

**Software Development Processes**

***ONE CGIAR - Big Data Platform***

**Compiled by**

|  |  |
| --- | --- |
| *Juan C. Cadavid* | *Full Stack Developer* |
|  |  |

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Reviewer** | **Description of the change** | |
| 1.0 | 11/03/2022 | Juan Carlos Cadavid |  | Initial version |

# INTRODUCTION

In software engineering, a software development process is the process of dividing software development work into smaller, parallel, or sequential steps or subprocesses to improve design, product management. It is also known as a software development life cycle (SDLC).

The purpose of this document is to guide stakeholders in each of the software development processes implemented to solve the problem posed.

Below you can see the Process Map that will be implemented in the One CGIAR platform solution - Big data Platform.

Diagram

Description automatically generated

Figure 1 - Process Map

# Software requirements specification (srs)

## Context

The Alliance is one of the institutions that support the CGIAR Big Data Platform. Several groups from this platform are working on a proposal for a global Big Data system that allows small and medium-sized farmers to access a robust system, with different services, to help them manage their crops.

## Requirements Elicitation

Is the practice of researching and discovering the requirements of a system from users, customers, and other stakeholders.

### Functional Requirements

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Icon  Description automatically generated **User Story** | | | | | | |
| **Project's name** | ONCE -CGIAR Big Data Platform | | | | | |
| **Number** | E1 | | | | | |
| **Title** | MANAGEMENT OF CROP ACTIVITIES | | | | | |
| **Module** | Operations | | | | | |
| **Stage** | Prepared | Verified | Approved | Rejected | Built | Released |
| **Priority** | High | | | | | |
| **Release** | 1 | | | | | |
| **Effort (H/M)** |  | | | | | |
| **Rol** | Farmer | | | | | |
| **Requirement** | When the farmer accesses the system, it should show the option to manage activities and within this option the farmer will have the functionality to create a new activity in his crop and this option will ask for basic data such as:  1. The type of crop: It is a drop-down list with all the types of crops created in the system associated with the farmer.  2. The type of activity: It is a list with all the activities that the farmer must follow up on  3. The detail of the activity: It is a text type field where the farmer can relate the news of the activity that he is carrying out.  4. The evidence of the activity: It is a text type field where the URL of an image uploaded by the farmer will be saved.  5. The date of the activity: It is the date on which the data of the activity will be saved, this date is generated by the system.  Next, the crop activity management process will be described through a use case diagram: | | | | | |
| **Benefit** | The farmer will be able to relate each of the activities carried out in his crop | | | | | |

|  |  |
| --- | --- |
| **Elaborated:** | Juan Carlos Cadavid |
| **Approved:** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Icon  Description automatically generated **User Story** | | | | | | |
| **Project's name** | ONCE -CGIAR Big Data Platform | | | | | |
| **Number** | E2 | | | | | |
| **Title** | CROP ACTIVITY REPORT | | | | | |
| **Module** | Reports | | | | | |
| **Stage** | Prepared | Verified | Approved | Rejected | Built | Released |
| **Priority** | High | | | | | |
| **Release** | 1 | | | | | |
| **Effort (H/M)** |  | | | | | |
| **Rol** | Farmer | | | | | |
| **Requirement** | The farmer will find in the main panel of the application the reports option that when clicked will display two types of reports that will be described below:  1. Activities report: This report will show the information of each of the farmer's activities filtered by a range of dates.  For this report, the farmer will have the option to filter the type of crop and the type of activity carried out. This report must be a table whose content must be consistent with the information previously entered by the farmer.  Data Table: Type of crop, Type of activity, Detail of activity, Evidence of activity and date of activity.  2. Weather report: This report will allow the farmer to analyze the weather data collected by the weather station installed on the farm.  Data Table: Type of crop, temperature, humidity, date.  Next, the process of generating reports for the farmer will be described through a use case diagram. | | | | | |
| **Benefit** | The farmer will be able to consult the information of the activities carried out in his crop and the climatic information. | | | | | |

|  |  |
| --- | --- |
| **Elaborated:** | Juan Carlos Cadavid |
| **Approved:** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Icon  Description automatically generated **User Story** | | | | | | |
| **Project's name** | ONCE -CGIAR Big Data Platform | | | | | |
| **Number** | E3 | | | | | |
| **Title** | INTEGRATIONS | | | | | |
| **Module** | N/A | | | | | |
| **Stage** | Prepared | Verified | Approved | Rejected | Built | Released |
| **Priority** | High | | | | | |
| **Release** | 1 | | | | | |
| **Effort (H/M)** |  | | | | | |
| **Rol** | System | | | | | |
| **Requirement** | The System must have the ability to integrate with other subsystems that will be detailed below:   1. IOT weather Station: This system must collect data such as temperature and humidity and must expose them through a web service which will be consumed by the ONCE -CGIAR Big Data Platform system and after that this system will send the data to an ETL that will transform the information and insert it into the data warehouse. 2. ETL System: This system must make requests every minute to the ONCE -CGIAR Big Data Platform system, which must make a request to the meteorological station and thus return the climatic information so that the ETL System can carry out the extraction, transformation, and load in the data warehouse. | | | | | |
| **Benefit** | The System must have integration with the weather station and with the ETL system | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Icon  Description automatically generated **User Story** | | | | | | |
| **Project's name** | ONCE -CGIAR Big Data Platform | | | | | |
| **Number** | E4 | | | | | |
| **Title** | RECOMMENDATION MODELS AND BUSINESS INTELLIGENCE | | | | | |
| **Module** | BI | | | | | |
| **Stage** | Prepared | Verified | Approved | Rejected | Built | Released |
| **Priority** | High | | | | | |
| **Release** | 1 | | | | | |
| **Effort (H/M)** |  | | | | | |
| **Rol** | Researcher | | | | | |
| **Requirement** | When researchers enter the application, they must find a BI module where they must find a control panel that provides them with the necessary information to carry out the respective models of agronomic recommendations, these models must be stored in the system for later analysis by a scientist leader. | | | | | |
| **Benefit** | The System must provide a control panel with the statistics of all the information collected so that the researcher can create recommendation models. | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Icon  Description automatically generated **User Story** | | | | | | |
| **Project's name** | ONCE -CGIAR Big Data Platform | | | | | |
| **Number** | E5 | | | | | |
| **Title** | RECOMMENDATION EVALUATION MODULE | | | | | |
| **Module** | BI | | | | | |
| **Stage** | Prepared | Verified | Approved | Rejected | Built | Released |
| **Priority** | High | | | | | |
| **Release** | 1 | | | | | |
| **Effort (H/M)** |  | | | | | |
| **Rol** | Leader scientist | | | | | |
| **Requirement** | A leading scientist on the platform will be able to choose the best models and implement them in an agroclimatic service, which will be available so that each farmer can receive feedback based on their data. | | | | | |
| **Benefit** | The system provides the information of the agroecological models so that the leading scientist can evaluate them and create a service for the farmer. | | | | | |

### Nonfunctional Requirements

|  |  |
| --- | --- |
| **ID** | **NFR-01** |
| **Name** | Scalability |
| **Description** | it is thought on a global scale, but that its deployment will be on demand and progressively |
| **Input** |  |
| **Output** |  |
| **Priority** | Medium |

|  |  |
| --- | --- |
| **ID** | **NFR-02** |
| **Name** | Interoperability |
| **Description** | The system should be easily integrated with other subsystems |
| **Input** |  |
| **Output** |  |
| **Priority** | Medium |

# design model

In this section you will find the artifacts elaborated for the detailed design of the ONCE -CGIAR Big Data Platform software.

For the development of this test, two types of diagrams will be used to identify the process and facilitate understanding to carry out a cleaner coding process.

Note: On this occasion, due to time issues, only the detailed design of one user stories (***E1 -MANAGEMENT OF CROP ACTIVITIES***) will be carried out.

## Communication Diagram

Chart

Description automatically generated

Figure 2 - E1: Create Activity

## Sequence Diagram

Diagram

Description automatically generated

Figure 3 - E1: Create Activity

# architecture model

The development of the solution is designed to meet the non-functional requirements detailed in number 1.2.2. and to be able to meet the standards expected by the stakeholder.

Based on the above, the solution will have 3 main architectural patterns that will be detailed below:

**1. Microservices:** This architecture will be immersed in the entire application and will allow us to separate different functionalities that it must fulfill.

In the application you can find two backend servers, two frontend servers, an ETL system, a BI system and two Databases. The application will be distributed in each of these, decoupling various functionalities into independent parts.

**2. Clean Architecture:** This architectural pattern will be present in the backend of the microservices and will allow us to separate the framework from the business logic and will allow us to have independence from the data model.

**3. Model View Controller (MVC):** This architectural pattern will be present in the microservices of the application front since the framework that is going to be used is Angular.

## Deployment Diagram

The solution is designed based on the following components:

1. Frontend: The solution will have two fronts, one for the farmer and the other for researchers and scientific leaders.

On the farmer's frontend there will be options to manage activities related to crops and generate weather and historical information reports on activities and services offered by researchers.

In the frontend of the researchers and scientific leaders, you will find a BI dashboard and the options to make recommendations and to establish services.

1. Backend: The solution will have two backend servers designed for data management.

a. Application Server: This server is designed to offer a REST service that will be consumed by the Frontend and in turn will save the information directly in the data warehouse.

b. Application Server IOT: This server will be in charge of communication with the weather station and which in turn will have an exposed service to be consumed by the ETL that will trigger an order every minute to obtain information from the weather station.

1. ETL System: This system will oversee extraction, transformation, and loading in the data warehouse, this system will make a request to the Application Server IOT every minute to obtain the data from the weather station.
2. Power BI: There will be a BI system (Microsoft Power BI) that will be consuming the information through a Microsoft Data Gateway hosted on the ETL server of the secondary database and will have a table of historical information obtained in import mode and another table in Direct Query mode to process information in real-time.
3. Data warehouse: There will be two databases that will be replicated in real-time where there will be a main and a slave database, in the main database the actions related to the collection of information will be carried out and in the database of secondary data, BI processes will be carried out.

Next, the design of the application will be shown based on the parameters mentioned above.

Graphical user interface, diagram

Description automatically generated

Figure 4 - Deployment Diagram

[Link Deployment Diagram](https://drive.google.com/file/d/1ynTw1N_HNey5rWICkHN7U1vh_V3Htz6M/view?usp=sharing)

## Infrastructure Diagram

Diagram

Description automatically generated

Figure 5 - AWS Cloud Architectural Design

Note: As part of the solution, the software architect proposes to replace the use of containers in Docker with the use of server less infrastructure such as AWS Lambda.

# code

**BACKEND**

Both Backend was built thinking that the project would be easy to scale and that it would allow an efficient integration with other systems, for this reason, it was developed using the latest generation programming languages and technologies which will be described below:

1. **TypeScript:** Is a programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. TypeScript is designed for the development of large applications and transcompiles to JavaScript.

A picture containing text, clipart

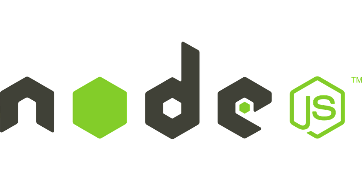
Description automatically generated

1. **JavaScript:** Is a programming language that is one of the core technologies of the World Wide Web (Internet), alongside HTML and CSS. Over 97% of websites use JavaScript on the client-side for web page behavior. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

Logo

Description automatically generated

1. **Node JS with the Express framework:** Is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.



To build the backend we use serverless framework which is a free and open-source web framework written using Node.js. Serverless is the first framework developed for building applications on AWS Lambda, a serverless computing platform provided by Amazon as a part of Amazon Web Services.



**Example request**

***Application Server***

Text

Description automatically generated

Functional Endpoint (AWS-Lambda) - <https://e8xgd8gi9c.execute-api.us-east-1.amazonaws.com/api/activities/registered>

***Application Server IOT***

Text

Description automatically generated

Functional Endpoint (AWS-Lambda) - <https://7weyk9znj0.execute-api.us-east-1.amazonaws.com/api/weather/information>

**FRONTEND**

The front of both proposed services is created using the Angular Framework.

Below you can see the appearance of the front of the application and the link to the functional page displayed in a Bucket S3

Graphical user interface, text, application

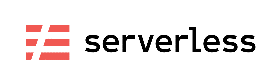
Description automatically generated with medium confidence

Functional Frontend (S3) - <http://one-cgiar-big-data.s3-website-us-west-1.amazonaws.com/wather>

# DEPLOY

The deployment of the project was carried out in the **AWS** cloud where the following components were configured:

1. Repository: <https://github.com/JuankCadavid/OneCGIARbigDataPlatform>
2. Backend: Two lambdas functions were created, each hosting a server with a REST API.



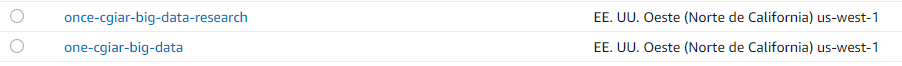
Graphical user interface, text, application, chat or text message

Description automatically generated Graphical user interface, text, table, email

Description automatically generated

* Application server : <https://e8xgd8gi9c.execute-api.us-east-1.amazonaws.com/api/activities/registered>
* Application server IOT : <https://7weyk9znj0.execute-api.us-east-1.amazonaws.com/api/weather/information>

1. Frontend: For the frontend, two S3 Buckets were created, each with the respective configuration to host static files.



* Frontend: <http://one-cgiar-big-data.s3-website-us-west-1.amazonaws.com/wather>

1. Data warehouse: An RDS with PostgreSQL was created to simulate the main DB

Host: onecgiarbigdata.ctwnnpfxxzig.us-east-1.rds.amazonaws.com

Graphical user interface, application

Description automatically generated