UNIVERSITÀ DEGLI STUDI DI BERGAMO

CORSO DI LAUREA MAGISTRALE DI INGEGNERIA INFORMATICA
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ServeEasy

Progetto del corso di
Progettazione, Algoritmi e Computabilità

Prof.ssa
Patrizia Scandurra

Candidati:
Giorgio Chirico
10xxxxx

Isaac Maffeis
1041473

Guyphard Ndombasi
10xxxxx

Salvatore Salamone
10xxxxx

Anno accademico 2023-2024

Indice

Elenco delle figure

Iterazione 0

1.1 Introduzione

Il sistema che si intende realizzare per il caso di studio è un software gestionale per ottimizzare la gestione delle comande di un ristorante, migliorando l'esperienza dei clienti, la produttività della cucina e l'efficacia della cassa. Il sistema si baserà sull'utilizzo di tablet, che permettono ai commensali di ordinare i piatti desiderati, inserendo eventuali note, visualizzando lo stato degli ordini e richiedere il conto in modo semplice e veloce. La cucina riceve le comande tramite una dashboard dedicata, che le ordina secondo un algoritmo di priorità basato su diversi parametri, come il tempo trascorso dall'ordinazione, la volontà del cliente, la durata di preparazione del piatto e altri fattori. La cucina può anche notificare il completamento di un ordine, che verrà visualizzato sul tablet del tavolo corrispondente. L'operatore di cassa sarà in grado di visualizzare il sommario degli ordini e stampare a schermo una ricevuta al cliente. L'amministratore del ristorante può personalizzare la configurazione delle sale e dei menu, registrare i tavoli e gli account, e visualizzare delle statistiche sulle ordinazioni effettuate. Il sistema offre anche delle funzionalità opzionali, come la possibilità di far arrivare i piatti tutti insieme al tavolo, di allegare note agli ordini in preparazione, chiedere il conto al tavolo. Il sistema si propone quindi di rendere più agile e soddisfacente il servizio di ristorazione, sfruttando le potenzialità della tecnologia e gli alti rendimenti di un algoritmo apposito.

1.2 Requisiti funzionali

I requisiti funzionali sono stati esplicitati mediante il diagramma UML dei casi d'uso in Figura 1.1, il quale è composto da 4 attori (Amministratore, Cucina, Cassiere e Cliente che tramite ereditarietà viene ridefinito in Cliente al tavolo oppure Cliente che effettua ordinazioni d'asporto) e 6 viste (vista amministratore, vista cucina, vista cassiere, vista cliente, vista cliente al tavolo e sistema).

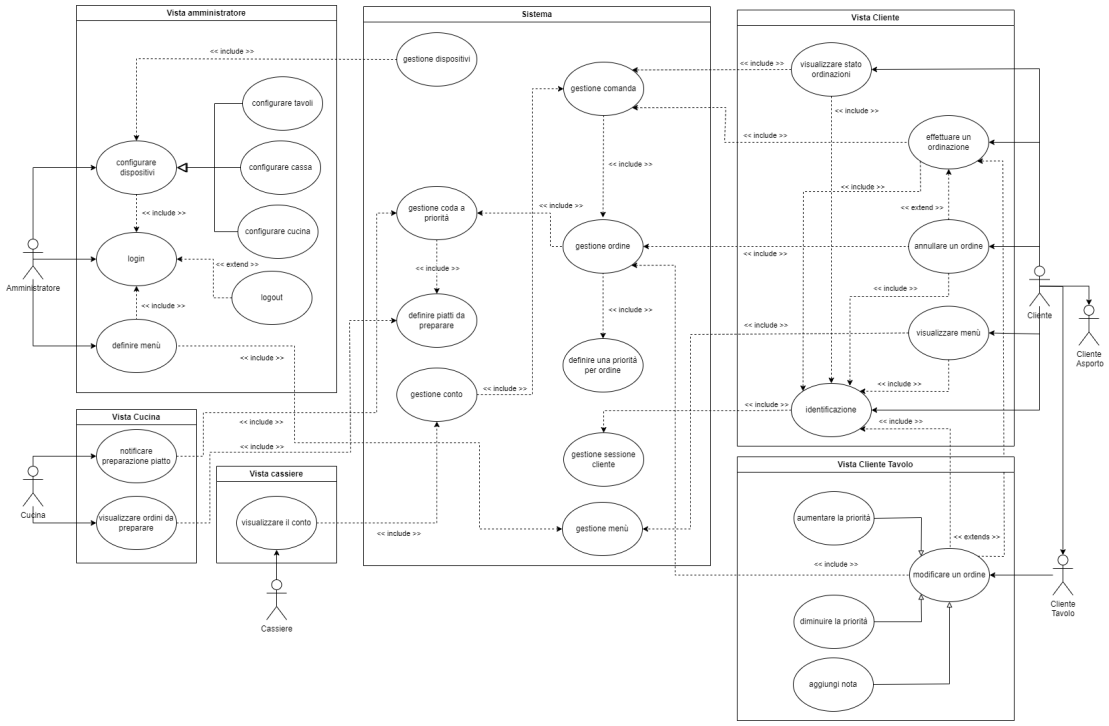

Figura 1.1: Diagramma dei casi d'uso

Per ottimizzare il processo di sviluppo, si è deciso di categorizzare le specifiche funzionali in tabelle con tre livelli di priorità: elevata, media e bassa. Nello specifico il primo livello è assegnato alla Tabella 1.1 a cui sono attribuiti i casi d'uso essenziali per il funzionamento dell'applicazione, i casi d'uso relativi alle funzionalità aggiuntive non critiche sono stati attribuiti alla Tabella 1.2 a priorità media, mentre il livello a bassa priorità che accoglie requisiti funzionali opzionali previsti per versioni successive alla Tabella 1.3 .

1.2.1 Priorità elevata

Codice	Titolo
UC1	Registrazione amministratore
UC2	Configurazione e gestione dispositivi (tavoli, cucina, cassa)
UC3	Login/logout amministratore
UC4	Gestione menù
UC5	Algoritmo gestione coda ordini
UC6	Cucina visualizza ordini da preparare
UC7	Cucina notifica preparazione piatto
UC8	Identificazione cliente
UC9	Cliente visualizza menu
UC10	Cliente effettua ordinazione
UC11	Gestione sessione cliente
UC12	Cassiere visualizza il conto

Tabella 1.1: Casi d'uso ad elevata priorità

1.2.2 Priorità media

Codice	Titolo
UC13	Modifica ordine al tavolo
UC14	Annullare un ordine
UC15	Modifica priorità piatto (aumentare, diminuire)
UC16	Cliente visualizza stato delle ordinazioni

Tabella 1.2: Casi d'uso a media priorità

1.2.3 Priorità bassa

Codice	Titolo
UC17	Identificazione cliente asporto
UC18	Cliente effettua ordinazione asporto
UC19	Gestione prioritaria ordinazioni da asporto
UC20	Strategia greedy "a gruppi" (ingrediente principale in comune)

Tabella 1.3: Casi d'uso a bassa priorità

1.3 Requisiti non funzionali

Il progetto verrà sviluppato tenendo considerazione delle performance, integrabilità, modificabilità, testabilità e sicurezza dei componenti.

1.3.1 Performance

L'algoritmo di priorità impiegato dalla cucina per la selezione degli ordini deve fornire risultati in un tempo utile. Allo stesso tempo, gli utenti dell'applicativo web devono poter accedere e aggiornare le informazioni in un tempo accettabile.

1.3.2 Integrabilità

Ogni componente di sistema deve collaborare con gli altri componenti in modo da garantire le funzionalità previste dal sistema. Questa caratteristica è essenziale per garantire il corretto funzionamento e la coerenza dell'intero sistema.

1.3.3 Modificabilità

Il software deve facilitare l'aggiunta di nuovi componenti e funzionalità.

1.3.4 Testabilità

Ogni componente deve poter permettere la progettazione, implementazione ed esecuzione di test efficaci, in modo da garantire una massima copertura di requisiti e funzionalità.

1.3.5 Sicurezza

Il sistema deve integrare meccanismi di autenticazione ed autorizzazione degli attori, in modo da garantire la gestione delle identità, oltre alla protezione dei dati e delle API da accessi non autorizzati. Risulta dunque necessaria una distinzione dei ruoli con cui gli attori accedono al sistema.

1.4 Topologia

Per il progetto è stata adottata una topologia three-tier al fine di separare in tre livelli distinti la presentazione dei dati, la gestione dell'applicativo e la mappatura dei dati sui dispositivi di archiviazione. Come si può vedere dalla Figura 1.2 il servizio è esposto tramite un web server, al quale i dispositivi clienti accedono, tramite richiesta HTTP/REST, per mezzo di una API unificata, con funzionalità di gateway. Il web-server usufruirà di database relazionali per lo storage (Data Layer), mentre sarà supportato da un database in-memory H2 (Application Layer) per avvantaggiarsi di una ridondanza dati, allo scopo di aumentare le performance lato client.

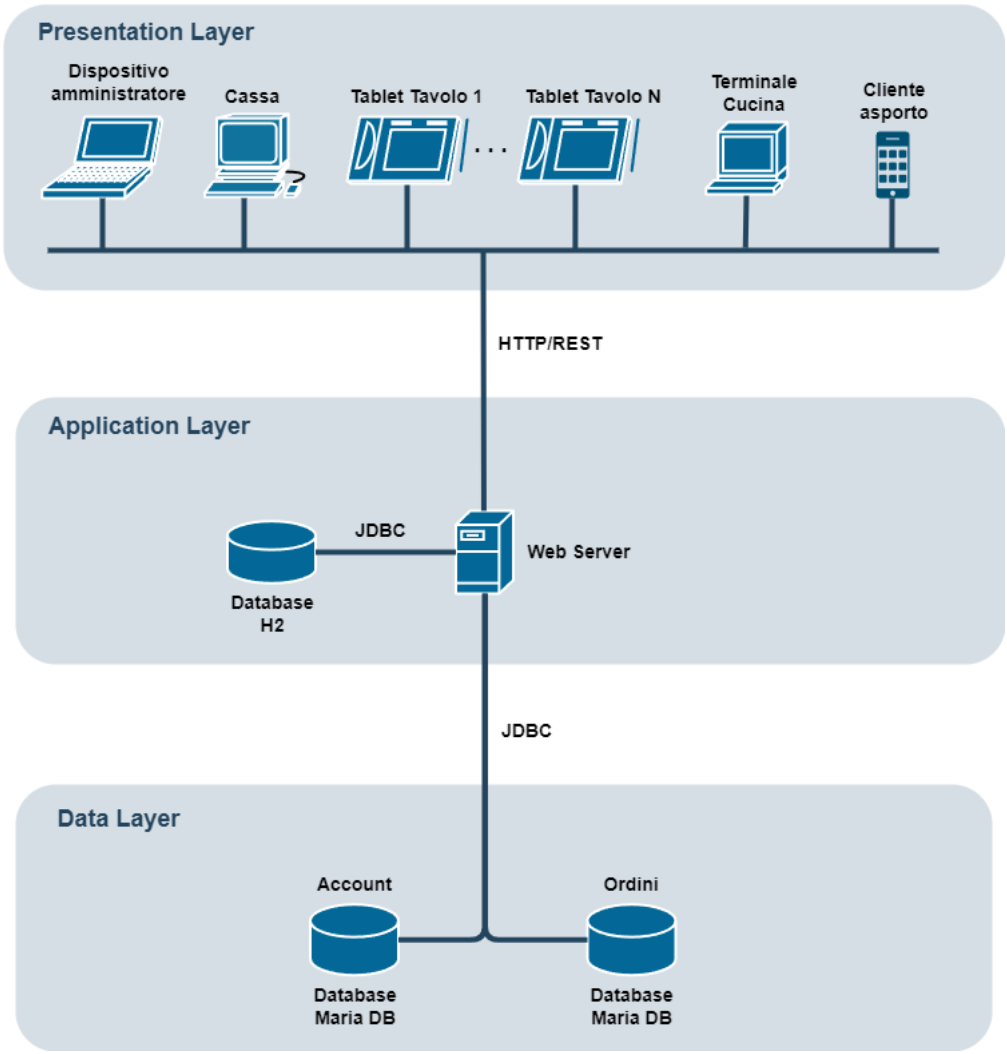

Figura 1.2: Topologia del sistema

1.5 Toolchain

Di seguito è presentata la toolchain utilizzata per lo sviluppo del progetto software

1.5.1 Modellazione

- draw.io: casi d'uso e topologia;

1.5.2 Stack applicativo

- Angular.js: front-end;
- Java Spring Boot 3.x.x: back-end;
- MariaDB: database per l'archiviazione;
- H2: database in-memory per rendere più efficiente l'estrazione dei dati;

1.5.3 Deployment

- Docker: piattaforma per container virtuali;
- Docker Compose: gestione app multi-container;

1.5.4 Gestore repository

- Git: Controllo versione per codice sorgente;
- GitHub: Piattaforma hosting e collaborativa per progetti Git;

1.5.5 Continuous Integration

- Maven: gestore di progetti e dipendenze Java;
- GitHub Action: piattaforma di automazione per repository GitHub;
- Jenkins: strumento di automazione per sviluppatori;

1.5.6 Analisi statica

- CodeMR su JetBrains: visualizzazione di alto livello di metriche qualitative del codice;

1.5.7 Analisi dinamica

- Postman: strumento per testare API e servizi;
- Garfana: analisi delle performance della rete di microservizi;
- JUNIT: framework per test unitari Java;

1.5.8 Documentazione e organizzazione del team

- Google Drive: servizio cloud per archiviazione;
- Documenti condivisi di Google: per elaborare la documentazione in modo condiviso;
- LaTeX: generazione documentazione;
- Microsoft Teams: per organizzazione e meeting;

1.5.9 Modello di sviluppo

Il modello adottato segue la filosofia AGILE, con enfasi sui seguenti aspetti-chiave:

- pair programming, per favorire creatività e controllo del lavoro prodotto;
- orientamento al risultato, con enfasi maggiore sulla generazione di codice funzionante e componenti completi prima della relativa documentazione;
- rapidità di risposta ai cambiamenti;
- collaborazione attiva col cliente, al fine di incontrare le sue necessità, garantire trasparenza e fornire feedback tempestivo sul lavoro di progetto;
- Proattività nell'identificazione e mitigazione dei rischi.