|  |  |  |  |
| --- | --- | --- | --- |
| P1 Group 37 | Brian Wu | Emerson Coronel | Logan Golia |
| netID | bw35 | edc6 | lsg3 |
| Github id | bwubuilder | EmersonCoronel | logangolia |

Comp 318 Design Document

Our database service is designed with three folders that contain packages handling functionality of those services:

The first is the authentication folder, which contains authorization.go. This package has function makeToken, which generates a randomized string token to map to the user, and function authHandler, which switches between the cases of POST (for login) and DELETE (for logout).

* POST takes the username from the request, maps it to a token, and returns the token to the user.
* DELETE checks token expiration then deletes the token from the store.

The second is the database folder, which handles the backend functionality of the database.

* Collection.go and document.go both define the structures which will be operated upon in the database. They both contain New functions for database creation and marshalDocument functions to check for json.Marshal errors.
* Parser.go takes the path returned from the html request and splits it into distinct parts.
* Database\_service.go contains the DatabaseService struct which is operated on by HTML methods defined in the package.
* (Add section for subscribe depending on if it is included in the methods or separately created in a helper function)

The third folder, handler, contains handler.go. Handler.go creates a new http.Handler and database service, and ‘handles’ switching between method requests on the database.

Handler gets used by main.go, which sets the port, schema, and token ptrs, then sets the server address, and then assigns the handler to the server. Upon running main.go, the server will be open and listening for events.

Design Principles

Open-Closed Principle:

By defining the functions separately in the database\_service package, only handling the functionality of actually operating on the database, we can extend its functionality while always keeping the ability to modify/post/delete the database. For example, the functions now utilize parser.go, which helps to parse the path of the html hyperlink, and subscribe.go, which handles user subscription to various documents or collections in the database.

Decoupling:

(interfaces)

Composition:

The collection.go and document.go files in our database compose the DatabaseService struct. The two different kinds of objects which can be stored in the database have different properties, and are thus composed in two different packages. However, they can both be operated upon in the database service by the various HTML methods, and they can both contain each other, the database struct takes a top level collection and can be composed of documents and collections as more data gets added.

Concurrency

main.go

authentication.go

handler.go

document.go

collection.go

parser.go

database\_service.go

* PUT
* GET
* POST
* PATCH
* DELETE
* OPTIONS