Backend APIs and Testing

Riya Dennis

(Adarga)

A bit about me:

I am Riya Dennis, currently working as a Senior Software Engineer at Adarga.

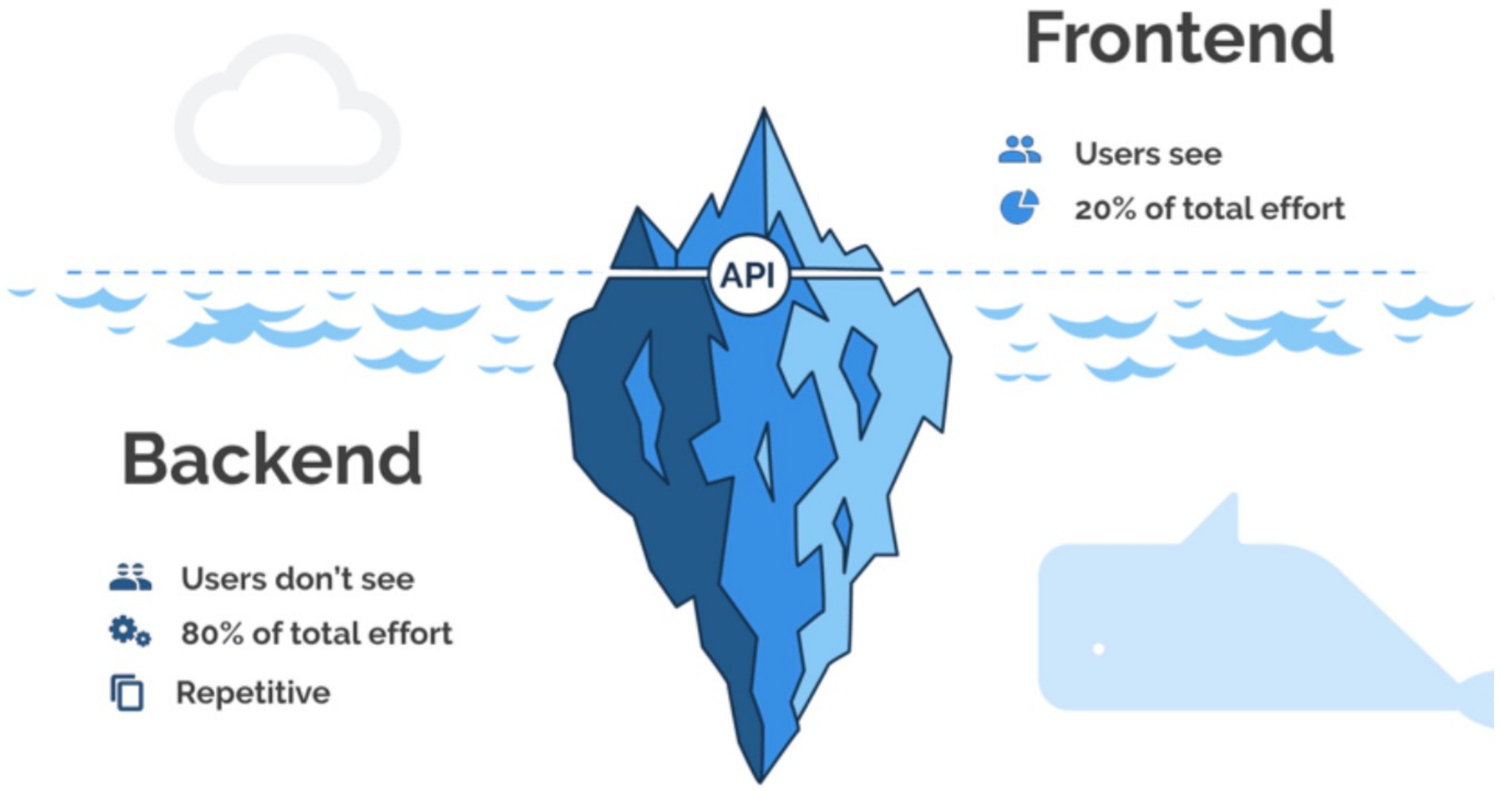
With over 12 years of experience in designing and developing backend systems, I have had the privilege of working in feature teams and collaborating closely with product, quality, and infrastructure teams.

Throughout my career, I have gained extensive expertise in developing backend systems for web and mobile applications using a wide range of technologies.

LinkedIn:

https://www.linkedin.com/in/riya-dennis-b631a026/

Backend is any part of a software that users do not see



If front end is the skin of a software application, backend is the meat and bones that is keeping it up and running.

Backend Systems’ Architecture

We have two approaches to design backend systems:

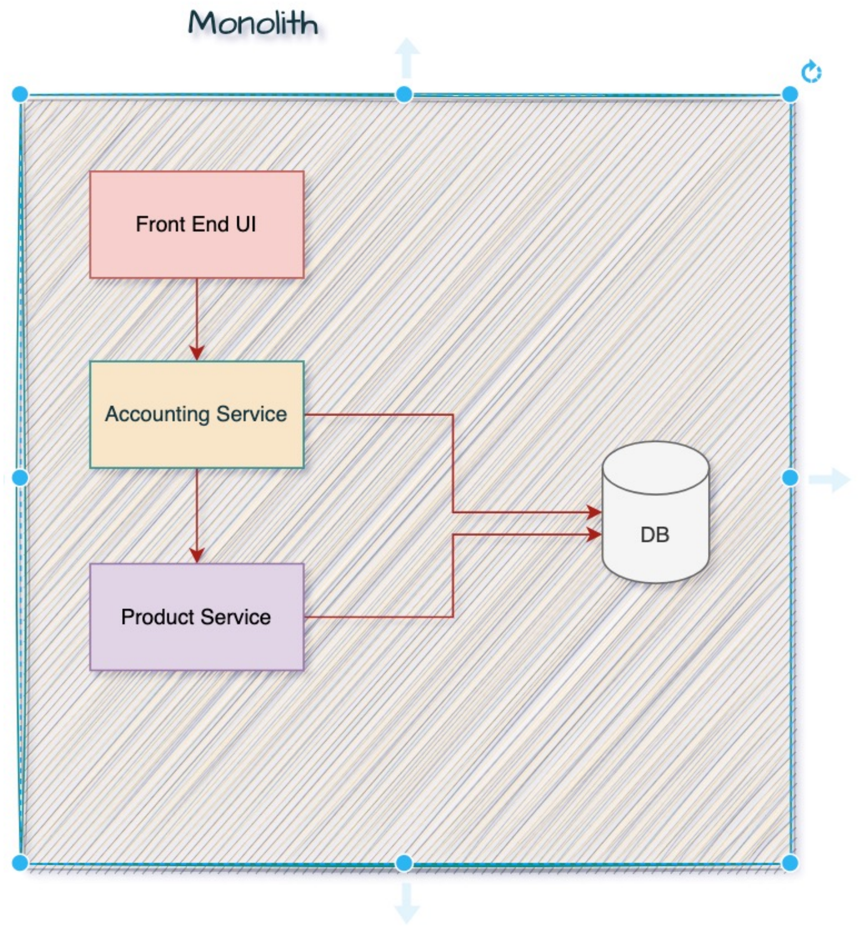
- Monoliths
- Microservices

The decision to choose which approach depends heavily on your organization structure and user requirements.

For smaller applications monoliths still will work as a viable solution. We need to always consider whether the complexity of microservices is worth the effort.

Monolithic Architecture

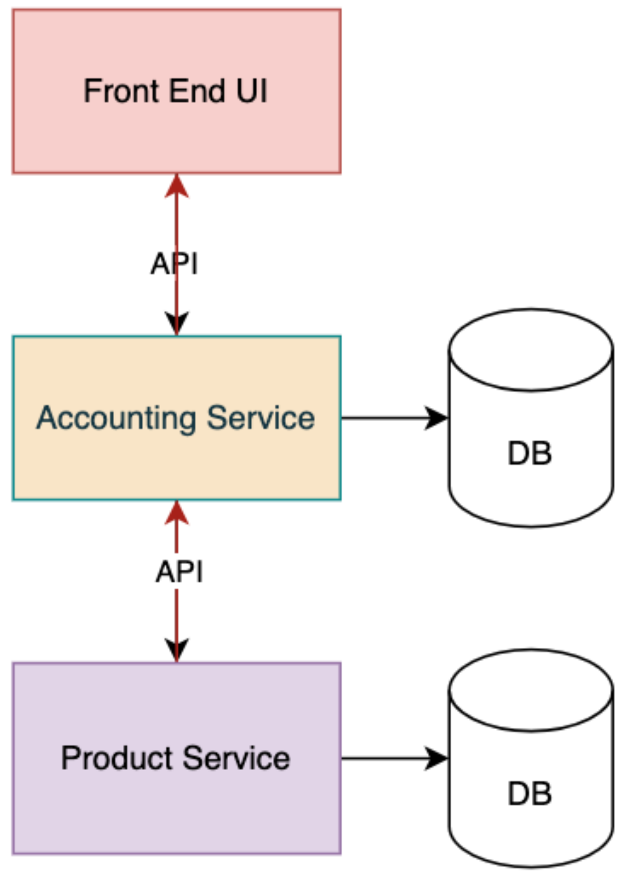
- Monolithic Architecture:
  - Structures the application as a single deployable/executable component that shares a single database.
  - The component contains all the application's subdomains.
  - Since there is only a single component, all operations are local.



Microservices Architecture (Synchronous)

Microservices is an architectural style that structures an application as a collection of services that are:

- Organized around business communication lines.
- Owned by a small team.
- Loosely coupled.
- Independently deployable.



Microservices use API’s

- **API (**Application Programming Interface) is the *doorways* or **frameworks** that allow data exchange between services.
- API defines a contract between data provider and consumer.
- API should be shared with rest of the organisation so that other teams can track the changes.
- There are different strategies to do this all of them should enable seamless communication between different teams.
- The changes should be forward and backward compatible.

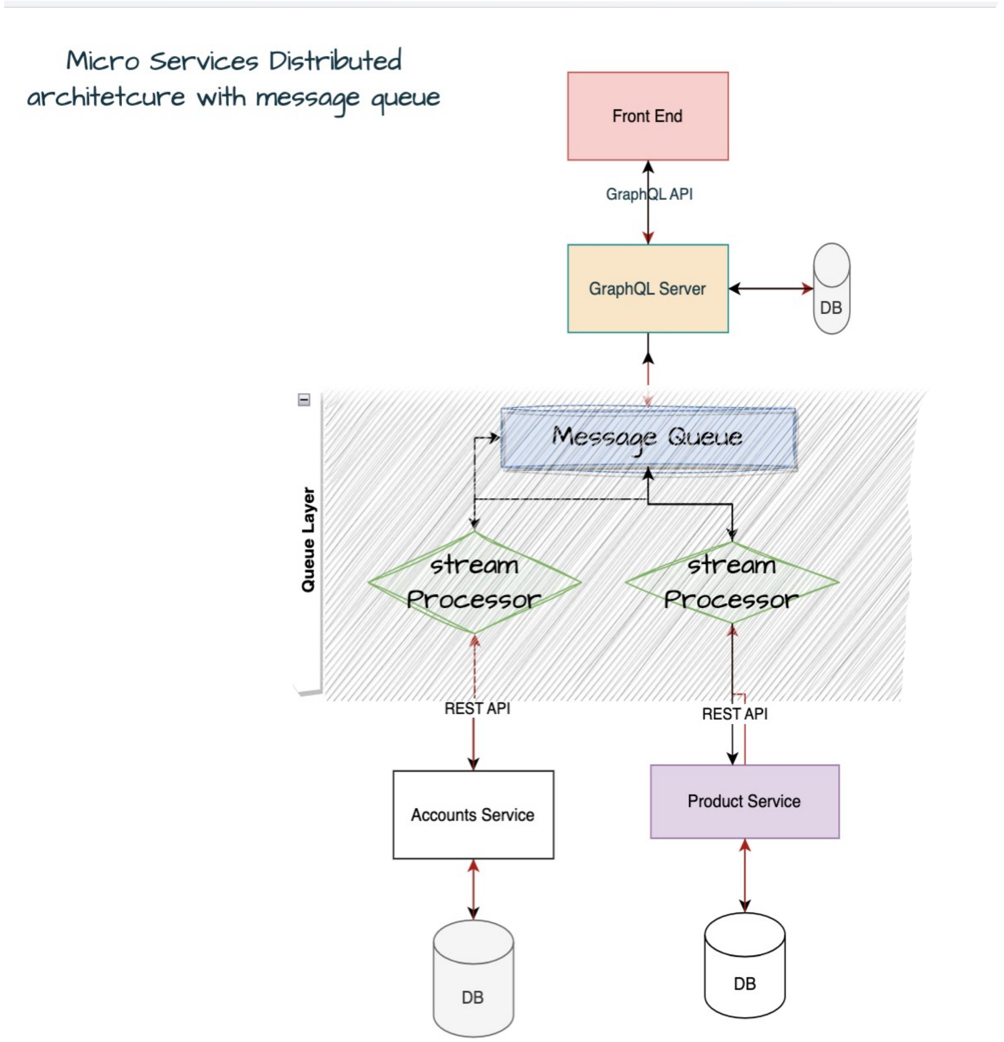
Asynchronous Messaging

There are two basic message patterns that microservices can use to communicate:

- Synchronous
- Asynchronous

Synchronous messaging the caller will wait for the response from the receiver.

In **Asynchronous** messaging a service sends a message and do not wait for a response. One or more services will process the message asynchronously



Types of API

- GraphQL
- REST

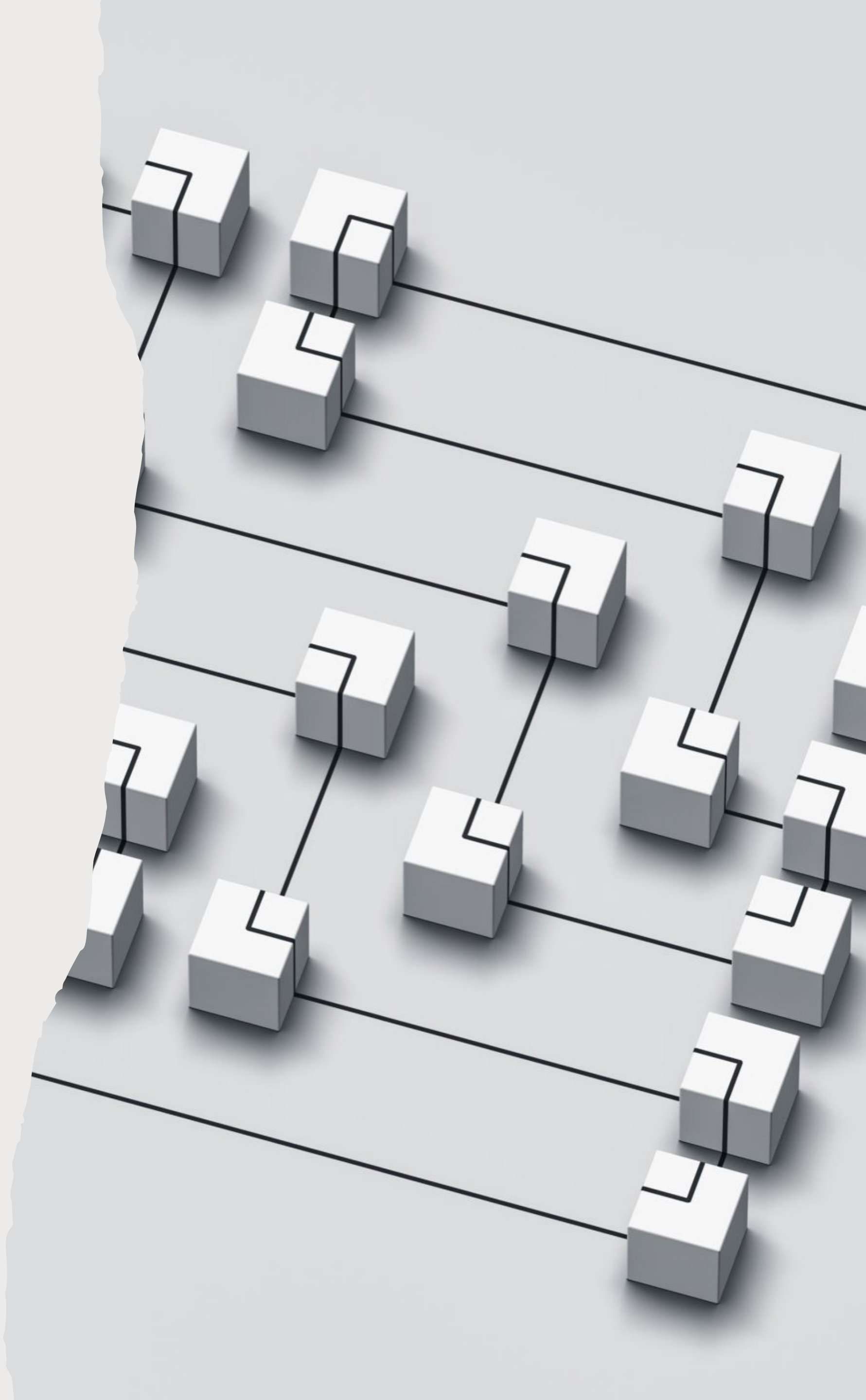
{ REST API }

GraphQL

- is a query language for APIs
- is a runtime for fulfilling those queries with your existing data.
- provides a complete and understandable description of the data in your API
- gives clients the power to ask for exactly what they need and nothing more.
- released by Facebook in 2015

Queries and Mutations

- GraphQL queries can traverse related objects and their fields, letting clients **fetch** lots of related data in one request, instead of making several roundtrips as one would need in a classic REST architecture
- Mutation is the right convention to send requests to **modify** or add server-side data.



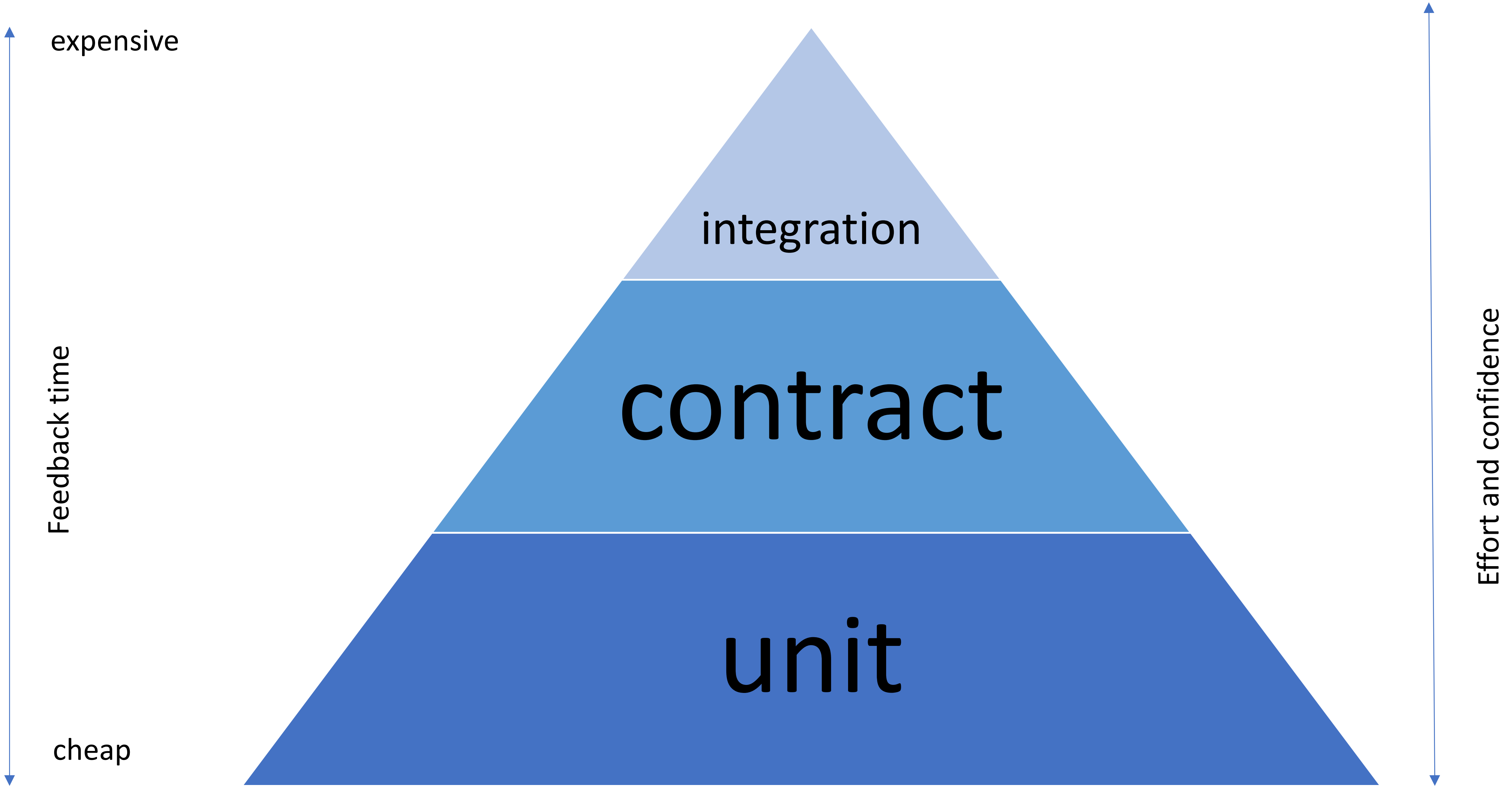
REST (Representational State Transfer)

- REST was introduced in 2000 and is been around for many years which makes the ecosystem more stable with lots of tooling and support
- Services that implement this architecture are call RESTful services.
- There are lots of free RESTful API's available for you to access and play with like :

https://rapidapi.com/

Creating REST API's is also easy and simple

API test pyramid in the demo



Tests in the demo explained

UNIT TESTS ARE WRITTEN TO TEST THE BASIC COMPONENTS OF A SERVICE THIS IS ACHIEVED BY MOCKING OTHER DEPENDENCIES LIKE DATABASE OR MESSAGE QUEUE.

CONTRACT TEST FOCUS ON INTERFACES AND INTEGRATIONS, DEPENDENCIES ARE MOCKED LIKE IN THE UNIT TESTS.

INTEGRATION TESTS WILL BRING UP ALL THE DEPENDENCIES OF THE SERVICE AND WILL INTERACT WITH TO TEST DIFFERENT SCENARIOS.

Thank you

Link for demo app in git hub:

https://github.com/riyadennis/sigist