

# Architekturdokumentation

Die Architektur des Programms ist in Backend, Frontend und Resources unterteilt.

Der Ordner Frontend enthält dabei 3 Unterordner. Jeder Unterordner übernimmt dabei einen gewissen Funktionsumfang:

- Im Ordner `parameter` befindet sich das gleichnamige Package, welches dafür sorgt, dass Parameter aus dem Request ausgelesen werden können (der Benutzername wird aus dem "Context" ermittelt und ggf. eine ID aus der URL). Dieses Package wird von allen Handlern aus dem Ordner `views` verwendet.
- Im Ordner `templates` finden sich sämtliche statische Dateien, die der Webserver verwendet (HTML, CSS und JavaScript). Zusätzlich gibt es noch ein Unterordner `page`, dieser dient als Factory und erzeugt statische Daten für die verschiedenen Ansichten des Webdienstes (z.B.: Title, Überschrift, CSS- und JavaScript Dateipfad).
- Im Ordner `views` sind alle verfügbaren Handler der Anwendung implementiert. Jede Funktion (aus der Anforderung) wurde dabei in einem eigenen Handler realisiert. Alle Handler die ein Template verwenden, benutzen dabei das Package `page`. Die Daten aus diesem Package, sowie die `layout.html` stellen dabei das Grundgerüst der Seite dar. Jede Ansicht kann dann ihr eigenes Template in die "Content-Area" des Gerüsts einsetzen. Mit Hilfe des Package `parameter` kann dann für jeden Handler der Benutzername und gegebenenfalls die Aktivitäts-ID ermittelt werden.

Alle Handler sind im WebServer mit ihren Routen registriert. Statische Dateien, wie Bilder, CSS und JavaScript Dateien werden über einen integrierten FileServer bereitgestellt.

Der Abschnitt des Anfragen-und Dateimanagements im Backend umfasst die Ordner `activity`, `gpxProcessing` und `storagemanagement`.

Das Package `activity` beinhaltet in der Datei `activity.go` das struct `activity`, um die Aktivität eines Users im Backend als Objekt anzulegen und darzustellen.

In `activityCache.go` ist ein Cache implementiert, der beim Start der Applikation generiert wird. Dieser kann 10 Aktivitäten anhand ihrer ID speichern und verfährt nach dem Prinzip "Least Recently Used".

Der `activityHandler.go` verarbeitet Anfragen aus dem Frontend und Umwandlungen von Aktivitätsobjekten. Zu den Anfragen aus dem Frontend gehören unter anderem das Bearbeiten, Erstellen, Löschen sowie das Bereitstellen einer oder mehrerer Aktivitäten. Weiterhin findet hier die Konvertierung von Aktivitäten zu einem `ByteArray` statt (Aktivität zu JSON) vice versa. Die Datei `activityHandler.go` bedient sich an `activityCache.go` und `storageManager.go`, welcher sich unter anderem mit der Verwaltung existenter Dateien im Dateisystem befasst und im Verlauf der Dokumentation genauer beschrieben wird. Wird so beispielsweise eine Aktivität durch den User gelöscht, erteilt der `activityHandler` sowohl den Befehl zur Löschung der Aktivität im Cache, als auch der zugehörigen, gespeicherten Dateien.

Innerhalb des Packages `gpxProcessing` befinden sich alle relevanten Dateien zum Einlesen und Verarbeiten von GPX-Dateien und Zip-Dateien, welche GPX-Dateien beinhalten. Die Struktur einer GPX-Datei zur Verarbeitung im Backend wird in der Datei `gpxFile.go` definiert.

Sie wird innerhalb des `gpxReader.go` mithilfe des Packages `encoding/xml` aus der Go Standardbibliothek direkt aus einer enthaltenen GPX-Datei geparsed und als Objekt angelegt. Hierbei findet eine Unterscheidung zwischen der Eingabe einer GPX-Datei und einer ZIP-Datei statt, damit diese entsprechend verarbeitet werden können.

Die Informationen aus den GPX-Objekten werden in `gpxCalculator.go` berechnet und überprüft. Hierzu zählt die Berechnung der Höchstgeschwindigkeit in Stundenkilometern, der Durchschnittsgeschwindigkeit in Stundenkilometern, der zurückgelegten Distanz in Kilometern und der Standzeit in Sekunden. Ein aufgezeichneter Wegpunkt wird als Punkt betrachtet, in welchem der User immobil war, ab einer Geschwindigkeit von unter einem Stundenkilometer. Weiterhin wird anhand der berechneten Durchschnittsgeschwindigkeit die Sportart validiert, d.h. es wird determiniert, ob die angegebene Sportart anhand der Geschwindigkeit realistisch ist oder nicht. Eine Sportart mit der Bezeichnung "Laufen" wird bei einer Durchschnittsgeschwindigkeit von bis zu 16 Stundenkilometern als realistisch gewertet. bei einer Geschwindigkeit von über 16 Stundenkilometern wird die Sportart als "Radfahren" gewertet. Orientiert wurde der Wert anhand dessen, dass die Durchschnittsgeschwindigkeit bei Joggern bei 7-8 Stundenkilometern liegt, der Rekord bei Marathonläufern hingegen bei ca. 20 Stundenkilometern liegt. Damit die Verifizierung der Sportart selbst bei fortgeschritteneren Usern nicht fehlschlägt, wurde dieser als Grenzwert zum Radfahren gewählt.

Die Datei `storageManager.go` befasst sich mit der direkten Verarbeitung von Dateien. Darunter fallen die Verwaltung von auf dem Dateisystem gespeicherten Dateien (Lesen, Schreiben, Bearbeiten und Löschen) und das Auslesen der Ordner der vorhandenen User.

Die Abschnitte des Webservers, Usermanagements und der Sicherheit wird im Backend von den Packages `"auth"`, `"hashAndSalt"`, `"user"` und `"webserver"` umfasst.

Im Package `"webserver"` wird der Webserver erstellt. Der Webserver unterstützt nur eine HTTPS-Verbindung. Dies wurde dadurch erreicht, dass ein `"self signed"`-Zertifikat und private Key erstellt und eingebunden worden. In diesem Package werden auch die einzelnen Handler des Frontends je nach URL aufgerufen. Bevor die HandlerFunction ausgeführt werden können, wird vorher der Wrapper für die BasicAuth aufgerufen. Dies wird verwendet, um den Benutzernamen und das Passwort abzufragen. Die BasicAuth ruft eine Funktion im Package `"auth"` auf, welches für den Zugangsdaten-Check zuständig ist und ein `"bool"` zurückbekommt, um den Zugang auf die Webseite zu gewähren oder nicht. Wenn die Zugangsdaten falsch eingegeben wurden, werden die HandlerFunctions nicht aufgerufen und es wird erneut nach den Zugangsdaten gefragt. Beim Abbrechen wird der Status `"unauthorized"` zurückgeliefert. Wenn die richtigen Zugangsdaten eingegeben werden, wird die entsprechende HandlerFunction aufgerufen, allerdings wurde von BasicAuth der Username in einem neuen Kontext dem Frontend übergeben. Zusätzlich werden im Package `"webserver"` die Konfiguration festgelegt und das Erstellen des Grundgerüsts im Storage ausgelöst.

Im Package `"user"` gibt es ein Objekt, welches entsprechend den Usernamen, das Passwort, den Salt und den Pfad des User-Verzeichnisses speichert. Die Daten kommen aus der Datei `"users.txt"`, welches im Package `"resources"` liegt. Die Datei mit den verschlüsselten Passwörtern ist mit Base64 codiert. Dies wird beim Erstellen der Objekte für den Zugangsdaten-Check in ein Byte-Array umgewandelt. Diese User-Objekte werden später im Package `"auth"` verwendet. Außerdem ist in diesem Package ein Cache implementiert, um nach jedem Request die Userdaten aus der Datei auszulesen. Zusätzlich wird vom Webserver der angegebene Speicherort übergeben und für jeden User ein Directory erstellt.

Das Package `"auth"` ist rein für den Zugangsdaten-Check verantwortlich. Dieser bekommt von `"user"` die User-Objekte übergeben und vergleicht die eingegebenen Daten mit dem aus der `"user.txt"`-Datei. Um die verschlüsselten Passwörter und das Salt vergleichen zu können nutzt `"auth"` Funktionen aus dem Package `"hashAndSalt"`.

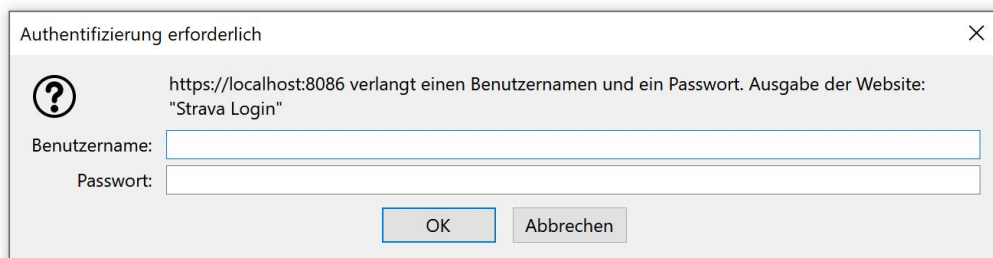
Für die Sicherheit ist das Package "hashAndSalt" zuständig. Es wird anhand des Passworts und zufällig generierten Byte-Folgen ein Salt generiert. Dieser Salt wird an das Passwort gehängt und mit SHA512 gehasht. Dies garantiert die Sicherheit, da beim Hashen das Passwort unleserlich gemacht wird und durch den Salt immer andere Werte entstehen. Beim Vergleichen wird an das angegebene Passwort vom "auth"-Package der entsprechende User-Salt angehängt und gehasht. Die beiden Passwörter, das gehashte und das aus der Datei, werden miteinander verglichen und entsprechend ein "bool" zurückgegeben.

# Anwenderdokumentation

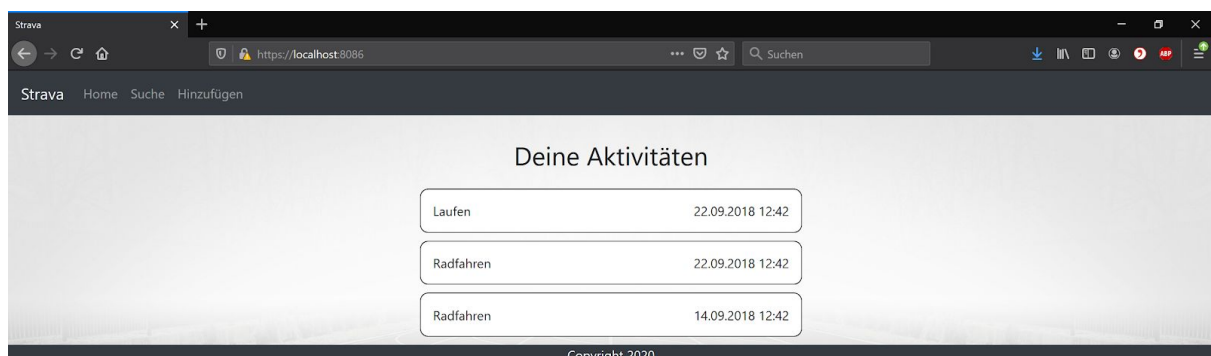
Benutzer der Webseite sind schon vorher festgelegt worden und liegen verschlüsselt mit Passwort und dazugehörigem Salt in einer Datei. Es sind insgesamt zwei User. Der Zugang geschieht über folgende Logindaten:

- Benutzername: user1, Passwort: go!Project?2020
- Benutzername: user2, Passwort: user2Password

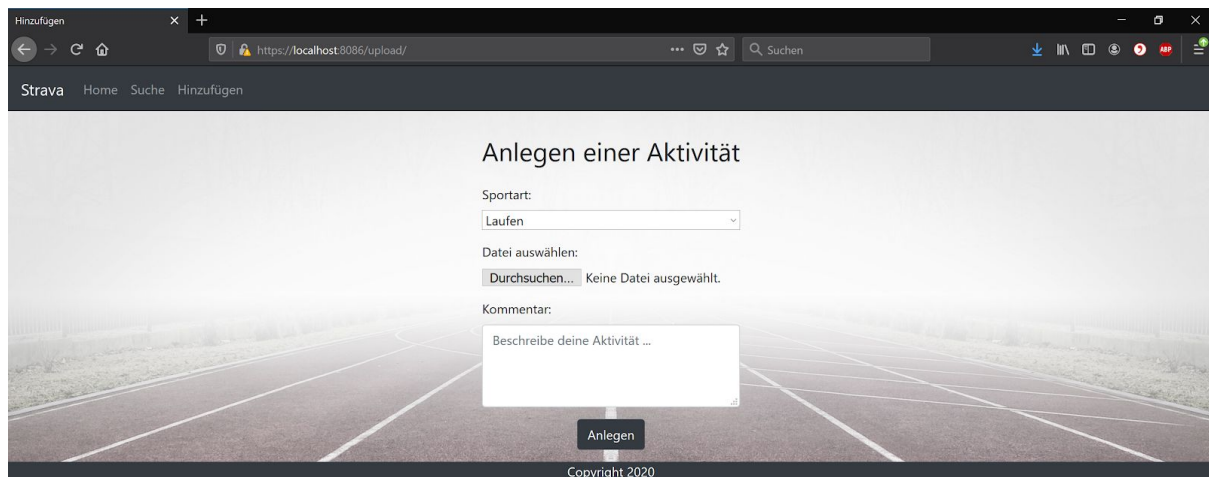
Beim Aufrufen der Webseite [https://localhost:\[port\]](https://localhost:[port]) ([port] ist hierbei der konfigurierte und angegebene Port vom Webserver, der variabel ist (sieht Betriebsdokumentation)) erscheint ein Authentifizierungsfenster, in dem die Eingabe des Benutzernamens und des Passworts möglich ist.



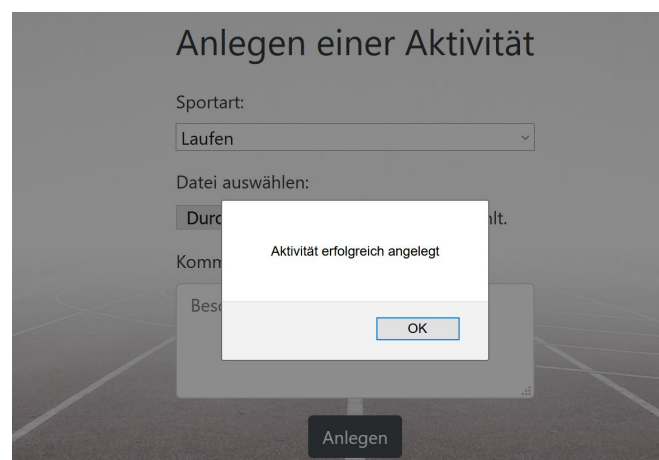
Nach dem Eingeben der richtigen Login-Daten erscheint das Home-Menü. Dort werden alle bereits hochgeladenen Aktivitäten angezeigt. Für jede Aktivität wird das Datum und die Sportart angegeben. Die Liste ist nach dem Datum der Aktivität sortiert, hierbei liegt die neueste Aktivität oben in der Übersicht. Die Aktivitätenliste kann beim erstmaligen Betrieb des Webserver leer sein.



Damit eine Aktivität angelegt werden kann, muss hierfür im Menü-Reiter “Hinzufügen” eine .gpx oder .zip Datei hochgeladen werden.

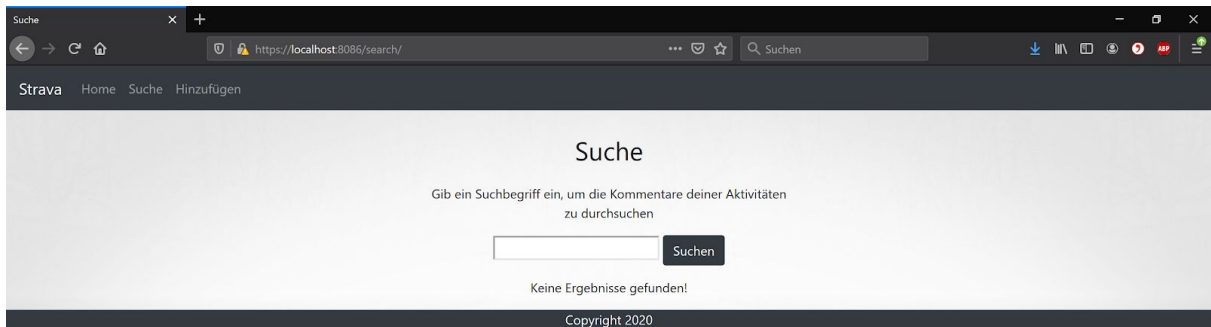


Es ist möglich beim Anlegen einer Aktivität die Sportart zu wählen und auf dem Button “Durchsuchen” eine Datei aus dem Dateisystem hochzuladen. Zu der Sportart kann zwischen “Laufen” und “Radfahren” gewählt werden. Die Anwendung unterstützt, zusätzlich zu den .gpx-Dateien, auch .zip-Dateien, welche eine .gpx-Datei enthalten. Außerdem kann auch ein Kommentar hinzugefügt werden. Beim Klicken auf den Button “Anlegen” wird die Aktivität angelegt und eine Popup-Meldung erscheint, die das nochmal bestätigt.

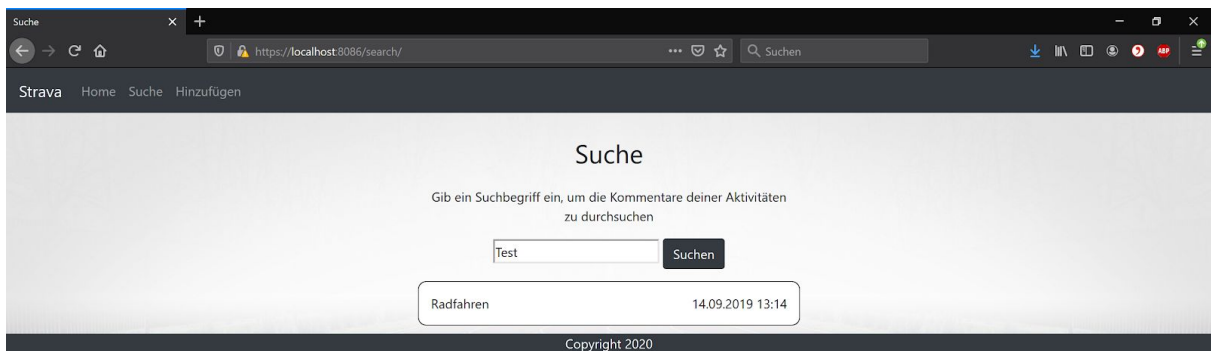


Um die angelegte Aktivität zu sehen, muss das Home-Menü über den Reiter “Home” aufgerufen werden.

Es ist möglich über den Reiter “Suche” nach Stichwörtern in den Kommentaren aller Aktivitäten zu suchen.



Nach dem Eingeben des Stichwortes werden alle Aktivitäten angezeigt, welche in den Kommentaren das angegebene Stichwort enthalten.

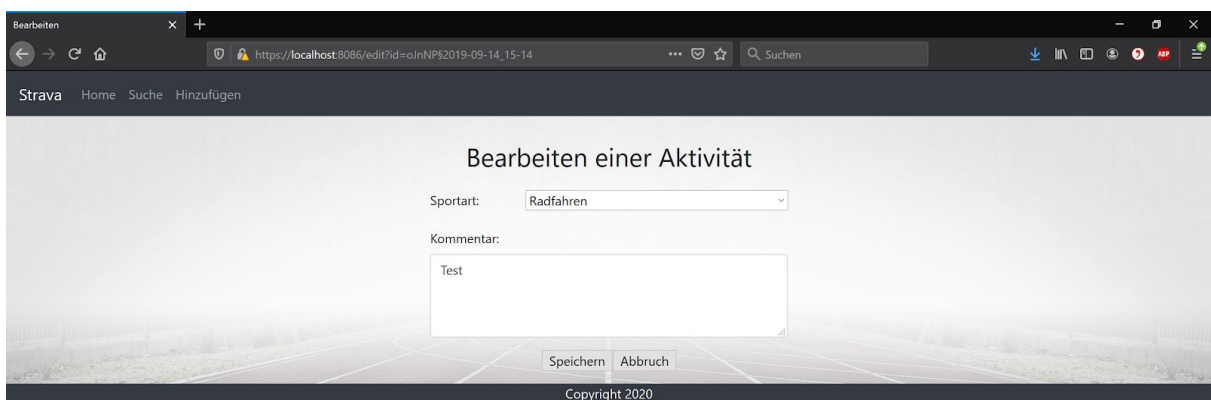



Aus dem Home-Menü kann auf eine beliebige Aktivität in der Liste geklickt werden, um nähere Informationen über diese zu erhalten. Folgende Informationen werden ausgewertet:

- Der Zeitstempel aus der Datei. Falls kein Zeitstempel vorhanden ist, wird das Upload-Datum übernommen
- Die Strecke in km (aus den GPS-Koordinaten)
- Die Durchschnittsgeschwindigkeit. In dieser Berechnung wird die Standzeit nicht miteinbezogen
- Die Maximalgeschwindigkeit
- Die Standzeit in Minuten, bei der sich nicht bewegt wurde
- Der beim Hinzufügen eingegebene Kommentar

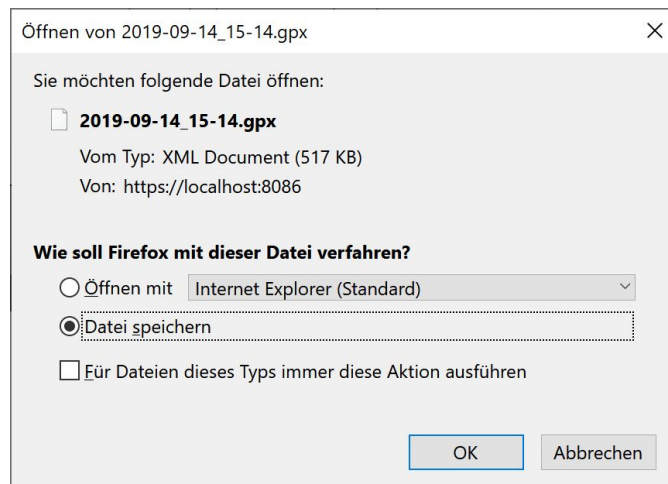



In der Detailansicht können verschiedene Aktionen ausgeführt werden. Die Aktionen sind “Bearbeiten”, “Herunterladen” und “Löschen”, die sich unter der Detailansicht befinden.




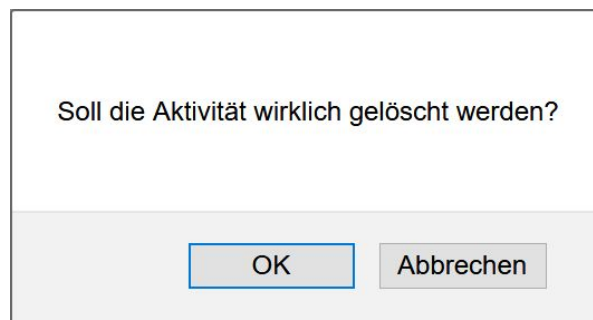
Beim Klicken auf das Symbol “Bearbeiten”  öffnet sich die Bearbeitungsansicht. Es kann hierbei die Sportart und das Kommentar geändert werden. Auf den Klick des Buttons “Speichern” wird die Änderung übernommen und es wird zur Detailansicht zurückgekehrt. Wenn jedoch “Abbruch” angeklickt wird, dann werden die Änderungen verworfen und zur Detailansicht zurückgekehrt.





Es ist möglich in der nächsten Aktion “Herunterladen”  die hochgeladene Datei wieder vom Server herunterladen zu können. Hierbei erscheint je nach verwendetem Browser eine Nachricht zum Bestätigen der Speicherung der Datei. In diesem Beispiel wurde Firefox verwendet.

In der Aktion “Löschen” mit dem Symbol  kann die angezeigte Aktivität mit samt der hochgeladenen Datei vom Server gelöscht werden. Auf den Klick des Buttons erscheint ein Bestätigungsdialog, um die Löschung bestätigen oder abbrechen zu können. Wenn dies bestätigt worden ist, wird wieder zum Home-Verzeichnis zurückgekehrt. Beim Klick auf “Abbrechen” wird die Aktivität nicht gelöscht.



# Betriebsdokumentation

Der Webserver wird über einen Aufruf aus der main.go gestartet. Darüber hinaus gibt es zusätzlich Möglichkeiten den Webserver mit Startparametern zu konfigurieren.

Der Startparameter „port“ ist für das Festlegen des Ports, auf dem der Webserver läuft, zuständig. Es sind nur Integer zulässig. Der Parameter kann auch weggelassen werden, dann startet der Webserver auf Port 443, da dies der Standard-Port für HTTPS-Verbindungen darstellt.

Für den Speicherort der GPX-Dateien und Aktivitätsinformationen ist der Startparameter „basedir“ verantwortlich. In diesem kann ein absoluter, existierender Pfad auf dem eigenen Dateisystem angegeben werden. Falls dieser Parameter nicht festgelegt wird oder der angegebene Pfad nicht existiert, ist der Speicherort für die Informationen das Homeverzeichnis in dem Ordner „storage“. Für jeden registrierten Benutzer wird dann beim Start der Anwendung ein Unterordner im Speicherpfad angelegt (sofern noch nicht vorhanden).

Auf der Konsole werden, während des Betriebs, evtl. geworfene Errors und Anmerkungen angezeigt, zusätzlich mit der Angabe des genauen Datums und der Uhrzeit.

Bei der Speicherung einer eingegebenen Datei wird diese in ihrem originalen Format (.gpx oder .zip) mit einer zugehörigen .json Datei, welche die daraus ermittelten Aktivitätsdaten enthält, in dem jeweiligen Nutzerordner platziert.

# Eigenbeiträge

## 2848869

Zu den Eigenleistungen von 2848869 gehören: Die Verarbeitung der GPX-Dateien, das Dateimanagement, die Verwaltung von Aktivitäten und die Implementierung des Activity-Caches.

Weiterhin gehört die Kommunikation mit dem Frontend zu den umfassten Tätigkeiten, d.h. es wurden benötigte Funktionen implementiert (activityhandler.go), um anhand von Eingabeparametern des Users erwartete Dateien zurückzusenden.

Umsetzung der Anforderungen:

- 4.1, 4.2
- 5.1, 5.2
- Abschnitt 6 vollständig
- 7.3, 7.4, 7.5, 7.6
- 8.1
- Abschnitt 9 vollständig
- 10.3

## 3861852

Zu den Eigenleistungen von 3861852 gehört: Das Erstellen und Designen aller Templates, sowie das Management der Server-Client Kommunikation. Für ein reaktives Erscheinungsbild des Web Dienstes, wurde neben eigenen HTML und CSS Dateien auch Bootstrap 4 eingebunden. Zusätzlich wurde ein Seitengerüst (layout.html) implementiert, welches über die Datei page.go für die jeweilige Ansicht konfiguriert werden kann. Darüber hinaus verfügt das Gerüst auch über eine Content-Area, in der das Template der entsprechenden Ansicht eingesetzt wird.

Umsetzung der Anforderungen:

- 2.2, 2.3
- 4.1, (4.2). 4.3, 4.4, 4.5
- 5.1, 5.2, 5.3

- 7.1, 7.2, 7.6
- 8.1
- 10.3

## 8089098

Zu den Aufgaben von 8089098 gehören: Das Erstellen des Webservers, das Usermanagement, den Aspekt der Sicherheit, die Konfiguration und das Storagemanagement. Zusätzlich zu den Aufgaben gehört die Kommunikation mit dem Frontend und dem Dateimanagement.

Umsetzung der Anforderungen:

- 2.1, 2.2
- Abschnitt 3 vollständig
- Abschnitt 9 vollständig
- Abschnitt 10 vollständig
- Abschnitt 11 vollständig