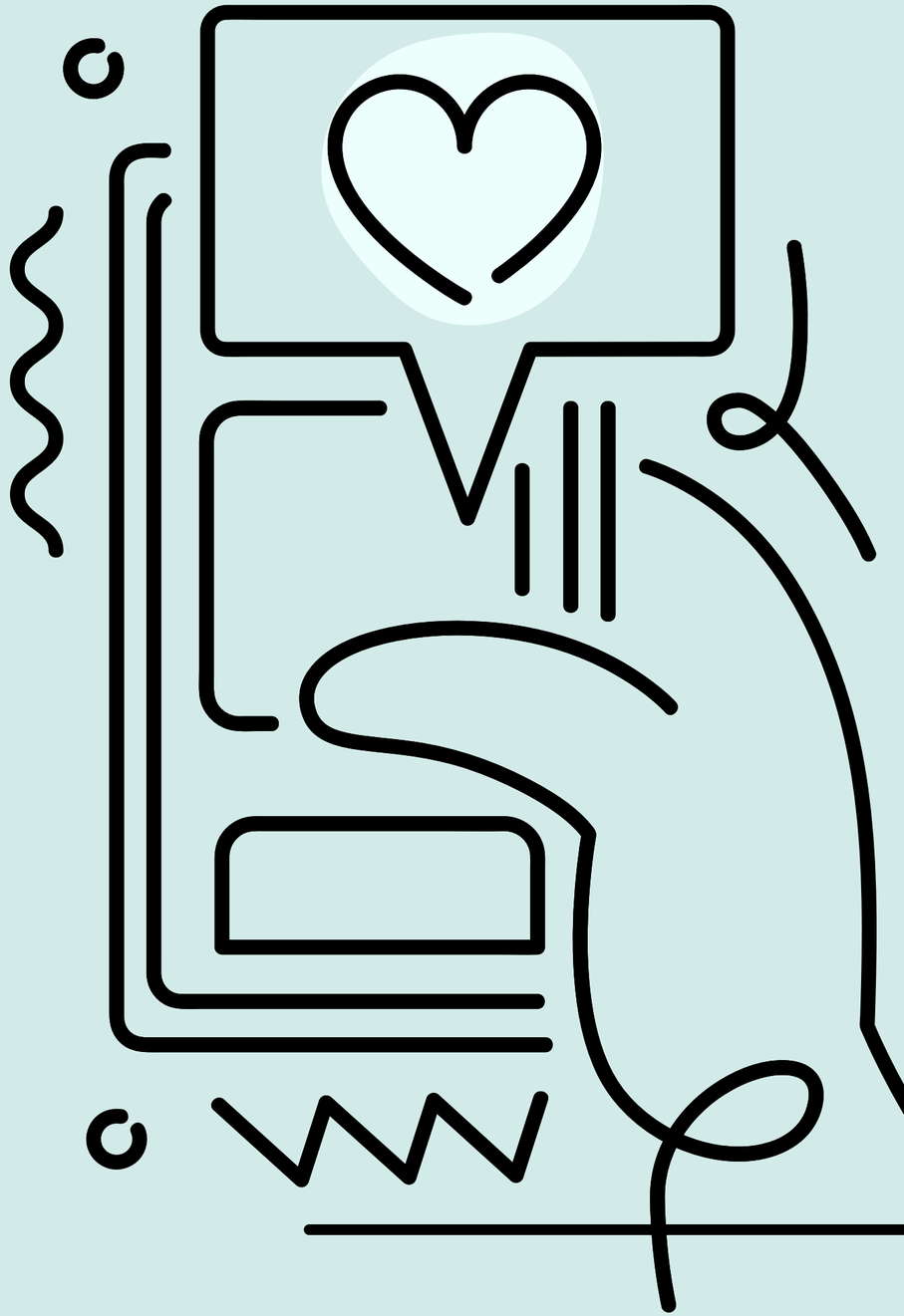
UD SHARE, A USER AND PUBLICATION MANAGEMENT PLATFORM WITH HYBRID DATABASE ARCHITECTURE



Date: July 09, 2025

Prepared by: Johan S. Ebratt, Paola B. Cuellar

PROBLEM STATEMENT

In a world where data is becoming more and more varied and there is a need to store everything, the opportunity arises to use different types of databases depending on the case. We want to understand and demonstrate how combining relational and non-relational databases can facilitate the resolution of information management and processing problems.

KEY FUNCTIONALITIES AND REQUIREMENTS

Non-Functional Requirements:

- 99.9% uptime with regional redundancy
- Auto-scaling infrastructure
- Query response times under 100ms

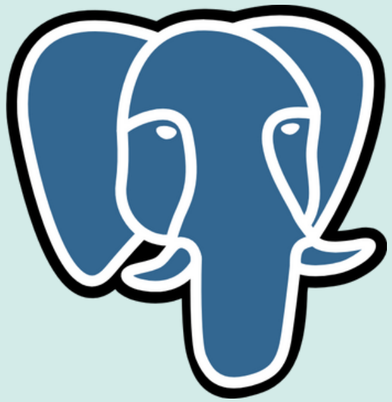
Key User Stories:

- Create, edit, and manage post visibility
- Follow/unfollow users and interact with posts
- Report content and moderate users
- Access analytics (for premium users)

TECHNOLOGIES USED

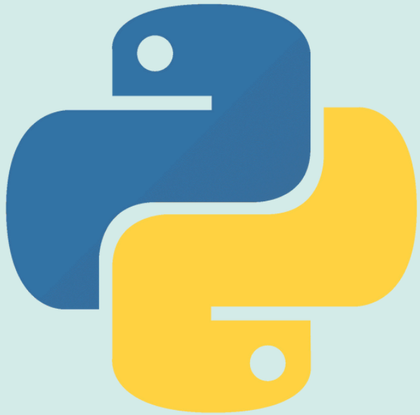
POSTGRESQL

For structured data management.



PYTHON

Main backend language.

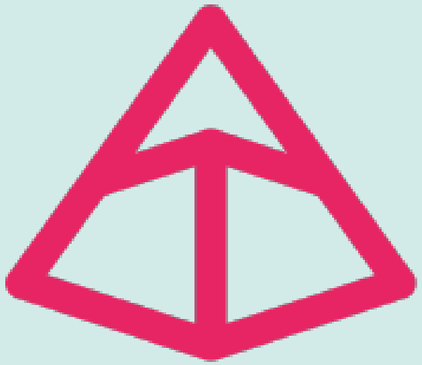


MONGODB

To store flexible or variable information.

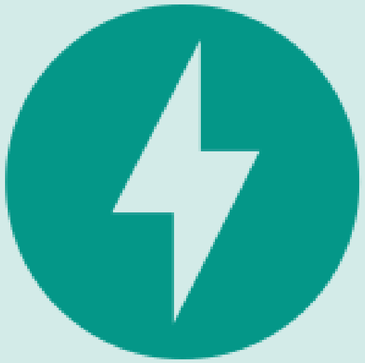
PYDANTIC

To validate and structure data in the API.



FASTAPI

As a framework to build the API efficiently.



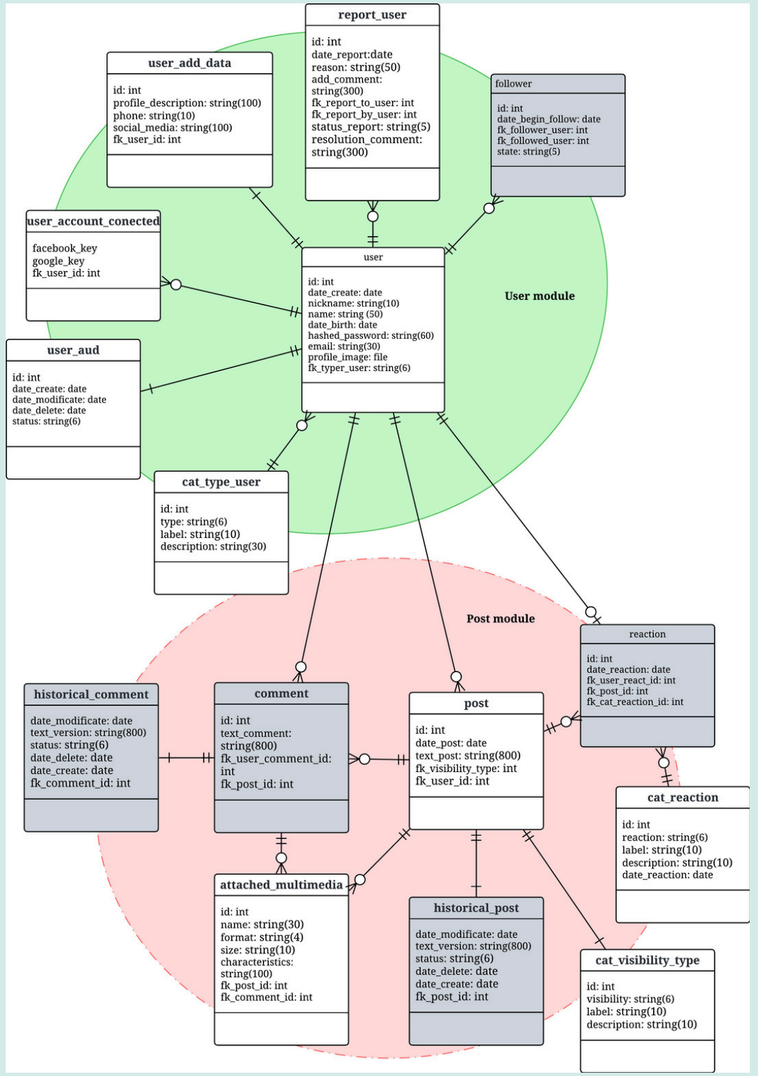
ENTITY-RELATIONSHIP MODEL

Modules:

- User (green): Account management, followers, reports and auditing
- Post (pink): Posts, comments, reactions and historics

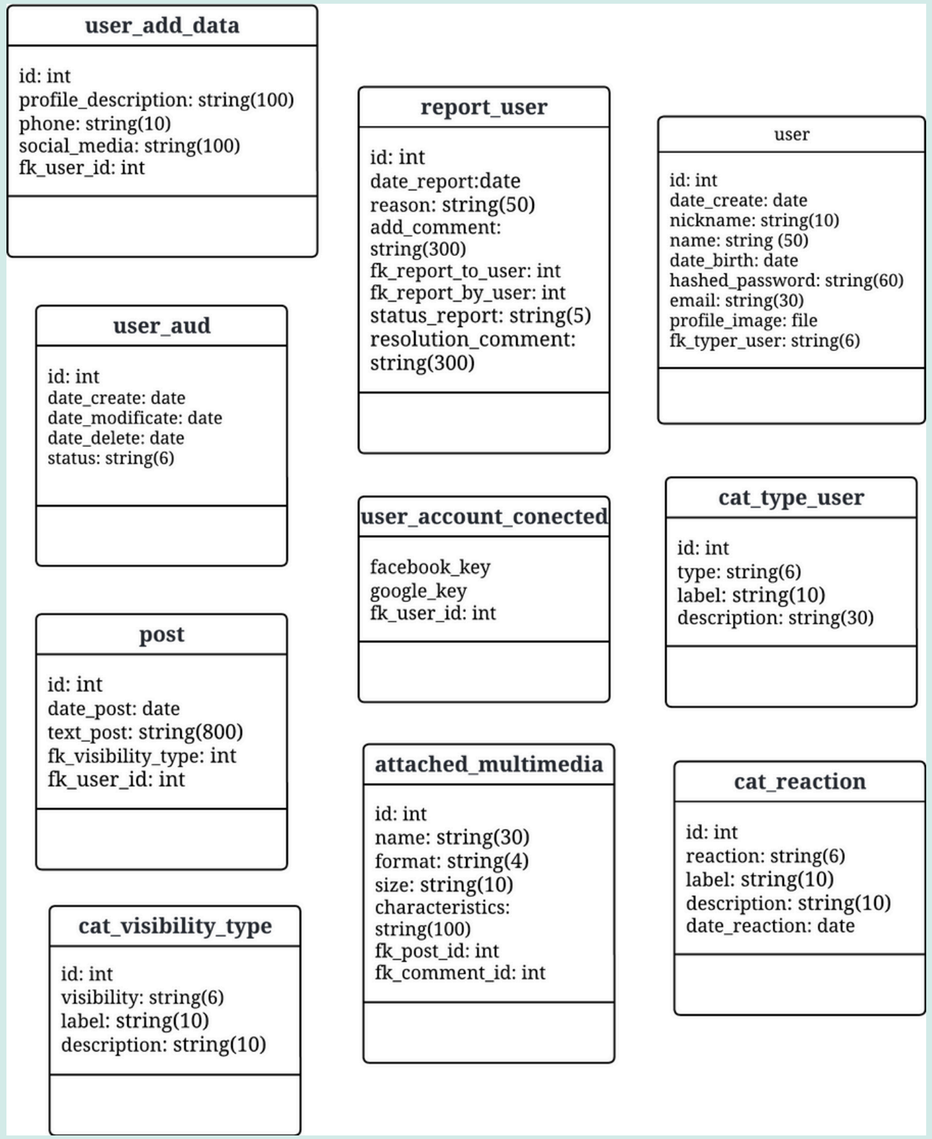
Key points:

- Normalization: Use of cat_* for types and states.
- History: Versioning and auditing of changes.
- Attachments: Files associated with post or comment.



WHY A SQL PART?

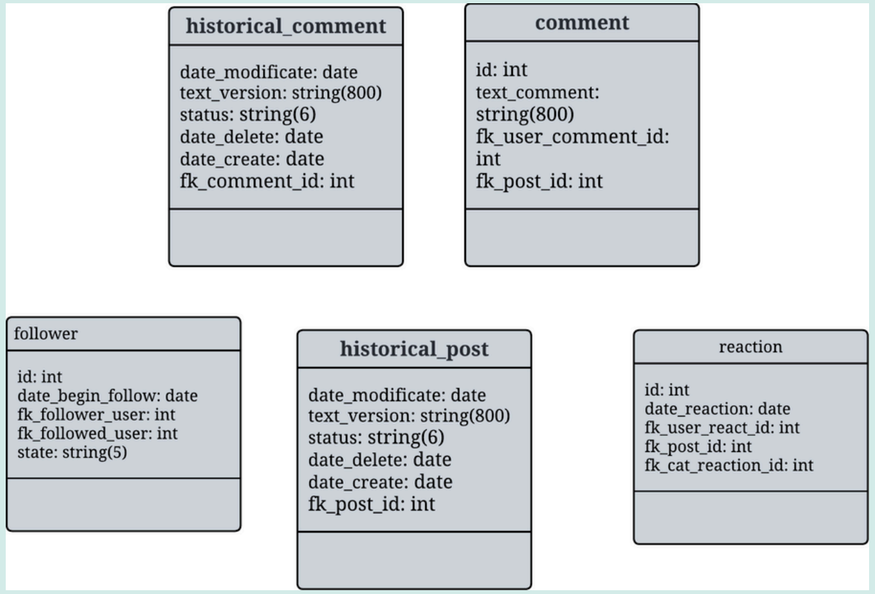
- Clear structure: Allows explicit definition of relationships using primary and foreign keys.
- Data integrity: Ensures the database follows strict rules (ACID) for reliable and consistent operations.
- Complex queries: Supports operations like JOIN, aggregations, and distributed transactions for more complex scenarios.
- Vertical scalability: Ideal for applications where data is highly interrelated and consistency is required.



WHY A NOSQL PART?

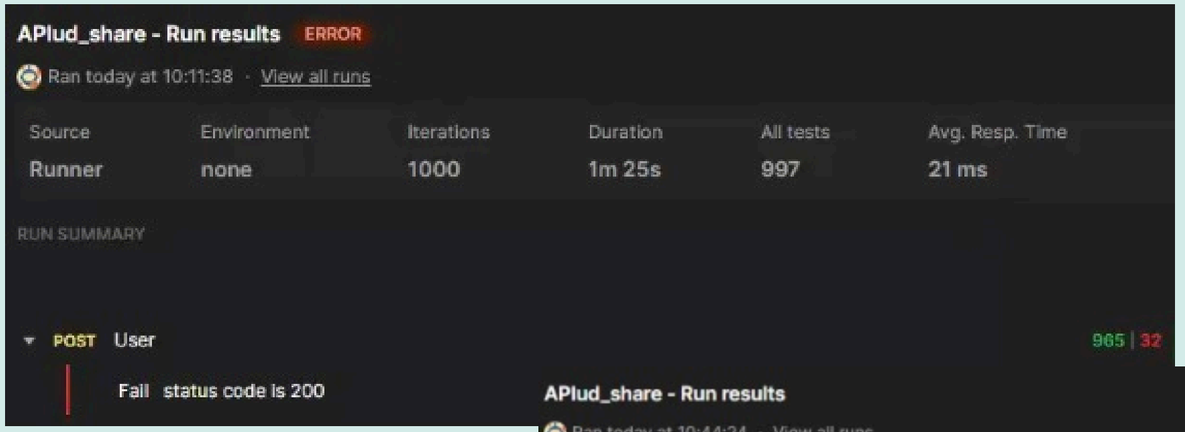
- Horizontal scalability: Handles large data volumes distributed without losing performance.
- Schema flexibility: Ideal when data is semi-structured or evolves rapidly.
- Performance: Designed to support fast operations with high read/write loads and low latency.
- Diverse models: Offers different types of databases (key-value, document, column, graph) tailored to specific use cases (e.g., MongoDB for JSON, Cassandra for columns).

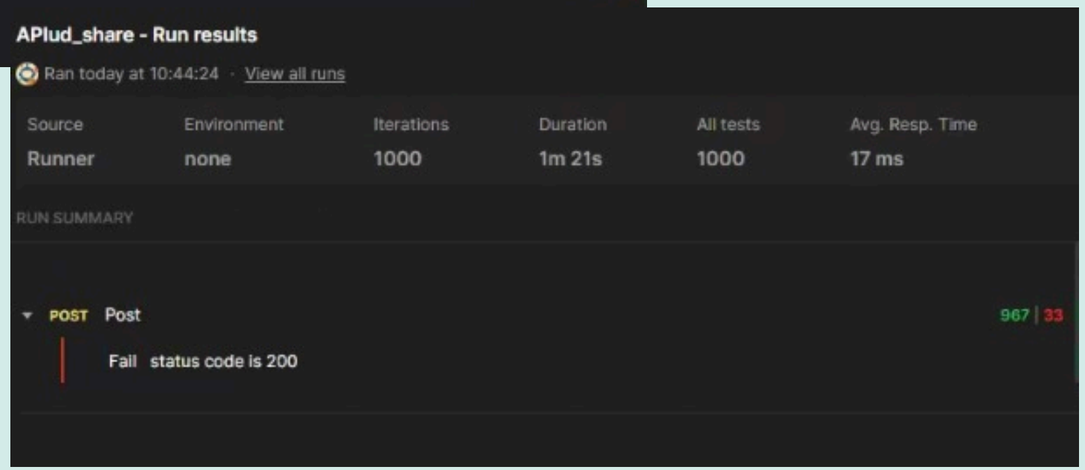
```
ud_sharem> show collections
comment
follower
historical_comment
historical_post
reaction
```

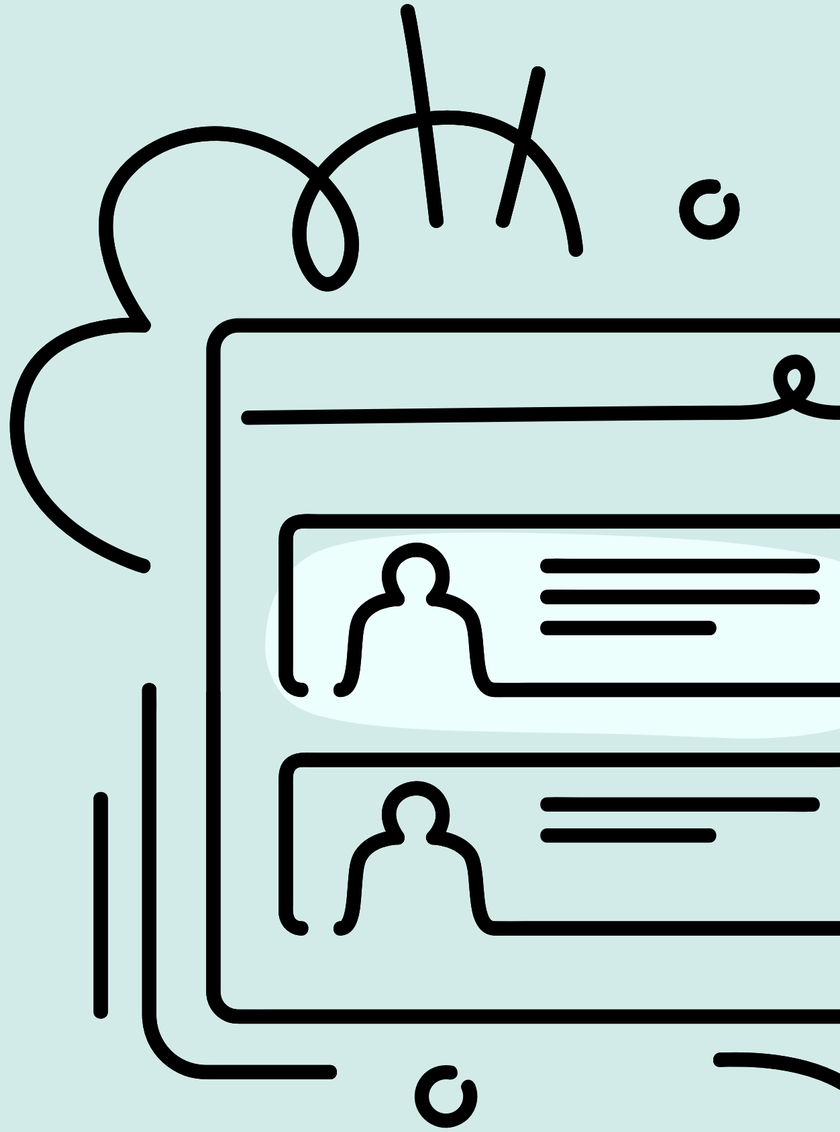


COMPARISON

For UD Share we load and request a good volume of data, below we can show what each database provides us in the processing of this data.







COMPARISON

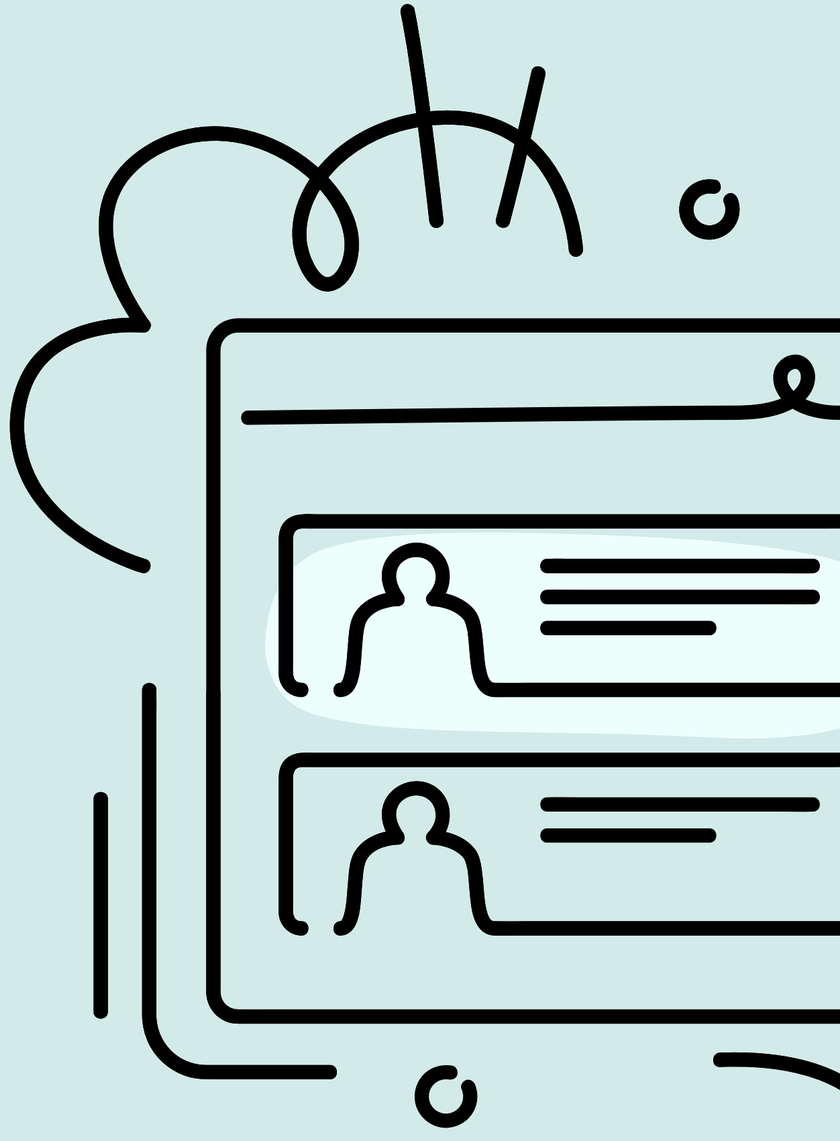
APIud_share - Run results

Ran today at 10:53:37 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1000	1m 12s	1000	11 ms

RUN SUMMARY

▸ POST Followers 1000 | 0



MODULARITY IN THE PROJECT

- Scalability: Easily add new features without affecting the system.
- Simplified maintenance: Changes or fixes are made independently in specific modules.
- Code reuse: Modules are reusable in other parts of the system or projects.
- Integration of new technologies: Allows the integration of new tools without compromising the system.
- Efficient collaboration: Teams can work in parallel on independent modules.
- Improved security: Each module has its own permission management and protection.

IN THE FUTURE...

Messaging Module:

- Real-time direct messaging between users.
- Features: Notifications, message history.

Kafka Integration:

- Use Apache Kafka for real-time data streaming.
- Event-based reactions: Handle reactions as events in Kafka for real-time processing.

Extended Interactions:

- Custom reactions: Introduce new reaction types (e.g., emojis, GIFs).
- Video comments: Enable video-based comments on posts.

AI Integration:

- Implement recommendation algorithms based on user behavior.
- Automatic moderation: AI-driven content filtering in real-time.

CONCLUSIONS

- Modularity is essential for scalability, flexibility, and ease of maintenance, allowing future improvements without disrupting the core system.
- Relational databases provide strong data integrity and consistency, ideal for managing user data, relationships, and complex queries.
- NoSQL databases excel in handling high volumes of data with fast read/write operations, making them perfect for real-time interactions like comments and reactions.

THANK YOU