ENTREGA PROYECTO #2

PARCHE LECTOR

Presentado por:
Nicolas Arciniegas
Juan Jose Alvarez Lozano
Julian Dario Colmenares Saenz
Julian Santiago Becerra Pulido
Sebastian Castañeda Garcia

Profesor:
Carlos Andres Sierra Virguez

Sabado 8 de Noviembre

Universidad Nacional de Colombia
Departamento de Ingeniería de sistemas
2025

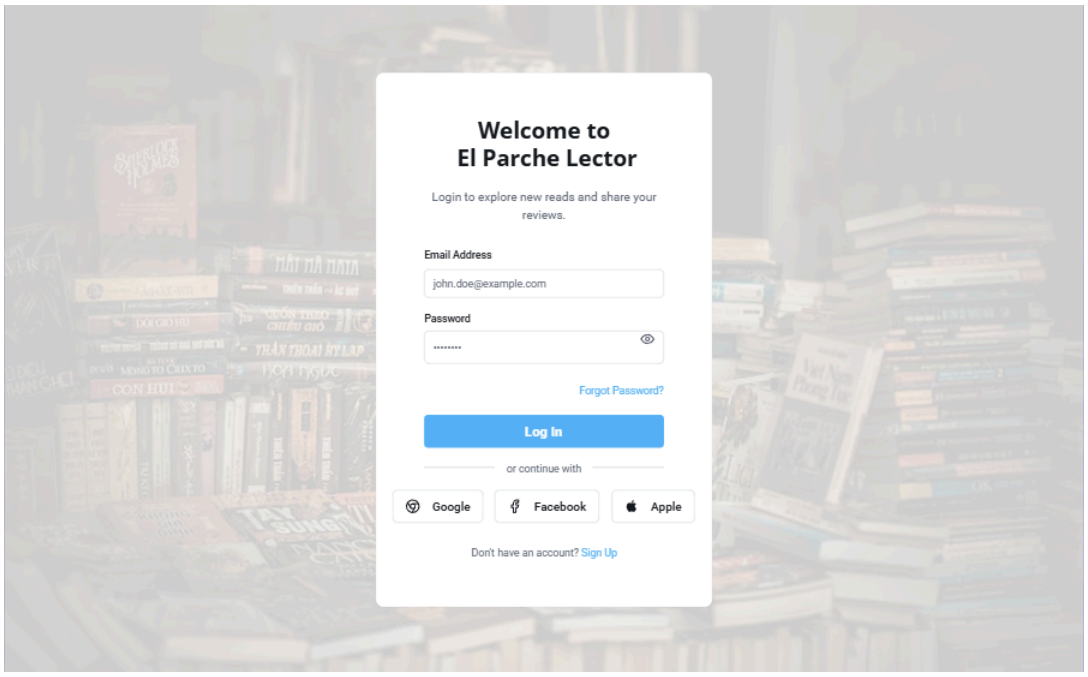
1. CRC Cards

Clase	Responsabilidades	Colaboradores
User	- Register and authenticate users- Edit profile (bio, photo, favorite genres)- Follow and unfollow other users- Manage their reading lists and reviews- View their feed and personal statistics	`Authentication, Profile, ReadingList, Review, Statistics, Book`
Authentication	- Register new users (email, Google, Apple)- Validate credentials- Generate and manage JWT tokens- Recover passwords	`User`
Profile	- Store personal information and preferences- Show recent activity- Configure privacy and photo	`User, Statistics, Review, ReadingList`
Book	- Store bibliographic data (title, author, year, ISBN, synopsis)- Show average rating and reviews- Facilitate search and filtering	`Author, Review, ReadingList`
Author	- Store name and biography- Associate their published books- Allow users to follow authors	`Book, User`
Review	- Allow creating, editing, and deleting reviews- Save text, rating, and date- Associate reviews with users and books- Calculate the book's average rating	`User, Book`
ReadingList	- Create, rename, or delete custom lists- Add or remove books- Classify books by status ("Reading," "Read," "Want to Read")	`User, Book`
Statistics	- Calculate metrics (books read, pages read, top genres)- Automatically update when user activity changes- Generate progress visualizations	`User, Book, ReadingList`
Feed	- Show activity of followed users (reviews, new lists, ratings)- Order publications chronologically- Allow interactions (likes, comments)	`User, Review, ReadingList`

SearchService	- Search for books, authors, or users- Filter results by genre, year, rating, etc.- Connect with external APIs (e.g., Google Books API)	`Book, Author, User`

2. Mockups

A. Login



Welcome to
El Parche Lector

Login to explore new reads and share your reviews.

Email Address

john.doe@example.com

Password

••••••••

Forgot Password?

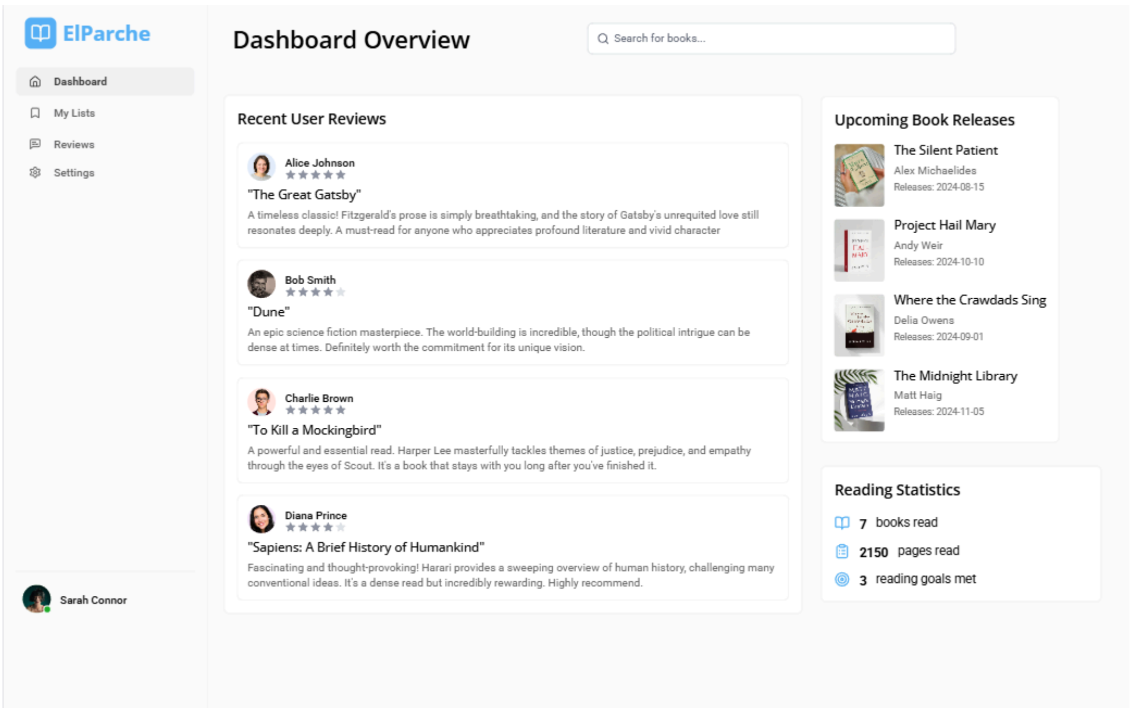
Log In

or continue with

Google Facebook Apple

Don't have an account? Sign Up

B. Dashboard:



ElParche

Dashboard
My Lists
Reviews
Settings

Sarah Connor

Dashboard Overview

Search for books...

Recent User Reviews

Alice Johnson ★★★★★
"The Great Gatsby"
A timeless classic! Fitzgerald's prose is simply breathtaking, and the story of Gatsby's unrequited love still resonates deeply. A must-read for anyone who appreciates profound literature and vivid character

Bob Smith ★★★★★
"Dune"
An epic science fiction masterpiece. The world-building is incredible, though the political intrigue can be dense at times. Definitely worth the commitment for its unique vision.

Charlie Brown ★★★★★
"To Kill a Mockingbird"
A powerful and essential read. Harper Lee masterfully tackles themes of justice, prejudice, and empathy through the eyes of Scout. It's a book that stays with you long after you've finished it.

Diana Prince ★★★★★
"Sapiens: A Brief History of Humankind"
Fascinating and thought-provoking! Harari provides a sweeping overview of human history, challenging many conventional ideas. It's a dense read but incredibly rewarding. Highly recommend.

Upcoming Book Releases

The Silent Patient
Alex Michaelides
Releases: 2024-08-15

Project Hail Mary
Andy Weir
Releases: 2024-10-10

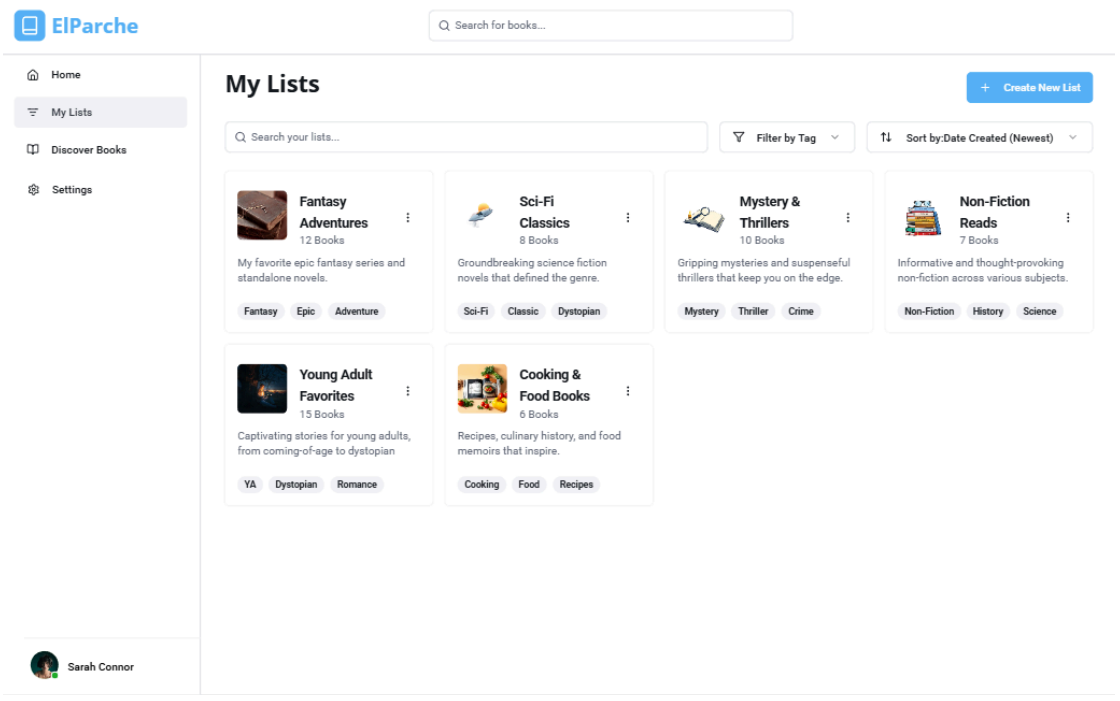
Where the Crawdads Sing
Delia Owens
Releases: 2024-09-01

The Midnight Library
Matt Haig
Releases: 2024-11-05

Reading Statistics

7 books read
2150 pages read
3 reading goals met

C. My lists:


ElParche

Search for books...

Home
My Lists
Discover Books
Settings

My Lists

+ Create New List

Search your lists...

Filter by Tag

Sort by:Date Created (Newest)

Fantasy Adventures
12 Books
My favorite epic fantasy series and standalone novels.
Fantasy · Epic · Adventure

Sci-Fi Classics
8 Books
Groundbreaking science fiction novels that defined the genre.
Sci-Fi · Classic · Dystopian

Mystery & Thrillers
10 Books
Gripping mysteries and suspenseful thrillers that keep you on the edge.
Mystery · Thriller · Crime

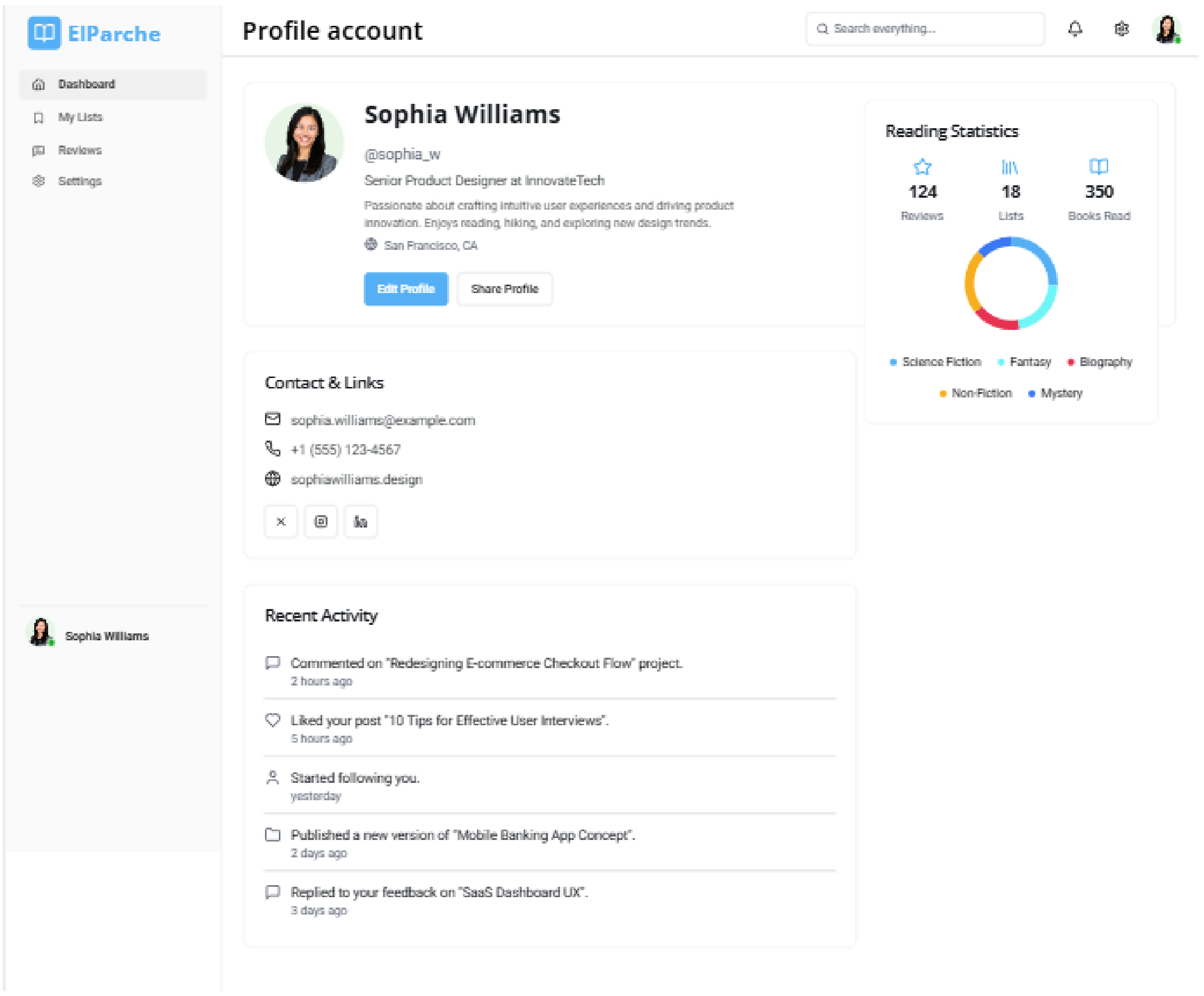
Non-Fiction Reads
7 Books
Informative and thought-provoking non-fiction across various subjects.
Non-Fiction · History · Science

Young Adult Favorites
15 Books
Captivating stories for young adults, from coming-of-age to dystopian
YA · Dystopian · Romance

Cooking & Food Books
6 Books
Recipes, culinary history, and food memoirs that inspire.
Cooking · Food · Recipes

Sarah Connor

D. Profile:


ElParche

Dashboard
My Lists
Reviews
Settings

Profile account

Search everything...

Sophia Williams
@sophia_w
Senior Product Designer at InnovateTech
Passionate about crafting intuitive user experiences and driving product innovation. Enjoys reading, hiking, and exploring new design trends.
San Francisco, CA

Edit Profile · Share Profile

Reading Statistics

124	18	350
Reviews	Lists	Books Read

● Science Fiction ● Fantasy ● Biography ● Non-Fiction ● Mystery

Contact & Links
sophia.williams@example.com
+1 (555) 123-4567
sophiawilliams.design

Recent Activity

Commented on "Redesigning E-commerce Checkout Flow" project.
2 hours ago

Liked your post "10 Tips for Effective User Interviews".
5 hours ago

Started following you.
yesterday

Published a new version of "Mobile Banking App Concept".
2 days ago

Replied to your feedback on "SaaS Dashboard UX".
3 days ago

Sophia Williams

3. Business Model Processes

1. User Registration and Onboarding

Objective: Help users create an account, verify their identity, complete their profile, and familiarize themselves with the platform to increase retention from the start. **Trigger:** The user accesses the app and chooses to create an account or log in. **Participants:** User - Frontend - Authentication Service - Email/OAuth Provider - User Database - Recommendation System - Profile System.

Main Steps:

1. User selects authentication method (Email, Google, or Apple); the system validates availability.
2. **Credential Validation:** For email, verify format and confirm via email; for OAuth, validate external token; create account in DB.
3. **Initial Profile Configuration:** Enter username, bio (optional), upload photo (optional), select favorite genres (minimum 3).
4. **Follow Suggestions:** System suggests popular users or users with similar tastes; user follows or skips.
5. **Feature Tour:** Show main features, explain review creation, and direct to the search for the first book.
6. Generate JWT and return to the frontend; save session/last access.

Exceptions/Compensations:

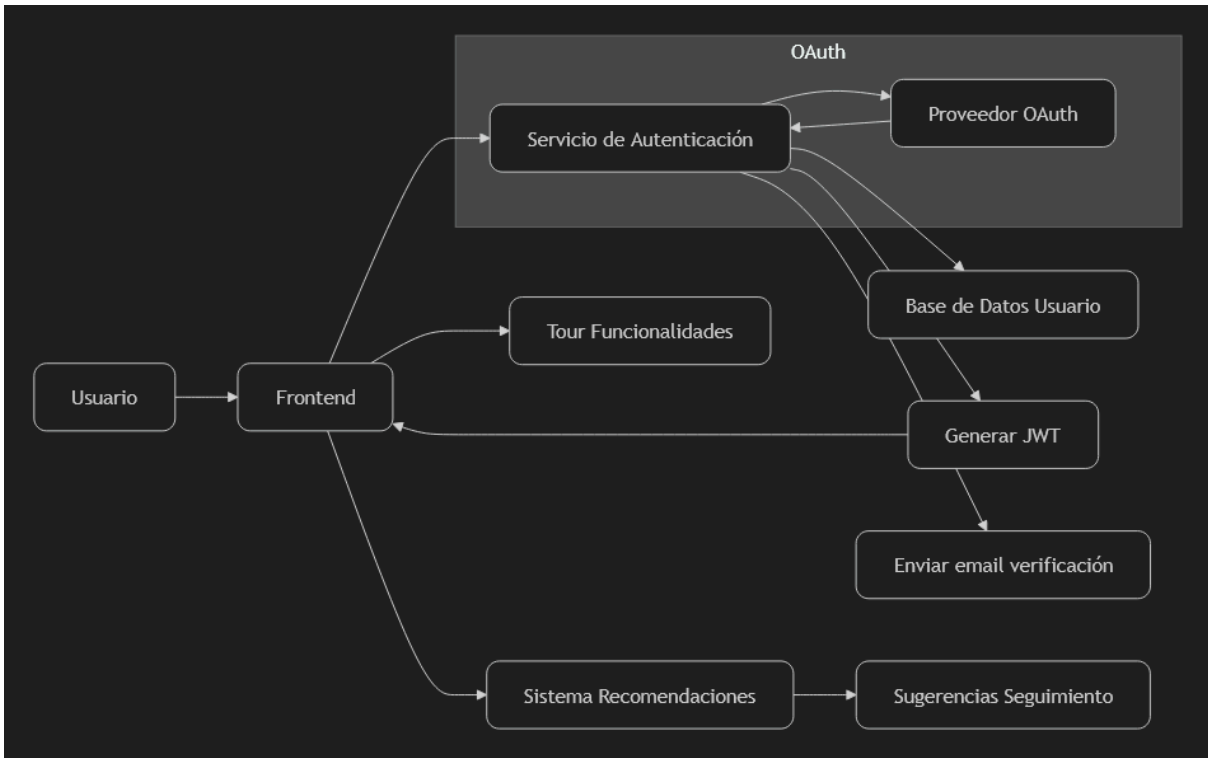
- Credential error → Notify and block after 5 attempts.
- Existing email → Offer password recovery.
- OAuth provider fails → Fallback to email registration.

Inputs/Outputs:

- **Input:** Email, password, OAuth data, initial profile data.
- **Output:** JWT, user created/validated, verification email, initial profile configured, tour completed.

KPIs: Registration conversion rate (users who complete onboarding), email verification rate, failed login attempts, average authentication time (<2s), % of users who follow at least 1 suggestion.

Diagram:



2. User Profile Management

Objective: Allow management of bio, photo, favorite genres, privacy, and data update to personalize the experience. **Trigger:** User requests to view/edit their profile. **Participants:** User - Frontend - Profile Service - Storage (images) - User Database - Statistics System.

Main Steps:

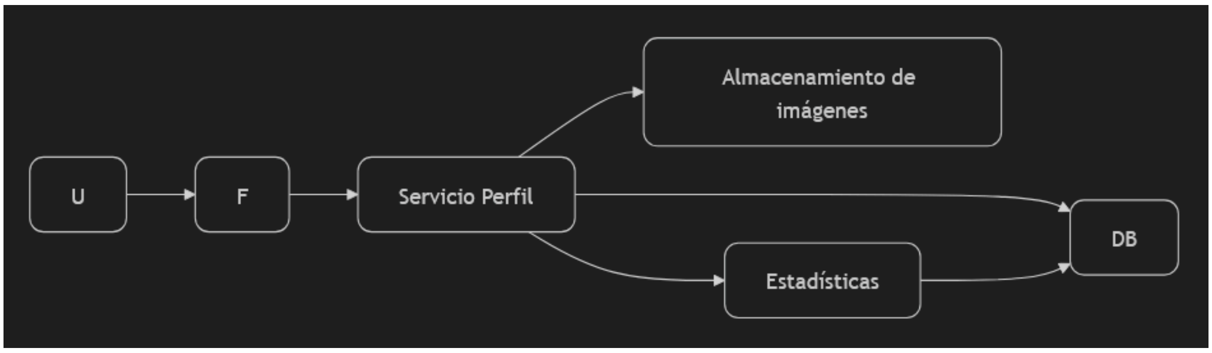
1. User opens the Profile section; frontend requests data from the Profile Service.
2. User edits fields (bio, genres, photo) and submits.
3. Profile Service validates, saves to DB, and stores the photo in CDN.
4. Update public statistics and notify Feed if privacy changes.
5. Display public statistics, recent reviews, and lists in the view.

Exceptions/Compensations: Upload failed → Retry or revert local changes.

Inputs/Outputs:

- **Input:** Edited data (bio, photo, genres).
- **Output:** Updated profile, generated visualizations.

KPIs: Profile completion ratio (>70%), average editing time (<3s for visible changes), profile photo usage.



3. Book and Author Management

Objective: Keep the catalog updated with metadata, ratings, and indexing for efficient searches. **Trigger:** New book identified via search or manual entry. **Participants:** User/Editor - Book Service - Author Service - Book Database - External API (Google Books/OpenLibrary) - Search System.

Main Steps:

1. Search/ingestion from external API (by title/author/ISBN) or manual entry.
2. Book Service normalizes metadata (title, author, synopsis, cover).
3. Associate or create Author entity; add book to DB if new.
4. Calculate/update the average rating with existing reviews.
5. Index book for Search Service.

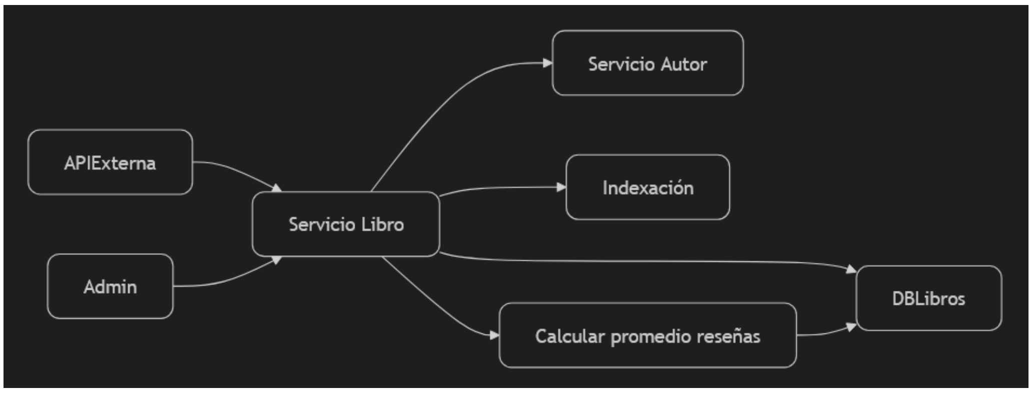
Exceptions/Compensations: Incomplete data → Mark as 'pending' and request enrichment.

Inputs/Outputs:

- **Input:** Search term or manual data.
- **Output:** Book/Author added, normalized metadata, indexed.

KPIs: Catalog coverage, ingestion time (<30s), % of books with complete metadata.

Diagram:



4. Review Lifecycle (Create-Edit-Delete)

Objective: Allow creating, publishing, editing, and deleting reviews, updating book metrics, and distributing to feeds. **Trigger:** User decides to review a book. **Participants:** User - Frontend - Review Service - Book Service - Moderation System - Feed System - Notification System - Statistics System.

Main Steps:

1. **Book Search and Selection:** Search by title/author/ISBN; consult local DB or external API; add if new.
2. **Review Creation:** Assign rating (1-5), write text, mark spoilers, indicate reading date.
3. **Status Update:** Mark as "Read," add to ReadingList, update statistics.
4. **Validation and Publication:** Validate length/rating; save to DB; recalculate book average.
5. **Distribution:** Notify followers; appear in feed, profile, and book page.
6. Allow editing/deletion with moderation if applicable.

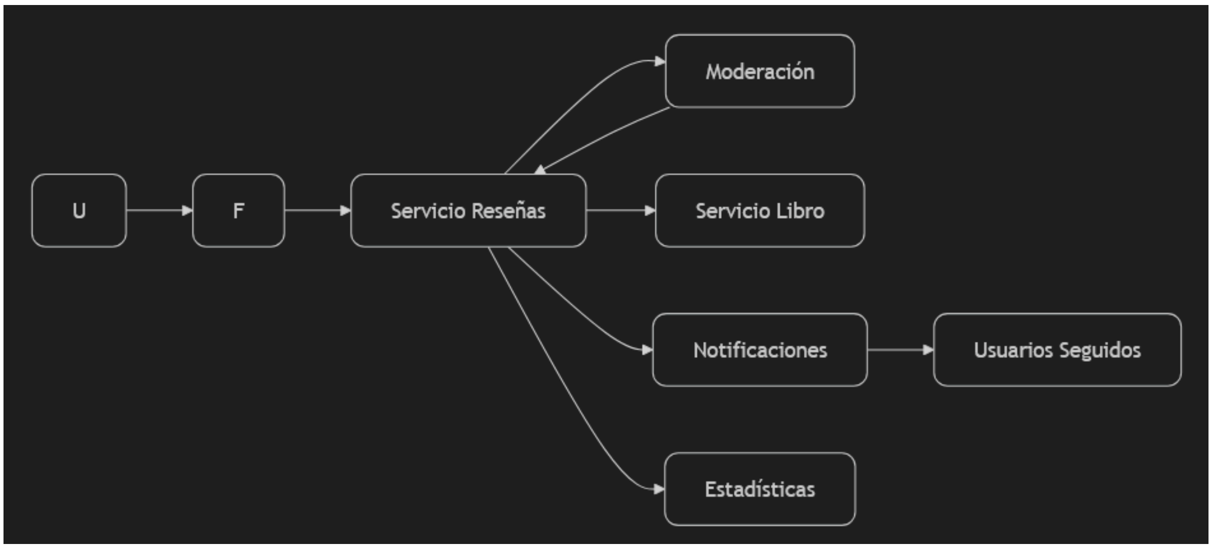
Exceptions/Compensations: Reported review → Mark for moderation, retract if inappropriate; automatic moderation fails → Manual queue.

Inputs/Outputs:

- **Input:** Review data (text, rating, book).
- **Output:** Review published, metrics updated, notifications sent.

KPIs: Reviews per user/month, editing rate, time to publication (<10s if automatic), % manually moderated, % of users who publish the first review.

Diagram:



5. Reading List Management

Objective: Organize books into automatic or custom lists, update statuses, and share for social interaction. **Trigger:** User decides to organize books or change the status. **Participants:** User - Frontend - ReadingList Service - Book Service - Statistics System - Feed System.

Main Steps:

1. **Type Selection:** Automatic ("Reading," "Read," "Want to Read") or custom.
2. **Custom Creation:** Enter name, description, privacy; create empty list.
3. **Book Addition:** Search/select book, choose list; save changes.
4. **Status Update:** Remove from other lists if it's a status list; update statistics.
5. **Visibility:** If public, appear on profile; generate shareable URL; optional activity on feed.

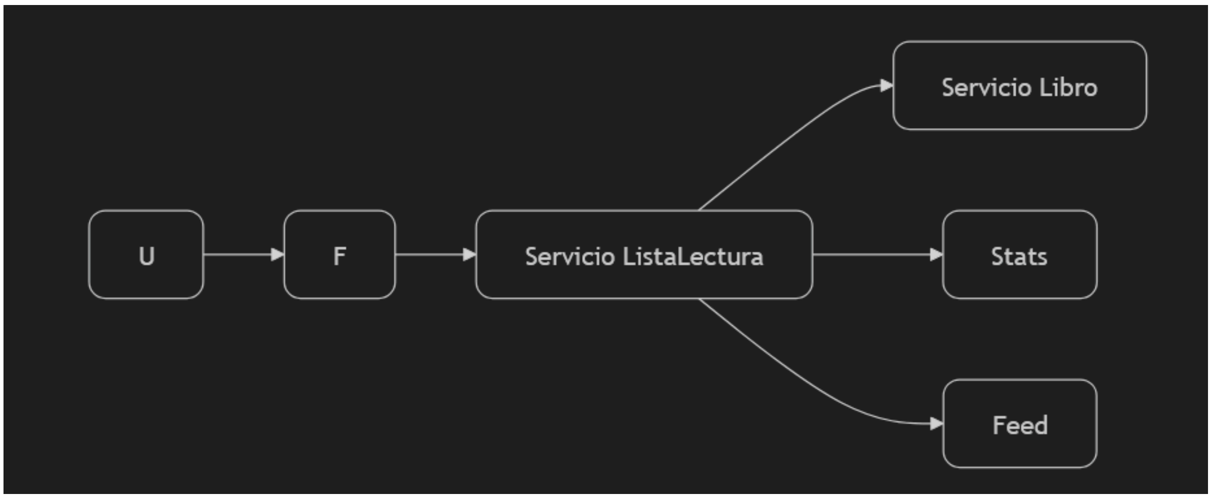
Exceptions/Compensations: Book not found → Offer import or suggestion.

Inputs/Outputs:

- **Input:** List name, selected books.
- **Output:** Updated list, statuses changed, shareable.

KPIs: Lists created per user, books per list, interactions on shared lists, visible changes <3s.

Diagram:



6. Social Feed and Interactions

Objective: Aggregate and present the activity of followed users chronologically, allowing interactions to increase engagement. **Trigger:** User accesses the feed or publishes activity. **Participants:** User - Frontend - Feed Service - Review Service - ReadingList Service - Interactions Service - Notification System.

Main Steps:

1. **Feed Request:** Identify followed users; user refreshes.
2. **Aggregation:** Consult recent activity (reviews, lists, books read, new followers).
3. **Ordering:** By date (most recent first); limit 20 per load; remove duplicates.
4. **Presentation:** Display with user, type, summary, options (like, comment).
5. **Interaction:** Record likes/comments; update metrics.
6. **Additional Load:** Infinite scroll for more batches.

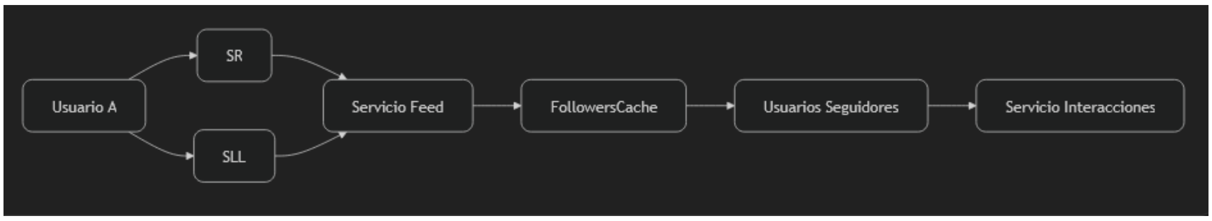
Exceptions/Compensations: Spam → Mark for moderation.

Inputs/Outputs:

- **Input:** Access to the feed.
- **Output:** Generated feed, registered interactions.

KPIs: Feed engagement (likes/comments), time on platform after viewing feed, delivery to followers <5s (soft), average response per publication.

Diagram:



7. Search and Book Discovery

Objective: Find books based on criteria, integrating external APIs for discovery. **Trigger:** User initiates search. **Participants:** User - Frontend - Search Service - Index (Elasticsearch) - External API - Book System - Review System - ReadingList System.

Main Steps:

1. **Input:** Term (title/author/ISBN), filters (genre, year, rating) or explore trending.
2. **Processing:** Consult local DB; if <10 results, external API; combine/order.
3. **Presentation:** Display with cover, title, average rating, reviews, user's list status.
4. **Refinement:** Adjust filters/order (relevance, popularity); update in real-time.
5. **Selection:** Show detailed page; actions (add list, review, view reviews).

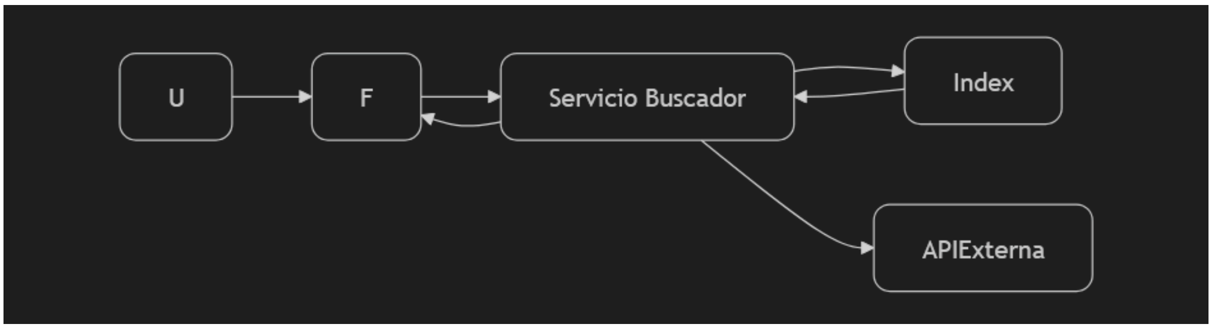
Exceptions/Compensations: Desynchronized index → Fallback to DB or API.

Inputs/Outputs:

- **Input:** Query/filters.
- **Output:** Paginated results, available actions.

KPIs: Search success rate, response time (TTFB <200ms in 95%), click-through rate, books searched vs. added.

Diagram:



8. Statistics Calculation and Visualizations

Objective: Calculate personal metrics and generate visualizations to motivate users. **Trigger:** Change in activity (book read, review, list). **Participants:** Event/Queue Service - Statistics Service - Data Warehouse - Frontend - ReadingList System - Review System - Book System.

Main Steps:

1. **Event Detection:** Change in lists/reviews.
2. **Collection:** Count books by status, sum pages, group by genre/year.
3. **Calculation:** Total read, pages, monthly average, % of genres, top authors read, average rating.
4. **Generation:** Graphs (bars by month, genre pie chart, progress line, annual comparisons).
5. **Storage:** Save to DB; update dashboard; generate annual summary.

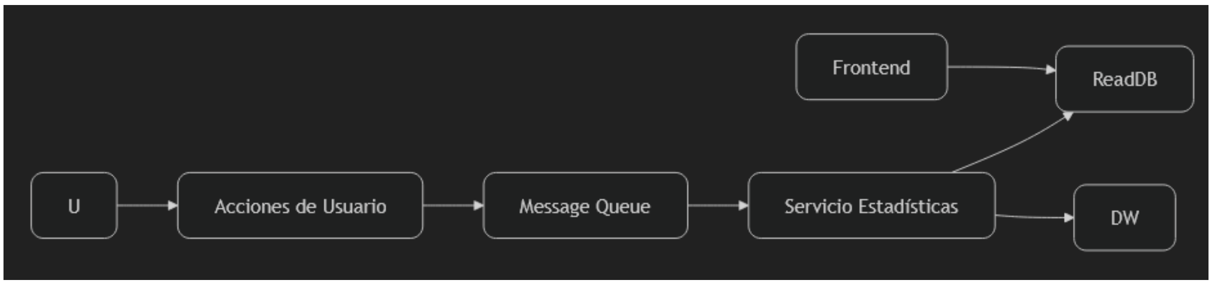
Exceptions/Compensations: Duplicate event → Deduplication by ID.

Inputs/Outputs:

- **Input:** Activity events.
- **Output:** Calculated statistics, visualizations.

KPIs: Statistics accuracy, update latency (<1 min), number of visualizations per user.

Diagram:



9. Notifications and Recovery

Objective: Deliver relevant notifications and manage access recovery. **Trigger:** Event (new follower, reply, verification). **Participants:** Notifications Service - Email/Push Provider - User - Authentication Service.

Main Steps:

1. Event generates message in queue.
2. Decide channel (email/push/in-app) based on preferences.
3. Deliver and register.
4. **In Recovery:** Generate temporary token, send it; validate when changing password.

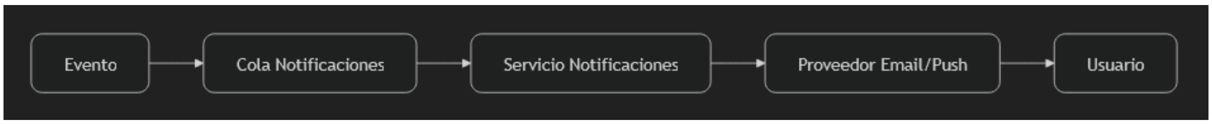
Exceptions/Compensations: Failed delivery → Retry.

Inputs/Outputs:

- **Input:** Event.
- **Output:** Notification delivered.

KPIs: Delivery rate, open rate, delivery time (push <10s, email <1 min).

Diagrama:



10. Administration, Moderation, and Reporting

Objective: Moderate content, manage the catalog, and review operational metrics. **Trigger:** Reported incident or administrative routine. **Participants:** Admin - Admin Tool - Services (Book, Reviews, Users) - Logs/Alerts.

Main Steps:

1. Review reports/moderation queue.

2. Take action (delete, suspend, correct).
3. Record audit and notify.
4. Generate reports (KPIs, incidents).

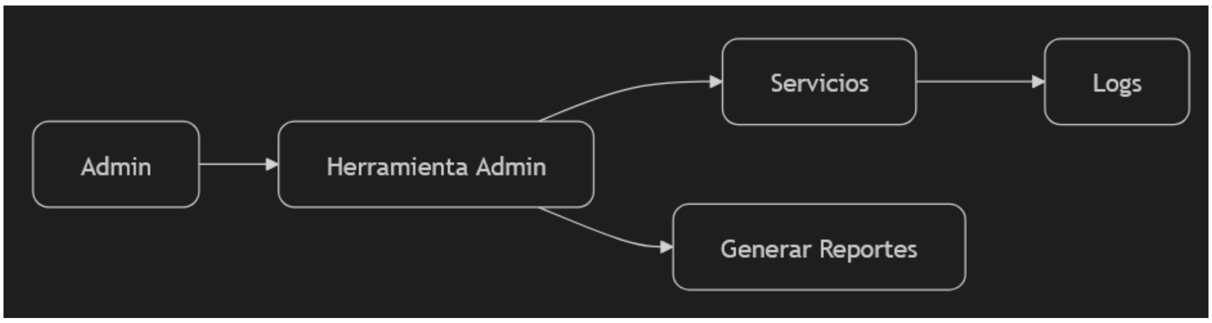
Exceptions/Compensations: Inaccurate automatic moderation → Manual review.

Inputs/Outputs:

- **Input:** Reports/incidents.
- **Output:** Actions taken, reports generated.

KPIs: Resolution time, number of incidents, automatic moderation accuracy.

Diagram:



4. Reference Plan: Owners, Triggers, and SLAs

Process	**Owner**	**Trigger**	**Key SLA**
Registration/Authentication	Auth Team	User creates/logs in	Authentication <2s, block after 5 failures
Profile	Product/UX	User edits	Visible changes <3s
Books/Authors	Catalog Team	Ingestion/API	Indexing <30s
Reviews	Community	User publishes	Publication <10s (automatic)
Lists	Product	User manages	Visible change <3s
Feed	Eng/Infra	User event	Delivery to followers <5s (soft)
Search	Search Team	User query	TTFB <200ms in 95%
Statistics	Data	Activity event	Update <1 min
Notifications	Ops	Event	Push <10s, email <1 min
Administration	Ops/Community	Incident/routine	Average resolution <24h

5. Architecture Diagram

This diagram defines a decoupled three-layer architecture (Frontend, Backend, Data) hosted on AWS, ideal for an MVP that needs to scale.

A. Presentation Layer (Edge)

This layer is responsible for delivering the user-facing application quickly and securely.

- **User (Client):** The end-user who accesses the "Reader Patch" application through a web browser.
- **Amazon S3 (Frontend):** An S3 bucket is configured to host the static files of the **React with TypeScript** application (HTML, CSS, JavaScript).
- **Amazon CloudFront (CDN):** Acts as the content delivery network.
  - When a user accesses the site, CloudFront delivers a cached copy of the React application from the nearest geographic location, reducing latency.
  - It also provides native HTTPS (SSL/TLS).

B. API and Application Layer (Backend)

All business logic resides here. It is designed as a **modular monolith**, meaning it is a single application (NestJS) but is internally organized following the "Services" defined in the business processes (such as **Authentication Service**, **Profile Service**, **Review Service**, etc.).

- **Amazon API Gateway:** It is the single entry point for all API requests from the frontend.
  - It manages routing, can apply security (like WAF), control traffic (rate limiting), and handle request authorization before they reach your business logic.
- **AWS Elastic Beanstalk (with NestJS):** For an MVP, Elastic Beanstalk (EB) is ideal.
  - You simply upload your **NestJS** code, and EB automatically manages server provisioning (EC2), load balancing, and autoscaling.
  - The single NestJS application will contain all the modules (controllers, services) that implement the business logic.
- **Amazon SQS (Message Queue):** Used to decouple heavy tasks or tasks that don't need to be instantaneous.
  - For example, when a user publishes a review, the API can save the review to the DB (synchronous) and then send a message to SQS for the **Statistics Service** to recalculate the book's average (asynchronous), ensuring the API responds quickly.

C. Data and Storage Layer

This layer manages the persistence of the application's data.

- **Amazon RDS (PostgreSQL):** An instance of Amazon Relational Database Service (RDS) with the **PostgreSQL** engine is used.

    ○ RDS is a managed service that handles backups, patching, and replication, allowing focus on the **data schema design** rather than database administration.

- **Amazon S3 (Data):** A second S3 bucket, separate from the frontend, is used to store user-generated files and content, such as profile photos or book covers, as mentioned in the "Profile Management" process.

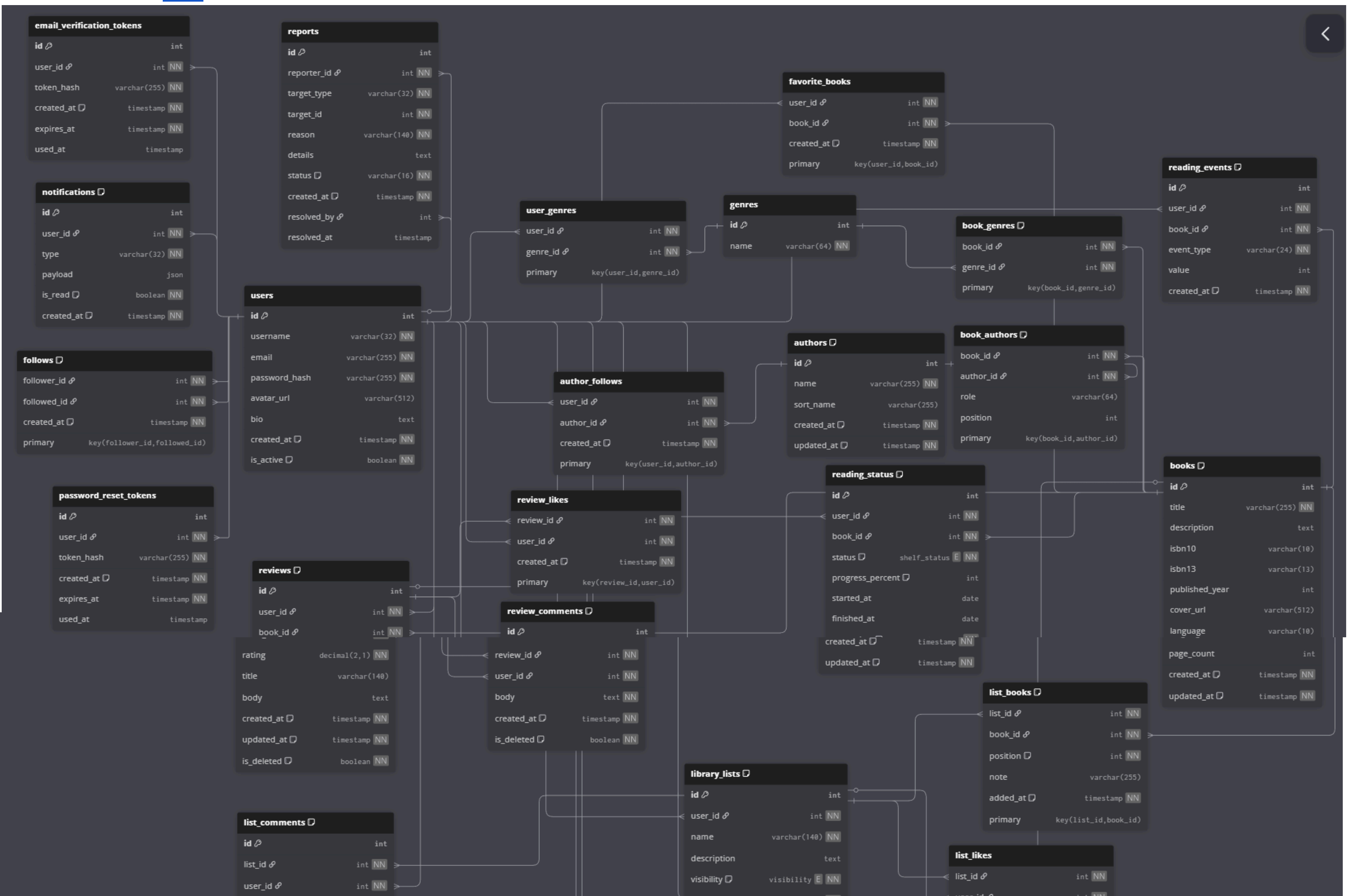
D. External Services and Notifications

These are components your application depends on to function.

- **Amazon SES (Simple Email Service):** Used by the backend to send transactional emails, such as account verification and password recovery.
- **OAuth Providers (External):** The **Authentication Service** will communicate with Google and Apple to validate social login tokens.
- **Book APIs (External):** The **Book Service** and **Search Service** will consult external APIs like Google Books to enrich the book catalog.

6. Relational Database Model

https://dbdiagram.io/d/RelationalDatabaseModel-El-Parche-Lector-690c199a6735e111707a20dd



email_verification_tokens
id int
user_id int NN
token_hash varchar(255) NN
created_at timestamp NN
expires_at timestamp NN
used_at timestamp
reports
id int
reporter_id int NN
target_type varchar(32) NN
target_id int NN
reason varchar(140) NN
details text
status varchar(16) NN
created_at timestamp NN
resolved_by int
resolved_at timestamp
favorite_books
user_id int NN
book_id int NN
created_at timestamp NN
primary key(user_id,book_id)
reading_events
id int
user_id int NN
book_id int NN
event_type varchar(24) NN
value int
created_at timestamp NN
notifications
id int
user_id int NN
type varchar(32) NN
payload json
is_read boolean NN
created_at timestamp NN
user_genres
user_id int NN
genre_id int NN
primary key(user_id,genre_id)
genres
id int
name varchar(64) NN
book_genres
book_id int NN
genre_id int NN
primary key(book_id,genre_id)
users
id int
username varchar(32) NN
email varchar(255) NN
password_hash varchar(255) NN
avatar_url varchar(512)
bio text
created_at timestamp NN
is_active boolean NN
follows
follower_id int NN
followed_id int NN
created_at timestamp NN
primary key(follower_id,followed_id)
authors
id int
name varchar(255) NN
sort_name varchar(255)
created_at timestamp NN
updated_at timestamp NN
book_authors
book_id int NN
author_id int NN
role varchar(64)
position int
primary key(book_id,author_id)
author_follows
user_id int NN
author_id int NN
created_at timestamp NN
primary key(user_id,author_id)
reading_status
id int
user_id int NN
book_id int NN
status shelf_status E NN
progress_percent int
started_at date
finished_at date
created_at timestamp NN
updated_at timestamp NN
books
id int
title varchar(255) NN
description text
isbn10 varchar(10)
isbn13 varchar(13)
published_year int
cover_url varchar(512)
language varchar(10)
page_count int
created_at timestamp NN
updated_at timestamp NN
password_reset_tokens
id int
user_id int NN
token_hash varchar(255) NN
created_at timestamp NN
expires_at timestamp NN
used_at timestamp
review_likes
review_id int NN
user_id int NN
created_at timestamp NN
primary key(review_id,user_id)
reviews
id int
user_id int NN
book_id int NN
rating decimal(2,1) NN
title varchar(140)
body text
created_at timestamp NN
updated_at timestamp NN
is_deleted boolean NN
review_comments
id int
review_id int NN
user_id int NN
body text NN
created_at timestamp NN
is_deleted boolean NN
list_books
list_id int NN
book_id int NN
position int NN
note varchar(255)
added_at timestamp NN
primary key(list_id,book_id)
library_lists
id int
user_id int NN
name varchar(140) NN
description text
visibility visibility E NN
list_comments
id int
list_id int NN
user_id int NN
list_likes
list_id int NN