Keretapi Tanah Melayu Berhad APIM Project

API Development

22 - 23 May 2025

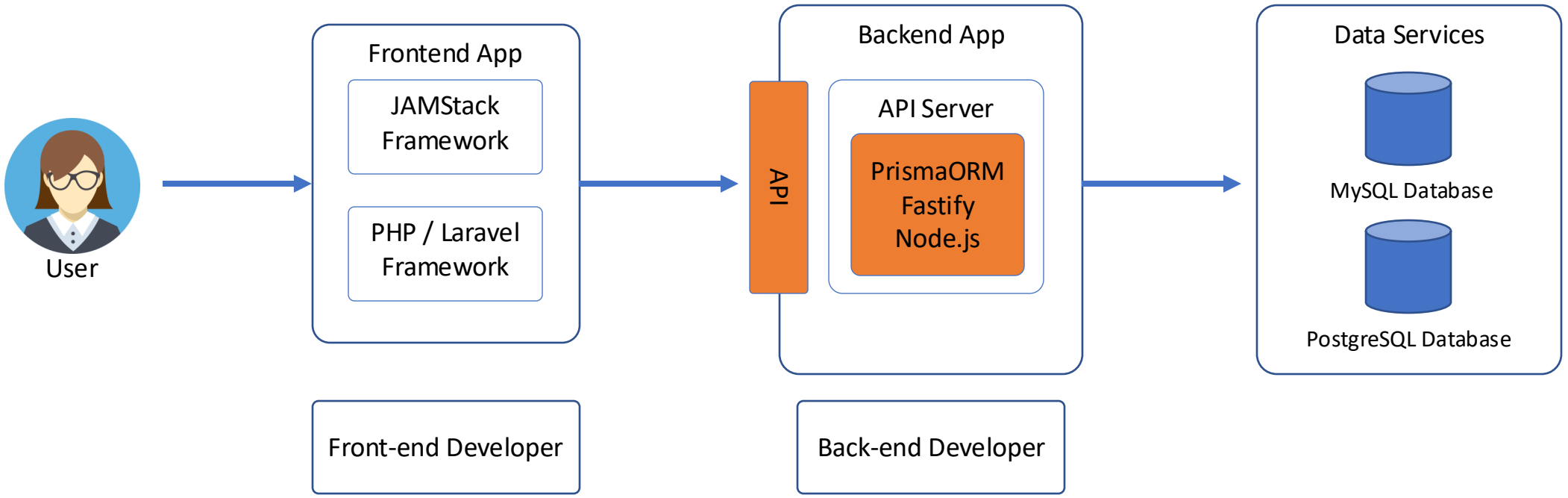
Objektif Program

- Untuk meningkat-kemahiran (up-skilling) dan kompetensi ICT Pegawai Kerajaan bagi merealisasikan aspirasi inisiatif MyDigital, Dasar 4iR negara dan Strategi *Cloud-First*.
- Untuk memahir-semula (re-skilling) tenaga kerja semasa agar kompetensi pegawai sektor awam terus relevan.
- Penerapan minda dan budaya kerja
- Ketersediaan Kompetensi dan Skillset yang perlu untuk industri

Objektif Latihan

- Menggunakan teknologi dan seni bina:
  - *Javascript & NodeJS*,
  - *Backend API* menggunakan *Fastify*
- Mendemonstrasikan kefahaman konsep pembangunan aplikasi moden: *Backend API*

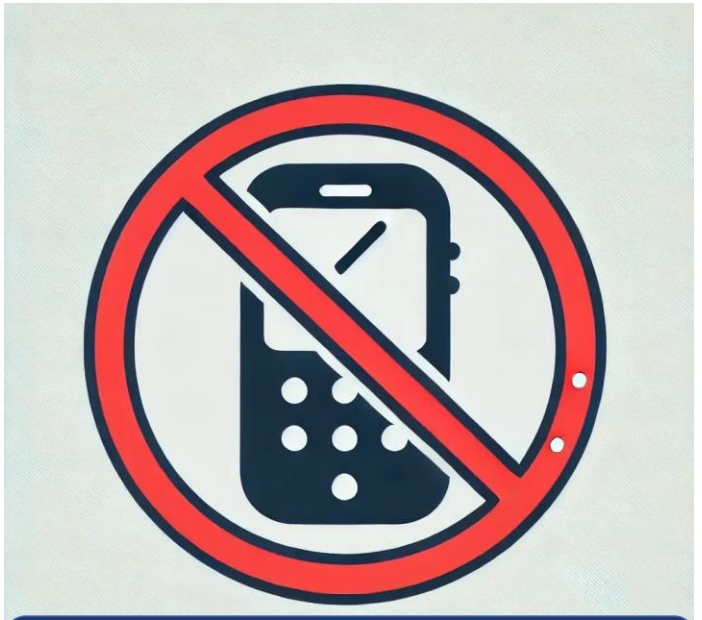
Skop Latihan



Kaedah Latihan

- Penyampaian dalam Bahasa Melayu / Bahasa Inggeris.
- Kandungan dan bahan latihan dalam Bahasa Melayu / Bahasa Inggeris.
- Latihan secara amali (*hands-on*).
- Panduan terperinci dengan dokumen langkah-demi-langkah (*step-by-step*).
- Latihan secara langsung menggunakan pengkomputeran awan jika perlu

Punctuality



No Handphone

Participate

Masa

9:30 AM	Mula
10:45 AM	Minum Pagi
12:00 PM	Makan Tengahari / Rehat / Solat
02:00 PM	Sambung Sesi
04:30	Minum petang / Tamat

Syarat Tambahan

- Menyediakan komputer dengan ciri-ciri seperti berikut:
  - Minimum ***RAM 8GB***
  - Minimum **storan 10GB**
  - Mempunyai ciri kamera web dan mikrofon
  - Menggunakan minimum sistem operasi *Windows 10 64-bit* atau *MacOS 11 Big Sur*
  - Siap dipasang aplikasi ***Visual Studio Code***
  - Mempunyai Hak Akses Sebagai Pentadbir (*Administrator Access Rights*) terutamanya jika menggunakan komputer yang dibekalkan oleh organisasi anda; i.e. anda tidak dihalang daripada memasang perisian di komputer tersebut.

Integrated Development Environment (IDE)

Visual Studio Code

- **Visual Studio Code or VS Code** is a code editor redefined and optimized for building and debugging modern web and cloud applications.
- a **free** source-code editor made by **Microsoft** for Windows, Linux and macOS.

https://code.visualstudio.com/download

Agenda – Hari 1

API Development

Time		Activities & Deliverables
From	To	
09:30	9:45	• Pengenalan Latihan
09:45	10:00	• Pengenalan kepada teknologi *Cloud Native*
10:00	10:15	• Pengenalan kepada *Microservices & API, JSON*
10:15	10:30	• Pengenalan kepada *Javascript & NodeJS*
10:30	10:45	• Pengenalan kepada Fastify
10:45	11:00	• Rehat

11:00	12:00	• *Fastify In-Depth & Exercises*
		  • *Logging*
		  • *Environment Variables*
		  • *Error Handling*

12:00	14:00	• Makan Tengah Hari & Solat

Agenda - Hari 2

API Development

Time		Activities & Deliverables
From	To	

14:00	15:30	• *Fastify In-Depth & Exercises*
		  • *Routing*
		  • *Validation & Serialization*

15:30	16:30	• *Fastify In-Depth & Exercises*
		  • Pengenalan kepada *ORM*
		  • Pengenalan kepada *Prisma ORM*

Agenda - Hari 2

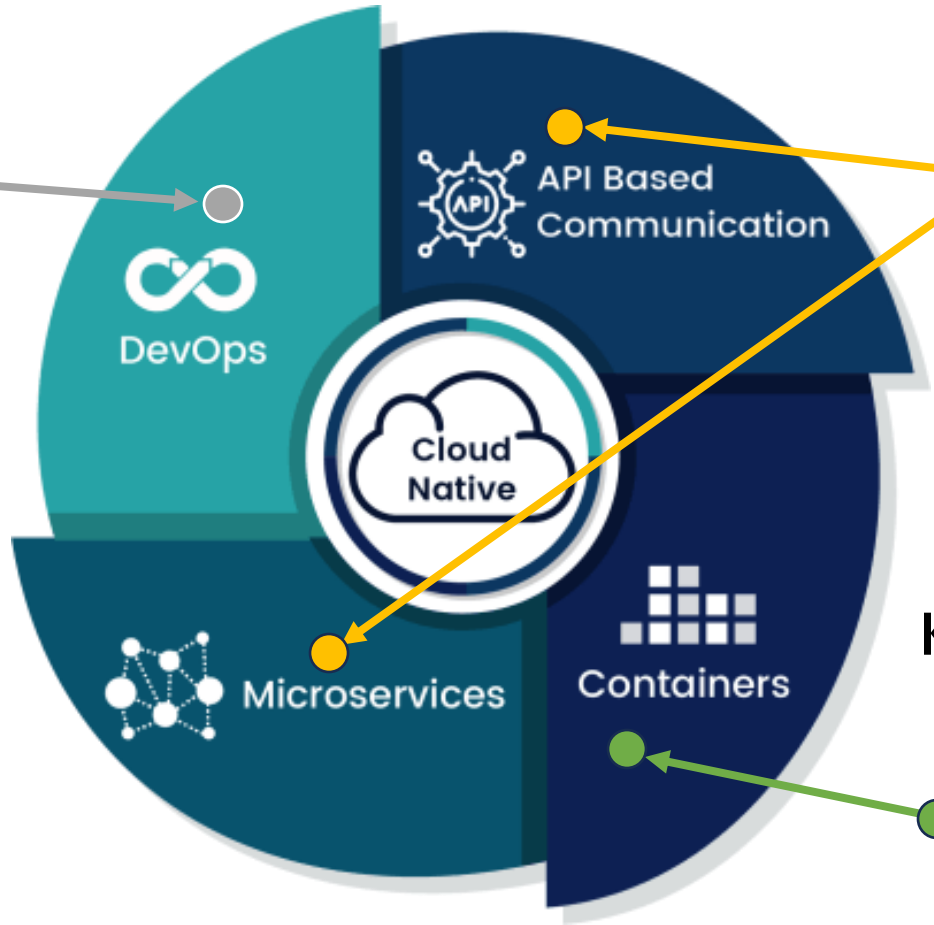
API Development

Time		Activities & Deliverables
From	To	
9:30	4:30	• Latihan Pembangunan API dari MySQL Database sediada • Latihan Pembangunan API dari MySQL Database baru

{ *Cloud Native* }

Apa itu *Cloud Native*?

Automasi &
Budaya Kerja



Pembangunan
Aplikasi

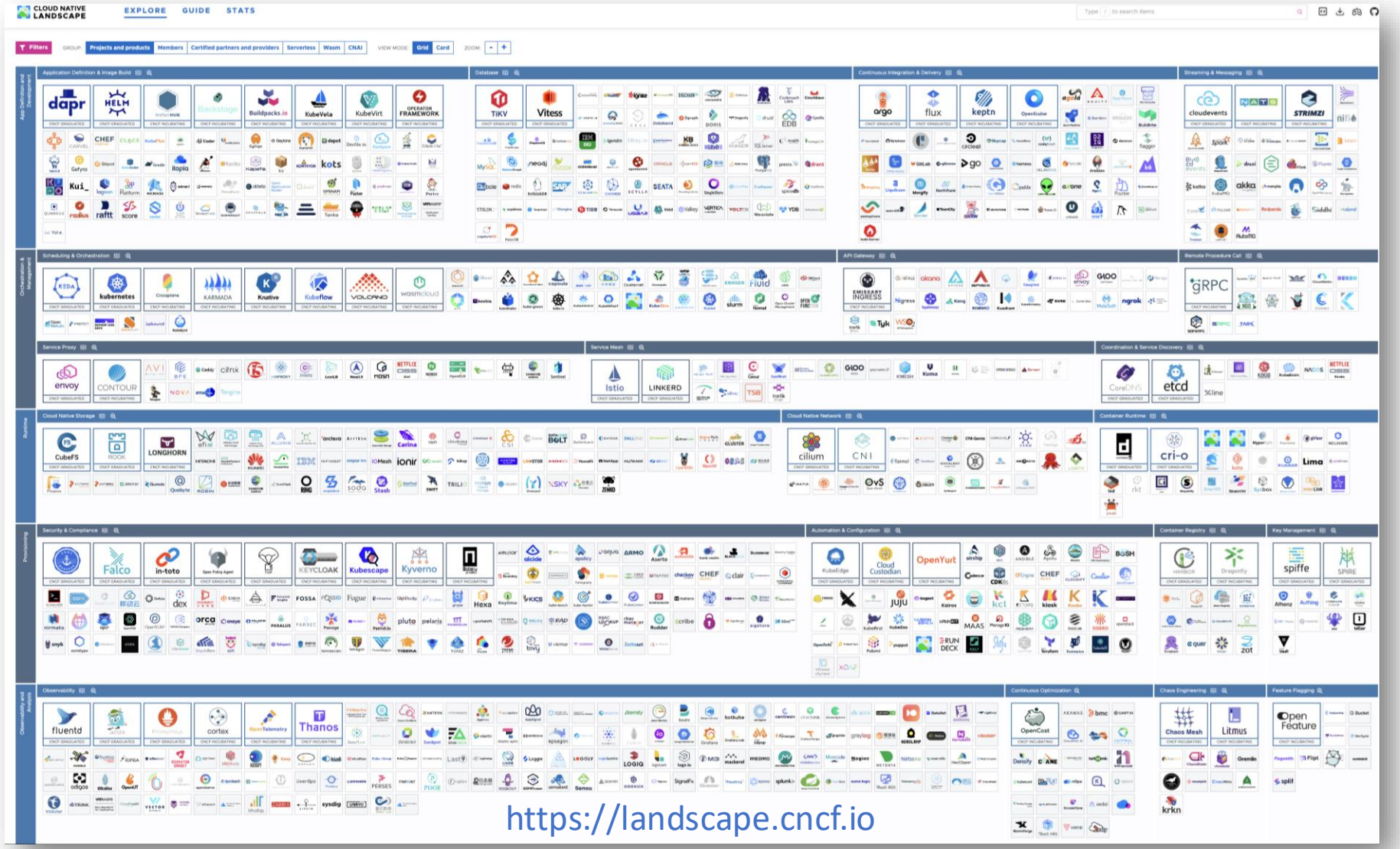
Kubernetes

Pembangunan
Infrastruktur

Apa itu Cloud Native?

- *CNCF is the open source, vendor-neutral hub of cloud native computing, hosting projects like Kubernetes and Prometheus to make cloud native universal and sustainable.*
- *CNCF brings together the world's top developers, end users, and vendors.*
- *CNCF is part of the nonprofit Linux Foundation.*
- *Website : cncf.io*

Cloud native Projects & Products



{ *Cloud Native Landscape* }

https://landscape.cncf.io

{ *Microservices & APIs* }

What is Microservices and API?

- *Microservices are independent processes that interact with other small processes and together shape applications.*
- *API stands for Application Programming Interface, where the keyword is interface. APIs are the doorways, so to speak, that allow developers to interact with an application.*

What is Microservices and API?

- *Microservices are an architectural style for web applications, where the functionality is divided up across small web services.*
- *APIs are the frameworks through which developers can interact with a web application.*
- *Microservices expose it functionalities via APIs*

MICROSERVICES

BEFORE

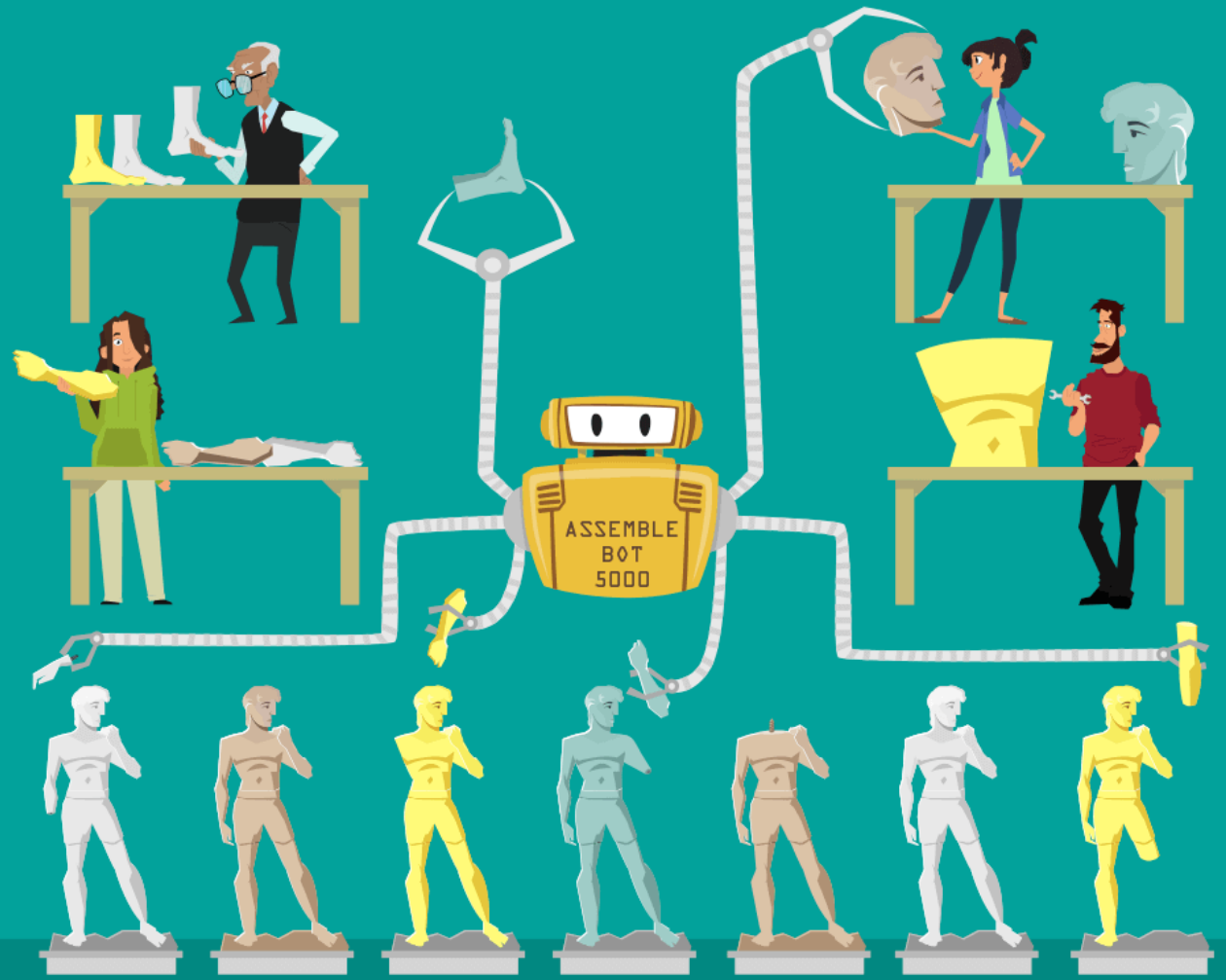
Tightly coupled components

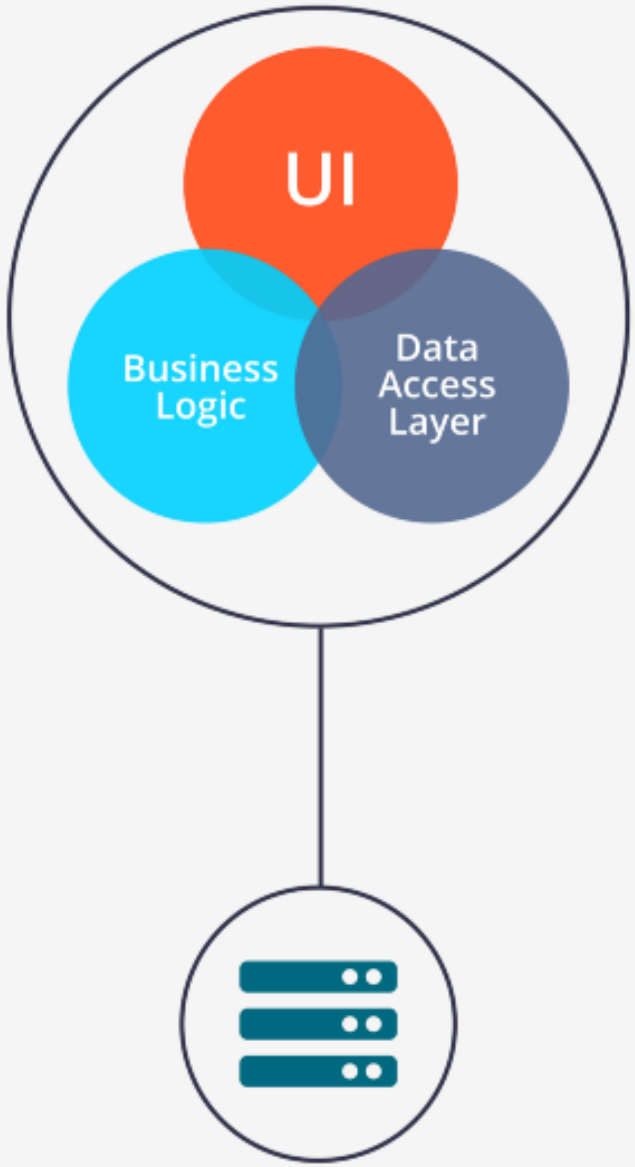
Slow deploy cycles waiting on integrated tests and teams

AFTER

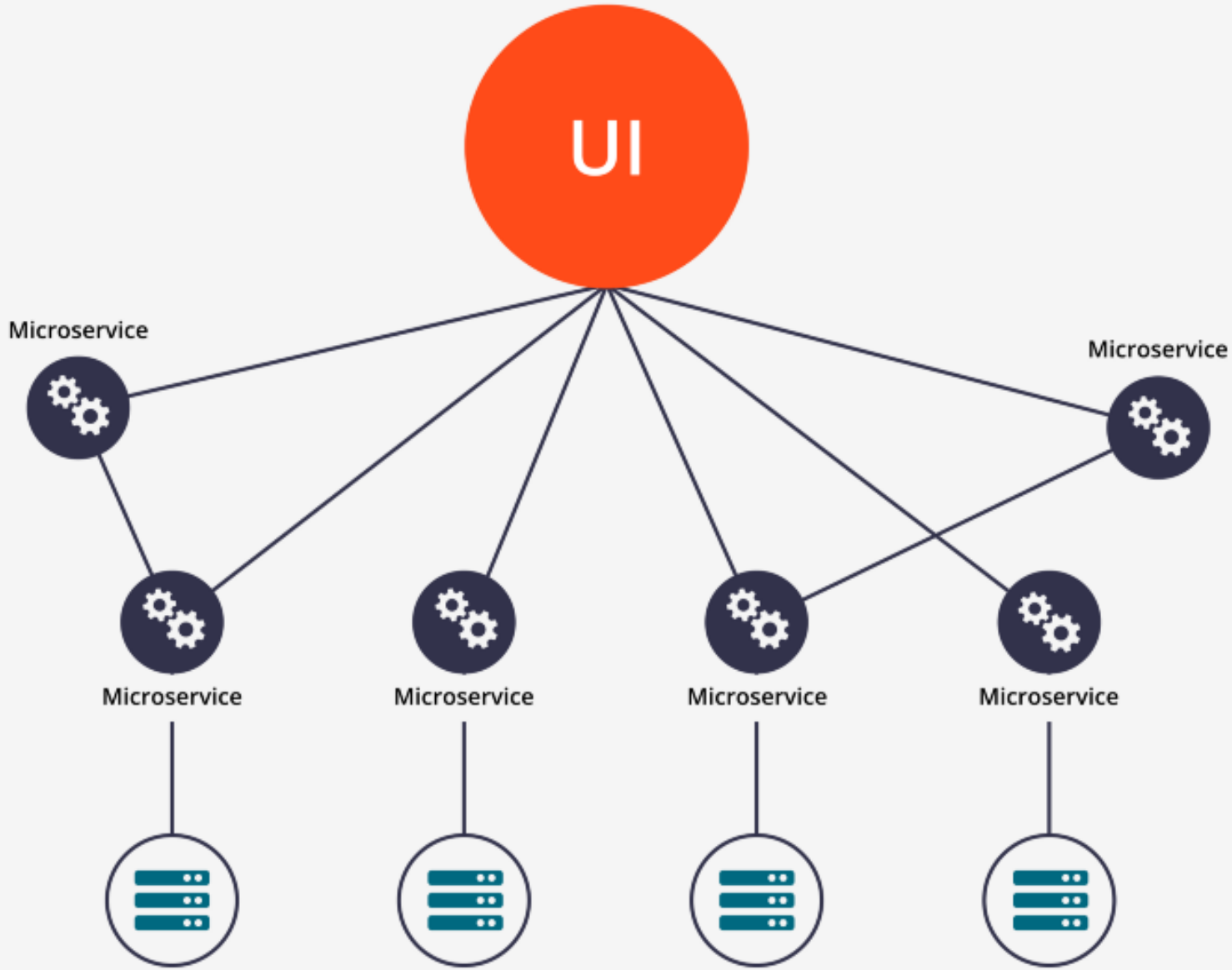
Loosely coupled components

Automated deploy without waiting on individual components

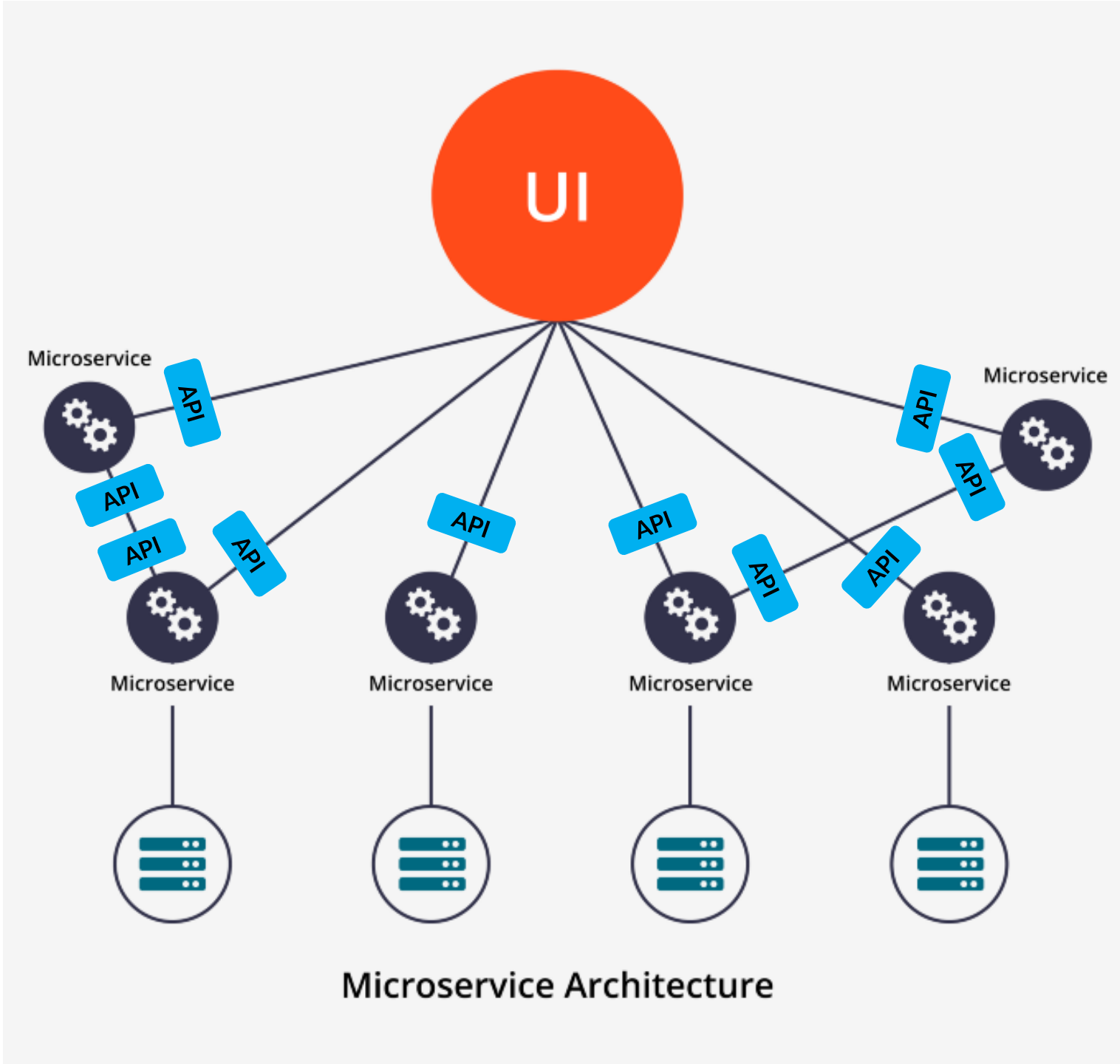




Monolithic Architecture

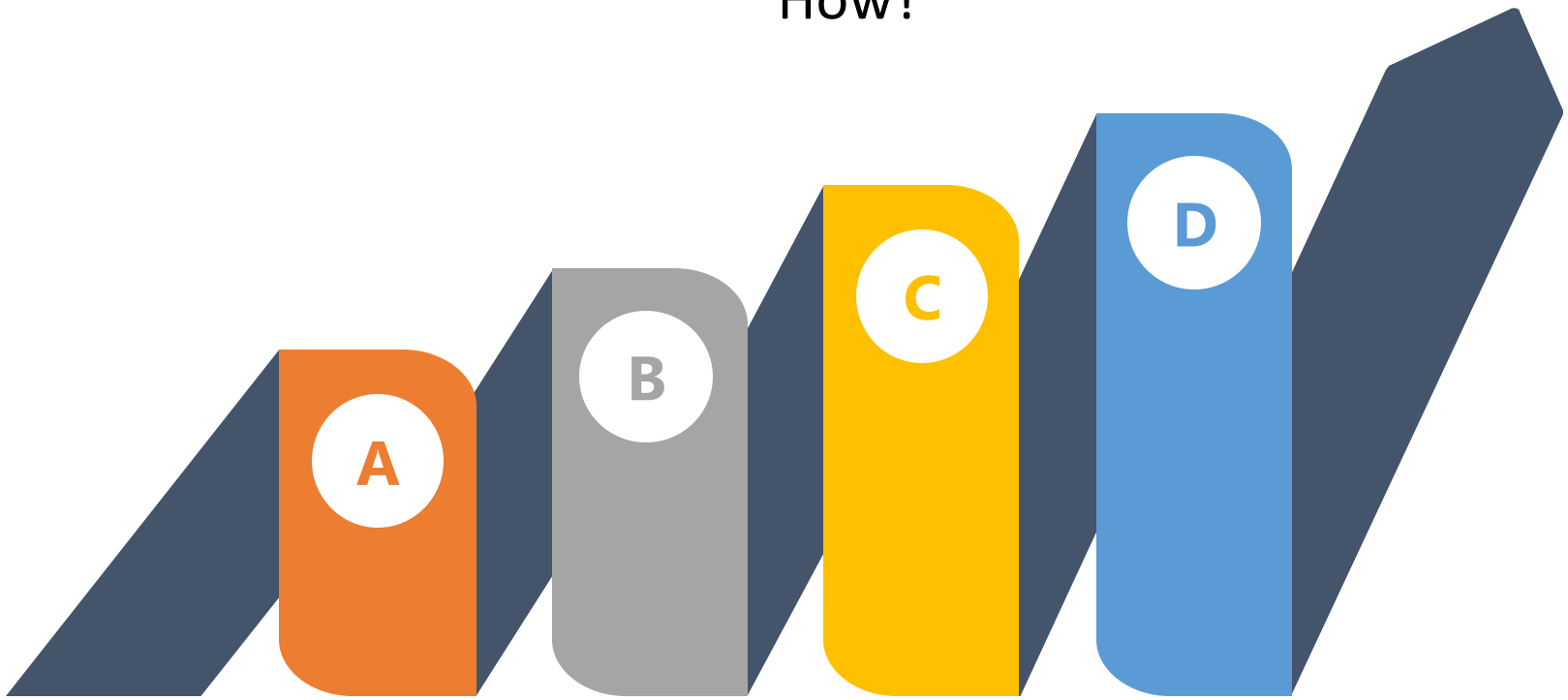


Microservice Architecture


UI
Microservice
API
API
API
API
API
API
API
API
API
Microservice
Microservice
Microservice
Microservice
Microservice
Microservice Architecture

API economy

How?



Expose

Expose organization digital services and assets as APIs

Manage

Manage and control the APIs

Compose

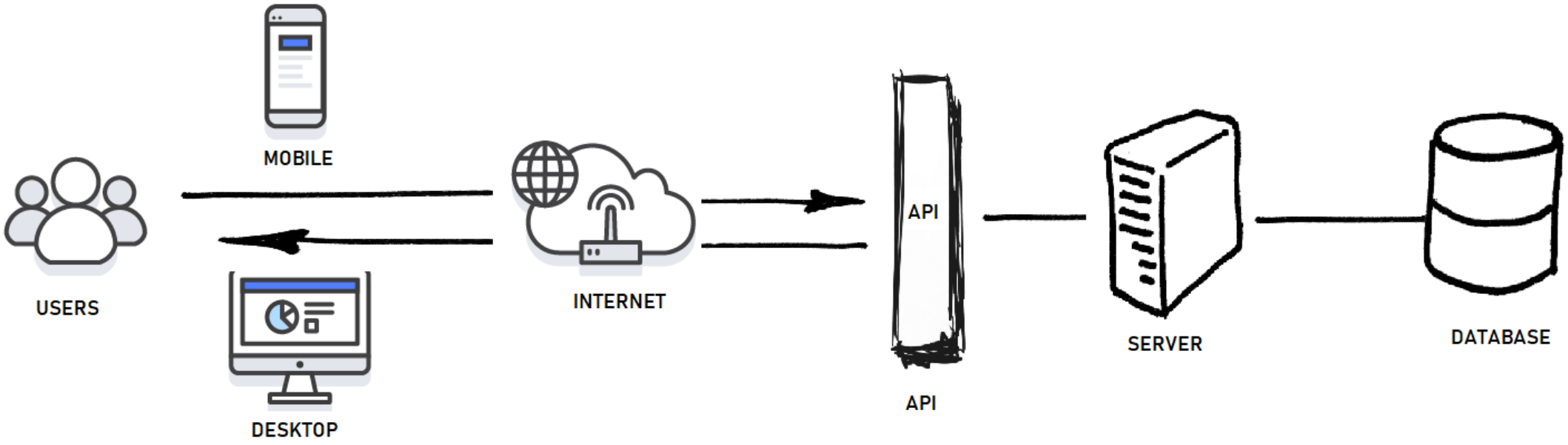
Compose and aggregate those APIs

Consume

Consume by application/mobile

{RESTful API}

- Created in 2000
- an acronym for REpresentational State Transfer.
- REST determines how the API looks like



{RESTful API}

URL	• http://server.com/api
Services	• daerah/johor_bahru
Verbs / Methods	• GET, POST, PUT, DELETE
Representations	• JSON • XML

JSON

- JSON stands for JavaScript Object Notation
- JSON is a lightweight format for storing and transporting data
- JSON is often used when data is sent from a server to a web page
- JSON is "self-describing" and easy to understand

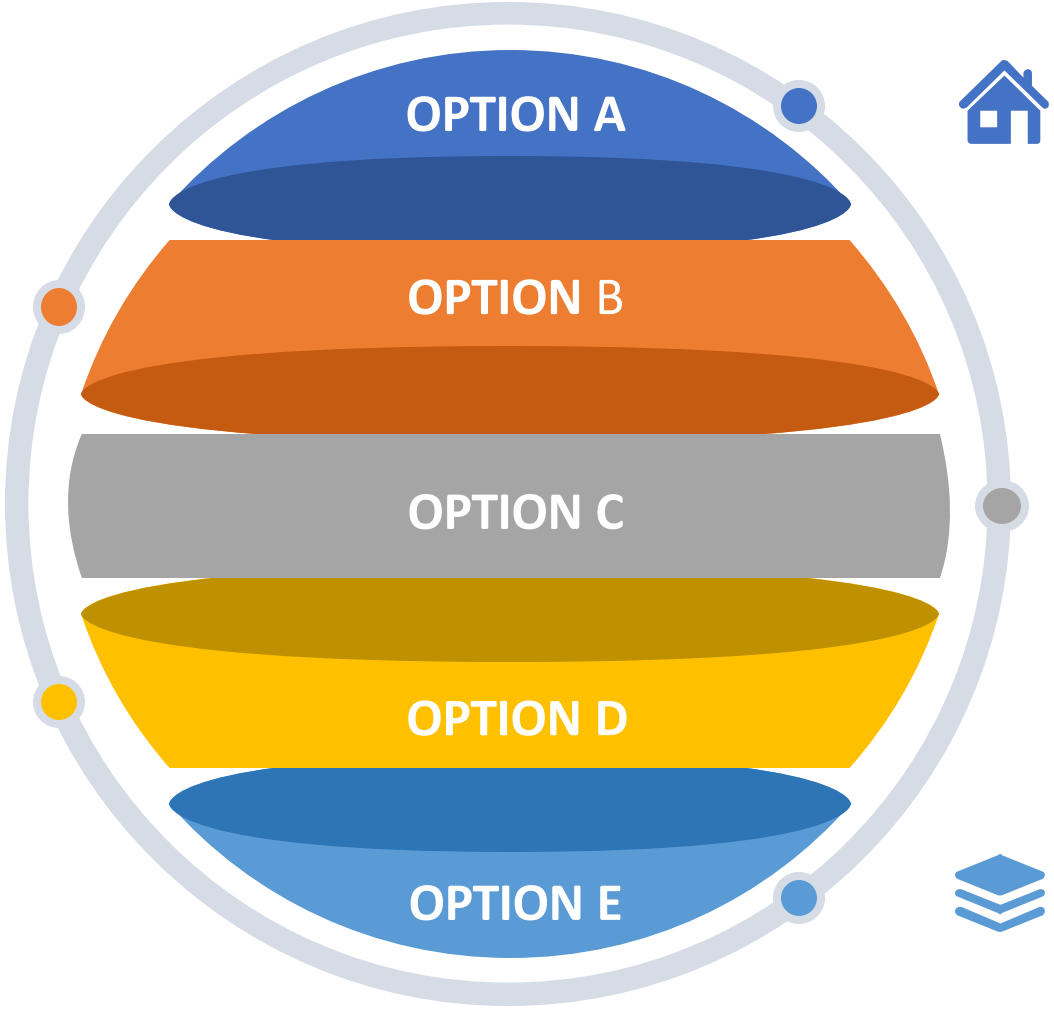
JSON

- JSON Data - A Name and a Value
  - “nama” : “hanafiah”
- JSON Objects - written inside curly braces
  - { “nama” : “hanafiah”, “alamat” : “Kota Tinggi” }
- JSON Arrays – objects inside square brackets
  - ”pengguna” : [

{ “nama” : “hanafiah”, “alamat” : “Kota Tinggi” },
{ “nama” : “hassan”, “alamat” : “Segamat” }
]

How to expose an API?

Type of API calls



Response - Request

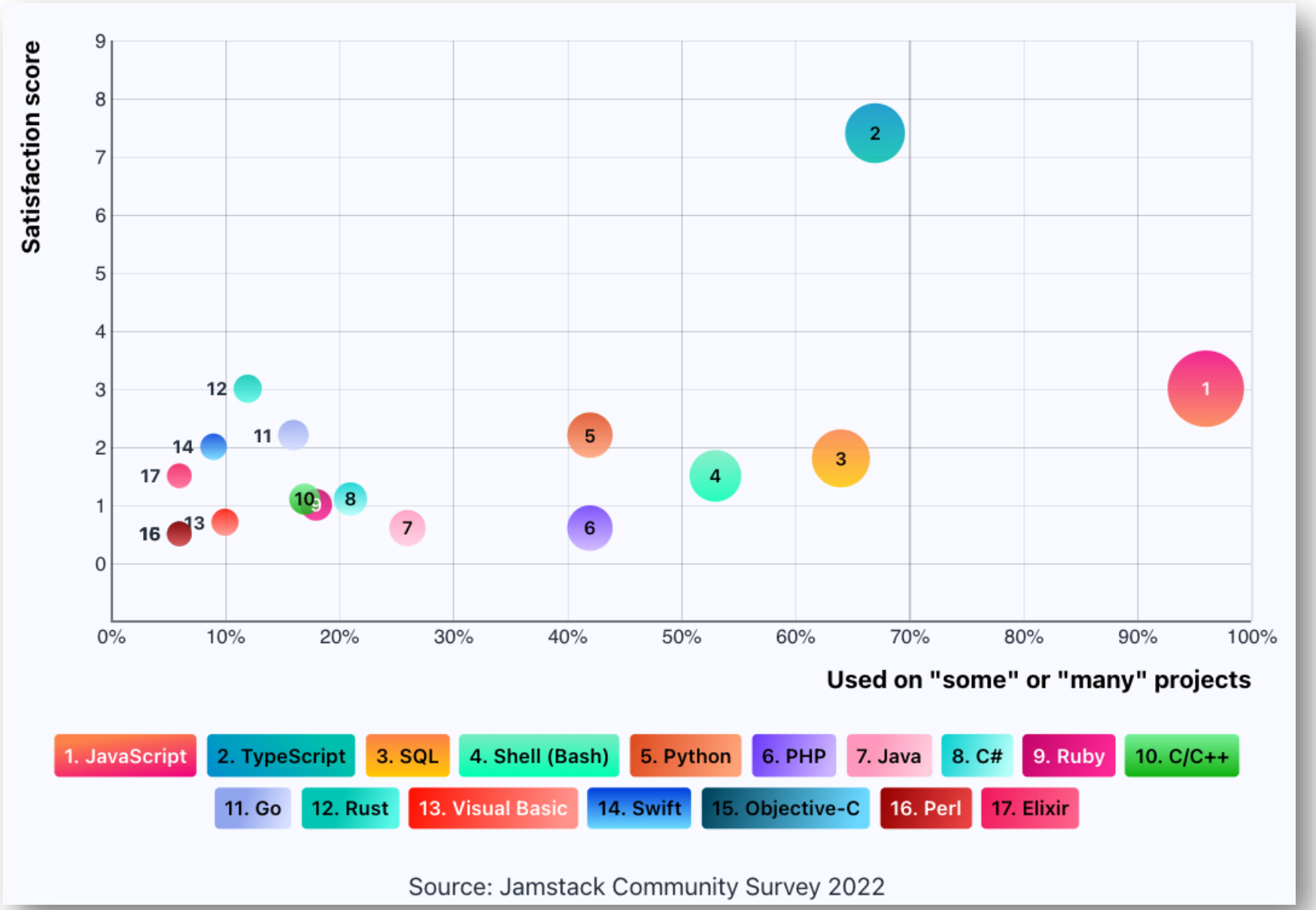
HTTP Streaming

WebSockets

Webhooks

HTTP long-polling

Programming languages by usage and satisfaction survey



- Created in 2009
- Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.
- an asynchronous event-driven JavaScript runtime.
- Node.js uses an event-driven, single-threaded, non-blocking I/O model that makes it lightweight and efficient.
- Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

- **npm** is the world's largest **Software Library** (Registry)
- **npm** is also a software **Package Manager** and **Installer**
- **npm** is now part of **GitHub** (**Microsoft**)

npx

- an npm package runner
- npx is a tool intended to help round out the experience of using packages from the npm registry—the same way npm makes it super easy to install and manage dependencies hosted on the registry, npx makes it easy to use CLI tools and other executables hosted on the registry.

Top NodeJS Frameworks 2024

Fastify

- Fastify was created by **David Mark Clements** and **Matteo Collina** in late 2016.
- The framework was inspired by Express.js and Hapi, aiming to provide a highly performant and low-overhead solution for building web applications and.
- Fastify has gained significant popularity and is used by major companies like Microsoft, Google, Alibaba, AWS, Facebook, and Tencent.

Pengenalan kepada Fastify

- **High Performance:** Fastify is known for its speed, capable of serving up to 30,000 requests per second depending on the complexity of the code.
- **Extensibility:** It is fully extensible via hooks, plugins, and decorators.
- **Schema-Based**: Fastify uses JSON Schema for route validation and output serialization, which enhances performance.
- **Logging**: It integrates with Pino, a fast logger, to minimize the performance cost of logging1.
- **Developer-Friendly**: The framework is built to be expressive and easy to use without sacrificing performance and security1.
- **TypeScript Ready**: Fastify supports TypeScript, making it a great choice for developers who prefer using this language1.

Pengenalan kepada Fastify

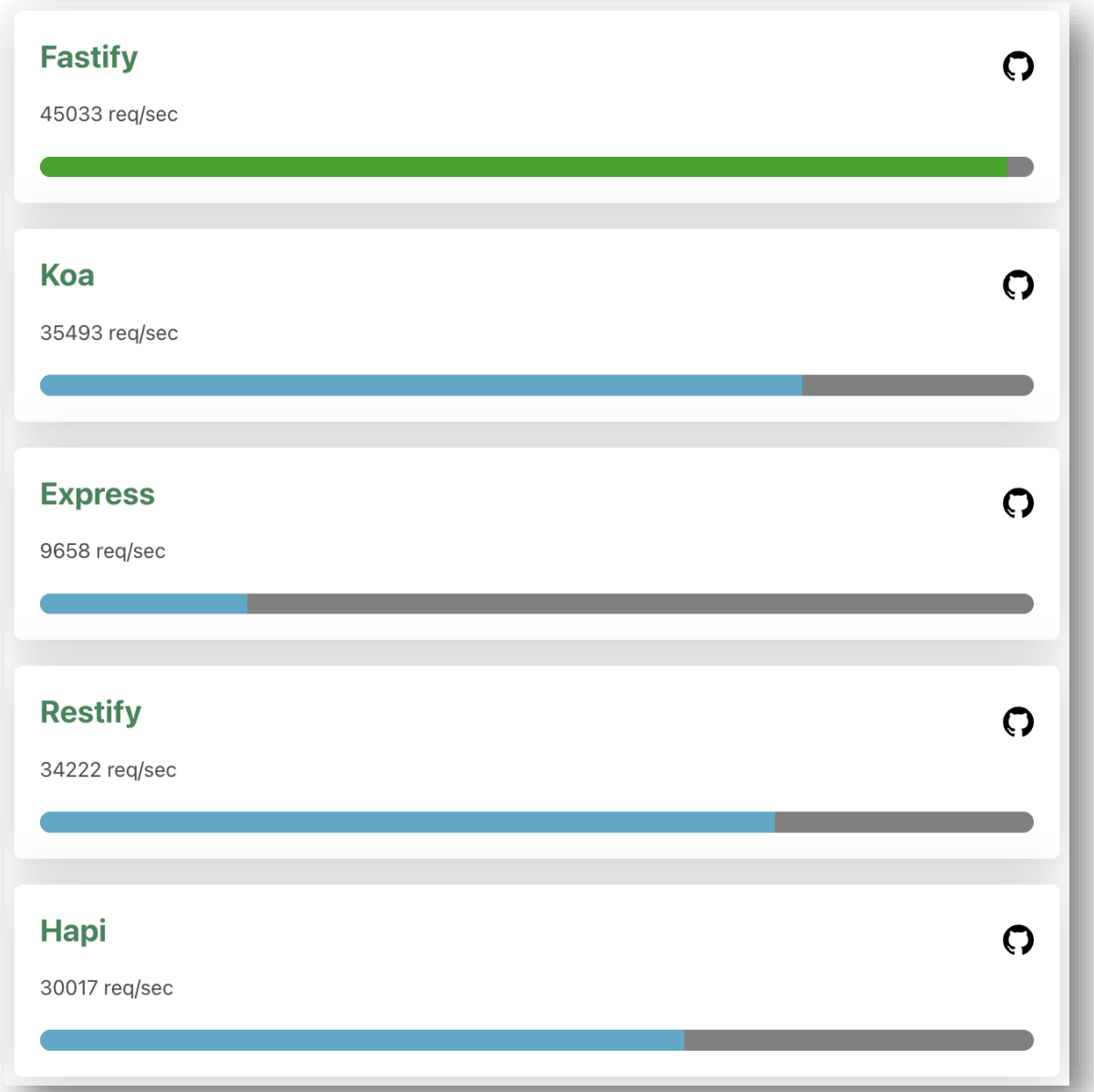
- "**Zero**" overhead in production
- Great for small and big projects
- Easy to migrate to microservices (or even serverless) and back
- Security & data validation
- Secure by default – only listen to 127.0.0.1 and :1. For Docker set to 0.0.0.0

Pengenalan kepada Fastify

- Default to listening only on the localhost 127.0.0.1 interface.
- To listen on all available IPv4 interfaces should be modified to listen on 0.0.0.0
- Specify ::1 to accept only local connections via IPv6. Or specify :: to accept connections on all IPv6 addresses, and, if the operating system supports it, also on all IPv4 addresses.
- To deploy to a Docker (or another type of) container using 0.0.0.0 or :: would be the easiest method for exposing the application.

```
fastify.listen({ port: 3000, host: '0.0.0.0' }, function (err, address) {
  if (err) {
    fastify.log.error(err)
    process.exit(1)
  }
  fastify.log.info(`server listening on ${address}`)
})
```

Benchmarks


Fastify
45033 req/sec
Koa
35493 req/sec
Express
9658 req/sec
Restify
34222 req/sec
Hapi
30017 req/sec

Latihan 1 – Pengenalan Fastify

Fastify Logging

- As Fastify is focused on performance, it uses **pino** as its logger, with the default log level, when enabled, set to **'info'**
- Logging is disabled by default, and you can enable it by passing

  { logger: true } or

  { logger: { level: 'info' } }

https://getpino.io/

https://getpino.io/#/docs/api

```
const fastify = require('fastify')({
        logger: true
})
```

Latihan 2 –Fastify Logging

Environment Variables

- An **environment variable** is a variable whose value is set outside the program, typically through functionality built into the operating system or microservice.
- An environment variable is made up of a name/value pair, and any number may be created and available for reference at a point in time.
- Use cases:
  - Execution mode (e.g., production, development, staging, etc.)
  - Domain names
  - API URL/URI's
  - Public and private authentication keys (only secure in server applications)
  - Group mail addresses, such as those for marketing, support, sales, etc.
  - Service account names

Latihan 3 –Environment Variables
Langkah 1.0

Exit application gracefully

- Exit / shutdown your process, gracefully using **close-with-grace** plugins
- The ability to get out of a crash or other problem situation in nodejs application
- Shutting down application gracefully is required to allow business continuity
- Fastify stops accepting new connections and works to close all outstanding keep-alive connections before exiting the process.

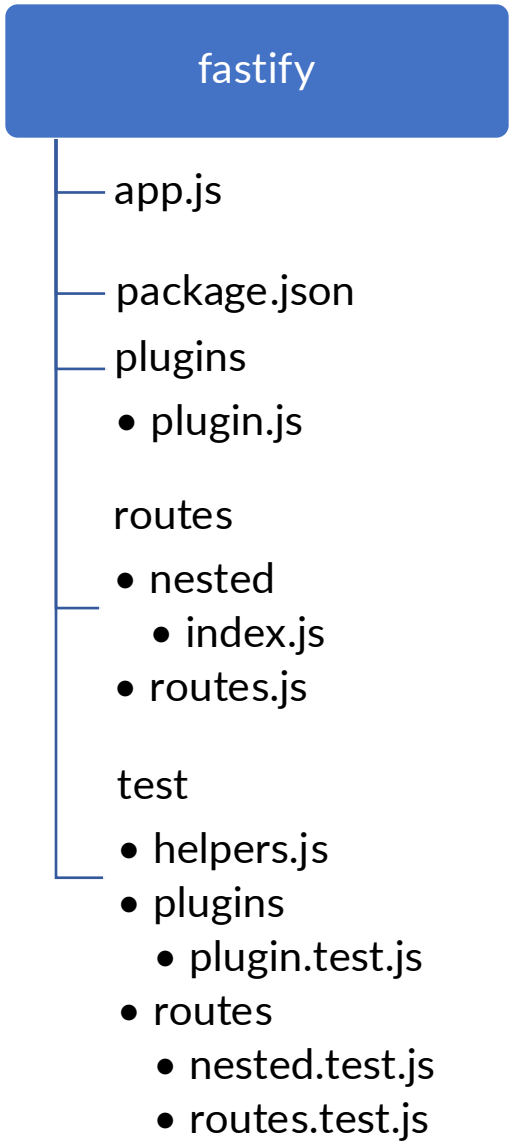
Latihan 3 –Environment Variables Langkah 2.0

Error Handling

- In Node.js, uncaught errors are likely to cause memory leaks, file descriptor leaks, and other major production issues. Domains were a failed attempt to fix this.
- Given that it is not possible to process all uncaught errors sensibly, the best way to deal with them is to crash.
- Catching Errors In Promises: If you are using promises, you should attach a .catch() handler synchronously.
- Fastify follows an all-or-nothing approach and aims to be lean and optimal as much as possible. The developer is responsible for making sure that the errors are handled properly.
- Error Response:
- ```
  {
    error: String // the HTTP error message
    code: String // the Fastify error code
    message: String // the user error message
    statusCode: Number // the HTTP status code
  }
  ```
  ```

Latihan 4 –Error Handling

Fastify File Structure Best Practices



Fastify Routes

- The route methods will configure the endpoints of your application.
- Two ways to declare a route with Fastify:
  - Full declaration
    - fastify.route(options)
  - Shorthand declaration
    - fastify.get(path, [options], handler)

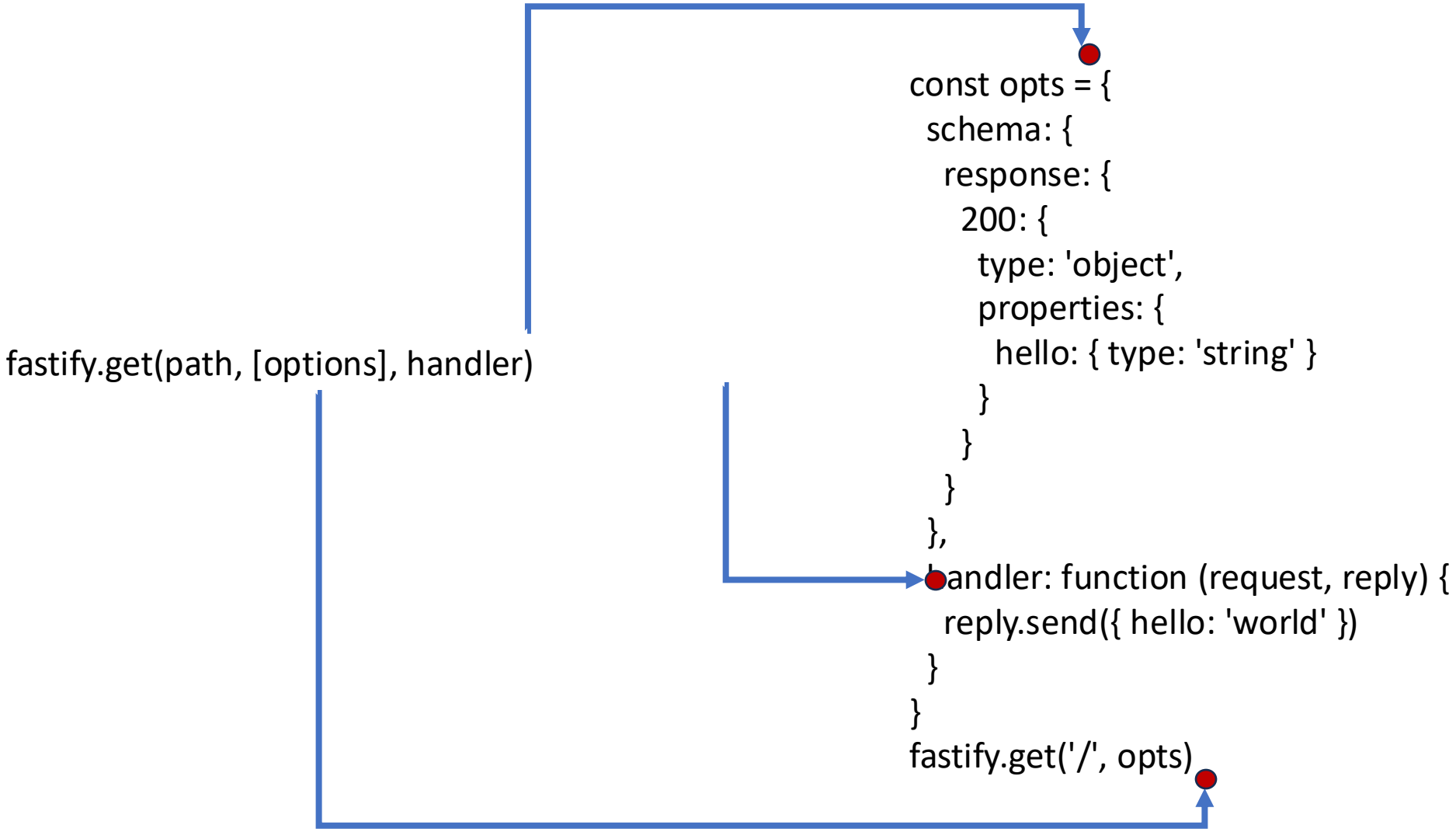
Fastify Routes - Full declaration

- fastify.route(options)
- Options:
  - method: currently it supports 'DELETE', 'GET', 'HEAD', 'PATCH', 'POST', 'PUT', 'OPTIONS', 'SEARCH', 'TRACE', 'PROPFIND', 'PROPPATCH', 'MKCOL', 'COPY', 'MOVE', 'LOCK', 'UNLOCK', 'REPORT' and 'MKCALENDAR'.
  - url: the path of the URL to match this route (alias: path).
  - schema: an object containing the schemas for the request and response. They need to be in JSON Schema format.
    - body: validates the body of the request if it is a POST, PUT, PATCH, TRACE, SEARCH, PROPFIND, PROPPATCH or LOCK method.
    - querystring or query: validates the querystring. This can be a complete JSON Schema object, with the property type of object and properties object of parameters, or simply the values of what would be contained in the properties object as shown below.
    - params: validates the params.
    - response: filter and generate a schema for the response, setting a schema allows us to have 10-20% more throughput.

Fastify Routes - Shorthand declaration

- fastify.get(path, [options], handler)
- fastify.head(path, [options], handler)
- fastify.post(path, [options], handler)
- fastify.put(path, [options], handler)
- fastify.delete(path, [options], handler)
- fastify.options(path, [options], handler)
- fastify.patch(path, [options], handler)

Fastify Routes - Shorthand declaration



Latihan 5 –Fastify Routing

Validation & Serialization

- Fastify uses a schema-based approach, and even if it is not mandatory we recommend using **JSON Schema** to validate your routes and serialize your outputs.
- Internally, Fastify compiles the schema into a highly performant function.
- Validation will only be attempted if the content type is application-json, as described in the documentation for the content type parser.
- The validation and the serialization tasks are processed by two different, and customizable, actors:
  - **Ajv v8** for the validation of a request
  - **fast-json-stringify** for the serialization of a response's body

Validation

- The route validation internally relies upon Ajv v8 which is a high-performance JSON Schema validator.
- The supported validations are:
  - **body**: validates the body of the request if it is a POST, PUT, or PATCH method.
  - **querystring** or **query**: validates the query string.
  - **params**: validates the route params.
  - **headers**: validates the request headers.
- All the validations can be a complete JSON Schema object (with a type property of 'object' and a 'properties' object containing parameters) or a simpler variation in which the type and properties attributes are forgone and the parameters are listed at the top level

Serialization

- Send your data to the clients as JSON, and Fastify using **fast-json-stringify**, to provide an output schema in the route options.
- To use an output schema, as it can drastically increase throughput and help prevent accidental disclosure of sensitive information.

Latihan 6 –Fastify Validation & Serialization

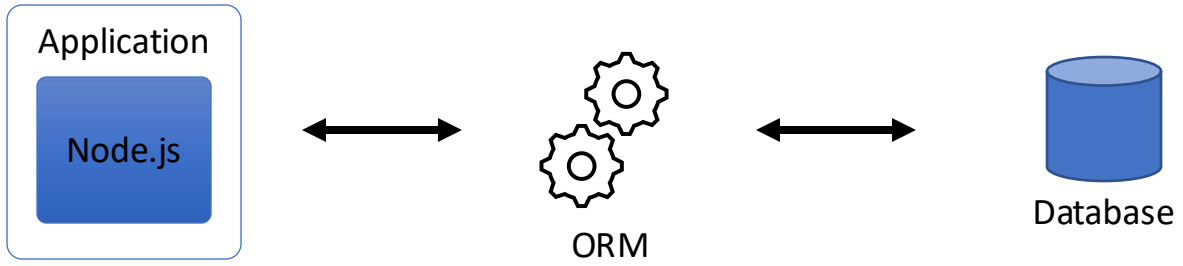
Modern ORM with Prisma

Ringkasan *MySQL Database*

- MySQL is an open source, relational database management system (RDBMS) based on structured query language (SQL).
- Created by a Swedish company called MySQL AB in 1995.
- It stores data in tables, columns and rows.
- Most popular RDBMS.

Apa itu ORM?

- **O**bject **R**elational **M**apping - is a technique that creates a **layer** between the language and the database **without** having to write SQL queries.



SQL: SELECT * FROM users WHERE email = 'test@test.com';

ORM:

```
var orm = require('generic-orm-libary’);
var user = orm("users").where({ email: 'test@test.com' });
```

Kelebihan ORM

- You get to write in the language you are already using.
- You don't have to be an expert in SQL queries, since you don't have to write SQL. Also, a lot of stuff is done automatically, hence Your code becomes short, clean, and easy to maintain
- It abstracts away the database system so that switching from MySQL to PostgreSQL, or whatever flavor you prefer.
- Depending on the ORM you get a lot of advanced features out of the box, such as support for transactions, connection pooling, migrations, seeds, streams, and all sorts of other goodies.
- Can potentially switch databases without any issues, as vendor-specific logic is abstracted in an ORM

NodeJS ORM

- Sequelize
  - supports PostgreSQL, MySQL, MariaDB, SQLite, and MSSQL.
- Knex.js
  - Support PostgreSQL, MSSQL, MySQL, MariaDB, SQLite3, Oracle, and Amazon Redshift
- Bookshelf.js
  - PostgreSQL, MySQL, and SQLite3
- Prisma (next-generation ORM)
  - PostgreSQL, MySQL, and SQLite3
  - MSSQL & MongoDB (preview)

- Eloquent - Laravel ORM
  - Support PostgreSQL, MySQL, SQLite, and MSSQL

Prisma ORM

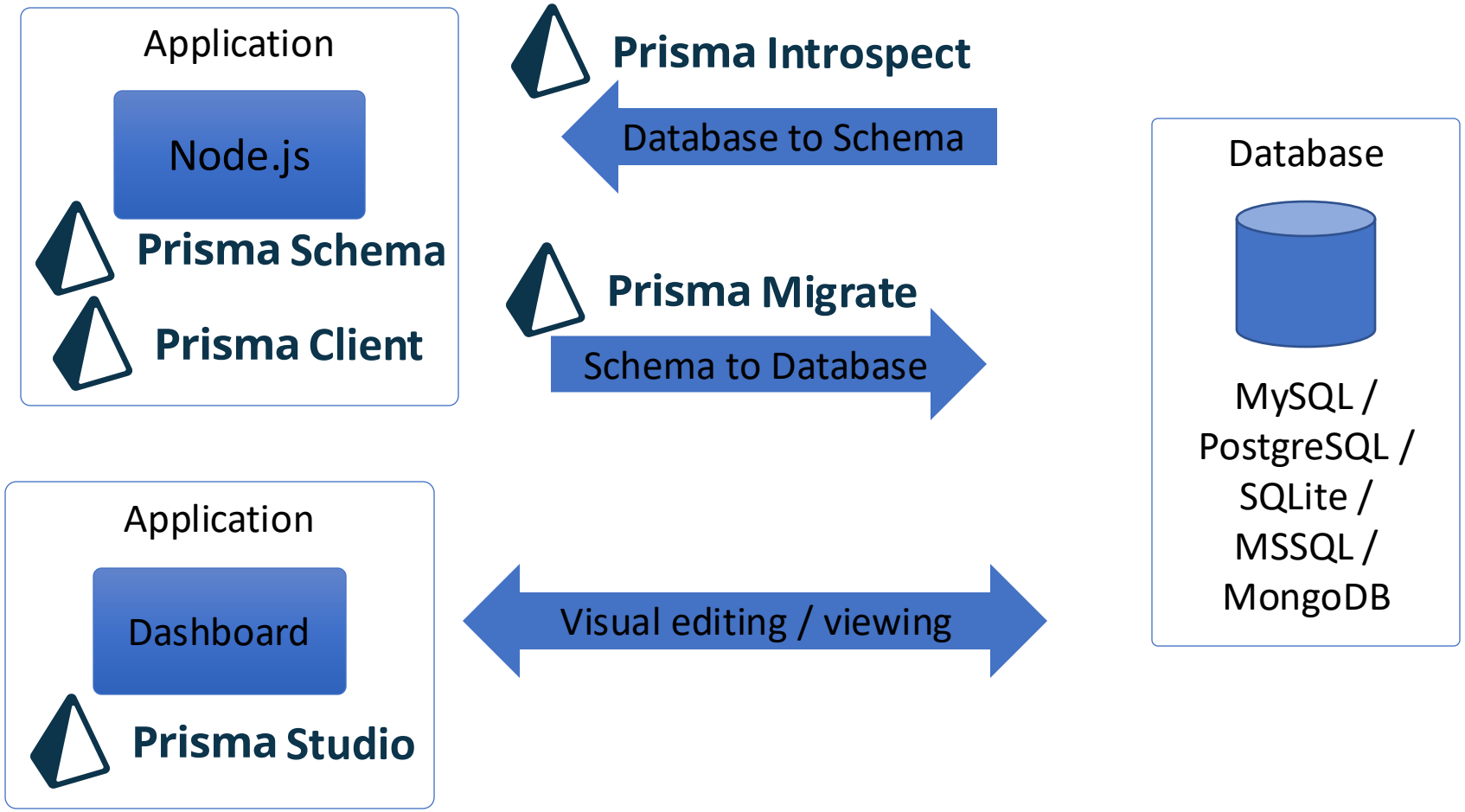
- an open source next-generation ORM toolset for Node.js and TypeScript.
- provides a type-safe API for submitting database queries, returning plain old JavaScript objects
- to build Server-side applications typically are API servers that expose data operations via technologies like REST, GraphQL or gRPC.

Komponen Prisma

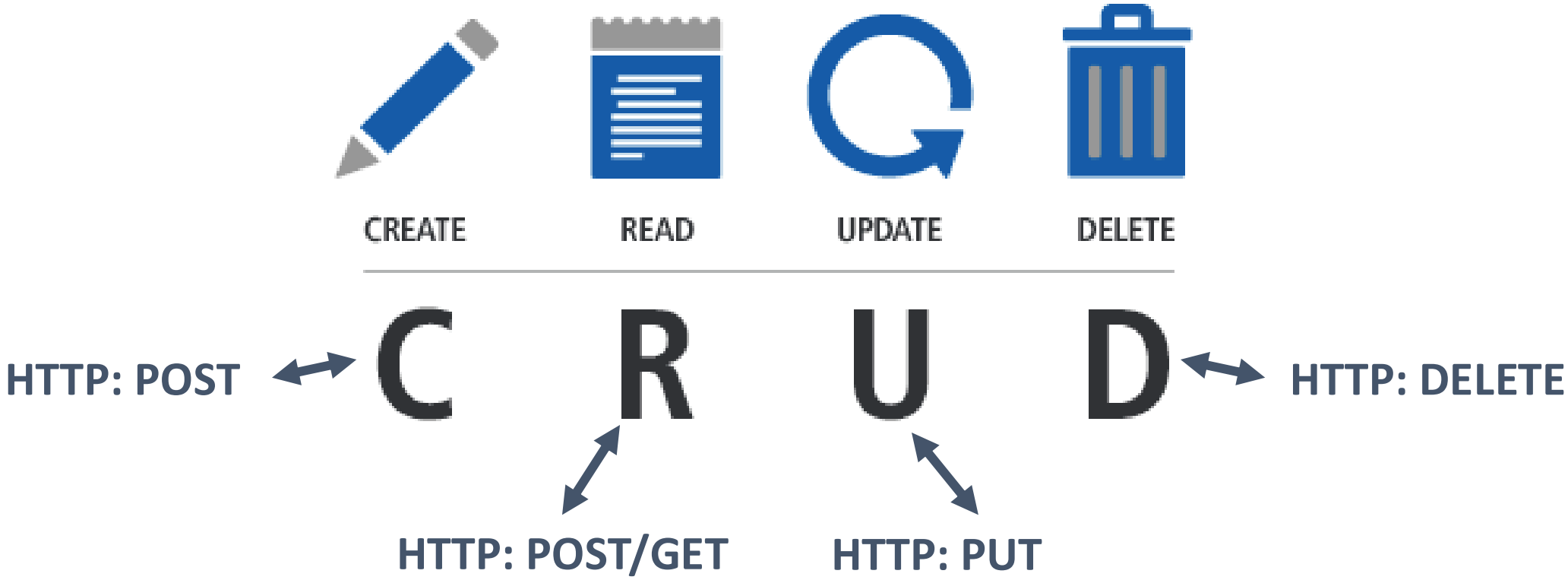
Prisma

- querying (with **Prisma Client**)
- data modeling (in the **Prisma schema**)
- auto-generate schema (with **Prisma Generate**)
- migrations (with **Prisma Migrate**)
- prototyping (via prisma db push)
- seeding (via prisma db seed)
- visual viewing and editing (with **Prisma Studio**)

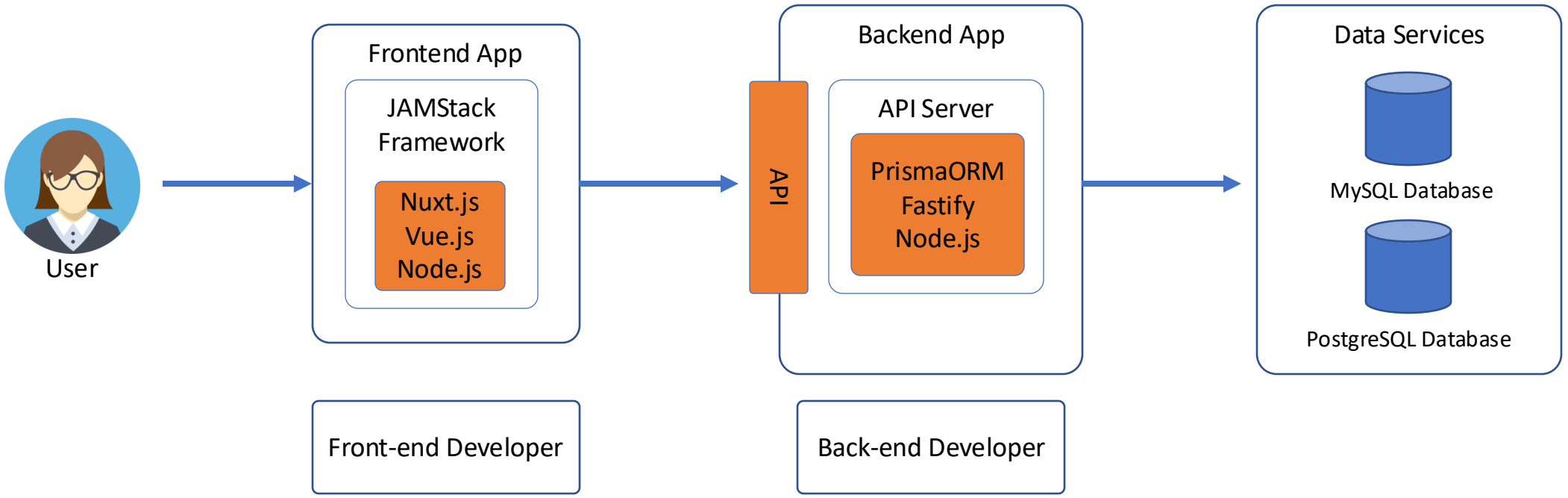
Komponen Prisma



How to expose an API?



Skop Latihan



Latihan 7 –Pembangunan API dari MySQL Database sediada

- Menggunakan MySQL dihoskan di *Public Cloud*
- Nama *database: testdb*

Latihan 8 –Pembangunan API dari MySQL Database yang baru

- Menggunakan MySQL dihoskan di Public Cloud
- Setiap peserta diberikan database: user1, user2….. userN

Thank You