A SCALABLE ARCHITECTURE DESIGN FOR SCHOOL MANAGEMENT SYSTEM

Introduction

A school management system (SMS) is a software solution for schools to keep track of their activities. These activities include, students' grades, admission, staff records, payment of fees, performance monitoring, student and staff attendance etc.

The SMS to be designed here has the following main parts:

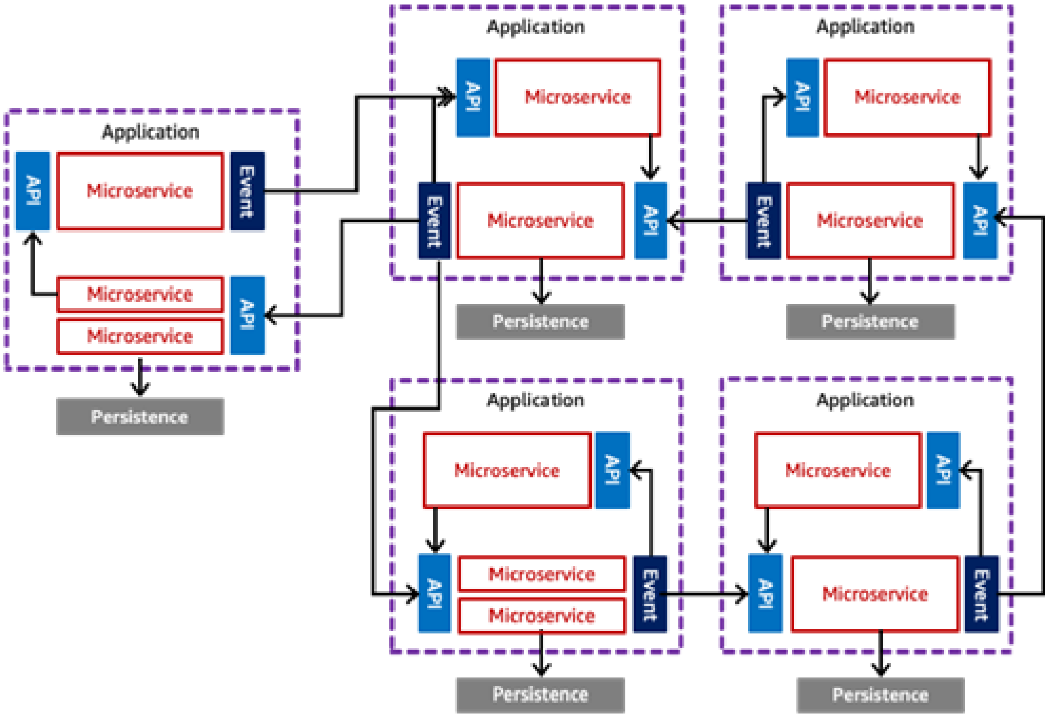
- Student Information System – Manages student data and student admissions
- Staff Information Management – Manages Staff data and hiring of new staff.
- Course Registration – Assigns courses to teachers and students to courses.
- Exam and Result Management – Manages student grades
- School Fees Management – Manages payment of fees
- Attendance Management – Handles student and staff attendance to school
- Inventory Management – Keeps tracks of items in the school
- Library Management – Keeps track of books in the school

The SMS described above pretty complex to build, thus the need for an efficient architecture to ensure the application thus not lag when usage increases and is also able to support users all over the world.

System Architecture

To ensure this system is scalable, robust and highly efficient, the SMS will be designed using a microservices architecture. By this, each business logic in the application is broken down into a unit that can function on its own. These units can then communicate with others by exposing APIs that other parts of the application can utilize. This form of development will ensure that unit testing is done easily and as the system grows maintenance will be easy.
Thus, each unit will have its own database it manages. Each database will also have a cache to save time when a request for data is repeated. This drastically helps in making the system very fast.

With this architecture, a single frontend can be built that accesses different microservices when an event is triggered. For example, when a user clicks to pay fees, the student management microservice is called by the school fees management to ensure that the money being payed is associated with the appropriate student.

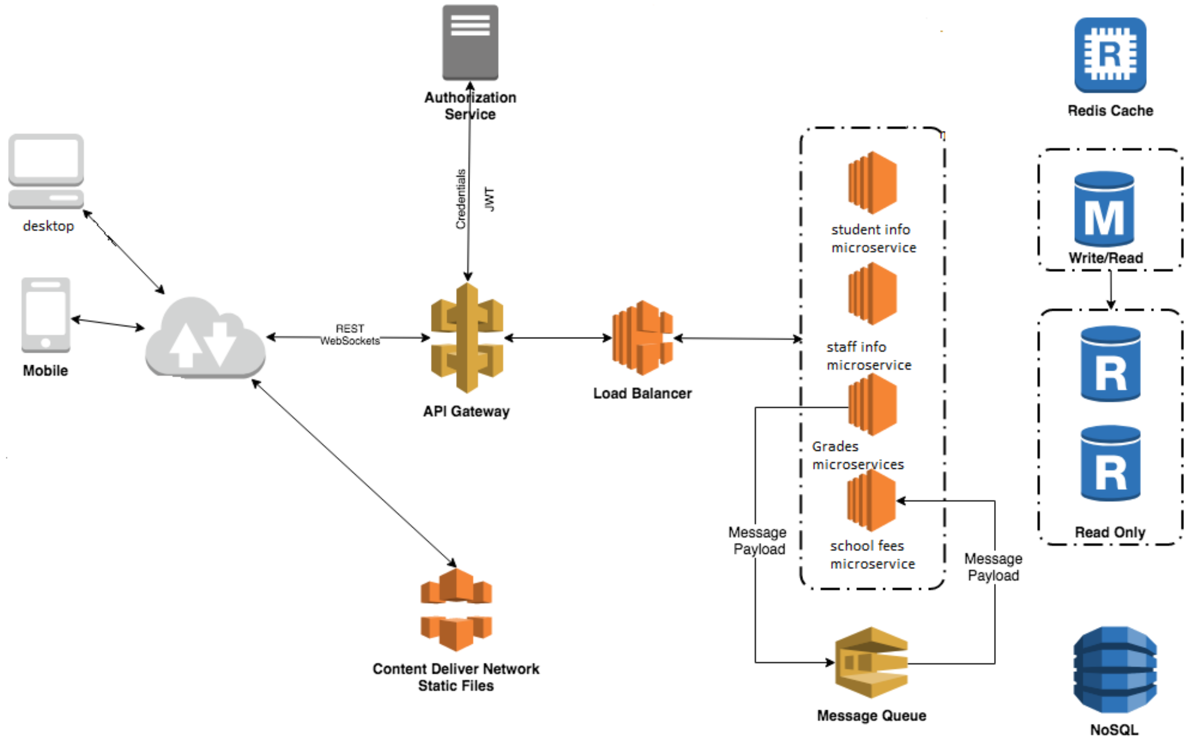


The image above describes how the individual components will work together.

User Interaction with System

Given that the system will be used by users all over the world, keeping the application on a single server will give poor user experience as page load will be slow increasing latency. To help in speed certain decisions has to be made. This includes, Using a Content Delivery Network for assets instead of getting static assets from the server site is hosted and Introduction of a load balancer between client and servers to help retrieving data faster and getting servers that are less busy to handle requests.

The image below describes how access will be done on the system


Authorization Service
Redis Cache
desktop
Credentials
JWT
student info microservice
Write/Read
Mobile
REST
WebSockets
API Gateway
Load Balancer
staff info microservice
Grades microservices
Message Payload
school fees microservice
Message Payload
Read Only
Content Deliver Network Static Files
Message Queue
NoSQL