

CasaNote

Titolo del progetto: CasaNote
Alunno/a: Andrea Casamatta, Paolo Comes, Matvej Rossi
Classe: I3BB
Anno scolastico: 2024/2025
Docente responsabile: Geo Petrini

1	Introduzione	3
1.1	Informazioni sul progetto	3
1.2	Scopo	3
2	Analisi	4
2.1	Analisi del dominio	4
2.2	Analisi e specifica dei requisiti	4
2.2.1	Spiegazione elementi tabella dei requisiti:.....	7
2.3	Use case	8
2.4	Pianificazione	9
2.5	Analisi dei mezzi.....	11
2.5.1	Software	11
2.5.2	Hardware.....	11
3	Progettazione	12
3.1	Design dell'architettura del sistema	12
3.1.1	Swimlanes	13
3.2	Design dei dati e database.....	16
3.3	Design delle interfacce.....	17
3.3.1	Registrazione utente	17
3.3.2	Login utente	18
3.3.3	Home.....	19
3.3.4	Creazione nota.....	20
3.3.5	Visualizzazione note	21
3.4	Design procedurale	Errore. Il segnalibro non è definito.
4	Implementazione	22
4.1	Sistema di sicurezza	22
4.2	Sistema di log	22
4.3	Sistema di validazione	23
4.3.1	Metodi	23
4.4	Classi Mapper	24
4.5	Controllers.....	26
4.5.1	Controller: manage	26
4.6	Disegno su canvas.....	29
4.7	Script gestione input.....	30
4.8	Views	31
4.9	Struttura.....	32
4.9.1	Flusso tipico nel progetto	32
5	Test.....	33
5.1	Protocollo di test.....	33
5.2	Risultati test.....	40
5.3	Mancanze/limitazioni conosciute.....	51
6	Consuntivo.....	52
7	Conclusioni	54
7.1	Sviluppi futuri.....	54
7.2	Considerazioni personali.....	54
8	Glossario	55
9	Bibliografia.....	57
9.1	Sitografia	57
10	Indice delle figure.....	57
11	Allegati	57

1 Introduzione

1.1 Informazioni sul progetto

- Allievi: Andrea Casamatta, Paolo Comes, Matvej Rossi
- Supervisore: Geo Petrini
- Classe: 3BB SAM Trevano, sezione informatica
- Data di inizio progetto: 8.01.2025
- Data di consegna: 05.05.2025

1.2 Scopo

Lo scopo del progetto CasaNote è quello di fornire un'applicazione web interattiva, che consenta all'utente di accedere, di creare, visualizzare e esportare note, dove si può oltre a scrivere, disegnare e registrare audio. Grazie alle diverse funzionalità, il progetto è utile a scopo personale per prendere note e appunti per esempio. Ogni utente su CasaNote avrà il proprio username e la propria password, dove vengono salvate le proprie note create.

2 Analisi

2.1 Analisi del dominio

CasaNote è un progetto ispirato a OneNote di Office, ma mira a colmare alcune lacune delle app attualmente disponibili, come Evernote, Notion o OneNote stesso. Anche se queste applicazioni sono molto utilizzate per la gestione delle informazioni, CasaNote si focalizza su esigenze specifiche di utenti come famiglie, studenti e professionisti, offrendo una soluzione semplice, sicura e rispettosa della privacy.

A differenza di altre app, CasaNote non richiede la raccolta di dati personali (come i cookie) e garantisce un accesso facile e gratuito. È progettata per essere utilizzata in vari contesti, come a casa, a scuola, al lavoro o in viaggio, con un'attenzione particolare alla sincronizzazione tra dispositivi e alla facilità di accesso. Gli utenti possono personalizzare i propri appunti e organizzare documenti e informazioni senza complicazioni.

CasaNote offre un'esperienza più user-friendly e attenzione all'accessibilità e sicurezza dei dati.

2.2 Analisi e specifica dei requisiti

Il sito web multiutente consente agli utenti di registrarsi, autenticarsi e gestire un blocco note digitale. Le funzionalità includono scrittura, modifica, salvataggio, registrazione audio, importazione di file e esportazione in ZIP. Il sito sarà responsive, con un'interfaccia semplice tramite Bootstrap e tutti i dati cifrati nel database. L'uso CAPTCHA, anche se è una funzione extra, potrebbe essere implementato.

ID: REQ-01	
Nome	Sito web multiutente
Priorità	1
Versione	1.0
Note	

ID: REQ-02	
Nome	Autenticazione
Priorità	1
Versione	1.0
Note	Autenticazione con email e password.

ID: REQ-03	
Nome	Registrazione
Priorità	1
Versione	1.0
Note	<ul style="list-style-type: none"> Non è richiesta conferma dell'autenticità È richiesta la mail
Sotto requisiti	
001	Come extra captcha v3

ID: REQ-04	
Nome	Utente gestisce blocco note
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Scrivere del testo, salvare testo, modificare titolo, modificare nota, cercare il titolo
002	Disegnare
003	Registrazione audio
004	Importare file di qualunque tipo
005	Esportare il blocco note come file zip

ID: REQ-05	
Nome	Utente elimina blocco note
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Conferma di eliminazione.

ID: REQ-06	
Nome	Sito deve essere responsive
Priorità	1
Versione	1.0
Note	

ID: REQ-07	
Nome	Dati nel db tutti cifrati
Priorità	1
Versione	1.0
Note	

ID: REQ-08	
Nome	Interfaccia semplice e intuitiva
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Bootstrap

2.2.1 Spiegazione elementi tabella dei requisiti:

ID: identificativo univoco del requisito

Nome: breve descrizione del requisito

Priorità: indica l'importanza di un requisito nell'insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

Versione: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata. Sulla documentazione apparirà solamente l'ultima versione, mentre le vecchie dovranno essere inserite nei diari.

Note: eventuali osservazioni importanti o riferimenti ad altri requisiti.

Sotto requisiti: elementi che compongono il requisito.

2.3 Use case

Questo è lo Use Case che descrive il funzionamento della mia applicazione. Come si può vedere l'utente può visualizzare le sue note può modificarle e eliminarle. La nota può essere esportata in formato pdf o txt. L'utente nella nota può allegare file, disegnare o registrare audio, formattare il testo oltre a scriverlo. Per fare tutto ciò l'utente deve aver prima effettuato l'autenticazione, altrimenti non potrà usare il sito.

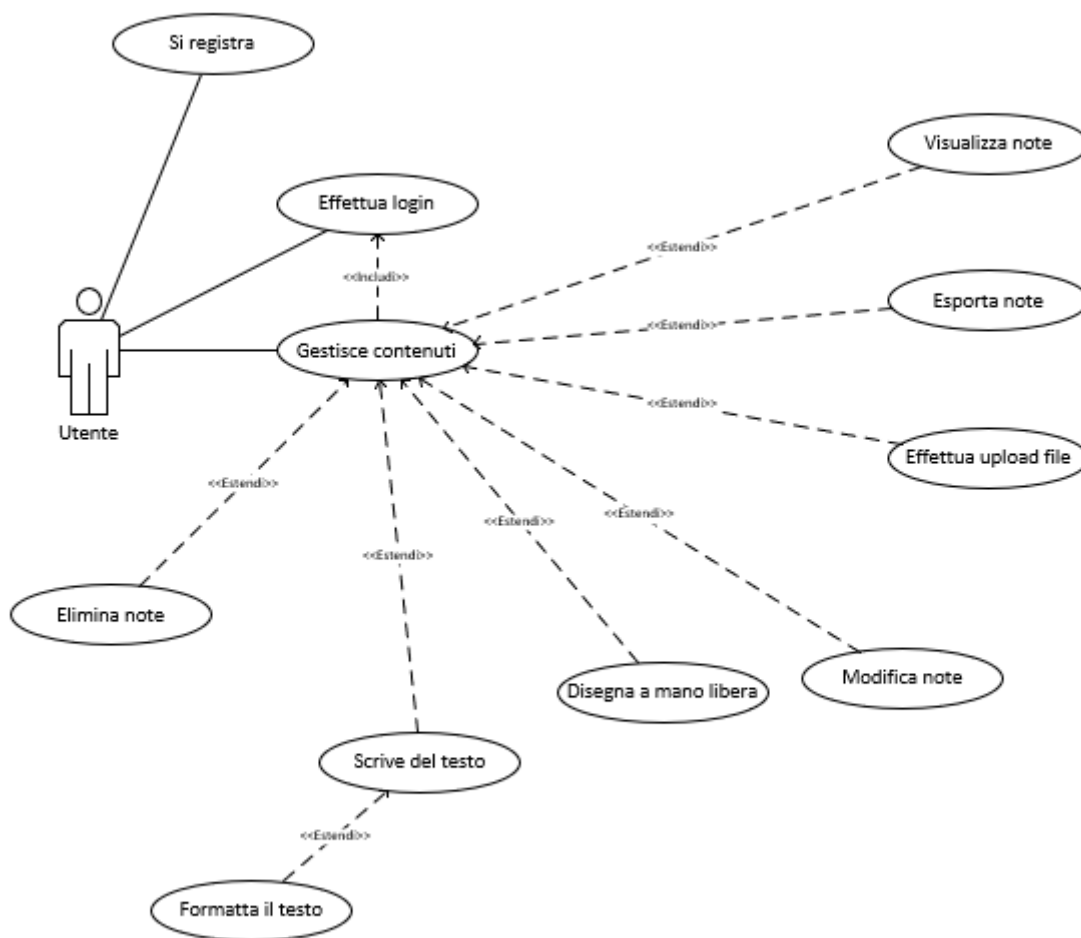



Figura 1 - Use Case

	SAMT – Sezione Informatica	Pagina 9 di 57
	CasaNote	

2.4 Pianificazione

Il seguente Gantt è stato utilizzato per dare una indicazione massima delle attività da svolgere e delle loro tempistiche, siccome noi utilizziamo il metodo Agile. Per la pianificazione oltre a Gantt abbiamo usato anche Trello per tenerci aggiornati per lo sprint.

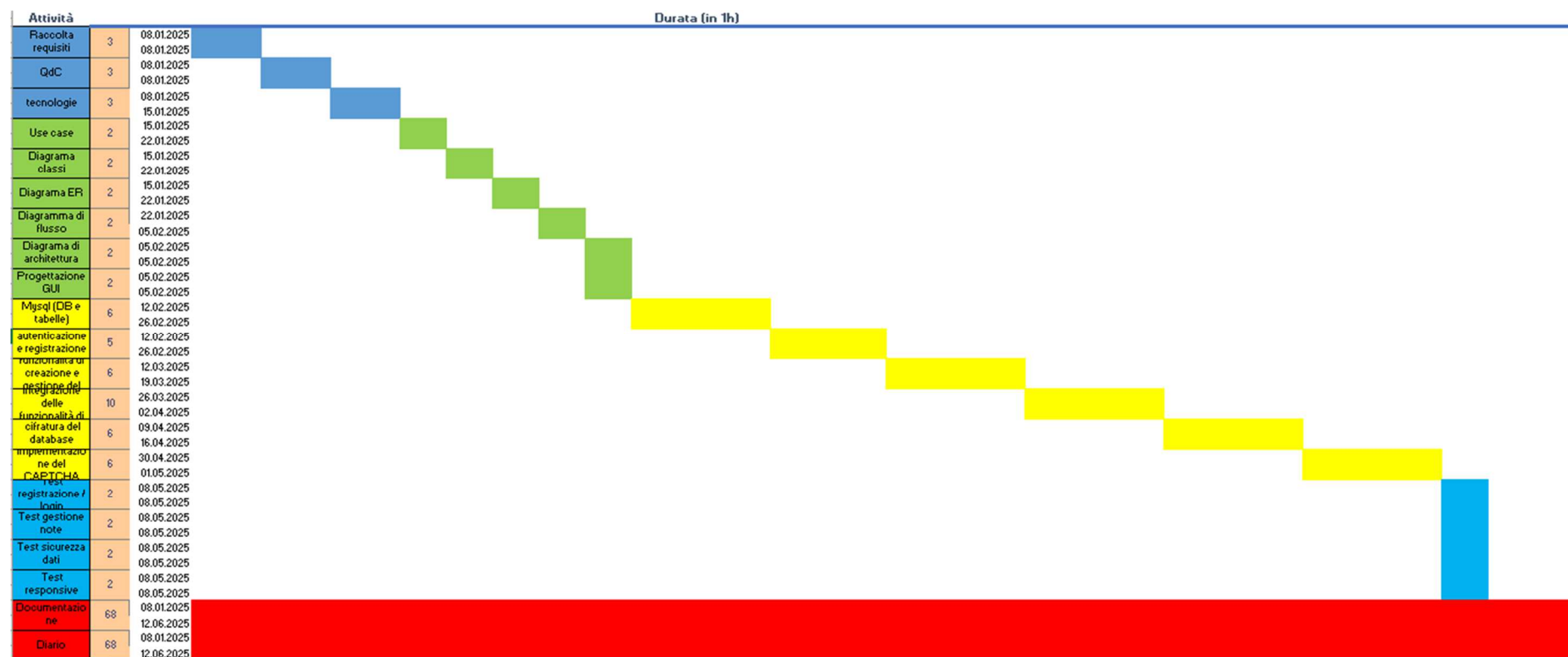


Figura 2 - Gantt

2.5 Analisi dei mezzi

2.5.1 Software

- Html
- Bootstrap
- php
- javascript
- css

2.5.2 Hardware

Per lo sviluppo del sito web avremmo bisogno di 1 computer della scuola con mouse e tastiera, il computer ha le seguenti caratteristiche:

- Versione 22H2 (build SO 19045.4780)
- Windows: Windows 10 Education
- RAM: 32 GB
- CPU: Intel® Core™ i7-13700 CPU @ 2.10 GHz
- GPU: NVIDIA T400 4GB

3 Progettazione

3.1 Design dell'architettura del sistema

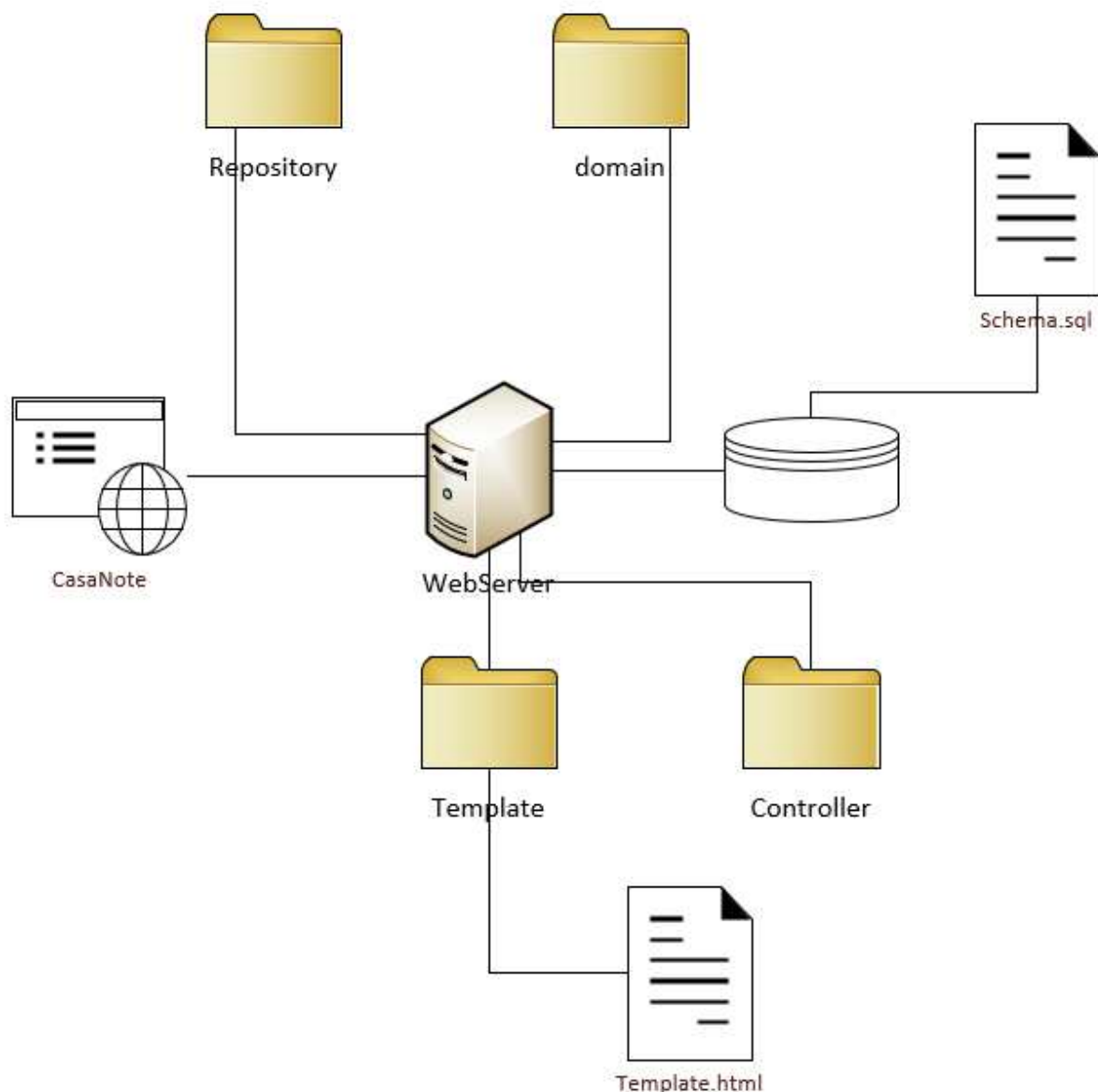


Figura 3 - Diagramma Architettura

L'architettura qui sopra descrive in linea di massima quanto e come dovrà essere implementata l'architettura del nostro progetto. Come si può vedere dall'immagine c'è un server centrale che gestirà le varie richieste e comunicazioni tra il browser ed esso. All'interno del server abbiamo il database, e le varie cartelle per la gestione e il funzionamento del sito, tra le quali i template, contenente i file html delle pagine, i controller per la gestione delle richieste, i repository per interfacciarsi con il database e poi le classi di dominio.

3.1.1 Swimlanes

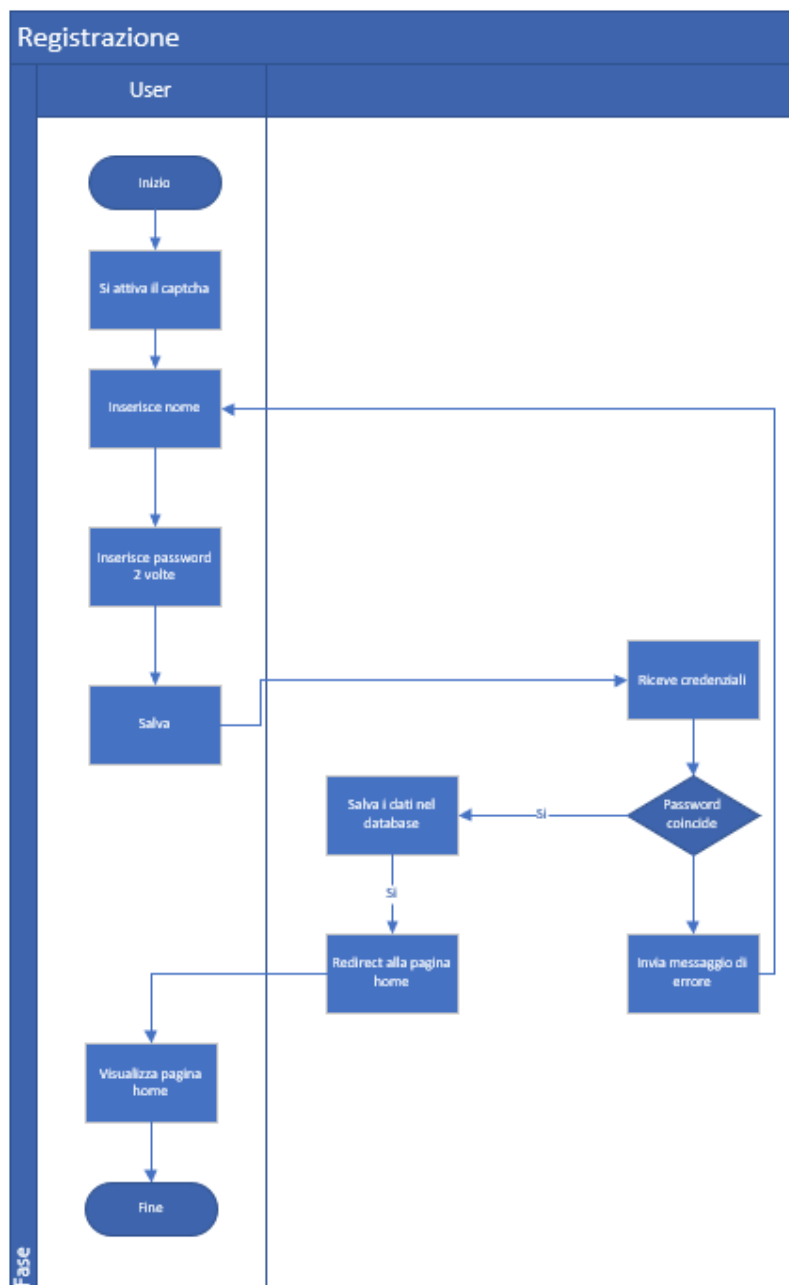


Figura 4 - Swimlanes registrazione

Questa swimlane è stata ideata per la registrazione di un nuovo utente. L'utente per fare l'accesso e poter usufruire del sito, dovrà prima essere registrato tramite nome utente e password. La password dovrà coincidere e il nome dovrà essere almeno di un carattere. Se la password non coincide allora verrà presentato un errore e l'utente ritorna alla pagina di registrazione. In caso di registrazione effettuata con successo l'utente visualizzerà la pagina home. Il captcha V3 verrà attivato alla pagina di registrazione per controllare che l'utente che ceca di registrarsi sia un umano e non un bot.

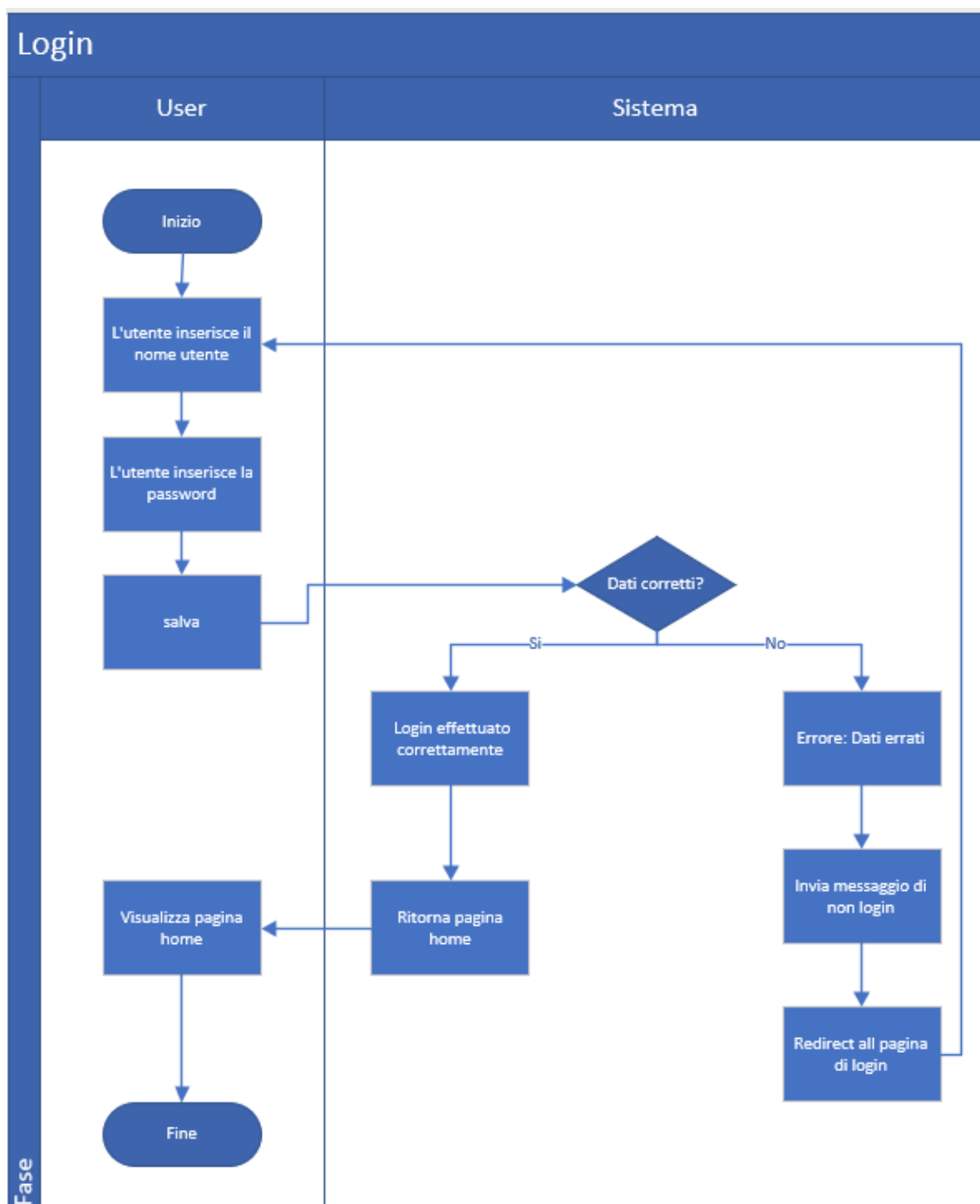


Figura 5 - Swimlanes login

Nello swimlane soprastante andiamo a analizzare il processo di login. Inizialmente l'utente interessato a logarsi avrà già completato il form di registrazione. Nella pagina di login, l'utente procede ad inserire il suo nome utente e password, dopodiché cliccherà sul pulsante conferma o salva, e i dati verranno inviati al server che controllerà se corrispondono ad un utente salvato nel DB, se non viene trovata una corrispondenza verranno inviati degli errori che poi verranno visualizzati, e l'utente verrà riportato alla pagina di login, mentre in caso contrario l'utente verrà fatto entrare nel sito e visualizzerà la pagina home.

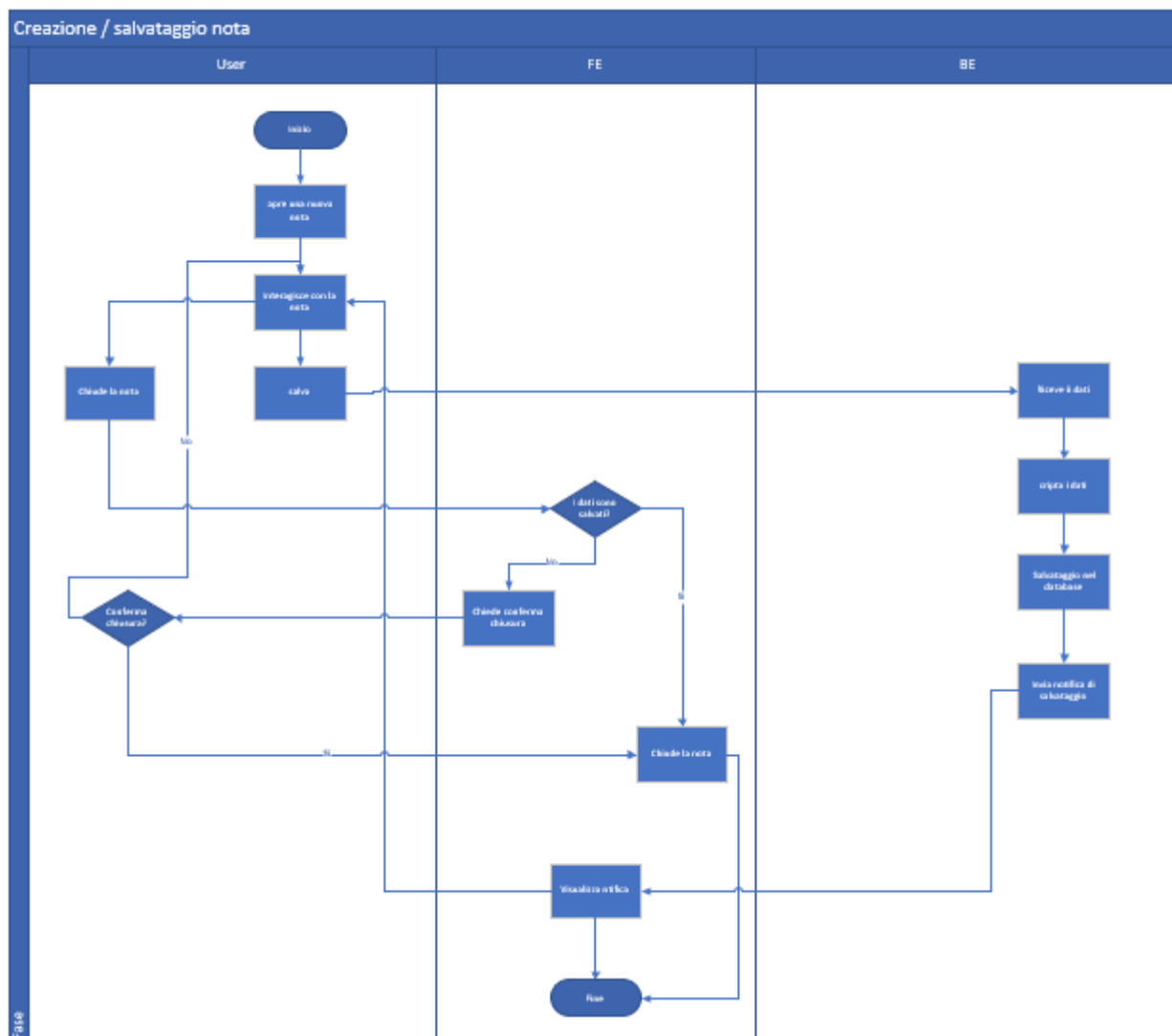


Figura 6 - Swimlanes creazione/salvataggio nota

In questo schema di swimlane viene mostrato come è stato pensato il salvataggio e creazione di una nota. In questo schema abbiamo tre componenti, l'utente il FE (front-end) e il BE (back-end). L'utente inizierà creando una nuova nota che potrà successivamente riempirla con vari contenuti (testuali, audio, immagini o file). Se l'utente chiude la nota verrà fatto un controllo se i dati sono salvati nel DB, e se non lo sono l'utente verrà chiesta la conferma di salvataggio. L'utente decide se salvarla o meno, se decide di no, allora torna a modificarla altrimenti la pagina della modifica della nota si chiude e l'utente visualizza la notifica del salvataggio della nota e torna alla home. Se l'utente mentre sta modificando la nota decide di cliccare sul pulsante salva allora il BE riceve i dati li cripta e li inserisce nel DB e l'utente viene notificato del salvataggio.

3.2 Design dei dati e database

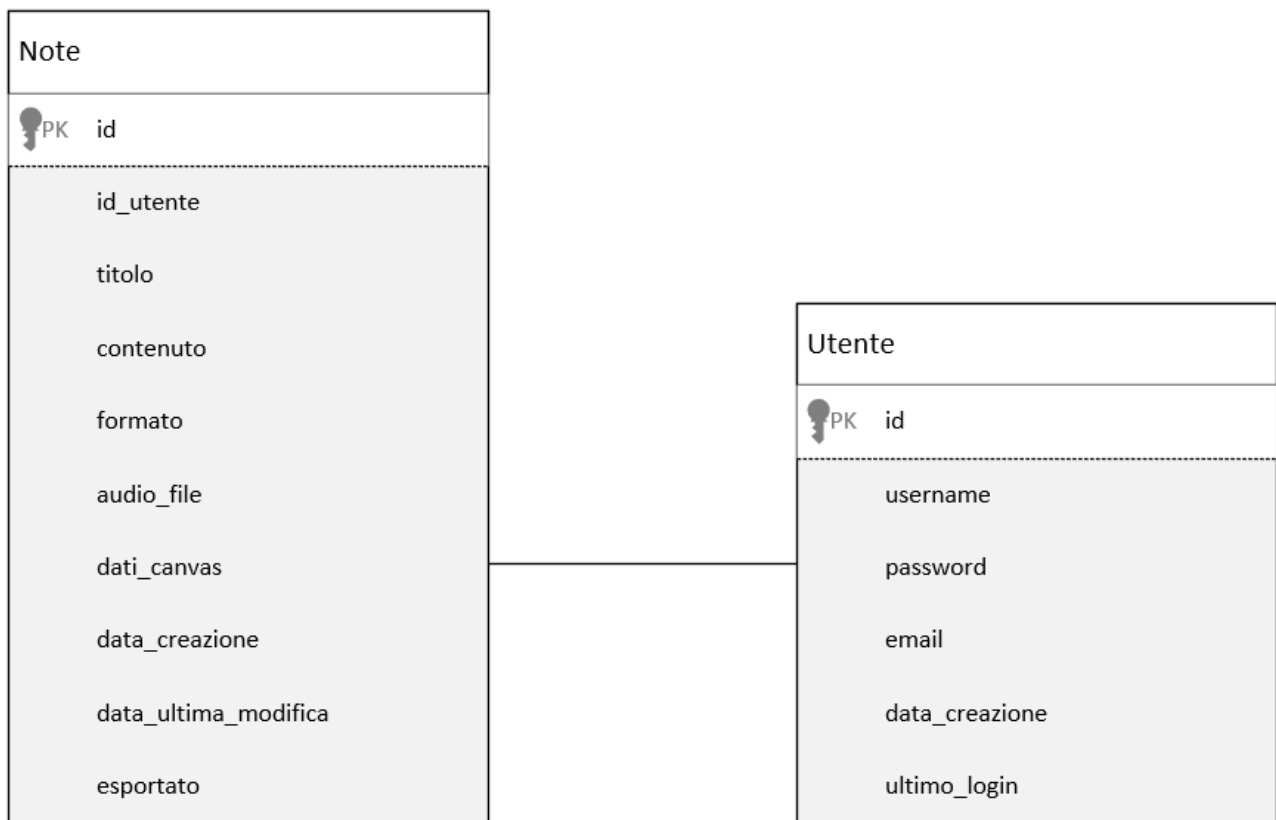


Figura 7 - Diagramma Classi

Sopra possiamo vedere come è stata progettata la gestione delle classi in questo progetto. A sinistra abbiamo la classe Note, che contiene vari attributi, tra cui l'id, che rappresenta la chiave primaria con cui verrà identificata ogni singola nota, il titolo (di tipo string), il contenuto (di tipo string), il formato (che può essere immagine o testo), l'audio_file (nel caso si necessiti di salvare un audio), il data_canvas (nel caso di un'immagine o disegni), la data_creazione di tipo date (che indica la data in cui è stata creata la nota), e infine esportato, che è un attributo di tipo boolean e avrà valore true se la nota è stata esportata.

3.3 Design delle interfacce

Queste interfacce non sono uguali a quelle effettive del progetto.

3.3.1 Registrazione utente

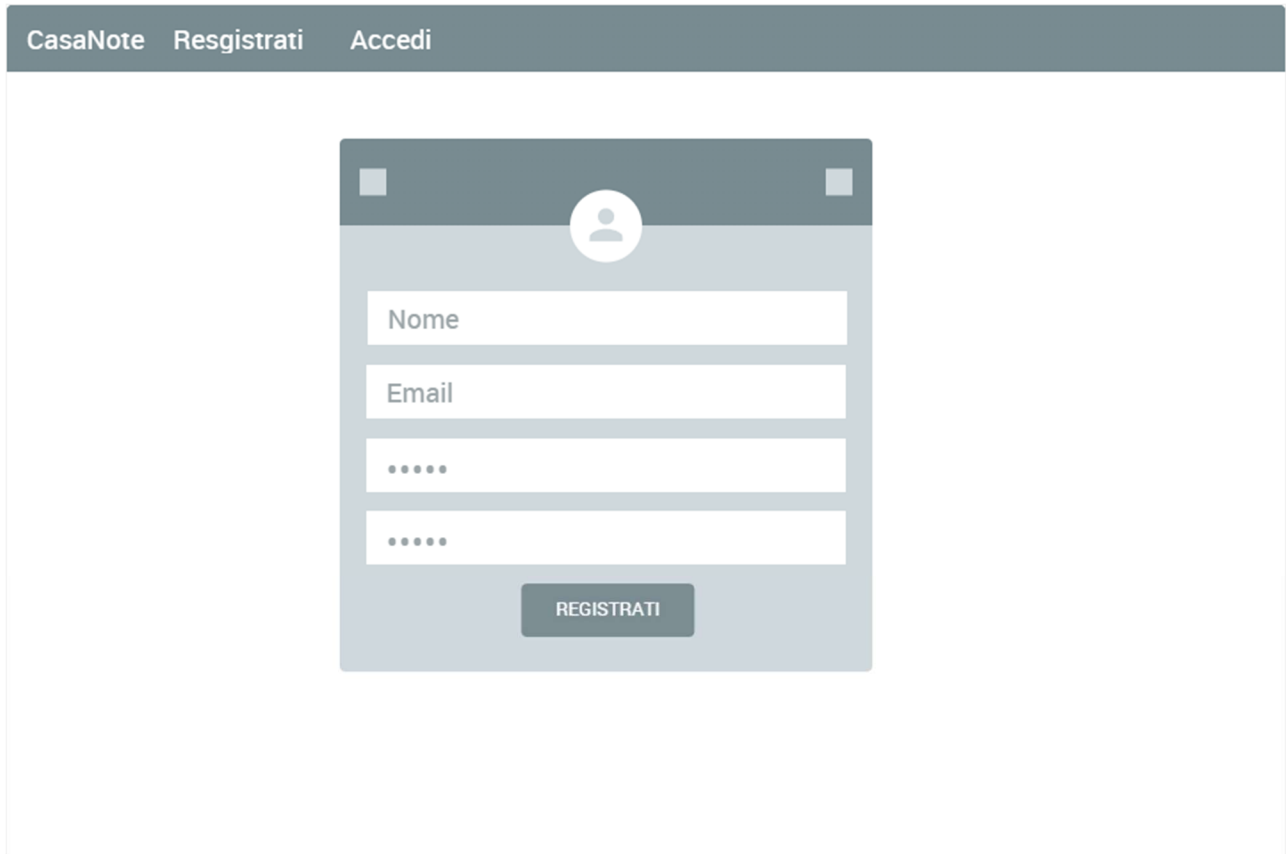


Figura 8 - Design registrazione utente

Quest'interfaccia è la pagina di registrazione l'utente inserisce username, email e password, se l'utente li immette in modo corretto dunque, senza errori, dopo aver inviato i dati con "registrati", viene salvato nel database e reindirizza nella pagina di home. Nel caso abbia già l'account può andare sulla pagina di login usando il collegamento alla pagina di accesso.

3.3.2 Login utente

CasaNote Resgistrati Accedi

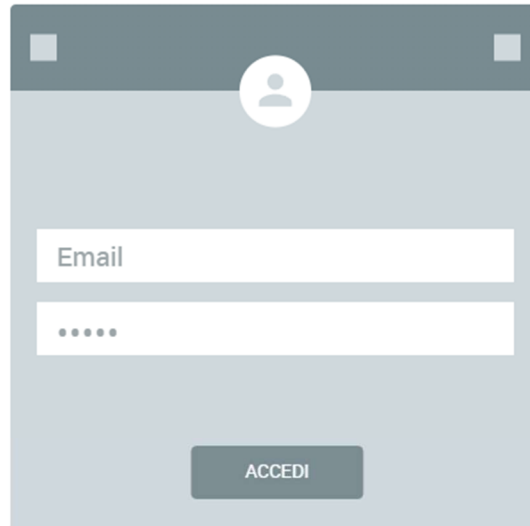
A user login form design. It features a dark grey header bar with a white user icon in the center. Below the header, there are two white input fields: the first is labeled 'Email' and the second contains five dots, indicating a password field. At the bottom of the form is a dark grey button with the text 'ACCEDI' in white capital letters.

Figura 9 - Design login utente

Questa è la pagina di login, dove l'utente mette i dati che ha usato per registrarsi, che dopo essere stati verificati, si accede alla pagina home del sito. Nel caso non abbia l'account può andare sulla pagina di registrazione.

3.3.3 Home

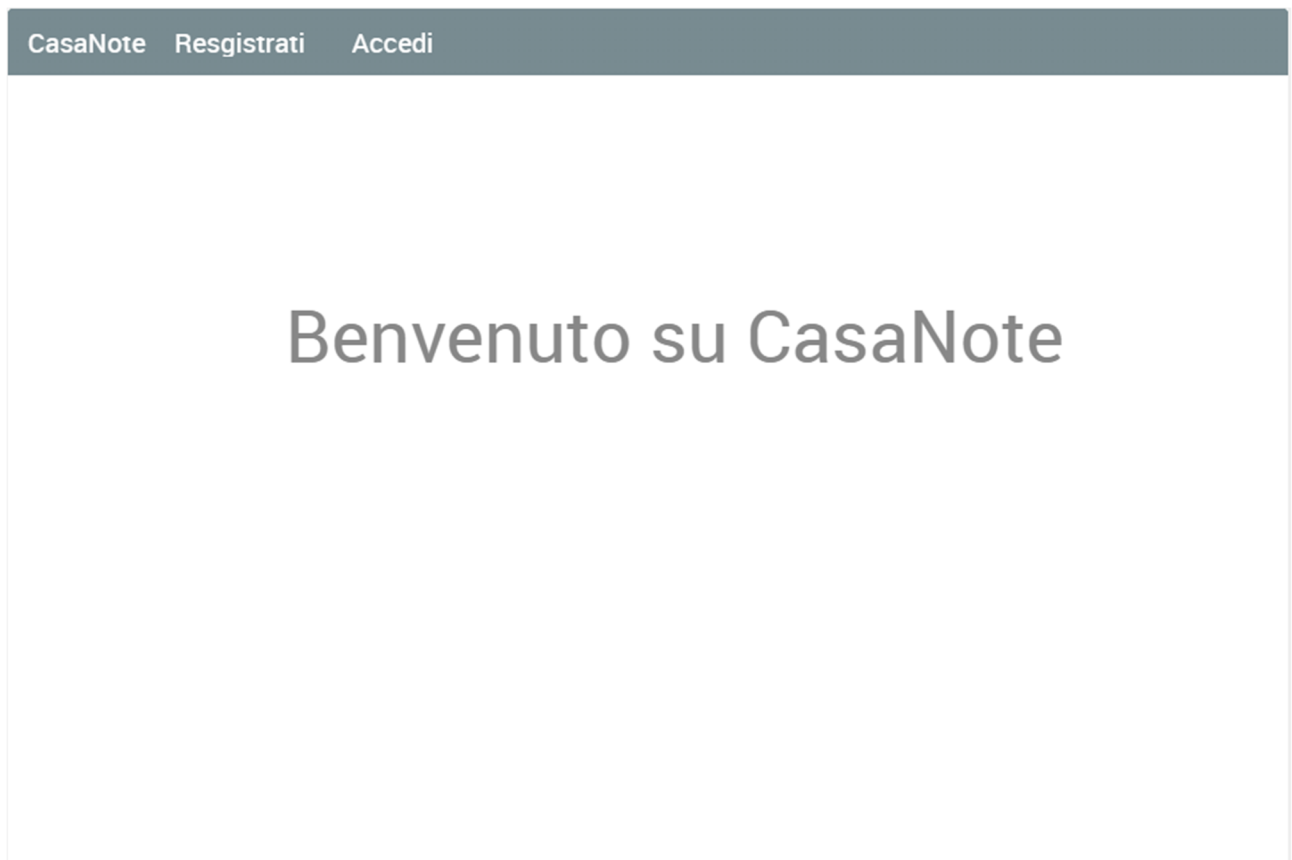


Figura 10 - Design home

Questa pagina è la home, in questa pagina si può scegliere se accedere o registrarsi. Questa è la pagina home, dove l'utente nel caso di percorso sbagliato viene reindirizzato.

3.3.4 Creazione nota

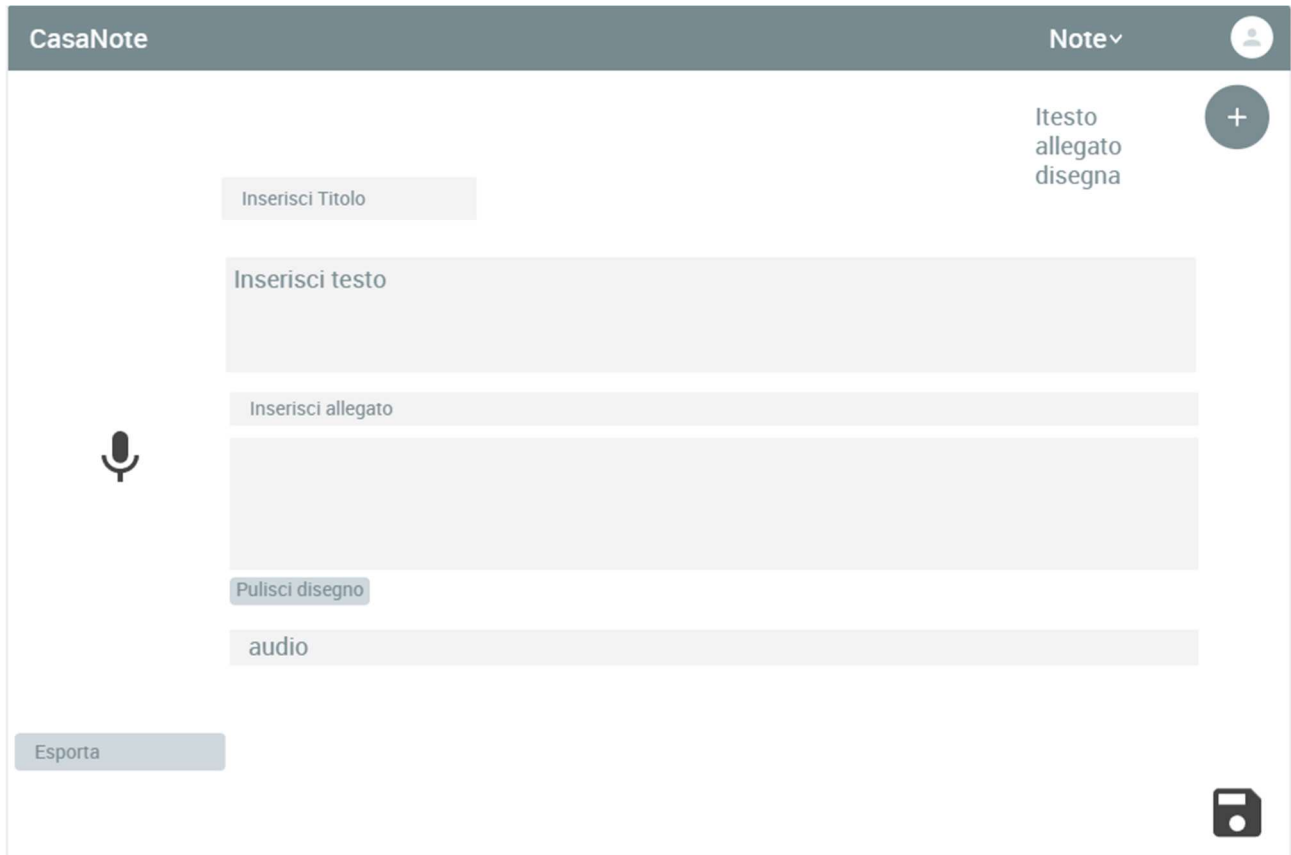


Figura 11 - Design creazione nota

Questa è la pagina dove viene creata la nota, la nota usando l'icona in basso a destra, viene salvata. Usando l'input in alto a destra, si accede alla funzionalità della pagina. Nella nota si può scrivere e disegnare. Nella nota si può importare un allegato, usando l'icona in basso a sinistra, che esportare in file di immagine (.PNG), in file pdf e in file di testo (.txt), solo se non si disegna nella nota. Ogni volta che si usa una funzionalità, viene aggiunto un riquadro.

3.3.5 Visualizzazione note

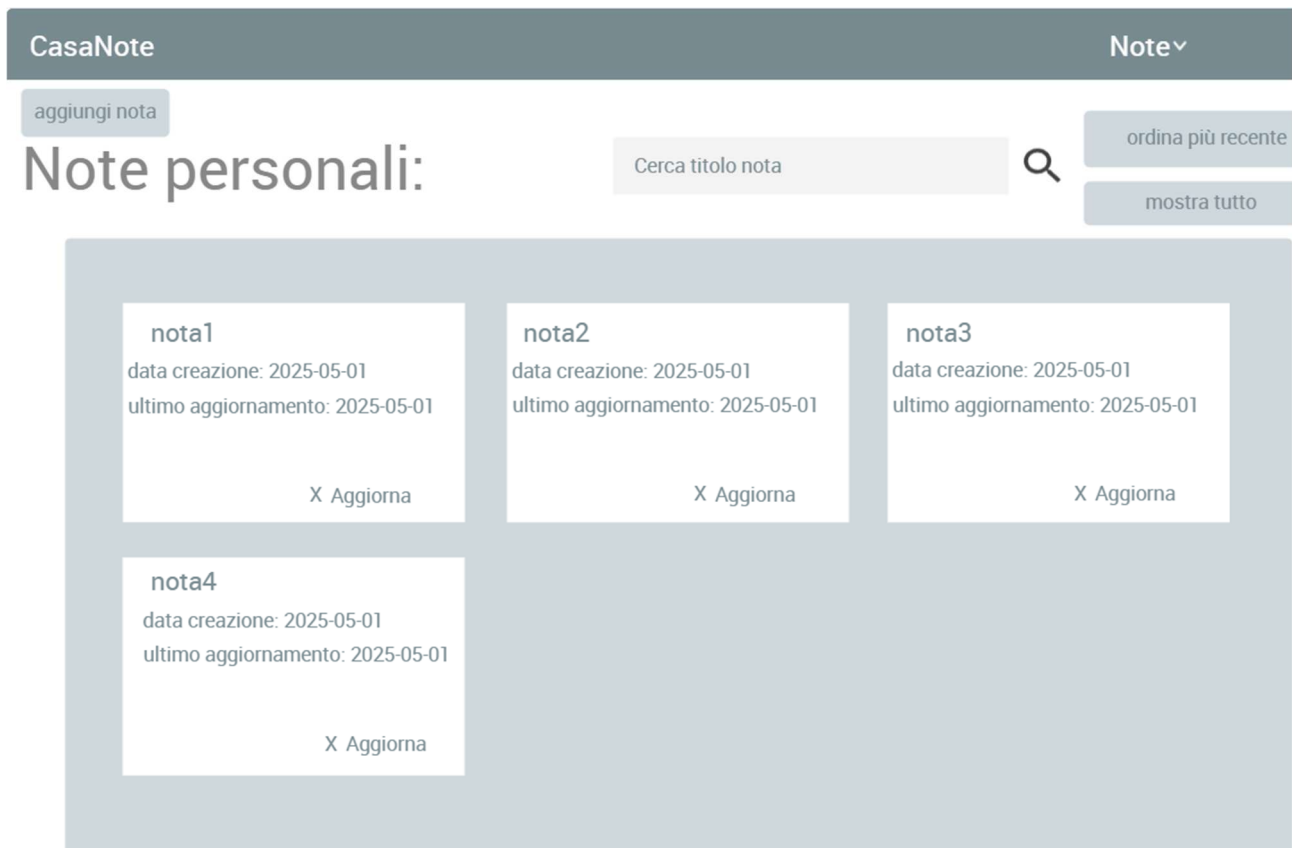


Figura 12 - Design visualizzazione note

In questa pagina viene mostrata la lista con i titoli e date delle proprie note create e salvate in precedenza. Per visualizzare una singola nota serve cliccare sopra quella che si vuole usare. Le note si possono filtrare in base alla data e cercare in base al titolo. Inoltre le note si possono eliminare, aggiungere e aggiornare.

4 Implementazione

4.1 Sistema di sicurezza

La classe security gestisce la crittografia e la decrittografia dei dati sensibili dell'intero database di CasaNote.

Per le operazioni di crittografia e decrittografia è utilizzato l'algoritmo AES-256-CBC.

La classe è composta da due metodi:

`encrypt($data):`

Il metodo cripta i dati utilizzando l'algoritmo AES-256-CBC e aggiunge il salting, ovvero l'IV casuale.

Parametri:

`$data`: mixed - dati da crittografare

Return:

string – Stringa in formato Base64 crittografata.

`Decrypt($data):`

Il seguente metodo prende i dati criptati e li decripta.

Parametri:

`$data` : mixed - dati crittografati in Base64 da decrittografare

Return:

mixed

4.2 Sistema di log

La classe logger gestisce il sistema di logging delle varie informazioni. È un sistema avanzato usato non solo nella fase di test. Serve per monitorare l'applicativo e per il debug.

Livelli di log:

1. INFO - Per le informazioni generali
2. NOTICE - Serve per notificare l'avvenimento di eventi o operazioni significative, come l'eliminazione di dati dal db
3. WARNING - Servono per tener traccia di eventi non critici
4. ERROR - Tengono traccia degli errori
5. FATAL - Errori critici che bloccano l'applicativo

La classe logger è un elemento importante per lo sviluppo dell'applicativo, permette di tener traccia e di monitorare le varie operazioni, tramite i log. Grazie all'automazione del sistema di log, è stato facile poter trovare errori, tracciare malfunzionamenti delle operazioni (inserimento di dati, o eliminazione di essi sul db) o falle del sistema.

4.3 Sistema di validazione

La classe validator fornisce le funzioni per garantire la sicurezza e l'integrità dei dati inseriti dall'utente nei form. Evita le varie injection, bloccandole.

Presenta diversi metodi per la validazione di dati.

4.3.1 Metodi

`validate($password):`

Il seguente metodo valida una password in base a dei criteri che definiscono come deve essere creata una password in modo che sia abbastanza forte.

Criteri:

- Lunghezza minima di 6 caratteri
- Almeno una lettera maiuscola
- Almeno una lettera minuscola
- Almeno un numero
- Almeno un carattere speciale

Parametri:

`$password (string)` – La password che imposta l'utente viene inviata come parametro

Return:

`bool` – true se la password corrisponde con tutti i criteri, false se almeno ad un non corrisponde.

`sanitizeInput($data):`

L'obiettivo di questa funzione è quello di pulire i dati ricevuti dai form in modo da eliminare eventuali caratteri non desiderati che possono creare danni al sistema.

Pulisce i dati di input da potenziali minacce come XSS.

Esegue varie operazioni sul parametro in entrata:

1. Rimozione degli spazi iniziali e finali
2. Rimozione dei backslash
3. Conversione dei caratteri speciali in HTML

Parametri:

`$data (string)` – dati ricevuti dall'utente che ha compilato i form

Return:

`$data (string)` – il parametro in entrata ma sanificato

`validateEmail($data):`

La seguente funzione verifica se il formato dell'email è corretto.

Parametri:

`$data (string)` – Indirizzo email dell'utente da validare

Return

`Bool` – true se la email è valida, false se non è valida

Per validare la email viene utilizzato il seguente filtro: `FILTER_SANITIZE_EMAIL`.

4.4 Classi Mapper

Il sistema di mapping implementa il pattern Data Access Object (DAO) per la gestione dell'accesso ai dati del database in modo sicuro e strutturato.

Le classi mapper forniscono per ogni tabella del database metodi per eseguire operazioni CRUD.

Tutte le operazioni di aggiornamento o creazione sono messe in sicurezza tramite l'utilizzo dei prepared statement e dei bind param che permettono di separare la logica mysql da quella di back-end.

Alcuni metodi importanti delle classi mapper:

Classe: noteMapper

fetchAll() : array

Seleziona tutte le note dell'utente loggato (recuperabile tramite la sessione) e le ritorna sotto forma di array. La funzione decrittografa automaticamente i titoli delle note.

Classe AttachmentMapper

saveAttachmentToDatabase(\$fileName, \$filePath, \$mimeType, \$noteId, \$attachmentType)

Il metodo salva le informazioni relative all'allegato nel db.

Sicurezza applicata:

1. Crittografia del nome file e del percorso
2. Logging delle operazioni effettuate
3. Uso di prepared statements

Return:

bool – Se l'operazione di inserimento va a buon fine ritorna true, altrimenti false.

Classe userModel:

La classe userModel è la classe che fornisce i metodi per l'autenticazione dell'utente, per creare e per eliminare un utente.

verifyUser(\$email,\$password)

Questo metodo permette di verificare l'esistenza e le credenziali dell'utente nel database tramite la email che è univoca e la sua password.

La password è hashata tramite l'algoritmo PASSWORD_DEFAULT.

Parametri:

email : string – email dell'utente

password : string – password dell'utente in chiaro

Return:

Array | null – Ritorna un array con i dati dell'utente se la password e la email sono corrette (esistono) altrimenti la funzione ritorna null.

`registerUser($name, $email, $password, $password_confirm)`

Il metodo registra il nuovo utente al sito, salvando i suoi dati nel database.

Per poter registrare l'utente si verifica che la `$password` e la `$password_confirm` coincidono.

La password prima di essere salvata nel database viene hashata, inoltre viene fatto un controllo che l'email non sia ancora stata utilizzata.

Il metodo per implementare le varie operazioni fa uso dei prepared statment per separare la logica.

Parametri:

`name`: string – nome utente compilato nel form

`email`: string – email già sanificata

`password`: string – password sanificata

`password_confirm`: string – conferma della password

Return:

`true` | `null` – viene ritornato `true` se l'esecuzione dell'operazione di inserimento di un nuovo utente va a buon fine, altrimenti ritorna `null`.

4.5 Controllers

I controller sono responsabili dell'implementazione della logica di business dell'applicativo. Essi gestiscono le richieste dell'utente, coordina i vari componenti del sistema e garantisce la sicurezza delle operazioni.

Nel nostro progetto abbiamo implementato 5 controller, ognuno con delle aree di utilizzo specifiche.

Per esempio il controller manage lavora esclusivamente sulle note e i suoi allegati.

Infatti è responsabile della gestione delle operazioni principali relative a note e allegati.

4.5.1 Controller: manage

`saveOrUpdateNote($id=null)`

Il metodo salva una nuova nota, o ne aggiorna una vecchia a dipendenza se il parametro `$id` ricevuto sia null (si tratta di una nuova nota) oppure not null (corrispondente all'id della nota che si vuole aggiornare).

Come ogni metodo delle classi controller i parametri ricevuti vengono prima fatti passare all'interno dei metodi di validazione per sanificarli ed evitare di fare danni al database o malfunzionamenti dovuti a caratteri indesiderati contenuti nei dati.

Se l'id non viene passato allora si procede a creare la nuova nota, mentre se l'id esiste allora viene aggiornata la nota corrispondente all'id passato.

`saveAttachment()`

Il metodo `saveAttachment` è un altro metodo fondamentale della classe controller manage perché è la classe che si occupa della gestione del salvataggio degli allegati associati ad una nota. I dati degli allegati vengono inviati al server tramite una richiesta POST e poi ogni attachment viene salvato in locale all'interno della cartella `uploads/note_xx/` all'interno della nota su cui è stato creato.

Parametri POST attesi e il loro significato:

Parametri	Tipo	Descrizione
Attachment_type	String	Indica il tipo di allegato, ovvero se si tratta di una immagine (draw) di un testo (text) o di un file (file). È essenziale per poi andare a salvare il file con la giusta estensione.
Attachment_file	File	Se <code>attachment_file == file</code> , quindi se si tratta di un file inserito dall'utente, e non un testo creato nella nota o un disegno.
Attachment_content	String	Contenuto dell'allegato che può essere in formato testuale o base64 per le immagini.

Output:

In caso di successo dell'upload degli allegati viene stampato nel log un messaggio per tracciare l'avvenimento.

Mentre in caso di errori (di validazione utente, o di validazione dei dati passati o delle operazioni di creazione del file non andate a buon fine) viene registrato l'errore nei log e mostra la pagina d'errore:
application/views/_templates/error.php

4.5.1.1 Dettagli funzionamento

La gestione dei tipi di allegati è la seguente:

1. Per prima cosa si verifica l'esistenza `$_FILES['attachment_file']`
2. Se esiste allora viene spostato il file all'interno della cartella della nota grazie alla funzione:
`move_upload_file()`
3. Dopodiché vengono salvati i metadati nel database.

Se l'allegato è 'text' allora estrae il contenuto dall'input che ha scritto l'utente.
Dopodiché viene salvato il file .txt con timestamp e poi viene salvato il path e i suoi metadati nel database.

Se l'allegato è un disegno, quindi 'draw', allora viene estratto il contenuto base64 contenuto in `$_POST['attachment_content']`, poi viene decodificato:

```
base64_decode(str_replace(...))
```

E infine salva il file con l'estensione png e salva i metadati nel database.

Perché è stato scelto il formato PNG e JPG?

Il formato PNG è stato scelto per svariati motivi, innanzi tutto mantiene la qualità originale e non introduce degradazioni visive. In più supporta le trasparenze come i background ed è ottimizzato per le immagini contenenti pochi colori. Mentre se avessimo dovuto trattare immagini ad alta risoluzione con una maggior peso nella memoria allora avremmo dovuto utilizzare JPG che con la compressione permette di ridurre il peso dell'immagine, ma non è il caso nostro dato che l'utente può disegnare in un form di dimensioni 500x300px con un solo colore.

Gestione errori:

Condizione	Azione
Utente non loggato nell'applicativo	Si chiama il metodo di login che lo indirizza al form di login
<code>notelid == null</code>	Mostra un messaggio di errore, logga il messaggio di errore e ritorna al metodo <code>index()</code>
Creazione cartella fallita	Crea la cartella con <code>mkdir()</code>
Nessun file inviato	Mostra errore nel log
Tipo di allegato non valido	Mostra errore nel log

zip(\$id=null)

Il metodo zip() è il metodo che si occupa di esportare tutti i file associati ad una specifica nota all'interno di una archivio ZIP. Una volta che l'utente ha scaricato la cartella della nota compressa l'archivio viene eliminato dalla cartella tmp del server.

Anche in questo metodo come parametro abbiamo l'id della nota che deve per forza esistere altrimenti viene stampato un errore.

Sicurezza implementata nel metodo:

Controllo che l'utente sia loggato

```
if (!$this->validator->isUserLoggedIn()) { ... }
```

Verifica della richiesta (deve essere POST

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') { ... }
```

Verifica che l'id non sia null ed esista.

Come funziona?

Il funzionamento del metodo è il seguente:

per prima cosa si crea il nome della nota, e si costruisce il nome della cartella nel seguente modo:

```
$notaName = "note_{$id}";
```

```
$zipFileName = "{$notaName}.zip";
```

Ovviamente tutti i passaggi sono loggati.

In seguito viene istanziato un nuovo oggetto di tipo ZipArchive che ci permette di creare l'archivio ZIP. Viene impostato il percorso temporaneo su dove salvare l'archivio:

```
$zipPath = 'application/tmp/' . $zipFileName;
```

E poi si tenta di aprire o creare in caso non riesce ad aprire la cartella compressa, viene aggiunto il flag CREATE così che in caso la cartella non esiste allora viene creata.

In caso di errori di apertura/creazione allora viene mostrato un errore e la funzione termina.

Una volta creata la cartella ZIP, viene individuata la cartella contenente i file della nota:

```
$directory = 'uploads/' . $notaName;
```

e poi andiamo ad ottenere la lista di tutti i file presenti nella cartella, che poi vengono aggiunti all'archivio ZIP mantenendo solo il nome base senza il path:

```
$files = glob($directory . '/*');
```

Una volta che si ha l'archivio sulla cartella tmp, lo si deve inviare al server.

Vengono impostate le intestazioni http:

- Content-Type: indica che il file è un archivio ZIP.
- Content-Disposition: forza il download con nome file specifico.
- Content-Length: indica la dimensione del file per gestire il download.

E poi tramite la funzione readfile() si invia l'archivio al browser.

Una volta inviato il file zip l'archivio viene rimosso con la funzione unlink(), evitando accumuli di file temporanei che col tempo vanno a riempire spazio in memoria inutile.

4.6 Disegno su canvas

Lo script JavaScript per la gestione del canvas è presente all'interno del file manageCanvas.php.

Lo script permette all'utente di disegnare a mano libera su un canvas HTML.

L'utente può utilizzare la gomma per cancellare, pulire completamente il disegno o disegnare con il colore nero.

Variabili:

Nome	Tipo	Funzione
isDrawing	Bool	Controlla se il mouse è premuto (quindi true) senno è false
currentColor	String	Colore con cui l'utente disegna
isErasing	Bool	Indica se la modalità gomma per cancellare è attiva (true) o no (false)
lastX	Int	Ultima posizione nell'asse delle X del puntatore del mouse, server per tracciare la linea di disegno
lastY	Int	Ultima posizione nell'asse delle y del puntatore del mouse, server per tracciare la linea di disegno

Lo script presenta varie funzioni per la gestione del canvas.

initDrawing() – Funzione che inizializza l'area di disegno, impostando gli eventi del mouse (mousedown, mouseup, mousemove, mouseout..) sul canvas e imposta il pennello.

Configurazione canvas:

```
ctx.lineJoin = 'round';
ctx.lineCap = 'round';
ctx.lineWidth = 2;
```

Spessore della linea è di 2px, con lo stile degli angoli (giunzioni) 'round' e lo stile delle estremità a 'round'.

setColor(color) – funzione che imposta il colore per il pennello

Parametri:

color: String – indica il colore desiderato dall'utente in formato HEX.

enableEraser() – funzione che permette l'utente di attivare la modalità gomma e cancellare il disegno.

La gomma non è altro che il cambio del colore del pennello, impostando il suo colore a bianco (ffffff).

Alcune considerazioni che possiamo trarre da questa implementazione del canvas è che il canvas non supporta ancora il touch, ma solo l'input del mouse, la cancellazione del canvas tramite gomma è hardcoded su bianco, quindi la gomma ha effetto di cancellazione solo sullo sfondo bianco. Lo scaling del canvas non è gestito in base al dispositivo.

4.7 Script gestione input

Lo script `manageInput` serve per gestire gli input degli allegati a livello di front-end, andando ad aggiungere dinamicamente input testuali, di file o canvas. Lo script permette anche di rimuovere i vari input ancora non salvati nel database.

Inoltre è presente anche un blocco di codice dedicato al salvataggio degli allegati tramite le richieste AJAX al server.

Le richieste lato back-end sono poi gestite dal controller `manage`.

Le sue funzionalità:

`addInput(type)` – aggiunge al DOM dinamicamente un input in base al tipo di input che ha cliccato l'utente.

Parametri:

`type: string` – `text` può essere `text` | `attachment` | `draw` ed indica il tipo di input da aggiungere.

Funzionalità:

Inserisce gli elementi nel container `#dynamic-fields`

Ogni blocco ha un pulsante “X” per la rimozione (`removeInputFromDOM`)

I bottoni “Salva” richiamano le rispettive funzioni di salvataggio:

- `saveTextAttachment()`
- `saveFileAttachment()`
- `saveDrawing()`

`removeInputFromDOM(button)`

La seguente funzione rimuove l'input associato al bottone “X” per l'eliminazione.

Infatti usa `closest('.dynamic-input')` per trovare e rimuovere l'input.

Funziona solo per gli elementi ancora non salvati.

`removeAttachmentFROMdb(attachmentId)`

Invia una richiesta POST al server per rimuovere un allegato esistente (già salvato) dal database.

Parametri:

`attachmentId (number)`: identificativo dell'allegato che deve essere eliminato

Funzionalità:

Mostra un `confirm()` per chiedere al conferma dell'eliminazione.

Invia l'ID al server (`manage/deleteAttachment`)

Se l'operazione ha successo, rimuove il relativo elemento dal DOM.

4.8 Views

Le pagine sono state implementate utilizzando bootstrap 5.3.0, framework che ci ha permesso di gestire il responsive ed avere delle pagine più ordinate e belle da vedere.

Nella cartella view/manage/ troviamo le view dedicate alla reazione ed aggiornamento delle note e dei loro allegati.

La pagina createNote.php fornisce un'interfaccia web per creare, modificare e visualizzare note con allegati di vario tipo (testo, file, disegni) e permette di esportare la nota con tutti i suoi allegati in un file ZIP.

Il file include i vari script JS per la gestione degli input:

```
include 'application/views/_templates/static/js/manageInput.php';
include 'application/views/_templates/static/js/manageCanvas.php';
```

Ci sono 2 form :

Il form principale per il salvataggio e aggiornamento della nota:

```
<form method="POST" action="<?php echo URL; ?>manage/saveOrUpdateNote<?php if (isset($note)):
echo '/' . $note->getId(); endif; ?>">
```

Con richiesta POST e URL di destinazione al metodo saveOrUpdateNote.

Il dropdown per aggiungere gli allegati viene abilitato una volta salvata la nota.

Gli allegati vengono visualizzati in questo modo:

Elenco degli allegati associati alla nota, ciascuno con:

Id come attributo data-id.

Contenuto mostrato in base al MIME type:

Immagini (image/*): mostrati dentro il tag .

Testi (text/*): contenuto mostrato in <pre>.

Altri file: mostrato nome file e percorso.

Pulsante di rimozione allegato che richiama removeAttachmentFROMdb(id).

Se non ci sono allegati, mostra messaggio "Inizia a creare!".

Il secondo form è quello di esportazione:

il pulsante esporta è attivo solo quando la nota è salvata e permette di chiamare il metodo zip() per scaricare la nota.

4.9 Struttura

La struttura del progetto utilizzata è MVC. La scelta non è solo dovuta al fatto perché è l'unica struttura che abbiamo imparato a PHP, ma anche perché la troviamo molto semplice da utilizzare soprattutto per la separazione delle responsabilità, quindi ogni componente (model view e controller) hanno compiti specifici e diversi.

Nella nostra struttura abbiamo aggiunto alcune particolarità. Nel nostro caso fuori dalla cartella application troviamo la cartella upload che contiene tutte le note con i suoi allegati.

Nella cartella _template/static/js troviamo invece tutti gli script JS utilizzati dalle view.

Oltre al Javascript Nella cartella _templates sono presenti i file di intestazione delle pagine, come header.php, e anche il footer.php, oppure le navbar (navbar2.php e navbar.php) in questo modo tutte le pagine sono coerenti e seguono tutte lo stesso stile.

Nel nostro MVC sotto config c'è il file config.php, questo file è il file di configurazione generico per l'applicazione, dove si possono definire delle variabili a cui hanno accesso tutti i file dell'applicativo, delle variabili globali come URL, o i parametri di collegamento al database come la porta, la password, lo username e il database.

Nella cartella libs sono contenute le varie librerie dell'applicazione come ad esempio la cartella del framework di bootstrap, oppure le classi che servono per automatizzare la sicurezza o il sistema di logging come per esempio validator.php o la classe logger.

Dentro la cartella application è presente anch'ella cartella tmp, con lo scopo di creare dei file ZIP temporanei da inviare al browser ogni volta che l'utente vuole fare il download di una nota, ovviamente una volta inviati verranno cancellati.

Infine è presente anche il file .htaccess ovvero il file che reindirizza tutte le richieste su index.php. Index.php è il file iniziale a cui arriva ogni richiesta.

4.9.1 Flusso tipico nel progetto

Utente visita la pagina nota:

Il Controller recupera i dati dal Model (note e attachments) e carica la View che mostra i dati.

Utente modifica o crea una nota:

Invia il form al Controller (es. saveOrUpdateNote), che aggiorna o crea la nota nel database via Model.

Utente aggiunge allegati o disegni:

Funzioni JS dinamiche aggiungono campi al form. I salvataggi specifici avvengono cliccando sul pulsante aggiorna, che aggiorna i dati.

Utente rimuove un allegato:

Viene inviata una richiesta al Controller per eliminare l'allegato, che aggiorna il Model e risponde alla View per aggiornare l'interfaccia.

Utente esporta la nota:

Il form per esportazione ZIP invia una richiesta al Controller, che crea un archivio ZIP contenente la nota e gli allegati e lo fornisce per il download.

5 Test

5.1 Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l'adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

Test Case:	TC-001	Nome:	Login utente con campi vuoti
Riferimento:	REQ-02		
Descrizione:	L'utente se non inserisce mail o password, l'applicativo avvisa che è presente un errore.		
Prerequisiti:	Sistema pronto per gestire questo problema.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente non inserisce la email 2. L'utente non inserisce la password 3. L'utente invia i dati 		
Risultati attesi:	Appare un avviso dove obbliga all'utente di inserire qualcosa nei campi.		

Test Case:	TC-002	Nome:	Login utente con valori non presenti
Riferimento:	REQ-02		
Descrizione:	L'utente nel caso prova ad autenticarsi con mail e password non presenti nel database, l'applicativo avvisa che è presente un errore.		
Prerequisiti:	Sistema pronto per gestire questo problema.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente inserisce la email 2. L'utente inserisce la password sbagliata 3. L'utente invia i dati 		
Risultati attesi:	Appare un avviso dove informa l'utente che questi valori inseriti di nome e password non sono presenti.		

Test Case:	TC-003	Nome:	Login utente con successo
Riferimento:	REQ-02		
Descrizione:	L'utente si autentica con nome mail e password, presenti del database, l'utente deve accedere alla pagina home.		
Prerequisiti:	Sistema pronto per gestire questo problema.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente inserisce la email 2. L'utente inserisce la password 3. L'utente invia i dati 		
Risultati attesi:	Appare la pagina della home dopo che il sistema ha verificato che l'utente sia presente nel sistema.		

Test Case:	TC-004	Nome:	Registrazione utente con campi vuoti
Riferimento:	REQ-03		
Descrizione:	Un nuovo utente se non inserisce nome, mail o password, l'applicativo avvisa che è presente un errore.		
Prerequisiti:	Sistema pronto per gestire questo problema.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente non inserisce il nome e la email 2. L'utente non inserisce la password 3. L'utente invia i dati 		
Risultati attesi:	Appare un avviso dove obbliga all'utente di inserire qualcosa nei campi.		

Test Case:	TC-005	Nome:	Registrazione utente con successo
Riferimento:	REQ-03		
Descrizione:	Un nuovo utente si registra con nome, email e password, il database deve aggiungere l'utente.		
Prerequisiti:	Sistema pronto per gestire questo problema.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente inserisce il nome e la email 2. L'utente inserisce la password 3. L'utente invia i dati 		
Risultati attesi:	Appare la pagina della home dopo che il sistema ha verificato che l'utente sia stato aggiunto correttamente nel sistema.		

Test Case:	TC-006	Nome:	Creazione e salvataggio di una nota
Riferimento:	REQ-04 REQ-08		
Descrizione:	Il test consiste nel creare una nota, il db dovrebbe aggiungere la nota, dopo aver inserito il titolo, di conseguenza l'elenco delle note dovrebbe mostrare la nuova nota creata.		
Prerequisiti:	Sistema pronto, con db, pagina dell'elenco delle note e pagina creazione note, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina dell'elenco delle note 2. L'utente va nella sezione di creazione della nota cliccando su "+Aggiungi Nota" 3. L'utente inserisce un titolo alla nota e salva la nota 4. L'utente torna alla pagina principale cliccando "Home" 		
Risultati attesi:	Appare l'elenco delle note dopo aver aggiunto la nota.		

Test Case:	TC-007	Nome:	Creazione e salvataggio di una nota senza titolo
Riferimento:	REQ-04 REQ-08		
Descrizione:	Il test consiste nel creare una nota senza mettere il titolo alla nota, il sistema dovrebbe riscontrare un problema e mostrare l'errore.		
Prerequisiti:	Sistema pronto, pagina dell'elenco delle note e pagina creazione note, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina dell'elenco delle note 2. L'utente va nella sezione di creazione della nota cliccando su "+Aggiungi Nota" 3. L'utente non inserisce un titolo alla nota 4. L'utente prova salvare la nota 		
Risultati attesi:	Appare la pagina della creazione nota mostrando l'errore.		

Test Case:	TC-008	Nome:	Modifica titolo di una nota
Riferimento:	REQ-04		
Descrizione:	Il test consiste nel modificare il titolo di una nota, di conseguenza l'elenco delle note dovrebbe mostrare il nuovo titolo della nota aggiornata.		
Prerequisiti:	Sistema pronto, pagina dell'elenco delle note e pagina creazione note, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina dell'elenco delle note 2. L'utente va nella sezione di modifica la nota cliccando su "Aggiorna" 3. L'utente inserisce un nuovo titolo alla nota 4. L'utente aggiorna la nota 		
Risultati attesi:	Appare la pagina dell'elenco delle note dopo aver modificato la nota, la nota con il nuovo titolo dovrebbe apparire nell'elenco.		

Test Case:	TC-009	Nome:	Eliminazione di una nota
Riferimento:	REQ-05 REQ-08		
Descrizione:	Il test consiste nell'eliminare una nota dall'elenco delle note, il db dovrebbe eliminare la nota, di conseguenza l'elenco delle note non dovrebbe mostrare più		
Prerequisiti:	Sistema pronto, con db, pagina dell'elenco delle note, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina dell'elenco delle note 2. L'utente elimina la nota cliccando la "X" 		
Risultati attesi:	Appare la pagina dell'elenco delle note dopo aver eliminato la nota, la nota non dovrebbe apparire nell'elenco.		

Test Case:	TC-010	Nome:	Ricerca di una nota
Riferimento:	REQ-04 REQ-08		
Descrizione:	Il test consiste nel cercare il titolo o una parte del titolo di una nota, di conseguenza l'elenco delle note dovrebbe mostrare le note con quella parte di testo inserito nel campo di ricerca.		
Prerequisiti:	Sistema pronto, pagina dell'elenco delle note, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina dell'elenco delle note 2. L'utente esegue una ricerca inserendo il testo 		
Risultati attesi:	Appare la pagina dell'elenco delle note con il titolo della nota che comprende il testo inserito.		

Test Case:	TC-011	Nome:	Ricerca di una nota senza titolo
Riferimento:	REQ-04 REQ-08		
Descrizione:	Il test consiste nel cercare il titolo o una parte del titolo di una nota senza inserire del testo nel campo di ricerca, di conseguenza l'elenco delle note dovrebbe mostrare tutte le note.		
Prerequisiti:	Sistema pronto, pagina dell'elenco delle note, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina dell'elenco delle note 2. L'utente esegue una ricerca senza inserire il testo 		
Risultati attesi:	Appare la pagina dell'elenco delle note con tutte le note.		

Test Case:	TC-012	Nome:	Ripristino della ricerca
Riferimento:	REQ-04 REQ-08		
Descrizione:	Il test consiste nel riportare l'elenco delle note con tutte le note, di conseguenza l'elenco delle note dovrebbe mostrare tutte le note.		
Prerequisiti:	Sistema pronto, pagina dell'elenco delle note, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina dell'elenco delle note 2. L'utente ripristina il filtro delle note con un button che mostra tutte le note 		
Risultati attesi:	Appare la pagina dell'elenco delle note con tutte le note.		

Test Case:	TC-013	Nome:	Ricerca con filtri
Riferimento:	REQ-04 REQ-08		
Descrizione:	Il test consiste nel riportare l'elenco delle note con tutte le note, in ordine di data dalla più recente alla meno recente.		
Prerequisiti:	Sistema pronto, pagina dell'elenco delle note, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina dell'elenco delle note 2. L'utente usa il filtro delle note con un button che ordina le note in ordine dalla più recente 		
Risultati attesi:	Appare la pagina dell'elenco delle note con tutte le note in ordine dalla più recente		

Test Case:	TC-014	Nome:	Sicurezza sistema – Cross-Site Scripting (XSS)
Riferimento:	REQ-03		
Descrizione:	Il test consiste nell' inserire nel campo di registrazione del nome uno script un codice, che genera un alert se vulnerabile.		
Prerequisiti:	Sistema pronto, pagina di registrazione, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina di registrazione 2. L'utente inserisce la seguente stringa nel campo del name: <script>alert('XSS') </script> 3. Il resto dell'autenticazione mette dati validi 4. L'utente accede 		
Risultati attesi:	Una volta entrato, se va alla pagina del profilo dovrebbe leggere come testo lo script inserito, simile a questo: Itscriptgtalert(#039XSS#039) lt/scriptgt		

Test Case:	TC-015	Nome:	Sicurezza sistema – SQL Injection
Riferimento:	REQ-03		
Descrizione:	Il test consiste nel verificare che non accetti SQL injection.		
Prerequisiti:	Sistema pronto, con le funzioni pronte.		
Procedura:	<ol style="list-style-type: none"> 1. Vai alla pagina di registrazione. 2. Inserisci nel campo name: admin' OR '1'='1 3. Compila email e password con valori validi. 4. Invia I dati 		
Risultati attesi:	I nome deve essere salvato o mostrato in output in forma "sanificata" (es. admin#039 OR #0391#039#0391).		

Test Case:	TC-016	Nome:	Sicurezza sistema, in note – Cross-Site Scripting (XSS)
Riferimento:	REQ-04		
Descrizione:	Il test consiste nell' inserire nel campo del titolo della nota uno script un codice, che genera un alert se vulnerabile.		
Prerequisiti:	Sistema pronto, pagina di note, con anche pagina di creazione delle note, con le funzioni pronte. Utente loggato, sistema attivo e funzionante		
Procedura:	<ol style="list-style-type: none"> 1. L'utente va sulla pagina delle note 2. L'utente inserisce la seguente stringa nel campo del titolo: <script>alert('XSS') </script> 3. L'utente salva il titolo 		
Risultati attesi:	L>alert non viene eseguito, viene visualizzato come: ltscriptgtalert#039XSS#039lt/scriptgt		

Test Case:	TC-017	Nome:	Verifica del comportamento responsive del sito
Riferimento:	REQ-06		
Descrizione:	Il sito deve adattarsi correttamente a diversi dispositivi (desktop, tablet, smartphone).		
Prerequisiti:	Sistema disponibile su server testabile con browser responsive.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il sito su un computer desktop. 2. Ridurre progressivamente la finestra del browser. 3. Aprire il sito su uno smartphone o emulare con strumenti developer (es. Chrome DevTools). 4. Verificare che: I menu si adattino; Le interfacce di registrazione, login e note siano leggibili; I bottoni rimangano accessibili 		
Risultati attesi:	Il layout si adatta senza elementi tagliati, sovrapposti o non cliccabili. Tutti i contenuti restano accessibili.		

Test Case:	TC-018	Nome:	Verifica della crittografia dei dati nel database
Riferimento:	REQ-07		
Descrizione:	I dati devono essere cifrati con AES-256-CBC nel database.		
Prerequisiti:	Accesso al database e conoscenza delle funzioni di crittografia implementate.		
Procedura:	<ol style="list-style-type: none"> 1. Creare una nota o registrare un utente. 2. Accedere al database e visualizzare i campi note, title, email o altri dati sensibili. 3. Verificare che i dati siano cifrati (es. stringa Base64 incomprensibile). 4. Verificare che la funzione decrypt() restituisca i dati leggibili solo tramite codice autorizzato. 		
Risultati attesi:	Il layout si adatta senza elementi tagliati, sovrapposti o non cliccabili. Tutti i contenuti restano accessibili.		

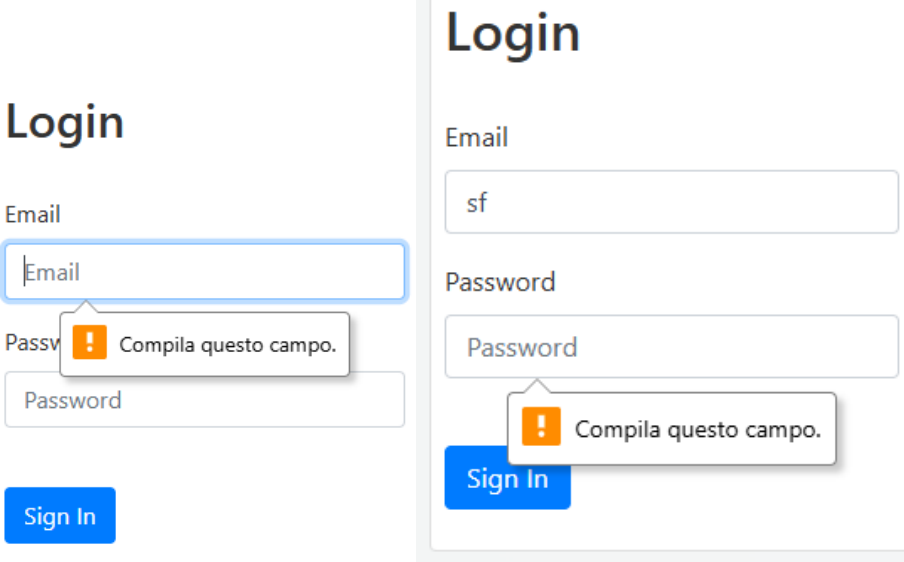
Test Case:	TC-019	Nome:	Aggiornamento, creazione e eliminazione degli attachments
Riferimento:	REQ-04		
Descrizione:	L'applicazione deve poter aggiungere e eliminare attachments. Gli attachments devono poter essere eliminati e salvati, sia al momento della creazione che dopo la creazione		
Prerequisiti:	Deve essere presente la pagina per la creazione delle note. Utente deve essere autenticato.		
Procedura:	<ol style="list-style-type: none"> 1. Inserire il titolo alla nota per poter aggiungere gli attachments 2. Aggiungere un campo di testo, un allegato e fare un disegno 3. Salvare il testo, salvare l'allegato e il disegno eliminarlo quando sia sta ancora disegnando 4. Aggiornare con il button "Aggiorna" 5. Eliminare l'allegato 6. Aggiornare con il button "Aggiorna" 		
Risultati attesi:	Gli attachments devono poter essere eliminati e salvati, sia al momento della creazione che dopo la creazione senza errori.		





Test Case:	TC-020	Nome:	Aggiornamento, creazione e eliminazione degli attachments Modifica Nome Utente
Riferimento:	REQ-04		
Descrizione:	L'utente può aggiornare il proprio nome tramite l'apposito campo nel profilo.		
Prerequisiti:	Utente loggato, con un nome già salvato nel profilo.		
Procedura:	<ol style="list-style-type: none"> 1. Andare alla sezione "Profilo". 2. Inserire un nuovo nome nel campo "Nuovo Nome"; Esempio: Giovanni 3. Cliccare su "Cambia Nome". 		
Risultati attesi:	Il nome viene aggiornato nel database e nella pagina profilo.		




Test Case:	TC-021	Nome:	Modifica Password
Riferimento:	REQ-04		
Descrizione:	L'utente può aggiornare la propria password nel profilo.		
Prerequisiti:	Utente loggato, password attuale conosciuta.		
Procedura:	<ol style="list-style-type: none"> 1. Andare alla sezione "Profilo". 2. Inserire: La password attuale corretta; Una nuova password valida, es. P@ssw0rd99 3. Cliccare su "Aggiorna Password". 		
Risultati attesi:	La password viene aggiornato nel database, cifrandola.		

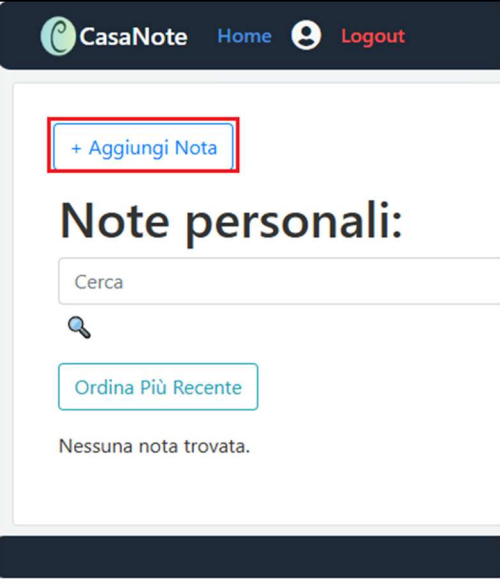
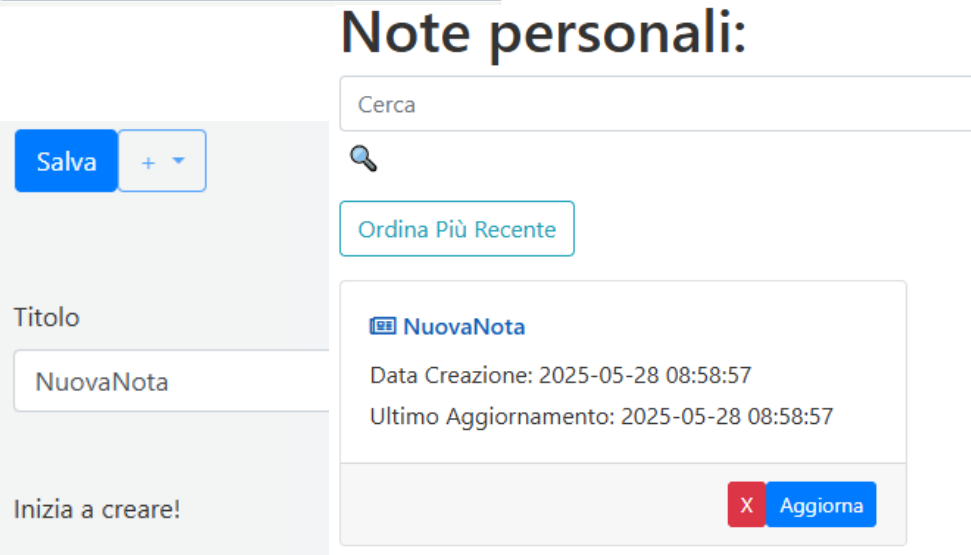
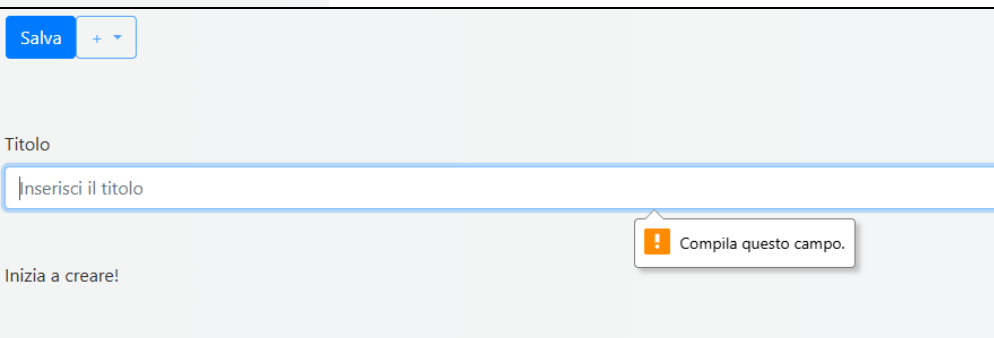
Test Case:	TC-022	Nome:	Eliminazione account utente
Riferimento:	REQ-05		
Descrizione:	L'utente deve poter eliminare il proprio account in modo sicuro e irreversibile.		
Prerequisiti:	Utente autenticato, pagina profilo presente e funzionante.		
Procedura:	1. Andare alla sezione "Profilo". 2. Eliminare l'utente cliccando su "Elimina Account" 3. Provare a fare il login		
Risultati attesi:	L'utente viene eliminato dal database.		

5.2 Risultati test

Test Case	Risultato ottenuto	Stato
TC-001		Passato
TC-002	<p>Errore</p> <ul style="list-style-type: none"> Errore nel login, email o password, errati o non esistenti 	Passato

TC-003	<h2>Login</h2> <p>Email</p> <input type="text" value="Utente@zn.ch"/> <p>Password</p> <input type="password" value="....."/> <p>Sign In</p> <div>  CasaNote Home  Logout </div> <p>+ Aggiungi Nota</p> <h3>Note personali:</h3> <div> <input type="text" value="Cerca"/>  </div> <div> Ordina Più Recente Mostra tutte le note </div> <p>Nessuna nota trovata.</p> <p>© CasaNote 2025</p>	Passato
TC-004	<h2>Register</h2> <p>Name</p> <input type="text" value="Name"/> <p>Email</p> <div>  Compila questo campo. </div> <input type="text" value="Email"/> <p>Password ⓘ</p> <input type="password" value="Password"/> <p>Confirm Password</p> <input type="password" value="Password"/> <p>Sign Up</p>	Passato

TC-005	<div><h2>Register</h2><p>Name</p><input type="text" value="dondon"/><p>Email</p><input type="text" value="Utente@zn.ch"/><p>Password ⓘ</p><input type="password" value="....."/><p>Confirm Password</p><input type="password" value="....."/><p>Sign Up</p><div> CasaNote Home  Logout</div><p>+ Aggiungi Nota</p><h3>Note personali:</h3><input type="text" value="Cerca"/><div></div><div>Ordina Più RecenteMostra tutte le note</div><p>Nessuna nota trovata.</p><div>© CasaNote 2025</div></div>	Passato
--------	---	---------

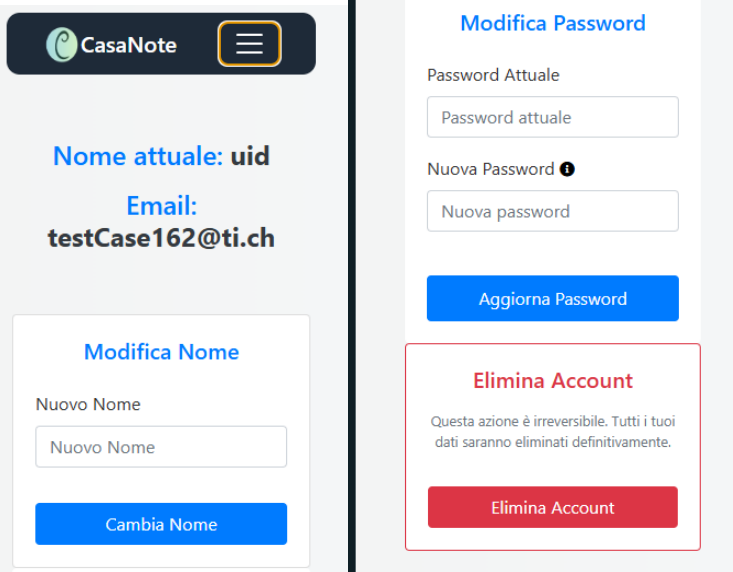
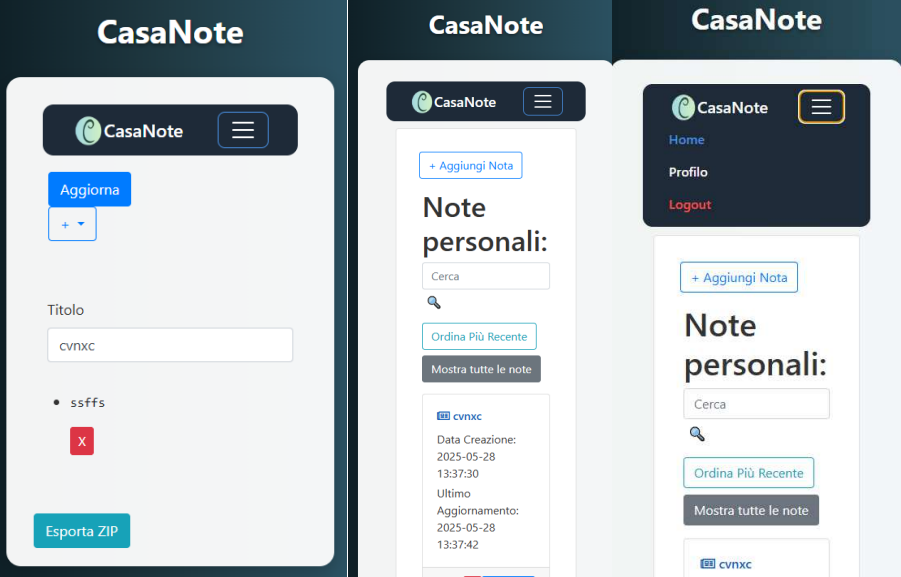
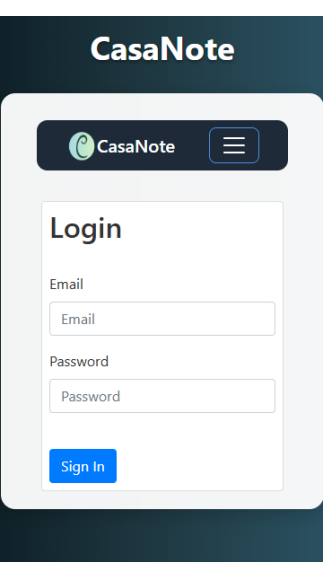
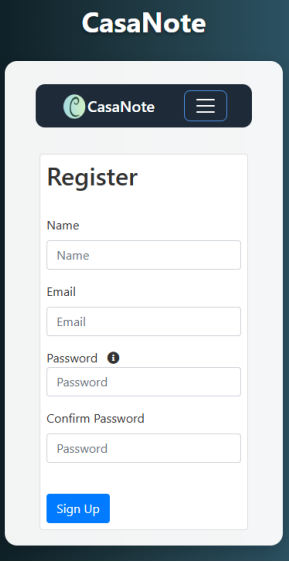
TC-006	 	Passato
TC-007		Passato

TC-008	<div> <div> Aggiorna + ▼ </div> <div> Titolo <div>NuovaNota</div> </div> <div>Inizia a creare!</div> </div> <div> <div> Aggiorna + ▼ </div> <div> Titolo <div>NotaAggiornata</div> </div> <div>Inizia a creare!</div> </div> <div> Ordina Meno Recente </div> <div> <div>📄 notaAggiornata</div> <div> Data Creazione: 2025-05-28 09:31:53 Ultimo Aggiornamento: 2025-05-28 10:14:02 </div> <div> <div>X</div> Aggiorna </div> </div>	Passato
TC-009	<div> localhost:8080 dice Sicuro di eliminare la nota? <div> <div>OK</div> <div>Annulla</div> </div> </div>	Passato

TC-010	<div><div>Note personali:</div><div><div>Cerca</div><div></div></div><div><div>Ordina Meno Recente</div><div>Mostra tutte le note</div></div><div><div><div>nota</div><div>Data Creazione: 2025-05-28 10:42:49</div><div>Ultimo Aggiornamento: 2025-05-28 11:05:41</div><div>X Aggiorna</div></div><div><div>jfdjfdjre</div><div>Data Creazione: 2025-05-28 10:43:01</div><div>Ultimo Aggiornamento: 2025-05-28 10:56:53</div><div>X Aggiorna</div></div><div><div>f</div><div>Data Creazione: 2025-05-28 10:55:57</div><div>Ultimo Aggiornamento: 2025-05-28 10:56:14</div><div>X Aggiorna</div></div></div><div><div>+ Aggiungi Nota</div></div><div><div>Note personali:</div><div><div>no</div><div></div></div><div><div>Ordina Meno Recente</div></div><div><div><div>nota</div><div>Data Creazione: 2025-05-28 10:42:49</div><div>Ultimo Aggiornamento: 2025-05-28 11:05:41</div><div>X Aggiorna</div></div></div></div></div>	Passato
TC-011	Quando si fa una ricerca senza titolo le mostra in ordine del filtro, o più recente o meno recente, a dipendenza di quello selezionato.	Passato
TC-012	<div><div><div>no</div><div></div></div><div><div>Ordina Meno Recente</div><div>Mostra tutte le note</div></div><div><div><div>nota</div><div>Data Creazione: 2025-05-28 10:42:49</div><div>Ultimo Aggiornamento: 2025-05-28 11:05:41</div><div>X Aggiorna</div></div></div><div><div>+ Aggiungi Nota</div></div><div><div>Note personali:</div><div><div>Cerca</div><div></div></div><div><div>Ordina Meno Recente</div><div>Mostra tutte le note</div></div><div><div><div>nota</div><div>Data Creazione: 2025-05-28 11:22:51</div><div>Ultimo Aggiornamento: 2025-05-28 11:22:57</div><div>X Aggiorna</div></div><div><div>sdgs</div><div>Data Creazione: 2025-05-28 11:21:29</div><div>Ultimo Aggiornamento: 2025-05-28 11:21:33</div><div>X Aggiorna</div></div><div><div>dsgsf</div><div>Data Creazione: 2025-05-28 11:21:18</div><div>Ultimo Aggiornamento: 2025-05-28 11:21:18</div><div>X Aggiorna</div></div></div></div></div>	Passato
TC-013	<div><div><div>+ Aggiungi Nota</div></div><div><div>Note personali:</div><div><div>Cerca</div><div></div></div><div><div>Ordina Meno Recente</div><div>Mostra tutte le note</div></div><div><div><div>nota</div><div>Data Creazione: 2025-05-28 11:22:51</div><div>Ultimo Aggiornamento: 2025-05-28 11:22:57</div><div>X Aggiorna</div></div><div><div>sdgs</div><div>Data Creazione: 2025-05-28 11:21:29</div><div>Ultimo Aggiornamento: 2025-05-28 11:21:33</div><div>X Aggiorna</div></div><div><div>dsgsf</div><div>Data Creazione: 2025-05-28 11:21:18</div><div>Ultimo Aggiornamento: 2025-05-28 11:21:18</div><div>X Aggiorna</div></div></div></div></div>	Passato

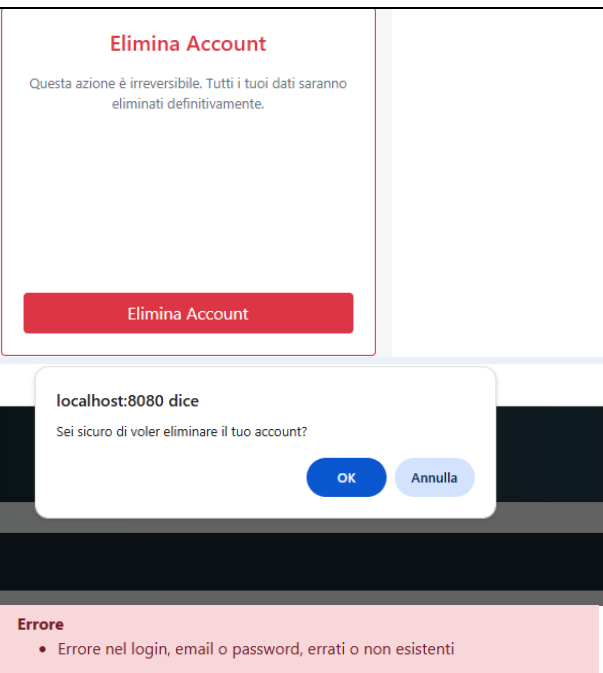
TC-014	<h2>Register</h2> <p>Name</p> <input type="text" value="<script>alert('XSS')</script>"/> <p>Email</p> <input type="text" value="provaXSS@ti.ch"/> <p>Password ⓘ</p> <input type="password" value="....."/> <p>Confirm Password</p> <input type="password" value="....."/> <p>Sign Up</p> <div> <p>Nome attuale:</p> <p>Itscriptgtalert(#039XSS#039)</p> <p>It/scriptgt</p> </div> <p>Browser sta trattando lo script come testo.</p>	Passato
TC-015	<h2>Register</h2> <p>Name</p> <input type="text" value="admin' OR '1'='1"/> <p>Email</p> <input type="text" value="testCase@ti.ch"/> <p>Password ⓘ</p> <input type="password" value="....."/> <p>Confirm Password</p> <input type="password" value="....."/> <p>Sign Up</p> <div> <p>Nome attuale:</p> <p>admin#039 OR</p> <p>#0391#039#0391</p> </div>	Passato

TC-016	<div> <div>Aggiorna</div> <div>+ ▼</div> <div>Titolo</div> <div><script>alert('XSS') </script></div> <div>Aggiorna</div> <div>+ ▼</div> <div>Titolo</div> <div>ltscriptgtalert#039XSS#039 lt/scriptgt</div> </div>	Passato
--------	--	---------

TC-017	   	Passato
--------	--	---------


TC-018	<pre>id nome_file percorso_file mime_type +-----+-----+-----+-----+ note_id +-----+-----+-----+-----+ 4 v0ZvZ890uUSm19oCt4yhFdhSUWkzcmZGSG1Gvk1SUL1JCYNJUVM xqG/3rBZ8dUG6Ci2Ife3S3pXYVXZWZwaGZBYndaVEhoU0J hdjZydkUck5hM2Uwc1VXaUp6RFl1iTuXG0Ettcn1mUFk4wXN6WEJyaFZtVVRkZWwxMU9Rdi9kd2tId1VsSf1XMXJ3PT0= image/png 6 oPXPCDVFN78nn+aGa8hxcjhUVEwzenV6Y2Rfa2RUR01iTEpDaE rq6cHy1e2sem8wqHY487BVBLe1VDMi9jenRteEQwVh0cGN Q7J0bX1t0VV5dWd0ZhdNVEhPY1VxSGs9 application</pre>	Passato
TC-019	<div><div>Titolo</div><div>titoloProva</div><div><div>ciao</div><div><div>X</div></div><div>Percorso: uploads/note_61/Documentazione_Sprint2.docx</div><div><div>X</div></div></div><div><div></div><div>disegna</div></div><div><div>Salva Disegno</div><div>Clear Drawing</div></div><div><div>X</div></div><div><div>Centro Professionale</div><div>localhost:8080 dice</div><div>Sei sicuro di procedere?</div><div><div>OK</div><div>Annulla</div></div></div><div><div>CasaNote</div><div>Aggiorna</div><div><div>+ ▼</div></div><div>Titolo</div><div>titoloProva</div><div><div>ciao</div><div><div>X</div></div><div>Percorso: uploads/note_61/Documentazione_Sprint2.docx</div><div><div>X</div></div></div></div><div><div>Aggiorna</div><div><div>+ ▼</div></div><div>Titolo</div><div>titoloProva</div><div><div>ciao</div><div><div>X</div></div></div></div></div>	Passato

TC-020	<div> <div> Nome attuale: sdf Email: provaHD@sa.com </div> <div> <div> <h3>Modifica Nome</h3> <p>Nuovo Nome</p> <input type="text"/> <p>Cambia Nome</p> </div> <div> <h3>Modifica Password</h3> <p>Password Attuale</p> <input type="password"/> <p>Nuova Password ⓘ</p> <input type="password"/> <p>Aggiorna Password</p> </div> <div> <h3>Elimina Account</h3> <p>Questa azione è irreversibile. Tutti i tuoi dati saranno eliminati definitivamente.</p> <p>Elimina Account</p> </div> </div> <div> Nome attuale: Giovanni <div> <h3>Modifica Nome</h3> <p>Nuovo Nome</p> <input type="text"/> <p>Cambia Nome</p> </div> <div> <h3>Modifica Password</h3> <p>Password Attuale</p> <input type="password"/> <p>Nuova Password ⓘ</p> <input type="password"/> <p>Aggiorna Password</p> </div> </div> </div>	Passato
TC-021	<div> <h3>Modifica Password</h3> <p>Password Attuale</p> <input type="password"/> <p>Nuova Password ⓘ</p> <input type="password"/> <p>Aggiorna Password</p> </div> <p>Ti rimanda alla pagina delle note</p>	Passato

TC-022		Passato
--------	---	---------

5.3 Mancanze/limitazioni conosciute

Era progettato una registrazione audio, ma dato che il progetto era da fare in 3 inizialmente, abbiamo ottimizzato le attività e la registrazione dell'audio per far meglio gli altri lavori l'abbiamo eliminata. Inoltre anche la metodologia ORM non è stata usata poiché abbiamo affrontato il tema verso maggio, cambiare il metodo usato fino ad ora non era conveniente.

	SAMT – Sezione Informatica	Pagina 52 di 57
	CasaNote	

6 Consuntivo

Nel Gantt consuntivo abbiamo impiegato più tempo del previsto, soprattutto a causa di alcune difficoltà incontrate durante la fase di implementazione. Un fattore importante è stata l'assenza di un membro del gruppo, poiché il progetto era inizialmente pensato per tre persone. Inoltre, abbiamo dovuto affrontare vari problemi tecnici che hanno richiesto tempo per essere risolti. Nonostante questi ostacoli, il tempo totale impiegato è stato in gran parte rispettato.

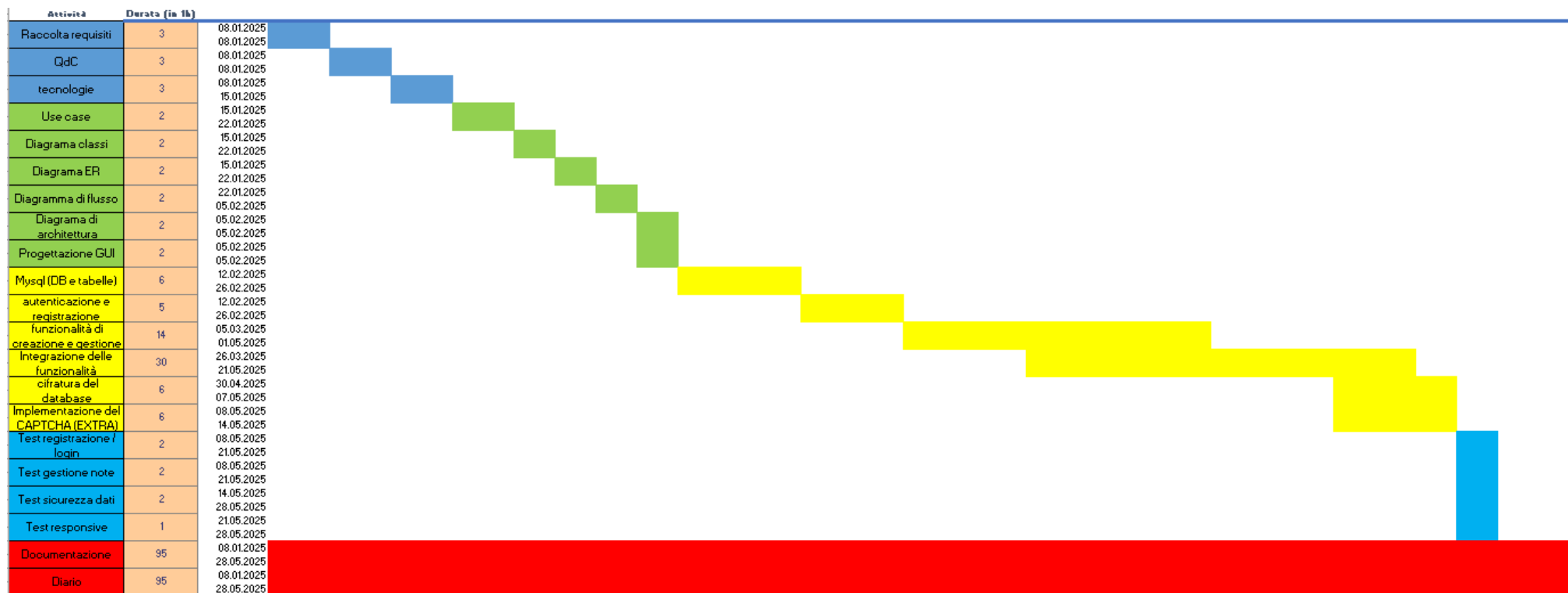


Figura 13 Gantt consuntivo

7 Conclusioni

Il progetto è flessibile e consente di creare un'applicazione web funzionale, semplice e intuitiva per la gestione di appunti con diverse applicazioni (test, file, design).

La funzionalità soddisfano tutti i requisiti iniziali, con un sistema completo che consente di creare, modificare, eliminare e associare note con gli allegati correlati. Include anche misure di sicurezza (controlli di autenticazione, convalida dell'input), nonché controllo degli errori.

Nonostante alcuni contenuti come la registrazione audio non siano stati implementati per motivi organizzativi e di priorità, il progetto può essere considerato un successo. L'obiettivo principale era creare un'applicazione concreta e stabile, che potesse essere utilizzata in situazioni reali o facilmente sviluppata per funzionalità future.

Da un punto di vista tecnico, il lavoro mi ha permesso di mettere in pratica concetti importanti dello sviluppo software, dal caricamento e modifica delle immagini in Base64, all'utilizzo di canvas HTML e all'integrazione client-server tramite elaborazione AJAX.

7.1 Sviluppi futuri

Sebbene il progetto sia completo nelle sue caratteristiche principali, ci sono diverse possibilità di miglioramento:

Supporto touch per canvas, applicabile a dispositivi mobili come tablet e smartphone.

Salva automaticamente le modifiche per evitare la perdita accidentale di dati.

Registrazione audio, una funzionalità inizialmente pianificata ma mai implementata a causa di limiti di tempo.

Editor di testo avanzato con formattazione (grassetto, elenchi, ecc.).

Esegui il backup e la sincronizzazione con i servizi cloud per una maggiore sicurezza dei dati.

Supporto multilingue per rendere l'app accessibile a un pubblico più vasto.

7.2 Considerazioni personali

Da questo progetto abbiamo imparato a lavorare seguendo una progettazione agile anziché waterfall, grazie al Trello abbiamo potuto imparare definire i vari task, con i vari requisiti e descrizioni e con la definition of don, usando la metodologia stormic. Oltre al modello agile abbiamo anche imparato a lavorare di gruppo e a comunicare, in questo modo abbiamo potuto organizzare al meglio le nostre task e a chiarirci sullo sviluppo di esse, andando ad alleggerire il carico di lavoro, perché ognuno sapeva quello che doveva eseguire.

Abbiamo imparato a gestire autonomamente il progetto da inizio alla fine organizzando tempistiche, risorse e carico di lavoro.

Inoltre, abbiamo sviluppato una maggiore flessibilità nel rispondere ai cambiamenti e alle esigenze/problemi o imprevisti emersi, che ne sono emersi parecchi, durante il lavoro, migliorando così la qualità finale del prodotto e anche la qualità della nostra organizzazione (più flessibile e pronta a rispondere ad ogni imprevisto). Questa esperienza ci ha anche insegnato l'importanza del feedback continuo all'interno del team per mantenere alta la motivazione e l'efficienza. Infine, abbiamo consolidato competenze tecniche fondamentali per futuri progetti in PHP e MVC.

8 Glossario

Termine	Definizione
AES-256-CBC	Algoritmo di crittografia simmetrica utilizzato per proteggere i dati sensibili nel database.
Allegato	File (immagine, audio, documento, ecc.) associato a una nota digitale.
AJAX	Tecnica per inviare/ricevere dati dal server in modo asincrono, migliorando la reattività dell'interfaccia.
AttachmentMapper	Classe responsabile della gestione e cifratura degli allegati salvati nel database.
authModel	Modulo che gestisce login e registrazione degli utenti, con verifica delle credenziali.
Base64	Metodo di codifica per rappresentare contenuti binari (es. immagini) in formato testuale.
Bootstrap	Framework CSS che consente la creazione rapida di interfacce responsive e moderne.
Canvas	Elemento HTML usato per il disegno a mano libera direttamente nella nota.
Controller	Componente MVC che gestisce le richieste dell'utente e coordina le operazioni tra vista e modello.
CRUD	Acronimo per le operazioni base: Create, Read, Update, Delete.
Captcha V3	Sistema di protezione contro i bot, integrato nella registrazione per garantire che l'utente sia umano.
DAO (Data Access Object)	Pattern che astrae l'accesso al database, migliorando modularità e manutenzione.
Encryption	Processo di cifratura dei dati per proteggerli da accessi non autorizzati.
Esportazione ZIP	Funzionalità che consente di esportare tutte le note e allegati in un file compresso ZIP.
Formattazione testo	Funzione che permette di modificare l'aspetto del testo (grassetto, corsivo, elenchi, ecc.).
Frontend (FE)	Parte visibile del sito con cui l'utente interagisce (HTML, CSS, JS).
Gestione note	Insieme di funzionalità relative alla creazione, modifica, eliminazione, ricerca ed esportazione delle note.
HTML	Linguaggio di markup per la struttura delle pagine web.
Interfaccia responsiva	Interfaccia adattabile a diversi dispositivi (PC, tablet, smartphone).
IV (Initialization Vector)	Valore random usato insieme alla chiave per aumentare la sicurezza della cifratura.
JavaScript	Linguaggio client-side che consente di implementare dinamiche nell'interfaccia utente.
Logger	Componente software che registra eventi, errori e operazioni di sistema.
Manage (Controller)	Controller principale per la gestione delle note, usato nelle azioni dell'utente autenticato.
Model	Componente MVC che rappresenta la logica e i dati dell'applicazione.
MVC	Architettura software composta da Model, View e Controller.

NoteMapper	Classe per gestire operazioni sul database riguardanti le note (creazione, modifica, cancellazione).
PDF/TXT Export	Funzionalità che permette di esportare una singola nota in formato PDF o TXT.
PHP	Linguaggio di programmazione server-side usato per la logica del backend.
Prepared Statement	Query SQL parametrizzate che proteggono da attacchi SQL Injection.
RAM/CPU/GPU	Componenti hardware del PC usato per lo sviluppo (32GB RAM, Intel i7, NVIDIA T400).
Repository	Parte del backend che si interfaccia direttamente con il database.
Routing	Sistema che definisce quale controller o funzione rispondere a un determinato URL.
Salvataggio automatico	Meccanismo che verifica se la nota è salvata prima di uscire o chiudere la pagina.
Sanitizzazione	Pulizia dei dati immessi per evitare rischi di sicurezza (es. XSS).
Sessione	Meccanismo che mantiene l'utente connesso e identifica le sue azioni durante la navigazione.
Swimlane	Diagramma che mostra il flusso delle attività tra attori e componenti (es. utente, FE, BE).
TextArea	Campo di testo multilinea usato per scrivere il contenuto delle note.
User-Friendly	Caratteristica dell'interfaccia pensata per essere semplice e intuitiva da usare.
Validator	Classe che verifica e filtra i dati immessi dall'utente.
View	Componente MVC che si occupa della parte visiva, cioè ciò che l'utente vede.
Windows 10 Education	Sistema operativo del PC utilizzato per sviluppare il progetto.
ZipArchive	Classe PHP che consente di creare file ZIP per l'esportazione delle note.

9 Bibliografia

9.1 Sitografia

<https://www.php.net/manual/en/>, PHP Manual.

<https://getbootstrap.com/docs/5.3/getting-started/introduction/>, Bootstrap 5.3 Documentation.

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API, MDN Web Docs – Canvas API, 18-03-2025.

<https://www.w3schools.com/php/>, W3School da 20-02-2025 a 28-05-2025.

https://www.w3schools.com/js/js_ajax_intro.asp, W3Schools – AJAX Introduction, 21-03-2025.

<https://stackoverflow.com/>, Stack Overflow – Q&A for Programmers, Consultazioni varie tra 10-03-2025 e 25-04-2025.

<https://jquery.com/>, jQuery – Official Documentation, 26-03-2025.

10 Indice delle figure

Figura 1 - Use Case.....	8
Figura 2 - Gantt.....	10
Figura 3 - Diagramma Architettura	12
Figura 4 - Swimlanes registrazione	13
Figura 5 - Swimlanes login	14
Figura 6 - Swimlanes creazione/salvataggio nota	15
Figura 7 - Diagramma Classi	16
Figura 8 - Design registrazione utente	17
Figura 9 - Design login utente	18
Figura 10 - Design home	19
Figura 11 - Design creazione nota	20
Figura 12 - Design visualizzazione note.....	21
Figura 13 Gantt consuntivo	53

11 Allegati

- Use Case
- Requisiti
- UI
- Gantt consuntivo e preventivo
- Diagramma delle Classi
- Diagramma di Architettura
- Diagramma di Flusso
- Diagramma ER
- Logo
- Tecnologie
- Documentazione deploy
- Documentazione di tutti gli Sprint