Golang Programming

Interfaces. Http handlers. Routing with ServeMux. Sorting

Homework 3 (Building simple web application) (1)

Implement a web server in Golang that allows to Create, Update, Read, and Delete (CRUD) Users:

1. Implement concrete type User with following fileds: ID, Username, Password, Email, FirstName, LastName, Enabled, Expired, and Role (allowed roles should be User and Admin).
2. User should implement Stringer interface with a method String() discussed during the lecture.
3. The user database (map) should be initialized with a default ADMIN user
4. Implement an HTTP server and handlers exposing the following HTTP methods and URLs, as well as a HTTP client (net/http) for testing these methods and URLs:

(- continues on the next slide -)

Homework 3 (Building simple web application) (cont)

- GET /users?sortBy={fileldname} – returns JSON array representation of all users collection sorted by the specified field or by ID if sortBy query parameter missing;
- POST /users – uses the request body in JSON format (Content Type: application/json) to create a new User and returns status code 201 Created with Location header containing the URL to GET the new user created, returns status code 400 Bad Request if the user data sent is invalid;
- GET /user/{userID} – returns JSON representation of the User with specified userID, or 404 Not Found if there is no user with specified userID;
- PUT /user/{userID} – updates the state of the User with specified ID and returns status 200 OK if successful, 404 Not Found if there is no user with specified ID, or 400 Bad Request if the user data sent is invalid;
- DELETE /user/{userID} – deletes the User with specified userID and returns status 200 OK, or 404 Not Found if there is no user with specified userID.

* To marshal/unmarshal users data into/from JSON format use methods `son.MarshalIndent()` and `json.Unmarshal()` presented on the next slide.

JSON Marshalling and Unmarshalling

```
// Structs --> JSON
data, err := json.Marshal(goBooks)
if err != nil {
        log.Fatalf("JSON marshaling failed: %s", err)
}
fmt.Printf("%s\n", data)

// Prettier formatting
data, err = json.MarshalIndent(goBooks, "", "     ")
if err != nil {
        log.Fatalf("JSON marshaling failed: %s", err)
}
fmt.Printf("%s\n", data)

// JSON -> structs
var books []Book
if err := json.Unmarshal(data, &books); err != nil {
        log.Fatalf("JSON unmarshaling failed: %s", err)
}
fmt.Println("AFTER UNMARSHAL:\n", books)
```