SDD

SYSTEM DESIGN DOCUMENT

Sommario

1. Introduction 3
   1. Purpose Of The System 3
   2. Design goals 3
   3. Definizione, acronimi e abbreviazioni 4
   4. Riferimenti 4
2. Proposed Software Architecture 5
   1. Overview 5
   2. Subsystem Decomposition 5
   3. Hardware/Software Mapping 6
   4. Persistent Data Manager 8
   5. Access Control And Security 12
   6. Global Software Control 13
   7. Boundary Condition 13
3. Subsystem services 14

# **Introduction**

## **Purpose Of The System**

Il sistema è stato progettato al fine di migliorare la distribuzione e la pubblicizzazione dei prodotti del negozio di elettronica. Precedentemente alla realizzazione del sistema l’approccio col cliente era esclusivamente fisico, quindi limitato e disorganizzato, con la nascita del sistema questo ambito viene migliorato. Ovviamente c’è un incremento dei guadagni e di conseguenza un’ evoluzione del negozio. Tutto ciò rende la gestione delle informazioni, come l’aggiornamento dei prodotti, l’appuntamento con il personale, più veloce e versatile. A differenza delle grandi aziende che mirano ad una clientela più ampia, una piccola realtà come “Tutto Elettronica” può offrire maggiore assistenza ai singoli clienti.

## **Design Goals**

* **Criteri utenti finali**
* **Usability:** L’interfaccia grafica permetterà all’ utente di poter selezionare in maniera molto semplice le varie funzioni concesse dal sito. Per facilitare la ricerca di uno specifico prodotto nella home è presente una barra di ricerca che permette di visualizzare il prodotto semplicemente e rapidamente.

La schermata “HOME” del sito permette di:

1. visualizzare i prodotti più prenotati;
2. effettuare la Log-In;
3. andare alla sezione per la riparazione dei prodotti.

* **Criteri di performance**
* **Performance:** Il software dovrà rispondere velocemente; le risposte dovranno essere fornite in un periodo pari a circa 1 secondo. Il numero di utenti che potranno collegarsi e acquistare prodotti contemporaneamente sarà dato dalla disponibilità del Web Server utilizzato. La latenza massima di attesa, per una risposta, non dovrà superare i 30 secondi.
* **Criteri di affidabilità**
* **Reliability:** Il software dovrà essere attivo 24 ore su 24. Inoltre, deve garantire la sicurezza su tutte le operazioni effettuate sia dai gestori sia dai clienti. Nel caso in cui si verifichino dei comportamenti anomali, da parte del gestore o del cliente, verranno notificati tramite degli avvisi.
* **Criteri di manutenibilità**

**Supportability:** Il software consisterà in un sistema client-server in cui il server sarà disponibile su ogni tipo di piattaforma e il client potrà collegarsi a tale server mediante un qualsiasi browser. Il software dovrà essere suddiviso in vari moduli per permettere una più facile modifica e aggiornabilità in futuro.

## **Definizione, acronimi e abbreviazioni**

Al momento non sono presenti definizioni, acronimi e abbreviazioni.

## **Riferimenti**

Nella realizzazione del sistema ci siamo confrontati con i più grandi e-commerce sul mercato come Amazon, Ebay.

# **PROPOSED SOFTWARE ARCHITECTURE**

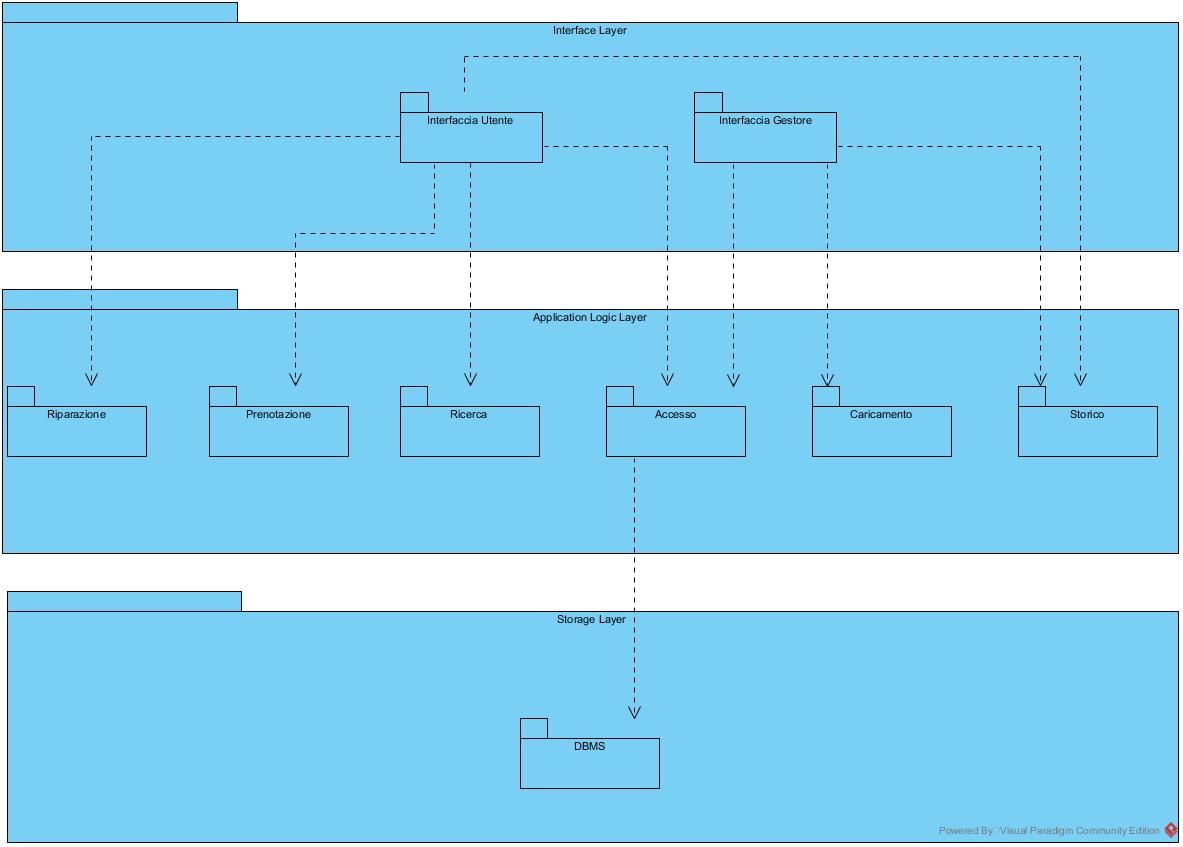
## **Overview**

Si propone un’architettura software ibrida. L’architettura software proposta è un’architettura client-server organizzata mediante una architettura Model-View-Control.

Sul lato client, dell’architettura proposta, è presente la parte View del software.

Sul lato server, invece, è presente la parte Control e la parte Model del software.

## **Subsystem Decomposition**



Utilizziamo un’architettura three-tier. I sottosistemi individuati sono:

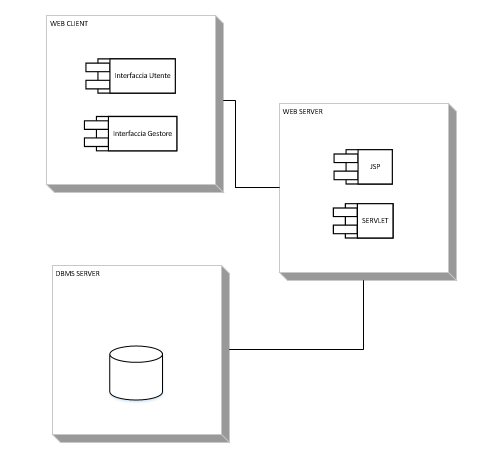
|  |  |
| --- | --- |
| **Interface Layer** |  |
| **Interfaccia utente** | Fornisce il servizio relativo all’implementazione |
| **Interfaccia Gestore** | fornisce il servizio relativo all’ implementazione dell’ interfaccia vista dai gestori. |

|  |  |
| --- | --- |
| **Application Logic Layer** |  |
| **Accesso** | 1. **Login:** servizio che permette l’ autenticazione al sistema. 2. **Registrazione:** servizio che permette la registrazione al sistema. 3. **Logout:** servizio che permette il logout dal sistema |
| **Ricerca** | Fornisce il servizio per la ricerca di un prodotto tramite il nome, la marca, il modello e il codice |
| **Riparazione** | Fornisce i servizi inerenti alla prenotazione di una riparazione di un prodotto:   1. **Prenota data:** servizio che permette al cliente di scegliere una data in cui portare il prodotto in riparazione 2. **Invia dettagli:** servizio che permette al cliente di inviare i dettagli sul prodotto malfunzionante. |
| **Prenotazione** | Fornisce i servizi inerenti alla prenotazione di uno o più prodotti:   1. **Aggiungi al carrello:** servizio che permette di aggiungere uno o più prodotti al carrello. 2. **Prenota prodotto:** servizio che permette di prenotare i prodotti inseriti nel carrello. 3. **Elimina prodotto:** servizio che permette di eliminare uno o più prodotti dal carrello. |
| **Caricamento** | Fornisce il servizio per il caricamento di un prodotto, una data o un ruolo utente da parte di un gestore |
| **Storico** | Fornisce i servizi inerenti alla visualizzazione delle operazioni effettuate:   1. **Storico Cliente:** servizio che permette ad un cliente di visualizzare lo storico inerente ai prodotti prenotati o alle riparazioni prenotate. 2. **Storico dei clienti:** servizio che permette ad un gestore di visualizzare gli storici dei prodotti acquistati fino a quel momento dagli utenti. |

|  |  |
| --- | --- |
| **Storage Layer** |  |
| **DBMS** | Fornisce il servizio che si occupa di interfacciare le richieste effettuate, dai vari sottosistemi, al database. |

## **Hardware/Software Mapping**

Il seguente diagramma di distribuzione UML illustra la mappatura hardware/software del sistema.



## **Persistent Data Manager**

Il sistema Tutto Elettronica si avvale dell’uso di un database di tipo relazionale. Questa

scelta è stata ponderata data la sicurezza offerta da un DBMS di ultima generazione,

assieme ad una maggiore affidabilità e garanzia di coerenza e facilità di gestione, nonché

dalla velocità di accesso e trasmissione dei dati.

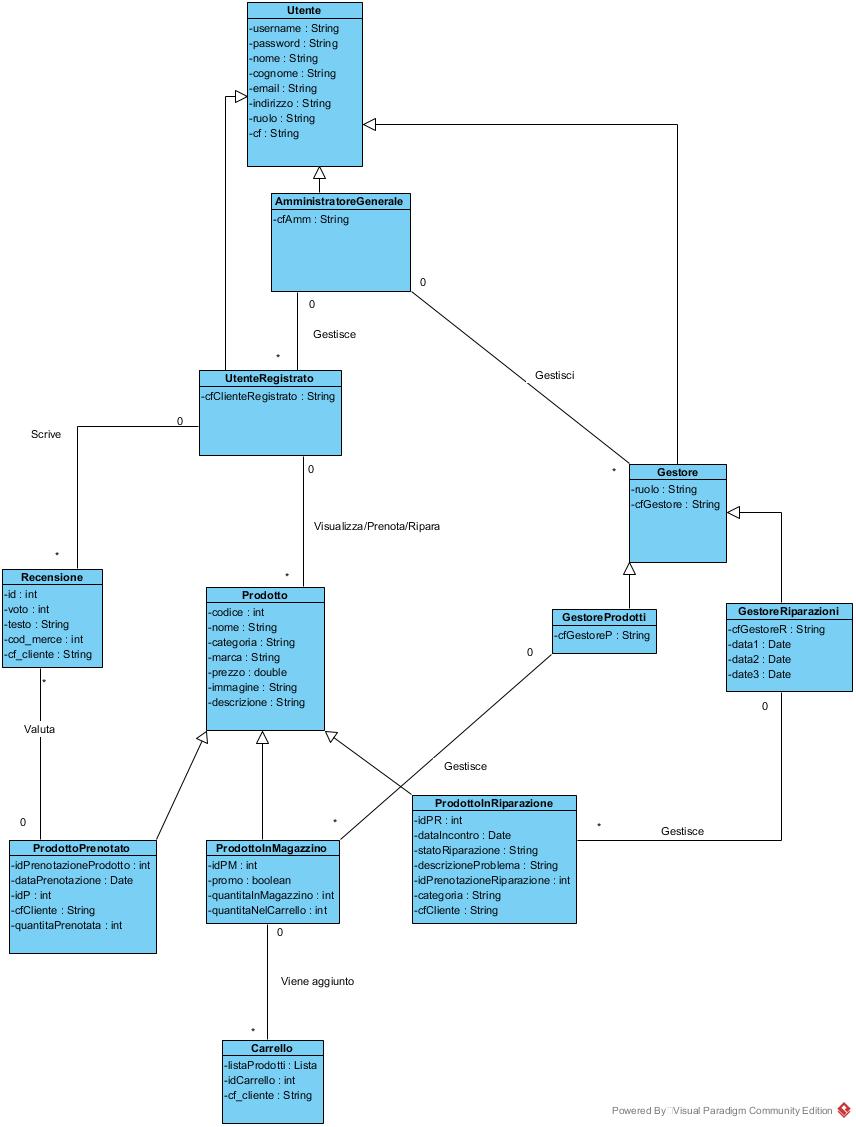
Un database di tipo relazionale, inoltre, permette la gestione e la memorizzazione

permanente di un grosso insieme di dati che devono e/o possono essere acceduti da utenti

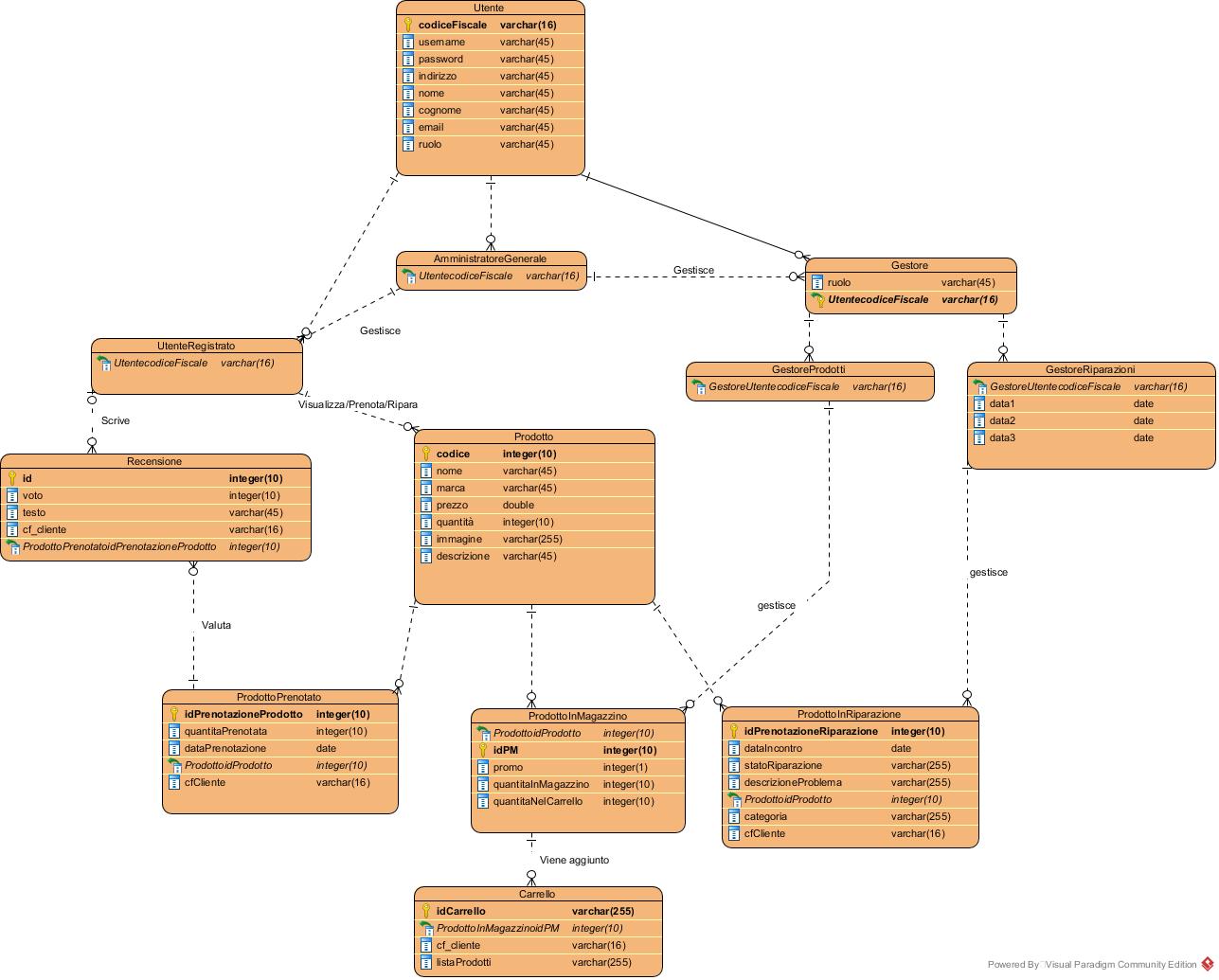
e applicazioni diverse a una granularità più fine.

Ovviamente, utilizzare un database relazionale significa avere a disposizione circa il triplo

dello spazio di memorizzazione richiesto per il corrispondente insieme di dati.



|  |  |  |
| --- | --- | --- |
| **TERMINE** | **DESCRIZIONE** | **SINONIMI** |
| Utente | Persona che si registra al sito utilizzato per la prenotazione dei prodotti online.  Persona che assolve il ruolo di Gestore del sito di vendita di prodotti online. | Cliente  Gestore |
| ProdottoInMagazzino | Prodotto a disposizione per la vendita |  |
| ProdottoInRiparazione | Prodotto consegnato per essere riparato |  |
| ProdottoPrenotato | Prodotto che è stato prenotato da un utente |  |
| Recensione | Commento su un prodotto prenotato effettuato da un utente del sito |  |
| Carrello | Lista dei prodotti selezionati da un utente |  |



UTENTE\_TABLE

|  |  |
| --- | --- |
| Codice Fiscale | Varchar(16), primary key |
| Username | Varchar(16) |
| Password | Varchar(16) |
| Nome | Varchar(16) |
| Cognome | Varchar(16) |
| Email | Varchar(30) |
| Indirizzo | Varchar(30) |
| Ruolo | Varchar(10) |

GESTORE\_TABLE

|  |  |
| --- | --- |
| Codice Fiscale | Varchar(16), primary key |
| Username | Varchar(16) |
| Password | Varchar(16) |
| Nome | Varchar(16) |
| Cognome | Varchar(16) |
| Email | Varchar(30) |
| Indirizzo | Varchar(30) |
| Ruolo | Varchar(30) |

PRODOTTOPRENOTATO\_TABLE

|  |  |
| --- | --- |
| ID\_PrenotazioneProdotto | Int, PrimaryKey |
| Nome | Varchar(16) |
| Tipologia | Varchar(16) |
| Marca | Varchar(16) |
| Prezzo | int(10,2) |
| QuantitàPrenotata | Int |
| Immagine | Varchar(100) |
| Descrizione | Varchar(300) |
| DataPrenotazione | date |
| IDProdotto | int |
| CFCliente | Varchar(16) |

PRODOTTOINRIPARAZIONE\_TABLE

|  |  |
| --- | --- |
| ID\_PrenotazioneRiparazione | Int, PrimaryKey |
| Nome | Varchar(16) |
| Categoria | Varchar(16) |
| Marca | Varchar(16) |
| Prezzo | numeric(10,2) |
| Immagine | Varchar(100) |
| Descrizione | Varchar(300) |
| Data\_Incontro | Date |
| StatoRiparazione | Varchar(255) |
| DescrizioneProblema | Varchar(255) |
| IDProdotto | Int |
| DataFineLavoro | Date |
| CFCliente | Varchar(16) |

PRODOTTOINMAGAZZINO\_TABLE

|  |  |
| --- | --- |
| ID\_ProdottoInMagazzino | Int, PrimaryKey |
| Nome | Varchar(16) |
| Tipologia | Varchar(16) |
| Marca | Varchar(16) |
| Prezzo | numeric(10,2) |
| QuantitàInMagazzino | Int |
| QuantitàNelCarrello | Int |
| Immagine | Varchar(100) |
| Descrizione | Varchar(300) |
| Promo | boolean |

CARRELLO\_TABLE

|  |  |
| --- | --- |
| IDCarrello | Int, primary key |
| ListaProdotti | Varchar(1000) |
| CF\_Cliente | Varchar(16) |

RECENSIONE\_TABLE

|  |  |
| --- | --- |
| IDRecensione | Int, primary key |
| CFCliente | Varchar(16) |
| Testo | Varchar(255) |
| Cod\_merce | int |
| Voto | int |

## **Access Control And Security**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ATTORI  OGGETTI | Utente non registrato | Utente Registrato | Gestore Prodotti | Gestore Riparazioni |
| Prodotto Prenotato |  | Visualizza  Recensisci |  |  |
| Prodotto in magazzino | Visualizza | Visualizza  Aggiungi al carrello | Aggiunge  Modifica  Rimuove |  |
| Prodotto in riparazione |  | Visualizza  Visualizza stato |  | Modificare stato riparazione  Inserire data di fine riparazione |
| Recensione |  | Visualizza |  |  |
| Carrello |  | Effettua prenotazione  Rimuovi prodotto |  |  |

Modelliamo gli accessi alle classi con una **Matrice di Accesso**: le colonne rappresentano gli attori del sistema e le righe rappresentano le classi di cui vogliamo controllare l’accesso. Un **Diritto di Accesso** è un’entrata nella matrice degli accessi ed elenca le operazioni che possono essere effettuate dagli attori sulle istanze delle classi.

## **Global Software Control**

Il sistema è un’applicazione Web gestita attraverso un server remoto. L’utente interagisce con le classi del sottosistema UI, dopo l’interazione con queste classi gli eventi vengono creati e inviati a oggetti remoti.

Per quanto riguarda il flusso di controllo esterno (fra sottosistemi), il server sarà sempre in funzione, in attesa di eventuali richieste di servizi da parte delle varie postazioni client; nel caso di più richieste contemporanee, il server utilizzerà la tecnica FIFO (First In First Out).

## **Boundary Condition**

**Start-Up** : descrive come il sistema o una componente è portato da uno stato non inizializzato a uno stato stabile (startup use case).

**Failure** (gestione delle eccezioni): le cause che li provocano possono essere molte (bug, errori, problemi esterni, come l’alimentazione elettrica) e buoni system design devono prevedere fatal failure (failure use case). I casi d’uso eccezionali estendono i casi d’uso più rilevanti.

|  |  |
| --- | --- |
| **Nome Use Case:** | Start-Server |
| **Condizioni di entrata:** | Il ServerAdministrator accede al server |
| **Flusso di eventi:** | 1. Dopo aver effettuato correttamente l’ accesso viene eseguito dal ServerAdministrator lo start-Up del server 2. Se Il Server era stato precedentemente chiuso normalmente, il server legge l’ elenco dei driver legittimi e la lista dei prodotti in vendita e prenotati. Se il server si è arresta in modo anomalo, notifica al ServerAdministrator ed esegue un controllo di coerenza ProductDBStore |
| **Condizioni di uscita:** | * Il Server è attivo e riceve una richiesta da parte del Client |

ProductDBStore: è responsabile della memorizzazione dei prodotti in vendita e prenotati in un database. Questo sottosistema supporta più driver simultanei. All’ avvio, il ProductDBStore rileva se è stato spento correttamente, in caso contratio esegue un controllo sui prodotti in vendita e prenotati e ripara i dati danneggiati se necessario.

1. **Subsystem Services**

|  |  |
| --- | --- |
| **UserManager** | |
| **Servizio** | **Descrizione** |
| public doSave(utente) | Il sottosistema permette di registrare uno utente nel sistema attraverso la compilazione di un apposito form. |
| public doUpdatePassword(cf, nuovaPassword) | Il sottosistema permette di modificare la password dell’account. |
| Public doUpdateEmail(cf, nuovaMail) | Il sottosistema permette di modificare l’e-mail dell’account. |
| public doDelete(utente) | Il sottosistema permette di cancellare un account. |
| public doRetrieveAll() | Il sottosistema permette di visualizzare gli utenti del sito |
| Public doRetrieveByKey(codiceFiscale) | Il sottosistema permette di visualizzare un utente del sito |
| Public doSaveCliente(utente) | Il sottosistema permette di salvare un cliente all’interno del sistema |
| Public doUpdateDate(cf, data) | Il sottosistema permette di aggiornare le date da parte del gestore delle riparazioni |
| Public doUpdateCliente(cliente, ruolo) | Il sottosistema permette di aggiornare il ruolo di un utente |
| Public doRetrieveUtente(user,pass) | Il sottosistema permette di ottenere un utente |
| Public doRetrieveByCodiceFiscale(codice, tipo) | Il sottosistema permette di visualizzare i prodotti prenotati da un utente |

|  |  |
| --- | --- |
| **ProdottoManager** | |
| **Servizio** | **Descrizione** |
| public doSave(prodotto) | Il sottosistema permette di aggiungere un prodotto al catalogo |
| public doUpdate(prodotto) | Il sottosistema permette di modificare un prodotto nel catalogo |
| public doDelete(prodotto) | Il sottosistema permette di eliminare un prodotto dal catalogo |
| public doProdottoByKey(id) | Il sottosistema permette di visualizzare un prodotto |
| public doRetriveAll() | Il sottosistema permette di visualizzare i prodotti presenti nel catalogo |
| Public doRetrieveOnSale() | Il sottosistema permette di visualizzare i prodotti in promozione |
| Public doUpdateQuantitaInMagazzino(codice, quantita) | Il sottosistema permette di aggiornare la quantità in magazzino di un prodotto |
| Public doUpdateQuantitaNelCarrello(codice, quantita) | Il sottosistema permette di aggiornare la quantità nel carrello |
| Public doRetrieveCategoria(categoria) | Il sottosistema permette di ottenere i prodotti di una determinata categoria |
| Public doUpdatePromo(codice, tipo) | Il sottosistema permette di aggiungere/togliere un prodotto dall’elenco dei prodotti in promozione |
| Public doUpdatePrezzo(codice, prezzo) | Il sottosistema permette di modificare il prezzo di un prodotto |
| Public doSaveInMagazzino(prodotto, promo, quantita) | Il sottosistema permette di salvare un prodotto in magazzino |

|  |  |
| --- | --- |
| **RiparazioneManager** | |
| **Servizio** | **Descrizione** |
| public doSave(data, prodotto, codice) | Il sottosistema permette di aggiungere un prodotto in riparazione |
| public doUpdateData(data, idRiparazione) | Il sottosistema permette di modificare le date degli incontri |
| public doUpdateStato(idRiparazione, data) | Il sottosistema permette di aggiornare lo stato della riparazione di un prodotto |
| public doRetrieveAll() | Il sottosistema permette di visualizzare le date stabilite dal gestore delle riparazioni |
| Public doRetrieveAllRiparazioni() | Il sottosistema permette di ottenere tutti i prodotti in riparazione |

|  |  |
| --- | --- |
| **Carrellomanager** | |
| **Servizio** | **Descrizione** |
| public doInsertProdotti(carrello) | Il sottosistema permette di aggiungere prodotti al carrello |
| public doDeleteProdotti(carrello) | Il sottosistema permette di eliminare prodotti dal carrello |
| public doRetrieveAll(codiceCliente) | Il sottosistema permette di visualizzare i prodotti presenti nel carrello |
| Public doPrenota(codiceCliente,prodotto) | Il sottosistema permette di prenotare un prodotto |
| Public doRetrieveByKey(codiceCliente) | Il sottosistema permette di visualizzare i prodotti presenti nel carrello |

|  |  |
| --- | --- |
| **RecensioneManager** | |
| **Servizio** | **Descrizione** |
| public doSave(prodotto,recensione) | Il sottosistema permette di aggiungere una recensione a un prodotto prenotato |