

SDD System Design Document

NewDM

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 0.8 |
| Data | 25/11/2020 |
| Destinatario | Prof.ssa F. Ferrucci |
| Presentato da | Cirillo Franco  Cirillo Luigi  Fusco Ciro  Aiello Vincenzo |
| Approvato da |  |

SDD System Design Document

**NewDM**

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 0.2 |
| Data | 25/11/2020 |
| Destinatario | Prof.ssa F. Ferrucci |
| Presentato da | Cirillo Franco  Cirillo Luigi  Fusco Ciro  Aiello Vincenzo |
| Approvato da |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Cambiamenti | Autori |
| 28/11/2020 | Vr0.1 | Obiettivi del sistema, design goals, definizioni, riferimenti, panoramica, architettura sistemi simili, architettura sistema proposto, panoramica, decomposizione in sottosistemi | [tutti] |
| 29/11/2020 | Vr0.2 | Mapping, gestione dati persistenti, controllo degli accessi, controllo del flusso | [tutti] |

RevisionHistory

Sommario

1. [1. Introduzione 4](#_Toc57579731)

[1.1 Obiettivi del sistema 4](#_Toc57579732)

[1.2 Design Goals & Trade-offs 4](#_Toc57579733)

[1.3 Definizioni, acronimi e abbreviazioni 6](#_Toc57579734)

[1.4 Riferimenti 6](#_Toc57579735)

[1.6 Panoramica 6](#_Toc57579736)

1. [2. Architettura di Sistemi simili 6](#_Toc57579737)
2. [3. Architettura del Sistema proposto 7](#_Toc57579738)

[3.1 Panoramica 7](#_Toc57579739)

[3.2 Decomposizione in sottosistemi 7](#_Toc57579740)

[3.2.1 Diagramma di deployment 8](#_Toc57579741)

[3.3 Mapping hardware/software 8](#_Toc57579742)

[3.4 Gestione dati persistenti 9](#_Toc57579743)

[3.5 Controllo degli accessi e sicurezza 9](#_Toc57579744)

[3.6 Controllo flusso globale del sistema 10](#_Toc57579745)

[3.7 Condizione limite 10](#_Toc57579746)

[3.7.1 Start-up 10](#_Toc57579747)

[3.7.2 Terminazione 11](#_Toc57579748)

[3.7.3 Fallimento 12](#_Toc57579749)

1. [4. Servizi dei sottosistemi 12](#_Toc57579750)
2. [Glossario 14](#_Toc57579751)

# 1. Introduzione

## 1.1 Obiettivi del sistema

Lo scopo di ogni grande azienda che si rispetti è garantire ai propri clienti sicurezza e affidabilità. Per far questo ci affidiamo ad un sistema che ci permetta di gestire sempre in maniera smart tutte le condizioni per soddisfare i nostri clienti, e semplificare la vita dei nostri dipendenti. Nasce così NewDM!

Il sistema proposto parte dall’idea di poter fornire una piattaforma che sia in grado di gestire al meglio le varie interazioni tra i dipendenti e i clienti. Si vuole quindi realizzare una piattaforma desktop che consentirà lo svolgimento delle attività in maniera più agevole possibile per il bacino di utenza a disposizione.

NewDM avrà necessita di gestire dati persistenti: la scelta, per motivi di efficienza, ricade in un database relazionale che permetterà di tener traccia di tutte le informazioni necessarie di interesse dell’azienda. Da tale database attingerà un applicazione desktop deputata alla gestione delle interazioni con l’utente e alla manipolazione dei suddetti dati.

La piattaforma avrà un controllo per gli accessi alle varie aree (commesso, magazziniere, addetto assistenza) autenticando i propri utenti con uno username e password.

## 1.2 Design Goals & Trade-offs

Illustriamo nella seguente tabella gli obbiettivi di design per il sistema e le relative priorità (a numeri più bassi corrispondono priorità più elevate). Per ogni obiettivo riportiamo anche l’origine, facendo riferimento, in particolare, all’identificativo del requisito non funzionale ad esso associato.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Priorità** | **ID** | **Descrizione** | **Categoria** | **Origine** |
| 3 | DG\_1 | *Tempi di risposta*: Il sistema dovrà garantire che la scannerizzazione e la visualizzazione del prezzo del prodotto devono avvenire in al più sec. 2. Il sistema dovrà garantire che il calcolo totale e l’emissione dello scontrino devono avvenire in al più sec. 3. | Performance | RNF-P1  RNF-P2 |
| 1 | DG\_2 | *Robustezza*: Vogliamo proporre un sistema che abbia la capacità di sopravvivere ad input non validi. Pertanto, il sistema deve garantire il filtraggio dei dati inconsistenti o errati inseriti dall’utente, invitandolo a reinserirli. | Dependability | RNF-A1 |
| 3 | DG\_3 | *Disponibilità*: Il sistema deve essere sempre disponibile durante l’orario di lavoro (6am to 10pm). | Dependability | RNF-P4 |
| 1 | DG\_4 | *Tolleranza ai fault*: Il sistema dovrà evitare il verificarsi di più di un fault al giorno, in modo da garantire l’efficienza del servizio, ma soprattutto dovrà evitare ogni perdita di dati registrati. | Dependability | RNF-A2 |
| 1 | DG\_5 | *Sicurezza*: Il sistema dovrà fornire un metodo di autenticazione sicuro in modo che i dati siano protetti da accessi fraudolenti. | Dependability | RNF-A3 |
| 2 | DG\_6 | *Costi di sviluppo*: Lo sviluppo del prodotto richiederà costi ridotti sia in termini di risorse umane (per cui è fissato un tetto di 75 ore-lavoro), sia in termini economici (per cui si punta a ricorrere a soluzioni off-the-shelf open source) | Costo | Top management |
| 1 | DG\_7 | *Modificabilità*: Il sistema prodotto deve offrire una buona modificabilità, in modo da poter modificare le funzionalità esistenti. | Manutenibilità | RNF-S2 |
| 1 | DG\_8 | *Estensibilità*: Il sistema prodotto deve offrire una buona estensibilità, in modo da poter aggiungere nuove funzionalità al sistema | Manutenibilità | RNF-S3 |
| 2 | DG\_9 | *Leggibilità*: Il codice prodotto dev’essere semplice da comprendere. | Manutenibilità | RNF-S1 |
| 2 | DG\_10 | *Usabilità*: Il sistema deve essere facile da apprendere ed intuitivo da utilizzare senza necessariamente consultare la documentazione. | End User | RNF-U1  RNF-U2 |

*Tempo di rilascio vs Funzionalità*

Sebbene i tempi siano piuttosto proibitivi, preferiamo consegnare con leggero ritardo un prodotto che faccia ciò che promette piuttosto che un prodotto che non possa essere utilizzato a causa della mancanza di funzionalità.

*Prestazioni vs Costi*

Considerato il budget ridotto a disposizione, si preferisce rientrare nei costi dedicando un numero ridotto di ore-lavoro alla massimizzazione delle prestazioni.

*Prestazioni vs Affidabilità*

I dati gestiti dal sistema sono piuttosto sensibili, pertanto preferiamo garantire un maggior controllo di input e consistenza a scapito dei tempi di risposta.

## 1.3 Definizioni, acronimi e abbreviazioni

* dbms: database management system
* DG: Design goal

## 1.4 Riferimenti

* Requisiti funzionali: Sezione 3.2 del RAD
* Requisiti non funzionali: Sezione 3.3 del RAD
* Libro:

-- Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition

Autori:

-- Bernd Bruegge & Allen H. Dutoit

## 1.6 Panoramica

Nel documento verranno affrontati l’analisi delle architetture di sistemi simili, la decomposizione in sottosistemi del sistema proposto con la definizione della strategia di deploy e le condizioni limite. Verranno quindi definiti i servizi esposti da ciascun sottosistema.

# 2. Architettura di Sistemi simili

Nel nostro caso non è presente un’architettura software già esistente, pertanto abbiamo analizzato le architetture di sistemi simili.

Il primo software gestionale preso in considerazione è "EuroAzienda Supermercati". Esso è composto da due moduli software: Parte amministrativa, per la gestione del magazzino e per l’analisi e modifica prezzi. Punto cassa per la gestione dello scarico magazzino e per l’emissione dello scontrino.

Il secondo software analizzato è “Software Foodmanager”. È un software gestionale multipiattaforma in versione cloud html5 e in versione desktop. Permette la gestione dei permessi di ciascun operatore e prevede due moduli: il primo per la gestione del magazzino e il secondo per la gestione lato cassa.

Dall’analisi di questi due software si è evinto che alla base di tutte le piattaforme è presente un database per la memorizzazione e gestione dei prodotti presenti in magazzino. Quest’ultimo è interrogato da applicazioni desktop/web progettate con diverse tecnologie. Deduciamo, quindi, che ognuna delle suddette piattaforme sia stata sviluppata secondo un’architettura a 3 strati.

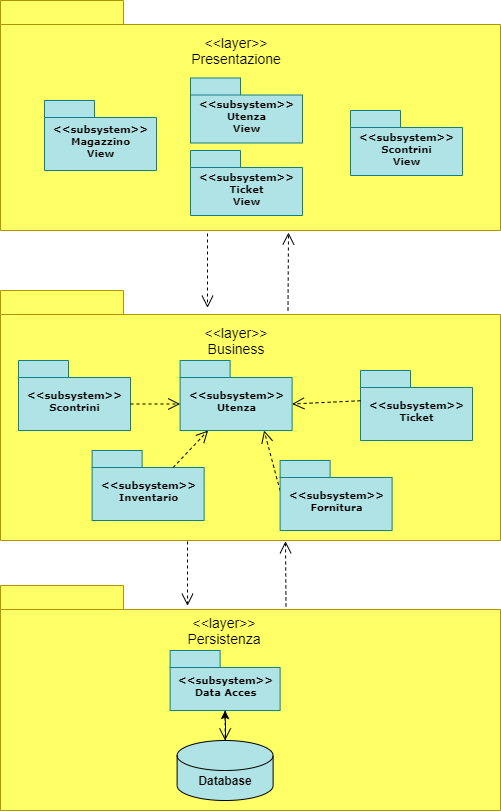
Nessuno dei software analizzati comprende la gestione dell’area per l’assistenza clienti. Inoltre, soltanto il secondo analizzato, prevede un’autenticazione che permette di rendere disponibili, determinate funzionalità, solamente ad utenti autorizzati.

# 3. Architettura del Sistema proposto

## 3.1 Panoramica

NewDM è una Piattaforma desktop che fornisce un’interfaccia per le sue funzionalità tra cui: la gestione dell’utenza, dei ticket, dell’emissione scontrini e del magazzino. Essa, inoltre, ricorre all’utilizzo di un database relazionale per il salvataggio dei dati persistenti.

## 3.2 Decomposizione in sottosistemi



Il sistema è suddiviso in 3 livelli logici: presentazione, business e persistenza che si occupano rispettivamente di presentazione delle informazioni all’utente, definizione della logica applicativa e gestione dei dati persistenti.

Il livello di presentazione è composto da cinque sottosistemi:

* Utenza View: definisce l’interfaccia utente per la parte relativa all’autenticazione
* ScontriniView: definisce l’interfaccia utente relativa al processo di emissione di scontrini
* TicketView: definisce l’interfaccia utente relativa alla gestione dei ticket
* MagazzinoView: definisce l’interfaccia utente per la gestione dei prodotti presenti in magazzino

Il livello di business è composto da cinque sottosistemi

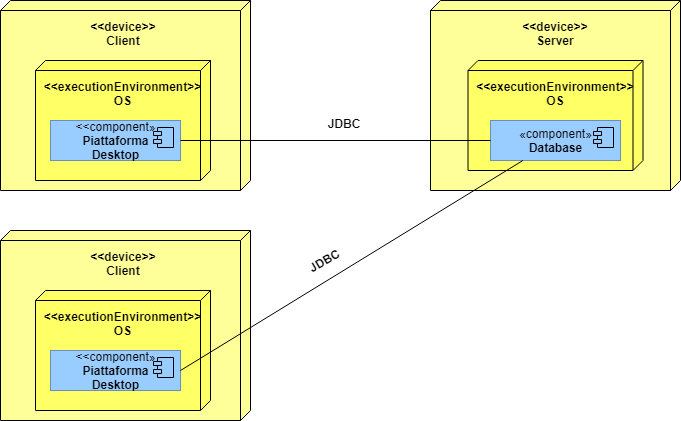
* Utenza: definisce l’utente (cassiere, magazziniere, addetto assistenza) ed offre i relativi servizi di autenticazione.
* Scontrini: modella tutto ciò che riguarda il processo di acquisto dei prodotti ed emissione scontrino
* Ticket: modella il processo di gestione ticket e tutte le sue operazioni
* Inventario: modella i prodotti nel magazzino e le relative operazioni
* Fornitura: modella il processo di richiesta prodotto
* Data Access: si occupa del reperimento e salvataggio delle informazioni manipolate dal sistema.

La suddivisione in sottosistemi è stata quindi realizzata tramite una strutturazione 3-tier, al fine di disaccoppiare l’interfaccia dalla logica di business dell’applicazione. Inoltre, è stata fatta questa scelta in modo da rendere possibile la modifica o la sostituzione di ciascuno dei tre moduli, con il minor impatto possibile, conferendo così scalabilità e manutenibilità all’ applicazione, in linea con i Design Goals, in particolare DG\_7 e DG\_8.

### 3.2.1 Diagramma di deployment

NewDM si compone di due elementi essenziali: una piattaforma desktop e un database MySQL eseguito su un server.

L’architettura scelta prevede l’interazione della piattaforma desktop con il database: queste due componenti sono installate nella stessa rete locale e si scambiano informazioni tramite un driver di protocollo di rete, ossia, JDBC. La piattaforma desktop è installata su più client in modo da favorire l’utilizzo in contemporaneo. È stato scelto di utilizzare un database MySQL per facilitare la gestione delle tabelle relazionali e per utilizzare delle query più veloci ed efficienti. Viene utilizzato un server locale per velocizzare l’interazione tra client e database. Inoltre, è stato scelto il driver JDBC in quanto favorisce e semplifica l'accesso e la gestione della persistenza dei dati sulle basi di dati da qualsiasi programma scritto con il linguaggio di programmazione Java.



## 3.3 Mapping hardware/software

NewDM si compone di due componenti principali:

* Piattaforma desktop, cui saranno allocati i layer di presentazione e business oltre al sottosistema di data access
* Database, realizzante il layer di persistenza

Il sistema necessita di diverse macchine con sistema operativo Windows al fine di garantire l’operabilità della suddetta piattaforma desktop.

Il sistema inoltre necessita di una macchina in grado di supportare il DBMS MySQL per garantire l’operabilità del database con cui le diverse piattaforme desktop si interfacciano.

La comunicazione avverrà tramite protocollo jdbc. Per la connessione alla rete locale verranno utilizzati cavi ethernet.

Le scelte rispecchiano i Design Goals definiti relativi alla Dependability, in particolare DG\_3, DG\_4 e DG\_5, si preferisce infatti mantenere un server in rete locale in grado di fornire servizi di DBMS per avere una maggiore sicurezza sui dati sensibili, una più sicura disponibilità e limitare il più possibile i fault relativi a cadute di connessione, a discapito dei costi di sviluppo.

## 3.4 Gestione dati persistenti

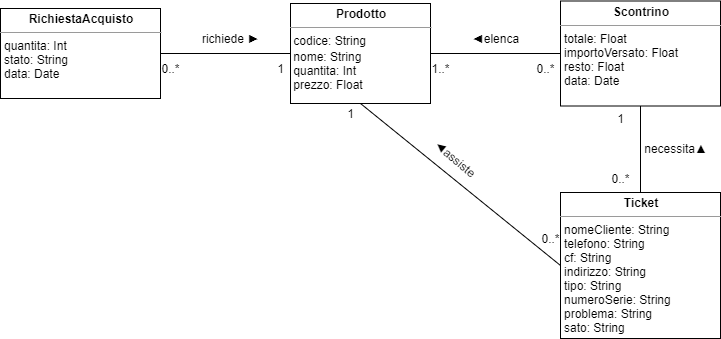
Per la gestione dei dati persistenti, NewDM si affida ad un database relazionale gestito tramite MySQL e del sottosistema Data Access per accedere ai dati.

La scelta fatta è in linea con i Design Goals, in particolare la robustezza, infatti si preferisce l’utilizzo di un DBMS al fine di garantire un accesso affidabile ai dati e di consentire più facilmente letture e scritture multiple, data la molteplicità di device che possono accedere alle informazioni.

Ulteriori vantaggi derivano dalla potenza del DBMS, che permette l’utilizzo di un linguaggio di manipolazione dei dati, consentendo così un accesso ad un livello più raffinato di dettaglio, necessario per le funzioni del nostro sistema.

Si è tenuto conto che queste scelte andranno a ridurre l’efficienza e la velocità, quindi le prestazioni del sistema, ma sono state fatte rispecchiando i trade off.

La struttura dei dati memorizzati segue lo schema in figura:



## 3.5 Controllo degli accessi e sicurezza

Il controllo di accesso è garantito tramite l’utilizzo di username e password, in modo da evitare l’accesso a dati sensibili e prevenire le modifiche non autorizzate. Inoltre, almeno nella prima versione non saranno implementati metodi di recupero o modifica password.

Il salvataggio delle password avverrà con cifratura. In particolare, verrà usato l’algoritmo di hashing SHA-1.

La scelta è in linea con i Design Goals, in quanto l’autenticazione richiesta offre un primo livello di sicurezza, inoltre, un controllo sulla tipologia di utente è eseguito in modo che ognuno si interfacci solamente con le proprie mansioni lavorative. Tuttavia, si tiene conto del fatto che tutte queste operazioni andranno a diminuire le prestazioni, in termini di velocità.

Le operazioni che gli utenti della piattaforma desktop possono effettuare sugli oggetti sono riportati nella tabella che segue:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Oggetti  Attore | Utenza | Prodotto | Scontrino | Ticket | Richiesta  fornitura |
| Cassiere | Autenticazione  Logout | Lettura Prodotto | Creazione Scontrino |  |  |
| Magazziniere | Autenticazione  Logout | Creazione e modifica Prodotto |  |  | Creazione  Richiesta  fornitura |
| Addetto Assistenza | Autenticazione  Logout |  |  | Crea nuovo ticket per prodotti in garanzia |  |

## 3.6 Controllo flusso globale del sistema

Il sistema adotta un controllo del flusso globale del tipo Event-Driven, dove le varie interfacce grafiche verranno utilizzate per navigare tutte le funzionalità del sistema. Il Server su cui è installato il DBMS consentirà a tutti i Client l’accesso ai dati persistenti, gestendo in maniera autonoma la concorrenza tramite l’utilizzo di lock sulle tabelle. Così facendo si rispetta il vincolo di atomicità e consistenza delle operazioni dei diversi Client, e per tanto non soffriranno di problemi legati alla lettura di informazioni non ancora completamente aggiornate. Le computazioni che invece non richiedono il salvataggio su Database verranno svolte localmente al Client che le invoca e per tanto saranno indipendenti.

Nonostante il Design Goal di performance richieda un tempo di risposta molto basso, si preferisce evitare l’accesso a informazioni incoerenti e/o non consistenti e puntare ad una affidabilità maggiore del sistema sebbene risulti evidente che gestire una singola richiesta alla volta comporta una perdita di tempo rispetto all’accesso parallelo.

## 3.7 Condizione limite



### 3.7.1 Start-up

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Identificativo  *UC\_GP1* | | | | | | *Start-up sistema* | *Data* | | *29/11/2020* |
| *Vers.* | *0.01.003* | |
| *Autore* | *Vincenzo Aiello* | |
| Descrizione | | | | | | *Lo use-case definisce la funzionalità di avvio del Sistema per*  *Il gestore della piattaforma* | | | |
| Attore Principale | | | | | | Gestore piattaforma  È interessato ad avviare il sistema | | | |
| Attori secondari | | | | | | N/A | | | |
| Entry Condition | | | | | | Il gestore ha accesso alla macchina su cui è installato  Il sistema | | | |
| Exit condition  On success | | | | | | Il Sistema è avviato correttamente | | | |
| Exit condition  On failure | | | | | | Il Sistema non è avviato | | | |
| Rilevanza/User Priority | | | | | | Elevata | | | |
| Frequenza stimata | | | | | | 1/giorno | | | |
| Extension point | | | | | | NA | | | |
| Generalization of | | | | | | NA | | | |
| Flusso di Eventi Principale/Main Scenario | | | | | | | | | |
| 1 | Gestore sistema: | | | Lancia il servizio MySQL su server Windows tramite l’apposito comando | | | | | |
| 2 | Sistema: | | | Comunica al gestore che lo startup si è concluso con successo | | | | | |
| 3 | Gestore sistema | | | Avvia l’applicazione desktop NewDM  su ogni client. | | | | | |
| 4 | Sistema: | | | Lancia l’applicazione desktop | | | | | |
|  | | | | | |  | | | |
| Scenario/Flusso di eventi di ERRORE: non è possibile avviare il sistema | | | | | | | | | |
| 2.1 | | Sistema: | | | | Mostra all’impiegato un messaggio che ne specifica il motivo | | | |
|  | |  | | | |  | | | |
| Scenario/Flusso di eventi di ERRORE: non è possibile avviare l’applicazione desktop | | | | | | | | | |
| 4.1 | | | Sistema: | | Mostra all’impiegato un messaggio che ne specifica il motivo | | | | |

### 

### 3.7.2 Terminazione

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Identificativo  *UC\_GP2* | | | | *Shutdown sistema* | *Data* | | *29/11/2020* |
| *Vers.* | *0.01.003* | |
| *Autore* | *Vincenzo Aiello* | |
| Descrizione | | | | *Lo use-case definisce la funzionalità di terminazione del sistema per*  *Il gestore della piattaforma* | | | |
| Attore Principale | | | | Gestore piattaforma  È interessato a terminare il sistema | | | |
| Attori secondari | | | | N/A | | | |
| Entry Condition | | | | Il gestore ha accesso alla macchina su cui  il sistema è in esecuzione | | | |
| Exit condition  On success | | | | Il Sistema è terminato correttamente | | | |
| Exit condition  On failure | | | | Il Sistema resta in esecuzione | | | |
| Rilevanza/User Priority | | | | Elevata | | | |
| Frequenza stimata | | | | 1/giorno | | | |
| Extension point | | | | NA | | | |
| Generalization of | | | | NA | | | |
| Flusso di Eventi Principale/Main Scenario | | | | | | | |
| 1 | Gestore sistema: | | Termina l’applicazione desktop su ogni client | | | | |
| 2 | Gestore sistema | | Termina server MySQL | | | | |
| 3 | Sistema: | | Comunica al gestore che lo shutdown si è concluso con successo | | | | |
|  | | | |  | | | |
| Scenario/Flusso di eventi di ERRORE: non è possibile terminare il sistema | | | | | | | |
| 3.1 | | Sistema: | | Mostra all’impiegato un messaggio che ne specifica il motivo | | | |
|  | |  | |  | | | |
| Scenario/Flusso di eventi di ERRORE: non è possibile terminare l’applicazione desktop | | | | | | | |
| 1.1 | | Sistema: | | Mostra all’impiegato un messaggio che ne specifica il motivo | | | |

### 3.7.3 Fallimento

NewDM può incorrere in diversi casi di fallimento, riguardanti sia l’hardware che il software:

* Fallimenti Hardware
  + *Crash del disco su cui i dati persistenti sono salvati*: il sistema non prevede alcuna strategia di backup e ripristino dei dati
* Fallimenti nell’ambiente di esecuzione
  + *Interruzione della fornitura elettrica al server*: il sistema non prevede alcuna strategia che ne garantisca l’operabilità in questo tipo di condizione
* Fallimenti Software
  + *Impossibilità di stabilire una connessione col database*: il sistema mostra all’utente una schermata che riporta il rilevamento di un errore interno

# 4. Servizi dei sottosistemi

|  |  |
| --- | --- |
| Data Access | |
| Servizio | Descrizione |
| Caricamento Utente | Restituisce l’Utente dal DB i cui campi username e password coincidono con i valori dati dal chiamante |
| Caricamento Prodotto | Restituisce tutti i campi del Prodotto presi dal DB |
| Inserimento Prodotto | Inserisce un nuovo prodotto nel DB |
| Aggiornamento Prodotto | Aggiorna le informazioni del prodotto sul DB |
| Caricamento Ticket | Restituisce tutti i campi del Ticket presi dal DB |
| Inserimento Ticket | Inserisce un nuovo Ticket nel DB |
| Aggiornamento Ticket | Aggiorna le informazioni del Ticket nel DB |
| Inserimento Cliente | Inserisce un cliente nel DB |
| Caricamento Cliente | Restituisce tutti i campi del Cliente preso da DB |
| Inserimento ProdottoAss | Inserisce un ProdottoAss nel DB |
| Caricamento ProdottoAss | Restituisce tutti i campi del ProdottoAss preso da DB |
| Inserimento Scontrino | Inserisce uno Scontrino nel DB |
| Caricamento Scontrino | Restituisce tutti i campi dello Scontrino dal DB |
| Inserimento Prodotto\_Scontrino | Inserisce una relazione tra uno o più Prodotto e lo Scontrino in cui è stato acquistato |
| Caricamento Prodotto\_Scontrino con Scontrino | Restituisce tutti i riferimenti ai Prodotti acquistati con lo Scontrino |
| Caricamento Prodotto\_Scontrino con Prodotto | Restituisce tutti il riferimento a tutti gli Scontrini nei quali è stato acquistato il Prodotto |
| Inserimento Richiesta\_Fornitura | Inserisce una Richiesta per la fornitura di un Prodotto nel BD |

|  |  |
| --- | --- |
| Utenza | |
| Servizio | Descrizione |
| Login | Permette di autenticare un Utente tramite Username e Password |
| Logout | Permette di rimuovere dal sistema l’utente che si era autenticato in precedenza |
| Ottenimento utente autenticato | Restituisce l’Utente attualmente autenticato nel sistema |

|  |  |
| --- | --- |
| Scontrino | |
| Servizio | Descrizione |
| Crea Scontrino | Crea un nuovo scontrino dati tutti i prodotti da inserire |

|  |  |
| --- | --- |
| Inventario | |
| Servizio | Descrizione |
| Crea Prodotto | Crea un Prodotto dati tutti i campi da inserire |
| Aggiorna Prodotto | Aggiorna un Prodotto dati tutti i campi da modificare |

|  |  |
| --- | --- |
| Fornitura | |
| Servizio | Descrizione |
| Crea nuova Richiesta\_Fornitura | Crea una Richiesta di fornitura di un Prodotto in una determinata quantità |
| Richiesta\_Fornitura da eseguire | Restituisce un elenco di Richieste da eseguire |
| Modifica stato Richiesta \_Fornitura | Modifica lo stato di una Richiesta |
| Stima Fornitura | Restituisce una stima calcolata da una IA della quantità da acquistare di un prodotto |

|  |  |
| --- | --- |
| Ticket | |
| Servizio | Descrizione |
| Crea Ticket | Crea un nuovo Ticket con tutti i dati necessari |
| Aggiorna stato Ticket | Aggiorna lo stato del Ticket per monitorarne l’avanzamento |

# Glossario

*DBMS*: sistema software per la gestione dei dati persistenti su database

*MySQL*: specifico DBMS