tavolo
	6		Mostra M1
	7	Preme sul tavolo appena occupato	
	8		Mostra M3
	9	Preme su "Nuova ordinazione"	
	10		Mostra M4
	11	Seleziona il menù	
	12		Ottieni categorie per il menù scelto
	13	Seleziona la categoria	
	14		Ottieni piatti per la categoria scelta
	15	Seleziona il piatto	
	16		Mostra M5
	17	Imposta le quantità ed eventuali note	
	18	Preme "Aggiungi"	
	19		Mostra M4
	20	Preme "Visualizza ordinazione"	
	21		Mostra M6
	22	Preme "OK"	

	23		Mostra M4
	24	Preme “Conferma ordinazione”	
	25		Mostra M7
	26	Preme “OK”	
	27		Invia ordinazione al server
	28		Mostra M1
<b>Extension #1</b>	<b>Step</b>	<b>Addetto alla sala</b>	<b>Sistema</b>
Il sistema non riesce a connettersi al server.	1.a		Mostra Popup di Errore (M8)
<b>Extension #2</b>	<b>Step</b>	<b>Addetto alla sala</b>	<b>Sistema</b>
L’utente vuole annullare l’ordinazione.	11.a	Preme “indietro”	
	12.a		Mostra M9
	13.a	Preme “SI”	
<b>Extension #3</b>	<b>Step</b>	<b>Addetto alla sala</b>	<b>Sistema</b>
L’utente seleziona un tavolo che ha già richiesto il conto.	2.a	Preme un tavolo con conto richiesto	
	3.a		Mostra M11
	4.a	Preme sul bottone “X”	
<b>Extension #4</b>	<b>Step</b>	<b>Addetto alla sala</b>	<b>Sistema</b>
L’utente invia un’ordinazione vuota.	15.a	Preme “Conferma ordinazione”	
	16.a		Mostra M12
<b>Subvariation #1</b>	<b>Step</b>	<b>Addetto alla sala</b>	<b>Sistema</b>
L’utente preme un tavolo già occupato.	2	Preme un tavolo occupato	
		Ritorna allo step 8 dello scenario principale.	
<b>Subvariation #2</b>	<b>Step</b>	<b>Addetto alla sala</b>	<b>Sistema</b>
L’utente vuole annullare l’inserimento di un piatto.	18	Preme “Annulla”	
		Ritorna allo step 10 dello scenario principale.	
<b>Subvariation #3</b>	<b>Step</b>	<b>Addetto alla sala</b>	<b>Sistema</b>
L’utente vuole modificare l’ordinazione prima di confermarla.	22	Modifica le quantità dei piatti oppure cancella un piatto	
	23	Preme “OK”	
		Ritorna allo step 23 dello scenario principale.	

<b>Subvariation #4</b>	<b>Step</b>	<b>Addetto alla sala</b>	<b>Sistema</b>
L'utente vuole visualizzare le informazioni di un piatto.	15	Preme sul bottone "info" di un piatto	
	16		Mostra M10
	17	Preme "OK"	
Ritorna allo step 15 dello scenario principale.			

## 2.3.3 Statistiche addetti alla sala

**Statistiche addetti alla sala**

**Ordini ed entrate totali**

Nome	Entrate	Ordini
Nome	3525	3525

**Ordini ed entrate giornaliere**

Addetto alla sala: Tutti

Data inizio: Sept 12, 2021 al Data fine: Oct 12, 2021

Y-axis: 10k, 30k, 50k, 100k

Legend: Entrate (Yellow), Ordini (Brown)

Mockup M1

**Statistiche addetti alla sala**

**Ordini ed entrate totali**

Nome	Entrate	Ordini
Nome		

**Ordini ed entrate giornaliere**

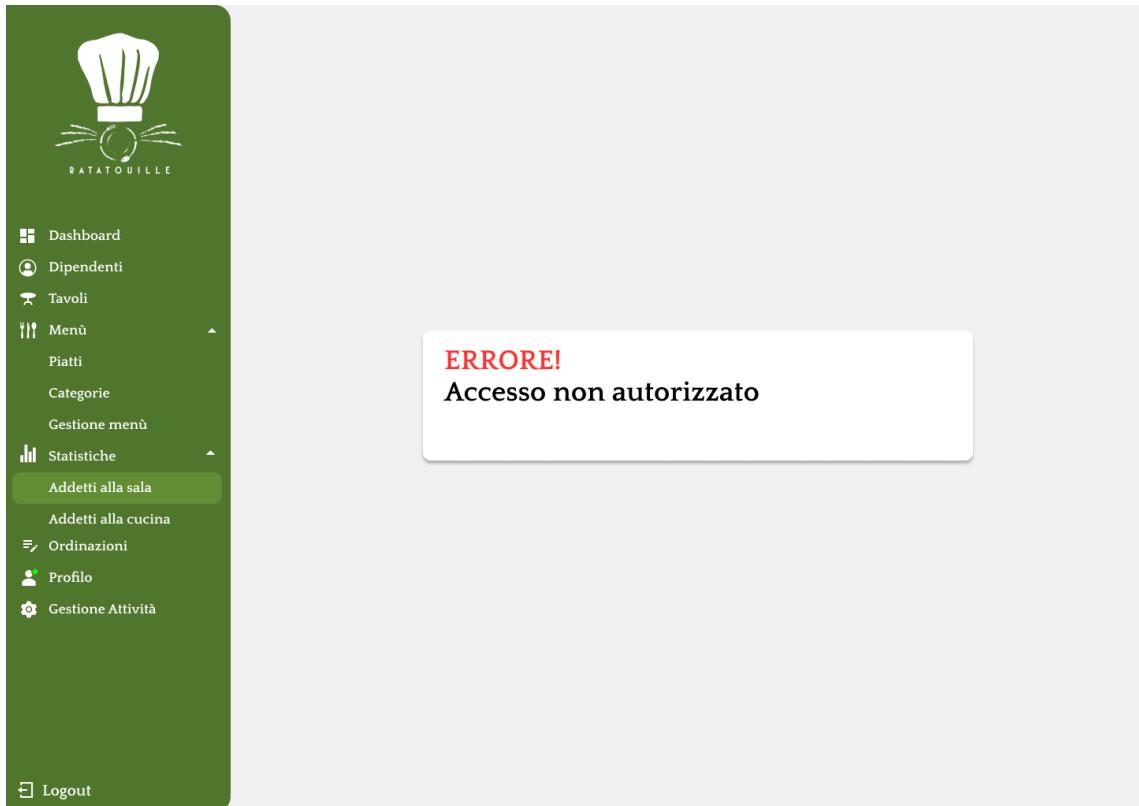
Addetto alla sala: Tutti

Data inizio: Sept 12, 2021 al Data fine: Oct 12, 2021

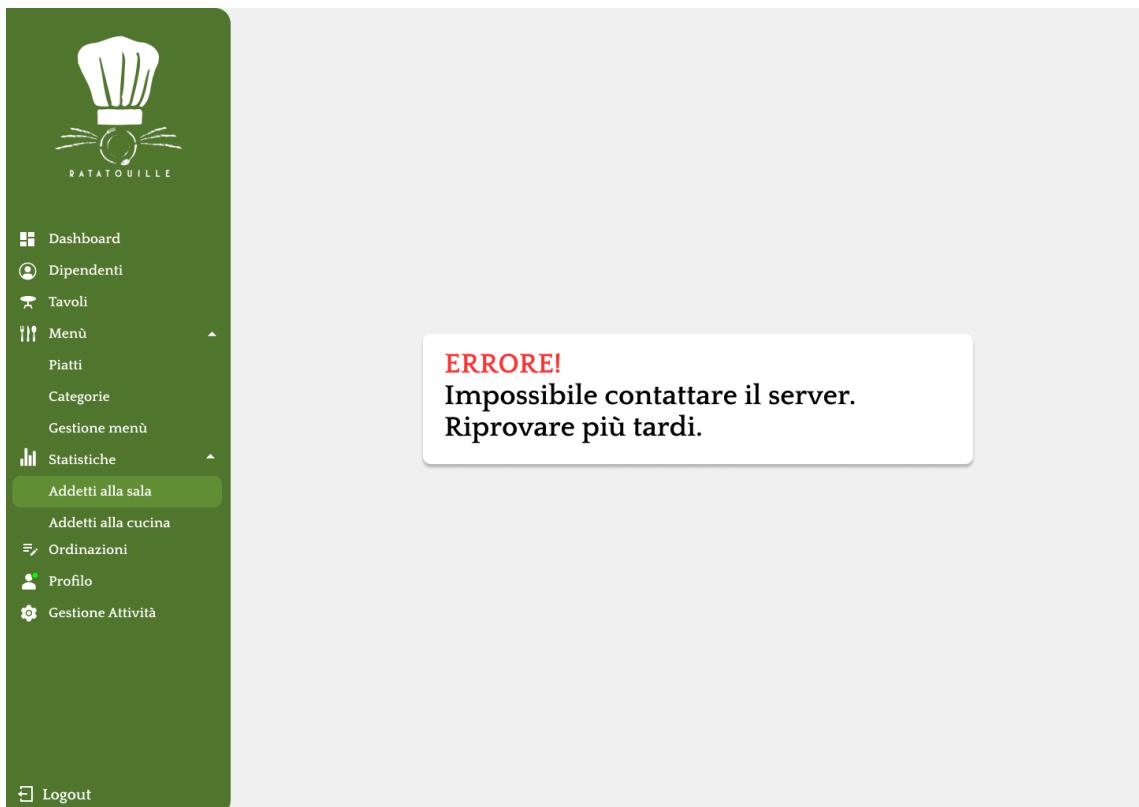
Y-axis: 10k, 30k, 50k, 100k

Legend: Entrate (Yellow), Ordini (Brown)

Mockup M2



Mockup M3



Mockup M4

<b>Use Case #3</b>	Visualizzare statistiche addetti alla sala		
<b>Goal in Context</b>	L'utente vuole visualizzare le statistiche degli addetti alla sala nell'ultimo mese		
<b>Preconditions</b>	L'utente deve essere autenticato e deve essere l'amministratore		
<b>Success End Condition</b>	L'utente riesce a visualizzare le statistiche		
<b>Failed End Condition</b>	L'utente non riesce a visualizzare le statistiche		
<b>Primary actor</b>	Amministratore		
<b>Trigger</b>	Dopo aver effettuato l'accesso, l'amministratore preme su "Statistiche addetti alla sala" nella SideBar		
<b>Description</b>	<b>Step</b>	<b>Amministratore</b>	<b>Sistema</b>
	1		Calcola ordini ed entrate di ogni dipendente nell'ultimo mese
	2		Calcola ordini ed entrate giornaliere di ogni dipendente nell'ultimo mese
	3		Mostra M1
<b>Extension #1</b> Il sistema non riesce a connettersi al server.	<b>Step</b>	<b>Amministratore</b>	<b>Sistema</b>
1.a			Mostra Popup di Errore (M4)
			Mostra M3
<b>Extension #2</b> L'utente non è l'amministratore.	<b>Step</b>	<b>Supervisore</b>	<b>Sistema</b>
<b>Extension #3</b> L'utente inserisce date non valide oppure non sono presenti statistiche nell'ultimo mese.	<b>Step</b>	<b>Amministratore</b>	<b>Sistema</b>
1.a			Mostra M2
	<b>Subvariation #1</b>	<b>Amministratore</b>	<b>Sistema</b>
	1	Cambia date del primo grafico	
	2		Calcola ordini ed entrate di ogni dipendente nelle date selezionate
Ritorna allo step 3 dello scenario principale.			

<b>Subvariation #2</b>	<b>Step</b>	<b>Amministratore</b>	<b>Sistema</b>
L'utente vuole cambiare le date del grafico ordini ed entrate giornaliere.	1	Cambia date del secondo grafico	
	2		Calcola ordini ed entrate giornaliere di ogni dipendente nelle date selezionate
Ritorna allo step 3 dello scenario principale.			

## 2.3.4 Visualizzazione ordinazioni (addetto alla cucina)

The interface displays two main sections: 'ORDINI DA EVADERE' (Pending Orders) and 'ORDINI EVASI' (Delivered Orders).

- ORDINI DA EVADERE:**
  - ORDINE N° 00000000** (TAVOLO N° 12) - Status: IN PREPARAZIONE. Contains: 1 Tris di bruschette (senza olio), 1 Tris di bruschette, 1 Spaghetti alla carbonara, 1 Spaghetti allo scoglio.
  - ORDINE N° 00000001** (TAVOLO N° 01) - Status: IN PREPARAZIONE. Contains: 1 Tris di bruschette, 2 Frittura all'italiana, 1 Pizza margherita, 1 Pizza marinara, 1 Pizza mimosa, 1 Pizza wurstel e patatine (rossa).
- ORDINI EVASI:**
  - ORDINE N° 00000002** (TAVOLO N° 09) - Status: EVASO. Contains: 1 Tris di bruschette, 2 Frittura all'italiana, 1 Pizza margherita, 1 Pizza marinara, 1 Pizza mimosa, 1 Pizza wurstel e patatine (rossa).
  - ORDINE N° 00000003** (TAVOLO N° 05) - Status: IN PREPARAZIONE. Contains: 1 Spaghetti alla carbonara, 1 Spaghetti allo scoglio.
  - ORDINE N° 00000004** (TAVOLO N° 08) - Status: IN PREPARAZIONE. Contains: 1 Spaghetti alla carbonara, 1 Spaghetti allo scoglio.

Mockup M1

A central error message box is displayed:

**ERRORE!**  
Impossibile contattare il server.  
Riprovarе più tardi.

Mockup M2

**ORDINI DA EVADERE**

**ORDINE N° 00000000  
TAVOLO N° 12**

- 1 Tris di bruschette
  - senza olio
- 1 Tris di bruschette
- 1 Spaghetti alla carbonara
- 1 Spaghetti allo scoglio

**ORDINE N° 00000001  
TAVOLO N° 01**

- 1 Tris di bruschette
- 2 Frittura all'italiana
- 1 Pizza margherita
- 1 Pizza marinara
- 1 Pizza mimosa
- 1 Pizza wurstel e patatine
  - rossa

**ORDINE N° 00000002  
TAVOLO N° 09**

- 1 Tris di bruschette
- 2 Frittura all'italiana
- 1 Pizza margherita
- 1 Pizza marinara
- 1 Pizza mimosa
- 1 Pizza wurstel e patatine
  - rossa

**ORDINE N° 00000003  
TAVOLO N° 05**

- 1 Spaghetti alla carbonara
- 1 Spaghetti allo scoglio

**ORDINE N° 00000004  
TAVOLO N° 08**

- 1 Spaghetti alla carbonara
- 1 Pizza margherita
  - con patate fritte
- 1 Pizza margherita
  - con prosciutto
- 1 Cacio e pepe
- 1 Coca cola 2L

**ORDINI EVASI**

Mockup M3

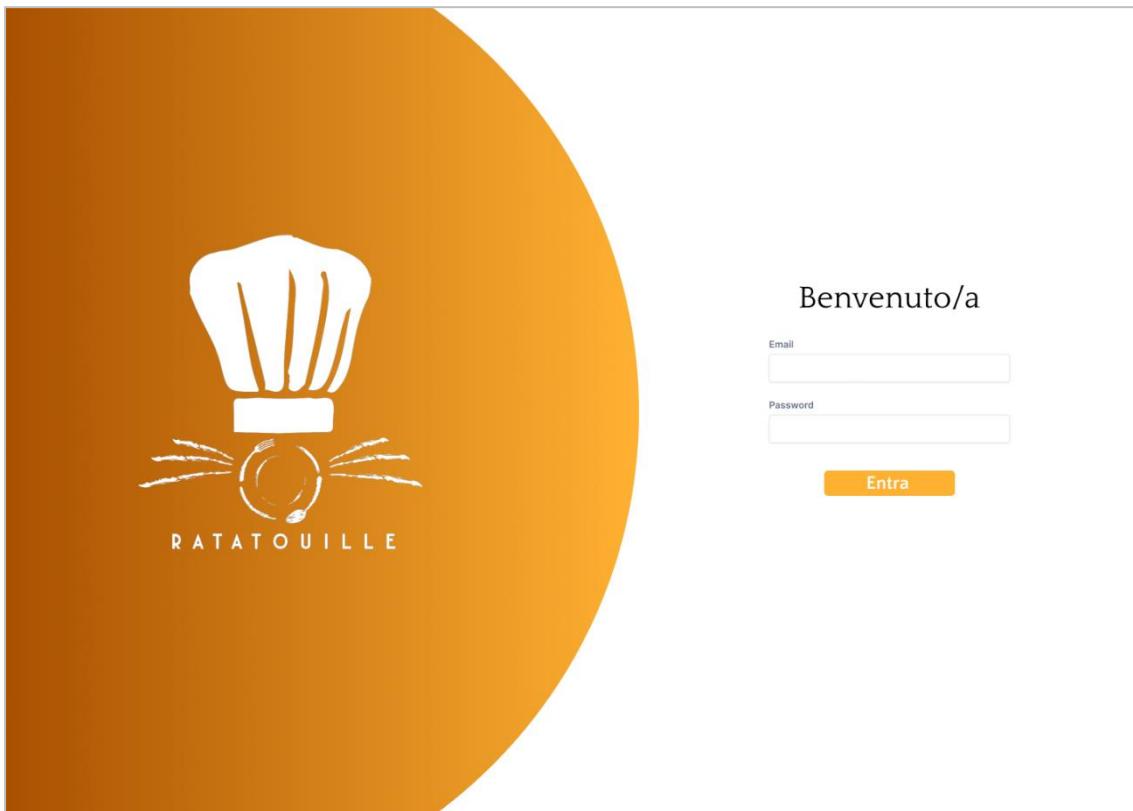
<b>Use Case #4</b>	Visualizzare ordinazioni		
<b>Goal in Context</b>	L'utente vuole visualizzare le ordinazioni in tempo reale		
<b>Preconditions</b>	L'utente deve essere autenticato e deve essere l'amministratore oppure un addetto alla cucina		
<b>Success End Condition</b>	L'utente riesce a visualizzare le ordinazioni		
<b>Failed End Condition</b>	L'utente non riesce a visualizzare le ordinazioni		
<b>Primary actor</b>	Addetto alla cucina, Amministratore		
<b>Trigger</b>	Dopo aver effettuato l'accesso, l'addetto visualizzerà in automatico la schermata delle ordinazioni. L'amministratore preme su "Ordinazioni" nella SideBar		
<b>Description</b>	<b>Step</b>	<b>Addetto alla cucina</b>	<b>Sistema</b>
	1		Trova le ordinazioni che ancora devono essere evase
	2		Mostra M1

<b>Extension #1</b>	<b>Step</b>	<b>Addetto alla cucina</b>	<b>Sistema</b>
Il sistema non riesce a connettersi al server.	1.a		Mostra Popup di Errore (M2)
<b>Subvariation #1</b>	<b>Step</b>	<b>Addetto alla cucina</b>	<b>Sistema</b>
L'utente vuole preparare un piatto	3	Preme su "Prepara"	
	4		Aggiorna stato piatto
	5		Aggiorna stato ordinazione
	Ritorna allo step 2 dello scenario principale.		
<b>Subvariation #2</b>	<b>Step</b>	<b>Addetto alla cucina</b>	<b>Sistema</b>
L'utente vuole evadere un piatto.	3	Preme su "Evadi"	
	4		Aggiorna stato piatto
	5		Aggiorna stato ordinazione
	Ritorna allo step 2 dello scenario principale.		
<b>Subvariation #3</b>	<b>Step</b>	<b>Addetto alla cucina</b>	<b>Sistema</b>
L'utente vuole visualizzare le ordinazioni evase.	3	Preme su "Ordini evasi"	
	4		Trova ordinazioni evase
	5		Mostra M3

## 2.4 Prototipazione visuale via Mock up

In questa sezione verrà descritta la prototipazione visuale via mock up. Ovvero, il mock up è una bozza del prodotto software, priva delle funzioni del prodotto finale, e utilizzato principalmente per scopi illustrativi.

### 2.4.1 Login



## 2.4.2 Mock up Amministratore

I seguenti mock up sono riferiti alle funzionalità dell'amministratore.

### Dashboard

Periodo di riferimento: mese corrente

Miglior addetto alla sala Nome	Miglior addetto alla cucina Nome	Ordinazioni Numero	Clienti Numero
-----------------------------------	-------------------------------------	-----------------------	-------------------

Entrate | Mese corrente Mese scorso

The chart displays daily entries over a two-week period. The brown line (Mese corrente) starts at approximately 10k on Monday, fluctuates between 10k and 20k, and ends at about 45k on Friday. The orange line (Mese scorso) follows a similar path but remains consistently lower, ending at approximately 35k on Friday.

Giorno	Mese corrente	Mese scorso
Mon	~10k	~10k
Tue	~12k	~12k
Wed	~15k	~15k
Thu	~18k	~18k
Fri	~20k	~20k
Sat	~25k	~25k
Sun	~28k	~28k
Mon	~30k	~30k
Tue	~32k	~32k
Wed	~35k	~35k
Thu	~38k	~38k
Fri	~40k	~38k

#### Piatti più venduti

Nome	Prezzo	Quantità	Totale
Pizza margherita	\$79.49	82	\$6,518.18
Risotto alla pescatora	\$128.50	37	\$4,754.50
Frittura all'italiana	\$39.99	64	\$2,559.36
Tris di bruschette	\$20.00	184	\$3,680.00
Tagliata di carne	\$28.49	64	\$1,965.81
Frittura di pesce	\$79.49	82	\$6,518.18

### Dipendenti

+ | 1 Selezionato | Elimina selezionato/i | Search

ID Utente	Utente	Ruolo
#CM9801	Katherine Moss	Amministratore
#CM9802	Koray Okumus	Amministratore
#CM9803	Lana Steiner	Supervisore
#CM9804	Natali Craig	Addetto alla cucina
<input checked="" type="checkbox"/> #CM9804	Orlando Diggs	Addetto alla sala

**Dipendenti**

**Aggiungi dipendente**

ID Utente	Utente	Ruolo
#1	Katherine Moss	Amministratore
#2	Koray Okumus	Supervisore
#3	Lana Steiner	Supervisore
#4	Natali Craig	Addetto alla cucina
<input checked="" type="checkbox"/> #5	Orlando Diggs	Addetto alla sala

Inserisci dipendente

**Tavoli**

- Libero
- Conto richiesto
- Occupato

1	2	3	4 Ospiti: 3	5 Ospiti: 3	6 Ospiti: 3
7 Ospiti: 3	8 Ospiti: 3	9 Ospiti: 3	10 Ospiti: 3		

**Tavoli**

- Libero
- Conto richiesto
- Occupato

Tavolo	Ospiti
1	0
2	0
3	0
4	3
5	3
6	3

**Nuovi ospiti**  
Hai selezionato il tavolo: 1  
Ospiti: 1

Annulla Occupava tavolo

**TAVOLO N° 5**

Ordine	Dettagli
1	Tris di bruschette senza olio
1	Tris di bruschette
1	Spaghetti alla carbonara
1	Spaghetti allo scoglio

**Tavoli**

- Libero
- Conto richiesto
- Occupato

Tavolo	Ospiti
1	0
2	0
3	0
4	3
5	3
6	3
7	3
8	3
9	3
10	3

**Nuovi ospiti**  
Hai selezionato il tavolo: 1  
Ospiti: 1

Annulla Occupava tavolo

**Nuova ordinazione**

**Richiedi conto**



## Tavoli

• Libero  
• Conto richiesto  
• Occupato

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b> Ospiti: 3
<b>5</b> Ospiti: 3	<b>6</b> Ospiti: 3	<b>7</b> Ospiti: 3	<b>8</b> Ospiti: 3
<b>9</b> Ospiti: 3	<b>10</b> Ospiti: 3		

[Stampa conto](#)



## Menù

Menù 1

Cerca

Antipasti      Primi      Primi di pesce      Secondi      Contorni      Dolci

Tris di bruschette	Frittura all'italiana	Fiori di zucca	Frittura di pesce
Insalata di polpi	Prosciutto e melone	Antipasto vegetariano	

[Indietro](#)      [Visualizza ordinazione](#)      [Conferma ordinazione](#)

The screenshot shows the Rataouille software interface. On the left is a green sidebar with a chef's hat icon and the text 'RATAOUILLE'. The sidebar contains the following menu items:

- Dashboard
- Dipendenti
- Tavoli
- Menù
- Piatti
- Categorie
- Gestione menù
- Statistiche
  - Addetti alla sala
  - Addetti alla cucina
- Ordinazioni
- Profilo
- Gestione Attività

Below the sidebar is a 'Logout' button.

The main area has a light gray background. At the top, there is a navigation bar with tabs: Antipasti, Primi, Primi di pesce, Secondi, Contorni, and Dolci. Below the tabs, several dish cards are displayed: Tris di bruschette, Frittura all'italiana, Fiori di zucca, Insalata di polpi, and Frittura di pesce. A search bar at the top right contains the placeholder 'Cerca'.

A central modal dialog is open, titled 'Aggiungi piatto a ordinazione'. It displays the dish 'Tris di bruschette' and a quantity input field set to '1'. There is a note input field below the quantity. At the bottom of the dialog are two buttons: 'Annulla' and 'Aggiungi'.

At the bottom of the main area are three buttons: 'Indietro', 'Visualizza ordinazione' (orange), and 'Conferma ordinazione' (green).

This screenshot shows the same Rataouille software interface as the previous one, but the central modal dialog is now displaying detailed information about the 'Tris di bruschette' dish.

The modal title is 'Tris di bruschette'. The description text reads: 'Bruschetta con pomodori gialli del Piennolo, bruschetta con pesto e pomodorini, bruschetta con pomodori rossi di San Marzano.' Below the description is a note: 'Allergeni: glutine'. At the bottom right of the modal is an 'OK' button.

The rest of the interface is identical to the first screenshot, including the sidebar, navigation bar, and bottom buttons.

The screenshot shows the Ratatouille management interface. On the left is a dark green sidebar with a chef's hat icon and the word "RATATOUILLE". The main menu items are: Dashboard, Dipendenti, Tavoli (selected), Menù, Piatti, Categorie, Gestione menù, Statistiche, Addetti alla sala, Addetti alla cucina, Ordinazioni, Profilo, and Gestione Attività. Below these are Logout, Indietro, Visualizza ordinazione, and Conferma ordinazione buttons.

**Menù**  
Menù 1

**Antipasti**   **Primi**   **Primi di pesce**   **Secondi**   **Contorni**   **Dolci**

**Ordinazione corrente**

Tris di bruschette	1	+	Chiudi
Senza olio			
Prosciutto e melone	1	+	Chiudi
Pizza Margherita			
Pizza Margherita	2	+	Chiudi

**Frittura di pesce**

**OK**

This screenshot shows the same interface as above, but the order window is now closed. The main menu items are: Dashboard, Dipendenti, Tavoli (selected), Menù, Piatti, Categorie, Gestione menù, Statistiche, Addetti alla sala, Addetti alla cucina, Ordinazioni, Profilo, and Gestione Attività. Below these are Logout, Indietro, Visualizza ordinazione, and Conferma ordinazione buttons.

**Menù**  
Menù 1

**Tris di carpaccio di pesce**   **Frittura di pesce**   **Filetto di pesce spada alla griglia**

**Indietro**   **Visualizza ordinazione**   **Conferma ordinazione**



RATATOUILLE

- Dashboard
- Dipendenti
- Tavoli
- Menù
- Piatti
- Categorie
- Gestione menù
- Statistiche**
- Addetti alla sala
- Addetti alla cucina
- Ordinazioni
- Profilo
- Gestione Attività

[Logout](#)

## Statistiche addetti alla sala

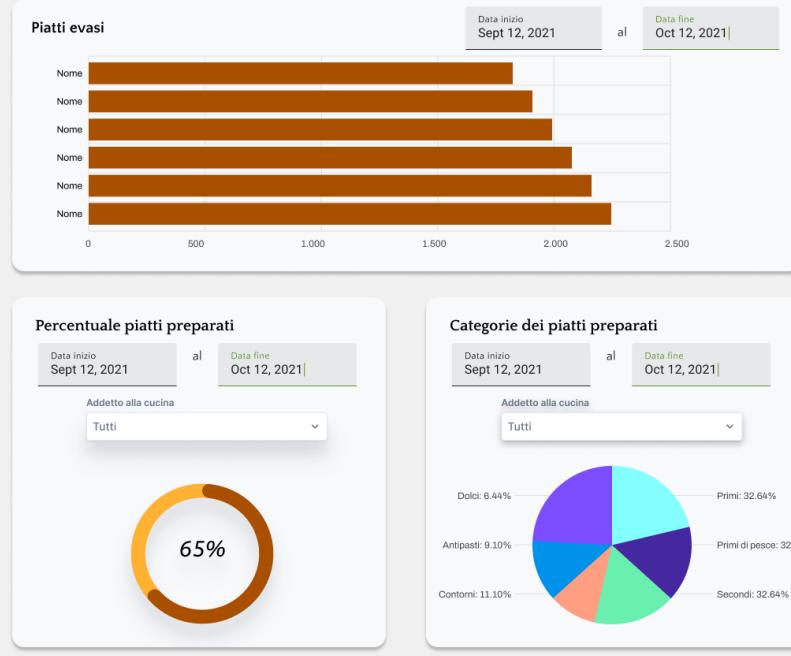



RATATOUILLE

- Dashboard
- Dipendenti
- Tavoli
- Menù
- Piatti
- Categorie
- Gestione menù
- Statistiche**
- Addetti alla cucina**
- Ordinazioni
- Profilo
- Gestione Attività

[Logout](#)

## Statistiche addetti alla cucina





**RATATOUILLE**

- [Dashboard](#)
- [Dipendenti](#)
- [Tavoli](#)
- [Menù](#)
  - [Piatti](#)
  - [Categorie](#)
  - [Gestione menù](#)
- [Statistiche](#)
  - [Addetti alla sala](#)
  - [Addetti alla cucina](#)
- [Ordinazioni](#)
- [Profilo](#)
- [Gestione Attività](#)

[Logout](#)

ORDINI DA EVADERE	ORDINI EVASI
<b>ORDINE N° 00000000</b> <b>TAVOLO N° 12</b> <hr/> 1 Tris di bruschette <ul style="list-style-type: none"> <li>· senza olio</li> </ul> 1 Tris di bruschette 1 Spaghetti alla carbonara 1 Spaghetti allo scoglio	<b>00:01:20</b> <a href="#" style="color: green; background-color: #2e3436; padding: 2px 10px; border-radius: 5px; text-decoration: none; font-weight: bold;">EVADI</a>  <b>IN PREPARAZIONE</b> <a href="#" style="color: orange; background-color: #ffccbc; padding: 2px 10px; border-radius: 5px; text-decoration: none; font-weight: bold;">PREPARA</a> <a href="#" style="color: orange; background-color: #ffccbc; padding: 2px 10px; border-radius: 5px; text-decoration: none; font-weight: bold;">PREPARA</a>
<b>ORDINE N° 00000001</b> <b>TAVOLO N° 01</b> <hr/> 1 Tris di bruschette 2 Frittura all'italiana 1 Pizza margherita 1 Pizza marinara 1 Pizza mimosa 1 Pizza wurstel e patatine <ul style="list-style-type: none"> <li>· rossa</li> </ul>	<b>00:01:15</b> <a href="#" style="color: grey; background-color: #d3d3d3; padding: 2px 10px; border-radius: 5px; text-decoration: none; font-weight: bold;">EVASO</a> <a href="#" style="color: grey; background-color: #d3d3d3; padding: 2px 10px; border-radius: 5px; text-decoration: none; font-weight: bold;">EVASO</a> <a href="#" style="color: orange; background-color: #ffccbc; padding: 2px 10px; border-radius: 5px; text-decoration: none; font-weight: bold;">PREPARA</a> <a href="#" style="color: orange; background-color: #ffccbc; padding: 2px 10px; border-radius: 5px; text-decoration: none; font-weight: bold;">PREPARA</a> <a href="#" style="color: orange; background-color: #ffccbc; padding: 2px 10px; border-radius: 5px; text-decoration: none; font-weight: bold;">PREPARA</a> <b>IN PREPARAZIONE</b>
<b>ORDINE N° 00000002</b> <b>TAVOLO N° 00</b>	<b>00:01:01</b>



**RATATOUILLE**

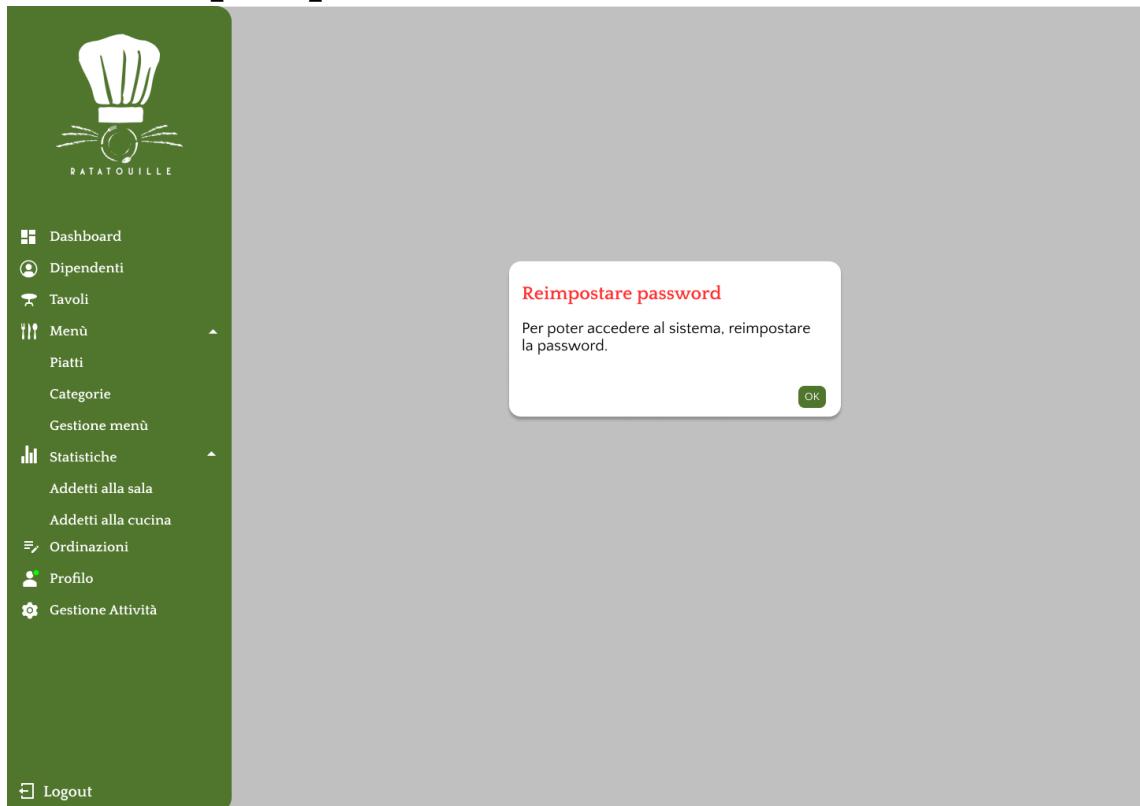
- [Dashboard](#)
- [Dipendenti](#)
- [Tavoli](#)
- [Menù](#)
  - [Piatti](#)
  - [Categorie](#)
  - [Gestione menù](#)
- [Statistiche](#)
  - [Addetti alla sala](#)
  - [Addetti alla cucina](#)
- [Ordinazioni](#)
- [Profilo](#)
- [Gestione Attività](#)

[Logout](#)

ORDINI DA EVADERE	ORDINI EVASI
<b>ORDINE N° 00000000</b> <b>TAVOLO N° 12</b> <hr/> 1 Tris di bruschette <ul style="list-style-type: none"> <li>· senza olio</li> </ul> 1 Tris di bruschette 1 Spaghetti alla carbonara 1 Spaghetti allo scoglio	<b>ORDINE N° 00000002</b> <b>TAVOLO N° 09</b> <hr/> 1 Tris di bruschette 2 Frittura all'italiana 1 Pizza margherita 1 Pizza marinara 1 Pizza mimosa 1 Pizza wurstel e patatine <ul style="list-style-type: none"> <li>· rossa</li> </ul>
<b>ORDINE N° 00000001</b> <b>TAVOLO N° 01</b> <hr/> 1 Tris di bruschette 2 Frittura all'italiana 1 Pizza margherita 1 Pizza marinara 1 Pizza mimosa 1 Pizza wurstel e patatine <ul style="list-style-type: none"> <li>· rossa</li> </ul>	<b>ORDINE N° 00000003</b> <b>TAVOLO N° 05</b> <hr/> 1 Spaghetti alla carbonara 1 Spaghetti allo scoglio
<b>ORDINE N° 00000004</b> <b>TAVOLO N° 08</b> <hr/> 1 Spaghetti alla carbonara	

The screenshot shows the 'Gestione Attività' (Activity Management) section of the Ratatouille application. On the left, a dark green sidebar menu lists various administrative functions: Dashboard, Dipendenti, Tavoli, Menù, Piatti, Categorie, Gestione menù, Statistiche, Addetti alla sala, Addetti alla cucina, Ordinazioni, Profilo, and Gestione Attività. The 'Gestione Attività' item is highlighted with a green background. At the bottom of the sidebar is a 'Logout' button. The main content area has a light gray background and features a title 'Gestione Attività' and a subtitle 'Informazioni attività'. It includes fields for 'Nome attività' (Text input), 'P.IVA' (Text input), and 'Indirizzo' (Text input). Below these is a dropdown menu for 'Numero tavoli' (1 to 5). A prominent orange button labeled 'Salva modifiche' (Save changes) is located at the bottom right of the form.

### 2.4.3 Mock up Supervisore





I seguenti mock up raffigurano funzionalità comuni sia al ruolo di Amministratore che a quello di Supervisore.

ID Piatto	Nome piatto	Categoria	Descrizione	Costo	Modifica
#1	Frittura all'italiana	Antipasto	Text	€10,00	<a href="#">Modifica</a>
#2	Tris di bruschette	Antipasto	Text	€10,00	<a href="#">Modifica</a>
#3	Risotto alla pescatora	Primo di pesce	Text	€10,00	<a href="#">Modifica</a>
#4	Frittura di pesce	Secondo	Text	€10,00	<a href="#">Modifica</a>
<input checked="" type="checkbox"/> #5	Cheesecake	Dolce	Text	€10,00	<a href="#">Modifica</a>

Nome piatto

Costo

Categoria

Descrizione

Allergeni

- Frutta a guscio
- Soia
- Lupino
- Semi di sesamo
- Latte
- Molluschi
- Uova
- Crostacei
- Glutine
- Pesci
- Sedano
- Senape
- Anidride solforosa e solfiti
- Arachidi

[Aggiungi piatto](#)

**Piatti**

**Modifica piatto**

Nome piatto	Costo	Categoria
Text	\$ 0.00	Antipasto

**Descrizione**

**Allergeni**

<input type="checkbox"/> Frutta a guscio	<input type="checkbox"/> Soia	<input type="checkbox"/> Lupino
<input type="checkbox"/> Semi di sesamo	<input type="checkbox"/> Latte	<input type="checkbox"/> Molluschi
<input type="checkbox"/> Uova	<input type="checkbox"/> Crostacei	<input type="checkbox"/> Glutine
<input type="checkbox"/> Pesce	<input type="checkbox"/> Sedano	<input type="checkbox"/> Senape
<input type="checkbox"/> Anidride solforosa e solfiti		

**Modifica piatto**

**Search**

ID Piatto	Nome piatto	Categoria	Descrizione	Costo	Azione
#1	Frittura all'italiana	Antipasto	Text	€10,00	Modifica ⓘ
#2	Tris di bruschette	Antipasto	Text	€10,00	Modifica ⓘ
#3	Risotto alla pescatora	Primo di pesce	Text	€10,00	Modifica ⓘ
#4	Frittura di pesce	Secondo	Text	€10,00	Modifica ⓘ
<input checked="" type="checkbox"/> #5	Cheesecake	Dolce	Text	€10,00	Modifica ⓘ

**Logout**

**Categorie**

**Primi piatti**

**Antipasti**

Katherine Moss	<b>Elimina</b>
Koray Okumus	<b>Elimina</b>

**Logout**

The screenshot shows the Rataouille application's interface. On the left is a dark green sidebar with a chef's hat icon and the word "RATAOUILLE". The sidebar contains the following navigation items:

- Dashboard
- Dipendenti
- Tavoli
- Menù
- Piatti
- Categorie
- Gestione menù
- Statistiche
  - Addetti alla sala
  - Addetti alla cucina
- Ordinazioni
- Profilo
- Gestione Attività

At the bottom of the sidebar is a "Logout" button.

The main content area has a light gray background. At the top, the word "Categorie" is displayed. Below it is a modal window titled "Aggiungi categoria" (Add category). The modal contains a "Nome categoria" input field with "Text" typed into it, and a "Menu" section with four checkboxes:

- Menù 1
- Menù 2
- Menù 3
- Menù 4

At the bottom right of the modal is a green "Aggiungi categoria" button.

Below the modal is a list of categories:

- Primi piatti
- Antipasti

Under "Primi piatti", there are two entries: "Katherine Moss" and "Koray Okumus", each with a red "Elimina" (Delete) link to its right.

At the top right of the main content area is a search bar with the placeholder "Q. Search" and a dropdown menu with the option "Modifica".

This screenshot is identical to the one above, showing the "Categorie" screen with the "Aggiungi categoria" modal open. The only difference is that the "Modifica categoria" button at the bottom of the modal is now highlighted in green, indicating it has been clicked.

The screenshot shows the 'Gestione menù' (Menu Management) screen. On the left is a dark green sidebar with a chef's hat icon and the word 'RATAOUILLE'. The main area has a light gray background with the title 'Gestione menù' at the top. Below it is a table with three rows:

	1 Selezionato	Elimina selezionato(i)
<input checked="" type="checkbox"/> Menu generale		Disabilita <input checked="" type="button"/> Abilita
<input type="checkbox"/> Menu estivo		Disabilita <input type="button"/> Abilita
<input type="checkbox"/> Menu pizze		Disabilita <input type="button"/> Abilita

At the bottom left of the sidebar is a 'Logout' button.

The screenshot shows the 'Gestione menù' (Menu Management) screen with an 'Aggiungi menù' (Add Menu) dialog box open. The dialog box has a title 'Aggiungi menù' and a sub-section 'Nome menù' with a text input field containing 'Text'. At the bottom right of the dialog box is a green button labeled 'Aggiungi menù'.

Below the dialog box is the same table as in the first screenshot:

	1 Selezionato	Elimina selezionato(i)
<input checked="" type="checkbox"/> Menu generale		Disabilita <input checked="" type="button"/> Abilita
<input type="checkbox"/> Menu estivo		Disabilita <input type="button"/> Abilita
<input type="checkbox"/> Menu pizze		Disabilita <input type="button"/> Abilita

At the bottom left of the sidebar is a 'Logout' button.



**Gestione menù**

Per aggiungere, eliminare o modificare una categoria clicca **QUI**.

**Menù 1**

Categoria	Disabilita	Abilita
Antipasti	<input checked="" type="checkbox"/>	Abilita
Frittura all'italiana croccchè, arancini, zeppole, polenta	<input checked="" type="checkbox"/>	Abilita
Tris di bruschette bruschetta con pomodorini gialli del Piennolo, bruschetta con pesto e pomodorini, bruschetta con pomodori rossi di san Marzano	<input checked="" type="checkbox"/>	Abilita
Primi piatti	<input checked="" type="checkbox"/>	Abilita
Primi piatti di pesce	<input type="checkbox"/>	Abilita
Contorni	<input checked="" type="checkbox"/>	Abilita
Dolci	<input checked="" type="checkbox"/>	Abilita

[Torna a Selezione menù](#)

[Salva modifiche](#)

[Logout](#)



**Modifica profilo**

Email

Password

[Salva modifiche](#)

[Logout](#)

## 2.4.3 Mock up Addetto alla sala

The screenshot shows the 'Tavoli' (Tables) section of the application. At the top, there is a navigation bar with a logo, the text 'Benvenuto/a, Nome!', a 'Profilo' (Profile) icon, and a 'Logout' button. Below the navigation bar, the title 'Tavoli' is displayed. A legend indicates three states: 'Libero' (Green), 'Conto richiesto' (Orange), and 'Occupato' (Red). The tables are represented by numbered boxes: 1, 2, 3, 4 (Ospiti: 3), 5, 6, 7, 8 (Ospiti: 3), 9 (Ospiti: 3), and 10 (Ospiti: 3). Tables 1, 2, and 3 are green (Libero). Tables 4, 5, 6, 7, 8, 9, and 10 are red (Occupato).

The screenshot shows the 'Tavoli' (Tables) section with a confirmation dialog box overlaid. The dialog box has a title 'Nuovi ospiti' (New guests) and a message 'Hai selezionato il tavolo: 1'. It contains a numeric input field with the value '1' and a '+' button. There are 'Annulla' (Cancel) and 'Occupava tavolo' (Book table) buttons. The background shows the same table status as the previous screenshot, with Table 1 now highlighted in orange (Conto richiesto).



## Tavoli

- Libero
- Conto richiesto
- Occupato



Indietro

Benvenuto/a, Nome!

Menù

Menù 1

Cerca

Antipasti      Primi      Primi di pesce      Secondi      Contorni      Dolci      Bibite

Tris di bruschette ⓘ      Frittura all'italiana ⓘ      Fiori di zucca ⓘ      Frittura di pesce ⓘ      Antipasto vegetariano ⓘ

Insalata di polpi ⓘ      Prosciutto e melone ⓘ

Visualizza ordinazione      Conferma ordinazione

Indietro Benvenuto/a, Nome!

Menù  
Menù 1

Antipasti Primi Dolci Bibite

Tris di bruschette Frittura all'italiana Insalata di polpi Prosciutto e melone

Antipasto vegetariano

**Tris di bruschette**

Bruschetta con pomodori gialli del Piennolo, bruschetta con pesto e pomodorini, bruschetta con pomodori rossi di San Marzano.

Allergeni: glutine

OK

Visualizza ordinazione Conferma ordinazione

This screenshot shows a detailed description of the 'Tris di bruschette' dish. The dish consists of three types of bruschetta: yellow tomatoes from Piennolo, pesto and cherry tomatoes, and red tomatoes from San Marzano. It also includes an allergen note indicating the presence of gluten.

Indietro Benvenuto/a, Nome!

Menù  
Menù 1

Antipasti Primi Dolci Bibite

Tris di bruschette Frittura all'italiana Insalata di polpi Prosciutto e melone

Antipasto vegetariano

**Aggiungi piatto a ordinazione**

Piatto: Tris di bruschette

Quantità: 1

Note

Aggiungi

Visualizza ordinazione Conferma ordinazione

This screenshot shows the process of adding the 'Tris di bruschette' dish to an order. The user has selected the dish and specified a quantity of 1. There is a note field available for additional instructions, which is currently empty. The 'Aggiungi' (Add) button is visible at the bottom right of the modal.

Indietro Benvenuto/a, Nome!

Menù  
Menù 1

Cerca

Antipasti Primi Dolci Bibite

Tris di bruschette (i) Frittura all'italiana (i) Antipasto vegetariano (i)

Insalata di polpi (i) Prosciutto e melone (i)

Ordinazione corrente

Tris di bruschette	(-) 1 (+)
Senza olio	
Prosciutto e melone	(-) 1 (+)
Pizza Margherita	(-) 2 (+)

OK

Visualizza ordinazione Conferma ordinazione

Indietro Benvenuto/a, Nome!

Menù  
Menù 1

Cerca

Frittura di pesce (i) Filetto di pesce spada alla griglia (i)

Tris di carpaccio di pesce (i)

Visualizza ordinazione Conferma ordinazione

## 2.4.4 Mock up addetto alla cucina

**BENVENUTO/A, NOME!**

**ORDINI DA EVADERE**

**ORDINI EVASI**

ORDINE N°	TAVOLO N°	TEMPO
00000000	12	00:01:20
00000001	01	00:01:15
00000002	09	00:01:01
00000003	05	00:00:53
00000004	08	00:00:31

**ORDINE N° 00000000 TAVOLO N° 12**

- 1 Tris di bruschette
  - senza olio
- 1 Tris di bruschette
- 1 Spaghetti alla carbonara
- 1 Spaghetti allo scoglio

**EVADI**

**IN PREPARAZIONE**

**ORDINE N° 00000001 TAVOLO N° 01**

- 1 Tris di bruschette
- 2 Frittura all'italiana
- 1 Pizza margherita
- 1 Pizza marinara
- 1 Pizza mimosa
- 1 Pizza wurstel e patatine
  - rossa

**EVASO**

**EVASO**

**PREPARA**

**PREPARA**

**PREPARA**

**IN PREPARAZIONE**

**ORDINE N° 00000002 TAVOLO N° 09**

- 1 Tris di bruschette
- 2 Frittura all'italiana
- 1 Pizza margherita
- 1 Pizza marinara
- 1 Pizza mimosa
- 1 Pizza wurstel e patatine

**EVASO**

**IN PREPARAZIONE**

**IN PREPARAZIONE**

**EVASI**

**EVASI**

**PREPARA**

**ORDINE N° 00000003 TAVOLO N° 05**

- 1 Spaghetti alla carbonara
- 1 Spaghetti allo scoglio

**IN PREPARAZIONE**

**IN PREPARAZIONE**

**ORDINE N° 00000004 TAVOLO N° 08**

**BENVENUTO/A, NOME!**

**ORDINI DA EVADERE**

**ORDINI EVASI**

ORDINE N°	TAVOLO N°
00000000	12
00000002	09
00000004	08
00000001	01
00000003	05

**ORDINE N° 00000000 TAVOLO N° 12**

- 1 Tris di bruschette
  - senza olio
- 1 Tris di bruschette
- 1 Spaghetti alla carbonara
- 1 Spaghetti allo scoglio

**ORDINE N° 00000002 TAVOLO N° 09**

- 1 Tris di bruschette
- 2 Frittura all'italiana
- 1 Pizza margherita
- 1 Pizza marinara
- 1 Pizza mimosa
- 1 Pizza wurstel e patatine
  - rossa

**ORDINE N° 00000004 TAVOLO N° 08**

- 1 Spaghetti alla carbonara
- 1 Pizza margherita
  - con patate fritte
- 1 Pizza margherita
  - con prosciutto
- 1 Cacio e pepe
- 1 Coca cola 2L

**ORDINE N° 00000001 TAVOLO N° 01**

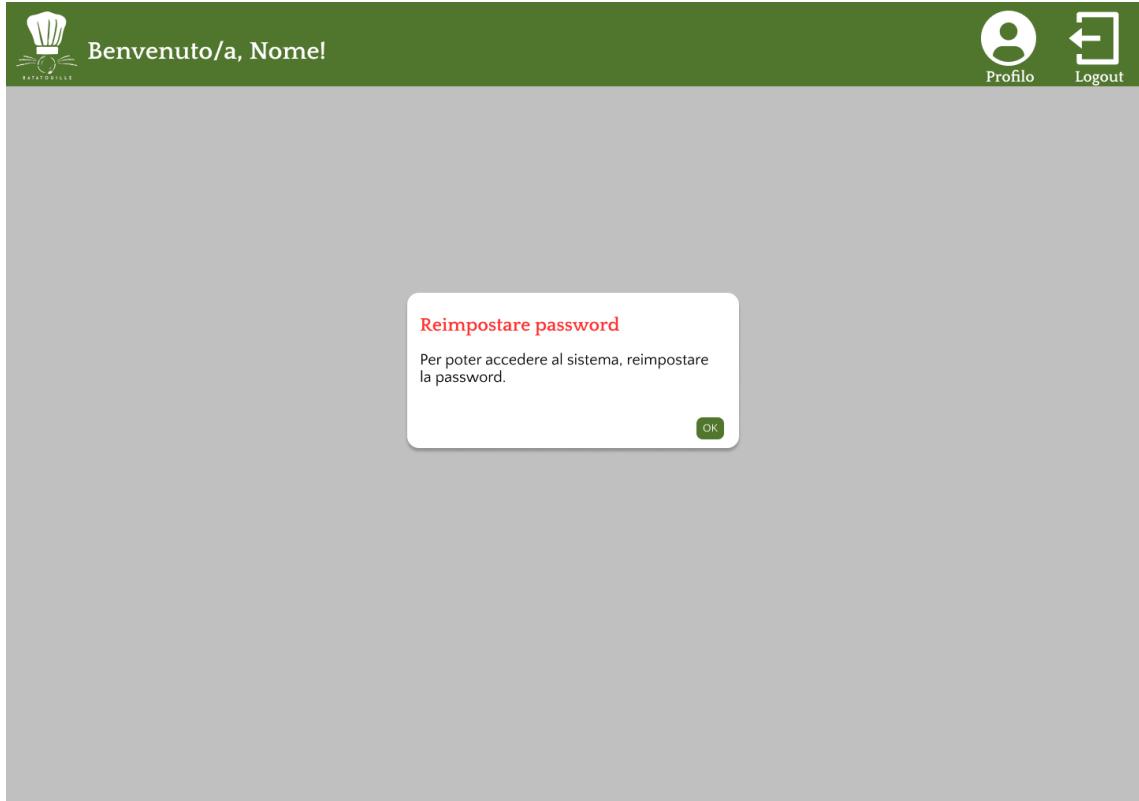
- 1 Tris di bruschette
- 2 Frittura all'italiana
- 1 Pizza margherita
- 1 Pizza marinara
- 1 Pizza mimosa
- 1 Pizza wurstel e patatine
  - rossa

**ORDINE N° 00000003 TAVOLO N° 05**

- 1 Spaghetti alla carbonara
- 1 Spaghetti allo scoglio

Il seguente mock up raffigura una schermata comune sia all'addetto alla sala che all'addetto alla cucina.

The mockup shows a top navigation bar with a back arrow, a chef icon, and the text "Benvenuto/a, Nome!". Below this, the title "Modifica profilo" is displayed. There are two input fields: "Email" and "Password", each with a placeholder "Text". A yellow "Salva modifiche" button is located at the bottom left of the form area.





### Reimposta password

Password

Text

Conferma Password

Text

**Salva password**

## 2.4.5 Popup

In questa sezione sono inseriti vari Popup che potrebbero apparire durante l'uso dell'applicativo.

### Errore caratteri password

Inserire una password con almeno 8 caratteri.

**OK**

### Errore categoria già presente

Questa categoria è già presente nei menù selezionati. Cambiare nome della categoria oppure cambiare menù.

**OK**

### Conferma eliminazione

Sei sicuro di voler eliminare questa categoria?

SI

NO

### Conferma eliminazione

Vuoi davvero eliminare gli utenti selezionati?

Annulla

Conferma eliminazione

### Conferma eliminazione

Sei sicuro di voler eliminare questo piatto?

SI

NO

### Conferma stampa conto

Vuoi davvero stampare il conto per il tavolo selezionato? Questa operazione libererà il tavolo

Annulla

Stampa conto

### Errore email già esistente

Esiste già un dipendente con questa email.  
Riprovare con una nuova email.

OK

### Sei sicuro di voler annullare?

Tornando indietro l'ordinazione attuale verrà annullata. Vuoi procedere con l'annullamento?

NO

SI

### Errore ordinazione vuota

Impossibile confermare l'ordinazione, poichè è vuota. Inserire almeno un piatto.

OK

### ERRORE!

Impossibile contattare il server.  
Riprovare più tardi.

### Errore piatto già presente

Questo piatto è già presente nella categoria selezionata. Rinominare il piatto o cambiare categoria.

OK

### Ordinazione inviata

Ordinazione inviata con successo!

OK

### Sei sicuro di voler tornare indietro?

Se torni indietro, le modifiche non salvate non verranno applicate!

SI

NO

### Errore selezione menù

Selezionare almeno un menù!

OK

### Inserimento riuscito

Inserimento avvenuto con successo!

OK

### Modifiche effettuate

Modifiche effettuate con successo!

OK

### Errore menù già esistente

Esiste già un menù con questo nome.  
Riprovare con un nome diverso.

OK

### Password non corrispondenti

Le password non corrispondono, riprovare.

OK

**ERRORE!**

**Accesso non autorizzato**

## 2.5 Individuazione target utenti

Individuare con precisione il target di riferimento rappresenta il punto di partenza di ogni vincente modello di business e analisi di mercato. Conoscere chi sono i destinatari del prodotto o servizio ti dà la possibilità di affinare nel migliore dei modi la tua offerta, rivolgendola a necessità, interessi e desideri specifici e promuovendola utilizzando i messaggi e i canali più efficaci per quel preciso gruppo di consumatori.

Uno dei principali strumenti utilizzati per identificare il target di riferimento è la cosiddetta segmentazione del mercato. Esistono diverse tecniche per raggruppare i consumatori allo scopo di individuare il miglior “bersaglio da colpire”. Una di esse, in particolare, suddivide il mercato in quattro gruppi omogenei, identificati attraverso caratteristiche comuni che fanno riferimento ai seguenti fattori: demografici (età, etnia, genere, ...), geografici (luogo, clima, ...), psicografici (interessi, valori, preferenze, ...) e comportamentali (comportamento in fase di acquisto/utilizzo del prodotto).

Per raccogliere informazioni sul mercato target è necessario affidarsi a dati attendibili e aggiornati, provenienti da fonti affidabili, ad esempio ricerche ISTAT, ricerche dell’Ufficio Studi Confcommercio, ricerche Federconsumatori. Un’altra tecnica può essere quella del Social Listening, cioè osservare cosa dicono sul web i consumatori di un particolare servizio/prodotto.

Vediamo quindi quali sono i possibili target della nostra applicazione. Ci aspettiamo di poter vendere il prodotto non solo ad aziende nascenti ma anche a quelle già avviate in cui si vuole facilitare alcuni processi.

Ci rivolgiamo quindi da una parte ad imprenditori del settore ristorazione (con età media 48 anni, poiché le imprese del settore gestite da under 35 rappresentano il 12% - Rapporto ristorazione FIPE su dati InfoCamere) ma anche a tutti i dipendenti di queste imprese: addetti alla sala con età media tra i 18 ed i 30 anni (con contratto part-time) e addetti alla cucina con età media più alta. Per la maggiore si tratta di uomini.

Fatte queste premesse, abbiamo quindi identificato le seguenti user personas (vedi voce [User Personas](#)):



**Nome:** Giuseppe Conti

**Età:** 58 anni

**Occupazione:** Amministratore di un ristorante

**Background:** Giuseppe Conti ha una vasta esperienza nel settore della ristorazione. Dopo aver lavorato come chef per molti anni, ha deciso di aprire il suo ristorante. Ora svolge il ruolo di amministratore, supervisionando tutte le operazioni del ristorante, inclusa la gestione del personale, la pianificazione del menu e la gestione finanziaria. Giuseppe è una persona meticolosa e attenta ai dettagli, che dedica tempo ed energia per garantire l'efficienza e il successo del suo ristorante. Utilizza principalmente un computer per eseguire le sue operazioni quotidiane e ha una buona conoscenza delle tecnologie informatiche. Preferisce un'interfaccia intuitiva e facile da usare, che gli consenta di accedere rapidamente alle informazioni e generare report utili per prendere decisioni strategiche.

**Obiettivi e bisogni:** Giuseppe desidera un applicativo di gestione delle ordinazioni e del menu che semplifichi il processo di gestione e consenta una visione chiara delle operazioni del ristorante. Vuole essere in grado di monitorare le vendite, le spese e i profitti in modo rapido e preciso.

**Aspettative dall'applicazione:** Giuseppe si aspetta che l'applicativo di gestione delle ordinazioni e del menu offra una panoramica completa delle attività del ristorante, comprese le vendite, le spese, i profitti e le statistiche chiave. Inoltre, vuole che l'applicativo sia intuitivo da utilizzare, con un'interfaccia user-friendly che semplifichi le operazioni amministrative quotidiane.



**Obiettivi e bisogni:** Luca si occupa di sviluppare e gestire il menu del ristorante. Desidera un applicativo di gestione del menu che gli consenta di creare, modificare e aggiornare facilmente le selezioni di piatti, le descrizioni e i prezzi. Ha bisogno di un sistema che gli permetta di monitorare le vendite dei piatti, raccogliere feedback dai clienti e valutare l'efficacia del menu. Inoltre, desidera strumenti di analisi per identificare le tendenze di consumo e apportare miglioramenti al menu in base ai dati raccolti.

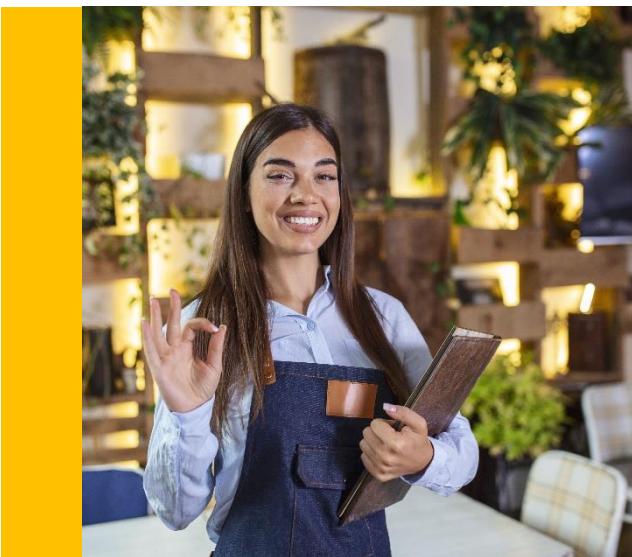
**Aspettative dall'applicazione:**  
Luca si aspetta che l'applicativo di gestione del menu gli consenta di creare, modificare e aggiornare facilmente il menu del ristorante. Desidera un'interfaccia intuitiva che gli permetta di visualizzare in modo chiaro e organizzato le selezioni di piatti, le descrizioni, i prezzi e le eventuali opzioni personalizzate. Inoltre, vuole che l'applicativo gli fornisca strumenti di analisi e report per valutare le vendite dei piatti, raccogliere feedback dai clienti e apportare miglioramenti al menu in base ai dati raccolti.

**Nome:** Luca De Santis

**Età:** 35 anni

**Occupazione:** Supervisore del ristorante

**Background:** Luca De Santis ha una vasta esperienza nel campo della gastronomia e dell'enogastronomia. Ha studiato culinaria presso una rinomata scuola alberghiera e ha lavorato come chef in diversi ristoranti di alto livello. Grazie alla sua conoscenza approfondita degli ingredienti, delle tecniche di cucina e delle tendenze gastronomiche, è stato promosso a supervisore del menu del ristorante. Luca è appassionato di creare esperienze culinarie uniche e di garantire che il menu del ristorante sia innovativo, equilibrato e in grado di soddisfare le esigenze dei clienti. Luca trascorre gran parte del suo tempo lavorando nel suo ufficio, esaminando le tendenze di mercato, sperimentando nuove ricette e analizzando i dati di vendita. Utilizza principalmente un computer e un monitor touch per accedere all'applicativo di gestione del menu. È creativo, attento ai dettagli e cerca costantemente di migliorare l'offerta culinaria del ristorante.



**Obiettivi e bisogni:** Marta desidera essere in grado di prendere le ordinazioni dei clienti in modo rapido ed efficiente, evitando errori e confusioni. Vuole fornire un servizio di alta qualità, mantenendo un flusso di lavoro fluido tra la sala e la cucina. Inoltre, desidera avere accesso a informazioni aggiornate sul menu, comprese le opzioni di personalizzazione e le allergie alimentari, al fine di soddisfare al meglio le esigenze dei clienti.

**Aspettative dall'applicazione:** Marta si aspetta che l'applicativo sia intuitivo e facile da usare, consentendole di prendere ordinazioni con pochi clic e senza dover scrivere manualmente ogni dettaglio. Desidera che l'applicativo mostri in modo chiaro il menu, incluse le opzioni di personalizzazione e le informazioni sulle allergie alimentari. Marta si aspetta che l'applicativo sia affidabile e reattivo, evitando interruzioni o rallentamenti che possano influire sul suo lavoro durante i momenti di picco.

**Nome:** Marta Bianchi

**Età:** 27 anni

**Occupazione:** Addetta alla sala del ristorante

**Background:** Marta Bianchi ha lavorato come cameriera per diversi anni e ha sviluppato una vasta esperienza nel fornire un servizio di qualità ai clienti. È appassionata del settore della ristorazione e ha una buona conoscenza del menu e delle bevande del ristorante in cui lavora. Marta è una persona energica, amichevole e sempre pronta ad assistere i clienti per garantire un'esperienza piacevole nel ristorante. Utilizza dispositivi mobili come smartphone o tablet per prendere le ordinazioni e comunicare con la cucina. È abile nel multitasking e si adatta facilmente alle nuove tecnologie.



**Obiettivi e bisogni:** Anna si occupa della preparazione dei piatti e della gestione generale della cucina. Desidera un'applicazione di gestione che le consenta di ricevere e visualizzare ordinazioni in modo chiaro e tempestivo, tenere traccia delle ricette e delle istruzioni di preparazione. Ha bisogno di uno strumento intuitivo e affidabile per semplificare le operazioni quotidiane in cucina e garantire una preparazione accurata e tempestiva dei piatti.

**Aspettative dall'applicazione:** Anna si aspetta che l'applicativo di gestione sia user-friendly e le consenta di visualizzare le ordinazioni in modo organizzato e dettagliato. Vuole che l'applicativo le fornisca istruzioni di preparazione chiare e precise per ogni piatto. Inoltre, desidera che l'applicativo le permetta di comunicare facilmente con gli altri membri del team di cucina, consentendo una collaborazione più efficace e una gestione fluida dei flussi di lavoro in cucina.

**Nome:** Anna Mwangi

**Età:** 42 anni

**Occupazione:** Addetta alla cucina

**Background:** Anna Mwangi ha una vasta esperienza nel settore della ristorazione, con oltre 20 anni di carriera come cuoca e chef in diversi ristoranti di successo. Ha iniziato la sua carriera come commessa di cucina e nel corso degli anni ha sviluppato una profonda conoscenza delle tecniche culinarie e una passione per la preparazione di piatti tradizionali e innovativi. Anna è conosciuta per la sua attenzione ai dettagli e la capacità di mantenere la calma anche durante i momenti di maggiore pressione in cucina. Trascorre la maggior parte del suo tempo nella cucina del ristorante, lavorando diligentemente con le attrezzature e gli ingredienti per preparare piatti deliziosi. Utilizza principalmente un computer o un tablet per accedere all'applicativo di gestione, visualizzare le ordinazioni e seguire le istruzioni di preparazione. Grazie alla sua esperienza, Anna è in grado di adattarsi facilmente alle diverse richieste dei clienti e ai cambiamenti nel menu.

## 2.6 Valutazione dell'usabilità a priori

La valutazione euristica dell'usabilità (vedi voce [Usabilità](#)) a priori è un metodo di valutazione che si concentra sull'identificazione dei problemi di usabilità in un'interfaccia utente, basandosi su principi e linee guida consolidate. Durante questa valutazione, gli esperti di usabilità applicano principi euristiche o regole generali per esaminare l'interfaccia e identificare i possibili problemi che potrebbero ostacolare l'esperienza utente. Alcune delle euristiche ampiamente conosciute nel campo dell'usabilità sono state formulate da Ben Shneiderman, uno dei pionieri dell'interazione uomo-macchina. Queste regole, conosciute come "**le otto regole d'oro di Shneiderman**", forniscono una guida per progettare un'interfaccia utente intuitiva ed efficace. In particolare, durante tutta la progettazione delle interfacce **sono state rispettate le seguenti regole:**

### 1. Coerenza a tutti i costi

Nell'applicativo sono presenti sequenze di azioni simili per situazioni simili, ad esempio tutti i form di inserimento (inserimento piatti, menu, categorie) e quelli di modifica, hanno tutti la stessa rappresentazione grafica e lo stesso numero di passaggi da effettuare. Anche le tabelle che elencano i piatti, le categorie, i menu e gli addetti sono fatte allo stesso modo. Il layout è coerente in tutte le interfacce.

### 2. Offrire riscontri informativi

Nell'applicativo ad ogni azione dell'utente è associato un riscontro grafico, che sia con dei popup (di errore, di conferma, di successo o informativi) o con aggiornamenti dell'interfaccia grafica.

### 3. Dialogo con gli utilizzatori

L'utente dell'applicativo riesce sempre a capire a che punto è rispetto all'azione che vuole portare a termine. Ad esempio, nei modali di inserimento, la fase iniziale è rappresentata dal form vuoto, la fase intermedia è rappresentata dal form pieno e la fase finale è rappresentata dal popup (di errore o successo).

### 4. Prevenire gli errori

Per prevenire gli errori, le interfacce, e in particolare gli elementi di input, sono stati creati in modo tale da non offrire piena libertà all'utente, così che il range possibile dei valori fosse ristretto a priori (ad esempio gli input di date, numeri).

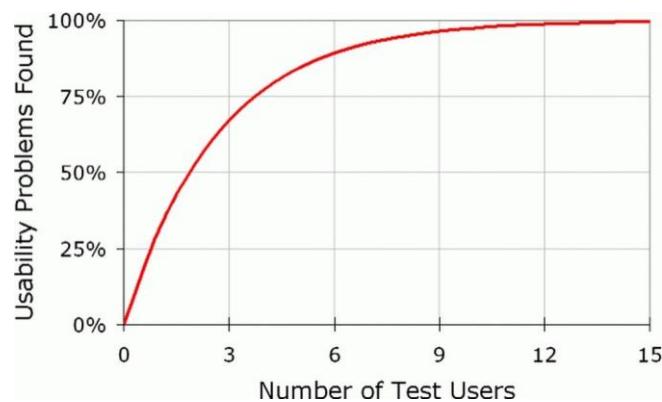
### 5. Garantire il controllo agli utenti

### 6. Ridurre il carico di memoria a breve termine

Tutti i form sono stati progettati in modo tale che si riesce a completare il form senza cambiare pagina

## 2.6.1 Test di usabilità

Per completare la valutazione dell’usabilità, oltre ad utilizzare le euristiche, è possibile effettuare dei test di usabilità (vedi voce [Test di usabilità](#)). Abbiamo deciso di impiegare un osservatore, un facilitatore e cinque valutatori. Secondo uno studio effettuato dalla [N/N Group](#), cinque valutatori bastano per scoprire circa l’85% dei problemi di usabilità.



I cinque valutatori sono così ripartiti:

- Un valutatore con bassa conoscenza tecnologica e bassa conoscenza del dominio: Giorgio M.
- Due valutatori con bassa conoscenza tecnologica e alta conoscenza del dominio: Pietro P., Lucia C.
- Un valutatore con alta conoscenza tecnologica e bassa conoscenza del dominio: Mario V.
- Un valutatore con alta conoscenza tecnologica e alta conoscenza del dominio: Rosa F.

I **compiti** assegnati ai valutatori sono i seguenti:

1. Prendere un’ordinazione
2. Evadere un’ordinazione
3. Visualizzare statistiche addetti alla sala con cambio date
4. Ordinare elementi menù

I test sono stati effettuati con metodologia **Mago di Oz**.

Valutatore	Compito 1	Compito 2	Compito 3	Compito 4
Giorgio M.	S	S	P	F
Pietro P.	S	S	S	P
Lucia C.	S	S	S	P
Mario V.	S	P	S	S
Rosa F.	S	S	S	S

dove: S = Successo (1), P = Successo Parziale (0.5) e F = Fallimento (0).

Il tasso di successo è quindi:  $(15 + (5 * 0,5)) / 20 = 88\%$ .

Il compito più complicato è stato quello di ordinare gli elementi all'interno del menu. La difficoltà riscontrata è stata l'assenza di un tasto per entrare nella schermata di modifica del menù.

## 2.7 Glossario

**UML (Unified Modeling Language)**: è un linguaggio standard per la modellazione dei sistemi software che utilizza diagrammi grafici. È ampiamente utilizzato per la progettazione, l'analisi e la documentazione dei sistemi complessi.

**Use Case Diagram**: è un tipo di diagramma UML che rappresenta le relazioni tra gli attori (utenti) e i casi d'uso (scenari) di un sistema.

**User personas**: sono rappresentazioni sintetiche e dettagliate di utenti tipici di un sistema o prodotto. Vengono create sulla base di ricerche e analisi dei bisogni degli utenti e comprendono informazioni demografiche, comportamentali, preferenze e obiettivi.

**Usabilità**: si riferisce alla facilità con cui gli utenti possono utilizzare un sistema o un prodotto per raggiungere i propri obiettivi in modo efficace ed efficiente, con soddisfazione.

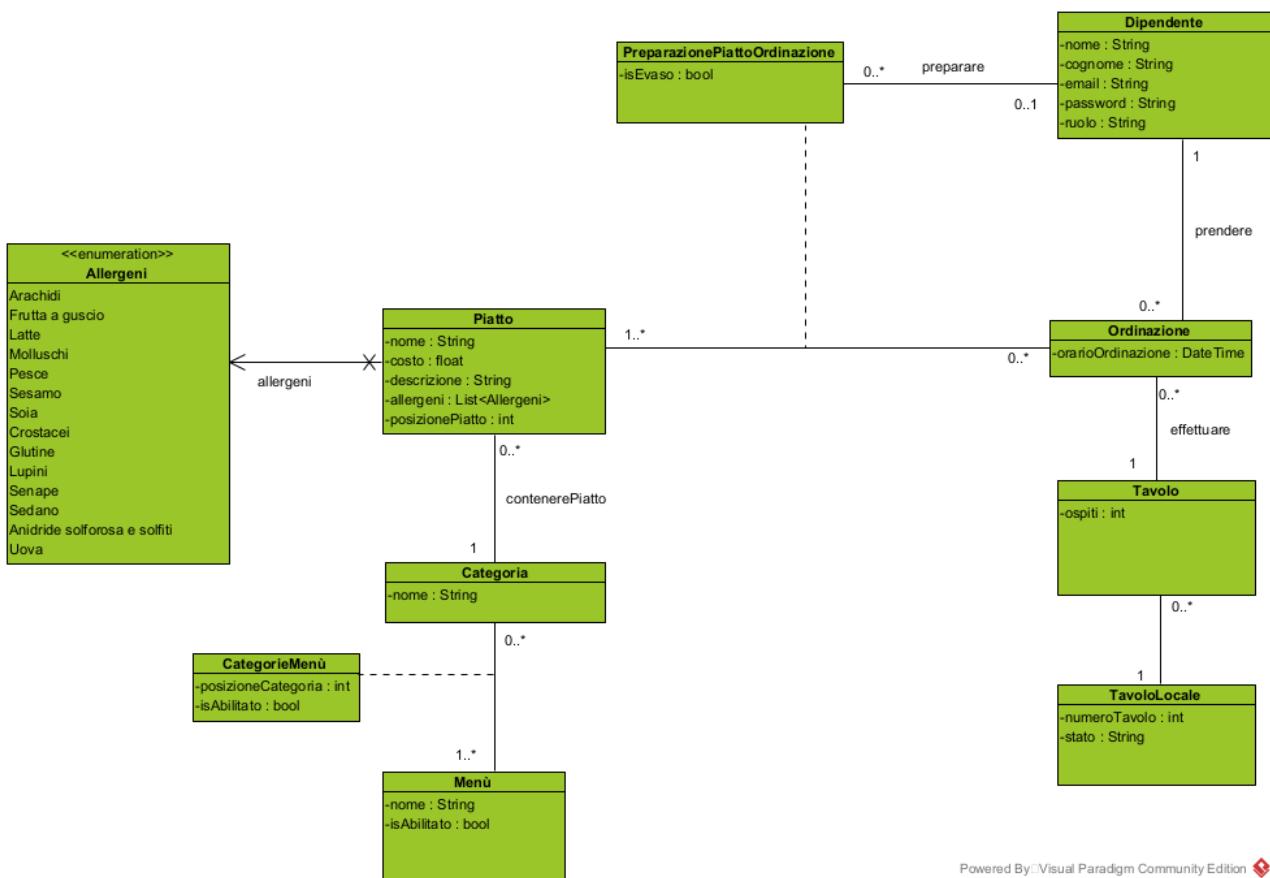
**Test di usabilità**: è un metodo di valutazione che coinvolge gli utenti nel testare un sistema o un prodotto per identificare i problemi di usabilità.

# 3 Specifica dei requisiti

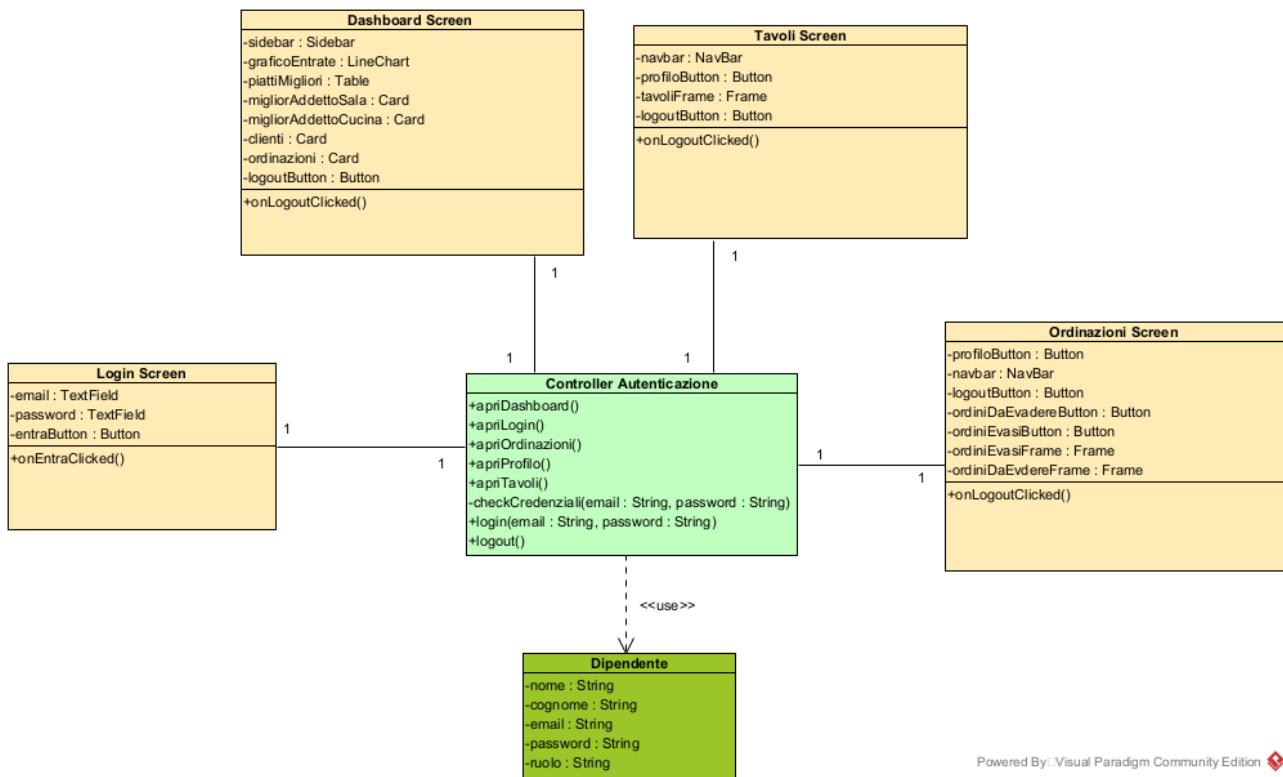
## 3.1 Classi, oggetti e relazioni di analisi

Per lo sviluppo dei seguenti Class Diagram è stato utilizzato il pattern architettonale **ECB**(Entity–Control–Boundary), il quale suddivide le classi individuate in base alle loro responsabilità nella realizzazione dei vari use cases. È stato deciso di suddividere i Class Diagram secondo le varie funzionalità, per questioni di leggibilità e complessità.

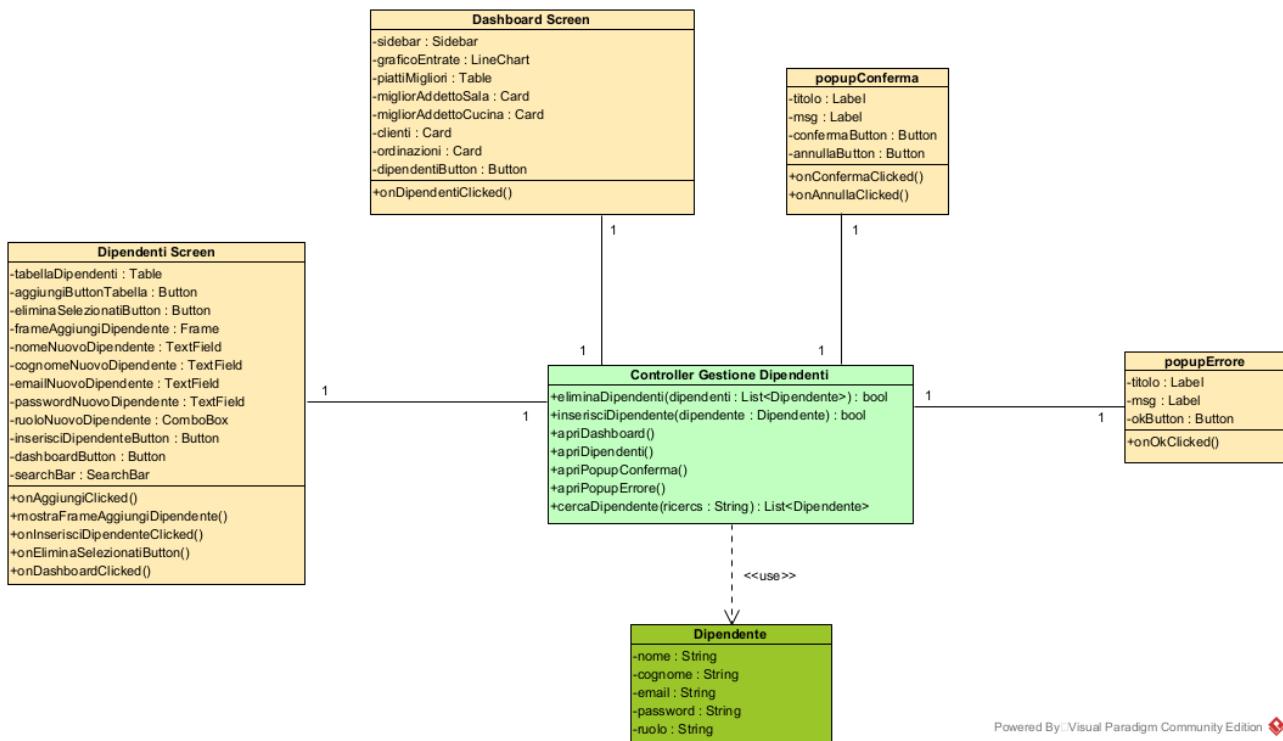
### 3.1.1 Entità



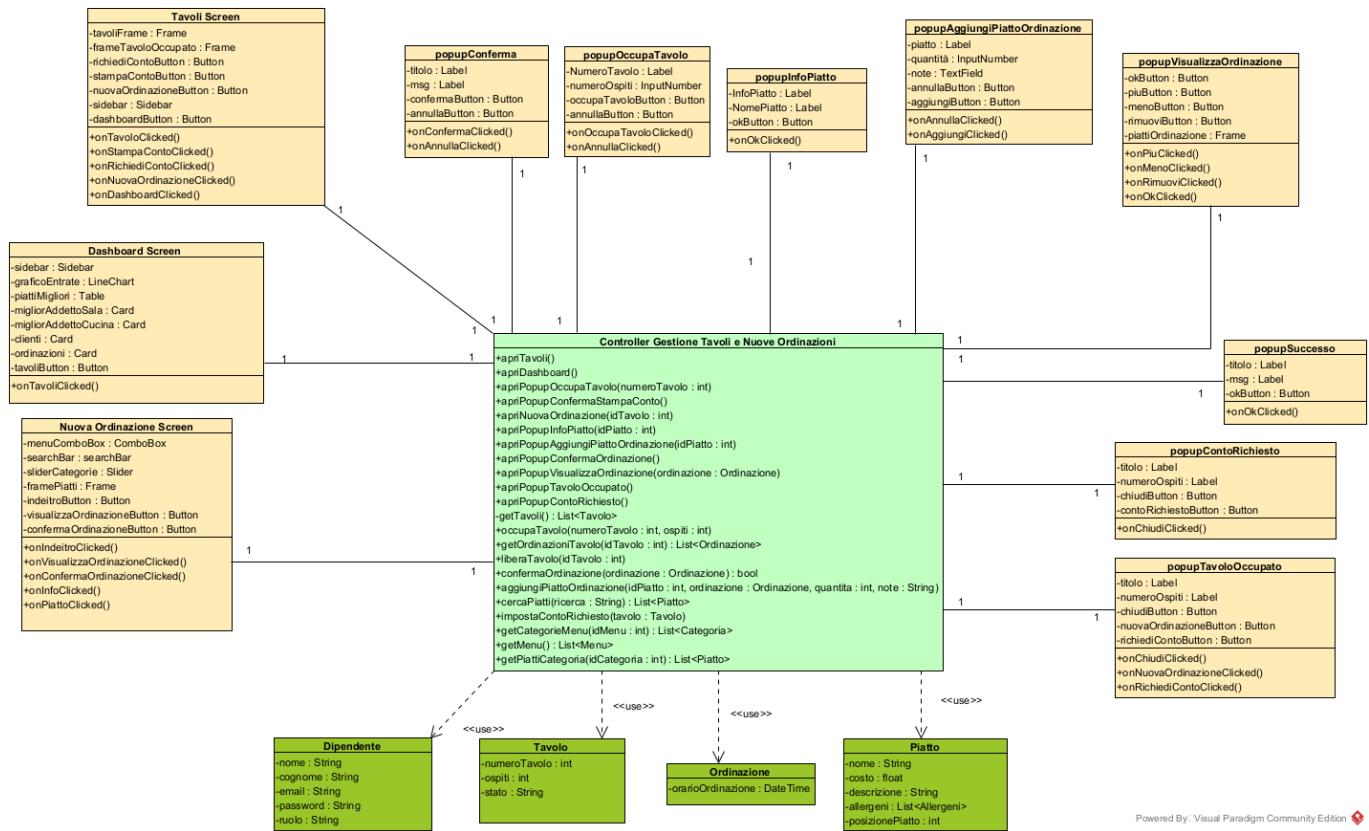
### 3.1.2 Class Diagram Autenticazione



### 3.1.3 Class Diagram Dipendenti

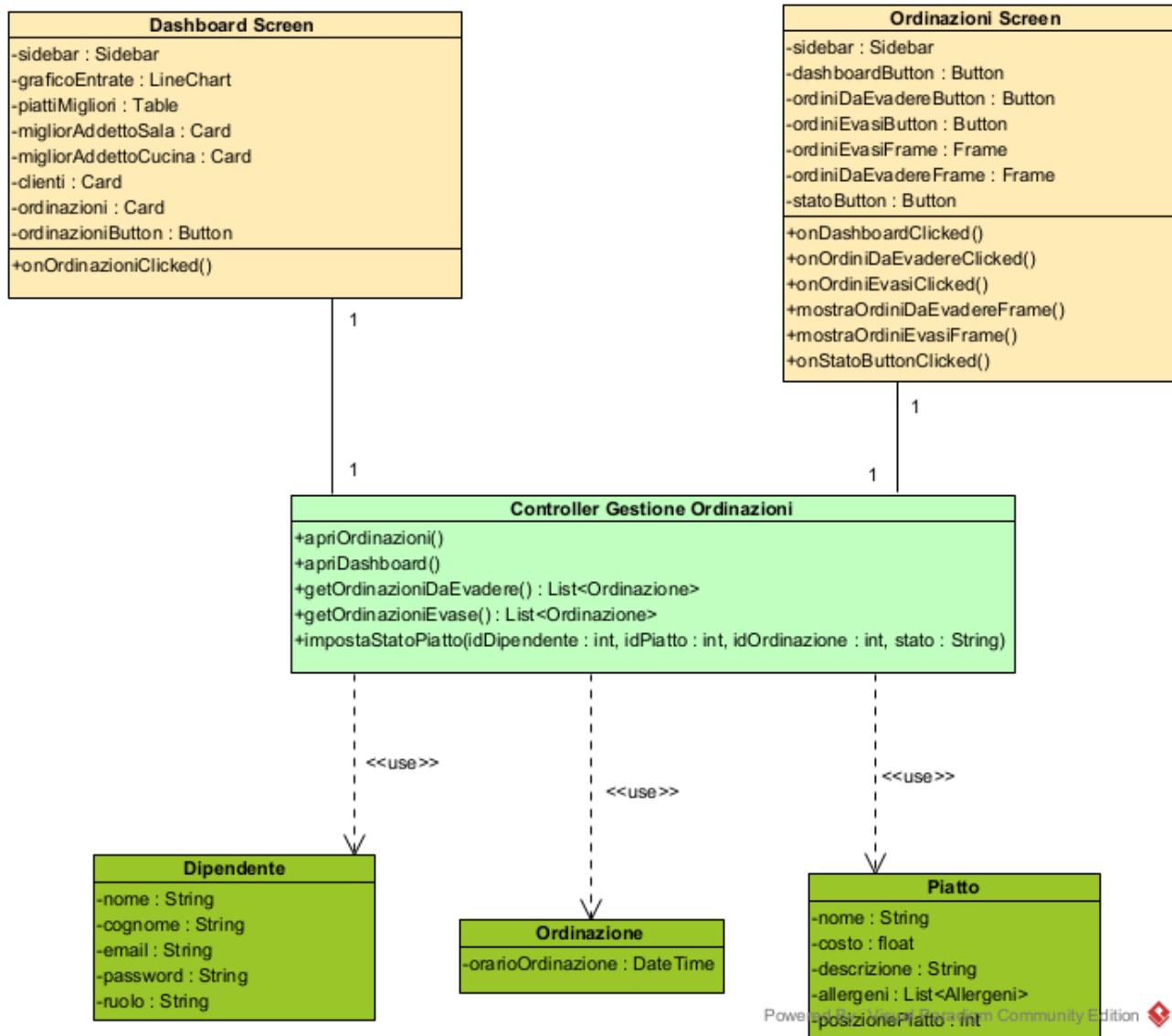


### 3.1.4 Class Diagram Tavoli e Nuove ordinazioni

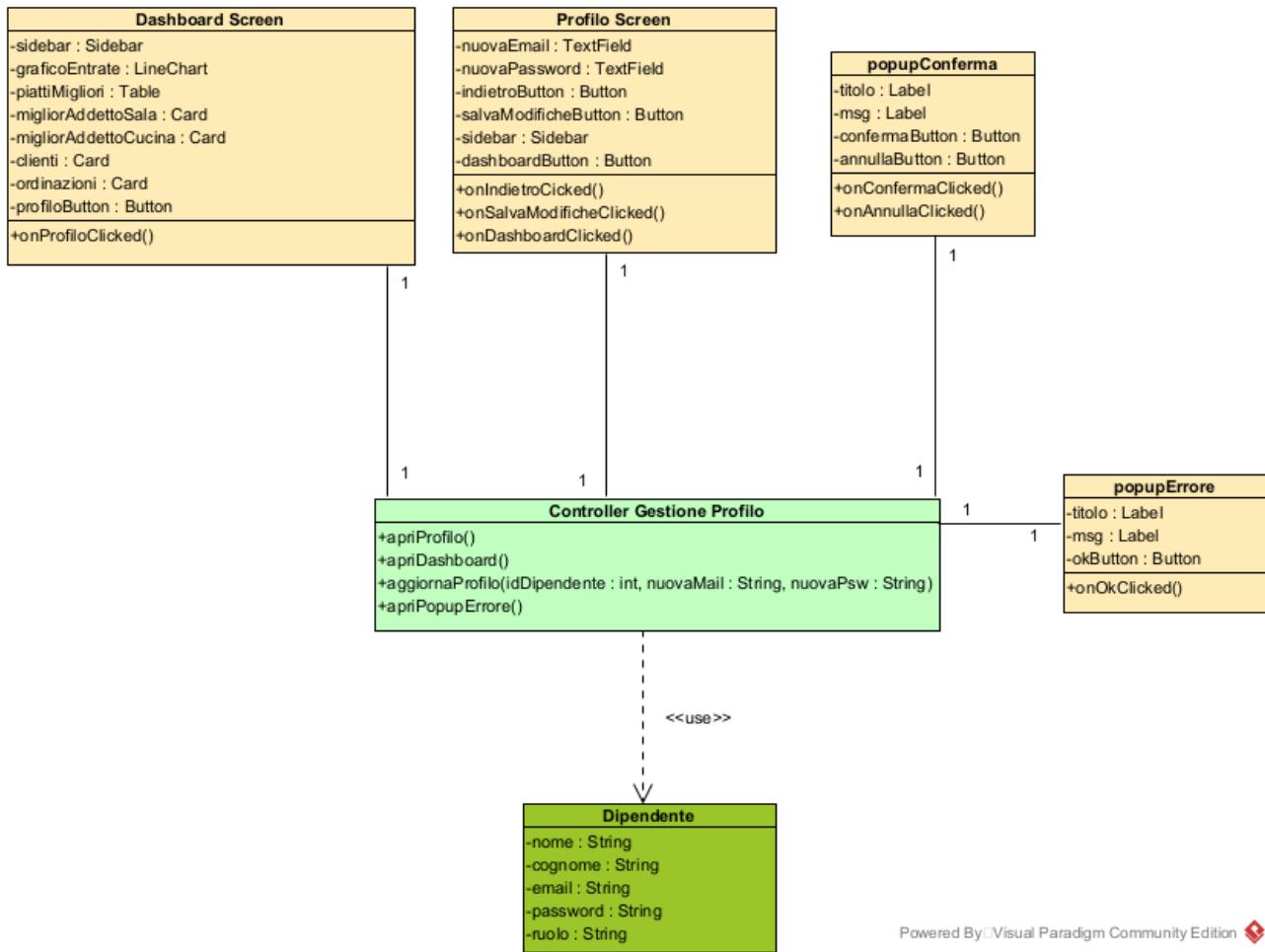


Powered By: Visual Paradigm Community Edition

### 3.1.5 Class Diagram Gestione ordinazioni

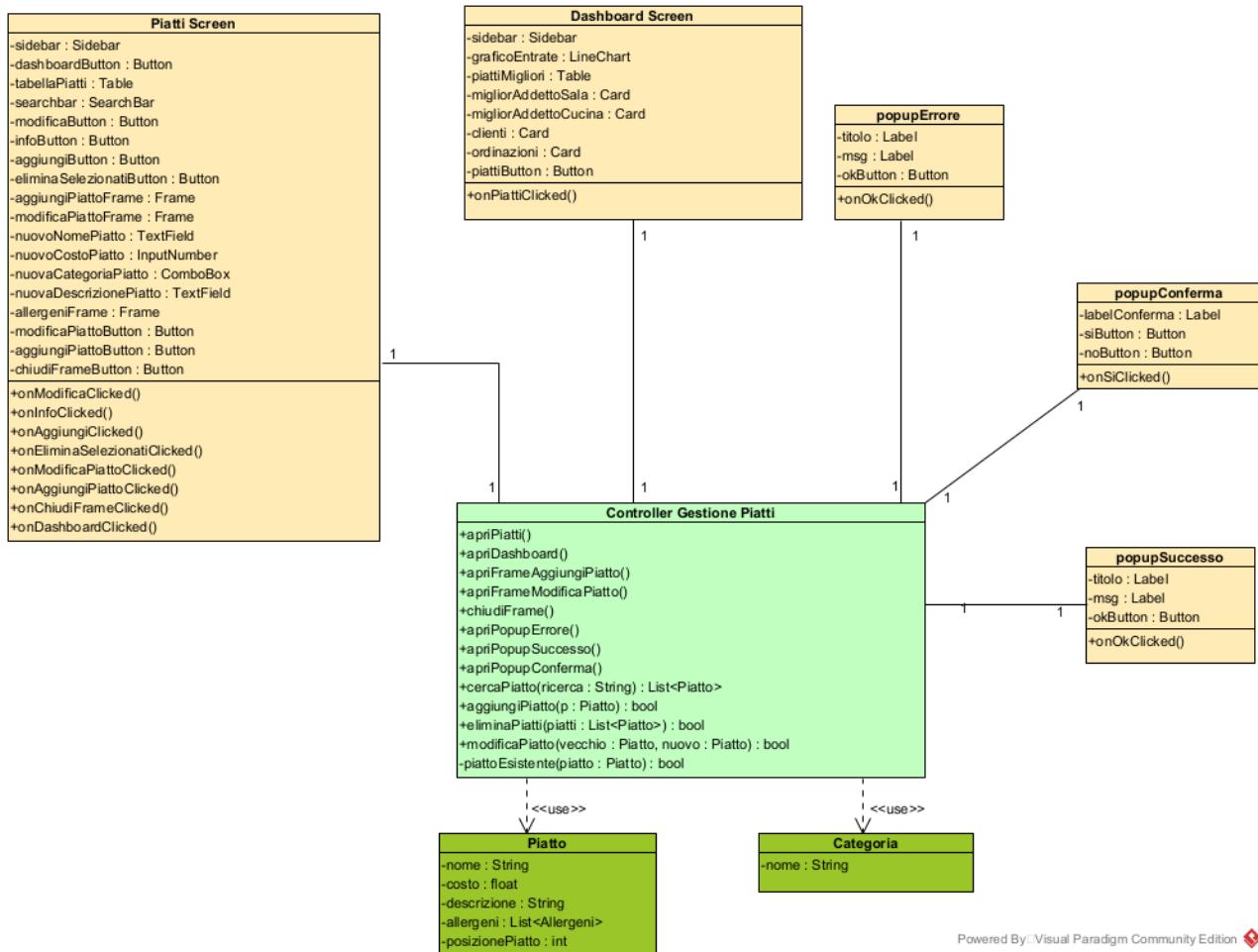


### 3.1.6 Class Diagram Gestione profilo



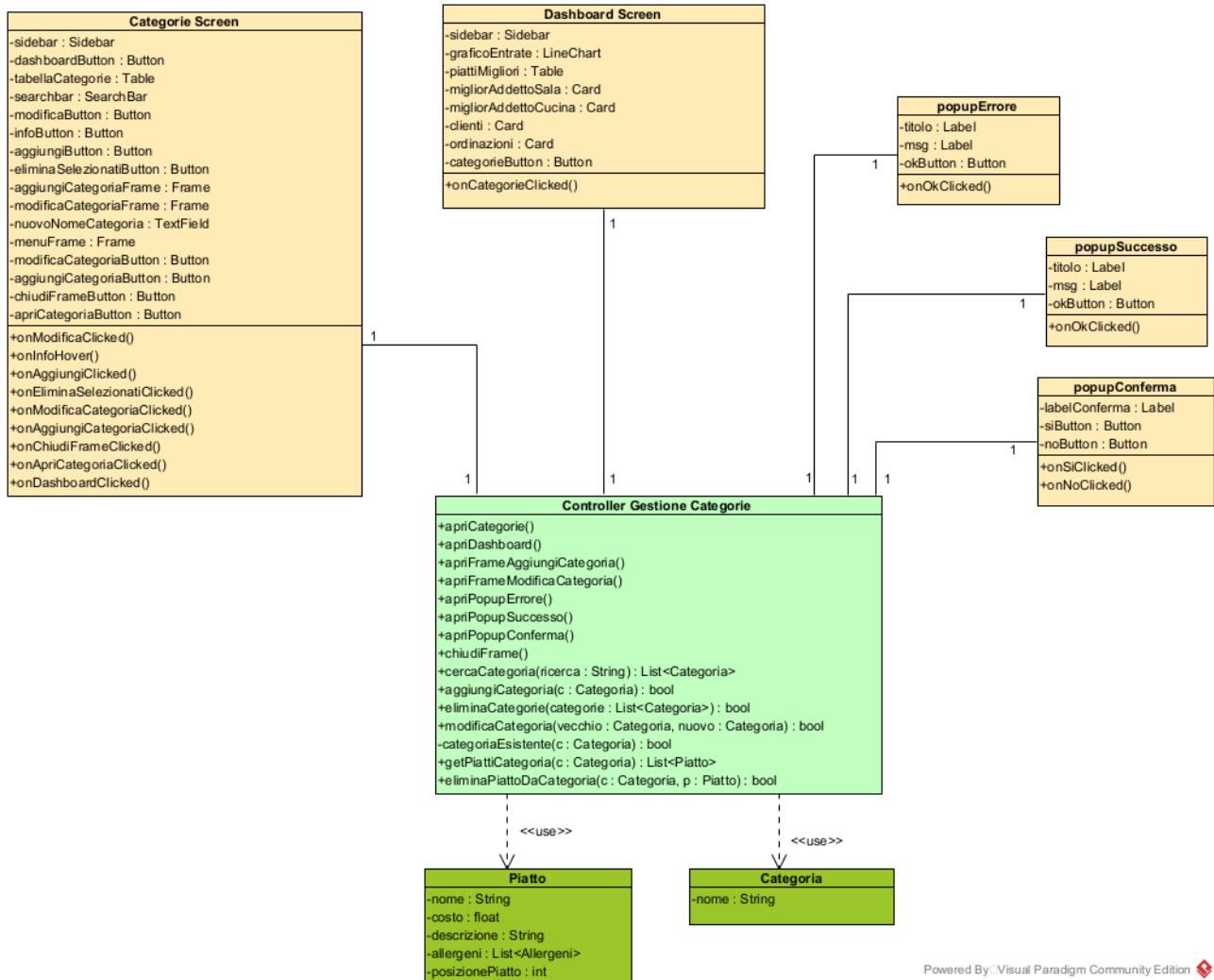
Powered By Visual Paradigm Community Edition

### 3.1.7 Class Diagram Gestione piatti



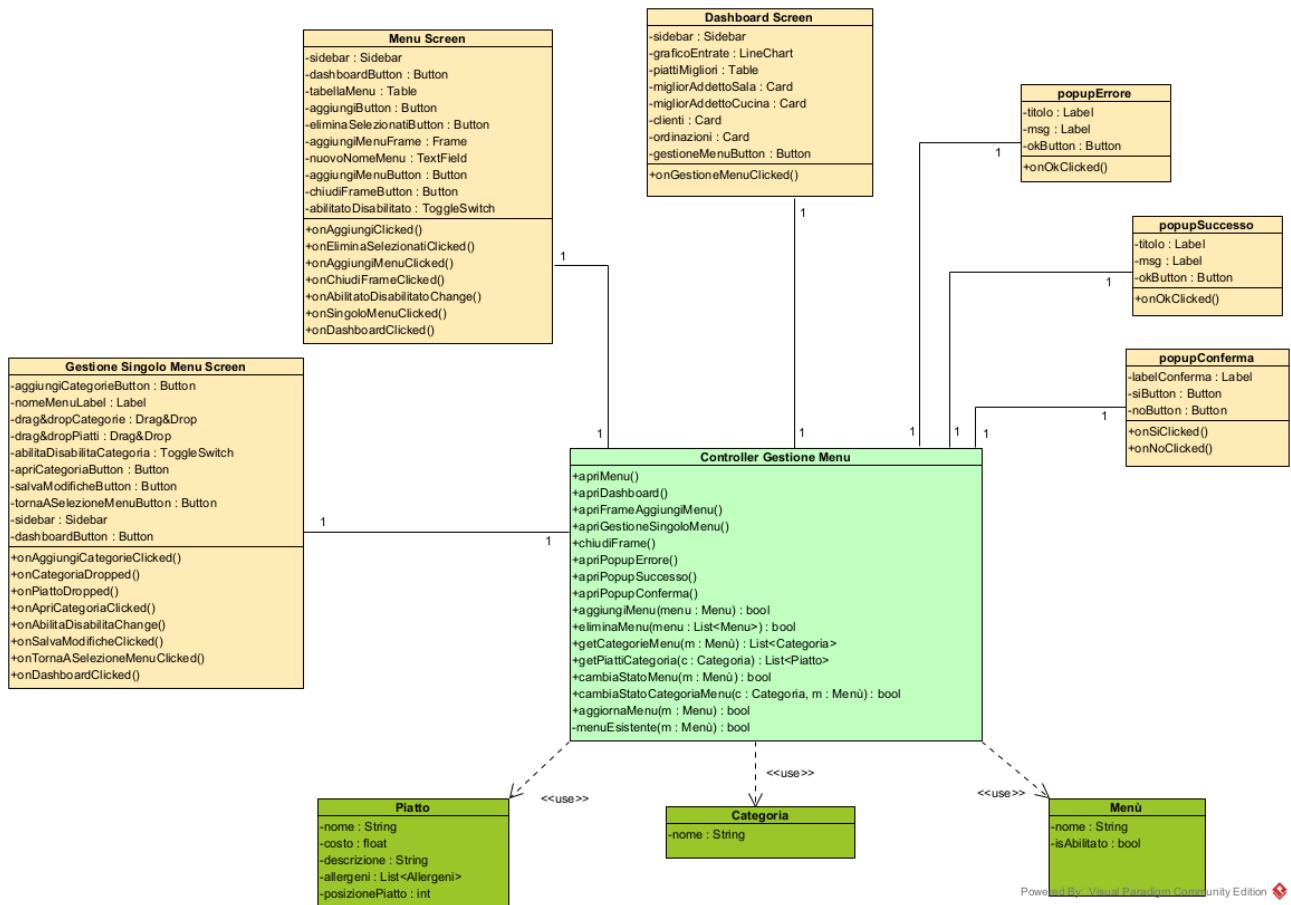
Powered By Visual Paradigm Community Edition

### 3.1.8 Class Diagram Gestione categorie



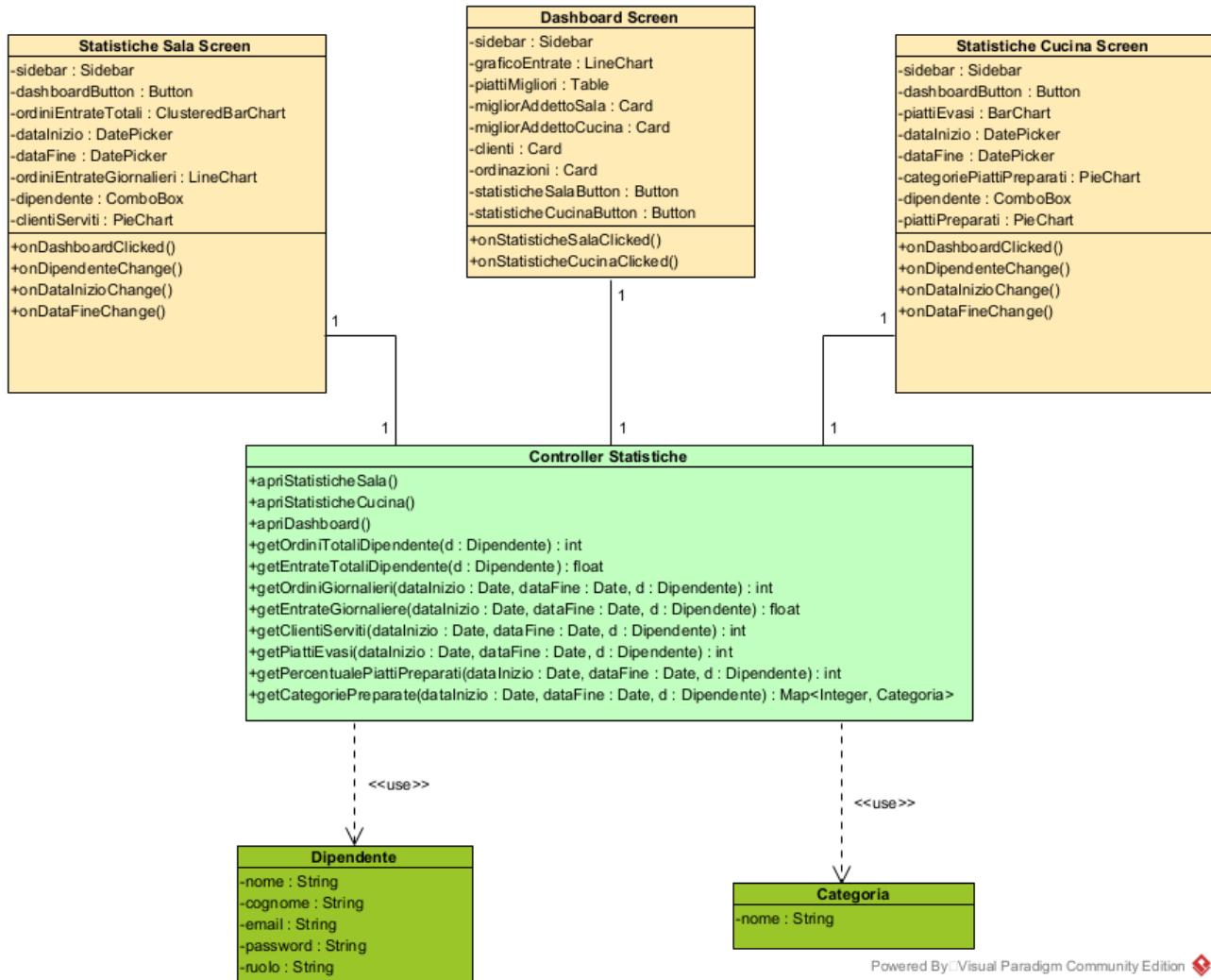
Powered By: Visual Paradigm Community Edition

### 3.1.9 Class Diagram Gestione menù



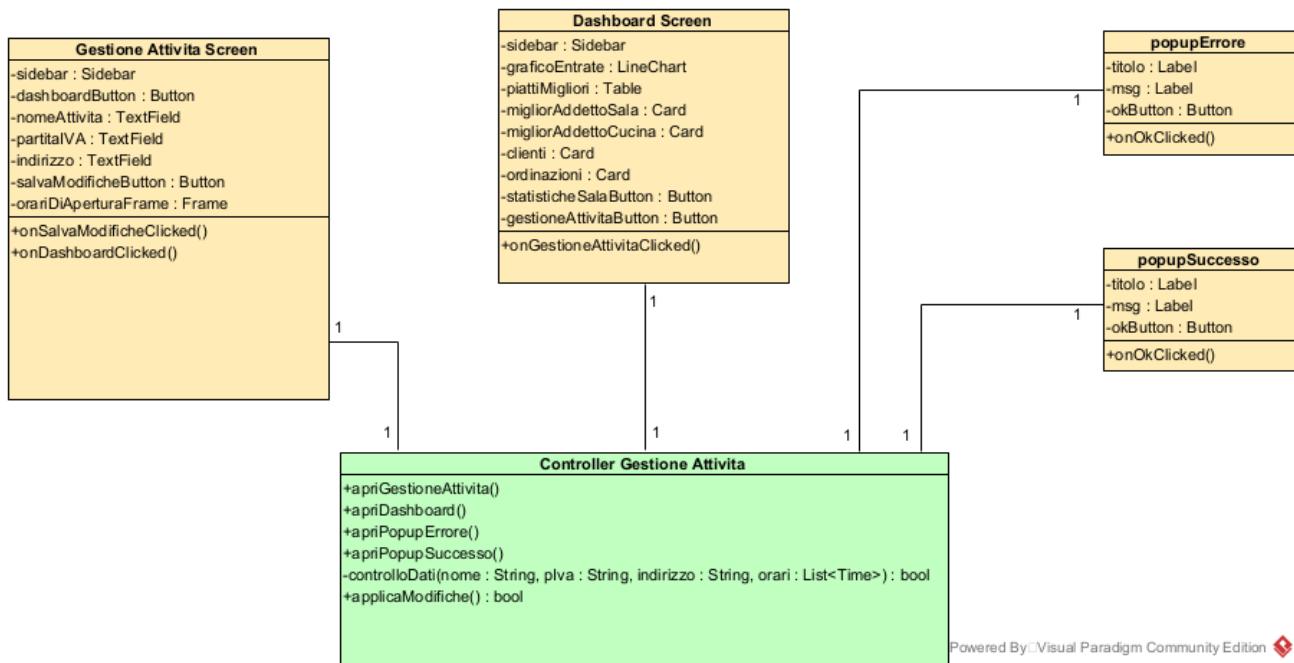
Powered By: Visual Paradigm Community Edition

### 3.1.10 Class Diagram Statistiche addetti alla sala e cucina



Powered By: Visual Paradigm Community Edition

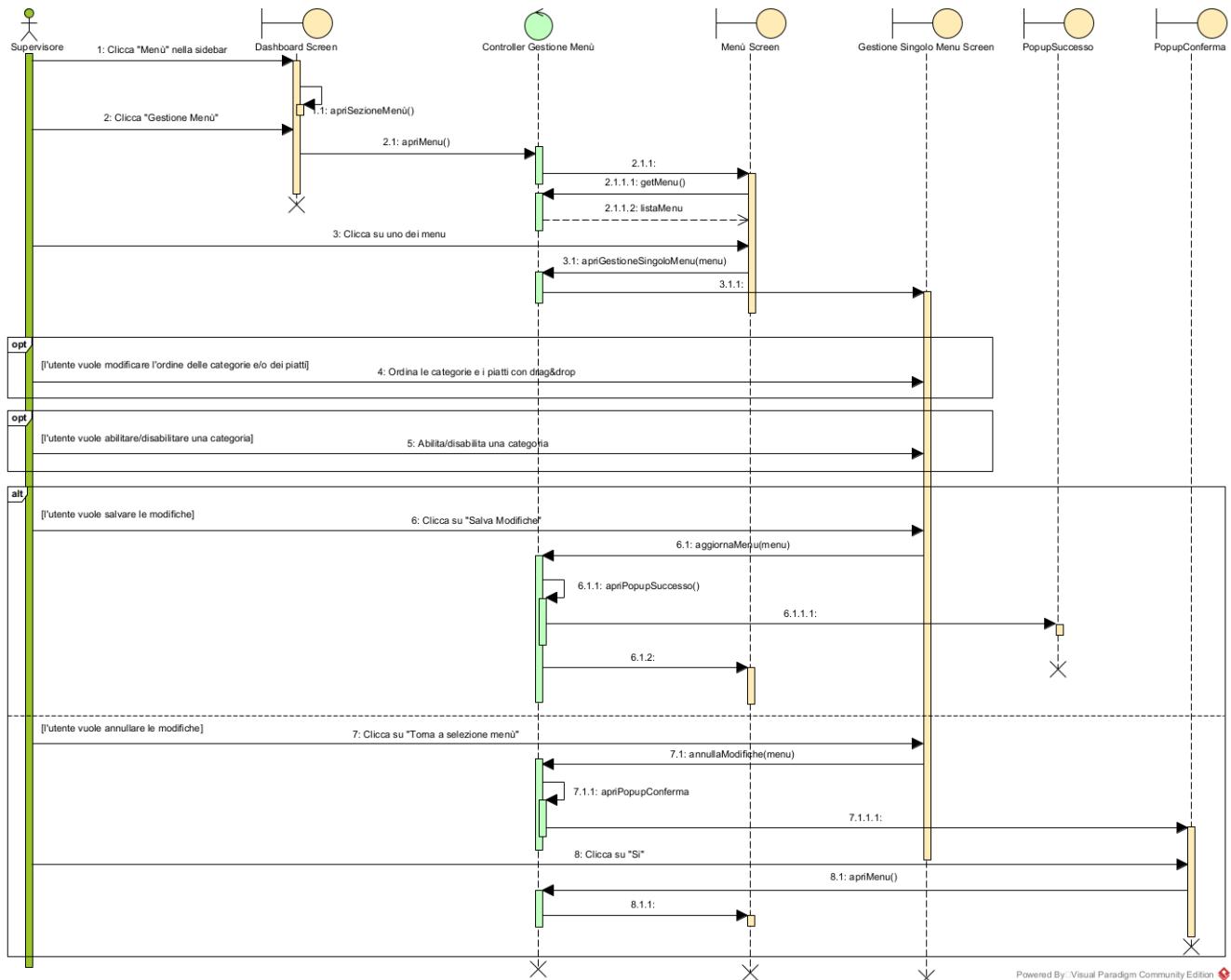
### 3.1.11 Class Diagram Gestione attività



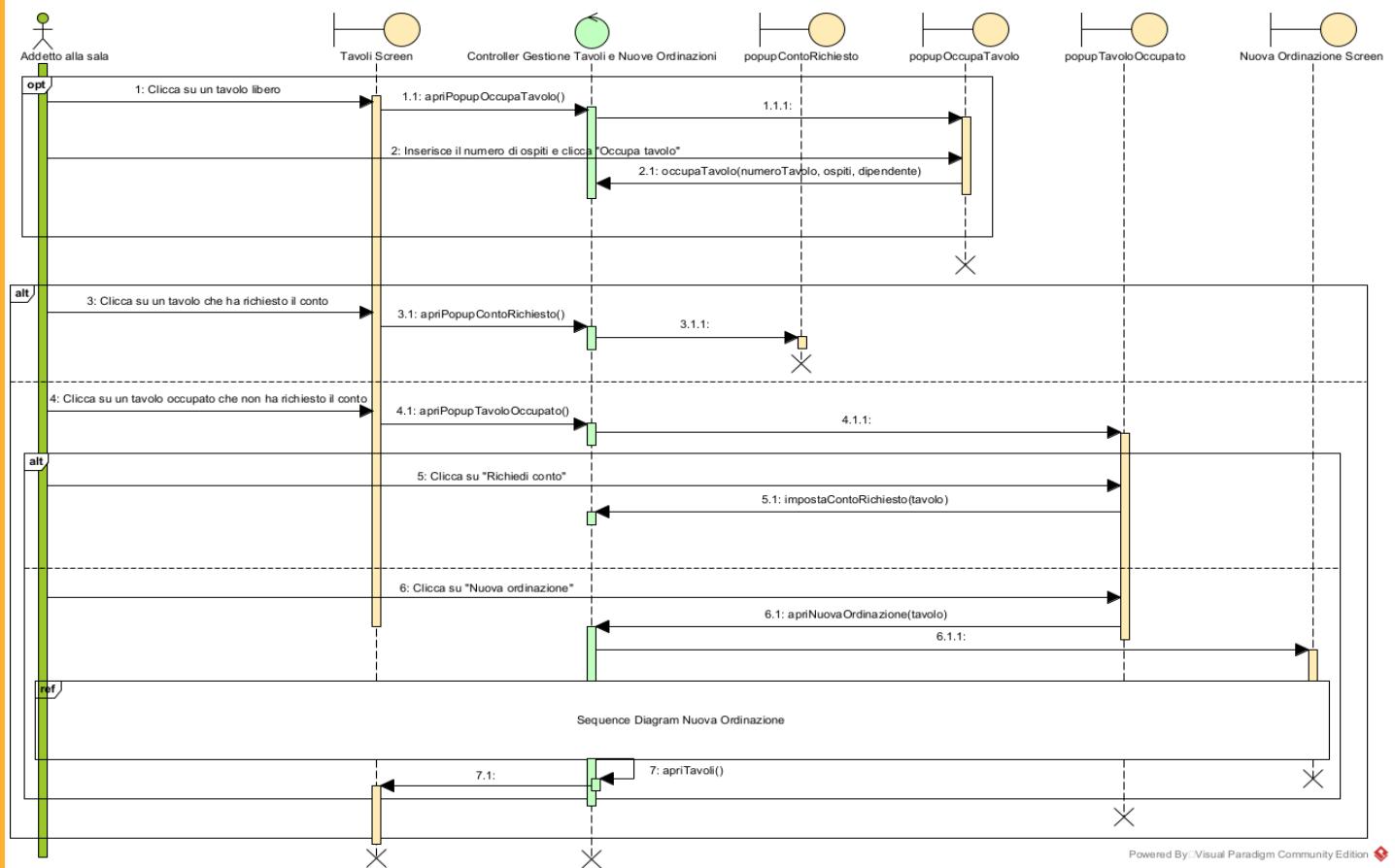
### 3.2 Sequence Diagram di analisi

In questa sezione sono presentati dei sequence diagram di analisi per due casi d'uso significativi, rispettivamente la funzionalità di organizzare menù e di prendere ordinazioni. Un Sequence Diagram mette in relazione Attori e Oggetti di un Sistema, evidenziando le loro interazioni dal punto di vista temporale. In breve, mostra l'ordine delle invocazioni (sequenza) da eseguire per un caso d'uso.

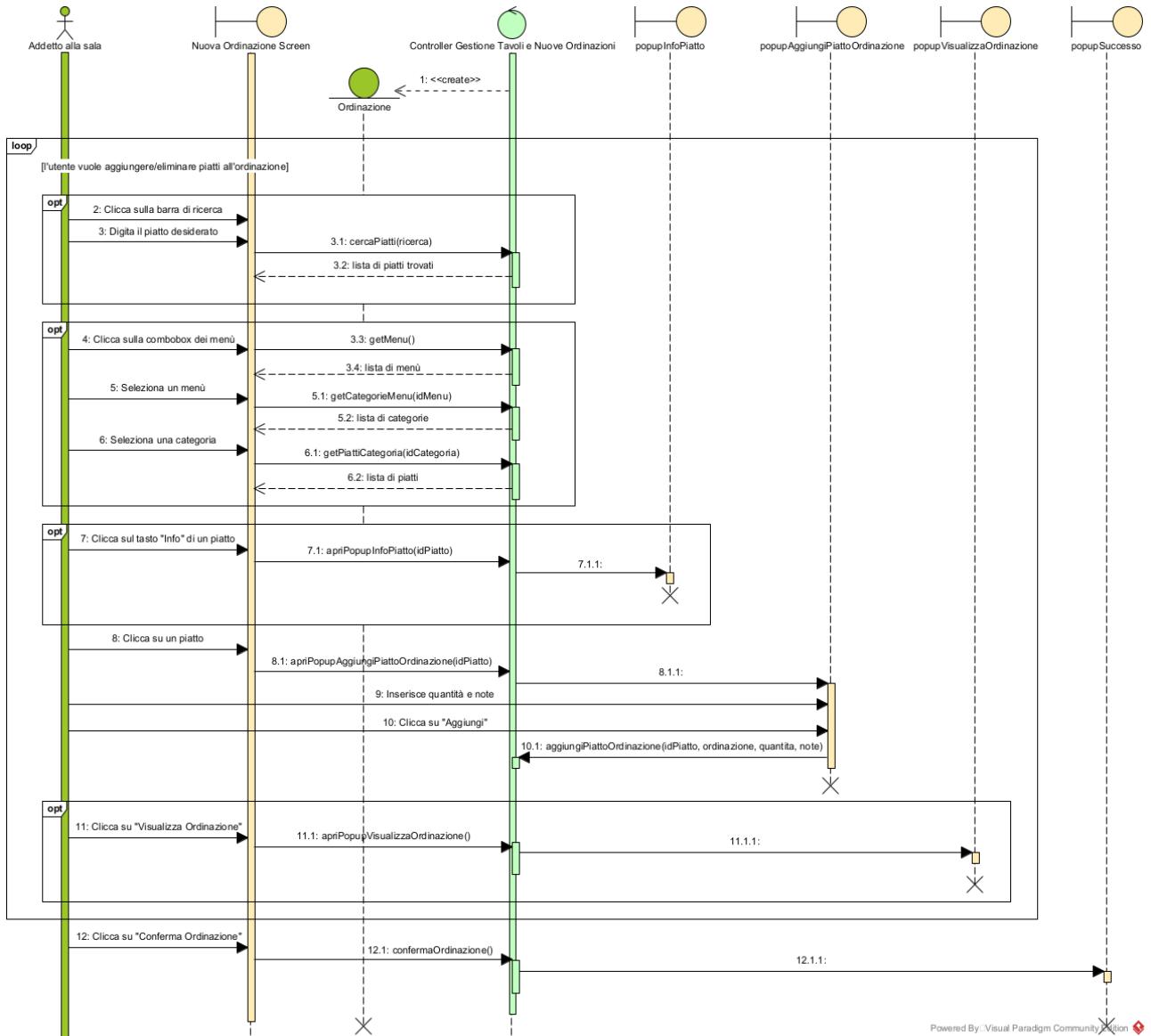
### 3.2.1 Organizzare menù



### 3.2.2 Prendere ordinazioni



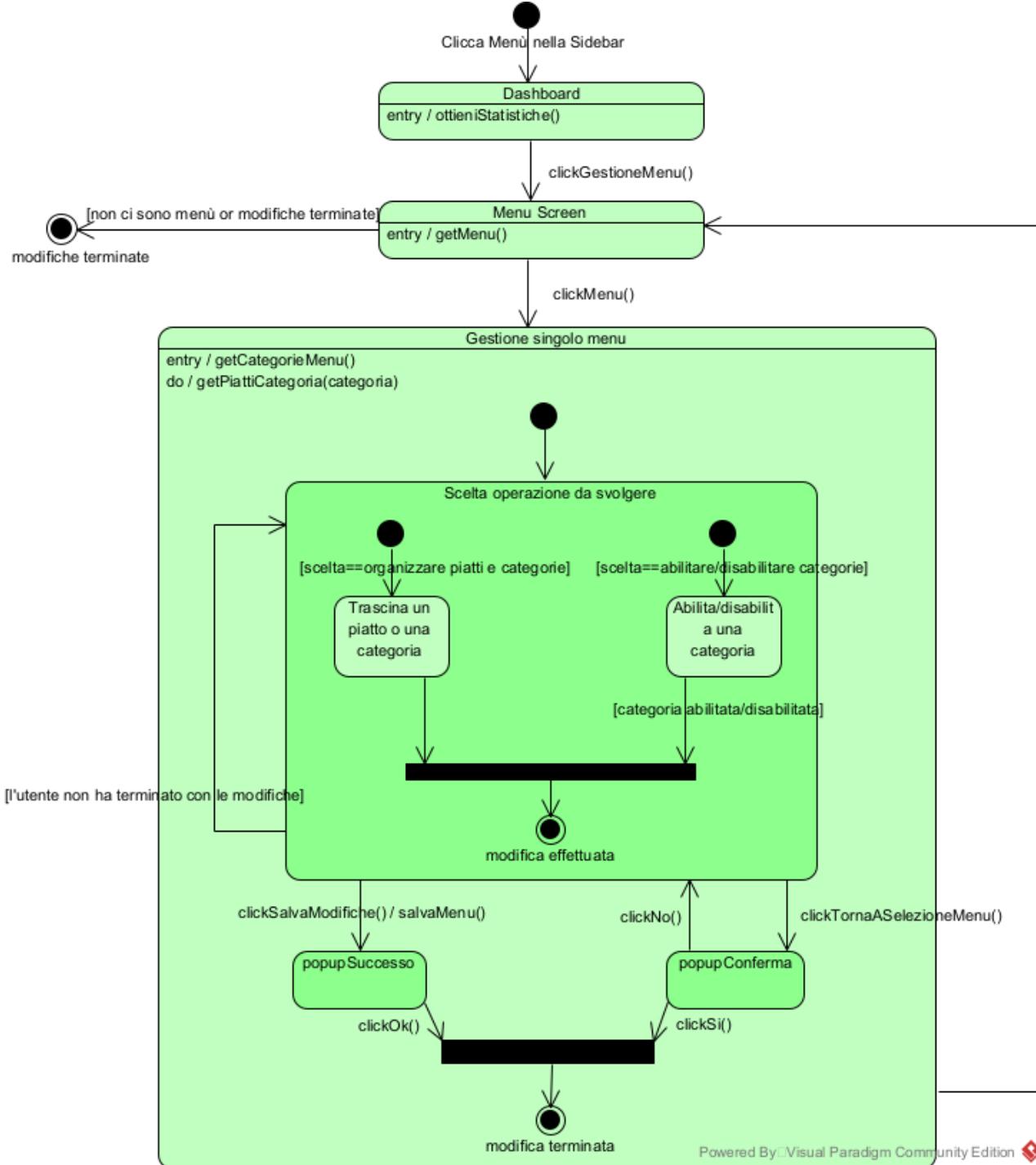
Powered By: Visual Paradigm Community Edition



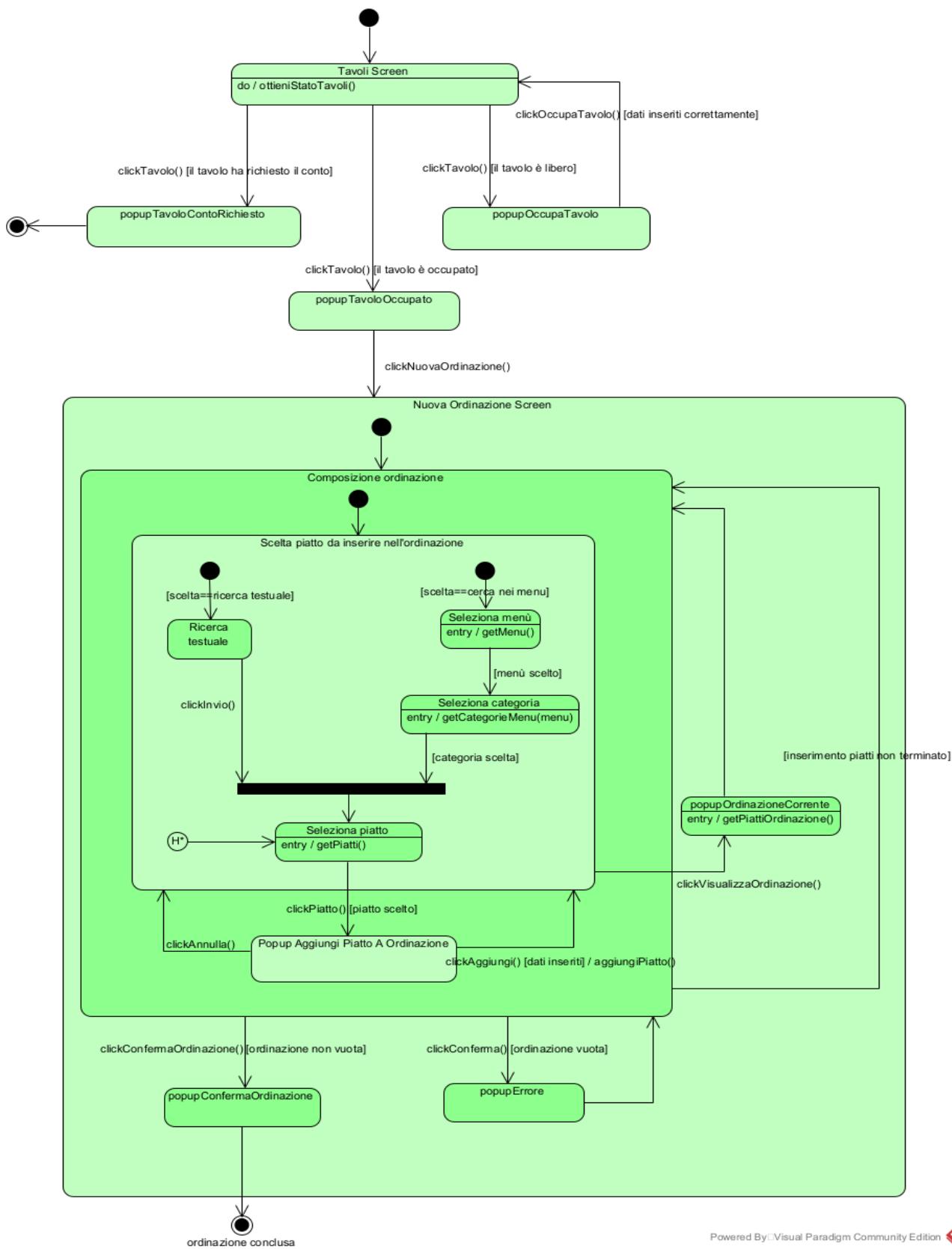
### 3.3 Statechart dell'interfaccia grafica

In questo paragrafo, rappresenteremo il funzionamento interno delle interfacce grafiche di due casi d'uso specifici: organizzare menù e prendere ordinazioni.

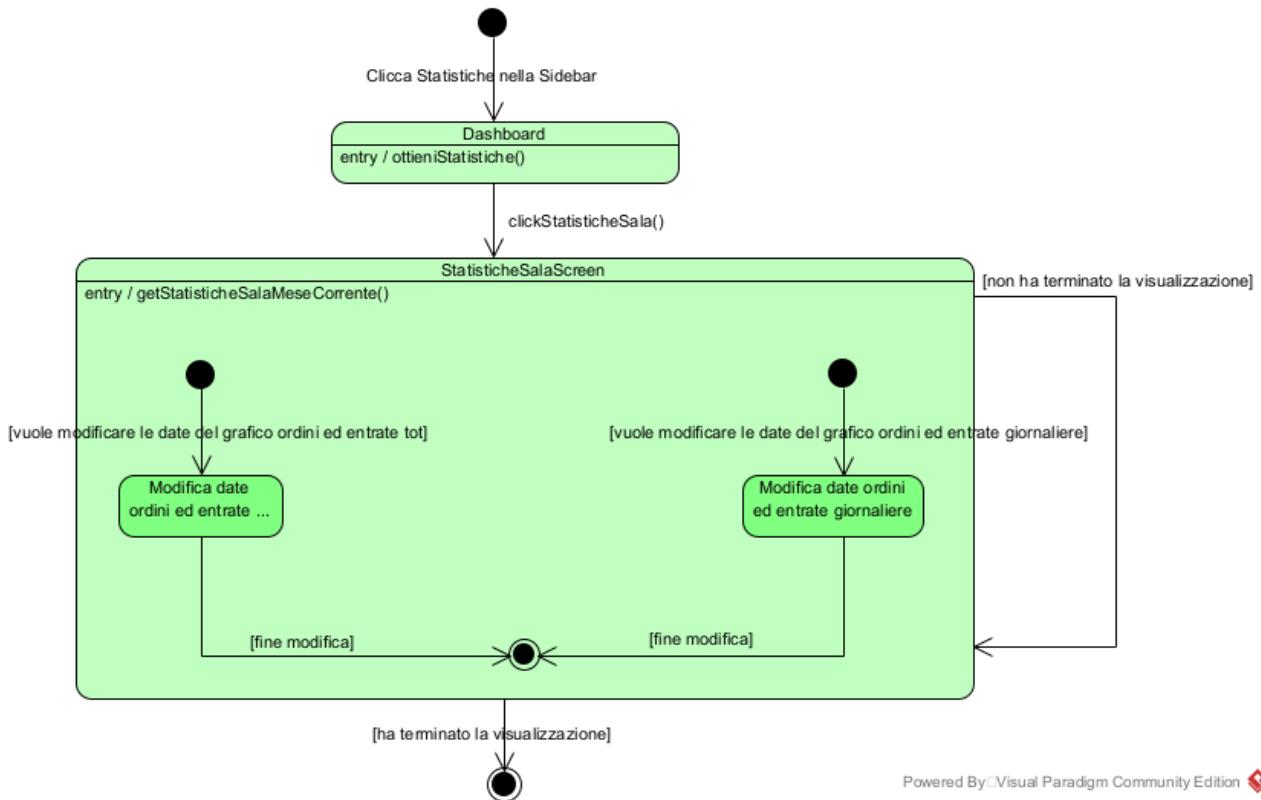
#### 3.3.1 Organizzare menù



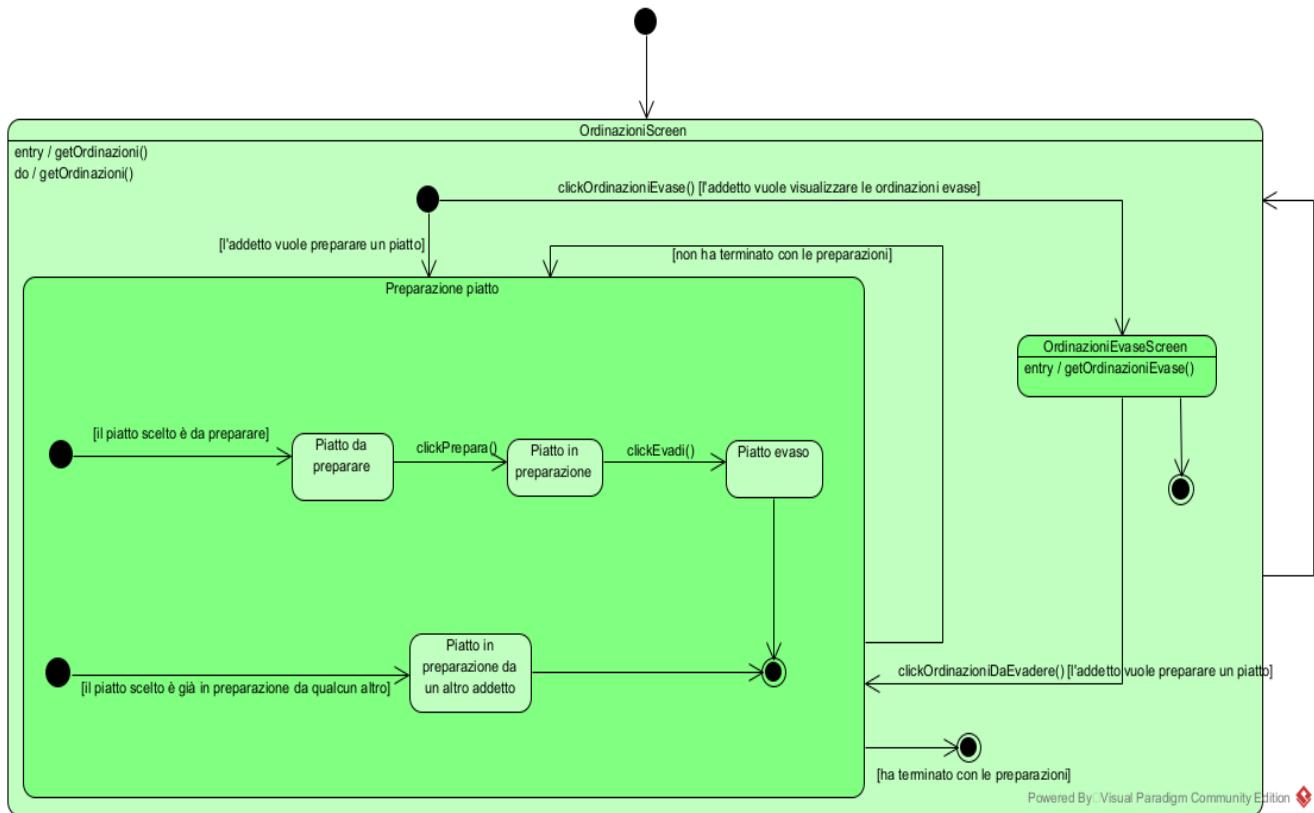
### 3.3.2 Effettuare nuova ordinazione



### 3.3.3 Statistiche addetti alla sala



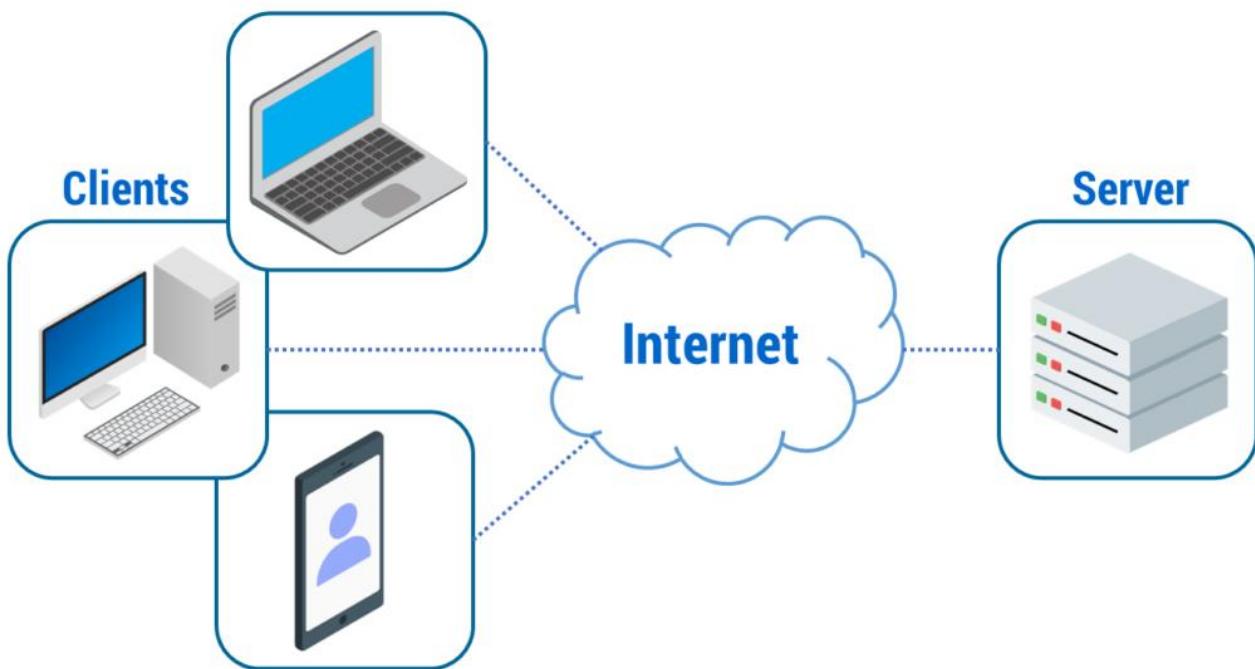
### 3.3.4 Visualizzazione ordinazioni (addetti alla cucina)



# **Documento di Design del Sistema**

## 4 Analisi dell'architettura

L'architettura generale, utilizzata da Ratatouille23, è di tipo **client-server**: i client sono rappresentati dai dispositivi mobili utilizzati dagli addetti e dall'amministratore, mentre il server corrisponde al backend utilizzato per gestire i servizi che Ratatouille23 mette a disposizione.



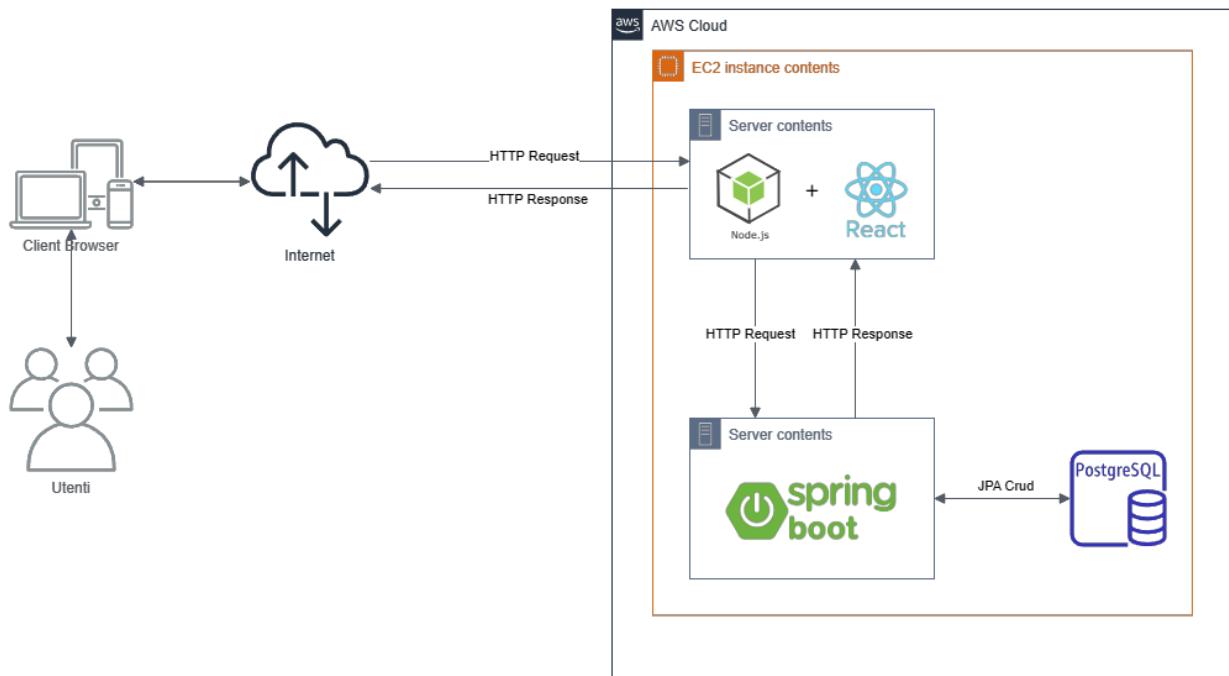
La comunicazione tra client e server avviene tramite Internet, utilizzando il protocollo **HTTP** (*HyperText Transfer Protocol*) e il concetto di **REST Api**.

I principi seguiti durante tutta la progettazione dell'applicativo sono stati i seguenti:

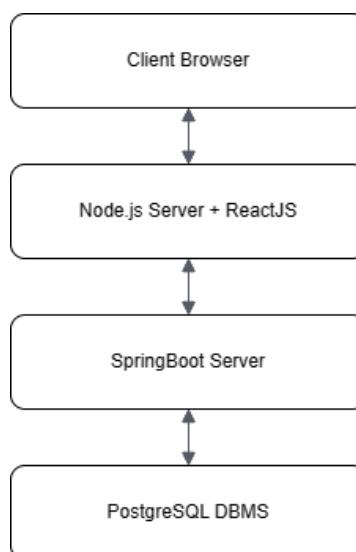
- Elasticità
- Scalabilità
- Astrazione
- Sicurezza
- Economicità
- Virtualizzazione

## 4.1 Architettura cloud

Per il deploy del backend e del frontend si è scelto di utilizzare servizi di **cloud computing**, nello specifico i servizi offerti da **AWS** (*Amazon Web Services*). Per evitare il **vendor lock-in**, cioè dipendere esclusivamente dai servizi offerti da un cloud provider, abbiamo deciso di utilizzare solo il servizio di macchine virtuali Amazon EC2 (*Elastic Compute Cloud*) che ci assicura elasticità e scalabilità dei servizi di Ratatouille23.

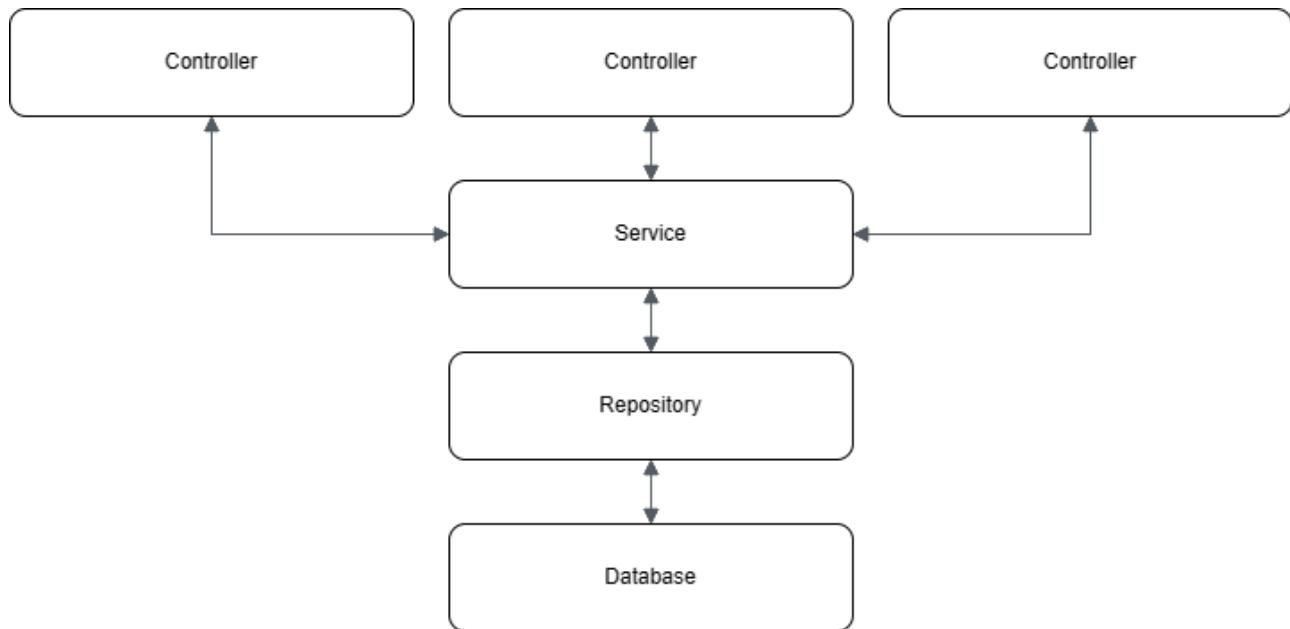


L'architettura generale è quindi a **quattro livelli**:



## 4.2 Il Server SpringBoot

L'architettura del backend SpringBoot è quella consigliata dal framework stesso, abbiamo cioè utilizzato il **pattern architettonico MVC (Model-View-Controller)**



Individuiamo i seguenti componenti:

- **Controller**: gestiscono le rotte su cui ricevono richieste HTTP, le elaborano e restituiscono il risultato tramite risposte HTTP
- **Service**: intermediario tra repository e controller, gestisce la dependency injection e la business logic dell'applicazione
- **Repository**: intermediario tra i service e il database sottostante attraverso ORM di JPA. Semplifica non solo tutte le operazioni CRUD (Create-Read-Update-Delete) sul database ma anche eventuali passaggi ad altri DBMS
- **Model**: rappresentano la struttura dei dati dell'applicazione e contengono le definizioni delle entità che vengono utilizzate per rappresentare e manipolare i dati
- **ModelDTO**: DTO sta per Data Transfer Object e rappresenta i dati in uscita/entrata al server (come body delle risposte/richieste HTTP)

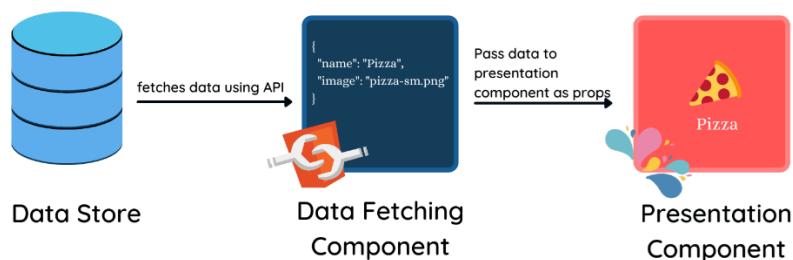
Vediamo quindi quali sono le rotte mappate dai controller:

Route	Descrizione
/api/auth	Route principale per l'autenticazione degli utenti
/api/categoria	Route principale per la gestione delle categorie
/api/categoriamenu	Route principale per la gestione delle categorie all'interno dei menu
/api/dipendente	Route principale per la gestione dei dipendenti
/api/menu	Route principale per la gestione dei menu
/api/ordinazione	Route principale per la gestione delle ordinazioni
/api/piatto	Route principale per la gestione dei piatti
/api/preparazione	Route principale per la gestione delle preparazioni dei piatti
/api/tavolo	Route principale per la gestione dei tavoli virtuali
/api/tavololocale	Route principale per la gestione dei tavoli presenti nell'attività

Ogni rotta ha poi delle sotto-rotte che rappresentano le diverse operazioni che possono essere svolte su uno specifico modello.

## 4.3 Il WebServer ReactJS

Anche in questo caso abbiamo seguito le linee guida consigliate dal framework ReactJS; abbiamo quindi utilizzato il **pattern architettonico Container-View** (o anche Container-Presentation, o Container Pattern). L'idea principale dietro il Container Pattern ruota attorno a due tipologie diverse di View rispettivamente chiamate **Container** e **Presenter/Presentation**. Il Container è responsabile di prendere i dati, ordinarli, filtrarli e altre operazioni e infine passarli al Presenter. I dati viaggiano sempre e solo dal Container al Presenter e mai viceversa.



Oltre ai diversi Container (al più uno per ogni pagina) e i Presenter (almeno uno per ogni pagina) abbiamo anche altri componenti molto importanti:

- **Services**: si occupano di effettuare le chiamate API attraverso protocollo HTTP, aspettano la risposta e la restituiscono al chiamante
- **Hooks**: sono una caratteristica introdotta in React che permette di utilizzare lo stato e altre funzionalità di React all'interno di componenti di tipo funzionale, senza la necessità di utilizzare una classe. Gli Hooks consentono di aggiungere funzionalità di React come lo stato locale, gli effetti collaterali e il context all'interno di un componente funzionale, rendendo il codice più conciso e facile da comprendere
- **Contexts**: sono una funzionalità di React che consente di condividere dati specifici in un albero di componenti senza doverli passare manualmente attraverso le props. I Contexts consentono di creare un "contesto" che può essere accessibile da componenti figlio nidificati all'interno di un componente padre

Per le richieste HTTP abbiamo utilizzato [Axios](#), che ci permette di serializzare/deserializzare gli oggetti in formato JSON (*Javascript Object Notation*). Grazie ad Axios, inoltre, viene gestita anche l'integrazione di API esterne come quelle di [OpenFoodFacts](#).

# 5 Tecnologie adoperate

Le tecnologie che abbiamo adoperato sono le seguenti:

- **SpringBoot**: è un framework per lo sviluppo di applicazioni Java che semplifica la creazione di applicazioni enterprise-ready. Spring Boot riduce la complessità della configurazione e dell'integrazione di componenti, fornendo un ambiente preconfigurato per lo sviluppo rapido di applicazioni Spring. I suoi vantaggi includono una configurazione semplificata, un avvio rapido dell'applicazione, la gestione semplificata delle dipendenze e strumenti integrati per il monitoraggio e la gestione delle applicazioni. È ampiamente utilizzato per la creazione di applicazioni Java scalabili e robuste. Le applicazioni SpringBoot vengono eseguite in un server Apache Tomcat
- **PostgreSQL**: è un potente sistema di gestione di database relazionali open source con funzionalità avanzate, scalabilità e sicurezza. Viene utilizzato per sviluppare applicazioni che richiedono la gestione efficiente di grandi volumi di dati e che richiedono funzionalità avanzate di query e manipolazione dei dati
- **ReactJS**: è una libreria JavaScript per lo sviluppo di interfacce utente reattive. Basato sui componenti, offre un Virtual DOM efficiente, un flusso di dati unidirezionale e JSX per scrivere codice HTML-like. ReactJS è ampiamente utilizzato per creare applicazioni web complesse grazie alla sua scalabilità, modularità e supporto di una vasta comunità. Le applicazioni ReactJS vengono eseguite all'interno di un server Node.js

## 5.1 Motivazioni

Una delle scelte più importanti è stata se sviluppare una web-app o un'applicazione nativa Android. Vediamo le motivazioni che ci hanno portato a preferire la prima alla seconda:

1. **Accessibilità multi-piattaforma**: una web-app è accessibile da diversi dispositivi e sistemi operativi, come computer desktop, laptop, tablet e smartphone, indipendentemente dal sistema operativo. Non è necessario sviluppare e mantenere versioni separate dell'app per ogni piattaforma.
2. **Aggiornamenti più rapidi**: con una web-app, puoi effettuare aggiornamenti in tempo reale sul server e i cambiamenti diventano immediatamente visibili per tutti gli utenti. Al contrario, le app Android richiedono l'invio di aggiornamenti tramite Google Play Store e gli utenti

devono scaricare e installare l'aggiornamento per accedere alle nuove funzionalità.

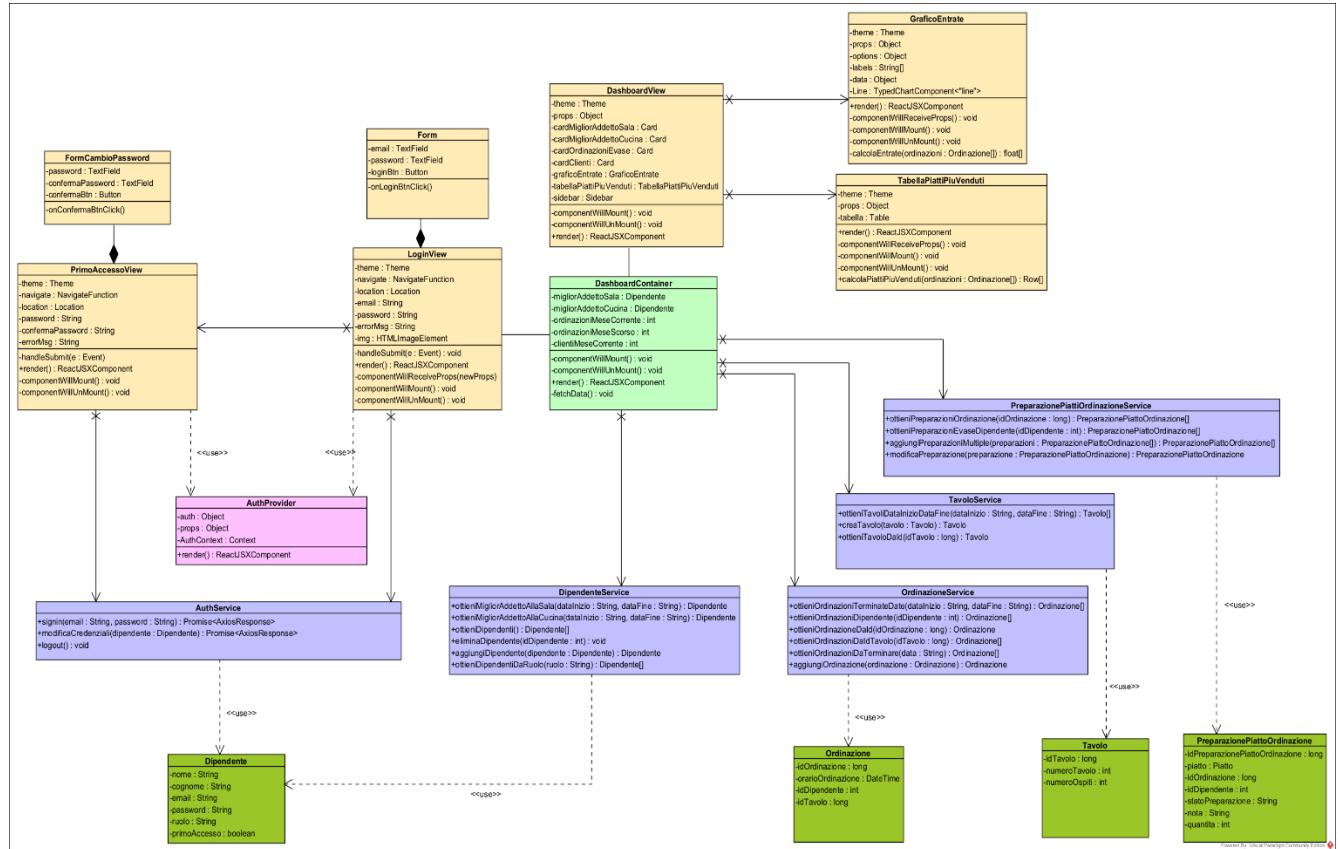
3. **Costi di sviluppo e manutenzione:** sviluppare una web-app può essere meno costoso rispetto allo sviluppo di un'applicazione Android, poiché richiede una singola base di codice per diverse piattaforme. Inoltre, la manutenzione e l'aggiornamento della web-app possono essere più semplici, come scritto al punto 2.
4. **Facilità di distribuzione:** con una web-app, non è necessario passare attraverso il processo di approvazione e pubblicazione sullo store delle app, come richiesto per le app Android. Puoi semplicemente pubblicare la web-app su un server e gli utenti possono accedervi tramite un browser web.
5. **Facilità di condivisione e accessibilità:** le web-app possono essere facilmente condivise tramite URL o link diretti, consentendo agli utenti di accedervi rapidamente senza dover scaricare e installare un'applicazione separata.

Tuttavia, è importante notare che le app Android hanno anche i loro vantaggi, come un accesso più diretto alle funzionalità del dispositivo e la possibilità di fornire un'esperienza utente più nativa.

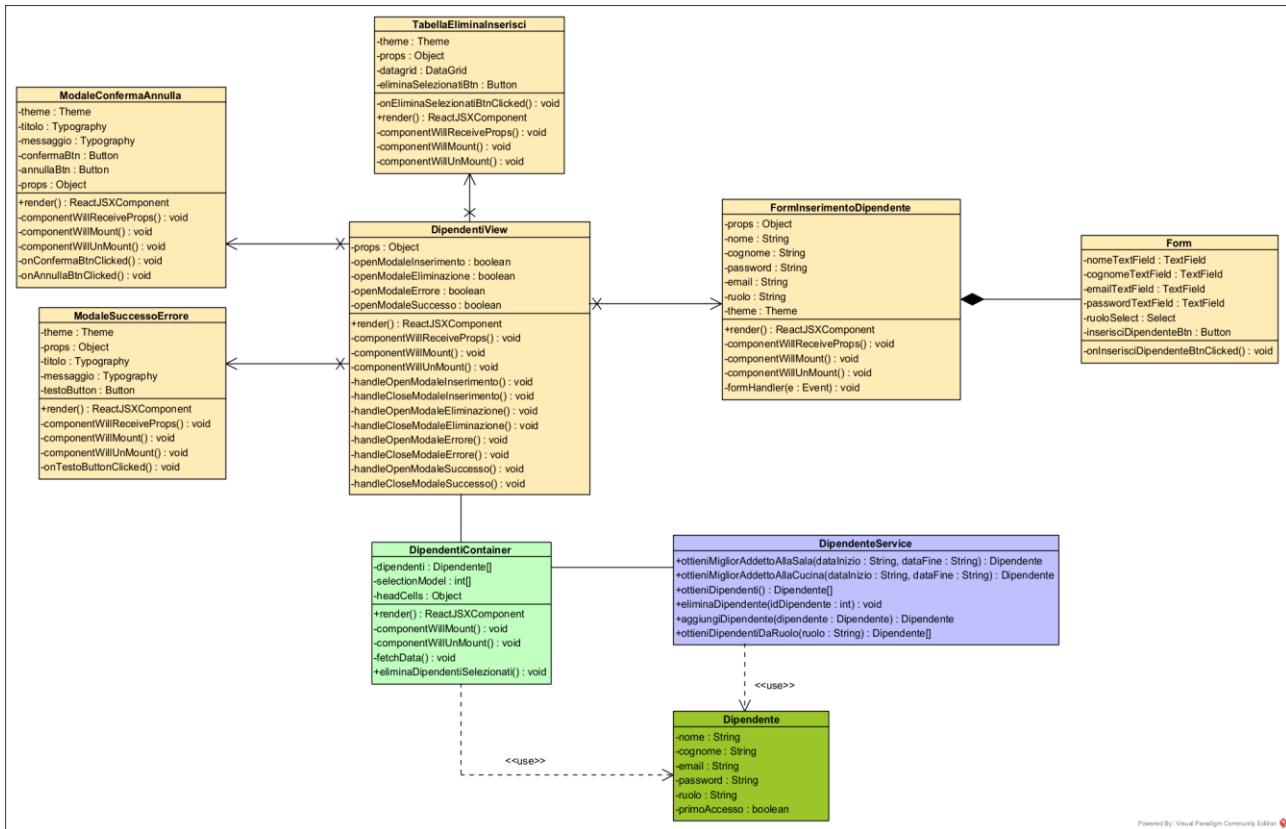
# 6 Class Diagram di Design

I Class Diagram sono suddivisi in funzionalità per questioni d'ordine.\

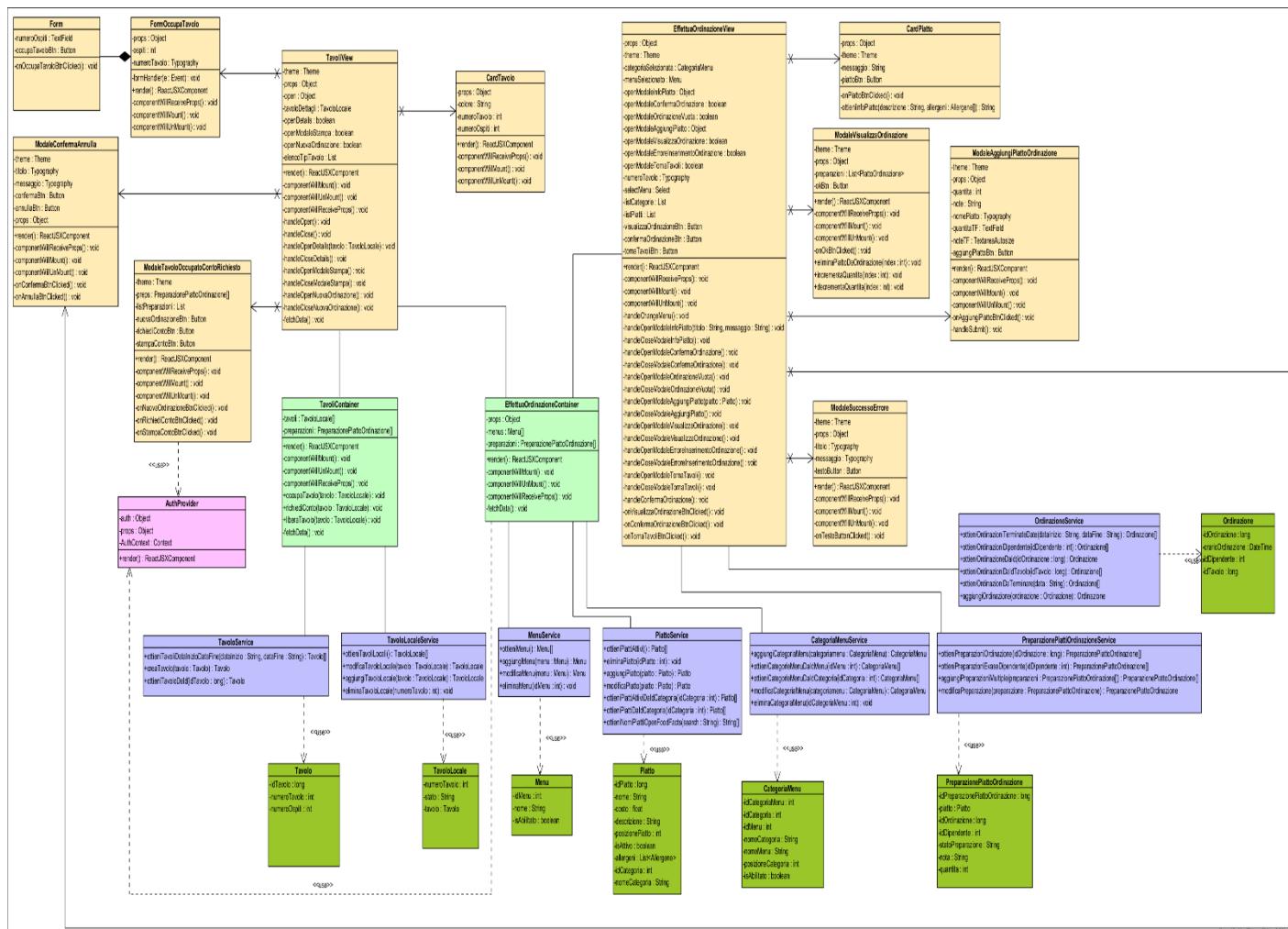
## 6.1 Autenticazione



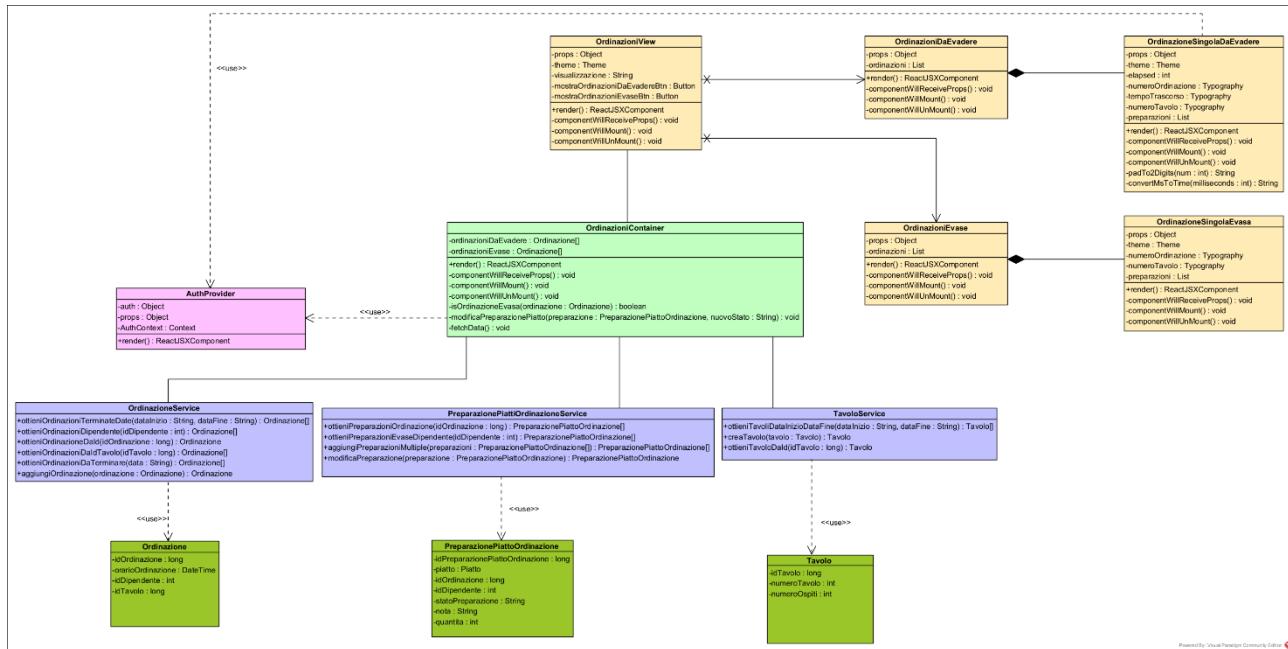
## 6.2 Dipendenti



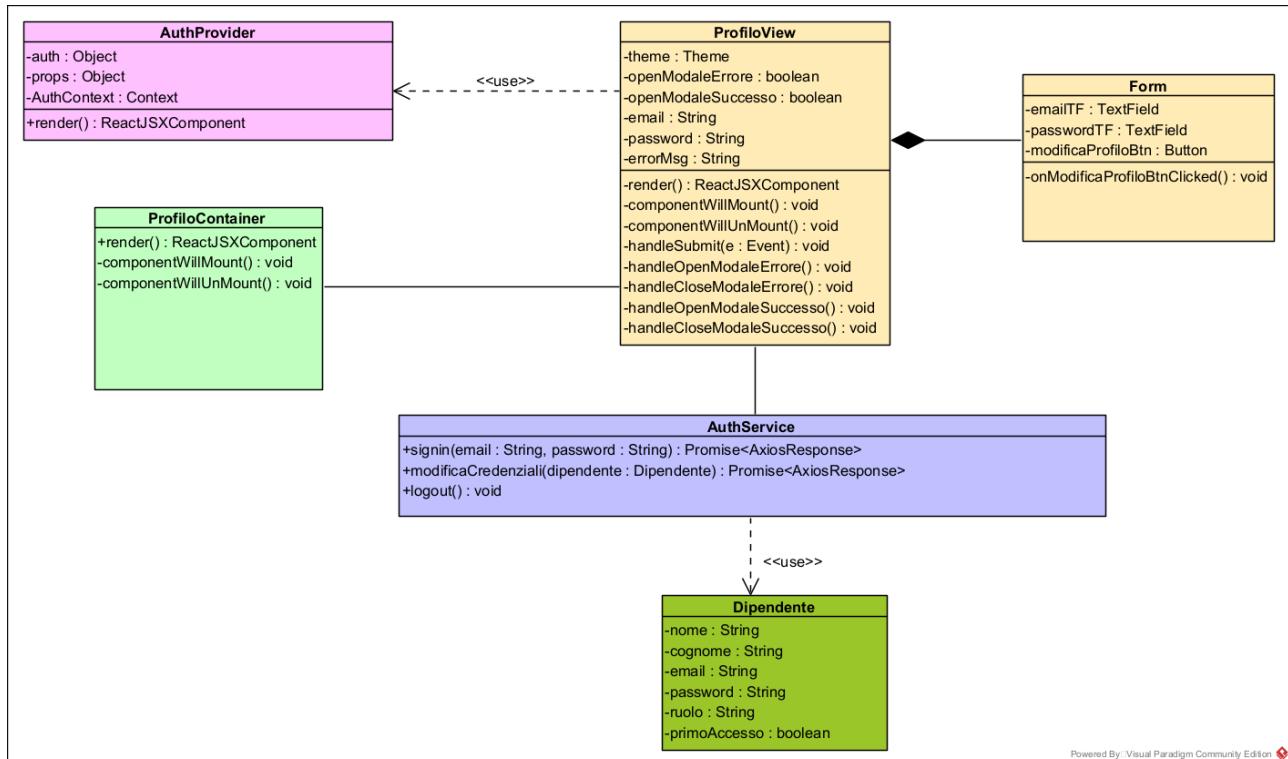
## 6.3 Tavoli e Nuova ordinazione



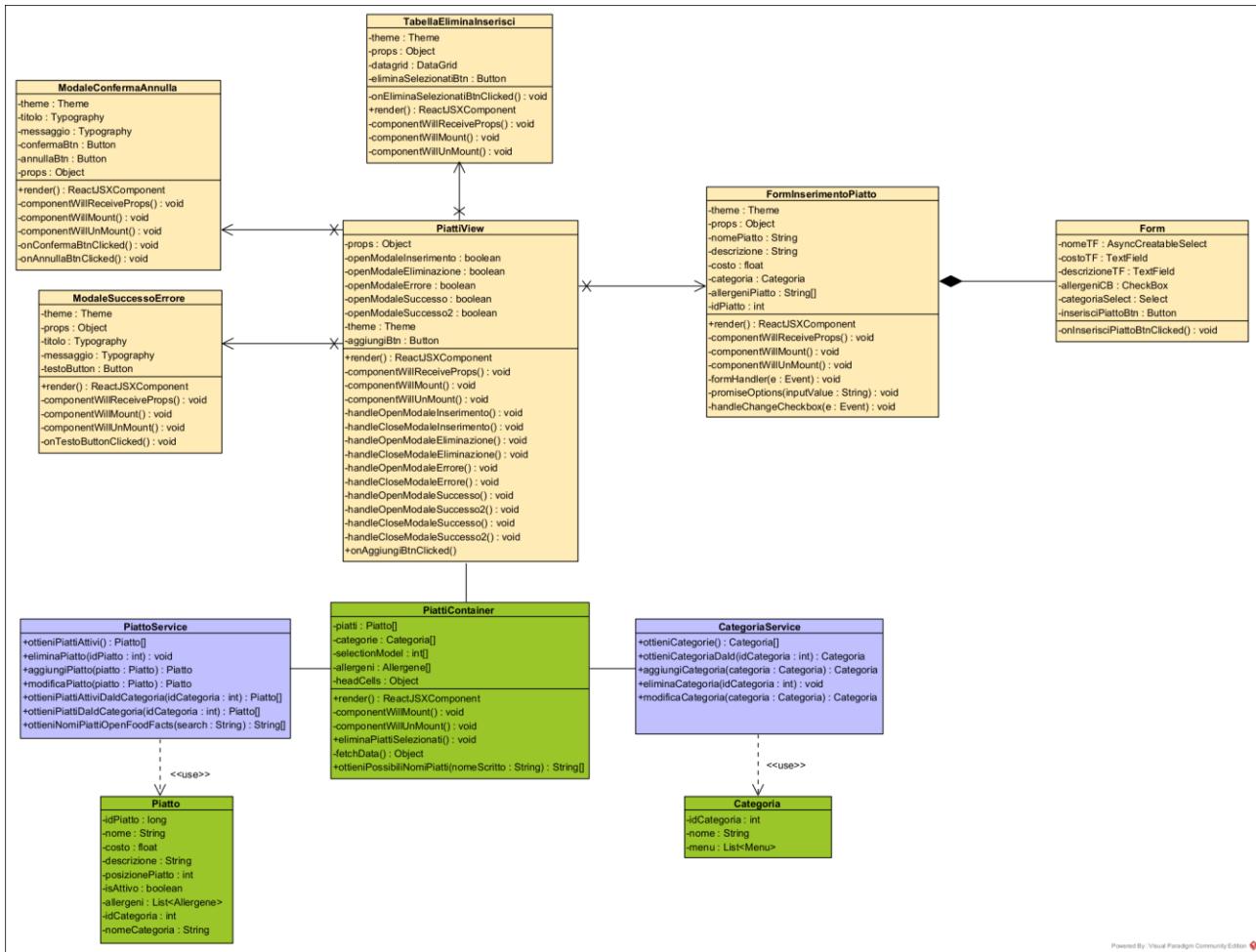
## 6.4 Gestione ordinazioni



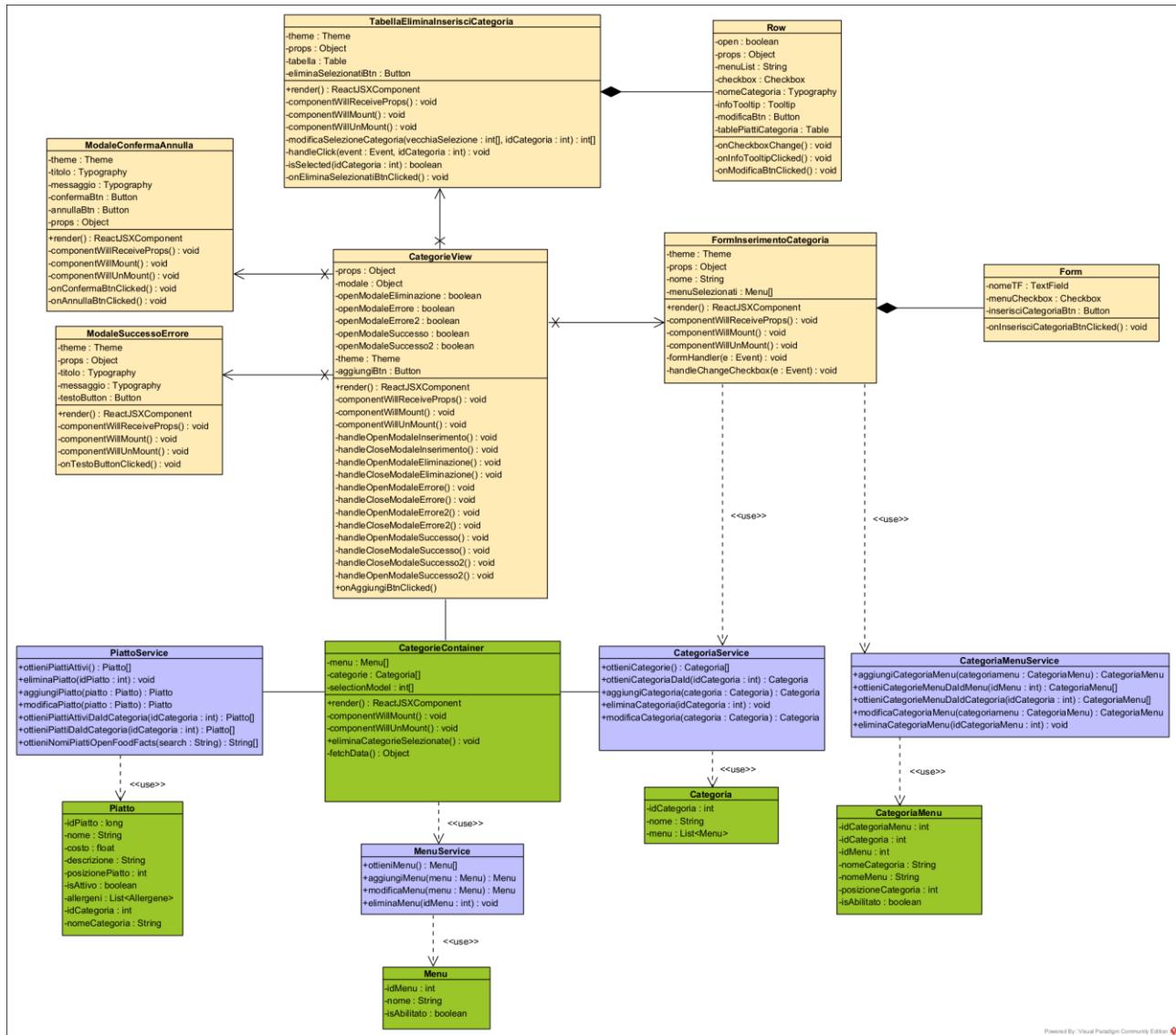
## 6.5 Gestione profilo



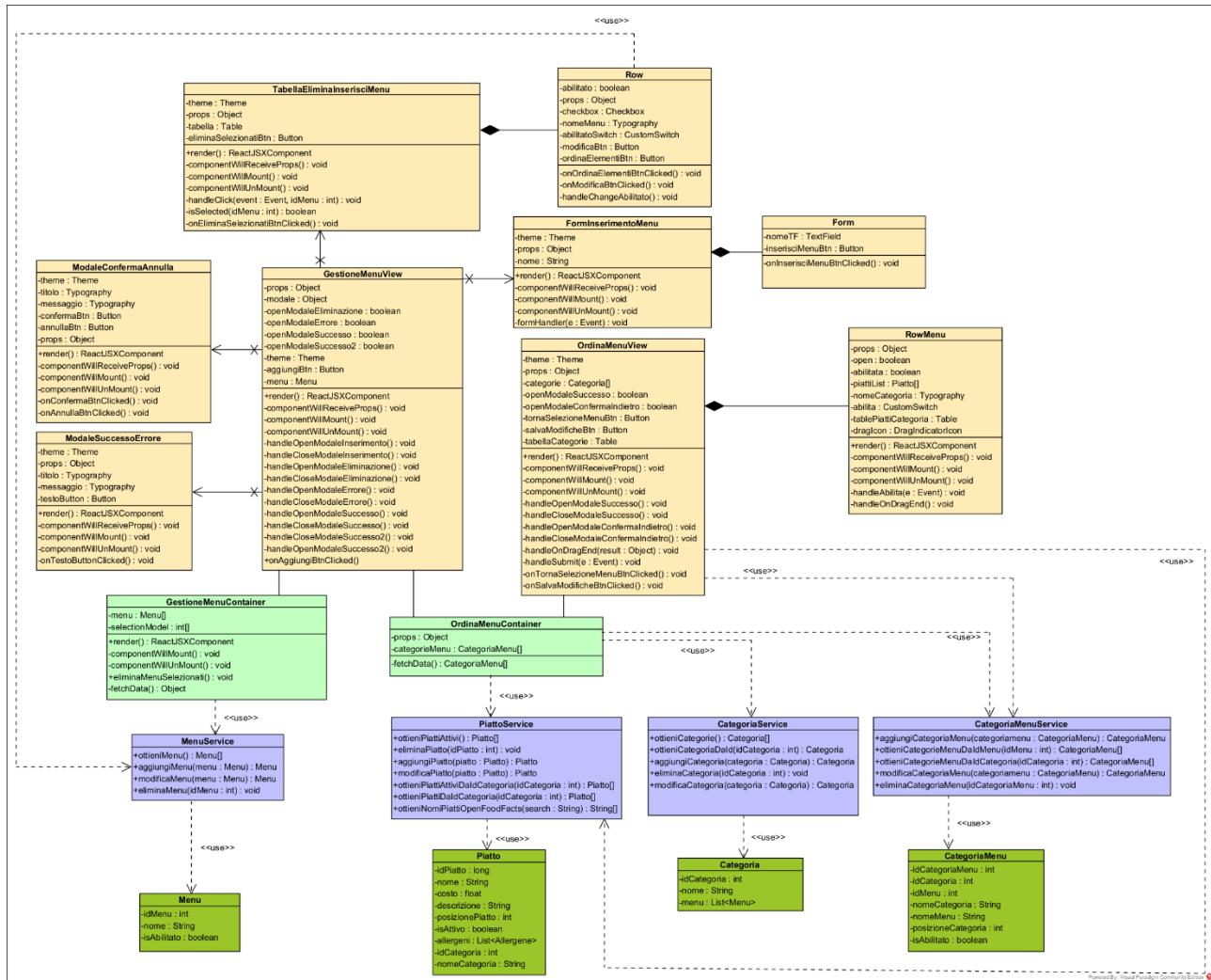
## 6.6 Gestione piatti



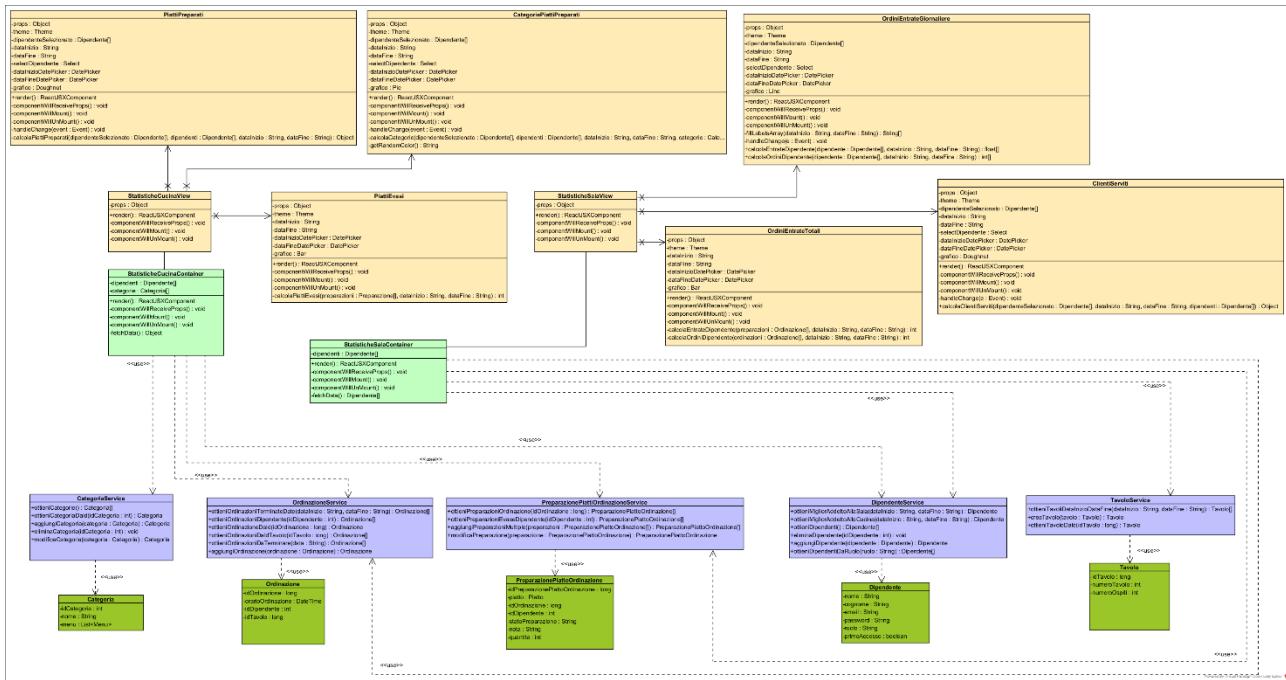
## 6.7 Gestione categorie



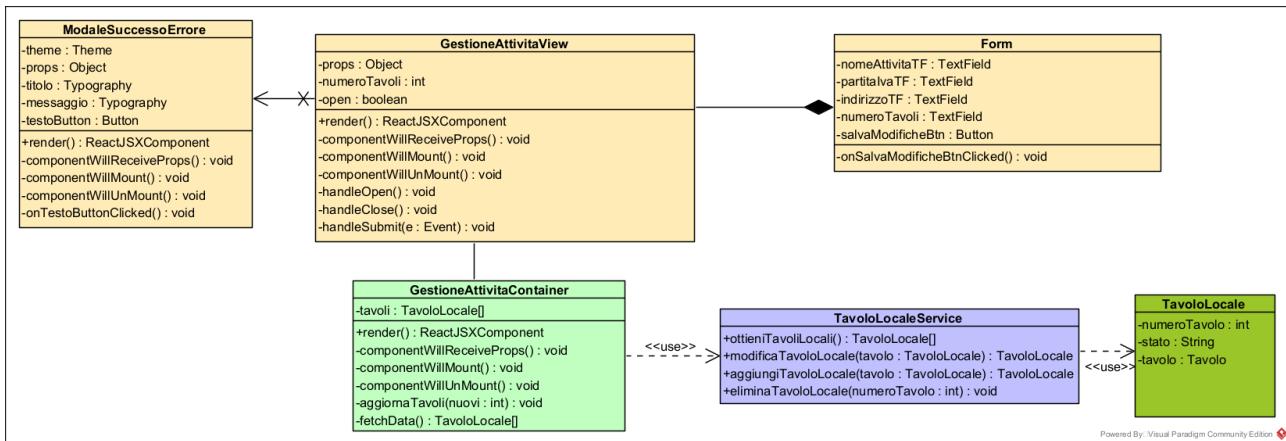
## 6.8 Gestione menù



## 6.9 Statistiche addetti alla sala e cucina

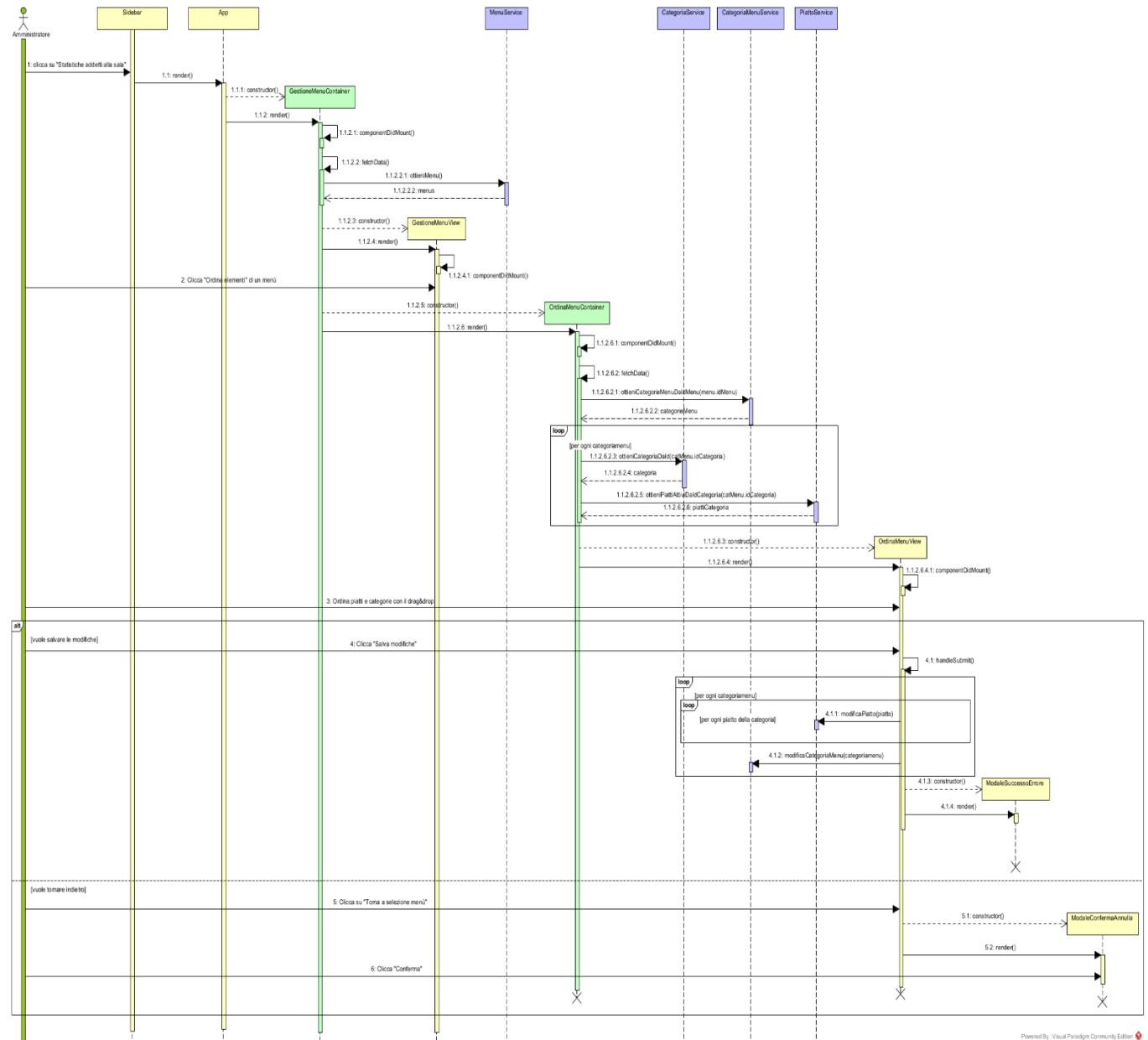


## 6.10 Gestione attività



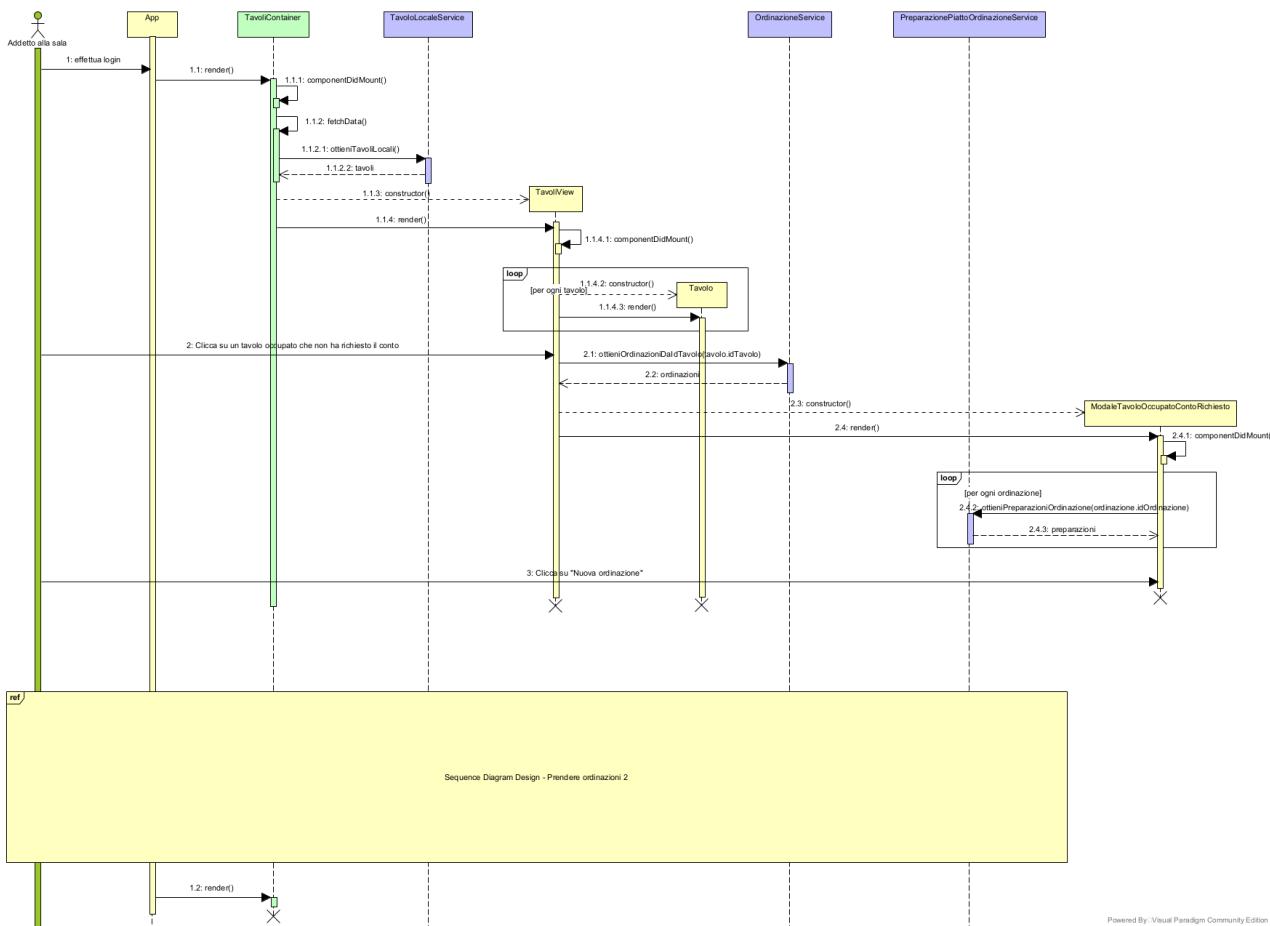
# 7 Sequence Diagram di Design

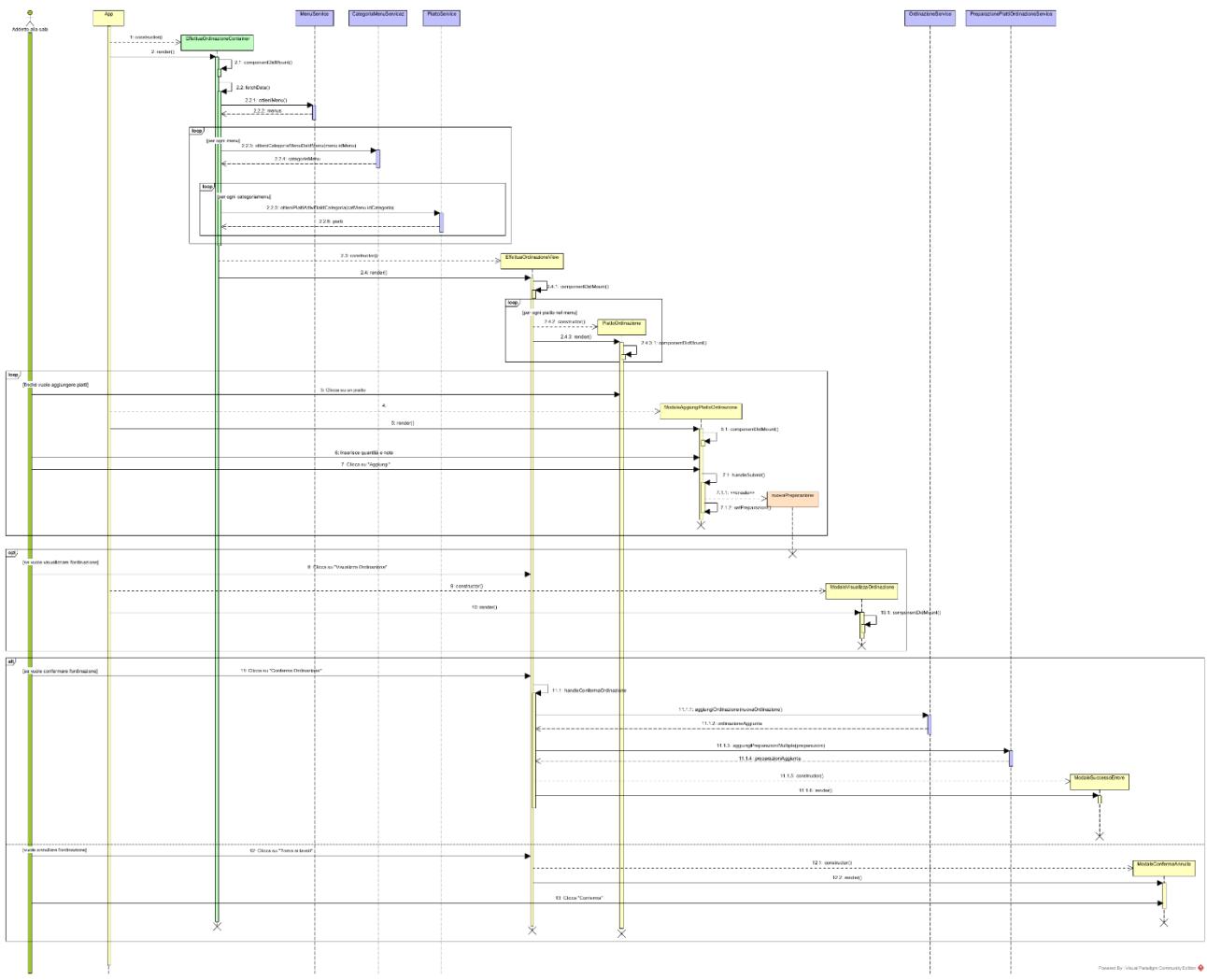
## 7.1 Organizzare menù



Per visualizzare il diagramma in alta definizione: [Sequence Diagram Design – Organizzare menù.png](#).

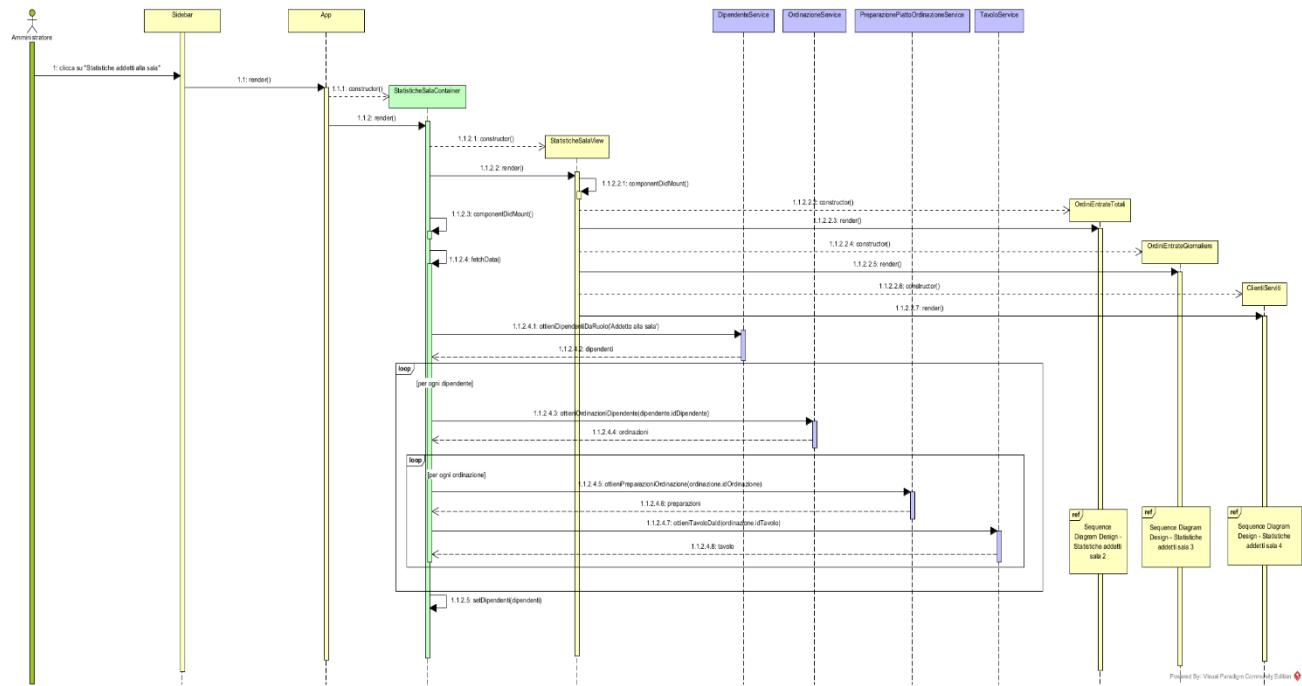
## 7.2 Prendere ordinazioni

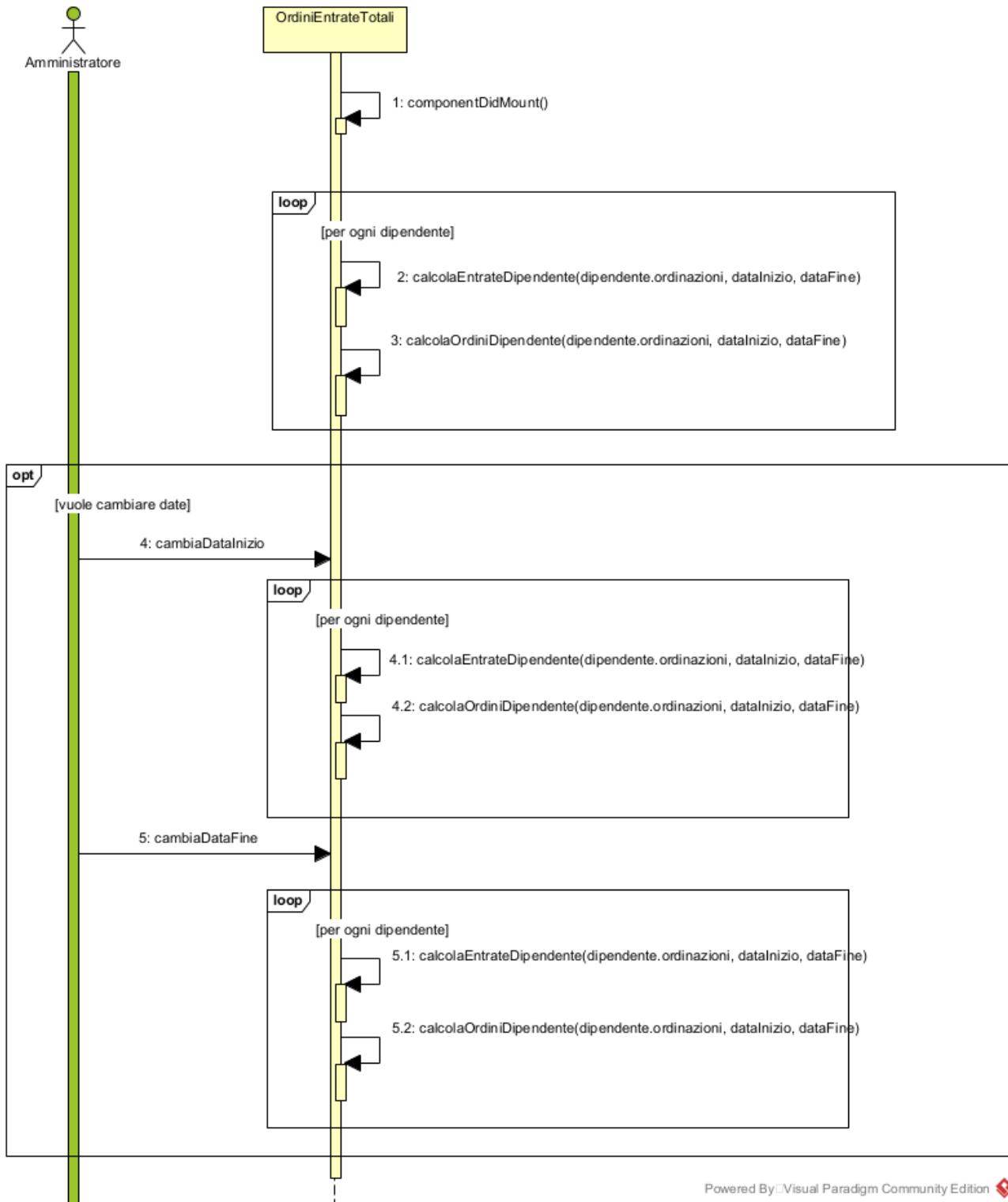




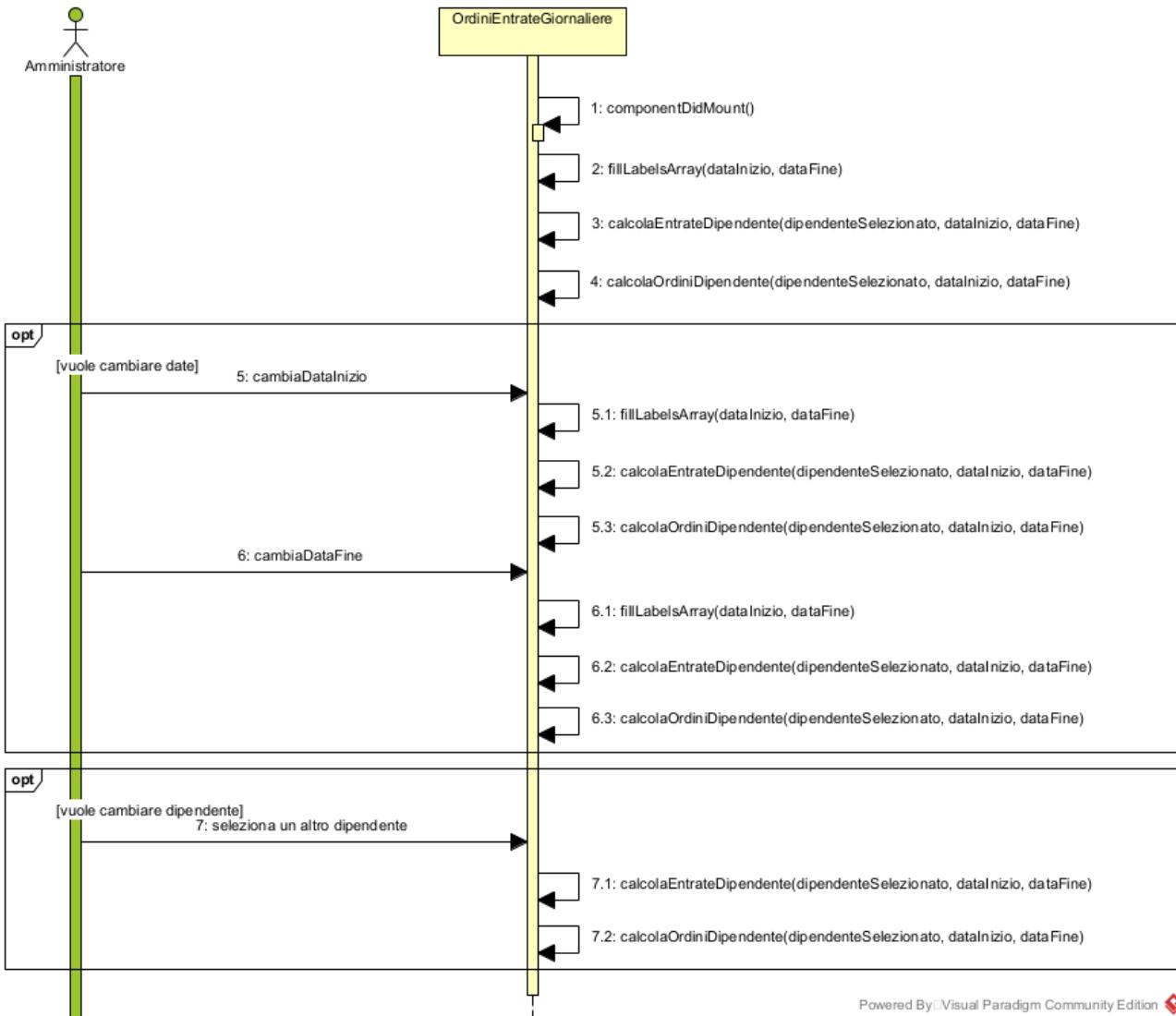
Per visualizzare il diagramma in alta definizione: [Sequence Diagram Design – Prendere ordinazioni 2.png](#).

## 7.3 Statistiche addetti alla sala

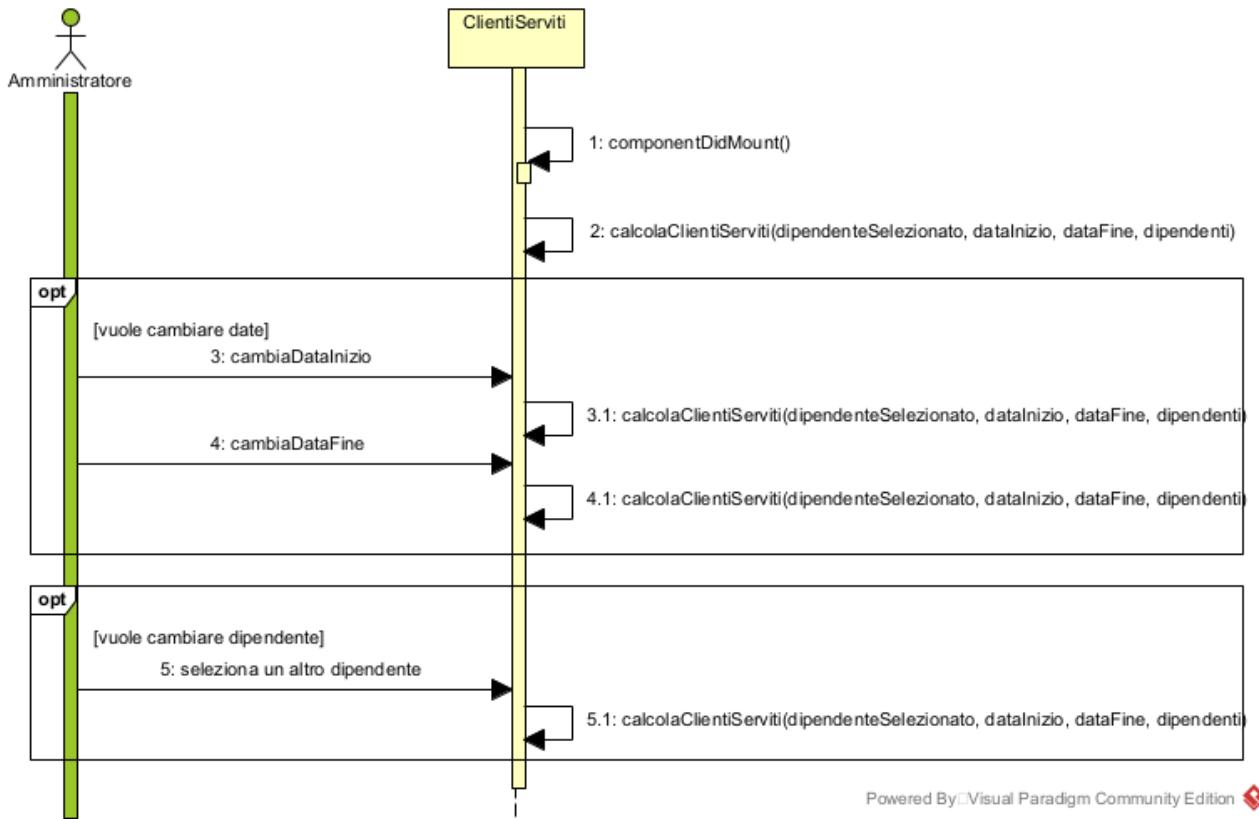




Powered By Visual Paradigm Community Edition

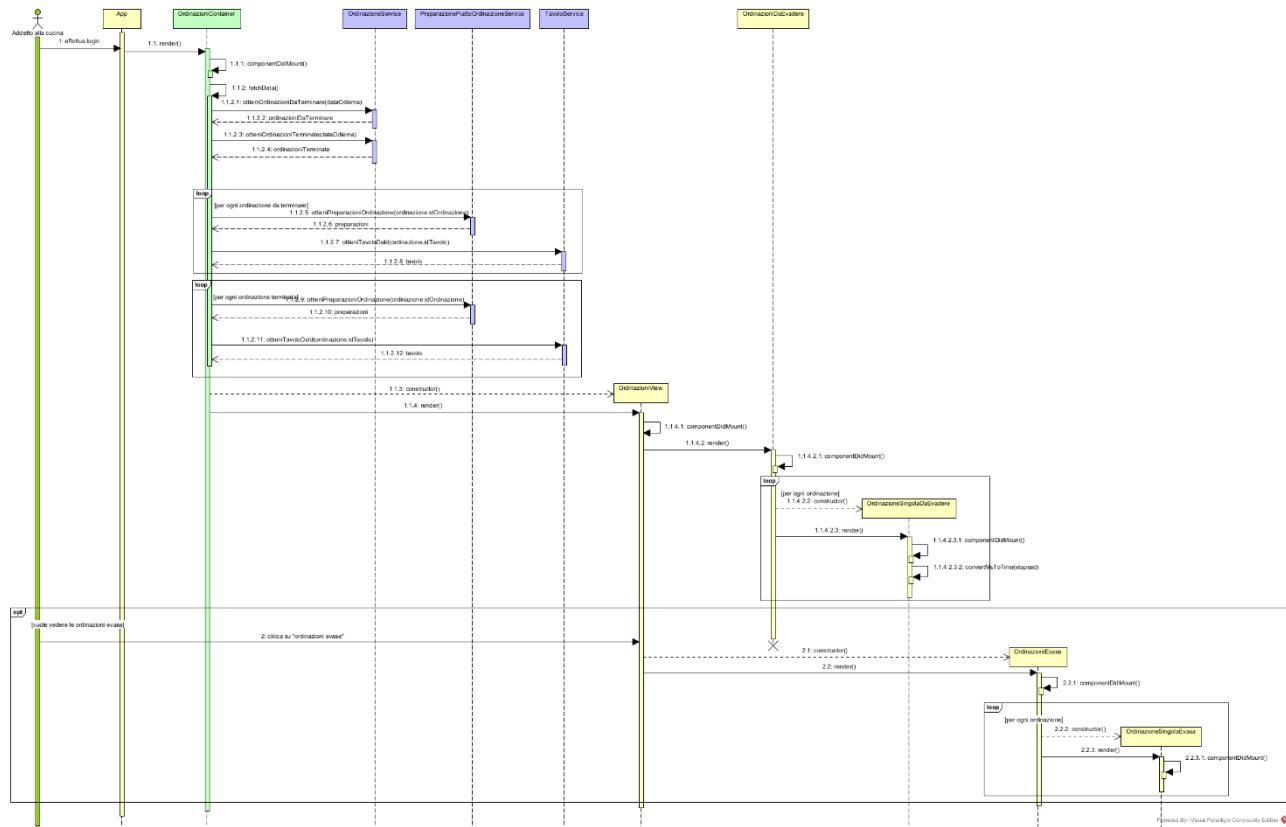


Powered By Visual Paradigm Community Edition



Powered By Visual Paradigm Community Edition

## 7.4 Visualizzazione ordinazioni (addetti alla cucina)



## **Testing e valutazione sul campo dell'usabilità**

## 8 Codice xUnit

Lo Unit Testing è uno strumento estremamente potente nella misurazione della qualità del software. Gli unit test forniscono un controllo fondamentale sul fatto che l'applicazione soddisfi le specifiche di progettazione del software e si comporti come previsto. Se fatto bene, uno unit test può:

- Ridurre i difetti ed esporli all'inizio del ciclo di vita dello sviluppo
- Migliorare la leggibilità del codice
- Facilitare il riuso del codice
- Rendere più veloce il rilascio

A titolo di esempio sono stati testati solo quattro funzionalità, tutte appartenenti all'applicazione del frontend. Per fare ciò è stato utilizzato il framework [Jest](#).

I metodi scelti sono i seguenti:

1. `calcolaNuovaSelezione`: utilizzato per aggiornare correttamente le categorie selezionate (tramite checkbox) dall'utente in caso di eliminazioni multiple
2. `ottieneMessaggio`: utilizzato per calcolare il messaggio da inserire all'interno di un popup mostrato in fase di nuova ordinazione per conoscere descrizione e allergeni di un piatto
3. `calcolaPiattiEvasi`: utilizzato per calcolare il numero di piatti evasi da uno degli addetti alla cucina in un determinato lasso di tempo
4. `calcolaOrdiniDipendente`: utilizzato per calcolare il numero di ordini presi da uno degli addetti alla sala in un determinato lasso di tempo

Per ogni metodo si è optato per testing di tipo Black Box con criterio di copertura **WECT** (*Weak Equivalence Class Testing*). Con questo criterio andiamo a scrivere un metodo di test per ogni classe di equivalenza non valida e un metodo che racchiuda più classi di equivalenza valide possibili. La scelta non è ricaduta su **SECT** (*Strong Equivalence Class Testing*) o altre strategie più robuste perché non sono metodi che possono impattare molto negativamente sull'applicazione in caso di malfunzionamento.

## 8.1 CalcolaNuovaSelezione

Il metodo in esame prende in input 2 parametri: vecchiaSelezione di tipo int[] e idCategoria di tipo int. Le **classi di equivalenza** individuate sono quindi le seguenti:

- vecchiaSelezione: CE1: []
- vecchiaSelezione: CE2: null
- vecchiaSelezione: CE3: undefined
- vecchiaSelezione: CE4: [1,2,3,...]
- vecchiaSelezione: CE5: [1,2,3,..., idCategoria,...]
- idCategoria: CE6: {MinInt...0}
- idCategoria: CE7: {1...MaxInt}

I **casi di test** individuati sono i seguenti:

- TC1: testVettoreVecchiaSelezioneNonDefinito copre CE3, CE7
- TC2: testIdCategoriaMinoreOugualeZero copre CE1, CE6
- TC3: testVettoreVuoto copre CE1, CE7
- TC4: testVettoreNull copre CE2, CE7
- TC5: testIdCategoriaNonPresenteNelVettore copre CE4, CE7
- TC6: testIdCategoriaPresenteNelVettore copre CE5, CE7

```
describe("Test modificaSelezioneCategoria", () => {
  test("testIdCategoriaPresenteNelVettore()", () => {
    const vecchiaSelezione = [1,2,3,4];
    const idCategoria = 4;

    const output = [1,2,3];

    expect(modificaSelezioneCategoria(vecchiaSelezione, idCategoria)).toEqual(output);

  });

  test("testIdCategoriaNonPresenteNelVettore()", () => {
    const vecchiaSelezione = [1,2,3];
    const idCategoria = 4;

    const output = [1,2,3,4];

    expect(modificaSelezioneCategoria(vecchiaSelezione, idCategoria)).toEqual(output);

  });

  test("testVettoreNull()", () => {
    const vecchiaSelezione = null;
    const idCategoria = 1;

    const output = [];

    expect(modificaSelezioneCategoria(vecchiaSelezione, idCategoria)).toEqual(output);

  });
};
```

```
test("testVettoreVuoto()", () => {
    const vecchiaSelezione = [];
    const idCategoria = 1;

    const output = [1];

    expect(modificaSelezioneCategoria(vecchiaSelezione, idCategoria)).toEqual(output);

});

test("testIdCategoriaMinore0UgualeZero()", () => {
    const vecchiaSelezione = [];
    const idCategoria = 0;

    const output = [];

    expect(modificaSelezioneCategoria(vecchiaSelezione, idCategoria)).toEqual(output);

});

test("testVettoreVecchiaSelezioneNonDefinito()", () => {
    const vecchiaSelezione = undefined;
    const idCategoria = 1;

    const output = [];

    expect(modificaSelezioneCategoria(vecchiaSelezione, idCategoria)).toEqual(output);

});
});
```

## 8.2 OttieniInfoPiatto

Il metodo in esame prende in input 2 parametri: descrizione di tipo String e allergeni di tipo Allergene[]. Le **classi di equivalenza** individuate sono quindi le seguenti:

- descrizione: CE1: null
- descrizione: CE2: undefined
- descrizione: CE3: “”
- descrizione: CE4: {testo}
- allergeni: CE5: null
- allergeni: CE6: undefined
- allergeni: CE7: []
- allergeni: CE8: [“Glutine”, ...]

I **casi di test** individuati sono i seguenti:

- TC1: testDescrizioneEAllergeniNull copre CE1, CE5
- TC2: testDescrizioneEAllergeniUndefined copre CE2, CE6
- TC3: testDescrizioneVuotaENessunAllergene copre CE3, CE7
- TC4: testDescrizioneEAllergeniPresenti copre CE4, CE8

```

describe("Test ottieniInfoPiatto", () => {
  test("testDescrizioneEAllergeniNull()", () => {
    const descrizione = null;
    const allergeni = null;

    const output = "Nessuna descrizione specificata\nAllergeni: Nessuno";

    expect(ottieniInfoPiatto(descrizione, allergeni)).toEqual(output);
  });

  test("testDescrizioneEAllergeniUndefined()", () => {
    const descrizione = undefined;
    const allergeni = undefined;

    const output = "Nessuna descrizione specificata\nAllergeni:
Nessuno";
    expect(ottieniInfoPiatto(descrizione, allergeni)).toEqual(output);
  });

  test("testDescrizioneVuotaENessunAllergene()", () => {
    const descrizione = "";
    const allergeni = [];

    const output = "Nessuna descrizione specificata\nAllergeni:
Nessuno";
    expect(ottieniInfoPiatto(descrizione, allergeni)).toEqual(output);
  });

  test("testDescrizioneEAllergeniPresenti()", () => {
    const descrizione = "Descrizione di prova";
    const allergeni = ["Glutine", "Molluschi"];

    const output = "Descrizione di prova\nAllergeni: Glutine,
Molluschi";
    expect(ottieniInfoPiatto(descrizione, allergeni)).toEqual(output);
  });
});

```

## 8.3 CalcolaPiattiEvasi

Il metodo in esame prende in input 3 parametri: preparazioni di tipo PreparazionePiattoOrdinazione[], dataInizio di tipo String e dataFine di tipo String. Le **classi di equivalenza** individuate sono quindi le seguenti:

- dataInizio: CE1: null
- dataInizio: CE2: undefined
- dataInizio: CE3: “”
- dataInizio: CE4: {stringhe che non rappresentano date, es: “ciao”}
- dataInizio: CE5: {date errate, es: 31 febbraio}
- dataInizio: CE6: {date corrette}
- dataFine: CE7: null
- dataFine: CE8: undefined
- dataFine: CE9: “”
- dataFine: CE10: {stringhe che non rappresentano date, es: “ciao”}
- dataFine: CE11: {date errate, es: 31 febbraio}
- dataFine: CE12: {date corrette}
- preparazioni: CE13: null
- preparazioni: CE14: undefined
- preparazioni: CE15: []
- preparazioni: CE16: vettore pieno
- dataInizio: CE17: {dataInizio <= dataFine}
- dataInizio: CE18: {dataInizio > dataFine}

I **casi di test** individuati sono i seguenti:

- TC1: testDataInizioNull copre CE1, CE12, CE16
- TC2: testDataInizioUndefined copre CE2, CE12, CE16
- TC3: testDataInizioVuota copre CE3, CE12, CE16
- TC4: testDataInizioNonRappresentativa copre CE4, CE12, CE16
- TC5: testDataInizioInesistente copre CE5, CE12, CE16
- TC6: testDataFineNull copre CE7, CE6, CE16
- TC7: testDataFineUndefined copre CE8, CE6, CE16
- TC8: testDataFineVuota copre CE9, CE6, CE16
- TC9: testDataFineNonRappresentativa copre CE10, CE6, CE16
- TC10: testDataFineInesistente copre CE11, CE6, CE16
- TC11: testDataFineAntecedenteDataInizio copre CE18, CE16, CE12, CE6
- TC12: testDataCorretteVettoreNull copre CE6, CE12, CE13, CE17

- TC13: testDateCorretteVettoreUndefined copre CE6, CE12, CE14, CE17
- TC14: testDateCorretteVettoreVuoto copre CE6, CE12, CE15, CE17
- TC15: testDateCorretteVettorePieno copre CE6, CE12, CE16, CE17

```

describe("Test calcolaPiattiEvasi", () => {
  test("testDataInizioNull()", () => {
    const dataInizio = null;
    const dataFine = "2023-06-29";
    const preparazioni = [ ... ];

    const output = 0;

    expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
  });

  test("testDataInizioUndefined()", () => {
    const dataInizio = undefined;
    const dataFine = "2023-01-22";
    const preparazioni = [ ... ];

    const output = 0;

    expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
  });

  test("testDataInizioVuota()", () => {
    const dataInizio = "";
    const dataFine = "2023-01-22";
    const preparazioni = [ ... ];

    const output = 0;

    expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
  });

  test("testDataInizioNonRappresentativa()", () => {
    const dataInizio = "ciao";
    const dataFine = "2023-01-22";
    const preparazioni = [ ... ];

    const output = 0;

    expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
  });
}

```

```

test("testDataInizioInesistente()", () => {
  const dataInizio = "2023-02-31";
  const dataFine = "2023-06-22";
  const preparazioni = [ ... ];

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineNull()", () => {
  const dataInizio = "2023-01-22";
  const dataFine = null;
  const preparazioni = [ ... ];

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineUndefined()", () => {
  const dataInizio = "2023-01-22";
  const dataFine = undefined;
  const preparazioni = [ ... ];

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineVuota()", () => {
  const dataInizio = "2023-01-22";
  const dataFine = "";
  const preparazioni = [ ... ];

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineNonRappresentativa()", () => {
  const dataInizio = "2023-01-22";
  const dataFine = "ciao";
  const preparazioni = [ ... ];

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

```

```

test("testDataFineInesistente()", () => {
  const dataInizio = "2023-06-01";
  const dataFine = "2023-06-31";
  const preparazioni = [ ... ];

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineAntecedenteDataInizio()", () => {
  const dataInizio = "2023-06-16";
  const dataFine = "2023-06-12";
  const preparazioni = [ ... ];

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDateCorretteVettoreNull()", () => {
  const dataInizio = "2023-06-12";
  const dataFine = "2023-06-16";
  const preparazioni = null;

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDateCorretteVettoreUndefined()", () => {
  const dataInizio = "2023-06-12";
  const dataFine = "2023-06-16";
  const preparazioni = undefined;

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDateCorretteVettoreVuoto()", () => {
  const dataInizio = "2023-06-12";
  const dataFine = "2023-06-16";
  const preparazioni = [];

  const output = 0;

  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

```

```

test("testDateCorretteVettorePieno()", () => {
  const dataInizio = "2023-06-12";
  const dataFine = "2023-06-16";
  const preparazioni = [
    {
      idPreparazionePiattoOrdinazione: 9,
      piatto: {
        idPiatto: 1,
        nome: "TestPiatto",
        costo: 10,
        descrizione: "Descrizione",
        posizionePiatto: 1,
        categoria: "CategoriaTest",
        idCategoria: 18,
        allergeni: ["Arachidi", "Frutta a guscio"],
        attivo: true,
      },
      idOrdinazione: 9,
      idDipendente: 2,
      statoPreparazione: "Evaso",
      quantita: 3,
      nota: "",
      ordinazione: {
        idOrdinazione: 9,
        orarioOrdinazione: "2023-06-14T15:38:52",
        idDipendente: 1,
        idTavolo: 2,
      },
    },
    {
      idPreparazionePiattoOrdinazione: 10,
      piatto: {
        idPiatto: 1,
        nome: "TestPiatto",
        costo: 10,
        descrizione: "Descrizione",
        posizionePiatto: 1,
        categoria: "CategoriaTest",
        idCategoria: 18,
        allergeni: ["Arachidi", "Frutta a guscio"],
        attivo: true,
      },
      idOrdinazione: 10,
      idDipendente: 2,
      statoPreparazione: "Evaso",
      quantita: 2,
      nota: "",
      ordinazione: {
        idOrdinazione: 10,
        orarioOrdinazione: "2023-06-14T15:38:52",
        idDipendente: 1,
        idTavolo: 2,
      },
    },
    {
      idPreparazionePiattoOrdinazione: 11,
      piatto: {
        idPiatto: 1,
        nome: "TestPiatto",
        costo: 10,
        descrizione: "Descrizione",
        posizionePiatto: 1,
        categoria: "CategoriaTest",
        idCategoria: 18,
        allergeni: ["Arachidi", "Frutta a guscio"],
        attivo: true,
      },
      idOrdinazione: 11,
      idDipendente: 2,
      statoPreparazione: "Evaso",
      quantita: 1,
      nota: "",
      ordinazione: {
        idOrdinazione: 11,
        orarioOrdinazione: "2023-06-14T15:38:52",
        idDipendente: 1,
        idTavolo: 2,
      },
    },
  ];
  const output = 6;
  expect(calcolaPiattiEvasi(preparazioni, dataInizio, dataFine)).toEqual(output);
});

```

## 8.4 CalcolaOrdiniDipendente

Il metodo in esame prende in input 3 parametri: ordinazioni di tipo `Ordinazione[]`, `dataInizio` di tipo `String` e `dataFine` di tipo `String`. Le **classi di equivalenza** individuate sono quindi le seguenti:

- `dataInizio: CE1: null`
- `dataInizio: CE2: undefined`
- `dataInizio: CE3: ""`
- `dataInizio: CE4: {stringhe che non rappresentano date, es: "ciao"}`
- `dataInizio: CE5: {date errate, es: 31 febbraio}`
- `dataInizio: CE6: {date corrette}`
- `dataFine: CE7: null`
- `dataFine: CE8: undefined`
- `dataFine: CE9: ""`
- `dataFine: CE10: {stringhe che non rappresentano date, es: "ciao"}`
- `dataFine: CE11: {date errate, es: 31 febbraio}`
- `dataFine: CE12: {date corrette}`
- `ordinazioni: CE13: null`
- `ordinazioni: CE14: undefined`
- `ordinazioni: CE15: []`
- `ordinazioni: CE16: vettore pieno`
- `dataInizio: CE17: {dataInizio <= dataFine}`
- `dataInizio: CE18: {dataInizio > dataFine}`

I **casi di test** individuati sono i seguenti:

- TC1: `testDataInizioNull` copre CE1, CE12, CE16
- TC2: `testDataInizioUndefined` copre CE2, CE12, CE16
- TC3: `testDataInizioVuota` copre CE3, CE12, CE16
- TC4: `testDataInizioNonRappresentativa` copre CE4, CE12, CE16
- TC5: `testDataInizioInesistente` copre CE5, CE12, CE16
- TC6: `testDataFineNull` copre CE7, CE6, CE16
- TC7: `testDataFineUndefined` copre CE8, CE6, CE16
- TC8: `testDataFineVuota` copre CE9, CE6, CE16
- TC9: `testDataFineNonRappresentativa` copre CE10, CE6, CE16
- TC10: `testDataFineInesistente` copre CE11, CE6, CE16
- TC11: `testDataFineAntecedenteDataInizio` copre CE18, CE16, CE12, CE6
- TC12: `testDateCorretteVettoreNull` copre CE6, CE12, CE13, CE17

- TC13: testDateCorretteVettoreUndefined copre CE6, CE12, CE14, CE17
- TC14: testDateCorretteVettoreVuoto copre CE6, CE12, CE15, CE17
- TC15: testDateCorretteVettorePieno copre CE6, CE12, CE16, CE17

```

describe("Test calcolaOrdiniDipendente", () => {
  test("testDataInizioNull()", () => {
    const dataInizio = null;
    const dataFine = "2023-06-29";
    const ordinazioni = [ ... ];

    const output = 0;

    expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
  });

  test("testDataInizioUndefined()", () => {
    const dataInizio = undefined;
    const dataFine = "2023-01-22";
    const ordinazioni = [ ... ];

    const output = 0;

    expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
  });

  test("testDataInizioVuota()", () => {
    const dataInizio = "";
    const dataFine = "2023-01-22";
    const ordinazioni = [ ... ];

    const output = 0;

    expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
  });

  test("testDataInizioNonRappresentativa()", () => {
    const dataInizio = "ciao";
    const dataFine = "2023-01-22";
    const ordinazioni = [ ... ];

    const output = 0;

    expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
  });
}

```

```
test("testDataInizioInesistente()", () => {
  const dataInizio = "2023-02-31";
  const dataFine = "2023-06-22";
  const ordinazioni = [ ... ];

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineNull()", () => {
  const dataInizio = "2023-01-22";
  const dataFine = null;
  const ordinazioni = [ ... ];

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineUndefined()", () => {
  const dataInizio = "2023-01-22";
  const dataFine = undefined;
  const ordinazioni = [ ... ];

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineVuota()", () => {
  const dataInizio = "2023-01-22";
  const dataFine = "";
  const ordinazioni = [ ... ];

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});
```

```
test("testDataFineNonRappresentativa()", () => {
  const dataInizio = "2023-01-22";
  const dataFine = "ciao";
  const ordinazioni = [ ... ];

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineInesistente()", () => {
  const dataInizio = "2023-06-01";
  const dataFine = "2023-06-31";
  const ordinazioni = [ ... ];

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDataFineAntecedenteDataInizio()", () => {
  const dataInizio = "2023-06-16";
  const dataFine = "2023-06-12";
  const ordinazioni = [ ... ];

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDateCorretteVettoreNull()", () => {
  const dataInizio = "2023-06-12";
  const dataFine = "2023-06-16";
  const ordinazioni = null;

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});
```

```
test("testDateCorretteVettoreUndefined()", () => {
  const dataInizio = "2023-06-12";
  const dataFine = "2023-06-16";
  const ordinazioni = undefined;

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});

test("testDateCorretteVettoreVuoto()", () => {
  const dataInizio = "2023-06-12";
  const dataFine = "2023-06-16";
  const ordinazioni = [];

  const output = 0;

  expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
});
```

```

    test("testDateCorretteVettorePieno()", () => {
      const dataInizio = "2023-06-12";
      const dataFine = "2023-06-16";
      const ordinazioni = [
        {
          "idOrdinazione": 3,
          "orarioOrdinazione": "2023-06-15T11:52:18",
          "idDipendente": 3,
          "idTavolo": 2,
          "preparazioni": [
            {
              "idPreparazionePiattoOrdinazione": 4,
              "piatto": {
                "idPiatto": 1,
                "nome": "TestPiatto",
                "costo": 10,
                "descrizione": "Descrizione",
                "posizionePiatto": 1,
                "categoria": "TestCategory",
                "idCategoria": 1,
                "allergeni": [
                  "Frutta a guscio"
                ],
                "attivo": true
              },
              "idOrdinazione": 3,
              "idDipendente": 2,
              "statoPreparazione": "Evaso",
              "quantita": 3,
              "nota": "Troppo buono"
            }
          ],
          "tavolo": {
            "idTavolo": 2,
            "numeroTavolo": 2,
            "numeroOspiti": 1
          }
        },
        {
          "idordinazione": 4,
          "orarioOrdinazione": "2023-06-13T11:52:18",
          "idDipendente": 3,
          "idTavolo": 4,
          "preparazioni": [
            {
              "idPreparazionePiattoOrdinazione": 4,
              "piatto": {
                "idPiatto": 1,
                "nome": "TestPiatto",
                "costo": 10,
                "descrizione": "Descrizione",
                "posizionePiatto": 1,
                "categoria": "TestCategory",
                "idCategoria": 1,
                "allergeni": [
                  "Frutta a guscio"
                ],
                "attivo": true
              },
              "idOrdinazione": 3,
              "idDipendente": 2,
              "statoPreparazione": "Evaso",
              "quantita": 3,
              "nota": "Troppo buono"
            }
          ],
          "tavolo": {
            "idTavolo": 2,
            "numeroTavolo": 2,
            "numeroOspiti": 1
          }
        }
      ];
      const output = 2;
      expect(calcolaOrdiniDipendente(ordinazioni, dataInizio, dataFine)).toEqual(output);
    });
  });

```

# 9 Valutazione dell'usabilità sul campo

La valutazione dell'usabilità sul campo è un metodo per misurare l'efficacia, l'efficienza e la soddisfazione dell'utente nell'utilizzo di un prodotto o servizio nel contesto reale. A differenza delle valutazioni a priori, la valutazione sul campo si basa sulla raccolta di dati direttamente dall'uso quotidiano del prodotto.

Coinvolge utenti rappresentativi che forniscono feedback sulle loro esperienze. I risultati aiutano noi sviluppatori a identificare i problemi di usabilità e migliorare il prodotto per soddisfare le esigenze degli utenti. In questa fase, abbiamo deciso di utilizzare due metodologie per la valutazione dell'usabilità: **rilascio del prodotto in beta testing e analisi dei file di log**.

## 9.1 Usability Test

Gli usability test sono metodologie di valutazione dell'usabilità che consentono di misurare la facilità d'uso e l'esperienza dell'utente durante l'interazione con un prodotto o sistema. Durante questi test, gli utenti selezionati completano compiti specifici mentre vengono osservati dai ricercatori. I risultati aiutano a identificare i problemi di usabilità e a fornire indicazioni per migliorare l'interfaccia e l'esperienza utente complessiva. I valutatori scelti sono 5:

Conoscenza tecnologica	Alta	Ezio G.	Eva C., Marco C.
	Bassa	Michele M.	Enzo I.
	Bassa	Alta	
Conoscenza del dominio			

È stato chiesto di utilizzare l'intero applicativo, prendendo però in considerazione 4 compiti principali: prendere un'ordinazione, evadere un'ordinazione, visualizzare le statistiche degli addetti alla sala e ordinare gli elementi in un menu.

### 1. Prendere un'ordinazione

Tutti i valutatori sono riusciti a compiere quest'azione con successo e senza particolari problemi.

### 2. Evadere un'ordinazione

Tutti i valutatori sono riusciti a compiere quest'azione con successo e senza particolari problemi.

### **3. Visualizzare le statistiche degli addetti alla sala**

Michele M. ha avuto difficoltà a cambiare le date, gli è stato successivamente spiegato che bisognava cliccare sull'icona del calendario oppure inserire manualmente una data nella casella di testo. Tutti gli altri valutatori sono riusciti a compiere l'azione con successo.

### **4. Ordinare gli elementi in un menu**

Michele M. ed Enzo I. hanno avuto difficoltà a compiere quest'azione, in quanto non conoscevano il simbolo del "Drag&Drop". In questo caso, è stato deciso di aggiungere una spiegazione testuale nella pagina dell'organizzazione menu.

Tutti gli altri valutatori sono riusciti a compiere l'azione con successo.

Per il calcolo del tasso di successo, quindi, ci siamo rifatti alla seguente tabella:

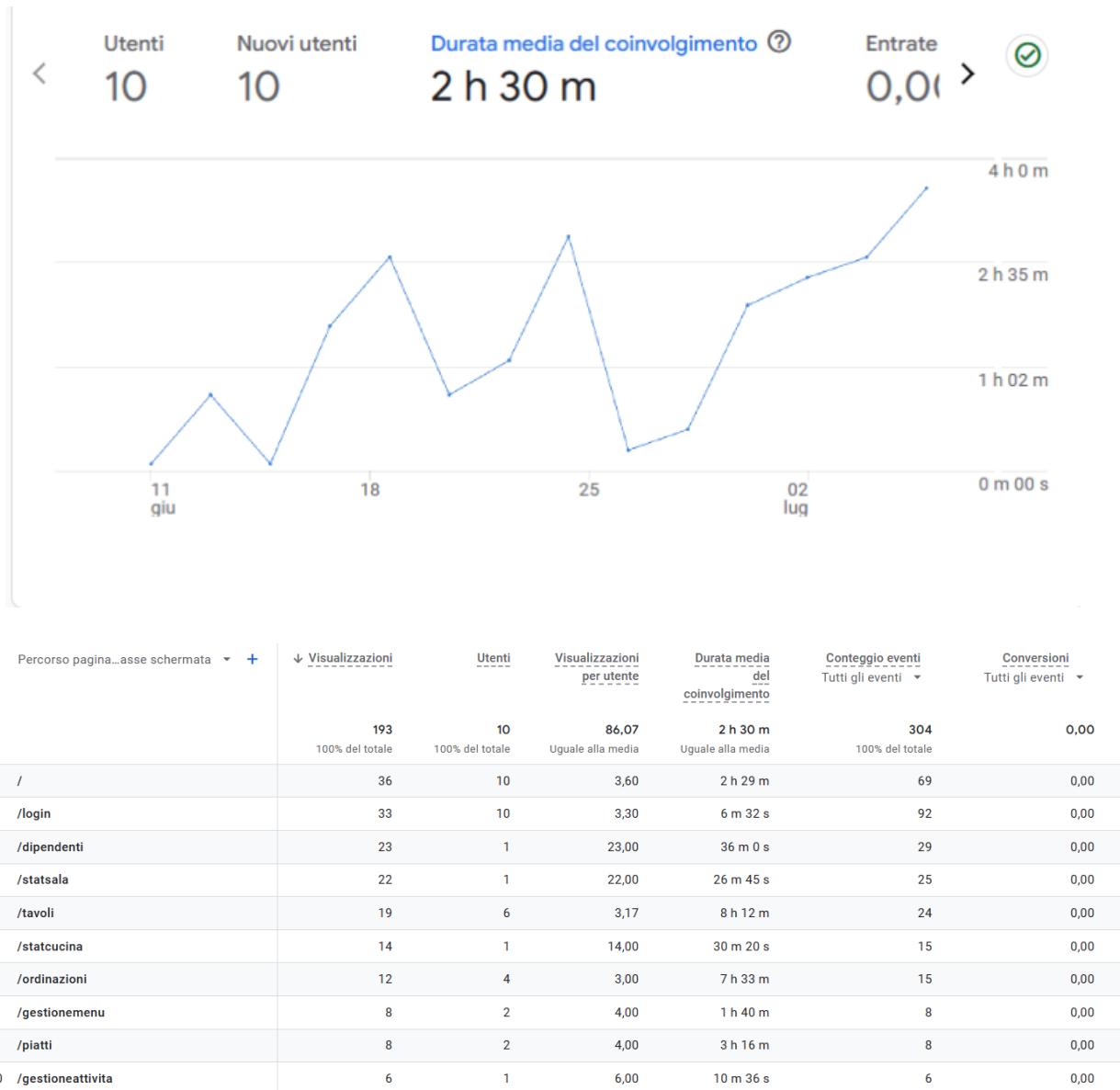
Valutatore	Compito 1	Compito 2	Compito 3	Compito 4
Ezio G.	S	S	S	S
Eva C.	S	S	S	S
Marco C.	S	S	S	S
Michele M.	S	S	P	P
Ezio I.	S	S	S	P

dove: Successo (S) vale 1 punto e Successo Parziale (P) vale 0.5.

Il tasso di successo è:  $(17 + (3 * 0.5)) / 20 = 92\%$ .

## 9.2 Analisi dei file di log

Per quanto riguarda l'analisi dei file di log, è stato deciso di utilizzare la piattaforma **Firebase**, in particolare lo strumento **Analytics**. Quest'ultimo è uno strumento di analisi web fornito da Google che consente di raccogliere, monitorare e analizzare i dati relativi al traffico e al comportamento degli utenti dell'applicativo. Offre una vasta gamma di funzionalità per comprendere come gli utenti interagiscono con il prodotto software.



Nome evento	+	↓ Conteggio eventi	Totale utenti	Conteggio eventi per utente	Entrate totali
		304 100% del totale	10 100% del totale	31,87 Uguale alla media	0,00 \$
1 <a href="#">page_view</a>		193	10	19,30	0,00 \$
2 <a href="#">scroll</a>		33	10	3,30	0,00 \$
3 <a href="#">form_start</a>		27	10	2,70	0,00 \$
4 <a href="#">user_engagement</a>		22	6	3,67	0,00 \$
5 <a href="#">session_start</a>		19	10	1,90	0,00 \$
6 <a href="#">first_visit</a>		10	10	1,00	0,00 \$

