API First Approach:

API-first means that we must plan how the API is going to behave before we start coding the application. It also means that we will commit to a consistent and reusable API specification before coding the application. This means that the form of our application and the coding process will reflect the API. Although this requires additional collaboration between developers, API consumers, and key stakeholders, it helps ensure a more adoptable, developer-friendly, and compatible product.

APIs are essential to microservices-based app development. Therefore, the growing popularity of microservices-based apps has led to a corresponding rise in API-first development strategies

Following are the some of the benefits of API-First Approach:

1. In API first approach, developers create a contract which is used by all other teams for integration. This even makes development teams independent to work in parallel
2. API-first design also allows most problems to be solved before any code is even written which helps prevent problems when it is time to integrate APIs with applications
3. API first also makes it possible to add new services and technologies to applications without having to re-architect the entire system
4. Well-designed, well-documented, consistent APIs provide positive developer experiences because it’s easier to reuse code and onboard developers, and it reduces the learning curve.
5. API first reduces the risk of failure by ensuring that APIs are reliable, consistent, and easy for developers to use.

Here are few things that should be part of API first plan

1. Figure out the key services based on the business offers, figure out and list down all the use cases in the business. Write down potential endpoints based on those use cases
2. Should recognize all the stakeholders and allow stakeholders to weigh in on the design of the API. This makes integration smooth
3. Create an API contract, which established a set of standards and best practices for designing API
4. A comprehensive, cohesive style guide ensures consistency across the teams of an organization who are building services. API status codes, versioning, error handling, and more will be standardized ensuring that APIs are designed the same way
5. Peer code reviews can help ensure the API design standards were followed that developers are producing quality code
6. Use tools like OpenAPI to automate process like generating API documentation, style validation, API mocking and versioning. Provide interactive documentation so that developers can try out API endpoints
7. Implements a system that helps to track and manage API for reusability

As part of the application development, we can choose API first strategy to design and develop API, but it is not enough for a good API management for integration with other teams. API Gateway helps in API management and simplifies.

Why do we need API gateway?

Let’s say we have developed an e-commerce website, when user logs in and lands on home page it shows up with product catalogue, ratings of the products, deals of the day and many more features. All of these were developed as separate services, If the front-end development team want to render the page need to do separate calls to all the services involved in the page which will be a low performance as we have many network calls to backend services.

Usually a much better approach is to use what is known as an [API Gateway](http://microservices.io/patterns/apigateway.html). An API Gateway is a server that is the single-entry point into the system. It is an abstraction layer for API to expose to external world. It is like the [Facade](https://en.wikipedia.org/wiki/Facade_pattern) pattern from object‑oriented design. The API Gateway encapsulates the internal system architecture and provides an API that is tailored to each client. It might have other responsibilities such as authentication, monitoring, load balancing, caching, request shaping and management, and static response handling

Advantages:

1. A major benefit of using an API Gateway is that it encapsulates the internal structure of the application. Rather than having to invoke specific services, clients simply talk to the gateway
2. The API Gateway provides each kind of client with a specific API. This reduces the number of round trips between the client and application. It also simplifies the client code.

Disadvantages:

1. The API Gateway becomes a development bottleneck. Developers must update the API Gateway in order to expose each microservice’s endpoints
2. Need to handle the scenarios if the API Gateway goes offline, there are patterns to handle this scenario

Features if API Gateway:

Request Routing:

One of the key functions of an API gateway is request routing. API gateway implements some API operations by routing requests to the corresponding service. When it receives a request, the API gateway consults a routing map that specifies which service to route the request to

API Composition:

The API composition offers the simplest way to gather data from multiple microservices

And give response. The API gateway provides API composition which enables different client requests efficiently retrieve data using single API request.

Protocol Translation:

An API gateway also provide protocol translation. It might provide RESTful API to external clients, even though the application services use a mixture of protocols internally, including REST and gRPC. When needed, the implementation of some API operations translates between RESTful external API and internal gRPC based APIs.

Client-Specific API Response:

The problem with a single API is that different clients often have different requirements. For example, request API operation will return product data, user data, and rating data. In some case, not all clients need all the data. Let say mobile client only needs a subset of the data. The solution is the API gateway provides each client with its own API. For example, an API gateway provides different API response for android, iOS, and web client.

Diagram

Description automatically generated

API gateway must be reliable. One way to achieve it is to run multiple instances of gateway behind a load balancer. If one instance fails, the load balancer will route requests to other instances. Also when an API gateway invokes a service, there’s always a chance that the service is slow or unavailable. The solution is an API gateway use the Circuit breaker pattern when invoking service.