Corna Giorgio 1074241

Rossi Diego 1073945

Progetto di ingegneria del software

FOOD APP

Obiettivo

- Il progetto riguarda lo sviluppo di un'applicazione rivolta a ristoranti e bar.
- In particolare, l'app fornisce la gestione delle comande prese dai camerieri, sia che lavorino appunto in un ristorante oppure in un bar.
- Inoltre, l'app fornisce un'interfaccia anche per la cucina, la quale può vedere gli ordini presi.

Difficoltà incontrate

Le difficoltà incontrate sono state principalmente tre:

- Implementazione del database
- Compatibilità di Android Studio tra Windows e Mac
- Test dei metodi del Database.

Paradigma di programmazione/modellazione utilizzato e tools

- Ambiente di sviluppo → Android Studio
- Linguaggio di programmazione → Java
- Modellazione → StarUML
- Database → SSMS/Linguaggio SQL

Software configuration management

- Tool utilizzato: Github

E’ servito per organizzare il progetto, condividere il codice e documentazione tramite le pull requests e porre/risolvere problem tramite gli issues.

Software life cycle

- Il software life cycle che abbiamo utilizzato è stato eXtreme Programming
- Il modello che abbiamo utilizzato per organizzare le fasi da svolgere è stato il modello a cascata.
- Durante la fase di implementazione, abbiamo seguito un modello Moscow per decidere quali funzioni implementare prima.

Qualità

- La gestione della qualità è stata gestita seguendo alcune qualità definite nel modello ISO 9126, quali:
- 1)MUTEVOLEZZA
- 2)COMPORTAMENTO TEMPORALE
- 3)OPERABILITA’

Requisiti

- I requisiti sono stati distinti in funzionali e non funzionali.
- Entrambi suddivisi in:

- Must have
- Should have
- Could have
- Won’t have

3.1) REQUISITI FUNZIONALI

1. MUST HAVE: NUOVA PRENOTAZIONE, VISUALIZZA PRENOTAZIONE, NUOVO ORDINE, VISUALIZZA ORDINE
2. SHOULD HAVE:MODIFICA PRENOTAZIONE, MODIFICA ORDINE, CANCELLA PRENOTAZIONE, CANCELLA ORDINE
3. COULD HAVE: CHECK PRENOTAZIONE,CHECK ORDINE
4. WON'T HAVE: LISTA PIATTI CON ALLERGENI, LISTA TAVOLI, NOTE ALL'INTERNO DELL' ORIDNE

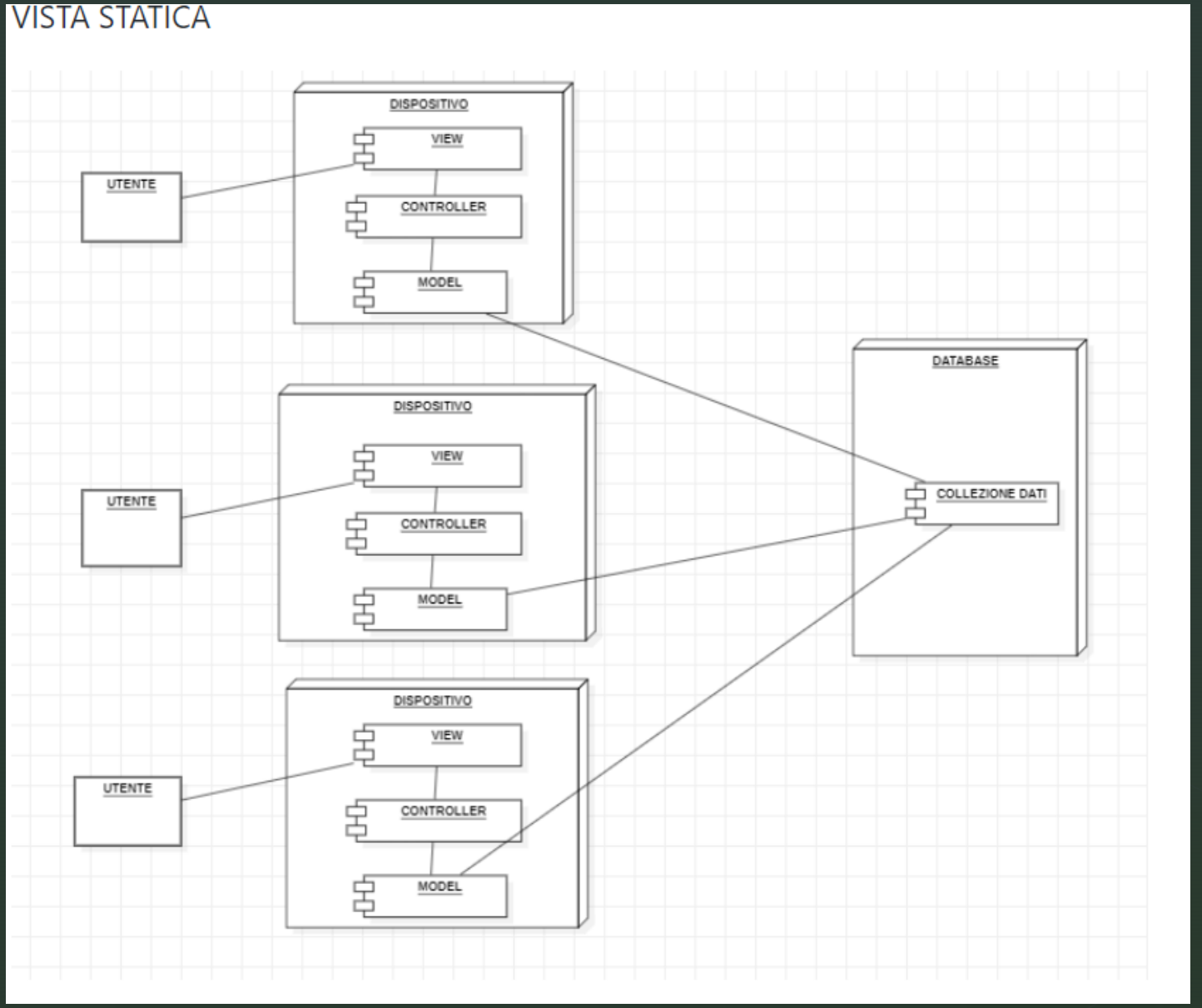
3.2) REQUISITI NON FUNZIONALI

1. MUST HAVE: interfaccia semplice ed intuitiva per non creare difficoltà nell' utilizzo
2. SHOULD HAVE: integrità dei dati nel database per non confondere ordini e prenotazioni.
3. COULD HAVE: tempi di risposta brevi (nell' ordine dei secondi).
4. WON'T HAVE: gestione della concorrenza nell' accesso al database.

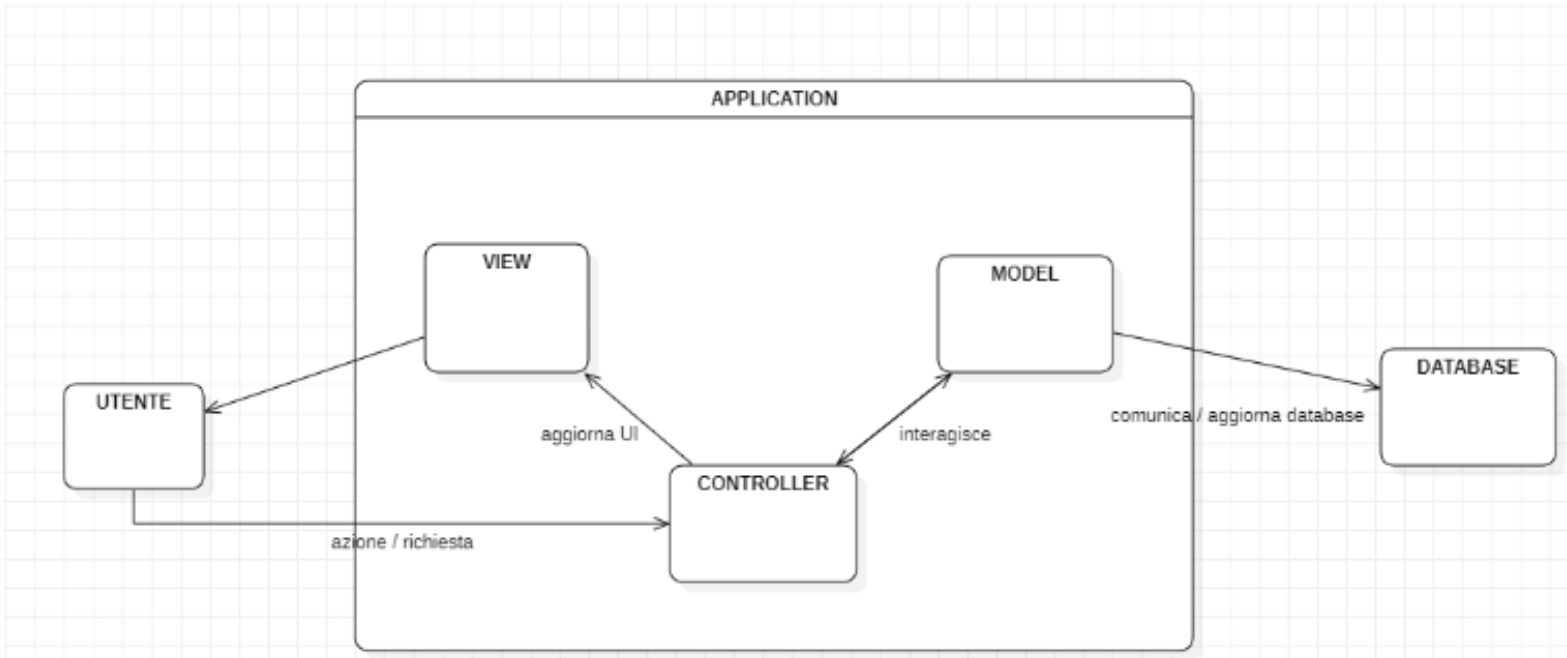
Architettura

- Lo stile architetturale che abbiamo deciso di seguire è stato il model view controller.
- Esso permette di mantenere un codice ordinato, facile da leggere e facilmente modificabile in futuro.

VISTA STATICA

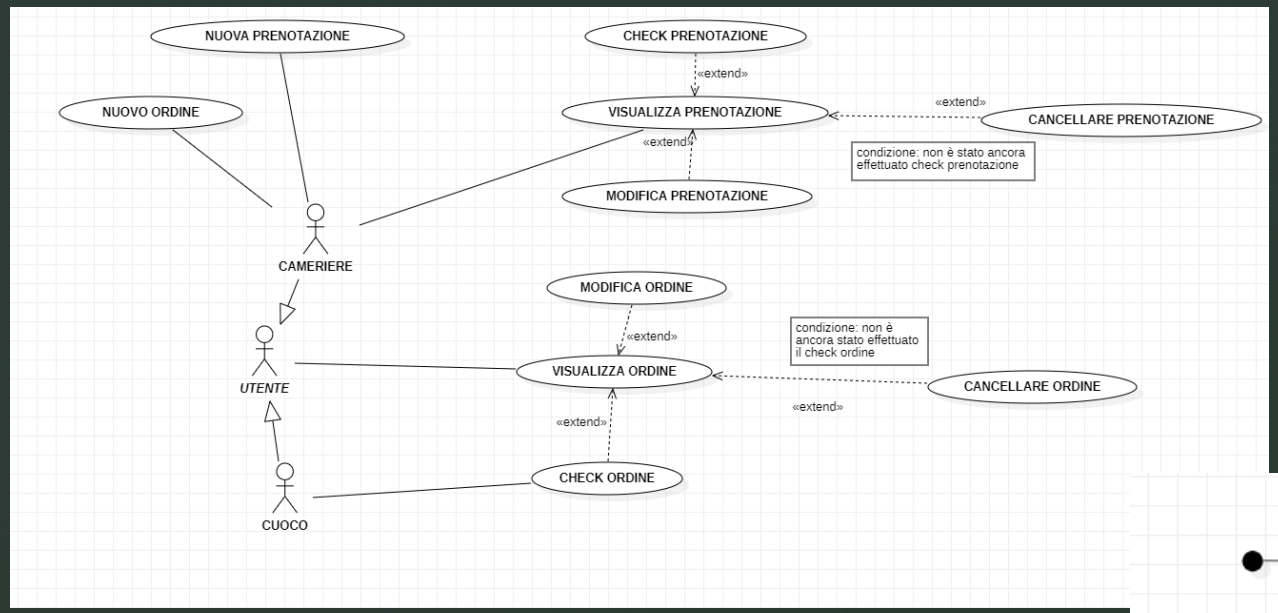


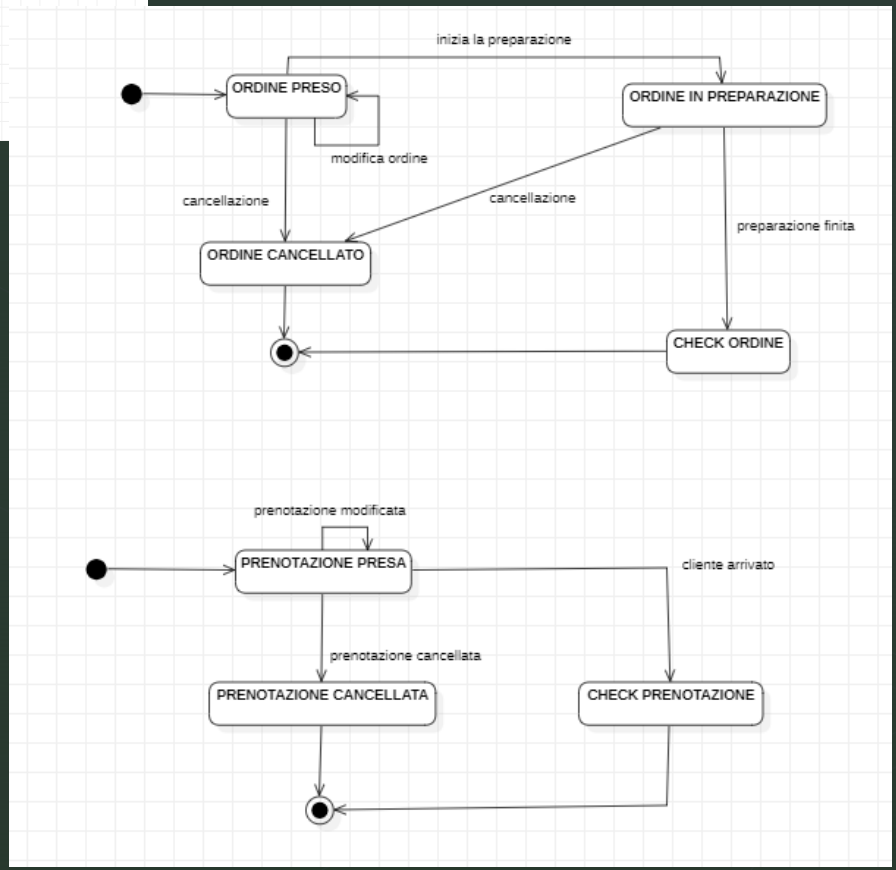
VISTA DINAMICA

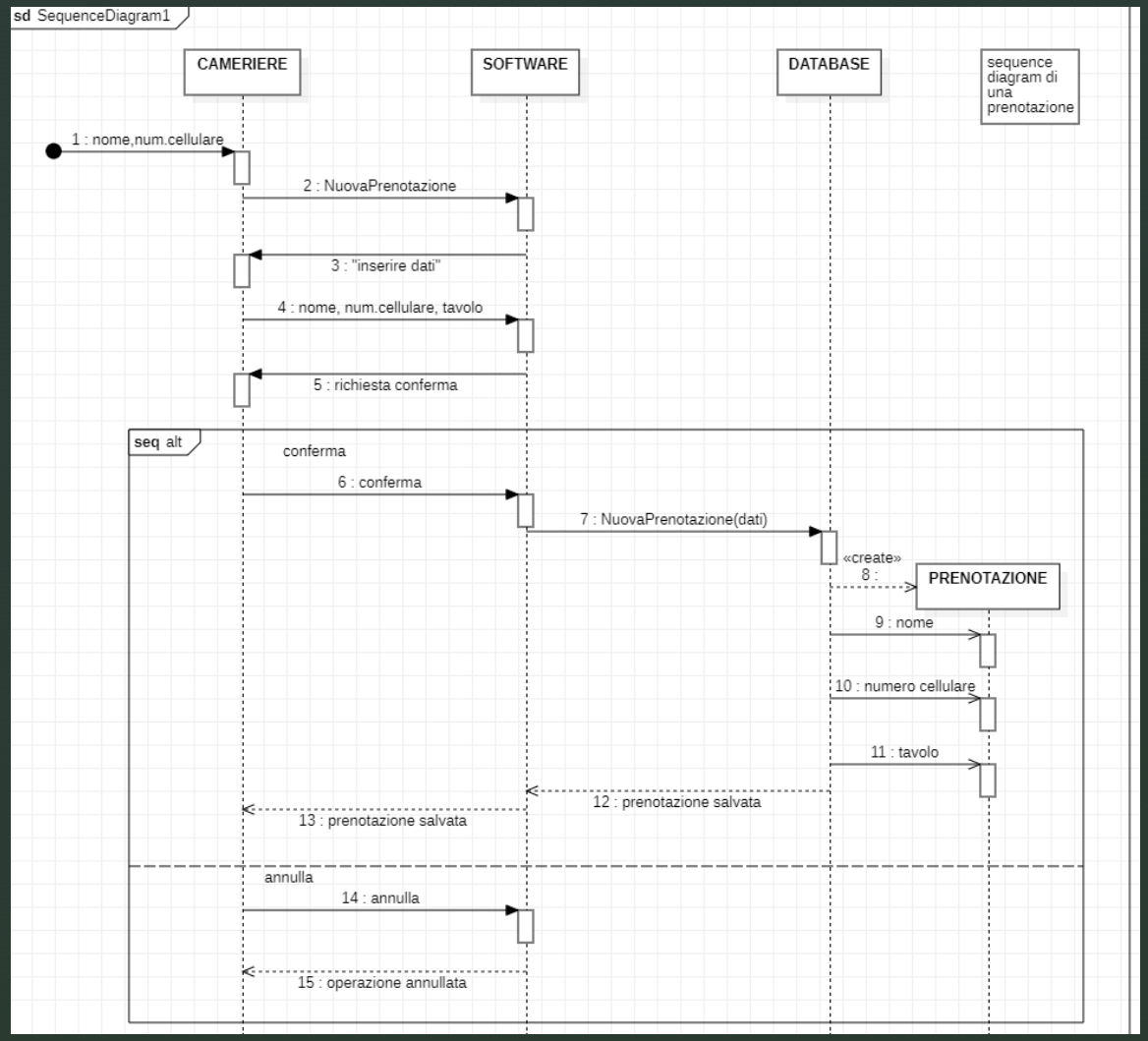


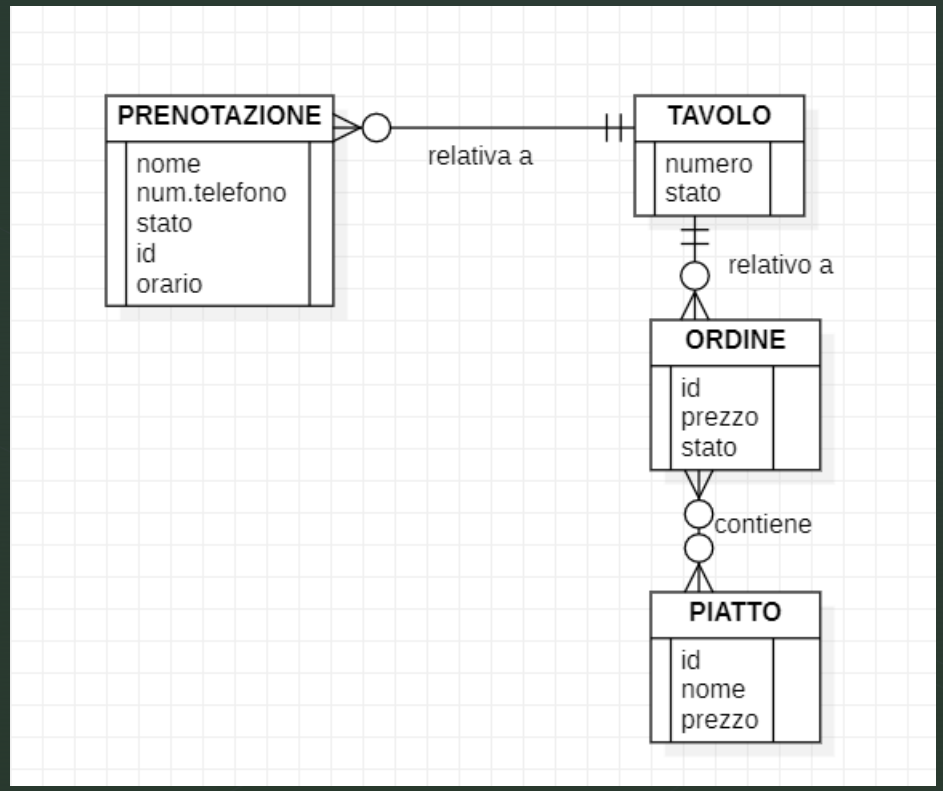
Modellazione

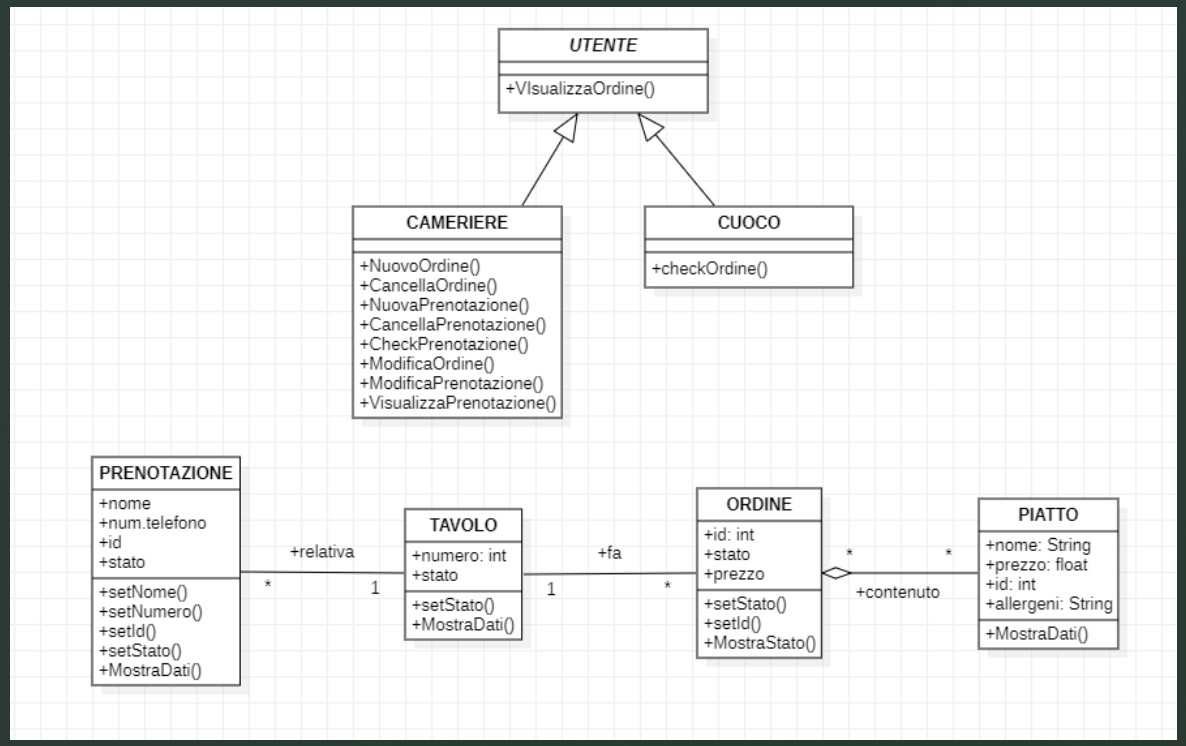
- Tramite starUML abbiamo modellato i seguenti diagrammi:
  - Casi d’uso
  - Attività
  - Classi
  - Sequenza
  - Stati

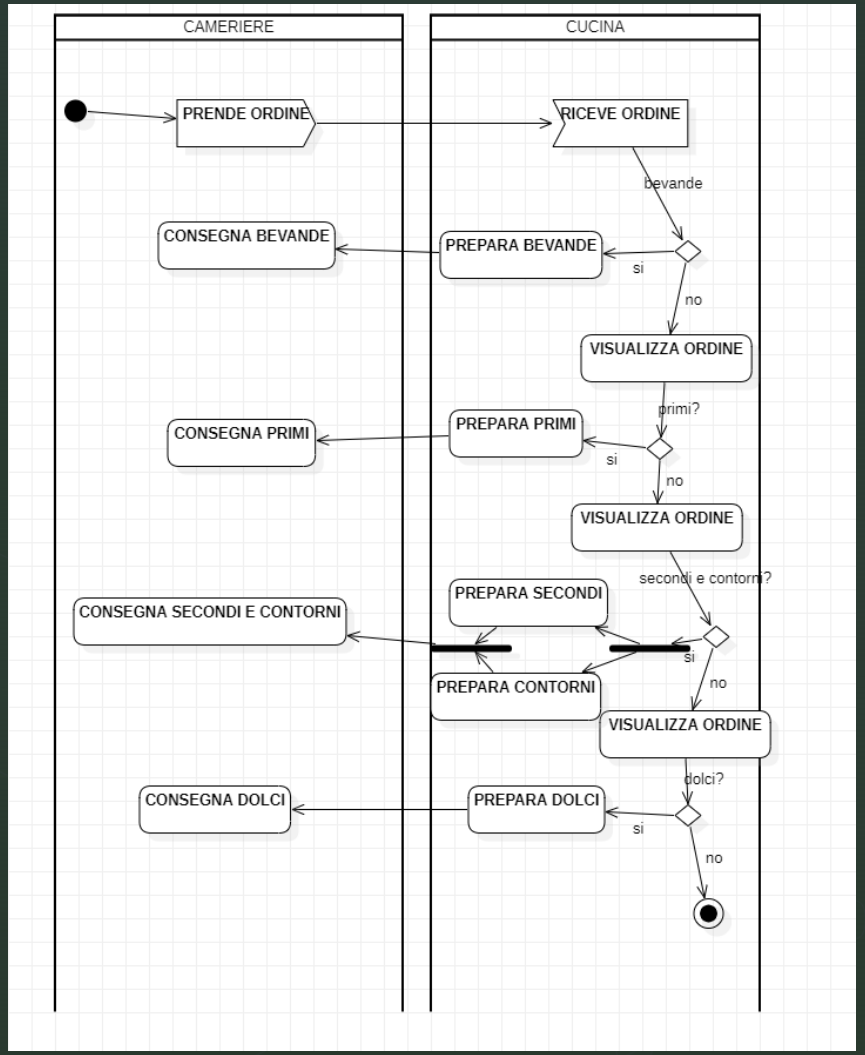

NUOVA PRENOTAZIONE
CHECK PRENOTAZIONE
«extend»
NUOVO ORDINE
VISUALIZZA PRENOTAZIONE
«extend»
CANCELLARE PRENOTAZIONE
condizione: non è stato ancora effettuato check prenotazione
«extend»
MODIFICA PRENOTAZIONE
CAMERIERE
MODIFICA ORDINE
«extend»
condizione: non è ancora stato effettuato il check ordine
UTENTE
VISUALIZZA ORDINE
CANCELLARE ORDINE
«extend»
«extend»
CHECK ORDINE
CUOCO


inizia la preparazione
ORDINE PRESO
ORDINE IN PREPARAZIONE
modifica ordine
cancellazione
cancellazione
preparazione finita
ORDINE CANCELLATO
CHECK ORDINE
prenotazione modificata
PRENOTAZIONE PRESA
cliente arrivato
prenotazione cancellata
PRENOTAZIONE CANCELLATA
CHECK PRENOTAZIONE


sd SequenceDiagram1
CAMERIERE
SOFTWARE
DATABASE
sequence diagram di una prenotazione
1 : nome,num.cellulare
2 : NuovaPrenotazione
3 : "inserire dati"
4 : nome, num.cellulare, tavolo
5 : richiesta conferma
seq alt
conferma
6 : conferma
7 : NuovaPrenotazione(dati)
«create»
8 :
PRENOTAZIONE
9 : nome
10 : numero cellulare
11 : tavolo
12 : prenotazione salvata
13 : prenotazione salvata
annulla
14 : annulla
15 : operazione annullata

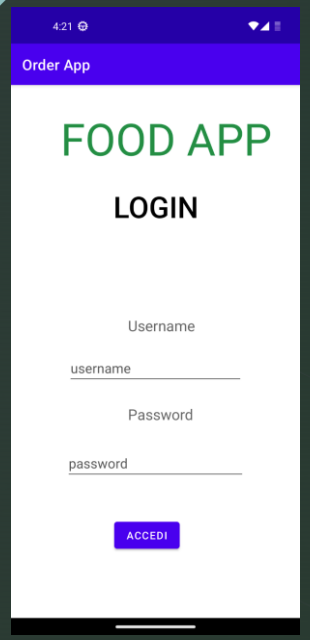

PRENOTAZIONE
nome
num.telefono
stato
id
orario
relativa a
TAVOLO
numero
stato
relativo a
ORDINE
id
prezzo
stato
contiene
PIATTO
id
nome
prezzo

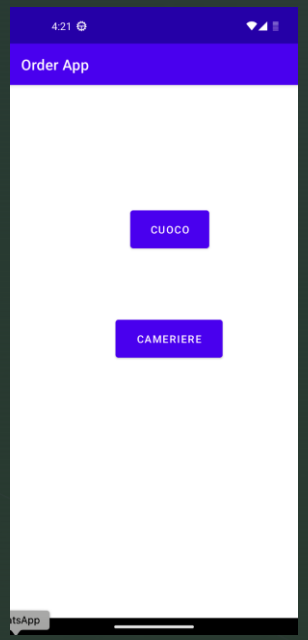

UTENTE
+VIsualizzaOrdine()
CAMERIERE
+NuovoOrdine()
+CancellaOrdine()
+NuovaPrenotazione()
+CancellaPrenotazione()
+CheckPrenotazione()
+ModificaOrdine()
+ModificaPrenotazione()
+VisualizzaPrenotazione()
CUOCO
+checkOrdine()
PRENOTAZIONE
+nome
+num.telefono
+id
+stato
+setNome()
+setNumero()
+setId()
+setStato()
+MostraDati()
+relativa
*
1
TAVOLO
+numero: int
+stato
+setStato()
+MostraDati()
+fa
1
*
ORDINE
+id: int
+stato
+prezzo
+setStato()
+setId()
+MostraStato()
*
*
+contenuto
PIATTO
+nome: String
+prezzo: float
+id: int
+allergeni: String
+MostraDati()

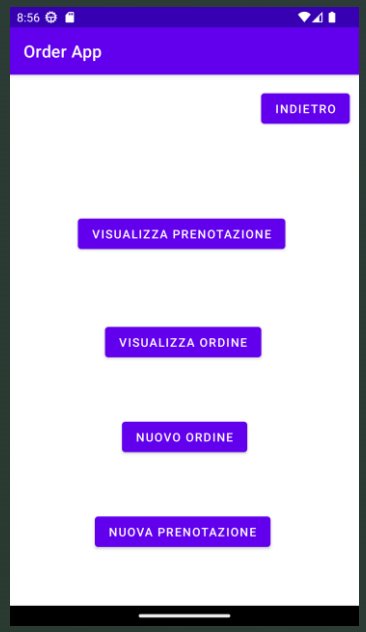

CAMERIERE
CUCINA
PRENDE ORDINE
RICEVE ORDINE
bevande
CONSEGNA BEVANDE
PREPARA BEVANDE
si
no
VISUALIZZA ORDINE
primi?
CONSEGNA PRIMI
PREPARA PRIMI
si
no
VISUALIZZA ORDINE
secondi e contorni?
CONSEGNA SECONDI E CONTORNI
PREPARA SECONDI
si
PREPARA CONTORNI
no
VISUALIZZA ORDINE
dolci?
CONSEGNA DOLCI
PREPARA DOLCI
si
no

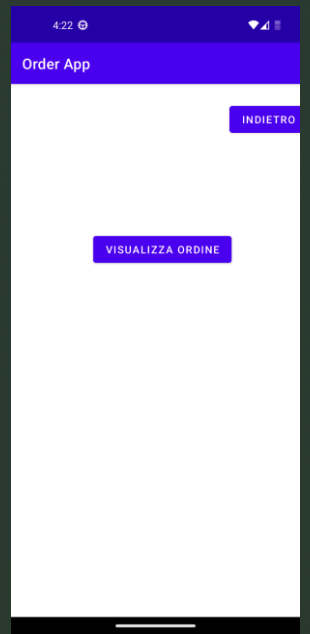
Implementazione

- Sono stati implementati principalmente i requisiti funzionali MUST HAVE descritti nel piano di progetto e nella documentazione dei requisiti.
- Ciò che non è stato implementato del tutto sono i vari requisiti SHOULD/COULD/WONT’ HAVE


4:21
Order App
FOOD APP
LOGIN
Username
username
Password
password
ACCEDI


4:21
Order App
CUOCO
CAMERIERE

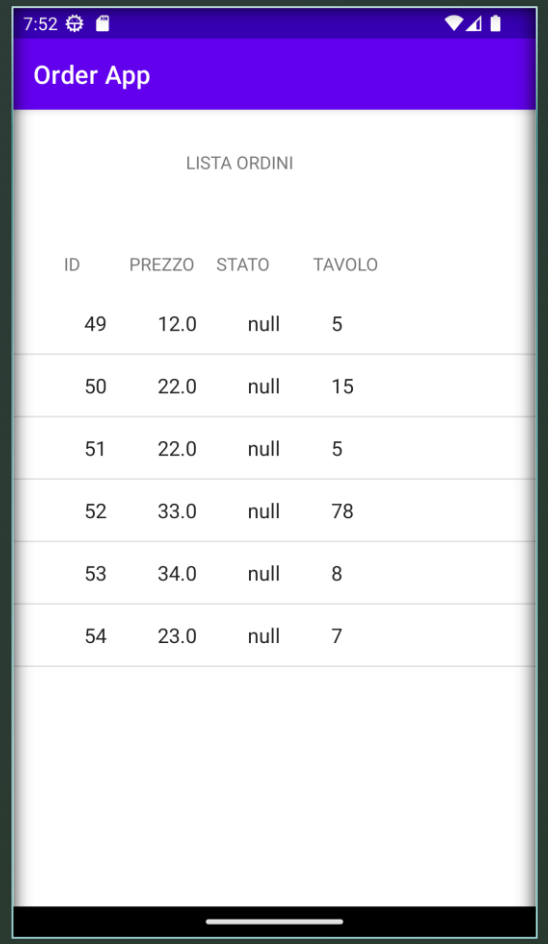

8:56
Order App
INDIETRO
VISUALIZZA PRENOTAZIONE
VISUALIZZA ORDINE
NUOVO ORDINE
NUOVA PRENOTAZIONE

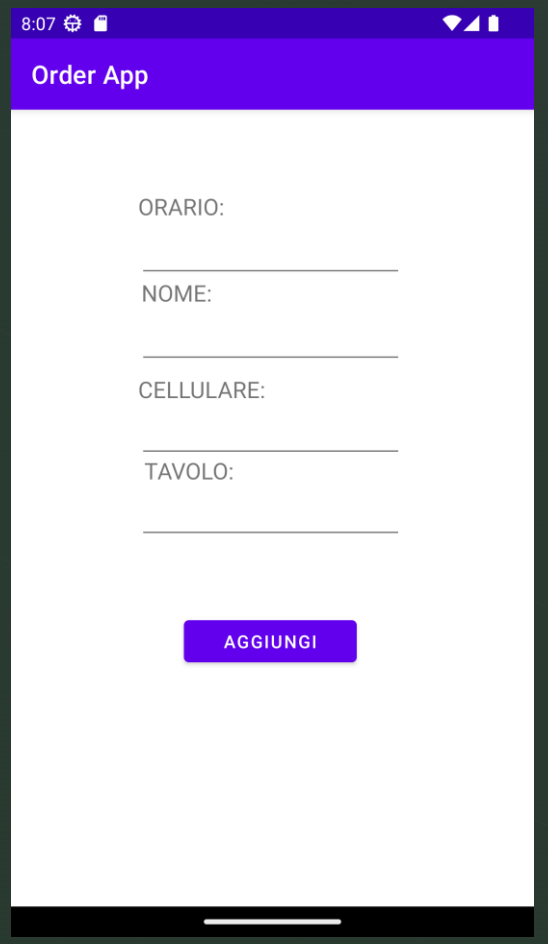

4:22
Order App
INDIETRO
VISUALIZZA ORDINE

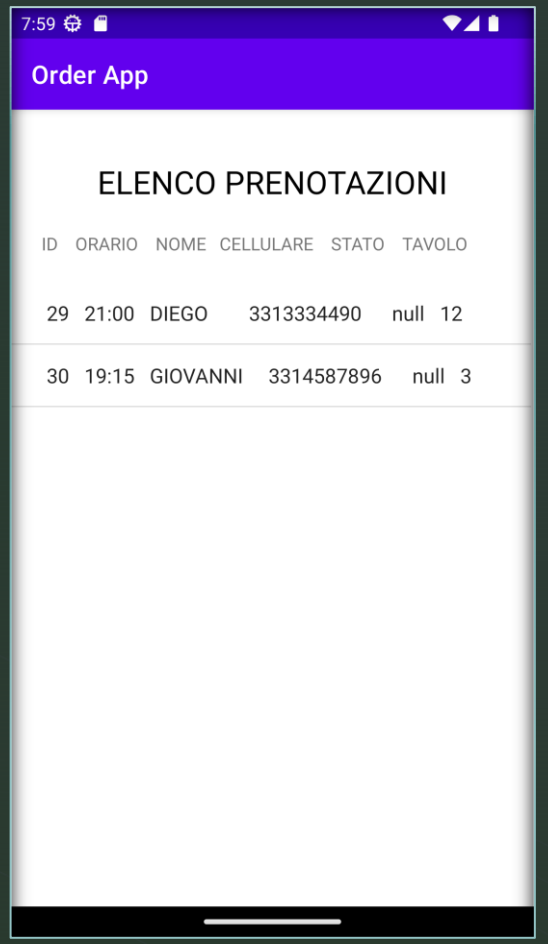

4:28
Order App
INDIETRO
TAVOLO
PASTA
RISO
CARNE
PESCE
PATATINE
INSALATA
ACQUA
VINO

DEMO

DEMO

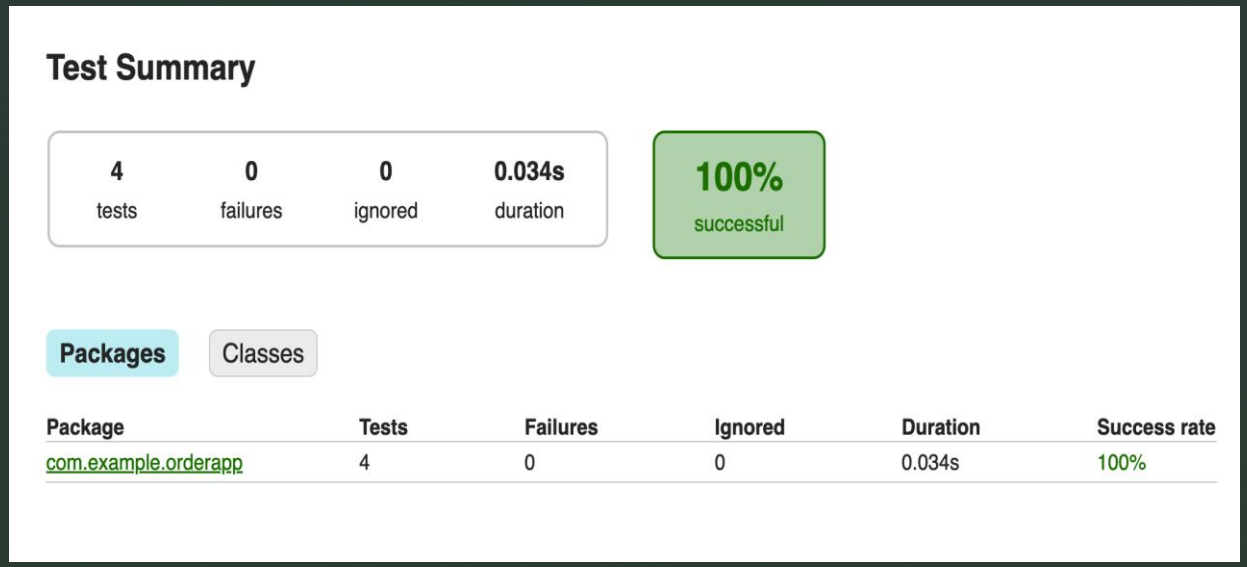

7:52
Order App
LISTA ORDINI
ID PREZZO STATO TAVOLO
49 12.0 null 5
50 22.0 null 15
51 22.0 null 5
52 33.0 null 78
53 34.0 null 8
54 23.0 null 7


8:07
Order App
ORARIO:
NOME:
CELLULARE:
TAVOLO:
AGGIUNGI


7:59
Order App
ELENCO PRENOTAZIONI
ID ORARIO NOME CELLULARE STATO TAVOLO
29 21:00 DIEGO 3313334490 null 12
30 19:15 GIOVANNI 3314587896 null 3

Testing Login

Per la fase di testing abbiamo utilizzato Junit.



Test Summary

4	0	0	0.034s
tests	failures	ignored	duration

100% successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.example.orderapp	4	0	0	0.034s	100%

Test per il Login

Manutenzione del software

- Dopo aver implementato ogni funzione abbiamo svolto una pratica di Refactoring per rendere il codice più leggibile e più concorde alle qualità seguite.