SYSTEM ARCHITECTURE

PLATFORM OF CLIMATIC AND AGROCLIMATE PROGNOSTICS

PRODUCT # 1 - USAID

WRITTEN BY:

STEVEN SOTELO

DECISION AND POLICY ANALYSIS - DAPA

THE INTERNATIONAL CENTER FOR TROPICAL AGRICULTURE - CIAT

CALI

2017

# Introduction

The development of this product begins with the preparation of requirements. Keeping track of these and giving them a reach is a decisive factor in the success of the project. These requirements were built in conjunction with the scientists, producers of the zones, professional technicians who assist to farmers and partnerships. In this system, the needs and feasibility were taken into account for each of the requirements in the development of the project. The usability of the user was evaluated in two important points that allowed observing that the system does greatly help in the assimilation of the information.

The development of the system had different releases that allowed those interested to observe the progress of the project. A continuous feedback of the progress of the project and the development of the platform, consolidated concepts, which initially were not very clear. The cooperation between the partnerships and CIAT, creating a system that can be adopted by each partnership to support their decision making.

This contributed to the design of an architecture that can be scalable and sustainable over time.

# System architecture

The system is the integration of different software components focused on providing information through climate services. The union of these software packages in a common environment and communicated by means of interfaces give robustness to the proposed solution to the automation of generation of forecast and visualization of the data. The approach of this architecture is the result of the analysis of the requirements obtained in different stages of the project, at each moment trying to adjust to the expectations of all users. Each part of the system has a goal that contributes to the common goal, by communicating with the other components; this objective is to give information to users. The design of each component is aimed at creating a solution that can be scalable and that can be maintained over time.

The architecture of the system can be analyzed from different points. The layer model view allows you to observe in a vertical way how the solution is built and how these components communicate. The component diagram view allows you to observe each software package that is built and its relationship with others. Finally, there is the view from a deployment model, which shows how communication between different parts of the system.

The support of the system is a persistent database implemented in the Database Management System (SGBD) Mongo, we can say then that it is the first layer of the system. The second layer extracts the information from the database, offering services to the other components that need to obtain them. The third layer is not abstract like the previous ones, but are applications with specific purposes inside the system: Forecast App, is a console application for the import and export of information from the database through text files, WebAdmin is an application web that allows the administration of the platform, while Web API is a web service that exposes the information to the internet. In the fourth layer are the client applications that use the services exposed by the lower level, this is the case of the Website (Viewing Site) and the forecast generator. In the uppermost layer are the software modules that allow the platform to be scalable.

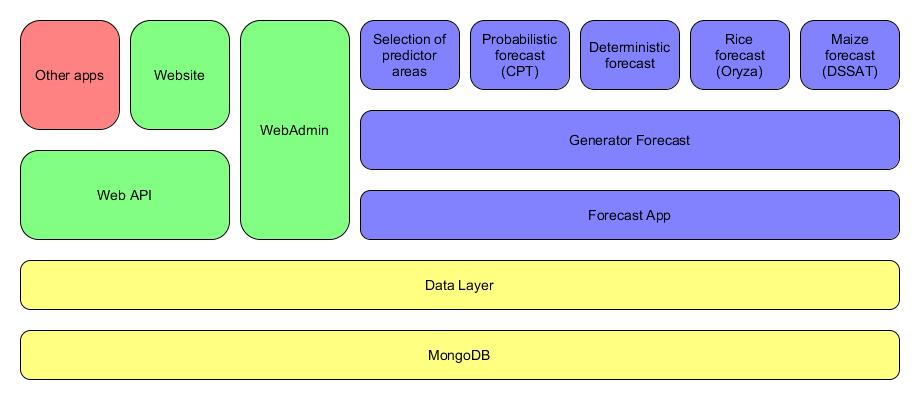


Figure 1. System architecture - System layers

Software packages are pieces of code that have a specific objective within the system. The development was created in components to simplify the complexity of the entire system, allowing easier maintenance. The interaction they have constitutes a robust platform. Each component can be replaced by some other code portion or an application, only if they retain the same communication interfaces; this allows the system to be scalable for future improvements. Here is the description of each component and its objective:

* • Mongo DB: It is a non-SQL persistent database. It is responsible for storing the necessary data for the platform. It does not store configuration files that may be required by other components (for example, files required by Oryza, DSSAT or CPT), but it does record the locations where they are located.
* Forecast.Data: It is a library in the form ORM[[1]](#footnote-1) in .Net Core. Este se encarga de realizar la conexión con la base de datos para extraer y enviar información. It is responsible for establishing the connection with the database to extract and send information. This is not considered an application; it is a library DLL[[2]](#footnote-2).
* Forecast.WebAdmin: It is an ASP.NET Core website. This website allows the administration of parameters that affect the entire system. Some of the functionalities are the administration of historical climate data, performance and parameterization of the execution of the forecast, allows the management of weather stations, cultivars and soils for predictions.
* Forecast.WebAPI: It is a WebAPI web service in .Net Core. This service exposes the data of the platform, both forecasting and historical information. This service does not have information capture functions. The data is displayed in two formats, depending on the needs of each user, it offers data in CSV and JSON format.
* Forecast.Web: It is an ASP.NET Core website. This website allows the display of forecast information and historical data. It is a website designed for the end user, so that the user can obtain as much information as possible in different ways.
* Forecast.ForecastApp: It is a .Net Core console application. This application allows you to insert and extract data from the database by making calls through the command console. Serves communication between the R and .Net Core environments.
* Run main: It is a script in the language of R. This script is a driver of the generation of forecasts. The main objective is to give an order to the inputs and outputs of the models used in the generation of forecasts.
* Rice model: It is a script in the language of R. This script is responsible for carrying out the agroclimatic forecast of rice based on climatic scenarios, cultivars, soils and agronomic management. It uses the Oryza 2000 application as its main tool.
* Maize model: It is a script in the language of R. This script is responsible for carrying out the agroclimatic forecast of maize based on climatic scenarios, cultivars, soils and agronomic management. It uses the DSSAT application as its main tool.
* Selection of predictor areas: It is a script in the language of R. This is responsible for the selection of predictive areas for the generation of climate forecast. The areas are obtained based on the sea surface temperature (SST) and some additional parameters.
* Probabilistic forecast: It is a script in the language of R. This is responsible for the generation of climate forecast. The tool for generating these predictions is called CPT Batch.
* Deterministic forecast: It is a script in the language of R. This is responsible for generating climate scenarios according to the probabilities of the weather forecast and historical data. It is the main input for each crop model.

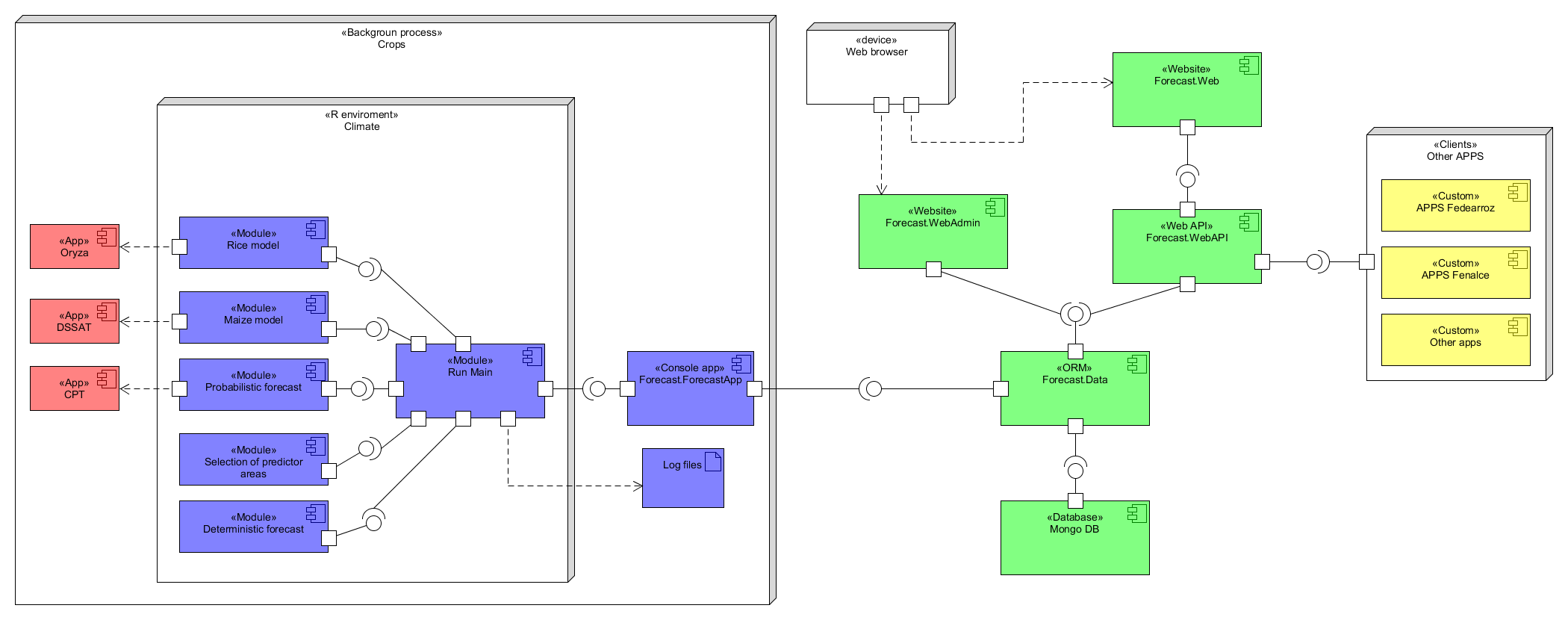


Figure 2. System architecture - Components diagram

In the deployment view we can see how the system behaves when the components communicate with each other. It is possible to observe what protocols and what additional tools are used throughout the system. The requirements that the whole platform has regarding the required infrastructure and its possible ways of implementation can be dimensioned. The additional technologies used by each component of the system are detailed in this diagram. It allows to observe some requirements that cannot be seen from the other edges of the system.

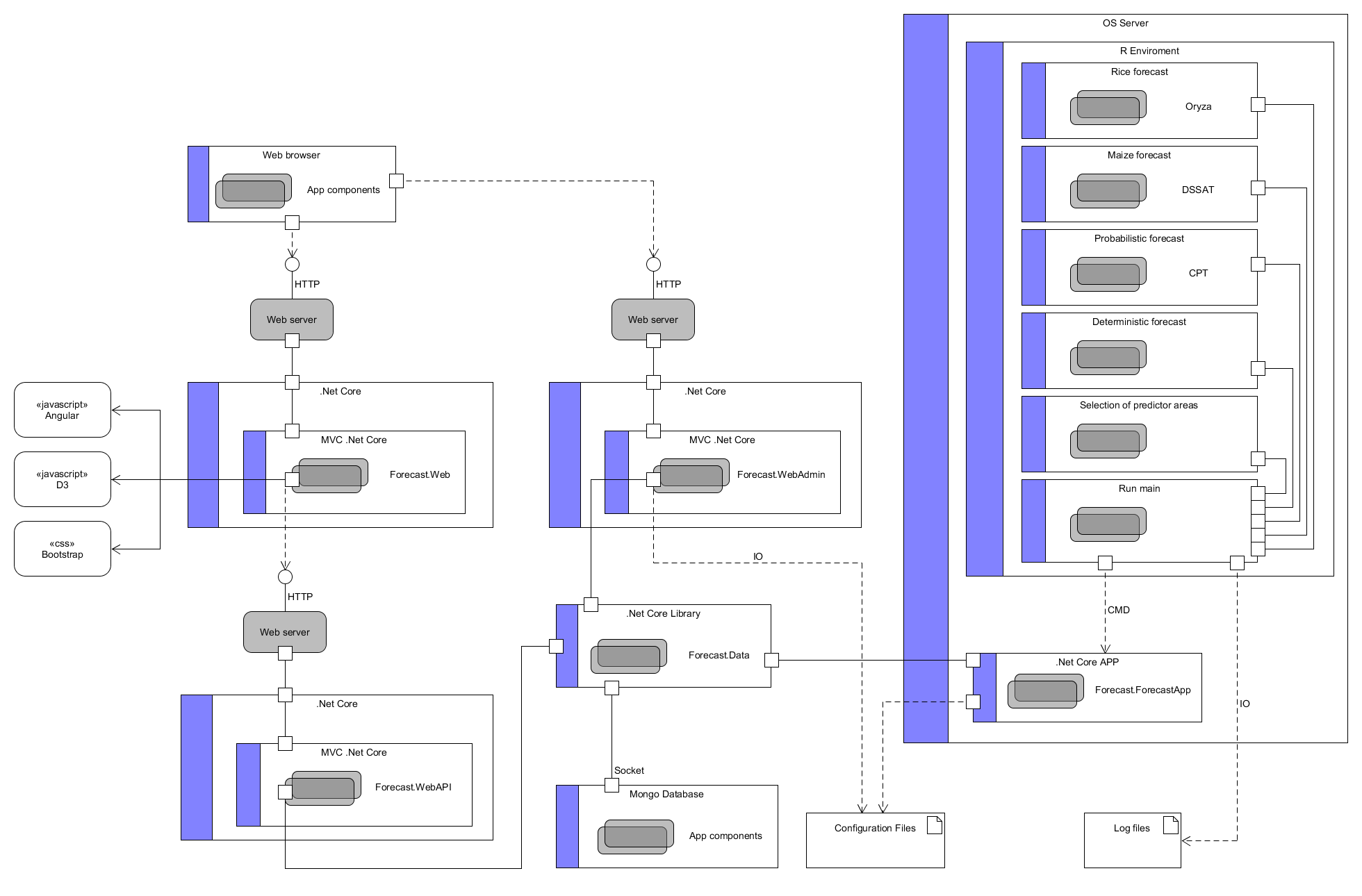


Figure 3. System architecture - Deployment and deployment diagram

# Database

The database provides information on the climatic and agroclimatic forecasts of the platform; it also contains the historical climate information of the climatic seasons and modeled data of the crops. The database has been modeled for the Mongo Database engine, which means that the entities do not constitute related tables, but NOSQL data entities that are stored in collections. In Figure 2, the relational model of the data can be observed.

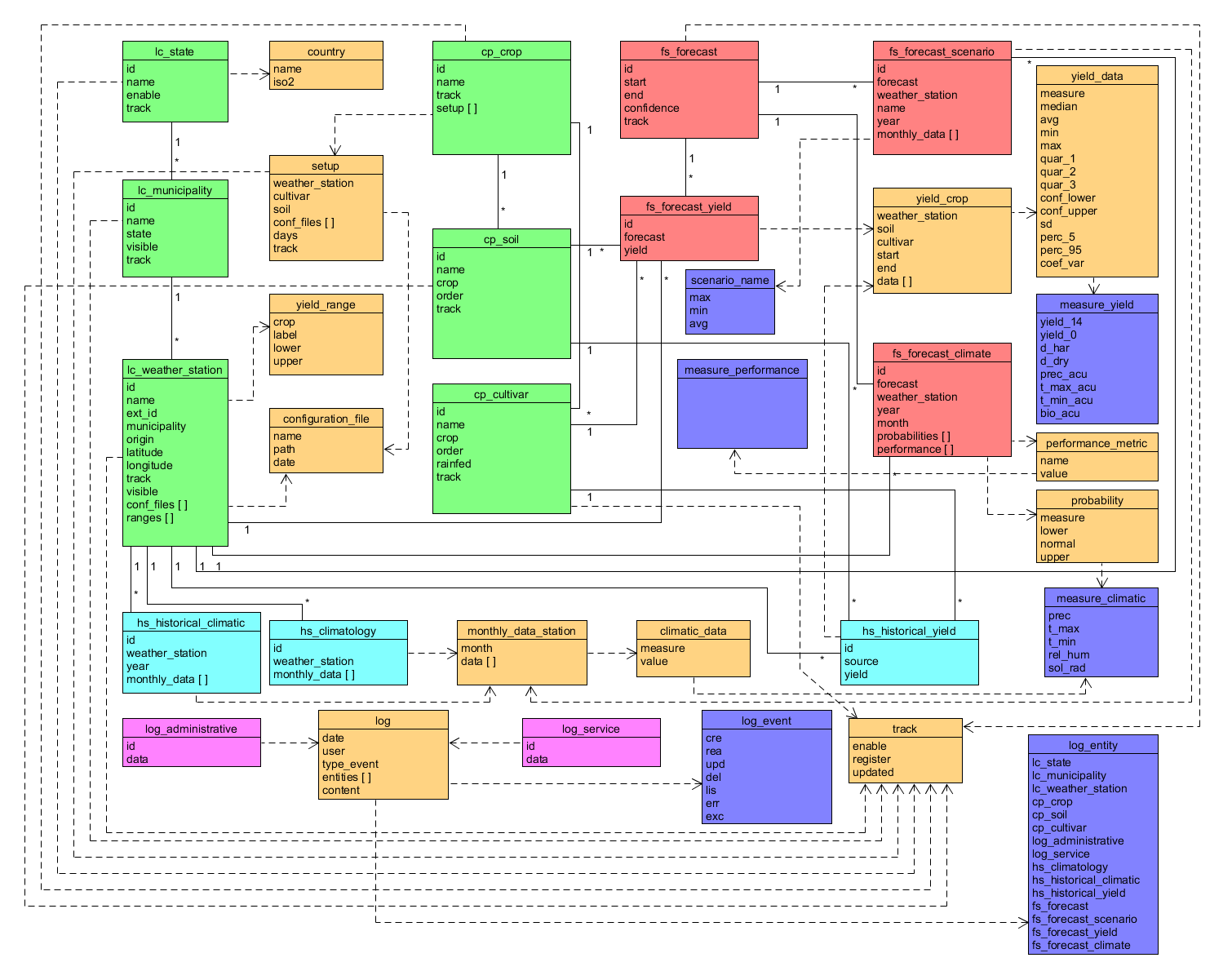


Figure 4 Database - Relational Model

The data dictionary can be found in the repository of the source code.

# Repositories of codes and additional documentation

The source codes of the entire system are stored on the Github platform. There are different repositories within this page:

* <https://github.com/CIAT-DAPA/usaid_forecast_web>
* <https://github.com/CIAT-DAPA/usaid_procesos_interfaz>
* <https://github.com/CIAT-DAPA/usaid_forecast>

Additional documentation such as user manuals, installation manuals, design details, data dictionary, can be found at the following link:

<https://github.com/CIAT-DAPA/usaid_forecast_web/tree/master/docs>

1. ORM – Object-relational mapping [↑](#footnote-ref-1)
2. DLL – Dynamic-link library [↑](#footnote-ref-2)