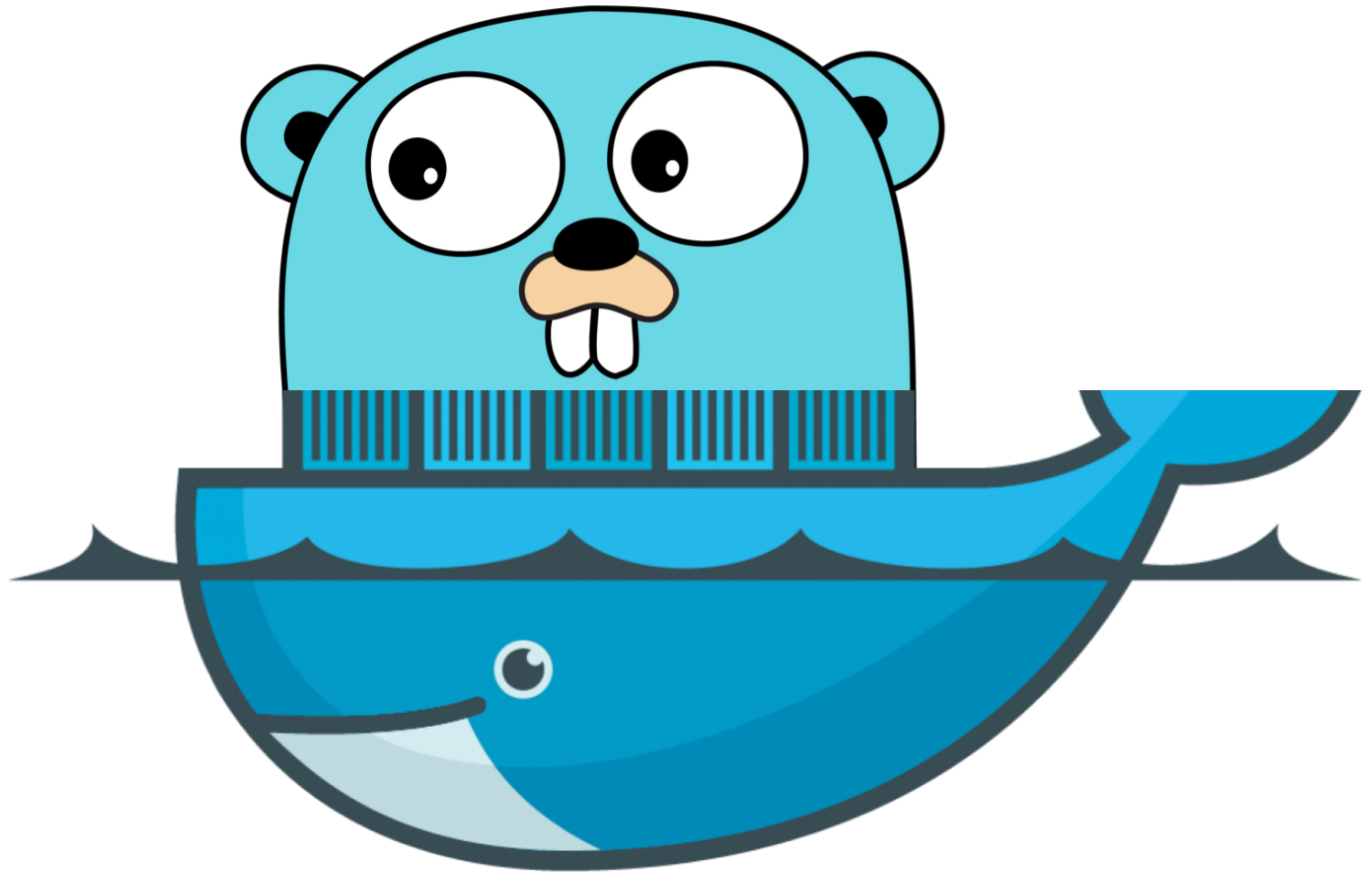
Golang Engineer Training

Simple CRUD REST API
& Simple Clean Architecture



200Lab
Education

Designed by **200lab Education - Nâng tầm chuyên môn, định hướng tương lai**
Mentor/Instructor: Viet Tran - Solution Architect 200lab

Agenda

- New project Golang (in Goland/VSCode IDE)
- Connect to MySQL with GORM (Golang library)
    - Install & use Golang packages.
    - Use environment to improve security
- Introduction to REST API
    - URL Convention
    - Build a simple CRUD REST API
- Simple clean architecture

New project in Goland IDE

- When create a new project, remember uncheck "Index entire GOPATH".
- Setup File Watcher: use "fmt" to format code when we save code.
- Setup your favorite theme (if you want).

200Lab Education

Connect to MySQL with GORM

- In terminal:
    - "go get -u gorm.io/gorm@v1.20.11"
    - "go get -u gorm.io/driver/mysql@v1.0.3
- Open file main.go:

```
import (
  "gorm.io/driver/mysql"
  "gorm.io/gorm"
)

func main() {
  // refer https://github.com/go-sql-driver/mysql#dsn-data-source-name for details
  dsn := "user:pass@tcp(127.0.0.1:3306)/dbname?charset=utf8mb4&parseTime=True&loc=Local"
  db, err := gorm.Open(mysql.Open(dsn), &gorm.Config{})
}
```

Use environment to improve security!!

```
import (
  "gorm.io/driver/mysql"
  "gorm.io/gorm"
  "os"
)

func main() {
   dns := os.GetEnv("DBConnectionStr")
   db, err := gorm.Open(mysql.Open(dsn), &gorm.Config{})
}
```

Connect to MySQL with GORM

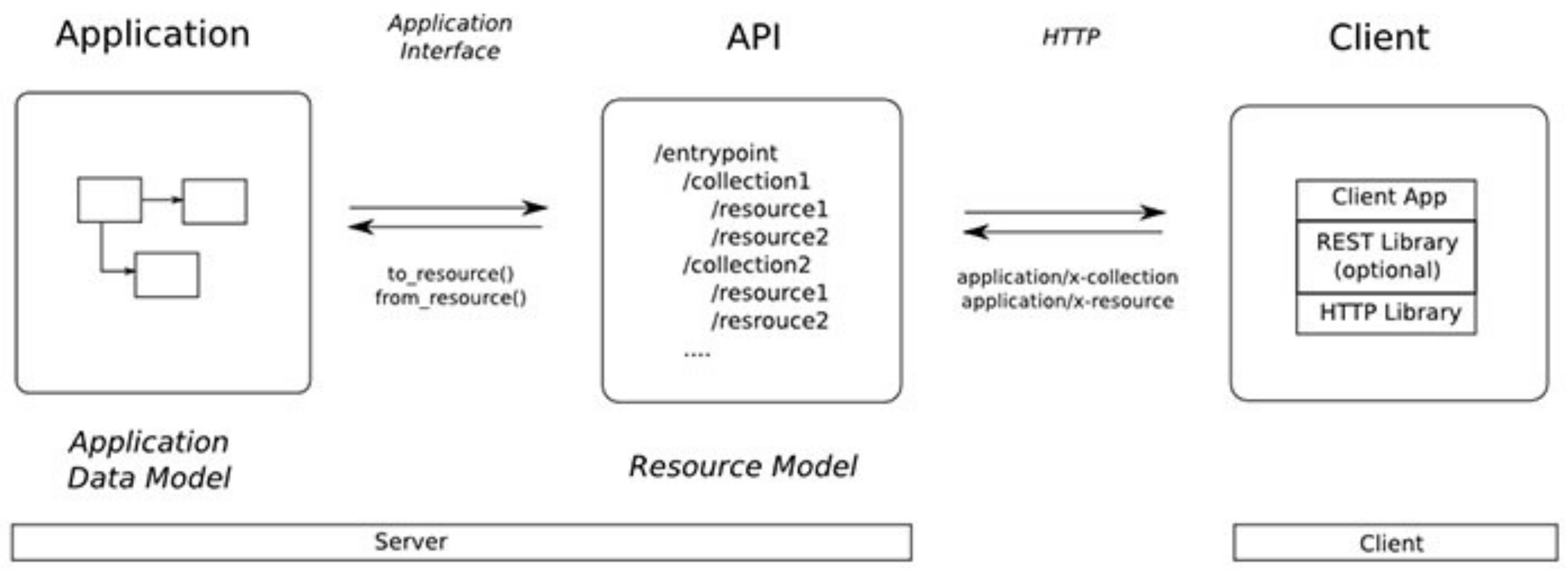
Demo in Golang

Connect, Insert, Delete, Select & Update

https://gorm.io

200Lab Education

Introduction to REST API

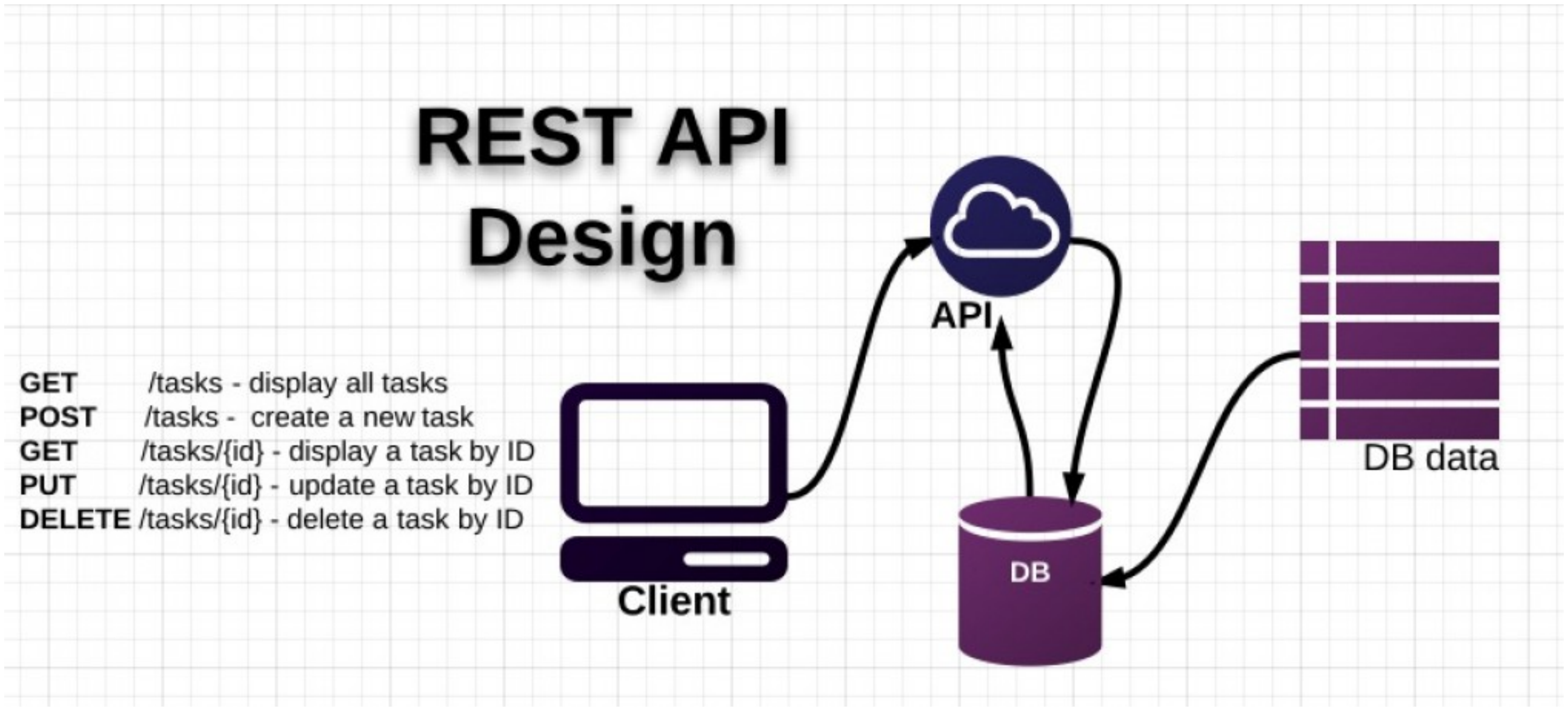


- REST = **RE**presentational **S**tate **T**ransfer
- Common methods: POST, GET, PUT, PATCH, DELETE
- Convention: https://restfulapi.net/resource-naming

200Lab
Education

On next section
(Because of lacking of time)

Simple convention in REST API

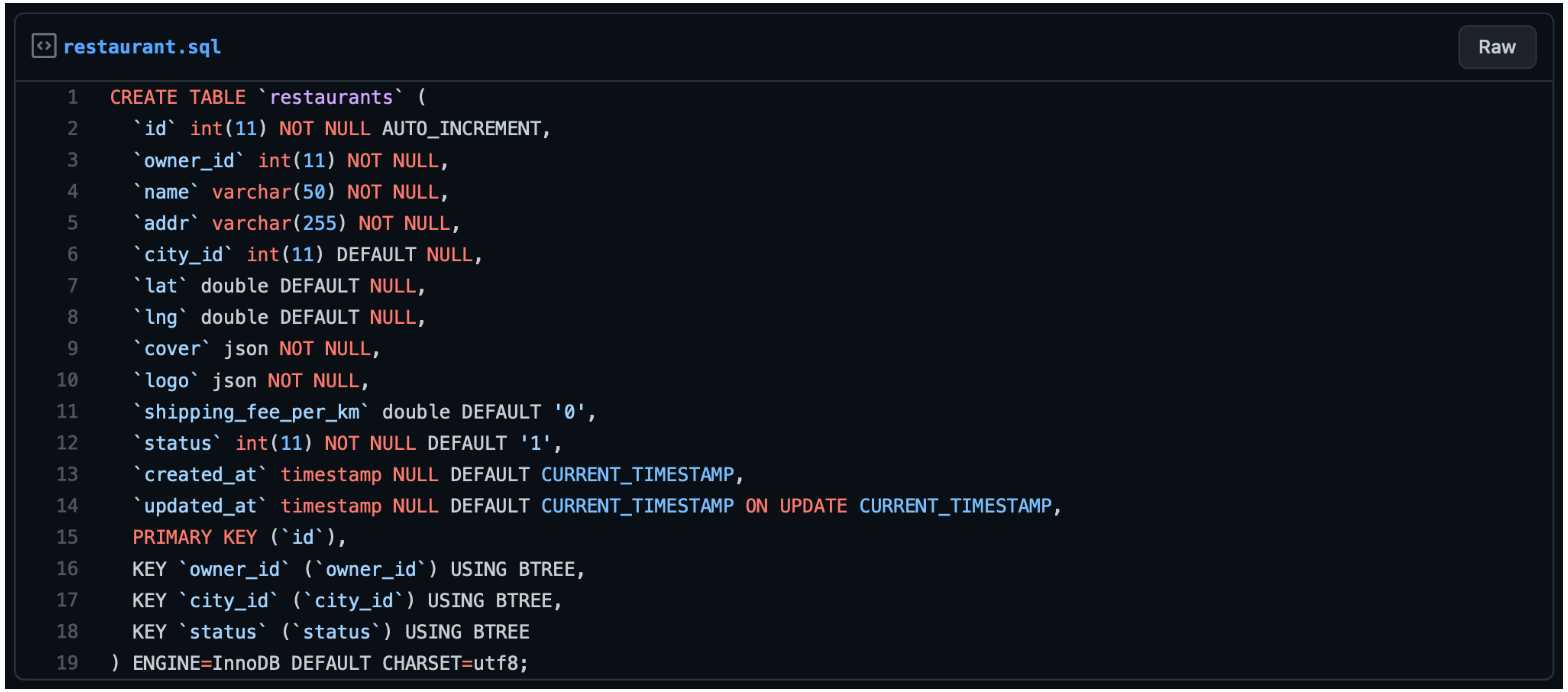


GET	/tasks - display all tasks
POST	/tasks - create a new task
GET	/tasks/{id} - display a task by ID
PUT	/tasks/{id} - update a task by ID
DELETE	/tasks/{id} - delete a task by ID

200Lab Education

Demo in Golang
Simple CRUD (Restaurant)

Data Scheme: https://gist.github.com/viettranx/b0a22a0a869309fc9c64fd820b1d0f29

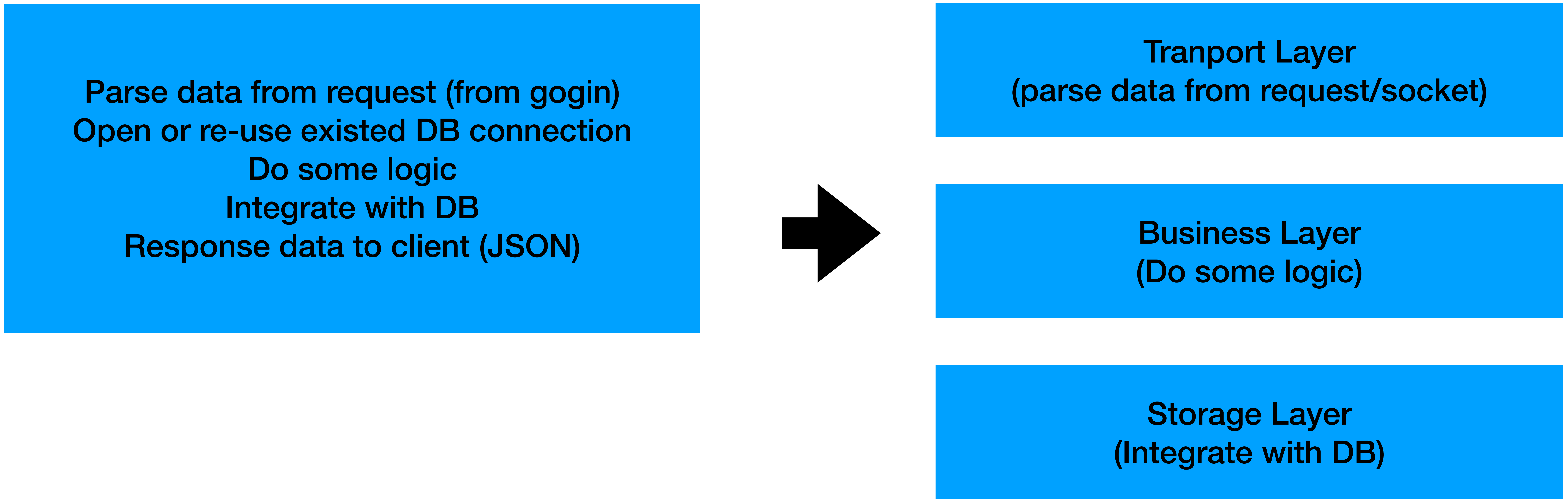


restaurant.sql Raw

```sql
CREATE TABLE `restaurants` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `owner_id` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `addr` varchar(255) NOT NULL,
  `city_id` int(11) DEFAULT NULL,
  `lat` double DEFAULT NULL,
  `lng` double DEFAULT NULL,
  `cover` json NOT NULL,
  `logo` json NOT NULL,
  `shipping_fee_per_km` double DEFAULT '0',
  `status` int(11) NOT NULL DEFAULT '1',
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  KEY `owner_id` (`owner_id`) USING BTREE,
  KEY `city_id` (`city_id`) USING BTREE,
  KEY `status` (`status`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

200Lab Education

Improve architecture


Parse data from request (from gogin)
Open or re-use existed DB connection
Do some logic
Integrate with DB
Response data to client (JSON)
Tranport Layer
(parse data from request/socket)
Business Layer
(Do some logic)
Storage Layer
(Integrate with DB)

Do it yourself

- Write some simple CRUD. Ex: Restaurant, Food, Category
- Challenge (break your limit):
    - Write a Register API. Be careful with storing user password.
    - Write a Login API, use JWT as access token.
    - Or how to return a list of Food (include category information in particular item).

200Lab will **review** and give you some **advice** you when you finish! Try your best!!

200Lab Education

Thank you.