



# SDD

# System Design

# Document

**Car – Zone**  
Versione 0.3

20/11/2024

**Coordinatore del progetto:**

Nome	Matricola
Francesco Pio Cataudo	0512116773

**Partecipanti:**

Nome	Matricola
Francesco Pio Cataudo	0512116773
Francesco Santoro	0512117079
Francesco Pio Bottaro	0512118180
Errico Aquino	0512117730

## Revision History

---

Data	Versione	Descrizione	Autori
<b>19/11/2024</b>	0.1	Strutturazione documento ed informazioni iniziali	Team
<b>20/11/2024</b>	0.2	Stesura decomposizione sottosistemi e mapping hardware-software	Team
<b>20/11/2024</b>	0.3	Stesura gestioni dati persistenti e controllo degli accessi e sicurezza	Team

## Sommario

Revision History.....	3
1.Introduzione .....	5
1.1 Scopo del Sistema .....	5
1.2 Design Goals .....	6
1.3 Definitions, acronyms, and abbreviations .....	9
1.4 Riferimenti .....	9
1.5 Overview .....	10
2. Architettura Software Attuale.....	11
3. Architettura Sistema Proposto .....	11
3.1 Overview .....	11
3.2 Decomposizione In Sottosistemi.....	12
3.3 Hardware/Software Mapping.....	14
3.4 Gestione Dati Persistenti .....	17
3.5 Controlli accessi e sicurezza .....	19
3.6 Controllo Flusso Globale Software.....	21
3.7 Condizione Limite .....	21
4 Glossario Dei Servizi Dei Sottosistemi.....	21

---

# 1.Introduzione

---

## 1.1 Scopo del Sistema

---

Car-Zone è una piattaforma online progettata per rivoluzionare l'esperienza di acquisto di autovetture, semplificando e ottimizzando il processo per clienti e concessionaria. L'obiettivo principale è offrire agli utenti la possibilità di esplorare, selezionare e acquistare un'auto direttamente online, eliminando la necessità di recarsi fisicamente in concessionaria. Questo sistema è pensato per rendere più efficienti tutte le fasi di vendita, riducendo il tempo e le risorse necessarie sia per i clienti che per l'azienda.

La piattaforma Car-Zone offre diverse funzionalità per rispondere alle esigenze di due gruppi principali di utenti: i clienti e gli amministratori.

Per i clienti, Car-Zone consente di registrarsi, effettuare il login, esplorare il catalogo delle auto disponibili, gestire ordini e visualizzare i dettagli delle transazioni. L'interfaccia utente è intuitiva e accessibile, consentendo una navigazione fluida tra le sezioni, facilitando la ricerca dei modelli desiderati e semplificando la finalizzazione degli acquisti.

Dal lato amministrativo, il sistema permette una gestione centralizzata e agevole del catalogo delle autovetture, con la possibilità di aggiungere nuovi modelli, modificarne le caratteristiche e gestire lo stato degli ordini. Gli amministratori possono monitorare le transazioni e visualizzare lo storico degli ordini, assicurando un controllo completo su tutte le operazioni effettuate dai clienti.

Per garantire un'esperienza di utilizzo ottimale, Car-Zone è costruito per essere affidabile, sicuro e scalabile, rispettando specifici requisiti non funzionali come l'accesso simultaneo da parte di più utenti, la sicurezza dei dati personali e le prestazioni rapide. Questo sistema è quindi pensato per supportare la crescita futura dell'azienda, integrando una gestione agile degli ordini e offrendo ai clienti un'esperienza d'acquisto comoda, veloce e moderna.

---

## 1.2 Design Goals

---

I criteri di progettazione sono organizzati in cinque gruppi: prestazioni, affidabilità, costi, manutenzione e criteri dell'utente finale.

### Performance

- **Tempo di risposta:** Car-Zone deve rispondere alle richieste dell'utente entro 5 secondi. Questo assicura un'esperienza utente rapida e reattiva, fondamentale per accessi frequenti al database e per la fruibilità del sito.
- **Throughput:** Il sistema deve essere in grado di gestire simultaneamente più richieste da diversi utenti, specialmente in momenti di traffico elevato. Sarà ottimizzato per processare molteplici richieste simultaneamente, riducendo al minimo i rallentamenti.
- **Uso della memoria:** Il sistema utilizzerà efficientemente la memoria, per bilanciare velocità e consumo di risorse.

### Dependability

- **Robustezza:** Car-Zone deve essere in grado di gestire input non validi senza causare interruzioni o comportamenti inaspettati. Verranno implementati controlli di validazione per prevenire crash e rispondere in modo sicuro agli input non corretti o fuori dai limiti.
- **Disponibilità:** Car-Zone deve garantire un'elevata disponibilità, assicurando che il sistema sia accessibile agli utenti per la maggior parte del tempo. L'obiettivo è mantenere una percentuale di uptime che permetta l'uso continuo per le attività quotidiane, minimizzando le interruzioni programmate per la manutenzione e riducendo al minimo i tempi di inattività non previsti.
- **Affidabilità:** Car-Zone deve mantenere un comportamento coerente e affidabile, con output che rispecchino correttamente le aspettative dell'utente anche in presenza di errori imprevisti.
- **Tolleranza ai guasti:** Il sistema deve continuare a funzionare in condizioni di errore (ad esempio, perdita temporanea di connettività).

- **Sicurezza:** Poiché Car-Zone gestirà dati sensibili degli utenti, il sistema sarà progettato per resistere ad attacchi malevoli, assicurando che le informazioni siano protette tramite autenticazione e crittografia adeguata.
- **Disponibilità:** Il sistema deve essere disponibile per gli utenti il più possibile, con una strategia di manutenzione programmata per minimizzare i tempi di inattività.

## Cost

- **Costo di sviluppo:** I costi iniziali di sviluppo saranno ottimizzati tramite una progettazione modulare e componenti riutilizzabili, riducendo il tempo necessario per future implementazioni e aggiornamenti.
- **Costo di distribuzione:** L'implementazione e l'installazione del sistema devono essere eseguite in modo da minimizzare i costi per gli utenti, offrendo compatibilità cross-platform per semplificare l'accesso.
- **Costo di manutenzione:** Il sistema sarà progettato per facilitare correzioni di bug e aggiornamenti, garantendo nel contempo un basso costo di manutenzione a lungo termine.
- **Costo di amministrazione:** Saranno implementati strumenti di amministrazione facili da usare per ridurre il tempo e il costo di gestione giornaliera del sistema.

## Maintenance

- **Estensibilità:** Car-Zone sarà progettato in modo modulare per permettere l'aggiunta di nuove funzionalità in futuro, con il minimo impatto sul sistema esistente.
- **Modificabilità:** Le funzionalità di base devono poter essere aggiornate o modificate con facilità, per poter adattare il sistema a nuovi requisiti o funzionalità richieste dal mercato.
- **Adattabilità:** Il sistema deve essere facilmente adattabile a diversi domini applicativi, permettendo il riutilizzo delle funzionalità principali in altri contesti o settori.
- **Portabilità:** Il sistema sarà sviluppato per essere compatibile con diverse piattaforme, così da facilitarne il trasferimento o l'adattamento in altre applicazioni o contesti.
- **Leggibilità:** Il codice del sistema deve essere scritto in modo chiaro e leggibile per facilitare la comprensione da parte degli sviluppatori, sia attuali che futuri.
- **Tracciabilità dei requisiti:** Ogni requisito sarà mappato al codice specifico per facilitare le verifiche e garantire che il sistema soddisfi le specifiche concordate.

## End User Criteri

- **Utilità:** Car-Zone deve supportare efficientemente le attività chiave dell'utente, come la ricerca e l'interazione con i veicoli disponibili, per garantire una user experience ottimale.
- **Usabilità:** L'interfaccia del sistema sarà intuitiva e facile da usare per utenti di diversi livelli di esperienza. L'accesso e la navigazione saranno ottimizzati per l'uso su desktop e dispositivi mobili, con un design responsive per migliorare l'accessibilità.

I trade-off sono necessari per bilanciare i design goals con le limitazioni di risorse, tempo, e budget.

## Tempo di risposta vs. Consumo di memoria

- **Descrizione:** Per ottenere un tempo di risposta veloce, Car-Zone potrebbe utilizzare tecniche di caching intensivo. Tuttavia, questo può aumentare l'uso di memoria, specialmente se viene memorizzata molta cache per migliorare le performance.
- **Risoluzione del Trade-Off:** Si potrebbe adottare una strategia di caching mirata, mantenendo solo le informazioni più frequentemente utilizzate per ridurre l'uso eccessivo di memoria, bilanciando così velocità e risorse di sistema.

## Costo di sviluppo vs. Qualità

- **Descrizione:** Se il budget fosse limitato, si potrebbero ridurre i costi di sviluppo adottando soluzioni meno costose o più rapide, ma ciò potrebbe influire sulla qualità del sistema finale.
- **Risoluzione del Trade-Off:** Saranno selezionate priorità tra le funzionalità principali e quelle secondarie. Funzionalità opzionali o estetiche potrebbero essere posticipate, garantendo il rilascio puntuale del sistema principale e ottimizzando i costi.

## Sicurezza vs. Usabilità

- **Descrizione:** L'incremento di misure di sicurezza, come autenticazioni multiple o restrizioni di accesso, potrebbe rendere il sistema meno intuitivo per gli utenti finali.
- **Risoluzione del Trade-Off:** Si bilancerà la sicurezza con l'usabilità integrando procedure di autenticazione semplici ma sicure, come l'autenticazione a due fattori opzionale, offrendo un buon livello di sicurezza senza compromettere eccessivamente la user experience.



## Scalabilità vs. Costi di Manutenzione

- **Descrizione:** Per permettere a Car-Zone di scalare a un numero crescente di utenti, potrebbe essere necessaria un'architettura più complessa e modulare, aumentando i costi di manutenzione a lungo termine.
- **Risoluzione del Trade-Off:** Si opterà per un'architettura modulare, ma solo per le componenti che richiedono una scalabilità immediata, implementando miglioramenti su altre parti del sistema solo quando necessario per evitare costi anticipati.

## Tempo di consegna vs. Funzionalità

- **Descrizione:** Rispettare le scadenze potrebbe richiedere di rinunciare ad alcune funzionalità o di ridurre il tempo dedicato ai test, influenzando sulla completezza e sull'affidabilità del sistema.
- **Risoluzione del Trade-Off:** In accordo con il cliente, si stabiliranno le funzionalità essenziali per il rilascio iniziale. Altre funzionalità meno critiche potranno essere incluse in aggiornamenti successivi, garantendo così che la consegna avvenga nei tempi stabiliti.

---

## 1.3 Definitions, acronyms, and abbreviations

---

---

## 1.4 Riferimenti

---

- Documento di Analisi dei requisiti relativo a questo progetto
- Object-oriented-Software-Engineering-3rd-Edition

---

## 1.5 Overview

---

Il documento è organizzato nel seguente modo.

• **Introduzione:** Definisce il contesto e gli obiettivi del sistema, evidenziando l'importanza di un design scalabile, sicuro ed efficiente per trasformare il processo di vendita di autovetture in un'esperienza digitale.

• **Architettura del Sistema Attuale:**

Analizza il sistema esistente, basato su processi manuali e privo di strumenti digitali, sottolineandone le inefficienze e le limitazioni operative.

• **Architettura del Sistema Proposto:**

Fornisce una visione d'insieme del nuovo sistema, includendo:

- Sottosistemi e interazioni.
- Allocazione hardware/software.
- Sicurezza, gestione dati e controllo dei flussi.

• **Condizioni Limite e Sicurezza:**

Descrive come il sistema affronterà scenari critici, garantendo resilienza, robustezza e protezione dei dati.

- **Glossario dei Servizi:**

Elenca e definisce i servizi principali offerti dai vari sottosistemi, fornendo una visione chiara delle funzionalità.

---

## 2. Architettura Software Attuale

---

Attualmente, il sistema utilizzato da Car-Zone si basa interamente su interazioni manuali e non dispone di una piattaforma online per la gestione delle vendite. Questo comporta che il cliente, interessato all'acquisto di un'auto, debba contattare la concessionaria tramite telefono o email per ottenere informazioni preliminari sulla disponibilità delle auto. Successivamente, è necessario fissare un appuntamento e recarsi fisicamente presso la concessionaria per consultare il catalogo delle auto e procedere all'acquisto.

Durante il processo, non esiste un sistema centralizzato per il tracciamento degli ordini. Le informazioni vengono comunicate al cliente tramite chiamate o email, mentre lo stato dell'ordine e le date di consegna sono gestiti manualmente. Infine, il cliente deve ritornare in concessionaria alla data prestabilita per ritirare il veicolo.

Questo approccio presenta diverse limitazioni. L'assenza di un sistema digitale rallenta il processo e richiede un notevole impegno di tempo da parte sia del cliente che dell'azienda. Inoltre, la gestione manuale delle operazioni aumenta il rischio di errori e rende il processo meno efficiente. Il catalogo dei veicoli non è accessibile online e le sue informazioni non sono aggiornate in tempo reale, limitando la visibilità delle offerte. L'accessibilità per i clienti è ridotta, poiché richiede necessariamente la presenza fisica in concessionaria.

---

## 3. Architettura Sistema Proposto

---

### 3.1 Overview

---

Il sistema proposto per **Car-Zone** si basa su un'architettura moderna organizzata secondo il pattern **MVC (Model-View-Controller)**, che permette di separare nettamente la gestione dei dati, la logica

applicativa e la presentazione. Questo approccio consente di superare le limitazioni del sistema attuale, garantendo maggiore efficienza, scalabilità e manutenibilità.

Le funzionalità offerte variano in base al tipo di utente:

- **I clienti registrati** possono esplorare il catalogo, visualizzare lo storico ordini e finalizzare gli acquisti direttamente online.
- **Gli amministratori** possono aggiungere, eliminare o modificare i veicoli nel catalogo, approvare gli ordini e visualizzare lo storico dei clienti e degli ordini.
- **Gli utenti non registrati** hanno accesso limitato al sistema, potendo consultare solo il catalogo delle auto e visualizzarle in modo dettagliato. Tuttavia, possono registrarsi per accedere alle funzionalità complete.

L'adozione di un'architettura MVC consente di separare chiaramente le responsabilità, migliorando la modularità del sistema. Ad esempio, eventuali modifiche al layout dell'interfaccia utente non richiedono interventi sul database o sulla logica di business. Allo stesso modo, l'aggiornamento delle regole aziendali può avvenire senza impattare la visualizzazione o l'esperienza utente.

---

## 3.2 Decomposizione In Sottosistemi

---

### Identificazione dei sottosistemi per CarZone

#### 1. Sottosistema di Autenticazione e Sicurezza

- **Descrizione:** Gestisce l'autenticazione degli utenti, il controllo degli accessi e la protezione dei dati sensibili tramite crittografia e altre misure di sicurezza.
- **Funzionalità principali:** Login, registrazione, gestione dei ruoli e autorizzazioni.

#### 2. Sottosistema di Gestione delle Auto

- **Descrizione:** Si occupa di organizzare, memorizzare e aggiornare i dati relativi alle auto disponibili sulla piattaforma.
- **Funzionalità principali:** Aggiunta, modifica, rimozione e visualizzazione dei dettagli delle auto (marca, modello, prezzo, ecc.).

#### 3. Sottosistema di Ricerca e Filtro

- **Descrizione:** Permette agli utenti di cercare e filtrare le auto in base a parametri come marca, modello, e tipo.
- **Funzionalità principali:** Implementazione di query dinamiche per la ricerca e opzioni di filtro avanzate.

#### 4. Sottosistema di Gestione degli Ordini

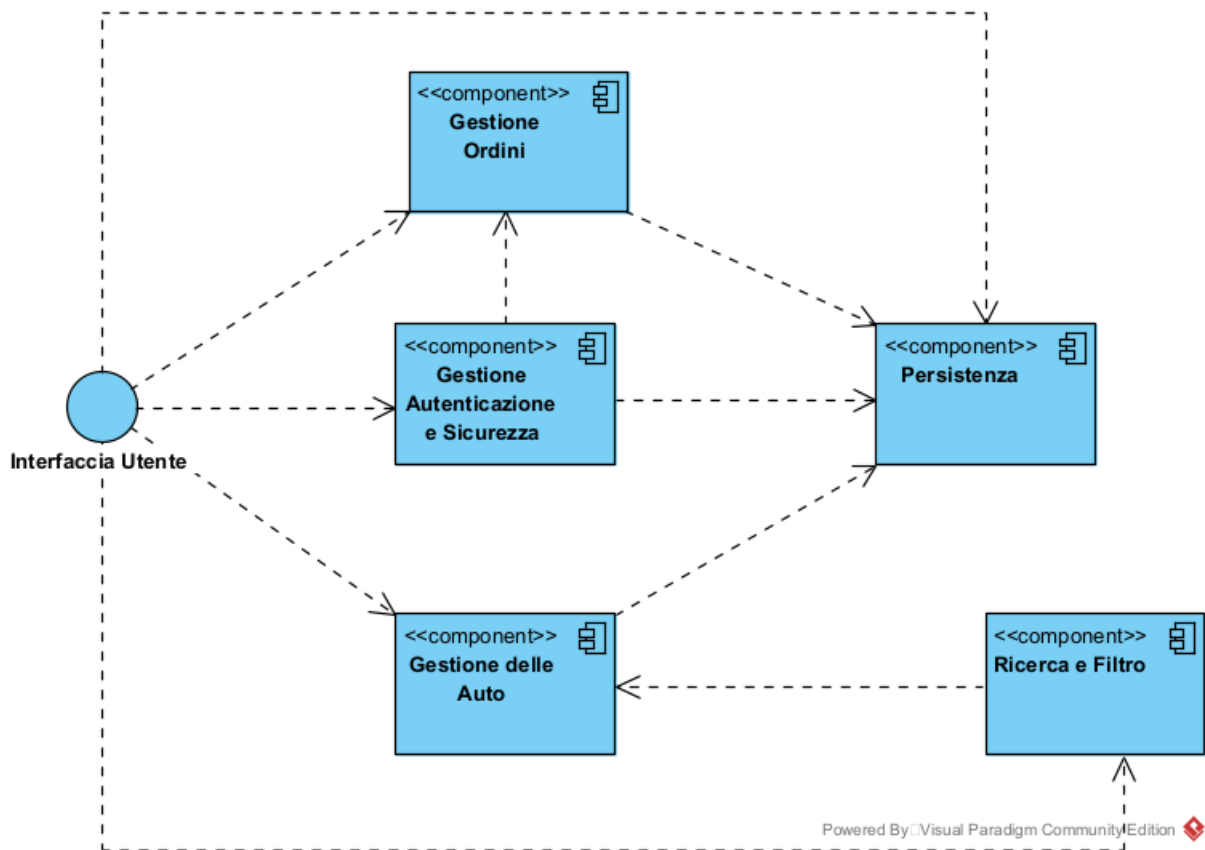
- **Descrizione:** Gestisce ordini delle auto effettuati dai clienti.
- **Funzionalità principali:** Conferma e monitoraggio degli ordini.

#### 5. Sottosistema di Interfaccia Utente (UI)

- **Descrizione:** Gestisce l'interfaccia utente per assicurare un'esperienza utente intuitiva e user-friendly sia su dispositivi desktop che mobili.
- **Funzionalità principali:** Pagine per la visualizzazione e interazione, feedback sulle azioni dell'utente e supporto per design responsive.

#### 6. Sottosistema di Persistenza

- **Descrizione:** Si occupa dell'archiviazione, recupero e gestione dei dati persistenti del sistema, inclusi utenti, veicoli e transazioni.
- **Funzionalità principali:** Operazioni CRUD (Create, Read, Update, Delete) ottimizzate per garantire tempi di risposta rapidi.



### 3.3 Hardware/Software Mapping

CarZone è un sistema distribuito che opera su una configurazione multi-nodo per gestire sia il carico di lavoro che la distribuzione geografica degli utenti. La mappatura hardware/software prevede tre nodi principali: il **Web Server**, il **Database Server** e il **Client**. Ogni nodo è progettato per ospitare specifici sottosistemi, ottimizzando le prestazioni e la modularità del sistema.

#### Nodo 1: Web Server

Il **Web Server** è il cuore del sistema, responsabile della gestione delle richieste degli utenti, dell'elaborazione delle logiche di business e della comunicazione con il database.

**Sottosistemi ospitati:**

1. **Sottosistema di autenticazione:** Gestisce il login, la registrazione e la validazione degli utenti.
  2. **Sottosistema di Gestione delle Auto:** Gestisce le operazioni CRUD (Create, Read, Update, Delete) sulle auto.
  3. **Sottosistema di Ricerca e Filtro:** Elabora le richieste di ricerca e applica filtri ai dati.
  4. **Sottosistema di Gestione degli Ordini:** Gestisce gli ordini.
- **Tecnologie:**
    - **Server Web:** Apache.
    - **Linguaggio di Programmazione:** Java.
    - **Protocollo di Comunicazione:** HTTP/HTTPS per comunicare con i client.

## Nodo 2: Database Server

Il **Database Server** è dedicato alla gestione e alla conservazione dei dati persistenti del sistema. Include tutti i dati relativi agli utenti, alle auto e agli ordini. Il nodo ospita un sistema di gestione del database (DBMS) come **MySQL**.

- **Sottosistemi ospitati:**
  1. **Persistence Subsystem:** Archivia e recupera dati tramite operazioni CRUD standard.
  2. **Data Synchronization:** Garantisce la consistenza dei dati tra il database e i nodi applicativi.
- **Tecnologie:**
  - **DBMS:** MySQL.
  - **Infrastruttura:** Server dedicato con storage ridondante per assicurare affidabilità.

## Nodo 3: Client

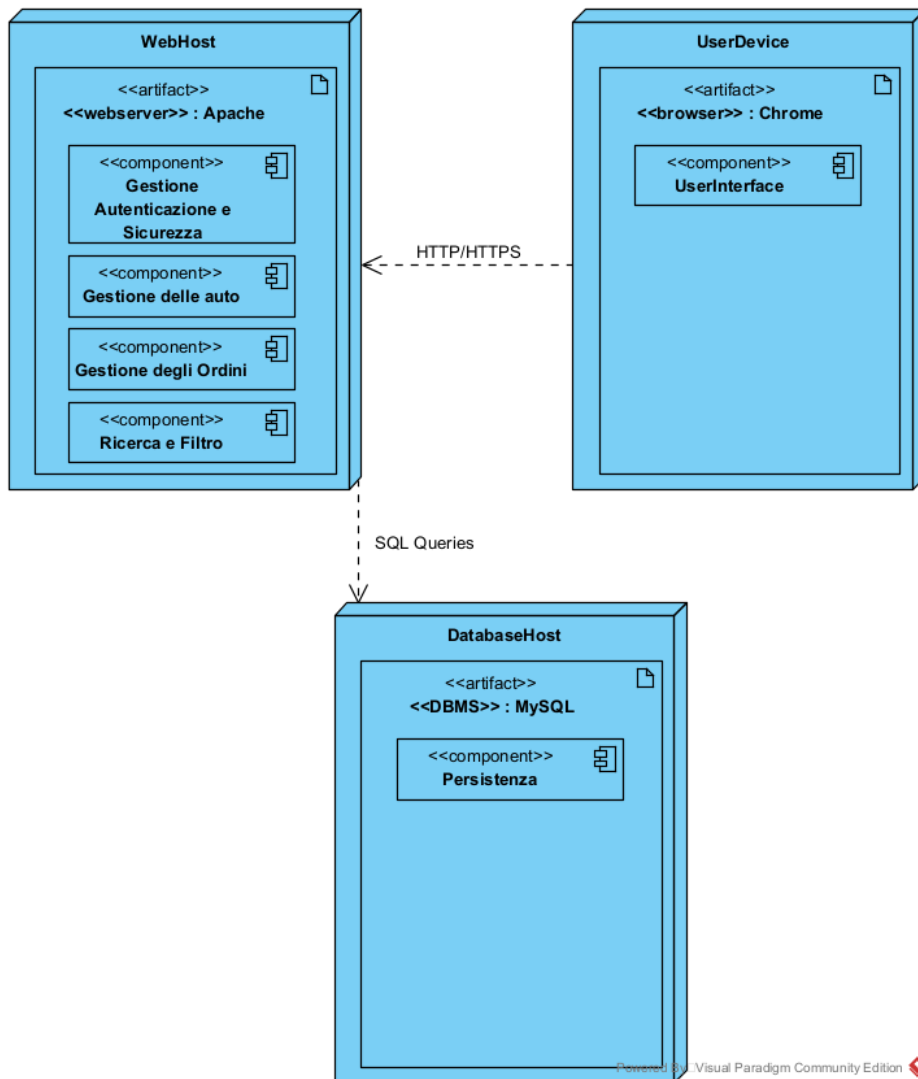
Il **Client** rappresenta l'interfaccia utente attraverso cui gli utenti interagiscono con CarZone.

- **Sottosistemi ospitati:**
  1. **Sottosistema di Interfaccia Utente:** Mostra le informazioni agli utenti e gestisce l'interazione tramite interfacce grafiche intuitive.
- **Tecnologie:**
  - **Browser:** Chrome, Safari, Firefox.

## Comunicazione tra Nodi

La comunicazione tra i nodi avviene tramite protocolli standard:

- **Client-Web Server:** Utilizza **HTTP/HTTPS** per inviare richieste e ricevere risposte.
- **Web Server-Database Server:** Utilizza query SQL per accedere e manipolare i dati nel database.





---

### 3.4 Gestione Dati Persistenti

---

La gestione dei dati persistenti è fondamentale per garantire che informazioni critiche siano conservate e recuperabili anche in caso di riavvio del sistema o di eventi imprevisti. Nel contesto del progetto, è stato deciso di utilizzare una strategia di gestione della persistenza basata su un Database Relazionale (MySQL) per sfruttare la sua capacità di gestire dati complessi e assicurare scalabilità, integrità e sicurezza.

#### Identificazione degli Oggetti Persistenti

Gli oggetti persistenti sono stati individuati attraverso l'analisi delle entità chiave che devono sopravvivere alla chiusura del sistema. Questi includono:

- **Utenti:** Informazioni su Admin e Clienti, accorpati in una singola entità per semplificare la gestione.
- **Auto:** Dati relativi alle auto come modello, prezzo, caratteristiche tecniche.
- **Ordini:** Informazioni sugli acquisti effettuati dagli utenti.

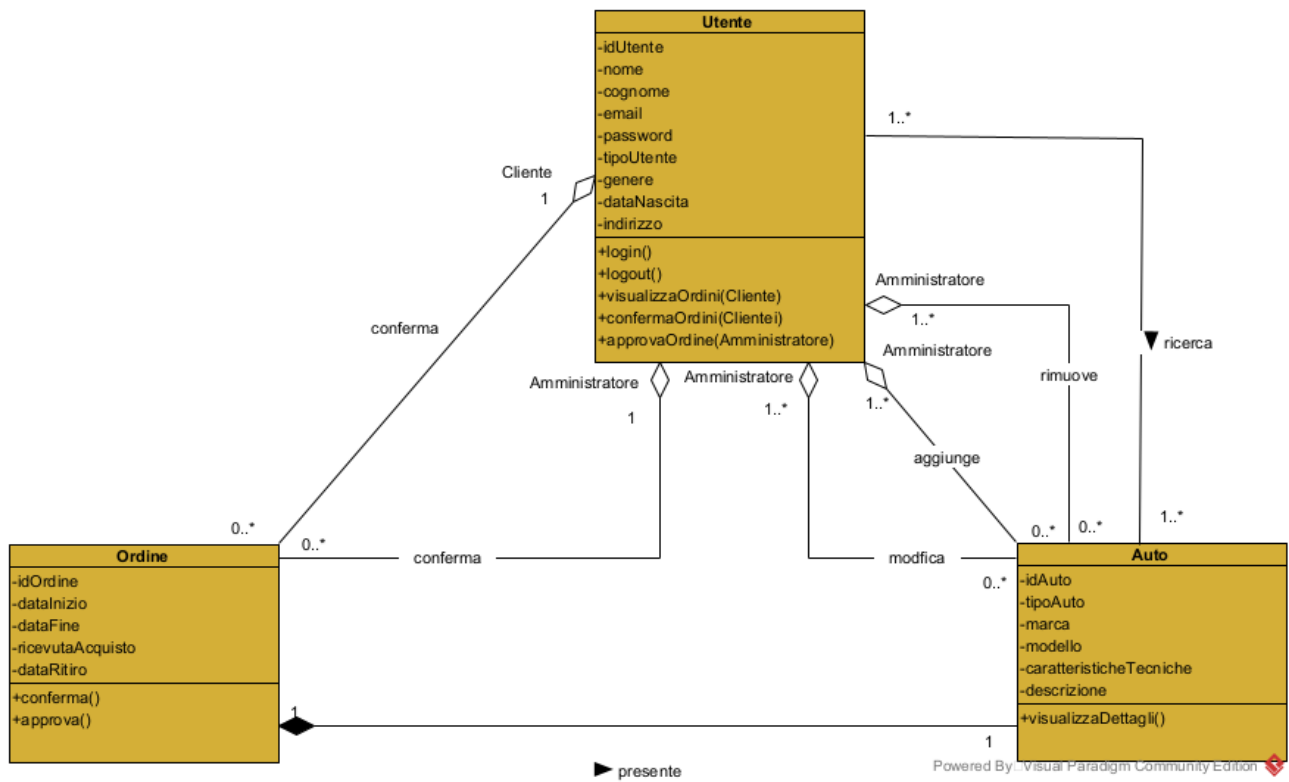
#### Accorpamento di Admin e Cliente

Per semplificare la gestione dei dati, è stato deciso di accorpare gli utenti **Admin** e **Cliente** in un'unica entità **Utente**, differenziandoli tramite un attributo ruolo. Questo approccio offre i seguenti vantaggi:

- Riduce la complessità delle tabelle nel database.
- Consente una gestione uniforme delle operazioni CRUD.
- Semplifica la gestione dei permessi e l'integrazione con i moduli di autenticazione.

La tabella **Utenti** include:

- **Attributi comuni:** ID, Nome, Cognome, Email, Password.
- **Ruolo:** Specifica se l'utente è un **Admin** o un **Cliente**.
- **Informazioni aggiuntive:** Per esempio, il Cliente può avere uno storico degli ordini, mentre l'Admin ha accesso a privilegi gestionali.



---

### 3.5 Controlli accessi e sicurezza

---

#### Matrice di Controllo degli Accessi (Access Control Matrix)

Ruolo / Sottosistema	Gestione Autovetture	Gestione Ordini
<b>Cliente</b>	- Visualizza il catalogo di autovetture	- Visualizza ordini - Confirma ordini
<b>Utente Non Registrato</b>	- Visualizza il catalogo di autovetture	- Nessun accesso
<b>Amministratore</b>	- Aggiunge, modifica, elimina autovetture	- Visualizza ordini clienti - Approva ordini clienti

##### 1. Cliente:

- Ha accesso in sola lettura al catalogo delle autovetture.
- Può visualizzare i propri ordini e confermarli.

##### 2. Utente Non Registrato:

- Può visualizzare il catalogo delle autovetture.
- Non ha accesso alla gestione degli ordini.

##### 3. Amministratore:

- Ha pieno controllo sul catalogo delle autovetture (può aggiungere, modificare o eliminare).
- Può visualizzare tutti gli ordini dei clienti e approvarli, se necessario.

#### Rappresentazione Matriciale

Ruolo / Classe	Autovettura	Ordine
<b>Cliente</b>	- Read	- Read, Update
<b>Utente Registrato</b>	- Read	- None
<b>Amministratore</b>	- Create, Read, Update, Delete	- Read

## Descrizione delle Opzioni Implementative

### 1. Global Access Table:

- Una tabella globale in cui ogni riga rappresenta una combinazione di attore-classe-operazione. Questo metodo è esplicito ma può richiedere molto spazio.

### 2. Access Control List (ACL):

- Una lista per ogni classe che contiene le coppie (attore, operazione). Ad esempio:

- **Autovettura ACL:**

- Cliente → Read
- Utente Registrato → Read
- Amministratore → CRUD

- **Ordine ACL:**

- Cliente → Read, Update (propri ordini)
- Amministratore → Read (tutti gli ordini)

- Questo metodo è utile per verificare rapidamente chi ha accesso a una classe.

### 3. Capabilities:

- Associando coppie (classe, operazione) agli attori:
  - Cliente → [(Autovettura, Read), (Ordine, Read), (Ordine, Update)]
  - Utente Registrato → [(Autovettura, Read)]
  - Amministratore → [(Autovettura, CRUD), (Ordine, Read)]

---

### 3.6 Controllo Flusso Globale Software

---

---

### 3.7 Condizione Limite

---

---

## 4 Glossario Dei Servizi Dei Sottosistemi

---