DHBW
Duale Hochschule
Baden-Württemberg
Mosbach

Strava-Light

Programmieren II - Programmentwurf

Prof. Dr. Helmut Neemann

13. Oktober 2019

Versionen

Datum	Stand
13.10.19	Initiale Version

Organisatorisches

- Die Abgabe des Programmentwurfs erfolgt spätestens am Freitag, den 12.01.2020 als gepackte Quellen im ZIP-Format per `pushci`.
- Die Bearbeitung der Aufgabe erfolgt in Gruppen von maximal drei Studierenden.
- Es muss eine Aufstellung der Beiträge der einzelnen Gruppenmitglieder abgegeben werden.
- Jede Gruppe fertigt eigenständig eine individuelle Lösung der Aufgabenstellung an und reicht diese wie oben angegeben ein.
- Die geforderte Funktionalität muss von Ihnen selbst implementiert werden.
- Sie können sich Anregungen aus anderen Projekten holen, allerdings muss in diesem Fall die Herkunft der Ideen bzw. von Teilen des Quellcodes in den betroffenen Quelldateien vermerkt sein.
- Werden Code-Bestandteile einer anderen Gruppe übernommen, wird die Abgabe mit 0% bewertet.

Viel Spaß und viel Erfolg bei der Bearbeitung!

Aufgabenstellung

Es soll ein System zur Verwaltung sportlicher Aktivitäten, vergleichbar mit Strava implementiert werden.

Anforderungen

1. Nicht funktional
   1.1) Die Implementierung hat in der Programmiersprache Go zu erfolgen.
   1.2) Es dürfen keine Pakete Dritter verwendet werden! Einzige Ausnahme ist ein Paket zur Vereinfachung der Tests.
   Empfohlen sei hier `github.com/stretchr/testify/assert`
   1.3) Alle Source-Dateien und die PDF-Datei müssen die Matrikelnummern aller Gruppenmitglieder enthalten.
2. Allgemein
   2.1) Die Anwendung soll mehrere Nutzer unterstützen. Die Nutzer sollen komplett voneinander getrennt sein. Es soll keinem Nutzer möglich sein, die Daten anderer Nutzer einzusehen.
   2.2) Die Anwendung soll unter Windows und Linux lauffähig sein.
   2.3) Es soll sowohl Firefox, Chrome und Edge in der jeweils aktuellen Version unterstützt werden. Diese Anforderung ist am einfachsten zu erfüllen, indem Sie auf komplexe JavaScript/CSS „spielereien“ verzichten. ;-)
3. Sicherheit
   3.1) Die Web-Seite soll nur per HTTPS erreichbar sein.
   3.2) Der Zugang für die Nutzer soll durch Benutzernamen und Passwort geschützt werden.
   3.3) Die Passwörter dürfen nicht im Klartext gespeichert werden.
   3.4) Es soll „salting“ eingesetzt werden.
   3.5) Alle Zugangsdaten sind in einer gemeinsamen Datei zu speichern.
4. Aktivität
   4.1) Eine Aktivität wird angelegt, indem eine gpx-Datei hochgeladen wird.
   4.2) Es sollen auch zip-Dateien unterstützt werden, welche eine gpx-Datei enthalten.
   4.3) Beim Upload soll die Art der Aktivität aus einer Liste ausgewählt werden können.
   4.4) Mindestens soll die Liste „Laufen“ und „Radfahren“ enthalten.
   4.5) Beim Upload soll ein Kommentar eingegeben werden können.

5. Bearbeitung der Aktivität
   5.1) Im Nachhinein soll die Art der Aktivität und der Kommentar bearbeitet werden können.
   5.2) Es soll möglich sein, eine Aktivität zu löschen.
   5.3) Vor dem Löschen soll ein Bestätigungsdialog erscheinen.
6. Auswertung der Aktivität nach dem Upload
   6.1) Aus den GPS-Koordinaten soll die Strecke in km errechnet werden.
   6.2) Wenn die GPX-Datei einen Zeitstempel enthält, soll die Durchschnittsgeschwindigkeit und die Maximalgeschwindigkeit ermittelt werden.
   6.3) Die Standzeit ist ebenfalls zu ermitteln (Gemeint ist die Zeit, für die sich der Sportler nicht bewegt hat, weil er z.B. an einer Ampel zu warten hatte).
   6.4) Bei der Berechnung der Durchschnittsgeschwindigkeit soll eine eventuelle Standzeit berücksichtigt werden. Das bedeutet, dass die Durchschnittsgeschwindigkeit sich nicht verringern soll, wenn der Läufer/Fahrer für einige Zeit warten muss.
   6.5) Ein offensichtlicher Eingabefehler bzgl. der Art der Aktivität soll anhand der Durchschnittsgeschwindigkeit automatisch korrigiert werden (z.B. ist ein 50km-Lauf mit $\overline{v} > 20$km/h sicher ein Eingabefehler.)
7. Übersicht der Aktivitäten
   7.1) Es soll eine Übersicht aller vorhandenen Aktivitäten angezeigt werden.
   7.2) Für jede Aktivität sollen das Datum der Aktivität und die bei der Auswertung bestimmten Werte angezeigt werden.
   7.3) Die Liste ist nach dem Datum der Aktivität zu sortieren, die neueste Aktivität soll oben stehen.
   7.4) Das Datum soll, wenn möglich, aus der gpx Datei extrahiert werden.
   7.5) Nur wenn kein Zeitstempel enthalten ist, soll das Upload-Datum verwendet werden.
   7.6) Es soll möglich sein die gpx/zip Datei herunter zu laden.
8. Suchfunktion
   8.1) Es soll eine Volltextsuche für die Kommentare implementiert werden.
9. Storage
   9.1) Alle Informationen sollen zusammen mit dem gpx-Dateien im Dateisystem gespeichert werden.
   9.2) Es sollen nicht alle Informationen in einer gemeinsamen Datei gespeichert werden.
   9.3) Es soll ein geeignetes Caching implementiert werden, d.h. es sollen **nicht** bei jedem Request alle Dateien neu eingelesen werden.

10. Konfiguration
    10.1) Die Konfiguration soll komplett über Startparameter erfolgen. (Siehe Package `flag`)
    10.2) Der Port muss sich über ein Flag festlegen lassen.
    10.3) „Hart kodierte“ absolute Pfade sind nicht erlaubt.
11. Betrieb
    11.1) Wird die Anwendung ohne Argumente gestartet, soll ein sinnvoller „default“ gewählt werden.
    11.2) Nicht vorhandene aber benötigte Order sollen ggfls. angelegt werden.
    11.3) Die Anwendung soll zwar HTTPS und die entsprechenden erforderlichen Zertifikate unterstützen, es kann jedoch davon ausgegangen werden, dass geeignete Zertifikate gestellt werden. Für Ihre Tests können Sie ein „self signed“ Zertifikat verwenden. Es ist nicht erforderlich zur Laufzeit Zertifikate zu erstellen o.ä.. Ebenso ist keine Let's Encrypt Anbindung erforderlich.

Optionale Anforderungen

12. Abschnittsmaxima und -minima
    12.1) Die Durchschnittsgeschwindigkeit auf dem schnellsten Kilometer und
    12.2) die Durchschnittsgeschwindigkeit auf dem langsamsten Kilometer sollen ermittelt werden.
13. Segmente
    13.1) Es sollen Segmente unterstützt werden.
    13.2) Ein Segment ist definiert durch zwei virtuelle Tore. Eines markiert die Startlinie und eines die Ziellinie des Segmentes.
    13.3) Ein Segment gilt als absolviert, wenn zunächst die Startlinie in der richtigen Richtung überquert wurde, und danach die Ziellinie in der richtigen Richtung überquert wurde.
    13.4) Für jedes absolvierte Segment sollen Strecke, Dauer und die Durchschnittsgeschwindigkeit angezeigt werden.
    13.5) Jedes definierte Segment soll einen eindeutigen Namen tragen.
    13.6) Die Segmente können über eine xml-Datei eingelesen werden, welche beim Start des Servers eingelesen wird.

WEB-Server in Go

Es gibt einige interessante Techniken, welche die Implementierung von WEB-Servern in Go unterstützen.

Der folgende sehr gelungene Vortrag sei empfohlen:

Mat Ryer: „Building APIs“
Golang UK Conference 2015
`https://youtu.be/tIm8UkSf6RA`

Eine weitere Quelle für Best-Practices findet sich in diesem Vortrag:
Peter Bourgon: „Best Practices in Production Environments“
QCon London 2016
`https://peter.bourgon.org/go-best-practices-2016/`

Abgabeumfang

- Der kompletter Source-Code incl. der Testfälle.
- Dokumentation des Source-Codes
  Die Dokumentation setzt sich aus zwei Bestandteilen zusammen: Zum einen aus der Dokumentation des Sourcecodes. Diese erfolgt am geeignetsten im Sourcecode selbst. Zum anderen aus der Dokumentation der Architektur. Diese soll geeignet sein, einem Außenstehenden einen Überblick über das Gesamtprojekt zu verschaffen, indem Fragen beantwortet werden wie: „Aus welchen Komponenten besteht das System?“ oder „Wie arbeiten die Komponenten zusammen?“. Bei Objektorientierten Projekten bietet sich z.B. ein Klassendiagramm an, um die Beziehungen zwischen den Klassen darzustellen, ist allein aber nicht ausreichend.
- Es soll **eine** PDF-Datei abgegeben werden, welche folgendes enthält:
  - Architekturdokumentation
  - Anwenderdokumentation
  - Dokumentation des Betriebs.
  - Jedes Gruppenmitglied ergänzt eine kurze Beschreibung des eigenen Beitrags zur Projektumsetzung ab (eine Seite reicht).
- Für die Bewertung werden nur die Sourcen und die eine PDF-Datei herangezogen.
- Alle Source-Dateien und die PDF-Datei müssen die Matrikelnummern aller Gruppenmitglieder enthalten.

Abgabe

Die Abgabe soll per `pushci` (`https://infprogs2.dhbw-mosbach.de`, User: `student`, PWD: `prog2inf17`) erfolgen. Hierbei handelt es sich um einen Dienst, welcher aus dem Intranet und dem Internet erreichbar ist.

Dieser Dienst übernimmt Ihre Datei, packt diese aus, überprüft einige formale Kriterien, compiliert die Sourcen und führt alle Tests aus. Eine Datei kann nur abgegeben werden, wenn alle Testfälle „grün“ sind.

Wer bereits mit einem Continuous-Integration-System gearbeitet hat, ist mit diesem Vorgehen sicher vertraut. Für Studierende, die noch nicht mit einem CI-System gearbeitet haben, ist das möglicherweise ungewohnt. Es ist daher dringend angeraten, sich mit diesem Dienst frühzeitig vertraut zu machen. Es wäre sehr ungeschickt, erst am Tag der Abgabe um 23:55 das erste mal zu versuchen, eine Datei „hochzuladen“.

Sie können beliebig oft eine Datei „hochladen“ und überprüfen lassen. Wenn diese Überprüfung erfolgreich war, kann die Datei abgegeben werden, Sie können die Abgabe jedoch auch bei erfolgreicher Prüfung noch abbrechen. Bitte geben Sie nur einmal eine finale Lösung ab, indem Sie nach der Überprüfung die Matrikelnummern der Gruppenmitglieder angeben. Nur im Ausnahmefall sollte eine Abgabe mehrfach erfolgen, wobei nur die zuletzt abgegebene Variante bewertet wird.

Bewertungskriterien

Ihr Programmentwurf wird nach folgenden Kriterien bewertet:

1. Abbildung der Anforderungen (s.o.)
2. Strukturierung und Konsistenz des Quellcodes
3. Ausreichende Testabdeckung des implementierten Codes. (gemessen mit `go test -cover`)
4. Sinnhaftigkeit der Tests (Sonderfälle, Grenzfälle, Orthogonalität usw.)
5. Qualität von Kommentaren und Dokumentation
6. Benutzerfreundlichkeit der Schnittstellen (APIs)
7. Optionale Features
8. Das Design der Web-Seite spielt keine Rolle solange sie benutzbar ist. Es ist kein aufwendiges JavaScript/CSS erforderlich!