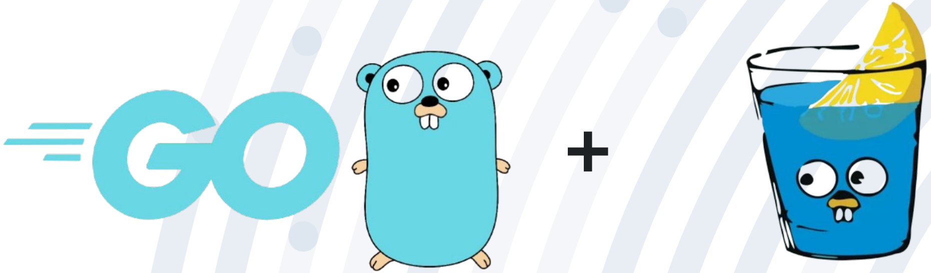
VietSoftware
INTERNATIONAL
Your Collaborative Development Partner

GO SEMINAR

Part 2: Building APIs with Golang



Presented by

GLEY Kodjo Achille

23/08/2024

PLAN

1 Introduction to Building APIs

2 Golang API frameworks

3 Golang API architecture

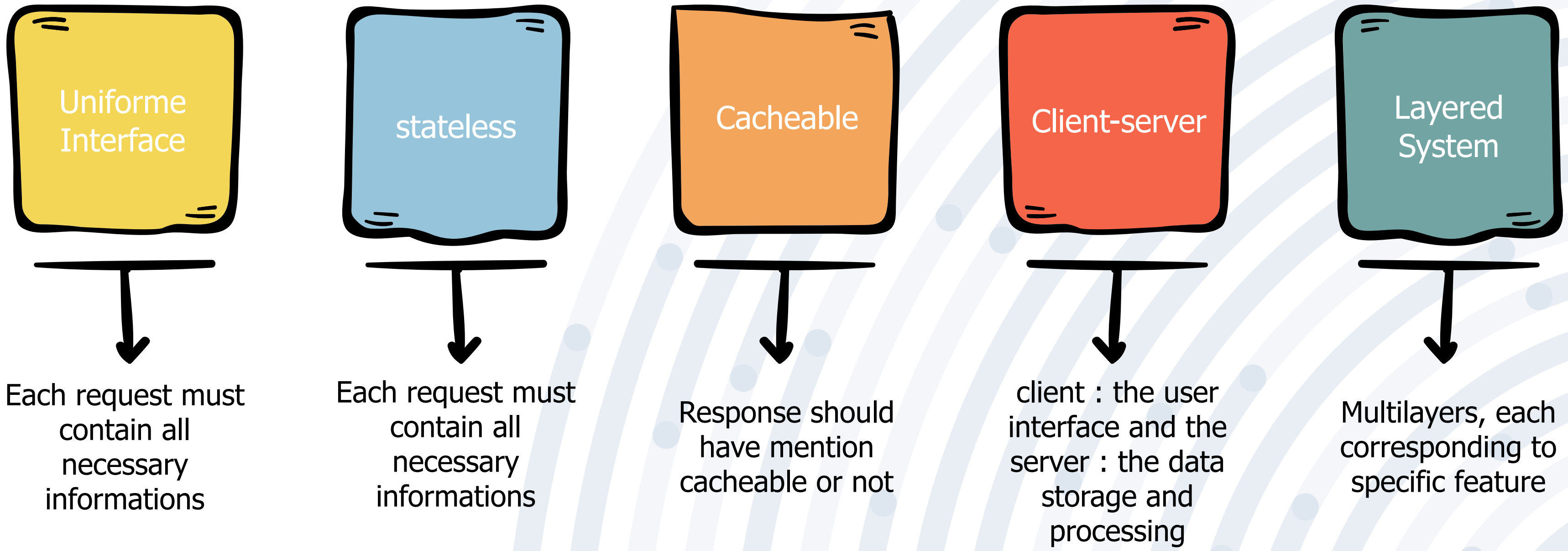
4 Golang api creation flow “use case Task manager”

5 Testing demo

6 Api deployment and Conclusion

1. Introduction to building API

- **RestFul Api principles**



Uniforme Interface

Each request must contain all necessary informations

stateless

Each request must contain all necessary informations

Cacheable

Response should have mention cacheable or not

Client-server

client : the user interface and the server : the data storage and processing

Layered System

Multilayers, each corresponding to specific feature

1. Introduction to building API

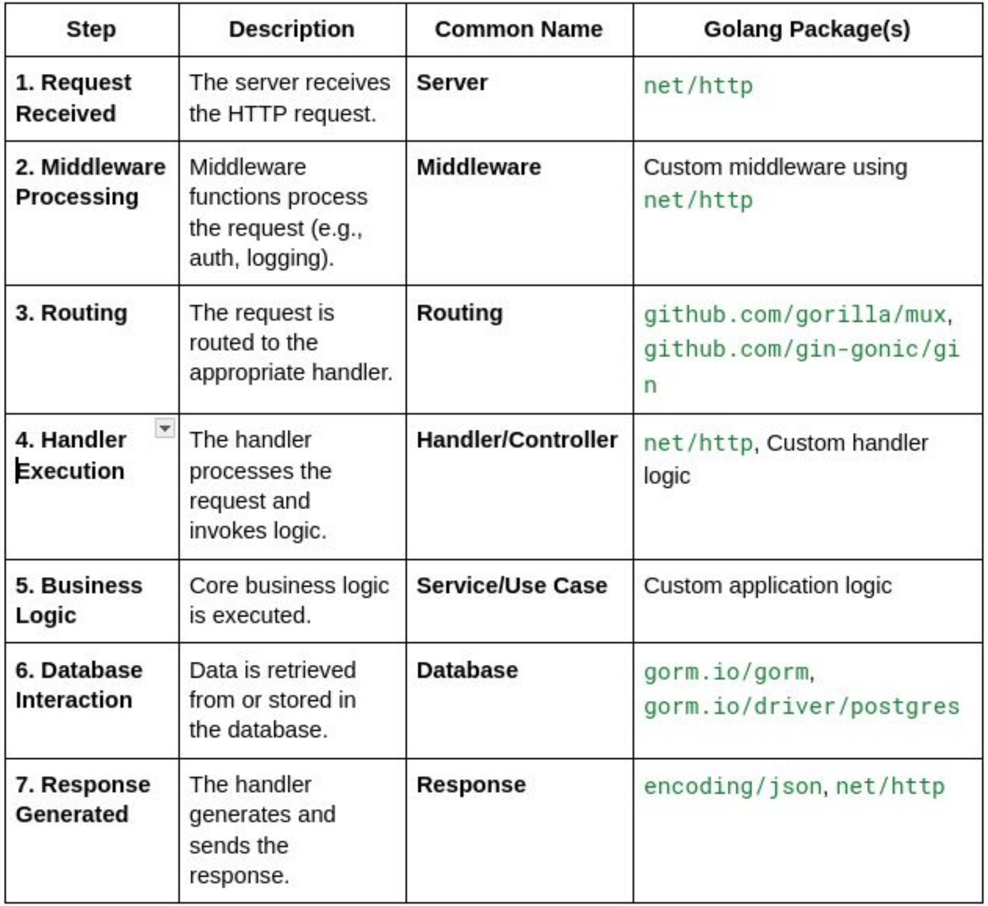
- **Rules**

URI	HTTP verb	Description
api/users	**GET**	Get all users
api/users/new	**GET**	Show form for adding new user
api/users/1	**GET**	Get a user with id=1
api/users	**POST**	Add a user
api/users/1/edit	**GET**	Show form for user with id=1
api/users/1	**PUT**	Update a user with id= 1
api/users/1	**DELETE**	Delete a user with id=1

2. Golang api frameworks

S.No	Framework	Icon	Features
1	Gin		• Data binding • JSON utilities • Fast HTTP router
2	Echo		• WebScokets • JSON/XML support • Middleware range
3	Beego	BEEGO	• Cache handling • MVC architecture • Automatic API docs
4	Fiber	Fiber	• Low memory usage • Fast HTTP engine • Built-in rate limiter
5	Gorilla		• URL path matching • Request handling • Middleware support

3. Golang api architecture



Step	Description	Common Name	Golang Package(s)
1. Request Received	The server receives the HTTP request.	**Server**	`net/http`
2. Middleware Processing	Middleware functions process the request (e.g., auth, logging).	**Middleware**	Custom middleware using `net/http`
3. Routing	The request is routed to the appropriate handler.	**Routing**	`github.com/gorilla/mux`, `github.com/gin-gonic/gin`
4. Handler Execution	The handler processes the request and invokes logic.	**Handler/Controller**	`net/http`, Custom handler logic
5. Business Logic	Core business logic is executed.	**Service/Use Case**	Custom application logic
6. Database Interaction	Data is retrieved from or stored in the database.	**Database**	`gorm.io/gorm`, `gorm.io/driver/postgres`
7. Response Generated	The handler generates and sends the response.	**Response**	`encoding/json`, `net/http`

4. Golang api creation flow

- **Use case “Task Manager”**

5. Testing demo

- **Use case “Task Manager”**

6. Deployment and Conclusion

- **Use case “Task Manager”**

Thanks!

Any questions?