ARQUITECTURA DEL SISTEMA

PLATAFORMA DE PRONÓSTICOS CLIMÁTICOS Y AGROCLIMÁTICOS

PRODUCTO #1 - USAID

ESCRITO POR:

STEVEN SOTELO

DECISION AND POLICY ANALYSIS - DAPA

CENTRO INTERNACIONAL DE AGRICULTURA TRÓPICAL - CIAT

CALI

2017

# Introducción

El desarrollo de este producto empieza con el levantamiento de requerimientos. Llevar un control de estos y darle un alcance es un factor decisivo en el éxito del proyecto. Estos requerimientos se construyeron en conjunto con los científicos, productores de las zonas, técnicos profesionales que brinda asistencia a los agricultores y gremios. En este sistema se tuvo en cuenta las necesidades de cada usuario, teniendo en cuenta la factibilidad de cada uno de los requisitos en la duración del proyecto. Se evalúo la usabilidad del usuario en dos puntos importantes del proyecto que permitieron ver que el sistema ayuda en gran medida en la asimilación de la información.

El desarrollo del sistema tuvo distintas versiones en forma de spring que permitieran ver a los stakeholders los avances del proyecto. Mediante retroalimentación continua sobre el avance del proyecto y el desarrollo de la plataforma se consolidaban conceptos que inicialmente no fueron muy claros al inicio. La cooperación entre entidades gremiales y CIAT, permitió crear un sistema que pueda ser adoptado por cada gremio para apoyar su toma de decisiones.

Lo anterior contribuyo en el diseño de arquitectura que pueda ser escalable y sostenible con el tiempo.

# Arquitectura del sistema

El sistema es la integración de distintos componentes de software enfocada en brindar información mediante servicios climáticos. La unión de estos paquetes de software en un ambiente en común y comunicados mediantes interfaces le da robustez a la solución planteada a la automatización de generación de pronóstico y visualización de estos datos. El planteamiento de esta arquitectura es el resultado del análisis de los requerimientos obtenidos en distintas etapas del proyecto, cada vez tratando de ajustarse a las expectativas de todos los usuarios. Cada parte del sistema tiene un objetivo propio que contribuye en objetivo en común al comunicarse con los otros componentes, este objetivo es el de dar información a los usuarios. El diseño de cada componente está direccionado en crear una solución que pueda ser escalable y permita mantenerse en el tiempo.

La arquitectura del sistema puede ser analizada desde distintos puntos de vista. La vista de modelo de capas permite observar de una forma vertical cómo está construida la solución y como se comunican sus componentes. La vista de diagrama de componentes permite ver cada paquete de software que está construido y su relación con los demás. Por último esta la vista desde un modelo de despliegue, que enseña cómo es la comunicación entre las distintas partes del sistema.

El cimiento del sistema es una base de datos persistente implementado en el Sistema Gestor de Base de Datos (SGBD) Mongo, se puede llamar a esta la primera capa del sistema. La segunda capa, extrae la información de la base de datos; ofreciendo servicios a los demás componentes que deseen obtenerlos. La tercera capa ya no es abstracta como las anteriores, sino que son aplicaciones con fines específicos dentro del sistema: Forecast App es una aplicación de consola para la importación y exportación de información de la base de datos mediante archivos planos, WebAdmin es una aplicación web que permite la administración de la plataforma, mientras que Web API es un servicio web que expone la información hacía internet. En la cuarta capa encontramos aplicaciones clientes que usan los servicios expuestos por el nivel inferior, es el caso del Website (Sitio de visualización) y el generador de pronósticos. En la capa más superior encontramos los módulos de software que permiten que la plataforma sea escalable.

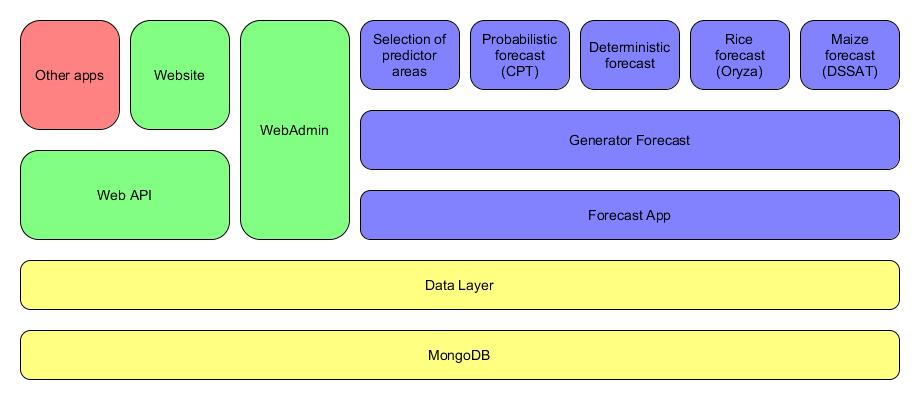


Figure 1 Arquitectura del sistema - Capas del sistema

Los paquetes de software son porciones de código que tienen un objetivo específico dentro del sistema. El desarrollo se realiza por componentes con el fin de simplificar la complejidad de todo el sistema, haciéndolo así más fácil de mantener en el tiempo. La interacción que tienen estos son los que constituyen a una plataforma robusta. Cada componente puede ser reemplazado por alguna otra porción de código o aplicación, siempre y cuando conserven las mismas interfaces de comunicación; esto permite que todo el sistema sea escalable frente nuevas oportunidades de mejora. Veamos la descripción de cada componente y su objetivo:

* Mongo DB: Es una base de datos persistente No SQL. Esta es la encargada de almacenar los datos necesarios para plataforma. Esta no almacena archivos de configuración que puedan ser requeridos por otros componentes (por ejemplo archivos requeridos por Oryza, DSSAT o CPT), pero si las ubicaciones de donde se encuentran.
* Forecast.Data: Es una librería en forma ORM[[1]](#footnote-1) en .Net Core. Este se encarga de realizar la conexión con la base de datos para extraer y enviar información. No constituye una aplicación cómo tal, ya que es un DLL[[2]](#footnote-2).
* Forecast.WebAdmin: Es un sitio web ASP.NET Core. Este sitio web permite la administración de parámetros que afectan todo el sistema. Algunas de las funcionalidades son la administración de datos históricos tanto de clima, como de rendimiento y parametrización de la ejecución del pronóstico, permite la gestión de estaciones climatológicas, cultivares y suelos para las predicciones.
* Forecast.WebAPI: Es un servicio web WebAPI en .Net Core. Este servicio expone los datos de la plataforma, tanto de pronósticos, como de información histórica. Este servicio no tiene funciones de captura de información. Los datos los expone en dos formatos, dependiendo de las necesidades de cada usuario, ofrece datos en formato CSV y JSON.
* Forecast.Web: Es un sitio web ASP.NET Core. Este sitio web permite la visualización de información de pronósticos y datos históricos. Es un sitio web orientado a usuario final, con el objetivo de permitirle a este obtener la mayor información posible de distintas maneras.
* Forecast.ForecastApp: Es una aplicación de consola .Net Core. Esta aplicación permite insertar y extraer datos de la base de datos mediante llamados por medio de la consola de comandos. Sirve como punto de comunicación entre los entornos de R y .Net Core.
* Run main: Es un script en R. Este script es un orquestador de la generación de pronósticos. Su principal objetivo es darle un orden a las entradas y salidas de los modelos utilizados en la generación de pronósticos.
* Rice model: Es un script en R. Este script es el encargado de realizar el pronóstico agroclimático de arroz con base a unos escenarios climáticos, cultivares, suelos y manejo agronómico. Este usa como herramienta principal la aplicación Oryza 2000.
* Maize model: Es un script en R. Este script es el encargado de realizar el pronóstico agroclimático de maíz con base a unos escenarios climáticos, cultivares, suelos y manejo agronómico. Este usa como herramienta principal la aplicación DSSAT.
* Selection of predictor areas: Es un script en R. Este se encarga de la selección de áreas predictoras para la generación del pronóstico climático. Las áreas las obtiene con base a la temperatura superficial del mar (TSM) y algunos parámetros adicionales.
* Probabilistic forecast: Es un script en R. Este se encarga de la generación de pronóstico climático. Su herramienta para la generación de estas predicciones es CPT Batch.
* Deterministic forecast: Es un script en R. Este se encarga de la generación de escenarios climáticos de acuerdo a las probabilidades del pronóstico climático y los datos históricos. Es el insumo principal para cada modelo de cultivo.

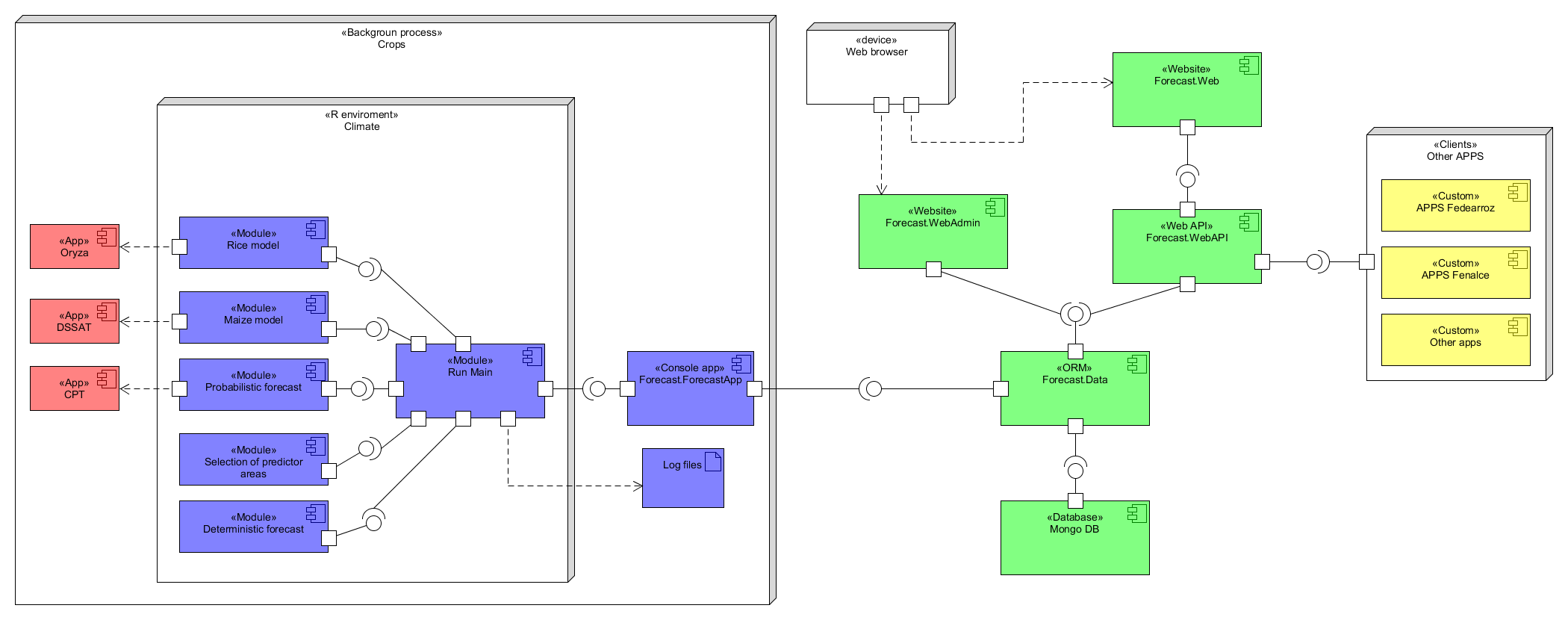


Figure 2 Arquitectura del sistema - Diagrama de componentes

En la vista de despliegue podemos ver como es el comportamiento del sistema cuando los componentes se comunican entre sí. Este nos permite observar mediante que protocolos y que herramientas adicionales son usadas en todo el sistema. Con esto dimensionamos los requisitos que tiene toda la plataforma en cuanto a la infraestructura requerida y sus posibles formas de implementación. Las tecnologías adicionales que son usadas por cada componente del sistema se detallan en este diagrama. Esto permite ver algunos otros requisitos que no se pueden ver desde las otras aristas del sistema.

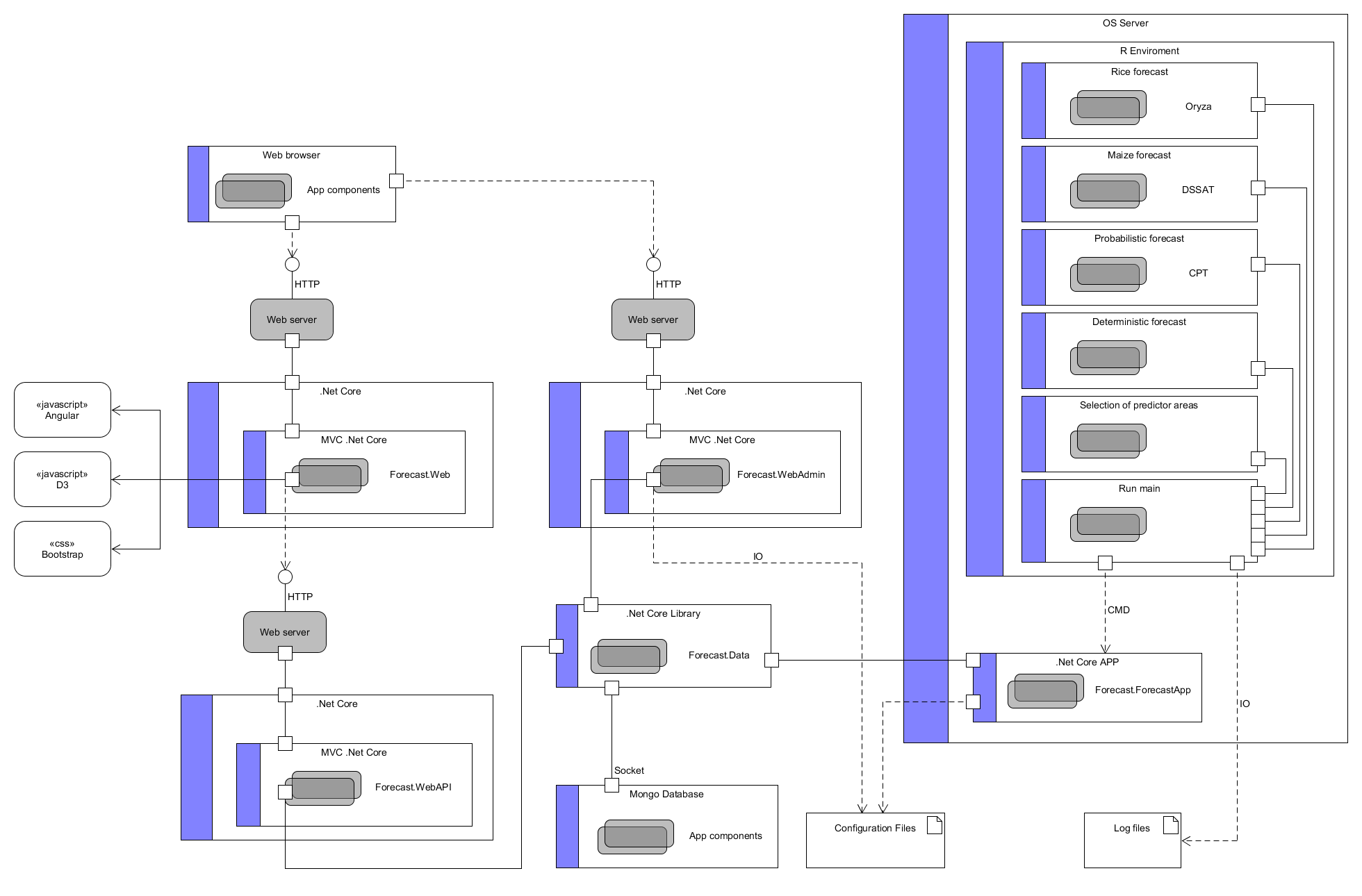


Figure 3 Arquitectura del sistema - Diagrama de implementación y despliegue

# Base de datos

Esta base de datos provee la información de los pronósticos climáticos y agroclimáticos de la plataforma, también contiene la información histórica de clima de las estaciones climáticas y datos modelados de los cultivos. La base de datos ha sido modelada para el motor de [Mongo Database](https://www.mongodb.com/), lo que quiere decir que las entidades no constituyen tablas relacionadas entre sí, sino entidades de datos [NOSQL](https://www.mongodb.com/nosql-explained) que son almacenadas en colecciones. En la Ilustración 2 se puede observar el modelo relacional de los datos.

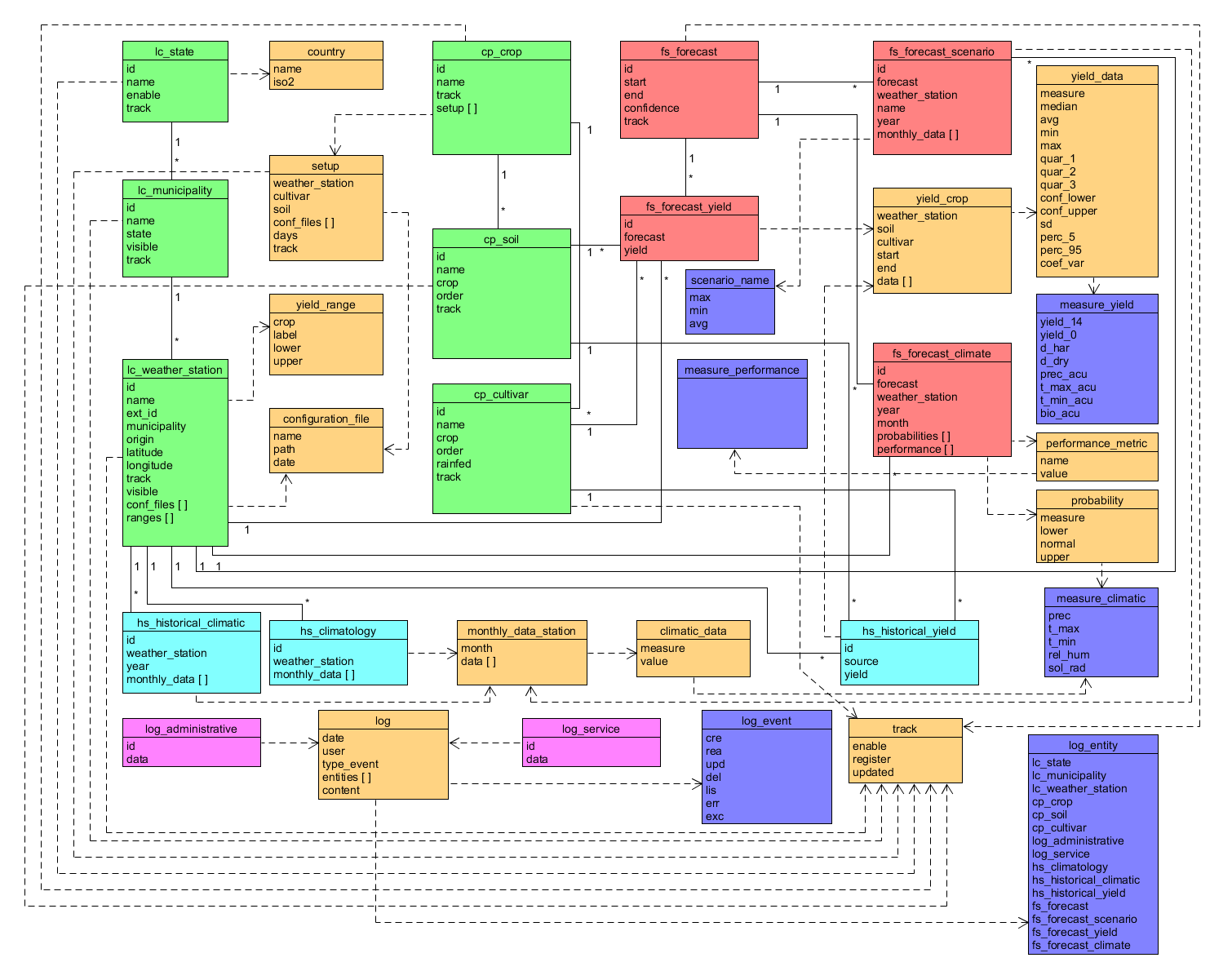


Figure 4 Base de datos - Modelo Relacional

El diccionario de datos se puede encontrar en el repositorio del código fuente.

# Repositorios de códigos y documentación adicional

Los códigos fuentes de todo el sistema se encuentran almacenados en la plataforma Github. Se encuentran en distintos repositorios dentro de esta página:

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

La documentación adicional como manuales de usuario, manuales de instalación, detalles de diseño, diccionario de datos, las puede encontrar en el siguiente enlace:

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

1. ORM – Mapeo Objeto-Relacion (por su nombre en inglés, Object-relational mapping) [↑](#footnote-ref-1)
2. DLL – Biblioteca de enlace dinámico (por su nombre en inglés, Dynamic-link library) [↑](#footnote-ref-2)