Multi-Tenancy in Smart City Platforms

Ioannis Nikolaou
PhD Candidate
University of Thessaly
Department of Business Administration
Larissa, Greece
ionikolaou@uth.gr

Leonidas Anthopoulos
Professor
University of Thessaly
Department of Business Administration
Larissa, Greece
lanthopo@uth.gr

ABSTRACT

Multi-tenancy emerged as a software architecture pattern in an effort to optimize the use of compute resources and minimize the operational cost of large scale deployments. Its applicability, however, needs to take into account each particular context as the challenges of this software architecture pattern may not make it an optimal choice in every situation. A Smart City Platform is by definition a type of software that is also expected to be deployed at a large scale. The applicability of the multi-tenancy architecture pattern in this context is debatable as the benefits it brings may not outweigh the challenges.

CCS CONCEPTS

• **Software and its engineering → Software design tradeoffs**; **Layered systems**.

KEYWORDS

multi-tenancy, smart-cities, smart-city platform, software architecture

ACM Reference Format:
Ioannis Nikolaou and Leonidas Anthopoulos. 2022. Multi-Tenancy in Smart City Platforms. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3487553.3524853

1 INTRODUCTION

The multi-tenancy architecture pattern has established itself in the past decades as the de-facto standard for Software-as-a-Service applications. The definition of multi-tenancy has been debated extensively and various definitions have been proposed as researchers approached it from different angles and levels of abstraction. Kabbedijk et al. [13] performed a comprehensive analysis of both academia and industry resources; they defined multi-tenancy as "... a property of a system where multiple customers, so-called tenants, transparently share the system's resources, such as services, applications, databases, or hardware, with the aim of lowering costs, while still being able to exclusively configure the system to the needs of the tenant".

However, multi-tenancy is far from a silver bullet and its applicability needs to be carefully analyzed taking into account the specifics of each particular deployment model [4]. This paper aims to contribute to the evaluation of multi-tenancy in the context of Smart Cities Platforms by performing a side by side comparison of the benefits and challenges of multi-tenant software architecture with the requirements and challenges of Smart City Platforms.

The paper is organized as follows. In the next section the emergency of multi-tenancy as an architectural pattern is presented. In Section 3 the multi-tenancy architecture benefits and challenges are discussed followed by Section 4, where the requirements for a Smart City Platform are analysed. A side-by-side comparison of multi-tenancy and Smart City Platform requirements is performed in Section 5. In Section 6 a proposal for an optimized Smart City Platform architecture is outlined. We conclude the paper with a short summary and proposals for future work.

2 THE EMERGENCE OF MULTI-TENANCY

Multi-tenancy emerged as a natural evolution of the Service Oriented Architecture (SOA) that dominated the enterprise software development in the '90s. The SOA architectural pattern favored the decomposition of monolithic applications to self-contained, loosely coupled services. The services were deployed in an application container, which facilitated their life-cycle management. The application container itself required additional compute resources, which ranged from moderate to significant depending on the additional services it provided.

2.1 The multi-instance pattern

With the introduction of early cloud computing infrastructure, the cost of running enterprise software on rented resources became increasingly inexpensive compared to running it on-premises. The Infrastructure-as-a-Service model (IaaS) provided economies of scale by using hardware level virtualization to run multiple deployment instances on the same bare metal resources thus maximizing their utilization. This gave rise to the multi-instance model which supported "several up to dozens of tenants" in each deployment [8].

The use of the hardware level virtualization alone, however, did not maximize cost benefits because each Virtual Machine (VM) consumed the resources required for the operating system and each instance of the application consumed the resources required for the application itself.

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524853

2.2 The multi-tenant pattern

The software vendors of enterprise applications soon realized that as their customers were migrating their operations to the cloud, additional savings were possible by sharing the application itself among multiple customers, or tenants. The main benefits of native multi tenant application were:

- lower costs of operations for the software provider; a large number of customers could be accommodated with only a few application instances
- higher utilization of hardware resources; multiple tenants shared the same compute resources through a single application

The native multi-tenancy deployments allowed the support of "a much larger number of tenants, usually in the hundreds or even thousands" [8]. This model was extremely attractive for software vendors of enterprise applications targeting Small to Medium sized companies (SMEs). It was a win-win proposal as the SME customers gained access to enterprise software they would otherwise not be able to afford as a dedicated deployment, whereas the vendors were renting their software to a much larger number of customers with the cost of essentially a single deployment. It should be noted that, depending on the business model of the software vendor, large customers had the option for a dedicated, single tenant deployment for which they paid a premium.

3 THE BENEFITS AND CHALLENGES OF MULTI-TENANCY

The primary goal of a multi-tenant architecture is to maximize the utilization of compute resources in order to minimize the operational costs. The main benefits of a multi-tenant architecture are [5]:

- higher utilization of hardware resources
- easier and cheaper application maintenance
- lower overall costs
- new data aggregation opportunities

The shift to native multi-tenancy however had challenges that required attention by the software vendors [5], [8]. These challenges were primarily related to the isolation among the tenants at multiple levels.

Security isolation The application needs to implement proper measures to protect each user of the platform from a malicious tenant that may initiate an attack from within the application.

Performance isolation The Quality of Service (QoS) of each tenant must be ensured and be in accordance with the agreed Service Level Agreement (SLA) which may differ among tenants.

Availability isolation The availability requirements of each tenant must be protected and any faults in the system must be controlled so that they do not propagate to all the tenants of the deployment.

Application customization Although the majority of the application's functionalities are the same for each tenant, some customization is expected which, depending on the application, may range from simple User Interface (UI) changes (e.g. logos, color scheme etc.) to customization of business process (e.g. document flows, report generation etc.)

In order to address these challenges, software vendors put additional effort in the development of their native multi-tenant application which increased both the technical and the development process complexity [4], [5].

The technical complexity increased because additional layers and safe-guards had to be implemented, tested and maintained. The development process complexity increased because the software development teams had to take into account the multi-tenancy support for every feature they designed and implemented. Even if only a subset of the development team was tasked with the multi-tenancy support as suggested by [8] or the multi-tenancy was introduced as a separate layer or group of components as suggested by [5], the fact of the matter is that the code base became significantly more complicated.

4 THE REQUIREMENTS OF A SMART CITY PLATFORM

According to ITU, the Smart City (SC) can be defined as "a city model that uses state-of-the art Information and Communication Technologies (ICT) to improve living, efficiency and competitiveness with respect to future generations" [11] or, according to ISO, "a city model that facilitates the planning, construction, management and smart services" [10]. The "smartness" of a city describes its ability to bring together all its resources, to effectively and seamlessly achieve the goals and fulfill the purposes it has set itself. The definition of a Smart City Platform according to ITU is "a computer system [...] that, under control of the city, uses information and ICT to access data sources and process them to offer urban operation and services to the city". In addition, "the SC platform is a city platform that offers direct integration of city platforms and systems, or through open interfaces between city platforms and third parties, to offer the urban operation and services supporting the functioning of city services, as well as efficiency, performance, security, and scalability" [12].

As a consequence of the above definitions, we can safely assume that a Smart City Platform must excel in the following areas:

- scalability to grow together with the Smart City's needs
- flexibility to integrate with different systems, services and data sources
- openness to expose data to third parties

A Smart City Platform must be able to integrate with sensors and actuators that make up the Internet Of Things (IoT) layer of the Smart City. These devices range widely in terms of capabilities, features and sophistication. The Smart City IoT landscape is comprised of a plethora of deployment models, communication protocols and operational practices [21], [20]. Furthermore, each Smart City has its own needs and priorities and its stakeholders will choose which areas they will focus in ranging from water supply networks to parking lots and from environmental monitoring to security surveillance systems. [22], [23].

On the other hand, a Smart City Platform must comply with state and country regulations that govern the way data are gathered, processed, stored and shared. Due to the sensitive nature of the information collected and the fact that it is frequently related to critical infrastructure (energy grid, water supply network, traffic management etc.) the requirements for security and data protection

can be very strict. In addition, the volume of the data and meta-data collected from the Smart City sensors add additional challenges in the protection of the privacy of each individual citizen [19], [6].

A Smart City Platform is expected to be operational 24/7/365 with minimum disruptions and down-time while at the same time receive regular updates to patch any security vulnerabilities and introduce new features. Furthermore, the Smart City infrastructure as a whole must operate in an environmentally friendly and sustainable way and consume as few resources as possible in order to reduce its environmental footprint [20], [1].

In conclusion, a Smart City Platform is expected to be flexible and adaptable to the particular needs of each city. The variability is very high at all levels, from the governance and regulation to the IoT devices and energy consumption, even for cities within the same region or country.

5 MULTI-TENANCY IN THE SMART CITY PLATFORM CONTEXT

In order to evaluate the applicability of a native multi-tenant architecture in a Smart City, we should examine if the benefits of the multi-tenancy can provide an edge in the Smart City context.

The main benefits of multi-tenancy as described in Section 3 are a) lower costs of operations, and b) higher hardware utilization while the assumptions for them to materialize are a) the size of each customer is too small to justify a dedicated deployment, and b) the needs of each customer are pretty much the same and only slight customizations may be required.

In the Smart City scale, both these assumptions do not hold as discussed in Section 4 because a) a Smart City Platform cannot be considered equivalent to an SME, and b) the needs for each Smart City are highly heterogeneous. In order to satisfy them, a multi-tenant platform will either be too generic to be useful or too customized to be worth the effort.

Looking at the challenges that native multi-tenancy brings as described in Section 3, these are primarily in the areas of security, performance, availability and customization.

As discussed in Section 4, these areas are of primary concern for a Smart City Platform. In fact, the software architecture of a Smart City Platform must embrace these requirements and register them among its strengths, not its weaknesses.

Finally, the additional complexity in the software development that a native multi-tenant platform requires is a significant disadvantage in the Smart City context as it will significantly hinder any customization and adaptation efforts that will be required to adapt the multi-tenant platform to the needs of each Smart City.

6 TOWARDS AN OPTIMIZED SMART CITY PLATFORM SOFTWARE ARCHITECTURE

Comparing the technology that was available during the period when multi-tenancy architecture pattern emerged with the current technological landscape, we can see that it has advanced greatly. It is now technically possible to support the development, deployment and operation of software architecture patterns that were not economically viable. The key technological advantages that make a different approach possible in the context of the Smart City Platforms are:

- containerization
- infrastructure automation
- on-demand resource allocation
- cloud native services

6.1 Containerization

The containerization technologies advanced the compute resources utilization a step further by allowing the virtualization to occur at the operating system level. A container-based deployment allows a number of deployment instances to run on the same hardware and operating system but in a completely sand-boxed environment, thus ensuring high levels of security isolation, performance and availability.

6.2 Infrastructure automation

The allocation of hardware resources and the creation of the underlying network infrastructure used to be a tedious and error-prone process. This increased the costs of designing, deploying and maintaining multiple application instances and was among the reasons the industry shifted towards the multi-tenant architecture pattern as described in Section 3.

The evolution and maturing of the cloud has given rise to tools and practices that allow the automation of the creation, update, monitoring and maintenance of completely virtual infrastructures. Tools like Ansible [3], SaltStack [18] and Terraform [9], to name just three, allow the fast, efficient and secure management of highly customized hardware and network topologies that can spread to hundreds of physical servers. In fact, it is the advancement of these technologies and practices that has given birth to a completely new domain in software engineering, the Development Operations or DevOps [15]. The DevOps practices in the Smart City context have been discussed in more detail in [17], [14].

6.3 On-demand resource provisioning

The cloud providers have invested huge amounts of capital and R&D resources to build the infrastructure required for the cloud to be considered practically infinite. In the process, the on-demand resource provisioning and usage has been developed and perfected. Now, with a few simple operations, anyone can gain access to compute resources of any size within a few minutes. More interestingly, however, the cloud has enabled us to reserve and pay for the required resources in a very granular way. It is now possible to spawn Virtual Machines in the cloud that use 1 Virtual CPU (vCPU) which corresponds to ½ of a CPU core of a multi-threaded CPU with 1GB of RAM and a few MBs of disk. These small sized VMs can make economically viable deployment models that were not possible in the past.

6.4 Cloud native services

The cloud providers, being themselves software vendors, have also taken steps to migrate their platforms to a multi-tenant model. In so doing they have implemented multi-tenancy in their core building blocks used for their applications and, furthermore, they have made them publicly available. These building blocks cover the needs of the low-level layers of practically any enterprise level application and include database, messaging middle-ware and storage. Some

examples are the Amazon Timestream [2], the Azure CosmosDB [16] and the Google Cloud Platform PubSub service [7].

The combination of these key technologies can provide the building blocks of an optimized Smart City Platform architecture.

6.5 Towards an optimized Smart City Platform software architecture

In order to optimize the architecture of a Smart City Platform, we should revisit the core assumption that it is beneficial to share the same platform among multiple tenants and instead design it with the more realistic assumption that the platform will be used by one Smart City at a time. This allows us to focus on the real needs of the Smart City and design the Platform around them. A set of principles that are important for an optimized Smart City Platform are:

- allow the execution of small scale "experiments" to allow the Smart City citizens and stakeholders to get acquainted with new technologies and services in a cost-effective way
- scale-up at the pace each city can support
- stay up-to-date with updates and best practices

These principles align better with a single-instance software architecture rather than a multi-tenant one. At the same time, these principles can be contained at the business or application layer of the platform which gives us the flexibility to select the technologies for the low-level layers independently in a cost-effective manner.

The observation above can in fact become the differentiating factor of an optimized Smart City Platform architecture. The core services and features of a Smart City Platform can be developed, deployed and operated with the assumption that they serve a single Smart City. The deployment of the infrastructure services on the other hand can be approached in a more flexible manner.

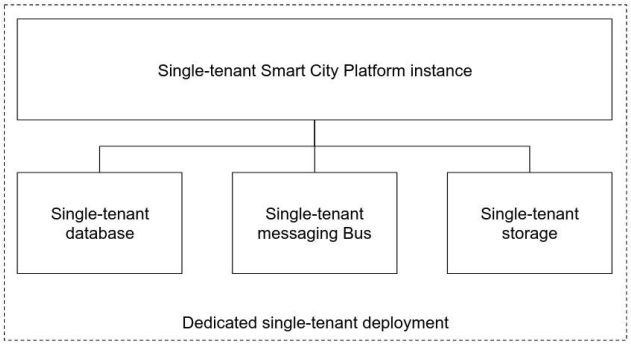

Figure 1: A hybrid Smart City Platform architecture

When the size of the Smart City Platform does not justify a full blown dedicated deployment, as for example during a pilot project, the infrastructure layer can be provided using the as-a-Service pattern by third parties in a native multi-tenant environment (Fig.1).

If, however, the Smart City's needs call for a dedicated, fully isolated environment, either because of its scale, or due to regulatory restrictions, the infrastructure services can be deployed and managed within the scope of the Smart City Platform in a pure single-tenant deployment software architecture (Fig.2).

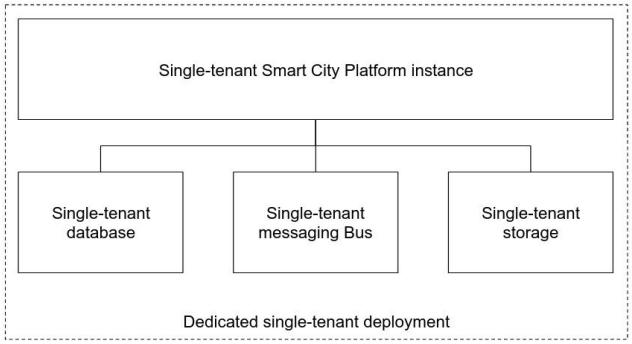

Figure 2: A dedicated Smart City Platform architecture

7 SUMMARY AND FUTURE RESEARCH

This paper presented a side-by-side comparison of the multi-tenant architecture and the requirements of a Smart City Platform. Based on the academic literature and the definitions from standardization bodies, evidence has been presented to support the idea that multi-tenancy may not be the best software architecture for a Smart City Platform. This remains to be verified in the field by an in-depth review of the state of the art in currently available Smart City Platforms. Additional research should also to be performed to evaluate the contribution of multi-tenancy to the success, or not, of these platforms in Smart City Platform project.

REFERENCES

[1] Faris. A. Almalki, S. H. Alsamhi, Radhya Sahal, Jahan Hassan, Ammar Hawbani, N. S. Rajput, Abdu Saif, Jeff Morgan, and John Breslin. 2021. Green IoT for Eco-Friendly and Sustainable Smart Cities: Future Directions and Opportunities. *Mobile Networks and Applications* (Aug 2021). https://doi.org/10.1007/s11036-021-01790-w

[2] Amazon. 2020. Amazon Timestream – Time Series Database – Amazon Web Services. https://aws.amazon.com/timestream/

[3] Ansible. 2016. Ansible is Simple IT Automation. https://www.ansible.com/

[4] Cor-Paul Bezemer and Andy Zaidman. 2010. Multi-tenant SaaS applications. *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE) on - IWPSE-EVOL '10* (2010). https://doi.org/10.1145/1862372.1862393

[5] Cor-Paul Bezemer, Andy Zaidman, Bart Platzbeecker, Toine Hurkmans, and Aad 't Hart. 2010. Enabling multi-tenancy: An industrial experience report. , 8 pages. https://doi.org/10.1109/ICSM.2010.5609735

[6] Trevor Braun, Benjamin C.M. Fung, Farkhund Iqbal, and Babar Shah. 2018. Security and privacy challenges in smart cities. *Sustainable Cities and Society* 39 (May 2018), 499–507. https://doi.org/10.1016/j.scs.2018.02.039

[7] Google. 2008. Pub/Sub for Application and Data Integration. https://cloud.google.com/pubsub/

[8] Chang Jie Guo, Wei Sun, Ying Huang, Zhi Hu Wang, and Bo Gao. 2007. A Framework for Native Multi-Tenancy Application Development and Management. *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)* (Jul 2007). https://doi.org/10.1109/cec-eee.2007.4

[9] Hashicorp. 2014. Terraform by HashiCorp. https://www.terraform.io/

[10] ISO. 2020. *Framework for integration and operation of smart community infrastructures — Part 1: Recommendations for considering opportunities and challenges from interactions in smart community infrastructures from relevant aspects through the life cycle.* Number 37155-1:2020. ISO. https://www.iso.org/standard/69241.html

[11] ITU-T. 2014. *An overview of smart sustainable cities and the role of information and communication technologies.* https://www.itu.int/en/itu-t/focusgroups/ssc/documents/approved_deliverables/tr-overview-ssc.docx

[12] ITU-T. 2018. *High-level requirements and reference framework of smart city platforms.* Number Y.4201. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.4201-201802-I!!PDF-E&type=items

[13] Jaap Kabbedijk, Cor-Paul Bezemer, Slinger Jansen, and Andy Zaidman. 2015. Defining multi-tenancy: A systematic mapping study on the academic and the

industrial perspective. *Journal of Systems and Software* 100 (Feb 2015), 139–148. https://doi.org/10.1016/j.jss.2014.10.034

[14] Hans Rüdiger Kaufmann, Dolores Bengoa, Christoph Sandbrink, Angeliki Kokkinaki, Achilles Kameas, Altheo Valentini, and Omiros Iatrellis. 2020. DevOps Competences for Smart City Administrators. *2521-3938* (Sep 2020), 213–223. https://repository.corp.at/719/

[15] Ruth W. Macarthy and Julian M. Bass. 2020. An Empirical Taxonomy of DevOps in Practice. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (Aug 2020). https://doi.org/10.1109/seaa51224.2020.00046

[16] Microsoft. 2017. Azure Cosmos DB | Microsoft Azure. https://azure.microsoft.com/en-us/services/cosmos-db/

[17] John Moore, Gerd Kortuem, Andrew Smith, Niaz Chowdhury, Jose Cavero, and Daniel Gooch. 2016. DevOps for the Urban IoT. *Proceedings of the Second International Conference on IoT in Urban Space* (May 2016). https://doi.org/10.1145/2962735.2962747

[18] Salt. 2011. Home. https://saltproject.io/

[19] M. Sen, A. Dutt, S. Agarwal, and A. Nath. 2013. Issues of Privacy and Security in the Role of Software in Smart Cities. , 518–523 pages. https://doi.org/10.1109/CSNT.2013.113

[20] Bhagya Nathali Silva, Murad Khan, and Kijun Han. 2018. Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities. *Sustainable Cities and Society* 38 (Apr 2018), 697–713. https://doi.org/10.1016/j.scs.2018.01.053

[21] Abbas Shah Syed, Daniel Sierra-Sosa, Anup Kumar, and Adel Elmaghraby. 2021. IoT in Smart Cities: A Survey of Technologies, Practices and Challenges. *Smart Cities* 4, 2 (Mar 2021), 429–475. https://doi.org/10.3390/smartcities4020024

[22] Ruben Sánchez-Corcuera, Adrián Nuñez-Marcos, Jesus Sesma-Solance, Aritz Bilbao-Jayo, Rubén Mulero, Unai Zulaika, Gorka Azkune, and Aitor Almeida. 2019. Smart cities survey: Technologies, application domains and challenges for the cities of the future. *International Journal of Distributed Sensor Networks* 15, 6 (Jun 2019), 155014771985398. https://doi.org/10.1177/1550147719853984

[23] Saber Talari, Miadreza Shafie-khah, Pierluigi Siano, Vincenzo Loia, Aurelio Tommasetti, and João Catalão. 2017. A Review of Smart Cities Based on the Internet of Things Concept. *Energies* 10, 4 (Mar 2017), 421. https://doi.org/10.3390/en10040421