Smart meeting room scheduling and management system for a university campus using Android tablets with Firebase backend and Headwind MDM

Stuardo Lucho[1], Angelo Velarde[1], and Mario Ampuero[2]
[1]Departamento de Ingeniería, Pontificia Universidad Católica del Perú, Lima, Perú, stuardo.lucho@pucp.edu.pe, angelo.velarde@pucp.pe
[2]Facultad de Ciencias e Ingeniería, Pontificia Universidad Católica del Perú, Lima, Perú, m.ampuero@pucp.pe

Abstract*– The proper use of the rooms on a university campus has become a priority, not only because of the situation we find ourselves in, but also because of the need to optimize the use of educational infrastructure and reduce operating costs. For this reason, the use of the Internet of Things in the development of reservation systems and monitoring of space occupancy, including an adequate management system for mobile devices is essential. In this work, a system has been developed that allows to visualize the occupation of the classrooms on a university campus. The necessary information is obtained from the Google Calendar API and displayed on an Android tablet located outside the room. The Android application has been developed using Flutter, and Firebase Realtime Database and Firebase Storage have been used for data storage and synchronization. Furthermore, as part of the system, a web application has been developed to manage the information stored in Firebase and a mobile device management system has been implemented to keep track of the tablets used, their updates and usage restrictions. The system was deployed on a cloud infrastructure using Google Cloud and Amazon AWS to reduce investment costs.

Keywords—Smart Meeting Room, Internet of Things, Firebase, Flutter, Headwind

I. Introduction

On a university campus there are various spaces that students can freely use, such as computer laboratories, spaces in a library or open classrooms. However, there are other environments that, due to the characteristics of the equipment they have or for security reasons [1], it is necessary that a person supervises the students in the use of the equipment or that the student who uses the environment identifies and reserves it to be responsible for the correct use of the equipment.

The people responsible for these specialized laboratories usually use different tools to manage occupancy, such as Microsoft Excel, Microsoft Outlook Calendar, Google Calendar [2], Microsoft 365, Calendly [3] or Doodle [4]. These tools allow the person in charge to organize and properly distribute the class and research sessions during the academic semester.

This study is carried out at a university in Lima, Peru, where in one of its technological careers, there is a laboratory (identified as v305) which has different telecommunications equipment such as routers, switches, and firewalls of various brands (Cisco, Huawei, Fortinet). This laboratory is for the exclusive use of laboratory sessions and research groups. For that reason, a person in charge is required to guarantee the safety and correct use of the equipment. However, if a student wants to use it, they can reserve the laboratory by sending an email to the laboratory manager, who receives the request, checks availability (which is found in a Google Calendar) and proceeds to make the reservation.

Students are unaware of the availability of the laboratory in advance and on many occasions, simultaneous requests must be rejected due to a prior reservation. This situation is repeated in other laboratories with similar equipment, such as the v304 where there is radio frequency equipment, or the v307 where there is electromagnetic simulation equipment. Even the department meeting room, has its availability in a Google Calendar but only the department secretary can add or edit reservations, generating a bottleneck and in the same way, the working staff (teachers and administrative) cannot see the availability of the environment. These situations are repeated in other laboratories and classrooms of the university.

The objective of this research is to develop a solution that allows students to see the occupation of the laboratory by integrating various services and applications. First, a mobile application developed in Flutter that will allow the student to view the current event that is happening in the classroom, as well as its daily and weekly availability, which is obtained in real time from a Google Calendar. This application will be installed on a tablet that is located outside the classroom, laboratory, or meeting room. Second, a web application to manage the Google Calendar of each laboratory and link them to each tablet. To enable high availability, all system storage and hosting resources are in the cloud of two service providers, such as Amazon AWS and Google Cloud. Likewise, a mobile device management (MDM) system has been implemented to monitor the tablets and display the configurations and updates of the operating system and the proposed application, automatically.

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2021.1.1.383
ISBN: 978-958-52071-8-9 ISSN: 2414-6390

The rest of the article is organized as follows. Section 2 briefly describes the work related to systems and tablets for viewing reserves. Section 3 describes the system architecture and solution design. Section 4 shows the components for deployment and deployment. Section 5 shows the results and a discussion about what was developed, and finally section 6 shows the conclusions.

II. RELATED WORK

In the literature, several proposals have been found that measure and show room availability called Smart Meeting Room (SMR) solutions, which use various IoT devices to provide the room with interactivity and automation. There are works such as [5],[6] and [7], where the occupancy of the room is shown based on a set of sensors, such as PIR (Passive Infrared) sensors, humidity sensors, which determine the presence or absence of people in the room, in such a way that, if a meeting had been agreed and the sensors do not detect presence in a time range, the room is released. Other papers, such as those presented in [8] and [9] focus on creating a system to manage room occupancy, through applications developed for mobile devices (Android and iOS) using (in one of them) Firebase as a backend to store information. Likewise, the work of [10] shows the use of Google Calendar to manage room occupancy through a web application.

In the literature review, it was not possible to find articles that provide a solution to the problem described in the present investigation. However, it was found that a widely used solution for the development of mobile applications in SMR systems was Flutter and as a backend system to store information, Firebase.

On the other hand, at a commercial level there are various solutions that show the availability of the room. Just to mention some of them: [11], [12], [13] and [14], which have various features such as:

- Integration with several calendar providers (Microsoft 365, Google Calendar, iCalendar, among others).
- Room reservations on the screen.
- Personalized welcome screen.
- Some of them have proximity sensors that turn off the screen for power saving.
- 802.1X authentication mechanism supported
- And many others advanced features.

All these solutions meet the requirements of our research with even more benefits; however, they are more oriented to the business field, due to their high cost (around $500 to $1000 per device) and lots of features. In our research many of these advanced features such as: allowing reservations through the tablet or supporting the 802.1X authentication mechanism, are not necessary, and the price per device is remarkably high.

On the university campus where the current research has been developed, the number of classrooms, laboratories and meeting rooms is around 400, therefore, buying one of these existing solutions is not feasible. Rather, it opens the possibility to create a particular low-cost solution to the problem posed.

III. SYSTEM DESIGN

In this section, the system components, their constraints, and their interaction between them are shown. The proposed solution developed is composed of 5 modules, as shown in Fig. 1.

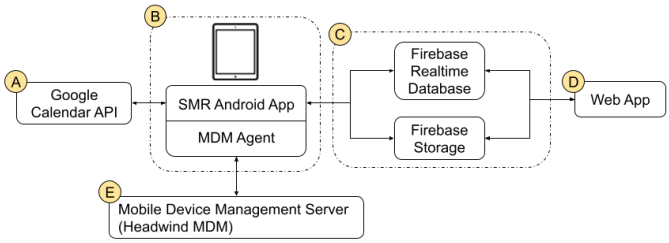

Fig. 1 System architecture

A. *Google Calendar API*

The Calendar API [15] allows the developer to integrate an application with Google Calendar, to find and view public calendar events and, if authorize, modify calendars and events. In this system, it will be used to obtain information about the event that is currently happening. For a mobile application to query the Google Calendar API, it is necessary to configure a project in Google Console, where the number of requests will be counted and discounted from the free quota provided by Google Calendar. After configuration, the application needs to authenticate itself through the oAuth2 protocol, where the username and password linked to the account are entered.

In this case, as the application that will communicate with Google Calendar will work in kiosk mode (it will be detailed in section 3B), there is no user who can log in, the access has been configured as a service account [16], [17], which allows authentication without the need to enter credentials, configuring the access permissions and the account in the Google console.

B. *Mobile Applications*

Two applications will be installed on the tablet:

1) SMR Android App: smart meeting room application developed in flutter, which allows, if required, to port the application to iOS in the future. This app shows:

- The availability of the room (classroom, laboratory, meeting room, etc.)
- A logo of the department to which that room belongs
- An announcement to the public if necessary.

The application gets the calendar ID from module C (Firebase), from the Firebase Realtime Database. With this ID, the application sends periodic requests to the Google Calendar API to obtain event information.

The frequency of updates depends on two factors: the free quota provided by the Google Calendar API and what is the maximum tolerable delay to display an event update in the application. Regarding the first point, the Google Calendar API has a quota of 1,000,000 daily queries per project within the Google console in the free version [18].

All tablets will initially be linked to a single project in the Google console. For this reason, if a 24/7 operation is desired, there would be 86,400 queries per tablet per day. In this case, the system would allow to have a maximum of 11 tablets simultaneously, obtaining the events from Google Calendar.

However, this value can negatively affect the network as there would be 11 tablets making requests every second, 24 hours a day. To avoid this, this calculation can be adjusted since the activities at the university are carried out in the range from 7 am to 11 pm, giving a total of 16 hours (57,600 seconds) of continuous operation. If an interquery time of 5 seconds is defined, the total number of queries is reduced to 11,520 per day and therefore, it allows 86 tablets to work in real time (with a maximum update delay of 5 seconds) without exceeding the free quota of the Google Calendar API.

An example of how the screen is displayed when there is an event is shown in Fig. 2. In the lower right section, there is a box which is an advertisement created in the web application (module E), obtained from the Firebase Realtime Database while the image is obtained from the Firebase Storage. In this case, updates are not constantly requested. Instead, a bidirectional WebSocket is opened between the application and Firebase (Realtime Database and Storage) and using the observer pattern (through listeners), it allows to receive notifications only when there is a change in Firebase, avoiding saturating the network with unnecessary traffic.

Fig. 2 Example of the application screen with the current event

When the "Semanal" (Weekly) button located in the upper right corner is pressed, a calendar with the room availability appears as shown in Fig. 3. This calendar is an embedded iframe and is refreshed only by pressing the "Semanal" or "Diario" (Daily) button again.

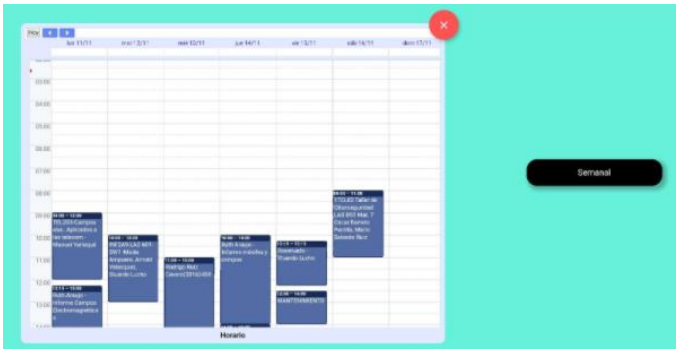

Fig. 3 Embedded calendar with weekly occupancy

If the tablet loses connection to the Internet, a message is displayed in the event box indicating that the room has no events and another message with the text "No connection" appears in the upper left, as shown in Fig. 4.

Fig. 4 Example of the application screen without connection

It is important to mention that the first time the application runs on the device, the following parameters must be entered (as shown in Fig. 5):

- The “Activo fijo” code, which is a code used by the university to identify every IT equipment.
- The location where the tablet is being physically installed, for example: Classroom 101.

Both parameters are stored in Firebase Realtime Database (module C), and will be used later by the web application (module D) to link a calendar to this tablet.

Finally, the *android: keepScreenOn* parameter has been configured in the Android Manifest, so that Android does not turn off the screen due to user inactivity.

Fig. 5 Application initial screen

2) MDM Agent: for tablet configuration management and monitoring, there are various solutions on the market, which are known as Mobile Device Management system (MDM) [19]. Among the open-source alternatives available such as Flyve [20] and Headwind [21], the MDM headwind was selected, since Flyve had more demanding hardware requirements and had many more options than headwind, making it heavier and with functionalities not needed for the current project.

In an MDM architecture [22] there is an MDM agent which is installed on the device to be monitored and an MDM server, where the policies to be applied, the configurations and other security restrictions necessary for the mobile device are defined [23].

The headwind MDM agent is installed on the tablet, and later this same agent installs the SMR App (section 3B.1).

Likewise, the MDM Agent is configured to start the SMR App in kiosk mode, which allows:

- Run as a single application on the device
- Do not allow notifications
- Do not allow closing the application
- Lock the start button

Finally, by using an MDM architecture, we can send new versions of the SMR application without having to do it manually on each Tablet.

C. *Firebase Realtime Database and Storage*

Both Firebase services have been used for the system's persistence layer, due to their direct integration with Flutter (module B) and with the web application to manage calendars (module D).

Firebase Realtime database [24] is a non-relational database that works under the Platform-As-A-Service schema and allows storing information in JSON format. Here the list of devices will be stored, each one identified by its "Activo Fijo" code. Likewise, for each one it will be saved:

- The physical location of the device
- The calendar ID to display on the device
- The ad, which is optional
- The path of the image to display, which is stored in Firebase Storage.

The data structure of two devices, in which the "Activo Fijo" code of the tablet is the node that groups the other elements is shown in in Fig. 6.

D. *Web Application*

The web application allows you to manage the devices registered in the Firebase Realtime Database, linking them with their respective calendar that will be displayed on the tablet. Likewise, it allows you to configure the ad per tablet and the image to be displayed.

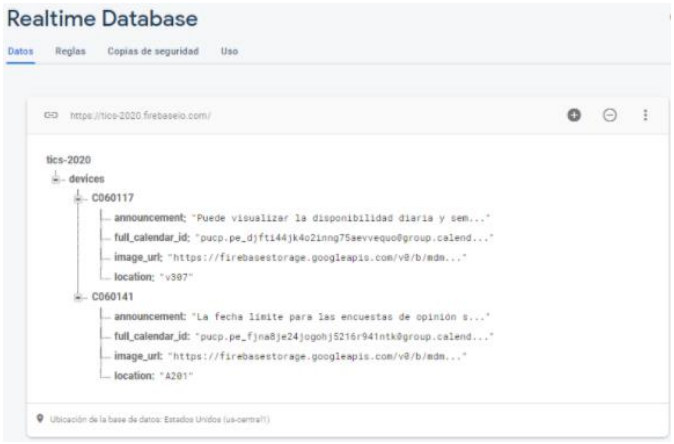

Fig. 6 Data structure in Firebase Realtime Database

The system has been developed using Spring Boot 2.1, with the Spring Web MVC module based on Java 8 and bootstrap as a CSS framework. The communication between the web application and Firebase is at JavaScript level, which provides a single point of persistence, to manage the stored data. An interface of this system is shown in Fig. 7.

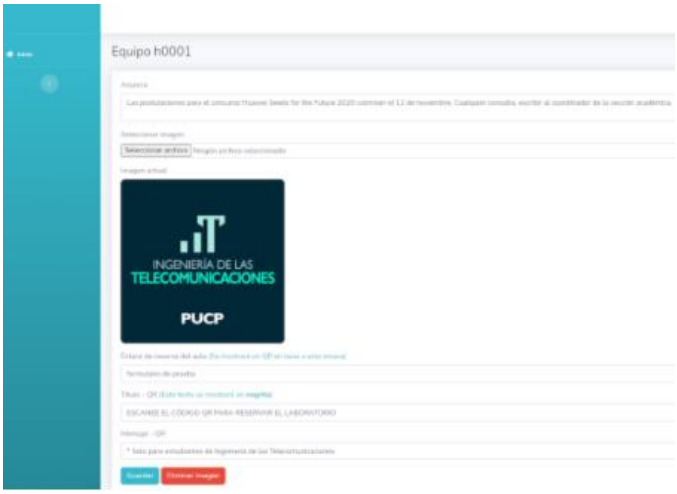

Fig. 7 Web Application interface

E. *Mobile Device Management Server*

The headwind MDM server allows to have a registry (inventory) and monitoring of all the tablets deployed on campus. Likewise, the MDM server is configured to deploy new versions of the SMR App through the MDM agent installed on each device. The importance of having a deployed MDM infrastructure is that it allows you to manage the new tablets that are added, track and monitor the current ones, and keep the SMR App updated. Also, it allows to configure the policies of the kiosk mode described in the section 3B.2.

IV. IMPLEMENTATION

The system has been implemented 100% in the cloud (except for the tablet application) to reduce investment costs (CAPEX) and have the elasticity of the cloud (if required). An architecture of the implementation of the modules with their respective cloud service is shown in Fig. 8.

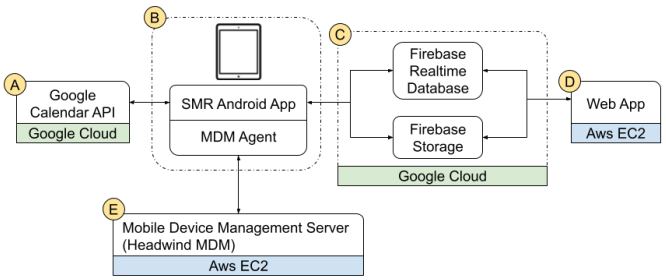

Fig. 8 System architecture with cloud deployment

1) Modules A and C: both the Google Calendar API module and the Firebase Realtime Database and Storage modules are hosted on Google Cloud. The Google Calendar API costs are 0, because according to the calculations (shown in section 3B), the limits provided by the API will not be exceeded. The free tier plan (Spark Plan) for Google Firebase is shown in Table 1.

TABLE I
FIREBASE SPARK PLAN [25]

Service	Property	Spark plan
Realtime Database	Simultaneous connections	100
	GB stored	1 GB
	GB downloaded	10 GB/month
Storage	GB stored	5 GB
	GB downloaded	1 GB/day
	Upload operations	20K/day
	Download operations	50K/day

Based on Table I, the tablet only reads from the database when the application is installed for the first time or at each restart of the application; while the web application is configured only once and then it would only be used to read and write in case it is required to modify the ad, the calendar ID or the image that is displayed on a tablet. For this reason, Firebase provides 10 GB of traffic in its free tier, which is an approximate (according to Firebase) to 200 million messages. Likewise, according to the calculations made in section 3B, the number of supported tablets (due to the limitation of Google Calendar API) is 86, which means that the limit of simultaneous connections is not exceeded.

Regarding Firebase Storage, in the current version the image is uploaded at its maximum resolution, considering a maximum of 1 image per tablet. If the user uploads a new image, the previous one is deleted and replaced by the one new. This allows that, even if the image is high definition, approximately 12 MB, with 86 tablets, a maximum storage of approximately 1,032 GB would be reached. Therefore, the free limit is not exceeded either.

2) Modules D and E: both the MDM server and the web application, both have been deployed in a single AWS EC2 instance. The characteristics are as follows:

- Type: t2.micro [26]
  - vCPU: 1
  - Memory (GiB): 1
  - Storage: gp2 (8GB)
  - Network performance: low
  - Open ports: 22 (ssh), 8080 (headwind server) and 80 for the web application
- Region: US East (N. Virginia)
- Operating system: Ubuntu Linux 18.04 (64 bit)
- Elastic IP: to avoid the IP change when restarting the instance in the event of any update or unforeseen event.

The reason for selecting AWS for the application deployment is that the university is an AWS partner through the AWS Educate program, which provides researchers, teachers, and students with several annual credits (renewable), which have been used for this Project. The number of credits of the account used is $ 200 and the costs associated [27] with the monthly deployment of the used instance are shown in Table II.

TABLE II
AWS ASSOCIATED COSTS

Service	Price	Monthly price (30 days)
t2.micro	$0.0116 per Hour	$8.352
Storage: 8 GiB	$0.10 per GB-month	$0.8
elastic IP Address	1 elastic IP at no charge	$0

In total, the monthly cost is around $9.1, therefore per year it would be approximately $109, which would be within the $ 200 margin provided by the annual AWS Educate program. Likewise, this referential cost can be used if a paid account wants to be used.

3) Module C: the application was deployed on two tablets (due to the current situation of the coronavirus pandemic, it has not been possible to deploy it in more rooms), using the MDM agent. The most relevant specifications of the tablets used are shown in Table III:

TABLE III
TABLETS USE IN THE PROJECT

Property	Tablet 1	Table 2
Provider	Samsung	
Model	SM-T820	SM-T590
Screen	9.7''	10.5''
Resolution	2048x1536	1920x1200
Processor	2.15 GHz	1.8 GHz
RAM	4 GB	4 GB
Storage	32 GB	32 GB
Operating system	Android 9	Android 9

V. RESULTS AND DISCUSSION

After completing the development and testing of the application in conjunction with the MDM, a case had to be designed that could hold the tablet while hiding the side buttons (on, off and volume) and the front buttons (Recent, home, back). These designs are shown in Fig. 9 and Fig. 10.

The case is the height and width of the tablet; However, it has a depth of 13 cm because, in addition to containing the tablet, a plug was installed behind it to connect it and keep it constantly energized. It also had a hinge with a padlock on the top that allowed only authorized people to open the box and change the tablet in case of failure.

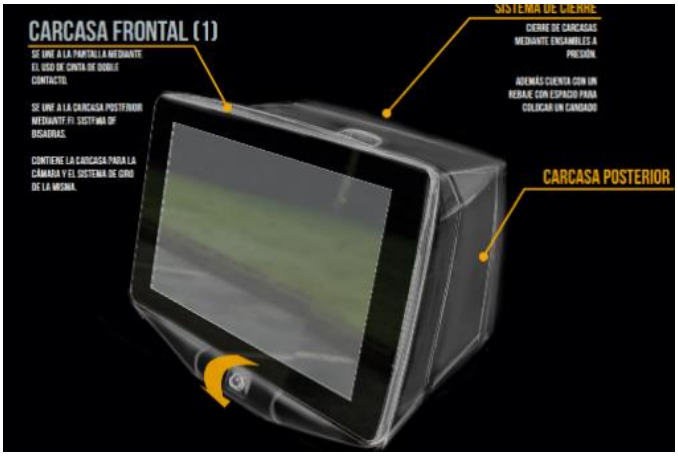

Fig. 9 Case 3D design – front view

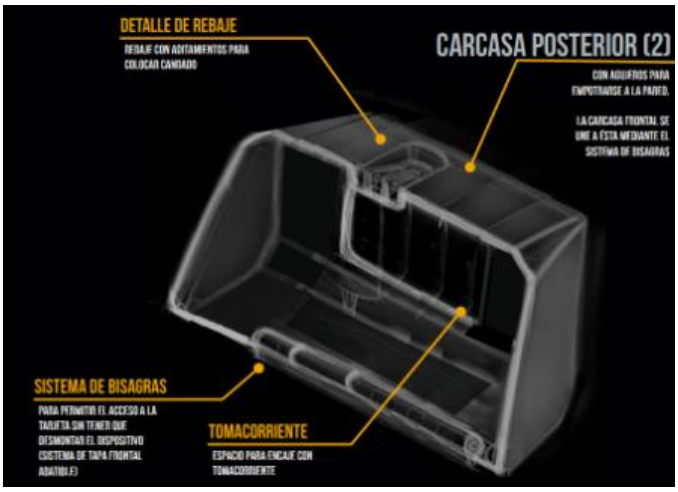

Fig. 10 Case 3D design – rear view

The case was designed by the digital manufacturing laboratory team and developed with the additive manufacturing system Fused Deposition Modelling (FDM) using 3D printing and placed outside two university laboratories (v306 and v307), located in the Electronics, Informatics and Telecommunications building, as shown in Fig. 11.

Fig. 11 Tablet mounted in the wall

An improvement has been proposed for future versions, in which the tablet automatically shuts down at 11pm and is then turned on by an external component at 7am. This would save energy and extend the life of the tablet.

On the other hand, the application works correctly but still consumes a high rate of traffic, since it is checking every 5 seconds whether there is a new event on the calendar at the current time, and for now, only for two tablets.

To make this service truly scalable and consume the least possible bandwidth from the university, an improvement has been proposed for the next version, using the push notification functionality of Google Calendar API, which warns when there is a change in the calendar [28]. Google Calendar push notifications do not send the content of a new event; otherwise, it only informs if there is any new, modified, or deleted event in the calendar. This functionality could be used to find out if there has been a change in the tablet's calendar, if so, make a request to the Google Calendar API and request the current event, thus saving a large amount of traffic.

VI. CONCLUSION

In this article, a Smart Meeting Room Scheduling and Management system has been developed that allows to visualize the availability of rooms (classrooms, laboratories, meeting rooms) of a university campus and whose configuration management is carried out through MDM Headwind.

The research project has used two cloud providers, Amazon AWS and Google Cloud. In Amazon AWS, the EC2 cloud computing service was used with a t2.micro instance for the deployment of the MDM server and the web application to manage the calendars and tablets. On the Google side, the Google Calendar API has been used to obtain the information of the calendars, whose initial configuration is carried out through the Google Console, and the Firebase Realtime Database and Storage, for the storage of data from tablets and calendars.

Concerning the cost issue, as all services are in the cloud, most of the cost is OPEX (operational expenses). The Amazon AWS service has an operation and maintenance cost (OPEX) but the use of AWS Educate credits has been considered as an educational institution with an agreement with Amazon AWS, which reduces these costs to 0. On Google's side, the cost is also 0, since the limits are not exceeded in any of the services used. Even if the entity had an on-premises infrastructure, the services deployed in Amazon AWS could be mounted on its own servers, since the requirements for backend operation are minimal.

Finally, this research presents in the results and discussion section, various improvement options, which can optimize bandwidth consumption as well as energy management. However, the initial proposal meets the research objectives and the requirements set out.

ACKNOWLEDGMENT

This work has been carried out within the Internet of Things research group (IoT-PUCP), belonging to the Pontifical Catholic University of Peru.

REFERENCES

[1] Z. H. Li, "Research on the statistical analyses and countermeasures of 100 laboratory accidents. J," *Exp. Technol. Manag.*, vol. 31, pp. 210–213, 2014.
[2] Google Inc., "Google Calendar." https://support.google.com/a/users/answer/9302892 (accessed Feb. 15, 2021).
[3] Calendly, "Calendly.com." https://calendly.com/ (accessed Dec. 20, 2020).
[4] Doodle, "Doodle.com." https://doodle.com/ (accessed Dec. 20, 2020).
[5] M. Saravanan and A. Das, "Smart real-time meeting room," in *2017 IEEE Region 10 Symposium (TENSYMP)*, 2017, pp. 1–5, doi: 10.1109/TENCONSpring.2017.8070069.
[6] L. Duc Tran *et al.*, "A smart meeting room scheduling and management system with utilization control and ad-hoc support based on real-time occupancy detection," in *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, 2016, pp. 186–191, doi: 10.1109/CCE.2016.7562634.
[7] L. M. Sánchez, I. Díaz-Oreiro, L. Quesada, L. A. Guerrero, and G. López, "Smart Meeting Room Management System Based on Real-Time Occupancy," in *2019 IV Jornadas Costarricenses de Investigación en Computación e Informática (JoCICI)*, 2019, pp. 1–6, doi: 10.1109/JoCICI48395.2019.9105174.
[8] H. Singh and R. R. Shah, "BOOKiiIT - Designing a Venue Booking System (Technical Demo)," in *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*, 2020, pp. 287–291, doi: 10.1109/BigMM50055.2020.00050.
[9] A. Praveen *et al.*, "Conference Room Booking Application using Flutter," in *2020 International Conference on Communication and Signal Processing (ICCSP)*, 2020, pp. 348–350, doi: 10.1109/ICCSP48568.2020.9182183.
[10] M. Wannous, H. Nakano, and T. Nagai, "Google Calendar™ for managing and monitoring the utilization of a web-based laboratory's resources," in *2011 IEEE Global Engineering Education Conference (EDUCON)*, 2011, pp. 210–213, doi: 10.1109/EDUCON.2011.5773138.
[11] Evoko, "Evoko Naso." https://evoko.se/ (accessed Feb. 10, 2021).
[12] TouchOne, "TouchOne-12-M." https://www.touchone.eu/GBR/Product-detail?productId=049C90152169E61180F55065F38B16F1. (accessed Feb. 15, 2021).
[13] Steelcase, "RoomWizard." https://www.steelcase.com/products/scheduling-systems/roomwizard/ (accessed Feb. 15, 2021).
[14] GoGetCorp, "GoGet." https://gogetcorp.com/ (accessed Feb. 15, 2021).
[15] Google Inc., "Google Calendar API." https://developers.google.com/calendar (accessed Feb. 15, 2021).
[16] Google Inc., "Google Console Service Account." https://cloud.google.com/iam/docs/service-accounts (accessed Feb. 15, 2021).
[17] Google Inc., "Using OAuth 2.0 for Server to Server Applications." https://developers.google.com/identity/protocols/oauth2/service-account (accessed Feb. 15, 2021).
[18] Google Inc., "Google Calendar pricing." https://developers.google.com/calendar/pricing (accessed Feb. 15, 2021).
[19] M. M. Yamin and B. Katt, "Mobile device management (MDM) technologies, issues and challenges," *ACM Int. Conf. Proceeding Ser.*, no. Mdm, pp. 143–147, 2019, doi: 10.1145/3309074.3309103.
[20] F. Corp., "Flyve MDM." https://www.flyve-mdm.com/ (accessed Feb. 15, 2021).
[21] H. Corp., "Headwind MDM." https://h-mdm.com/ (accessed Feb. 15, 2021).
[22] H. Batool and A. Masood, "Enterprise Mobile Device Management Requirements and Features," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 109–114, doi: 10.1109/INFOCOMWKSHPS50562.2020.9162763.
[23] D. Hayes, F. Cappa, and N. A. Le-Khac, "An effective approach to mobile device management: Security and privacy issues associated with mobile applications," *Digit. Bus.*, vol. 1, no. 1, p. 100001, 2020, doi: 10.1016/j.digbus.2020.100001.
[24] Google Inc., "Firebase Realtime Database." https://firebase.google.com/docs/database (accessed Feb. 15, 2021).
[25] Google Inc., "Firebase princing." https://firebase.google.com/pricing (accessed Feb. 15, 2021).
[26] Amazon AWS, "AWS instance types." https://aws.amazon.com/es/ec2/instance-types/ (accessed Feb. 15, 2021).
[27] Amazon AWS, "Amazon AWS pricing." https://aws.amazon.com/ec2/pricing/on-demand/?nc1=h_ls (accessed Feb. 15, 2021).
[28] Google Inc., "Google Calendar push notifications." https://developers.google.com/calendar/v3/push (accessed Feb. 15, 2020).