IT Systems Engineering
Universität Potsdam

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 16, 2017

(Niklas Kiefer)

Contents

1. Introduction

Die deutsche Schul-Cloud soll "dabei helfen, die digitale Transformation in Schulen zu meistern und den faecheruuebergreifenden 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 entwurfen 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.

In dieser Bachelor-Arbeit wird eine solche Architektur für ein verteiltes Dateiverwaltungssystem beschrieben und entworfen.

2. Related Work

3. Concept

4. Implementation

5. Evaluation

6. Future Work

7. Conclusion

A. Appendix

```java
public class SensorActivity extends Activity implements SensorEventListener {
  private SensorManager sensorManager;
  private Sensor accelerometer;

  @Override
  public final void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
  }

  @Override
  protected void onResume() {
    super.onResume();
    sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
  }

  @Override
  protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(this);
  }

  @Override
  public final void onSensorChanged(SensorEvent event) {
    StringBuilder log = new StringBuilder("Acceleration:");
    log.append(" X: ").append(String.valueOf(event.values[0]));
    log.append(" Y: ").append(String.valueOf(event.values[1]));
    log.append(" Z: ").append(String.valueOf(event.values[2]));
    System.out.println(log.toString());
  }

  @Override
  public final void onAccuracyChanged(Sensor sensor, int accuracy) {
    // sensor accuracy changed
  }
}
```

Listing 1: Activity with lifecycle callbacks