

Photo-Server

PROJEKTARBEIT

Programmieren II - Go

von

Rudolf Gaab, Johannes Weis, Yannis Luithle

Inhalt

1	Architekturdokumentation.....	4
1.1	Photo-Server Main	4
1.2	Handler	5
1.3	packageObjects	5
1.4	packageTools	6
1.5	Batchupload Main	6
2	Anwenderdokumentation.....	7
2.1	Login/Registration	7
2.2	Persönliche Gallery.....	7
2.3	Bilder bestellen & kommentieren	8
2.4	Orders.....	9
2.5	Upload	10
2.6	Diashow (Feature)	11
3	Dokumentation des Betriebs - PhotoServer	12
3.1	Umgebung	12
3.2	Parameter	12
3.3	Dateien	12
4	Dokumentation des Betriebs – Batchupload	14
4.1	Umgebung	14
4.2	Parameter	14
5	Kurzbeschreibung der Beiträge zur Projektumsetzung	15
5.1	Rudolf Gaab	15
5.2	Johannes Weis.....	15
5.3	Yannis Luithle	15
6	Quellen	17

Gruppenmitglieder:

Name	-	Matrikelnummer
Rudolf Gaab	-	9122564
Johannes Weis	-	2227134
Yannis Luithle	-	3886565

1 Architekturdokumentation

1.1 Photo-Server Main

Der Photo-Server besteht neben der main noch aus den Ordner

- packageHandler für die Handler der Endpunkte,
- packageObjects für die benötigten Structs/Objekte und deren Funktionen wie speichern, laden und verwalten,
- packageTools für weitere Nützliche Tools wie das Auslesen der EXIF-Daten, das Zippen oder weiterer allgemeiner Funktionen

In der Main werden die Parameter für die Ports geholt und für das Starten und Initialisieren des Servers verwendet.

```
var port = flag.Int( name: "port", value: 4443, usage: "Port")
var certificates = flag.String( name: "certificates", value: "static/ssl", usage: "SSL-Certificates")
```

Zudem werden hier alle Endpunkte definiert und an weitere Handlerfunktion delegiert.

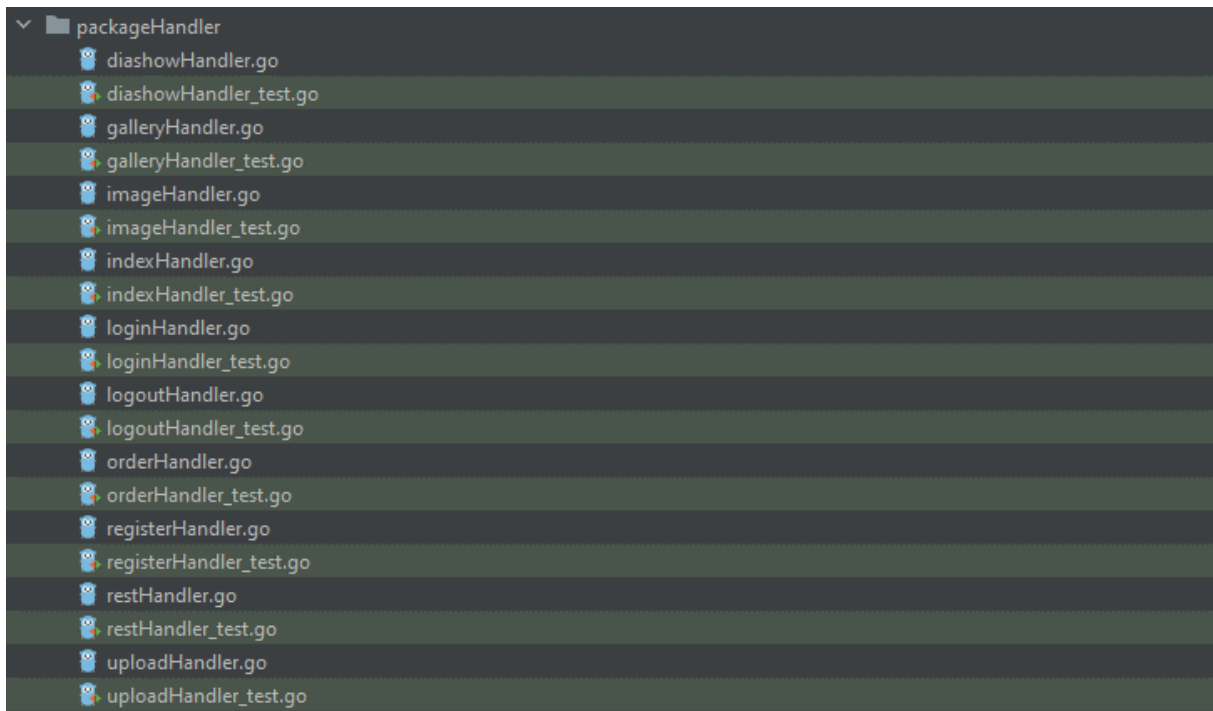
```
http.HandleFunc( pattern: "/", packageHandler.IndexHandler)
http.HandleFunc( pattern: "/login", packageHandler.LoginHandler)
http.HandleFunc( pattern: "/register", packageHandler.RegisterHandler)
http.HandleFunc( pattern: "/logout", packageHandler.LogoutHandler)
http.HandleFunc( pattern: "/diashow", packageHandler.DiashowHandler)
http.HandleFunc( pattern: "/upload", packageHandler.UploadHandler)
http.HandleFunc( pattern: "/gallery", packageHandler.GalleryHandler)
http.HandleFunc( pattern: "/order", packageHandler.OrderHandler)
http.HandleFunc( pattern: "/image", packageHandler.ImageHandler)

http.HandleFunc( pattern: "/api", packageHandler.RESTHandler)

fs := http.FileServer(http.Dir("./static/images"))
http.Handle( pattern: "/images/", http.StripPrefix( prefix: "/images", fs))
```

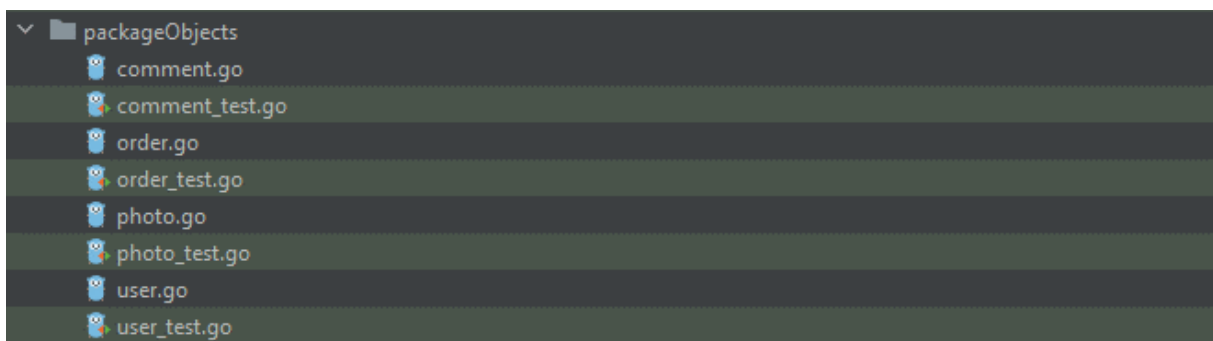
1.2 Handler

Als nächstes gibt es die Handlerklassen, welche alle Handler enthalten.



1.3 packageObjects

Als nächstes gibt es die packageObjects. Alle Handler greifen auf diese Funktionen zu, da diese die Funktionen für das Speichern, Laden und verwalten enthalten. In den Handlern kann somit schnell die Session oder die Bilder abgerufen werden.



Ein Beispiel aus der comment.go: Es wird zunächst ein Typ definiert und anschließend mit Funktionen die Daten verwaltet.

```
type Comment struct {  
    Comment string `json:"comment"`  
    Date    string `json:"date"`  
    Hash    string `json:"hash"`  
}
```

Gibt eine Liste alle Kommentare für einen Nutzer wieder:

```
func GetAllCommentsByUser(username string) *[]Comment
```

Erstellt einen neuen Kommentar und fügt ihn der Liste hinzu. Diese wird dann gespeichert:

```
func AddComment(username string, hash string, commentStr string) *Comment
```

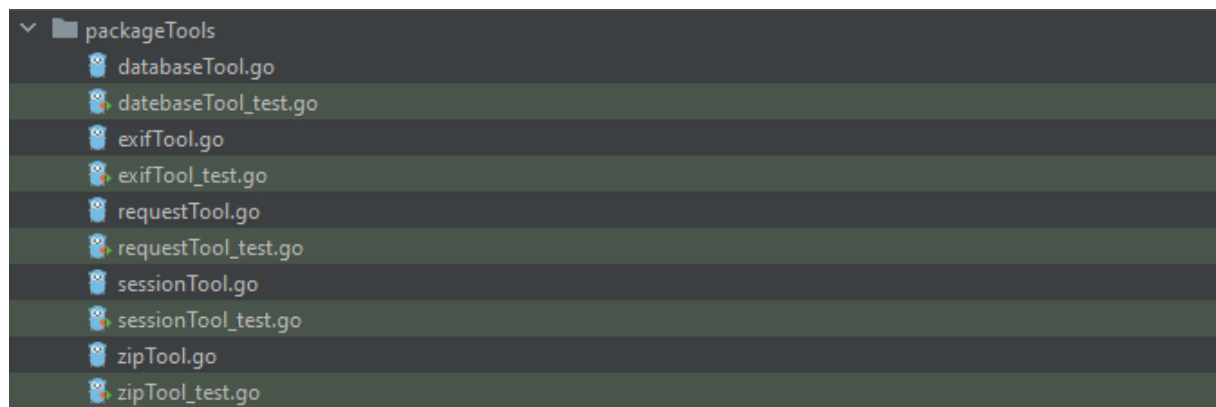
Filtern der Kommentare nach einem Hash. Gibt eine Resultatliste zurück:

```
func FilterAllCommentsByHash(comments *[]Comment, hash string) *[]Comment
```

Mit den anderen Funktionen geschieht dies ähnlich.

1.4 packageTools

Die packageTools enthalten wichtige Funktionen, welche ausgelagert wurden oder sich gut in Tools zusammenfassen können.



1.5 Batchupload Main

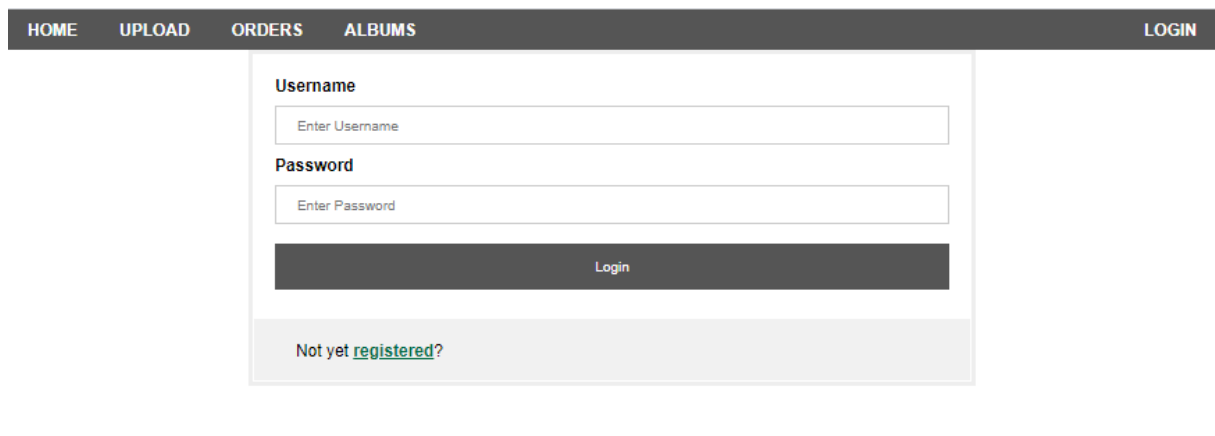
Im Batchupload werden die Parameter eingelesen und das requestTool aus den packageTools verwendet. Es wird jede Datei einzeln hochgeladen. So wird mit dem ganzen Ordner verfahren.

2 Anwenderdokumentation

Dieses Kapitel soll als Durchführungsbeispiel dienen. Es werden die einzelnen Templates auf dem Browser Google Chrome aufgezeigt. Der Nutzer kann mithilfe der Leiste oben durch die verschiedenen Templates navigieren.

2.1 Login/Registration

Auf der Home-Seite wird der Nutzer dazu aufgefordert sich Anzumelden oder zu Registrieren, wenn dieser noch nicht eingeloggt ist. Drückt dieser dann auf das Fenster rechts erscheint der Nutzer auf dieser Seite (Abbildung 1: Login-Bereich)



The screenshot shows a web application interface. At the top, there is a dark navigation bar with the following links: HOME, UPLOAD, ORDERS, ALBUMS, and LOGIN. Below the navigation bar, the main content area is divided into two sections. The left section contains a login form with the following elements: a label 'Username' above a text input field with placeholder text 'Enter Username'; a label 'Password' above a text input field with placeholder text 'Enter Password'; a dark 'Login' button; and a link 'Not yet [registered?](#)' below the button. The right section is empty.

Abbildung 1 Login-Bereich

Hier hat der Nutzer die Möglichkeit sich entweder Einzuloggen kann sich jedoch, wenn er das noch nicht getan hat einen „Account“ anlegen. Dabei erscheint ein weiteres Fenster und der Nutzer kann sein Wunsch-Nutzernamen eintragen und durch zweimaliges eintragen eines Wunsch-Passwortes es somit bestimmen. Es erscheint eine Willkommensnachricht, welche einen mithilfe von Hyperlinks zu den essentiellen Seiten führt.

2.2 Persönliche Gallery

Besitzt der Nutzer schon einen Account, kann er sich anmelden und gelangt danach sofort in seine persönliche Gallery (siehe Abbildung 2: persönliche Gallery). Für die

Anwendungsdokumentation wurde ein Nutzer „hneemann“ mit Beispielfotos um den kompletten Anwendungsumfang der Webservices aufzuzeigen.

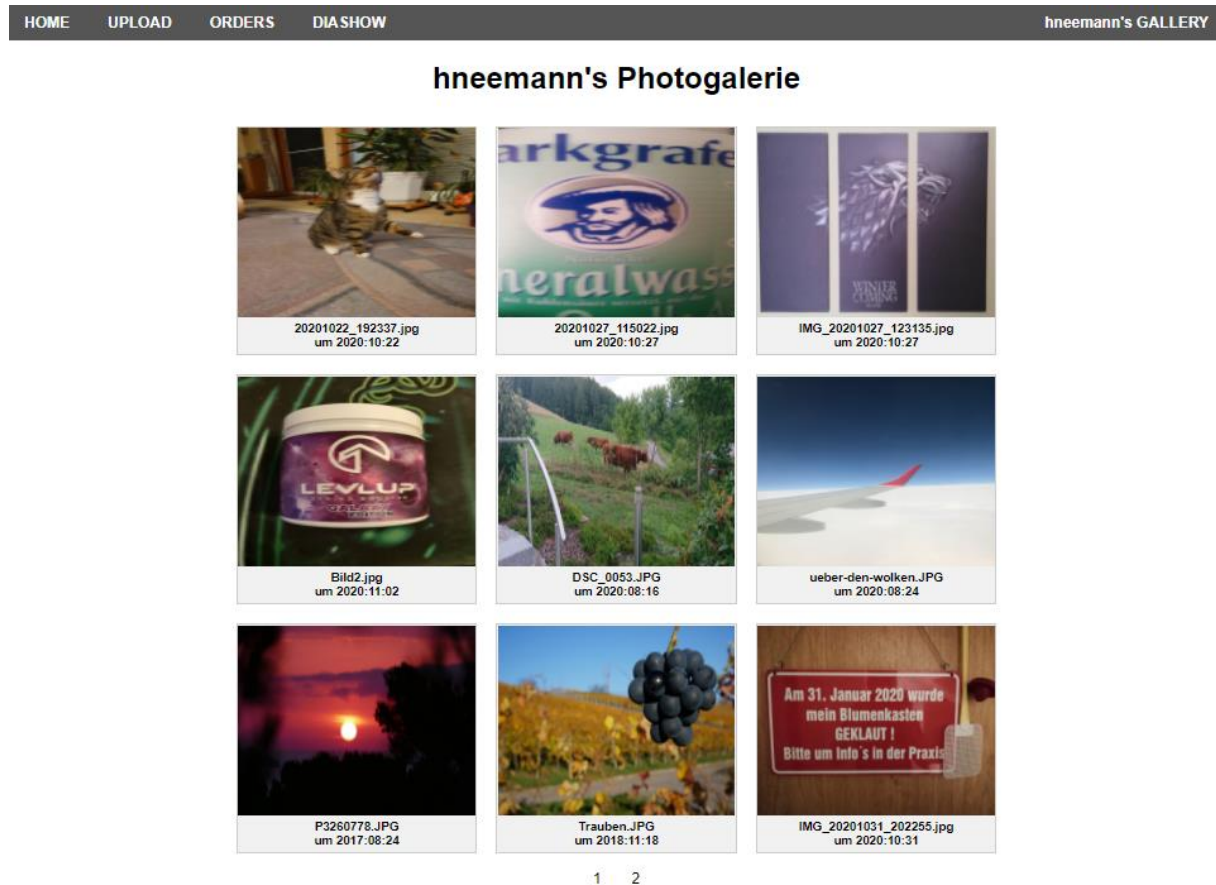


Abbildung 2 persönliche Gallery

In der Persönlichen Gallery sieht der Nutzer die zuerst hinzugefügten Bilder erscheinen als erstes. Unter jedem Bild steht der Dateiname und das Datum wann dieses Bild erstellt/geschossen wurde. Mithilfe von Pagination kann der Nutzer ganz unten auf die nächste Seite gehen, in der er dann Bilder mit neuerem Hochladdatum zu finden sind.

2.3 Bilder bestellen & kommentieren

Klickt der Nutzer auf das Bild in der Gallery gelangt er zum Kommentar- und Bestellbereich des Bilder (siehe Abbildung 3). In dieser Sektion kann der Nutzer eine beliebige Anzahl an Bilder in drei Formaten bestellen. Unter dem Bestellfeld (siehe Abbildung 3: grüner Bereich) besitzt der Nutzer die Möglichkeit einen Kommentar zu

verfassen (siehe Abbildung 3: blauer Bereich). In diesem Beispiel wurden einige Kommentare eingefügt. Diese sind von alt nach neu, oben angefangen, chronologisch aufgereiht. Bei jedem Kommentar steht noch das Verfassungsdatum dabei. Kommentare können mithilfe von dem Textfeld unten verfasst werden.

The screenshot shows the 'hneemann's GALLERY' website. At the top is a navigation bar with links: HOME, UPLOAD, ORDERS, DIASHOW. The main content area displays a photo of a cat jumping, titled '20201022_192337.jpg', with a timestamp 'Geschossen am 2020:10:22'. To the right of the photo is a form titled 'Füge das Bild deiner Bestellliste hinzu'. This form includes an 'Amount' input field, a 'Format' dropdown menu set to '3x4', and an 'Add to Order' button. Below this form is a 'Kommentare' section, which is highlighted with a blue border. It contains three comments: 'Das ist eine wirklich schöne Katze' (dated 2021.01.10 14:15:01), 'Weihnachten war schön' (dated 2021.01.10 14:15:17), and 'Silvester auch' (dated 2021.01.10 14:15:27). At the bottom of the comment section is a 'Schreibe einen Kommentar' form with an 'Enter Comment' input field and a 'Kommentar absetzen' button.

Abbildung 3 Kommentar- und Bestellbereich

Auf der Kommentar- und Bestellseite sieht der Nutzer ebenfalls nochmal den Dateinamen und das Datum an dem das Bild geschossen wurde.

2.4 Orders

Über die Navigationsleiste oben, kann der Nutzer auf die Bestellübersicht(=Orders) gelangen.

Dort besitzt er eine Komplette Übersicht seiner Bestellungen (siehe Abbildung 4: Bestellübersicht). Er kann die Bilder nochmals einsehen in welcher Anzahl diese bestellt wurden und in welchem Format. Wurde ein Bild in zwei Formaten bestellt werden für jeweils ein weiteres Format eine weitere Bestellung hinzugefügt. Jede Bestellung kann einzeln storniert/gelöscht werden. Es kann aber auch die komplette

Bestellliste gelöscht oder auch bestellt werden. Im zweiten Fall bekommt der Nutzer die Bilder in einer ZIP heruntergeladen.

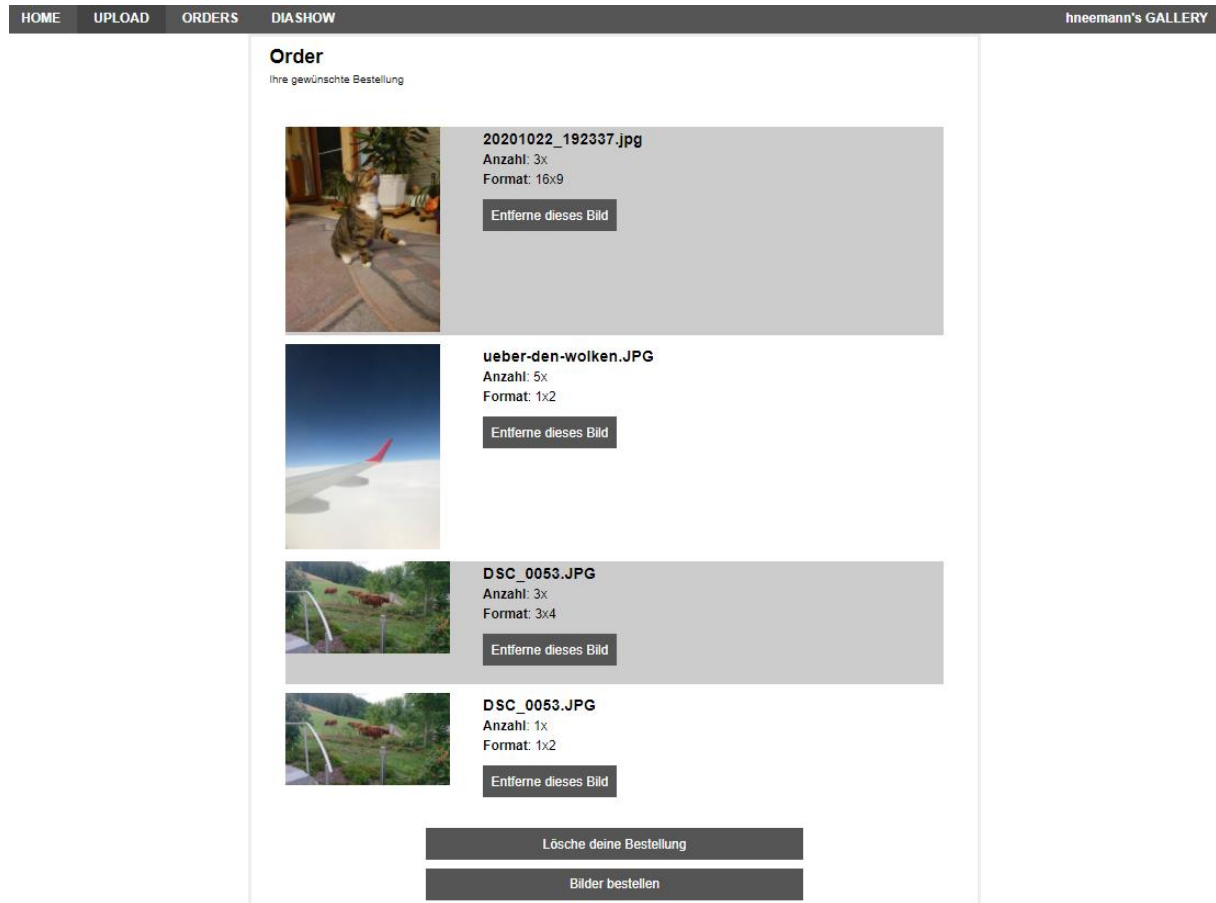


Abbildung 4 Bestellübersicht

2.5 Upload

Hier kann der Nutzer seine Bilder hochladen (siehe Abbildung 5: Upload). Die Bilder können nur einzeln hochgeladen werden. Diese erscheinen dann in der Persönlichen Gallery.

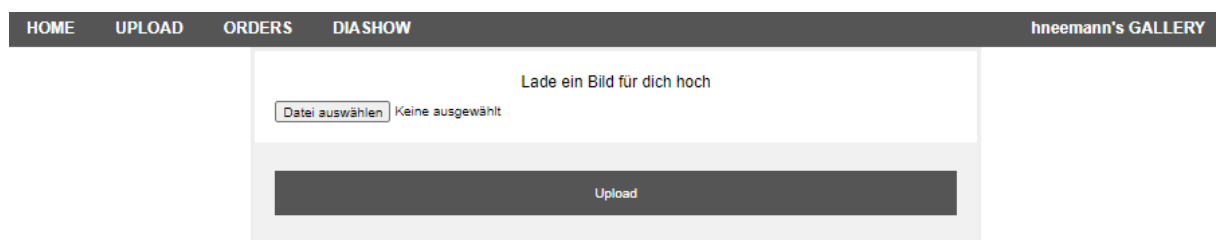


Abbildung 5 Upload

2.6 Diashow (Feature)

Die Zusatzfunktion Diashow kann über die Navigationsleiste erreicht werden. In dem Bereich „Diashow“ werden die letzten fünf hochgeladenen Bilder als Diashow angezeigt. Da die Zusatzfunktion Fotogruppen nicht implementiert wurde, wurde sich wenigstens dafür entschieden eine abgeschwächte Version der Diashow mit hinzuzufügen.



Schaue dir deine letzten fünf Bilder an



3 Dokumentation des Betriebs - PhotoServer

3.1 Umgebung






Der Server kann unter Windows, Linux und OSX zum Laufen gebracht werden. Zudem kann der Server lokal unter per HTTPS unter „https://localhost:4443“ erreicht werden. Des Weiteren werden alle gängigen Browser unterstützt. Dazu zählen der Firefox, Google Chrome, Opera und Safari.

3.2 Parameter

Dem Server können Parameter mitgegeben werden, um ihn zu konfigurieren. Hierbei gibt es einmal den Parameter -port, mit welchem der HTTPS Port eingestellt werden kann. Zudem gibt es den Parameter -certificates, mit welchem man das Verzeichnis festlegen kann, in welchem sich die für HTTPS benötigten Zertifikate cert.pem und key.pem befinden. Diese können von Let's Encrypt kommen oder auch einfach selbst erstellt werden mithilfe des Programmes OpenSSL.

3.3 Dateien

Der Server hat in seinem Stammverzeichnis, also von dort aus wo der Server gestartet wird, einen static-Ordner, in welchem sich einige für den Server benötigte und angelegte Dateien befinden.

 data	10.01.2021 16:34	Dateiordner
 images	10.01.2021 16:30	Dateiordner
 orders	10.01.2021 14:27	Dateiordner
 ssl	05.01.2021 00:36	Dateiordner
 template	10.01.2021 18:22	Dateiordner

Im data-Ordner liegen die json-Dateien für die Kommentare, die Users, die Orders und die Fotos. In der Datei users.json stehen die Nutzernamen und die mit dem Salt gehashten Passwörter.

Im images-Order liegen sämtliche Bilder, welche hochgeladen werden. Dies ist auch der Ordner, der als FileServer dient und die Bilder zur Verfügung stellt. Dabei sind die Bilder als hash_des_bildes.jpg abgespeichert, um auch Duplikate erkennen zu können.

Im orders-Ordner liegen die Dateien, welche bei einer Bestellung als Order erstellt werden. Hier landen dann die Daten mit dem Namen username.zip, welche sich dann abgeholt werden können.

Im ssl-Ordner liegen die Dateien cert.pem und key.pem. Diese werden für die SSL-Verschlüsselung benötigt. Dieser Ordner enthält die Beispieldateien und ist der Standardwert für den Pfad zu den Dateien, sofern kein anderer per Parameter übergeben wurde.

Zum Schluss gibt es noch den template-Ordner, welche die html-Template-Dateien für die Go-Template-Engine enthält. Diese sollte nicht verändert werden, da diese den Code (Mustache-Schreibweise) enthalten und die übergeben Daten enthalten und auswerten. Man könnte jedoch das Design (CSS) ändern.

4 Dokumentation des Betriebs – Batchupload

4.1 Umgebung

Der Batchupload kann einfach per Kommandozeile gestartet werden und mit ihm kann ein ganzer Ordner Bilder auf einmal auf den Useraccount auf dem Photoserver hochgeladen werden.

4.2 Parameter

Auch der Batchupload hat einige Parameter, welche übergeben werden könne oder müssen. Mit -data kann der Ordner spezifiziert werden, welcher hochgeladen werden soll. Zudem muss es auf dem Photo-Server einen Account geben, wobei man diese Daten zum Authentifizieren mitgeben muss. Dies geschieht mit den Parametern -username und -password.

Beispiel: User: admin, Passwort: 123456, Ordner mit Bilder: ...Desktop\Ordner

Mithilfe des folgenden Commands kann der ganze Inhalt des Ordner auf den Photo-Server hochgeladen werden.

```
go run cmd/batchupload/main.go -data=C:\Users\User\Desktop\Ordner -  
username=admin -password=123456
```

5 Kurzbeschreibung der Beiträge zur Projektumsetzung

In diesem Kapitel werden die Einzelleistungen der jeweiligen Gruppenmitgliedern beschrieben.

5.1 Rudolf Gaab

Ich für meinen Teil habe mich hauptsächlich um die Gestaltung und Einbindung der Templates in Go gekümmert. Die Templates sind im HTML (Hypertext Markup Language) Format gehalten, so ist auch der Aufbau. Die einzelnen HTML Komponenten wurde mithilfe von CSS (Cascading Style Sheets) gestaltet. Zu den erstellten Templates gehören zum Beispiel die Seite für die Bilder mit den Kommentaren, die Slideshow und einige weitere. Zudem habe ich mich teilweise um die Mustache-Schreibweise gekümmert und die Handler im Code erstellt sowie diese einem Endpoint (z.B. /diashow) zugeordnet. Zudem habe ich noch einen großen Teil der Dokumentation geschrieben.

5.2 Johannes Weis

Bei diesem Projekt habe ich mich hauptsächlich um die Tools und zusammen mit Rudolf um die Handler gekümmert. Zunächst kümmerte ich mich hierbei um das Registrieren und Einloggen eines Users. Weitere Tätigkeiten waren hierbei das Auslesen der Datetime im ExifHeader oder auch das Hochladen von Dateien welches über eine REST-Schnittstelle ermöglicht wurde. Auch das Zippen von den Dateien gehörte zu meinem Aufgabengebiet.

Zum Schluss habe ich noch habe ich noch einige Tests geschrieben und den Code kommentiert. Letztlich habe ich noch eine Kleinigkeit an der Dokumentation ergänzt.

5.3 Yannis Luithle

Ich habe mich hauptsächlich um die im Projekt verwendeten Objekte und deren Speicherung und Verwaltung gekümmert. Dazu zählen User, Photo, Comment und Order. Beim User ging es darum, notwendige Funktionen zu implementieren, welche später verwendet werden können. So sollte beispielsweise alles durch „salting“ sicherer gemacht werden. Dabei ging es vor allem um den Aufbau der Objekte und das Laden und Speichern sowie einige Filterfunktionen. Neben den Objekten habe ich noch für ein paar Endpunkte wie zum Beispiel /orders oder /image die die Handlerlogik geschrieben und die Templates erstellt, damit diese Funktionieren. Um die Gestaltung habe ich mich nicht gekümmert. Ansonsten wurden überall noch weitere benötigte

Funktionen implementiert (z.B. in den packageTools). Am Ende habe ich noch einigen Code kommentiert und Tests geschrieben sowie ein wenig in der Dokumentation geschrieben.

6 Quellen

- Skript Programmieren 2 Go von Prof. Dr. Neemann