Einkaufsservice

In der derzeitigen Situation sollen ältere Personen Einkäufe durch jüngere Personen übernehmen lassen. Wir wollen für dieses Szenario eine einfache Web-basierte Applikation erstellen. Hilfesuchende, ältere Personen soll es ermöglicht werden, dass Sie eine „Einkaufsliste“ online stellen – dies soll natürlich möglichst einfach möglich sein (Überlegungen zu UX!). Angemeldete Freiwillige sollen einfach bekannt geben können, dass Sie einen gewissen Einkauf übernehmen möchten.

1 Funktionale Anforderungen

1.1 Einkaufsliste

- Besitzt Ersteller, Datum bis wann der Einkauf erledigt werden soll, Artikel die gekauft werden sollen
- Artikel besitzen Bezeichnung, Menge, sowie einen Höchstpreis der gezahlt werden soll
- Pro Einkaufsliste können beliebig viele Artikel angelegt werden.
- BenutzerInnen müssen sich anmelden, damit die Anwendung genutzt werden kann
- Für den Freiwilligen soll es möglich sein, den tatsächlich bezahlten Gesamtpreis zu hinterlegen
- Es muss gespeichert werden, welche Einkaufsliste von welchen Hilfesuchenden erstellt und von welchem/r Freiwilligen übernommen wurde
- Sobald eine Einkaufsliste von einem/r Freiwilligen übernommen wurde, wird diese Liste für andere Freiwillige ausgeblendet
- Die Einkaufsliste wird dann vom/von der Freiwilligen als erledigt markiert (alle Artikel besorgt), das UI spiegelt diesen Status für den/die Hilfesuchende wieder
- Benutzeraktionen werden in einem Log erfasst und gespeichert (IP Adresse, Aktion, Benutzername, Timestamp)

1.2 BenutzerInnen / Rollen

Es existieren folgende Benutzerrollen:

1. **HilfesuchendeR**
   Kann Einkaufslisten anlegen (1:n) und verwalten
2. **FreiwilligeR**
   Sieht alle nicht zugeordneten Listen sowie jene, die er/sie schon übernommen hat. Er/sie kann die Einkaufsliste Artikel für Artikel abarbeiten.

HINWEIS: der Registrierungsprozess für BenutzerInnen muss nicht implementiert werden. (BenutzerInnen können direkt in der Datenbank hinterlegt werden.)

2 Non-funktionale Anforderungen

2.1 Programmierung Server

Umsetzung mittels PHP >= 7.3 (OOP) – achten Sie auf einen sauberen Aufbau und eine Trennung der einzelnen Anwendungsschichten. Achten Sie auch darauf, die Features von PHP 7 (Typisierung, Operatoren) zu nutzen. Für die Anbindung der Datenquelle benutzen Sie die Bibliothek *PDO*. Sichern Sie Ihre Anwendung gegen SQL-Injection und XSS-Angriffe durch serverseitige Prüfung der Eingaben und Parameter.

Bei Benutzeraktionen werden zusätzlich Daten zur Identifizierung mitgespeichert werden (bspw. IP-Adresse der BenutzerInnen)

2.2 Datenbank

Die Datenverwaltung erfolgt mittels MySQL 5 (InnoDB). Achten Sie auf die Grundregeln des Datenbank-Designs (Datentypen, Normalisierung, sinnvolle Constraints, …). Hinweis: Datensätze, die von den BenzuerInnen gelöscht wurden, sollten nicht direkt aus der DB gelöscht werden, sondern lediglich mit einem entsprechenden Flag gekennzeichnet werden.

2.3 Client / Frontend

Die Verwendung von Javascript (auch Frameworks, aber lediglich UI Frameworks wie bspw. JQuery UI - keine SPA-Frameworks wie Angular, React oder VueJS) ist erlaubt. Die Logik muss serverseitig implementiert werden. Der HTML-Code sollte HTML5 valide sein, sämtliche Formatierungen via CSS umgesetzt werden. Die Benutzeroberfläche sollte intuitiv bedienbar sein und bei neuen BenutzerInnen der Anwendung keinerlei Erklärungsbedarf erfordern. Achten Sie insbesondere auf Validierung von Eingaben sowie die Fehlerausgabe.
Auch eine gezielte Verwendung von AJAX ist möglich, sofern diese für Sie sinnvoll erscheint (ist jedoch nicht zwingend Voraussetzung).

HINWEIS: das „Design“ sollte einfach, aber funktional sein – legen Sie Augenmerk auf intuitive Bedienbarkeit und logische Führung der BenutzerInnen durch die Website (Navigationselemente, Feedback bei Benutzeraktionen, Formular-Validierungen & -Layout, …). Achten Sie insbesondere auf eine übersichtliche Darstellung, wenn die Listen länger werden.

3 Form der Abgabe

3.1 Dokumentation

Erstellen Sie für Ihre Datenbank-Umsetzung ein ER- oder UML-Diagramm, welches die Entitäten und Beziehungen visualisiert. Für die Anwendung erstellen Sie bitte eine textuelle Beschreibung und / oder eine dokumentierte Skizze der Architektur.

3.2 Testfälle

Führen Sie ausführliche Tests Ihrer Anwendung durch und dokumentieren Sie diese. Testen Sie auch fehlerhafte Eingaben! Ihre Testfälle sollten auf jeden Fall folgende Zustände testen:

- Benutzerlogin (inkl. Fehlerüberprüfung und -behandlung) der unterschiedlichen Rollen
- Darstellung der Applikation aus Sicht der unterschiedlichen Rollen inkl. Status (offene Listen, eigene Listen, abgearbeitete Listen, bereits erledigte Artikel)
- Hilfesuchender verwaltet Listen (erstellen, ändern, löschen)
- Hinzufügen von Artikeln
- Freiwilliger übernimmt Liste und arbeitet diese ab

Erstellen Sie eine entsprechende Dokumentation und dokumentieren Sie darüber hinaus die Testfälle in Form eines PDF-Dokuments.

3.3 Anwendung

1. **Datenbank**
   SQL – Dump inkl. Testdaten und `CREATE DATABASE` statement
   Datenbankname: **`fh_2020_scm4_[IHRE_MATRIKELNUMMER]`**
   Benutzername: **`fh_2020_scm4`**
   Passwort: **`fh_2020_scm4`**
2. **Code**
   Zip-Archiv, Startdokument = index.php, muss auf Standard – XAMPP Installation lauffähig sein. Die Dokumentation speichern Sie bitte im Unterverzeichnis `/doc`.

3.4 Deadline

Laden Sie Ihre Anwendung als **eine Datei im ZIP-Format** in der entsprechenden Abgabe via Moodle hoch. Den konkreten Termin entnehmen Sie bitte dem Abgabe-Modul in Moodle.

3.5 Beurteilung

Die Beurteilung der Übung erfolgt im Rahmen einer Präsentation vor der eigenen Gruppe. Bei dieser Präsentation stellen Sie bitte Ihre Lösung vor, demonstrieren die Testfälle und stehen für technische und inhaltliche Fragen bzw. Code-Reviews zur Verfügung.

Der genaue Termin wird noch kommuniziert.