AY 2021/2022

Politecnico di Milano

DD: Design Document

Ottavia Belotti Alessio Braccini Riccardo Izzo

Professor
Elisabetta Di Nitto

Version 1.0
December 7, 2021

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a full technical description of the system described in the RASD document. In this design document we discuss about both hardware and software architectures in terms of interaction among the components that represent the system. Moreover there are mentions about the implementation, testing and integration process. This document will include technical language so is primarily addressed to programmers but stakeholders are also invited to read it in order to understand the characteristics of the project.

1.2 Scope

The scope of this design document is to define the behavior of the system in both general and critical cases, and to design the architecture of the system by describing logical allocation of the components and the interaction between them. This document also extends in part to the implementation and testing plan, where one possible course of action is explained, user interface design of user applications and requirements traceability relating to the RASD.

1.3 Definitions, acronyms, abbreviations

Acronyms

- **DREAM**: *Data-driven predictive farming*
- **RASD**: Requirement Analysis and Specification Document
- **DD**: Design Document
- **API**: Application Programming Interface
- **DBMS**: Database Management System
- **UML**: Unified Modeling Language
- **GPS**: Global Positioning System
- **IT**: Information Technology

- **GUI**: Graphic User Interface

1.4 Revision history

1.5 Reference documents

- Specification document: "Assignment RDD AY 2021-2022"
- Requirements Analysis Specification Document (RASD)
- UML documentation: https://www.uml-diagrams.org/
- ArchiMate documentation: https://pubs.opengroup.org/architecture/archimate3-doc/
- Slides of the lectures

1.6 Document structure

- **Section 1** gives a brief description of the design document, it describes the purpose and the scope of it including all the definitions, acronyms and abbreviations used.
- **Section 2** delves deeply into the system architecture by providing a detailed description of the components, the interfaces and all the technical choices made for the development of the application. It also includes detailed sequence, component and ArchiMate diagrams that describes in depth the system.
- **Section 3** contains a complete description of the user interface, it includes all the client-side mockups with some graphs useful to understand the correct execution flow.
- **Section 4** links the RASD and the DD, it maps the goals and the requirements described in the RASD to the actual functionalities presented in this DD.
- **Section 5** presents a description of the implementation, testing and integration phases of the system components.

2 Architectural Design

2.1 Overview

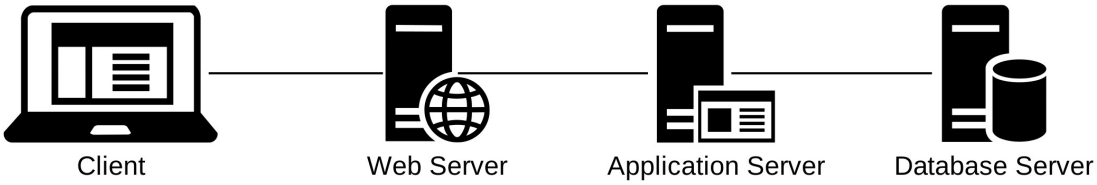

Figure 1: Four-Tier Architecture Scheme

The system is a distributed application that follow the common client-server paradigm. The architecture of the application is structured in three logic layers:

- **Presentation Layer (P)**: it manages the presentation logic and handles the user actions. It is characterized by a GUI (Graphic User Interface) that allow the user to interact with the application in a simple and effective way.
- **Logic or Application Layer (A)**: it manages all the functionalities that has to be provided to the users, it is also responsible of data exchange between the client and the data sources.
- **Data Layer (D)**: it manages the access to data sources, it gets data from the database and move them through the other layers. It is essential to guarantee a high level of abstraction from the database in order to provide a model easy to use.

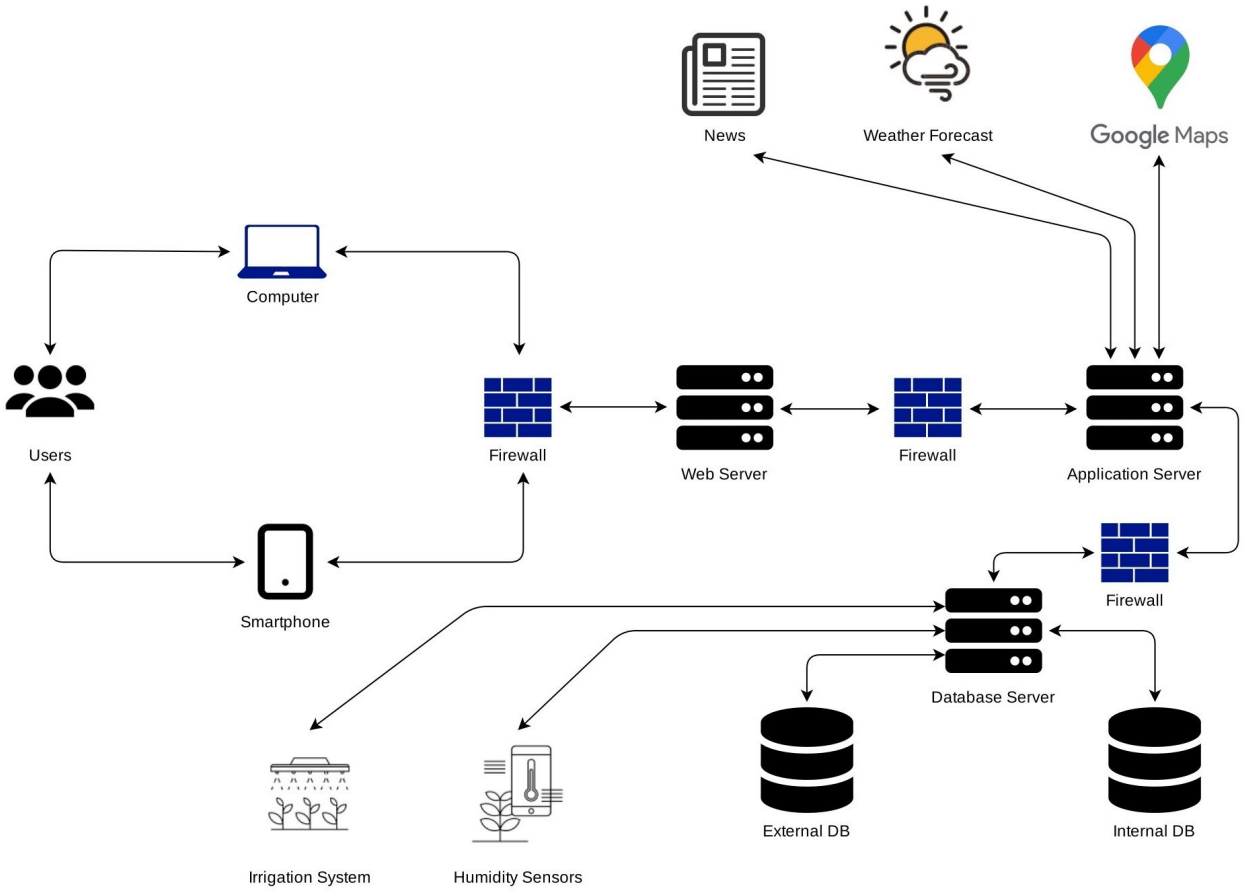

Figure 2: High Level Architecture

The system, as shown in *Figure (1)*, is based on a four-tier architecture (Client, Web Server, Application Server and Database Server), this ensures more flexibility and high scalability. The tiers are separated by firewalls in order to guarantee a higher level of security of the whole system. A thin client is used to prevent heavy computation load client side, all the heavy operations are executed at the server side. The client's devices can be a personal computer or a mobile device, both communicate directly with the Web Server through a web browser. The Application Server communicates

with the Database Server and transform data with business logic. It also manages the news and the weather forecast services. This allow to retrieve data like the weather directly from it in order to show them to the user. Finally, to enable the geolocalization, the Application Server uses the API provided by Google Maps. For what concern the other components, such as the water irrigation system and the humidity sensors, they are connected to the Database Server that exchange data with the Application Server. The Web Server also manages the data coming from the irrigation system and the humidity sensors. All the components will be described in depth in the following sections.

2.2 Component view

General Component View

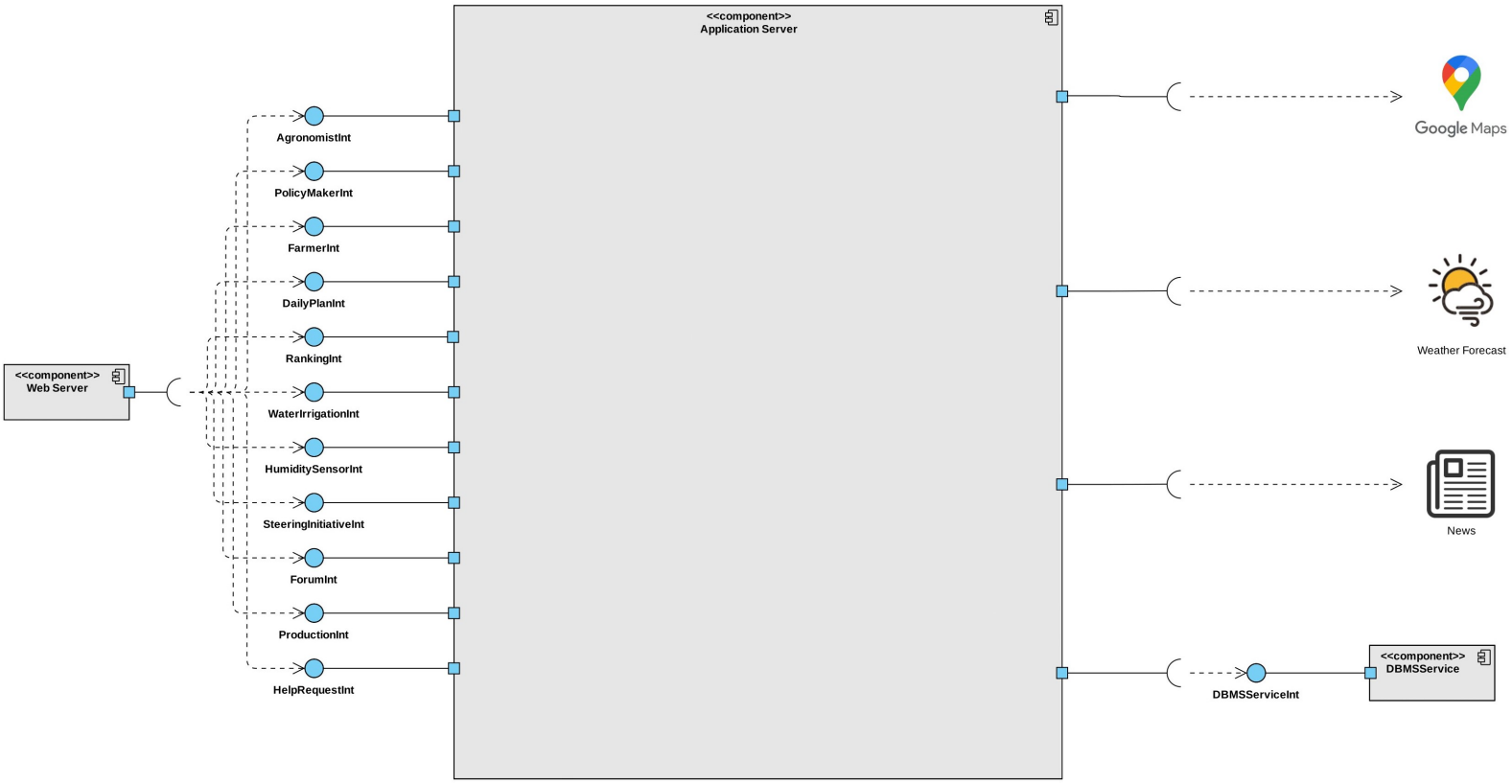

Figure 3: General Component Diagram

This image gives an high level representation of the components of the system. On the left are shown the provided interfaces between the Web Server and the Application Server that in this scheme is represented as a "black box", a complete description of it is provided in the next section. All the interfaces basically represents the main functionalities requested by the client application. On the right there are the requested interfaces, one of this

is responsible of the geolocalization and is provided by the Google Maps API. Moreover there are two interfaces that manage the weather forecast and the news services. Finally, the DBMS interface manages the DBMS service and is responsible of the communication between the Application Server and the Database Server.

Application Server Component View

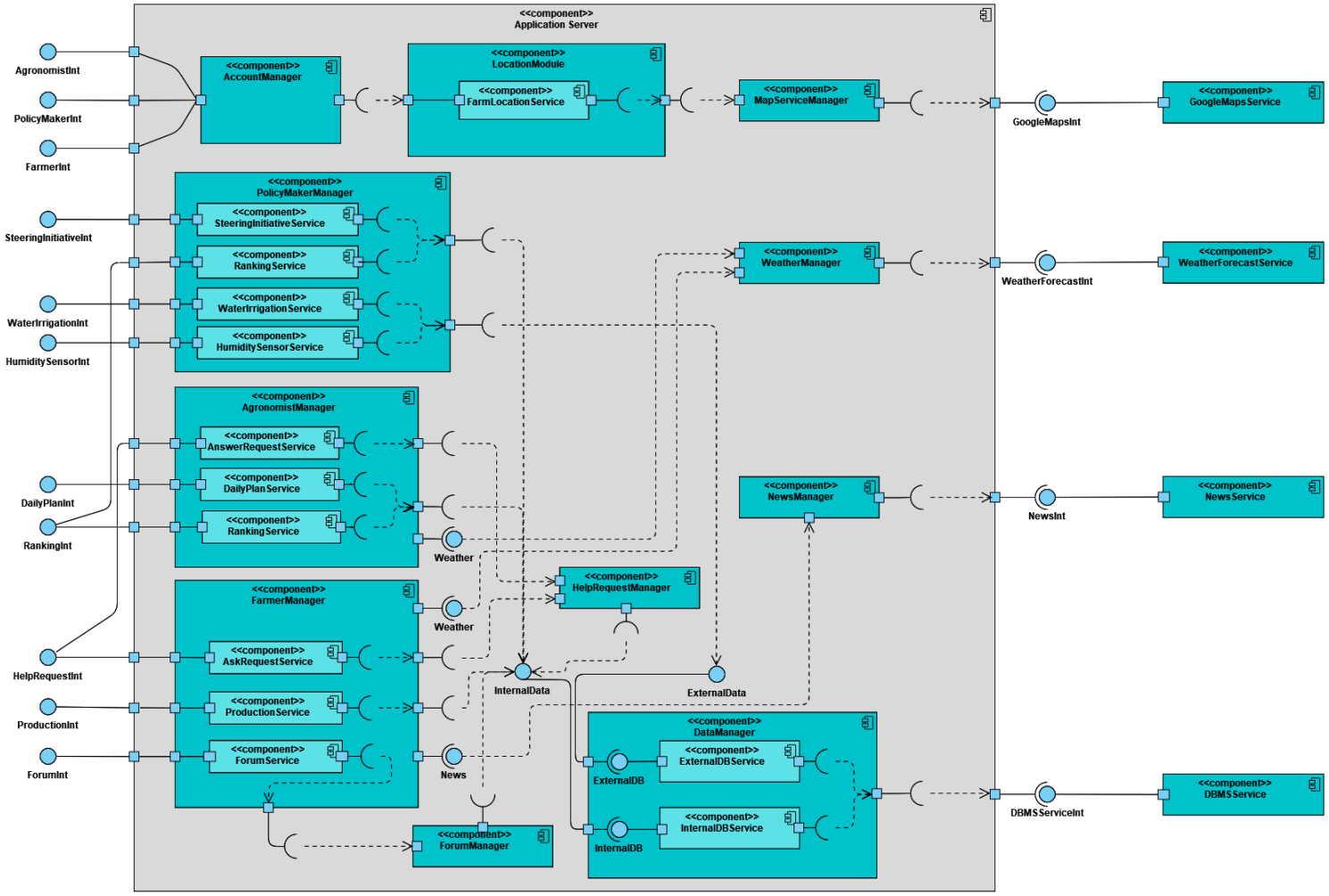

Figure 4: Application Server Component Diagram

The following component diagram gives a detailed view of the Application Server, it shows the internal structure and the interaction between the components. External elements in the diagram are represented in a simplified way.

- **AccountManager**: this component handles all the basic requests made by the client. This includes the authentication process that is composed by the log in, the log out and the sign up services. It also includes the edit profile process responsible of the editing of the user account. Once a user is logged in, all the specific functionalities are provided by the component that manages this type of user. Finally it communicates with the *LocationModule* that provides the geolocalization service.

- **LocationModule**: this module provides the interface that allow the geolocalization of the farm based on the GPS and on the address provided by the farmer. In order to provide this service it communicates with the map service manager.

- **MapServiceManager**: this component communicates directly with the external API provided by Google Maps, it provides informations regarding the region and allow the user to visualize the map of a specific location. Mainly it adapts the data received by the API in a comprehensible way for the other components, it also manages the API requests.

- **PolicyMakerManager**: it manages the policy maker services, these include: steering initiative, ranking, water irrigation and humidity sensors. The steering initiative service and the ranking service are mapped through an interface to the internal database, on the other side the water irrigation service and the humidity sensor service are considered external services and for this reason are linked to the external database.

- **AgronomistManager**: it manages the agronomist services, these include: answer help request, daily plan and ranking. It includes the weather external interface, it communicates with the weather component in order to provide the weather forecast service to the agronomist.

The daily plan service and the ranking service are linked to the internal database, instead the answer request service communicates directly with the *HelpRequestManager* component.

- **FarmerManager**: it manages the farmer services, these include: ask help request, production and forum. It includes two external interfaces: the first one communicates with the *WeatherManager* component in order to provide the weather forecast service to the farmer, the other one communicates with the *NewsManager* component. In this case only the production service is linked directly to the internal database, the ask request service is linked to the *HelpRequestManager* component while the forum service expose an interface to connect to the *ForumManager* component.

- **ForumManager**: this component manages the forum section, in particular it is responsible of the management of all the topics with the related messages between farmers.

- **DataManager**: it provides access to the external interface of the database, it manages queries and interacts with both the internal and the external databases. It includes two components: one is responsible of the operations with the external database, the other of the internal database. Both these components expose the related interface in order to use the service. This component is connected to the *DBMSService* external component, this establishes the connection between the Application Server and the Database Server.

- **HelpRequestManager**: it manages the help requests between farmers and agronomists. It communicates with the *FarmerManager* component for the help requests to ask, on the other way it communicates with the *AgronomistManager* component for the ones to answer. Finally it provides an interface that communicate with the internal database in order to store the requests.

- **NewsManager**: this component manages the service related to the news, it provides updated suggestions about crops and fertilizers for the farmers. To do this it interacts with the external component *NewsService*.

- **WeatherManager**: this component manages the service related to the weather forecast. It interacts with the external component *WeatherForecastService* in order to get the weather data.

2.3 Deployment view

2.4 Runtime view

Here are represented the runtime view of some relevant uses case of the system through some sequence diagram. In the diagrams is omitted the part regarding the user because it has been considered superfluous to the understanding of the interaction. In successive diagrams some part like the login phase or the come back to home page were omitted for the above motivations.

Sign Up

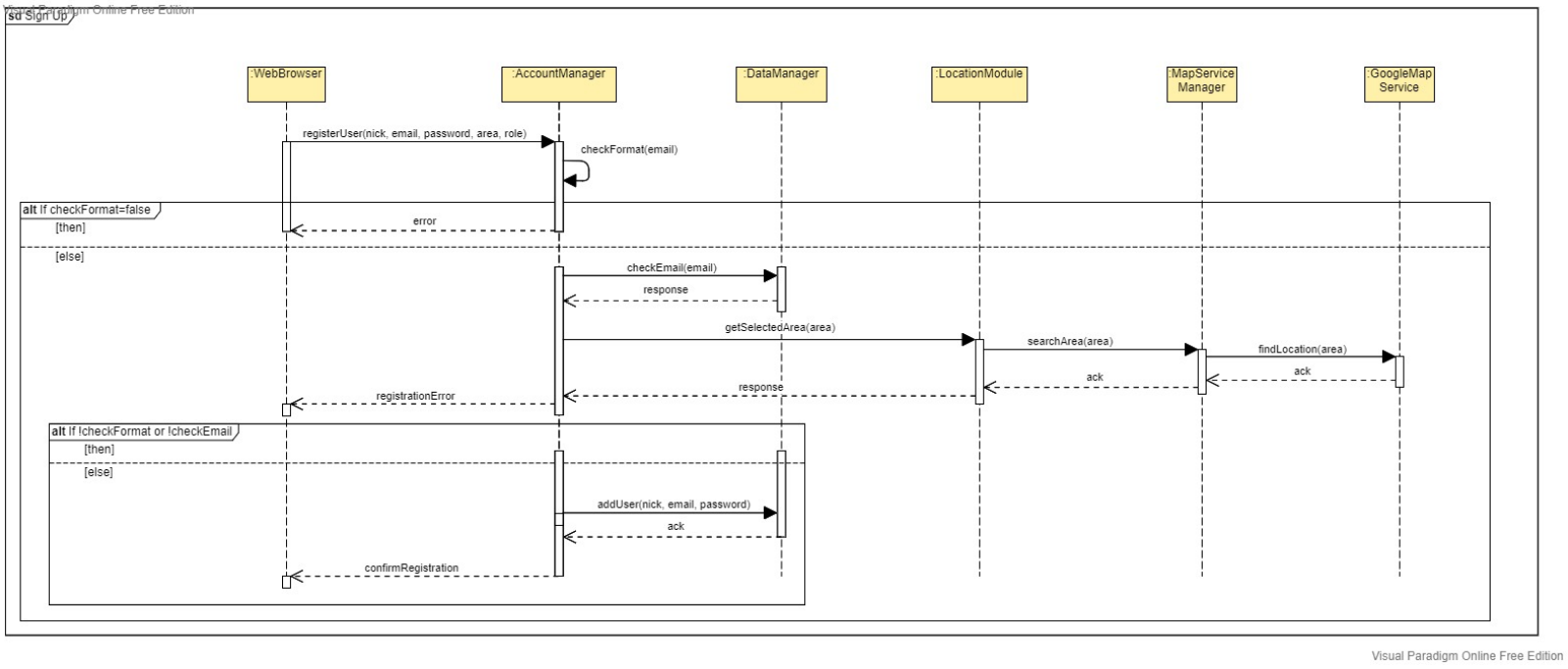

Figure 5: Sign Up phase

Sign up is the usual, user register itself by putting username, email, password job role and area. If the user is a farmer he put also the crop type. Location module and Map service are used in this phase in order to allow the users to select their location using the GPS. However user can also choose to select manually its location without the use of GPS.

At the end of the process, if every checks were ok the user is saved into the database by the Data Manager component.

Log In

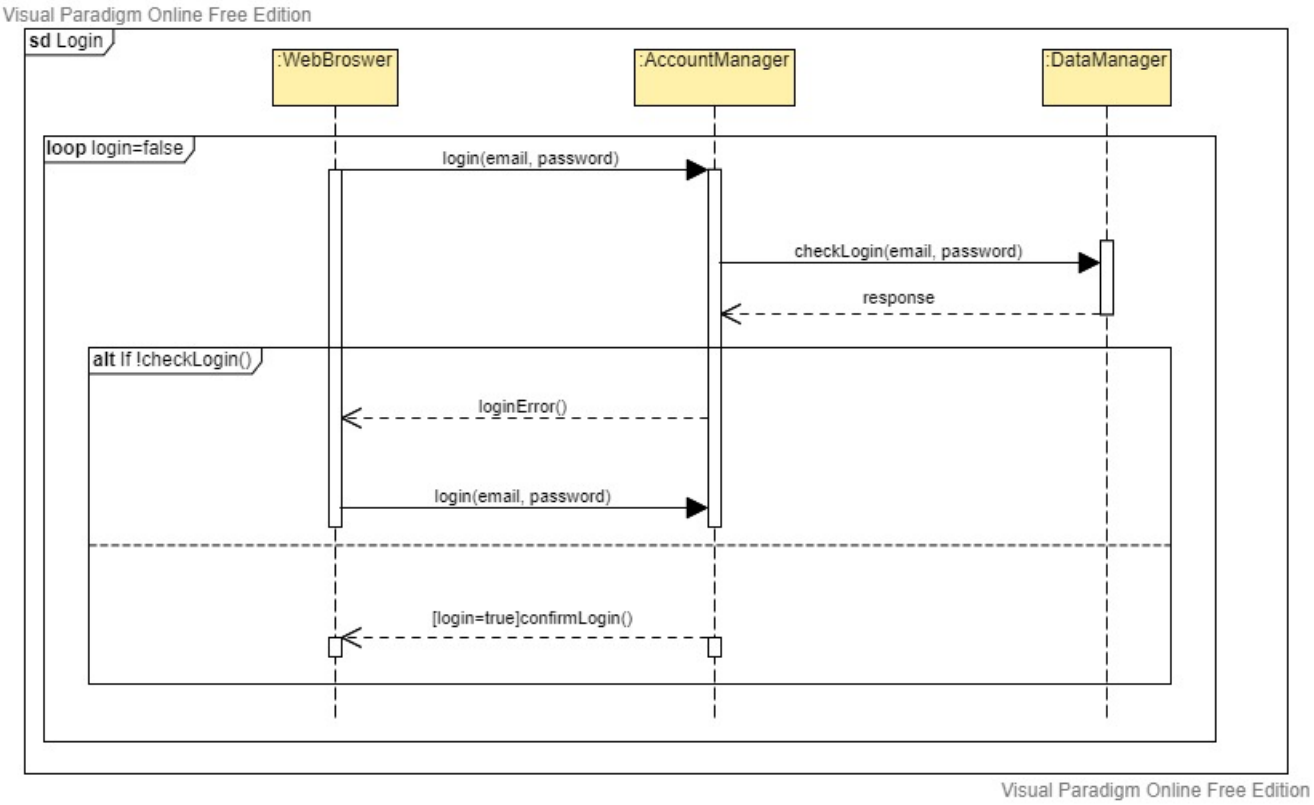

Figure 6: Login Phase

Login phase is very simple, users put their email (primary and unique key of the database) and password and the sistem check if they're correct. In case of success the user can log in the system and use its functionalities. Every user will login the system accordingly the role that they had choosen in the Sign Up phase.

View Steering Initiative Reports

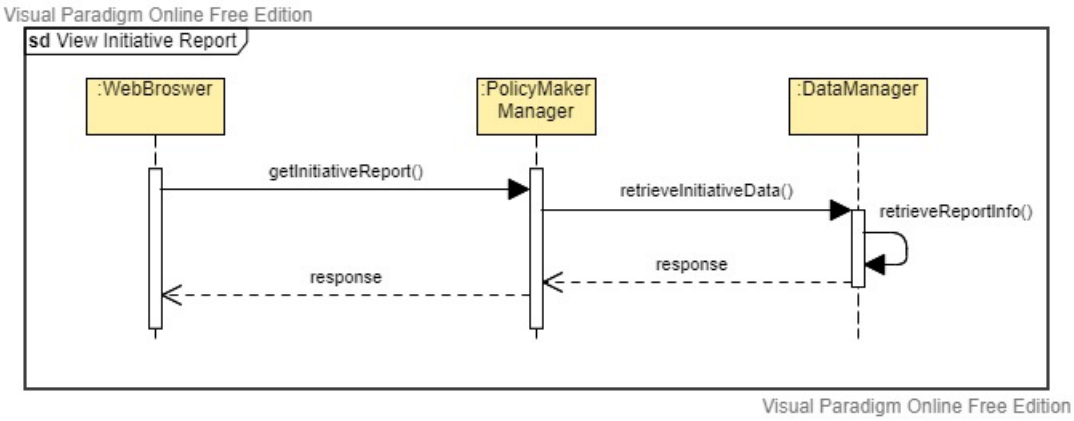

Figure 7: View Initiative reports

This sequence diagram represents the interactions between components in order to display to the policy maker the steering initiative that are stored in the database.

Check Soil Humidity Data

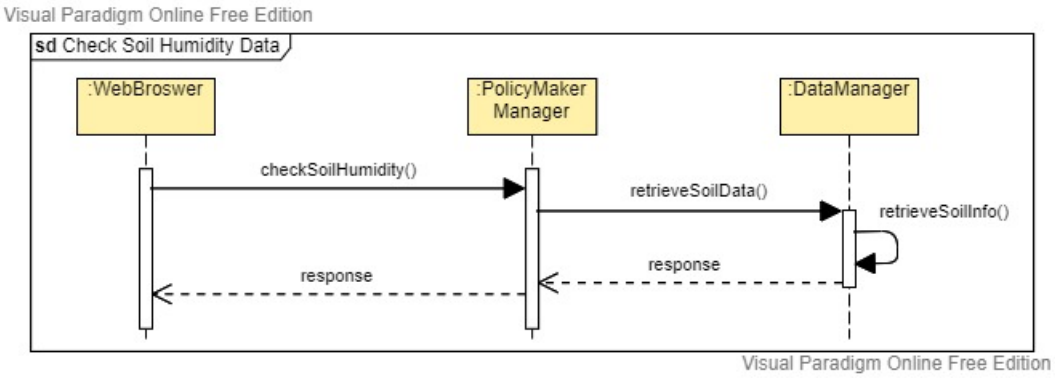

Figure 8: Check Soil Humidity Sensors Data

This sequence diagram represents the interactions between components in order to display to the policy maker the sensors data regarding the soil humidity. These data are taken, by a """""""method""""""" in the application server, from an external source and they're put in the system database in order to manage them in a easiesr way.

Check Water Irrigation Data

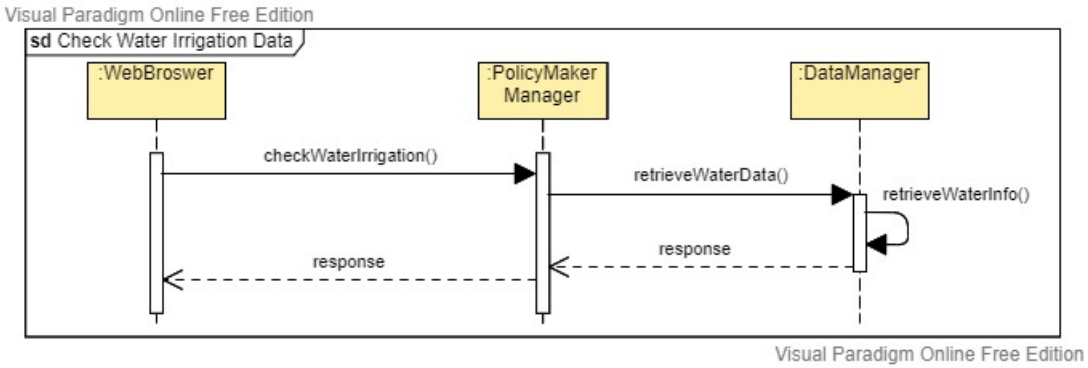

Figure 9: Check Water Irrigation Sensors Data

This sequence diagram represents the interactions between components in order to display to the policy maker the sensors data regarding the water irrigation. The function is the same that the case above.

View Farmers Ranking

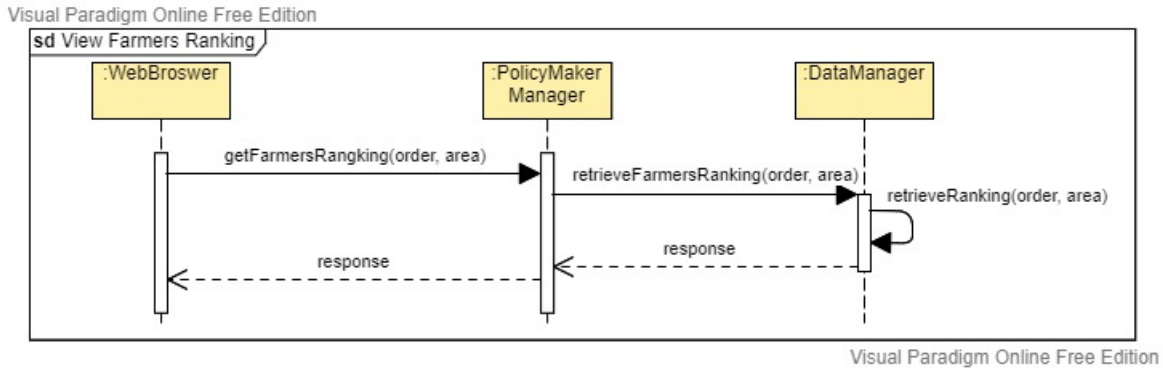

Figure 10: View Farmers Ranking

This sequence diagram represents the interactions between components in order to display to policy makers and agronomists the ranking farmers ranking. It will be explained better later in the document. The function includes a parameter called 'area' in order to specify which area the user wants the rank. In order to have code reusability this function can be used by either policy makers or agronomists. When a policy maker wants to see the rank the parameter 'area' will be set to 'all' in order to obtain the entire farmers rank. Instead, when an agronomist wants to use it 'area' will be automatically taken from its responsibility area.

View Specific Farmers Informations

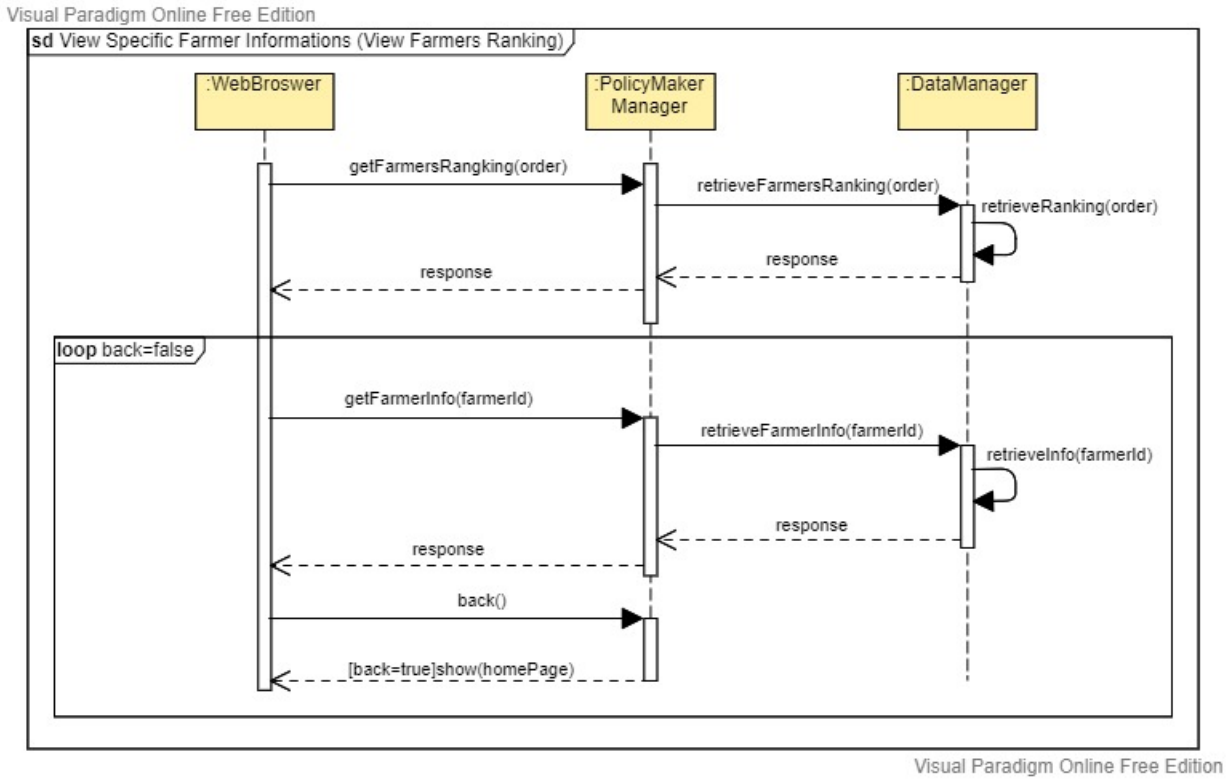

Figure 11: View Specific Farmer Informations

This sequence diagram represents the interactions between components to search for the informations of specific farmer information.

Profile Edit

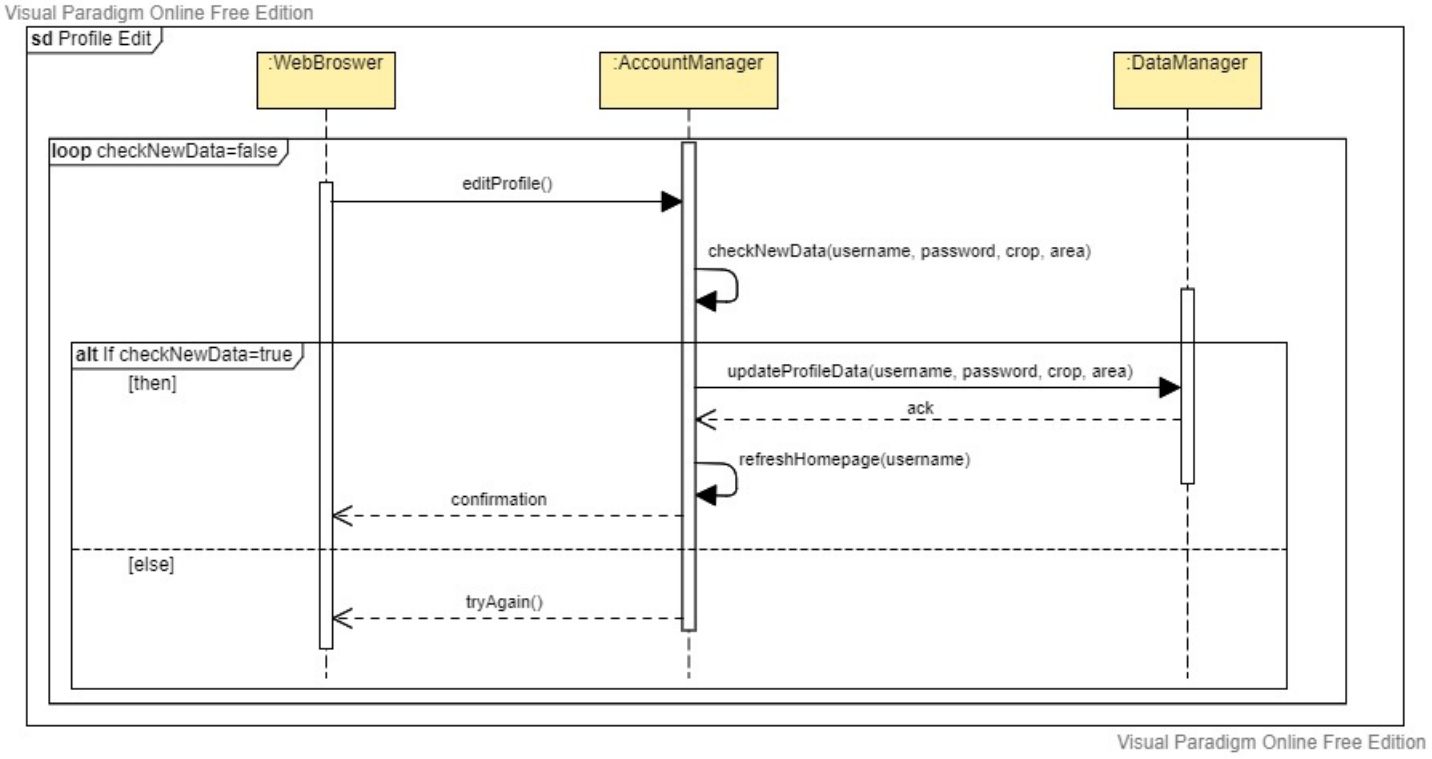

Figure 12: Edit Profile

This sequence diagram represents the interactions between components to change users profile settings like username, password, area or crop type if it's a farmer.

Insert Production Data

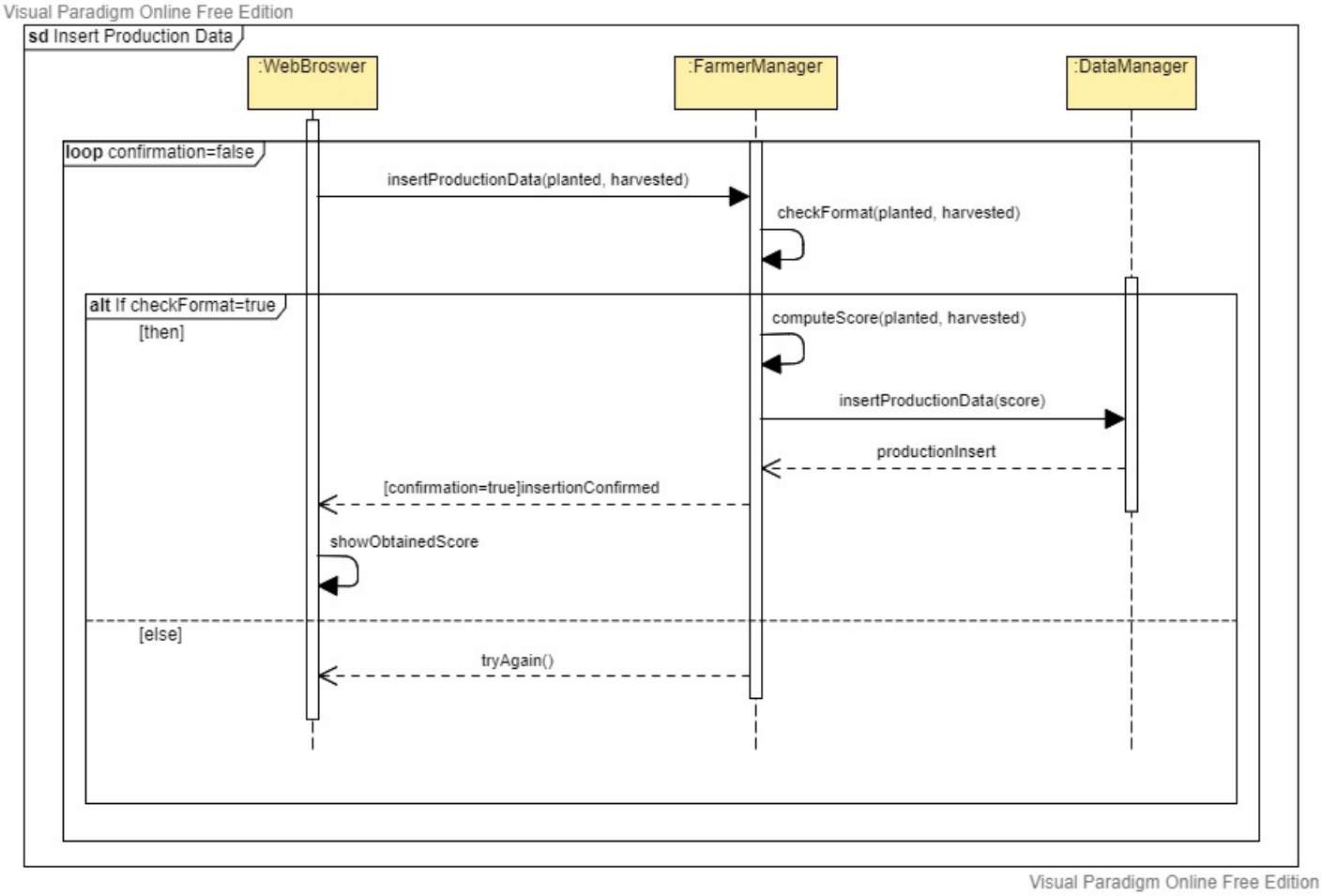

Figure 13: Insert Production Data

This sequence diagram represents the interactions between components in order to insert farmers production data. During this phase the rank is calculated in order to store it as an attribute of the farmer that insert the data.

Check News

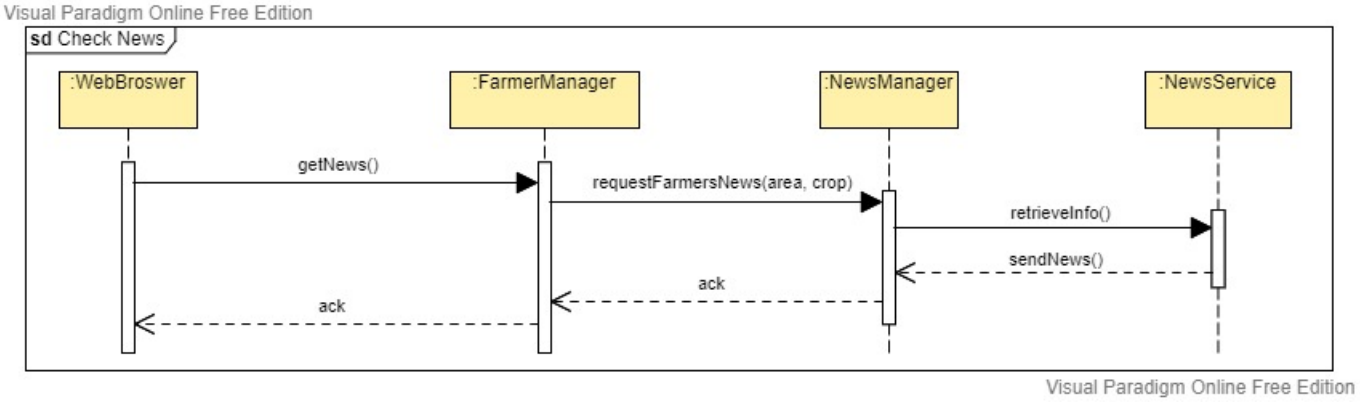

Figure 14: Check News

This sequence diagram represents the interactions between components in order to displays to the farmers the news regarding their interests, such as area or crop type. It uses the NewsService component in order to get always up to date news and all it's managed by the NewsManager component.

Forum

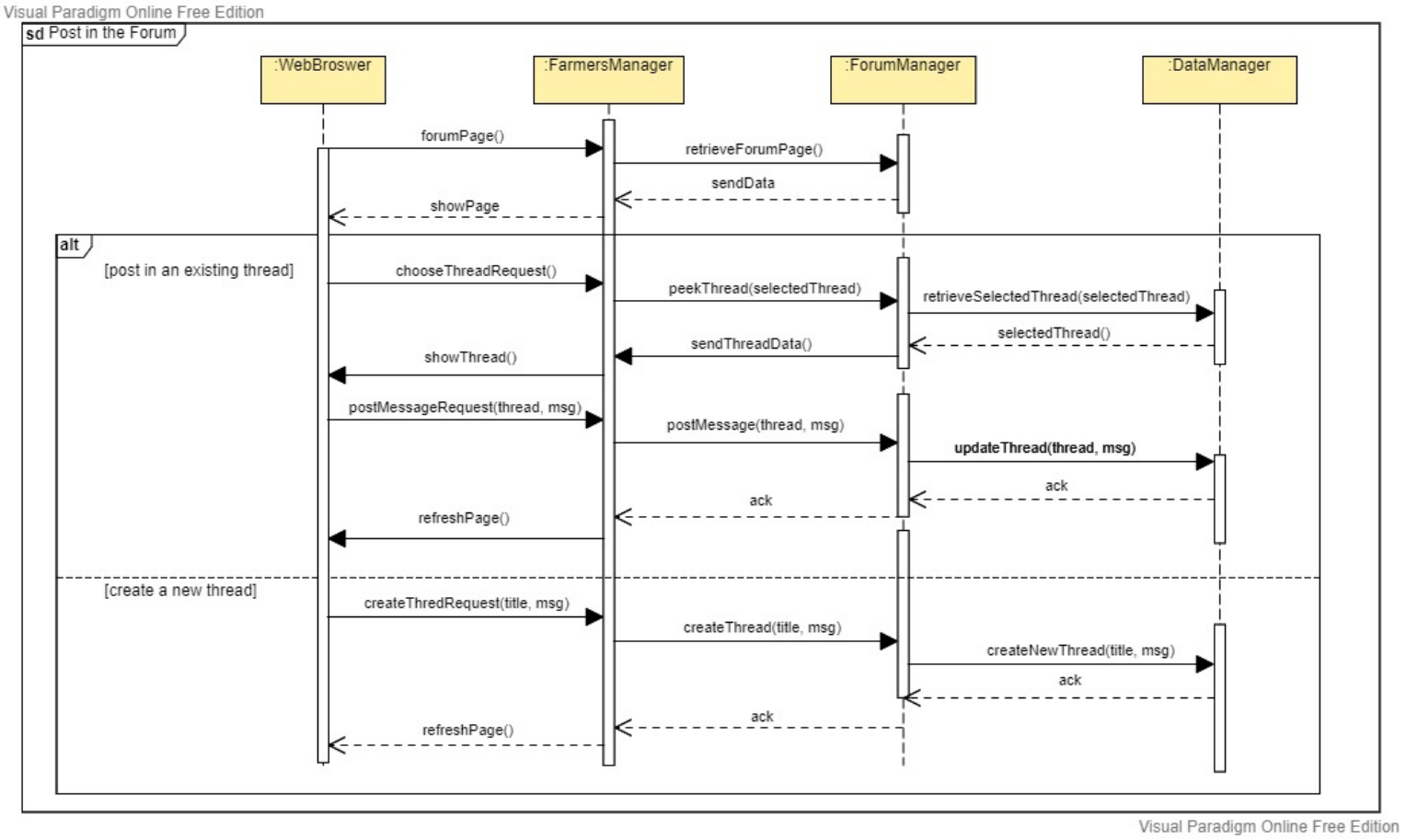

Figure 15: Interact with Forum

This sequence diagram represents the interactions between the components composing the forum. It is split into two part the one that allows the user to make a post in an already existing thread and the one that allows the user to create a new thread.

Check Weathar Forecast

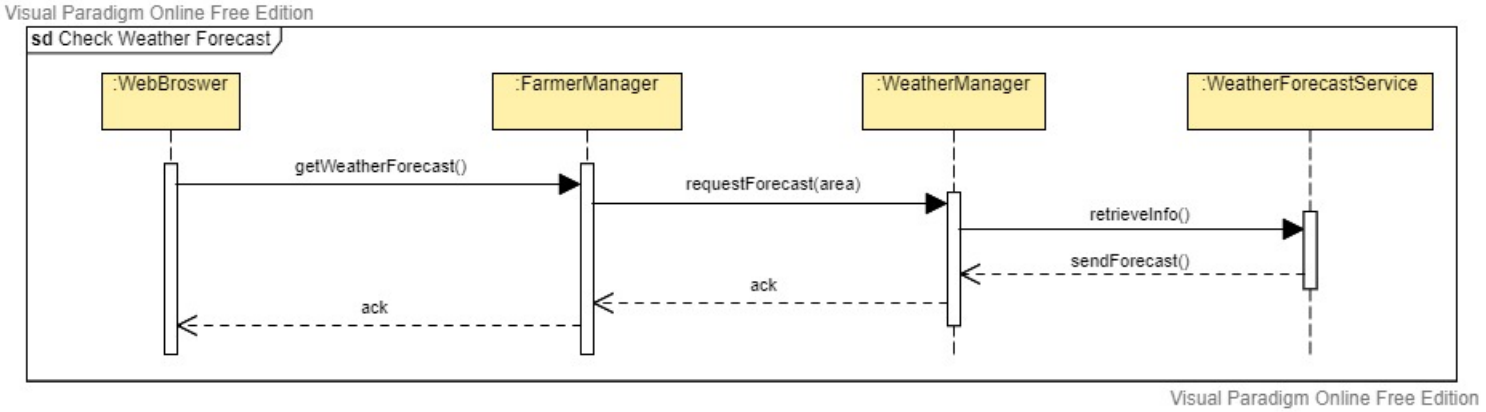

Figure 16: Check the Weather Forecasat

This sequence diagram represents the interactions between components in order to displays to the users the weather forecast in relation to their area. It uses the WeatherForecastService component in order to get always up to date forecast and all it's managed by the WeatherManager component.

Help Request Interaction

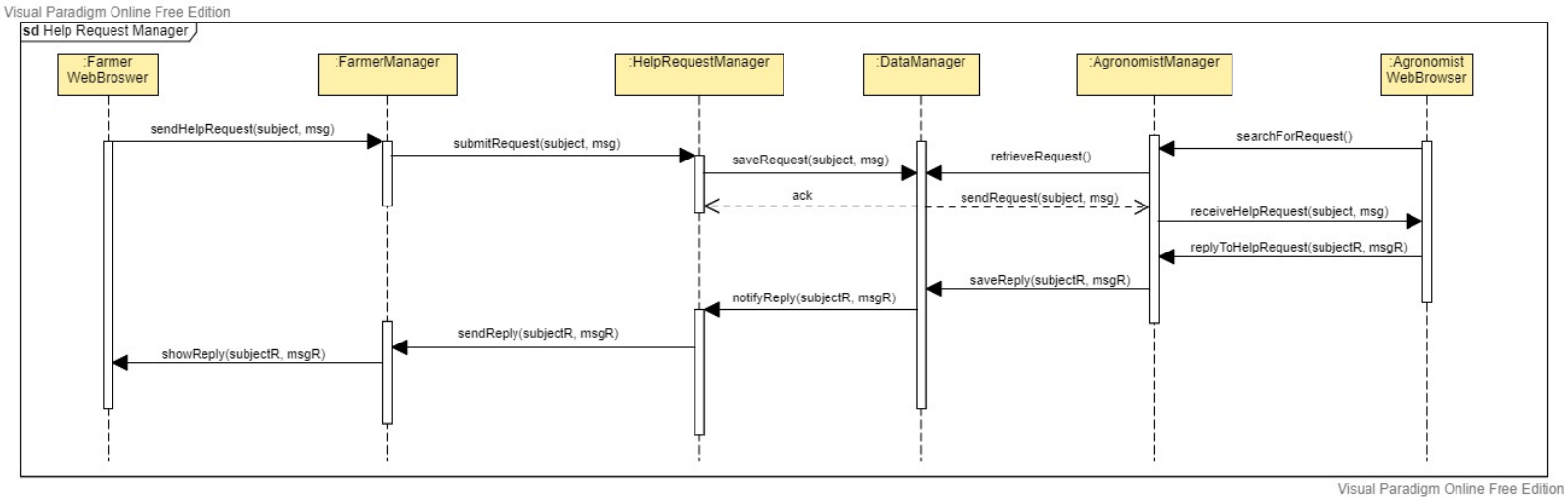

Figure 17: Help Request Interactions

This sequence diagram represents the interactions between the components needed to manage the help request that the farmers send to the agronomists. When a farmer send an help request the system save it in the database. When the agronomist is online an automatic script checks if there are some unread messages in the database and sends them to the agronomist. The agronomist replys to the request and saves it into the database. When the faremer is online another automatic script checks if there are some unread reply in the database and displays them to the farmer. This method works also if one of the two actor is offline because the saving of the messages in the database.

Daily Plan Management

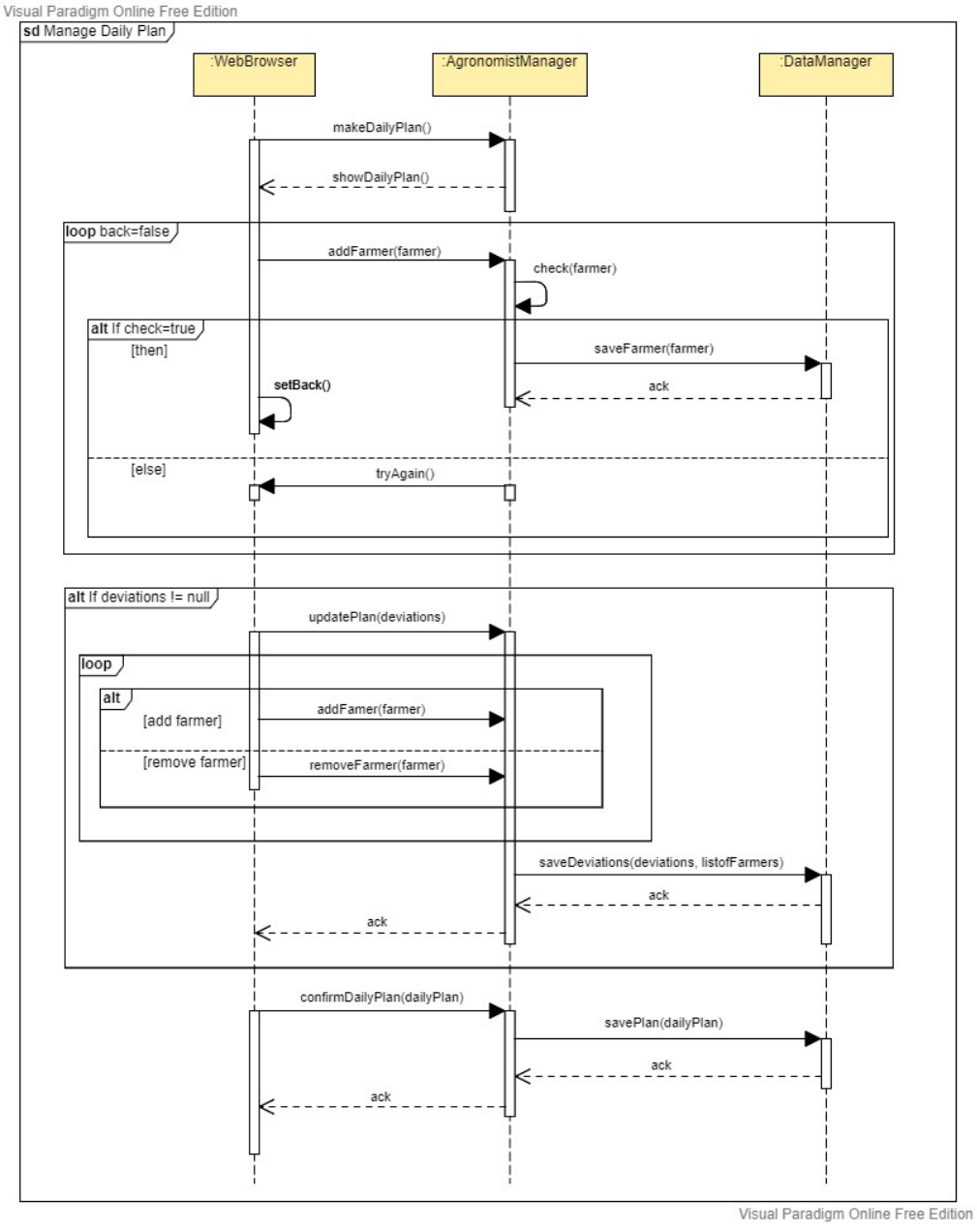

Figure 18: Daily Plan Interactions

This sequence diagram represents the interactions between components in order to manage the daily plan. In the diagram are present either the creation and the updating of the plan.

2.5 Component interfaces

2.6 Selected architectural styles and patterns

2.7 Other design decisions

3 User Interface Design

4 Requirements Traceability

5 Implementation, Integration and Test Plan

6 Effort Spent

7 References