Microservice Architecture for a school Management System

School Management System

Student Management

Non-Teaching Staff Management

Financial Information Management

Teaching Staff Management

School Management System Flowchart

The school management system has four main operations: Student management, Staff management, Financial Information management and Nonteaching staff management. The architecture to be used for microservice architecture.

What is microservice architecture?

A microservices architecture is an approach to building a server application as a set of small services. Each microservice is hhighly maintainable and testable, loosely coupled, independently deployable, organized around business capabilities, and owned by a small team.

Each service runs in its own process and communicates with other processes using protocols such as HTTP/HTTPS, Web Sockets.

This architecture utilizes APIs to pass information, such as user queries or a data stream, from one service to another.

Why choose microservices architecture for the school management system?

The management system can be broken down into specific end-to-end domain or business capability within a certain context boundary, and each developed autonomously and be deployable independently, each with its own related domain data model and domain logic.

It can be broken down into microservices like student management, staff management, financial information management and nonteaching staff management that communicate with each other through API calls.

This has the advantage of enabling scale out independently. That is, instead of having a single application that you must scale out as a unit, you can instead scale out specific microservices. That way, you can scale just the functional area that needs more processing power or network bandwidth to support demand, rather than scaling out other areas of the application that don't need to be scaled. That means cost savings because you need less hardware.

Also, as the constituent services are small, they can be built by one or more small teams from the beginning separated by service boundaries which make it easier to scale up the development effort if need be.

Microservices also offer improved fault isolation whereby in the case of an error in one service the whole application doesn’t necessarily stop functioning. When the error is fixed, it can be deployed only for the respective service instead of redeploying an entire application.