Software Architecture Specification

version 1.0

by Ievgen Pervushyn

Revision History

Date	Version	Description	Author
June 22, 2020	1.0	Initial Software Architecture Specification	I. Pervushyn

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This Software Architecture Specification provides an architectural overview of the Storage Client Webapp with Redundant Deals: A web application and storage deal agent that supports users storing data redundantly with multiple miners. The app will track active storage deals and allow users to perform data retrieval.

2. Architectural Representation

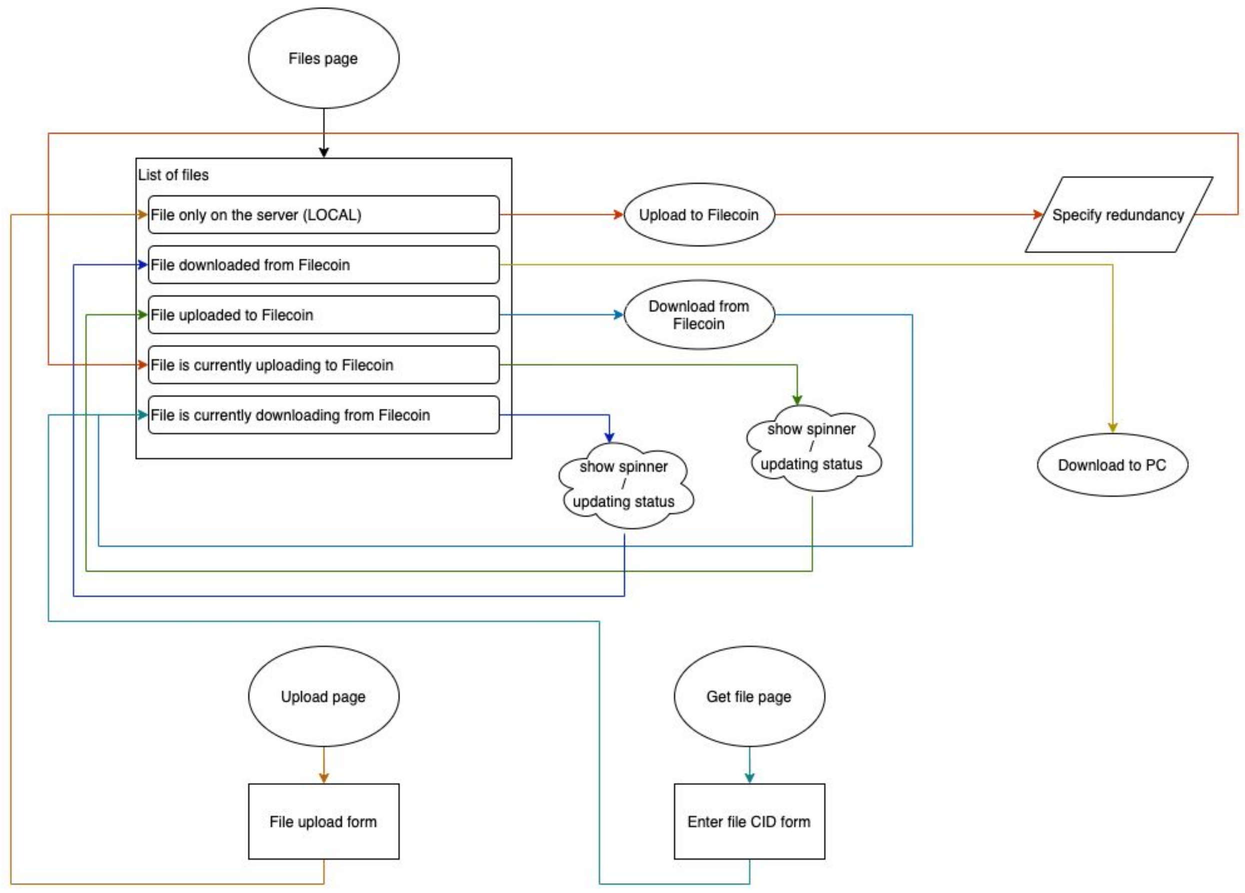
This document presents the architecture as a series of views; use case view, logical view, process view and deployment view. There is no separate implementation view described in this document. These are views on an underlying Unified Modeling Language (UML) model developed using diagrams.net.

3. Use-Case View

A description of the use-case view of the software architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.
The Storage Client Webapp with Redundant Deals use cases are:

- Display FIL Balance and Main Address
- Upload file
- Store redundant copies of the file with multiple miners
- Retrieve a file from one of the miners it was stored to
- Download file to the user’s computer
- Track the status of storing/retrieving process

3.1 Architecturally-Significant Use Cases


Files page
List of files
File only on the server (LOCAL)
File downloaded from Filecoin
File uploaded to Filecoin
File is currently uploading to Filecoin
File is currently downloading from Filecoin
Upload to Filecoin
Specify redundancy
Download from Filecoin
show spinner / updating status
show spinner / updating status
Download to PC
Upload page
File upload form
Get file page
Enter file CID form

Upload to Filecoin:

Brief Description: The process of initiating deals with multiple Filecoin storage miners. The number of redundant deals is specified by the user. Pre-filled value is 3. Also, it display how much would it cost to store.

Download from Filecoin:

The process of retrieving a file from the Filecoin storage miner. There are two possible way to initiate this process: 1. By clicking on Download button near previously-stored file; 2. From the *Get File Page* by entering the CID of the file you need to retrieve.

Download to PC:

The process allows user to download a retrieved file to their computer.

File upload form:

The form allows user to select a file from their computer to prepare it to be stored to the Filecoin. As soon as the file uploaded it will be shown on the Files list as Uploaded.
Get File Page:
The page contains a field to enter CID of the file you want to retrieve from the Filecoin.

4. Logical View

A description of the logical view of the architecture. Describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

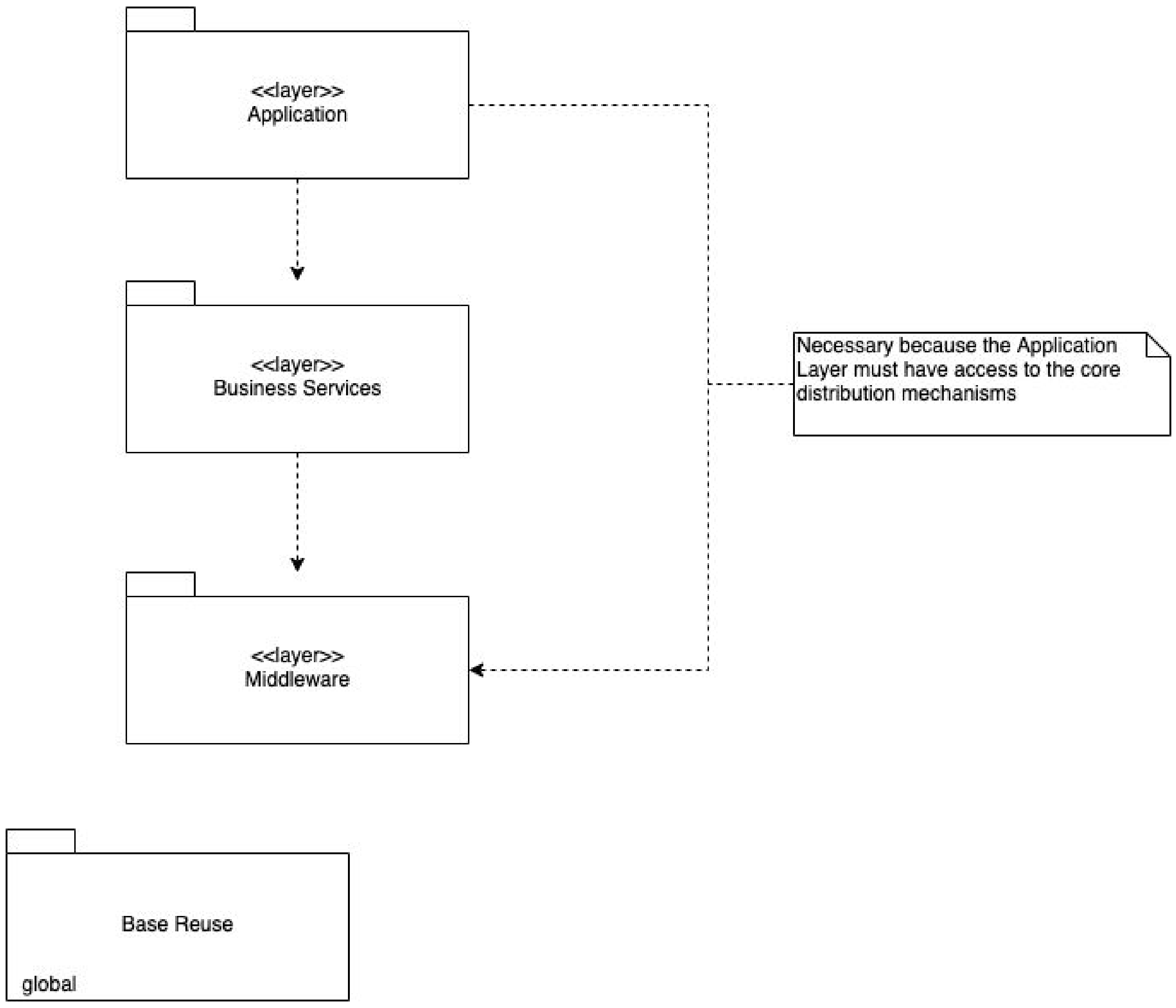
The logical view of the application is comprised of the 3 main packages: User Interface, Business Services, and Business Objects.

The User Interface Package contains classes for each of the forms that the actors use to communicate with the System. Boundary classes exist to support lists, forms, processes.

The Business Services Package contains control classes for interfacing with the Filecoin, controlling files store and retrieval processes.

The Business Objects Package includes entity classes for the file artifacts (i.e. uploaded file, stored file, retrieved file, etc) and boundary classes for the interface with the Filecoin.

4.1 Architecture Overview – Package and Subsystem Layering


<<layer>>
Application
<<layer>>
Business Services
<<layer>>
Middleware
Necessary because the Application Layer must have access to the core distribution mechanisms
Base Reuse
global

4.1.1 Application layer

This application layer has all the boundary classes that represent the application screens that the user sees. This layer depends upon the Process Objects layer; that straddles the separation of the client from mid-tier.

4.1.2 Business Services layer

The Business Services process layer has all the controller classes that represent the use case managers that drive the application behavior. This layer represents the client-to-mid-tier border. The Business Services layer depends upon the Process Objects layer; that straddles the separation of the client from mid-tier.

4.1.3 Middleware layer

The Middleware layer supports access to Relational DBMS and Filecoin.

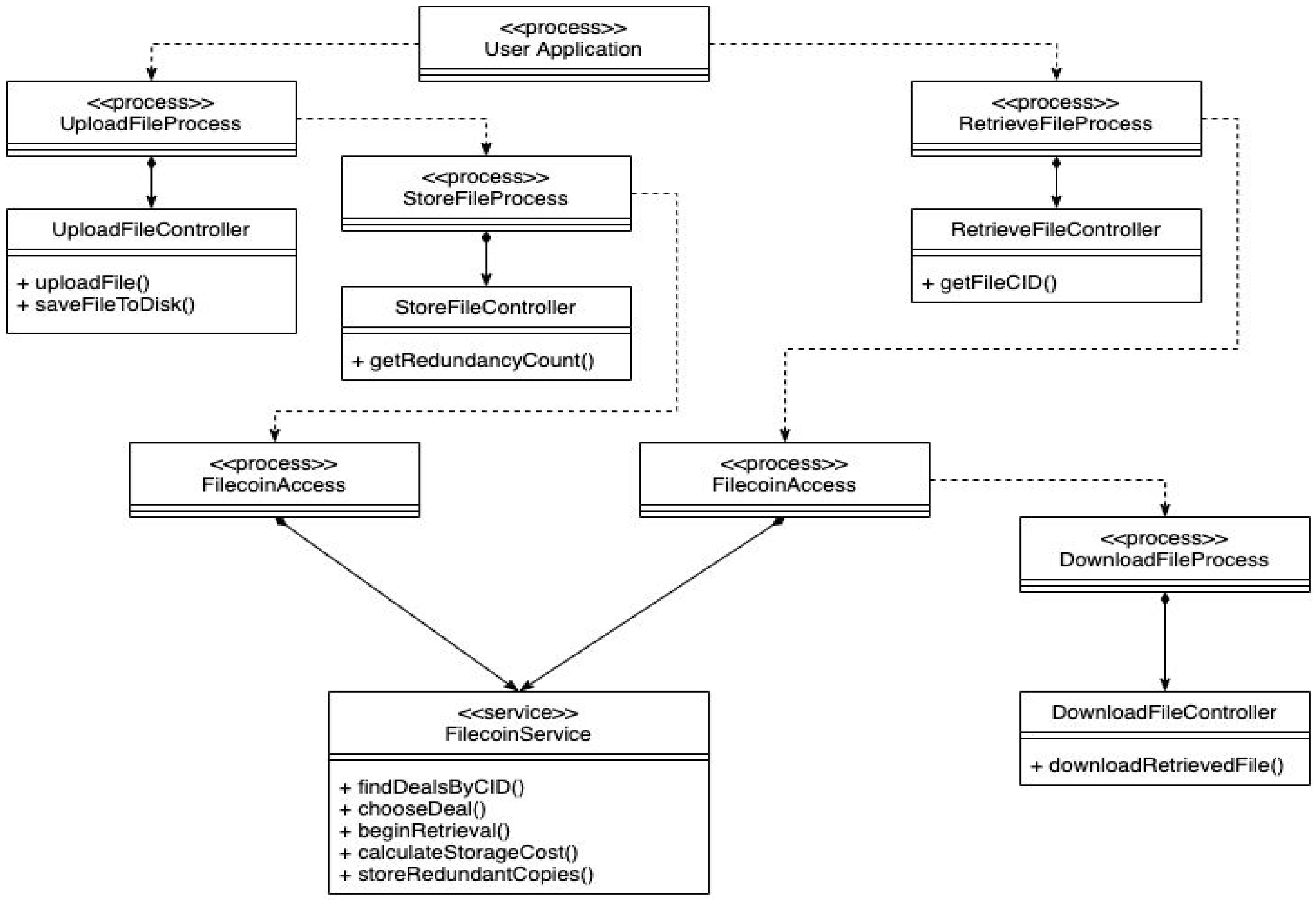
4.1.4 Base Reuse

The Base Reuse package includes classes to support list functions and patterns.

5. Process View

5.1 Processes

A description of the process view of the architecture. Describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.
The Process Model illustrates the data store and retrieval classes organized as executable processes. Processes exist to support file uploading, storage, retrieval and downloading and also to display current FIL balance.



<<process>> User Application
<<process>> UploadFileProcess
<<process>> RetrieveFileProcess
<<process>> StoreFileProcess
UploadFileController
+ uploadFile()
+ saveFileToDisk()
StoreFileController
+ getRedundancyCount()
RetrieveFileController
+ getFileCID()
<<process>> FilecoinAccess
<<process>> FilecoinAccess
<<process>> DownloadFileProcess
DownloadFileController
+ downloadRetrievedFile()
<<service>> FilecoinService
+ findDealsByCID()
+ chooseDeal()
+ beginRetrieval()
+ calculateStorageCost()
+ storeRedundantCopies()

5.1.1 User Application

Manages user functionality, including user interface processing and coordination with the business processes.

5.1.2 UploadFileProcess

This process is about to upload a file from user's computer to the server. It will be stored on the server's disk space. Once it uploaded it can be used to process file storage.

5.1.3 StoreFileProcess

This one will ask the user about number of redundant copies they want to create, display the total amount of FIL needed to initiate all the deals and gives control to the FilecoinService

5.1.4 FilecoinAccess (storing)

This is for calculating the cost of the deals and initiating the storing process with multiple storage miners.

5.1.5 RetrieveFileProcess

Allows a user to initiate a file retrieval process. It asks for a file CID and then contacts the Filecoin service though FilecoinAccess (5.1.6). This process also can be initiated directly after the StoreFileProcess (5.1.3) as soon as we know the file's CID and it's stored.

5.1.6 FilecoinAccess (retrieval)

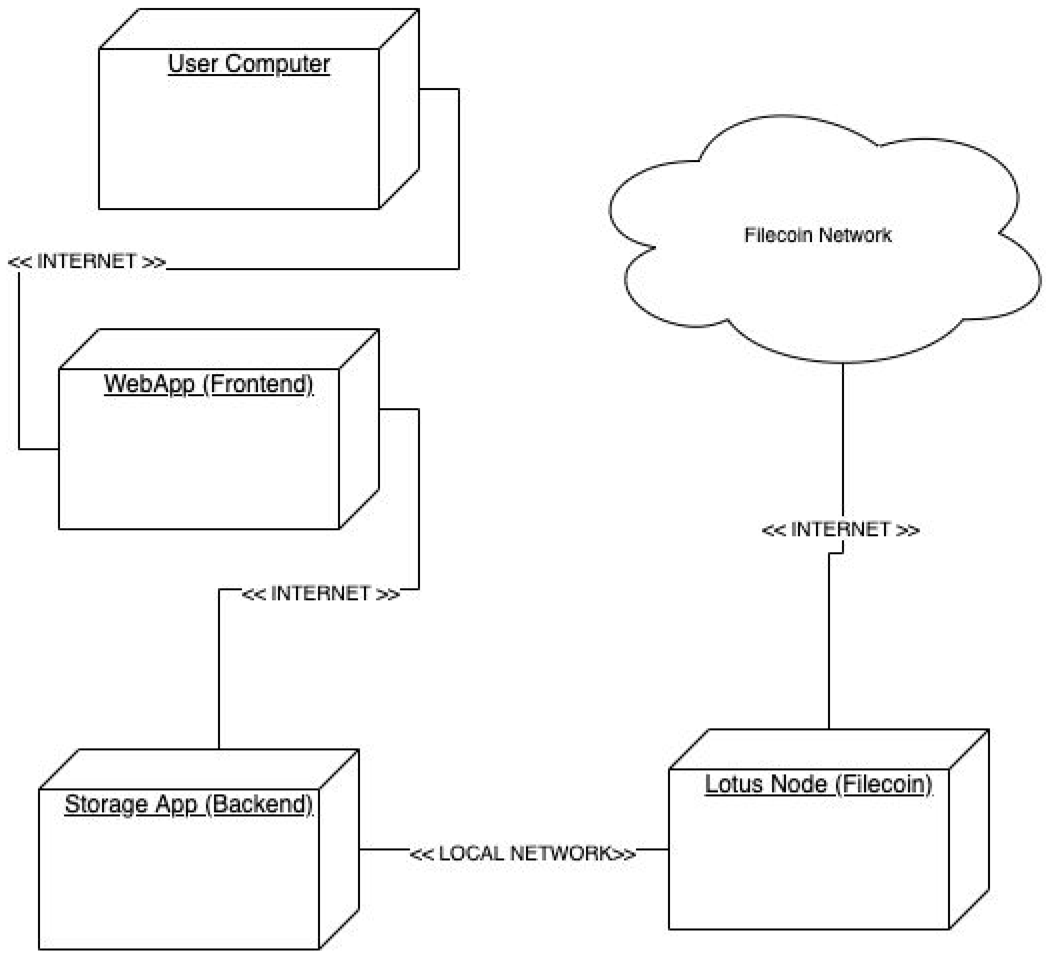
Connecting to Filecoin network to find a deals for a CID specified. Then it will choose the storage node and begin the retrieval process.

5.1.7 DownloadFileProcess

Once the file is retrieved it could be downloaded from the server to the user's computer.

6. Deployment

6.1 Deployment View



6.1.1 User Computer

Users access the application via their computer (laptop, pc, smartphone, tablet, etc.)

6.1.2 WebApp (Frontend)

The special frontend server (ReactJS) intended to display web user interface. It allows user to see all needed information and initiate required processes

6.1.3 Storage App (Backend)

Backend server (Go) used as a main computation unit. It receives commands from the user, does calculations and sends commands to the Lotus Node

6.1.4 Lotus Node (Filecoin)

This type of server is the filecoin lotus node set up. It communicates with the Filecoin Network.

6.2 Orchestration

All deployments (6.2 – 6.4) could be deployed using Docker containers or orchestrated via Kubernetes.