System Design Document

AlcoList

Riferimento	
Versione	0.9
Data	
Destinatario	Prof. Gravino
Presentato da	Baldi Maria Rosaria Conte Melania Di Zenzo Carmine Federico Zaccardi Mario
Approvato da	

Sommario

Revision History

DATA	VERSIONE	DESCRIZIONE	AUTORI
30/11/2022	0.1	Prima stesura	BMR, CM, DCF, ZM
03/12/2022	0.2	Trade-Off, Controllo Flusso globale e condizioni limite	DCF, ZM
05/12/2022	0.3	Mapping Hardware/Software	BMR
07/12/2022	0.4	Obiettivi di Design, Controllo Accesso e Sicurezza, Architettura Sistema Proposto	BMR, CM, DCF, ZM
09/12/2022	0.5	Decomposizione in Sottosistemi, Servizi dei Sottosistemi	BMR, CM, DCF, ZM
11/12/2022	0.6	Dizionario dei Dati	DCF, ZM
17/12/2022	0.7	Prima Revisione	BMR, CM, DCF, ZM
12/01/2023	0.8	Modifiche trade-off post Revisione	BMR, DCF, ZM
13/01/2023	0.9	Modifica Class Diagram	BMR, CM

Team Members

NOME	ACRONIMO	INFORMAZIONI DI CONTATTO
Baldi Maria Rosaria	BMR	m.baldi24@studenti.unisa.it
Conte Melania	CM	m.conte49@studenti.unisa.it
Di Zenzo Carmine Federico	DCF	c.dizenzo2@studenti.unisa.it
Zaccardi Mario	ZM	m.zaccardi@studenti.unisa.it

1 Introduzione

1.1 Scopo del Sistema

Come meglio illustrato nel paragrafo 1.1 del documento "RAD", la realizzazione di AlcoList ha come obiettivo quello di essere di supporto ai proprietari di Bar nel compiere le loro operazioni di gestione di ordini e inventario. In particolare, con questo progetto, la gestione delle operazioni citate risulterà più rapida, efficiente e precisa.

1.2 Obiettivi di Design

Di seguito, vengono descritti gli obiettivi di design che devono essere rispettati nella realizzazione del Sistema. Tali obiettivi sono organizzati in cinque categorie:

1. **Criteri di performance**
   - **Tempo di risposta**
     - Il cambiamento di stato della comanda deve essere effettuato entro 2 secondi.
2. **Criteri di dependability**
   - **Robustezza**
     - Gli input non validi inseriti dall'utente devono essere segnalati con messaggi d'errore.
   - **Disponibilità**
     - Il Sistema deve essere disponibile H24 ad eccezione delle ore 9:00 alle 10:00 per riservare un eventuale orario per interventi di manutenzione del Sistema.
   - **Tolleranza ai guasti**
     - In caso guasti nell'architettura infrastrutturale fisica della nostra piattaforma, il sistema dovrà reindirizzare le richieste ad un'altra replica del sistema funzionante.
   - **Scalabilità**
     - Il sistema prevede di aumentare o diminuire le risorse in funzione delle richieste degli utenti.
   - **Sicurezza**
     - Il Sistema deve garantire il rispetto delle leggi sulla privacy, memorizzando soltanto le informazioni consentite dalle normative stesse.
   - **Autorizzazione**
     - Il Sistema si assicura che gli utenti finali possano accedere a determinate sezioni dell'applicazione in base al loro ruolo.

3. **Criteri di costo**
   - **Costi di sviluppo**
     - Il costo complessivo del progetto ammonta ad un massimo di 200 ore (max 50 ore per ogni membro del team).
4. **Criteri di manutenzione**
   - **Estensibilità**
     - Il Sistema deve essere progettato in modo tale che sia possibile aggiungere moduli su richiesta del cliente.
   - **Manutenibilità**
     - Tutte le costanti applicative saranno modificabili tramite appositi file di configurazione e specifiche tabelle del database.
   - **Portabilità**
     - Il Sistema deve essere fruibile su tutti i dispositivi mobile e desktop in maniera indipendente dal sistema operativo o dall'hardware utilizzato.
5. **Criteri dell'utente finale**
   - **Usabilità**
     - Il Sistema deve avere un'interfaccia semplice e immediata in modo da consentire un'interazione rapida ed efficiente, così da ridurre i tempi di controllo.

1.2.1 Design Trade-Off

Tempo di rilascio Vs Correttezza

Per rispettare le scadenze del Progetto potrebbe essere necessaria l'implementazione parziale di alcune delle funzionalità richieste.

Dovremmo implementare un QR-code attraverso il quale un cliente può visionare il menù ma anche qui eviteremo di farlo.

Tempo di rilascio Vs Sicurezza

Per rispettare le scadenze del Progetto si è deciso di effettuare solo i controlli più rilevanti in termini di accesso alle varie funzionalità in base al ruolo dell'utente a discapito della sicurezza del servizio.

Tempo di rilascio Vs Manutenibilità

Per rispettare le scadenze del Progetto si considera l'eventualità della presenza di alcuni bug noti con la possibilità di risolverli con una versione successiva, oppure di poter aggiungere nuove funzionalità non specificate all'interno dei requisiti.

Prestazioni Vs Costi

Per garantire la disponibilità del servizio in maniera persistente, si è deciso di utilizzare un'infrastruttura basata su cloud a discapito dei costi di realizzazione e manutenzione.

Prestazioni Vs Sicurezza

Per avere un sistema responsivo e veloce, si è preferito evitare controlli ridondanti a discapito della sicurezza per migliorare le prestazioni.

1.3 Definizioni e acronimi

In questa sezione descriveremo i termini che sono stati utilizzati all'interno del documento stesso divisi in tre sezioni principali: definizioni, acronimi ed abbreviazioni.

1. **Definizioni**:
   - **Ordinazione**: nome con il quale ci si riferisce all'operazione effettuata dal manager per rimpinguare il magazzino;
   - **Comanda**: nome con il quale ci si riferisce all'operazione effettuata dal cameriere al fine di assegnare un'ordinazione ad un tavolo;
   - **Bartender**: nome con cui si indica un operatore bar specializzato in miscelazione. È una figura professionale che utilizza tecniche che velocizzano molto la preparazione dei cocktail.
   - **Manager**: nome con cui si indica la figura amministrativa della piattaforma, la quale possiede gli accessi a tutte le operazioni disponibili.
2. **Acronimi**:
   - **RV_[n]**: Regole di Vincolo_[numero]
   - **VIR[n]**: Vincolo di Integrità Referenziale_[numero];
   - **RDBMS**: Relational Database Management System (gestore di database basati sul modello relazionale);
   - **DB**: DataBase, ovvero "Base di Dati" utilizzata per memorizzare i dati persistenti;
   - **CRUD**: Create, Read, Update e Delete. Queste quattro parole fanno riferimento alle quattro principali operazioni che si svolgono su un database relazionale, ovvero la creazione di tabelle, dati e relazioni (create), la loro lettura (read), la loro modifica tramite aggiornamento (update) o eliminazione (delete);
   - **UUID**: Universally Unique Identifier;
   - **SDD**: System Design Document;
   - **GDPR**: General Data Protection Regulation;
   - **UI**: User Interface;
   - **UX**: User Experience;
   - **WCAG**: Web Content Accessibility Guidelines;
   - **IBA**: International Bartender Association.

1.4 Riferimenti

Per stilare la presente documentazione, si è preso come riferimento le slide fornite dal Docente del corso di Ingegneria del Software, Carmine Gravino, inserite nella sezione "M3" della piattaforma di e-learning della facoltà di Informatica. Inoltre, è stato consultato il libro di testo "Object-Oriented Software Engineering Using UML, Patterns and Java: Third Edition, di Bernd Bruegge ed Allen H. Dutoit".

1.5 Panoramica

Dopo questa prima sezione di introduzione del presente Documento, il punto 2 descriverà brevemente l'architettura del Sistema corrente, mentre al punto 3 verrà fornita una dettagliata descrizione dell'architettura del Sistema proposto. In particolare, questa sezione descriverà la decomposizione in sottosistemi, la corrispondenza tra hardware e software, la gestione dei dati persistenti, il controllo degli accessi, la sicurezza, il flusso di controllo e la gestione delle condizioni limite del Sistema proposto. Infine, il punto 4 descriverà i servizi offerti dai sottosistemi individuati.

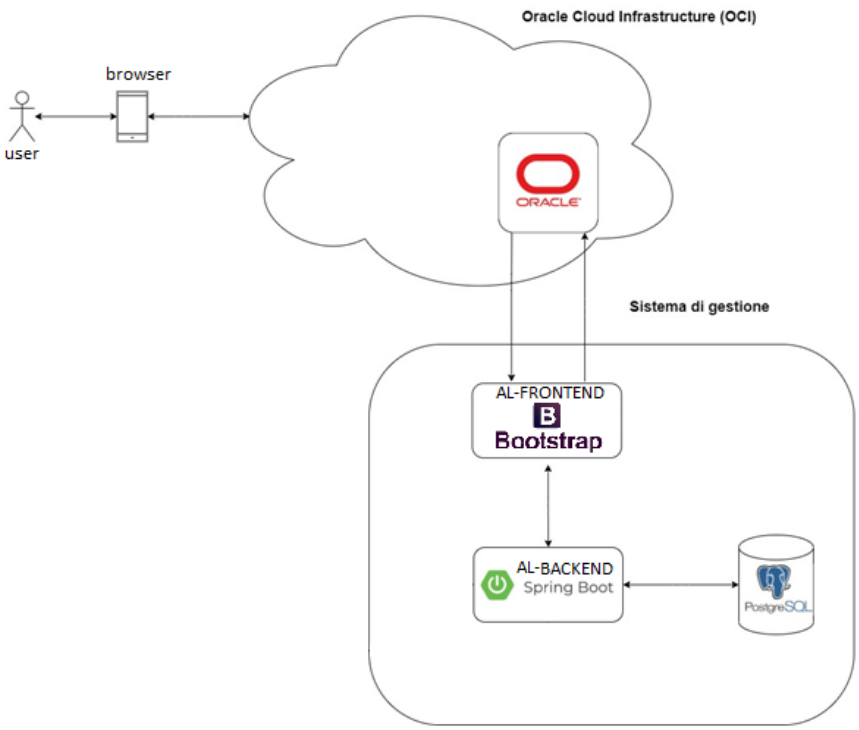
2 Architettura Sistema corrente

Il Sistema che si vuole realizzare è un progetto Greenfield, sviluppato a partire da una situazione reale non informatizzata. Inoltre, le possibili alternative a questo software sul mercato non considerano tutte le funzionalità di AlcoList in un unico servizio e quindi, non esiste una reale architettura a cui è possibile confrontare in maniera ragionevole il sistema.

3 Architettura Sistema Proposto

Il Sistema proposto è una Web Application che vuole facilitare, velocizzare e rendere più precisa la gestione di un Bar tenendo conto di magazzino, dipendenti e statistiche di vendita.

L'architettura di AlcoList è composta da due ambienti: uno dedicato al DEV (development) per lo sviluppo dei componenti costituenti della piattaforma ed uno dedicato al PROD (produzione) per la messa in esecuzione della piattaforma in ambiente di produzione. Tutti i componenti sviluppati verranno riportati in container Docker creati a partire da immagini le quali sono il risultato di un'opportuna compilazione di Dockerfile. Un container è un'unità software standard che impacchetta il codice e tutte le sue dipendenze in modo che l'applicazione venga eseguita in modo rapido e affidabile in maniera indipendente dall'hardware. Un'immagine del contenitore Docker è un pacchetto di software leggero, autonomo ed eseguibile che include tutto il necessario per eseguire un'applicazione: codice, runtime, strumenti di sistema, librerie di sistema e impostazioni. Le componenti sviluppate sono: AL-frontend e AL-backend, gestite da un docker compose in ambiente DEV e da un orchestratore di container Kubernates in ambiente PROD. Per lo sviluppo dell'ambiente DEV si sfruttano le macchine locali, mentre, per l'ambiente PROD si sfruttano le potenzialità offerte dal cloud, nel nostro caso Oracle Cloud Infrastructure. Di seguito un'immagine riassuntiva dell'infrastruttura dell'ambiente di PROD.

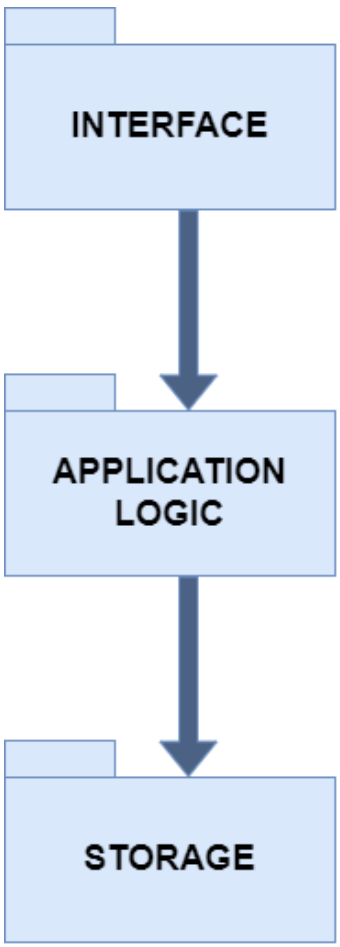


L'architettura software scelta per la realizzazione del Sistema è la Three Tier, la quale offre la possibilità di eseguire ciascun tier sulla propria infrastruttura, offre numerosi vantaggi, tra cui uno sviluppo più veloce e una maggiore scalabilità, affidabilità e sicurezza.

Questo pattern architetturale si compone di tre livelli:

1. **Interface**: è il tier di presentazione, dunque delle interfacce utente. Si occupa di mostrare le informazioni all'utente e di raccogliere informazioni da quest'ultimo. Nella implementazione sarà gestito dalle componenti di front-end;
2. **Application Logic**: è il tier della logica di business dell'applicazione. Si occupa di elaborare le informazioni raccolte nel tier Interface e di aggiungere, rimuovere o modificare i dati nel tier Storage. Nella implementazione sarà gestito dalla componente di backend;
3. **Storage**: è il tier dell'accesso ai dati. Si occupa della persistenza delle informazioni e della loro gestione e archiviazione. Nella implementazione sarà gestito dal database PostgreSQL;

Nel Sistema realizzato con la suddetta architettura tutte le comunicazioni passano attraverso l'Application Logic tier; l'Interface tier e lo Storage tier non comunicano mai direttamente tra loro. Al fine di favorire la comunicazione e il disaccoppiamento tra Application Logic e Interface viene sfruttata la progettazione API RESTful che mira a supportare l'indipendenza dalla piattaforma e l'evoluzione del servizio.



3.1 Panoramica

Di seguito, vengono illustrati i risultati della fase di progettazione del Sistema Proposto. Viene descritta l'architettura del sistema, stabilita in base alla struttura del sito web e alle funzionalità che deve offrire. Particolare attenzione è stata riposta nella gestione dei dati persistenti, i quali risultano fondamentali allo scopo del sistema proposto.

3.2 Decomposizione in Sottosistemi

L'**Application Logic** layer contiene le seguenti componenti:

- **UserAccount:** si occupa di gestire tutte le operazioni che possono essere effettuate per gli utenti
- **Cocktail: s**i occupa di gestire tutte le operazioni che possono essere effettuate sui cocktail
- **Ingredient: s**i occupa di gestire tutte le operazioni che possono essere effettuate per gli ingredienti
- **Product: s**i occupa di gestire tutte le operazioni che possono essere effettuate sui prodotti
- **Ordination: s**i occupa di gestire tutte le operazioni che possono essere effettuate sugli ordini
- **Tables:** si occupa di gestire tutte le operazioni che possono essere effettuate sui tavoli.
- **Statistics:** si occupa di gestire tutte le operazioni che possono essere effettuate per il recupero delle statistiche della dashboard.

Lo **Storage** layer contiene le seguenti componenti:

- **CocktailRepository;**
- **UserAccountRepository;**
- **ProductRepository;**
- **IngredientRepository**
- **MessageRepository**
- **OrderedCocktailRepository**
- **OrdinationRepository**
- **TablesRepository**

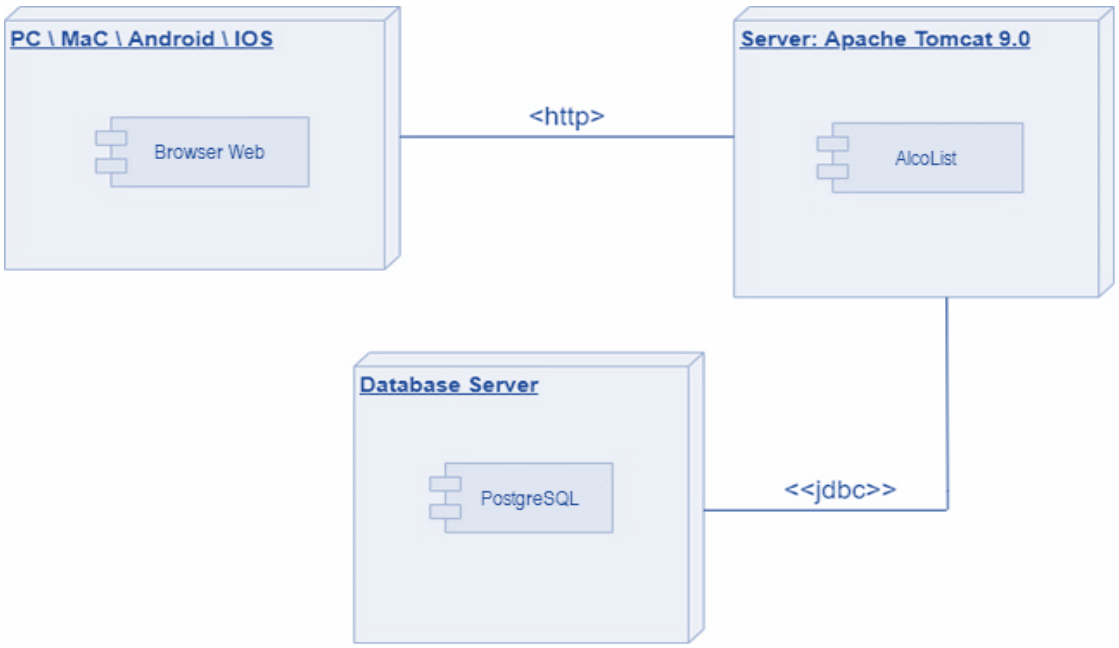
Le componenti appena elencate modellano le caratteristiche della relativa entità e consentono di interfacciarsi con la stessa nel database per compiere le operazioni CRUD;

L'**Interface** layer contiene le seguenti componenti:

- **UtenteGUI:** interfaccia comune a tutti gli utenti della piattaforma e che contiene anche le interfacce:
  - **CamiereGUI:** interfaccia grafica per il Cameriere.
  - **BartenderGUI:** interfaccia grafica per il Bartender.
  - **ManagerGUI:** interfaccia grafica per il Manager.

3.3 Mapping Hardware/Software

La piattaforma AlcoList è realizzata come una Web Application. Per poter interagire con il Sistema è necessario accedervi tramite un web browser. Le componenti di frontend sono sviluppate utilizzando JavaScript, jQuery ed Ajax per comunicare col backend e Bootstrap per mostrare graficamente le informazioni all'utente. Per la comunicazione con il backend si è preferito utilizzare API REST tramite il protocollo HTTP e lo scambio dati utilizzando JSON per avere un'indipendenza da linguaggio del backend. La persistenza dei dati è invece mantenuta tramite l'utilizzo di un database PostgreSQL, che sarà contattato dal Server tramite JDBC. Per garantire scalabilità e persistenza ai guasti il sistema preposto sarà ospitato su Cluster Kubernetes.



3.4 Gestione dei Dati Persistenti

Per la memorizzazione dei dati persistenti si è deciso di utilizzare un RDBMS (Relational Database Management System) poiché permette di accedere in modo semplice ed efficiente ai dati, conservandone la consistenza, la privatezza e l'affidabilità. Possiamo effettuare ricerche complesse (ad esempio, la ricerca di report sulla base di un filtro scelto) che, se compiute su tradizionali archivi analogici, comporterebbero ampio dispendio di tempo e risorse.

Inoltre, fornisce un accesso concorrente ai dati mantenendone la coerenza anche in condizione di multiutenza e, soprattutto, possiede un meccanismo di permessi per cui utenti con operazioni diverse possono accedere a sezioni diverse della base di dati in maniera protetta.

3.4.1 Dizionario dei dati

ENTITÀ	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
Role	Rappresenta i possibili ruoli che possono assumere i dipendenti.	- name	Id
User_Account	Rappresenta un utente all'interno della piattaforma.	- date_delete - email - main_role - name - password - surname - uuid	Id
User_Account_Roles	Rappresenta i ruoli associati ad un singolo utente.	- user_account_id - roles_id - uuid	id
Tables	Rappresenta i tavoli presenti nel locale.	- number - seats - uuid	Id
Product	Rappresenta il singolo prodotto presente in magazzino.	- alcoholic_percentage - category - is_present - name - ml - pathfileImg - uuid	Id
Cocktail	Rappresenta il Cocktail.	- name - description - favour - isalcoholic - isiba - price - pathfileImg - uuid	Id
Ingredient	Rappresenta un ingrediente del cocktail riferito ad uno specifico prodotto.	- product - is_optional - quantity - cocktail	id

Ordination	Rappresenta la comanda.	- status - total - created_by - delivered_by - executed_by - id_tables - uuid	Id
Ordination_ordered_cocktails	Rappresenta tutti i cocktails ordinati per una specifica comanda.	- ordination_id - ordered_cocktails_id	

RELAZIONE	DESCRIZIONE	ENTITÀ COINVOLTE
Possedere	Associa ad un Cocktail uno o più Ingredient.	Cocktail (1, N) Ingredient (1, 1)
Appartenere	Associa ad un Ordination uno o più Cocktail.	Ordination (1, N) Cocktail (1, N)
Occupare	Associa più Role a più User_Account.	Role (1, N) User_Account (1, N)
Consegnare	Associa un Table ad un Ordination.	Table (0, N) Ordination (1, 1)

REGOLE DI VINCOLO
(RV1) L'UUID dev'essere univoco e non può essere NULL.
(RV2) La password di un User_Account deve essere almeno di 8 caratteri e contenente almeno un carattere speciale, uno maiuscolo, uno minuscolo ed almeno un numero.
(RV3) Tutti i campi non devono superare i 50 caratteri di lunghezza.

Modello Logico

Role (id, name)

User_Account (id, date_delete, email, surname, name, main_role, password, uuid)

User_Account_Roles (id, *user_account_id, roles_id, uuid*)

Tables (id, number, seats, uuid)

Product (id, name, category, ml, alcoholic_percentage, is_present, pathfileImg, uuid)

Cocktail (id, name, description, favour, isalcoholic, isiba, price, pathfileImg, uuid)

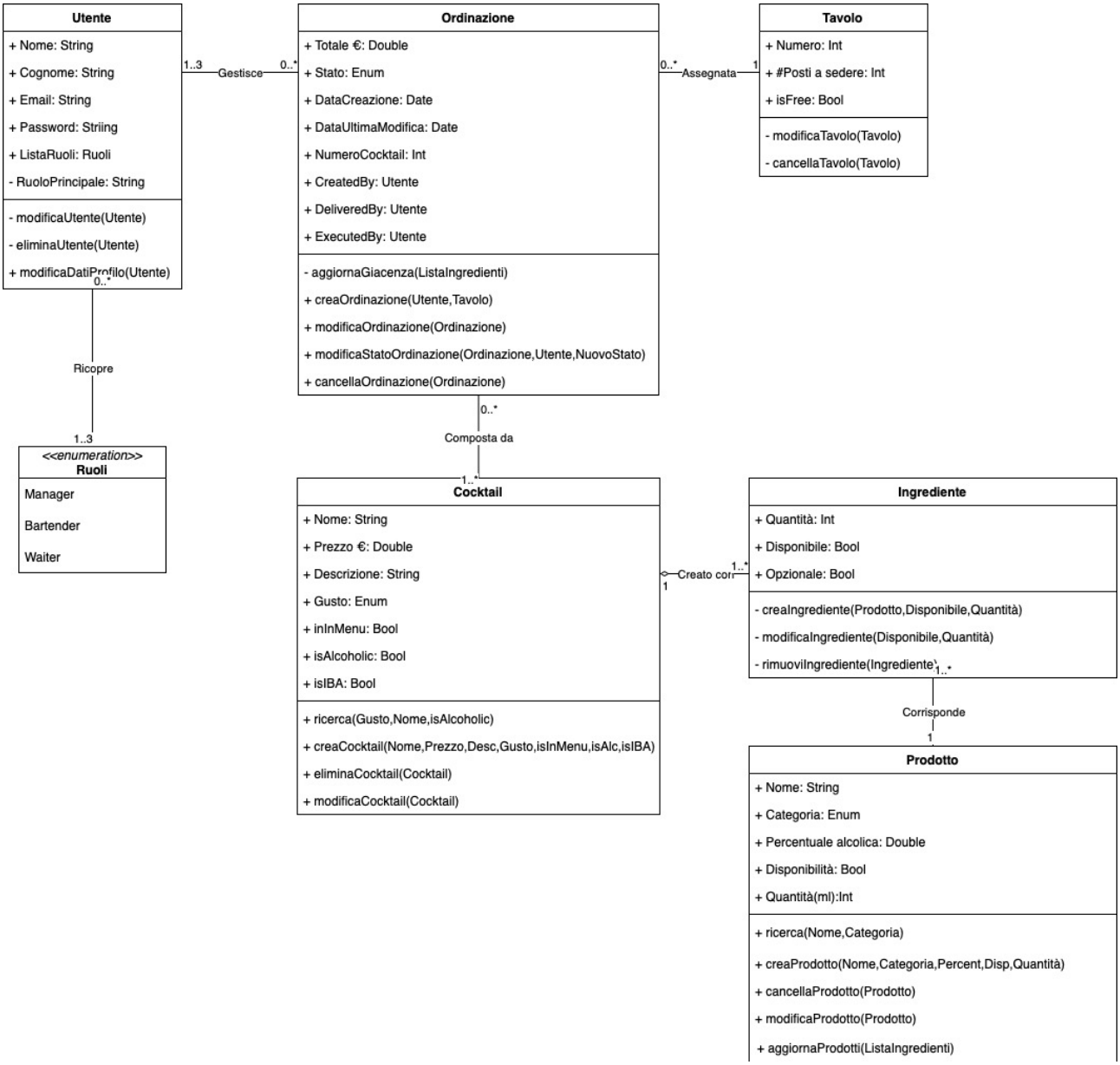
Ingridient (id, is_optional, quantity, *product*, *cocktail*)

Ordination (id, status, total, created_by, delivered_by, executed_by, uuid, *id_tables*)

Ordination_ordered_cocktails (*ordination_id, ordered_cocktail_id*)

VINCOLI DI INTEGRITÀ REFERENZIALE
(VIF1) La chiave esterna "user_account_id" della tabella User_Account_Roles" ha un vincolo di integrità referenziale con la chiave primaria "*id*" della tabella "*User_Account*".
(VIF2) La chiave esterna "roles_id" della tabella "*User_Account_Roles*" ha un vincolo di integrità referenziale con la chiave primaria "*id*" della tabella "*Role*".
(VIF3) La chiave esterna "*product*" della tabella "*Ingridient*" ha un vincolo di integrità referenziale con la chiave primaria "id" della tabella "*product*".
(VIF4) La chiave esterna "cocktail" della tabella "*Ingridient*" ha un vincolo di integrità referenziale con la chiave primaria "id" della tabella "*cocktail*".
(VIF5) La chiave esterna "id_tables" della tabella "*Ordination*" ha un vincolo di integrità referenziale con la chiave primaria "*id*" della tabella "*Tables*".
(VIF6) La chiave esterna "ordination_id" della tabella "Ordination_ordered_cocktails" ha un vincolo di integrità referenziale con la chiave primaria "id" della tabella "Ordination".
(VIF7) La chiave esterna "*ordered_cocktail_id*" della tabella "*Ordination_ordered_cocktails*" ha un vincolo di integrità referenziale con la chiave primaria "id" della tabella "*Cocktail*".

Class Diagram Ristrutturato



3.5 Controllo degli accessi e sicurezza

La sicurezza per l'accesso al Sistema è garantita tramite autenticazione con credenziali personali. Ciascun Cameriere e Bartender ha delle credenziali che gli sono state assegnate al momento dell'assunzione e che saranno registrate dal Manager (unico user che potrà generarne di nuovi).
Per motivi di privacy la password di Cameriere e Bartender viene autogenerata alla creazione ed inviata alla e-mail inserita dal Manager nel form di registrazione.
Nella matrice di accesso che segue riportiamo:

- nella prima colonna, gli attori del Sistema;
- nella prima riga, un'istanza delle classi del nostro Sistema;
- nelle celle risultanti dall'incrocio tra riga e colonna, abbiamo le operazioni che l'attore può compiere sull'oggetto in questione.

	ORDINAZIONE	COCKTAIL	INGREDIENTE	PRODOTTO	USER
MANAGER	1. Crea comanda 2. Modifica comanda 3. Elimina comanda	1. Visualizza cocktail 2. Modifica cocktail 3. Crea cocktail 4. Elimina cocktail	1. Crea ingrediente 2. Elimina ingrediente	1. Crea prodotto 2. Elimina prodotto	1. Crea user 2. Elimina user 3. Modifica user
BARTENDER	1. Crea comanda 2. Modifica comanda 3. Elimina comanda	1. Visualizza cocktail 2. Modifica cocktail 3. Crea cocktail 4. Elimina cocktail	1. Crea ingrediente 2. Elimina Ingrediente		
CAMERIERE	1. Crea comanda 2. Modifica comanda 3. Elimina comanda				

3.6 Controllo del flusso globale del Software

Il Sistema sfrutta l'architettura "event driven" poiché il ciclo principale aspetta il verificarsi di un evento per eseguire una determinata azione. Nel nostro caso un esempio concreto potrebbe essere il Sistema che attende la ricezione di una nuova comanda per registrarla nel database.

3.7 Condizione limite

Avvio sistema. All'avvio, il Sistema necessita di un Web Server che fornisca il servizio di accesso ad un database (nel nostro caso PostgreSQL) per la gestione dei dati persistenti e l'interpretazione ed esecuzione del codice lato Server. Quando un utente (Manager, Bartender o Cameriere) eseguirà il login nel Sistema, gli verrà presentata una pagina nel Browser Web con le operazioni che gli è consentito svolgere a seconda del affidatogli.

Terminazione. Una volta chiusa l'applicazione, il Sistema eseguirà un logout automatico terminando la sessione dell'utente. Il Sistema non si occupa di salvare eventuali dati lasciati in sospeso durante l'utilizzo. Pertanto, al riavvio del Sistema, quest'ultimo non ripresenterà i dati immessi in precedenza se questi non sono stati opportunamente salvati.

Fallimento. Si possono individuare diverse situazioni di fallimento: (CLOUD)

- Nel caso di guasti dovuti al sovraccarico del database con successivo fallimento dello stesso, di interruzioni inaspettate dell'alimentazione o nel caso in cui si verifichi un errore nell'hardware (ad es. un dispositivo di archiviazione di massa), il sistema sfrutta repliche dell'ambiente utilizzato per gestire gli accessi al DB per rendere sempre disponibile la risorsa, utilizzando l'orchestratore di container Kubernates.
- Se un utente invia al Sistema informazioni errate oppure l'utente non sottomette delle informazioni per la corretta esecuzione di +un'operazione, il Sistema risponderà con un messaggio di errore;

4 Servizi dei Sottosistemi

Interface:

- **CamerieреGUI**:
  - **Gestione Comanda**: permette al cameriere di visualizzare lo stato delle comande per ogni tavolo ed innescarne la creazione di nuove.
- **BartenderGUI**:
  - **Gestione Cocktail**: permette al bartender di visualizzare o comporre un nuovo cocktail a partire dagli ingredienti disponibili e listati (con relativa modifica o eliminazione);
  - **Gestione Ingredienti**: permette al bartender di aggiungere, modificare o eliminare un ingrediente;
- **ManagerGUI**
  - **Gestione Dipendenti:** permette la visualizzazione in lista di tutti gli user (con relativa aggiunta, modifica o eliminazione) memorizzati nel Sistema;
  - **Gestione Menù:** permette la visualizzazione in lista di tutti i cocktail offerti dal Bar e la modifica della stessa;
  - **Gestione Magazzino**: permette la visualizzazione in lista di tutti i prodotti (con relativa aggiunta, modifica o eliminazione) presenti in magazzino (e le relative quantità);
  - **Visualizza Statistiche** : permette di avere un'overview del sistema e dell'andamento del Bar.
- **AutenticazioneGUI**
  - **Login**: permette al generico user di autenticarsi tramite credenziali;
  - **Logout**: permette al generico user di disconnettersi;

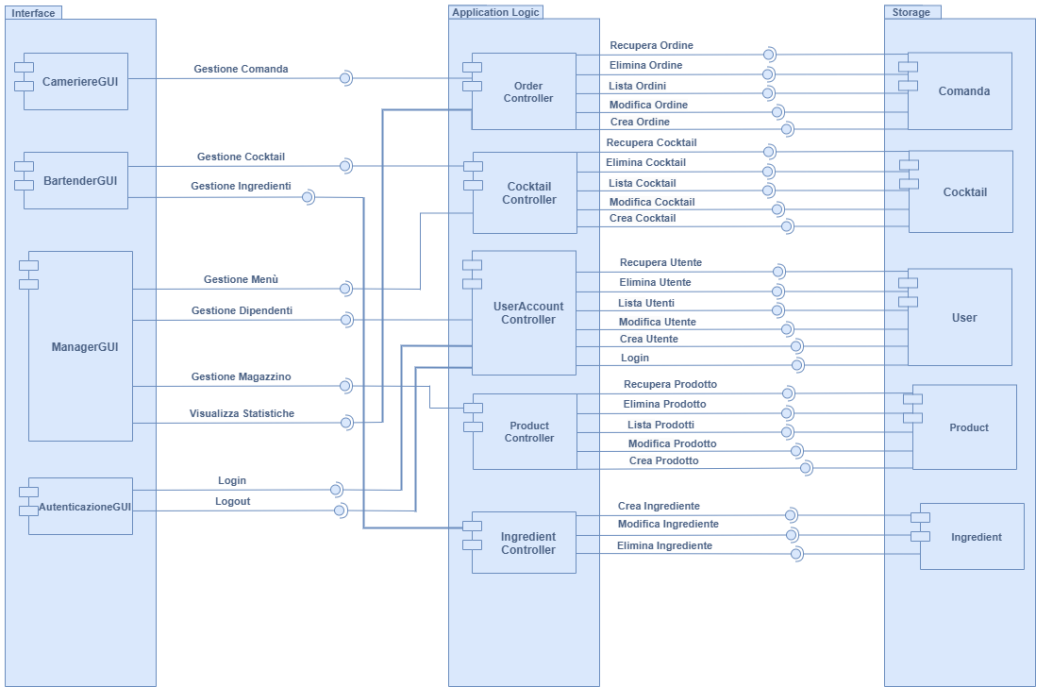
Application Logic:

- **UserAccountController**:
  - **Crea utente**: salva i dati dell'utente nel DB;
  - **Recupera utente**: restituisce i dati di un utente;
  - **Lista utenti**: recupera una lista di utenti ricercandoli per ruolo;
  - **Cancella utente**: elimina un utente dal DB.
  - **Modifica utente**: permette di modificare i dati collegati all'utente
  - **Login**: verifica le credenziali inserite e in caso di dati corretti restituisce i dati dell'utente
  - **Logout**: permette la disconnessione dal sistema dell'utente loggato

- **CocktailController**:
  - **Recupero cocktail**: restituisce i dati collegati alla ricetta del cocktail richiesta inclusi gli ingredienti collegati;
  - **Elimina cocktail**: elimina il cocktail con i relativi ingredienti collegati;
  - **Recupera lista paginata di cocktail**: recupera una lista di cocktail;
  - **Modifica cocktail**: permette di modificare le informazioni generiche collegate ad un cocktail;
  - **Crea cocktail**: crea un nuovo cocktail;
- **IngredientController**:
  - **Crea Ingrediente**: aggiunge un ingrediente ad un cocktail;
  - **Elimina ingrediente**: elimina un ingrediente collegato ad un cocktail;
  - **Modifica ingrediente**: cancella i dati di un Docente dal DB.

- **ProductController**:
  - **Crea prodotto**: aggiunge un prodotto;
  - **Elimina prodotto**: elimina un prodotto;
  - **Modifica prodotto:** modifica i dati di un prodotto inclusa la giacenza.
  - **Lista prodotti**: restituisce una lista di prodotti.

- **OrderController**:
  - **Recupero ordine**: restituisce i dati collegati all'ordinazione inclusi i cocktail collegati;
  - **Elimina ordine**: elimina l'ordine;
  - **Lista ordini**: recupera una lista di ordini;
  - **Modifica ordine**: permette di modificare un'ordinazione con il relativo stato;
  - **Crea ordine**: crea un nuovo ordine e lo salva sul DB;



5 Glossario

In questa sezione descriveremo i termini tecnici che sono stati utilizzati all'interno del Documento stesso.

- **API**: Application Program Interface, acronimo che indentifica delle librerie che forniscono funzioni implementate.
- **Login**: operazione di autenticazione di un Docente o di un Direttore di Dipartimento mediante l'uso di credenziali personali.
- **Logout**: disconnessione dalla propria area personale.
- **Form**: insieme di campi da compilare e sottomettere. I campi possono essere obbligatori o facoltativi.
- **User**: persona che sfrutta le funzionalità del Sistema.
- **Web Application**: applicazione accessibile via Web per mezzo di una rete Internet.
- **Web Browser/Browser**: applicazione software per l'acquisizione, la presentazione e la navigazione di risorse sul web.
- **Web Server**: applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client, tipicamente un Web Browser.
- **Greenfield**: aggettivo di un progetto che indica la realizzazione dello stesso a partire solamente dall'idea, senza l'utilizzo di componenti già implementate da altri e disponibili online come risorse pubbliche.
- **Throughput**: quantità di lavoro effettuata in un dato lasso di tempo.
- **Layer**: indica uno strato, un livello, una partizione.

- **RDBMS**: Relational Database Managament System.
- **PostegreSQL**: è un RDBMS basato sul linguaggio SQL, composto da un client a riga di comando e un server.
- **Three Tier**: letteralmente "tre livelli", è il nome proprio di un pattern utilizzato per la realizzazione dell'architettura software di un Sistema.
- **Bug**: errore di funzionamento di un sistema, di una funzionalità o di un programma.
- **PROD**: diminutivo di production (produzione).
- **DEV**: diminutivo di development (sviluppo).