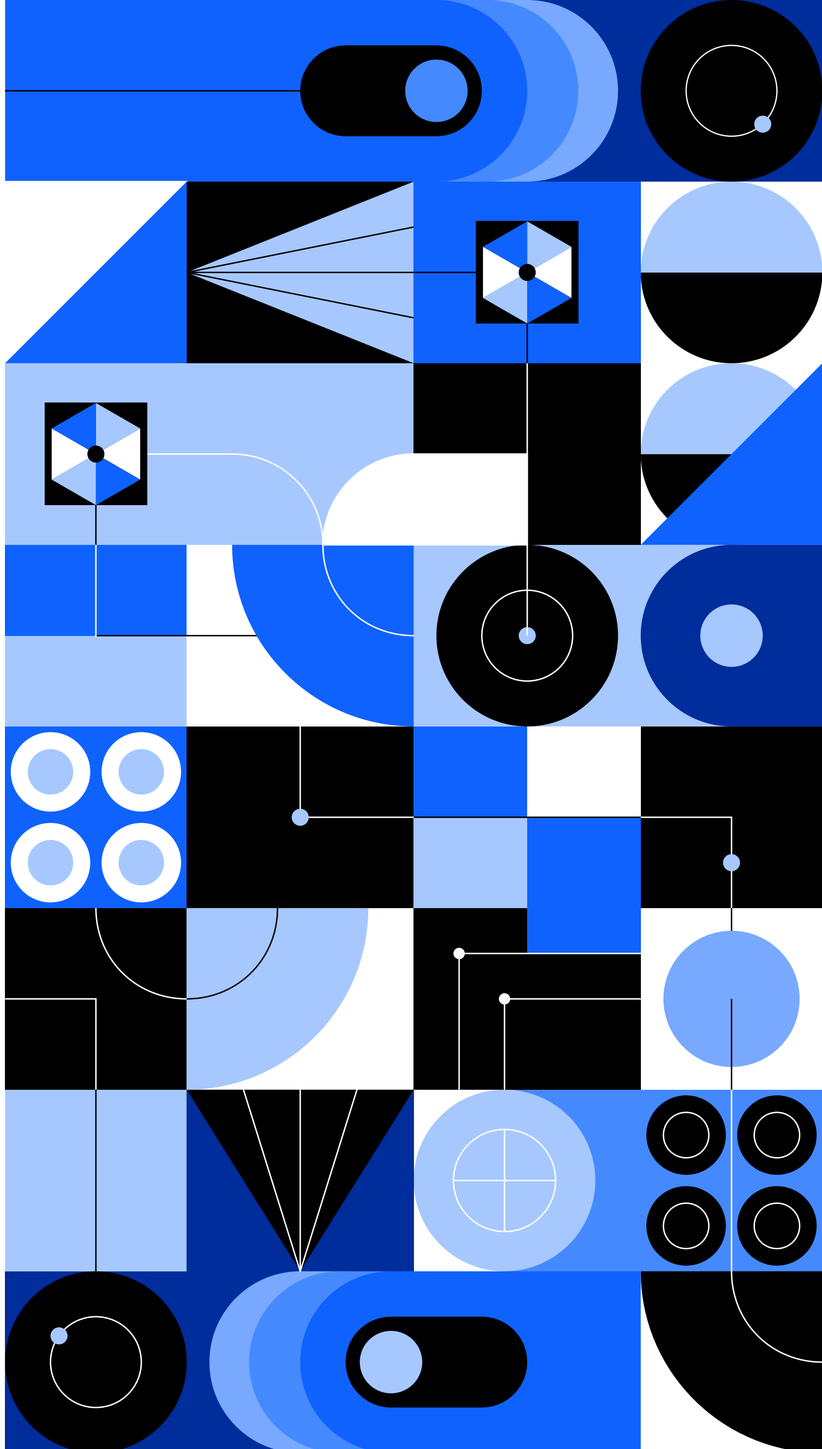
IBM Accelerate Software Developer Track

Wednesdays June 5th – July 25th
6:00pm-8:00pm Eastern Time



Help us identify our Rock Stars!

How?

- Complete the nomination form: https://ibm.biz/accelerate-sw-rockstars

Who can submit a nomination?

- Students
- Coaches
- Instructors
- Coordinators
- Everyone!

IBM Accelerate Software Developer Track

Wednesday
July 10, 2024

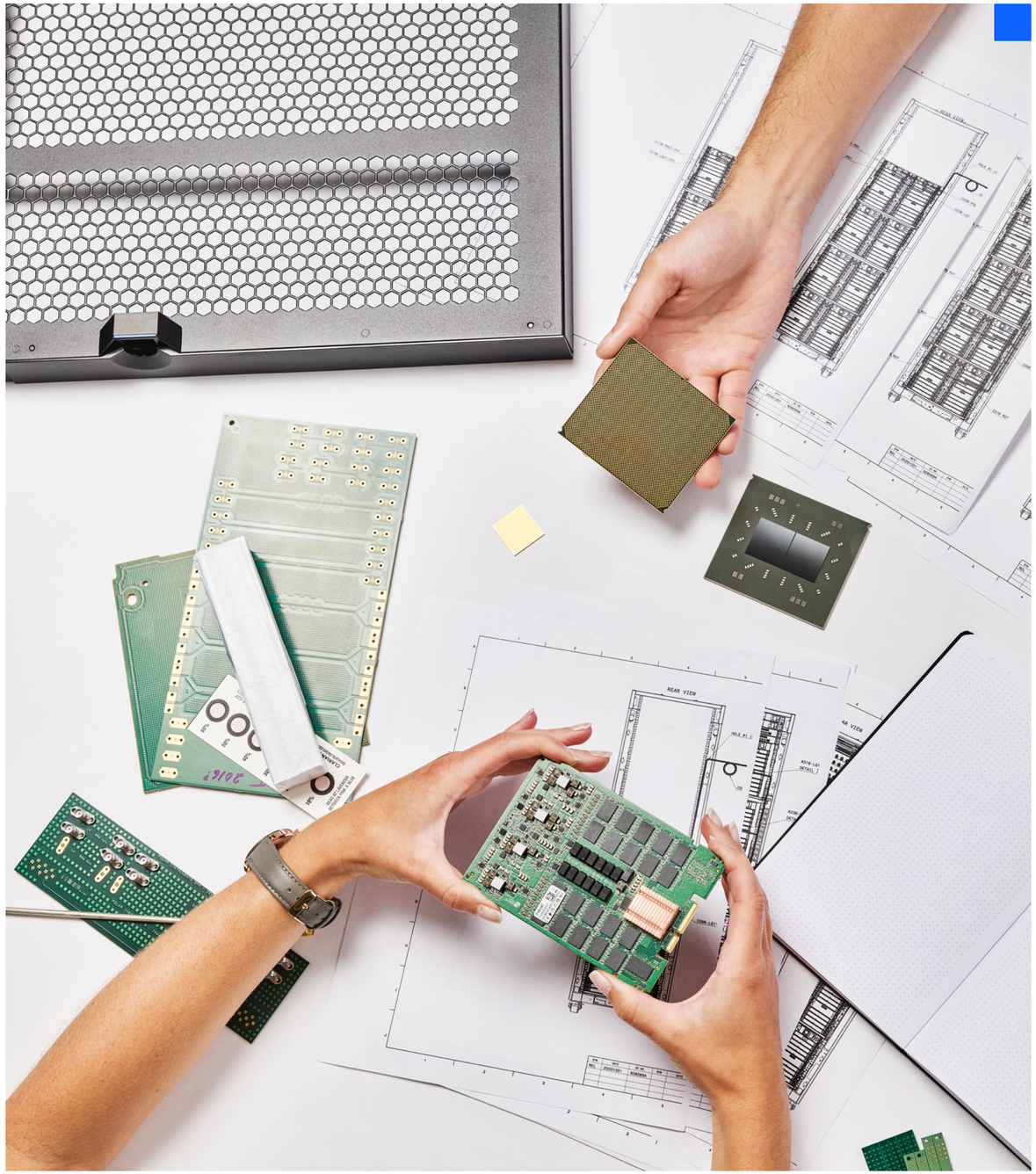
Week 6:
Functional Back End Hosts, SDKs & Design Patterns

Speakers:
Marc Velasco
Mishika S Gaur

Session Agenda

1. Anatomy of Backend Services
2. SDKs and APIs
3. Architecture Patterns for Backend Services
4. Testing
5. Lab

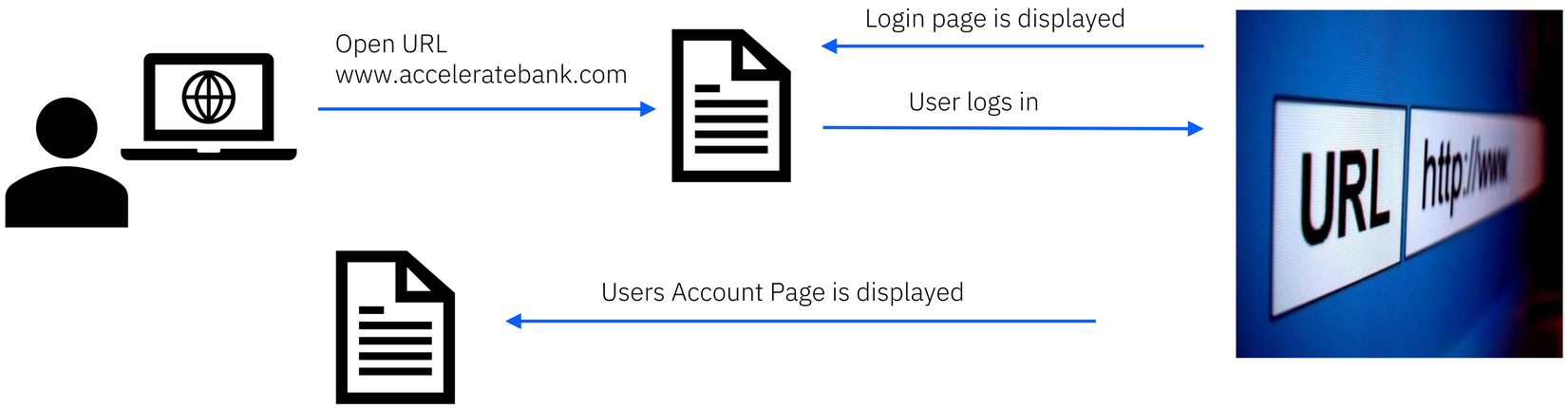
Anatomy of Backend Services



IBM Accelerate 2024

An everyday example:

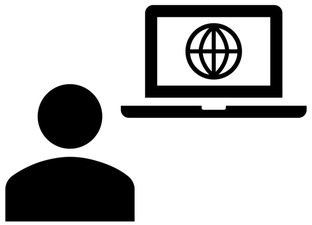
User logs into online banking



Topics:

- What is a session?
  - A remembered set of data that is tied to a user, for example a user is logged in so they don't need to login again if visiting the same site in short order
- What is state and how does it relate?
  - State is any data we need to remember for a user or the work they're doing

Now let’s look at some more complicated tasks



User initiates deposit for check at www.acceleratebank.com

Page to start deposit returned

User inputs amount

Users submits picture of front of check

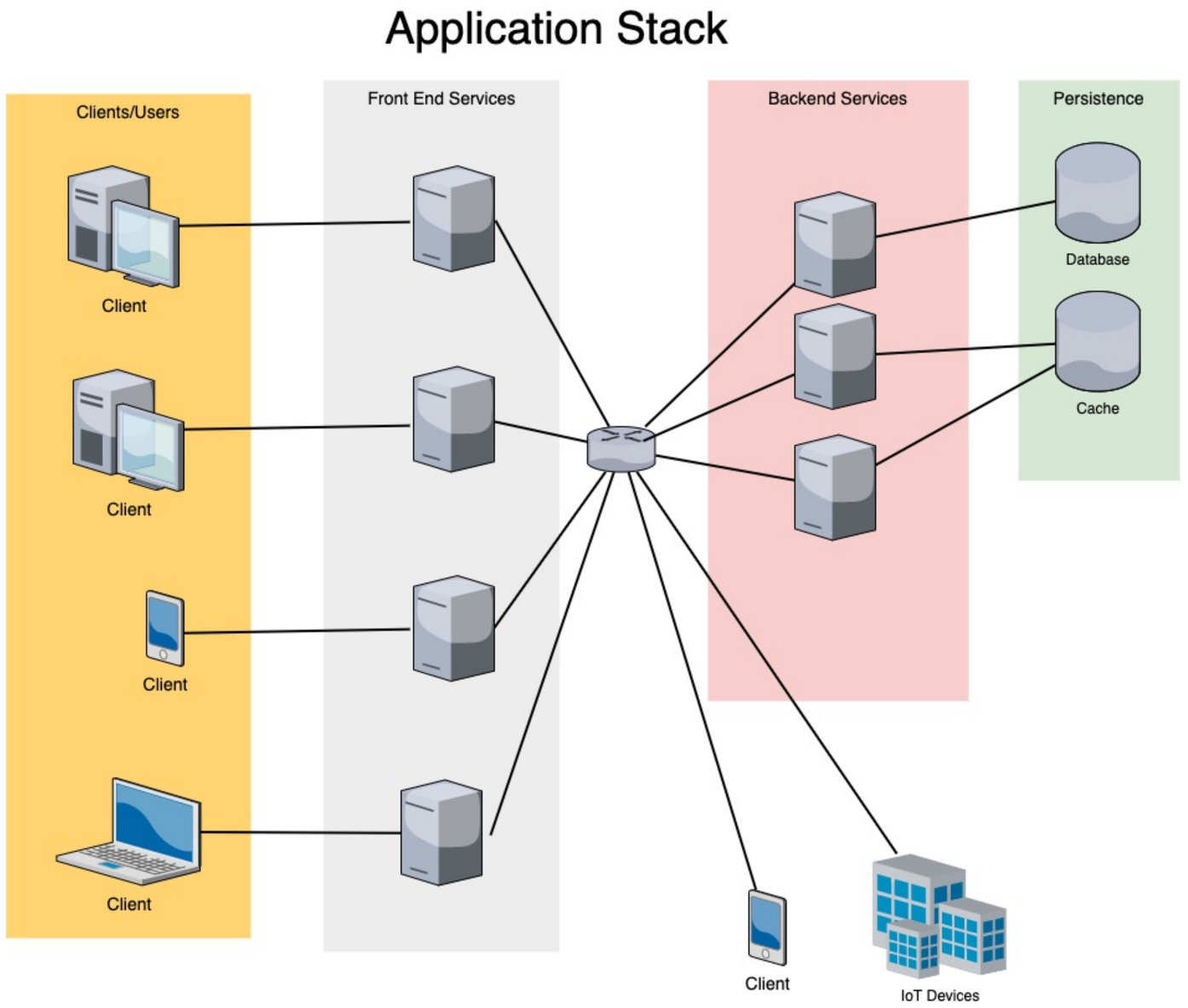
Users submits picture of back of check

User receives confirmation

Topics:

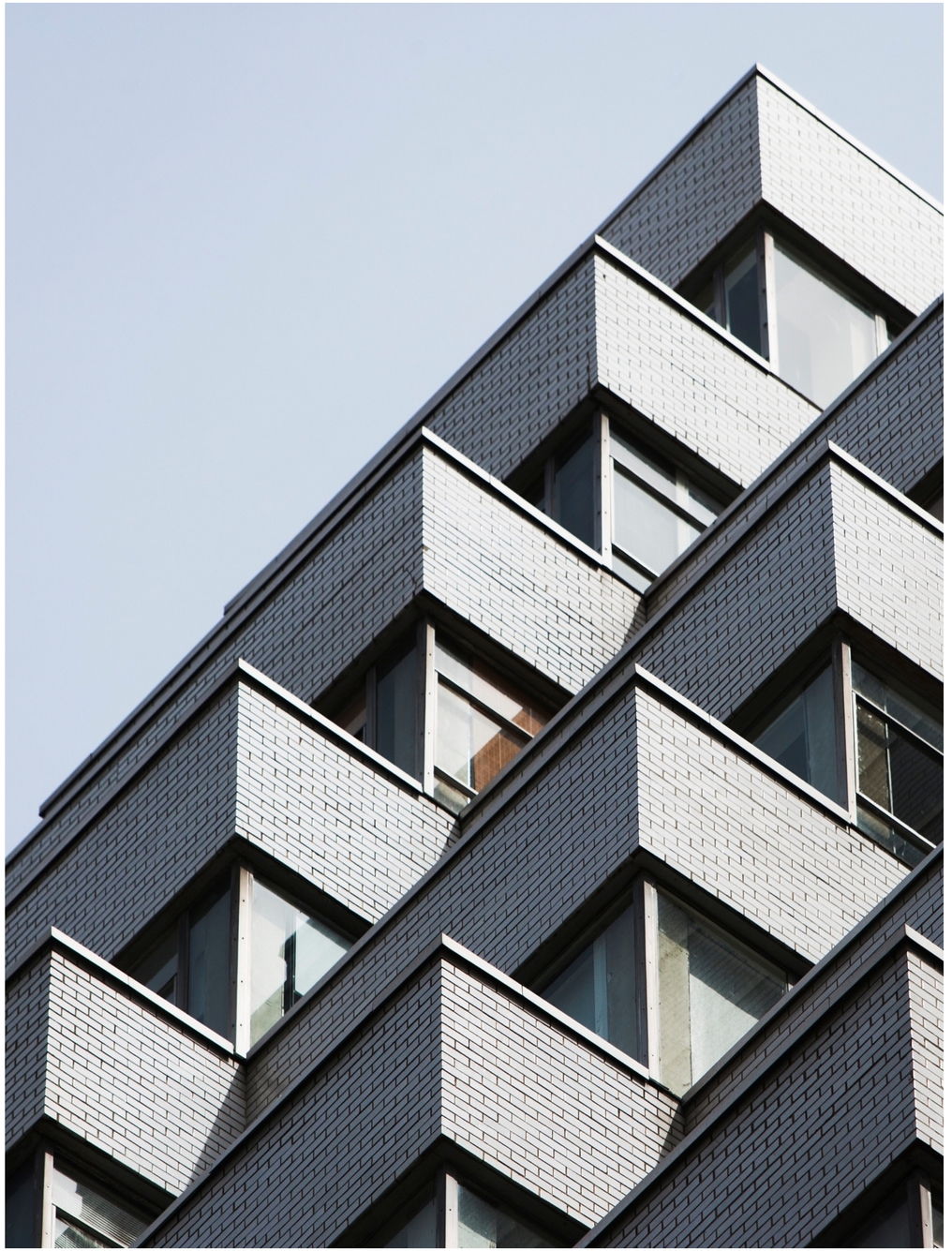
- What if one of these calls fails, slow network etc?
- What happens if a user gets sidetracked between submitting one of the check pictures?

A quick refresher on terminology and topology



What is a Backend Service?

- We have applications or users or devices that do something. They rely on backend services to do that "something".
- So we can say that a "backend service" is:
  - A "service" is code that makes itself available to respond to requests
  - Backend refers to the fact that other services, applications, devices, rely on it to do something in the backend of the application stack
  - Very often backend services "get" or "set/create" data
    - Get bank account balance
    - Deposit $100 in account, add $100 to account balance
    - Which means they often provide access to persistence – databases, cache, nosql, any type of storage for data



Quiz Time

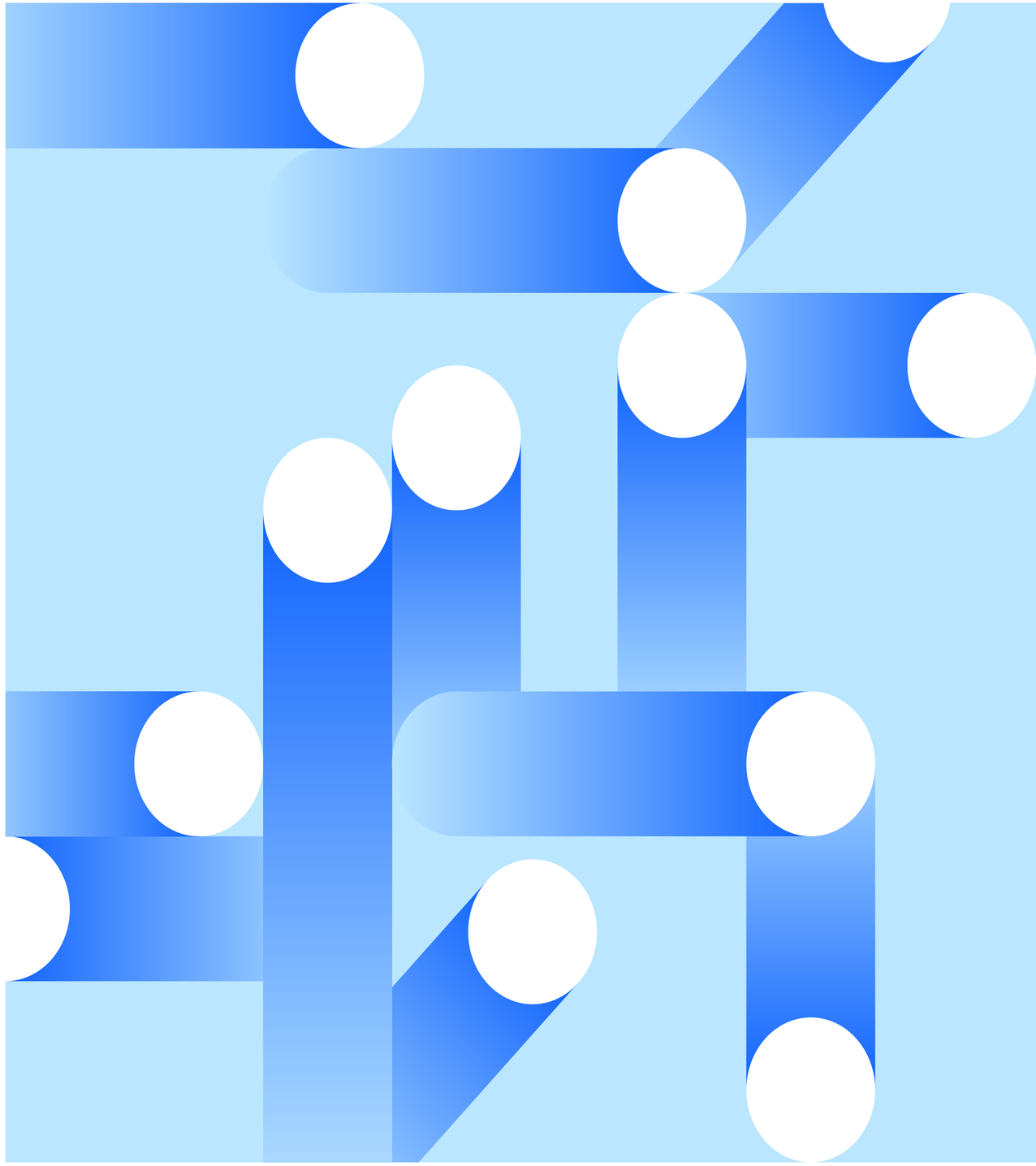
True or False Question:

Backend services are responsible for

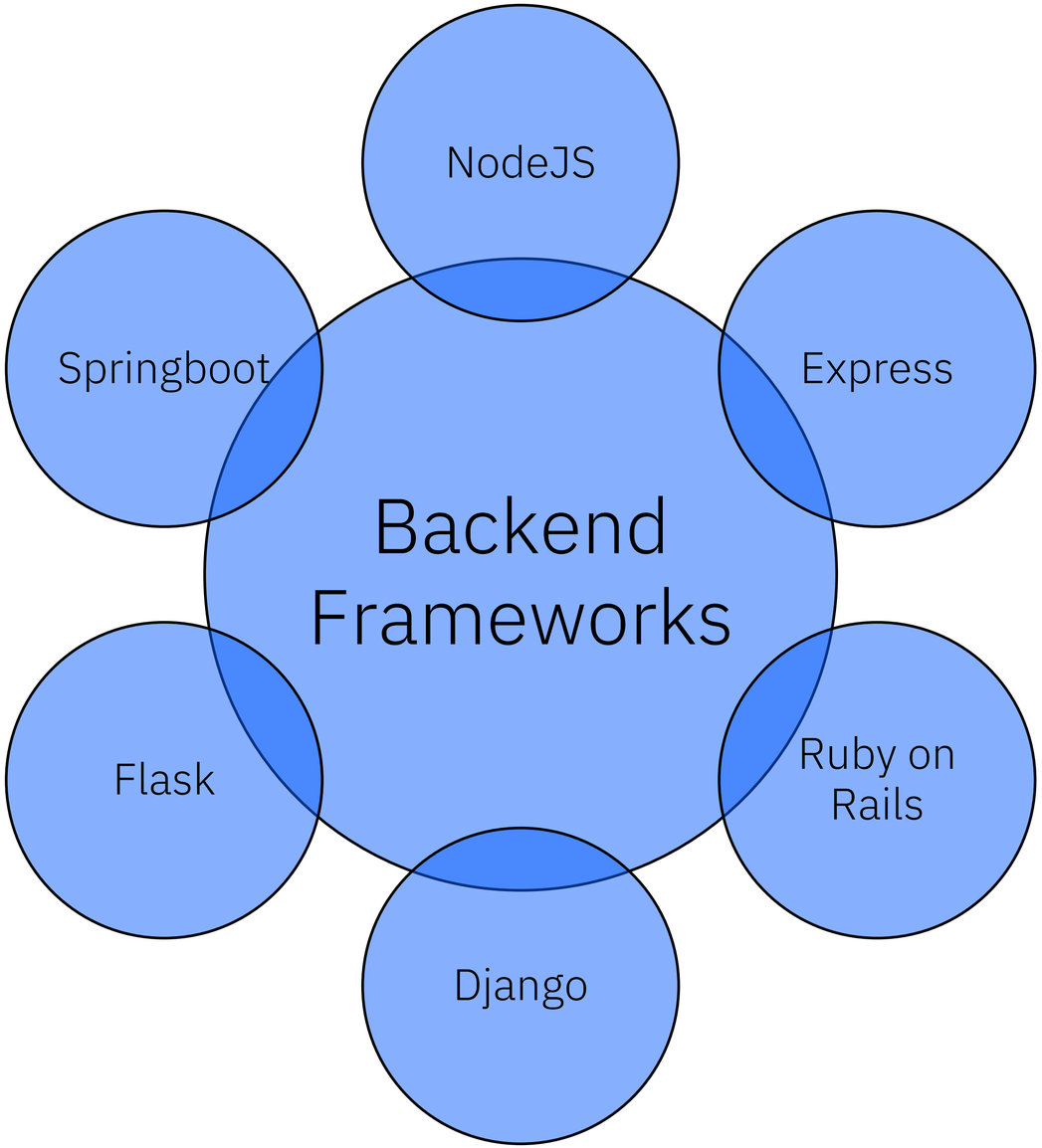
both getting and setting data,

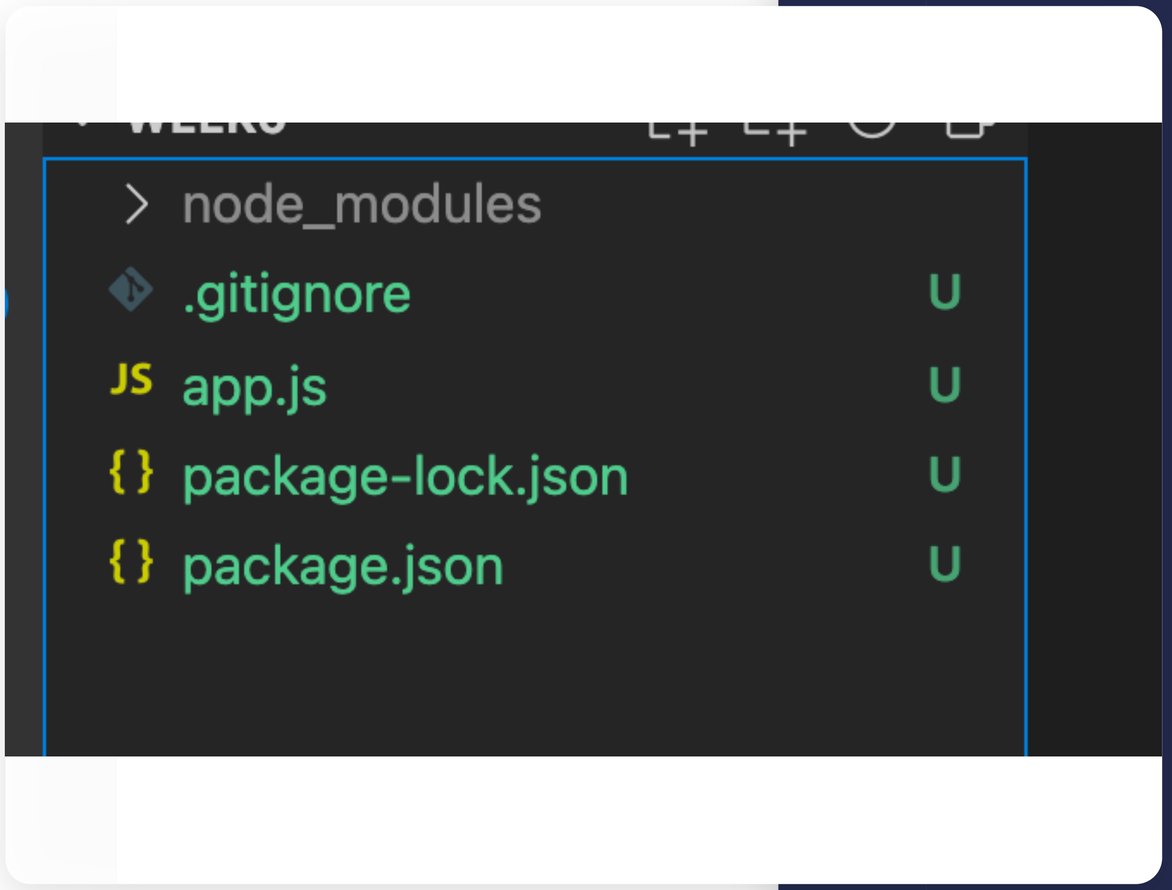
such as retrieving a bank account balance

or depositing money into an account.



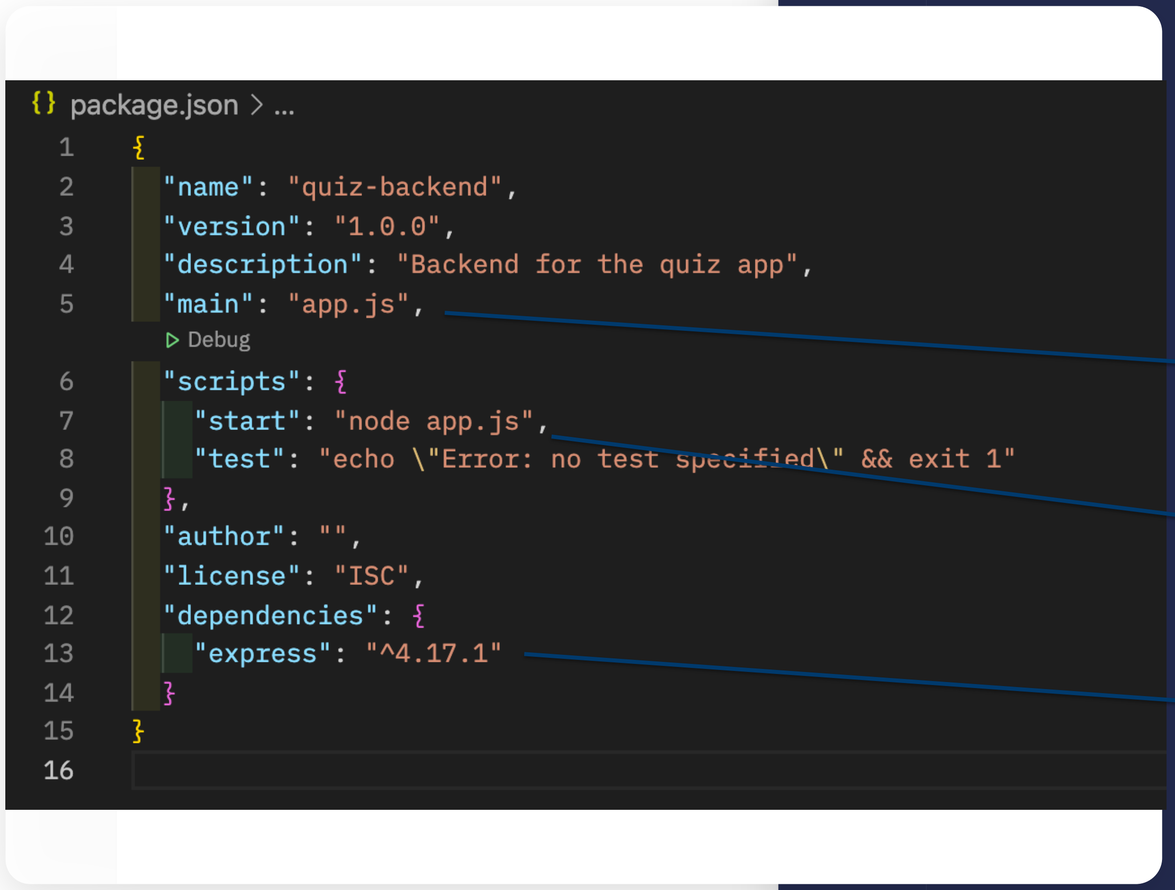
Backend Service Technologies





Code structure

Installed packages and scripts



The main file

Script to run the app

Express installed as a dependency

Lets make a backend Server!

What does this code do?

- Create an app object from express that was imported in line 1
- Set a port where the server would exist
- In app.get, we are saying that when the user asks for ”/” route, send the response defined in line 7.
- Finally start the server on defined port and log the message when it successfully runs.

app.js > ...

```
1   const express = require('express');
2   const app = express();
3
4   const port = 3000;
5
6   app.get('/', (req, res) => {
7     res.send('Hello World!');
8   });
9
10  app.listen(port, () => {
11    console.log(`server started on port ${port}`);
12  });
13
```

npm run start

Routing

app.js

```
10  app.get('/data', (req, res) => {
11    res.send('this would GET the data \
12    from the database')
13  })
14
15  app.post('/create', (req, res) => {
16    res.send('this would POST a new item \
17    to the database')
18  })
19
20  app.update('/data/:id', (req, res) => {
21    res.send('you get get the :id of the \
22    data that you want to update')
23  })
24
25  app.delete('/data/:id', (req, res) => {
26    res.send('you delete data by the :id \
27    and send your confirmation here');
28  })
29
```

Routing

You define routing using methods of the Express app object that correspond to HTTP methods; for example, app.get() to handle GET requests and app.post() to handle POST requests. For a full list, see app.METHOD.

You can also use app.all() to handle all HTTP methods and app.use() to specify middleware as the callback function

CRUD Operations

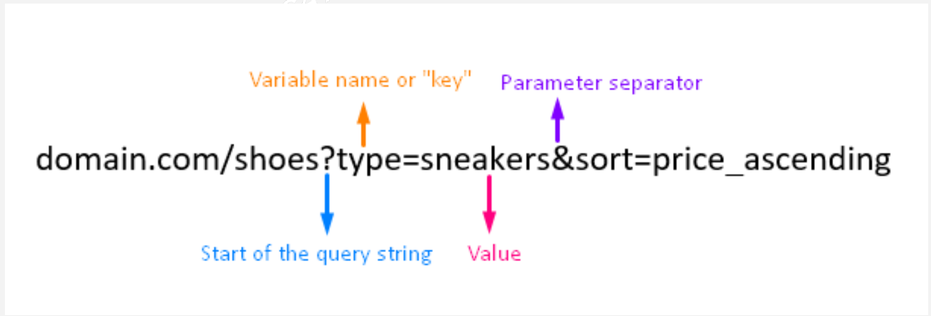
When we are building APIs, we want our models to provide four basic types of functionality. The model must be able to Create, Read, Update, and Delete resources.

In express, this can be done using:

- Get
- Post
- Put
- Delete

Creating JSON API

You need a body parser to convert your response to JSON. Use the JSON middleware like this:



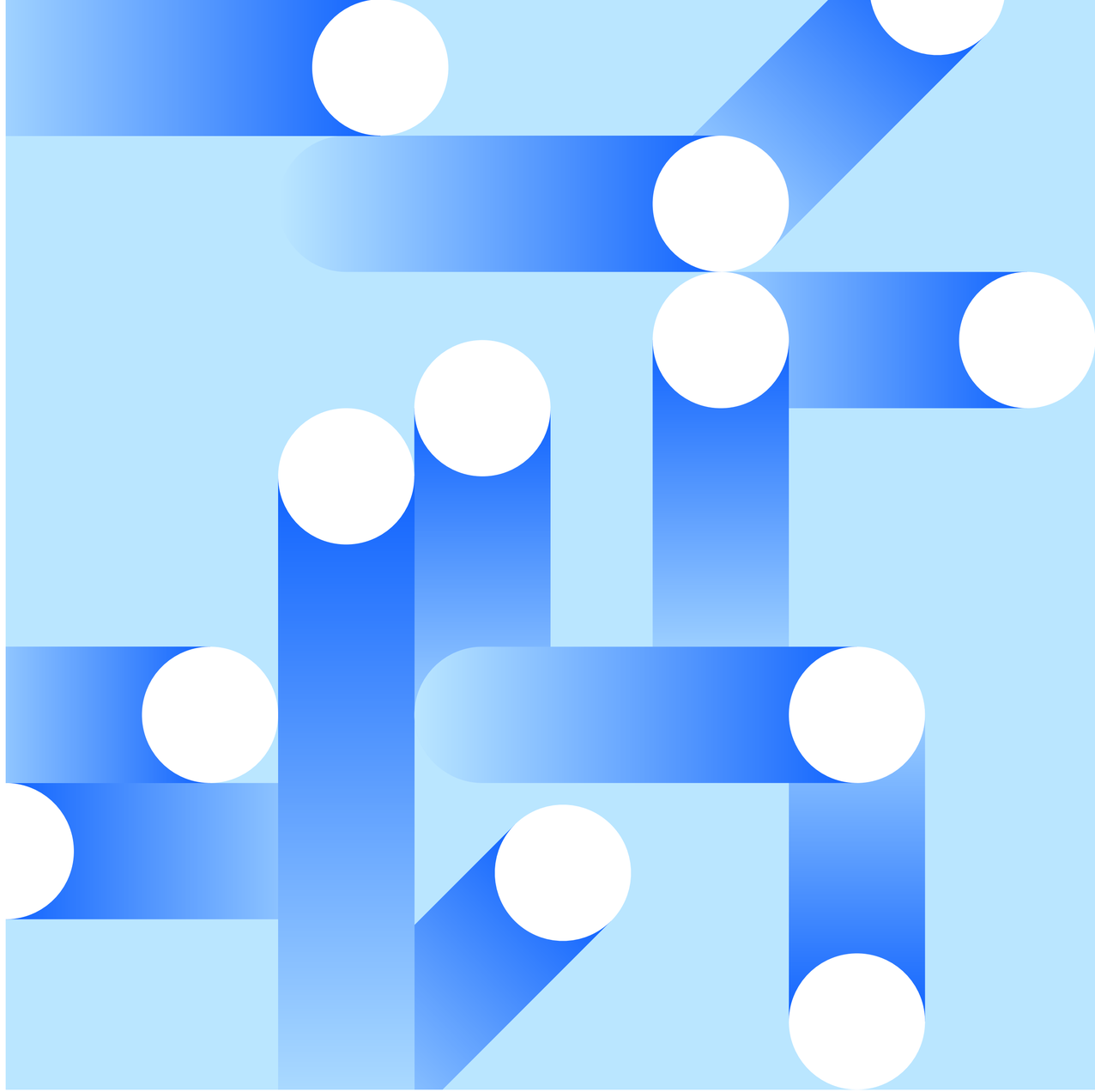

```
$.ajax({
    type: "POST",
    url: "jmtyjmall/api/1.0/activityTask/queryType",
    data: JSON.stringify({}),
    success: function (result) {
        console.log(result);
    },
    error: function (e) {
        console.log(e.status);
        console.log(e.responseText);
    }
});
```

URL & Body Params - Frontend

```
const express = require('express');
const app = express();

const port = 3000;

app.use(express.json());

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.get('/data', (req, res) => {
  res.send('this would GET the data \
  from the database')
})
```

Express - Backend

SDKs and APIs



SDK

A Software development kit (SDK) is a collection of software development tools.

API

An application programming interface (API) allows your app to interact with an external service a simple set of commands.

Fun facts

- SDK can contain APIs
- Some APIs have SDK options
- Most software use SDKs and APIS
- They save development time and effort

An Analogy

SDK represents the entire house: all of the rooms, furniture, telephone lines, and other components. An API represents just the telephone lines that allow communication in and out of the house.

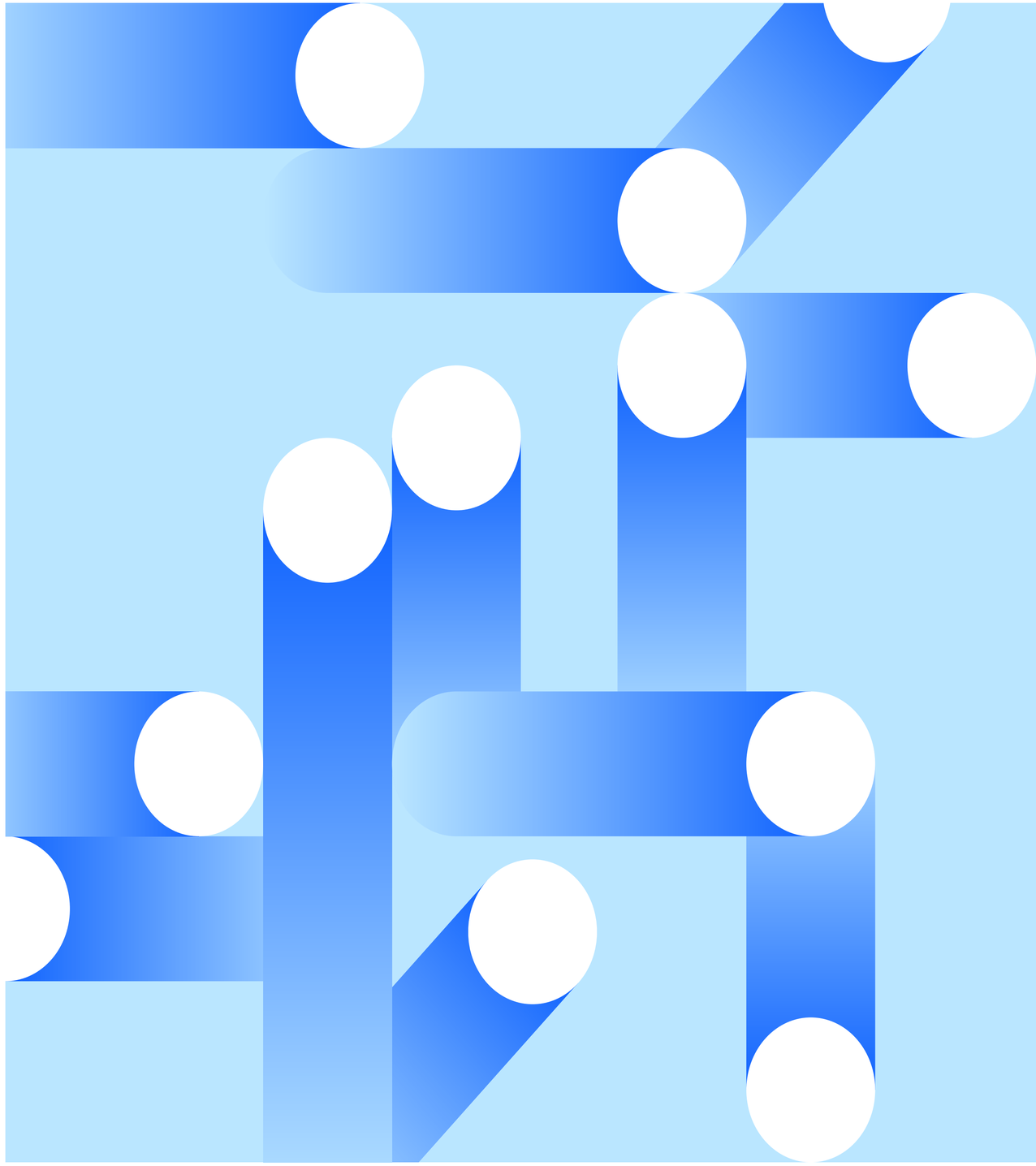
Firebase

IBM Cloud

Quiz Time

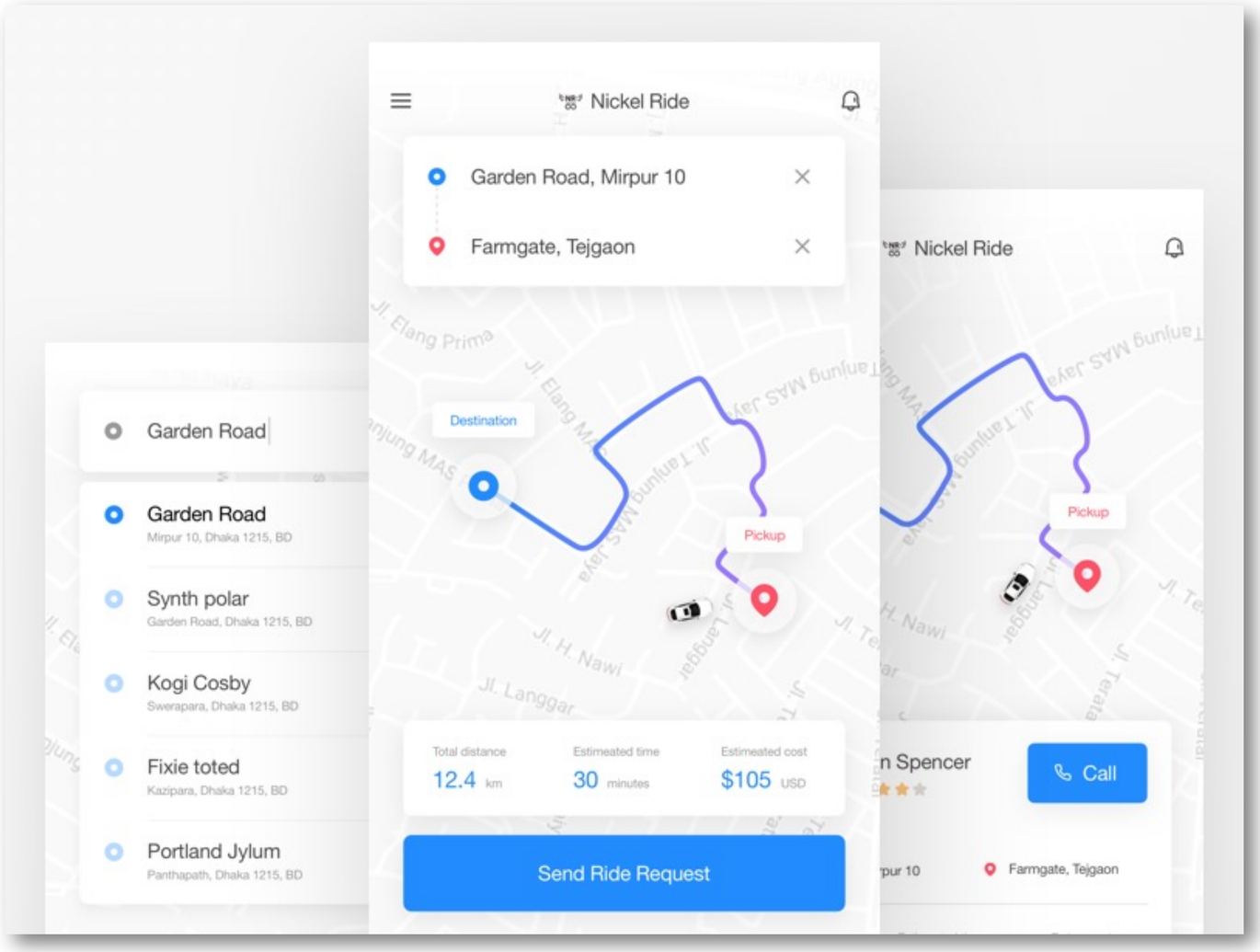
True or False Question:

An SDK typically includes tools for debugging, testing, and building applications.



Imagine a ride sharing application

Think of the things happening in the app and how APIs can be used to make a fully functional application



APIs used in a Ride-sharing application

SMS API like Nexmo
Notify you that the driver has arrived at the pickup location.

Telesign
Verify that you own the phone number you provided at sign-up.

Stripe
When it's time to pay, your payment is likely processed with an API like Stripe

Maps API
Calculating the time and distance of the ride

REST APIs

- **REST aka Representational State Transfer**, is an architectural style for structuring APIs.
- Can use different formats for sending data: XML, JSON, plain text, HTML
- Rest has the following requirements:
  - Uniform interface.
  - Client-server decoupling.
  - Statelessness
  - Cacheability
  - Layered system architecture
  - Code on demand (optional)
- Rest APIS respond to HTTP requests
  - Get
  - Post
  - Put
  - Delete
- OpenAPI- a standard for REST APIs self-documention and auto-generate code to use REST APIs. I.e. create an OpenAPI compatible description of your REST API and you can use it with a wide number of clients and code with little effort

SOAP APIS

- Simple Object Access Protocol (SOAP)
- Depends heavily on XML, strongly typed schemas, a WSDL is a contract
- Differences with REST
  - Independent of transport (REST requires HTTP)
  - SOAP is rigid in requests/responses structure
  - Great for separate of concerns and distributed environments
  - Error handling is tightly integrated to functionality
  - Extended functionality (extra processing can be done)

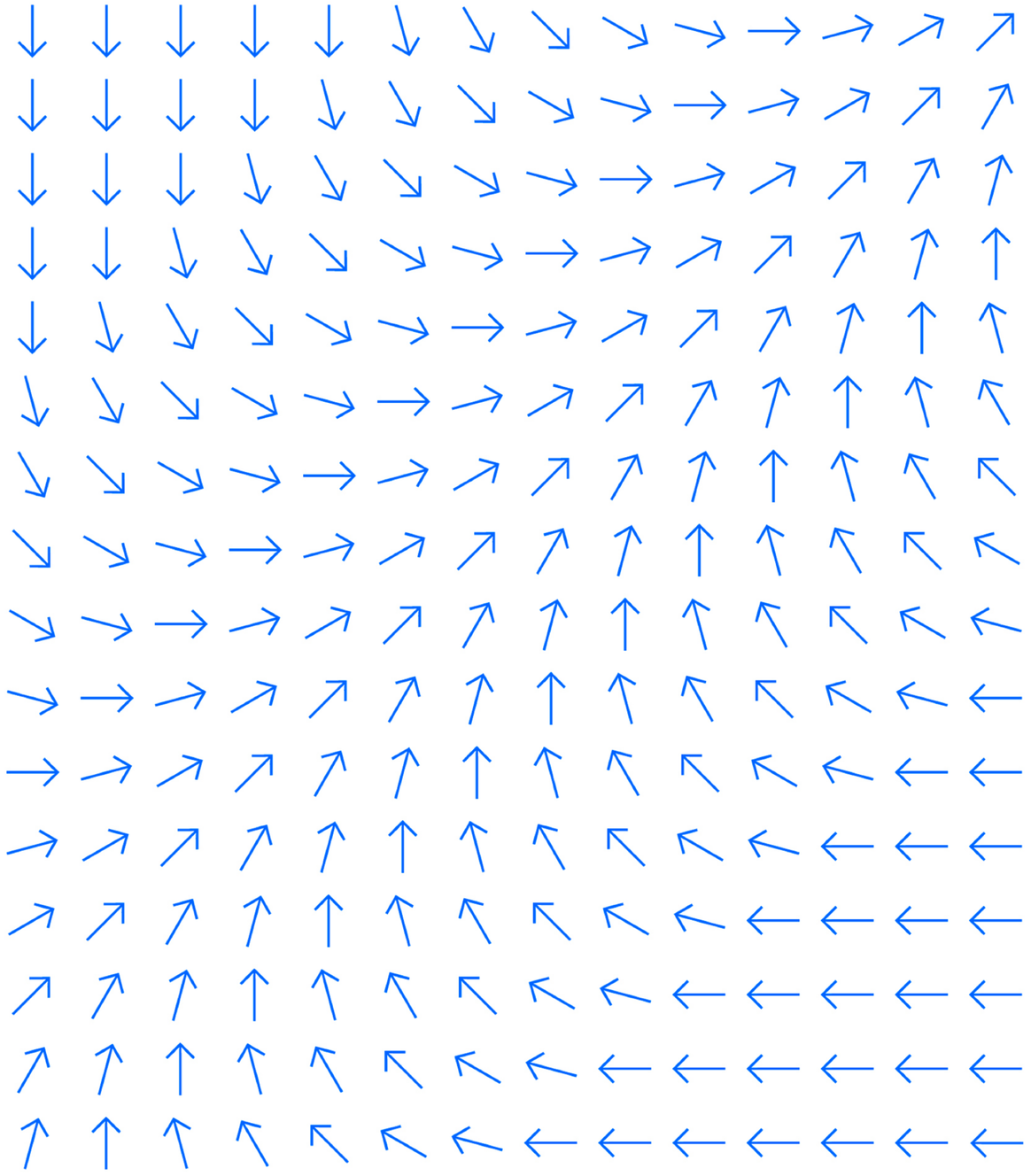
RPC/GRPC

- Remote Procedure Call (RPC)
  - Generic for all protocols
  - Request/Response protocol meant for communication with distributed systems
  - A client sends a request to a remote server
    - Client is blocked until response is received
    - Unless we are asynchronous – doing stuff while waiting for a response
- Google Remote Procedure Call (GRPC) – a modern open-source RPC framework that can be used to create services that can be called by clients remotely
  - gRPC is specific to HTTP protocol
  - Stubs abstract the usage of the HTTP protocol

GraphQL

- A query language for APIs
- You can specify what to ask for, and it’s exactly what you get (minimal query, minimal results) - efficiency
- Allows for the evolution of your application/queries over time, queries get more complex, more fields needed

Architectural Patterns for Backend Services



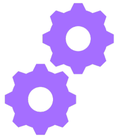
IBM

IBM Accelerate 2024

How did we get here? Aka Application Architecture Evolution

Single Servers - Monolith

Vertically Scaled



Separation of services

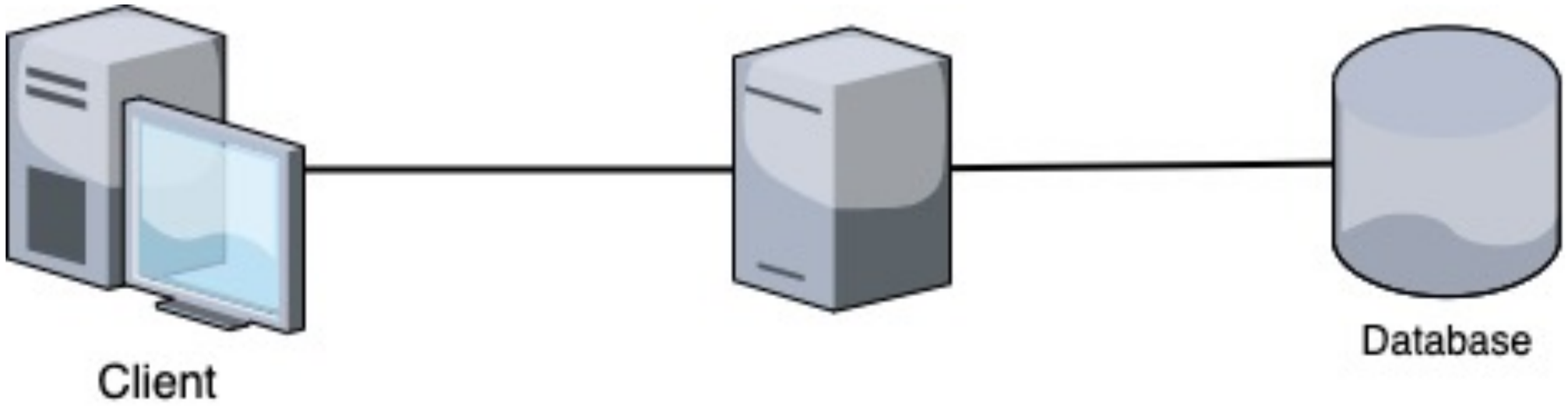
Each service in it’s own machine

CDN

Application/Functional separation

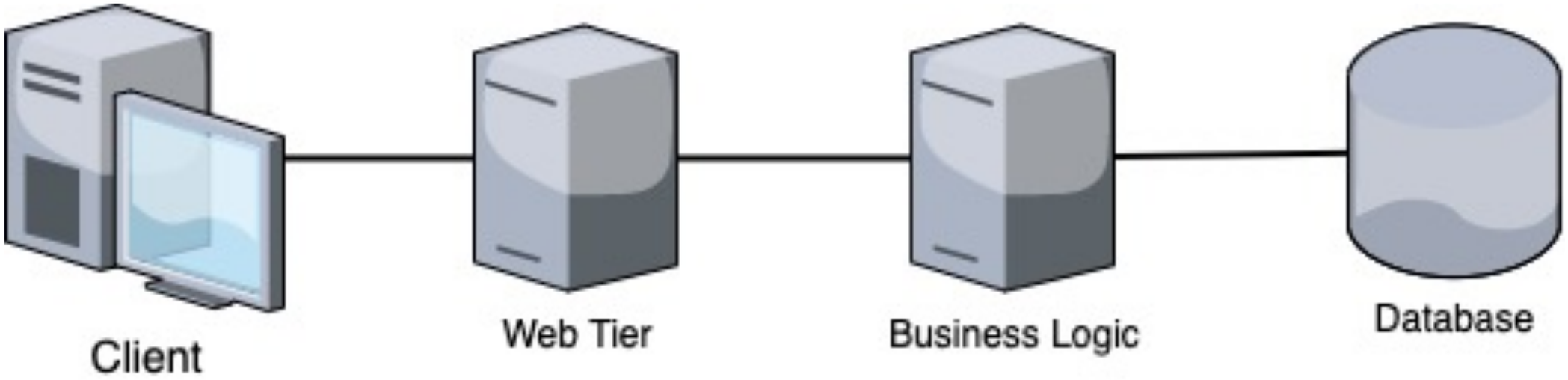
Global architectures

Edge, geo load balancing



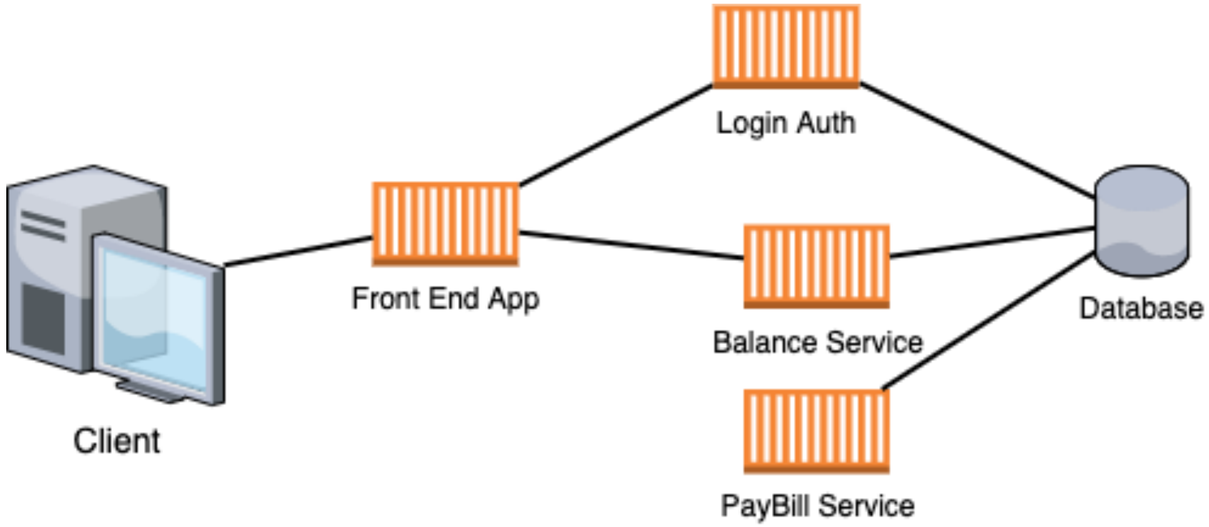
Single Server Monolith

- Pros:
  - Easy to setup
  - Cheap at first - less infrastructure and tooling
- Cons:
  - All services are connected
  - Update one component you have to bring it all down



Separation of Services

- Pros:
  - Separation allows for isolation of service maintenance, blast radius of outages
- Cons:
  - Larger topology of servers and services, still have a single point of failure for the services



Application/Functional Separation

- Pros:
  - Separation allows for independence
- Cons:
  - Larger topology of servers and services

Application Architectures

Layered

Microservices

Serverless

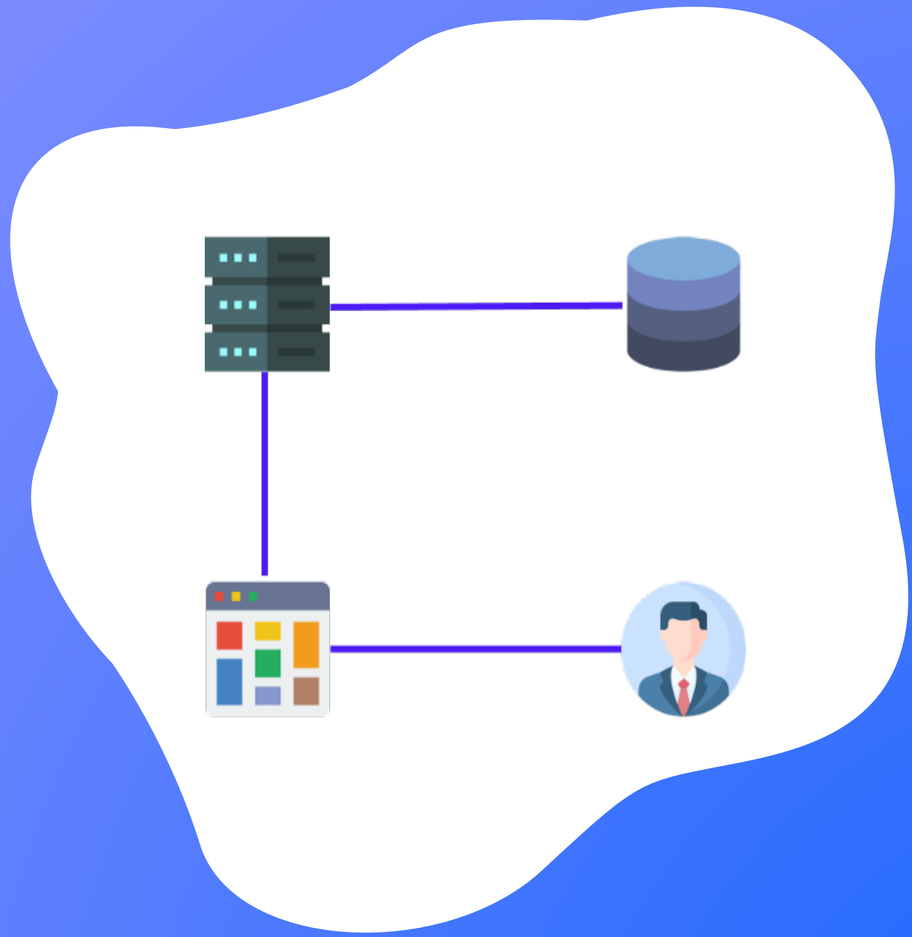
Messaging

SOA

Client Server

MVC - Model View Controller

Facebook's data (model) is stored in a database which is accessed from a server (controller). The controller is responsible for communication data transfer from the model and the UI (view) which we see when we go to facebook.com.

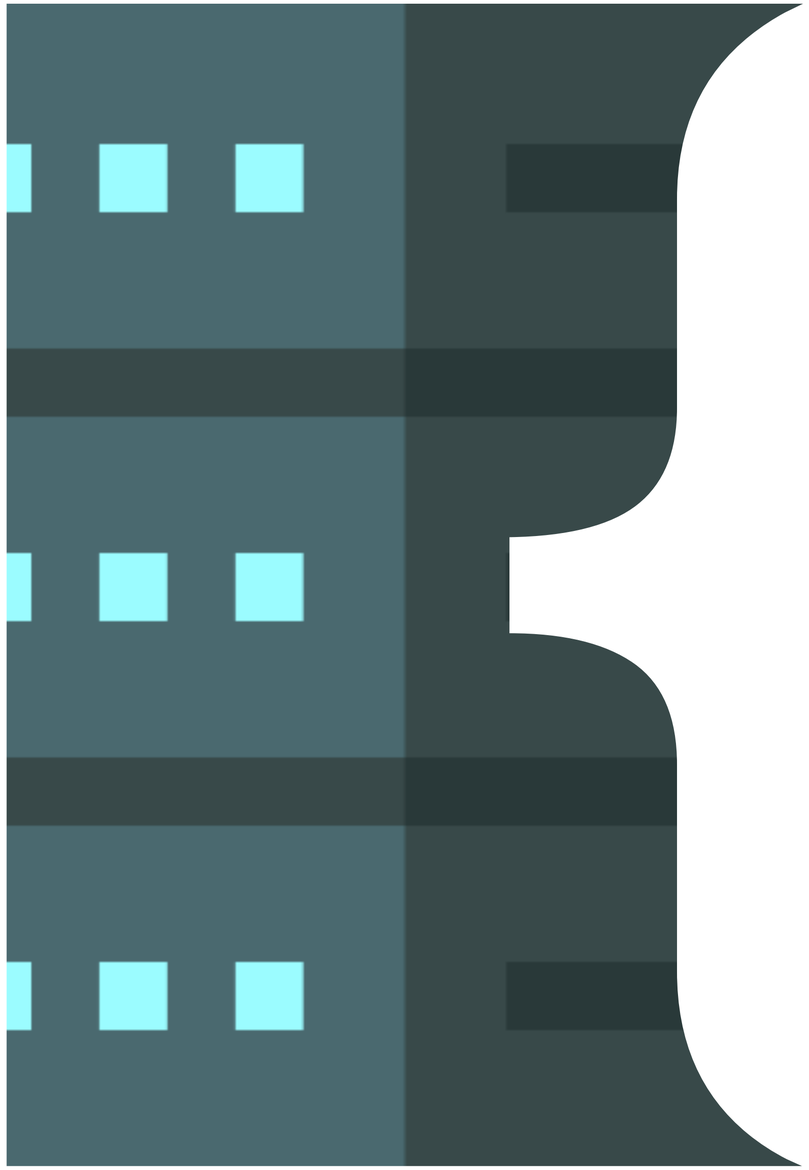


Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data.

View

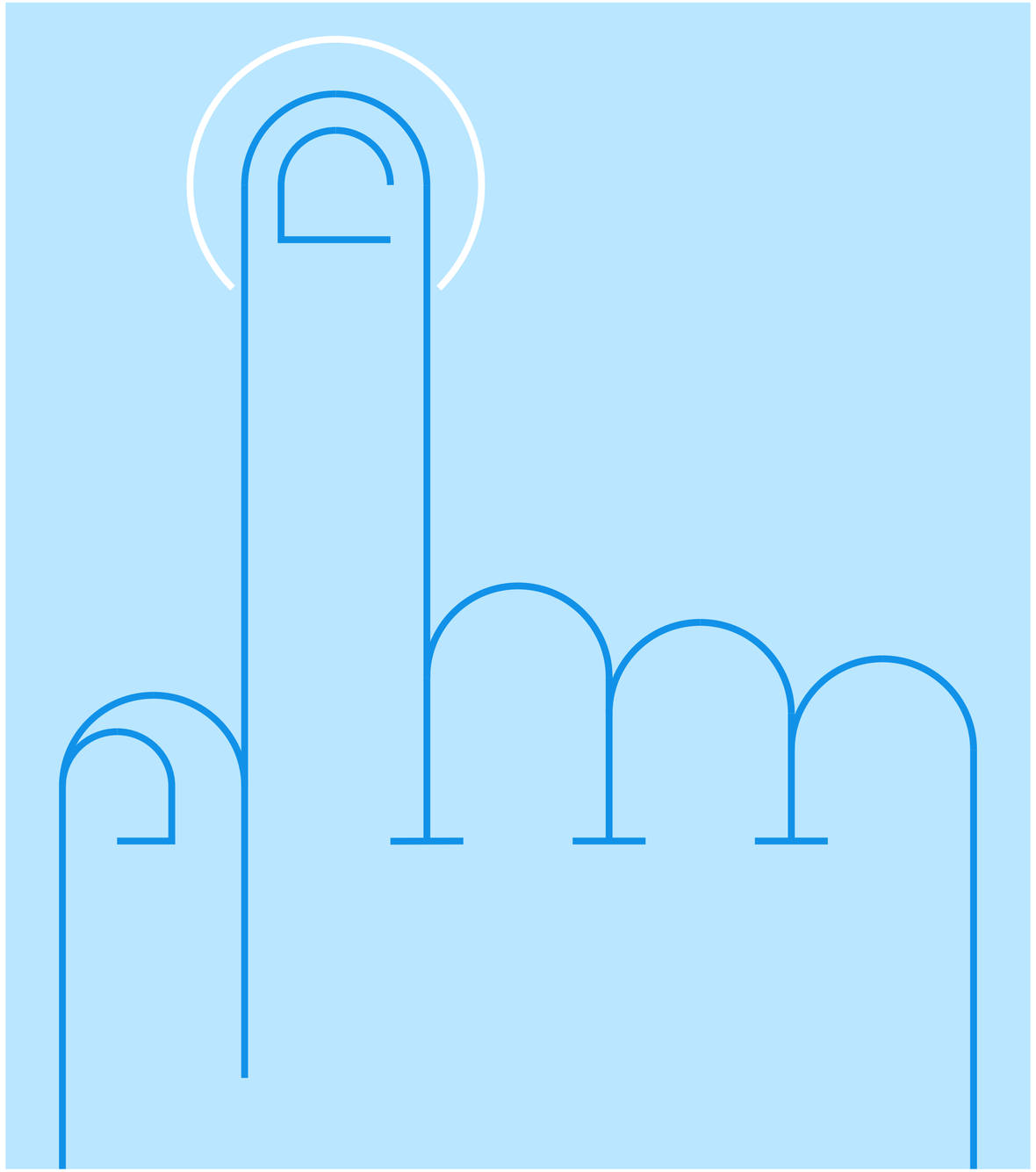
The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc., that the final user interacts with.



Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component, and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

Testing



Unit tests

Unit tests are typically **automated tests** written and run by software developers to ensure that a section of an application meets its design and behaves as intended. A **unit** could be an entire module in procedural programming, but it is more commonly an individual function or procedure.

Benefits of unit testing

- You can change, add old code faster if the code is properly tested
- Finds software bugs easily (example in the next slide)
- Helps in integration with other code.
- Provides documentation of what the function should do
- Forces you to think about the design of your software
- Reduces business costs

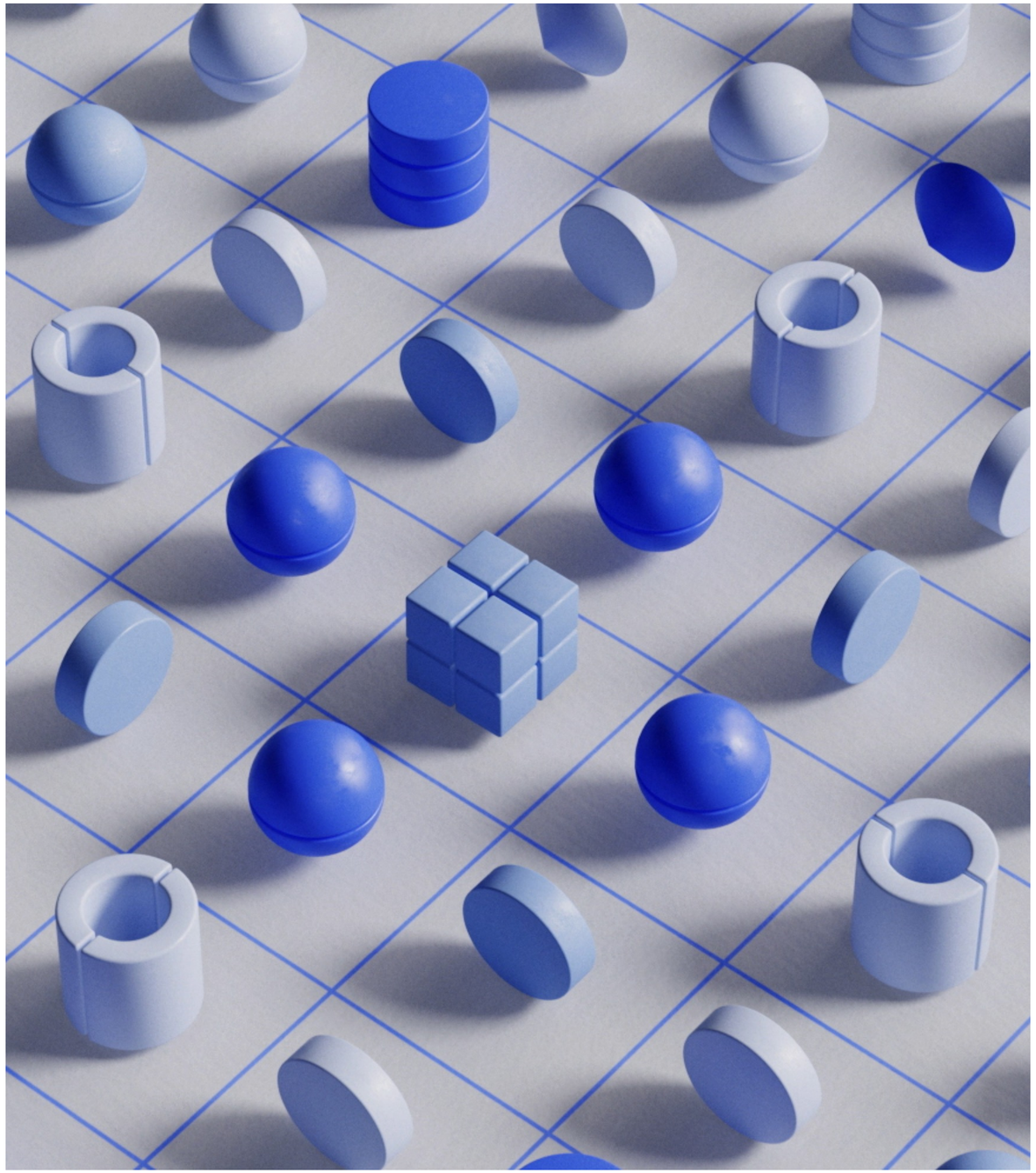
Other Types of Testing

- Integration - These tests run when components are “integrated” together
- Smoke/Build - Tests run when we build software, if it’s smoking (lots of tests are failing) we don’t want to use that build, or take it with known risks
- System – denote tests that perform and end-to-end test of the whole system, simulate a user flow, the entire use of all components, but with an environment similar to a real customer deployment of our application
- Security – tests to focus on security vulnerabilities, ways to hack into the system, get and access protected data and privileges
- Load – a “load” is a work that is sent to your application or software, the goal being it is a prolonged set of work to put the system in a constant state of use, this is to identify things like memory and thread leaks, and ensure the system can tolerate a steady state of work
- Performance – a performance test runs a varying set of work loads against the system to measure how well the system performs, this could be rendering time, transaction throughput etc.
- Stress – put an extreme amount of work against the system and see how it performs, this is to stress memory, CPU, or disk utilization to high extremes, make sure the system doesn’t fall over or crash, and can recover

Continuous Delivery/Integration/Pipeline tests

- Testing for most development organizations will involve some type of continual test run against every build or every deployment.
- This ensures best practices – identifying build failures early, pinpointing failures, ensuring healthy builds
- A pipeline of testing ensures that there is no human intervention required to promote something to production, the tests from one environment to another should have to pass for something to get promoted to production via automation. This satisfies security compliance and performance optimization requirements.

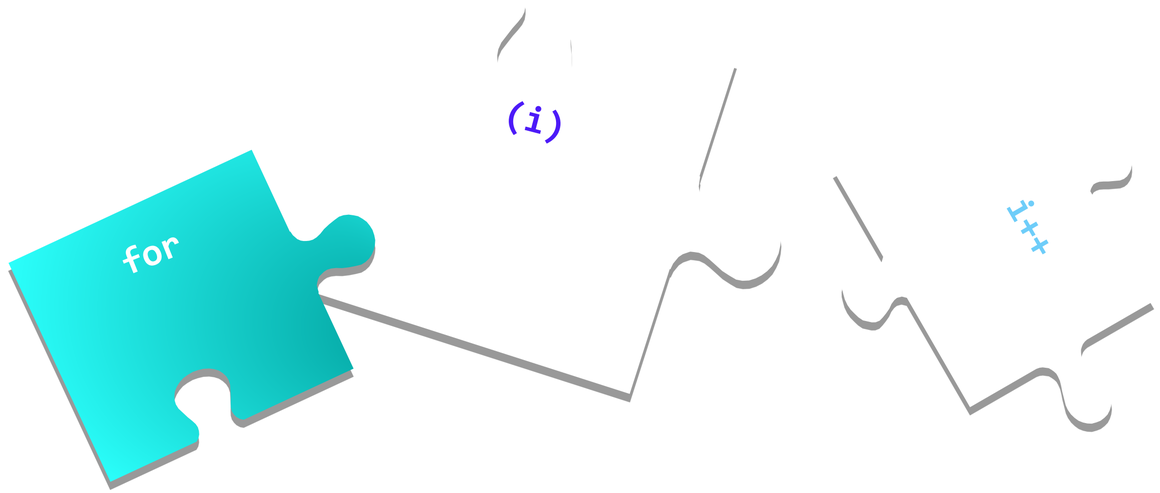
Week 6 – Lab Todo List



Lab: ToDo List Backend Service Implementation

In this lab you'll be developing backend services that provide new functionality for our ToDo list.

You will be developing two new features in your ToDo List backend services. These services can be utilized by any consumer of the ToDo backend services like a UI or other application.



ToDo List: Retrieve all Tasks from the DB

Feature requirements:

- A user needs to retrieve all Tasks from the backend
- Test via CLI

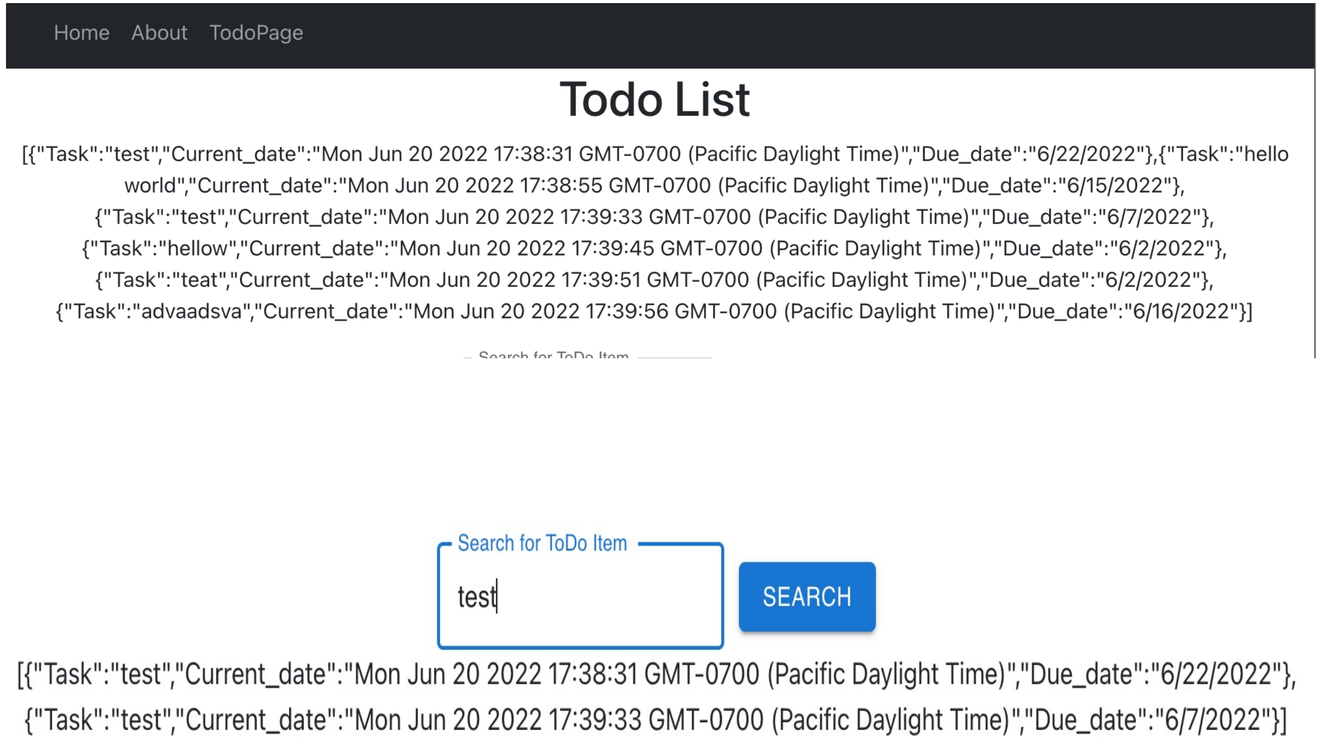
ToDo List: Search for a Task

- User wants to search for a task with a specific name
- Search for and return all tasks matching the name (I.e. the name doesn’t have to be unique)

- Test via CLI

Bonus: ToDo List: Utilize in the UI Application

- When you open a page return all the Todo's stored in the database, I.e. populate with current list of Todos
- Add a search button and return the task list for a user's search field on button click



Future Task (Optional) - ToDo List: Use Cloudant as the backend data store

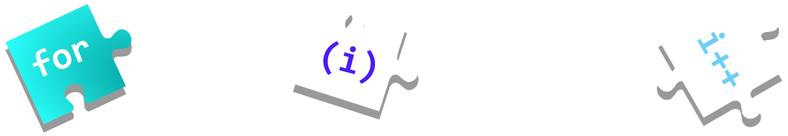
Note: You'll need an IBM Cloud account to do this

Feature requirements:

- Utilize a Cloudant DB to store Todos
- Retrieve all Todos from Cloudant
- Search for Todos from Cloudant

Lab Preview

API Endpoint	Parameters	Return Value	Example
/get/items	None	Json list of tasks	http://localhost:8080/get/items
/get/searchitem	taskname	Json list of tasks	http://localhost:8080/get/searchitem?taskname=test



Reading Material

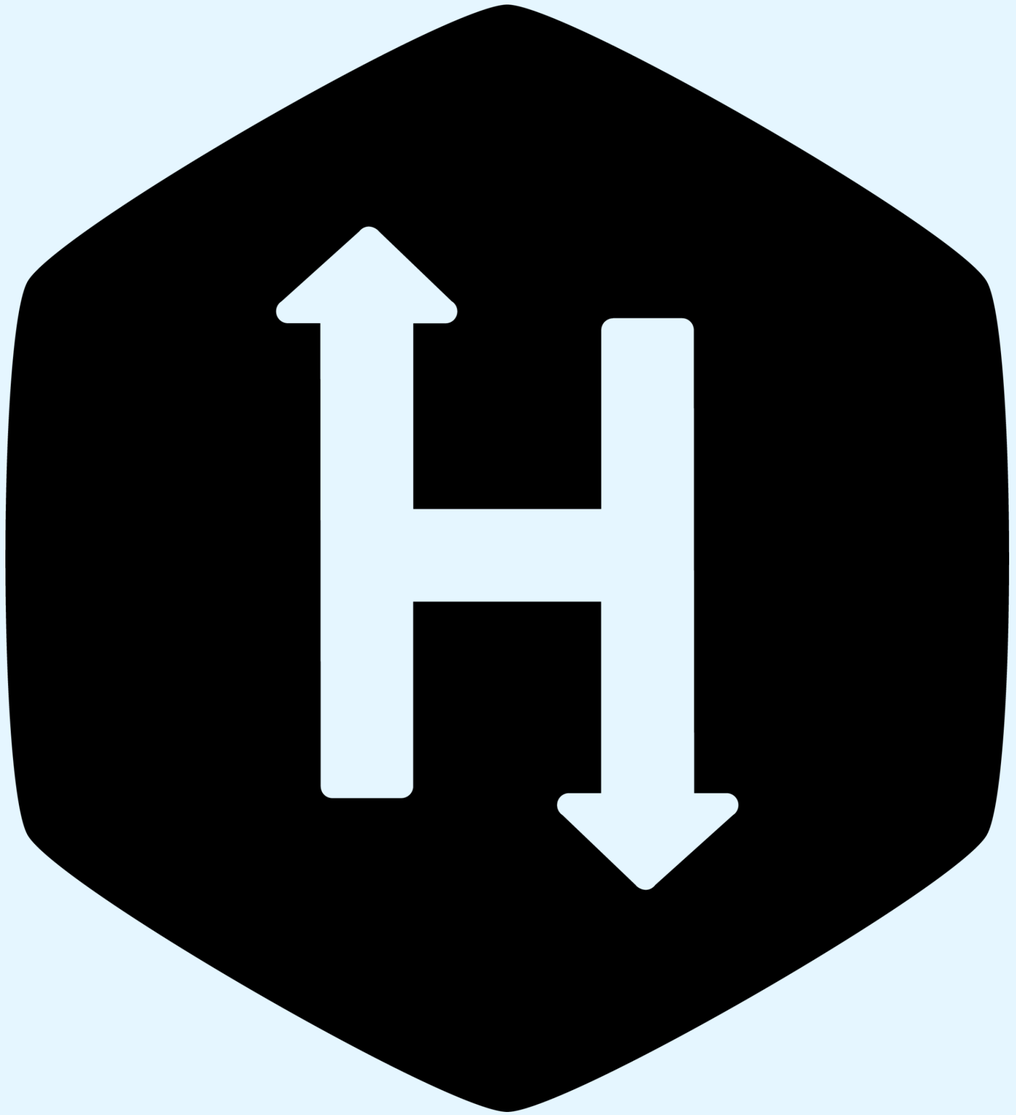
- What is a REST API
  - https://www.redhat.com/en/topics/api/what-is-a-rest-api
- Rest APIS
  - https://www.ibm.com/cloud/learn/rest-apis
- Microservices Architecture
  - https://www.ibm.com/cloud/architecture/architectures/microservices
- Modernizing Applications
  - https://www.ibm.com/cloud/architecture/architectures/application-modernization

Thank you from your Week 6 Team!

Marc Velasco
STSM Site Reliability Engineering - MultiCloud SaaS Platform - Leader of Southern California Familia

Mishika Singh Gaur
Multi Cloud SaaS Platform – Cloud Infrastructure
Backend Developer

Required HackerRank Test



IBM

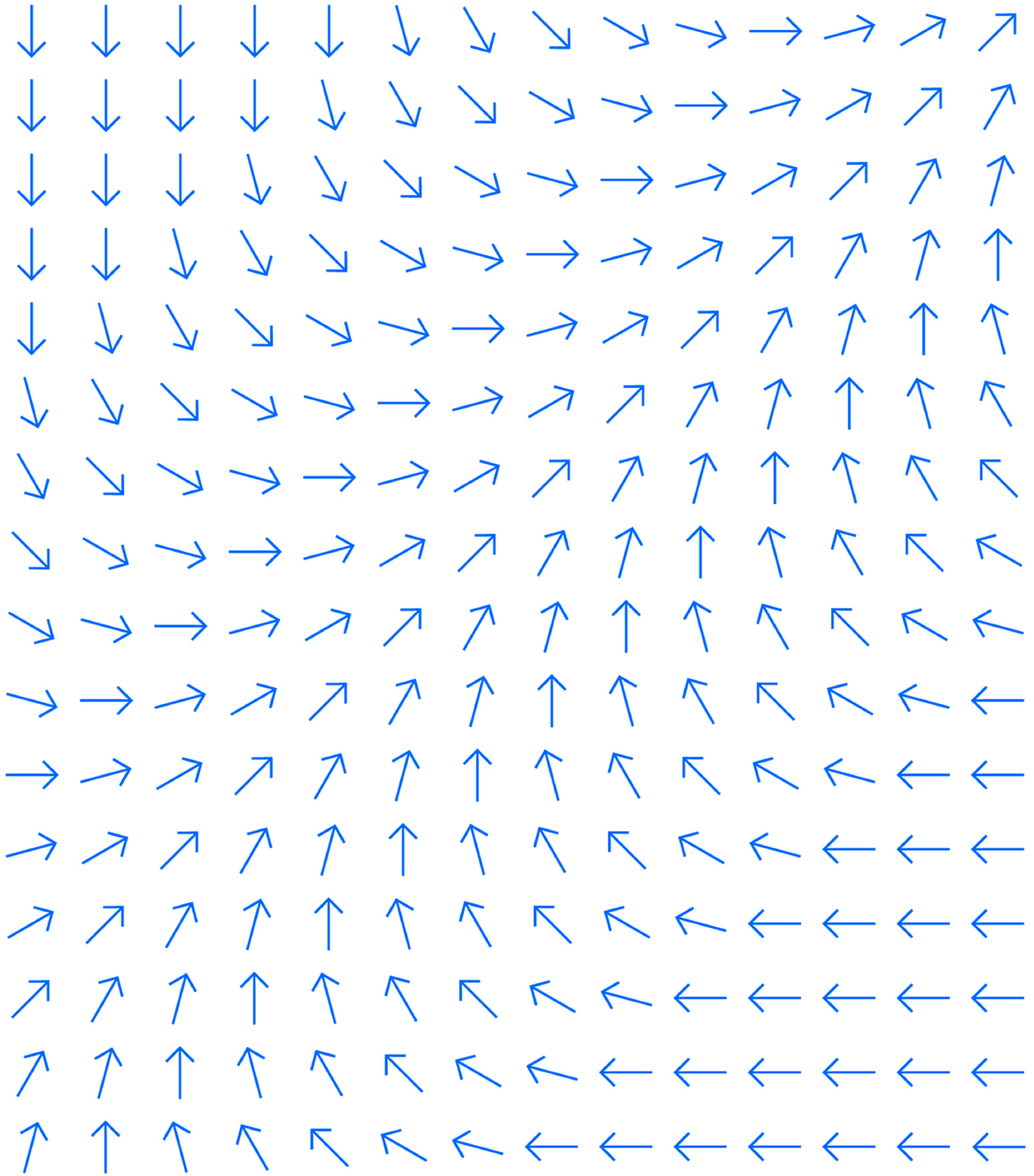
Required HackerRank Test for Week 6

- Tomorrow (Thursday), you should receive an email containing a link to a HackerRank quiz covering content from Weeks 4 and 6.
  - From IBM Corporation Hiring Team – support@hackerrankforwork.com
- Test will include **6 questions**
- Required to earn the IBM Accelerate Badge
- Passing score: 4/6
- Due by next Tuesday, July 16 11:59 PM ET

Required HackerRank Test: Preparation

- Review labs and presentations from prior weeks
- Complete optional HackerRank tests from weeks 4 and 6
  - Same form and style as required test
- Use Slack and/or Office Hours to answer any questions

Next Steps



Next Steps

1. Finish your lab (https://ibm.biz/accelerate-sw-week6) by Monday, July 15, 11:59 PM ET.
2. If you need further help, ask in the slack channel (#ibm-accelerate-2024-software) or join the office hours (Thursday at 6:00 PM ET).
3. **Complete the required HackerRank test by Tuesday, July 16, 11:59 PM ET.**
4. Check out the solutions for the lab and HackerRank test once available.
5. Slides and video recordings of the session are available in our Box folder.
6. Watch for Week 7 materials to start to arrive by Monday.
7. Network and connect with your peers today and after today's session.