APPLICATION PROGRAMMING INTERFACES [API]

-Subhiksha.B M.Sc. [IT]

What is an API?

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

What is an Interface?

An interface is a device or a system that unrelated entities use to interact. According to this definition, a remote control is an interface between you and a television set, the English language is an interface between two people.

API has two sides.

- ❖ Consumer of API
- ❖ Producer of API.

Who provide an API to use in our application are called providers of API.

Example:

If you are going to create an application, in that application you want to use mapping technology. Instead of writing all the code to create a map in your application, you can just write a single code to call Google map's API and use the information which was brought by that API in your application.

Contract:

If the consumers want to use a particular API from the provider, they must have a technical contract between them. Without that contract, consumer can't use that API.

Example of API from Real World: Wall Socket

API- Wall socket

- Consumer-some devices like computer, hairdryer etc.
- Provider-Power stations.

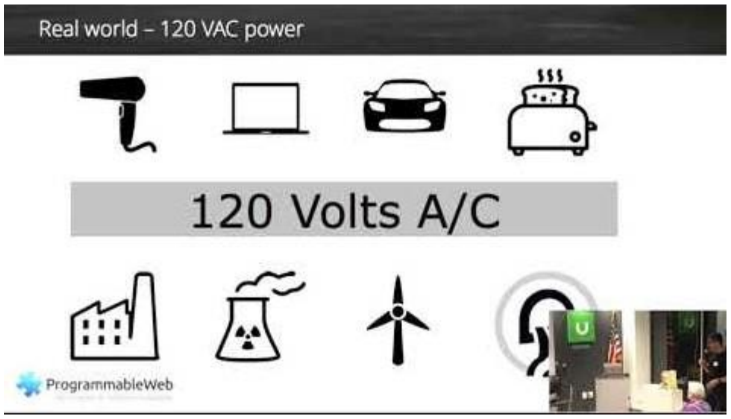
Whenever we plug any devices to the wall socket, devices uses the electricity provided by the power stations through the wall socket.

Wall Socket

FLEXIBILITY PROVIDED BY API

In the wall socket example, the consumer (devices like hair dryer, heater, desktop etc.) don't know about the type of electricity provided by the provider's

Side. The same situation also occurs in the consumer's side. The only contract between the consumers and provider of the API in this wall socket is 120 volts A/C.



TYPES OF CONSUMERS:

- ❖ Web apps
- ❖ Server apps
- ❖ Mobile apps
- ❖ Desktop apps
- ❖ Devices [On the IOT]. Sometimes these will also act as providers of an API.

API DRIVE ORGANISATIONAL CULTURE

MONOLITHIC APPLICATION:

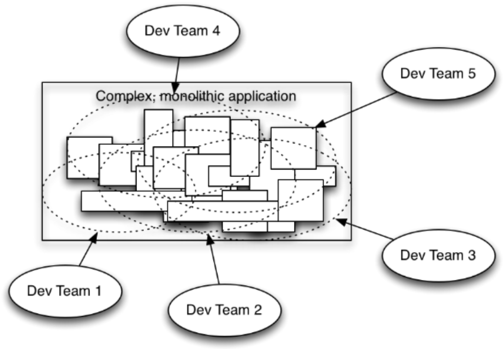
It is a traditional application which is complex, highly intensive, and large.

Parts inside the monolithic application are overlapping with each other.

Whenever changes occur in a particular part inside a monolithic application, the changes in the part doesn't know what impact is going to happen in the monolithic application.

Development team work on the same part on the same time. It will create breakage on the application. It may reduce the speed of the application. Conflicting source code will give undesirable outcomes to the application .To avoid these problems, decompose the parts inside the monolithic application. Each parts are around by a layer called API. This infrastructure is called service oriented infrastructure.

Example: Google Map's API.

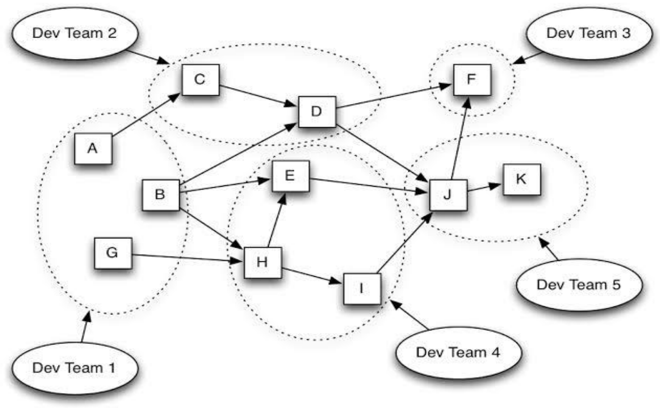


MONOLITHIC APPLICATION

In 2002,CEO of Amazon announced to his company that all team will expose their data and functionality through service interfaces . Teams must communicate with each other through these interfaces. He also announced that their information’s are available to developers in the outside world. By doing this , Amazon is the biggest API provider in the planet.

MICROSERVICES ARCHITECTURE:

Pattern that allows software to be developed into relatively small , distinct components. Each component is abstracted by an API and provides a distinct subset of the functionality of the entire application.



MICROSERVICES ARCHITECTURE

MICROSERVICES PROS:

- ❖ Scaling
- ❖ Team assignment and focus
- ❖ Complexity localization
- ❖ Reusability.

TYPES OF API:

- ❖ Web\Network API
- ❖ Product API
- ❖ Standard API
- ❖ System/Embedded API.

API SCOPES:

- ❖ Single purpose API
- ❖ Aggregate API
- ❖ Micro services API.

ADDRESSING IT CONCERNS:

- ❖ Efficiency\Reusability
  - Productivity
  - Developers write 1 line of code vs 10,000 lines of code.
  - Cost savings
  - Less wastage.
- ❖ Flexibility
  - Language independence
  - Scalability
  - Performance.
- ❖ CLIENT SECURITY RISKS
  - App source code exposure
  - Shared passwords.
- ❖ BACKEND SECURITY RISKS
  - Database exposure
  - Rate limiting
  - Improperly secured endpoints.
  - Clear text data at rest.
- ❖ NETWORK SECURITY RISKS
  - Credential theft.
- ❖ Simplification.

THANK YOU