POLITECNICO
MILANO 1863

Software Engineering 2 Project
Academic Year 2021 - 2022

DREAM

Design Document

Valeria Detomas Sofia Martellozzo

Professor

Elisabetta Di Nitto

Version 1
January 3, 2022

Contents

1 Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, Abbreviations

1.4 Revision history

1.5 Reference Documents

1.6 Document Structure

2 Architectural Design

2.1 High-level components and their interaction

In the following section it is provided a high level view of the architecture of the system, which is structured following the three logic layer:

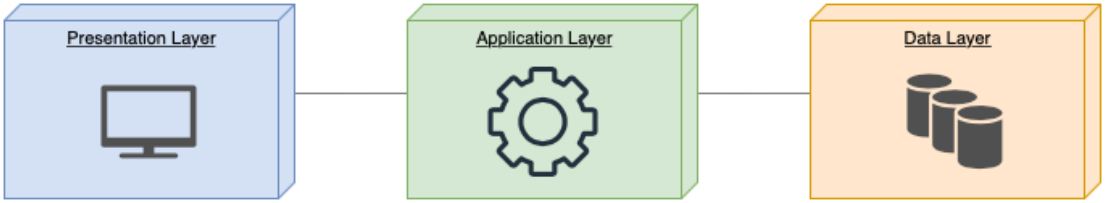

Figure 1: three layer architecture

- **Presentation Layer (P)**: The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. This top-level tier can run on a web browser, as desktop application, or a graphical user interface (GUI), for example.

  Web presentation tiers are usually developed using HTML, CSS and JavaScript. Desktop applications can be written in a variety of languages depending on the platform.

- **Business Logic or Application Layer(L)**: The application tier is the heart of the application. In this tier, information collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier.

  The application tier is typically developed using Python, Java, Perl, PHP or Ruby, and communicates with the data tier using API calls.

- **Data Layer (D)**: The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed. This can be a relational database management system such as PostgreSQL, MySQL, MariaDB, Oracle, DB2, Informix or Microsoft SQL Server, or in a NoSQL Database server such as Cassandra, CouchDB or MongoDB.

In this case the system is a distributed application that follows the client-server paradigm: it is a two-tier architecture, consisting of a presentation and a data tier. the business logic lives in the data tier only. In a two-tier application,

all communication goes through the application tier. The presentation tier and the data tier cannot communicate directly with one another.

Client and server are being allocated into different physical machines and their communication takes place via other components and interfaces, located in the middle of the structure and composed by hardware and software modules. The client is a Web Application, with is by definition a thin client, because of its total dependency from the server; so it only contains the presentation layer.

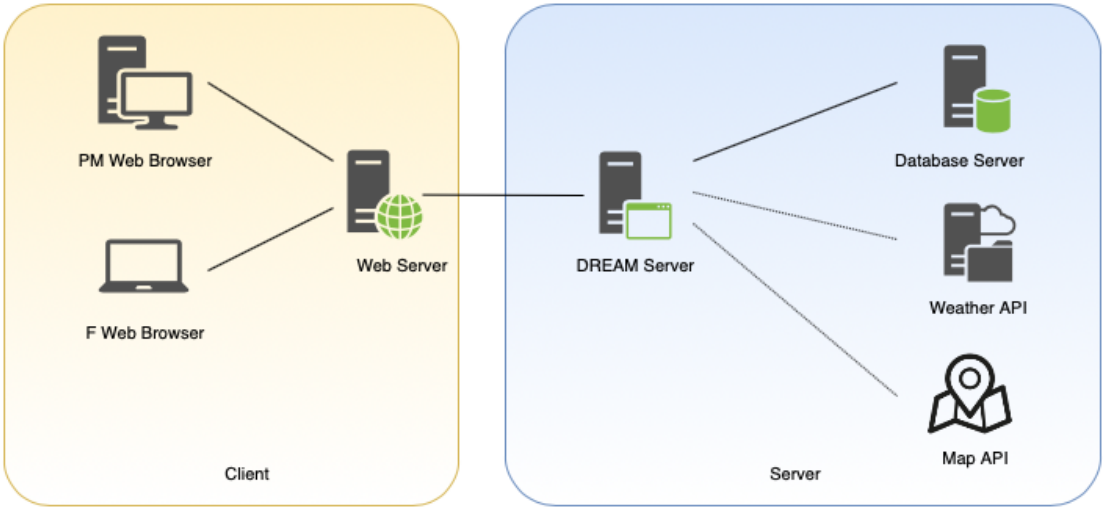

Figure 2: Dream system diagram.

- **Server side:**
  - ApplicationServer (DREAM Server): it is the central point of the system. Is a server whit all the application logic, that communicates with the other servers.
  - Database Server: this is the server where all the application data are stored.
  - Weather API: external API used to retrieve data about weather in the territory. This information will be used to fill each farm page.
  - Map API: external API used to retrieve data about the territory.
- **Client side:**
  - Policy Maker Web Browser: browser used by the policy maker from their work desk to access to the system
  - Farmer Web Browser: browser used by the farmer to access to the system

2.2 Component view

In this section it is provided a description of the components and interfaces of the system, how they are organized internally and how they communicate with each other.

2.2.1 High Level

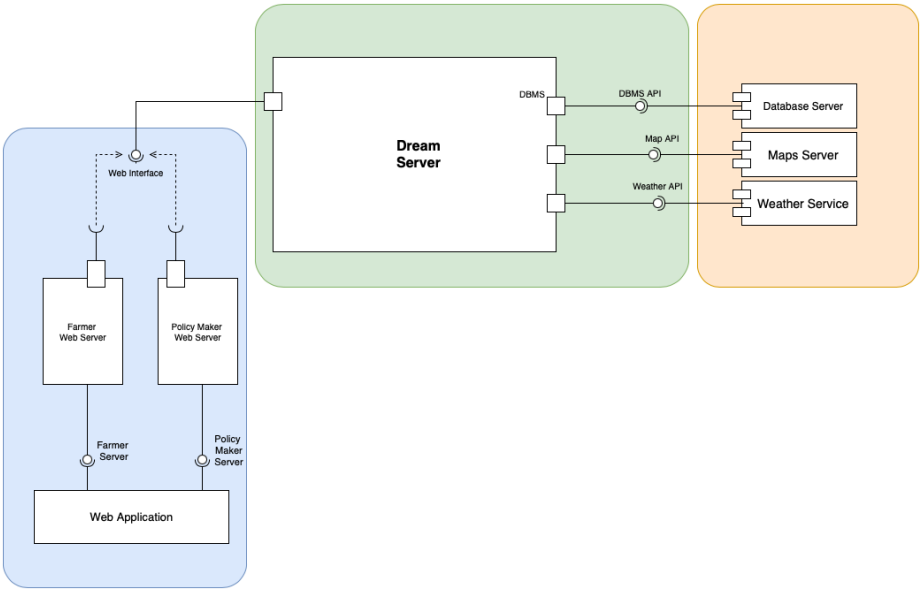

Figure 3: Component view.

- **Dream Server**: this component contains the hole application logic of the system. The interfaces it provides allow the user, both farmer and policy maker, to comunicate with the server; them also allow the server to interact with external system that provide different kind of data.
- **Web Application**: it rapresents the web application reachable by any web browser. It is the first component that any user uses to connect with the system.
- **Farmer Web Server**: it provides the interface to a farmer to interact with the system. It has the minimum logic to make visible the content of the application provided by the server of the system.
- **PolicyMaker Web Server**: the same as the one above but specific for the policy maker.

- **Database Server**: it provides the interfaces in all the processes that need to require or store information from the database of the system
- **Maps Server**: it provides the interfaces when the Dream server needs the maps data to make it visible and fill it with all the farms in the sistem, locating them by the position provided from their owner in the registration phase.
- **Weather Server**: it provides the interfaces to Dream server to retreive weather data of the territory.

2.2.2 Server

More specific and detailed on the server inner components.

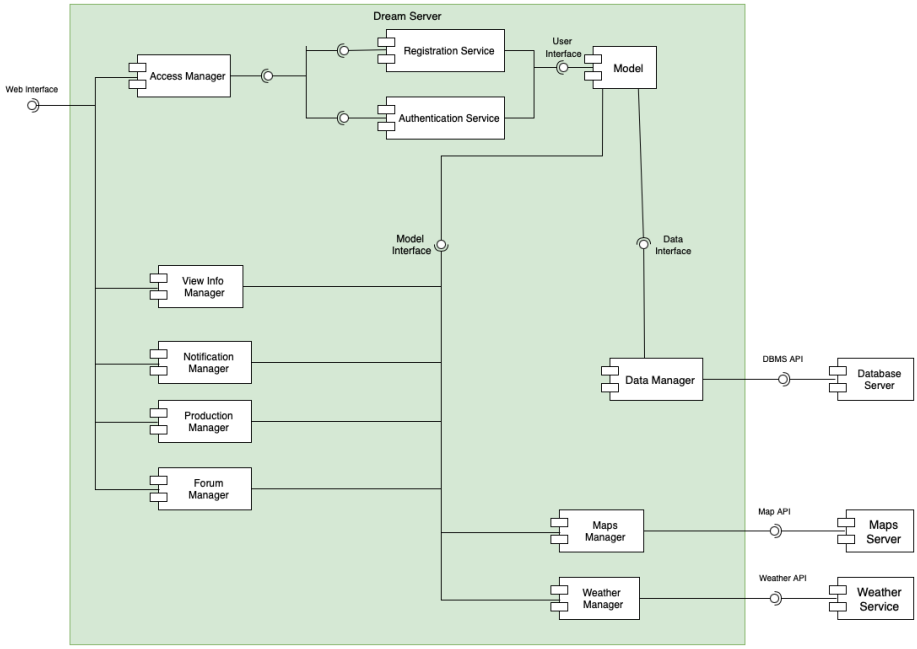

Figure 4: Inner server component view2.

- **Access Manager**: this component provide the Web interface to the user and make login and sign up opratation possible. It recognize the operation required and if it is permitted (sign up only for new farmers) comunicate with the right component that follow.
- **Registration Service**: if a request of a new registration occupr, it check the credential submitted by the new user and create the object that represent it, that will be stored in the Model; in fact it has an iterface to comunicate with the Model.

- **Authentication Service**: this component is called by the Access manager when an authentication request occur. In this case it collect the data about this specific user and verify if the credential submitted correspond. When them maches it let the user to access the system, otherwise generate an error signal.
- **ViewInfoManager**: this component is used by the system to collect the data from the Model request by the user. It operates when a farm's page has to be constructed with all the information visible in it.
- **Notification Manager**: this component has the ability to differ based on the notification it receives:
  - advice: when a farmer submit this type of notification his only duty is the one to make possible the saving of it.
  - help: when a farmer submit a request of help this component select randomly one policy maker (from the ones saved in the system) and send him the notification.
  - solution: when a policy maker decides to respond to a request of help, he send this type of notification. This componenet forward it to the addressee.
  - evaluation: when a policy maker evaluate a farm, this component takes care of sending it to the owner of the farm.

  For all thee four type of notifications it attach the current day and time of the subbmission. This component also make possible to the user to visualize the list of all notifications received and, if asked, visualize in details the one selected.
- **Production Manager**: this component, by its connection to the model via Model interface, povides all production data required for a specific farm.
- **Forum Manager**: this component manage all the messages comunication between farmers via forum. It gets all the messages in it, saved as List in the Model and comunicate to it the new ones, that has to mìbe saved (attaching to it the date and time).
- **Model**: this component have a main role in the system because it rapresents the data so all other components needs to interact with it, to ptovide the page and informations required.
- **Data Manager**: it manages all the process where data are needed by the system. It uses methods provided by the DBMS API to execute queries on the database and object-relational mapping to the Model.
- **Maps Manager**: this component is used to retreive map data, in order to provide a visualization of where each farmer registered in the system are located in the territory.

- **Weather Manager**: this component is used to retreive weather data specific in the position of the farms. It map to the Model the information from a relational to an object construct as a structure where the key is the farm position and in the value another structure to linked each day the weather and temperature.

2.3 Deployment view

The deplyment diagram in figure .. shows the allocation of the software components in the physical tiers of the system. The system is organized into a three-tier application. This type of architecture can be beneficial because each tier can be developed in parallel and maintained as single modules on separate platforms.

- **Presentation Tier**: it is the user interface of the application, where the user interacts with the application. In this case it can only be a computer with a web browser running on an operating system (for example MacOS). It is composed by the web app an the web server. The web server is responsible for the communication between application server and the client.
- **Application Tier**: it is the logic tier of the application. Where all the information collected in the previous tier are processed using business logic. This tier is also responsible for the communication with the data tier through the DBMS gateway. The most important thing is that all communication of the system goes through this tier.
- **Data Tier**: it is a database server for managing read and write access to the database. Therefore all the information processed by the application tier are stored and managed here. Data tier is also independent of the two previous tiers.

2.4 Runtime view

In this section the focus is on the specific (dynamic) interaction between the components of the system; in other world the behaviour of the system at runtime. The functionality offered by the component are the same as the ones in the RASD, but this time the focus is on how the internal components provides it. (MAKE SURE THAT THE OPERATION AND COMPONENT ARE ALL DEFINED IN THE COMPONENT DIAG/VIEW, in particular the interfaces that receives the msg)

1. **farmer registration**
   In this sequence diagram is shown the sign up operation by a new farmer. After the user fill the form provided by the web application with: name, surmane, farm's name, position, email and password. The Dream server stores all this information as a Farmer object in the Model, and also saves

them in the database. At the end of the process redirects the user to the login page, only if some error occurred the data are not saved and the user is asked to repeat the operation.

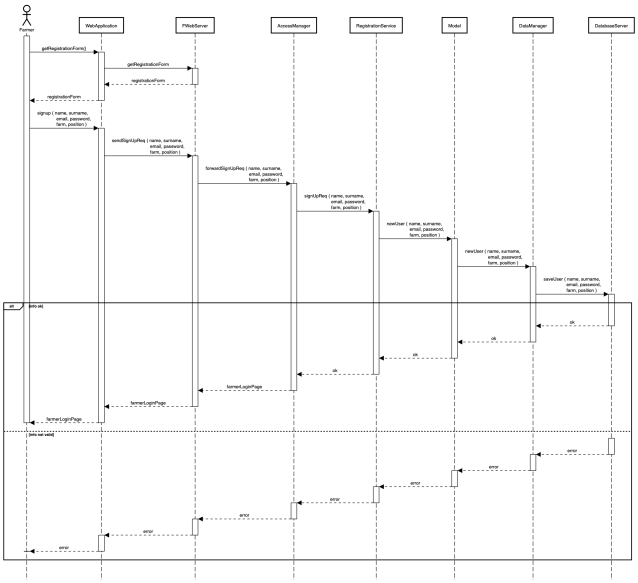

Figure 5: *SignUp* sequence diagram

2. **login**
   In this sequence diagram the process of user login is shown, at first for the farmer and then for the policy maker. For both are almost the same, except that the farmers needs to provide email and password as access credentials and the policy maker a code and password. Another difference is in the server that forward the request to login to the Dream server component. If the credentials are wrong or missing the system gave an error message, if not the home page of the user is returned.

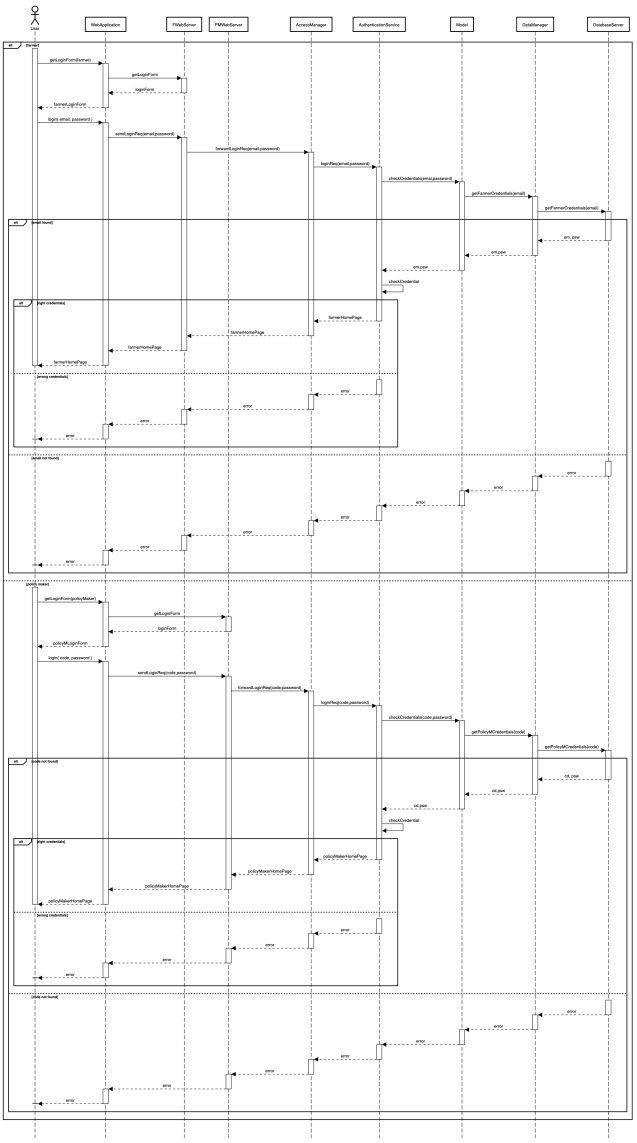

Figure 6: *Login* sequence diagram

3. **farmer read and write on forum**
   In this sequence is shown how the system provides the forum with all the messages yet sended after the request of a farmer and a form where he can write a message and a button to send it. As shown, due to the Forum Manager, when a message is send it is saved whit the references of the sender and the date and time. These addictonal information are used to

sort the messages by the timestamp and specify the sender name near the message itself.

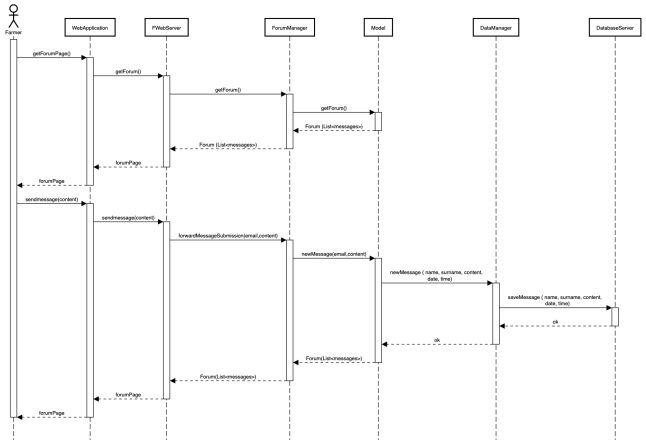

Figure 7: *Save message* sequence diagram

4. **farmer submit production data**
   Here is shown how a farmer can submit new information about his production. The Web Application provides him a form to fill with the type of production, the quantity collected and the date on which he collected it. The Server, or better its Production Manager component, build this information as an object collected in the Model and saved then in the database.

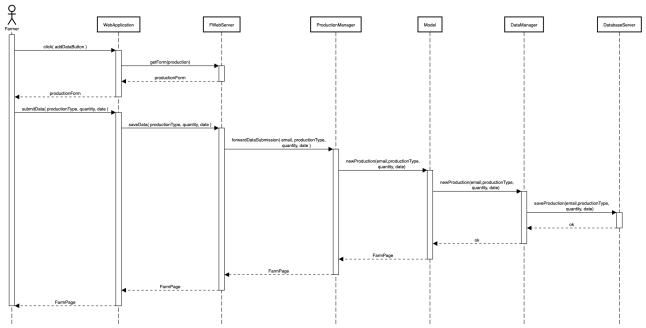

Figure 8: *Submit production data* sequence diagram

5. **map visualization**
   Here is shown how the system provides the visualization of the map to the user, with different level of visibility based on their permission.

- the user is a farmer: the Model contruct the object that rapresenting the map filled with all the farms saved in the system and only the type of production that are producted in it.
- the user is a policy maker: the Model contruct the map object with the farms located on it with basic production data and the evaluetion of it.

From the Map page is possible, by a click on a specific button, for policy maker to evaluate a farm.

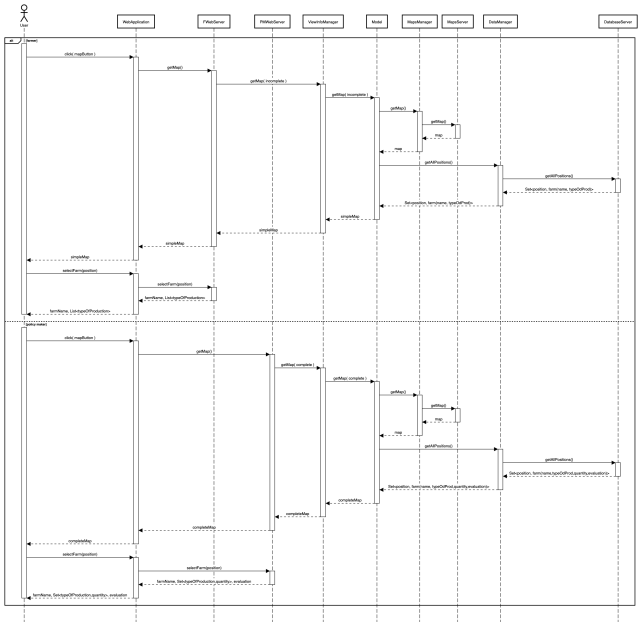

Figure 9: *Visualize map* sequence diagram

6. **policy maker makes an evaluation**
   –Description–

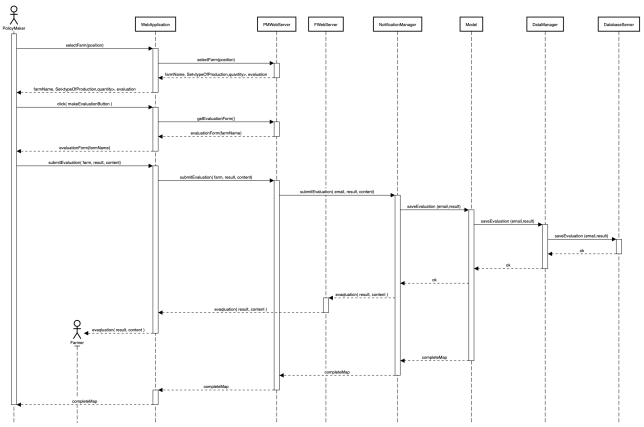

Figure 10: *Make evaluation* sequence diagram

7. **farm's page visualization**
   –Description–

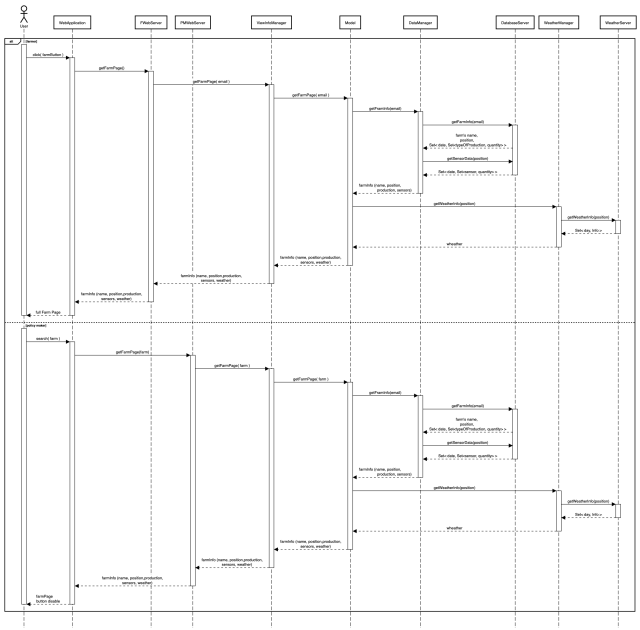

Figure 11: *Visualize farm page* sequence diagram

8. **notification submittion**
   –Description–

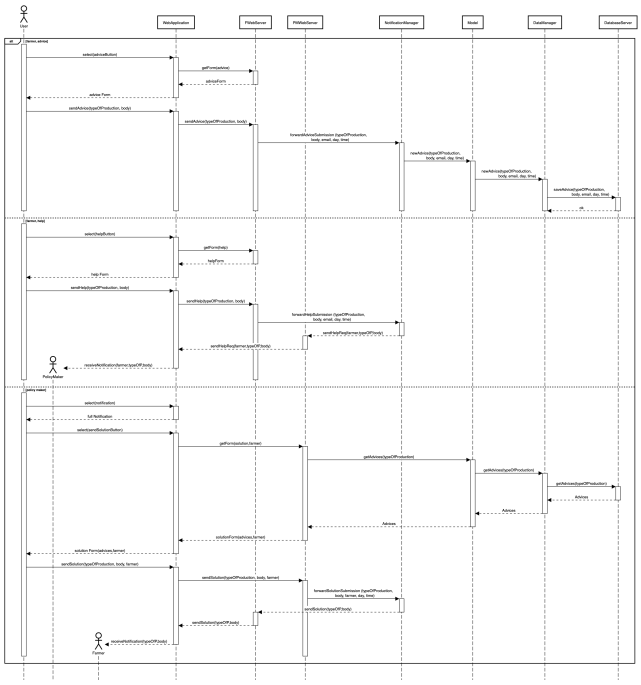

Figure 12: *Send notifications* sequence diagram

9. **notification visualization**
   –Description–

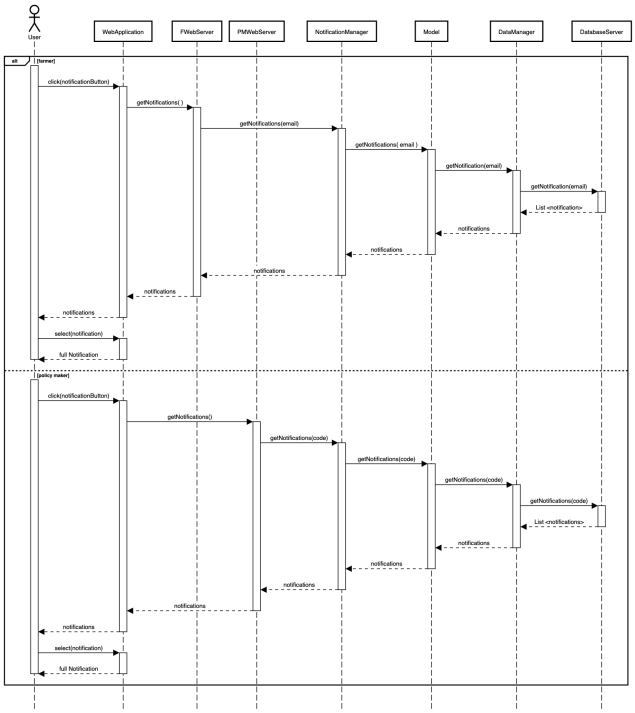

Figure 13: *Visualize notifications* sequence diagram

2.5 Component interfaces

2.6 Architectural styles and patterns

2.7 Other design decisions

3 User Interface Design

In this section there is a complete view of hot the web application is going to look. It includes diagrams that show hot the users can navigate in the user interfaces offered by the aplication. Its main purpose is to describe in details the mockups that are already presented in the RASD (reference). Therefore it is easier to understand the main features that the system offers.

3.1 UX diagrams

This subsection focuses on the flow of the windows of the web application, both for farmers and policy makers.

4 Requirements Traceability

5 Implementation, Integration and Test Plan

6 Effort Spent

Student	Time for S.1	Time for S2	Time for S.3	Time for S.4	Time for S.5
Valeria Detomas	h	h	h	h	h
Sofia Martellozzo	h	h	h	h	h

7 Software and Tools used

- LaTeX as document preparation system
- Lucidfor the state chart
- SequenceDiagram.org for the sequence diagrams
- Umletino for UML diagram
- Diagrams for the use case diagram
- Balsamig for the mockups
- **Alloy** as a model analyzer
- GitHub as verion control system.

8 References

- Specification document: R&DD Assignment A.Y. 2021-2022
- alloytool.org : Alloy Documentation