

Fakultät Ingenieurwissenschaften und Informatik  
Studiengang Medieninformatik

Modul

Objektorientierte Analyse und Design

**Dokumentation**

**Entwicklung einer interaktiven und von mehreren Nutzern bemalbare Zeichenfläche als Single-Page-Webanwendung mit Angular JS, Node.js (Websockets) und Netty**

|  |  |
| --- | --- |
| ***Autoren:*** | Führs, Alexander  Hüsemann, Christoph  Graf, Daniel |
| ***Matr.-Nr.:*** | 919369  908397  914743 |
| ***E-Mail:*** | [daniel.graf@hs-osnabrueck.de](mailto:daniel.graf@hs-osnabrueck.de) |
| ***Dozentin:*** | Prof. Dr. Stephan Kleuker |

**I. Inhaltsverzeichnis**

[1. Einleitung 1](#_Toc95940982)

[2. Vorstellung der Technologie 2](#_Toc95940983)

[3. Anforderungsanalyse 3](#_Toc95940984)

[4. Modellierung 4](#_Toc95940985)

[4.1. Front-End Modellierung 4](#_Toc95940986)

[4.1.1. Komponente – „AppComponent“ 4](#_Toc95940987)

[4.1.2. Komponente – „DrawboardComponent“ 4](#_Toc95940988)

[4.1.3. Komponente – „DrawtoolsComponent“ 4](#_Toc95940989)

[4.1.4. Komponente – „HomeComponent“ 4](#_Toc95940990)

[4.1.5. Komponente – „JoinboardComponent“ 4](#_Toc95940991)

[4.2. Back-End Implementierung 4](#_Toc95940992)

[5. Implementierung 5](#_Toc95940993)

[5.1. Front-End Implementierung 5](#_Toc95940994)

[6. Fazit 8](#_Toc95940995)

[Erklärung 14](#_Toc95940996)

II. Abkürzungsverzeichnis

HTML Hypertext Markup Language

UML Unified Modelling Language

QR **Q**uick **R**esponse

UI **U**se**r I**nterface

UX **U**se**r** E**x**perience

#### III. Abbildungsverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

#### IV. Tabellenverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

# Einleitung

In unserem Projekt erstellen wir eine Website, auf der ein Nutzer sich mit seinem Namen oder Anonym anmelden und mit verschiedenen Werkzeugen eine Zeichenfläche bearbeiten kann. Erstellt der Nutzer zuvor eine Session oder tritt einer Session eines anderen Nutzers bei, so können alle Nutzer der Session in einem Chat miteinander kommunizieren und die Zeichenfläche gemeinsam bearbeiten. Die verwendeten Technologien sind Netty als Server, WebSockets (Socket.IO) für die Kommunikation zwischen Server und Clients und AngularJS als JavaScript – Framework zur Erstellung unserer Webanwendung.

Ich habe keine Ahnung was ich sonst dazu schreiben soll …

# Vorstellung der Technologie

Text

# Anforderungsanalyse

Als Ausgangspunkt für unsere Anforderungsanalyse habe wir 5 Use-Cases erstellt: „Anmelden auf der Website“, „Starten einer Zeichen-Session“, „Beitreten einer Zeichen-Session“, „Bemalen der Zeichenfläche“ und „Mainaktivität“.

Beim ersten Use-Case, dem Anmelden auf der Website muss unser System dem Nutzer die Möglichkeit bieten sich entweder als Gast (anonym) oder mit einem Benutzernamen anzumelden. Des Weiteren muss es dazu fähig sein den Namen des Nutzers zu speichern und sollte diesen auch nach der Speicherung anzeigen können.

Im zweiten Use-Case „Starten einer Zeichen-Session“ muss das System dem Nutzer (Host) die Möglichkeit bieten eine neue Session zu starten und diese danach zu konfigurieren. Nach dem Konfigurieren muss das System einen Session-Key erstellen und sollte diesen anzeigen können. Mit dem Session-Key können dann andere Nutzer der Session beitreten (Use-Case „Beitreten einer Zeichen-Session“). Nach dem Anzeigen des Session-Keys muss das System ihn abspeichern können.

Der dritte Use-Case ist „Beitreten einer Zeichen-Session“. Nach dem Anzeigen der Möglichkeit zum Session-Beitritt muss der Nutzer sich mit dem Session-Keys eines anderen Nutzers in der Session anmelden können. Diese Eingabe wird dann vom System validiert. Nach erfolgreicher Validierung wird eine Verbindung zur Session aufgebaut, andernfalls wird diese Verbindung nicht hergestellt.

Im Use-Case „Bemalen der Zeichenfläche“ muss das System als erstes überprüfen, ob eine Session existiert und nach erfolgreicher Sessionüberprüfung (Session existiert) wird es die anderen Nutzer der Session anzeigen. Ein Bemalen der Zeichenfläche ist auch ohne existierende Session möglich, allerdings dann nur für den Nutzer alleine.

Nach der Sessionüberprüfung soll der Nutzer das Zeichentool ändern und die Zeichenfläche mit dem ausgewählten Tool bearbeiten können. Darauffolgend muss das System die Zeichenfläche bei allen Nutzern aktualisieren und wird diese anzeigen. Zuletzt muss der Nutzer die Möglichkeit haben die Session zu schließen bzw. diese zu verlassen (wenn zuvor einer Sitzung beigetreten wurde). Beim Scheitern der Schließung bleibt der Nutzer in der Zeichenfläche und kann wieder das Zeichentool ändern bzw. die Zeichenfläche bearbeiten.

Zu guter Letzt kommt noch das Use-Case „Mainaktivität“. Hierbei wird nur beschrieben, dass das System allgemein dazu fähig sein wird alle Interaktionsmöglichkeiten anzuzeigen und dem Nutzer die Möglichkeit der Auswahl einer Interaktion bietet. Außerdem muss das System nach der Auswahl einer Zeichensession die Anmeldung bei der Website überprüfen und nach dem Scheitern dieser dem Nutzer die Möglichkeit bieten sich selbst anzumelden.

# Modellierung

Hallo

# Front-End Modellierung

Die grundlegenden Klassenstruktur in Angular ist als Model-View-Controller Pattern aufgebaut und ist wie folgt:

* „Components“ – Controller
* „Component-Template“ (HTML) – View
* „Services“ – Model

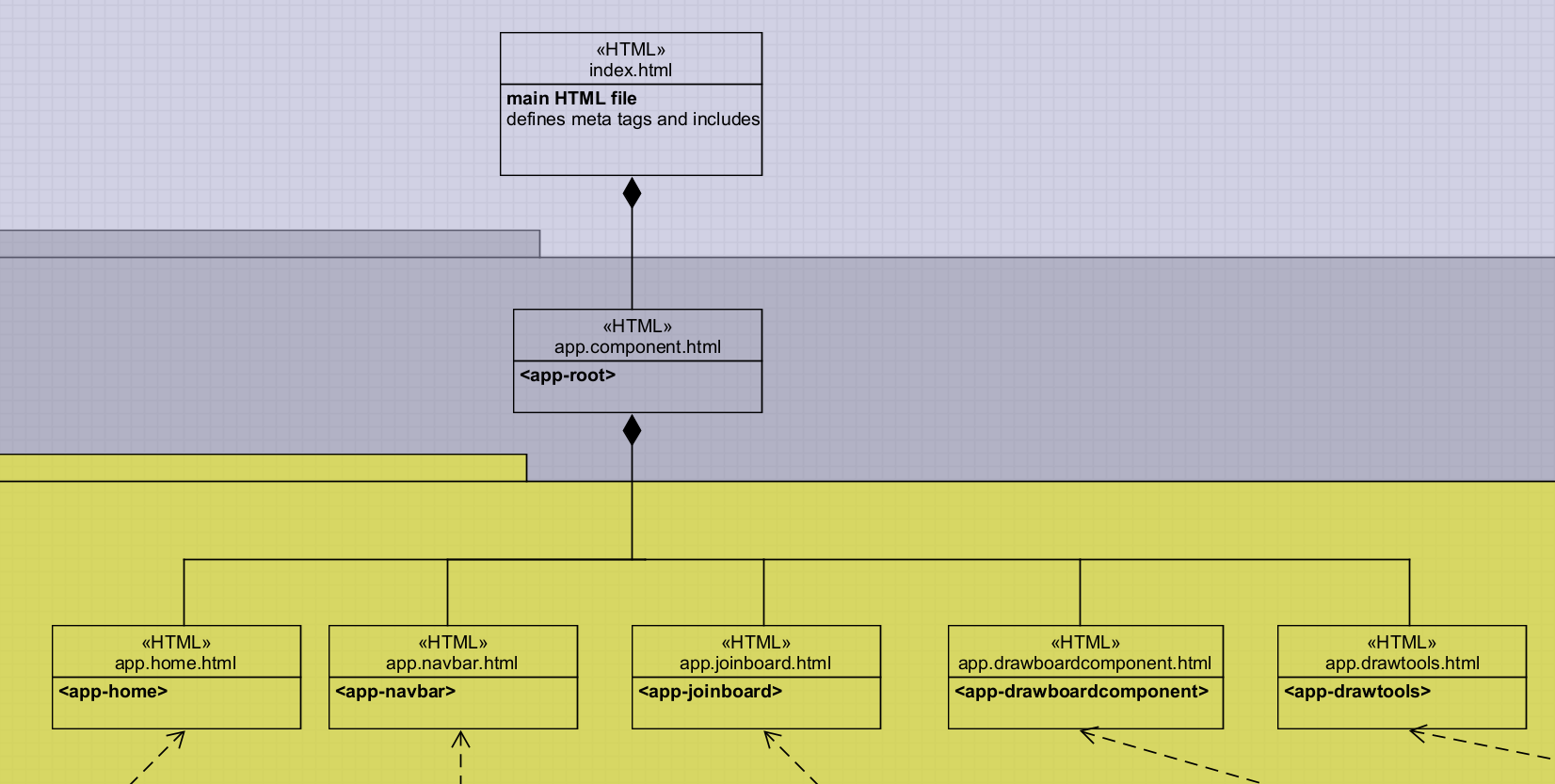
Die Komponenten und Views sind hierbei immer fest verbunden um im Laufe der UML-Struktur der Front-End Anwendung ergibt sich eine hierarchische Organisation zwischen diesen. So stellt „AppModule“ & „index.html“ das Kernstück der Anwendung dar, da es alle weiteren Komponenten/HTML-Template Paare definiert und so diese davon abhängig sind (vgl. Abbildung 1). Diese HTML-Klassen sind dabei immer unmittelbar mit den gleichnamigen Komponenten verknüpft und spiegeln somit das Verhältnis [View ⬄ Controller] dar.

Abbildung 1: Hierarchische View(HTML)-Struktur der Front-End Anwendung in Angular

Daraufhin folgt „AppComponent“ & „app.component.html“, welches die erste realisierte Komponente darstellt. Innerhalb dieser Komponente werden auch die anderen erstellten Komponenten des Programmes angezeigt. Diese weiteren Komponenten haben wir basierend auf den Anforderungen in fünf Komponenten aufgeteilt: „HomeComponent“, „NavBarComponent“, „JoinboardComponent“, „DrawboardComponent“ und „DrawtoolsComponent“. Wobei „NavBarComponent“ hier nur die obere Navigationsleiste anzeigt, welches für diesen Iterationsschritt jedoch unwichtig ist. Im späteren Verlauf können hier aber auch weiter Funktionen eingebaut werden.

# Komponente – „AppComponent“

„AppComponent“ stellt hier den ersten wichtigen Schlüsselpunkt der Anwendung dar, so überprüft dieses programmiertechnisch, ob der User sich schon angemeldet hat und manipuliert entsprechend die Haupt-HTML, um andere Komponenten anzuzeigen und nicht mehr relevante zu verbergen. Diese Verifizierung geschieht jedoch in anderen Klassen und so wird hier ein Observer-Pattern genutzt, um „AppComponent“ über den Anmeldestatus des Nutzers zu informieren. Dazu wird hier „UserHandlerService“ genutzt, woraufhin sich „AppComponent“ anmeldet. Erlangt nun die Komponente „HomeComponent“ den Anmeldestatus des Nutzers und gibt diesen and „UserHandlerService“ weiter, so wird „AppComponent“ darüber informiert und kann dementsprechend die HTML Anpassen, indem es das Document-Object-Model dieser ändert.

# Komponente – „DrawboardComponent“

Die Klasse „DrawboardComponent“ ist für die Funktionalität und Abhandlung der Zeichenfläche verantwortlich. So verschickt diese Events an den Server und reagiert auch entsprechend auf empfangende Events. Um die Datenübertragung der Zeichenfläche zu strukturieren, benutzt diese Komponente folgende Klassen:

* „Point“: Datenobjekt mit 4 Variablen (x-Koordinate, y-Koordinate, Größe und RGB-Farbe)
* „Paths“: n-stelliges Array von „Point“
* „ClientPaths“: Beinhaltet eindeutige Session-ID und ein n-stelliges Array von „Path“

So kann für jeden verbundenen Client eine Summe von verschiedenen Pfaden bestehend aus einer Anzahl von Punkten zugewiesen werden. Die Wichtigkeit dieser Strukturierung folgt mit der von „DrawboardComponent“ delegierten Klasse „canvas.js“. Diese ist für das Rendern der Zeichenfläche verantwortlich und bietet dafür bestimmte Methoden für diese Komponente dar. Das Zeichnen basiert hier nämlich auf eine bestehende Sammlung von Punkten, welche dann innerhalb der Logik von „canvas.js“ gezeichnet werden soll. Durch dieses System können die verschiedenen Pfade dann zwischen den verschiedenen Clients auch untereinander ausgetauscht werden, so dass jeder Client die Pfade des anderen erhält, um so eine gemeinsame Zeichnen-Session zu gewährleisten. Auch delegiert diese Klasse einer Instanz von „socketio.js“, um im Fall eines Session-Beitritts verschiedene Events an der Server schicken und empfangen kann. Dies muss jedoch nicht immer der Fall sein, da der Nutzer auch im offline-Modus die Zeichenfläche bemalen kann.

# Komponente – „DrawtoolsComponent“

Um auch die Zeichentools für den Nutzer zu bieten, wurde die Komponente „DrawtoolsComponent“ und deren View „app.drawtools.html“ erstellt. Diese erschafft zusammen mit der Klasse „canvas.js“ die Funktionalität für die Zeichentools und soll es dem Nutzer erlauben, die Zeichendicke als auch Farbe einzustellen. So arbeitet „canvas.js“ Hand-in-Hand mit der HTML-View von „DrawtoolsComponent“ und reagiert auch entsprechende Veränderung der Zeichentools.

# Komponente – „HomeComponent“

Nachdem die fundamental wichtigen Komponenten für den Zeichenprozess erläutert wurden, kommen wir nun zu den Komponenten, welche für die Client-to-Client Abhandlungen verantwortlich sind.

„HomeComponent“ ist eine davon und bearbeitet die Nutzereingaben für den zu anmeldenden Namen und Passwort. Diese Komponente delegiert eine Instanz des „UserHandlerService“ und informiert bei einem passendem Namen und Passwort die Beobachter dieses Services, so dass diese die Nutzerdaten erhalten können.

# Komponente – „JoinboardComponent“

Als Herzstück der Zeichen-Sessions, ist diese Komponente verantwortlich für die Zusammenführung der verschiedenen Clients in eine Session. Es handelt die Ereignisse ab, ob ein Nutzer eine Session erstellen möchte, beitreten möchte, verlassen möchte sowie auch die verschiedenen Chat-Abhandlungen. Diese werden dann vom View dieser Komponente UI-technisch dargestellt.

Um die Nutzerdaten zu erhalten, erhält diese Komponente eine Instanz des „UserHandlerService“ und meldet sich auf diesen als Beobachter an.

Da diese Komponente auf client-to-client Events basiert hat sie entsprechend auch eine Instanz des „socketio.js“, um die Verbindung zu initialisieren und um Events an den Server zu schicken und zu erhalten. Desweitern hat sie eine Instanz der „DrawboardComponent“, um diese mittels „initSocketConnection()“ zum Server zu verbinden, so dass diese dann die „ClientPaths“ an andere Clients schicken kann.

# Back-End Implementierung

# Implementierung

Nach der Modellierung im UML kommt es nun zu der letztendlichen Implementierung im Code. Dazu wurde die Struktur auch wie im UML in zwei große Hauptkomponenten unterteilt. Zum einen wurde der Client, welche das Front-End des Systems darstellt und mittels Angular und JQuery in den Sprachen JavaScript, TypeScript und HTML realisiert wurde. Der Server hingegen wurde mittels Netty und Spring in Java entwickelt.

# Front-End Implementierung

Dies in Kapitel 4 beschriebenen Komponenten und Services lassen sich mit einfachen Befehlen im Angular Projekt generieren und sind dann auch direkt in den Abhängigkeiten verknüpft („ng generate service/component <name>“). Als ergänzende Klassen wurde z. B. „canvas.js“ & „socketio.js“ erstellt, welche sich um bestimmte Funktionalitäten kümmern, welche in JavaScript realisiert wurden.

Ein Bild, das Text, Screenshot, Monitor enthält.

Automatisch generierte BeschreibungDas Hauptfeature der Anwendung ist die Zeichenfläche, welche wie schon beschrieben von den Komponenten „DrawboardComponent“, „DrawtoolsComponent“ sowie „canvas.js“ gesteuert wird. Nennenswert sind hier auch die Undo- & Redo-buttons, welche es dem Nutzer erlauben, sein gemaltes rückgängig zu machen (Siehe Abbildung 2).

Abbildung 2: Zeichenfläche mit verschiedenen Buttons und Zeichentools

Die grundlegende Idee hier ist, dass der Nutzer mittels des Zeichnens nur einen Array von „Points“ füllt, welche für jede bemalte Position die Koordinaten, Farbe und Zeichendicke speichert. Dieser Array wird dann von „canvas.js“ iteriert und mittels dieses Pfades können diese Punkte auf der Zeichenfläche dargestellt werden (siehe Abbildung 3). Jedoch stellen diese einzelnen Punkte für sich alleinstehend keinen gezeichneten Strich dar und so ergänz die Funktion „fillPoints()“ die Striche zwischen den Verschiedenen Punkten.

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungInsgesamt werden dann die Pfade für den aktuellen Nutzer, sowie auch im Falle einer Zeichen-Session, die Pfade der anderen Nutzer so gespeichert, um ein gemeinsames Bild zu erschaffen. Jeder andere Client in der Session wird mittels seine eindeutigen SessionID identifiziert und das erlaubt es auch, die Pfade der verschiedenen Nutzer voneinander zu unterscheiden. Somit kann ein Client lokal seine Zeichnung abwischen, jedoch werden die Pfade der anderen nicht gelöscht.

Abbildung 3: Funktion, welche über die Pfade iteriert und die Punkte visuell als Striche darstellt

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungDank dieser Struktur können dann auch die eigenen Pfade einfach an die anderen verbundenen Zeichner gesendet werden. Dazu wird der eigene Pfad in das gesonderte Datenobjekt „ClientPaths“ übergeben und dann letztendlich als JSON-String an den Server geschickt. Dieser reagiert dann auf dieses Event(„sendCanvasPathDataToServer“) und übermittelt dann diese Pfade an die anderen verbundenen Clients. Durch die eindeutige Client-SessionID ist es dann möglich, jedem verbundenen Client unmittelbar seine Pfade zuzuweisen.

Abbildung 4: Übertragung der eigenen Pfade an den Server, welcher diese an die anderen verbundenen Clients in der Session vermittelt

# Fazit

Im nächsten Iterationsschritt sollte die aktuelle Programmstruktur überarbeitet werden, d. h. einige Funktionalitäten bestimmter Komponenten auslagern, um die Kohäsion zu verringern. Auch hätten wir uns konkret nur auf den ersten Iterationsschritt fokussieren müssen, einige aus dem zweiten Schritt wurden auch schon implementiert, jedoch hatte dies zur Folge, dass es insgesamt vom Aufwand her zu viel wurde und wir so einige Sachen vernachlässigen mussten.

Insgesamt kann man jedoch sagen, dass die grundlegenden Anforderungen aus dem ersten Inkrementationsschritt durchaus eingehalten wurden und wir diese auch erfolgreich in den Prototypen einbinden konnten. Dabei haben wir auch alle erarbeiteten Nutzeranforderungen implementieren und erfolgreich umsetzen können

V. Referenzen

References

[1] L. Nielsen, *Personas - User Focused Design,* 2nd ed. London: Springer London, 2019.

[2] A. Endmann and D. Keßner, “User Journey Mapping – A Method in User Experience Design,” *i-com*, vol. 15, no. 1, pp. 105–110, 2016, doi: 10.1515/icom-2016-0010.

[3] R. Hamilton and L. L. Price, “Consumer journeys: developing consumer-based strategy,” *J. of the Acad. Mark. Sci.*, vol. 47, no. 2, pp. 187–191, 2019, doi: 10.1007/s11747-019-00636-y.

[4] D. Saffer, *Designing for interaction: Creating innovative applications and devices,* 2nd ed. Berkeley, CA: New Riders, 2010.

[5] DI Andrea Heistinger, *Qualitative Interviews – Ein Leitfaden zu Vorbereitung und Qualitative Interviews - Ein Leitfaden zu Vorbereitung und Durchführung inklusive einiger theoretischer Anmerkungen.* [Online]. Available: https://​www.uibk.ac.at​/​iezw/​mitarbeiterinnen/​senior-​lecturer/​bernd\_​lederer/​downloads/​durchfuehrung\_​von\_​qualitativen\_​interviews\_​uniwien.pdf

[6] P. Yan and J. Guo, “The research of Web usability design,” in *The 2nd International Conference on Computer and Automation Engineering (ICCAE), 2010: 26 - 28 Feb. 2010, Singapore ; proceedings*, Singapore, 2010, pp. 480–483.

[7] B. Shneiderman and C. Plaisant, *Designing the user interface: Strategies for effective human-computer interaction*, 2016.

[8] S. Thesmann, *Interface design: Usability, user experience und accessibility im Web gestalten,* 2nd ed. Wiesbaden: Springer Vieweg, 2016.

[9] F. Makedon, Ed., *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments: https://www.researchgate.net/publication/221410323\_Human-Computer\_Interaction\_and\_the\_older\_adult\_An\_example\_using\_user\_research\_and\_personas*. New York, NY: ACM, 2010.

[10] Y. Yesilada and S. Harper, *Web Accessibility: A Foundation for Research,* 2nd ed., 2019.

[11] G. Perlman, *Interaction Design beyond human-computer interaction*. New York: John Wiley and Sons inc. (Hsrg.), 2002.

[12] M. Hua and Q. Fei, “The value of unconscious behavior on interaction design,” in *IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design, 2009: CAID & CD 2009 ; Wenzhou, China, 26 - 29 Nov. 2009*, Wenzhou, China, 2009, pp. 336–339.

VI. Anhang

[**Anhang 1: Anti-Persona Alex Führs** 18](#_Toc83058234)

[**Anhang 2: Sekundär-Persona Kilian van Derbot** 19](#_Toc83058235)

[**Anhang 3: Primär-Persona Theresa Kjellberg** 20](#_Toc83058236)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**Anhang 1: Anti-Persona Alex Führs**

**Anhang 2: Sekundär-Persona Kilian van Derbot**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**Anhang 3: Primär-Persona Theresa Kjellberg**

# Erklärung

Ich versichere, dass ich die Arbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Osnabrück, 17.02.2022   
Ort, Datum Daniel Graf, Alexander Führs, Christoph Hüsemann