Anforderungen aus dem Projektantrag

- Interne und externes Monitoring - wir nennen es Zone
- Verfügbarkeiten messen durch Erreichbarkeit (200), Zertifikat vorhanden, Performance in Millisekunden.
- Nutzererfahrungen sammeln (manuelles Feedback durch Klicken, Meldung von Störungen über das Tool, Erfassung von Störungen.
- Bewerten der Applikation im gesamten (zu und abnahme).
- frühzeitige Identifikation systemkritischer Ereignisse identifizieren
- Ungeplante Ausfälle messen und reduzieren
- kontinuierliche Analyse von Verfügbarkeit und Performance messen
- Lösung als Grundlage für datenbasierte Optimierungsmassnahmen
- Alarming- und Eskalationsmechanismen
- Benutzerfeedback messen
- Geschäftskritische Applikationen erfassen
- Grün = Applikation antwortet korrekt
- Orange = Applikation hat einen Fehler erzeugt
- Rot = Applikation hat >3 Fehler hintereinander erzeugt
- Maintenance = Blau
- Bei Rot werden Eskalationsmails an ICT-Contact und Service Owner ausgelöst. Ggf. gibt es ein automatisches Update in einem Alerting Tool

Entitäten und Attribute

Zone

Repräsentiert interne und externe Monitoring-Zonen.

- **id** (UUID, Primärschlüssel)
- **name** (Text, z. B. "Intern", "Extern")

Applikation

Die zu überwachenden Anwendungen.

- **id** (UUID, Primärschlüssel)

- **name** (Text)
- **url** (Text, für Erreichbarkeitsprüfung)
- **zone_id** (UUID, Fremdschlüssel zu Zone)
- **ict_contact_id** (UUID, Fremdschlüssel zu Subscriber, für Eskalationsmails)
- **service_owner_id** (UUID, Fremdschlüssel zu Subscriber, für Eskalationsmails)
- **is_critical** (Boolean, markiert geschäftskritische Applikationen)

Probe (bereits vorhanden)

Führt Messungen für Applikationen durch.

- **id** (UUID, Primärschlüssel)
- **name** (Text)
- **applikation_id** (UUID, Fremdschlüssel zu Applikation)
- **interval** (Integer, Messintervall in Sekunden)

Subscriber (bereits vorhanden)

Benutzer oder Systeme, die Benachrichtigungen erhalten.

- **id** (UUID, Primärschlüssel)
- **name** (Text)
- **email** (Text, für Benachrichtigungen)

ProbeSubscriber (bereits vorhanden)

Verknüpfung zwischen Probe und Subscriber.

- **probe_id** (UUID, Fremdschlüssel zu Probe)
- **subscriber_id** (UUID, Fremdschlüssel zu Subscriber)
- Primärschlüssel: Kombination aus **probe_id** und **subscriber_id**

Alert

Speichert Alarmierungen und deren Status.

- **id** (UUID, Primärschlüssel)
- **applikation_id** (UUID, Fremdschlüssel zu Applikation)
- **status** (Text, z. B. "Grün", "Orange", "Rot", "Blau")
- **message** (Text, Details zum Alert)

- **created_at** (Timestamp, Zeitpunkt des Alerts)

Feedback

Erfasst Nutzererfahrung und Störungsmeldungen.

- **id** (UUID, Primärschlüssel)
- **applikation_id** (UUID, Fremdschlüssel zu Applikation)
- **subscriber_id** (UUID, Fremdschlüssel zu Subscriber, optional)
- **comment** (Text, Benutzerfeedback oder Störungsmeldung)
- **created_at** (Timestamp, Zeitpunkt des Feedbacks)

Maintenance

Wartungszeiten für Applikationen (Status "Blau").

- **id** (UUID, Primärschlüssel)
- **applikation_id** (UUID, Fremdschlüssel zu Applikation)
- **start_time** (Timestamp, Beginn der Wartung)
- **end_time** (Timestamp, Ende der Wartung)
- **description** (Text, optional)

Measurement

Speichert Messergebnisse zur Verfügbarkeit und Performance.

- **id** (UUID, Primärschlüssel)
- **probe_id** (UUID, Fremdschlüssel zu Probe)
- **timestamp** (Timestamp, Zeitpunkt der Messung)
- **status_code** (Integer, z. B. HTTP 200 für Erreichbarkeit)
- **response_time** (Float, Antwortzeit in Millisekunden)
- **certificate_valid** (Boolean, Zertifikat vorhanden und gültig)

ErrorLog

Protokolliert ungeplante Ausfälle und Fehler.

- **id** (UUID, Primärschlüssel)
- **applikation_id** (UUID, Fremdschlüssel zu Applikation)
- **timestamp** (Timestamp, Zeitpunkt des Fehlers)

- **error_message** (Text, Details zum Fehler)

Beziehungen zwischen Entitäten

- **Zone → Applikation (1:n)**: Eine Zone kann mehrere Applikationen haben.
- **Applikation → Probe (1:n)**: Eine Applikation wird von mehreren Probes überwacht.
- **Applikation → Alert (1:n)**: Eine Applikation kann mehrere Alerts auslösen.
- **Applikation → Feedback (1:n)**: Eine Applikation kann mehrere Feedback-Einträge haben.
- **Applikation → Maintenance (1:n)**: Eine Applikation kann mehrere Wartungsfenster haben.
- **Applikation → ICT Contact und Service Owner (jeweils 1:1)**: Beide sind Subscriber.
- **Probe → ProbeSubscriber (1:n)**: Eine Probe kann mehrere Subscriber haben.
- **Subscriber → ProbeSubscriber (1:n)**: Ein Subscriber kann mehreren Probes zugeordnet sein.
- **Probe → Measurement (1:n)**: Eine Probe erzeugt mehrere Messungen.

Erfüllung der Anforderungen

- **Interne und externe Zonen**: Durch die Zone-Tabelle abgebildet.
- **Verfügbarkeit und Performance messen**: Measurement speichert Statuscode (z. B. 200), Antwortzeit und Zertifikatsstatus.
- **Nutzererfahrung sammeln**: Feedback ermöglicht Kommentare und Störungsmeldungen.
- **Bewertung der Applikation**: Durch Analyse von Measurement und Feedback möglich.
- **Frühzeitige Identifikation kritischer Ereignisse**: Alert mit Status "Rot" bei >3 Fehlern, unterstützt durch ErrorLog.
- **Ungeplante Ausfälle messen**: ErrorLog protokolliert Fehlerdetails.
- **Kontinuierliche Analyse**: Measurement liefert Daten für Verfügbarkeit und Performance.
- **Datenbasierte Optimierung**: Alle Tabellen bieten Grundlagen für Analysen.

Alarmierungs- und Eskalationsmechanismen

- **Status in Alert**: Grün (korrekt), Orange (1 Fehler), Rot (>3 Fehler), Blau (Wartung).
- Bei "Rot" werden Eskalationsmails an **ict_contact_id** und **service_owner_id** gesendet (via Trigger oder Anwendungslogik).
- **Benutzerfeedback messen**: Via Feedback-Tabelle.
- **Geschäftskritische Applikationen**: **is_critical** in Applikation.

Prometheus-Konfiguration

Template beschreibung kommt.

Ausgabe als YAML

```
- job_name: '<probe_name>'
  static_configs:
    - targets: ['<applikation_url>']
```

Optional kann eine separate Tabelle **PrometheusConfig** erstellt werden:

- **id** (UUID, Primärschlüssel)
- **probe_id** (UUID, Fremdschlüssel zu Probe)
- **config** (Text, YAML/JSON für Prometheus)

Zusammenfassung

Das Datenmodell besteht aus den Tabellen **Zone**, **Applikation**, **Probe**, **Subscriber**, **ProbeSubscriber**, **Alert**, **Feedback**, **Maintenance**, **Measurement** und **ErrorLog**. Es erfüllt alle Anforderungen des Monitoring-Systems, inklusive Verfügbarkeitsmessung, Alarmierung, Nutzerfeedback und der Generierung von Prometheus-Konfigurationen.

Datenbankmodell für Monitoring-System

Dieses Dokument beschreibt das Datenbankmodell für das Monitoring-System in PostgreSQL, inklusive Tabellen, Primär- und Fremdschlüssel sowie Beispiel-Datensätzen.

Tabellenübersicht

Tabelle	Beschreibung	Primärschlüssel	Fremdschlüssel
`zone`	Interne und externe Monitoring-Zonen	`id`	-
`subscriber`	Benutzer für Benachrichtigungen	`id`	-
`applikation`	Zu überwachende Anwendungen	`id`	`zone_id`, `ict_contact_id`, `service_owner_id`
`probe`	Messungen für Applikationen	`id`	`applikation_id`
`probe_subscriber`	Verknüpfung Probe-Subscriber	`probe_id`, `subscriber_id`	`probe_id`, `subscriber_id`

`alert`	Alarmierungen und Status	`id`	`applikation_id`
`feedback`	Nutzerfeedback und Störungsmeldungen	`id`	`applikation_id`, `subscriber_id`
`maintenance`	Wartungsfenster	`id`	`applikation_id`
`measurement`	Messergebnisse	`id`	`probe_id`
`error_log`	Fehlerprotokoll	`id`	`applikation_id`

Entity-Relationship-Diagramm (ERD)

Das folgende Diagramm zeigt die Beziehungen zwischen den Entitäten. Es wird in PlantUML-Syntax geschrieben und kann mit entsprechenden Tools (z. B. AsciiDoc-Plugins) gerendert werden.