System Design Document

BeeHave

Riferimento	C14_SDD_ver 1.0
Versione	1.0
Data	12/12/2022
Destinatario	Prof.ssa Filomena Ferrucci, Prof. Fabio Palomba
Presentato da	C14 Team A.P.Hive
Approvato da	Gianmario Voria, Antonio Della Porta

Revision History

Data	Versione	Descrizione	Autori
20/11/2022	0.1	Prima stesura	GV, ADP
25/11/2022	0.2	Scrittura dei Design Goals	Tutto il Team
25/11/2022	0.3	Scrittura dei Trade Offs	Tutto il Team
26/11/2022	0.4	Scrittura del Paragrafi: 1.1, 1.2	GR, LB
26/11/2022	0.5	Definizione Sottosistemi	Tutto il Team
28/11/2022	0.6	Scrittura Introduzione 3.4	GR, TDP
28/11/2022	0.7	Scrittura paragrafi 1.3, 1.5, 3.3	TDP
29/11/2022	0.8	Creazione Diagramma Architetturale	Tutto il Team
30/11/2022	0.9	Creazione Matrice degli accessi	Tutto il Team
30/11/2022	0.10	Scrittura paragrafi 3.6	IG
01/12/2022	0.11	Creazione del Deployment Diagram	Tutto il Team
01/12/2022	0.12	Scrittura sezione 4 Sottosistema GAU	IG
02/12/2022	0.13	Definizione Use Case Boundary Conditions: UCBC_01, UCBC_02, UCBC_03	Tutto il Team
08/12/2022	0.14	Scrittura Glossario	LB
11/12/2022	1.0	Revisione del documento	Tutto il Team

Team Members

Nome	Ruolo	Acronimo	Contatto
Gianmario Voria	Project Manager	GV	g.voria6@studenti.unisa.it
Antonio Della Porta	Project Manager	ADP	a.dellaporta26@studenti.unisa.it
Luigi Bacco	Team Member	LB	l.bacco2@studenti.unisa.it
Irene Gaita	Team Member	IG	i.gaita1@studenti.unisa.it
Maria Lucia Fede	Team Member	MLF	m.fede1@studenti.unisa.it
Gianluca Ronga	Team Member	GR	g.ronga5@studenti.unisa.it
Thomas De Palma	Team Member	TDP	t.depalma@studenti.unisa.it

Sommario

1. Introduzione

1.1. Obiettivo del Sistema

BeeHave intende porsi come punto di contatto fra apicoltori e clienti, col fine ultimo di supportare l'espansione del mercato apistico per favorire la diffusione delle api.

Il sistema permette l'iscrizione sia di apicoltori, che potranno prestare un servizio di assistenza agli utenti, sia di clienti, la quale registrazione è necessaria per concludere acquisti e adozioni.

BeeHave si struttura principalmente in tre componenti: la vendita del miele da parte degli apicoltori, il processo di adozione degli alveari in modo parziale o totale, e l'assistenza agli utenti.

1.2. Design Goals

In questa sezione sono specificati tutti i design goals cui il sistema punta a raggiungere.
Ogni Design goal ha:

- Un id
- Un rank
- Un nome
- Una descrizione
- Una categoria (Dependability, Performance, Maintenance, Cost, End User)
- I/Il RNF a cui è associato

Ad ogni design goal è assegnato un numero, detto Rank, il quale designa l'importanza del design goal a cui esso fa riferimento, un numero più basso indica una priorità più alta e viceversa.

La sezione corrente specifica, inoltre, una serie di **trade-offs**. Data l'impossibilità dell'esistenza di un sistema perfetto, questi ultimi ci consentono di bilanciare alcune scelte, in modo tale da avere miglioramenti in alcuni aspetti a discapito di altri.
Ogni Trade-Off ha:

- Un nome che indica gli aspetti di riferimento
- Una descrizione che indica le scelte prese dagli sviluppatori

Rank	Design Goal	Descrizione	Categoria	RNF
8	**DG_1 Facilità d'utilizzo**	Il sistema deve risultare facilmente comprensibile ed utilizzabile anche da un'utenza meno esperta, facendo uso delle "8 regole d'oro di Shneirderman" per il design delle interfacce grafiche.	End User	RNF_USA_1
9	**DG_2 Interfaccia intuitiva**	Il sistema deve garantire un design chiaro e utilizzabile. Deve essere raggiunto attraverso un'interfaccia utente che permette di eseguire azioni in modo semplice, familiare e consistente, rendendo ben esplicita la funzionalità di ogni elemento visibile.	End User	RNF_USA_2
4	**DG_3 Integrità delle operazioni**	Il sistema deve garantire l'integrità delle operazioni, assicurando che esse siano eseguite interamente e con successo attraverso le transazioni.	Dependability	RNF_AFF_1

2	**DG_4 Confidenzialità dei dati**	Il sistema deve permettere una divisione tra le varie categorie di utenti, ovvero garantire che i dati e le risorse siano preservati dal possibile utilizzo o accesso da parte di soggetti non autorizzati a compiere una determinata operazione. Il sistema deve effettuare un controllo mirato sugli accessi alle risorse.	Dependability	RNF_AFF_3
12	**DG_5 Notifica errori del sistema**	Il sistema deve segnalare all'utente l'errore, notificandoglielo attraverso un messaggio e aiutandolo in seguito a completare le operazioni che stava svolgendo.	End User	RNF_AFF_4
7	**DG_6 Navigazione concorrente**	Il sistema deve essere affidabile in situazioni di carico elevato garantendo prestazioni elevate anche con un alto numero di utenti connessi in contemporanea assegnando un sottoprocesso ad ogni sessione utente.	Performance	RNF_PRES_4

5	**DG_7 Tempi di risposta**	l sistema deve garantire un tempo di risposta non superiore a 6 secondi.	Performance	RNF_PRES_2
10	**DG_8 Sistema Responsive**	Il sistema deve essere provvisto di un'interfaccia grafica di tipo responsive in grado di adattarsi ad ogni tipo di schermo utilizzando breakpoints per l'implementazione del design.	End User	RNF_PRES_3
6	**DG_9 Portabilità del sistema**	Il sistema deve essere facilmente modificabile sia su ambiente Windows che su ambiente Macintosh utilizzando il linguaggio Python, basato su VM.	Maintenance	RNF_SOST_1, RNF_SOST_2
11	**DG_10 Piattaforma Web**	Il sistema deve essere sviluppato seguendo lo standard architetturale Three Tier di una piattaforma web based.	Maintenance	RNF_IMP_1
13	**DG_11 Vincoli di Manutenibilità**	Il sistema deve essere progettato per garantire semplicità ed economicità nell'esecuzione di manutenzioni ordinarie e aggiornamenti bisettimanali scomponendolo in moduli.	Maintenance	RNF_IMP_4

1	**DG_12 Persistenza dei dati**	Il servizio di persistenza dei dati verrà garantito tramite l'utilizzo di un database MySQL.	Dependability	RNF_INT_1
3	**DG_13 Rispetto della privacy**	Il sistema deve garantire il rispetto delle leggi in materia di privacy, specificatamente riguardanti il trattamento e la protezione dei dati personali non fornendo dati dell'utente a soggetti terzi.	End User	RNF_LEG_1
14	**DG_14 Rispetto dell'apicoltura**	Il sistema deve richiedere ad ogni apicoltore, in fase di registrazione, il rispetto della legge sulla disciplina dell'apicoltura 2004/313.	End User	RNF_LEG_2

TRADE-OFF

Trade-off	Descrizione
Concorrenza vs Costi	Per ottimizzare la gestione della concorrenza si può ricorrere all'utilizzo di processori aggiuntivi con maggiori capacità di calcolo mirate all'aumento di velocità nel multithreading. L'utilizzo di hardware più performante comporta un aumento dei costi.
Triade (CIA) vs Tempi di rilascio aggiornamenti	Garantiamo una maggiore confidenzialità, integrità e disponibilità a discapito dell'aumento dei ritardi dei tempi di aggiornamento bisettimanali.

1.3. Definizioni, Acronimi e Abbreviazioni

Si elencano tutte le definizioni, acronimi e abbreviazioni presenti all'interno del documento:

- **GU:** Gestione Utente
- **GAU:** Gestione Assistenza Utente
- **GV:** Gestione Vendita
- **GA:** Gestione Adozioni
- **RF:** Requisito Funzionale
- **RNF:** Requisito Non Funzionale
- **SDD:** System Design Document
- **DB:** DataBase
- **DG:** Design Goal
- **CIA:** Confidenzialità, Integrità, Disponibilità
- **USA:** Usabilità
- **LEG:** Legalità
- **INT:** Interfaccia
- **IMP:** Implementazione
- **SOST:** Sostenibilità
- **PRES:** Prestazioni
- **AFF:** Affidabilità
- **UCBC:** Use Case Boundary Conditions
- **DBMS:** DataBase Management System
- **UC:** Use Case
- **HTTP:** HyperText Transfer Protocol
- **HTTPS:** HyperText Transfer Protocol Secure
- **HTML:** HyperText Markup Language
- **CSS:** Cascading Style Sheets
- **JS:** JavaScript
- **SQL:** Structured Query Language
- **VM:** Virtual Machine
- **ER:** Entity-Relationship
- **MVC:** Model View Control

1.4. Riferimenti

Libro: -- Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition
Autori: -- Bernd Bruegge & Allen H.

- Di seguito una lista di riferimenti ad altri documenti utili durante la lettura:
  - Statement Of Work;
  - Test Plan;
  - Matrice di tracciabilità;
  - RAD;

1.5. Organizzazione del Documento

Il documento è organizzato in quattro punti

- **Introduzione:** Si introducono brevemente gli obiettivi del sistema e si descrivono i design goals in ordine di priorità e con i vari trade off. Si conclude questa prima sezione con le varie definizioni, acronimi e abbreviazioni usate nel documento e con i vari riferimenti.
- **Architettura del Sistema Attuale:** Si descrive l'architettura software del sistema corrente per poi avere la possibilità di metterla a confronto con l'architettura del sistema proposto.
- **Architettura del Sistema Proposto:** Si definisce come viene partizionato il sistema in sottosistemi, il mapping Hardware/Software e la gestione dei dati persistenti.
- **Servizi dei sottosistemi:** Si descrive la struttura dei sottosistemi con i vari servizi che forniranno e le condizioni limite del sistema.
- **Glossario:** Sono elencati tutti i termini tecnici, e le relative definizioni, presenti all'interno del documento

2. Architettura del Sistema Attuale

Al momento, non esista alcun software che metta a disposizione tutte le funzionalità offerte da BeeHave. Per questo motivo, non è possibile non è possibile effettuare il confronto con architetture già esistenti.

3. Architettura del Sistema Proposto

Questo paragrafo contiene una descrizione del sistema BeeHave dal punto di vista architetturale e dei servizi offerti.

3.1. Sintesi del Sistema

BeeHave ha come obbiettivo principale quello di sensibilizzare le persone sulla tematica della lotta all'estinzione delle api. Attraverso una piattaforma online viene fornita agli

apicoltori la possibilità di autofinanziarsi creando uno spazio virtuale sul quale mettere in adozione i propri alveari e gestire la vendita del miele prodotto.

Il sistema proposto è basato su una architettura **three-tier**, mantenendo la classica suddivisione dei layer in Model, View e Controller.

Per quanto riguarda la parte di front-end e generazione delle View, verranno utilizzati HTML, CSS, JS e Python.

La logica della piattaforma verrà invece sviluppata interamente in Python, attraverso il framework Flask per la gestione delle chiamate al back-end e il reindirizzamento alle View.

Per quanto riguarda la persistenza dei dati, verrà utilizzato un database MySQL, al quale sarà possibile collegarsi tramite framework Flask stesso.

3.2. Decomposizione in sottosistemi

Nella presente sezione sono stati individuati i seguenti sottosistemi:

- **Gestione Utente:** Si occupa di gestire la registrazione e l'autenticazione dei vari utenti, ossia cliente e apicoltore. Si occupa anche delle funzionalità di login, logout, visualizzazione area utente, modifica dati account ed eliminazione account.
- **Gestione Adozioni:** Per il cliente si occupa di gestire le funzionalità riguardanti la visualizzazione e l'adozione dei vari alveari disponibili, con le relative informazioni, la visualizzazione degli alveari adottati e il relativo stato. Per l'apicoltore si occupa delle funzioni riguardanti l'inserimento di un alveare disponibile per l'adozione e l'aggiornamento del suo stato.
- **Gestione Vendita:** Per l'apicoltore si occupa di gestire le funzionalità riguardanti l'inserimento, la modifica, la cancellazione di un prodotto e la visualizzazione dei prodotti che ha messo in vendita. Per il cliente si occupa delle funzioni di visualizzazione del catalogo dei prodotti, con la possibilità di filtrare la ricerca, di visualizzazione di un singolo prodotto e del suo acquisto.
- **Gestione Assistenza Utente:** Per l'apicoltore si occupa di gestire la funzione di inserimento della propria area assistenza e di filtrare le richieste. Per il cliente si occupa di gestire la funzionalità di richiesta assistenza. Per entrambi si occupa di gestire le funzionalità di visualizzazione e chiusura di un ticket assistenza.
- **Interfaccia Persistenza:** si interpone tra i sottosistemi e il sottosistema Persistenza.
- **Persistenza:** Gestisce la persistenza dei dati con un database.

DIAGRAMMA DELLE COMPONENTI

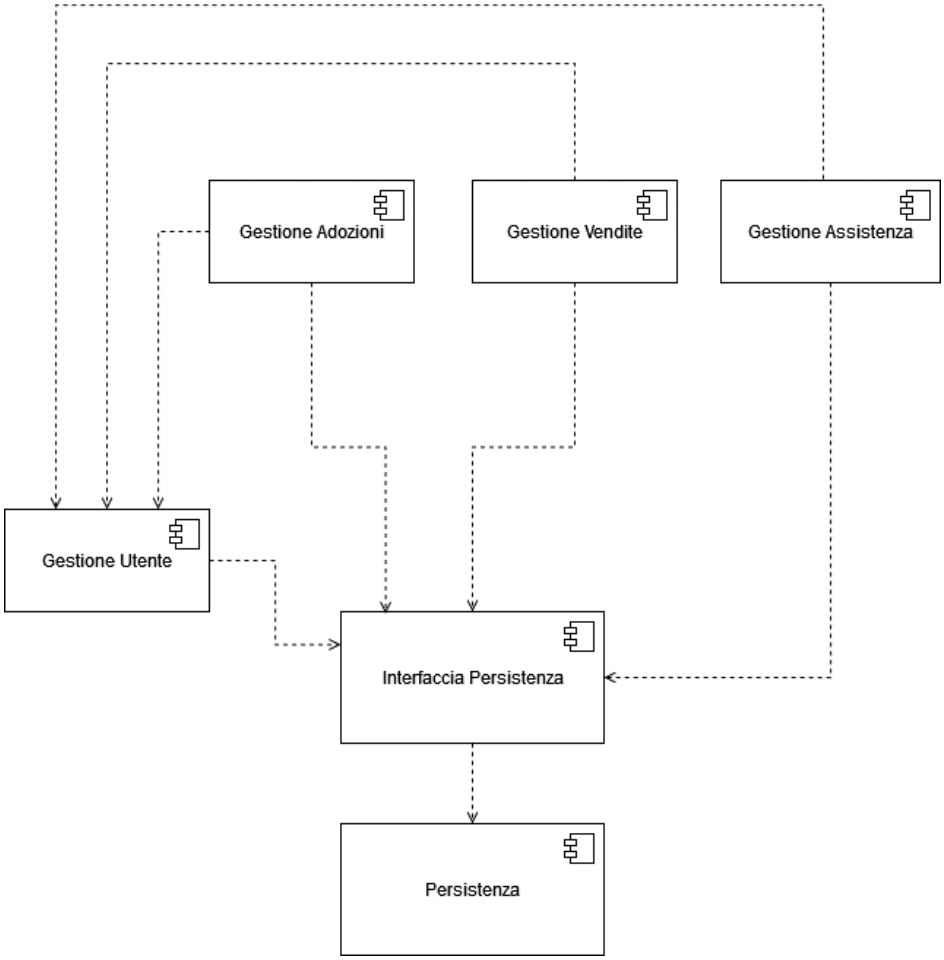
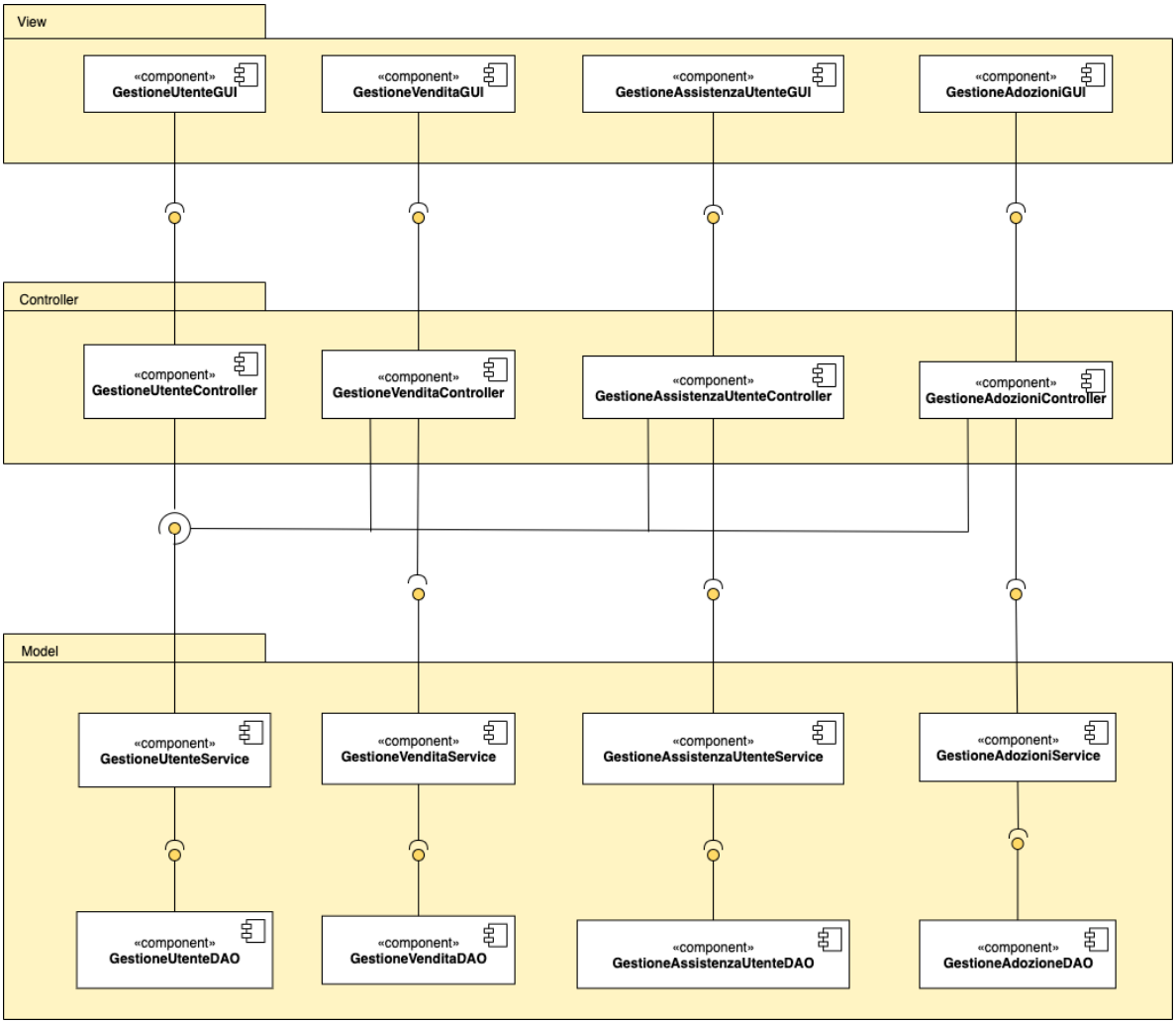
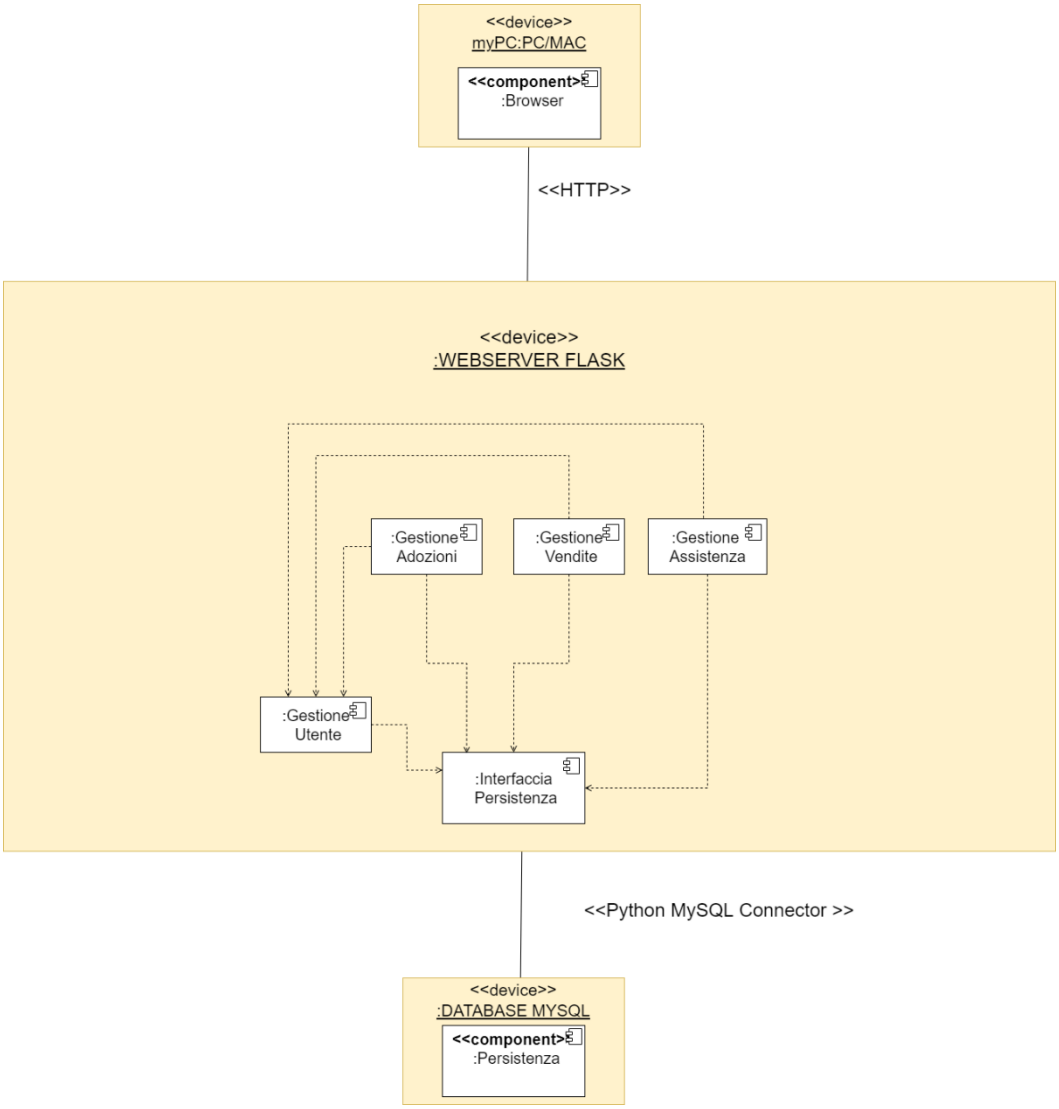


DIAGRAMMA ARCHITETTURALE



3.3. Mapping Hardware/Software

Il sistema utilizzerà una struttura hardware con architettura di tipo Client-Server. Il client è rappresentato da una qualsiasi macchina con una connessione Internet e un web browser installato su di esso, in modo da potersi connettere al server, tramite protocollo HTTPS, per interagire col sistema. Il nodo server è costituito da un dispositivo connesso alla rete Internet il quale eroga i servizi che riguardano il sistema proposto, gestendo la logica di business e la comunicazione con i diversi client. Attraverso l'utilizzo del DBMS MySQL, si gestiscono i dati persistenti relativi al dominio di applicazione.



3.4. Gestione dei dati persistenti

Per il salvataggio e la gestione dei dati persistenti del sistema, si è scelto di utilizzare un database relazionale, per garantire una facilità di accesso e di utilizzo dei dati garantendo la loro consistenza tramite l'utilizzo di un DBMS.
È stato scelto l'utilizzo di un DBMS, ed in particolare di MySQL, per questi aspetti:

- **Consistenza dei dati** grazie all'utilizzo di vincoli, utilizzati per non alterare la coerenza degli elementi ad ogni operazione di modifica del DBMS.

- **Protezione dei dati**, possibile grazie alla funzionalità in MySQL di permettere un accesso protetto ai dati, limitando l'accesso degli Utenti senza determinate autorizzazioni.

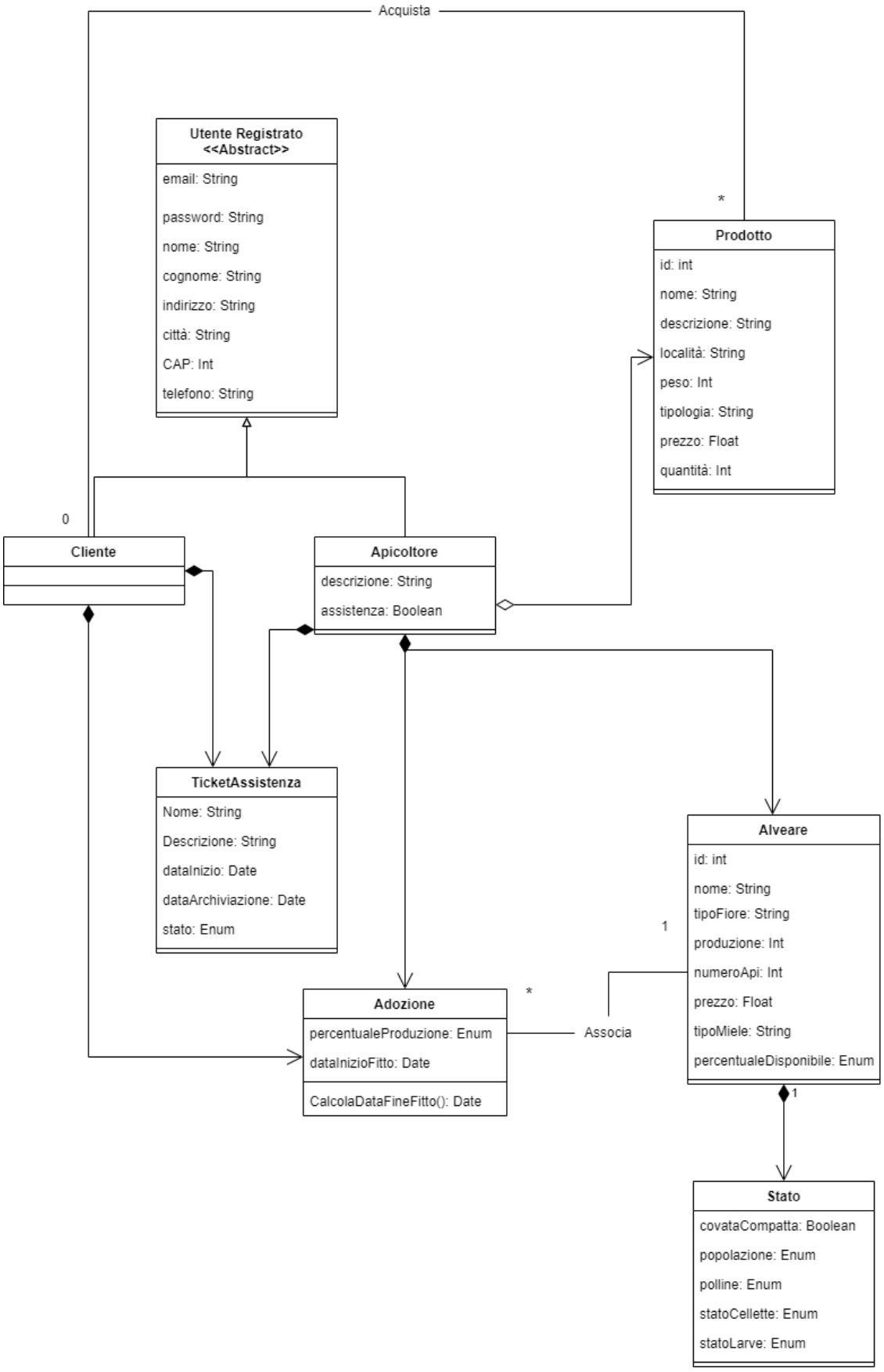
- **Transazioni**, in MySQL è possibile effettuare delle operazioni atomiche tramite l'utilizzo di transazioni; esse sono strutturate in modo che il codice al loro interna venga eseguito o per intero o per nulla, senza mezzi termini, garantendo così una più efficiente consistenza dei dati.

- **Affidabilità dei dati**, i DBMS permettono di effettuare dei backup, ovvero delle copie dei dati attualmente presenti nel database, cosicché in caso di malfunzionamenti hardware o software sarà possibile ripristinare lo stato della base di dati.

L'utilizzo di MySQL è stato preferito rispetto ad altri DBMS data la sua facilità di utilizzo, di implementazione e l'esperienza che ogni membro del team ha avuto con esso in passato.

Per quanto riguarda la definizione delle entità e delle relazioni del database, si può fare riferimento al Class Diagram già mostrato nel documento di analisi dei requisiti, che viene riportato anche di seguito. La scelta di basarsi sul sopracitato artefatto è data dal fatto che, per l'estrazione delle sue componenti, sono state usate le stesse euristiche che sarebbero state usate per la definizione di un modello ER, risultando quindi in uno spreco di effort. Dallo stesso Class Diagram si possono evincere i tipi degli attributi delle entità, motivo per il quale è stata omessa la definizione del dizionario dei dati.

CLASS DIAGRAM



3.5. Sicurezza ed Accessi

Di seguito viene mostrata la matrice degli accessi, con l'obiettivo di specificare, per ogni attore, a quali servizi è permesso l'accesso per ogni sottosistema.

Attori / Oggetti	Cliente	Apicoltore	Ospite
Gestione Utente	Login Logout Visualizza Area Utente Modifica Area Utente Eliminazione Account	Login Logout Visualizza Area Utente Modifica Area Utente Eliminazione Account	Registrazione Apicoltore Registrazione Cliente
Gestione Vendita	Visualizza Catalogo Visualizza Prodotto Visualizza Info Prodotto Acquista Prodotto Filtra Catalogo	Inserimento Prodotto Modifica Prodotto Cancellazione Prodotto Visualizza Prodotti In Vendita	Visualizza Catalogo Visualizza Prodotto Filtra Catalogo
Gestione Adozioni	Adotta alveare Visualizza Catalogo Alveari Visualizza Info Alveare Visualizza Stato Alveare	Messa in Adozione Alveare Aggiornare Stato Alveare Aggiorna Info Alveare Rimozione Alveare	Visualizza Catalogo Alveari Visualizza Info Alveari
Gestione Assistenza Utente	Visualizza Ticket Creato Visualizza Ticket archiviati Creazione Ticket Richiedi Eliminazione Ticket Visualizza Lista Assistenti	Creazione Area Assistenza Visualizza Ticket Archiviazione Ticket Eliminazione Ticket	

3.6. Controllo Globale del Software

Il sistema BeeHave fornisce funzionalità che richiedono una continua interazione da parte dell'utente col sistema. Per questo motivo abbiamo adottato un controllo del flusso globale del sistema di tipo event-driven. Il flusso globale di tipo event-driven permette di rappresentare un sistema interattivo per cui ogni funzionalità viene avviata in seguito ad un comando impartito dall'utente tramite l'uso di un'interfaccia grafica.

Quando un utente vuole accedere ed utilizzare una funzionalità del sistema può farlo tramite l'interfaccia grafica la quale selezionerà il controllo corrispondente. L'azione scatenerà un evento, il quale verrà gestito dal suo handler, che a sua volta indirizzerà il controllo del flusso di eventi al sottosistema che si occupa della logica di controllo e gestore del controllo che poi si rivolge ai servizi per la logica applicativa

3.7. Condizioni limite

Nella presente sezione vengono mostrati i casi d'uso eccezionali relativi ad avvio, spegnimento ed eventuale errore del sistema.

Identificativo *UCBC_01*	Avvio del sistema	**Data**	02/12/2022
		Vers.	1.01.000
		Autore	Luigi Bacco
Descrizione	Lo UC illustra la funzionalità di avvio del sistema.		
Attore Principale	Developer		
Attori Secondari	**NA**		
Entry Condition	Il sistema è spento AND Il developer accede al server AND Il developer invia il segnale di avvio del sistema		
Exit Condition *On Success*	Il sistema è correttamente avviato		
Exit Condition *On Failure*	Il sistema non è avviato		
Main Scenario - Flusso di eventi principale			

1	**Sistema**		Il server si avvia
2	**Sistema**		Il server si collega al Database
3	**Sistema**		Il server effettua la connessione alla rete internet
4	**Sistema**		Il sistema si mette in attesa di ricevere richieste
1° Scenario - Flusso di eventi alternativo: Connessione al Database non riuscita			
1	**Sistema**		Invia segnale di mancato collegamento con il DB
2	**Developer**	Controlla il collegamento con il DB	
3	**Developer**	Effettua il collegamento con il DB	
2° Scenario - Flusso di eventi alternativo: Connessione ad Internet non riuscita			
1	**Sistema**		Invia segnale di mancata connessione ad Internet
2	**Developer**	Controlla connessione ad Internet	
3	**Developer**	Effettua collegamento ad Internet	

Identificativo *UCBC_02*	Spegnimento del Sistema	**Data**	02/12/2022
		Vers.	1.00.000
		Autore	Irene Gaita, Gianluca Ronga
Descrizione	Lo UC illustra la funzionalità di spegnimento del sistema		
Attore Principale	Developer		
Attori Secondari	**NA**		
Entry Condition	Il Developer accede al Server. AND Il Sistema è stato precedentemente avviato. AND Il Sistema non è stato ancora spento.		
Exit Condition *On Success*	Il sistema viene spento correttamente		
Exit Condition *On Failure*	Il sistema non viene spento		
Main Scenario - Flusso di eventi principale			
1	Developer	Invia un segnale di spegnimento al Sistema	
2	Sistema		Controlla che non ci siano connessioni attive con l'esterno, se ciò è verificato, termina l'esecuzione del sistema.
1° Scenario - Flusso di eventi alternativo: Ci sono connessioni ancora attive (si attende uno slot di tempo prima dello spegnimento per risolvere connessioni attive verso l'esterno)			

1	**Sistema**		Notifica al Developer che ci sono ancora connessioni attive verso l'esterno.
2	**Sistema**		Attende uno slot di tempo per rispondere a eventuali richieste dall'esterno, non generando nuove connessioni se non allo scopo di rispondere a richieste già in corso.
3	**Sistema**		Controlla che non ci siano connessioni ancora aperte da o verso l'esterno. Se non ci sono, termina l'esecuzione del sistema.
4	**Sistema**		Notifica il Developer dell'avvenuto spegnimento del sistema.
2° Scenario - Flusso di eventi alternativo: Ci sono connessioni ancora aperte (interrompe all'istante le connessioni verso l'esterno)			
1	**Sistema**		Interrompe le connessioni verso l'esterno.
2	**Sistema**		Notifica il Developer dell'avvenuto spegnimento del sistema e del numero di connessioni interrotte.

Identificativo *UCBC_03*	Errore dati persistenti del Sistema	**Data**	02/12/2022
		Vers.	1.00.000
		Autore	Thomas De Palma, Maria Lucia Fede

Descrizione	Lo UC descrive il comportamento del Sistema in caso in cui è impossibile accedere ai dati persistenti.
Attore Principale	Developer
Attori Secondari	**NA**
Entry Condition	Si verifica un errore nell'accesso ai dati persistenti del Sistema OR I dati persistenti sono corrotti
Exit Condition *On Success*	Il sistema riprende il normale funzionamento.
Exit Condition *On Failure*	Il sistema non riprende il normale funzionamento.

Main Scenario - Flusso di eventi principale			
1	**Sistema**		Notifica lo sviluppatore con un messaggio di errore riguardo l'impossibilità di accedere ai dati persistenti.
2	**Sistema**		Smette di processare ulteriori richieste.
3	**Developer**	Include (UCBC_02)	
4	**Developer**	Risolve errore critico e ripristina accessibilità ai dati	
5	**Developer**	Include (UCBC_01)	

4. Servizi dei sottosistemi

In questa sezione vengono elencati i servizi dei sottosistemi precedentemente menzionati.

SOTTOSISTEMA GESTIONE UTENTE

Servizi	Descrizione
Registrazione Cliente	Questo servizio permette la registrazione di un cliente.
Registrazione Apicoltore	Questo servizio permette la registrazione di un apicoltore.
Login	Questo servizio permette l'accesso al sistema ad un account precedentemente registrato.
Logout	Questo servizio permette la disconnessione dal sistema ad un account precedentemente acceduto.
Elimina Account	Questo servizio permette l'eliminazione del proprio account.
Modifica Dati Account	Questo servizio permette la modifica dei propri dati personali.

SOTTOSISTEMA GESTIONE ADOZIONI

Servizi	Descrizione
Messa in adozione alveare	Questo servizio permette la messa in adozione di un alveare.
Adotta Alveare	Questo servizio permette ad un cliente di adottare un alveare.
Lista alveari in adozione	Questo servizio permette la restituzione della lista degli alveari attualmente disponibili all'adozione.
Lista alveari adottati	Questo servizio permette la restituzione della lista degli alveari, attualmente adottati, da parte di un cliente.
Informazioni alveare	Questo servizio permette la restituzione delle informazioni concerni gli alveari in adozione.
Rimozione disponibilità alveare	Questo servizio permette ad un apicoltore di rimuovere un alveare dal catalogo delle arnie in adozione.

Inserimento opzioni adozione	Questo servizio permette al cliente, in procinto di adottare un alveare, la selezione delle opzioni di adozione.
Stato alveare	Questo servizio permette al cliente di restituire lo stato di un alveare adottato.
Aggiorna stato alveare	Questo servizio permette all'apicoltore di aggiornare lo stato di un suo alveare.

SOTTOSISTEMA GESTIONE VENDITA

Servizi	Descrizione
Vendita Prodotto	Questo servizio permette ad un apicoltore di mettere in vendita il proprio prodotto.
Acquisto Prodotto	Questo servizio consente ad un cliente di acquistare un prodotto dal catalogo.
Rimozione Prodotto	Questo servizio consente ad un apicoltore di rimuovere un prodotto dal catalogo.

Modifica informazioni prodotto	Questo servizio permette all'apicoltore di modificare le informazioni dei propri prodotti in vendita.
Applicazione filtri al catalogo	Questo servizio consente di selezionare i prodotti del catalogo secondo l'applicazione di alcuni filtri di ricerca.
Lista prodotti in vendita	Questo servizio permette all'apicoltore la restituzione dei propri prodotti in vendita.

SOTTOSISTEMA GESTIONE ASSISTENZA UTENTE

Servizi	Descrizione
Chiusura ticket	Questo servizio permette ad un apicoltore di chiudere una richiesta di assistenza
Creazione area assistenza	Questo servizio permette all'apicoltore di creare la propria area assistenza
Richiesta chiusura ticket	Questo servizio permette al cliente di richiedere a chiusura di un ticket

Archiviazione ticket	Questo servizio permette all'apicoltore di archiviare un ticket di assistenza
Creazione ticket	Questo servizio permette al cliente di creare un ticket di assistenza

5. Glossario

Terminologia	Definizione
Triade CIA	I requisiti fondamentali per garantire la sicurezza delle informazioni, è composto da Confidenzialità, Integrità e Disponibilità.
Three-Tier	Architettura software composta da tre strati disaccoppiati. Gli strati sono: Presentazione, Logica, Persistenza.
DBMS	Un Database Management System è un sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database

MySQL	DBMS relazionale open source, utilizza SQL come linguaggio di query.
Design Pattern	Template di soluzioni da applicare per la risoluzione di un problema che si può presentare in diverse situazioni durante le fasi di progettazione e sviluppo del software.
Model View Control	Il design pattern MVC specifica che un'applicazione è costituita da un modello di dati, informazioni di presentazione e informazioni di controllo. Il modello richiede che ognuno di questi sia separato in diversi oggetti.
Framework	Architettura logica di supporto sulla quale un software può essere progettato e realizzato fornendo un'infrastruttura generale lasciando al programmatore il contenuto vero e proprio dell'applicazione.
Flask	Framework basato su Python utilizzato per sviluppare applicazione web-based.
Modello ER	Modello utilizzato per la rappresentazione grafica dei dati. In questo modello vengono definite entità e relazioni tra le entità. Utilizzato per progettare schemi di basi di dati.

Developer	Sviluppatore del sistema, incaricato anche di effettuare manutenzione e gestire casi di avvio, spegnimento ed errore del sistema.
HTML, CSS, JS, Python	Tecnologie utilizzate per lo sviluppo del sistema.
HTTPS	Protocollo di comunicazione sicura per contenuti di informazione sul web.