

# Document and Task Management

Software Architect

Author  
**Ian Torres**

Software Engineer

# What we know?

The system shall:

- Allow a user to add documents
- Version documents
- Allow a user add tasks to a document
- Allow a user to download a document

# What we need to considerate

- UI currently with design
- Product started stories but **needs to ensure technical solution before continuing**
- Solution needs to consider how integrations with client document management will be affected / handled
- This roadmap item is critical to new client launch, which is expected to be completed by october.

# Let's create a solution!

1. We need to define a **Solution** including **Architect Documentation**.
2. The solution must support Customer integrations.
3. The solution must be a **MVP**.

# About the **Backend/Frontend** Premises

## We have a existing code base

This solution will be integrated with existing project framework and libraries.

## We are using Laravel and Vue

This solution can use framework components and patterns.

## We have a MySQL and Redis

We have persistent and non-persistent storage.

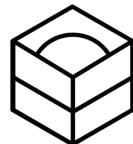
## We are using the Cloud and Continuous Integration/Deployment

We can consume AWS and the solution must consider testing and deployment instructions.

# About the **Solution**

# About the **Solution** Definitions

## Models



Are classes designed to work with database information

## Observers



Are classes designed to react on models events dispatching

## Policies



Are classes designed to authorize user actions

## Storages

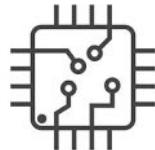


Are classes designed to implement customer storage adapters

# About the **Solution**

## More Definitions

### Controllers



Are classes designed to interact with models and returns HTTP responses

### Components



Are classes or definitions which doesn't have any dependencies.

### Requests



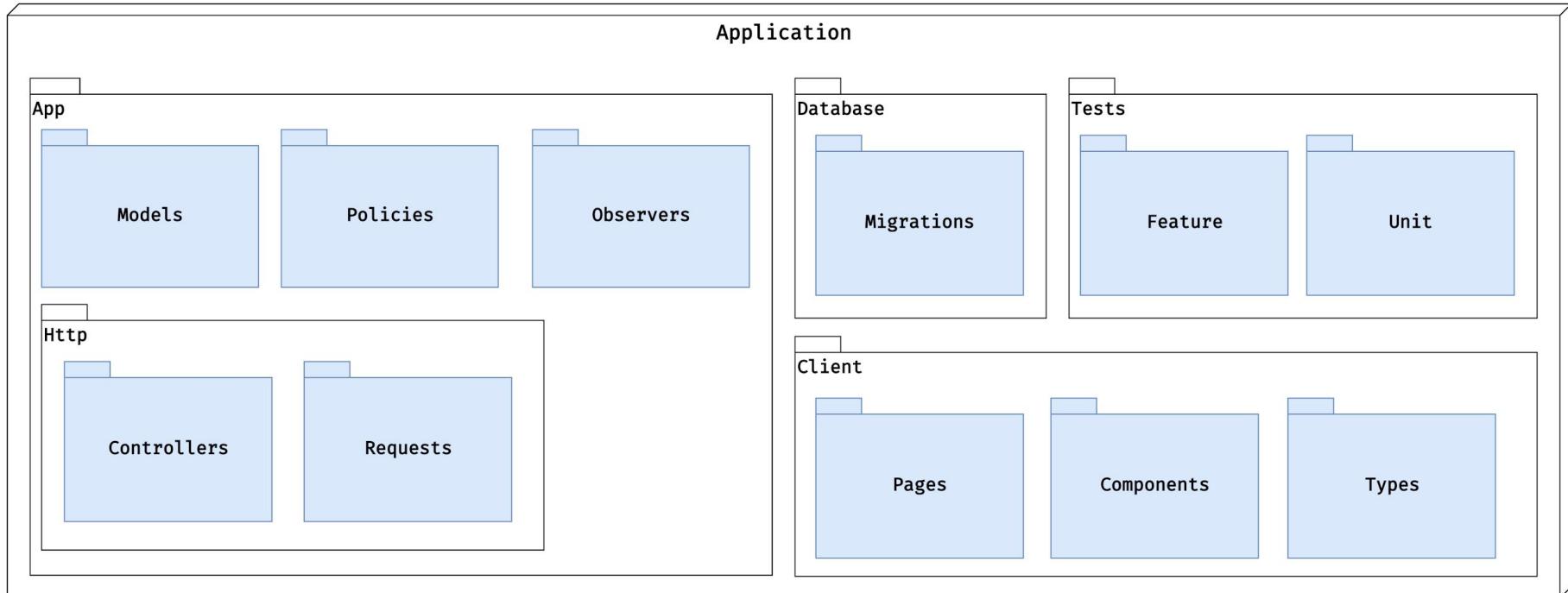
Are classes designed to authorize and validates HTTP requests and payloads

### Types

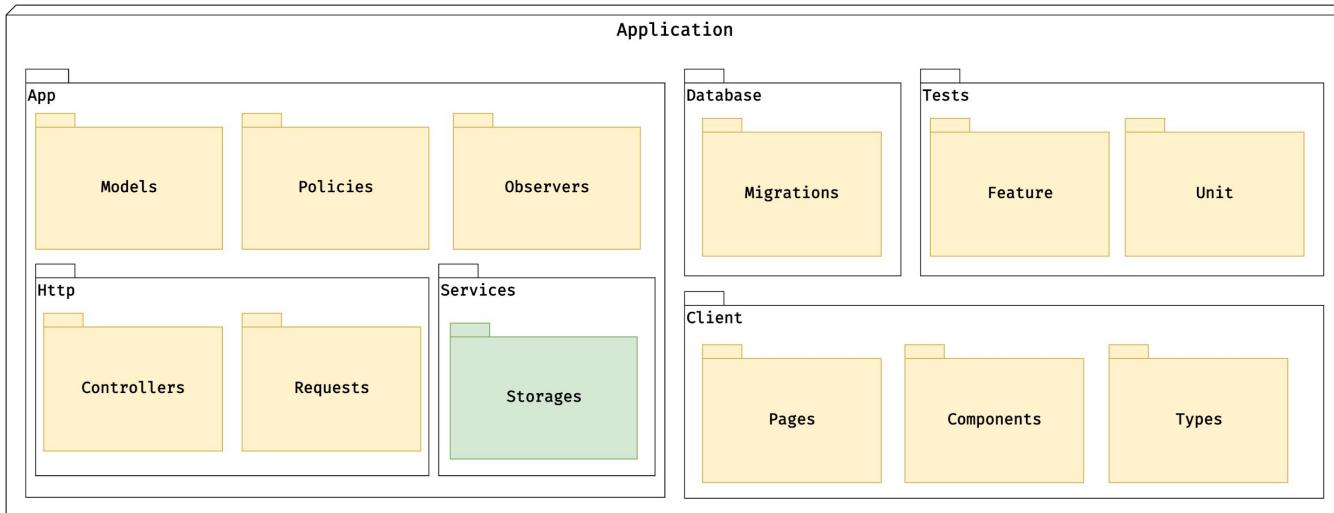


Are TypeScript models designed to store application instances states.

# Existing Project Structure



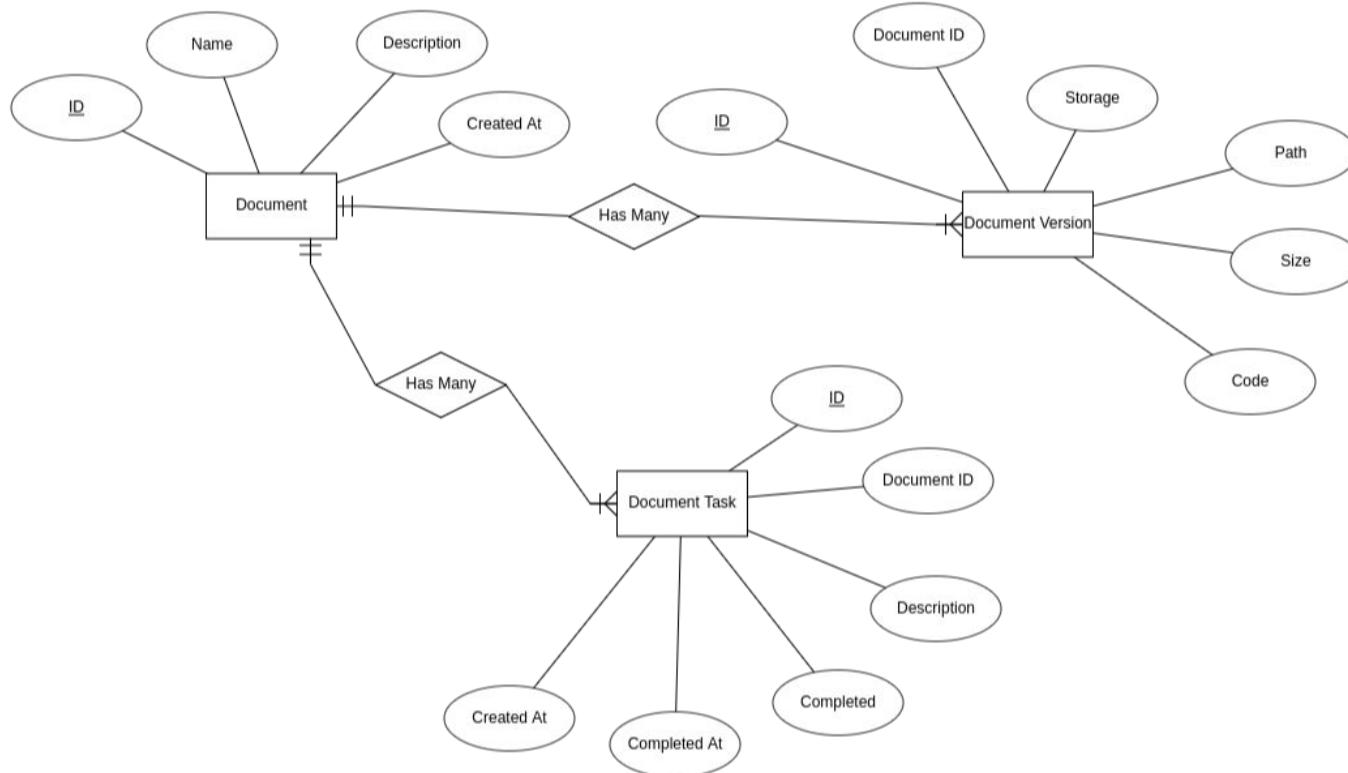
# Proposed Project Structure

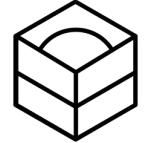


## Note

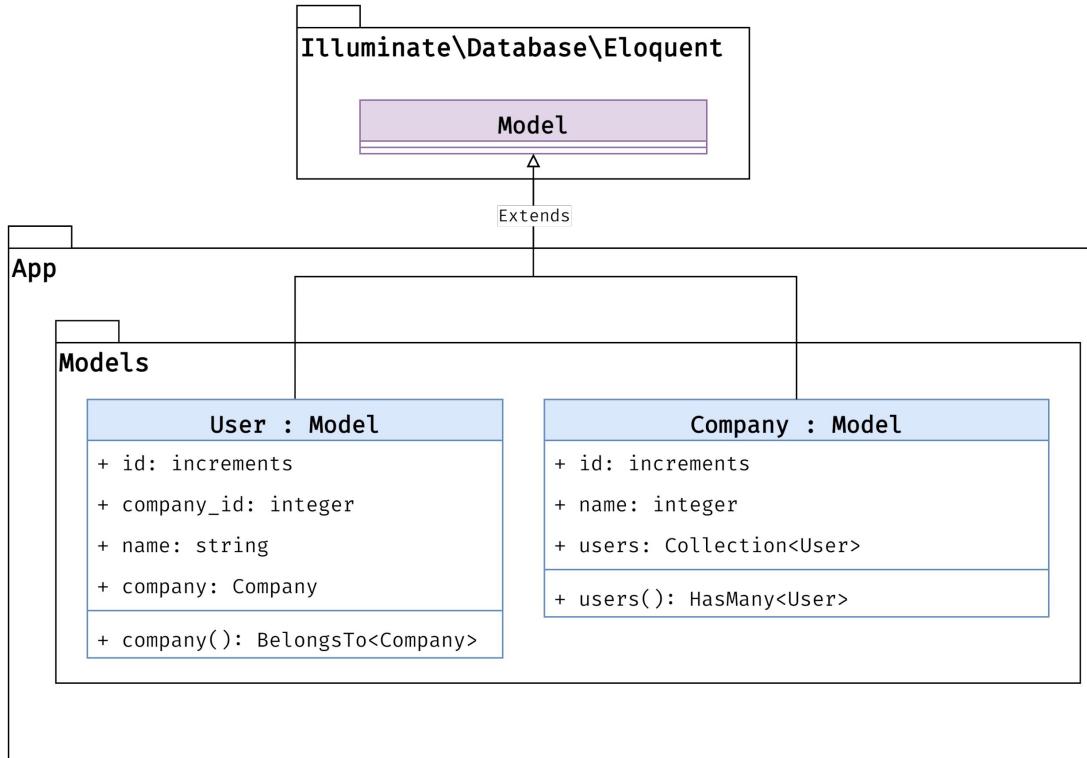
Yellow are touched packages and Green are new packages.

# Expected Entities

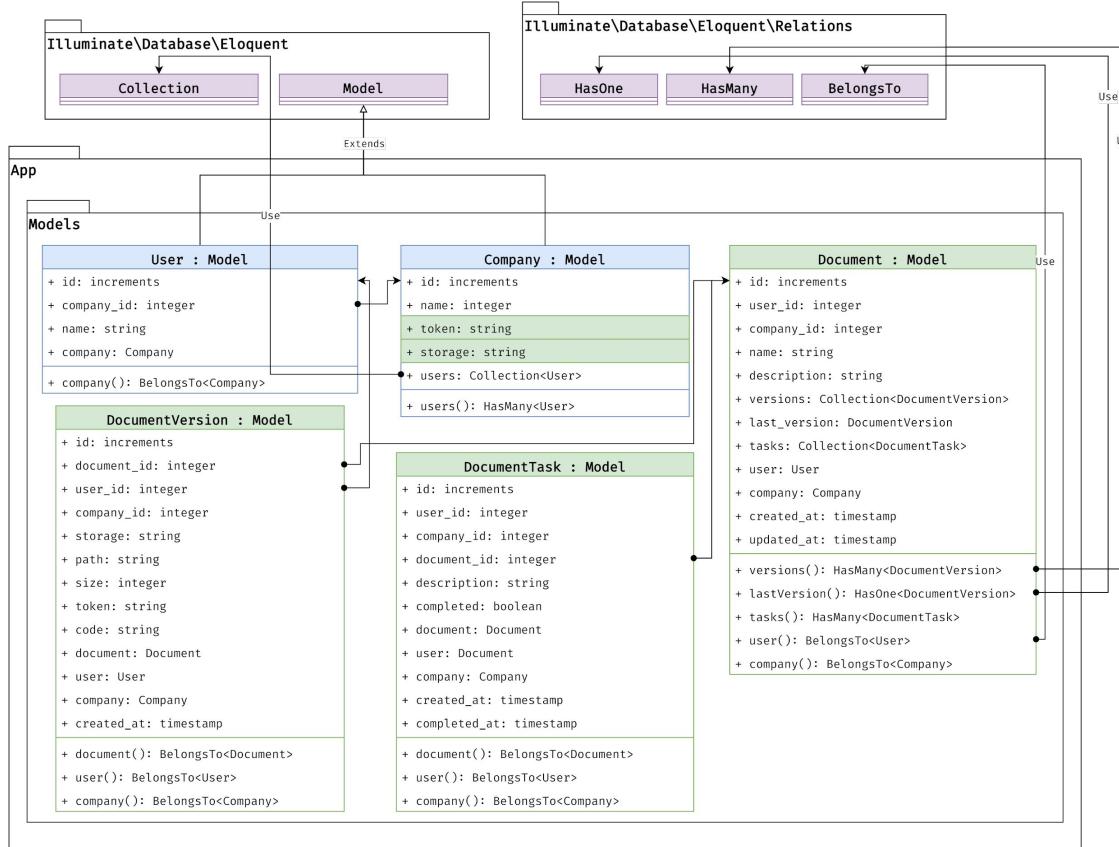
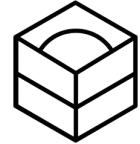




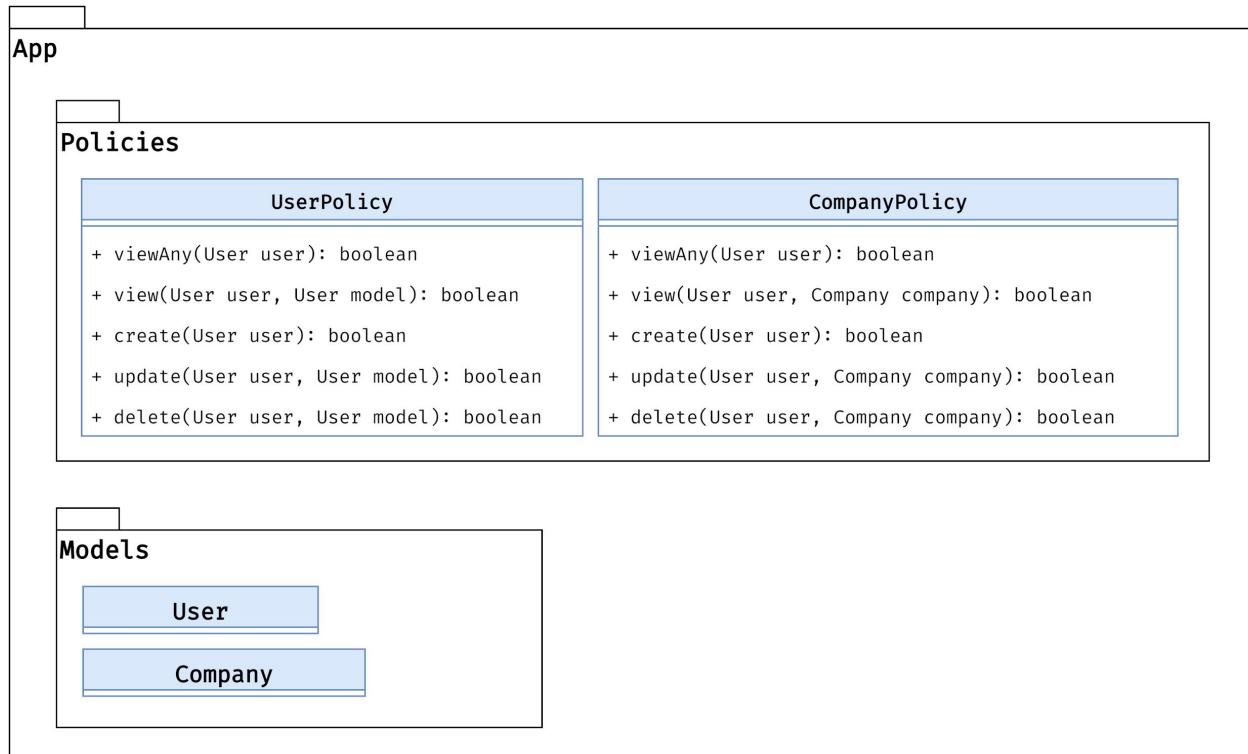
# Expected Models



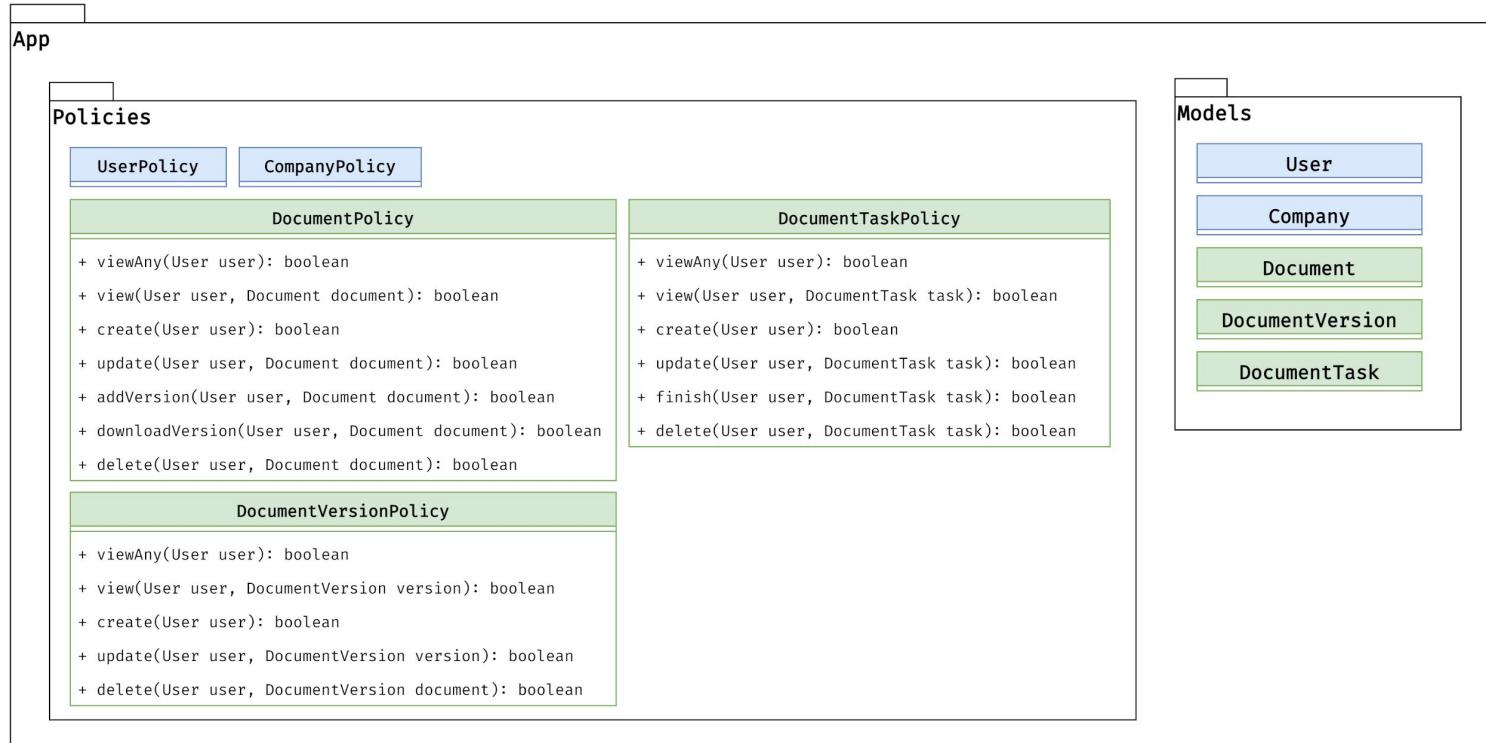
# Proposed Models



# Expected Policies



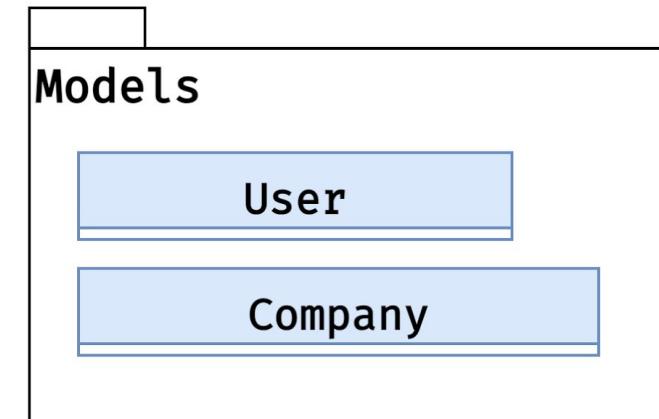
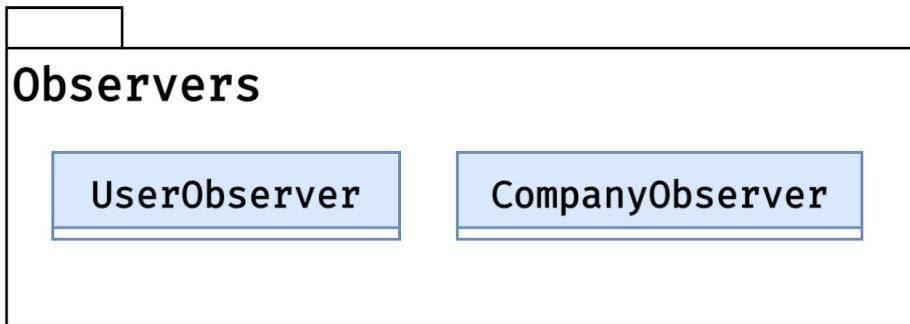
# Proposed Policies



# Expected Observers



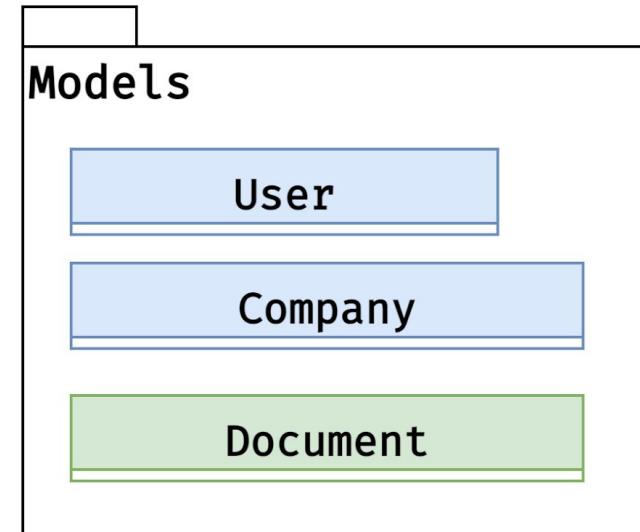
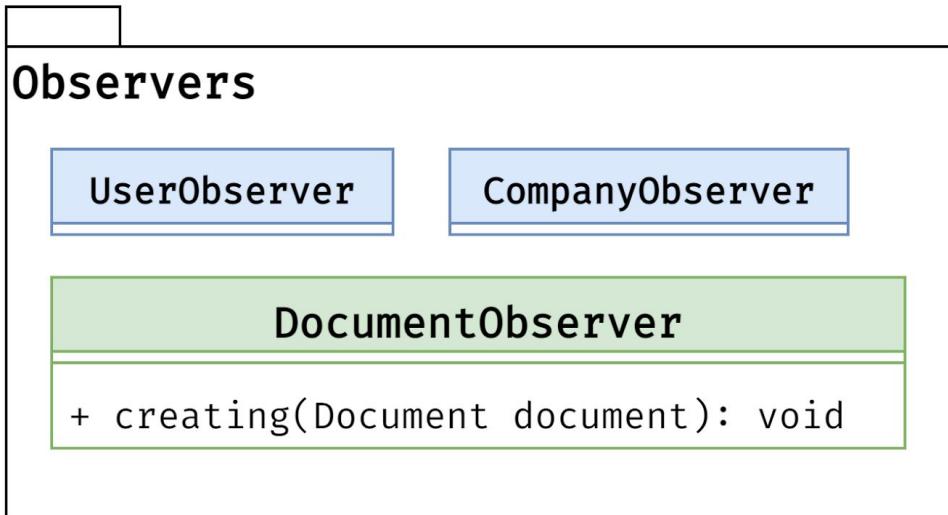
App



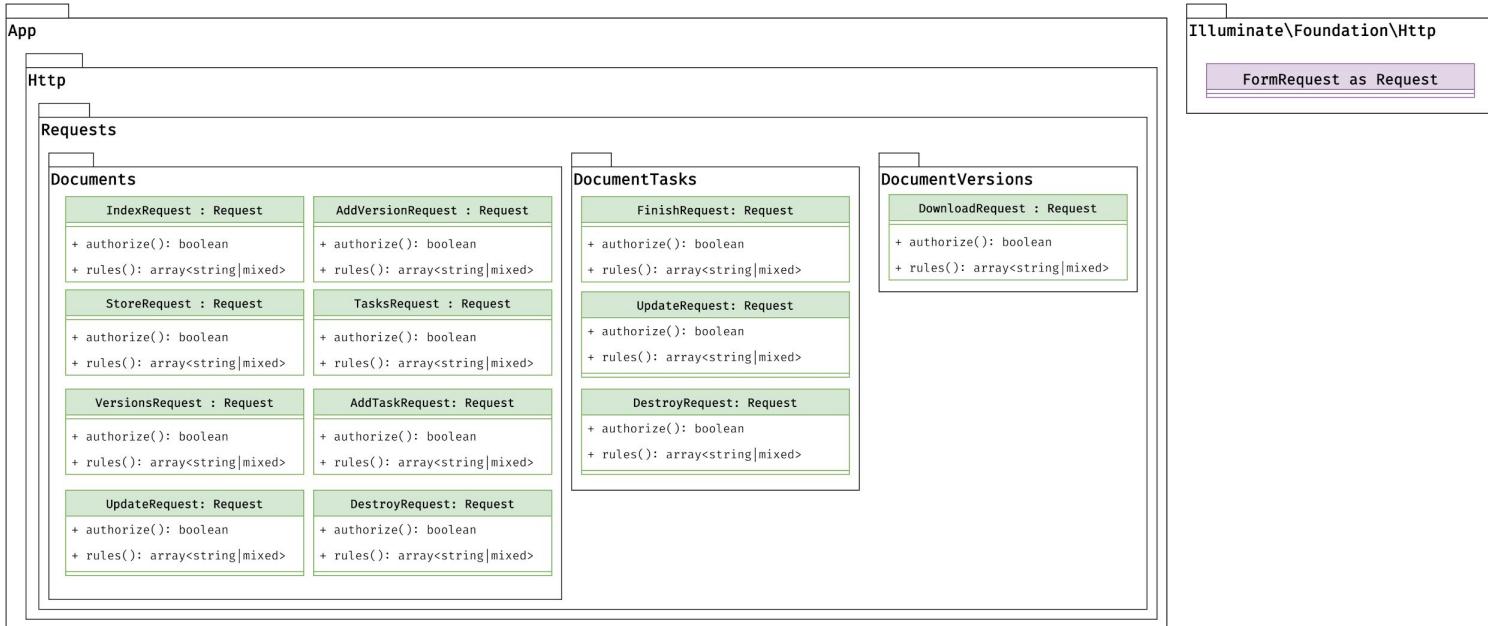
# Proposed Observers



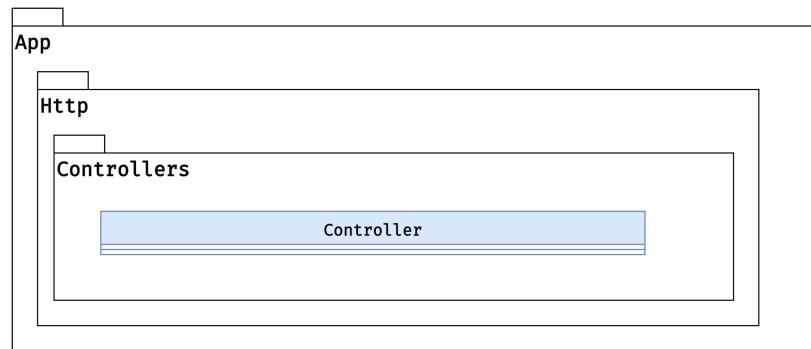
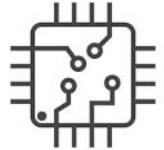
App

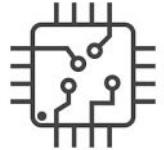


# Proposed Requests

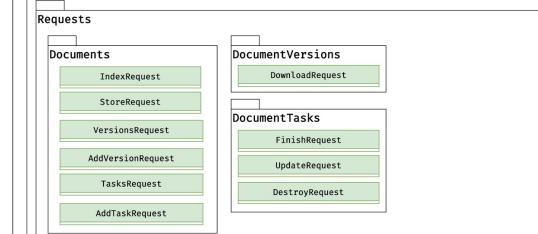
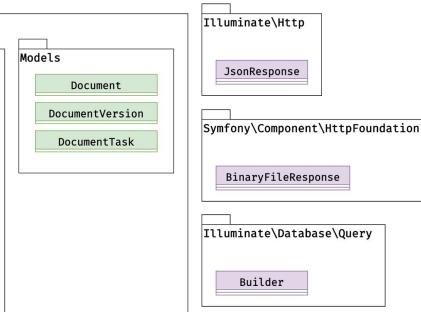
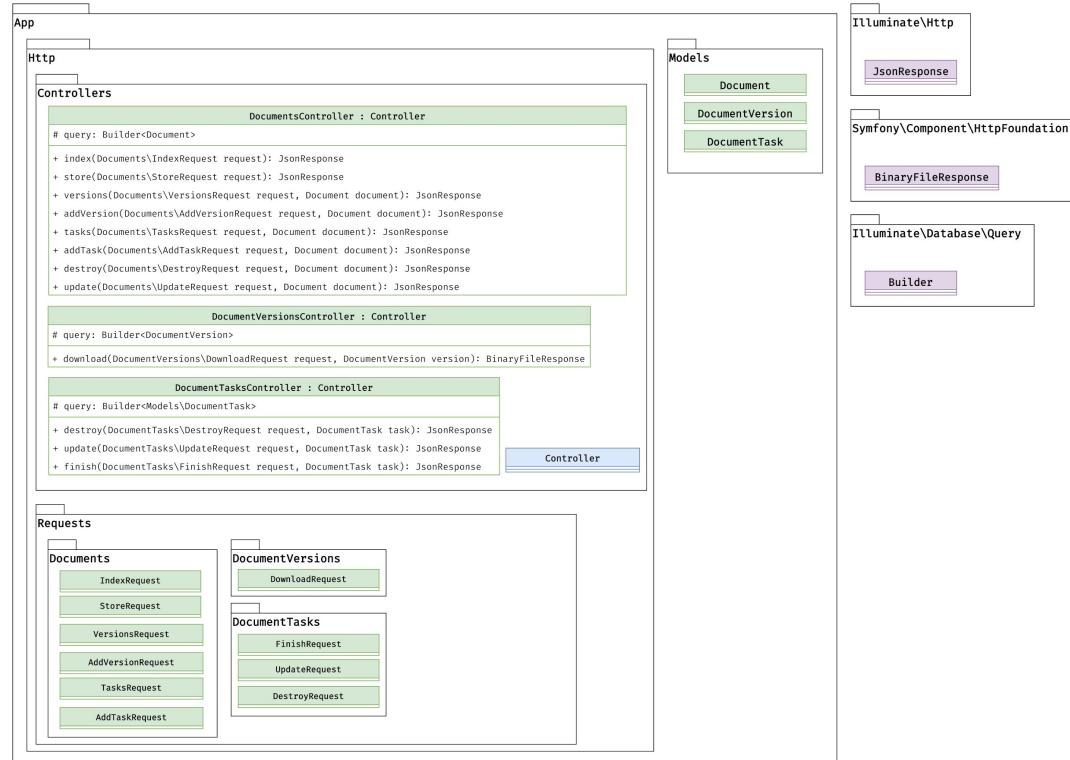


# Expected Controllers



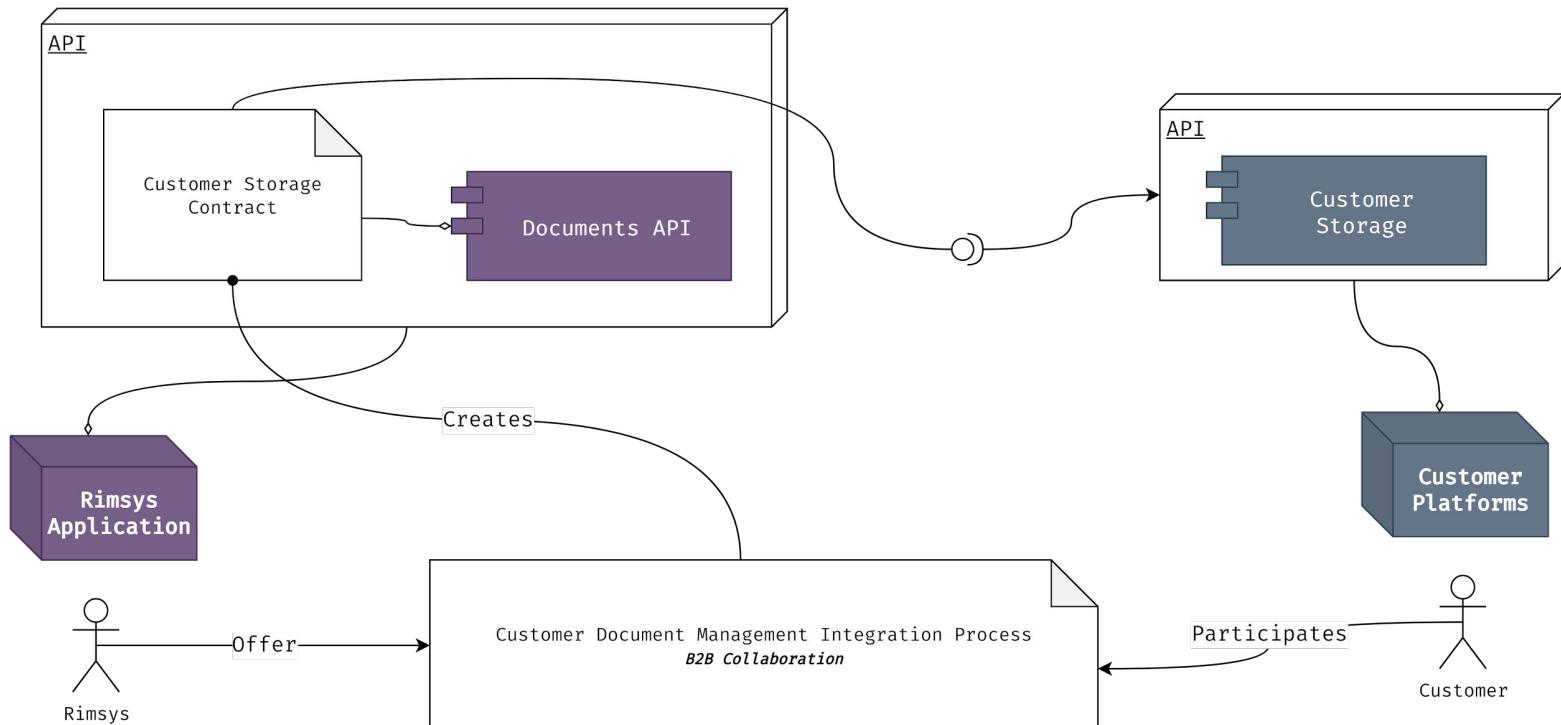


# Proposed Controllers

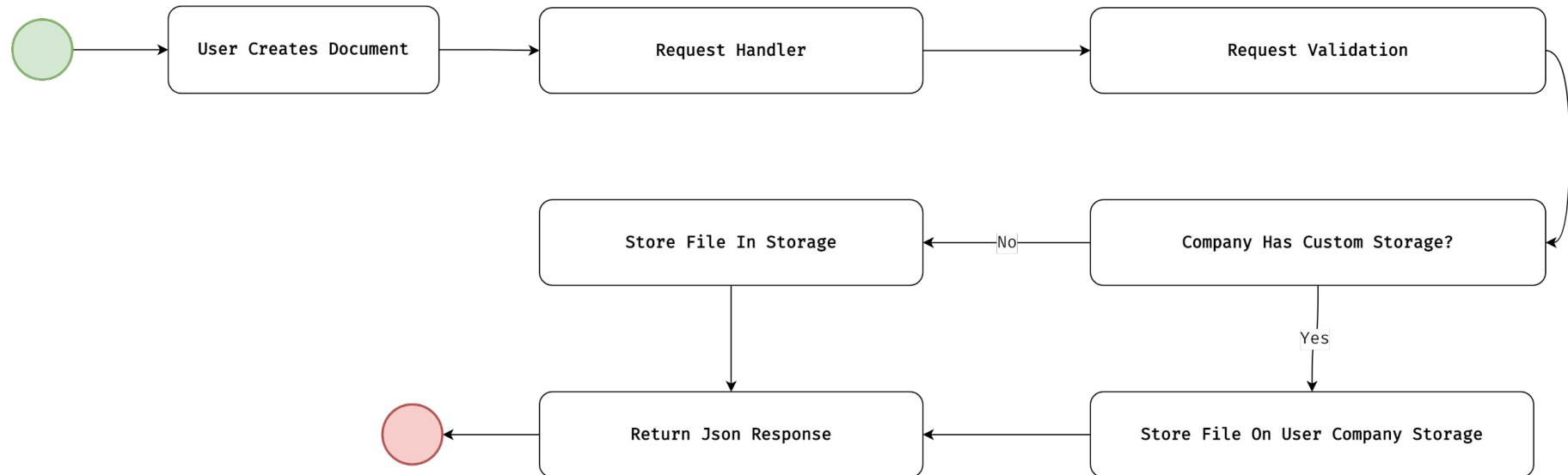


# About the Storage

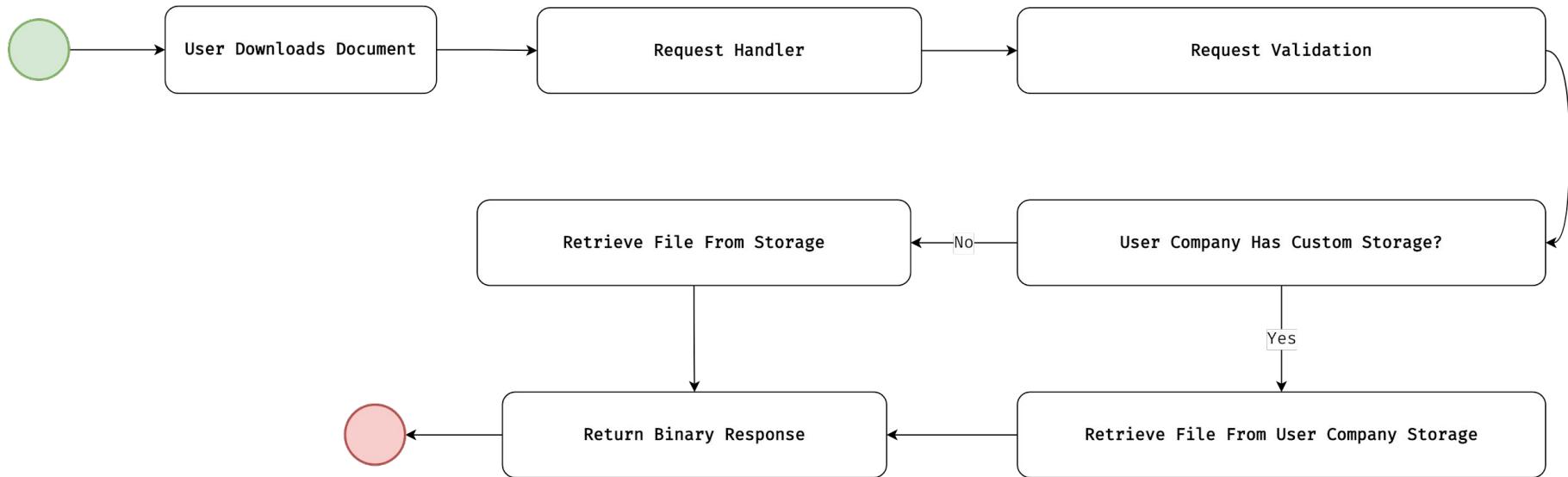
# Overview



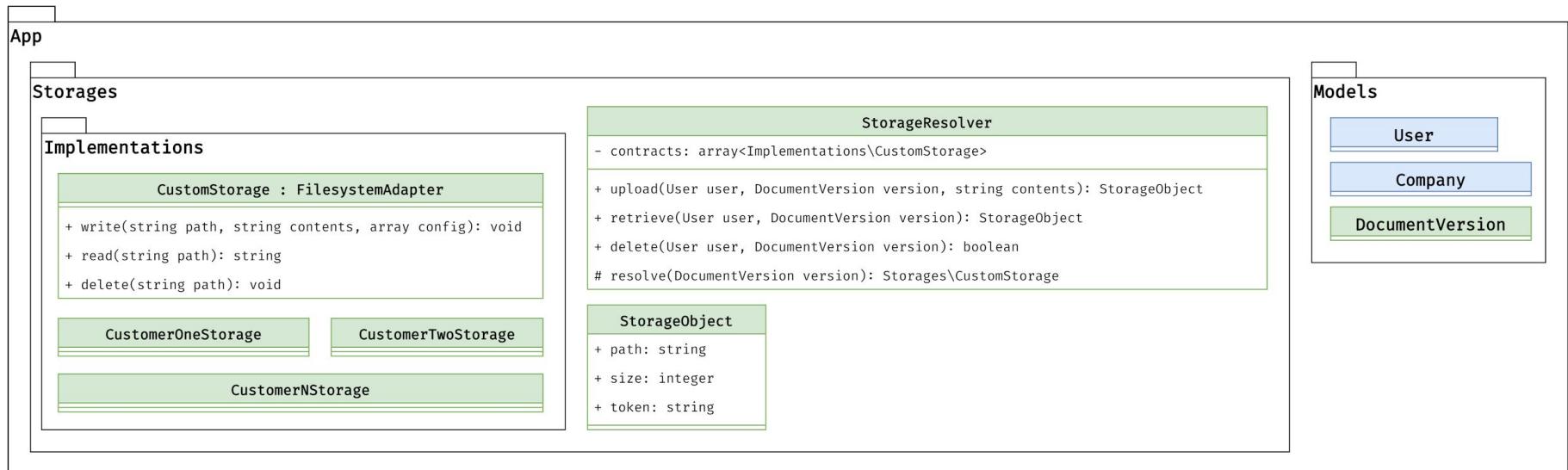
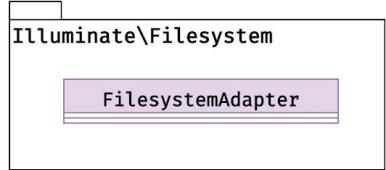
# How Works?



# What about download?



# Proposed Solution



# About the **Frontend**

# About the Frontend

## Proposed UI : Documents

### Documents List

Application Navbar

## Documents

Add Document

Manage your documents and tasks

Records				
Name	Version	Tasks	Actions	
Project Plan Updated 3 hours ago	John Doe 1	7	4/5	<button>View</button> <button>Edit</button> <button>Remove</button>
Project Charter Updated 2 days ago	2	10/10	10/10	<button>View</button> <button>Edit</button> <button>Remove</button>
Business Case Updated 1 day ago	3	30/30	30/30	<button>View</button> <button>Edit</button> <button>Remove</button>
Project Communication Plan Updated 6 hours ago	4	15/20	15/20	<button>View</button> <button>Edit</button> <button>Remove</button>
Scope Statement Updated 6 hours ago	5	3/5	3/5	<button>View</button> <button>Edit</button> <button>Remove</button>

Previous

5 of 10 documents

Next

# About the Frontend

## Proposed UI : Add Documents

Add Documents

Application Navbar

### Documents

Manage your documents and tasks

Project P Updated 3
Project C Updated 2
Business C Updated 1
Project C Updated 6
Scope Stat Updated 6

Add new Document

Name

Description

Select a file in your local machine

Cancel  Store

Previous  5 of 10 documents

Add Document

Remove  Remove  Remove  Remove

# About the Frontend

## Proposed UI : Versions

### Document Versions

Application Navbar

#### Project Plan

Created by  John Doe at 10 Days Ago  
Updated At 3 hours ago

[Back](#) [Add Version](#)

[Versions](#) [Tasks](#)

Records		
Version	Created At	Actions
7	 3 hours ago	<a href="#">Download</a>
6	 2 days ago	<a href="#">Download</a>
5	 4 days ago	<a href="#">Download</a>
4	 5 days ago	<a href="#">Download</a>
3	 9 days ago	<a href="#">Download</a>

[Previous](#) 5 of 7 versions [Next](#)

# About the Frontend

## Proposed UI : Add Versions

Add Version

The proposed UI for adding a new version is titled "Add Version". It features an "Application Navbar" at the top. Below it, the "Project Plan" section shows it was created by "John Doe" 10 days ago and updated at a later date. A "Back" button and a green "Add Version" button are present. A central modal window titled "Add new Version" contains fields for "Project Plan" (with a file selection icon) and "Select a file in your local machine". It includes "Cancel" and "Store" buttons. Below the modal, two previous versions are listed: one from 5 days ago and another from 9 days ago, each with a "Download" button. Navigation buttons "Previous" and "Next" are at the bottom, along with a message "5 of 7 versions".

Application Navbar

**Project Plan**

Created by John Doe at 10 Days Ago

Updated At

Add new Version

Project Plan

Select a file in your local machine

Cancel Store

4 5 days ago Download

3 9 days ago Download

Previous 5 of 7 versions Next

# About the Frontend

## Proposed UI : Tasks

### Document Tasks

Application Navbar

#### Project Plan

Created by  John Doe at 10 days ago  
Updated At 3 days ago

[Back](#) [Add Task](#)

[Versions](#) [Tasks](#)

Records		
Description	Status	Actions
Define the scope	Completed 8 days ago	
Define milestones	Completed 7 days ago	
Define phases	Completed 7 days ago	
Define activities	Completed 5 days ago	
Define efforts	Pending	<a href="#">Finish</a> <a href="#">Edit</a> <a href="#">Remove</a>

[Previous](#) 5 of 5 tasks [Next](#)

# About the Frontend

## Proposed UI : Add Task

Add Task

Application Navbar

### Project Plan

Created by  John Doe at 10 days ago  
Updated At 3 days ago

Back Add Task

#### Add new Task

Project Plan

Description

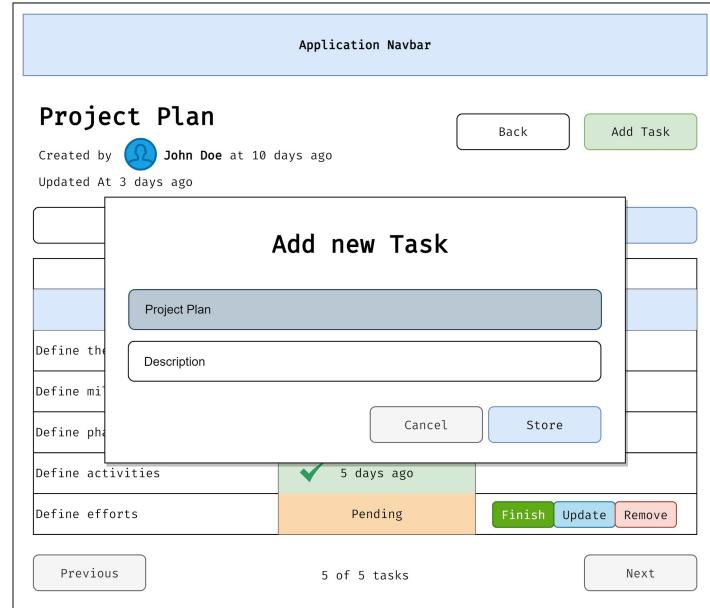
Cancel Store

Define activities 5 days ago

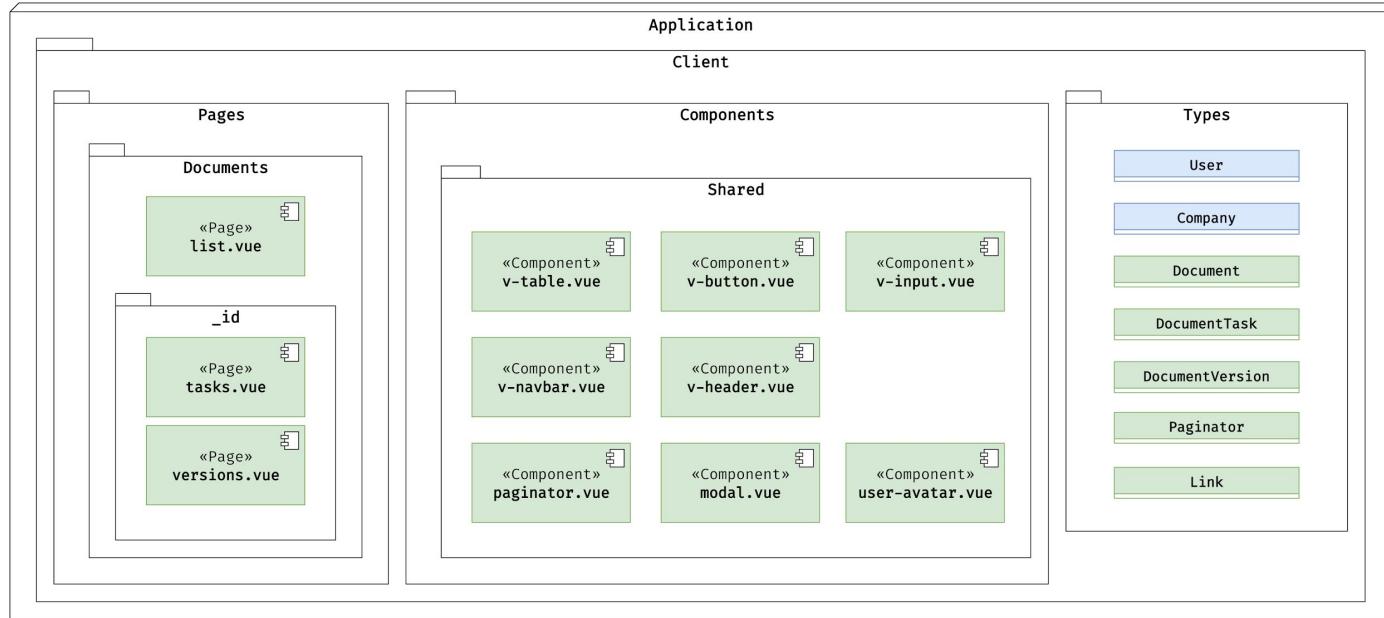
Define efforts Pending

Finish Update Remove

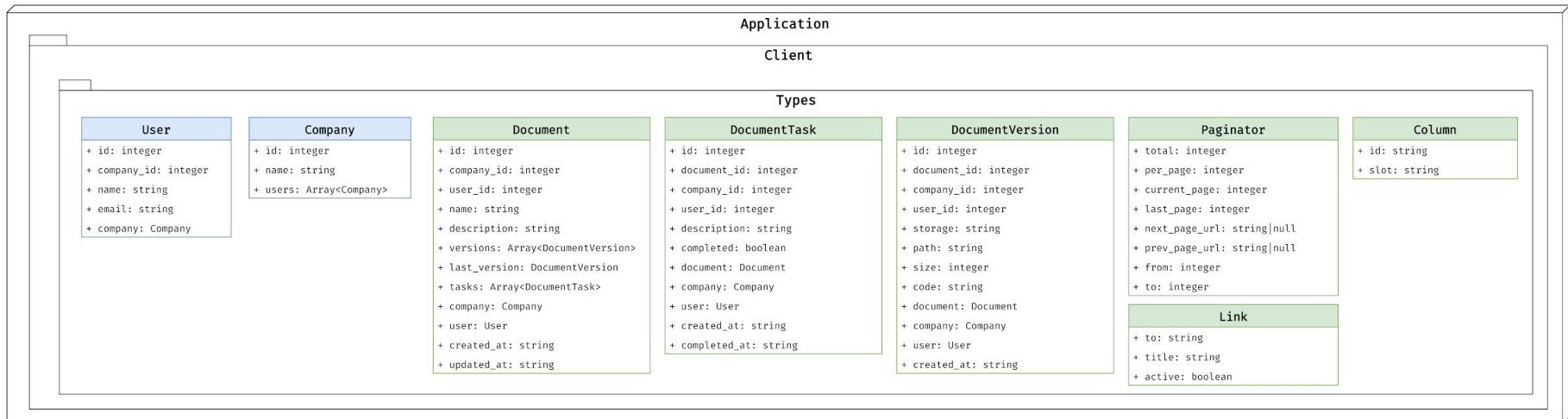
Previous 5 of 5 tasks Next



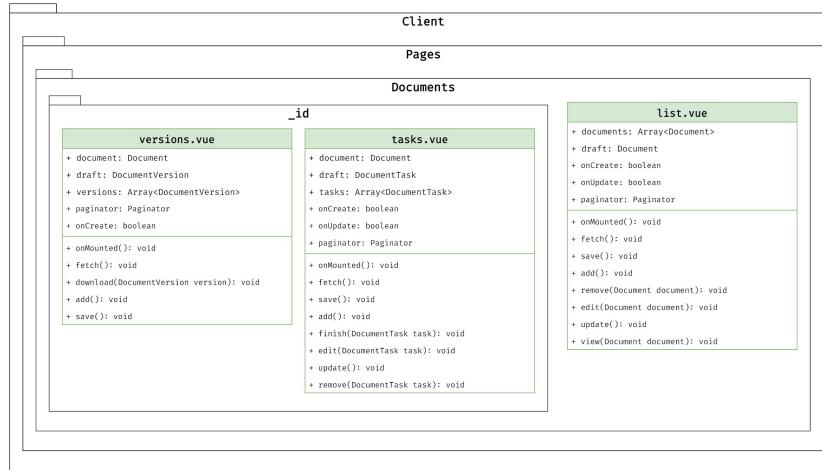
# Proposed Components



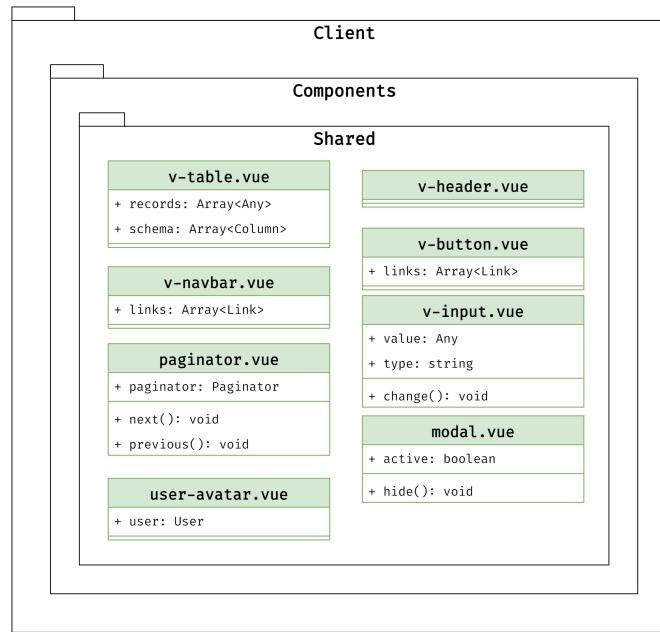
# Proposed Types



# Proposed Pages



# Proposed Components



# About the Resources

# About the **Resources**

## Basic premises

We have a Product Team and  
Development Team

We have UI design resources.  
We don't need define guidelines,  
We need to use it.

Development Team knows about:

- PHP/Laravel
- Nuxt / Vue.js
- MySQL/Redis
- AWS/Kubernetes
- GitHub Actions/ArgoCD

# About the Planning and Estimation

# About the Planning and Estimation

## Basic premises

1. We have 1 week sprints.
2. We have almost 3 weeks.
3. We can estimate using 1 to 10 points of efforts:
  - a. 1 is too easy and takes 1 hour.
  - b. 3 is almost easy but takes 6 hours.
  - c. 5 is not easy/hard and takes more than a day.
  - d. 7 is a little bit hard and takes more than 2 days.
  - e. 10 is hard so takes like 1 week.
4. We can prioritize using 1 to 10 points:
  - a. 1 is a not prioritized task.
  - b. 5 is a medium prioritized task.
  - c. 10 is absolute relevant task and must be done ASAP.
5. We can assign the last week for bug fixes / fast tracking sprint

# The Stories

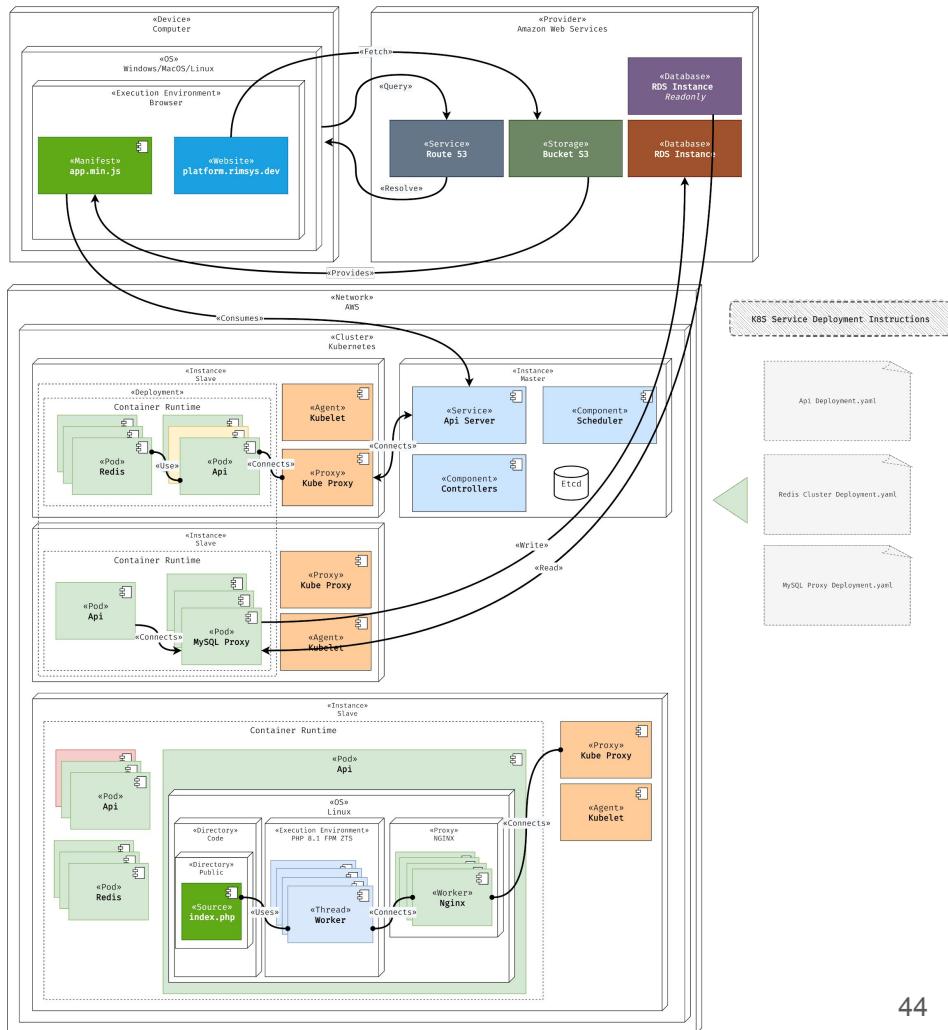
ID	Role	Goal	Scenario	Acceptance Criteria	Stories Dependencies	Tasks Dependencies
1	As a User	I want to view all the documents	So I can see the documents state	1. The documents must be listed using a paginated table. 2. The table must be sorted by the user companying and sorted by the updated or descendant. 3. The table must show: 3.1. The document code 3.2. The document update 3.3. The document description 3.4. The last version number or code 3.5. The available actions for every document 3.6. The available actions for every document	None	Backend Tasks: 1, 3, 7, 11 Frontend Tasks: 12, 13, 14, 15, 16, 17, 18, 19
2	As a User	I want to add documents	So I can push local files as documents to the system	1. The documents must be created individually using a form. 2. The form must include: 2.1. The document code field. 2.2. The document description field. 2.3. The document update field. 2.4. The local file must be used to create the first document version. 2.5. The file must be converted to the current meetings. 2.6. The create must be finish when the user confirms the creation by pressing "Store" button on the model. 2.7. The file must be uploaded to the system. 2.8. The model must disappear when the file is fully loaded.	1	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 13, 15, 17, 18, 19, 21
3	As a User	I want to update documents	So I can change the document details	1. The documents must be updated individually using a form. 2. The form must be displayed on a triggered modal view by pressing "Edit" button on the document table row. 3. The form must include: 3.1. The existing document name field. 3.2. The existing document description field. 4. The update must be finish when the user confirms the change by pressing the "Update" button on the modal. 4.1. The document must be updated to the current meetings. 4.2. The documents list must be reloaded after the update. 4.3. The document must be converted to the current meetings. 4.4. The update can be cancelled by pressing "Cancel" button on the modal to hide the form.	1	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 13, 15, 17, 18, 19, 24
4	As a User	I want to remove documents	So I can delete unused documents	1. The documents must be deleted individually using a form. 2. The form must be displayed on a triggered modal view by pressing "Remove" button on the document table row. 3. The form must include: 3.1. The warning message of the irreversible effect of the action. 3.2. The document name as the main field. 4. The update must be finish when the user confirms the delete by pressing the "Delete" button on the modal. 4.1. The document must be removed from the system. 4.2. The document must be converted to the current meetings. 5. The remove can be cancelled by pressing "Cancel" button on the modal to hide the form.	1	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 13, 15, 17, 18, 19, 27
5	As a User	I want to view the document versions	So I can see the document changes over the time	1. The versions must be listed when the user is looking the document table row. 2. The versions must be listed using a paginated table and sorted by the version code descendant. 3. The table must be sorted by the user companying and sorted by the updated or descendant. 4. The table must show: 4.1. The version code 4.2. The version created at. 4.3. The version description. 4.4. The available actions for every version.	1	Backend Tasks: 1, 2, 4, 7, 11 Frontend Tasks: 12, 13, 14, 15, 16, 19, 19
6	As a User	I want to add document versions	So I can update documents to the system using local files	1. The versions can be created when the user is looking the document versions. 2. The form must be displayed on a triggered modal view by pressing "Add version" button. 3. The form must include: 3.1. The document name field on reading field. 3.2. The document description field. 3.3. The version code must be the document last version plus one. 4. The version must be created individually using a form. 5. The version must be added to the document last version by pressing "Store" button on the modal. 6.1. The versions list must be reloaded after the create. 6.2. The document must be converted to the current meetings. 7. The create can be cancelled by pressing "Cancel" button on the modal to hide the form.	1,5	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 13, 15, 17, 18, 19, 22
7	As a User	I want to download document versions	So I can retrieve documents from the system storage	1. The versions can be download when the user is looking the document versions. 2. The versions must be listed using a triggered modal view on the version available actions.	1,5	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 13, 15, 17, 18, 19, 20
8	As a User	I want to view the document tasks	So I can see the pending and completed work	1. The tasks must be appear by pressing "Tasks" at the resignation when the user is looking the document versions. 2. The tasks must be listed using a paginated table. 3. The table must be sorted by the user companying and sorted by the completed_at descendant and the updated_, or descendant showing the not completed first. 4. The table must show: 4.1. The task status and if it's completed must include the updated_, or timestamp if not then must include the created_, or timestamp. 4.2. The task status and if it's completed must include the updated_, or timestamp if not then must include the created_, or timestamp. 4.3. The task status and if it's completed must include the updated_, or timestamp if not then must include the created_, or timestamp.	1,5	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 12, 13, 14, 15, 16, 17, 18, 19, 19
9	As a User	I want to add document tasks	So I can define pending work to be done on the document	1. The tasks can be created when the user is looking the document tasks. 2. The tasks must be created individually using a form. 3. The form must be displayed on a triggered modal view by pressing "Add task" button. 4. The form must include: 4.1. The task description field on reading field. 4.2. The task description field. 5. The task must be created when the user confirms the creation by pressing "Store" button on the modal. 5.1. The tasks list must be reloaded after the create. 5.2. The document must be converted to the current meetings. 6. The create can be cancelled by pressing "Cancel" button on the modal to hide the form.	1,5,8	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 13, 15, 17, 18, 19, 23
10	As a User	I want to update document tasks	So I can change the task description	1. The tasks must be updated when the user is looking the document tasks. 2. The form must be displayed on a triggered modal view by pressing "Edit" button on the task available actions. 3. The form must include: 3.1. The existing task description field on reading field. 3.2. The task description field on reading field. 4. The update must be finish when the user confirms the change by pressing the "Update" button on the modal. 4.1. The task must be updated to the current meetings. 4.2. The tasks list must be reloaded after the update. 4.3. The task must be converted to the current meetings. 5. The update can be cancelled by pressing "Cancel" button on the modal to hide the form.	1,5,8	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 13, 15, 17, 18, 19, 25
11	As a User	I want to remove document tasks	So I can delete non-required tasks	1. The tasks must be deleted individually using a form. 2. The form must be displayed on a triggered modal view by pressing "Remove" button on the task available actions. 3. The form must include: 3.1. The warning message of the irreversible effect of the action. 3.2. The task description field on reading field. 4. The update must be finish when the user confirms the delete by pressing the "Delete" button on the modal. 4.1. The task must be removed from the system. 4.2. The task must be converted to the current meetings. 5. The remove can be cancelled by pressing "Cancel" button on the modal to hide the form.	1,5,8	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 13, 15, 17, 18, 19, 28
12	As a User	I want to finish document tasks	So I can marks pending task as completed	1. The tasks must be finished individually using a form. 2. The tasks finish must be triggered by pressing "Finish" button on the version available actions.	1,5,8	Backend Tasks: 1, 2, 3, 6, 7, 11 Frontend Tasks: 13, 15, 17, 18, 19, 29
13	As a User	I want to have custom document management integration	So I can implement the system with my existing systems	1. The documents versions files must be stored on the customer existing document systems when are created.	Name	Backend Tasks: 1, 2, 3, 4, 6, 8, 9, 10

# The Tasks

ID	Requirements	Type	Scope	Estimation	Priority	Dependencies
1	Implement Models and Database Migrations	Technical	Backend	3	10	
2	Create Controllers, Policies, Requests and Observers	Technical	Backend	1	10	1
3	Implement DocumentsController and his related Requests	Technical	Backend	5	9	2
4	Implement DocumentVersionsController and his related Requests	Technical	Backend	2	8	2
5	Implement DocumentTasksController and his related Requests	Technical	Backend	2	8	2
6	Implement DocumentObserver created method to create the related document version one	Technical	Backend	2	6	2
7	Implement Policies methods and use on Controllers	Technical	Backend	2	4	3, 4, 5
8	Create StorageObject, StorageResolver and CustomStorage classes with empty methods	Technical	Backend	1	9	1
9	Implement StorageResolver methods using user company storage and token properties	Technical	Backend	3	8	8
10	Implement StorageResolver on oddVersion and store methods of DocumentsController when user company storage isn't empty	Technical	Backend	2	8	8
11	Implement Routers for Controllers	Technical	Backend	1	6	2
ID	Requirements	Type	Scope	Estimation	Priority	Dependencies
12	Implement Paginator, Column and Link Types	Technical	Frontend	1	10	
13	Implement Document, DocumentTask and DocumentVersion Types	Technical	Frontend	2	10	
14	Implement VTable component using VSlots and VSlotProps to render table headers and rows	Technical	Frontend	5	9	12
15	Implement VNavbar, VHeader, VButton and VInput components	Technical	Frontend	3	9	12
16	Implement Paginator component	Technical	Frontend	3	9	12, 13
17	Implement Modal, Paginator and User Avatar components	Technical	Frontend	3	9	12, 13
18	Create List.vue, Versions.vue and Tasks.vue including properties and empty methods	Technical	Frontend	5	9	1, 2
19	Implement List.vue, Versions.vue and Tasks.vue fetch methods using VTable, Column, Paginator and invoke in onMounted	Technical	Frontend	5	8	15
20	Implement Versions.vue download method to retrieve a binary response	Technical	Frontend	1	8	15
21	Implement List.vue create form template in a modal, add and save methods	Technical	Frontend	2	8	11, 15
22	Implement Versions.vue create form template in a modal, add and save methods	Technical	Frontend	2	8	11, 15
23	Implement Tasks.vue create form template in a modal, add and save methods	Technical	Frontend	2	8	11, 15
24	Implement List.vue update form template in a modal, edit and update methods	Technical	Frontend	2	8	11, 15
25	Implement Tasks.vue update form template in a modal, edit and update methods	Technical	Frontend	2	8	11, 15
26	Implement Tasks.vue finish method and button calling the finish task route	Technical	Frontend	2	8	11, 15
27	Implement List.vue remove method and button calling the destroy document route	Technical	Frontend	2	8	11, 15
28	Implement Tasks.vue remove method and button calling the destroy task route	Technical	Frontend	2	8	11, 15
ID	Requirements	Type	Scope	Estimation	Priority	Dependencies
29	Define user stories	Documentation	Product Management			
30	Define acceptance criterias	Documentation	Product Management			
31	Define fast tracking feature corrective plan	Product	Product Management			
31	Implement user test cases	Testing	Quality Assurance			
31	Implement integration test cases	Testing	Quality Assurance			
32	Automate test cases	Automation	Quality Assurance			
ID	Requirements	Type	Scope	Estimation	Priority	Dependencies
33	Design a customer document management integration process and procedures using BPMN	Business				
34	Design a customer document management integration plan (CDMI)	Business				
35	Define risk management concerns and include articles to the CDMI	Business				
36	Define technical requirements to include as articles to the CDMI	Technical/Business				
37	Offer the document management integration to customers	Business				

# About the Deployment

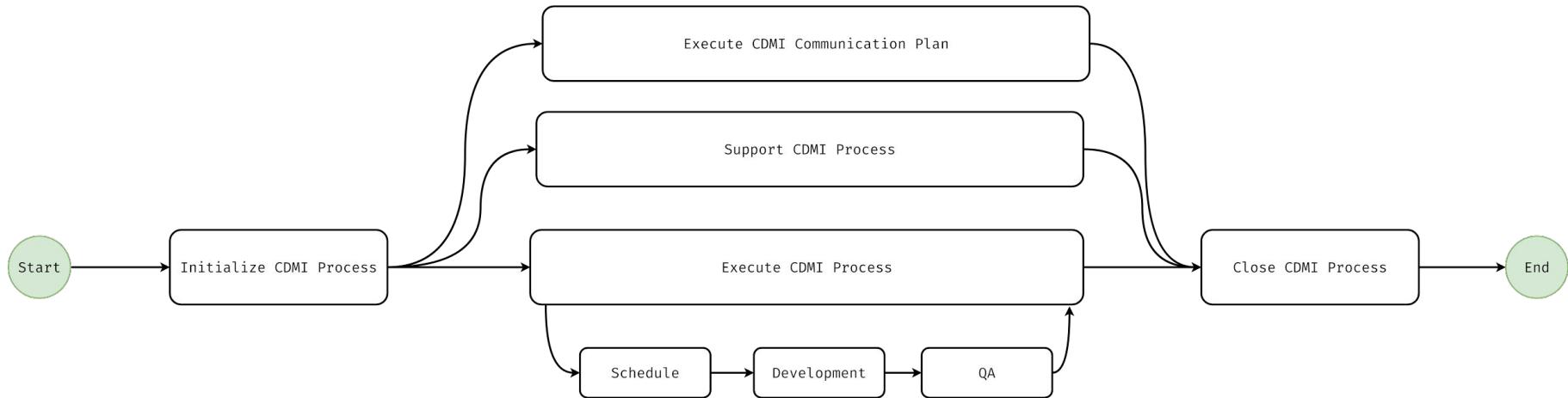
# About the Deployment Diagram



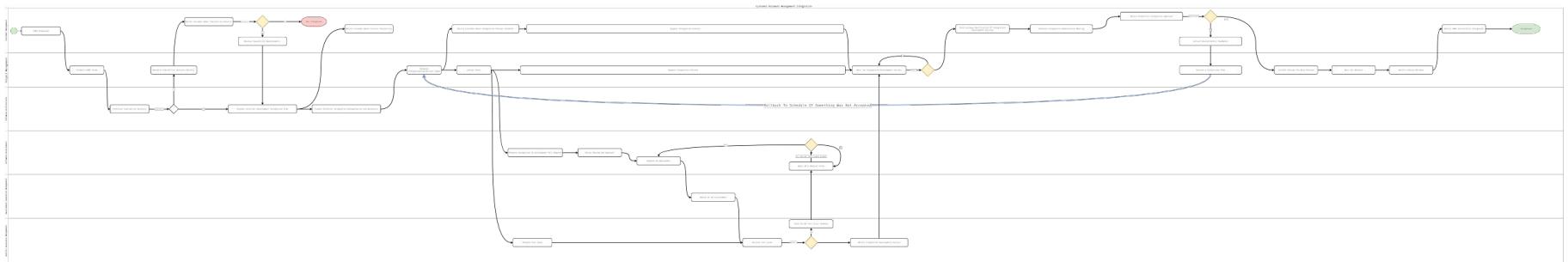
# About the Integrations

# The Simplified Process

Customer Document Management Integration (CDMI)



# The Process



# About the Conclusions

# About the **Conclusions**

With the previous results we can:

- Solution implements the features described on slide 2.
- Offer technical resources as diagrams and user stories to the development team in order to reduce the development risks and optimize the time on feature implementations.
- Offer a scaffolding to define a customer document management integration process.
- Offer a well-organized user stories and tasks backlog to make sprint plannings with less difficult as possible.