AY 2021/2022

POLITECNICO
MILANO 1863

DD: Design Document

Stefano Brunati: 10623921

Edoardo Cappelletti: 10622565

Gabriele Curti: 10624502

Professor

Elisabetta Di Nitto

Contents

1 Introduction

In modern India's economy agriculture plays a pivotal role. More than 58% of rural households depend on jobs in the agricultural sphere as the principal means of livelihood. Moreover, 80% of these households are smallholder farmers with less than 2 hectares of farmland, from which more than a fifth is below poverty.

World population keeps growing. It's estimated that there will be 9.7 billion people by 2050. Food demands are expected to grow anywhere between 59% to 98%.

Climate change effects are predicted to impact everything across the food and farm systems, from productivity to livelihoods, predicting a 4% to 26% loss of net income for farmers by the end of the century.

Covid-19 pandemic has greatly highlighted the vulnerabilities and fragility of our food supply chains, telling us that we need a more resilient food system and that we can't forget about marginalized communities or smallholder farmers. All these reasons call for a revamp of the entire mechanism that brings food from farms to our plates.

It's more important, now than ever, that we develop and adopt innovative methodologies and technologies that can help bolster countries against food supply challenges and shocks.

In this contest we propose our solution for the Telangana food system, the 11th largest state in India with a geographical area of 112 077 km2 and 35 193 978 residents.

1.1 Purpose

Our goal will be to design and develop a community-centric system that will support the agricultural community via a data-driven approach, bolstering both production and welfare of the farmer population.

Stakeholders of this project will be of three main categories:

- Farmers in the Telangana region. They will be aided in their work from the data that will be available to them. By accessing weather forecasts and critical information when necessary they will be able to both ease their work and get more in return.
- Agronomist involved in aiding farmers in the Telangana region. They will be aided in the organization of their daily visits and in responding to help requests, permitting more mirated and specific work on needing farmers.
- Policy makers in the Telangana region. By seeing specific performance data they will be able to check the results of the rule they applied, and through a direct connection to the farmers they will be able to quickly publish new advice and rules.

1.1.1 Phenomena

User login	World Shared
User registration	World Shared
Check username and password	Machine
Visualize weather forecasts	World Shared
Visualize personalized suggestions	World Shared
Insert data about production (and problems)	World Shared
Request for help and suggestions by agronomists and other farmers	World Shared
Get notification for help answers	Machine Shared
Respond to a request for suggestions and help	World Shared
Create discussion forums with other farmers	World Shared
Read a discussion forum	World Shared
Respond in a discussion forum	World Shared
Get notification from forum answers	Machine Shared
Receive requests of best practises	Machine Shared
Send best practises to policy makers	World Shared
Work on the crops	World
Get notification for new blog post	Machine Shared
Read blog post	World Shared
Receive incentive notification	Machine Shared
Choose responsibility area	World Shared
Receive help requests from farmers	Machine Shared
Respond with suggestions to farmers	World Shared
Visualize weather forecast in the area	World Shared
Visualize farmer performance data	World Shared
Visualize daily visit plan	World Shared
Modify daily visit plan (before the confirmation)	World Shared
Confirm the execution of the plan	World Shared
Specify deviations from the plan	World Shared
Visit farmers	World
Visualize farmers performance data	World Shared
Request best practices to the “resilient” farmers	World Shared
Receive best practices	Machine Shared
Publish best practice on a blog	World Shared
Decide and send special incentives	World Shared
Visualize crops performance data	World Shared

Table 1: Phenomena

1.1.2 Goals

- G1: Increase the overall welfare and production of the Telangana region. By facilitating the communication and the collaboration between farmers, policy makers, and agronomists the aim is to increase the wellbeing of farmers inhabiting the Telangana region.
- G2: Aid policy makers in the decisional process. Policy makers can see production data in order to decide the incentives for farmers, or whether the current policies are performing well or should be changed (in order to constantly improve Telangana's production).
- G3: Aid the farmers in the management of their productions. Farmers will receive personalized suggestions and best practices, and they will also have the possibilities to ask for help to both other farmers (by lending/renting equipment or giving advice) or to agronomists.
- G4: Aid agronomist works to help farmers and check crops production. Creating and modifying a daily plan will help them organize their visits and maximize their help in a well-specified zone of expertise.

1.2 Definitions, Acronyms, Abbreviations

1.2.1 Definitions

- Farmer: a person who cultivates crops
- Resilient farmer: a farmer whose production is good despite meteorological adverse events.
- Agronomist: an expert in the science of soil management and crop production.
- Policy maker: a person in charge of formulating policies, related to the food system.
- Production: total crops-output generated. Could be related to a single farmer, a zone or the entire Telangana's state.
- Personalized suggestions: indication directly focused on a specific farmer, such as specific crops to plant or specific fertilizers to use based on location and type of production.
- Welfare: overall well-being of farmers which translates into the reduction of poverty and simplification of work (discussion with other farmers, suggestions, personalized data based on location).
- Best practices: cultivation procedure that has been shown by experience to produce optimal results (not only in terms of achieved final production but also in terms of resilience to adversities) and that should be proposed for widespread adoption.
- Responsibility area: zone of which an agronomist is in charge of, with the purpose of increasing its welfare and production.
- Visit: it refers to the agronomist going to a specific farm of his competence, and is identified by a date, a variable time-slot (deviations may occur) and a reason.
- Notification: alert that a certain event has occurred. Could be an email or an automated message sent to the smartphone when the app is not running.

1.3 Revision History

January x, 2022: version 1.0 (first release)

1.4 Reference Documents

- Specification document: "RDD Assignment A.Y. 2021-2022"
- Course slides
- UML official specification https://www.omg.org/spec/UML/

1.5 Document Structure

- Chapter 1: Introduction. This section provides an overall description of the system scope and purpose, together with some information about this document.
- Chapter 2: Architectural Design. This section is addressed to the developer team and offers a more detailed description of the architecture of the system. The first part describes the chosen paradigm and the overall split of the system into several layers. Furthermore, a high-level description of the system is provided, together with a presentation of the modules composing its nodes.
- Chapter 3: User Interface Design. This section is useful for graphical designers of the S2B and contains several mockups of the application, together with some charts useful to understand the correct flow of execution of it.
- Chapter 4: Requirements Traceability. This section acts as a bridge between the RASD and DD document, providing a complete mapping of the requirements and goals described in the RASD to the logical modules presented in this document.
- Section 5: Implementation, Integration and Test Plan. The last section is addressed to the developer team and describes the procedures followed for implementing, testing and integrating the components of our S2B. There will be a detailed description of the core functionalities of it, together with a complete report about how to implement and test them.

2 Architectural Design

2.1 Overview

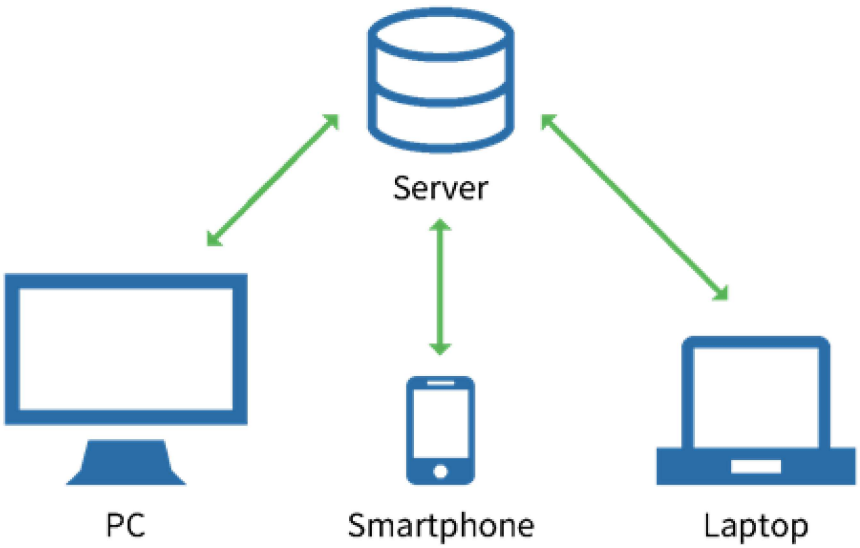

Figure 1: Client-Server architecture

The system is a distributed application which follows the well known client-server paradigm. It implements the thin-client technique, so as to facilitate supporting different client platforms.

There are two main types of clients, the first will be the Web Application, and the second will be the Mobile Application. All the system business logic and data management will be contained and executed in the server.

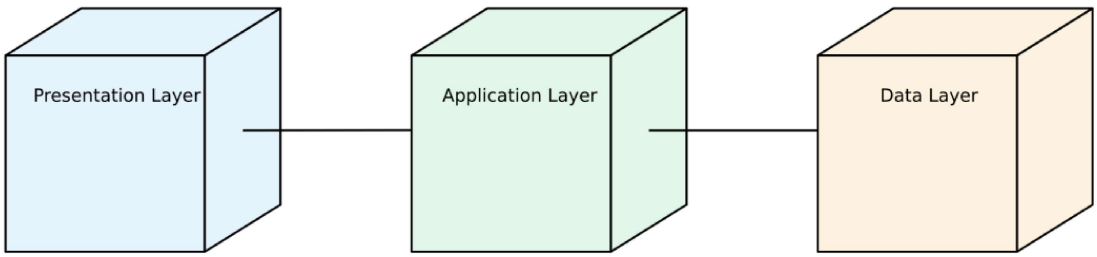

Figure 2: Three-Layers architecture

The system has three layers:

- *Presentation layer*: manages the presentation logic and the user interaction.
- *Application layer*: manages the business logic and functions that the system must provide.
- *Data layer*: manages the storage and retrieval of data.

These layers will be physically separated in the system, and will communicate through known APIs.

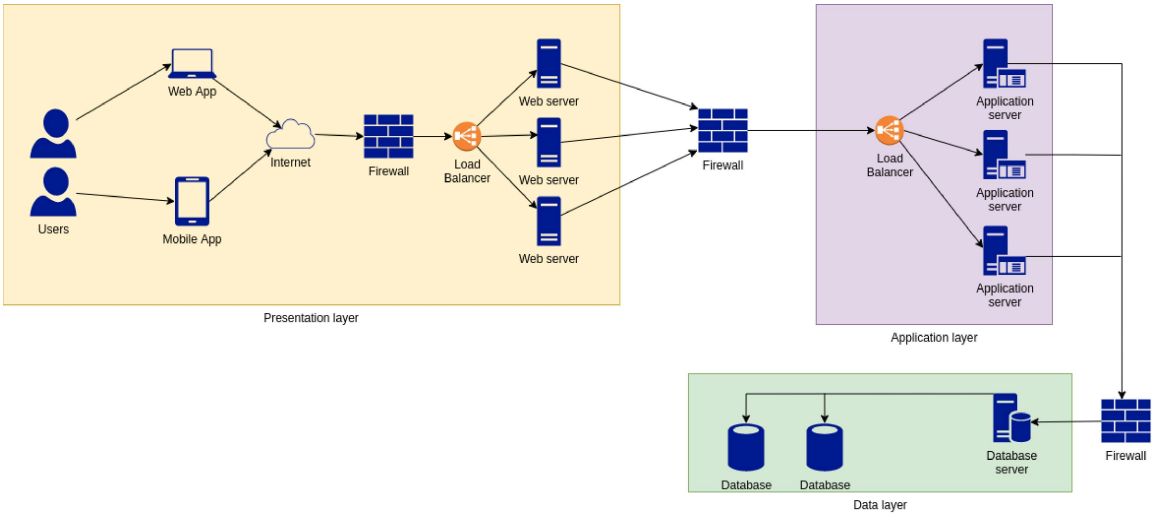

Figure 3: Architecture

The system also has four tiers:

- *Client*: is either a web browser or a smartphone and will render the user interface.
- *Web Server*: manages the incoming connections, prepares the user web pages and communicates requests to the application server.
- *Application Server*: implements the business logic.
- *Data Server*: implements the data storage and retrieval.

The first two tiers are used for the presentation layer, while the 3rd tier is for the application layer and the 4th tier is for the data layer.

The client will communicate with the web servers, which will act as middleware and will interface with the application layer through its APIs. The application layer will then communicate to the data layer using the DBMS APIs.

The application server APIs are RESTful, to better ease the portability of the client code and to guarantee flexibility for scaling. For communicating with the data layer, the ORM technique is used to exploit the object-oriented paradigm to also access the relational database.

Every physical layer will be separated through a firewall and every communication will be encrypted using the HTTPS standard.

The system component will now be described in more depth.

2.2 Component view

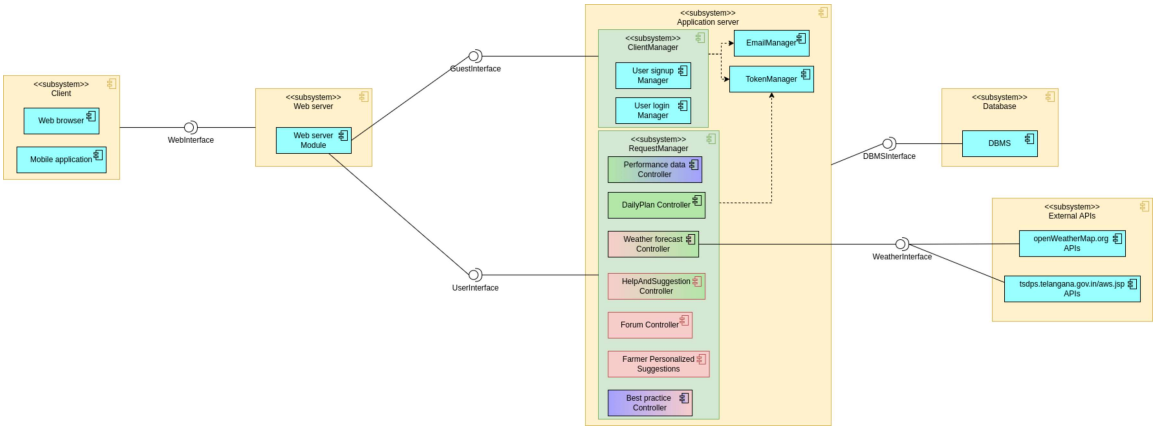

Figure 4: Component View

In the above figure we can see in more detail the components that are part of our system.

- **Web Server**: it has the function of routing requests from the clients to the application server and prepares part of the presentation.
- **Client manager**: it handles all the requests from unauthenticated users, giving them the possibilities of signup or authenticating them. It uses the login interface to manage all the requests and is capable of releasing authentication tokens that will be used for service with the other RESTful APIs. The token will also identify a request for its type of user, if it's a farmer, agronomist or policy maker.
- **Request manager**: it checks a request token and route each request to the correct controller. Each controller has all the functions needed to support its feature. They are color coded to express which user type they must serve, green for agronomists, red for farmers and blue for policy makers.
- **Email manager**: it handles the management of email, with built in components for scheduling or sending.
- **Token manager**: it manages, creates, checks and deletes authentication token, used for authenticating the REST requests.
- **DBMS**: it is the relational database controller that manages the storage and retrieval of data, but also the replication and security of such.
- **External APIs**: they represent the APIs of external services, for retrieving weather data.

2.3 Deployment view

Figure 5: Deployment View

In the above figure is shown the deployment diagram, that shows the needed components for the system to run correctly. Each of these devices has its own operating system where software components run. Copies of these devices can be added to scale out the system.

The tiers are four:

- **Client tier**: these are the client machines, which will run either a browser or the mobile application. To extend the audience the support will be to all major browsers (running on all major os) and mobile operating systems.
- **Web server tier**: this tier includes replicated web servers which receive requests from the clients and routes them to the API of the application server, preparing an HTML page to be rendered on the client, using also client-side scripting and style sheets.
- **Application server tier**: this tier contains replicated application servers which will serve the requests coming from the web servers. It implements all the business logic. To store and retrieve data it will communicate to the Data server tier through the DBMS interface.
- **Data server tier**: this tier includes the machine that will execute the DBMS and the data storage devices. It will execute the requests coming from the application server, storing data securely and with backups options.

2.4 Runtime view

2.4.1 User Sign Up

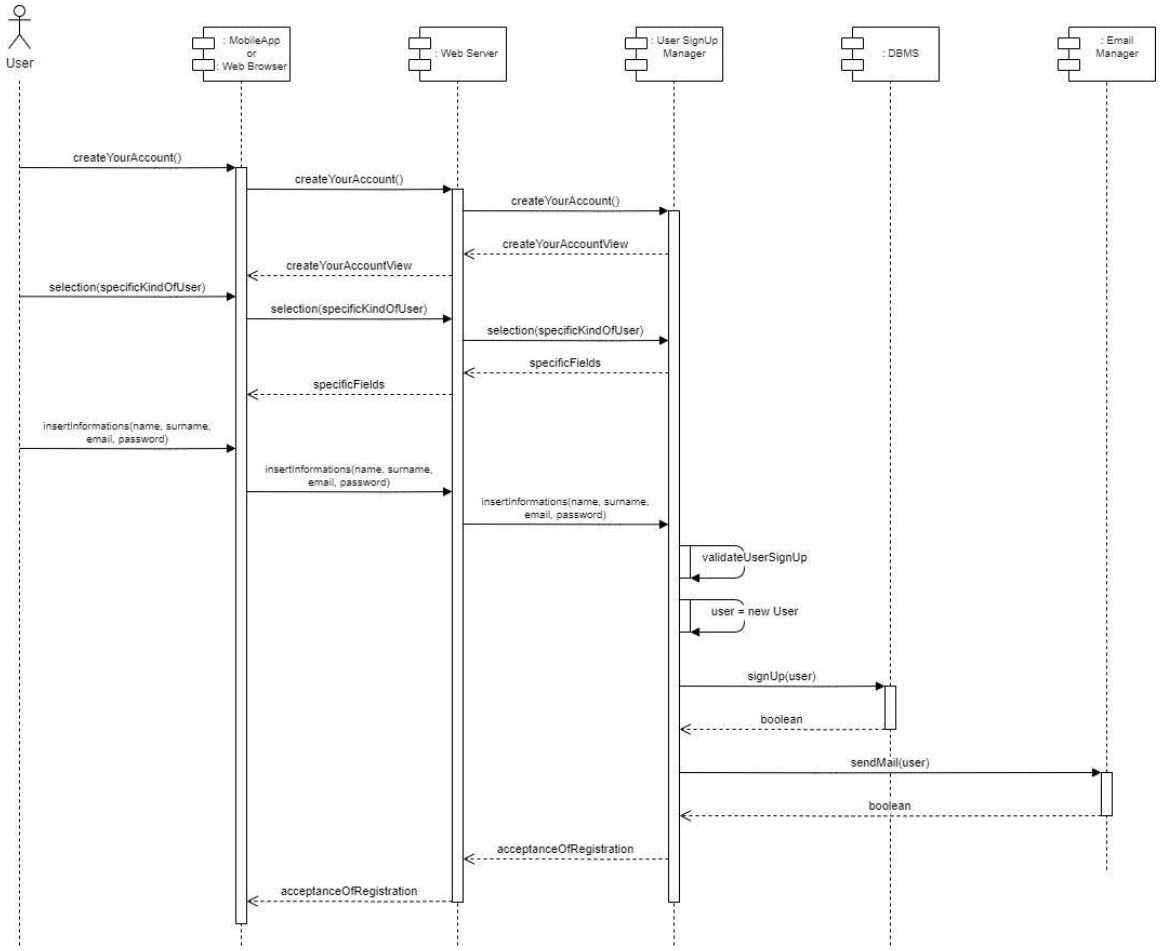

Figure 6: User Sign Up

The diagram above represents the process of signing up a user (Farmer, Agronomist, or Policy Maker).

The three cases are considered together in order to avoid useless repetitions, since the only things that changes are the "specific fields" requested by the system to the user (which are the information inserted lately by the user), according to the selection made by the user itself between the three possible alternatives presented in "createYourAccountView", which are: Farmer, Agronomist, and Policy Maker.

- If the user is a *Farmer*, the specific fields are: name, surname, email, password, farm's name, farm's location.
- If the user is an *Agronomist*, the specific fields are: name, surname, email, password, responsible area.
- If the user is an *Policy Maker*, the specific fields are: name, surname, email, password.

2.4.2 User Login

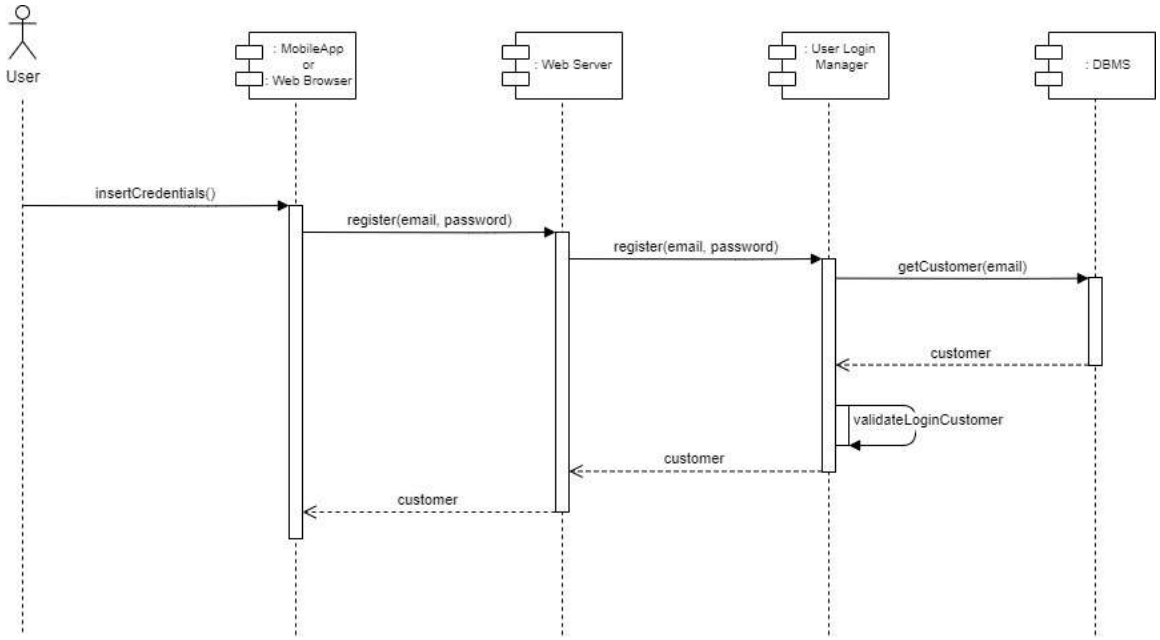

Figure 7: User Login

The diagram above represents the process of login of a user, who can be a Farmer, or an Agronomist, or a Policy Maker.

The user inserts his credentials (email and password), the systems checks if the credentials are correct and then shows the home page according to the user type.

2.4.3 Farmer visualize personalized suggestions

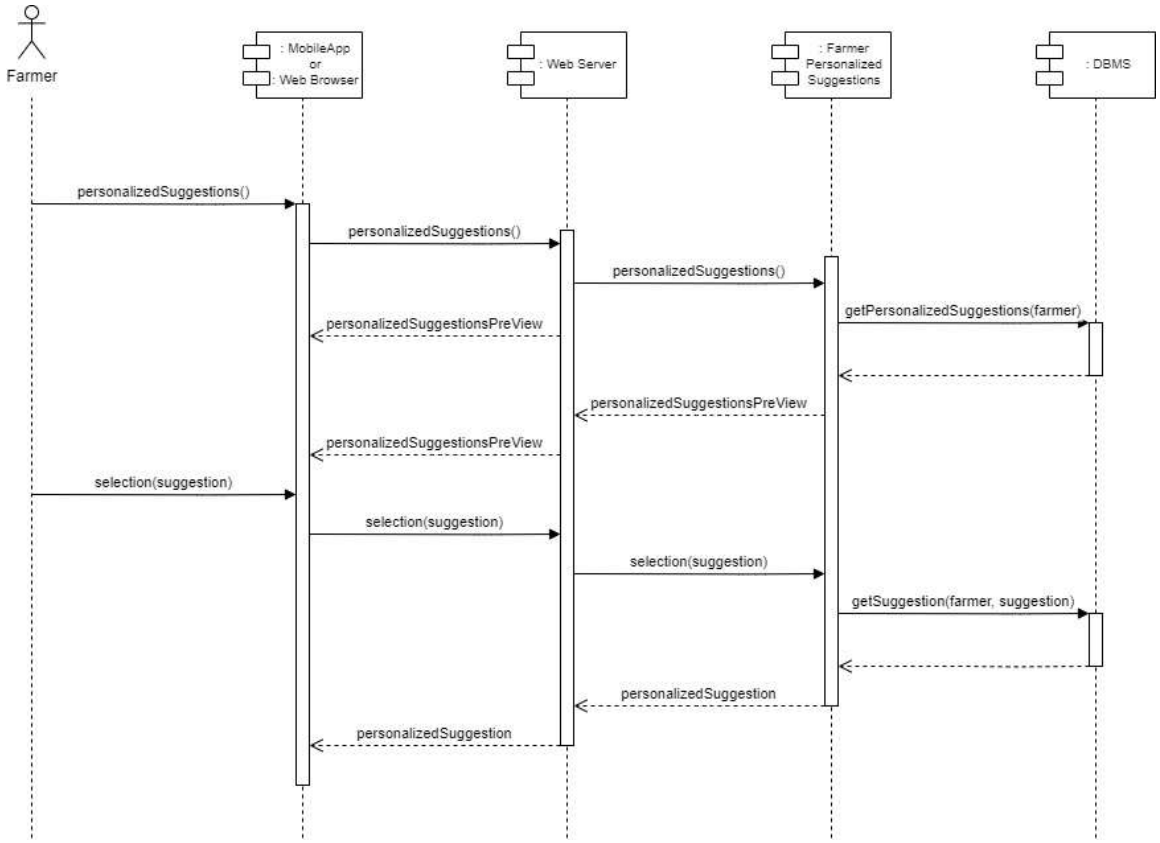

Figure 8: Farmer visualize personalized suggestions

The diagram above represents the process of viewing the personalized suggestions of a farmer, who is already on his home page.

The system displays a preview of all the suggestions, then the farmer selects a specific suggestions through the ones showed, thus the system displays the selected suggestion with all the particulars.

2.4.4 Farmer inserts production data

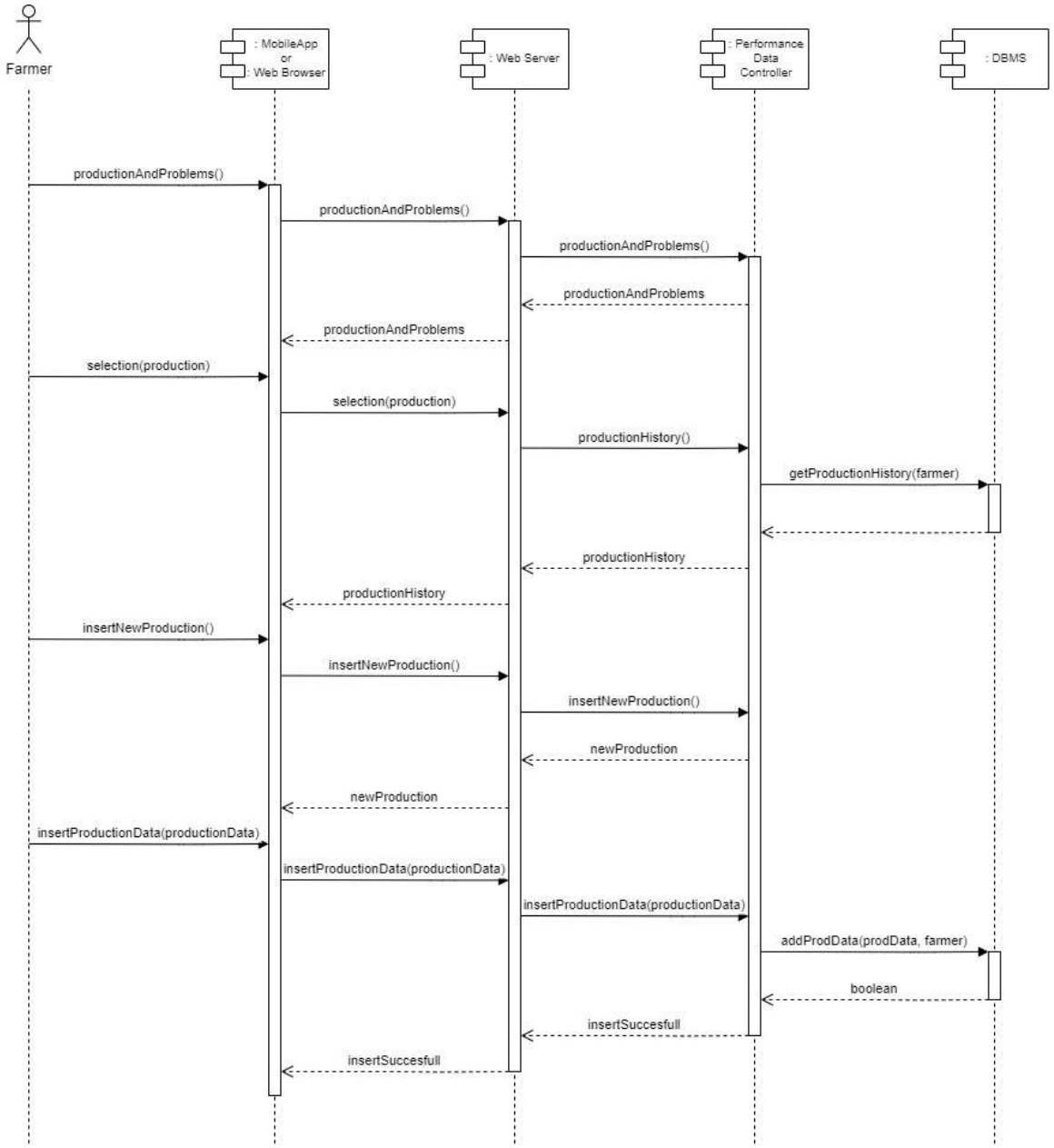

Figure 9: Farmer inserts production data

The diagram above represents the process of inserting into the system the production data by a farmer, who is already on his home page.

The system first asks between inserting production data and reporting a problem, and then, after the farmer selects to insert production data, displays the production data history of the farmer.

The farmer is now able to click on the button "Insert Production Data", and after that the systems shows a list of fields that must be inserted by the farmer to correctly insert the production.

The farmer inserts all the data, so that the system is able to store the production into the DB and, if everything works properly, displays a message saying that the data are stored.

2.4.5 Farmer reports a problem

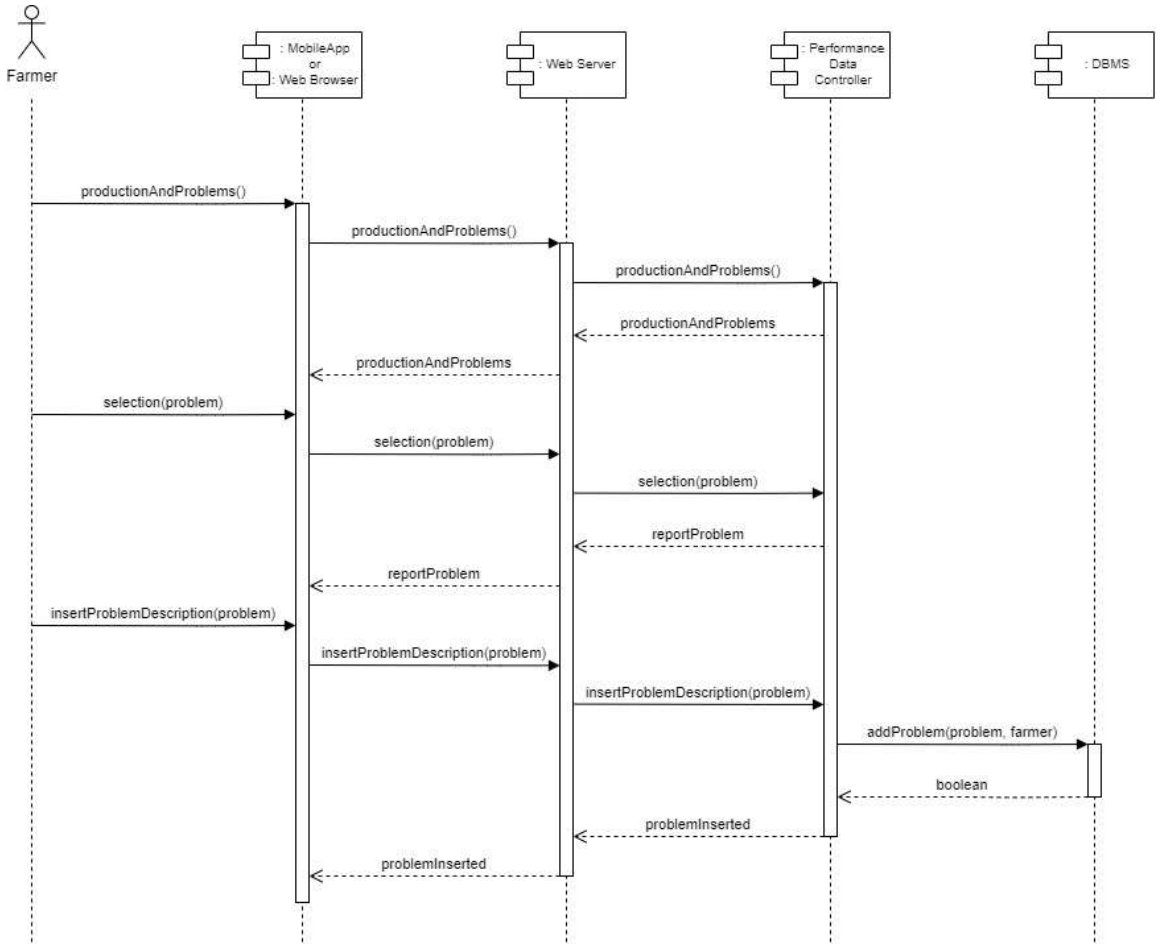

Figure 10: Farmer reports a problem

The diagram above represents the process of reporting a problem by a farmer, who is already on his home page.

The system first asks between inserting production data and reporting a problem, and then, after the farmer selects to report a problem, asks a description of the problem and shows the agronomists responsible of the farmer's area.

The farmer can now insert the description of the problem and selects an agronomist to which request for help.

The system is able to store the problem into the DB and, if everything works properly, displays a message saying that the problem is stored and then the farmer home page.

2.4.6 Farmer requests for help and suggestions

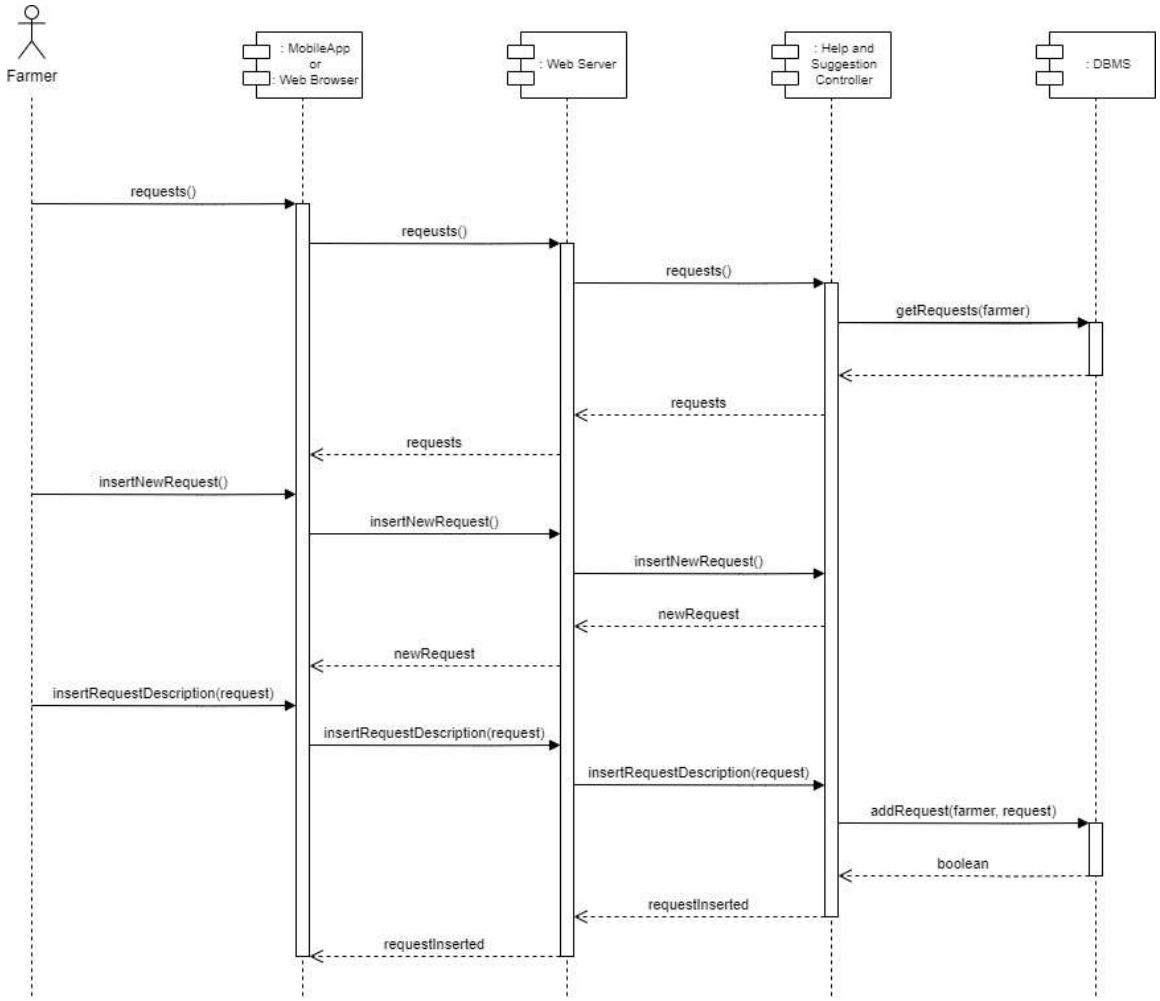

Figure 11: Farmer requests help

The diagram above represents the process of requesting help by a farmer, who is already on his home page.

The system first displays a list of the most recent received, and then, after the farmer selects to make a new request, asks a description of the problem and a preferred receiver (receiver can be either a farmer or an agronomist).

The farmer can now insert the description of the problem and can selects a farmer or an agronomist to which request for help.

The system is able to store the problem into the DB, to contact the selected receiver and, if everything works properly, to display a message saying that the request for help is sent.

2.4.7 Farmer writes a new post on a forum

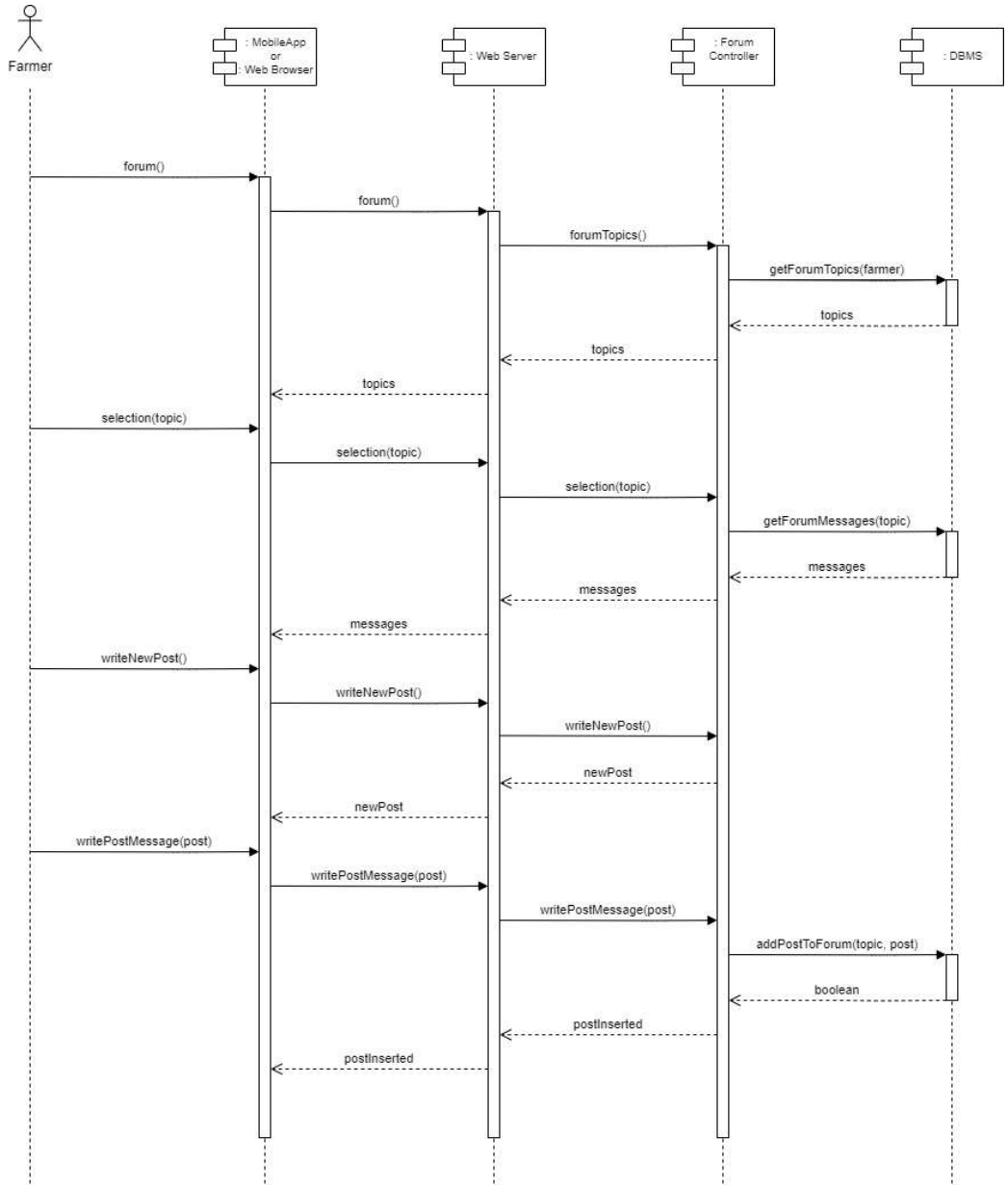

Figure 12: Farmer requests help

The diagram above represents the process of writing a new post on the forum by a farmer, who is already on his home page.

The system first displays a list of all the topics of the forum, and then, after the farmer selects a specific topic, shows the latest messages of the selected topic.

The farmer clicks on the new post button, and then he can write the message; the system store the message into the DB and, if everything works properly, displays a message saying that the post has been created.

2.4.8 Farmer subscribes to a forum's topic

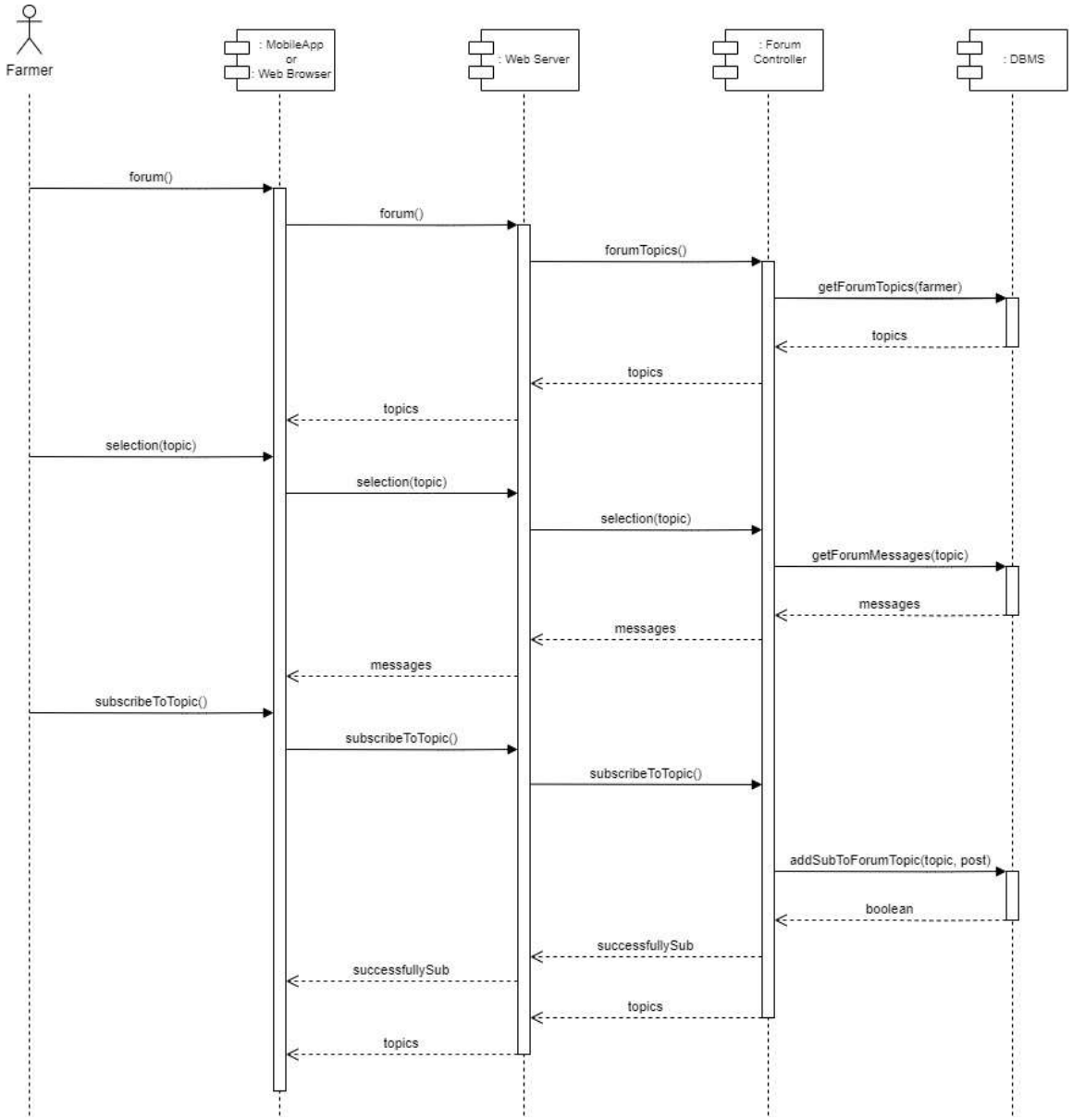

Figure 13: Farmer subscribes to a forum's topic

The diagram above represents the process of subscribing to a topic in the forum by a farmer, who is already on his home page.

The system first displays a list of all the topics of the forum, and then, after the farmer selects a specific topic, shows the latest messages of the selected topic.

The farmer clicks on the subscribe button; the system store the subscription into the DB and, if everything works properly, displays a message saying that the subscription has been successful and then shows the latest messages related to the topic.

2.4.9 Agronomist visualizes a daily plan

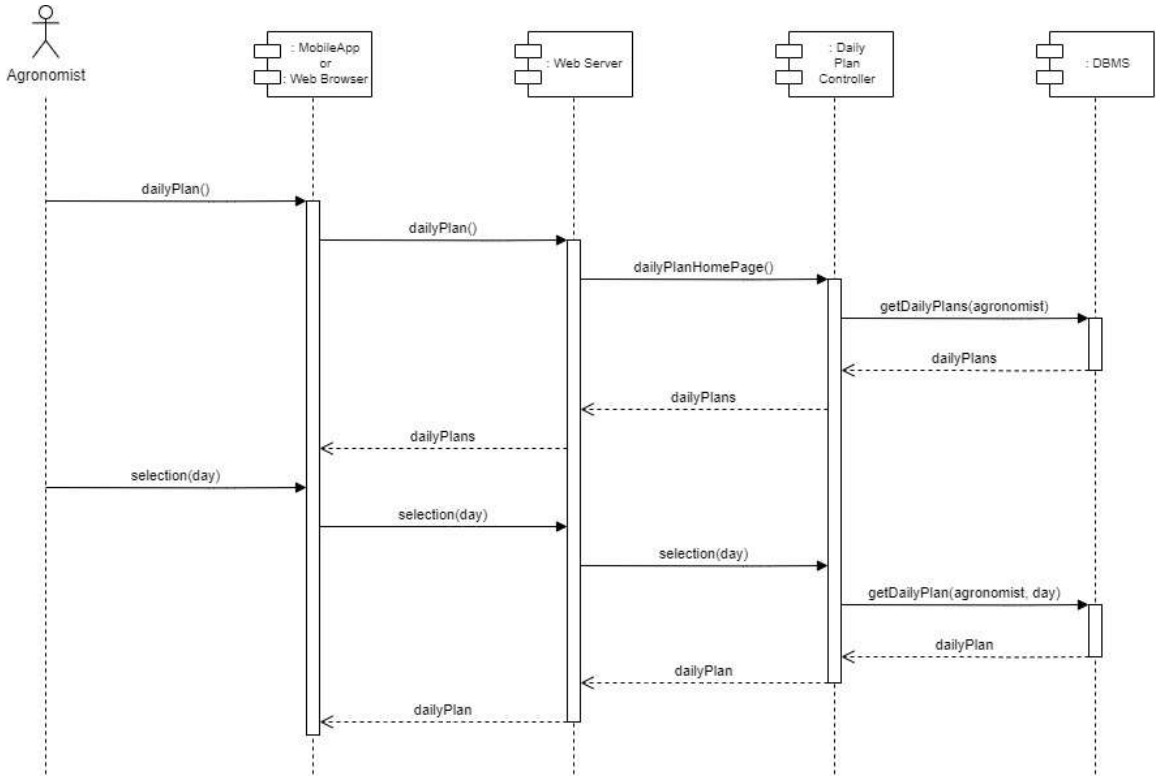

Figure 14: Agronomist visualizes a daily plan

The diagram above represents the process of viewing a daily plan (already inserted) by an agronomist, who is already on his home page.

The system first displays a month calendar with, and then, after the agronomist selects a specific day, shows the daily plan of the selected day.

2.4.10 Agronomist inserts a daily plan

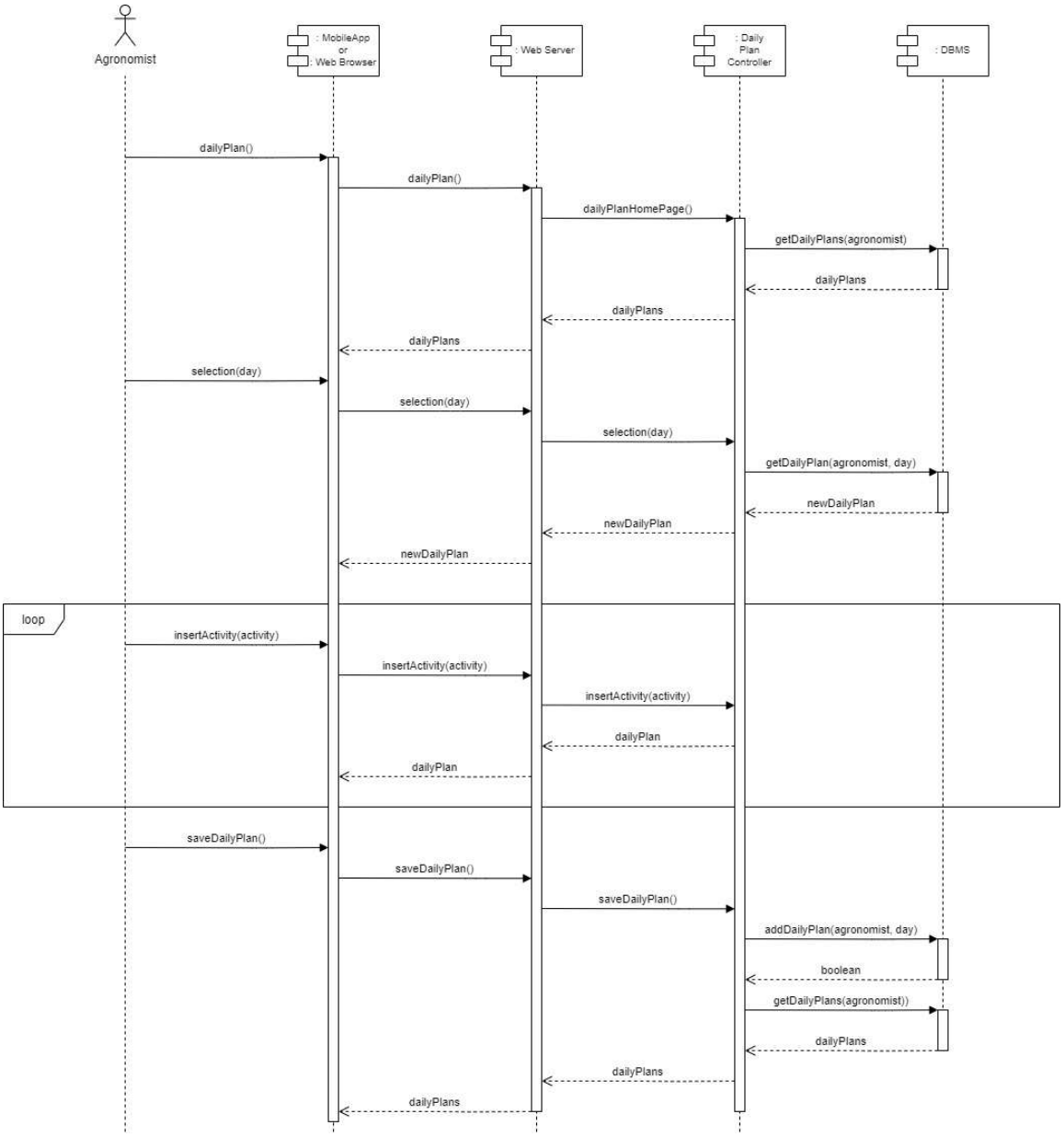

Figure 15: Agronomist visualizes a daily plan

The diagram above represents the process of inserting a daily plan by an agronomist, who is already on his home page.

The system first displays a month calendar, and then, after the agronomist selects a specific day (with no daily plan inserted), allows the agronomist to recursively inserting new activity for the day.

When the agronomist clicks on save button, the system stores the new daily plan into the DB and, if everything works properly, displays a message saying that the daily plan has been successfully inserted and then shows again the calendar view.

2.4.11 Agronomist closes a daily plan

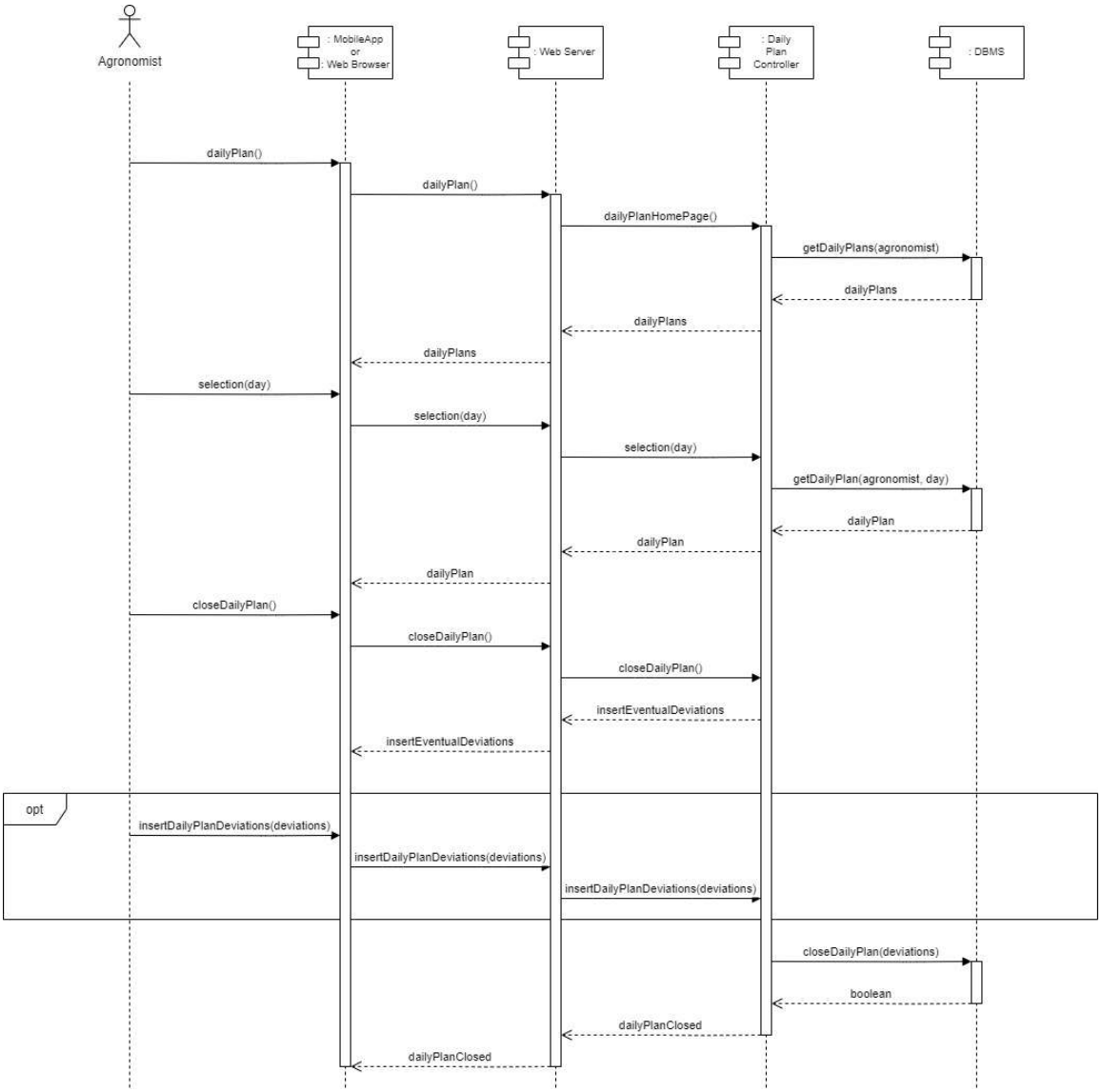

Figure 16: Agronomist confirms the execution of a daily plan (specifying eventually deviations)

The diagram above represents the process of closing a daily plan by an agronomist, who is already on his home page.

The system first displays a month calendar, then, after the agronomist selects a specific day (with a daily plan inserted), shows the daily plan of the selected day.

When the agronomist clicks on confirm and specify deviations button, the system allows to insert optionally some deviations; the agronomist can just click on the confirm button, or can first insert the deviations and then click on the button. The daily plan is closed into the DB and, if everything works properly, displays a message saying that the daily plan has been successfully closed and then shows the agronomist home page.

2.4.12 Policy Maker visualizes farmers' production data

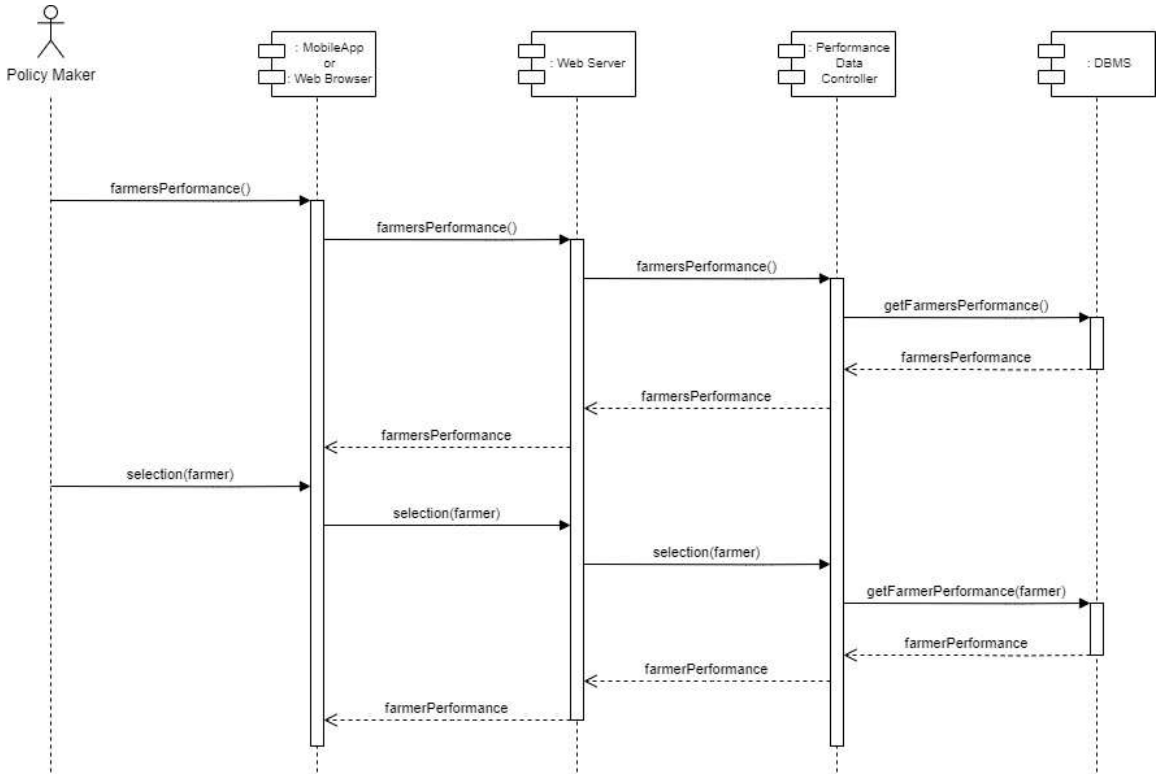

Figure 17: Policy Maker visualizes farmers' production data

The diagram above represents the process of visualizing the production data of farmers by a policy maker, who is already on his home page.

The system first displays a graph with the overall results of the year and a list of farmers ordered by increasing performance, then, after the policy maker clicks on a specific farmer (one on top of the list), shows the production data about the farmer (corn produced, energy, fertilizer, and water used per unit).

2.4.13 Policy Maker asks for best practices

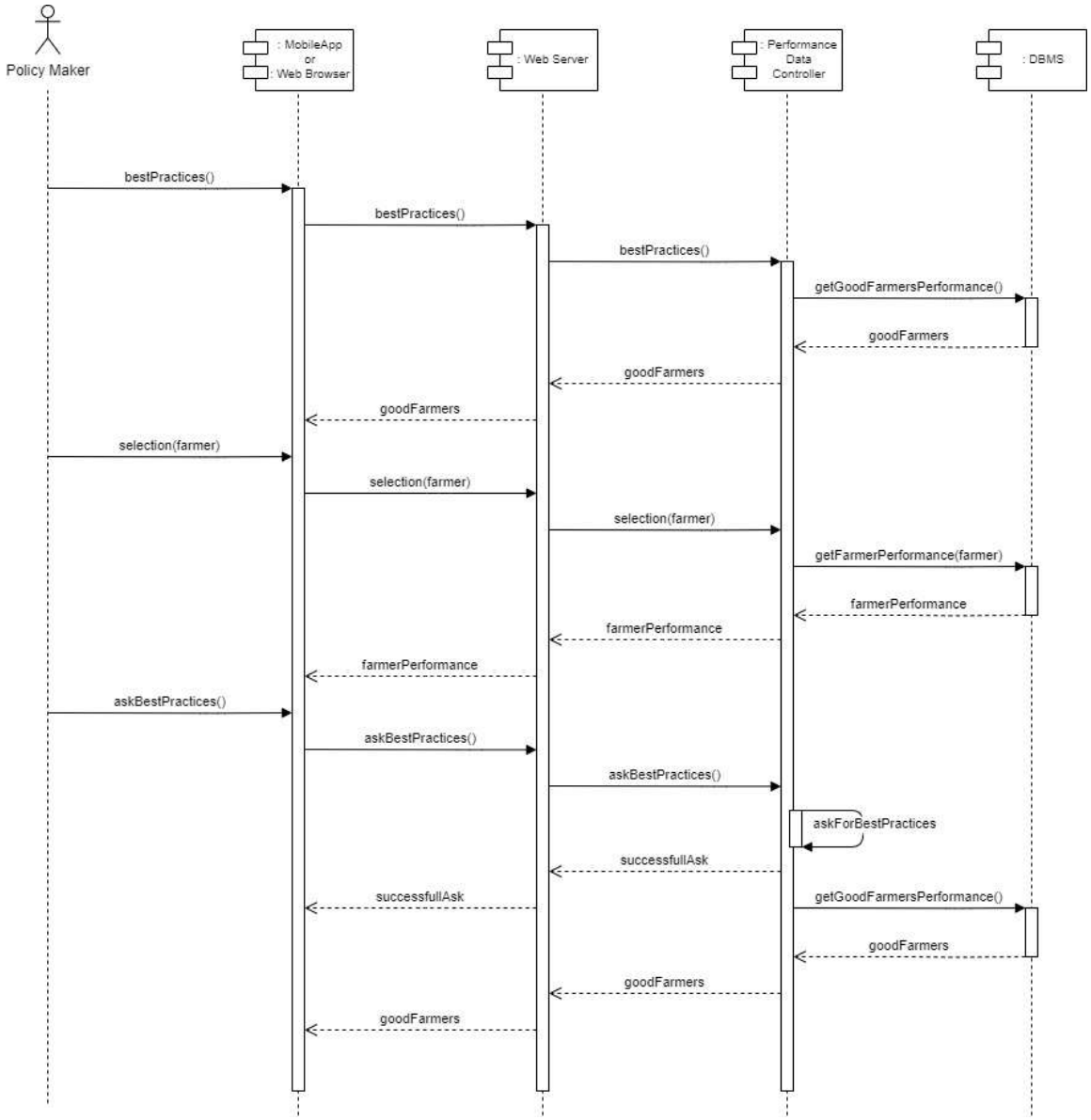

Figure 18: Policy Maker asks for best practice

The diagram above represents the process of visualizing the production data of farmers by a policy maker, who is already on his home page.

The system first displays a graph with the overall results of the year and a list of farmers ordered by increasing performance, then, after the policy maker clicks on a specific farmer (one on top of the list), shows the production data about the farmer.

The policy maker clicks on ask for best practice button, and then the system shows a predefined message to be sent to the selected well performing farmer; when the policy maker clicks on send button, the system send the request to the farmer and shows again the list of farmers ordered by performance.

2.5 Component interfaces

2.6 Architectural styles and patterns

2.6.1 Four-tier system architecture

We chose this architecture mostly for portability: it decouples the client implementation from the business logic so it is easier to support a vast number of client types.

A light client also means it can be supported on a vast variety of even older hardware. Other benefit from this choice are:

- *Flexibility*: the communication through known interfaces makes the modification of single components easier without the need to check the entire system functionality.
- *Scalability*: the separation of layers makes it easier to expand just the critical components, saving money and time on less critical parts of the system.
- *Fault tolerance*: the presence of redundant servers guarantees functionality in the case of high load or some server fault.
- *Load distribution*: the load balancer could be dynamically implemented to preview high spikes of load and dynamically assign and free computation power from the critical components. Especially useful if it is deployed in a server farm.

2.6.2 RESTful architecture

The choice of the RESTful architecture to communicate to the application server brings many advantages:

- *Scalability*, *flexibility* and *portability*: the stateless nature of the server leads to an easier maintenance, meaning that a server can be migrated or added without any changes to data.
- *Independence*: the greater separation between the presentation and application layer makes the development of clients easier, so they can be built independently from the rest of the system.

2.6.3 Model View Controller

Model-View-Controller is a design pattern used to divide the program into three interconnected elements. It is useful to separate the internal representation of information from how the information is then presented to the user. The three components are:

- *Model*:the system internal dynamic data structure. It directly manages data and rules of the application. It keeps the internal state consistent.
- *Controller*: accepts input from the view and applies elaboration to the model.
- *View*: representation of the internal data for the user. Multiple views of the same information are possible.