

UNIVERSITÄT SIEGEN

PROJEKTBERICHT

Design eines Blogsystems als dezentrales Soziales Netzwerk

Autor:
Malte PASKUDA

Betreuer:
Dr. Lars FISCHER

März 2013



Zusammenfassung

Im Rahmen dieser Projektarbeit wurde ein Blogsystem entworfen, entwickelt und getestet, das Grundfunktionen eines dezentralen Sozialen Netzwerkes enthält. Der Auswahl dieser Grundfunktionen - Freundeslisten, Nachrichtenversand und Aktivitätenstream - lagen aus Kurzinterviews erstellte Personas zugrunde. Abschließend wurde ein Nutzertest durchgeführt und ausgewertet, aus dem sich Erkenntnisse zur weiteren Entwicklung eines solchen Netzwerkes gewinnen lassen.

1 Einleitung

Obwohl das Internet im Laufe seiner Entwicklung immer weiter Teil des Alltagslebens der Menschen geworden ist, hat nicht jeder Nutzer des Internets seinen eigenen Webserver. Stattdessen werden sogenannte Soziale Netzwerke wie Facebook häufig genutzt, auf denen Nutzer ihre eigene Seite anlegen und sich mit anderen Nutzern vernetzen können. Doch so viele Informationen über und so viel Kontrolle über das Alltagsleben der Menschen in die Hände einzelner großer Konzerne zu geben, wie das bei Sozialen Netzwerken aufgrund ihrer zentralen Struktur derzeit der Fall ist, birgt Risiken, von denen der Verlust der Daten im Falle des Niedergangs des Konzerns noch das kleinste ist. Wären Soziale Netzwerke dagegen dezentrale Systeme, könnte zumindest der unmittelbare Einfluss des Betreiberkonzerns reduziert werden. Daher wurde der Versuch unternommen, ein dezentrales Soziales Netzwerk zu entwerfen.

2 Systembeschreibung

Dsnblog¹ ist eine Ruby/Sinatra Anwendung mit Sqlite als Datenbank, die einem Blogsystem nachempfunden wurde. Als externes Loginsystem wurde Browserid (Persona) von Mozilla genutzt, wodurch sich jeder Nutzer mit einer beliebigen Emailadresse einloggen kann. Der erste Nutzer, der dies macht, wird als Administrator in der Datenbank gespeichert.

Dass Blogs mit ihren Verbindungen untereinander einem Sozialen Netzwerk entsprechen ist der Kerngedanke von *dsnblog*. Ein Soziales Netzwerk kann definiert werden als Software mit den folgenden Möglichkeiten (Boyd and Ellison [2007]):

1. Erschaffung eines semi-öffentlichen Profiles, das also bestimmte Informationen nur einem begrenzten Nutzerkreis öffentlich machen kann, innerhalb eines begrenzten Systems
2. Veröffentlichen einer Freundesliste von mit dem Nutzer verbundenen Nutzern
3. Sehen und Traversieren der eigenen Liste von Verbindungen und der anderer Nutzer

¹<https://github.com/onli/dsnblog>

Gleichzeitig haben Blogs meiner Einschätzung nach eine Reihe von Vorteilen gegenüber etablierten Sozialen Netzwerken. Der hauptsächliche: Sie können üblicherweise auf Servern der Nutzer laufen - sie sind ein dezentrales System - wodurch der Nutzer vollen Zugriff auf, und damit Kontrolle über, seine Daten hat.

Normale Blogaktivitäten wie das Schreiben von Einträgen, Kommentieren und Senden von Ping- und Trackbacks kann jede Installation wie ein normaler Blog völlig unabhängig von allen anderen Installationen ausführen. Für die Funktionen des Sozialen Netzwerkes wurde dagegen ein eigenes Protokoll entworfen, das mit dsns (siehe Hiekmann [2013]) als Vermittler diese Funktionen ermöglicht.

Dieses dsns-System wurde als zentrale Infrastruktur entworfen. Bei der Installation eines *dsnblogs* wird die URL und die identifizierende Emailadresse an dsns übermittelt und dort gespeichert. Die zur Email gehörende URL kann dann später abgefragt werden. Dies ermöglicht die in den folgenden Abschnitten beschriebenen Funktionen.

2.1 Freunde hinzufügen

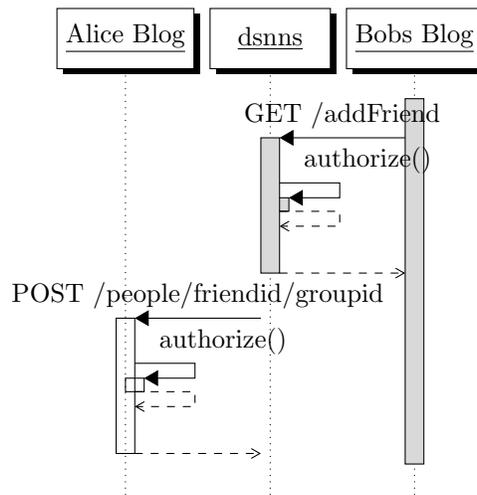


Abbildung 2: Protokoll: Freund hinzufügen

Um Bob als Freund hinzuzufügen, muss Alice den Blog von Bob besuchen. Auf dem Blog befindet sich ein Button "Add Bob as friend". Dieser Button leitet Alice zu dsns weiter, demgegenüber sich Alice als Alice identifiziert, wodurch dsns Alice zu ihrem eigenen Blog weiterleiten kann. Dort wird /post/people/friend aufgerufen, wobei die Emailadresse von Bob als Post-Parameter übergeben wird. Wieder muss Alice sich authentifizieren, dann kann ihre *dsnblog*-Installation Bob als Freund in die Datenbank eintragen (siehe Abbildung 2).

Herausforderung beim Erstellen dieser Funktion war die Tatsache, dass im Moment des Besuches von Bobs Blog dieser nicht weiß, wie die URL des Blog von Alice lautet. Deshalb musste die Umleitung über dsnnns eingebaut werden.

2.2 Streamgenerierung

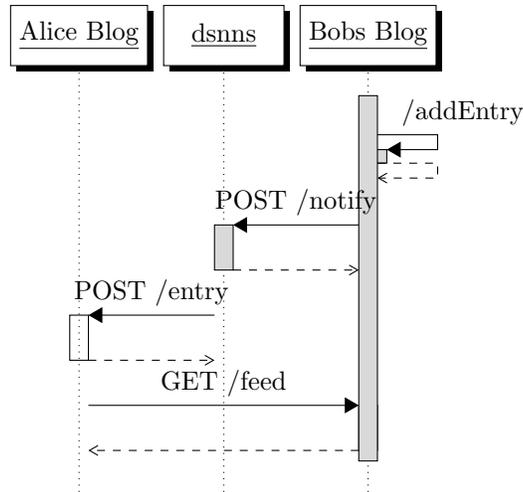


Abbildung 3: Protokoll: Stream generieren

Um einen Stream der Aktivitäten der Freunde zu generieren, muss *dsnblog* mitbekommen, wenn auf den Blogs der Freunde neue Einträge erscheinen. Diese Blogs wissen aber möglicherweise gar nicht, dass Alice sie als Freunde eingetragen hat, denn Freundesbeziehungen sind unilateral, daher können diese Blogs Alice Blogs nicht direkt benachrichtigen. Stattdessen agiert dsnnns als Hub: Bei der Freundeshinzufügung trägt sich Alices *dsnblog* bei dsnnns als Abonnent ein und immer wenn ein Blog einen Eintrag veröffentlicht, pingt dieser dsnnns an, das dann wiederum die Blogs der Abonnenten benachrichtigen kann. Um den Stream zu generieren, wird der RSS-Feed des Blogs gezogen und neue Einträge in der Datenbank gespeichert (siehe Abbildung 3).

2.3 Nachrichtenversand

Will Alice Bob eine Nachricht schreiben, generiert Alices *dsnblog* einen Key und verschlüsselt mit diesem per blowfish-CBC den Text der Nachricht. Bobs Blog wird der Key geschickt, der daraufhin die verschlüsselte Nachricht von Alices Blog holen kann. Dass Bobs selbst die Nachricht von Alices Blog holt soll verhindern, dass Nachrichten untergeschoben werden können, denn Bobs Blog muss dafür von dsnnns die URL von Alices Blog holen und hat so weitestgehende Sicherheit, dass die Nachricht tatsächlich von Alices Blog stammt. Es besteht also kein echter Schutz gegen einen Man-in-the-middle-Angriff, aber ein

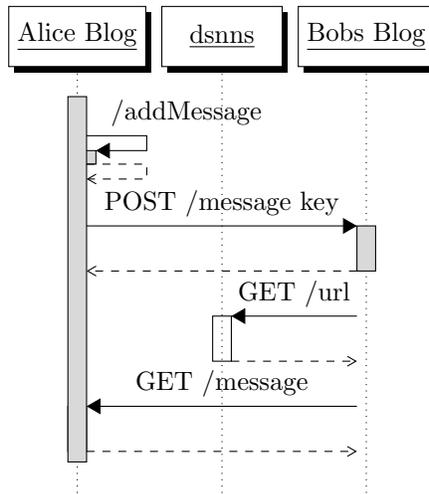


Abbildung 4: Protokoll: Nachricht versenden

Angreifer braucht die Fähigkeit, die Anfrage von Bobs Blog an Alices Blog abzufangen und selbst zu beantworten, um eine Nachricht mit Alice als Absender zu fälschen.

Hat er schließlich Key und verschlüsselte Nachricht, kann Bob diese entschlüsseln (siehe Abbildung 4).

2.4 Verwandte Systeme

Im Folgenden werden verwandte Systeme vorgestellt, die entweder für den Bereich der Social Networks (Facebook) oder Blogs (Wordpress) aufgrund ihrer Verbreitung besondere Relevanz haben, ebenfalls eine verteilte Infrastruktur nutzen (Diaspora) oder den Gedanken, aus Blogs ein soziales Netzwerk zu machen, verfolgen (Xanga).

2.4.1 Facebook

Facebook ist das derzeit größte Soziale Netzwerk². Es beinhaltet einen Freundesgraph, Profile, eine Chatfunktion und Fotoalben, zusätzlich ist es eine Plattform für weitere Anwendungen wie Spiele, die innerhalb des Netzwerkes ausgeführt werden können. Die Daten der Nutzer werden zentral auf der Serverinfrastruktur des börsennotierten Unternehmens gespeichert.

Dsnblog soll ähnliche Funktionen wie Facebook unterstützen, dabei aber durch eine dezentrale Serverinfrastruktur dem Nutzer Kontrolle über seine Daten geben.

²siehe SOCIAL PLATFORMS GWI.8 UPDATE: Decline of Local Social Media Platforms, <http://globalwebindex.net/thinking/social-platforms-gwi-8-update-decline-of-local-social-media-platforms/>

2.4.2 Wordpress

Wordpress ist die meistverbreitete Blogsoftware ³. Wordpress-Blogs können von Nutzern auf eigenen Servern gehostet werden, alternativ bietet die Wordpress.com-Plattform kostenlose Blogs an. Einzelne Blogs kann man als Teil eines Sozialen Netzwerkes betrachten, wobei dann Track- bzw Pingbacks die Verbindungen zwischen den einzelnen Blogs bilden würden. Generell gelten Blogs jedoch nicht als Soziales Netzwerk und es fehlen ihnen allgemein und Wordpress im speziellen auch Funktionen, die von Sozialen Netzwerken normalerweise beherrscht werden, wie die Möglichkeit, eine explizite Freundesliste zu führen (Boyd and Ellison [2007]).

Dsnblog ist in erster Linie eine Blogsoftware und orientiert sich daher bei der Interfacegestaltung stark an Wordpress. Im Gegensatz zu Wordpress sollen zusätzliche Funktionen das Soziale Netzwerk, das Blogs bilden, explizit sichtbar und kontrollierbar machen.

2.4.3 Diaspora

Diaspora ist ein dezentrales Soziales Netzwerk. Implementiert unter Nutzung des Frameworks Ruby on Rails, können Nutzer die Software entweder auf eigenen Servern hosten oder bei bestehenden Installationen, sogenannten pods, Accounts anlegen. Diaspora bildet die grundlegenden Funktionen eines Sozialen Netzwerkes ab, arbeitet jedoch mit einer verteilten Infrastruktur und gibt dem Nutzer so die volle Kontrolle über seine Daten.

Diaspora wurde anfangs durch eine Kickstarter-Kampagne finanziert. Grundgedanke von *dsnblog* ist, eine ähnliche Plattform zu erschaffen, aber durch die Bloganleihen eine Nutzerschaft zu erreichen, die der Idee von selbstgehosteter Software positiv gegenüberstehen sollte.

2.4.4 Xanga

Xanga bezeichnet sich selbst als "the blogging community". Auf der Webseite können Nutzer Blogs anlegen und zusätzlich Funktionen Sozialer Netzwerke nutzen, u.a. eine Freundesliste und ein Profil anlegen, Nachrichten verschicken sowie Fotos und Videos hochladen.

Bloggingplattformen wie xanga (und auch Wordpress.com) erfüllen grundsätzlich das Ziel, aufbauend auf einem Blogsystem ein Soziales Netzwerk zu erbauen, das System ist aber nicht dezentral und damit nicht unter Kontrolle der Nutzer.

3 Designablauf

Die Grundlage des Systems, die Blogelemente der Seite, wurden in einem ersten Schritt eigenen subjektiven Vorstellungen folgend entworfen und implementiert.

³siehe Usage of content management systems for websites, http://w3techs.com/technologies/overview/content_management/all

tiert. Die später implementierten Funktionen Sozialer Netzwerke wurden unter Berücksichtigung von Nutzerfeedback ausgewählt. Wichtiger Bestandteil des Entwicklungsprozesses ist der im nächsten Abschnitt beschriebene Nutzertest, dessen Ergebnisse bei der weiteren iterativen Entwicklung berücksichtigt werden sollten.

3.1 Designvorgaben

Das Design des Blogsystems folgt einer Reihe von eigenen Vorstellungen. Zum einen war das Ziel, eine möglichst simple Oberfläche zu bauen, aufbauend aus der Erfahrung bei der Mitarbeit eines bestehenden und etablierten Blogsystems. Gleichzeitig wurde eine Frontendadministration angestrebt: Statt Artikel in einem Adminbackend zu schreiben und dort auch den Blog zu konfigurieren, sollte alles im eigentlichen Blog einstellbar sein. Ein eingeloggtter Admin sieht Schaltflächen, die ein normaler Besucher nicht sieht, insbesondere den Artikeleditor. Mit dieser sofortigen Verfügbarkeit des Editors soll ein einfacherer Workflow beim Schreiben neuer Artikel erreicht werden, indem der Gulf of Execution (Norman [2002]) reduziert wird⁴. Dies entspricht dem typischen Nutzerinterface-Paradigma in bestehenden Sozialen Netzwerken.

3.2 Interviews, Personas

Um den Aspekt des Sozialen Netzwerks zu designen, wurde zuerst ein Kurzfragebogen erstellt und eine kleine Menge potentieller Nutzer per Email befragt. Aus der Analyse ihrer Antworten wurden drei Persona erstellt, welche die mögliche Zielgruppe beschreiben: Blogger, Verweigerer und Facebooker, in Reihenfolge des Zielgruppenfokus. Aus den Antworten gingen auch die Funktionen hervor, die derzeit am meisten benutzt werden: Die soziale Teilhabe am Leben der Freunde durch einen Stream, in dem ihre Aktivitäten gelistet werden, und der Nachrichtenversand. Daher wurden für die erste Ausbaustufe diese Funktionen eingebaut.

3.3 Verwandte Systeme und Skizzen

Für die Gestaltung des Nutzerinterfaces der Funktionen des Systems wurden bestehende Systeme betrachtet und versucht, Konventionen von dort zu übernehmen. So wurde die Gestaltung des Artikeleditors (siehe Abbildung 5) und des Kommentarformulars (siehe Abbildung 6) der Gestaltung derselben in Blogsystemen wie Wordpress und Serendipity angelehnt.

Der Artikeleditor hat zwei Eingabefelder, das obere für den Titel, das untere und größere für den Artikelinhalt. Buttons zwischen den Feldern vereinfachen das Formatieren des Artikels. Die beiden Buttons zum Anzeigen der Vorschau bzw zum Speichern des Artikels sind unten rechts platziert.

⁴Diese Erwartung formuliert beispielweise der Blogger Sascha Lobo unter <http://saschalobo.com/2011/10/16/die-abschaffung-der-ruckseite-des-blogs/>

Title

B *I* 

Abbildung 5: Artikeleditor

Das Kommentarformular hat drei Eingabefelder für die Metadaten, dem Namen des Kommentators, seiner Email-Adresse und einer URL. Nach dem Kommentareignabefeld steht eine Checkbox, um über Folgekommentare benachrichtigt zu werden, wenn die Emailadresse angegeben wurde.

Für die Gestaltung des Streams wurde der Stream von Facebook und Blogplaneten wie ubuntusers.de/Planet betrachtet. Das Nachrichtenmenü orientiert sich an Emailclients und der Buttonlösung von Facebook, die den Empfänger und nicht den bestehenden Nachrichtenstream zum Thema der ersten Nutzerinteraktion macht und damit in den Fokus stellt.

4 Nutzertestplan

In diesem Abschnitt wird der Plan des Nutzertest beschrieben.

4.1 Zweck

Dsnblog in seiner derzeitigen Form ist ein Blogsystem mit den Funktionen eines sozialen Netzwerkes, die in der ersten Nutzerbefragung, mithilfe derer auch die designstützenden Personas erstellt wurden, als die wichtigsten Funktionen erschienen. In diesem Nutzertest soll beobachtet werden, ob Nutzer mit dem System zurechtkommen, wenn sie die grundlegenden Funktionen, wie die Einrichtung des Systems selbst und die Kommunikation mit Freunden, ausführen sollen. Da dies der erste Nutzertest ist, abgesehen von einem ersten explorativen Demolauf mit einem technisch äußerst versierten und mit der Theorie hinter dem System vertrauten Nutzer als Ersttester, ist das Ziel das Finden grober

The image shows a web form for submitting a comment. It consists of the following elements from top to bottom: three input fields labeled 'Name', 'Email', and 'Homepage'; a large, empty text area for the comment; a checkbox labeled 'Notify me for new comments'; a small icon of a hand with a pencil; and a rectangular button labeled 'comment'.

Abbildung 6: Kommentarformular

Schwächen in der Effektivität des Systems, wobei Effektivität *ISO 9241-11* folgend definiert wird. Die Nutzer werden recht deutlich geführt, um wirklich die Hauptfunktionen des Systems zu nutzen, und es soll dabei beobachtet werden, ob sie bei einer dieser grundlegenden Aufgaben Probleme bei der Durchführung haben. Dies würde bedeuten, dass das Design, der Workflow oder die pure Mechanik dieser Funktion noch nicht für die Zielgruppe geeignet ist - und im Idealfall wird durch die Beobachtung deutlich, was genau das Problem verursacht.

4.2 Ablauf

Im Vorhinein hat der Testleiter einen Blog für den Tester auf einem Server vorbereitet, aber die Online-Installation nicht gestartet. Ein anderer Blog, der Demoblog des virtuellen Freundes, ist fertig installiert und hat mindestens einen Artikel. Der Tester wird gebeten, wenn möglich laut zu denken, um die Beobachtung zu erleichtern (Lewis and Rieman [1993]). Der Testleiter bittet den Tester, nacheinander die Aufgaben durchzuführen, indem er jeweils die entsprechende Testanweisung wiedergibt und dann ihre Ausführung beobachtet.

Um den Ablauf des Tests nicht zu sehr zu beeinflussen und dem Experimenters Bias zu begegnen, wird der Testleiter versuchen, nur zu reden wenn nötig. Nur wenn ein Tester nach mehreren Versuchen nicht weiterkommt oder eine Frage so wiederholt, dass die Antwort "Versuch es bitte erstmal" nicht mehr angemessen erscheint, wird er helfend eingreifen.

4.3 Tests

4.3.1 Test 1: Installation

Testeranweisung "Auf www.... steht ein Blog bereit. Bitte gehe auf die Seite, beende die Installation und benenne deinen Blog."

Hintergrund Die Installation ist ein geführter Drei-Schritt-Wizard, der potentiell Probleme haben könnte:

Das genutzte Persona-System (browserid) könnte dem Nutzer zu unbekannt sein, um von ihm bedenken- und problemlos genutzt zu werden. Das Eintragen in dsnns erfordert das Klicken auf einen Link und das erneute Einloggen auf einer anderen Seite, was insbesondere verwirrend sein könnte, wenn die Weiterleitung zurück zur Installationsseite - wie im ersten Test und anders als auf dem System der Entwickler - nicht wie erwartet funktioniert.

Das Testen der Installation könnte zeigen, wie ihr Ablauf klarer gestaltet werden kann, falls das notwendig sein sollte.

Nach der Installation den Blog zu benennen soll prüfen, ob die in die Oberfläche integrierte Administrator-Funktion gefunden und verstanden wird.

4.3.2 Test 2: Artikel

Testeranweisung “Bitte schreibe einen kurzen Artikel. Um sicherzugehen, dass alles funktioniert, kommentiere nun bitte deinen Artikel.”

Hintergrund Bloggen und Kommentieren sind trotz des Social-Network-Hintergrundes die Hauptfunktionen der Seite. Hier erwarte ich keine Probleme, weil das Design stark an bestehende Systeme angelehnt ist, allerdings wäre es umso wichtiger, doch vorhandene zu finden.

4.3.3 Test 3: Freunde verwalten

Testeranweisung “Auf www.... ist ein Blog deines Freundes. Adde ihn als deinen Freund. Werfe nun bitte einen kurzen Blick auf seine Artikel.”

Hintergrund Freunde zu finden und hinzuzufügen ist der Grundstein aller weiteren Social-Network-Funktionen, sowohl der implementierten wie weiteren denkbaren, und gleichzeitig bei einem dezentralen Netzwerk eine größere technische Herausforderung als bei einem zentralen System. Es sind noch keine Funktionen implementiert, die beim Finden von Freunden helfen würden, deswegen kann erstmal nur getestet werden, ob das Hinzufügen eines Freundes mit bekannter Adresse problemlos funktioniert.

Hinter der Anweisung, die Artikel des Freundes zu lesen, verbirgt sich die Frage, ob der Nutzer die Stream-Funktion mit den Artikeln seiner Freunde in seinem eigenen Blog entdeckt, oder ob er die Funktion ignoriert und den Demoblog aufruft.

4.3.4 Test 4: Nachrichten

Testeranweisung “In der Zwischenzeit wurde dir eine Nachricht geschrieben. Bitte lese sie und antworte ihr.”

Hintergrund Hier soll der Testleiter aus dem Demoblog heraus, am besten parallel zum vorherigen Schritt, im Zweifel danach, eine Nachricht an den Blog des Testers schicken. Das Nachrichtensystem ist abgesehen vom Stream die Hauptfunktion, die das System von einem normalen Blog unterscheidet. Es zu Testen soll prüfen, ob Nutzer mit der gebauten Oberfläche zurechtkommen und ob sie die Benachrichtigung wie erwartet sehen und einordnen.

5 Durchgeführte Nutzertests

Der Test wurde mit drei Testnutzern durchgeführt.

5.1 Testergebnisse

Die Tests zeigten ein paar wiederkehrende Probleme, aber auch, dass einige der genutzten Interaktionskonzepte sofort verstanden wurden. Als verwirrend wurden Elemente empfunden, die nicht vollständig gestylt waren und das Persona-System mit der sich öffnenden Nutzerkontenverwaltung nach der ersten Anmeldung. Die sich ähnlich wie in bekannte Systemen gestaltete Artikeleditor und das Kommentaformular wurden dagegen schnell und teilweise spontan verstanden.

5.2 Test 1

Alle Testnutzer konnten die Installation beenden. Bei zwei der Testnutzern war jedoch Unterstützung notwendig, um nach der Nutzerkontenerstellung die Installation fortzusetzen. Die Blogbenennung wurde nur von einem Testnutzer gefunden und konnte genutzt werden, ein zweiter fand sie, bei ihm funktionierte das Eingabefeld jedoch nicht.

5.3 Test 2

Alle Testnutzer konnten einen Artikel schreiben und einen Testkommentar absenden. Bei einem Testnutzer traten Probleme mit verschlucktem Text hinter Umlauten auf, dies konnte jedoch vom Nutzer selbstständig umgangen und der Test trotzdem beendet werden.

5.4 Test 3

Alle Nutzer konnten selbstständig einen Freund hinzufügen. Bei einem Nutzer wurde das Ergebnis nicht korrekt in der Datenbank gespeichert. Der Stream wurde nur von einem Testnutzer bemerkt - die Testnutzer erfüllten den Test, indem sie die Einträge auf dem Demoblog aufriefen.

5.5 Test 4

Dieser Test musste im Testverlauf umgewandelt werden. Aufgrund nicht vorhergesehener Server-Interkommunikationsprobleme konnten Nachrichten nicht

tatsächlich versandt werden. Deshalb wurden entgegen der Testbeschreibung die Testnutzer aufgefordert, selbst eine Nachricht zu verfassen, ohne vorher eine erhalten zu haben.

Es war zweien der Testnutzer möglich, eine Nachricht zu verfassen und abzusenden. Vom dritten Nutzer konnte dieser Test nicht durchgeführt werden, da die Vorbedingung (Freund in Datenbank gespeichert) nicht erfüllt war.

5.6 Testergebnisbeschreibung

5.6.1 Testnutzer 1

Der erste Test mit einer Testperson, die am ehesten dem Verweigerer-Persona entspricht, zeigte eine Reihe von technischen Problemen. Durch eine fehlerhafte Serverkonfiguration konnten Test 3 und 4 nicht erfolgreich durchgeführt werden. Durch einen Fehler bei der Testvorbereitung funktionierte die Installation in Test 1 nicht sofort. Bei den folgenden Tests konnten diese Fehler vermieden werden.

Trotzdem gab es eine Reihe von Beobachtungen, die für die weitere Entwicklung hilfreich sein könnten. Darunter waren auch Stellen, an denen das Design wie gewünscht funktionierte: So wurde der Artikeleditor als solcher identifiziert und auch der Freund-Hinzufügen-Button wurde direkt gesehen und erkannt. Desweiteren wurde die Vorschau-Funktion ohne Aufforderung beim Schreiben des Artikel dazu genutzt, Fehler des Systems (hier: Probleme mit den Umlauten) vorab zu erkennen und den Artikel entsprechen anzupassen. Die Tooltips der grafischen Buttons wurden selbstständig aufgerufen und schienen die Funktion der Buttons klarer zu machen.

Jedoch war der simple Link mit dem Text ädd to dsnnünverständlich - weder wurde der Begriff verstanden, noch wurde der Link als klares Zeichen gedeutet, da jetzt draufklicken zu müssen. Die Blogbenennungsoption wurde nicht eindeutig als solche erkannt und funktionierte im verwendeten Browser Opera auch nicht. Beim Hinzufügen eines Freundes wurde der sehr ungestylte Konfirmationsdialog auf dsnns kritisch betrachtet. Verwirrung verursachte auch Persona, weil es nach dem Bestätigen der Email entgegen anderer Meldung bei der Dateneingabe nicht direkt zur Blogseite weiterleitete, sondern eine Nutzerkontenübersichtsseite öffnete, die gescrollt, neu geladen und von der schließlich zur Hauptseite des Personas-Projekt navigiert wurde, die ohne Eingriff des Testleiters aber nicht in den Installationsablauf eingeordnet werden konnte. Beim Schreiben eines Kommentars verwirrte der Editierenbutton für die Admin-Emailadresse.

Generell für dezentrale Netzwerke und nicht nur für dieses interessant könnte die Reaktion auf den notwendigen erneuten Login auf dem dsnns-System sein. Obwohl keine Verwirrung ob des Modells der verteilten Blogs geäußert wurde, war dieser erneute Login für den Nutzer unerwartet. Dass hier eine neue Seite aufgerufen wurde und deswegen der erneute Login notwendig war, wurde nicht sofort erkannt und es kann davon ausgegangen werden, dass ein weniger technikaffiner Nutzer das gar nicht realisiert hätte.

5.6.2 Testnutzer 2

Die zweite Testperson entsprach am ehesten der Persona des hauptsächlichen Nutzers eines Sozialen Netzwerks, obwohl eine Verweigerungshaltung gegenüber Facebook besteht. Der Test gelang mit weniger technischen Problemen, nur der Nachrichtenversand funktionierte nicht.

Der Tester geriet in ähnliche Probleme wie Testperson 1. Wieder war die Weiterleitung in die Persona-Nutzerverwaltung eine kritische Situation, in der Hilfe nötig war. Die Bestätigungsmeldung auf dsnns wurde als Code verstanden, daher erst nicht gelesen und dann nicht verstanden, schließlich doch bestätigt. Der Stream wurde nicht entdeckt.

Neu war eine kleinere Stolperstelle beim Kommentieren: Als URL wurde erst eine ungültige URL eingegeben und die Fehlermarkierung erst beim Absenden bemerkt, woraufhin diese korrigiert wurde. Und bei der Eingabe der Emailadresse zum Einloggen wurde bestätigungssuchend nachgefragt, dies war also nicht ganz klar oder nicht bedenkenlos.

Der Nachrichtenversand funktionierte technisch nicht. Allerdings wurde, einmal gefunden, das Nachrichtenmenü problemlos durchlaufen. Das Finden war problematisch, weil zu Testbeginn der Tester auf dem Zielblog der Nachricht war, und er zuerst dort nach einer Möglichkeit suchte, die Nachricht abzusen- den. Das führte zu der Frage, ob die Nachricht überhaupt im Browser gesendet werden solle, was deutlich machte, dass die vorhandene Emailadresse und damit die Möglichkeit des Mailversandes bewusst war.

5.6.3 Testnutzer 3

Die dritte Testperson entsprach der Persona des Bloggers, er bezeichnete sich auch selbst als jemand, der "viel netzwerk-tüchtig entsprechend viele Kontakte hat. Teilweise gab er unter diesem Augenmerk Anmerkungen zum Design. Überhaupt hatte Testperson 3 einige Anmerkungen, bemerkte viele Stellen, die ihn störten, formulierte aber auch ein Abschlusssstatement, demnach ihm die Seite sehr gut gefalle.

Bestätigt wurden die Erkenntnisse aus den vorherigen Tests: Der Installer ist besonders beim Add to dsnnsLink nicht klar genug, der Artikeleditor wurde gleich verstanden, ebenso das Kommentarformular und das Nachrichtenmenü. Er formulierte teilweise auch explizit, was in den vorigen Tests schon beobachtet wurde, insbesondere dass der Bestätigungsdialog beim Hinzufügen von Freunden nicht klar wahrgenommen werden würde.

Es gab aber auch Abweichungen: So drückte der Tester aus, dass er die Frontend-Administrationsbutton nach erledigter Aufgabe gerne verschwinden lassen würde. Sie drückten nicht nur die Affordance aus, hier etwas tun zu können (den Blogtitel zu setzen), sondern ihre Präsenz fordere auch dazu auf. Generell führe die Frontend-Administration mit ihren englischen Meldungen zwar zu einem schönen Design, irritiere ihn jedoch im Workflow. Desweiteren wurde in Frage gestellt, warum der Stream öffentlich sei, und die Erwartung formuliert, bei einem Klick auf Stream auch in einem anderen Blog auf den

eigenen zu kommen (dem Verhalten zentraler Sozialer Netzwerke folgend).

6 Fazit

Das Design eines dezentralen Sozialen Networks muss das Problem lösen, dem Nutzer auf verschiedenen Nodes Kontrolle zu geben. Der Wechsel zwischen URLs, was zu erneuten Logins und entsprechenden Meldungen führt, irritierte alle Testnutzer, wenn die Meldungen nicht eindeutig gestaltet waren.

Im Nutzertest erschien es als grundsätzlich unproblematisch, den eigenen und fremde Blogs an eine Emailadresse zu knüpfen, darüber zu identifizieren und das Loginsystem an das Persona-System von Mozilla auszulagern. Jedoch irritierte die Weiterleitung in die Persona-Nutzerkontenverwaltung beim ersten Login, was durch technische Mittel zu vermeiden ist - jedoch auch zeigt, dass der Wechsel zwischen verschiedenen URLs, Diensten und Tabs eine große Hürde beim Erledigen einer Aufgabe sein kann, da sie sogar zu völliger Desorientierung führte.

Von den Testern nicht angemerkt wurde der bisherige völlige Verzicht auf jedwede Form von Sichtbarkeitseinstellungen. Dem Blog-Paradigma folgend waren Artikel immer öffentlich, anders als in Sozialen Netzwerken. Da dies nicht irritierte gehe ich davon aus, dass dies angesichts des blogartigen Nutzerinterfaces erwartet wurde.

Konkrete sich aus den Tests ergebende Änderung ist das Unterbinden der Nutzerweiterleitung auf die Persona-Nutzerkontenweiterleitung. Deutlich wurde auch, dass alle Dialoge und die Installation in ein einheitliches grafisches Design eingebettet werden müssen, um den Nutzer weniger zu irritieren. Zur Vereinfachung der Streamgenerierung angesichts der aufgetretenen Fehler in den Tests wurde `rsspusher`⁵ entwickelt, wodurch das Abfragen der abonnierten RSS-Feeds ausgelagert werden kann.

Um das System weiterzuentwickeln, müssten zuerst die verbliebenen im Test aufgetretenen technischen Problemen - wie die auf dem Testsystem nicht funktionierende Server-Interkommunikation - gelöst werden. Auf der Designseite wäre zu überlegen, wie der Stream gestaltet und so gekennzeichnet werden kann, dass er wahrgenommen wird, und ob er öffentlich sichtbar sein sollte oder nicht. Schließlich müsste das System von der Zielgruppe getestet und validiert werden und mit der Zielgruppe überlegt werden, ob - und wenn ja, welche - weiteren Funktionen integriert werden sollten. Welche Anforderungen bestehen, wenn das System tatsächlich als dezentrales Soziales Netzwerk genutzt werden soll?

⁵<https://github.com/onli/rsspusher>

Literatur

- Rebecca Blood. Weblogs: A history and perspectives, 2000. URL http://www.rebeccablood.net/essays/weblog_history.html.
- Danah M. Boyd and Nicole B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007. ISSN 1083-6101. doi: 10.1111/j.1083-6101.2007.00393.x. URL <http://dx.doi.org/10.1111/j.1083-6101.2007.00393.x>.
- Heinz Karsten Hiekmann. Dsnns - name service architecture component of distributed systems. 2013.
- Clayton Lewis and John Rieman. Task-Centered User Interface Design: A Practical Introduction, 1993. URL http://grouplab.cpsc.ucalgary.ca/saul/hci_topics/tcsd-book/chap-1_v-1.html.
- Donald A. Norman. *The Design of Everyday Things*. Basic Books, 2002. ISBN 9780465067107.