



Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Computer Science Department

Semesterprojekt

Serviceorientierte Architekturen im
Masterstudiengang Informatik

HighPerformance

von Nikolas Rist, Kai Bepperling

Erstprüfer: Prof. Dr. Sascha Alda

Eingereicht am: 20. September 2019

Inhaltsverzeichnis

Einführung und Ziele	2
Aufgabenstellung	2
CEO	2
Senior HR Consultant	2
IT-Admin	2
Szenarien	3
Qualitätsziele	3
Stakeholder	3
Randbedingungen	4
Kontextabgrenzung	6
Fachlicher Kontext	6
Technischer Kontext	6
Bausteinsicht	9
Laufzeitsicht	10
Verteilungssicht	11
Querschnittliche Konzepte	11
Entwurfsentscheidungen	12
Retroperspektive	13
Skalierung	13
Risiken und Herausforderungen	13
Offene Punkte	14

Einführung und Ziele

Ziel des Projekts ist die Optimierung der Performanceevaluierung der Mitarbeiter der *SmartHoover Ltd.* Im Vorraus wurden Interviews mit den Stakeholdern geführt, um das Projekt konkret definieren zu können.

Aufgabenstellung

Der bisherige Prozess erfordert viele manuelle Schritte und findet zum Großteil auf Papier statt. Die Informationen sind allerdings in den Systemen OpenCRX und OrangeHRM bereits vorhanden und sollen auch zukünftig genutzt werden. Die verschiedenen Stakeholder, welche im Abschnitt beschrieben werden, haben in einem Interview einige Anforderungen an das neue System gestellt.

CEO

- Die berechneten Boni müssen manuell freigegeben werden.
- Änderung der berechneten Boni muss vor Freigabe möglich sein.
- Vereinfachung des gesamten Prozesses durch weniger manuelle Schritte.

Senior HR Consultant

- Der Prozess muss beschleunigt werden.
- Weniger manuelle Interaktionen mit den Systemen OrangeHRM, OpenCRX und dem Datenbanktool.
- Weniger Papierarbeit.

IT-Admin

- Loslösung aus dem Prozess.

Szenarien

In diesem Abschnitt werden zwei mögliche Nutzungsszenarien dargestellt.

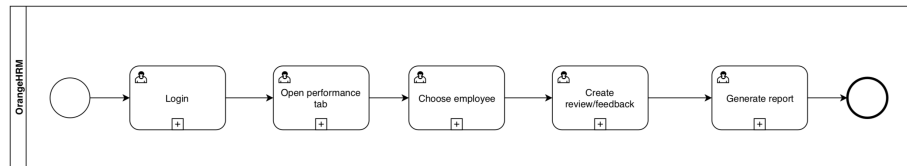


Abbildung 1: Manuelle Berechnung der Boni

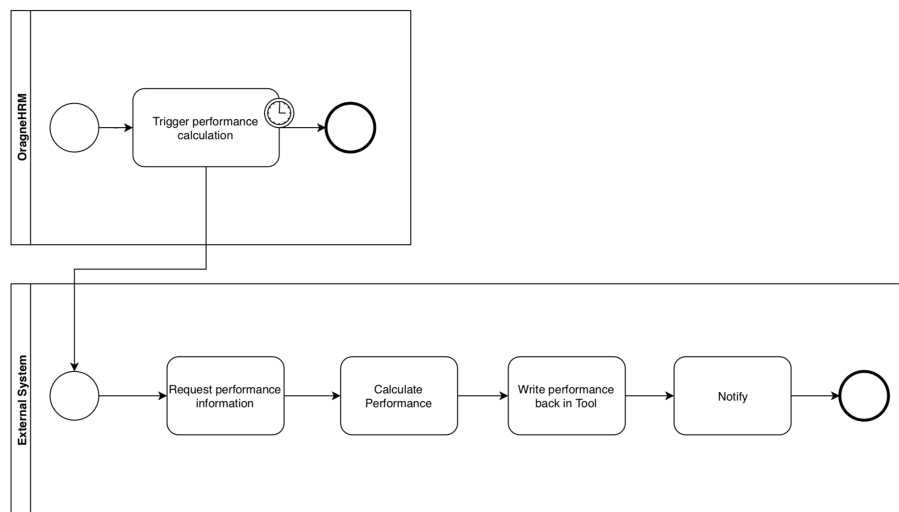


Abbildung 2: Vollautomatisierte Berechnung der Boni

Qualitätsziele

1. Prozessbeschleunigung
2. Weniger manuelle Schritte
3. Weniger Prozessteilnehmer

Stakeholder

Rolle	Kontakt	Erwartungshaltung
<i>CEO</i>	<i>Dr. Michael Moore</i>	<i>Keine Vollautomatisierung, Segnet alle Boni manuell ab</i>
<i>HR Senior Consultant</i>	<i>Chantal Banks</i>	<i>Weniger manuelle Schritte, Keine Papierarbeit mehr</i>
<i>IT-Admin</i>	<i>Tom Foster</i>	<i>Keine Berührungspunkte mit dem gesamten Prozess</i>

Randbedingungen

Beschreibung der zu nutzenden externen Systemen

Was ist die prinzipielle Aufgabe der Anwendung OpenCRX?

- CRM System
- Account management
- Product and Price Management
- Sales Pipeline
- Issue tracking
- Groupware
- mail, contact and calendar management

Was ist die prinzipielle Aufgabe der Anwendung OrangeHRM?

- Resource Management
- performance Management
- Administration and Personal Information Management
- Recruitment etc.
- Employee Self Service (Time Tracking)

Welche grundlegenden Funktionen besitzen diese Anwendungen?

OrangeHRM bietet Funktionalitäten zum Verwalten des Personals und alles was dazu gehört:

- Mitarbeiterverwaltung inkl. Mitarbeiter Self Service
- Dashboards
- Mitarbeiter Training
- Reiseplanung
- Dokumentenmanager
- OpenCRX bietet Funktionalitäten zur Verwaltung von Kunden
- Kundensupport
- Customer Success
- Marketing
- Analytics
- Sales Management

Welche Geschäftsobjekte werden in OpenCRX bzw. OrangeHRM verwaltet?

- Kunden
- Mitarbeiter

Welche Art Schnittstellen bieten OpenCRX bzw. OrangeHRM an?

- OpenCRX:
 - REST API
 - AirSync ActiveSync
 - User Interface (WebUI)
- OrangeHRM:
 - REST API
 - Mobile APP

– User Interface (WebUI)

Kontextabgrenzung

Fachlicher Kontext

Der Prozess kann von einem HR Senior über das Userinterface der Camunda Plattform (Tasklist) gestartet werden. Beim Start muss der entsprechende Mitarbeiter in das angebotene Formular-Feld eingetragen werden.

Weiterhin wird dem CEO in Camunda ein entsprechendes Formular mit einer Aufstellung der berechneten Boni zur Verfügung gestellt. In diesen kann er Korrekturen an den einzelnen Boni vornehmen und entsprechend speichern damit diese im fortlaufenden Prozess berücksichtigt werden.

Technischer Kontext

Die Camunda-Engine nutzt die bereitgestellte RESTful-Schnittstelle des Data-Collectors, um entsprechende Informationen über den zu bearbeitenden Salesman auszutauschen. Hierbei wird das sogenannte ClientInfoDTO ausgetauscht.

Data Collector - Objekte

```
data class ClientInfoDTO(  
    var salesmanName: String = "",  
    var salesInfos: ArrayList<SalesInfoDTO> = ArrayList()  
)  
  
data class SalesInfoDTO(  
    var productName: String = "",  
    var clientName: String = "",  
    var clientRanking: Int = 0,  
    var quantity: Double = 0.0,  
    var bonus: Int = 0  
)
```

Zum kommunizieren mit den externen Systemen, OrangeHRM und OpenCRX, nutzt der Data-Collector eine entsprechende RESTful Schnittstelle.

OrangeHRM - Objekte

```
data class Employee(  
    val firstName: String,  
    val lastName: String,  
    val employeeId: String,  
    val jobTitle: String  
)  
  
data class Organization(  
    val id: String,  
    val name: String,  
    val email: String?,  
    val country: String,  
    val numberOfEmployees: String  
)
```

OpenCRX - Objekte

```
data class Account(  
    val firstName: String? = "",  
    val lastName: String? = "",  
    val fullName: String? = "",  
    val familyStatus: Int = 0,  
    val organization: String? = "",  
    val jobTitle: String? = "",  
    val gender: Int = 0,  
    val preferredSpokenLanguage: Int = 0,  
    val accountRating: Int = 0,  
    val industry: String? = "",  
    val annualIncomeCurrency: Int = 0,  
    @JsonProperty("@href") val accountUrl: String? = ""  
)
```

Abbildung 3 zeigt die entsprechenden Kommunikationskanäle via HTTP zwischen den technischen Systemen.

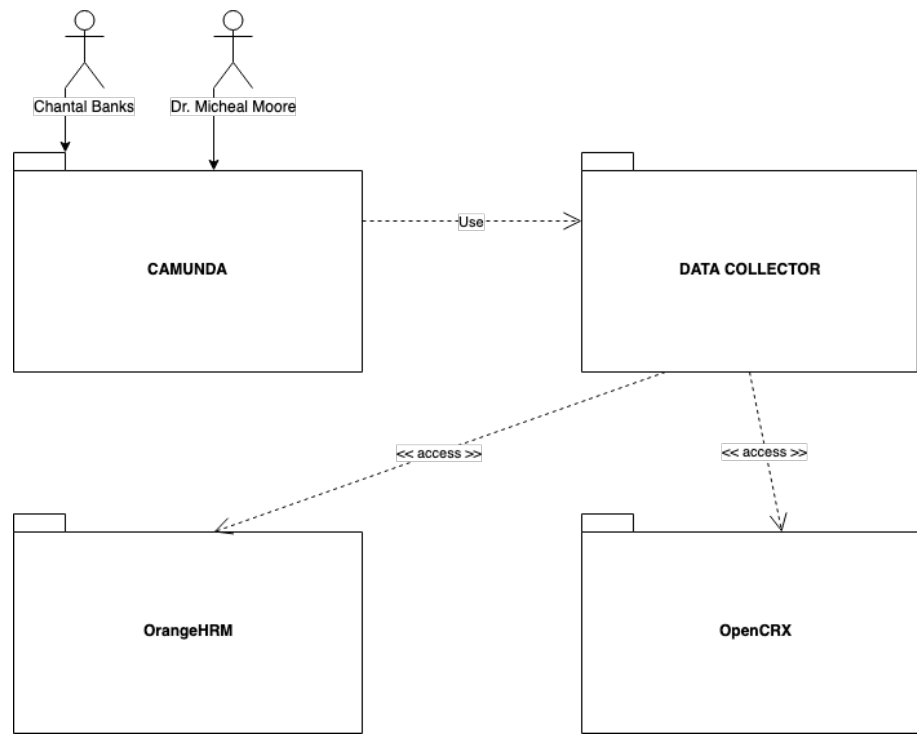


Abbildung 3: Kontext des Systems

Bausteinsicht

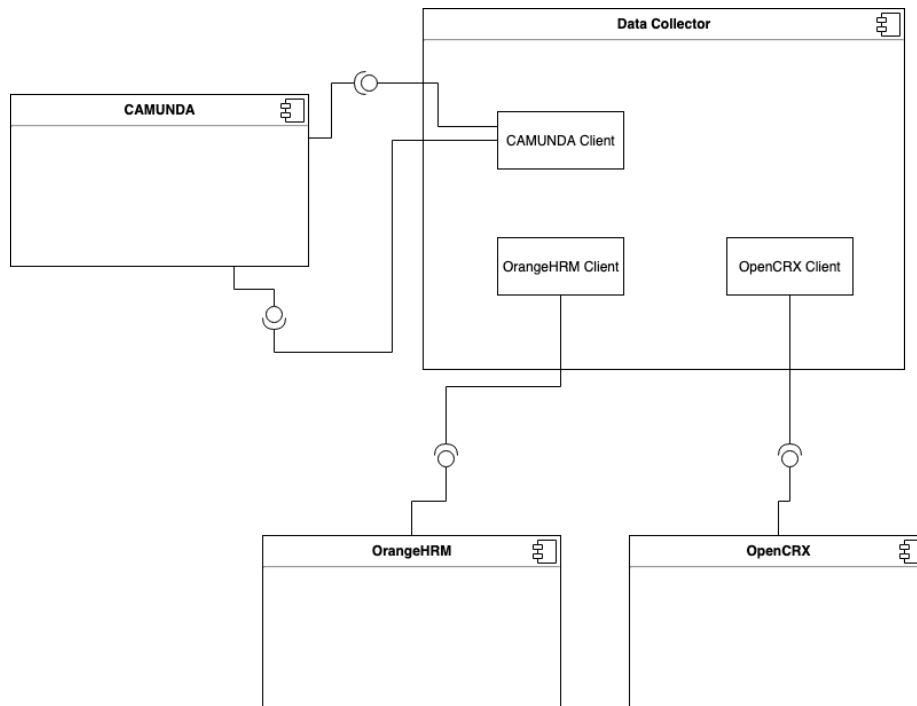


Abbildung 4: Bausteinsicht

Laufzeitsicht

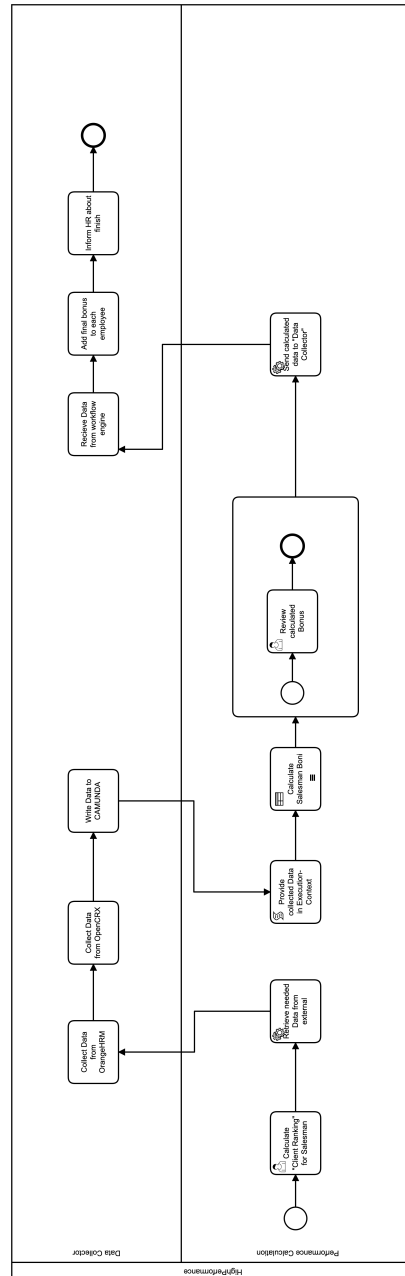


Abbildung 5: Laufzeitsicht

Verteilungssicht

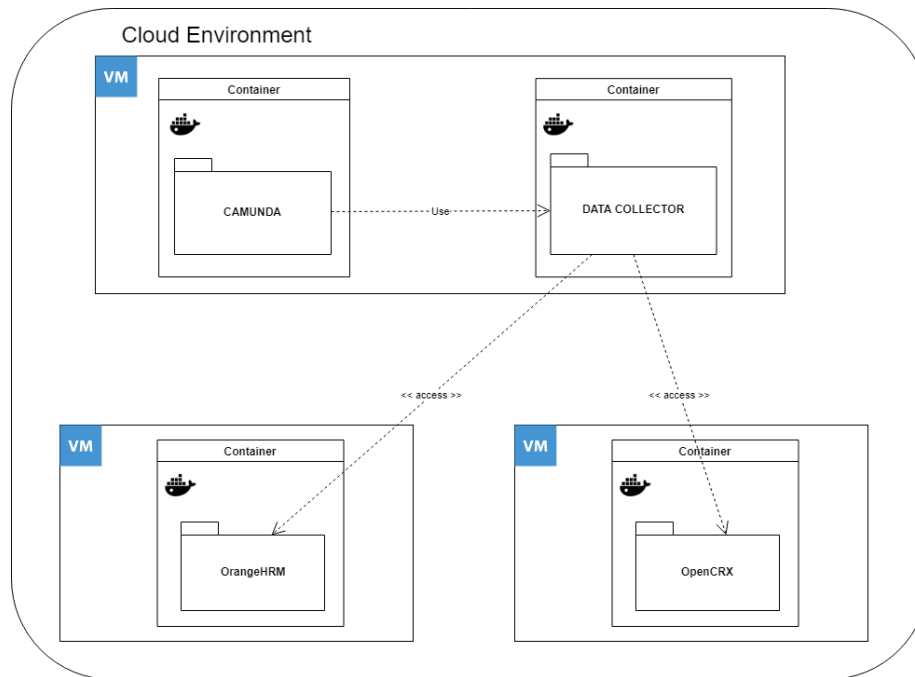


Abbildung 6: Verteilungssicht

Querschnittliche Konzepte

Dieser Abschnitt beschreibt übergreifende, prinzipielle Regelungen und Lösungsansätze, die an mehreren Stellen (= *querschnittlich*) relevant sind.

Architekturstil

Wir nutzen das Architektur-Muster der Microservices, da wir einen Integrations-Service zwischen den einzelnen Systemen benötigen. Dieser soll nur durch eine RESTful-API abstrahiert und darüber angesprochen werden. Diese Abstraktion bietet die Möglichkeit den Service auszutauschen, ohne die Camunda-Plattform anpassen zu müssen.

Deployment

Das einfache Austauschen wird begünstigt durch die Entscheidung, den Service

als Docker-Container zur Verfügung zu stellen und zu betreiben. Somit ist er Plattform unabhängig und kann im Bedarfsfalls mehrfach deployed werden.

Workflow-Engine

Die Camunda-Plattform wurde als Basis für die Workflow-Implementierung genutzt, da sie eine out-of-the-box Nutzeroberfläche zur Verfügung stellt und somit einen guten Überblick für nicht-technische Mitarbeiter ermöglicht. Weiterhin kann der gesamte Prozess mittels BPMN standardisiert visualisiert werden. Camunda bietet des Weiteren ein Nutzeroberfläche, um den aktuellen Stand des Prozesses detailliert zu betrachten.

Data-Collector

Der Data-Collector bildet das Herzstück des Gesamtsystems, da es alle drei genutzten Services OpenCRX, OrangeHRM und Camunda miteinander verbindet. Dieser Service wurde in der Sprache Kotlin, mit dem Web-Framework Ktor, entwickelt. Ktor bietet einen modular aufgebauten Web-Server, mit dem man leichtgewichtige RESTful APIs implementieren kann. Kotlin und Ktor sind *state-of-the-art* Technologien. Für eine Weiterentwicklung des Services besteht durch eine solch junge Programmiersprache allerdings kein Risiko, da Kotlin inherent mit Java verwendet werden kann.

Entwurfsentscheidungen

Zur prototypischen Implementierung wurden verschiedene Sprachen und Technologien verwendet, welche in diesem Abschnitt erläutert werden.

Wie vorgegeben wurde zur modellierung des Workflows das Tool Camunda eingesetzt. Mithilfe von Camunda wurde der gesamte Prozess modelliert und ausgeführt. Das Camunda Cockpit und die Tasklist eignen sich hervorragend zum starten und verwalten der Prozesse. Da im verlaufe des Prozesses auch mit Fremdsystemen kommuniziert werden muss, wurde ein Microservice in der Sprache Kotlin entwickelt. Dieser Service übernimmt die Kommunikation mit OrangeHRM und OpenCRX über die dokumentierten RESTful API's. Um eine konsistente Architektur bereitzustellen bietet der Microservice ebenfalls RESTful Schnittstellen an, welche aus der JavaDelegate Klasse aus Camunda heraus aufgerufen wird. Alle Anwendungen werden in eigenen, weitestgehend isolierten, und orchestriert in Docker Containern bereitgestellt.

Retroperspektive

Skalierung

In der aktuellen prototypischen Implementierung existieren zwei Skalierungsprobleme. Die erste Herausforderung ist das starten des Prozesses. In der aktuellen Implementierung kann die Bonusberechnung nur für einen Mitarbeiter durchgeführt werden. In einer Weiterentwicklung des Tools müsste es erreicht werden, dass der Prozess automatisch alle Salesmen herausfiltert und die Bonusberechnung startet. Das zweite Problem bezüglich der Skalierung ist die Anforderung, dass der CEO Michael Moore die berechneten Boni manuell überprüfen und freigeben muss. Der implementierte Microservice *data-collector* wurde von Beginn an skalierbar entwickelt, sodass ausschließlich die Anforderung auf Kundenseite angepasst werden muss.

Risiken und Herausforderungen

Im Verlaufe des Projekts haben sich einige Teilbereiche als Herausforderung dargestellt. Der Informationsaustausch zwischen Camunda und dem Microservice bzw. der Entscheidungstabelle war aufgrund unpräziser Dokumentation schwierig. Ebenso war die Erstellung der Camunda Forms nicht offensichtlich dokumentiert, sodass eine weitreichende Recherche durchgeführt werden musste.

Es stellte sich außerdem heraus dass die eingesetzten Fremdsysteme OpenCRX und OrangeHRM komplex modellierte API's anbieten. Bei OrangeHRM sind zwei verschiedene API versionen dokumentiert, wobei nur eine spezielle Version eingesetzt wird, welche allerdings nicht alle Funktionalitäten anbietet. Aus diesem Grund musste für das Zurückschreiben der berechneten Boni ein *custom-field* angelegt werden.

Offene Punkte

In diesem Abschnitt werden die offenen Punkte der prototypischen Implementierung aufgezeigt

- Automatische Prozesswiederholung

- Automatische Zuweisung des Prozesses an CEO
- Berechnung für alle Salesmen
- Vollautomatisierung (beschränkt durch Anforderungen des Kunden)
- API Clients in eigene Services extrahieren