Dokumentation zum Fortgeschrittenenpraktikum

ChronicleDB

Arbeitsgruppe Datenbanksysteme

Philipps-Universität Marburg

Davit Davtyan Lars Happel Johannes Buder

WS 2021/2022

Inhaltsverzeichnis

1 Planung

1.1 Kurzbeschreibung

ChronicleDB ist ein Prototyp eines Zeitreihen-Datenbanksystems, entwickelt in der AG Datenbanksysteme, welcher einen Datenstrom von Ereignissen (Ereignisstrom) verarbeitet und persistiert. Ein Ereignis besteht aus einem Zeitstempel (timestamp) und einem Anhang (payload). Ein Beispiel für einen Ereignisstrom bietet z.B. ein Temperatursensor, welcher in bestimmten zeitlichen Abständen seine Messung (Ereignis) an das System schickt. Dabei enthält timestamp den Zeitpunkt der Messung und payload die gemessene Temperatur.

Das innerhalb eines Semesters zu erstellende Admin Panel soll dabei stets erweiterbar sein, damit es auch in Zukunft noch mit gemeinsam mit ChronicleDB einsetzbar ist.

1.2 Entwicklungsprozess

1.2.1 Zu erstellende Dokumente

Während des Entwicklungszeitraum sollen für alle Team-Treffen, mindestens einmal wöchentlich Protokolle angefertigt werden, in denen die besprochenen Themen und ermittelten Ziele festgehalten werden.

1.2.2 Rollenvergabe

Die Protokollantenrolle in Treffen, sowie die Aufgabenverteilung bei der Softwareentwicklung werden flexibel festgelegt. Da das Projekt in erster Linie zur Übung dienen soll, ist es sinnvoll, dass jedes Teammitglied möglichst jede Aufgabe einmal übernimmt.

1.2.3 Integrierte Entwicklungsumgebung

Visual Studio Code zusammen mit Node JS und dem Angular Framework.

1.3 Team

Unser Team besteht aus drei Mitgliedern. Lars Happel, dem das Repository gehört, Davit Davtyan und Johannes Buder. Die Aufgabenbereiche haben wir auf alle gleichermaßen aufgeteilt, um jedem die Möglichkeit zu bieten, aus allen Bereichen zu lernen.

1.4 Risikomanagement

Ein nicht zu vernachlässigendes Risiko ist, dass wir mit Webentwicklung insbesondere aber mit der Programmiersprache TypeScript/JavaScript und dem Framework Angular keine bis sehr wenig Erfahrung haben. Ebenso ist ChronicleDB ein für uns zuvor unbekanntes Projekt, in das wir uns zunächst einarbeiten mussten. Da die Dokumentation des eben genannten Projekts Lücken aufweist, hatten wir zudem Schwierigkeiten, uns dort einzuarbeiten.

Da wir bereits in der Vergangenheit viel mit Java programmiert haben, haben wir uns für TypeScript entschieden, welches Java mehr ähnelt als JavaScript. Dadurch wollen wir das Risiko, dass wir nicht mit der Programmiersprache umgehen können, minimieren. Zudem haben wir uns einen Online-Angular-Kurs gekauft, der uns grundlegende aber auch tiefer gehende Themen von Angular und TypeScript näher bringt, sodass wir dieses Risiko primär behandeln können. Mit diesem Kurs können wir mit passenden Beispielen unser bisheriges Wissen erweitern. Da er ziemlich ausführlich ist, schätzen wir das Risiko, dass wir mit

der Sprache und dem Framework nicht umgehen können, auf sehr gering ein. Falls doch noch Probleme auftreten, gibt es auf StackOverflow etliche Artikel über eben genannte Themen, die uns weiterhelfen.

1.5 Zeitplan

Durch die Leitlinie des Praktikums ist eine grobe Meilensteinplanung vorgegeben. Wir planen, uns in diesem Projekt an diesem Vorschlag zu orientieren. Da wir jedoch bisher keine Erfahrung mit der Sprache, der Library und dem Programm haben, planen wir primär für den ersten (aber auch noch für den zweiten) Meilenstein mehr Zeit ein, um uns das nötige Wissen aneignen und ausprobieren können.

Das Projekt wird dabei durchgehend für den Betreuer verfügbar sein.

1.5.1 Zusammenfassung der Meilensteine

1. Meilenstein
   - REST API / JSON verstehen
   - GUI mit Button für create stream, wie pgAdmin Interface oder eigenes Interface
2. Meilenstein
   - Buttons für insert event, usw. alle anderen (siehe REST Spezifikation auf GitHub/dbs)
3. Meilenstein
   - User-Verwaltung: Admin und User Rechte - Java via Postgres oder SpringBoot oder per Hand | Rust via Postgres oder per Hand
   - Authentifizierung und ggf. Anpassung der GUI, sodass nur das angezeigt wird, was auch möglich ist
4. Meilenstein
   - Jobs einführen, sodass eine Anfrage wiederholt automatisch ausgeführt wird
   - Status visualisieren
   - Tests und Dokumentationen
5. Abgabe
   - Software und Bericht
   - Vorstellung

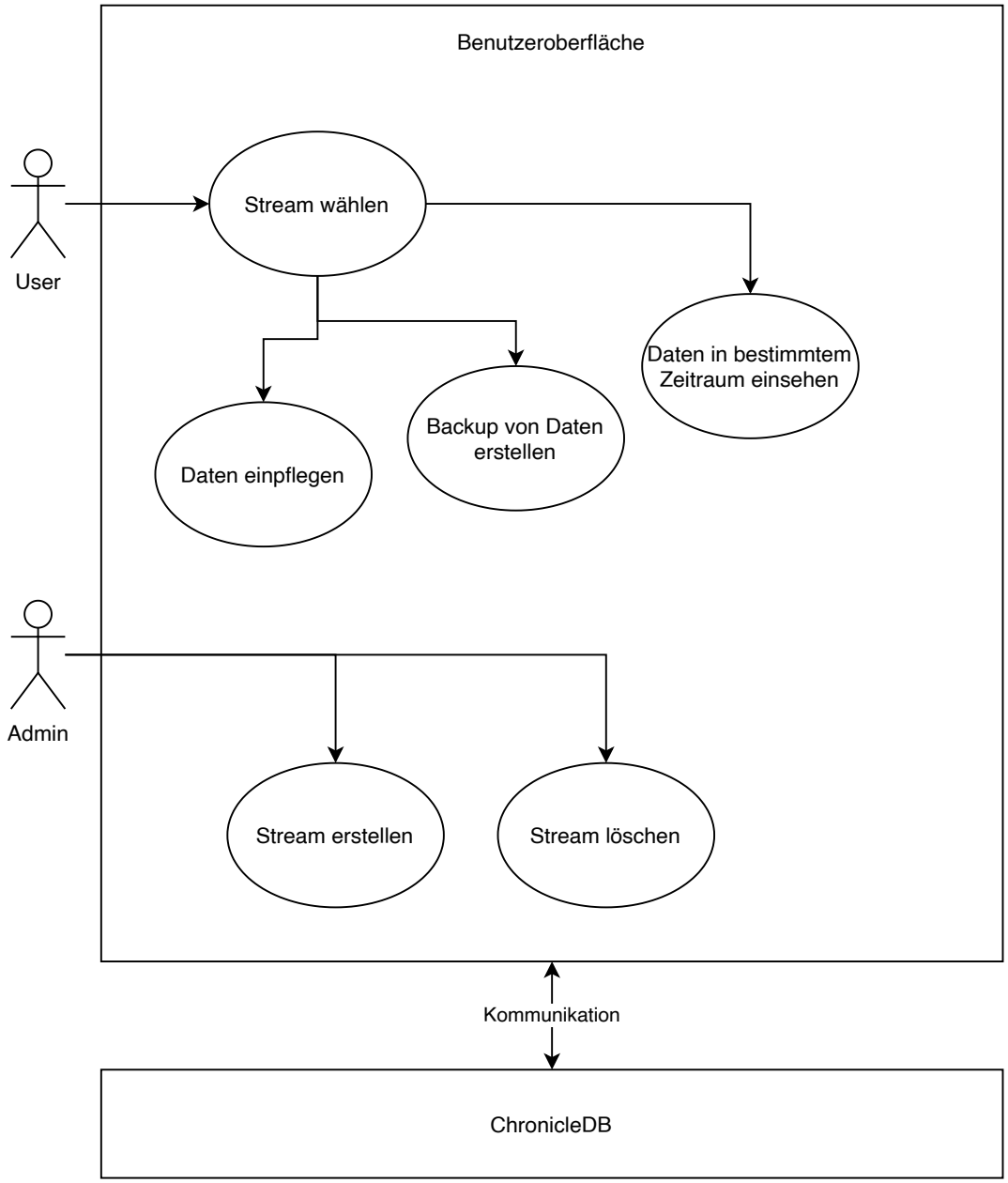
2 Anforderungsanalyse

2.1 Definition des Zielsystems

Das Zielsystem muss primär die Funktionalitäten von Angular und somit die von HTML und JavaScript unterstützen. Falls ChronicleDB lokal ausgeführt werden soll, müssen ebenso die Anforderungen an das Projekt erfüllt sein.

2.2 Funktionale Anforderungen

2.2.1 Übersicht



2.2.2 User

Stream auswählen (notwendig)

Vorbedingung Der User ist in die Weboberfläche eingeloggt. Es ist noch kein Stream zur Bearbeitung ausgewählt.

Ablauf Der User wählt aus einer Liste der angebundenen und ihm verfügbaren Streams einen zur Bearbeitung aus.

Daten einpflegen (notwendig)

Vorbedingung Der User ist eingeloggt und ein zur Bearbeitung freigeschalteter Stream ist ausgewählt.

Ablauf Der User wählt die Option zum Einpflegen von Daten. Ein Dialog öffnet sich zur Auswahl einer Quelldatei. Der User gibt den Pfad zur Quelldatei an und die darin enthaltenen Daten werden der Datenbank hinzugefügt.

Backup von Daten erstellen (optional)

Vorbedingung Der User ist eingeloggt und ein zur Bearbeitung freigeschalteter Stream ist ausgewählt.

Ablauf Der User wählt die Backup Option. Eine Backup-Datei mit den Daten des aktuellen Streams wird als Download bereit gestellt.

Daten in einem bestimmten Zeitraum einsehen (notwendig)

Vorbedingung Der User ist eingeloggt und ein zur Bearbeitung freigeschalteter Stream ist ausgewählt.

Ablauf Der User wählt die "Time Travel" Funktion. Ein Dialog fragt den Start- und Endzeitpunkt der einzusehenden Daten ab. Nach korrekter Eingabe werden die Daten aus diesem Zeitraum angezeigt.

2.2.3 Admin

Stream erstellen (notwendig)

Vorbedingung Der Admin ist in die Weboberfläche eingeloggt. Er befindet sich auf der Seite zum erstellen von Streams.

Ablauf Der Admin definiert die URL des Servers und Spezifiziert, wie ein Event des Streams aussehen soll. Dazu wählt er zunächst aus, ob das Event nur aus einem Datenfeld (Single) bestehen soll, oder ob es aus mehreren (Compound oder VarCompound) bestehen soll. Entsprechend der Auswahl erstellt der Admin die gewünschte Anzahl an Komponenten und spezifiziert für jede einzelne, ob es eine Liste oder ein einzelner Wert sein soll, den Datentyp (Integer, String, etc.) und einen Subtyp (8bit Integer, 16bit Integer, 32bit Integer, ...). Falls eine Liste oder ein String ausgewählt wurde, soll zudem die Standardgröße der Liste bzw. die Standardgröße des Strings angegeben werden.

Optional kann der Admin noch genauere Konfigurationen des Streams vornehmen, indem er die Standardeinstellungen zum Debugging, I/O, zu der Blockgröße, zum Cache und zum Kompressionsverfahren ändert.

Sobald er alles eingestellt hat, kann er den Create-Stream-Button verwenden, um diesen zu erstellen.

2.3 Nicht-Funktionale Anforderungen

2.3.1 Qualität des Systems/Codes

Es soll eine möglichst hohe Code-Abdeckung durch Systemtests erreicht werden. Es werden die gängigen Code-Konventionen für TypeScript eingehalten.

2.3.2 Benutzbarkeit

Das System soll für einen User, der damit noch nicht vertraut ist einfach verwendbar sein. Dies wird erreicht durch die Verwendung von Tooltips und gängigen Layouts wie z.B. Navbar. Der User wird vor

der Eingabe ungültiger Daten abgehalten und ggfs. durch Nachrichten auf Eingabefehler hingewiesen.

Die Zielgruppe sind User, welche auf die in ChronicleDB zur Verfügung gestellten Daten zugreifen und diese ggfs. bearbeiten wollen.

2.3.3 Technische Anforderungen

Die Weboberfläche muss kompatibel sein zum ChronicleDB System der AG Datenbanken. Sie muss zudem auf gängigen Browsern (Chrome, Firefox, Edge) lauffähig sein.

2.3.4 Anforderungen an die Dokumentation

Die Anforderungen werden durch die Leitlinie des Fortgeschrittenenpraktikums vorgegeben.