Sponti – Spontanhelfer-Teamorganisation

Schlussbericht

Zuwendungsempfänger:

Alexander Blum, Angelika Ehrlich, Thomas Mielke, Sarah Stoffels und Nils Witt

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS22S18 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Im Vorhaben Sponti sollte ein Portal realisiert werden zur Registrierung und Verwaltung von potenziellen Helfenden in Krisensituationen für Organisationen, Vereine oder andere Gruppen. Dabei sollten bestimmte Fähigkeiten und Materialien, die sie für verschiedenartige Einsätze zur Verfügung stellen können, mit aufgenommen werden. In konkreten Einsatzfällen sollen dann die Details kurzfristig kommuniziert und die Personen eingeladen werden können.

Meilensteine

- Implementierung der Verwaltung von Gruppen und Material im Portal
- Implementierung einer Terminverwaltung mit Rückmeldungs-Kanälen und -Auswertungen
- Evaluierung von datenschutzkonformen Messenger-Lösungen (z.B. Matrix-Server)
- Integration des Messengers in das bestehende Portal
- Bereitstellung der Serverinhalte als API
- Bereitstellung der User-Interfaces als App mit Push-Funktion ("progressive web app" oder native Apps)
- nach Möglichkeit auch Anbindung eines freien Pushservices (unifiedpush.org - im Gegensatz zu den kommerziellen proprietären Push-Services von Apple und Google)
- Ggf. Anbindung weiterer Kommunikationswege (SMS o.ä.)
- Ggf. Anbindung von Social-Media-Kanälen

Beitrag des Projektes zu den Zielen der Förderinitiative „Software-Sprint“

Es hat sich beim Einsatz der Vorläuferversion dieser Fördersoftware bei der Nutzung zur Einsatzkoordination gezeigt, dass es keine fertige Open Source-Lösung gibt, um sowohl Organisatoren, als auch Teilnehmern sehr kurzfristig verschiedene Kanäle zum Austausch und zur Organisation zu ermöglichen. Daher ist es ein Kernziel des Vorhabens, eine solche Lösung aus Open Source-Komponenten datenschutzfreundlich zu bauen, die (selbst-)organisierte Gruppen selbst betreiben oder durch einen unabhängigen Hoster betreiben lassen können, sowie bei Bedarf anpassen und erweitern können.

Zudem kann diese Software nicht nur von Organisationen sondern auch von „losen Gruppen“ ohne feste Struktur zu deren Koordination von z.B. Hilfsaktionen verwendet werden und unterstützt somit „Civic Tech“.

Ausführliche Darstellung der Ergebnisse

Technologien und Aufbau

Für das Backend wurde wie bereits in der Vorgängersoftware das Python-Framework Django verwendet, die Datenstruktur allerdings sehr ausgeweitet.

Zur Bereitstellung von API-Funktionen wurde GraphQL angebunden und für die API-Authentifizierung JSON Web Token (JWT) implementiert.

Für die Datenhaltung kommen Postgresql und Redis zum Einsatz. Mailhog wird für das E-Mail-Handling zu Testzwecken verwendet.

Frontends wurden mit React und MAUI umgesetzt und sind unabhängig voneinander verwendbar. Sie kommunizieren über die API-Funktionen mit dem Backend.

Zum Deployment der Anwendungen auf Servern wurden Automatisierungs-Skripte für die Container-Technologie Docker erstellt und jeder Serverdienst als separater Container angelegt.

Beschreibung der typischen Use Cases und Abläufe

Zunächst muss in der Software von administrativer Seite eine Organisation angelegt werden. In diesem Bereich kann dann die zuständige Organisation Projekte (Abbildung rechts) und Einsätze anlegen.

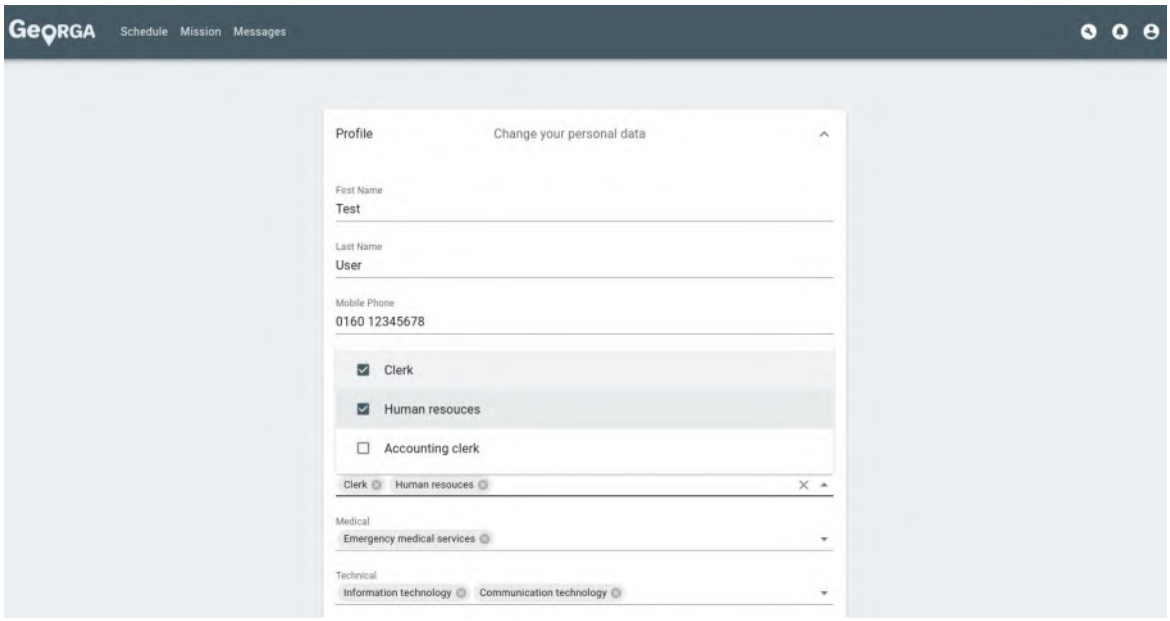


Native App - Projektliste

Nutzer können sich registrieren und ihre persönlichen Details eingeben sowie später editieren (Abbildung unten).

Personen können sich auch für mehrere Organisationen in der gleichen App anmelden, sofern in der Softwareinstanz mehrere Organisationen serverseitig angelegt wurden.

Einsätze enthalten weitere Details, u.a. gewünschte Tätigkeitsrollen, deren konkrete Aufgaben, Einsatzschichten und einen oder mehrere hinterlegte Einsatzorte. Die registrierten Personen, die sich für die Teilnahme an (Hilfs-)einsätzen interessieren, können dann konfigurieren, welche Projekte und Einsätzen sie in ihrer Weboberfläche oder App sehen wollen. Ebenso lässt sich einstellen, dass sie regelmäßige Anfragen dafür erhalten und über welche Kommunikationskanäle diese Anfragen erfolgen sollen.



React Frontend - Ansicht zur Bearbeitung des Nutzerprofils

Erreichung der Meilensteine

Wir konnten somit folgende Meilensteine erarbeiten:

- Verwaltung von Gruppen und Material im Portal: Materialverwaltung möglich
- Terminverwaltung mit Rückmeldungs-Kanälen und -Auswertungen: **ausführlich**, einige Aspekte im Frontend noch zu implementieren
- Evaluierung von datenschutzkonformen Messenger-Lösungen (z.B. Matrix-Server): **Konzept** mit Fokus auf Matrix zur Erstellung von kurzzeitigen Chaträumen passend zu Einsätzen und deren Teilnehmern
- Bereitstellung der Serverinhalte als API: **ausführlich** mit GraphQL
- Bereitstellung der User-Interfaces als App mit Push-Funktion ("progressive web app" oder native Apps): React Frontend erstellt (auch als PWA nutzbar), native App wurde mit MAUI erstellt
- Anbindung eines freien Pushservices: **anzubinden** über Channel-Datenstruktur unserer Software
- Anbindung weiterer Kommunikationswege (SMS o.ä.): **anzubinden** über Channel-Datenstruktur unserer Software

Die Integration des Messengers in das bestehende Portal sowie Anbindung an Social Media hätten leider den möglichen Zeitrahmen und Aufwand gesprengt.

Erkenntnisse aus der Projektarbeit

Als Backend für die Software verwenden wir Django. Dies ist ein etabliertes und gut gepflegtes Framework, jedoch sind wir bei unserer Implementierung an einigen Stellen an Grenzen gestoßen, an denen benötigte Funktionen manuelle Anpassungen erforderten. Dies betrifft eine Änderung der Subscription-Library nach Upgrade auf Django 4.1 und eigene Implementierung eines Rechtesystems, das auch auf Datenobjektebene (statt nur Klassenebene) Zuordnungen ermöglicht. Zudem haben wir die Erweiterung „graphene-django“ für die Anbindung von GraphQL angepasst, u.a.: zur Verwendung unterschiedlicher IDs für Dateneinträge per API und Datenbank-intern (aus Security-Gründen), Logging, „paging/slicing“ von Listen mit Einhaltung der Spezifikation von „Global Relay API“.

Die Begleitung durch die Open Knowledge Foundation hat uns u.a. Coachings zum Thema UI/UX ermöglicht und somit noch mehr Aspekte aufgezeigt, die es den Benutzern einfacher machen können, die Thematik und Möglichkeiten der Plattform und App unkompliziert zu erfassen. Ein Aspekt wäre, bei möglicher Zusammenarbeit mit Organisationen speziell auf diese zugeschnittene In-App-Tutorials zu integrieren, die den Benutzern ihre Möglichkeiten erläutern. Ebenso wären Integrationen in Social-Media-Systeme gut zur Steigerung des Bekanntheitsgrads und für die Benutzer ein weiterer Motivationsaspekt (z.B. wenn Teilnahme an Einsätzen im eigenen Social-Media-Account über die App kundgetan werden kann).

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Der Fokus der im Vorhaben entwickelten Software lag im Gegensatz zur Vorgängerversion nicht auf einem speziellen Einsatzszenario des derzeitigen Nutzers sondern wurde generalisierbar für verschiedene Arten von Organisationen und möglichen Einsätzen durchdacht. Somit können auch Gruppen mit weniger Strukturen als z.B. eine Hilfsorganisation diese Software zur Koordination verwenden.

Durch den modularen Aufbau und generalisierte Schnittstellen können verschiedene Kommunikationssysteme, je nach Bedarf der nutzenden Gruppe, angebunden werden. Die Benachrichtigungen („Message“) auf verschiedenen Ebenen sind in der Datenstruktur generalisiert, können aber an verschiedene „Channels“ wie In-App-Notifications, Push-Nachrichten, E-Mail, Chat und SMS verteilt werden.

Das Projekt kann von den nutzenden Gruppen selbst gehostet und erweitert werden, wir können diesen bei Bedarf aber auch die technische Bereitstellung anbieten.

Mögliche Weiterentwicklungen betreffen genauere Anpassung von Features auf spezielle Organisationen aber auch generalisierbare Schnittstellen z.B. zu verschiedenen bekannten Kommunikationssystemen. Aktuell finden unsererseits Gespräche mit Organisationen zur Nutzung der Software statt, die recht flache Hierarchien haben und unserer Hoffnung nach gut zur Open-Source-Community passen.

Für unsere persönliche fachliche Weiterentwicklung war das Projekt sehr geeignet, da es sich über verschiedene Technologien und Ebenen (Frontend, Backend, native App) erstreckt und an bestehende Open-Source-Projekte anzubinden ist.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Solito als übergreifendes Frontend-Framework

Der Ansatz, mit Solito Aufwand und Code durch ein übergreifendes Frontend-Framework zu sparen, war leider nicht vielversprechend, da geplante Features wie z.B. Responsive UI für Web viel umständlicher umzusetzen waren, als wenn man sie ohne Solito in React implementiert. Auch ist das Solito-Projekt bisher im Wesentlichen von einem einzigen Entwickler abhängig im Gegesatz zu React.

Rechtliche Diskussionen und Vermittlungsprobleme bzgl. Open-Source-Lizenzvorteile

Mit der Organisation (im Folgenden „HiOrg“ genannt), die die Vorversion der hier geförderten Software genutzt hat, war zunächst geplant, mögliche fachliche Anforderungen als Musterbeispiel für Hilfsorganisationen für das Förderprojekt zu erarbeiten.

Inzwischen hatte jedoch das Thema Spontanhelferkoordination allgemein an „Prominenz“ gewonnen und es gibt nun im Umfeld der typischen Hilfsorganisationen einige, die eine dafür nutzbare Software als „Software as a Service“ mit proprietär entwickelter Software vertreiben. Dementsprechend gab es nun auch bei der HiOrg einen solchen Plan und daher wurde innerhalb der HiOrg die Zusammenarbeit mit uns zu fachlichen Definitionen nur unter der Bedingung erlaubt, dass die daraus entstehende Software exklusiv und mit alleinigem Wiederverkaufsrecht für die HiOrg entwickelt worden wäre, was der ursprünglichen Entwicklung als Open-Source-Software allerdings entgegensteht und mit dem Förderprojekt keinesfalls vereinbar ist.

Insgesamt gab es bei der HiOrg durchaus viele Befürworter der Zusammenarbeit unter den gegebenen Förderbedingungen, jedoch entschied sich der Vorstand dagegen.

Es gab auch an überregionaler Stelle dieser HiOrg Interesse, sich die ursprüngliche Version und später die geförderte Software anzuschauen, da man dort vor einem Kaufentscheidungsprozess für eine solche Software stand, allerdings wurde uns keine detailliertere Kommunikation mehr mit dieser Stelle erlaubt, da die HiOrg selbst ihrer überregionalen Stelle diese Software unter ihrem eigenem Namen verkaufen wollte und die Personen unserer GbR als Subunternehmer oder Angestellte unter Vertrag nehmen wollte.

Dies wurde durch eine Person der HiOrg sogar in einem persönlichen Gespräch mit einer Person unserer GbR durch Druck und Androhung von rechtlichen Schritten versucht, zu bewirken, sowie dass wir das Förderprojekt abbrechen sollten.

Allerdings gab es zwischen HiOrg und den Personen unserer GbR nie einen Arbeits- oder Auftragsvertrag oder eine Vergütung und solche Ansprüche sind nicht von einer ehrenamtlichen Vereinsmitgliedschaft gedeckt. Da wir uns vor allem wegen der offenen Lizenz und mit den Plänen einer Nutzbarkeit der Software für die Allgemeinheit für den Prototype Fund beworben hatten, war hier nun eine Zusammenarbeit mit der HiOrg unsererseits ausgeschlossen.

Insgesamt zeichnen unsere bisherigen Erkenntnisse nicht das Bild, dass von den klassischen Hilfsorganisationen in Deutschland eine übergreifende Zusammenarbeit an Software durch die Mitwirkung an und Investition in Open-Source-Projekte gewünscht ist, allerdings sind die möglichen Vorteile nach unserem Eindruck dort auch noch zu wenig bekannt.

Das gleiche Problem sehen wir bezüglich agiler Softwareentwicklung und Beratungsverträgen, die für eine Zusammenarbeit zwischen Open-Source-Projekt und Kunde an anderen Stellen gängige Praxis ist. In den klassischen Strukturen herrscht allerdings meist die Ausschreibungspraxis vor, in der eine Software zum Vergleich eine unveränderliche Feature-Liste und einen einmaligen Fixpreis haben muss und anderenfalls nicht in der Auswahl berücksichtig wird.

Daher braucht es unserer Meinung nach noch etwas Arbeit, Open-Source-Projekte und Vertragstypen für agile Softwareentwicklung überhaupt in diesem Umfeld bekanntzumachen. Es wäre eine gute Vorbildfunktion, wenn bald allgemein bei Behörden, behördennahen Institutionen und steuerlich oder durch Spenden finanzierte Organisationen mehr Priorität auf Investitionen in Open-Source-Projekte gelegt würde, damit Budgets nicht mehr mehrfach für identische Programmieraufwände und Software ausgegeben werden.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Projekhomepage:

https://www.georga.app

GitHub:

https://github.com/georga-app/georga-server-django

https://github.com/georga-app/georga-client-react

https://github.com/georga-app/GeorgaMobileClient

https://github.com/georga-app/georga-docker

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Arbeitspaket	Geplante Stunden	Geleistete Stunden
Softwareentwicklung und Test (Backend, DB, API, Schnittstellen)	400	393,75
Softwareentwicklung und Test (Frontend, Webseite)	120	105,75
Softwareentwicklung und Test (Frontend - speziell App)	180	250,5
Systemadministration	40	29,75
Geschäftsführung & Organisatorisches	50	60
Projektleitung	40	65,25
Anforderungsanalyse	80	150,80
Öffentlichkeitsarbeit	42	11,80
Summe	**952**	**1067,6**

Die verbrauchten Stunden stimmen grob mit der Planung überein, außer dass wir bei der App-Entwicklung und Anforderungsanalyse noch mehr Zeit verbraucht haben.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Es gab keine Entwicklungen anderer Personen oder Institutionen, die Einfluss auf unsere Arbeiten oder Zielsetzungen hatten. Allerdings konnten wir beobachten, dass das Thema Spontanhelfer-Koordinationssoftware bei Hilfsorganisationen an Bedeutung gewinnt und sogar Softwareprodukte dafür entwickelt werden, jedoch alle ohne offenen Code und oft in Kombination mit weiteren Notfallrettungs-Funktionen (z.B. https://www.retter4u.app/idee-konzept/). Nach unserem Wissen ist bisher keine alternative Open-Source-Lösung erschienen.