CloudFarm: Management of Farms and Crops Data on the Cloud

Apostolos Rousalis¹, Stelios Sotiriadis¹,², Euripides G.M. Petrakis¹
¹School of Electronic and Computer Engineering, Technical University of Crete, Chania Greece
²Computer Engineering Research Group, University of Toronto, Toronto, Canada
apostolosr@hotmail.com, s.sotiriadis@utoronto.ca, petakis@intelligence.tuc.gr

Abstract—Recent technological advances in Cloud Computing technology paved the way for developing and offering advanced services for remote monitoring in many industry areas; including the agricultural sector. In this work, we developed a Cloud-based farm management information system, referred as CloudFarm. CloudFarm offers a variety of services that helps farmers to manage and keep control of their farms and crops, by supporting all the basic agricultural activities according to set of good agricultural practices. CloudFarm is accessible to farmers, as a mobile application running on a smartphone or tablet that connects with a Cloud service for data analysis and orchestration. The “on the go” system means that farmers can manage and capture scheduled or performed activities directly from fields, while using the smartphone application that is synchronized with data stored permanently on the cloud so that farmers can have always access to up-date and real-time information. Furthermore, CloudFarm offers a Web application where farmers can monitor their data and certified advisors can monitor the farmers’ agricultural activities and advise them about their farm status and best planning of farm and crops works.

Keywords—Cloud Computing, Farm management, Crops management

I. INTRODUCTION (HEADING 1)

Nowadays, there is a constant challenge for food producers to grow safe, healthy products in a responsible way. There is a huge need for farmers to conform with Good Agricultural Practices (GAP)¹ codes, standards and regulations in order to guarantee food safety and product quality. Farmers must be well organized and able to manage all the details of the farm and be certified from certified organizations by providing information that covers a period from the seeding process to the multiple farming activities until post-harvest management. By adopting the use of modern information technology along with approved management techniques and engineering technology, the development of farm management information systems is the new way of optimizing farm production in terms of product quality and operation efficiency [2, 3]. There is an information flow that provides the farmer with external knowledge and decision support in order to perform efficient field operations, and it serves as a means of transmitting data about farm and field operations. A large amount of data from fields activities are collected by electronic devices and transmitted to remote storage infrastructures. As an additional benefit, various stakeholders such as, agronomists, government and legislation bodies, industries, tap into this information system of data flow to collect and analyze information.

This work focuses on the development of a farm management system that is not running locally on the farmer’s personal computer but is an “on the go” system meaning that farmer s can capture performed activities directly from field s, using a mobile application. Both farmers and their advisors can monitor the farm data through a web application accessed from a web browser over the Internet. Our proposed solution exploits state-of-the-art technology solutions in computer and internet technology using the internet such as remote activity management and monitoring via mobile devices (smartphones, tablets). The concept of cloud computing gets significant attention during the latest years and many companies recognized the advantages and the impact can have in people’s life. Today, it is evolved to an important technology for application developers and end users by allowing on demand and remote access of resources. Cloud allows real time data collection and analysis in an efficient way by offering an informative view of resources, remote data management, easy access and economic benefits. Taking cloud computing benefits into consideration, we develop an application that is easy to use and is made accessible to farmers, as a mobile application running on a smartphone or tablet and connects with the cloud, and as a web application running in the cloud. Our ambition is to increase the acceptance of farmers with the minimum exposure to technology, as all communication is through user-friendly interfaces and other means (e.g. instructions by narration and iconic interfaces).

We aim to develop a farm management information system, called CloudFarm, running on the cloud with main purpose to increase farmers’ farm s productivity. It will support the management (e.g. monitoring, recording) of all the basic agricultural activities such as spraying, harvest, fertilization according to a collection of agricultural rules (GAP). As referred to above, CloudFarm is running on the cloud, which means it can be accessed anywhere and anytime by a mobile application or a web application with an internet connection. The following summarizes the advantages and main characteristics of CloudFarm application:

¹ https://extension.psu.edu/food-safety-and-processing/commercial-food-processing
It runs on the cloud and assists farmers in keeping track of their crop and harvesting activities, anytime, anywhere, through friendly interfaces running on a smartphone, tablet or desktop.

It reduces the paperwork and increases safer data collection by minimizing human errors and information loss.

It can be used off-line in cases of loss of internet connection and allows data collection off-line.

It syncs local data with data stored on the cloud when the internet connection is back.

It helps farmers to get a certification for their products and processes, which increases the direct revenues for their products and the loss of income due to products non-sold.

It helps farmers to save money in production by maximizing the economic benefit for the farmer. The benefit comes from reduction of inputs (agrochemicals) and/or achieved better performance.

Also, the fact that the system is on the cloud also provides the following advantages:

Users can access the services from any place at any time. If necessary, a user can acquire more storage resources.

Users are charged per use of the resources. This way they know how much they are paying and can manage it according to their needs.

Minimizes hardware and software procurement and possession (e.g., maintenance) costs. This responsibility is left to the cloud provider.

CloudFarm application resorts entirely to the cloud provider to guarantee good services provision based on good practices and Service Level Agreements (SLA) agreed with the end-users (the farmers).

II. Motivation

The increasing pressure on farmers to follow the Good Agricultural Practices enforced them to provisions and restrictions in the use of input products such as fertilizers and agrochemicals. Farmers deal with huge managerial load and need to keep track of huge information loads in order to keep up to date records of farm operations (e.g., crop operations). This process is time-consuming and suspicious to human errors, due to the fact that all information must be checked and recorded by the farmer, that can easily lead to false results and decisions. Most farmers are using spreadsheets to keep their data organized and they have to insert all information manually and cross-check the performed agricultural activities themselves, such as remembering the exact days of the performed activities, the chemical amount consumed in a spraying activity, updating the stock amount after a spraying or fertilization activity, keep track of the field and crop history and more. All these types of cross-checking are time-consuming for farmers and there is also a huge risk of forgetting details about performed activities such as dates, used pesticides, used quantities and false or incomplete information in spreadsheets, might contribute to losing a certification, due to the fact that their data is not according to good agriculture principles.

The increasing use of computers, automate the way of handling and processing farm data but still many farm management (FMS) software systems are made to be used by farmers as well as advisors who are working in front of a desktop computer rather than on the field and they are not able to leave the computer desk where the systems are developed. Besides the risk of incomplete or false information (as data input is done off-line), there is also the risk of data loss, as farm data stored locally in a computer, data can be damaged because of a computer virus or mechanical or software damage. So, there is a need of a farm management information system that runs from a safe remote infrastructure that can be accessed from a Web browser over the internet, minimizing this way the farmers’ hardware maintains costs.

III. System Requirements Analysis

This section focuses on the system design aspect. A part of the study was the interviewing process of several farmers and agronomists in order to capture their needs according to their requirements on using the system. Differences in their viewpoints are mapped to different classification per user type (e.g. farmers, agronomists and system administrators). Users can interact with the CloudFarm system running on the cloud using a mobile application that runs on a smart-phone or tablet and a web application, as Figure 10 illustrates. The web application provides a digital dashboard allowing users to monitor input data. On the Cloud, farmers can view reports from agriculture activities, advisors can supervise multiple farmers by monitoring their farm data, and administrators can manage users and keep the database updated. The mobile application is only used by farmers when a farm activity (e.g. spray, harvest, fertilization) takes place.

Next we describe the different farmer roles.

- Farmers, are the users that control the overall farm activities through the Web application and also input information about farming actions (e.g. spraying,
internet is back.

synchronize with data stored persistently in the cloud in case all data are cached locally and synchronize with data stored persistently in the cloud when internet is back.

The main features of CloudFarm are (a) capturing information directly from the field during an agriculture activity using a mobile device (A GPS confirms the location of the performed activity), (b) it is fully functional in agriculture areas without a network connectivity, (c) complies to Crop, Field and Storehouse Management procedures according to Good Agriculture Practices directives. The system functionality is exposed to farmers in two ways, either via a mobile application running on a mobile device (smartphone or tablet) or via a Web interface running on the cloud. The former is intended to be used by farmers while on the field while the latter is intended to be used for accessing data online in the cloud. The main difference between the two applications is that the mobile application works in two modes. Firstly online and also in offline mode, in case of loss of internet connection in which case all data are cached locally and synchronize with data stored persistently in the cloud when internet is back.

The functionality for farmers on the mobile application includes the following:

- Create crop (sowing): Registers the process where farmers plant seeds into an empty field. The plants category and species are selected from a catalogue provided from a Plant Protection Service of the Greek Ministry of Agriculture.
- Harvest crop: Registration of the crop work (e.g., harvesting vegetables from a field) and registering weights or units. Harvest is allowed only if the minimum waiting date after the last spraying is over. Otherwise if there isn’t any spraying activity recently performed the crop can be harvested.
- Spray crop: Registration of spraying work (for protection from harmful insects, or diseases). The allowed chemicals are selected from the catalogue of the Greek ministry of Agriculture.
- Fertilize crop: Registration of the process of using natural general type fertilizer or synthetic new - type fertilizer essential to the growth of plants.
- Remove crop: Removes the crop and the field turns inactive which signals the end of the crop season.
- Insert supply to Storehouse: Registration of purchased supplies such as fertilizers or chemicals from the list of the Ministry of Agriculture.
- Stock supervision: View available quantities of supplies in the storehouse.
- Login into CloudFarm: A user can log in to CloudFarm with a FIWARE account.
- Registration: Upon first successful login into CloudFarm using a FIWARE account, the user is prompted to fill up a registration form in order to become known as a CloudFarm user.

The functionality for advisors on the cloud (exposed via Web Application) includes the following:

- Crop Management with easy access to Crops Details using a Map with fields, view crops history, view detailed reports about crop’s overall harvests, sprayings and fertilizations.
- Storehouse Management including actions such as view supplies availability, view supply consumption, view detailed supply consumption reports on where and when fertilizers or sprays have been used.
- Fields Management including view fields on the Map, insert or delete a field.
- Advisors Management including view their advisors, accept a pending supervision advisor request, remove an advisor from advisors list.

The functionality for Advisors on the cloud (exposed via Web Application) includes (a) access to the overall farm condition (crops, harvests, sprayings and fertilizations) per farm and per field, (b) access to the storehouses with details and consumption of supplies per farmer and per field, (c) advisors for farmers by sending an advising request to farmer, stop advising a farmer. The functionality for system administrators on the cloud (exposed via Web Application) includes: (a) management of pending Users: Decline or approve pending request, manage Users: Add User, Delete Users, View Users, (b) add, update plant protection products, details such as chemical dosage per crop and chemical packages and (c) management of fields: add or remove a farmer’s Field.

IV. SYSTEM IMPLEMENTATION

In this section, we present the CloudFarm system architecture diagram as shown in Figure 2. The system comprises of two main architectural blocks referred to as, a) the Front - End comprising of the Web application and the mobile platform application, b) the Back-End comprising of services for the management of information stored permanently in the cloud. Building upon principles of Service Oriented Architectures (SOA) CloudFarm is realized by means of well-defined components (services) communicating with each other using REST protocol. Next, we present those services, and then we separately analyze the functionality and the implementation of each. The building blocks of the architecture are the Front - End the part of the CloudFarm system which could be visible to the users through a mobile application running on a smartphone or tablet and a Web application running on the Back-End. The Back - end comprised of services running on the cloud, and the Web Application (a Web interface to allow system users interact with the cloud using a Web browser).
A. Back-end

The Back-end is the main part of the CloudFarm system hosted entirely on the cloud and it combines all system services such as storage, authentication, plant protection Web service and Application logic that fulfill the system requirements. The application logic is the “brain” of the CloudFarm system and provides a REST API that supports multiple endpoints and methods to system services and it can be used by clients in order to access the system’s resources. Every single request to the API is processed first by the Application logic to validate the data, as also to authenticate the user who submits the request via the Authentication Service. Each request to REST API is processed first by the Application Logic to validate the data, as also to authenticate the user who submits the request via the Authentication Service.

C. Storage service

This service is responsible for storing and retrieving data from a database. Its main functionalities are offered as a REST API making data storing and retrieving easy for developers and users. It is implementing using a CouchDB server which provides a well-structured REST HTTP API 30 for accessing or processing the data. It can host multiple databases which store JSON documents. In our system, each farmer has his own database in CouchDB server, comprised of multiple JSON documents (crops, harvests etc.). Farmers can read, edit, and insert database documents in their databases by sending requests to the API, along with credentials obtained in the form of access tokens by the user identification and authorization service. Each database defines its own security policy document by storing the users who have access to the database. Server admins have full access and can read, edit, and delete documents in every database in the CouchDB server. Advisors can only read documents from farmers who advise.

D. Authentication and authorization service

The authentication service is provided by Keyrock Identity Management Generic Enabler (IdM) that complies with the OAuth2 and supports token-based authentication. The reason why we choose a token-based authentication, is because it supports the stateless principle of REST API, so there is no need to keep a session stored in the server, the token is a self-contained entity that conveys all user information. The token lives in local storage on the Front-End. An end-user visits CloudFarm web Application or opens the mobile app.

- A user entering Cloud Farm application is redirected to Keyrock Identity Management GE of FIWARE in order to login into the CloudFarm.

- The use clicks a “Log In” button on CloudFarm site or app and are redirected to FIWARE IdM (Identity Manager) website and is prompted to log in with a FIWARE account and accept certain permissions.

- On successful login, the authorization server will redirect the user back to application along with an access token.

- On every single request to CloudFarm API (Resource server) must provide the access token, so the Back-End can verify the user’s identity by asking Keyrock to identify the user with the given access token.

REST API implementation and methods of Keyrock Identity Manager can be found at FIWARE Lab Instance documentation 32.

E. Plant protection service

This service was developed in order to provide all the information on plant protection products and fertilizers according to the catalogues of the Greek Ministry of Agriculture. Constitutes an effort of a REST Web Service for the implementation of the Plant Protection Catalogues provided...
by the Greek Ministry of Agriculture. It’s a public RESTful Web service which fully provides all the details for plants, chemicals and fertilizers used in crop activities. Currently the Greek Ministry of Agriculture does not offer any web service where developers or third - party applications can consume agricultural data. For the implementation of this service we download a public instance of the database of Greek Ministry of Agriculture and replicate it into a database to our CouchDB server. It uses as format type JSON documents and it’s a public web service for anyone interested in the development of any related agricultural application which needs that kind of agricultural information. Clients can send requests to the web service in order to fetch data in JSON format.

F. Users service

This services manage s users by creating or removing one. There are two kind of users the CloudFarm users and the Pending users. When a user is registered in the CloudFarm by submitting a form, the user is considered as a pending user and must wait to be approved by the administrators. Administrators can accept or decline pending requests. If an administrator decides to accept a pending request, then the user is being removed from the pending users and inserted to CloudFarm users with a role given from the administrator. Table 2, shows the REST API implementation of this service method. The user id for each user is the email encrypted using Adler - 32 33 encryption hash function. When the user is registered into the system, the user’s email that obtained from Keyrock Id entity Management GE is hashed in order the system to create the user id. So, the user id and the user email are related.

G. Fields service

The main purpose of this service is to create, read, update and delete a farmer’s field. Any request to this service must be authenticated first from the authentication service to identify the user who wants to access field resource. A farmer can create or delete a field resource stored in the farmer database. An advisor can read the fields of the advised farmers. Administrators have full access to farmer’s fields and use all CRUD operations on a field resource. Table 3, shows the REST API implementation of this service methods.

H. Crops service

The main purpose of this service is to create, read, update and delete (remove) a farmer’s crop. In crop creation, the field where the crop is going to be created must be checked if already has an active crop. After a successful crop creation, the field resource is updated as an active field with the basic information about the crop. So, crop service uses the fields service to update the field in which a farmer wants to create a crop. In addition, a crop resource can be updated with given data. Moreover, a crop can be removed, means that the end of crop period is over and the crop must also be removed from the field. This method is the delete function which does not basically remove the crop resource from the database just updates the crop status to inactive. After a successful removal of crop, the Crops Service calls the Fields Service in order to update field’s status from active to inactive. Any request to this service must be authenticated first from the authentication service to identify the user who wants to access the crop resource. A farmer can create, read, update or delete (remove) a crop. An advisor can read crops data of farmers who advises. Administrators have full access to farmers’ crops data and can create, read, update or delete any crop resource. Table 4, shows the REST API implementation of this service methods.

I. Harvest service

The main purpose of this service is to create and read a harvested crop performed on farmer’s crop and to protect the consumer’s health from the chemicals side - effects. A Harvest is only performed if the minimum required harvest date of crop after a spraying action has passed. In order to get the minimum harvest date of crop, the service uses the Crops Service to obtain crop’s information. The quantity harvested is measured in kilos or pieces. Any request to this service must be authenticated first from the authentication service to identify the user who wants to perform harvest and access crop’s information. Only a farmer can perform a harvest (create harvest). An advisor can read harvests data of farmers who advises. Administrators have full access to farmer harvests data and can read, update or delete any harvest resource. Table 5, shows the REST API implementation of this service methods.

J. Storehouse service

The main purpose of this service is to manage farmer’s storehouse supplies (items). A farmer can register (create) a purchased supply and read supplies details. A supply in storehouse is automatically updated after a spraying or fertilization action. A storehouse item can be removed (deleted) only when the supply was inserted by mistake and it hasn’t consumed anywhere. Any request to this service must be authenticated first from the authentication service to identify the user who wants to access the supplies. Only a farmer can insert, update and delete a supply. An advisor can only read supplies of farmers’ who advises. Administrators have full access to farmers’ storehouses data and can read, update or delete any storehouse item resource. Table 6, shows the REST API implementation of this service methods.

K. Spray service

The main purpose of this service, is to manage sprayings performed on farmer’s crops according to agricultural rules and principles in order to protect the consumer’s health from the chemicals side - effects. A spraying is only performed with a permitted chemical and only if the minimum required harvest date of crop after a spraying action has passed. In order to get the minimum harvest date of crop, the service uses the Crops Service to obtain crop’s information. Moreover, this service communicates with the Plant Protection Service to get the chemical details and calculates the crop’s new possible harvest date using those details. The new possible harvest date calculated as the number of days that the crop cannot be harvested due to chemical side - effects from the spraying date. Before spraying the availability of the chemical in storehouse should be checked. Thus, the storehouse service is called to provide the amount of the chemical in the storehouse. The needed proportion of the chemical is automatic calculated (in
cube centimeters) according to the chemical details obtained from the Plant Protection Service and the field's acres. After a successful spraying the Sprays Service calls the Crops Service to update the crop's harvest date and also calls the Storehouse Service to update the new stock amount. This service uses an efficient algorithm on how the quantity will be consumed on a spraying when more than one packages of the supply are existing in storehouse. Moreover, this services is responsible to restore stock to previous amounts after a removal of wrong spraying insertion. Any request to this service must be authenticated first from the authentication service to identify the user who wants to do a spraying. Only a farmer can perform a spraying activity. An advisor can read spraying data from farmers who advises. Administrators have full access to farmers' spraying data and can read, update or delete any spraying resource. Table 7, shows the REST API implementation of this system service.

L. Spray service

The main purpose of this service, is to manage fertilizations performed on farmer’s crops. A fertilization can use two types of fertilizers, the general type fertilizer and the new type fertilizers.

The user must insert in kilograms the dosage of the fertilizer. In order to perform crop’s fertilization, the service calls the Crops Service to obtain crop’s information, and the Storehouse Service to check the fertilizer’s availability in the storehouse. This service uses an efficient algorithm on how the quantity will be consumed on a fertilization when more than one packages of the supply are existing in storehouse. After a successful fertilization, the Fertilization Service calls the Storehouse Service to update the new stock amount. Any request to this service must be authenticated first from the authentication service to identify the user who wants to do a fertilization. Only a farmer can perform a fertilization. An advisor can read fertilizations data from farmers who advises. Administrators have full access to farmers' fertilizations and can read, update or delete any fertilization resource. Table 8, shows the REST API implementation of this service methods.

M. Advisors service

The main purpose of this service, is to manage farmers’ advisors and the pending advising requests which are sending from the advisors to the farmers. In the one hand, an advisor can read all the farmers details who advises and send (create) an advising request to a farmer. An advisor can also stop advising a farmer by deleting him from his farmers advising list. In the other hand, a farmer can, read all the advisors who advise him, accept or decline an advising request and delete an advisor from his advisors list. When the farmer accepts advisor’s pending request, then the advisors is added in the farmer’s advisors list. Any request to this service must be authenticated first from the authentication service to identify the farmer or advisor who wants to access the service. Administrators have full access to this service and can read, update or delete an advisor resource. Table 9, shows the REST API implementation of this service methods.

N. CloudFarm Front-End

The mobile application developed as a hybrid application using the web technologies such as HTML 34, CSS 35 and JavaScript 36 hosted in application’s WebView 37. The reason why the hybrid approach was chosen is:

1. The need of a mobile application to work on multiple platforms such as Android and iOS without developing versions for every platform. We have one codebase for mobile platforms (Android, iOS). So, we will be definitely saving on development time.

2. We can easily develop the mobile application using Web technologies without being familiar with the programming language and environment of each mobile platform.

3. The ability of application to be distributed through the app stores.

The CloudFarm mobile application is developed using Ionic 38 framework based on Apache Cordova, AngularJS, HTML and CSS. The plugins were used are the geolocation to locate farmer’s position, the network information which quickly checks the network status and the InAppBrowser 39 which launch a URL in another in-app browser (redirects from services).

One of the main goals of the mobile application, was to support offline functionality so it can be used by farmers, even if there is no internet connection, because some agricultural lands might not be covered with a 3G or 4G network. The restriction of the internet connection led to implement services that can run offline scenarios, locally inside the mobile application logic. Those services exchange data with a local storage without the need to interact with the cloud when the application is offline. With the offline syncing end-users, can perform agricultural activities without internet connection. After the device is back online, the changes are synced with the Back-End using the API. Another main goal of the mobile application was the ability to locate a farmer inside a field.

O. Application Logic

Is the most crucial service of the mobile application. It has the logic intelligence which is used for the communication between the CloudFarm system (Back-End) and the services of the application. It also supports the ability to perform agricultural activities when the mobile application is offline and when is back online it begins the syncing process between the local data stored in local storage service and the Back-End. It also manages the User interface of the mobile application where farmers can interact with the system. It follows the Good Agriculture Practices principles and standards for every agricultural activity performed by a farmer, through user interface, such as harvest, spraying, fertilization. Of course, the log in function cannot be provided offline and it is only available when the device is connected to the internet because it must interact with the Authorization Service and Users Service in the Back-End in order to verify the user. So, the
application can work offline only when the farmer has previously logged in.

As described earlier, farmers should be able to capture farm activities performed on fields instantly by using the mobile application. Thus, the application should detect the field in which the farmer is performing those activities. For that reason, the application logic uses the Geolocation Service (an Apache Cordova plugin) which returns the geographic location of the farmer and the radius based on information about cell towers, WiFi 40 nodes, GPS 41. The application logic in order to identify in which field the farmer is located it uses an algorithm which uses the farmer’s location gained from Geolocation Service. The algorithm uses a point (latitude, longitude) which indicates the current location of the smartphone, gained from geolocation service (GPS). Next, it should check if the point is really within a field. A field is represented as a polygon, consisted of multiple points. If the farmer’s point is inside a field is calculated by iterating all over the polygon’s points (every point being x,y) and finds the min ,max values of x and y. A point is inside a polygon if it has not have x value smaller than xmin and greater than xmax or y value smaller than ymin and greater than ymax. It is a very simple algorithm and runs very fast. This algorithm quickly excludes a point if it is not within a polygon. The application logic runs this algorithm for each field in order to identify if the user is locating inside a field or not.

```javascript
if (farmer.x < Xmin || farmer.x > Xmax || farmer.y < Ymin || farmer.y > Ymax) {
  // Definitely not within the field (polygon)
}
```

### P. Local Logic

This service is responsible to store application’s data locally in the device. The mobile application can be used offline, if there is no internet connection, so the application should be able to store data locally using PouchDB [1] database which provides datastore for offline applications [1]. PouchDB is a lightweight open-source database written in JavaScript inspired by Apache CouchDB that stores JSON documents and runs well within the hybrid application’s WebView. When the application is connected to the internet, in order to support offline functionality, fetches farmers needed data from the Back-End such as active crops, crops sprays, crops harvests, crops fertilizations, fields, storehouse items, and stores the data in the local database. In that way, if the device goes offline it has all the required data. In the offline mode, during an agricultural activity the application stores the data in the local database as JSON objects. When the internet connection is back, the local database syncs data with the remote database on the cloud using the API. For example, if a farmer has performed a spraying activity offline, a spraying JSON document is stored in the local database and when the application is online sends the spraying data to Sprays Service in the Back-End using asynchronous call (AJAX 44) to S prays Service endpoint. While the application is online all the performed activities are sent directly to the Back-End and synced with the local database by fetching from the cloud only the data that changed by the performed activity. So, if the application suddenly loses the internet connection it can be fully functional by having all the required data.

Moreover, the mobile application must support the Plant Protection Service which provides all the necessary information about plants and chemicals according to the Greek ministry of Agriculture. Thus, if there’s no internet connection the mobile application cannot get the data from the Plant Protection Service. To solve this problem, when the application is online it gets the plants and chemicals data from the Plant Protection Service running in the cloud (Back-End) and stores them in the local storage.

### Q. Web APP

As we discuss in system architecture (chapter 3.8.1) the second Front-End is a Web interface that allows system users to interact with the Back-End in the cloud using a Web browser. The web application is as a single page application (SPA) developed with AngularJS framework, HTML, CSS, PHP. The AngularJS service is a core AngularJS service that facilitates communication with the REST API via AJAX 47 (asynchronous HTTP requests, for transferring data between client and server. Using AJAX asynchronous HTTP requests to communicate with the back-end, the application can still use a responsive Web page, while the request is waiting for a response from the Back-End. In that way, the web application does not need to reload the whole page to get new information. The Front-End works with the following logic. A user opens a Web browser and enters the URL of CloudFarm Web application. Then the Web applications is stored and runs inside the Web browser. If the user requests to view data, then an asynchronous http request is sent from the browser to the REST API, by using the AngularJS http service, in order the browser to get the required data. The AngularJS displays dynamically this data in a visual representation in the HTML document without refreshing and re-rendering the whole web page.

### V. API PERFORMANCE

In order to evaluate the system performance, we chose a demanding use case scenario in terms of computing resources. This scenario is the execution of a spraying activity. Inserting a spraying into the system communicates numerous system services such as Keyrock Identity Manager (Authorization Service), Sprays Service, Crops Service, Storehouse Service and Storage Service.

In the first experiment, we set concurrency equal to 1 in order to run 2000 requests which were submitted one at a time to the server (sequential requests). Concurrency equal to 1 means that all the requests are submitted sequential to the API. The test complete d2000 sequential requests in 164.2 seconds with total data transferred 1556 kb. Before we execute the benchmark the virtual machine CPU usage was at 1%. While benchmark was running the highest virtual machine CPU usage was 24% which occurred after 1800 completed requests. As expected, while the test was running we noticed that there is a
low CPU usage, due to the fact that only one request was executed per time.

In the second experiment, we set the concurrency number to 50 in order to run 2000 requests where simultaneously 50 different users, are doing 40 sequential requests. The test completed in 8.9 seconds with total data transferred 1556 kb. Before we execute the test, the virtual machine CPU usage was at 1%. While the test was running the highest virtual machine CPU (peak) usage was 60%, which occurred after 1000 completed requests. The results shown that the API performance is quite good, beside the incensement of CPU and RAM usage. This occurs because parallel requests bind the system resources until they get the response from the system. Each request to API can generate multiple sub-requests to CloudFarm services in order to return the required information. The demand of processing all those requests at the same time, increases the CPU and RAM usage of our virtual machine.

In the third experiment, we set concurrency equal to 200 in order to run 2000 requests where simultaneously 200 different users, are doing 10 sequential requests. The test completed in 10.2 seconds with total data transferred 1556 kb. Before we execute the test, the virtual machine CPU usage was at 1%. While the test was running the highest virtual machine CPU (peak) usage was 99%, which occurred after 1200 completed requests. The results shown that the API struggles to handle all requests because the CPU is reaching the highest limits. The above results show, that after increasing more the number of parallel requests to the API, the system performance drops dramatically. This happens because more parallel requests bind more system resources, until the time they will get served. Each request to API can generate multiple sub-requests to CloudFarm services in order to return the required information. This demand of processing all those sub-requests at the same time, increases the CPU and RAM usage of our virtual machine. When the CPU approaches its limits, all the requests at that time are delayed because they CPU cannot handle more requests at that time.

VI. CONCLUSIONS

The goal of this work was to create a system for managing and monitoring farmers’ farms and their crop’s data using services which take advantage of Cloud Computing capabilities, combined with the idea of the Internet of Things. To achieve that, we implemented a mobile application for smartphones or tablets which is used by farmers to perform agricultural activities inside the field and a web application which can be used by farmers and advisors in order to monitor the farm data. The developed system is further evaluated by load testing to ensure the applicability of CloudFarm under severe load conditions. CloudFarm will be an essential part for farmers, helping them to manage their farms organization, supervision and certification. It will help farmers to smartsly monitor their farm and to have access to useful information that will help them make the best decisions. They can also be able to view in every moment the efficiency of every crop and they can act objectively. In the future, we plan to implement a task scheduling service where farmers can schedule agriculture activities and can be notified when a scheduled activity is delayed.

REFERENCES