Service Application Architectures

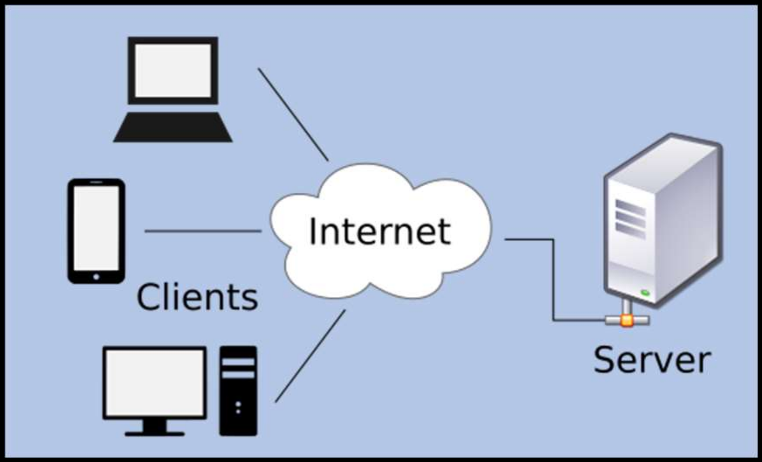
What do we mean by Service?

- A **Service** is a running application or process accessible by users and other applications.
- Email is an example of a **Service Application**.
- Most services are complex, consisting of multiple dependent services.
- For example to offer a web-based Email service application like Gmail you need these dependent services at minimum:
  - Web server (HTTP)
  - Message transfer agent (SMTP)
  - Message Store (IMAP)

What is Application Architecture?

- Defines how the workloads of a service are partitioned or subdivided over a network.
- This is done so that:
  - More than one client can access the service
  - The workload can be distributed to achieve better performance

- **Also Known As…**
- Multitier Architecture
- Multilayered Architecture
- Client-Server Architecture



Layers of An Application

Presentation

Code and layout responsible for the user interface

Business Logic

Transformational logic at the heart of what the application actually does.

Data Access

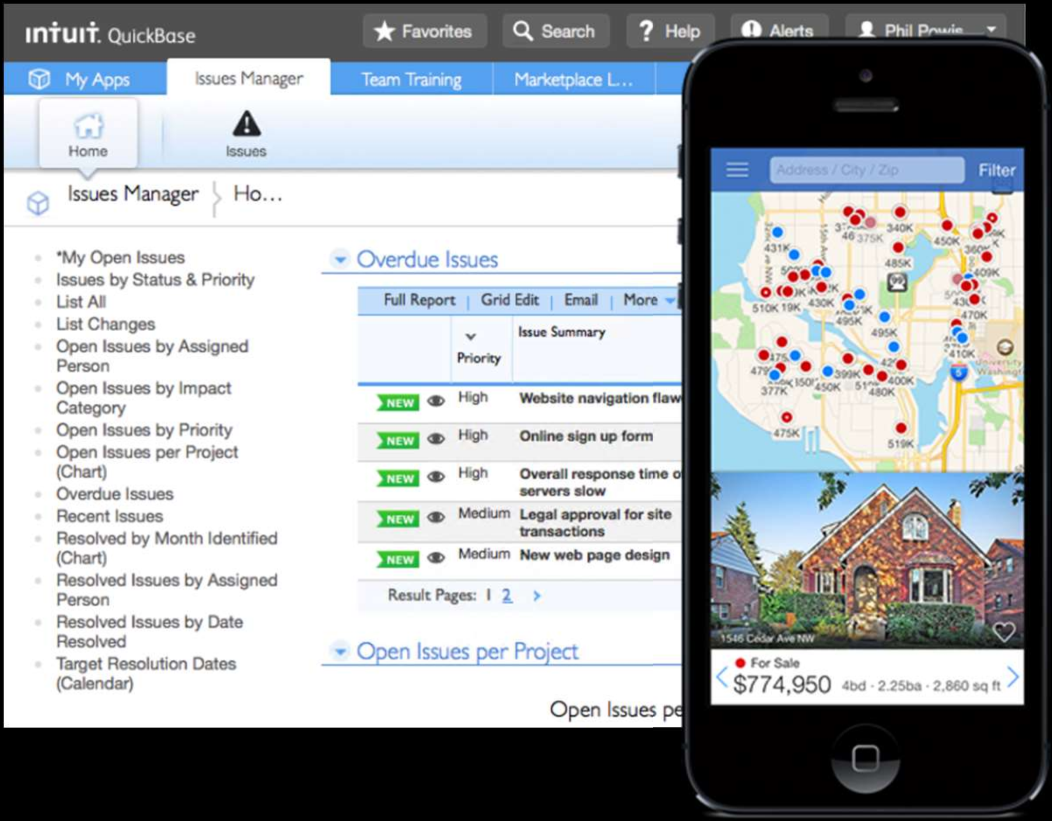
Create, Read, Update and Delete (CRUD) operations

Data

Data storage and retrieval of Relevant Data.

Presentation Layer

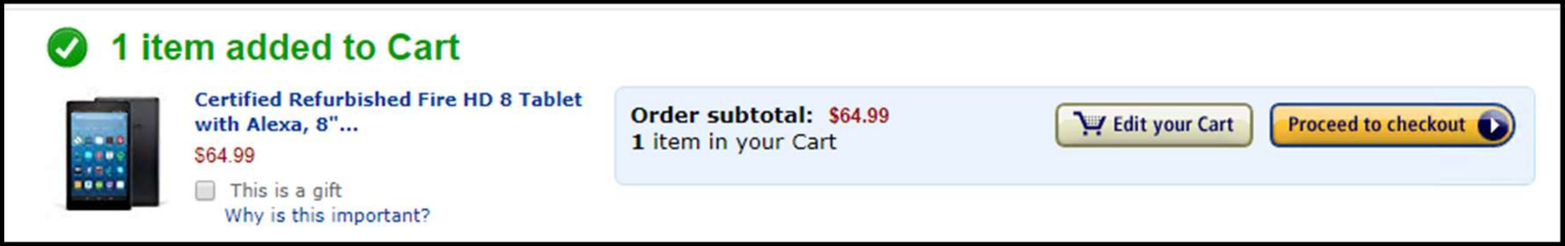
- Code which addresses User interface concerns
- Web: HTML and CSS
- Mobile: Xcode interface builder / Android studio
- Windows / Mac / Linux: Varies



	Priority	Issue Summary
NEW	High	Website navigation flaw
NEW	High	Online sign up form
NEW	High	Overall response time o servers slow
NEW	Medium	Legal approval for site transactions
NEW	Medium	New web page design

Business Logic Layer

- Code to address the Transformational Logic of the application; part of the application's functionality
- Written in a programming language: Java, JavaScript, Python, C#, etc…


1 item added to Cart

Certified Refurbished Fire HD 8 Tablet with Alexa, 8"...

$64.99

This is a gift

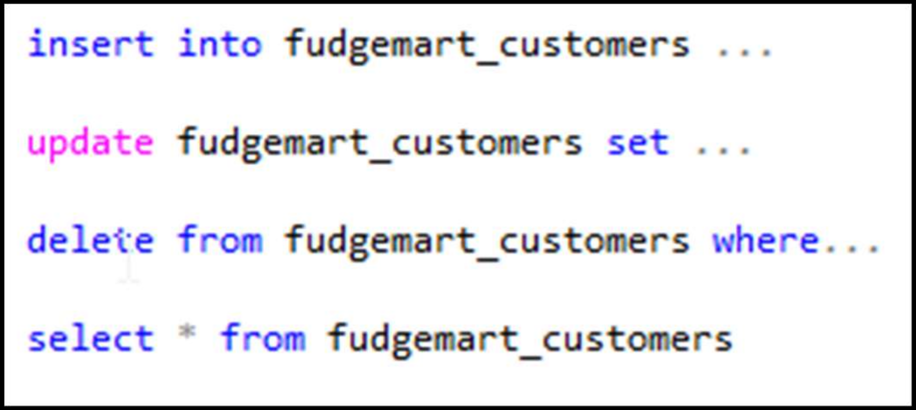
Why is this important?

Order subtotal: $64.99

1 item in your Cart

Data Access Layer

- Responsible for CRUD (Create, Read, Update, Delete) Operations
- Code which transforms operations into the DSL (Domain specific Language) to communicate with the database. (Typically SQL).



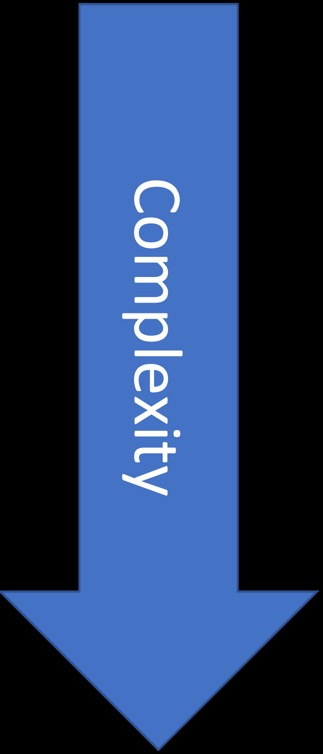
```
insert into fudgemart_customers ...

update fudgemart_customers set ...

delete from fudgemart_customers where...

select * from fudgemart_customers
```

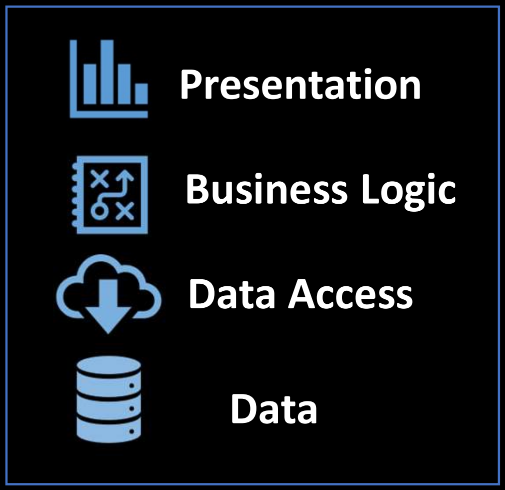
Types of Application Architectures



1. Monolithic
2. Monolithic over distributed storage
3. Two-tier thin client
4. Two-tier fat-client
5. Three Tier
6. N-Tier
7. Enterprise Service Bus
8. Micro Services

A Monolithic Application

- All layers within a single system
- Simplest design
- Single-User. Single Site. No Scale.
- Multiple uses, multiple instances.
- Example: MS Word, application on your phone.

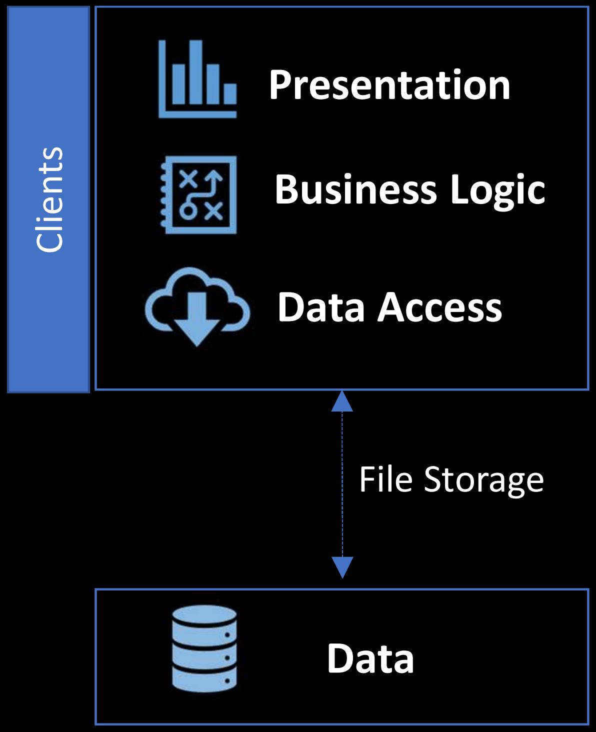


Example: Monolithic Application

- Can two people work on the same PowerPoint file at the same time? No!
- Can everyone in your group edit the same word document at the same time? No!

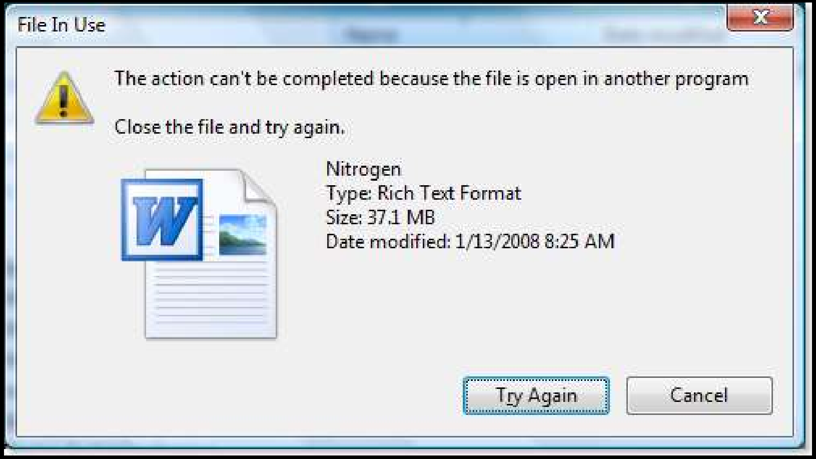
Monolithic over Distributed Storage

- Data storage is over a network but the rest of the application is monolithic.
- Single-user multi-site.
- Example: Two people share a word document over Google Drive.



Classic Issue With Monolithic + Storage

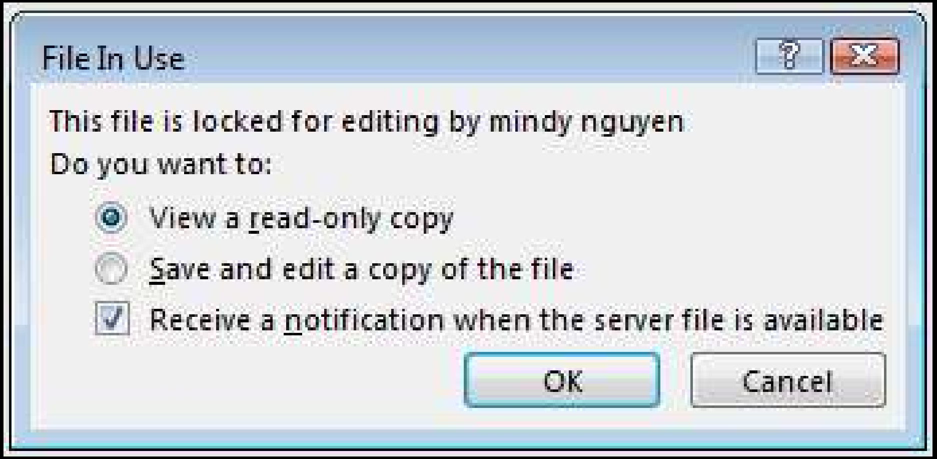
- Concurrency was not built into the application!


File In Use

The action can't be completed because the file is open in another program

Close the file and try again.

Nitrogen
Type: Rich Text Format
Size: 37.1 MB
Date modified: 1/13/2008 8:25 AM


File In Use

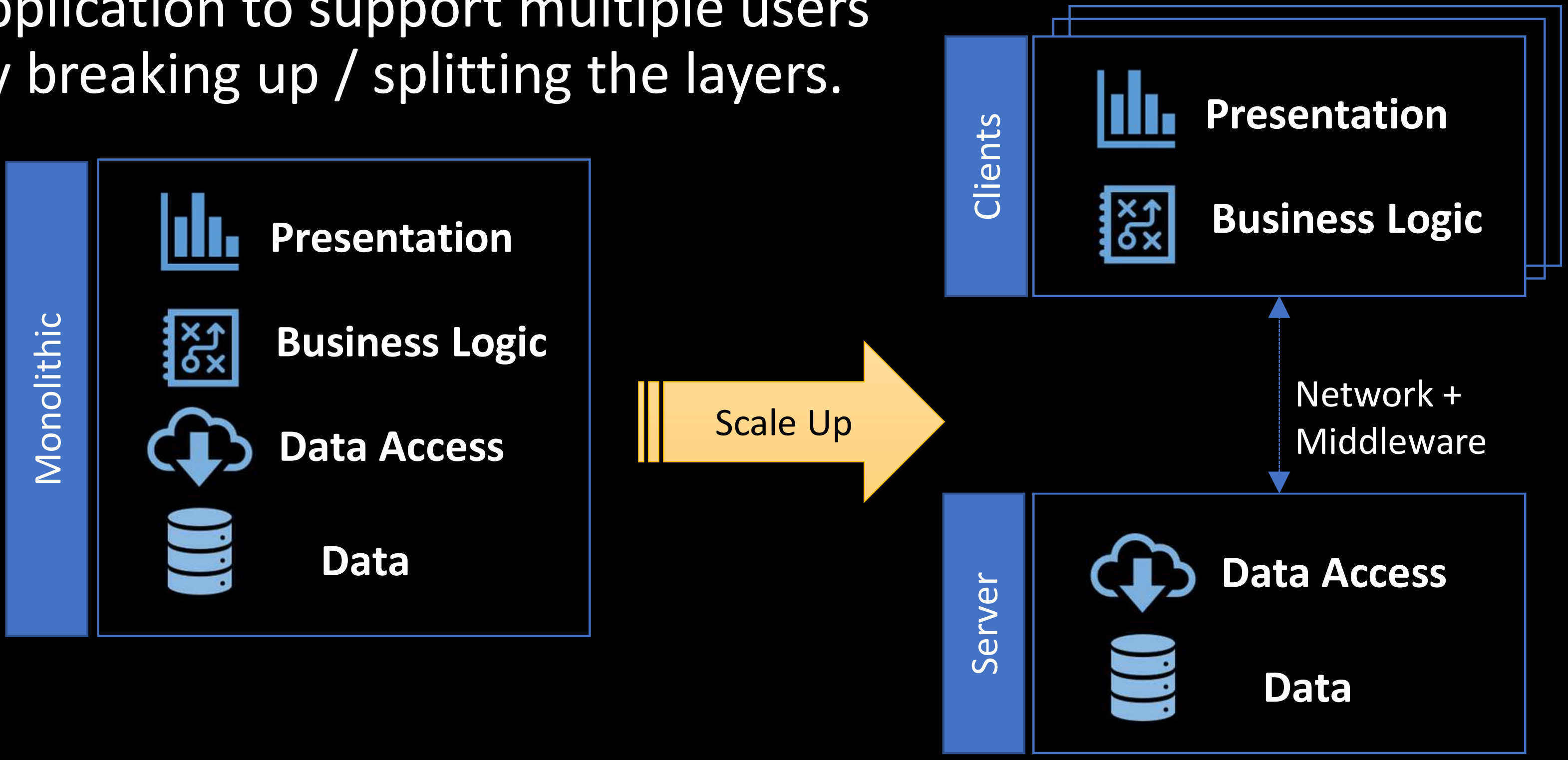
This file is locked for editing by mindy nguyen

Do you want to:

[✓ View a read-only copy]
Save and edit a copy of the file
[✓ Receive a notification when the server file is available]

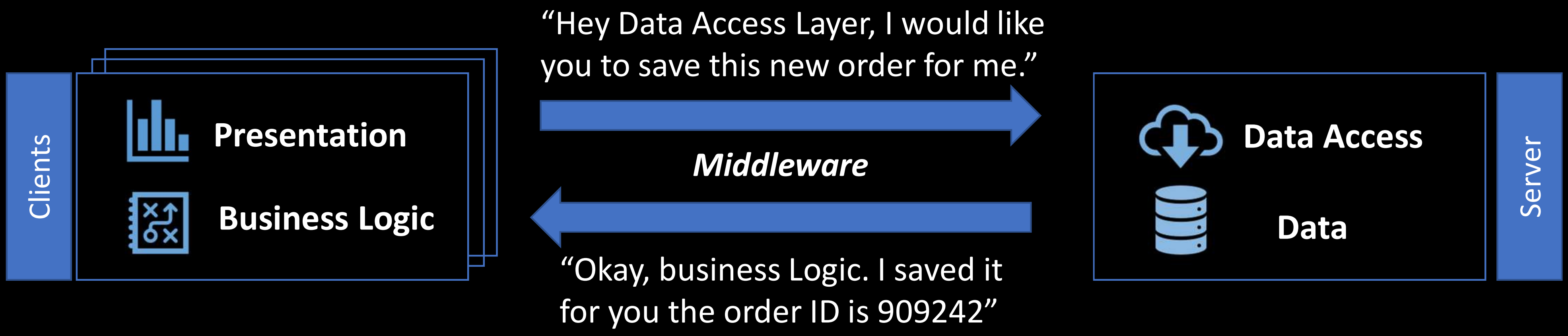
Scaling Up

- We re-design / program the application to support multiple users by breaking up / splitting the layers.



Breaking Up Is Hard To Do!

- When you separate an application into parts and run them separately they now require a way to communicate with each other!


Clients
Presentation
Business Logic
"Hey Data Access Layer, I would like you to save this new order for me."
Middleware
"Okay, business Logic. I saved it for you the order ID is 909242"
Data Access
Data
Server

Middleware

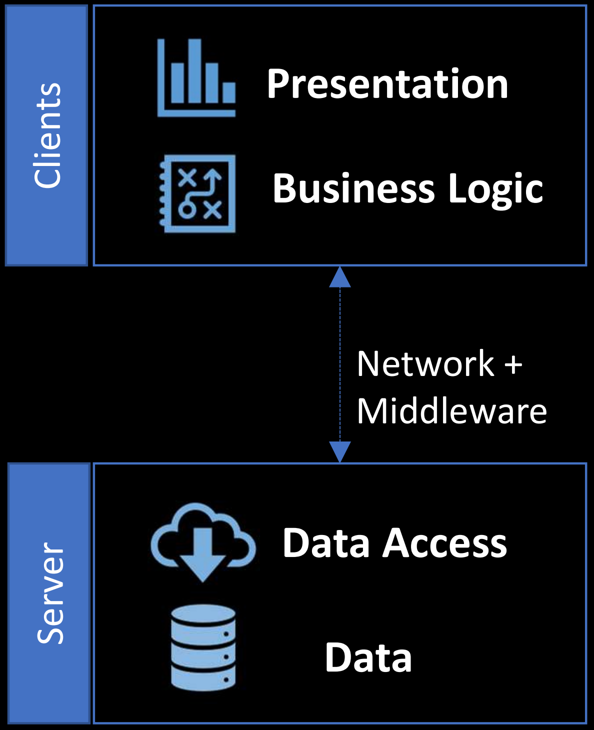
- Middleware is software which provides inter-process communications between the layers of an application.
- It is required whenever an application is split into layers over a network.
- There are different types of middleware for access between the different layers.

Middleware Alphabet Soup

Name	What is it / What it does
ODBC (open database connectivity)	Database management systems access
CORBA (common object request broker architecture)	Business logic procedure call and data exchange
REST (representational state transfer)	A "pattern" that uses HTTP protocol for business logic / data exchange. Foundation of most web API's
SOAP (simple object access protocol)	Like REST but more overhead / payload.
ODATA (open data protocol)	A spec for CRUD over HTTP using the REST patters

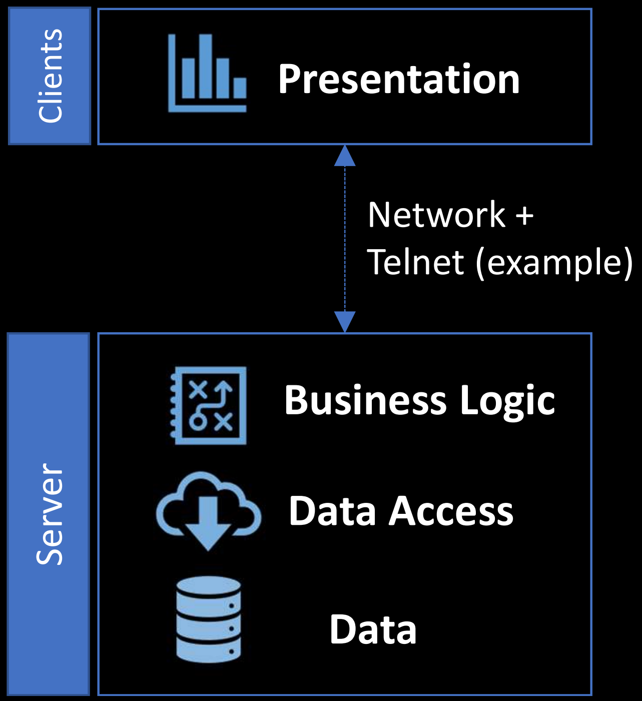
2-Tier Client/Server

- In 2 Tier client/server the application is split in two parts separated by one layer of middleware.
- This makes the application multi-user and multi-Site.
- Common architecture in the pre-consumer Internet era of the 90's.

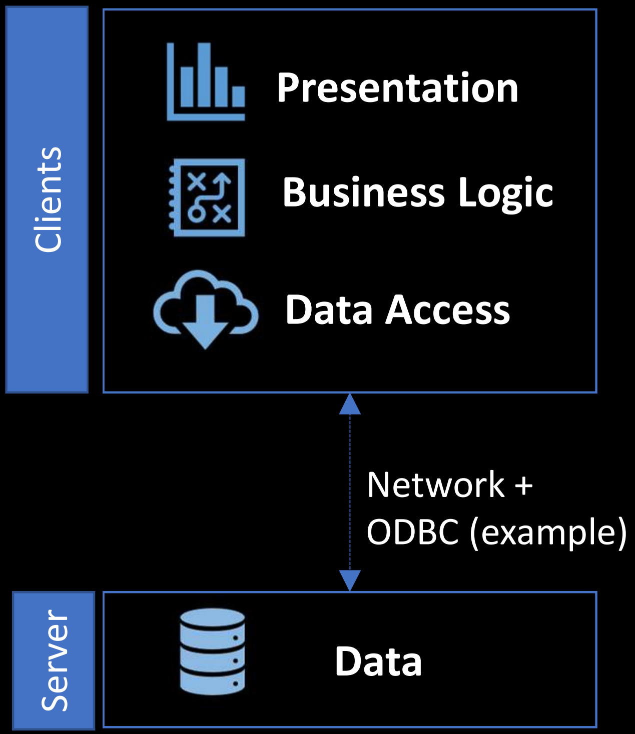


Where is the Business Logic?

- "Thin Client"



- "Fat Client"

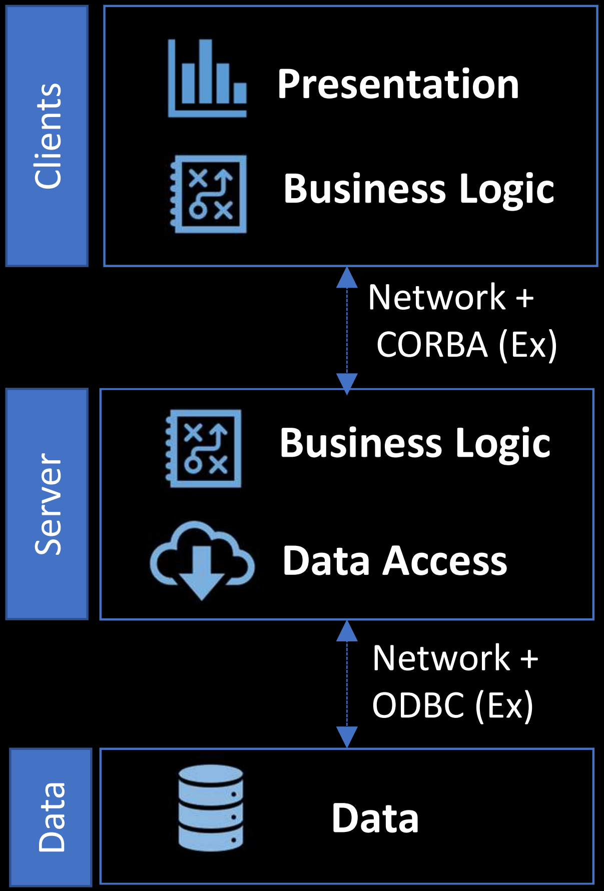


Thin-Client, Fat-Client examples

- Fat Clients:
  - The application itself must be installed before you can use it.
  - Playing a game like Madden or Fortnite
  - Microsoft Outlook for Email
- Thin Clients:
  - Nothing needs to be installed for specific to the application for it to be used.
  - Playing a game in your Web Browser
  - Gmail or Yahoo Mail

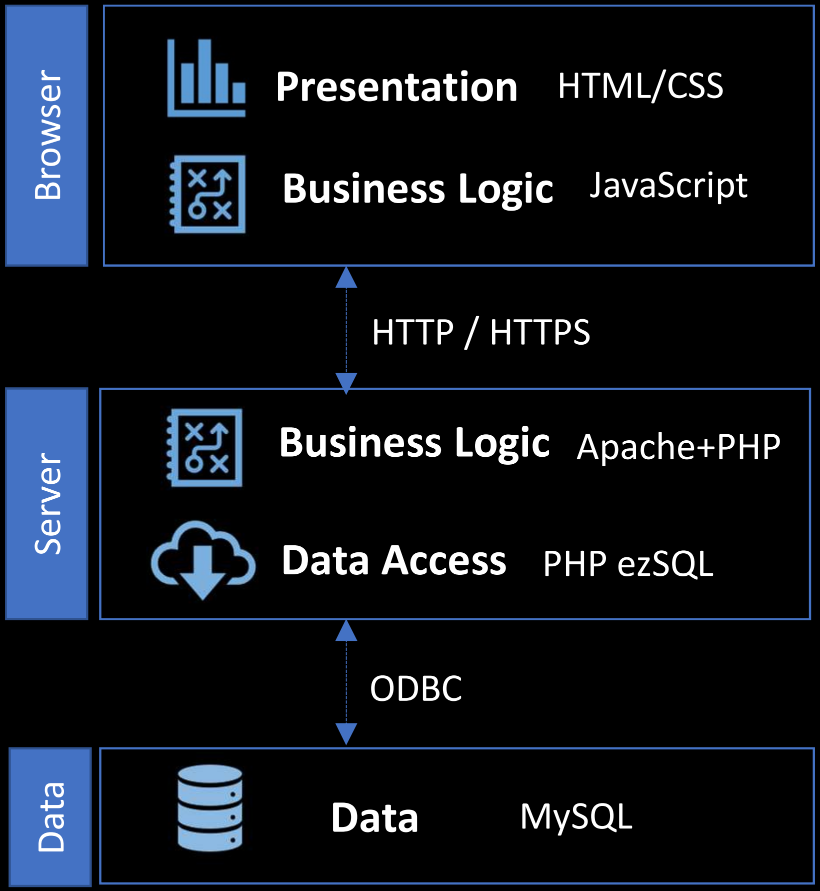
3-Tier Client/Server

- In 3 Tier client/server the application is split into 3 parts. Typically with a business logic and data access layer in the middle tier.
- Multi-user, Multi-Site.
- Scales Vertically better than 2-tier
- Majority of business logic is on the server
- Common architecture during the Internet boom.



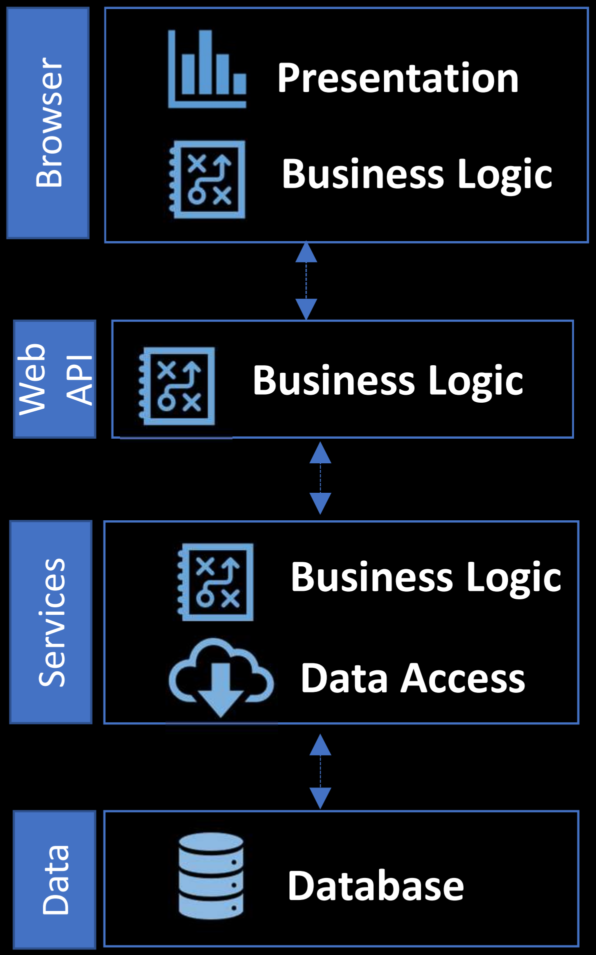
Web 3-Tier Example (Wordpress)

- Here’s is how the popular web application WordPress is architected.



N-Tier

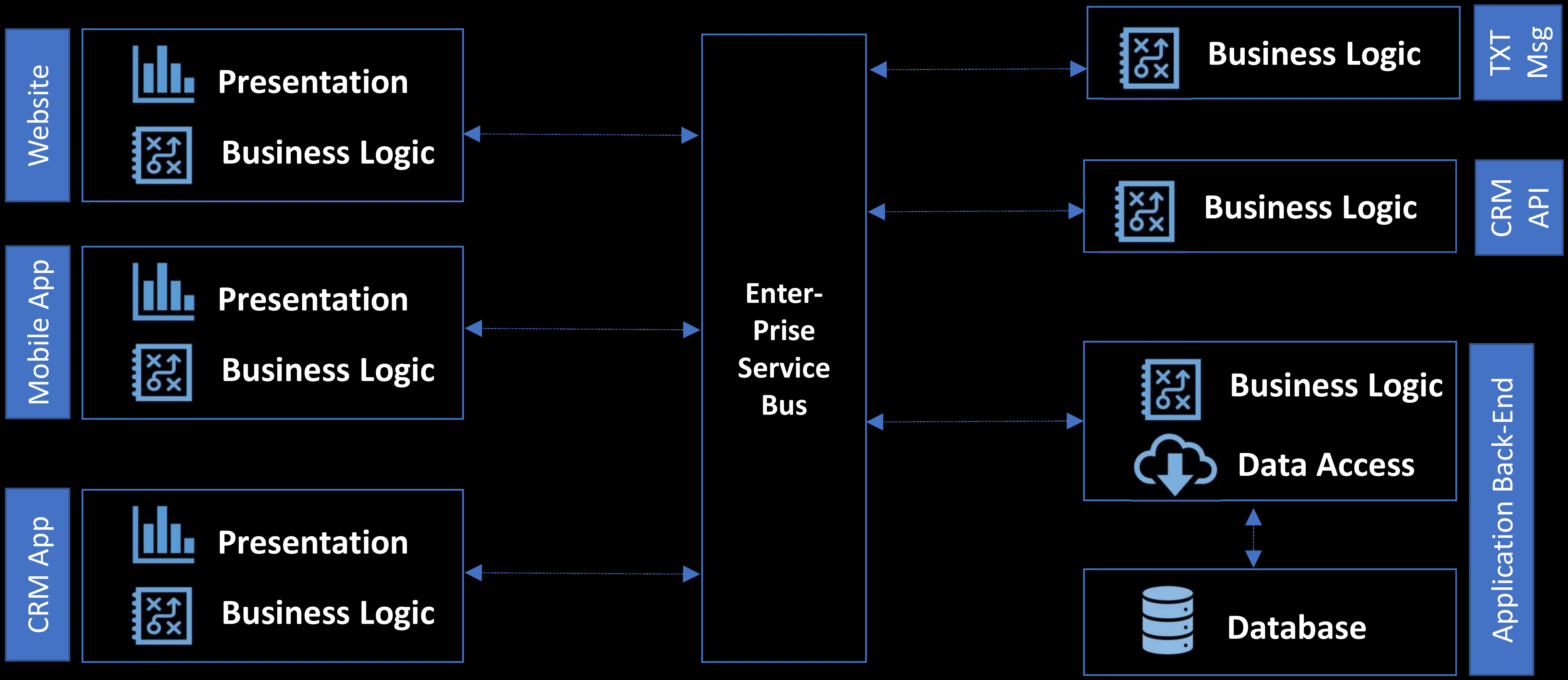
- Break Up the Business Logic Even More… into as many Tiers as required.
- That’s a lot of middleware. How do we deal with all that inter-process communication?



Enterprise Service Bus

- The ESB is a software application which manages the commuinication among independent systems.
- It provides a consistent messaging API and guarantees delivery of information.
- It’s a more robust middleware replacement used as the message backbone for N-tier applications.
- Multiple applications share messages across the same bus. This is the foundation of **Service-Oriented Architecture**

Service-Oriented Architecture



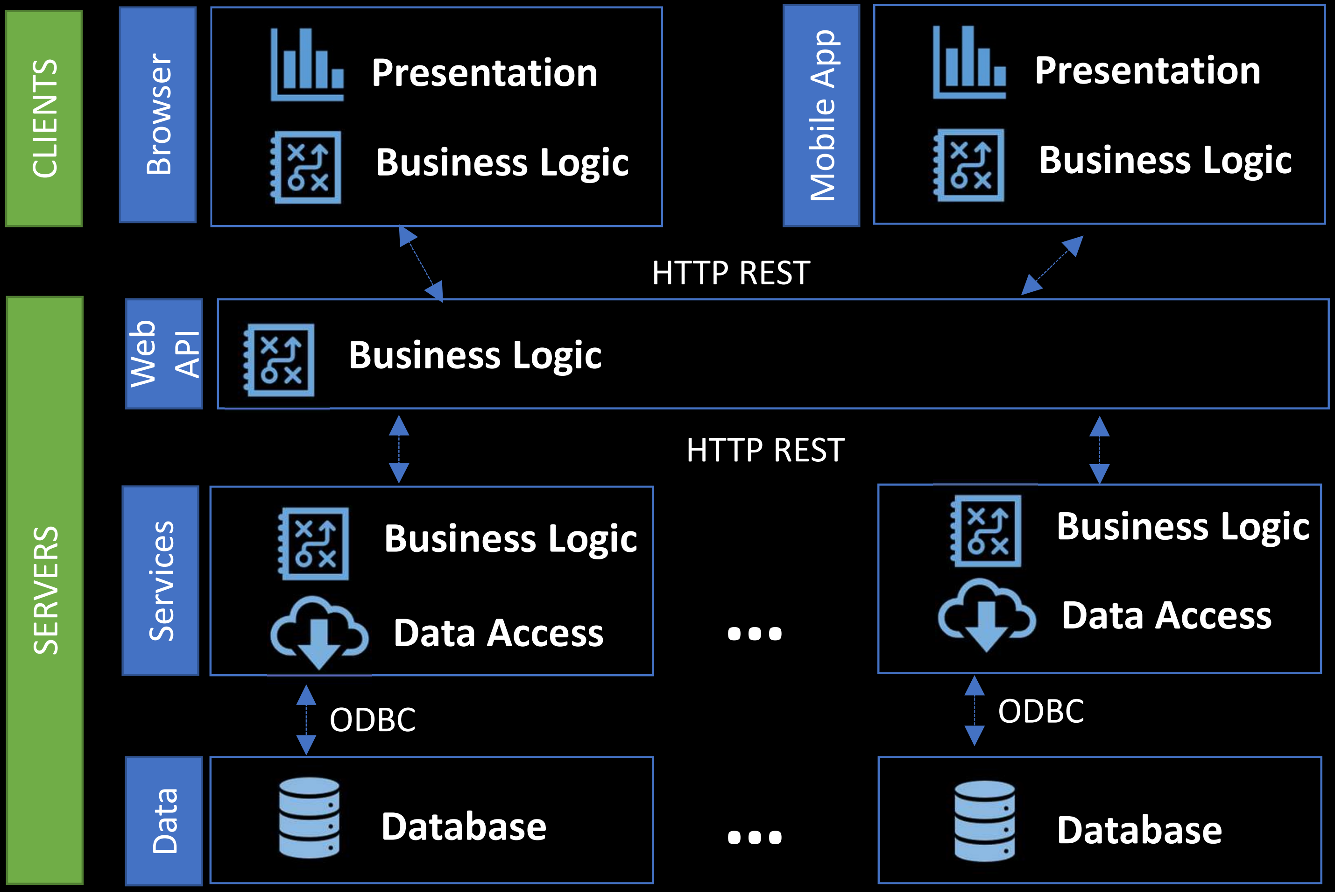
Web As Middleware and Microservices

- The Internet ushered in major changes for application development.
- The SOAP and REST protocols over HTTP made it easy for developers to divide up the layers of their application and split business logic into manageable microservices.
- These microservices manage a single responsibility, making the application easier to update and manage.

Typical Micro Services Responsibilities

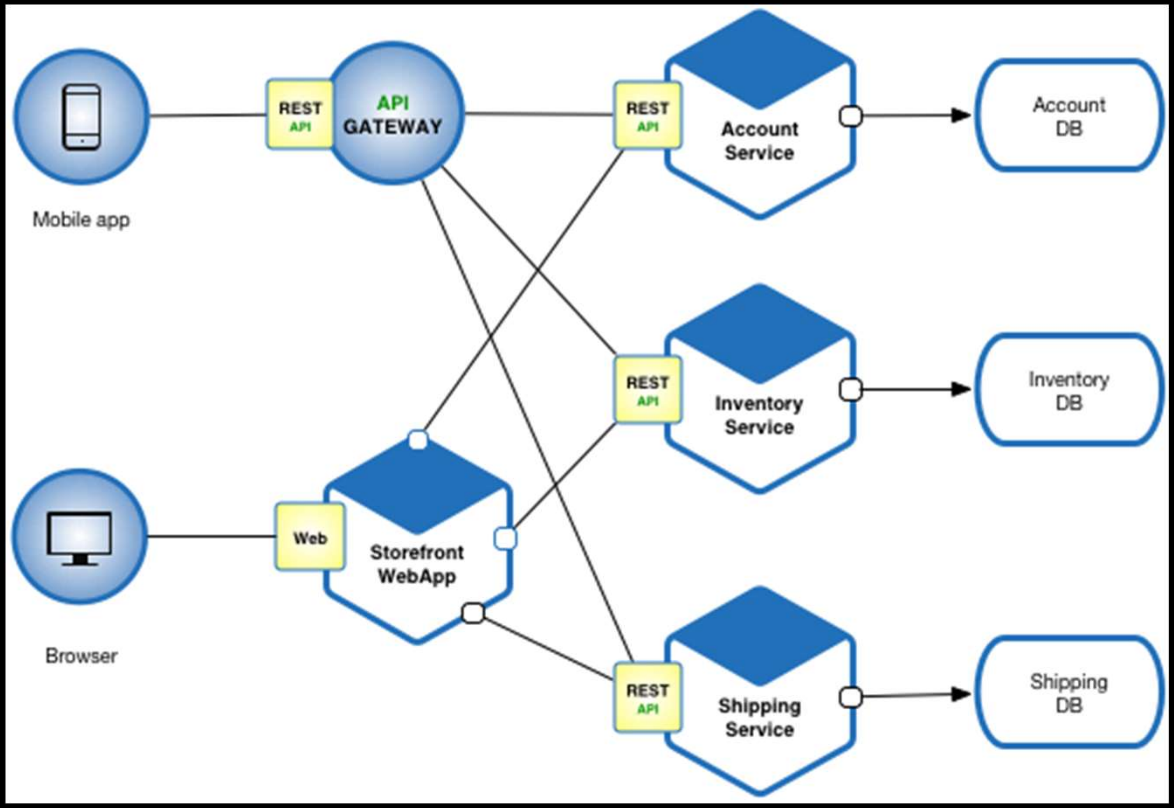
- Business Capability
  - Customers
  - Orders
  - Inventory
- Messaging
  - Email
  - Push Notification
  - TXT Alert
- Function / Task
  - Transcode Video
  - Convert a File
  - Close-Caption Video
  - Machine Learning
- Other Data
  - Write Log Information
  - Usage Statistics (Telemetry)

Micro Services



Micro Services Example

- Brower and Mobile app are clients
- HTTP and REST API's are the Middleware
- Separate Services for each business capability of the application


Mobile app
REST API
API GATEWAY
REST API
Account Service
Account DB
Browser
Web
Storefront WebApp
REST API
Inventory Service
Inventory DB
REST API
Shipping Service
Shipping DB