My Bachelorthesis title

Bachelorarbeit von Philipp Hinz
Tag der Einreichung: 21. Januar 2023

1. Gutachten: Gutachter 1
2. Gutachten: Gutachter 2
3. Gutachten: noch einer
4. Gutachten: falls das immernoch nicht reicht
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Institut
Arbeitsgruppe

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Philipp Hinz, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 21. Januar 2023

P. Hinz

Inhaltsverzeichnis

1 Introduction

1.1 Case Study

2 Background and Related Work

2.1 Local-First

2.2 CvRDT and CmRDT

2.3 Technical Stack

3 Extendable Commutative Replicable Data Types

In this chapter we will introduce the concept of Extendable Commutative Replicable Data Types (ECmRDT) and present our implementation of this concept.

3.1 Motivation

CvRDTs and CmRDTs do not care about authentication and authorization. This is usally not a problem in trusted environments like a cluster of servers. But used in client running applications (local-first applications) we usally can not trust our clients. Therefore we need to add authentication and authorization to our data types.

Additionally local-first applications require end-to-end encryption if we want to store our data on a server. This is usally difficult to combine with authentication and authorization.

Our here proposed solution is specially designed for a single server, but the concepts can be easily applied to peer-to-peer applications. We explore the use in peer-to-peer applications a bit in the future work section.

Initially we tried to design a CRDT specifically for authentication and authorization. But we found out that a more abstract approach is more flexible and easier to implement. Therefore we designed a new data type called Extendable Commutative Replicable Data Type (ECmRDT) with authentication and authorization as an extension.

3.2 Overview

We base our ECmRDT on CmRDTs and therefore make use of event sourcing. Because we design our ECmRDT to be used in server/client applications we only use direct event sourcing in storing the events on the server. On the client we use a more classic approach and only store the last state of the data type and all pending events to be sent to the server. Incoming events are applied to the last state to get the new state.

The core feature of our ECmRDT is the concept of extensions. Extensions are a way to add functionality by mutation of state through events or validation of events.

3.2.1 Concepts

```
        case class ECmRDT[A, C <: IdentityContext](
                val state: A,
                val clock: VectorClock = VectorClock(Map.empty)
        )
```

3.2.2 Extensions and EffectPipeline

3.3 ECmRDT Example

3.4 Conclusion

4 Ratable

4.1 Architecture

4.1.1 Core

4.1.2 Functions

4.1.3 Webapp

4.2 Implementation

4.3 Evaluation

5 Future Work

5.1 Ratable

5.2 ECmRDT