Università degli Studi di Verona

DIPARTIMENTO DI INFORMATICA
Corso di Laurea in Informatica

Tesi di laurea

Architetture per la gestione di ordini in locali di ristorazione

Studente:
VACCARI NICOLAS
Matricola 436988

Anno Accademico 2022-2023

Indice

1 Introduzione **5**
1.1 Prefazione 5
1.2 Panoramica sul mercato della ristorazione 5
1.3 Tipologie di ristorazione 6
1.4 Il peso della ristorazione collettiva 7

2 Caso d'uso: Refettorio aziendale collettivo **9**
2.1 Requisiti di ente e gestore 9
2.2 Analisi dei requisiti 10
2.3 Analisi delle funzionalità 10
2.3.1 Gestione di una coda digitale 10
2.3.2 Utente che effettua una prenotazione 11
2.3.3 Utente che ordina in loco 12
2.3.4 Gestione del flusso dell'operatore 12
2.4 Analisi tecnica 13
2.4.1 Architettura 13
2.4.2 Esempio di flusso completo 15
2.4.3 Analisi criticità: Ridondanza del sistema centrale . . . 16
2.4.4 Analisi criticità: Determinare l'ordine dei dispositivi nella coda virtuale 17
2.4.5 Analisi criticità: ottenimento del numero in coda . . . 18
2.4.6 Analisi criticità: gestire l'invio dei messaggi 18
2.5 Conclusioni 18

3 Terminologie **19**

4 Dedica **21**

Capitolo 1

Introduzione

1.1 Prefazione

Questo documento ha come obbiettivi:

- dare al lettore una panoramica dello stato della ristorazione al momento della scrittura
- quali possono essere le esigenze degli enti e gestori che mettono a disposizione e operano un servizio di ristorazione aziendale collettivo attraverso un caso d'uso
- quali soluzioni digitali è possibile adottare al fine di soddisfare i requisiti, quali problemi sussistono nella realizzazione di tali servizi, con un occhio di riguardo per gli aspetti di usabilità sia verso gli operatori del servizio, sia verso i consumatori finali

Il documento non ha come obbiettivo quello di illustrare le procedure burocratiche, economiche e operative che riguardano il mondo della ristorazione o le aziende coinvolte.

1.2 Panoramica sul mercato della ristorazione

A dicembre 2022 negli archivi delle Camere di Commercio italiane risultavano attive 335.817 imprese appartenenti al codice di attività 56.0 con il quale vengono classificati i servizi di ristorazione, un numero che non stupisce sia per un fattore culturale, sia perché i pubblici esercizi di questo tipo sono una

realtà ampiamente diffusa in ogni regione e che non ha eguali con nessun'altra tipologia di servizio rivolta alle persone presente in Italia.
Nonostante negli ultimi 4 anni il settore abbia subito delle forti perdite dovute principalmente alla pandemia COVID-19, all'inflazione e alla diffusione dell'ideologia "Stay at home" con annesso food delivery, il 2022 ha dimostrato che il mercato è in fase di stabilizzazione e ripresa, con valore stimato di circa 82 miliardi, avvicinandosi così al valore del 2019 pari a 85.5 miliardi. [2]
Questo settore, prevalentemente composto da manodopera umana e colmo di legislazioni da rispettare, ha da sempre lasciato poco spazio all'introduzione della tecnologia, sia per una questione pratica di sostituzione del lavoratore, sia per una questione di convenienza economica.
Fortunatamente per il settore IT, grazie al cambiamento dello stile di vita della persona media influenzato dai fattori citati prima, ora i ristoratori (specialmente i ristoratori collettivi) sono sempre più alla ricerca di nuove soluzioni per attrarre nuovi clienti e generare più vendite utilizzando gli stessi locali, aprendo così nuove possibilità di sviluppo e di conseguenza nuove sfide tecnologiche da risolvere.

1.3 Tipologie di ristorazione

Il mondo della ristorazione si divide principalmente in due categorie:

- **Ristorazione commerciale**: si rivolge direttamente al consumatore finale, con una clientela di tipo occasionale (in quanto sono i clienti che decidono di recarsi in quella specifica ubicazione) e nella maggior parte dei casi propone alimenti che sono preparati e consumati nello stesso luogo. In confronto con la ristorazione collettiva, il costo medio dei pasti è maggiore, ed il numero di individui che è possibile ospitare contemporaneamente è solitamente inferiore. In questa categoria rientrano:
  - **Ristorazione tradizionale**: la più comune e conosciuta, include trattorie, pizzerie, bistrot, ...
  - **Ristorazione agrituristica**: locali interni ad aziende agricole che servono pietanze con le loro materie prime
  - **Ristorazione alberghiera**: svolta all'interno di strutture alberghiere per completare l'offerta dei propri servizi
  - **Ristorazione veloce**: quella in più rapida espansione, include tutte le attività come fastfood, snackbar, tavole calde, ...

- **Ristorazione collettiva**: si occupa della preparazione e consegna di pasti su larga scala, con alimenti preparati in grandi cucine industriali dedicate e successivamente distribuiti a cucine più piccole collocate all'interno dei refettori che hanno il compito di "ripristinare" la pietanza e consegnarla al consumatore finale. I clienti di questo servizio sono tipicamente i fruitori dell'ambiente in cui si trova il locale (es: i dipendenti di una azienda), diventando quindi clienti abituali e la loro scelta deriva principalmente da una necessità di dover consumare almeno un pasto durante il loro periodo di permanenza nell'ubicazione. I costi sono contenuti, e molto spesso una parte del costo è carico dell'ente che mette a disposizione il servizio, ritagliandolo come un piccolo benefit. Il modus operandi di queste imprese è quello di proporre un menù fisso, accompagnato da uno o più operatori che gestiscono una linea self-service dove i clienti si accodano per acquistare la pietanza. In questa categoria rientrano:
  - **Ristorazione aziendale**: la si trova all'interno di imprese di medie o grandi dimensioni, ed è dedicata alla preparazione del pranzo o cena per i dipendenti
  - **Ristorazione scolastica**: dedicata alla preparazione di pasti per gli studenti o dipendenti di scuole e università
  - **Ristorazione socio-sanitaria**: dedicata alla preparazione di pasti all'interno di ospedali, cliniche e case di riposo
  - **Ristorazione assistenziale**: dedicata alla preparazione di pasti da distribuire al domicilio di persone non autosufficienti
  - **Ristorazione comunitaria**: presente all'interno di caserme, istituti religiosi e carceri penitenziari

1.4 Il peso della ristorazione collettiva

La ristorazione collettiva nasce con l'obbiettivo di colmare l'esigenza di tutti quegli individui che hanno la necessità di consumare almeno un pasto fuori casa, solitamente per un fattore di tempo a disposizione. Ad oggi possiamo considerare la ristorazione collettiva non più come "un plus" disponibile a pochi, ma una necessità di una società in continua evoluzione. Fornendo un esempio nel settore IT, possiamo notare come le 5 aziende FAANG offrano un servizio di ristorazione di alta qualità aperto 14 ore al giorno ai propri lavoratori, e di come questo servizio venga valorizzato da loro stessi come un punto di forza fondamentale per l'azienda stessa.

Uscendo dal mondo delle migliori aziende IT, troviamo comunque sempre più casi dove gli enti puntano ad offrire come benefit un servizio mensa di media-alta qualità, sia ai propri dipendenti che ai propri ospiti.
Ma quanto può incidere un servizio di ristorazione?
Un caso che può far riflettere è la realtà di Google, dove per convincere i propri dipendenti a tornare in ufficio e abbandonare il lavoro da casa, offre a chiunque accetti un servizio di creazione di una dieta bilanciata da parte di un team nutrizionale specializzato, e tutti i pasti a qualsiasi ora del giorno a carico dell'azienda. Al di fuori di questo elemento, possiamo prendere in considerazione un articolo della Cornell University [1], che ci fa notare i seguenti punti:

- I gruppi di individui che mangiano assieme, tendono a sviluppare rapporti umani migliori, che spesso si traduce in una produttività maggiore in gruppi di grandi dimensioni
- I singoli individui tendono a sentirsi più importanti e valorizzati quando il proprio ente mette a disposizione un servizio di qualità, aumentandone il morale e la motivazione
- Nel contesto aziendale, un dipendente può rivalutare la scelta di cambiare lavoro quando l'azienda offre un pasto garantito e di qualità, riducendo quindi le probabilità di turnover per l'azienda

Capitolo 2

Caso d'uso: Refettorio aziendale collettivo

2.1 Requisiti di ente e gestore

Requisiti dell'azienda ente:

- Possibilità di prenotare in anticipo il proprio pasto
- Possibilità di ordinare in loco il proprio pasto
- Garantire che gli ordini effettuati vengano serviti sulla base dell'ordine di ingresso in coda al refettorio da parte del singolo utente

Requisti dell'azienda gestore:

- Tenere traccia di tutti gli individui che mangiano in mensa, al fine di effettuare la rendicontazione
- Menù fisso con costo prestabilito e una combinazione fissa composta da un primo piatto, un secondo piatto e un contorno
- Regolamentare gli orari entro il quale è possibile prenotare il pasto
- Regolamentare gli orari in cui il servizio è attivo ed è possibile accedere alla mensa
- Due addetti alla linea self, uno per la gestione dei primi piatti e uno per la gestione dei secondi piatti e dei contorni

2.2 Analisi dei requisiti

Analizzando i requisiti di entrambe le parti possiamo stilare i seguenti punti:

- E' necessario introdurre un sistema centrale di backoffice per il salvataggio dei menù, il salvataggio degli ordini in loco e delle prenotazioni, l'elaborazione dei pagamenti, invio di email di conferma per le prenotazioni, regolazione degli orari di servizio, l'anagrafica degli utenti finali, l'anagrafica degli utenti operatori e l'anagrafica dei dispositivi hardware in uso all'interno del refettorio
- E' necessario realizzare un metodo di prenotazione digitale per l'utente accessibile all'esterno del refettorio; idealmente, una pagina web o una applicazione mobile
- E' necessario inserire un dispositivo hardware che consenta di effettuare ordini in loco
- E' necessario inserire un dispositivo hardware che consenta di identificare gli utenti che entrano in mensa con già una prenotazione effettuata
- E' necessario inserire due dispositivi hardware che consentano agli operatori della linea self di controllare l'avanzamento della coda virtuale degli utenti, e di indicare quali utenti sono stati serviti al fine di registrare l'erogazione del pasto all'interno del sistema centrale

2.3 Analisi delle funzionalità

All'interno di questo caso d'uso verranno tralasciate dall'analisi tutte le operazioni di "backoffice", quali:

- Costruzione di ricette, menù e combinazioni
- Gestione dei pagamenti, rendicontazioni, rimborsi
- Anagrafiche utenti, operatori e dispositivi

2.3.1 Gestione di una coda digitale

L'obbiettivo è quello di rendere i dispositivi fisici degli operatori "stupidi", e di delegare la logica che decreta l'avanzamento dei consumatori all'interno della coda al sistema centrale, resiliente nel cloud (andando incontro anche alle normative emanate da AGID per favorire l'utilizzo di piattaforme in

cloud) e non in un locale fisico dell'ubicazione. Questo modello presenta i seguenti pro e contro:

- (+) Consente di poter aggiustare al volo la posizione degli utenti, o di rimuovere degli utenti dal servizio in caso di anomalie con il loro ordine
- (+) Semplifica lo sviluppo, in quanto non è necessario implementare logiche di controllo per ogni tipologia di dispositivo fisico
- (+) Semplifica l'infrastruttura sistemistica per l'ubicazione, in quanto non essendo necessario un locale tecnico dedicato per la gestione di server del sistema centrale, si riducono le superfici d'attacco, si riducono i costi di startup e i costi di installazione dell'hardware e di mantenimento
- (+) Si riducono i costi per i dispositivi fisici, in quanto non sono necessari dispositivi molto potenti o con funzionalità di calcolo/comunicazione avanzate
- (-) Sono necessarie risorse maggiori per il sistema centrale (anche se il costo di esse può essere ammortizzato inserendole nel canone economico del servizio)

2.3.2 Utente che effettua una prenotazione

1. Il consumatore effettua la prenotazione tramite uno dei mezzi messi a disposizione, come app o pagina web
2. Il sistema verifica che la prenotazione rispetti tutti i parametri, e se la verifica va a buon fine, registra la prenotazione e invia una email di conferma all'utente che ha effettuato la prenotazione
3. Arrivato il giorno per cui ha effettuato la prenotazione, l'utente si reca nel refettorio, si identifica all'apposito tornello di ingresso, e riceve un codice numerico univoco che identifica il suo ordine e ne indica il suo posto nella coda. A questo punto l'utente può inserirsi in coda
4. Una volta giunto alla sezione dei primi piatti, l'operatore procede a consegnare il pasto, segnando sul suo dispositivo che l'utente è stato servito. Questo consente all'utente di avanzare alla sezione successiva, ovvero quella dei secondi piatti, dove a sua volta l'operatore dei secondi piatti si occuperà di servirlo.

5. Quando l'utente raggiunge l'ultima sezione, l'operatore che serve l'utente ha il compito di completare il flusso del consumatore, segnandolo come servizio erogato, e il consumatore sarà libero di abbandonare la coda per completare il pasto

2.3.3 Utente che ordina in loco

1. Il consumatore si reca fisicamente all'ubicazione per effettuare una consumazione

2. Attraverso il totem delle ordinazioni, il cliente effettua l'ordine e una volta verificato dal sistema centrale riceve il codice numerico che indica il suo posto nella coda

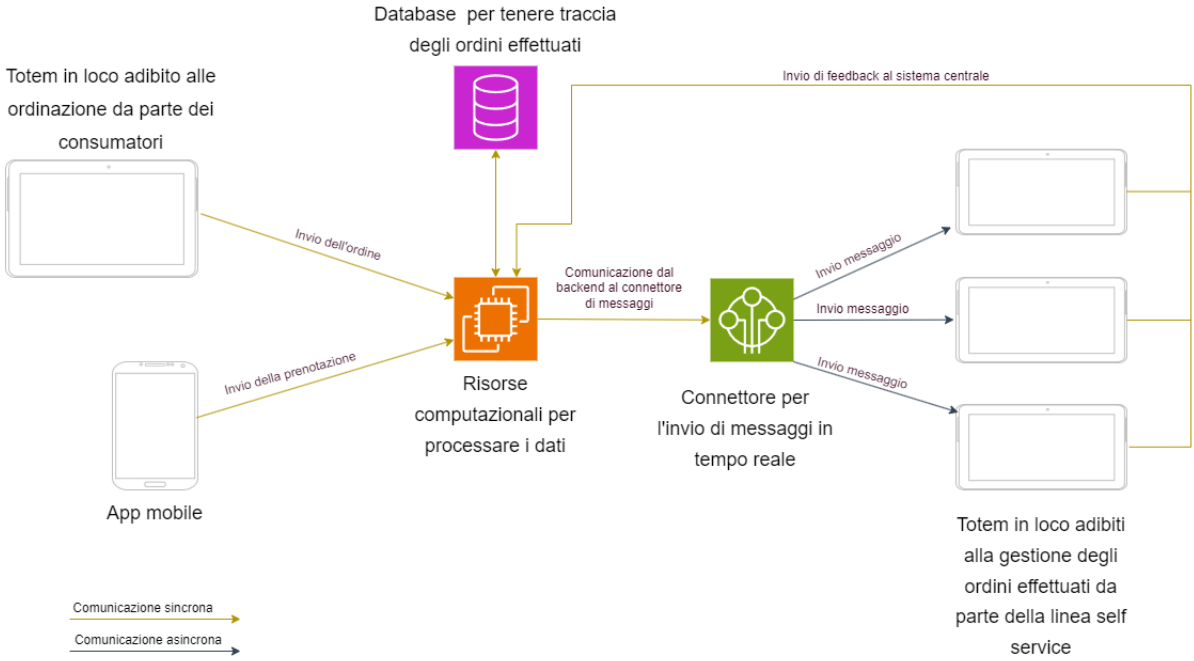
3. Arrivato il giorno per cui ha effettuato la prenotazione, l'utente si reca nel refettorio, si identifica all'apposito tornello di ingresso, e riceve un codice numerico univoco che identifica il suo ordine e ne indica il suo posto nella coda. A questo punto l'utente può inserirsi in coda

4. I punti 4 e 5 sono analoghi alla sezione precedente

2.3.4 Gestione del flusso dell'operatore

Gli operatori hanno il compito di coordinare l'avanzamento della coda nelle varie postazioni del servizio. Nel nostro caso, utilizzeremo una coda unica per i commensali, avendo tutti l'obbligo di consumare sia primi che secondi piatti. Quando l'utente entra nella coda, viene automaticamente segnata come destinazione la postazione dei primi piatti. Una volta che l'utente ha ricevuto il primo piatto dall'operatore, l'operatore conferma sul display del dispositivo che l'utente è stato servito, e al consumatore viene impostata come destinazione la postazione dei secondi piatti, che raggiungerà anche fisicamente. Giunto alla postazione dei secondi piatti, l'operatore serve l'utente e sul display del suo dispositivo segna l'utente come servito. Essendo quella dei secondi piatti l'ultima postazione della coda, l'utente viene rimosso dalla coda dal sistema centrale in quanto il suo flusso è stato completato. Il ruolo dell'operatore è quindi fondamentale per far corrispondere l'avanzamento nella realtà a quello nella coda virtuale all'interno del sistema.

Figura 2.1: Architettura generale per l'implementazione



2.4 Analisi tecnica

2.4.1 Architettura

L'architettura si compone di 4 tipologie di elementi:

- **Strumenti per la creazione di ordini**: nel nostro caso tramite totem e app mobile. Sia l'app che il totem possono essere realizzati con Flutter, un innovativo framework per lo sviluppo di app cross-piattaforma che funziona per iOS, Android, Linux, Windows e Web. La scelta di questo framework rappresenta un grande vantaggio competitivo, in quanto:
  - Garantisce correttezza e "feature-parity" tra le varie piattaforme, in quanto provenienti tutte da un unico codice sorgente
  - Consente di iniziare con un team di sviluppo limitato, in quanto è necessario selezionare solamente figure con skills omogenee
  - Al momento è il framework più popolare del mercato, avendo superato React Native, rappresentando quindi una scelta solida su cui investire anche a livello di management aziendale [4]

  - Avendo la possibilità di essere compilata anche per iOS e Android, consente di raggiungere un bacino di utenti molto più ampio rispetto ad una pagina web navigabile da pc fisso o portatile [3]

- **Risorse per la computazione dei dati**: nel nostro caso specifico abbiamo necessità di avere uno o più copie dell'applicativo che gestisce tutte le operazioni di logica delle code e di archiviazione dei dati. In particolare, possiamo strutturare le risorse in due componenti:

  - **Un database SQL**: In particolar modo PostgreSQL, facilmente scalabile, ACID-compliant ed open source. La scelta di un database SQL rispetto ad un database NoSQL è stata influenzata in particolar modo dall'incapacità dei database NoSQL di creare vincoli UNIQUE, cosa che a noi interessa al fine di evitare inserimenti doppi di individui in coda e ridurre i problemi di concorrenza
  - **Un applicativo**: Lo stack di questo applicativo varia dalle necessità dell'azienda, ma soprattutto dalle skills tecnologiche già presenti all'interno dell'azienda. Un linguaggio che sicuramente può soddisfare le necessità del nostro applicativo è GoLang:
    * Ogni applicativo in go è facilmente containerizzabile e deployabile su qualsiasi infrastruttura moderna come Kubernetes
    * Il linguaggio è semplice da prendere in mano e presenta poche ambiguità, perfetto sia per grandi aziende che per startup
    * L'ecosistema per la scrittura di applicativi web concorrenti è molto vasto, avendo una user-base che sviluppa principalmente applicativi web o applicativi di rete
    * La concorrenza è un cittadino di prima classe tramite "Goroutines" e "Channels"

    L'applicativo dovrà quindi esporre una serie di chiamate HTTP per permettere di effettuare tutte le operazioni necessarie, e dovrà poter comunicare con il connettore di invio messaggi e il database sql. L'applicativo è state-less, ovvero non tiene nulla nella memoria, e processa i dati singolarmente per ogni richiesta. Questa scelta è fatta per permettere l'aggiunta / distruzione di istanze on-the-fly tramite Kubernetes senza doversi preoccupare di sincronizzare le varie istanze tra di loro.

- **Connettore per l'invio di messaggi in tempo reale**: questo componente ha il compito di creare degli specifici canali d'ascolto chiamati

"topic", a cui i client possono connettersi ed ascoltare i messaggi inviati in questo canale e a loro volta scrivere all'interno del canale. Questa funzione è svolta tramite il protocollo di comunicazione MQTT utilizzando il modello Pub/Sub. Nel nostro caso, MQTT mette a disposizione dei canali al cui interno vengono inviati dei messaggi JSON codificati in base64. Il nome del canale ha la seguente struttura: **id-univoco-cliente/devices/id-univoco-device**. Ogni device fisico si mette in ascolto sul canale che corrisponde al suo id, così da ricevere solamente i messaggi a lui riservati. In caso di messaggi "globali" per tutti i device collegati ad una particolare coda, è possibile utilizzare il canale con la seguente struttura: **id-univoco-cliente/queues/id-univoco-coda**. Il protocollo MQTT ci garantisce l'invio dei messaggi al meglio di uno, non escludendo quindi la possibilità di ricevere messaggi duplicati. Sarà nostro compito capire come ignorare eventuali messaggi doppi. Con la nostra configurazione, i dispositivi fisici fungono solamente da subscribers, e l'unico publisher è il sistema centrale, che invierà messaggi al connettore tramite una chiamata HTTP dedicata che funge da servizio di ingest dei messaggi. Si è scelto di fare affidamento ad AWS IoT, un servizio cloud completamente gestito e scalabile fino a 100 milioni di messaggi al secondo e oltre 10 milioni di dispositivi connessi contemporaneamente.

- **Device per la gestione dei flussi**: analogo al caso dei totem, la scelta più conveniente rimane sempre Flutter.

2.4.2 Esempio di flusso completo

Procediamo ora ad illustrare come viene creato e completato un flusso di coda a partire da un utente che effettua un ordine in loco:

1. L'utente completa l'acquisto sul totem per gli ordini

2. Il totem invia una richiesta HTTP al sistema centrale per verificare la validità dell'ordine

3. Supponendo che tutto sia andato a buon fine, il sistema centrale procede attraverso la sezione critica di assegnare un numero univoco all'ordine, e una volta determinato lo restituisce al totem in risposta alla chiamata HTTP precedente. Infine il totem stampa il codice ricevuto.

4. Una volta determinato il codice, il sistema centrale procede a determinare quali dispositivi vanno avvisati dell'arrivo dell'utente, e successi-

vamente invia una richiesta HTTP al punto di ingest del componente MQTT, fornendo il messaggio e i canali a cui inviarlo.

5. Nel nostro caso, siccome ci troviamo alla creazione del flusso, questo messaggio verrà recapitato al dispositivo assegnato alla postazione dei primi piatti.

6. Una volta ricevuto, viene inserito nella coda all'interno del dispositivo che avanzerà man mano al comando dell'operatore.

7. Alla richiesta di avanzamento del nostro messaggio da parte dell'operatore, il dispositivo invia una chiamata HTTP al sistema centrale, chiedendo l'avanzamento del flusso al dispositivo successivo o la sua terminazione. Il sistema centrale computa il successivo display, e procede inviando un messaggio al display successivo sempre tramite il componente MQTT.

8. Il ciclo si ripete fino a quando, arrivati all'ultimo dispositivo fisico, il sistema determina che la coda per il flusso è terminata, e procede al suo completamento, terminando di inviare messaggi inerenti al flusso.

2.4.3 Analisi criticità: Ridondanza del sistema centrale

Come citato in precedenza, avendo delegato il compito al sistema centrale di gestire le logiche di ordine e avanzamento, è necessario che il sistema centrale sia sempre operativo e raggiungibile da qualsiasi parte del mondo. Per questo motivo, appoggiarsi a un provider cloud con servizi gestiti è sicuramente una scelta farevole. I provider cloud mascherano la complessità di gestire una infrastruttura ridondata e si occupano loro di garantire percentuali di uptime annuali molto alte, fino a 99.995%, ovvero meno di 30 minuti di disservizio l'anno. Non è però sufficiente scegliere una infrastruttura sicura, ma è necessario progettare anche l'applicativo per gestire gli errori imprevisti, gestire la scalabilità durante i picchi, e saper recuperare le informazioni in caso di errori. Uno strumento adatto a questo compito è Kubernetes, un framework nato per semplificare lo scaling di applicazioni in maniera uniforme, sfruttando la tecnologia dei container linux. Possiamo quindi utilizzare:

- **AWS EC2** per le risorse computazionali pure
- **AWS EKS** per la gestione delle risorse tramite Kubernetes
- **AWS RDS** per un database sql postgres replicato

- **AWS IOT** come connettore MQTT distribuito
- **AWS S3** per salvare tutti i backup dei dati ad intervalli regolari (es: ogni 6 ore)

2.4.4 Analisi criticità: Determinare l'ordine dei dispositivi nella coda virtuale

Uno dei primi problemi da risolvere è capire come strutturare una anagrafica per salvare l'ordine dei dispositivi all'interno di una coda prevenendo doppioni. Questo problema può essere risolto sfruttando una delle caratteristiche dei database SQL, ovvero il vincolo **UNIQUE**. La nostra tabella dati può essere strutturata nel seguente modo:

Nome colonna	Tipologia	Argomenti	Descrizione
id	INT(11) UNSIGNED	NOT NULL PRIMARY KEY	Chiave primaria della riga di associazione.
dispositivo_fisico_id	INT(11) UNSIGNED	NOT NULL	ID del dispositivo da associare alla coda.
coda_id	INT(11) UNSIGNED	NOT NULL	ID della coda a cui associare il dispositivo.
dispositivo_ordine	INT(2) UNSIGNED	NOT NULL	Ordine virtuale del dispositivo a partire da 1.

Questa struttura dati ci consente di associare qualsiasi dispositivo fisico a qualsiasi coda ed in qualsiasi ordine. Noi vogliamo associare un dispositivo solamente una volta ad una coda, e vogliamo assicurarci che per una coda non ci siano mai due dispositivi con lo stesso ordine. Per farlo, possiamo utilizzare due vincoli UNIQUE:

- **UNIQUE(dispositivo_fisico_id, coda_id)**: al fine di tenere attivo il principio che un dispositivo fisico può essere collegato solamente ad una coda contemporaneamente
- **UNIQUE(dispositivo_ordine, coda_id)**: al fine di tenere attivo il principio che non possono esistere più dispositivi nella stessa coda con lo stesso ordine

2.4.5 Analisi criticità: ottenimento del numero in coda

Il numero di coda è il modo visivo di comunicare agli utenti che all'interno del refettorio la loro posizione ed il loro turno. Deciderlo in un ambiente corrente non è banale, in quanto vi è la possibilità di creare dei numeri duplicati se non gestiti correttamente. Per aggirare questa criticità, ci affidiamo nuovamente alle funzionalità offerte dal database SQL, in questo caso una transazione atomica combinata ad un lock sulla tabella in scrittura e lettura. Lo pseudo codice dell'applicativo per la gestione di questa fase critica è il seguente:

1: $retries \leftarrow 5$
2: $nuovoNumero \leftarrow -1$
3: **while** $retries \geq 0$ **do**
4: acquisisci lock sulla tabella con timeout di 100ms
5: **if** lock acquisito **then**
6: leggi ultimo numero inserito
7: $nuovoNumero \leftarrow ultimoNumero + 1$
8: salva a database con una transazione atomica
9: rilascia il lock
10: break
11: **else**
12: $retries \leftarrow retries - 1$
13: **end if**
14: **end while**
15: **if** $nuovoNumero ==$ -1 **then**
16: throw errore inserimento fallito
17: **end if**

Questo algoritmo garantisce di non avere deadlock, in quanto una risorsa non viene mai tenuta per un tempo illimitato, e allo stesso tempo si assicura che durante la fase critica concessa dal lock il dato che stato leggendo è integro.

2.4.6 Analisi criticità: gestire l'invio dei messaggi

2.5 Conclusioni

Capitolo 3

Terminologie

- **FAANG**: Facebook, Apple, Amazon, Netflix, Google
- **AGID**: Agenzia per l'Italia Digitale
- **ACID**: Atomicity, Consistency, Isolation, e Durability

Capitolo 4

Dedica

Bibliografia

[1] Susan Kelley. *Groups that eat together perform better together*. Cornell University, 2015

[2] Fipe. *Ristorazione - Rapporto annuale 2023*. Confcommercio, 2023

[3] Statista. *Market share of desktop, mobile, tablet, and console devices in Italy from 2016 to 2022*. Statista, 2023

[4] Flatirons. *Popularity of Flutter vs. React Native in 2024*. Flatirons, 2024