POLITECNICO
MILANO 1863

Computer Science and Engineering

Software engineering 2 Project

DREAMS

DD

Desing Document

Versione 1.0 – 09/01/2022

Pistone Santi Pier – 996419

Zouzoua Axel Israel Ble - 968931

Sommario

1 Introduction 3
A. Purpose 3
B. Scope 3
C. Definitions, acronyms and abbreviations 3
C.1. Definitions 3
C.2. Acronyms 3
C.3. Abbreviation 4
D. Revision History 4
E. References 4
F. Document Structure 5
2. Architectural Design 6
A. Overview 6
B. Component view 6
C. Deployment view 12
D. Runtime view 13
E. Component interface 18
F. Selected architectural styles and patterns 23
G. Other design decisions 24
3. User Interface Design 24
4. Requirements Traceability 29
5. Implementation, Integration and test plan 32
Implementation Plan 32
Integration Plan 32
Testing Plan 33

1 Introduction

A. Purpose

This document is intended to illustrate more in depth the structural aspects of DREAMS and give more technical details about it, rather than the RASD does.
The actual document, indeed, is primarily addressed to the software development team and wants to highlight the architectural design of the system from different point of view, namely:

- The high level architecture
- The main components and their interactions, that is their interfaces provided one for another
- The Runtime behavior
- The design patterns

B. Scope

The product addresses two categories of users – Policy Makers and Farmers – each with different needs and concerns; therefore it has to be designed in such a way that it satisfies both parts' necessities.

C. Definitions, acronyms and abbreviations

C.1. Definitions

- Special incentive : an encouragement given by the policy maker to the farmer
- Weather forecast: meteo conditions of a given location
- Forum discussion: a virtual discussion between the farmers
- Meteorological adverse event: particularly bad weather conditions a farmer could encounter
- HelpTicket: special help support request generated by a farmer to a policy maker
- Direct Connection: a virtual connection a policy maker can establish between 2 farmers; one having bad performances and the other having good performances so that he can help the first one into improving his production

C.2. Acronyms

- RASD : Requirement Analysis and Specification Document

- API : Application Programming Interface
- DD : Design Document
- DBMS : DataBase Management System

C.3. Abbreviation

- WP : World Phenomena
- SP : Shared Phenomena
- G : Goal
- D : Domain
- R : Requirements
- OS : Operating System
- MVC : Model View Controller

D. Revision History

Version	Date	Description
1.0	09/01/2022	First Version

E. References

- Specification Document “Assignments AA 2021-2022.pdf”
- Telenagana state documentation : https://agri.telangana.gov.in/content.php?U=3%20&&%20T=Action%20Plan
- https://agri.telangana.gov.in/open_record_view.php?ID=959
- AlloyDynamic Model example: http://homepage.cs.uiowa.edu/~tinelli/classes/181/
- Spring10/Notes/09-dynamic-models.pdf"
- IEEE Std 830-1993 - IEEE Guide to Software Requirements Specifications.
- IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.

F. Document Structure

1. INTRODUCTION

A. Purpose
B. Scope
C. Definitions, Acronyms, Abbreviations
D. Revision history
E. Reference Documents
F. Document Structure

2. ARCHITECTURAL DESIGN

A. Overview: High-level components and their interaction
B. Component view
C. Deployment view
D. Runtime view: You can use sequence diagrams to describe the way components interact
to accomplish specific tasks typically related to your use cases
E. Component interfaces
F. Selected architectural styles and patterns: Please explain which styles/patterns you used,
why, and how
G. Other design decisions

3. USER INTERFACE DESIGN

Provide an overview on how the user interface(s) of your system will look like; if you have included this part in the RASD, you can simply refer to what you have already done, possibly providing here some extensions if applicable.

4. REQUIREMENTS TRACEABILITY

Explain how the requirements you have defined in the RASD
map to the design elements that you have defined in this document.

5. IMPLEMENTATION, INTEGRATION AND TEST PLAN:

Identify here the order in which you plan
to implement the subcomponents of your system and the order in which you plan to integrate
such subcomponents and test the integration.

6. EFFORT SPENT:

In this section you will include information about the number of hours each group member has worked for this document.

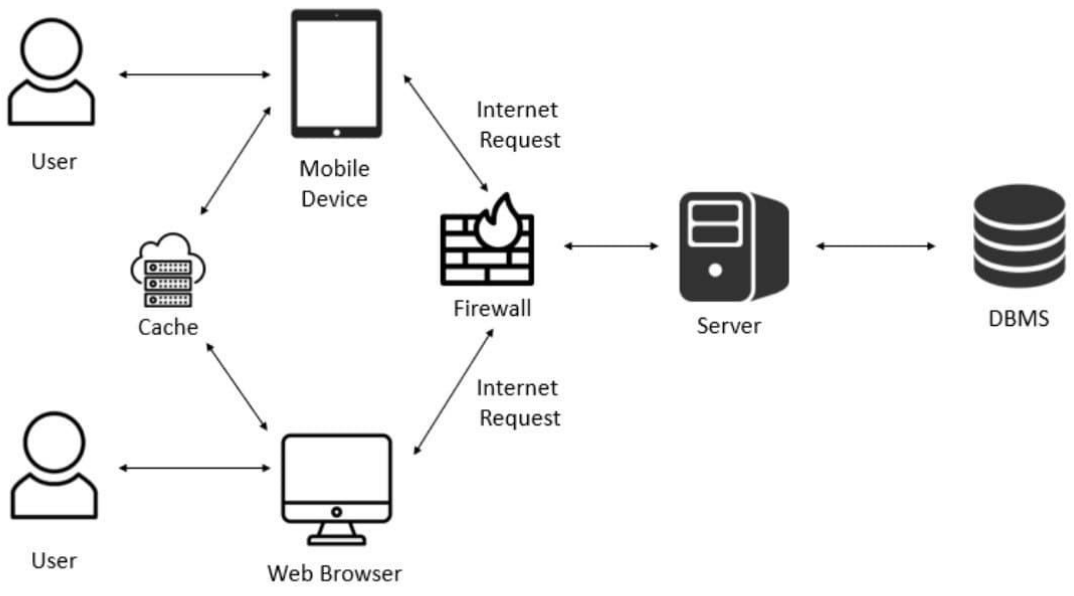
7. REFERENCES

2. Architectural Design

A. Overview

The figure below shows the interaction between the system and the user at a very high level. User's device can be either a mobile device (smartphone or tablet) or a computer. The software application is available only for tablets, while the other devices will be provided with the web application only.
These devices constitute the client part of the system, and therefore will contain the user application functionalities.

The server part will contain all the most internal software side components of the system on a device and the database on another device.
The choice of such a distribution and architectural style will be better explained in the next sections.



B. Component view

The following diagrams illustrate the main components of the system and the interfaces through which they connect and interact.
As mentioned before, the system is provided with a client-server architectural style:

- The client side is composed of three elements which are policy maker application, farmer application and admin application.

- The server side is made of five main components:

    - The Production Manager component will oversee supporting anything directly related with the production information to be communicated or visualized by the user.

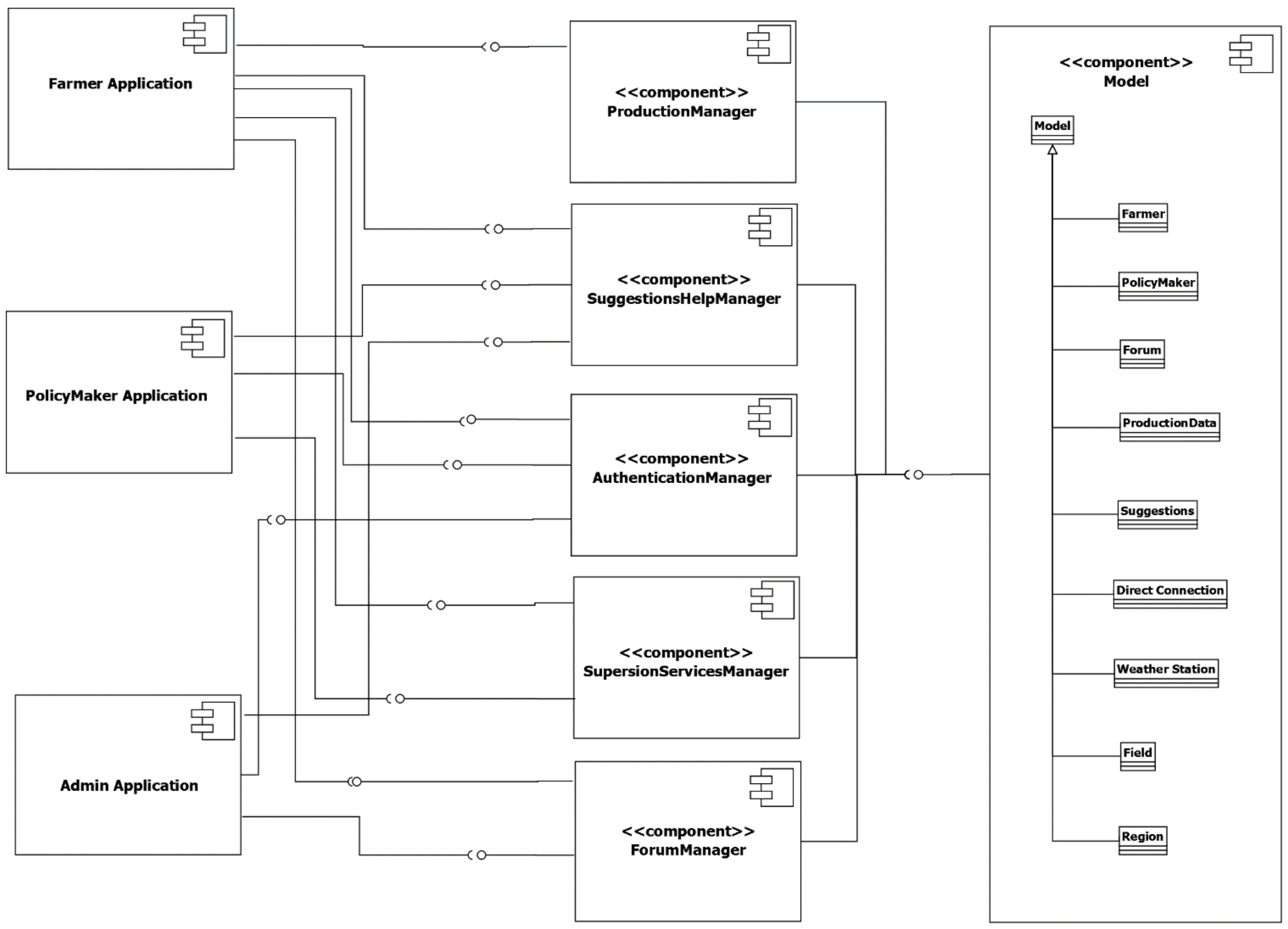
    - The Suggestions & Help Manager will support all the personalized suggestions of the farmers and all the process related to the generation and gestion of a ticket.

    - The Authentication Manager component provides the interfaces necessary to manage the access to the system of the different categories of users and their accounts management features, guaranteeing security.

    - The Supervision Services Manager is responsible for carrying out the activities of closer supervision of the policy maker. Indeed, it provides the interfaces able to manage the pairing of farmers and the chat services.

    - The Forum Manager contains the interfaces needed to bear all the forum activities of the farmers.

Each of these components is the better analyzed further in this section.

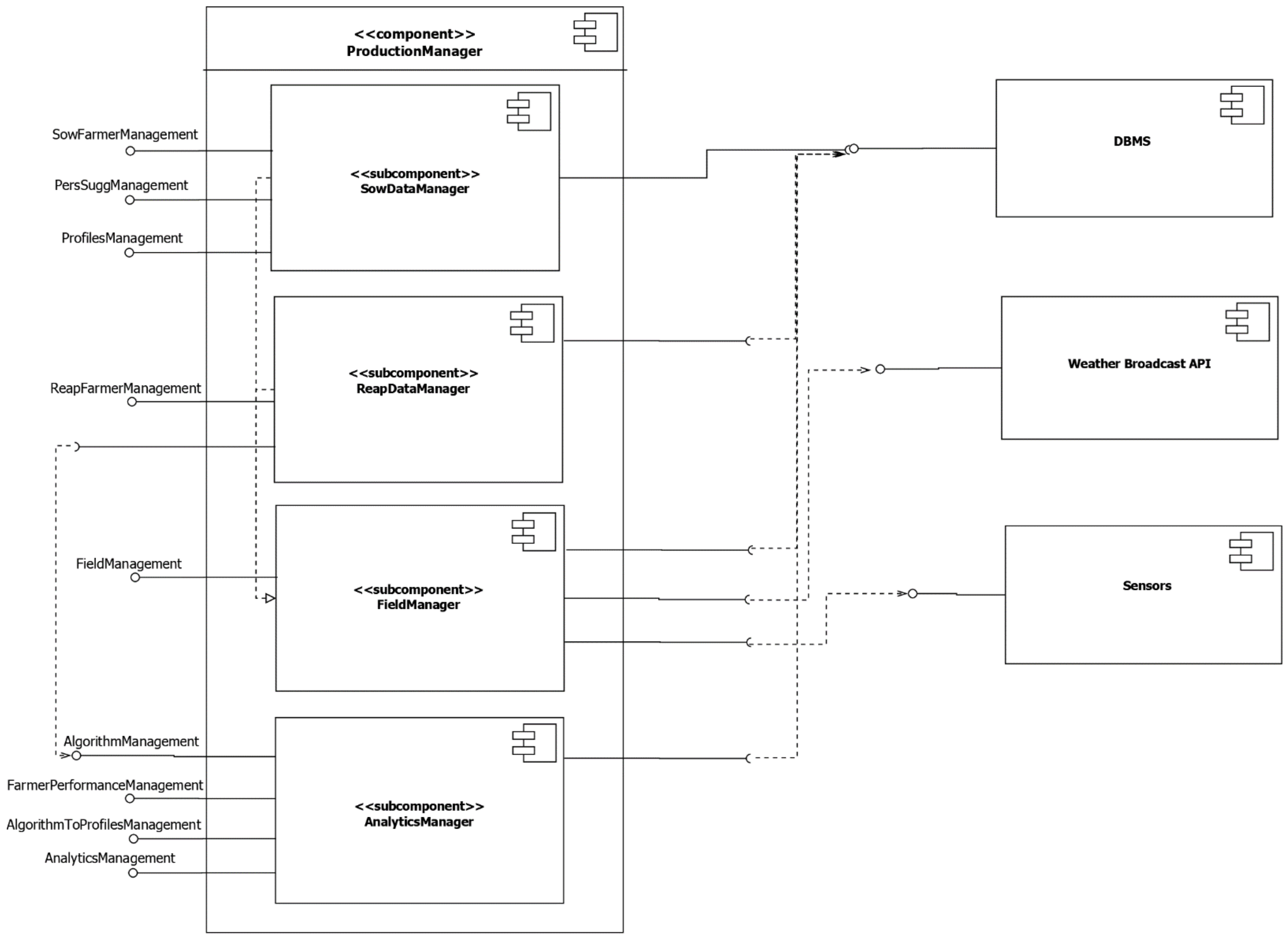


Production Manager component projection

This component is constituted of four subcomponents:

- *SowData Manager* handles the farmer's production information from the beginning of his seasonal farming activity.
- *ReapData Manager* handles the farmer's production data at the end of his seasonal farming activity; those data are then conveyed to the Analytics Manager.
- *Field Manager* responsible for the interfacing with the external components such as the weather broadcast API and sensors deployed within each field; the first two components depend on this one.
- *and Analytics Manager* responsible for the algorithm that provides evaluations of the farmers' performances and their evolution in time.

The interfaces provided by these subcomponents are: *SowFarmerManagement, PerSuggManagement, ProfilesManagement, ReapFarmerManagement, FieldManagement, AlgorithmManagement, AlgorithmToProfilesManagement and AnalyticsManagement.*

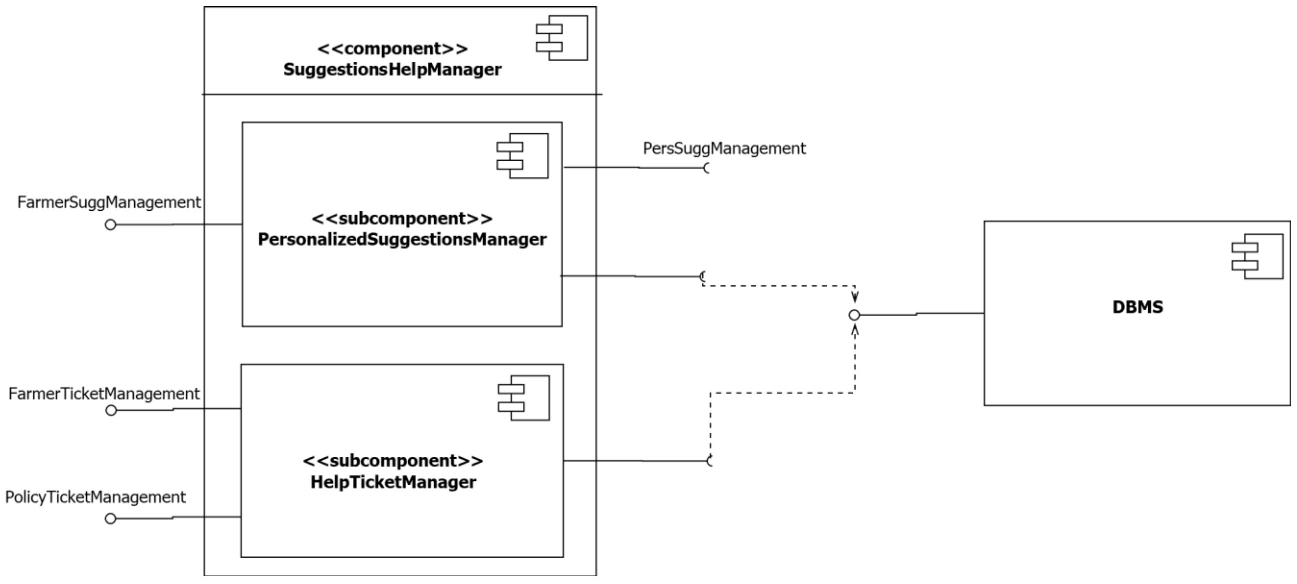


Suggestions & Help Manager component projection

It counts two subcomponents:

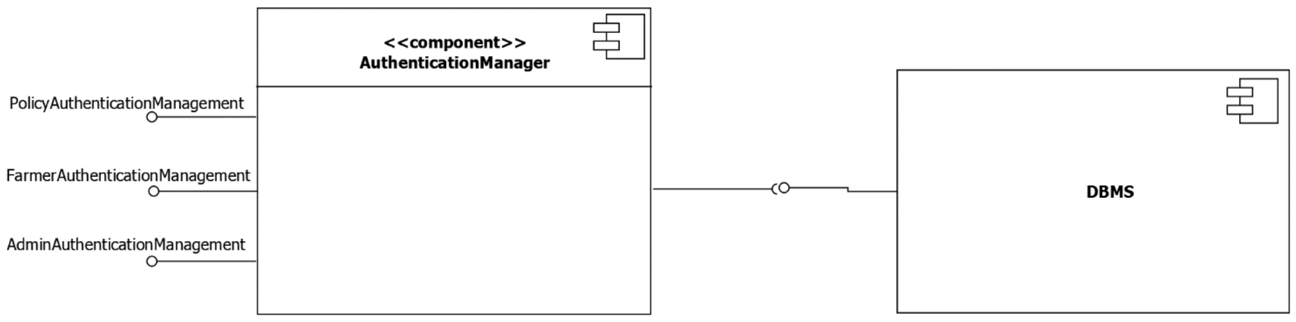
- *PersonalizedSuggestions Manager:* interfaces with the SowData Manager subcomponent through *PersSuggManagement*, it allows the farmer to deal with all the farming indications he is suggested by the agronomists.
- *and HelpTicket Manager*: supports the whole generation and gestion of a ticket whenever a farmer needs help.

The interfaces dispensed by these two components are: *FarmerSuggManagement, FarmerTicketManagement and PolicyTicketManagement*



Authentication Manager component projection

The interfaces provided by this component are: *PolicyAuthenticationManagement, FarmerAuthenticationManagement and AdminAuthenticationManagement.*



<<component>>
AuthenticationManager
PolicyAuthenticationManagement
FarmerAuthenticationManagement
AdminAuthenticationManagement
DBMS

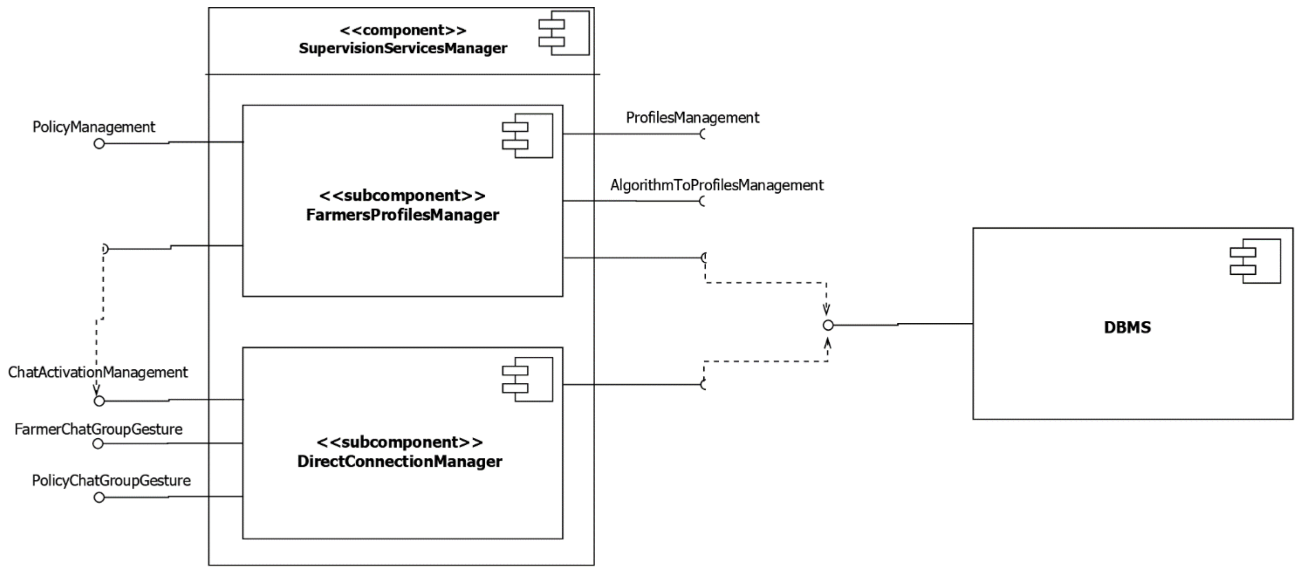
Supervision Services Manager component projection

In the supervision services manager there are two subcomponents:

- *FarmersProfiles Manager* : allows the policy maker to surf through the profiles of the farmers under his supervision and possibly establish a connection between them willingly.

- *and DirectConnection Manager:* supports the chat service whenever a “direct connection” is established.

The interfaces provided by these components are: *PolicyManagement, ChatActivationManagement, FarmerChatGroupGesture and PolicyChatGroupGesture*

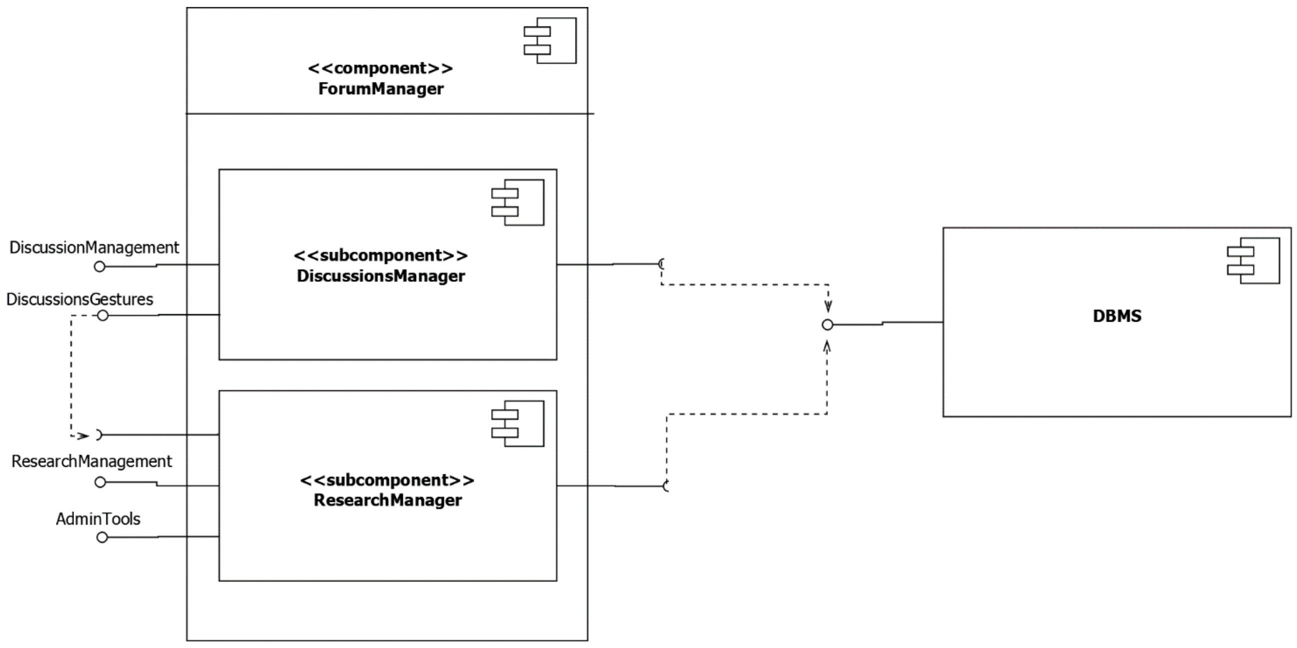


Forum Manager component projection

The forum manager component is of two subcomponents:

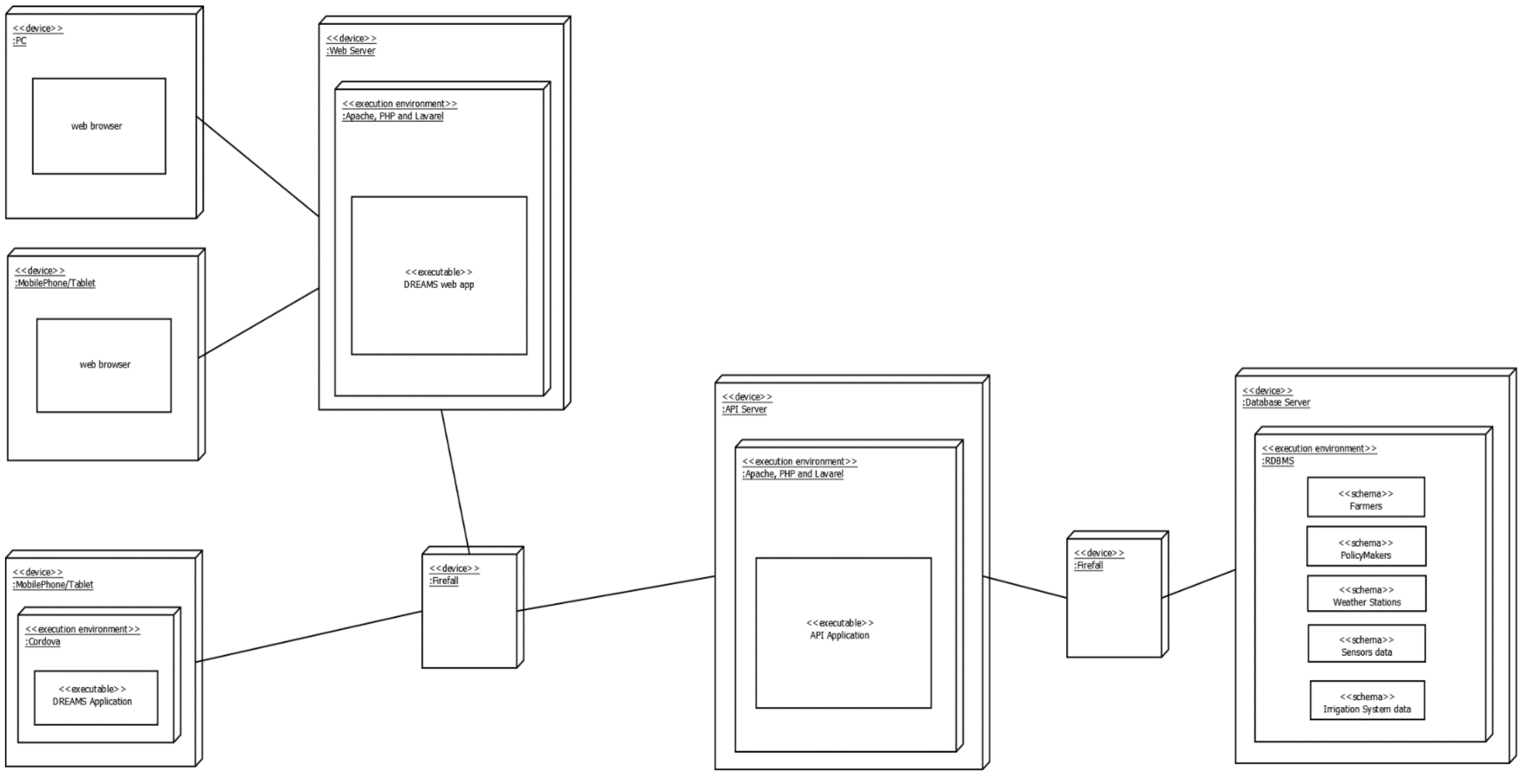
- *DiscussionsManager:* supports all the activities related to the discussions, from joining the discussion to adding an answer to it.
- *and ResearchManager:* provides the interfaces necessary to manage the activity of searching for discussions within the forum area.

The interfaces provided by these two subcomponents are: *DiscussionManagement, DiscussionsGestures, ResearchManagement and AdminTools*



C. Deployment view

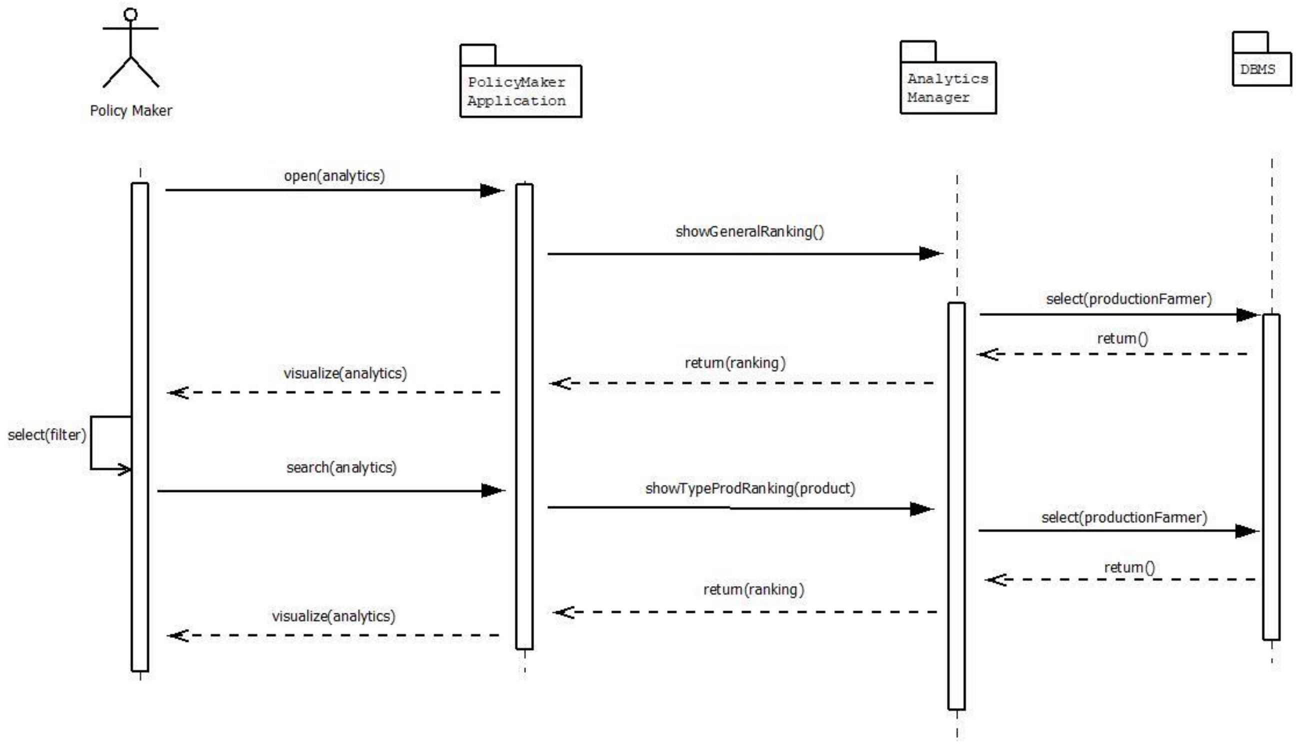
The following diagram shows the execution architecture of the system and represents the distribution of the software artifacts to deployment targets.
In the diagram ‘<<executable>>’ and ‘<<schema>>’ are standard UML stereotypes that apply as artifacts.
The overall architecture is developed in a Three Tiers architectural style.



D. Runtime view

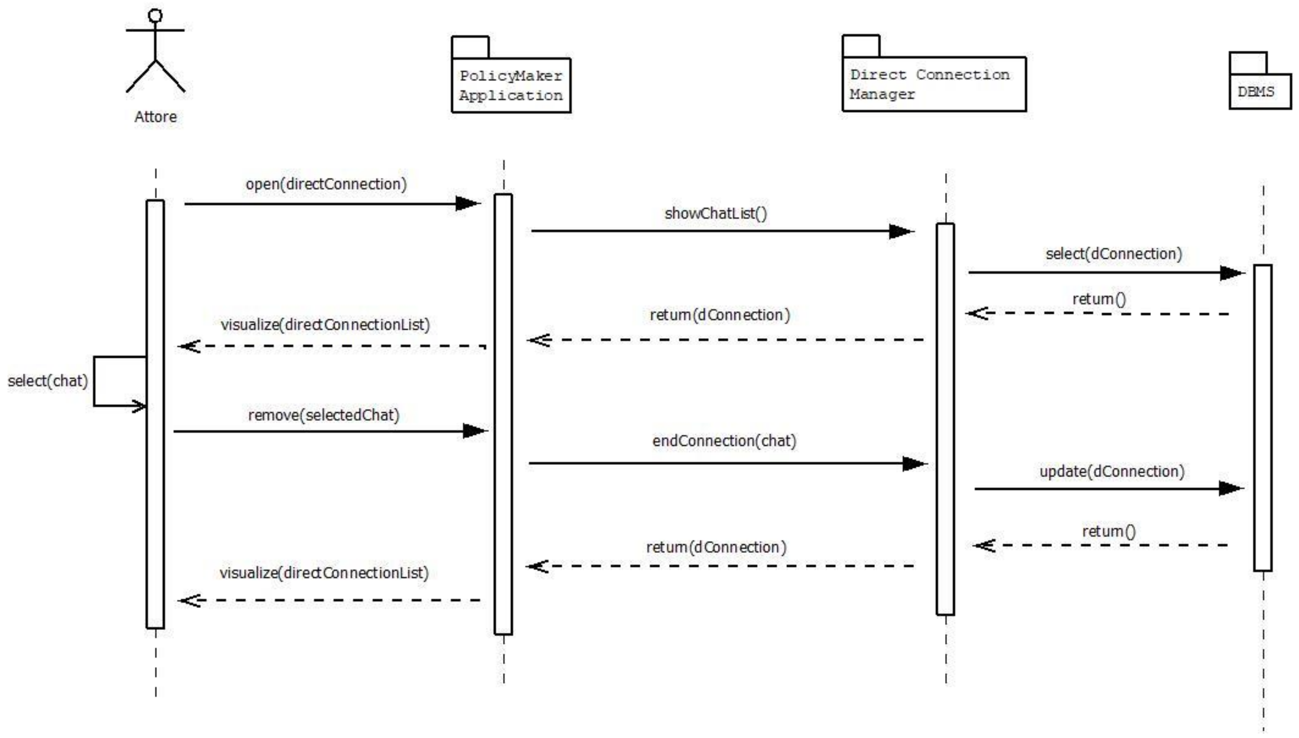
Policy Maker visualizes analytics

The policymaker accesses the analytics page through his application, and calls the showGeneralRanking () function, which is offered by the AnalyticsManager subcomponent, which generates the general ranking of the farmers under his control directly from the database and shows it to him. Subsequently, the PolicyMaker decides to apply filters to show the ranking based on some criteria (of its choice), and the system performs a search: in this case the application calls the showTypeProdRanking (product) function, a management function of more defined ranking, based on the type of product selected.



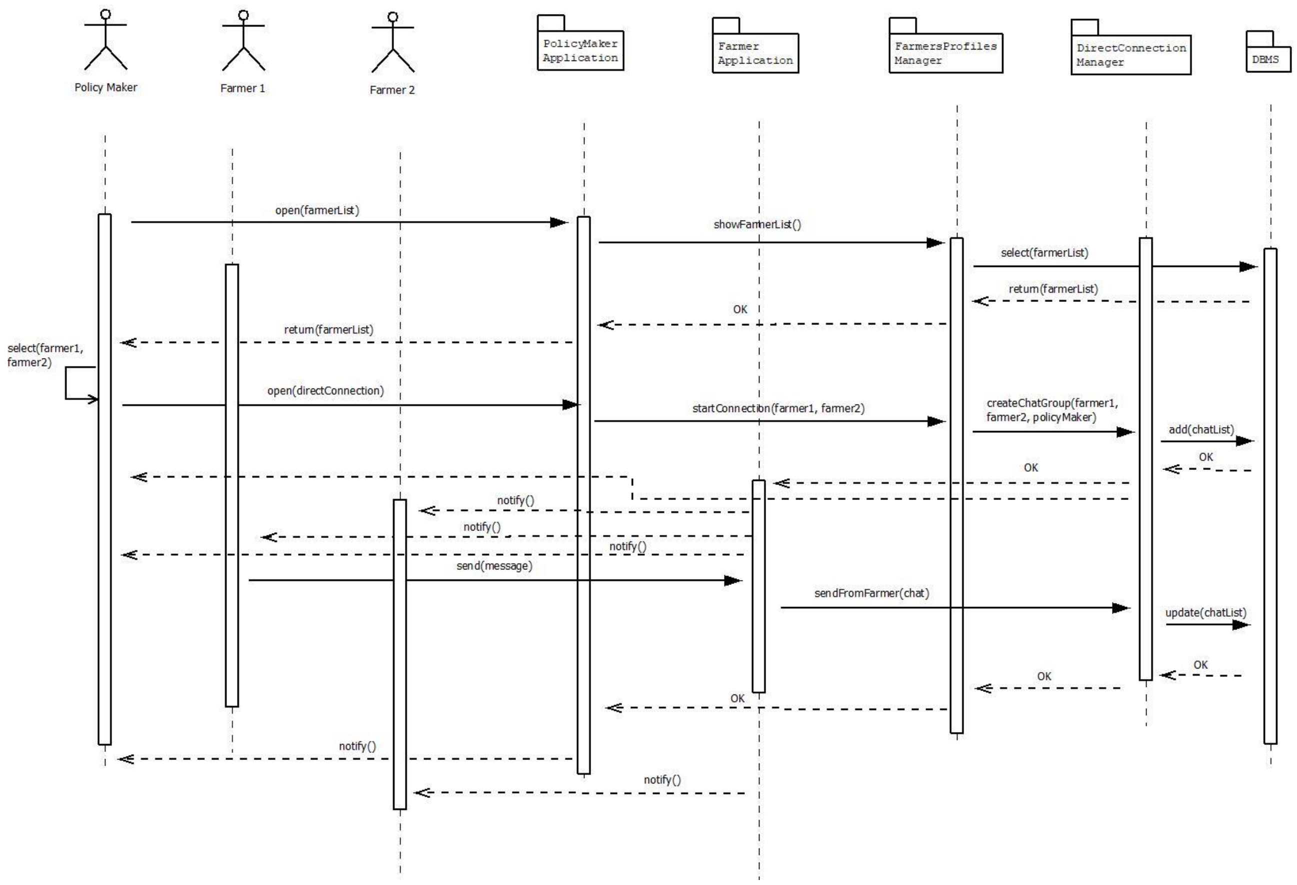
End of a direct connection by a Policy Maker

A PolicyMaker, after opening a direct connection between 2 farmers, can decide to close it at any time, therefore, through the PolicyMaker Application, he requests the chatList in the System; the PolicyMaker then decides which chat he wants to close, selects it and finally clicks on remove: the System receives the endConnection call and updates the database, in order to keep track of the closed directConnections; finally it displays the updated chatList.



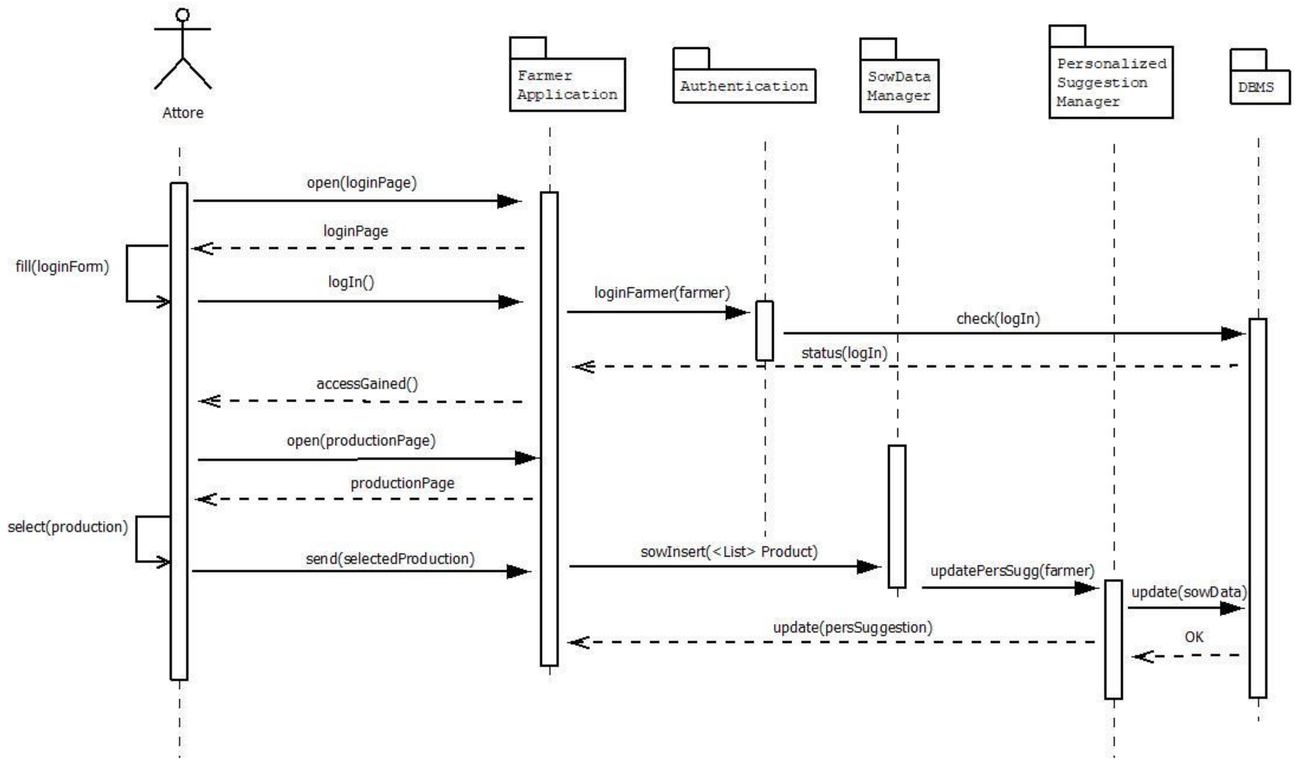
Policy Maker establishes a direct connection

In the following, it is shown how a 'direct connection' is created.
The policy maker, from the FarmerProfiles Manager is shown the list of the profiles of the farmers under his supervision through showFarmerList(); there he can select a farmer and another one and then link them; a chat group containing the two farmers and the policy maker is then create through createChatGroup(); the chat gestures are also unlocked to the users.



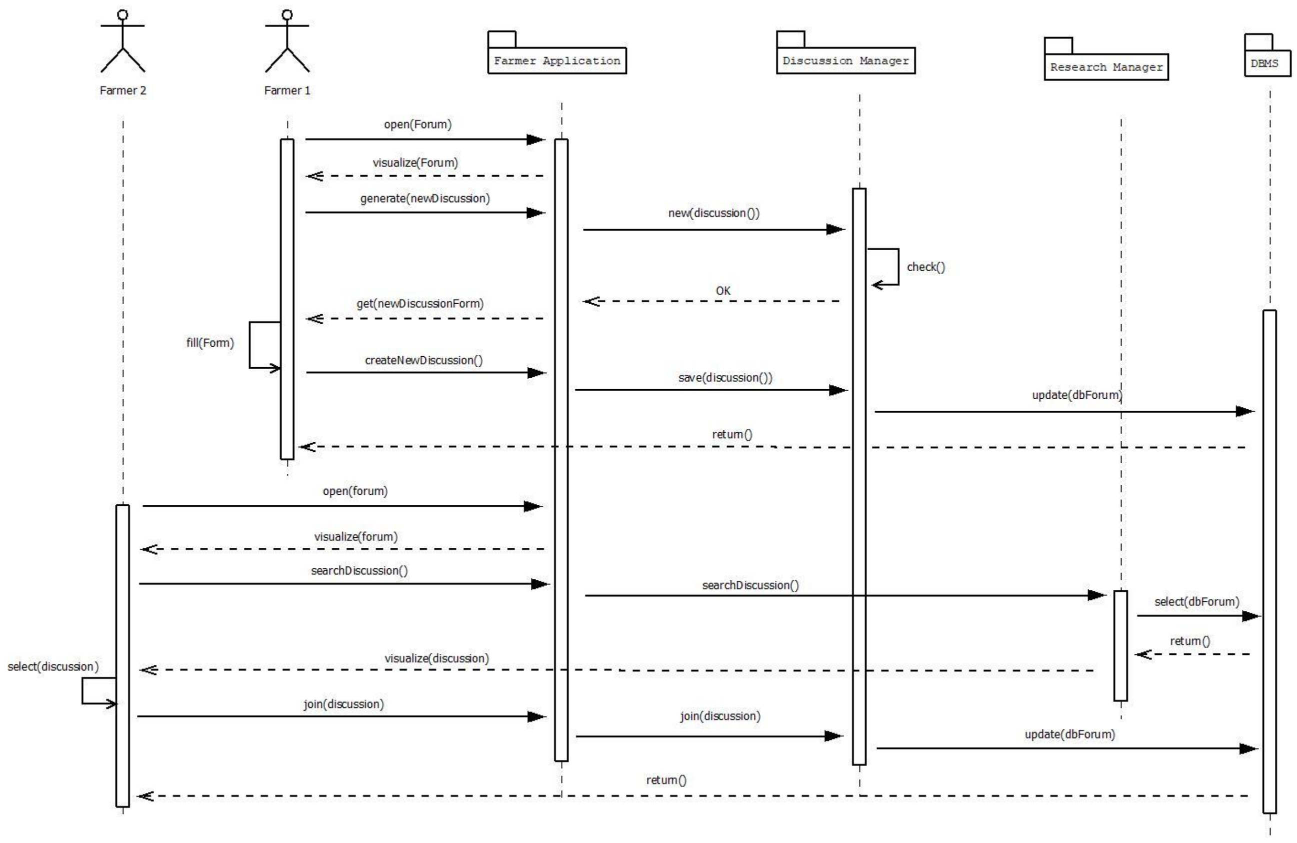
First login of a Farmer

On his first login a farmer has to insert the information about his production data, especially, information about the products he is going to cultivate. By doing so he will be able to unlock the other functionalities of the software dedicated to him such the personalized suggestions section that allows him to keep himself updated.
The operations and the interactions between the involved components of the system in such an activity are illustrated in the following diagram.



Forum: creation of a new discussion / joining for a discussion

Here are showed two farmers doing two different activities.
The first one is interested into creating a new forum discussion, while the second one is interested into searching for a discussion about his curiosity.
The operations required and the interactions between the involved components for both the operations are highlighted in the following diagram.



E. Component interface

Here are showed in detail the functions and operations offered by each of the interfaces.

- SowFarmerManagement and ReapFarmerManagement interface directly with the Farmer application and allows the user to manage the submission and visualization of the seasonal farming data.

- PersSuggManagement allows the Production Manager and the Suggestions & Help Manager components to interface; indeed it helps into the updating the visualization of the farmer's personalized suggestions as his sow data are changed or updated.

- ProfilesManagement allows FarmerProfiles Manager and Production Manager to interface.

- FieldManagement rules the visualization of the weather broadcast, the soil humidity, the quantity of used water within a field, for all the fields of a farmer. It interfaces directly with the farmer Application.

- AlgorithmManagement allows ReapData Manager and Analytics Manager to interface;

- FarmerPerformanceManagement and AlgorithmToProfilesManagement have nearly the same functionalities; while the first interfaces directly with the Farmer Application allowing the visualization of the performances of the farmer, the second one interfaces with the FarmersProfiles Manager component, offering a summary of the analytics.

- AnalyticsManagement interfaces with the Policy Maker application and is responsible for conveying all the well detailed output from the algorithm that is in charge of providing a measurement of the performances of the farmers and of their evolution in time.

- PolicyManagement interfaces with the Policy maker Application; it offers the functionalities to surf through the farmers profiles.

- ChatActivationManagement allows the two subcomponents FarmersProfiles Manager and DirectConnection Manager to interface.

- PolicyChatGroupGesture and FarmerChatGroupGesture interface with the Policy Maker and Farmer Applications respectively; they offer to the respective personalities all the chat service functionalities they need.

- PolicyAuthenticationManagement, FarmerAuthenticationManagement and AdminAuthenticationManagement interface with their homologue client components ensuring the functionalities need for the accessibility and security within the system.

- FarmerSuggManagement interfaces with the Farmer Application; it oversees conveying the data concerning the well update personalized suggestions of the farmer.

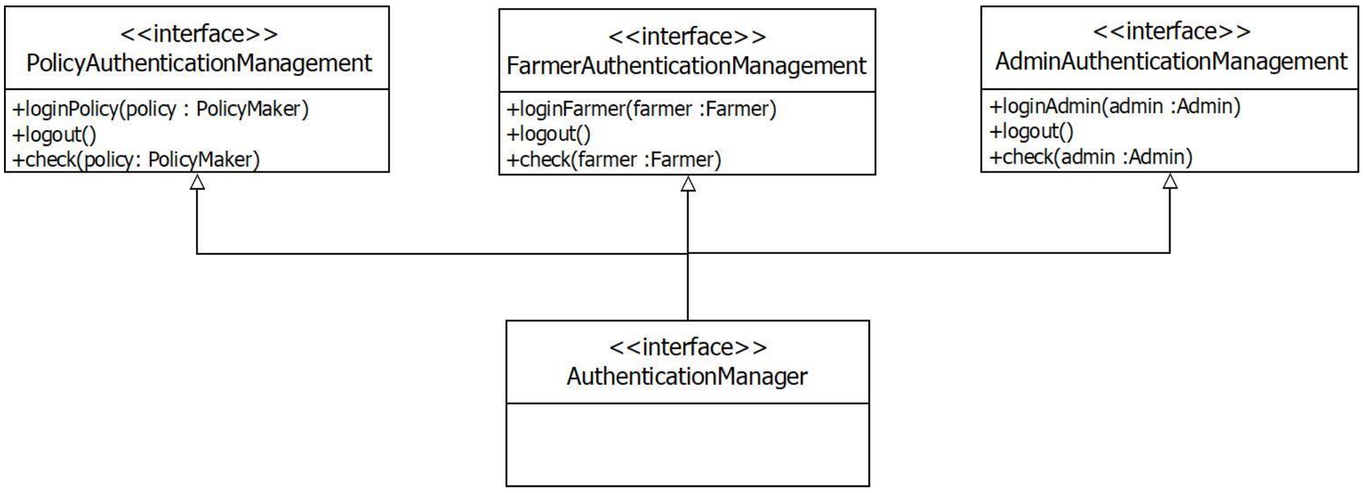
- FarmerTicketManagement and PolicyTicketManagement interface with the Farmer Application and Policy Maker Appplication; it is responsible for offering the tools necessary to deal with the generation by a farmer and the gestion by a policy maker of a ‘helpTicket’.

- DiscussionsGestures and DiscussionManagement: they both interface with the Farmer Application; they offer the tools to support and manage a discussion.

- ResearchManagement also interfaces directly with the farmer Application; it ensures the well-functioning of the “search engine” within the forum.

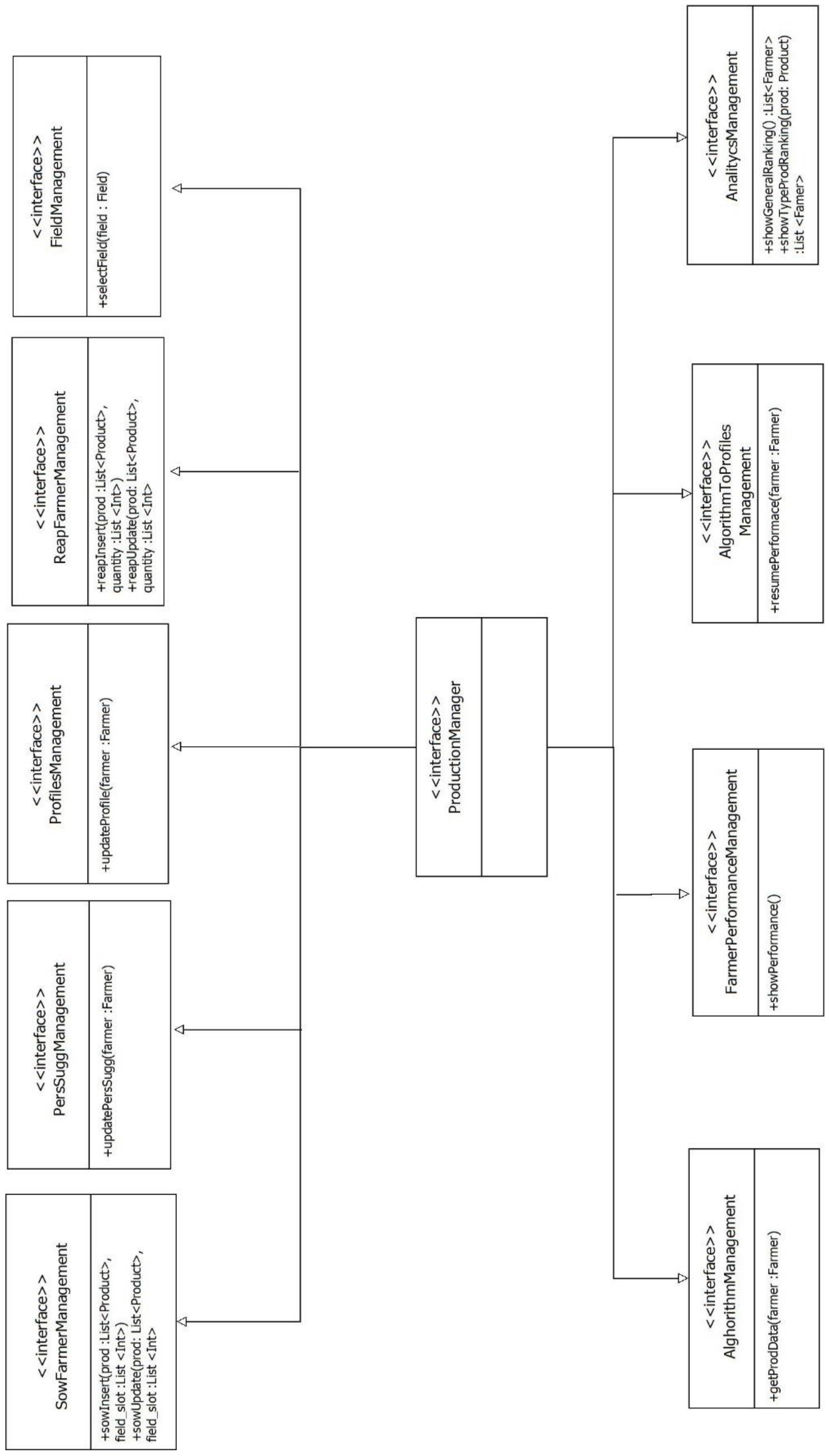
- AdminTools interfaces with the Admin Application; it is conceived to support the tools the forum moderators need to rule the forum.

Authentication Manager Interfaces

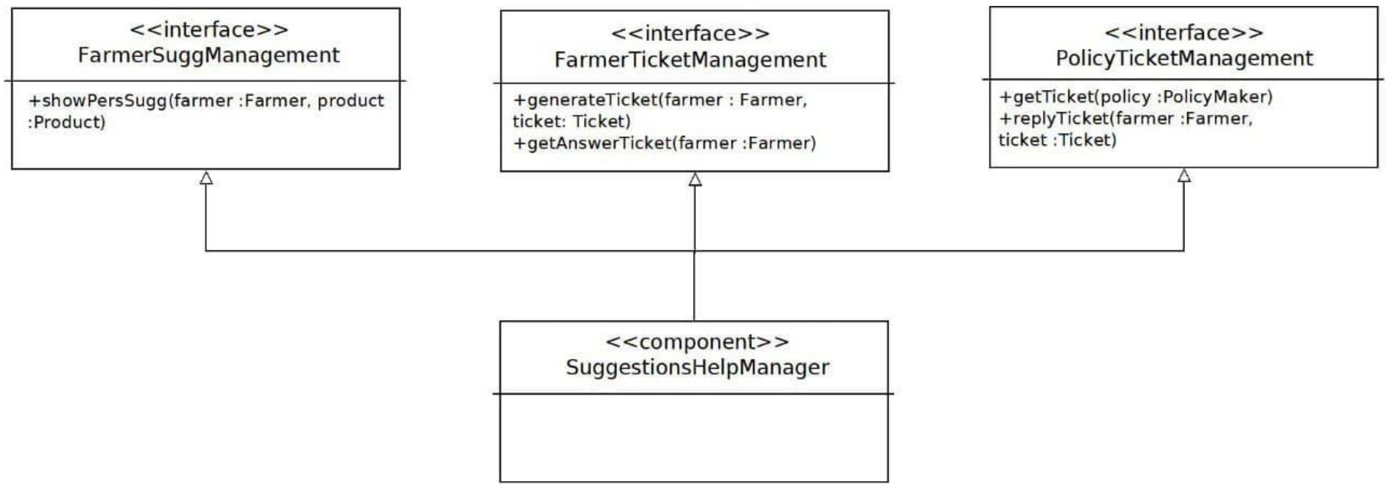


<<interface>>
PolicyAuthenticationManagement
+loginPolicy(policy : PolicyMaker)
+logout()
+check(policy: PolicyMaker)
<<interface>>
FarmerAuthenticationManagement
+loginFarmer(farmer :Farmer)
+logout()
+check(farmer :Farmer)
<<interface>>
AdminAuthenticationManagement
+loginAdmin(admin :Admin)
+logout()
+check(admin :Admin)
<<interface>>
AuthenticationManager

Production Manager Interfaces

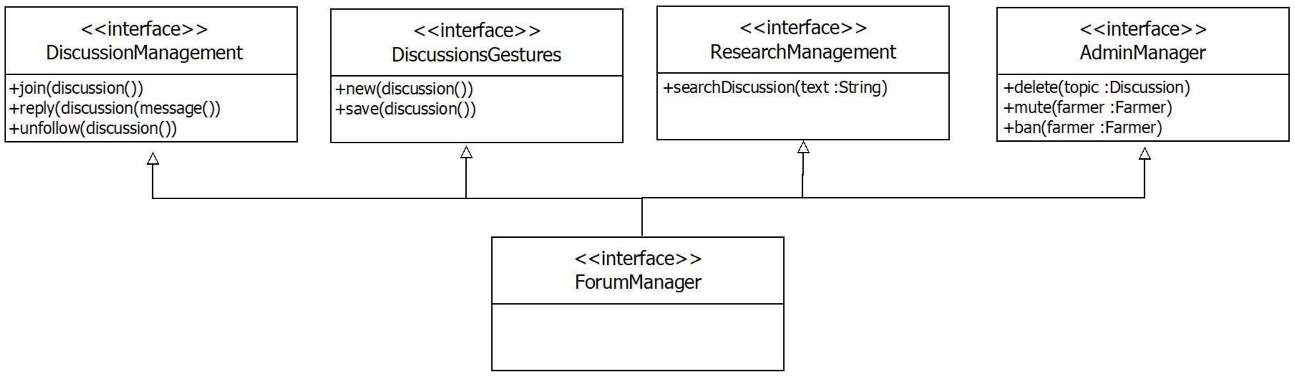

<<interface>>
FieldManagement
+selectField(field : Field)
<<interface>>
ReapFarmerManagement
+reapInsert(prod :List<Product>, quantity :List <Int>)
+reapUpdate(prod: List<Product>, quantity :List <Int>
<<interface>>
ProfilesManagement
+updateProfile(farmer :Farmer)
<<interface>>
PersSuggManagement
+updatePersSugg(farmer :Farmer)
<<interface>>
SowFarmerManagement
+sowInsert(prod :List<Product>, field_slot :List <Int>)
+sowUpdate(prod: List<Product>, field_slot :List <Int>
<<interface>>
ProductionManager
<<interface>>
AnalitycsManagement
+showGeneralRanking() :List<Farmer>
+showTypeProdRanking(prod: Product) :List <Famer>
<<interface>>
AlgorithmToProfiles Management
+resumePerformace(farmer :Farmer)
<<interface>>
FarmerPerformanceManagement
+showPerformance()
<<interface>>
AlghorithmManagement
+getProdData(farmer :Farmer)

Suggestions & Help Manager Interfaces

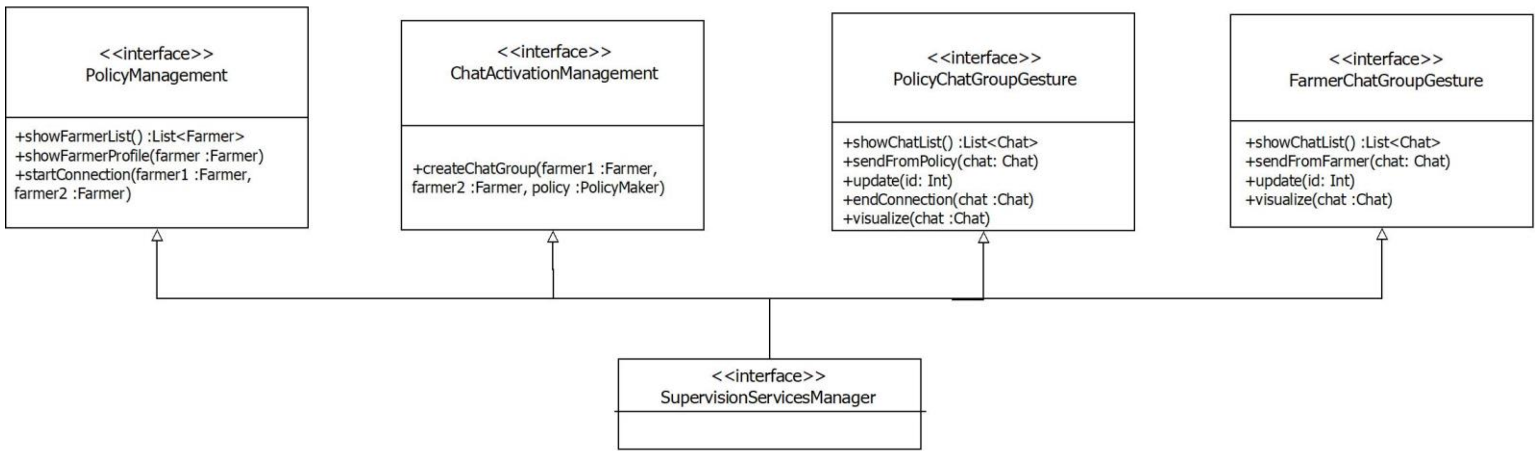


<<interface>>
FarmerSuggManagement
+showPersSugg(farmer :Farmer, product :Product)
<<interface>>
FarmerTicketManagement
+generateTicket(farmer : Farmer, ticket: Ticket)
+getAnswerTicket(farmer :Farmer)
<<interface>>
PolicyTicketManagement
+getTicket(policy :PolicyMaker)
+replyTicket(farmer :Farmer, ticket :Ticket)
<<component>>
SuggestionsHelpManager

Forum Manager Interfaces



Supervision Services Manager Interfaces



<<interface>>
PolicyManagement
+showFarmerList() :List<Farmer>
+showFarmerProfile(farmer :Farmer)
+startConnection(farmer1 :Farmer, farmer2 :Farmer)
<<interface>>
ChatActivationManagement
+createChatGroup(farmer1 :Farmer, farmer2 :Farmer, policy :PolicyMaker)
<<interface>>
PolicyChatGroupGesture
+showChatList() :List<Chat>
+sendFromPolicy(chat: Chat)
+update(id: Int)
+endConnection(chat :Chat)
+visualize(chat :Chat)
<<interface>>
FarmerChatGroupGesture
+showChatList() :List<Chat>
+sendFromFarmer(chat: Chat)
+update(id: Int)
+visualize(chat :Chat)
<<interface>>
SupervisionServicesManager

F. Selected architectural styles and patterns

Model View Controller (MVC)

Model-View-Controller pattern has been widely in our application.
It has been selected to adopt Model-View-Controller (MVC) in order to guarantee the maintainability and the reusability of the code. This software pattern is particularly adapted for the development of both web and mobile application in an object-oriented style of programming as java. MVC separates the application into multiple layers of functionalities:

• View: responsible for allowing the users to visualize the data correctly and for taking the users inputs

• Model: responsible for managing the data of the application and for updating the other components

• Controller: responsible for connecting the users' inputs and making decisions for the view. The controller receives the input, optionally validates it and then passes it to the model. Thanks to this it is possible to create components independently of each other and simultaneous development is simplified.

Client – Server

The application is strongly based on a client-server communication model. The clients being the users applications (mobile and browser).
The clients are thin, thus to let the application run on low-resources devices. This approach has been chosen for different reasons:

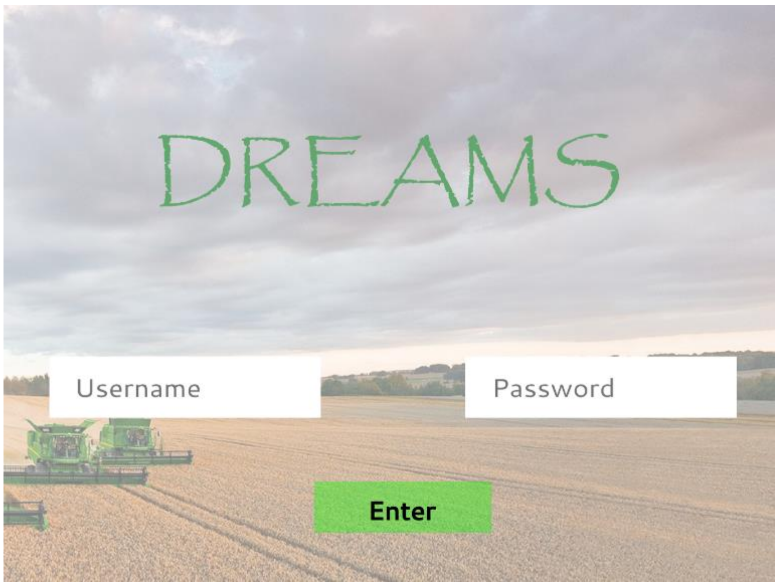
- Data synchronization: there is only one application that manage the data
- Having one unique server application improves the maintenability of our system.
- the application is indipendednt from the number of clients connected (it can be scaled up).
- improves the security between clients, that known only the server endpoint but not other clients.
- It is practical.

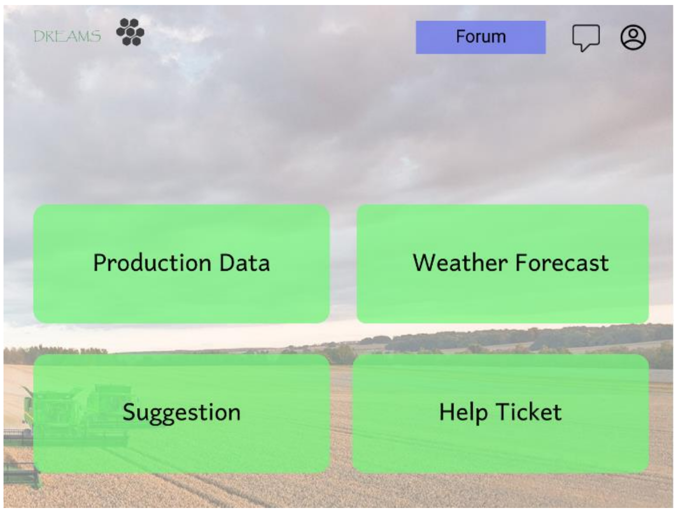
Our Application server will use the Laravel PHP framework, which is an MVC framework. Our Web interface will use AngularJS, which is an MVC framework.

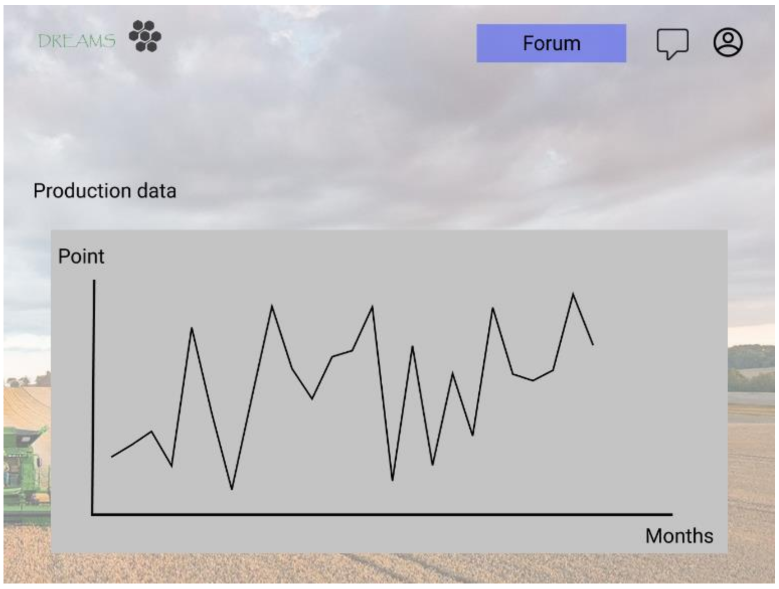
G. Other design decisions

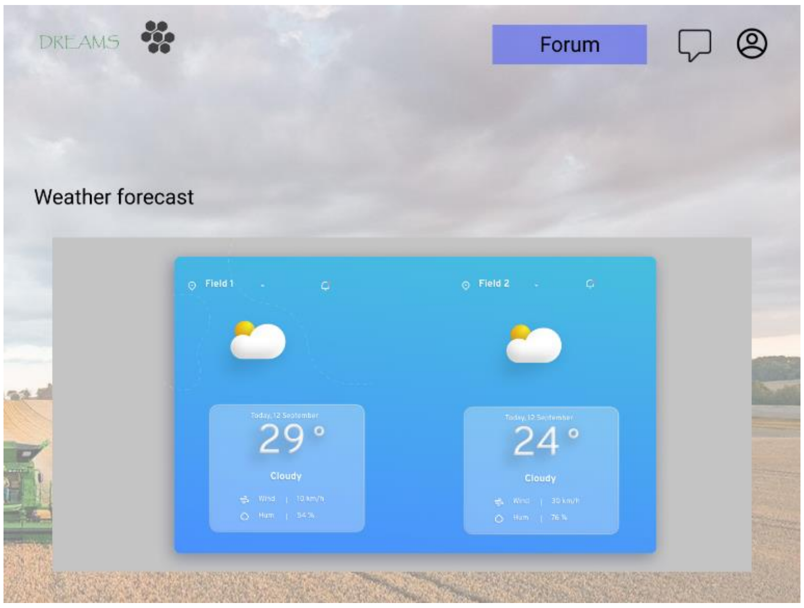
3. User Interface Design

The following are some mockups showing how the system should appear to the user.


DREAMS
Username
Password
Enter


DREAMS
Forum
Production Data
Weather Forecast
Suggestion
Help Ticket


DREAMS
Forum
Production data
Point
Months



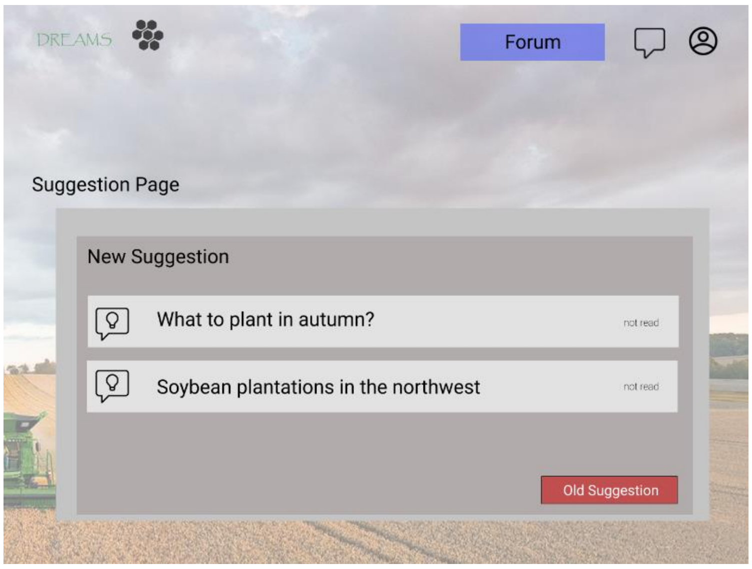
Weather forecast

Field 1

Today, 12 September
29 °
Cloudy
Wind | 10 km/h
Hum | 54 %

Field 2

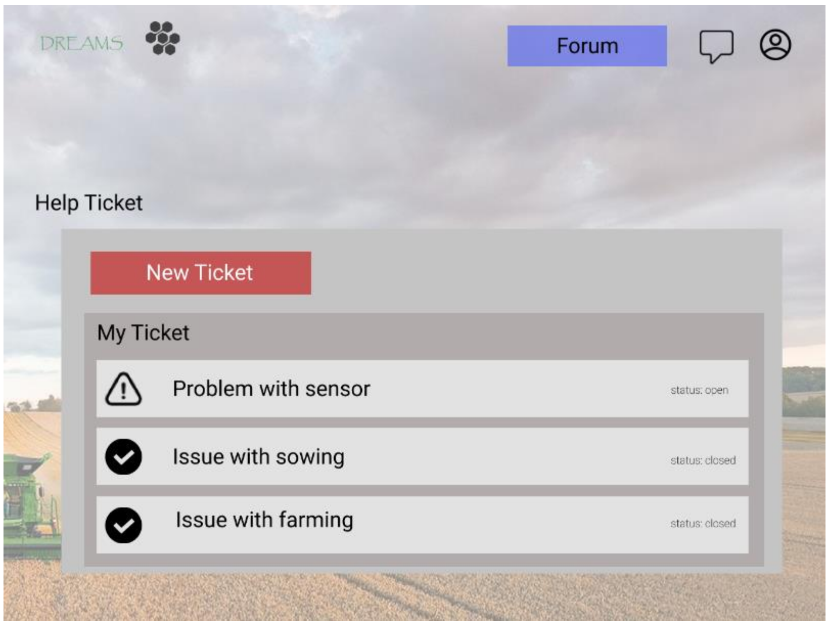
Today, 12 September
24 °
Cloudy
Wind | 30 km/h
Hum | 76 %



Suggestion Page

New Suggestion

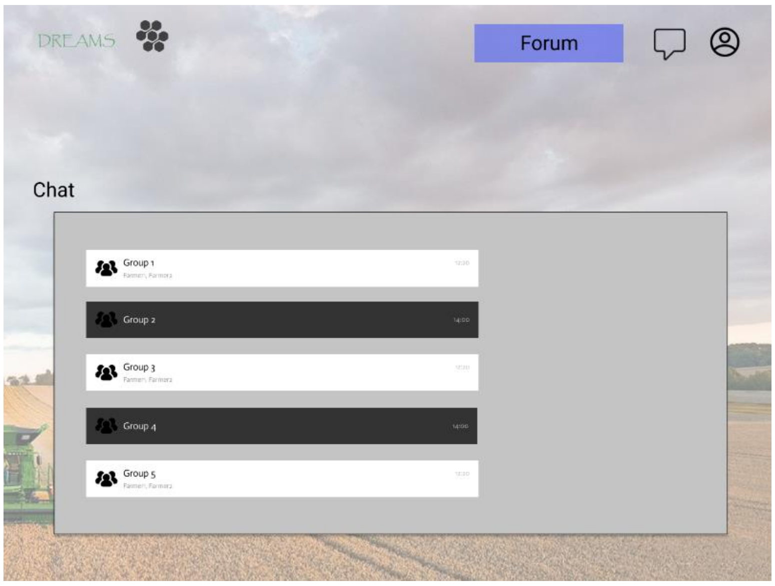
What to plant in autumn?	not read
Soybean plantations in the northwest	not read

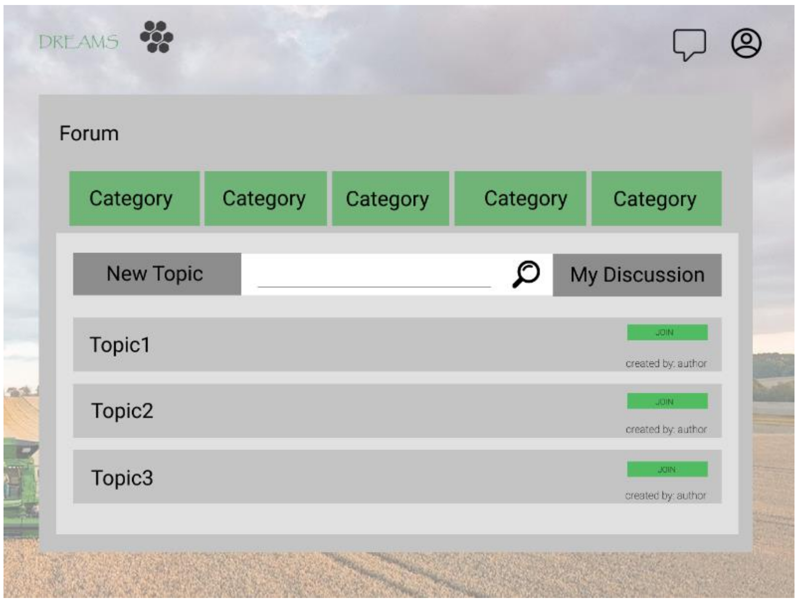


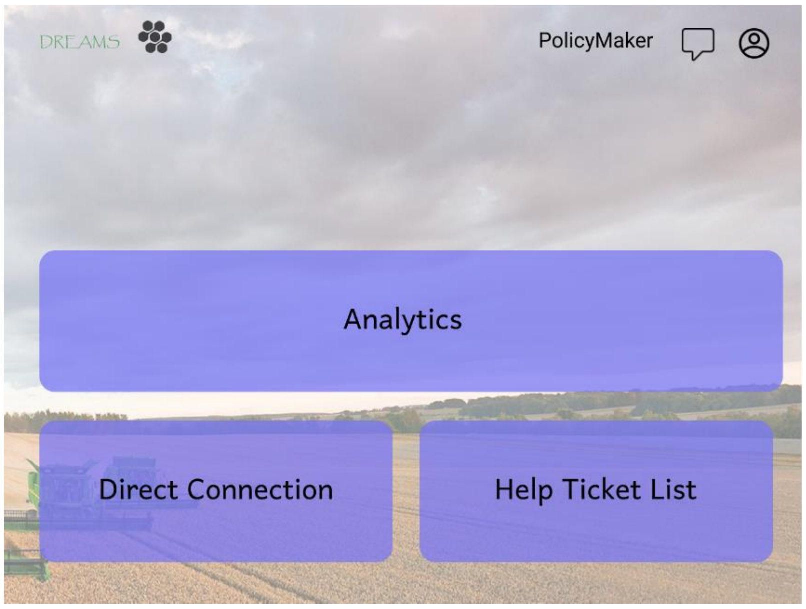
Help Ticket

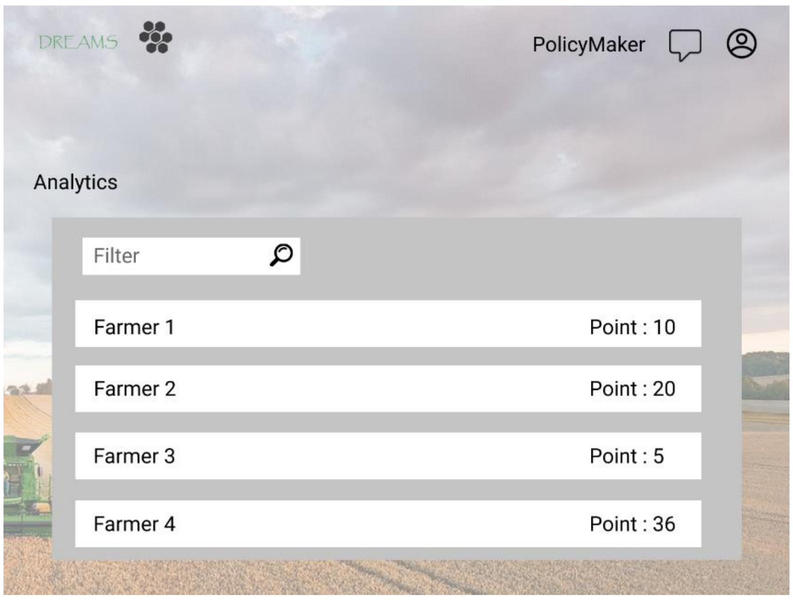
My Ticket

Problem with sensor	status: open
Issue with sowing	status: closed
Issue with farming	status: closed


DREAMS
Forum
Chat
Group 1
Farmer1, Farmer2
12:20
Group 2
14:00
Group 3
Farmer1, Farmer2
12:20
Group 4
14:00
Group 5
Farmer1, Farmer2
12:20


DREAMS
Forum
Category
Category
Category
Category
Category
New Topic
My Discussion
Topic1
JOIN
created by: author
Topic2
JOIN
created by: author
Topic3
JOIN
created by: author





Farmer 1	Point : 10
Farmer 2	Point : 20
Farmer 3	Point : 5
Farmer 4	Point : 36

4. Requirements Traceability

To ensure coherence and consistency between the RASD document and the design document, the designed and elaborated components of the system should be able to guarantee the reliability of the functional requirements introduced into the RASD document.

Therefore, the following represents a mapping between them functional requirements and the actual components: first the requirement and then the components needed to guarantee it.

- [R1]: A farmer that uses the system should be logged
  - Authentication Manager
- [R2]: A farmer obtains the standard DREAMS home page after he has logged-in
  - Authentication Manager
- [R3]: A policy maker that uses the system should be logged
  - Authentication Manager
- [R4]: A Policy Maker obtains the DREAMS special home page after he has logged-in
  - Authentication Manager
- [R5]: A farmer can see the weather broadcast of every field in his possession
  - Production Manager
    - Field Manager
- [R6]: A farmer can visualize only the production data related to him
  - Production Manager
    - SowData Manager
    - ReapData Manager
- [R7]: The system allows the access to the forum to farmers only.
  - Authentication Manager
  - Forum Manager
- [R8]: The system allows a farmer to do research within the forum
  - Forum Manager
    - Research Manager
- [R9]: The system allows a farmer to create a discussion within the forum
  - Forum Manager
    - Discussion Manager

- [R10]: The system allows a farmer to join pictures to the description of a discussion when creating it
  - Forum Manager
    - Discussion Manager

- [R11]: A farmer can add an answer to a discussion
  - Forum Manager
    - Discussion Manager

- [R12]: The system allows a farmer to add pictures to an answer
  - Forum Manager
    - Discussion Manager

- [R13]: All the pictures added should be taken directly from the software
  - Suggestions & Help Manager
    - HelpTicket Manager
  - Supervision Services Manager
    - Direct Connection Manager
  - Forum Manager
    - Discussion Manager

- [R14]: The system allows a farmer to visualize the suggestions provided by the government (Agronomists)
  - Suggestions & Help Manager
    - PersonalizedSuggestions Manager

- [R15]: The system allows a farmer to generate a HelpTicket so he can receive more suggestions and help
  - Suggestions & Help Manager
    - HelpTicket Manager

- [R16]: The system allows a farmer to add pictures to a HelpTicket when creating it
  - Suggestions & Help Manager
    - HelpTicket Manager

- [R17]: The system allows a policy maker to visualize all the production analytics in output from the classification algorithm
  - Production Manager
    - Analytics Manager

- [R18]: The system does not allow the farmer to see the analytics, but his performance grades only
  - Production Manager
    - Analytics Manager

- [R19]: The system notifies a farmer every time there is an activity in a forum discussion he follows
    - Forum Manager
        - Discussion Manager

- [R20]: A farmer that creates or adds an answer to an already existing forum discussion, automatically starts following it
    - Forum Manager
        - Discussion Manager

- [R21]: The system allows a farmer to follow an already existing forum discussion even without adding an answer to it
    - Forum Manager
        - Discussion Manager
- [R22]: The system allows only a policy maker to establish a Direct Connection; A Direct Connection can be established only between two farmers at a time
    - Supervision services Manager
        - FarmersProfiles Manager
        - Direct Connection Manager

- [R23]: The system allows a policy maker to directly reply to a HelpTicket
    - Suggestions & Help Manager
        - HelpTicket Manager

- [R24]: The system allows a policy maker to visualize all the generic production data of each of the farmers under his supervision
    - Supervision services Manager
        - FarmersProfiles Manager

- [R25]: A policy maker can comment on farmers performances grades calculated by the algorithm
    - Production Manager
        - Analytics Manager

- [R26]: When a group chat is created, the system automatically names it with the pair of farmers names
    - Supervision services Manager
        - Direct Connection Manager

5. Implementation, Integration and test plan

Implementation Plan

In order to implement and test our system in a way that is consistent it is chosen to implement the components separately. To better achieve this separation, a bottom-up approach is used for the implementation, the integration and the test as well.

Based on the subdivisions highlighted in views of the sections above, the system to be implemented can be assumed to be the following:

- Client Application
- Server software
- DBMS

The first components ever to be implemented are those related to the Database. This will allow a more practical and efficient gestion from DREAMS.

About the server software, its first component to be implemented will the Production Manager component; this one is further composed of four subcomponents: SowData Manager and ReapData Manager will be implemented in parallel but after Field Manager, since the first two depend on the last one. Analytics Manager, the fourth subcomponent will be implemented last; indeed, this component is charged with providing rankings and evaluations from the production data.

The second component of the server software to be implemented is the Suggestions & Help Manager component along with its subcomponents.

The Supervision Services component can be implemented either at the same time or after dependingly

The last components of the server software are the Forum Manager and the Authentication Manager: the first one is a kind of standalone component; therefore, it could also be implemented parallel to the others; the second one instead is implemented last not for order of importance, but difficulty and its hidden meaning of “closing the implementation” of the server side of the system.

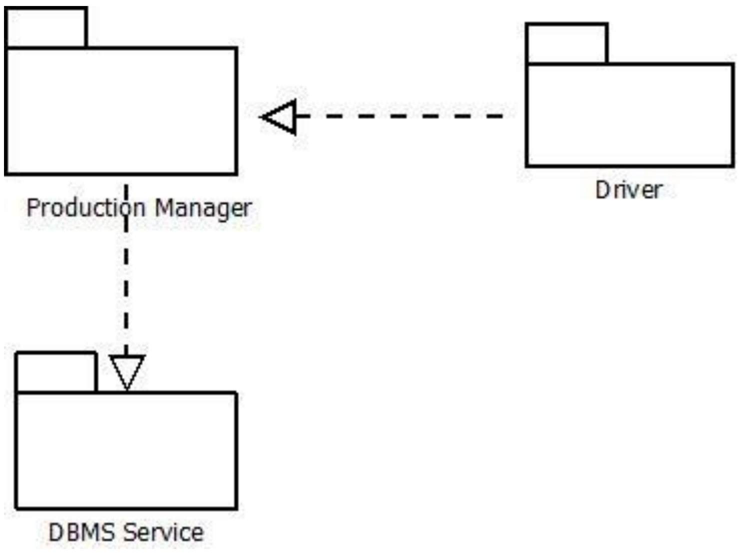
Then the client application can be implemented: both sequential and parallel implementation of these components are allowed.

Integration Plan

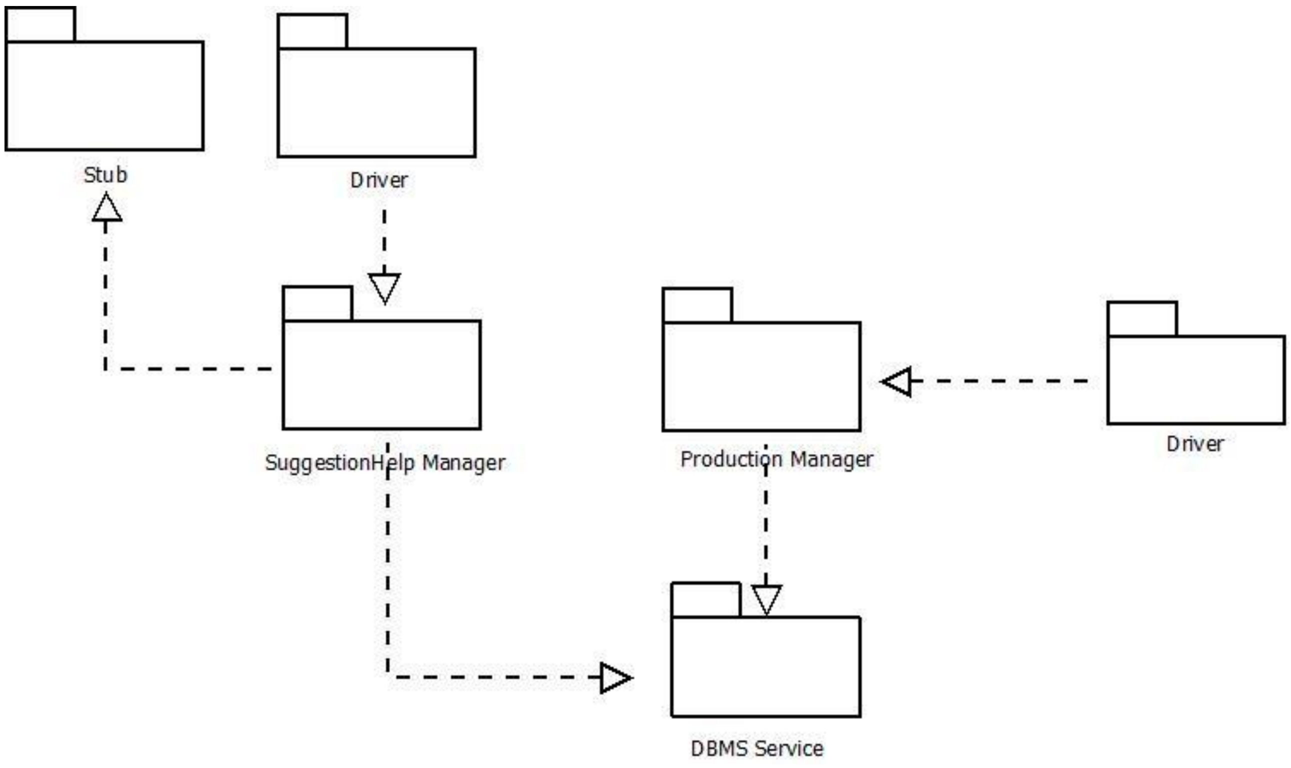
The integration of the different components is divided into phases as well.

Here, Unit testing should be done before integrating a component with the rest of the already developed system, in order to ensure that the offered functionalities work correctly in an isolated testing environment.

After the integration, an integration test should follow, to guarantee the correctness of the system at each integration step. First, we show the integration of the main component, the ProductionManager.

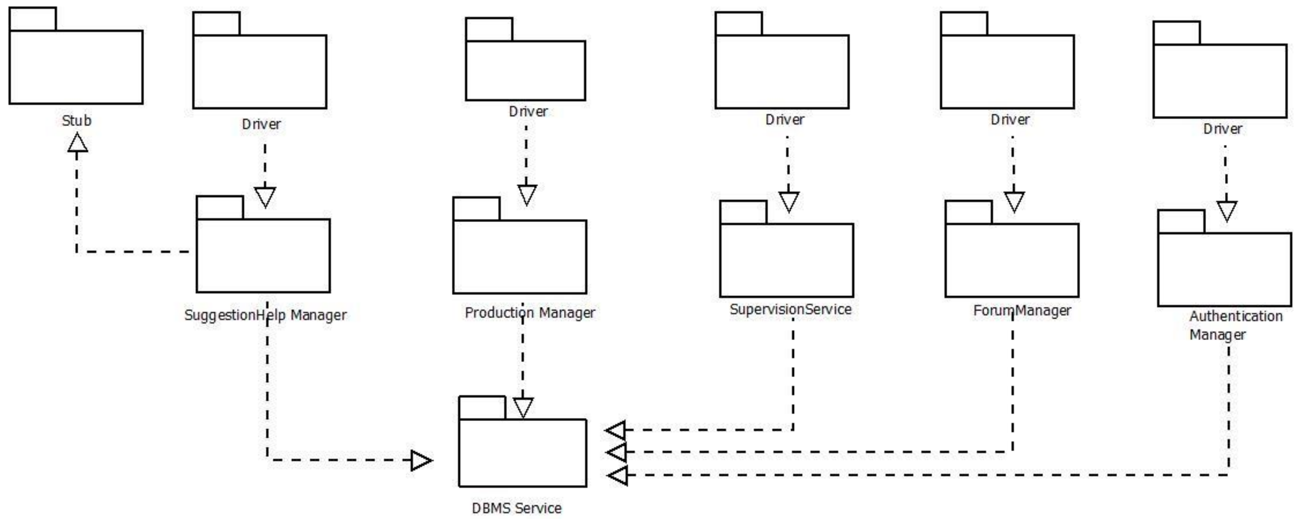


We see that it is directly linked with the ProductionManager and uses a Driver: that is a module that replaces the calling modules, in order to test the module. Soon after we are going to integrate the second module, the SuggestionHelpManager:



In this case we need both a Driver and a Stub, that is a dummy module, not yet implemented, which allows us to evaluate and test the HelpTicket and Direct Connection system.

Finally, we implement everything else.



6. Effort Spent

Student 1 (anonymous)

Topic	Hours
General Reasoning	**3:00h**
Component View	**5:00h**
Mockups	**6:00h**
Deployment View	**4:30h**
Document Organization	**4:00h**
Requirement Traceability	**5:30h**

Student 2 (anonymous)

Topic	Hours
General Reasoning	**4:00h**
Component View	**12:00h**
Component Interfaces	**9:30h**
Runtime View	**4:30h**
Document Organization	**4:00h**
Other design decisions	**5:00h**
Implementation, integration and test plan	**5:30h**

7. References

All the diagrams have been made with DIA.