DHBW
Duale Hochschule
Baden-Württemberg
Mosbach

Fileserver

Programmieren II - Programmentwurf

Prof. Dr. Helmut Neemann

12. Oktober 2016

Versionen

Datum	Stand
19.09.16	Initiale Version
12.10.16	Konkretisierung Zertifikate im Anschnitt Betrieb (Seite 3); Korrektur der Nummerierung der optionalen Anforderungen; Ergänzung der URL des Abgabedienstes (Seite 5)

Organisatorisches

- Die Abgabe des Programmentwurfs erfolgt spätestens am Freitag, den 23.12.2016 als gepackte Quellen im ZIP-Format per `pushci`.
- Die Bearbeitung der Aufgabe erfolgt in Gruppen von maximal drei Studierenden.
- 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 im Kommentar Ihrer Quelldatei(en) vermerkt sein.

Viel Spaß und viel Erfolg bei der Bearbeitung!

Aufgabenstellung

Es soll ein einfacher Fileserver implementiert werden. Diese soll es ermöglichen Dateien zwischen Rechnern auszutauschen. Dazu soll es eine Möglichkeit geben Dateien „hochzuladen“. Diese soll der User danach herunterladen und auch wieder löschen können. Das System soll Mehrbenutzerfähig sein: Es soll mehrere Nutzer geben, und diese sollen gegenseitig ihre Dateien nicht sehen bzw. „herunterladen“ oder löschen können.

Anforderungen

1. Nicht funktional
   1.1) Der Code soll im Paket `de/vorlesung/projekt/[Gruppen-ID]` liegen, damit alle Lösungen parallel im Source-Tree gehalten werden können.
   1.2) Es dürfen keine Pakete Dritter verwendet werden! Einzige Ausnahme sind Pakete zur Vereinfachung der Tests und der Fehlerbehandlung. Empfohlen sei hier `github.com/stretchr/testify/assert` und `github.com/pkg/errors`.
   1.3) Alle Source-Dateien und die PDF-Datei müssen die Matrikelnummern aller Gruppenmitglieder enthalten. Die Angabe der Namen ist optional aber hilfreich.
2. Allgemein
   2.1) Die Web-Seite soll nur per HTTPS erreichbar sein.
   2.2) Der Zugang soll durch Benutzernamen und Passwort geschützt werden.
   2.3) Die Passwörter dürfen nicht im Klartext gespeichert werden.
   2.4) Es soll „salting“ eingesetzt werden.
   2.5) Alle Zugangsdaten sind in einer gemeinsamen Datei zu speichern.
   2.6) Die Anwendung soll unter Windows und Linux lauffähig sein
   2.7) Es soll sowohl IE 11 als auch 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 und ein 90er Jahre Look&Feel in Kauf nehmen. ;-)
3. WEB-Oberfläche, Anmeldung
   3.1) Die Anmeldung soll über eine Web-Seite erfolgen.
      - Zur weiteren Identifikation des Nutzers soll ein Session-ID Cookie verwendet werden.
      - Der Session-Timeout soll 15 min betragen, und über Flags einstellbar sein.

3.2) Neue Nutzer sollen selbst einen Zugang anlegen können.
   - Dazu gibt der Nutzer einen Usernamen und ein Passwort ein.
   - Existiert der Nutzer schon, soll er eine Fehlermeldung angezeigt bekommen.

3.3) Ein Nutzer soll sein Passwort ändern können.

3.4) Um einen Download auch per Script (z.B. wget) zu ermöglichen, soll es eine zusätzliche URL geben, welche den Download über eine Authentifizierung per Basic-Auth erlaubt.

4. WEB-Oberfläche, Funktion

4.1) Es soll eine Tabelle mit den verfügbaren Dateien vorhanden sein.
   - In der Tabelle sollen der Dateiname, die Dateigröße und das Dateidatum enthalten sein.
   - In der Tabelle sollen auch die Unterordner sichtbar sein.

4.2) Es soll möglich sein, Dateien „hochzuladen“

4.3) Es soll möglich sein, Dateien „herunterzuladen“

4.4) Es möglich sein, Dateien zu löschen.

4.5) Es sollen sich Unterordner anlegen lassen.

4.6) Auch in diese Unterordner sollen sich Dateien laden lassen.

4.7) Die Anwendung soll Multiuser-fähig sein:
   - Es sollen sich mehrere Nutzer gleichzeitig anmelden können.
   - Jeder Nutzer soll nur seine eigenen Dateien sehen und bearbeiten können.

5. Konfiguration

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

5.2) Der Port muss sich über ein Flag festlegen lassen.

5.3) „Hart kodierte“ Pfade sind nicht erlaubt.

6. Betrieb

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

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

6.3) 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 „self signed“ Zertifikate verwenden. Es ist nicht erforderlich zur Laufzeit Zertifikate zu erstellen o.ä..

Optionale Anforderungen

7. Zonen
   7.1) Die Überprüfung der Passwörter soll in einer eigenen Anwendung erfolgen.
   7.2) Diese soll nur über HTTPS erreichbar sein.
   7.3) An diese Anwendung soll Passwort/Username übermittelt werden, und es erfolgt dann ggfl. eine Freigabe für den Fileserver.
   7.4) Auch hier soll sich der Port über ein Flag einstellen lassen.
   7.5) Es soll möglich sein, beide Serveranwendungen zu Testzwecken auf dem selben Rechner zu betreiben.

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 incl. Klassendiagramm.
- Es soll **eine** PDF-Datei abgegeben werden, welche folgendes enthält:
  - Anwender-Dokumentation
  - 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. Die Angabe der Namen ist optional aber hilfreich.

Abgabe

Die Abgabe soll per `pushci` (`https://infprogs2.dhbw-mosbach.de`, User: `student`, PWD: `prog216`) erfolgen. Hierbei handelt es sich um einen Dienst, welcher aus dem Intranet und dem Internet erreichbar ist.

Dieser Dienst übernimmt Ihre Datei, packt sie aus, überprüft einige formale Kriterien, compiliert die Sourcen und führt alle Tests aus. Eine Datei kann nur abgegeben werden, wenn alle Tests „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 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. Dies sollten Sie natürlich nur einmal mit Ihrer finalen Lösung tun.

Bewertungskriterien

Ihr Programmentwurf wird nach folgenden Kriterien (mit abnehmender Gewichtung aufgeführt) 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!