Introduction

🏆 A2.3 Learning Activity

- Software architecture patterns.

✏️ Development

1. Consider applying the following architectural patterns to the case study.
    - ☐ Client-server architecture pattern.
    - ☐ Layered architecture pattern.
    - ☐ Architecture pattern model view controller.
    - ☐ Micro-service oriented architecture pattern.
    - ☐ Architectural pattern pipes and filters.
    - ☐ Event-based architecture pattern.
    - ☐ Repository or whiteboard architecture pattern
    - ☐ Publisher/subscriber architecture pattern.
    - ☐ Master-slave architecture pattern.
    - ☐ Point-to-point architecture pattern.
2. Take as a basis the architectural views as well as the elaborated UML diagrams, to apply the pattern that you consider the most appropriate for each of the following scenarios.

    - 2.1 Considering that it is desired to **maintain and scale the system**, it seeks to develop the application through the decomposition of small independent and isolated services, which consume an external interface to communicate to a database server.
    - 2.2 **Confidentiality and security** being attributes or requirements for the case study, it seeks to structure each of the components that are going to be programmed in groups of sub-tasks, where each of these sub-tasks must be communicated an intermediate layer and this to another upper layer.
      2.3 Looking for the **availability** of the system, it is proposed to install two service servers, where they make their requests to a third server in which the database would be stored.
      2.4 Identifying that data **integrity** is a requirement, it is proposed to replicate and synchronize the database stored within the main server to another, considering the main as master and secondary as slave.
    - 2.5 Considering that the client requested that every time a failure occurs, **alarms** should be provided to the different users regardless of where they are located, it is noted that a service provider must be hired to trigger the alarms of these reach the required users.

Architectural patterns
"Conservance"

Customer

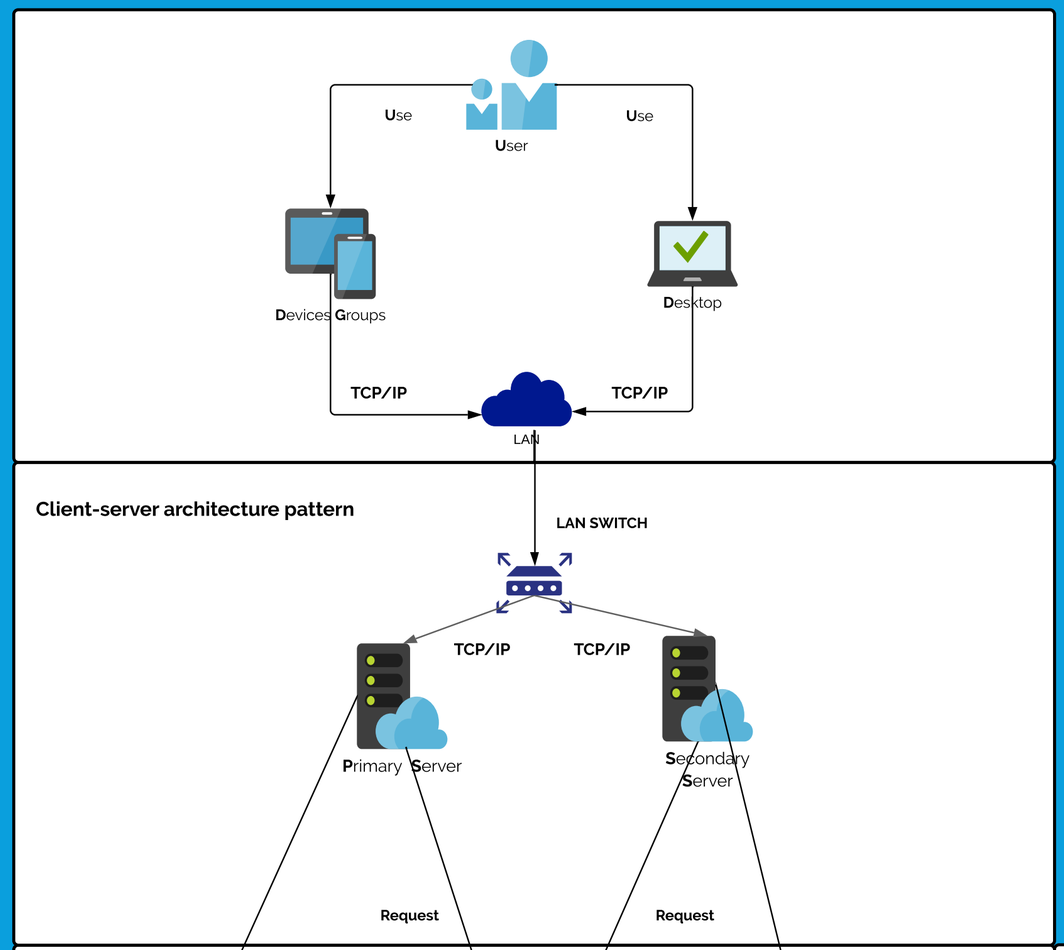
Server

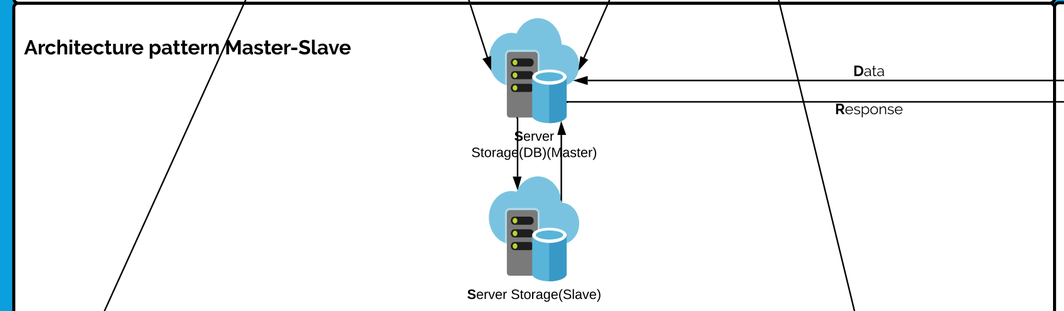
DataBase

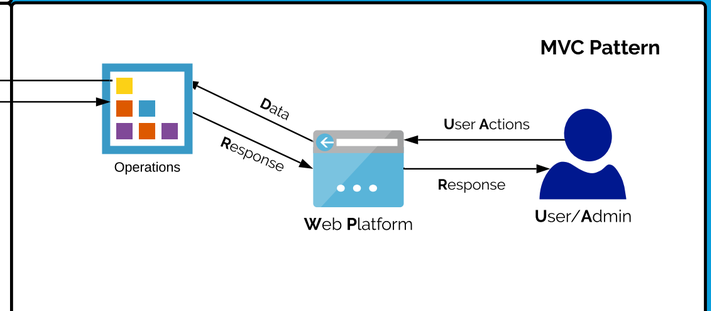
Micro-Services

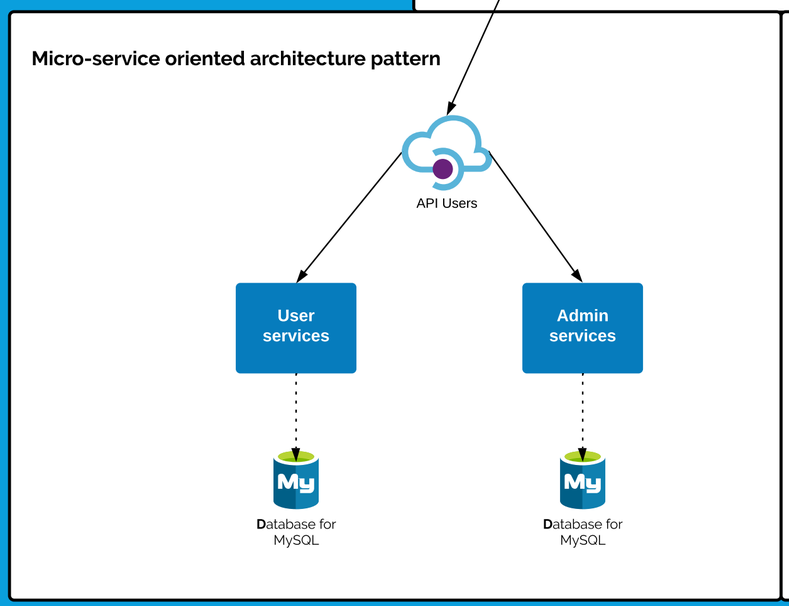
Structure

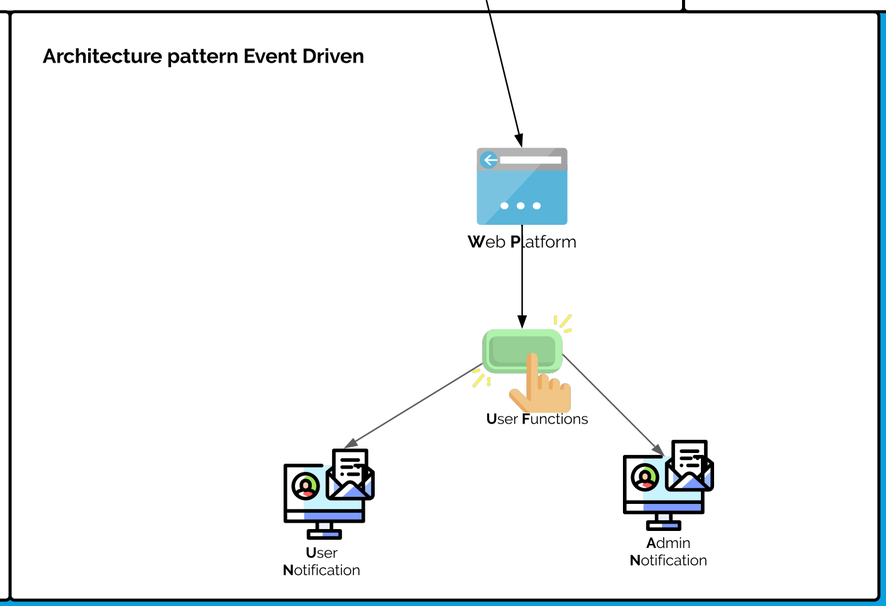
Application











Conclution by Edson:

Through the use of architecture patterns, we can identify the patterns that can be implemented in our system, in this case “conservance” and by the display of the diagrams previously made on past activities it becomes more easy since that highlight features that allow you to identify patterns, and even by reading the document that was attached in the classroom it is possible to identify each pattern (advantages and disadvantages) and thus be able to select the most appropriate version to use, to then make the implementation diagram within our system.

Conclution written by Antonio:

With the use of Architecture Design we can plan and observe how the use case to be developed is structured, the interaction of the various components(database, API's etc.) within the system can be observed, on this occasion five architectural patterns were used to perform it.Each pattern has different characteristics, advantages and disadvantages for its implementation, something in common is that in all it conflicts with performance, however we looked for those that fit directly with the system that we wanted to develop.

Conclution written by Joseph

In the structure of the case study that we had to carry out, we found the use of 5 architectural patterns feasible since the relationship that existed between the patterns that we put has a greater relationship which helped to give a better sense to the entire structure.

Most of the patterns that we put started from a server since it is responsible for sending the responses to the requests and being able to conclude with a result. In turn, possible risks that were seen if a server crashed were also feasible, at that time we will use a pattern in charge of supplanting the master server. Of the patterns that we put, the one that caught my attention was that of events since that way we could realize what is happening in the entire environment of the web application and through notifications know what is happening.

 Go to my GitHub repository