**Integrations-Projekt CAS HTML5**

**Integrations-Projekt CAS HTML5**

**Studierende:** Walter Raaflaub Remo Peduzzi

# Projekt-Dokumentation

**Vokko Voting App**

# Thema

Abstimmungen und Wahlen in Vereinen, KMUs oder sonstigen kleineren Teams durchführen.

# Umfeld, Ausgangslage

Die Projektidee entstand innerhalb eines anderen Studiengangs (MSc in Innovation and Entrepreneurship HEC Paris), den einer der beiden Studierenden (Remo Peduzzi) parallel besucht. Dort ist die Idee, eine universelle E-Voting App für diverse Einsatzbereiche zu kreieren. Das MVP fokussiert aber auf das Setup und die Durchführung einer Generalversammlung (General Meeting).

# Aufgabenstellung & Ziel des Projektes

Wir haben uns entschieden, uns auf folgende Szenarien auszurichten:

* Interaktiv durchgeführte Veranstaltungen, also «Live»-Abstimmungen und -Wahlen, nicht solche, die sich über mehrere Tage oder Wochen erstrecken
* Kleinere Teams wie zum Beispiel Vereine oder KMUs.

Viele Institutionen wie zum Beispiel Vereine konnten während der Pandemie ihre Mitgliederversammlungen nicht wie gewohnt ausführen und mussten auf improvisierte Alternativen ausweichen, zum Beispiel Abstimmungen auf dem Postweg – mit dem Nachteil, dass vor dem Entscheid keine Diskussion stattfand und keine Fragen gestellt werden konnten. Umgekehrt konnten die Mitgliederversammlungen auch nicht einfach per Zoom durchgeführt werden, weil ein einfaches, vertrauenswürdiges Tool zum Durchführen der Abstimmungen fehlte.

Die E-Voting-App füllt diese Lücke. Damit kann man Versammlungen remote oder hybrid durchführen, so dass jedes mit einem Handy oder Browser bewehrte Mitglied bei den Abstimmungen und Wahlen teilnehmen kann und die Ergebnisse sehen kann.

Der Organisator kann im Voraus Wahlen bzw. Abstimmungsfragen erfassen.

Mit Online-Kollaboration-Tools wie Mentimeter oder Miro kann man zwar auch Votings durchführen. Es fehlen aber Transparenz und Nachvollziehbarkeit.

# Scope des Projektes

Die E-Voting-App soll folgende Features beinhalten:

* Erfassung von Abstimmungsfragen bzw. Wahlen mit Kandidaten
* Durchführung der Versammlung
  + Organisator-Sicht:
    - Der Organisator bestimmt Start, Dauer und Ende der Wahl
    - End-Resultatanzeige
  + Wähler-Sicht:
    - Stimmabgabe resp. Kandidatenauswahl
    - End-Resultatanzeige
* Nach der Versammlung: Reports + Charts über die einzelnen Entscheide (Ja, Nein, Enthaltungen etc.)
* Erweiterungsmöglichkeiten
  + Wähler verwalten
  + Kandidatenprofile pflegen
  + Laufende Resultatanzeige
  + Eigene Stimmabgabe überprüfen, Visualsierung der Blockchain
  + Rolle „Verifier“: Überprüfung der Resultate

# Use Cases



|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC001 Register** | |
| **Actor** | Voter/ Organizer / Verifier | |
| **Pre-Condition** | None | |
| **Post-Condition** | User is registered | |
| **Basic Path** | **Actor** | **System** |
| * User opens app. * App asks the user if he wants to restore from Google Drive. * User chooses not to. * User clicks on Registration Link/ Button * User puts in some personal data including a unique identifier such as email. * User chooses if he wants to back up his data on Google Drive * User fills in the Google Credentials and a Master Passcode | System creates keypair for the Users and saves it to the localStorage  System registers the user on the public blockchain (public key and public identifiers only)  System encrypts all the localStorage Data and saves it to the Google Drive |

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC002 CAST VOTE** | |
| **Actor** | Voter | |
| **Pre-Condition** | * User is registered * User has been invited to the meeting * User has joined the meeting | |
| **Post-Condition** | User cannot vote for the specific voting anymore | |
| **Basic Path** | **Actor** | **System** |
| * User receives notification of a starting vote * User choses an option * User confirms the option * User is redirected to the home screen | System registered that the user voted on the blockchain and registers the option chosen without any link. |

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC003 ACCEPT INVITATION TO A VOTING** | |
| **Actor** | Voter | |
| **Pre-Condition** | * User is registered * User has been invited to the meeting | |
| **Post-Condition** | User is a registered voter for the voting | |
| **Basic Path** | **Actor** | **System** |
| * User clicks on the email link | Browser is opened by the email client  System registered that the user for the voting behind the scenes |

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC004 VERFIY ONE OF MY OWN VOTES** | |
| **Actor** | Voter | |
| **Pre-Condition** | * User is registered * User has been invited to the meeting * User has joined the meeting | |
| **Post-Condition** | None | |
| **Basic Path** | **Actor** | **System** |
| * User opens the app * The user navigates to the voting he wants to verify * User verifies the cote | The system verifies the block in the chain.  The system does some fancy animation to illustrate verification. It turns either green or red depending on the outcome. |

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC005 CREATE A VOTING** | |
| **Actor** | Organizer | |
| **Pre-Condition** | User is registered | |
| **Post-Condition** | None | |
| **Basic Path** | **Actor** | **System** |
| * User opens the app * User navigates to the organizer’s view * User creates a vote |  |

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC006 PUBLISH RESULTS** | |
| **Actor** | Organizer | |
| **Pre-Condition** | Meeting has ended | |
| **Post-Condition** | None | |
| **Basic Path** | **Actor** | **System** |
| * User opens the app * User navigates to the organizer’s view * User navigates to the voting * User publishes the results | System writes a block with the results. The block is visible to all invited voters at least or even open to the world. |

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC007 PREPARE A MEETING** | |
| **Actor** | Organizer | |
| **Pre-Condition** | User is registered | |
| **Post-Condition** | None | |
| **Basic Path** | **Actor** | **System** |
| * User opens the app * User navigates to the organizer’s view * User creates a new general meeting * User defines the date and title general meeting * User adds motions (questions with options or candidates) * User adds voters (participants) |  |

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC008 RUN A LIVE MEETING** | |
| **Actor** | Organizer | |
| **Pre-Condition** | User is registered | |
| **Post-Condition** | None | |
| **Basic Path** | **Actor** | **System** |
| * User opens the app * User navigates to the organizer’s view * User navigates to the general meeting * User starts the meeting |  |

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC009 INVITE VOTERS** | |
| **Actor** | Organizer | |
| **Pre-Condition** | User is registered | |
| **Post-Condition** | None | |
| **Basic Path** | **Actor** | **System** |
| * User opens the app * User navigates to the organizer’s view * User invites voters to the voting |  |

|  |  |  |
| --- | --- | --- |
| **Use Case** | **UC010 VERIFY VOTING RESULTS** | |
| **Actor** | Verifier | |
| **Pre-Condition** | User is registered, the voting ended, and the results were published | |
| **Post-Condition** | None | |
| **Basic Path** | **Actor** | **System** |
| * User opens the app * User navigates to the verifier’s view * User navigates to the voting to verify * The user verifies the voting |  |

# Umsetzung

|  |  |  |
| --- | --- | --- |
| **Use Case** | **Priority** | **Umgesetzt?** |
| **UC002 CAST VOTE** | HIGH | Ja |
| **UC005 CREATE A VOTING** | HIGH | Teilweise |
| **UC007 PREPARE A MEETING** | HIGH | Teilweise |
| **UC006 PUBLISH RESULTS** | HIGH | Ja |
| **UC008 RUN A LIVE MEETING** | HIGH | Ja |
| **UC009 INVITE VOTERS** | MEDIUM | Ja |
| **UC003 ACCEPT INVITATION TO A VOTING** | MEDIUM | Ja |
| **UC001 REGISTER** | LOW | Teilweise |
| **UC004 VERIFY ONE OF MY OWN VOTES** | LOW | Nein |
| **UC010 VERFIY RESULTS** | LOW | Nein |

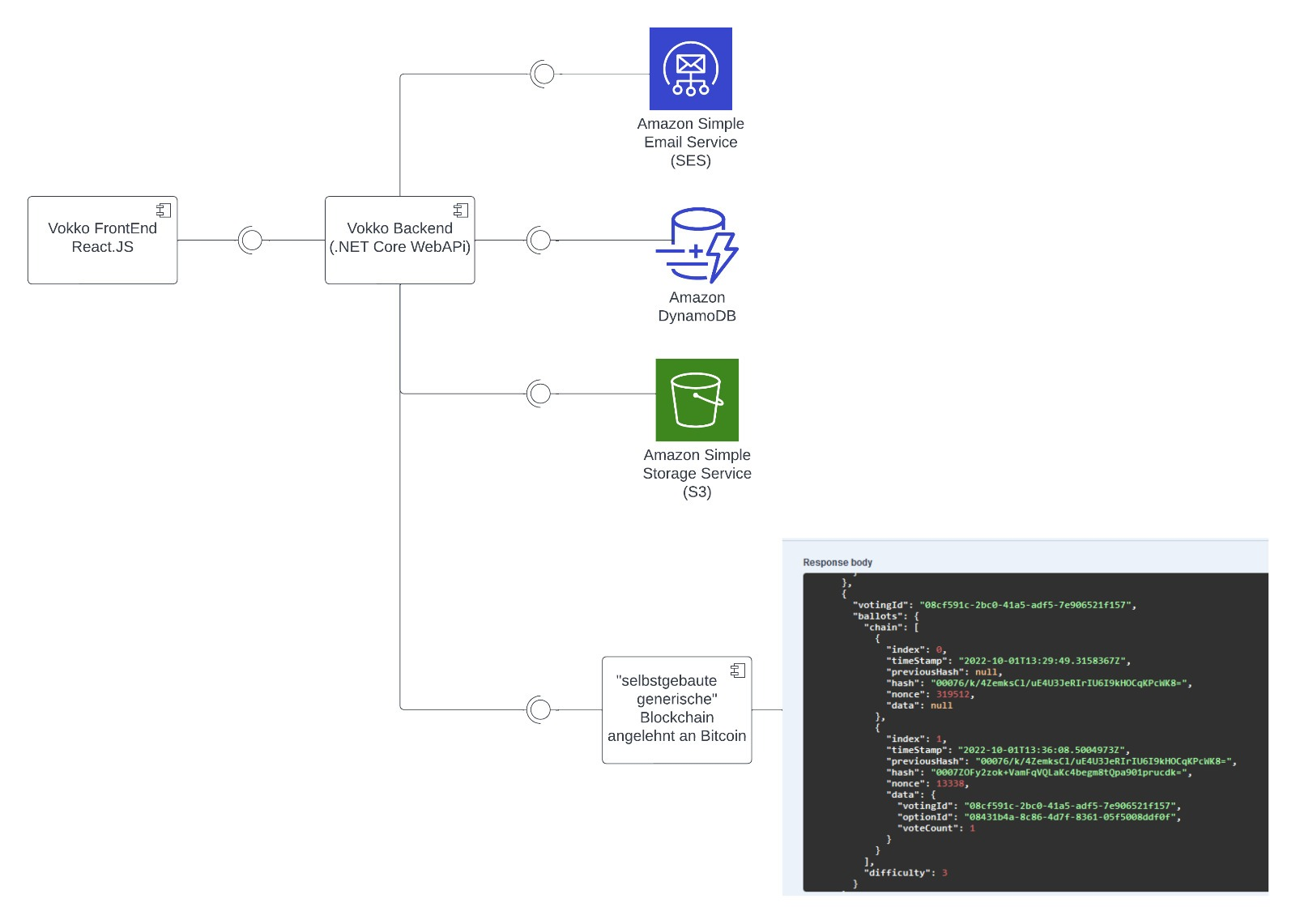
## **Funktionsumfang des Projektes, Auflistung der Features**

Als Funktionsumfang unseres Projekts haben wir uns an obenstehender Liste orientiert. Setup und haben wir bewusst minimalistisch gehalten. Auf detaillierte Konfigurationsdialoge haben wir verzichtet und sie teilweise durch Uploads ersetzt. Die Registrierung der Benutzer haben wir sehr einfach gehalten. Auf Authentifizierung und eine Benutzerverwaltung haben wir verzichtet: alle sehen alles.

Der Fokus lag auf der Durchführung und der Darstellung der Resultate. Auf die Möglichkeit, die Vorlagen ad hoc zu ergänzen oder abzuändern, mussten wir aus Zeitgründen verzichten.

# Grob-Architektur und Design

# Architecture and System Context



# API-Endpoints

Im Moment existieren folgende Endpoints, die auch voll funktionieren. Es braucht sicher noch Endpoints für Generalversammlungen. Weitere werden falls nötig erstellt und wenn möglich werden Dinge auch nur gemockt.

Graphical user interface, text, application, email

Description automatically generated

# Überblick über die zu realisierenden Screens

* Screens für die Organizer-View sind auf Desktop oder Tablet ausgerichtet
  + Meeting-Ablauf wird durch eine Timeline dominiert
  + Zu untersuchende Varianten: vertikale vs. Horizontale Timeline
  + Responsive Design, aber Optimierung für kleine Displays erst in zweiter Priorität
  + Die Organizer-Views sind bewusst eher einfach gehalten, z.T. im Formular-Stil
  + Der Versammlungs-Ablauf wird durch den Organizer gesteuert
    - Abstimmungen und Wahlen haben eine im Voraus bestimmte Zeitdauer
    - Können aber auch manuell beendet werden
  + Meeting-Setup und Meeting-Durchführung haben unterschiedliche, auf den Flow angepasste Darstellung derselben Daten
    - Zweite Prio: Änderung des Setups bei laufendem Meeting (zB Anträge)
* Screens für die Voter-View sind auf Mobile ausgerichtet
  + Meeting-Ablauf wird durch eine Timeline dominiert
  + Zu untersuchende Varianten: vertikale vs. Horizontale Timeline
  + Das UI soll den Voter möglichst nicht verwirren, daher Fokus auf allgemein verständliche UI-Idiome (Scrolling, Button, Checkbox)
  + Der Versammlungs-Ablauf wird durch den Organizer gesteuert
    - Der Voter folgt diesem Ablauf
    - Ausserhalb der eigentlichen Abstimmungen/Wahlen kann er durch die Vorlagen browsen

|  |  |  |  |
| --- | --- | --- | --- |
| Actor | Flow | Aktionen |  |
| Organizer | **Meeting-Übersicht und Resultate** | * Durch aktuelle und vergangene Meetings browsen * Reports |  |
| **Meeting-Setup** | * Neues Meeting * Titel und Datum * Vorlagen * Teilnehmer * Einladungen senden |  |
| **Meeting durch-führen** | * Meeting starten * Meeting beenden * Variante: horizontale Timeline |  |
| **Abstim-mung** | * Abstimmung starten * Abstimmung beenden * Resultate |  |
| **Wahlen** | * Wahl starten * Wahl beenden   Resultate |  |
| Voter | **Einladung** | * Per E-Mail-Link teilnehmen | Ein Bild, das Text enthält.  Automatisch generierte Beschreibung |
| **Registrie-rung** |  |  |
| **Übersicht** | * Durch die Abstimmungs-/ Wahlvorlagen browsen * Kandidaten ansehen * Bisherige Resultate ansehen * Variante: horizontale Timeline |  |
| **Abstim-men** | * Sachentscheide * Optionen:   + Ja   + Nein * Enthaltung |  |
| **Wahlen** | * *n* aus *m* Kandidaten wählen |  |

# Continuous Development and Integration

Wir werden bestehende Cloud Formation Templates verwenden um die CI/CD-Pipelines für das Frontend und Back-End in einem neuen Amazon AWS Account zu erstellen.

Diagram

Description automatically generated

# Eingesetzte Technologien und Thirdparty-Libraries

Build-Chain und tools:

* create-react-app, webpack, eslint
* jest (für Tests des Crypto-API)

Im Frontend:

* React
* Material UI
* i18next (Internationalisierung deutsch/englisch)
* Crypto API
* Local Storage
* chart.js, react-chartjs-2 (Diagramme)

Kommunikation mit dem Backend:

* SignalR Client (Messaging)
* REST (axios, react query)

Backend:

* .NET Core Web API (restful Webservice)
* SignalR Hubs

# Design und Umsetzung

Aufbau und Strukturierung der Umsetzung

* Zuerst top-down Festlegung der Grobstruktur: Routing, global benötigte Kontexte
* Anschliessend bottom-up: die Oberflächen nach Priorität der Features.
* Der Sourcecode ist nach Features strukturiert und nicht nach Artefakt-Typ.
* Durch das Backend generierte TypeScript-Model-Klassen

Design-Entscheide:

|  |  |
| --- | --- |
| Entscheid | Begründung |
| Einsatz von create-react-app mit webpack, Entscheid gegen vite | Mehr Mainstream, mehr Dokumentation und Anleitungen im Web |
| Wir machen eine App, nicht separate Apps für Organizer und Wähler. | Vereinfacht die Technologiewahl, zeitliche Limiten |
| Für Styling und Eingabeverhalten setzen wir auf Material UI | Pragmatische Lösung, die mit vertretbarem Aufwand ansehnliche Ergebnisse bringt. |
| Die Oberflächen sind responsive. Jene für die Wähler sind primär auf kleine Displays ausgerichtet, jene für den Organizer auf Desktops. | Material UI hat gute Unterstützung für Responsive Design, es war mit vertretbarem Aufwand umzusetzen |
| Statemanagement mithilfe von Contexts und Hooks. | Kein Einsatz eines überdimensionierten Frameworks (wie redux).  Contexts einsetzen, um Property-Drilling zu begrenzen. |
| Zugriffe auf das REST-Backend mit react-query, Schreibzugriffe mit useMutation | Ein einfaches Pattern, eine pragmatische Lösung für das Reload-Problem |
|  |  |

# Herausforderungen

* Technologiewahl: zuerst wollten wir vite einsetzen, da es zu kleineren Bundles führt, waren aber mit der Konfiguration überfordert und wechselten zu create-react-app mit webpack.
* Verheiraten von verschieden Programmiermodellen, Einbindung des Crypto-APIs in die React-App, allgemein API-Zugriffe (File-API, LocalStorage) in Hooks verpacken. Lösung: Test-driven development mit Jest, generell: Best Practices anwenden.

# Lessons Learned

* Strukturiertes Vorgehen ist King. In kleinen Schritten vorwärtsgehen.
* Man muss mit etwas beginnen und darf nicht im Status der Analyse steckenbleiben.
* Zeit ist eine wertvolle Ressource.
* TypeScript ist nicht nur ein Vorteil, es kann einen auch ausbremsen.

# Projektablauf und Erfahrungen

Was gut funktioniert hat:

* Toolchain machte weniger Probleme als erwartet (keine tagelangen Konfigurationsorgien)
* Material UI ist ein sehr mächtiges Framework und nimmt einem enorm viele Detailprobleme ab

Was verbessert werden könnte:

* Kurs-Timing ist suboptimal. Man hat zu spät das nötige Rüstzeug, um mit dem Projekt durchstarten zu können.

# Ausblick

Remo wird das Projekt in seiner Gruppe an der HEC (MSc in Innovation and Entrepreneurship HEC Paris) präsentieren und deren Feedback einholen.