Software Architecture

Presented by:

Hamed Mohseni - hamohseni@unal.edu.co

Gabriel Mauricio Molina Perez - Gabriel Mauricio Molina Perez

Harold Velasco - Harold Armando Velazco Ayala

Daniel Galvis - Carlos Daniel Galvis Nino

Teacher:

Jeisson Andres Vergara Vargas

June 2023

National University of Colombia

Faculty of Engineering

Department of Systems and Industrial Engineering

2023

Content

1. Introduction **2**
1.1. Team 2
1.1.1. Name (1A, 1B, ..., 2A, 2B, ...) 2
1.1.2. Members 2
1.2.1. Name 3
1.2.2. Logo 3
1.2.3. Description 3
2.1. Decomposition View 4
2.1.1. Graphical Representation 4
2.1.2. View Description. 4
2.2. Components and Connectors (C&C) View 5
2.2.1. Graphical Representation 5
2.2.2. View Description. 6
2.3. Data Model View 8
2.3.1. Graphical Representation 8
2.3.2. View Description. 11

1. Introduction

1.1. Equipment

1.1.1. Name (1A, 1B, ..., 2A, 2B, ...)

Team: 2D

1.1.2. Members

- Carlos Daniel Galvis Nino
- Hamed Mohseni
- Gabriel Mauricio Molina Perez
- Harold Armando Velazco Ayala

1.2. Software System

1.2.1. Name

SIBUN: National University Welfare Information System

1.2.2. Logo

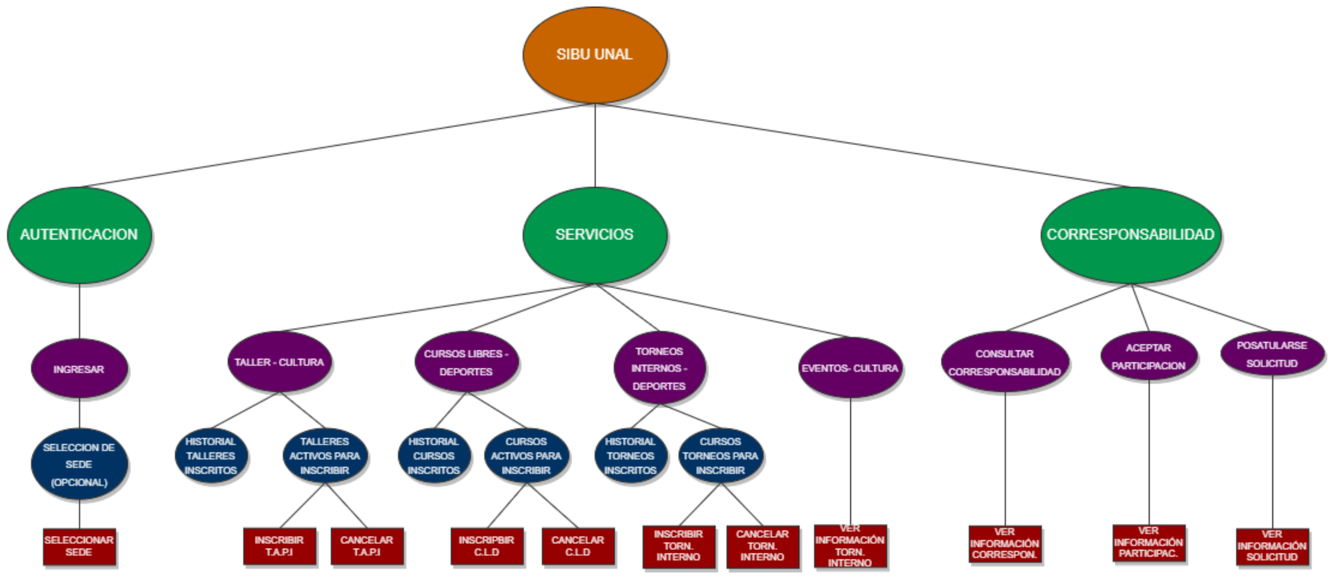
1.2.3. Description

SIBUN is responsible for managing and administering calls for management and promotion, workshops and events both cultural and sporting, free courses and sports tournaments, among other activities of the Welfare Department of the National University of Colombia.
The objective is to develop SIBUN with a microservices architecture with various programming languages and SQL and non-SQL databases, whose functionality is similar to that of SIBU, the system on which we base our project.

2. Architectural Views

2.1. Decomposition View

2.1.1. Graphical Representation


SIBU UNAL
AUTENTICACION
SERVICIOS
CORRESPONSABILIDAD
INGRESAR
SELECCION DE SEDE (OPCIONAL)
SELECCIONAR SEDE
TALLER - CULTURA
HISTORIAL TALLERES INSCRITOS
TALLERES ACTIVOS PARA INSCRIBIR
INSCRIBIR T.A.P.I
CANCELAR T.A.P.I
CURSOS LIBRES - DEPORTES
HISTORIAL CURSOS INSCRITOS
CURSOS ACTIVOS PARA INSCRIBIR
INSCRIPBIR C.L.D
CANCELAR C.L.D
TORNEOS INTERNOS - DEPORTES
HISTORIAL TORNEOS INSCRITOS
CURSOS TORNEOS PARA INSCRIBIR
INSCRIBIR TORN. INTERNO
CANCELAR TORN. INTERNO
EVENTOS- CULTURA
VER INFORMACIÓN TORN. INTERNO
CONSULTAR CORRESPONSABILIDAD
VER INFORMACIÓN CORRESPON.
ACEPTAR PARTICIPACION
VER INFORMACIÓN PARTICIPAC.
POSATULARSE SOLICITUD
VER INFORMACIÓN SOLICITUD

2.1.2. Description of the view.

Project: SIBU UNAL

Modules:

- Authentication: Module where user data is reviewed
- Services: The different services provided by the platform.
- Co-responsibility: Section responsible for assigning a job to a user who must perform this task.

Submodules:

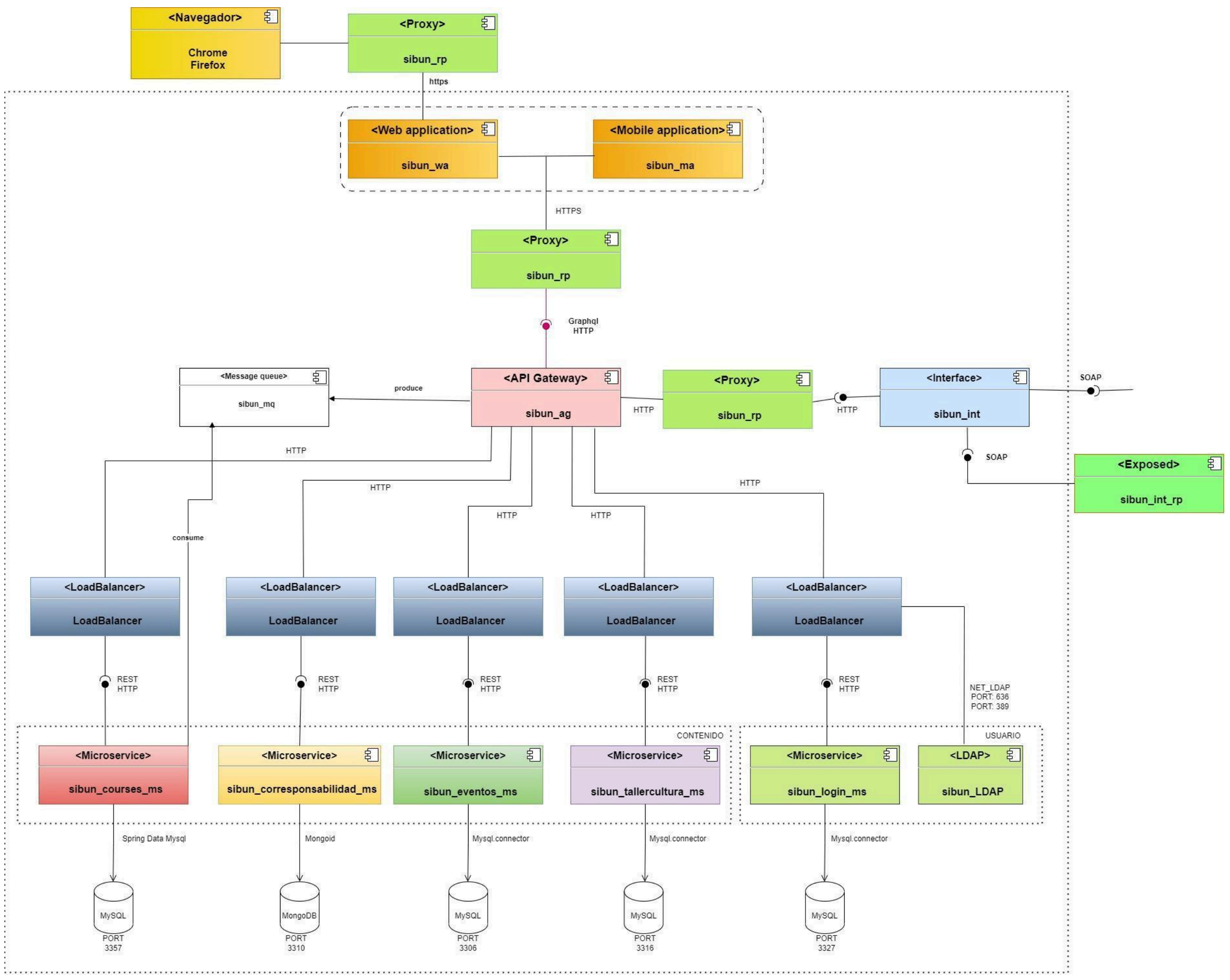
- Get into: Process where you try to enter values to verify whether you can access the system or not.
- Workshop-Culture: Registration for cultural workshops offered by university welfare.
- Free Courses - Sports: Creation of courses and their respective registrations
- Internal Tournaments - Sports: Creation of different tournaments that encourage the integration of active students.
- Events - Culture: Publication of events in the culture area, such as talks, activities and meetings.
- Consult Co-responsibility: The different types of aid needed at the university are sought.
- Accept Participation: Accept to be included in a co-responsibility participation by university welfare.
- Apply Application: Apply for the need to participate in a co-responsibility.

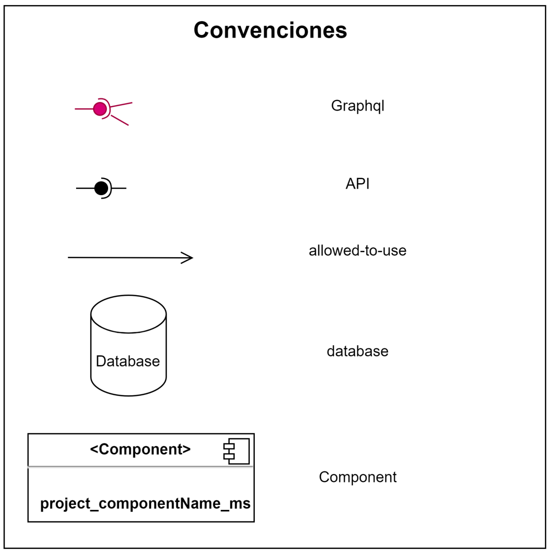
Features:

- Select location: The user must be able to select a location when making a deposit.
- History of registered workshops: You can view all the workshops in which a user has managed to register.
- Active workshops to register for: These are workshops that are active during that period.
- Register workshop: The user must be able to register for the workshop.
- Cancel workshop: The user must be able to cancel the workshop.
- History of enrolled courses: You can view all the courses in which a user has managed to enroll.
- Active courses to enroll in: These are courses that are active during that period.
- Enroll in available free courses: The user must be able to enroll in the free course.
- Cancel available free courses: The user must be able to cancel his/her registration to the free course.
- History of registered tournaments: You can view all the tournaments in which a user has managed to register.
- Active tournaments to register for: These are tournaments that are active during that period.
- Register Internal Tournament: The user must be able to register for the internal tournament.
- Cancel Internal Tournament: The user must be able to cancel his registration to the internal tournament.
- View information on cultural events: Users will be able to consult information on events.
- View co-responsibility information: You can see the basic data corresponding to the user and their co-responsibility.
- View participation information: See the status of your participation in the calls for co-responsibility
- View request information: View the information submitted and provided for co-responsibility.

2.2. Components and Connectors (C&C) View

2.2.1. Graphical Representation





2.2.2. Description of the view.

Components:

- sibun_corresponsabilidad_ms: This microservice is in charge of the co-responsibility of the students of the institution where they will be assigned a job to do in exchange for the benefits obtained. It was carried out in
- sibun_eventos_ms: This microservice is in charge of the events offered by the institution, where students will have to pay and register. It was made in Javascript.
- sibun_cursos_ms: This microservice is in charge of the free courses offered by the institution, where students will have to pay and register. It was created in Java, in the Spring Boot framework.
- sibun_torneosinternos_ms: This microservice is in charge of the internal tournaments within the university for the different sports. It was made in Javascript, specifically in Node Js and Express JS.
- sibun_tallercultura_ms: This microservice is responsible for containing information about the institution's culture workshops. It was created in Python, using the Django framework.
- sibun_login_ms: This microservice is responsible for granting access to the platform to the institution's authorized users. It was created in PHP, in the Laravel framework.
- sibun_corresponsabilidad_db:This database was created in mysql and contains the access data for the microservice.
- sibun_eventos_db: This database was created in mysql and contains the access data for the microservice.
- sibun_cursos_db: This database was created in mysql and contains the access data for the microservice.
- sibun_torneosinternos_db: This database was created in MongoDB and contains the microservice data.
- sibun_tallercultura_db: This database was created in mysql and contains the access data for the microservice.
- sibun_login_db: This database was created in firebase and contains the system login details.
- sibun_ag: API Gateway is a component in charge of directing and filtering requests from each of the microservices present in the sibun App. It was made in Javascript and Node Js. In the GraphQl API
- sibun_wa_rp: Reverse proxy from the browser to the web component, redirecting requests.
- sibun_ma_rp: Reverse proxy of the mobile application to the API Gateway, redirecting requests.
- sibun_int: interface component that exposes a soap web service
- sibun_int_rp: Reverse proxy that redirects requests to the interface.

Connectors:

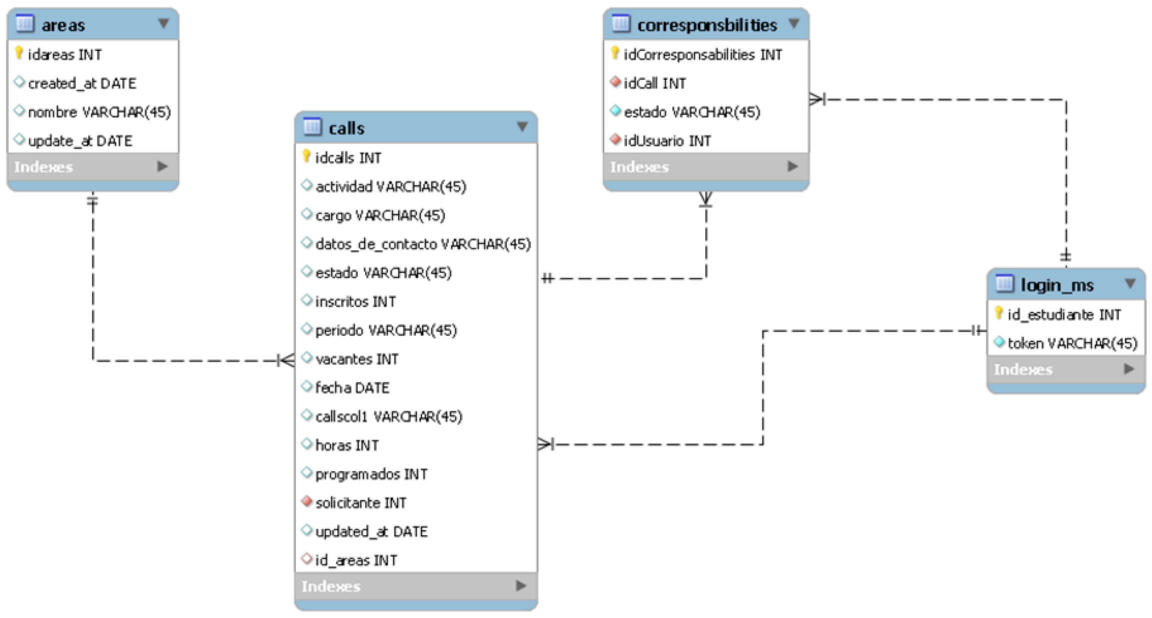
- Mysql.connector (python): The connector is used to connect to mySQL server from python, you just need to install it and import it into the py document.

- Mysql.connector (php): The connector is used to connect to mySQL server for php language, implemented in php 7.0-apache version and mysql 5.7, connect a relational database and a REST-API in the database.php file.
- Mongoose (node): The connector is used to connect to MongoDB using the Express.js framework for the backend; it is imported as an npm package.
- Mongoid (ruby on rails): The connector is used for data access in MongoDB.
- Spring Data Mysql (java): The connector is used to connect to mySQL server from java, you only need to install and import it.
- API: This is the connector responsible for connecting to API Gateway with a specific microservice.
- Graphql: This connector is responsible for connecting the API Gateway with the web and mobile components.
- HTTP: connector responsible for communication between mobile and apigateway to direct requests.
- HTTPS: connector with SSL certificates responsible for communication between the browser and the web component.
- SOAP: connector responsible for exposing web services to be consumed.

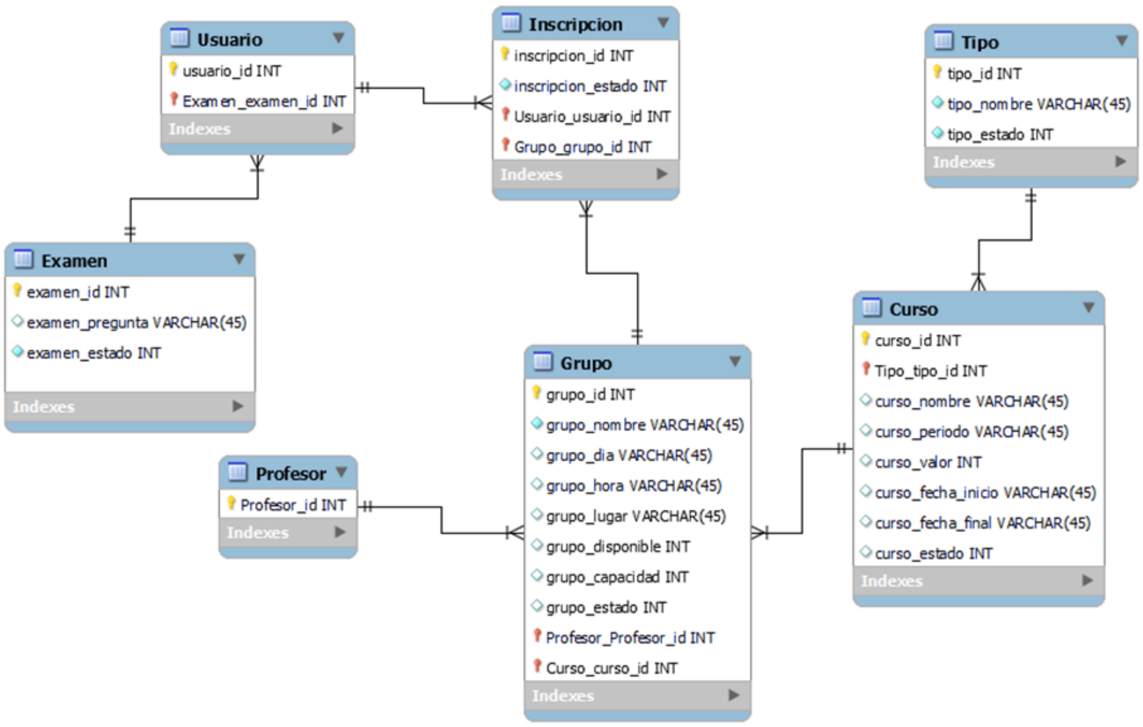
2.3. Data Model View

2.3.1. Graphical Representation

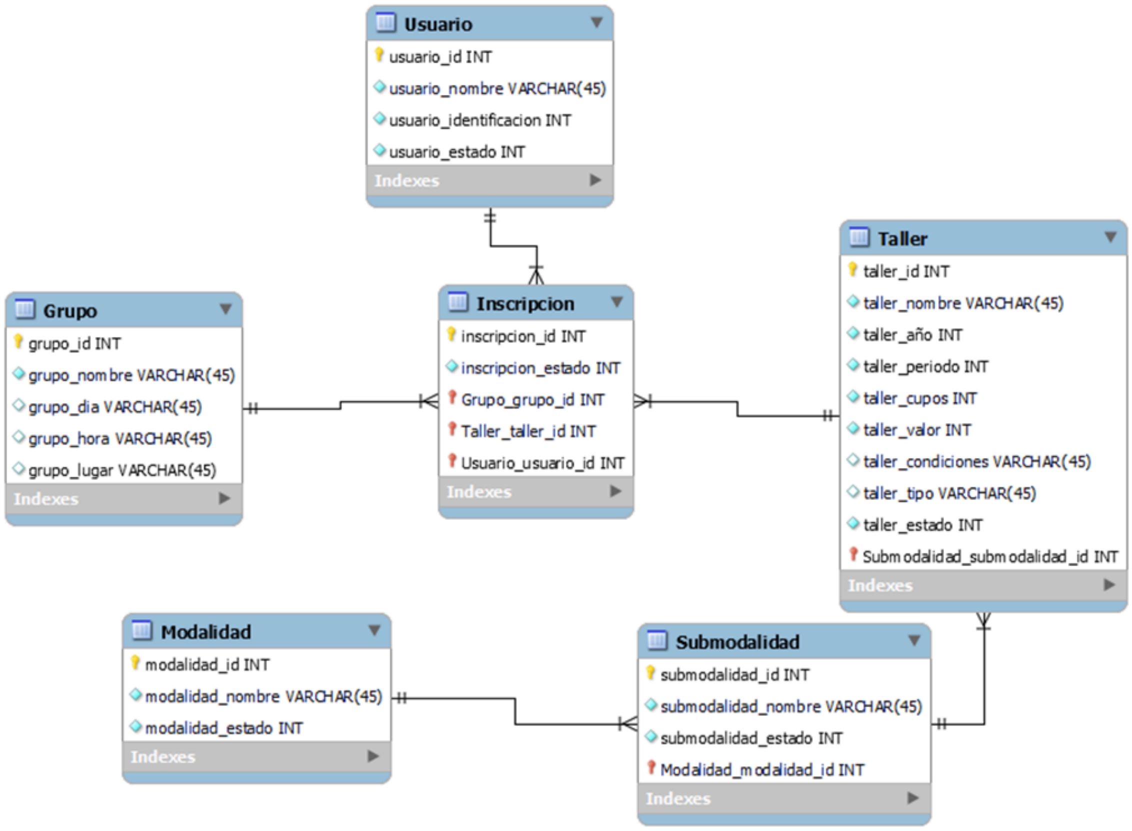
sibun_coresponsibility_db:



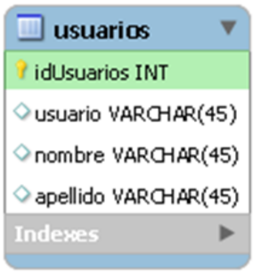
sibun_cursos_db:



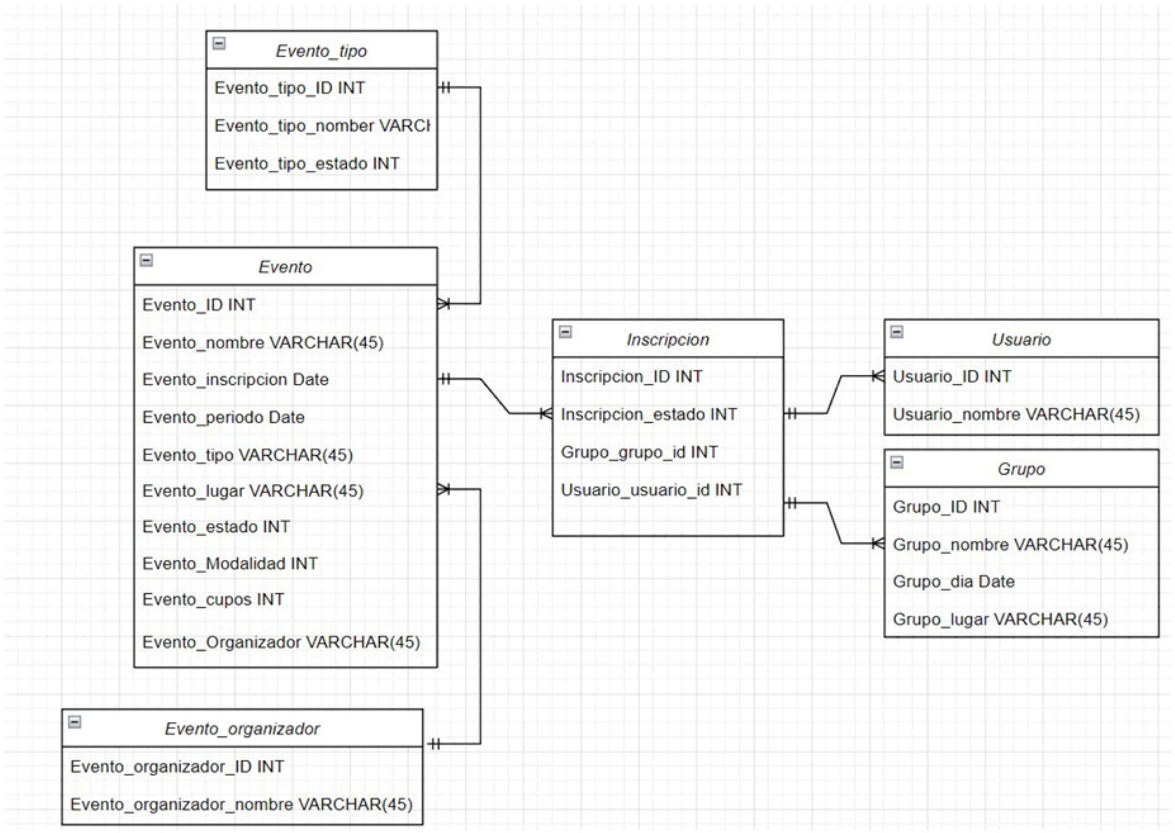
sibun_tallercultura_db:



sibun_login_db:



sibun_events_db:

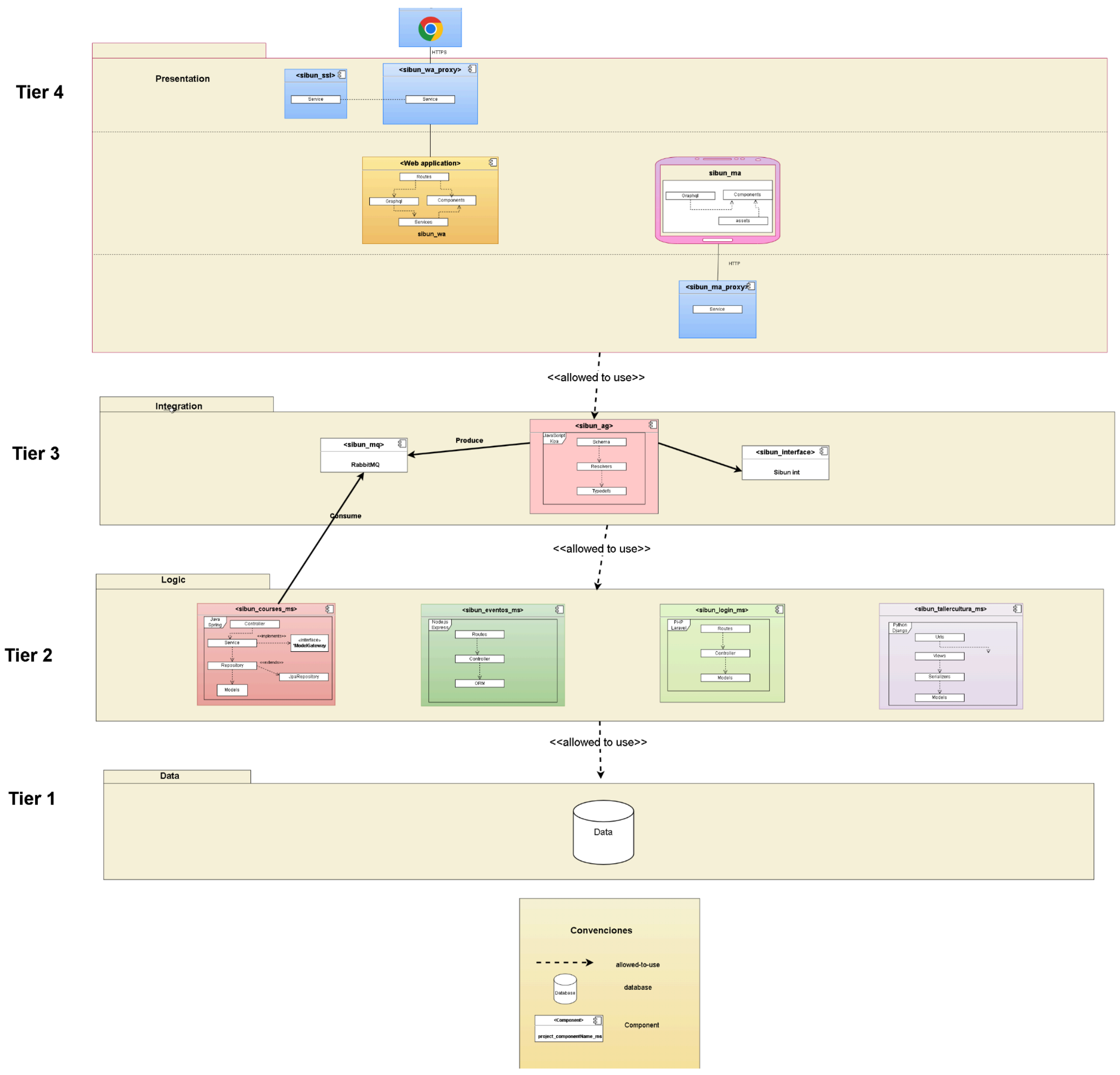


2.3.2. Description of the view.

- sibun_corresponsabilidad_db: This database was made in SQL and has 4 tables where you have data about the student, vacancies and areas of the institution.
- sibun_cursos_db: This database was created in SQL, it has 7 tables with information on student registrations, payment, assigned teacher, and type of course.
- sibun_tallercultura_db: This database was created in SQL, it has 6 tables, with the student's data, their registration and the workshop with its modality and submodality.
- sibun_eventos_db: This database was created in SQL, it has 5 tables for events, students, registration and the organizer.
- sibun_login_db: This database was created in Nosql, firebase, it has a single table with the data of the students in the system.

2.4. Layer View

2.4.1. Graphical Representation



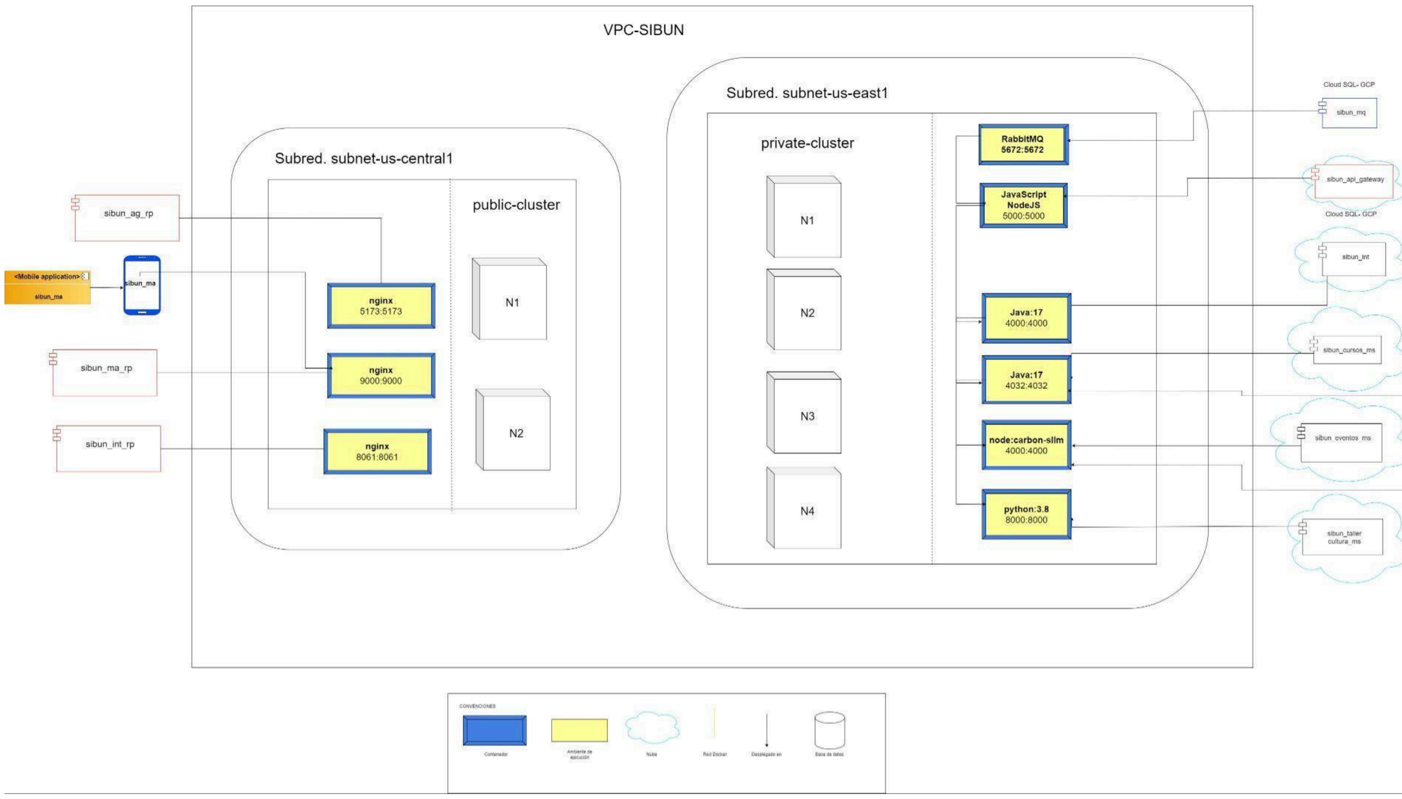
2.4.2. View description.

It consists of two layers:

- Data Layer: This is the layer responsible for providing the information stored in the system's different databases.
- Logical layer: This layer is accessed by the integration layer through the sibun_ag which ensures that requests are redirected to each of the microservices, depending on the desired purpose.
- Integration layer: This layer is accessed by the presentation layer to make query and mutation type requests, in order to bring data.
- Representation layer: This layer is responsible for presenting all the information through views and routing to various pages.

2.5. Deployment View

2.5.1. Graphical Representation



2.5.2. View description.

Cloud: There are two microservices that are based on the cloud SIBUN_Firebase_db and sibun_mq, the first is based on services in Google and the second on AWS services.

Container: This is where the execution environments are housed, along with their dependencies to carry out the execution properly, and they require connectors to relate different containers.

Execution environment: contains the internal logic of each microservice, to manage the data.

- sibun_cursos_ms: is set to Java language:17 on ports 4032:4032.
- sibun_eventos_ms: runs in the NodeJS:carbon-slim language on ports 4000:4000.
- sibun_corresponsabilidad_ms: runs on Ruby:3.1.3 on ports 3010:3010.
- sibun_tallercultura_ms: is set to Python:3.8 language on ports 8000:8000.
- sibun_login_ms: is set to PHP:8.2 on ports 2626:2626.

Docker network: Docker environment connectors to make connections between containers, using defined ports.

Database: Persistence of data stored in the system, requires connections with containers, so that this data is managed.

- ❖ sibun_cursos_db: is set to MySQL language on ports 3057:3057.
- ❖ sibun_eventos_ms: is set to MySQL language on ports 3306:3306.
- ❖ sibun_corresponsabilidad_ms: is set to MongoDB language on ports 27018:27018.
- ❖ sibun_tallercultura_ms: is set to MySQL language on ports 3326:3326.
- ❖ sibun_login_ms: Set to MySQL language on ports 3316:3316.

3. Quality Attributes

3.1 Security

3.1.1 Login:

data type boolean.

3.1.2 Token:

To maintain the session, the system uses a token generated in the backend, which is updated for each request made. If the token has already expired, the session ends and returns a boolean data type.

3.1.3 LDAP:

The LDAP protocol was used to manage the credentials and permissions of the users of our application, through a client-server connection. This is to perform a double validation of the user through LDAP and the authentication microservices database.

3.1.4 Reverse Proxy:

A reverse proxy has been implemented to validate the origin of the request made to the API gateway component and uses the HTTPS protocol to make access more secure and prevent illegal access from suspicious sources, the certificate has been generated using OPENSSL and the proxy was generated with Nginx Proxy Management .

3.1.5 HTTPS:

The web component adds an SSL certificate that was generated as follows:

- Using the commands

```
> openssl genrsa -out key.pem 2048

> openssl req -new -key key.pem -out csr.pem

> openssl x509 -req -days 365 -in csr.pem -signkey key.pem -out cert.pem
```

- After this, the Nginx Proxy Management tool was used to configure custom SSL with the name Sibún and adding it to the reverse proxy.

3.2 Interoperability

3.2.1 REST and Graphql:

It was already developed in previous prototypes

3.2.2 SOAP Web Service:

A software system created to expose a SOAP service. This system feeds request information to the API-Gateway component by means of a reverse proxy. In addition to this, the component also exposes a WSDL file, which specifies the exposed functions along with the objects required for the requests.

3.3 Performance and Scalability

3.3.1 Main node and nodes

The nodes, both parent and child, were deployed as instances in the Google Cloud. And with the help of Docker they were linked so that the parent node has the child nodes available.

3.3.2 Horizontal Scaling Ms

For each of the microservices listed above, a Docker image was created and uploaded to Docker Hub. This image is deployed on a node and scaled as many times as needed (limited by the infrastructure).