Progetto “Note Condivise”

Corso di Ingegneria del Software
Anno Accademico 2024-2025
Corso di Laurea in Informatica per il Management

Docente del corso – Prof. Davide Rossi
Tutor del corso – Dr. Michele Dinelli

Autori
Maurizio Amadori – Matricola: 0001078717
Francesco Maria Fuligni – Matricola: 0001068987
Lorenzo Magrini – Matricola: 0001070628
Roberto Zanolli – Matricola: 0001070505

Modello di Processo

Ibrido strutturato-agile suggerito dalle specifiche del progetto

Il modello di processo adottato segue un ibrido strutturato-agile, composto da due fasi:

1. **Fase di *inception***
   Riguarda la produzione dei seguenti artefatti:
   - Modello dei casi d'uso
   - Modello di dominio
   - Glossario

2. **Fase di *construction***
   Riguarda la costruzione del sistema software, seguendo le pratiche di gestione di processo di **Scrum**.
   Sono stati realizzati 4 sprint. Per ogni sprint è stato definito un Product Owner, uno Scrum Master e due Developers. Ruotando i ruoli ad ogni iterazione, ogni membro del gruppo ha svolto tutti i ruoli (1 volta product owner, 1 volta scrum master, 2 volte developer).
   Per ogni sprint sono stati rispettati gli eventi Scrum:
   - Sprint Planning: il primo giorno di ogni sprint, tra tutti i componenti del gruppo
   - Daily Scrum: ogni giorno, tra i developers
   - Sprint Review e Sprint Retrospective: l'ultimo giorno di ogni sprint, tra tutti i componenti del gruppo

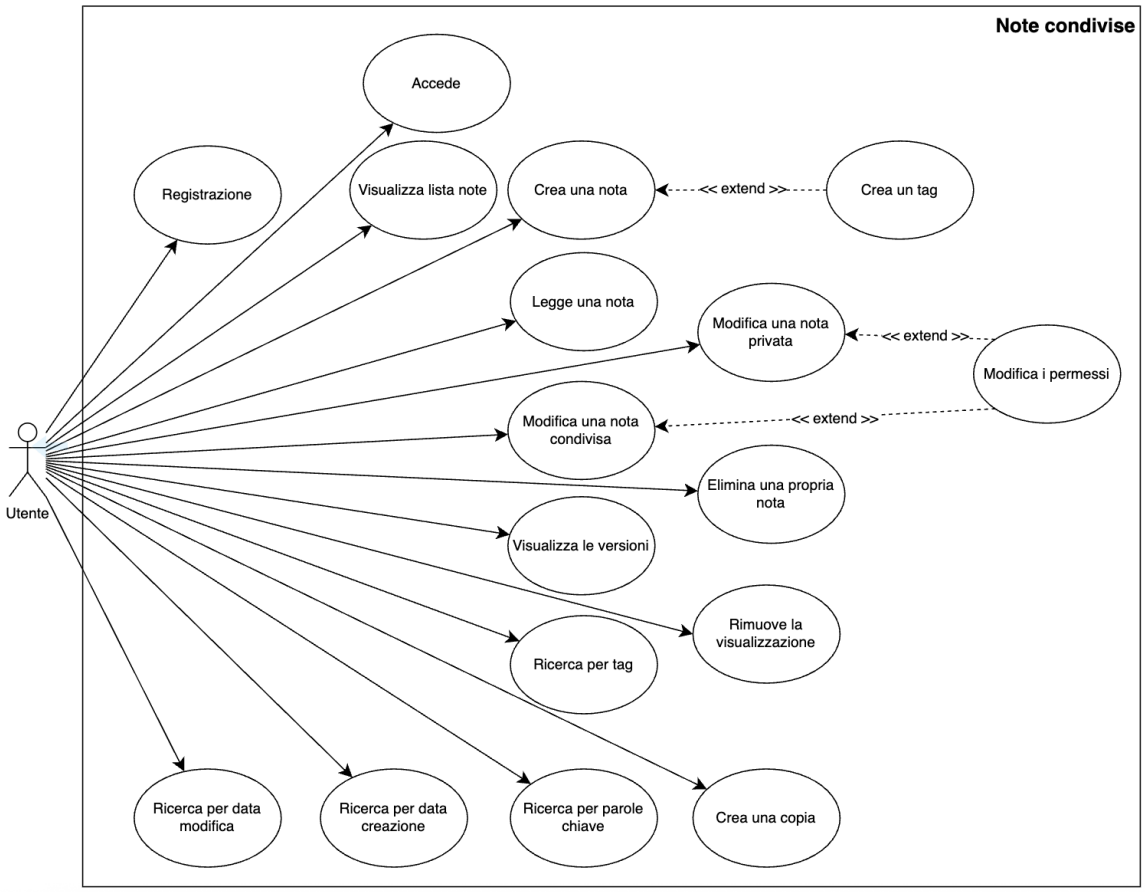
   Sono stati prodotti gli artefatti Scrum:
   - Product Backlog
   - Sprint Backlog
   - Burn Down Chart

   Per ogni sprint, le task sono state gestite utilizzando una Kanban Board.

Modello dei Casi d’Uso

Diagramma e descrizione testuale

Diagramma dei casi d’uso



Sistema:

- Note Condivise

Attori:

- Utente

Obiettivi Utente:

- Registrazione
- Accesso
- Visualizzazione della lista delle note
- Creazione di una nota
- Lettura di una nota
- Modifica di una nota privata

- Modifica di una nota condivisa
- Elimina una propria nota
- Modifica dei permessi di una nota
- Creazione di tag
- Visualizzazione delle versioni di una nota
- Creazione di una copia di una nota
- Ricerca per parole chiave
- Ricerca per tag
- Rimozione della visualizzazione

Descrizione testuale dei casi d'uso

Registrazione
- Id: UC1
- Attori: Utente
- Preconditions: l'utente non è registrato
- Main sequence: l'utente seleziona l'opzione per registrarsi, l'utente compila i campi richiesti, l'utente invia la registrazione
- Postconditions: l'utente è registrato

Accede
- Id: UC2
- Attori: Utente
- Preconditions: l'utente è registrato, l'utente non è autenticato
- Main sequence: l'utente seleziona l'opzione per autenticarsi, l'utente inserisce le proprie credenziali, l'utente invia la richiesta di login
- Postconditions: l'utente è autenticato

Visualizza lista note
- Id: UC3
- Attori: Utente
- Preconditions: l'utente ha eseguito l'accesso
- Main sequence: l'utente visualizza la lista delle proprie note private e delle note condivise con lui

Precondizione comune per tutti i casi d'uso seguenti è che l'utente abbia effettuato l'accesso.

Crea una nota
- Id: UC4
- Attori: Utente
- Main sequence: l'utente seleziona l'opzione per creare una nota, l'utente compila i campi richiesti, l'utente crea la nota
- Postcondizione: la nuova nota è creata

Legge una nota

- Id: UC5
- Attori: Utente
- Preconditions: la lista delle note condivise è mostrata
- Main sequence: l'utente seleziona una nota
- Postconditions: il dettaglio della nota è mostrato

Modifica una nota privata
- Id: UC6
- Attori: Utente
- Preconditions: l'utente visualizza una nota privata di cui è creatore
- Main sequence: l'utente modifica il contenuto della nota, l'utente invia le modifiche
- Postcondition: la nota modificata è salvata

Modifica una nota condivisa
- Id: UC7
- Attori: Utente
- Preconditions: l'utente visualizza una nota su cui ha permesso di modifica
- Main sequence: l'utente modifica il contenuto della nota, l'utente invia le modifiche
- Alternative sequence: il sistema segnala la presenza di conflitti, l'utente sceglie quale contenuto della nota mantenere, l'utente invia le modifiche
- Alternative sequence: il sistema mostra un messaggio di errore (nota eliminata, o i permessi sono stati revocati)
- Postcondition: la nota modificata è salvata *oppure* la modifica viene scartata

Elimina una propria nota
- Id: UC8
- Attori: Utente
- Preconditions: l'utente visualizza una nota di cui è creatore
- Main sequence: l'utente seleziona l'opzione per eliminare la nota
- Postconditions: la nota è eliminata

Modifica i permessi
- Id: UC9
- Attori: Utente
- Preconditions: l'utente è il creatore della nota selezionata
- Main sequence: l'utente modifica il campo relativo al permesso di condivisione
- Postconditions: il campo dei permessi di condivisione è aggiornatoù
- *estende UC6 e UC7*

Rimuove la visualizzazione
- Id: UC10
- Attori: Utente
- Precondition: l'utente seleziona una nota condivisa di cui non è il creatore
- Main sequence: l'utente seleziona l'opzione per eliminarsi dalla condivisione della nota
- Postconditions: la nota non è più condivisa con l'utente

Crea un tag
- Id: UC11

- Attori: Utente
- Preconditions: l'utente sta modificando o creando una nota
- Main sequence: l'utente compila il campo richiesto, l'utente preme il pulsante per salvare il tag
- Postconditions: il nuovo tag è salvato e presente fra i disponibili
- *estende UC4*

Visualizza le versioni
- Id: UC12
- Attori: Utente
- Main sequence: l'utente seleziona una nota, l'utente seleziona l'opzione per visualizzare lo storico delle versioni di una nota
- Postconditions: le versioni della nota sono mostrate

Crea una copia
- Id: UC13
- Attori: Utente
- Main sequence: l'utente seleziona una nota, l'utente seleziona l'opzione per copiare la nota
- Postconditions: la nota copiata è salvata come una nuova nota

Ricerca per parole chiave
- Id: UC14
- Attori: Utente
- Main sequence: l'utente immette il testo da cercare, l'utente preme il tasto per effettuare la ricerca
- Postconditions: i risultati della ricerca sono mostrati

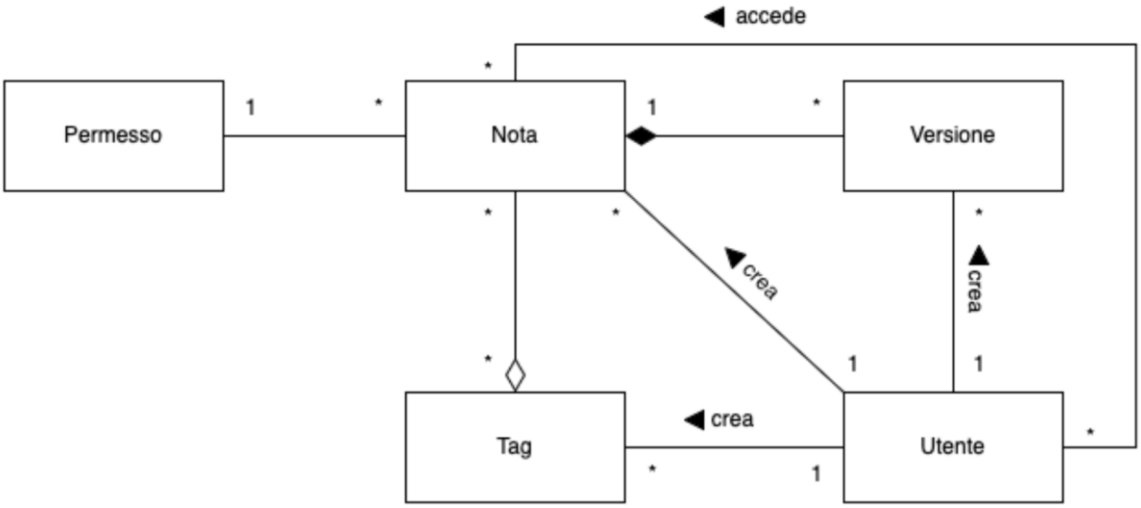
Ricerca per tag
- Id: UC15
- Attori: Utente
- Main sequence: l'utente seleziona i tag per cui vuole effettuare la ricerca, l'utente preme il tasto per effettuare la ricerca
  Postconditions: i risultati della ricerca sono mostrati

Ricerca per data modifica
- Id: UC16
- Attori: Utente
- Main sequence: l'utente seleziona la data di modifica per cui vuole effettuare la ricerca, l'utente preme il tasto per effettuare la ricerca
  Postconditions: i risultati della ricerca sono mostrati

Ricerca per data creazione
- Id: UC17
- Attori: Utente
- Main sequence: l'utente seleziona la data di creazione per cui vuole effettuare la ricerca, l'utente preme il tasto per effettuare la ricerca
  Postconditions: i risultati della ricerca sono mostrati

Modello di Dominio

Diagramma delle classi UML



Glossario

Principali termini e concetti presentati nel documento di specifica

Nota	Un’entità testuale (contenuto fino a 280 caratteri) creata da un utente, con data di creazione e data di ultima modifica; può essere privata o condivisa.
Cartella	Struttura gerarchica in cui organizzare più note, per raggrupparle in base a tipologie o progetti comuni.
Tag	Etichetta assegnabile a una o più note per favorire la classificazione e la ricerca tramite parole-chiave.
Utente proprietario	L’utente che ha creato una nota e ne detiene il controllo sui permessi di condivisione e modifica.
Permesso Privato	Configurazione di una nota visibile e modificabile solo dal proprietario.
Permesso di Condivisione in Sola Lettura	Livello di accesso che consente ad altri utenti di visualizzare la nota, ma non di modificarla.
Permesso di Condivisione in Scrittura	Livello di accesso che consente ad altri utenti di modificare la nota condivisa
Versionamento	Meccanismo per salvare ogni modifica come nuova versione della nota, consentendo tracciamento cronologico dei cambiamenti.

Conflitto di Salvataggio Asincrono	Situazione in cui un salvataggio tardivo sovrascrive modifiche intermedie effettuate da un altro utente: il sistema deve decidere se sovrascrivere, unire o notificare il conflitto.
Ricerca per Parola-chiave	Funzionalità che filtra le note in base a termini presenti nel titolo o nel contenuto.
Filtri (tag, data)	Meccanismi per restringere i risultati di ricerca sulla base di tag assegnati, autore della nota o intervallo di date di creazione/modifica.
Registrazione Utente	Processo di creazione di un account con username e password per poter accedere e gestire note.
Persistenza dei Dati	Salvaguardia permanente di note, permessi e versioni in un archivio (MapDB), rendendo le informazioni disponibili dopo riavvii.

Product Backlog

Definito sulla base dei casi d'uso e delle task da implementare

Gli elementi del backlog sono ordinati per priorità e così prelevati per l'inserimento in ogni sprint backlog. Ogni elemento, laddove necessario, è stato raffinato nei punti che lo seguono, rappresentanti le singole task aggiunte allo sprint backlog.
In ogni sprint, le task sono state gestite utilizzando una kanban board e al termine dell'iterazione il product backlog è stato rivisto coerentemente con il lavoro da svolgere nel periodo successivo.

Legenda
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Sprint 5

Registrazione

1. creazione form per la raccolta dei dati
2. raccolta e validazione dei dati dal form
3. salvataggio dati utente sul db (con crittografia password)
4. redirect verso l'interfaccia di login

Accesso

1. creazione interfaccia di login
2. acquisizione dati dal form
3. verifica della correttezza dei dati dell'utente
4. redirect verso l'interfaccia home

Creazione di una nota

1. Implementazione dei Permessi
2. creazione form per la raccolta dei dati
3. raccolta dei dati sul server
4. assegnazione tag in creazione
5. salvataggio della nota a db

Implementazione persistenza dati

1. set up della libreria per la persistenza

Lettura di una nota

1. creazione interfaccia home
2. raccolta delle note visualizzabili dall'utente dal db
3. visualizzazione dell'anteprima delle note

Modifica di una nota privata

1. creazione dell'interfaccia di modifica
2. raccolta e implementazione del salvataggio dei dati
3. assegnazione dei tag in modifica

Elimina una propria nota

1. Aggiornamento interfaccia ed eliminazione da db

Creazione di una copia di una nota

1. Aggiornamento interfaccia
2. raccolta e salvataggio dei dati della nuova nota

Ricerca per parole chiave

1. Aggiornamento interfaccia home
2. implementazione logica di ricerca

Ricerca per tag

1. Aggiornamento interfaccia home
2. implementazione logica di ricerca

Modifica dei permessi

- aggiunta dei permessi
- controlli operazioni
- gestione visualizzazione
- predisposizione gestione modifica

Aggiornamento della versione alla modifica

Visualizzazione delle versioni di una nota

GitHub Action: esecuzione test per autorizzare una pr

Makefile per lancio comandi

Refactor Nota

Gestione concorrenza in modifica

- aggiungere controllo versione all'invio della modifica
- visualizzare le diverse versioni se differenti
- permettere di scegliere quale versione tenere
- segnalare errori se cambiano i permessi o la nota viene eliminata

Aggiornamento filtri

- filtro su data creazione nota
- filtro su data modifica nota

Refactor sessione

- mantenimento dei dati di sessione con Jetty

Refactor servlet

- pattern **proxy** per la gestione delle operazioni sugli oggetti e i controlli sui permessi (nel caso di Note)

Refactor factory

- rendere factory singleton

Refactor versionamento

- implementare pattern **memento** se possibile

Refresh:

- refresh campi dettaglio nota al click su modifica

Concorrenza:

- controllo sull'indice della versione
- risoluzione di bug

Filtri:

- bottone per pulizia filtri in schermata home
- filtro per data creazione

Services:

- implementazione pattern Façade per i servlet

Test:

- implementazione test concorrenza
- test refactoring per migliorare il codice

Astrazioni:

- introduzione di interfacce per l'accesso agli oggetti

Pulizia codice:

- refactor codice per renderlo più chiaro e leggibile ove necessario
- uniformare i commenti

Controllo degli artefatti prodotti

Sprint Backlogs e Burn Down Chart

Elementi prelevati dal product backlog e raffinati per ogni iterazione

Gli sprint backlog sono mostrati anche nelle istantanee della Kanban board inserite nel diario del progetto, all'inizio ed al termine di ogni iterazione.

Sprint 1

Development Team: Maurizio Amadori, Roberto Zanolli
Scrum Master: Francesco Maria Fuligni
Product Owner: Lorenzo Magrini

Implementazione persistenza dati

1. set up della libreria per la persistenza

Registrazione

1. creazione form per la raccolta dei dati
2. raccolta e validazione dei dati dal form
3. salvataggio dati utente sul db (con crittografia password)
4. redirect verso l'interfaccia di login

Accesso

1. creazione interfaccia di login
2. acquisizione dati dal form
3. verifica della correttezza dei dati dell'utente
4. redirect verso l'interfaccia home

Creazione di una nota

1. Implementazione dei Permessi
2. creazione form per la raccolta dei dati
3. raccolta dei dati sul server
4. assegnazione tag in creazione
5. salvataggio della nota a db

Sprint 2

Development Team: Francesco Maria Fuligni, Lorenzo Magrini
Scrum Master: Maurizio Amadori
Product Owner: Roberto Zanolli

Creazione di una copia di una nota

1. Aggiornamento interfaccia
2. raccolta e salvataggio dei dati della nuova nota

Ricerca per parole chiave

1. Aggiornamento interfaccia home
2. implementazione logica di ricerca

Ricerca per tag

1. Aggiornamento interfaccia home
2. implementazione logica di ricerca

Lettura di una nota

1. creazione interfaccia home
2. raccolta delle note visualizzabili dall'utente dal db
3. visualizzazione dell'anteprima delle note

Modifica di una nota privata

1. creazione dell'interfaccia di modifica
2. raccolta e implementazione del salvataggio dei dati
3. assegnazione dei tag in modifica

Sprint 3

Development Team: Francesco Maria Fuligni, Roberto Zanolli
Scrum Master: Lorenzo Magrini
Product Owner: Maurizio Amadori

GitHub Actions:

1. esecuzione test

Modifica dei permessi

1. aggiunta dei permessi
2. controlli operazioni
3. gestione visualizzazione
4. predisposizione gestione modifica

Visualizzazione delle versioni di una nota

Modifica di una nota privata

1. aggiornamento della versione

Makefile per lancio comandi (da directory GitHub)

Refactor Note

Sprint 4

Development Team: Maurizio Amadori, Lorenzo Magrini
Scrum Master: Roberto Zanolli
Product Owner: Francesco Maria Fuligni

Gestione concorrenza in modifica

- aggiungere controllo versione all'invio della modifica
- visualizzare le diverse versioni se differenti
- permettere di scegliere quale versione tenere
- segnalare errori se cambiano i permessi o la nota viene eliminata
- risoluzione bug aggiunta tag in concorrenza

Refactor sessione

- valutare se mantenere la sessione o eliminarla
- migliorare l'implementazione della sessione se tenuta
- refactor dei test coerentemente

Filtri:

- ricerca per data di ultima modifica

Sprint 5

Development Team: Francesco Maria Fuligni, Lorenzo Magrini
Scrum Master: Roberto Zanolli
Product Owner: Maurizio Amadori

Refresh:

- refresh campi dettaglio nota al click su modifica

Concorrenza:

- controllo sull'indice della versione
- risoluzione di bug

Filtri:

- bottone per pulizia filtri in schermata home
- filtro per data creazione

Services:

- implementazione pattern Façade per i servlet

Test:

- implementazione test concorrenza
- test refactoring per migliorare il codice

Astrazioni:

- introduzione di interfacce per l'accesso agli oggetti

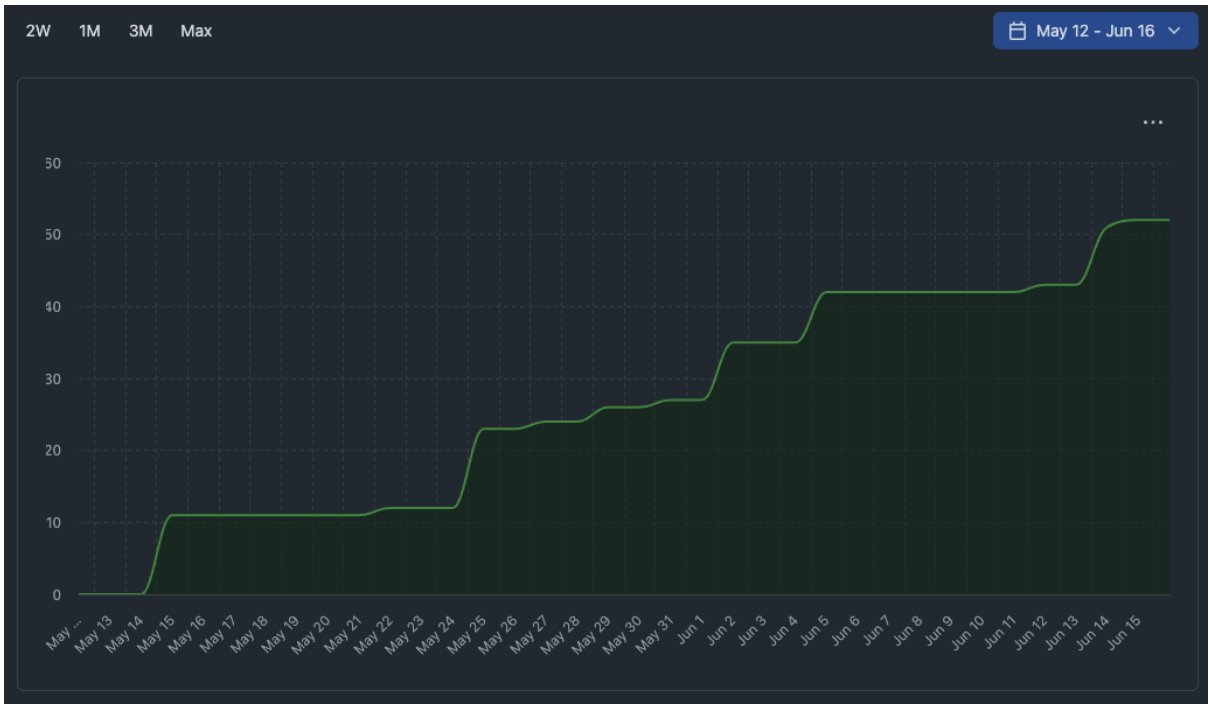
Pulizia codice:

- refactor codice per renderlo più chiaro e leggibile ove necessario
- uniformare i commenti

Controllo degli artefatti prodotti

Burn Down (Up) Chart

Il burn down chart è stato realizzato a partire dalla Kanban board utilizzata per tenere traccia delle task di ogni sprint. Tramite lo strumento è stato prodotto un burn up chart, nel quale è mostrato il numero di task completate nel corso del tempo.



Manuale Utente

Guida accessibile ad utenti inesperti per l’utilizzo del prodotto

1. Registrazione e Accesso

Quando l’utente apre l’app, dovrà effettuare:

- Registrazione inserendo una email ed una password. L’email identifica univocamente l’utente nel sistema.
- Login utilizzando le credenziali registrate nel sistema.

2. Visualizzazione delle Note - Home

Una volta effettuato il login, l’utente visualizza una lista con tutte le note di cui dispone i permessi (di sua proprietà o concesso).

3. Filtro Note - Home

Le note possono essere filtrate utilizzando:

- Data dell’ultima modifica, specificando un intervallo di date
- Data di creazione, specificando un intervallo di date
- Tag associati
- Parole chiave, contenute nel titolo o nel corpo della nota

Per utilizzare i filtri, selezionare i tag o digitare i criteri negli appositi campi.

4. Azioni Principali - Home

Dalla schermata principale è possibile:

- Effettuare il logout (cliccando su “Esci”).
- Creare una nuova nota (cliccando su “Nuova Nota”).
- Visualizzare il dettaglio di una nota esistente (cliccando sul pulsante “Visualizza Nota” della nota desiderata).

5. Visualizzazione e Modifica Nota - Dettaglio Nota

Selezionando una nota dalla lista si visualizzano:

- Titolo della nota (ultima versione)
- Contenuto della nota (ultima versione)
- Data di creazione e di ultima modifica
- Tag associati
- Bottone per rimuoversi dalla visualizzazione della nota (non la si visualizzerà più nella lista delle note). Il proprietario della nota non può eseguire questa operazione.
- Bottone per visualizzare lo storico delle versioni della nota

Se l’utente ha il permesso di scrittura, potrà modificare titolo, contenuto, tag e i permessi di condivisione (solo qualora fosse il proprietario), premendo poi “Salva” per registrare le modifiche.
Le modifiche saranno visualizzabili nello storico delle versioni, dal dettaglio della nota.

6. Gestione Conflitti - Modifica

In caso di conflitto (quando un utente tenta di modificare una nota a partire da una versione che è diversa dall’ultima memorizzata per quella nota nel sistema):

- L’applicazione mostrerà chiaramente entrambe le versioni della nota (quella modificata dall’utente e quella aggiornata presente sul database).
- L’utente potrà selezionare quale versione mantenere, modificandole, premendo sul pulsante relativo alla versione desiderata.

Dopo la selezione, il contenuto scelto della nota scelta sarà salvato come la versione corrente. Qualora nella fase di salvataggio si verificasse un nuovo conflitto, l’utente sarà nuovamente reindirizzato all’interfaccia di risoluzione del conflitto.

Manuale Sviluppatore

Istruzioni su come installare e lanciare su un nuovo computer il software sviluppato a partire dall'ottenimento del codice sorgente

Strumenti

Requisiti

- Java
- Maven
- GWT (Google Web Toolkit)
- JUnit per unit testing
- Git (versionamento)
- IDE consigliato: VSCode

Stack Tecnologico

Per la realizzazione del progetto è stato utilizzato il seguente stack tecnologico:

- **Java** per l'implementazione
- **Maven** per la gestione delle dipendenze e del processo di build
- **GWT** per la realizzazione delle pagine web
- **MapDB** per la persistenza dei dati
- **Gson** per l'interazione con JSON
- **JUnit** per la realizzazione di unit testing

Librerie utilizzate

Durante lo sviluppo sono state scelte le seguenti librerie, installate mediante maven:

- **Password4j** per l'hashing delle password al salvataggio su database
- **Mockito** per la realizzazione di mock per test
- **Gson** per gestione e parsing JSON
- **MapDB** per persistenza dei dati

Installazione del Progetto

Clonare la repository:

```
git clone https://github.com/RobertoZanolli/progetto-sweng-24-25.git
cd gwt-notes-maven
```

Compilare e avviare l'applicazione:

```
# Avviare il CodeServer per la parte client
mvn -U -e gwt:codeserver -pl notes-client -am
```

```
# In una nuova finestra, avviare il server Jetty
mvn -U jetty:run -pl notes-server -am -Denv=dev
```

Accedere all'applicazione tramite il browser all'indirizzo: **http://localhost:8080/**

Il file README.md contiene credenziali utente già registrate nel sistema, con alcune note associate, per utilizzare l'applicativo ed eseguirne una demo.

Makefile

Alla radice del progetto è presente un Makefile che semplifica le operazioni principali. I target disponibili sono:

- **help**: mostra l'elenco dei comandi disponibili (make help).
- **codeserver**: avvia il GWT CodeServer per lo sviluppo client (make codeserver).
- **run-server**: avvia Jetty sul modulo server in modalità dev (make run-server).
- **compile-server**: ricompila solo il modulo server dopo modifiche (make compile-server).
- **test**: esegue tutti i test JUnit del progetto (make test).
- **clean-test**: pulisce i file di test e ricompila i test (make clean-test).

```
make help           # mostra l'elenco dei comandi
make codeserver     # avvia il client
make run-server     # avvia il server
make compile-server # ricompila il server dopo modifiche backend
make test           # esegue tutti i test
make clean-test     # pulisce e testa di nuovo
```

Struttura del Progetto

- **notes-client:** contiene la parte frontend realizzata con GWT (interfaccia utente, pannelli, sessioni).

- **notes-server:** contiene la logica di business, servlet, gestione autenticazione e database.
- **notes-shared:** contiene le classi condivise tra client e server (entità, servizi condivisi).

Gestione della Persistenza

Il progetto utilizza MapDB per la persistenza dei dati. Ogni classe per la gestione di un database estende la classe AbstractDB, che implementa metodi di base per le operazioni sui database comuni dovute all'utilizzo di MapDB.

Le classi per accedere al database, implementate come Singleton, sono:

- **NoteDB.java**: Gestisce le operazioni CRUD per le note.
- **UserDB.java**: Gestisce le operazioni per gli utenti.
- **TagDB.java**: Gestisce le operazioni sui tag.

Servlet

Ogni servlet rappresenta una rotta per interagire con l'apposito backend. Le classi Servlet si occupano di gestire le operazioni richieste lato client e richiamare gli appositi metodi lato server.

- **LoginServlet.java:** Gestisce il login degli utenti (autenticazione e creazione della sessione).
- **RegisterServlet.java:** Gestisce la registrazione di nuovi utenti (controlli validità e salvataggio).
- **NoteServlet.java:** Gestisce le operazioni CRUD per le note (creazione, lettura, modifica e eliminazione).
- **TagServlet.java:** Gestisce operazioni sui tag associati alle note.
- **HideNoteServlet.java**: Gestisce l'operazione di "hide" di una nota.

Service

Associata ad ogni classe Servlet è presente un Service che si occupa di implementare le funzionalità richieste dal Servlet. Opera secondo il pattern Façade, rappresentando il punto di accesso ai metodi delle singole classi che implementano le funzionalità.

Testing

Per ogni servlet è stato definito un Unit Test che si occupa di testare tutte le operazioni eseguite da ogni servlet. Per ogni operazione sono state testate casistiche di successo e insuccesso che possono verificarsi durante l'utilizzo dell'applicazione.

Sessione

Lato client, le informazioni relative alla sessione sono mantenute all'interno di una classe Singleton, Session, che si occupa di memorizzare l'email dell'utente che ha effettuato l'accesso.
Lato server, i dati di sessione sono gestiti tramite HttpSession, memorizzando la mail dell'utente al login. Tramite HttpSession avviene il trasferimento di dati di sessione tra client e server ad ogni operazione richiamata.

Astrazioni sugli Oggetti

Ogni oggetto che implementa la logica di business dell'applicazione (Note, User, Tag, Version) possiede una propria interfaccia, implementata dalla rispettiva classe concreta. Gli oggetti che interagiscono con questi, richiamano i metodi dell'interfaccia.
Le Factories definite per la creazione di tali oggetti restituiscono la tipologia definita dall'interfaccia. In questo modo, le dipendenze sono rivolte verso le astrazioni.
Le Factories responsabili della creazione di ogni oggetto contengono metodi per la creazione a partire dall'inserimento di parametri o da Json. Il parsing del Json viene delegato dalla factory al parser.

Json Parsing

Lato client, i metodi di parsing definiti per creare gli oggetti sono contenuti nella classe JsonParserUtil.
Lato server, è definita un'interfaccia JsonParser che fornisce metodi per creare un'istanza dell'oggetto a partire dal suo Json e viceversa. L'interfaccia è implementata dalla classe GsonJsonParser, che sfrutta la libreria Gson per le operazioni di parisng.

Navigazione Web

L'applicazione web è sviluppato come single page app, il contenuto dei pannelli viene aggiornato in base agli eventi scatenati dall'utente.
Il main panel viene inizializzato in notes.java, tutti i vari pannelli sono disponibili nella cartella notes-client.
Ė possibile aggiungere servlet e viste nel file web.xml come nell'esempio:

```
<welcome-file-list>
       <welcome-file>notes.html</welcome-file>
   </welcome-file-list>
   <servlet>
       <servlet-name>RegisterServlet</servlet-name>
       <servlet-class>com.google.gwt.sample.notes.server.RegisterServlet</servlet-
class>
   </servlet>
   <servlet-mapping>
       <servlet-name>RegisterServlet</servlet-name>
       <url-pattern>/api/register</url-pattern>
   </servlet-mapping>
```

Aggiungere dipendenze

Per aggiungere nuove dipendenze a Maven è sufficiente recarsi nel pom.xml del modulo in cui si vuole utilizzare la dipendenza ed inserirla come in esempio:

```
       <dependency>
         <groupId>com.password4j</groupId>
         <artifactId>password4j</artifactId>
         <version>1.6.1</version>
       </dependency>
```

Git e GitHub

Per la gestione del progetto è stato utilizzato il sistema di versionamento del codice Git, sfruttando l'interfaccia fornita da GitHub.
Per l'implementazione e l'aggiornamento di ogni funzionalità o la risoluzione di un bug è stato definito un branch specifico.
Al termine di ogni modifica è stata creata una pull request sul branch development, contenente in ogni momento l'ultima versione correttamente funzionante dell'applicativo.
Ogni pull request, prima di essere confermata, è stata oggetto di review da parte di un membro del gruppo, diverso da colui che ha scritto il codice ivi contenuto.
Per eseguire la release finale del progetto è stato aggiornato il branch main allo stato di develop ed è stato eseguito il rilascio su tale branch.
È stata inserito un workflow con GitHub Actions per verificare, al momento di apertura di una pull request sul branch development, che tutti i test siano eseguiti correttamente senza generare errori. Il workflow può essere lanciato anche manualmente dalla sezione apposita.

Informazioni per manutenzione e contribuzione

Design patterns applicati

Per gestire l'assegnazione delle responsabilità e garantire un artefatto finale estendibile e manutenibile sono stati utilizzati i seguenti patterns:

- Per creare oggetti User, Tag, Note, Version si utilizza il pattern Factory. Ogni Factory si occupa di istanziare un oggetto di un tipo specifico, restituendone un'istanza. Le Factories sono state trattate come un Singleton.
- L'accesso alle risorse persistenti (file .db) è gestito mediante Singleton
- Ogni servlet delega logica di business ad un apposito service (Façade), che si occupa dell'interazione con le singole classi.
- Session (presente solo lato client) e le utils per i JSON (JsonParserUtil lato client e GsonJsonParser lato server) sono dei pure fabrication
- La gestione dei permessi sfrutta il pattern State: il comportamento dei metodi canView e canRead si modifica a seconda dello stato dei permessi della nota. In particolare, l'enum Permission rappresenta la State interface con i metodi canView e canRead ridefiniti in ogni valore.

Struttura del progetto

```
├── notes-client
│   ├── pom.xml
│   ├── src
│   │   └── main
│   │       └── java
│   │           └── com
│   │               └── google
│   │                   └── gwt
│   │                       └── sample
│   │                           └── notes
│   │                               ├── client
│   │                               │   ├── CreateNotePanel.java
│   │                               │   ├── HomePanel.java
│   │                               │   ├── JsonParserUtil.java
│   │                               │   ├── LoginPanel.java
│   │                               │   ├── NoteDetailPanel.java
│   │                               │   ├── NoteHistoryPanel.java
│   │                               │   ├── notes.java
│   │                               │   ├── RegistrationPanel.java
│   │                               │   ├── Session.java
│   │                               │   ├── ViewConflictPanel.java
│   │                               │   ├── ViewNotesFilter.java
│   │                               │   └── ViewNotesPanel.java
│   │                               └── notes.gwt.xml
```

```
|   └── target
|       ├── //contiene i file che vengono generati
├── notes-server
|   ├── pom.xml
|   ├── src
|   |   ├── main
|   |   |   ├── java
|   |   |   |   └── com
|   |   |   |       └── google
|   |   |   |           └── gwt
|   |   |   |               └── sample
|   |   |   |                   └── notes
|   |   |   |                       └── server
|   |   |   |                           ├── AbstractDB.java
|   |   |   |                           ├── GsonJsonParser.java
|   |   |   |                           ├── HideNoteService.java
|   |   |   |                           ├── HideNoteServlet.java
|   |   |   |                           ├── JsonParser.java
|   |   |   |                           ├── LoginService.java
|   |   |   |                           ├── LoginServlet.java
|   |   |   |                           ├── MapDBConstants.java
|   |   |   |                           ├── NoteDB.java
|   |   |   |                           ├── NoteFactory.java
|   |   |   |                           ├── NoteService.java
|   |   |   |                           ├── NoteServlet.java
|   |   |   |                           ├── RegisterService.java
|   |   |   |                           ├── RegisterServlet.java
|   |   |   |                           ├── ServiceException.java
|   |   |   |                           ├── TagDB.java
|   |   |   |                           ├── TagFactory.java
|   |   |   |                           ├── TagService.java
|   |   |   |                           ├── TagServlet.java
|   |   |   |                           ├── UserDB.java
|   |   |   |                           ├── UserFactory.java
|   |   |   |                           └── VersionFactory.java
|   |   |   ├── jettyconf
|   |   |   |   └── context.xml
|   |   |   ├── tomcatconf
|   |   |   |   └── context.xml
|   |   |   └── webapp
|   |   |       ├── CreateNote.html
|   |   |       ├── favicon.ico
|   |   |       ├── notes.css
|   |   |       ├── notes.html
|   |   |       ├── ViewNotes.html
|   |   |       └── WEB-INF
```

```
│   │   │           └── web.xml
│   │   └── test
│   │       └── java
│   │           └── com
│   │               └── google
│   │                   └── gwt
│   │                       └── sample
│   │                           └── notes
│   │                               └── server
│   │                                   ├── HideNoteServletTest.java
│   │                                   ├── LoginServletTest.java
│   │                                   ├── NoteServletTest.java
│   │                                   ├── RegisterServletTest.java
│   │                                   └── TagServletTest.java
│   ├── tagsTest.db
│   ├── tagsTest.db.wal.0
│   └── target
│       ├── // contiene i file che vengono generati
├── notes-shared
│   ├── pom.xml
│   ├── src
│   │   └── main
│   │       └── java
│   │           └── com
│   │               └── google
│   │                   └── gwt
│   │                       └── sample
│   │                           └── notes
│   │                               └── shared
│   │                                   ├── ConcreteNote.java
│   │                                   ├── ConcreteTag.java
│   │                                   ├── ConcreteUser.java
│   │                                   ├── ConcreteVersion.java
│   │                                   ├── IdGenerator.java
│   │                                   ├── Note.java
│   │                                   ├── NoteIdGenerator.java
│   │                                   ├── Permission.java
│   │                                   ├── Tag.java
│   │                                   ├── User.java
│   │                                   └── Version.java
│   └── target
│       ├── //contiene i file che vengono generati
├── notes.db
├── pom.xml
├── README.md
├── tags.db
```

```
├── target
│   └── //contiene i file che vengono generati
└── users.db
```

Diario

Descrizione degli eventi svolti durante il progetto

12/05/2025 - Incontro preliminare

Scelta dello stack da utilizzare, pianificazione della struttura degli sprint e dei ruoli, configurazione GWT.

13/05/2025 - Inception (1)

Configurazione GWT, creazione del Modello di Dominio, del Modello dei Casi d'Uso, definizione dei ruoli per il primo sprint, scrum events primo sprint.

14/05/2025 - Inception (2)

Terminazione del Modello di Dominio e del Modello dei Casi d'Uso. Definizione di tutti gli artefatti da produrre per la consegna. Creazione repository del progetto.

Dal 15/05/2025 - Construction

Creazione del progetto.
Di seguito sono riportati tutti gli eventi Scrum svolti per ogni sprint, oltre agli stati della Kanban ad inizio e fine di ogni sprint ed alla suddivisione dei ruoli Scrum durante gli sprint.

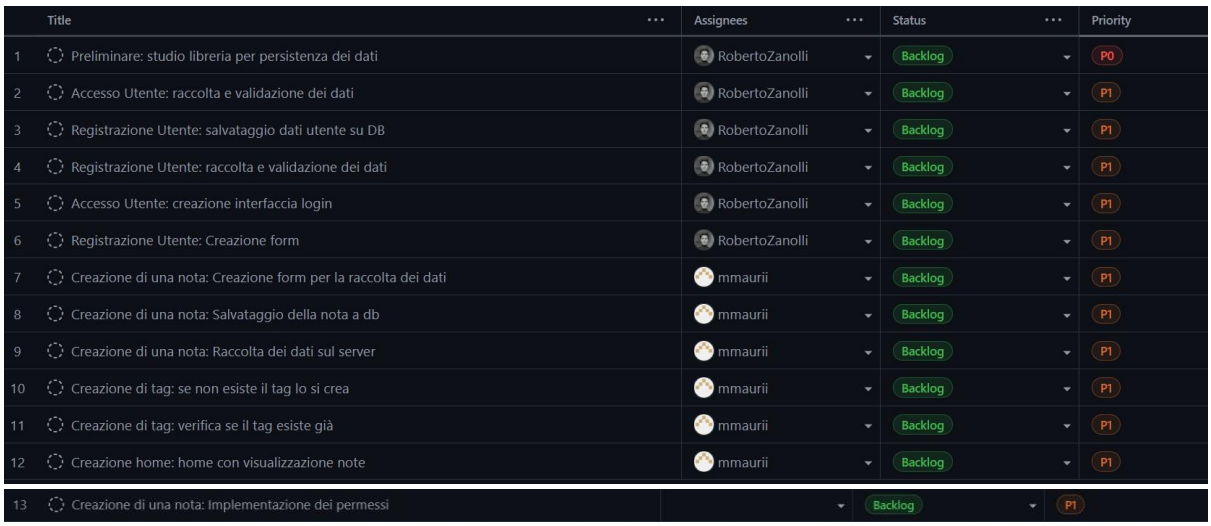
Sprint 1

Development Team: Maurizio Amadori, Roberto Zanolli
Scrum Master: Francesco Maria Fuligni
Product Owner: Lorenzo Magrini

16/05/2025 - Sprint Planning

Definizione del Product Backlog e dello Sprint Backlog del primo sprint, definizione task della settimana per i developer.

Kanban inizio sprint:



	Title	Assignees	Status	Priority
1	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Backlog	P0
2	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	Backlog	P1
3	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	Backlog	P1
4	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	Backlog	P1
5	Accesso Utente: creazione interfaccia login	RobertoZanolli	Backlog	P1
6	Registrazione Utente: Creazione form	RobertoZanolli	Backlog	P1
7	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	Backlog	P1
8	Creazione di una nota: Salvataggio della nota a db	mmaurii	Backlog	P1
9	Creazione di una nota: Raccolta dei dati sul server	mmaurii	Backlog	P1
10	Creazione di tag: se non esiste il tag lo si crea	mmaurii	Backlog	P1
11	Creazione di tag: verifica se il tag esiste già	mmaurii	Backlog	P1
12	Creazione home: home con visualizzazione note	mmaurii	Backlog	P1
13	Creazione di una nota: Implementazione dei permessi		Backlog	P1

17/05/2025 - Daily Scrum

Roberto Zanolli: Ricerca documentazioni su stack e pulizia repository (iniziata precedentemente). Condiviso documento con spiegazione sintetica comandi maven e aggiunta di librerie con maven.

18/05/2025 - Daily Scrum

Roberto Zanolli: Inizio implementazione registrazione utente, logica gestione persistenza per user, implementazione test e servlet per accettare registrazione.

19/05/2025 - Daily Scrum

Roberto Zanolli: Finita implementazione registrazione utente compresa di interfaccia.
Maurizio Amadori e Roberto Zanolli: discussione riguardo testing e debugging.

20/05/2025 - Daily Scrum

Maurizio Amadori: creazione dell'interfaccia di creazione delle note e degli oggetti di interfaccia insieme alla business logic
Roberto Zanolli: Inizio implementazione login, intervento su file di registrazione per introdurre password 4j.

21/05/2025 - Daily Scrum

Maurizio Amadori: implementazione dei test e completamento della business logic di creazione note e tag. Ho avuto difficoltà con l'utilizzo di mapDB nell'implementazione della persistenza.
Roberto Zanolli: Fine implementazione login.

22/05/2025 - Sprint Review e Retrospective

Implementazione logica di creazione note, logica di creazione tag, interfaccia di creazione note, interfaccia di login e logica di login e registrazione. Problemi di concorrenza mapDB risolti con **approccio singleton**. Problema di Creazione note con id univoco partendo da un JSON risolto con una factory e l'implementazione di un generatore di id univoci synchronized. Aggiunta la libreria mockito per i test e Password 4j per l'hashing delle passwords. Ridurre la granularità dei task aggiunti allo sprint backlog per il prossimo sprint.

Kanban fine sprint:

1	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Done	P0
2	Creazione home: home con visualizzazione note	mmaurii	Ready	P1
3	Creazione di una nota: Implementazione dei permessi		Backlog	P1
4	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	In review	P1
5	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	In review	P1
6	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	In review	P1
7	Registrazione Utente: Creazione form	RobertoZanolli	In review	P1
8	Accesso Utente: creazione interfaccia login	RobertoZanolli	In review	P1
9	Creazione di tag: verifica se il tag esiste già	mmaurii	In progress	P1
10	Creazione di tag: se non esiste il tag lo si crea	mmaurii	In progress	P1
11	Creazione di una nota: Raccolta dei dati sul server	mmaurii	In progress	P1
12	Creazione di una nota: Salvataggio della nota a db	mmaurii	In progress	P1
13	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	In progress	P1

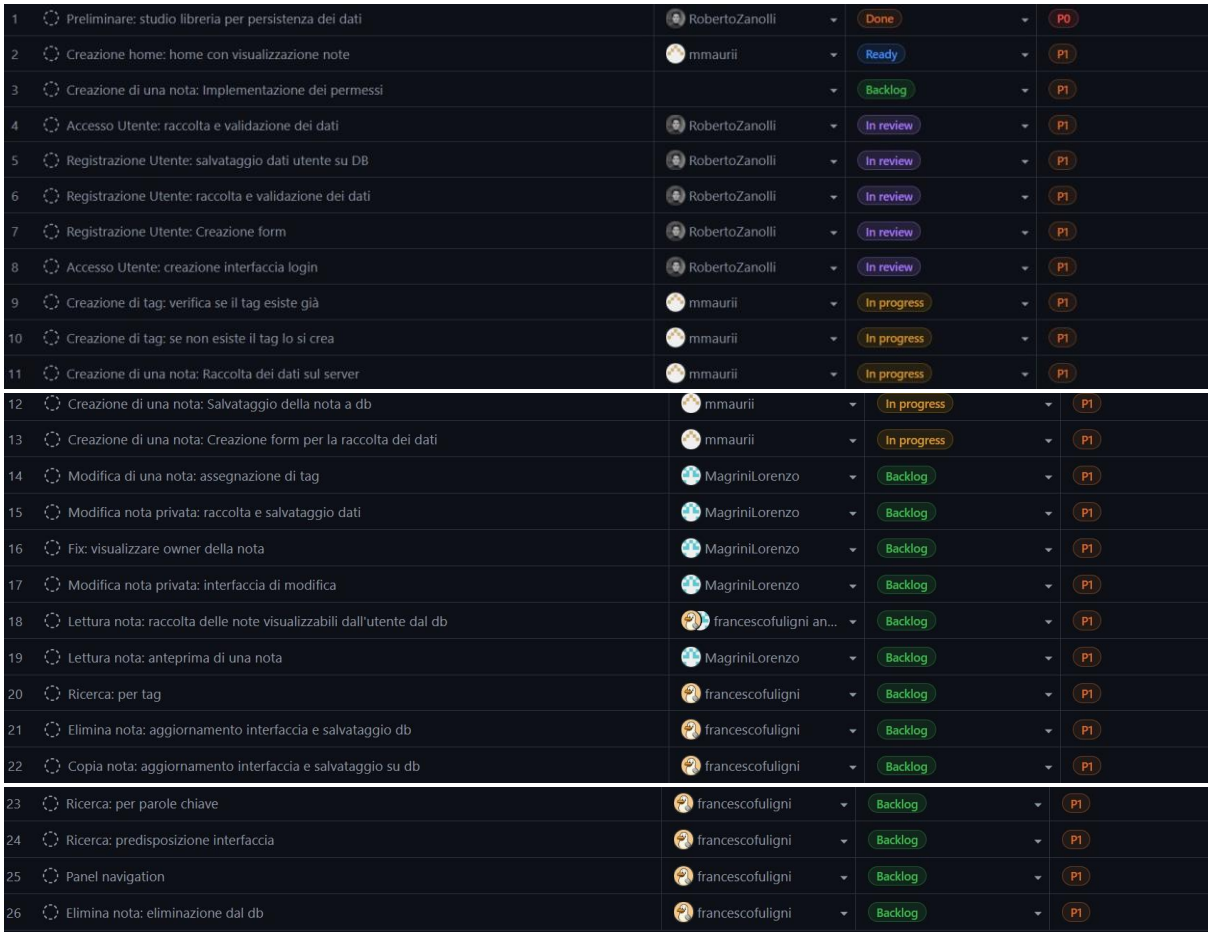
Sprint 2

Development Team: Francesco Maria Fuligni, Lorenzo Magrini
Scrum Master: Maurizio Amadori
Product Owner: Roberto Zanolli

23/05/2025 - Sprint Planning

Discussione di quali item svolgere durante lo svolgimento dello sprint sulla base di quelli che sono bloccati o meno dopo lo svolgimento dell'ultimo sprint. Spiegazione della struttura del progetto e logica di test. Confronto sulla modalità di realizzazione dell'interfaccia e della struttura server side.

Kanban inizio sprint:



1	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Done	P0
2	Creazione home: home con visualizzazione note	mmaurii	Ready	P1
3	Creazione di una nota: Implementazione dei permessi		Backlog	P1
4	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	In review	P1
5	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	In review	P1
6	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	In review	P1
7	Registrazione Utente: Creazione form	RobertoZanolli	In review	P1
8	Accesso Utente: creazione interfaccia login	RobertoZanolli	In review	P1
9	Creazione di tag: verifica se il tag esiste già	mmaurii	In progress	P1
10	Creazione di tag: se non esiste il tag lo si crea	mmaurii	In progress	P1
11	Creazione di una nota: Raccolta dei dati sul server	mmaurii	In progress	P1
12	Creazione di una nota: Salvataggio della nota a db	mmaurii	In progress	P1
13	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	In progress	P1
14	Modifica di una nota: assegnazione di tag	MagriniLorenzo	Backlog	P1
15	Modifica nota privata: raccolta e salvataggio dati	MagriniLorenzo	Backlog	P1
16	Fix: visualizzare owner della nota	MagriniLorenzo	Backlog	P1
17	Modifica nota privata: interfaccia di modifica	MagriniLorenzo	Backlog	P1
18	Lettura nota: raccolta delle note visualizzabili dall'utente dal db	francescofuligni an...	Backlog	P1
19	Lettura nota: anteprima di una nota	MagriniLorenzo	Backlog	P1
20	Ricerca: per tag	francescofuligni	Backlog	P1
21	Elimina nota: aggiornamento interfaccia e salvataggio db	francescofuligni	Backlog	P1
22	Copia nota: aggiornamento interfaccia e salvataggio su db	francescofuligni	Backlog	P1
23	Ricerca: per parole chiave	francescofuligni	Backlog	P1
24	Ricerca: predisposizione interfaccia	francescofuligni	Backlog	P1
25	Panel navigation	francescofuligni	Backlog	P1
26	Elimina nota: eliminazione dal db	francescofuligni	Backlog	P1

24/05/2025 - Daily Scrum

Maurizio Amadori: completamento dei test per la creazione di tag e note.

25/05/2025 - Daily Scrum

Francesco Maria Fuligni: Predisposizione dell'interfaccia per il filtro sulle note e delle strutture server side per l'eliminazione delle note.
Lorenzo Magrini: Analisi struttura del codice creato nello sprint precedente e della struttura prevista dallo stack.

26/05/2025 - Daily Scrum

Francesco Maria Fuligni: predisposizione delle classi servlet per le operazioni di copia e di get delle note, creazione di una classe astratta estesa dalle classi DB per uniformità tra le strutture e riuso di codice.

27/05/2025 - Daily Scrum

Francesco Maria Fuligni: navigazione tra i pannelli, aggiornamento delle classi Servlet, realizzazione e miglioramento di test.
Lorenzo Magrini: Modifica del panel View Notes per inserire una lista di tutte le note.

28/05/2025 - Daily Scrum

Francesco Maria Fuligni: implementazione della ricerca tramite filtri, collegamento tra operazioni di eliminazione e copia delle note a backend con frontend.
Lorenzo Magrini: Creazione del panel per visionare i dettagli delle note, ed inizio creazione logica di modifica note.

29/05/2025 - Sprint Review e Retrospective

Implementata vista delle note completa e dettaglio di una nota. Realizzata la ricerca tramite filtri su parole chiave e tag delle note. Impostazione della navigazione tra le interfacce e predisposizione per la gestione dei permessi e del mantenimento di dati di sessione. Realizzata l'interfaccia per la modifica di una nota. Modificata la struttura delle classi DB server side, aggiornamento della classe Nota per implementare nuove funzionalità e dei test. Pulizia del codice con rimozione di classi non più necessarie.

Kanban fine sprint:

1	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Done	P0
2	Creazione home: home con visualizzazione note	mmaurii	Done	P1
3	Creazione di una nota: Implementazione dei permessi		Backlog	P1
4	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1
5	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	Done	P1
6	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1
7	Registrazione Utente: Creazione form	RobertoZanolli	Done	P1
8	Accesso Utente: creazione interfaccia login	RobertoZanolli	Done	P1
9	Creazione di tag: verifica se il tag esiste già	mmaurii	Done	P1
10	Creazione di tag: se non esiste il tag lo si crea	mmaurii	Done	P1
11	Creazione di una nota: Raccolta dei dati sul server	mmaurii	Done	P1
12	Creazione di una nota: Salvataggio della nota a db	mmaurii	Done	P1
13	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	Done	P1
14	Modifica di una nota: assegnazione di tag	MagriniLorenzo	In progress	P1
15	Modifica nota privata: raccolta e salvataggio dati	MagriniLorenzo	In progress	P1
16	Fix: visualizzare owner della nota	MagriniLorenzo	In progress	P1
17	Modifica nota privata: interfaccia di modifica	MagriniLorenzo	In review	P1
18	Lettura nota: raccolta delle note visualizzabili dall'utente dal db	francescofuligni an...	Done	P1
19	Lettura nota: anteprima di una nota	MagriniLorenzo	Done	P1
20	Ricerca: per tag	francescofuligni	Done	P1
21	Elimina nota: aggiornamento interfaccia e salvataggio db	francescofuligni	Done	P1
22	Copia nota: aggiornamento interfaccia e salvataggio su db	francescofuligni	Done	P1
23	Ricerca: per parole chiave	francescofuligni	Done	P1
24	Ricerca: predisposizione interfaccia	francescofuligni	Done	P1
25	Panel navigation	francescofuligni	Done	P1
26	Elimina nota: eliminazione dal db	francescofuligni	Done	P1
27	Fix: ricerca contemporanea tag e keywords	francescofuligni	Done	P1

Sprint 3

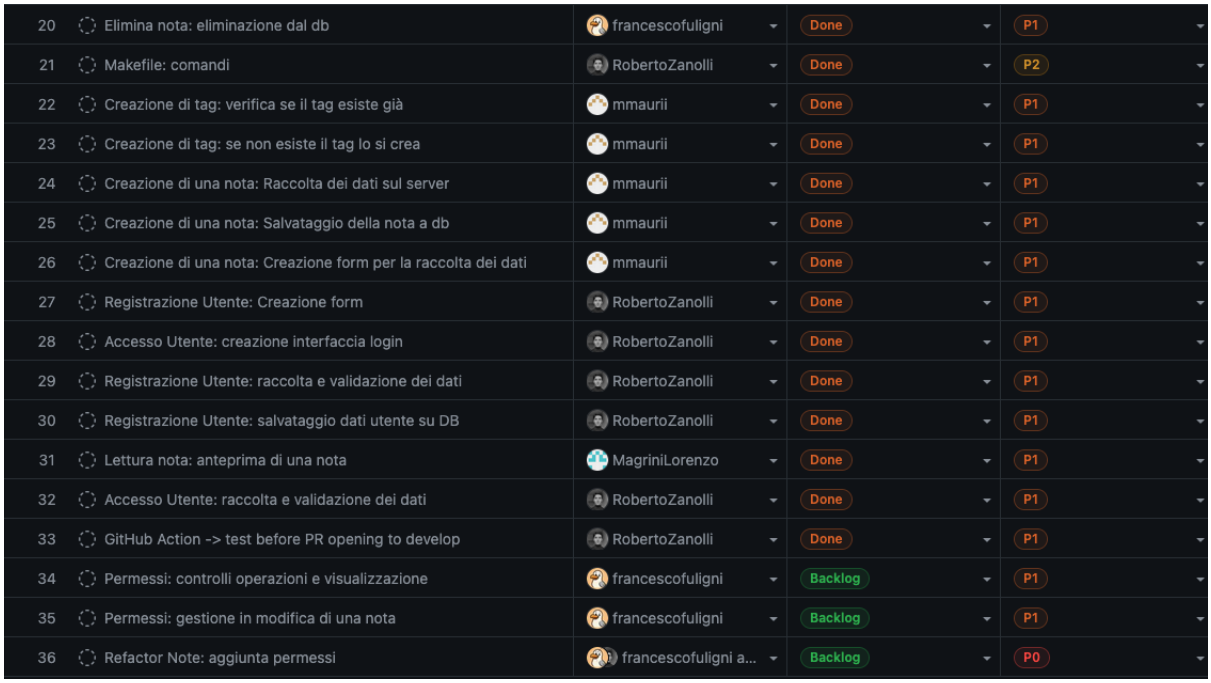
Development Team: Francesco Maria Fuligni, Roberto Zanolli
Scrum Master: Lorenzo Magrini
Product Owner: Maurizio Amadori

30/05/2025 - Sprint Planning

Discussione di quali item svolgere durante lo svolgimento dello sprint sulla base di quelli che sono bloccati o meno dopo lo svolgimento dell'ultimo sprint. Scelta di non gestire ancora la concorrenza e la modifica in concorrenza, preferendo procedere al completamento delle feature mancanti, come la gestione del versionamento delle note e dei permessi per accesso e modifica delle stesse.

Kanban inizio sprint:

1	Refactor Note: mantenimento versionamento	francescofuligni a...	Ready	P0
2	Versionamento: aggiornamento versione alla modifica della nota	RobertoZanolli	Ready	P1
3	Refactor Note: owner email	francescofuligni a...	Ready	P2
4	Permessi: login e logout	francescofuligni	Backlog	P2
5	Versionamento: visualizzazione delle versioni	RobertoZanolli	Backlog	P2
6	Fix: ricerca contemporanea tag e keywords	francescofuligni	Done	P1
7	Modifica di una nota: assegnazione di tag	MagriniLorenzo	Done	P1
8	Creazione home: home con visualizzazione note	mmaurii	Done	P1
9	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Done	P0
10	Modifica nota privata: interfaccia di modifica	MagriniLorenzo	Done	P1
11	Fix: visualizzare owner della nota	MagriniLorenzo	Done	P1
12	Modifica nota privata: raccolta e salvataggio dati	MagriniLorenzo	Done	P1
13	Ricerca: per tag	francescofuligni	Done	P1
14	Lettura nota: raccolta delle note visualizzabili dall'utente dal db	francescofuligni a...	Done	P1
15	Elimina nota: aggiornamento interfaccia e salvataggio db	francescofuligni	Done	P1
16	Copia nota: aggiornamento interfaccia e salvataggio su db	francescofuligni	Done	P1
17	Ricerca: per parole chiave	francescofuligni	Done	P1
18	Ricerca: predisposizione interfaccia	francescofuligni	Done	P1
19	Panel navigation	francescofuligni	Done	P1



20	Elimina nota: eliminazione dal db	francescofuligni	Done	P1
21	Makefile: comandi	RobertoZanolli	Done	P2
22	Creazione di tag: verifica se il tag esiste già	mmaurii	Done	P1
23	Creazione di tag: se non esiste il tag lo si crea	mmaurii	Done	P1
24	Creazione di una nota: Raccolta dei dati sul server	mmaurii	Done	P1
25	Creazione di una nota: Salvataggio della nota a db	mmaurii	Done	P1
26	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	Done	P1
27	Registrazione Utente: Creazione form	RobertoZanolli	Done	P1
28	Accesso Utente: creazione interfaccia login	RobertoZanolli	Done	P1
29	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1
30	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	Done	P1
31	Lettura nota: anteprima di una nota	MagriniLorenzo	Done	P1
32	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1
33	GitHub Action -> test before PR opening to develop	RobertoZanolli	Done	P1
34	Permessi: controlli operazioni e visualizzazione	francescofuligni	Backlog	P1
35	Permessi: gestione in modifica di una nota	francescofuligni	Backlog	P1
36	Refactor Note: aggiunta permessi	francescofuligni a...	Backlog	P0

31/05/2025 - Daily Scrum

Francesco Maria Fuligni e Roberto Zanolli (pair programming): refactoring della struttura delle note e delle factory per predisposizione all'implementazione di permessi e versionamento.

01/06/2025 - Daily Scrum

Francesco Maria Fuligni e Roberto Zanolli (pair programming): implementazione di una struttura per memorizzare dati di sessione, refactoring dei servlet per operazioni sulle note.
Lorenzo Magrini: Terminata modifica delle note.

02/06/2025 - Daily Scrum

Roberto Zanolli: creazione dell'interfaccia per la visualizzazione delle versioni delle note, fix di bug nella creazione delle note, refactoring servlet e factory per operazioni sulle note, refactoring test per testare varie casistiche con la nuova struttura delle versioni.

03/06/2025 - Daily Scrum

Francesco Maria Fuligni: implementazione backend dei permessi su una nota, gestione della rimozione dalla visualizzazione, modifica dell'interfaccia per la corretta gestione dei permessi.

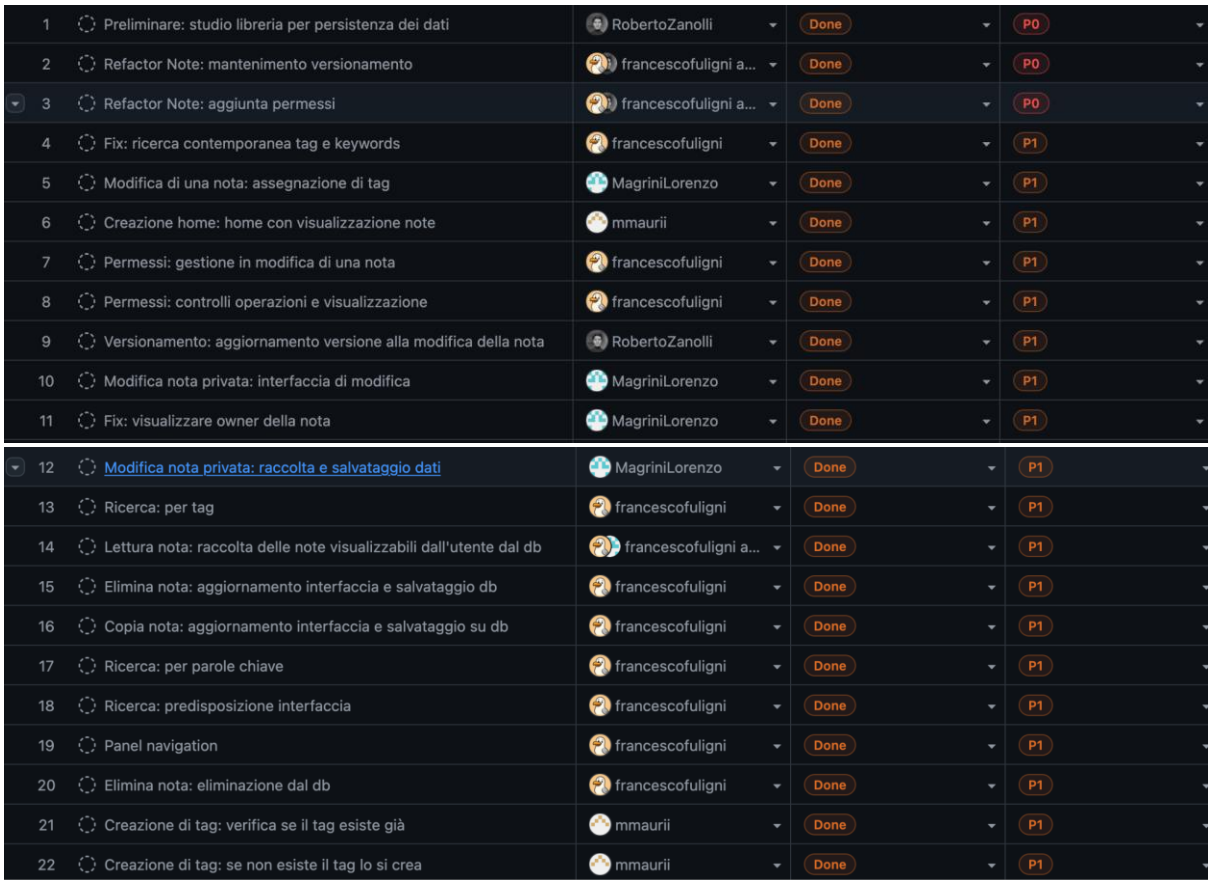
04/06/2025 - Daily Scrum

Francesco Maria Fuligni e Roberto Zanolli (pair programming): fix di bug nella copia di una nota e nella modifica dei permessi di una nota, controlli aggiuntivi sui permessi, fix bug nella memorizzazione della mail del creatore in una nota, fix bug nel formato di memorizzazione delle date backend.

05/06/2025 - Sprint Review e Retrospective

Illustrazione della struttura modificata delle note e dei servlet per la gestione delle stesse. Aggiornamento sulle modalità di implementazione dei permessi e su come vengono effettuati i controlli a frontend e backend. Confronto sulle modalità di visualizzazione delle versioni di una nota e sui processi di modifica e di copia di una nota. Predisposizione del codice per l'implementazione delle feature di gestione della concorrenza.

Kanban fine sprint:



1	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Done	P0
2	Refactor Note: mantenimento versionamento	francescofuligni a...	Done	P0
3	Refactor Note: aggiunta permessi	francescofuligni a...	Done	P0
4	Fix: ricerca contemporanea tag e keywords	francescofuligni	Done	P1
5	Modifica di una nota: assegnazione di tag	MagriniLorenzo	Done	P1
6	Creazione home: home con visualizzazione note	mmaurii	Done	P1
7	Permessi: gestione in modifica di una nota	francescofuligni	Done	P1
8	Permessi: controlli operazioni e visualizzazione	francescofuligni	Done	P1
9	Versionamento: aggiornamento versione alla modifica della nota	RobertoZanolli	Done	P1
10	Modifica nota privata: interfaccia di modifica	MagriniLorenzo	Done	P1
11	Fix: visualizzare owner della nota	MagriniLorenzo	Done	P1
12	Modifica nota privata: raccolta e salvataggio dati	MagriniLorenzo	Done	P1
13	Ricerca: per tag	francescofuligni	Done	P1
14	Lettura nota: raccolta delle note visualizzabili dall'utente dal db	francescofuligni a...	Done	P1
15	Elimina nota: aggiornamento interfaccia e salvataggio db	francescofuligni	Done	P1
16	Copia nota: aggiornamento interfaccia e salvataggio su db	francescofuligni	Done	P1
17	Ricerca: per parole chiave	francescofuligni	Done	P1
18	Ricerca: predisposizione interfaccia	francescofuligni	Done	P1
19	Panel navigation	francescofuligni	Done	P1
20	Elimina nota: eliminazione dal db	francescofuligni	Done	P1
21	Creazione di tag: verifica se il tag esiste già	mmaurii	Done	P1
22	Creazione di tag: se non esiste il tag lo si crea	mmaurii	Done	P1

23	Creazione di tag: se non esiste il tag lo si crea	mmaurii	Done	P1
24	Creazione di una nota: Raccolta dei dati sul server	mmaurii	Done	P1
25	Creazione di una nota: Salvataggio della nota a db	mmaurii	Done	P1
26	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	Done	P1
27	Registrazione Utente: Creazione form	RobertoZanolli	Done	P1
28	Accesso Utente: creazione interfaccia login	RobertoZanolli	Done	P1
29	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1
30	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	Done	P1
31	Lettura nota: anteprima di una nota	MagriniLorenzo	Done	P1
32	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1
33	GitHub Action -> test before PR opening to develop	RobertoZanolli	Done	P1
34	Permessi: controlli operazioni e visualizzazione	francescofuligni	Backlog	P1

Sprint 4

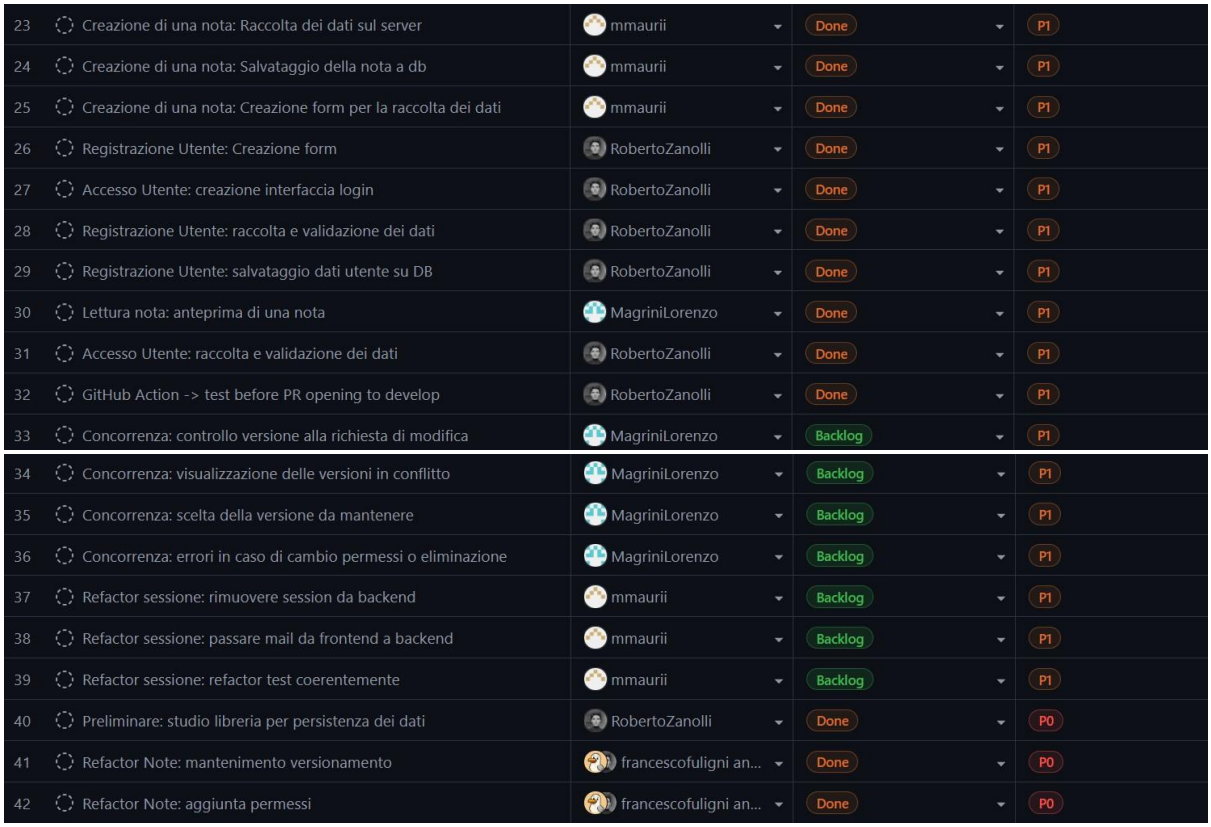
Development Team: Maurizio Amadori, Lorenzo Magrini
Scrum Master: Roberto Zanolli
Product Owner: Francesco Maria Fuligni

06/06/2025 - Sprint Planning

Pianificazione dell'approccio per la gestione della concorrenza in modifica delle note. Confronto sulla possibilità di refactor del codice. Miglioramento della struttura di gestione della mail utente a backend: creare una nuova struttura per gestire la sessione. Definizione delle attività mancanti al termine del progetto.

Kanban inizio sprint:

1	Refactor Note: owner email	francescofuligni an...	Done	P2
2	Makefile: comandi	RobertoZanolli	Done	P2
3	Versionamento: visualizzazione delle versioni	RobertoZanolli	Done	P2
4	Fix: ricerca contemporanea tag e keywords	francescofuligni	Done	P1
5	Modifica di una nota: assegnazione di tag	MagriniLorenzo	Done	P1
6	Creazione home: home con visualizzazione note	mmaurii	Done	P1
7	Permessi: gestione in modifica di una nota	francescofuligni	Done	P1
8	Permessi: controlli operazioni e visualizzazione	francescofuligni	Done	P1
9	Versionamento: aggiornamento versione alla modifica della nota	RobertoZanolli	Done	P1
10	Modifica nota privata: interfaccia di modifica	MagriniLorenzo	Done	P1
11	Fix: visualizzare owner della nota	MagriniLorenzo	Done	P1
12	Modifica nota privata: raccolta e salvataggio dati	MagriniLorenzo	Done	P1
13	Ricerca: per tag	francescofuligni	Done	P1
14	Lettura nota: raccolta delle note visualizzabili dall'utente dal db	francescofuligni an...	Done	P1
15	Elimina nota: aggiornamento interfaccia e salvataggio db	francescofuligni	Done	P1
16	Copia nota: aggiornamento interfaccia e salvataggio su db	francescofuligni	Done	P1
17	Ricerca: per parole chiave	francescofuligni	Done	P1
18	Ricerca: predisposizione interfaccia	francescofuligni	Done	P1
19	Panel navigation	francescofuligni	Done	P1
20	Elimina nota: eliminazione dal db	francescofuligni	Done	P1
21	Creazione di tag: verifica se il tag esiste già	mmaurii	Done	P1
22	Creazione di tag: se non esiste il tag lo si crea	mmaurii	Done	P1



23	Creazione di una nota: Raccolta dei dati sul server	mmaurii	Done	P1
24	Creazione di una nota: Salvataggio della nota a db	mmaurii	Done	P1
25	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	Done	P1
26	Registrazione Utente: Creazione form	RobertoZanolli	Done	P1
27	Accesso Utente: creazione interfaccia login	RobertoZanolli	Done	P1
28	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1
29	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	Done	P1
30	Lettura nota: anteprima di una nota	MagriniLorenzo	Done	P1
31	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1
32	GitHub Action -> test before PR opening to develop	RobertoZanolli	Done	P1
33	Concorrenza: controllo versione alla richiesta di modifica	MagriniLorenzo	Backlog	P1
34	Concorrenza: visualizzazione delle versioni in conflitto	MagriniLorenzo	Backlog	P1
35	Concorrenza: scelta della versione da mantenere	MagriniLorenzo	Backlog	P1
36	Concorrenza: errori in caso di cambio permessi o eliminazione	MagriniLorenzo	Backlog	P1
37	Refactor sessione: rimuovere session da backend	mmaurii	Backlog	P1
38	Refactor sessione: passare mail da frontend a backend	mmaurii	Backlog	P1
39	Refactor sessione: refactor test coerentemente	mmaurii	Backlog	P1
40	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Done	P0
41	Refactor Note: mantenimento versionamento	francescofuligni an...	Done	P0
42	Refactor Note: aggiunta permessi	francescofuligni an...	Done	P0

07/06/2025 - Daily Scrum

Maurizio Amadori: Valutazione refactoring della sessione e riflessione su possibili modalità di implementazione.

08/06/2025 - Daily Scrum

Maurizio Amadori: refactoring del codice per valutare il corretto funzionamento della sessione.
Lorenzo Magrini: Studio della possibile gestione dei conflitti.

09/06/2025 - Daily Scrum

Maurizio Amadori: si è deciso di tenere la sessione e di procedere con il test e la correzione di eventuali bug.

10/06/2025 - Daily Scrum

Maurizio Amadori: scrittura test e bug fix della sessione.

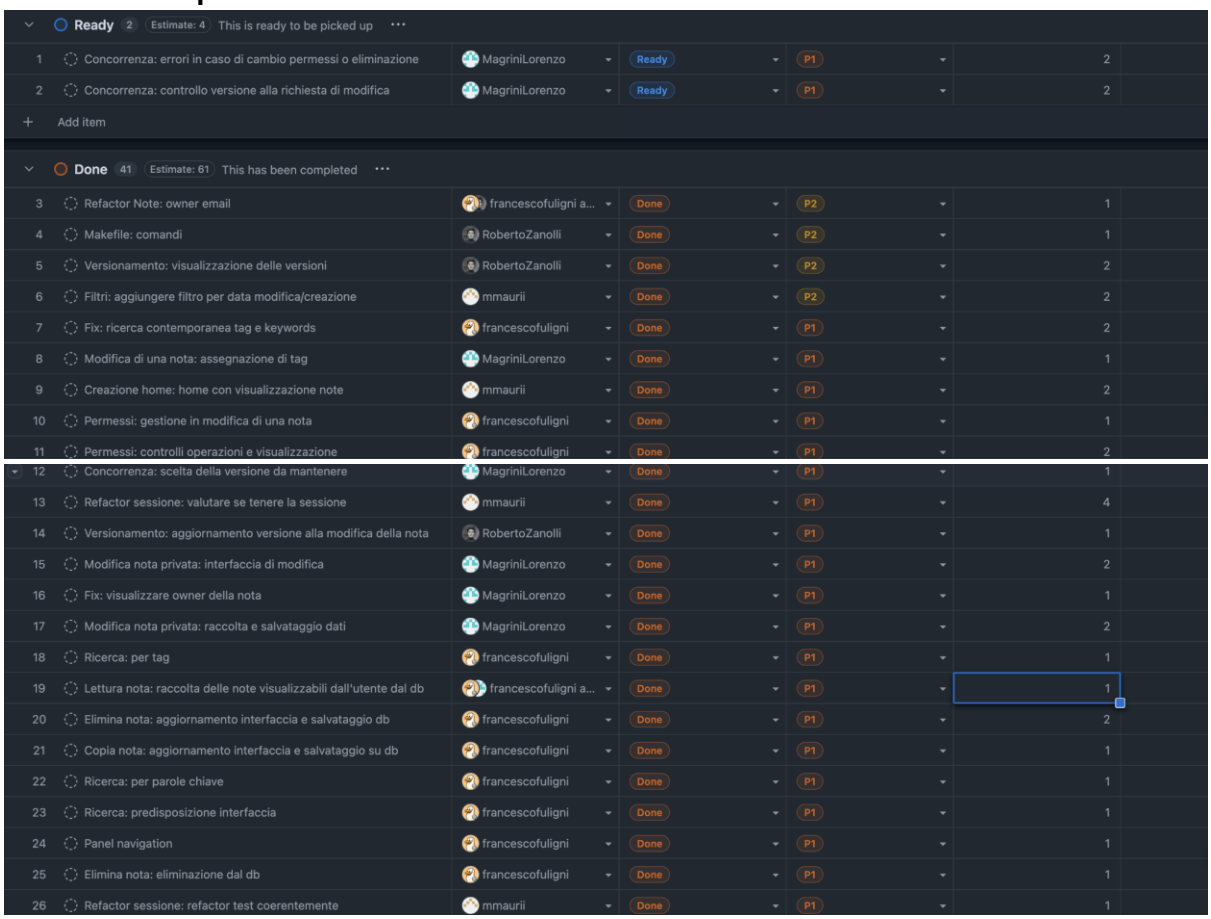
11/06/2025 - Daily Scrum

Maurizio Amadori: aggiungerò la funzione di filtro in base alla data di modifica, implementerò l'interfaccia per la gestione dei conflitti e del suo inserimento nella navigazione.
Lorenzo Magrini: Implementata logica di cattura e gestione del conflitto in modifica sulle note.

12/06/2025 - Sprint Review e Retrospective

Illustrazione del meccanismo di gestione dei conflitti implementati. Spiegazione del nuovo meccanismo di gestione della sessione. Implementazione di nuovi filtri per la ricerca delle note sulla base delle date di ultima modifica. Definizione di uno sprint extra per la risoluzione di bug individuati nella gestione della concorrenza e il completamento di funzionalità mancanti.

Kanban fine sprint:



Ready 2	Estimate: 4	This is ready to be picked up			
1	Concorrenza: errori in caso di cambio permessi o eliminazione	MagriniLorenzo	Ready	P1	2
2	Concorrenza: controllo versione alla richiesta di modifica	MagriniLorenzo	Ready	P1	2
+	Add item				
Done 41	Estimate: 61	This has been completed			
3	Refactor Note: owner email	francescofuligni a...	Done	P2	1
4	Makefile: comandi	RobertoZanolli	Done	P2	1
5	Versionamento: visualizzazione delle versioni	RobertoZanolli	Done	P2	2
6	Filtri: aggiungere filtro per data modifica/creazione	mmaurii	Done	P2	2
7	Fix: ricerca contemporanea tag e keywords	francescofuligni	Done	P1	2
8	Modifica di una nota: assegnazione di tag	MagriniLorenzo	Done	P1	1
9	Creazione home: home con visualizzazione note	mmaurii	Done	P1	2
10	Permessi: gestione in modifica di una nota	francescofuligni	Done	P1	1
11	Permessi: controlli operazioni e visualizzazione	francescofuligni	Done	P1	2
12	Concorrenza: scelta della versione da mantenere	MagriniLorenzo	Done	P1	1
13	Refactor sessione: valutare se tenere la sessione	mmaurii	Done	P1	4
14	Versionamento: aggiornamento versione alla modifica della nota	RobertoZanolli	Done	P1	1
15	Modifica nota privata: interfaccia di modifica	MagriniLorenzo	Done	P1	2
16	Fix: visualizzare owner della nota	MagriniLorenzo	Done	P1	1
17	Modifica nota privata: raccolta e salvataggio dati	MagriniLorenzo	Done	P1	2
18	Ricerca: per tag	francescofuligni	Done	P1	1
19	Lettura nota: raccolta delle note visualizzabili dall'utente dal db	francescofuligni a...	Done	P1	1
20	Elimina nota: aggiornamento interfaccia e salvataggio db	francescofuligni	Done	P1	2
21	Copia nota: aggiornamento interfaccia e salvataggio su db	francescofuligni	Done	P1	1
22	Ricerca: per parole chiave	francescofuligni	Done	P1	1
23	Ricerca: predisposizione interfaccia	francescofuligni	Done	P1	1
24	Panel navigation	francescofuligni	Done	P1	1
25	Elimina nota: eliminazione dal db	francescofuligni	Done	P1	1
26	Refactor sessione: refactor test coerentemente	mmaurii	Done	P1	1

27	Refactor sessione: migliorare l'implementazione	mmaurii	Done	P1	1
28	Creazione di tag: verifica se il tag esiste già	mmaurii	Done	P1	1
29	Creazione di tag: se non esiste il tag lo si crea	mmaurii	Done	P1	1
30	Creazione di una nota: Raccolta dei dati sul server	mmaurii	Done	P1	1
31	Creazione di una nota: Salvataggio della nota a db	mmaurii	Done	P1	2
32	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	Done	P1	1
33	Registrazione Utente: Creazione form	RobertoZanolli	Done	P1	1
34	Accesso Utente: creazione interfaccia login	RobertoZanolli	Done	P1	1
35	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1	1
36	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	Done	P1	1
37	Lettura nota: anteprima di una nota	MagriniLorenzo	Done	P1	1
38	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1	1
39	GitHub Action -> test before PR opening to develop	RobertoZanolli	Done	P1	2
40	Concorrenza: visualizzazione delle versioni in conflitto	mmaurii	Done	P1	5
41	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Done	P0	2
42	Refactor Note: mantenimento versionamento	francescofuligni a...	Done	P0	3
43	Refactor Note: aggiunta permessi	francescofuligni a...	Done	P0	1

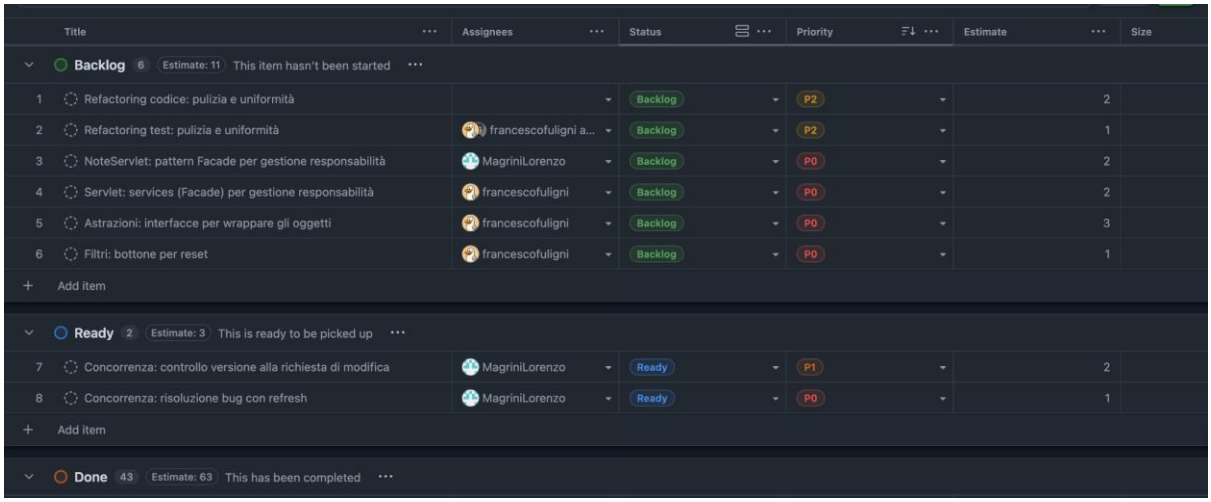
Sprint 5

Development Team: Francesco Maria Fuligni, Lorenzo Magrini,
Scrum Master: Roberto Zanolli
Product Owner: Maurizio Amadori

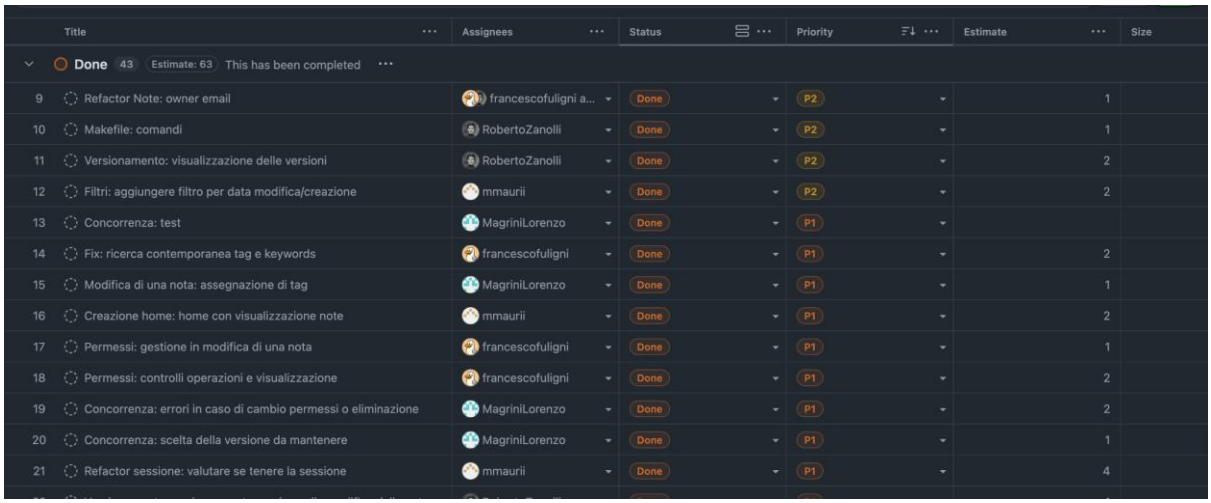
13/06/2025 - Sprint Planning

Pianificazione dei miglioramenti da applicare al codice e controllo delle funzionalità residue da implementare. Risoluzione di bug nella gestione dei conflitti. Miglioramento dei meccanismi di gestione dei conflitti e di filtraggio della lista delle note. Controllo e consolidamento degli artefatti realizzati per la consegna del progetto.

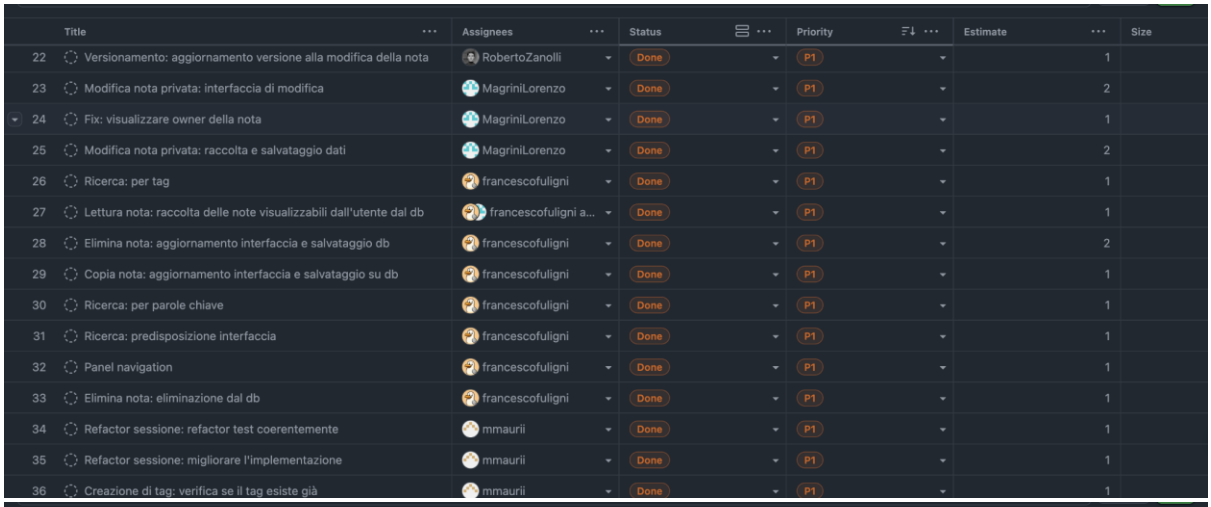
Kanban inizio sprint:



	Title	Assignees	Status	Priority	Estimate	Size
Backlog 6 Estimate: 11 This item hasn't been started						
1	Refactoring codice: pulizia e uniformità		Backlog	P2	2	
2	Refactoring test: pulizia e uniformità	francescofuligni a...	Backlog	P2	1	
3	NoteServlet: pattern Facade per gestione responsabilità	MagriniLorenzo	Backlog	P0	2	
4	Servlet: services (Facade) per gestione responsabilità	francescofuligni	Backlog	P0	2	
5	Astrazioni: interfacce per wrappare gli oggetti	francescofuligni	Backlog	P0	3	
6	Filtri: bottone per reset	francescofuligni	Backlog	P0	1	
+ Add item						
Ready 2 Estimate: 3 This is ready to be picked up						
7	Concorrenza: controllo versione alla richiesta di modifica	MagriniLorenzo	Ready	P1	2	
8	Concorrenza: risoluzione bug con refresh	MagriniLorenzo	Ready	P0	1	
+ Add item						
Done 43 Estimate: 63 This has been completed						



	Title	Assignees	Status	Priority	Estimate	Size
Done 43 Estimate: 63 This has been completed						
9	Refactor Note: owner email	francescofuligni a...	Done	P2	1	
10	Makefile: comandi	RobertoZanolli	Done	P2	1	
11	Versionamento: visualizzazione delle versioni	RobertoZanolli	Done	P2	2	
12	Filtri: aggiungere filtro per data modifica/creazione	mmaurii	Done	P2	2	
13	Concorrenza: test	MagriniLorenzo	Done	P1		
14	Fix: ricerca contemporanea tag e keywords	francescofuligni	Done	P1	2	
15	Modifica di una nota: assegnazione di tag	MagriniLorenzo	Done	P1	1	
16	Creazione home: home con visualizzazione note	mmaurii	Done	P1	2	
17	Permessi: gestione in modifica di una nota	francescofuligni	Done	P1	1	
18	Permessi: controlli operazioni e visualizzazione	francescofuligni	Done	P1	2	
19	Concorrenza: errori in caso di cambio permessi o eliminazione	MagriniLorenzo	Done	P1	2	
20	Concorrenza: scelta della versione da mantenere	MagriniLorenzo	Done	P1	1	
21	Refactor sessione: valutare se tenere la sessione	mmaurii	Done	P1	4	



	Title	Assignees	Status	Priority	Estimate	Size
22	Versionamento: aggiornamento versione alla modifica della nota	RobertoZanolli	Done	P1	1	
23	Modifica nota privata: interfaccia di modifica	MagriniLorenzo	Done	P1	2	
24	Fix: visualizzare owner della nota	MagriniLorenzo	Done	P1	1	
25	Modifica nota privata: raccolta e salvataggio dati	MagriniLorenzo	Done	P1	2	
26	Ricerca: per tag	francescofuligni	Done	P1	1	
27	Lettura nota: raccolta delle note visualizzabili dall'utente dal db	francescofuligni a...	Done	P1	1	
28	Elimina nota: aggiornamento interfaccia e salvataggio db	francescofuligni	Done	P1	2	
29	Copia nota: aggiornamento interfaccia e salvataggio su db	francescofuligni	Done	P1	1	
30	Ricerca: per parole chiave	francescofuligni	Done	P1	1	
31	Ricerca: predisposizione interfaccia	francescofuligni	Done	P1	1	
32	Panel navigation	francescofuligni	Done	P1	1	
33	Elimina nota: eliminazione dal db	francescofuligni	Done	P1	1	
34	Refactor sessione: refactor test coerentemente	mmaurii	Done	P1	1	
35	Refactor sessione: migliorare l'implementazione	mmaurii	Done	P1	1	
36	Creazione di tag: verifica se il tag esiste già	mmaurii	Done	P1	1	

	Title	Assignees	Status	Priority	Estimate	Size
37	Creazione di tag: se non esiste il tag lo si crea	mmaurii	Done	P1	1	
38	Creazione di una nota: Raccolta dei dati sul server	mmaurii	Done	P1	1	
39	Creazione di una nota: Salvataggio della nota a db	mmaurii	Done	P1	2	
40	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	Done	P1	1	
41	Registrazione Utente: Creazione form	RobertoZanolli	Done	P1	1	
42	Accesso Utente: creazione interfaccia login	RobertoZanolli	Done	P1	1	
43	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1	1	
44	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	Done	P1	1	
45	Lettura nota: anteprima di una nota	MagriniLorenzo	Done	P1	1	
46	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1	1	
47	GitHub Action -> test before PR opening to develop	RobertoZanolli	Done	P1	2	
48	Concorrenza: visualizzazione delle versioni in conflitto	mmaurii	Done	P1	5	
49	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Done	P0	2	
50	Refactor Note: mantenimento versionamento	francescofuligni a...	Done	P0	3	
51	Refactor Note: aggiunta permessi	francescofuligni a...	Done	P0	1	

14/06/2025 - Daily Scrum

Lorenzo Magrini: Inserita Façade per la gestione del NoteServlet separando le responsabilità, aggiunta dei test sulla gestione della concorrenza. Aggiungerò test per la concorrenza nei test su Note Servlet
Francesco Maria Fuligni: implementate interfacce per migliorare il codice in accesso agli oggetti, creazione di services per migliorare la gestione delle responsabilità dei singoli servlet, refactoring del codice coerentemente.
Francesco Maria Fuligni e Roberto Zanolli (pair programming): Testing delle funzionalità implementate, pulizia del codice nei test ed ove necessario, uniformità dei commenti e dei messaggi di errore.

15/06/2025 - Sprint Review e Retrospective

Demo di tutte le funzionalità dell'applicazione. Controllo dei requisiti richiesti e di quelli implementati e di tutti gli artefatti prodotti. Consegna del progetto.

Kanban fine sprint:

	Title	Assignees	Status	Priority	Estimate	Size
1	Refactor Note: owner email	francescofuligni a...	Done	P2	1	
2	Makefile: comandi	RobertoZanolli	Done	P2	1	
3	Versionamento: visualizzazione delle versioni	RobertoZanolli	Done	P2	2	
4	Filtri: aggiungere filtro per data modifica/creazione	mmaurii	Done	P2	2	
5	Refactoring test: pulizia e uniformità	RobertoZanolli	Done	P2	1	
6	Refactoring codice: pulizia e uniformità	RobertoZanolli	Done	P2	2	
7	Concorrenza: test	MagriniLorenzo	Done	P1		
8	Fix: ricerca contemporanea tag e keywords	francescofuligni	Done	P1	2	
9	Modifica di una nota: assegnazione di tag	MagriniLorenzo	Done	P1	1	
10	Creazione home: home con visualizzazione note	mmaurii	Done	P1	2	
11	Permessi: gestione in modifica di una nota	francescofuligni	Done	P1	1	
12	Permessi: controlli operazioni e visualizzazione	francescofuligni	Done	P1	2	
13	Concorrenza: errori in caso di cambio permessi o eliminazione	MagriniLorenzo	Done	P1	2	
14	Concorrenza: scelta della versione da mantenere	MagriniLorenzo	Done	P1	1	
15	Refactor sessione: valutare se tenere la sessione	mmaurii	Done	P1	4	
16	Versionamento: aggiornamento versione alla modifica della nota	RobertoZanolli	Done	P1	1	
17	Modifica nota privata: interfaccia di modifica	MagriniLorenzo	Done	P1	2	
18	Fix: visualizzare owner della nota	MagriniLorenzo	Done	P1	1	

	Title	Assignees	Status	Priority	Estimate	Size
19	Modifica nota privata: raccolta e salvataggio dati	MagriniLorenzo	Done	P1	2	
20	Ricerca: per tag	francescofuligni	Done	P1	1	
21	Lettura nota: raccolta delle note visualizzabili dall'utente dal db	francescofuligni a...	Done	P1	1	
22	Elimina nota: aggiornamento interfaccia e salvataggio db	francescofuligni	Done	P1	2	
23	Copia nota: aggiornamento interfaccia e salvataggio su db	francescofuligni	Done	P1	1	
24	Ricerca: per parole chiave	francescofuligni	Done	P1	1	
25	Ricerca: predisposizione interfaccia	francescofuligni	Done	P1	1	
26	Panel navigation	francescofuligni	Done	P1	1	
27	Elimina nota: eliminazione dal db	francescofuligni	Done	P1	1	
28	Concorrenza: controllo versione alla richiesta di modifica	MagriniLorenzo	Done	P1	2	
29	Refactor sessione: refactor test coerentemente	mmaurii	Done	P1	1	
30	Refactor sessione: migliorare l'implementazione	mmaurii	Done	P1	1	
31	Creazione di tag: verifica se il tag esiste già	mmaurii	Done	P1	1	
32	Creazione di tag: se non esiste il tag lo si crea	mmaurii	Done	P1	1	
33	Creazione di una nota: Raccolta dei dati sul server	mmaurii	Done	P1	1	
34	Creazione di una nota: Salvataggio della nota a db	mmaurii	Done	P1	2	
35	Creazione di una nota: Creazione form per la raccolta dei dati	mmaurii	Done	P1	1	
36	Registrazione Utente: Creazione form	RobertoZanolli	Done	P1	1	

37	Accesso Utente: creazione interfaccia login	RobertoZanolli	Done	P1	1	
38	Registrazione Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1	1	
39	Registrazione Utente: salvataggio dati utente su DB	RobertoZanolli	Done	P1	1	
40	Lettura nota: anteprima di una nota	MagriniLorenzo	Done	P1	1	
41	Accesso Utente: raccolta e validazione dei dati	RobertoZanolli	Done	P1	1	
42	GitHub Action -> test before PR opening to develop	RobertoZanolli	Done	P1	2	
43	Concorrenza: visualizzazione delle versioni in conflitto	mmaurii	Done	P1	5	
44	RELEASE E CONSEGNA	francescofuligni, ...	Ready	P0	1	
45	Preliminare: studio libreria per persistenza dei dati	RobertoZanolli	Done	P0	2	
46	Filtri: bottone per reset	francescofuligni	Done	P0	1	
47	Refactor Note: mantenimento versionamento	francescofuligni a...	Done	P0	3	
48	Concorrenza: risoluzione bug owner	MagriniLorenzo	Done	P0	1	
49	Servlet: services (Facade) per gestione responsabilità	francescofuligni	Done	P0	2	
50	Refactor Note: aggiunta permessi	francescofuligni a...	Done	P0	1	
51	NoteServlet: pattern Facade per gestione responsabilità	MagriniLorenzo	Done	P0	2	
52	Astrazioni: interfacce per wrappare gli oggetti	francescofuligni	Done	P0	3	