Strava-Light GO-Projekt

Programmieren - DHBW Mosbach
5422223
9872387
8190324

1 Architekturdokumentation

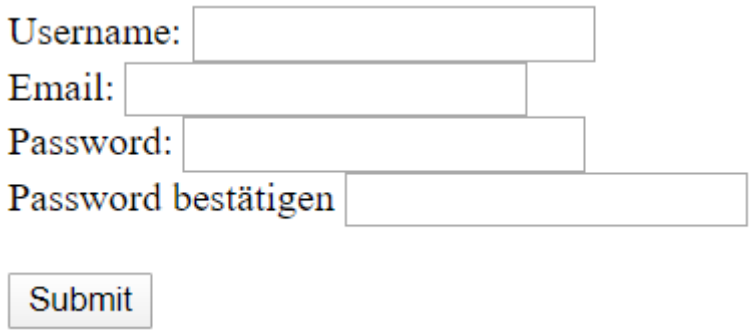
Aufgebaut ist das Projekt aus 3 Verzeichnissen. Dem Verzeichnis Backend, das das Unterverzeichnis „main“ enthält, in welchem die Go-Klassen und die Test Klassen gespeichert sind. Dort liegt auch die „go.mod“ und das „main“ Verzeichnis muss „gebuildet“ werden, um das Projekt zu starten. Ein weiteres Verzeichnis ist das Frontend, in dem alle HTML Dateien gespeichert sind, auf die der Nutzer geleitet werden kann. Hierbei gibt es zusätzlich noch Templates, in welches dynamisch durch Daten vom Backend erweitert werden kann. Beim Verzeichnis „DataStorage“ handelt es sich um einen zentralen Speicher, aller im Projekt benötigten Daten. Dort werden sowohl die Nutzerdaten, die Aktivitätsdaten, aber auch die hochgeladenen GPX Dateien aller Nutzer gespeichert, damit ein Nutzer sie zu einem späteren Zeitpunkt wieder runterladen kann. In der Datei Webrequests werden alle Anfragen von den Nutzern bearbeitet. Hier stehen die verschiedenen Handler bereit und leiten die Anfrage an weitere Klassen weiter. In der Datei RegisterAndSessionHandler wird alles rund um den Nutzer und valide Sitzungen abgearbeitet. Für einen permanenten Speicher wird die UserDataDB.csv benutzt. Inder Datei UploadFileHandler wird die Funktion vom hochladen von Aktivitäten definiert und die Speicherung berechneter Streckendaten in die ActivityDB.csv. Die Berechnung der Streckendaten erfolgt in der gpx.go Datei. Für die Anzeige von Daten für einen jeweiligen Nutzer, Suche, Bearbeiten und Löschen stehen in der Dashboard.go die benötigten Funktionen bereit. In der main Datei werden die Flags/Startparameter ausgewertet, falls gesetzt. Ansonsten sind dort die Standardwerte hinterlegt.

2 Anwenderdokumentation / Dokumentation des Betriebes

2.1 Registrieren

Die Registrierungsseite kann mit dem Parameter /Register.html erreicht werden. Hierbei wird mithilfe einer Email, Benutzername und Passwort registriert. Der Benutzername und die Email Adresse dürfen nicht schon in der Datenbank

vorhanden sein, sondern müssen einzigartig sein. Die beiden Passwortfelder müssen übereinstimmen, und werden vom Backend geprüft. Zudem muss das ausgewählte Passwort mindestens acht Buchstaben lang sein. Das Passwort wird hier mit einem Salt mit der Hashfunktion Sha512 in der UserDataDB.csv abgespeichert. Die Daten Nutzername, Email und zusätzlich der von Backend vergebenen UserID werden dort auch gespeichert. Die UserID ist eine Zahl größer als 0 und bei jeder Registrierung wird eine neue Zahl, die größer als die aktuell höchste UserID gewählt. Nach erfolgreicher Registrierung wird auf die Dashboardseite weitergeleitet.



2.2 Einloggen

Die Anmeldeseite kann mit dem Parameter /Login.html erreicht werden.

Bei einem falschen Passwort oder einem nicht vorhandenem Benutzer kommt eine Fehlermeldung, wie im Folgenden zu sehen:

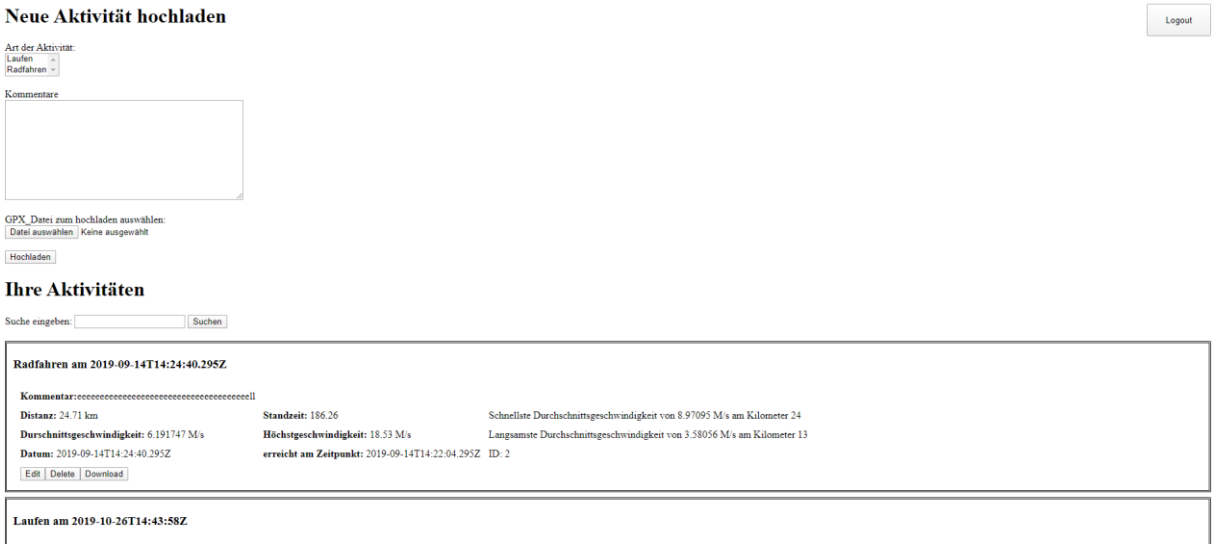
Durch eine valide Anmeldung generiert das Backend einen zufälligen Sitzungsschlüssel (Sessionkey), welcher mit einer Zuordnung zum Nutzer im Backend in einer Map zwischengespeichert wird.Der Sitzungschlüssel ist hierbei nur

eine gewisse Zeit gültig. Diese Information ist auch in der Map im Backend vermerkt. Der Sitzungschlüssel wird dem Nutzer anschließend als Cookie auth gesetzt.

Dieser Cookie wird nun bei jeder Anfrage des Nutzers mitgeschickt und überprüft, ob der Nutzer berechtigt ist und für welchen Nutzer gerade Daten geladen werden sollen. (über verknüpfte UserID). Der Cookie wird beim Ausloggen (Logout Button auf Landing Page) gelöscht.

2.3 Landing Page /Aktivitäten laden

Beim Aufrufen der Landing Page für eingeloggte Benutzer wird die Funktion viewDashboardHandler() aufgerufen, welche zuerst überprüft, ob die aktuelle Session überhaupt noch gültig ist. Ist die gültig wird die Funktion getDataForUser(), der auch die zugehörige UserID über die Funktion getUID() übermittelt wird. Über die Funktion readAcivityDB() wird die CSV Datei mithilfe eines Scanners Zeile für Zeile durchsucht, an den Kommas geteilt und die jeweiligen Werte in eine Map mit dem Namen activityMap geladen. Jede Zeile der Map wird an der Stelle der UserID auf die übergebene UserID des aktuellen Nutzers überprüft und bei Übereinstimmung in eine seperate Map mit dem Namen activityMapForUser geladen. Dies bedeutet das in dieser Map nur die Aktivitäten des Nutzers stehen, der den Aufruf ausführt. Die Map wird über Execute() im viewDashboardHandler() mit dem Responsewriter für das Frontend verbunden. Wenn sich der Nutzer also neu registriert hat oder sich einloggt, ladet er auf der Landing Page, auf der der Nutzer seine hochgeladenen Aktivitäten sieht. Das ganze sieht wie folgt aus:



2.4 Aktivitäten hochladen

Auf der Landing Page hat der Nutzer ebenfalls die Möglichkeit eine neue Aktivität hochladen. Über eine HTML Form kann der Nutzer die Aktivitätsart ein Kommentar auswählen und die zu auswertende GPX Datei hochladen. Wenn der Nutzer mehrere GPX Dateien hat, kann er diese auch in einer Zip Datei verpacken und diese hochladen. Beim drücken des Buttons „Hochladen“ wird über den Webservice die Funktion uploadHandler() aufgerufen und die in der Form definierten Werte übergeben. Als erstes wird in dieser Funktion überprüft ob der Sessionkey des Nutzers noch gültig ist oder ob dieser sich erneut anmelden muss. Wenn es sich bei der hochgeladenen Datei um eine Zip Datei handelt, wird diese erst extrahiert. Wenn es sich bei der hochgeladenen Datei oder bei einer der extrahierten Files um eine GPX File handelt wird der Inhalt in Bytes gepackt und in eine Temporäre Datei geladen, dessen Datei Name mitgespeichert wird. Anschließend folgt die Auswertung der Datei. Die ausgewerteten Daten der GPX Datei werden in eine Map geladen und anschließend mit allen Informationen in die CSV Datei ActivityDB geladen. Danach wird wieder der viewDashboardHandler() aufgerufen, der die Aktivitäten eines Benutzer an das Frontend übermittelt und alle Aktivitäten eines Nutzer auf der Landing Page anzeigt

2.5 Aktivitäten editieren

Beim drücken des Editieren Buttons wird der editHandler() aufgerufen, dem ebenfalls die zu bearbeitende AcivityID übergeben wird. Ebenfalls wie beim Löschen wird jede Zeile die nicht mit der AcivityID übereinstimmt in eine neue CSV Datei kopiert. Die Zeile, die mit der AcivityID übereinstimmt, wird in bearbeiteter Version der neu erstellten CSV Datei angehängt. Anschließend wird die erstellte CSV Datei in die Hauptdatei umbenannt und die alte Hauptdatei als Backup verwendet.

2.6 Aktivitäten löschen

Beim drücken des Löschen Buttons wird der removeHandler() aufgerufen, dem eine AcivityID übergeben wird. Nach dieser ActivityID wird die vorher aktualisierte Map mit allen Aktivitäten durchsucht und alle Zeilen, die diese ID nicht erhalten in eine neue CSV Datei geschrieben. Diese neue CSV Datei wird anschließend zur ActivityDB CSV Datei benannt und die alte zu einer Backup Datei.

2.7 Aktivitäten downloaden

Mit dem Klick auf den Download-Button bei der jeweiligen Aktivität, wird die dazugehörige GPX-Datei heruntergeladen. Der dazugehörige Handler ist der “downloadHandler”, der die activityID ausliest und dann die activitymap wieder füllt in dem mit readAcivityDB() die AcitvityDB ausgelesen wird. Dann wird mithilfe einer For-Schleife die aktuelle acitvityMap nach der zu downloadenden Datei durchsucht. Und der Filename/filepfad ausgelesen. Dann wird die file in Bytes zur Übermittlung ans Frontend gepackt.

2.8 Aktivität durch Kommentar suchen

Man kann nach Kommentaren auf “home” suchen. In dem in das Suchfeld hinter “Suche eingeben:” etwas eingegeben wird und dann der Button “Suchen” bestätigt wird, wird der searchCommentHandler aufgerufen. In der Funktion searchCommentHandler wird der Suchstring in SearchString gespeichert und dieser mit der UserID der Funktion search übergeben. Die search-Funktion enthält eine CommentaryMap. Mit readActivityDB() werden alle Aktivitäten in einer map “acivitymap” geladen. Anschließend wird über diese activitymap mithilfe einer for-schleife gegangen und immer wenn die UserID mit der übermittelten UserID übereinstimmt und der dazugehörige Kommentar den übergebenen Suchstring enthält, wird die Aktivität in einer neuen Map “commentaryMap” zwischengespeichert. Anschließend wird diese commentaryMap zurückgegeben und an das Frontend übermittelt.

3 Dokumentation des Betriebs

Zur Inbetriebnahme kann die mitgelieferte Strava-Light.exe ausgeführt werden.

Zudem können die Parameter

- Länge des SessionKeys (-sesionKeyLen)
- Länge des Salts (-saltLen)
- Port der Anwendung (-port)

beim Start der Anwendung in der Konsole mitangegeben werden.

4 Kurze Beschreibung eigener Teil

4.1 Eigener Teil 5422223

Bei dem GO-Projekt war ich hauptsächlich für die Auswertung der GPX Dateien und die Funktionalitäten der Aktivitäten zuständig. Als Einstieg habe ich mich der Aufgabe gewidmet, die Struktur einer GPX Datei zu verstehen und dadurch an die benötigten Inhalte der Datei zu kommen, um sie anschließend auszuwerten zu können. Danach habe ich die komplette Aufgabe 6 (Auswertung der Aktivität nach dem Upload) in Go-Code geschrieben und das Hochladen über das Frontend eingebunden. Für die Berechnung der Distanz aus zwei GPS Punkten habe ich die Formel in Code geschrieben. Die Geschwindigkeit wird berechnet indem die Distanz durch die dafür benötigte Zeit berechnet wird. Die benötigte Zeit bekommt man, indem man die beiden Zeitstempel der zwei Points in ein Datumsformat konvertiert und anschließend subtrahiert. Zur Auswertung habe ich noch die Aufgabe 12 (Abschnittsmaxima und -minima) dazu ergänzt. Beim Upload habe ich im Backend die dazugehörige Funktion mit der Unzip Funktion geschrieben. Die Ergebnisse der Auswertung und die Übergaben vom Frontend habe ich in einer Map mit benötigten Werten gespeichert. Nach erfolgreichem Auswerten wird die Map ich in der Funktion AppendtoDBACT() in eine CSV Datei gespeichert. Ebenfalls zuständig war ich für das Auslesen der CSV Datei mit den Aktivitäten, wenn beispielsweise die Seite aufgerufen wird und nur die Aktivitäten eines Nutzers geladen werden sollen. Zunächst werden alle Aktivitäten der CSV in eine Map geladen und über die bekomme UserID nur die Aktivitäten selektiert, die übereinstimmen. Anschließend werden diese dem Frontend übergeben. Zusätzlich war ich noch für das Editieren und Löschen einer Aktivität im Backend zuständig. Dafür habe ich die beiden Handler geschrieben und die jeweils neue Erstellung einer temporären Datei, die die AktivityDB durch Umbenennen ersetzt.

4.2 Eigener Teil 9872387

Beim Strava-Light-Projekt war ich für die Volltextsuche verantwortlich. Bei der Suche handelt es sich “nur” um 100% genaue Übereinstimmungen. Im Suchfeld bei “Suche eingeben:” wird der gesuchte String eingegeben und mithilfe des Buttons “Suchen” alle dazugehörigen Kommentare die diesen String beinhalten gesucht und angezeigt. Beim Klicken auf “Suchen” wird der searchCommentHandler aufgerufen, der dann die neue CommentaryMap zurückgibt, die nur die Aktivitäten liefert, bei den ein Teil

des Kommentars mit der Suche exakt übereinstimmen. Außerdem habe ich folgende HTML-Seiten im Frontend geschrieben. Dazu gehören u.a. die “landing.html”, “login.html”, “register.html” und “index.html”. Außerdem habe ich ein paar Tests in der gpx_test.go geschrieben. Ich habe mich auch um den Download der GPX-Files gekümmert.

4.3 Eigener Teil 8190324

Zu Beginn habe ich mich mit der Login- und Registrierungslogik beschäftigt. Hierfür wurde die Datei RegisterAndSessionHandler.go geschrieben. Hierbei geht es um die Registrierung mit validen Daten und eine Speicherung dieser Anmeldeinformationen in einer CSV Datei. Bei Passwörtern habe ich mithilfe eines Zufallsgenerators einen zufälligen Salt gebildet und das Passwort damit gehasht und abgespeichert. Beim Login werden diese Daten wiederum gelesen und mit den Anmeldedaten abgeglichen. Anschließend habe ich die Anmeldefunktion mithilfe eines Sitzungsschlüssels realisiert. Dieser Sitzungsschlüssel wird dem Nutzer als Cookie gespeichert, damit er sich nicht jedes Mal neu anmelden muss. In der WebRequests.go habe ich mich um die Handler und den allgemeinen Aufruf und Aufbau der verschiedenen Handler und Status Codes gekümmert. Zusätzlich habe ich mich um den Aufruf mit HTTPS und dem nötigen Zertifikat gekümmert. Auch die Angabe von Startparametern in der main.go habe ich implementiert. Die Tests für die Webrequests, RegisterAndSessionHandler und Dashboard Funktionen wurden von mir geschrieben. Des Weiteren habe ich mich um das Problem gekümmert, dass beim Testen die Pfade nicht mehr gestimmt haben (mit dem Workaround ../../). Außerdem habe ich mich noch um die Anfragen vom Frontend ans Backend gekümmert. Damit die Anfrage die richtigen Daten mitsendet, habe ich eine Lösung mit jQuery bzw. Javascript implementiert. Dieser Code wird in den Templates mitgeschickt. Bei Fehlermeldungen in der Anmeldung oder Registeriung habe ich dafür gesorgt, dass der Fehler mithilfe des Templates beim Nutzer angezeigt wird.