CineList

SUPERVISORI

Prof.ssa Genoveffa Tortora

Dott. Luigi Di Biasi

CANDIDATI

Carmine Iemmino
Matricola: 0522501697
Angelo Nazzaro
Matricola: 0522501774

Basi di Dati II, Corso di Laurea Magistrale
Dipartimento di Informatica
Università degli Studi di Salerno
Fisciano, Italia

Anno Accademico 2023-2024

Indice

CAPITOLO 1

Analisi dei Requisiti

1.1 Raccolta delle Specifiche della Realtà d'Interesse

1.1.1 Descrizione

Si vuole progettare una base di dati per il supporto ad un'applicazione che permetta di visualizzare informazioni sui films e salvare la lista dei film guardati e/o da guardare. Ogni film ha come attributi principali titolo, media voto, data di rilascio, introiti, budget, registi che l'hanno diretto, attori che hanno recitato, generi a cui appartiene e trama. Gli utenti sono identificati tramite e-mail e hanno una password e un nickname (casuale se non scelto).

1.1.2 Realtà d'Interesse

La realtà che andiamo a rappresentare riguarda la gestione di un'applicazione che permetta il facile accesso ad informazioni di film e il tracciamento dei film guardati e da guardare. Un'applicazione del genere permette di fare una scelta più informata prima di scegliere il film da guardare e poi di esprimere una propria recensione dopo averlo guardato. Gli utenti che accedono all'applicazione possono:

- Ricercare film per titolo, regista, attore, genere, anno di rilascio;
- Visualizzare i film più popolari;
- Visualizzare i film con media voto più alta;
- Visualizzare la lista dei film guardati;
- Visualizzare la lista dei film da guardare;
- Aggiungere/Rimuovere un film (d)alla lista dei film da guardare;
- Aggiungere/Rimuovere un film (d)alla lista dei film guardati;
- Recensire un film;
- Aggiornare la recensione di un film;
- Visualizzare la lista dei film preferiti;
- Aggiungere/Rimuovere un film (d)alla lista dei film preferiti;

Per tal motivo, proproniamo lo sviluppo di "CineList".

1.1.3 Glossario dei Termini

Termine	**Significato**
Utente/User	Persona iscritta all'applicazione.
Film/Movie	Principale oggetto di interesse dell'applicazione; opera d'arte visiva.
Recensione/Review	Recensione scritta da parte di un utente dell'applicazione con un voto.
Attori/Actors	Persone che recitano in un film.
Registi/Directors	Persone che dirigono un film.
Troupe	Raggruppamento di attori e registi. Tutto il personale che partecipa alla realizzazione di un film.

Tabella 1.1. Glossario

CAPITOLO 2

Progettazione della Base di Dati

2.1 Analisi e Pre-processing del Dataset

Il dataset scelto, disponibile al seguente link, si incentra sullo scenario cinematografico. Nello specifico, contiene informazioni sui film organizzate come di seguito:

- ID;
- Votazione Media;
- Numero di Voti;
- Stato (Rilasciato, In produzione, Altro);
- Data di Rilascio;
- Incasso;
- Durata;
- Budget;
- IMDb Id;
- Linguaggio Originale;
- Titolo Originale;
- Descrizione Breve;
- Popolarità;
- Tagline;
- Generi;
- Compagnie di Produzione;
- Nazioni di Produzione;
- Linguaggi Parlati;
- Cast;

- Direttori;
- Direttori della Fotografia;
- Scrittori;
- Produttori;
- Compositori Musicali;

Al fine di consentire una corretta integrazione dei dati, questi sono stati sottoposti a una fase di pre-processing, sfruttando una strategia ETL. Nello specifico, sono state rimosse le colonne riguardo gli scrittori, produttori, compositori musicali e direttori della fotografia in quanto non ritenute di particolare interesse da parte dell'utilizzatore medio dell'applicazione.

Inoltre, sono state rimosse tutte le tuple con valori null, così da considerare solo i film aventi informazioni complete. Sono stati aggiunti i campi "poster" e "anno di rilascio". L'anno di rilascio è stato ricavato da "data di rilascio" ed è stato aggiunto così da non dover effettuare ogni volta formattazioni per alcune viste che necessitano solo dell'anno di rilascio. Inoltre, per motivi di sono considerati solo i film usciti negli ultimi 30 anni.

La fase di pre-processing ha ridotto il numero di tuple da circa 900.000 a circa 40.000.

2.2 Vincoli non Esprimibili

Di seguito, sono elencati i vincoli che non possono essere espressi nello schema:

Identificativo	**Descrizione**
RV1	Il campo *Type* in **Troupe** deve essere uno tra "actor" o "director".
RV2	Il valore del campo *Vote* in **Review** deve essere compreso tra 0 e 5.
RV3	Ogni *username* in **User** deve essere unico.
RV4	In **User**, per ogni record in *movies_list*, se il campo *favourite* è impostato su "true", allora il campo *watched* deve essere "true". Tuttavia, il contrario non è necessariamente vero.

Tabella 2.1. Elenco dei vincoli non esprimibili nello schema

2.3 Regole di Derivazione

Di seguito, sono elencate le regole di derivazione:

2.4 Tavola dei Volumi

Di seguito, è mostrata la tavola dei volumi. I volumi relativi alle Troupe e alle Reviews sono stati inferiti in seguito a una breve analisi statistica, la quale ha rivelato che **in media** sono presenti circa 30 attori per film e che ogni film ha circa 10 review.

Identificativo	Campo Derivato	Calcolo
RD1	*vote_ count* in Movie	Derivato dal numero di recensioni che contengono lo stesso Movie_ID.
RD2	*vote_ average* in Movie	Calcolato dalla somma dei voti delle recensioni per il film, diviso per il *vote_ count*.
RD3	*watched_ count* in Movie	Numero di volte in cui il film compare nelle liste degli utenti con *watched* impostato su "true".
RD4	*added_ count* in Movie	Numero di volte in cui il film compare nelle liste degli utenti con *watched* impostato su "false".

Tabella 2.2. Regole di derivazione

Collezione	Volume
Movie	40.000
Troupe	1.200.000
Review	400.000
User	1.000

Tabella 2.3. Tavola dei Volumi

2.5 Tavola delle Operazioni

Di seguito, è mostrata la tavola delle operazioni:

Identificativo	Operazione	Tipo	Frequenza
Op1	Selezione film per titolo/regista/attore	Read	700/giorno
Op2	Selezione film per genere	Read	200/giorno
Op3	Selezione film per anno di rilascio	Read	100/giorno
Op4	Ordinamento film per popolarità (decrescente)	Read	200/giorno
Op5	Ordinamento film per voto medio (decrescente)	Read	250/giorno
Op6	Aggiunta di un film alla lista utente come "da guardare"	Write	300/giorno
Op7	Aggiunta di un film alla lista utente come "guardato"	Write	400/giorno
Op8	Aggiornamento di un film della lista utente da "da guardare" a "guardato"	Update	150/giorno
Op9	Aggiornamento di un film della lista utente da "guardato" a "da guardare"	Update	50/giorno
Op10	Selezione film da guardare	Read	500/giorno
Op11	Selezione film guardati	Read	200/giorno
Op12	Rimozione di un film dalla lista utente	Delete	100/giorno
Op13	Aggiunta di una recensione da parte di un utente	Write	250/giorno
Op14	Modifica di un recensione da parte di un utente	Update	10/settimana
Op15	Aggiunta di un film alla lista dei preferiti di un utente	Write	70/giorno
Op16	Rimozione di un film dalla lista dei preferiti di un utente	Update	30/settimana

Tabella 2.4. Tavola delle Operazioni

2.6 Document Schema

Di seguito è riportata l'analisi delle ridondanze effettuata per la progettazione del document schema finale. In particolare, sono indicati i pattern adottati e le relative motivazioni. Nello specifico, abbiamo impiegato i seguenti pattern:

- **Approximation Pattern**
- **Embedded Document Pattern**;
- **Extended Reference Pattern**;
- **Subset Pattern**;
- References Document Pattern;

Movie In questa collezione, i generi, le compagnie e le nazioni di produzione, e i linguaggi parlati sono stati inseriti come array di stringhe, seguendo il **Embedded Document Pattern**. Si è scelto questo pattern perché creare una collezione separata per ogni campo avrebbe richiesto l'esecuzione di più lookup per ottenere tutti i dati del film, aumentando significativamente il costo computazionale.

Gli attori e i registi sono stati inseriti tramite l'**Extended Reference Pattern**. Sono stati inclusi solo i campi più frequentemente accessibili (nome e immagine di profilo); gli altri dati possono essere recuperati tramite lookup nella collezione Troupe quando necessario. Inoltre, le review sono state inserite adottando il **Subset Pattern** mantenendo un array delle cinque recensioni più recenti. Sebbene ciò aumenti la complessità durante l'inserimento di una nuova recensione, poiché è necessario aggiornare sia la collezione delle recensioni sia l'array delle recensioni recenti nella collezione movies, questa scelta velocizza l'operazione di lettura in quanto quest'ultima è maggiore rispetto a quella di scrittura. Infine, è stato utilizzato l'**Approximation Pattern** per la gestione dei campi "vote_average", "vote_count", "watched_count", "added_count": esso consiste nel non aggiornare continuamente i valori dei campi citati, ma solo una volta raggiunta una certa soglia, fornendo un informazione meno precisa ma evitando numerose operazioni di scrittura/aggiornamento. Nel nostro caso abbiamo impostato una soglia di 10: il vote_average, il vote_count, watched_count e added_count vengono aggiornati nel database solo dopo essere aumentati di 10 unità.

Troupe In questa collezione, tramite l'**Extended Reference Pattern**, sono stati inseriti i film in cui è comparso l'attore/regista. Solamente i campi essenziali sono salvati come titolo, poster, data di rilascio, generi, durata e voto medio. Ciò permette di recuperare velocemente i film in cui è comparso l'attore/regista, senza necessità di lookup costosi.

Review In questa collezione, l'**Extended Reference Pattern** viene utilizzato per includere l'username dell'utente che ha scritto la recensione. Questo consente di recuperare rapidamente l'username da mostrare insieme alla recensione. Inoltre, anziché mantenere una lista di ID di recensioni nella collezione Movie, si è deciso di aggiungere il campo "movie_id" alle recensioni. Questo segue una variante del References Document Pattern, ideale quando si ha un numero indefinito di documenti sul lato "a uno" della relazione

User Al fine di recuperare velocemente le informazioni riguardanti i film salvati dall'utente come "guardati" o "da guardare", questi sono stati inseriti tramite l'**Extended Reference Pattern**, salvando solo i campi relativi al titolo e al poster.

In Figura 2.1 è mostrato il document schema adottato.

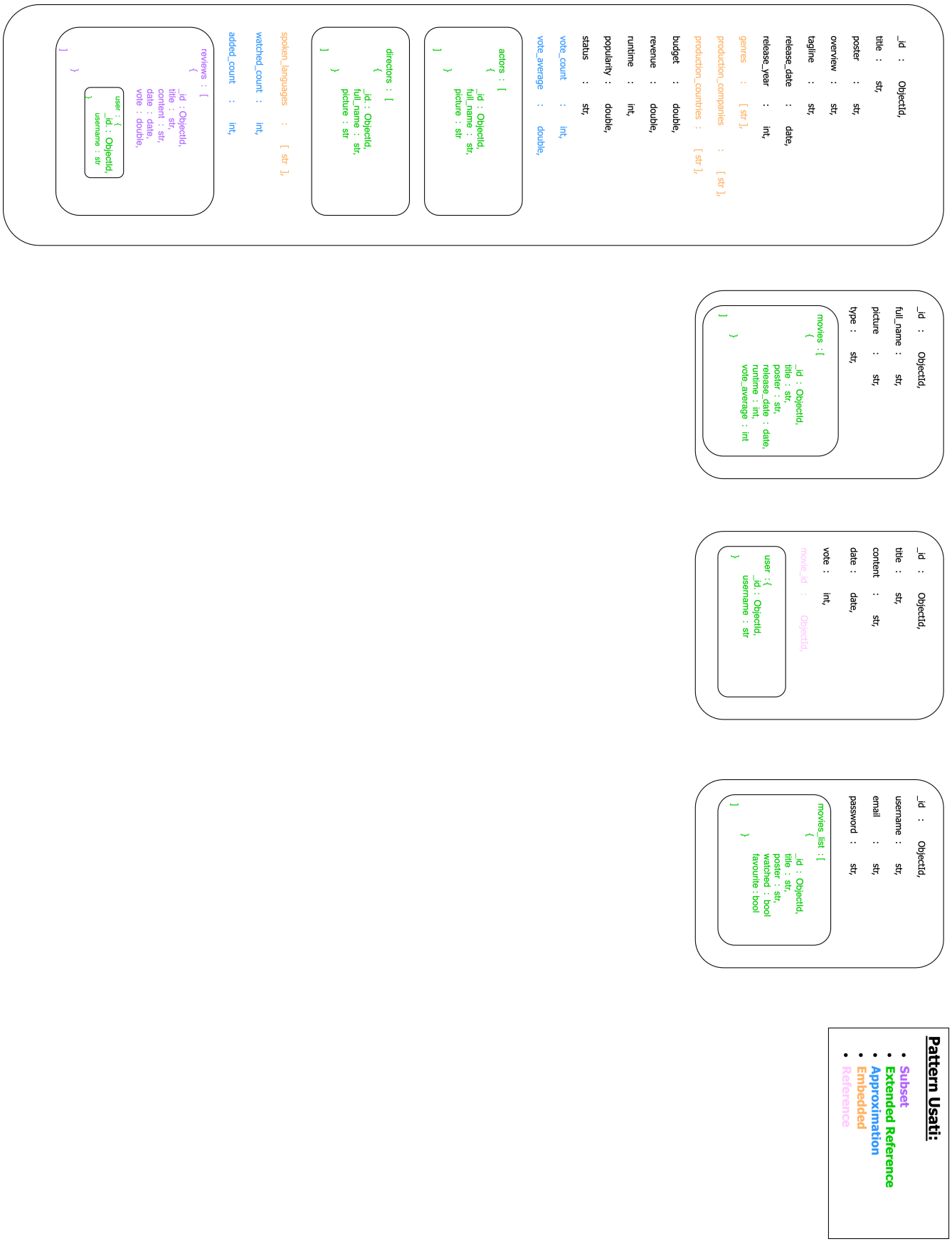

Figura 2.1. Document Schema.

CAPITOLO 3

Implementazione dell'Applicativo

3.1 Tecnologie Impiegate

Per la realizzazione dell'applicativo proposto, sono state utilizzate le seguenti tecnologie:

- **MongoDB**: scelto come database NoSQL per la sua popolarità e le elevate prestazioni, consente una gestione flessibile dei dati grazie alla struttura a documenti.
- **Flask**: scelto come framework per la realizzazione del backend grazie alla sua semplicità d'uso, leggerezza e facilità di implementazione. È altamente estensibile e ben supportato dalla comunità.
- **PyMongo**: impiegato come middleware per l'interazione tra MongoDB e Flask, consente di eseguire operazioni CRUD (Create, Read, Update, Delete) sui dati.
- **Pandas**: utilizzato per le operazioni di pre-processing del dataset, come descritto in **??**. Permette di manipolare e analizzare i dati in modo efficiente.
- **Swift**: impiegato per la realizzazione del frontend dell'applicazione, garantendo una user experience fluida e reattiva su dispositivi iOS.

3.2 Architettura

3.2.1 Backend

Il backend dell'applicativo è stato realizzato utilizzando Flask, un microframework per Python che permette di creare rapidamente applicazioni web robuste. Flask è stato scelto per la sua semplicità e leggerezza, che consentono una rapida implementazione e prototipazione. L'integrazione con il database MongoDB è stata gestita tramite PyMongo, una libreria che fornisce un'interfaccia per interagire con MongoDB da Python. Questo permette di eseguire operazioni di lettura e scrittura sui dati in modo semplice ed efficiente. Il backend fornisce principalmente tre API, dedicate rispettivamente alle operazioni sui film, sugli utenti e sulla troupe. Queste API sono descritte in dettaglio in Tabella 2.4.

3.2.2 Frontend

Il frontend è stato sviluppato utilizzando Swift, il linguaggio di programmazione creato da Apple per lo sviluppo di applicazioni iOS. Swift offre performance elevate e una sintassi moderna, facilitando lo sviluppo di interfacce utente interattive e performanti. L'applicazione è strutturata in diverse viste, ciascuna delle quali rappresenta una parte dell'esperienza utente: le principali sono la MyMovieListView

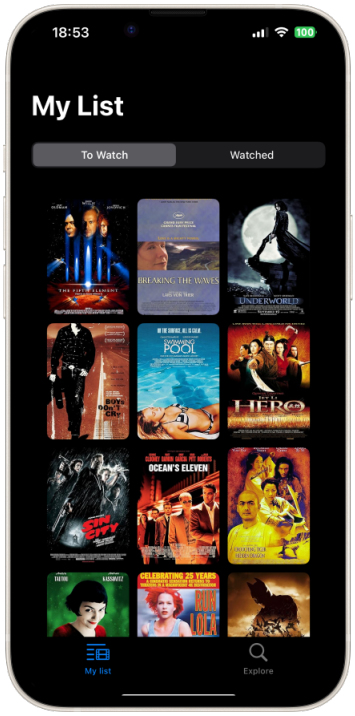

Figura 3.1. Lista dei "miei" film.

in cui l'utente può visualizzare i propri film da guardare, guardati e preferiti e la ExploreView, in cui è possibile ricercare nuovi titoli, ordinati per popolarità, voto medio, data di uscita, etc.

Nel resto del documento sono presenti anteprime dell'applicazione.

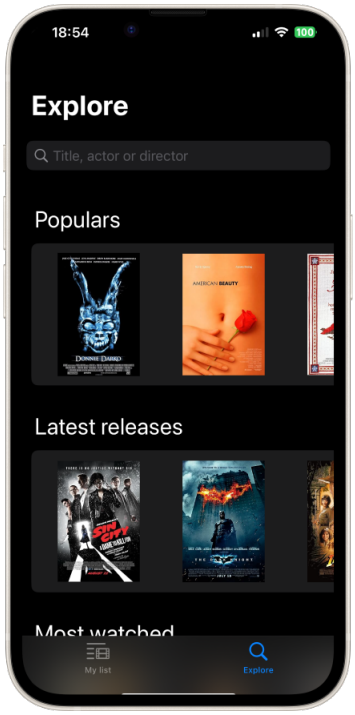

Figura 3.2. Home page.

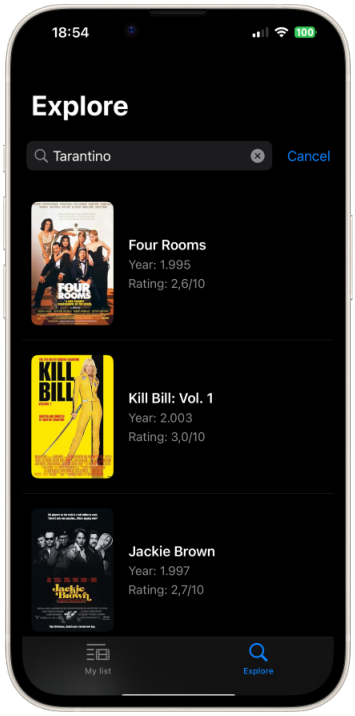

Figura 3.3. Home Page: Ricerca.

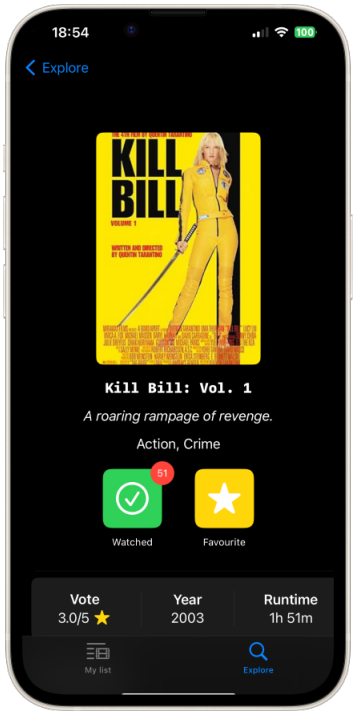

Figura 3.4. Pagina singolo film.

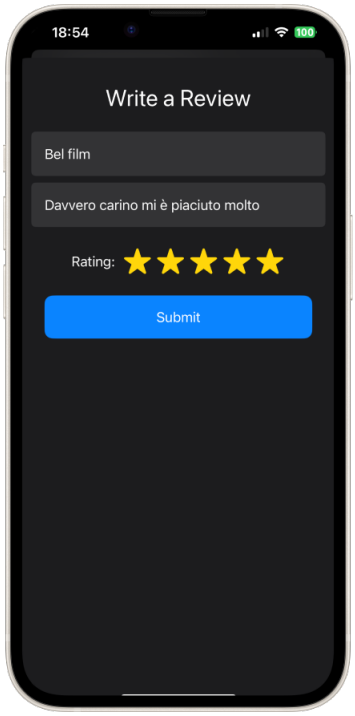

Figura 3.5. Recensione.

LaXidiaN

A LaTeXtemplate inspired by Obisdian.

This document was written using LaXidiaN, a sophisticated LaTeX template inspired by Obisidian. If you have any feedback you´d like to share, be it a complaint or a new feature you´d like to see, please open an issue or a pull request on our github repository.

LaXidiaN repository