Übungsblatt 1: WT:I, WT:II

Bis zum 25.04.2019, 23:59, sind Lösungen zu folgenden Aufgaben abzugeben:
1b, 2a, 3c,d 4a,d 5a,c 6a,b,d 7, 8.

Aufgabe 1 : Web-basierte Informationssysteme (0+1+0 Punkte)

(a) Informationssysteme sind offen, dynamisch, komplex. Erläutern Sie diesen Sachverhalt anhand eines Ihnen bekannten Web-basierten Informationssystems.

(b) Erläutern Sie den Unterschied zwischen dem Internet und dem World Wide Web (WWW).

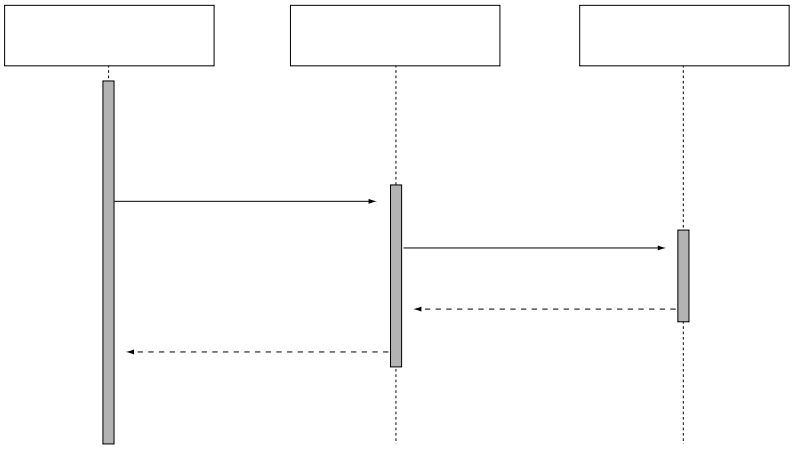
(c) Nennen Sie je ein Beispiel für interaktive, transaktionsorientierte, workflow-basierte, portalorientierte und ubiquitäre Web-basierte Systeme.

Aufgabe 2 : Modellierung Web-basierter Informationssysteme (2+0 Punkte)

Informieren Sie sich über Sequenzdiagramme in der Unified Modeling Language (UML).

(a) Eine Anwendung mit einer 3-Tier-Architektur ist die Produktsuche auf Shopping-Seiten wie amazon.de. Ein Beispiel aus dieser Anwendung ist die personalisierte Suche nach Büchern eines Autors. Erstellen Sie das zu dieser Büchersuche passende Sequenzdiagramm in dem Sie die folgenden Begriffe an passender Stelle unten eintragen:

- Ergebnisdaten
- Anforderung der Bücher eines Autors
- Amazon-Applikation: Suche, Profilanalyse
- Amazon Produktdatenbank
- Präsentation der Amazon Web-Seite
- Datenbankanfragen
- Rückgabe der aufbereiteten Ergebnisdaten



(b) Ein weiterer Diagrammtyp der UML ist das Anwendungsfalldiagramm (Use-Case-Diagramm). Modellieren Sie das Szenario „Auktionsplatz“ (Beispiel Ebay) mithilfe eines Use-Case-Diagramms. Es sollen mindestens 3 Akteure und 4 Use-Cases darin vorkommen.

Aufgabe 3 : Übertragungstechniken in Rechnernetzen (0+0+2+2+0 Punkte)

(a) Was versteht man unter Broadcasting?

(b) Nennen Sie zwei Merkmale der verbindungslosen Kommunikation.

(c) Was versteht man unter Paketvermittlung und Leitungsvermittlung?

(d) Nennen Sie jeweils einen Grund, warum für die verbindungsorientierte VoIP-Telefonie das verbindungslose UDP, und für den verbindunglosen Email-Versand via SMTP das verbindungsorientierte TCP genutzt wird.

(e) Diskutieren Sie den Zusammenhang zwischen Quality of Service (QoS) und der Netzneutralität.

Aufgabe 4 : Internet-Protokoll (1+0+0+1 Punkte)

Informieren Sie sich über das Internet-Protokoll (IP) und lösen Sie die folgenden Aufgaben.

(a) Erläutern Sie, was man im Kontext des Internet-Protokolls unter Fragmentierung versteht.

(b) Beschreiben Sie knapp den prinzipiellen Aufbau eines IP-Paketes. Quelle: RFC 791.

(c) Geben Sie mindestens zwei Beispiele für Protokolle an, die auf dem IP aufbauen.

(d) Warum wurde IPv6 eingeführt ?

Aufgabe 5 : Multiple Choice: HTTP/2 (1+0+1 Punkte)

Seit Februar 2015 ist HTTP/2, der Nachfolger des aktuell eingestzten HTTP/1.1, welches 1999 verabschiedet wurde, final. Informieren Sie sich über die neue Version des Hypertext-Transfer-Protokolls und beantworten Sie die folgenden Fragen (Geeignete Quelle: RFC 7540).

(a) Das neue HTTP/2 hat vor allem zum Ziel...
- ☐ Die Latenz beim Aufruf einer Webseite zu verringern
- ☐ Die Farbdarstellung von Bildern zu verbessern
- ☐ Die Netzwerkressourcen besser auszulasten

(b) Folgende Charakteristika weist HTTP 1.1 auf:
- ☐ Es handelt sich um ein Textprotokoll
- ☐ Es handelt sich um ein Binärprotokoll
- ☐ Es sieht eine Kompression der Header vor
- ☐ Erlaubt mehrere Requests über dieselbe TCP-Verbindung

(c) Folgende Charakteristika weist HTTP/2 auf:
- ☐ Es handelt sich um ein Textprotokoll
- ☐ Es handelt sich um ein Binärprotokoll
- ☐ Es sieht eine Kompression der Header vor
- ☐ Erlaubt mehrere Requests über dieselbe TCP-Verbindung

Beachten Sie, dass zu einer Frage mehrere Antworten zutreffen können. Eine Frage gilt als richtig beantwortet, falls alle zutreffenden und keine unzutreffende Antwort angekreuzt ist.

Aufgabe 6 : SMTP (1+2+0+1+0 Punkte)

Informieren Sie sich über das Simple Mail Transfer Protocol (SMTP). Gute Quellen hierfür sind die RFC 821 und Wikipedia.

(a) Ist SMTP, genauso wie HTTP, ein zustandsloses Protokoll? Begründen Sie Ihre Antwort.

(b) Beschreiben Sie mithilfe eines UML-Sequenzdiagramms das erfolgreiche Versenden einer Nachricht mit dem SMTP-Protokoll.

(c) Beschreiben Sie grafisch den Aufbau einer Textnachricht nach RFC 822. Orientieren Sie sich dabei an der grafischen Notation der HTTP-Request-Message.

(d) Verbinden Sie sich mittels des `telnet`-Kommandos mit `mailgate.uni-weimar.de` auf Port 25. Versenden Sie eine Mail an sich selbst mit Absender `frank.walter.steinmeier@bundespraesident.de` und Betreff „Einladung zur Verleihung des Verdienstkreuzes 1. Klasse“, in der Sie benachrichtigt werden, dass Ihnen das Verdienstkreuz erster Klasse verliehen werden soll. Dokumentieren Sie Ihr Vorgehen, indem Sie Ihre Befehle sowie die Antworten des Mailservers kopieren und als Lösung abgeben. (Hinweis: Sie können sich nur von einem Rechner innerhalb der Uni, z. B. einem Pool-Rechner, mit dem Mailserver an Port 25 verbinden.)

(e) Warum lässt sich der Mailserver der Uni Weimar nicht zum Versenden von anonymen SPAM-Nachrichten missbrauchen?

Aufgabe 7 : Java Programmierung: Netspeak API (2 Punkte)

Forken sie das Repository netspeak-requests. Schreiben Sie ein Java-Programm, dass den Schreibassistenz-Service Netspeak anfragt und die Antwort ausgibt. Die Anfrage-URL ist `http://api.netspeak.org/netspeak3/search?query=ANFRAGE` (`ANFRAGE` entsprechend ersetzen). Lassen sie den Benutzer eine Anfrage eingeben, oder übergeben sie die Anfrage als Kommandozeilenparameter. Testen sie ihr Programm mit der Anfrage *waiting for ? response* (Häufigsten Worte zwischen „waiting for“ und „response“).

Hinweis: Um die Anfrage für die Verwendung in der URL zu kodieren, kann in Java die Funktion `URLEncoder.encode(anfrage, "utf-8")` benutzt werden.

Aufgabe 8 : Programmiermodul: File Server P (4+3+0 Punkte)

Schreiben Sie ein Java-Programm, dass Inhalte eines Ordners über HTTP-GET zur Verfügung stellt.

Forken sie das Repository file-server. Die enthaltene `Server.java` implementiert einen Socket Server gemäß den Vorlesungsfolien und nutzt die Methode `handle` der `FileRequestHandler.java` zur Abhandlung von Anfragen. Diese Methode soll in dieser Aufgabe so abgeändert werden, dass die Inhalte des Ordners `www-root` und dessen Kindverzeichnisse zur Verfügung gestellt werden („Document Root“).

(a) Für jede Anfrage soll der passende Statuscode ermittelt und eine entsprechende Statuszeile erzeugt werden. Folgende Statuscodes sollen von dem Programm korrekt erkannt und behandelt werden:

- **200**: Die Anfrage ist syntaktisch und semantisch korrekt und kann verarbeitet werden.
  Beispiel: `GET / HTTP/1.1`
  Statuszeile: `HTTP/1.1 200 OK`
- **400**: Die Anfrage ist syntaktisch nicht korrekt.
  Beispiel: `GET /`
  Statuszeile: `HTTP/1.1 400 Bad Request`

- **404**: Der angefragte Pfad existiert nicht im Document Root (semantisch inkorrekt).
  Beispiel: `GET /x HTTP/1.1`
  Statuszeile: `HTTP/1.1 404 Not Found`
- **501**: Die verwendete HTTP-Methode ist im Programm nicht implementiert (alle außer GET).
  Beispiel: `HEAD / HTTP/1.1`
  Statuszeile: `HTTP/1.1 501 Not Implemented`
- **505**: Die angegebene HTTP-Version wird vom Programm nicht unterstützt (alle außer 1.1).
  Beispiel: `GET / HTTP/2`
  Statuszeile: `HTTP/1.1 505 HTTP Version Not Supported`

(b) Für Anfragen an existierende Dateien soll die jeweilige Datei übertragen werden. Folgende Headerfelder sollen gesetzt werden:

- Date: Datum und Uhrzeit der Anfrage
- Content-Type: Medientyp der Datei
- Content-Length: Größe der Datei
- Last-Modified: Letztes Änderungsdatum der Datei

Der Response Body soll aus dem Inhalt der Datei bestehen. Zu beachten ist, dass sich zwischen Header und Body *zwei* Zeilenumbrüche befinden müssen.

(c) Für Anfragen an existierende Ordner soll dessen Inhalt gelistet werden. Folgende Headerfelder sollen gesetzt werden (siehe oben): Date, Last-Modified. Der Response Body soll die Ordner und Dateien im angefragten Ordner als einfache Text-Liste enthalten.

Beispiel: `GET /index.html HTTP/1.1`

Antwort:

```
HTTP/1.1 200 OK
Date: Thu, 14 Mar 2019 10:03:50 CET
Content-Type: text/html
Content-Length: 174
Last-Modified: Fri, 12 Apr 2019 12:51:32 CEST

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Hello</title>
</head>
<body>
  <h1>Hello World!</h1>
  <a href="img/portrait.jpg">Picture</a>
</body>
</html>
```

Hinweise zur Abgabe

- Erstellen Sie *eine* PDF-Datei, in der Texte und Grafiken zu den Aufgaben enthalten sind.
- Quellcode (Java, JavaScript, PHP, HTML, CSS, XML, XSL, XSD, etc.) innerhalb des PDF-Dokuments wird nicht korrigiert. Quellcode jeder Aufgabe soll, wie im Tutorium vorgestellt, über GitLab verwaltet werden.
- Referenzieren Sie Ihre Quellcode-Dateien in dem PDF-Dokument, so dass sie einer Aufgabe eindeutig zuzuordnen sind. Verlinken sie sie entsprechendern Repositories.

- Abzugeben ist *eine* PDF-Datei die wie folgt benannt ist:
  `<Nachname>-<MatrikelNr>-webtec-blatt<Übungsblattnummer>.pdf`.
- Gruppenabgaben mit bis zu drei Personen sind erlaubt; pro Gruppe genügt dann eine Abgabe, in der alle Gruppenmitglieder mit Namen, Matrikel, und E-Mail-Adresse verzeichnet sind
- Senden Sie die PDF-Datei an Matti Wiegmann (matti.wiegmann@uni-weimar.de).