



**Bachelor's Thesis**

# **Providing a distributed file storage for German Schul-Cloud**

**Erstellung einer verteilten Dateiverwaltung für die deutsche  
Schul-Cloud**

by

**Niklas Kiefer**

Potsdam, July 2017

**Supervisor**

Prof. Dr. Christoph Meinel,  
Jan Renz

**Internet-Technologies and Systems Group**

## Disclaimer

I certify that the material contained in this dissertation is my own work and does not contain significant portions of unreferenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project and all associated documentation.

Hiermit versichere ich, dass diese Arbeit selbständig verfasst wurde und dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt wurden. Diese Aussage trifft auch für alle Implementierungen und Dokumentationen im Rahmen dieses Projektes zu.

Potsdam, May 26, 2017

---

(Niklas Kiefer)

## **Contents**

<b>1. Introduction</b>	<b>1</b>
<b>2. Related Work</b>	<b>2</b>
<b>3. Concept</b>	<b>3</b>
<b>4. Implementation</b>	<b>4</b>
<b>5. Evaluation</b>	<b>5</b>
<b>6. Future Work</b>	<b>6</b>
<b>7. Conclusion</b>	<b>7</b>
<b>Bibliography</b>	<b>8</b>
<b>A. Appendix</b>	<b>8</b>

## 1. Introduction

Die deutsche Schul-Cloud soll "dabei helfen, die digitale Transformation in Schulen zu meistern und den fächerübergreifenden Unterricht mit digitalen Inhalten zu bereichern" [1]. Ein Schwerpunkt ist hierbei das Arbeiten mit verschiedenen Dateiformaten im Unterrichtskontext. Dazu soll die Schul-Cloud eine Möglichkeit schaffen, eigene Dateien zu verwalten und sie unter Lehrern und Schülern zu teilen. Dafür soll keine neue Dateiverwaltungstechnologie entworfen werden, sondern auf bestehende Systeme zurückgeführt werden. Viel mehr soll für den Nutzer der Schul-Cloud möglich sein, ein bestehendes Dateisystem weiter zu nutzen und in die Schul-Cloud zu integrieren. Außerdem soll es möglich sein, Dateien auf verschiedenen Systemen zu lagern, um eine flexible Architektur zu schaffen. Dies ist vor allem dann nötig, wenn große Datenmengen von sehr vielen Schulen, Schülern und Lehrern entstehen. Dann ist es wichtig, eine Lösung in Form von verteilten FileStorage zu finden.

In dieser Bachelor-Arbeit wird eine solche Architektur für ein verteiltes Dateiverwaltungssystem beschrieben und entworfen. Begonnen wird mit **Related Work** (2), in welcher sich über die bestehende Situation von Dateiverwaltung im Schulunterricht auseinander wird. Danach wird ein **Konzept** (3) für ein verteiltes System beschrieben und anschließend eine mögliche **Implementierung** (4) aufgezeigt. In dieser wird eine Anbindung von FileStorage-Providern im Schul-Cloud Server und Client beschrieben. Diese Anbindungsmöglichkeit wird anschließend **evaluiert** (5). Dann werden mögliche **Anschlusszenarien** (6) aufgezeigt, inwieweit dieses Konzept ausgebaut werden kann. Zum Ende wird die Arbeit **zusammengefasst** (7).

## 2. Related Work

### **3. Concept**

## 4. Implementation

## 5. Evaluation



## 6. Future Work

## 7. Conclusion

## A. Appendix

```
1 public class SensorActivity extends Activity implements SensorEventListener {
2     private SensorManager sensorManager;
3     private Sensor accelerometer;
4
5     @Override
6     public final void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.main);
9         sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
10        accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
11    }
12
13    @Override
14    protected void onResume() {
15        super.onResume();
16        sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
17    }
18
19    @Override
20    protected void onPause() {
21        super.onPause();
22        sensorManager.unregisterListener(this);
23    }
24
25    @Override
26    public final void onSensorChanged(SensorEvent event) {
27        StringBuilder log = new StringBuilder("Acceleration:");
28        log.append(" X: ").append(String.valueOf(event.values[0]));
29        log.append(" Y: ").append(String.valueOf(event.values[1]));
30        log.append(" Z: ").append(String.valueOf(event.values[2]));
31        System.out.println(log.toString());
32    }
33
34    @Override
35    public final void onAccuracyChanged(Sensor sensor, int accuracy) {
36        // sensor accuracy changed
37    }
38 }
```

Listing 1: Activity with lifecycle callbacks