Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung

Mario Kaufmann
Pascal Birnstill
Erik Krempel

Implementierung

Version 1.0

Privacy Crash Cam App für Android

Fabian Wenzel
Giorgio Gross
Christoph Hörtnagl
David Laubenstein
Josh Romanowski

15. Februar 2017

Inhaltsverzeichnis

1 Einleitung **3**

2 Änderungen **4**
2.1 App . . . 4
2.1.1 AsyncPersistor . . . 4
2.1.2 Encryptor . . . 4
2.1.3 Ringbuffer . . . 4
2.2 Web-Dienst . . . 5
2.2.1 ServerProxy . . . 5
2.2.2 Main . . . 5
2.2.3 AccountManager . . . 5
2.2.4 VideoManager . . . 5
2.2.5 Account . . . 5
2.2.6 Metadata . . . 6
2.2.7 LocationConfig . . . 6
2.2.8 VideoProcessingChain . . . 6
2.3 Interface . . . 6
2.3.1 Accountverwaltung . . . 6
2.3.2 Download . . . 6
2.3.3 Account . . . 6
2.3.4 VideoTable . . . 6
2.3.5 Video . . . 6

3 Implementierungsplan **7**
3.1 Einleitung . . . 7
3.2 Ursprünglicher Implementierungsplan . . . 8
3.3 Tatsächlicher Implementierungsplan . . . 9
3.4 Gründe für die Unterschiede . . . 10

1 Einleitung

Dieses Dokument versucht einen Überblick darüber zu geben, welche Änderungen vorgenommen wurden, um aus unserem Entwurfsdokument für die *PrivacyCrashCam* ein vollständiges, funktionierendes Programm zu erstellen. Unser Ziel ist hierbei, die Gründe für die Änderungen zu erläutern und aufzuzeigen, wie der Entwurf geändert wurde um Probleme zu beheben.

Daher wird zu jedem Modul kurz beschrieben, ob, was (Attribute, Methoden, Parameter) und wieso verändert wurde (2). Zudem wird auf allgemeine Probleme, die bei der Implementierung auftraten, wie Probleme mit Build-Tools eingegangen.

Am Schluss des Dokuments befindet sich ein Vergleich zwischen unserem ursprünglichen und tatsächlichen Implementierungsplan (3) und entsprechende Erläuterungen, weshalb sich diese gegebenenfalls unterscheiden (3.4).

2 Änderungen

2.1 App

2.1.1 AsyncPersistor

Dem Konstruktor muss zusätzlich der Context übergeben werden, damit das Public Key File für das Encryption-Modul geladen werden kann.

2.1.2 Encryptor

Die Methode encrypt() von Encrpytor nimmt nun die Parameter encrypt(File[ ] input, File[ ] output, InputStream publicKey, File encKey). Da nun File Arrays übergeben werden wird erlaubt eine beliebige Anzahl Files mit einem einzigen SecretKey zu verschlüsseln und dadurch eine bessere Erweiterbarkeit erreicht.

Der InputStream ist nötig, da Android nur Klassen mit zugriff auf den Context erlaubt Resourcen zu laden. Dies wirkt sich auch auf IKeyEncryptor aus.

2.1.3 Ringbuffer

Uns war von Beginn an klar, dass der Ringbuffer der App, der die Video-Aufzeichnungen der App vor der Persistierung puffert, eine Herausforderung werden würde. Daher haben wir uns bereits in der Entwurfsphase intensiv mit der Umsetzung auseinander gesetzt und uns dafür entschieden kurze Video-Stücke in einer Warteschlange zu speichern und diese beim Persistieren zusammenzufügen.

Bei der Implementierung stießen wir auf das Problem, dass der MediaRecorder, der die Video-Stücke aufnimmt, asynchron zum Schreiben des Videos Rückmeldung gibt, er hätte die Aufnahme beendet. Dennoch waren die Video-Files zu dem Zeitpunkt nicht immer vollständig geschrieben, sodass wir einen Mechanismus entwickeln mussten, um sicherzustellen, dass die Files fertig geschrieben werden, bevor wir versuchen zu persistieren. Unser erster Versuch war einen Android-FileObserver auf die Files zu setzen und dadurch Rückmeldung zu erhalten, wenn die Files fertig geschrieben sind und beim Abrufen der Daten zu warten, bis dies für alle Files in der Warteschlange gilt. Es wurden jedoch nicht alle Events empfangen, da manche Dateien bereits vor dem Einfügen in die Warteschlange fertig waren. Die Funktion getData() wurde in demandData() umbenannt um klarzustellen, dass eventuell Wartezeit auftreten kann.

Dieses Problem konnten wir schließlich lösen, indem wir den FileObserver auf dem Ordner mit den Videos setzten anstatt auf die Videos selbst.

Um die Implementierung zu erleichtern, entschieden wir uns zudem den Ringbuffer nicht mehr generisch zu Implementieren und haben ihn infolge dessen zu VideoRingBuffer umbenannt.

Für eine erweiterte Funktionalität wurden die Methoden flushAll() und den Inhalt des Puffers zu leeren und destroy() zum Löschen des Puffers hinzugefügt.

2.2 Web-Dienst

2.2.1 ServerProxy

Der upload()-Methode wurde ein weiterer Parameter hinzugefügt um Informationen über den Video-Namen weitergeben zu können.

getVideosByAccount() wurde zu getVideos() umbenannt, da der Parameter accountData den Suffix redundant macht.

2.2.2 Main

Es wurde eine Funktion restartServer() hinzugefügt, falls man zusätzliche Logik einfügen möchte, die nicht der startServer() und stopServer() Routine entspricht (z.B. zum schnelleren Start bei täglichen Neustarts).

2.2.3 AccountManager

Die Methoden setMail() und setPassword() wurden zu changeAccount() zusammengefasst, um die Schnittstelle zu vereinfachen.

Damit der man der Account Klasse das Salt zum hashen mitgeben kann wurde die Funktion getSalt() hinzugefügt, die das Salt des Accounts aus der Datenbank abfragt.

2.2.4 VideoManager

Da Web-Dienst und Web-Interface zwei vollständig unabhängige Projekte sind ist eine Kommunikation über die Datencontainerklasse VideoInfo nicht direkt möglich. Um dennoch Informationen austauschen zu können wurde die Methode getVideoInfoList() so abgeändert, dass sie nun einen JSON-String erstellt, der versendet und auf dem Interface wieder ausgelesen werden kann.

2.2.5 Account

Das lesen des Hash-Salt aus der Datenbank wird vom AccountManager geregelt, da die Account Klasse keinen Zugriff auf die Datenbank hat. Daher war es notwendig die Methode hashPassword() öffentlich zu machen.

2.2.6 Metadata

Die Typen von date (String -> long) und von gForce (Vector3D -> float[]) wurden angepasst und dem Pendant der App zu entsprechen.

2.2.7 LocationConfig

Der LocationConfig mussten weitere Locations für Resources-Ordner hinzugefügt werden, falls man die Ordnerstruktur verändern möchte.

Da wir uns entschieden haben einen eigenen Ordner für log-Files zu erstellen wurde auch dieser hinzugefügt.

2.2.8 VideoProcessingChain

Um es bei der Fehlerbehandlung zu ermöglichen einzelne Chains aufzuräumen, wurde eine Funktion cleanUp() hinzugefügt. Diese dupliziert zur Zeit die Funktionalität von deleteTmpFiles(), ist aber nützlich falls man die Fehlerbehandlung erweitern möchte.

Es wurden getter für response und videoName hinzugefügt.

2.3 Interface

2.3.1 Login

Die LoginView wird bei jedem Start angezeigt noch bevor der Navigator erzeugt wird. Dadurch wird verhindert, dass durch vor und zurückspringen der Login übergangen werden kann. Zusätzlich hat die MyUI Klasse eine Methode login() bekommen, die den Navigator und das Menü erzeugt und dann zur VideoView wechselt.

2.3.2 AccountView

Die AccountView hat eine Refernez auf die UI bekommen, somit kann nach Ändern oder Löschen eines Accounts der Benutzer ausgeloggt wird.

2.3.3 AccountDataManager

Die Klasse hatte einen Account als Klassen Attribut, dies wäre zum Problem geworden wenn mehrere Benutzer gleichzeitig eingeloggt sind. Deshalb wurde dieses Attribut durch ein von Vaadin bereitgestelltes Sessionattribut ersetzt. Nun kann über wo der Account benötigt wird er aus der Session via key gelesen werden.

2.3.4 VideoDataManager

2.3.5 Download

viel

2.3.6 Account

Da wie bei dem VideoManager keine Datenobjekte zwischen Interface und Dienst ausgetauscht werden können, wird auch hier nun ein JSON-String erstellt. Dafür wurde die Methode getAsJson() eingeführt.

2.3.7 Video

Beim erzeugen der Liste der Videos, werden zuerst die Videos in die Liste eingefügt und anschließend die Meta Informationen hinzugefügt. Deshalb wird ein Konstruktor in der Klasse Video benötigt, mit dem man ein Video ohne Meta Informationen erzeugen kann.

2.3.8 MailService

Die Klasse hat eine Methode bekommen mit der sie e-mail Adressen auf korrekte Syntax prüfen kann.

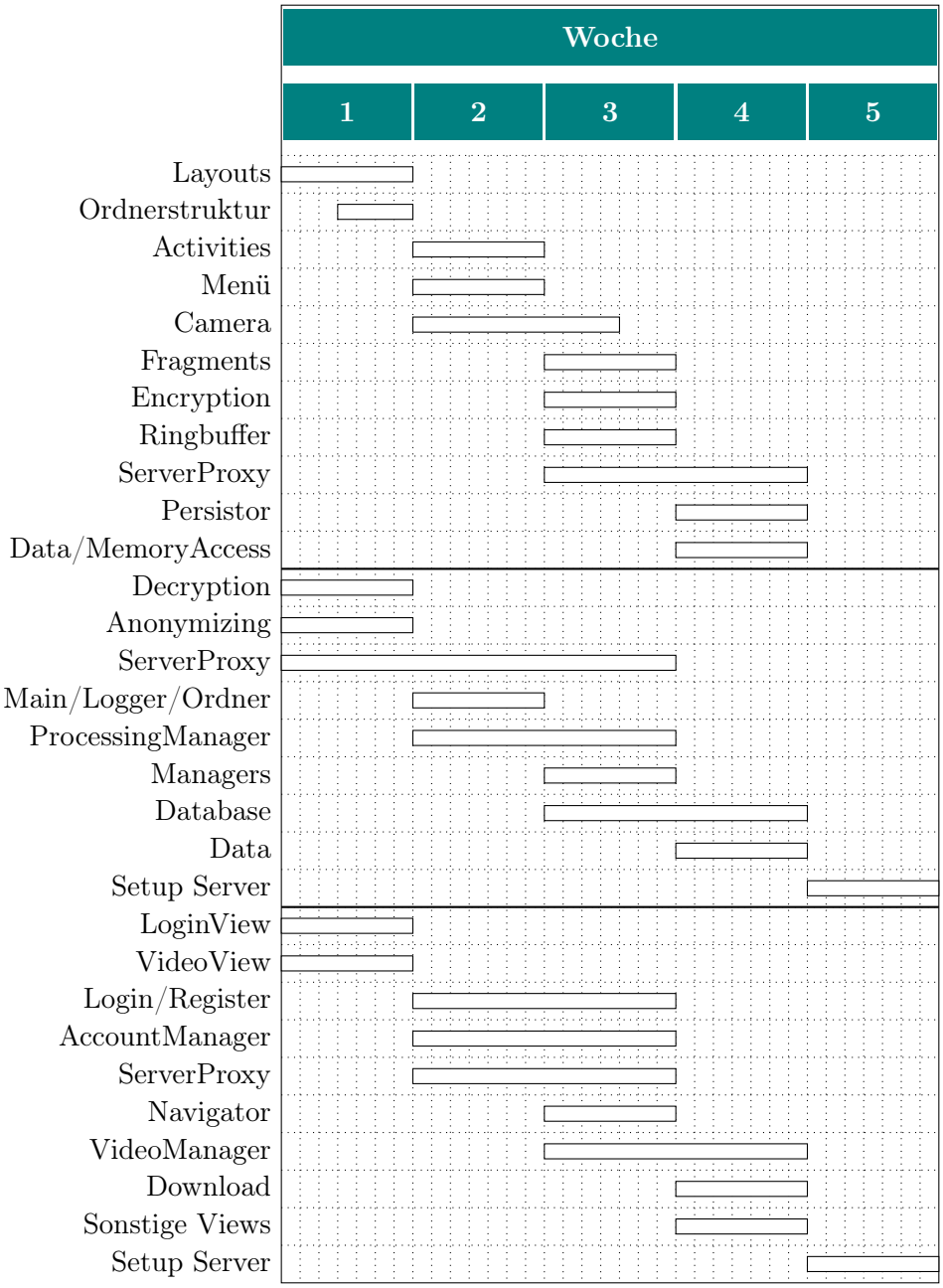
3 Implementierungsplan

3.1 Einleitung

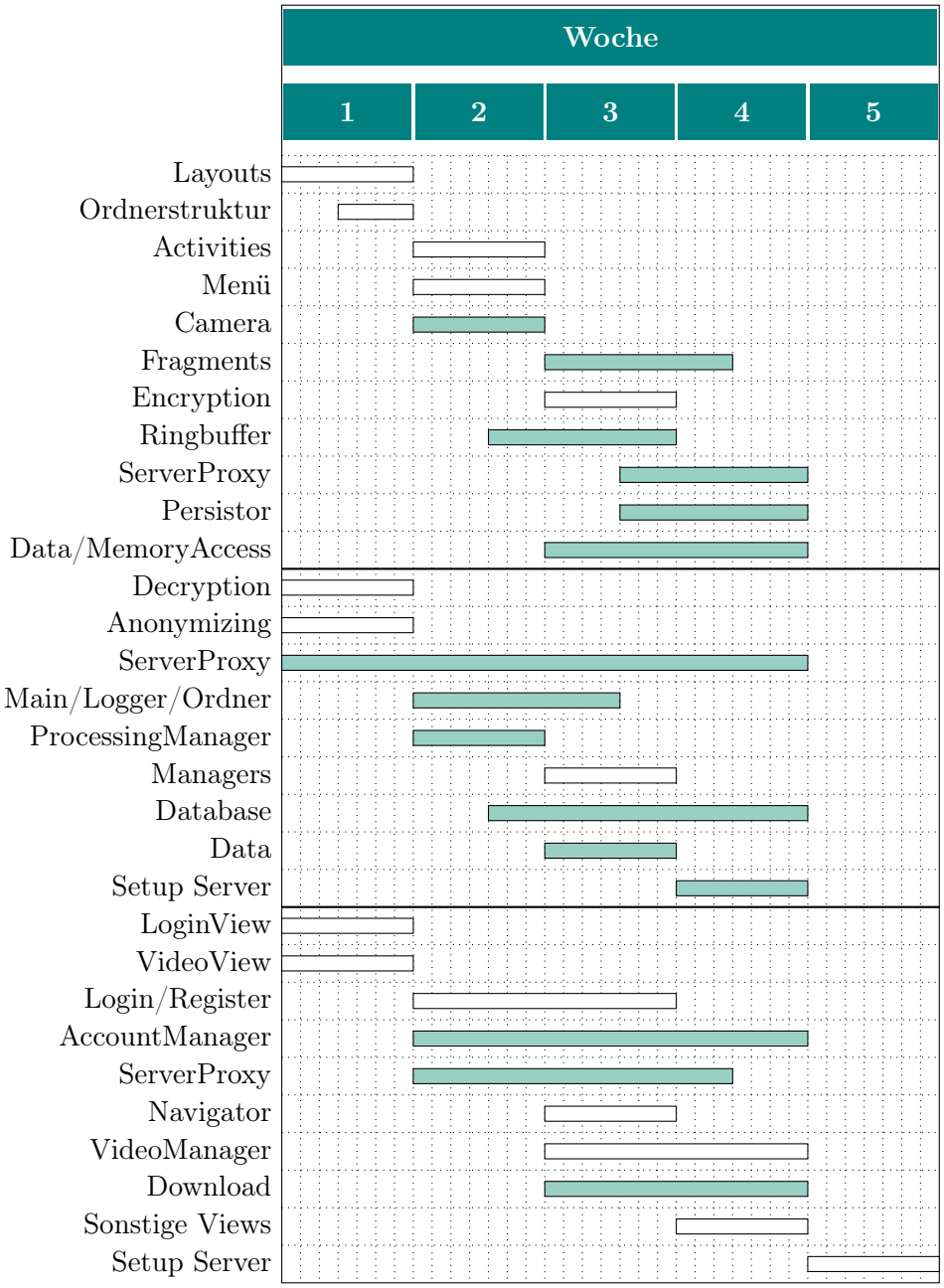
Um eine schnelle und reibungslose Implementierung zu garantieren haben wir einen Plan erstellt, in dem wir festhalten, welches Modul wann implementiert wird.

Bei der Erstellung des Plans war unser zentrales Ziel eine hierarchische Implementierung zu erreichen. Das heißt, dass zunächst die Grundfunktionalität implementiert wird und danach schrittweise weitere, immer weniger essentielle Komponenten hinzugefügt werden. Dadurch wird gewährleistet, dass Probleme, die die Umsetzung der *PrivacyCrashCam* als Ganzes gefährden frühzeitig erkannt und behandelt werden können.

3.2 Ursprünglicher Implementierungsplan



3.3 Tatsächlicher Implementierungsplan



3.4 Gründe für die Unterschiede

- **Ringbuffer/Persistor**
  Durch bereits genannte Probleme mit der Umsetzung des Ringbuffers (2.1.3) hat sich die Fertigstellung verzögert. Da der Persistor von den Daten des Ringbuffers abhängt, hat auch er sich verzögert.

- **Data/Memory Access**
  Nicht alle Methoden des MemoryAccess-Moduls wurden direkt benötigt. Daher haben wir die Fertigstellung des Moduls verzögert, um zuerst den Persistor fertig stellen zu können.

- **ServerProxy**
  Da nach und nach die Funktionalität des Web-Interfaces und der App erweitert wurden, stellten sich auch immer wieder neue kleine Probleme mit dem ServerProxy heraus, wodurch immer wieder kleine Änderungen nötig wurden und die Fertigstellung verzögerten.

- **Database**
  Weil die Implementierung des Passwordhashens eine Anpassung des DatabaseManagers erforderte, musste der eigentlich abgeschlossene Manager noch einmal aufgegriffen werden.

- **ServerProxy/AccountManager (Interface)** Da der der ServerProxy des Interfaces mit dem ServerProxy des Web-Dienstes abgeglichen werden musste (3.4) hat sich die Fertigstellung des AccountManagers ebenfalls herausgezögert.

- **Download**
  Wie bereits erläutert erwies sich der Video-Download des Web-Interface als problematisch (2.3.2). Infolge dessen wurde auch hier ein Mehraufwand nötig.

- **Beschleunigungen**
  Durch die Vorarbeit in der Entwurfsphase konnten einige Module schneller abgeschlossen werden als zunächst vermutet. Einzelne Funktionen waren bereits geschrieben und getestet und mussten daher nur noch eingefügt werden. Außerdem hatten wir uns schon vor der Implementierung mit den Technologien auseinandergesetzt, was den Implementierungsprozess beschleunigte.
  - Camera-Modul App
  - VideoProcessingManager Web-Dienst
  - Server Setup Web-Dienst
  - Data-Modul Web-Dienst