WORKSHOP 2 - GOOGLE DRIVE CLONE

Cristian David Casas Diaz - Cód: 20192020091

Dylan Alejandro Solarte Rincón - 20201020088

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

PROFESSOR:

CARLOS ANDRÉS SIERRA VIRGUEZ

FRANCISCO JOSÉ DE CALDAS DISTRICT UNIVERSITY

COMPUTER ENGINEERING

DATABASES II

June 2025, Bogotá D.C.

Index

Introduction and Business Context

Problem Definition

Cloud storage solutions have become an integral part of personal and organizational digital infrastructures. However, existing commercial services like Google Drive, Dropbox, and OneDrive often present challenges such as limited flexibility, high costs, and inadequate analytics capabilities for smaller enterprises and educational institutions. Particularly, these solutions frequently fail to efficiently handle large data volumes while maintaining performance and robust analytics integration, hindering the ability of users and administrators to derive insightful and actionable data.

Business Scenario and Opportunity

The growing digital data generation across diverse sectors—including education, SMEs, and individual users—demands secure, scalable, and cost-effective cloud storage platforms. There exists a substantial market opportunity for a cloud storage service tailored specifically to these segments. Such a platform would emphasize affordability, enhanced analytics integration, and robust performance, addressing the existing gaps in the market.

Proposed Solution

Our project aims to create a secure and scalable cloud storage platform inspired by Google Drive but uniquely tailored to offer advanced analytics, superior data management, and robust security features within a cost-effective model. The platform will leverage a hybrid database approach, combining relational databases for structured user data and NoSQL databases for metadata and analytics. By employing a hexagonal (ports and adapters) architecture, we will achieve high modularity, ease of testing, and system maintainability.

Key differentiators of our platform include:

- Advanced analytics integration for enhanced insights into data usage and storage patterns.
- Modular architecture ensuring maintainability, scalability, and efficient performance.
- Cost-effective tiered storage solutions to optimize long-term costs and performance.

Scope and Objectives

The primary scope of the project includes:

- Secure file storage with metadata management.
- Role-based access control and secure file-sharing capabilities.
- Real-time analytics dashboard to visualize storage usage and user activity.
- Responsive web-based interface for user interaction.

Exclusions from the scope clearly define that there will be no native mobile or desktop applications, offline synchronization, or real-time collaborative editing.

Improved Database Architecture

This system architecture represents a scalable and secure cloud-based file management platform inspired by Google Drive. The platform is designed using a hexagonal (ports and adapters) architecture, ensuring modularity, ease of testing, and maintainability.

[Link Repository Architecture V 2.0]

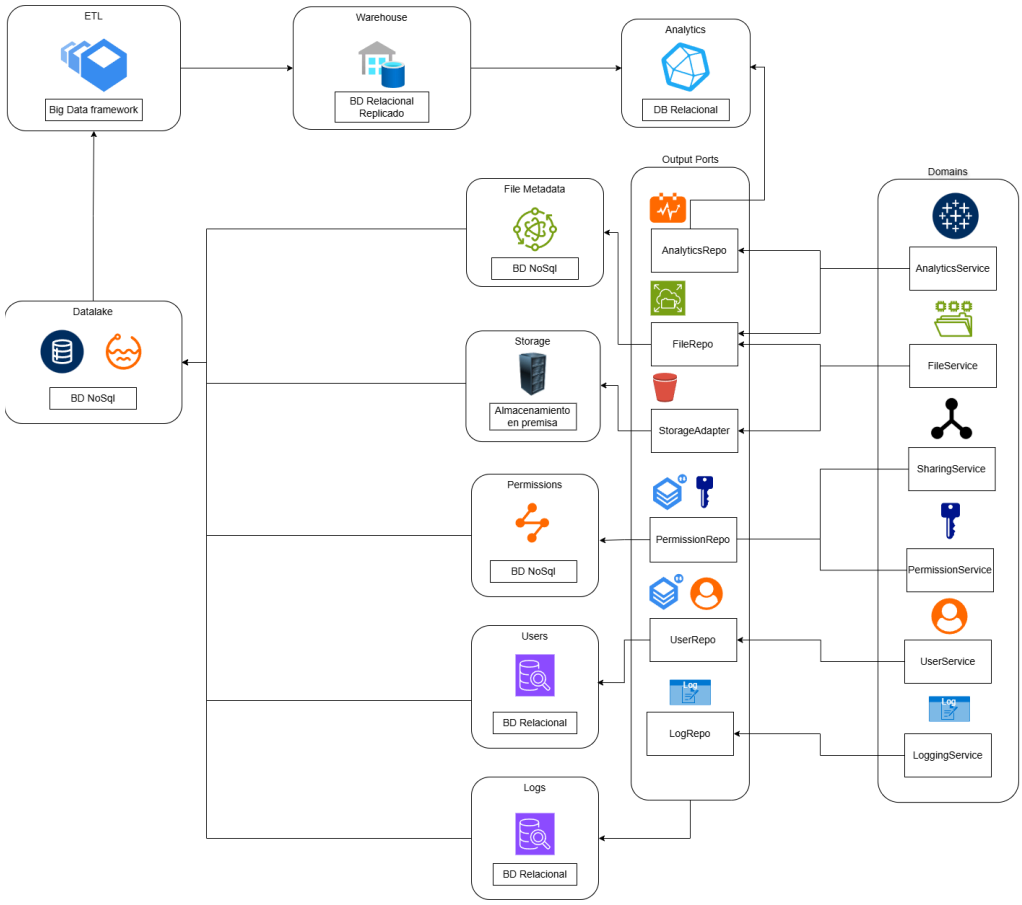

Fig. Database Architecture

Roles and technologies of Each Component:

User: Web client.

API Gateway: Responsible for orchestrating communication between services and between the logic and presentation layers.

Domains: Encapsulate each service module required for the application's operation and data analysis.

- **AnalyticsService:** Responsible for managing the logic and communication with the analytics repository.
- **FileService:** Responsible for managing the logic and communication with file repositories and the storage adapter.
- **SharingService & PermissionService:** Responsible for managing the logic and communication with the permissions repositories.
- **UserService:** Responsible for managing the logic and communication with the user repository.
- **LogService:** Responsible for managing the logic and communication with the log repository.

Repositories: Responsible for connecting to and performing operations related to databases and storage.

File Metadata DB: A NoSQL database for handling file metadata. NoSQL is chosen due to the variability of file types and the specific characteristics each one may have.

Storage DB: This refers to on-premise storage for static files, aimed at providing access and downloads. An on-premise service is selected as it offers better long-term cost efficiency.

Permissions DB: Due to the complex nature of permissions, a non-relational database is the better option.

Users DB: A relational (SQL) database will be used to store user information. This type of database is chosen because the data is easily structured.

Log DB: Similar to the users case, logs are easily structured and will be stored in a relational database.

Datalake: A NoSQL database will be used to store information obtained from the data sources (the databases of each domain), acting as a datalake to store data for future cleaning and transformation.

ETL: This module will handle the data cleaning and transformation processes for storage in the data warehouse.

Warehouse: A series of relational databases, along with their replicas, will be used to store information after the ETL process and distribute it to the data marts.

Analytics: An analytics module that will function as a data mart to store information related to metrics and analytics.

System Information Requirements

User Information

Required data:

- Name, email, encrypted password
- Email verification status
- User type (standard, premium, administrator)
- Activity history (last login, changes, sharing events)

Related use cases:

- Registration and login (RF1.1 to RF1.3)
- Password recovery and profile editing (RF1.4, RF1.5)
- Access permission management (RF5.5, RF5.8)
- Two-factor authentication (RF8.4)

Files and Metadata

- **Required data:**

- File ID, name, extension, size
- File owner
- Upload date, modification date, last view
- Status (active, in trash, permanently deleted)
- File version

Related use cases:

- File upload, download, and organization (RF2.1 to RF2.6, RF3.1)

- Deleted file recovery (RF2.6)
- Quick preview (RF2.3, RF9.1)
- Large file support (User Story: *Large File Support*)

Folder Structure

Required data:

- Folder ID, name, parent-child hierarchy
- Folder owner
- Creation and modification dates

Related use cases:

- Folder creation and organization (RF3.1, RF3.2)
- Renaming and moving files/folders (User Story: *Rename and Move Files*)

Access Permissions and Sharing

Required data:

- Shared file or folder
- Users with access and their permissions (view, edit, download)
- Passwords for shared links
- Access expiration date

Related use cases:

- Sharing with users or via links (RF5.1 to RF5.8)
- User Stories: *Access Permissions*, *Shareable Links*

Activity Logs and Auditing

Required data:

- Type of activity (upload, edit, download, share)
- User who performed the action
- Date and time

- IP address or origin

Related use cases:

- Activity log display (User Story: *Activity Log*)
- Security auditing (RF8.3)

Search and Tagging Data

Required data:

- Search indexes by name, type, date, size
- Assigned tags
- Favorite files

Related use cases:

- Advanced search (RF4.1 to RF4.3)
- User Stories: *Quick Search*, *Favorite Files and Folders*

Statistics and Analytics

Required data:

- Storage usage per user
- Most accessed files
- User behavior (access frequency, most used file types)
- System performance

Related use cases:

- User Story: *Analytics Dashboard*
- RF11.1 to RF11.9

Security and Compliance

Required data:

- Successful and failed login attempts
- Malware events or violations

- Encrypted backups

Related use cases:

- Encryption, 2FA, antivirus scanning (RF8.1, RF8.2, RNF4.6)

Queries for System Information Requirements

The following queries have been designed to meet the system's various information requirements, each with a defined purpose and corresponding requirement to fulfill.

Information Requirement	Purpose	Example SQL and NoSQL Queries
User Information	Retrieve basic user profile and status	SELECT user_id, full_name, email, account_type, last_login, email_verified FROM User WHERE is_actived = TRUE;

Files and Metadata	List files uploaded by a user with metadata.	db.files.find(
		{ owner_id: 123, is_deleted: false },
		{
		file_id: 1,
		file_name: 1,
		file_type: 1,
		size_kb: 1,
		created_at: 1,
		last_modified: 1
		}
		);

Folder Structure	Retrieve a user's folder hierarchy.	db.folders.find(
		{ owner_user_id: 123, is_deleted: false },
		{
		folder_id: 1,
		folder_name: 1,
		parent_folder_id: 1,
		created_at: 1
		}
		).sort({ parent_folder_id: 1, created_at: 1 });

Access Permissions and Sharing	Query active access permissions for shared resources.	db.sharing.find({ is_active: true, shared_with_id: 123 }, {

		resource_id: 1, shared_with_id: 1, permission_level: 1, shared_at: 1, expire_at: 1 });
Activity Logs and Auditing	Show last 10 actions by a user	SELECT activity_type, resource_id, timestamp, ip_address FROM Activity_Log WHERE user_id = 123 ORDER BY timestamp DESC LIMIT 10;

Search and Tagging	Find files associated with a specific tag.	db.resource_tags.aggregate([
		{ $match: { tag_name: "project2025" } },
		{
		$lookup: {
		from: "files",
		localField: "resource_id",
		foreignField: "file_id",
		as: "files"
		}
		},
		{ $unwind: "$files" },
		{ $project: { file_id: "$files.file_id", file_name: "$files.file_name" } }
		]);

Statistics and Analytics	Calculate used and remaining storage for a user	SELECT u.user_id, u.used_storage, u.storage_quota, (u.storage_quota - u.used_storage) AS remaining_storage FROM User u WHERE u.user_id = 123;
Security and Compliance	Audit failed login attempts for a user	SELECT timestamp, ip_address, user_agent FROM Authentication_Log WHERE user_id = 123 AND success = FALSE ORDER BY timestamp DESC;

Improve Workshop 1

Business Model Canvas Overview

The Business Model Canvas developed for this project emphasizes value propositions such as:

- Secure, scalable, and cost-effective cloud storage.
- Integrated advanced analytics for enhanced decision-making.
- Granular security settings for controlled data access.

Revenue streams will primarily derive from premium subscriptions offering advanced security features, extended storage capabilities, and customized enterprise solutions.

Analysis of Business Model Components

- **Value Propositions**: Provide affordable cloud storage enhanced with analytics, targeted toward educational institutions and SMEs.
- **Customer Segments**: Primarily SMEs, educational institutions, and individual users seeking robust analytics capabilities.
- **Revenue Streams**: Subscription-based premium services and customizable enterprise solutions.
- **Key Activities**: Development and maintenance of cloud infrastructure, security and compliance management, analytics integration.
- **Cost Structure**: Driven by development, cloud hosting, infrastructure maintenance, and support staff salaries.

This structured introduction and business context clearly outline the project's motivation, opportunities, and strategic approach, providing a comprehensive understanding for stakeholders.

Data Source and Structure Analysis

Data Management

Our platform will manage diverse data types:

- **User Data**: Registration details, authentication logs, subscription status, and activity history.
- **File Metadata**: File names, types, sizes, timestamps, and version history.

- **Sharing and Permissions Data**: Access control lists (ACLs), shared links, expiration and permission rules.
- **Activity Logs**: User interactions including file operations, access events, and modifications.
- **Analytics Data**: Usage patterns, storage statistics, performance metrics.
- **Security Data**: Authentication attempts, security event logs, encrypted backups.

Volume, Velocity, Variety, and Source Analysis

- **Volume**: Based on Gartner's forecast, cloud storage market growth averages around 21% annually, supporting scalability targets of approximately 5,000 concurrent users per infrastructure node (Gartner, 2024).
- **Velocity**: Data velocity is guided by industry benchmarks for cloud uploads, targeting an upload speed of 1 MB/s per concurrent user, which aligns with typical upload speeds seen in standard cloud services according to CloudHarmony's Cloud Speed Tests (CloudHarmony, 2024).
- **Variety**: The platform's data will encompass structured (user information) and unstructured formats (files, logs, metadata), conforming to hybrid database models supported by IBM's best practices for cloud storage solutions (IBM Cloud, 2024).
- **Source Types**: User interactions, automated analytics, and system security events, as outlined by Cisco's framework for cloud security and data management (Cisco, 2024).

Functional Requirements

Before each module, a **targeted analysis** based on industry studies and benchmarks is provided.

Module 1: User Management (High)

- **Targeted Analysis:** The *Customer Identity Trends Report 2023* by Okta shows that nearly 60 % of consumers prefer simple and secure registration and login processes; friction in authentication reduces conversion rates and increases support costs .

1. **FR1.** As a user, I want to register with an email and password to access the platform.

2. **FR2.** As a user, I want to verify my account via an email link to activate my profile.

3. **FR3.** As a user, I want to log in with my validated credentials to access my files.

4. **FR4.** As a user, I want to recover my password via email to restore access.

Module 2: File Management (High)

- **Targeted Analysis:** According to Nielsen Norman Group, the most frequent actions in cloud-storage systems are sharing information, accessing files anywhere, and data

synchronization; uploading, downloading, and previewing files are critical for user adoption .

5. **FR5.** As a user, I want to upload files (documents, images, videos) by dragging them into the browser to store them in the cloud.

6. **FR6.** As a user, I want to download individual or multiple files (as a ZIP) to have them locally.

7. **FR7.** As a user, I want to preview PDFs and images without downloading them to save time.

8. **FR8.** As a user, I want to rename, move, and delete files to keep my space organized.

Module 3: Organization and Search (Medium)

- **Targeted Analysis:** Nielsen's "Consistency and Standards" heuristic highlights the need for predictable structures; organizing folders and offering robust search reduces errors and improves user efficiency .

9. **FR9.** As a user, I want to create, rename, and delete folders to structure my files.

10. **FR10.** As a user, I want to search by name, type, and date to quickly locate my documents.

11. **FR11.** As a user, I want to filter and sort search results (by date, size) to refine my queries.

Module 4: Sharing and Collaboration (Medium)

- **Targeted Analysis:** Sharing information is the main use case for cloud storage; enabling configurable links and notifications is key to effective collaboration .

12. **FR12.** As a user, I want to generate shareable links with configurable permissions (read-only, download) to collaborate.

13. **FR13.** As a user, I want to receive notifications when someone requests access to my files so I can approve or reject.

14. **FR14.** As a user, I want to revoke shared access permissions to maintain control over my resources.

Module 5: Security and Authentication (High)

- **Targeted Analysis:** RSA Security's *Top Trends in Identity for 2024* indicates that 90 % of organizations experience identity-related incidents; implementing 2FA and access auditing

mitigates critical risks .

15. **FR15.** As a user, I want to enable two-factor authentication to strengthen my account security.

16. **FR16.** As a user, I want to view an activity log (who and when accessed my files) to audit access.

Module 6: Administration and Monitoring (Low)

- **Targeted Analysis:** Gartner's *Magic Quadrant for Strategic Cloud Platform Services* highlights real-time monitoring and metrics as a differentiator for managing cloud platforms .

17. **FR17.** As an administrator, I want to view a metrics dashboard (storage usage, number of users) to optimize resources.

2. Non-Functional Requirements

Note: Given the limitations of a two-person team, the numeric targets have been adjusted to be realistic and achievable.

- ### Performance

    - *Targeted Analysis:* Object storage platforms exhibit average access latencies of 20–30 ms per operation .

    - **RNF1.** The system must render folder views containing up to 1,000 files in under **1 second**.

- ### Scalability & Availability

    - *Targeted Analysis:* NIST SP 800-53 Rev. 5 recommends N+1 replication and horizontal scaling to maintain high availability in critical environments .

    - **RNF2.** The architecture must support horizontal scaling, handling at least **500** concurrent users per node at < 70 % CPU utilization.

    - **RNF3.** Ensure **99.9 %** availability through N+1 replication and daily backups.

- **Security**
    - *Targeted Analysis:* The Cisco Cloud Controls Framework mandates AES-256 encryption at rest and TLS 1.3 in transit to comply with international standards .
    - **RNF4.** All files must be encrypted at rest with AES-256 and in transit with TLS 1.3.
- **Usability**
    - *Targeted Analysis:* A Nielsen Norman Group study shows a task success rate of 82 % on new websites in 2016; setting a target of 80 % aligns with this trend .
    - **RNF5.** At least **80 %** of new users must be able to upload, download, and share files without external guidance in their first session (< 3 minutes).

User Stories

The user stories for this project are defined following the industry-standard INVEST criteria (Independent, Negotiable, Valuable, Estimable, Small, Testable). Prioritization follows the standard three-tiered priority classification of High, Medium, and Low, aligning with best practices in agile project management to clearly indicate the urgency and criticality of each story. Estimates are performed initially using Story Points based on complexity, uncertainty, and effort, and subsequently converted into development days collaboratively determined by the team, with one Story Point roughly equivalent to one development day.

Criterion	Description
INVEST	Independent, Negotiable, Valuable, Estimable, Small, Testable
Prioritization	Three-tiered (High, Medium, Low) aligning with agile best practices to indicate urgency and criticality
Estimation Method	Story Points based on complexity, uncertainty, and effort, converted collaboratively into days (1 Point ≈ 1 day)

Revised User Stories

Title: File Synchronization	**Priority:** High	**Estimate:** 8 days
User Story: As a user, I want real-time synchronization of my files across multiple devices to always access the most recent version without manual updates.		
Acceptance Criteria: ● Files automatically sync across devices when a change occurs. ● Users receive notifications if synchronization conflicts arise.		

Title: Activity Log	**Priority:** Medium	**Estimate:** 5 days
User Story: As a user, I want an intuitive activity log showing detailed file access and modification history to track changes efficiently.		
Acceptance Criteria: ● Activity logs clearly show file access type, timestamp, and the user responsible. ● Logs support filtering by date, action type, and user.		

Title: Analytics Dashboard	**Priority:** High	**Estimate:** 13 days
User Story: As an administrator, I want comprehensive analytics dashboards with interactive visualizations of storage usage, user activity, and performance metrics to make informed management decisions.		
Acceptance Criteria: ● Dashboard displays real-time analytics data clearly.		

- Supports drill-down features for detailed insights.

Entity-Relationship Diagram

Step 1. Define Components (Context & Scope)

Domain: "Drive-style" system for users, folders, files, versions, tags, favorites, auditing and sharing.

Scope:

- **User metadata** (account, profile, quotas)
- **Folder & file hierarchy**
- **File versioning**
- **Tagging & favorites**
- **Sharing (links & permissions)**
- **Activity and authentication logs**

Step 2. Identify Main Entities

1. **User**
2. **Folder**
3. **File**
4. **Version_History**
5. **Tag**
6. **Resource_Tag** (associative entity)
7. **Favorite**

8. **Sharing**
9. **Sharing_Link**
10. **Activity_Log**
11. **Authentication_Log**

Step 3. Define Attributes per Entity

Entity	Key Attributes & Notes
User	user_id (PK), email, password_hash, full_name, created_at, last_login, email_verified, is_active, account_type, storage_quota, used_storage
Folder	folder_id (PK), owner_id (FK→User), parent_folder_id (self-FK), name, created_at, last_modified, is_deleted
File	file_id (PK), owner_id (FK→User), parent_folder_id (FK→Folder), name, mime_type, size, checksum, created_at, last_modified, is_deleted, encryption_key
Version_History	version_id (PK), file_id (FK→File), created_at, version_path, size, checksum
Tag	tag_id (PK), created_by (FK→User), tag_name

Resource_Tag	resource_tag_id (PK), resource_id, resource_type, tag_id (FK→Tag)
Favorite	favorite_id (PK), user_id (FK→User), resource_id, resource_type, added_at
Sharing	sharing_id (PK), resource_id, resource_type, shared_by (FK→User), shared_with (FK→User), permission_level, shared_at, expires_at, is_active
Sharing_Link	link_id (PK), resource_id, resource_type, access_token, permission_level, created_at, expires_at, password_protected, password_hash, is_active
Activity_Log	log_id (PK), user_id (FK→User), activity_type, resource_id, resource_type, timestamp, ip_address, user_agent, is_active
Authentication_Log	auth_log_id (PK), user_id (FK→User), action_type (login/logout), timestamp, ip_address, user_agent, success

Step 4. Define Relationships

- **User 1–N Folder** (a user owns many folders)
- **Folder 1–N File** (a folder contains many files)
- **File 1–N Version_History** (a file may have multiple versions)
- **User 1–N Tag** (each user creates tags)

- **Tag N–M Resource** (file or folder) → via **Resource_Tag**
- **User 1–N Favorite** (any resource)
- **User 1–N Sharing** as both sharer and sharee (direct sharing)
- **Resource 1–N Sharing_Link** (public share links)
- **User 1–N Activity_Log**
- **User 1–N Authentication_Log**

Step 5. Define Cardinalities

Relationship	Cardinality
User → Folder	1 to N
Folder → Folder (parent ⇄ child)	0..1 to N
Folder → File	1 to N

File → Version_History	1 to N
User → Tag	1 to N
Tag ↔ Resource (via Resource_Tag)	N to M
User → Favorite	1 to N
User (shared_by) → Sharing	1 to N
Resource → Sharing_Link	1 to N
User → Activity_Log	1 to N
User → Authentication_Log	1 to N

Step 6. First ER Draft

In this step we produce the initial Entity–Relationship diagram based on the entities and attributes already defined. The aim is to visualize how **User**, **Folder**, **File**, and the supporting entities (**Tag**, **Favorite**, **Activity_Log**, **Authentication_Log**, **Resource_Tag**, **Sharing**, **Sharing_Link**, **Version_History**) relate to one another before refining cardinalities and resolving any M:N

relationships.

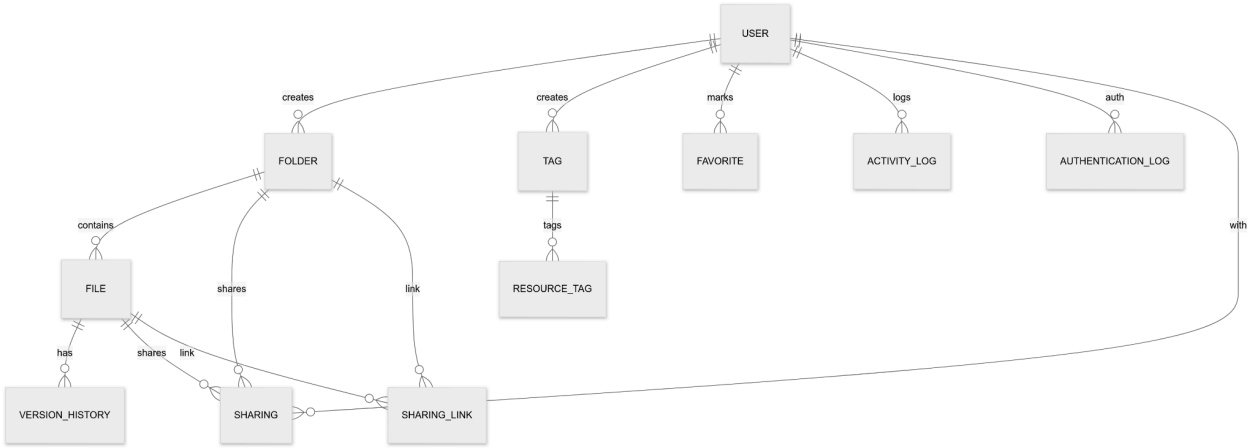

Fig 2. First ER Draft

Step 7. Split Many-to-Many Relationships

- **Tag ↔ Resource** modeled as **Resource_Tag** (auto-increment PK + FKs).
- **Sharing** handles a ternary association (resource_type + shared_with).
- **Favorite** is a 1–N with resource typing.

Step 8. Second ER Draft (Refined)

In this refinement we explicitly **separate out** the high-volume, flexible-schema tables—**Version_History**, **Activity_Log** and **Authentication_Log**—to reside in a NoSQL database. Meanwhile, our relational schema is tightened: the associative tables **Resource_Tag**, **Favorite**, **Sharing** and **Sharing_Link** are each extended with an explicit resource_type field (e.g. “file” or “folder”), so they can uniformly manage both resource types within PostgreSQL. This separation optimizes performance and clarity across SQL and NoSQL stores.

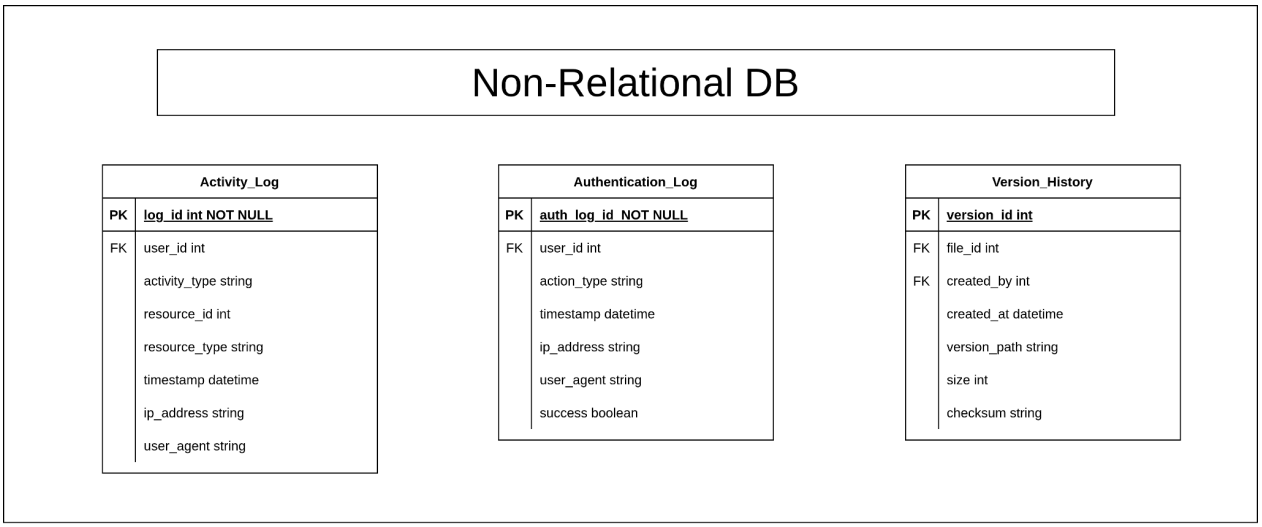

Fig 3. Second ER Draft (Refined)

Step 9. Map ER to Physical Structures

- Convert each ER entity to a relational table:
    - **PK** as INT AUTO_INCREMENT NOT NULL
    - Audit columns (created_at, last_modified) as TIMESTAMP
    - Boolean flags (is_deleted, is_active)

Step 10. Define Constraints & Properties

- **PRIMARY KEY** on each …_id column.
- **FOREIGN KEY** with ON DELETE CASCADE or SET NULL:
    - Deleting a **User** cascades folders/files.
    - Deleting a parent **Folder** cascades or nullifies children.
- **UNIQUE** constraint on User.email.
- **NOT NULL** on mandatory fields (folder_name, file_name, etc.).

- **CHECK** constraints (e.g. valid permission_level).
- **Indexes**:
    - (resource_type, resource_id) for share/favorite/tag tables.
    - (user_id, timestamp) for logs.
- **Data types**:
    - VARCHAR(255) for short strings, TEXT for long strings.
    - BOOLEAN flags.
    - INT for keys and numeric fields.
    - DECIMAL(12,2) for quotas or prices.

With all primary and foreign keys defined, cascading and null-setting rules in place, unique and not-null constraints enforced, and appropriate indexes and data types selected, the result is a robust hybrid schema. Core entities and their associations (User, Folder, File, Tag, Resource_Tag, Favorite, Sharing, Sharing_Link) live in a relational database under strict relational constraints, while high-volume collections (Activity_Log, Authentication_Log, Version_History) are stored in a non-relational model for scalable, flexible storage. This split ensures both data integrity and performance.

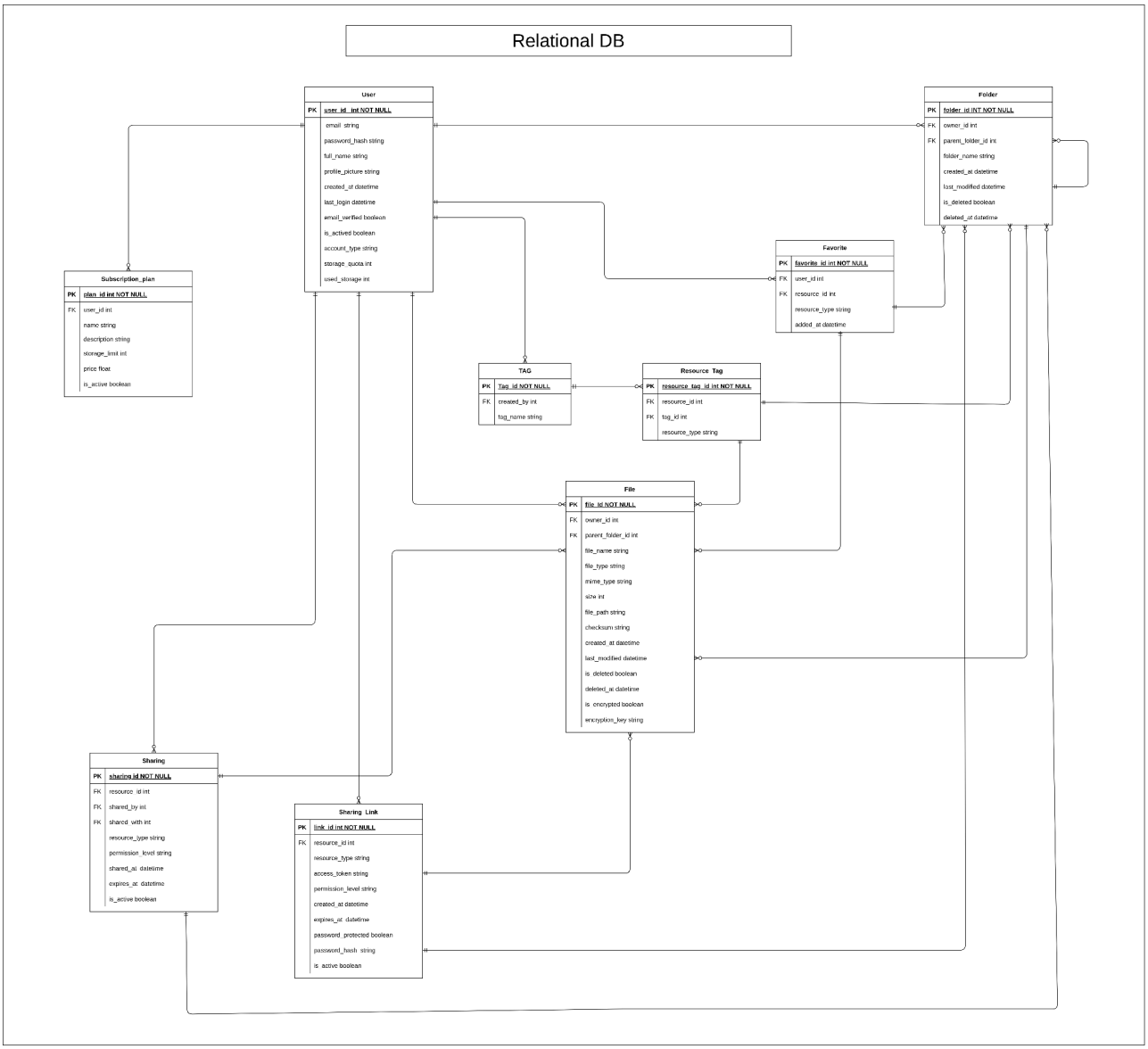

Fig 4. Relational Database Model

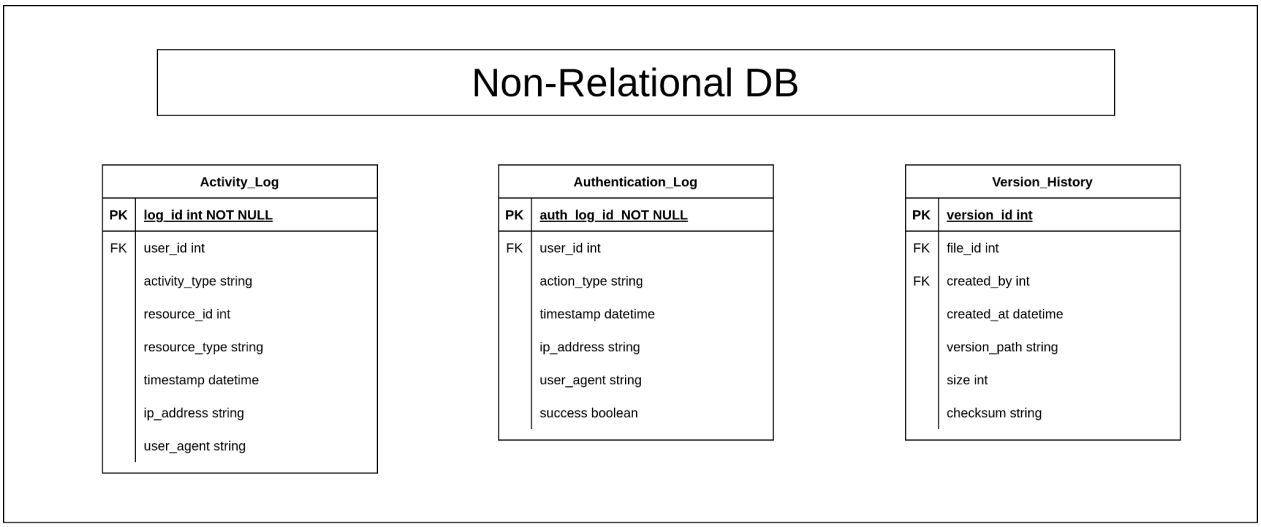

Fig 5. Non - Relational DB

References

Amazon Web Services. (n.d.). *What is data architecture?* AWS. Retrieved May 13, 2025, from https://aws.amazon.com/es/what-is/data-architecture/

Business model canvas examples. (n.d.). *Corporate Finance Institute*. Retrieved March 17, 2025, from https://corporatefinanceinstitute.com/resources/management/business-model-canvas-examples/

IBM. (n.d.). *What is data architecture?* IBM. Retrieved May 13, 2025, from https://www.ibm.com/es-es/topics/data-architecture

UNIR. (n.d.). *Entity-relationship model*. UNIR Revista. Retrieved May 13, 2025, from https://www.unir.net/revista/ingenieria/modelo-entidad-relacion/

Workspace, G. (n.d.). *Google Drive: Share files online with secure cloud storage*. Google Workspace. Retrieved March 17, 2025, from https://workspace.google.com/intl/es-419/products/drive/

Gartner. (2024). Gartner expects end-user cloud spending to increase 21.5% globally to $723 billion in 2025. *CIODive*, retrieved from https://www.ciodive.com/news/cloud-spend-growth-forecast-2025-gartner/733401/?utm_source=chatgpt.com

CloudHarmony. (2010). CloudHarmony benchmarks detail inter-cloud issues. *Horizoniq*. Retrieved from https://www.horizoniq.com/blog/latency-matters-in-the-cloud-cloudharmony-benchmarks-detail-inter-cloud-issues/?utm_source=chatgpt.com

Cisco. (2024). Cloud Security Framework. Retrieved from https://www.cisco.com/c/en/us/about/trust-center/compliance/ccf.html?utm_source=chatgpt.com

Cisco. (2024). *Cloud Security Framework*. Retrieved from https://www.cisco.com/c/en/us/about/trust-center/compliance/ccf.html?utm_source=chatgpt.com

Gartner. (2023). *Magic Quadrant for Strategic Cloud Platform Services*. Retrieved from https://www.gartner.com/en/documents/xyz

Greptime et al. (2024). *Cloud Storage Data Request Model*. Retrieved from https://www.greptime.io/reports/cloud-storage-data-request-model

Nielsen Norman Group. (2016). *Twenty Years of Usability Studies*. Retrieved from https://www.nngroup.com/articles/twenty-years-usability-studies/

Nielsen Norman Group. (2018). *Mental Models for Cloud-Storage Systems*. Retrieved from https://www.nngroup.com/articles/mental-models-cloud-storage/

Nielsen Norman Group. (2019). *Maintain Consistency and Adhere to Standards (Usability Heuristic #4)*. Retrieved from https://www.nngroup.com/articles/consistency-standards/

NIST. (2020). *Security and Privacy Controls for Information Systems and Organizations (SP 800-53 Rev. 5)*. Retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf

Okta. (2023). *Customer Identity Trends Report 2023*. Retrieved from https://www.okta.com/resources/customer-identity-trends/

RSA Security. (2024). *Top Trends in Identity for 2024*. Retrieved from https://www.rsa.com/en-us/content/top-trends-identity-2024.html