

**UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE**

**DIPARTIMENTO DI INGEGNERIA**



**CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE PER LA CYBERSECURITY**

**PROJECT WORK**

**Tema selezionato**

**Elettronica Iovine s.r.l.**

**DOCENTE**

PROF. Daniele Granata

**STUDENTE**

Antonio Gaudino, 0334000091

## Traccia d'esame – Progetto di sistema informativo per la gestione della piattaforma web di “Elettronica Iovine srl”.

Il presente elaborato documenta lo sviluppo di un progetto di sistema informativo, assegnato come traccia d'esame, finalizzato alla realizzazione di una piattaforma web basata su un database relazionale (SQL) per conto dell'azienda “Elettronica Iovine srl”. L'obiettivo del progetto è la progettazione e implementazione di un sistema integrato per la gestione dei principali servizi aziendali, con particolare riferimento alle seguenti funzionalità:

- **E-commerce:** modulo dedicato alla vendita al dettaglio di prodotti di informatica ed elettronica, con gestione del catalogo, carrello e ordini.
- **Assistenza Clienti:** sistema per la gestione delle richieste di supporto tecnico, che consente agli utenti di aprire ticket di assistenza relativi ai prodotti acquistati.
- **Area Riservata:** sezione ad accesso limitato per il personale autorizzato, con funzionalità per l'aggiunta e la gestione dei prodotti in vendita, la visualizzazione dei clienti e delle loro richieste di supporto, nonché l'assegnazione degli interventi ai tecnici incaricati.

### Tipologie di Utenza e Funzionalità

La piattaforma è progettata per supportare due diverse tipologie di utenza, ciascuna con specifici privilegi e funzionalità:

- **Clienti:** gli utenti registrati come clienti possono accedere a un insieme di funzionalità che consentono di:
  - Acquistare prodotti presenti nel catalogo online;
  - Aprire ticket di assistenza per segnalare malfunzionamenti o richiedere supporto tecnico su prodotti acquistati;
  - Visualizzare le proprie informazioni personali e il saldo disponibile (ricaricabile esclusivamente presso la sede fisica dell'azienda);
  - Lasciare una recensione sugli interventi di assistenza ricevuti.
- **Tecnici:** gli utenti con profilo tecnico hanno accesso a funzionalità gestionali e operative, tra cui:
  - Aggiungere o rimuovere prodotti dal listino disponibile nell'e-commerce;
  - Prendere in carico le richieste di assistenza aperte dai clienti;
  - Visualizzare gli ordini effettuati e le informazioni associate ai clienti.

Questa suddivisione dei ruoli consente di garantire un flusso operativo coerente con le esigenze aziendali, assicurando al contempo la protezione e la corretta gestione delle informazioni sensibili.

Il progetto evidenzia una notevole lungimiranza, ponendo solide basi per l'integrazione futura di funzionalità attualmente non incluse, quali: la registrazione degli utenti cliente, una sezione dedicata ai pagamenti, l'adeguamento al GDPR, oltre ad eventuali estensioni future. La piattaforma sviluppata per Elettronica Iovine rappresenta quindi un'ottima base scalabile, pronta ad accogliere evoluzioni e miglioramenti nel tempo.

## Obiettivo del progetto

Il progetto prevede lo sviluppo di un applicativo web con un database al fine di migliorare la gestione delle attività commerciali.

## Descrizione delle Entità del Sistema

La piattaforma gestisce due tipologie di utenza: Cliente e Tecnico, entrambe derivano da una generalizzazione dell'entità Utente.

La generalizzazione è totale e disgiunta: ogni utente appartiene obbligatoriamente a una sola delle due sottocategorie.

- L'entità Utente contiene i seguenti attributi:  
(nome, cognome, e-mail, telefono, tipo)

### 1. Profilo Clientela

È una specializzazione dell'entità Utente riservata agli utenti che effettuano acquisti tramite la piattaforma.

Contiene i seguenti attributi:  
(saldo, residenza)

### 2. Profilo Tecnico

È una specializzazione dell'entità Utente riservata agli utenti che si occupano della gestione e manutenzione dei prodotti.

Contiene il seguente attributo:  
(specializzazione)

## Prodotto Acquistabile

Rappresenta i prodotti disponibili all'acquisto sulla piattaforma.

Contiene i seguenti attributi:

(id\_prodotto, nome, immagine, descrizione, prezzo, quantità\_rimanente)

## Ordine

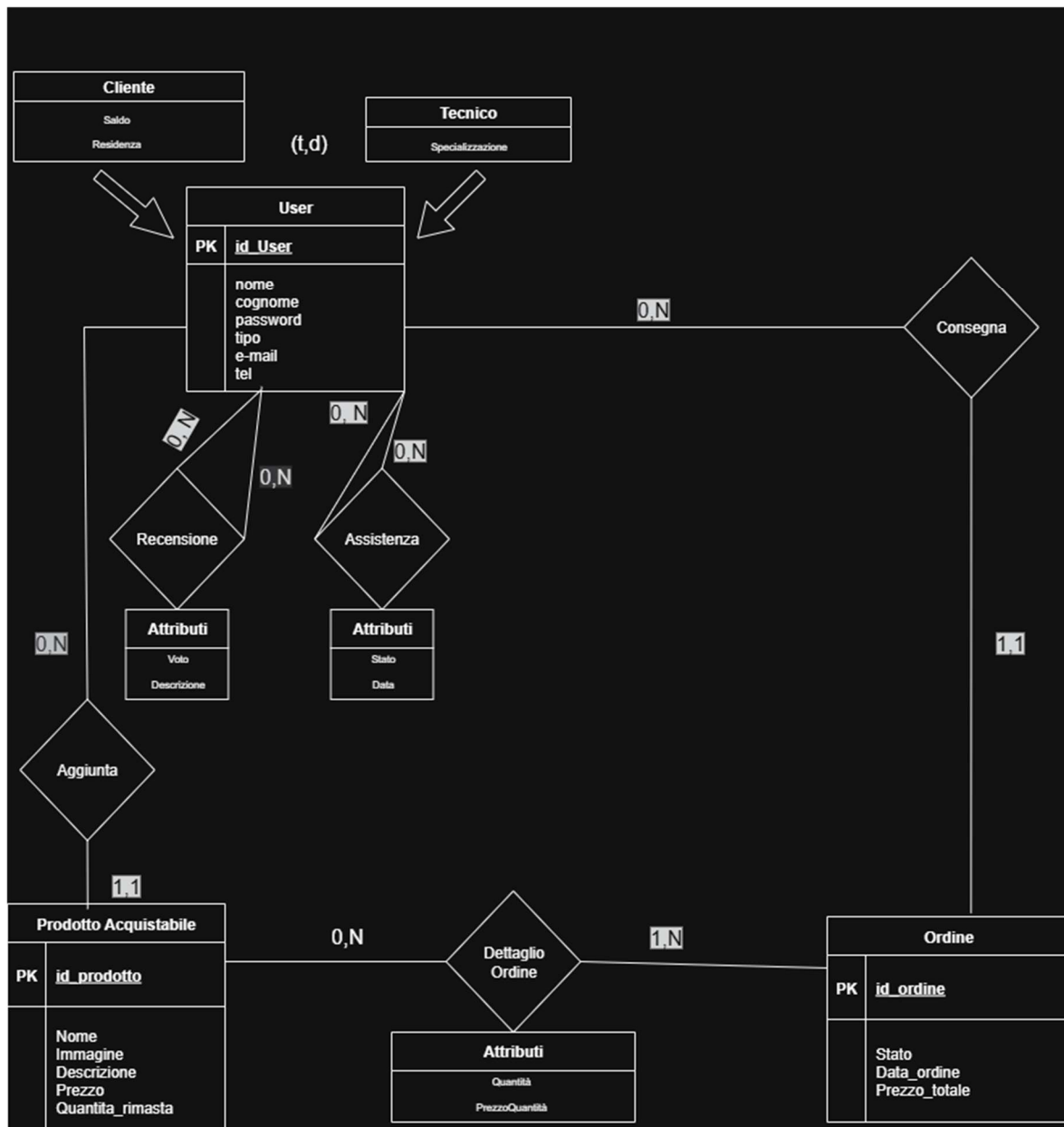
Rappresenta un ordine effettuato da un cliente sulla piattaforma.

Contiene i seguenti attributi:

(id\_ordine, stato, data\_ordine, prezzo\_totale)

## Analisi e progettazione concettuale

### Modello Entità Relazione



### Implementazione della Generalizzazione degli Utenti

Il progetto prevede l'implementazione dell'entità **Utente** mediante una generalizzazione totale e disgiunta, che definisce due sottotipi: **Cliente** e **Tecnico**.

- **Generalizzazione Totale:** ogni istanza dell'entità padre **Utente** deve appartenere ad almeno una delle entità figlie. In altre parole, se esiste un utente, allora è necessariamente un **Cliente** oppure un **Tecnico**.
- **Generalizzazione Disgiunta:** ogni istanza dell'entità padre **Utente** può appartenere a una sola tra le entità figlie. Un utente non può essere contemporaneamente sia **Cliente** sia **Tecnico**.

## Strategie di Generalizzazione

### SCENARIO 1 — Tutto nel padre (un'unica entità con un attributo "tipo")

Tutti gli attributi comuni e specifici delle sottocategorie sono contenuti **in un'unica entità** (Utente), con un attributo discriminante (es. tipo = Cliente o Tecnico). Gli attributi specifici di ciascuna sottocategoria vengono **lasciati a NULL** se non rilevanti.

- **Utente**(id, nome, email, **tipo**, saldo, residenza, specializzazione)

#### Vantaggi:

- **Modello semplice:** una sola tabella (più facile da gestire e interrogare).
- **Nessuna JOIN** necessaria per accedere ai dati di un utente.
- Utile se il numero di attributi specifici è limitato e le differenze non sono troppo grandi.

#### Svantaggi:

- Molti **valori NULL** per attributi non applicabili (es. saldo per un tecnico).
- **Integrità più fragile:** nessuna garanzia che gli attributi specifici corrispondano al tipo.
- Poco flessibile se le sottocategorie crescono in numero o complessità.

### SCENARIO 2 — Due entità separate (generalizzazione con entità padre + figlie)

Si crea un'entità padre (Utente) con attributi comuni e due entità figlie (Cliente, Tecnico) con gli attributi specifici. Le entità figlie sono collegate al padre tramite **chiave esterna primaria** (es. stesso id\_user).

- **Utente**(id, nome, email)
- **Cliente**(id\_user, saldo, residenza)
- **Tecnico**(id\_user, specializzazione)

#### Vantaggi:

- **Pulizia e chiarezza semantica:** ogni tabella contiene solo dati rilevanti.
- **Integrità forte:** è chiaro cosa può appartenere a chi.
- **Estendibilità:** facile aggiungere nuovi ruoli (Admin, Venditore, ecc.)
- Adatto per sistemi complessi o con molti dati eterogenei.

#### Svantaggi:

- Richiede **JOIN** per accedere a tutti i dati di un utente.
- Maggiore complessità nella gestione (soprattutto nelle query o interfacce grafiche).
- Un po' più difficile da progettare e mantenere se il sistema è piccolo o molto semplice.

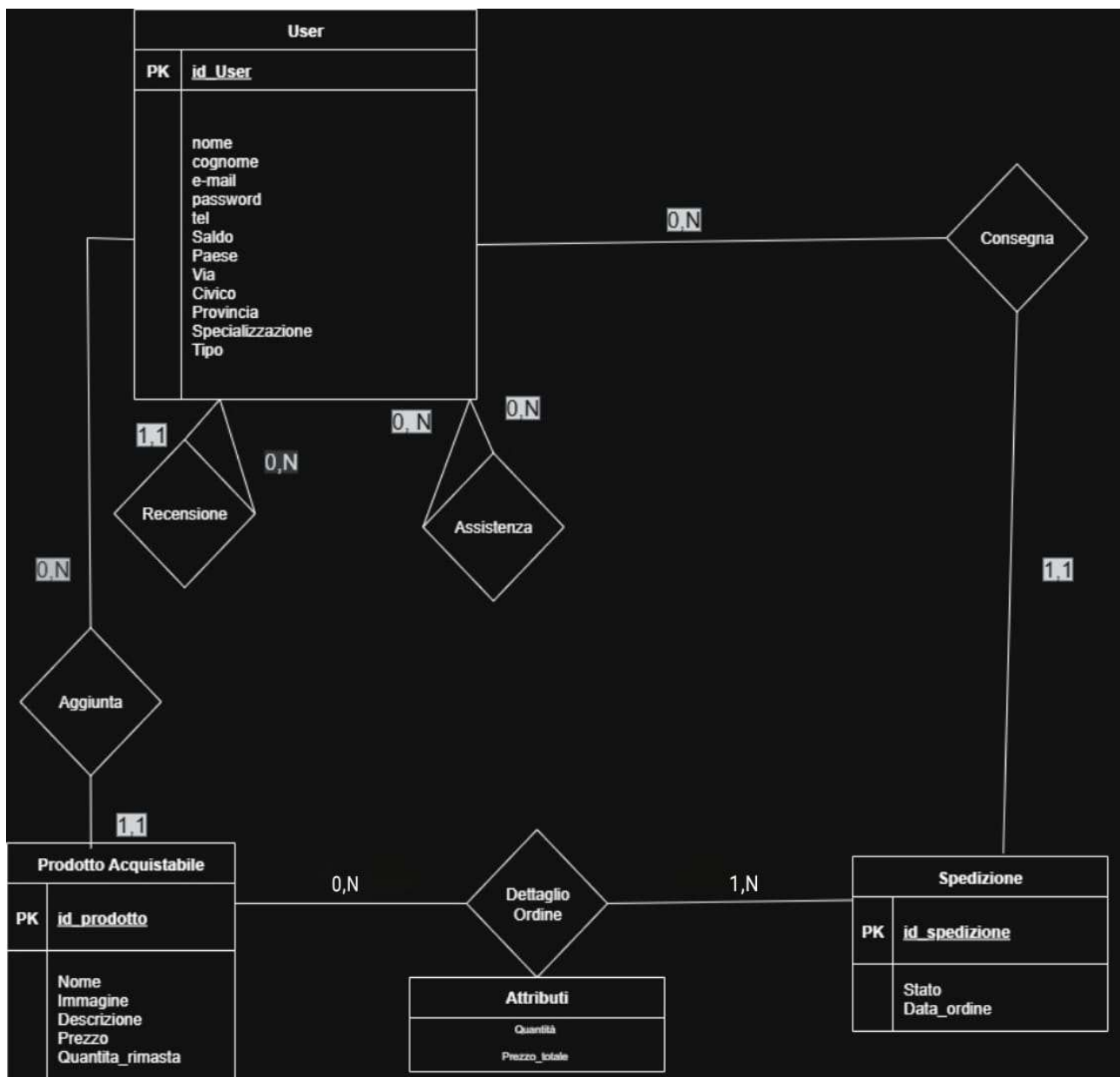
Per la gestione delle due tipologie di utenza (Clienti e Tecnici) **si è scelto di rappresentare entrambi all'interno di un'unica entità Utente, distinguendoli tramite un attributo tipo**. Questa scelta è stata motivata dalla semplicità del sistema e dal numero contenuto di attributi specifici, che rende accettabile la presenza di valori nulli e consente di evitare l'uso di tabelle separate e operazioni di join nelle interrogazioni.

Si noti, la relazione ricorsiva “Recensione” e “Assistenza” derivante dal fatto che un Utente di tipo Cliente può mettersi in relazione con un altro utente di tipo Tecnico. Per risolvere problematiche di incoerenza si è preferito aggiungere quindi una variabile enum in User definita come “tipo”

Stessa cosa per la relazione “Aggiunta”, può essere esaudita solo se tipo = Tecnico

Medesimo discorso per la relazione “Consegna”, può essere esaudita solo se tipo = Cliente

### Diagramma ER Ottimizzato



## Discussione cardinalità

- ***User(0:N) → Recensione ← User (0:N):***

Un User (tipo Cliente) può scrivere più recensioni

Un User (Tipo Tecnico) può ricevere più recensioni

- ***User(0:N) → Assistenza ← User (0:N):***

Un User (tipo Cliente) può fare diverse richieste di assistenza

Un User (Tipo Tecnico) può ricevere più richieste di assistenza

- ***User(0:N) → Aggiunta ← Prodotto Acquistabile(1:1):***

Un User (Tipo Tecnico) può inserire più prodotti sulla piattaforma

Un prodotto deve essere aggiunto da 1 e 1 solo User (Tipo Tecnico)

- ***Prodotto Acquistabile(0:N) → Dettaglio Ordine ← Ordine(1:N):***

Un prodotto può essere selezionato per più ordini

Un ordine deve avere almeno 1 prodotto

- ***Ordine(1:1) → Consegna ← User(0:N):***

Un ordine deve essere fatto da 1 e 1 solo User (Tipo Cliente)

Un User (Tipo Cliente) può fare più ordini

## Modello logico

**User** (id\_User, nome, cognome, email (UNIQUE), password, tel (UNIQUE), tipo (Cliente, Tecnico), saldo (NULLABLE, Cliente), Residenza (NULLABLE, Cliente), specializzazione (NULLABLE, Tecnico))

- L'entità User contiene le informazioni di profilazione delle due entità figlie: Cliente e Tecnico. L'attributo e-mail è considerabile unico per non provocare problemi di

identificazione. Gli attributi “**Saldo e Residenza**” sono considerabili **NULLABLE** per gli **User Tecnici**. L’attributo “**Specializzazione**” è considerato **NULLABLE** per gli **User Cliente**

**Recensione**(id\_cliente, id\_tecnico, voto, descrizione)

- La tabella Recensione registra eventuali commenti o valutazioni espressi dal cliente nei confronti di un tecnico, a seguito dell’esecuzione di un intervento di assistenza. Essa consente di monitorare la qualità del servizio offerto e raccogliere feedback utili al miglioramento.

**Assistenza**(id\_cliente, tecnico, Stato, Data, Intestazione, Descrizione)

- L’entità *Assistenza* contiene tutte le informazioni relative a una richiesta di supporto tecnico inoltrata da un cliente. Ogni richiesta è associata a un tecnico incaricato e include dettagli quali la data, la descrizione del problema e lo stato dell’intervento.

**Prodotto Acquistabile** (id\_prodotto, Nome, Immagine, Descrizione, Prezzo, Quantita\_rimasta, id\_User: User)

- L’entità *Prodotto* rappresenta i beni vendibili tramite la piattaforma. I prodotti possono essere inseriti e gestiti esclusivamente da utenti con ruolo tecnico (*User Tecnico*), i quali ne definiscono le caratteristiche principali (nome, descrizione, prezzo, disponibilità, ecc.).

**Dettaglio Ordine** (id\_prodotto, id\_ordine, Quantità, PrezzoQuantità)

- Questa entità mantiene traccia della selezione di uno o più prodotti da parte dell’utente, inclusa la quantità desiderata. È funzionale alla costruzione dell’ordine e alla successiva generazione del documento di vendita.

**Ordine** (id\_ordine, Stato, Data\_ordine, Prezzo\_totale, id\_user:User)

- L’entità *Ordine* raccoglie l’insieme dei prodotti selezionati da un cliente, con l’obiettivo di finalizzare la transazione commerciale. Include informazioni come l’identificativo dell’utente, la data dell’ordine, il totale dell’importo e lo stato della lavorazione.



**Entità e Vincoli****User**

Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
<b>Id_User</b>	Identificativo utente	INT		PK, AUTO_INCREMENT
<b>Nome</b>	Nome utente	VARCHAT	30	NOT NULL
<b>Cognome</b>	Cognome Utente	VARCHAR	30	NOT NULL
<b>Email</b>	Indirizzo email per l'accesso	VARCHAR	50	NOT NULL, UNIQUE
<b>Password</b>	Parola segreta per accesso	VARCHAR	255	NOT NULL
<b>Tel</b>	Numero di telefono	VARCHAR	15	UNIQUE
<b>Tipo</b>	Cliente/Tecnico	ENUM		NOT NULL, scelta = 'Cliente' o 'Tecnico'
<b>Residenza</b>	Indirizzo per la spedizione	VARCHAR	100	NULLABLE
<b>Saldo</b>	Disponibilità economica per acquisti	INT	255	NULLABLE
<b>Specializzazione</b>	Qualità operative	VARCHAR	50	NULLABLE

Vincoli di unicità:

- email è **UNIQUE**
- tel è **UNIQUE**

Vincoli di dominio:

- **tipo** ∈ {'Cliente', 'Tecnico'}
- **saldo e residenza** possono essere **NULL** solo se tipo = 'Tecnico'
- **specializzazione** può essere **NULL** solo se tipo = 'Cliente'

## Recensione

Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
<b>Id_cliente</b>	Id di chi invia la recensione	INT		PK, FK → User:id_user,
<b>Id_tecnico</b>	Id di chi invia la recensione	INT		PK, FK → User:id_user
<b>Voto</b>	Valutazione in base 5 dell'assistenza ricevuta	INT	2	NOT NULL, Valori ammessi = da 1 a 5
<b>Descrizione</b>	Motivazione valutazione	TEXT		NULLABLE

Vincoli di integrità referenziale:

- **id\_cliente** → User(id\_User) con tipo = 'Cliente'
- **id\_tecnico** → User(id\_User) con tipo = 'Tecnico'

Vincoli di dominio:

- **voto** deve essere **compreso tra 1 e 5**

**Assistenza**

Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
<b>Id_cliente</b>	Id di chi richiede	INT		PK, FK → User:id_User
<b>Id_tecnico</b>	Id di chi accoglie	INT		PK, FK → User:id_User
<b>StatoAssistenza</b>	Breve descrizione temporale dell'operazione	ENUM		NOT NULL, scelta 'accolto' o 'in lavorazione' o 'da ritirare'
<b>Data</b>	Motivazione valutazione	DATE		NOT NULL
<b>Intestazione</b>	parte iniziale, precede il corpo del messaggio e contiene informazioni	TEXT		NOT NULL
<b>Descrizione</b>	Corpo del messaggio	TEXT		NULLABLE

Vincoli di integrità referenziale:

- **id\_cliente** → User(id\_User) con tipo = 'Cliente'
- **id\_tecnico** → User(id\_User) con tipo = 'Tecnico'

Vincoli di dominio:

- **Stato** ∈ {'In attesa', 'In corso', 'Completata', 'Annullata'}

**Prodotto Acquistabile**

Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
<b>Id_prodotto</b>	Identificativo prodotto	INT		PK
<b>Nome</b>	Nome del prodotto	VARCHAR		NOT NULL
<b>Immagine</b>	File per la rappresentazione grafica del prodotto	IMAGE		NULLABLE
<b>Descrizione</b>	Motivazione valutazione	VARCHAR		NOT NULL
<b>Prezzo</b>	Costo in euro del prodotto	FLOAT		NOT NULL, valori ammessi $\geq 0$
<b>Quantita Rimasta</b>	Numero di prodotti rimasti in magazzino	INT		NOT NULL, valori ammessi $\geq 0$
<b>Id_Tecnico</b>	Identificativo del tecnico che ha aggiunto il prodotto	INT		FK $\rightarrow$ User: id_User

Vincoli di integrità referenziale:

- **id\_User  $\rightarrow$  User(id\_User) (deve essere un Tecnico)**

Vincoli di dominio:

- **Prezzo  $\geq 0$**
- **Quantita\_rimasta  $\geq 0$**

## Dettaglio Ordine

Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
<b>Id_prodotto</b>	Identificativo prodotto acquistabile	INT		PK, FK → Prodotto Acquistabile: Id_prodotto
<b>Id_Ordine</b>	Identificativo ordine fatto dal cliente	INT		PK, FK → Ordine:Id_ordine
<b>Quantità</b>	Numero di prodotti selezionati	INT		NOT NULL, valori ammessi > 0
<b>QuantitaPrezzo</b>	Prezzo Caricato nell'ordine in riferimento ai prodotti selezionati	FLOAT		NOT NULL, valori ammessi >= 0

Vincoli di integrità referenziale:

- **id\_prodotto → Prodotto\_Acquistabile(id\_prodotto)**
- **id\_ordine → Ordine(id\_ordine)**

Vincoli di dominio:

- **Quantità > 0**
- **PrezzoQuantità ≥ 0**

## Ordine

Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
<b>Id_ordine</b>	Identificativo spedizione	INT		PK
<b>Stato_Ordine</b>	Breve descrizione temporale dell'ordine	ENUM		NOT NULL, scelta 'accolto' o 'in lavorazione' o 'da ritirare'
<b>Data_Ordine</b>	Data conferma ordine	DATE		NOT NULL
<b>PrezzoTotale</b>	Sommatoria prezzi prodotti	TEXT		NOT NULL, valori ammessi $\geq 0$
<b>Id_Cliente</b>	Identificativo cliente che esegue ordine	INT		FK → User:Id_User

Vincoli di integrità referenziale:

- **id\_user → User(id\_User) (deve essere un Cliente)**

Vincoli di dominio:

- **Stato** ∈ {'In attesa', 'Pagato', 'Spedito', 'Consegnato', 'Annullato'}
- **Prezzo\_totale** ≥ 0

## Vincoli Interrelazionali

Nel sistema, la **tabella Recensione** rappresenta l'azione con cui un utente valuta un altro.

Per garantire la coerenza semantica dei dati, è necessario che:

- Il campo **id\_cliente** della tabella Recensione faccia riferimento a un utente registrato nella tabella Utente che abbia il campo tipo impostato a '**Cliente**'.
- Il campo **id\_tecnico**, anch'esso riferito alla tabella Utente, debba corrispondere a un utente con tipo = '**Tecnico**'.
- Inoltre, una recensione può essere inserita solo se tra il cliente e il tecnico coinvolti esiste una relazione pregressa nella tabella Assistenza, e questa assistenza risulta **completata**. Ciò significa che deve esistere una riga nella tabella Assistenza in cui **id\_cliente** e **id\_tecnico** coincidono con quelli della recensione e il campo **stato** è pari a 'Completata'.

La **tabella Assistenza** registra una richiesta di supporto da parte di un cliente a un tecnico. Anche in questo caso, è fondamentale rispettare i ruoli degli utenti:

- Il campo `id_cliente` deve riferirsi a un utente della tabella `Utente` con tipo uguale a **'Cliente'**, mentre
- Il campo `id_tecnico` deve puntare a un utente con tipo uguale a **'Tecnico'**.

Questo vincolo impedisce che utenti con ruoli non previsti possano avviare o ricevere richieste di assistenza, mantenendo la coerenza funzionale del sistema.

La **tabella Prodotto** (o eventualmente una tabella relazionale come `AggiuntaProdotto`) rappresenta i prodotti inseriti nel sistema. Anche qui si applica una logica basata sui ruoli:

- Solo un utente con tipo = **'Tecnico'** è autorizzato ad aggiungere un prodotto. Pertanto, il campo `id_tecnico` che indica chi ha inserito il prodotto deve riferirsi a un utente tecnico nella tabella `Utente`.

Questo riflette l'organizzazione interna del sistema, dove solo i tecnici hanno il compito di gestire il catalogo prodotti.

Infine, la **tabella Consegna** o `Ordine` si riferisce al processo di ricezione di un prodotto da parte di un utente. In questo caso:

- Il campo `id_cliente`, che indica il destinatario della consegna, deve sempre fare riferimento a un utente con tipo = **'Cliente'**.

Questo vincolo garantisce che solo i clienti finali possano effettuare ordini e ricevere prodotti, escludendo altre tipologie di utenti da questo tipo di operazione.

## Implementazione del Sistema Informativo

L'implementazione del sistema informativo è stata realizzata utilizzando il framework Django, scelto per la sua capacità di supportare lo sviluppo di applicazioni web scalabili, sicure e facilmente manutenibili.

Il sistema è progettato per gestire i dati secondo un modello logico predefinito e per offrire accesso a un insieme di funzionalità chiave.

### Funzionalità Principali dell'Applicazione

L'applicazione supporta molteplici funzionalità principali, selezionate in base alle esigenze commerciali e organizzative, garantendo una gestione efficace e intuitiva dell'intero sistema.

#### 1. Schermate di Accesso Differenziato

Accesso separato per le diverse tipologie di utenti, con interfacce personalizzate:

- Clienti: *accedono a funzionalità legate all'acquisto e all'assistenza.*
- Tecnici: *accedono a strumenti di gestione, supporto e analisi.*

## 2. Sezione Cliente

- Visualizzazione del catalogo prodotti  
*Permette di esplorare l'offerta disponibile, con dettagli tecnici e immagini dei prodotti.*
- Aggiunta prodotti al carrello  
*Consente di selezionare i prodotti desiderati e prepararli per l'acquisto o la richiesta.*
- Richiesta di assistenza tecnica  
*Modulo per inviare segnalazioni o richieste di supporto tecnico.*
- Visualizzazione del profilo personale  
*Accesso ai dati personali.*
- Accesso e gestione del carrello  
*Visualizzazione, modifica e conferma dei prodotti selezionati.*

## 3. Sezione Tecnico

- Visualizzazione delle richieste di assistenza  
*Elenco delle richieste ricevute dai clienti, con possibilità di gestione e aggiornamento dello stato.*
- Aggiunta di nuovi prodotti al catalogo  
*Inserimento di nuovi articoli con descrizioni, immagini e specifiche tecniche.*
- Gestione completa del catalogo prodotti  
*Modifica, aggiornamento o eliminazione dei prodotti esistenti.*
- Elenco clienti con funzionalità di  
*Ricerca avanzata per:*
  - Nome
  - Cognome
  - Email
- Sezione statistiche per l'analisi delle attività  
*Dashboard con dati per monitorare l'andamento delle richieste, l'attività dei clienti e l'efficienza del sistema.*



## Selezione Login

Permette la selezione per l'area dedicata

The screenshot shows the login page for 'Elettronica Iovine'. At the top is a blue header with the company name. Below it, a grey box contains the following text: 'Benvenuto nel nostro portale', a paragraph about the company's 20-year history and services, another paragraph about their specialization in repairs and technical assistance, and a prompt to 'Seleziona il tuo profilo per accedere ai servizi'. At the bottom of this box are two blue buttons labeled 'Cliente' and 'Tecnico'.

Comprende:

- Due tasti di selezione per l'accesso ai servizi
- Breve descrizione sull'attività

## Accesso Cliente

The screenshot shows the 'Accesso Cliente' login form. It has a title 'Accesso Cliente' and a subtitle 'Accedi per gestire le tue richieste di assistenza'. There are two input fields: 'Email' with the value 'mario.rossi@email.it' and 'Password' with masked characters. Below the fields is a blue 'Accedi' button. At the bottom, there are two links: 'Torna alla homepage' and 'Password dimenticata?'.

Comprende:

- form per inserimento mail e password (hashate con SHA256)
- tasto di accesso

## Dashboard Cliente

**Elettronica Iovine**Benvenuto, Mario RossiCarrelloProfiloLogout

### Dashboard Cliente

Benvenuto nella tua area personale. Da qui puoi gestire le tue richieste di assistenza e acquistare prodotti.

### Catalogo Prodotti

Visualizza il nostro catalogo completo di prodotti disponibili per l'acquisto.

[Vai al catalogo](#)

### Le tue richieste

Non hai ancora richieste di assistenza attive.

### Nuova richiesta

Clicca sul pulsante sottostante per aprire una nuova richiesta di assistenza.

[Nuova richiesta](#)

### Contiene:

- Tasto per accedere al carrello
- Tasto per accedere alle informazioni del profilo
- Tasto per effettuare il logout
- Tasto per accedere al catalogo clienti
- Tasto per creazione nuova richiesta
- Form di visualizzazione richieste create


## Catalogo prodotti


**Elettronica Iovine**Benvenuto, Mario RossiLogout


[Torna alla dashboard](#)


### Catalogo Prodotti

Cerca prodotti...

**PlayStation 5 Slim**  
  
**€ 499.00**  
Console Sony di ultima generazione, data di uscita 2020.  
**Disponibile**  
[Aggiungi all'ordine](#)

**Tv Sony 55"**  
  
**€ 649.00**  
Tv Sony QLED da 55 pollici.  
**Disponibile**  
[Aggiungi all'ordine](#)

**Meta Quest 3**  
  
**€ 699.00**  
Visore realtà aumentata, ultima generazione.  
**Disponibile**  
[Aggiungi all'ordine](#)

**Lenovo IdeaPad IP 3 Notebook**  
  
**€ 449.00**  
Modello del processore i5-12450H, RAM installata 16 GB, Capacità SSD 512 GB. Sistema operativo Windows 11 Home.  
**Disponibile**  
[Aggiungi all'ordine](#)

## Catalogo Prodotti

Lenovo

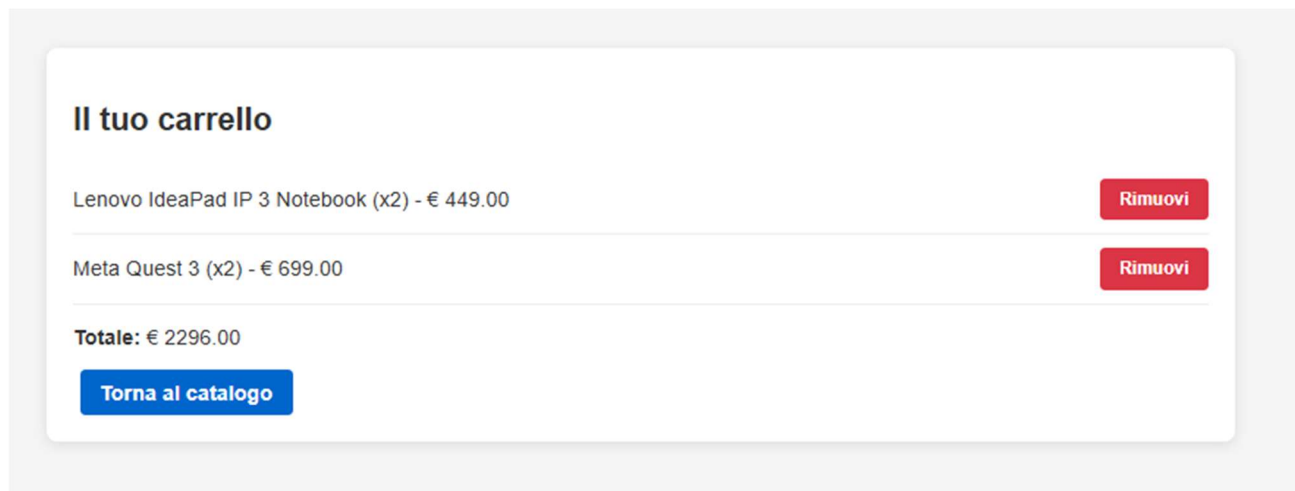
### Lenovo IdeaPad IP 3 Notebook

  
**€ 449.00**  
Modello del processore i5-12450H, RAM installata 16 GB, Capacità SSD 512 GB. Sistema operativo Windows 11 Home.  
**Disponibile**  
[Aggiungi all'ordine](#)

### Contiene:

- ricerca per nome
- Tasto per tornare alla dashboard
- Tasto per l'aggiunta del prodotto all'ordine

## Carrello



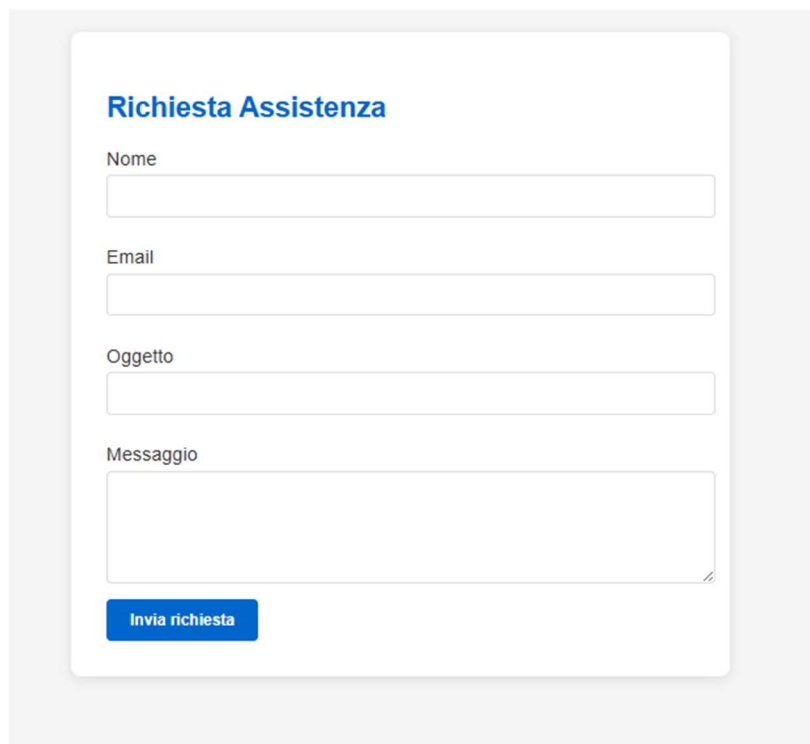
**Il tuo carrello**

Lenovo IdeaPad IP 3 Notebook (x2) - € 449.00	<a href="#">Rimuovi</a>
Meta Quest 3 (x2) - € 699.00	<a href="#">Rimuovi</a>
<b>Totale:</b> € 2296.00	
<a href="#">Torna al catalogo</a>	

Contiene:

- Possibilità di rimozione di prodotto dal carrello

## Richiesta Assistenza



**Richiesta Assistenza**

Nome

Email

Oggetto

Messaggio

[Invia richiesta](#)

Contiene:

- Form di inserimento informazioni personali
- Casella oggetto e testo per inserire le informazioni

## Informazioni Personali

**Profilo Cliente**[Torna alla Dashboard](#)

**Nome:** Mario

**Cognome:** Rossi

**Email:** mario.rossi@email.it

**Saldo:** € 100.00

**Residenza:** Via Roma 1, Milano

Contiene:

- Informazioni personali come:
  - Nome
  - Cognome
  - Email
  - Saldo
  - Residenza

## Accesso Tecnico

**Area Riservata Tecnici**

**Email:**

**Password:**

[Accedi all'area tecnica](#)

[Torna alla homepage](#)

Comprende:

- form per inserimento mail e password (hashate con SHA256)
- tasto di accesso

## Dashboard Tecnico

The screenshot shows the 'Dashboard Tecnico' for 'Elettronica Iovine'. The header is dark blue with the company name on the left and a welcome message 'Benvenuto, Paolo Esposito' with a 'Logout' button on the right. The main content area is light gray and contains several white cards. The first card is the 'Dashboard Tecnico' overview, followed by 'Richieste di Assistenza' (with buttons for 'Tutte le richieste', 'Nuove richieste', and 'In lavorazione'), 'Gestione Prodotti' (with buttons for 'Aggiungi prodotto' and 'Catalogo prodotti'), 'Clienti' (with an 'Elenco clienti' button), and 'Recensioni' (with a 'Tutte le recensioni' button). At the bottom, a 'Statistiche' card shows a summary of recent activities: 'Richieste in attesa: 0', 'Richieste in lavorazione: 0', 'Prodotti gestiti: 0', and 'Valutazione media: N/A'.

Contiene:

- Tasto richieste di Assistenza
- Tasto elenco Clienti
- Tasto aggiunta prodotto
- Tasto visualizzazione e modifica catalogo prodotti
- Tasto elenco recensioni
- Sezione Statistiche
- Elenco clienti

## Sezione Recensioni

The screenshot shows the 'Recensioni Ricevute' section. The header is blue with the title on the left and a yellow 'Torna alla Dashboard' button on the right. The main content area is light gray and features a white card titled 'Le tue recensioni' which displays the message 'No recensioni :P'.

## Elenco Clienti

**Elettronica Iovine**Benvenuto, Paolo EspositoLogout

[← Torna alla Dashboard](#)

### Elenco Clienti

Lista completa di tutti i clienti registrati nel sistema.

Cerca cliente per nome, cognome o email... Cerca

Nome	Cognome	Email	Residenza	Saldo	Azioni
Mario	Rossi	mario.rossi@email.it	Via Roma 1, Milano	€ 100.00	<a href="#">Dettagli</a>
Luigi	Bianchi	luigi.bianchi@email.it	Via Garibaldi 2, Roma	€ 150.00	<a href="#">Dettagli</a>
Anna	Verdi	anna.verdi@email.it	Via Napoli 3, Torino	€ 200.00	<a href="#">Dettagli</a>

« 1 2 3 »

Lista completa di tutti i clienti registrati nel sistema.

anna

Nome	Cognome	Email	Residenza	Saldo
Anna	Verdi	anna.verdi@email.it	Via Napoli 3, Torino	€ 200.0

« 1 2 3 »

## Contiene:

- Le varie informazioni dei clienti
- Casella di ricerca per:
  - nome
  - cognome
  - email

## Richieste Assistenza

**Richieste di Assistenza Ricevute**[Torna alla Dashboard](#)

### Le tue richieste di assistenza

Nessuna Richiesta!!!

## Aggiunta Prodotto

**Elettronica Iovine**

DashboardLogout

### Aggiungi Nuovo Prodotto

Nome Prodotto\*

Prezzo (€)\*

Quantità Disponibile\*

Descrizione Prodotto\*

Immagine Prodotto

Scegli fileNessun file selezionato

Formati supportati: JPG, PNG, GIF. Dimensione massima: 5MB

AnnullaSalva Prodotto

Contiene:

- Selezione aggiunta:
  - Nome Prodotto
  - Prezzo
  - Quantità disponibile
  - Descrizione Prodotto
  - Immagine Prodotto


## Catalogo prodotti

[← Torna alla Dashboard](#)


### Catalogo Prodotti

Visualizza e modifica i prodotti disponibili nel catalogo.


Cerca prodotto per nome, marca o categoria...Cerca




**PlayStation 5 Slim**  
Quantità: 7  
Console Sony di ultima generazione, data di uscita 2020  
**€ 499.00**  
Disponibile (7 in magazzino)  
[Elimina](#)



**Tv Sony 55"**  
Quantità: 2  
Tv Sony QHD da 55 pollici  
**€ 649.00**  
Disponibile (2 in magazzino)  
[Elimina](#)



**Meta Quest 3**  
Quantità: 2  
Visore realtà aumentata, ultima generazione  
**€ 699.00**  
Disponibile (2 in magazzino)  
[Elimina](#)



pag. 23

Analoga alla sezione in Cliente ma con possibilità di vedere la quantità rimanente e possibilità di eliminare il prodotto.

## Gestione delle Sessioni

La gestione delle sessioni rappresenta un aspetto fondamentale del sistema informativo sviluppato per il progetto **Elettronicalovine**, in quanto consente di mantenere lo stato dell'utente autenticato durante l'interazione con l'applicazione web, garantendo al contempo sicurezza, personalizzazione e controllo degli accessi.

### 1. Obiettivi della Gestione delle Sessioni

- Identificare in modo univoco l'utente autenticato (cliente o tecnico)
- Mantenere lo stato dell'utente durante la navigazione
- Gestire i permessi e l'accesso alle funzionalità in base al ruolo
- Conservare dati temporanei come il contenuto del carrello

## 2. Implementazione Tecnica

### 2.1 Configurazione Django

Nel file settings.py, la gestione delle sessioni è abilitata tramite il middleware SessionMiddleware e l'applicazione django.contrib.sessions. Le sessioni sono memorizzate nel database predefinito (SQLite) e associate a ciascun utente tramite un identificatore univoco salvato in un cookie.

### 2.2 Login e Identificazione Utente

Nel file views.py, le funzioni cliente\_login e tecnico\_login gestiscono l'autenticazione. In caso di successo, vengono salvati nella sessione i seguenti dati:

```
request.session['user_id'] = utente.id_utente
```

```
request.session['user_type'] = 'Cliente' # oppure 'Tecnico'
```

Queste informazioni vengono poi utilizzate per:

- Verificare il tipo di utente (is\_cliente, is\_tecnico)
- Proteggere le viste tramite decoratori personalizzati (@cliente\_required, @tecnico\_required)
- Personalizzare l'interfaccia utente

### 2.3 Persistenza del Carrello

Il contenuto del carrello viene gestito tramite la sessione:

```
carrello = request.session.get('carrello', {})
```

```
request.session['carrello'] = carrello
```



### 3. Sicurezza delle Sessioni

- I template cliente\_login.html e tecnico\_login.html includono il token CSRF ({{% csrf\_token %}}), proteggendo l'applicazione da attacchi di tipo Cross-Site Request Forgery.
- I cookie di sessione sono protetti da default con l'opzione HttpOnly, impedendo l'accesso da script lato client.

### 4. Logout e Pulizia della Sessione

Per garantire la corretta terminazione della sessione e la protezione dei dati utente, è stata implementata una funzione di logout. Essa consente di eliminare tutte le informazioni salvate nella sessione corrente:

```
def logout_tecnico(request):  
    request.session.flush()  
    return redirect('tecnico_login')  
  
-----  
def logout_cliente(request):  
    request.session.flush()  
    return redirect('cliente_login')
```

### Simulazione di SQL Injection

#### Scenario di vulnerabilità

Si consideri un'applicazione web che gestisce l'autenticazione degli utenti attraverso la costruzione manuale di una query SQL, come mostrato nel seguente esempio:

```
username = request.POST['username']  
password = request.POST['password']  
query = f"SELECT * FROM utenti WHERE username = '{username}' AND password = '{password}'"
```

In questo caso, i dati forniti dall'utente sono concatenati direttamente all'interno della query SQL, senza alcuna forma di controllo o validazione.

#### Dinamica dell'attacco

Un utente malintenzionato può sfruttare questa vulnerabilità fornendo un input costruito per alterare la logica della query SQL ed eludere il sistema di autenticazione.

**Esempio di input:**

- **Username:** admin' OR '1'='1
- **Password:** qualsiasi\_valore

Query risultante: `SELECT * FROM utenti WHERE username = 'admin' OR '1'='1' AND password = 'qualsiasi_valore'`

Poiché l'espressione '1'='1' risulta sempre vera, l'intera condizione della clausola WHERE può essere soddisfatta, anche senza credenziali valide, consentendo l'accesso non autorizzato.

**Vulnerabilità introdotte****1. Concatenazione diretta dell'input utente:**

L'input viene inserito direttamente nella stringa SQL senza validazione, rendendo possibile l'iniezione di codice arbitrario.

**2. Nessuna separazione tra dati e logica SQL:**

L'interprete SQL non distingue tra dati e comandi, permettendo che istruzioni inserite dall'utente vengano interpretate come parte del codice.

**3. Alterazione del flusso di controllo della query:**

È possibile manipolare la logica condizionale (tramite operatori come OR, AND, o commenti --) per eludere i controlli di autenticazione o accedere ad altri dati.

Come sarebbe il codice VULNERABILE

Un esempio vulnerabile sarebbe costruire la query manualmente concatenando le stringhe:

```
from django.db import connection
```

```
def login_vulnerabile(request):
```

```
    email = request.POST.get('email')
```

```
    password = request.POST.get('password')
```

```
    query = f"SELECT * FROM Utente WHERE email = '{email}' AND password = '{password}'"
```

```
    with connection.cursor() as cursor:
```

```
        cursor.execute(query)
```

```
        utente = cursor.fetchone()
```

In questo esempio, un attaccante potrebbe fornire un'email del tipo: ' OR '1'='1 e accedere senza conoscere le credenziali corrette. Il problema risiede nella concatenazione diretta dell'input, che consente di iniettare codice SQL arbitrario all'interno della query.

## Prevenzione della SQL Injection

### 1. Utilizzo dell'ORM Django

Django mette a disposizione un Object Relational Mapper (ORM) che consente di interagire con il database in modo sicuro, evitando la scrittura diretta di query SQL:

try:

```
user = Utente.objects.get(username=username, password=password)
```

except Utente.DoesNotExist:

```
user = None
```

- L'ORM si occupa automaticamente della sanitizzazione degli input.
- I dati vengono separati dal codice SQL e non possono modificarne la struttura.
- Qualsiasi input, anche se manipolato, viene interpretato come semplice valore da confrontare.

### 2. Utilizzo di query parametrizzate

Qualora si renda necessario scrivere query SQL dirette, è fondamentale utilizzare query parametrizzate per garantire la separazione tra codice e dati.

```
from django.db import connection
```

```
with connection.cursor() as cursor:
```

```
cursor.execute("SELECT * FROM Utente WHERE email = %s AND password = %s", [email, password])
```

```
utente = cursor.fetchone()
```

#### Motivazioni della sicurezza:

- I parametri (%s) sono trattati separatamente dal codice SQL.
- L'adapter del database si occupa di gestire correttamente i valori forniti.
- Qualsiasi input utente viene neutralizzato come valore e non può modificare la logica della query.

## Analisi delle vulnerabilità tramite SQLmap

Per verificare la presenza effettiva di una vulnerabilità di tipo SQL Injection, è possibile avvalersi di strumenti di test automatizzati. Uno dei più diffusi e completi è **SQLmap**, un framework open-source progettato per individuare ed eventualmente sfruttare automaticamente iniezioni SQL nei parametri delle applicazioni web.

SQLmap consente di condurre test su input sospetti e, qualora venga rilevata una vulnerabilità, fornisce funzionalità avanzate per accedere, estrarre o manipolare i dati presenti nel database.

### Funzionalità principali offerte da SQLmap:

- **Rilevamento automatico dell'iniezione:** tramite tecniche come boolean-based, time-based, error-based, tra le altre.
- **Estrazione dei dati (data dumping):** tabelle, righe, colonne e credenziali possono essere recuperate qualora l'iniezione sia confermata.
- **Esecuzione di operazioni avanzate:** quali identificazione del tipo e versione del DBMS, lettura di file sul server (se consentito), e persino apertura di shell se i privilegi lo permettono.

## Esecuzione di un test con SQLmap

Si consideri un'applicazione web che gestisce l'autenticazione degli utenti mediante richiesta HTTP POST, simile alla seguente:

OST /login HTTP/1.1

Host: localhost:8000

Content-Type: application/x-www-form-urlencoded

username=admin&password=1234

Questa richiesta può essere intercettata e salvata in un file di testo, ad esempio login\_request.txt. Una volta acquisita, SQLmap può essere eseguito con il seguente comando: `sqlmap -r login_request.txt --level=5 --risk=3 --dump`

### Parametri principali del comando:

- -r: indica il file contenente la richiesta HTTP completa.
- --level=5: definisce la profondità dell'analisi (più è alto, maggiori sono i payload provati).
- --risk=3: specifica il livello di impatto delle tecniche da utilizzare (da 1 a 3).
- --dump: richiede l'estrazione dei dati qualora la vulnerabilità sia confermata.

### Esito atteso

Nel caso in cui il sistema sia effettivamente vulnerabile, SQLmap è in grado di:

- Confermare l'esistenza della falla
- Visualizzare i nomi delle tabelle e i dati contenuti nel database
- Fornire un'interfaccia per ulteriori interazioni con il database compromesso

In contesto accademico o di test, SQLmap rappresenta un valido supporto per verificare la robustezza delle applicazioni e valutare l'efficacia delle contromisure implementate, come l'uso di ORM o query parametrizzate.

Le SQL Injection costituiscono una delle vulnerabilità più pericolose per le applicazioni che interagiscono con un database. La scrittura manuale di query SQL attraverso la concatenazione di input è fortemente sconsigliata. L'adozione delle pratiche sovraccitate garantiscono che i dati forniti dagli utenti siano gestiti in modo sicuro.

### Difesa contro attacchi a forza bruta

Per prevenire tentativi di accesso non autorizzati tramite **brute-force**, è buona pratica limitare i tentativi consecutivi di login. Una contromisura efficace consiste nel **bloccare temporaneamente l'accesso dopo 5 tentativi falliti**, ad esempio per **2 minuti**.

Questo blocco può essere applicato per IP o per utente, e riduce drasticamente la possibilità di indovinare credenziali tramite script automatizzati.

```
< /cliente/
def cliente_login(request): 1 usage new *
    max_tentativi = 5
    blocco_minuti = 2
    blocco_fino = request.session.get('blocco_cliente', 0)

    if time.time() < blocco_fino:

        secondi_restanti = int(blocco_fino - time.time())
        messages.error(request, message: f'Troppi tentativi falliti. Riprova tra {secondi_restanti} secondi.')
        return render(request, template_name: 'Cliente/cliente_login.html')

request.session['tentativi_cliente'] = 0
request.session['blocco_cliente'] = 0

if request.method == 'POST':
    tentativi = request.session.get('tentativi_cliente', 0)
    email = request.POST.get('email')
    password = request.POST.get('password')
    password_hash = hash_password_sha256(password)
    try:
        utente = Utente.objects.get(email=email, password=password_hash, tipo='Cliente')
        request.session['user_id'] = utente.id_utente
        request.session['user_type'] = 'Cliente'
        request.session['tentativi_cliente'] = 0
        request.session['blocco_cliente'] = 0
        return redirect(request.GET.get('next', 'homepage_cliente'))
    except Utente.DoesNotExist:
        tentativi += 1
        request.session['tentativi_cliente'] = tentativi
        if tentativi >= max_tentativi:
            request.session['blocco_cliente'] = time.time() + blocco_minuti * 60
            messages.error(request, message: 'Troppi tentativi falliti. Attendi 2 minuti prima di riprovare.')
        else:
            messages.error(request, message: f'Credenziali non valide. Tentativo {tentativi} di {max_tentativi}.')
        return render(request, template_name: 'Cliente/cliente_login.html')
return render(request, template_name: 'Cliente/cliente_login.html')
```