Alejandro Torre
Amaury Morales
Josué Torres

Final Project

Team #1. Toj Tu Yeetel.

Repository: https://github.com/alextorrer/TojTuYeetel_Project

Description

Toj Tu Yeetel (Healthy crop in Mayan language) is an information system for farmer people where they can track their crops, they can report on the pests and plagues that their crops are suffering, as well as view on a map and in a 'feed' different plague reports in greenhouses or nearby crops, so that they can take action before they can fall victim to a nearby plague and to prevent the economic loss that results from disposing of contaminated product.

Functional Requirements

#FR00.- The user can register in the system.

#FR01.- The user will be able to log in to the system.

#FR02.- The system should display in the feed reports sent by other users.

#FR03.- The system should display on a map reports sent by other users.

#FR04.- The user will be able to access the map from a report in the feed.

#FR05.- The user can add a crop to track it.

#FR06.- The user will be able to observe all the information of his crop.

#FR07.- The user may mark a crop as "harvested".

#FR08.- The user may make multiple reports of multiple plagues.

#FR09.- The system must have three sections (feed, map and crops) in the main panel.

#FR10.- The system must operate the database using Hibernate.

#FR11.- The system must use the Google Maps API to complete the #RF03.

#FR12.- The map will *center* the user's current location.

#FR13.- The system should show the locations of the user's registered crops.

Alejandro Torre
Amaury Morales
Josué Torres

Non-Functional Requirements

#NFR00.- The experience of users regarding the use of computer systems should be a "sufficient" level.

#NFR01.- Exceptions should be handled with JOptionPane's to inform the user.

#NFR02.- The system shall handle/create exceptions of the following types:

- Input values by the user.
- Database (JPA, JDBC).
- Google Maps API.

#NFR03.- The system should be considered to move to a mobile platform in the future.

#NFR04.- If the user does not get the current location, the midpoint between the crop locations will be used as the *center* on the map.

#NFR04.1.- In case of no registered crops, a standard point according to the time zone of the user's operating system will be used as a *center* on the map.

Evolution of Requirements

As the weeks passed away in the class, we realized that all FR and NFR could not be completely covered.

At the end, the covered FR were:

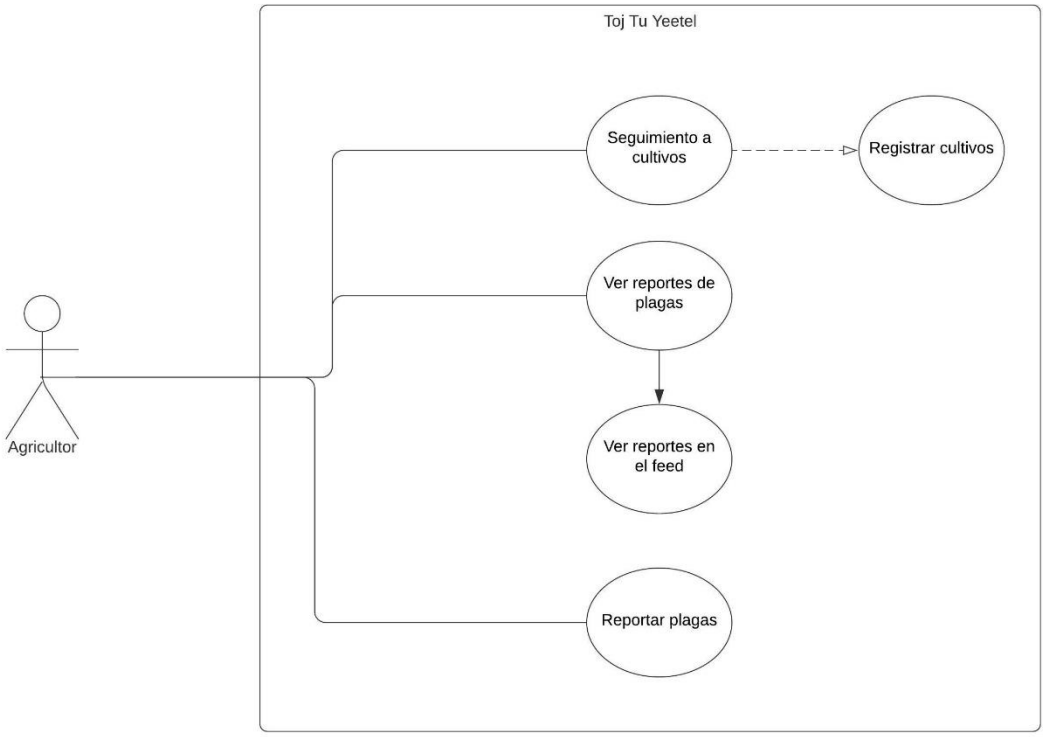
- #RF02.- The system should display reports sent by other users in the feed.
- #RF05.- The user can add a crop to track it.
- #RF06.- The user will be able to observe all the information of his crop.
- #RF08.- The user may make multiple reports of multiple plagues.
- #RF09.- The system must have three sections (feed, map and crops) in the main panel.
- #RF10.- The system must operate the database using Hibernate.

The covered NFR were:

- #RNF00.- The experience of users regarding the use of computer systems should be of a "sufficient" level.
- #RNF01.- Exceptions should be handled with JOptionPane's to inform the user.

- #RNF02.- The system must handle/create exceptions (User Entries, DB).

Use Case Diagram.



Use Cases

Name	CU-01 Display reports
Actor(s)	Farmer (user)
Description	The user will be able to view in the feed reports from other users.
Preconditions	The user must be logged in. The user must be in the 'reports' *tab.*
Normal flow	1. The user scrolls to view reports from other users.
Alternative flow	
Exceptions	1. If the user does not have an internet connection, it will show that the reports could not be loaded.

Name	CU-02 Create a report

Actor(s)	Farmer (user)
Description	The user will be able to make reports on found plagues.
Preconditions	The user must be logged in. The user must be in the *'reports' tab.* The user must have pressed the "Report Plague" button.
Normal flow	1. The user enters a title to the report. 2. The user selects a plague type from the report. 3. The user writes a plague description. 4. The user presses the "Send Report" button.
Alternative flow	1. The user presses the "Return" button to return to the main menu.
Exceptions	1. If the user does not have an internet connection when pressing the "Send Report" button, the error will be displayed to try later. 2. If the user does not complete all the required fields when presses the "Submit Report" button, the error will be displayed, and the user will be able to fill them.

Name	CU-03 Register crops
Actor(s)	Farmer (user)
Description	The user will be able to register crops to track them.
Preconditions	The user must be logged in. The user must be in the *tab* "My crops". The user must have pressed the "Add Crop" button.
Normal flow	1. The user enters a name for the crop. 2. The user selects a crop type. 3. The user defines the planting date of the crop. 4. The user defines the harvest date of the crop. 5. The user enters the location of the crop. 1. The user presses the "Register Crop" button.
Alternative flow	1. The user presses the "Return" button to return to the main menu.
Exceptions	1. If the user does not have an internet connection when pressing the "Register Crop" button, the error will be displayed to try later. 2. If the user does not complete all the required fields when presses the "Register Crop" button, the error will be displayed, and the user will be able to fill them.

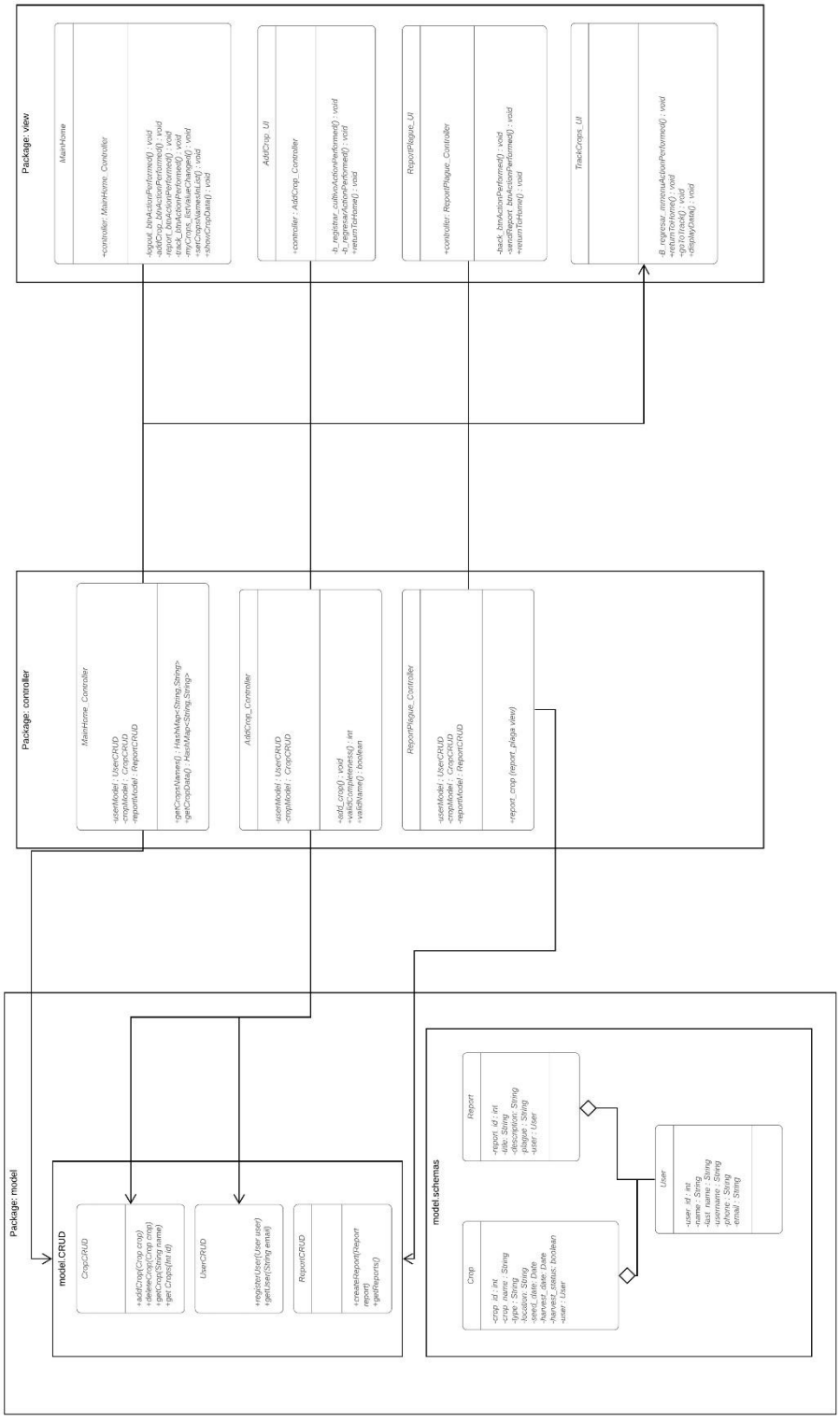
Name	CU-04 Crop tracking
Actor(s)	Farmer (user)

Alejandro Torre
Amaury Morales
Josué Torres

Description	The user can track and view more information about each one of his crops.
Preconditions	The user must be logged in. The user must have registered at least one crop. The user must be in the "My Crops" *tab.*
Normal flow	1. The user presses the "Track" button next to the crop name. 2. The user can observe the planting and harvest dates. 3. The user can observe a progress bar according to the harvest date.
Alternative flow	1. The user presses the "Track" button next to the crop name. 2. The user presses the "Return" button to return to the main menu.
Exceptions	If the user does not have an internet connection when pressing the "Track" button, the error will be displayed to try later.

Alejandro Torre
Amaury Morales
Josué Torres

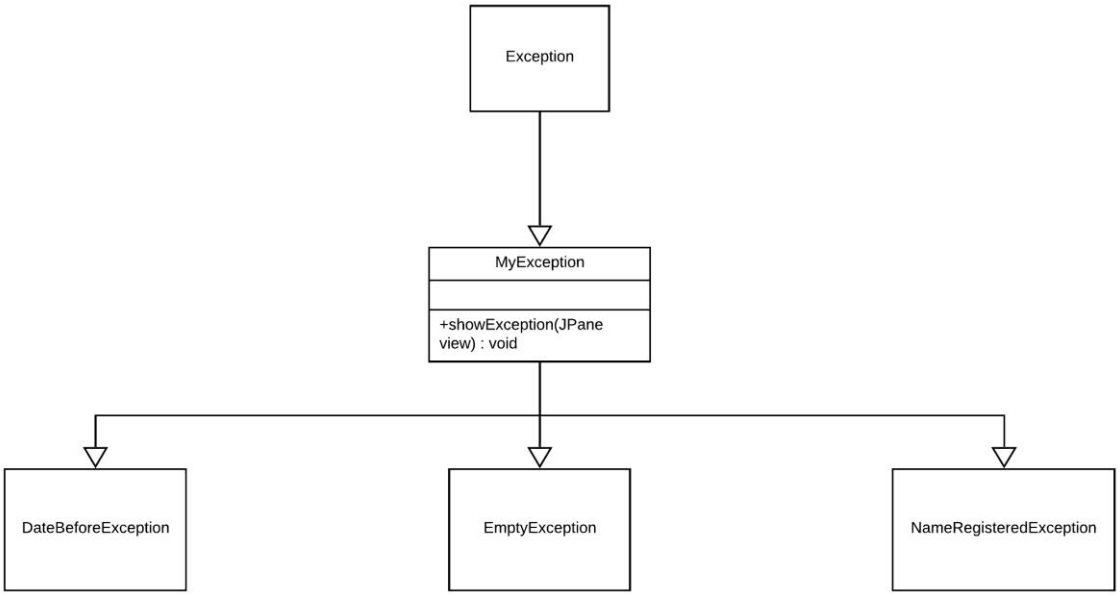
Class and MVC Diagram



As can be seen in the diagram above, 3 main packages corresponding to the MVC design pattern (Model, View, Controller) are created.

The layer corresponding to the view does not have reference to the model but does have a reference to the controller. The layer corresponding to the controller has model and view references (sent by parameter). The layer corresponding to the model does not have reference to any other layer.

The interaction between the Controller and View layers are mostly by a View reference that is received by the Controller methods. On the other hand, the connection between the Controller and the Model is just as simple as the Controller calls the Model methods.



Here, is shown how we handle the Controller Exceptions.

Development Process

To distribute tasks individually, first, we rely on the MVC design pattern, so each member was assigned an MVC layer to work with:

- Alejandro Torre (Model)

- Amaury Morales (Controller)
- Josue Torres (View)

In the table below, it can be shown the project distribution in different tasks with a different ponderation each one. We decide an arbitrary range of [1,3] based on complexity and duration.

Task	Responsible	Value	Product/Artifact
Create the repository	Alex	1	-
Create the main UI, segmented by tabs.	Josue	2	MainHome.java
Create the UI to add a new crop.	Josue	2	AddCrop_UI.java
Create a base form Pane.	Josue	1	Base_frame.java
Create the UI to report a plague.	Josue	1.5	ReportPlague_UI.java
Create the UI to track crops.	Josue	1.5	TrackCrops_UI.java
Handle the transitions in the different UI's.	Josue	5	MouseclickedEvent()
Valid the input information from the 'add new crop UI'.	Amaury	2	AddCrop_Controller.java
Valid the input information from the 'report a plague UI'.	Amaury	2	ReportPlague_Controller.java
Create Exceptions	Alex	2	Controller.exceptions
Show the exceptions as JOptionPane (adding a method to the Exceptions classes)	Alex	2	Exception classes with a showException() method.
Show the registered crops in the UI.	Alex	3	MainHome.java (My Crops tab)
Pass the registered crops to the view	Alex	1	MainHome_Controller.java
Show the crop information on the UI	Alex/Josue	2/2	TrackCrops_UI.java
Show the reports in the UI.	Josue	3	MainHome.postReports()
Pass the reports to the view.	Josue	1	MainHome_Controller.java method to get the reports
Set up the Database on Heroku	Alex	1	-
Configure the project for Hibernate	Alex	2	pom.xml, persistence.xml
Create the Entities' classes	Alex	3	Crop.java, User.java, Report.java
Create CRUD methods for the Entities	Alex	3	CropCRUD.java, UserCRUD.java, ReportCRUD.java
Documentation	Alex	5	This, tojtuyeetel.mp4

Alejandro Torre
Amaury Morales
Josué Torres

Metrics

The metric was handled like this:

- The sum of the ‘value’ column is the 100% of the project.
- Each member sums his corresponding values.
- We proceed to make the arithmetical adjustments to set each one the contribution percentage.

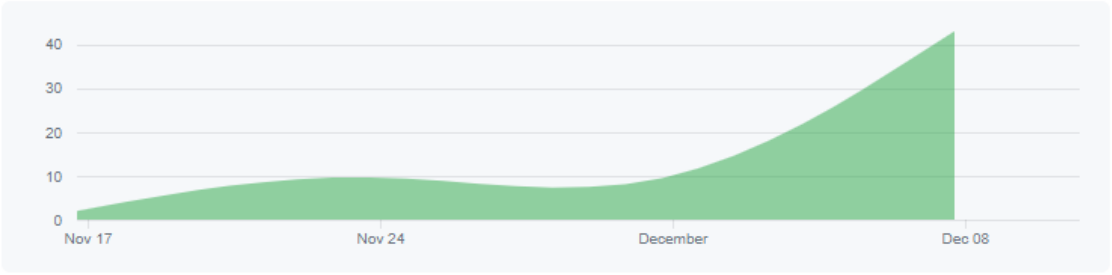
Total: 48

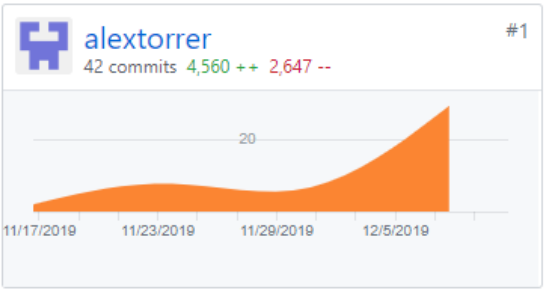
Alex: 25/48 : **52%**

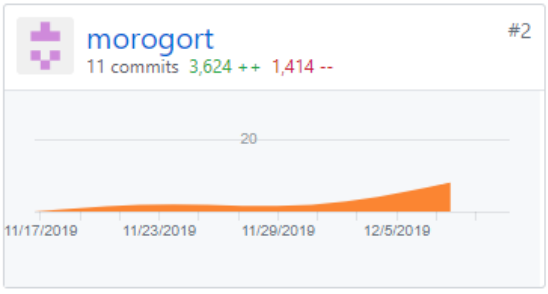
Josue: 19/48 : **40%**

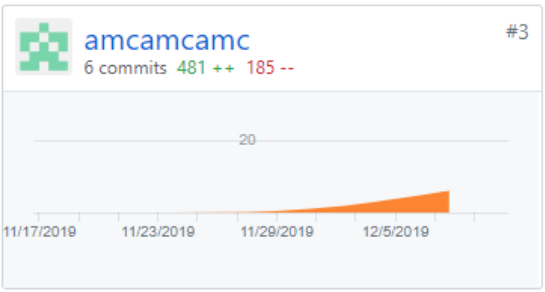
Amaury: 4/48 : **8%**

Repository Contributions









Alejandro Torre
Amaury Morales
Josué Torres

Additional Elements

- MySQL database on Heroku.
- Java and MySQL connection with Hibernate.
- Improve English speaking on presentations.