

Dedication

With valiant heart nothing is impossible
An easy conscience all is accessible
When there is thirst to learn
Good things come to those who wait
When there is the concern of carrying out an intention
All becomes easy to arrive at our ends
In spite of the obstacles which are opposed
In spite of the difficulties which interpose

The studies are before all
Our single and only asset
They represent the light of our existence
The brilliant star of our rejoicing
Hoping for epic following days
A glorious and magic future
Wishing that the fruit of our furnished efforts
Day and night, will carry out us towards flowered happiness

Today, gathered here near the jury's,
Will make sign of perseverance
And whom we would be enchanted
By our honoured work

Moez Kchaou

Acknowledgments

I make a point of thanking and of testifying all my recognition with the following people, who allowed me to live this experiment enriching and full with interest on the professional plan and staff during these months in the town of Sfax.

My thanks are addressed initially to any person who contributed for my success during this internship those of my family, specifically my FATHER, my MOTHER and my BROTHER .

In particular for the whole of knowledge (as well technical as general) as I could acquire throughout my trainings at Private Polytechnic Institute of Advanced Sciences of Sfax "IPSAS".

I would like to thank "Mr.Ahmed Kharrat", Assistant Professor at Polytechnic Institute of Advanced Sciences of Sfax "IPSAS", for confidence that he granted to me by directing my graduation project. I express my sincere thanks for his invaluable advices and time that he devoted me.

I make a point of thanking particularly Mr Yassin guirat Director company znet-it to have accommodated me within their company, to have trusted to me for the realization of their business application of the insurances and other tasks which were entrusted to me.

Finally, I thank the entire administrative body and professors of Polytechnic Institute of Advanced Sciences of Sfax "IPSAS" for their availability and continuous help for the realization of my project.

Table of contents

General Introduction.....	
CHAPTER 1: General Context.....	1
I. INTRODUCTION.....	1
II. PROJECT FRAMEWORK.....	1
II.1 PRESENTATION Of ZNET-IT.....	1
II.2 MISSION.....	2
II.3 BUSINESS AREAS.....	2
III. ANALYSIS OF THE EXISTING.....	2
III.1 Study of the existing.....	2
III.2 Criticism of the existing.....	3
III.3 Suggested Solutions.....	5
IV. OBJECTIFS TO REACH.....	6
V. ESTIMATED PLANNING:.....	7
VI. CONCLUSION:.....	7
CHAPTER 2: Analysis and specification of requirements.....	8
I. INTRODUCTION.....	8
II. REQUIREMENT ANALYSIS.....	8
II.1 ACTORS IDENTIFICATION.....	8
II.2 Functional Requirements.....	9
II.3 Non-Functional Requirements.....	10
III. CHOICE OF DESIGN METHODOLOGY AND JUSTIFICATION.....	11
IV. APPLICATION CONCEPTION.....	11
IV.1 Use Case Diagrams.....	11
IV.2 Use case specification.....	12
IV.2.1 "Authentication" use case diagram.....	12
IV.2.2 Description "Authentication".....	12
IV.2.3 "Manage users" use case diagram.....	13
IV.2.4 Textual description of "Manage users" use case.....	14
IV.2.5 "Manage documents " use case diagram.....	18
IV.2.6Textual description of " Manage documents" use case.....	18
V. SEQUENCE DIAGRAM.....	25

V.1 "Authentication" use case sequence diagram.....	26
V.2 "Add User" use case sequence diagram	27
V.3 "update User" use case sequence diagram.....	28
V.4 "Delete User" use case sequence diagram	29
V.5 "Consult User" use case sequence diagram	30
V.6 "Add document" use case sequence diagram.....	31
V.7 "update document" use case sequence diagram.....	32
V.8 "search document" use case sequence diagram.....	33
V.9 "Add folder" use case sequence diagram	34
V.10 "Delete folder" use case sequence diagram.....	35
V.11 "Move to folders" use case sequence diagram.....	36
V.12 "update folder" use case sequence diagram.....	37
VI. CONCEPTUAL DATA MODELING.....	38
VI.1 Data Dictionary.....	38
VI.2 Classes Representation.....	41
VI.3 Associations Representation.....	43
VI.4 Class Diagram.....	44
VII. ACTIVITY DIAGRAM.....	46
VII.1 System activity diagram.....	46
VII.1.1 Definition of system activity diagram.....	46
VII.1.2 Presentation of activity diagram of system.....	47
VII.1.2.1 Activity diagram use case system "Authentication".....	47
VII.1.2.2 use case "Manage User"	48
VII.1.2.2.1 use case system activity diagram "Add User".....	48
VII.1.2.2.2 use case system activity diagram "update User".....	49
VII.1.2.2.3 use case system activity diagram "Delete User".....	50
VII.1.2.3 Use case "Manage document"	51
VII.1.2.3.1 Use case system activity diagram "Add document".....	51
VII.1.2.3.2 Use case system activity diagram "update document".....	52
VII.1.2.3.3 Use case system activity diagram "search document".....	53
VII.1.2.3.4 Use case system activity diagram "delete document".....	54
VII.1.2.3.5 Use case system activity diagram "Download document".....	55
VIII. CONCLUSION.....	55
CHAPTER 3: REALIZATION.....	56
I. INTRODUCTION.....	56
II. TECHNICAL STUDY.....	56

II.1 Realization Environment.....	56
II.1.1 HARDWARE ENVIRONMENT.....	56
II.1.2 .SOFTWARE AND DEVELOPMENT TOOLS ENVIRONMENT.....	57
II.2.Choice Of Architecture.....	63
III. PRODUCTION OF THE PROGRAM.....	67
III.1 Front-End.....	67
III.1.1 Authentication interface.....	67
III.1.2 Dashboard interface.....	69
III.1.3 List documents.....	69
III.1.4 add document.....	71
III.1.5 interface list folders.....	71
III.1.6 interface folders details.....	73
III.1.7 interface audit.....	74
III.1.8 List users and administrators.....	75
III.1.9 Add users with their role	76
III.1.10 interface Settings.....	77
III.2 Back-End.....	78
III.2.1 Adding google firebase dependency to spring boot (Maven).....	78
III.2.2 Intialization Firebase.....	78
III.2.3 File Service.....	79
III.2.4 FileController.....	80
III.3 Firebase.....	81
III.3.1 General Settings.....	81
III.3.2 Firestore Database.....	81
III.3.2.1 Collection "Files"	81
III.3.2.2 Collection "Folders"	82
III.3.2.3 Collection "Users"	82
III.4 Hosting application on railway.app.....	83
III.5 Extracting the application API from railway.app.....	83
IV. CONCLUSION.....	84
General Conclusion And Perspectives.....	85
Webography.....	87

List of figures

Figure 1 - logo of the company Znet-it.....	1
Figure 2 - Document Management System Work.....	3
Figure 3 - "Authentication" use case diagram.....	12
Figure 4 - "Manage users" use case diagram.....	14
Figure 5 "Manage documents " use case diagram	18
Figure 6 - "Authentication" sequence diagram.....	26
Figure 7 - "Add User" sequence diagram.....	27
Figure 8 - "update User" sequence diagram.....	28
Figure 9 - "Delete User" sequence diagram.....	29
Figure 10 - "Consult User" sequence diagram.....	30
Figure 11 - "Add document" sequence diagram	31
Figure 12 - "update document" sequence diagram.....	32
Figure 13 - "search document" sequence diagram	33
Figure 14 - "Add folder" sequence diagram	34
Figure 15 - "Delete folder" sequence diagram.....	35
Figure 16 - "Move to folder" sequence diagram.....	36
Figure 17 - sequence diagram "update folder".....	37
Figure 18 - Class Diagram.....	45
Figure 19- system activity diagram "Authentication"	47
Figure 20 - system activity diagram "Add User".....	48
Figure 21- system activity diagram "update User".....	49
Figure 22- system activity diagram "Delete User".....	50
Figure 23- system activity diagram "Add document".....	51
Figure 24 - system activity diagram "update document".....	52

Figure 25- system activity diagram "search document"	53
Figure 26- system activity diagram "delete document"	54
Figure 27- system activity diagram "Download document"	55
Figure 28 - Dart logo.....	57
Figure 29 - flutter logo.....	58
Figure 30 - spring boot logo.....	58
Figure 31 - Rest logo.....	59
Figure 32 - Firebase logo.....	59
Figure 33 - IntelliJ logo.....	60
Figure 34 - Android logo.....	60
Figure 35 - visual studio logo.....	61
Figure 36 - git logo.....	61
Figure 37 - github logo.....	62
Figure 38 - draw.io logo.....	62
Figure 39 - railway.app logo.....	62
Figure 40 - Application architecture.....	63
Figure 41 - mvc model principle.....	65
Figure 42 - interface authentication.....	68
Figure 43 - interface dashboard.....	71
Figure 44 - interface list documents.....	70
Figure 45 - interface add document.....	71
Figure 46 -interface List folder.....	72
Figure 47 - interface folder "IT"	73
Figure 48 - interface audit.....	74
Figure 49- interface list users.....	75
Figure 50 - interface add users.....	76

Figure 51 - interface Settings.....	77
Figure 52- dependency firebase in pom.xml.....	78
Figure 53 - FirebaseInitialization.java.....	78
Figure 54 - Method getFileDetails().....	79
Figure 55 - Method updateFile.....	79
Figure 56 - Method deleteFileByTittle.....	80
Figure 57 - FileController.java.....	80
Figure 58 - All details database flutter-dms.....	81
Figure 59 - Collection Files.....	81
Figure 60 - Collection Folder.....	82
Figure 61 - Collection Users.....	82
Figure 62 - hosting app in railway.app.....	83
Figure 63 - API application file management.....	83
Figure 64 - All data API application file management.....	84

List of tables

Table 1 -Gantt chart.....	7
Table 2 - The list of actors.....	8
Table 3 - Textual description of "Authentication"	12
Table 4 - Textual description of "User addition" use case.....	14
Table 5 - Textual description of "Edit user" use case.....	15
Table 6 - Textual description of "Consult user" use case.....	16
Table 7- Textual description of "Delete user" use case.....	17
Table 8- Textual description of "Add document" use case.....	19
Table 9- Textual description of "Edit document" use case.....	19
Table 10- Textual description of "Delete document" use case.....	20
Table 11-Textual description of "Search document" use case.....	21
Table 12 Textual description of "Download document" use case.....	22
Table 13- Textual description of "Move to folders" use case	23
Table 14- Textual description of "Add folder" use case.....	23
Table 15- Textual description of "Delete folder" use case.....	24
Table 16 Data Dictionary of «User» class.....	38
Table 17- Data Dictionary of «document» class.....	38
Table 18- Data Dictionary of «role» class.....	39
Table 19- Data Dictionary of «folder» class.....	39
Table 20- Data Dictionary of «Pages» class.....	39
Table 21- Data Dictionary of «DocumentArchive» class.....	40
Table 22- Data Dictionary of «Admin» class.....	40
Table 23- Data Dictionary of «Guest» class.....	40
Table 24- Data Dictionary of «IT» class.....	41
Table 25- Description of Class Methods.....	41

<u>Table 26- Associations linking our classes.....</u>	<u>44</u>
<u>Table 27- Characteristics of the materials used.....</u>	<u>57</u>

Acronyms

DMS:document management system

EDM: electronic document management

ECM: Enterprise Content Management

SDM :Service Delivery Manager

SRM :Service Request Manager

IIM:intelligent information management

TTD: test-driven development

AMI :Asynchronous Method Invocation

TTD:test-driven development

General Introduction

This report is the witness of four months of internship within Znet-it, within the framework of my graduation project engineering. Thanks to the progress of scientific and technological research, computer science has not stopped evolving and adapting to human needs. Above all, let's not forget to mention that man is becoming day by day dependent on his computer since it facilitates his task and helps him with his various tools to carry out his work with perspectivism and improvement. With the appearance of development tools, programming and with the evolution of methods for designing and modeling information systems, computer science has become as reliable and is integrated in all other fields. New information and telecommunication technologies are becoming indispensable in our daily lives and occupy a very important place in our lives since it contributes to the economic, social and cultural development of countries. In this perspective, and as part of our graduation project, carried out within Znet-it company, we are interested in the development of an electronic document management management platform. After collecting the information and having done the preliminary studies, we developed our vision of the application, its architecture, as well as defining the necessary tools and technologies.

This document is organized into three chapters: In the first we put the project in its general context. In the second chapter takes place the study of the existing and the specification of the needs. Then the third chapter is devoted to the analysis and detailed design of the future application. we present the development environment used followed by a description of the interfaces realized. Finally, we conclude our work with a general conclusion with an exposition of the contributions and perspectives.

CHAPTER 1: General Context

I. INTRODUCTION

The objective of this chapter is to put the work in its general context. First of all, we start with the presentation of the establishment that welcomed us and make a general presentation of our application and its objectives. Then we define the scope of the study and the objectives to be achieved. Then, we will begin the study of existing applications integrating in our same context.

II. PROJECT FRAMEWORK

This project carried within the framework of my graduation project within company Znet-it. Our project carried out at Znet-IT, consists in designing and implementing an EDM platform solution in order to obtain a classification and archiving method that will put an end to the daily problems of document management and facilitate the search and exploitation of documents.

II.1 PRESENTATION Of ZNET-IT

Znet-IT is a startup launched in 2021 that works in the field of information technology.

In addition, offering innovative services and services to help companies achieve their business objectives by providing them with the appropriate software and hardware infrastructure according to their needs and business requirements [1].



Figure 1 - logo of the company Znet-it

II.2 MISSION

Our mission is to provide our customers with the most appropriate and simple development solutions with a tool adapted to your company while respecting your business process by a team of developers who adapt to your needs, while being close to you, and listening to you. Technology development and services is simply what we love to do, and we do it well.[]

II.3 BUSINESS AREAS

Web and Mobile Development, IoT/AI, Embedded Development, UI/UX design, Server configuration and maintenance, Consulting.

III. ANALYSIS OF THE EXISTING

In order to develop a document management system that meets the required to analyze informatics services in order to its inadequacies. This part is made up of three parts:

- Study of the existing
- Criticism of the existing
- Proposed solution.

III.1 Study of the existing

Electronic Document management (EDM) is a field of information management that aims to digitize, organize, store and manage electronic documents within an organization. There are many studies on (EDM), as it has become a common practice in many sectors of activity.

Some of the most common studies on the (EDM) include:

-Market Research: These studies assess the state of the (EDM) market, including current trends, the opportunities, challenges and growth forecasts.

-Case Studies: The case studies present concrete examples of the application of EDM in specific companies or organizations. They can include examples of how EDM has improved the efficiency, productivity and results of the company.

-Technical Studies: These studies focus on the technical aspects of the EDM, including document management systems, storage solutions, security standards, etc. Many document management systems extend beyond basic functionality to include document-related workflows. There are a ton of other add-ons, features and capabilities touted by some document management systems – which has led to other names and designations like enterprise content management ECM, enterprise information management EIM and intelligent information management IIM. The term document management often refers to an overarching strategy for how a company stores, manages and tracks its electronic documents.



Figure 2 - Document Management System Work [2]

III.2 Criticism of the existing

✓ **Not Tracking Document Expiry Dates**

Most documents have expiration dates attached to them, especially if they are supplier-related certificates or product documents. Those contract records must be renewed before the expiry date, or else their conditions will no longer be valid, which could have legal implications or even terminate long-standing partnerships. If you set up an automated system, you won't have to worry about expired

documentation possibly jeopardizing your internal processes. This can be done using Microsoft Sharepoint, since it allows you to create site-specific content types. By creating a document template that requires an expiration date for the document, you can then set up your DMS to complete an action once that date arrives. This can look like a notification that is sent out to predetermined employees, or simply automatically archiving the document once it has expired.

✓ **Not Training Your Employees**

Another area you can't overlook is instructing your employees on how to use the DMS so they're able to take over certain maintenance responsibilities. This is an essential step on a company-wide level because document management training encourages more effective collaboration between departments following the same regulatory practices.

Without adequate training, your employees will not know where to find information about the company policy, onboarding protocols, audit reports, and other third-party agreements. If there are fewer mistakes on a doc file, it can be easily restored from the backup drive.

To minimize user errors and avoid confusion, you want to have naming and tagging conventions in place that will limit the number of tags and prevent documents from getting lost in the overarching database. Training should cover how to load tags and upload documents with the right metadata.

✓ **Overloading Your DMS**

A DMS may get bogged down by too many permissions in its document repository. Instead of assigning one-off permissions to various folders and docs, you can avoid disorganized sharing by grouping files into a base-level permission, only to add new ones as needed.

To keep disorganization from happening, don't add too many users with different access levels and refrain from saving duplicate folders and docs. Scan all your files and clean them by using Power Automate flow, as this allows you to create an auto-archive system for docs based on the date created or last modified.

When it comes to duplicate files or versioning, SharePoint stores the version history of your documents, so you don't need to create a new version every time it's sent for approval or a round of edits. A way around sharing multiple copies that clog up your DMS is returning to a previous version whenever you don't approve of certain edits.

✓ **Migrating Directly From File Share to a DMS**

When switching from a file share system to a document management system, it's advised that you don't just migrate all of your content over without sorting it first. Evaluate what the cloud has to offer before you shift from your current platform.

Prioritize features like tagging, which will optimize finding documents by saving you time. It's better to have a functional search mechanism for refining a search via its date, client, or location, rather than simply relying on deep, nested folders to finish the job.

III.3 Suggested Solutions

In order to obtain the graduation engineering, ZNET-IT proposed to me to conceive and develop a system which makes it possible to manage all the activity of documents management system.

A document management system (DMS) is a software solution that helps organizations efficiently store, organize, manage, and track their digital documents. Here are some suggested solutions for implementing a document management system:

✓ **On-Premises DMS:**

if you to have full control over your document management system and data, you can deploy an on-premises DMS solution. Some popular options include OpenDocMan, LogicalDOC, or Alfresco Community Edition.

✓ **Enterprise Content Management (ECM) Systems:**

ECM systems provide comprehensive document management capabilities, including document capture, indexing, workflow automation, and records

management. Examples of ECM solutions are Microsoft SharePoint, Laserfiche, and IBM FileNet.

✓ **Document Collaboration Tools:**

Consider using collaboration tools that offer document management features. For instance, tools like Microsoft Teams, Slack, or Asana allow teams to collaborate on documents, track changes, and maintain document versions.

✓ **Document Scanning:**

To digitize physical documents, consider using document scanning and optical character recognition (OCR) software. These tools can convert paper documents into searchable digital files.

✓ **Security and Access Controls:**

Ensure your DMS has robust security measures to protect sensitive information. Implement user access controls, encryption, and audit trails to monitor document access and modifications. Additionally, consider data backup and disaster recovery plans.

✓ **Integration with Existing Systems:**

If you already use other business applications such as customer relationship management (CRM) or enterprise resource planning (ERP) systems, choose a DMS that integrates seamlessly with them. This will streamline workflows and improve productivity.

✓ **Mobile Access:**

Consider a DMS that offers mobile apps or responsive web interfaces, allowing users to access, view, and collaborate on documents from their smartphones or tablets.

IV. OBJECTIFS TO REACH

- Provide safe storage and backup of all documents in a project library.
- Provide clarity regarding which version of a deliverable is the latest version.
- Provide a clear record of approved deliverables over the life of the project.

- Provide measures to maintain restricted access to confidential documents.
- Provide an accurate and complete archive of project documents to the permanent organization at the end of the project.

V. ESTIMATED PLANNING:

It is difficult to talk about a project before starting an analysis of the work to be done. However, it is unavailable to make a first general estimate in order to be able to frame the project and launch it. In what follows, we will draw up our plan from the idea of the project to its realization.

Table 1 - Gantt chart

	March				April				May				June		
	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3
Initialization of the project															
Discovery of the existing															
Needs analysis															
Training															
Training languages															
Study of the project															
Functional study															
Design															
Production															
Creating backend and NoSQL															
Creation of interfaces															
Design and ergonomics															
Source code testing															
Implementation of the system															
Deployment of the system															
Writing of the report															
Preparing the book															

VI. CONCLUSION:

In this chapter, we have presented the objective of our graduation project, we have studied closely the existing system. In the next chapter, we begin conceptual modeling of our system.

CHAPTER 2: Analysis and specification of requirements

I. INTRODUCTION

This chapter represents our startup during which we will specify the requirements of the different users. A study of functional and non-functional needs is then necessary. Finally, we will explain the proposed architecture and the working environment.

II. REQUIREMENT ANALYSIS

The objectives of this section are to introduce the various actors who interact with the application and to define functional and non-functional needs.

II.1 ACTORS IDENTIFICATION

The objective of all the use cases is to describe the requirements of the functioning of the system, each use case corresponds to a business function of our system.

Table 2 - The list of actors

List of actors	Role of the actor
Admin	Manages all the functionalities of the DMS.
IT	Takes care of the management of documents related to computer maintenance and tasks related to the field of information technology.
Guest	It is only viewing documents.

II.2 Functional Requirements

It's about the features of the system. These are the needs specifying a behavior input/output of the system. The system must allow the user to perform the following operations:

✓ User management

it is a tool for performing management operations such as adding, the total deletion or archiving, modification, search and consultation of information characterizing each of the users. This module must contain all the following tools :

- Add specific roles to the company in a dynamic way.
- Change the role of a given user.
- Add the specific departments of the company in a dynamic way.

✓ Document management

The application must ensure proper document management, namely, the documents Word, Excel, PDF, Zip ... It will offer several features:

- Document import: documents can come from the user's machine (internal document) or from other applications (external document).
- View the documents in a web browser: the files must be presented without deformation or missing elements while keeping the specifications of each type of document.
- Browse the different versions of a given document: the application must manage the versions (revisions) of the documents. These versions can be consulted.
- Index documents for a quick search: the application must have a search engine capable of saving the contents of documents and performing a quick search on a large amount of documents.
- Export and download documents in several formats: a user can export and then download a document in different formats according to his access rights.

- Document classification: a flexible classification of documents according to their types.
- Document archiving: this is the end of the life cycle of a document in a company.

✓ **Rights management:**

Each actor has an account to access the system, and for each account, he is assigned a role or a profile that defines the functionalities that the user cannot exceed.

✓ **Control management:**

The administrator must carry out one or more managers for each treatment by specifying the start and end dates of each of them, he can modify them in case the treatment has changed.

✓ **Sending notification:**

A notification is sent when a new document is imported to all users.

II.3 Non-Functional Requirements

A no-functional need is a need specifying system properties, such as environment and implementation constraints, performance, dependency, platform, ease, maintenance, extensibility and reliability requirements.

No-functional needs are therefore independent needs of the users: rather, they are constraints that are assumed to ensure the quality and proper functioning of the system. Thus, like any application, ours must present an ergonomic interface.

Given its use, our application must provide simple and clear interfaces, so the system should take into account the following constraints:

✓ **Need availability and reliability:**

The system must be operational continuously, except during the phases of regular maintenance of the system by the administrator.

✓ **Performance needs:**

Describe the execution performance of the system, usually in terms of response time. The system must guarantee a minimum response time: The loading of a web page in the browser should not take more than 15 seconds in normal condition.

✓ **The interfaces must be ergonomic:**

The interface of the application should be simple and practical so that the user can use it without referring to special knowledge. In other words, the information must be readable and easy to access by any user.

III. CHOICE OF DESIGN METHODOLOGY AND JUSTIFICATION

One of the main steps representing the strategic phase when developing an application is conception. It is essential to use a conception methodology. Among the most well-known design languages are Merise and UML (Unified Modeling Language). In fact, I chose to do our conceptual study with UML language, this choice is justified by the fact that:

- UML is a language that covers the software development cycle, from requirements specification to implementation. The use of this language is both a gain in precision and a guarantee of stability.
 - This language offers a set of diagrams that allow to represent a system in a graphic way covering the different levels of abstraction, which helps to understand the operation of the system.
- ✓ It allows a better communication with the user:
- ➔ Distortions between user expectations and results are minimized.
 - ➔ Use cases allow the user to better understand and follow the design analysis phases.
 - ➔ The prototype is used from the beginning of the project so that the user can specify his needs, thus reducing the risk of failure.

IV. APPLICATION CONCEPTION

IV.1 Use Case Diagrams

The use case diagram represents the structure of the major functionalities needed by system users. It is based on two major concepts such as: the actor and the use case. In this diagram, modeling and structuring of interactions are ensured between the actors and the objects that the system implements.

A use case diagram can group several notations according to its category and the criterion chosen. We note a few:

- The association relation: associating an actor with a use case.
- The extension relationship: the source use case adds its behavior to the destination use case if a condition is fulfilled.
- The inclusion relation: the instance of the source use case contains the behavior described by the destination use case.

IV.2 Use case specification

IV.2.1 "Authentication" use case diagram

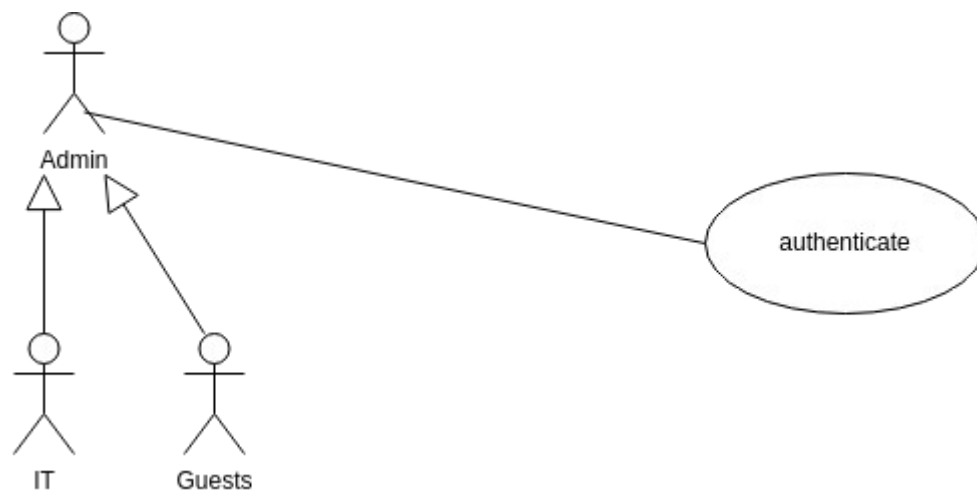


Figure 3 - "Authentication" use case diagram

IV.2.2 Description "Authentication"

Table 3 - Textual description of "Authentication"

Name of the use case	Authenticate.
Actors	Admin

Description	The actor authenticates himself using his email address and his password.
Pre Condition	- Authentication operation requested by the actor.
Post Condition	Authenticated actor.
Basic scenario	<ol style="list-style-type: none">1. The actor enters his email and password.2. The system checks the actor's data.3. The system displays an authentication success message.4. The system displays the home interface specific to the actor.
Exceptional scenario	<p>invalid email or password :</p> <ul style="list-style-type: none">- The system displays an error message.- Return to step 1 of the basic scenario.

IV.2.3 "Manage users" use case diagram

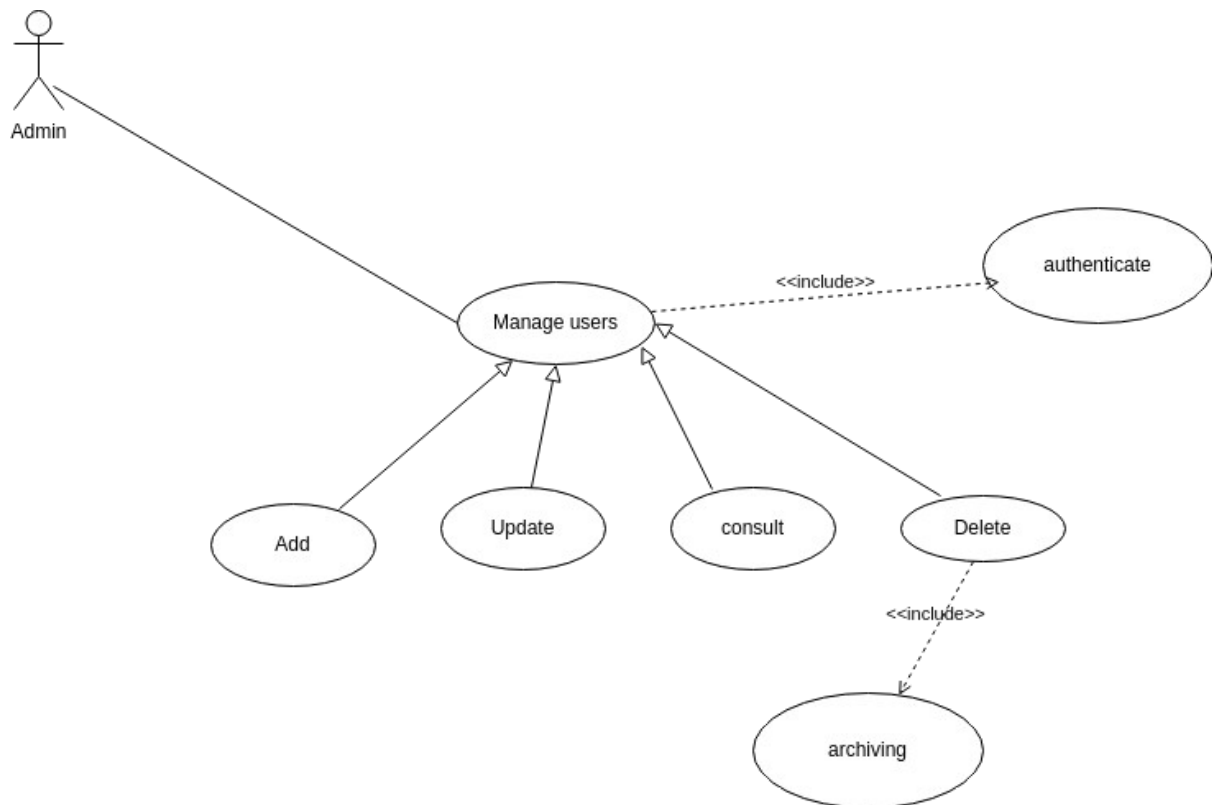


Figure 4 – "Manage users" use case diagram

IV.2.4 Textual description of "Manage users" use case

After accessing his workspace, Admin has the right to add, modify, delete to consult, to do a quick search and to delete a user and archiving.

The use cases for this process are detailed below:

- User addition

Table 4 - Textual description of "User addition" use case

Name of the use case	User addition
Actors	Admin
Description	The administrator enters the data concerning the user such as email, Full name and other information.
Pre Condition	- Chosen addition operation.

	- Authenticated administrator.
Post Condition	User added.
Basic scenario	<ol style="list-style-type: none"> 1. "include" to authenticate. 2. The administrator enters the data relating to the user 3. The administrator validates the operation. 4. The system checks the input. 5. The system saves the user's data. 6. The system displays a message of "Successful addition". 7. The user enters the User Name and the password. 8. The system checks the input. 9. The system saves the new data.
Exceptional scenario	<ol style="list-style-type: none"> 4.1. Invalid data : <ul style="list-style-type: none"> - The system displays an error message: "missing field". - Return to step 2 of the basic scenario. 4.2. Already existing user : <ul style="list-style-type: none"> - The system displays a message "Already existing user" - Return to step 2 of the basic scenario. 9.1. An already existing email <ul style="list-style-type: none"> -The system displays a message "it is necessary to change the email" -Return to step 8 of the basic scenario

➤ Edit user

Table 5 - Textual description of "Edit user" use case

Name of the use case	Edit user
Actors	Admin
Description	The actor has the possibility to modify the user's data.
Pre Condition	- the authenticated Admin. - Selected modification operation.
Post Condition	Modified user data.
Basic scenario	1. "include" to authenticate. 2. The Admin modifies the selected user data. 3. The Admin validates the modification. 4. The system checks the modification. 5. The system saves the change. 6. The system displays a "success" message.
Exceptional scenario	- Invalid modification - the system displays an error message "missing fields" -Return to step 2 of basic scenario

➤ Consult user

Table 6 - Textual description of "Consult user" use case

Name of the use case	Consult user
Actors	Admin
Description	The actor selects The user from those displayed in a list to consult its detailed technical sheet.
Pre Condition	- Authenticated actor.

	- Selected consultation operation.
Post Condition	Profile consulted.
Basic scenario	<ol style="list-style-type: none"> 1. "include" to authenticate. 2. The actor requests the consultation of the user 3. The system displays a user list. 4. The actor selects a user to consult. 5. The system provides the user profile.
Exceptional scenario	<ul style="list-style-type: none"> - The user list is empty - return to the basic scenario step 2

➤ Delete user

Table 7 - Textual description of "Delete user" use case

Name of the use case	Delete user
Actors	Admin
Description	The actor has the option of delete archive the user.
Pre Condition	<ul style="list-style-type: none"> - the authenticated actor. - Selected deletion operation.
Post Condition	Deleted or archived user.
Basic scenario	<ol style="list-style-type: none"> 1. "include" to authenticate. 2. The actor deletes The selected user. 3 The system asks to choose between deleting and the archiving of the user. 4. The actor validates his choice.

	5. The system deletes or archives the user. 6. The system displays a "success" message.
Exceptional scenario	5. The user cancels the deletion : - Return to step 2 of the basic scenario.

IV.2.5 "Manage documents " use case diagram

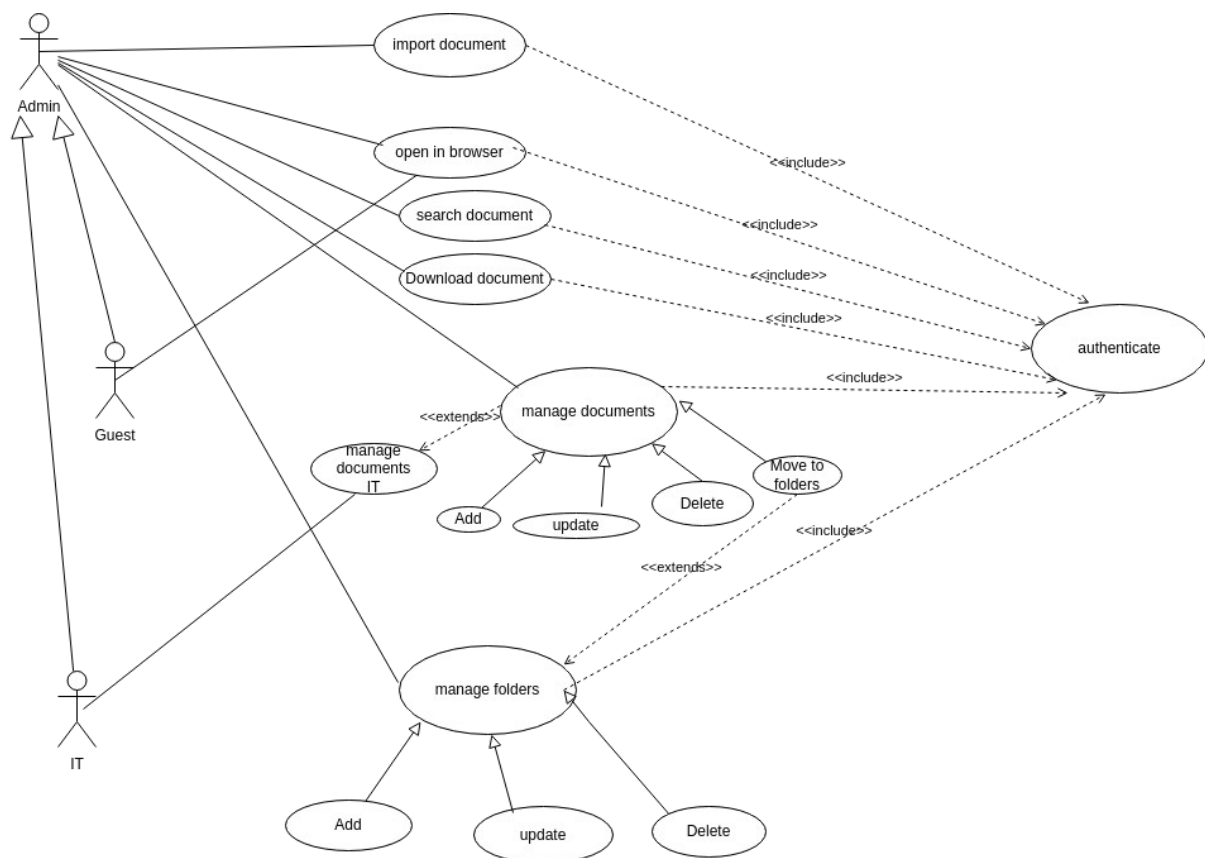


Figure 5 - "Manage documents " use case diagram

V.2.6 Textual description of " Manage documents" use case

After each having access to their workspace, the actors have the right to add, to modify, consult the documents and their detailed information.

The use cases for this process are detailed below:

➤ Add document

Table 8 - Textual description of "Add document" use case

Name of the use case	Add document
Actors	Admin
Description	The actors add the necessary documents
Pre Condition	- Chosen add operation. - authenticated actors.
Post Condition	Document added.
Basic scenario	1. "include" to authenticate. 2. The actors add the documents and the information necessary. 3. The actors of the operation. 4. The system checks the input. 5. The system saves the added documents.
Exceptional scenario	4.1. Invalid data : - The system displays an error message: "missing field". - Return to step 2 of the basic scenario.

➤ Edit document

Table 9 - Textual description of "Edit document" use case

Name of the use case	Edit document
Actors	Admin

Description	The actors have the possibility to modify the documents or their information.
Pre Condition	<ul style="list-style-type: none"> - authenticated actors. - Selected modification operation.
Post Condition	<p>Document or modified document information.l. "include" to authenticate.</p> <ol style="list-style-type: none"> 2. The actor modifies the title and description of the document 3. The actor validates the modification. 4. The system checks the modification.
Basic scenario	<ol style="list-style-type: none"> 1. "include" to authenticate. 2. The actor modifies the title 3. The actor validates the modification. 4. The system checks the modification.l. "include" to authenticate. 2. The actor modifies the title and description of the document 3. The actor validates the modification. 4. The system checks the modification.
Exceptional scenario	<ol style="list-style-type: none"> 4. Invalid modification <ul style="list-style-type: none"> - The system displays a "missing fields" error message. - Return to the basic scenario step 2.

➤ Delete document

Table 10 - Textual description of "Delete document" use case

Name of the use case	Delete document
----------------------	-----------------

Actors	Admin
Description	The actor has the possibility to delete a document
Pre Condition	- the authenticated actor. - Selected deletion operation.
Post Condition	The document is deleted.
Basic scenario	1. "include" to authenticate. 2. The actor deletes The selected document. 3. The system deletes document. 4. The system displays a "success" message.
Exceptional scenario	3. The user cancels the deletion : - Return to step 2 of the basic scenario.

➤ Search document

Table 11 - Textual description of "Search document" use case

Name of the use case	Search document
Actors	Admin,
Description	This use case allows the actor to search for a existing document.
Pre Condition	Document found
Post Condition	Document sent.
Basic scenario	1. "include" to authenticate. 2. The actor enters the title of the document to be searched

	3. The system displays a list of documents. 4. The actor selects a document to search for. 5. The actor chooses the operation to enter
Exceptional scenario	3.1. The document list is empty 3.1.1 return to step 2 of the basic scenario

➤ Download document

Table 12 - Textual description of "Download document" use case

Name of the use case	Download document
Actors	Admin
Description	The actors have the possibility to download the document
Pre Condition	- Authenticated actors. - Select download operation.
Post Condition	Downloaded document
Basic scenario	1. "include" to authenticate. 2. The actor downloads the selected document. 3. The system asks for confirmation of the format of download. 4. The actor validates the format. 5. The system displays a message of "download success".
Exceptional scenario	3. The administrator cancels the download : - Return to the basic scenario step 2.

➤ Move to folders

Table 13 - Textual description of "Move to folders" use case

Name of the use case	Move documents to folders
Actors	Admin
Description	This use case allows the actor to move documents in folders
Pre Condition	- authenticated actor. - Choosing the right folder.
Post Condition	Document moved to folder.
Basic scenario	1. "include" to authenticate. 2. The actor chooses document 3. the actor taps icon. 4. The actor choose the folder convient. 5. The actor validates the modification. 6. The system checks the move of document.
Exceptional scenario	3. The user cancels the movement of document : - Return to the basic scenario step 2.

➤ Add folder

Table 14 - Textual description of "Add folder" use case

Name of the use case	Add folder
Actors	Admin
Description	The actors add the necessary folders

Pre Condition	<ul style="list-style-type: none"> - Chosen add operation. - authenticated actors.
Post Condition	Folder added.
Basic scenario	<ol style="list-style-type: none"> 1. "include" to authenticate. 2. The actors add the name of folder. 3. The actors of the operation. 4. The system checks the input. 5. The system create document.
Exceptional scenario	<ol style="list-style-type: none"> 4.1. Invalid data : <ul style="list-style-type: none"> - The system displays an error message: "missing field". - Return to step 2 of the basic scenario.

➤ Delete folder

Table 15 - Textual description of "Delete folder" use case

Name of the use case	Delete folder
Actors	Admin
Description	The actor has the possibility to delete a folder
Pre Condition	<ul style="list-style-type: none"> - the authenticated actor. - Selected deletion operation.
Post Condition	The folder is deleted.
Basic scenario	<ol style="list-style-type: none"> 1. "include" to authenticate. 2. The actor deletes The selected folder. 3. The system deletes foldert.

	4. The system displays a "success" message.
Exceptional scenario	3. The user cancels the deletion : - Return to step 2 of the basic scenario.

V. SEQUENCE DIAGRAM

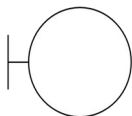
A sequence diagram is a graphic document that shows for case scenarios accurate use, generated events and interactions between objects based on ordered messages. Each message passing through a link is symbolized by an expression that carries an expression.

The reading is done from top to bottom, and the chronological order must respect this meaning.

The realization of a sequence diagram makes it possible to list the methods that will be needed during the development phase.

Interaction frameworks For more complex cases, algorithms can be integrated into the Sequence diagrams. There are three types of objects in UML:

➔ Boundary:



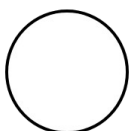
- It represents the interface between the actor of a use case and the system.

➔ Control:



- These classes direct activities between boundary classes and Entity classes.

➔ Entity:



- It is the classes described in the use cases that represent the data during processes.

V.1 "Authentication" use case sequence diagram

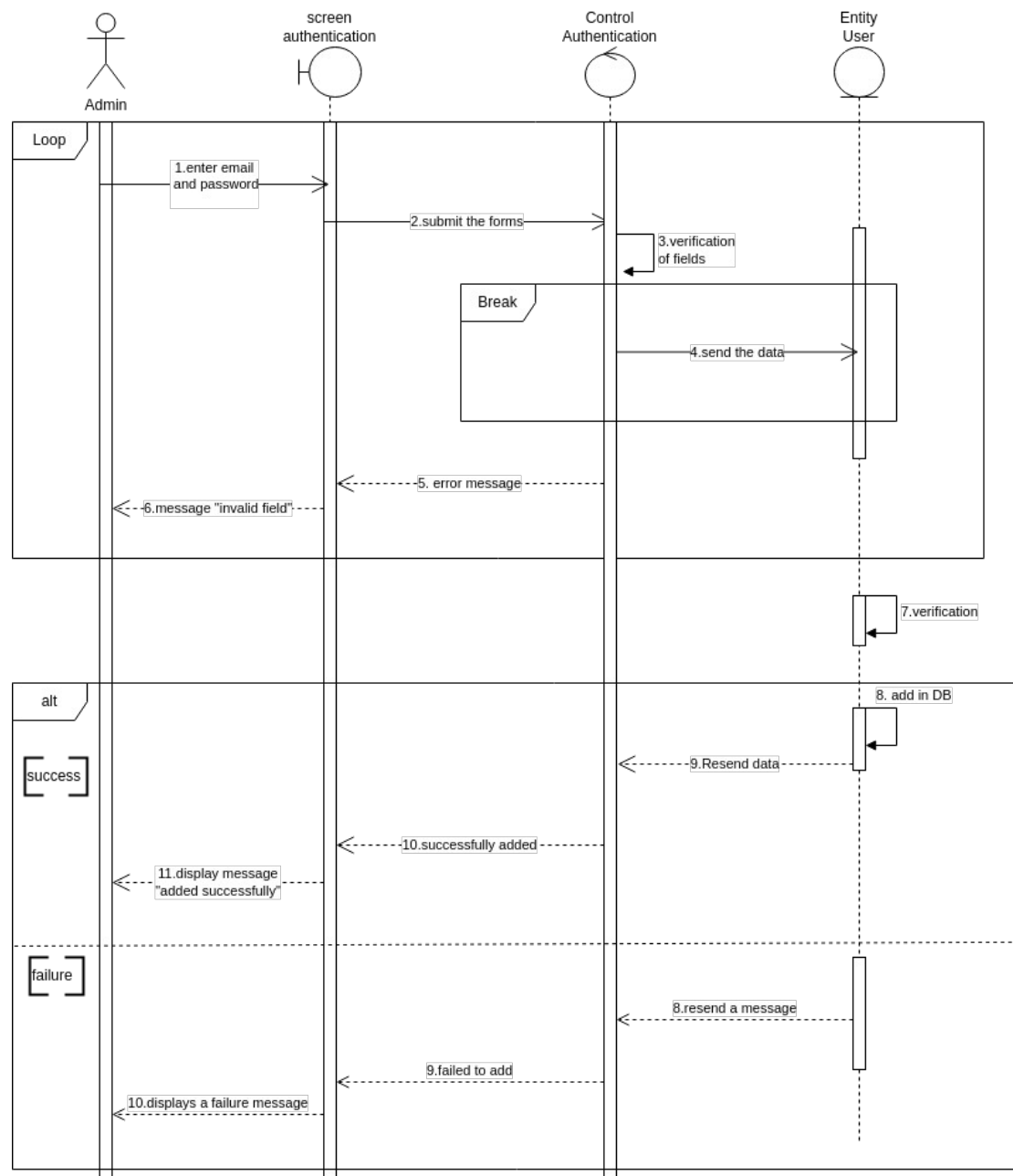


Figure 6 - "Authentication" sequence diagram

V.2 "Add User" use case sequence diagram

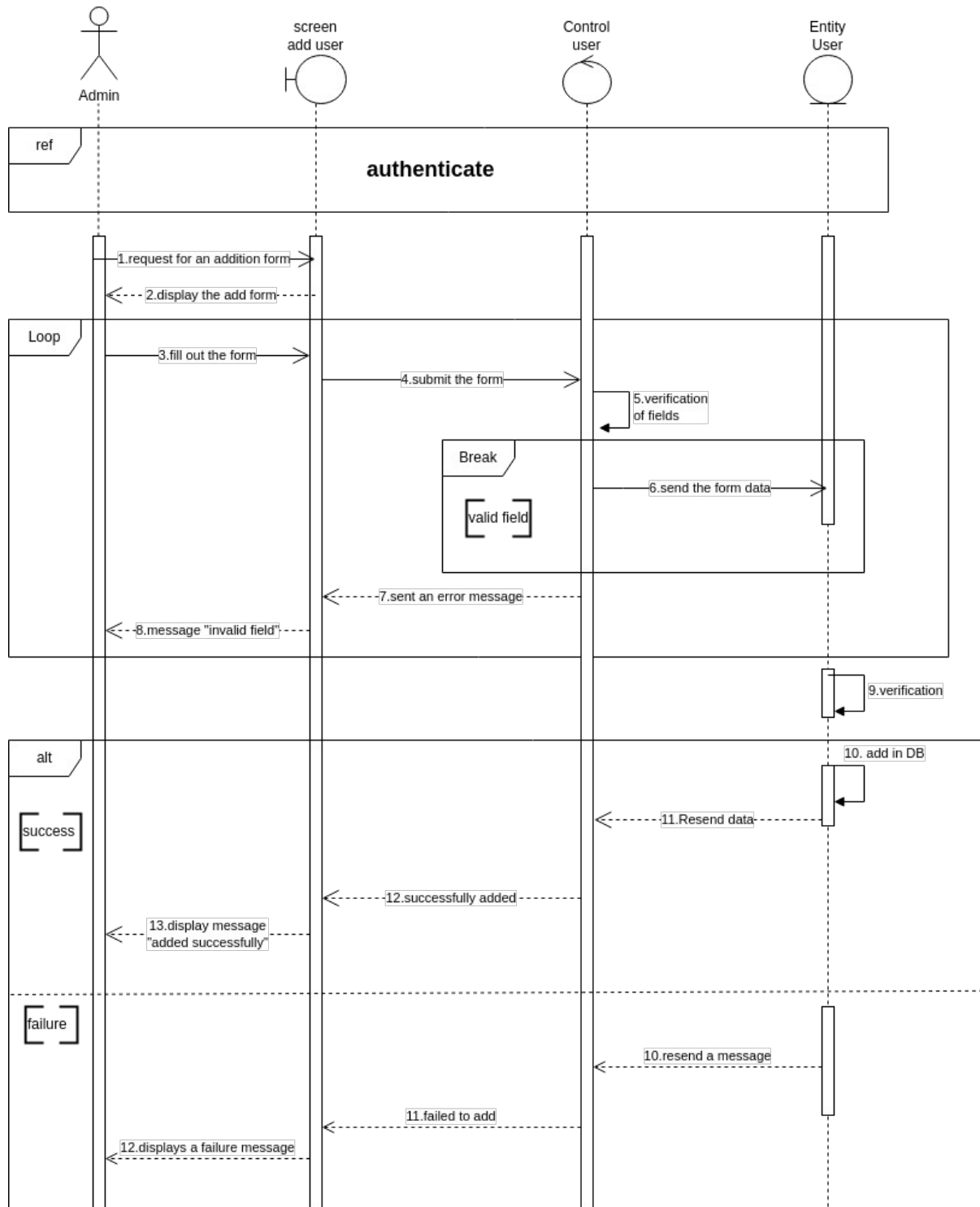


Figure 7 - "Add User" sequence diagram

V.3 "update User" use case sequence diagram

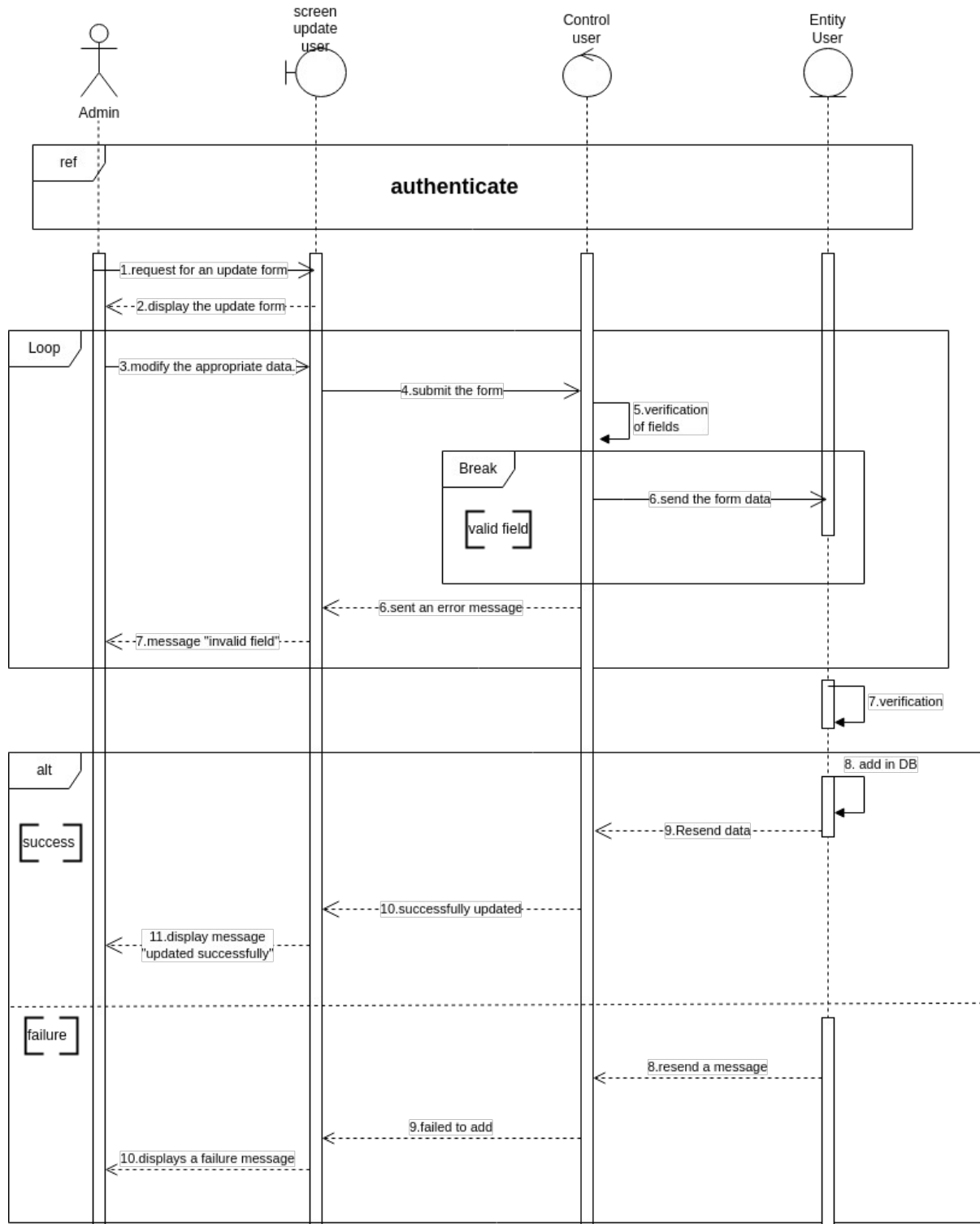


Figure 8 - "update User" sequence diagram

V.4 "Delete User" use case sequence diagram

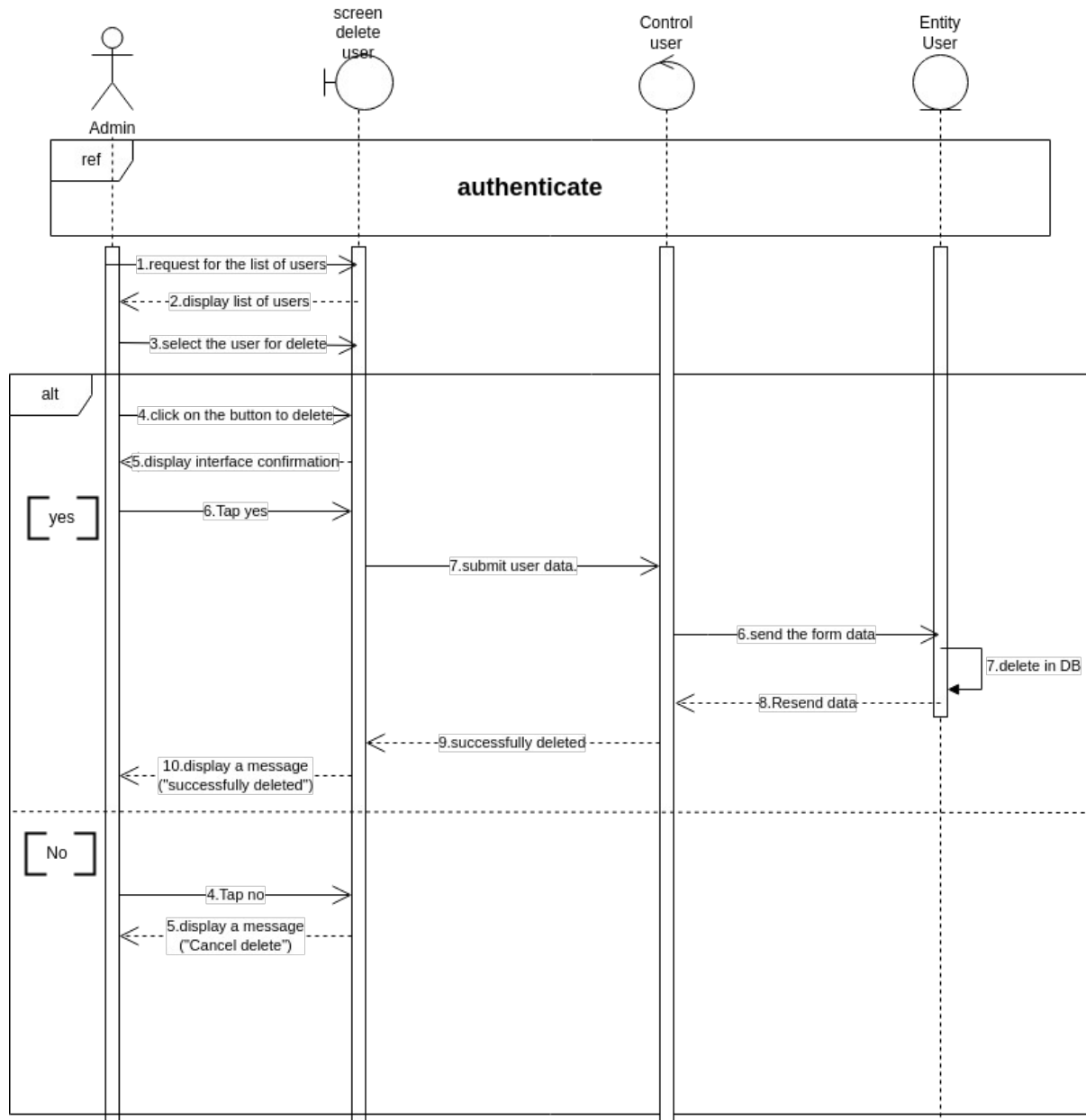


Figure 9 - "Delete User" sequence diagram

V.5 "Consult User" use case sequence diagram

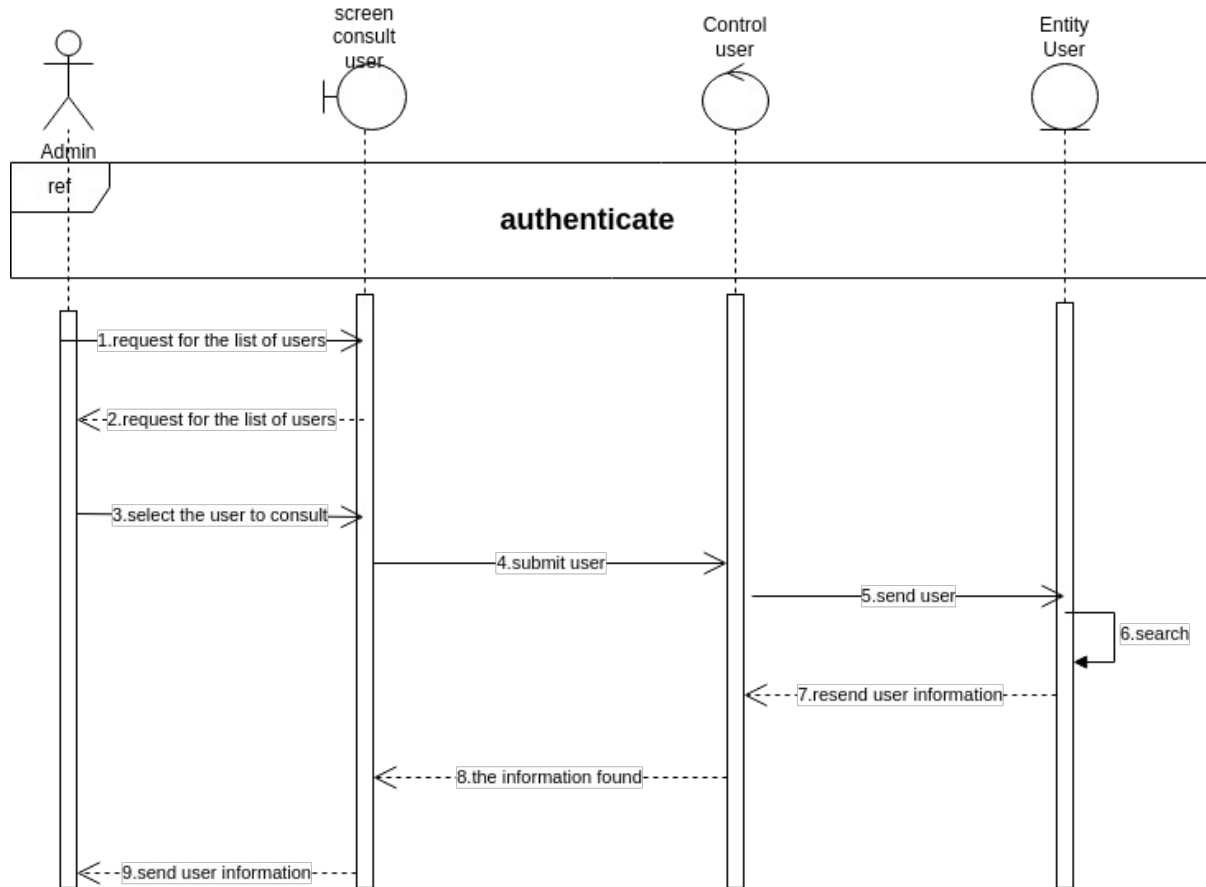


Figure 10 - "Consult User" sequence diagram

V.6 "Add document" use case sequence diagram

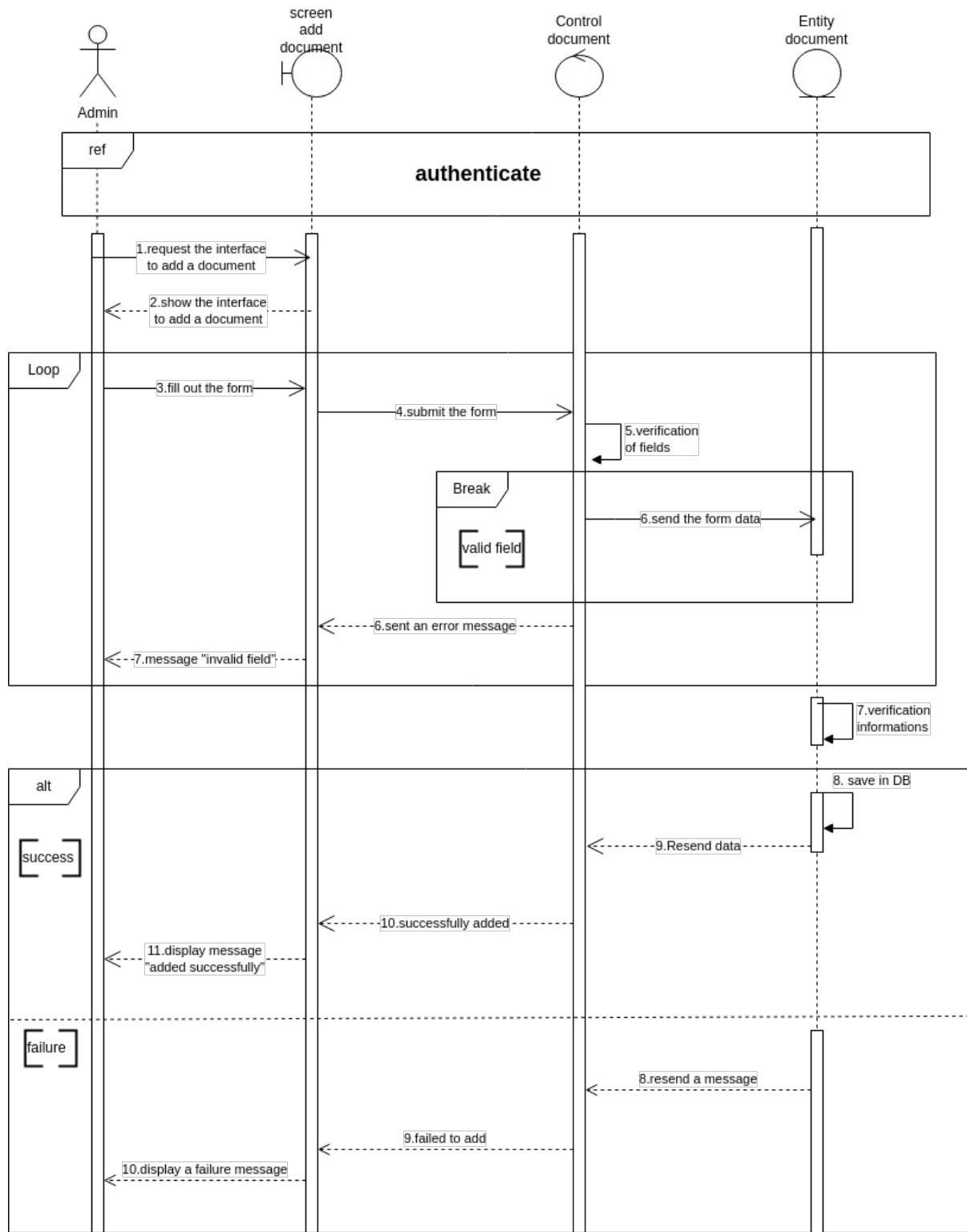


Figure 11 - "Add document" sequence diagram

V.7 "update document" use case sequence diagram

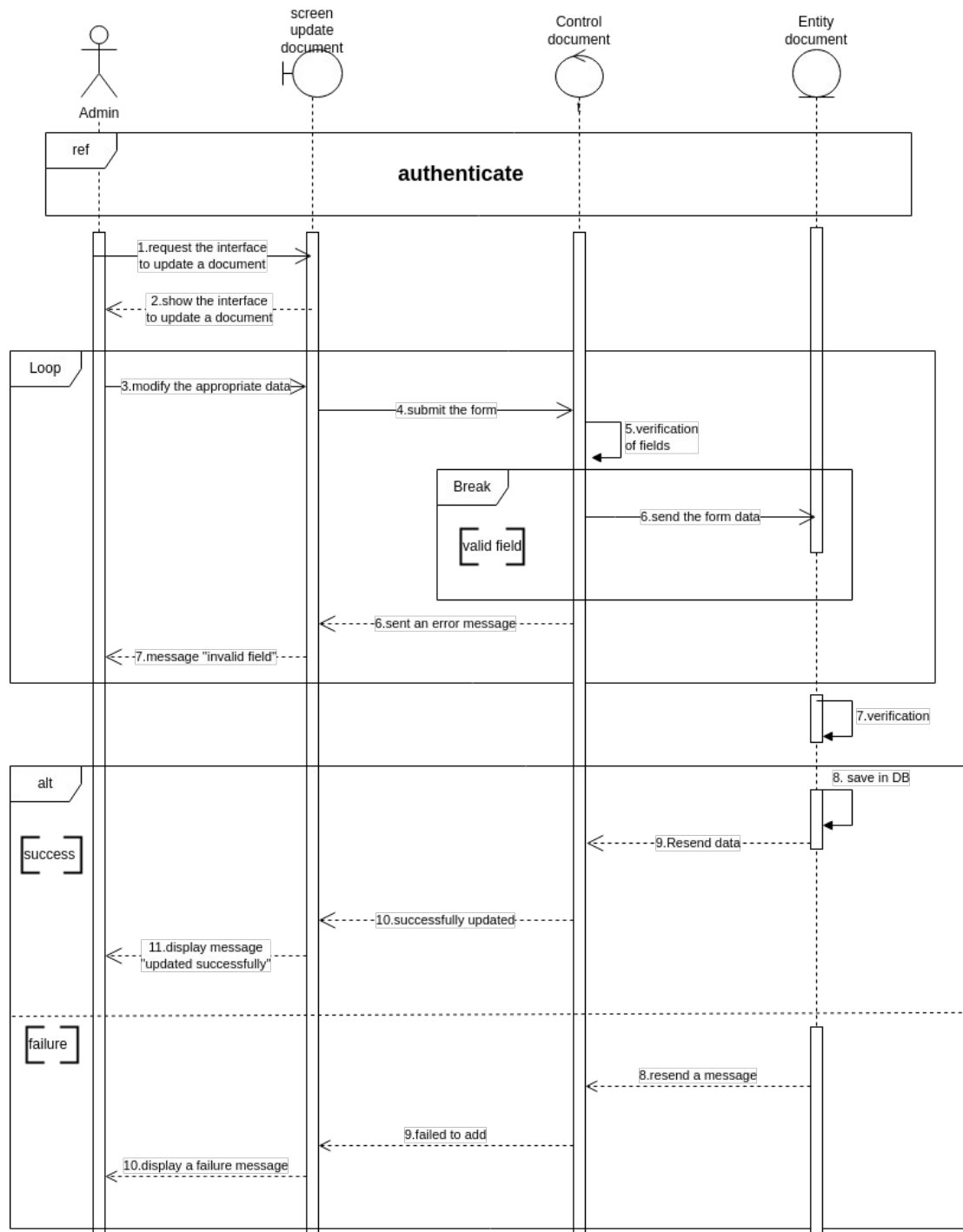


Figure 12 - "update document" sequence diagram

V.8 "search document" use case sequence diagram

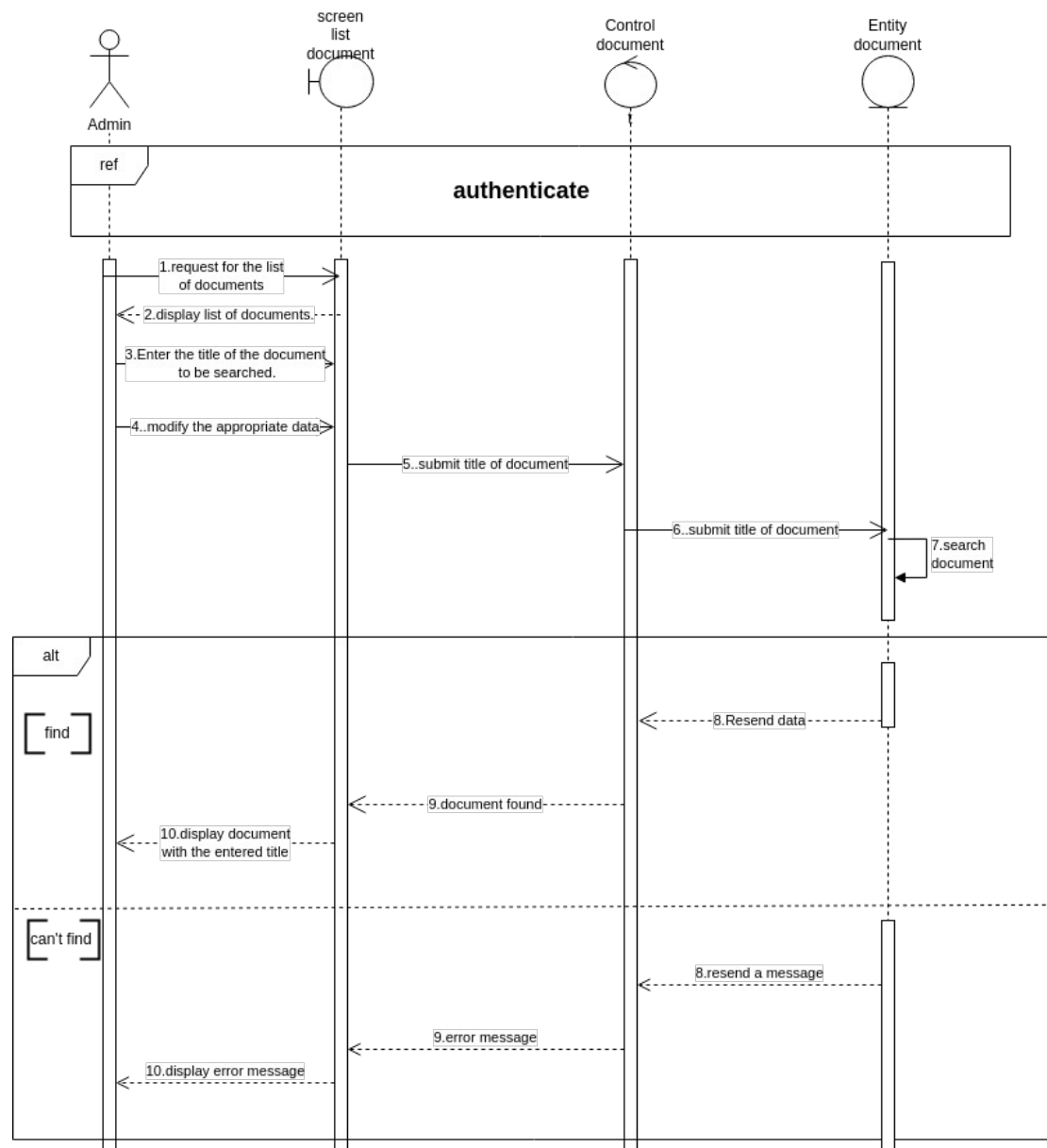


Figure 13 - "search document" sequence diagram

V.9 "Add folder" use case sequence diagram

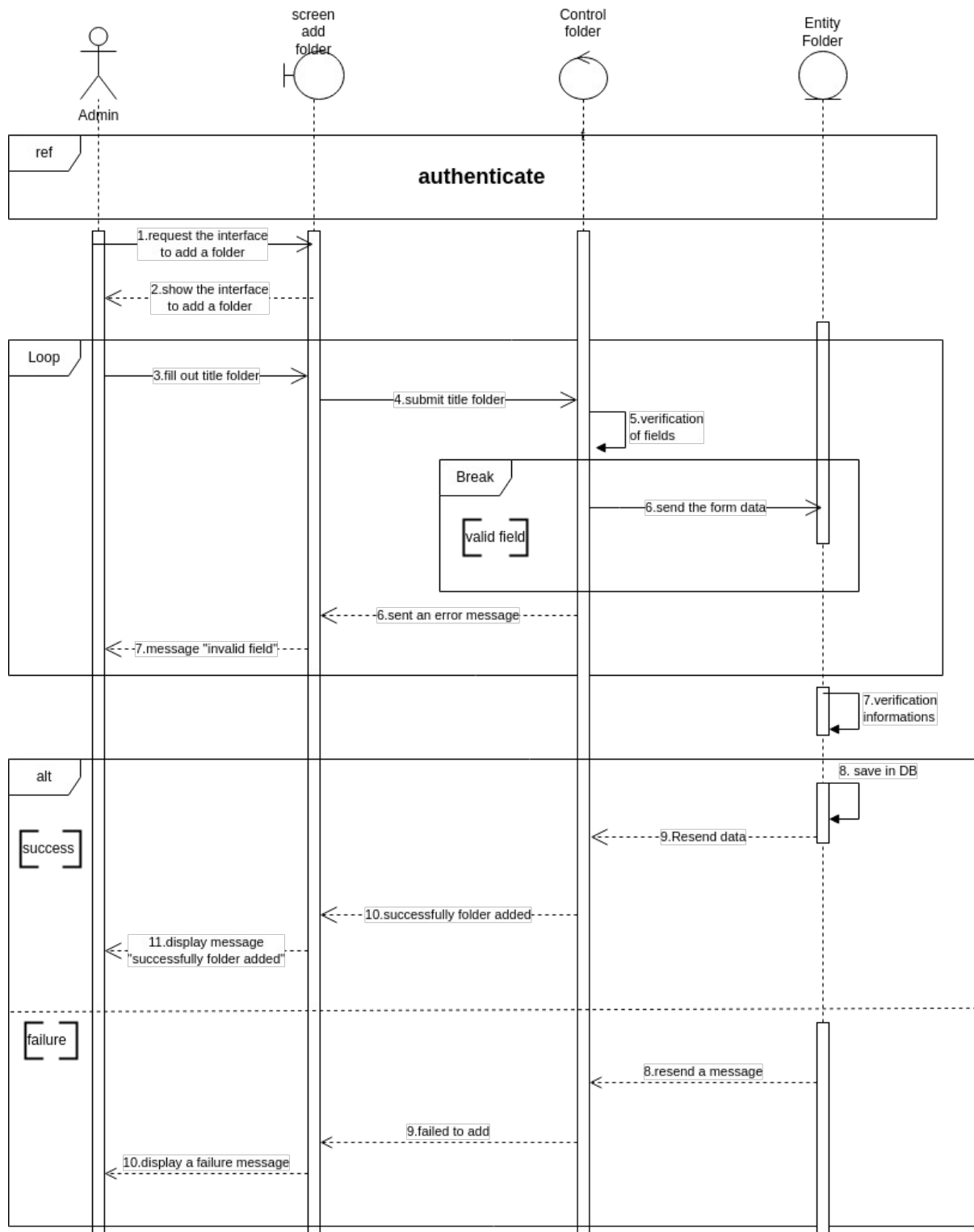


Figure 14 - "Add folder" sequence diagram

V.10 "Delete folder" use case sequence diagram

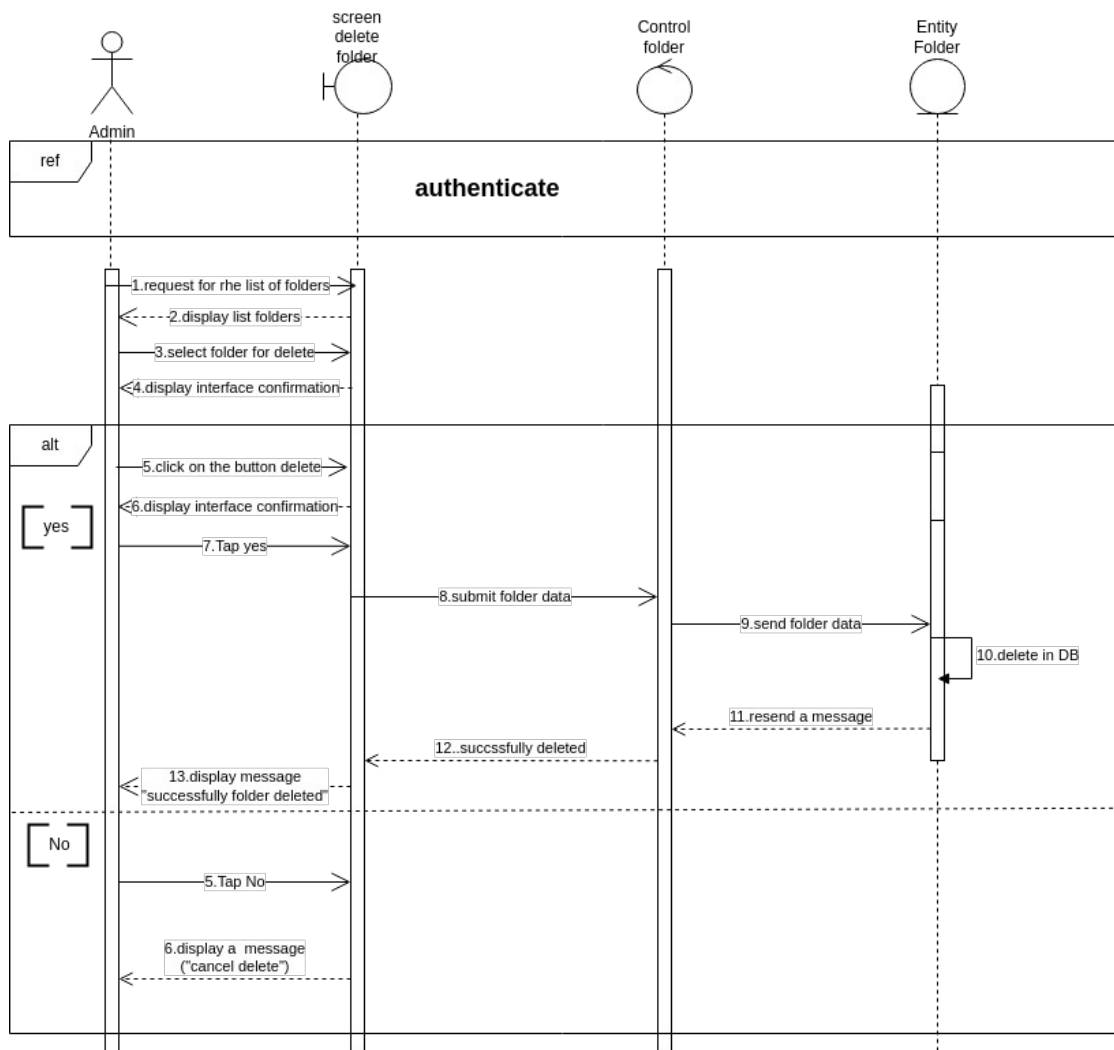


Figure 15 - "Delete folder" sequence diagram

V.11 "Move to folders" use case sequence diagram

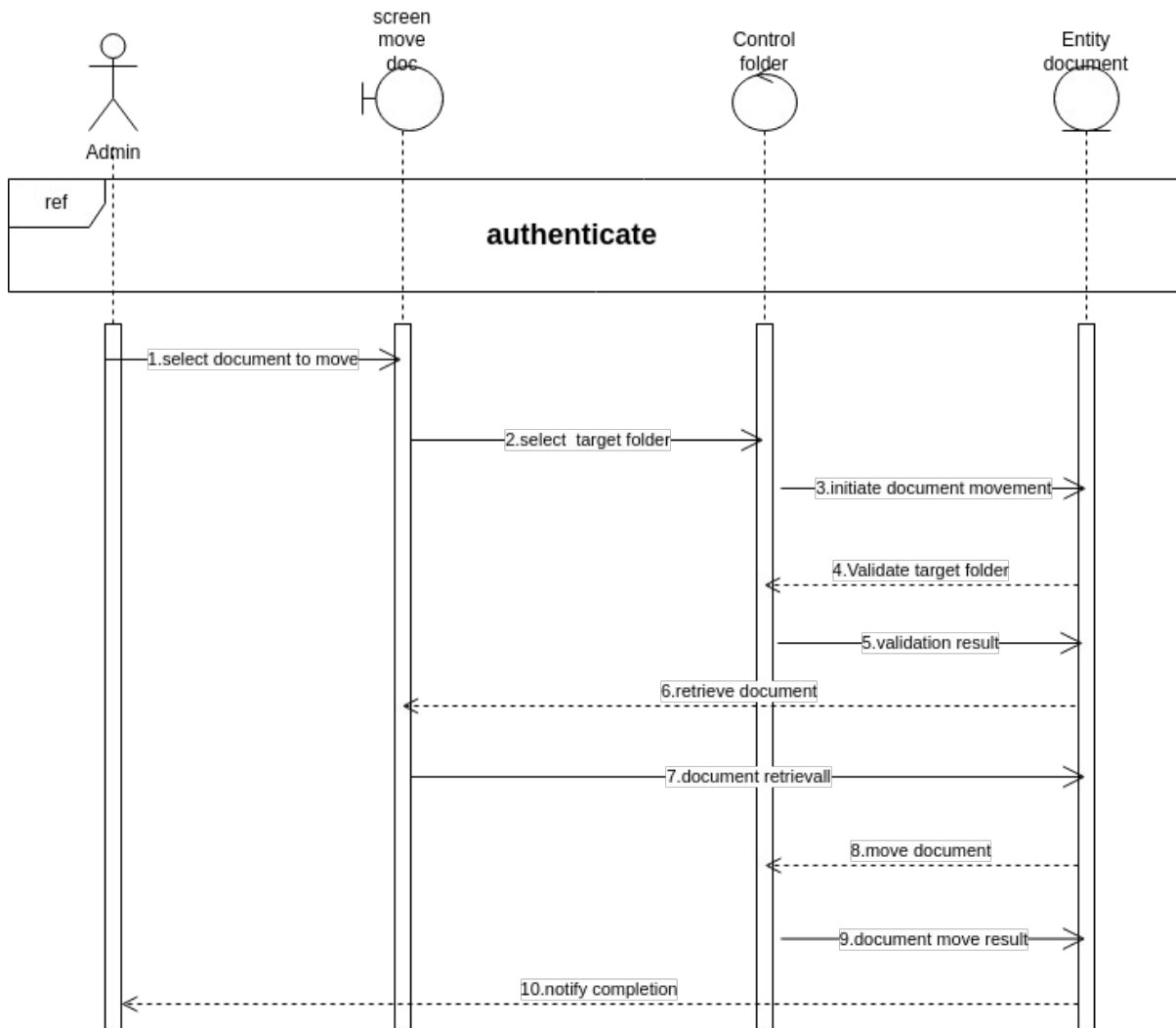


Figure 16 - "Move to folder" sequence diagram

V.12 "update folder" use case sequence diagram

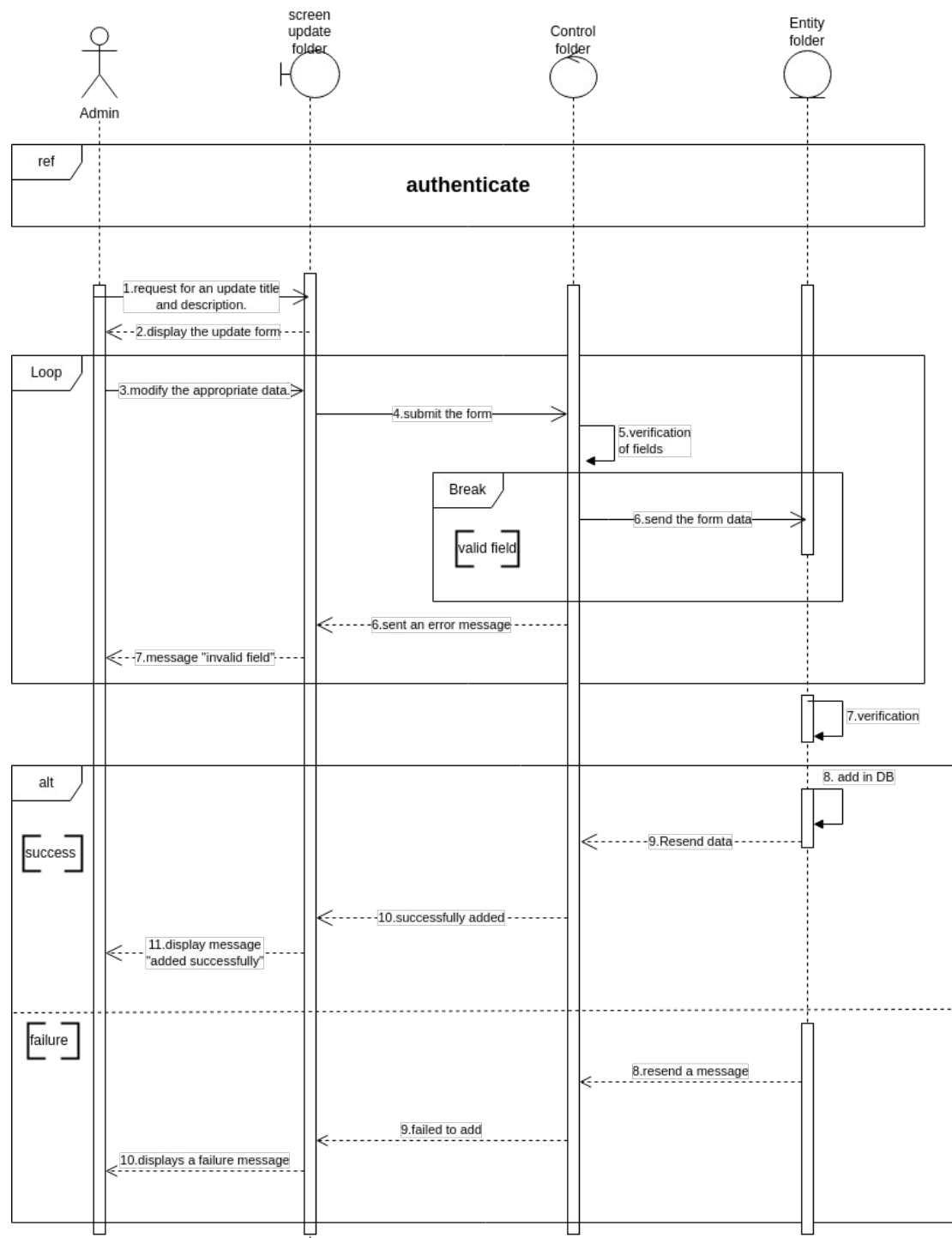


Figure 17 - sequence diagram "update folder"

VI. CONCEPTUAL DATA MODELING

VI.1 Data Dictionary

Data dictionary is the first step in our analysis of the database which allowed me to accurately identify all the data that this system must handle. The table below lists the data manipulated by the organization and their number, code and designation.

Table 16 - Data Dictionary of «User» class

Number	Attribute	Designation	Type
Class: User			
1	id_user	Identification of users	integer
2	email	Mail of users	string
3	Full name	Name of users	string
4	Mobile_phone	Mobile of users	integer
5	Password	Password of users	string

Table 17 - Data Dictionary of «document» class

Number	Attribute	Designation	Type
Class: Document			
1	id_doc	Identification of documents	integer
2	title	Title of documents	string
3	path	path of document	string

4	Category	Category of document	string
5	description	Description of document	string

Table 18 - Data Dictionary of «role» class

Number	Attribute	Designation	Type
Class: Role			
1	id_role	Identification of documents	integer
2	name_role	name of roles	string
3	type_role	Type of roles	string

Table 19: Data Dictionary of «folder» class

Number	Attribute	Designation	Type
Class: folder			
1	id_f	Identification of folders	integer
2	name_f	name of roles	string

Table 20 - Data Dictionary of «Pages» class

Number	Attribute	Designation	Type
Class: Pages			
1	id_pages	Identification of pages	integer

Table 21 - Data Dictionary of «DocumentArchive» class

Number	Attribute	Designation	Type
Class: DocumentArchive			
1	id_doc_arch	Identification of documents archives	integer

Table 22 - Data Dictionary of «Admin» class

Number	Attribute	Designation	Type
Class: Admin			
1	id_admin	Identification of administrator	integer
2	Admin_name	nameof administrator	string

Table 23 - Data Dictionary of «Guest» class

Number	Attribute	Designation	Type
Class: Admin			
1	id_guest	Identification of guest	integer
2	Guest_name	Name of Guest	string

Table 24 - Data Dictionary of «IT» class

Number	Attribute	Designation	Type
Class: Admin			
1	id_IT	Identification of IT	integer
2	IT_name	Name of IT	string

VI.2 Classes Representation

Object modalization is used in the UML language to define core objects and the architecture of the application. These objects are created as a class instance and dynamically interact to provide the behavior described by use cases.

Object modeling defines the behavior required by the different classes to ensure the proper implementation of use cases and management rules.

The objects form the basis of the application architecture, they can be reused across application domains or can be identified and derived directly use cases or areas of application. A class is composed by:

- Attributes: Representing data whose values represent the state of the object.
- The method: These are operations applicable to objects.

After presenting the refined data dictionary, we can define the classes and their methods and attributes which are presented in the following table:

Table 25 - Description of Class Methods

Class Name	List Of Attributes	methods	Description
User	id_user	Add_u()	Insert user in data base
	email Full name	update_u()	Update user

	Mobile_phone	consult_u()	Consult user
	Password	delete_u()	Delete user from data base
Document	id_doc	import_d()	Import file or picture to save in data base
	title	open_navigator()	Open app in navigator
	path		
	Category	Download_d()	Download document to pdf
	Description	Search_d()	Search document
		Add_d()	Insert document in data base
		update_d()	Update title and description of document
		move_d()	Move document to any folder
Role		delete_d()	Delete document from data base
	id_role	Add_r()	Insert role
	name_role	update_r()	Update role
folder	type_role	delete_r()	Delete role
	id_f	Add_f()	Add folder in data base
	name_f	update_f()	Update name of

			folder
		delete_f()	Delete folder from data base
		consult_f()	Consult folder

VI.3 Associations Representation

An association is a relationship between two classes (binary association) or more (N-ary association). This is a bidirectional semantic connection that describes the Connections between their bodies. An association therefore indicates that it may have Links between instances of the associated classes.

In addition, the association may have constraints that are ordered, and / or exclusive. In fact there are various types of association:

- Simple association: It represents a pure structural relationship between two classes.

The two classes are conceptually at the same level, not being more important One than the other.

- Association of aggregation: Association non-symmetric expresses a strong coupling, It is a relation of subordination and represents a relation of type "ensemble / element"
- Association of composition: It is a strong aggregation whose life cycles of Component and the compound coincide.
- Generalization Association: It is inheritance that represents a relationship Parent / child. The associations present between the classes of our system.
- Data carrier association: It is an association that contains at least one property Below we present all the associations linking our classes:

Table 26 - Associations linking our classes

Association Name	Type	Class 1	Cardinality		Class 2
is accessed by	Simple Association	Document	0..*	1..*	User
has	Simple Association	User	1..*	1	Role
is accessed by	Simple Association	User	1..*	0..*	Folder
contains	Simple Association	Document	1..*	1..*	Folder
describes	Simple Association	Document	1	1	DocumentArchive
compose	Data Carrier Association	Document	1	1..*	Pages

VI.4 Class Diagram

A class diagram is a collection of static modeling elements (Classes, packages ...), which shows the structure of a model. The class diagram is a static representation of the information system that allows designers to visualize relationships in a system.

The class diagram is considered the most important of the modeling conceptual.

In fact, this is a static view because we do not temporal in the behavior of the system.

The class diagram models the concepts the scope of application as well as the internal concepts created from scratch in the of the implementation of our application.

Each programming language provides a means specific to implement different paradigms (pointers or not, inheritance or not, etc.), but the class diagram makes it possible to model the classes of the system and their relations thirty regardless of the use of a particular programming language.

The main elements of this static view are the classes and their relation.

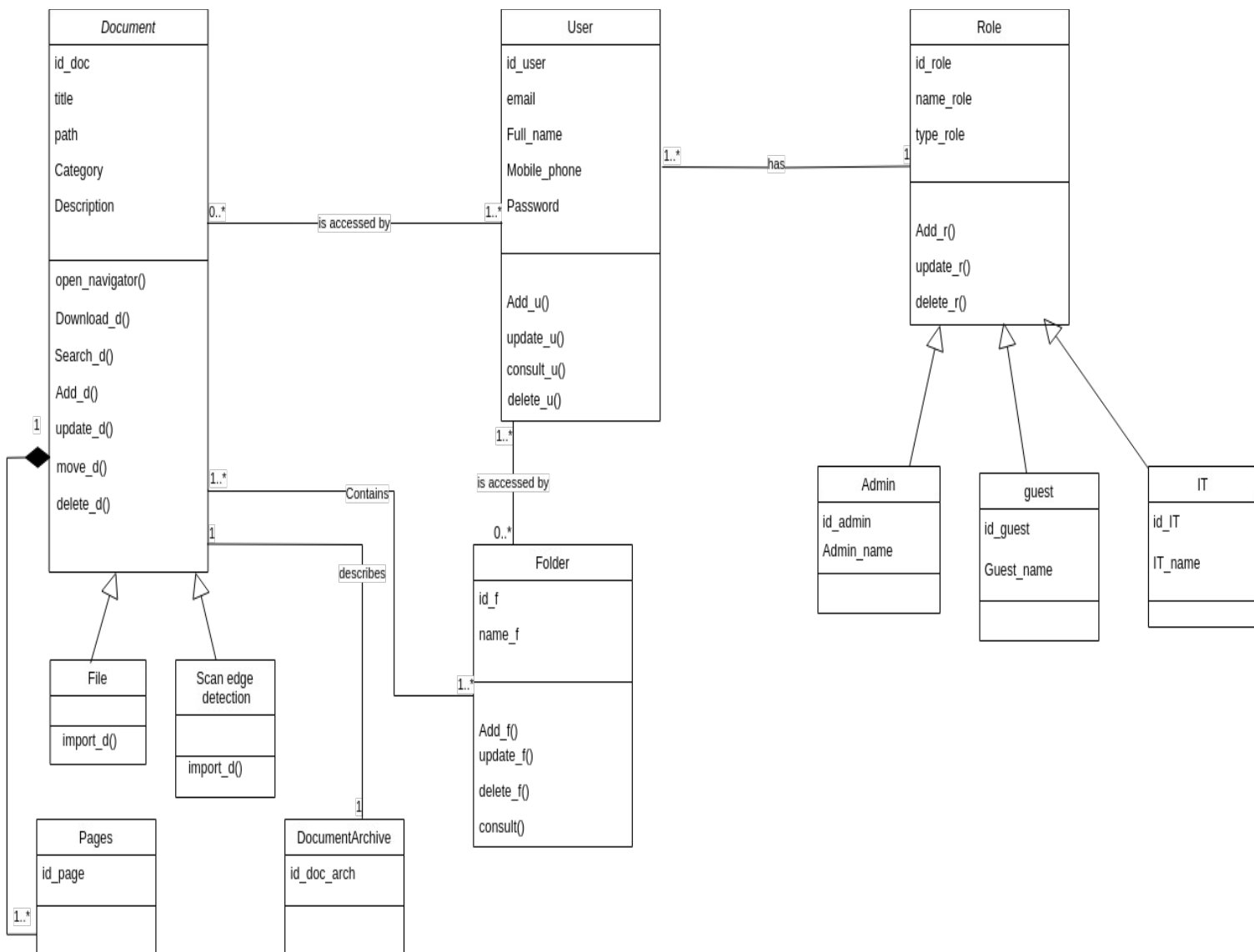


Figure 18 - Class Diagram

VII. ACTIVITY DIAGRAM

Activity diagrams allow to focus on treatment. They are Particularly adapted to the modeling of the flow of control flows and data streams. They allow to represent graphically the behavior of a method or sequence of a use case.

VII.1 System activity diagram

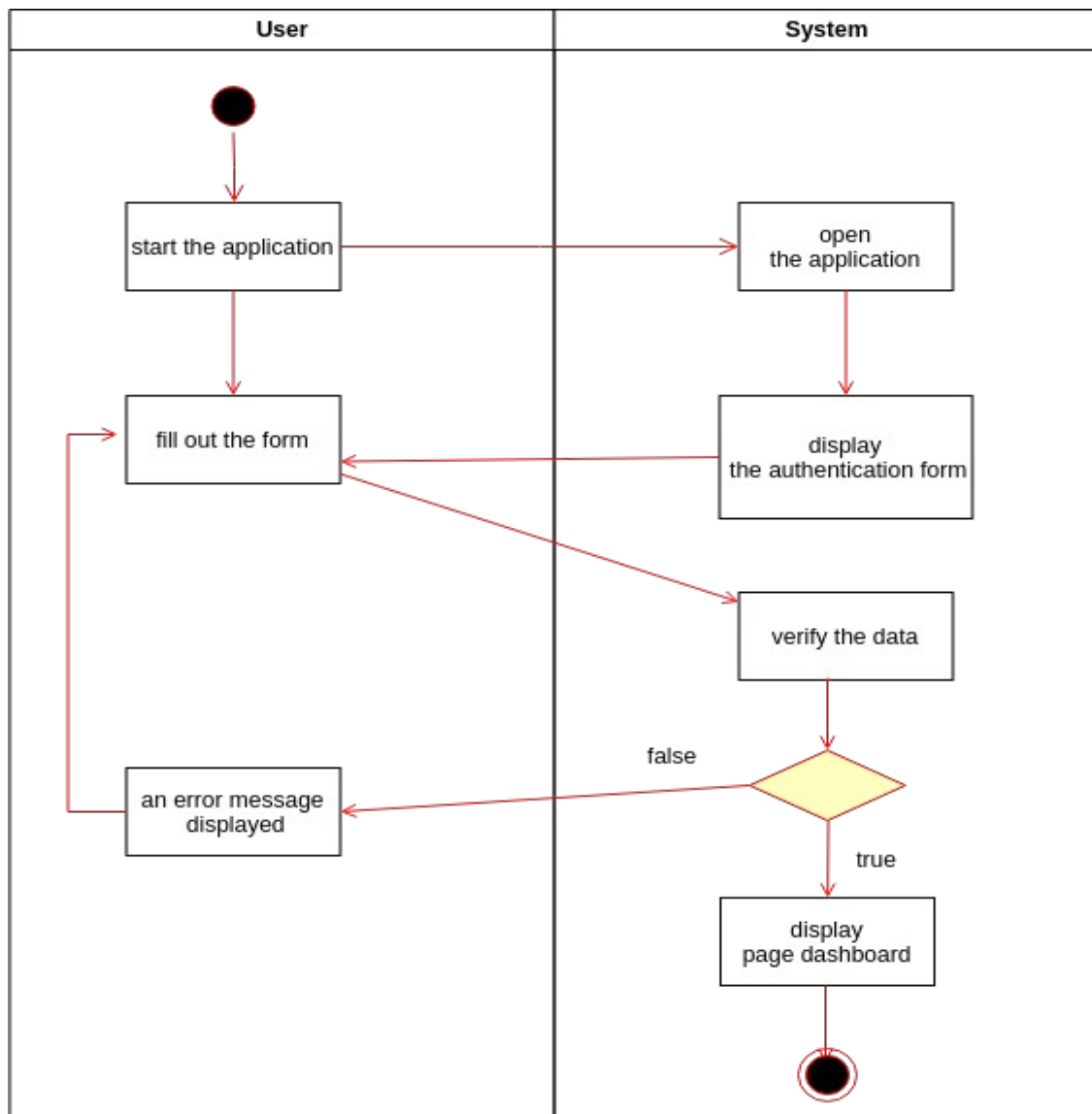
VII.1.1 Definition of system activity diagram

An activity diagram makes it possible to model the behavior of the system, the sequence of which actions and their conditions of execution. Actions are the basic units of system behavior.

An activity diagram allows you to group and dissociate actions. If an action can be divided into several actions in sequence, you can create an activity representing them.

VII.1.2 Presentation of activity diagram of system

VII.1.2.1 Activity diagram use case system "Authentication"

*Figure 19 - system activity diagram "Authentication"*

VII.1.2.2 Use case "Manage User"

VII.1.2.2.1 Use case system activity diagram "Add User"

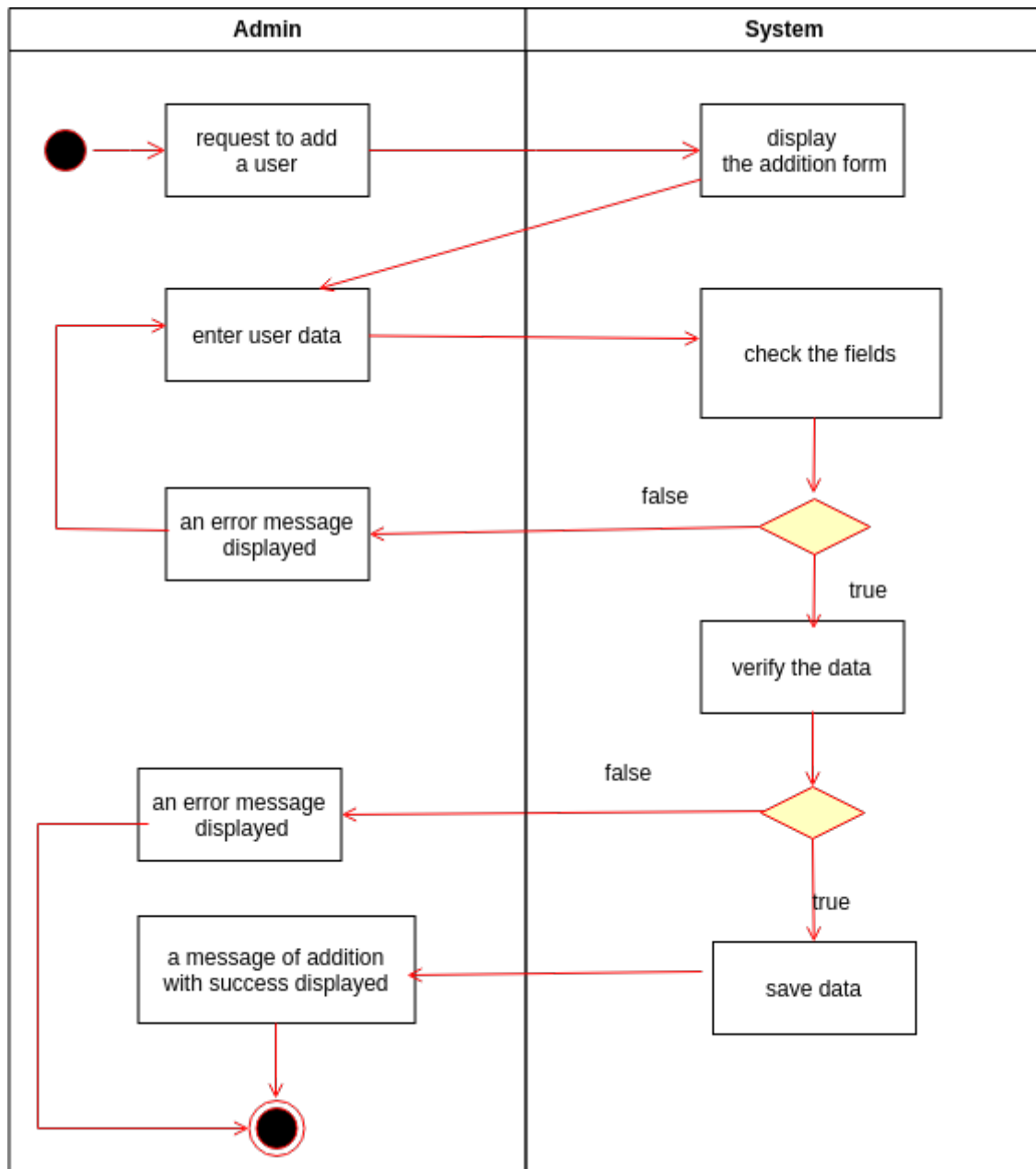


Figure 20 - system activity diagram "Add User"

VII.1.2.2.2 Use case system activity diagram "update User"

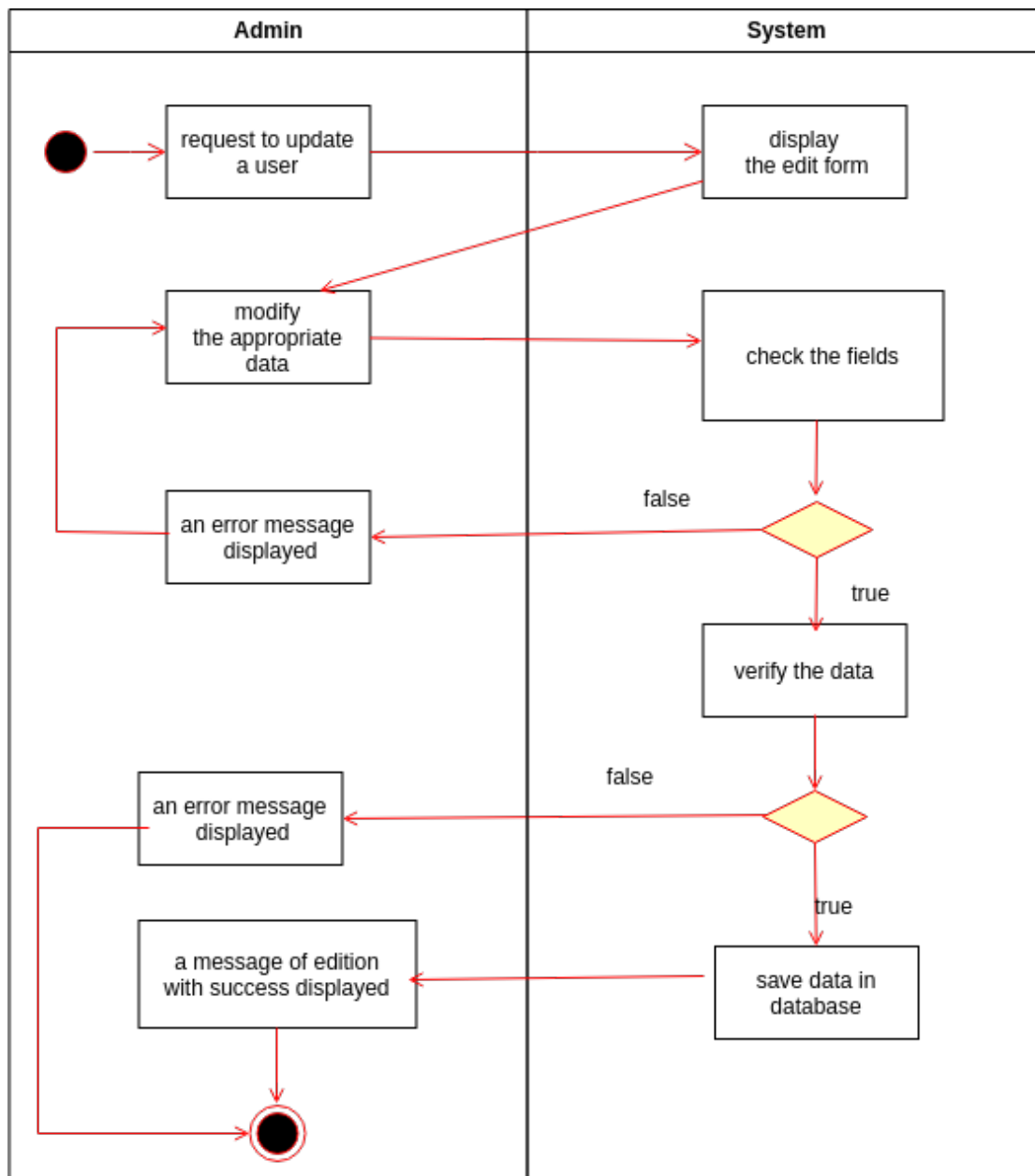


Figure 21 - system activity diagram "update User"

VII.1.2.2.3 Use case system activity diagram "Delete User"

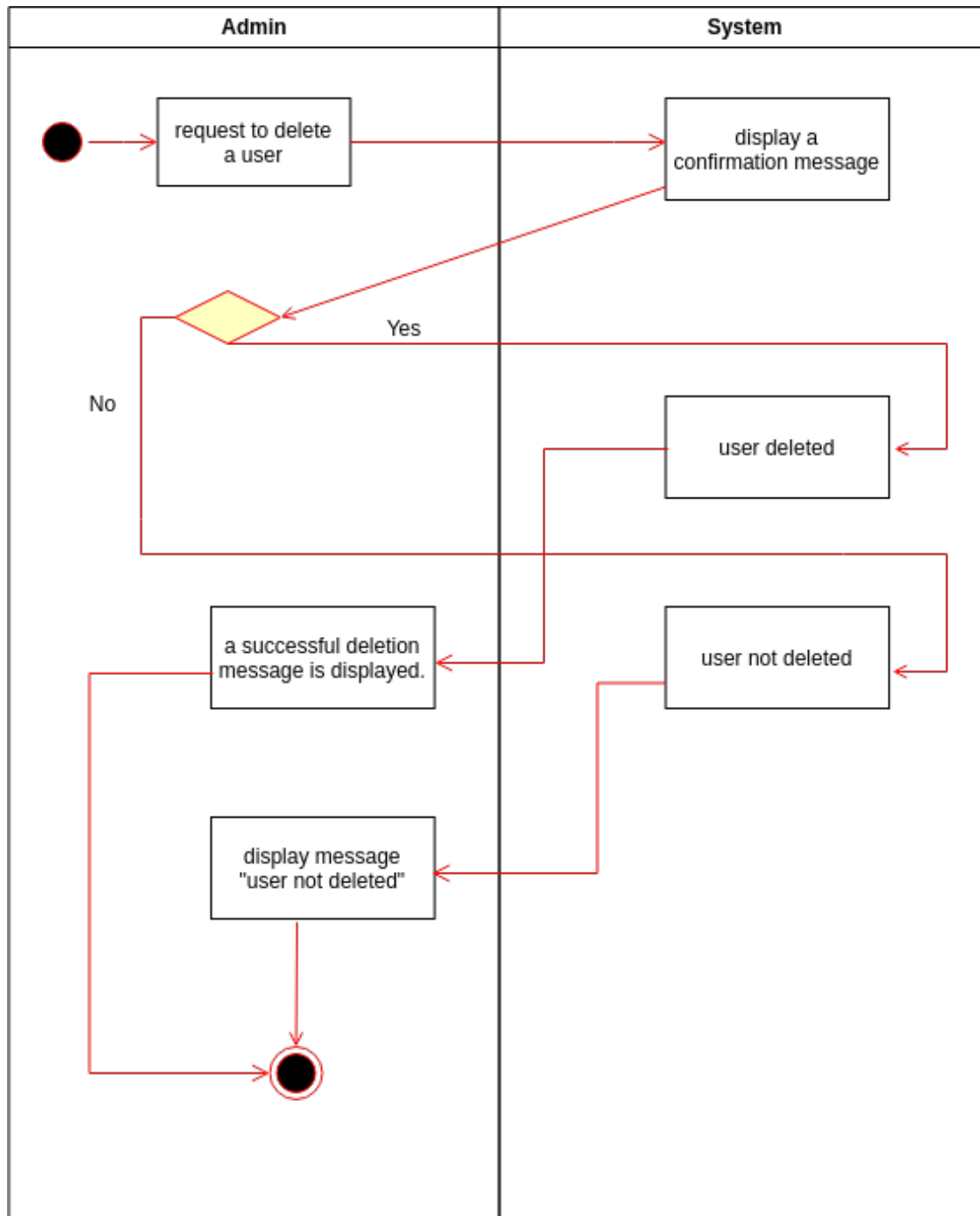


Figure 22 - system activity diagram "Delete User"

VII.1.2.3 Use case "Manage document"

VII.1.2.3.1 Use case system activity diagram "Add document"

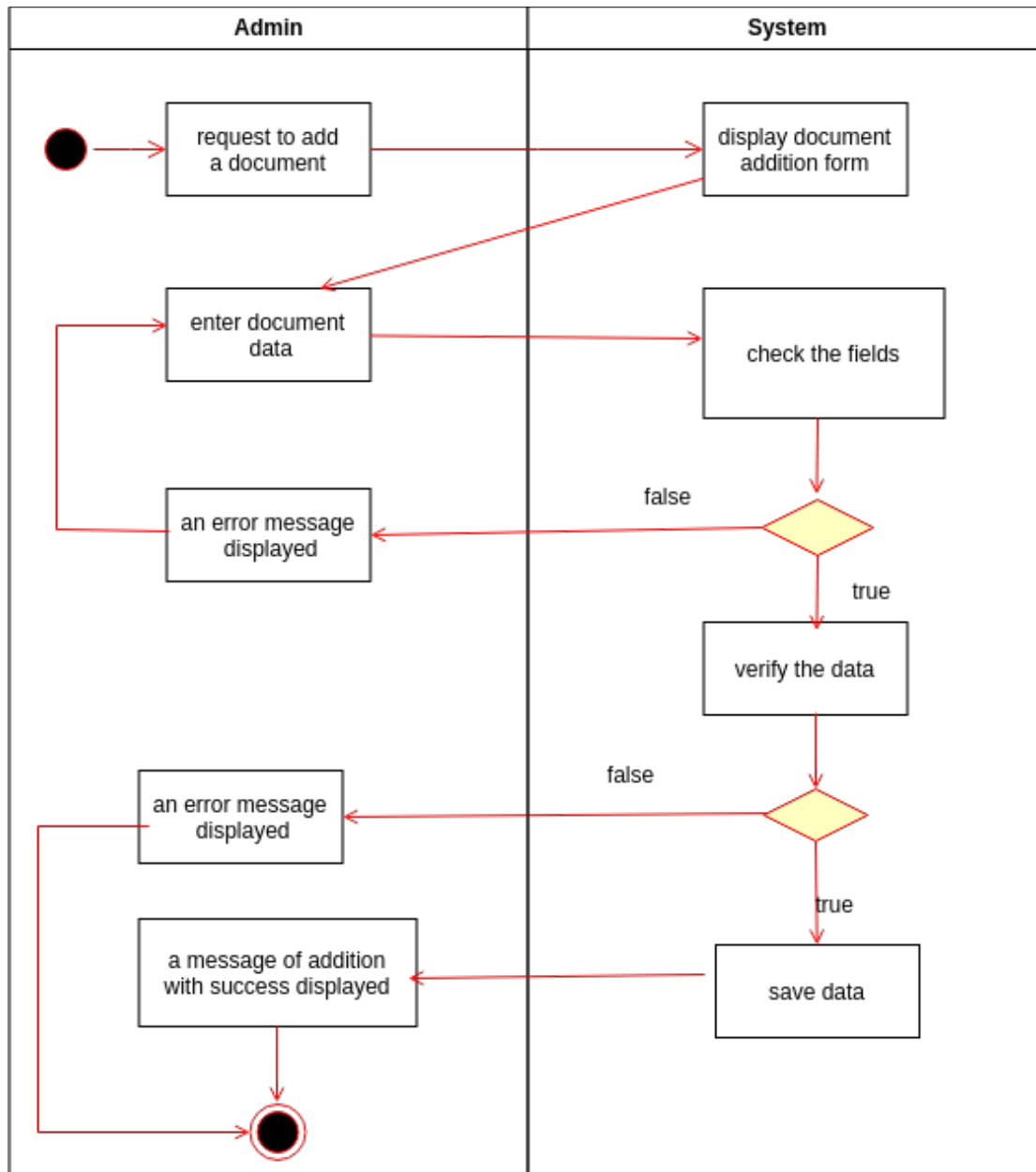


Figure 23 - system activity diagram "Add document"

VII.1.2.3.2 Use case system activity diagram "update document"

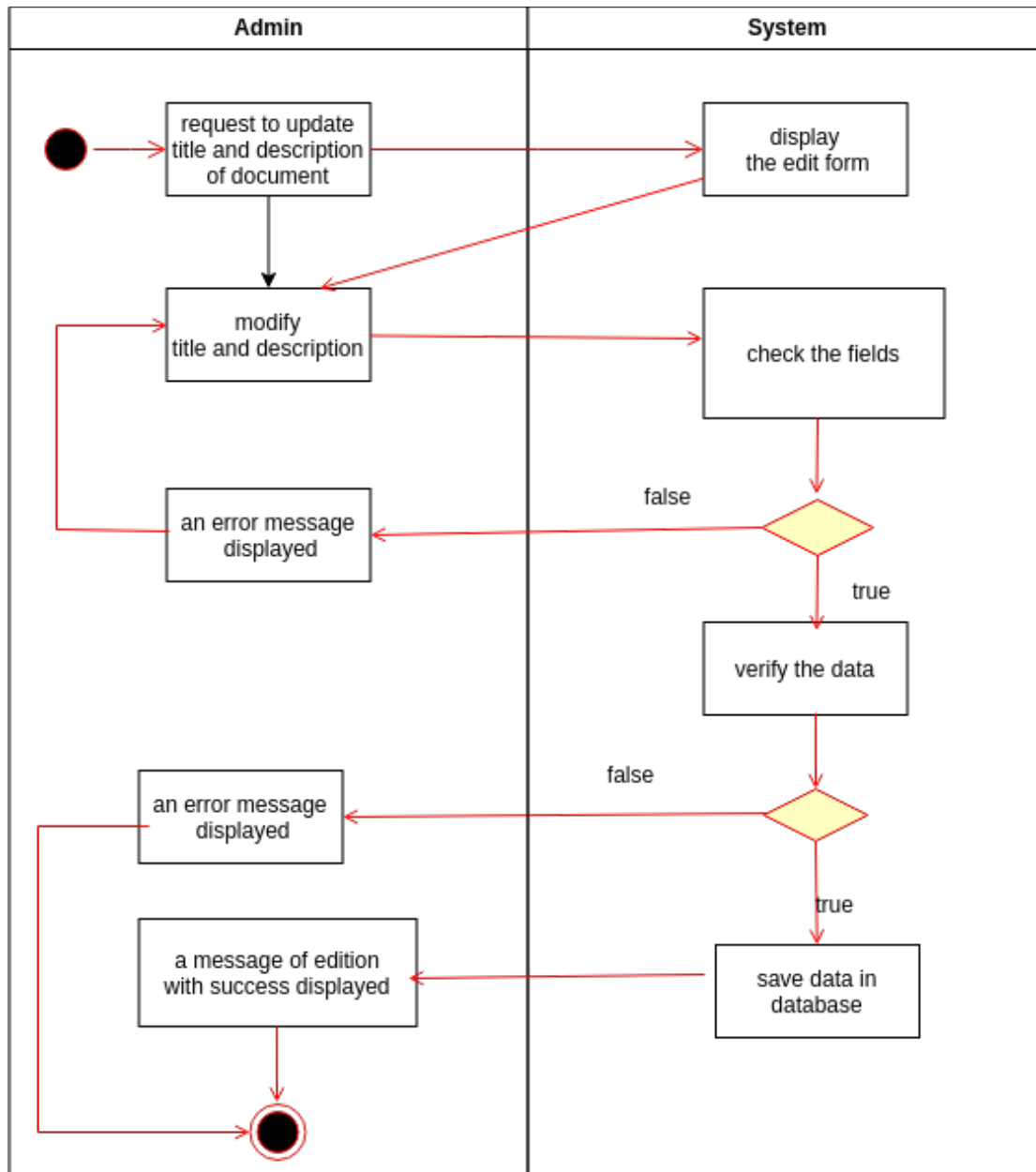
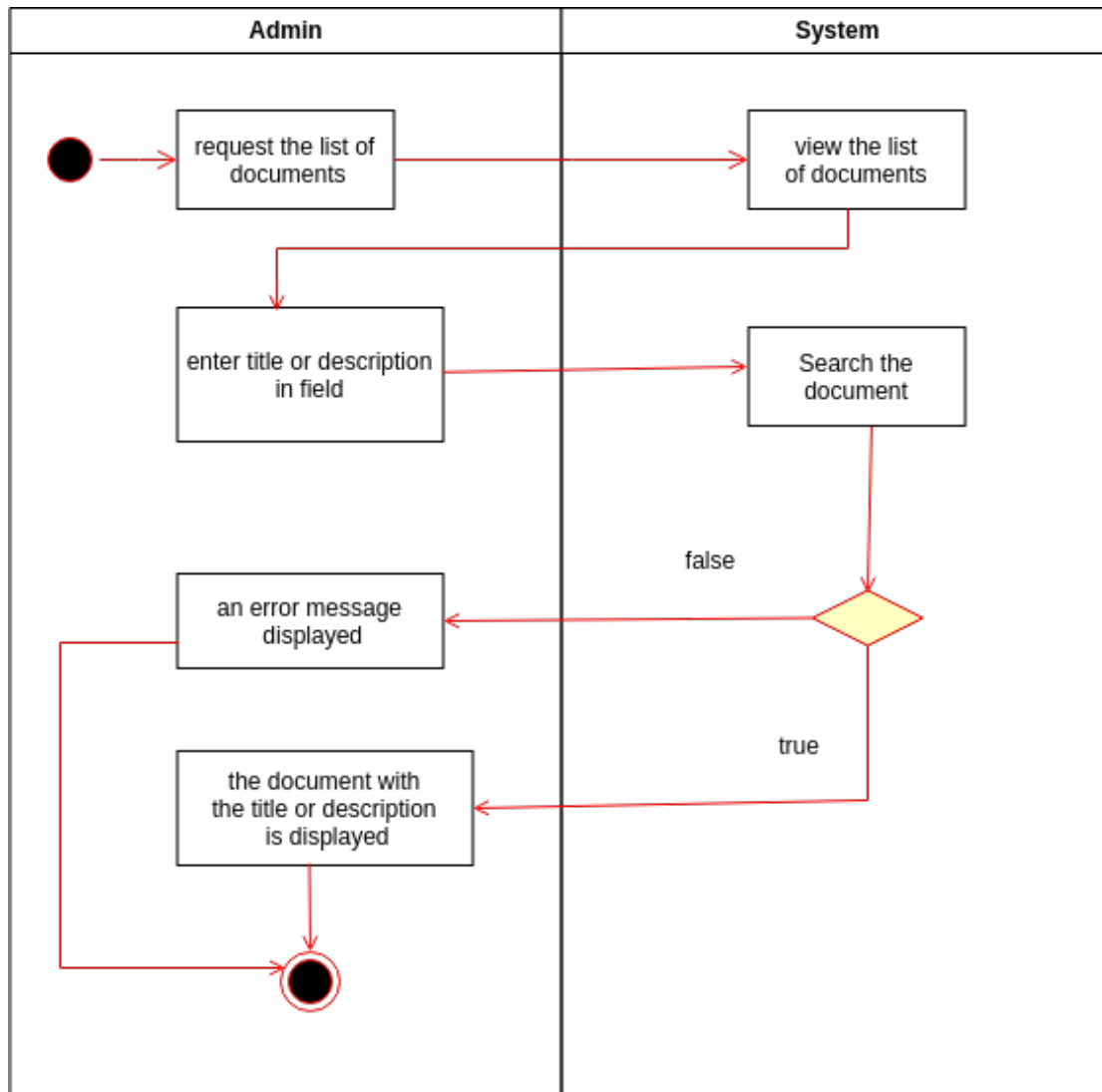


Figure 24 - system activity diagram "update document"

VII.1.2.3.3 Use case system activity diagram "search document"

*Figure 25 - system activity diagram "search document"*

VII.1.2.3.4 Use case system activity diagram "delete document"

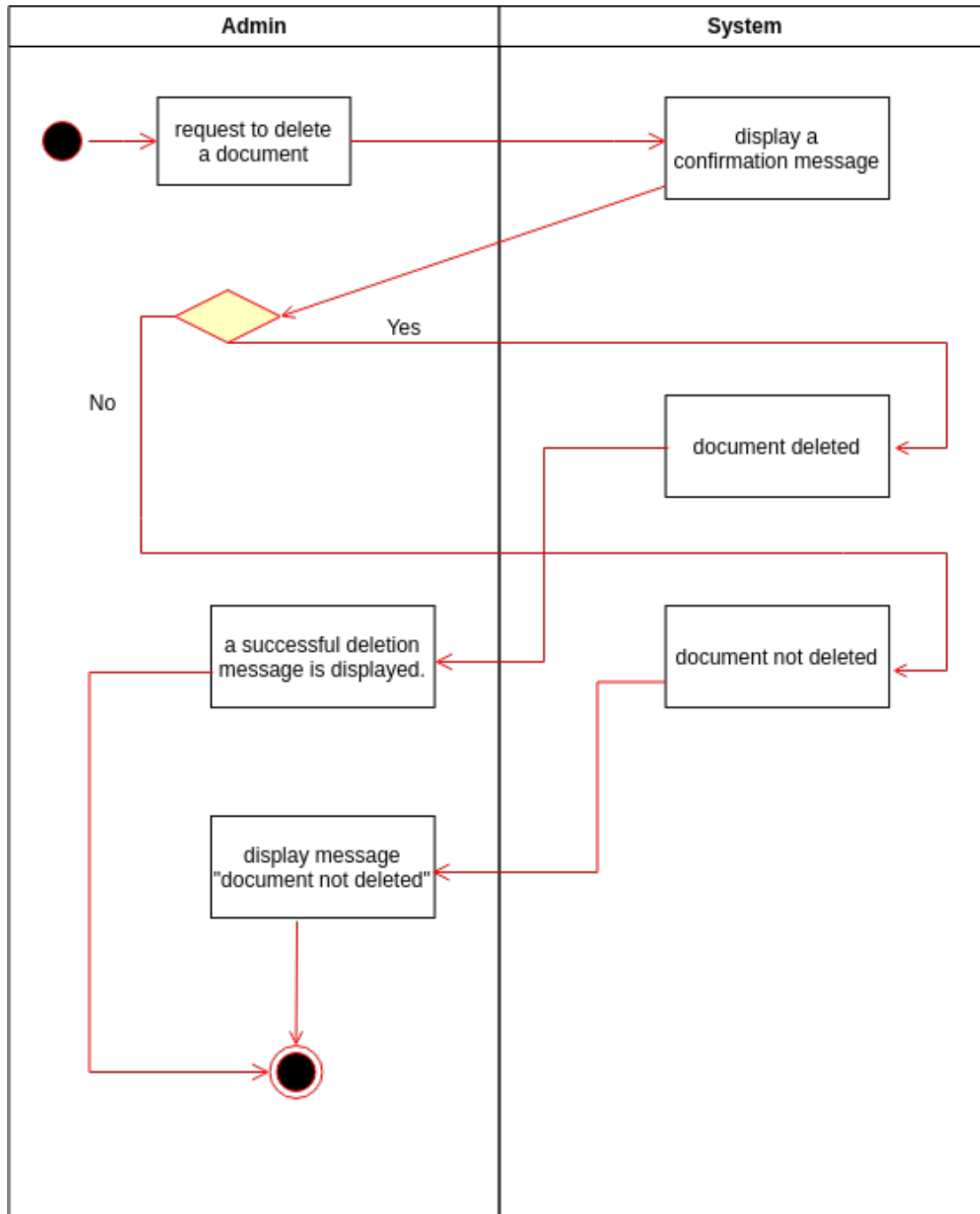


Figure 26 - system activity diagram "delete document"

VII.1.2.3.5 Use case system activity diagram "Download document"

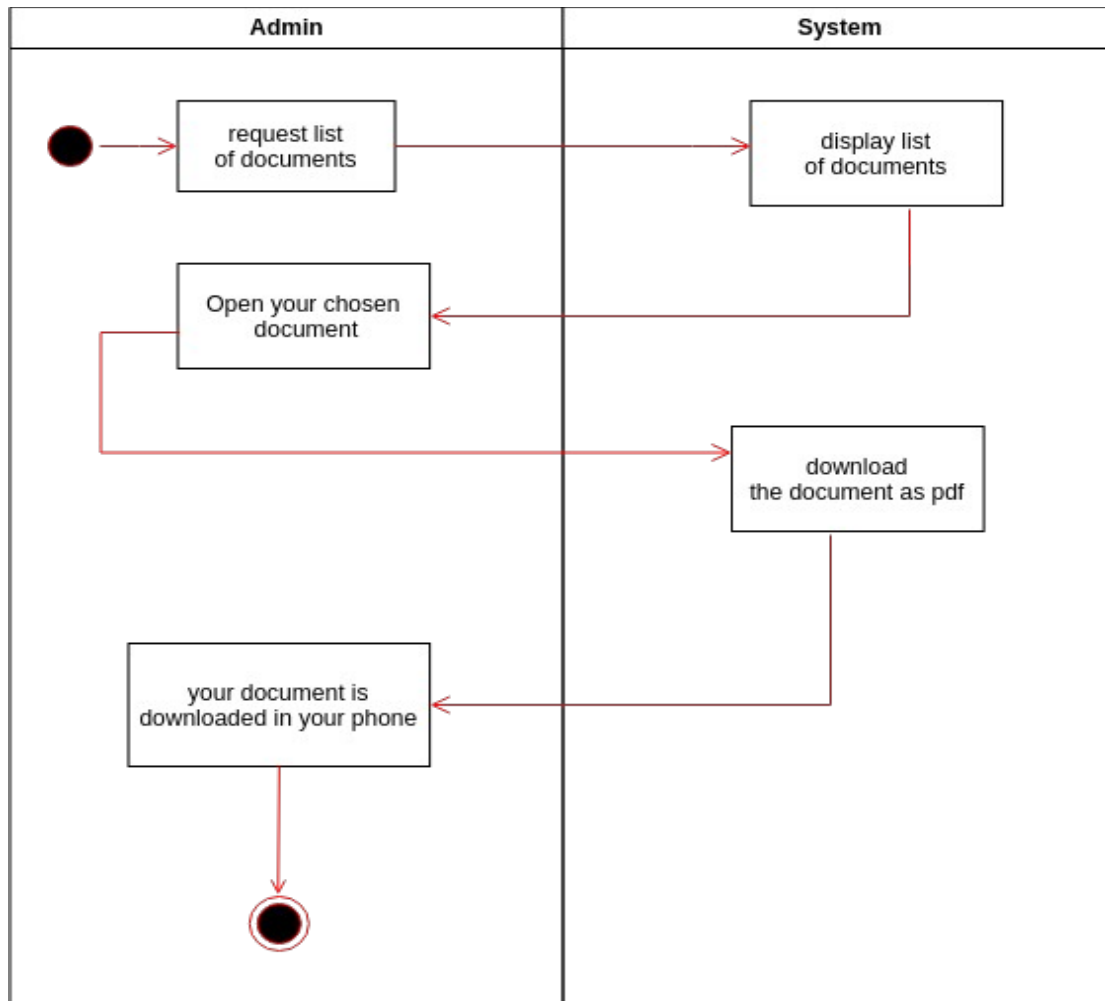


Figure 27 - system activity diagram "Download document"

VIII. CONCLUSION

The conceptual design of our application has allowed us to simplify the understanding of our work while allowing to visually express an object solution. We have presented in this chapter the static and dynamic aspects of our application.

After the reequred specification of the application needs and its design, we continue in next chapter describing realization tools toimplement our solution.

CHAPTER 3: REALIZATION

I. INTRODUCTION

The implementation phase is considered as the final concretization of the whole method Design. First, we carry out a technical study in which we describe the software resources used in the development of our project. We present, in first, our choice of work environment, in which we specify the hardware-software environment we used to realize this application. Then, we detail the architecture, and we present some interfaces made for illustrate how some of the system's activities.

II. TECHNICAL STUDY

This phase consists of adapting the design to the technical architecture. Its objective is to Describe the functional plan and the processing of the solution to be carried out in detail.

II.1 Realization Environment

In this section we will detail the different tools used for the realization of the project. We used the flutter application insured for several reasons, and spring boot for back end.

We preferred it because it is more often present in the offers hosting which allows us to have a greater compatibility, let us better master the flutter, and we were able to include more functions at the application.

With regard to the database system the issue did not arise much as in the majority of the hosting offers.

II.1.1 HARDWARE ENVIRONMENT

The hardware which has favored the realization of our application corresponds to a development computer, having the following characteristics:

Table 27 - Characteristics of the materials used

Characteristics	PC
Brand	SAMSUNG
Processor	Intel® Core™ i7-3630QM CPU @ 2.40GHz × 8
Random Access Memory (RAM)	8.0 GB
Hard disk	512.1 GB + 1 TO Extern
Operating system	Ubuntu 22.04.2 LTS

II.1.2 SOFTWARE AND DEVELOPMENT TOOLS ENVIRONMENT

✓ Dart:

Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development. Dart is designed for a technical envelope that is particularly suited to client development, prioritizing both development (sub-second stateful hotreload) and high-quality production experiences across a wide variety of compilation targets (web, mobile, and desktop).

Dart also forms the foundation of Flutter. Dart provides the language and runtimes that power Flutter apps, but Dart also supports many core developer tasks like formatting, analyzing, and testing code.[3]



Figure 28 - Dart logo

✓ Flutter:

Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems such as iOS and Android, while also allowing applications to interface directly with underlying platform services. The goal is to enable developers to deliver high-performance apps that feel natural on different platforms, embracing differences where they exist while sharing as much code as possible. During development, Flutter apps run in a VM that offers stateful hot reload of changes without needing a full recompile. For release, Flutter apps are compiled directly to machine code, whether Intel x64 or ARM Instructions, or to JavaScript if targeting the web. The framework is open source, with a permissive BSD license, and has a thriving ecosystem of third-party packages that supplement the core library functionality.[4]



Figure 29 - flutter logo

✓ spring boot:

Spring Boot is an open-source Java-based framework used to create a micro-Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. [5]



Figure 30 - spring boot logo

✓ Rest:

An API is a set of definitions and protocols for building and integrating application software. It's sometimes referred to as a contract between an information provider and an information user establishing the content required from the consumer (the call) and the content required by the producer (the response). In other words, if you want to interact with a computer or system to

retrieve information or perform a function, an API helps you communicate what you want to that system so it can understand and fulfill the request. [6]



Figure 31 - Rest logo

✓ **Firebase**

Firebase is a Cloud-hosted, NoSQL database that uses a document-model. It can be horizontally scaled while letting you store and synchronize data in real-time among users. This is great for applications that are used across multiple devices such as mobile applications.

Firebase is optimized for offline use with strong user-based security that allows for serverless based apps as well. Firebase is built on the Google infrastructure and is built to scale automatically.

In addition to standard NoSQL database functionality, Firebase includes analytics, authentication, performance monitoring, messaging, crash reporting and much more. Because it is a Google product, there is also integration into a lot of other products. This includes integration with Google Ads, AdMob, Google Marketing Platform, the Play Store, Data Studio, BigQuery, Slack, Jira, and more. The Firebase APIs are packaged into a single SDK that can be expanded to multiple platforms and languages. This includes C++ and Unity, which are both popular for mobile development. [7]



Figure 32 - Firebase logo

✓ **IntelliJ IDEA:**

IntelliJ IDEA is the leading IDE for Java and Kotlin development. It helps you stay productive with a suite of efficiency-enhancing features such as intelligent coding assistance, reliable refactorings, instant code navigation, built-in developer tools, web and enterprise development support, and much more.[8]

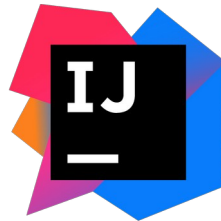


Figure 33 - IntelliJ logo

✓ **Android Studio:**

Android's open-source model encourages innovation by giving device makers the freedom to customize their phones and the Android OS, as well as preinstall any apps they choose. So consumers get more choices when it comes to devices and apps. Android is open to everyone: developers, designers and device makers.

That means more people can experiment, imagine and create things the world has never seen. On Android, you get to decide when and if your data is shared, like your Web & App Activity or Location History. If an app accesses your location while you are not using it, you'll get a notification. And if you ever want to change permissions, all your Privacy settings are in one place. It's privacy, with you in the driver's seat.[9]



Figure 34 - Android logo

✓ **visual studio code:**

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). [10]

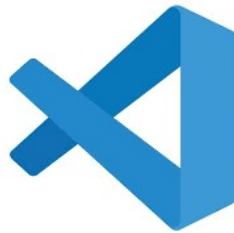


Figure 35 - visual studio logo

✓ **git:**

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

It is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows. The Git feature that really makes it stand apart from nearly every other SCM out there is its branching model.

Git allows and encourages you to have multiple local branches that can be entirely independent of each other. The creation, merging, and deletion of those lines of development takes seconds.[11]



Figure 36 - git logo

✓ **GitHub:**

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. GitHub offers essentials like repositories, branches, commits, and pull requests. You'll create your own Hello World repository and learn GitHub's pull request workflow, a popular way to create and review code.[12]



Figure 37 - github logo

✓ **Draw.io:**

draw.io is an open source technology stack for building diagramming applications, and the world's most widely used browser-based end-user diagramming software.

Our model is based on the viral effect of a free, open source application.[13]

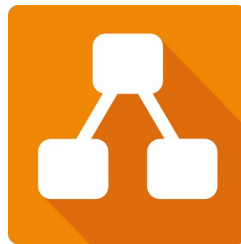


Figure 38 - draw.io logo

✓ **railway.app:**

Railway is a deployment platform where you can provision infrastructure, develop with that infrastructure locally, and then deploy to the cloud.[14]



Figure 39 - railway.app logo

II.2 Choice Of Architecture

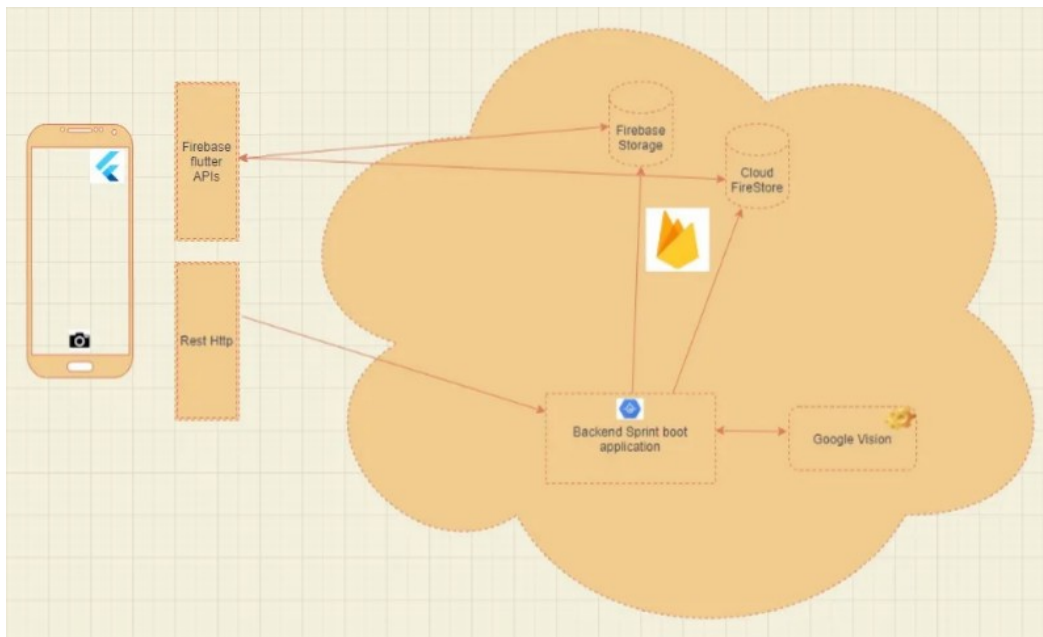


Figure 40 - Application Architecture

✓ **Firebase Flutter APIs:**

- ◆ Firebase Flutter APIs provide a set of tools and services for integrating Firebase services into a Flutter application.
- ◆ It allows developers to interact with Firebase services such as Firebase Authentication, Cloud Firestore, and Firebase Storage.

✓ **Firebase Storage:**

- ◆ Firebase Storage is a cloud-based storage solution provided by Firebase.
- ◆ It can be used to store and retrieve user-generated content like images, videos, and other files.
- ◆ Firebase Storage is often integrated with Firebase Flutter APIs to handle media storage operations in a Flutter application.

✓ **Cloud Firestore:**

- ◆ Cloud Firestore is a NoSQL document database offered by Firebase.

- ◆ It provides real-time synchronization and offline support, making it suitable for building reactive and responsive applications.
- ◆ Firebase Flutter APIs enable interaction with Cloud Firestore, allowing developers to store, retrieve, and manage data in the database.
- ✓ **Rest Http:**
 - ◆ Rest Http is a package in Flutter that facilitates making RESTful API calls.
 - ◆ It allows Flutter applications to communicate with external APIs or services using HTTP requests and handle responses.
 - ◆ Rest Http can be used in combination with Firebase Flutter APIs or independently to interact with RESTful APIs from a Flutter application.
- ✓ **Backend Spring Boot application:**
 - ◆ The Backend Spring Boot application serves as the server-side component of the overall application architecture.
 - ◆ It is responsible for handling requests from the Flutter application and performing business logic, data processing, and database operations.
 - ◆ The backend application can integrate with Firebase services, such as Cloud Firestore, to retrieve or update data based on the requests received from the Flutter app.
- ✓ **Google Vision:**
 - ◆ Google Vision provides powerful image analysis capabilities through its APIs.
 - ◆ It offers features like image recognition, text extraction, and object detection.
 - ◆ Google Vision can be integrated into a Flutter application to perform image analysis tasks on user-uploaded images.
 - ◆ The Flutter app can use Firebase Flutter APIs to upload the images to Firebase Storage, and then Google Vision APIs can be utilized to analyze and extract information from those images.

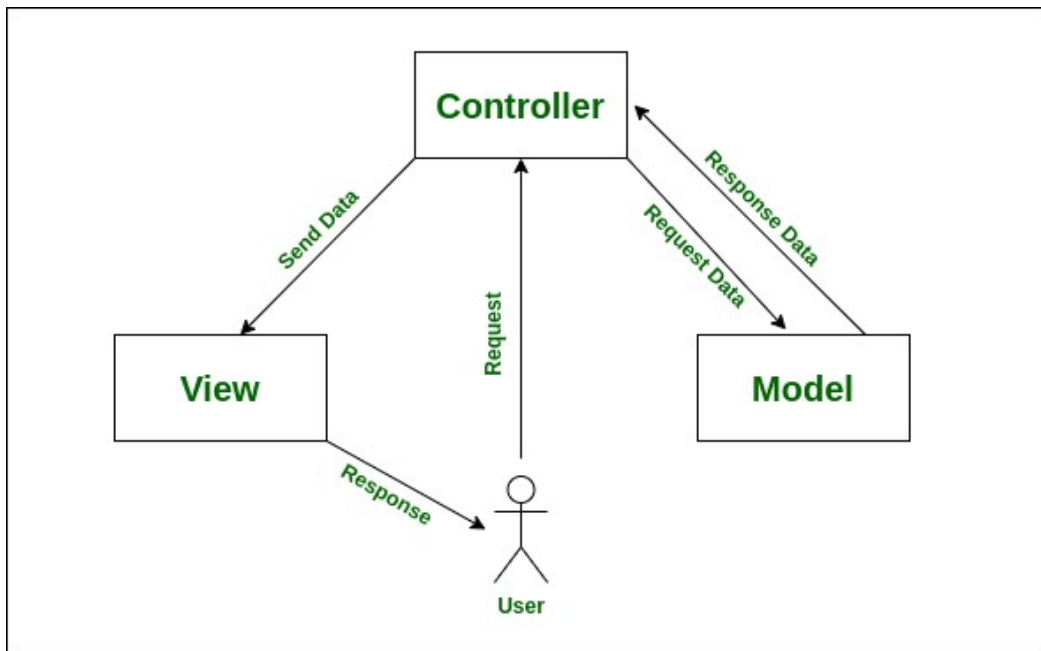


Figure 41 - mvc model principle

MVC, short for Model, View, and Controller, is a methodology or architectural pattern used for efficiently relating the user interface to underlying data models and organizing to relate the application code. MVC is primarily used to separate an application into three main components: Model, View, and Controller.

✓ **Three Levels Of MVC Model:**

➔ **Model:** This level is considered the lowest level when compared with the View and Controller. It primarily represents the data to the user and defines the storage of all the application's data objects. It also contains the application logic.

➔ **Views:** This level is majorly associated with the User Interface(UI) and it is used to provide the visual representation of the MVC model. In simpler terms, this level deals with displaying the actual output to the user. It also handles the communication between the user (inputs, requests, etc.) and the controller.

➔ **Controller:** This level takes care of the request handler. The controller completes the cycle of taking the user output, converting it into desired messages.

✓ **Benefits of using MVC :**

- ➔ **Organizes large-size web applications:** As there is segregation of the code among the three levels, it becomes extremely easy to divide and organize web application logic into large-scale applications (which are required to be managed by large teams of developers). The major advantage of using such code practices is that it helps to find specific portions of code quickly and allows the addition of new functionality with ease.
- ➔ **Supports Asynchronous Method Invocation (AMI):** Since the MVC architecture works well with JavaScript and its frameworks, it is no surprise that it also supports the use of Asynchronous Method Invocation (AMI), allowing developers to build faster loading web applications. It means that MVC applications can be made to work even with PDF files, site-specific browsers, and also for desktop widgets.
- ➔ **Easily Modifiable :** Using the MVC methodology allows easy modification of the entire application. Adding/updating the new type of views is simplified in the MVC pattern (as a single section is independent of the other sections). So, any changes in a certain section of the application will never affect the entire architecture. This, in turn, will help to increase the flexibility and scalability of the application.
- ➔ **Faster Development Process:** As there is segregation of the code among the three levels, developing web applications using the MVC model allows one developer to work on a particular section (say, the view) while another can work on any other section (say, the controller) simultaneously. This allows for easy implementation of business logic as well as helps to accelerate the development process fourfold. It has been observed that when compared to other development models, the MVC model ends up showing higher development speeds (up to three times).
- ➔ **Easy planning and maintenance :** The MVC paradigm is helpful during the initial planning phase of the application because it gives the developer an outline of how to arrange their ideas into actual code. It is also a great tool to help limit code duplication, and allow easy maintenance of the application.

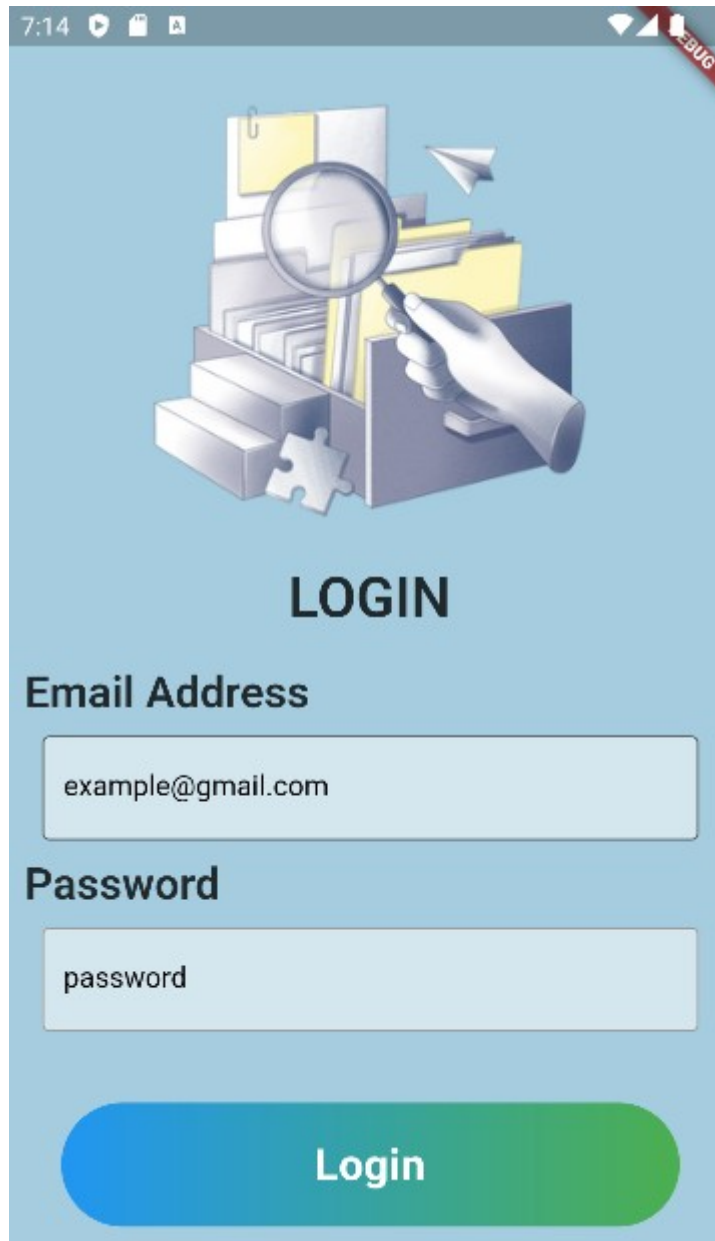
- ➔ **Returns data without formatting :** By returning un-formatted data, the MVC framework empowers you to create your own view engine. For example, any type of data can be formatted using HTML, but with the MVC framework, you can also format the data using the Macromedia Flash or Dream viewer. It is helpful for the developers because the same components can be re-used with any interface.
- ➔ **Supports TTD (test-driven development) :** A major advantage of the MVC pattern is that it simplifies the testing process by a great deal. It makes it easier to debug large-scale applications as multiple levels are structurally defined and properly written in the application. Thus making it trouble-free to develop an application with unit tests.
- ➔ **Multiple Views :** In the MVC architecture, developing different view components for your model component is easily achievable. It empowers you to develop different view components, thus limiting code duplication as it separates data and business logic.
- ➔ **SEO-Friendly Platform:** The MVC platform hugely supports the development of SEO-friendly web applications. To generate more visits from a particular application, MVC provides an easy way out to develop SEO-friendly RESTful URLs.[15]

III. PRODUCTION OF THE PROGRAM

III.1 Front-End

III.1.1 Authentication interface

As a registered user, the workspace allows him certain features. These the latter depend on the user's role. Once the user launches the application, the initial interface is as follows :



7:14

DEBUG

LOGIN

Email Address

example@gmail.com

Password

password

Login

Figure 42 - interface authentication

Fill in the two lines with your login details; your password appears in the form of a star (****) to prevent someone from reading on the screen the word of pass as you type. Of course, you validate your entry by the "Login" button. Two possible cases can occur :

- The email and / or password is incorrect so return to the same interface.
- Then the identification is successful and you access the main interface.

III.1.2 Dashboard interface

After successful authentication, the welcome interface presented below is displayed **Dashboard** contains diagrams to display the number of files per folder.

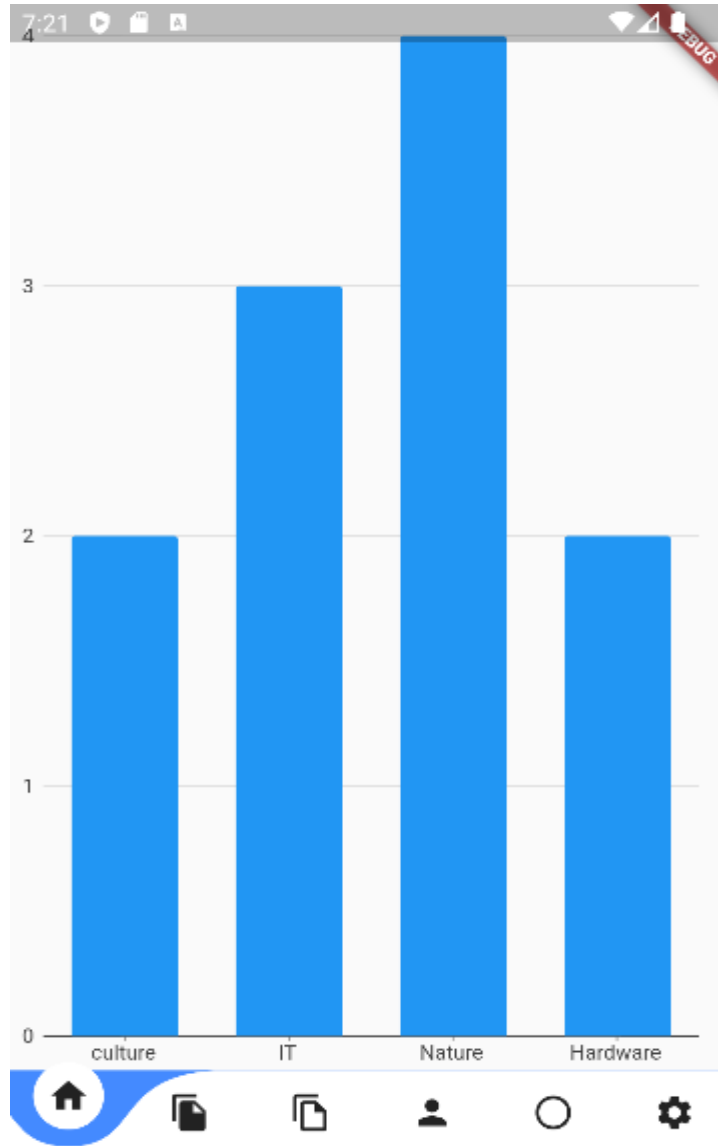


Figure 43 - interface dashboard

III.1.3 List documents

This interface represents the list of all the documents saved in the application.

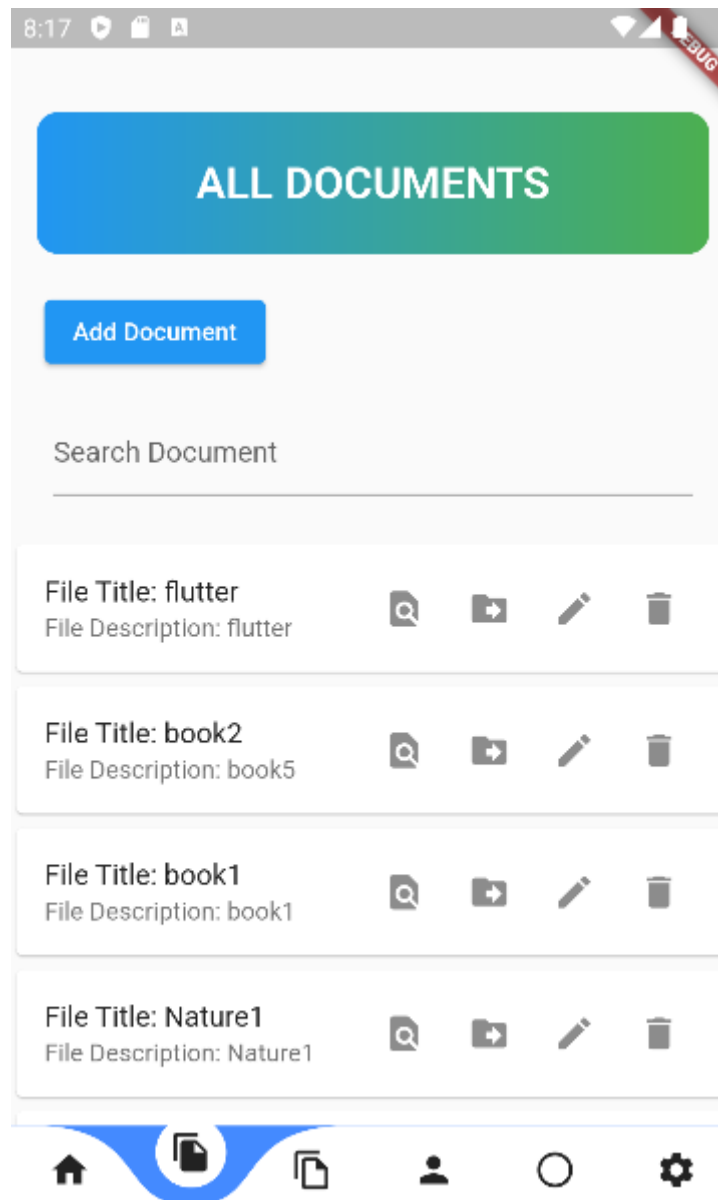


Figure 44 - interface list documents



-Open the document in browser to view picture, we can export picture to pdf.



-Move the document to any folder choosen



-Edit the title and description of document.



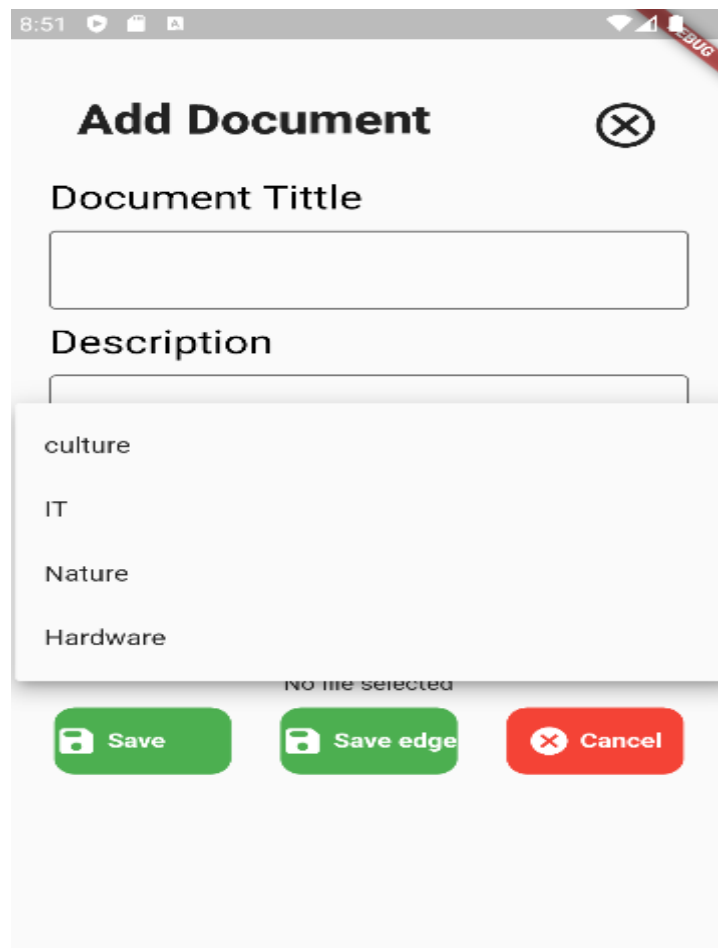
-Delete document from platform.

Search Document

-Search document by title or description.

III.1.4 Add document

fill out form title, description, choose the folder and import document from gallery or we can scan by camera "scan edge"



The screenshot displays the 'Add Document' interface on a mobile device. At the top, the title 'Add Document' is centered, with a close button (X in a circle) on the right. Below the title, there are two input fields: 'Document Tittle' and 'Description'. A folder selection dropdown menu is open, showing a list of categories: 'culture', 'IT', 'Nature', and 'Hardware'. Below the dropdown, the text 'No file selected' is visible. At the bottom, there are three buttons: 'Save' (green), 'Save edge' (green), and 'Cancel' (red). The status bar at the top shows the time 8:51 and various icons.

Figure 45 - interface add document

III.1.5 interface list folders

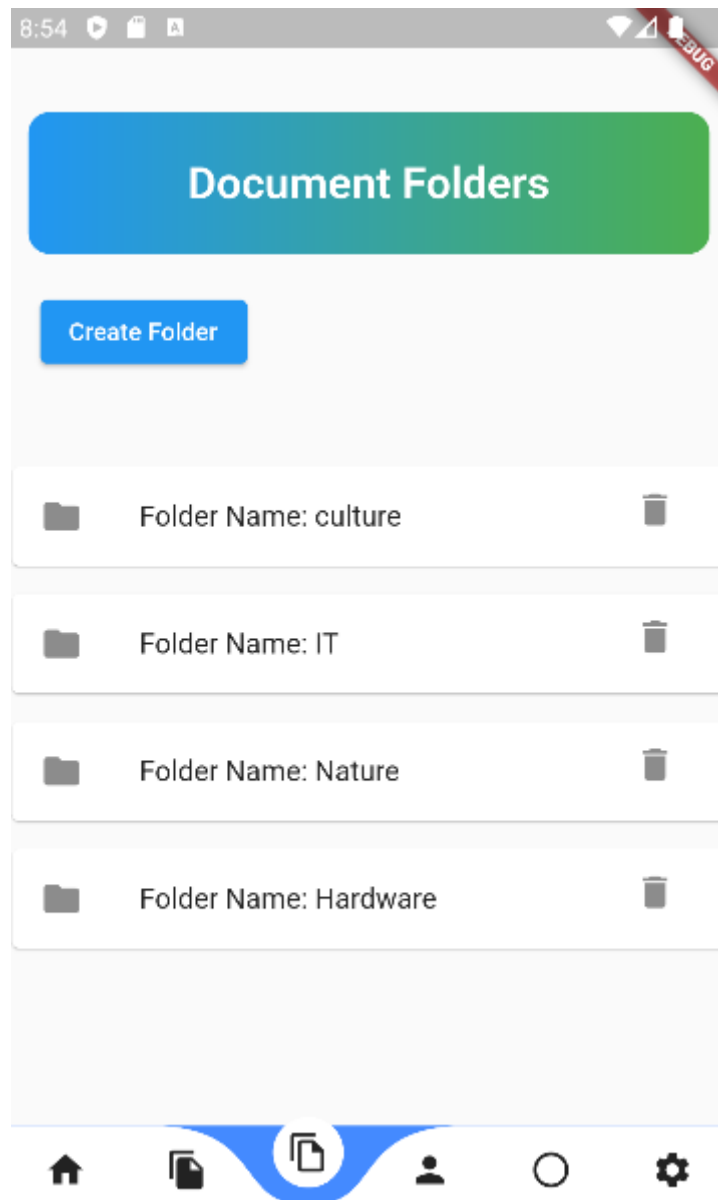


Figure 46 - interface List folder

III.1.6 interface folders details

This interface has a list of documents on each folder for example: tap on Folder Name IT.

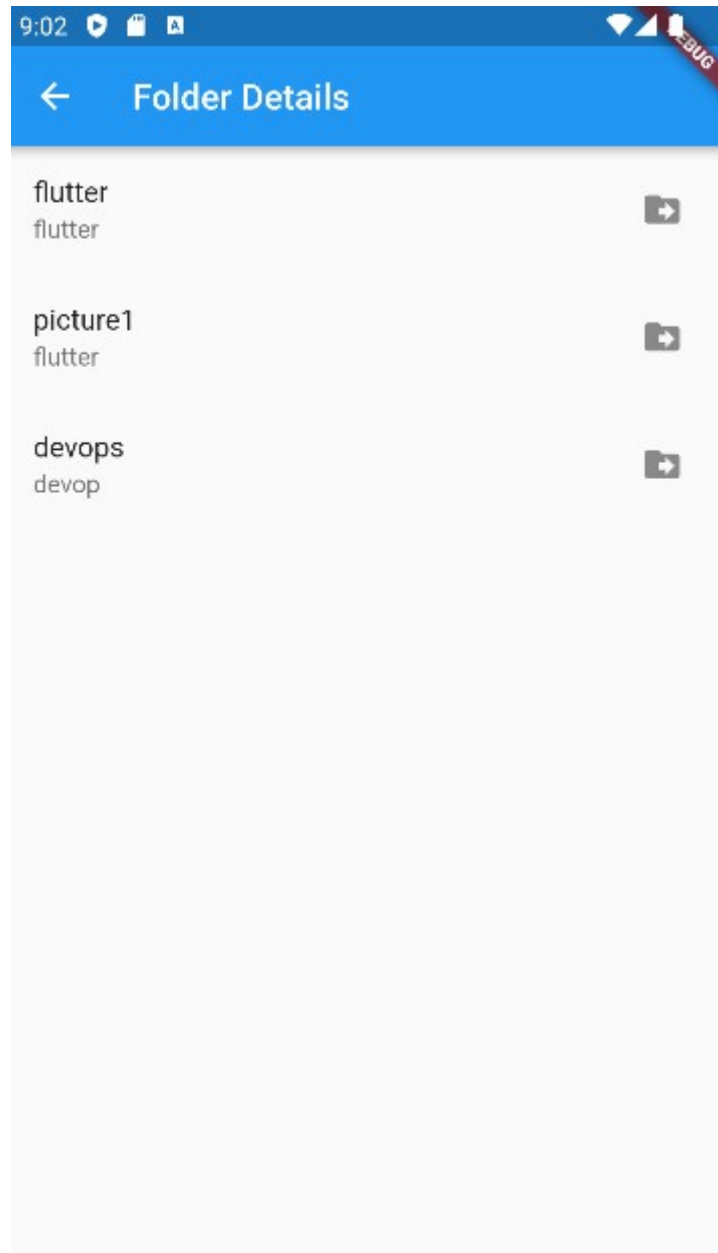


Figure 47 - interface folder "IT"

 - we can to move documents to any folder

III.1.7 interface audit

Provide detailed information on each addition made by the user on each document. The information contains file title, user name, date and hour uploaded document.

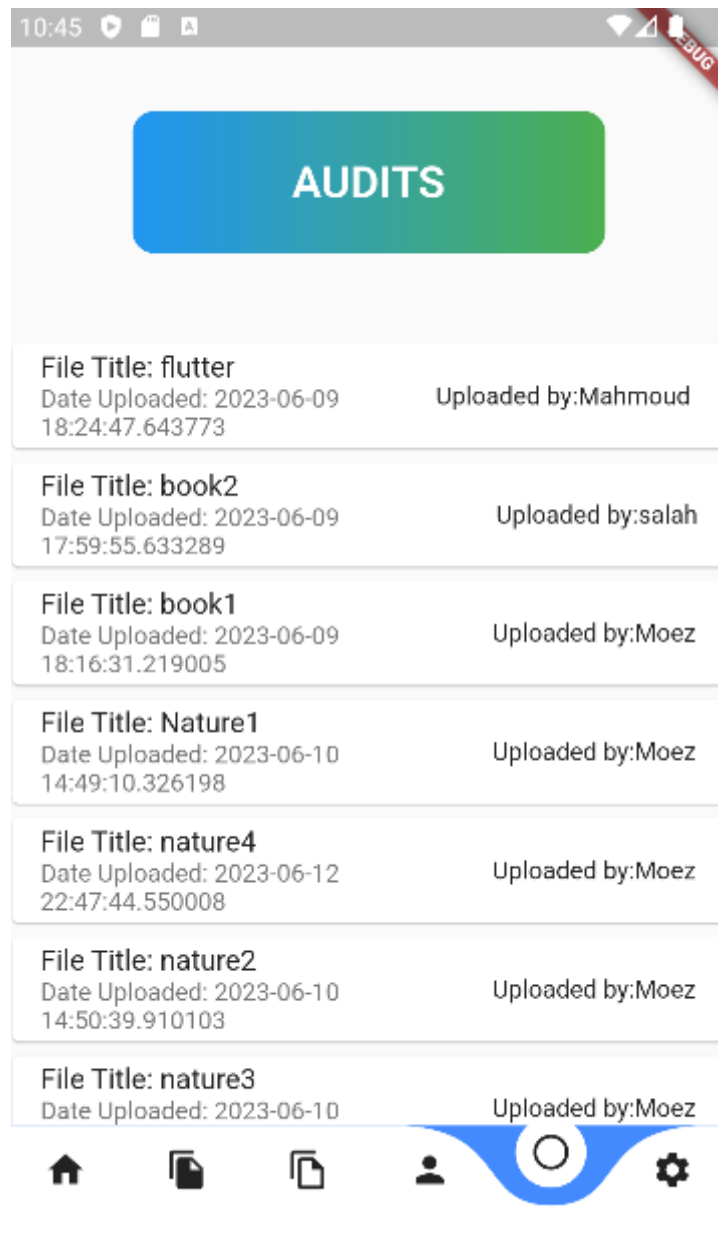
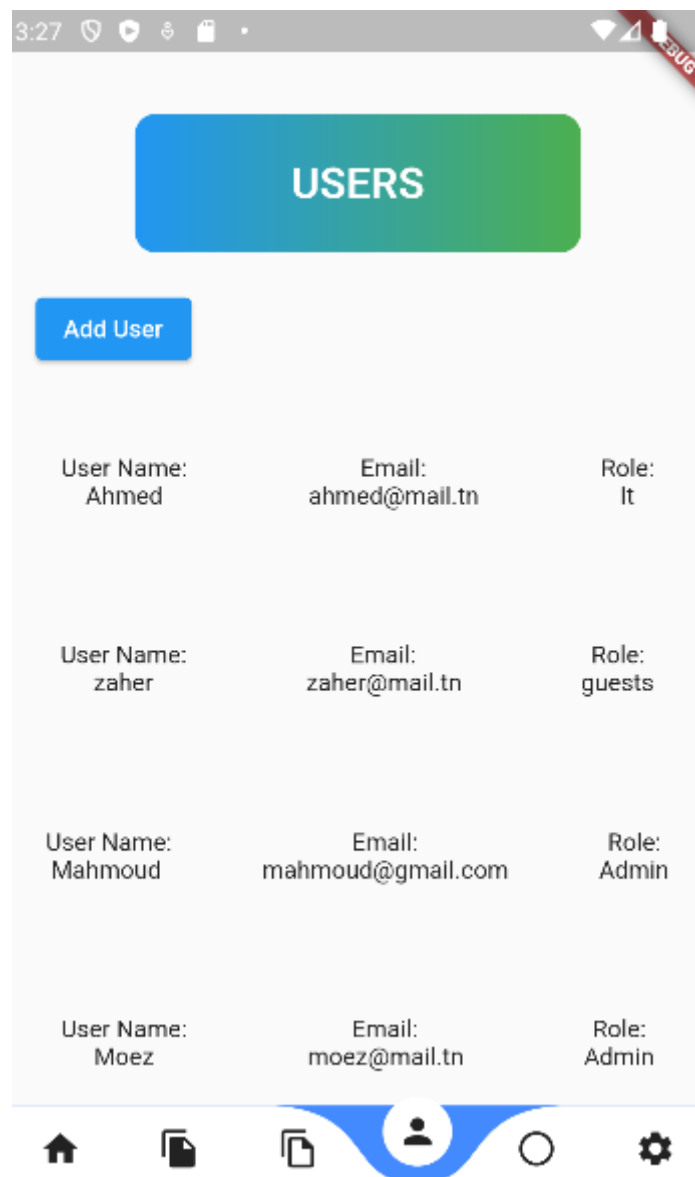


Figure 48 - interface audit

III.1.8 List users and administrators

*Figure 49 - interface list users*

III.1.9 Add users with their role

This interface allows the Admin to add a new user of the application by the allocation of : The basic information which is: the E-mail name to create an account to each user Professional information that are: the role (admin,it,guests)

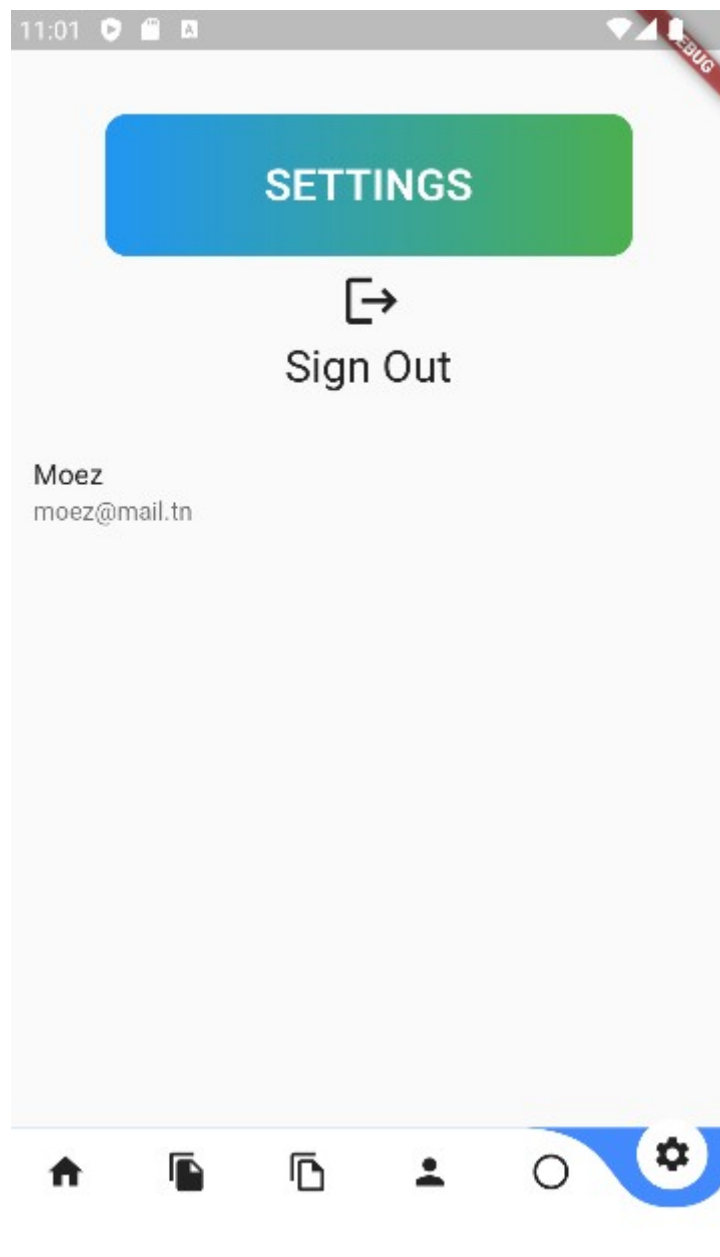
The screenshot shows a mobile application interface for adding a new user. The form consists of the following elements:

- Name:** A text input field containing the value "mohamed".
- Mobile Number:** A text input field containing the value "98741258".
- Email:** A text input field containing the value "mohamed@mail.com".
- Password:** A text input field with masked characters ".....".
- Confirm Password:** A text input field (partially visible).
- Role Selection:** A dropdown menu with three options: "Admin", "It", and "guest".
- Buttons:** A green "Save" button and a red "Cancel" button at the bottom.

A red banner with the word "DEBUG" is located in the top right corner of the screen.

Figure 50 - interface add users

III.1.10 interface Settings

*Figure 51 - interface Settings*

III.2 Back-End

III.2.1 Adding google firebase dependency to spring boot (Maven)

We have our Firebase project ready, it's time to code the server component that will send notifications. Besides the regular Spring Boot starters for an MVC application, we must also add the firebase-admin dependency:

```
</dependency>
<dependency>
  <groupId>com.google.firebase</groupId>
  <artifactId>firebase-admin</artifactId>
  <version>9.1.1</version>
</dependency>
</dependencies>
```

Figure 52 - dependency firebase in pom.xml

III.2.2 Intialization Firebase

Initializing Firebase utilizes the Spring framework and the `@PostConstruct` annotation to execute the `initialization()` method after the bean has been constructed the `FileInputStream` is instantiated with the correct file path, allowing the "flutter.json" file to be opened.

```
@Service
public class FirebaseInitialization {
    @PostConstruct
    public void initialization(){

        FileInputStream serviceAccount =
            null;
        try {
            serviceAccount = new FileInputStream( name: "./flutter-24cbd-firebase-adminsdk-yxeww-7

        FirebaseOptions options = new FirebaseOptions.Builder()
            .setCredentials(GoogleCredentials.fromStream(serviceAccount))
            .build();
            com.google.auth.oauth2
            public class GoogleCredentials
```

Figure 53 – FirebaseInitialization.java

III.2.3 File Service

- `getFileDetails()`

➔ It begins by getting an instance of Firestore from `FirestoreClient`. Firestore is a NoSQL document database provided by Google Cloud Platform.

➔ The `getFileDetails()` method retrieves the details of files stored in a Firestore database collection. It returns a list of File objects obtained from the collection.

```
public List<File> getFileDetails() throws ExecutionException, InterruptedException {
    Firestore dbFirestore= FirestoreClient.getFirestore();
    Iterable<DocumentReference> documentReferences=dbFirestore.collection(COLLECTION_NAME).listDocuments();
    Iterator<DocumentReference> iterator=documentReferences.iterator();
    List<File> fileList=new ArrayList<>();
    File file=null;
    while (iterator.hasNext()){
        DocumentReference documentReference1=iterator.next();
        ApiFuture<DocumentSnapshot> future=documentReference1.get();
        DocumentSnapshot document=future.get();
        file=document.toObject(File.class);
        fileList.add(file);
    }
    return fileList;
}
```

Figure 54 - Method `getFileDetails()`

- `updateFile(File file)`
- update the title or description of document

```
public String updateFile(File file) throws ExecutionException, InterruptedException {
    Firestore dbFirestore=FirestoreClient.getFirestore();
    ApiFuture<WriteResult> collectionApiFuture= (ApiFuture<WriteResult>) dbFirestore.collection(COLLECTION_NAME)
        .document(file.getTitle()).set(file);
    return collectionApiFuture.get().getUpdateTime().toString();
}
```

Figure 55 - Method `updateFile`

- deleteFileByTittle(String tittle)

```

public static String deleteFileByTittle(String tittle) throws ExecutionException, InterruptedException {
    Firestore dbFirestore = FirestoreClient.getFirestore();
    CollectionReference collectionRef = dbFirestore.collection(COLLECTION_NAME);

    Query query = collectionRef.whereEqualTo("tittle", tittle);
    ApiFuture<QuerySnapshot> querySnapshot = query.get();
    List<QueryDocumentSnapshot> documents = querySnapshot.get().getDocuments();

    if (!documents.isEmpty()) {
        // Delete the document
        documents.get(0).getReference().delete();
        return "Document deleted";
    } else {
        // Document with the provided tittle does not exist
        return "Document not found";
    }
}

```

Figure 56 - Method deleteFileByTittle

III.2.4 FileController

the FileController class is annotated with `@RestController`, indicating that it is a controller class responsible for handling HTTP requests and returning responses in a RESTful API.

`@Autowired` indicates the FileService dependency will be automatically injected into the FileController class by the Spring framework, also filecontroller contains methods, it invokes the methods of the fileService instance.

```

@RestController
@RequestMapping("/api")
public class FileController {

    @Autowired
    private FileService fileService;

    @GetMapping("/Files")
    public List<File> getAllFiles() throws ExecutionException, InterruptedException {
        return fileService.getFileDetails();
    }

    @PutMapping("/Files")
    public String update(@RequestBody File file) throws ExecutionException, InterruptedException {
        return fileService.updateFile(file);
    }

    @GetMapping("/Files/{category}")
    public List<File> getProductsByCategory(@PathVariable String category) throws ExecutionException, InterruptedException {
        return fileService.getProductsByCategory(category);
    }

    @DeleteMapping("/Files/{name}")
    public String deleteProduct(@PathVariable String name) throws ExecutionException, InterruptedException {
        return fileService.deleteFileByTittle(name);
    }
}

```

Figure 57 – FileController.java

III.3 Firebase

III.3.1 General Settings

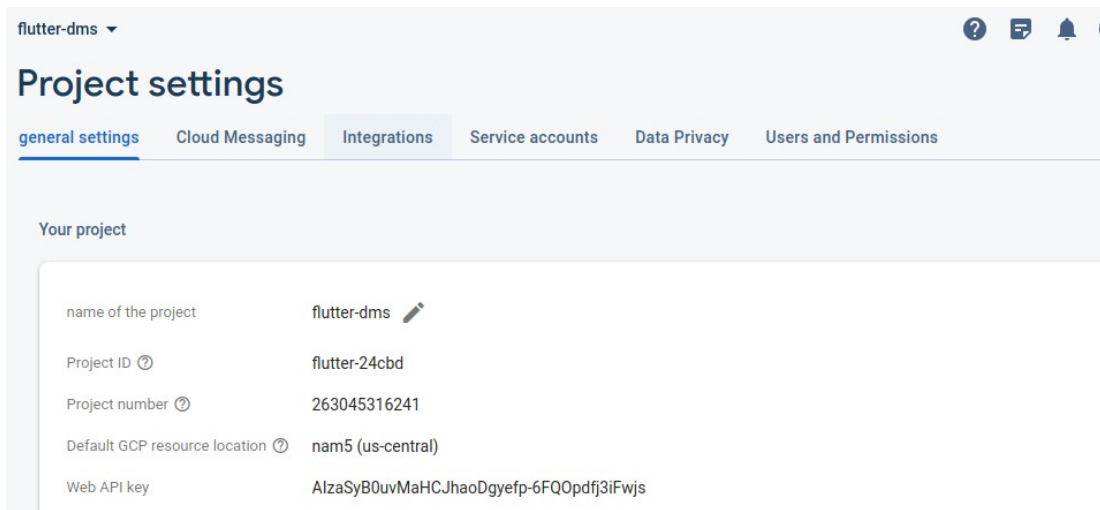


Figure 58 - All details database flutter-dms

III.3.2 Firestore Database

III.3.2.1 Collection “Files”

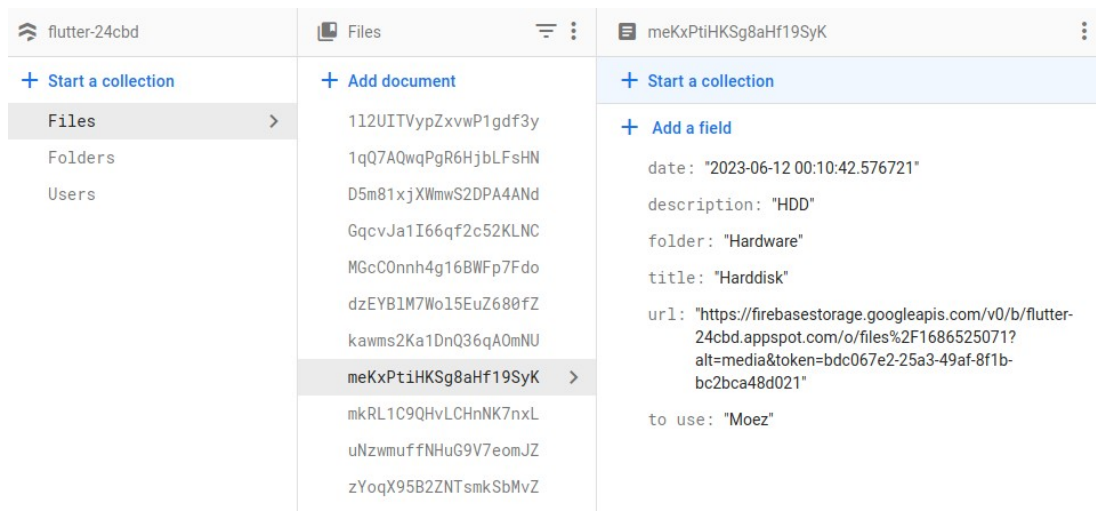


Figure 59 - Collection Files

III.3.2.2 Collection “Folders”

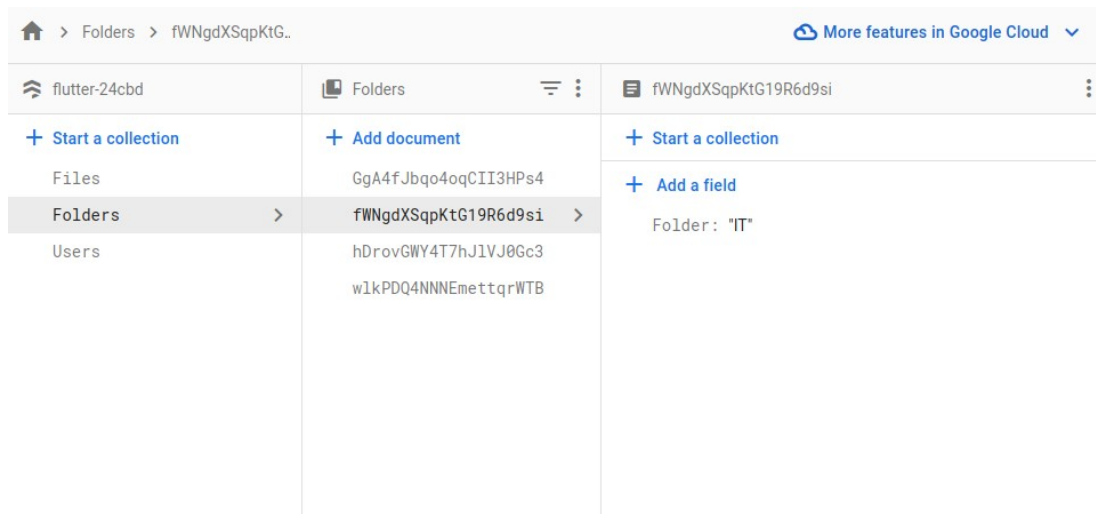


Figure 60 - Collection Folder

III.3.2.3 Collection “Users”

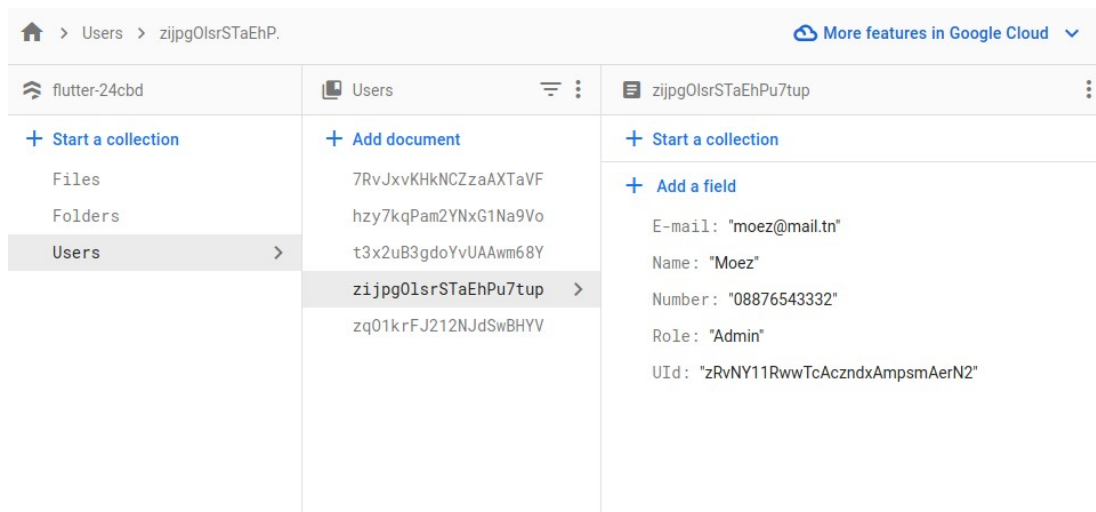


Figure 61 - Collection Users

III.4 Hosting application on railway.app

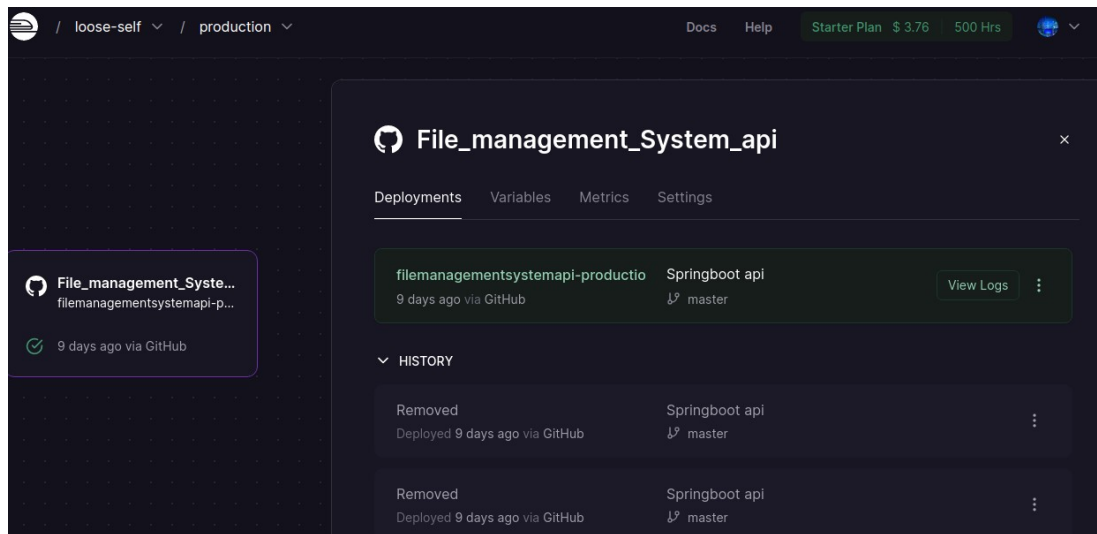


Figure 62 - hosting app in railway.app

III.5 Extracting the application API from railway.app

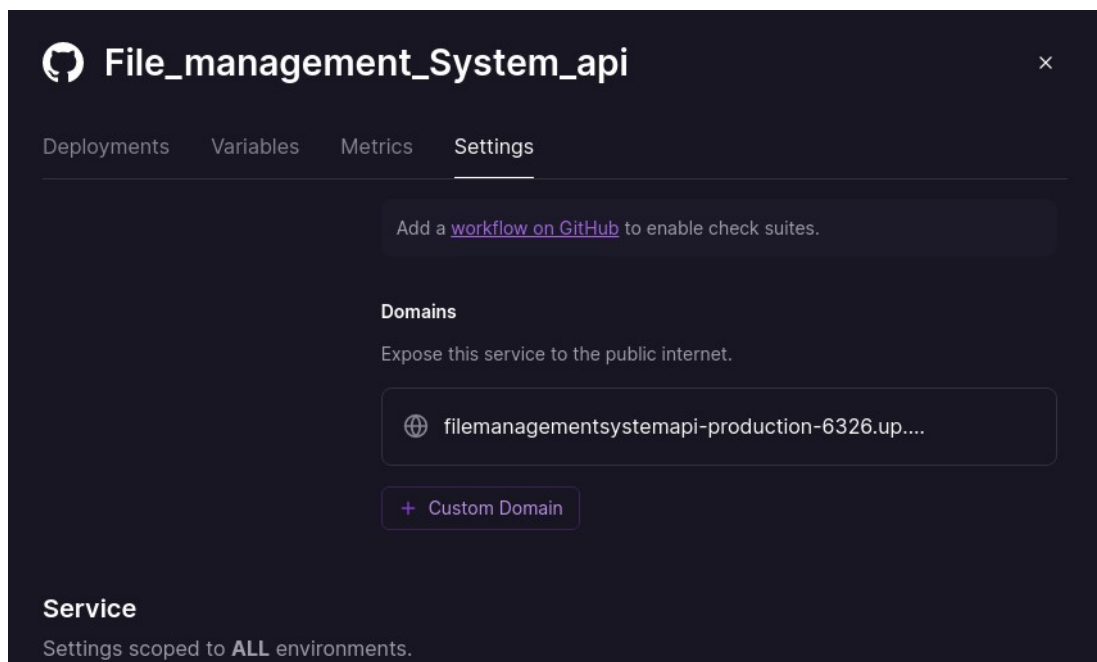


Figure 63 - API application file management

- API application document management system:

➔ <https://filemanagementsystemapi-production-6326.up.railway.app/api/Files>

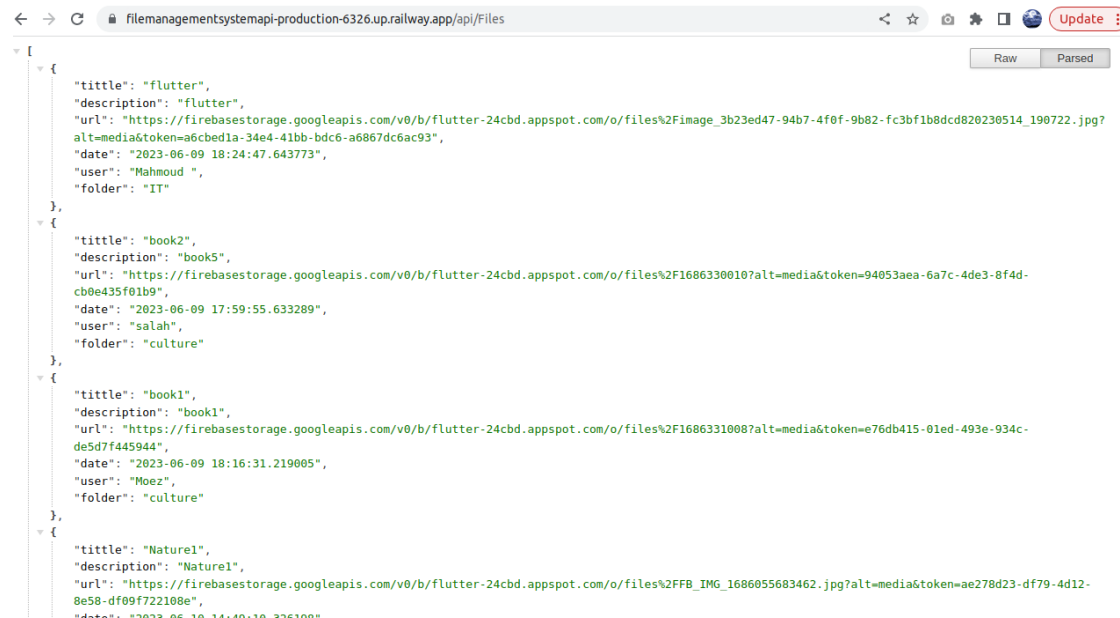


Figure 64 - All data API application file management

IV. CONCLUSION

During this chapter, we have presented the different development tools as well as screen previews testifying to the different facets of our project. In the phase of realization, we have developed the elements constituting our application while respecting both the needs that the design has elaborated. We now have only to conclude and summarize the contributions of this project as well as the possible perspectives.

General Conclusion And Perspectives

The objective of this project was the realization of an application mobile electronic document management.

To do this, we were first invited to present the general framework of our project by presenting the host organization and the problematic. Then we

we have exposed the study of the existing and the specification of the needs. The next step is analysis and design. And finally, we exposed the work done at through screenshots.

The realization of my internship allowed me to acquire three main contributions, these are the most substantial, for me, since they revealed themselves, during the writing of my report.

The first contribution, on a personal level, of this internship will undoubtedly have been the duty to achieve objectives in a limited time. This contribution is substantial for us since it conditions the fact, in the future, of being efficient in our profession. Knowing how to manage our time in order to fulfill goals set in advance is crucial in the professional world.

The second contribution, obtained by the realization of this internship, is teamwork. The feeling of belonging relative to this team was exacerbated by a relationship in the normality of things. This contribution is important for us insofar as teamwork makes it possible to become aware of the interdependence of the actors composing it for the more or less rapid progression of the personal work produced.

The third major contribution provided by this internship is the discovery of the professional environment. With the discovery of professionals, in their environment, in their daily work, in their way of approaching a problem and solving it.

in addition perspective refers to a software solution or platform designed to organize, store, and manage electronic documents and files from various perspectives or viewpoints. It aims to provide users with multiple ways to access, navigate, and interact with documents, allowing them to gain different insights and understandings based on their specific needs or roles.

Here are some key features and benefits of a perspective document management system:

- ✓ **Cloud-Based DMS:** Consider using a cloud-based DMS solution like Google Drive, Microsoft OneDrive, or Dropbox. These platforms offer robust document management features, including version control, collaboration, and secure access from anywhere.
- ✓ **Metadata and Tagging:** Implement a consistent metadata and tagging system to enhance document searchability and categorization. This will make it easier for users to find relevant documents based on specific criteria.
- ✓ **Integration and compatibility:** Integration with other systems, such as email clients, project management tools, or customer relationship management (CRM) platforms, can enhance the efficiency and effectiveness of document management processes.

Webography

[1].<https://znetit.com/>

[2].<https://planningtank.com/computer-applications/electronic-document-management-system-edms>

[3].<https://dart.dev/>

[4].<https://docs.flutter.dev/resources/architectural-overview>

[5].https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm

[6].<https://www.redhat.com/en/topics/api/what-is-a-rest-api>

[7].<https://www.databasejournal.com/features/introduction-to-firebase/>

[8].<https://www.jetbrains.com/idea/features/>

[9].<https://www.android.com/what-is-android/>

[10].<https://code.visualstudio.com/docs>

[11].<https://git-scm.com/about>

[12].<https://docs.github.com/en/desktop/installing-and-configuring-github-desktop/overview/creating-your-first-repository-using-github-desktop>

[13].<https://www.drawio.com/about>

[14].<https://docs.railway.app/getting-started>

[15].<https://www.geeksforgeeks.org/benefit-of-using-mvc/>