DHBW
Duale Hochschule
Baden-Württemberg
Mosbach

Anwesenheitsliste

Programmieren II - Programmentwurf

Prof. Dr. Helmut Neemann

11. Oktober 2021

Versionen

Datum	Stand
03.10.21	Initiale Version

Organisatorisches

- Die Abgabe des Programmentwurfs erfolgt spätestens am 23.12.21 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 dieser Veranstaltung übernommen, wird die Abgabe beider Gruppen mit 0% bewertet.
- Leider ist es schon mehrfach vorgekommen, dass Gruppen den Code einer anderen Gruppe übernommen haben, welcher in einem öffentlichen git-Repository gehostet war. Daher muss ich leider dringend empfehlen, für die Zusammenarbeit innerhalb einer Gruppe ein privates Repository zu verwenden.

Viel Spaß und viel Erfolg bei der Bearbeitung!

Aufgabenstellung

Es soll eine für Besucher einfach zu benutzende Anwesenheitsliste implementiert werden. Basieren soll diese Liste auf dem Scannen von dynamischen QR-Codes. Die Funktionsweise ist dabei wie folgt:

1. Der Besucher scannt mit der Photo-App seines Handys einen QR-Code.
2. Der QR-Code enthält einen Link, welcher auf eine Anmeldeseite führt.
3. Auf der Anmeldeseite kann sich der Besucher mit Namen, und Adresse anmelden.
4. Der Link innerhalb des QR-Codes soll nur für wenige Minuten gültig sein.
5. Danach wird ein neuer QR-Code erzeugt und angezeigt.
6. Dies stellt sicher, dass kein gespeicherter Link zu einer Anmeldung genutzt werden kann, und der Besucher tatsächlich vor Ort ist.

Anforderungen

1. Nicht funktional
   - 1.1) Die Implementierung hat in der Programmiersprache Go zu erfolgen.
   - 1.2) Es dürfen keine Pakete Dritter verwendet werden! Es gibt zwei Ausnahmen: Ein Paket zur Vereinfachung der Tests. Empfohlen sei hier `github.com/stretchr/testify/assert` und ein Paket zur Erzeugung der QR-Codes. Empfohlen sei für diesen Zweck `github.com/skip2/go-qrcode`.
   - 1.3) Alle Source-Dateien und die PDF-Datei müssen die Matrikelnummern aller Gruppenmitglieder enthalten.
2. Allgemein
   - 2.1) Die Anwendung soll sowohl eine WEB-Seite zur Anzeige des QR-Codes als auch die WEB-Seiten zur An- und Abmeldung ausliefern.
   - 2.2) Die Seite mit den QR-Codes soll unter einer anderen Port-Adresse erreichbar sein als die Seiten für An- und Abmeldung.
   - 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-Seiten sollen nur per HTTPS erreichbar sein.
   - 3.2) Der Zugang für die Nutzer zur Anmeldeseite soll durch ein in der URL kodiertes Token von kurzer Gültigkeitsdauer geschützt werden.
   - 3.3) Die Seite, welche den QR-Code anzeigt, soll ohne Zugangsbeschränkung verfügbar sein.

4. Orte
   - 4.1) Es sollen verschiedene Orte unterstützt werden.
   - 4.2) Für jeden Ort soll ein eigener QR-Code erzeugt werden.
   - 4.3) Für jeden Ort gibt es eine eigene Seite, welche den Namen des Ortes und den QR-Code anzeigt.
   - 4.4) Jeder Ort hat seine eigenen Access-Tokens.
   - 4.5) Die Liste der Orte soll über eine XML-Datei vorgegeben werden.
5. QR-Code Seite
   - 5.1) Der QR-Code soll über eine WEB-Seite angezeigt werden.
   - 5.2) Der QR-Code auf der WEB-Seite soll automatisch aktualisiert werden.
   - 5.3) Die im QR-Code codierte URL soll ein sich änderndes Token enthalten.
   - 5.4) Der Token im QR-Code soll nur für eine kurze, über eine Startparameter festzulegende Zeit gültig sein.
   - 5.5) Die URL der An- und Abmeldeseiten, welche im QR-Code kodiert wird, soll sich über einen Startparameter festlegen lassen.
6. An- und Abmeldeseiten
   - 6.1) Der Zugang soll nur mit dem aktuellen und dem vorherigen Token möglich sein.
   - 6.2) Ältere Token sollen ihre Gültigkeit verlieren.
   - 6.3) Bei der Anmeldung soll Name und Adresse eingegeben werden.
   - 6.4) Bei der Abmeldung soll die Eingabe nicht noch einmal erforderlich sein.
   - 6.5) Auch bei einer erneuten Anmeldung soll die Eingabe von Name und Adresse nicht noch einmal erforderlich sein.
7. Journal
   - 7.1) Jede An- und Abmeldung soll in einem Journal dokumentiert werden.
   - 7.2) Das Journal soll eine einfache Textdatei sein.
   - 7.3) Es soll für jeden Tag eine neue Datei angelegt werden.
   - 7.4) Alle An- und Abmeldungen für alle Orte sollen in einer gemeinsamen Datei gespeichert werden, so dass auf dem Server nur eine Journal-Datei pro Tag erzeugt wird.
8. Analyse
   - 8.1) Es soll ein Kommandozeilen-Tool geben, mit welchem die Journal-Datei eines Tages gelesen werden kann.
   - 8.2) Dieses soll in der Lage sein, zu ermitteln, an welchen Orten sich eine Person an diesem Tag aufgehalten hat.
   - 8.3) Des weiteren soll dieses Tool Anwesenheitslisten für einen anzugebenden Ort

als CSV-Datei extrahieren können.

8.4) Diese CSV-Datei soll sich mit MS-Excel und LibreOffice öffnen lassen.

9. Konfiguration

9.1) Die Konfiguration soll komplett über Startparameter erfolgen. (Siehe Package `flag`)

9.2) Die beiden Ports müssen sich über Flags festlegen lassen.

9.3) Die Gültigkeitsdauer der Token muss sich über ein Flag festlegen lassen.

9.4) „Hart kodierte“ absolute Pfade sind nicht erlaubt.

10. Betrieb

10.1) Wird die Anwendung ohne Startparameter gestartet, soll ein sinnvoller „default“ gewählt werden.

10.2) Nicht vorhandene aber benötigte Order sollen ggfls. angelegt werden.

10.3) 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

11. Analyse-Tool

11.1) Es soll sich automatisch eine Liste aller möglichen Kontakte einer Person ermitteln lassen.

11.2) Diese Liste soll die Dauer jedes Kontakts enthalten.

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 komplette 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.
- Die Abgabe soll **eine** PDF-Datei beinhalten, welche mindestens die folgenden Kapitel hat:
  - 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. Der Dienst ist während des ganzen Semesters verfügbar. Wenn diese Überprüfung erfolgreich war, kann die Datei abgegeben werden, Sie können die Abgabe jedoch auch bei erfolgreicher

Prüfung einfach 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.