VOKKO

YOU DECIDE

The e-voting platform

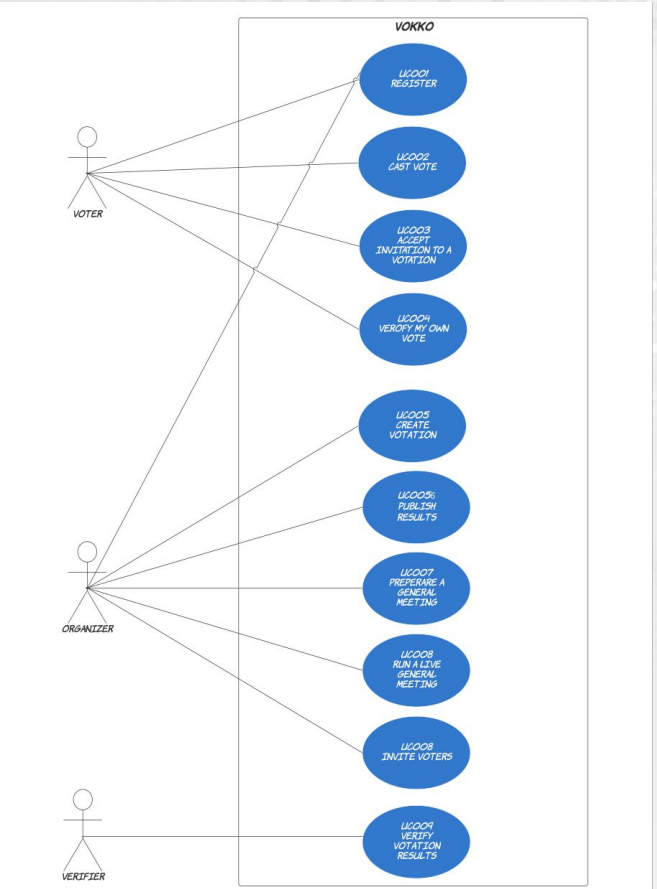
Thema und Motivation

- Abstimmungen und Wahlen in Vereinen, KMUs oder sonstigen kleineren Teams durchführen
- Die Projektidee entstand innerhalb eines anderen Studiengangs (MSc in Innovation and Entrepreneurship HEC Paris), den einer der beiden Studierenden (Remo Peduzzi) parallel besucht.

Ziel des Projektes

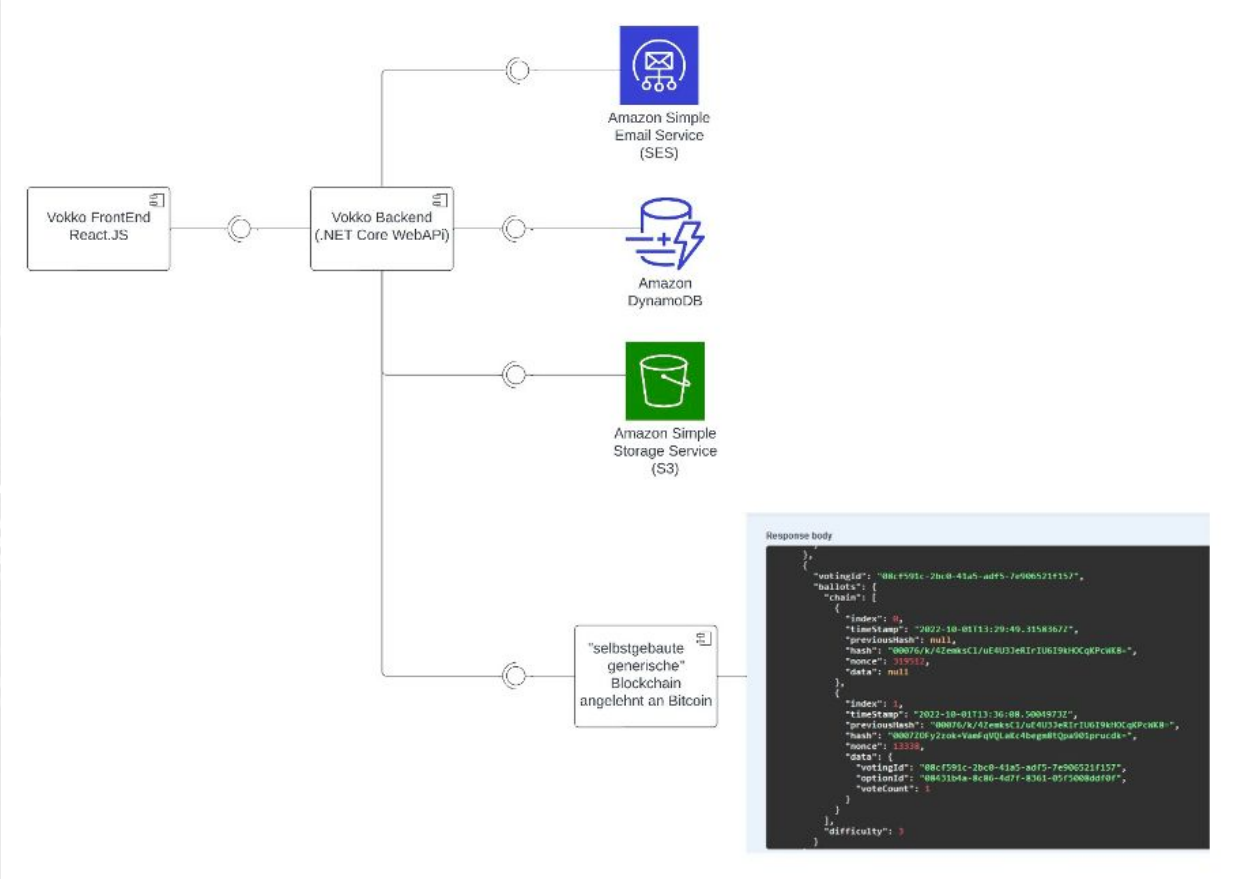
- Interaktiv durchgeführte Veranstaltungen, also «Live»-Abstimmungen und -Wahlen, nicht solche, die sich über mehrere Tage oder Wochen erstrecken
- Kleinere Teams wie zum Beispiel Vereine oder KMU

Use Cases & Umsetzung

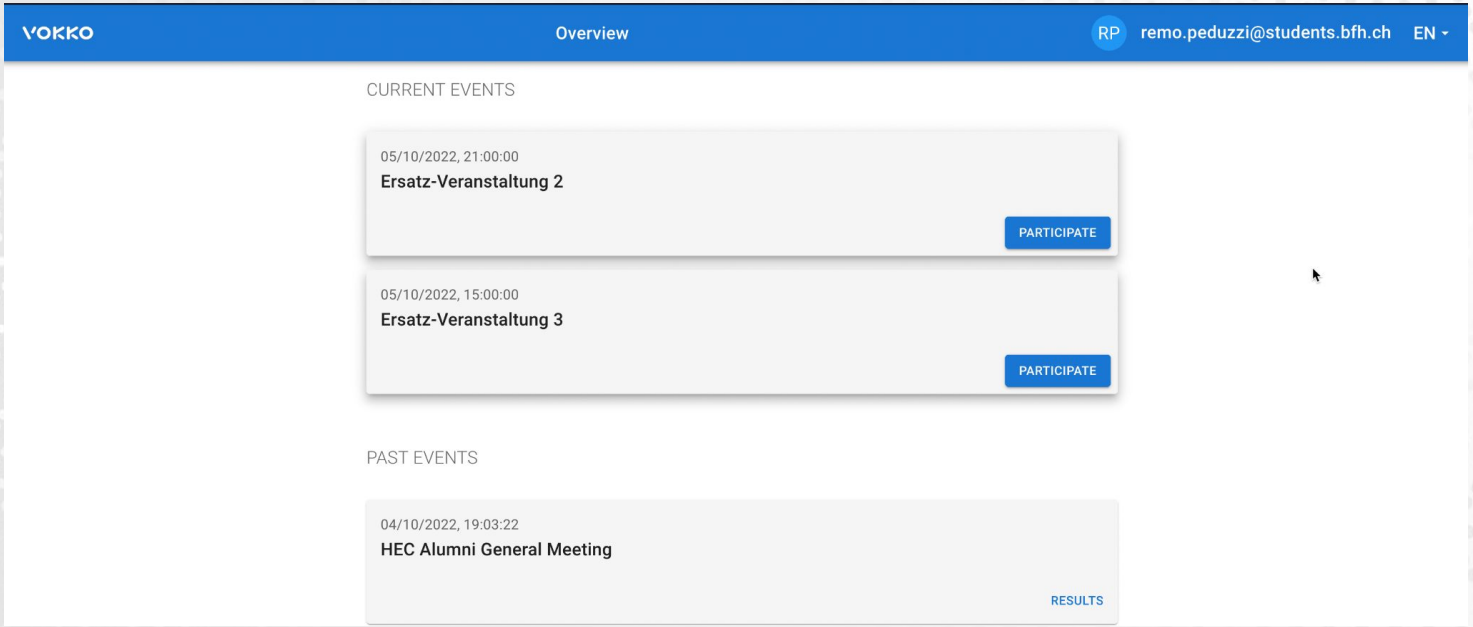


Use Case	Priorität	Umgesetzt?
UC002 CAST VOTE	HIGH	Ja
UC005 CREATE A VOTING	HIGH	Teilweise
UC007 PREPARE A MEETING	HIGH	Teilweise
UC006 PUBLISH RESULTS	HIGH	Ja
UC008 RUN A LIVE MEETING	HIGH	Ja
UC009 INVITE VOTERS	MEDIUM	Ja
UC003 ACCEPT INVITATION TO A VOTING	MEDIUM	Ja
UC001 REGISTER	LOW	Teilweise
UC004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

Architecture



DEMO


VOKKO
Overview
RP
remo.peduzzi@students.bfh.ch
EN
CURRENT EVENTS
05/10/2022, 21:00:00
Ersatz-Veranstaltung 2
PARTICIPATE
05/10/2022, 15:00:00
Ersatz-Veranstaltung 3
PARTICIPATE
PAST EVENTS
04/10/2022, 19:03:22
HEC Alumni General Meeting
RESULTS

Eingesetzte Technologien und Libraries

- Im Frontend:
  - Material UI
  - Google Webfonts
  - i18next (Internationalisierung deutsch/englisch)
  - Crypto API
  - LocalStorage API
  - File API
  - chart.js, react-chartjs-2 (Diagramme)
- Build-Chain und Tools
  - create-react-app, webpack, eslint
  - jest (für Tests des Crypto-API)
- Kommunikation mit dem Backend:
  - SignalR Client (Messaging)
  - REST (axios, react query)
- Backend:
  - .NET Core Web API (restful Webservice)
  - SignalR Hubs

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

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 vorwärts gehen.
- 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.

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 Detail-Probleme ab
- Was verbessert werden könnte:
  - Kurs-Timing ist suboptimal. Man hat zu spät das nötige Rüstzeug, um mit dem Projekt durchstarten zu können.