Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 191 (2021) 505–510

Procedia Computer Science

www.elsevier.com/locate/procedia

International Workshop on Edge IA-IoT for Smart Agriculture (SA2IOT)
August 9-12, 2021, Leuven, Belgium

Logical Computer Vision on IOT

OQAIDI Mohammed*, AIT ABDELOUAHID Rachida, MARZAK Abdelaziz

Faculty of Science Ben M'sik, Hassan 2 University, Casablanca, Morocco

Abstract

With the Internet of Things, an infinite number of devices must generate and share information flows that describe real life. this is why the IoT software platform is very complicated and composed of several entities that depend heavily on each other to connect the tangible world of objects to the virtual world and embeds an intelligence that offers multiple possibilities, stores and analyzes the measurement taken by a sensor , in order to monitor and control connected objects or to create a history allowing prediction and these strong dependencies which complicate the development and maintenance circuit and the evolution of this software, Not all projects adopt a formally identical architecture, however it is possible to schematize an optimal architecture with weak dependencies.

Keywords: Internet of Things, Smart City, microservices architectures, IoT, IoT platform

1. Introduction

The use of connected objects is exploding, the rise of IoT is a reality, but outages affect 64% of users worldwide and computer bugs in IoT applications can cause heavy damage, which is driving us new cloud technologies. , micro services, the latter which requires a well optimized and divided IOT application in order to be able to ensure the delivery of quality applications and a good digital experience which leads us to study the dependencies between business entities and therefore provide a logical vision on the vertical and horizontal division either between two business entities and in the same entity

* Corresponding author. Tel.: Tel.: +2126-6194-4452.
E-mail address: moqaidi@gmail.com

10.1016/j.procs.2021.07.065

2. Related Works

Numerous projects have been achieved to optimize the IOT architecture and the more scalable architectures are the micro-service oriented architectures [6] [7] [8] which structures an application as a set of loosely coupled services and its usefulness is only when update a resource, only the microservices containing this resource will be updated, the entire application remaining compatible with the modification, unlike the entire application in a classic architecture

3. Our Proposition

We propose a conceptual approach of a target architecture which will correct the problem of strong dependence between business entities to be able to have a scalable and maintainable architecture.

There is a very strong dependence between the models of the different functional modules. This often leads to inconsistencies in the backup of the data in the database. Indeed, the data supposed to be managed by a module is found carried out by another functional module.

We therefore have a strong dependence between the different functional modules and on the other hand a referencing in both directions between the different layers whether it is within the same module or inter-module so there is no border between the business processes and persistent objects and objects sent to the end user, our objective is to give a conceptual approach of a target architecture which will correct the problem of strong dependence between business entities to be able to have a scalable and maintainable architecture (See Fig.1)

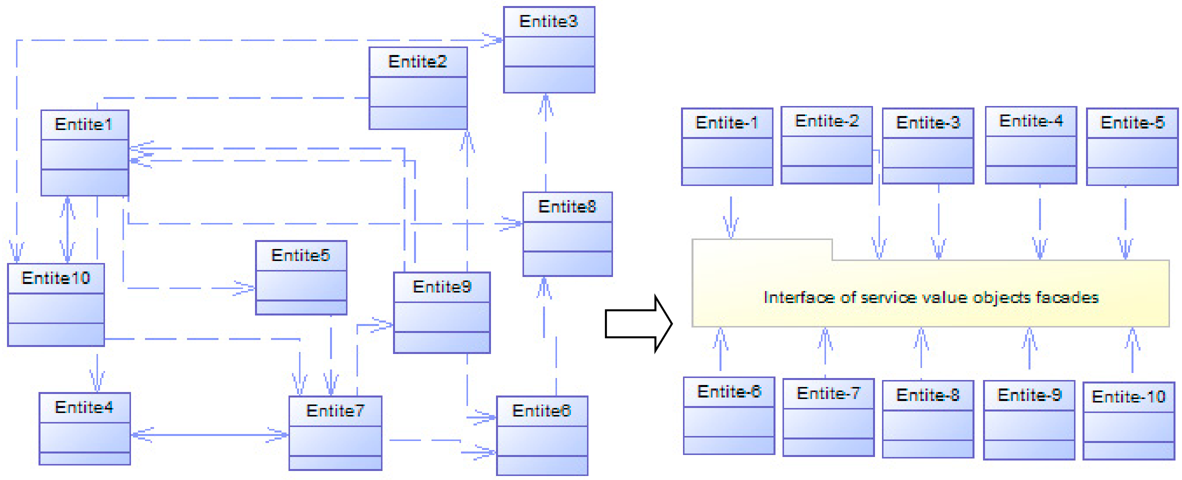

Figure 1 - Target architecture

The package diagram on the left reflects the reality of the current IOT application, a very strong coupling between the different components:

• A very strong dependence between the functional modules

• Circular dependencies between software layers, whether within the same module or between modules.

The package diagram on the right offers an alternative based on the principle of low cut.

• Separation between functional areas

• Separation between software layers

The target architecture is a layered architecture((See Fig.2)

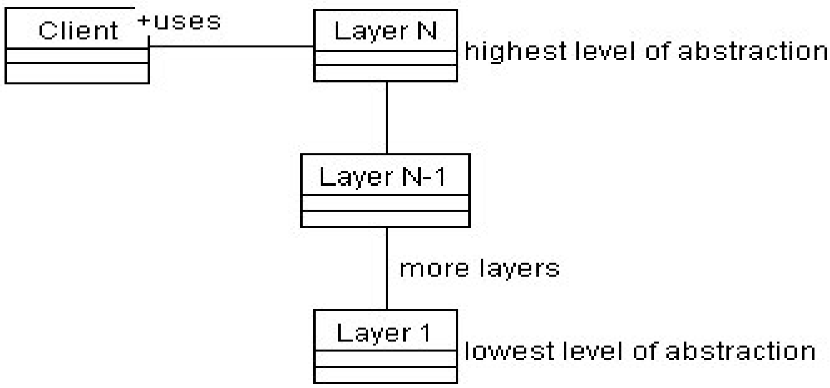

Figure 2- Layered architecture model

This model leads us to identify a set of architectural layers on which the application will be based. This separation makes it possible to clearly identify responsibility for each layer and to master the dependencies between them. Example, Persistence layer, Services, ...
The level N layer depends only on the level N-1 layer (apart from the model). Lower level layers cannot reference upper layers under any circumstances. Dependence is always top-down. We offer a classic architecture in 3 independent layers:

• Presentation
• Service
• Device (connected object)
• Persistence

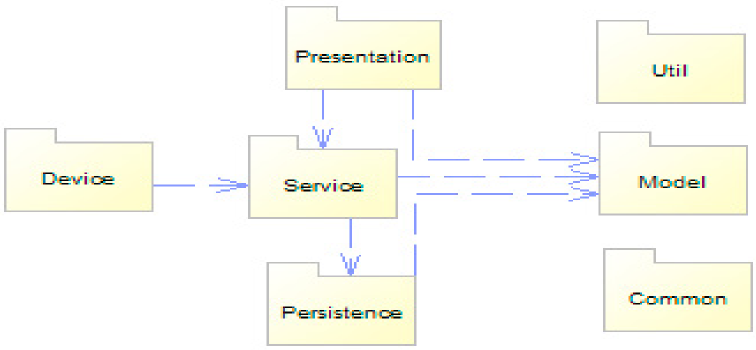

Figure 3 - Layered architecture

These layers are; made independent using interfaces, communicate using domain objects (Model), packaged separately, tested separately. The proposed breakdown for our unified architecture which will be carried out in step 2 includes the following elements:

• All service calls between modules must go through the "IServiceOffert" and "IServiceConsomme" interfaces of the modules in question.

Under no circumstances should a reference be made in a module A to another class of another module B, in particular a Use Case of module A must under no circumstances directly call another U se Case of module B.,

• The exchange objects between the services of the different modules will be done via the Result interface containing messages and / or value objects which correspond to the contract defined between the two modules.

• The breakdown of the model for each module.

The transversal modules will have the following particularity:

• They offer transversal services to all modules. These services follow the same structure as the services offered by a module.

• The classes of the repository model will not be split from the modules using the Repository classes. These will be accessed directly.

• The division of the classes of the model consists in treating the following different cases. Let's take the example of a CMA model class of module A which references a CMB model class of model B, we have two scenarios:

Table 1: The classes division of models

Case	Description	Proposed solution
Case One To One	CMA has a refCMB attribute which is of type CMB	- Replace the refCMB type with a standard type (the type of the identifier of the CMB class) - Change the mapping of the refCMB attribute
Case One To Many	CMA has a refsCMB attribute which is a collection of type CMB	-The person in charge of the refactoring of module B must perform the operations mentioned in Scenario 1 (Case One To One) when refactoring module B -Remove the refsCMB attribute from the CMA class, as well as the corresponding getter and setter -Delete the corresponding mapping in mapping file -Create the method “Collection <CMB_VO> getAll <refCMB> for <CMA> (String identifierCMA)” in the “ServiceOffert” class of the CMB module. (CMB_VO represents the value object class grouping together all the attributes of the CMB class useful for the CMA module. The CMB_VO class must be in the "services.valueobjects. <moduleB>" package).
Case Many To Many	If CMB does not have a CMA type refsCMA attribute, this means that there is TABLE_RELATION_CMA_CMB relation table which makes the link between TABLE_CMA and TABLE_CMB	-In the package of module B, create the model class "Association <CMA>" this class must contain a ref <CMB> attribute whose type is that of the identifier of the CMB class and a ref <CMA> attribute whose type is that of the identifier of the CMA class -Create in the mapping file of module B the mapping between the new class creates "Association <CMA>" and the relation table TABLE_RELATION_CMA_CMB (associate the attribute ref <CMB> with the column TABLE_RELATION_CMB_FK and the attribute ref <CMA> with the TABLE_RELATION_CMA_FK column) -Add in the CMB class a ref <CMA> attribute whose type is that of the new class "Association <CMA>" of the newly created model, -Add the corresponding mapping in the framework mapping file of module B for the CMB class to take into account the newly created ref <CMA> attribute. -Remove from the framework mapping the mapping of the refCMB attribute of the CMA class, -Remove the refCMB attribute from the CMA class -The old code, which consists of adding CMB entities in the collection refCMB attribute of the CMA class, must be modified so as to create persistent objects of type "Association <CMA>" (by setting all these

	attributes), -If at the time of these changes, dependencies have been introduced in module A to module B (when creating the "Association <CMA>" objects), in this case you must follow the same procedure described in the services offered and services consumed, -Create the method “Collection <CMB_VO> getAll <refCMB> for <CMA> (String identifierCMA)” in the “ServiceOffert” class of the CMB module. (CMB_VO represents the value object class grouping together all the attributes of the CMB class useful for the CMA module. The CMB_VO class must be in the "services.valueobjects. <moduleB>" package)

We will be based on the calculation of the level of dependence of the modules to prioritize the modules to be extracted within the framework of a progressive division. The calculation of the level of dependence of the modulus X is defined by the torque.

• Number of services offered to other modules
• Number of modules depending on the module X

A module has priority if

• It offers very few services to the other modules because the installation of its service facade is faster
• If few modules depend on it because this limits the impact on the existing one, only the modules which depend on it are updated to go through the service facade.

Phase 1 : Initialization - Phase 2 : Prioritize modules	
Inputs	List of candidate modules
Tools	Calculation of dependency level
Actions	Gradually extract the modules while limiting the impact on the existing The complexity of the module is not taken into account Only the number of services offered by one module to other modules Number of modules that depend on the module being processed Example Iteration 1 : Module 4 Iteration 2 : Module 4 + Module 2
Outputs	Prioritized list of modules(example below)

3.1. Example of dependency diagram

The following example shows a dependency diagram of 4 modules. Module 2, for example, offers 5 services in Module 1.

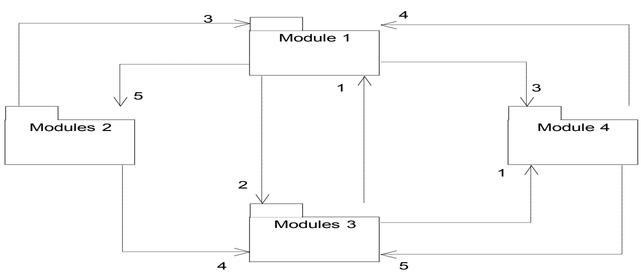

Figure 4 - dependency diagram

The following table shows a projection of the diagram above: each module in a row offers a number of services to the modules in column.

Table 2: Module Priority

	Module1	**Module2**	**Module3**	**Module4**
Module1		3	1	4
Module2	5			
Module3	2	4		5
Module4	3		1	

From the projection matrix above, we deduce the following prioritized list:

1. Module 4 (3 + 1, 2)
2. Module 2 (5, 2)
3. Module 1 (3 + 1 + 4, 3)
4. Module 3 (2 + 4 + 5, 2)

With Module X (Number of services offered by X, number of modules that depend on X to arbitrate in the event of equality in terms of the number of services offered).

The number of services offered by a module (which will appear on the facade) is at most the sum of the services offered for all the modules.

4. Conclusion and Future Work

In our future work, we will develop a generic and configurable scalable distributed architecture to integrate sensors and devices and able to manage all possible errors in a readable way to be able to monitor and control all objects.

References

[1] S. Zdonik, S. Shekhar, J. Katz, X. Wu, L. Jain, D. Padua, X. (Sherman) Shen, B. Furht, V.S.Subrahmanian, M. Hebert, K. Ikeuchi, B. Siciliano, S. Jajodia, and N. Lee, Enhanced Building Information Models Using IoT Services and Integration Patterns," Springer, 2015.

[2] S. gallagher, "Mastering Docker," Packt, Décembre 2015

[3] T. Vresk, and I. Čavrak, "Architecture of an Interoperable IoT Platform Based on Microservices," accessible en ligne à "http://docs.miproproceedings.com/cts/cts_21_3800.pdf"

[4] Pauline F, Thierry D, Ali B, Franck Gautier 'Analyse et Perspective-de l'avenir de l'Internet des Objets à l'Horizons 2013-2020,' citc, octobre 2013, accessible en ligne sur :"http://www.citceurarfid.com/uploads/CITCEuraRFID-Analyse-Perspective-d-avenir-del-Internet-des-Objets-Horizons-2013-2020.pdf"

[5] Institut Montaigne, 'Big data et objets connectés Faire de la France un champion de la révolution numérique, www.institutmontaigne.org, Avril 2015.

[6] "Cloud management platform based on micro-service architecture," CN Patent App. CN 201,510,623,507.

[7] J. Lewis, m. fowler. "Microservicesa definition of this new architectural term," "http://www.martinfowler.com/articles/microservices.html"

[8] A. Zanella, N. Bui, A. Castellani, L. Vangelista, et M. Zorzi, "Internet of Things for Smart Cities," accessible en ligne sur "http://www.dei.unipd.it/~zanella/PAPER/CR_2014/IoTSmartCity2014 _CR.pdf"

[9] S. Gopalani, et R. Arora, "Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means," International Journal of Computer Applications (0975 – 8887) Volume 113 – No. 1, March 2015, accessible en ligne sur "http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.1639&re p=rep1&type=pdf" [10] M.Zaharia, R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M.J. Franklin, A. Ghodsi, J. Gonzalez, et S. Shenker, 'Apache Spark: a unified engine for big data processing,' Ion Stoica Communications of the ACM, Vol. 59 No. 11, Pages 56-65, November 2016,.

[11] Konstantinos Vandikas, Vlasios Tsiatsis, "Performance Evaluation of an IoT Platform,"Fighth International Conference on Next Generation Mobile Applications, Services and Technologies,2014