POLITECNICO
MILANO 1863

Software Engineering 2 Project
Academic Year 2021 - 2022

DREAM

Design Document

Valeria Detomas Sofia Martellozzo

Professor

Elisabetta Di Nitto

Version 1
December 31, 2021

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 is provided an high view of the architecture of the system, that is structured following the three logic layer:

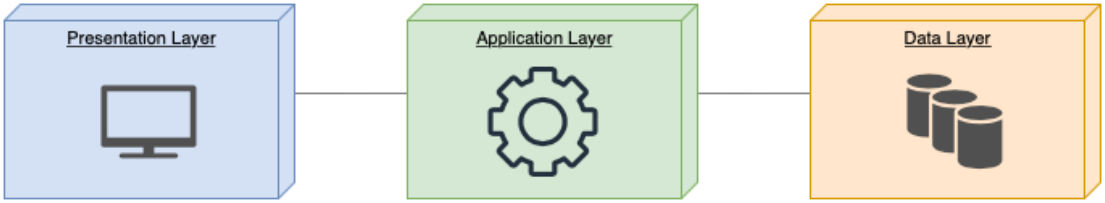

Figure 1: three layer architecture

- **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.

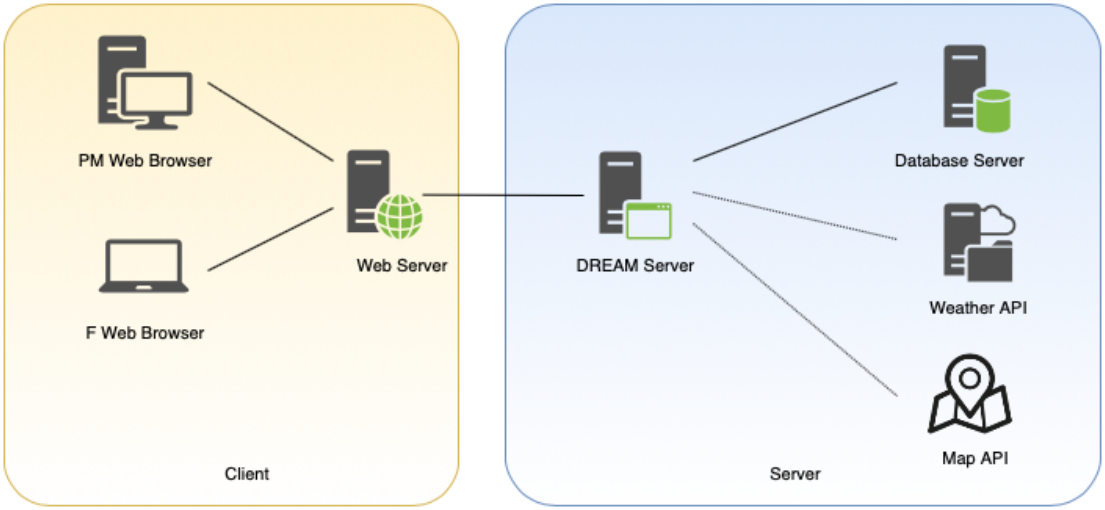

Figure 2: Dream system diagram.

- **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 is provided a description of the components and interfaces of the system, how they are organised internally and how they comunicate with each other.

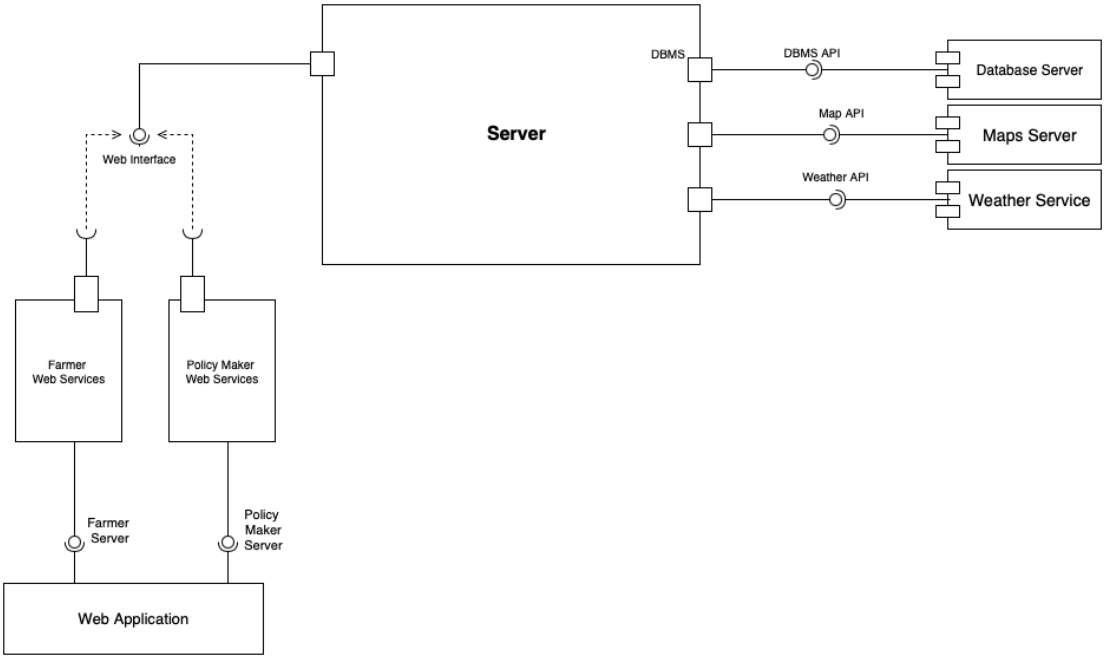

Figure 3: Component view.

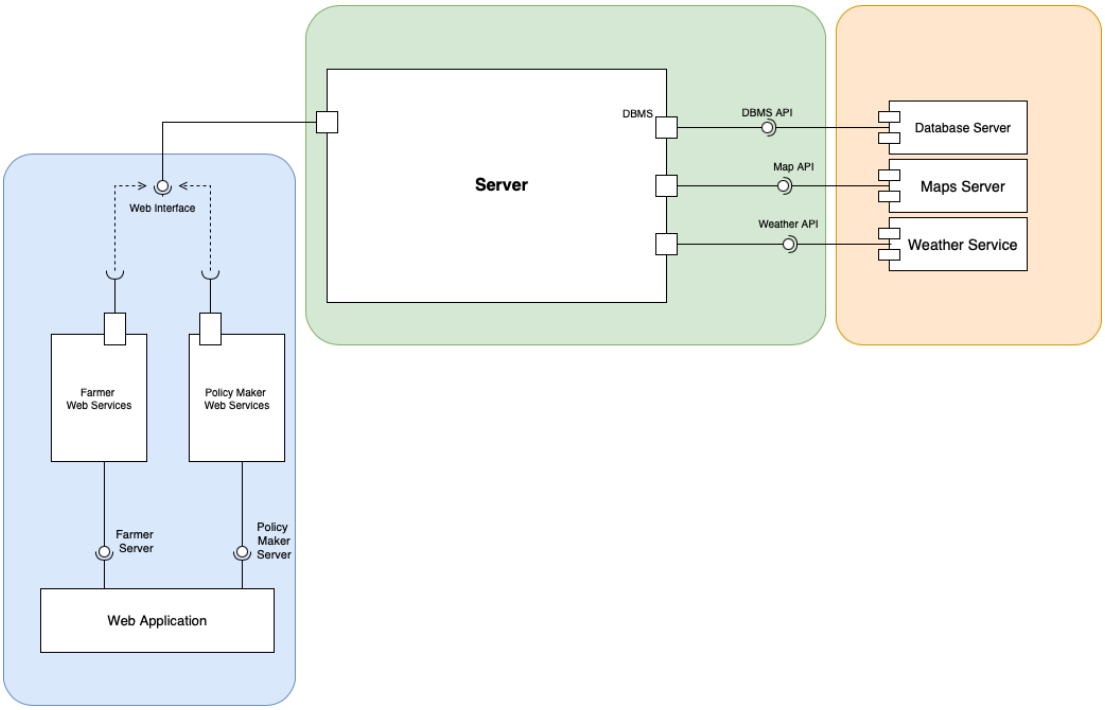

Figure 4: Component view.

2.3 Deployment view

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
2. f/pm login
3. f send msg on forum
4. f submit production data
5. f/pm find farm on the map

6. f/pm visualize farm info (page)
7. submit notification (f: help, advice — pm: solution)
8. f/pm visualize notificationpm ipdate the map (evaluation)

2.5 Component interfaces

2.6 Architectural styles and patterns

2.7 Other design decisions

3 User Interface Design

3.1

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