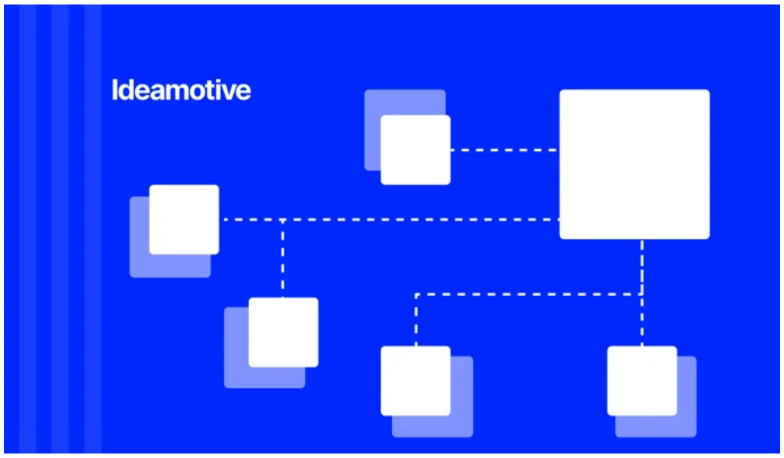
Software Architecture Design Best Practices You Should Know

Sep 27, 2022 9 min read

Dawid Karczewski

Senior full stack developer and CTO at Ideamotive.



Trending articles

What Are The Best Frontend Frameworks To Use In 2023?

Dawid Karczewski 16 min read

21 Dazzling Examples of Mobile App UI Design to Inspire You in 2023

Hire Java Developers

When developing a product, it is important to consider how it will be maintained in the future. The specifics of adding changes and editing functionality should be thought through. In addition, it is important to understand how external elements of the interface interact with internal processes, and by what architecture principles users will interact with the program.

The below-listed software architecture design best practices help with this. But first, let's figure out what software architecture is, how it works, and why software architecture design is needed.

What is software architecture?

Software architecture is the designed structure of the program, which includes defining the interaction of the interface components with the internal processes of the program. Simply put, this is a kind of approach that defines which functions are responsible for what and how they interact with each other.

There is no precise understanding and clear formulation of this process. The main task is to create a logical structure of the program and simplify the interaction between software developers. In this way, we can reach a high level of scalability of an architecture. In turn, this makes it possible to make changes to the program in the future, working on specific aspects, rather than redesigning the entire software.

Software architecture design best practices ensure that the application will perform the tasks and follow the purpose defined during the initial stages of development.

Node.js vs Java: Choosing Perfect Technology for Your Backend
Dawid Karczewski 6 min read

Picking The Best Language For iOS App Development In 2023
Michał Pruciak 7 min read

Next.js vs React: Which Framework Is Better For your Front-end?
Dawid Karczewski 11 min read

The application architecture performs a number of important tasks:

- Defines the structure of the program and allows you to understand how it is arranged, and at which levels certain tasks and functions are performed;
- Defines the behavior and interaction of elements, which makes it clear what happens if a certain action is performed;
- Defines significant and minor elements, which allows estimating the cost of development. Also it helps to understand which elements must be implemented, and which can be abandoned for reasons of economy;
- Helps to understand how scalable the application is, how difficult it will be to implement new features and what technology stack to use;
- Allows you to meet the client's needs and adapt the program to mutually exclusive requirements, for example, a high level of functionality and defining time limits, in this case, it becomes clear how to implement it;
- Allows you to understand the logical relationships in the program;
- Makes it possible to correctly maintain documentation and clearly describe the functionality, which significantly simplifies further maintenance of the program, making changes and working with the existing functionality.

These are the main tasks performed by the software architecture. By following the software architecture design best practices, you significantly simplify development and clearly understand what will be the result and how all functions will work.

Hire Java Developers

Software architects

Who they are

A software architect (system architect, IT architect) is a specialist who builds complex IT systems to solve business problems. A system architect is well versed in business processes and sees how a business problem can be solved using a variety of information technologies. They play an important role in designing the system architecture and defining the software product.

What does a software architect do?

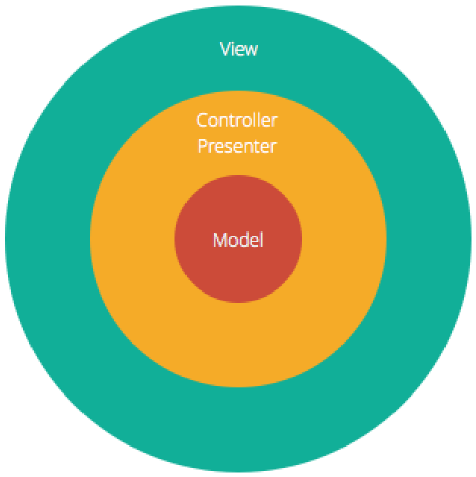
- Studies of the subject area for the implementation and/or development of applied information systems (for example analyzing best practices in iOS app development)
- Analyzes the software architecture design best practices to choose the one that fits the most
- Participates in interviews with customers, business experts, and users of information systems to study the current principles of organizing the course of processes
- Reviews and organizes project documentation
- Prepares technical documents describing the entities, relationships, and processes of the subject area using special notations
- Participates in the setting of tasks and the development of technical specifications
- Collects, analyzes, and documents the functional requirements for software
- Controls development

Hire Java Developers

- Participates in testing the prototype of the developed system
- Participates in the training of system users
- Analyzes the risks and causes of errors during system development

Guidelines and principles to organize software systems

Layered architecture



This kind of software architecture design best practices works on the principle of separation of concerns. The software is divided into layers that lie on top of each other, and each of them performs a specific duty.

The architecture divides the software into the following layers:

1. The Presentation Layer contains the user interface and is responsible for providing a good user experience.
2. The Business Logic Layer, as the name suggests, contains the application's business logic. It separates UI/UX from business-related computing. This allows

Hire Java Developers

3. The Data Link Layer is responsible for interacting with persistent stores such as databases and other processing of information that is not related to the business.

Data and controls flow through each layer in the design and are passed from one to the next. This system also increases the level of abstraction and, to some extent, even the stability of the software.

Advantages

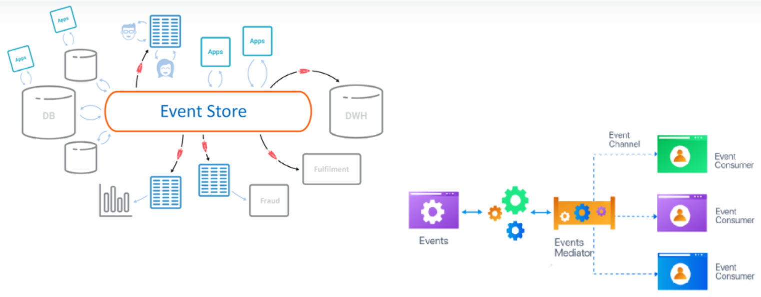
- Easier implementation compared to other approaches.
- Offers abstraction by separating responsibilities between levels.
- Isolation protects some layers from changes to others.
- Increases software manageability through loose coupling.

Flaws

- Doesn't offer much scalability.
- Software created with this approach will have a monolithic structure that makes modifications more difficult.
- The data must pass through each layer, even if it does not need to be transmitted from specific layers.

Prominent real-life example: Gmail.

Hire Java Developers



Companies like Uber, Twitter, and LinkedIn are all about real-time updates: Uber ride notifications, friends' tweets, and helpful tips from industry experts that can be received and shared within seconds.

As soon as information enters the network, it immediately becomes available to everyone around. Users will always love this simple and fast availability of information - they are always on the lookout for similar improvements in their quality of life.

To meet the ever-increasing demands of consumers, many companies are forced to abandon traditional request-response structures in favor of event-driven architectures.

Unlike request-response systems, in an event-driven architecture, the requestor sends an event (usually a message with a [header] and a [payload]) to the distribution layer. Further, this message can be received by a service that listens to a specific topic. This service can then use the payload of the message in some way or pass the message on to another service.

Advantages

- Real-time output
- Easy scalability

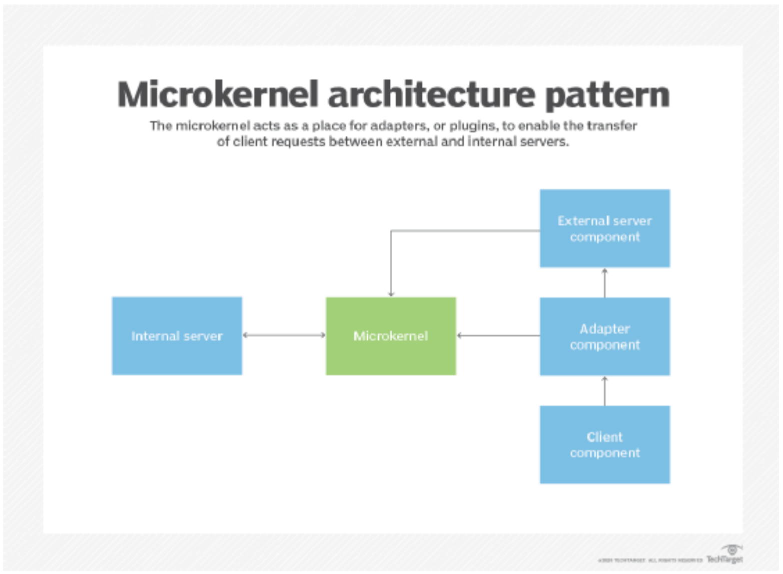
Hire Java Developers

Flaws

- Increased complexity for the API provider
- Ambiguity in API Analytics
- Limited governance, standardization, and experience/developer support

Prominent real-life examples: Netflix, Unilever, SAP, Amazon, LinkedIn, and the Federal Aviation Administration.

Microkernel architecture


Microkernel architecture pattern
The microkernel acts as a place for adapters, or plugins, to enable the transfer of client requests between external and internal servers.
External server component
Internal server
Microkernel
Adapter component
Client component
TechTarget

Microkernel architecture is an alternative to the classical way of building an operating system. Classical architecture in this case refers to the structural organization of the OS, according to which all the main functions of the operating system that make up the multilayer kernel are performed in privileged mode.

The essence of this kind of software architecture design best practices is as follows. In privileged mode, only a very small part of the OS, called the microkernel, remains running.

Hire Java Developers

specific modules, as well as modules that perform basic (but not all) kernel functions for process management, interrupt handling, virtual memory management, message forwarding, and I/O device management related to loading or reading device registers. All other higher-level kernel functions are packaged as user-mode applications.

Advantages

- Meets most of the requirements for modern operating systems to a high degree
- Portability - all machine-dependent code is isolated in the microkernel, so fewer changes are required to port the system to a new processor and they are all logically grouped together
- Extensibility
- Reliability
- Creating good prerequisites for supporting distributed applications

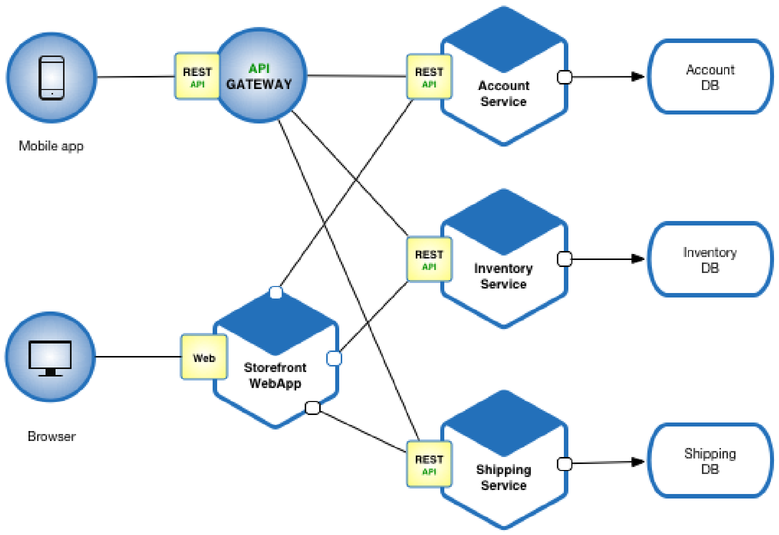
Flaws

These benefits come at the cost of reduced performance, and this is the main drawback of the microkernel architecture. With a classical organization of the OS, the execution of a system call is accompanied by two switching modes, and with a micronuclear organization - by four.

Thus, an operating system based on a microkernel, all other things being equal, will always be less productive than an OS with a classic kernel. It is for this reason that the microkernel approach has not received the widespread adoption it was predicted to have.

Hire Java Developers

Microservices architecture



In this approach, the application is developed as a set of small services, each running in its own process and communicating with lightweight mechanisms, usually an API for an HTTP resource.

These services are based on business capabilities and can be deployed independently using a fully automated mechanism.

Centralized management between services is minimal. They can be written in different languages, and use different data storage technologies.

The architecture works on the principle of service componentization. It separates the software into various isolated components (services), each of which has a single responsibility. Changes in one service should not affect others.

One of the most popular software architecture design best practices lies in building microservices in .NET.

Composition of microservices

Hire Java Developers

following 5 components:

1. Services;
2. Service Bus;
3. External configuration;
4. API gateway;
5. Containers.

Characteristics of microservices

A microservice architecture should include the following characteristics:

- Componentization through services.
- Organize business opportunities.
- Focused on products, not projects.
- Smart endpoints and dumb pipes.
- Decentralized management.
- Decentralized data management.
- Infrastructure automation.
- Failure protection.
- Evolutionary design.

It is recommended to develop each microservice separately under the control of different teams. Since data is transmitted using a standard protocol and data format, the structure of one service will not affect the functionality of related ones.

Advantages of microservices

- Offers low coupling due to its high degree of isolation.
- Increases modularity.

Hire Java Developers

- Ease of modification can speed up iterations.
- Allows you to implement an improved error handling system.
- Solves the data flow issues that a layered architecture has.

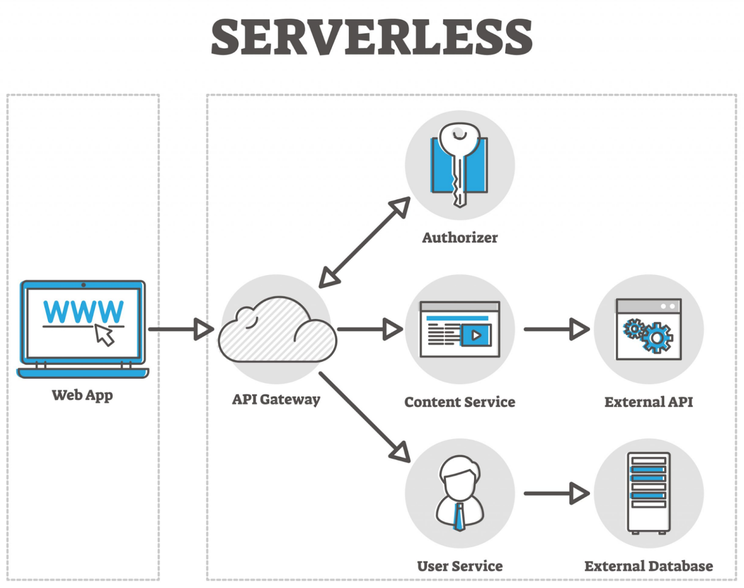
Flaws

- Increased risk of failure when communicating between services.
- A large number of services are difficult to manage.
- Requires addressing issues such as network latency, load balancing, and other challenges inherent in a distributed architecture.
- Needs comprehensive testing in a distributed environment.
- Implementation will take much longer.

Prominent real-life examples: Uber, Netflix, Amazon, eBay, and SoundCloud.

Serverless

Hire Java Developers



Serverless computing (or serverless technologies as they are sometimes called) is a promising cloud computing technology model that has emerged on the application development and architecture horizon in recent years. It is precisely because of the desire to capitalize on the huge potential of serverless frameworks that many major market players have been caught up in the general boom in cloud services.

Software industry giants such as Google, Microsoft, IBM, and Amazon are already offering their customers to apply the software architecture design best practices. Thanks to them you can migrate local business processes and achieve operational efficiency on flagship serverless platforms such as AWS Lambda and Azure Functions.

In simple terms, serverless architecture is an event-driven and query-driven technology solution that allows application developers to create efficient cloud environments with all the computing resources needed to run a seamless development process. Such frameworks are very convenient, especially in the context of tight deadlines and resource-intensive tasks.

Hire Java Developers

increase the effectiveness of other business process optimization practices, including DevOps and Agile.

Comparing serverless vs microservices architecture the former looks much more promising for application developers as it enables on-demand cloud workspaces. This means that serverless functions only run when a certain event is committed. After that, the functions perform a sequence of operations depending on the commands received from users. The serverless platform then applies a set of pre-prepared algorithms and rules, performs calculations, and provides up-to-date results.

Advantages

- Cost savings
- Increasing innovation potential
- Reduced time to market
- No need to select, protect, update or manage operating systems
- No need to choose the size of servers, monitor or scale them
- There is no risk of overpaying for extra resources
- There is no risk of performance degradation due to insufficient resources

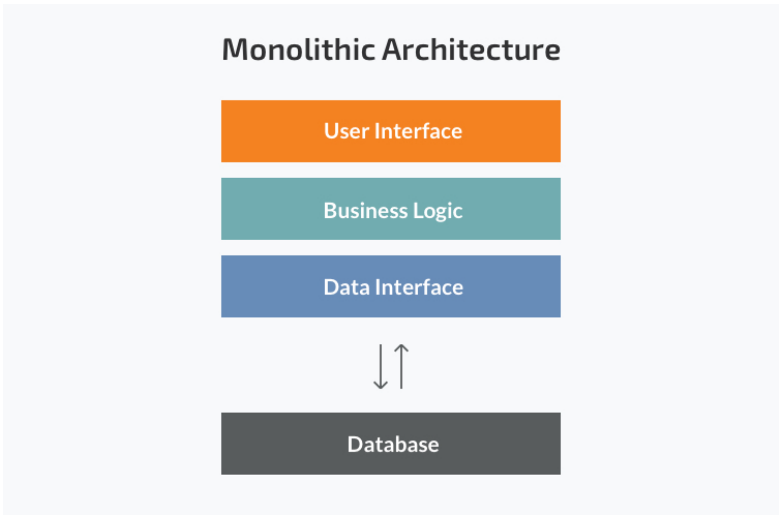
Flaws

On the other hand, the transfer of control of a part of the systems to a third party also gives rise to its drawbacks. For example, this may result in some loss of control and, for example, unexpected API upgrades. Also, splitting one application into several parts with various services woven into

Hire Java Developers

Prominent real-life examples: Netflix, Codepen, Zalora, Coca-Cola, and Nordstrom.

Monolithic



This approach is considered the best when building software products at startups. Many modern companies choose a monolithic architecture because this set of software architecture design best practices is comfortable when working with small groups of developers.

When using it, all components of the program are interrelated and interchangeable - this helps to develop the program as an autonomous and self-sufficient one. Monolithic architecture is considered traditional and proven in application development, but at the same time, many developers consider this approach to application implementation old-fashioned and no longer good for anything.

To understand whether monolithic or microservices are right for you, you need to consider the advantages and disadvantages of a monolithic architecture.

Hire Java Developers

- Cross-cutting problems are practically non-existent
- Improved performance

Flaws

- Large amount of code
- Difficult to modernize
- Flexibility is limited
- Dependency between components

Prominent real-life examples: SalesForce and Hubspot

Roles and resources needed to implement software architecture design best practices

All you need to do to build a decent software architecture is hire a team of engineers and UX/UI designers. Depending on a technology you may require:

- Ruby on Rails developers
- React coders
- React Native engineers
- etc.

For a more detailed team structure, one should consider attracting devs of different skill levels.

Hire Java Developers

An aspiring IT architect with less than a year of experience in the profession should already have the following skills:

- be able to collect requirements for software development;
- take part in the design of information systems;
- independently be able to design part of the architecture of services;
- prepare technical documentation;
- participate in the organization of final tests of software complexes.

Middle

More responsibility and independence are added to the duties of a specialist with 1-3 years of experience. Also, mid-level software architect has flexible skills that will help them manage a development team in the future.

So the programmer must:

- have the skills to describe the system architecture;
- have Enterprise, Solution, and Technical Architecture design skills;
- create architectural artifacts;
- be able to work with microservice architecture;
- have good communication skills.

Senior

A specialist with more than three years of experience already leads the team. They have developed managerial and client

Hire Java Developers

- knowledge of programming languages;
- study of project documents for completeness and absence of contradictions;
- analysis of decisions, making changes in the course of work;
- code quality control;
- quality control and deadlines.

Bottom line

The main purpose of architecture is to define the requirements that affect the structure of an application. A well-thought-out architecture reduces the business risk associated with building a technical solution and builds a bridge between business and technical requirements.

Although following the software architecture design best practices may seem expensive for a particular company, it will be much more costly to maintain the application and ensure the functionality of the functions.

Therefore it is advisable to deal with seasoned professionals (be it Flutter developers or JavaScript engineers) from day one.

Software Web Development Backend

Dawid Karczewski

Dawid is a full stack developer experienced in creating Ruby on Rails and React Native apps from naught to implementation.

Hire Java Developers