Development of mobile IoT solutions: approaches, architectures, and methodologies

Article (Accepted Version)

Development of mobile IoT solutions: approaches, architectures, and methodologies

Naercio Magaia, Pedro Gomes, Lion Silva, Breno Sousa,
Constandinos X. Mavromoustakis, *Senior Member, IEEE*, and George Mastorakis

***Abstract*—Modern Living, as we know it, has been impacted meaningfully by the Internet of Things (IoT). IoT consists of a network of things that collect data from machines (e.g., mobile devices) and people. Mobile application development is a flourishing tendency, given the increasing popularity of smartphones. Nowadays, users are accessing their desired services on the smartphone by means of dedicated applications as the latter offers a more customized and prompt service. In addition, companies are also looking to persuade users by offering interactive and effective mobile applications.**

Mobile application developers are using IoT to develop better applications. However, there is no generalized consensus on the selection of best architecture or even the most suitable communications protocols to be used on an IoT application development. Therefore, this article aims at presenting approaches, architectures, and methodologies relevant to the development of mobile IoT solutions.

***Index Terms*—Mobile development, IoT, Methodology, Architecture.**

I. Introduction

THE i-mode's launch in 1999 by NTT Docomo gained nationwide popularity in the Japanese mobile phone market due to the fact of it being the first integrated online application (app) store for mobile phones. Notably, the revenue-sharing business model used by Docomo allowed content creators and app providers to keep up to 91% of revenue generated by i-mode subscribers, which also facilitated rapid i-mode ecosystem growth [1]. Almost a decade later, Apple's iPhone and App Store were released, and the marketplace introduced third-party app development and distribution to the platform, which allowed consumers buying and transferring new apps. The latter release also motivated the popularity growth of mobile apps. After that, companies have been attempting to reach out to as much consumers as possible via these apps, which nowadays are a popular aspect in many business domains. Some examples include entertainment and safety-critical apps, which contribute considerably to the profitable mobile device market that is highly competitive, often releasing apps at a high pace.

The Internet of Things (IoT) [2] and its development is progressively affecting our lives, being the latter fostered by the exponential growth in the number of things (i.e., objects) connected to the Internet. An IoT is a gigantic network of interconnected things such as mobile phones, coffee makers, washing machines, wearable devices, and so on. IoT applications in the industry include predictive maintenance, smart metering, asset tracking, self-driving vehicles, etc. It is also becoming omnipresent at every level of the society, for example, on smart roads, by pushing messages to the driver, in smart homes, by controlling appliances through an application running on the smartphone, in the field of smart health, e.g., hospitals, care centers, by wearing accessories or part of clothing, etc.

From a software engineering[1] point of view, IoT applications, on the one hand, are distributed over a broad range of processing nodes, and, on the other hand, have high heterogeneity of the nodes and the protocols used between them. The combination of the latter characteristics presents some challenges, specifically: (i) IoT software should scale both in the number of nodes as well as in the number of types and platforms, conversely to other research areas where homogeneity keeps software complexity manageable; (ii) IoT system developers should leverage the diverse resources and decentralized computing power provided by IoT devices, on the contrary to existing tools and techniques that hide it hence enabling developers to write apps that execute similarly on different platforms; (iii) IoT applications should not be isolated or proprietary, but rather combine generic, application-specific and legacy things.

The development of IoT systems encompasses a set of key abstractions, which could be deduced from the analysis of general features [3]. First, it is necessary to identify the actors, i.e., the system stakeholders and users. The latter will own, manage, or even use the system and its functionalities, therefore, the requirements should be elicited to them. Second, a more comprehensive approach to requirements gathering (i.e., functional and nonfunctional ones) should be performed conversely to a simple understanding of the services that things and groups of things should provide. Third, the things involved in the requirements' implementation can correspond

N. Magaia, P. Gomes, L. Silva, and B. Sousa are with Faculty of Sciences, University of Lisbon, Campo Grande, 1749-016 Lisbon, Portugal. (e-mail: ndmagaia@ciencias.ulisboa.pt).

N. Magaia and L. Silva are also with LASIGE, Department of Computer Science.

C. Mavromoustakis is with the Department of Computer Science, University of Nicosia, 46 Makedonitissas Avenue, CY-2417, Nicosia, Cyprus (email: mavromoustakis.c@unic.ac.cy).

G. Mastorakis is with the Department of Management Science and Technology, Hellenic Mediterranean University, Agios Nikolaos, 72100, Crete, Greece (email: gmastorakis@hmu.gr).

[1]Software engineering can be defined as a process by which someone (i.e., an individual or a team) organizes and manages the creation of complex computer programs.

to a variety of objects and devices, depending on a multitude of technologies and capabilities.

Certain software engineering processes can be leveraged by companies to improve application development processes hence gaining a competitive benefit [4]. Mobile application development can be considered as similar to software engineering given the existence of common issues such as integration with hardware, traditional security issues, performance, reliability, and storage limitations. Nevertheless, there are some less commonly found requirements in traditional software applications if compared to mobile ones, which include potential interaction with other application, sensor handling, the existence of native and hybrid (i.e., mobile web) applications, families of software and hardware platforms, security, user interfaces, the complexity of testing and power consumption [5].

Given the generalized lack of consensus in IoT applications development, this article aims at presenting approaches, architectures, and methodologies relevant to the development of mobile IoT solutions.

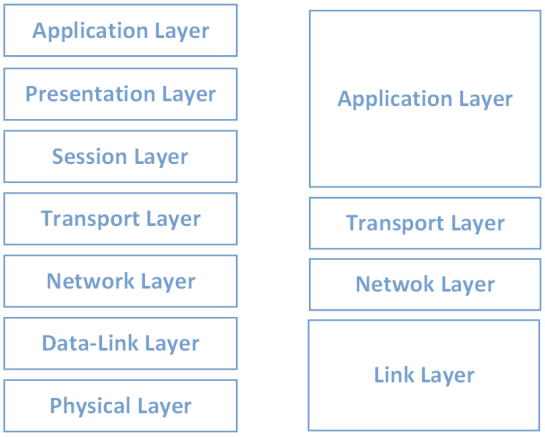
The remainder of this article is as follows. Section II and III present IoT and AR concepts, respectively. Section IV presents mobile development approaches and tools. Section V presents the architecture patterns used in mobile development. In Section VI, communication technologies between mobile applications are presented. Section VII presents the concepts of relational and non-relational databases. In Section VIII, software development methodologies are presented. Section IX presents an overview of a mobile solution integrating IoT and AR. Sections IX and IX present challenges and open issues, and future directions, respectively. Finally, Section XII presents concluding remarks.

II. Internet of Things

IoT is increasingly impacting our lives as all gadgets or devices are already networked to communicate with each other. There are sectors such as healthcare, security, surveillance, and product management using IoT-based technologies, allowing for a wealth of new services to meet users' needs.

According to [6], an IoT system should contain the following eight characteristics:

1) *Heterogeneity.* Due to the existence of several devices and protocols.
2) *Scalability.* Due to the massive number of services that should be available.
3) *Ubiquitous data exchange.* It allows the exchange of information through wireless technologies. This is an important point in the availability of a service, as it can cause problems when there is no communication between devices and services.
4) *Optimized energy solutions.* Because there is a huge variety of devices and communications between them, minimizing communications-related energy consumption is a primary concern.
5) *Localization and tracking capability.* IoT entities can be identified and provide short-range wireless communications through device location.



(a) The OSI model (b) The TCP/IP model

Fig. 1. The OSI and TCP/IP layer models

6) *Self-organizing capability.* Nodes in an IoT network have the ability to organize autonomously into ad hoc networks to provide the means to perform tasks and share data without user intervention.
7) *Semantic interoperability and data management.* Due to massive data exchange and device diversity, there is a need to ensure interoperability between different applications by providing appropriate formats, languages, and standards.
8) *Security and privacy.* An IoT system must ensure security and preserve data privacy.

In the following sections, a set of most used architectural styles for engineering IoT applications, that is, driving the way IoT components shall be combined together, are provided. Besides, sensors and actuators responsible for detecting and affecting changes in the environment/equipment used in IoT, and some of the most widely adopted IoT networking protocols are also presented.

A. Architectural styles

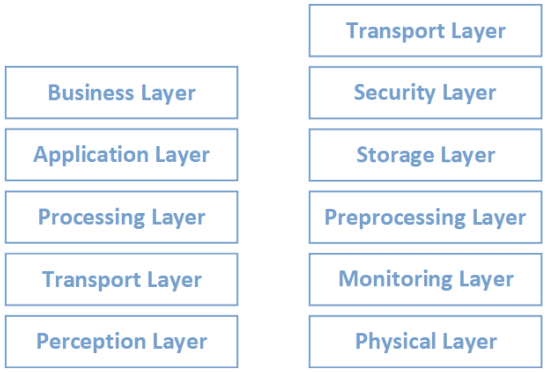
Many styles have been proposed over the years, and the following subsections list them based on their relevance.

1) Layered architecture: In 1974, Vint Cerf et al. [7] described the TCP/IP model, i.e., a set of communications protocols used in computer networks. Almost a decade later, the Open System Interconnection (OSI) model, which is a more comprehensive reference architecture for general networking systems, was formally adopted by ISO as an international standard. Figure 1 presents the OSI and TCP/IP models.

As IoT applications grew in software deployments, there was a need to create industry reference architectures. The main objective was to facilitate interoperability and simplify development. However, there is no consensus on an IoT architecture that is accepted by all.

Similar to the ISO reference architecture, IoT architectures can be classified based on layers, systems, and interactions [8].

a) Three to five layers: The three-tier architecture came first and was considered as the simplest as it represents the basic idea of IoT being composed by perception, network,



(a) Five layers' architecture

(b) Fog architecture

Fig. 2. IoT Architectural styles

and application layers. The first refers to the physical layer, where sensors are located to detect and gather information. The second layer is responsible for connecting network devices and their servers and can be used to transmit data. Finally, the application layer provides user-specific services.

Because these layers are not sufficient in many IoT models, it was necessary to create an architecture this time with five layers maintaining the perception and application layers, and adding the transport, processing, and business layers, as can be seen in Figure 2(a). The transport layer is responsible for transmitting the data received from the perception layer to the processing layer. Various forms of transmission are used, such as wireless or Bluetooth, using protocols such as IPv6 [9]. The processing layer primarily stores, analyzes, and processes information from the transport layer. It can be designated as middleware because it can manage and deliver services to lower layers. Technologies such as cloud computing, database, or large data processing are used in this layer.

2) Cloud and Fog based architecture: The cloud computing model [10] is a very efficient alternative to the data center for customers using web applications or large amounts of data. In some system architectures, data processing is done through computers in the cloud to achieve flexibility and scalability. Cloud computing is typically divided into three levels, namely Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS). In this system, the cloud stores and analyzes all the data it receives. However, to avoid the latter, fog computing was designed.

Fog computing [11] is a virtualized platform that provides computing services, storage between end devices using cloud computing. This new model presents a layered approach where sensors and gateways that are part of the system containing the ability to process and analyze data. As can be seen in Figure 2(b), this architecture contains six layers designated transport, security, storage, preprocessing, monitoring, and physical. The monitoring layer controls resources, energy, and service responses. The preprocessing layer acts as a filter for data at the ends of a system.

3) SoA and Microservices: Service-Oriented Architecture (SoA) is a standard that is applied at the architectural level in the construction of enterprise software. It aims to separate and specialize the components of the architecture so that they are each owner of and responsible for a single service (function). Besides that, these services have to be connected to each other in the simplest way possible (with almost no knowledge of each other's internal implementation) through a common protocol, such as a RESTful API. As the service modules are like black boxes, the end result is transparency [12].

An SoA architecture is not made of a fixed set of organized layers, but of practices that provide the evolution and scalability of applications, and these practices are mostly to separate each service in different nodes and let them communicate by message exchange [13].

The goals of SoA are to have: standard communication protocol, loose coupling, cohesion, composability, stateless, each service to be autonomous, to provide meta-data about itself, to do abstraction in such a way that the logic and technology of the service are invisible outside of it.

Similarly, but at a more specific level, microservices are at the level of the application, and not of the system architecture from scratch. Each microservice has its own data and does not necessarily need to share a common communication protocol with other microservices. They are standalone applications, which can scale better on their own, not necessarily needing other applications in the business scope to do so as well. It is debatable if there is a fundamental difference in both. However, it is more appropriate to consider that microservices is a case of SOA [14] since the differences are in the realization of each and not precisely in the conceptual architectural structure.

SoA and microservices are widely used. For instance, the former is used due to the very nature of IoT, that is, a heterogeneous system, which needs to interoperate with different types of data, protocols, sensors, energy requirements, among others [15]. IoT devices can be seen as services. Each device is self-contained, with its sensors and processors, and are specialized in a particular task (e.g., measuring temperature, filming, saving data), which meets both SoA and microservices aspects [16].

In SoA, and particularly in microservices, it is desirable that the various services know something about each other, which is called the "health" of the service, so that eventually, one service does not call another that does not work if it is possible to have a substitute, hence preventing fault cascading. The latter helps both to reduce the traffic itself from information to non-functional nodes and adequately replace, if possible, part of the system with calls to other parties [17].

4) Social IoT: Links between various devices should be considered in the same way that humans form relationships. There are three principles of a Social Internet of Things (SIoT) system. The first principle is based on the ability to browse the network to discover devices and services effectively and ensure scalability. The second one emphasizes the degree of reliability to take advantage of interactions between "friend" devices. In the last one, models designed to study social networks can be reused to address IoT-related issues, which is intrinsically related to large networks of interconnected objects [18].

Given these principles, an architecture based on social relations has been proposed. In it, the server-side architecture has three layers: the first layer contains a database that stores

device details such as their attributes, meta information, and their relationships. The second layer contains the code to interact with devices, query their status, and use a subset of them to provide a service. The third layer (i.e., the application layer) provides services to users. On the device side, there are two layers. The first one allows a device to connect to another device and exchange information. The other layer, which is called the social layer, allows executing requests and interacting with the server application layer.

B. Sensors and Actuators

Many industrial activities contain sensors responsible for detecting changes in the environment/equipment, which are usually used in factories, customer houses, assembly lines or even remote locations. For example, in areas such as servicing or repairing equipment, the ability to use sensors on multiple machines allows to realize or even anticipate problems. Many of the sensors used in IoT are typically small, inexpensive, and energy-efficient. First, let us consider sensors used daily that are built into our mobile phone. A smartphone is a useful and user-friendly device that has a number of integrated communication and data processing features. Typically, such devices have sensors such as an accelerometer, which allows the user to measure speed in three dimensions, a gyroscope that senses a user's orientation, camera, magnetometer that detects magnetic fields, GPS, light sensor, and the latest already have humidity sensors. However, depending on the business area, the purpose of a sensor is different. For example, a pressure sensor (that is, a barometer and pressure gauge) can be used in an industrial environment to detect and measure the pressure of a gas, or an optical sensor to obtain fiber optics information.

When talking about sensors, possible actuators should be referenced. An actuator is a device that can affect a change in an environment by converting electrical energy into some kind of useful energy. Some examples of actuators are cooling equipment, speakers, lights, and monitors. An actuator can also alert a user to a problem through an alarm or notification.

C. Networking protocols

Figure 3 shows some of the most widely adopted IoT networking protocols mapped to the TCP/IP model.

1) Link-layer technologies: IoT link-layer technologies cellular, Wi-Fi, and Ethernet, along with more specific solutions such as Low Power Wide Area Network (LPWAN), Bluetooth Low Energy (BLE), ZigBee, NFC, and RFID [19]. Next, some technologies are briefly described:

a) LPWAN: LPWAN is a technology intended for low-power, long-range wireless communication, being best for large-scale deployments of low-power IoT devices. Its technologies include Long Range physical layer protocol (LoRa), Haystack, SigFox, Long Term Evolution for Machines (LTE-M), and Narrow-Band IoT (NB-IoT).

b) Cellular: NB-IoT and LTE-M standards focus on low-power, low-cost IoT communication in existing mobile networks. They were built exclusively for IoT, although current mobile technologies are also often adopted for long-range wireless communication.

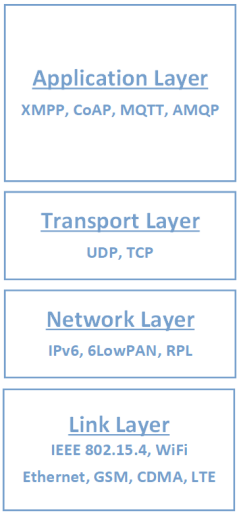


Fig. 3. IoT networking protocols considering the TCP/IP model

c) BLE: BLE is a low-power version of the popular Bluetooth, being designed for short-range communication with a single primary device controlling several secondary devices. It is often used by personal IoT devices.

d) ZigBee: ZigBee has a longer range and a slightly lower data rate than BLE. It was designed for building and home automation applications.

e) Wi-Fi: Wi-Fi is a family of wireless networking technologies based on the IEEE 802.11 family of standards. Nonetheless, IoT devices might only use 802.11b/g due to power conservation reasons.

2) Internet layer technologies: IoT Internet layer technologies include IPv6, IPv6 Low Power Wireless Personal Area Network (6LoWPAN), and IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [19]. Specially,

a) IPv6: Nowadays, IoT applications usually use IPv6 over legacy IPv4, given the current number of connected IoT devices. However, many IoT devices do not require public addresses, therefore being deployed in private networks and only communicating via gateways to external devices or services.

b) 6LoWPAN: 6LoWPAN allows IPv6 to be used over 802.15.4 wireless networks. It is frequently used for wireless sensor networks and in-home automation applications.

c) RPL: RPL is designed for routing packets over low-power and constrained networks such as those implemented over 6LoWPAN (e.g., wireless sensor networks), and can compute the optimal path through network nodes' graph-based on dynamic metrics and constraints.

3) Application layer technologies: The application layer acts as an abstraction that encompasses protocols that allow communications between various applications. Traditional protocols used at the application layer were not an option for IoT services due to the restrictions of low power and lossy networks (LLNs) [20]. The following protocols are used to update servers with current values of end devices and carry application instructions to those devices [21].

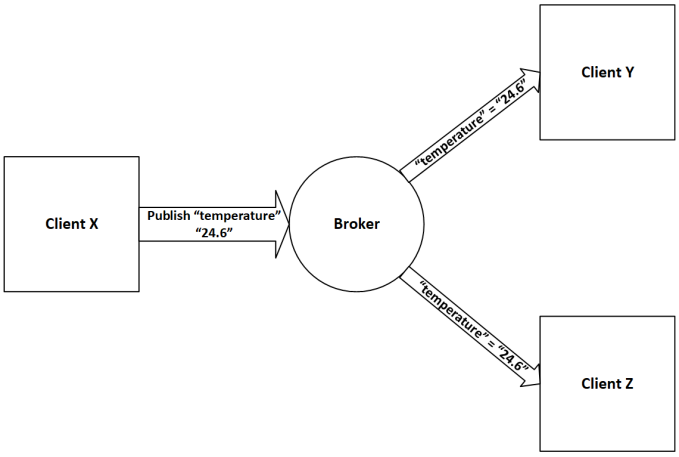


Fig. 4. The publisher/subscriber model

a) Constrained Application Protocol (CoAP): CoAP is a request/response protocol designed for Machine-to-Machine (M2M) applications to ensure interoperability using the Hypertext Transfer Protocol (HTTP) often used in combination with 6LoWPAN over the User Datagram Protocol (UDP) [22] protocol. It uses the Datagram Transport Layer Security (DTLS) [23] for secure communications.

Although it was designed for the IoT and M2M communications, no built-in security features are included. DTLS running on top of UDP is used to secure CoAP transactions. CoAP is HTTP-compatible by design, although CoAP over DTLS's diverse packet structure might confuse HTTP servers [21].

b) Message Queue Telemetry Transport (MQTT): MQTT is a publisher/subscriber-based messaging protocol designed for M2M communications, that is, in low bandwidth situations for IoT devices (e.g., sensors) on unreliable networks. Each device can be a publisher that sends topic-based messages to a broker or event bus, and a subscriber that receives this automatic message whenever there is an update to a subscribed topic. As an example, consider Figure 4, where client X posted in the broker a temperature topic, and clients Y and Z subscribed to this topic.

Even though running over TCP, MQTT have by design low overhead if compared to other TCP applications, therefore making it a suitable protocol for IoT compared to the CoAP request/response method. Since CoAP is UDP based it has less overhead; however, the probability of a packet loss is higher compared to TCP.

For example, MQTT's publish/subscribe architecture is more suitable for the IoT than CoAP's request/response as messages do need to be responded to. That is, the MQTT approach uses less network bandwidth and reduces message processing, which also extends devices' battery lifetime [24].

c) The Extensible Messaging and Presence Protocol (XMPP): XMPP has been in use for over a decade and has been used all over the internet, and by being an old protocol, it may be insufficient to provide services needed for new applications. However, it has been gaining much attention as a protocol to be used for IoT. XMPP uses the TCP protocol supporting the publisher/subscriber architecture as well as request/response. It allows real-time communications when it comes to small message volume and guaranteed security through TLS/SSL. However, this protocol does not provide QoS options that can make it impractical for M2M communications [21]. Similar to MQTT, XMPP supports the publish/subscribe architecture hence being more appropriate for IoT if compared to CoAP's request/response approach. XMPP is primarily used with smart appliances.

d) Restful Services: The Representational State Transfer (REST) [21] is not a protocol but an architectural style that uses some HTTP methods to provide a request/response messaging system. REST is important in IoT as it is supported by all M2M cloud platforms (i.e., Amazon, Google, etc.) and can be easily deployed on mobile devices. Nevertheless, most M2M platforms do not support Hypertext Transfer Protocol Secure (HTTPS). Instead, they provide authentication keys. Another limitation of this architecture is related to the fact of being difficult to implement [21].

Given the recent popularity of mobile apps, the extra overhead caused by request/response protocols affects devices' batteries due to its continuous or long polling for values, mainly if there are no new updates. Publish/subscribe approaches such as MQTT or XMPP can avoid the latter issue. In contrast, CoAP, which is the lightweight version of REST, has the same disadvantages. Nevertheless, and conversely to REST, it can be used by constrained resource devices as it runs over UDP.

e) Advanced Message Queuing Protocol (AMQP): AMQP has emerged in the financial industry since it is a standard used for messaging between applications and organizations. It can link systems, business processes to convey the requested instructions reliably [25]. This protocol can use several transport protocols, nonetheless, preferably uses TCP to ensure reliability. It uses publisher/subscriber communication messages. When there are network disruptions, it ensures reliability when storing and forwarding messages. The limitation of this protocol is related to the low success rate when the bandwidth is low [21].

According to [26], [27], AMQP's success rate increases as the bandwidth increases, and it can send more messages per second if compared with REST. In addition, an AMQP environment spread across many continents and with thousand users can process millions of messages per day [27]. Moreover, JPMorgan uses AMQP to send 1 billion messages per day, which are used in its mission-critical systems around the world [28].

f) Websocket: The Websocket protocol [21] was created to reduce Internet communication overhead by providing full-duplex communication (i.e., real-time simultaneous data transmission). Websocket is not a request/response or publisher/subscriber-based protocol and it runs over TCP, allowing to use TLS/SSL over communications. However, for the IoT aspect, there is a sub-protocol named Websocket Application Messaging Protocol (WAMP) [29], which allows the publisher/subscriber standard to be adapted. This protocol has the same limitation as XMPP and REST, as both do not provide QoS.

Differently from previous protocols, Websocket is neither designed for resource-constrained devices nor suitable for IoT

applications because of its client/server-based architecture. Nonetheless, it is secure and minimizes overhead, and can provide efficient messaging systems if used along with WAMP [21].

g) A comparison in light of an IoT application: In [30], four application layer protocols, namely, CoAP, MQTT, XMPP, and WebSocket, were compared in light of a smart parking application using their open-source implementations and aiming to measure their response time with different traffic load.

MQTT and CoAP have a much higher response time if compared to XMPP as they use a message queue to process client requests. CoAP performs better at lower traffic load as it benefits from using UDP. Conversely, MQTT performs better since it uses some extra optimization features of the protocol. XMPP performs better at low traffic load by transferring messages directly over a TCP connection meanwhile, WebSocket uses an additional WebSocket handshake. However, as load increases, the latter takes advantage of multiplexing to reduce the mean response time.

D. Middleware

Ubiquitous computing is the core of the IoT model, thus allowing all devices to be linked together. To ensure this, there must be well-defined interoperability and standards in these heterogeneous devices. The solution to abstracting these standards and formats is to have a middleware platform that acts as a software bridge between devices and the application.

According to [8], there are several middleware approaches, and they can be classified according to their design:

a) Event-based: It is an event-based architecture, consisting of two components: clients and brokers. Clients can be event publishers or subscribers and use services provided by the middleware to communicate. Brokers represent middleware and provide a distributed implementation [31].

b) Service-oriented: It is based on the Service-Oriented Architectures (SOA), containing independent modules that provide services through the available interfaces. Certain features are abstracted by a set of services used by the application. There is a repository where services are placed, and clients can access these services.

c) Database-oriented: In this approach, the IoT device network is considered as a database system. Through an interface, applications can query the database using a query language.

d) Semantics: It is based on interoperability between different devices. Devices incorporate different data formats and ontologies and are linked to a framework, allowing the exchange of this data. For each device, an adapter is required to map N patterns to an abstract pattern. This technique allows multiple physical resources to communicate with each other, even if devices do not implement the same protocols.

e) Application-specific: This type of middleware is specifically used in the application domain for which it is developed, hence allowing adjusting the architecture based on the requirements.

Table I lists IoT applications by type, domain, and usage. According to type, they can be either Infrastructure or Smart Things. The former, if IoT is used to maintain, scale, or empower its underlying network, or the latter, if it goes to the broad area of providing some intelligent, valuable computation and structured data gathering from diverse resources. Application domain refers to the vast identified areas of applicability; meanwhile, application usage specifies its purpose.

III. Augmented reality

Augmented reality (AR) offers a perfect interface to IoT applications by adding an additional layer with virtual information about smart objects and services with the user's view of the real world. Therefore, it allows the latter to interact with the real-world object in addition to getting extra contextual information about it, apart from information regarding adjacent objects [139].

The following fundamental requirements define AR, namely (i) combination of both contents (i.e., real and virtual), (ii) instantaneous interaction, and (iii) precise alignment of both real and virtual objects [140]. The latter requirements also describe an AR system's technical requirements. This system can merge both objects (i.e., real and virtual ones), permitting interactions with the consumer, being also possible to locate users and virtual objects' position via the tracking system.

Considering the enormous potential of AR and IoT, numerous challenges exist to integrate them. However, (i) comprehending how more instinctive this connection can be made remains a primary challenge for developing IoT or AR apps. Besides, debug and repair faults by making use of real-time IoT data, thus offering a better quality of service (QoS), remains a challenge for entities using AR.

It is also important to comprehend the differences between AR, virtual reality (VR), and mixed reality (MR), as they all have the ability to alter our perception of the real world. VR is a computer-created experience that allows a user to interact with a virtual environment. AR is the overlap of digital content in the real world. It is the most affordable technology because people can use their smartphones or tablets to run AR applications [141]. Lastly, MR concerns the fact that the virtual object is superimposed on the real environment allowing interaction with that same object.

A. AR Software Development Kits

With the increase of hardware performance on mobile devices, numerous AR Software Development Kits (SDKs) have emerged. AR SDKs allow introducing features such as recognition, tracking, and rendering, within an application [142]. The recognition component works like the AR "brain'", the tracking component as the "eyes", and the rendering component is concerned with imaginary objects that can be viewed through the mobile device.

Please note that the selection of the best AR SDK is made according to the functionality of the mobile app envisioned. Nonetheless, there are several major criteria that could be taken into account such as:

- type of license, as there are free and commercial licenses. Free functionality is typically limited, and to build a

TABLE I
CURRENT IOT APPLICATIONS BY TYPE, DOMAIN, AND USAGE

Type	Domain	Usage	Publication(s)
Infrastructure	Networking	5G Internet of Things, Networking Provision, Middleware	[32], [33], [34], [35], [36], [37], [38], [39]
	Computing Paradigms	Fog, Cloud, Edge	[40], [41], [42], [43], [44], [45], [46], [47], [48]
	Sensing and Sensor	Sensing, Sensor	[49], [50], [51], [52], [53], [54]
	Computation and Storage	Big Data, Data fusion	[55], [56], [57], [58]
Infrastructure/Smart Things	AI	Deep Learning, Machine Learning	[59], [60], [61], [62], [63], [64], [65], [66], [67], [68]
Smart Things	Smart Cities	Smart Parking, Traffic Congestion, Smart Lightning, Waste Management, Smart Roads	[69], [70], [71], [72], [73], [74], [75], [76]
	Smart Environment	Forest Fire Detection,	[77], [78], [79], [80], [81]
	Smart Water	Potable Water Monitoring, Pollution levels in the sea, Chemical Leakage Detection	[82], [83], [84], [85], [86]
	Smart Metering	Smart Grids, Silos Stock Calculation	[87], [88], [89], [90], [91], [92]
	Security and Emergencies	Perimeter Access Control, Surveillance and Monitoring	[93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103]
	Retail	Intelligent Shopping Applications, Smart Product Management	[104], [105], [106], [107], [108], [109], [110], [111]
	Logistics	Item Location, Fleet Tracking, Food Supply, Supply Chain Control	[112], [113], [114], [115], [116]
	Industry	M2M auto Diagnosis and Control, M2M Messaging, Indoor Air Quality, Temperature Monitoring, Indoor Location	[117], [118], [119], [120], [121], [122], [123], [124]
	Smart Agriculture	Green House Smart Monitoring, Smart irrigation, Meteorological Station Network, Fire Detection	[125], [126], [127], [128], [129],
	Smart Farming	Animal Tracking, Food Supply	[130], [131], [132]
	Smart Home	Energy and Water Use, Remote Controlling, Home Integration	[133], [134]
	eHealth	Patients Monitoring, Fall Detection, Viruses	[135], [136], [137], [138]

function-rich mobile app, commercial ones may be necessary;
- supported platforms, it is always advisable to check which platforms the mobile app works with. Indeed, most SDKs support both Android and iOS;
- smart glasses support, it would be advantageous to build mobile apps compatible with smart glasses given their recent popularity. They offer a way to overlay AR objects on the real-world also allowing hands-free AR experiences conversely to smartphones;
- Unity support, as it one of the most advanced game engines actually, which is used to create games for computers and consoles as well as being capable of powering AR mobile apps;
- cloud recognition, which enables creating mobile apps capable of recognizing lots of different markers by storing them in the cloud instead of requiring much space on a mobile device;
- 3D tracking, which allows creating mobile games and AR apps for e-commerce by recognizing 3D objects, such as cups, cylinders, boxes, toys.
- geolocation, which is essential for creating location-based AR mobile apps, and
- Simultaneous Localization and Mapping (SLAM), which allows applications to map an environment and track their own movements in it.

Below, a reference will be made to the majority of AR SDKs.

1) ArCore: ArCore [143] is an open-source framework from Google for building AR applications. It uses three features to integrate virtual content into the real world: motion tracking, environmental understanding, and light estimation. Motion tracking uses the camera and sensors such as the accelerometer and gyroscope to find the user's position and relative orientation, making it possible to move to different places where the object will be in its initial position. Environmental understanding allows the mobile device to detect the size and location of all types of surfaces: horizontal, vertical, and angular. Lastly, light estimation allows the mobile device to calculate ambient light conditions. Another feature to consider for ArCore is Cloud Anchor [14] since it allows multiple users to see and interact with the same virtual object in the same real environment. The main limitation of ArCore is the need for mobile devices running Android OS higher than 7.0 and iOS 11.0.

2) ArKit: ArKit [144] is a framework that combines real-world images, virtual images, and sensors. The first version of ArKit allowed the detection of 2D images. However, with ArKit 2 it was possible to extend this functionality to allow tracking of 2D images and add the ability to detect common 3D objects such as sculptures, toys, or furniture. Arkit can automatically detect flat surfaces through characteristic points and correlate those points with a vector. Both ArCore and ArKit have the ability to add an anchor allowing virtual objects to remain where they were initially placed, regardless of the position of the advisor. However, ArKit is only available for iOS.

3) Vuforia: Vuforia [145] is considered as one of the most widely used SDK for developing AR applications. It allows creating and managing one's own markers (i.e., VuMark) and has the ability to track flat images, use geolocation and multiple targets simultaneously. It contains stable vision-based

image recognition techniques, allowing taking advantage of the capabilities of mobile applications. Vuforia allows tracking of five objects simultaneously, regardless of low light conditions. Another feature is to let the application keep a consistent reference even when targets are not visible.

4) ArToolkit: ArToolKit [145] is considered to be one of the most widely used libraries for developing AR-enabled applications, as it is a constantly updated open-source tool that allows being exported to other platforms. Initially, this library was designed to be used on computers rather than on mobile devices because it was very challenging to implement on the latter. Nevertheless, over time, it was necessary to adapt this library for mobile devices. Its features include tracking capability of any standard that has a square marker, tracking fast enough to use real-time AR application, and supporting multiple formats such as RGB, YUV, etc.

5) Wikitude: Wikitude [142] is an SDK that includes image recognition, tracking, or rendering of 3D models and provides location-based AR. Wikitude uses a hybrid approach because it uses web technologies that allow developers to have augmented reality experiences on various platforms [146]. These experiments use an ARchictect interface to create AR objects, so they can integrate the SDK into a mobile application through a specific component called ArchitectView. Wikitude allows loading 3D objects; however, none of the common 3D file formats are supported as it only supports Wikitude 3D Format (i.e., .wt3 format). To facilitate the conversion of conventional .wt3 formats, it has developed a desktop application (also known as Wikitude 3D encoder) allowing to import formats such as *.fbx to *.wt3 [144]. The newly implemented features include 3D model tracking and Simultaneous Localization and Mapping (SLAM) functionality.

6) EasyAr: EasyAr [147] is a framework that offers a free version with the ability to recognize and track QR code and flat images. However, if there is a need to use SLAM or 3D object tracking functionality, one will need to purchase the paid version, which can be seen as a limitation of this framework.

IV. Mobile application development

Mobile application development is a complex process that requires knowledge of different approaches. Certain factors such as integration with the device hardware, security, performance, reliability, and storage limitations should be taken into account when developing mobile IoT applications [5]. Each factor can be decisive for the proper functioning and success of an application because each has its own advantages.

A. Mobile development approaches

Here, two development approaches are referred, namely native and cross-platform.

1) Native Development: Native applications are built using SDKs and platform-specific development tools, such as Android Studio [148] for Google's Android and Xcode [149] for Apple's iOS. The advantages of native mobile applications are that they have full access to the device features, APIs available on each platform, and the best possible performance

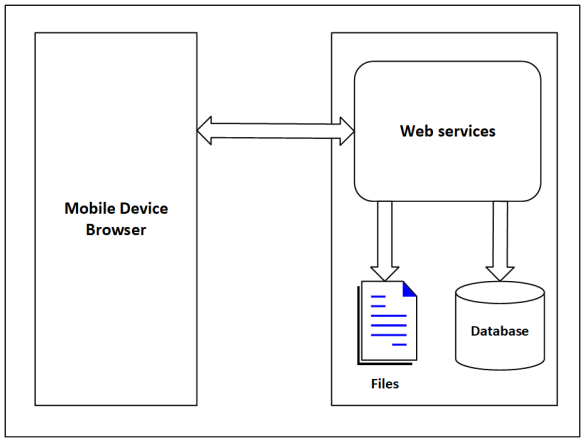


Fig. 5. The web Approach

[150]. When building very complex applications or using 3D modeling, native development is the best approach, since it requires access to external frameworks and libraries [151]. As stated by MGI Research entitled the Buyer Guide for Mobile Enterprise Application Platforms (MEAP) [150], the native architecture tends to offer a richer, more graphically engaging user experience, high performance, and the ability to integrate with features and capabilities of enterprise systems backend. The main disadvantages are (i) the development time of an application, as it is necessary to develop for the various platforms, (ii) the costs, and (iii) the various programming languages that need to be acquired.

2) Cross-platform development: There are four cross-platform approaches to mobile application development, specifically hybrid, web, interpreted, and cross-compiled.

a) The hybrid approach: The hybrid approach [152] is considered as web and native because it uses technologies such as HTML, CSS, and JavaScript, and runs within a virtual machine from a mobile device. The main advantages of this approach are that it can reuse the user interface on different native platforms, as the business logic is platform-independent and can use device resources. The main drawback is the performance when compared to native applications, as they require a web browser to run. The hybrid approach makes use of a JavaScript abstraction layer. It is, therefore, more susceptible to vulnerabilities, although it is possible to reuse the user interface for multiple platforms, making the user experience, not the best.

PhoneGap [153] is an example of a framework that follows a hybrid approach, as it is a mobile application development tool that uses web-programming languages such as HTML5, CSS3, and JavaScript.

b) The web approach: The main feature of this type of approach is that a mobile application does not need to be installed on a mobile device as it is executed through a web browser.

There are two ways to target this approach to mobile devices: responsive web design and mobile web app [150]. In the first approach, developers focus on just modifying the layout to fit a smaller screen. The only advantage is

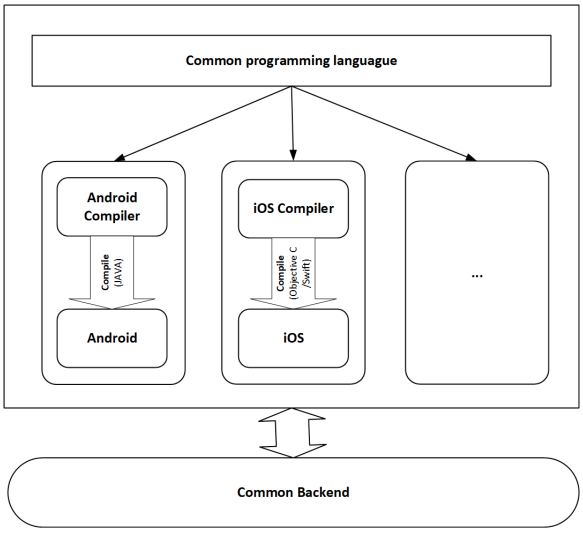


Fig. 6. The cross-compiled approach

having only a single code through languages such as HTML, JavaScript, and CSS. However, it limits the ability to create an experiment for a mobile device. In the second approach, mobile devices are detected and directed to an optimized web application where an interface is created according to specific conventions of the mobile device. The advantage of this approach is that it is not necessary to create multiple implementations of user interfaces. In addition to the above advantages, it is important to note that the application and its data are stored on the server, and no maintenance is required on the mobile device. However, this can be a disadvantage as it may have a performance lower than the other approaches by being mobile network-dependent [152]. As can be seen in Figure 5, on the left side is the browser of the mobile device responsible for placing requests to the server. On the right side is where all the business logic of the application and its data are stored.

c) The interpreted approach: In this type of approach, native code is automatically generated at runtime, having an interpreter responsible for this task. The interpreter subsequently interacts with an abstraction layer to gain access to the native APIs of each platform.

The main advantages of this approach are the efficiency of access to native interfaces and the reuse of business logic for various platforms [154]. Performance can sometimes be affected by access to this abstraction layer at runtime.

d) The cross-compiled approach: This methodology is cross-platform as it uses a common programming language. As can be seen in Figure 6, this approach makes use of a cross compiler responsible for compiling this same language into native code for a specific platform [154]. Its main advantage is that it achieves the same performance as native applications besides providing all native interface features. However, despite allowing access to some device features such as camera and GPS, if the app requires direct access to device hardware, this approach is not the best and most efficient one [155]. It is best used for simple applications.

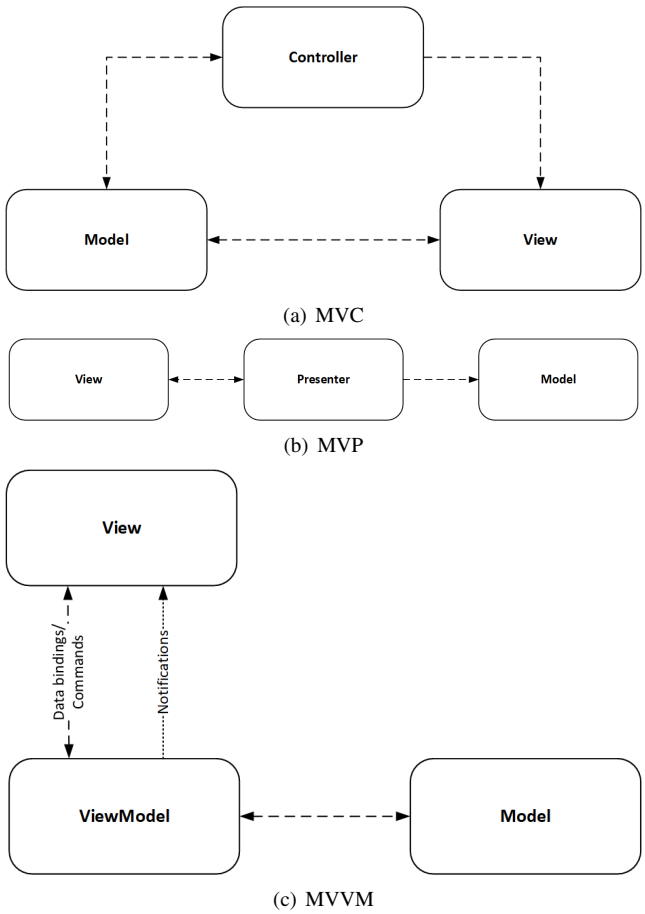


Fig. 7. Architecture patterns

B. Mobile development tools

Mobile development tools are designed to assist in the creation of mobile applications. There are two types of platforms, namely native and cross-platform. Native tools help to create specialized apps that operate ease and high quality and can take advantage of all features on their designated platform. On the other hand, cross-platform tools enable creating a generic app for multiple platforms simultaneously, significantly reducing the costs and time needed to create an app. However, non-platform specific applications tend to have more issues and lower quality compared to native ones.

1) Android Studio: Android Studio is the official Integrated Development Environment (IDE) for Google's Android OS, based on JetBrains' IntelliJ IDEA [156] software and conceived in particular for Android development. This IDE allows having a flexible build system, emulator, test tools, frameworks, and tools for parsing code for error detection. By default, there are three modules to provide access to files of a project: manifest, java, res. The manifest module contains all permissions and important information, the java module contains all Java code files, including the test code, and the res module holds all non-code resources, such as images, XML layout, etc.

2) Xcode: Xcode is a development environment for developing applications for MacOS. Xcode provides tools for building an application to its execution by providing support

for languages such as Swift and C. Integrated into this development environment, the design interface next to the implementation code can be visualized, allowing to get a visual sense of the application.

3) ViroReact: ViroReack [157] is a cross-platform as it allows developers to develop quickly Augmented Reality (AR) applications using React Native. The latter allows creating mobile applications using JavaScript. ViroReack allows to reuse code for Android and iOS and get maximum hardware performance for AR applications. The platform is comprised of two components, namely the Native 3D rendering engine and a React extension for Virtual Reality (VR) and VR development. For the development of AR, it allows integrating frameworks such as ArCore and ArKit.

4) Visual Studio: Microsoft Visual Studio [158] is an IDE from Microsoft used to develop mobile apps as well as computer programs. It uses Microsoft software development platforms and can produce both native and managed code. Its code editor supports, for example, IntelliSense, code refactoring, integrated debugger, code profiler, and designer. It can be used with many programming languages such as C++, C#, JavaScript, TypeScript, XML, HTML, and CSS. Visual Studio functionality can be significantly expanded via plug-ins. Visual Studio Code [159] is a free, lightweight cross-platform source code editor that runs on Windows, MacOS, and Linux.

5) Ionic: The Ionic Framework [160] is an open-source, cross-platform toolkit for mobile as well as desktop app development using web technologies (i.e., HTML, CSS, and JavaScript) with support for user interface (UI) frameworks such as Angular, React or Vue. It focuses on the app's frontend user experience and interaction, that is, controls, interactions, gestures, and animations. It mimics native app UI guidelines besides using native SDKs to bring together the best of the UI standards and device features of native apps. Besides, app developers can use the same code in various platforms through adaptive styling.

V. Architecture patterns

In software engineering, an architectural pattern is a reusable solution to a particular problem that occurs in a given context. It is, therefore, crucial to understand how the main parts of IoT systems are integrated and how data flows. It is vital to have a good structure when developing IoT software, and for that to happen, three patterns are presented below: Model-View-Controller (MVC), Model-View-Presenter (MVP), and Model View View Model (MVVM).

A. The MVC pattern

In software where the interface tends to change more than the business logic, it is necessary to separate application functionality from the interface. This pattern was one of the first to be conceived, and it consists of three independent components, namely View, Controller, and Model [161].

The Model component is responsible for managing business logic and manipulating data. It responds to requests for information about its state and to instructions to change that state [162]. The Controller component interprets actions the user provides, informing the Model and View components to change the data/state as requested. For instance, there may be multiple Controllers, each with a different functionality associated with a master Controller. The View component aims to present the data to the user, which is manipulated by the Controller component and later by the Model.

Figure 7(a) shows the MVC structure where both the Controller and View depend on the Model, the Controller to update data, and the View to obtain this data. However, the Model does not depend on anyone, and this is one of the main benefits of this independence since it allows the Model to be tested independently without affecting other components [163].

The MVC pattern is a common approach for mobile application development. The most essential operation of mobile applications is to retrieve data from a data store and update the user interface (UI) with the recently requested data taking into account the inputs of the user. This pattern aims to separate the components of the UI (View), core functionality and data (Model), and the user inputs' response (Controller). An example of a mobile application using this pattern is given in Section IX.

B. The MVP pattern

The MVP pattern [161] is a variant of MVC created specifically for test automation, where the goal was to increase the amount of code to be tested. The MVP pattern consists of three components, namely Model, View, and Presenter [164].

The Model component has the same concept as MVC, that is, it is responsible for responding to requests for information about its state and to instructions to change that state. View is the component responsible for displaying information, and in Android, it represents, for example, activity.xml. The difference with MVC is the Presenter component. This component aims to link the View and Model by responding to events and changing the Model. In MVC, the Controller is responsible for determining which Views will respond to any action, beside sometimes executing actions based on the events on the View. In MVP, which uses passive View pattern, actions are forwarded by the View to the Presenter.

Figure 7(b) shows the MVP structure. The View captures user events and sends them to the Presenter. The latter makes a decision and updates the former. If necessary, it can also communicate with the Model if information needs to be updated.

C. The MVVM pattern

MVVM is a pattern derived from MVC. However, it was initially designed especially for the Windows Presentation Foundation (WPF) [161]. WPF is a Microsoft system that renders user interfaces in Windows-based programs. MVVM is a pattern that is easily adapted to development platforms to separate the interface from the application logic, where the View is the responsibility of a designer rather than the programmer. The MVVM pattern consists of three components, namely View, ViewModel, and Model. The Model component is responsible for responding to information requests. The

View component is responsible for displaying the information, and in the case of WPF is defined using the eXtensible Application Markup Language (XAML). In this pattern, Views have a many-to-one relationship with ViewModels, meaning different Views can use the same ViewModel. The ViewModel component is an intermediary between the Model and the View responsible for managing the View's logic. This component interacts with the Model for information and then sends it to the View, where it will be presented to the user.

The Data Binding represented in Figure 7(c) is a mechanism that allows linking what is presented in the View to the contents of the Model. In data binding, the ViewModel does not need to notify changes via code to the View, as the latter itself knows when data is loaded [164]. An example of a Data Binding-based implementation can be seen in [165]. The ViewModel implements properties and commands to which the View can bind and, in case of change, notifies that View through events.

VI. REST AND SOAP

Web services are increasingly being used as a new emerging technology for communication between mobile applications, not being IoT applications an exception. Therefore, mobile devices can use web services to establish communication between a client and a server. Nowadays, the most essential requirement is for these services to be uninterrupted and light with regards to mobile communications [166]. Web services' communication is based on the REST architecture and the SOAP protocol.

A. Simple Object Access Protocol

The Simple Object Access Protocol (SOAP) is a communication protocol used in web services that provides a simple mechanism for exchanging structured information in a distributed and decentralized environment using eXtensible Markup Language (XML) [167]. One of the goals of SOAP design is its simplicity and extensibility. This protocol can be used on many systems ranging from message exchange to remote procedure calls (RPC). RPC defines a convention that can be used to represent procedure calls and responses. SOAP messages consist of an envelope, header, and the body of the message. The envelope identifies the XML document as a SOAP message. The header contains the optional attributes of the message, and the message body contains request and response information.

B. Representational State Transfer

The Representational State Transfer (REST) [166] is an architecture for providing standards between web computing systems to facilitate their communication. REST-based web services are easy to be used across mobile platforms because client-server communication is accomplished through a single request and response protocol. The main feature of a REST architecture in mobile applications is that it is stateless (i.e., the server does not store any state about the client session), which minimizes the volatility of the connections. It is based on Uniform Resource Identifiers (URIs), hence being easy to invoke the services. Generating a REST-based request does not imply being technology-dependent; however, for web services, it is common to use HTTP request through HEAD, POST, PUT, GET, and DELETE operations. REST responses are also based on HTTP, thus allowing obtaining a result through specific codes, and their representation can be in various formats (i.e., XML, HTML, JSON). Second, REST supports all types of data; thus, it is not necessary to interpret/format messages similarly to the SOAP protocol, in which parsing of the XML message is required. As a result, REST Web services provide greater flexibility with respect to the type of data returned. Although the SOAP framework is more secure compared to REST, factors such as integration complexity related to processing issues and resource consumption levels that apply to mobile applications, one can conclude that REST proves to be the most appropriate approach for developing mobile applications [168].

VII. RELATIONAL AND NON-RELATIONAL DATABASES

Besides communication between IoT applications, currently, various application domains exist, and a tremendous amount of data is foreseen to be produced by IoT devices. Therefore, and with the generation of heterogeneous data, many challenges arise in how to store, transfer or even manage this data efficiently. This section revisits relational and non-relational database concepts and discusses their suitability for IoT applications.

A. Relational databases

Traditional databases, which use Structured Query Language (SQL), are focused on consistency and follow the ACID (Atomicity, Consistency, Isolation, and Durability) properties when transacting [169]. Atomicity refers to transactions that are performed or discarded, and there can be no partially performed transactions. Consistency means that the system reverts to the previous stable state by propagating this information across all nodes when a transaction fails. In Isolation, transactions are performed independently, and lastly, durability refers to transactions that can be stored in records so that they are not lost if there is a need to recover the system.

Relational databases contain a set of tables, each containing columns with attributes and each row of the table representing a single instance of data. This type of database works best with structured data; nevertheless, existing some limitations regarding scalability and complexity [170]. With the increase of users and stored data, there is a need to ensure system scalability and the possibility to distribute the database across multiple servers. However, this causes problems because such databases are not designed to function with partitioned data. Another limitation is the complexity of working with relational databases, as it requires converting all data into tables even when data have not been designed for that.

B. Non-relational databases

Non-relational databases (NoSQL) do not rely on ACID properties, but rather on BASE properties (Basically Available,

Soft state, Eventual consistency), giving priority to availability and performance. The soft state consists of low data consistency across all replicas, and eventual consistency indicates that after any data change, the system does not have to reflect these changes immediately. Over time the data will be consistent across all replicas; however, consistency is not the most valued in these properties.

There are five NoSQL database types, namely key-value store, column-oriented, document-based, graph databases, and object-oriented [170], [171], [172], [173].

- *Key-value stores*, as the name suggests, are a very efficient type where values are indexed to a key, which allows users to obtain formers by specifying only the latter. They can store both structured as well as unstructured data. When prioritizing fast searches where scalability is considered rather than consistency and multiple storage options, NoSQL is a good option. Examples of this type are DynamoDB, RIAK, Redis, and Scalaris.
- *Column-oriented databases* contain one extendable column of closely related data, i.e., column family, conversely storing information in a structured table of rows and columns. This method indexes data by row, column, and timestamp. Rows and columns are identified by a key, and timestamp allows differentiating between different versions. Examples of this type are Cassandra, HyperTable, and Hbase.
- *Document-based stores* refer to databases that store information in the form of documents whose formats are XML, JSON, etc. Documents are indexed using a unique key that represents it. However, they are more complex compared to the information that is stored in the Key-value type. This type offers great performance and a horizontal scalability option. Examples of this type of database are SimpleDB, Terrastore, MongoDB, and CouchDB.
- *Graph databases* store information through a graph containing nodes and edges, where nodes act as objects and edges as the link between objects. It uses a free adjacency index technique, where each node points to the next node. This method is ACID-compliant and very scalable. An example of this type is Neo4j, Titan, and OrientDB.
- *Object-oriented databases*, data is stored as objects. This type of database can be considered as being a combination of object-oriented programming and database principles. Object-oriented is only faster when the database is object-oriented and not when the data type is simple and without large relationships. An example of this type is db4o.

NoSQL's main advantages over SQL include providing several easily scalable data models, not having a database administrator, being fast, efficient, and flexible, and managing hardware failures [172]. However, not only does it have advantages compared to relational databases, but it also has disadvantages such as the lack of a query language, being recent, and being more difficult to maintain.

C. SQL and NoSQL Databases from an IoT Perspective

As previously mentioned, SQL Database stores data in a tabular form (i.e., rows and columns) following a relational data model. In such a model, associated tables may be linked collectively. On the other hand, NoSQL databases support schema-free storage of data in various forms, such as document, graph, among others.

From an IoT perspective, in the selection of the right database between the two, the following requirements need to be considered:

1) scalability, given that SQL and NoSQL support vertical and horizontal scalability, respectively. SQL is more related to the ability to increase a single node's performance with the addition of resources such as memory or processors to it. NoSQL consists of increasing the number of nodes, thereby sharing the load among them. Horizontal scaling is more appropriate for IoT application as it allows for future expansion whenever necessary, therefore avoiding a high initial investment;
2) data retrieval, the speed of data retrieval is a must from an IoT point of view due to the need to fetch data for further processing from the database. SQL join statements may be a time-consuming process because of the structured nature of the database, in opposition to NoSQL, where the need to combine and then display data is reduced, which saves response time;
3) security, given the need to transfer sensitive data in many IoT applications, secure communication channels are desirable. However, most organizations are not familiarized with NoSQL technology, which can be prone to security issues, which is different from SQL that is more mature and has support for many security features such as authentication, data confidentiality, and integrity.

For instance, an IoT application could use a NoSQL database to analyze the relationships between different sensors, the environment where they are, how they are being used, etc., on the contrary, to simply connecting a sensor to the cloud and logging the data, which can be the case of an SQL one. In addition, a comparison of three databases, namely Cassandra and MongoDB (NoSQL databases), and PostgreSQL (SQL database) regarding data writing and reading operations is performed [174]. If there is only one client, both MongoDB and PostgreSQL achieved superior performance when performing write and read operations. If there are several clients and multiple orders, only Cassandra could perform a greater number of operations.

In summary, NoSQL is more appropriate to store IoT data due to its horizontal scalability, schema-less storage, support for unstructured data.

VIII. Software Development Methodologies

One of the first decisions faced when developing an IoT project is which development methodology should be used. The software development life cycle is a process of building software systems that includes several phases, from preliminary analysis to testing and evaluation [175]. Here, the software development lifecycle is grouped in traditional and Agile methodologies. On the other hand, there are several traditional methodologies, such as Waterfall, V-Model, and Unified Process.

A. Traditional methodologies

Traditional methodologies [175] emerged in a context where the cost of making changes to software was very high, and there was a need to plan and document everything before any implementation. Although there are several processes for software development, these methodologies are based on a sequential series of steps: requirements definition, design and planning, implementation, and testing. Each step can only be started after the end of the previous step is approved, and there is hardly any return.

The steps are:

1) The first phase allows determining the characteristics of a project, i.e., understand the requirements, their constraints, and the time it will take to carry out the various phases of the project.
2) The architectural design and planning phase allows understanding issues that may arise throughout the project and provides a guide for those participating in the project.
3) In the development phase, the project is segmented by several teams and developed according to specifications.
4) Testing steps usually overlap the development phase to ensure that the issues in the various steps are resolved early on. When the project is being finalized, the client will participate in the tests in order to get feedback.

The success of these methodologies depends on the requirements analysis and previous knowledge acquired. The primary concern of programmers is no longer meeting the requirements that were set at the beginning of the project, but rather leaving the customer satisfied with the final delivery [176]. This would only be possible if there were changes in the scope of the project by modifying the requirements and technologies previously established.

B. Agile methodology

In the early 1990s, Ken Schwaber and Jeff Sutherland developed the Scrum, which is an Agile process framework, to help organizations struggling with complex development projects. Despite also being used to explore other intricate work, research, and advanced technologies, Scrum's emphasis is on software development.

In 2001 when Ken, Jeff, and other experts came together to discuss software development processes, the Agile 'Software Development' Manifesto [177] emerged. This methodology has had a significant impact on the way software is developed, as it allows methods to deal quickly with various changes, deadlines, and needs throughout the project. This methodology is based on the following values: (i) interactions between individuals, processes, and tools, (ii) functional software according to documentation, (iii) client collaboration in contract negotiation, and (iv) the ability to respond to changes following a plan [178].

Interactions between individuals allow the exchange of information, minimize the necessary documentation and, if necessary, make changes. There are many agile methods such as Agile Modeling, Extreme Programming (XP), Scrum Development, Dynamic Software Development Method (DSDM), Feature Driven Development (FDD), Adaptive Software Development (ASD), Kanban, Lean Software Development (LSD), Rapid Application Development (RAD), Scrumban, and Crystal Methodology [179]. Each one of these methods addresses different perspectives from this methodology. For instance, DSDM requires prototyping to understand the areas of most significant risk; meanwhile, Scrum emphasizes daily meetings where work is being done. The top priority of agile methods is to deliver value to customers frequently, besides the return on investment [175]. The big difference with traditional methods is the ability to deliver results cheaply, quickly, and with requirements that can still be changed. Agile methods emphasize teamwork and rapid response to change, while conventional methods prioritize contracts, plans, and working documents.

C. IoT Software Development

Nowadays, organizations are trying to implement agile methodologies for IoT development conversely to traditional methods such as waterfall that are collapsing under the enormous weight of the IoT. That is motivated by the fact that in the past, the development process stopped with deployment. However, with IoT, there is a growing expectation for post-deployment content and updates, which means that organizations need enhanced change and management solutions as well as to think about testing over an even greater development cycle.

Alternatively, Agile methodologies are specially equipped for dealing with the requirements of the connected device. Given that frequent updates are essentially a requirement with Agile, the desire of the end-user is satisfied with a constantly updated device. In addition, it also satisfies the need of the developer for a manageable development schedule and the business requirements to respond promptly to market needs.

Usually, development tended to happen in silos; that is, each team focused on their specific task to carry the project to completion. With IoT, user expectations fuel the need for more collaboration between teams, which is the aim of the agile methodology. With it, teams are expected to able to collaborate on products in brand new ways, which will result in shorter lead times, faster overall development efficiency, and more product updates and releases.

The complexity of IoT and traditional application development is similar since the result is a system to be developed [180]. Nonetheless, and due to the nature of IoT devices, there is an extra layer of difficulty in developing IoT applications, which are considered their major challenge to be solved. Specifically, the management of a large number of devices in a distributed way, in addition to intermittent connectivity and latency in communication, which compromises the reliability of the messages exchanged.

Table II presents some of the challenges for the development of IoT applications.

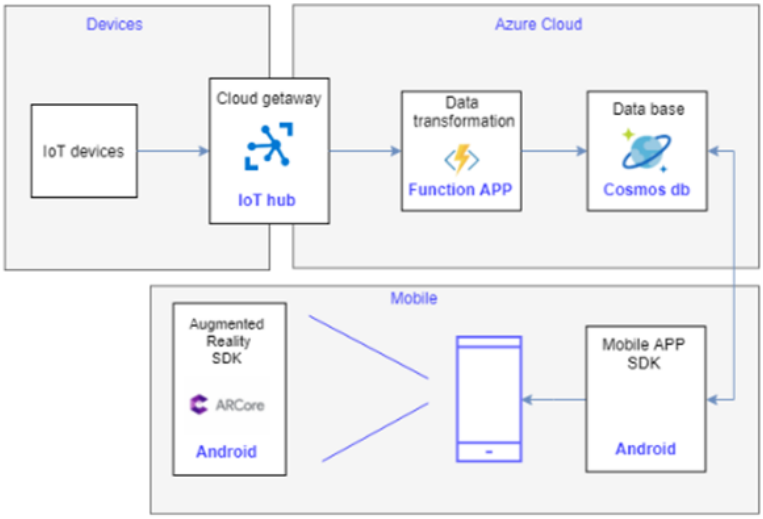


Fig. 8. Overview of a mobile solution integrating IoT and AR

TABLE II
CHALLENGES FOR THE DEVELOPMENT OF IOT APPLICATIONS

Distributed Computing	Deployment-in-the-Large and Rapid Deployment Cycles	Other Important Emerging Trends and Predictions
Heterogeneity and Diversity of Devices – the variety of devices connected to the network presents characteristics that must be considered in the process of implementing new functionalities, as well as the computational resources available for use.	*Number of IoT Devices* – the delay in system development can be directly influenced by the number of devices on the network by placing an additional layer of complexity in its management.	*Cellular IoT Radio Technologies* – the communication of IoT devices will be performed through Cellular IoT Radio Technologies with the Cloud without the need for additional Bluetooth settings or Wi-Fi security mechanisms.
Unreliable Connectivity – the size and complexity of the IoT network grow as new devices are connected, contributing to the delay in messages and difficulty in managing routes to minimize their effects, which can make the data collected unreliable.	*Delay and Synchronization* – as IoT devices can perform critical tasks, the software update process may have to be delayed and synchronized until all devices receive the updates.	*Intelligence* – intelligent mechanisms will enable data processing and management via edge devices by removing centralized processing from cloud applications.
Software Fault-tolerant – the occurrence of network failures should be a factor to be considered when implementing the systems since the data collected by the IoT sensors are unique, and their non-processing could drastically influence the performance of the generated output.	*DevOps methodologies* – System updates can occur in different phases throughout the day.	*Isomorphic IoT system architectures* – is about increasing the resource capacity of IoT devices, for which they have supported new applications, in addition to facilitating the migration of applications.

IX. AN OVERVIEW OF A MOBILE SOLUTION INTEGRATING IOT AND AR

The impacts of AR and IoT on Field Force Management[2] focus essentially on optimizing time and costs when a piece of equipment needs maintenance because it allows a technician to obtain and view information that is difficult to obtain manually. Approximately 50% of the time used to maintain a piece of equipment is spent exclusively in locating/identifying the problem [181]. Besides, the use of AR and IoT in maintenance procedures can reduce the time of a task and the mistakes that could be made. Repairing modules by viewing a 3D model of the equipment or obtaining sensor information and repair instructions are examples where AR and IoT can be adopted. In addition to these features, the ability to have a sensor-equipped infrastructure allows to anticipate and prevent future problems, significantly reducing equipment downtime.

[2]Field Force Management is the department responsible for managing a company's resources or employees.

The main motivation here is the development of a mobile application to provide support in troubleshooting and prevention of equipment problems. Thus, by using IoT, one can obtain certain information from a device and view it together with the 3D model of the device using AR in a mobile application. Another motivation is the support of AR instructions to make the repair of equipment more efficient.

Figure 8 presents a description of the solution, including the different IoT and AR components. The mobile solution overview can be segmented into three blocks, namely devices, cloud, and mobile. The cloud block uses Azure PaaS services to create IoT solutions. IoT devices contain sensors connected directly to a machine and can log securely to the cloud to send and receive data. The Cloud gateway is responsible for receiving information from devices. Azure IoT hub [182] is a service that enables reliable two-way communications between IoT devices and the solution backend. In this case, it is a message broker between the devices and the data transformation service. Data transformation manipulates the

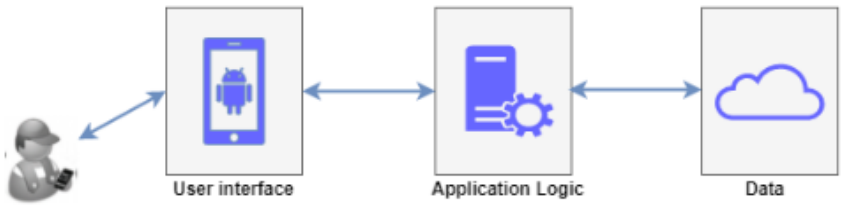


Fig. 9. Mobile solution architecture

telemetry flow, for instance, by converting various types of data into JSON format, which in this particular case is called Function APP and serves to extract information from events that were generated. Azure features have integration with the IoT hub and Cosmos DB. Cosmos DB is classified as a NoSQL database that receives the information from the IoT hub and then generates a collection of documents with the same information. Each device contains a unique identifier (id) so that information can be stored in the database according to this id.

In the mobile solution block, an android SDK is used for the SQL Application Programming Interface (API) of the Azure Cosmos DB in order to obtain the stored information. The SDK is used as a database interface that supports offline persistence of reading and writing documents. When it is necessary to have access to documents stored in the database of their devices, a user, through SQL queries, can get the information to later view in AR.

In the AR component, another SDK called ArCore, which is an open-source framework from Google to create AR applications, is used. In this context, ArCore is used to reproduce the 3D models of an object and, in addition, with the information stored in the Cosmos DB from IoT, visualize its respective problems in that same 3D object.

As can be seen in Figure 9, the mobile solution is divided into three layers: (i) the presentation layer; (ii) the application layer; (iii) the data access layer. This solution was built based on the MVC architecture standard; and , in Figure 9 the View can be seen as the User Interface, the Controller can be seen as the Application Logic, and the Model can be seen as the Data. The user interface, as its name suggests, is the layer that makes the user interact with the application. In this layer are located all the XML files responsible for the content presentation.

Application logic is the layer responsible for managing communication between the user interface and database. It acts as an intermediary between the mobile application and the database, querying whenever necessary to retrieve or change data in the database.

The data access layer corresponds to the connections made to the database where the information is stored. In the mobile solution, it corresponds to classes responsible for querying Cosmos DB through an SDK, which serves as an interface with the database. Whenever there are certain events at the application layer, it communicates with the data access layer that later communicates with the database. Sometimes the application layer can also communicate with the database due to the asynchronous tasks that the application is performing. This wire standard was adopted due to its great use in android applications and ease of implementation.

X. Challenges and Open Issues

With the exponential increase in the number of devices that constitute the IoT network, besides their diverse mobility characteristics and heterogeneity, the challenge of developing optimized mobile IoT solutions that meet the application requirements and the nodes that provide network services increases. Below, some of the problems that need to be better addressed are listed:

- *Delay:* The message exchange delay between different devices working or not in a coordinated manner plays a fundamental role so that the problem of information loss is reduced. Developers need to ensure that the minimum amount of information is lost, for example, when high packet delay or loss occurs [183], [184].
- *Communication Protocols:* The selection of a communication protocol to develop mobile solutions in different situations depends on the development team [185]. The development of new protocols or modifications of existing solutions is a nontrivial task since it must support numerous devices exchanging data over the network without reducing the network nodes' performance.
- *Data management:* The development team needs to take into account different aspects such as the secure storage of collected information and the design of optimized systems that are capable of performing queries in large databases, in addition to ensuring the lowest consumption of devices' resources. Besides managing the storage, information security aspects must be brought to light by developers, such as the development of contingency measures in the occurrence of data failure or inconsistencies, access control to the collected data, among others.
- *Design and Development:* Good software quality is a vast field [186], and developers have to not only produce applications but also produce them in a way that is modular, easy to understand, maintain, evolve, and capable of scaling. The right choice of architectural standards and design patterns directly affect how much maintainability the application will have over time, as well as providing a low Technical Debt for later development teams [187],

[188]. Bad architectural design is an easy issue to slip into and leads to application crashes, inconsistent user interface, high energy expenditure, besides making maintenance difficult over time. The first two factors affect the user experience, and many users abandon the application in its first uses [189]

- *Architectural Pattern:* During the system development process, developers can choose architectural patterns for software development according to their experiences. The latter is not trivial as it is necessary to take into account some characteristics of the system to be developed [190], that is, (1) dynamic scenario, as the devices' behavior varies according to the environment; (2) complexity, due to the different communication technologies, protocols and device characteristics; (3) reliability, as the system modeling process must bring to light the reliability requirement to reduce problems in information exchange between different devices; (4) security, as it must be guaranteed that information exchanged between third party systems is not performed in an unauthorized way; (5) software and hardware integration, given the need to ensure integration between different hardware and software.

In addition to the challenges that increase the level of complexity for the development of optimized mobile IoT solutions, the following aspects should also be considered:

- *Quality Control:* As mobile IoT apps gain complexity and are applied in critical areas such as healthcare, developers need to ensure that apps are error-free and safe, as they can put human lives at risk.
- *User-Friendly Design:* developers need to provide solutions that are not complex at the level of utilization. However, such solutions must be robust, reliable, and customizable according to the case.
- *Cross-Platform Deployment:* IoT devices are heterogeneous at the hardware and software levels, and different operating systems, protocols, and architectures can be used in real application scenarios. Developers should consider such features when developing mobile IoT solutions.

XI. Future Directions

The development of mobile IoT solutions goes through different processes of analysis and validation of the proposed solutions, whether regarding security and privacy aspects, resources management, or data gathering/storage, among others [191], [192]. To this end, some of the open research opportunities are listed below:

- *Privacy:* The gathering of personal information by IoT devices occurs in a broad manner and without clearly specifying which data or information is being collected and processed, in addition to the lack of appropriate security mechanisms to ensure that confidential information is not public or even that no information leakage occurs. For instance, in 2018, the General Data Protection Regulation (GDPR) [193] was introduced by the European Union to regulate the gathering and processing of personal information in computing environments. Compliance with the regulation applied to the mobile IoT solutions is a broad and promising field to be explored, as different personal devices are in a constant process of collecting and processing user data.
- *Deep Learning (DL):* The development of applications that use optimized computing resources is one of the significant challenges of IoT, as new services/applications arise that demand greater availability of resources. The use of DL, in the context of mobile IoT, allows its use in different scenarios such as data collection and analysis [194], [195], task optimization [196], IoT network optimization [197], among others.
- *Security:* Although a great effort is put into developing new security tools and methodologies, the dynamic mobile IoT landscape enables the appearance of new security breaches that can compromise the application/device/network and information theft. However, even if different techniques and mechanisms, coupled with Artificial Intelligence algorithms, are widely proposed/adopted, different attacks remain without 100% detection rates or optimized solutions.
- *Augmented Reality:* The use of augmented reality (AR) has been gaining space in recent years by being applied in the various scenarios of IoT applications. The development of mobile applications encompasses different areas of knowledge, ranging from health [198], education [199], and industrial [139] applications. An in-depth study of how the dynamic characteristics of IoT devices can influence the performance of AR solutions, as well as the modeling process of IoT systems, can be affected by their use, is necessary.

XII. Conclusion

The popularity of mobile apps has escalated almost three decades since their first appearance. These days, businesses from several domains use mobile apps strategically to reach as many customers as possible. IoT, on the other hand, is an ecosystem of networked devices that share data to improve efficiency and to serve customers better. Mobile app development can be used to manage and improve the apps development process, therefore, enabling companies to gain a competitive advantage. It also provides support in troubleshooting and prevention of equipment problems. Additionally, the ability to have a sensor-equipped infrastructure allows, for example, to anticipate and prevent future problems, significantly reducing equipment downtime and costs.

Acknowledgment

The authors would like to thank Ana Paula Afonso. This work was sponsored by FCT through the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020.

References

[1] M. J. Cronin, *Smartphones and wireless services*. Cambridge University Press, 2010, p. 135–168.

[2] Z. Rashid, J. Melià-Seguí, R. Pous, and E. Peig, "Using Augmented Reality and Internet of Things to improve accessibility of people with motor disabilities in the context of Smart Cities," *Future Generation Computer Systems*, vol. 76, pp. 248–261, nov 2017.

[3] F. Zambonelli, "Key abstractions for iot-oriented software engineering," *IEEE Software*, vol. 34, no. 1, pp. 38–45, Jan 2017.

[4] R. Jabangwe, H. Edison, and A. N. Duc, "Software engineering process models for mobile app development: A systematic literature review," *Journal of Systems and Software*, vol. 145, pp. 98 – 111, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121218301638

[5] A. I. Wasserman, "Software engineering issues for mobile application development," in *Proceedings of the FSE/SDP Workshop on the Future of Software Engineering Research, FoSER 2010*. New York, New York, USA: ACM Press, 2010, pp. 397–400. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1882362.1882443

[6] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, sep 2012.

[7] V. G. Cerf and R. E. Kahn, "A protocol for packet network intercommunication," *Computer Communication Review*, vol. 35, no. 2, pp. 71–82, 2005.

[8] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *Journal of Electrical and Computer Engineering*, vol. 2017, pp. 1–25, jan 2017.

[9] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du, "Research on the architecture of Internet of Things," in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*. IEEE, aug 2010, pp. 484–487.

[10] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," Tech. Rep. [Online]. Available: http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

[11] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*. New York, New York, USA: ACM Press, 2012, p. 13.

[12] N. Niknejad, W. Ismail, I. Ghani, B. Nazari, M. Bahari, and A. R. B. C. Hussin, "Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation," *Information Systems*, vol. 91, p. 101491, 2020. [Online]. Available: https://doi.org/10.1016/j.is.2020.101491

[13] E. El-Sheikh, A. Zimmermann, and L. C. Jain, "Emerging Trends and Enterprise Service-Oriented in the Evolution of Architectures," *Intelligent Systems Reference Library*, vol. 111, pp. 1–265, 2016. [Online]. Available: https://www.springer.com/gp/book/9783319405629

[14] O. Zimmermann, "Microservices tenets: Agile approach to service development and deployment," *Computer Science - Research and Development*, vol. 32, no. 3-4, pp. 301–310, 2017.

[15] H. Muccini and M. T. Moghaddam, "Iot architectural styles," in *Software Architecture*, C. E. Cuesta, D. Garlan, and J. Pérez, Eds. Cham: Springer International Publishing, 2018, pp. 68–85.

[16] J. Thönes, "Microservices," *IEEE Software*, vol. 32, no. 1, pp. 116–116, Jan 2015.

[17] B. Butzin, F. Golatowski, and D. Timmermann, "Microservices approach for the internet of things," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2016, pp. 1–6.

[18] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, nov 2012.

[19] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.

[20] A. Čolaković and M. Hadžialić, "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues," *Computer Networks*, vol. 144, pp. 17–39, oct 2018.

[21] V. Karagiannis, F. Vázquez-Gallego, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the Internet of Things Communication Protocols for the Internet of Things (IoT) View project Channel Hopping and Blacklisting Strategies to Improve the Performance of IEEE 802.15.4 TSCH View project A Survey on Application Layer Protocols for the Internet of Things," Tech. Rep., 2015.

[22] J. Kurose and K. Ross, *Computer Networking - A top-down approach*. Pearson, 2012.

[23] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," Tech. Rep., 2008. [Online]. Available: https://tools.ietf.org/html/rfc5246

[24] D. Thangavel, X. Ma, A. Valera, H. Tan, and C. K. Tan, "Performance evaluation of mqtt and coap via a common middleware," in *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, April 2014, pp. 1–6.

[25] S. Vinoski, "Advanced Message Queuing Protocol," *IEEE Internet Computing*, vol. 10, no. 6, pp. 87–89, nov 2006.

[26] F. T. Johnsen, T. H. Bloebaum, M. Avlesen, S. Spjelkavik, and B. Vik, "Evaluation of transport protocols for web services," in *2013 Military Communications and Information Systems Conference*, Oct 2013, pp. 1–6.

[27] J. L. Fernandes, I. C. Lopes, J. J. P. C. Rodrigues, and S. Ullah, "Performance evaluation of restful web services and amqp protocol," in *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2013, pp. 810–815.

[28] Advanced Message Queuing Protocol, "Products and Success Stories," 2017. [Online]. Available: http://www.amqp.org/about/examples

[29] Crossbar.io Technologies GmbH, "The Web Application Messaging Protocol — Web Application Messaging Protocol version 2 documentation." [Online]. Available: https://wamp-proto.org/

[30] P. Kayal and H. Perros, "A comparison of iot application layer protocols through a smart parking implementation," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, March 2017, pp. 331–336.

[31] P. Pietzuch and J. Bacon, "Hermes: a distributed event-based middleware architecture," in *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*. IEEE Comput. Soc, pp. 611–618.

[32] S. K. Lee, M. Bae, and H. Kim, "Future of IoT networks: A survey," *Applied Sciences (Switzerland)*, vol. 7, no. 10, pp. 1–25, 2017.

[33] S. Li, L. D. Xu, and S. Zhao, "5G Internet of Things: A survey," vol. 10, pp. 1–9, 2018.

[34] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges," vol. 6, pp. 3619–3647, 2017.

[35] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," vol. 7, no. 1, pp. 16–32, 2020.

[36] Y. B. Lin, H. C. Tseng, Y. W. Lin, and L. J. Chen, "NB-IoTtalk: A Service Platform for Fast Development of NB-IoT Applications," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 928–939, 2019.

[37] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of LoRaWAN for IoT: From technology to application," *Sensors (Switzerland)*, vol. 18, no. 11, 2018.

[38] R. S. Sinha, Y. Wei, and S. H. Hwang, "A survey on LPWA technology: LoRa and NB-IoT," pp. 14–21, 2017.

[39] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling Technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[40] F. Jalali, S. Khodadustan, C. Gray, K. Hinton, and F. Suits, "Greening IoT with Fog: A Survey," in *Proceedings - 2017 IEEE 1st International Conference on Edge Computing, EDGE 2017*, 2017, pp. 25–31.

[41] F. Jalali, A. Vishwanath, J. De Hoog, and F. Suits, "Interconnecting Fog computing and microgrids for greening IoT," in *IEEE PES Innovative Smart Grid Technologies Conference Europe*. IEEE, 2016, pp. 693–698.

[42] K. Ahmad, O. Mohammad, M. Atieh, and H. Ramadan, "IoT: Architecture, Challenges, and Solutions Using Fog Network and Application Classification," in *ACIT 2018 - 19th International Arab Conference on Information Technology*. IEEE, 2019.

[43] O. Salman, I. Elhajj, A. Chehab, and A. Kayssi, "IoT survey: An SDN and fog computing perspective," *Computer Networks*, vol. 143, no. 2018, pp. 221–246, 2018.

[44] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, "A survey on fog computing for the Internet of Things," *Pervasive and Mobile Computing*, vol. 52, pp. 71–99, 2019.

[45] F. Li, M. Vögler, M. Claeßens, and S. Dustdar, "Towards automated iot application deployment by a cloud-based approach," in *Proceedings - IEEE 6th International Conference on Service-Oriented Computing and Applications, SOCA 2013*, 2013, pp. 61–68.

[46] P. P. Ray, "A survey of IoT cloud platforms," *Future Computing and Informatics Journal*, vol. 1, no. 1-2, pp. 35–46, 2016.

[47] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A Survey on the Edge Computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2017.

[48] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, 2018.

[49] H. Djelouat, A. Amira, and F. Bensaali, "Compressive sensing-based IoT applications: A review," vol. 7, no. 4, 2018.

[50] K. AbualSaud, T. M. Elfouly, T. Khattab, E. Yaacoub, L. S. Ismail, M. H. Ahmed, and M. Guizani, "A Survey on Mobile Crowd-Sensing and Its Applications in the IoT Era," *IEEE Access*, vol. 7, pp. 3855–3881, 2019.

[51] P. M. Santos, J. G. Rodrigues, S. B. Cruz, T. Lourenço, P. M. D'Orey, Y. Luis, C. Rocha, S. Sousa, S. Crisóstomo, C. Queirós, S. Sargento, A. Aguiar, and J. Barros, "PortoLivingLab: An IoT-Based Sensing Platform for Smart Cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 523–532, 2018.

[52] A. Noda, O. Fukuda, H. Okumura, and K. Arai, "Behavior analysis of a small animal using IoT sensor system," in *ICIIBMS 2017 - 2nd International Conference on Intelligent Informatics and Biomedical Sciences*, vol. 2018-Janua, 2018, pp. 9–10.

[53] A. P. Plageras, K. E. Psannis, C. Stergiou, H. Wang, and B. B. Gupta, "Efficient IoT-based sensor BIG Data collection–processing and analysis in smart buildings," *Future Generation Computer Systems*, vol. 82, pp. 349–357, 2018.

[54] J. Yang, H. Zou, H. Jiang, and L. Xie, "Device-Free Occupant Activity Sensing Using WiFi-Enabled IoT Devices for Smart Homes," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3991–4002, 2018.

[55] M. Wazid, A. K. Das, R. Hussain, G. Succi, and J. J. Rodrigues, "Authentication in cloud-driven IoT-based big data environment: Survey and outlook," *Journal of Systems Architecture*, vol. 97, no. September 2018, pp. 185–196, 2019.

[56] D. Mourtzis, E. Vlachou, and N. Milas, "Industrial Big Data as a Result of IoT Adoption in Manufacturing," in *Procedia CIRP*, vol. 55. The Author(s), 2016, pp. 290–295.

[57] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[58] F. Alam, R. Mehmood, I. Katib, N. N. Albogami, and A. Albeshri, "Data Fusion and IoT for Smart Ubiquitous Environments: A Survey," *IEEE Access*, vol. 5, pp. 9533–9554, 2017.

[59] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," vol. 20, no. 4, pp. 2923–2960, 2018.

[60] A. Kanawaday and A. Sane, "Machine learning for predictive maintenance of industrial machines using IoT sensor data," in *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, vol. 2017-Novem, no. Figure 2, 2018, pp. 87–90.

[61] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for Internet of Things," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 8, pp. 1399–1417, 2018.

[62] W. L. Chen, Y. B. Lin, F. L. Ng, C. Y. Liu, and Y. W. Lin, "RiceTalk: Rice Blast Detection Using Internet of Things and Artificial Intelligence Technologies," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1001–1010, feb 2020.

[63] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, apr 2020.

[64] F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato, "An Intelligent Traffic Load Prediction-Based Adaptive Channel Assignment Algorithm in SDN-IoT: A Deep Learning Approach," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5141–5154, 2018.

[65] J. Tang, D. Sun, S. Liu, and J. L. Gaudiot, "Enabling Deep Learning on IoT Devices," *Computer*, vol. 50, no. 10, pp. 92–96, 2017.

[66] H. Li, K. Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[67] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J. S. Oh, "Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 624–635, 2018.

[68] S. K. Singh, S. Rathore, and J. H. Park, "BlockIoTIntelligence: A Blockchain-enabled Intelligent IoT Architecture with Artificial Intelligence," *Future Generation Computer Systems*, vol. 110, pp. 721–743, 2020.

[69] C. A. Medina, M. R. Perez, and L. C. Trujillo, "IoT paradigm into the smart city vision: a survey," in *Proceedings - 2017 IEEE International Conference on Internet of Things, IEEE Green Computing and Communications, IEEE Cyber, Physical and Social Computing, IEEE Smart Data, iThings-GreenCom-CPSCom-SmartData 2017*, vol. 2018-Janua, 2018, pp. 695–704.

[70] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-Khah, and P. Siano, "Iot-based smart cities: A survey," in *EEEIC 2016 - International Conference on Environment and Electrical Engineering*. IEEE, 2016, pp. 2–7.

[71] F. Al-Turjman and A. Malekloo, "Smart parking in IoT-enabled cities: A survey," *Sustainable Cities and Society*, vol. 49, no. December 2018, 2019.

[72] K. Pardini, J. J. Rodrigues, S. A. Kozlov, N. Kumar, and V. Furtado, "IoT-based solid waste management solutions: A survey," *Journal of Sensor and Actuator Networks*, vol. 8, no. 1, 2019.

[73] P. Fraga-Lamas, T. M. Fernández-Caramés, and L. Castedo, "Towards the internet of smart trains: A review on industrial IoT-connected railways," *Sensors (Switzerland)*, vol. 17, no. 6, 2017.

[74] R. Ramakrishnan and L. Gaur, "Smart electricity distribution in residential areas: Internet of Things (IoT) based advanced metering infrastructure and cloud analytics," in *2016 International Conference on Internet of Things and Applications, IOTA 2016*. Institute of Electrical and Electronics Engineers Inc., sep 2016, pp. 46–51.

[75] A. Chowdhury, "Priority based and secured traffic management system for emergency vehicle using IoT," in *Proceedings - 2016 International Conference on Engineering and MIS, ICEMIS 2016*. IEEE, 2016, pp. 1–6.

[76] Z. Lv, B. Hu, and H. Lv, "Infrastructure Monitoring and Operation for Smart Cities Based on IoT System," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1957–1962, 2020.

[77] G. Mei, N. Xu, J. Qin, B. Wang, and P. Qi, "A Survey of Internet of Things (IoT) for Geohazard Prevention: Applications, Technologies, and Challenges," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4371–4386, 2020.

[78] J. Shah and B. Mishra, "IoT enabled environmental monitoring system for smart cities," in *2016 International Conference on Internet of Things and Applications, IOTA 2016*. IEEE, 2016, pp. 383–388.

[79] S. L. Ullo and G. R. Sinha, "Advances in smart environment monitoring systems using iot and sensors," *Sensors (Switzerland)*, vol. 20, no. 11, 2020.

[80] M. S. Jamil, M. A. Jamil, A. Mazhar, A. Ikram, A. Ahmed, and U. Munawar, "Smart Environment Monitoring System by Employing Wireless Sensor Networks on Vehicles for Pollution Free Smart Cities," in *Procedia Engineering*, vol. 107. Elsevier B.V., 2015, pp. 480–484.

[81] A. Alshamsi, Y. Anwar, M. Almulla, M. Aldohoori, N. Hamad, and M. Awad, "Monitoring pollution: Applying IoT to create a smart environment," in *2017 International Conference on Electrical and Computing Technologies and Applications, ICECTA 2017*, vol. 2018-Janua, 2017, pp. 1–4.

[82] M. Suresh, U. Muthukumar, and J. Chandapillai, "A novel smart water-meter based on IoT and smartphone app for city distribution management," in *TENSYMP 2017 - IEEE International Symposium on Technologies for Smart Cities*, 2017, pp. 1–5.

[83] S. Wadekar, V. Vakare, R. Prajapati, S. Yadav, and V. Yadav, "Smart water management using IOT," in *2016 5th International Conference on Wireless Networks and Embedded Systems, WECON 2016*. IEEE, 2017, pp. 3–6.

[84] P. P. Shah, A. A. Patil, and S. S. Ingleshwar, "IoT based smart water tank with Android application," in *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017*, 2017, pp. 600–603.

[85] C. Kamienski, J. P. Soininen, M. Taumberger, R. Dantas, A. Toscano, T. S. Cinotti, R. F. Maia, and A. T. Neto, "Smart water management platform: IoT-based precision irrigation for agriculture," *Sensors (Switzerland)*, vol. 19, no. 2, pp. 6–11, 2019.

[86] M. Saravanan, A. Das, and V. Iyer, "Smart water grid management using LPWAN IoT technology," in *GIoTS 2017 - Global Internet of Things Summit, Proceedings*, 2017.

[87] R. Bikmetov, M. Y. A. Raja, and T. U. Sane, "Infrastructure and applications of Internet of Things in smart grids: A survey," in *2017 North American Power Symposium, NAPS 2017*, 2017.

[88] M. Carratu, M. Ferro, A. Pietrosanto, and V. Paciello, "Smart Power Meter for the IoT," in *Proceedings - IEEE 16th International Conference on Industrial Informatics, INDIN 2018*, 2018, pp. 514–519.

[89] F. Al-Turjman and M. Abujubbeh, "IoT-enabled smart grid via SM: An overview," *Future Generation Computer Systems*, vol. 96, pp. 579–590, 2019.

[90] M. Aboelmaged, Y. Abdelghani, and M. A. E. Ghany, "Wireless IoT based metering system for energy efficient smart cites," in *Proceedings*

of the International Conference on Microelectronics, ICM, vol. 2017-Decem, no. Icm. IEEE, 2018, pp. 1–4.

[91] M. Pennacchioni, M. G. Di Benedette, T. Pecorella, C. Carlini, and P. Obino, "NB-IoT system deployment for smart metering: Evaluation of coverage and capacity performances," in *2017 AEIT International Annual Conference: Infrastructures for Energy and ICT: Opportunities for Fostering Innovation, AEIT 2017*, vol. 2017-Janua, no. 1, 2017, pp. 1–6.

[92] J. Lloret, J. Tomas, A. Canovas, and L. Parra, "An Integrated IoT Architecture for Smart Metering," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 50–57, 2016.

[93] S. K. Datta, J. L. Dugelay, and C. Bonnet, "IoT Based UAV Platform for Emergency Services," in *9th International Conference on Information and Communication Technology Convergence: ICT Convergence Powered by Smart Intelligence, ICTC 2018*. IEEE, 2018, pp. 144–147.

[94] Y. Krytska, I. Skarga-Bandurova, and A. Velykzhanin, "IoT-based situation awareness support system for real-Time emergency management," in *Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017*, vol. 2, 2017, pp. 955–960.

[95] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-Based IoT Platform: A Crowd Surveillance Use Case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, 2017.

[96] N. Al-Nabhan, N. Al-Aboody, and A. B. Alim Al Islam, "A hybrid IoT-based approach for emergency evacuation," *Computer Networks*, vol. 155, pp. 87–97, 2019.

[97] D. B. Arbia, M. M. Alam, A. Kadri, E. B. Hamida, and R. Attia, "Enhanced IoT-based end-to-end emergency and disaster relief system," *Journal of Sensor and Actuator Networks*, vol. 6, no. 3, pp. 1–18, 2017.

[98] A. F. Santamaria, P. Raimondo, M. Tropea, F. De Rango, and C. Aiello, "An IoT surveillance system based on a decentralised architecture," *Sensors (Switzerland)*, vol. 19, no. 6, 2019.

[99] N. Patil, S. Ambatkar, and S. Kakde, "IoT based smart surveillance security system using raspberry Pi," in *Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017*, vol. 2018-Janua, 2018, pp. 344–348.

[100] R. Dautov, S. Distefano, D. Bruneo, F. Longo, G. Merlino, A. Puliafito, and R. Buyya, "Metropolitan intelligent surveillance systems for urban areas by harnessing IoT and edge computing paradigms," *Software - Practice and Experience*, vol. 48, no. 8, pp. 1475–1492, 2018.

[101] A. P. Plageras, K. E. Psannis, Y. Ishibashi, and B. G. Kim, "IoT-based surveillance system for ubiquitous healthcare," in *IECON Proceedings (Industrial Electronics Conference)*, vol. 0. IEEE, 2016, pp. 6226–6230.

[102] S. A. I. Quadri and P. Sathish, "IoT based home automation and surveillance system," in *Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems, ICICCS 2017*, vol. 2018-Janua. Institute of Electrical and Electronics Engineers Inc., jul 2017, pp. 861–866.

[103] L. Hu and Q. Ni, "IoT-Driven Automated Object Detection Algorithm for Urban Surveillance Systems in Smart Cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 747–754, 2018.

[104] A. Khanna and R. Tomar, "IoT based interactive shopping ecosystem," in *Proceedings on 2016 2nd International Conference on Next Generation Computing Technologies, NGCT 2016*, no. October. IEEE, 2017, pp. 40–45.

[105] H. Fu, G. Manogaran, K. Wu, M. Cao, S. Jiang, and A. Yang, "Intelligent decision-making of online shopping behavior based on internet of things," *International Journal of Information Management*, vol. 50, no. April 2019, pp. 515–525, 2020.

[106] J. Rezazadeh, K. Sandrasegaran, and X. Kong, "A location-based smart shopping system with IoT technology," in *IEEE World Forum on Internet of Things, WF-IoT 2018 - Proceedings*, vol. 2018-Janua. IEEE, 2018, pp. 748–753.

[107] R. Li, T. Song, N. Capurso, J. Yu, and X. Cheng, "IoT applications on secure smart shopping," in *Proceedings - 2016 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI 2016*, vol. 2018-Janua, no. 6, 2018, pp. 238–243.

[108] N. N. Dlamini and K. Johnston, "The use, benefits and challenges of using the Internet of Things (IoT) in retail businesses: A literature review," in *Proceedings - 2016 3rd International Conference on Advances in Computing, Communication and Engineering, ICACCE 2016*. IEEE, 2017, pp. 430–436.

[109] M. S. Balaji and S. K. Roy, "Value co-creation with Internet of things technology in the retail industry," *Journal of Marketing Management*, vol. 33, no. 1-2, pp. 7–31, 2017.

[110] L. Liu, B. Zhou, Z. Zou, S. C. Yeh, and L. Zheng, "A Smart Unstaffed Retail Shop Based on Artificial Intelligence and IoT," in *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD*, vol. 2018-Septe. IEEE, 2018, pp. 2018–2021.

[111] F. Caro and R. Sadr, "The Internet of Things (IoT) in retail: Bridging supply and demand," *Business Horizons*, vol. 62, no. 1, pp. 47–54, 2019.

[112] Y. Zhang, Z. Guo, J. Lv, and Y. Liu, "A Framework for Smart Production-Logistics Systems Based on CPS and Industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4019–4032, sep 2018.

[113] E. Manavalan and K. Jayakrishna, "A review of Internet of Things (IoT) embedded sustainable supply chain for industry 4.0 requirements," *Computers and Industrial Engineering*, vol. 127, no. November 2017, pp. 925–953, 2019.

[114] Y. Zhang, L. Zhao, and C. Qian, "Modeling of an IoT-enabled supply chain for perishable food with two-echelon supply hubs," *Industrial Management and Data Systems*, vol. 117, no. 9, pp. 1890–1905, 2017.

[115] A. R. Laxmi and A. Mishra, "RFID based Logistic Management System using Internet of Things (IoT)," in *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, no. Iceca. IEEE, 2018, pp. 556–559.

[116] C. N. Verdouw, R. M. Robbemond, T. Verwaart, J. Wolfert, and A. J. Beulens, "A reference architecture for IoT-based logistic information systems in agri-food supply chains," *Enterprise Information Systems*, vol. 12, no. 7, pp. 755–779, 2018.

[117] A. K. Ghosh, A. M. Ullah, and A. Kubo, "Hidden Markov model-based digital twin construction for futuristic manufacturing systems," in *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 33, no. 3, 2019, pp. 317–331.

[118] D. Kwon, M. R. Hodkiewicz, J. Fan, T. Shibutani, and M. G. Pecht, "IoT-Based Prognostics and Systems Health Management for Industrial Applications," *IEEE Access*, vol. 4, pp. 3659–3670, 2016.

[119] J. Cheng, W. Chen, F. Tao, and C. L. Lin, "Industrial IoT in 5G environment towards smart manufacturing," *Journal of Industrial Information Integration*, vol. 10, no. March, pp. 10–19, 2018.

[120] Z. Meng, Z. Wu, C. Muvianto, and J. Gray, "A Data-Oriented M2M Messaging Mechanism for Industrial IoT Applications," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 236–246, 2017.

[121] G. Oguntala, R. Abd-Alhameed, S. Jones, J. Noras, M. Patwary, and J. Rodriguez, "Indoor location identification technologies for real-time IoT-based applications: An inclusive survey," *Computer Science Review*, vol. 30, pp. 55–79, 2018.

[122] S. Mumtaz, A. Alsohaily, Z. Pang, A. Rayes, K. F. Tsang, and J. Rodriguez, "Massive Internet of Things for Industrial Applications," *Indutrial Electronics Magazine*, no. march, pp. 28–33, 2017.

[123] D. Raposo, A. Rodrigues, S. Sinche, J. S. Silva, and F. Boavida, "Industrial IoT monitoring: Technologies and architecture proposal," *Sensors (Switzerland)*, vol. 18, no. 10, pp. 1–32, 2018.

[124] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[125] J. M. Talavera, L. E. Tobón, J. A. Gómez, M. A. Culman, J. M. Aranda, D. T. Parra, L. A. Quiroz, A. Hoyos, and L. E. Garreta, "Review of IoT applications in agro-industrial and environmental fields," pp. 283–297, 2017.

[126] M. S. Mekala and P. Viswanathan, "A Survey: Smart agriculture IoT with cloud computing," in *2017 International Conference on Microelectronic Devices, Circuits and Systems, ICMDCS 2017*, vol. 2017-Janua, 2017, pp. 1–7.

[127] N. Ahmed, D. De, and I. Hussain, "Internet of Things (IoT) for Smart Precision Agriculture and Farming in Rural Areas," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4890–4899, 2018.

[128] A. Kamilaris, F. Gao, F. X. Prenafeta-Boldu, and M. I. Ali, "Agri-IoT: A semantic framework for Internet of Things-enabled smart farming applications," in *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, 2017, pp. 442–447.

[129] D. Liu, X. Cao, C. Huang, and L. Ji, "Intelligent agriculture greenhouse environment monitoring system based on IOT technology," in *Proceedings - 2015 International Conference on Intelligent Transportation, Big Data and Smart City, ICITBS 2015*, 2016, pp. 487–490.

[130] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, "A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming," *IEEE Access*, vol. 7, pp. 156 237–156 271, 2019.

[131] R. Dagar, S. Som, and S. K. Khatri, "Smart Farming - IoT in Agriculture," in *Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018*, no. Icirca. IEEE, 2018, pp. 1052–1056.

[132] M. A. Zamora-Izquierdo, J. Santa, J. A. Martínez, V. Martínez, and A. F. Skarmeta, "Smart farming IoT platform based on edge and cloud computing," *Biosystems Engineering*, vol. 177, pp. 4–17, 2019.

[133] A. A. Zaidan, B. B. Zaidan, M. Y. Qahtan, O. S. Albahri, A. S. Albahri, M. Alaa, F. M. Jumaah, M. Talal, K. L. Tan, W. L. Shir, and C. K. Lim, "A survey on communication components for IoT-based technologies in smart homes," *Telecommunication Systems*, vol. 69, no. 1, pp. 1–25, 2018.

[134] M. Al-Kuwari, A. Ramadan, Y. Ismael, L. Al-Sughair, A. Gastli, and M. Benammar, "Smart-home automation using IoT-based sensing and monitoring platform," in *Proceedings - 2018 IEEE 12th International Conference on Compatibility, Power Electronics and Power Engineering, CPE-POWERENG 2018*. IEEE, 2018, pp. 1–6.

[135] M. M. Dhanvijay and S. C. Patil, "Internet of Things: A survey of enabling technologies in healthcare and its applications," pp. 113–131, 2019.

[136] M. M. Alam, H. Malik, M. I. Khan, T. Pardy, A. Kuusik, and Y. Le Moullec, "A survey on the roles of communication technologies in IoT-Based personalized healthcare applications," *IEEE Access*, vol. 6, pp. 36 611–36 631, 2018.

[137] J. Qi, P. Yang, G. Min, O. Amft, F. Dong, and L. Xu, "Advanced internet of things for personalised healthcare systems: A survey," pp. 132–149, 2017.

[138] S. Y. Y. Tun, S. Madanian, and F. Mirza, "Internet of things (IoT) applications for elderly care: a reflective review," *Aging Clinical and Experimental Research*, no. 0123456789, 2020.

[139] P. Gomes, N. Magaia, and N. Neves, "Industrial and artificial internet of things with augmented reality," in *Convergence of Artificial Intelligence and the Internet of Things*. Handbook Springer-Verlag, 2020, pp. 323–346.

[140] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.

[141] G. Bryksin, "VR, AR, MR: Which Reality Technology to Choose for Your Business," 2018. [Online]. Available: https://rubygarage.org/blog/difference-between-ar-vr-mr

[142] D. Amin and S. Govilkar, "Comparative Study of Augmented Reality Sdk's," *International Journal on Computational Science & Applications*, vol. 5, no. 1, pp. 11–26, 2015.

[143] R. Amadeo, "Google's ARCore brings augmented reality to millions of Android devices | Ars Technica," 2017. [Online]. Available: https://arstechnica.com/gadgets/2017/08/googles-arcore-brings-augmented-reality-to-millions-of-android-devices/

[144] Apple Inc., "Augmented Reality - ARKit 3 - Apple Developer," 2019. [Online]. Available: https://developer.apple.com/augmented-reality/arkit/

[145] F. Herpich, R. L. M. Guarese, and L. M. R. Tarouco, "A Comparative Analysis of Augmented Reality Frameworks Aimed at the Development of Educational Applications," *Creative Education*, vol. 08, no. 09, pp. 1433–1451, jul 2017.

[146] T. Domhan, "Augmented Reality on Android Smartphones," vol. 7, no. 10, p. 47, 2010.

[147] L. Visionstar Information Technology (Shanghai) Co., "Getting Started with EasyAR SDK," 2018. [Online]. Available: https://www.easyar.com/doc/EasyAR SDK/Getting Started/Getting-Started-with-EasyAR.html#what-is-easyar-sdk

[148] B. C. Zapata, *Android Studio Application Development*. Packt Publishing, 2013.

[149] Apple Developer, "Xcode," 2019. [Online]. Available: https://developer.apple.com/xcode/

[150] T. Anglin, "Approaching Mobile: Understandng the ThreeWays to Build Mobile Apps," Tech. Rep. [Online]. Available: https://www.telerik.com/docs/default-source/whitepapers/choose-right-approach-mobile-app-developmentbb581d10116543e79a9febdb187fd0a3.pdf?sfvrsn=0

[151] D. Titov, "Native vs. Cross-Platform: Which Reigns in AR App Development World?" 2019. [Online]. Available: https://invisible.toys/native-vs-cross-platform/

[152] C. Rahul Raj and Seshu Babu Tolety, "A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach," in *2012 Annual IEEE India Conference (INDICON)*. IEEE, dec 2012, pp. 625–629.

[153] Adobe, "PhoneGap Documentation," 2013. [Online]. Available: http://docs.phonegap.com/

[154] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," in *2012 16th International Conference on Intelligence in Next Generation Networks*. IEEE, oct 2012, pp. 179–186.

[155] M. Latif, Y. Lakhrissi, E. H. Nfaoui, and N. Es-Sbai, "Cross platform approach for mobile application development: A survey," in *2016 International Conference on Information Technology for Organizations Development, IT4OD 2016*. IEEE, mar 2016, pp. 1–5.

[156] JetBrains, "IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains," 2000. [Online]. Available: https://blog.jetbrains.com/idea/ https://www.jetbrains.com/idea/

[157] Viro Media Inc., "ViroReact," 2019. [Online]. Available: https://viromedia.com/viroreact

[158] Microsoft, "Overview of Visual Studio." [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019#popular-productivity-features%0Ahttps://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019

[159] Visual Studio Code, "Documentation for Visual Studio Code," 2015. [Online]. Available: https://code.visualstudio.com/docs

[160] Ionic, "Core Concepts - Ionic Documentation," 2019. [Online]. Available: https://ionicframework.com/docs/core-concepts/fundamentals

[161] G. Douglas, "Difference between MVVM and MVP | Difference Between," 2017. [Online]. Available: http://www.differencebetween.net/technology/difference-between-mvvm-and-mvp/

[162] T. Reenskaug, "The Model-View-Controller (MVC) Its Past and Present," Tech. Rep., 2003. [Online]. Available: http://heim.ifi.uio.no/ trygver/2003/javazone-jaoo/MVC_pattern.pdf

[163] G. E. Krasner and S. T. Pope, "A cookbook for using the model-view controller user interface paradigm in Smalltalk-80," *Journal of Object-Oriented Programming*, vol. 1, no. 3, pp. 26–49, 1988. [Online]. Available: https://dl.acm.org/citation.cfm?id=50757.50759

[164] T. Lou, "A comparison of Android Native App Architecture MVC, MVP and MVVM," Ph.D. dissertation, Aalto University, 2016. [Online]. Available: https://pure.tue.nl/ws/portalfiles/portal/48628529/Lou_2016.pdf

[165] Windows Dev Center, "Data binding in depth - Windows UWP applications | Microsoft Docs," 2018. [Online]. Available: https://docs.microsoft.com/en-us/windows/uwp/data-binding/data-binding-in-depth

[166] J. H. Christensen and J. H., "Using RESTful web-services and cloud computing to create next generation mobile applications," in *Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications - OOPSLA '09*. New York, New York, USA: ACM Press, 2009, p. 627.

[167] G. Mein, S. Pal, G. Dhondu, T. K. Anand, A. Stojanovic, M. Al-Ghosein, and P. M. Oeuvray, "Simple object access protocol," nov 1997. [Online]. Available: https://patents.google.com/patent/US6457066B1/en

[168] K. S. Wagh, K. Wagh, and R. Thool, "A Comparative study of SOAP vs REST web services provisioning techniques for mobile host Person Re-Identification View project MANET topology View project A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host," *Journal of Information Engineering and Applications*, vol. 2, no. 5, 2012.

[169] S. Rautmare and D. M. Bhalerao, "MySQL and NoSQL database comparison for IoT application," in *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*. IEEE, oct 2016, pp. 235–238.

[170] N. Leavitt, "Will NoSQL Databases Live Up to Their Promise?" *Computer*, vol. 43, no. 2, pp. 12–14, feb 2010.

[171] R. Cattell and Rick, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Record*, vol. 39, no. 4, p. 12, may 2011.

[172] A. Nayak, A. Poriya, and D. Poojary, "Type of NOSQL Databases and its Comparison with Relational Databases," *International Journal of Applied Information Systems (IJAIS)*, vol. 5, no. 4, 2013.

[173] A. M. de Souza, E. P. V. Prado, V. Sun, and M. Fantinato, "Critérios para Seleção de SGBD NoSQL: o Ponto de Vista de Especialistas com base na Literatura," *Anais do Simpósio Brasileiro de Sistemas de Informação (SBSI)*, pp. 149–160, may 2014.

[174] J. S. van der Veen, B. van der Waaij, and R. J. Meijer, "Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual," in *2012 IEEE Fifth International Conference on Cloud Computing*. IEEE, jun 2012, pp. 431–438.

[175] A. Choudury, "Software Development Life Cycle: agile vs waterfall," Singapore, 2013.

[176] J. Highsmith and A. Cockburn, "Agile software development: the business of innovation," *Computer*, vol. 34, no. 9, pp. 120–127, 2001.

[177] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for Agile Software Development Twelve Principles of Agile Software," Tech. Rep., 2001. [Online]. Available: http://www.agilemanifesto.org

[178] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, vol. 50, no. 9-10, pp. 833–859, aug 2008.

[179] R. Kumar, P. Maheshwary, and T. Malche, "Inside Agile Family Software Development Methodologies," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 6, pp. 650–660, 2019.

[180] A. Taivalsaari and T. Mikkonen, "On the development of IoT systems," in *2018 3rd International Conference on Fog and Mobile Edge Computing, FMEC 2018*. Institute of Electrical and Electronics Engineers Inc., may 2018, pp. 13–19.

[181] P. Gomes, N. Magaia, and N. Neves, "Industrial and artificial internet of things with augmented reality," in *Internet of Things*. Springer, 2020, pp. 323–346. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-44907-0_13

[182] N. Berdy, R. Shahan, and D. Betts, "Introduction to Azure IoT Hub | Microsoft Docs," 2018. [Online]. Available: https://docs.microsoft.com/en-gb/azure/iot-hub/about-iot-hub

[183] N. Ramachandran and V. Perumal, "Delay-aware heterogeneous cluster-based data acquisition in Internet of Things," *Computers and Electrical Engineering*, vol. 65, pp. 44–58, jan 2018.

[184] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal Placement of Cloudlets for Access Delay Minimization in SDN-Based Internet of Things Networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1334–1344, apr 2018.

[185] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," *ACM Computing Surveys*, vol. 51, no. 6, pp. 1–29, jan 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3292674

[186] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice , Second Edition*, 3rd ed. Addison-Wesley, 2003.

[187] J. Yli-Huumo, A. Maglyas, and K. Smolander, "How do software development teams manage technical debt? – An empirical study," *Journal of Systems and Software*, vol. 120, pp. 195–218, 2016.

[188] N. Rios, M. G. de Mendonça Neto, and R. O. Spínola, "A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners," *Information and Software Technology*, vol. 102, no. May, pp. 117–145, 2018.

[189] S. Ickin, K. Petersen, and J. Gonzalez-Huerta, "Why do users install and delete apps? A survey study," in *Lecture Notes in Business Information Processing*, vol. 304, 2017, pp. 186–191.

[190] P. M. Jacob and P. Mani, "Software architecture pattern selection model for internet of things based systems," *IET Software*, vol. 12, no. 5, pp. 390–396, oct 2018.

[191] Y. B. Zikria, H. Yu, M. K. Afzal, M. H. Rehmani, and O. Hahm, "Internet of Things (IoT): Operating System, Applications and Protocols Design, and Validation Techniques," *Future Generation Computer Systems*, vol. 88, pp. 699–706, nov 2018.

[192] T. Luo, J. Huang, S. S. Kanhere, J. Zhang, and S. K. Das, "Improving IoT data quality in mobile crowd sensing: A cross validation approach," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5651–5664, jun 2019.

[193] European Commission, "Data protection." [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection_en

[194] E. Adi, A. Anwar, Z. Baig, and S. Zeadally, "Machine learning and data analytics for the IoT," *Neural Computing and Applications*, vol. 32, no. 20, pp. 16 205–16 233, oct 2020.

[195] E. P. Júnior, I. L. D. Delmiro, N. Magaia, F. M. Maia, M. M. Hassan, V. H. C. Albuquerque, and G. Fortino, "Intelligent Sensory Pen for Aiding in the Diagnosis of Parkinson's Disease from Dynamic Handwriting Analysis," *Sensors*, vol. 20, no. 20, p. 5840, oct 2020.

[196] O. Irshad, M. U. G. Khan, R. Iqbal, S. Basheer, and A. K. Bashir, "Performance optimization of IoT based biological systems using deep learning," *Computer Communications*, vol. 155, pp. 24–31, apr 2020.

[197] N. N. Srinidhi, S. M. Dilip Kumar, and K. R. Venugopal, "Network optimizations in the Internet of Things: A review," *Engineering Science and Technology, an International Journal*, vol. 22, no. 1, pp. 1–21, feb 2019.

[198] F. Ghorbani, M. Kia, M. Delrobaei, and Q. Rahman, "Evaluating the Possibility of Integrating Augmented Reality and Internet of Things Technologies to Help Patients with Alzheimer's Disease," in *2019 26th National and 4th International Iranian Conference on Biomedical Engineering, ICBME 2019*. Institute of Electrical and Electronics Engineers Inc., nov 2019, pp. 139–144.

[199] S. Paul, S. Hamad, and S. Khalid, "The Role of AR/VR in an IoT connected digital enterprise for smart education," in *ITT 2019 - Information Technology Trends: Emerging Technologies Blockchain and IoT*. Institute of Electrical and Electronics Engineers Inc., nov 2019, pp. 305–308.