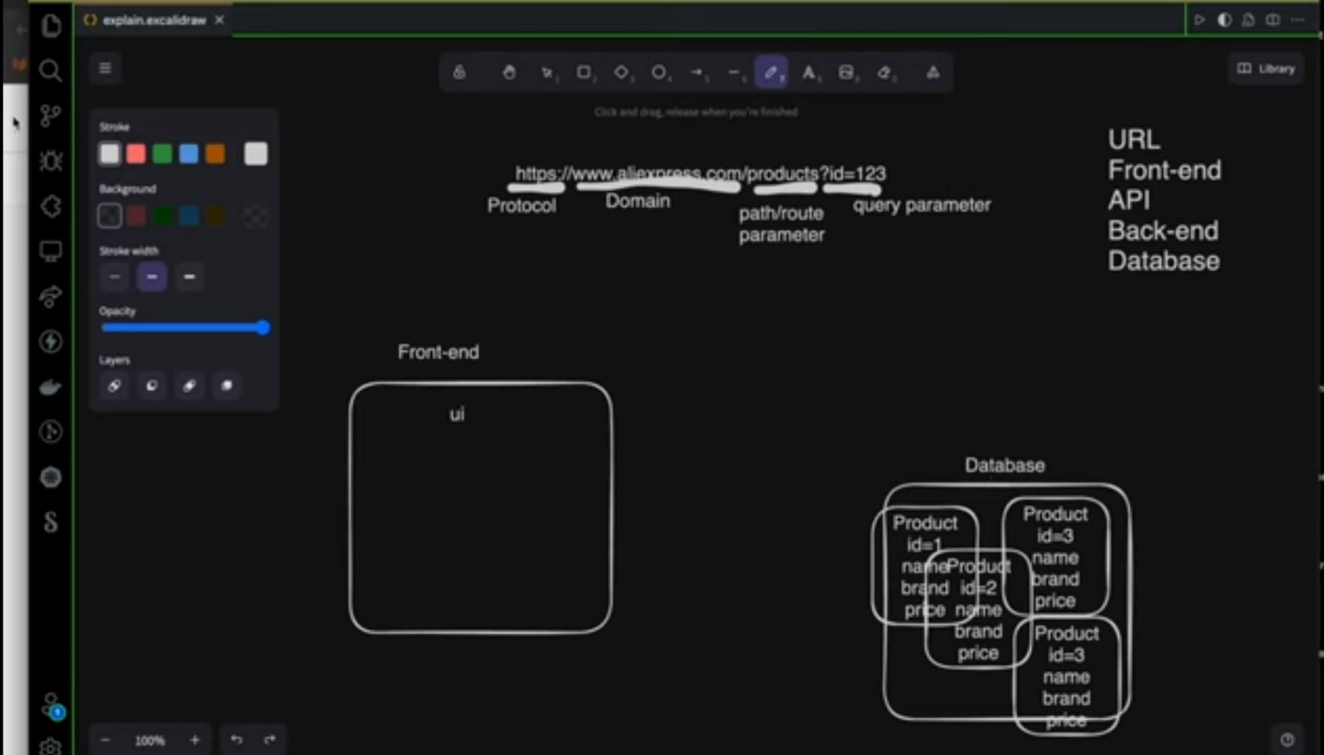
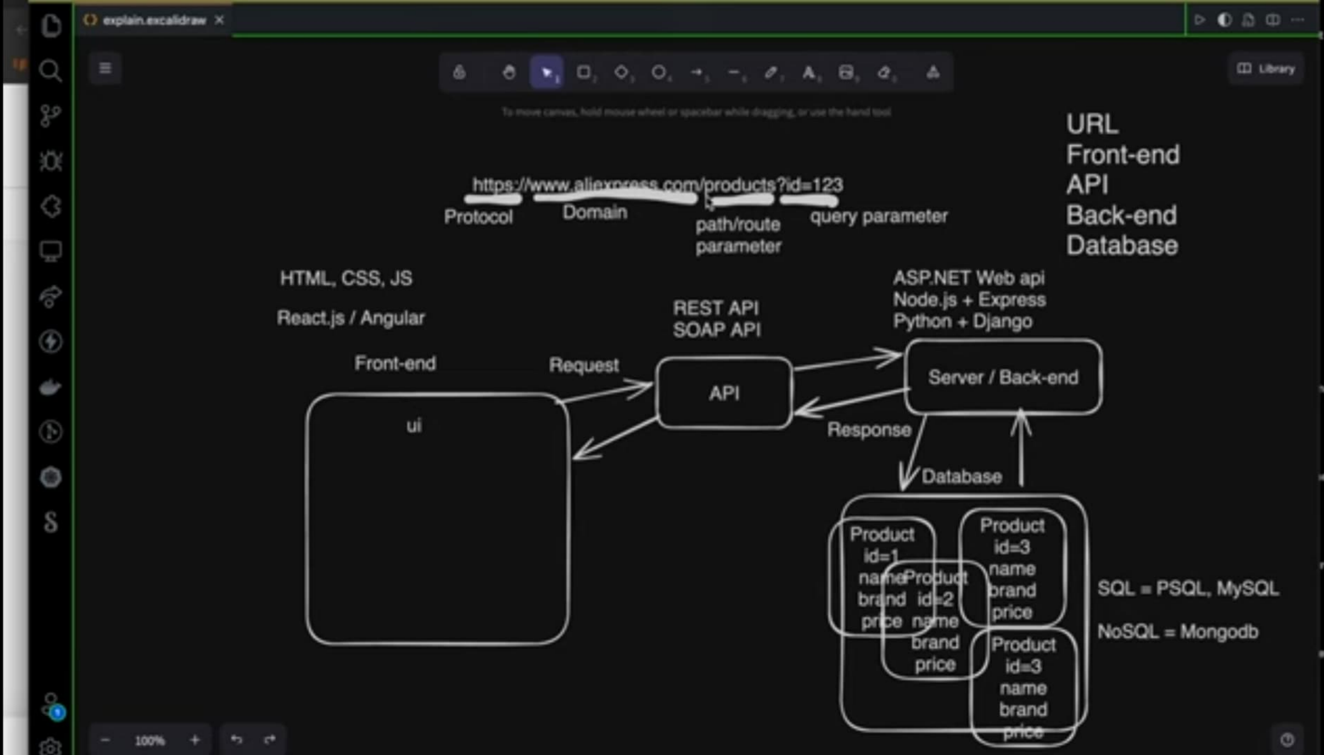
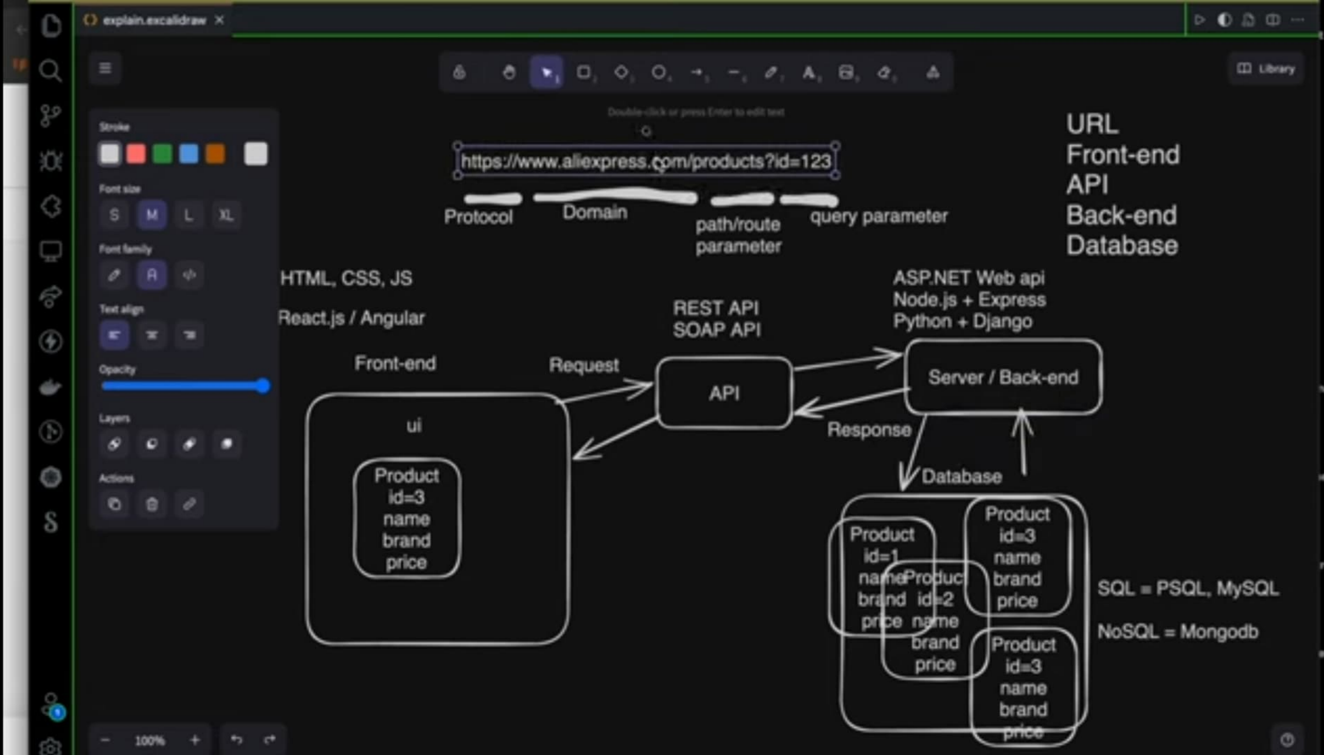
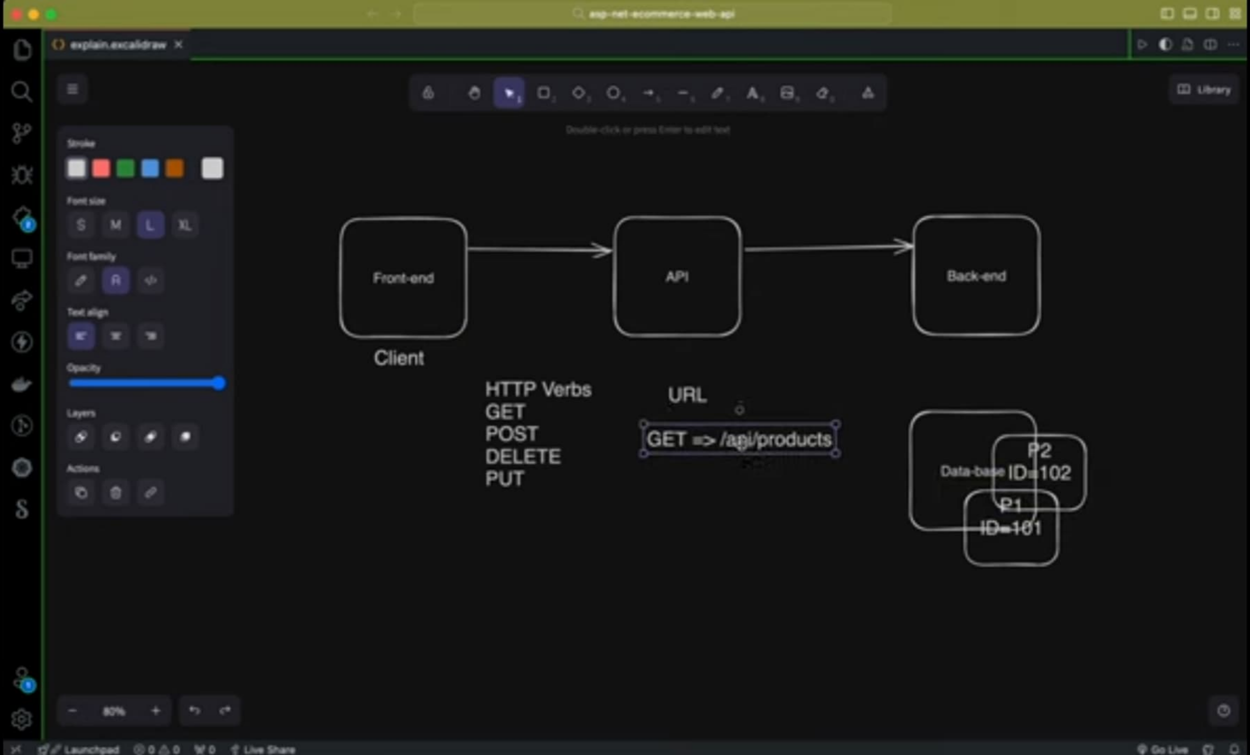
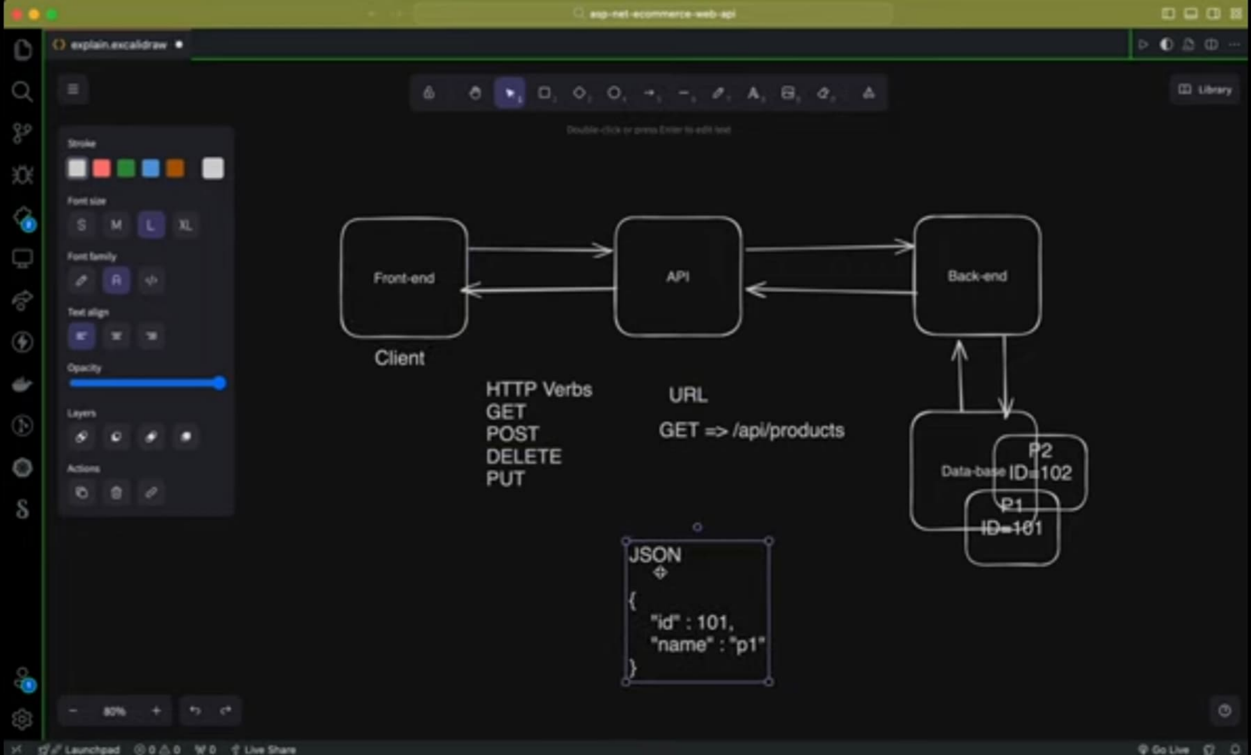

explain.excalidraw
Library
Click and drag, release when you're finished
Stroke
Background
Stroke width
Opacity
Layers
100%
https://www.aliexpress.com/products?id=123
Protocol
Domain
path/route parameter
query parameter
URL
Front-end
API
Back-end
Database
Front-end
ui
Database
Product id=1 name brand price
Product id=2 name brand price
Product id=3 name brand price
Product id=3 name brand price


explain.excalidraw
Library
To move canvas, hold mouse wheel or spacebar while dragging, or use the hand tool
https://www.aliexpress.com/products?id=123
Protocol
Domain
path/route parameter
query parameter
URL
Front-end
API
Back-end
Database
HTML, CSS, JS
React.js / Angular
Front-end
ui
Request
REST API
SOAP API
API
ASP.NET Web api
Node.js + Express
Python + Django
Server / Back-end
Response
Database
Product id=1 name brand price
Product id=2 name brand price
Product id=3 name brand price
Product id=3 name brand price
SQL = PSQL, MySQL
NoSQL = Mongodb
100%


explain.excalidraw
https://www.aliexpress.com/products?id=123
Protocol
Domain
path/route parameter
query parameter
URL
Front-end
API
Back-end
Database
HTML, CSS, JS
React.js / Angular
REST API
SOAP API
ASP.NET Web api
Node.js + Express
Python + Django
Front-end
Request
API
Server / Back-end
Response
ui
Product id=3 name brand price
Database
Product id=1 name brand price
Product id=2 name brand price
Product id=3 name brand price
Product id=3 name brand price
SQL = PSQL, MySQL
NoSQL = Mongodb


asp-net-ecommerce-web-api
explain.excalidraw
Library
Stroke
Font size
S
M
L
XL
Font family
Text align
Opacity
Layers
Actions
80%
Front-end
API
Back-end
Client
HTTP Verbs
GET
POST
DELETE
PUT
URL
GET => /api/products
Data-base
P2
ID=102
P1
ID=101
Launchpad
Live Share
Go Live



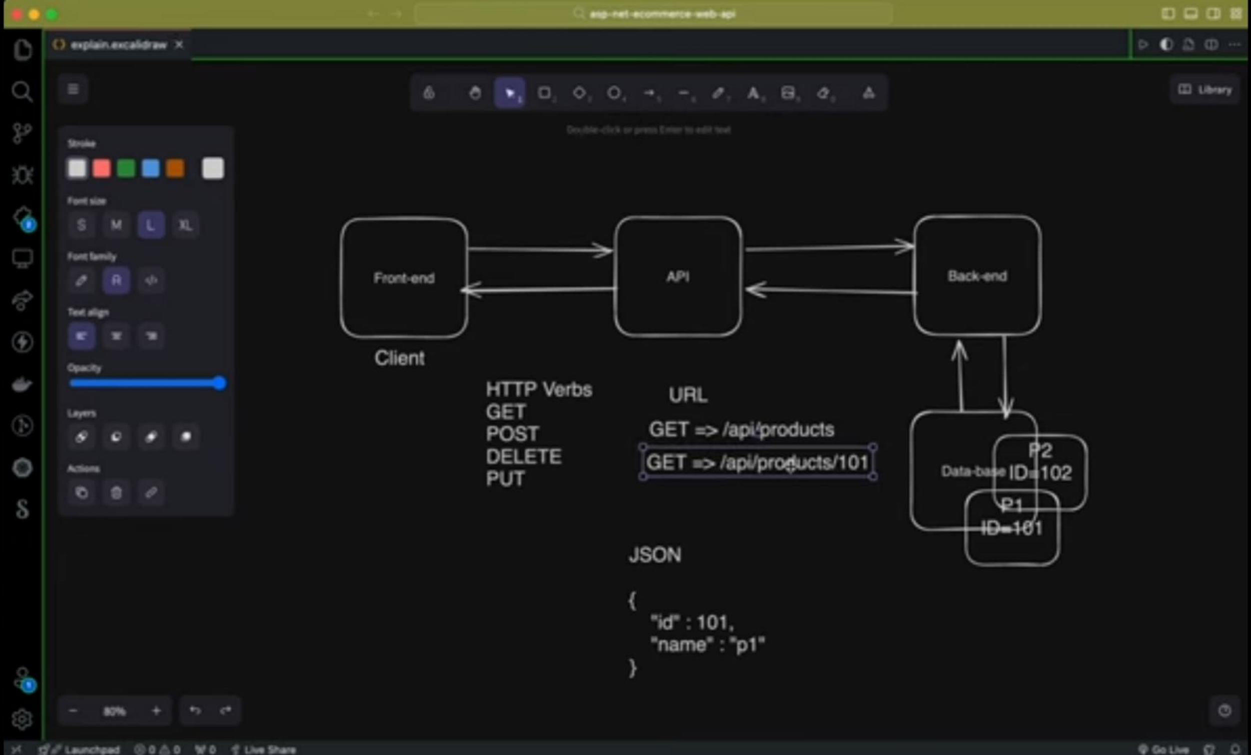
HTTP Verbs
GET
POST
DELETE
PUT

URL

GET => /api/products

JSON

```
{
  "id" : 101,
  "name" : "p1"
}
```



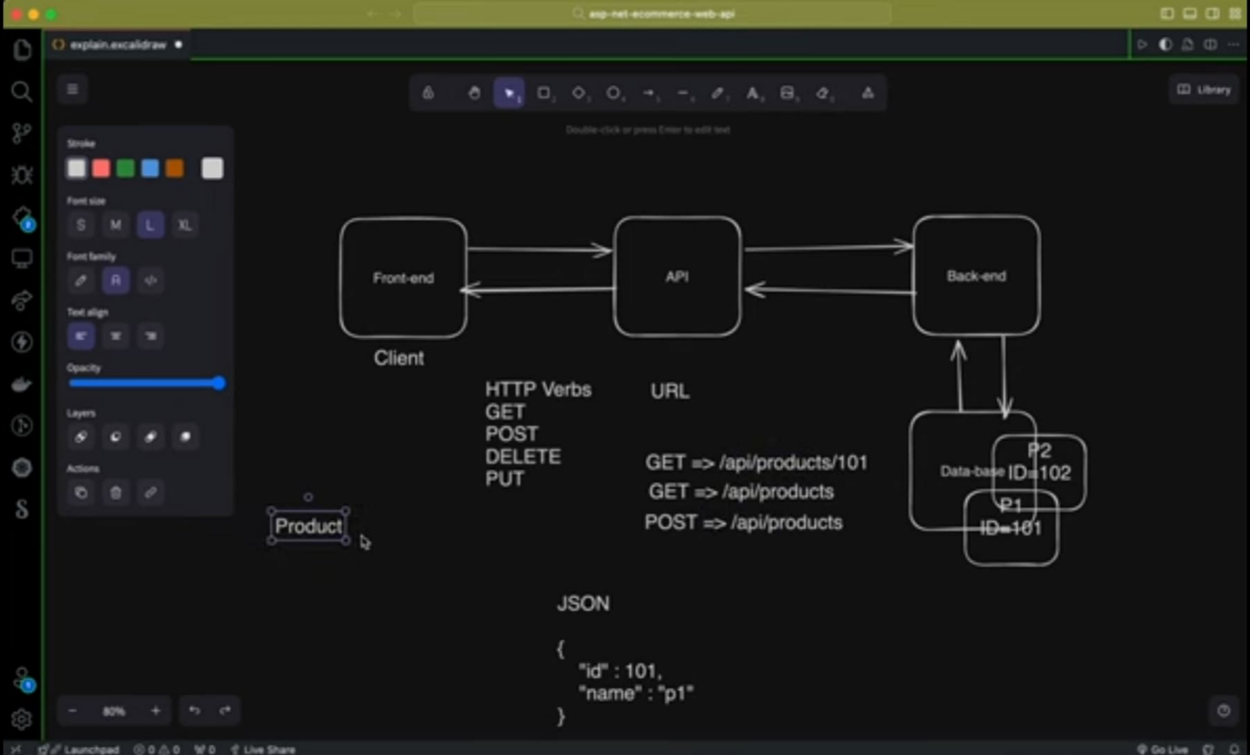
HTTP Verbs
GET
POST
DELETE
PUT

URL

GET => /api/products
GET => /api/products/101

JSON

```
{
   "id" : 101,
   "name" : "p1"
}
```



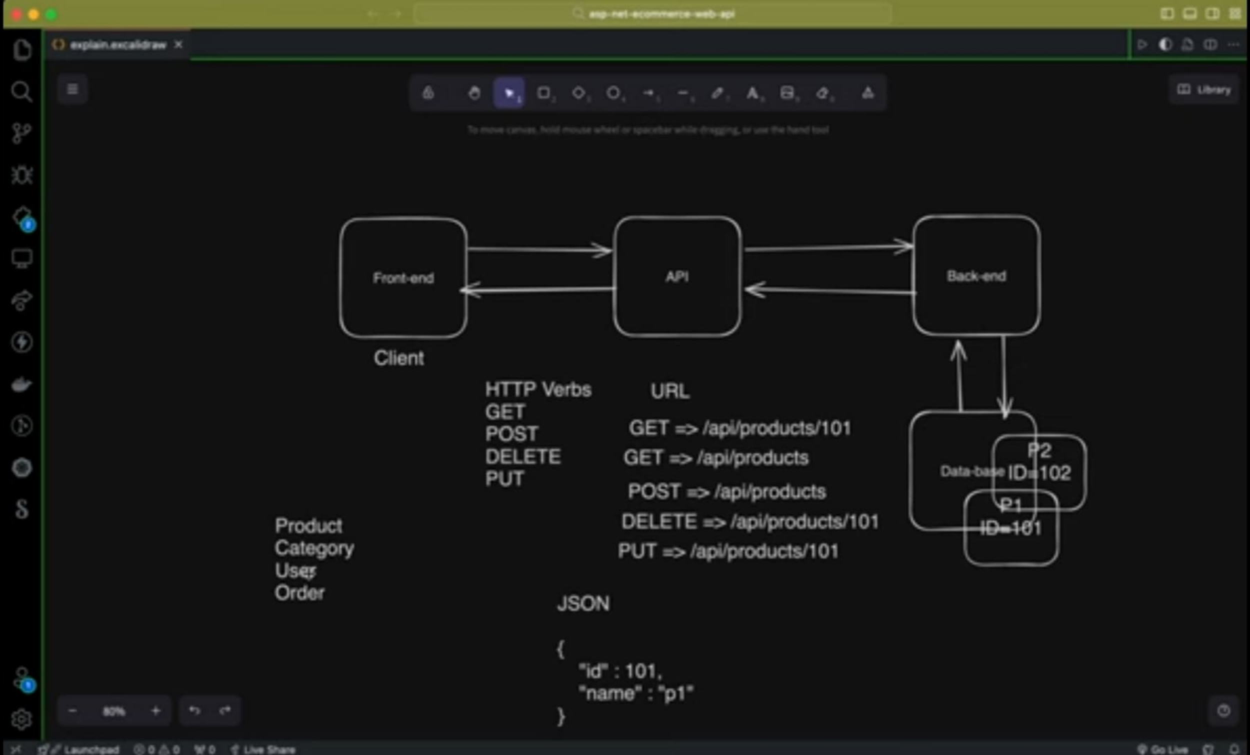
HTTP Verbs
GET
POST
DELETE
PUT

URL

GET => /api/products/101
GET => /api/products
POST => /api/products

JSON

```
{
  "id" : 101,
  "name" : "p1"
}
```



HTTP Verbs
GET
POST
DELETE
PUT

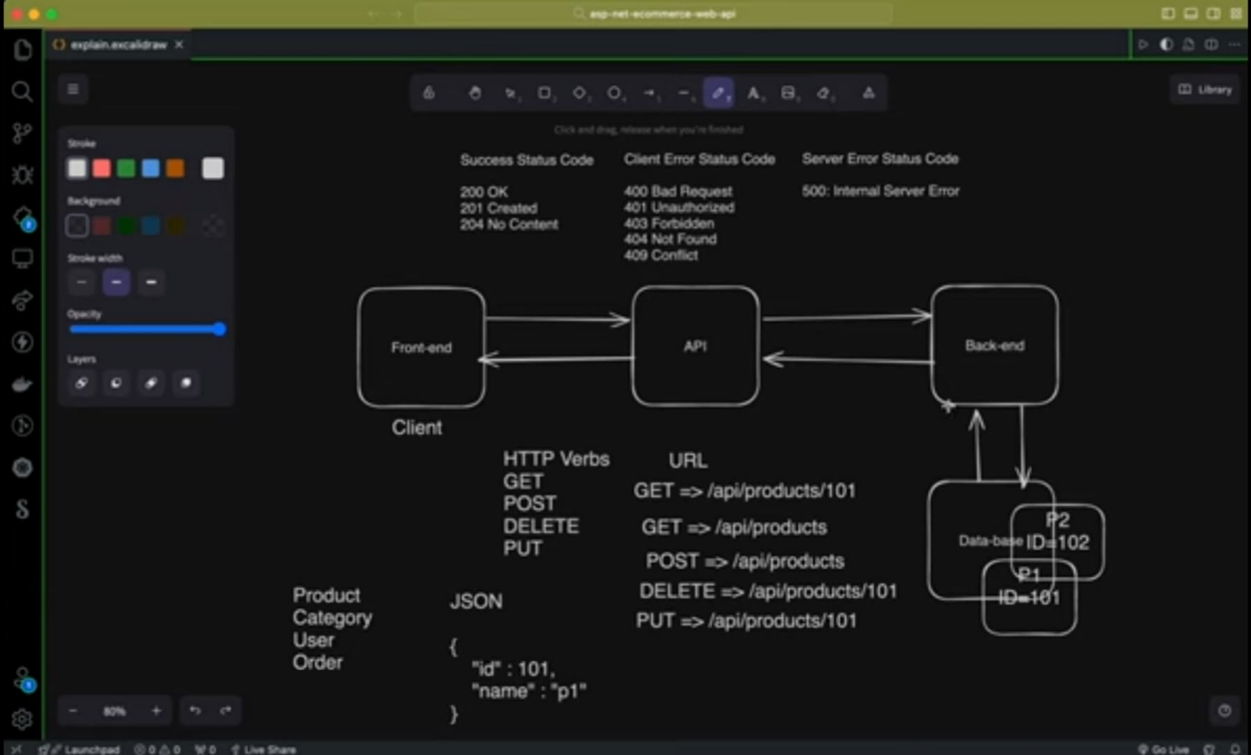
URL

GET => /api/products/101
GET => /api/products
POST => /api/products
DELETE => /api/products/101
PUT => /api/products/101

Product
Category
User
Order

JSON

```
{
   "id" : 101,
   "name" : "p1"
}
```



Success Status Code

200 OK
201 Created
204 No Content

Client Error Status Code

400 Bad Request
401 Unauthorized
403 Forbidden
404 Not Found
409 Conflict

Server Error Status Code

500: Internal Server Error

Client

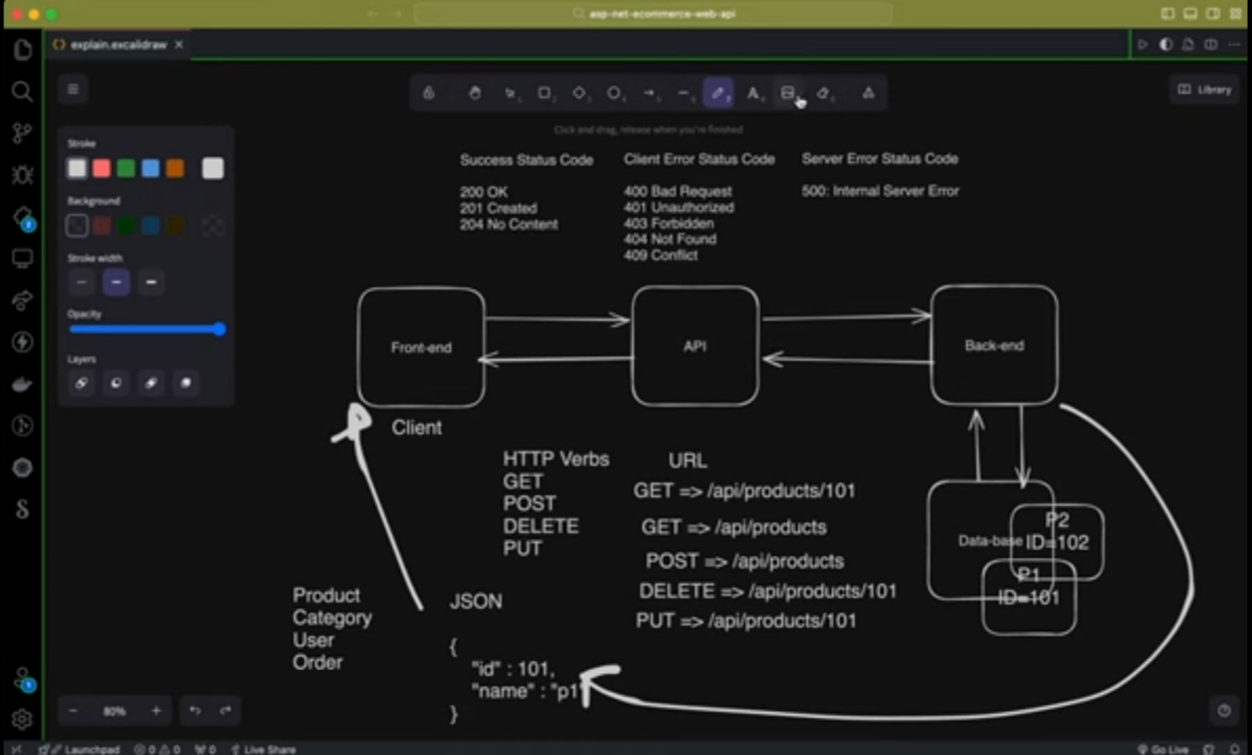
HTTP Verbs
GET
POST
DELETE
PUT

URL
GET => /api/products/101
GET => /api/products
POST => /api/products
DELETE => /api/products/101
PUT => /api/products/101

Product
Category
User
Order

JSON

```
{
   "id" : 101,
   "name" : "p1"
}
```



Success Status Code	Client Error Status Code	Server Error Status Code
200 OK	400 Bad Request	500: Internal Server Error
201 Created	401 Unauthorized	
204 No Content	403 Forbidden	
	404 Not Found	
	409 Conflict	

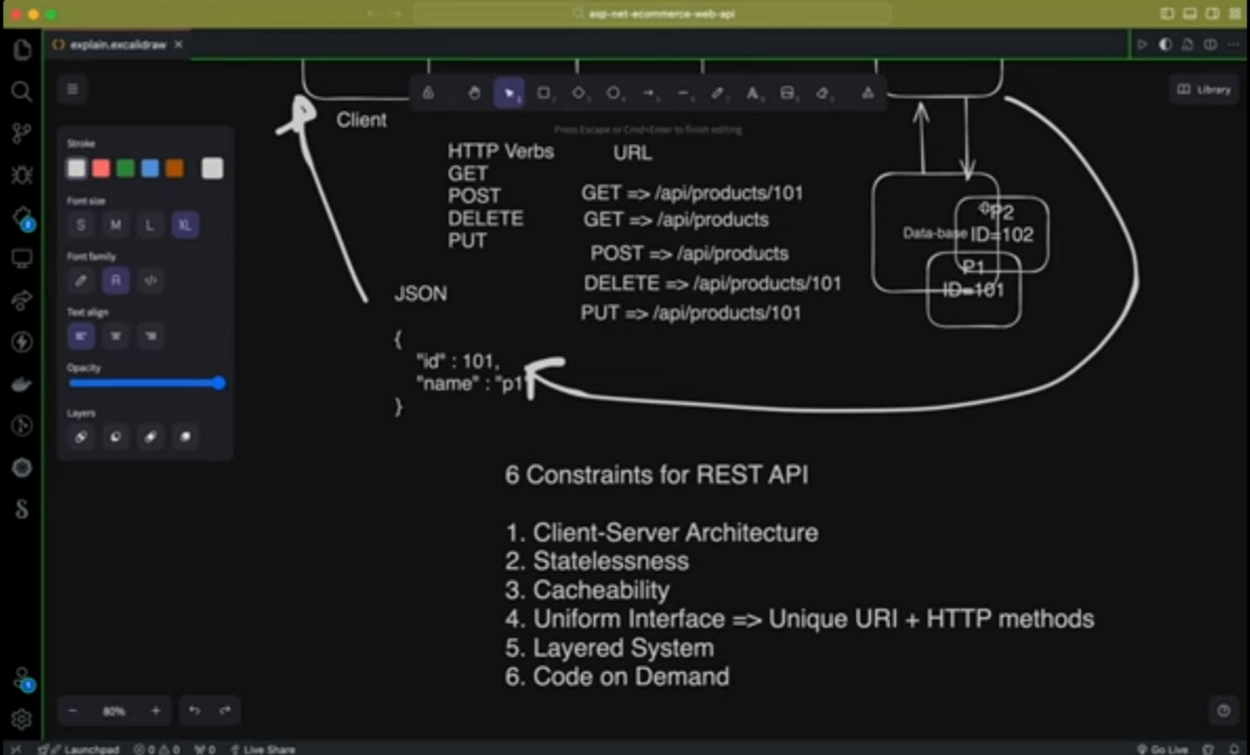
HTTP Verbs
GET
POST
DELETE
PUT

URL
GET => /api/products/101
GET => /api/products
POST => /api/products
DELETE => /api/products/101
PUT => /api/products/101

Product
Category
User
Order

JSON

```
{
  "id" : 101,
  "name" : "p1"
}
```



HTTP Verbs
GET
POST
DELETE
PUT

URL

GET => /api/products/101
GET => /api/products
POST => /api/products
DELETE => /api/products/101
PUT => /api/products/101

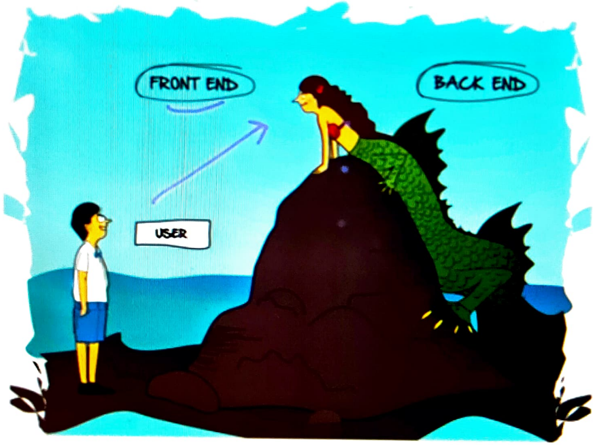
JSON

```
{
  "id" : 101,
  "name" : "p1"
}
```

6 Constraints for REST API

1. Client-Server Architecture
2. Statelessness
3. Cacheability
4. Uniform Interface => Unique URI + HTTP methods
5. Layered System
6. Code on Demand

$$\text{Full-Stack} = \underbrace{\text{Front-end}}_{} + \boxed{\underbrace{\text{Back-end}}_{}}$$


FRONT END
BACK END
USER

Leave a rating Your pr

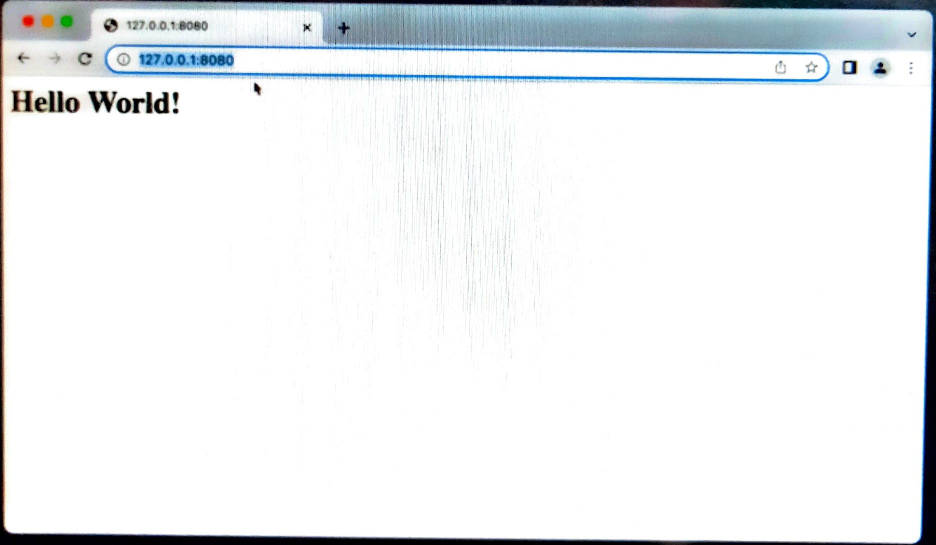
What is the backend?

Server

Application

Database

The
App Brewery
www.appbrewery.com

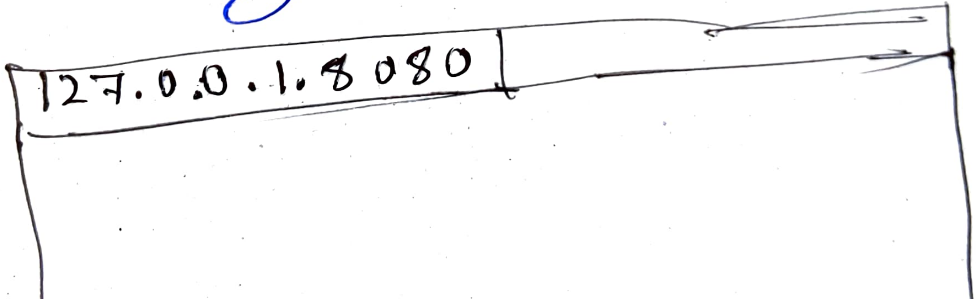


Hello World!

the web-app retrive data — th
do function in
d

Server

• Any computer, that connect with network can act as server. and when we ~~will~~ would be build our web site locally, we will be using something called local host


127.0.0.1.8080

Web Development Bootcamp Leave a rating Your pro

Application

Logic

Respond to Requests

Status Code

R.

Application

Im'm logic will determine how respond to the request from the browser. Depending on the request is like user click on button and, that request comp through the server and to application and the application should know ~~the~~ how response by send particular html page.

- But sometime brows respons with different thing other than html code, it could data or status code like 404

404 — try tell the browser that that particular request is invaild and application does not known how to response with the request

Database

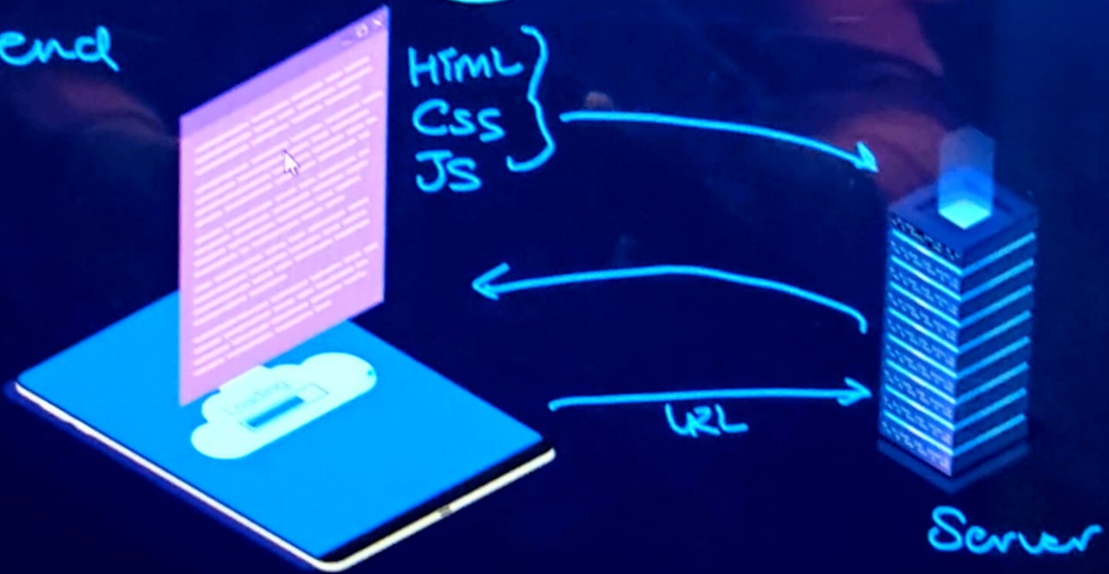
Store Data

Persist

Static

Web Page

Frontend

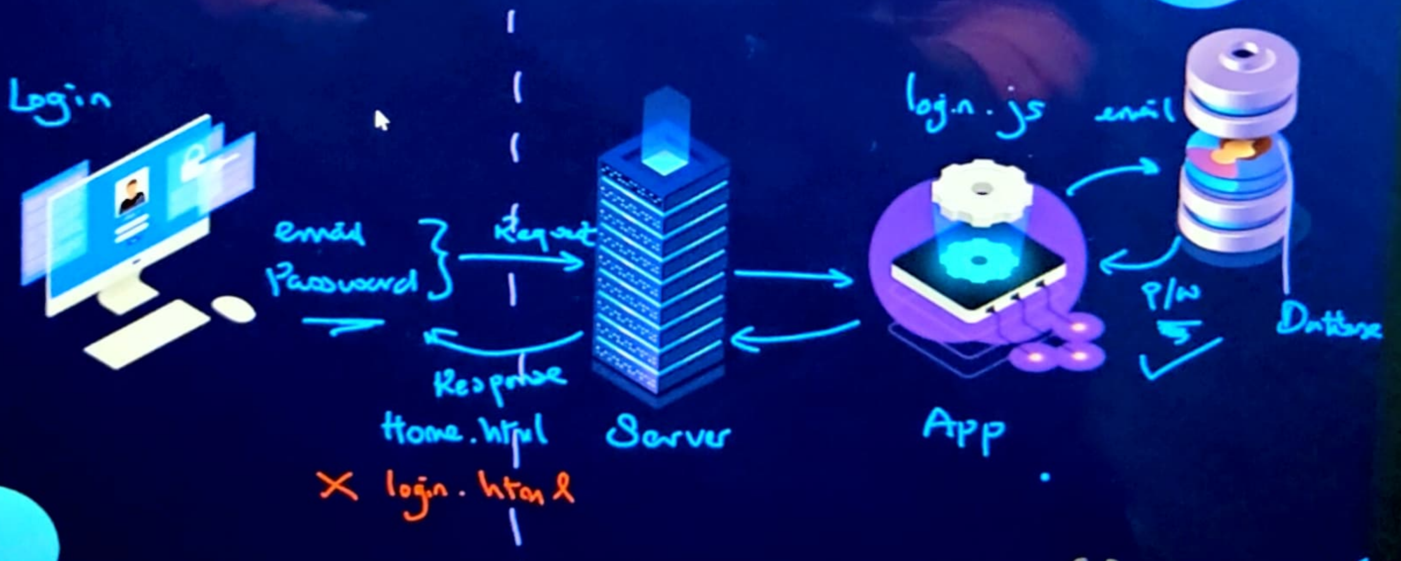


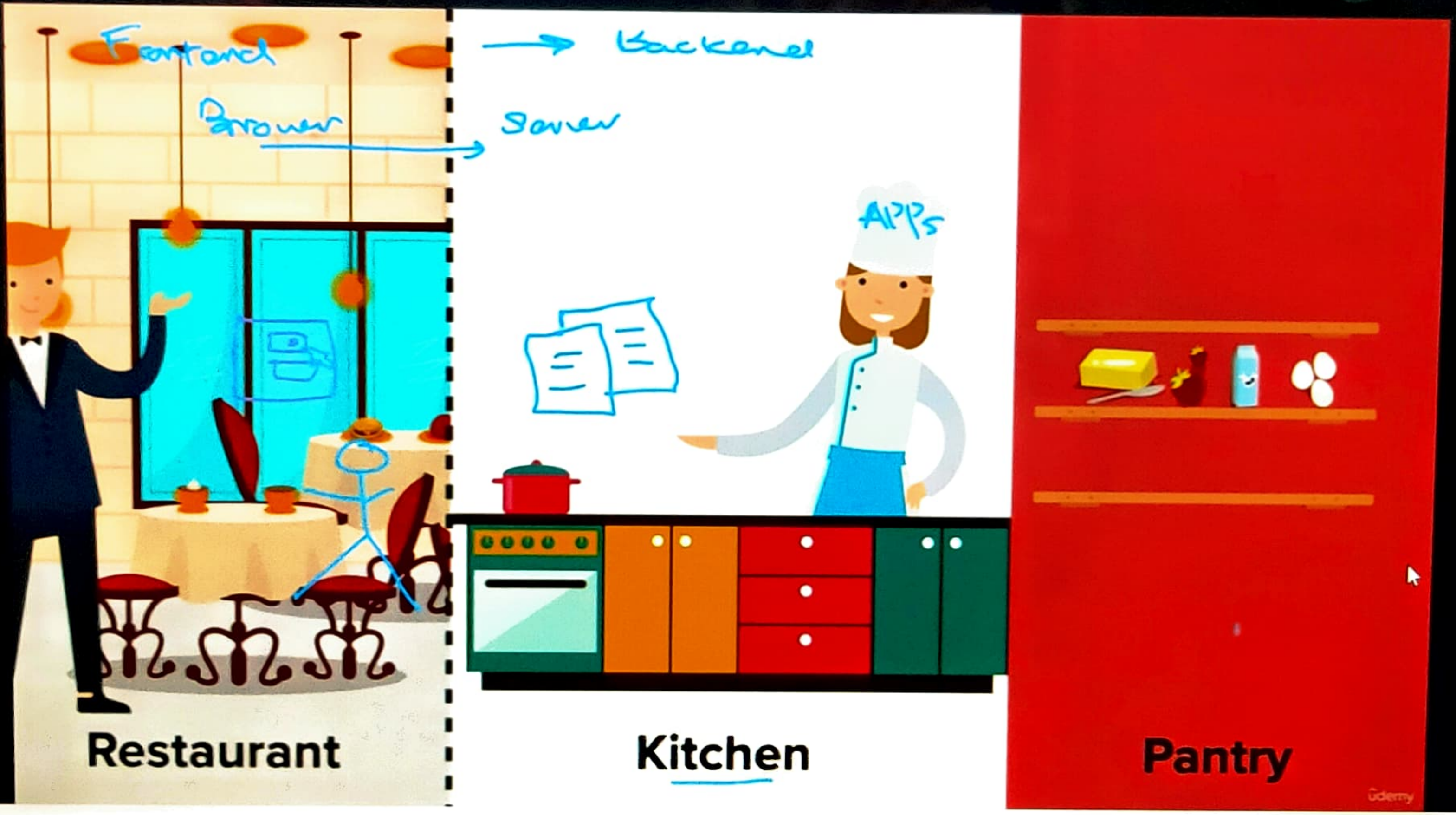
www.appbrewery.com

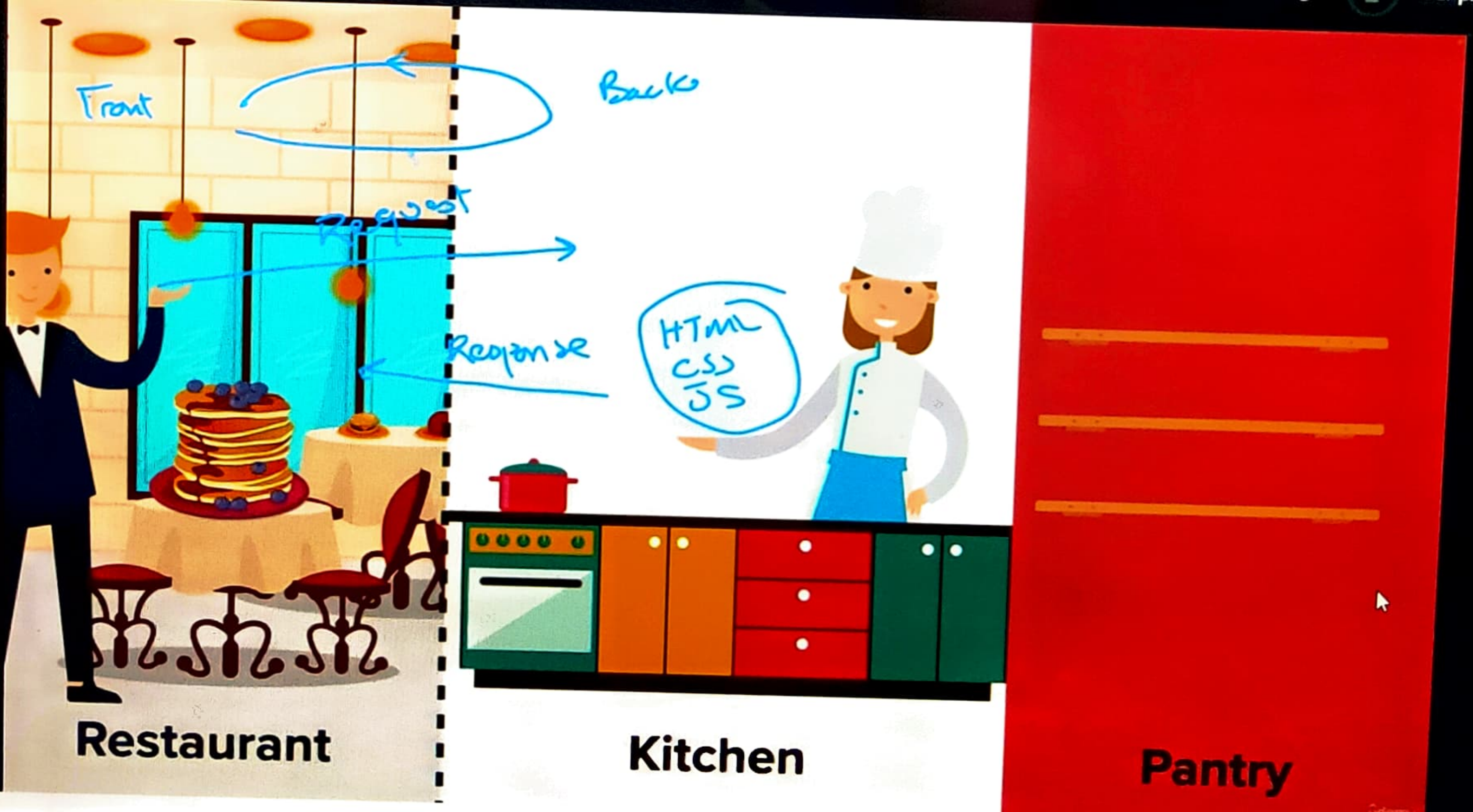
The App Brewery

Leave a rating Your progress

Web App



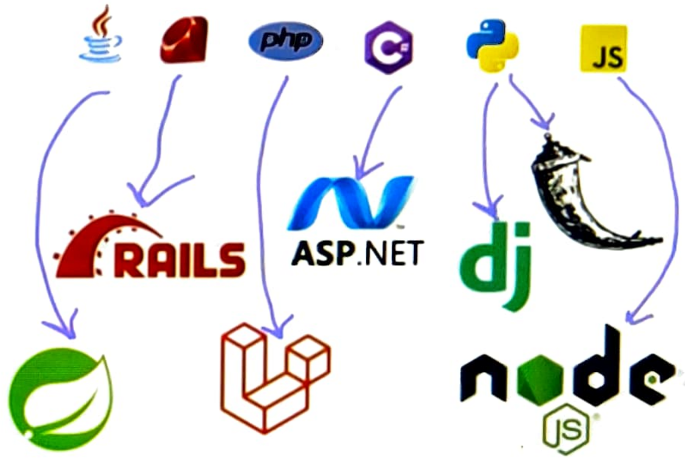

Frontend
Backend
Browser
Server
APPs
Restaurant
Kitchen
Pantry
Udemy


Front
Back
Request
Response
HTML
CSS
JS
Restaurant
Kitchen
Pantry

Front-end

HTML CSS JS

Back-end


php
JS
RAILS
ASP.NET
dj
node
JS