

## Traccia d'Esame - Progetto di un Sistema Informativo per la Clinica Privata *ReViva*

Il presente lavoro prevede la progettazione e la realizzazione di un sistema informativo completo e integrato per la Clinica Privata «*ReViva*», una struttura sanitaria specializzata in servizi di riabilitazione e in ricoveri per interventi chirurgici programmati. L'iniziativa si inserisce in un più ampio processo di digitalizzazione e modernizzazione gestionale, con l'obiettivo di migliorare l'efficienza operativa, la qualità dell'assistenza e la sicurezza nella gestione dei dati clinici.

L'obiettivo principale è quello di sostituire le attuali procedure manuali con un'applicazione web intuitiva, responsive e sicura, con lo scopo di supportare il personale medico e sanitario nella gestione quotidiana delle attività cliniche e amministrative. La piattaforma è pensata per ottimizzare il monitoraggio del soggiorno dei pazienti, la programmazione dei trattamenti, nonché l'esecuzione dei piani terapeutici e riabilitativi.

Il cuore dell'applicazione sarà costituito da un database relazionale (SQL), progettato per garantire organizzazione, coerenza e tracciabilità dei dati sanitari. Il database sarà responsabile della gestione di tutte le informazioni essenziali relative a:

- Anagrafica dei pazienti;
- Cartelle cliniche digitali;
- Prenotazioni di ricoveri e trattamenti;
- Personale sanitario (medici e operatori);
- Eventi formativi e informativi della clinica;
- Recensioni e valutazioni degli utenti.

Il sistema prevede una gestione granulare del personale della clinica e degli assistiti:

- **Medici:** potranno accedere a funzionalità avanzate come la redazione delle cartelle cliniche, la supervisione dei ricoveri;
- **Operatori sanitari:** disporranno di strumenti per l'assistenza ai pazienti a seconda del trattamento scelto;
- **Pazienti:** una volta registrati nel sistema, avranno una cartella clinica personale, nella quale saranno documentate le diagnosi, le terapie prescritte, gli interventi subiti e il medico responsabile del caso.

Per migliorare l'interazione con i pazienti, l'applicazione web integrerà uno spazio dedicato alle opinioni, commenti e valutazioni da parte degli utenti. Al termine di ciascun trattamento o soggiorno, i pazienti potranno esprimere un giudizio sulla qualità del servizio ricevuto, contribuendo così a un sistema di feedback trasparente e costruttivo per il miglioramento continuo delle prestazioni sanitarie.

La struttura «*ReViva*», oltre all'attività clinica, organizza regolarmente:

- Eventi informativi e giornate aperte al pubblico (screening, prevenzione);
- Seminari e incontri con specialisti;
- Corsi di formazione e aggiornamento per il personale medico e sanitario.

Il sistema permetterà di gestire queste iniziative in modo completo: pubblicazione degli eventi, raccolta iscrizioni online, gestione delle presenze effettive e archiviazione delle informazioni per finalità organizzative e formative.

## Obiettivi del Progetto

Il progetto prevede lo sviluppo di un sistema informativo per la clinica privata, con l'obiettivo di supportare la trasformazione digitale delle attività sanitarie e gestionali.

### 1. Gestione dei profili dei pazienti

Ogni paziente disporrà di un profilo personale nel sistema, contenente:

- Informazioni anagrafiche (nome, cognome, codice fiscale, data di nascita, cellulare e indirizzo);
- Dati clinici rilevanti (patologie pregresse, allergie, terapie in corso);
- Cartella Clinica con aggiornamenti in tempo reale a cura del personale sanitario e Storico dei trattamenti ricevuti;

### 2. Gestione e monitoraggio delle prenotazioni

Il sistema permetterà una gestione strutturata delle prenotazioni con lo stato della "Richiesta, Confermata e Cancellata". L'algoritmo di prenotazione terrà conto della disponibilità di orari dei medici e operatori sanitari.

### 3. Accesso e utilizzo della cartella clinica elettronica

I medici e gli operatori sanitari potranno accedere alla cartella clinica elettronica dei pazienti. Essi potranno:

- Consultare lo storico clinico del paziente;
- registrare nuove diagnosi;
- prescrivere e consultare terapie farmacologiche o riabilitative;
- registrare l'esecuzione di interventi medici e delle terapie riabilitative.

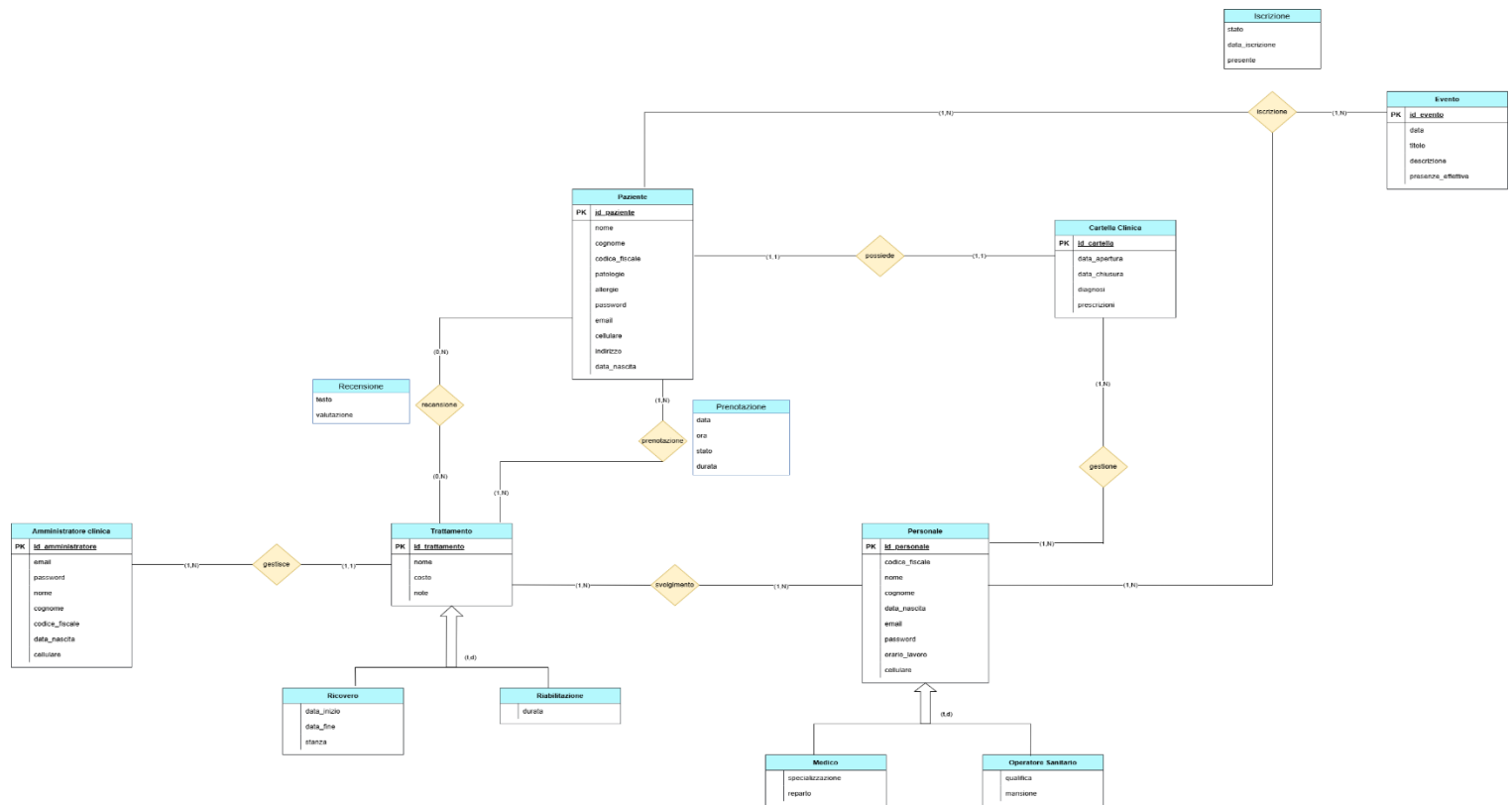
### 4. Gestione dei ruoli e delle funzionalità differenziate

Il sistema prevede un controllo degli accessi differenziato per categoria di utente:

- **Pazienti:** possono accedere solo alle proprie informazioni e funzioni correlate (prenotazioni, cartella clinica personale, eventi, recensioni).

- **Personale:** possono accedere ai dati clinici dei pazienti assegnati, con funzionalità specifiche in base al ruolo, iscriversi ad un evento.
- **Amministratore Clinica:** ha accesso completo alle funzionalità di gestione del sistema, tra cui inserimento/modifica dei trattamenti ed eventi.

## 1. Analisi e progettazione concettuale: Modello E/R



Primo prototipo del modello E/R con generalizzazione totale disgiunta. Il progetto prevede l'utilizzo di generalizzazioni totali e disgiunte per le entità "Trattamento" e "Personale", al fine di modellare correttamente le specializzazioni logiche previste nel dominio applicativo della clinica privata.

La generalizzazione di *Trattamento* definisce due sottotipi: Ricovero e Riabilitazione.

- **Totale:** ogni istanza dell'entità Trattamento deve appartenere obbligatoriamente a una delle due specializzazioni. Un trattamento generico non può esistere.
- **Disgiunta:** un trattamento può essere o un ricovero o una seduta riabilitativa, ma non entrambi contemporaneamente.

L'entità *Personale* è specializzata in: Medico ed Operatore Sanitario.

- **Totale:** ogni persona appartenente al personale sanitario deve essere identificata come medico o operatore. Non esistono figure generiche nel sistema.
- **Disgiunta:** ogni individuo del personale sanitario può ricoprire solo uno dei due ruoli. Non è ammesso che una stessa persona sia contemporaneamente medico ed operatore.

**Possibili soluzioni della generalizzazione**

- 1) Strategia "Tutto nel padre": La generalizzazione ingloba completamente le specializzazioni, assorbendo gli attributi delle entità figlie. È una soluzione conveniente quando le operazioni non differenziano tra le occorrenze delle diverse entità figlie. Tuttavia, può introdurre valori NULL per attributi non applicabili a tutti i tipi.

- Trattamento: (id\_trattamento, nome, costo, note, tipo [ricovero, riabilitazione], data\_inizio, data\_fine, stanza, durata)
- Personale: (id\_personale, codice\_fiscale, nome, cognome, data\_nascita, email, password, ruolo [medico, operatore\_sanitario], orario\_lavoro, specializzazione, reparto, qualifica, mansione, cellulare)

Vantaggi: Meno tabelle, modello più semplice

Svantaggi: Presenza di molti valori NULL se alcuni attributi non si applicano a tutte le entità

- 2) Strategia "Tutto nelle figlie": Consiste nell'accentramento dei dati direttamente nelle entità figlie. Le specializzazioni ereditano gli attributi della generalizzazione, che vengono replicati in ciascuna entità figlia. È una strategia applicabile solo in caso di generalizzazione totale (cioè ogni istanza dell'entità padre appartiene necessariamente a una figlia). È utile quando le operazioni da svolgere sono diverse per ciascun tipo specializzato.

- Medico: (id, email, password, codice\_fiscale, nome, cognome, data\_nascita, orario\_lavoro, cellulare, specializzazione, reparto)
- Operatore\_Sanitario: (email, password, codice\_fiscale, nome, cognome, data\_nascita, orario\_lavoro, cellulare, qualifica, mansione)
- Ricovero: (id, nome, costo, note, data di inizio, data di fine, stanza)
- Riabilitazione: (id, nome, costo, note, durata)

Vantaggi: Nessun valore NULL, poiché ogni attributo è pertinente all'entità.

Svantaggi: Maggior numero di tabelle



Stessa cosa accade per la tabella *Personale*, dove gli attributi delle specializzazioni vengono assorbiti nella super-entità “Personale”, con eventuali valori “NULL” se non applicabili alla tipologia di personale.

- Personale: id\_personale, codice\_fiscale, nome, cognome, data\_nascita, email, password, orario\_lavoro, cellulare.
  - Medico: specializzazione, reparto.
  - Operatore Sanitario: qualifica, mansione.

## 2. Progettazione Logica

La progettazione logica del database è stata realizzata sulla base del modello concettuale E/R precedentemente descritto, adottando una generalizzazione totale e disgiunta sia per l'entità “Trattamento” (specializzata in Ricovero e Riabilitazione), sia per l'entità “Personale” (specializzata in Medico e Operatore Sanitario).

**Paziente:** (id\_paziente, nome, cognome, codice\_fiscale (UNIQUE), patologie, allergie, email (UNIQUE), password, cellulare, indirizzo, data\_nascita)

- L'entità Paziente contiene le informazioni anagrafiche e sanitarie principali di ciascun paziente. Attributi come il codice fiscale e l'email sono definiti come univoci per garantire l'identificazione non ambigua e la corretta autenticazione degli utenti nel sistema.

**Cartella\_Clinica:** (id\_cartella, id\_paziente : Paziente (UNIQUE), id\_trattamento: Trattamento, data\_apertura, data\_chiusura, diagnosi, prescrizioni, interventi):

- Ogni paziente è associato a una e una sola cartella clinica, che può contenere riferimenti a uno o più trattamenti ricevuti. La relazione 1:1 con Paziente è stata implementata includendo la chiave primaria di Paziente come chiave esterna e univoca.

**Trattamento:** (id\_trattamento, nome, costo, id\_paziente : Paziente, note (NULLABLE), tipo(ricovero, riabilitazione), data\_inizio (NULLABLE, ricovero), data\_fine (NULLABLE, ricovero), stanza (NULLABLE, ricovero), durata (NULLABLE, riabilitazione), id\_amministratore : Amministratore\_Clinica):

- La tabella Trattamento comprende sia i ricoveri che le sedute riabilitative. Per distinguere le due tipologie, è stato introdotto l'attributo discriminante **tipo** (valori: ricovero, riabilitazione). Gli attributi specifici delle specializzazioni sono presenti nella tabella e possono assumere valore NULL quando non rilevanti.
- Inoltre poiché tra “Amministratore\_Clinica” e “Trattamento” c'è una relazione (N,1) è stata inserita la chiave primaria di “Amministratore\_Clinica” in “Trattamento”.

**Prenotazione:** (id\_prenotazione, data, ora, stato (richiesta, confermata, cancellata), id\_paziente : Paziente, id\_trattamento : Trattamento, durata, id\_personale : Personale):

- La tabella “Prenotazione” registra la richiesta e l’assegnazione di trattamenti da parte dei pazienti. Ogni prenotazione è legata a un paziente, un trattamento e un membro del personale responsabile. Lo stato della prenotazione può essere: richiesta, confermata, cancellata.

**Evento:** (id\_evento, data, titolo, descrizione, presenze\_effettive):

- Gli eventi (incontri informativi, giornate di screening gratuito, ecc..) sono entità autonome, a cui pazienti e personale possono iscriversi. Sono registrate le presenze effettive.

**Personale:** (id\_personale, codice\_fiscale (UNIQUE), nome, cognome, data\_nascita, email (UNIQUE), password, ruolo(medico, operatore\_sanitario), orario\_lavoro, specializzazione (NULLABLE, medico), reparto (NULLABLE, medico), qualifica (NULLABLE, operatore), mansione (NULLABLE, operatore), cellulare):

- La tabella Personale include sia i medici che gli operatori sanitari. La distinzione tra le due figure professionali è gestita mediante l’attributo discriminante **ruolo**. Gli attributi non pertinenti al tipo di personale specifico possono restare NULL.

**Iscrizione:** (id\_paziente : Paziente, id\_personale : Personale, id\_evento : Evento, id\_iscrizione, stato (iscritto, annullato), data\_iscrizione, presente (BOOLEAN)):

- Associa pazienti e personale agli eventi. Ogni iscrizione è tracciata con stato (iscritto, annullato), data di iscrizione e partecipazione effettiva.

**Gestione:** (id\_cartella : Cartella\_Clinica, id\_personale : Personale):

- Relazione N:N tra Cartella\_Clinica e Personale, utile per tracciare quali membri del personale hanno avuto accesso o responsabilità sulla cartella clinica di un paziente.

**Recensione:** (id\_paziente : Paziente, id\_trattamento : Trattamento, id\_recensione, testo, valutazione (range 1-5)):

- I pazienti possono lasciare recensioni sui trattamenti ricevuti, con un punteggio compreso tra 1 e 5.

**Svolgimento:** (id\_trattamento : Trattamento, id\_personale : Personale):

- Questa tabella rappresenta la partecipazione del personale allo svolgimento di un trattamento, abilitando un’associazione N:N.

*Entità e Relazioni***Paziente**

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_paziente	Identificatore univoco paziente	INT		PK, AUTO_INCREMENT
nome	Nome del paziente	VARCHAR	50	NOT NULL
cognome	Cognome del paziente	VARCHAR	50	NOT NULL
codice_fiscale	Codice fiscale paziente	VARCHAR	16	NOT NULL, UNIQUE
patologie	Patologie del paziente	TEXT		
allergie	Allergie del paziente	TEXT		
email	Email del paziente	VARCHAR	100	NOT NULL, UNIQUE
password	Password di accesso	VARCHAR	255	NOT NULL
cellulare	Numero cellulare	VARCHAR	15	
indirizzo	Indirizzo di residenza	VARCHAR	200	
data_nascita	Data di nascita	DATE		NOT NULL

Codice fiscale ed email unici per pazienti e personale (**vincolo di chiave**). Ogni paziente e ogni membro del personale (sia medico che operatore sanitario) è identificato in modo univoco tramite il codice fiscale e l'email. Questi attributi devono essere unici per impedire la registrazione di utenti duplicati e per consentire il login sicuro e personale.

**Cartella Clinica**

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_cartella	Identificatore univoco cartella	INT		PK, AUTO_INCREMENT
id_paziente	FK Paziente	INT		NOT NULL, UNIQUE, FK → Paziente



Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_trattamento	FK Trattamento	INT		NOT NULL, FK → Trattamento
data_apertura	Data apertura cartella	DATE		NOT NULL
data_chiusura	Data chiusura cartella	DATE		NULLABLE
diagnosi	Diagnosi effettuate	TEXT		NOT NULL
prescrizioni	Prescrizioni mediche	TEXT		NOT NULL
interventi	Interventi eseguiti	TEXT		NULLABLE

Ogni paziente può avere una sola cartella clinica attiva (**vincolo di chiave**), con una relazione uno a uno (1:1) tra la tabella Cartella\_Clinica e Paziente.

### Trattamento

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_trattamento	Identificatore univoco trattamento	INT		PK, AUTO_INCREMENT
nome	Nome del trattamento	VARCHAR	100	NOT NULL
costo	Costo del trattamento	DECIMAL(8,2)		NOT NULL
id_paziente	FK Paziente	INT		NOT NULL, FK → Paziente
note	Note aggiuntive	TEXT		NULLABLE
tipo	Tipo trattamento (ricovero, riabilitazione)	VARCHAR	20	NOT NULL, scelta = 'ricovero' o 'riabilitazione'
data_inizio	Data inizio ricovero	DATE		NULLABLE (riempire solo se tipo = ricovero)
data_fine	Data fine ricovero	DATE		NULLABLE (riempire solo se tipo = ricovero)

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
stanza	Numero stanza ricovero	VARCHAR	10	NULLABLE (riempire solo se tipo = ricovero)
durata	Durata trattamento (minuti)	INT		NULLABLE (riempire solo se tipo = riabilitazione)
id_amministratore	FK Amministratore	INT		NOT NULL, FK → Amministratore_Clinica

Il campo “Tipo” può essere definito come ricovero o riabilitazione, a seconda della scelta i campi relativi saranno compilati (**vincolo di dominio**).

### Prenotazione

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_prenotazione	Identificatore prenotazione	INT		PK, AUTO_INCREMENT
data	Data prenotazione	DATE		NOT NULL
ora	Ora prenotazione	TIME		NOT NULL
stato	Stato prenotazione (richiesta, confermata, cancellata)	VARCHAR	20	NOT NULL, valori ammessi = richiesta, confermata, cancellata
durata	Durata in minuti	INT		NOT NULL
id_paziente	FK Paziente	INT		NOT NULL, FK → Paziente
id_trattamento	FK Trattamento	INT		NOT NULL, FK → Trattamento
id_personale	FK Personale	INT		NOT NULL, FK → Personale

Il campo stato delle prenotazioni può assumere solo valori specifici (**vincolo di dominio**). I valori ammessi sono: richiesta, confermata, cancellata.

**Evento**

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_evento	Identificatore evento	INT		PK, AUTO_INCREMENT
data	Data evento	DATE		NOT NULL
titolo	Titolo evento	VARCHAR	150	NOT NULL
descrizione	Descrizione evento	TEXT		NULLABLE
presenze_effettive	Numero presenze effettive	INT		NULLABLE

**Recensione**

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_recensione	Identificatore recensione	INT		PK, AUTO_INCREMENT
id_paziente	FK Paziente	INT		NOT NULL, FK → Paziente
id_trattamento	FK Trattamento	INT		NOT NULL, FK → Trattamento
testo	Testo recensione	TEXT		NULLABLE
valutazione	Valutazione da 1 a 5	INT		NOT NULL, valori ammessi = da 1 a 5

Il campo valutazione delle recensioni deve essere un numero intero tra 1 e 5 (**vincolo di dominio**). Questo vincolo limita il range della valutazione per standardizzare il feedback.

**Amministratore Clinica**

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_amministratore	Identificatore amministratore	INT		PK, AUTO_INCREMENT
email	Email amministratore	VARCHAR	100	NOT NULL, UNIQUE
password	Password	VARCHAR	255	NOT NULL

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
nome	Nome	VARCHAR	50	NOT NULL
cognome	Cognome	VARCHAR	50	NOT NULL
codice_fiscale	Codice fiscale	VARCHAR	16	NOT NULL, UNIQUE
cellulare	Cellulare	VARCHAR	15	NULLABLE
data_nascita	Data nascita	DATE		NOT NULL

### Personale

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_personale	Identificatore personale	INT		PK, AUTO_INCREMENT
codice_fiscale	Codice fiscale	VARCHAR	16	NOT NULL, UNIQUE
nome	Nome	VARCHAR	50	NOT NULL
cognome	Cognome	VARCHAR	50	NOT NULL
data_nascita	Data nascita	DATE		NOT NULL
email	Email	VARCHAR	100	NOT NULL, UNIQUE
password	Password	VARCHAR	255	NOT NULL
ruolo	Ruolo (medico, operatore_sanitario)	VARCHAR	30	NOT NULL, scelta = 'medico' o 'operatore_sanitario'
orario_lavoro	Orario di lavoro	VARCHAR	50	NULLABLE
specializzazione	Specializzazione	VARCHAR	100	NULLABLE (riempire solo se ruolo = medico)
reparto	Reparto	VARCHAR	50	NULLABLE (riempire solo se ruolo = medico)
qualifica	Qualifica	VARCHAR	50	NULLABLE (riempire solo se ruolo = operatore_sanitario)

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
mansione	Mansione	VARCHAR	100	NULLABLE (riempire solo se ruolo = operatore_sanitario)
cellulare	Cellulare	VARCHAR	15	NULLABLE

Il campo Ruolo può essere definito come medico oppure come operatore sanitario (**vincolo di dominio**), a seconda della scelta i campi relativi saranno compilati.

### Iscrizione

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_iscrizione	Identificatore iscrizione	INT		PK, AUTO_INCREMENT
id_paziente	FK Paziente	INT		NOT NULL, FK → Paziente
id_personale	FK Personale	INT		NULLABLE, FK → Personale
id_evento	FK Evento	INT		NOT NULL, FK → Evento
stato	Stato iscrizione (iscritto, annullato)	VARCHAR	20	NOT NULL, valori ammessi = iscritto, annullato
data_iscrizione	Data iscrizione	DATE		NOT NULL
presente	Presenza effettiva (boolean)	BOOLEAN		NOT NULL, DEFAULT FALSE

Un paziente può iscriversi solo una volta a un evento (**vincolo di tupla**). La combinazione id\_paziente - id\_evento nella tabella Iscrizione deve essere unica.

### Gestione

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_cartella	FK Cartella Clinica	INT		PK, FK → Cartella_Clinica
id_personale	FK Personale	INT		PK, FK → Personale

## Svolgimento

Nome campo	Descrizione	Tipo dati	Lunghezza	Vincoli
id_trattamento	FK Trattamento	INT		PK, FK → Trattamento
id_personale	FK Personale	INT		PK, FK → Personale

### Vincoli interrelazionali:

- Ogni trattamento deve essere associato a un paziente esistente. Il trattamento è sempre eseguito per un paziente registrato nel sistema.
- Ogni prenotazione deve riferirsi a un trattamento e a un paziente validi. Le prenotazioni sono legate sia al paziente che al trattamento che vuole ricevere.
- Ogni recensione deve riferirsi a un trattamento effettivamente eseguito dal paziente.
- Ogni iscrizione ad un evento deve riferirsi a un paziente e ad un evento validi, oppure ad un evento ed ad un membro del personale.
- Ogni ricovero deve essere gestito da un medico, ogni riabilitazione da un operatore sanitario.

## 3. Implementazione Sistema Informativo

L'implementazione del sistema informativo è stata realizzata utilizzando il framework Django, che consente di sviluppare un'applicazione web scalabile, sicura e facilmente mantenibile. Il sistema è progettato per gestire i dati secondo il modello logico precedentemente definito e per offrire accesso a un insieme di funzionalità chiave, rivolte a diverse categorie di utenti: pazienti, personale sanitario e amministratori.

L'applicazione supporta almeno quattro funzionalità principali, selezionate in base alle esigenze cliniche e organizzative, garantendo una gestione efficace e intuitiva:

### 1. Registrazione e autenticazione degli utenti:

- I pazienti possono registrarsi autonomamente creando un account personale tramite un modulo dedicato. Successivamente possono effettuare il login per accedere alle funzionalità riservate.
- Il personale sanitario e l'amministratore della clinica accede solo tramite login, con credenziali già definite e riportate nel database.

### 2. Prenotazione e gestione appuntamenti:

- I pazienti hanno la possibilità di prenotare appuntamenti per trattamenti specifici, scegliendo data e ora in base alle disponibilità del personale. Il sistema consente inoltre di annullarle.
- Il personale può confermare le richieste al termine del trattamento scelto, compilando i campi mancanti della cartella clinica.

3. **Inserimento e consultazione delle recensioni:** Dopo aver completato un trattamento, i pazienti possono lasciare recensioni e valutazioni, che contribuiscono a migliorare la qualità del servizio. Le recensioni sono rese visibili cliccando su “Recensioni” nella homepage.
4. **Gestione degli eventi e seminari:** La clinica organizza eventi formativi e informativi aperti a pazienti e personale. Gli utenti possono visualizzare il calendario eventi, iscriversi o annullare l'iscrizione. Questo favorisce l'aggiornamento continuo e la partecipazione attiva alla vita della clinica.
5. **Consultazione della Cartella Clinica:**
  - I pazienti possono accedere in modo sicuro alla propria cartella clinica digitale per consultare diagnosi, prescrizioni, interventi e trattamenti in corso.
  - Il personale medico aggiorna costantemente queste informazioni, garantendo una visione sempre attuale dello stato di salute del paziente.
6. **Creazione eventi e trattamenti:** l'amministratore della clinica è colui che imposta il catalogo effettivo degli eventi che la clinica ospiterà e aggiorna i trattamenti che verranno effettuati in struttura.

### Homepage

La homepage rappresenta la vetrina pubblica della clinica ed è accessibile senza autenticazione.



### Orari della Clinica

Giorno	Orario
Lunedì - Sabato	08:00 – 20:00
Domenica	Chiuso

### Contatti

**Email**[info@clinicareviva.it](mailto:info@clinicareviva.it)**Telefono**[+39 0813 456789](tel:+390813456789)**Indirizzo**Via della Salute, 10  
80040 Napoli (Na)

#### Comprende:

- **Storia Clinica:** sezione dedicata alla descrizione della clinica.
- **Accesso Area Privata:** pulsante che reindirizza l'utente al portale di autenticazione.
- **Orari e Contatti:** informazioni su orari di apertura, numero di telefono, e-mail e locazione fisica.
- **Trattamenti:** elenco aggiornato dei trattamenti offerti dalla clinica. Se l'utente possiede già un profilo può procedere alla prenotazione.
- **Eventi:** sezione informativa su eventi futuri. Accesso riservato per la registrazione.
- **Recensioni:** area dedicata alle opinioni e valutazioni dei clienti sui trattamenti effettuati.

#### Area Riservata – Gestione Autenticata (con sessioni)

La piattaforma implementa l'autenticazione e la gestione delle sessioni Django per garantire che ogni utente acceda solo alle funzionalità permesse in base al proprio profilo. Il login è differenziato in tre tipologie:

**Clinica ReViva**[Trattamenti](#) [Eventi](#) [Recensioni](#)[← Indietro](#)

### Accedi alla tua Area Privata

**Paziente**

Accedi come paziente per consultare i tuoi dati clinici.

Accedi

**Personale Medico**

Accedi come medico o operatore sanitario per gestire i pazienti.

Accedi

**Amministratore**

Accesso riservato per gestione sistema e utenti.

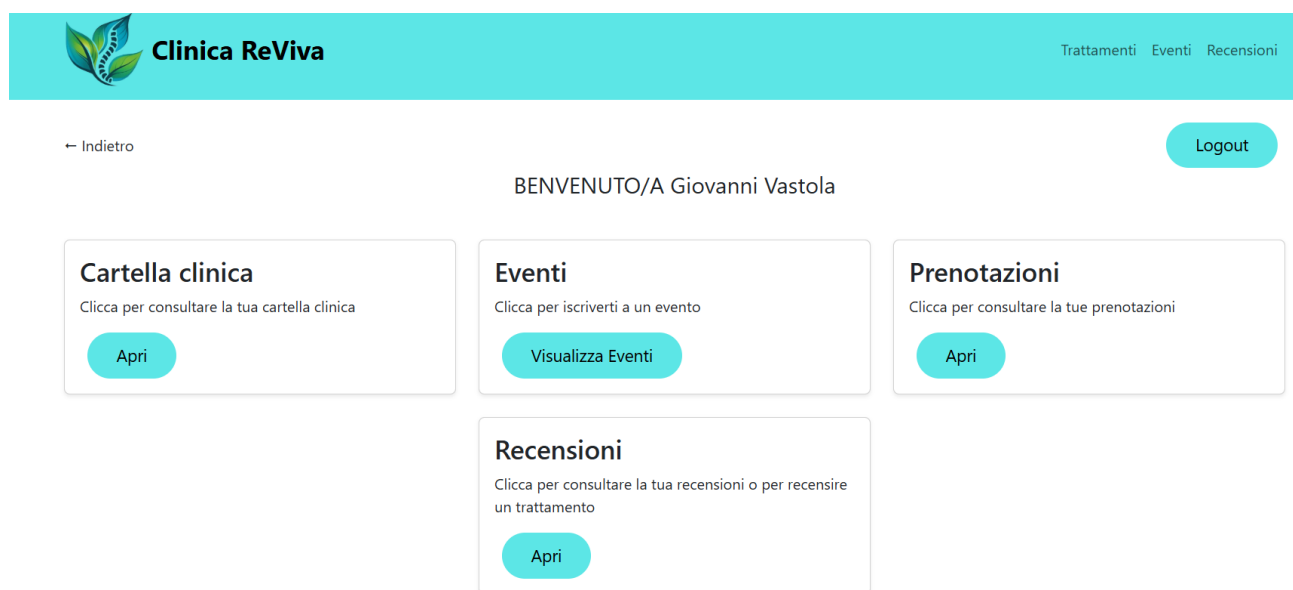
Accedi



## Login Paziente

I pazienti hanno una dashboard personale con funzionalità personalizzate:


- **Registrazione Nuovo Account:** se il paziente non è registrato, può creare un account compilando un modulo.
- **Cartella Clinica Personale:** dopo l'accesso, può visualizzare i propri dati sanitari aggiornati.
- **Prenotazione Trattamenti:** possibilità di effettuare una nuova prenotazione o annullare una esistente.
- **Iscrizione Eventi:** può iscriversi a eventi pubblici organizzati dalla clinica.
- **Recensioni:** può lasciare una recensione solo per i trattamenti che ha realmente effettuato (verifica tramite sessione).



## Login Personale Medico

Il personale medico dispone di strumenti clinici e informativi:

- **Accesso Cartelle Cliniche:** possibilità di consultare, creare, modificare o eliminare le cartelle cliniche dei pazienti.
- **Iscrizione Eventi:** può iscriversi agli eventi della clinica utili per l'aggiornamento professionale.
- **Visualizzazione Appuntamenti:** elenco degli appuntamenti previsti con i pazienti.

 **Clinica ReViva**

Trattamenti Eventi Recensioni

← Indietro Logout

BENVENUTO/A Mario Rossi

**Cartella clinica**  
Clicca per consultare le cartelle cliniche dei pazienti  
Apri


**Eventi**  
Clicca per iscriverti a un evento  
Visualizza Eventi

**Appuntamenti**  
Clicca per consultare i tuoi appuntamenti  
Apri

## Login Amministratore

L'amministratore ha accesso completo alle funzionalità gestionali della clinica:

- **Gestione Trattamenti:** visualizzazione della lista dei trattamenti disponibili, possibilità di aggiungere, modificare o eliminare un trattamento.
- **Gestione Eventi:** creare nuovi eventi, modificare dettagli esistenti, eliminare eventi, aggiornare le presenze.
- **Visualizzazione Prenotazioni:** accesso a una lista dettagliata di tutte le prenotazioni effettuate dai pazienti.

 **Clinica ReViva**

Trattamenti Eventi Recensioni

← Indietro Logout

BENVENUTO/A Giulia Ambrosio

**Lista Trattamenti Clinica**  
Clicca per consultare la tua lista dei trattamenti  
Apri

**Lista Eventi Clinica**  
Clicca per consultare la tua lista degli eventi  
Apri

**Lista Prenotazioni Clinica**  
Clicca per consultare la tua lista delle prenotazioni  
Apri

## Gestione delle sessioni

Nel progetto, le sessioni svolgono un ruolo fondamentale nella gestione dell'autenticazione e dell'autorizzazione degli utenti, ovvero nel mantenere lo stato di login durante la navigazione tra le varie pagine dell'applicazione web. Questo è particolarmente importante perché HTTP è un protocollo stateless, quindi senza un meccanismo come le sessioni, ogni richiesta sarebbe trattata come indipendente e anonima.

Django fornisce un sistema di autenticazione integrato che gestisce login, logout, registrazione, controllo dei permessi e gestione delle credenziali. Tuttavia la scelta di non usare il sistema `auth_user` di Django è motivata dalla necessità di gestire tre ruoli distinti con attributi, metodi e interfacce differenti. Mantenere modelli separati per ciascun tipo di utente (Paziente, Personale, Amministratore) consente un'organizzazione del codice più chiara.

## Login e creazione della sessione

Quando un utente (paziente, personale o amministratore) effettua il login con successo, dopo aver verificato che le credenziali siano corrette (tramite query al database con password hashata), viene memorizzata nella sessione una chiave identificativa univoca, in questo caso l'email dell'utente:

- `request.session['paziente_email'] = email` per il paziente
- `request.session['personale_email'] = email` per il personale
- `request.session['amministratore_email'] = email` per l'amministratore della clinica

In questo modo, ogni volta che l'utente farà una nuova richiesta, il sistema riconoscerà la sua sessione attiva tramite questa informazione, ciò impedisce che un utente autenticato come paziente possa accedere per errore o malintenzionatamente alle funzioni dedicate al personale.

## Esempio di login

```
def autenticazione_paziente(email, password):  
    return Paziente.objects.filter(email=email, password=password).exists()
```

- Con questa funzione viene verificato se esiste un paziente nel database con l'email e la password specificate, essa viene ripresa nella funzione di login a seguire.

```
def login_paziente(request):  
    if request.method == 'POST':  
        email = request.POST.get('email')  
        password = request.POST.get('password')  
        hashed_password =  
        hashlib.md5(password.encode()).hexdigest()  
        #print (email, " : ", hashed_password)  
        if autenticazione_paziente(email, hashed_password):  
            request.session['paziente_email'] = email
```

```
        return redirect('area_riservata_paziente')
    else:
        return render(request, 'login_paziente.html',
                        {'error': "Credenziali non valide, riprova",
                         'email': email})
    else:
        return render(request, 'login_paziente.html')
```

- Gestione del processo di login del paziente, verificando le credenziali e creando la sessione. La funzione accetta solo richieste di tipo POST per il login. Se la richiesta non è POST (es. accesso diretto tramite URL), mostra la pagina di login senza effettuare alcuna verifica. Prende l'email inserita dall'utente tramite *request.POST.get('email')*. Prende la password inserita tramite *request.POST.get('password')*.
- Se esiste un paziente con email e password corrispondenti viene creata una sessione per l'utente inserendo l'email nella chiave 'paziente\_email'. L'utente viene reindirizzato all'area riservata paziente *redirect('area\_riservata\_paziente')*. Altrimenti viene mostrata nuovamente la pagina di login.

Nelle altre funzioni, che interessano le singole aree private, per controllare che l'utente è ancora in sessione è usato “*if 'paziente\_email' in request.session*”:

```
def area_riservata_paziente(request):
    if 'paziente_email' in request.session:
        email = request.session['paziente_email']
        paziente = Paziente.objects.get(email=email)
        return render(request, 'area_riservata_paziente.html', {'paziente': paziente})
    else:
        return redirect('login_paziente')
```

- Se la chiave non è presente, significa che l'utente non è autenticato e viene reindirizzato alla pagina di login. Questo evita accessi non autorizzati.

Quando l'utente decide di uscire (logout), la sessione viene completamente svuotata tramite:

```
def logout_paziente(request):
    request.session.flush()
    return redirect('login_paziente')
```

- Ciò comporta la cancellazione di tutte le informazioni di sessione memorizzate, invalidando la sessione attuale. L'utente dovrà quindi effettuare nuovamente il login per accedere alle aree protette.

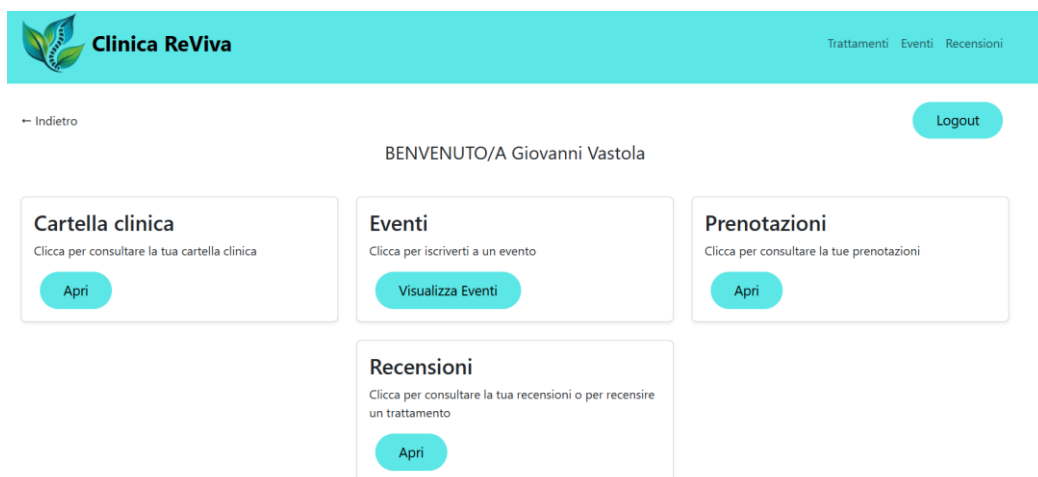
## Esempio pratico

Accedo come paziente:



The image shows a 'Login Paziente' (Patient Login) form. It has a title 'Login Paziente' at the top. Below it, there are two input fields: 'Email:' with the value 'paziente2@clinica.it' and 'Password:' with a masked password '.....'. A large teal 'Login' button is below the password field. At the bottom, there is a link 'Non hai un account?' and a 'Registrati' button.

Ho già un account valido, le credenziali inserite sono corrette, vengo reindirizzato nella mia area privata:



The image shows a patient dashboard for 'Clinica ReViva'. The header is teal with the clinic logo and name, and links for 'Trattamenti', 'Eventi', and 'Recensioni'. Below the header, there is a 'Logout' button. The main content area is titled 'BENVENUTO/A Giovanni Vastola'. It contains four cards: 'Cartella clinica' (with an 'Apri' button), 'Eventi' (with a 'Visualizza Eventi' button), 'Prenotazioni' (with an 'Apri' button), and 'Recensioni' (with an 'Apri' button). Each card has a description of what the user can do by clicking the button.

Analizzando il traffico HTTP (utilizzando Wireshark) durante l'utilizzo dell'applicazione web, si può osservare la creazione automatica di due cookie principali:

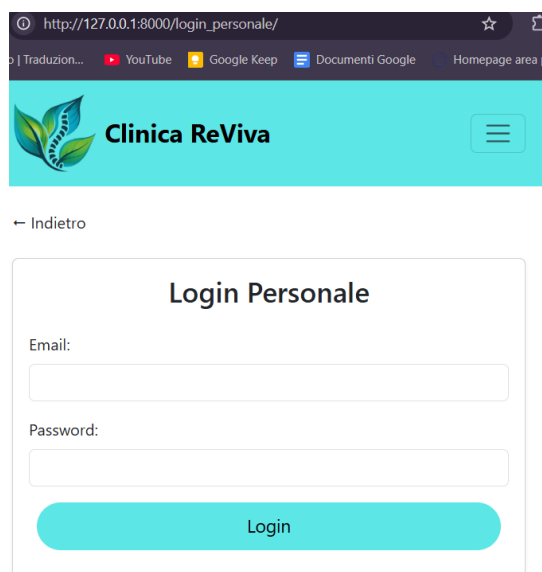
- **CsrfToken** = IvCm60YVvP5HyP9BcT7B1DLsJ60KPFXF: utilizzato da Django per proteggere i moduli POST da attacchi CSRF (Cross-Site Request Forgery).
- **Sessionid** = elqyqugcmkqenwrhob10jay3z18j42ey: identificativo univoco della sessione utente attiva. Se viene effettuato il logout il sessionid scompare.

```
GET /logout_paziente/ HTTP/1.1
Host: 127.0.0.1:8000
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="137", "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:8000/area_riservata_paziente/
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: csrftoken=IvCm60YVvP5HyP9BcT7B1DLsJ60KPFXF; sessionid=elqyqugcmkqenwrhob10jay3zl8j42ey
```

```
GET /login_paziente/ HTTP/1.1
Host: 127.0.0.1:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
sec-ch-ua: "Google Chrome";v="137", "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Referer: http://127.0.0.1:8000/area_riservata_paziente/
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: csrftoken=IvCm60YVvP5HyP9BcT7B1DLsJ60KPFXF
```

Dopo aver effettuato l'accesso come paziente, se si tenta manualmente di accedere all'area riservata di un altro ruolo (ad esempio andando su "[http://127.0.0.1:8000/area\\_riservata\\_personale/](http://127.0.0.1:8000/area_riservata_personale/)"), l'applicazione web non permette l'accesso diretto: l'utente viene correttamente reindirizzato alla pagina di login del personale.

Questo comportamento è garantito dal controllo sulla sessione implementato lato server: viene verificata la presenza della chiave *personale\_email* nella sessione. Poiché tale chiave non esiste (l'utente è autenticato solo come paziente), l'accesso viene negato.



← Indietro

### Implementazione SQL Injection

La SQL Injection (SQLi) è una delle vulnerabilità di sicurezza web più diffuse e pericolose. Consiste nell'inserimento di comandi SQL malevoli all'interno dei campi di input dell'applicazione, con l'obiettivo di manipolare le query inviate al database. Uno degli scopi più comuni di questo attacco è l'estrazione non autorizzata di dati sensibili. L'attacco SQLi si basa spesso sull'interruzione anticipata di una stringa SQL seguita da un'iniezione di codice. Per ignorare la parte restante della query, si utilizza il carattere di commento `--`, che fa sì che tutto ciò che segue venga ignorato dal database durante l'esecuzione.

Per simulare una SQL injection nel progetto è stata creata una funzione di login da sostituire a quella originale per testarla.

```
@csrf_exempt
def login_vulnerabile_paziente(request):
    if request.method == "POST":
        email = request.POST.get('email')
        password = request.POST.get('password')
        #query SQL vulnerabile
        query = f"SELECT * FROM clinica_paziente WHERE email = '{email}' AND password = '{password}'"
        with connection.cursor() as cursor:
            cursor.execute(query)
            row = cursor.fetchone()
            if row:
                return render(request, 'area_riservata_paziente.html', {'paziente':row})
            else:
                return render(request, 'login_vulnerabile_paziente.html', {'error': 'Credenziali non valide.'})
    else:
        return render(request, 'login_vulnerabile_paziente.html')
```

Le variabili email e password, ricevute da una richiesta POST, sono inserite direttamente nella query SQL senza alcun filtro, rendendo il codice vulnerabile a SQL injection.

- `SELECT * FROM clinica_paziente WHERE email = '{email}' AND password = '{password}'`.

Il decoratore `@csrf_exempt` disattiva la protezione CSRF di Django, consentendo l'invio di richieste POST anche da strumenti automatizzati come sqlmap.

*Cursor* ci permette di puntare al database, tramite cui possiamo eseguire i comandi SQL e recuperare dati. Viene instaurata la connessione, per eseguire la query è usato *execute*, in seguito i dati sono recuperati con *fetchone* che prende la prima riga del risultato. Se è trovato un risultato valido dopo l'esecuzione della query *row* conterrà i dati attesi e si accederà alla pagina prevista.

Per analizzare la vulnerabilità della pagina è stato utilizzato “Sqlmap”, esso ha confermato in modo definitivo che il parametro email del modulo di login è vulnerabile a SQL Injection.

Come input nel prompt dei comandi è stato inviato questo:

- `python sqlmap.py -u "http://127.0.0.1:8000/login_vulnerabile_paziente/" --data "email=test&password=test" --batch -dbs`

L'analisi ha rivelato che il parametro email è vulnerabile a questi tipi di SQL Injection:

- 1) **Boolean (blind)-based:** Modifica la logica della query per ottenere risposte vere o false.  
Esempio: ' OR 1=1 - -

[15:19:08] [INFO] POST parameter 'email' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)' injectable

- 2) **Error-based:** Estrae dati sfruttando i messaggi di errore del database.  
Esempio: ' OR (SELECT 1 FROM (SELECT COUNT(\*), CONCAT(..., FLOOR(RAND(0)))) x)- -

[15:19:09] [INFO] POST parameter 'email' is 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable

- 3) **Stacked queries:** Permette l'esecuzione di più query nello stesso comando, separandole con il ;.  
Esempio: '; SELECT SLEEP(5)#

[15:19:19] [INFO] POST parameter 'email' appears to be 'MySQL >= 5.0.12 stacked queries (comment)' injectable

- 4) **Time-based blind:** Deduce informazioni basandosi sul ritardo nella risposta del server, utilizzando comandi come SLEEP().  
Esempio: ' AND SLEEP(5)—

[15:19:29] [INFO] POST parameter 'email' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable

È stata creata anche una pagina HTML dedicata per testare il login vulnerabile. Per facilitare l'iniezione, il campo email è stato reso un campo di tipo testo semplice, e si è usato l'attributo *novalidate* per disattivare i controlli automatici del browser (come il *required* o la validazione dell'email):

```
<form method="POST" novalidate>
  {% csrf_token %}
  <div class="mb-3">
    <label for="email" class="form-label">Email:</label>
    <input type="text" class="form-control" id="email" name="email" required>
```



</form>

- L'uso di *novalidate* consente l'invio del form anche con campi vuoti o valori non conformi.

## Esempio di attacco SQL Injection

Se accediamo alla pagina web della clinica e ci dirigiamo nell'area riservata, selezionando il login del paziente, possiamo provare ad entrare nel sistema usando come email: ' OR '1'='1' - - (importante mettere lo spazio dopo i trattini per far capire che si tratta di un commento) e come password un campo qualsiasi.

The screenshot shows a web browser at the URL `http://127.0.0.1:8000/login_vulnerabile_paziente/`. The page header for "Clinica ReViva" is visible. The "Login Paziente" form has the following fields:

- Email: `' OR '1'='1' --`
- Password: `....`
- Login button

Cliccando su login saremo indirizzati verso l'area personale di un paziente.

The screenshot shows the patient dashboard after a successful login. The URL remains `http://127.0.0.1:8000/login_vulnerabile_paziente/`. The page header includes "Clinica ReViva" and navigation links for "Trattamenti", "Eventi", and "Recensioni". A "Logout" button is in the top right. The main content area is titled "BENVENUTO/A" and contains four cards:

- Cartella clinica**: "Clicca per consultare la tua cartella clinica" with an "Apri" button.
- Eventi**: "Clicca per iscriverti a un evento" with a "Visualizza Eventi" button.
- Prenotazioni**: "Clicca per consultare la tue prenotazioni" with an "Apri" button.
- Recensioni**: "Clicca per consultare la tua recensioni o per recensire un trattamento" with an "Apri" button.

Passando il campo 'OR '1'='1' - - nell'email la query si trasforma in:

- `SELECT * FROM clinica_paziente WHERE email = " OR '1'='1' AND password = " OR '1'='1';`

Poiché '1'='1' è sempre vera, il database restituirà comunque dei risultati e l'utente verrà autenticato senza credenziali valide, accedendo così all'area riservata.

## SQL Injection soluzione

Nel progetto della clinica privata, realizzato con Django, il rischio di SQL Injection è mitigato in modo nativo grazie all'utilizzo dell'ORM (Object-Relational Mapping) di Django, che genera query SQL in modo sicuro, escludendo automaticamente input potenzialmente pericolosi. Ad esempio, la creazione di un paziente avviene come di seguito:

```
paziente=Paziente.objects.create(
    email=email,
    password=hashed_password,
    nome=nome,
    cognome=cognome,
    codice_fiscale=codice_fiscale,
    patologie=patologie,
    allergie=allergie,
    cellulare=cellulare,
    indirizzo=indirizzo,
    data_nascita=data_nascita
)
```

Inoltre per accedere all'area riservata la funzione viene imposta in questo modo:

```
def area_riservata_paziente(request):
    if 'paziente_email' in request.session:
        email = request.session['paziente_email']
        paziente = Paziente.objects.get(email=email)
        return render(request, 'area_riservata_paziente.html', {'paziente': paziente})
    else:
        return redirect('login_paziente')
```

- L'uso di `Paziente.objects.get(email=email)` genera una query sicura e parametrizzata. Inoltre viene fatto un controllo di sessione come specificato sopra.

Per la protezione delle credenziali degli utenti, le password non vengono mai memorizzate in chiaro nel database. Durante il login, la password inserita viene trasformata in una stringa hash mediante l'algoritmo MD5, aumentando ulteriormente la sicurezza:

```
password = request.POST.get('password')
hashed_password = hashlib.md5(password.encode()).hexdigest()
```

Nel progetto è stata posta particolare attenzione alla sicurezza dei dati sensibili, considerando la natura sanitaria dell'applicazione. Una delle principali minacce a cui si è voluto dare risposta è il Cross-Site Request Forgery (CSRF). Tutti i file costituenti le pagine HTML dell'applicazione web includono automaticamente il token CSRF tramite il template tag Django: `{% csrf_token %}`.

La sua presenza impedisce che un attaccante possa forzare l'utente ad inviare involontariamente dati o modificare informazioni, proteggendo così tutte le operazioni critiche dell'applicazione. Nel contesto specifico del progetto, sono protette da CSRF operazioni fondamentali come:

- la modifica di cartelle cliniche da parte del personale sanitario;
- la prenotazione di appuntamenti da parte dei pazienti;
- l'inserimento di recensioni;
- l'iscrizione ad eventi.

A rafforzare ulteriormente questa protezione, è stato utilizzato il decoratore `@require_POST` nelle viste Django che gestiscono l'invio di dati sensibili. Nel progetto, questo approccio è stato applicato in modo mirato nelle viste che gestiscono operazioni come:

- l'eliminazione di un trattamento;
- l'iscrizione ad un evento.

❖ ❖ ❖ ❖ ❖ ❖ ❖ ❖ ❖ ❖ ❖

## Sommario

Traccia d'Esame - Progetto di Sistema Informativo per la Clinica Privata <b>ReViva</b> .....	1
Obiettivi del Progetto .....	2
1. Analisi e progettazione concettuale: Modello E/R .....	3
2. Progettazione Logica .....	6
3. Implementazione Sistema Informativo .....	14