Inhaltsverzeichnis

1 Einleitung

Javascript hat sich im Laufe der Jahre zu einer ernstzunehmenden Programmiersprache entwickelt. Die Sprache hat sich zu einer einfach zu bedienenden und gut optimierbaren Programmiersprache entwickelt und wird in diesem Zusammenhang sehr oft in Verbindung mit Webanwendungen verwendet. Hierbei ist die einfache Programmierung sowie Implementierung von sogenannten Websockets von Vorteil. Sie sind in der Lage für die parallele Kommunikation zwischen Servern und Clients zu sorgen und können somit Anwendungen im Web effizient handhaben. Im Folgenden Projekt wird eine einfache Anwendung basierend auf einem WebSocket erstellt und beschrieben. Das Handling wurde mit Hilfe von JavaScript realisiert. In der Anwendung können verschiedene Muster und Bilder in einem Canvas gezeichnet werden. Nutzer können auf der Seite mit Hilfe eines Chats kommunizieren, die Änderungen von Bildern durch andere verfolgen und Nachrichten von anderen Nutzern empfangen.

2 Implementierung

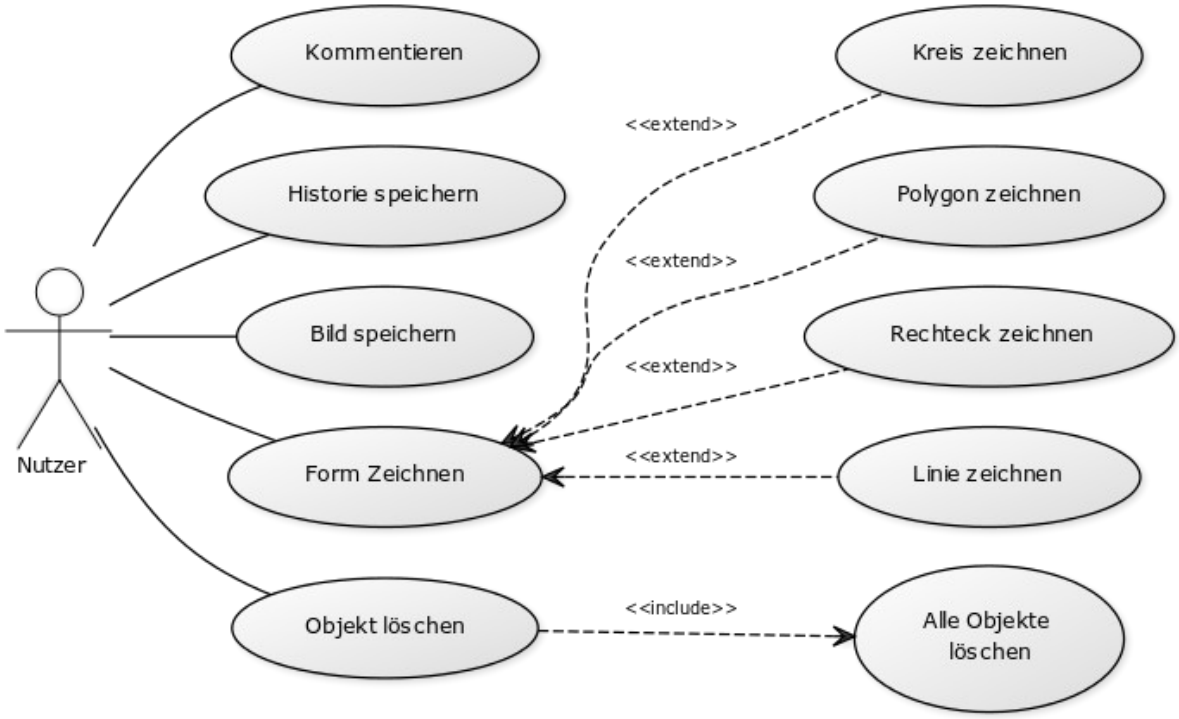
2.1 Konzept

Da keine genauere Problematik/keine genauere Aufgabe in Bezug auf die Vorlesung vorgegeben war, musste das Team sich selbstständig ein Konzept für eine Anwendung ausarbeiten. Da keine Vorerfahrung im WebSocket-Bereich vorhanden war hat sich das Team für eine einfache Single-Page Anwendung entschieden, um die Vorteile der Kommunikation via WebSocket zu nutzen und um etwas über die Möglichkeiten und Vorteile dieser zu lernen. Die Idee war hierbei eine Plattform zu entwickeln, die Formen und Bilder verarbeiten und erstellen kann. Verschiedene Nutzer sollten hierbei zusammen in der Lage sein an einem Bild zu arbeiten und über die Seite zu kommunizieren. Auch sollte eine ÄnderungsHistory und eine Möglichkeit zum Speichern eines Bildes auf der Festplatte der Nutzer ermöglicht werden.

2.2 Technische Umgebung

Für die Implementierung des Websockets wurde das von Java gelieferte Package (javax.websocket) verwendet, da für einfache Tätigkeiten das Standartpackage ausreichend war. Für das Handling der JSon-Messages wurden Packages von Codehaus (codehaus.jackson/codehaus.jettison) verwendet. Für die Datenverwaltung wurde ein JavaScript-Handler geschrieben, der sämtliche Calls verwaltet und so die Kommunikation mit dem WebSocket ermöglicht.

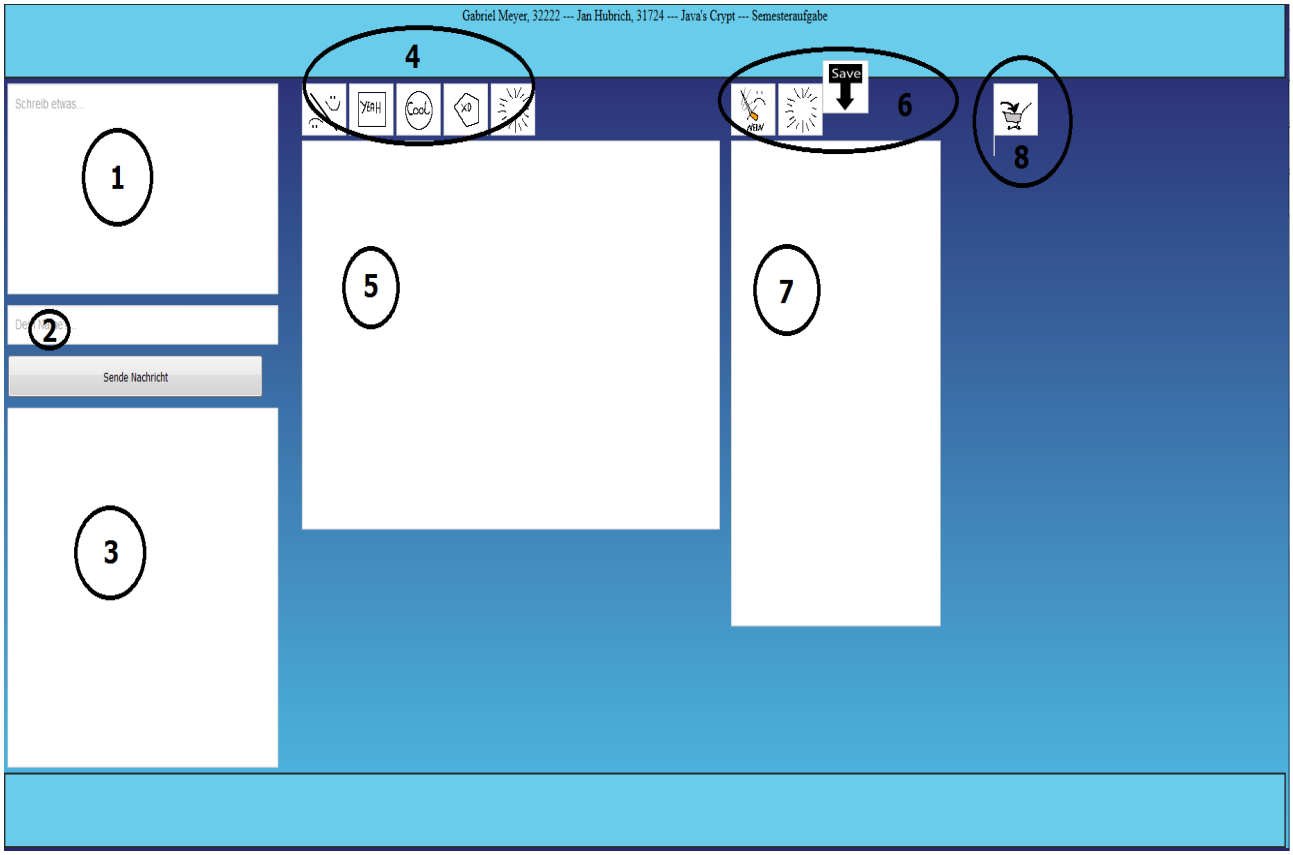
2.3 Anwendungsfälle



Anwendungsfall	Funktion
Kommentieren	Ein Kommentar wird zusammen mit dem angegebenen Namen erstellt und gesendet
History speichern	Die History wird als JSON-Objekt auf der Festplatte gespeichert

Bild speichern	Das gezeichnete Bild wird auf der Festplatte als JPG gespeichert
Form zeichnen	Zeichnet eine der gewählten Formen auf das Canvas
Objekt löschen	Löscht ein ausgewähltes Objekt aus der History
Kreis zeichnen	Zeichnet einen Kreis an die gewählte Position im Canvas
Polygon zeichnen	Zeichnet einen Kreis an die gewählte Position im Canvas
Rechteck zeichnen	Zeichnet einen Kreis an die gewählte Position im Canvas
Linie zeichnen	Zeichnet einen Kreis an die gewählte Position im Canvas
Alle Objekte löschen	Löscht alle Objekte in der History

2.4 Layout



Nummer	Beschreibung
1	Textfeld; Hier können Kommentare geschrieben werden, um mit anderen Nutzern zu kommunizieren
2	Textfeld; Hier kann der Nutzername angegeben werden
3	Anzeigefeld; Hier werden die Kommentare wie in einem Chat angezeigt
4	Buttons; Hier können verschiedene Formen zum Zeichnen ausgewählt werden, der letzte Button sorgt dafür das eben gezeichnete Objekt zu löschen
5	Canvas; Dies ist die Zeichenfläche zum Zeichnen des Bildes
6	Buttons; Hier können Objekte aus der History gelöscht, oder die History gespeichert werden
7	Anzeigefeld; Hier wird die History angezeigt
8	Button; Dieser Button dient dem Speichern des Bildes auf der Festplatte in JPG-Format

2.5 Beschriebung der Implementierung

2.5.1 WebSocket

Zuerst wurden passende Packages zur Implementierung eines WebSockets eingebunden, hierfür wurde der klassische JavaX WebSocket gewählt. Danach wurde geklärt, welche Annotationen (@OnOpen, @OnClose etc.) gewählt und welche Methoden (z.B contextInitialized() des Servlets) überlagert werden müssen. Das Versenden und Empfangen von Nachrichten wurde mittels JSON-Nachrichten gelöst, um eine einfach zu verwendenende Kommunikationsbasis zu bilden. Hierfür wurden für sämtliche Elemente De- und Encoder geschrieben und im StartUpListener durch die web.xml mit dem WebSocket verbunden, um passende Nachrichten zu erzeugen und für den Socket lesbar zu machen. Um den JavaScript-Handler anzubinden musste die URL und die Methoden, die aufgerufen werden können festgelegt werden. Zusätzlich wurde erarbeitet, wie (Aufbau der Nachricht) und wann die Nachrichten versendet werden müssen,

um den gewünschten Effekt zu erzielen. Um die Nachrichten zu erhalten werden sie mithilfe der send() Methode des WebSockets im JavaScript-Handler versendet und über ein Session-Set an die anderen Nutzer weitergeleitet. Während der Server anfangs noch in Java programmiert war, wurde später beschlossen diesen ebenfalls auf JavaScript umzustellen, um diesen dem Kontext des Projekts gemäß zu implementieren (dementsprechend sind einige Vorgänge, die zunächst umgesetzt wurden wieder weggefallen; diese werden aber trotzdem für den Bericht ebenfalls beschrieben).

2.5.2 Canvas

Für das Canvas und für die dazugehörigen Buttons wurden zunächst passende HTML-Elemente eingefügt und mit ID's zum Ansprechen versehen. Jedem Element wurde De- und ein Encoder zugeordnet um auf die dazugehörigen Events zu reagieren und passende JSON-Strings an den Socket zu versendne. Die Methoden wurden einfach aufgebaut, um so häufig den gleichen Ablauf zu gewährleisten und für verschiedene Elemente mit gleichem JSON-String verwendbar sind. Der Button selbst ändert hierbei nur den Typ der Zeichnung. Der passende JSON-String wird dann aufgebaut und an den Socket geschickt und entschlüsselt, woraufhin dieser das passende Element erzeugt.

2.5.3 Chat

Zunächst wurden zwei Textfelder (Name, Nachricht) und ein Anzeigefeld (Nachrichtenobjekte) als HTML-Elemente erstellt und mit ID's versehen. Danach wurde ein Listener zum Senden des Textes implementiert und passende De- und Encoder für den WebSocket geschrieben, um den Nachrichtenfluss zu gewährleisten.

2.5.4 Aufbau einer History

Um die History aufzubauen werden zunächst wieder die HTML-Elemente angelegt und De- und Encoder für die JSON-Strings angelegt und im StartUpListener importiert und registriert. Für das Speichern der History wurde

eine Hash-Map angelegt, die sämtliche Nachrichten speichert und im Falle eines neuen Teilnehmers (onOpen() wird aufgerufen) die gesamte History an diesen verteilt. Um das Problem eines doppelten Eintrags (sollte man selbst das Objekt gezeichnet haben würde die Nachricht doppelt auftreten) zu umgehen, wurden Hilfsmethoden eingebaut, die dies unterbinden. Um das Löschen der History zu signalisieren wurden Hilfsmethoden eingebaut, um diese separat bei jedem teilnehmenden Nutzer löschen zu können.

2.5.5 Speichern von Objekten

Um das Speichern von Objekten (in diesem Fall das gezeichnete Bild und die History) wurde ein eigener HTML-Element Typ <a> angelegt. Dieser Typ besitzt das Attribut „Download" , mit dem die Objekte heruntergeladen werden können. Im Falle des Bildes wird die Datei als JPG auf der Festplatte gespeichert, für die History wird mit einer Hilfsmethode ein String erstellt und in einer Datei gespeichert. Dieser enthält alle Objekte der History.

2.6 Probleme der Implementierung

Im Folgenden werden kurz die größeren Probleme beschrieben, die bei der Programmierung und Implementierung auftraten.

Da für das Team am Anfang des Projekts sämtliche JavaScript/Web-Anwendungen etwas neues waren, stellte sich zunächst das Problem, wie man denn überhaupt einen WebSocket aufsetzt und diesen dazu bringt zu agieren und zu reagieren. Zunächst wurden Tutorials als Hilfestellung genutzt, die auch funktioniert haben, bis der Server dann aber wirklich funktioniert hat wie es gewünscht war verging etwas Zeit. Dies lag vorallem an einem zusammenhängenden Problem, der Nachrichtenübermittlung und der sinnvollen Nutzung von JSON-Strings. Ohne größere Erfahrung hat sich das Team mit De- und Encodern und der Verwendung ebendieser auf dem WebSocket etwas schwer getan, wodurch die ersten funktionierenden Nachrichten sich etwas Zeit ließen. Ausserdem war es ein Ziel möglichst wenig

Code für den gleichen Aufwand zu produzieren, was durch verschiedene Ansätze beider Teammitglieder anfangs etwas schwierig zu handhaben war. Schlussendlich konnte dieses Problem gelöst werden, indem möglichst vereinheitlichte Methoden geschrieben und aufgerufen werden, die ebenfalls eine Hilfsmethode aufrufen und so eine einheitliche Struktur in Bezug auf die JSON-Strings erzeugt wurde.

Das zweite größere Problem war die Funktionalität zum Download von Dateien bereitzustellen. Zunächst wurde versucht dies über Methoden und Button mit Hilfe von DataStreams zu bewerkstelligen, allerdings führte dies nicht ansatzweise zu den gewünschten Ergebnissen. Nachdem Google zu Rate gezogen wurde, kam das Team zu dem Entschluss ein eigenes HTML-Element für den Download zu erzeugen, das die Verarbeitung der Daten und das Speichern deutlich erleichterte.

2.7 Umstellung in der Implementierung

Am Ende der Implementierung hat sich das Team entschieden den Server ebenfalls auf JavaScript umzustellen, dies hat zu einigen Veränderungen geführt. Sämtliche En- und Decoder fielen aufgrund der identischen Sprache weg und wurden nicht länger benötigt. Der Server wurde nichtmehr via URL sondern durch das Framework angesprochen, indem man den passenden Typ in der Nachricht angeht (z.B. Chat, History etc.). Vorher mussten diese Typen im JSON-String deklariert werden.

3 Fazit

Abschließend lässt sich sagen, dass das Projekt erfolgreich war, auch wenn kleinere Probleme entstanden wurde doch die gewünschte Anwendung realisiert. Der größte Fehler war es wohl, den Server nicht sofort auf JavaScript-Basis aufzusetzen, dies hätte viel Arbeit erspart.

JavaScript scheint dafür gemacht zu sein kleine und mittelgroße Anwendungen relativ zügig und einfach zu schreiben und zu implementieren; die Funktionen sind einfach und simpel und trotzdem scheint die Programmierung ähnlich

mächtig wie in Java zu sein. In Bezug auf JavaScript war es eine neue Erfahrung, wobei die Programmierung mit JS, dank der Ähnlichkeit zu Java relativ gut machbar war und relativ sauber vonstatten ging. Auch wenn die Anwendung eher die Basics von JS und der Implementierung von WebSockets verwendet, ist dies für die Zukunft eine gute Erfahung und dank des Voranschreitens von JavaScript auch ein sinnvolles Projekt gewesen (WebSockets und Web-Anwendungen werden in den nächsten Jahren vermutlich nicht weniger). Im Zusammenhang mit JavaScript hätte sich das Team mehr Vorkenntnis (nicht in der Vorlesung an sich, sondern in früheren Semestern, vor allem in der Wirtschaftsinformatik) gewünscht, da es mittlerweile eine sehr mächtige Sprache zu sein scheint und diese auch in der Projektphase vonnöten gewesen wäre (hier haben andere die Arbeit übernommen, aber die Vorkenntnis an sich wäre von Vorteil gewesen).