Spezifikation tessera

1.0 Einleitung

tessera ist eine Web-App zur Erstellung von Fotomosaiken. Die Bedienung erfolgt über ein Rich-Client-Web-UI.

Benutzer können sich bei tessera anmelden und digitalisierte Fotos oder sonstige Bilder in einen persönlichen Bereich der Anwendung hochladen. Diese Bilder können als Basismotiv oder als Mosaikelemente (Kacheln) verwendet werden.

tessera zerlegt das Basismotiv in Quadrate und ersetzt diese durch farblich ähnliche Kacheln. In dem so generierten Mosaikbild ist das Basismotiv gut erkennbar.

Kacheln befinden sich in pools. Benutzer können neue pools erzeugen und mit Bildern auffüllen. Pro pool wird eine bestimmte Kachelgröße festgelegt (immer quadratisch). tessera verwendet nur den zentralen quadratischen Teil der hochgeladenen Kachelbilder und skaliert dann auf die entsprechende Kachelgröße des pools.

Pools können auch mit zufällig erzeugten Bildern gefüllt werden. Ein entsprechender Generator wird von tessera bereitgestellt.

Um die Eignung der Pools beurteilen zu können, kann die Helligkeits- und Farbverteilung der Kacheln und der Basismotive grafisch dargestellt werden.

Aus der Zuordnung eines pools zu einem Basismotiv wird schließlich ein Mosaikbild erzeugt, das angezeigt und heruntergeladen werden kann.

1.1 Allgemeine Anforderungen

- tessera soll als Serveranwendung erstellt werden, die über ein Rich-Client-UI bedient wird.
- Eine einfache Benutzerverwaltung soll sicherstellen, dass nur dem jeweiligen Benutzer das Recht zur Erstellung und Änderung seiner Inhalte zugestanden wird.
- Die Serveranwendung soll alle über das Client-UI hochgeladenen Daten in einem DBMS speichern und verwalten.
- Alle erforderlichen Bildbearbeitungsprozesse der Rohdaten sollen serverseitig automatisiert ausgeführt werden.
- Die Berechnung und Erzeugung der Kachel-Pool-Bilder soll serverseitig erfolgen.
- Die Berechnung und Erzeugung der Mosaiken soll serverseitig erfolgen.
- Die Berechnung und Erzeugung der Helligkeits- und Farbverteilunggrafiken soll serverseitig erfolgen.
- Die Serveranwendung soll vollständig über das browserbasierte Rich-Client-UI bedienbar sein.

2.0 Funktionsanforderungen

2.1 Benutzerverwaltung

- Eine Benutzerverwaltung soll sicherstellen, dass nur registrierte Benutzer Zugang zu tessera erhalten.
- Neue Benutzer sollen sich durch Angabe eines Benutzernamens und eines Kennworts registrieren können.
- Für die Namens- und Kennwortfestlegung soll ein sinnvoller Zeichenvorrat und eine sinnvolle Zeichenkettenlänge definiert werden. Bei der Registrierung soll dies angezeigt und überprüft werden.
- Benutzer sollen ihr Konto mit allen damit verbundenen Dateninhalten löschen können.

2.2 User Interface

Die Benutzerschnittstelle soll folgende Funktionsbereiche abdecken:

- Anmeldung und Registrierung von Benutzern (siehe 2.1)
- Erstellung und Inspektion der Basismotive
- Erstellung und Inspektion der Pools
- Erstellung und Inspektion von Mosaiken

2.2.1 Basismotive

- Digitalisierte Bilder sollen in eine Basismotivsammlung geladen werden können.
- Es sollen mehrere benannte Sammlungen angelegt werden können.
- Ein „Motivbrowser“ soll die Bilder der Sammlungen als thumbnails anzeigen.
- Zu jedem Bild sollen die Informationen: Name, Breite/Höhe in Pixel sowie die Helligkeits- und Farbverteilung (grafisch dargestellt) einfach abrufbar sein.
- Es soll möglich sein, mit tessera skalierte Varianten von Basismotiven zu erstellen und in der Sammlung zu speichern. Die Namensgebung der Varianten soll sinnvoll automatisiert erfolgen.

2.2.2 Pools

- Pools sollen quadratische Kachelbilder gleicher Größe enthalten.
- Es sollen mehrere benannte Pools angelegt werden können.
- Pools sollen Kacheln enthalten, die entweder aus hochgeladenen Bildern oder mit einem Bildgenerator erzeugt wurden.
- Bei der Definition neuer Pools soll die Kachelgröße festgelegt werden.
- Werden Bilder in einen Pool geladen, so soll automatisch das größte Quadrat herausgeschnitten, auf die Kachelgröße des Pools skaliert und abgespeichert werden.

- Dieser *crop-and-scale* Prozess soll im „batch-Betrieb“ erfolgen können, sodass mehrere Bilder zunächst hochgeladen und dann verarbeitet werden können. Der Benutzer soll ständig über den Fortschritt dieser Verarbeitung informiert werden.
- Die Originale der Kacheln sollen nur temporär gespeichert werden.
- Es soll ein Generator zur Erstellung von Kacheln zur Verfügung stehen.
- Der Generator soll parametrisierbar sein mit der Kachelgröße, der Anzahl der Kacheln und mindestens zwei weiteren Parametern, welche die zufällige Farbgebung beeinflussen.
- Zu jedem Pool sollen die Informationen: Name, Kachelgröße in Pixel sowie die Helligkeits- und Farbverteilung (grafisch dargestellt) einfach abrufbar sein.
- Enthält der Pool generierte Bilder, sollen auch die beim Generieren verwendeten Parameter abrufbar sein.

2.2.3 Mosaikgenerator

- Der Mosaikgenerator soll aus einem gewählten Basismotiv und einem gewählten Pool ein Bildmosaik erzeugen.
- Für jeden Bildpunkt (Pixel) des Basismotivs wird eine passende Kachel gewählt und in das Ergebnisbild (Mosaik) eingesetzt.
- Der Algorithmus zur Kachelauswahl soll basieren auf dem Vergleich der mittleren Farbabstände (siehe `HinweiseZurHausarbeitWS15.pdf`).
- Folgende Optionen soll der Mosaikgenerator bieten:
  - mehrfache Verwendung von Kacheln in einem Mosaik zulassen/unterbinden
  - jede Kachel wird zufällig aus den N am besten geeigneten Kacheln gewählt (N soll dann angegeben werden können), oder es wird die am besten geeignete Kachel gewählt.
- Mosaiken sollen in einer Sammlung gespeichert werden.
- Alle bei der Generierung verwendeten Einstellungen und Optionen sollen mit dem Mosaik gespeichert werden.
- Mosaiken und die dazugehörigen Informationen sollen auch - wie bei den Motivsammlungen und Pools - als thumbnails dargestellt und mit einem „Browser“ inspizierbar sein.
- Mosaiken sollen auch in endgültiger Größe angezeigt werden können.
- Mosaiken sollen heruntergeladen werden können.

2.3 Automatische Hintergrundoperationen

- Folgende Operationen sollen automatisch in der Serveranwendung ablaufen:
  - crop-and-scale der hochgeladenen Kacheln
  - Berechnung der Helligkeits- und Farbverteilung der Pools und Basismotive
  - Berechnung/Aktualisierung der Helligkeits- und Farbverteilungsgrafiken

3.0 Implementationsanforderungen

3.1 Datenbank

- Es soll mit dem DBMS MongoDB eine Datenbank erstellt werden.
- In der DB sollen alle Zugangsdaten (Name, Kennwort) und Sitzungsinformationen gespeichert werden.
- Alle „Benutzer-Dateien“ sollen mit GridFs gespeichert werden.
- In der DB sollen alle tessera-Inhalte gespeichert werden:
  - Sammlungen der Basismotive mit generierten Varianten
  - alle Pools
  - Sammlungen der erzeugten Mosaiken
- Außerhalb der DB, im Filesystem, sollen nur Dateien der Anwendung selbst gespeichert werden (HTML, CSS, JS, Go-Quellcode, ...).

3.2 Sprachen, Werkzeuge, Techniken

- Für alle Bilddateien (hochgeladene, skalierte, generierte) soll mindestens das Format JPEG unterstützt werden.
- Clientseitig soll verwendet werden:
  - HTML5, CSS
  - JavaScript/DOM-Scripting
  - Ajax
- Die Benutzerschnittstelle soll durchgängig als Rich-Client-Anwendung in Ajax-Technik erstellt werden.
- Die Client-Anwendung soll für die aktuelle Version von Google Chrome oder Mozilla Firefox ausgelegt sein.
- Serverseitig soll verwendet werden:
  - die Sprache Go, Go-templates und die bei golang.org verfügbaren Pakete (zum `image/draw` Paket siehe: http://blog.golang.org/go-imagedraw-package)
  - MongoDB mit GridFS (mgo-API)
- Es sollen keine zusätzlichen Bibliotheken oder Frameworks verwendet werden.
- Das Layout und die Gestaltung der Client-Anwendungen wird nicht vorgegeben. Vielmehr soll die intuitive Bedienbarkeit im Vordergrund stehen.

4.0 Leistungsanforderungen

- Zum Abgabetermin soll:
  - die MongoDB-Datenbank auf borsti lauffähig installiert sein.
    Der **DB-Bezeichner** soll sein:
    HA15DB_vorname_nachname_matrnr, z.B.: **`HA15DB_donald_duck_42`**
  - die Anwendung selbst als zip-Datei hochgeladen werden (s. Teilnahmebedingungen)
    Die zip-Datei soll genau einen **Ordner** mit dem Bezeichner **`vorname_nachname_matrnr_tessera`** enthalten. Dieser Ordner enthält die gesamte Anwendung, also alle Go-Quelldateien, Template-Dateien/Ordner, statischen Inhalt, JS, CSS usw. und einen *doku*-Ordner.
    Wird der Ordner in einen **`golang/src`** Ordner kopiert, so ist die Anwendung lauffähig, sofern das IP-i/f zur borsti-MongoDB verfügbar ist. Die Anwendung soll dann mit **`http://borsti.inf.fh-flensburg.de:4242/tessera`** aufrufbar sein.
- Das abgelieferte System soll mindestens zwei pools enthalten:
  - einen mit generierten Kacheln
  - einen mit aus Fotos herausgeschnittenen und skalierten Kacheln
- Das abgelieferte System soll mindestens zwei mit dem eigenen System erzeugte Mosaiken enthalten, welche auf zwei unterschiedlichen Originalmotiven basieren.
  - ein Mosaik aus selbst generierten Kacheln (OriginalA)
  - ein Mosaik aus Foto-Kacheln (OriginalB)
  - eines dieser Mosaiken soll ein erkennbares Portrait der ProgrammautorIn sein.
- Die Zugangsdaten für das abgelieferte System sollen sein:

  Benutzername: `robert`
  Kennwort: `silvers`

5.0 Dokumentation

Die Dokumentation (Projektordner) soll insbesondere beschreiben:

- das UI-Konzept und das Layout der Anwendung
- die Datenbankstruktur
- die Go-Anwendungsstruktur (Pakete, handler, server, templates etc.)
- die Algorithmen und Datenstrukturen zur Mosaikerstellung
- die Zustandsverwaltung und die Ajax-Kommunikation
- Probleme, Besonderheiten, Bemerkenswertes
- evtl. nicht erfüllte Anforderungen

- Die Dokumentation soll aus Skizzen, Diagrammen, Bildern etc. und kurzen, verständlichen, gehaltvollen, textuellen Beschreibungen bestehen. Langatmiges Geschwafel führt zur Abwertung.
- Der Umfang der Text-Dokumentation (ohne Quellcode, Skizzen, Diagramme, Bilder) soll **10 A4 Seiten** nicht übersteigen (Richtwert 400 Wörter/Seite).
- Seiten ohne Header-Informationen (siehe Teilnahmebedingungen), loses Blattwerk, rückwärts auf dem Kopf eingeheftete und zusammen getackerte Seiten werden bei der Bewertung ignoriert.