Dipartimento di Ingegneria e Scienza dell'Informazione

Progetto:

EasyLib

Titolo del documento:

Sviluppo

Document Info

Doc. Name	*D2-EasyLib_Sviluppo*	Doc. Number	D3 V1.x
Description	Documento di sviluppo dell'applicazione		

INDICE

Sommario

Scopo del documento

Il presente documento riporta tutte le informazioni necessarie per lo sviluppo di una parte dell'applicazione EasyLib. In particolare, presenta tutti gli artefatti necessari per realizzare i servizi di gestione dei libri e dei prestiti con l'applicazione EasyLib.
Partendo dalla descrizione delle User Stories, il documento prosegue con gli User Flow, e la presentazione delle web APIs del servizio web, proseguendo con l'organizzazione del codice, il modello dati, e ulteriori informazioni su testing. Infine, una sezione dedicata al front-end e una agli aspetti di deployment.

1. User Stories

Nel presente capitolo vengono riportate le user stories individuate del progetto EasyLib partendo dai requisiti individuati nel documento precedente (D1).

User Story 1:
Scelta tra credenziali locali o autenticazione Google per utenti/operatori

Come utente o operatore della biblioteca, voglio scegliere tra il login con credenziali locali o tramite Google in modo da poter selezionare il metodo di accesso che preferisco

Criteri di accettazione:

- Entrambi, utenti e operatori, possono scegliere tra l'inserimento di nome utente e password o il pulsante "Accedi con Google".
- Dopo un login riuscito, il sistema reindirizza l'utente alla dashboard corrispondente al suo ruolo
- Se il login fallisce (con credenziali locali o Google), il sistema fornisce messaggi di errore adeguati.

TASKS – User Story 1:

1. **Integrare entrambe le opzioni di login (locali e Google) nella stessa pagina**
   - Posizionare correttamente sia il form di login con credenziali locali che il pulsante di login con Google.
2. **Configurare il flusso decisionale per il login**
   - Implementare la logica che distingue quale metodo di autenticazione (locale o Google) è stato scelto e avvia il processo corretto.
3. **Gestire il routing dopo il login per entrambi i ruoli (utente e operatore)**
   - Assicurarsi che dopo l'autenticazione l'utente venga reindirizzato alla dashboard corretta in base al suo ruolo (utente o operatore).
4. **Testare entrambe le modalità di accesso**
   - Testare l'autenticazione con credenziali locali per utenti e operatori.
   - Testare l'autenticazione Google per utenti e operatori.
   - Verificare che entrambe le opzioni funzionino correttamente e conducano alla dashboard giusta.
5. **Implementare messaggi d'errore distinti per ciascun metodo**
   - Creare messaggi di errore specifici per problemi relativi al login con credenziali locali o tramite Google.

User Story 2:
Ricerca di libri per titolo, autore, casa editrice o ISBN

Come utente della biblioteca, voglio poter cercare i libri presenti nel catalogo utilizzando uno o più criteri di ricerca (titolo, autore, casa editrice o ISBN), in modo da trovare facilmente i libri che mi interessano

Criteri di accettazione:

- L'utente può inserire uno o più criteri di ricerca nel campo apposito (titolo, autore, casa editrice o ISBN).
- Il sistema restituisce una lista di risultati che corrispondono ai criteri inseriti.
- Ogni risultato deve mostrare dettagli come titolo, autore, anno di pubblicazione, disponibilità e posizione in biblioteca.
- Se non vengono trovati risultati, viene mostrato un messaggio informativo.

TASKS – User Story 2:

1. **Creare il campo di ricerca nella UI**
   - Progettare e sviluppare il campo di ricerca dove gli utenti possono inserire uno o più criteri (titolo, autore, casa editrice, ISBN).
   - Aggiungere placeholder e istruzioni per aiutare gli utenti a capire come utilizzare il campo.
2. **Implementare la logica di ricerca per singoli criteri**
   - Scrivere la logica backend per gestire la ricerca per ciascun criterio (titolo, autore, casa editrice, ISBN) separatamente.
   - Collegare la ricerca con il database del catalogo bibliografico.
3. **Recuperare e visualizzare i risultati della ricercar**
   - Implementare il recupero dei risultati dal database in base ai criteri inseriti dall'utente.
   - Mostrare i dettagli per ogni risultato: titolo del libro, autore, anno di pubblicazione, disponibilità e posizione in biblioteca.
4. **Gestire i casi senza risultati**
   - Implementare la logica per mostrare un messaggio informativo quando non vengono trovati risultati in base ai criteri di ricerca.
5. **Testare la funzionalità di ricerca per singolo criterio**
   - Testare la ricerca per ciascun criterio (titolo, autore, casa editrice, ISBN) per assicurarsi che funzioni correttamente.
   - Verificare che la visualizzazione dei risultati sia corretta e coerente con i dettagli richiesti.

User Story 3:
Ricerca avanzata con combinazione di filtri

Come utente della biblioteca, voglio poter combinare più filtri di ricerca (titolo, autore, casa editrice e ISBN), in modo da affinare i risultati e trovare in modo più preciso il libro che sto cercando.

Criteri di accettazione:

- L'utente può inserire più parametri contemporaneamente (ad esempio, titolo e autore) per restringere i risultati della ricerca.
- Il sistema restituisce una lista filtrata di risultati basati su tutti i parametri inseriti.
- I dettagli del risultato devono includere titolo, autore, anno di pubblicazione, disponibilità e posizione in biblioteca.
- La ricerca deve funzionare anche se alcuni parametri non sono inseriti (ad esempio, solo titolo e autore, senza ISBN).

TASKS – User Story 3:

1. **Modificare il campo di ricerca per supportare la combinazione di criteri**
    - Estendere la funzionalità del campo di ricerca per consentire l'inserimento di più parametri contemporaneamente (ad esempio, titolo e autore).
    - Aggiungere un'interfaccia che permetta agli utenti di combinare filtri in modo semplice e intuitivo.
2. **Implementare la logica di ricerca con più criteri combinati**
    - Sviluppare la logica back end che permetta di combinare più criteri di ricerca (titolo, autore, casa editrice, ISBN) per restringere i risultati.
    - Assicurarsi che i criteri possano essere utilizzati anche se parzialmente inseriti (ad esempio, solo titolo e autore).
3. **Recuperare e visualizzare i risultati della ricerca combinate**
    - Implementare la logica per filtrare i risultati in base a più criteri.
    - Mostrare i risultati correttamente con tutti i dettagli richiesti (titolo, autore, anno di pubblicazione, disponibilità e posizione).
4. **Testare la combinazione di criteri per la ricerca avanzata**
    - Testare la ricerca combinata con più criteri per garantire che i risultati siano corretti.
    - Verificare che la ricerca funzioni correttamente con combinazioni diverse di filtri (ad esempio, titolo + autore, autore + casa editrice, ecc.).
5. **Gestire i casi senza risultati per la ricerca combinate**
    - Implementare un messaggio di errore o informativo quando non ci sono risultati basati sui filtri combinati.
    - Testare la corretta gestione dei casi senza risultati.

2. User Flow

In questa sezione del documento di sviluppo riportiamo gli "user flows" per il ruolo di studente della nostra applicazione. Figura x descrive lo user flow relativo alla gestione dei libri nella nostra applicazione EasyLib. L'utente generico può consultare la lista dei libri presenti nel sistema. Lo studente, dopo essersi loggato, può prendere un libro in prestito.

In figura presentiamo anche la relazione tra le varie azioni disponibili e le features dell'applicazione. La legenda in alto descrive i simboli usati nello user flow.

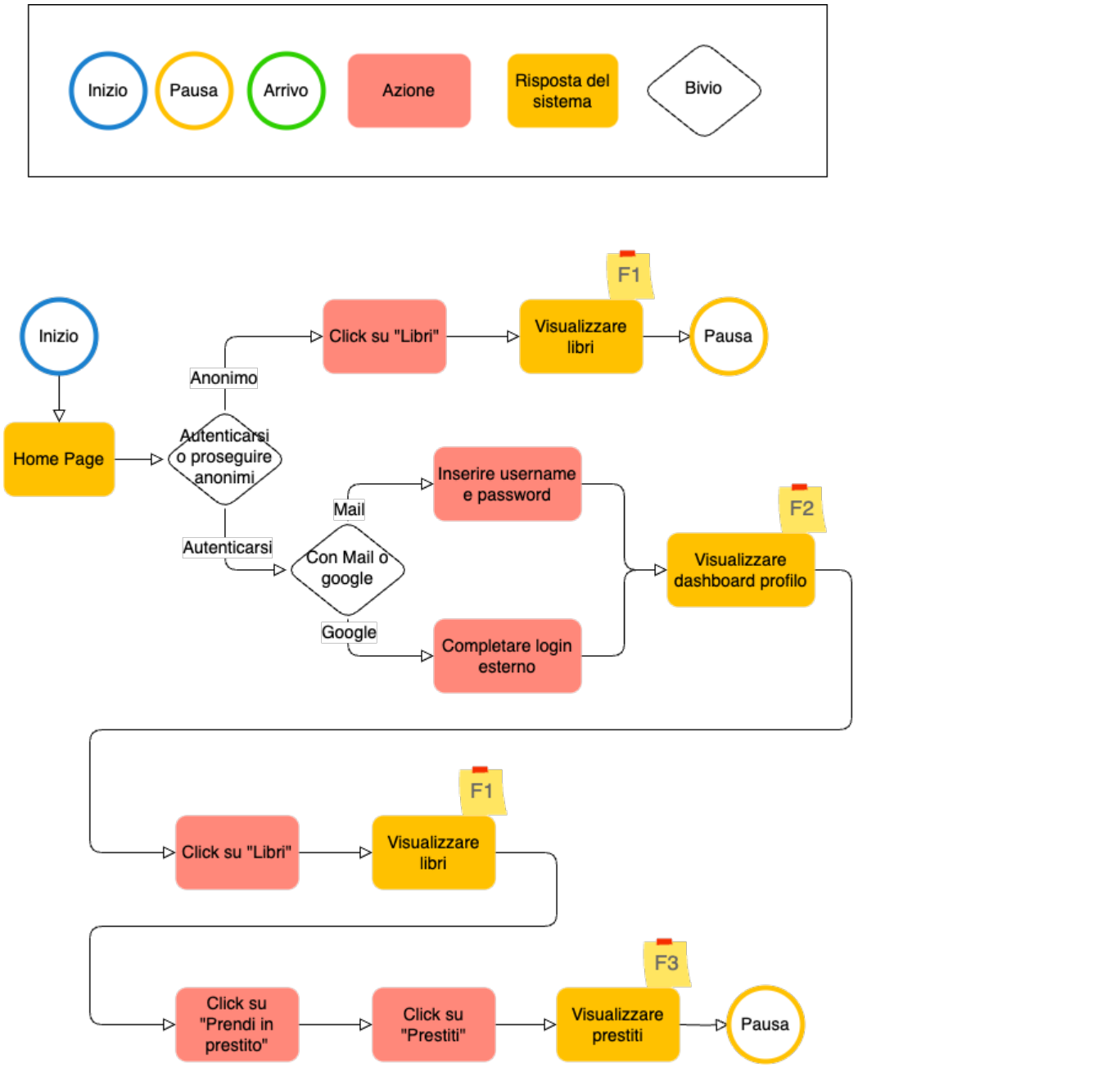

Figura 1 Diagramma esemplificativo! Alcuni elementi potrebbero non corrispondere al resto del progetto EasyLib

3. Web APIs

Le api sono state documentate secondo le specifiche openapi3 e la documentazione è consultabile pubblicamente su swagger/apiary/altro al seguente indirizzo https://easylib.docs.apiary.io/#.

<<<Ulteriori note su scelte di design delle api>>>

La specifica delle API è disponibile nel repository al link https://github.com/unitn-software-engineering/EasyLib/blob/master/oas3.yaml. Il contenuto del file .yaml è qui riportato per esteso:

```yaml
openapi: 3.0.0
info:
  version: '1.0'
  title: "EasyLib OpenAPI 3.0"
  description: API for managing book lendings.
  license:
    name: MIT
servers:
  - url: http://localhost:8000/api/v1
    description: Localhost
paths:
  /students:
    post:
      description: >-
        Creates a new student in the system.
      summary: Register a new student
      requestBody:
        content:
          application/json:
            schema:
              type: object
              required:
                - email
              properties:
                email:
                  type: string
                  description: 'Email address of the student'
      responses:
        '201':
          description: 'User created. Link in the Location header'
          headers:
            'Location':
              schema:
                type: string
              description: Link to the newly created student.
```

```yaml
  /books:
    get:
      description: >-
        Gets the list of books.
        It is possible to show users by their role /users?role={role}
      summary: View all books
      parameters:
        - in: query
          name: role
          schema:
            type: string
            enum: [user, poweruser, admin]
      responses:
        '200':
          description: 'Collection of books'
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Book'
  /booklendings:
    post:
      description: >-
        Creates a new booklending.
      summary: Borrow a book
      responses:
        '201':
          description: 'Booklending created. Link in the Location header'
          headers:
            'Location':
              schema:
                type: string
              description: Link to the newly created booklending.
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Booklending'

components:
  schemas:
    Student:
      type: object
      required:
        - id
        - email
      properties:
        id:
          type: integer
          description: 'ID of the user'
```

```
      email:
        type: string
        description: 'Email address of the user'
  Book:
    type: object
    required:
      - title
      - author
      - ISBN
      - status
    properties:
      title:
        type: string
        description: 'Title of the book'
      author:
        type: string
        description: 'Author of the book'
      ISBN:
        type: string
        description: 'ISBN of the book'
      status:
        type: string
        enum: [available, lended]
        description: 'Tells whether the book is currently available or not'
  Booklending:
    type: object
    required:
    - student
    - book
    properties:
      user:
        type: string
        description: 'Link to the user'
      book:
        type: integer
        description: 'Link to the book'
```

4. Implementation

L'applicazione è stata sviluppata utilizzando le seguenti tecnologie: NodeJS e VueJS, per la gestione dei dati abbiamo utilizzato MongoDB. Lo stack tecnologico è stato motivato dal materiale fornito nel corso / conoscenze pregresse / altro ….

4.1. Repository Organization

Il codice del progetto (https://github.com/unitn-software-engineering/EasyLib) è organizzato secondo la seguente struttura:

- /src — cartella file sorgenti
  - /api — endpoint per le risorse studenti e prestiti
  - /middleware — middleware di autenticazione
  - /models — modelli dati mongoose
  - app.js — applicazione Express.js
- /frontend — codice per la parte del front-end
- /doc
  - openapi3.yaml — documentazione api
- package.json — file di configurazione del progetto npm
- .gitignore — configurazione repository git
- .env.example — esempio di file di configurazione variabili d'ambiente

4.2. Branching strategy e organizzazione del lavoro

Lo sviluppo ha visto una collaborazione attiva tra i membri del gruppo. Ci siamo divisi il lavoro in questo modo: … In totale abbiamo eseguito più di xx commit distribuiti tra i membri in questo modo …. Lo sbilanciamento tra il numero di commit è dovuto a errori/codice di librerie esterne erroneamente committato/alto numero di piccoli commit vs numero di commit contenuto ma commit molto consistenti…

A livello di repository abbiamo optato per una strategia di branching GitFlow Workflow. Durante lo sviluppo di nuove funzionalità abbiamo lavorato sui seguenti branch, sempre derivandoli dal branch develop:

- "books": il branch è stato usato per sviluppare il modello dati e le api relative alla risorsa libri;
- "home_page": il branch è stato dedicato allo sviluppo della prima pagina di benvenuto del sito;
- …

4.3. Dependencies

Il progetto npm si basa sui seguenti moduli esterni:

- Cors — Per il supporto alle chiamate cors
- Express — Framework web per il backend
- Google-auth-library — Supporto al login google
- Jsonwebtoken — Autenticazione JWT
- Mongoose — Libreria per interfacciarsi con mongoDB

E le seguenti dipendenze di sviluppo:

- Dotenv — Gestione variabili d'ambiente in ambiente dev da file.env
- Jest — Testing
- Supertest — Testing endpoint express

4.4. Database

Per la gestione dei dati utili all'applicazione abbiamo definito due principali strutture dati tramite l'utilizzo di mongoose.

Students: …

Booklendings: …

4.5. Testing

Le api e il relativo codice presentano una test-suite che consente di verificarne il corretto funzionamento. I test sono utilizzata all'interno della configurazione di CI/CD.

L'implementazione dei test è organizzata in file .test.js che affiancano i vari file dell'implementazione / I test sono implementati tutti in un'unica cartella …

Dei xxx casi di test che sono stati definiti, yyy sono stati implementati.

N. Test Case	Descrizione Test Case	Test Data	Precondizioni	Dipendenze	Risultato Atteso	Risultato riscontrato	Note
1	Creazione di un'account in modo corretto	<username> non vuota <password> rispettosa delle politiche di sicurezza	<username> mai inserita prima nel sistema	---	Viene creata l'account specificato. Il sistema risponde con <msg OK>		
1.1	Creazione di un'account specificando uno username già esistente	<username> già inserita nel sistema	<username> già inserita nel sistema	Questo caso di test deve essere fatto dopo il caso di testo numero 1 specificando la stessa <username>	Viene mostrato un messaggio di errore <error msg user exists>, l'account non viene creata ed il sistema mostra alcuni username disponibili da utilizzare		
2	Creazione di un'account non specificando lo username	<username> Vuota	---	---	Viene mostrato <errore msg empty user> e l'account non viene creato		
3	Creazione di account violando le politiche di sicurezza (password banale, conferma password errata, …)	<password> non rispettosa delle politiche di sicurezza	---	---	Viene mostrato un messaggio di errore e l'account non viene creata		

5. FrontEnd

Il FrontEnd fornisce le funzionalità di visualizzazione, inserimento e cancellazione dei dati dell'applicazione EasyLib. In particolare, l'applicazione è composta da una Home Page, da una pagina per la gestione dei libri e una pagina per la gestione dei prestiti.

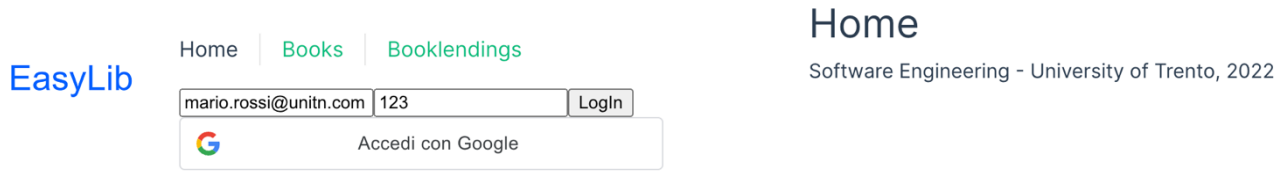

Figura 2 Home Page

In Figura 3 è mostrata l'home page dell'applicazione. Da qui l'utente può autenticarsi e navigare nelle altre pagine dell'app.

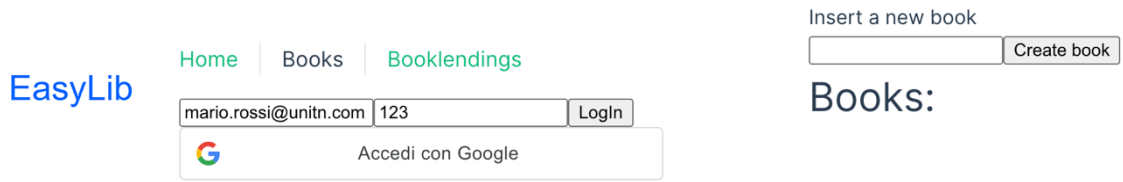

Figura 4 Schermata libreria

In Figura 5 è mostrata lista dei libri prenotabili dall'utente. Inoltre è possibile inserire nuovi libri specificandone il titolo.

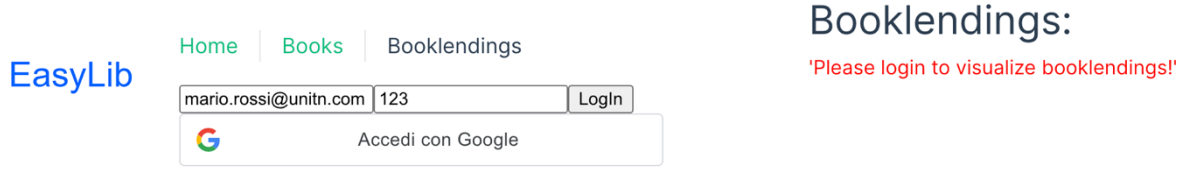

Figura 6 Schermata libri in prestito

In Figura 7 è visibile la lista dei libri attualmente in prestito dall'utente. Come utente non loggato ricevo un messaggio di errore.

6. Deployment

Un'istanza del backend è ospitata sulla piattaforma render.com ed è disponibile al link https://easy-lib.onrender.com/. <<<altri eventuali link di deploy>>> Il frontend è deployato separatamente al link https://easylibvue.onrender.com/.

La configurazione CI/CD basata sulle GitHub Actions esegue automaticamente il deploy del branch main quando vengono superati con successo tutti i test. <<<altri dettagli>>>

Per accedere al deploy dell'applicazione è possibile utilizzare i seguenti account:

- Utente studente
    - Username: mario.rossi@unitn.it Password: 123
- Altra tipologia di utente
    - …

Per problemi con il deploy contattare <<< mail amministratore / mail di tutti i membri >>>