# Tübinger Softwareprojekt 2015/16: Authentifizierungs- und Autorisierungssystem

Teilnehmer: Clemens Geibel, Lucca Hellriegel, Andreas Herzog, Joshua Hartmann,
Thomas Sachs, Mario Rose, Phil Szalay, Maximilian Rösch
Betreuer: Tillmann Rendel

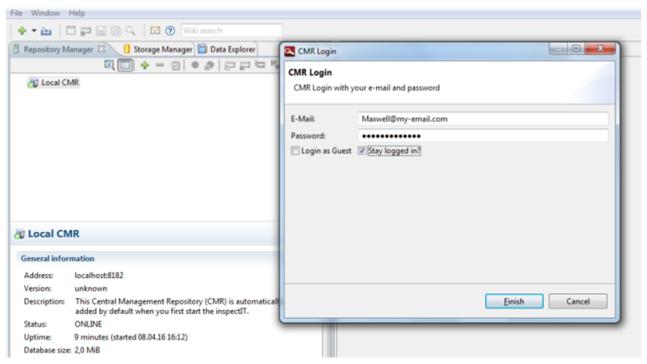
## 1 Einleitung

Im Rahmen des Tübinger Softwareprojekts 2015/16 hatten wir die Gelegenheit an einem Java Open Source Projekt mitzuwirken. Die Firma NovaTec GmbH entwickelt seit einigen Jahren ein Analyse-Werkzeug zur Performance-Optimierung von Java-Anwendungen mit dem Namen inspectIT. Dieses wurde im Jahr 2015 zu einem Open Source Projekt, welches wir als Team von acht Studenten erweitern durften. Genauer gesagt galt es, eine Benutzerverwaltung zu integrieren, die die kritischen Funktionen des Client-Server-Netzwerkes schützt und die Nutzung dieser überwacht.

## 2 inspectIT

InspectIT besteht aus drei Komponenten, dem inspectIT Agent, dem CMR (Central Measurement Repository) und dem Client.
Der inspectIT Agent wird in die Anwendung integriert, die überwacht bzw. analysiert werden soll.
Der CMR erhält die Messdaten von einem oder mehreren Agents und stellt diese dem Client zur Verfügung.
Der Client wird zur Analyse und zum Monitoring der Anwendung verwendet und lässt sich über ein User Interface steuern. Um Daten zu erhalten und Aktionen durchzuführen, stellt er Anfragen an den CMR, kann jedoch auch mit lokalen Daten arbeiten.

## 3 Sicherheitskonzept

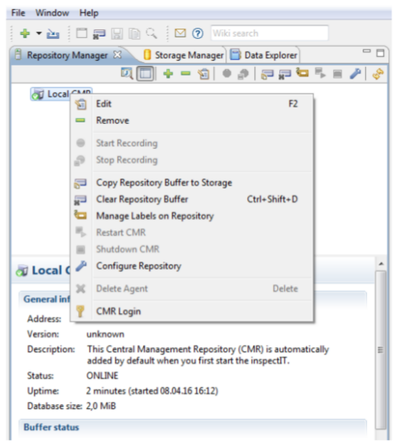



Das Sicherheitskonzept besteht aus mehreren Bereichen.

### 3.1 Datenbank und Verwaltung der User und Rollen

Neu in die Datenstruktur integriert wurden User, spezielle Berechtigungen und die Zusammenfassung dieser Berechtigungen zu Rollen, welche an User vergeben werden können. Im Detail musste die serverseitige H2 Datenbank umgestaltet und die Kommunikation über serialisierte Java-Objekte erweitert werden.

### 3.2 Visuelle Beschränkung




Dieser Bereich spielt sich auf dem Client, also dem User Interface, ab. Hier gibt es viele Möglichkeiten Operationen auszuführen oder Daten einzusehen, die nicht für jeden Benutzer gedacht sind. Zum Beispiel das Herunterfahren des CMR-Servers.
Um dem Benutzer zu zeigen, zu was er berechtigt ist, werden Optionen, die er nicht durchführen darf, ausgegraut und sind somit nicht mehr auswählbar. Außerdem werden Daten, die er nicht einsehen darf, vollständig ausgeblendet. Somit ist auf der Client-Seite gewährleistet, dass der Benutzer keine Buttons betätigen kann, deren Funktion ihm nicht gestattet ist. Außerdem werden beispielsweise Speicherorte, deren Inhalt er nicht einsehen darf, gar nicht erst angezeigt.
Praktisch gesehen nutzten wir die technischen Möglichkeiten der Eclipse Rich-Client-Platform, welche das User Interface Design erleichtern. Dafür war der vorsichtige Umgang mit vorhandenen Perspektiven und Bildelementen vonnöten.

### 3.3 Serverseitige Beschränkung

Trotz clientseitiger visueller Beschränkungen sollte auch auf dem CMR-Server geprüft werden, ob der Client berechtigt ist, gewisse Anfragen durchzuführen. Hierfür verwenden wir Apache Shiro, ein Open Source Java Security Framework, das unter anderem für Verwaltung von Authentifizierung, Autorisierung und Session Management zuständig ist.
Für die Kommunikation zwischen Client und CMR wird das Spring Framework verwendet, weshalb die Konfiguration von Shiro über Spring geschieht. Durch die korrekte Konfiguration von Shiro können nun sehr einfach Methoden auf dem CMR beschränkt werden. Dazu prüft Shiro, welcher Benutzer versucht, die Methode auszuführen und führt sie nur aus, wenn der Benutzer über die benötigten Berechtigungen verfügt.
Somit ist auch auf der Server-Seite gewährleistet, dass keine Methoden unberechtigt ausgeführt werden können, wie beispielsweise durch die Umgehung der clientseitigen visuellen Beschränkungen.

## 4 Organisation

Zu Beginn des Projekts haben wir uns in wöchentlichen Treffen hauptsächlich die Grundlagen der in inspectIT verwendeten Technologien angeeignet. Nachdem wir den umfangreichen Code erhalten haben, der zu Beginn des Projekts noch nicht Open Source war, begannen wir zunächst damit, uns einen groben Überblick darüber zu schaffen. Außerdem machten wir uns klar, wie die verschiedenen Technologien praktisch angewendet wurden.
Nun haben wir eine grobe Struktur erarbeitet, nach der wir im weiteren Projektverlauf das Authentifizierungs- und Autorisierungssystem entwickelt haben.
Zur Entwicklung haben wir unsere Aufgaben im Aufgabenmanagementsystem Jira als Tickets definiert und diese in mehreren Abschnitten bzw. Sprints abgearbeitet. Nach beendeten Sprints wurden diese im Team besprochen und anschließend der nächste Sprint geplant.
Zum Austausch im Team haben wir die Kommunikationsplattform Slack verwendet und regelmäßige Treffen abgehalten, um Aktuelles zu besprechen und zu planen. Unseren Code haben wir über den Online-Dienst GitHub verwaltet.

## 5 Schwierigkeiten und Erfahrungswerte

An dieser Stelle möchten wir einige persönliche Erfahrungen vermitteln, die vielleicht auch für zukünftige Projekt-Teilnehmer spannend sein dürften.
Im Grunde genommen wird einem Teilnehmer der Sinn des Projektes vor allem erst im Verlauf des Semesters klar: Es wird seltener nachgefragt, wie man vorankommt und es kommt darauf an, Verantwortung für den Verlauf des Projektes zu übernehmen. Es kommt auf jedes Teammitglied an. Natürlich bedarf es hier auch an Organisationstalent und Team-Management, um am Ende ein solch umfangreiches Projekt erfolgreich auf die Beine zu stellen.
Solltest Du auch davor stehen, ein größeres Industrie-Projekt anzugehen, dann bereite dich darauf vor, dass man sich ungefähr soviel selbstständig erarbeiten muss, wie man es sonst in einer Uni-Veranstaltung lernen würde. Die Anfänge in Java beispielsweise und "Enterprise" Java sind zwei unterschiedliche Welten. Viele Features bedeuten viel Leistungsoptimierung und den Einsatz von Plugins/Frameworks, die einen Mehrwert zur Anwendung beitragen.

## 6 Daten

Folgende Änderungen an dem GitHub-Repository entstanden im Laufe des Projekts durch unser Team:
Innerhalb von 215 Commits:
22.127 Codezeilen hinzugefügt und 15.426 Codezeilen entfernt. Die meisten Commits gab es in der zweiten Projekthälfte.