Low code NETIoT for IoT Application

Shrabanti Saha Rimi (1377509)
Department of Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
shrabanti.rimi@stud.fra-uas.de

Bhargav Vinubhai Anghan (1387230)
Department of Computer Science and Engineering)
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
bhargav.anghan@stud.fra-uas.de

***Abstract*—Innovative techniques for IoT from hardware to intelligent applications, are needed to enable mainstream adoption of the Internet of Things. There are lots of ways for collecting data and supporting a unique set of IoT services in the field of IoT platforms.The responsibility for designing the application that uses the acquired data is provided to an external developer who works with the IoT platform's APIs.In this paper we present a possible approach, where the platform is built to serve applications that depend on a certain set of data from low-code IoT devices. We support modular applications that integrate a lot of features, IoT data gathering through queuing, processing, and integrated data visualization all are included using NETIoT low-code approach.**

***Index Terms*—Internet of Things, hpaPaas, monitoring, NETIoT Low code, Air pollution control system**

I. Introduction

The Internet of Things (IoT) has as of late arisen as a significant part in the development of a future civilization [12]. The rapid evolution of communication techniques in the low power wide-area network (LPWAN) arena, together with advancements on the path to 5G communications, provide a the huge push toward IoT deployment [13]. A flood of projects appeared, and a bevy of platforms emerged each targeting a different facet of the solution development process.Smart devices like IoT devices and sensors, for example, can assist each other and other objects. Smart systems that are connected to the internet of things can help with the design of large and small systems [1]. IoT can be connected to Smart Cities to provide a range of benefits including incredible homes can be built by utilizing the Internet of Things, which can automate and govern various aspects of our homes, such as doors, lighting, windows, distributed entertainment, freezers, and irrigation systems [14].

A new low-code or no-code application development trend has arisen, seeking to bridge the gap between technology and field professionals by allowing line-of-business employees to create their own cloud applications without writing code or relying on programmers [13]. Gartner created the term ”high productivity application platform as a service” (hpaPaaS) [13],and has been publishing annual Magic Quadrants on the topic since.Platform-as-a-Service (PaaS) providers for the Internet of Things ensures that all data collected by sensors or other similar devices are received and delivered to other services in a highly available, scalable, and secure manner, where it can be stored, examined, analyzed, and used to provide a response for other devices [7].

We can use particulars platforms in IoT to manage the system with other effects and also it makes whole IoT system more reliable and usable for everyone. Its development is a time-consuming and precious process that requires a high position of technological knowledge [8]. This paper promotes the use of Low-code platforms like NETIoT (low-code platforms or framework) to accelerate the development of IoT operations. To a high level of user experience, the platform features an intertwined configurable dashboard. This element can be used to offer drag and drop capability to both internet and mobile apps. In this case, it will allow the rearrangement of dashboard cards to epitomize dashboards. In order to assess the established IoT platform, an IoT device prototype was created using some modules, as well as a set of detectors and selectors that link to the IoT platform using the Message Queuing Telemetry Transport(MQTT) protocol. MQTT clients are very small and require minimal resources so can be used on small micro controllers. MQTT takes into consideration informing between devices to the cloud and the cloud to the devices. This makes for simple telecom messages to gatherings of things.MQTT can scale to interface with a large number of IoT devices. Reliability of message delivery is important for many IoT use cases. This implies that the source and beneficiary have no immediate association.The information sources report their information through distribution and all beneficiaries with interest in specific messages get the information conveyed in light of the fact that they have enlisted as supporters.

The low-code application framework is cost-effective. Low-code solutions use fewer engineering resources, which reduces costs [13]. A variety of developer abilities can also be put to use in a variety of ways, resulting in more code generation. Low-code development systems offer a large number of simple modules [2]. Companies may be able to discover reduced workers who can create low-code solutions with no previous experience. By empowering line-of-business staff to fabricate their own cloud applications without composing code or depending on developers, an extraordinary failure code application improvement pattern has arisen that means overcoming any barrier among innovation and field experts. Gartner made the expression ”High-productivity application platform-as-a-service” (hpaPaaS) [11] and has been distributing yearly Magic Quadrants on the point since. Stage as-a-administration ”platform-as-a-service” (PaaS) suppliers for the Internet of Things guarantee that all information gathered by sensors or other comparative gadgets is gotten and conveyed to different administrations in a profoundly accessible, versatile, and secure way, where it very well may be put away, inspected, dissected, and used to give a reaction to different gadgets [7].

NETIoT’s essential targets are to ensure high accessibility, adaptability, and future turn of events. A microservices-based design was utilized to execute the NETIoT worldview to fulfill these goals. Each microservice can be freely evolved from others and each part has its own data set, so we can coordinate a number of sensors into one application utilizing microservices. Microservices are not difficult to make and send rapidly. A microservices-based design ensures the framework is high accessibility furthermore, adaptability. Equipment parts that are now viable with our framework and are easy to arrange and introduce in the fitting situation to deal with client necessities (like sensors and gadget passages). The latest sensors available are viable with NETIoT, we essentially register them inside the connection point, and information starts to flood in right away. Also, it is made to scale for organizations, everything being equal, so we do not need to stress over keeping up with our own advanced framework. Make proactive and insightful choices in view of utilizing all of the business information by utilizing With NETIoT, we have admittance to a large number of instruments that can promptly get to the information rolling in from your sensors [14]. We can get applications that explicitly fit your business from our advanced commercial center, for example, those that robotize accuracy ranch activities or oversee lodgings with many rooms.

In this paper, we will examine low-code approaches for delivering data from sensors to the NETIoT system without involving platform configuration or sensor interaction and also how measurements from IoT different sensors to the NETIoT network in a low-code way, with no sensor intervention and no platform configuration effort. The presentation will include design considerations, a novel set of approaches for device management and decoding operations, as well as a detailed assessment and comparison of various possibilities. Scalable, adaptable, fault-tolerant, and high availability describe the NETIoT framework.

II. NETIOT PLATFORM

NETIoT is an IoT framework built on hpaPaaS ideas and targeted at the applications provided by a users IoT device collection [13]. Its primary goal is to make sure high reliability, scalability, and development in the future. It includes everything from device management to application areas in the development of IoT solutions [13]. Figure 1 shows how the user can make use of his hardware devices by running programs that are suitable for his range design on the run. From a technical approach, the platform was developed using a microservices-based architecture. Its goal is to make sure high reliability, scalability, fault tolerance, and development in the future [13].

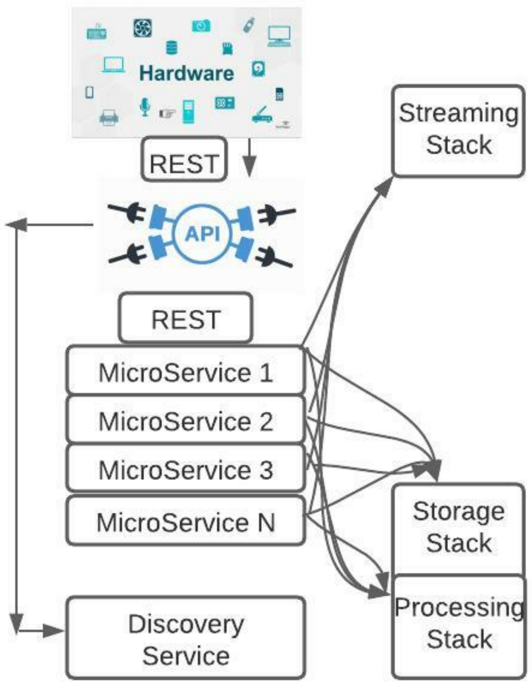

Fig. 1. NETIoT architecture.

We propose and create NETIoT, a unique IoT platform based on hpaPaaS principles, to resolve the actual drawbacks of IoT PaaS providers [7] . The main goals are to help people with the setup of IoT solutions and to provide tools for fast configuration, installation, and implementation of IoT projects in the cloud. The NETIoT model is based on microservices, which has the benefit of allowing each component to be built individually [7]. Microservices are generally faster and less expensive to develop than traditional services; each the component has its own database; microservices are quick to build and deploy, depending on service requirements; and a the microservices-based structure provides high availability and scalability [8].

Air pollution tracking (data about environmental damage at an urban scale is gathered from mobile sensors), constructing tracking (information with compassion readings is collected from detectors inside a tower) [7], and smart agriculture (showing a case study for tracking IoT data for crafts and currently reside stocks) are all utilization situations for which NETIoT will provide services [7].

NETIoT has two main components: The ability to add a new system for new uses and the decision to allow clients to develop their customized services to better suit their needs [4].

NETIoT offers different services for multiple use-cases each end-user employee is assigned a unique ID in the NETIoT system [4]; all data sent by hardware devices is collected via MQTT protocol is known as device gateway and data is sent to the cloud gateway. Each end-user must build a list of devices for the NETIoT platform to operate as well as choose which services they want to use and which sensors will provide all of the data needed for each service [4]. Communication is another scalable aspect of this system. The HTTP and MQTT interfaces are being used by NETIoT. MQTT is used to send data from sensors to device gateways, whereas HTTPS is used to communicate from device gateways to our service. [5].

After data is received and saved in a database, it must be handled and analyzed. The two types of operations that NETIoT can manage are bulk and actual [6]. Batch production could be used to analyze all collected various types of sensors over a specific period of time and predict a potential outcome evolution, whereas real-time analysis could be used to identify key situations. NETIoT will use Apache Spark [6], and the overall algorithm for huge data analytics that could be used in batch or actual. Sparks main benefit is the ability to conduct all analysis in memory, as well as support for complex analytics and real-time streaming analytics using only a variety of languages like Java, Scala, and Python [6]. NETIoT can hold a lot of data, the storage layer provides flexibility, quickness, and reliability [3]. Apache Cassandra is the best system on the market for these demands because it was designed to scale essentially linear, data duplication dynamically in one or more computer servers, and handle backup events [3] .

DETAILED ARCHITECTURE of NETIoT: It enables the integration of limitless various sensors and even the virtualization and evaluation of the results they provide. Data will be collected using LoRa-enabled devices or mobile devices with their very own sensors, such as cellphones, in the first version of the platform [5]. Clients would only need to connect their sensors with the platform and choose which surveillance application they would like to use [5]. The remainder of this section provides a comprehensive overview of the Platform, including an analysis of each component.

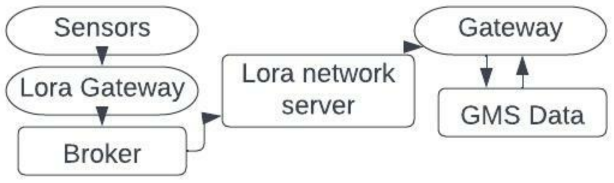

Fig. 2. NETIoT hardware architecture

The only connection channel in the system is the device gateways duty to deliver messages to the API gateway [7]. All hardware components either the LoRa network server or smartphones directly refer to it, as shown in Fig. 1. The API gateway provides a bridge between the ”outer world” (data collecting) and the software [7]. User ID and password of costumer received by API gateway after API gateway sends the data to AUTH microservices and store the user data in the storage stack. Also, all sensor device data is stored in a storage stack.

The sensors that gather data and transmit it to the NETIoT platform are called computer hardware. NETIoT supports the following devices in its initial version [7]. Standalone sensors can transmit information to the system without the help of a third party. Current NETIoT sensors can monitor air quality, buildings, sound, temperature, and humidity, among other things. They use the standard LoRa template , which requires minimum knowledge of the header and the sensor values, to send data to a LoRa gateway (Fig. 2). [7].

Using an MQTT protocol the LoRa device gateway collects information from sensors and sends it to the LoRa server computer [7]. It also has the ability to deliver conversation info back. RabbitMQ has been used by NETIoT for this feature. Finally, the LoRa computer system collects input from the LoRa gateway verifies for redundant data items and then utilizes REST to transmit the remainder of information to the database gateway . It is important to note that the central server is not doing any extra data evaluation all following actions are conducted by the API gateway and microservices.

The other device will not be capable of communicating data directly to a central via LoRaWAN so that data would be sent to smartphones which will act as collecting before getting sent to the system [7]. Sensors integrated within mobile devices (such as GPS and barometers) likewise, fit into this group because the information they collect is available directly to the smartphone. Because the file type for standalone sensors is the same, everything that enters the system is in the same format and is small. Data from non-standalone devices is acquired using a smartphone app that works on both Android and iOS. Before using NETIoT for data gathering, users must generate a virtual DevEUI for each device. Customers will be able to use the smartphone apps to register, choose real DevEUIs for which they want to submit information, locate sensors, read or see data from sensors, and send data collected from sensors to the NETIoT system [7].

Every part of this system is a separate microservice that executes in a Docker container. consider a simple temperature measurement program [11]. It will include among other things, a display system, a data management component (which reads the temperature from the database and sends an alert when a certain threshold is reached), and a notification module that is SMS-based [11]. Apps make regular requests to the storage stack in addition to reading data from the sensors or other programs so they are stateless. The memory stack validates all of the devices in the software and delivers the information requested, which would be processed using Socket Buffer [11].

Storage, compute, and streaming are the three NETIoT layers, as previously indicated. A number of databases are included in the storage stack for storing data ranging from sensor data to application data [5]. Its features include serializing, getting data for applications, and other operations. Sensor data is saved in a Cassandra database, whereas other data (app data, user data, and so on) is stored in a PostgreSQL database [5]. The processing stack comprises tools that make data processing easier, such as Apache Spark, while the streaming stack includes tools that aid with continuous data [7].

III. USE CASE

Air pollution is the greatest well-being risk in Europe and its impacts are significant. Cardiovascular sickness and stroke are the most well-known reason for unexpected losses owing to ”the air contamination and are liable for 80 percent of the cases” [9]. In expansion to causing unexpected passing, air contamination increments the frequency of a wide scope of illnesses (e.g., respiratory also, cardiovascular illness and malignant growth), with both long haul furthermore, transient consequences for well-being, including underneath laid out levels by the World Health Organization [10]. Different reports show that air contamination has additionally been related to influences on richness, pregnancy, youngsters, and kids [7].

Through NETIoT, we can make the execution of such air contamination checking application is achievable and very without any problem. For our situation, we can incorporate one contamination sensor (static sensor) close to a client is home, and a contamination sensor to be associated by Bluetooth to the telephone for correspondence purposes (portable sensor), as portrayed in Fig. 3. With such an application, air contamination sensors gather the data which is additionally utilized for different administrations [7]. For instance, the client runs an application on his cell phone that alarms at whatever point he strolls into a region where some air contamination the part is checked to pass air quality limits or the The ongoing normal is higher contrasted with the qualities saw for the similar regions the prior days. Besides, the client can be informed when the versatile air contamination sensor drains its battery (to be able to change or supplant the battery) [7]. In this model, we expect that the client needs to get an alarm (through an SMS on his telephone) at whatever point a certain contamination limit is reached (at one or the other sensor) [7].

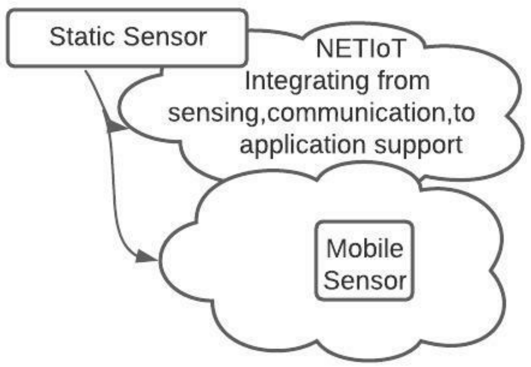

Fig. 3. The conception of an intertwined IoT platform to support the illustration of air pollution monitoring operation.

To be enlisted in NETIoT, the client gets to the API passage, where first he enrolls as a legitimate client. Then, he registers the DevEUI of the gadget he needs to add and maps it to one of the current applications. A comparable rationale applies while enlisting the versatile sensor in NETIoT, with the distinction that we set the air contamination sensors along with a battery sensor [7]. The client is given the arrangement of found applications that can work with the sort and number of characterized sensors. Allow us to accept the client decides to connect his sensors with the ”Air contamination for my home” and ”Battery alert” applications. The settings boundaries are again sensor-arranged (e.g., battery limit or the rate edge above which to create an alarm) and per-application [7].On the portable application, the client logs with something very similar the record used to enroll in NETIoT is API entryway.

Moreover, the application permits the representation of the detected qualities.When the sensors begin gathering information, they send them to the gadget entryway, then they arrive at the API door, which then, at that point, stores them in the capacity stack, solely after checking on the off chance that the DevEUI of the information is enlisted in the framework and related to an application [7].

Presently the API entryway sends a solicitation to the application to get data on the chosen boundaries. The outcome will be a rundown of boundaries that are sensor-based (e.g., the client needs to choose which is a proper sensor) or application-based (e.g., the telephone number where cautions are to be sent). NETIoT further creates an extraordinary application ID, to which the two sensors will be related (together with the metadata, for example, the cautioning telephone number). For the ”Battery alert” application, just the portable sensor is related [7]. The settings boundaries are again sensor-arranged (e.g., battery limit or the rate edge above which to create an alarm) and per-application. On the portable application, the client logs with something very similar account used to enroll in NETIoT is API passage. In the settings menu, the client has a rundown of DevEUIs created for his record. To each such DevEUI, he can relate the equipment address of the Bluetooth sensors [7]. The thought is to at the same time utilize various Bluetooth-based sensors, so the application will examine for Bluetooth gadgets and rundown all the accessible ones. The detected information is coded in the LoRa design (because of the restricted payload unique mark) and sent straightforwardly to the primary NETIoT entryway [7]. Moreover, the application permits the representation of the detected qualities. When the sensors begin gathering information, they send them to the gadget passage, then they arrive at the API entryway, which then, at that point, stores them in the capacity stack, solely after confirming assuming the DevEUI of the information is enlisted in the framework and related to an application [7]. If so, the information are decoded in the JSON design introduced in Sect. IV and are saved in the data set, both in a crude and in a JSON design, along with when the API passage got the information [7].

To give a model, the air contamination application characterizes the accompanying gadgets: a chart looking at least two air contamination boundaries (particles being checked), a heatmap powerfully built-in view of the qualities being caught by the versatile sensors, and a table with cautions for passing characterized cautioning edges for the checked boundaries [7]. Additionally, for the battery ready application, we characterize two gadgets: a measure with the last observed battery level, and a table with alarms being recognized for conveyed battery levels. On the backend, an application is made out of [7]:

1) ”A communication module with the I/O server, defining all read/write functionality with potential helper functions (like coding/decoding the dataset);” [7].

2) ”The Web administration modules (with capacities for getParametersInfo() or get visualize());” [7].

3) ”The module for calling advanced computations (like computing the average of values for a period)” [7].

IV. Conclusion

In colorful situations in the IoT platform terrain, there are enhanced styles for acquiring data and allowing specific sets of IoT services.IN this paper We demonstrated NETIoT, a new IoT platform that seamlessly connects detectors with operations, bridging the customer-technology peak and meeting the requirements of coming- generation IoT- enabled systems. Because of its microservices-grounded design, NETIoT promises advanced vacuity, scalability, and ease of erecting new operations from being detectors to contending systems. Toward the finish of this proposition, it is feasible to say that the principal goals were accomplished. An IoT stage was fabricated utilizing a low-code stage, exhibiting the flexibility of these sorts of stages and their promising future. One of the objectives was to demonstrate the way that the particular engineering would be able, notwithstanding the utilization of a low-code improvement stage speeds up the improvements to an extraordinary degree. In spite of the fact that estimating improvements consistently, the portable application was unrealistic, was underlying seven days, while the web application required roughly two months. Not just the reuse of code because of the design sped up improvements yet additionally, in In our perspective, creating utilizing low-code stages is unequivocally quicker looked at with more ”customary” approaches.

Acknowledgment

This paper is prepared under the guidance of Prof. Dr. Oliver Hahm during the study for the master’s degree in High Integrity Systems for the subject of Internet of Things (IoT). Also grateful to Prof. Dr.Oliver Hahm for his constant motivation and guidance.

References

[1] González García, C., Meana-Llorián, D., Pelayo, G., Bustelo, C. and Cueva-Lovelle, J.M., 2017. A review about smart objects, sensors, and actuators.

[2] Gartner. (2018) Magic quadrant for enterprise high-productivity application platform as a service. [Online]. Available: https://www. gartner.com/doc/3872957 https://www.gartner.com/en/documents/3956079/magic-quadrantfor-enterprise-low-code-application-platf.

[3] Betts, D., Dominguez, J., Melnik, G., Simonazzi, F. and Subramanian, M., 2013. Exploring CQRS and Event Sourcing: A journey into high scalability, availability, and maintainability with Windows Azure.

[4] Merkel, D., 2014. Docker: lightweight linux containers for consistent development and deployment. Linux journal, 2014(239), p.2.

[5] Hunkeler, U., Truong, H.L. and Stanford-Clark, A., 2008, January. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE’08) (pp. 791-798). IEEE.

[6] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D.B., Amde, M., Owen, S. and Xin, D., 2016. Mllib: Machine learning in apache spark. The Journal of Machine Learning Research, 17(1), pp.1235-1241.

[7] Rogojanu, T., Ghita, M., Stanciu, V., Ciobanu, R.I., Marin, R.C., Pop, F. and Dobre, C., 2018, June. Netiot: A versatile iot platform integrating sensors and applications. In 2018 Global Internet of Things Summit (GIoTS) (pp. 1-6). IEEE.

[8] Lopes, J.P.D., 2021. A Customizable IoT Platform Developed Using Low-Code (Doctoral dissertation).

[9] Sims, C., Hull, C., Stark, E. and Barbour, R., 2015. Key learnings from ten years of monitoring and management interventions at the Bluff Point and Studland Bay Wind Farms: results of a review. Wind and Wildlife, pp.125-144.

[10] Kumar, P., Skouloudis, A.N., Bell, M., Viana, M., Carotta, M.C., Biskos, G. and Morawska, L., 2016. Real-time sensors for indoor air monitoring and challenges ahead in deploying them to urban buildings. Science of the Total Environment, 560, pp.150-159.

[11] Varda, K., 2008. Protocol buffers: Google’s data interchange format. Google Open Source Blog, Available at least as early as Jul, 72, p.23.

[12] Ardito, C., Buono, P., Desolda, G. and Matera, M., 2017, July. Empowering CH experts to produce IoT-enhanced visits. In Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization (pp. 327-328).

[13] Pantelimon, S.G., Rogojanu, T., Braileanu, A., Stanciu, V.D. and Dobre, C., 2019, April. Towards a seamless integration of iot devices with iot platforms using a low-code approach. In 2019 IEEE 5th World Forum on Internet of Things (WF-IoT) (pp. 566-571). IEEE.

[14] Belhumeur, P.N., Hespanha, J.P. and Kriegman, D.J., 1997. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on pattern analysis and machine intelligence, 19(7), pp.711-720.