Entwicklung eines Twitter-Klons

Konzeption und Gestaltung

Joan-Angelo Douvere  
Melina Brkic  
Egor Hans  
Pongsakorn Chuaynum

[Jahr]

Inhaltsverzeichnis

[1. Anforderungsanalyse 0](#_Toc465293404)

[1.1. Kurzbeschreibung 0](#_Toc465293405)

[1.2. Anwendungsfunktionen / Use-Cases 0](#_Toc465293406)

[1.3. Nicht-funktionale Anforderungen 1](#_Toc465293407)

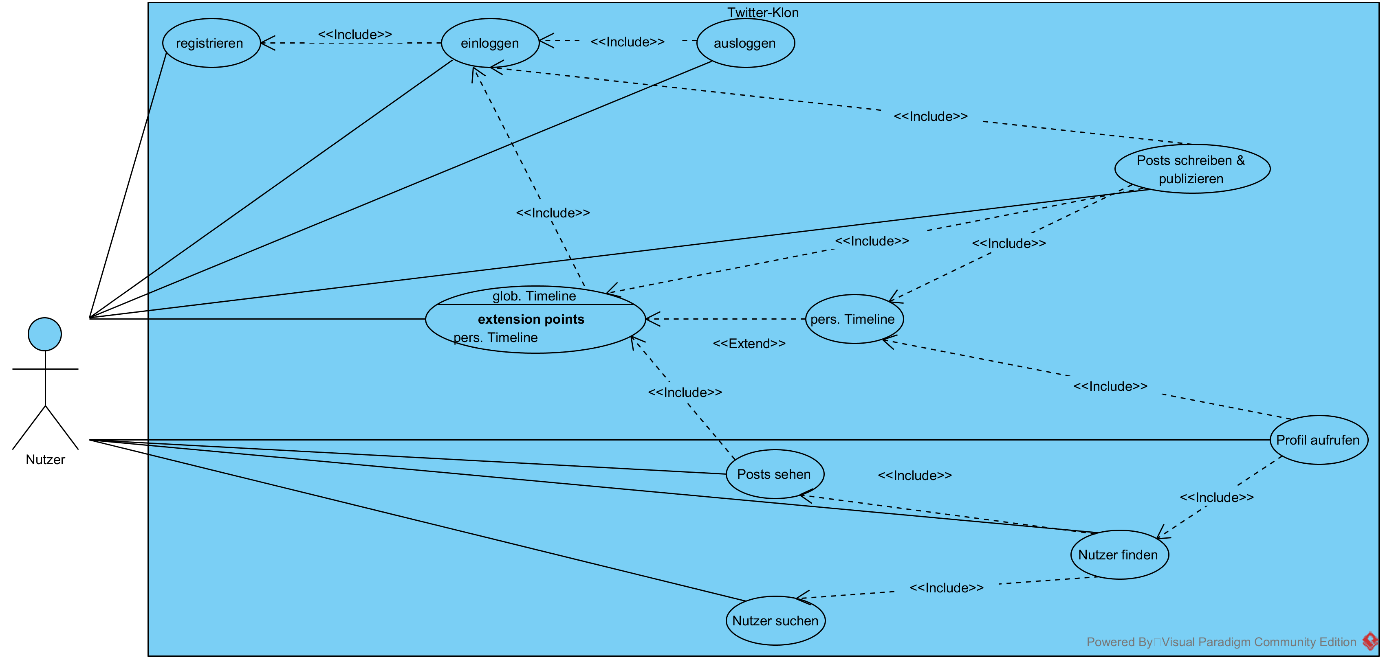
[1.4. Verwendete Techniken 1](#_Toc465293408)

[2. Seitennavigation 2](#_Toc465293409)

[3. Mockup Screenshots 2](#_Toc465293410)

[4. Datenmodell 2](#_Toc465293411)

# Anforderungsanalyse



## Kurzbeschreibung

Bei der Web Anwendung handelt es sich um eine einfache Social Media Anwendung, bei der **Nutzer** einen persönlichen (Micro-) **Blog** aus Kurznachrichten (**Posts**) erstellen können, den sie mit den anderen Nutzern der Community teilen.

## Anwendungsfunktionen / Use-Cases

* + 1. Als Nutzer will ich mich **registrieren**, um von anderen Nutzern identifiziert werden zu können
       1. Name
       2. Passwort
       3. Nur wenn nicht angemeldet
    2. Als Nutzer will ich mich **einloggen**, um auf meine Funktionen zugreifen zu können
       1. „Login“
       2. Begrenzter Zeitraum, danach 🡪 1.2.3.
       3. Nur wenn nicht angemeldet
    3. Als angemeldeter Nutzer will ich mich **ausloggen**, um mein Konto zu sperren
       1. „Logout“
       2. Nur wenn angemeldet
       3. Siehe auch 1.2.2.2.
    4. Als angemeldeter Nutzer will ich Posts **schreiben und publizieren**
       1. Nur wenn angemeldet
       2. Maximal 140 Zeichen
    5. Als Nutzer will ich Posts auf Timelines **sehen**
       1. Nachrichtentext
       2. Autor
       3. Zeitpunkt der Publikation
       4. Begrenzte Anzahl von Posts pro Seite
       5. Möglichkeit, Seiten zu wechseln (Paging/Pagination)  
          Alternativ: kontinuierliches Nachladen während Down-Scrolling
       6. Unmittelbarer Hinweis, falls ein neuer Post publiziert wird, während man Timeline anschaut
    6. Als Nutzer will ich mir eine **globale Timeline** anzeigen lassen
       1. Mit allen Posts des Systems
    7. Als angemeldeter Nutzer will ich mir eine **persönliche Timeline** anzeigen lassen
       1. Mit eigenen Posts und Posts von Nutzern der Community, denen ich folge
       2. Nur wenn angemeldet
    8. Als angemeldeter Nutzer will ich andere Nutzer **finden**
       1. Als Autoren von Posts auf der globalen Timeline
       2. Über **Explizite Suchfunktion**
          1. Erlaubt die Eingabe des Anfangs eines Namens
          2. Nicht Case-Sensitiv (aAbBcC)
          3. Das Ergebnis wird als **Nutzerliste** angezeigt

Auch Nutzerlisten verwenden Paging oder Nachladen zur Übersicht

* + - 1. Optional: durch Erwähnung innerhalb des Nachrichtentextes
      2. Nur wenn angemeldet
    1. Als angemeldeter Nutzer will ich die **Profile** anderer Nutzer, die ich gefunden habe, sehen
       1. Ein Profil ist eine Timeline, die nur die Posts des jeweiligen Nutzers zeigt.
       2. Es ist auf einem Profil auch möglich, einem Nutzer zu folgen
       3. Zu jedem Profil gehören zwei Follower Listen
          1. Eine enthält die Nutzer, denen der jeweilige Nutzer selbst folgt
          2. Eine enthält solche Nutzer, die ihm folgen

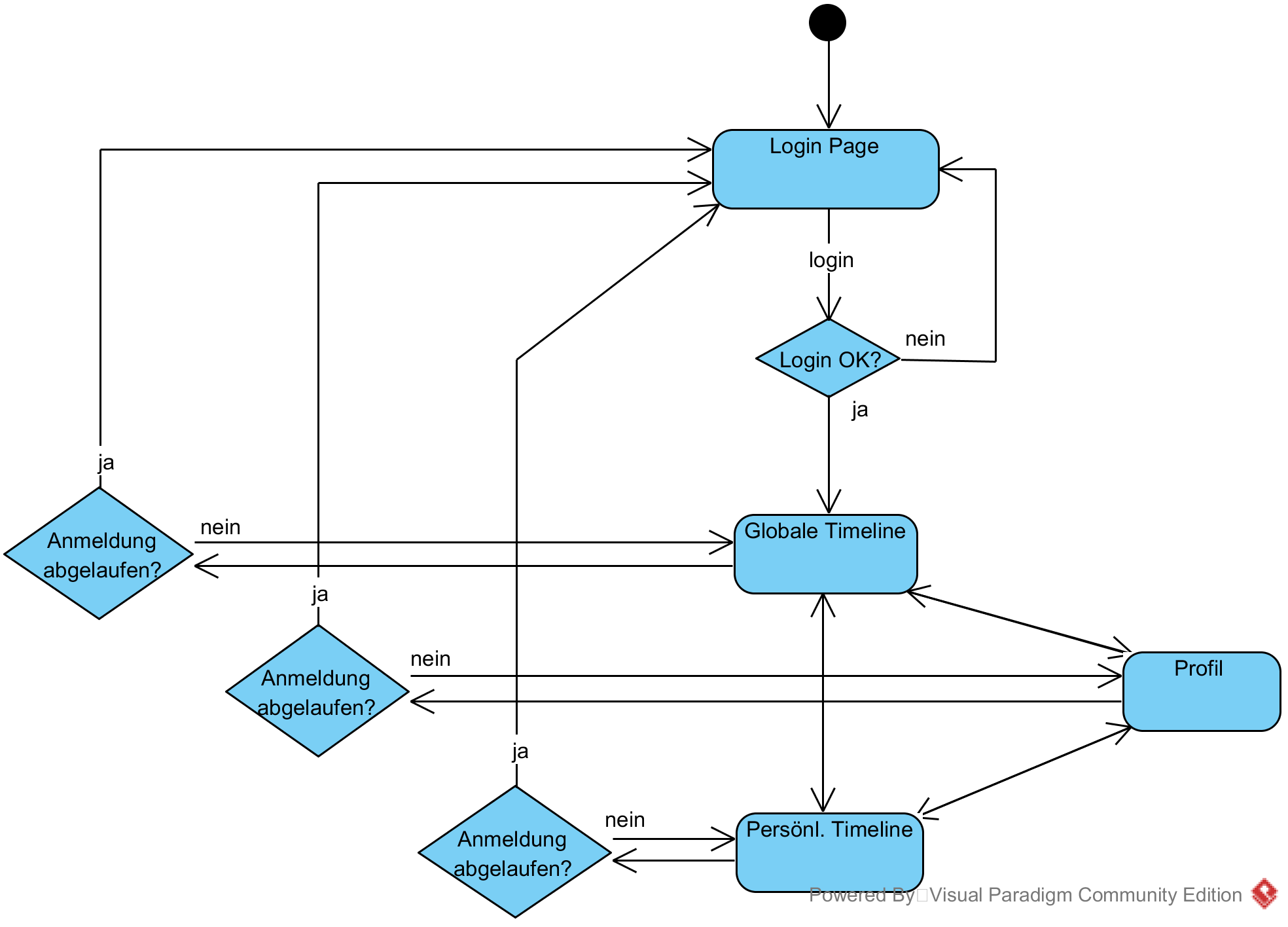
## Nicht-funktionale Anforderungen

* + 1. Die Anwendung soll **Replikationstransparent** sein
       1. Bei einer Weiterleitung auf ein anderes Webserverreplikat (Lastverteilung), soll die Anmeldung erhalten bleiben. (Lastverteilung muss nicht implementiert werden)
    2. Hinweise zu neuen Posts sollen angemeldeten Nutzern in **nahezu Echtzeit** erscheinen.
       1. Insbesondere soll dies unabhängig von dem Aufruf einer neuen Seite durch den Nutzer geschehen.
       2. Der Hinweis soll durch ein partielles Update der aktuellen Seite auf Basis von asynchroner Kommunikation realisiert werden.
       3. Solche Hinweise erhält ein Nutzer zudem auch dann, wenn zugrundeliegende Posts auf anderen Webserverreplikaten verfasst wurden.
    3. **Responsive Design** für korrekte und komfortable Darstellung auf Desktop und Mobile.

## Verwendete Techniken

* + 1. **3-Tier-Client-Server-Architektur**
    2. Serverseitig **Java + Spring** Framework
    3. Soll MVC-Muster nach Vorgabe von Spring Web MVC folgen und Thymeleaf Template Engine nutzen.
    4. Datenebene durch **Redis** Key-Value-Store realisieren
    5. Empfehlung: Clientseitig **Bootstrap** & ggf. Javascript/**jQuery**
    6. Asynchrone Kommunikation zwischen Browser und Webserver über **Web Sockets**
       1. Wird in Spring mit dem Messaging Protokoll **STOMP** und einem einfachen Message Broker ergänzt
       2. Der Nachrichtenaustausch zwischen Webserverreplikaten soll über Redis Publish/Subscribe Channels erfolgen

# Seitennavigation



# Mockup Screenshots

# Datenmodell

