Publikation

“SAVE THE WORLD” - BUTTON

MASTER THESIS MAS-SE 2016, HSR

Autoren	Stefan Walser; Simon Kuppelwieser
Version	1.0
Status	Freigegeben
Pfad	https://d.docs.live.net/2b2c11cc3e12b959/Dokumente/HSR/4. Semester/Masterarbeit/02_Arbeit/1_Dokumente/1_Projektdokumente/80_Publikation.docx
Letzte Aktualisierung	22.08.2018

Version	Datum	Signatur	Änderung
0.0	17.08.2018	Stefan Walser	Ersterstellung
0.0	19.08.2018	Stefan Walser	Diverse Kapitel ergänzt
0.0	20.08.2018	Simon Kuppelwieser	Kapitel realisierte Lösung erstellt
1.0	22.08.2018	Simon Kuppelwieser	Freigabe

Abstract

Ziel der Arbeit war die Entwicklung einer Webapplikation, auf welcher sich potentielle Spender mit gemeinnützigen Organisationen in Verbindung bringen können, um diese finanziell zu unterstützen. Im Fokus stand dabei die Einfachheit des Spendenvorgangs, welcher durch einen Internet of Things (IoT)-Button ausgelöst wird. Dazu personalisiert ein Spender seinen IoT-Button über die Plattform mit der Zielorganisation, dem gewünschten Spendenbetrag und seinen Spendenlimiten. Eine simple Betätigung des Buttons genügt anschliessend um eine Spende abzusetzen.

Das Produkt der Arbeit besteht im Wesentlichen aus den drei Teilen: «Frontend, Backend und IoT-Button». Im Mittelpunkt der Entwicklung stand eine gute Wartbarkeit, Testbarkeit und Erweiterbarkeit. Deshalb wurde die Applikation konsequenterweise in ein Frontend und ein Backend unterteilt. Des Weiteren wurde stark auf Dependency Inversion und auf Dependency Injection gesetzt. Das Frontend wurde in C# programmiert und als ASP.NET Core 2.0 Webapplikation mit MVC Pattern realisiert. Das Backend wurde ebenfalls in C# programmiert und als ein ASP.NET Core 2.0 basierendes REST API umgesetzt. Die Struktur im Backend stützt sich auf den Clean Architecture Ansatz und setzt vermehrt auf asynchrone Prozesse. Sowohl Frontend als auch Backend werden auf einem Applikationsserver in der Microsoft Azure Cloud gehostet. Als Basis für den IoT-Button wurde ein ESP-12F Mikrokontroller der ESP8266 Familie mit Arduino Bootloader verwendet. Die Software wurde in C++ programmiert.

Da bei einer Spende eine Zahlungsabwicklung getätigt wird, durfte die Kommunikationssicherheit nicht vernachlässigt werden. Eine Verschlüsselung der Kommunikation zwischen IoT-Button und REST API auf Basis eines Challenge Response-Verfahrens mit Secret Key stellt sicher, dass nur eine autorisierte Spende ausgeführt werden kann und schützt gleichzeitig vor Replay Attacken.

Die im Rahmen der Arbeit gestellten Anforderungen konnten erfüllt werden und mit der entwickelten Lösung wurde ein Proof of Concept fertiggestellt. Für eine kommerzielle Verwendung des Produkts müsste allerdings weitere Zeit in die Entwicklung investiert werden. Unter anderem muss ein realer Payment Provider eingebunden, die Button-Software um eine Update Schnittstelle erweitert, sowie die Hardware (Gehäuse etc.) optimiert und für die Massenproduktionsfähigkeit getrimmt werden.

Management Summary

Ausgangslage und Projektziele

Ziel der Arbeit war die Entwicklung einer Webapplikation (EasyDonate), auf welcher sich potentielle Spender mit gemeinnützigen Organisationen in Verbindung bringen können, um diese finanziell zu unterstützen. Im Fokus stand dabei die Einfachheit des Spendenvorgangs, welcher durch einen Internet of Things (IoT)-Button ausgelöst wird.

Neben den Funktionalen Anforderungen, stand eine gute Wartbarkeit, Testbarkeit und Erweiterbarkeit im Mittelpunkt der Entwicklung. Folglich wurde viel Zeit in das Design der Architektur investiert.

Vorgehen und Technologien

Das Projekt wurde nach einem agilen, iterativen Ansatz umgesetzt. Der Entwicklungsprozess setzt auf eine Continuous Deployment Pipeline. Durch die Automatisierung des Deployments konnte der administrative Aufwand minimiert und die Agilität optimiert werden. Die Pipeline wurde komplett in Visual Studio Team Services (VSTS) umgesetzt.

Die Webapplikation wurde in C# programmiert und als ASP.NET Core Applikation umgesetzt. Für das Hosting der Applikation wird auf die Microsoft Azure Cloud gesetzt. Die Software des IoT-Buttons wurde in C++ programmiert und nutzt einen Arduino Bootloader. Der IoT-Button basiert auf einem ESP-12F Mikrokontroller der ESP8266 Familie.

Ergebnis

Anhand der erarbeiteten Anforderungen wurde ein Prototyp der EasyDonate Webapplikation und des dazugehörigen IoT-Buttons entwickelt.

Die EasyDonate Webapplikation besteht dabei aus den beiden Komponenten Frontend und Backend. Das Frontend wurde in C# programmiert und als ASP.NET Core 2.0 Webapplikation mit MVC Pattern realisiert. Das Backend wurde ebenfalls in C# programmiert und als ein ASP.NET Core 2.0 basierendes REST API umgesetzt. Die Struktur im Backend stützt sich auf den Clean Architecture Ansatz und setzt vermehrt auf asynchrone Prozesse. Sowohl Frontend als auch Backend werden auf einem Applikationsserver in der Microsoft Azure Cloud gehostet. Als Basis für den IoT-Button wurde ein ESP-12F Mikrokontroller der ESP8266 Familie mit Arduino Bootloader verwendet. Die Software wurde in C++ programmiert.

Über die EasyDonate Plattform können sich Benutzer als Organisation oder als Spender registrieren. Als Organisation kann das öffentliche Profil erstellt und verwaltet werden. Spender können ihr Spendenprofil und ihren persönlichen IoT-Button konfigurieren. Über den IoT-Button kann eine Spende ausgelöst werden. Administratoren können Mitgliedschaftsanfragen von Organisationen verwalten und neue Administratoren anlegen. Als Gast können die Organisationsprofile, sowie diverse Statistiken über den Betrieb der Plattform eingesehen werden.

Inhaltsverzeichnis

1 Einführung

1.1 Zweck

Dieses Dokument beschreibt das Projekt „Save The World – Button“ und das dabei entwickelte Produkt EasyDonate. Das Projekt wird von der Idee bis zur Umsetzung vorgestellt. Es wird auf Details der Anforderungsanalyse, Architektur und der realisierten Lösung eingegangen.

1.2 Rechtliche Hinweise

Das nachfolgend beschrieben Projekt wurde komplett im Rahmen der Masterarbeit «Save The World» -Button für den MAS-SE 2016 an der HSR realisiert. Die Urheber- und Nutzungsrechte bleiben bei den Autoren. Die HSR hat aber das Recht, die Masterarbeit weiter zu verwenden, insbesondere den Studierenden als Beispiel und Grundlage für weitere Arbeiten abzugeben. Alle in der Applikation verwendeten Bildquellen stehen unter der Creative Commons CC0 Lizenz zur freien kommerziellen Verwendung zur Verfügung.

2 Definitionen und Glossar

Begriff	Bedeutung
Azure	Cloud Service von Microsoft für das Hosting von Applikationen, Servern etc.
DIF	Dependency Injection Framework
Jwt	Ein JSON Web Token (JWT) ist ein auf JSON basiertes Access-Token. Das JWT ermöglicht den Austausch von verifizierbaren Claims.
MVC	Model-View-Controller Pattern
Nonce	Abkürzung für „used only once“ oder „number used once“
NPO	Non-Profit-Organisation, auch als Organisation bezeichnet
ORM	Object-Relationales Mapping
POCO	Plain Old C# Object (normales C# Objekt)
REST	Representational State Transfer
SQL	Structured Query Language
SRP	Single Responsibility Principle
VSTS	Visual Studio Team Services
OTA	Over the air update
SPIFFS	SPI Flash File System
HMAC	Keyed-Hash Message Authentication Code

Tabelle 1 Definitionen und Glossar

3 Anforderungsanalyse

In den nachfolgenden Kapiteln wird das Resultat der Anforderungsanalyse vorgestellt. Es wird der Systemkontext, die Use Cases, die nicht funktionalen Anforderungen und das grundlegende Domänenmodell präsentiert.

3.1 Produkt Funktion

Für einen Spender welcher eine gemeinnützige Organisation finanziell unterstützen möchte ist der „Save The World – Button“ ein IoT-Button mit Anbindung an ein Online Spenden Portal. Über das Online Portal kann konfiguriert werden welcher Betrag der Spender durch das betätigen des IoT-Button an welche Organisation spenden möchte.

Das Online Portal soll eine Übersicht bieten über die gemeinnützigen Organisationen welchen gespendet werden kann und Spendenstatistiken anzeigen können.

3.2 Systemkontext

Im nachfolgenden Diagramm wird das System sowie die externen Akteure und die Umsysteme dargestellt. Die Verbindungslinien zeigen die Kommunikationswege.

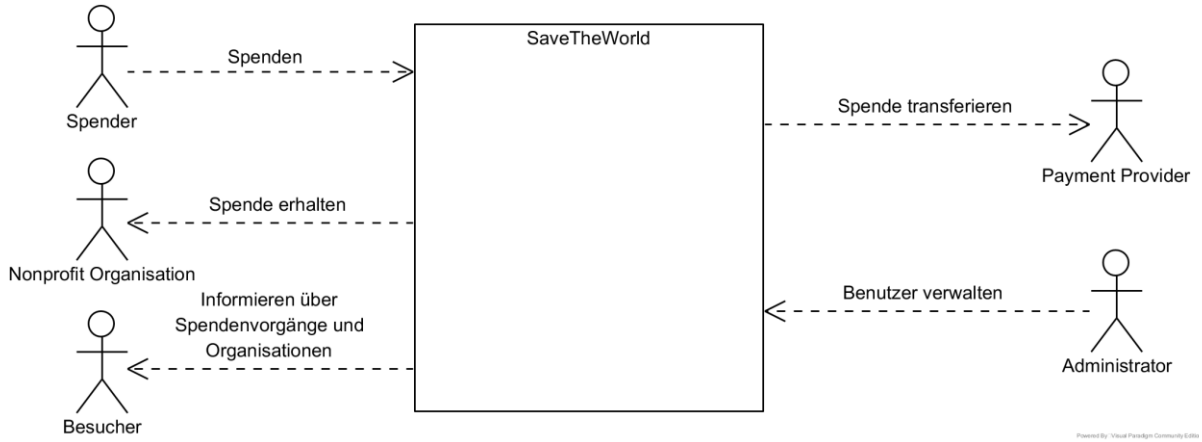

Abbildung 1 Systemkontextdiagramm SaveTheWorld

3.2.1 Akteure

3.2.1.1 Spender

Der Spender ist ein Endanwender der Applikation. Er möchte über die Applikation eine gemeinnützige Organisation durch Spenden unterstützen. Zudem möchte er sich über gemeinnützige Organisationen informieren und aktuelle Spende Statistiken einsehen.

3.2.1.2 Non-Profit-Organisation

Die Non-Profit-Organisation (nachfolgend auch Organisation genannt) ist der zweite Endanwender der Applikation. Ihr Ziel ist es sich selbst über die Plattform zu präsentieren und dadurch Spender zu mobilisieren.

3.2.1.3 Administrator

Der Administrator ist verantwortlich für die Verwaltung der Spender und der Non-Profit-Organisationen und erteilt den Non-Profit-Organisationen eine Mitgliedschaft.

3.2.1.4 Besucher

Der Besucher (nachfolgend auch Gast genannt) ist eine Person, welche sich auf der Plattform befindet ohne dabei eingeloggt zu sein.

Der Besucher kann durch unterschiedliche Beweggründe auf die Plattform gelangt sein.

- Zufällig auf die Plattform gestossen
- Möchte sich über Organisationen informieren
- Möchte sehen was gespendet wird.
- Möchte als Spender oder Organisation beitreten
- Ist bereits registriert und möchte sich einloggen

3.2.1.5 Payment Provider

Die Zahlungsabwicklung erfolgt über einen externen Payment Provider, welcher nicht teil des Systems ist, aber sich innerhalb des Systemkontextes befindet.

3.2.2 Einschränkungen

Als Payment Provider wird im Rahmen der Projekt Arbeit ein Dummy Payment Provider verwendet. Dieser soll einfach durch einen echten Payment Provider ausgetauscht werden können.

3.3 Anwendungsfälle

In den nachfolgenden Kapiteln werden die essentiellen Anwendungsfälle anhand kurzer Beschreibungen genauer spezifiziert. Alle Anwendungsfälle sind aus der Sicht des Systems formuliert.

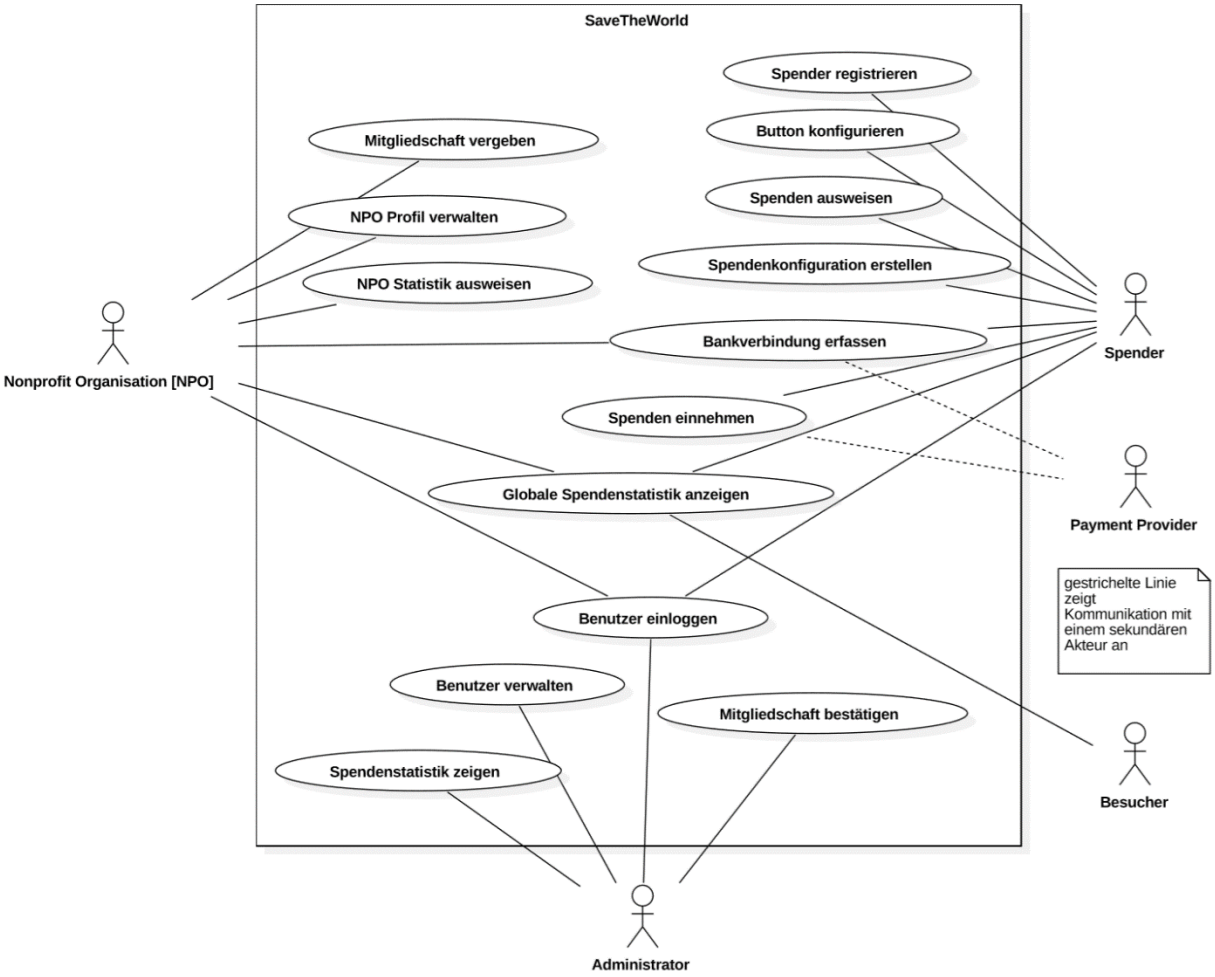

Abbildung 2 Übersicht Anwendungsfälle

3.3.1 Spender bezogene Anwendungsfälle

3.3.1.1 UC-001: Spenden einnehmen

Motivation	Als Spender möchte ich eine Gemeinnützige Organisation mit einer Spende unterstützen
Auslöserin	Spender
Auslösende Nachricht	Spende einnehmen(Spender, NPO, Spendenbetrag)
Reaktionen	Bekanntgabe des Ausführungsstatus (erfolgreich/ nicht erfolgreich)
Kurzfassung des Ablaufs	Ein Spender spendet mit Spende einnehmen() einen Spendenbetrag an eine NPO. Dem Spender wird mitgeteilt ob die Spende erfolgreich ausgeführt werden konnte.

Tabelle 2 UC-001: Spenden einnehmen

3.3.1.2 UC-002: Spender registrieren

Motivation	Als Spender möchte ich mich registrieren um Gemeinnützige Organisationen mit einer Spende unterstützen zu können
Auslöserin	Spender
Auslösende Nachricht	Spender registrieren (Spender)
Reaktionen	Bestätigungsemail bei erfolgreicher Registration
Kurzfassung des Ablaufs	Ein Spender registriert sich mit Spender Registrieren (). Wenn die Registrierung erfolgreich ausgeführt werden konnte wird der Spender durch ein Bestätigungsmail informiert.

Tabelle 3 UC-002: Spender registrieren

3.3.1.3 UC-003: Button konfigurieren

Motivation	Als Spender möchte ich meinen persönlichen IoT Button mit dem lokalen WiFi Netzwerk verknüpfen und so konfigurieren können, dass bei Betätigung des Buttons, an die von mir ausgewählte Organisation gespendet wird.
Auslöserin	Spender
Auslösende Nachricht	Button konfigurieren (Identifikation, Secret)
Reaktionen	-
Kurzfassung des Ablaufs	Ein Spender konfiguriert seinen persönlichen IoT-Button durch Button konfigurieren (). Dabei kann das Wifi Netzwerk eingestellt und die Identifikation sowie das Secret (private key) hinterlegt werden. Wenn die Konfiguration erfolgreich war, werden die Daten persistent auf dem Button gespeichert.

Tabelle 4 UC-003: Button konfigurieren

3.3.1.4 UC-005: Spendenkonfiguration erstellen

Motivation	Als Spender möchte ich konfigurieren können an welches Unternehmen welcher Betrag gespendet wird bei Betätigen des Buttons.
Auslöserin	Spender
Auslösende Nachricht	Spendenkonfiguration erstellen (NPO, Betrag)
Reaktionen	Bekanntgabe des Erstellungsstatus
Kurzfassung des Ablaufs	Ein Spender erstellt mit Spendenkonfiguration erstellen() seine Spendenkonfiguration. Dabei gibt er die gewünschte Gemeinnützige Organisation an sowie der Spendenbetrag pro Knopfdruck. Zusätzliche Parameter wie minimale Zeit zwischen 2 Spende Vorgängen und maximaler Betrag pro Zeiteinheit können erfasst werden. Dem Spender wird mitgeteilt ob die Spendenkonfiguration erfolgreich erstellt werden konnte.

Tabelle 5 UC-005: Spendenkonfiguration erstellen

3.3.1.5 UC-006: Spenden ausweisen

Motivation	Als Spender möchte ich meine getätigten Spenden einsehen.
Auslöserin	Spender
Auslösende Nachricht	Spenden ausweisen (Zeitraum)
Reaktionen	Übersicht über getätigte Spenden anzeigen
Kurzfassung des Ablaufs	Ein Spender kann mit Spenden ausweisen() einsehen welche Spenden er bereits getätigt hat.

Tabelle 6 UC-006: Spenden ausweisen

3.3.2 NPO bezogene Anwendungsfälle

3.3.2.1 UC-007: Mitgliedschaft vergeben

Motivation	Als Gemeinnützige Organisation möchte ich Mitglied werden, damit ich Spenden erhalten kann.
Auslöserin	NPO
Auslösende Nachricht	Mitgliedschaft vergeben (Organisationsinformationen)
Reaktionen	Bekanntgabe das der Antrag zur Aufnahme auf der Plattform erfolgreich eingegangen ist.
Kurzfassung des Ablaufs	Eine Gemeinnützige Organisation kann mit Mitgliedschaft vergeben() eine Mitgliedschaft beantragen um Spenden zu erhalten. Die Organisation wird benachrichtigt ob sie erfolgreich als Mitglied aufgenommen wurde oder nicht, sobald ein Administrator mit Mitgliedschaft bestätigen() die Mitgliedschaft bestätigt.

Tabelle 7 UC-007: Mitgliedschaft vergeben

3.3.2.2 UC-008: NPO Profil verwalten

Motivation	Als Gemeinnützige Organisation möchte ich mein öffentliches Profil pflegen und verwalten.
Auslöserin	NPO
Auslösende Nachricht	NPO Profil verwalten (Mitgliednummer)
Reaktionen	Bekanntgeben ob Profiländerungen erfolgreich gespeichert werden konnten oder nicht.
Kurzfassung des Ablaufs	Eine Gemeinnützige Organisation kann mit NPO Profil verwalten(), ihr öffentlich, auf der Plattform einsehbares Profil bearbeiten. Nach erfolgter Bearbeitung wird der Organisation mitgeteilt ob die Änderungen erfolgreich gespeichert wurden konnten oder nicht.

Tabelle 8 UC-008: NPO Profil verwalten

3.3.2.3 UC-009: NPO Statistik ausweisen

Motivation	Als Gemeinnützige Organisation möchte ich meine erhaltenen Spenden einsehen.
Auslöserin	NPO
Auslösende Nachricht	NPO Statistik ausweisen (Zeitraum)
Reaktionen	Übersicht über erhaltene Spenden anzeigen
Kurzfassung des Ablaufs	Eine Gemeinnützige Organisation kann mit NPO Statistik ausweisen() ihre erhaltenen Spenden einsehen.

Tabelle 9 UC-009: NPO Statistik ausweisen

3.3.3 Allgemeine Anwendungsfälle

3.3.3.1 UC-010: Globale Spendenstatistik anzeigen

Motivation	Als Besucher möchte ich Statistiken über die globalen Spendenaktivitäten einsehen.
Auslöserin	Besucher, Spender, NPO, Administrator
Auslösende Nachricht	Globale Spendenstatistik anzeigen()
Reaktionen	Anzeige von globalen Spendenstatistiken wie: Gesamthaft gespendeter Betrag Spender mit den meisten Spenden NPO mit den meisten erhaltenen Spenden
Kurzfassung des Ablaufs	Ein Besucher kann mit Globale Spendenstatistik anzeigen() diverse Statistiken über die globalen Spendenaktivitäten einsehen.

Tabelle 10 UC-010: Globale Spendenstatistik anzeigen

3.3.3.2 UC-011: Benutzer einloggen

Motivation	Als Benutzer, welcher unterschiedliche Rollen wie Spender, NPO oder Administrator haben kann, möchte ich mich einloggen können um Zugang zu den für mich relevanten Funktionen im System zu erhalten.
Auslöserin	Spender, NPO, Administrator
Auslösende Nachricht	Benutzer einloggen(Benutzername, Passwort)
Reaktionen	Bekannt geben ob die Benutzeranmeldeinformationen korrekt sind oder nicht.
Kurzfassung des Ablaufs	Ein Benutzer kann sich mit Benutzer einloggen() auf der Plattform Authentisieren. Bei erfolgreichem Login erhält der Benutzer Zugriff auf seine persönlichen Einstellungen.

Tabelle 11 UC-011: Benutzer einloggen

3.3.3.3 UC-004: Bankverbindung erfassen

Motivation	Als Spender und als NPO möchte ich meine Bankverbindung erfassen können.
Auslöserin	Spender, NPO
Auslösende Nachricht	Bankverbindung erfassen ()
Reaktionen	Bekanntgabe des Erfassungsstatus
Kurzfassung des Ablaufs	Ein Spender oder eine Organisation erfasst mit Bankverbindung erfassen() seine Bankverbindung. Es wird mitgeteilt ob die Bankverbindung erfolgreich erfasst wurden konnte.

Tabelle 12 UC-004: Bankverbindung erfassen

3.3.4 Administrator bezogene Anwendungsfälle

3.3.4.1 UC-012: Benutzer verwalten

Motivation	Als Administrator möchte ich Benutzer erstellen, anzeigen, bearbeiten und löschen können.
Auslöserin	Administrator
Auslösende Nachricht	Benutzer verwalten()
Reaktionen	Anzeige einer Benutzerverwaltung welche die CRUD Operationen unterstützt.
Kurzfassung des Ablaufs	Ein Benutzer kann mit Benutzer verwalten() in eine Benutzerverwaltung einsteigen, welche es ihm ermöglicht die CRUD Operationen auf dem Benutzerdatenbestand auszuführen.

Tabelle 13 UC-012: Benutzer verwalten

3.3.4.2 UC-013: Spendenstatistik anzeigen

Motivation	Als Administrator möchte ich diverse Spendenstatistiken einsehen können.
Auslöserin	Administrator
Auslösende Nachricht	Spendenstatistik anzeigen(Abfragekriterien)
Reaktionen	Anzeige von Spendenstatistiken.
Kurzfassung des Ablaufs	Ein Administrator kann mit Spendenstatistik anzeigen() Spendenstatistiken gemäss den Abfragekriterien generieren und anzeigen lassen.

Tabelle 14 UC-013: Spendenstatistik anzeigen

3.3.4.3 UC-014: Mitgliedschaft bestätigen

Motivation	Als Administrator möchte ich Mitgliedschafts Anfragen von gemeinnützigen Organisationen annehmen oder ablehnen können.
Auslöserin	Administrator
Auslösende Nachricht	Mitgliedschaft bestätigen(ja/nein)
Reaktionen	Mitglied wird benachrichtigt ob Mitgliedschaft angenommen oder abgelehnt worden ist.
Kurzfassung des Ablaufs	Ein Administrator kann mit Mitgliegschaft bestätigen() eine Mitgliedschaft annehmen oder ablehnen. Die NPO wird daraufhin über den Mitgliedschaftsstatus informiert.

Tabelle 15 UC-014: Mitgliedschaft bestätigen

3.4 Nicht funktionale Anforderungen

ID	MuSCoW	Status	Kategorie	Umgesetzt	Titel	Beschreibung
NF-1	S	erfasst	Security	outscoped	Zahlungsabwicklung	Die Zahlungsabwicklung soll durch einen externen Paymentprovider erfolgen.
NF-2	Mu	erfasst	Security	ja	Passwort Persistenz	Passwörter werden nur als Hash mit einem Salt gespeichert.
NF-3	Co	erfasst	Security	outscoped	Schutz vor unerlaubtem Zugang	Der Zugriff auf die Benutzerkonten soll vor einem Brute-Force Angriff durch eine maximale Anzahl Login Versuche (5 pro Minute) und ein Timeout (2 Minuten) geschützt sein.
NF-4	S	erfasst	Security	ja	Passwortrichtline	Passworter sollen aus mindestens 8 Zeichen bestehen, davon: - mindestens 1 Sonderzeichen - mindestens je 1 Klein- und Grossbuchstabe - mindestens 1 Zahl
NF-5	Mu	erfasst	Security		Übertragunsprotokoll	Kommunikation über das Internet soll verschlüsselt über SSL / TLS statt finden. Als Protokoll soll HTTPS verwendet werden.
NF-6	S	erfasst	Security	ja	Inaktivitäts Timeout	Nach 10 Minuten ohne Aktivität wird die aktive Session für ungültig erklärt.
NF-7	Mu	erfasst	Usability	ja	Spende Prozess	Ein Spendenvorgang soll in einem Interaktionsschritt (1 Knopfdruck) mit dem System erfolgen, sofern alles korrekt konfiguriert wurde.
NF-8	S	erfasst	Usability	ja	User Interface	Das System soll sich dem Benutzer in einer einheitlichen Form präsentieren. Dabei soll Wert auf eine Positionierung von Bedienelementen der Wichtigkeit entsprechend gelegt werden.
NF-9	S	erfasst	Usability	outscoped	Signalisation von längeren Operationen	Bei Operationen welche durchschnittlich länger als 1s dauern soll dem Benutzer grafisch Signalisiert werden das eine Verarbeitung im Gange ist. Zum Beispiel durch einen Ladebalken.
NF-10	S	erfasst	Usability	ja	Erstinbetriebnahme	Die initiale Konfiguration des Systems soll in weniger als 10 Minuten erfolgen. Danach sollen bereits Spenden getätigt werden können.
NF-11	S	erfasst	Availability	ja	System Verfügbarkeit	Das System soll im Jahresschnitt zu 90% der Zeit verfügbar sein.
NF-12	S	erfasst	Performance	ja	Ausführungsgeschwindigkeit Spendenvorgang	Ein Spendenvorgang, ausgelöst über den IoT Button, soll in 90% aller Fälle in weniger als 12 Sekunden abgewickelt werden. (Von der Betätigung des Hardware Buttons bis die Transaktion erfolgreich verbucht worden ist.)
NF-13	S	erfasst	Performance	ja	Reaktionszeit	Die durchschnittliche Reaktionszeit für das laden der Website, navigieren und refreshen der Website soll maximal 200ms betragen.
NF-14	S	erfasst	Performance	ja	Verarbeitungszeit	Bei der Durchführung von Operationen mit Berechnungen und Datenverarbeitung über die Website, soll die durchschnittliche Reaktionszeit maximal 1s betragen.
NF-15	S	erfasst	Performance	ja	Query- und Reportingzeit	Bei der Durchführung von komplexeren Queries über die Website soll die durchschnittliche Reaktionszeit maximal 10s betragen. Wenn Reports generiert werden müssen soll die durchschnittliche Reaktionszeit maximal 30s betragen.
NF-16	S	erfasst	Capacity	ja	Speicher	Das System soll initial mindestens Speicherplatz für folgende Entitäten bereitstellen: - 50 Benutzerprofile (bei der durchschnittlich ermittelten Grössen) - 1000 Transaktionen Da sich diese Zahlen in Zukunft stark vergrössern werden, muss der Speicherplatz skalierbar sein.
NF-17	S	erfasst	Capacity	ja	Durchsatz	Das System soll zu Beginn im Durchschnitt mindestens 10 Spendenvorgänge pro Minute verarbeiten können. Da der Durchsatz in Zukunft stark vergrössert werden muss, muss der Durchsatz skalierbar sein.
NF-18	Mu	erfasst	Supportability	Ja	Austauschbarkeit einzelner Komponenten	Die Systemarchitektur soll die Austauschbarkeit der folgenden Module ermöglichen: Benutzeroberfläche, Payment Provider
NF-19	Mu	erfasst	Testability	Ja	Testbares Design	Die Software soll so aufgebaut werden, dass die einzelnen Module isoliert getestet werden können.
NF-20	Mu	erfasst	Reusability	Ja	Wiederverwendung einzelner Komponenten	Die Software soll so aufgebaut werden, dass das Backend loose vom Frontend gekoppelt ist, damit zukünftig auch ein Frontend in einer anderen Technologie verwendet werden kann.

Abbildung 3 Übersicht der nicht funktionalen Anforderungen

3.5 Schnittstellen innerhalb des Systems

Die Kommunikation zwischen Frontend, Backend und IoT-Button soll über eine Technologie unabhängige Schnittstelle stattfinden. Das Frontend sollte austauschbar sein.

4 Architektur

Die Hauptziele der gewählten Architektur sind, eine gute Erweiterbarkeit, Flexibilität und Testbarkeit. Um dies zu erreichen, wurde die Webapplikation konsequenterweise in ein Frontend und ein Backend unterteilt (siehe 4.3 und 4.4), sodass prinzipiell mehrere Frontend Clients auf das bestehende Backend aufgesetzt werden können. Es wurde stark auf die Programmierparadigmen Dependency Inversion und Dependency Injection gesetzt.

Das Frontend wurde in C# programmiert und als ASP.NET Core 2.0 Webapplikation mit MVC Pattern realisiert. Das Backend wurde ebenfalls in C# programmiert und als ein ASP.NET Core 2.0 basierendes REST API umgesetzt. Die Struktur im Backend stützt sich auf den Clean Architecture Ansatz und setzt vermehrt auf asynchrone Prozesse. Sowohl Frontend als auch Backend werden auf einem Applikationsserver in der Microsoft Azure Cloud gehostet. Als Basis für den IoT-Button (siehe 4.5) wurde ein ESP-12F Mikrokontroller der ESP8266 Familie mit Arduino Bootloader verwendet. Die Software wurde in C++ programmiert.

4.1 Systemübersicht

In Abbildung 4 wird eine Übersicht über die Deployment Umgebung gegeben. Sie besteht aus zwei Nodes, der Microsoft Azure Cloud und dem ESP8266 Microkontroller. In der Azure Cloud werden das Frontend/ Webapplikation sowie das Backend/ REST API in einem gemeinsamen App Service gehostet. Das Backend verwendet für die Datenpersistierung eine SQL-Datenbank, welche als eigener Server aufgesetzt ist. Sowohl Frontend als auch Backend benutzen für die Archivierung der Logfiles ein Speicherkonto/ Blob-Storage, welches ebenfalls in der Azure Cloud gehostet wird. Die IoT-Button Software wird auf dem ESP8266 betrieben und benutzt das Backend via REST Schnittstelle.

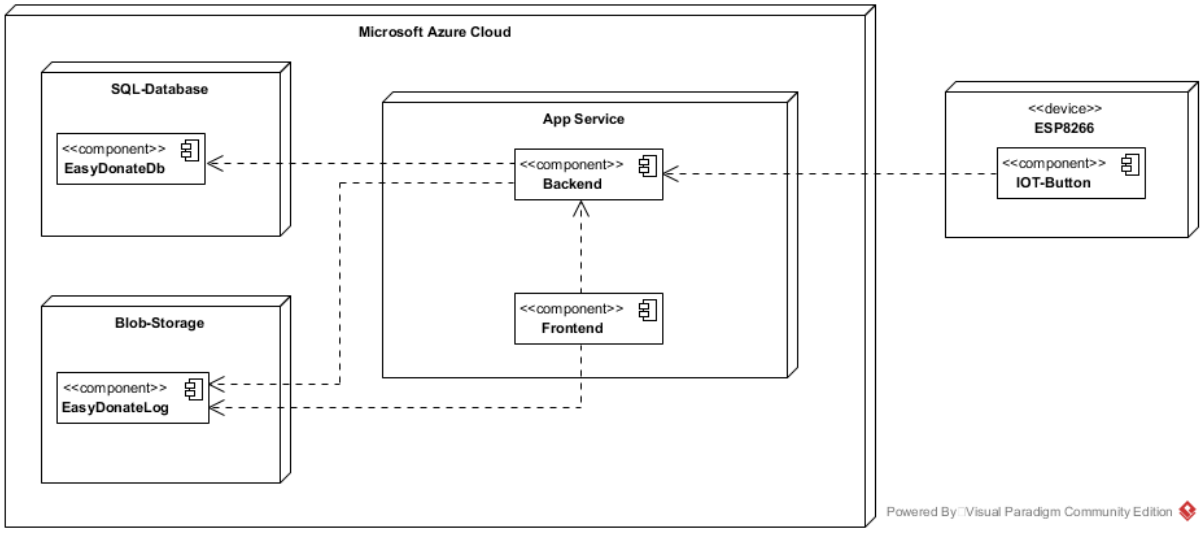

Abbildung 4 Deployment Diagramm

4.2 Domänenmodell

Das nachfolgende Modell bildet die Entitäten ab, welche dem Domänenmodell zugrunde liegen. Diese werden benötigt, um die fachliche Logik abzubilden.

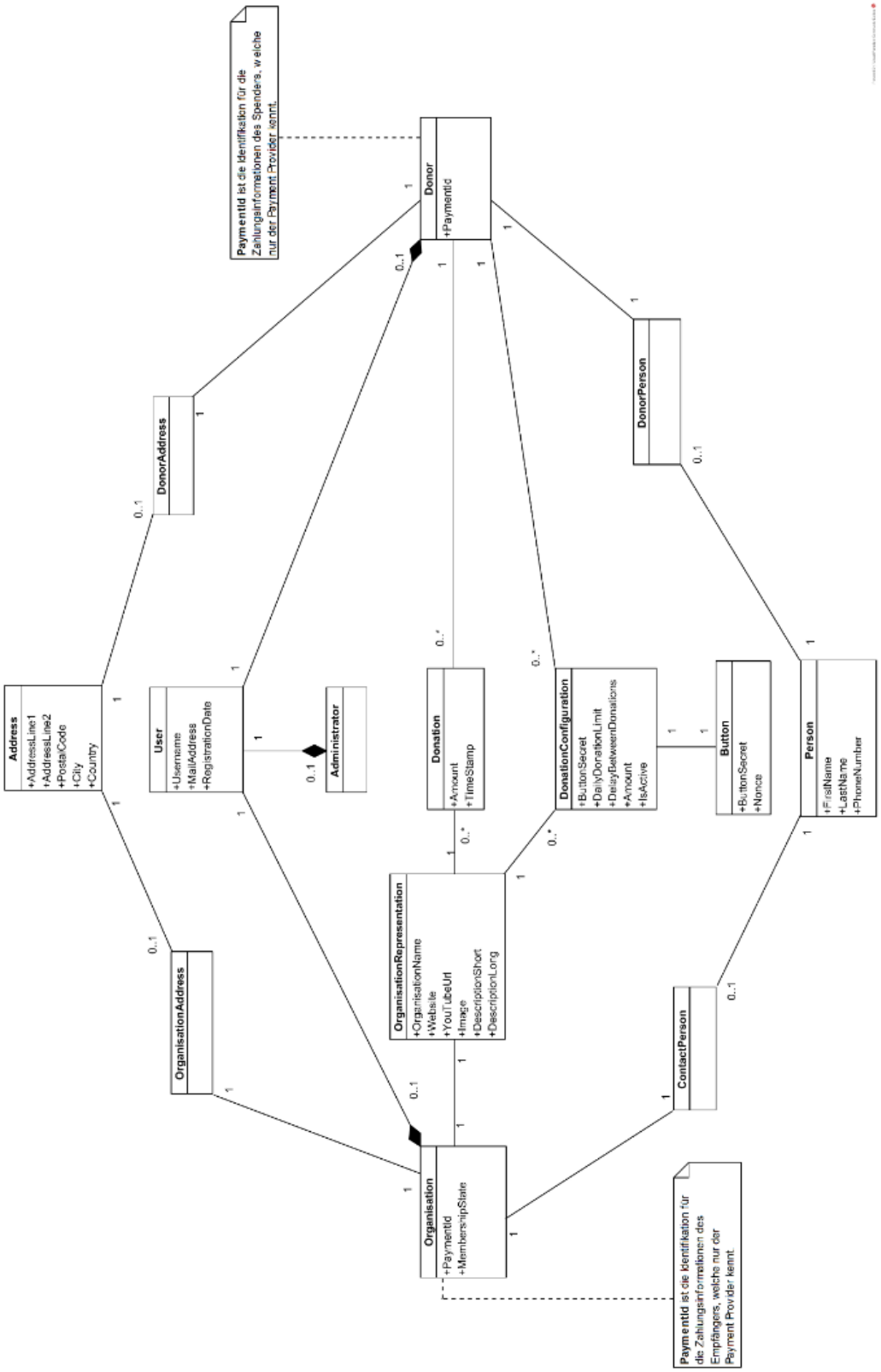

Abbildung 5 Domänenmodell

4.2.1 User

Repräsentiert einen Benutzer der Plattform

4.2.2 Organisation

Repräsentiert einen Benutzer der Plattform in der Rolle einer Non-Profit-Organisation (NPO).

4.2.3 Donor

Repräsentiert einen Benutzer der Plattform in der Rolle eines Spenders.

4.2.4 Administrator

Repräsentiert einen Benutzer der Plattform in der Rolle eines Administrators.

4.2.5 Donation

Repräsentiert eine Spende.

4.2.6 DonationConfiguration

Repräsentiert eine Spenden Konfiguration.

4.2.7 Button

Repräsentiert eine Konfiguration eines IoT-Buttons.

4.2.8 OrganisationRepresentation

Repräsentiert das öffentlich einsehbare Profil einer Organisation.

4.2.9 Address

Repräsentiert eine Adresse.

4.2.10 DonorAddress

Repräsentiert die Beziehung zwischen einem Spender und einer Adresse.

4.2.11OrganisationAddress

Repräsentiert die Beziehung zwischen einer Organisation und einer Adresse.

4.2.12Person

Repräsentiert eine natürliche Person.

4.2.13DonorPerson

Repräsentiert die Beziehung zwischen einem Spender und einer Person.

4.2.14ContactPerson

Repräsentiert die Beziehung zwischen einer Organisation und einer Person. Die Person dient in dieser Beziehung als Kontaktperson der Organisation.

4.3 Frontend

4.3.1 MVC

Das Frontend ist als ASP.NET Core 2.0 (1) MVC Webapplikation umgesetzt. Der Aufbau folgt dem Model-View-Controller (MVC) Pattern. Durch die Aufteilung der Applikation in die drei Gruppen Model, View und Controller wird eine klare Aufteilung der Zuständigkeit erreicht, wie es nach dem Single Responsibility Principle (SRP) gefordert ist. Die spezifische Aufgabenverteilung wird in den nachfolgenden Kapiteln diskutiert.

4.3.1.1 Model

Das Modell beinhaltet im MVC die Daten für die Präsentation benötigt werden. In der verwendeten Architektur werden die Models (siehe Abbildung 6) nochmals in ViewModels und ApiModels unterteilt. Die ViewModels sind hierbei die Träger der Information für die View. Über die ApiModels findet die Kommunikation mit dem Backend statt.

4.3.1.2 View

Die Views übernehmen die Präsentation der in den ViewModels beinhalteten Daten im User Interface. Die Views selbst sind grösstenteils frei von Applikationslogik und stellen lediglich die Struktur für die Darstellung bereit.

4.3.1.3 Controller

Die Controller werden zur Orchestration der Komponenten im MVC verwendet. Sie reagieren auf Events, welche über das User Interface ausgelöst werden, koordinieren die Verarbeitung der Modelle im Hintergrund und entscheiden welche View als Antwort gerendert wird.

4.3.2 Clients

Entgegen dem einfachsten Aufbau der Microsoft MVC Webapplikation kennt das Frontend in der gewählten Architektur keine eigene Datenpersistierung. Für das Speichern und die Bereitstellung der Daten ist das Backend verantwortlich. Um die Kommunikation zwischen Frontend und Backend zu vereinfachen, wurden Clients eingeführt (siehe Kapitel 4.6.2). Die Clients übernehmen den Aufbau der Verbindung, das Transformieren der Messages und das Absetzen der Requests. Die Clients stellen ein sauberes Interface für die Verwendung in den Controllern zur Verfügung.

4.3.3 Design

Für die Gestaltung der HTML Seiten wird auf Bootstrap V4.1.1 gesetzt. Der Einsatz von Bootstrap vereinfacht die einheitliche Gestaltung, öffnet die Tür zum Einsatz von vielen vorgefertigten Komponenten und ermöglicht den Zugang zu einfachen Mitteln für ein Responsive Design.

4.3.4 Paketdiagramm

Abbildung 6 zeigt eine Übersicht über die Paketstruktur im Frontend. Das Fronend besteht aus lediglich einem Projekt mit verschiedenen Modulen.

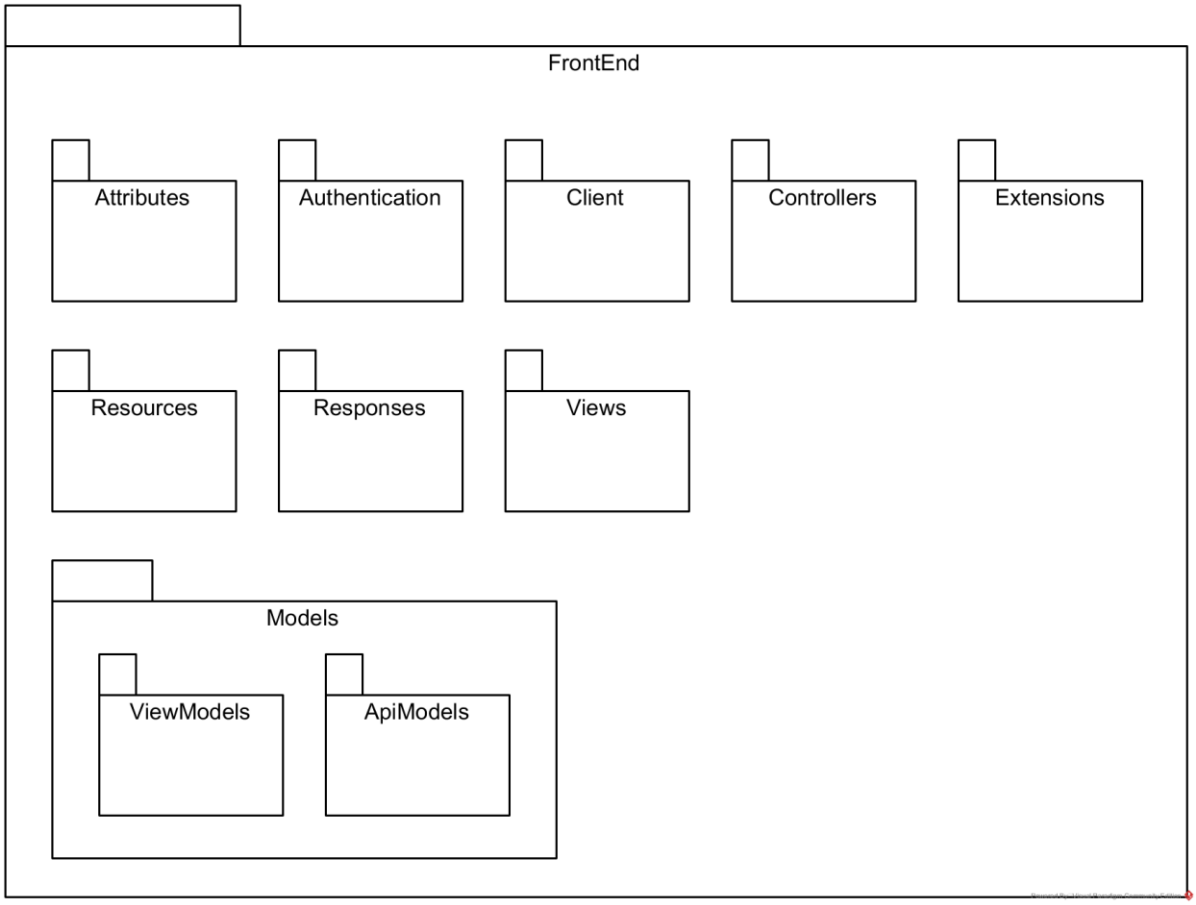

Abbildung 6 Frontend Architektur als Paketdiagramm

4.4 Backend

4.4.1 Clean Architecture

Um den Anforderungen an das Backend gerecht zu werden, wurde diese als Clean Architecture (2) umgesetzt. Ziel des Clean Architecture Ansatzes ist es, einen zentralen Core zu definieren, welcher die Business Logik beinhaltet und von welchem alle umliegenden Komponenten, wie zum Beispiel die Infrastructure und das User Interface oder WebApi abhängig sind. Die Umkehrung der Abhängigkeit wird durch eine Dependency Inversion erzielt. Dazu werden Abstraktionen und Interfaces im Kern definiert und dann von konkreten Typen, beispielsweis in der Infrastructure, implementiert. Die Business Logik selbst stellt ihre Funktionalität in Form von Domain Services zur Verfügung. Das beschriebene Design lässt sich am besten als «Zwiebelschalen»-Modell wie in Abbildung 7 darstellen.

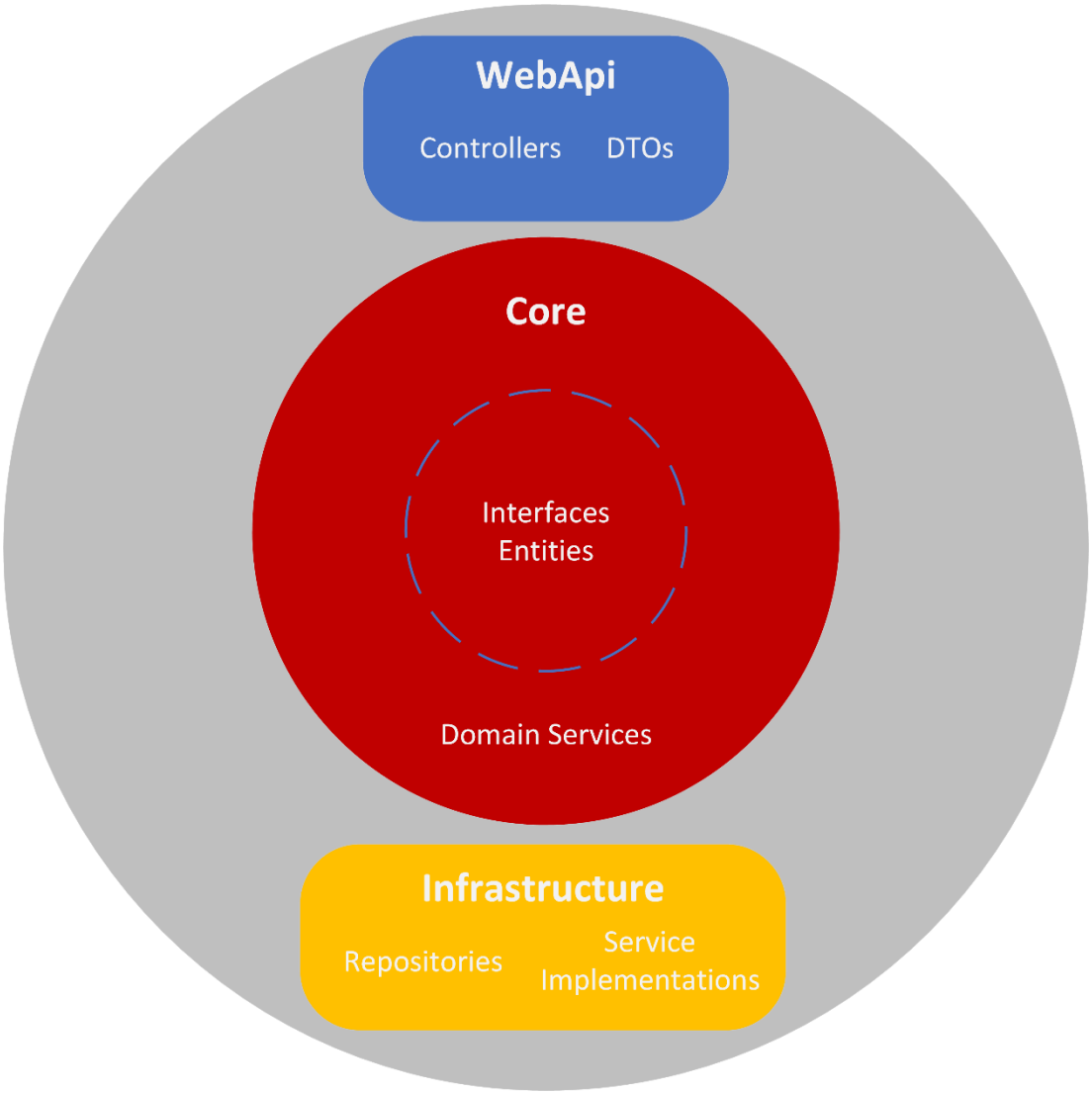

Abbildung 7 Backend Architektur im Clean Architecture Ansatz als Onion View

Wenn man diese Architektur dennoch als Layer darstellt, dann ergibt sich eine Sicht wie in Abbildung 8. Dabei stellen die durchgezogenen Pfeile Compile Time- und die gestrichelten Pfeile Run Time Abhängigkeiten dar.

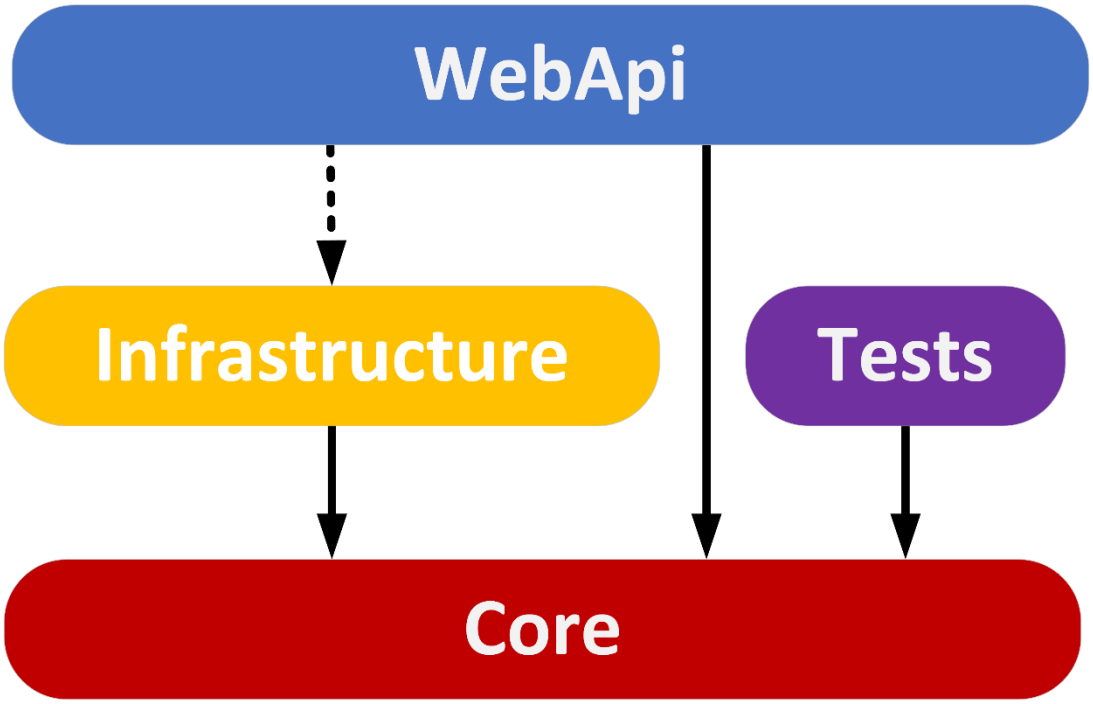

Abbildung 8 Clean Architecture als Layer Architektur dargestellt

Das WebApi weiss zur Compile Time nichts von der konkreten Implementierung der Infrastruktur, sondern kennt nur das im Kern definierte Interface. Um diesen Aufbau lauffähig zu machen wird ein Dependency Injection Framework benötigt. In den .Net Core Webapplikation kann hierfür auf das integrierte Dependency Injection Framework zurückgegriffen werden.

Die Vorteile dieser Architektur im Vergleich zu einer konventionellen Drei-Schichten-Architektur sind insbesondere:

- Reduzierte Koppelung zwischen Klassen
- Verbesserte Wiederverwendbarkeit
- Verbesserte Wartbarkeit
- Verbesserte Testbarkeit

4.4.2 REST Schnittstelle

Nachfolgend sind die im Backend zur Verfügung stehenden Ressourcen der REST Schnittstelle mit den zugehörigen HTTP Verben aufgeführt.

Eine detaillierte Übersicht über die REST Schnittstelle mit Beispielen und allen Datenmodellen kann über das Verzeichnis «/api/Swagger» der Webapplikation aufgerufen werden. In der aktuellen Form ist die Applikation auf Azure gehostet und auf die REST Definition kann wie folgt zugegriffen werden:

http://easydonate.azurewebsites.net/api/Swagger/

Die REST Schnittstelle bietet aktuell nicht für alle Resources die vollständigen CRUD Operationen an. Es wurden lediglich die Operationen umgesetzt, welche für die Funktionalität der geplanten Use-Cases erforderlich waren.

4.4.2.1 Accounts

Der Account ist keine Ressource im eigentlichen Sinn der Rest Definition, sondern eher ein Controller zum Verwalten von Benutzerkonten.

HTTP Verb	Endpoint
POST	/Accounts/Login
POST	/Accounts/RegisterDonor
POST	/Accounts/RegisterOrganisation
POST	/Accounts/RegisterAdministrator
GET	/Accounts/ConfirmEmail
POST	/Accounts/ForgotPassword
POST	/Accounts/ResetPassword
POST	/Accounts/ChangePassword

4.4.2.2 Administrators

Die Ressource Administrator repräsentiert einen Benutzer auf der Plattform in der Rolle eines Administrators.

HTTP Verb	Endpoint
GET	/Administrators
GET	/Administrators/{id}
PUT	/Administrators/{id}
DELETE	/Administrators/{id}

4.4.2.3 Buttons

Die Ressource Button repräsentiert die Konfiguration eines IoT-Buttons.

HTTP Verb	Endpoint
GET	/Buttons/{id}
GET	/Buttons/{id}/nonces

4.4.2.4 DonationConfigurations

Die Ressource DonationConfiguration repräsentiert eine Spendenkonfiguration.

HTTP Verb	Endpoint
PUT	/DonationConfigurations/{id}

4.4.2.5 Donations

Die Ressource Donation repräsentiert eine Spende.

HTTP Verb	Endpoint
GET	/Donations
POST	/Donations
GET	/Donations/{id}
DELETE	/Donations/{id}

4.4.2.6 Donors

Die Ressource Donor repräsentiert einen Benutzer auf der Plattform in der Rolle eines Spenders. Auf die abhängigen Entitäten des aktuell eingeloggten Spenders kann jeweils über den Teil Pfad «/Donors/current/..» zugegriffen werden.

HTTP Verb	Endpoint
GET	/Donors
GET	/Donors/{id}
PUT	/Donors/{id}
DELETE	/Donors/{id}
GET	/Donors/current/donationconfigurations
GET	/Donors/current/persons
POST	/Donors/current/persons
PUT	/Donors/current/persons/{id}
GET	/Donors/current/addresses
POST	/Donors/current/addresses
PUT	/Donors/current/addresses/{id}
PUT	/Donors/current/donationconfigurations/active/organisationRepresentation/{id}
GET	/Donors/current/buttons
GET	/Donors/current/donations

4.4.2.7 Organisations

Die Ressource Organisation repräsentiert einen Benutzer auf der Plattform in der Rolle einer Non-Profit-Organisation. Auf die abhängigen Entitäten der aktuell eingeloggten Organisation kann jeweils über den Teil Pfad «/Organisations/current/..» zugegriffen werden.

HTTP Verb	Endpoint
GET	/Organisations
GET	/Organisations/representations
GET	/Organisations/representations/{id}
PUT	/Organisations/representations/{id}
GET	/Organisations/current/representations
PUT	/Organisations/{id}/membershipState
DELETE	/Organisations/{id}
GET	/Organisations/current/donations
GET	/Organisations/current/addresses
POST	/Organisations/current/addresses
PUT	/Organisations/current/addresses/{id}
GET	/Organisations/current/persons
POST	/Organisations/current/persons
PUT	/Organisations/current/persons/{id}

4.4.2.8 Statistics

Die Statistic ist keine Ressource im eigentlichen Sinn der Rest Definition, sondern eher ein Controller zum Abfragen von diversen Statistiken, welche für das Reporting auf der Spendenplattform verwendet werden.

HTTP Verb	Endpoint
GET	/Statistics/RecentDonations
GET	/Statistics/DonationsWithinTimeSpan
GET	/Statistics/UsersWithinTimeSpan
GET	/Statistics/DonorsByDonationAmount
GET	/Statistics/OrganisationsByDonationAmount
GET	/Statistics/Overview

4.4.3 Paketdiagramm

Abbildung 9 zeigt eine Übersicht über die Paketstruktur im Backend. Das Backend besteht aus den drei Projekten WebApi (REST API), Infrastructure und Core, welche wiederum diverse Module beinhalten. Die Abhängigkeiten fliessen alle in Richtung Core.

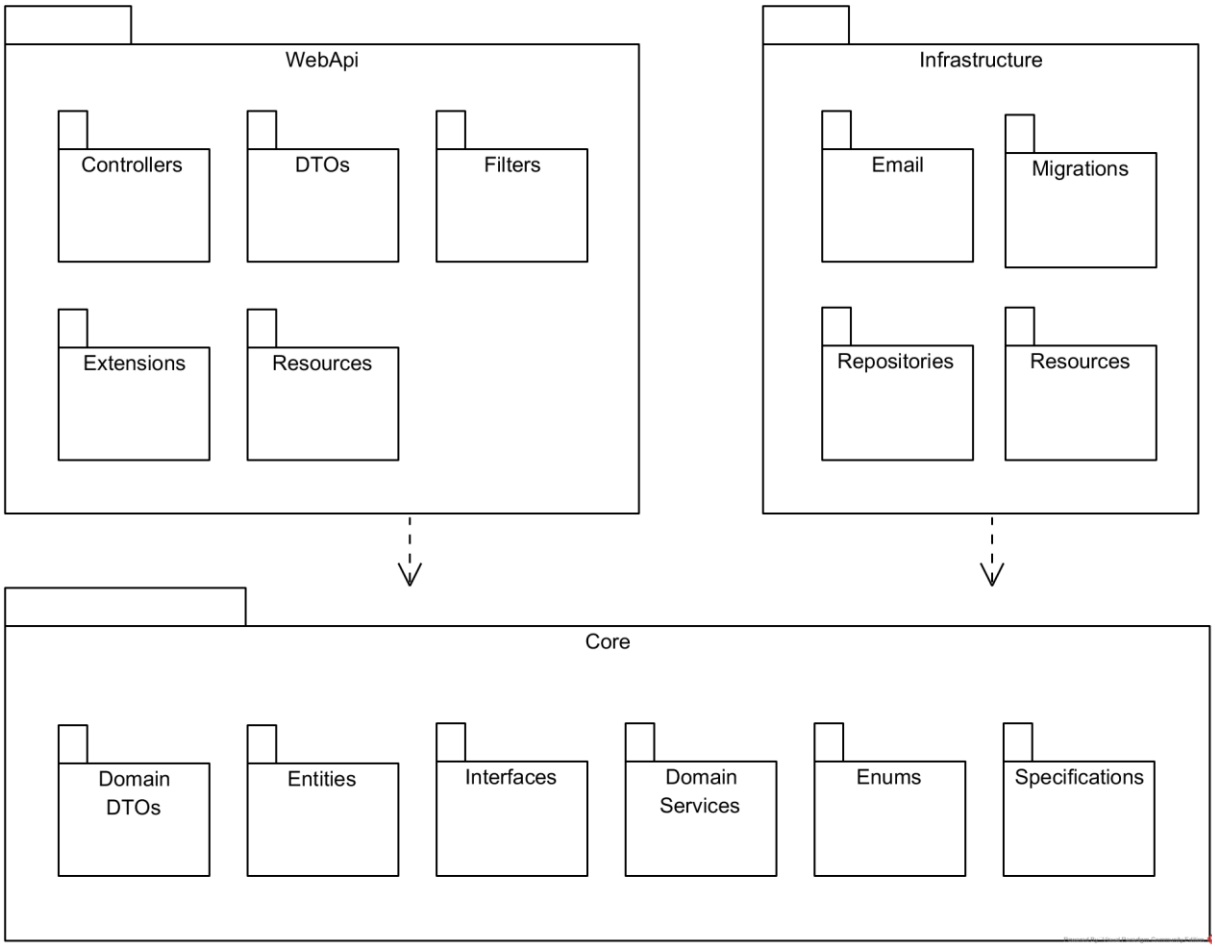

Abbildung 9 Backend Architektur als Paketdiagramm mit den Abhängigkeiten

Das WebApi basiert auf einer Microsoft ASP.NET Core 2.0 Applikationen, welches bereits ein Dependency Injection Framework (DIF) integriert. Um die konkreten Implementierungen der Interfaces zu konfigurieren, müssen diese dem DIF bekannt sein. Deshalb gibt es in Realität auch eine Abhängigkeit vom WebApi zur Infrastructure. Die Konfiguration des DIF ist aber die einzige Ausnahme und der einzige Verwendungszweck der Referenz zwischen WebApi und Infrastructure.

4.5 IoT-Button

4.5.1 Hardware

Die IoT-Button Software wird auf einem Mikrocontroller vom Typ ESP-12F betrieben. Dieser Mikrokontroller, welcher zur ESP8266 Familie gehört, zeichnet sich durch einen günstigen Preis sowie eine geringe Leistungsaufnahme aus. Der Prozessorkern basiert auf einer 32-Bit Architektur und arbeitet mit einem Systemtakt von 80 MHz – 160 MHz. Das Modul beinhaltet ein TCP/IP Protokoll Stack und verfügt über ein Wifi-Modul.

4.5.1.1 Schaltplan

Der Schaltkreis für den IoT-Button wurde so einfach wie möglich gehalten. Für einen optimalen Betrieb wäre ein Kondensator zwischen VCC und GND sinnvoll. Dadurch könnten Lastspitzen ausgeglichen werden. Die Eingänge werden idealerweise über einen Pullup – Pulldown Widerstand betrieben, damit diese einen definierten Pegel besitzen. Während der Entwicklungsphase war der Betrieb des IoT-Buttons dadurch nicht eingeschränkt.

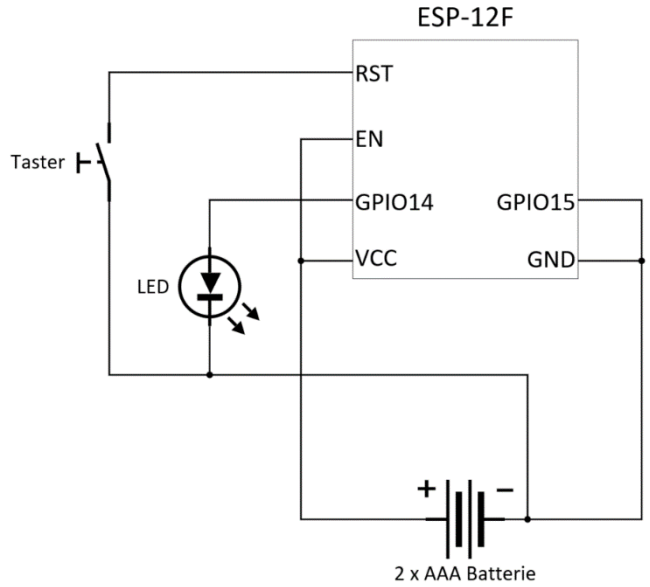

Abbildung 10 Schaltplan IoT-Button

4.5.2 Software

Da der Arduino Bootloader auf den Mikrocontroller geladen wird, besteht das Grundgerüst der Software aus den Methoden «setup()» und «loop()». Die «setup()» Methode wird einmalig während des Startup ausgeführt. Danach wird die «loop()» Methode in einer Endlosschlaufe durchlaufen. Die aktuelle Implementierung verwendet lediglich die «setup()» Methode. Bei Verwendung des IoT-Buttons erwacht dieser aus dem Deep Sleep, führt seine Arbeit durch und retourniert danach wieder in den Deep Sleep bis zur nächsten Aktivierung.

4.5.2.1 Eingesetzte Libraries

Für die Konfiguration des IoT-Button wird die Library WiFiManager (3) eingesetzt. Diese stellt ein Konfigurationsportal zur Verfügung über welches der IoT-Button mit einem WLAN-Netzwerk verbunden und die Konfiguration des IoT-Buttons eingegeben werden kann.

4.5.3 Datenpersistierung

Der Microcontroller verfügt über einen 4 MB SPI Flash Speicher. Davon können bis zu 3MB als SPI Flash File System (SPIFFS) genutzt werden. Die Konfiguration des IoT-Buttons wird im JSON Format auf diesem File System persistiert.

4.6 Software Komponenten

4.6.1 Repository und Specification Pattern

Das Repository-Pattern abstrahiert die Daten aus der Datenbank damit die Anwendung über ein einfaches Interface auf diese Sammlung von Daten zugreifen kann. Das Hinzufügen, Lesen, Aktualisieren und Entfernen von Elementen aus dieser Sammlung erfolgt über eine Reihe von Methoden. Durch den Einsatz dieses Musters wird eine lose Kopplung erreicht und das Unit Testing erleichtert.

Um Code Duplikation zu vermeiden wurde ein generisches Repository implementiert. Alle Entitäten sind abgeleitet von BaseEntity. Dies stellt sicher, dass jede Entität eine Guid als Primary Key besitzt.

```
public interface IAsyncRepository<T> where T : BaseEntity
{
    Task<T> GetByIdAsync(Guid id);
    Task<T> GetSingleBySpecAsync(ISpecification<T> spec);
    Task<List<T>> ListAllAsync();
    Task<List<T>> ListAsync(ISpecification<T> spec);
    Task<T> AddAsync(T entity);
    Task UpdateAsync(T entity);
    Task DeleteAsync(T entity);
}

public class BaseEntity
{
    public Guid Id { get; set; }
}
```

Abbildung 11 Schnittstelle des Repository Pattern und Basis Klasse aller Entitäten

Da nahezu alle Services im Backend asynchron ablaufen, wurde das Repository ebenfalls asynchron implementiert.

Die Repository Methoden geben ausschliesslich Listen von Entitäten zurück. Es wurde absichtlich darauf verzichtet Abfragen zurückzugeben in Form von «IQueryables». Dies wäre zwar sehr bequem, da die Abfrage vor der Ausführung weiter verfeinert werden könnte. Ein Problem bei diesem Ansatz ist allerdings, dass die Geschäftslogik so in höhere Anwendungsschichten übergeht und dort dupliziert wird.

4.6.1.1 Specification Pattern

Da die Repositories keine Abfragen zurückgeben, besteht die Gefahr, dass die Repositories mit unzähligen Abfragemethoden aufgebläht werden. Die Lösung dafür ist die Trennung von Abfragen in eigene Typen mit Hilfe des «Specification Pattern». Die «Specification» kann ein Ausdruck enthalten, der zum Filtern der Abfrage verwendet wird. Ebenfalls können über «Includes» in Beziehung stehende Daten mitgeladen werden.

4.6.2 Client

Das Konzept rund um die Clients im Frontend vereinfacht den Zugriff auf das REST API des Backends. Es übernimmt den Aufbau der Verbindung, die Transformation (Serialisierung/ Deserialisierung) der Messages und das Absetzen der Requests. Eine Übersicht zu den wichtigsten Klassen findet man in Abbildung 12.

Für jede Ressource im Backend wurde ein spezifischer Client angelegt, welcher vom ClientBase ableitet. In Abbildung 12 wäre dies der DonorClient. Die Ressourcen spezifischen Clients definieren den Pfad zum passenden Endpoint und stellen sinnvolle Methoden zum Abruf und zum Update der relevanten Entitäten zur Verfügung. Der ApiClient wird in der Client Implementierung für die Kommunikation mit der REST Schnittstelle verwendet. Er baut also den Request inkl. Header und Content auf, sorgt dafür, dass das JWT übergeben wird und serialisiert die Message zu einem JSON. Die Antworten werden dann in den ressourcenspezifischen Clients von einer HttpResponseMessage in eine ApiResonse umgewandelt. Die Transformation wird dabei von der ClientBase übernommen, welche auch als Basisklasse für die konkreten Clients dient. Die ApiResponse beinhaltet die Antwort in einer für die Verwendung in den Controller vorbereiteten Art und Weise. Sie gibt Auskunft über den Erfolg der Operation, beinhaltet Fehlerinformationen und falls vorhanden die angeforderte Ressource.

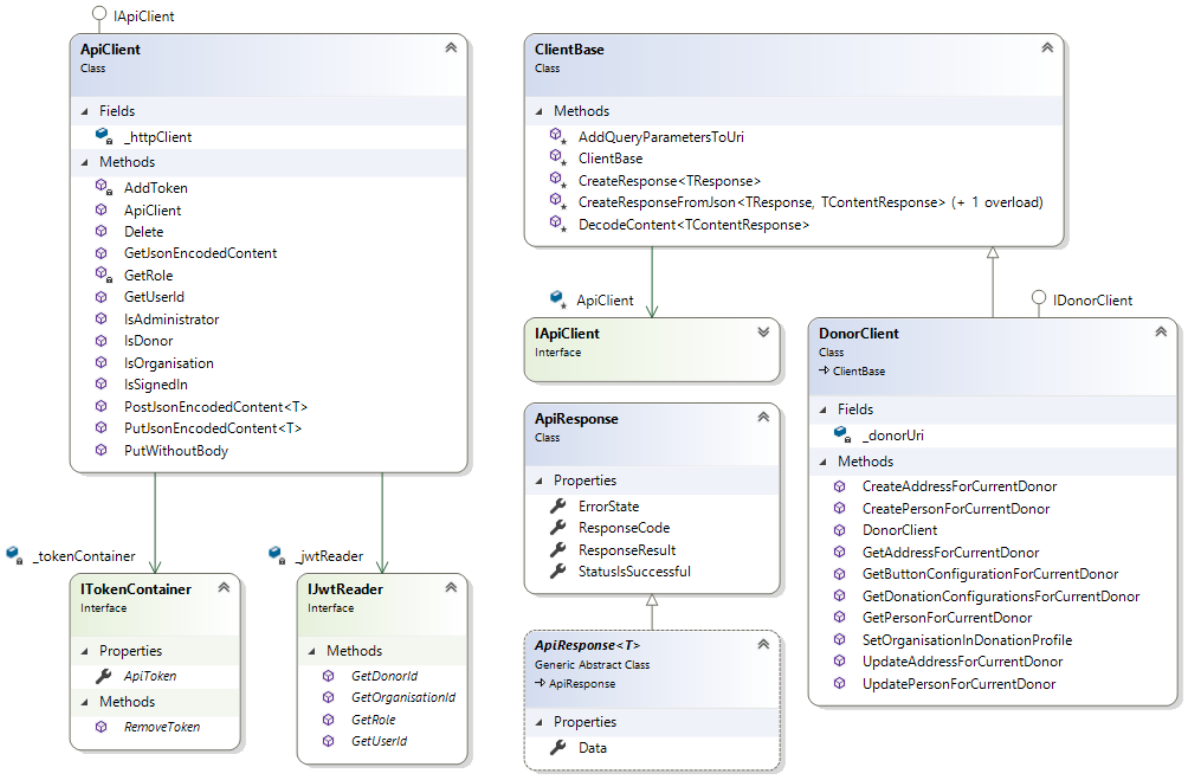

Abbildung 12 Klassendiagramm des Client Konzepts mit den abhängigen Klassen

4.6.3 Benutzerverwaltung

Für die Verwaltung der Benutzer mit Logins, Rollen, etc. wird das ASP.NET Identity Framework (4) von Microsoft verwendet. Damit das Benutzerverwaltungsframework jederzeit einfach ausgetauscht werden kann, wurde eine Fassade um das Identity Framework gebaut. Die Definition des Interfaces IUserManager welche die Schnittstelle der Fassade bestimmt, liegt im Package Core. Realisiert wird das Interface von der Fassadenklasse, welche sich im Package Infrastructure befindet. Dadurch benötigt das Package Core weder auf die Identity Library noch auf die konkrete Implementierung der Fassadenklasse eine Referenz. Es ist also eine maximale Entkopplung gewährleistet.

Die Umsetzung der Benutzerverwaltung erfordert unter anderem, dass es möglich ist, mehrere Entitäten ganz oder gar nicht zu manipulieren. Diese transaktionale Funktionalität stellt die Klasse TransaktionsManager zur Verfügung, welche als ein Wrapper um die EntitiyFramework Transaktions-Funktionalität fungiert.

Für die Verwaltung der Benutzer in den Rollen Spender, Organisation und Administrator existiert jeweils eine eigene Implementation des IUserManager Interface. Die konkreten Implementierungen der Verwaltung für die Spender und die Organisationen werden auf den folgenden Seiten beschrieben.

4.6.3.1 DonorManager

Der DonorManager stellt die Funktionalität zur Verfügung um Spender zu verwalten.

Bei der Erstellung eines Spenders wird folgendes sichergestellt:

- Entität Spender ist erstellt
- Benutzer mit Rolle Spender ist erstellt
- IoT-Buttonkonfiguration ist erstellt
- Spendenkonfiguration mit Default Werten ist erstellt

Beim Löschen eines Spenders wird sichergestellt, dass alle zu diesem Spender gehörenden Datensätze gelöscht werden.

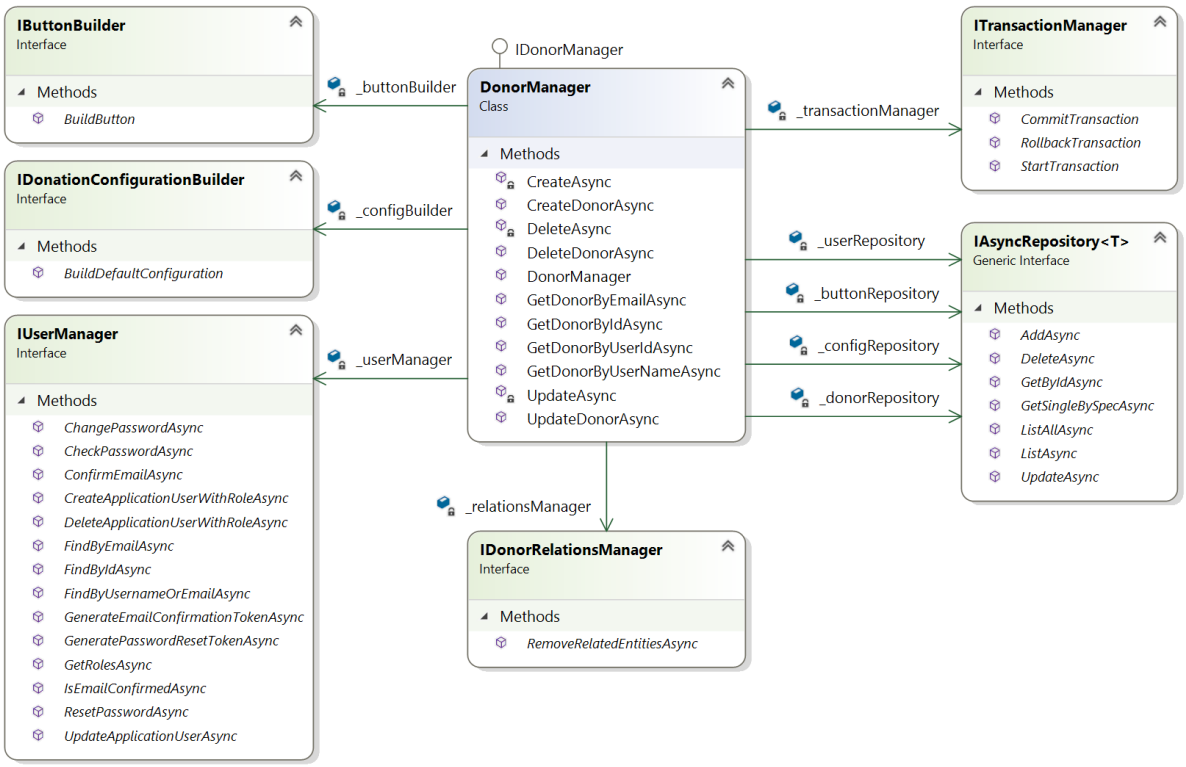

Abbildung 13 Klassendiagramm des DonorManager mit den abhängigen Klassen

4.6.3.2 OrganisationManager

Der OrganisationManager stellt die Funktionalität zur Verwaltung von Organisationen zur Verfügung.

Bei der Erstellung einer Organisation wird folgendes sichergestellt:

- Entität Organisation ist erstellt
- Benutzer mit Rolle Organisation ist erstellt

Beim Löschen einer Organisation wird sichergestellt, dass alle zu dieser Organisation gehörenden Datensätze gelöscht werden.

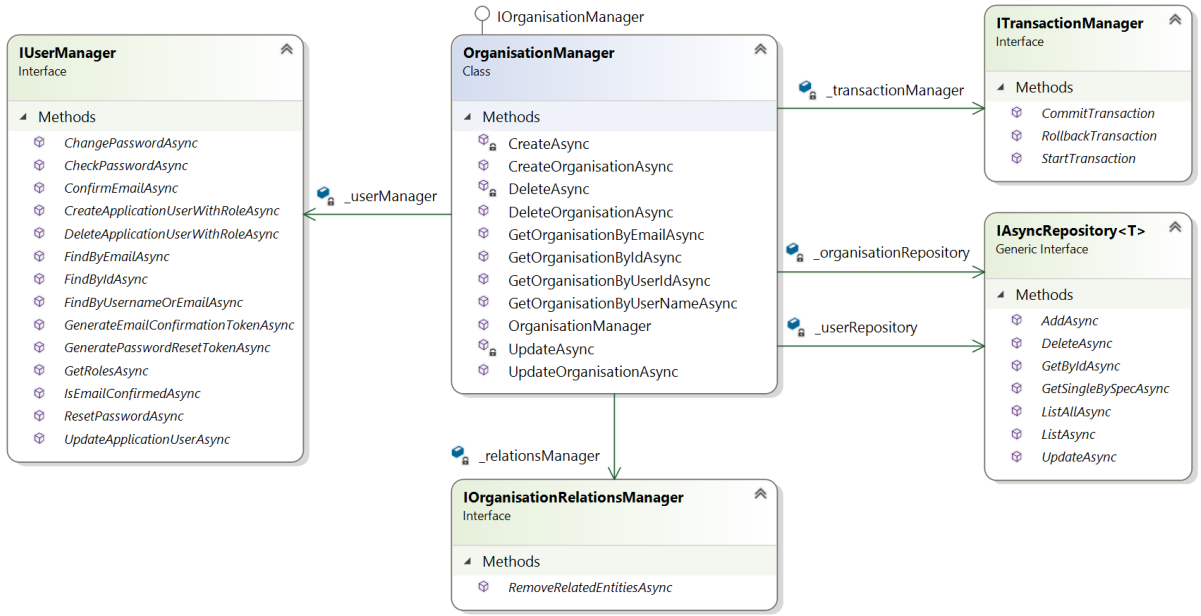

Abbildung 14 Klassendiagramm des OrganisationManager mit den abhängigen Klassen

4.6.4 Payment Provider

Im Rahmen der Masterarbeit wird kein kommerzieller Payment Provider implementiert. Es wurden aber architektonische Vorbereitungen getroffen, sodass dieser zu späterem Zeitpunkt relativ einfach integriert werden kann.

Die Zahlungsabwicklung über den Payment Provider wird über ein IPaymentProvider und ein IPaymentManager abstrahiert.

Jeder konkrete Payment Provider muss das Interface IPaymentProvider implementieren. Im Falle der Masterarbeit ist lediglich ein DummyPaymentProvider implementiert.

Dieses definiert folgende Methoden:

```
public interface IPaymentProvider
{
    string GetCustomerId(string paymentInformation);
    string GetAccountId(string accountInformation);
    Task ChargeCustomerAsync(ChargeDto charge);
    Task PayoutCustomerAsync(PayoutDto payout);
}
```

Abbildung 15 Payment Provider Interface

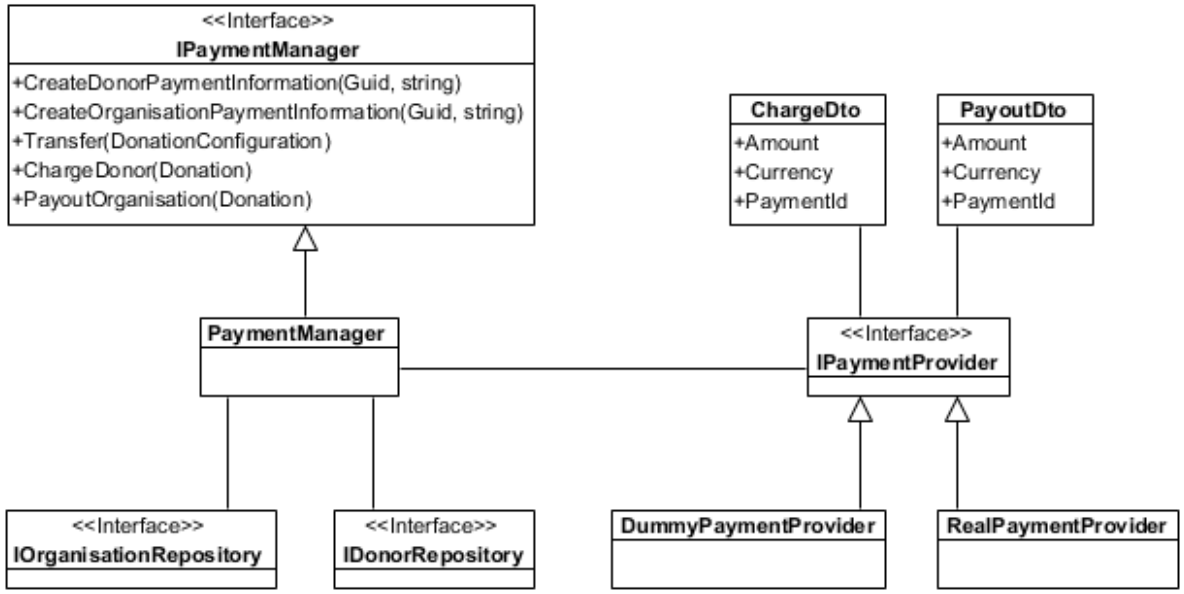

Abbildung 16 UML Diagramm IPaymentManager und IPaymentProvider

4.7 Wichtige Abläufe

In den nachfolgenden Kapiteln werden die wichtigsten Abläufe in Form von Sequenzdiagrammen erläutert.

4.7.1 IoT-Button

Abbildung 17 zeigt eine Übersicht über die Prozesse, welche bei Betätigung des IoT-Buttons ausgeführt werden können.

Falls keine Verbindung mit dem gespeicherten WiFi-Netzwerk aufgebaut werden kann oder noch keine Verbindung initialisiert wurde, wird das Konfigurationsportal gestartet. Der detaillierte Ablauf der Konfiguration ist in Abbildung 18 dargestellt.

Wenn der IoT-Button betätigt wird und eine erfolgreiche Verbindung zu einem WiFi Access Point aufgebaut werden kann, wird der Spendenprozess gestartet. Dieser Prozess ist im Detail in Abbildung 19 aufgeführt.

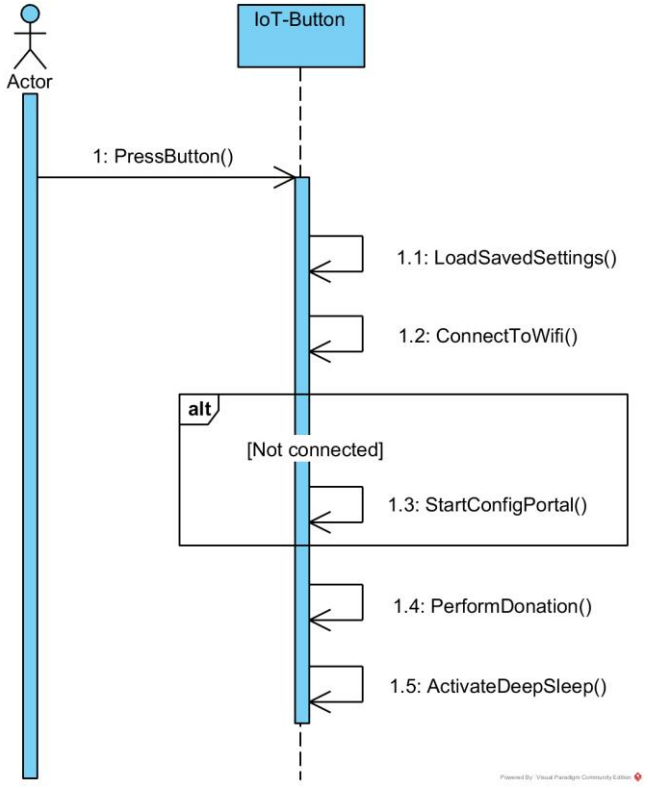

Abbildung 17 Übersicht über die ablaufenden Prozesse bei Betätigung des IoT-Buttons

4.7.1.1 Konfiguration

Wenn sich ein neuer Spender auf der Plattform registriert, sind seine ersten Schritte, seinen persönlichen IoT-Button mit dem Internet zu verbinden und sein Spenderprofil mit dem IoT-Button zu verknüpfen. Hierzu erhält der Spender auf der Plattform eine Button-ID, sowie ein Secret, welches durch einen kryptografischen Zufallsgenerator generiert wird.

Beides wird in der Datenbank der Plattform gespeichert. Die Button-ID und das Secret werden beim konfigurieren des IoT-Buttons von Hand über ein Internetfähiges Endgerät eingegeben. Der IoT-Button befindet sich dabei im Accesspoint Betrieb und bietet eine Weboberfläche für die Konfiguration an. Im gleichen Zug wird auch das Wi-Fi Netz konfiguriert, über welches sich der Button später mit dem Internet verbindet um einen Request für eine Spendentransaktion abzusetzen.

Dieser Konfigurationsprozess wird in Abbildung 18 gezeigt.

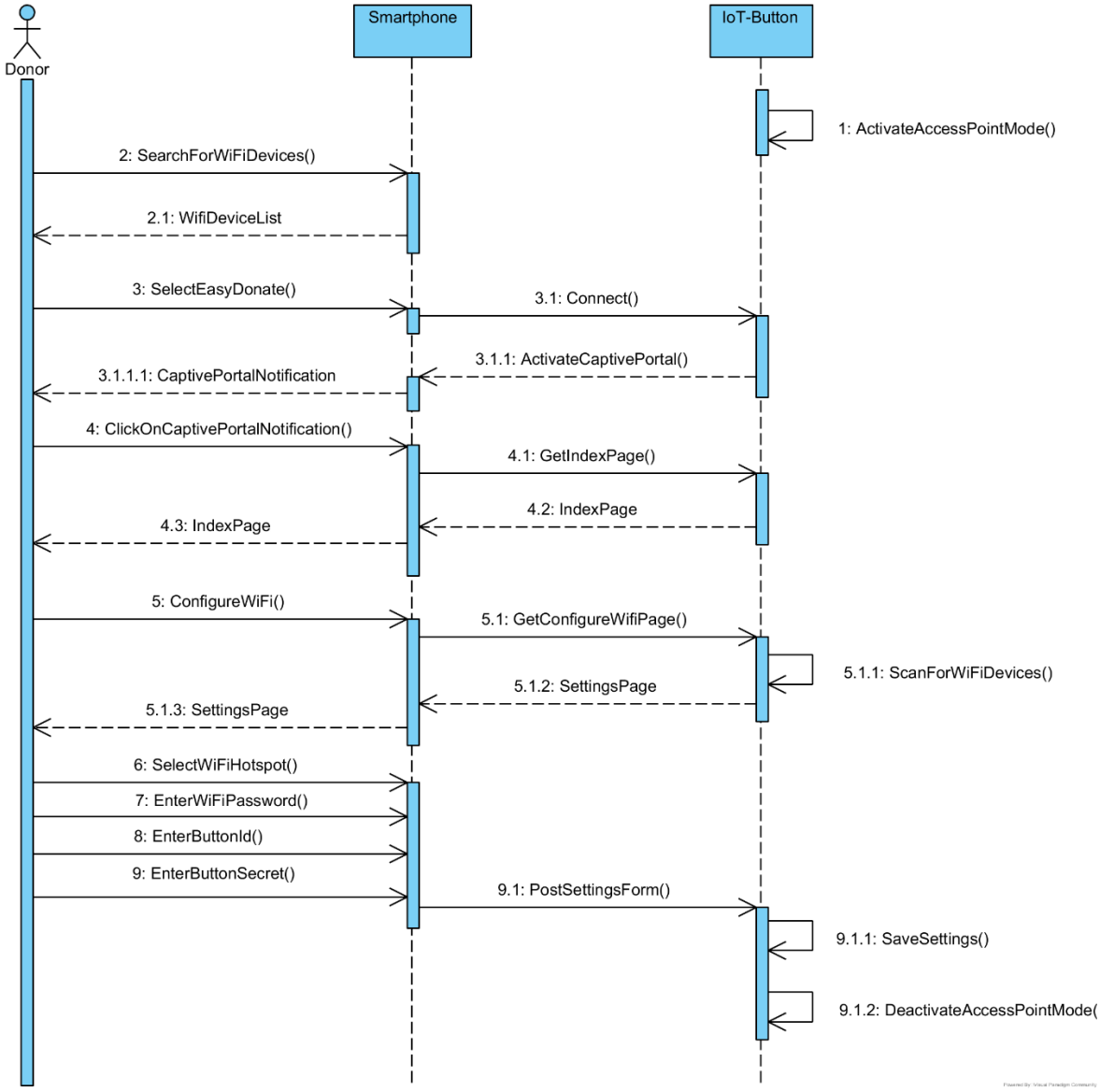

Abbildung 18 Detaillierter Ablauf der Konfiguration des Iot-Buttons

4.7.1.2 Spende Prozess durchführen

Wenn der Button nach erfolgreicher Erstkonfiguration gedrückt wird, ruft dieser einen Endpoint der Spendenplattform API auf und übergibt ihm seine Button-ID. Dieser generiert eine Nonce durch einen Krypto Zufallsgenerator und liefert diese an den Button zurück. Gleichzeitig wird die Nonce in der Datenbank abgespeichert und der Button-ID zugewiesen.

Der Button errechnet danach aus der Nonce und dem Secret einen HMAC (Keyed-Hash Message Authentication Code). Mit diesem HMAC und der Button-ID wird dann der Endpoint aufgerufen, über welchen die Spende Transaktion getriggert wird.

Der Button kann authentifiziert werden, indem der Webserver die Nonce und das Secret zu der Button-ID im Request von der Datenbank lädt und ebenfalls den HMAC berechnet. Wenn der errechnete Wert mit dem Wert im Request übereinstimmt wird der Spendenvorgang ausgelöst und die Nonce aus der Datenbank gelöscht. Der Ablauf des Spende Prozess wird in Abbildung 19 grafisch dargestellt.

Bei der Kommunikation handelt es sich also um eine klassische Verschlüsslung durch einen gemeinsamen Private Key (Secret), welche durch eine Nonce zusätzlich vor Replay Attacken geschützt wird.

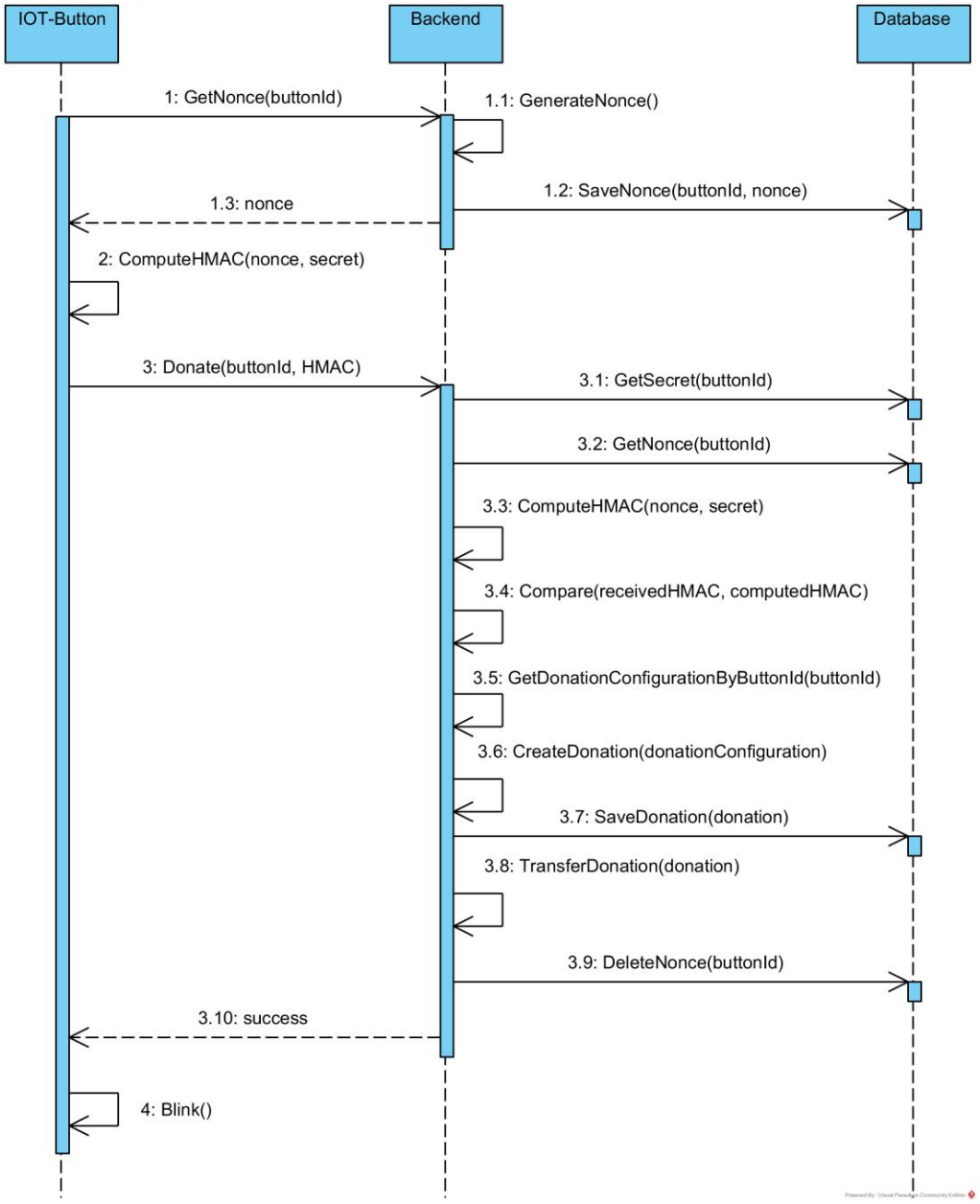

Abbildung 19 IoT-Button Spendeprozess

4.7.2 Registrierungsprozess

Der Registrierungsprozess von Spender und Organisation sind mehrheitlich identisch und unterscheiden sich lediglich in wenigen Details.

Bei beiden Usern wird zur Entität Spender oder Organisation zusätzlich ein Identity User erstellt, welcher von der entsprechenden Entität referenziert wird (Komposition).

Nach erfolgreicher Registrierung wird eine E-Mail mit einem Bestätigungslink versendet. Erst nachdem dieser Link bestätigt wurde, ist es möglich sich als User auf der Plattform anzumelden.

Der ganze Registrierungsprozess wird in einer Transaktion abgewickelt, damit keine inkonsistente Datensätze entstehen können.

4.7.2.1 Spender

Beim Spender wird zusätzlich eine Button Entität erzeugt sowie eine Spendenkonfiguration mit Default Werten für die Ausgabe-Limiten. So kann nach dem Login direkt eine Organisation ausgewählt werden und die «Button Credentials» stehen für das Einrichten des IoT-Buttons bereit.

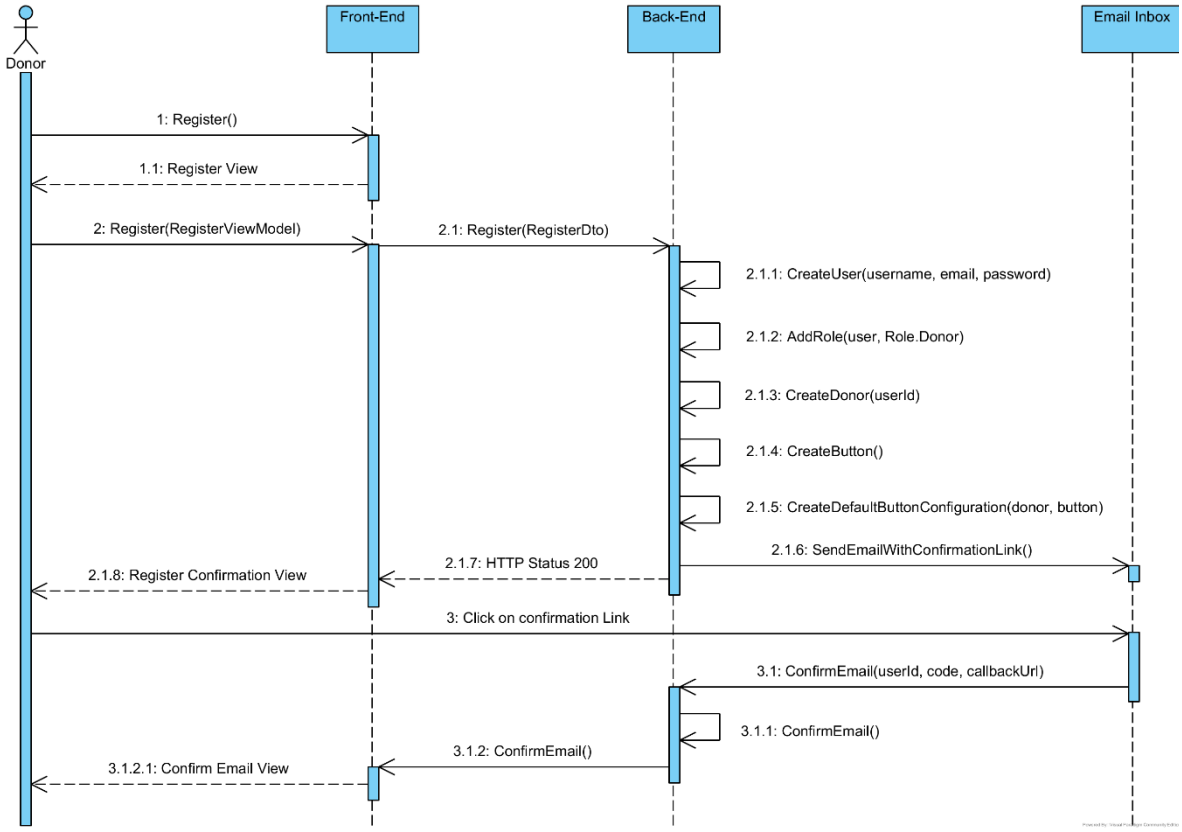

Abbildung 20 Registrierungsprozess eines Spenders

4.7.2.2 Organisation

Wenn sich eine Organisation erfolgreich registriert hat, ist es zusätzlich erforderlich, dass ein Administrator die Registrierungs-Anfrage einer Organisation frei gibt. Dies ist in Abbildung 21 nicht ersichtlich. Erst nach der Freigabe ist ein erfolgreiches Login auf dem Portal möglich. Dieser zusätzliche Schritt soll vor Missbrauch der Plattform schützen, da zunächst evaluiert wird, ob hinter der sich registrierenden Organisation eine «echte» gemeinnützige Organisation steckt.

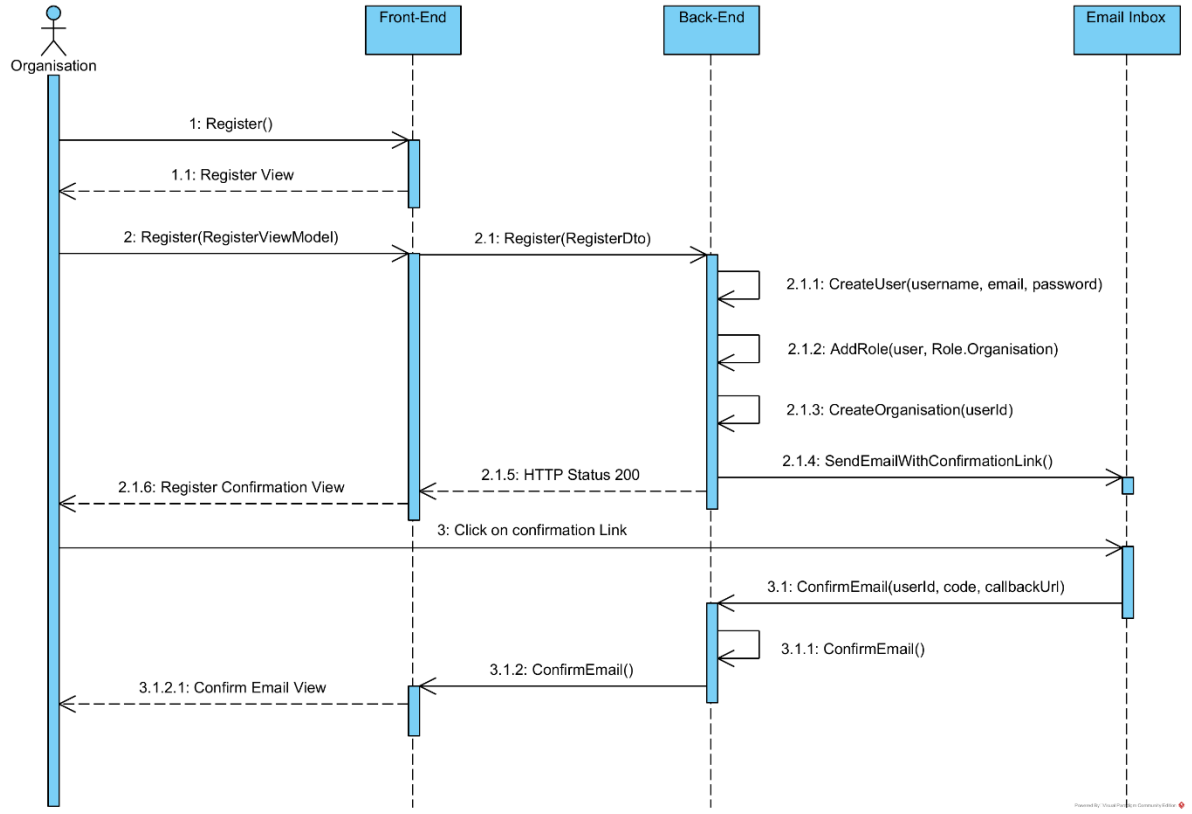

Abbildung 21 Registrierungsprozess einer Organisation

4.7.3 Login Prozess

Wenn beim Login das Passwort korrekt eingegeben wurde, wird ein Jwt mit «Claims» für den betreffenden User ausgestellt. Diese «Claims» beinhalten unter anderem Daten wir die User-ID oder die betreffende User Rolle (Spender, Organisation, Administrator). Dieses Jwt wird in der Frontend Komponente in der Session des Benutzers gespeichert und muss bei jedem Zugriff auf das Backend mitgeschickt werden.

Das Backend überprüft bei jedem Aufruf die Integrität und Gültigkeit des Jwt. Die Claims werden ausgelesen und anhand dessen entschieden ob der Zugriff auf eine Ressource gestattet ist oder durch einen «HTTP 401 Unauthorized» Error abgelehnt wird. Der Prozess ist in Abbildung 22 dargestellt.

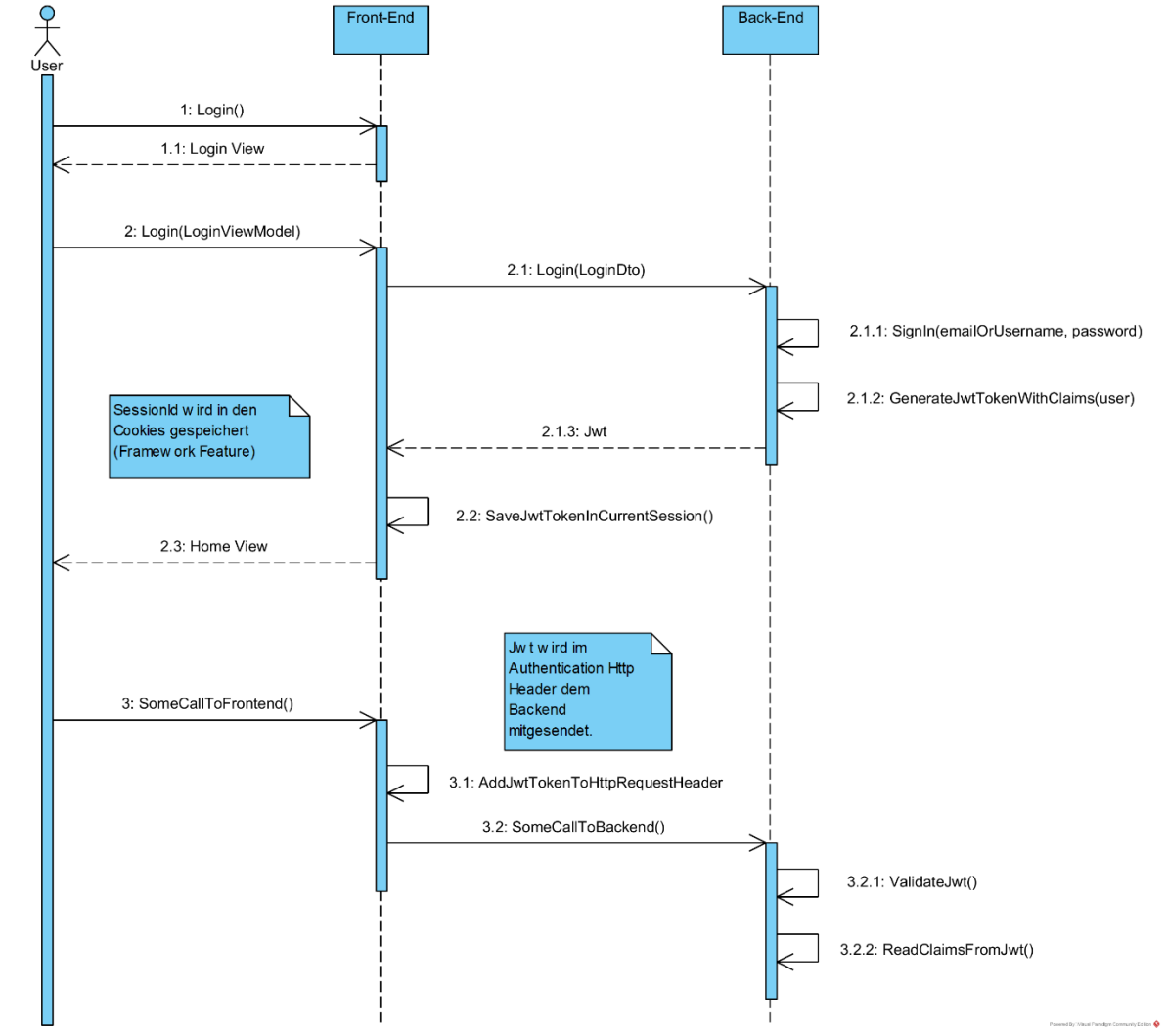

Abbildung 22 Login Prozess

4.8 Externe Schnittstellen und Libraries

4.8.1 Email Provider: Sendgrid

Für verschiedene Prozesse auf der Plattform (Registration Confirmation, Password Reset, etc.) wird auf Emails zur Kommunikation zwischen Plattformbetreiber und Benutzer gesetzt. Als Email Provider wird auf SendGrid (5) gesetzt. SendGrid wird in Version 9.9.0 eingesetzt. Mit einem gratis SendGrid Account stehen einem pro Monat 25'000 Mails zur Verfügung. Diese Anzahl ist für den Testbetrieb mehr als ausreichend und könnte bei einem kommerziellen Einsatz jederzeit durch eine Paid Subscription erhöht werden.

In Abbildung 23 wird der Einsatz des SendGrid API zum Versand einer Email anhand eines Beispiels aus dem Quick Start Tutorial beschrieben.

```
using SendGrid;
using SendGrid.Helpers.Mail;
using System;
using System.Threading.Tasks;

namespace Example
{
    internal class Example
    {
        private static void Main()
        {
            Execute().Wait();
        }

              static async Task Execute()
        {
            var apiKey =
Environment.GetEnvironmentVariable("NAME_OF_THE_ENVIRONMENT_VARIABLE_FOR_YOUR_SENDGRID_KEY"
);
            var client = new SendGridClient(apiKey);
            var msg = new SendGridMessage()
            {
                From = new EmailAddress("test@example.com", "DX Team"),
                Subject = "Sending with SendGrid is Fun",
                PlainTextContent = "and easy to do anywhere, even with C#",
                HtmlContent = "<strong>and easy to do anywhere, even with C#</strong>"
            };
            msg.AddTo(new EmailAddress("test@example.com", "Test User"));
            var response = await client.SendEmailAsync(msg);
        }
    }
}
```

Abbildung 23 Beispielcode für den Einsatz der SendGrid API zum asynchronen Versenden einer Email **(5)**

4.8.2 Charts.js

Charts.js (6) ist eine Open Source Library mit welcher einfache HTML5 Graphiken erstellt werden können. Für die Darstellung wird das Canvas Element verwendet. Aktuell werden 8 Arten von Diagrammen unterstützt, unter Anderen:

- Bar charts
- Line charts
- Area charts
- Pie/ Doughnut charts

Abbildung 4 zeigt ein Beispieldiagramm, wie es mit Hilfe der Library erstellt werden kann.

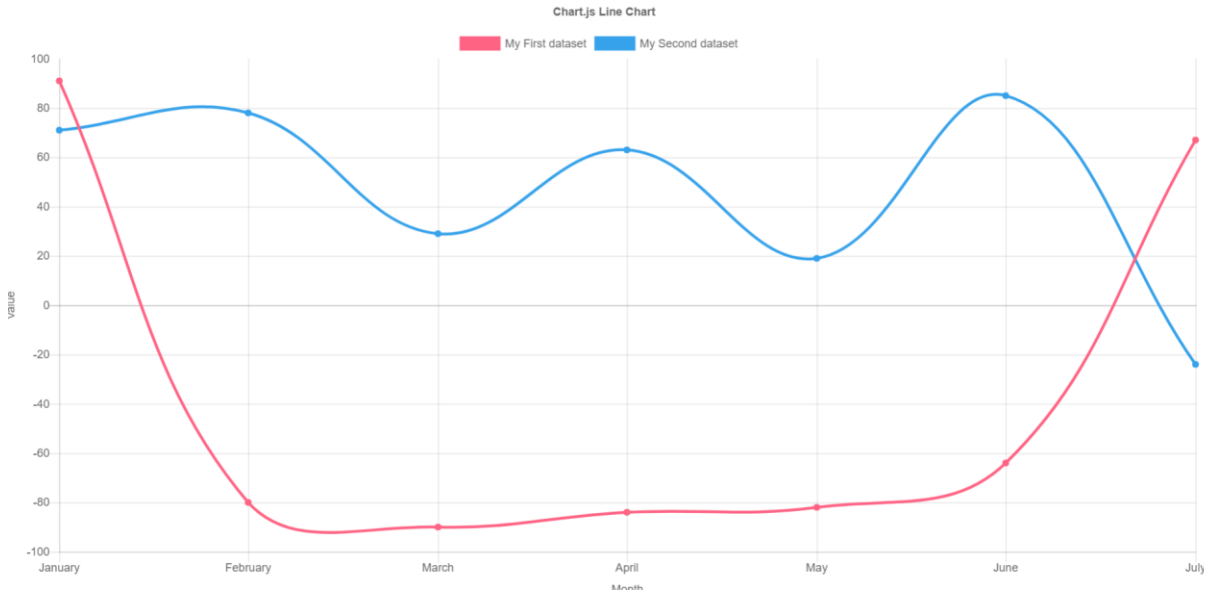

Abbildung 24 Beispieldiagramm von Charts.js **(6)**

Auf der EasyDonate Plattform wird Charts.js verwendet um die verschiedenen Diagramme im Zusammenhang mit dem Reporting der Statistiken zu erstellen.

4.9 Prozesse und Threads

4.9.1 Task-based Asynchronous Pattern (TAP)

Es wird stark auf das «async - await» Muster gesetzt. Dieses erlaubt es einen Task parallel zu starten und bei Terminierung des Tasks mit einem «await» Keyword fortzufahren (continuation).

Dadurch können mehrere Clients parallel bedient werden und Ressourcen werden nicht unnötig lange blockiert.

4.9.2 Schutz auf gemeinsame Ressourcen

Um den Zugriff auf gemeinsame Ressourcen zu schützen wird darauf geachtet, dass die Objekte mit der korrekten Lifetime im DI-Container registriert werden. Dadurch wird z.B. verhindert, dass sich Threads von unterschiedlichen Usern dieselben Objekte teilen.

Statische Klassen werden, wenn immer möglich vermieden. Auch aus Gründen der Testbarkeit.

Sollte es nicht möglich sein durch die Lifetime einen Zugriff auf gemeinsame Ressourcen zu verhindern, werden synchronisations-Mechanismen wie Locks, Semaphoren oder ähnliches verwendet.

Die parallelen Zugriffe auf die Datenbank werden durch das Transaktionsbasierte Verhalten des Datenbankmanagementsystems geregelt.

4.10 Persistierung

Für die Datenpersistierung wird eine relationale SQL Datenbank verwendet, welche über das Datenbankmanagementsystem Microsoft SQL Server verwaltet wird.

Eine relationale Datenbank ist ideal dafür geeignet um ein Datenmodell, welches aus vielen Beziehungen besteht abzubilden und zu persistieren.

Da die verwendete Azure Cloud ebenfalls von Microsoft betrieben wird ist eine reibungslose Integration der SQL Datenbank möglich.

Als ORM wird auf das Entity Framework Core gesetzt, weil damit der Code First Ansatz verwendet werden kann und eine optimale Integration in Asp.Net Core gewährleistet ist. Im Code First Ansatz werden zunächst die Entitäten als POCO's modelliert und daraus die entsprechenden Tabellen generiert.

4.10.1Datenstruktur

4.10.1.1 Business Entitäten

Die nachfolgende Abbildung zeigt die Struktur der Tabellen in der SQL Datenbank.

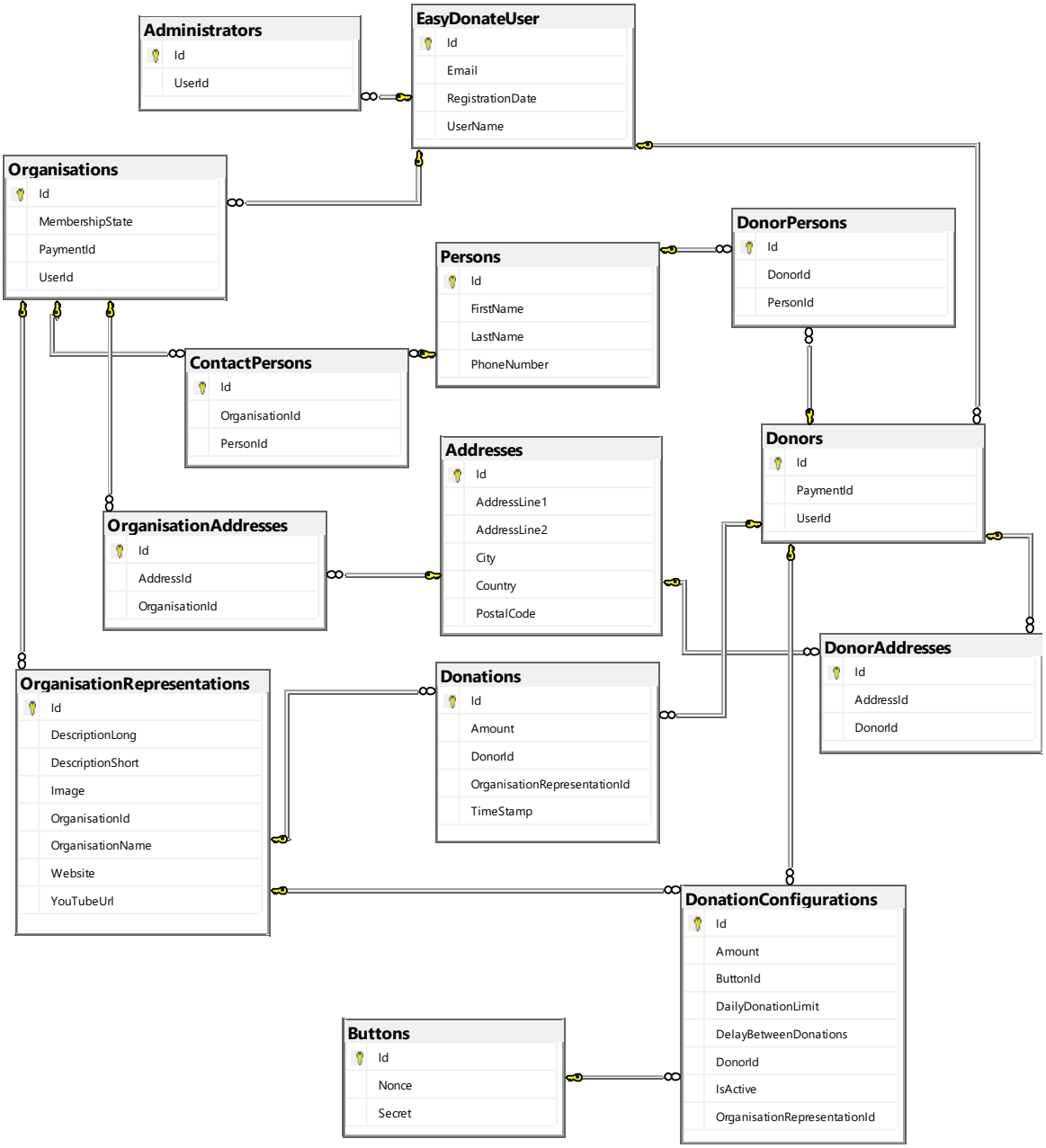

Abbildung 25 Datenbank Modell mit den Business Entitäten

4.10.1.2 Identity Framework Tabellen

Die Tabellen für das Identity Framework werden automatisch angelegt wenn der verwendete «DbContext» von «IdentityDbContext» ableitet. Diese Tabellen beinhalten die Benutzerverwaltung mit Rollen, Claims, usw.

Der Zugriff auf diese Tabellen erfolgt über das Identity Framework.

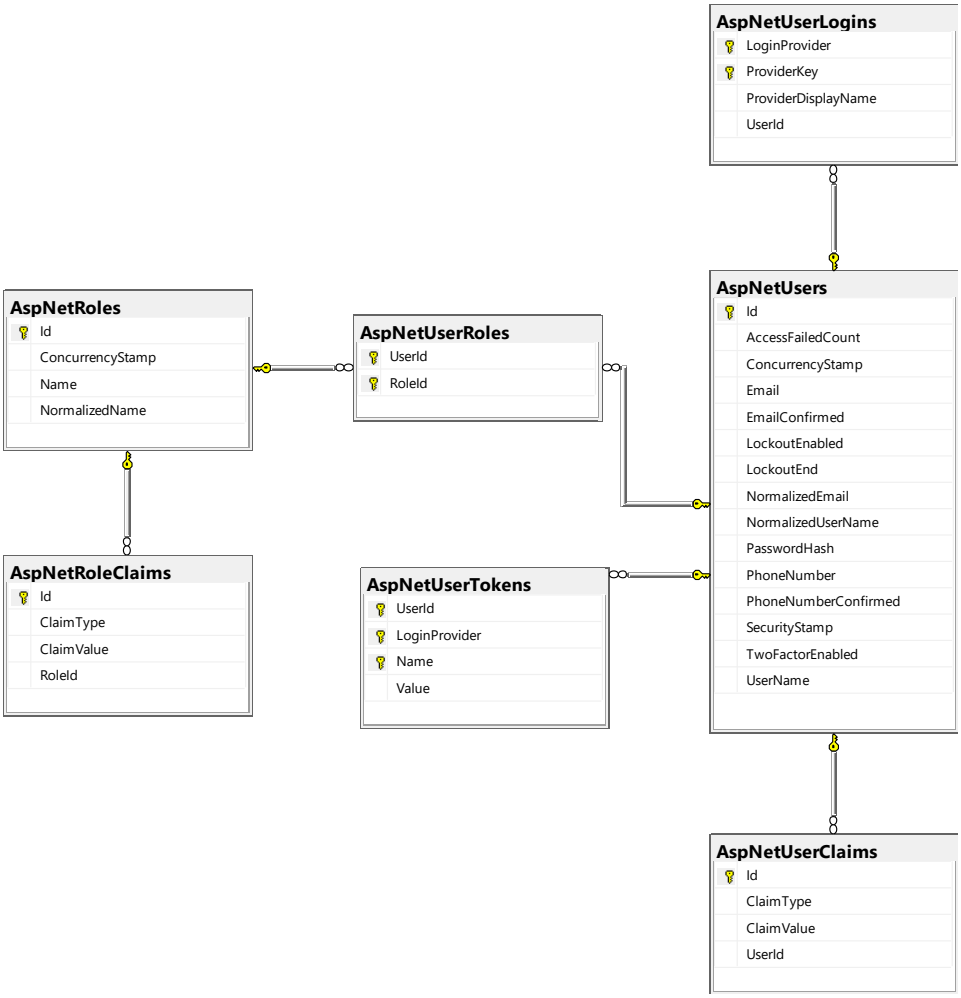

Abbildung 26 Datenbank Modell mit den Identity Framework Entitäten

5 Anwendungssicherheit

5.1 Risiken

Im Allgemeinen sind viele Szenarien mit möglichen Risiken identifizierbar. Im Rahmen dieser Arbeit wurde aber insbesondere die Zahlungsabwicklung beim Spende Vorgang als kritisch erachtet. Beim Umgang mit Zahlungsinformationen und dem Durchführen von Zahlungen muss ein spezielles Augenmerk auf die Sicherheit der Daten gelegt werden. Die Handhabung der Zahlungsinformationen wird umgangen, indem ein externer Payment Provider (sieh auch 4.6.4) damit beauftragt wird. Zudem wird im produktiven Betrieb auf HTTPS gesetzt um eine abhörsichere Übertragung von Daten zu ermöglichen. Weiterhin sind aber die Möglichkeit einer Replay Attacke und der Durchführung einer unautorisierten Spende im Namen eines anderen Spenders denkbar. Diese Szenarien werden nachfolgend diskutiert.

5.1.1 Replay Attacken

Bei einer Replay Attacke wird ein Request aufgezeichnet und erneut abgesetzt. Es wäre also denkbar, dass im Spende Prozess der Request des IoT-Button an den Server Endpoint aufgezeichnet und wiederholt abgesetzt wird. Dadurch könnten ungewollte Spenden zu Lasten des Spenders ausführt werden.

5.1.2 Durchführung einer nicht autorisierten Spende

Eine nicht autorisierte Person könnte versuchen den Endpoint für den Spende Prozess auszulösen, wenn dieser lediglich über eine eindeutige Kennung einem Spender zugewiesen wäre. Es wäre beispielsweise denkbar den Server mit einem Brute Force Angriff zu bombardieren, in der Hoffnung einen gültigen Spendenvorgang auszulösen.

5.1.3 Massnahmen

Um den diskutierten Risiken im Zusammenhang mit dem Spende Prozess vorzubeugen wurde eine Verschlüsselung der Kommunikation zwischen IoT-Button und REST API auf Basis eines Challenge Response-Verfahrens mit Secret Key implementiert. Dieser Prozess wird im Kapitel 4.7.1.2 im Detail erklärt.

6 Deployment

Das Deployment der Webapplikation (Frontend und Backend) ist automatisiert und setzt auf einen Continuous Deployment Prozess. Hierzu wird die komplette Prozesskette in VSTS (7) abgebildet. Abbildung 27 visualisiert die vereinfachte Abfolge der Schritte in der Prozesskette. Die blauen Schritte laufen vollständig automatisiert ab, während die grünen Schritte den Eingriff eines Entwicklers erfordern.

Abbildung 27 Continuous Deployment Pipeline

Gestartet wird die Kette durch ein Commit eines Entwicklers und dem Durchführen eines anschliessenden Pull Requests. Der Pull Request löst einen Build und einen anschliessende Durchführung der Unit Tests (Test) aus. Gleichzeitig werden ein oder mehrere ausgewählte Entwickler zum Durchführen eines Code Reviews aufgefordert. Die Reviewer haben nun die Möglichkeit den Pull Request anzunehmen (Approve) oder den Commit zur Überarbeitung an den Requester zurück zu weisen. Nach einer Überarbeitung und der Freigabe durch den Reviewer wird ein finaler Build inkl. Testdurchführung angestossen. Die Artefakte des Builds werden durch eine Realease auf das Zielsystem deployt.

Abbildung 28 illustriert einen Auszug aus der Build-Definition in den VSTS Konfigurationen. Die Durchführung des Ersten vom Zweiten Build in der beschriebenen Prozesskette unterscheidet sich lediglich durch den Schritt „Publish Artifact“, welcher nur vor dem Release durchgeführt wird.

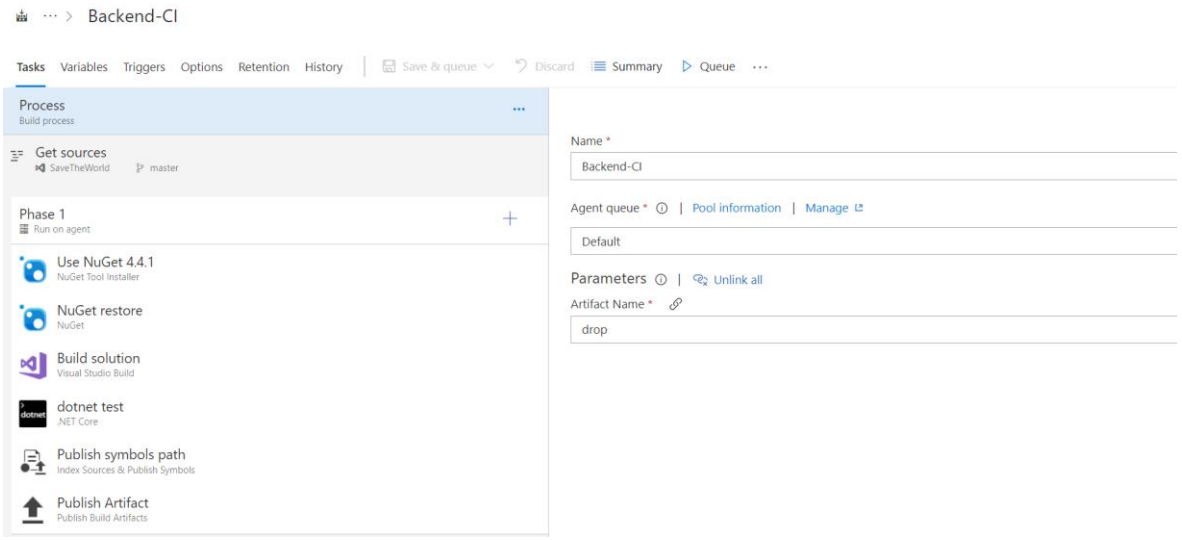

Abbildung 28 Build-Definition für das Backend

In Abbildung 29 wird die Release-Definition für das Backend dargestellt. Der Release besteht aus einem einzigen Schritt, in welchem die vorbereiteten Artefakte auf die Azure Umgebung deployt werden.

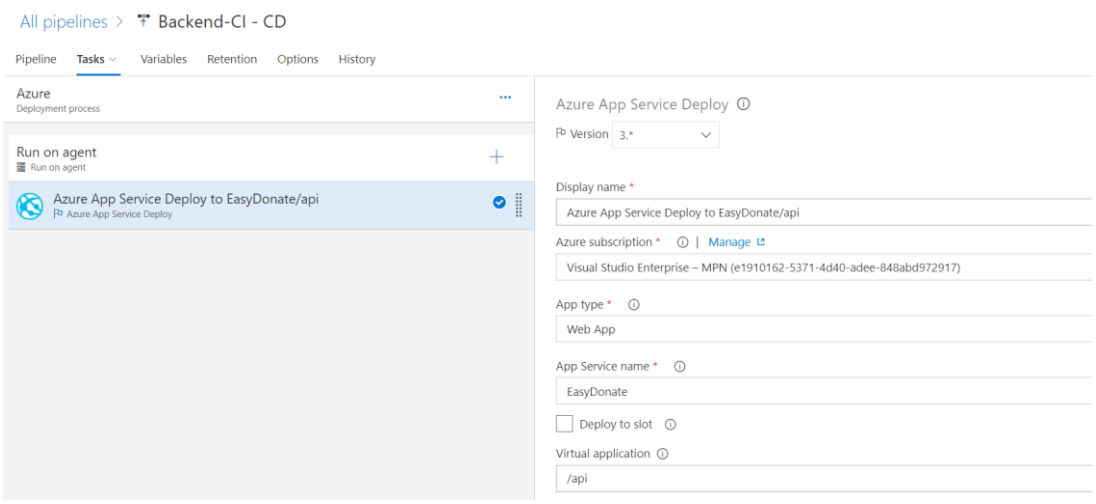

Abbildung 29 Release-Definition für das Backend mit Azure App Service Deployment Schritt

Für das Frontend sind im VSTS analoge Build- und Release-Definitionen definiert.

7 Realisierte Lösung - EasyDonate

Das realisierte Produkt lautet auf den Namen EasyDonate. Die Umsetzung der Use Cases aus Kapitel 3 wird in den folgenden Kapiteln aufgezeigt. Es handelt sich dabei um Auszüge aus dem für das Produkt erstellten Benutzerhandbuch. Die Formulierung richtet sich dabei immer an einen Endnutzer.

7.1 Kurzbeschrieb

Ziel der Webapplikation EasyDonate ist es, potentielle Spender mit gemeinnützigen Organisationen in Verbindung zu bringen, sodass sie diese finanziell unterstützen können. Im Fokus steht dabei die Einfachheit des Spendenvorgangs, welcher durch einen Internet of Things (IoT)-Button ausgelöst wird.

7.2 Einstieg auf die Plattform

Sie gelangen über die URL http://easydonate.azurewebsites.net/ auf die EasyDonate Plattform.

7.2.1 Funktionen im Überblick

Die Plattform kann aus der Sicht von vier unterschiedlichen Benutzergruppen aufgerufen werden (Gast, Spender, Organisation und Administrator). Die zur Verfügung stehende Funktionalität ist dabei von der Benutzergruppe abhängig.

Als Gast können Sie sich die Organisationen auf der Plattform ansehen, Globale Spendenstatistiken betrachten oder sich selbst als Spender oder Organisation registrieren.

Als Spender können Sie sich ein Spendenprofil einrichten und so Ihre bevorzugte Organisation mit jeder Betätigung Ihres IoT-Button unterstützen. Falls Sie noch keinen eigenen IoT-Button besitzen können Sie diesen direkt auf der Plattform bestellen.

Als Organisation können Sie die Spender dazu mobilisieren Sie als Organisation finanziell zu unterstützen. Dies erreichen Sie durch eine ansprechende Präsentation Ihrer Organisation auf unserer Plattform

Als Administrator können Sie Mitgliedschaftsanfragen von Organisationen Verwalten und weitere Administratoren Benutzerkonten erstellen.

In den nachfolgenden Kapiteln werden Benutzergruppen spezifischen Funktionalitäten im Detail erläutert und deren Benutzung mit bebilderten Anleitungen aufgezeigt.

7.3 Benutzung als Gast

Als Gast gilt ein nicht registrierter Benutzer der Plattform. Alle Funktionen, die dem Gast zur Verfügung stehen, können auch von den weiteren Benutzergruppen genutzt werden.

7.3.1 Startseite

Die Startseite ist der Einstiegspunkt für alle Nutzer. Von hier aus werden alle weiteren Funktionalitäten aufgerufen. Über ein Sliding Carousel wird ein Überblick über die Funktionen der Plattform gegeben. Es werden drei zufällig gewählte Organisationen als mögliche Spendenempfänger dargestellt. Des Weiteren informiert ein Echtzeit Feed über eingehende Spenden.

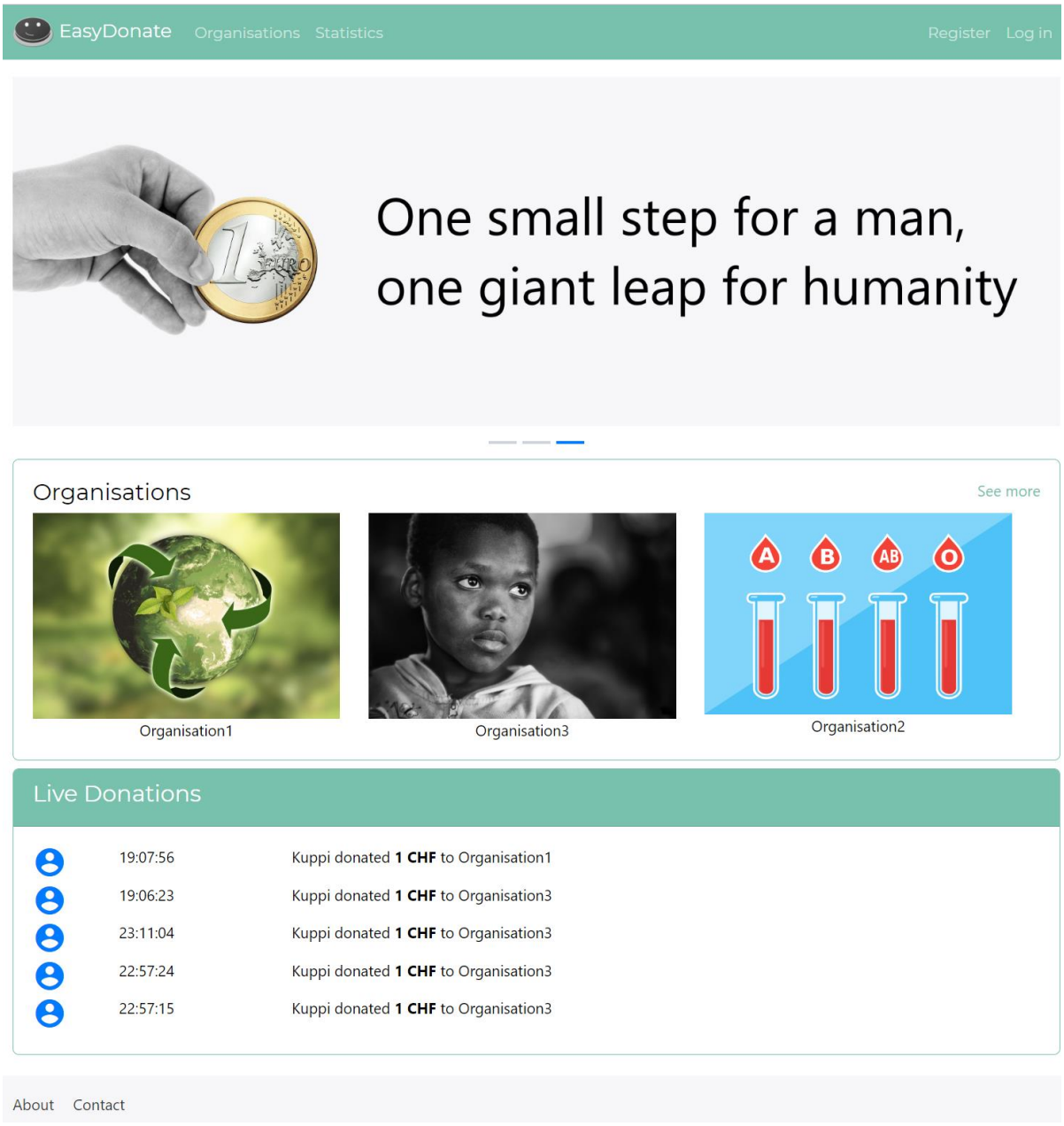

Abbildung 30 Startseite der EasyDonate Plattform

7.3.2 Organisationen ansehen

Über den Link «Organisations» in der Navigationsleiste gelangen Sie zu einer Übersicht über die Organisationen auf der Plattform. Als registrierter Spender können Sie diese Organisationen finanziell unterstützen.

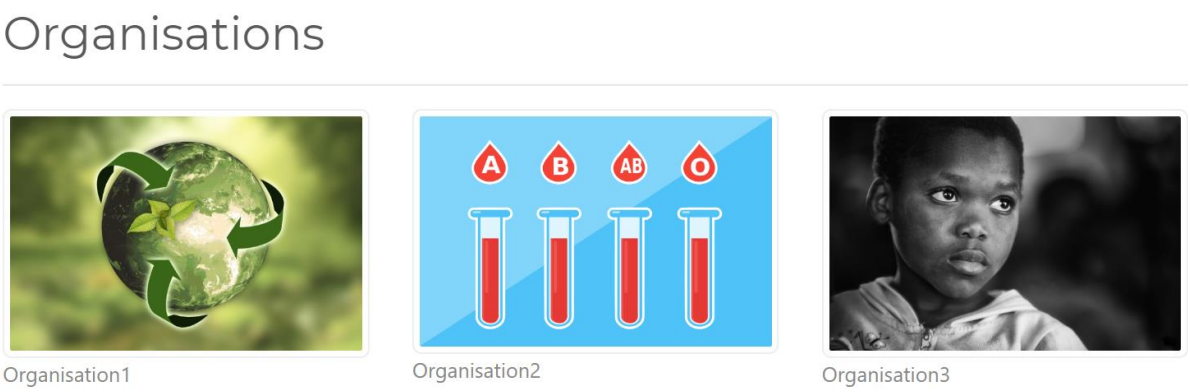

Abbildung 31 Übersicht Organisationen

Wenn Sie mehr über eine Organisation erfahren möchten können Sie die Detailansicht aufrufen indem Sie auf das Bild der Organisation klicken.

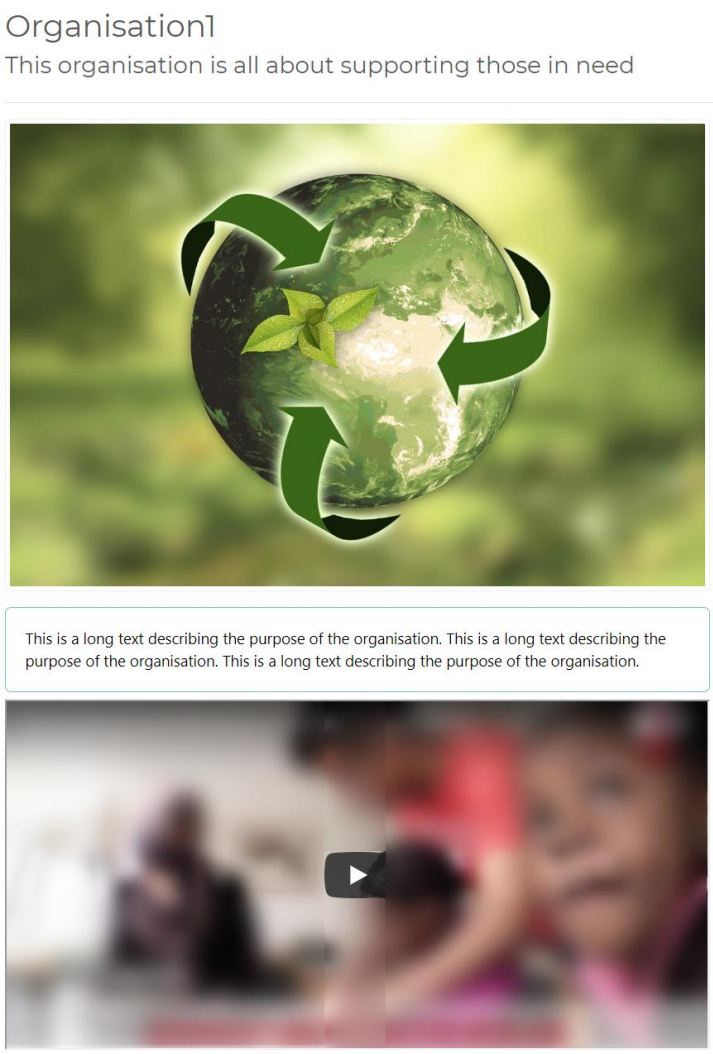

Abbildung 32 Übersicht Organisationen

In der Detailansicht sehen Sie das Anzeigebild, eine Beschreibung und ein Video zur Organisation.

7.3.3 Statistiken ansehen

Über den Link «Statistics» in der Navigationsleiste gelangen Sie zu einer Übersicht über diverse Statistiken zu den globalen Spende-Ereignissen und dem Wachstum der Plattform.

Die Datumsangaben werden dabei jeweils im Format MM/DD/YYYY angezeigt.

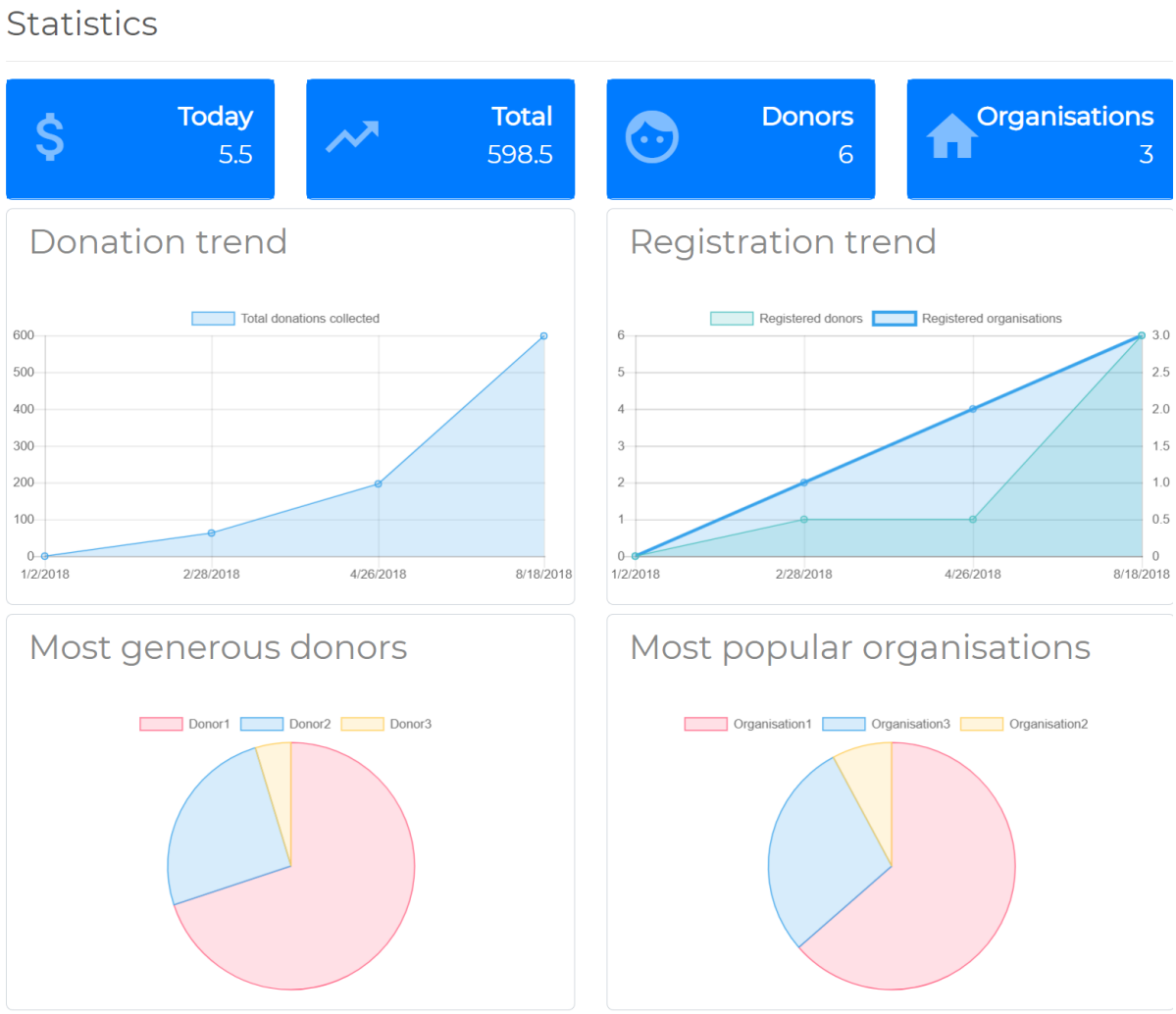

Abbildung 33 Globale Spendenstatistiken

7.3.3.1 Gesamtübersicht

Die Gesamtübersicht gibt mit wenigen Kennzahlen einen Überblick über die globalen Spende-Aktivitäten und die Grösse der Plattform gemessen an den Mitgliederzahlen.

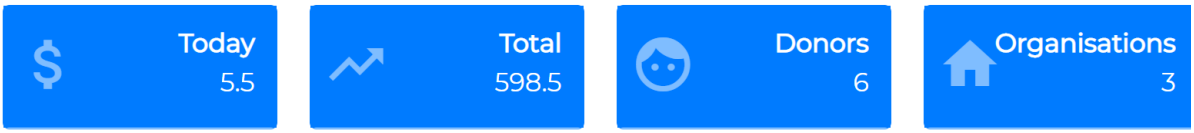

Abbildung 34 Gesamtübersicht Anzeige

Anzeigeelement	Beschreibung
Today	Totalbetrag aller eingegangenen Spenden heute.
Total	Totalbetrag aller eingegangenen Spenden seit Inbetriebnahme der Plattform.
Donors	Anzahl aktiver Spender auf der Plattform.
Organisations	Anzahl Organisationen welchen gespendet werden kann.

Tabelle 16 Erklärung Anzeigeelemente der Gesamtübersicht

7.3.3.2 Spende Verlauf

In dem Diagramm «Donation trend» wird der Verlauf der eingegangenen Spenden seit Inbetriebnahme der Plattform bis zum aktuellen Datum angezeigt.

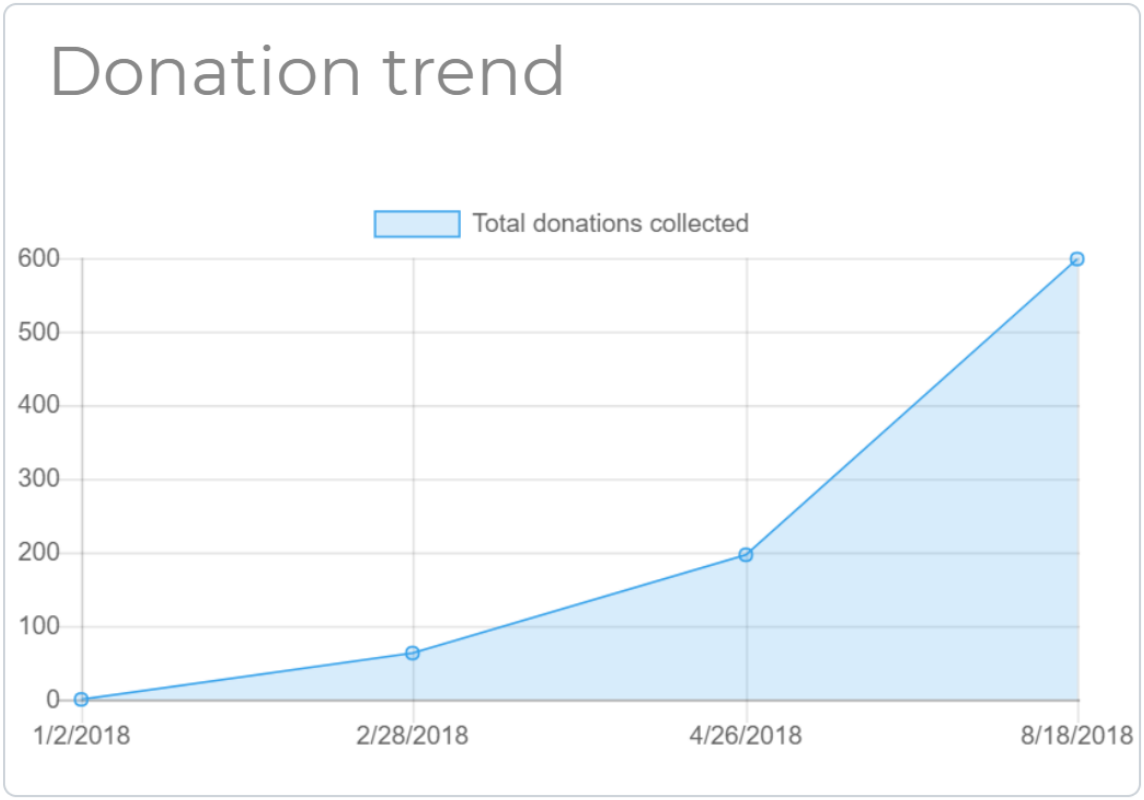

Abbildung 35 Diagramm Spende Verlauf

7.3.3.3 Registration Verlauf

In dem Diagramm «Registration trend» wird der Verlauf der Anzahl registrierter Spender und Organisationen seit Inbetriebnahme der Plattform bis zum aktuellen Datum aufgezeigt.

Abbildung 36 Diagramm Registration Verlauf

7.3.3.4 Grosszügigste Spender

In dem Diagramm «Most generous donors» werden die Grosszügigsten angezeigt. Für die Auswertung wird die Summe aller eingegangenen Spenden pro Spender berücksichtigt.

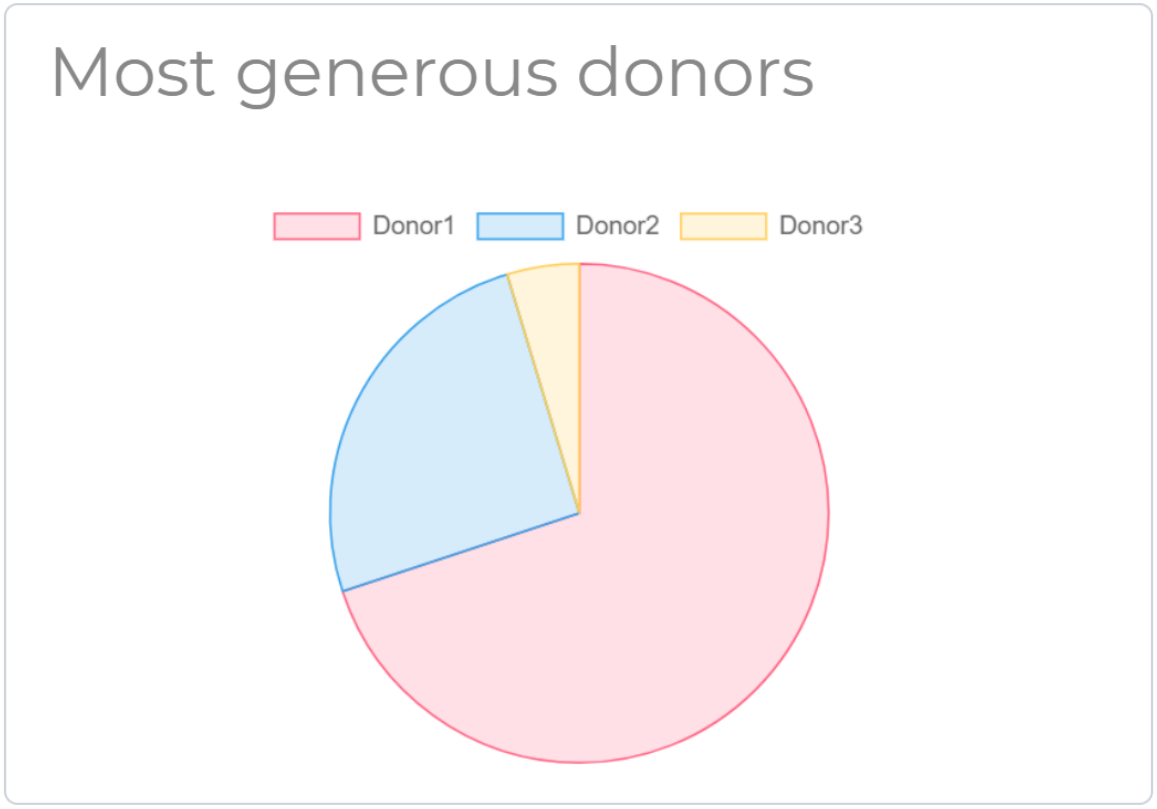

Abbildung 37 Diagramm Grosszügigste Spender

7.3.3.5 Populärste Organisationen

Das Diagramm «Most popular organisations» zeigt die Organisationen auf, welche sich der grössten Unterstützung erfreuen. Für die Auswertung wird die Summe aller eingegangenen Spenden pro Organisation berücksichtigt.

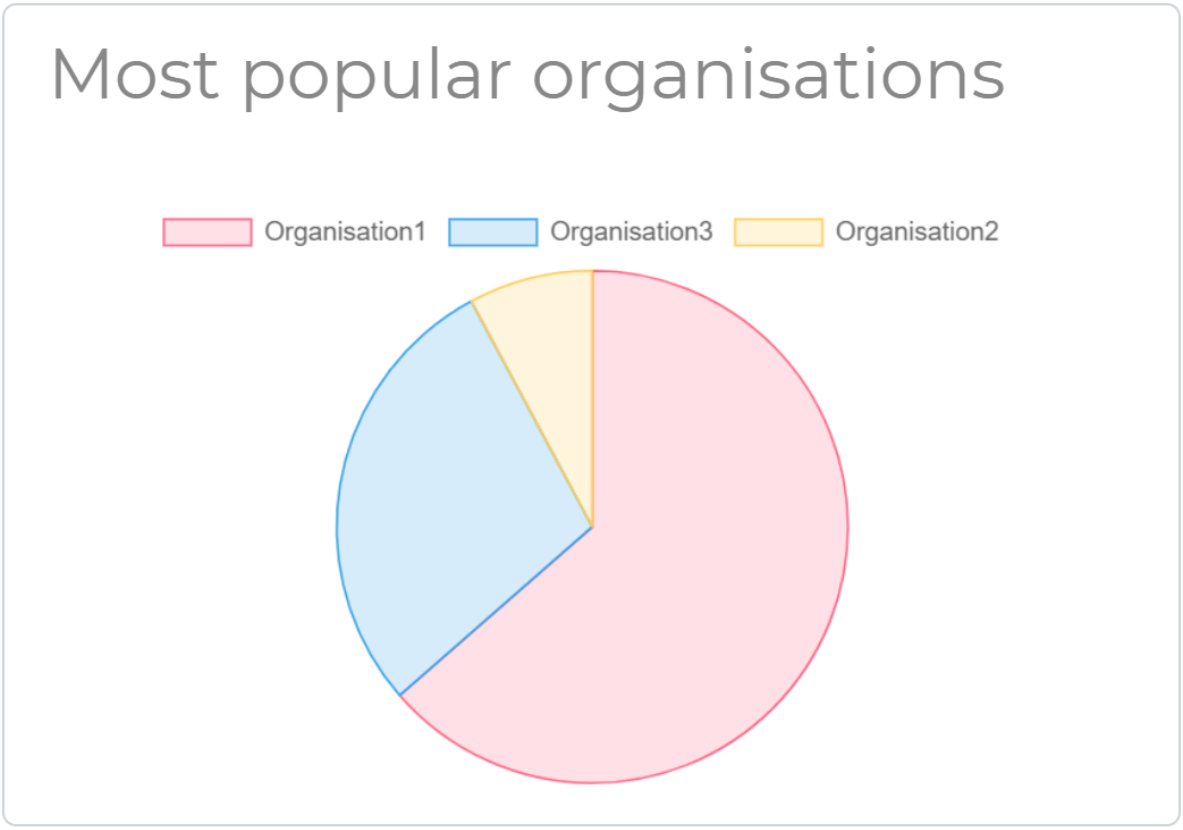

Abbildung 38 Diagramm populärste Organisationen

7.3.4 Registrierung als Spender

In der Navigationsleiste unter «Register» können Sie sich als Spender registrieren, indem Sie den Reiter «Donor» auswählen. Geben Sie Ihren gewünschten Benutzernamen und Ihre E-Mail-Adresse ein. Beachten Sie, dass Ihr persönliches Passwort aus mindestens 8 Zeichen besteht, mindestens ein Klein – und einen Grossbuchstaben, sowie mindestens eine Zahl und ein Sonderzeichen enthalten muss.

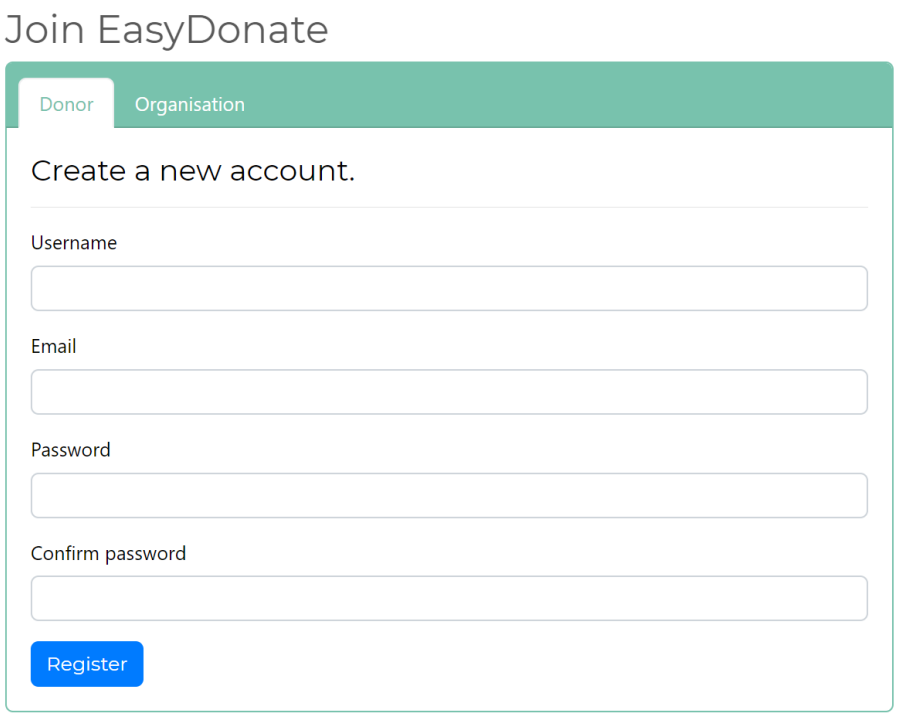

Abbildung 39 Registrierungsformular für Spender

Wenn die Registrierung erfolgreich war, sollten Sie eine Bestätigung sehen.

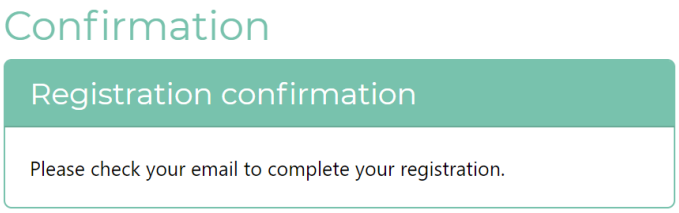

Abbildung 40 E-Mail Versand Bestätigung

Ebenfalls sollten Sie nun eine E-Mail mit einem Bestätigungslink erhalten haben.

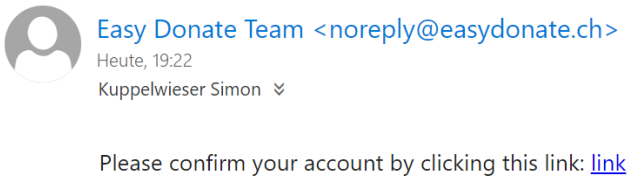

Abbildung 41 E-Mail mit Bestätigungslink

Sobald Sie auf diesen Link klicken erscheint eine Bestätigungsmeldung und Sie können die Dienste der Plattform in Anspruch nehmen.

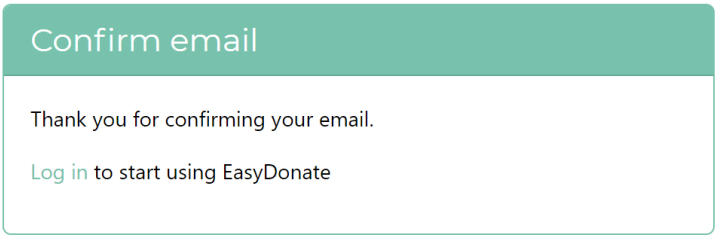

Abbildung 42 E-Mail Bestätigung

7.3.5 Registrierung als Organisation

In der Navigationsleiste unter «Register» können Sie sich als Organisation registrieren, indem Sie den Reiter «Organisation» auswählen. Geben Sie Ihren gewünschten Benutzernamen und Ihre E-Mail-Adresse ein. Ebenfalls sind einige Informationen über Ihr Unternehmen erforderlich, damit wir Sie persönlich Kontaktieren können.

Beachten Sie, dass Ihr persönliches Passwort aus mindestens 8 Zeichen besteht, mindestens einen Klein – und einen Grossbuchstaben, sowie mindestens eine Zahl und ein Sonderzeichen enthält.

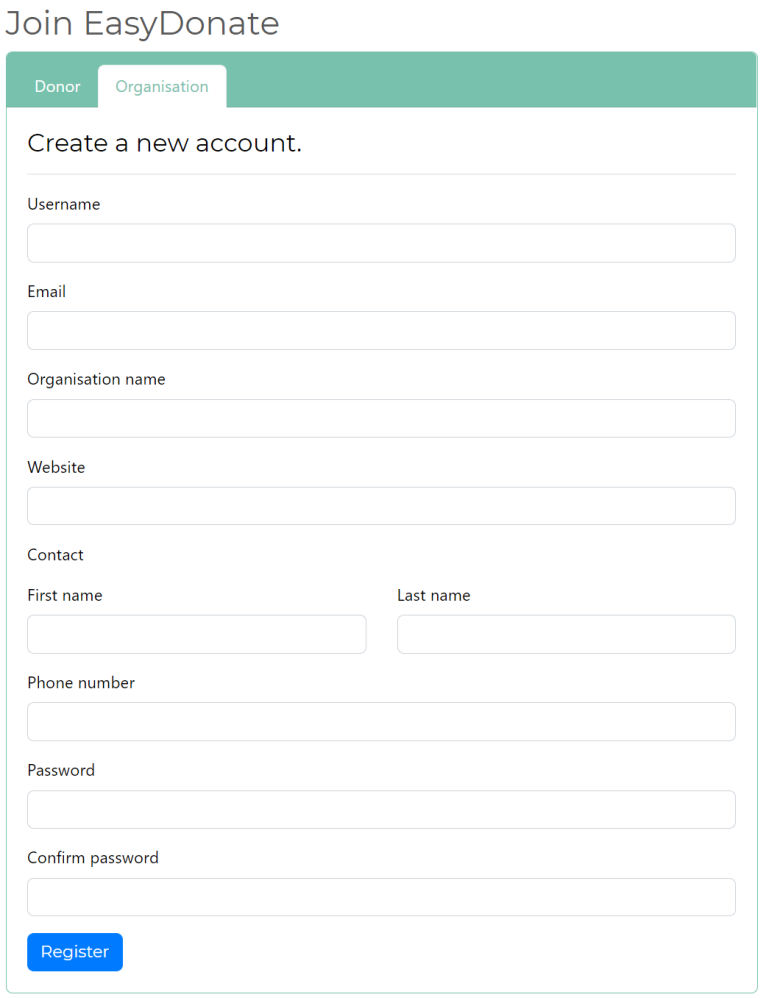

Abbildung 43 Registrierungsformular für Organisationen

Wenn die Registrierung erfolgreich war, sollten Sie eine Bestätigung sehen.

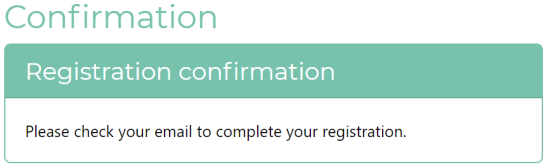

Abbildung 44 E-Mail Versand Bestätigung

Ebenfalls sollten Sie nun eine E-Mail mit einem Bestätigungslink erhalten haben.

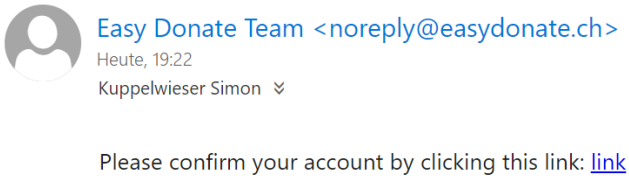

Abbildung 45 E-Mail mit Bestätigungslink

Sobald Sie auf diesen Link klicken erscheint eine Bestätigungsmeldung. Ab diesem Zeitpunkt können Sie sich einloggen und Ihr Profil auf der Plattform pflegen.

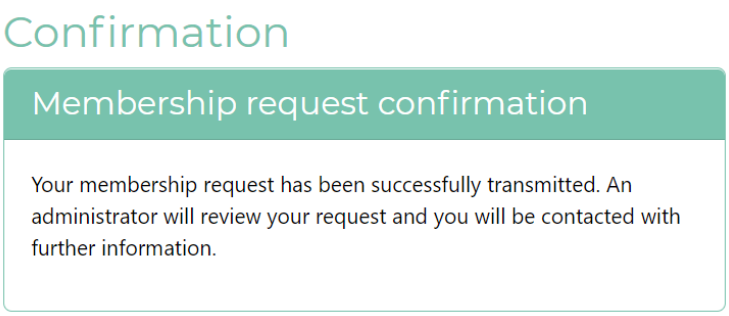

Abbildung 46 Bestätigung Mitgliedschaft wird geprüft

Ihre Mitgliedschaft wird nun von uns geprüft und wir werden Sie Kontaktieren. Sobald wir Sie offiziell als Organisation aufgenommen haben erhalten Sie ein Bestätigungsemail. Ab diesem Zeitpunkt kann Ihre Organisation öffentlich eingesehen werden und Sie können Spenden empfangen.

7.4 Benutzung als User

Als User zählen die drei registrierten Benutzergruppen Spender, Organisation und Administrator.

7.4.1 Einloggen

Wenn Sie sich erfolgreich als Spender (Kapitel 7.3.4) oder Organisation (Kapitel 7.3.5) registriert haben oder Sie ein Administrator sind, können sie sich in der Navigationsleiste unter «Log in» einloggen.

Falls Sie Ihr Passwort vergessen haben, klicken Sie auf den Link «Forgot your password?» auf der rechten Seite und geben Sie Ihre E-Mail-Adresse ein. Es wird Ihnen ein Link zugesendet, mit dem Sie Ihr Passwort zurücksetzen können.

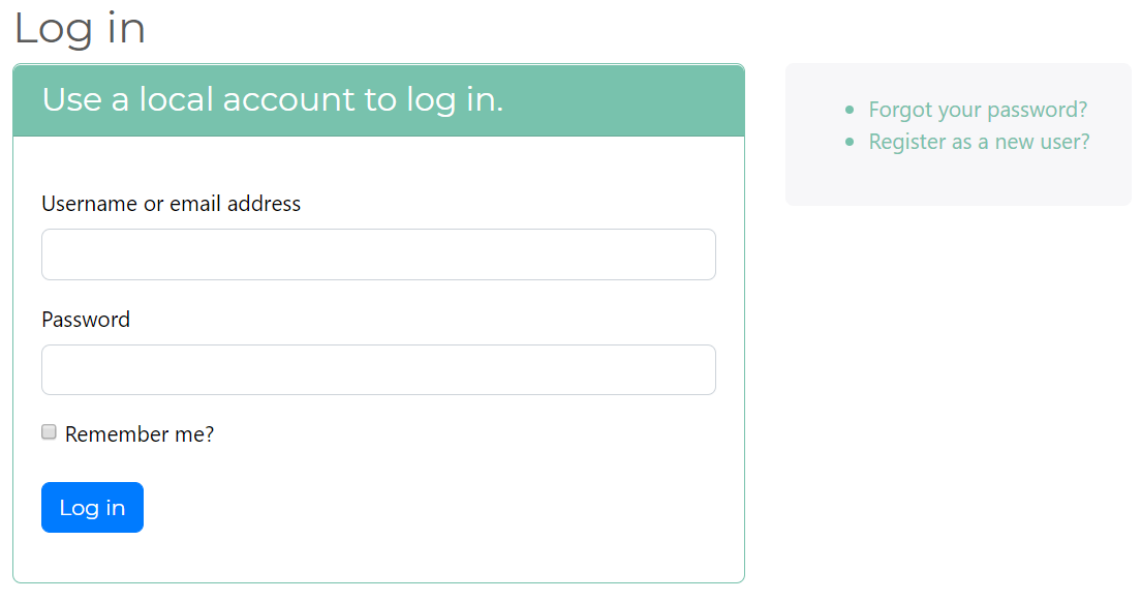

Abbildung 47 Login Formular

7.4.2 Passwort ändern

Ihr persönliches Passwort können Sie jederzeit in den Profileinstellungen unter «Change password» ändern. Dabei geben Sie ihr Altes - und das gewünschte neue Passwort ein. Zur Bestätigung geben Sie Ihr neues Passwort ein zweites Mal ein. Über die Schaltfläche «Update» bestätigen Sie die Änderung.

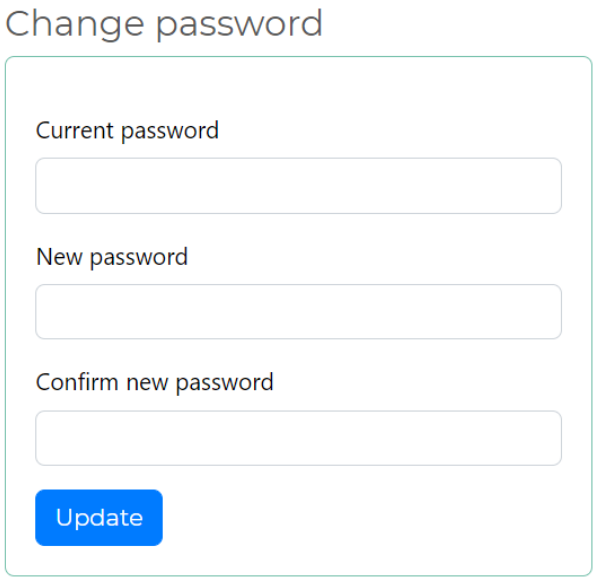

Abbildung 48 Formular um das Passwort zu ändern

Wenn Ihr Passwort erfolgreich geändert wurde, wird dies durch eine Bestätigungsmeldung signalisiert.

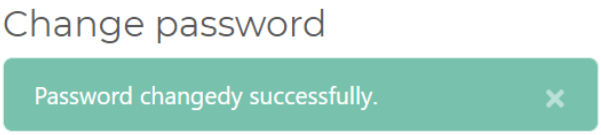

Abbildung 49 Bestätigungsmeldung: Passwort wurde erfolgreich aktualisiert

7.4.3 Kontaktinformationen anpassen

Ihre Kontakt-Informationen können Sie jederzeit in den Profileinstellungen unter «Contact information» ändern. Über die Schaltfläche «Update» speichern Sie die Änderungen.

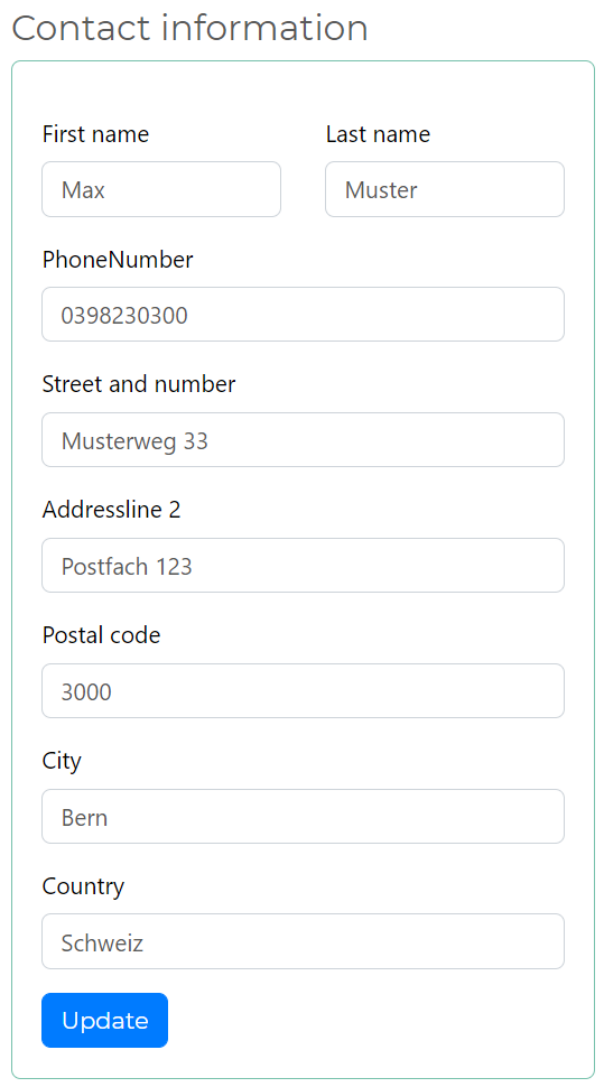

Abbildung 50 Formular für die Anzeige und die Aktualisierung der Kontaktinformationen

7.5 Benutzung als Spender

7.5.1 Profil Verwalten

Ihr persönliches Profil können Sie in der Navigationsleiste unter «Settings» verwalten. Auf der linken Navigationsleiste können Sie die einzelnen Einstellungsgruppen verwalten. Die jeweils aktive wird dabei grün hinterlegt. «Change password» und «Contact information» wurde bereits in den Kapiteln 7.4.2 und 7.4.3 beschrieben.

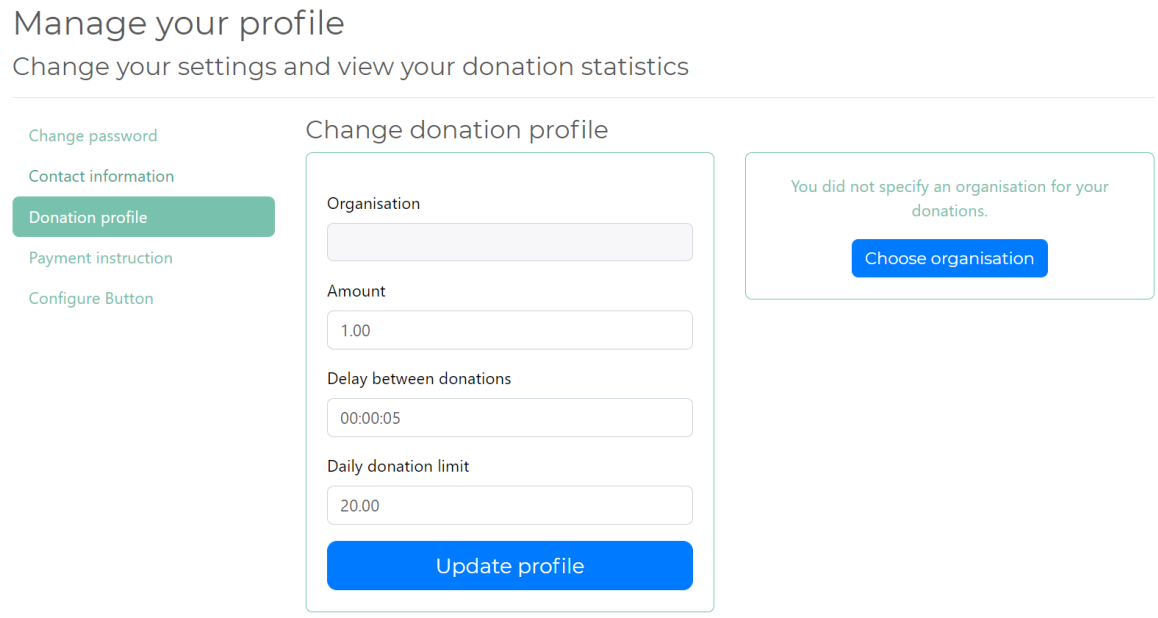

Abbildung 51 Profil Verwaltung

7.5.1.1 Spendenkonfiguration erstellen

Ihre Spendenkonfiguration können Sie in Ihren Profileinstellungen unter «Donation profile» verwalten. Es stehen Ihnen dabei die in Tabelle 17 beschriebenen Parameter zur Verfügung. Mit «Update profile» wird das Profil mit den eingegebenen Daten aktualisiert. Eine Beschreibung zu den Parametern finden Sie in Tabelle 17.

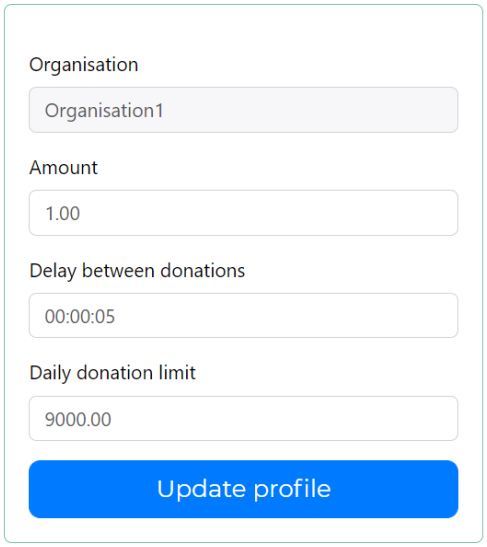

Abbildung 52 Formular für die Spende Konfiguration

Einstellung	Beschreibung
Amount	Betrag welcher pro Betätigung des IoT-Buttons gespendet wird.
Delay between donations	Zeit welche zwischen 2 Spende Vorgängen mindestens vergehen muss, bis die Spende ausgeführt wird.
Daily donation limit	Maximaler Betrag, welcher an einem einzelnen Tag gespendet werden kann.

Tabelle 17 Spendenkonfiguration Parameter

In der aktuellen Version kann die Empfängerorganisation der Spenden nicht über das «Donation profile» aktualisiert werden. Um die Empfängerorganisation zu wechseln muss zur Detailansicht der Organisationen gewechselt werden.

7.5.1.2 Empfängerorganisation konfigurieren

Nun können Sie in der Navigationsleiste unter «Organisations» eine Organisation auswählen welche Sie gerne finanziell unterstützen möchten.

Wenn Sie eine Organisation anklicken sehen Sie oben rechts ein durchgestrichenes Dollarzeichen, wenn diese Organisation nicht in Ihrem Spendenprofil als aktuelle Organisation eingestellt ist.

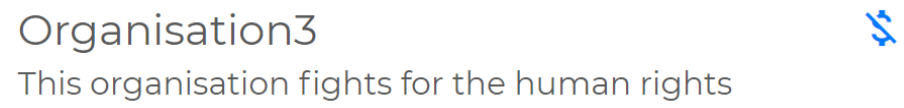

Abbildung 53 Durchgestrichenes Dollarzeichen in der Überschrift

Wenn Sie nun diese Organisation auswählen möchten, klicken Sie auf das durchgestrichene Dollarzeichen und bestätigen das Pop-Up Info Fenster mit „Yes“.

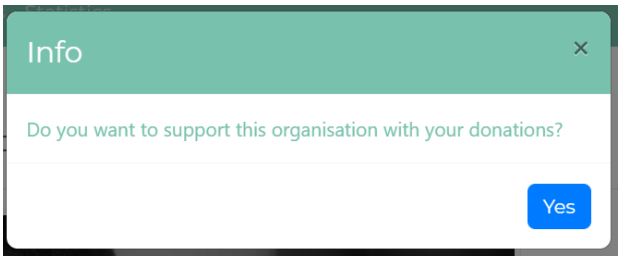

Abbildung 54 Dialogbox Auswahl Bestätigung Organisation

Nun erhält die eben ausgewählte Organisation Spenden von Ihnen, wenn Sie Ihren IoT-Button betätigen. In Ihrem Spende Profil wird nun ebenfalls diese Organisation angezeigt.

7.5.1.3 Zahlungsmittel hinterlegen

Ihr Zahlungsmittel können Sie in Ihren Profileinstellungen unter «Payment instruction» hinterlegen. Geben Sie die Informationen einer gültigen Kreditkarte ein. Diese wird jeweils belastet, wenn Sie ihren IoT-Button betätigen.

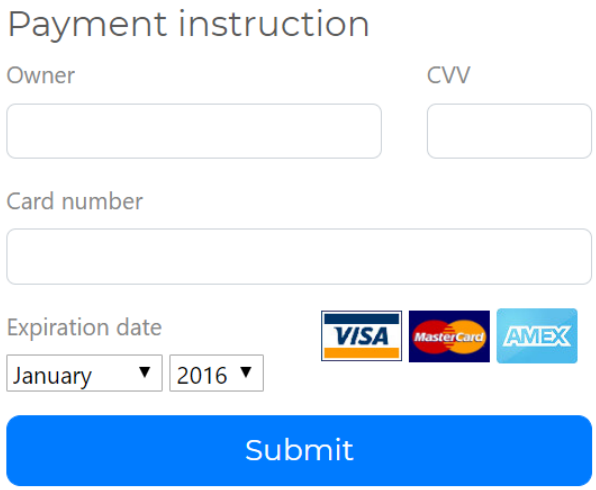

Abbildung 55 Formular für die Zahlungsinformationen. Kreditkarten Bilder von Quelle (8)

Hinweis: In der aktuellen Version werden die Kreditkarten Daten noch nicht verarbeitet. Es werden keine Belastungen durchgeführt.

7.5.2 IoT-Button bestellen

Falls Sie noch keinen persönlichen IoT-Button besitzen, können Sie diesen in der Navigationsleiste unter «Settings/Configure Button» bestellen indem Sie auf «Order a button» klicken. Geben Sie nun Ihre Anschrift ein und bestätigen Sie mit «Order».

Die Kosten inkl. Versand werden von Ihrem hinterlegten Zahlungsmittel belastet.

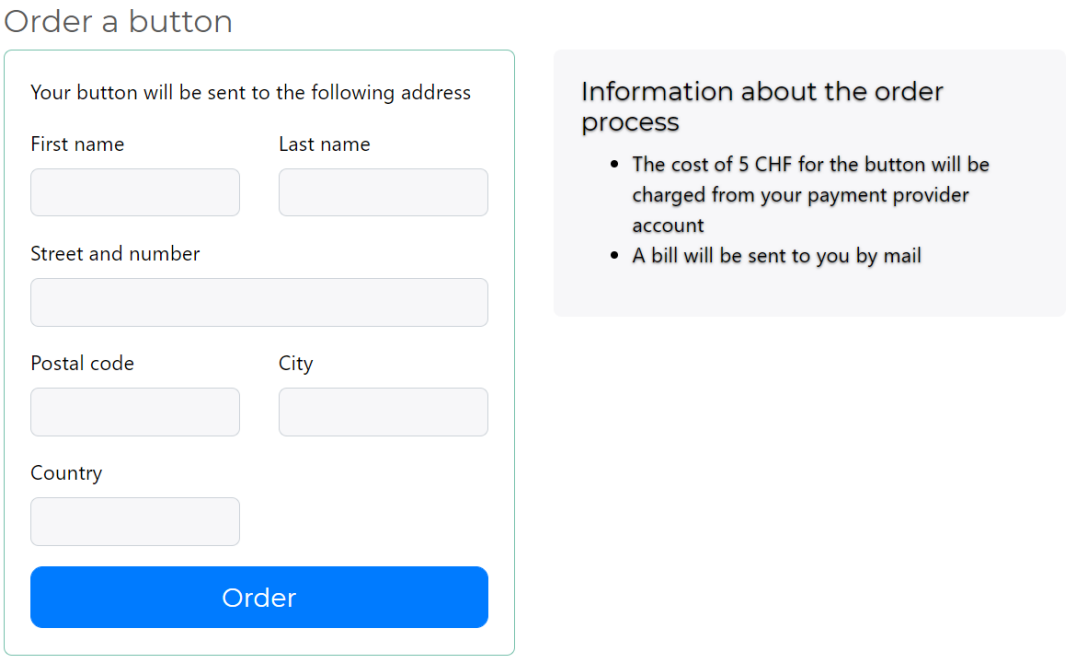

Abbildung 56 Bestellformular für einen IoT-Button

Hinweis: In der aktuellen Form wird noch kein Bestellprozess ausgelöst. Dieses Formular dient lediglich als Vorbereitung für die spätere Umsetzung.

7.5.3 IoT-Button In Betrieb nehmen

Setzen Sie zwei AAA-Batterien in den IoT-Button ein. Achten Sie dabei auf die Polarität.

Abbildung 57 IoT-Button mit eingesetzten AAA-Batterien

Sobald die Batterien eingelegt worden sind, sollte der Knopf blau leuchten.

Der IoT-Button befindet sich nun im Access Point Betrieb und kann mit jedem WLAN tauglichen Gerät unter dem Namen «EasyDonate» gefunden werden.

Jetzt kann mit der Konfiguration begonnen werden (Kapitel 7.5.4).

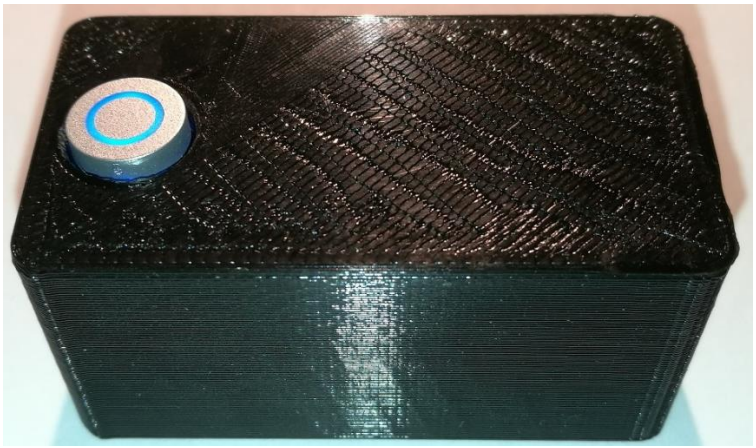

Abbildung 58 IoT-Button ist eingeschaltet und leuchtet blau

Hinweis: Falls Sie sich entscheiden den IoT-Button später zu konfigurieren, sollten Sie die Batterien wieder entfernen. Ansonsten verbleibt der IoT-Button so lange im Access Point Betrieb, bis das WLAN Netzwerk korrekt konfiguriert wurde.

7.5.4 IoT-Button konfigurieren

Die Daten für die Konfiguration ihres persönlichen IoT-Buttons finden Sie in der Navigationsleiste unter «Settings/Configure Button».

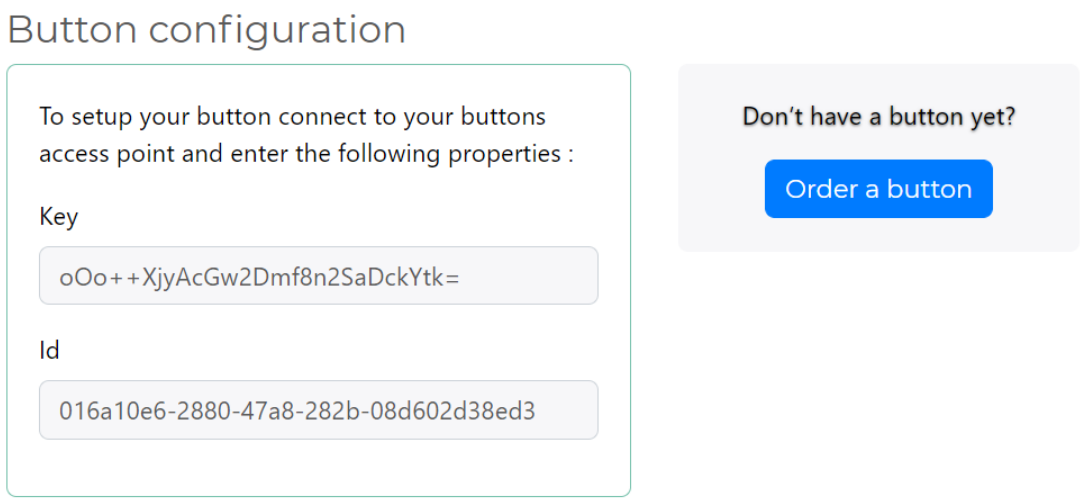

Abbildung 59 Persönliche IoT-Button Konfiguration

Stellen Sie sicher, dass Sie den IoT-Button korrekt in Betrieb genommen haben, bevor Sie fortfahren (Kapitel 7.5.3).

Wenn der Knopf des IoT-Button blau leuchtet suchen Sie mit einem WLAN tauglichen Gerät (Smartphone, Computer, etc.) nach einem Access Point mit dem Namen «EasyDonate» und wählen Sie diesen aus. Auf Ihrem Gerät sollte nun ein Pop-Up erscheinen, welches Sie auffordert sich im WLAN-Netzwerk «EasyDonate» anzumelden. Klicken Sie diesen Dialog an.

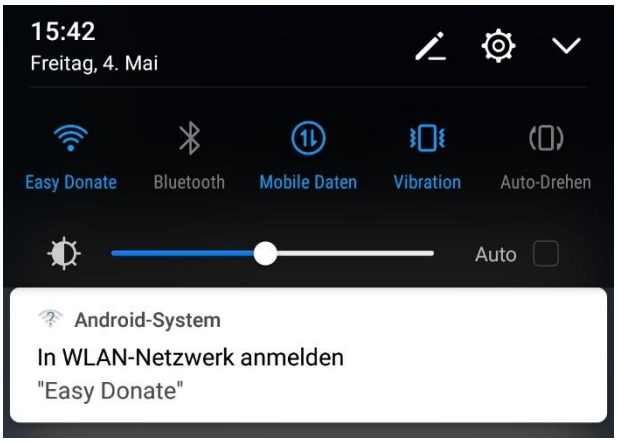

Abbildung 60 Pop-Up «In WLAN-Netzwerk anmelden»

Es wird Ihnen ein Einstellungsmenü angezeigt. Wählen sie den Punkt „Configure WiFi“ aus.

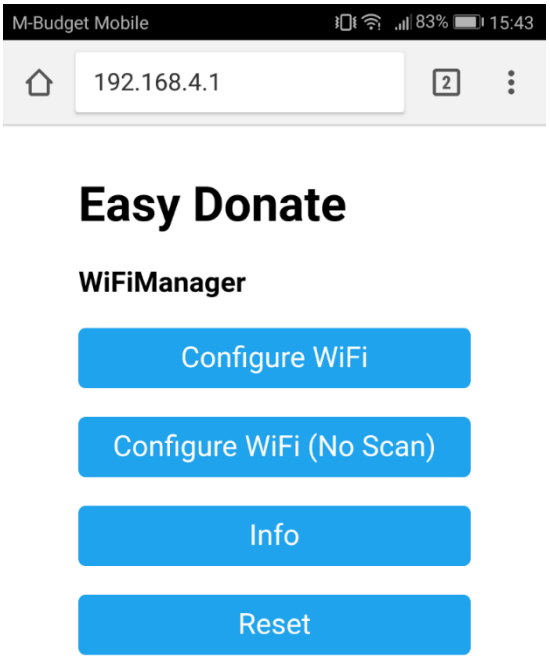

Abbildung 61 Konfigurations Menü

Der IoT-Button sucht nun nach verfügbaren WLAN Netzwerken. Alle gefundenen Netzwerke werden in einer Liste angezeigt. Das Schloss signalisiert, dass das Netz verschlüsselt ist. Die Zahl daneben zeigt die Signalstärke in Prozent an. Nachdem Sie Ihr lokales WLAN Netz ausgewählt haben, geben Sie im Feld «password» ihr WLAN Passwort ein.

Geben Sie in das Feld «buttonId» die Id und in das Feld «secret» den Key aus Ihrer Button Konfiguration ein (Dargestellt in Abbildung 59).

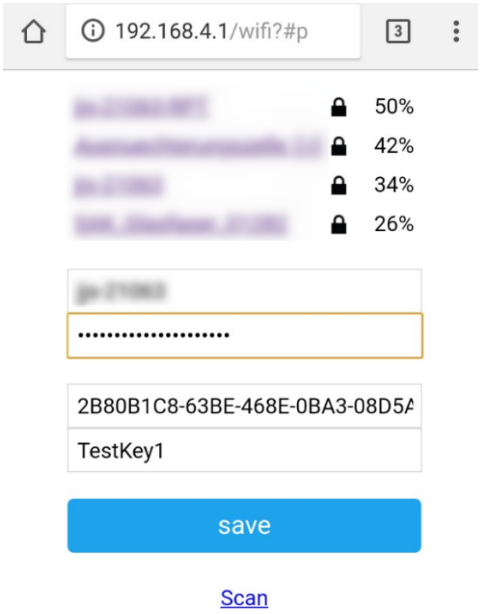

Abbildung 62 IoT-Button Einstellungen

Mit „save“ bestätigen Sie die Eingaben.

Wenn alle Daten korrekt eingegeben wurden und eine Verbindung zu dem eingegebenen WLAN Netwerk hergestellt werden kann, erlischt das blaue Licht des IoT-Button Knopfs.

Falls der Knopf weiterhin blau leuchtet muss die Konfiguration widerholt und auf die korrekte Eingabe aller Daten geachtet werden.

7.5.5 Spende durchführen

Stellen Sie sicher, dass Sie Ihr Spendenprofil eingestellt (Kapitel 7.5.1.1) und Ihren IoT-Button korrekt konfiguriert (Kapitel 7.5.4) haben.

Wenn Sie nun den Knopf auf Ihrem IoT-Button betätigen beginnt dieser zu leuchten. Sobald der Knopf 3-mal kurz aufblinkt und dann erlischt, wurde die Spende erfolgreich ausgeführt.

Falls der Knopf erlisch ohne 3-mal zu blinken, wurde keine Spende durchgeführt.

Dies könnte folgende Ursachen haben:

- Der von Ihnen konfigurierten tägliche maximal Spende Betrag ist erreicht
- Eingestellte minimale Zeit zwischen 2 Spende Vorgängen wurde nicht eingehalten
- Es wurden keine oder ungültige Zahlungsinformationen hinterlegt (aktuell noch nicht relevant)
- Der IoT-Button wurde nicht korrekt konfiguriert

7.6 Benutzung als Organisation

7.6.1 Profil Verwalten

Die Verwaltung des Profils Ihrer Organisation erreichen Sie über einen Klick auf «Settings» in der Navigationsleiste. Auf der linken Navigationsleiste können Sie die einzelnen Einstellungsgruppen verwalten. Die jeweils aktive wird dabei grün hinterlegt. «Change password» und «Contact information» wurde bereits in den Kapiteln 7.4.2 und 7.4.3 beschrieben.

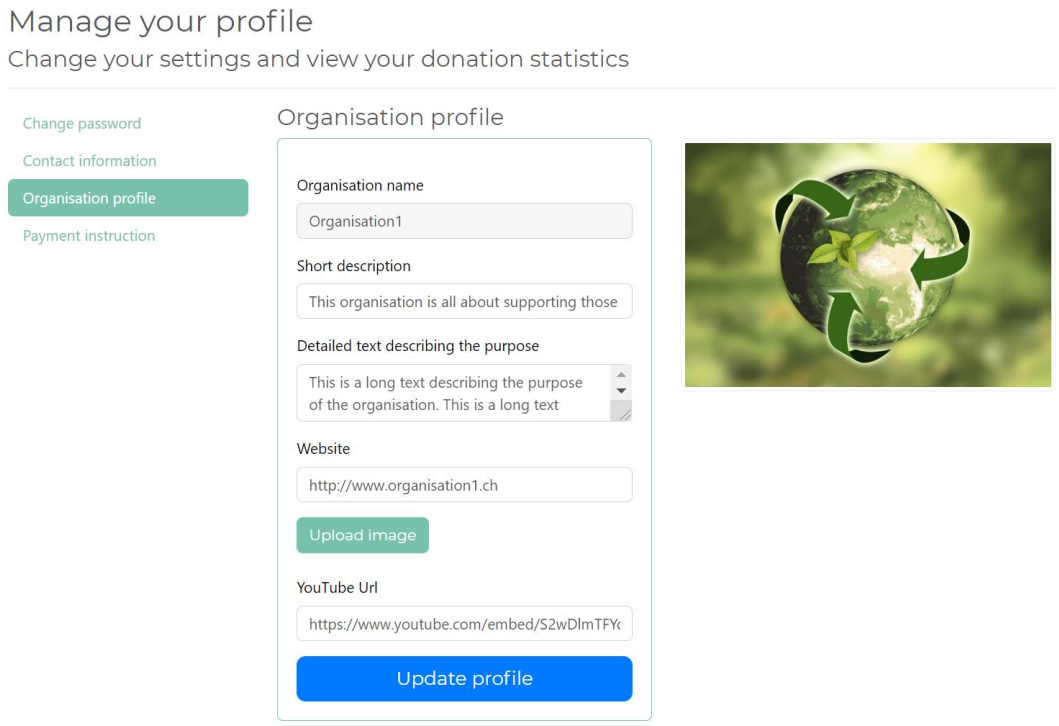

Abbildung 63 Profil Verwaltung Organisation

7.6.1.1 Öffentliches Profil verwalten

Den öffentlichen Auftritt Ihrer Organisation können Sie in den Profileinstellungen unter «Organisation profile» pflegen. Alle Informationen, die Sie dort eingeben sind Öffentlich einsehbar. Eine Beschreibung zu den verschiedenen Parametern finden Sie in Tabelle 18.

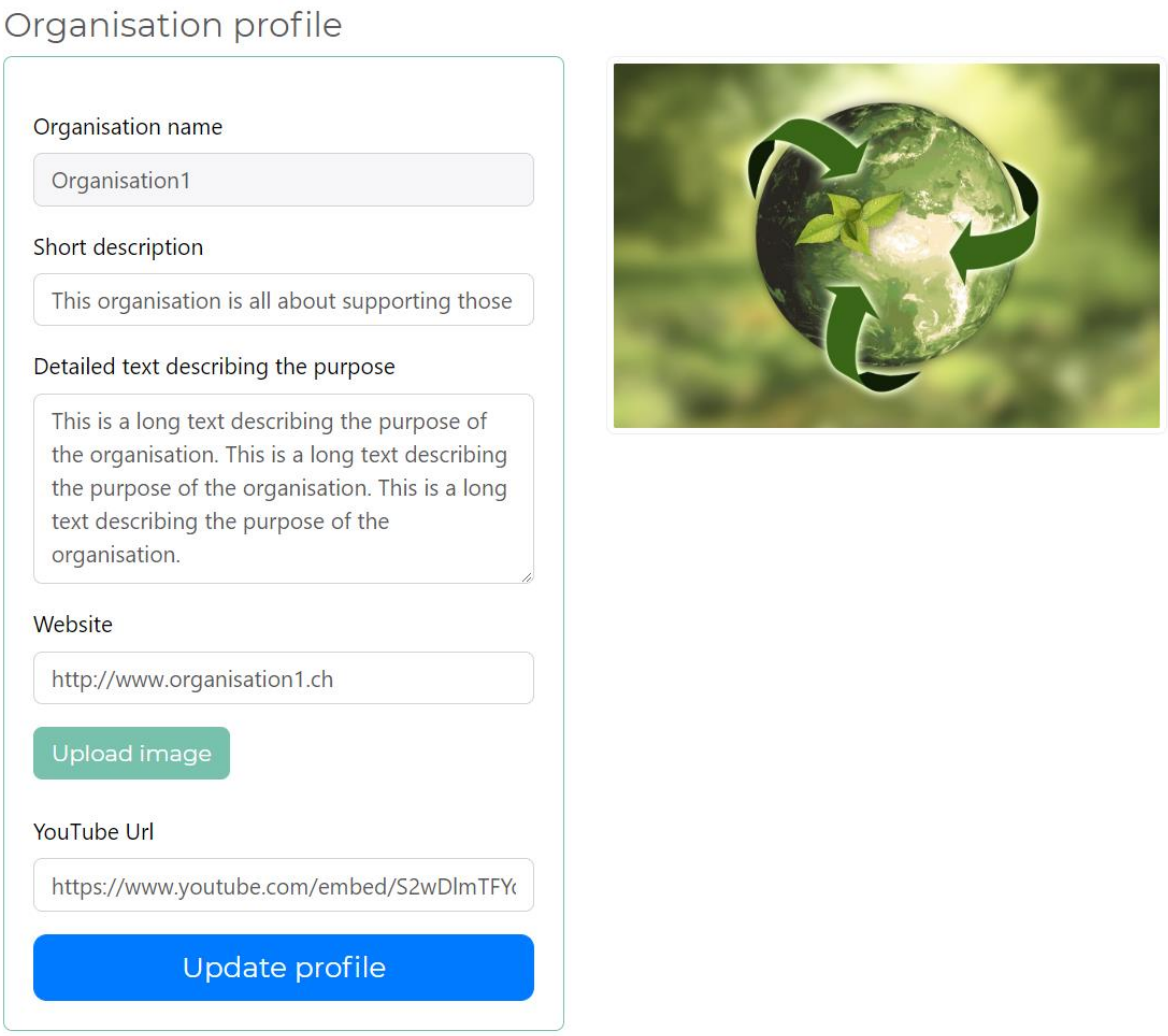

Abbildung 64 Formular für das öffentliche Profil

Einstellung	Beschreibung
Organisation name	Der Name ihrer Organisation. Dieser wurde bereits bei der Registrierung angegeben.
Short description	Beschreibt Ihre Organisation in einem Satz.
Detailed text	Beschreibt Ihre Organisation ausführlich.
Webseite	Link zu einer öffentlichen Webseite Ihrer Organisation.
Upload image	Über diese Schaltfläche können Sie ein Anzeigebild auswählen und hochladen.
YouTube Url	URL zu einem YouTube Video welches Ihre Organisation beschreibt. Hinweis: Es können nur embeddable YouTube Links verwendet werden. Diese sind am Zusatz «embed» im Pfad erkennbar. Bsp: https://www.youtube.com/**embed**/jLcznUls5CI

Tabelle 18 Einstellungen öffentliches Spendenprofil Organisation

7.6.1.2 Bankverbindung hinterlegen

Die Bankverbindung zu Ihrer Organisation können Sie in Ihren Profileinstellungen unter «Payment instruction» hinterlegen. Auf dieses Konto werden die empfangenen Spenden transferiert.

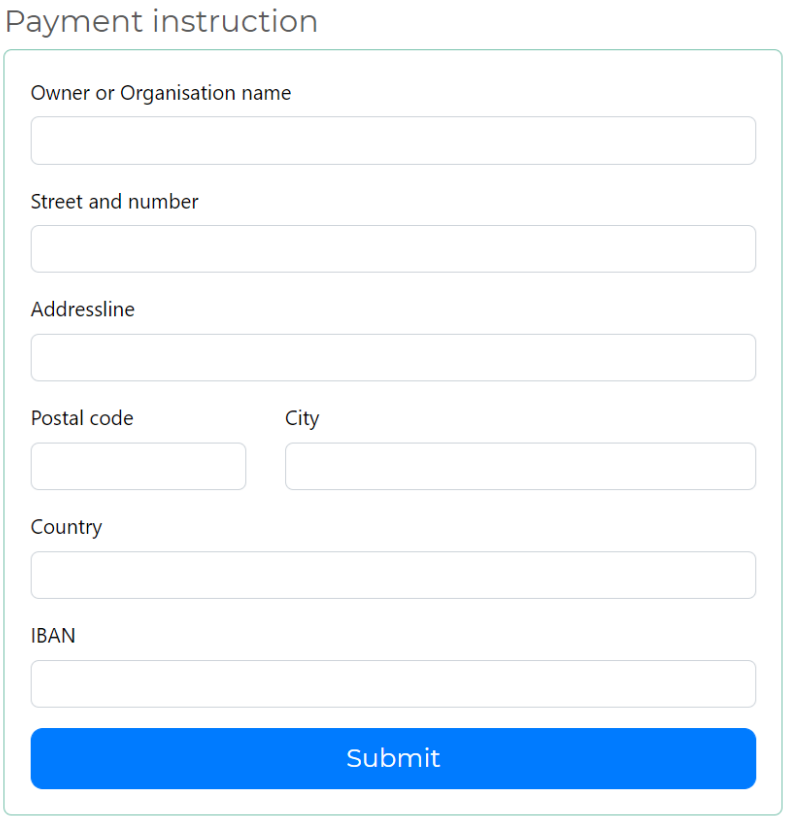

Abbildung 65 Formular für die Zahlungsinformationen

Hinweis: Die eingegebenen Daten werden aktuell noch nicht verarbeitet. Es können noch keine Spenden empfangen werden.

7.7 Benutzung als Administrator

7.7.1 Mitgliedschaftsanfragen verwalten

In der Navigationsleiste unter «Settings/Manage requests» sehen Sie eine Liste mit offenen Mitgliedschaftsanfragen von sich bewerbenden Organisationen. Mit «Accept» bestätigen Sie die Mitgliedschaftsanfrage. Die Organisation kann ab diesem Zeitpunkt die Dienste der Plattform in Anspruch nehmen. Mit «Deny» wird die Mitgliedschaft abgelehnt. Sobald eine dieser Aktionen ausgeführt wurde, verschwindet die Mitgliedschaftsanfrage und sie sehen eine Bestätigungs-Messagebox.

Bevor Sie die Mitgliedschaft annehmen, können Sie auch die Detailansicht der Organisation unter «Details» aufrufen.

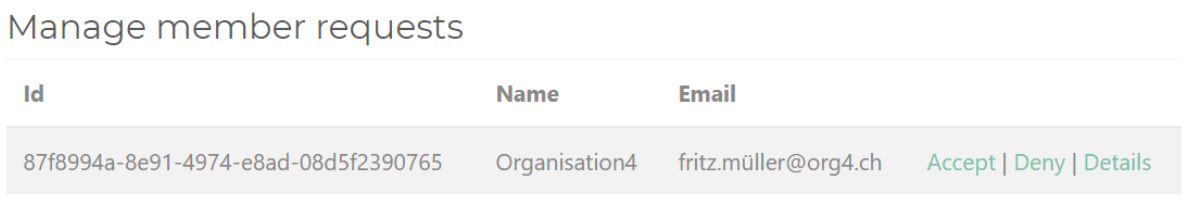

Abbildung 66 Liste mit offenen Mitgliedschaftsanfragen

7.7.2 Neuen Administrator erstellen

In der Navigationsleiste unter «Settings/Create administrator» können Sie jederzeit einen neuen Administrator erstellen. Dazu geben Sie den gewünschten Benutzernamen, die E-Mail-Adresse und ein Passwort ein. Mit der Schaltfläche «Create» bestätigen Sie die Eingaben und der Administrator wird erstellt. An die eingegebene E-Mail-Adresse wird eine Bestätigungs-E-Mail gesendet. Nachdem der Link in dem E-Mail angeklickt wurde, kann das neue Administrator Konto verwendet werden.

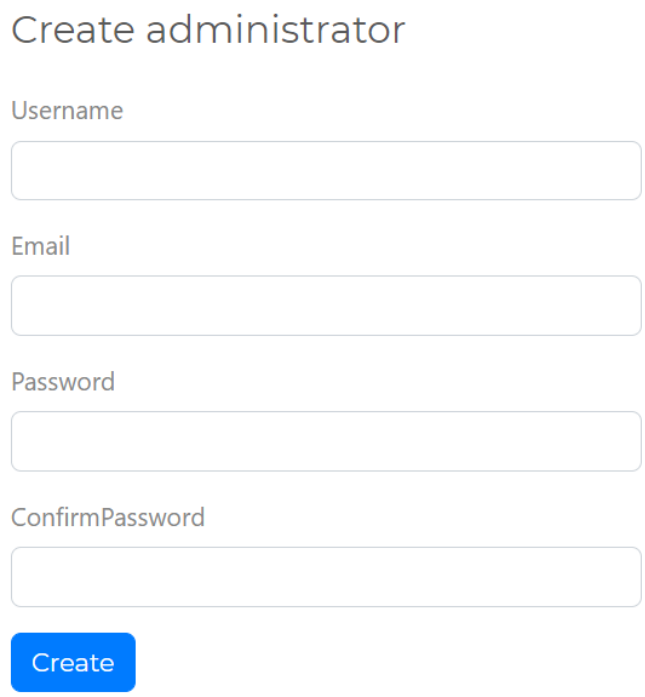

Abbildung 67 Formular für die Erstellung eines Administrators

8 Zusammenfassung und Ausblick

Die im Rahmen der Arbeit gestellten Anforderungen konnten erfüllt werden und mit der entwickelten Lösung wurde ein Proof of Concept fertiggestellt. Für eine kommerzielle Verwendung des Produkts müsste allerdings weitere Zeit in die Entwicklung investiert werden.

8.1 IoT-Button

Im Bereich des IoT-Buttons stehen diverse Verbesserungen sowohl auf Hardware- als auch auf Softwareebene an. Unter anderem sind dies:

- Implementierung einer OTA Update Möglichkeit.
- Verbesserung des Schaltkreises inkl. Erweiterung um einen Reset Button.
- Verbesserung des Deployment Vorgangs, des Fertigungsprozesses, des Gehäusedesigns, sowie Optimierung für die Massenproduktionsfähigkeit.

8.2 Webapplikation

Die Webapplikation müsste mindestens um die noch nicht umgesetzten funktionalen Anforderungen erweitert werden. Zudem sollte in die folgenden Bereiche investiert werden:

- Das Design der Oberfläche sollte professionalisiert werden.
- Implementierung eines Paging Mechanismus und einer Suchfunktionalität wo dies angebracht ist (Organisationsübersicht, Benutzerverwaltung des Administrators).
- Verbesserung der Validierung aller Requests im Frontend und im Backend.
- Überprüfung der Sicherheit und Erweiterung, wo notwendig.
- Fertigstellung der Benutzerverwaltung.
- Erweiterung um einen Prozess zur Bestellung eines Buttons über die Plattform.
- Integration eines kommerziellen Payment Providers.

8.3 Allgemein

Der Entwicklungs- und Release Prozess sollte wie folgt optimiert werden:

- Einführung von Git-Flow als Branching Strategie bevor die Plattform in Produktion geht
- Einführung einer automatisierten Versionierung in der Continuous Deployment Pipeline
- Bereitstellung mehrerer Deployment Umgebungen für Testing, Abnahme und Produktion

9 Verzeichnisse

9.1 Referenzen

1. Introduction to ASP.NET Core. [Online] 09. 08 2018. https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.0.

2. Architecting Modern Web Applications with ASP.NET Core and Microsoft Azure. [Online] 09. 08 2018. https://aka.ms/webappebook.

3. ESP8266 WiFiManager. [Online] 11. 08 2018. https://github.com/tzapu/WiFiManager.

4. Introduction to ASP.NET Identity. [Online] https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity.

5. SendGrid. [Online] 11. 08 2018. https://sendgrid.com/.

6. Charts.js. [Online] 16. 08 2018. www.charts.org.

7. Visual Studio Team Services. [Online] [Zitat vom: 11. 08 2018.] https://visualstudio.microsoft.com/de/team-services/.

8. Credit card logos. [Online] 21. 08 2018. http://www.credit-card-logos.com/.

9.2 Tabellenverzeichnis

9.3 Abbildungsverzeichnis