The role of backend and frontend information systems infrastructure

Sherzod Baxtiyorovich Rajabov
sh.rajabov@tsue.uz
Tashkent State University of Economics

Abstract: This paper is based on using of backend and frontend information systems infrastructure in programming processes. And it illustrates how increasing the role of the ICT sector in the modern era.

Keyword: Backend, frontend, information systems, server, serverless backend, server-oriented backend, software, software technology.

INTRODUCTION

A backend system refers to any structure or setup that runs and supports corporate back-office applications. Backend systems could take the form of servers, mainframes, and other systems that offer data services. Simply put, they are computers and devices that end-users don't see since they work in the background. Nevertheless, a backend system plays a critical role in any organization's operation.

When you think about a backend system doing several things in the background, you might think of a chaotic scene in a fast-food chain's kitchen. Someone is in charge of preparing the drinks while another operates the fry station. Instead of several pieces of equipment, though, a backend system could include a single computer only.

Separating the frontend from the backend makes everything simpler. After all, you seldom see a waitress prepare the meals that he/she serves.

So, the backend covers the following aspects of a program:

1) The hardware resources
2) The software technology
3) The network infrastructure

There are three main classes of server architecture. The three classes or server architecture are:

1) Serverless Backend.
2) Server-oriented Backend.
3) Decentralized Backend.

Serverless backend is one of the most popular server architectures these days. The server management and maintenance are under some third-party service providers. These providers are also responsible for the backup and security of the system.

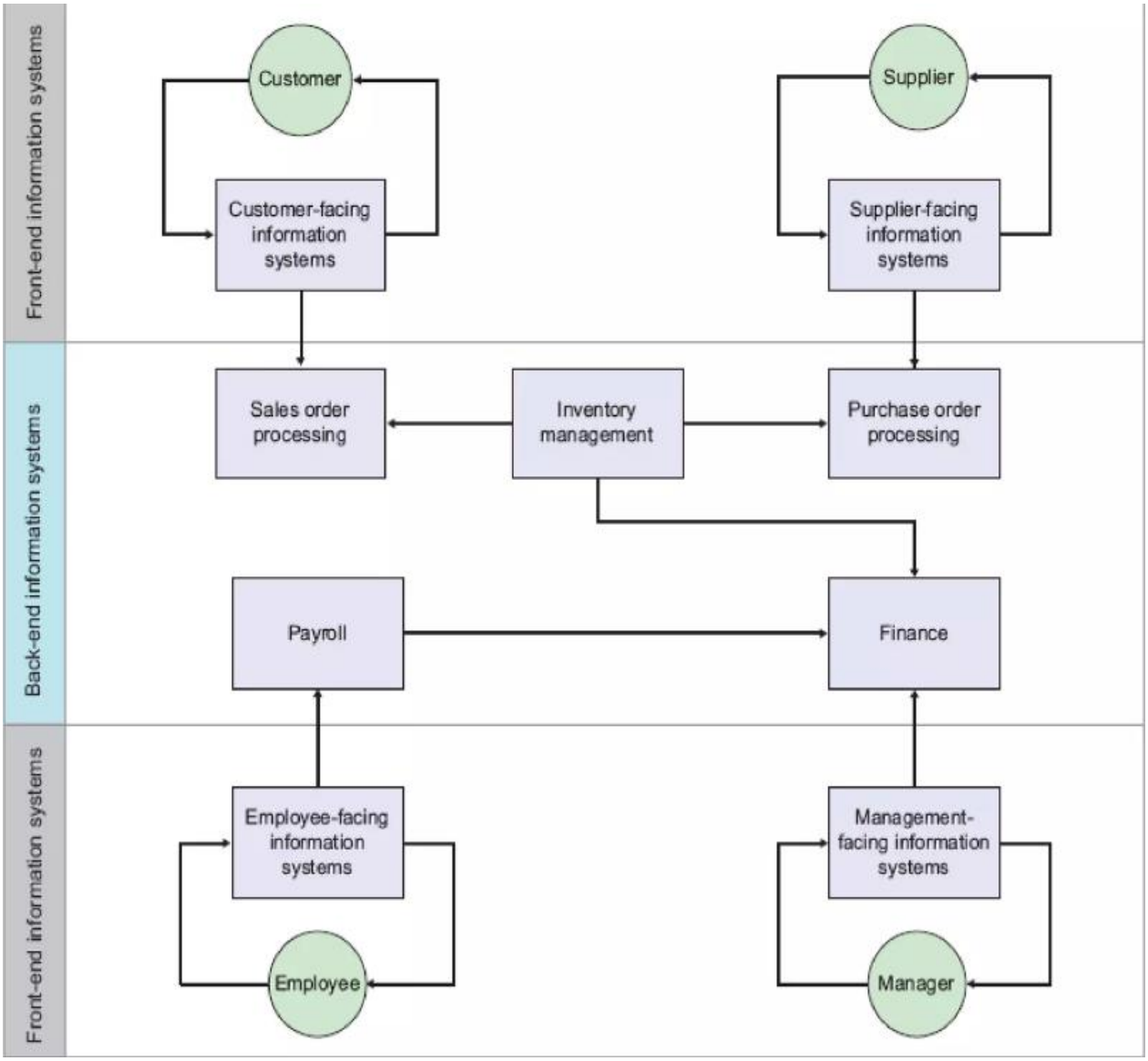


Figure 1. Back-end and Front-end information systems infrastructure.

As the service provider manages the server, there is no need for server management in the program. It makes it a magnificent architecture for the developers. The speed of development is the main reason here.

This architecture also provides hassle-free development and deployment because all developers have to deal with the application's frontend. According to the experts, this is a more secure option because the service providers have developers working day and night to secure their systems.

However, being a cloud-based technology, the chances of attacks are also high. So, it is a great option to take protective measures. Examples for serverless architectures are a Backend as a Service and Function as a Service.

Examples of serverless backends are Back4app, Firebase, and Kinvey.

Server-Oriented Backend. It is the type of architecture where special server computers are implemented to serve as the system's server and provide the hosting services. It is also one of the most popular architectures.

The best part about using this architecture is using a physical or cloud-based server according to your requirements. Some other benefits are listed below.

Data sharing.

Integrations.

Single access point.

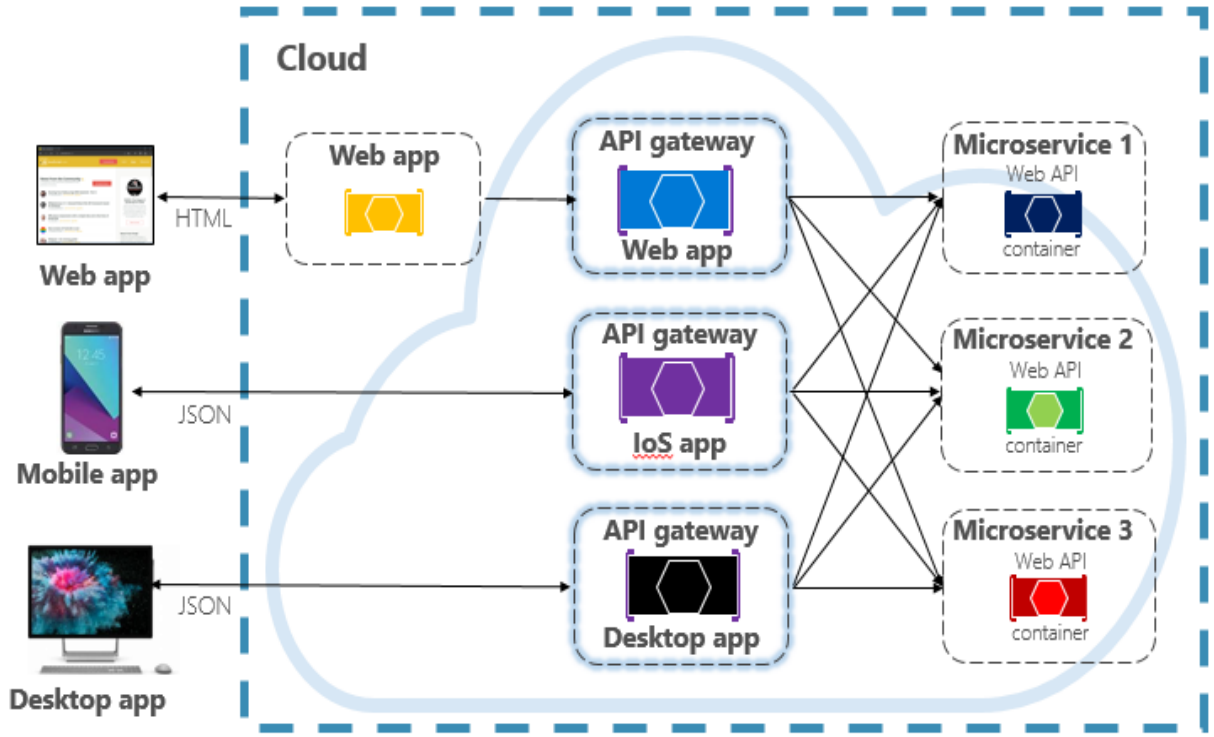
Examples of server-oriented backends are AWS, Azure, Digital Ocean, etc.

Decentralized Backend. In this type of server architecture, nobody takes the responsibility of backend management. Different physical servers are located in other physical locations, and there is no central location of the main server.

The networks made on this type of architecture are usually open-source, and they provide peer-to-peer networking. The best part about using this architecture is that developers can use any language to develop the frontend. However, the front-end hosting is needed to be done by other apps like IPFS or Swarm.

Frontend information systems infrastructure.

Frontend infrastructure teams empower product teams with the foundational frontend ecosystem and reliable, performant, and developer-friendly tools to efficiently build great user experiences.



Firure 2. Front-end client communication structure.

Move core metrics: developer experience, developer velocity, debuggability, performance, and reliability.

Enhance developer productivity by improving the tooling setup.

Linting: enforce best practices with static analysis and eslint rules.

Unit and end to end tests.

Deployment: continuous integration and continuous delivery

Type System: consistent and less-risky applications.

Shared Configurations: create infrastructure so teams can start building new frontend applications without needing to know tricky configuration details

Build System: bundling frontend applications.

Testing: Infrastructure and testing framework integrations enabling developers to write a comprehensive set of unit, integration, and end-to-end tests

Observability: Client-side web logging libraries, integration with vendor error monitoring solutions, alert generators for standard web metrics as well as their usage in automated canary analysis, and testing solutions to ensure logging quality

Shape the architecture of frontend systems

Define patterns for UIs (design system).

Define patterns for data fetching.

Define patterns for frontend-backend relation: graphql, BFF, rest APIs.

Make platform-wide changes and upgrade the entire codebase.

Replace old libraries with new standards.

Build a strong culture with the foundational platform knowledge.

Partnering with product teams to encourage adoption of tools and frameworks.

Share your experiences and expertise with those around you, and multiply your impact through thoughtful teaching, influencing, and setting examples.

Improve end-user experience by building infrastructure to support UX consistency across products.

Optimize the client-side performance of web applications.

Support teams to build consistent experiences through design systems.

Monitoring systems: monitoring errors in the application.

Research and test new languages, libraries and frameworks and evaluate their potential to make sure we never stop innovating.

Understand developer pain points and common questions in frontend development, and aim to improve or answer them.

Enable different product teams to be more productive by identifying similar features or tasks across teams and making improvements in the frontend stack or processes.

Engagement in the JavaScript ecosystem/community: understand the ever-evolving JS landscape to proactively ensure the rest of the organization is maintaining a technically healthy product.

Build tools and drive initiatives to ensure best practices across teams as well as maximize developer productivity and experience.

Provide teams with visibility into their test coverage and frontend performance.

Build tools and processes to increase automated testing adoption in the org

Build tooling to provide teams with visibility into their test coverage and frontend performance.

References

1. Porter, M.(2001). Strategy and the Internet. Harvard Business Review: 79 (3) 62–78

2. Bursztynsky , J. (2020, August 19). Apple becomes first U.S. company to reach a $2 trillion market cap. CNBC.

3. Information systems in economics: a textbook for academic baccalaureate. / Under the editorship of V.N. Volkova, V.N. Yuryev. – Yurayt Publishing House, 2018

4. Lapidus L.V. Digital economy: management of electronic business and electronic commerce: monograph / L.V. Lapidus. – M.: INFRA-M. 2019.-381 p.

5. Titorenko G.A. Information technology management. Tutorial. - M: UNITY-DANA, 20 - 439C

6. Asanov R.K. Formation of the concept of "digital economy" in modern science / R.K. Asanov // Socio-economic sciences and humanitarian research. – 20– No. – S. 143–148.