Implementing BLoC Architecture in a Mobile Cloud Computing Environment

Gabriel Antonio González Caraballo, Department of Computer Engineering, Universidad del Norte

Abstract—Mobile devices are a technology that has had an exponential growth over the last decade. With such multiplicity comes ubiquity. Rising end users demands introduce the need for better performant mobile applications. Limitations of mobile devices, such as computing capacity and performance, and battery life; can be overcome by offloading computational intensive tasks to a remote server i.e. the Cloud. This paper introduces a cross-platform mobile cloud application, named Project HeatMaps, that follows the BLoC design pattern and allows users to retrieve demographic heat maps based on their location so as to avoid highly congested zones. The proposed system features Microservices and Serverless architectures. The data gathering scheme proposed follows the crowdsourcing principle.

Index Terms—Mobile Cloud Computing, Serverless, Microservices, Mobile development, BLoC

1 Introduction

Mobile cloud computing [57] combines the advantages of two emerging technologies i.e. cloud computing and mobile technology, that have gone through a maturation process over the last decade. According to Data Reportal there are approximately 2.87 smartphones users worldwide as of 2020. More specifically, in the United States more than 285 million people own smartphones according to Pew Research Center. Currently, Android holds the biggest market share in mobile operating systems, with an estimated 87% market share by 2023 [5]. With such abundance of mobile devices, the market of mobile applications has skyrocketed. Valuates Reports claims the market size of mobile applications will reach USD 407.31 billions by 2026. New smartphones extend the capabilities of previous smartphones thus enabling them to run more complex or performant applications.

However, smartphones are constrained by factors such as relatively limited computing power and battery. There exists different approaches that aim to overcome limitations of smartphones in order to provide richer applications. The most researched approach, discussed in this article, is augmentation via computational offloading. Computational offloading refers to distributing the code so as to relegate computational heavy tasks to a better performant back end i.e. Cloud server, thus aiming to preserve battery life on mobile devices while obtaining results more efficiently.
Computational offloading in the context of mobile cloud computing can be performed in several fashions. Namely, remote cloud offloading; cloudlet offloading and femto-cloud offloading. Cloud Computational offloading allows mobile devices to harvest the –apparently– infinite resources of the cloud.

According to [12] the global cloud computing market size is expected to grow from USD 371.4 billion in 2020 to USD 832.1 billion by 2025. Main cloud providers include Microsoft's Azure [13], Google Cloud Platform [56] and Amazon Web Services (AWS) [8]. Their market share's as of the third quarter of 2020 are respectively: 19%, 7% and 32% [11, 58].
Mobile cloud applications have a wide range of implementations, that includes healthcare, banking, e-learning, gaming, point-of-sale, entertainment, mixed reality, transportation, among several others 18. In the current context of COVID-19 several mobile applications have been developed in attempt to handle the pandemic. The mobile application discussed in this paper aims to reduce contagion in controlled environments e.g. a university, by providing users with demographical heat maps thus allowing them to avoid zones with high traffic.
The rest of this papers is organized as follows, Section 2 presents a literature review of concepts related to mobile cloud computing and mobile application development, Section 3 presents the design and implementation of the proposed application; finally, Section 4 concludes the paper and suggests future work.

2 Literature Review

According to [53], MCC is "an emerging mobile cloud paradigm which leverage mobile computing, networking and cloud computing to study mobile service models, develop mobile cloud infrastructures, platforms and service applications for mobile clients". In a similar fashion, [35] defines MCC as "a model for transparent elastic augmentation of mobile device capabilities via ubiquitous wireless access to cloud storage and computing resources, with context-aware dynamic adjusting of offloading in respect to change in operating conditions, while preserving available sensing and interactivity capabilities of mobile devices [65, 30]. [38] identifies the primary distinct service features and requirements of MCC: Offer on-demand mobile data services on a mobile cloud based on service level agreements, Pay-as-you-use business model, offloading computing power demands from mobile devices to clouds, multi-tenanted mobile data service, highly elastic and scalable data service and, highly reliable and available mobile data service.

Implementing BLoC Architecture in a Mobile Cloud Computing Environment

Gabriel Antonio González Caraballo, Department of Computer Engineering, Universidad del Norte

Abstract—Mobile devices are a technology that has had an exponential growth over the last decade. With such multiplicity comes ubiquity. Rising end users demands introduce the need for better performant mobile applications. Limitations of mobile devices, such as computing capacity and performance, and battery life; can be overcome by offloading computational intensive tasks to a remote server i.e. the Cloud. This paper introduces a cross-platform mobile cloud application, named Project HeatMaps, that follows the BLoC design pattern and allows users to retrieve demographic heat maps based on their location so as to avoid highly congested zones. The proposed system features Microservices and Serverless architectures. The data gathering scheme proposed follows the crowdsourcing principle.

Index Terms—Mobile Cloud Computing, Serverless, Microservices, Mobile development, BLoC

1 Introduction

Mobile cloud computing [57] combines the advantages of two emerging technologies i.e. cloud computing and mobile technology, that have gone through a maturation process over the last decade. According to Data Reportal there are approximately 2.87 smartphones users worldwide as of 2020. More specifically, in the United States more than 285 million people own smartphones according to Pew Research Center. Currently, Android holds the biggest market share in mobile operating systems, with an estimated 87% market share by 2023 [5]. With such abundance of mobile devices, the market of mobile applications has skyrocketed. Valuates Reports claims the market size of mobile applications will reach USD 407.31 billions by 2026. New smartphones extend the capabilities of previous smartphones thus enabling them to run more complex or performant applications.

However, smartphones are constrained by factors such as relatively limited computing power and battery. There exists different approaches that aim to overcome limitations of smartphones in order to provide richer applications. The most researched approach, discussed in this article, is augmentation via computational offloading. Computational offloading refers to distributing the code so as to relegate computational heavy tasks to a better performant back end i.e. Cloud server, thus aiming to preserve battery life on mobile devices while obtaining results more efficiently.
Computational offloading in the context of mobile cloud computing can be performed in several fashions. Namely, remote cloud offloading; cloudlet offloading and femto-cloud offloading. Cloud Computational offloading allows mobile devices to harvest the –apparently– infinite resources of the cloud.

According to [12] the global cloud computing market size is expected to grow from USD 371.4 billion in 2020 to USD 832.1 billion by 2025. Main cloud providers include Microsoft's Azure [13], Google Cloud Platform [56] and Amazon Web Services (AWS) [8]. Their market share's as of the third quarter of 2020 are respectively: 19%, 7% and 32% [11, 58].
Mobile cloud applications have a wide range of implementations, that includes healthcare, banking, e-learning, gaming, point-of-sale, entertainment, mixed reality, transportation, among several others 18. In the current context of COVID-19 several mobile applications have been developed in attempt to handle the pandemic. The mobile application discussed in this paper aims to reduce contagion in controlled environments e.g. a university, by providing users with demographical heat maps thus allowing them to avoid zones with high traffic.
The rest of this papers is organized as follows, Section 2 presents a literature review of concepts related to mobile cloud computing and mobile application development, Section 3 presents the design and implementation of the proposed application; finally, Section 4 concludes the paper and suggests future work.

2 Literature Review

According to [53], MCC is "an emerging mobile cloud paradigm which leverage mobile computing, networking and cloud computing to study mobile service models, develop mobile cloud infrastructures, platforms and service applications for mobile clients". In a similar fashion, [35] defines MCC as "a model for transparent elastic augmentation of mobile device capabilities via ubiquitous wireless access to cloud storage and computing resources, with context-aware dynamic adjusting of offloading in respect to change in operating conditions, while preserving available sensing and interactivity capabilities of mobile devices [65, 30]. [38] identifies the primary distinct service features and requirements of MCC: Offer on-demand mobile data services on a mobile cloud based on service level agreements, Pay-as-you-use business model, offloading computing power demands from mobile devices to clouds, multi-tenanted mobile data service, highly elastic and scalable data service and, highly reliable and available mobile data service.

According to NIST [41] (United State's National Institute of Standards and Technology), Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. [22] affirms that the main difference between Cloud Computing and Mobile Cloud Computing lies on where the processing is done. In the latter the processing and data storage occur externally, while in the first usually only data storage occurs externally. The cloud computing model is composed of five essential characteristics, three service models, and four deployment models.

The before mentioned characteristics are, namely: on demand self-service, broad network access, resource pooling, rapid elasticity and measured service. Sehgal [55]claims that the three main operational characteristics of the cloud are: automated provisioning and data migration, transparent scaling and on-demand self-service. Both sets of characteristics of the cloud are considered to be complementary. The main three service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The deployment models are related to the organization that owns the cloud and are: public cloud, private cloud, hybrid cloud and community cloud [62, 4].

According to [55] the cloud computing model has several advantages over traditional computing models, such as: shared infrastructure reduces costs, pay as-you-go i.e. the user only pays for the resources consumed, elasticity on demand, increased focus on the application layer and, leveraging the hardware concerns to the cloud provider. However, such advantages introduces a series of risks to be considered. The set of risks includes losing knowledge and direct control over underlying hardware, noisy neighbors potentially affecting performance, difficulty to diagnose performance issues due to virtualization and limited visibility, potential security risks of placing mission critical data on remote servers and vendor lock-in, that refers to difficulties on retrieving data from a specific vendor.

The mobile cloud computing model benefits from characteristics of mobile devices (e.g. smartphones, tablets, laptops) such as portability, accessibility, elevated multiplicity, immediacy of access; while overcoming their main limitations such as limited computational resources and battery life [1, 39, 66, 2, 57]. [22] identifies five key resource limitations of mobile devices: CPU, memory, storage, battery and time. To achieve this, the mobile cloud computing model leverages cloud services to offload heavy computational tasks to the cloud [49, 65], thus preserving the battery life of mobile devices and achieving results, and supporting applications, that would otherwise take too long for the user to have a positive user experience (UX) or would be unfeasible on mobile devices [31, 22]. [22] points out industries that would benefit from mobile cloud services being e-commerce, healthcare [29], e-learning, education, finance, point-of-sale and transportation [2].

[38] identifies issues concerning mobile cloud applications which are as follows. First, data security and mobile user privacy are needed to guarantee the user's trust in the mobile system. Second, multi-tenancy and customization in mobile data service, considering multi tenancy as a one of the key features of MCC and Mobile SaaS, the need for supporting multi-tenanted mobile data services arose. [39] points out writing mobile cloud computing applications is not straightforward since it implies parallelization, partitioning and distribution of the code. Furthermore, mobile developers need to address the high variability in network conditions induced by the constant mobility of devices.

[22] identifies the following MCC application models: augmented execution, where processes are offloaded to the cloud via virtualization; elastic partitioned/modularized applications, where unlike augmented execution granularity of partitioning occurs at the application layer instead of the process layer; application mobility, ad-hoc mobile cloud and; cyber foraging, where dynamically discovered resources are used for remote execution. The main difference between the ad-hoc mobile cloud model and cyber-foraging consists on where the resources are located. In an ad-hoc mobile cloud environment, resources are on mobile devices. Cyber-foraging utilizes remote resources.

Taking into consideration the limitations in terms of storage in mobile devices, [37] indicates one alternative approach to mobile-based databases are cloud-based databases that support mobile users and applications on mobile devices. To this end, authors present a mobile enabled cloud database solution i.e. framework, called MCloudDB "which can be used by developers as a bridge to connect and integrate mobile applications with back-end cloud databases to support mobile cloud data management and access services". Regarding multi-tenancy [37], in a mobile cloud environment tenants can access services at the following different levels increasing in abstraction: physical data store level (which refers to the data storage infrastructure), logic database level, data table level, application level and, user interface level.

Identity as a Service is one representative example of a mobile cloud service, which is composed of the following components: Authentication, Authorization, User Management and Central User Repository. Authors in [44] introduce a series of criteria for evaluating Identity and Access Management (IAM) services. The criteria are as follows: extensive authentication and authorization support, user-friendly single sign-on support, lightweight standard/protocol; platform independent, vendor neutral and open standard; scalable standard, web and native mobile apps support, consumer and enterprise support, immediate revocation of access support, end user delegated authorization support, data integrity support, data confidentiality support, and mobile standard.

[2] points out several application fields for mobile cloud computing. They are as follows. Mobile trade, where users can carry out mobile banking, mobile broker operations, mobile marketing and advertising. Mobile healthcare [29], which provides: rendering medical monitoring services for observation of patients in real time at any place, development of emergency control system, organization of mobile medical devices equipped with an alarm system and, storage of medical experience and data of patients used in studies. Mobile training, where companies can train their employees using mobile solutions. Mobile gaming, more and more games that consume cloud services e.g. in mul-

According to NIST [41] (United State's National Institute of Standards and Technology), Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. [22] affirms that the main difference between Cloud Computing and Mobile Cloud Computing lies on where the processing is done. In the latter the processing and data storage occur externally, while in the first usually only data storage occurs externally. The cloud computing model is composed of five essential characteristics, three service models, and four deployment models.

The before mentioned characteristics are, namely: on demand self-service, broad network access, resource pooling, rapid elasticity and measured service. Sehgal [55]claims that the three main operational characteristics of the cloud are: automated provisioning and data migration, transparent scaling and on-demand self-service. Both sets of characteristics of the cloud are considered to be complementary. The main three service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The deployment models are related to the organization that owns the cloud and are: public cloud, private cloud, hybrid cloud and community cloud [62, 4].

According to [55] the cloud computing model has several advantages over traditional computing models, such as: shared infrastructure reduces costs, pay as-you-go i.e. the user only pays for the resources consumed, elasticity on demand, increased focus on the application layer and, leveraging the hardware concerns to the cloud provider. However, such advantages introduces a series of risks to be considered. The set of risks includes losing knowledge and direct control over underlying hardware, noisy neighbors potentially affecting performance, difficulty to diagnose performance issues due to virtualization and limited visibility, potential security risks of placing mission critical data on remote servers and vendor lock-in, that refers to difficulties on retrieving data from a specific vendor.

The mobile cloud computing model benefits from characteristics of mobile devices (e.g. smartphones, tablets, laptops) such as portability, accessibility, elevated multiplicity, immediacy of access; while overcoming their main limitations such as limited computational resources and battery life [1, 39, 66, 2, 57]. [22] identifies five key resource limitations of mobile devices: CPU, memory, storage, battery and time. To achieve this, the mobile cloud computing model leverages cloud services to offload heavy computational tasks to the cloud [49, 65], thus preserving the battery life of mobile devices and achieving results, and supporting applications, that would otherwise take too long for the user to have a positive user experience (UX) or would be unfeasible on mobile devices [31, 22]. [22] points out industries that would benefit from mobile cloud services being e-commerce, healthcare [29], e-learning, education, finance, point-of-sale and transportation [2].

[38] identifies issues concerning mobile cloud applications which are as follows. First, data security and mobile user privacy are needed to guarantee the user's trust in the mobile system. Second, multi-tenancy and customization in mobile data service, considering multi tenancy as a one of the key features of MCC and Mobile SaaS, the need for supporting multi-tenanted mobile data services arose. [39] points out writing mobile cloud computing applications is not straightforward since it implies parallelization, partitioning and distribution of the code. Furthermore, mobile developers need to address the high variability in network conditions induced by the constant mobility of devices.

[22] identifies the following MCC application models: augmented execution, where processes are offloaded to the cloud via virtualization; elastic partitioned/modularized applications, where unlike augmented execution granularity of partitioning occurs at the application layer instead of the process layer; application mobility, ad-hoc mobile cloud and; cyber foraging, where dynamically discovered resources are used for remote execution. The main difference between the ad-hoc mobile cloud model and cyber-foraging consists on where the resources are located. In an ad-hoc mobile cloud environment, resources are on mobile devices. Cyber-foraging utilizes remote resources.

Taking into consideration the limitations in terms of storage in mobile devices, [37] indicates one alternative approach to mobile-based databases are cloud-based databases that support mobile users and applications on mobile devices. To this end, authors present a mobile enabled cloud database solution i.e. framework, called MCloudDB "which can be used by developers as a bridge to connect and integrate mobile applications with back-end cloud databases to support mobile cloud data management and access services". Regarding multi-tenancy [37], in a mobile cloud environment tenants can access services at the following different levels increasing in abstraction: physical data store level (which refers to the data storage infrastructure), logic database level, data table level, application level and, user interface level.

Identity as a Service is one representative example of a mobile cloud service, which is composed of the following components: Authentication, Authorization, User Management and Central User Repository. Authors in [44] introduce a series of criteria for evaluating Identity and Access Management (IAM) services. The criteria are as follows: extensive authentication and authorization support, user-friendly single sign-on support, lightweight standard/protocol; platform independent, vendor neutral and open standard; scalable standard, web and native mobile apps support, consumer and enterprise support, immediate revocation of access support, end user delegated authorization support, data integrity support, data confidentiality support, and mobile standard.

[2] points out several application fields for mobile cloud computing. They are as follows. Mobile trade, where users can carry out mobile banking, mobile broker operations, mobile marketing and advertising. Mobile healthcare [29], which provides: rendering medical monitoring services for observation of patients in real time at any place, development of emergency control system, organization of mobile medical devices equipped with an alarm system and, storage of medical experience and data of patients used in studies. Mobile training, where companies can train their employees using mobile solutions. Mobile gaming, more and more games that consume cloud services e.g. in mul-

tiplayer experiences networking, or storing user profiles.

Authors in [29] identify the potentials of mobile cloud computing applied to the field of healthcare. Hence, they propose a Medical Mobile Cloud Multi Agent System, named 2MCMAS, that combines mobile cloud computing and multi agent systems in order to provide cloud services for the healthcare industry.

[57] presents the proposal and design of a smartphone application, named Mission Swatcha, that uses cloud services as a bridge to connect citizens and authorities as a means to pursue a cleaner society. The application features mainly user profile, complaint submission and registry. Authorities can track complaints in real time and post updates on the complaints' status.

Authors in [26] identify the need for interactive learning tools, mainly in STEM fields (but can be extended to other fields) that facilitate teaching and learning and to make them more productive. Authors point out there are significant indicators that show that the usage of technology enhanced learning improves the quality of both teaching on learning. Research shows that mobile devices can be beneficial when used properly in a classroom environment. To this end, they propose a Mobile Response System (MRS) that uses mobile devices and cloud services in interactive problem solving. The proposed system's infrastructure aims to reduce instructor workload, give students transparent access and provide extensibility to different disciplines. MRS is defined by the authors as a software environment that eases in-class problem solving exercises and provides immediate feedback on submissions using mobile devices.

Authors in [64] propose a remote engine management system based on mobile cloud services. The Electronic Control Unit (ECU) of a motor allows not only to control it, but to monitor it and diagnose faults. The system proposes gathers ECU data in a mobile device e.g. smartphone, laptop; which then sends the data to the cloud platform, where big data analysis is performed. The mobile devices acts as a network broker between the ECUs and the centralized remote cloud.

Authors in [60] propose two mobile cloud applications for university environments. Specifically, the first is related to fire detection and the second to traffic monitoring. Both applications follow the same infrastructure and architecture. Sensors are connected to mobile devices, which are consumers of cloud services.

2.1 Mobile Computational Offloading

There are three main types of mobile computational offloading regarding the type of cloud environments being used. The type of the cloud offloading system being implemented affects the round-trip time and determines the configurational granularity. The three types are listed as follows. First, public (remote) cloud offloading, having the coarsest granularity in configuration, where computational heavy tasks are performed on a remote cloud server and their results are transmitted back to the mobile device. Second, cloudlet offloading, having a medium level of granularity in configuration, where a reduced cloud environment located close to the user can be used to perform computational heavy tasks instead of relegating them to the remote cloud or, to facilitate exchange of data between the client and the remote cloud [49, 59, 60]. Thus, reducing overall round-trip time. Third, ad-hoc cloud offloading, having the finest granularity in configuration, where a cloud environment is conformed of a heterogeneous set of mobile devices connected e.g. via WiFi, that share their computational resources thus enabling the relegation of tasks to others user thus increasing utilization. The three types described can be used in combination to optimize utilization, runtime, round-trip time and battery life.

Authors in [31] propose the Femtocloud system, which is a reduced cloud environment composed of mobile devices. The femtocloud system provides a highly dynamic, self-configuring and multi-tenant mobile cloud trough a cluster of mobile devices. Femtoclouds are also referred to as ad-hoc clouds, as mentioned above. Systems such as femtoclouds, where heterogeneous mobile devices share resources, are in need of a task scheduling algorithm in order to properly divide the workload and migrate the resources. [42] claims that in an ad-hoc cloud environment, pooled resources constitute a supercomputing node. Authors in [42] propose an implementation of the ad-hoc cloud architecture in an automated video surveillance application. Such application features object detection, tracking and classification, behavior analysis and action task. They also propose a task allocation scheme for the Femtocloud system.

[49] introduces the design and implementation of Context Aware Mobile Cloud Services (CAMCS), a middleware framework designed to deliver an integrated experience of the mobile cloud services to the user. CAMCS features cloud personal assistant task models, context-awareness with context processor and, service discovery and consumption.

The COMPSs [40] programming model aims to allow developers to code applications without awareness of parallelism or infrastructure details. Authors in [39] propose an infrastructure unaware framework named COMPSs-Mobile that allows developers to code regular Android applications that are transparently parallelized and partially offloaded to remote cloud resources. The COMPSs-Mobile framework features an offloading mechanism that determines if a task should run locally or remotely depending on factors such as round-trip time, delay and energy consumption. [39] points out that data transfers are one of the most influential factors on the economic, energetic and temporal cost of running and application.

2.2 Serverless and Microservices Architectures

One representative example of cloud services are serverless functions. Serverless functions are functions that run on demand on servers transparent both to the end user as to the developer, with minimal deployment effort. [46] points out no server deployment and management are needed. Serverless functions are invoked as a best-effort service [46] and are event triggered [32]. Serverless functions allow computational intensive code to be executed on a completely abstract infrastructure handled by the cloud service provider.

Serverless architecture implements code offloading via serverless functions in order to deploy web/mobile applications with features beyond the limitations of the clients.

tiplayer experiences networking, or storing user profiles.

Authors in [29] identify the potentials of mobile cloud computing applied to the field of healthcare. Hence, they propose a Medical Mobile Cloud Multi Agent System, named 2MCMAS, that combines mobile cloud computing and multi agent systems in order to provide cloud services for the healthcare industry.

[57] presents the proposal and design of a smartphone application, named Mission Swatcha, that uses cloud services as a bridge to connect citizens and authorities as a means to pursue a cleaner society. The application features mainly user profile, complaint submission and registry. Authorities can track complaints in real time and post updates on the complaints' status.

Authors in [26] identify the need for interactive learning tools, mainly in STEM fields (but can be extended to other fields) that facilitate teaching and learning and to make them more productive. Authors point out there are significant indicators that show that the usage of technology enhanced learning improves the quality of both teaching on learning. Research shows that mobile devices can be beneficial when used properly in a classroom environment. To this end, they propose a Mobile Response System (MRS) that uses mobile devices and cloud services in interactive problem solving. The proposed system's infrastructure aims to reduce instructor workload, give students transparent access and provide extensibility to different disciplines. MRS is defined by the authors as a software environment that eases in-class problem solving exercises and provides immediate feedback on submissions using mobile devices.

Authors in [64] propose a remote engine management system based on mobile cloud services. The Electronic Control Unit (ECU) of a motor allows not only to control it, but to monitor it and diagnose faults. The system proposes gathers ECU data in a mobile device e.g. smartphone, laptop; which then sends the data to the cloud platform, where big data analysis is performed. The mobile devices acts as a network broker between the ECUs and the centralized remote cloud.

Authors in [60] propose two mobile cloud applications for university environments. Specifically, the first is related to fire detection and the second to traffic monitoring. Both applications follow the same infrastructure and architecture. Sensors are connected to mobile devices, which are consumers of cloud services.

2.1 Mobile Computational Offloading

There are three main types of mobile computational offloading regarding the type of cloud environments being used. The type of the cloud offloading system being implemented affects the round-trip time and determines the configurational granularity. The three types are listed as follows. First, public (remote) cloud offloading, having the coarsest granularity in configuration, where computational heavy tasks are performed on a remote cloud server and their results are transmitted back to the mobile device. Second, cloudlet offloading, having a medium level of granularity in configuration, where a reduced cloud environment located close to the user can be used to perform computational heavy tasks instead of relegating them to the remote cloud or, to facilitate exchange of data between the client and the remote cloud [49, 59, 60]. Thus, reducing overall round-trip time. Third, ad-hoc cloud offloading, having the finest granularity in configuration, where a cloud environment is conformed of a heterogeneous set of mobile devices connected e.g. via WiFi, that share their computational resources thus enabling the relegation of tasks to others user thus increasing utilization. The three types described can be used in combination to optimize utilization, runtime, round-trip time and battery life.

Authors in [31] propose the Femtocloud system, which is a reduced cloud environment composed of mobile devices. The femtocloud system provides a highly dynamic, self-configuring and multi-tenant mobile cloud trough a cluster of mobile devices. Femtoclouds are also referred to as ad-hoc clouds, as mentioned above. Systems such as femtoclouds, where heterogeneous mobile devices share resources, are in need of a task scheduling algorithm in order to properly divide the workload and migrate the resources. [42] claims that in an ad-hoc cloud environment, pooled resources constitute a supercomputing node. Authors in [42] propose an implementation of the ad-hoc cloud architecture in an automated video surveillance application. Such application features object detection, tracking and classification, behavior analysis and action task. They also propose a task allocation scheme for the Femtocloud system.

[49] introduces the design and implementation of Context Aware Mobile Cloud Services (CAMCS), a middleware framework designed to deliver an integrated experience of the mobile cloud services to the user. CAMCS features cloud personal assistant task models, context-awareness with context processor and, service discovery and consumption.

The COMPSs [40] programming model aims to allow developers to code applications without awareness of parallelism or infrastructure details. Authors in [39] propose an infrastructure unaware framework named COMPSs-Mobile that allows developers to code regular Android applications that are transparently parallelized and partially offloaded to remote cloud resources. The COMPSs-Mobile framework features an offloading mechanism that determines if a task should run locally or remotely depending on factors such as round-trip time, delay and energy consumption. [39] points out that data transfers are one of the most influential factors on the economic, energetic and temporal cost of running and application.

2.2 Serverless and Microservices Architectures

One representative example of cloud services are serverless functions. Serverless functions are functions that run on demand on servers transparent both to the end user as to the developer, with minimal deployment effort. [46] points out no server deployment and management are needed. Serverless functions are invoked as a best-effort service [46] and are event triggered [32]. Serverless functions allow computational intensive code to be executed on a completely abstract infrastructure handled by the cloud service provider.

Serverless architecture implements code offloading via serverless functions in order to deploy web/mobile applications with features beyond the limitations of the clients.

Serverless functions abstract infrastructure, thus they can be classified as Functions as a Service [33], which is a type of Platform as a Service. Main provider's serverless functions services include Microsoft's Azure Functions, Amazon's AWS Lambda, Google Cloud Functions and PubNub Functions. Google also provides Firebase Functions, which were conceived to act as a backend for mobile applications. Currently it supports both mobile and web applications.

Since serverless architecture allows decoupling functions from the main codebase to specific functions that run remotely, they can be used on implementations of microservices architecture. The microservices architecture focuses on decoupling an application's codebase so that each microservice has only one functionality and orchestrating the different microservices to act as a distributed whole. In a microservices architecture, features of the application are deployed separately as microservices following the single responsibility principle [51]. [7] points out each microservice should be flexible and provide a clear API for composability. The operation of a Microservices architecture is controlled by an orchestrator microservice. Microservices architecture is a paradigm that aims to overcome the limitations of traditional monolithic architectures. A monolith is considered to be a program or application that depends on a single complex codebase, where there exists little feature independency as compared to a microservices application or system. In a monolithic architecture the complete set of features of a program is deployed as a whole [51].

2.3 Native vs Hybrid Mobile Development Frameworks

Currently there exists a wide range of options regarding the mobile development framework to be used. On one hand, native applications are those developed for a specific operating system [48]. The main advantage of native development is that the application can fully exploit the client capabilities e.g. sensors, cameras. Native Android development is written in Java or Kotlin (which runs on the java virtual machine and was designed with null safety as a priority) using the Android Studio development environment. Native iOS development is written in Objective-C or Swift (designed by Apple) using the Xcode development environment [9]. The downside of native mobile application development is that for an application to work in both platforms, two different codebases need to be developed and maintained [16]. [21] indicates migrating an application from an operative system to another is not straightforward as translating the code as differences such as languages, libraries, tools, design principles and specific hardware features.

On the other hand, hybrid development refers to developing cross-platform applications i.e. applications that can run on different operating systems with little to no changes on the code depending on the platform. Hybrid mobile development is mainly carried out within a framework. There are two types of hybrid mobile development frameworks according to the compilation process. They are also referred to as Cross-Platform development frameworks [9]. First, web based mobile frameworks such as Ionic and Cordova aim to provide a lightweight mobile experience by using a single codebase (intended to fit any platform) which consists on a WebView bundled inside an application installer. Web based mobile applications are written using HTML, CSS and JavaScript [9], as any web application [54]. These applications have the advantage of requiring little storage and processing capacity, since most of the codebase resides on the cloud and the client's only task is to render the view and passing events to the cloud. However, their main drawback consists on the fact that they cannot access the full potential of mobile devices since they are utterly device agnostic. This limitation of web based mobile applications difficulties drawing the line between a web based mobile application and a progressive web application (PWA) [9]. Second, in order to overcome the limitations of web based mobile applications while leveraging the extended capabilities of native development, native-hybrid frameworks arose. Native-hybrid mobile development frameworks aim to develop a single codebase that on building transpiles to native language, thus being able to exploit the full capabilities of the client. Representative examples of native-hybrid frameworks are: Xamarin [63], developed by Microsoft, where mobile applications are developed using C that is later transpiled at runtime to Objective-C in the case of iOS and Java in the case of Android, with a market share of 14% [17] of cross-platform mobile development frameworks as of 2020 ; React Native [50], developed by Facebook based on the web application framework React, where mobile apps are written using JavaScript and CSS (HTML is embedded in the JavaScript code via JSX [28]), with a 42% market share; and Flutter [25], developed by Google, with a market share of 39%, where applications are developed using Dart, a language specifically designed by Google to develop reactive applications, both mobile and web, based on the object oriented programming principles of Java and the versatility of JavaScript.

2.4 Dart and Flutter

Dart was first introduced by Google as a programming language in late 2011, defined by [19] as a class-based optionally typed programming language for building web applications. Currently, according to [61] types in dart are not optional but inferred by the Dart Analyzer. [20] also points out Dart has become a general purpose programming language. The author in [19] identifies three main goals that Dart aims to achieve: create a structured yet flexible language for web programming, making it feel familiar and natural to developers thus easy to learn, and ensuring delivery of high performance on all modern web browsers and environments; ranging from client devices i.e. front end, to server-side execution i.e. back-end. [19, 36] indicate that Dart code can be executed in two ways: first, on a native virtual machine or; second, using a JavaScript engine that translates Dart code into JavaScript.

According to the official Dart documentation [36], Dart is an object-oriented, class-based, garbage-collected language with a C-style syntax i.e. extends concepts of the ALGOL language family [3]. [20], that shares many features of Java and C, and others with JavaScript. Dart supports interfaces, mixins, abstract classes, reified generics and type inference i.e. omitted type annotations are resolved implicitly by the analyzer. [20] affirms Dart allows developers to provide type annotations selectively thus determining which parts

Serverless functions abstract infrastructure, thus they can be classified as Functions as a Service [33], which is a type of Platform as a Service. Main provider's serverless functions services include Microsoft's Azure Functions, Amazon's AWS Lambda, Google Cloud Functions and PubNub Functions. Google also provides Firebase Functions, which were conceived to act as a backend for mobile applications. Currently it supports both mobile and web applications.

Since serverless architecture allows decoupling functions from the main codebase to specific functions that run remotely, they can be used on implementations of microservices architecture. The microservices architecture focuses on decoupling an application's codebase so that each microservice has only one functionality and orchestrating the different microservices to act as a distributed whole. In a microservices architecture, features of the application are deployed separately as microservices following the single responsibility principle [51]. [7] points out each microservice should be flexible and provide a clear API for composability. The operation of a Microservices architecture is controlled by an orchestrator microservice. Microservices architecture is a paradigm that aims to overcome the limitations of traditional monolithic architectures. A monolith is considered to be a program or application that depends on a single complex codebase, where there exists little feature independency as compared to a microservices application or system. In a monolithic architecture the complete set of features of a program is deployed as a whole [51].

2.3 Native vs Hybrid Mobile Development Frameworks

Currently there exists a wide range of options regarding the mobile development framework to be used. On one hand, native applications are those developed for a specific operating system [48]. The main advantage of native development is that the application can fully exploit the client capabilities e.g. sensors, cameras. Native Android development is written in Java or Kotlin (which runs on the java virtual machine and was designed with null safety as a priority) using the Android Studio development environment. Native iOS development is written in Objective-C or Swift (designed by Apple) using the Xcode development environment [9]. The downside of native mobile application development is that for an application to work in both platforms, two different codebases need to be developed and maintained [16]. [21] indicates migrating an application from an operative system to another is not straightforward as translating the code as differences such as languages, libraries, tools, design principles and specific hardware features.

On the other hand, hybrid development refers to developing cross-platform applications i.e. applications that can run on different operating systems with little to no changes on the code depending on the platform. Hybrid mobile development is mainly carried out within a framework. There are two types of hybrid mobile development frameworks according to the compilation process. They are also referred to as Cross-Platform development frameworks [9]. First, web based mobile frameworks such as Ionic and Cordova aim to provide a lightweight mobile experience by using a single codebase (intended to fit any platform) which consists on a WebView bundled inside an application installer. Web based mobile applications are written using HTML, CSS and JavaScript [9], as any web application [54]. These applications have the advantage of requiring little storage and processing capacity, since most of the codebase resides on the cloud and the client's only task is to render the view and passing events to the cloud. However, their main drawback consists on the fact that they cannot access the full potential of mobile devices since they are utterly device agnostic. This limitation of web based mobile applications difficulties drawing the line between a web based mobile application and a progressive web application (PWA) [9]. Second, in order to overcome the limitations of web based mobile applications while leveraging the extended capabilities of native development, native-hybrid frameworks arose. Native-hybrid mobile development frameworks aim to develop a single codebase that on building transpiles to native language, thus being able to exploit the full capabilities of the client. Representative examples of native-hybrid frameworks are: Xamarin [63], developed by Microsoft, where mobile applications are developed using C that is later transpiled at runtime to Objective-C in the case of iOS and Java in the case of Android, with a market share of 14% [17] of cross-platform mobile development frameworks as of 2020 ; React Native [50], developed by Facebook based on the web application framework React, where mobile apps are written using JavaScript and CSS (HTML is embedded in the JavaScript code via JSX [28]), with a 42% market share; and Flutter [25], developed by Google, with a market share of 39%, where applications are developed using Dart, a language specifically designed by Google to develop reactive applications, both mobile and web, based on the object oriented programming principles of Java and the versatility of JavaScript.

2.4 Dart and Flutter

Dart was first introduced by Google as a programming language in late 2011, defined by [19] as a class-based optionally typed programming language for building web applications. Currently, according to [61] types in dart are not optional but inferred by the Dart Analyzer. [20] also points out Dart has become a general purpose programming language. The author in [19] identifies three main goals that Dart aims to achieve: create a structured yet flexible language for web programming, making it feel familiar and natural to developers thus easy to learn, and ensuring delivery of high performance on all modern web browsers and environments; ranging from client devices i.e. front end, to server-side execution i.e. back-end. [19, 36] indicate that Dart code can be executed in two ways: first, on a native virtual machine or; second, using a JavaScript engine that translates Dart code into JavaScript.

According to the official Dart documentation [36], Dart is an object-oriented, class-based, garbage-collected language with a C-style syntax i.e. extends concepts of the ALGOL language family [3]. [20], that shares many features of Java and C, and others with JavaScript. Dart supports interfaces, mixins, abstract classes, reified generics and type inference i.e. omitted type annotations are resolved implicitly by the analyzer. [20] affirms Dart allows developers to provide type annotations selectively thus determining which parts

of the program are statically typed. [34] points out that despite being object oriented, Dart can be used in functional programming. Hence, Dart is classified as a multi-paradigm programming language or general purpose language [20].

Flutter [25] is an open source cross-platform user interface software development framework created by Google. Flutter is used in the development of applications for web browsers, Android, iOS, iPadOS, Windows, MacOS and Google's Fuchsia. It aims to provide cross-platform applications from a single codebase with little alterations regarding the running platform e.g. certain additional configurations are needed when developing a mobile app, for Android and iOS. Flutter is designed to allow code reuse across operating systems, while allowing applications to access the underlying services of devices.

2.5 Mobile Application Design Patterns

Most industry-level mobile applications follow at least an application design pattern in order to develop and maintain an organized codebase. Design patterns are intended to facilitate code reusability and readability, thus enabling developers to write better code that would be possibly useful in future projects. Design patterns are multi-tiered in nature and aim to provide separation of concerns e.g. no business logic should be implemented on the view code; in applications so as to increase modularity, scalability, maintainability, reusability, readability [18, 45, 52].

Common Android and iOS design patterns include Model View Controller (MVC), Model View ViewModel (MVVM) and Model View Presenter (MVP). Flutter introduces additionally the Business Logic Components (BLoC) pattern. The patterns aforementioned feature a three tier architecture composed of the following entities [43].

First, the Model layer is where the data administration occurs. The Model layer shapes the data via Classes (models) and is in charge of handling databases. It is not concerned on how the data should be presented and operates with "raw" data. Authors in [43] point out the model layer should have a standardized universal interface so as to be consumed by different components of the application. The model layer concept is understood equally in the context of MVC and MVVM.

Second, The View layer (screens) is in charge of displaying the output of the application. In the case of MVC, the View layer is in charge of formatting the raw data it takes as an input from the Model layer and displaying it on a graphical interface. In the case of MVVM, the view layer is completely unconcerned with the processing of data. It should only display the output of the application trough a graphic interface.

Third, the third layer differences are more prominent between the two traditional models considered. On one hand, the Controller layer is in charge of managing/handling all the application processes [**mvc**]. The Controller layer receives external requests and fulfills its requirements. After that, it transfers the data received to the Model in order to display it properly on the view. On the other hand, the ViewModel layer is an abstraction of the view and a wrapper for the modeled data. It can be considered as a bridge between the model and the view, and contains functions and methods that the View layer can use to control the Model [18, 45, 52].

The Business Logic Components (BLoC) [10] is a design pattern introduced for Flutter and later used by Angular [6]. BLoC also features a three tier architecture composed by UI layer, BLoC layer and Data layer. It should be taken into consideration that BLoC offers a better separation of concerns that others architectures mentioned. The Data layer is in charge of retrieving/manipulating data obtained by different sources. It can be decomposed in two parts: data provider, that provides raw data and should be generic and versatile, usually contains CRUD operations and; repositories. The repository layer is a wrapper around one or more providers that abstracts them and communicates to the bloc layer. The Business Logic layer is in charge of the state of the application, responding input from the presentation layer with new states. This layer can depend on at least one repository to retrieve the data needed. The UI or Presentation layer is in charge to determine the best way to display the data based on at least a bloc state. Besides handling user interaction, it manages the application lifecycle.

The aforementioned architecture is illustrated on Figure 1 [10]. Notice that the UI layer exchanges messages with the bloc layer being completely unaware of the data layer. In a similar fashion, the data layer replies to blocs requests without considering the UI. The UI passes events to the bloc layer that are mapped to states, which are passed back to the UI to render graphically. The bloc layer requests data from the data layer. A bloc component can be understood as a stream of states triggered by events.

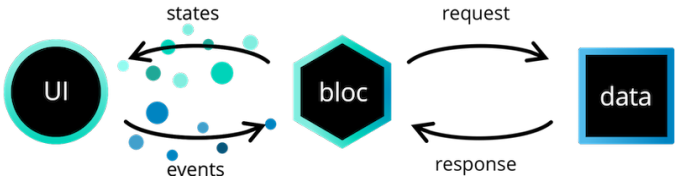

Fig. 1. High-level Architecture of BLoC

3 Proposed Work

The proposed system is composed of three main components: a mobile client (front-end), a remote cloud database and cloud serverless functions (back-end). The technology stack used in the implementation of the system was:

- Flutter [25].
- Cloud Firestore [14], a NoSQL Document-based Database as a Service provided by the PaaS suite, Firebase [23].
- Cloud Functions [15], a serverless Functions as a Service provided by Firebase.
- Firebase Authentication [24], an Identity as a Service provided by Firebase.
- Google Maps Platform [27]

The mobile application (front-end) was developed using the Flutter framework in order to maximize the target audience without writing completely different codebases for the two main mobile OS platforms i.e. Android and iOS.

of the program are statically typed. [34] points out that despite being object oriented, Dart can be used in functional programming. Hence, Dart is classified as a multi-paradigm programming language or general purpose language [20].

Flutter [25] is an open source cross-platform user interface software development framework created by Google. Flutter is used in the development of applications for web browsers, Android, iOS, iPadOS, Windows, MacOS and Google's Fuchsia. It aims to provide cross-platform applications from a single codebase with little alterations regarding the running platform e.g. certain additional configurations are needed when developing a mobile app, for Android and iOS. Flutter is designed to allow code reuse across operating systems, while allowing applications to access the underlying services of devices.

2.5 Mobile Application Design Patterns

Most industry-level mobile applications follow at least an application design pattern in order to develop and maintain an organized codebase. Design patterns are intended to facilitate code reusability and readability, thus enabling developers to write better code that would be possibly useful in future projects. Design patterns are multi-tiered in nature and aim to provide separation of concerns e.g. no business logic should be implemented on the view code; in applications so as to increase modularity, scalability, maintainability, reusability, readability [18, 45, 52].

Common Android and iOS design patterns include Model View Controller (MVC), Model View ViewModel (MVVM) and Model View Presenter (MVP). Flutter introduces additionally the Business Logic Components (BLoC) pattern. The patterns aforementioned feature a three tier architecture composed of the following entities [43].

First, the Model layer is where the data administration occurs. The Model layer shapes the data via Classes (models) and is in charge of handling databases. It is not concerned on how the data should be presented and operates with "raw" data. Authors in [43] point out the model layer should have a standardized universal interface so as to be consumed by different components of the application. The model layer concept is understood equally in the context of MVC and MVVM.

Second, The View layer (screens) is in charge of displaying the output of the application. In the case of MVC, the View layer is in charge of formatting the raw data it takes as an input from the Model layer and displaying it on a graphical interface. In the case of MVVM, the view layer is completely unconcerned with the processing of data. It should only display the output of the application trough a graphic interface.

Third, the third layer differences are more prominent between the two traditional models considered. On one hand, the Controller layer is in charge of managing/handling all the application processes [**mvc**]. The Controller layer receives external requests and fulfills its requirements. After that, it transfers the data received to the Model in order to display it properly on the view. On the other hand, the ViewModel layer is an abstraction of the view and a wrapper for the modeled data. It can be considered as a bridge between the model and the view, and contains functions and methods that the View layer can use to control the Model [18, 45, 52].

The Business Logic Components (BLoC) [10] is a design pattern introduced for Flutter and later used by Angular [6]. BLoC also features a three tier architecture composed by UI layer, BLoC layer and Data layer. It should be taken into consideration that BLoC offers a better separation of concerns that others architectures mentioned. The Data layer is in charge of retrieving/manipulating data obtained by different sources. It can be decomposed in two parts: data provider, that provides raw data and should be generic and versatile, usually contains CRUD operations and; repositories. The repository layer is a wrapper around one or more providers that abstracts them and communicates to the bloc layer. The Business Logic layer is in charge of the state of the application, responding input from the presentation layer with new states. This layer can depend on at least one repository to retrieve the data needed. The UI or Presentation layer is in charge to determine the best way to display the data based on at least a bloc state. Besides handling user interaction, it manages the application lifecycle.

The aforementioned architecture is illustrated on Figure 1 [10]. Notice that the UI layer exchanges messages with the bloc layer being completely unaware of the data layer. In a similar fashion, the data layer replies to blocs requests without considering the UI. The UI passes events to the bloc layer that are mapped to states, which are passed back to the UI to render graphically. The bloc layer requests data from the data layer. A bloc component can be understood as a stream of states triggered by events.

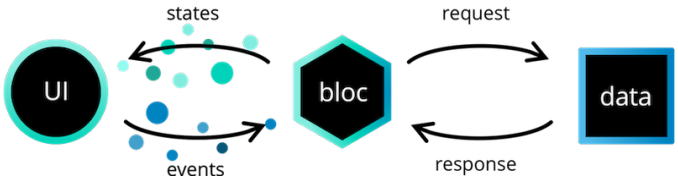

Fig. 1. High-level Architecture of BLoC

3 Proposed Work

The proposed system is composed of three main components: a mobile client (front-end), a remote cloud database and cloud serverless functions (back-end). The technology stack used in the implementation of the system was:

- Flutter [25].
- Cloud Firestore [14], a NoSQL Document-based Database as a Service provided by the PaaS suite, Firebase [23].
- Cloud Functions [15], a serverless Functions as a Service provided by Firebase.
- Firebase Authentication [24], an Identity as a Service provided by Firebase.
- Google Maps Platform [27]

The mobile application (front-end) was developed using the Flutter framework in order to maximize the target audience without writing completely different codebases for the two main mobile OS platforms i.e. Android and iOS.

The mobile application was coded using Dart following the BLoC design architecture. BLoC allowed to decouple the mobile application into completely independent features, thus facilitating scalability and maintainability. The main features of the mobile application are:

- User authentication via email/password and anonymous authentication.
- Retrieving user GPS coordinates periodically and posting the coordinates to a remote database, Firestore.
- Rendering and displaying demographic heat map based on the cloud function results obtained.

In terms of mobile development architecture, implemented BLoC design pattern consisted on: two repositories, one for authentication and one for the location and heat map data, that handles both the GPS of the device and the communication with the database and the serverless function; three blocs, the first for handling authentication status and switching screens based on it, the second for handling user location gathering and uploading, the third for invoking the cloud function and downloading the heat map generated; and two screens: an authentication one and a map view tabbed with controls. The before mentioned code architecture can be seen on Figure 2.

The back-end business logic was implemented using JavaScript's engine Node.js [47] functions on Firebase Cloud Functions. The remote database used was a Firestore instance. The main features of the back-end application are:

- Storing every user's location, which is updated periodically by the mobile app.
- Storing every user's heat map, which is generated on request by the mobile application through a cloud function.
- Generate a demographic heat map based on the locations of other users in the vicinity of the user, determined by a radius that the user can configure. The Cloud function invoked collects from the database instance the list of locations in the vicinity of the client and generates a heat map that is sent back to the user.

Figure 3 depicts the architecture of the system implemented and the data flow. It can be seen that users update their GPS coordinates on the remote database via the mobile client. The mobile client also triggers periodically a cloud function instance to generate a heat map that is stored on the database, then retrieved by the user.

Figure 4 shows the output of the system on an android emulator. The results were calculated using a randomly generated set of 1000 users (around 10% of the usual population of the university) each with its coordinates generated using a Gaussian distribution.

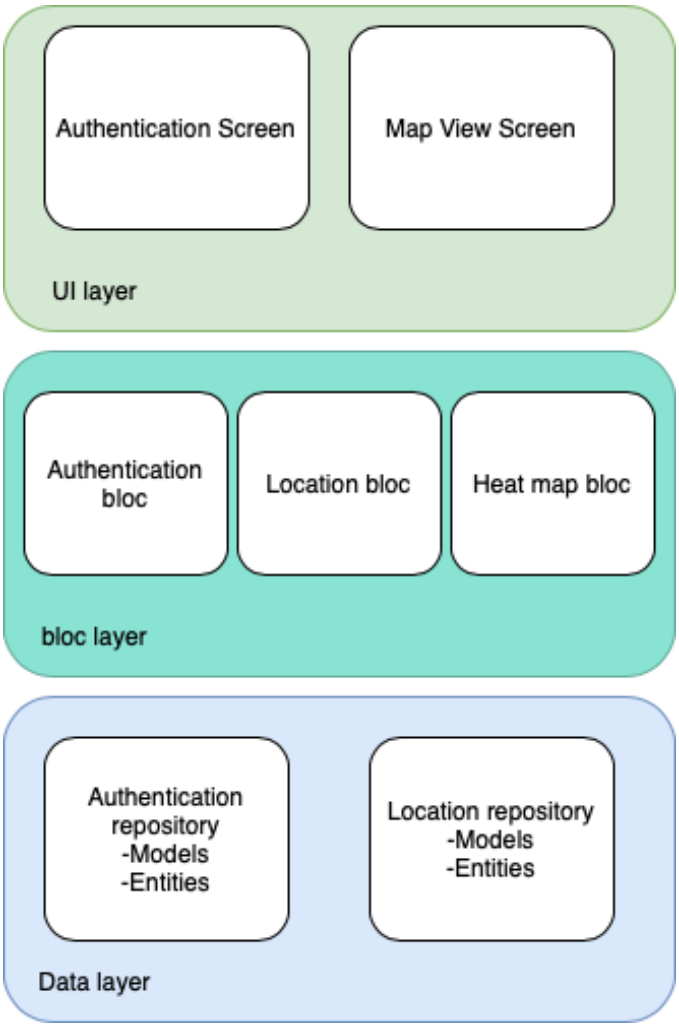

Fig. 2. High-level Code Architecture

4 Conclusions and Future Work

Code offloading can improve the mobile user experience (UX) by performing computationally intensive tasks e.g. searching the database for the other users in the vicinity of a given user, in remote cloud servers, while preserving battery life of mobile devices and reducing the delay between input (request) and output (response).

This paper gives insights regarding the design, implementation and features of Project HeatMaps, a mobile cross-platform cloud computing application that allows users to read demographic heat maps generated based on their location in order to avoid highly concurred zones. This application is considered to be useful in the context of COVID-19 since it helps users step one step further away from contagion. The application makes use of the crowdsourcing principle to harvest relevant data from users.

The system architecture proposed on this paper can be replicated or extended with different features. BLoC design pattern highly facilitates the scalability of the application both vertically i.e. improving the existing features and horizontally i.e. adding new features. Also, the NoSQL document based database, Firestore, can be changed for a graph database in order to find results i.e. locations of other users in the vicinity, faster.

The overall system designed and developed is not industry-ready. It should be noticed that it needs further development and testing in order to be deployed in production.

To the best of our knowledge, there exists no other re-

The mobile application was coded using Dart following the BLoC design architecture. BLoC allowed to decouple the mobile application into completely independent features, thus facilitating scalability and maintainability. The main features of the mobile application are:

- User authentication via email/password and anonymous authentication.
- Retrieving user GPS coordinates periodically and posting the coordinates to a remote database, Firestore.
- Rendering and displaying demographic heat map based on the cloud function results obtained.

In terms of mobile development architecture, implemented BLoC design pattern consisted on: two repositories, one for authentication and one for the location and heat map data, that handles both the GPS of the device and the communication with the database and the serverless function; three blocs, the first for handling authentication status and switching screens based on it, the second for handling user location gathering and uploading, the third for invoking the cloud function and downloading the heat map generated; and two screens: an authentication one and a map view tabbed with controls. The before mentioned code architecture can be seen on Figure 2.

The back-end business logic was implemented using JavaScript's engine Node.js [47] functions on Firebase Cloud Functions. The remote database used was a Firestore instance. The main features of the back-end application are:

- Storing every user's location, which is updated periodically by the mobile app.
- Storing every user's heat map, which is generated on request by the mobile application through a cloud function.
- Generate a demographic heat map based on the locations of other users in the vicinity of the user, determined by a radius that the user can configure. The Cloud function invoked collects from the database instance the list of locations in the vicinity of the client and generates a heat map that is sent back to the user.

Figure 3 depicts the architecture of the system implemented and the data flow. It can be seen that users update their GPS coordinates on the remote database via the mobile client. The mobile client also triggers periodically a cloud function instance to generate a heat map that is stored on the database, then retrieved by the user.

Figure 4 shows the output of the system on an android emulator. The results were calculated using a randomly generated set of 1000 users (around 10% of the usual population of the university) each with its coordinates generated using a Gaussian distribution.

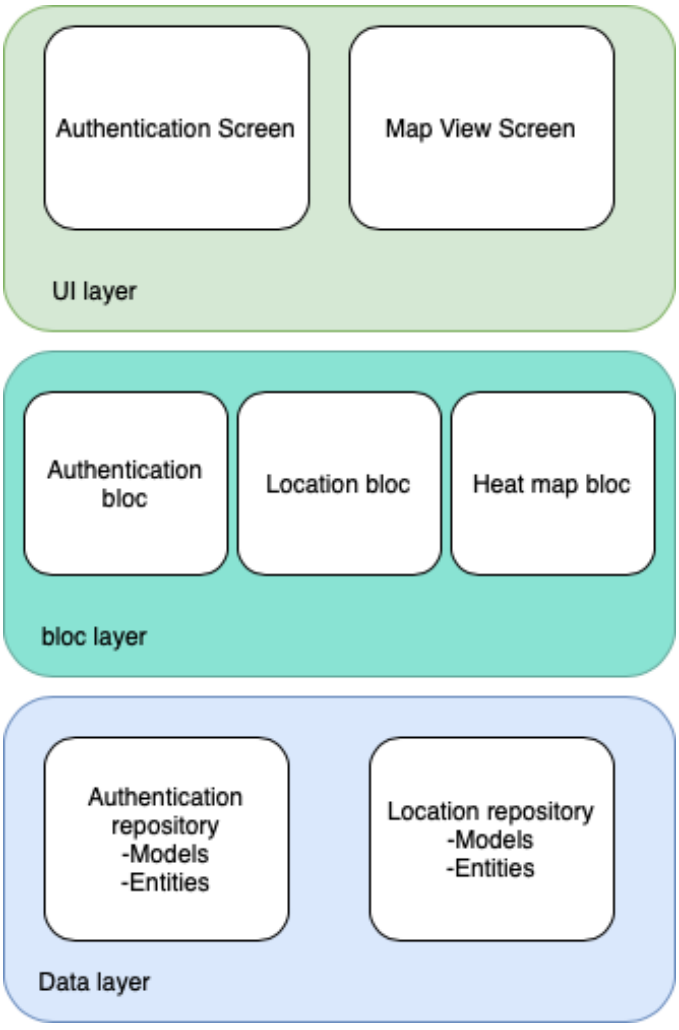

Fig. 2. High-level Code Architecture

4 Conclusions and Future Work

Code offloading can improve the mobile user experience (UX) by performing computationally intensive tasks e.g. searching the database for the other users in the vicinity of a given user, in remote cloud servers, while preserving battery life of mobile devices and reducing the delay between input (request) and output (response).

This paper gives insights regarding the design, implementation and features of Project HeatMaps, a mobile cross-platform cloud computing application that allows users to read demographic heat maps generated based on their location in order to avoid highly concurred zones. This application is considered to be useful in the context of COVID-19 since it helps users step one step further away from contagion. The application makes use of the crowdsourcing principle to harvest relevant data from users.

The system architecture proposed on this paper can be replicated or extended with different features. BLoC design pattern highly facilitates the scalability of the application both vertically i.e. improving the existing features and horizontally i.e. adding new features. Also, the NoSQL document based database, Firestore, can be changed for a graph database in order to find results i.e. locations of other users in the vicinity, faster.

The overall system designed and developed is not industry-ready. It should be noticed that it needs further development and testing in order to be deployed in production.

To the best of our knowledge, there exists no other re-

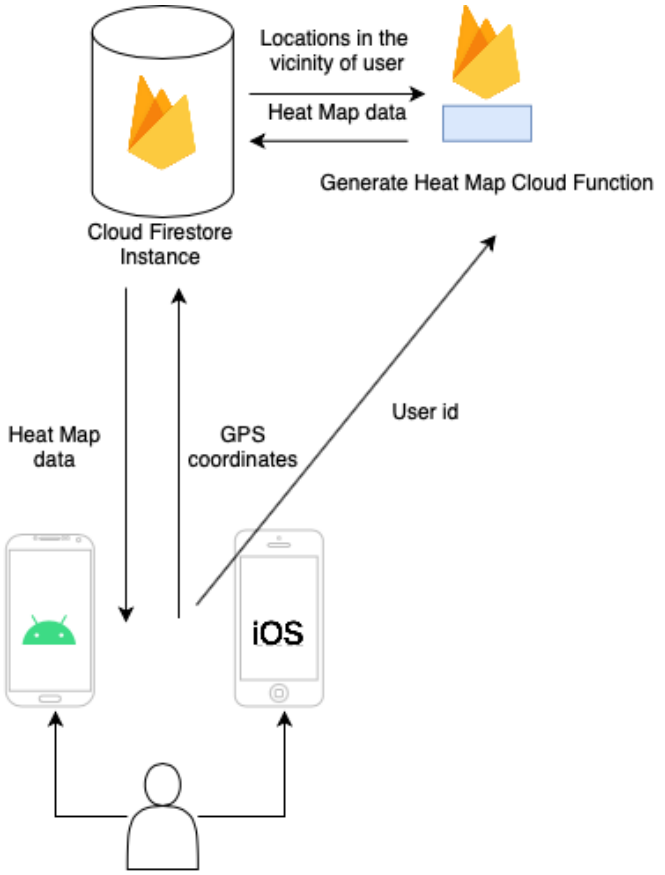

Fig. 3. Overall system architecture and data flow.

search discussing the BLoC design pattern. Further research could be aimed at exploiting the potential and capabilities of the BLoC architecture pattern.

References

[1] S. Abolfazli, A. Gani, and M. Chen. "HMCC: A Hybrid Mobile Cloud Computing Framework Exploiting Heterogeneous Resources". In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. Mar. 2015, pp. 157–162. DOI: 10.1109/MobileCloud.2015.28.

[2] R. G. Alakbarov and O. R. Alakbarov. "Mobile clouds computing: Current state, architecture and problems". In: *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. Feb. 2017, pp. 1–6. DOI: 10.1109/ICECCT.2017.8117874.

[3] *Algol Family*. URL: http://wiki.c2.com/?AlgolFamily (visited on 11/25/2020).

[4] S Anandamurugan, T Priyaa, and M. C. Arvind Babu. *Cloud computing: an innovative technology for Linux and Android platforms*. English. OCLC: 992743019. 2017. ISBN: 978-93-85750-78-6.

[5] *Android vs iOS market share 2023*. en. URL: https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/ (visited on 11/24/2020).

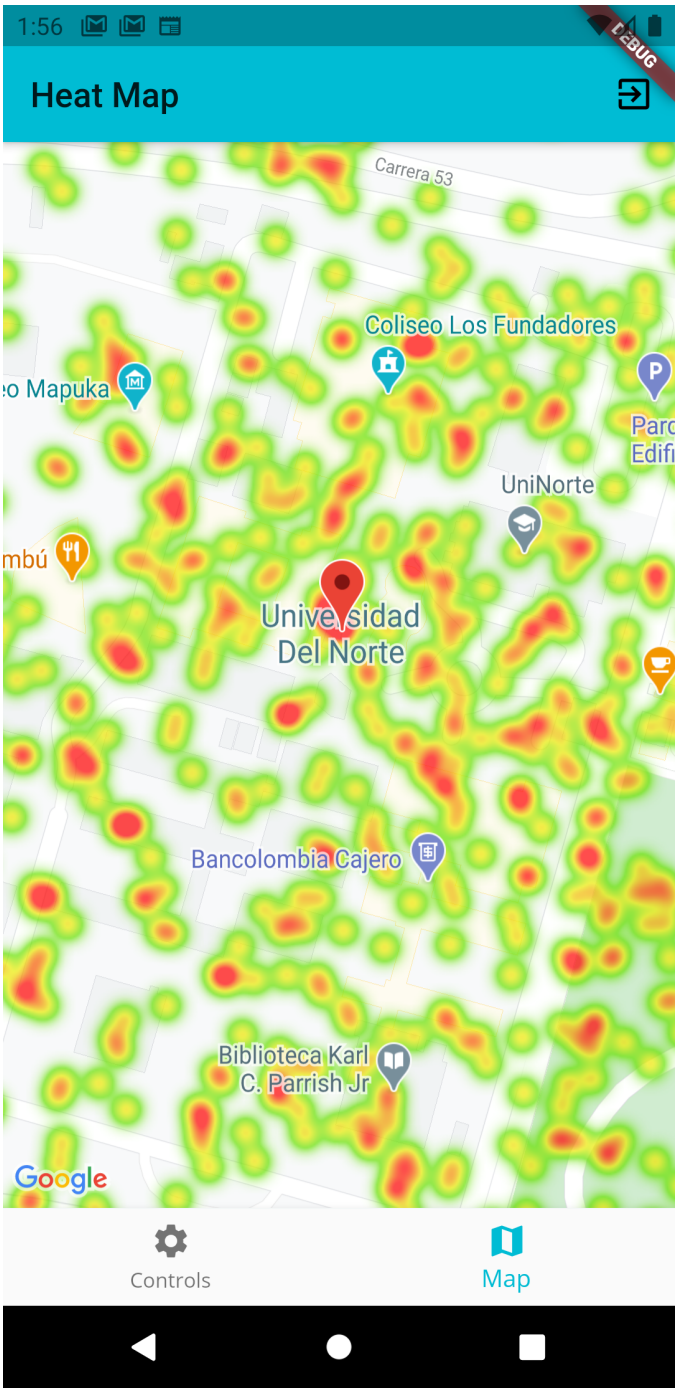

Fig. 4. User interface of Project HeatMaps

[6] *Angular*. URL: https://angular.io/ (visited on 11/24/2020).

[7] Harold Aragon et al. "Workload Characterization of a Software-as-a-Service Web Application Implemented with a Microservices Architecture". In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW '19. event-place: San Francisco, USA. New York, NY, USA: Association for Computing Machinery, 2019, pp. 746–750. ISBN: 978-1-4503-6675-5. DOI: 10.1145/3308560.3316466. URL: https://doi.org/10.1145/3308560.3316466.

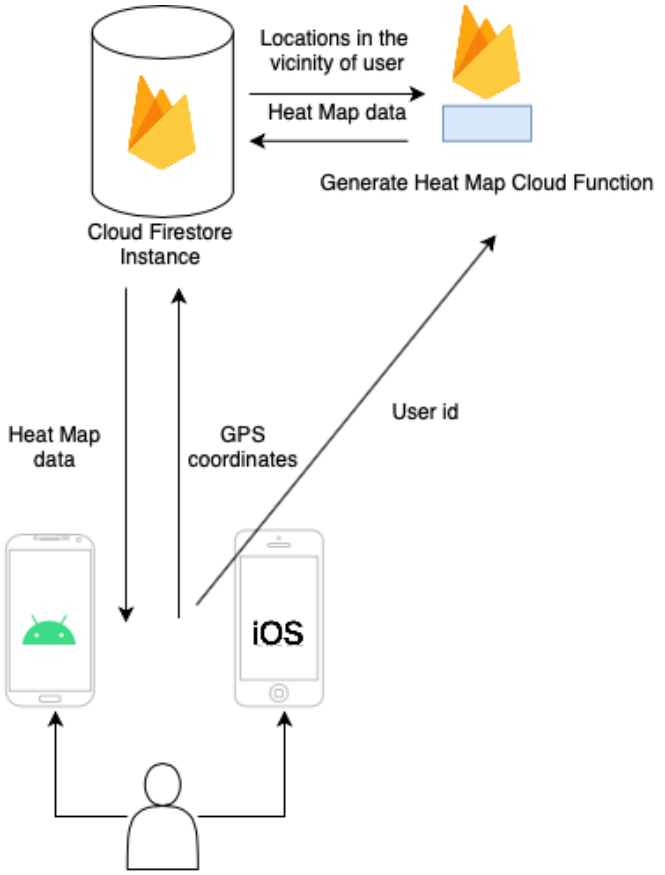

Fig. 3. Overall system architecture and data flow.

search discussing the BLoC design pattern. Further research could be aimed at exploiting the potential and capabilities of the BLoC architecture pattern.

References

[1] S. Abolfazli, A. Gani, and M. Chen. "HMCC: A Hybrid Mobile Cloud Computing Framework Exploiting Heterogeneous Resources". In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. Mar. 2015, pp. 157–162. DOI: 10.1109/MobileCloud.2015.28.

[2] R. G. Alakbarov and O. R. Alakbarov. "Mobile clouds computing: Current state, architecture and problems". In: *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. Feb. 2017, pp. 1–6. DOI: 10.1109/ICECCT.2017.8117874.

[3] *Algol Family*. URL: http://wiki.c2.com/?AlgolFamily (visited on 11/25/2020).

[4] S Anandamurugan, T Priyaa, and M. C. Arvind Babu. *Cloud computing: an innovative technology for Linux and Android platforms*. English. OCLC: 992743019. 2017. ISBN: 978-93-85750-78-6.

[5] *Android vs iOS market share 2023*. en. URL: https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/ (visited on 11/24/2020).

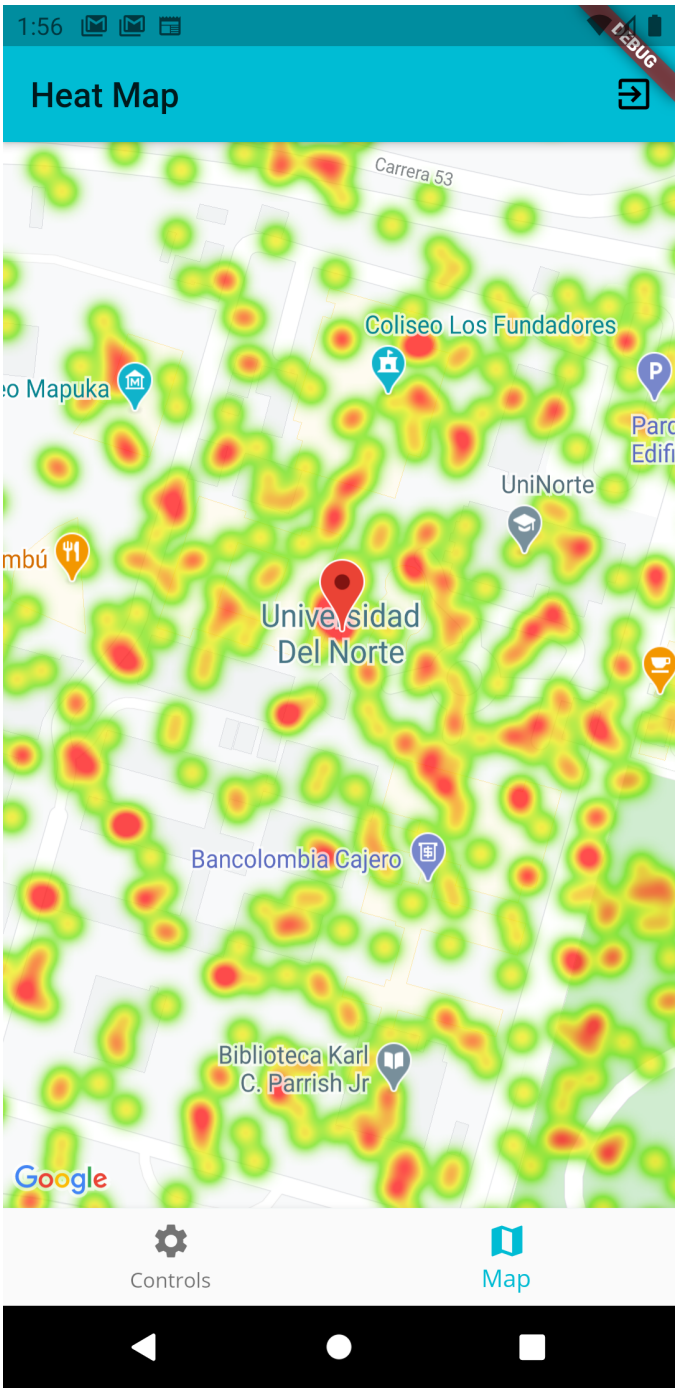

Fig. 4. User interface of Project HeatMaps

[6] *Angular*. URL: https://angular.io/ (visited on 11/24/2020).

[7] Harold Aragon et al. "Workload Characterization of a Software-as-a-Service Web Application Implemented with a Microservices Architecture". In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW '19. event-place: San Francisco, USA. New York, NY, USA: Association for Computing Machinery, 2019, pp. 746–750. ISBN: 978-1-4503-6675-5. DOI: 10.1145/3308560.3316466. URL: https://doi.org/10.1145/3308560.3316466.

[8] *AWS — Cloud Computing - Servicios de informática en la nube*. es-ES. URL: https://aws.amazon.com/es/ (visited on 11/24/2020).

[9] Andreas Biørn-Hansen, Tor-Morten Grønli, and Gheorghita Ghinea. "A Survey and Taxonomy of Core Concepts and Research Challenges in Cross-Platform Mobile Development". In: *ACM Comput. Surv.* 51.5 (Nov. 2018). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 0360-0300. DOI: 10.1145/3241739. URL: https://doi.org/10.1145/3241739.

[10] *Bloc*. URL: https://bloclibrary.dev/#/ (visited on 11/24/2020).

[11] *Canalys - The leading global technology market analyst firm*. en-US. URL: https://www.canalys.com/ (visited on 11/24/2020).

[12] *Cloud Computing Market Size, Share and Global Market Forecast to 2025 — COVID-19 Impact Analysis — MarketsandMarkets*. URL: https://www.marketsandmarkets.com/Market-Reports/cloud-computing-market-234.html (visited on 11/24/2020).

[13] *Cloud Computing Services — Microsoft Azure*. en. URL: https://azure.microsoft.com/en-us/ (visited on 11/24/2020).

[14] *Cloud Firestore*. es-419. URL: https://firebase.google.com/docs/firestore?hl=es-419 (visited on 11/24/2020).

[15] *Cloud Functions para Firebase*. es-419. URL: https://firebase.google.com/docs/functions?hl=es-419 (visited on 11/24/2020).

[16] Existek-Software Development Company. *Hybrid VS Native App: Which one to choose for your business?* en. Mar. 2020. URL: https://medium.com/existek/hybrid-vs-native-app-which-one-to-choose-for-your-business-e51542554078 (visited on 11/24/2020).

[17] *Cross-platform mobile frameworks used by global developers 2020*. en. URL: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/ (visited on 11/24/2020).

[18] Aymen Daoudi et al. "An Exploratory Study of MVC-Based Architectural Patterns in Android Apps". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. SAC '19. event-place: Limassol, Cyprus. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1711–1720. ISBN: 978-1-4503-5933-7. DOI: 10.1145/3297280.3297447. URL: https://doi.org/10.1145/3297280.3297447.

[19] *Dart: a language for structured web programming - The official Google Code blog*. Oct. 2011. URL: http://googlecode.blogspot.com/2011/10/dart-language-for-structured-web.html (visited on 11/25/2020).

[20] Erik Ernst et al. "Message Safety in Dart". In: *Proceedings of the 11th Symposium on Dynamic Languages*. DLS 2015. event-place: Pittsburgh, PA, USA. New York, NY, USA: Association for Computing Machinery, 2015, pp. 41–53. ISBN: 978-1-4503-3690-1. DOI: 10.1145/2816707.2816711. URL: https://doi.org/10.1145/2816707.2816711.

[21] Xiaochao Fan and Kenny Wong. "Migrating User Interfaces in Native Mobile Applications: Android to IOS". In: *Proceedings of the International Conference on Mobile Software Engineering and Systems*. MOBILESoft '16. event-place: Austin, Texas. New York, NY, USA: Association for Computing Machinery, 2016, pp. 210–213. ISBN: 978-1-4503-4178-3. DOI: 10.1145/2897073.2897101. URL: https://doi.org/10.1145/2897073.2897101.

[22] S. Farrugia. "Mobile Cloud Computing Techniques for Extending Computation and Resources in Mobile Devices". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Mar. 2016, pp. 1–10. DOI: 10.1109/MobileCloud.2016.26.

[23] *Firebase*. es-419. URL: https://firebase.google.com/?hl=es-419 (visited on 11/24/2020).

[24] *Firebase Authentication — Simple, free multi-platform sign-in*. es-419. URL: https://firebase.google.com/products/auth?hl=es-419 (visited on 11/24/2020).

[25] *Flutter - Beautiful native apps in record time*. en. URL: https://flutter.dev/ (visited on 11/24/2020).

[26] M. M. Fuad and D. Deb. "Cloud-Enabled Hybrid Architecture for In-Class Interactive Learning Using Mobile Device". In: *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Apr. 2017, pp. 149–152. DOI: 10.1109/MobileCloud.2017.15.

[27] *Google Maps Platform*. en. URL: https://developers.google.com/maps/documentation (visited on 11/24/2020).

[28] *Introducing JSX – React*. en. URL: https://reactjs.org/docs/introducing-jsx.html (visited on 11/24/2020).

[29] H. Jemal et al. "Cloud computing and mobile devices based system for healthcare application". In: *2015 IEEE International Symposium on Technology and Society (ISTAS)*. ISSN: 2158-3412. Nov. 2015, pp. 1–5. DOI: 10.1109/ISTAS.2015.7439407.

[30] P. Joshi et al. "Understanding the Challenges in Mobile Computation Offloading to Cloud through Experimentation". In: *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*. May 2015, pp. 158–159. DOI: 10.1109/MobileSoft.2015.43.

[31] K. Habak et al. "Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge". In: *2015 IEEE 8th International Conference on Cloud Computing*. Journal Abbreviation: 2015 IEEE 8th International Conference on Cloud Computing. July 2015, pp. 9–16. ISBN: 2159-6190. DOI: 10.1109/CLOUD.2015.12.

[32] Ali Kanso and Alaa Youssef. "Serverless: Beyond the Cloud". In: *Proceedings of the 2nd International Workshop on Serverless Computing*. WoSC '17. event-place: Las Vegas, Nevada. New York, NY, USA: Association for Computing Machinery, 2017, pp. 6–10. ISBN: 978-1-4503-5434-9. DOI: 10.1145/3154847.3154854. URL: https://doi.org/10.1145/3154847.3154854.

[33] Nima Kaviani, Dmitriy Kalinin, and Michael Maximilien. "Towards Serverless as Commodity: A Case of Knative". In: *Proceedings of the 5th International Workshop on Serverless Computing*. WOSC '19. event-place: Davis, CA, USA. New York, NY, USA: Association for Computing Machinery, 2019, pp. 13–18. ISBN: 978-1-4503-7038-7. DOI: 10.1145/3366623.3368135. URL: https://doi.org/10.1145/3366623.3368135.

[8] *AWS — Cloud Computing - Servicios de informática en la nube*. es-ES. URL: https://aws.amazon.com/es/ (visited on 11/24/2020).

[9] Andreas Biørn-Hansen, Tor-Morten Grønli, and Gheorghita Ghinea. "A Survey and Taxonomy of Core Concepts and Research Challenges in Cross-Platform Mobile Development". In: *ACM Comput. Surv.* 51.5 (Nov. 2018). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 0360-0300. DOI: 10.1145/3241739. URL: https://doi.org/10.1145/3241739.

[10] *Bloc*. URL: https://bloclibrary.dev/#/ (visited on 11/24/2020).

[11] *Canalys - The leading global technology market analyst firm*. en-US. URL: https://www.canalys.com/ (visited on 11/24/2020).

[12] *Cloud Computing Market Size, Share and Global Market Forecast to 2025 — COVID-19 Impact Analysis — MarketsandMarkets*. URL: https://www.marketsandmarkets.com/Market-Reports/cloud-computing-market-234.html (visited on 11/24/2020).

[13] *Cloud Computing Services — Microsoft Azure*. en. URL: https://azure.microsoft.com/en-us/ (visited on 11/24/2020).

[14] *Cloud Firestore*. es-419. URL: https://firebase.google.com/docs/firestore?hl=es-419 (visited on 11/24/2020).

[15] *Cloud Functions para Firebase*. es-419. URL: https://firebase.google.com/docs/functions?hl=es-419 (visited on 11/24/2020).

[16] Existek-Software Development Company. *Hybrid VS Native App: Which one to choose for your business?* en. Mar. 2020. URL: https://medium.com/existek/hybrid-vs-native-app-which-one-to-choose-for-your-business-e51542554078 (visited on 11/24/2020).

[17] *Cross-platform mobile frameworks used by global developers 2020*. en. URL: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/ (visited on 11/24/2020).

[18] Aymen Daoudi et al. "An Exploratory Study of MVC-Based Architectural Patterns in Android Apps". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. SAC '19. event-place: Limassol, Cyprus. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1711–1720. ISBN: 978-1-4503-5933-7. DOI: 10.1145/3297280.3297447. URL: https://doi.org/10.1145/3297280.3297447.

[19] *Dart: a language for structured web programming - The official Google Code blog*. Oct. 2011. URL: http://googlecode.blogspot.com/2011/10/dart-language-for-structured-web.html (visited on 11/25/2020).

[20] Erik Ernst et al. "Message Safety in Dart". In: *Proceedings of the 11th Symposium on Dynamic Languages*. DLS 2015. event-place: Pittsburgh, PA, USA. New York, NY, USA: Association for Computing Machinery, 2015, pp. 41–53. ISBN: 978-1-4503-3690-1. DOI: 10.1145/2816707.2816711. URL: https://doi.org/10.1145/2816707.2816711.

[21] Xiaochao Fan and Kenny Wong. "Migrating User Interfaces in Native Mobile Applications: Android to IOS". In: *Proceedings of the International Conference on Mobile Software Engineering and Systems*. MOBILESoft '16. event-place: Austin, Texas. New York, NY, USA: Association for Computing Machinery, 2016, pp. 210–213. ISBN: 978-1-4503-4178-3. DOI: 10.1145/2897073.2897101. URL: https://doi.org/10.1145/2897073.2897101.

[22] S. Farrugia. "Mobile Cloud Computing Techniques for Extending Computation and Resources in Mobile Devices". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Mar. 2016, pp. 1–10. DOI: 10.1109/MobileCloud.2016.26.

[23] *Firebase*. es-419. URL: https://firebase.google.com/?hl=es-419 (visited on 11/24/2020).

[24] *Firebase Authentication — Simple, free multi-platform sign-in*. es-419. URL: https://firebase.google.com/products/auth?hl=es-419 (visited on 11/24/2020).

[25] *Flutter - Beautiful native apps in record time*. en. URL: https://flutter.dev/ (visited on 11/24/2020).

[26] M. M. Fuad and D. Deb. "Cloud-Enabled Hybrid Architecture for In-Class Interactive Learning Using Mobile Device". In: *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Apr. 2017, pp. 149–152. DOI: 10.1109/MobileCloud.2017.15.

[27] *Google Maps Platform*. en. URL: https://developers.google.com/maps/documentation (visited on 11/24/2020).

[28] *Introducing JSX – React*. en. URL: https://reactjs.org/docs/introducing-jsx.html (visited on 11/24/2020).

[29] H. Jemal et al. "Cloud computing and mobile devices based system for healthcare application". In: *2015 IEEE International Symposium on Technology and Society (ISTAS)*. ISSN: 2158-3412. Nov. 2015, pp. 1–5. DOI: 10.1109/ISTAS.2015.7439407.

[30] P. Joshi et al. "Understanding the Challenges in Mobile Computation Offloading to Cloud through Experimentation". In: *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*. May 2015, pp. 158–159. DOI: 10.1109/MobileSoft.2015.43.

[31] K. Habak et al. "Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge". In: *2015 IEEE 8th International Conference on Cloud Computing*. Journal Abbreviation: 2015 IEEE 8th International Conference on Cloud Computing. July 2015, pp. 9–16. ISBN: 2159-6190. DOI: 10.1109/CLOUD.2015.12.

[32] Ali Kanso and Alaa Youssef. "Serverless: Beyond the Cloud". In: *Proceedings of the 2nd International Workshop on Serverless Computing*. WoSC '17. event-place: Las Vegas, Nevada. New York, NY, USA: Association for Computing Machinery, 2017, pp. 6–10. ISBN: 978-1-4503-5434-9. DOI: 10.1145/3154847.3154854. URL: https://doi.org/10.1145/3154847.3154854.

[33] Nima Kaviani, Dmitriy Kalinin, and Michael Maximilien. "Towards Serverless as Commodity: A Case of Knative". In: *Proceedings of the 5th International Workshop on Serverless Computing*. WOSC '19. event-place: Davis, CA, USA. New York, NY, USA: Association for Computing Machinery, 2019, pp. 13–18. ISBN: 978-1-4503-7038-7. DOI: 10.1145/3366623.3368135. URL: https://doi.org/10.1145/3366623.3368135.

[34] David Kopec. *Dart for absolute beginners*. The expert's voice in Web development. OCLC: ocn883364025. Berkeley, California: Apress, 2014. ISBN: 978-1-4302-6481-1 978-1-4302-6482-8.

[35] Dejan Kovachev, Yiwei Cao, and Ralf Klamma. "Mobile Cloud Computing: A Comparison of Application Models". In: *arXiv:1107.4940 [cs]* (July 2011). arXiv: 1107.4940. URL: http://arxiv.org/abs/1107.4940 (visited on 11/24/2020).

[36] *Language tour — Dart*. URL: https://dart.dev/guides/language/language-tour (visited on 11/25/2020).

[37] Lihui Lei et al. "MCloudDB: A Mobile Cloud Database Service Framework". In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. San Francisco, CA, USA: IEEE, Mar. 2015, pp. 6–15. ISBN: 978-1-4799-8977-5. DOI: 10.1109/MobileCloud.2015.30. URL: https://ieeexplore.ieee.org/document/7130864 (visited on 11/24/2020).

[38] Shuyu Li and Jerry Gao. "Moving from Mobile Databases to Mobile Cloud Data Services". In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. San Francisco, CA, USA: IEEE, Mar. 2015, pp. 235–236. ISBN: 978-1-4799-8977-5. DOI: 10.1109/MobileCloud.2015.33. URL: https://ieeexplore.ieee.org/document/7130893 (visited on 11/24/2020).

[39] F. Lordan and R. M. Badia. "COMPSs-Mobile: Parallel Programming for Mobile-Cloud Computing". In: *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. May 2016, pp. 497–500. DOI: 10.1109/CCGrid.2016.16.

[40] Francesc Lordan et al. "ServiceSs: An Interoperable Programming Framework for the Cloud". In: *Journal of Grid Computing* 12.1 (Mar. 2014), pp. 67–91. ISSN: 1572-9184. DOI: 10.1007/s10723-013-9272-5. URL: https://doi.org/10.1007/s10723-013-9272-5.

[41] Peter Mell and Tim Grance. *The NIST Definition of Cloud Computing*. en. Tech. rep. NIST Special Publication (SP) 800-145. National Institute of Standards and Technology, Sept. 2011. DOI: https://doi.org/10.6028/NIST.SP.800-145. URL: https://csrc.nist.gov/publications/detail/sp/800-145/final (visited on 11/24/2020).

[42] Minseok Jang, Myong-Soon Park, and S. C. Shah. "A mobile ad hoc cloud for automated video surveillance system". In: *2017 International Conference on Computing, Networking and Communications (ICNC)*. Jan. 2017, pp. 1001–1005. DOI: 10.1109/ICCNC.2017.7876271.

[43] *MVC vs. MVP vs. MVVM: Which Pattern to Choose for Android App Development? — Hacker Noon*. URL: https://hackernoon.com/mvc-vs-mvp-vs-mvvm-which-pattern-to-choose-for-android-application-development-8u493ay1 (visited on 11/24/2020).

[44] Nitin Naik and Paul Jenkins. "A Secure Mobile Cloud Identity: Criteria for Effective Identity and Access Management Standards". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Oxford, United Kingdom: IEEE, Mar. 2016, pp. 89–90. ISBN: 978-1-5090-1754-6. DOI: 10.1109/MobileCloud.2016.22. URL: https://ieeexplore.ieee.org/document/7474415 (visited on 11/24/2020).

[45] Wasana Ngaogate. "Integrating Flyweight Design Pattern and MVC in Development of Web Application". In: *Proceedings of the 2020 2nd International Conference on Information Technology and Computer Communications*. ITCC 2020. event-place: Kuala Lumpur, Malaysia. New York, NY, USA: Association for Computing Machinery, 2020, pp. 27–31. ISBN: 978-1-4503-7539-9. DOI: 10.1145/3417473.3417478. URL: https://doi.org/10.1145/3417473.3417478.

[46] Hai Duc Nguyen et al. "Real-Time Serverless: Enabling Application Performance Guarantees". In: *Proceedings of the 5th International Workshop on Serverless Computing*. WOSC '19. event-place: Davis, CA, USA. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–6. ISBN: 978-1-4503-7038-7. DOI: 10.1145/3366623.3368133. URL: https://doi.org/10.1145/3366623.3368133.

[47] Node.js. *Node.js*. en. URL: https://nodejs.org/en/ (visited on 11/24/2020).

[48] Robin Nunkesser. "Beyond Web/Native/Hybrid: A New Taxonomy for Mobile App Development". In: *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems*. MOBILESoft '18. event-place: Gothenburg, Sweden. New York, NY, USA: Association for Computing Machinery, 2018, pp. 214–218. ISBN: 978-1-4503-5712-8. DOI: 10.1145/3197231.3197260. URL: https://doi.org/10.1145/3197231.3197260.

[49] M. J. O'Sullivan and D. Grigoras. "Context Aware Mobile Cloud Services: A User Experience Oriented Middleware for Mobile Cloud Computing". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Mar. 2016, pp. 67–72. DOI: 10.1109/MobileCloud.2016.13.

[50] *React Native*. en. URL: https://reactnative.dev/ (visited on 11/24/2020).

[51] Zhongshan Ren et al. "Migrating Web Applications from Monolithic Structure to Microservices Architecture". In: *Proceedings of the Tenth Asia-Pacific Symposium on Internetware*. Internetware '18. event-place: Beijing, China. New York, NY, USA: Association for Computing Machinery, 2018. ISBN: 978-1-4503-6590-1. DOI: 10.1145/3275219.3275230. URL: https://doi.org/10.1145/3275219.3275230.

[52] Arnold Rosenbloom. "A Simple MVC Framework for Web Development Courses". In: *Proceedings of the 23rd Western Canadian Conference on Computing Education*. WCCCE '18. event-place: Victoria, BC, Canada. New York, NY, USA: Association for Computing Machinery, 2018. ISBN: 978-1-4503-5805-7. DOI: 10.1145/3209635.3209637. URL: https://doi.org/10.1145/3209635.3209637.

[53] Ruay-Shiung-Chang et al. "Mobile Cloud Computing Research - Issues, Challenges and Needs". In: *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. Mar. 2013, pp. 442–453. DOI: 10.1109/SOSE.2013.96.

[54] Denisson Santana dos Santos et al. "Recommendation System for Cross-Platform Mobile Development

[34] David Kopec. *Dart for absolute beginners*. The expert's voice in Web development. OCLC: ocn883364025. Berkeley, California: Apress, 2014. ISBN: 978-1-4302-6481-1 978-1-4302-6482-8.

[35] Dejan Kovachev, Yiwei Cao, and Ralf Klamma. "Mobile Cloud Computing: A Comparison of Application Models". In: *arXiv:1107.4940 [cs]* (July 2011). arXiv: 1107.4940. URL: http://arxiv.org/abs/1107.4940 (visited on 11/24/2020).

[36] *Language tour — Dart*. URL: https://dart.dev/guides/language/language-tour (visited on 11/25/2020).

[37] Lihui Lei et al. "MCloudDB: A Mobile Cloud Database Service Framework". In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. San Francisco, CA, USA: IEEE, Mar. 2015, pp. 6–15. ISBN: 978-1-4799-8977-5. DOI: 10.1109/MobileCloud.2015.30. URL: https://ieeexplore.ieee.org/document/7130864 (visited on 11/24/2020).

[38] Shuyu Li and Jerry Gao. "Moving from Mobile Databases to Mobile Cloud Data Services". In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. San Francisco, CA, USA: IEEE, Mar. 2015, pp. 235–236. ISBN: 978-1-4799-8977-5. DOI: 10.1109/MobileCloud.2015.33. URL: https://ieeexplore.ieee.org/document/7130893 (visited on 11/24/2020).

[39] F. Lordan and R. M. Badia. "COMPSs-Mobile: Parallel Programming for Mobile-Cloud Computing". In: *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. May 2016, pp. 497–500. DOI: 10.1109/CCGrid.2016.16.

[40] Francesc Lordan et al. "ServiceSs: An Interoperable Programming Framework for the Cloud". In: *Journal of Grid Computing* 12.1 (Mar. 2014), pp. 67–91. ISSN: 1572-9184. DOI: 10.1007/s10723-013-9272-5. URL: https://doi.org/10.1007/s10723-013-9272-5.

[41] Peter Mell and Tim Grance. *The NIST Definition of Cloud Computing*. en. Tech. rep. NIST Special Publication (SP) 800-145. National Institute of Standards and Technology, Sept. 2011. DOI: https://doi.org/10.6028/NIST.SP.800-145. URL: https://csrc.nist.gov/publications/detail/sp/800-145/final (visited on 11/24/2020).

[42] Minseok Jang, Myong-Soon Park, and S. C. Shah. "A mobile ad hoc cloud for automated video surveillance system". In: *2017 International Conference on Computing, Networking and Communications (ICNC)*. Jan. 2017, pp. 1001–1005. DOI: 10.1109/ICCNC.2017.7876271.

[43] *MVC vs. MVP vs. MVVM: Which Pattern to Choose for Android App Development? — Hacker Noon*. URL: https://hackernoon.com/mvc-vs-mvp-vs-mvvm-which-pattern-to-choose-for-android-application-development-8u493ay1 (visited on 11/24/2020).

[44] Nitin Naik and Paul Jenkins. "A Secure Mobile Cloud Identity: Criteria for Effective Identity and Access Management Standards". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Oxford, United Kingdom: IEEE, Mar. 2016, pp. 89–90. ISBN: 978-1-5090-1754-6. DOI: 10.1109/MobileCloud.2016.22. URL: https://ieeexplore.ieee.org/document/7474415 (visited on 11/24/2020).

[45] Wasana Ngaogate. "Integrating Flyweight Design Pattern and MVC in Development of Web Application". In: *Proceedings of the 2020 2nd International Conference on Information Technology and Computer Communications*. ITCC 2020. event-place: Kuala Lumpur, Malaysia. New York, NY, USA: Association for Computing Machinery, 2020, pp. 27–31. ISBN: 978-1-4503-7539-9. DOI: 10.1145/3417473.3417478. URL: https://doi.org/10.1145/3417473.3417478.

[46] Hai Duc Nguyen et al. "Real-Time Serverless: Enabling Application Performance Guarantees". In: *Proceedings of the 5th International Workshop on Serverless Computing*. WOSC '19. event-place: Davis, CA, USA. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–6. ISBN: 978-1-4503-7038-7. DOI: 10.1145/3366623.3368133. URL: https://doi.org/10.1145/3366623.3368133.

[47] Node.js. *Node.js*. en. URL: https://nodejs.org/en/ (visited on 11/24/2020).

[48] Robin Nunkesser. "Beyond Web/Native/Hybrid: A New Taxonomy for Mobile App Development". In: *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems*. MOBILESoft '18. event-place: Gothenburg, Sweden. New York, NY, USA: Association for Computing Machinery, 2018, pp. 214–218. ISBN: 978-1-4503-5712-8. DOI: 10.1145/3197231.3197260. URL: https://doi.org/10.1145/3197231.3197260.

[49] M. J. O'Sullivan and D. Grigoras. "Context Aware Mobile Cloud Services: A User Experience Oriented Middleware for Mobile Cloud Computing". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Mar. 2016, pp. 67–72. DOI: 10.1109/MobileCloud.2016.13.

[50] *React Native*. en. URL: https://reactnative.dev/ (visited on 11/24/2020).

[51] Zhongshan Ren et al. "Migrating Web Applications from Monolithic Structure to Microservices Architecture". In: *Proceedings of the Tenth Asia-Pacific Symposium on Internetware*. Internetware '18. event-place: Beijing, China. New York, NY, USA: Association for Computing Machinery, 2018. ISBN: 978-1-4503-6590-1. DOI: 10.1145/3275219.3275230. URL: https://doi.org/10.1145/3275219.3275230.

[52] Arnold Rosenbloom. "A Simple MVC Framework for Web Development Courses". In: *Proceedings of the 23rd Western Canadian Conference on Computing Education*. WCCCE '18. event-place: Victoria, BC, Canada. New York, NY, USA: Association for Computing Machinery, 2018. ISBN: 978-1-4503-5805-7. DOI: 10.1145/3209635.3209637. URL: https://doi.org/10.1145/3209635.3209637.

[53] Ruay-Shiung-Chang et al. "Mobile Cloud Computing Research - Issues, Challenges and Needs". In: *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. Mar. 2013, pp. 442–453. DOI: 10.1109/SOSE.2013.96.

[54] Denisson Santana dos Santos et al. "Recommendation System for Cross-Platform Mobile Development

Framework". In: *Proceedings of the XV Brazilian Symposium on Information Systems*. SBSI'19. event-place: Aracaju, Brazil. New York, NY, USA: Association for Computing Machinery, 2019. ISBN: 978-1-4503-7237-4. DOI: 10.1145/3330204.3330279. URL: https://doi.org/10.1145/3330204.3330279.

[55] Naresh Kumar Sehgal, Pramod Chandra P Bhatt, and John M Acken. *Cloud computing with security: concepts and practices*. English. OCLC: 1128833033. 2020. ISBN: 978-3-030-24612-9 978-3-030-24611-2.

[56] *Servicios de cloud computing*. es. URL: https://cloud.google.com/?hl=es (visited on 11/24/2020).

[57] A. Sharma. "Mission Swachhta : Mobile application based on Mobile Cloud Computing". In: *2020 10th International Conference on Cloud Computing, Data Science Engineering (Confluence)*. Jan. 2020, pp. 133–138. DOI: 10.1109/Confluence47617.2020.9057926.

[58] Katy Stalcup. *AWS vs Azure vs Google Cloud Market Share 2020: What the Latest Data Shows*. en-US. Nov. 2020. URL: www.parkmycloud.com (visited on 11/24/2020).

[59] N. Takahashi, H. Tanaka, and R. Kawamura. "Analysis of Process Assignment in Multi-tier mobile Cloud Computing and Application to Edge Accelerated Web Browsing". In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. Mar. 2015, pp. 233–234. DOI: 10.1109/MobileCloud.2015.23.

[60] L. A. Tawalbeh and W. Bakhader. "A Mobile Cloud System for Different Useful Applications". In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. Aug. 2016, pp. 295–298. DOI: 10.1109/W-FiCloud.2016.66.

[61] *The Dart type system*. URL: https://dart.dev/guides/language/type-system (visited on 11/25/2020).

[62] Lizhe Wang et al. *Cloud Computing*. English. OCLC: 1105773204. 2011. URL: https://www.safaribooksonline.com/complete/auth0oauth2/&state=/library/view//9781439856420/?ar (visited on 11/24/2020).

[63] *Xamarin — Open-source mobile app platform for .NET*. en. URL: https://dotnet.microsoft.com/apps/xamarin (visited on 11/24/2020).

[64] J. Xiong and H. Gu. "A Remote Engine Health Management System Based on Mobile Cloud Computing". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Mar. 2016, pp. 87–88. DOI: 10.1109/MobileCloud.2016.18.

[65] Z. Zhang and S. Li. "A Survey of Computational Offloading in Mobile Cloud Computing". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Mar. 2016, pp. 81–82. DOI: 10.1109/MobileCloud.2016.15.

[66] B. Zhou et al. "mCloud: A Context-Aware Offloading Framework for Heterogeneous Mobile Cloud". In: *IEEE Transactions on Services Computing* 10.5 (Sept. 2017), pp. 797–810. ISSN: 1939-1374. DOI: 10.1109/TSC.2015.2511002.

Framework". In: *Proceedings of the XV Brazilian Symposium on Information Systems*. SBSI'19. event-place: Aracaju, Brazil. New York, NY, USA: Association for Computing Machinery, 2019. ISBN: 978-1-4503-7237-4. DOI: 10.1145/3330204.3330279. URL: https://doi.org/10.1145/3330204.3330279.

[55] Naresh Kumar Sehgal, Pramod Chandra P Bhatt, and John M Acken. *Cloud computing with security: concepts and practices*. English. OCLC: 1128833033. 2020. ISBN: 978-3-030-24612-9 978-3-030-24611-2.

[56] *Servicios de cloud computing*. es. URL: https://cloud.google.com/?hl=es (visited on 11/24/2020).

[57] A. Sharma. "Mission Swachhta : Mobile application based on Mobile Cloud Computing". In: *2020 10th International Conference on Cloud Computing, Data Science Engineering (Confluence)*. Jan. 2020, pp. 133–138. DOI: 10.1109/Confluence47617.2020.9057926.

[58] Katy Stalcup. *AWS vs Azure vs Google Cloud Market Share 2020: What the Latest Data Shows*. en-US. Nov. 2020. URL: www.parkmycloud.com (visited on 11/24/2020).

[59] N. Takahashi, H. Tanaka, and R. Kawamura. "Analysis of Process Assignment in Multi-tier mobile Cloud Computing and Application to Edge Accelerated Web Browsing". In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. Mar. 2015, pp. 233–234. DOI: 10.1109/MobileCloud.2015.23.

[60] L. A. Tawalbeh and W. Bakhader. "A Mobile Cloud System for Different Useful Applications". In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. Aug. 2016, pp. 295–298. DOI: 10.1109/W-FiCloud.2016.66.

[61] *The Dart type system*. URL: https://dart.dev/guides/language/type-system (visited on 11/25/2020).

[62] Lizhe Wang et al. *Cloud Computing*. English. OCLC: 1105773204. 2011. URL: https://www.safaribooksonline.com/complete/auth0oauth2/&state=/library/view//9781439856420/?ar (visited on 11/24/2020).

[63] *Xamarin — Open-source mobile app platform for .NET*. en. URL: https://dotnet.microsoft.com/apps/xamarin (visited on 11/24/2020).

[64] J. Xiong and H. Gu. "A Remote Engine Health Management System Based on Mobile Cloud Computing". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Mar. 2016, pp. 87–88. DOI: 10.1109/MobileCloud.2016.18.

[65] Z. Zhang and S. Li. "A Survey of Computational Offloading in Mobile Cloud Computing". In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. Mar. 2016, pp. 81–82. DOI: 10.1109/MobileCloud.2016.15.

[66] B. Zhou et al. "mCloud: A Context-Aware Offloading Framework for Heterogeneous Mobile Cloud". In: *IEEE Transactions on Services Computing* 10.5 (Sept. 2017), pp. 797–810. ISSN: 1939-1374. DOI: 10.1109/TSC.2015.2511002.