Course: Domain Driven Design & Microservices for Architects

Section: Microservices API & GraphQL

http://acloudfan.com/

Contact: raj@acloudfan.com

Discount Link to course:

https://www.udemy.com/course/domain-driven-design-and-microservices/?referralCode=C5DCD3C4CC0F0298EC1A

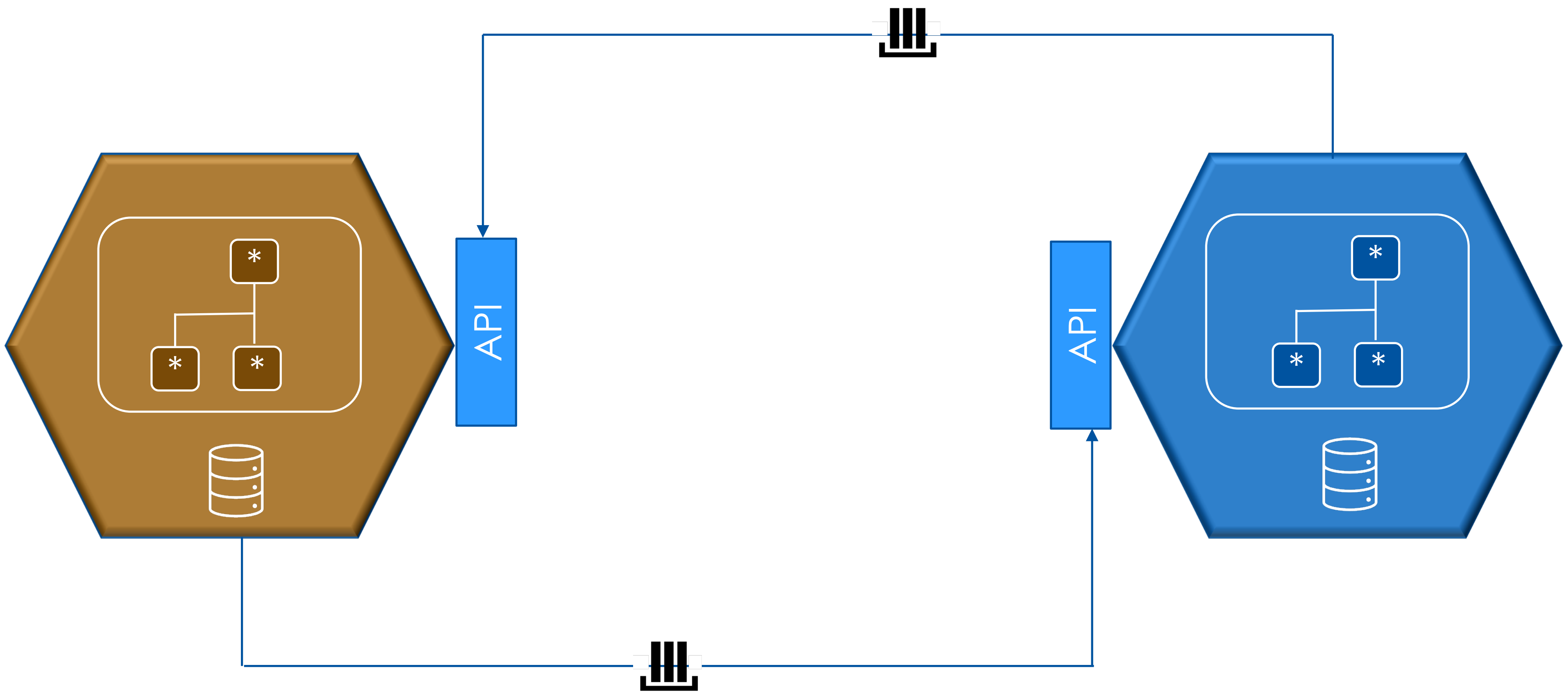
API & Microservices

“ An Application Programming Interface is an interface that defines interactions between multiple software applications or mixed hardware-software intermediaries

- *Wikipedia*

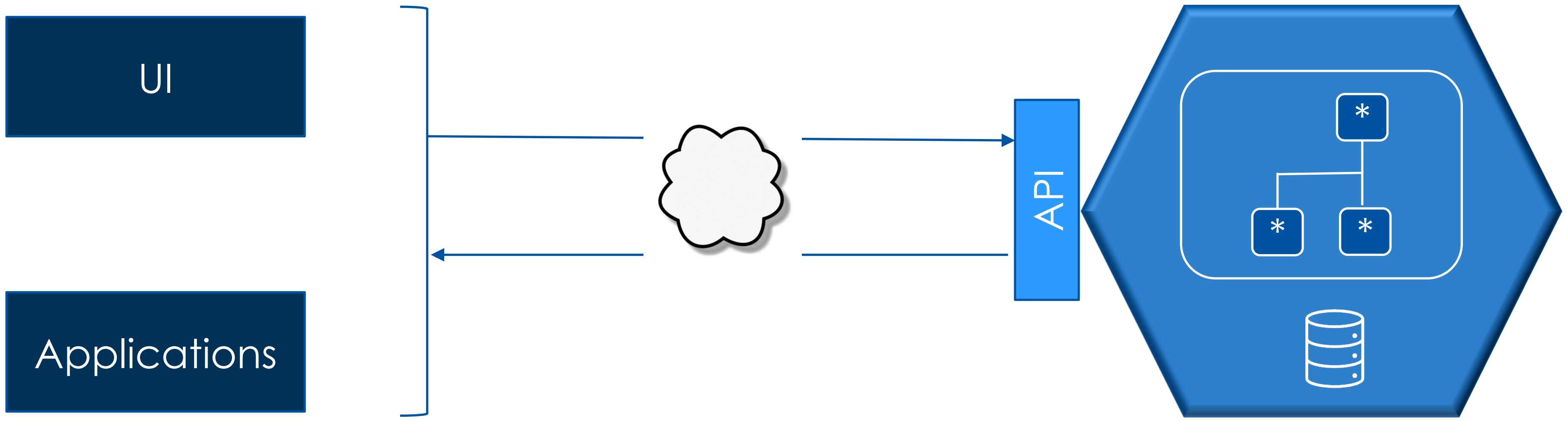
API & Microservices

Microservices interact via API



API & Microservices

External components interact via API



HTTP(s) is commonly used for such API

API & Microservices

Microservices may realize API in multiple ways

RESTful

GraphQL

Contract is defined in different ways | Both have pros & cons

API & Microservices

API Consumer considerations

API Client

- Application performance | complexity
- Impact due to changes in API
- Managing endpoints information

API & Microservices

Section objective

1 REST API

2 GraphQL

3 API Management

4 Service discovery

4 API Gateway pattern

REST over HTTP

REST API Resources

1. Resources
2. API Data Formats
3. Managing the state of Resources

REST API exposes an Endpoint (URI)

Endpoint is used for managing the state of the resources

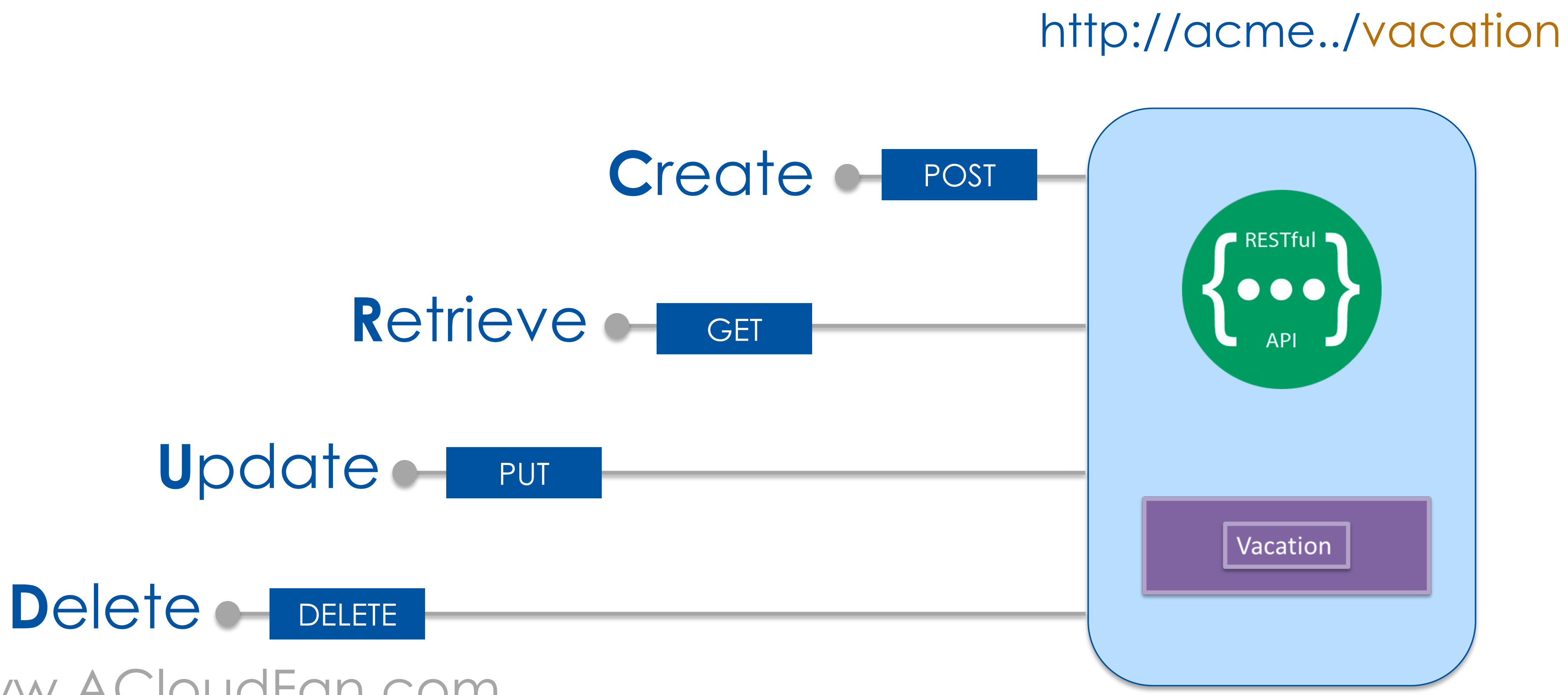
Create

Retrieve

Update

Delete

Use appropriate HTTP Verb for CRUD



API Management

API Common Concerns Management

*

1 API Consumer Types

2 API Management

3 Why use API management?

3 Types of API Consumers

Private
Or
Internal

E.g., other Microservices

Public
Or
External

E.g., Independent Travel bloggers

Partner

E.g., Resellers | Affiliates

3 Types of API Consumers

NO difference from implementation perspectives

Difference is in how these API are managed e.g., Security, Capabilities

3 Types of API Consumers

Allowed Invoke API **5,000 times per second**

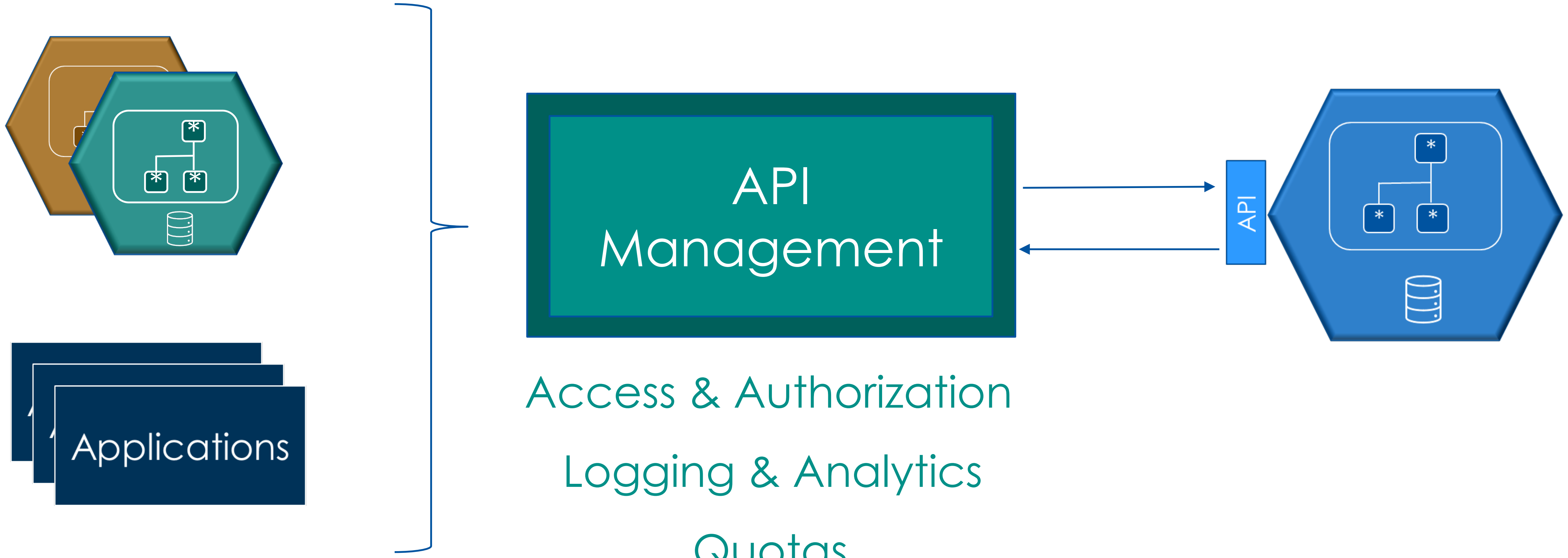
Allowed Invoke API **5 times per second**

Access to **ALL features**

Access to **ONLY certain GET calls**

API Management platforms

Used for addressing common API concerns



Access & Authorization

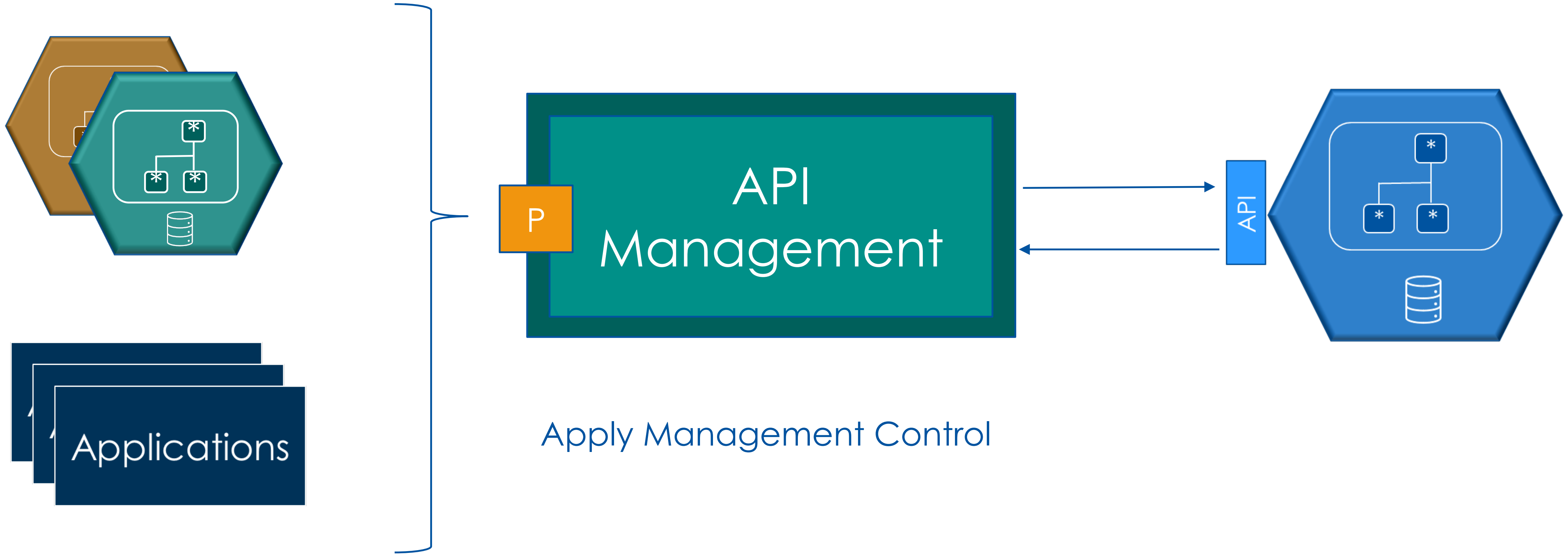
Logging & Analytics

Quotas

API Documentation

API Management platforms

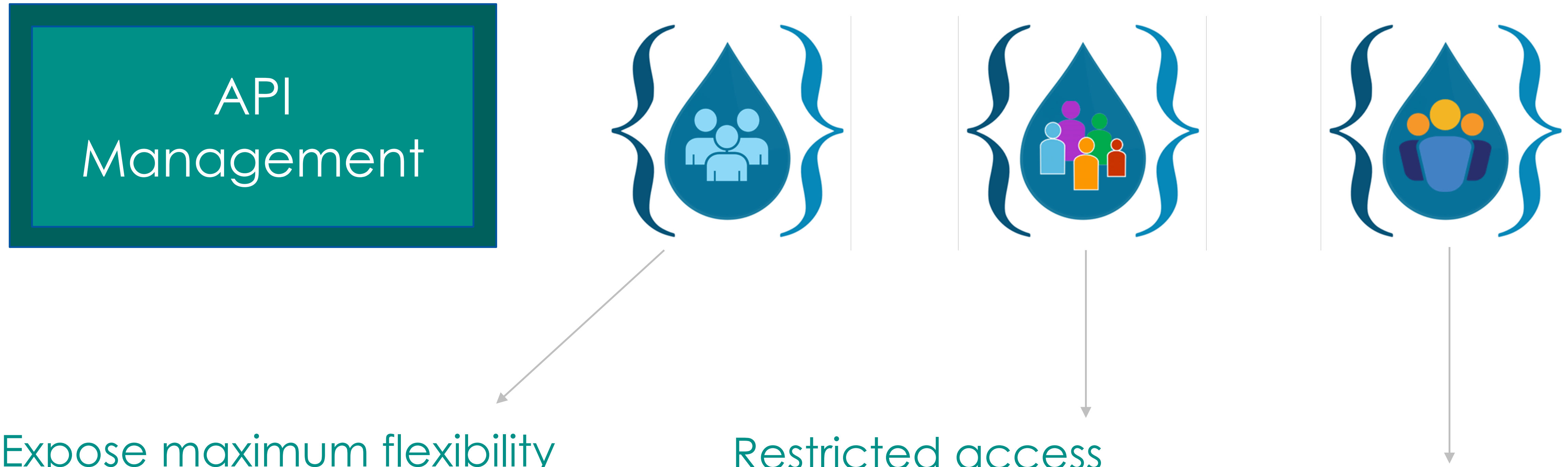
Used for addressing common API concerns



Apply Management Control

API Management platforms

Offers declarative | policy based management features



Expose maximum flexibility

Maximum calls/second

Restricted access

Limit the call/second

Defined SLA

API Management platforms

Offers declarative | policy based management features

API Management

JSON is commonly used for policies

How policies are defined - depends on the API management product

API Management

apigee

MuleSoft®

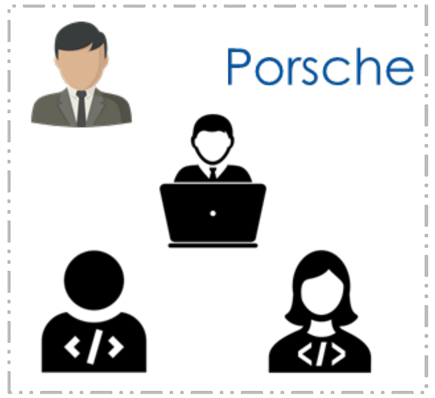
WSO2

Amazon
API Gateway

Azure API Management
Microsoft Azure

API Management Benefits

Offload the common Concerns to API Management



Focus is on Domain | Business Logic

Microservices code is cleaner

Change management is easier

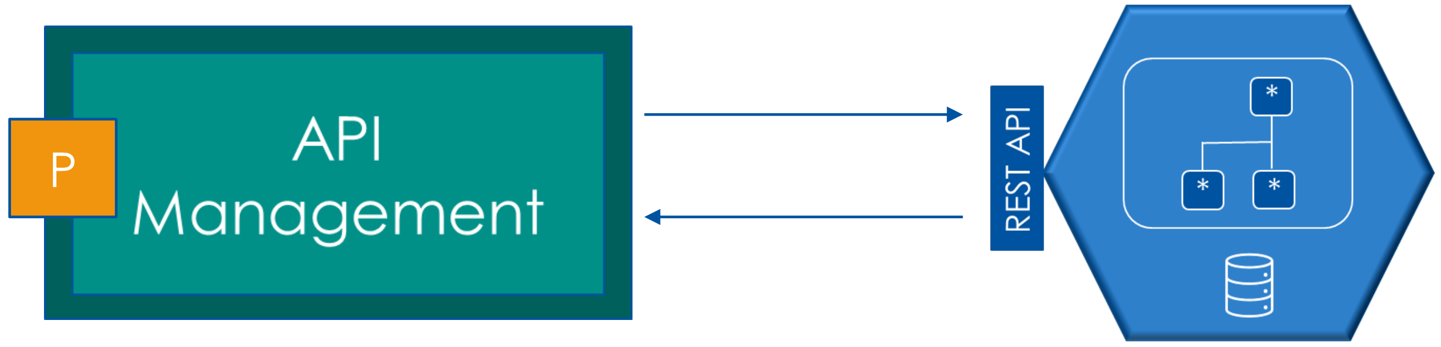
Quick Review

Private API
Internal Consumers

Public API
Internal Consumers

Partner API
Internal Consumers

Microservices Implements ONLY the business logic



Address the common concerns
realized by way of *declarative policies*

ACME Products API

ACME partnership with 3rd parties

ACME Travel

1. ACME Sales Channel strategy
2. Need for an API
3. API Consumer pattern

Paul, Product Manager

IT Lead

I am responsible for the product design and provider relationships. These products are what customers buy from Acme.

Based on the market research I pick up the parts of the product, sometime we refer to these products as bundle. There are certain markup guidelines that I need to follow in order to make the product profitable. Also I need to take into account the seasonality.

Correct pricing of the bundle requires careful negotiations with the providers. Some providers such as airlines & Hotels offer us bulk prices which are below the Market Prices. Some providers prefer to work with us on commissions. We sign contracts with providers that lists the commission structure as well as any penalties and the terms.

Product a.k.a. Bundle

Provider

Customer

Markup

Seasonality

Bulk Prices

Market Price

Commissions

We would like to expand our sales channels

Expand partner network by providing easy access to ACME products

Anyone on the web should be able to sell our products bundles and make $$ in the form of commissions !!

IT Lead

Build a REST API for external consumers

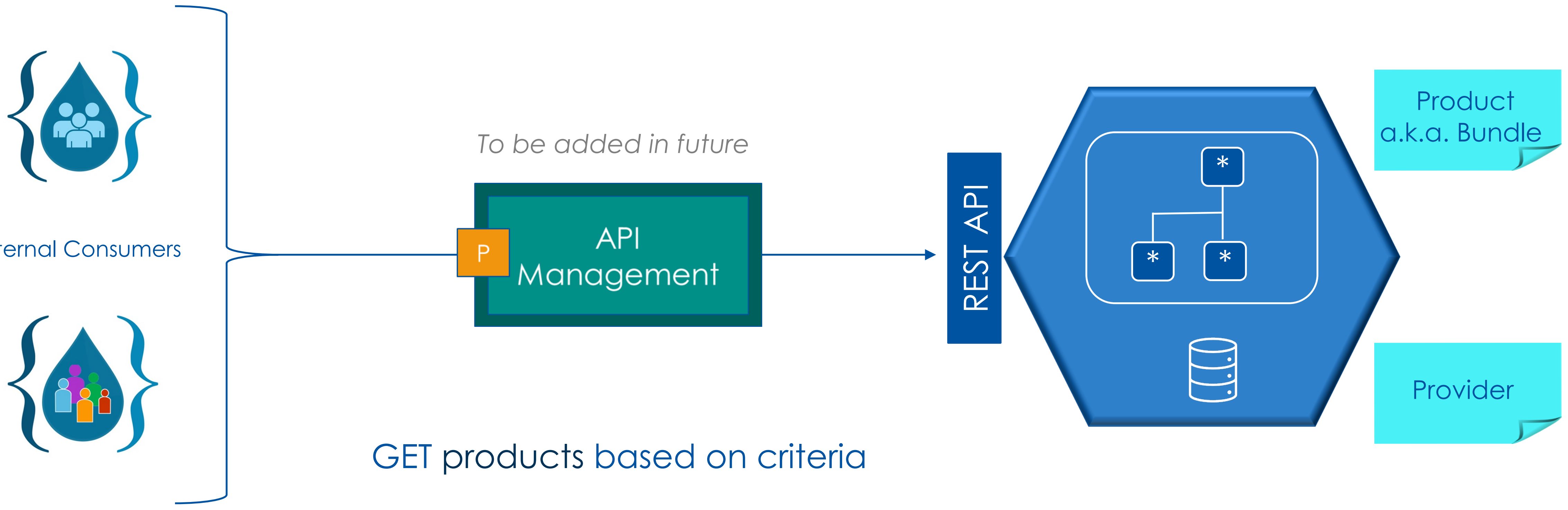
Messaging is not an option so we will use REST/HTTP API

This API will be open for use by any Public Developers (Blogger)

API management will be used for security, quota setup etc.

Scope of Products API (*version 1*)

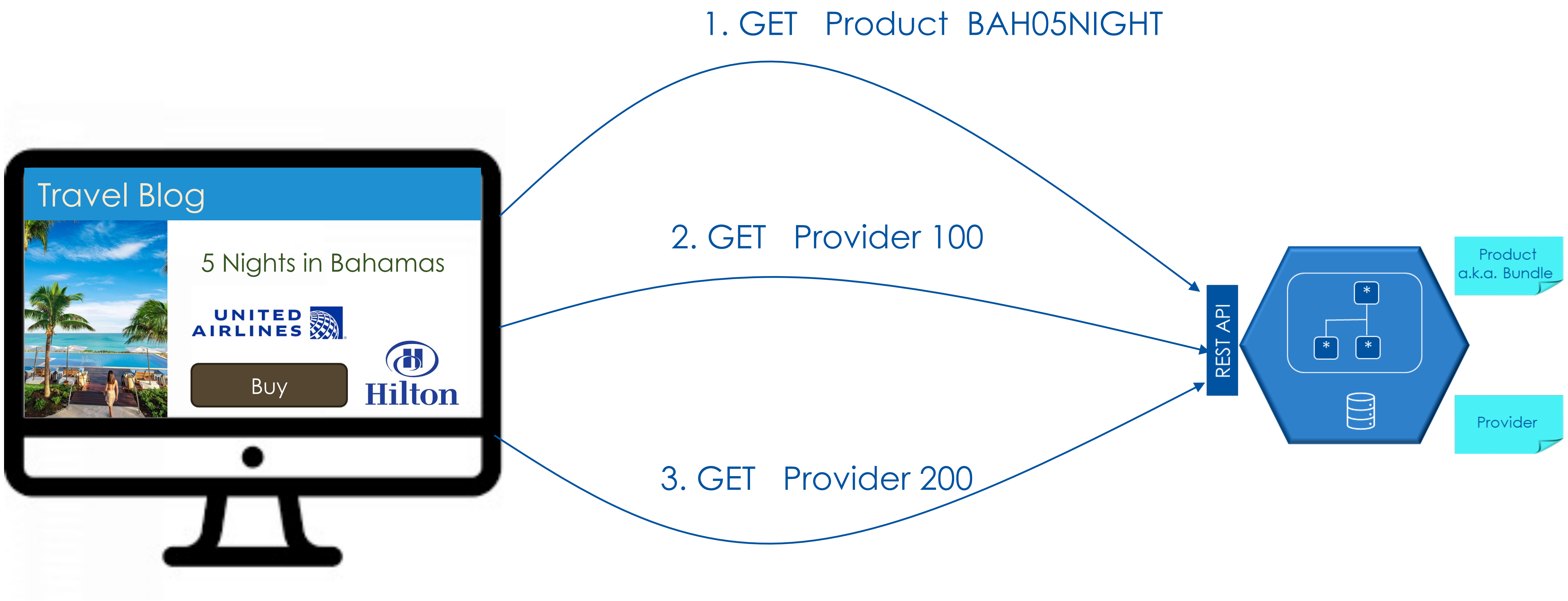
Consumers should be able to GET product Info



GET products based on criteria

GET provider based on criteria

REST API Consumer Pattern



REST API Implementation

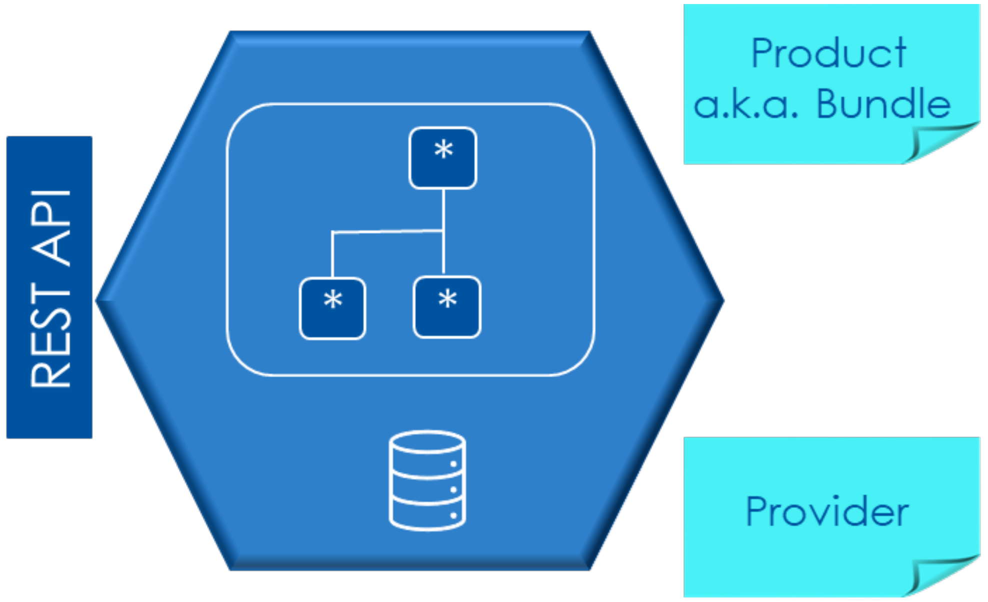
node JS

express

Flask

web development, one drop at a time

spring boot



ACME Product API

A REST API providing access to vacation packages & providers

1. Class diagram - products model
2. REST API in Action
3. Code Walkthrough

Intent is to demonstrate a working REST API

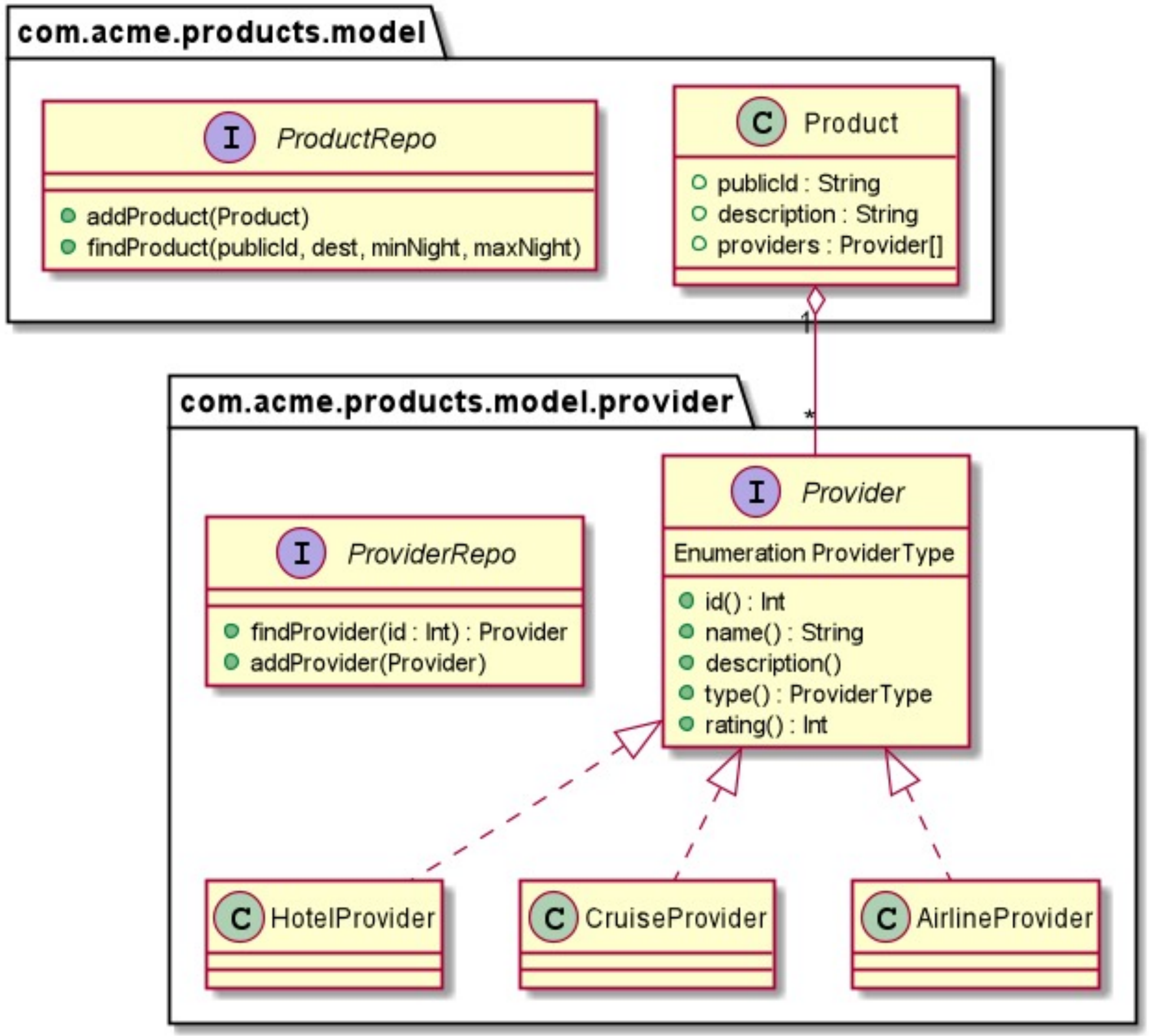
Repository

https://github.com/acloudfan/MSFA-ACME-**Products**-v1.0.git

Branch: api

Products domain model (draft version)

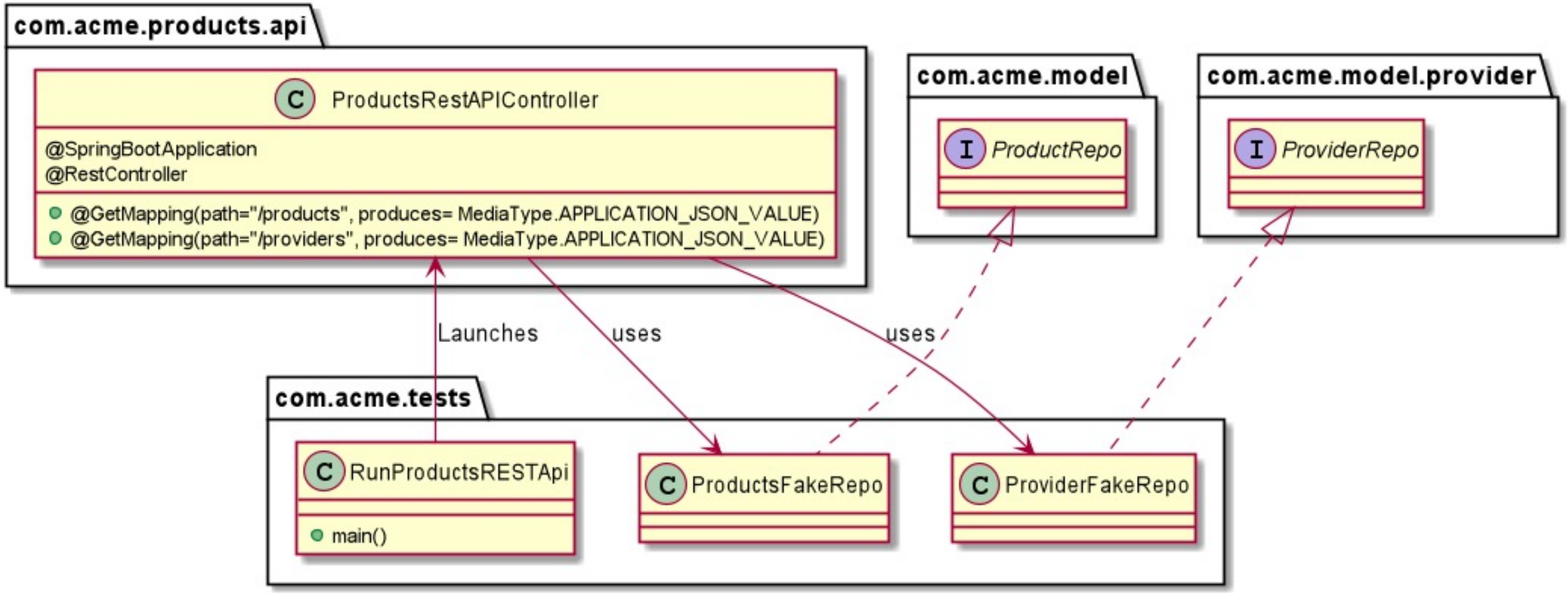
uml/products.model.class.puml



www.ACloudFan.com

Products REST API Controller

uml/restapi.class.puml



Products REST API Access

Product
a.k.a. Bundle

http://localhost:8080/products ?publicId=### &dest=### &minNight=### &maxNight=###

Provider

http://localhost:8080/providers ?id=###

Introduction to GraphQL

Addressing the challenges with REST API

*

1. What is GraphQL? Issues it addresses?
2. GraphQL Server Flow
3. REST Vs. GraphQL

GraphQL

“ A Query Language for API that is NOT tied to any specific database or technology or network protocol

It's a specification for Query Language - https://spec.graphql.org/

HISTORY

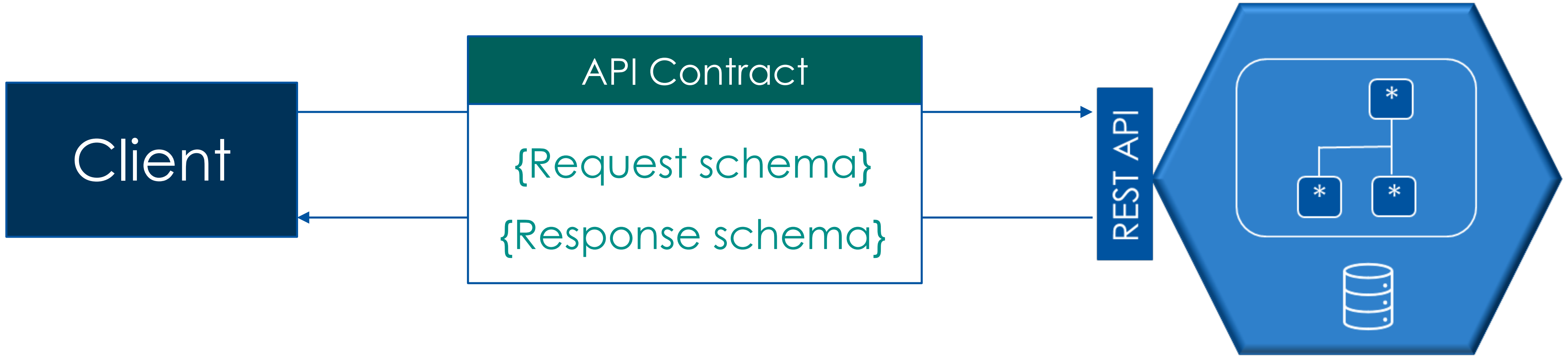
Facebook used it initially for their mobile app in 2012

Open sourced it in **2015**

https://graphql.org/users/

REST API Contract

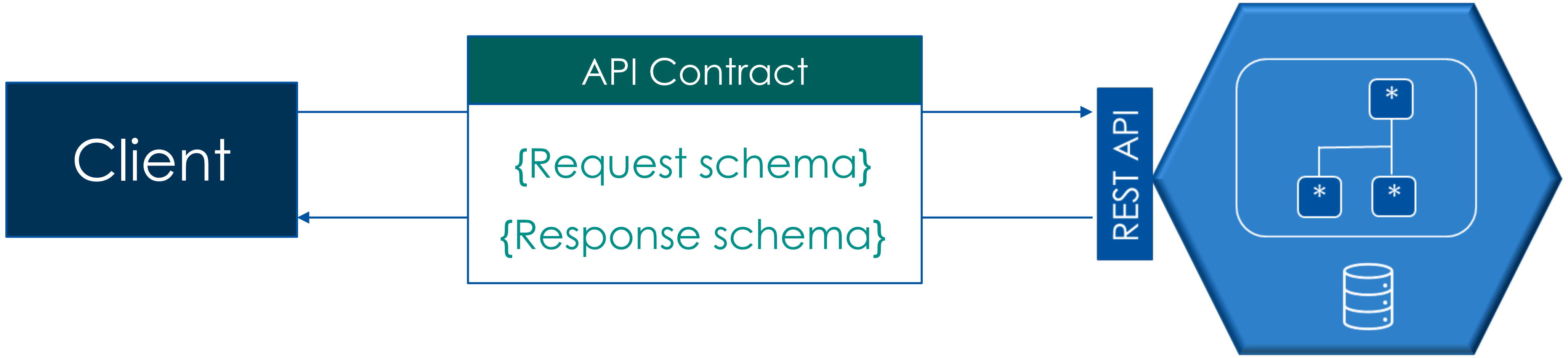
API response is fixed, and client has no control over it



- Structure of response is fixed
- Client receives all of the data whether it needs or not

REST API Contract

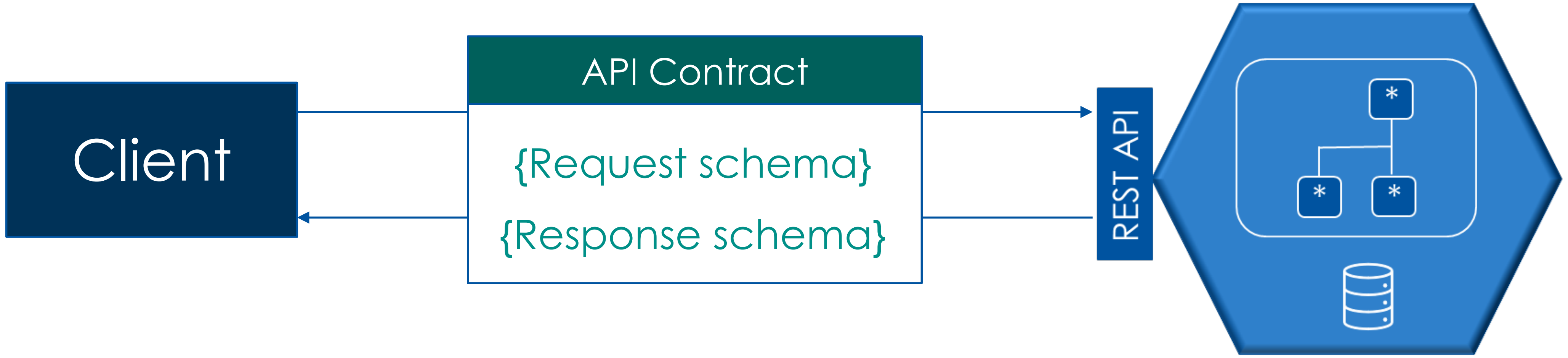
API response is fixed, and client has no control over it



- Its like executing a "SELECT * FROM TABLE …."

REST API Contract

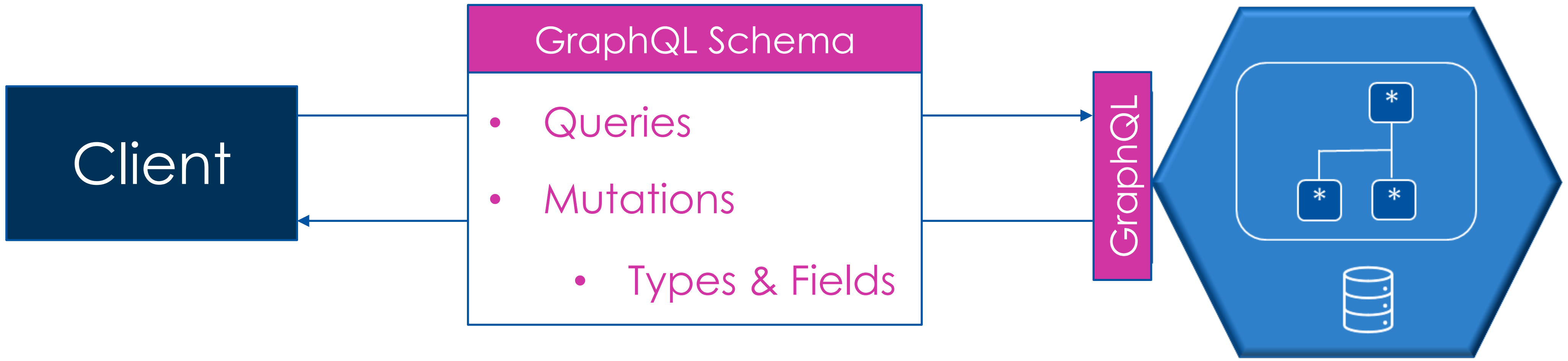
API response is fixed, and client has no control over it



Referred to as Over-Fetching issue

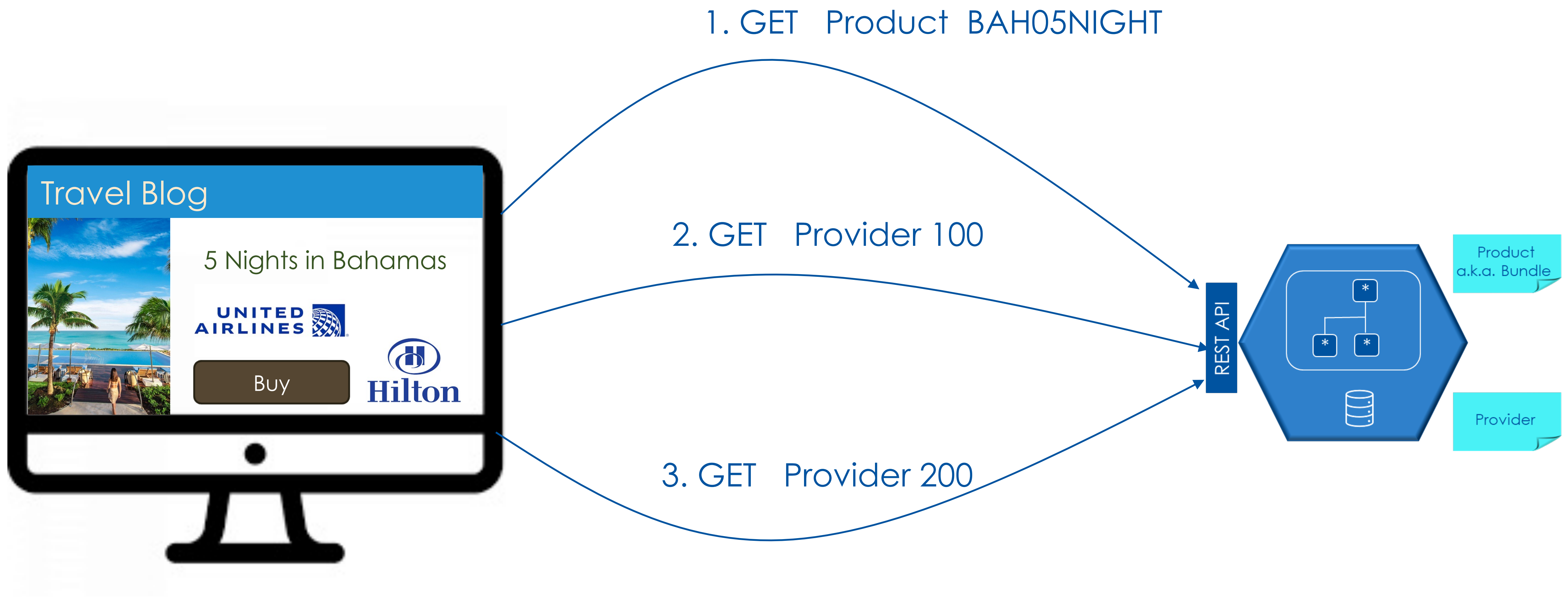
GraphQL Contract

Client controls the response content



- Client tells server what it needs
- Its like executing a "SELECT name, ssn, phone FROM TABLE ….WHERE …"

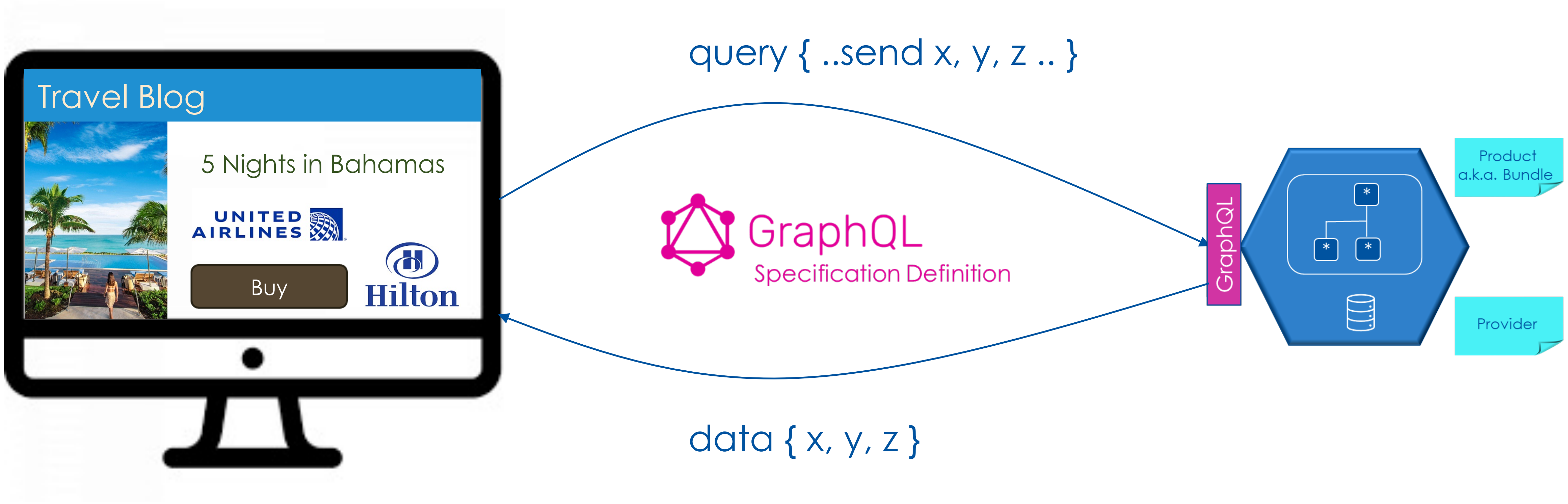
REST API Granularity



Under-Fetching issue => Performance hit

GraphQL API

REST API contract replaced with a GraphQL schema

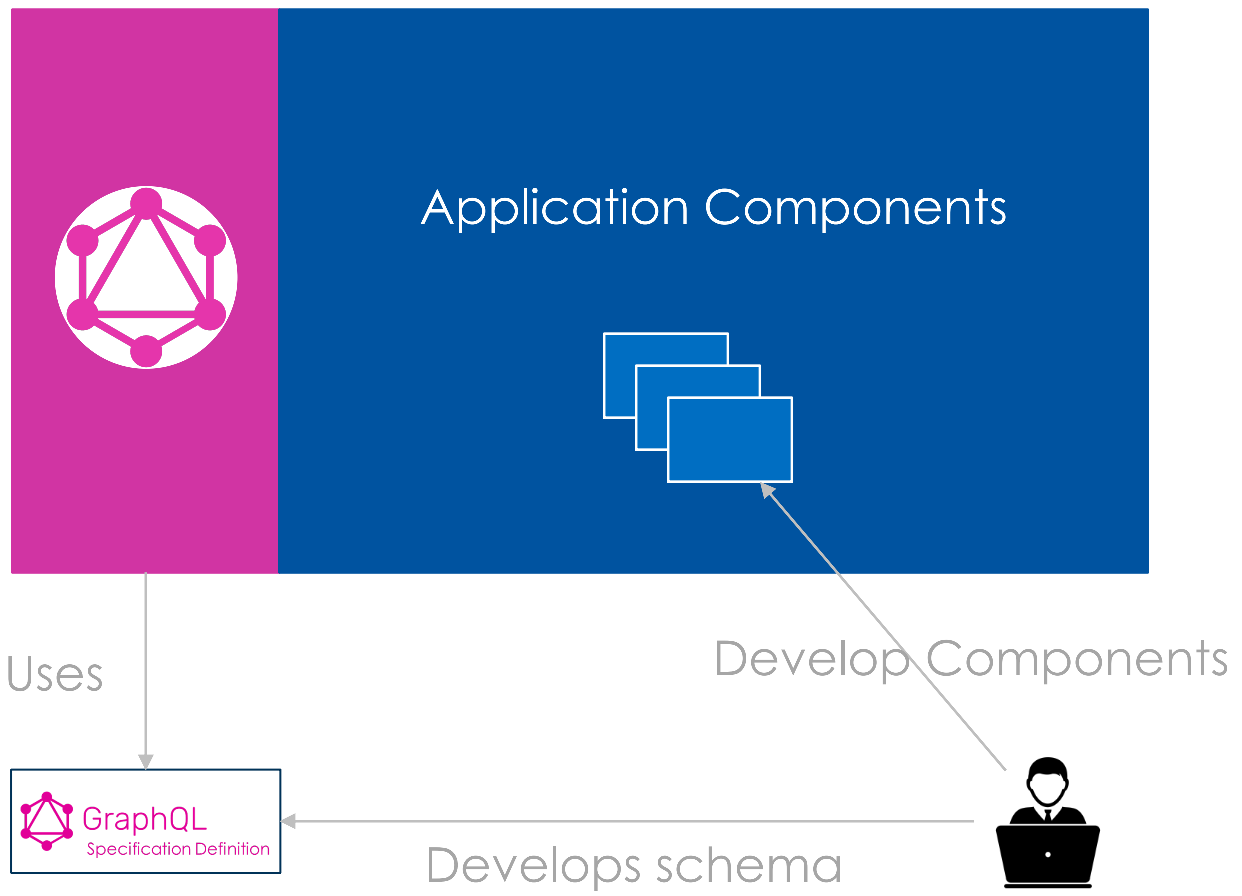


GraphQL Addresses the Over & Under Fetching !!!

GraphQL Server

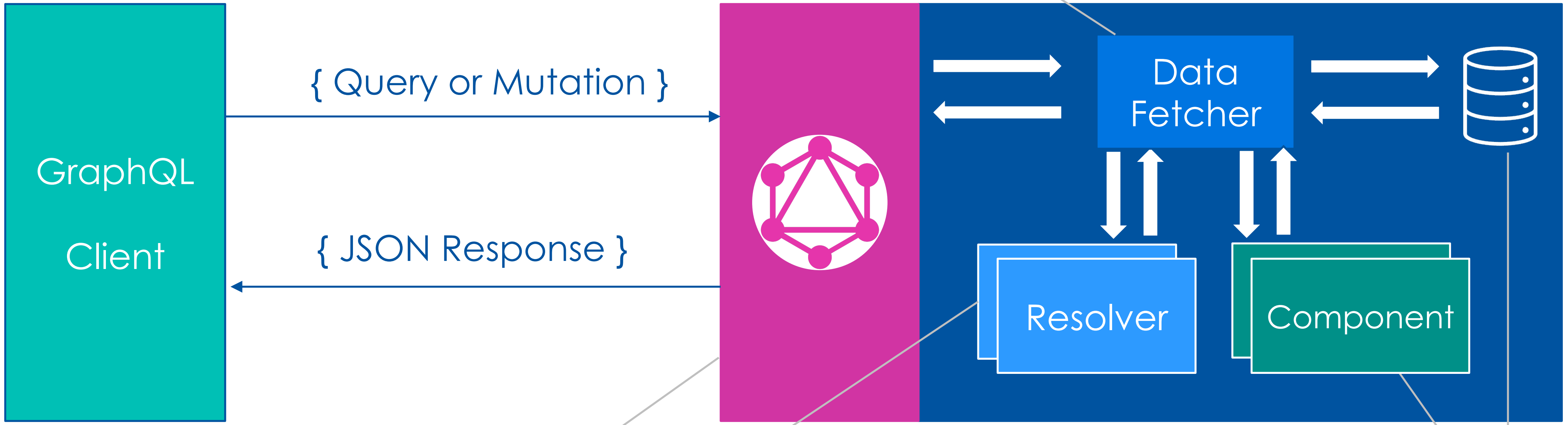
Implemented as a layer to manage client interactions

- Implements GraphQL Specification
- Multiple implementations
- Components depend on framework



General implementation

- Provides implementation of operation



- Provides a function() per field
- Data
- Validates and resolves the query | mutation

GraphQL Language Support

https://graphql.org/code/

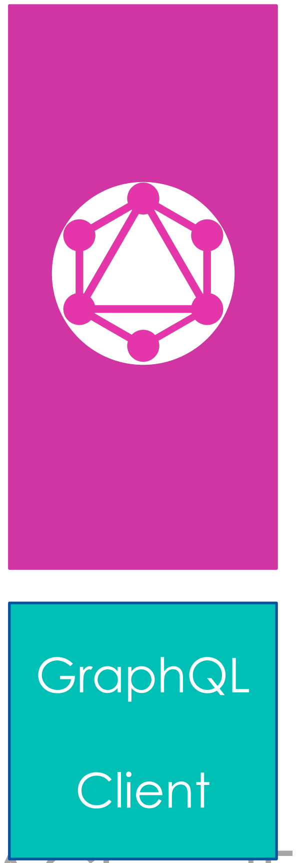
Server implementation & Client libraries

GraphQL Client

JavaScript	Go	PHP	Python	Java / Kotlin	C# / .NET
Ruby	Rust	Elixir	Swift / Objective-C	Scala	Flutter
Clojure	C / C++	Haskell	Elm	OCaml / Reason	Erlang
Groovy	R	Julia	Perl	D	

GraphQL Server

Implementation available in multiple languages



Graphene Python

spring

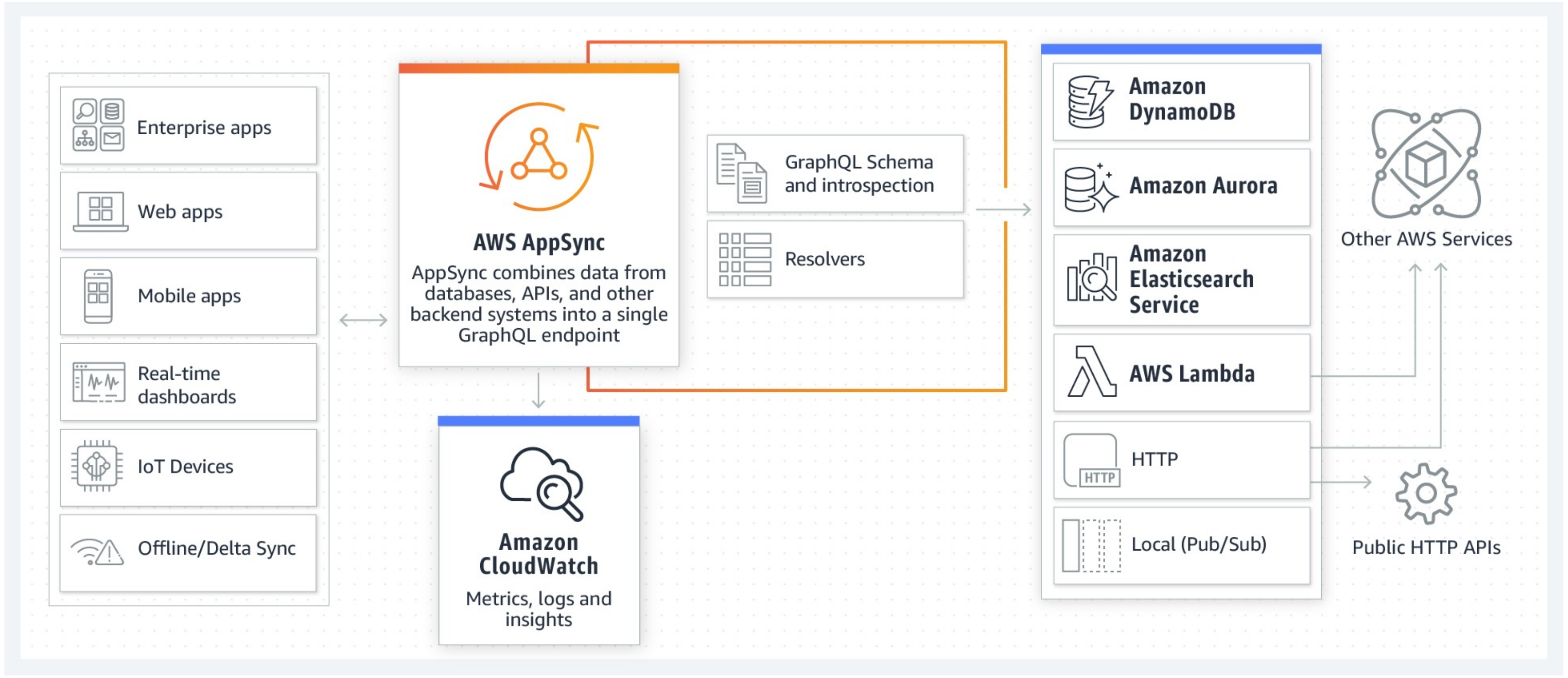
node js express

APOLLO

https://graphql.org/code/

GraphQL Server

Any service may be exposed over the GraphQL !!!



https://aws.amazon.com/appsync/

GraphQL Advantages

No over- and under-fetching by clients

Application developers are in control

Documentation available in the form of Schema

Error responses are detailed

GraphQL Disadvantages

Performance challenges with complex queries

Web caching is not as straightforward

Steeper learning curve compared to REST

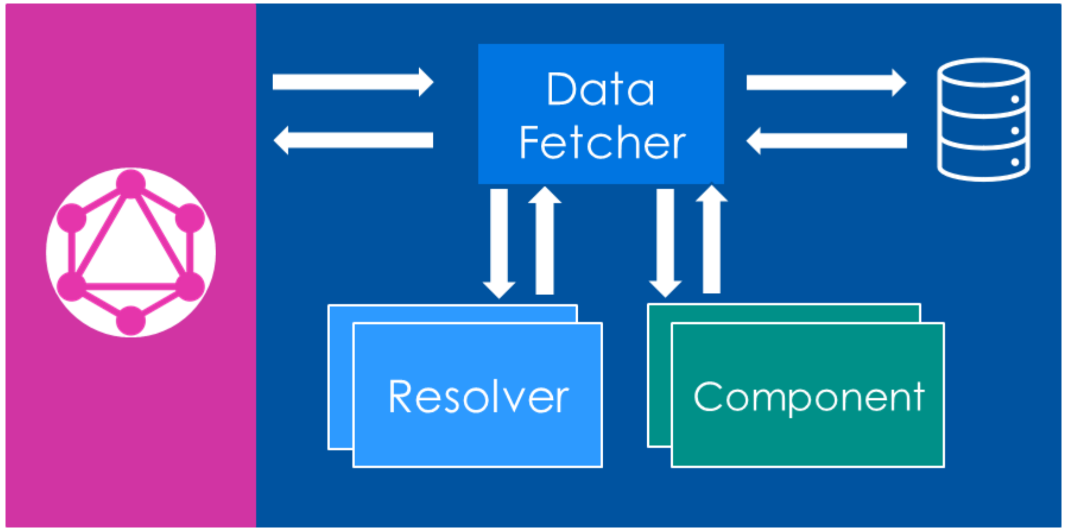
	REST API	GraphQL
Design	• Endpoints & Resources	• Single Endpoint & Schema
Control	• Server controls response	• Client in control
Operations	• CRUD - HTTP Verbs	• Query, Mutation & Subscription
Performance	• Network round trips	• Network traffic reduced
Use Cases	• Resource driven apps	• Data driven apps

Quick Review

GraphQL is a specification for API

- Addresses Under/Over Fetching
- Server implements the specification

GraphQL API developer provides the Schema & application Components needed by Server



Schema Definition Language

An introduction to the SDL

*

1 GraphQL Type System

2 Query execution

3 Schema considerations | tips

Learn more about SDL

https://graphql.org/learn

Intent is to provide an overview; please refer to documentation for details

?

RESTful

GraphQL

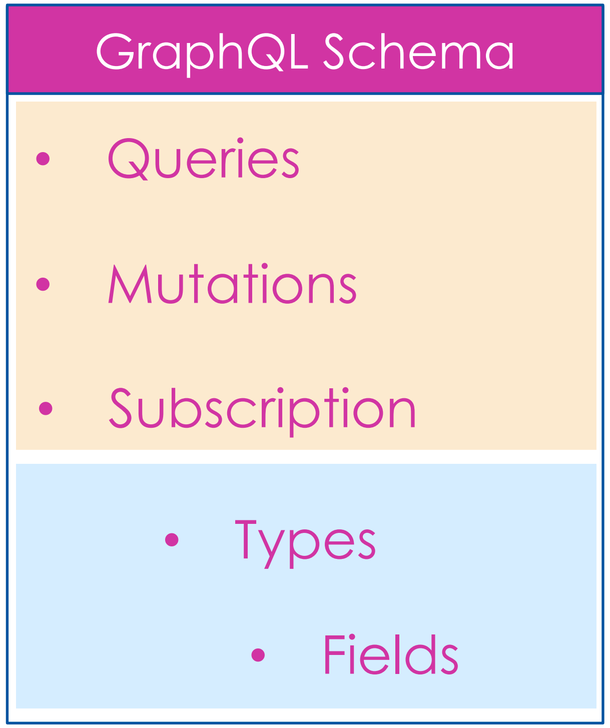
What would you use for Microservices?

They are NOT mutually exclusive

Create API as RESTful | GraphQL depending on the usage

Schema Definition Language (SDL)

Standard language for defining the schema

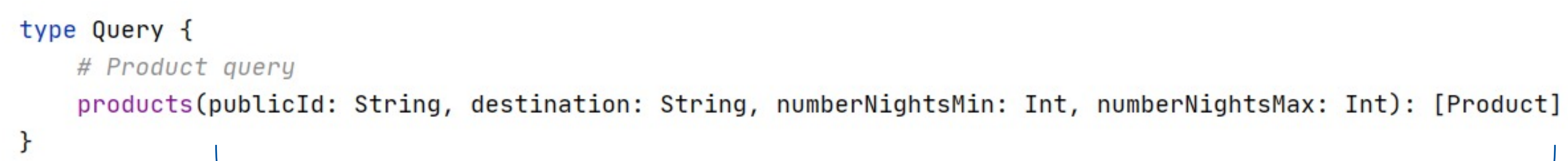


Operations
- Standard operations

Types
- Server defined objects

Root types a.k.a. Operations

query Retrieval of objects defined on the server



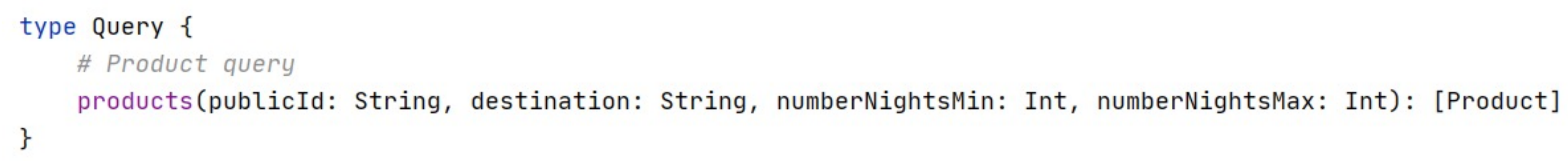
```
type Query {
    # Product query
    products(publicId: String, destination: String, numberNightsMin: Int, numberNightsMax: Int): [Product]
}
```

- Arguments may be marked as required
- Required argument types are suffixed with '!'

E.g., publicId: String!

Root types a.k.a. Operations

query	Retrieval of objects defined on the server



```
type Query {
    # Product query
    products(publicId: String, destination: String, numberNightsMin: Int, numberNightsMax: Int): [Product]
}
```

mutation	Modifies the data on the server
subscription	Server pushes data to client in response to events

GraphQL Type System

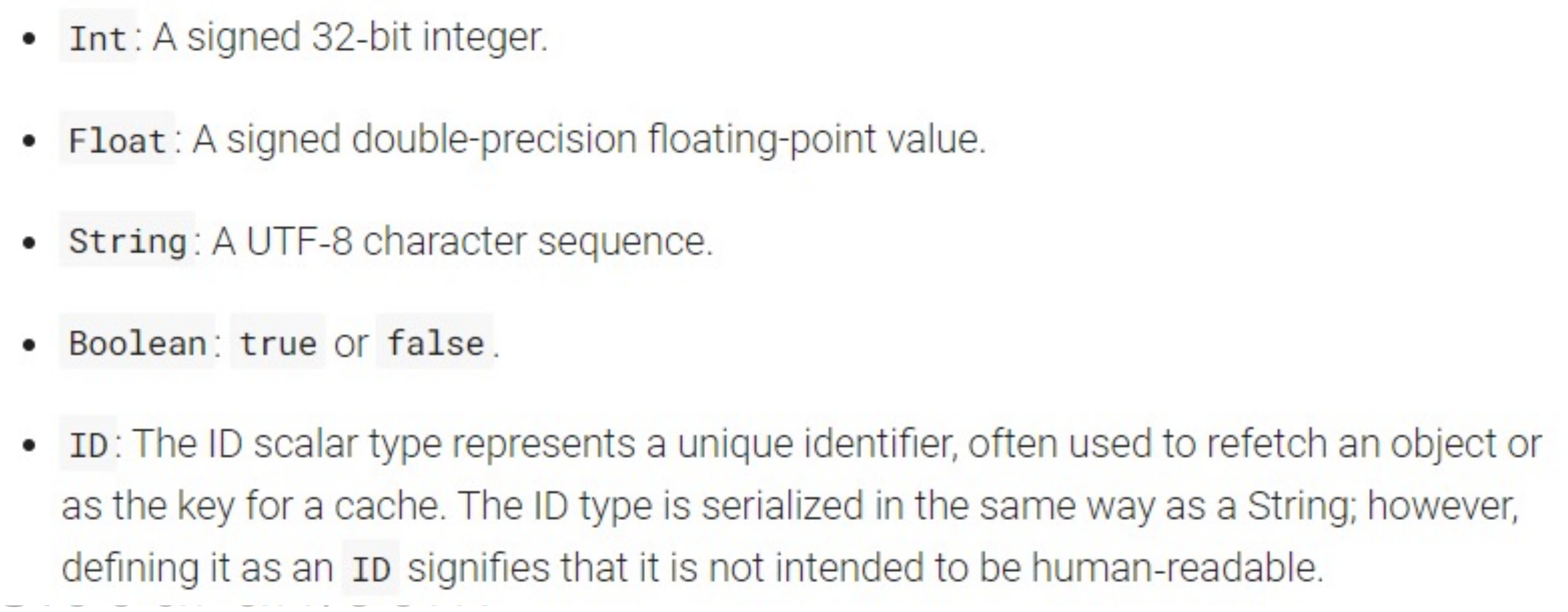
“ GraphQL service defines a set of types which completely describe the set of possible data you can query on the service. The incoming queries are validated and executed against that schema

Types & Fields

type Structure of the domain object definition

Scalar types

- Complex types (server defined)



- `Int`: A signed 32-bit integer.
- `Float`: A signed double-precision floating-point value.
- `String`: A UTF-8 character sequence.
- `Boolean`: `true` or `false`.
- `ID`: The ID scalar type represents a unique identifier, often used to refetch an object or as the key for a cache. The ID type is serialized in the same way as a String; however, defining it as an `ID` signifies that it is not intended to be human-readable.

Types & Fields

type	Structure of the domain object definition
field	Attribute in an object
	An attribute has type : scalar or complex

Example : Types & Fields

Name

```
type Product {
    publicId: String!
    description: String!
    numberNights: Int!
    destination: String!
    providers: [Provider!]!
}
```

Scalar type - REQUIRED

Array of complex type

- Elements are required

Example : Types & Fields

```
type Product {
    publicId: String!
    description: String!
    numberNights: Int!
    destination: String!
    providers: [Provider!]!
}
```

```
type Provider {
    id: Int!
    type: String!
    name: String!
    rating: Int!
    description: String!
}
```

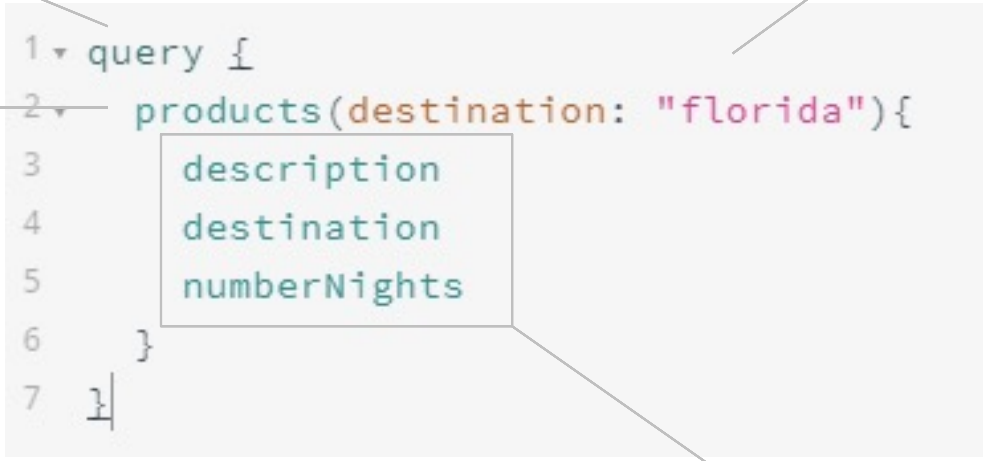
Nested Types considerations

Nested types may impact the performance

Nested types will increase server complexity

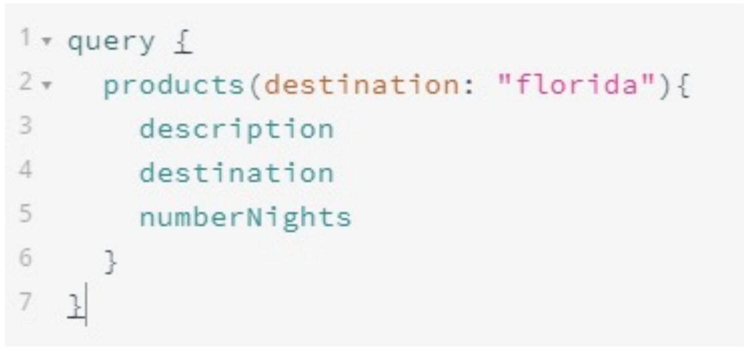
Query Execution

Client sends a JSON like document as a request to server

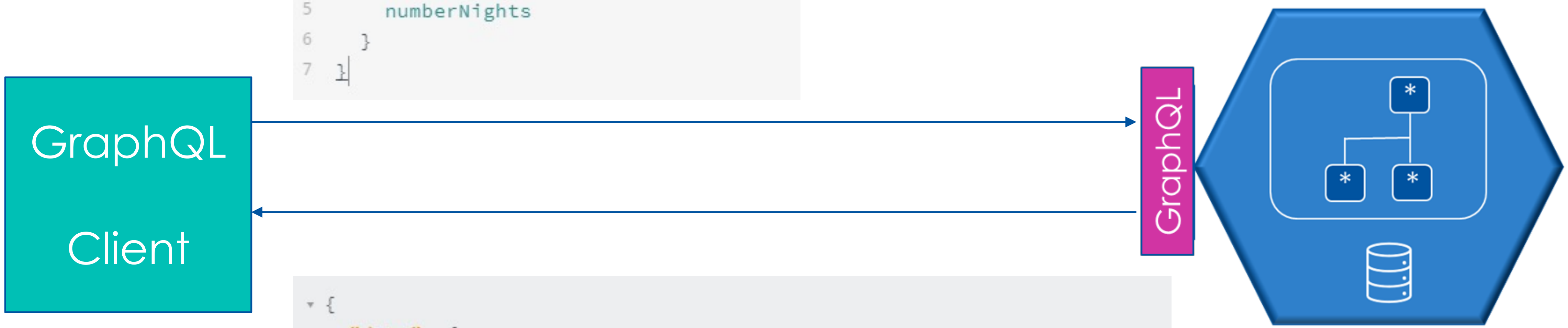

Root type set to query
Zero or more arguments
Named query that server can execute
Fields to be returned in response

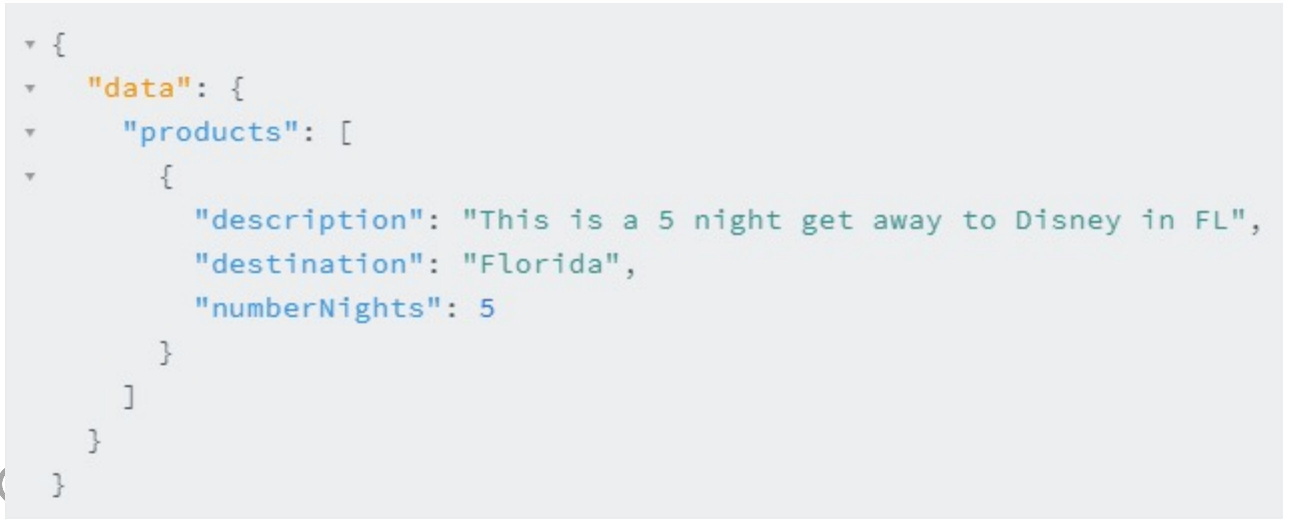
```
query {
  products(destination: "florida"){
    description
    destination
    numberNights
  }
}
```

Example: Query



```
query {
  products(destination: "florida"){
    description
    destination
    numberNights
  }
}
```





```
{
  "data": {
    "products": [
      {
        "description": "This is a 5 night get away to Disney in FL",
        "destination": "Florida",
        "numberNights": 5
      }
    ]
  }
}
```

GraphQL Developer Tools

GraphiQL

A GUI for editing and testing GraphQL queries and mutations

GRAPHQLEDITOR

GraphQL
Playground

GraphDoc

Generate static documentation for schema

Designing the Schema

Think of it as a Shared Language

- It should use the Ubiquitous Language for the domain
- Take an evolutionary approach to create the api
- Design, by thinking about "How" it will be used by clients

https://graphql.org/learn/thinking-in-graphs/

Quick Review

SDL used for defining the schema

Server uses schema to:

- Validate the requests
- Create the responses

Client uses schema to:

- Create the requests
- Parse the responses

ACME Products GraphQL

Giving the developers control of the response

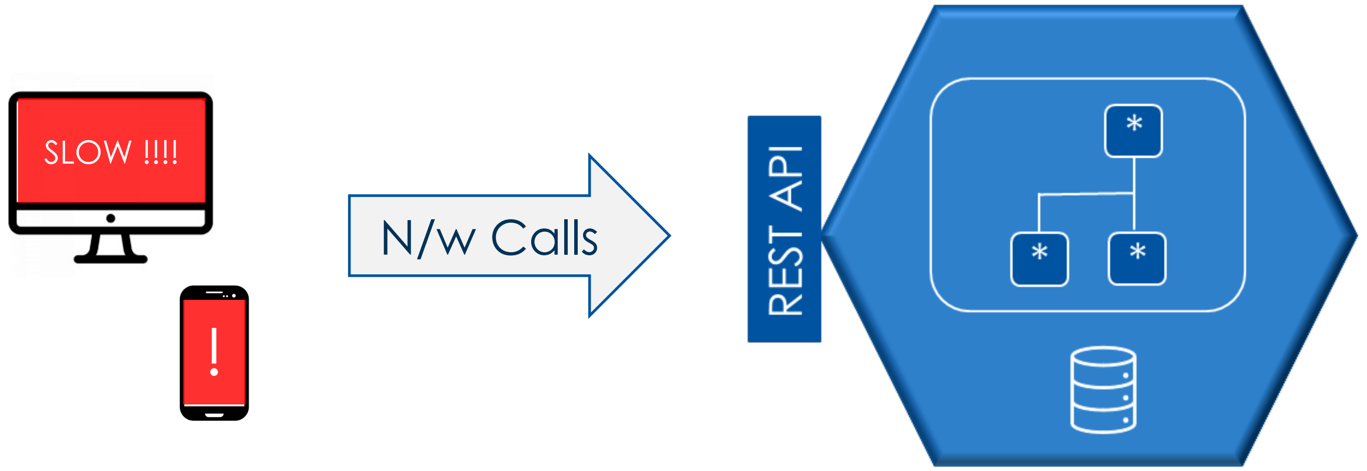
ACME Travel

1. REST API issues
2. Products GraphQL API
3. Schema Definition for Products & Providers

ACME REST API

App developers are complaining

- Requires complex logic to be built in the applications
- Performance is bad due to multiple network calls



IT Lead

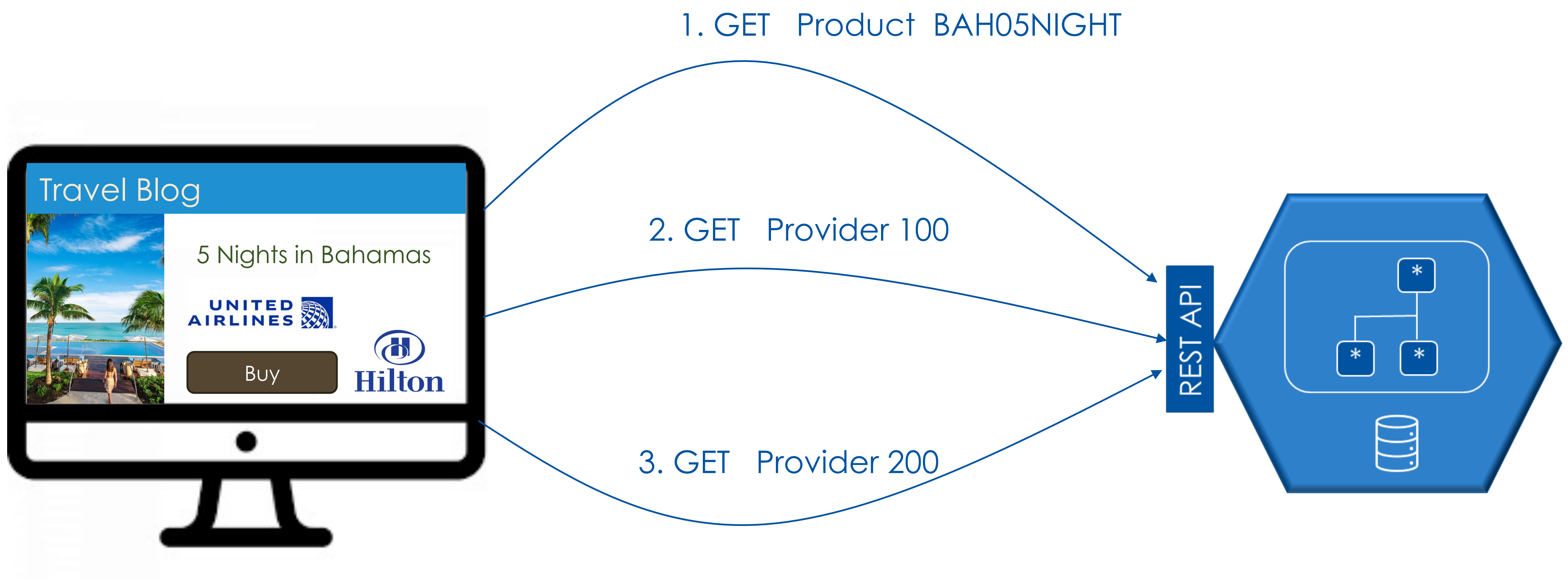
We have an Over-Fetching issue !!!

Reduce the amount of data sent from the API

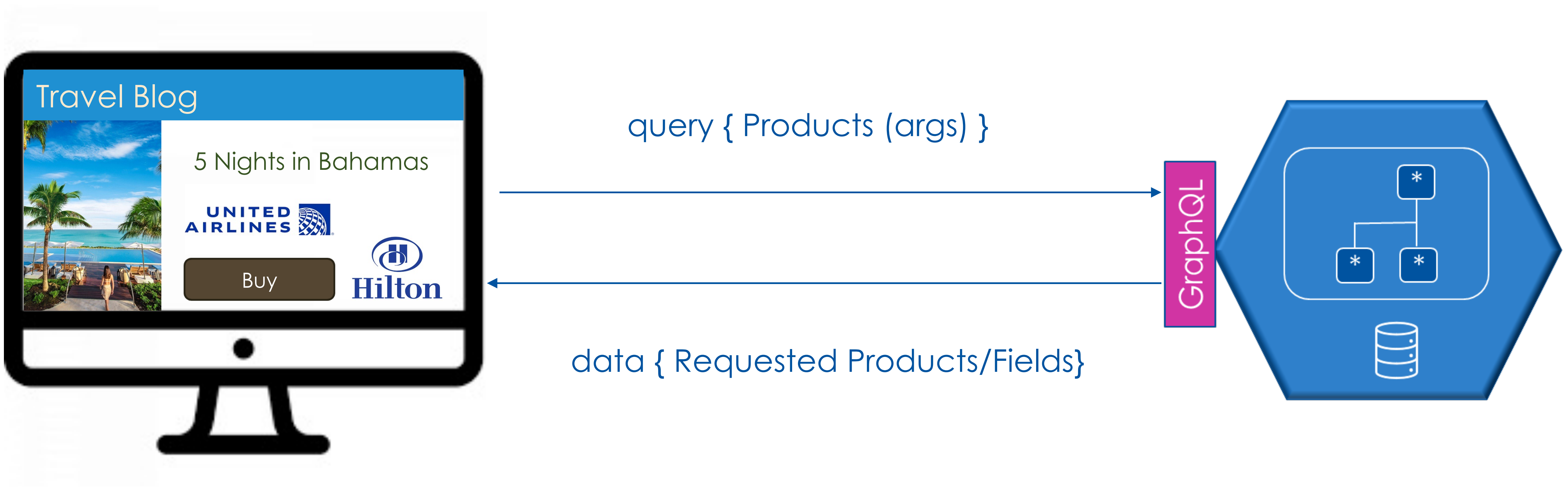
Give developer control over what they want to receive

GraphQL is meant for these kind of scenarios

ACME product REST API



ACME product REST API



Products Query

Put together the schema definition

Query

Product

a.k.a. Bundle

Provider

- Common endpoint for all Query & Mutation operations

  http://host.com/graphql

Products & Providers Query

Schema Definition walkthrough

https://github.com/acloudfan/MSFA-ACME-**Products**-v1.0.git

Branch: api

Products GraphQL Implementation

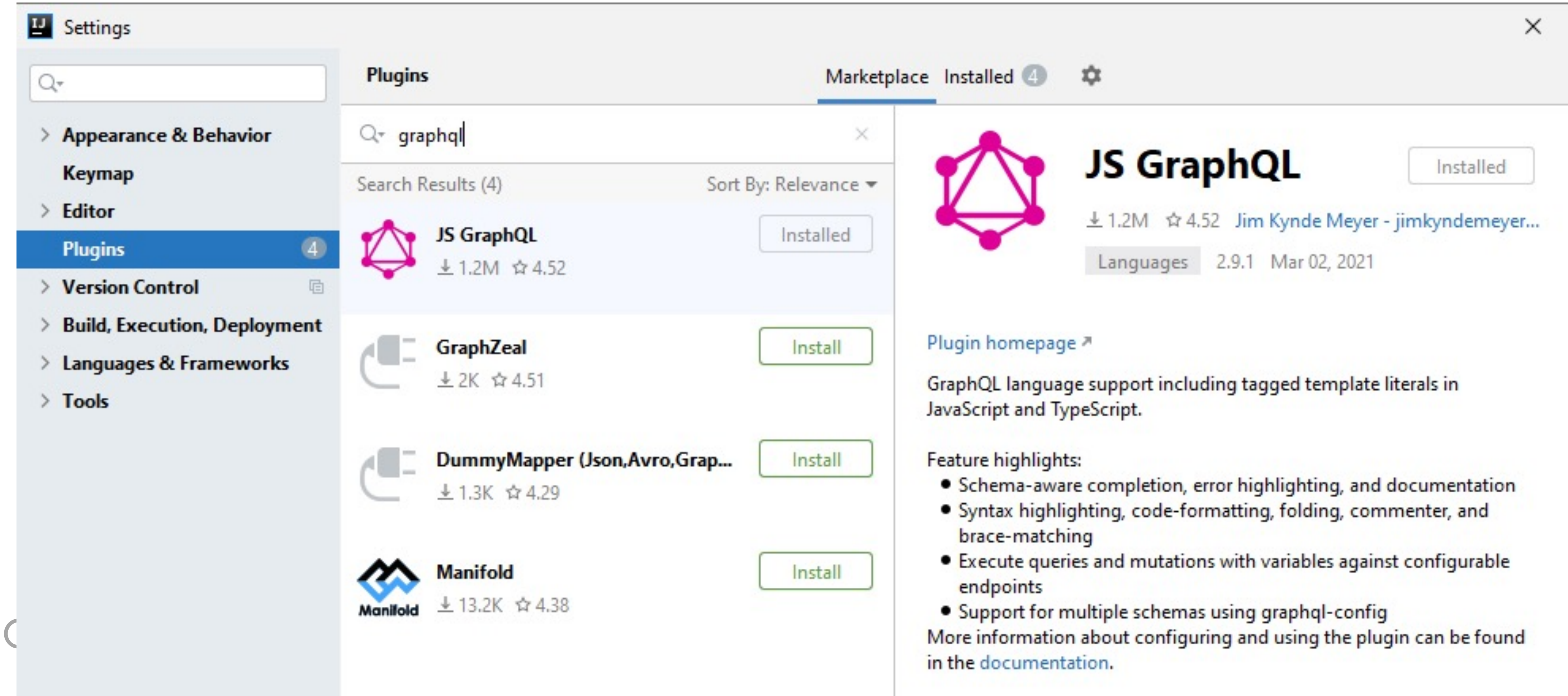
ACME Products GraphQL API in Action

1 GraphQL in Action

2 Class diagram walkthrough

3 Code walkthrough

IntelliJ : GraphQL plugin



JS GraphQL — 1.2M ☆ 4.52 — Installed
GraphZeal — 2K ☆ 4.51 — Install
DummyMapper (Json,Avro,Grap... — 1.3K ☆ 4.29 — Install
Manifold — 13.2K ☆ 4.38 — Install

JS GraphQL — Installed
1.2M ☆ 4.52 Jim Kynde Meyer - jimkyndemeyer...
Languages 2.9.1 Mar 02, 2021

Plugin homepage ↗

GraphQL language support including tagged template literals in JavaScript and TypeScript.

Feature highlights:
- Schema-aware completion, error highlighting, and documentation
- Syntax highlighting, code-formatting, folding, commenter, and brace-matching
- Execute queries and mutations with variables against configurable endpoints
- Support for multiple schemas using graphql-config

More information about configuring and using the plugin can be found in the documentation.

Uses the graphql-java implementation of the specs

https://graphql-java.com/

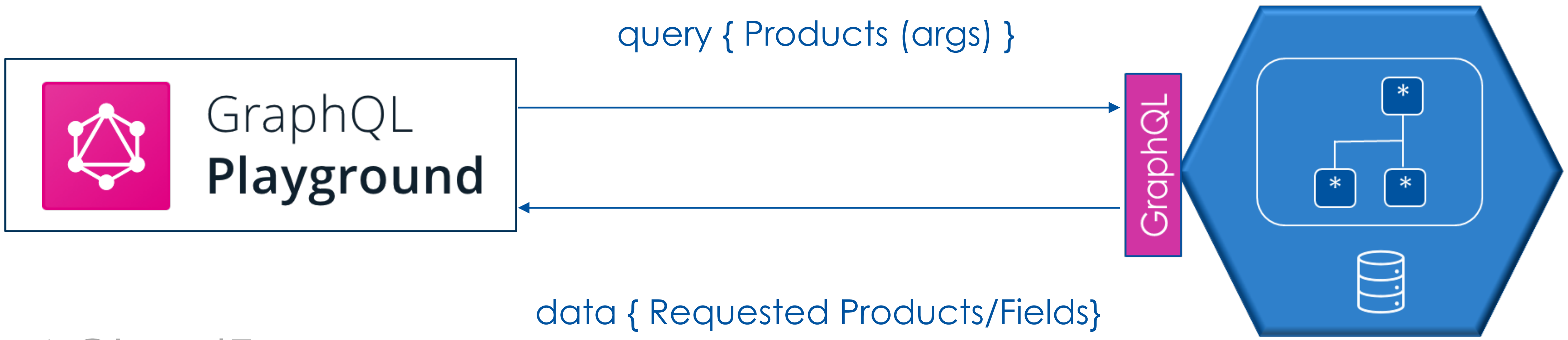
GraphQL **Playground**

https://github.com/graphql/graphql-playground

Testing

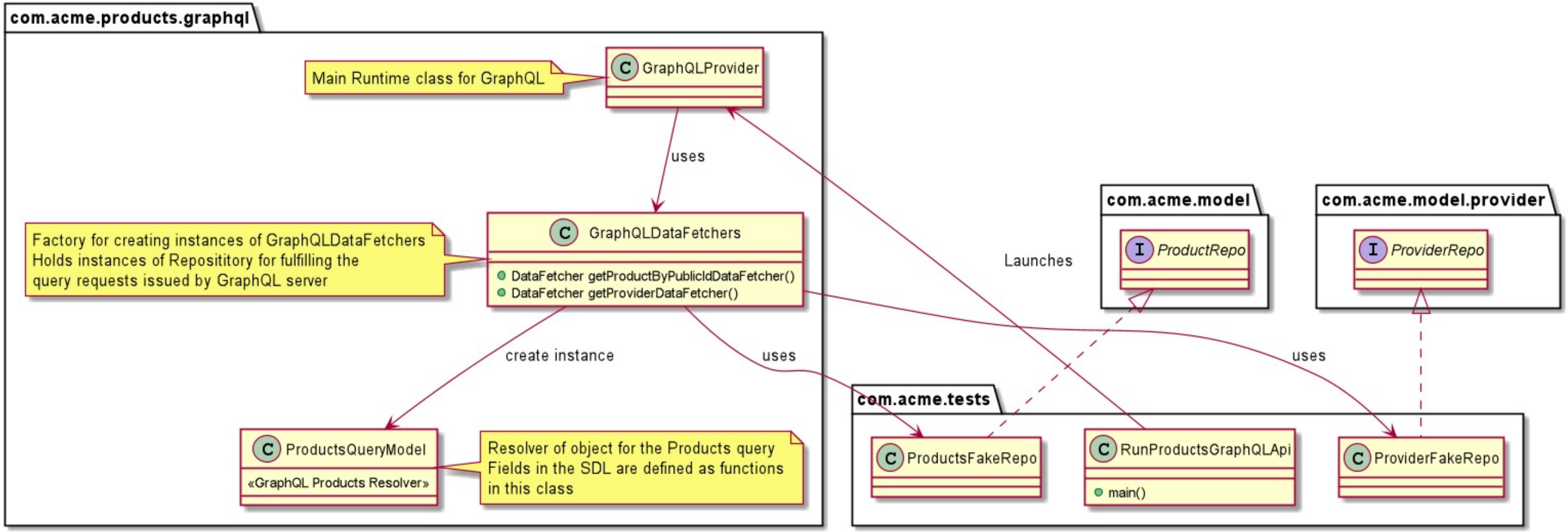
1. Launch the GraphQL API Server

2. Use GraphQL playground to execute the queries



Products GraphQL Classes

uml/api/graphql.class.puml


com.acme.products.graphql
Main Runtime class for GraphQL
GraphQLProvider
uses
Factory for creating instances of GraphQLDataFetchers
Holds instances of Repositiory for fulfilling the
query requests issued by GraphQL server
GraphQLDataFetchers
DataFetcher getProductByPublicIdDataFetcher()
DataFetcher getProviderDataFetcher()
create instance
uses
ProductsQueryModel
«GraphQL Products Resolver»
Resolver of object for the Products query
Fields in the SDL are defined as functions
in this class
Launches
com.acme.model
ProductRepo
com.acme.model.provider
ProviderRepo
uses
com.acme.tests
ProductsFakeRepo
RunProductsGraphQLApi
main()
ProviderFakeRepo

Data Fetchers | Query Resolvers

GraphQL invokes the Data Fetchers for executing ops

GraphQL

Specification Definition

- Data Fetcher for Product
- Data Fetcher for Provider

```
type Query {
    products(publicId: String, destination: String,
        numberNightsMin: Int, numberNightsMax: Int): [Product]
    providers(id: Int!): Provider
}
```

Resolver for Product

GraphQL

Specification Definition

```
# This is the representation the package
type Product {
    publicId: String!
    description: String!
    numberNights: Int!
    destination: String!
    providers: [Provider!]
}
```

```
public class ProductsQueryModel {

    // Holds the package object
    private Product vProduct;

    // Holds the providers
    ArrayList<Provider>  providers;

    public ProductsQueryModel(Product vProduct, ArrayList<Provider>  providers){
        this.vProduct = vProduct;
        this.providers = providers;
    }

    // Exposes the same methods as the Package object
    public String getDescription() { return vProduct.getDescription(); }

    public String getPublicId() { return vProduct.getPublicId(); }

    public String getDestination() { return vProduct.getDestination(); }

    public int getNumberNights() { return vProduct.getNumberNights(); }

    // This one is different from the Package class
    public ArrayList<Provider> getProviders() { return providers; }
}
```

Quick Review

GraphQL

Specification Definition

```
type Query {
    products(publicId: String, destination: String,
        numberNightsMin: Int, numberNightsMax: Int): [Product]
    providers(id: Int!): Provider
}
```