System Architecture Manual

CRNube StoreIT System

Authors:

María Segura Largaespada

Amanda Zamora Ramírez

Santiago Araya Espinoza

Alcides Jiménez Carrillo

Laura Flores Barrantes

Abril Urcuyo Arce

Release date: 13-05-2026

Version: 5.0

Revision History

Date	Version	Description	Author
07-05-26	5.0	The following sections were updated: New user registration, Transaction logs, Subsystem or Module decomposition view, Updated System Backlog, and Data Layer.	María Segura Largaespada Amanda Zamora Ramírez Santiago Araya Espinoza Alcides Jiménez Carrillo Laura Flores Barrantes Abril Urcuyo Arce
11-03-26	4.1	Section 3.3 of the document (Updated System Backlog) was updated.	María Segura Largaespada Amanda Zamora Ramírez Santiago Araya Espinoza Alcides Jiménez Carrillo Laura Flores Barrantes Abril Urcuyo Arce
12-02-26	4.0	The following sections were updated: Technical Platform, Business Layer, Data Layer, and Process View. Changes are based on progress made during the first 2 Sprints of the project. Section 3.3 remains pending.	María Segura Largaespada Amanda Zamora Ramírez Santiago Araya Espinoza Alcides Jiménez Carrillo Laura Flores Barrantes Abril Urcuyo Arce
10-09-25	3.0	The data layer diagram for the web application was updated.	María Segura Largaespada Amanda Zamora Ramírez Santiago Araya Espinoza Alcides Jiménez Carrillo Laura Flores Barrantes Abril Urcuyo Arce
17-08-25	2.0	Changes suggested by the lead professor after the first review were implemented.	María Segura Largaespada Amanda Zamora Ramírez Santiago Araya Espinoza Alcides Jiménez Carrillo Laura Flores Barrantes Abril Urcuyo Arce

13-08-25	1.2	First 100% complete version of the document. Compared to the previous version, changes suggested by the lead professor and the company were incorporated.	María Segura Largaespada Amanda Zamora Ramírez Santiago Araya Espinoza Alcides Jiménez Carrillo Laura Flores Barrantes Abril Urcuyo Arce
05-08-25	1.1	Practically the entire document was completed. Only the updated system backlog and the scenario view were missing.	María Segura Largaespada Amanda Zamora Ramírez Santiago Araya Espinoza Alcides Jiménez Carrillo Laura Flores Barrantes Abril Urcuyo Arce
30-07-25	1.0	First draft of the document. Points 1 and 2 were completed. All diagrams remained pending.	María Segura Largaespada Amanda Zamora Ramírez Santiago Araya Espinoza Alcides Jiménez Carrillo Laura Flores Barrantes Abril Urcuyo Arce

Table of Contents

1. Introduction 5
   1.1 Purpose 5
   1.2 Scope of the Architecture Document 5
   1.3 Related Documents 5
2. Considerations and Limitations 6
   2.1 Technical Platform 6
   2.2 Portability 7
   2.3 Security 7
      2.3.1 Password Composition and Assignment Controls (Security Policies). 7
      2.3.2 New User Registration 8
      2.3.3 User and System Component Access Control. 10
      2.3.4 Transaction Logs 13
   2.4 Reliability/Availability 14
   2.5 Performance 15
      2.5.1 Performance Requirements 15
      2.5.2 Transaction Volume 15
3. Architecture Representation 16
   3.1 Architecture Style to Implement in the Project 16
   3.2 Subsystem or Module Decomposition View 18
   3.3 Updated System Backlog 21
4. System Views 21
   4.1 Logical View 22
      4.1.1 Interface Layer (User Interface Design) 22
      4.1.2 Business Layer 23
      4.1.3 Data Layer 26
   4.2 Development or Implementation View 28
   4.3 Process View 29
   4.4 Physical View 32
   4.5 Scenario View 33
5. Annexes 35

System Architecture

1. Introduction

This CRNube StoreIT System Architecture document aims to provide a structured and detailed description of the technical environment in which the solution was developed and implemented. Throughout its sections, it addresses the conceptual elements, design artifacts, and technological decisions that underpin the project. In this way, it provides a clear view of the components, standards, and workflows agreed upon by the development team and CRNube.

1.1 Purpose

The purpose of this document is to summarize the system architecture.

1.2 Scope of the Architecture Document

Identify the technical elements required for the comprehensive development of the software project called CRNube StoreIT, which will be implemented at the company CRNube.

1.3 Related Documents

The following documents were used as a basis and reference for the preparation of this System Architecture document:

Code	Description	Annex
IS-012	Documento Visión del Proyecto	
IS-014	Documento Especificación de Requerimientos de Software	
12-2025	Minuta 12-2025 30-07-25	
13-2025	Minuta 13-2025 13-08-25	

2. Considerations and Limitations

This section presents the technical requirements and special conditions that will be considered in the development and implementation of the System. Non-functional requirements related to execution, availability, fault tolerance, integrity, etc., are addressed.

2.1 Technical Platform

This project was implemented on a technical platform composed of technologies and tools that meet the previously established non-functional requirements of compatibility, scalability, security, and free software nature.

- Programming Languages
    - PHP for the development of the web application.
    - Python for the development of the desktop application and part of its respective installer.

- Frameworks, Libraries, or Systems
    - Inno Setup for the desktop application installer.
    - PySide6 for the graphical interface of the desktop application.

- Database Management Systems (DBMS)
    - MariaDB for the web application, ensuring compatibility and scalability.
    - SQLite for the desktop application, optimizing local storage and portability.

- Development Environments (IDE)
    - Visual Studio Code for PHP and Python
    - phpMyAdmin for MariaDB and DB Browser for SQLite

- Additional Tools and Utilities
    - Git and GitHub for version control and open source collaboration.

2.2 Portability

The system is designed to be accessible and functional across various platforms, which include the following:

- Web application: Is accessible through web browsers, such as: Google Chrome, Microsoft Edge, and Mozilla Firefox.
- Desktop application: Is available for computers running the Windows 11 operating system.
- Installer: Is distributed as an executable compatible with Windows 11.

2.3 Security

One of the main focuses of this application is to make it quite robust in terms of security; therefore, the following security policies and mechanisms were developed.

2.3.1 Password Composition and Assignment Controls (Security Policies).

The system security policies will cover different aspects to ensure high confidentiality in the files backed up by the system, among which are:

- **Two-Factor Authentication:** All system users must enable two-factor authentication with Google Authenticator, Microsoft Authenticator, or another compatible app to access the web application's features, without which they will also not be able to access the desktop application.

- **IP Authentication:** Only IP addresses that the user has registered as valid through the web application will be able to access the desktop application.

- **Password Format:** Every password must contain a minimum of 8 characters, including at least one uppercase letter, one lowercase letter, one number, and one special character. Passwords cannot exceed 30 characters.

- **Password Renewal Period:** The system superadministrator has the authority to choose the expiration date for their own password and the passwords of each system user individually. The available expiration options are: after 30, 60, or a

maximum of 90 days. The superadministrator may also allow a password to never expire indefinitely, in order to leave it to their judgment to weigh the importance of security against the inconvenience or risk of changing passwords regularly.

All these aspects were discussed and approved on multiple occasions in meetings with the company, receiving feedback from the Product Owner, the Technical Lead, and the Expert User on aspects such as security and the convenience of the proposed policies for users.

2.3.2 New User Registration

The system to be developed will have three types of users, each with their own set of privileges, which will be described below:

- **Basic User**
    - Synchronize files from the cloud to the local folder.
    - Synchronize files from the local folder to the cloud.
    - View from the web application which files are backed up in the cloud.
    - Reset their password in case of forgetting or expiration.
    - Register, delete, and view their authorized IPv4 addresses for accessing the desktop application.
    - Download backed-up files from the web application.
    - View information about their capacity and consumption level of the storage quota in the hosting.
    - Check which files were last synchronized from the desktop app.
    - Temporarily pause and resume the synchronization of a file within the backed-up folder.
    - Configure and reset their own multi-factor authentication.
    - Maintain and view a log of actions performed on their backed-up files.

- **Administrator**

  Has a higher level of access than a basic user; therefore, they have all the privileges of a basic user plus the following:

  - Create accounts for new users.
  - Disable accounts of users with a lower authorization level than their own.
  - View all existing system users (except the superadministrator), their assigned storage quota, and their consumed storage quota.
  - Assign and edit the storage quota granted to each user with a role equal to or lower than their own.

- **Superadministrator**

  This is the most powerful user in the system and has all the administrator privileges plus the following:

  - Configure the initial application parameters (database credentials, Mailjet, and hCaptcha)
  - Delete and reactivate (enable) system users.
  - Configure the possible expiration of passwords for all registered system users.
  - Maintain and view a log of the history of authorized IP addresses for each user.
  - View backed-up files belonging to all existing users.
  - Download the backed-up files of any user.

2.3.3 User and System Component Access Control.

The main features accessible to each user according to their role are detailed below:

User Role / *Feature*	*Superadministrator*	*Administrator*	*Basic User*
Web Configuration Management Module in the Hosting	X		
Module for Managing the Installation and Configuration of CRNube-StoreIT on the Desktop	X	X	X
Desktop Synchronization Management Module	X	X	X
Web Session Management Module	X	X	X
Local Session Management Module	X	X	X
IP Management Module	X	X	X
Account Configuration Module	X	X	X
Information Display Module	X	X	X
Synchronize files from the web application	X	X	X
View and download own files	X	X	X

View and download files from other users	X		
Create additional user accounts	X	X	
Edit granted storage quotas	X	X	
Disable user accounts	X	X	
Enable user accounts	X		
Delete user accounts	X		
Configure password expiration	X		

Additionally, the main interfaces accessible to each type of user according to their role are detailed below:

User Role / *Feature*	*Superadministrator*	*Administrator*	*Basic User*
Login (Web)	X	X	X
My Files (Web)	X	X	X
Manage IPs (Web)	X	X	X

Account Configuration (Web)	X	X	X
Register Current IP (Web)	X	X	X
Manage Users (Web)	X	X	
Create Account (Web)	X	X	
Everyone's IPs (Web)	X		
Everyone's Files (Web)	X		
Login (Desktop)	X	X	X
Backed up Files (Desktop)	X	X	X
Storage State (Desktop)	X	X	X
Synchronization Log (Desktop)	X	X	X

2.3.4 Transaction Logs

At CRNubeStoreIT, logs are not implemented as a corporate internal control but as self-management tools so users can audit and debug their own operations. The logs that the system generates and stores, both in the desktop application and on the web, are described below:

- **Log of Actions Performed on Files in the Desktop Application:**
  - Available for all user types/roles.
  - Records every operation of creating, modifying, deleting, renaming, restoring, pausing or resuming synchronization of a file, as well as if there was an error.
  - Stores the date, time, and type of operation, linked to the corresponding file metadata through a foreign key in the tbl_FileLogs table pointing to tbl_FileMetadata.
  - Stored in the local SQLite database of the desktop application; records with synchronization or restauration operations and errors can be viewed at the interface level.
  - Its purpose is to allow the user to review step by step what changes have been applied to their files and diagnose possible synchronization errors.

- **Log of IP Addresses Registered in the Web Application:**
  - Only available for the Superadministrator at an application level.
  - Stores all IPs associated with each user, even if the IP or user was deleted. No additional table is required; tbl_IpAddresses is used.
  - When viewed, the following are displayed: registration date, username (derived from the user ID, foreign key), and IP status (active, deleted, or blocked).
  - Stored in the MariaDB database of the web application and can be viewed at the interface level.
  - Its purpose is to allow the superadministrator to view this history, with search options included, in order to maintain control over the IPs that have access to the system.
- **Log of Files Synchronized from the Desktop Application:**
  - Only available for the administrators of the hosting where the web application is installed.

- Stores all synchronization actions from the client to the server. The tbl_SyncEvents table is used to store this data.
- For each record, stores a reference to the associated user, file, and device, as well as the event type, its date and time, among others.
- Stored in the MariaDB database of the web application and not visible at the application interface level.
- Its purpose is to maintain a record of synchronizations performed from the desktop application to enable synchronization from the web application.

Note: During the requirements gathering phase, the inclusion of a log for operations on user accounts (creation, disabling, deletion, enabling, quota editing, etc.) was also evaluated. Ultimately, this was defined as an optional functional requirement and will only be implemented if schedules and additional resources allow.

2.4 Reliability/Availability

The architecture of the CRNube StoreIT system is designed to ensure high levels of reliability and availability, especially considering that it is a file backup solution for professionals and small businesses.

The system must be continuously available to users, allowing access to their backed-up files at any time. To this end, the following aspects have been considered:

- Web application: Will be hosted on hosting services that support PHP and MariaDB, with constant monitoring and periodic backup of the system configuration.
- Desktop application: Designed to operate autonomously and synchronize with the cloud when a connection is available, allowing some local operations even during periods of disconnection.
- Automatic synchronization: The architecture includes synchronization mechanisms that are triggered upon detecting file changes, reducing dependency on manual actions and improving the availability of up-to-date data.

System reliability is ensured through:

- Strict validations at login, including multi-factor authentication (2FA) and control of authorized IP addresses.
- Logs that record all relevant actions on local files and IP address records, enabling auditing and recovery from failures.
- Progressive IP blocking for failed attempts in the desktop application, to prevent unauthorized access and protect system integrity.

These elements, together with a modular and scalable architecture, allow CRNube StoreIT to maintain predictable, secure, and available behavior at all times, even under adverse conditions or human errors.

2.5 Performance

2.5.1 Performance Requirements

It is complex to set definitive performance metrics, as much of the behavior will depend on the hosting and infrastructure each user chooses. However, the system will be designed to maximize code efficiency in critical operations (e.g., file synchronization) and to ensure that any change propagates in a time that does not negatively affect the user experience.

2.5.2 Transaction Volume

Being Open Source software, the actual transaction load will vary depending on the use and the context in which the application is used. For example, an independent user could generate dozens of backup operations daily, while in a small office with multiple workstations, the monthly volume could reach hundreds or thousands of synchronizations. Since this figure directly depends on usage and the hosting environment, no concrete numerical value is defined at this stage. However, the architecture aims to adapt to different demand levels without requiring code changes.

3. Architecture Representation

3.1 Architecture Style to Implement in the Project

The architecture of the complete system is Client-Server, which organizes the system into two main roles: one or more clients (send requests to the server and wait for a response) and a server (hosts and provides some service or resource). **In this case, the web application acts as the central server** that exposes services and maintains a database with information of all users, while **the local application fulfills the client role,** as it invoke the server's REST APIs to log in, obtain storage information, among others. This facilitates centralized security control and, at the same time, allows multiple different clients to interact uniformly (same rules and interfaces) with the hosting. Additionally, internally, both have a different architecture:

Internal Architecture of the Web Application: N-Tiers

For the web application, an N-Tiers architecture is used, where each layer provides services to the upper layer and consumes services from the lower layer. In addition, each layer is isolated, which enhances security, a central element within this system. Furthermore, by separating responsibilities into different layers, it facilitates maintenance and testing. The architecture of each layer is broken down below:

1. **Presentation Layer (Model View Controller + REST)**
    - **MVC** separates the presentation (View) from the logic (Model) and uses a Controller to manage user interactions, updating the model and view appropriately. On the other hand, **REST** views everything as a resource with a URL, operated through standard HTTP methods (GET, POST, etc.). Modern web applications use RESTful APIs for their simplicity and scalability.
    - **Justification:** Combining MVC and REST improves code organization clarity and facilitates testing and maintenance. Additionally, providing a uniform API allows different clients to consume the same features without duplicating logic (located in the business layer, as Controllers delegate tasks related to external requests to the lower layer). It also reduces coupling in the frontend, facilitates the assignment of parallel work roles (interface design and logic) and improves the ability to modify the interface without touching the core logic.

2. **Business Layer**
    - A Service handles the business logic in a centralized manner, ensuring that business rules are applied consistently and reusably, regardless of whether the request comes from the web, the API, or an internal process. It is also the appropriate layer for controlling transactions and data integrity without mixing it with presentation, as well as coordinating calls to the lower layer's repositories.

3. **Data Layer (Database-Centered)**
    - **The database-centered architecture** consists of different components that communicate through shared data repositories and are relatively independent, as they interact only through the data store (MariaDB database).
    - **Justification:** Ensures consistency among the operations performed by the different modules (user management, IP management, etc.), facilitates the application of integrity rules, and supports auditing and traceability, as all information is stored and retrieved from the same database, ensuring that the most current and correct version is always used.

Internal Architecture of the Desktop Application

- **Event-Driven:**
    - **In the Event-Driven architecture** components communicate through asynchronously distributed events. It consists of event/signal emitters that publish messages, and subscribers that receive and handle them.
    - **Justification:** It is ideal for file synchronization, as it immediately detects changes in the backed-up folder and generates events such as “file modified”, “pause activated”, or “synchronization complete”. These events trigger actions such as updating the log or executing synchronization, which optimizes the system’s response, especially during periods of high activity.

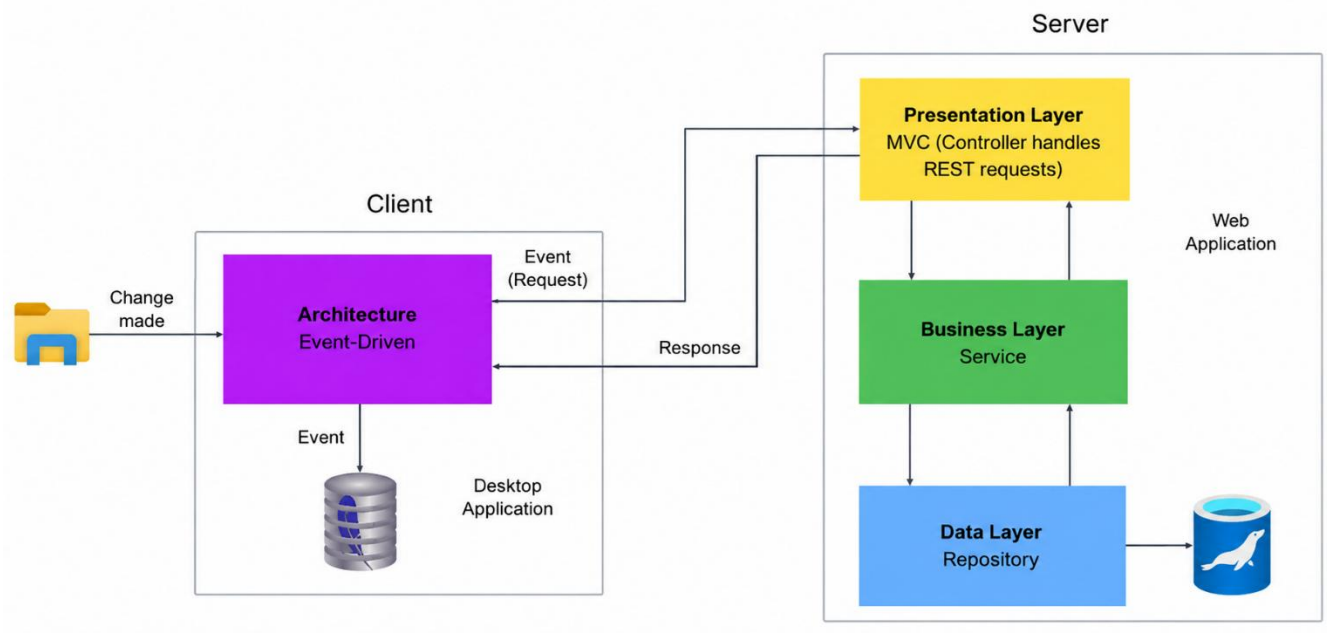

Figure 1. Architecture style diagram for CRNube StoreIT

3.2 Subsystem or Module Decomposition View

This view presents the functional decomposition of the CRNube StoreIT system into modules visible from the end user's perspective. Two separate diagrams are included: one showing the modules that make up the **web application** and another showing the modules of the **desktop application**.

Each module represents a capability or functional area, and arrows indicate dependencies or logical flows between said modules within each system:

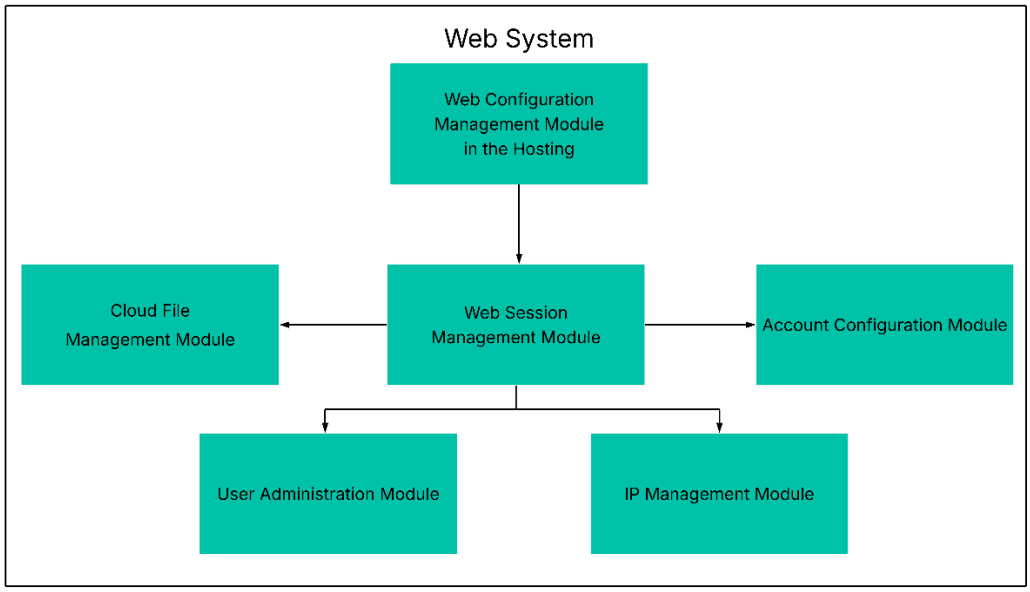

Figure 2. Module view of the web application

The following describes the different modules into which the web application is divided:

Web Configuration Management Module in the Hosting:

It includes the installation and configuration of the web application on the hosting, configuring the database credentials, Mailjet and hCaptcha, and creating the superadministrator.

Web Session Management Module:

The module allows logging in and out. It also allows blocking the user's IP address if several failed login attempts are made. Additionally, it is responsible for validating the passwords and 2FA tokens required to access the application.

IP Management Module:

The module allows registering, viewing, searching, and deleting IPs in the system. Additionally, in the case of the superadministrator, it is possible to view every IP address that has been registered in the app, even after its deletion.

User Administration Module:

The module allows creating, deleting, disabling, enabling, searching, and viewing accounts of created users. It also includes the editing of storage quotas and the configuration indicating when the password expires.

Cloud File Management Module:

The module allows you to search for, view, and download backed-up files (synchronized from the desktop application). Additionally, the super administrator can view and download files for all users on the system.

Account Configuration Module:

This module allows configuring the multi-factor authenticator, creating a new password, and activating a user account via a link sent to their email. It also allows resetting the password and the 2FA configuration.

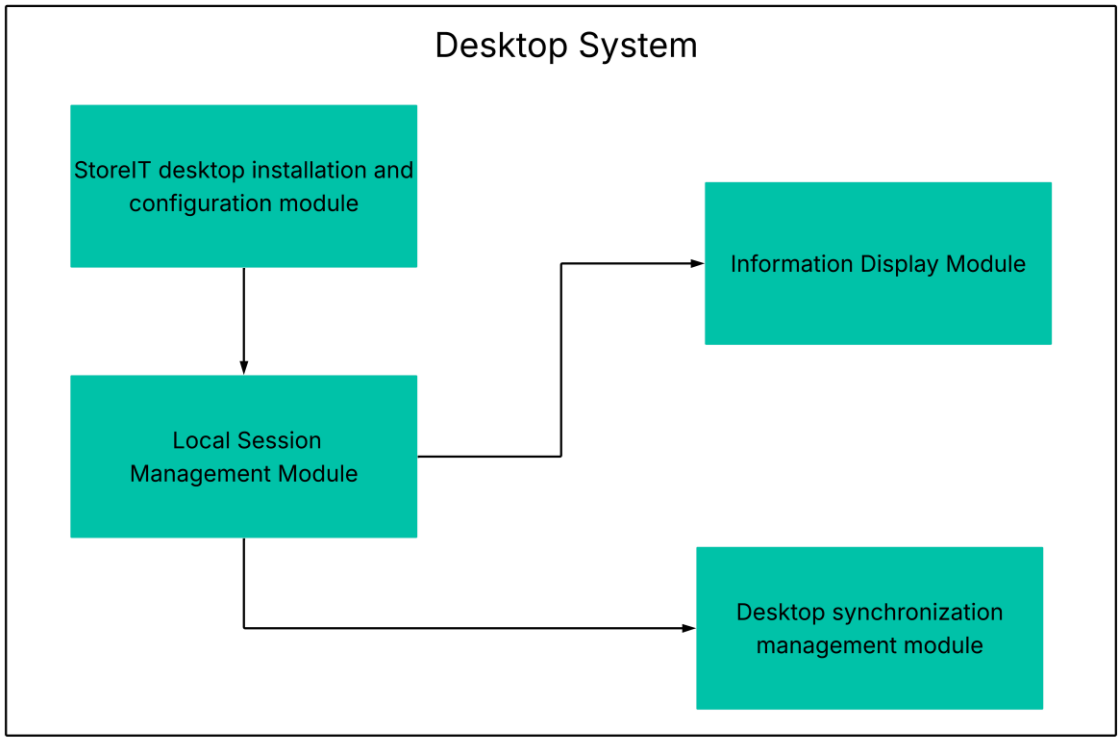

Figure 3. Module view of the desktop application

A continuación, se describen los distintos módulos en los que se dividen la desktop application:

StoreIT desktop installation and configuration module:

This module allows you to install the desktop application, verifying the hosting to be used (that it is compatible and has the web application installed). It also includes the selection of the folder to be backed up.

Local Session Management Module:

The module allows users to log in and out. It also validates that, even if a user's credentials are correct, they are only allowed access if they are the user who performed the initial installation and configuration of the local system.

Desktop synchronization management module:

Allows synchronizing files to the hosting. It also includes the functionality to pause and resume synchronization of a specific file and maintains a local log of actions performed on them (synchronization, pause, resume, etc.).

Information Display Module:

This module allows the user to view the status of their storage and the last 100 synchronization attempts, indicating, for example, whether their synchronization was successful or not.

3.3 Updated System Backlog

Currently, the system backlog is empty as the development of all the functionalities initially planned for the applications that make up CRNube StoreIT has been completed.

4. System Views

This section describes the most significant parts of the architecture in the design model, using Kruchten's 4+1 model as a basis: Logical view, deployment (or development) view, process view, physical view, and scenarios.

4.1 Logical View

This view represents the functionality that the system will provide to end users. It includes the design structure, main classes or modules, and relationships – all focused on functionality. This view will be divided into the three sections dictated by Layered Programming.

4.1.1 Interface Layer (User Interface Design)

The following diagrams present the user interface design of **CRNube StoreIT** in two areas: the **web application** and the **desktop application**. Each diagram shows the main screens and the navigation routes or flows between them:

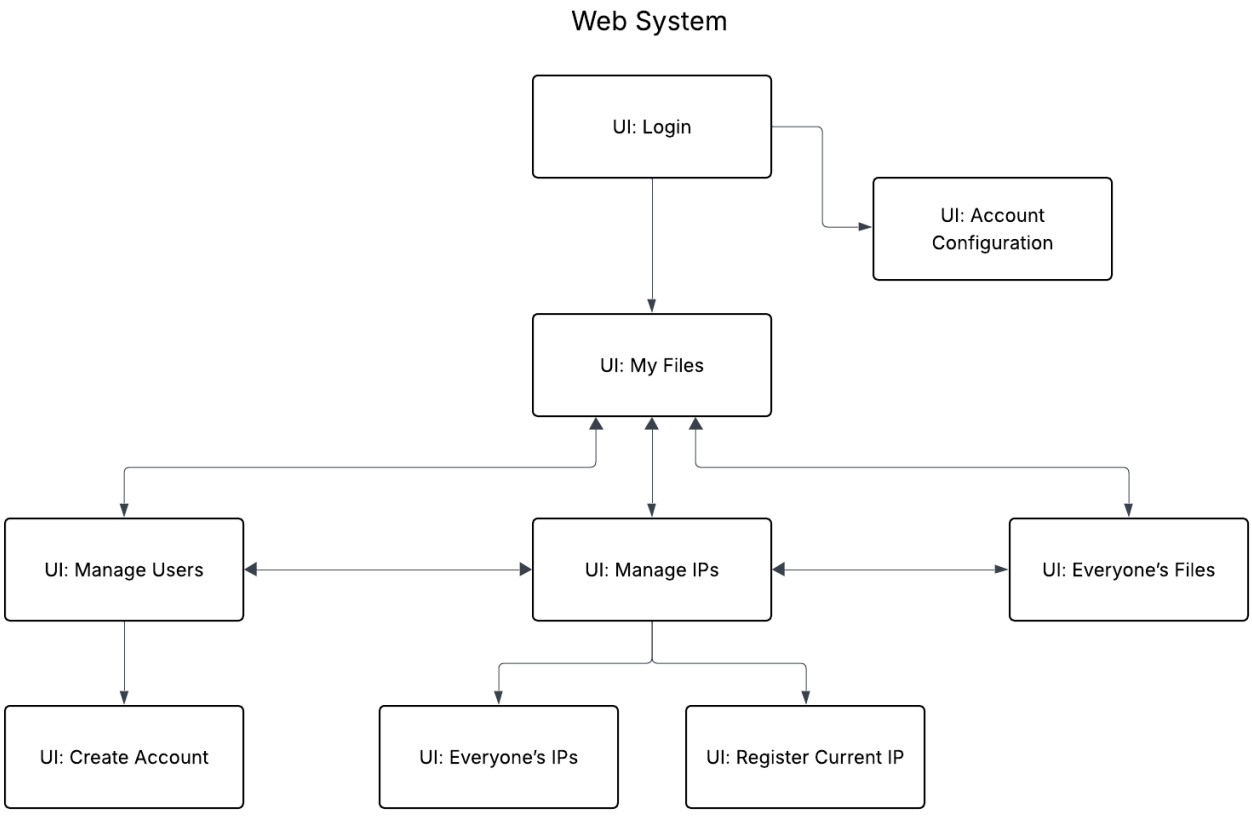

Figure 4. Interface layer of the web application

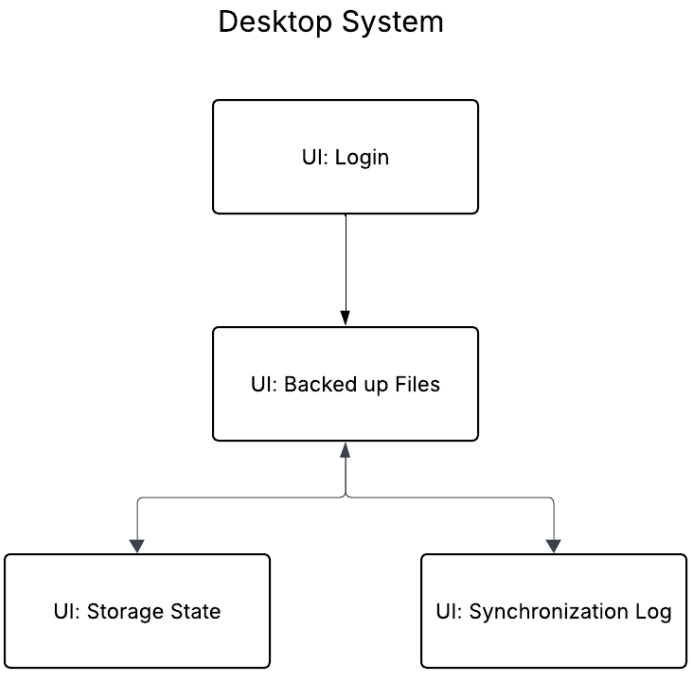

Figure 5. Interface layer of the desktop application

4.1.2 Business Layer

Web application:

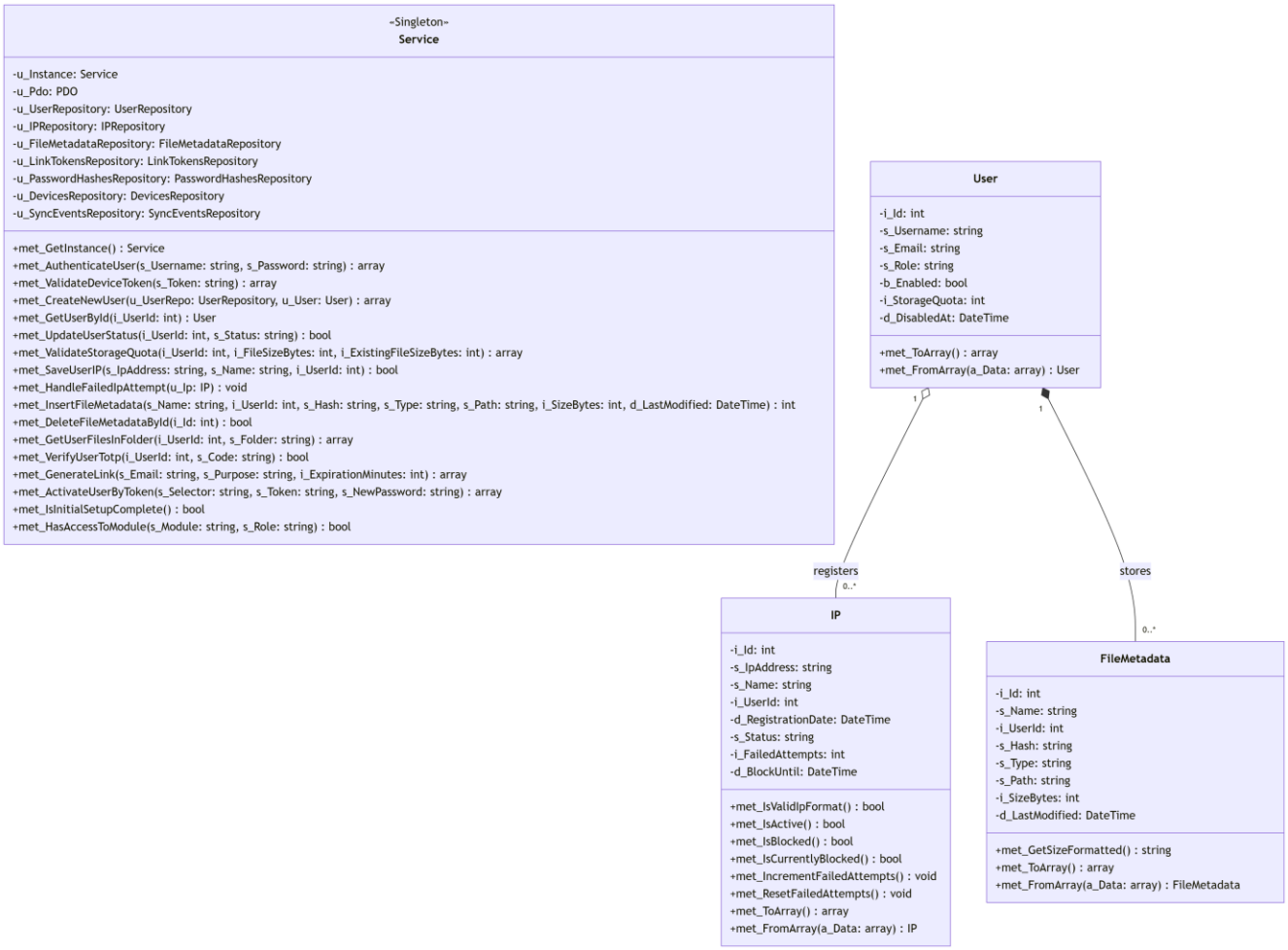

Figure 6. Business Layer de la web application

Desktop application:

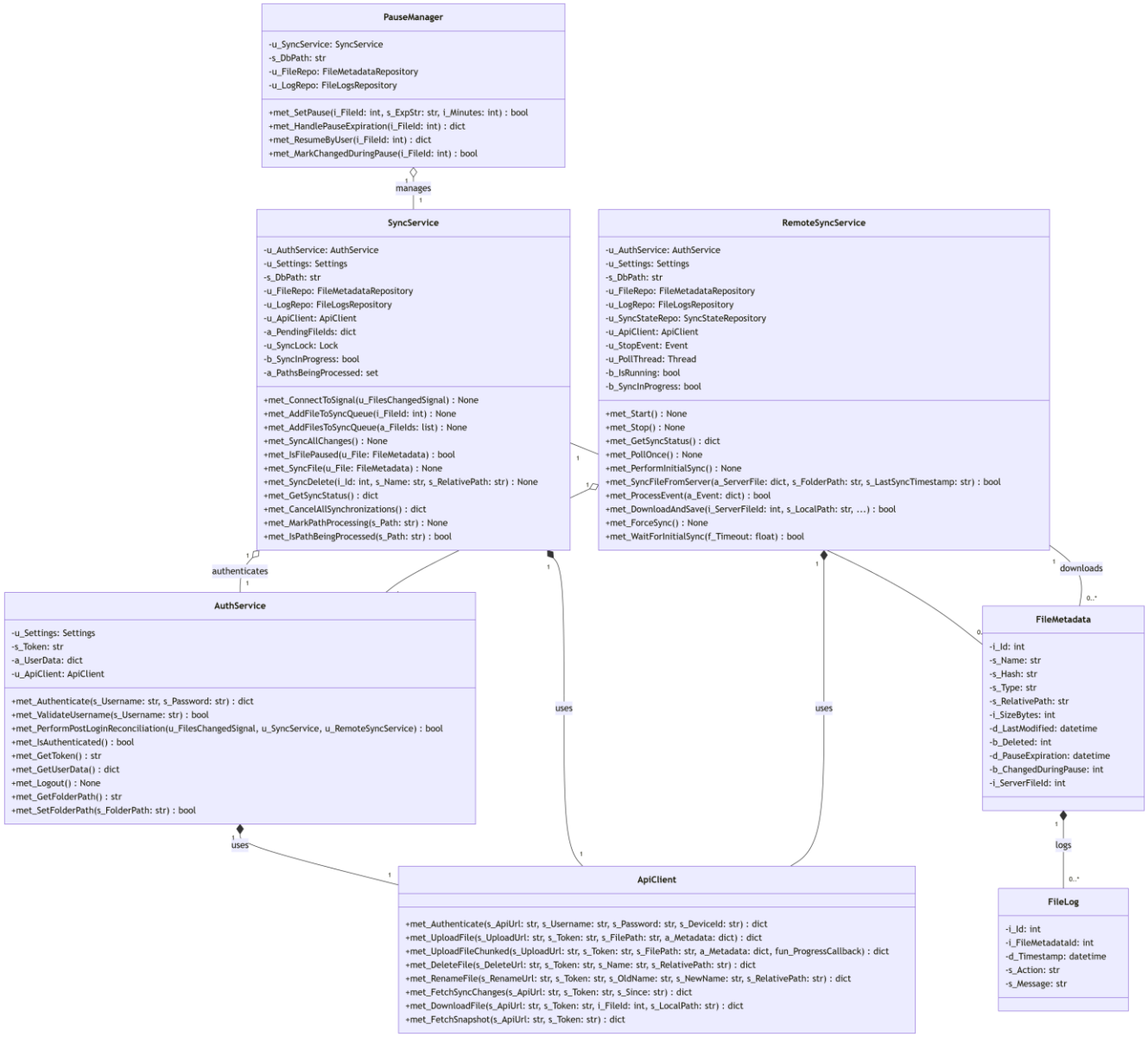

Figure 7. Business Layer de la desktop application

Note: Constructors, getters, and setters have been omitted; only entities and Services are included, for which the most important methods were selected because there are too many and including all of them would affect the readability of the diagram.

4.1.3 Data Layer

Web application:

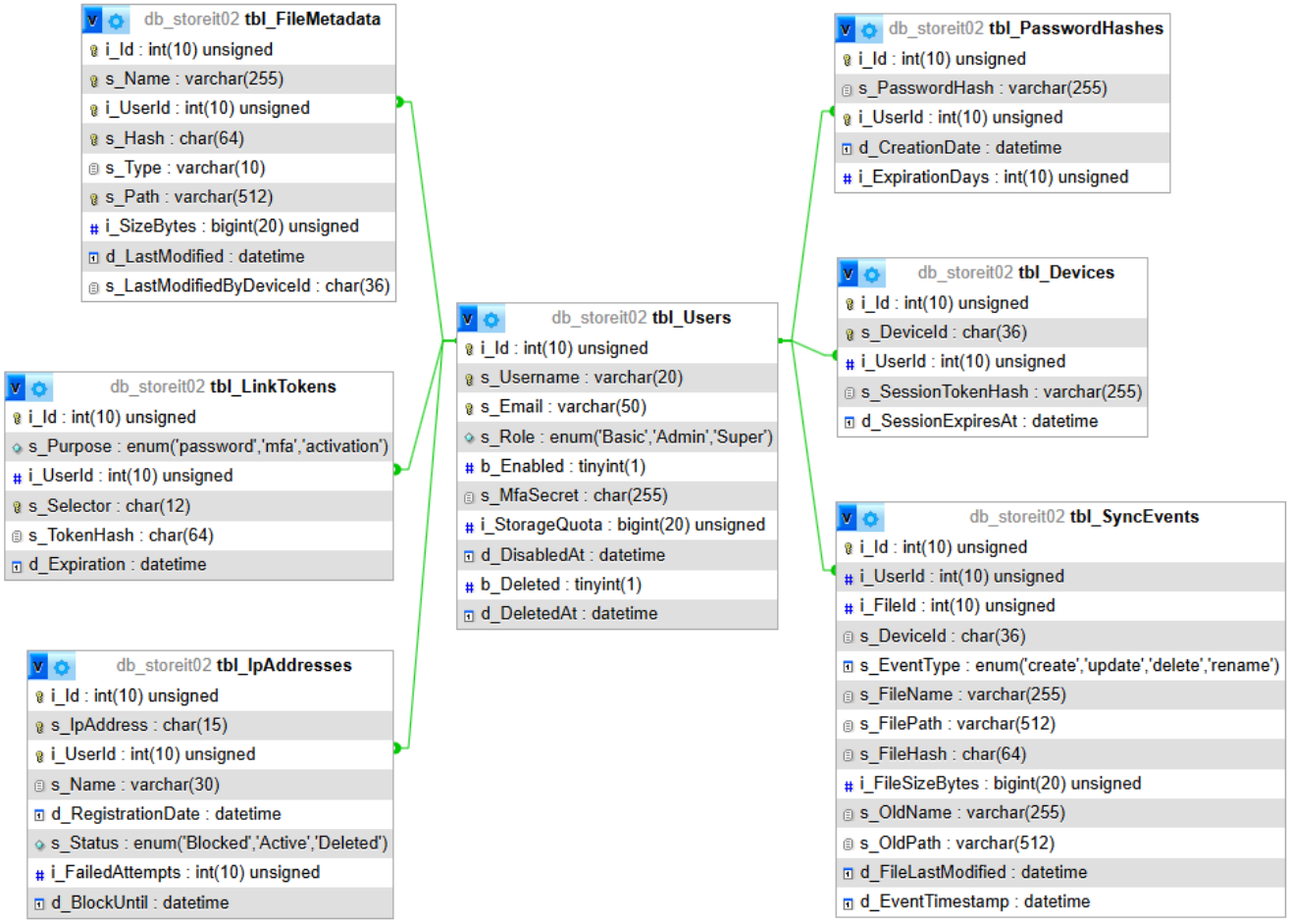

Figure 8. Data layer of the web application

Desktop application:

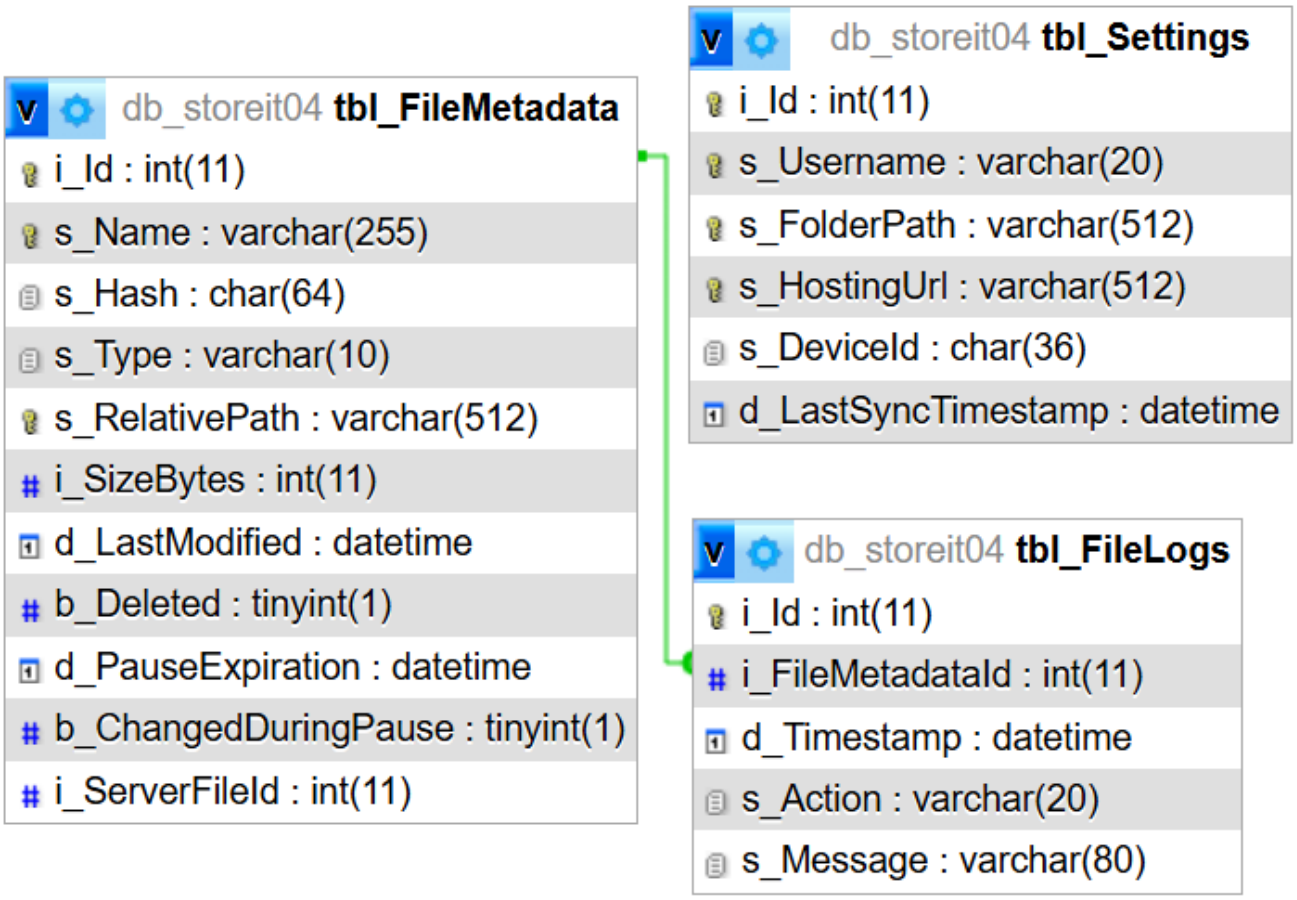

Figure 9. Data layer of the desktop application

4.2 Development or Implementation View

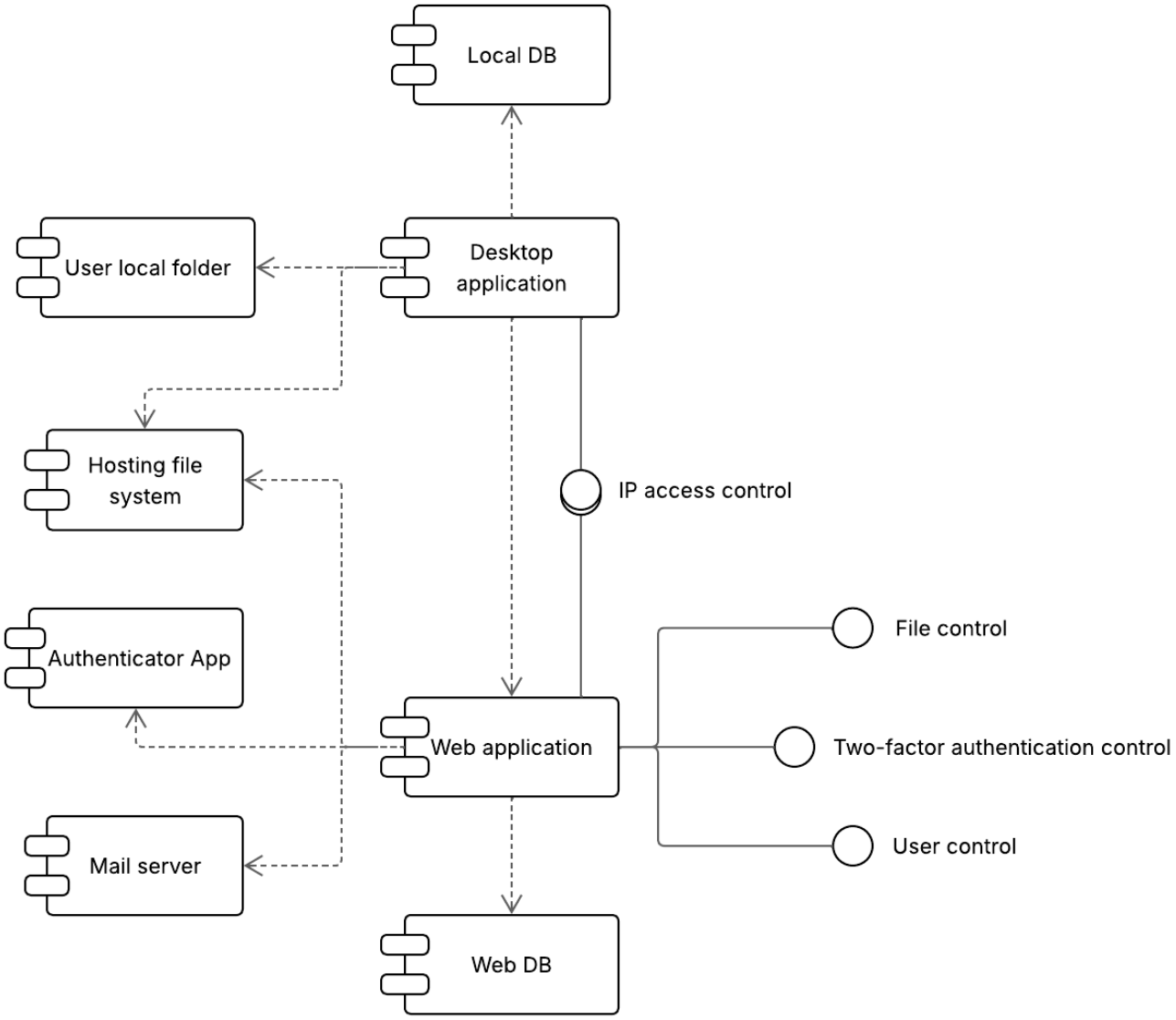

Figure 10. System Component Diagram

4.3 Process View

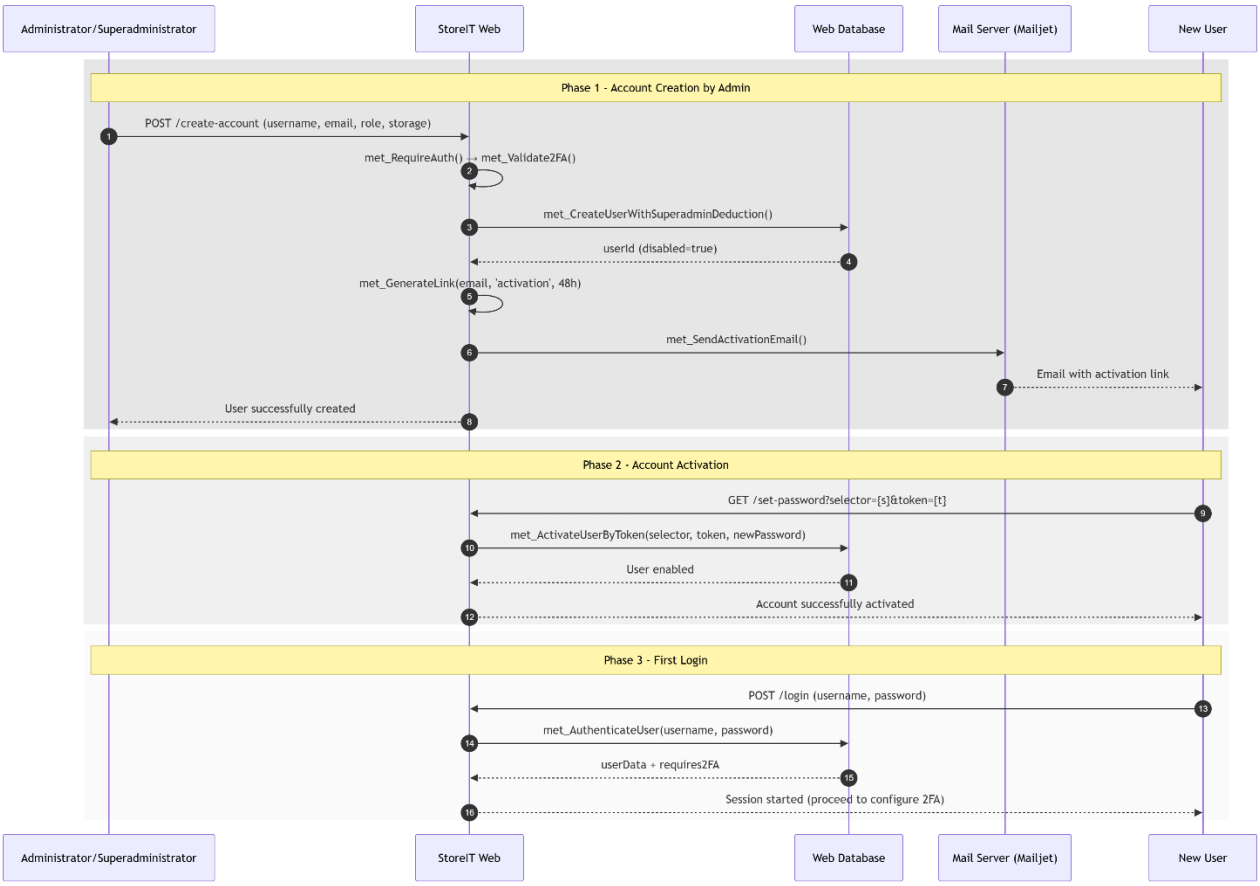

Figure 11. System Sequence Diagrams: User account creation and activation.

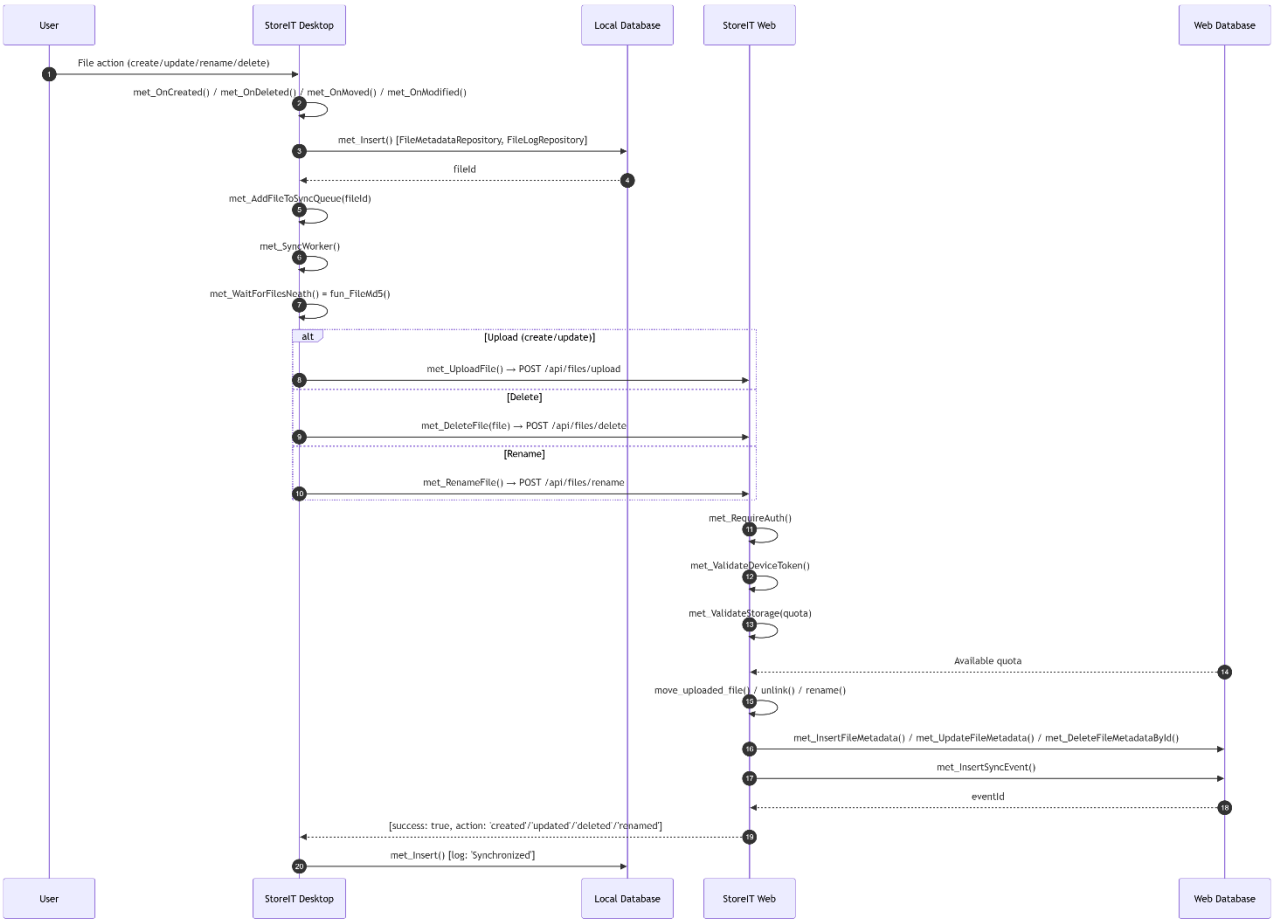

Figure 12. System Sequence Diagram: File synchronization from StoreIT Desktop to StoreIT Web

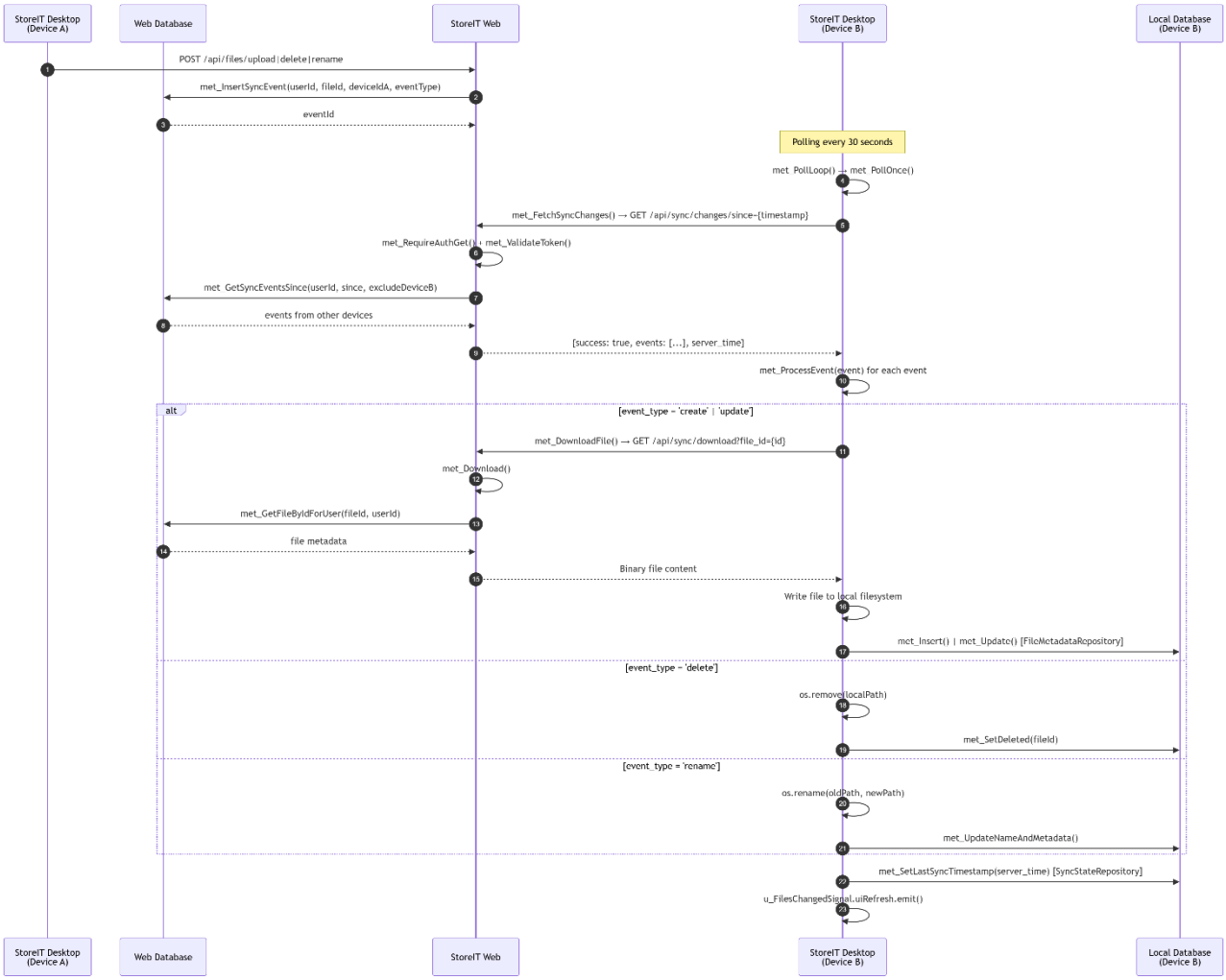

Figure 13. System Sequence Diagram: File synchronization from StoreIT Web to StoreIT Desktop

Note: These diagrams are a broad representation of the main flows the system carries out. Parameter names may not be 100% accurate, as they are simplified to make the diagram more readable.

4.4 Physical View

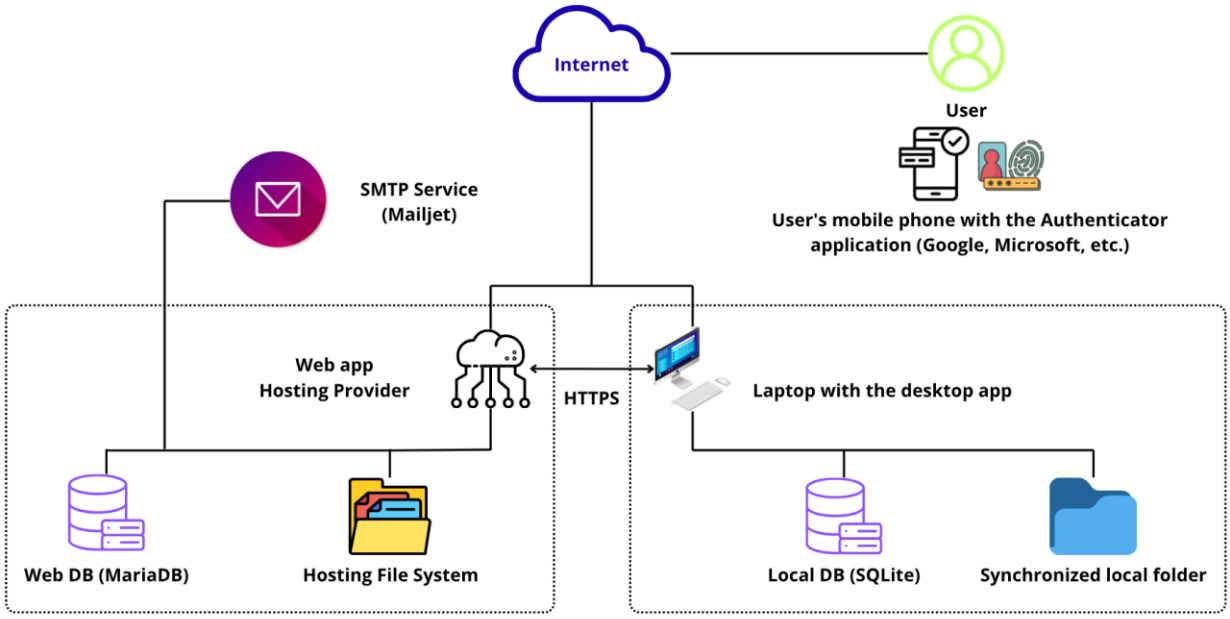

Figure 14. Deployment Diagram

The desktop application (Windows 11) communicates via HTTPS with the web application (PHP) deployed on the hosting provider, which in turn accesses MariaDB and the hosting file system. The SMTP service (via Mailjet) will be used for transactional emails. The 2FA TOTP is generated offline on the user's phone and validated on the backend.

4.5 Scenario View

Use Case Diagram for the Desktop Application:

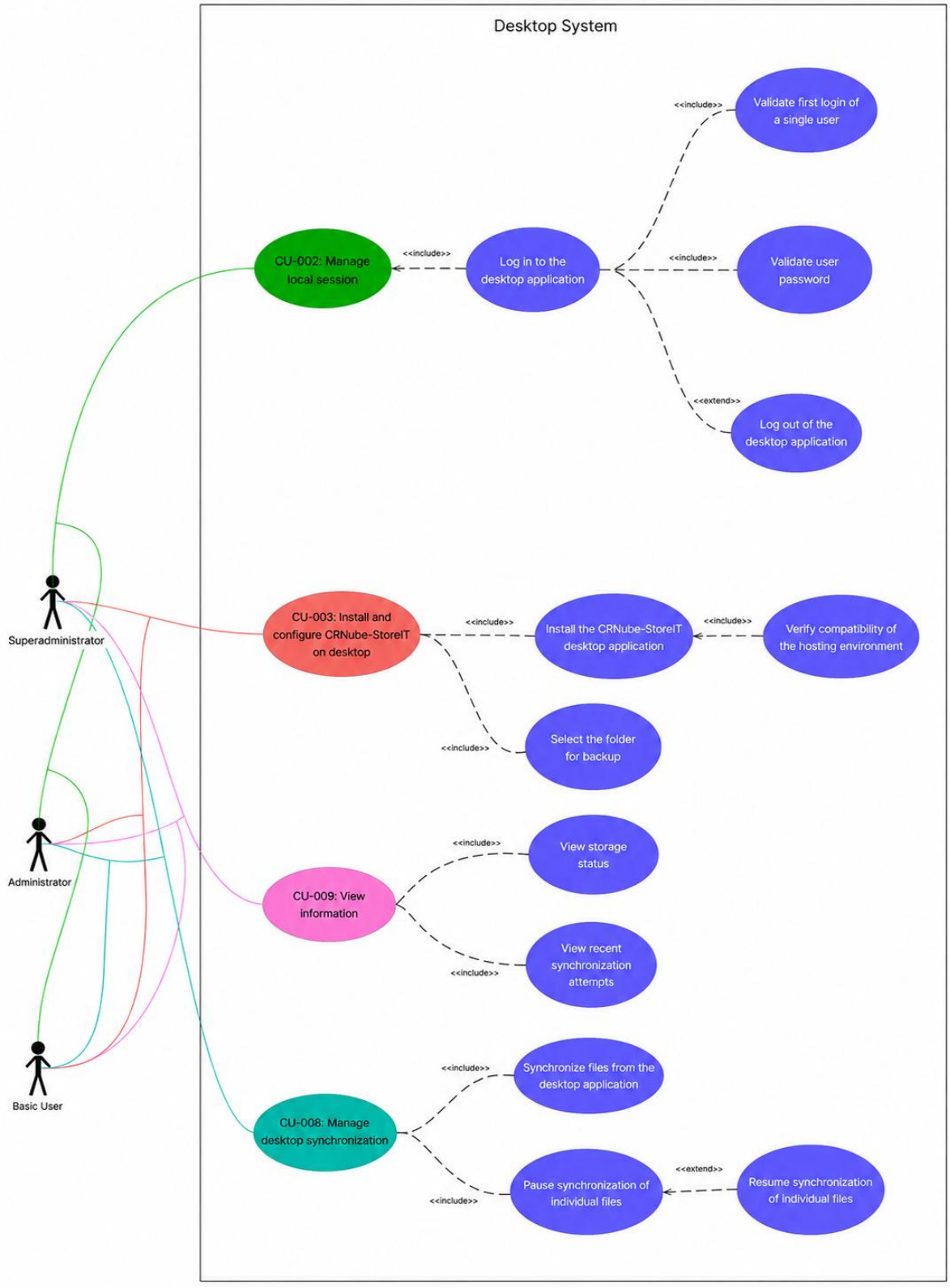

Figure 15. Use Case Diagram for the Desktop Application

Use Case Diagram for the Web Application:

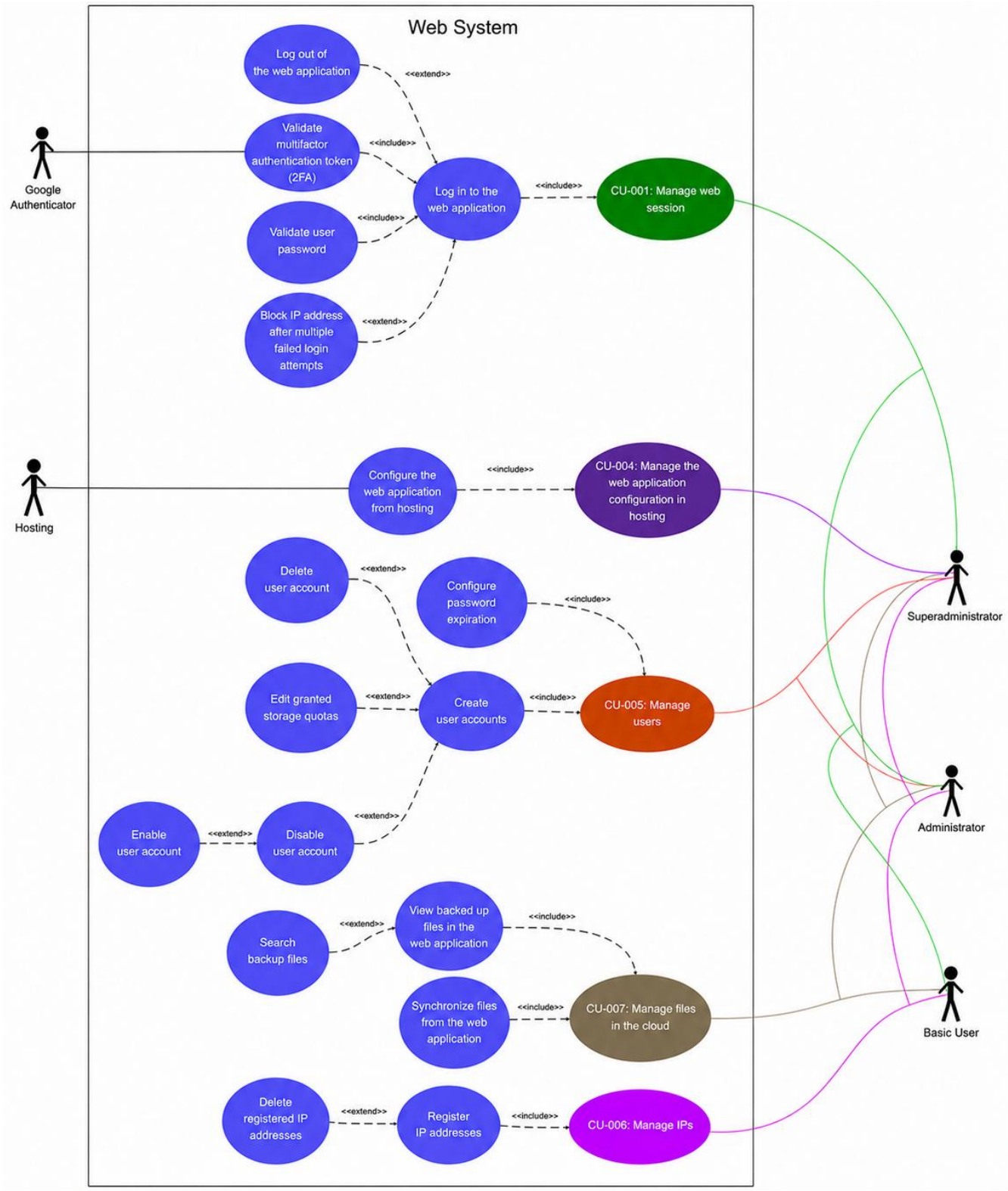

Figure 16. Use Case Diagram for the Web Application

5. Annexes