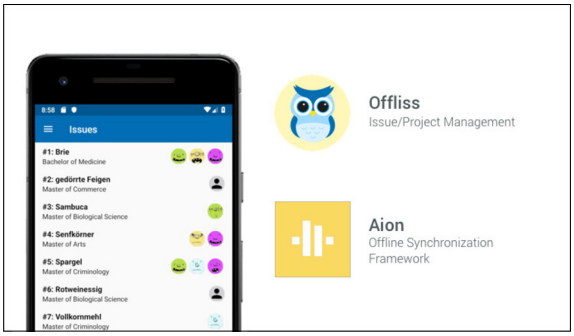
Pascal
Bertschi

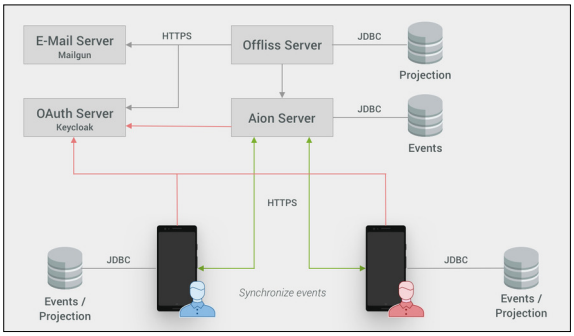
Diplomand	Pascal Bertschi
Examinator	Prof. Stefan F. Keller
Experte	Claude Eisenhut, Eisenhut Informatik AG, Burgdorf, BE
Themengebiet	Software Engineering - Core Systems

Event-basierte Offline-Synchronisation

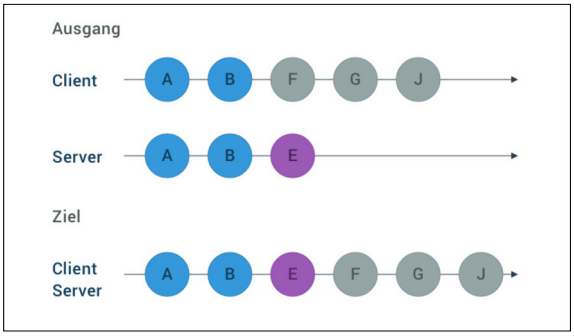
Datensynchronisation für mobile Applikationen



Screenshot vom Mobile App "Offliss" mit fiktiven Issues sowie dessen Logo und das Logo vom Framework "Aion"
Eigene Darstellung



Deployment-Diagramm mit zwei Offliss-Clients im Zusammenspiel mit Aion und externen Systemen (E-Mail und OAuth)
Eigene Darstellung



Synchronisation einer Client Event-Timeline, die mit der Server-Event-Timeline zusammen geführt wird
Eigene Darstellung

Ausgangslage: Viele Frameworks und Datenbank-Produkte werben mit Offline-Fähigkeiten. Die Umsetzung von Offline-Anwendungen ist jedoch aufwendiger als man denkt. Bei der Softwareentwicklung von mobilen Offline-Applikationen muss man sich mit neuen, bisher unbekannten Herausforderungen auseinandersetzen. Beispielsweise müssen Kompromisse eingegangen werden, um die Funktionalität den Anforderungen entsprechend umzusetzen. Die eingeschränkten Ressourcen eines mobilen Endgeräts (Speicherplatz, CPU/Performanz) sind zusätzliche Schwierigkeiten. Des Weiteren kann auf den mobilen Geräten nicht dieselbe Software zur Datenverwaltung eingesetzt werden wie im Backend, weil mobile Apps offline nicht mit dem Server interagieren können und andere Software-Bibliotheken und Betriebssysteme verwenden.

Ziel der Arbeit: Aufgrund der gesammelten Ergebnissen einer Auswahl bestehender Lösungen, soll ein Framework names "Aion" realisiert werden. Das Resultat soll das Entwickeln einer Offline-Anwendung sowohl vereinfachen als auch beschleunigen. Anhand eines Anwendungsbeispiels aus der realen Arbeitswelt, "Offliss" genannt, wird die Funktionsweise des Frameworks veranschaulicht.
Die Lösung soll universell einsetzbar sein und die vielfältigsten Anforderungen erfüllen, die aus eigener Erfahrung zusammengestellt wurden. Die Aktionen der einzelnen Benutzer, wie das Erstellen, Ändern oder Löschen eines Datensatzes, sollen durch Synchronisation zusammengeführt werden können. Konflikte, die eventuell entstehen können, werden ohne Benutzereingriffe aufgelöst. Diese Synchronisations-Entscheidungen sind von den Benutzern einseh- und nachvollziehbar. Wo nötig könnten sie eingreifen und die Entscheidung durch Überschreiben korrigieren.

Ergebnis: Das Aion-Framework wurde mit dem Architektur-Pattern "Event Sourcing" umgesetzt. Der Ansatz, lediglich die Zustandsübergänge (Events) zu synchronisieren und nicht eine Differenz der Zustände, hat sich bewährt. In der Applikation werden die Events zu sogenannten "Projections" verarbeitet. Projections sind mit objektorientierten Datenbank-Views vergleichbar. Event Sourcing ist gut test- und nachvollziehbar, sowohl für Entwickler als auch für Anwender. Jede Benutzeraktion stellt ein Event auf einer "Timeline" (Zeitachse) dar. Der Server bestimmt die Reihenfolge der Events. Die Endgeräte passen sich an diese an und fügen ihre Events als Letztes ein.
Die Schnittstellen von Aion sind projekt- und technologieunabhängig. Die Kommunikation mit den Umsystemen erfolgt über HTTPS. Durch Konfigurationen lässt sich Aion steuern, ohne dass in dessen Code Anpassungen vorgenommen werden müssen. Aion wurde in Java realisiert.
Offliss ist eine Issue-Verwaltung, die als Android-App ebenfalls in Java umgesetzt wurde. Alle Benutzeraktionen, wie "Projekt hinzufügen", "Issue bearbeiten" etc., sind offline verfügbar. Ausserdem wurden externe Systeme wie der OAuth-Server Keycloak (Autorisierung) und der E-Mailserver Mailgun angebunden. Mit Offliss wurde demonstriert, dass Aion alle definierten Anforderungen lösen kann.