BÄRNHÄCKT

D2-PLAN Nachhaltigkeit in Gemeinden ermöglichen

Technische Informationen für die Jury

Technische Informationen für die Jury

Aktueller Stand des Sourcecodes

- https://github.com/natrontech/360-noscope

Ausgangslage

- *Worauf habt ihr euch fokussiert?*
  - Usability/ User Experience
    - Darstellung der wichtigen Informationen in einem übersichtlichen Webinterface
  - Extensibility
    - Einfache und standardisierte Erweiterbarkeit von weiteren «Data Crawler»
  - Scalabilty
    - Schnelles, automatisiertes Deployment mit Skalierungsmöglichkeit
- *Welche technischen Grundsatzentscheide habt ihr gefällt?*
  - Mehrere Komponente welche über standardisierte und dokumentierte Schnittstellen miteinander kommunizieren

Technischer Aufbau

- *Welche Komponenten und Frameworks habt ihr verwendet?*
  - Backend
    - Analyzer: Java/ SpringBoot
    - UI: Pocketbase
  - Frontend:
    - UI: Svelte
  - Data Crawler: Python
  - Data Persistence: Elasticsearch
  - Deployment: Kubernetes mit FluxCD
- *Wozu und wie werden diese eingesetzt?*
  - Die «Data Crawler» holen Daten von anderen APIs im Internet (z.B. GeoAdmin API) und speichern diese in den Elasticsearch
  - Der Analyzer liest die Daten aus dem Elasticsearch, transformiert diese gemäss den Indikatoren-Definitionen und stellt diese in einem einheitlichen Format als REST-API dem Frontend zur Verfügung
  - Das Frontend liest die Daten von der REST-API des Analyzer und stellt diese den Endbenutzer übersichtlich dar. Zusätzlich können Umfragen für die Gemeinde erstellt werden und über einen Link von den Bewohnern ausgefüllt werden
  - Der Analyzer, das UI und der Elasticsearch laufen als Container auf einem Kubernetes Cluster und sind aus dem Internet erreichbar. Das Deployment ist per FluxCD vollautomatisiert und kann schnell und beliebig skaliert werden (horizontal & vertikal)

Implementation

- *Gibt es etwas Spezielles, was ihr zur Implementation erwähnen wollt?*
  - Bei der Implementation wurde der Fokus auf das Aufzeigen der Machbarkeit gelegt. Gleichzeitig wurde die Architektur so gewählt, dass man diese beliebig skalieren kann.
  - Wir können Geometry-Objekte mit den Data Crawler ins Elasticsearch speichern, diese im Analyser mithilfe von GIS-Tools übereinanderlegen (intersections, area contains point) und andere Operationen durchführen.
- *Was ist aus technischer Sicht besonders cool an eurer Lösung?*
  - Es können nun weitere «Data Crawler» laufend hinzugefügt werden, der Standard ist definiert, die Machbarkeit wurde mit ersten Indikatoren gezeigt
  - Cloud-Ready, skaliert extrem gut, 12 Factor, vollautomatisiertes Deployment

Abgrenzung / Offene Punkte

- *Welche Abgrenzungen habt ihr bewusst vorgenommen und damit nicht implementiert? Weshalb?*
    - Wir haben den Fokus auf Indikatoren gelegt, welche in guter Qualität über eine API-Schnittstelle verfügbar sind und somit automatisiert verarbeitet werden können. Aus Zeitgründen und der Qualität der zur Verfügung stehenden Daten konnten wir nur einige wenige Indikatoren vollständig umsetzen. Somit müssen nun weitere Indikatoren entwickelt und hinzugefügt werden.