mukul rajpoot

WEB ARCHITECTURE BASICS

PART-2

mukul rajpoot

DIFFERENT TIERS IN WEB ARCHITECTURE

Think of a tier as a logical separation of components in an application or a service. This separation is at a component level, not the code level.

What do I mean by components?

- Database
- Backend application server
- User interface
- Messaging
- Caching etc.

These are the components that make up a web Application.

There are mainly 4 tiers:

- Single Tier
- Two-Tier
- Three-Tier
- N-Tier

mukul rajpoot

SINGLE-TIER APPLICATIONS

In a single-tier application, the user interface, backend business logic, and the database reside in the same machine.

Typical examples of single-tier applications are desktop applications like MS Office, PC Games, image editing software like Gimp, Photoshop, etc.

Upside of Single-Tier:

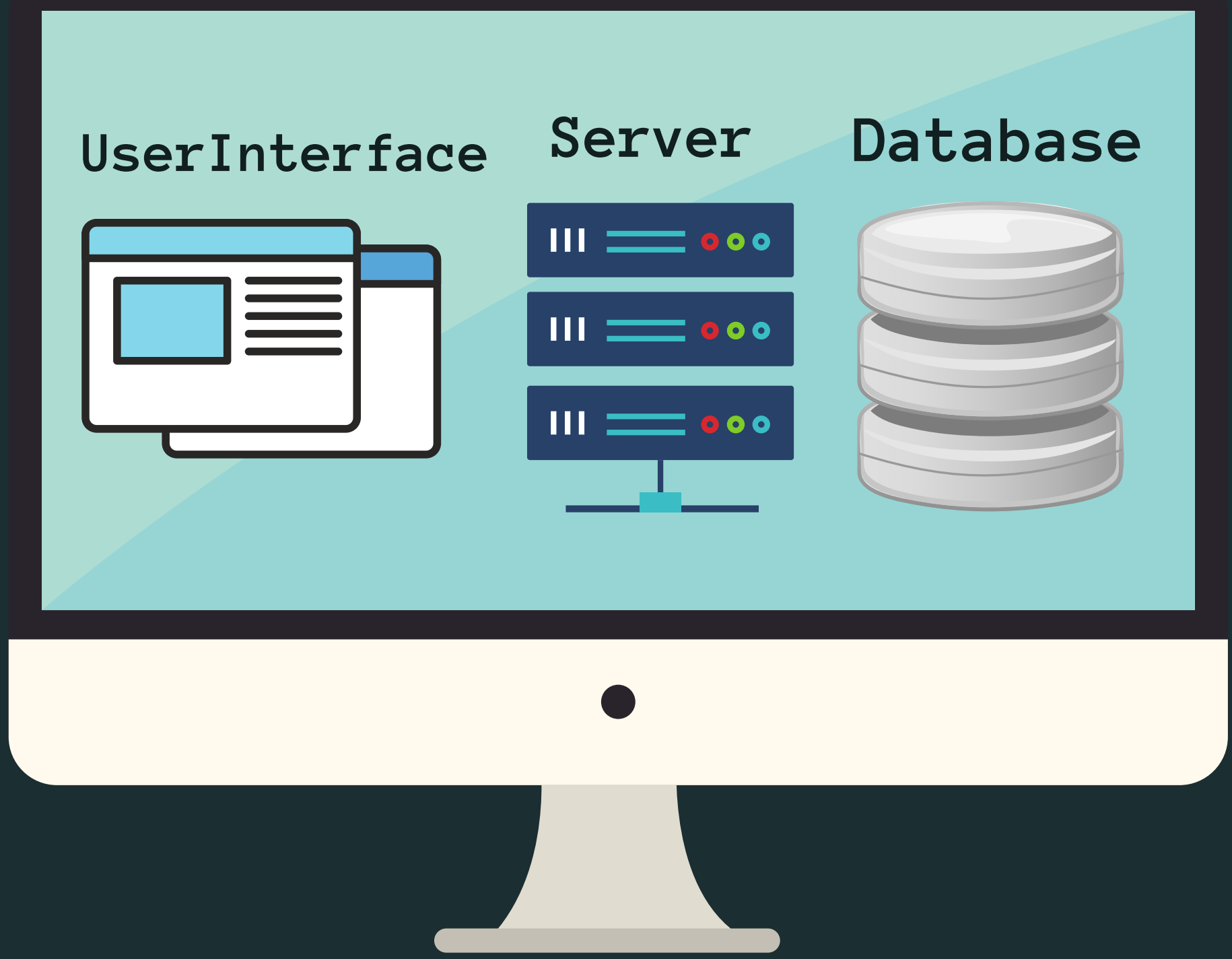
- No Network Latency
- Data is highly secured

Downside of Single-Tier:

- Application publisher has no control after shipping
- The code is also vulnerable to being tweaked and reversed engineered.

mukul rajpoot

SINGLE-TIER APPLICATIONS



mukul rajpoot

TWO-TIER APPLICATIONS

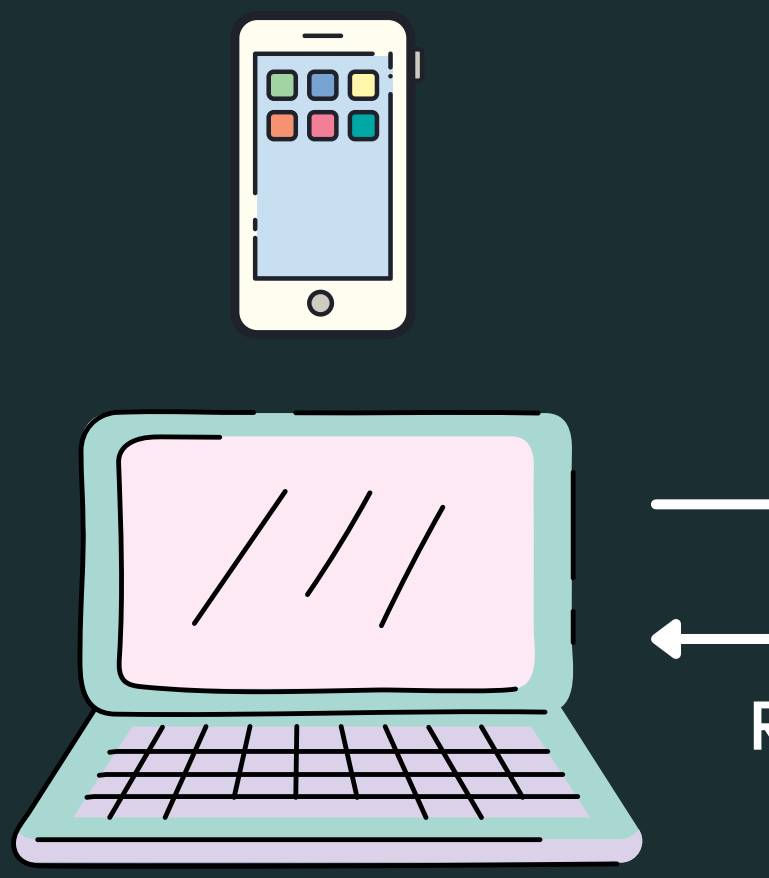
A two-tier application involves a client and a server. The client contains the user interface with the business logic in one machine. Meanwhile, the backend server includes the database running on a different machine. The database server is hosted by the business and has control over it.

Taking a to-do list app as an example, the application makes a call to the database server only when the user has finished creating their list and wants to persist the data.

Another good example of two-tier apps is mobile app-based games like BGMI (PUBG). The game files are pretty heavy, and they only get downloaded on the client once when the user uses the application for the first time. And they make the network calls to the backend only to persist the game state.

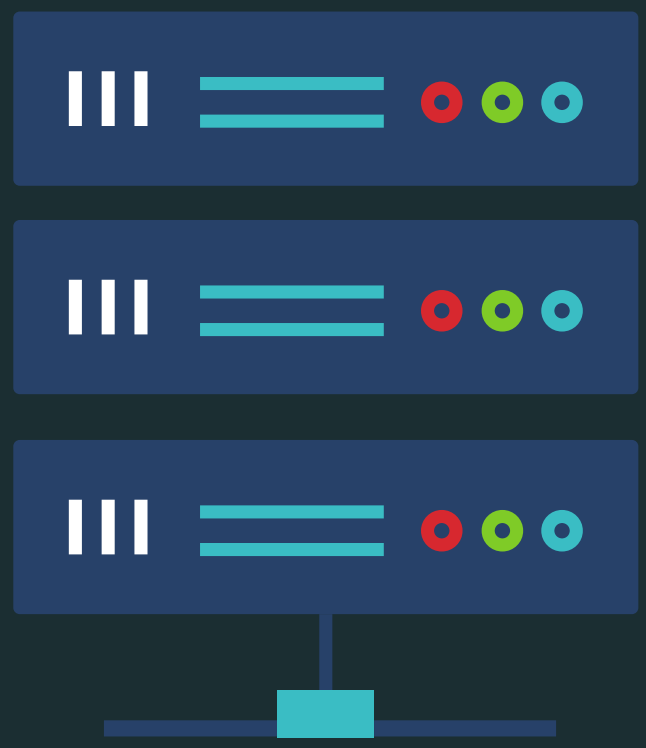
TWO-TIER APPLICATIONS

mukul rajpoot



Request

Response



mukul rajpoot

THREE-TIER APPLICATIONS

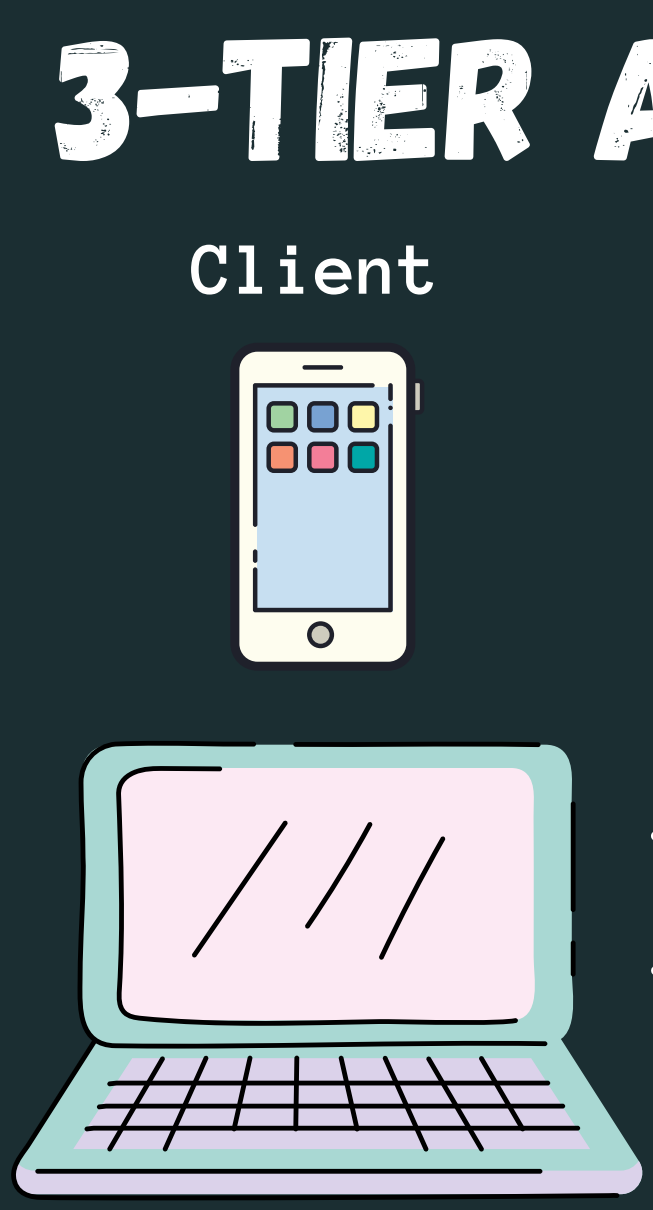
Three-tier applications are pretty popular and largely used on the web. Almost all simple websites like blogs, news websites, etc., are part of this category.

In a three-tier application, the user interface, business logic, and the database all reside on different machines and, thus, have different tiers. They are physically separated.

Let’s take the example of a simple Hashnode blog. The user interface will be written using Next.js and CSS and the backend application logic will run on a server like Node.js and the database will be MongoDB. A three-tier architecture works best for simple use cases.

3-TIER APPLICATIONS

munkul rajpoot



mukul rajpoot

N-TIER APPLICATIONS

An n-tier application is an application that has more than three components like

- Cache
- Message queues for asynchronous behavior
- Load balancers
- Search servers for searching through massive amounts of data
- Components involved in processing massive amounts of data
- Components running heterogeneous tech commonly known as web services, microservices, etc.

All the social applications like Instagram, Facebook, TikTok, large-scale consumer services like Uber, Airbnb, online massive multiplayer games like PUBG etc., are n-tier applications. n-tier applications are more popularly known as distributed systems.

mukul rajpoot

WHAT'S THE NEED FOR SO MANY TIERS?

Two software design principles that are key to explaining this are the single responsibility principle and the separation of concerns.

Single responsibility principle means giving one dedicated responsibility to a component and letting it execute it flawlessly, be it saving data, running the application logic or ensuring the delivery of the messages throughout the system.

Separation of concerns loosely means the same thing, be concerned about your work only and stop worrying about the rest of the stuff. These principles act at all the levels of the service, be it the tier level or the code level.

HAPPY LEARNING

mukul rajpoot