Introduction

🏆 A2.3 Learning activity

- Software architecture patterns

📘 Instructions

- Based on the software architecture styles and patterns documentation, develop the system architecture for the case study.
- All activity or challenge must be done using the **MarkDown style with .md** extension and the VSCode development environment, or you can use any platform for example **Notion**, and it must be elaborated as a **single page** document, that is to say if the document has images, links or any external document it must be accessed from tags and links, and it must be named with the nomenclature **A2.3_ActivityName_StudentName.pdf.**.
- It is required that the .MD contains a tag of the link to the repository of your document in GITHUB, for example **Link to my GitHub** and at the conclusion of the challenge it should be uploaded to github.
- From the **.md** file export a **.pdf** file that should be uploaded to classroom within its corresponding section, serving as evidence of your delivery, since being the **official** platform here you will receive the qualification of your activity.
- Considering that the .PDF file, which was obtained from the .MD file, both must be identical.
- Your repository, in addition to having a **readme**.md file in its root directory, with information such as student data, work team, subject, career, advisor data, and even logo or images, must have a contents section or index, which are actually links or **links to your .md** documents, *avoid using text* to indicate internal or external links.
- We propose a structure such as this one indicated below, however you can use any other that supports you to organize your repository.

```
| readme.md
| | blog
| | | C2.1_x.md
| | | C2.2_x.md
| | | C2.3_x.md
| | img
| | docs
| | | A2.1_x.md
| | | A2.2_x.md
| | | A2.3_x.md
```

✏️ Development

1. Consider applying the following architecture patterns to the case study

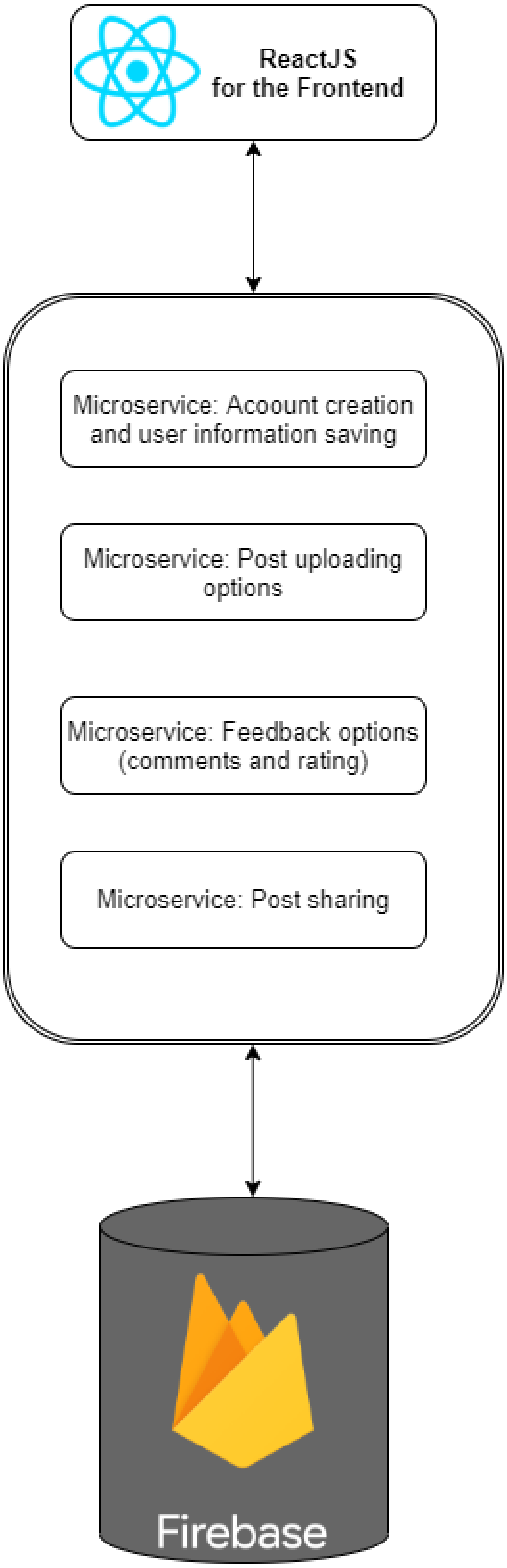
   - ☑ Client-server architecture pattern

- ☑ Layered architecture pattern
- ☑ Architecture pattern Model view controller
- ☑ Micro-service oriented architecture pattern
- ☑ Architecture pattern Pipes and filters
- ☑ Event-driven architecture pattern
- ☑ Repository or blackboard 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.

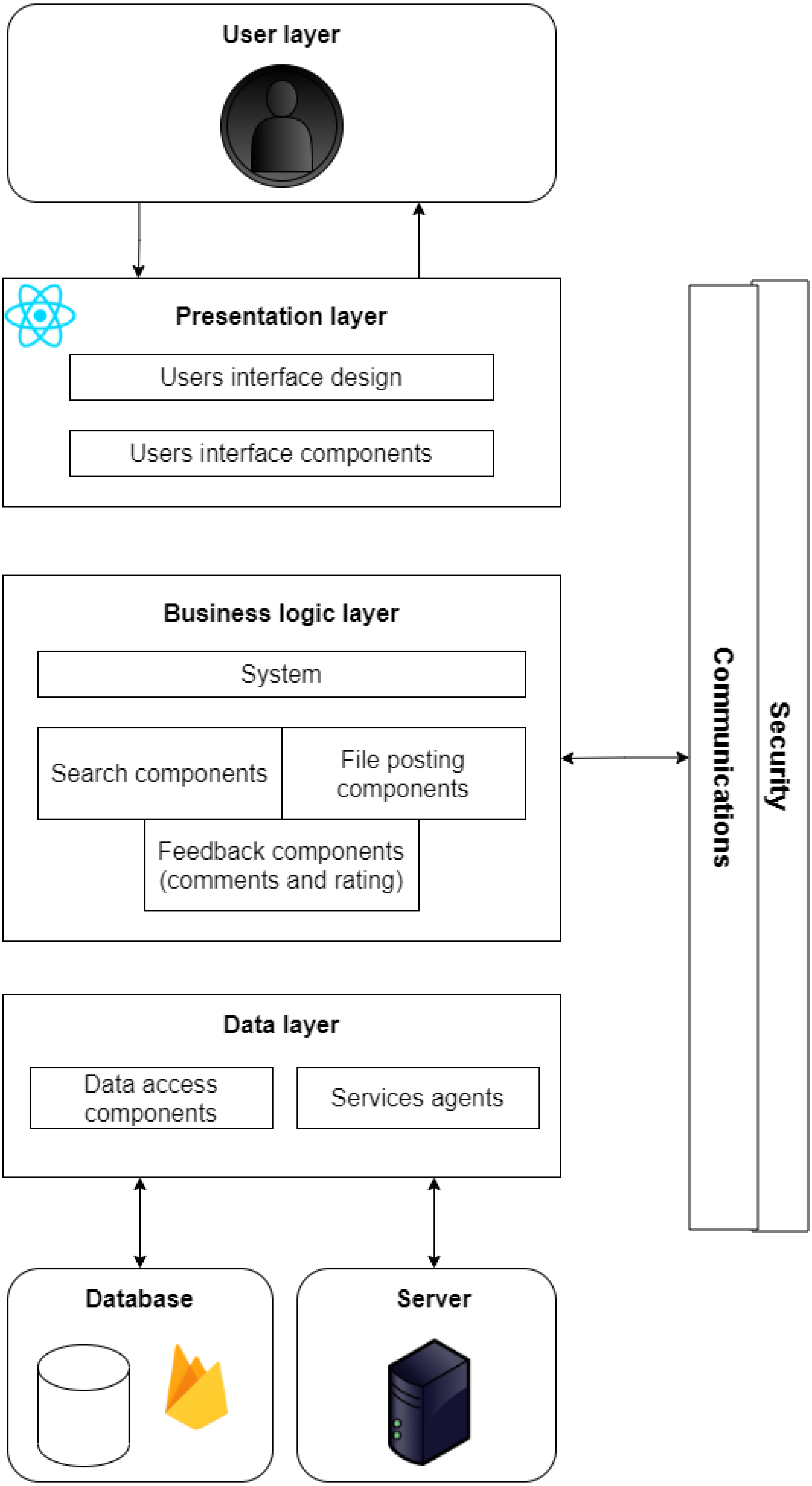
    - 2.1 Considering that it is desired to maintain and scale the system, it is sought to develop the application to through the decomposition of small independent and isolated services, which consume a external interface to communicate to a database server.

Micro-service oriented architecture pattern


ReactJS
for the Frontend
Microservice: Acoount creation and user information saving
Microservice: Post uploading options
Microservice: Feedback options (comments and rating)
Microservice: Post sharing
Firebase

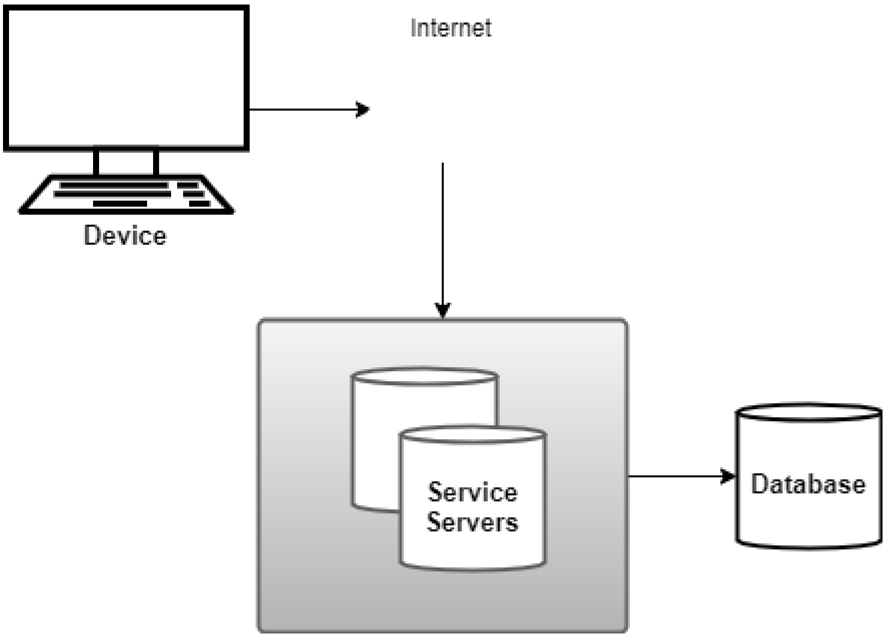
- 2.2 Being confidentiality and security attributes or requirements for the case study, it is seeks to structure each of the components to be programmed into groups of sub-tasks, where each of these sub-tasks must communicate an intermediate layer and this to another layer higher.

Layered architecture pattern


User layer
Presentation layer
Users interface design
Users interface components
Business logic layer
System
Search components
File posting components
Feedback components (comments and rating)
Data layer
Data access components
Services agents
Database
Server
Communications
Security

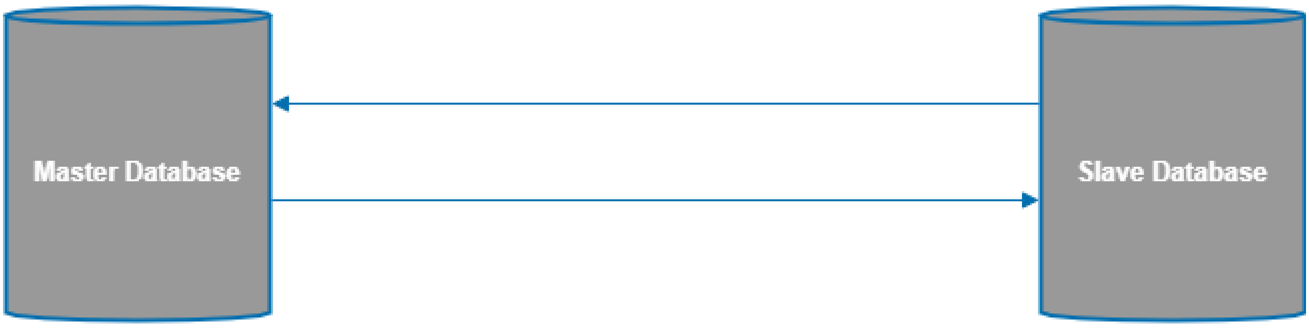
- 2.3 Looking for the availability of the system, it is proposed to install two service servers, where these make their requests to a third server where the database would be stored

Client-server architecture pattern



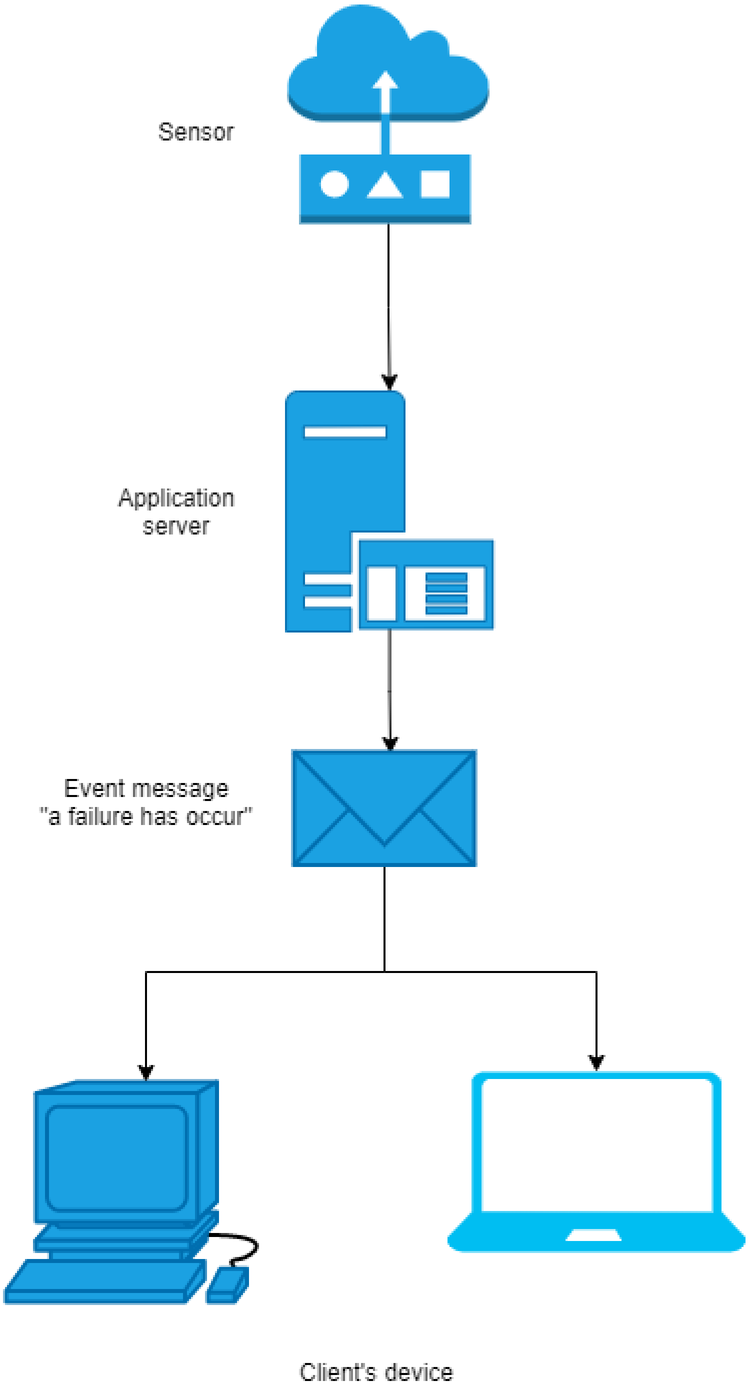
- 2.4 Identifying that the integrity of the data is a requirement, it is proposed to replicate and synchronize the database of data stored within the main server to another, considering the main one as the master and secondary as a slave.

Master-slave architecture pattern



- 2.5 Considering that the client requested that each time a failure occurs, it must be provided alarms to the different users regardless of the place where they are, it is observed that You must hire a service provider to trigger the alarms and they reach users required.

Event-driven architecture pattern



Conclusions

- **Bernal Arellano Roberto:** The making of this activity was so useful for me, with this archeticture patterns we were able to develop solutions to common problems based on giving descriptions of our platform elements.
- **Cisneros Acosta Jose Enrique:** Continuing the documentation of a project, we have the next step that are the architecture patterns that allow us to see a new perspective of our project and what will be its function in detail.
- **Pimienta Castillo Kevin Aryam Cristopher:** In this activity we look for information about the architecture patterns, of which we only knew the client-server pattern. The publisher / subscriber

architecture pattern, I think it could be used in our case study. We should take into account what each pattern does, as it is useful enough.

- **Sandoval Salazar Esmeralda:** The architecture patterns are useful for when you are starting a project and you do not know where to start, also help us define the application from a bigger perspective, with the architecture pattern you focus on a great level and have a single objective.

Bomb: Rubric

Criteria	Description	Scoring
Instructions	Are each of the items listed in the Instructions section met?	10
Development	Did you respond to each of the points requested in the development of the activity?	60
Demonstration	Is the student presented during the explanation of the functionality of the activity?	20
Conclusions	Is a personal opinion of the activity included for each of the team members?	10

🏫 Github repository