

Erweiterung SAP Commerce Backoffice um Groovy Konsole

IPA

Tabea Bolliger.

08.04.2024

Versionshistorie

Version	Datum	Autor	Beschreibung der Änderungen
1.0	18.03.2024	Tabea Bolliger	Struktur, Organisation & Verwaltung
2.0	19.03.2024	Tabea Bolliger	Diagramme, Konzepte
3.0	20.03.2024	Tabea Bolliger	Realisierung
4.0	25.03.2024	Tabea Bolliger	Realisierung
5.0	27.03.2024	Tabea Bolliger	Realisierung
6.0	28.03.2024	Tabea Bolliger	Realisierung
7.0	02.04.2024	Tabea Bolliger	Realisierung
8.0	03.04.2024	Tabea Bolliger	Dokumentation, Testing Vorbereitung, Fehler behoben
9.0	04.04.2024	Tabea Bolliger	Dokumentation, Testing
10.0	08.04.2024	Tabea Bolliger	Dokumentation

Freigabe

Version	Gültig ab	Freigabe durch	Bemerkungen
Final	08.04.2024	Autor	Finale Version der Dokumentation

Inhaltsverzeichnis

Organisation der Arbeitsergebnisse	5
Versionsverwaltung	6
Abbildungsverzeichnis	9
Tabellenverzeichnis	11
Umfeld	13
Projektaufbauorganisation	13
Deklaration der Firmenstandards	14
Zeitplan	15
Arbeitsjournal	17
Montag, 18.03.2024 (7h 30min)	17
Dienstag, 19.03.2024 (8h)	18
Mittwoch, 20.03.2024 (8h)	19
Montag, 25.03.2024 (8h 30min)	20
Mittwoch, 27.03.2024 (8h)	21
Donnerstag, 28.03.2024 (8h)	22
Dienstag, 02.04.2024 (8h)	23
Mittwoch, 03.04.2024 (8h)	24
Donnerstag, 04.04.2024 (8h)	25
Montag, 08.04.2024 (8h)	26
Projekt	28
Kurzfassung	28
Einleitung	29
Informieren	30
Aufgabenstellung und Vorgaben	30
Fragen	30
Planen	32
Realisierungskonzept	32
Systemdiagramm	33
Use Case	34
Aktivitätsdiagramm	36
Testkonzept	38
Logging-Konzept	41
Entscheiden	43
Lösungsweg	43
Realisieren	44
Projektumgebung	44
Extension und Perspektive	49
Groovy Konsole	54
ImpEx	61

Kontrollieren	62
Testprotokoll	62
Logging Protokoll	64
Logging Resultat	65
Auswerten	66
Reflexion	66
Glossar	67
Quellenverzeichnis	69
KI-Werkzeuge	71
Anhang	73
Benutzeranleitung	73
Quellcode	75

Organisation der Arbeitsergebnisse

Ich habe zur Sicherung meiner Arbeit das Projekt in einem GitLab Repository hochgeladen.

The screenshot shows a GitLab repository page for 'BackofficeExtensionIPA'. At the top, there's a brown header bar with the repository name and a 'Project ID: 4947'. Below it, a dark header bar displays '19 Commits', '1 Branch', '0 Tags', and '193 KiB Project Storage'. On the right, there are buttons for 'Star' (0), 'Forks' (0), and a clone link. The main content area shows a commit history with one entry: 'Code clean up' by 'Tabea Bolliger' just now, with a commit hash '1cbd365d'. Below the commit history are buttons for 'Add README', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Enable Auto DevOps', 'Add Kubernetes cluster', 'Set up CI/CD', and 'Add Wiki'. A 'Configure Integrations' button is also present. A table at the bottom lists a single file: 'MerkleExtension' with a 'Last commit' of 'Code clean up' by 'Tabea Bolliger' just now.

Abbildung 1: BackofficeExtensionIPA Repository

Um die Dokumentation und den Zeitplan täglich zu sichern habe ich eine Onlineversion auf unserem internen OneDrive erstellt.

The screenshot shows a OneDrive folder structure. The root folder is 'IPA-ErweiterungSAPCommerceBackoffice'. Inside, there are two subfolders: 'Dokumentation-Versionierung' and 'Zeitplan-Versionierung', both last modified on March 18 by 'Tabea Bolliger'. Each subfolder contains 9 items.

Abbildung 2: IPA Dokumentation und Zeitplan Versionierung

Versionsverwaltung

Dentsu OneDrive

OneDrive ist ein Cloud-Speicher die Daten auf einem entfernten Server speichert. Wenn ich meine IPA-Dokumentation und den Zeitplan also auf OneDrive hochlade, bedeutet dies, dass ich die Dokumente im Internet speichere. Das dient mir dann als weitere Absicherung, falls mein Laptop vielleicht abstürzt und meine lokalen Dokumente verloren gehen.



Abbildung 3: Neuer Ordner in der OneDrive

Damit das auch alles weiterhin strukturiert ist, habe ich im OneDrive unter «Meine Dateien» einen Ordner erstellt mit meinem IPA-Titel. Darin habe ich zwei weitere Ordner erstellt, einen für die Dokumentation und den zweiten für meinen Zeitplan. Ich lade dann täglich meine Dokumentation und den Zeitplan hoch, dies erleichtert dann auch die Übersicht von meinem Fortschritt und aktuellen Stand.

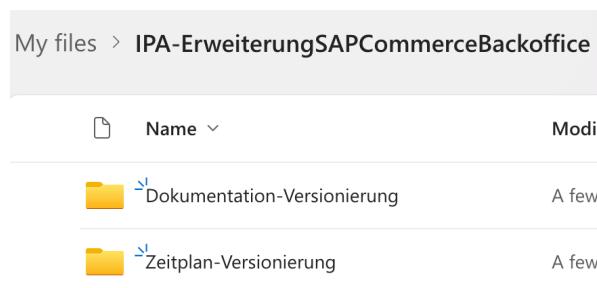


Abbildung 4: Erstellte Ordner

GitLab Repository

GitLab ist eine Webanwendung, welche zur Versionsverwaltung von Softwareprojekten dient. Sie basiert auf Git und ist eine Open-Source-Anwendung und eignet sich durch ihre Funktionen für Firmen und deren Projekte. Ich kann auf GitLab mein Projekt in ein so genanntes «Repository» hochladen.

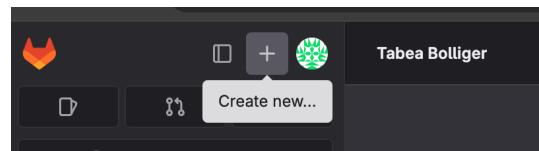


Abbildung 5 Neues Projekt in GitLab erstellen

Ich habe mich auf meinem GitLab angemeldet und kann mit dem Plus Symbol ein neues Projekt erstellen. Dieses werde ich dann «BackofficeExtensionIPA» nennen und speichere es auf meinem Benutzer «tbolliger» ab, damit es nicht in einer Gruppe von meiner Firma landet.

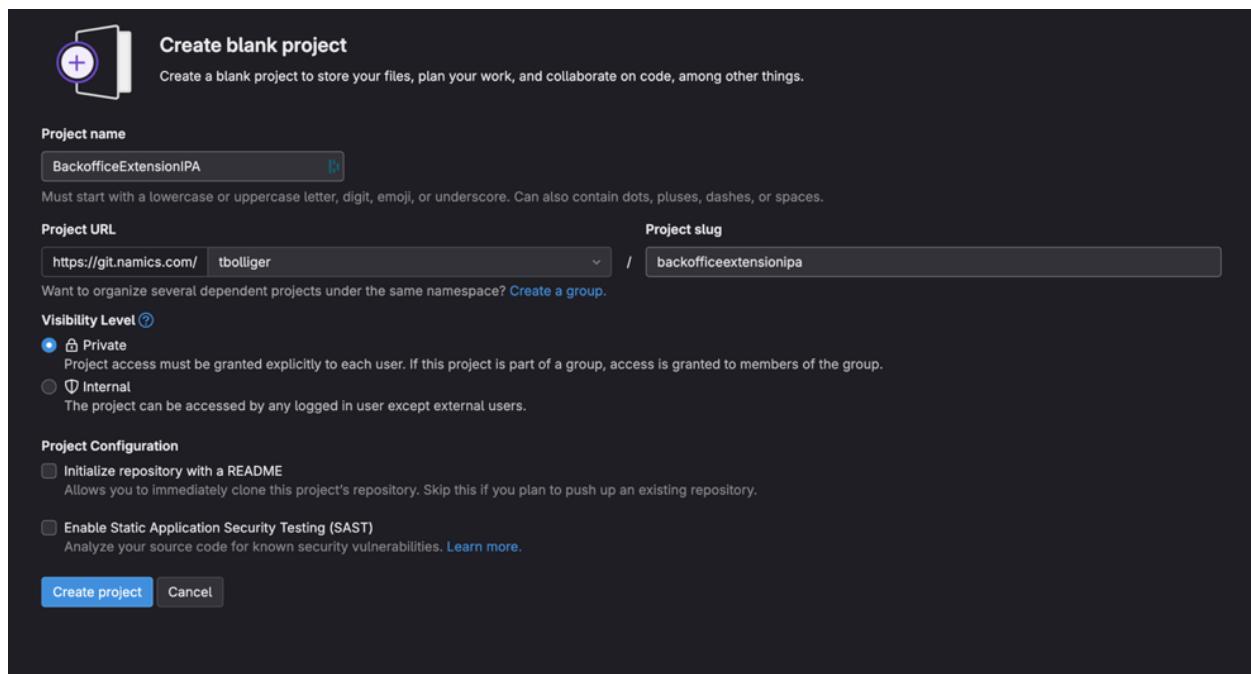


Abbildung 6: Konfigurationen des neuen Projekts

Ich wechsle dann in meine CLI und navigiere in den «custom» Ordner. Diesen Schritt kann ich erst durchführen, wenn ich die Extension bereits generiert habe. In dem «custom» Ordner wird nämlich der ganze selbstgeschriebene Code gespeichert, zusätzlich werde ich auch nur diesen Ordner in das GitLab hochladen da die maximale Grösse, für Datei Uploads, nur 50MB beträgt.

```
Last login: Wed Mar 20 10:13:00 on ttys001
tbolliger@CH-C02D1540MD6R:~/custom%
```

Abbildung 7: git Initialisierung

Damit ich das Repository von der CLI initialisieren kann brauche ich Git zuerst auf meinem Mac. Ich werde hierfür den Befehl «brew install git» von Homebrew, ein Werkzeug für einfache Installationen von UNIX-Werkzeugen. Mit dem Git Befehl «git init» kann ich dann ein neues Repository initialisieren. Den Befehl «git add .» ermöglicht mir alle Dateien in dem «custom» Ordner in das initialisierte Repository hinzuzufügen. Damit ich und auch andere Entwickler dann wissen, was ich genau am Code angepasst habe, werde ich mit dem Befehl «git commit -m» eine Nachricht zu meinen sogenannten commits dazugeben.

```
[tbolli01@CH-C02D1540MD6R custom % git add .
[git %
[tbolli01@CH-C02D1540MD6R custom % git commit -m "Arbeitspaket Extension: Neue Extension erstellt"]
```

Abbildung 8: git erster Commit

Ich muss ebenfalls sicherstellen, dass mein lokales Repository mit dem auf GitLab verbunden ist. Deshalb werde ich mit dem Befehl «git remote add origin» und der URL meines GitLab Repository mein zuvor erstelltes Projekt hinzufügen. Mit dem Befehl «git push -u -f origin master» kann ich dann meinen Commit von vorhin auch in das Repository im GitLab hochladen. Jedes Mal, wenn ich nun mein GitLab Projekt aktualisieren will muss ich zuerst mit «git add» meine Dateien hinzufügen, dann den Commit mit «git commit» tätigen und dort auch «-m» benutzen damit ich eine Nachricht hinterlassen kann und zu guter Letzt immer den «git push» Befehl ausführen damit auch andere meine Änderungen sehen können

```
[tbolli01@CH-C02D1540MD6R custom % git remote add origin https://git.namics.com/tbolliger/backofficeextensionipa
[tbolli01@CH-C02D1540MD6R custom % git push -u -f origin master]
```

Abbildung 9: git Verknüpfung und erster push

Abbildungsverzeichnis

Abbildung 1: BackofficeExtensionIPA Repository	5
Abbildung 2: IPA Dokumentation und Zeitplan Versionierung	5
Abbildung 3: Neuer Ordner in der OneDrive	6
Abbildung 4: Erstellte Ordner	6
Abbildung 5 Neues Projekt in GitLab erstellen	7
Abbildung 6: Konfigurationen des neuen Projekts	7
Abbildung 7: git Initialisierung	7
Abbildung 8: git erster Commit	8
Abbildung 9: git Verknüpfung und erster push	8
Abbildung 10: Zeitplan	16
Abbildung 11: Systemdiagramm	33
Abbildung 12: Anwendungsfallsdiagramm 1	34
Abbildung 13: Anwendungsfallsdiagramm 2	35
Abbildung 14: Aktivitätsdiagramm 1	36
Abbildung 15: Aktivitätsdiagramm 2	37
Abbildung 16: Ordner des Abschluss Projekts	44
Abbildung 17: SAP Commerce Versionen	44
Abbildung 18: ".zip" Datei der benutzten SAP Commerce Version	44
Abbildung 19: touch Befehl in der CLI	45
Abbildung 20: Inhalt des "setjavahome.sh" Skript	45
Abbildung 21: echo Befehl in der CLI	45
Abbildung 22: erneuter echo Befehl in der CLI	45
Abbildung 23: Navigation in den installer Ordner	46
Abbildung 24: Liste der verfügbaren Recipes	46
Abbildung 25: bin Verzeichnis	47
Abbildung 26: Projekt in IntelliJ IDEA importieren	47
Abbildung 27: Import Prozess	48
Abbildung 28: Struktur des Projekts in IntelliJ IDEA	48
Abbildung 29: Ausführung des "setantenv.sh" Skript	49
Abbildung 30: Neue Projektstruktur in der IDE	49
Abbildung 31: Backoffice Ansicht im Orchestrator Modus	50
Abbildung 32: MerkleExtension-backoffice-widgets.xml	51
Abbildung 33: Restriktionen der Perspektive im Orchestrator Modus setzen	51
Abbildung 34: cockpit-config.xml Datei im Orchestrator Modus zurücksetzen	52
Abbildung 35: Veraltetes ImpEx mit Benutzerberechtigung	52
Abbildung 36: perspective-switcher als admin	53
Abbildung 37: perspective-switcher als groovyAdmin	53
Abbildung 38: MerkleExtension-backoffice-config.xml	53
Abbildung 39: definition.xml	54

Abbildung 40: groovyKonsole.zul	55
Abbildung 41: Zusätzliche Konfigurationen in local.properties	55
Abbildung 42: Force Stop von Java Prozess im Activity Monitor	56
Abbildung 43: Registration von dem ScriptingLanguagesService Bean	57
Abbildung 44: executeScriptFromString Methode im Controller	58
Abbildung 45: groovyKonsole.zul	59
Abbildung 46: executeScriptFromFileUpload Methode im Controller	60
Abbildung 47: executeScript Methode im Controller	61
Abbildung 48: Finalisierte ImpEx Datei von Benutzer und Benutzergruppen	61
Abbildung 49: Anmeldefenster des Backoffice	73
Abbildung 50: Groovy Konsole in der Merkle Extension	73
Abbildung 51: Eingabefeld der Groovy Konsole	74
Abbildung 52: Pop-Up Fenster mit dem Resultat des Skripts	74
Abbildung 53: Datei die in der Konsole hochgeladen wurde	74

Tabellenverzeichnis

Tabelle 1: Arbeitsjurnaleintrag, Tag 1	17
Tabelle 2: Arbeitsjurnaleintrag, Tag 2	18
Tabelle 3: Arbeitsjurnaleintrag, Tag 3	19
Tabelle 4: Arbeitsjurnaleintrag, Tag 4	20
Tabelle 5: Arbeitsjurnaleintrag, Tag 5	21
Tabelle 6: Arbeitsjurnaleintrag, Tag 6	22
Tabelle 7: Arbeitsjurnaleintrag, Tag 7	23
Tabelle 8: Arbeitsjurnaleintrag, Tag 8	24
Tabelle 9: Arbeitsjurnaleintrag, Tag 9	25
Tabelle 10: <i>Arbeitsjurnaleintrag, Tag 10</i>	26
Tabelle 11: Angabe PC und Software	38
Tabelle 12: Testfall 1	39
Tabelle 13: Testfall 2	39
Tabelle 14: Testfall 3	39
Tabelle 15: Testfall 4	40
Tabelle 16: Testfall 5	40
Tabelle 17: Testfall 6	40
Tabelle 18: Loggingfall 1	41
Tabelle 19: Loggingfall 2	41
Tabelle 20: Loggingfall 3	41
Tabelle 21: Loggingfall 4	42
Tabelle 22: Loggingfall 5	42
Tabelle 23: Loggingfall 6	42
Tabelle 24: Loggingfall 7	42
Tabelle 25: Loggingfall 8	42
Tabelle 26: Testergebnisse	62
Tabelle 27: Loggingergebnisse	64
Tabelle 28: Glossar	67
Tabelle 29: Glossar	68

Teil A: Kriterien Fachkompetenz

Erweiterung SAP Commerce Backoffice um Groovy Konsole

Tabea Bolliger.

08.04.2024

Umfeld

Projektaufbauorganisation

Lehrbetrieb und Durchföhrungsart:

Merkle Switzerland AG
PULS 5 – Giessereihalle, Giessereistrasse 18
8005 Zürich
044 228 67 77

Kandidat:

Tabea Bolliger
Hofwisen 19
8627 Grüningen
077 467 89 14
tabea.bolliger@merkle.com

Berufsbildner/Lehrfirma:

Janes Thomas
Merkle Switzerland AG
PULS 5 – Giessereihalle, Giessereistrasse 18
8005 Zürich
janes.thomas@merkle.com
079 295 82 88

Verantwortliche Fachkraft:

Janes Thomas
Merkle Switzerland AG
janes.thomas@merkle.com
079 295 82 88

Hauptexperte:

Mauro Russo
Google Switzerland GmbH
0795055482
mauror@google.com

Deklaration der Firmenstandards

Dokumentation Vorlagen

In unserer Firma benutzen wir gewisse Vorlagen für Dokumente und Powerpoint Präsentation. In unserer Intranet Plattform „The Hub“ sind alle Vorlagen auf einer Seite eingetragen und können von Mitarbeiter aus der gesamten DACH-Region verwendet werden

Dentsu OneDrive

Die Dentsu OneDrive wird von allen Mitarbeitern genutzt, um ihre Dateien online abzulegen und abzusichern.

GitLab

In GitLab sind alle SAP-Gruppen und Instanzen der SAP Commerce Plattform unserer Firma gespeichert. Wir Entwickler benutzen das GitLab auch um unsere Projekte Online abzuspeichern und während unserer Arbeit zu aktualisieren.

IntelliJ IDEA

Die IntelliJ IDEA ist eine Entwicklungsumgebung. Lernende, bekommen zu Beginn ihrer Ausbildung eine 4-jährige Lizenz für diese IDE und arbeiten während der Lehre mit dieser Umgebung. Die IntelliJ IDEA wird von uns SAP Backend Entwickler genutzt.

Zeitplan

Folgende Termine sind für den Ablauf der IPA gültig:

Starttermin: 18.03.2024

Datum der Abgabe: 08.04.2024

Der folgende Zeitplan wurde mit den verschiedenen Arbeitspaketen erstellt. Darin ist der Soll-Ist-Vergleich ersichtlich. Der Zeitplan ist im 2-Stunden-Raster aufgebaut, Grün ist die Soll-Zeit, Rot die Ist-Zeit und Violett sind die Meilensteine. Die Blauen Zellen sind ebenfalls Ist-Zeit jedoch sind diese spezifisch für 1 Stunden Arbeitspakete. Die Meilensteine benutze ich nach jeder IPERKA-Phase.

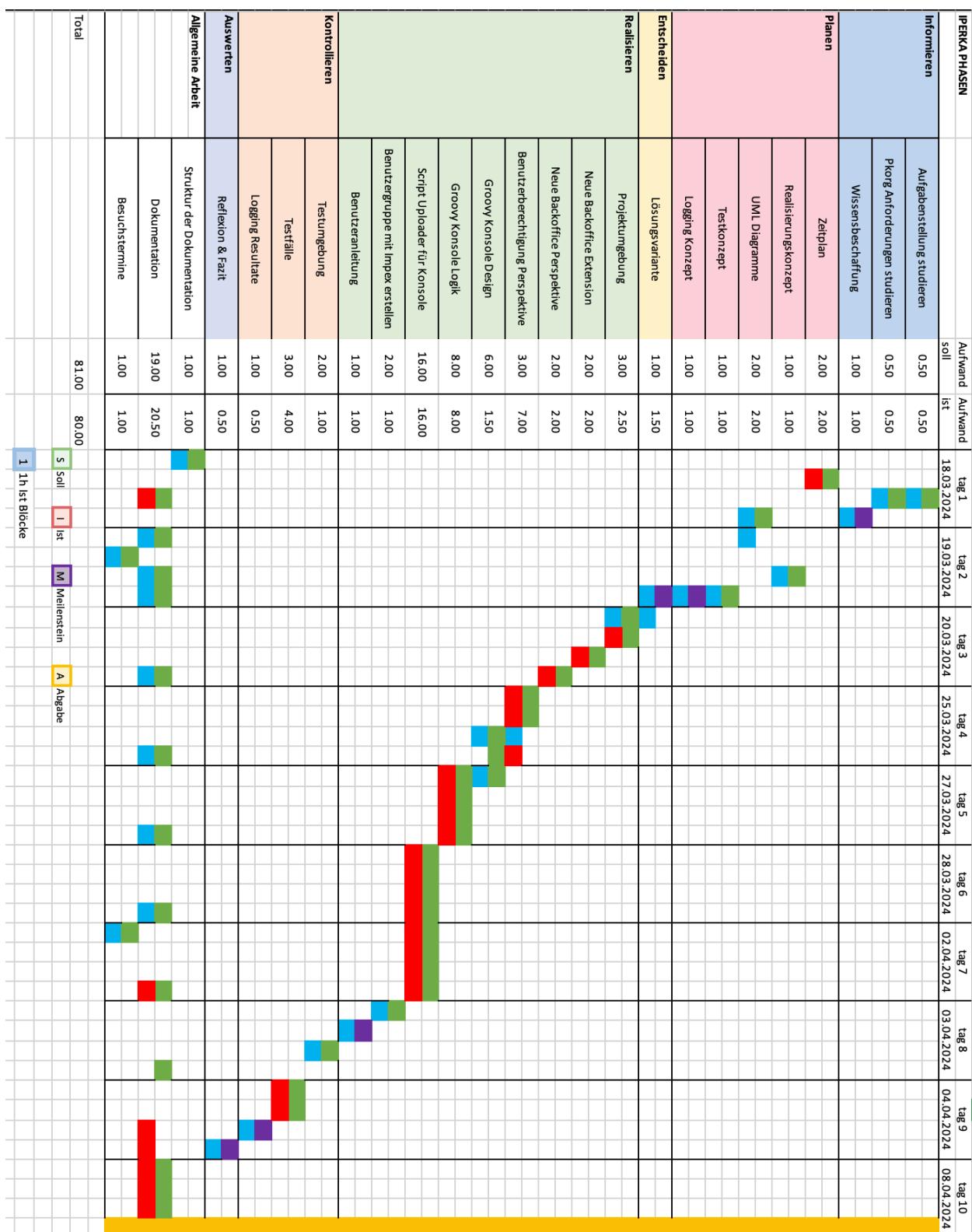


Abbildung 10: Zeitplan

Arbeitsjournal

Montag, 18.03.2024 (7h 30min)

Tätigkeiten	<ul style="list-style-type: none"> - Dokumentation Struktur - Zeitplan - Organisation & Verwaltung der Arbeitsergebnisse - UML Diagramme angefangen
Erreichte Ziele	<ul style="list-style-type: none"> - Dokumentation Struktur fertig - Zeitplan Struktur fertig
Probleme	Keine
Hilfestellung	Keine
Ausserplanmässige Arbeiten	Keine
Reflexion	<p>Ich hatte heute einen guten Start in die IPA. Ich konnte die Struktur der Dokumentation und des Zeitplanes anfangen und fertigstellen. Zudem habe ich schon einiges notiert für die verschiedenen Abschnitte in der Doku. Am Nachmittag habe ich dann meinen Fortschritt und den Zeitplan mit meinem Berufsbildner besprochen und mich auf den ersten Besuchstermin vorbereitet.</p>

Tabelle 1: Arbeitsjournaleintrag, Tag 1

Dienstag, 19.03.2024 (8h)

Tätigkeiten	<ul style="list-style-type: none"> - Zeitplan überarbeitet - UML Diagramme - Besuchstermin - Realisierungs-, Logging- und Testkonzepte - Lösungsvariante
Erreichte Ziele	<ul style="list-style-type: none"> - Zeitplan fertig überarbeitet - UML Diagramme fertig - Konzepte fertig
Probleme	Keine
Hilfestellung	Keine
Ausserplanmässige Arbeiten	Keine
Reflexion	<p>Ich habe heute morgen meinen Zeitplan nochmals überarbeitet. Glücklicherweise konnte ich die UML Diagramme zeitgemäß fertigstellen. Heute hatte ich auch noch den ersten Besuchstermin mit dem Hauptexperten, er gab mir viele Tipps und ich konnte mit ihm auch meine Dokumentation und den Zeitplan anschauen. Ich hatte gegen Ende des Tages etwas Mühe, ich hatte nicht genügend Schlaf und konnte mich nicht mehr so gut konzentrieren. Dadurch konnte ich meine Lösungsvariante nicht fertig schreiben. Ich habe jedoch meine Notizen festgehalten und werde diese Morgen in ganzen Sätzen fertig schreiben.</p>

Tabelle 2: Arbeitsjournaleintrag, Tag 2

Mittwoch, 20.03.2024 (8h)

Tätigkeiten	Projektumgebung aufsetzen Neue Backoffice Extension Neue Backoffice Perspektive
Erreichte Ziele	Projektumgebung aufsetzen Neue Backoffice Extension Neue Backoffice Perspektive
Probleme	Bei der Aufsetzung merkte ich, dass es die eine Instanz, die ich zuvor benutzen wollte, nicht mehr gab, ich habe dann meinen Berufsbildner gefragt, ob ich auch eine andere Version der Instanz benutzen konnte. Er überprüfte dann für mich, ob meine neu gewählte Instanz auch funktionieren würde und konnte mir bestätigen, diese neue Instanz zu benutzen.
Hilfestellung	Janes Thomas
Ausserplanmässige Arbeiten	Lösungsvariante fertig geschrieben Systemdiagramm Beschreibung
Reflexion	Heute morgen habe ich bemerkt, dass ich die Beschreibung des Systemdiagramm gestern vergessen habe. Die war eine Korrektur von wenigen Minuten, weshalb ich es nicht im Zeitplan nochmals eingetragen habe. Danach konnte ich auch noch meine Lösungsvariante fertig schreiben und mit dem Projekt fortfahren. Es gab hier fast keine Probleme und ich konnte alles sauber aufsetzen, meine Schritte habe ich mir Screenshots festgehalten und werde sie dann unter der Realisierungsphase dokumentieren. Danach habe ich die Extension und Perspektive erstellt, dies hat auch gut geklappt da ich im Internet auf dem Hilfsportal von SAP einige Infos bekam, welche ich dann ebenfalls in der Dokumentation erwähne.

Tabelle 3: Arbeitsjournaleneintrag, Tag 3

Montag, 25.03.2024 (8h 30min)

Tätigkeiten	Versionshistorie in der Dokumentation nachgetragen Zeitplan angepasst Berechtigungen Perspektive Design Konsole
Erreichte Ziele	Design der Konsole
Probleme	Backoffice Ansicht.
Hilfestellung	Janes Thomas
Ausserplanmässige Arbeiten	Versionshistorie von den vorherigen Tagen nachgetragen Zeitplan angepasst
Reflexion	Ich habe heute morgen noch die Versionshistorie in meiner Dokumentation aktualisiert und meinen Zeitplan angepasst. Danach habe ich mit der Benutzerberechtigung für die Perspektive angefangen und meine Benutzerrollen und Benutzer erstellt, glücklicherweise konnte ich daran konzentriert arbeiten. Ich merkte aber, dass dieses Arbeitspaket mich wesentlich länger beschäftigte als ich eingeplant habe. Gegen Ende des Tages ist dann auch die Backoffice Ansicht kaputt gegangen, ich habe daraufhin meinen Berufsbildner im Daily um Hilfe gefragt und wir versuchten das Problem zusammen zu lösen. Die genauen Schritte habe ich im Realisierungsabschnitt meiner Dokumentation erklärt

Tabelle 4: Arbeitsjournaleneintrag, Tag 4

Mittwoch, 27.03.2024 (8h)

Tätigkeiten	Berechtigung der Perspektive Groovy Konsole Design Groovy Konsole Skript Logik
Erreichte Ziele	Berechtigung der Perspektive Groovy Konsole Design
Probleme	Backoffice konnte nicht mehr aufgerufen werden.
Hilfestellung	James Thomas
Ausserplanmässige Arbeiten	Berechtigung der Perspektive noch fertig gemacht
Reflexion	Die Probleme, die ich am Montag hatte, haben mich am Morgen etwas demotiviert. Trotzdem habe ich weitergearbeitet und versuchte diese Probleme noch zu lösen. Mit Absprache mit meinem Berufsbildner habe ich dann aber trotz dieser Probleme mit dem nächsten Schritt weitergemacht, damit ich nicht viel Zeit verliere. Heute habe ich an der Logik für die Groovy Konsole gearbeitet und konnte das Design relativ schnell fertigstellen. Am Nachmittag konnte ich dann durch einen erneuten Fehler schon wieder nicht auf das Backoffice zugreifen, ich habe mit meinem Berufsbildner dann meinen Code und die Logs im Terminal untersucht. Er hat mir das Projekt nochmals sauber aufgestartet, ich konnte dann trotzdem nicht das Backoffice im Browser aufrufen. Wir haben dann im Activity Monitor geschaut, ob es Java Prozesse gab, welche nicht richtig beendet wurden. Diese Java Prozesse haben wir dann durch den Monitor beendet und ich konnte danach wieder auf das Backoffice zugreifen. In meiner Dokumentation habe ich auch noch festgehalten, wie wir dies im Activity Monitor gemacht haben.

Tabelle 5: Arbeitsjournaleintrag, Tag 5

Donnerstag, 28.03.2024 (8h)

Tätigkeiten	Groovy Konsole Logik Groovy Skript Upload Logik
Erreichte Ziele	Teil der Groovy Skript Upload Logik war erfolgreich
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	Backoffice Ansicht korrigiert, der Benutzer kann das Backoffice wieder normal sehen.
Reflexion	Heute hat mich der Fehler von Gestern immer noch genervt, deshalb habe ich meine Extension nochmals neu generiert. Dies hat nicht viel Zeit gebraucht, da ich ja schon wusste wie ich sie generieren musste und alle relevanten Dateien auf meinem GitLab Repository hatte. Ich habe dann einen Fehler bei meiner Benutzerberechtigungen bemerkt und diesen noch schnell korrigiert. Für das Hochladen meiner Skript Datei in der Groovy Konsole habe ich eine gute Vorlage auf dem ZKFiddle Forum gefunden und konnte somit Zeit sparen für einen Teil der Logik. Ich werde meine Schritte dazu genauer in dem Realisierungs Abschnitt meiner Dokumentation festhalten. Gegen Ende dieses Tages habe ich mich auf die Dokumentation konzentriert. Nach den Ostern werde ich mit meiner Logik in der Konsole weiterarbeiten.

Tabelle 6: Arbeitsjournaleintrag, Tag 6

Dienstag, 02.04.2024 (8h)

Tätigkeiten	Experten Besuch Skript Uploader für Groovy Konsole Dokumentation
Erreichte Ziele	Groovy Konsole Fehler behoben und abgeschlossen
Probleme	Debugging Modus hat nicht funktioniert ScriptingLanguagesService war null
Hilfestellung	Janes Thomas
Ausserplanmässige Arbeiten	Ich habe heute noch Fehler aus meiner Logik für das Groovy Skript Textfeld behoben.
Reflexion	Ich startete den Tag heute mit dem Expertenbesuch und habe dort dem HEX meinen aktuellen Stand gezeigt. Ich arbeitete weiterhin an meinem Code, Janes Thomas hat mir hier beim debugging und mit den Beans geholfen. Ich habe die genaue Problematik und Vorgehensweise wieder im Realisierungsteil der Dokumentation festgehalten. Ich merkte dann, dass ich den Skript Uploader sehr wahrscheinlich nicht abschliessen konnte, dies habe ich auch Janes Thomas mitgeteilt. Es nervt mich ein wenig, aber mir ist bewusst, dass ich mich auf die Dokumentation fokussieren muss. Ich habe, somit beschlossen heute die Logik in der Groovy Konsole abzuschliessen und morgen mit den nächsten Schritten zu beginnen. Damit sollte ich keinen Stress mit der Testphase haben und ich kann genügend Zeit in die Dokumentation selbst investieren.

Tabelle 7: Arbeitsjournaleintrag, Tag 7

Mittwoch, 03.04.2024 (8h)

Tätigkeiten	Benutzeranleitung Benutzergruppe mit Impex Testumgebung
Erreichte Ziele	Groovy Konsole fertig Benutzergruppe mit Impex Testklasse erstellt
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	Probleme der Groovy Konsole behoben
Reflexion	Ich habe heute einen Fehler in meinem Zeitplan entdeckt. Für die Testumgebung habe ich 2 Stunden eingeschätzt aber einen 4 Stunden Block eingetragen, diesen habe ich schnell behoben. Dann fuhr ich mit der Benutzeranleitung fort und mit dem Impex für die Benutzergruppe und den Benutzer. Dies konnte ich zügig abschliessen und hatte deshalb Zeit übrig, welche ich benutzte die Fehler von meiner Logik für die Groovy Konsole und den Script Uploader zu beheben. Mir ist es dann auch gelungen diese Fehler zu korrigieren, ich habe die Problematik und meine Vorgehensweise auch hier im Realisierungsteil meiner Dokumentation festgehalten. Morgen werde ich mich auf das Testing und weiterhin auf die Dokumentation fokussieren.

Tabelle 8: Arbeitsjournaleintrag, Tag 8

Donnerstag, 04.04.2024 (8h)

Tätigkeiten	Testing Dokumentation
Erreichte Ziele	Testprotokoll Loggingprotokoll
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Ich habe für das Testing länger gebraucht als ich eingeplant habe, jedoch konnte ich mir genügend Notizen über die Testfälle schreiben und werde diesen am Montag in der Dokumentation festhalten. Da ich zu viel Zeit mit meinem unvollständigen Code verbracht habe, bin ich heute ziemlich in den Stress über meine Dokumentation gekommen. Zum Glück habe ich aber während der IPA meinen Fortschritt in Stichworten festgehalten, dies wird mir bei dem Formulieren der Sätze helfen.

Tabelle 9: Arbeitsjournaleintrag, Tag 9

Montag, 08.04.2024 (8h)

Tätigkeiten	Dokumentation
Erreichte Ziele	Dokumentation abgeschlossen IPA abgegeben
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Ich kam heute noch etwas in den Stress, da Heute die Abgabe der Arbeit war. Ich konnte die Dokumentation frühzeitig abschliessen und habe meine Kollegen gefragt, ob sie das Dokument durchlesen könnten. Dadurch bekam ich schon etwas Feedback und konnte meine Texte ausbessern.

 Tabelle 10: *Arbeitsjournaleneintrag, Tag 10*

Teil B: Dokumentation

Erweiterung SAP Commerce Backoffice um Groovy Konsole

Tabea Bolliger.

08.04.2024

Projekt

Kurzfassung

Die Kurzfassung der Probe IPA vermittelt dem Leser einen Überblick der Arbeit. Zudem dient es zur Erleichterung, um den Inhalt des Projekts leichter verständlich zu machen.

Ausgangssituation

In dieser IPA ging es darum für das Backoffice der SAP Commerce Plattform eine neue Modul Erweiterung, eine sogenannte Extension, zu entwickeln. Diese Extension muss dann durch eine Ansicht, auch bekannt als Perspektive, erweitert werden. Die Perspektive, soll es Entwicklern ermöglichen die Groovy Konsole aus der HAC auch in dem Backoffice zu benutzen. Es sollte die Arbeit erleichtern, da man dadurch nicht immer von einem Tool in das andere wechseln und navigieren muss. Die Gestaltung der Konsole in der Backoffice Ansicht sollte ähnlich wie die Konsole in der HAC sein. Der Entwickler kann in einem Textfeld sein Skript eintippen oder eine Datei hochladen und ausführen. Die Konsole, welche sich im Backoffice befindet, soll zusätzlich mit einer Skript-Upload Funktion ergänzt werden. Die «groovyAdmin» Benutzer sind die einzigen Mitarbeiter, welche diese Groovy Konsole Perspektive zusätzlich zu den anderen Ansichten im Backoffice jeweils sehen können, da sie Teil der Gruppe «groovybackofficemanagergroup» sind. Diese Benutzer Konfiguration wird dann in einer ImpEx gespeichert und kann auch von anderen Entwicklern in der HAC importiert werden.

Umsetzung

Die neue Extension wird durch «ant extgen» generiert. Die neue Extension soll auf dem «ybackoffice» Modul von SAP basieren. Im Backoffice wurde dann mit Hilfe des Orchestrator Modus die Perspektive hinzugefügt. Die Konfigurationen, welche ich im Orchestrator Modus betätigt habe, werde ich auch in bereits generierten Konfigurationsdateien ergänzen. Das Widget «perspective-view-switcher» wird in der «MerkleExtension-backoffice-config.xml» Datei bearbeitet, hier wird die auch die Benutzerberechtigung der Benutzergruppe benutzt. In der «MerkleExtension-backoffice-widgets.xml» Datei wird die Konfiguration der Perspektive hinzugefügt und die Zugangsrechte der Gruppen Autorität ergänzt. Die Gestaltung der Konsole wird durch weitere Widgets gemacht. Diese muss ich in den Dateien «definition.xml» definieren und in «groovyKonsole.zul» visuell bearbeiten. In den Java Klassen wurden für die Logik der Eingabe, Uploads und Ausführung der Skripte erstellt. Der Benutzer sollte das Resultat der ausgeführten Skripte in einem weiteren Feld unterhalb der Eingabe und Skript-Uploads sehen, allfällige Benachrichtigungen sollten ebenfalls in der Backoffice Ansicht sichtbar sein.

Ergebnis

Die Extension konnte erfolgreich, auf Basis von dem «ybackoffice» Modul, durch «ant extgen» generiert werden. Die Ansicht konnte dann im Orchestrator Modus bearbeitet werden und die Konfigurationen wurde den entsprechenden Dateien hinzugefügt. Die Textfelder Skript-Upload Felder sind ebenfalls vorhanden. Die Benutzerberechtigungen wurde korrekt erstellt und in einer ImpEx Datei gespeichert. Jedoch kann der «groovyAdmin» nur die Groovy Konsole sehen und keine weiteren Ansichten. Die Meldungen und Resultate wurde jeweils in einer «MessageBox» gespeichert und im Backoffice als ein Pop-Up Fenster abgebildet. Die Ausführung von Skripten durch Eingabe im Textfeld, funktioniert ohne Probleme. Entwickler können alle möglichen Dateien in die Konsole hochladen, jedoch ist die Ausführung der Dateien fehlerhaft und kann nicht betätig werden.

Einleitung

Das Projekt ist nach der IPERKA-Methode aufgebaut. In IPERKA sind die folgenden 6 Schritte oder auch, „Phasen“ genannt, notwendig um die Methode effektiv umzusetzen. Die Titelfarben der verschiedenen IPERKA Phasen sind in der gleichen Farbe eingefärbt wie im Zeitplan.¹

Informieren:

Der Auftrag ist so gut wie möglich geklärt und allfällige Fragen sind geklärt. Diese Phase ist wichtig, um einen möglichst akkuraten Zeitplan zu erstellen und zum Verständnis was überhaupt erwartet wird.

Planen:

Um einen guten und strukturierten Projektablauf zu garantieren, muss die Arbeit präzise geplant werden. Es werden Lösungswege, Testkonzepte und Realisierungskonzepte erstellt.

Entscheiden:

Die Lösungswege werden miteinander verglichen und einer wird ausgewählt. Man muss sich hierbei fragen, ob die Idee sinnvoll ist und man diese auch umsetzen kann.

Realisieren:

Erst nachdem alles geplant und entschieden worden ist, beginnt man mit der Realisierung des Projektes. Die Abläufe und Fortschritte werden regelmässig protokolliert und die Ist-Zeiten werden im Zeitplan eingetragen.

Kontrollieren:

Die Testdaten werden so getestet, wie es im Testkonzept geplant wurde und auch erst nach dem Abschliessen der Entwicklung. Fehlgeschlagene Testfälle werden schnellstmöglich korrigiert oder in der Dokumentation als Fehler vermerkt.

Auswerten:

Abschliessend wird eine Reflexion erstellt, in welcher man seine Erkenntnisse während der Arbeit berichtet. Man geht wieder durch alle Schritte, während dem Projekt durch und analysiert, wo man gut oder schlechter war und wo man Verbesserungen machen könnte. Eine Reflexion ist auch sehr praktisch für zukünftige Arbeiten.

¹ Bexio.com: IPERKA Methode

Informieren

Das Informieren ist die erste Phase der IPERKA-Methode

Es ist wichtig in dieser Phase möglichst viele Fragen über die Aufgabenstellung zu klären, damit man in der Planungsphase alles reibungslos einplanen kann.

Aufgabenstellung und Vorgaben

Die Verantwortliche Fachkraft muss vor Beginn der IPA eine detaillierte Aufgabenstellung der gesamten Arbeit im PkOrg deklarieren. Ich habe für die Aufgabenstellung den Text aus PkOrg kopiert und hier hinzugefügt

1. Es muss ein neues Backoffice Modul generiert werden, in welchem die Anpassungen durchgeführt werden. Die Namensgebung ist frei zu wählen.
2. Es muss eine neue Benutzergruppe innerhalb der SAP Commerce Instanz erstellt werden ("groovyAdmin"). Diese Benutzergruppe muss via ImpEx importiert werden können.
3. Das Backoffice Modul wird um eine neue Perspektive erweitert. Diese ist im Perspektiven-Dropdown ersichtlich. Nur die korrekte Benutzergruppe erhält Zugriff auf diese Perspektive.
4. Diese Perspektive enthält folgende Elemente:
 - Eine Textbox, in welche Groovy Skripte eingegeben werden.
 - Ein Button, mit welchem eingegebene Skripte ausgeführt werden.
 - Ein Feld, in welchem lokale Skripte hochgeladen werden können. Es soll keine Restriktionen der Art der Dateien geben.
 - Ein weiterer Button, welcher das hochgeladene Skript ausführt.
 - Ein Resultatsbereich unterhalb der beiden oberen Elemente, in welchem die Ergebnisse der Skripte angezeigt werden.
5. Der Benutzer muss informiert werden, wenn ein Skript durch einen inhaltlichen Fehler nicht ausgeführt werden kann. Es muss unterschieden werden zwischen einem Skript, welches ein leeres Resultat hat und eines, welches nicht ausgeführt werden kann.
6. Der Benutzer muss informiert werden, wenn entweder ein leeres Skript im Textfeld oder als Datei ausgeführt wird. Eine entsprechende Warnung soll angezeigt werden, die den Benutzer darüber informiert, dass das Feld leer ist / keine Datei vorhanden ist. Der Inhalt der Info Benachrichtigungen und das Design der Perspektivenelemente ist frei zu wählen.

Die Entwicklung basiert auf der Programmiersprache Java. Als IDE soll IntelliJ IDEA verwendet werden.

Fragen

Während den ersten Tagen habe ich mir alles Fragen aufgeschrieben, die mir in den Sinn gekommen sind und auf welche ich eine Antwort brauchte. Ich konnte diese Fragen dann am zweiten Tag, während des Besuchstermins mit dem Experten, stellen und dokumentierte seine Antworten.

Tabea Bolliger, KAND: Ist die Struktur der Dokumentation in Ordnung?

Mauro Russo, HEX: Ja, du darfst auch Inspirationen oder Vorlagen von anderen Personen benutzen, solange du diese dann auch in der Dokumentation referenzierst.

Tabea Bolliger, KAND: Ist die Struktur des Zeitplanes in Ordnung?

Mauro Russo, HEX: Ja die ist in Ordnung.

Tabea Bolliger, KAND: Ist die Struktur des Arbeitsjournals in Ordnung?

Mauro Russo, HEX: Ja. Hier ist es wichtig, dass du alle wichtigen Punkte aus dem Leitfaden mit dabei hast. Am Ende der IPA kannst du dann auch schauen, ob du zwei Arbeitsjournaleinträge eventuell auf einer gleichen Seite haben kannst.

Tabea Bolliger, KAND: Muss ich die Aufgabenstellung aus dem PkOrg auch in meiner Dokumentation haben, oder reicht es, wenn ich es in meinen eigenen Worten zusammenfassen?

Mauro Russo, HEX: Das ist dir überlassen. Wenn es nicht zu viel Aufwand ist, kannst du das zusammenfassen, ansonsten kannst du die Aufgabenstellung auf in die Dokumentation hineinkopieren.

Tabea Bolliger, KAND: Muss das Quellenverzeichnis auch alphabetisch sein?

Mauro Russo, HEX: Das ist auch wieder dir überlassen. Wichtig ist es, dass du eine klare Struktur hast und auch alle Quellen direkt referenzierst.

Tabea Bolliger, KAND: Dürfen KI-Tools benutzt werden, um bspw. selbstgeschriebene Texte grammatisch zu überprüfen oder um englische Texte zu übersetzen?

Mauro Russo, HEX: Das ist erlaubt, du musst es einfach in der Dokumentation deklarieren und festhalten.

Tabea Bolliger, KAND: Muss ich die Symbole in den UML-Diagrammen auch erklären oder genügt es einfach das Diagramm selbst zu beschreiben?

Mauro Russo, HEX: Du musst nicht, wichtig ist die Beschreibung der Diagramme, aber wenn du allenfalls ungewöhnliche Symbole benutzt, kann eine Erklärung schon gut sein.

Tabea Bolliger, KAND: Ist die Projektaufbauorganisation in Ordnung? Soll ich noch die Abkürzungen der Personen aufschreiben?

Mauro Russo, HEX: Die Abkürzungen brauchst du nicht aufzuschreiben.

Tabea Bolliger, KAND: Bei Fällen von Beerdigungen, brauche ich dafür auch ein Zeugnis?

Mauro Russo, HEX: Nein. Es wäre gut, wenn du uns den genauen Termin sagen könntest, damit wir Experten dann schauen können, ob wir die IPA-Tage verschieben müssten.

Planen

Realisierungskonzept

Für diese Arbeit werde ich eine Demo Instanz der SAP Commerce Plattform installieren, diese kann ich aus einem GitLab Repository von unserer SAP-Gruppe aus der Firma herunterladen und aufsetzen. Das Projekt wird auf meinem lokalen Gerät sein, aber um die Daten zu sichern und einer meiner individuellen Bewertungskriterien zu erfüllen, werde ich das Projekt in einem neuen GitLab Repository hinzufügen. Das Repository wird dann während meiner Programmertage täglich mit sogenannten «commits» aktualisiert.

Sobald das Projekt aufgesetzt wurde, kann ich auch schon mit dem Befehl «ant extgen» meine Extension für das SAP Commerce Backoffice generieren lassen. Nachdem die Extension erstellt wurde, füge ich meine eigene Perspektive hinzu. Hier muss ich sicherstellen, dass auch nur die bestimmte Benutzergruppe «groovyAdmin» Zugriff auf die Perspektive hat. Dies bedeutet, dass ich zuvor auch schon diese Gruppe erstellen muss. Dies kann ich entweder im SAP Commerce Backoffice manuell machen oder durch eine ImpEx Datei die Daten in das SAP Commerce Backoffice importieren. Die «groovyAdmin» Benutzergruppe ist, wie vorhin schon erwähnt, die einzige Gruppe, welche die neue Perspektive sehen und benutzen kann. Dafür muss ich sicherstellen, dass in der Backoffice Ansicht die Perspektive, im Dropdown, auch nur angezeigt wird, wenn der angemeldete Benutzer ein «groovyAdmin» ist. Der «groovyAdmin» muss auf diese Perspektive zugreifen können und die Groovy Konsole soll ersichtlich sein. Diese Konsole soll die gleiche Funktion haben, wie diese aus der SAP Commerce «Hybris Administration Console». Mit Hilfe von sogenannten Widgets kann ich diese Konsole im Backoffice nachbauen. Hier wird mir auch der Orchestrator Modus aus dem Backoffice helfen. Die Widgets kann ich dann in meinen Projektdateien mit ZUL-Dateien verändern und damit meine Änderung aus dem Backoffice auch nicht verloren gehen muss ich meine eigene «cockpit-config.xml» Datei erstellen. Mein gesamtes Vorgehen werde ich dann in der Realisierungsphase besser und genauer erklären.

Meine eigene Konsole wird ein Textfeld beinhalten, dadurch können Entwickler ihre eigenen Skripte schreiben und mit einem Knopfclick dieses eingegebene Skript ausführen. Wenn es hier zu Komplikationen führt, wird der Entwickler darauf hingewiesen indem in der CLI eine LOG-Warnung erscheint. Benutzer können auch von ihrem lokalen Gerät eine Datei hochladen, es wird hier keine Restriktionen geben, jedoch wird ein Fehler auftreten, wenn die hochgeladene Datei kein Skript ist. Unterhalb des Textfeldes und dem Feld für die hochgeladenen Dateien werde ich dann ein weiteres Feld für die Resultate der Skripte hinzufügen.

In der IDE werde ich dann die gesamte Logik der Konsole in einer Java Klasse hinzufügen, meine ZUL-Dateien werden ebenfalls in der IDE bearbeitet. Hier ist es wichtig, dass ich das Muster von MVC einhalte und meine Klassen in die korrekten Ordner einteile. Wie oben erwähnt, werde ich die Benutzergruppe in einer ImpEx Datei erstellen, damit auch andere Entwickler diese Benutzergruppe in ihre SAP Commerce Instanz importieren können.

Zuletzt werde ich noch eine Benutzeranleitung erstellen und darin erklären wie man als «groovyAdmin» agieren kann und wie man die Groovy Konsole benutzt. Diese Benutzeranleitung wird dann im Anhang hinzugefügt.

Systemdiagramm

Die Entwickler sind in der «groovyAdmin» Benutzergruppe. Diese können die SAP Commerce Plattform Instanz. Sie melden sich dann im Backoffice der SAP Commerce Plattform an. Da die Entwickler die Berechtigungen der «groovyAdmin» Gruppe haben, können sie die Perspektive der Groovy Konsole sehen und darauf zugreifen.

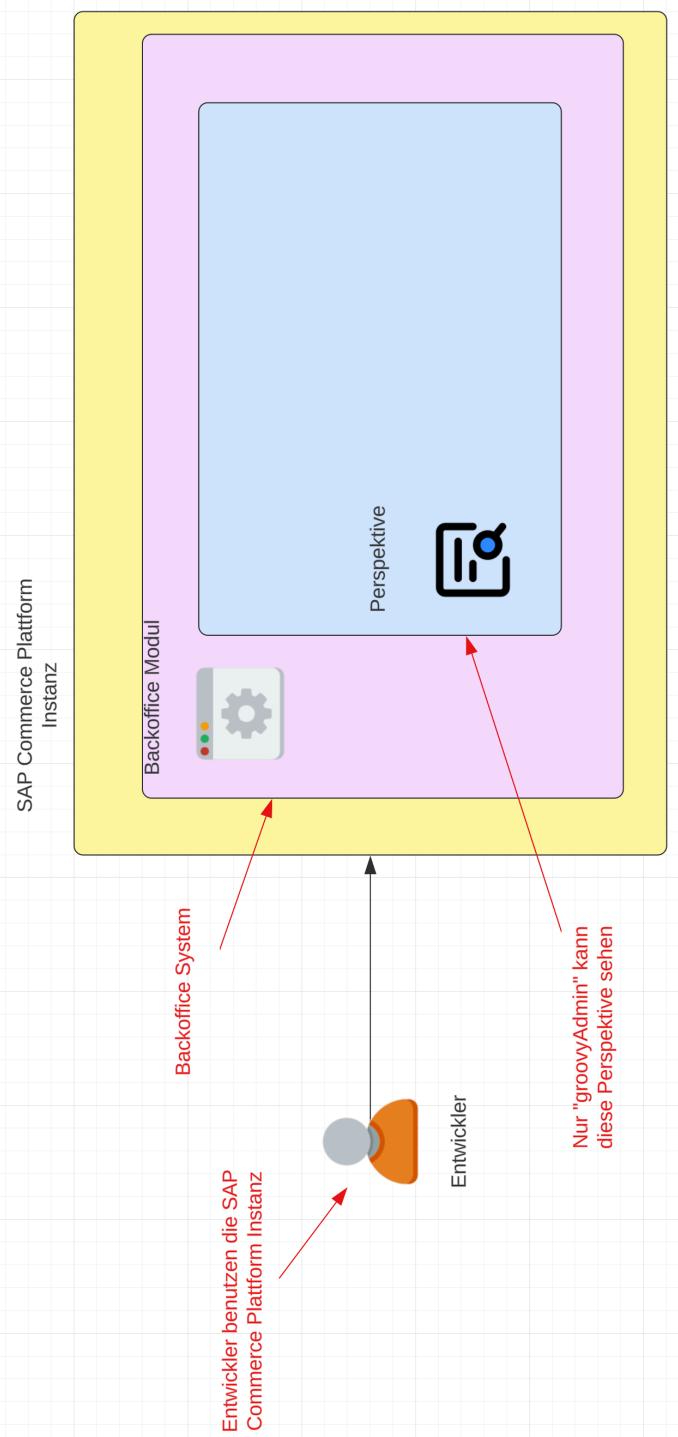


Abbildung 11: Systemdiagramm

Use Case

Der Benutzer meldet sich nach Start der SAP Commerce Plattform im Backoffice an, er ist bereits in der Benutzergruppe «groovyAdmin». Nachdem sich der Benutzer angemeldet hat, kann er im Backoffice durch ein Dropdown die verschiedenen Perspektiven sehen und diese auswählen.

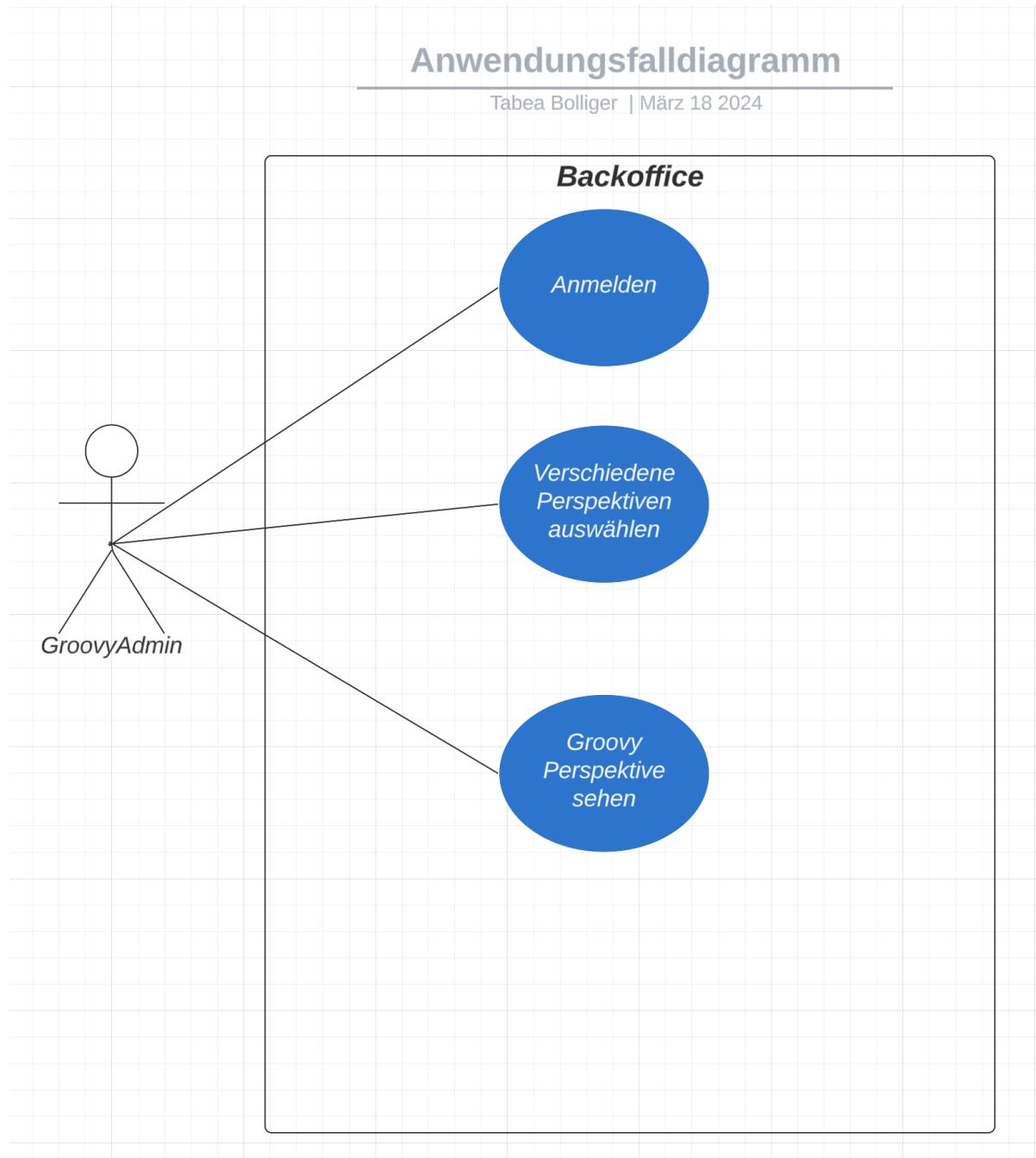


Abbildung 12: Anwendungsfalldiagramm 1

Der angemeldete «groovyAdmin» kann die Groovy Perspektive auswählen. Wenn die Perspektive gewechselt wurde, kann der Entwickler das Layout der Groovy Konsole sehen und auswählen, ob er ein Skript selbst schreiben möchte oder ob er von seinem lokalen Gerät eine Datei hochladen möchte. Der Entwickler hat Knöpfe zur Verfügung, mit welchen er die Skripte ausführen kann.

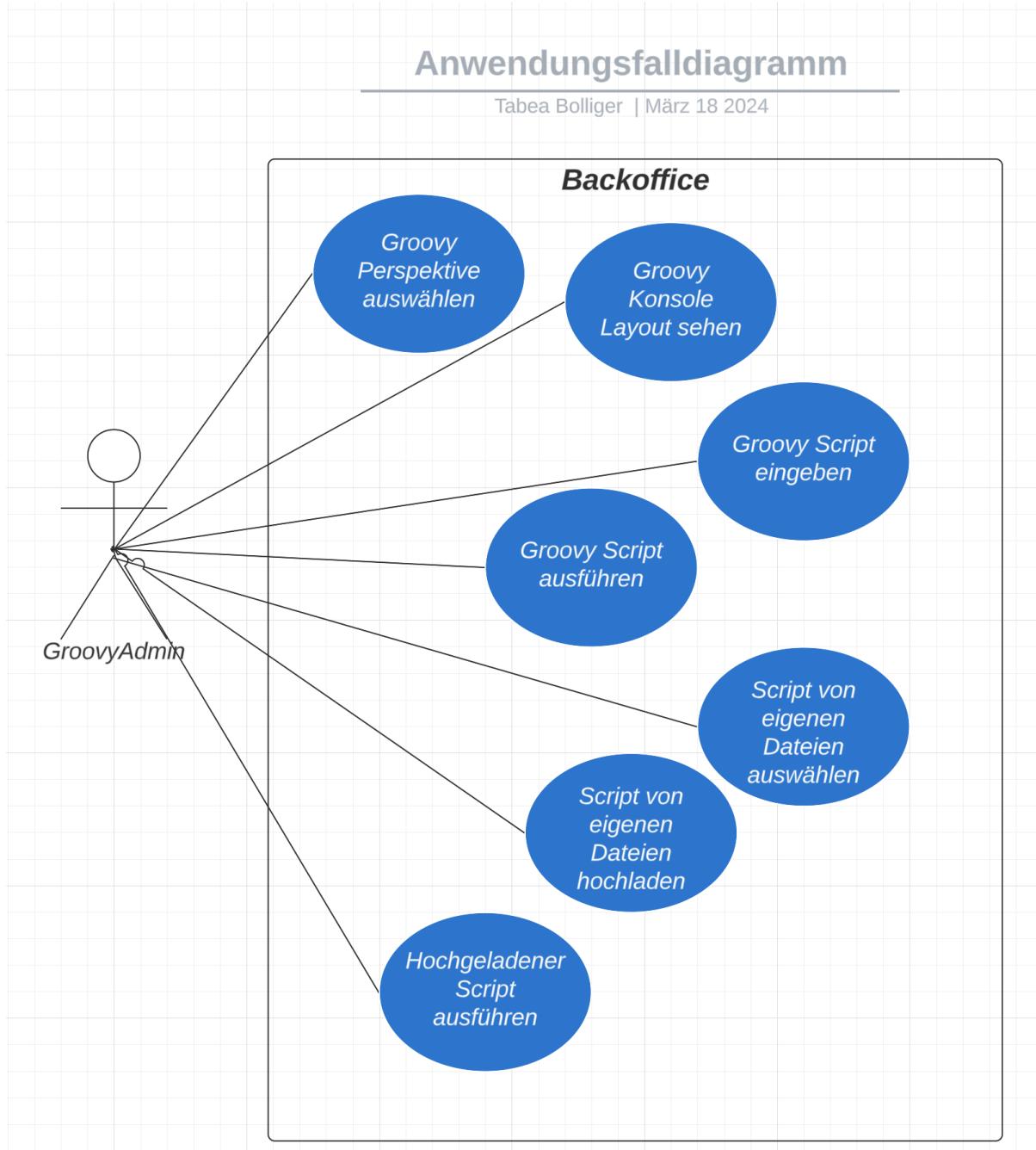


Abbildung 13: Anwendungsfalldiagramm 2

Aktivitätsdiagramm

Der Benutzer meldet sich im Backoffice an, seine Eingaben werden überprüft, falls er kein «groovyAdmin» ist, so wird er trotzdem ins Backoffice angemeldet, kann aber die Groovy Perspektive nicht sehen. Wenn er in der Benutzergruppe ist, wird er angemeldet und kann die Perspektive sehen und benutzen.

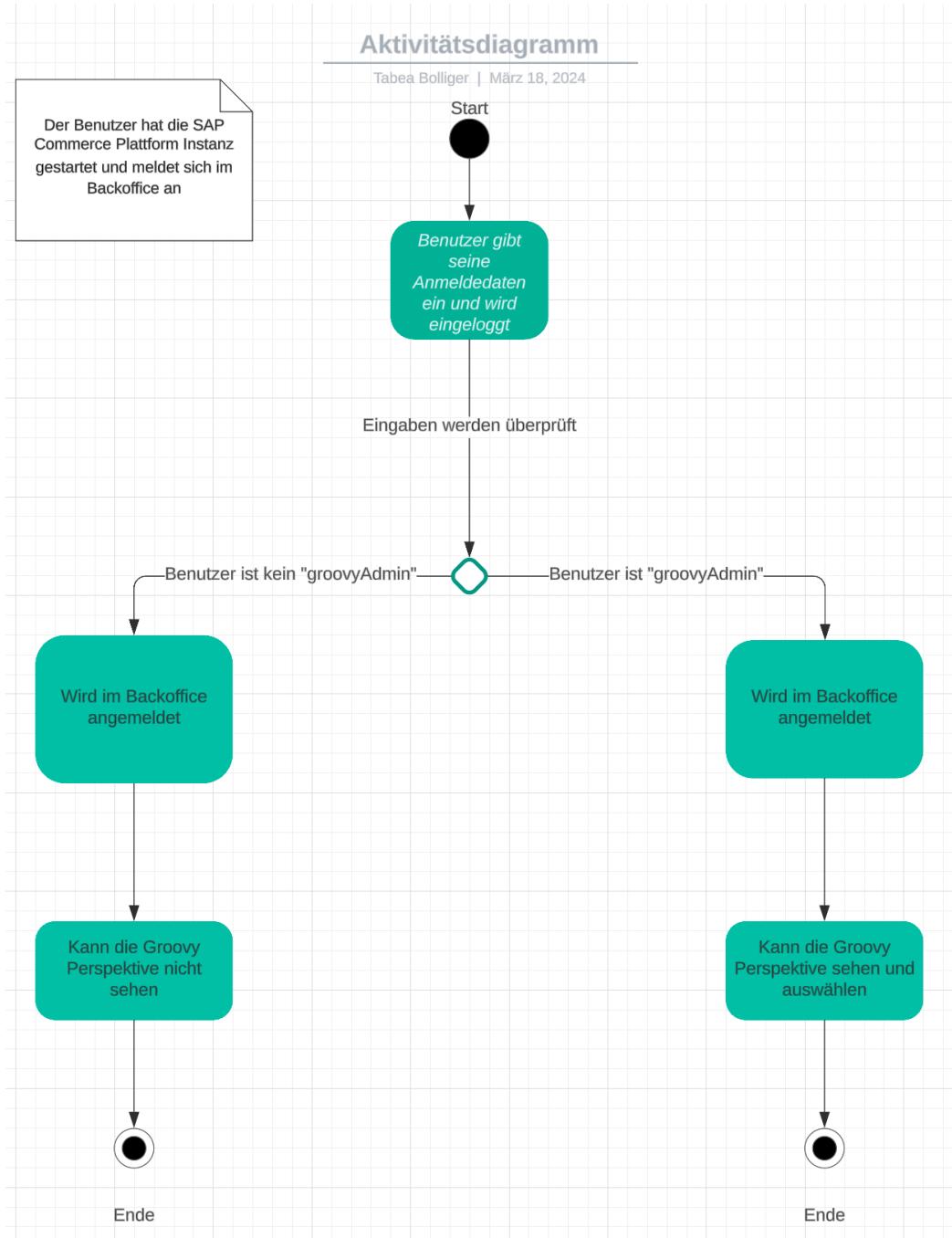


Abbildung 14: Aktivitätsdiagramm 1

Das Anmeldeverfahren hat bereits stattgefunden, der Benutzer ist ein «groovyAdmin». Nach dem der Entwickler die Perspektive gewechselt hat kann er sein Skript in das Textfeld eingeben und mit einem Knopfdruck sein Skript ausführen. Falls seine Eingaben fehlerhaft oder leer sind, wird der Benutzer darauf aufmerksam gemacht. Wenn die Eingaben gültig sind, wird das Skript ausgeführt und das Resultat wird in einem separaten Feld aufgezeigt. Der Benutzer kann auch eine Datei von seinem lokalen Gerät hochladen, hierbei gibt es keine Restriktionen. Möchte der Entwickler jedoch eine Datei ausführen, welche kein Skript ist, wird er ebenfalls darauf aufmerksam gemacht und die Datei wird nicht ausgeführt. Wenn die Datei ein Skript ist, wird dies überprüft und ausgeführt. Bei fehlerhaftem oder leerem Skript wird dem Benutzer ein leeres Resultat und eine Warnung zurückgegeben, bei korrektem Skript bekommt der Entwickler sein Resultat in einem Feld.

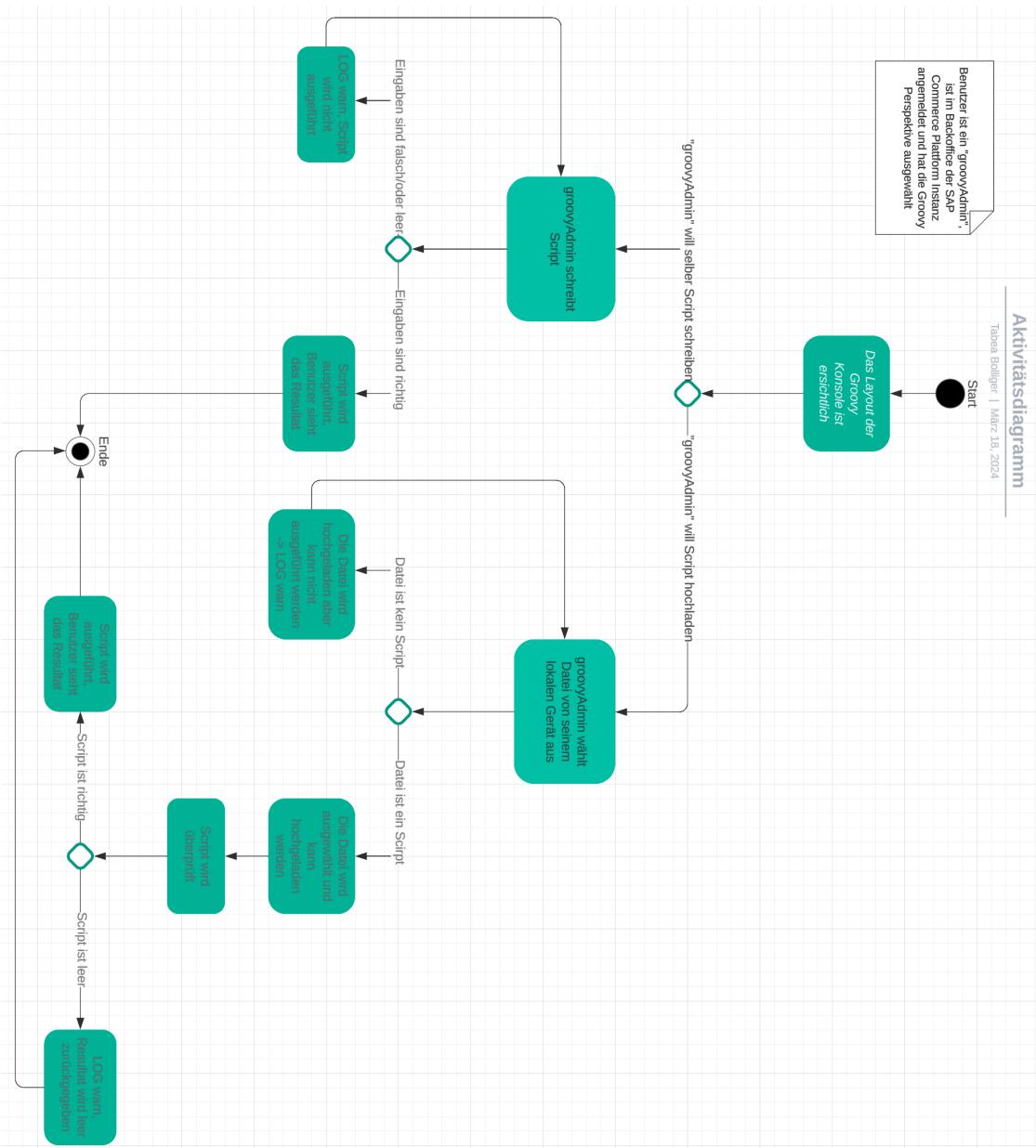


Abbildung 15: Aktivitätsdiagramm 2

Testkonzept

Im Testing gibt es verschiedene Arten oder Stufen, dazu hier eine Auflistung:²

- Unit Tests
 - Sehr nah an der Quelle, gut für einzelne Funktionen
- Integrations Tests
 - Integrierte Komponente testen
- System Tests
 - Das gesamte System wird getestet
- Abnahmetests
 - Das finale Produkt wird getestet

Angabe PC und Software

Name	MacBook Pro
Prozessor	2.4 GHz 8-Core Intel Core i9
Ram	16 GB
Testumfeld	HAC / Backoffice
Betriebssystem	Ventura 13.6.1

Tabelle 11: Angabe PC und Software

² [Richard Seidl: Teststufen](#)

Testfall	Nr.1
Anforderung:	Der Benutzer kann von der Standard Backoffice Ansicht in die Groovy Perspektive wechseln
Beschreibung:	Perspektive kann gewechselt werden
Voraussetzung:	Der Benutzer muss in der «groovyAdmin» Gruppe sein
Eingabe:	Benutzer wählt die Perspektive von dem Dropdown aus
Erwartetes Resultat:	Es wurde erfolgreich in die Perspektive gewechselt

Tabelle 12: Testfall 1

Testfall	Nr.2
Anforderung:	Der Benutzer kann in der Groovy Perspektive die Konsole sehen und in dem Textfeld sein Skript eingeben
Beschreibung:	Skript Eingabe in der Groovy Konsole
Voraussetzung:	Der Benutzer ist in der richtigen Perspektive und hat die Benutzerrechte
Eingabe:	Benutzer gibt sein Skript in das Textfeld ein
Erwartetes Resultat:	Das Skript konnte in das Feld eingetippt werden

Tabelle 13: Testfall 2

Testfall	Nr.3
Anforderung:	Der Benutzer kann in der Groovy Konsole das eingegeben Skript mit einem Knopf ausführen
Beschreibung:	Skript wird ausgeführt
Voraussetzung:	Benutzer hat sein Skript in das Feld eingegeben
Eingabe:	Benutzer drückt auf den «Ausführen» Knopf
Erwartetes Resultat:	Das Skript wurde überprüft und ausgeführt

Tabelle 14: Testfall 3

Testfall	Nr.4
Anforderung:	Der Benutzer kann in der Groovy Konsole eine Datei hochladen
Beschreibung:	Datei hochladen
Voraussetzung:	Benutzer ist in der richtigen Perspektive und hat die Benutzerrechte
Eingabe:	Benutzer drückt auf den «Datei hochladen» Knopf
Erwartetes Resultat:	Benutzer kann eine beliebige Datei hochladen

Tabelle 15: Testfall 4

Testfall	Nr.5
Anforderung:	Der Benutzer kann die hochgeladene Datei ausführen. Die Datei wird überprüft, ob es ein Skript ist oder nicht
Beschreibung:	Hochgeladenes Skript ausführen
Voraussetzung:	Benutzer hat eine Datei hochgeladen
Eingabe:	Benutzer drückt auf den «Ausführen» Knopf
Erwartetes Resultat:	Das Skript wurde ausgeführt, der Benutzer bekommt eine Meldung, falls die Datei kein ausführbares Skript ist.

Tabelle 16: Testfall 5

Testfall	Nr.6
Anforderung:	Die Resultate der Skripte werden in einem separaten Feld aufgezeigt
Beschreibung:	Resultat
Voraussetzung:	Benutzer hat ein Skript ausgeführt
Eingabe:	-
Erwartetes Resultat:	Das Resultat wird korrekt aufgezeigt, bei fehlerhaften Eingaben wird der Benutzer darauf aufmerksam gemacht und das Resultat wird leer zurückgegeben.

Tabelle 17: Testfall 6

Logging-Konzept

Log4j ist das Framework in Java mit welchem Entwickler das Logging von Anwendungsmeldungen verfolgen und entwickeln. Hier eine Auflistung der verschiedenen Log Level und meine Loggingfälle für meine Arbeit:³

- log.fatal
 - Diese Fehler brechen das Programm ab
- log.error
 - Diese Fehler behindern die Funktion der Anwendung, oder sind ein erwartender Programmfehler
- log.warn
 - Benutzerfehler oder ungünstiger Programmzustand, fehlerhafter Aufruf einer Schnittstelle
- log.info
 - Laufzeitinformationen, Benutzeranmeldungen und -abmeldungen, Start und Ende
- log.debug
 - Informationen zum Programmablauf
- log.trace
 - Detaillierte Verfolgung des Programmablaufs

Logginfall	Nr.1
LOG-Level:	LOG.Info
Meldung:	Eingaben werden überprüft
Anforderung:	Der Benutzer hat sein Skript in dem Textfeld eingegeben.

Tabelle 18: Loggingfall 1

Logginfall	Nr.2
LOG-Level:	LOG.Info
Meldung:	Eingaben sind richtig, Skript wurde ausgeführt
Anforderung:	Die Eingaben waren korrekt und das Skript wurde ausgeführt

Tabelle 19: Loggingfall 2

Logginfall	Nr.3
LOG-Level:	LOG.Warn
Meldung:	Skript konnte nicht ausgeführt werden
Anforderung:	Eingaben des Benutzers sind falsch, das Skript wurde nicht ausgeführt

Tabelle 20: Loggingfall 3

³ [Tutorialspoint: Log4j Logging Level](#)

Logginfall	Nr.4
LOG-Level:	LOG.Info
Meldung:	Datei wurde hochgeladen
Anforderung:	Der Benutzer hat eine Datei hochgeladen.

Tabelle 21: Loggingfall 4

Logginfall	Nr.5
LOG-Level:	LOG.Warn
Meldung:	Datei ist kein Skript
Anforderung:	Der Benutzer hat eine Datei hochgeladen, das kein Skript ist.

Tabelle 22: Loggingfall 5

Logginfall	Nr.6
LOG-Level:	LOG.Info
Meldung:	Skript wurde ausgeführt
Anforderung:	Der Benutzer hat seine Skript Datei ausgeführt

Tabelle 23: Loggingfall 6

Logginfall	Nr.7
LOG-Level:	LOG.Warn
Meldung:	Datei wurde ausgeführt, leeres Resultat wird zurückgegeben
Anforderung:	Der Benutzer hat eine Datei ausgeführt, welches kein Skript ist. Er bekommt ein leeres Resultat zurück

Tabelle 24: Loggingfall 7

Logginfall	Nr.8
LOG-Level:	LOG.Warn
Meldung:	Keine Eingaben
Anforderung:	Der Benutzer drückt den «Ausführen» Knopf obwohl er kein Skript eingetippt oder hochgeladen hat.

Tabelle 25: Loggingfall 8

Entscheiden

Das Entscheiden ist die dritte Phase der IPERKA-Methode. In dieser Phase wird der Lösungsweg entschieden und festgehalten.

Lösungsweg

Ich werde in dieser Arbeit die Projektmethode IPERKA verwenden, es gibt hier aber auch weitere Methoden wie bspw. Scrum, Alpen Methode oder Agile. Ich habe mich jedoch für IPERKA entschieden, da ich damit auch schon in der Schule gearbeitet habe und dies mir am meisten vertraut ist.

Wie mir schon vorgegeben wurde benutze ich für diese Arbeit die Programmiersprache Java und werde meine Dateien in der integrierten Entwicklungsumgebung IntelliJ IDEA bearbeiten. Ich werde mich bei der Struktur an das MVC(Model-View-Controller) Muster halten, welches in der Programmierwelt benutzt wird, um die relevante Logik in drei verbundene Elemente aufteilt⁴. Wie bereits oftmals erwähnt, werde ich in diesem Projekt mit SAP Commerce arbeiten. SAP ist eine Firma, die viele Lösungen für Unternehmen und deren Kunden entwickelt und zur Verfügung stellt, darunter auch die SAP Commerce Plattform. Die Commerce Plattform dient als Lösung für Unternehmen mit E-Business auch als Onlineshops bekannt.⁵

Von der Commerce Plattform benutze ich dann eine Instanz, welche ich mit sogenannten „Recipes“ initialisieren kann und schlussendlich aufstarten kann. In der Instanz der SAP Commerce Plattform befindet sich das Backoffice, dort sind alle Daten von Produkten, Kunden usw. gespeichert. In dieser Backoffice Ansicht werde ich dann auch meine eigene Perspektive für die Groovy Konsole einbauen. Bei dieser Perspektive ist es wichtig, dass auch nur eine bestimmte Benutzergruppe Zugriff darauf hat. Diese Gruppe werde ich dann erstellen und eine ImpEx Datei dafür aufbereiten damit auch andere diese Gruppe in ihre SAP Commerce Instanz per Import hinzufügen können. „Recipes“ oder auch „installer recipes“ werde strikt nur für Demo- und Entwicklungszwecke benutzt. Sie sind ein vereinfachter und schneller weg um die Erweiterungen und AddOns, für die SAP Commerce Platform, zu installieren.⁶ Es gibt eine Vielzahl von verschiedenen «recipes» die man dafür benutzen kann, nach Absprache mit meinem Berufsbildner haben wir dann aber entschieden, dass ich das «cx-china recipe» benutzen werde. Grund dafür war schlicht, dass dieses «recipe» schneller initialisiert wird und bereits Daten von Produkten und Kunden in dem Backoffice hat.

Die erstellte Konsole in der neuen Perspektive ist für Groovy Skripte. Groovy ist eine dynamische Sprache für die Java-Plattform, sie wurde von Apache entwickelt. Die Sprache bietet so einige leistungsstarke Funktionen an, darunter Skripting-Fähigkeiten und funktionale Programmierung.⁷

Testing und Logging wird ebenfalls Teil meiner Arbeit sein. Hier wurde mir auch schon vorgegeben mit welchen Frameworks ich arbeiten muss. Meine Klassen werde ich in der IDE mit dem JUnit Framework testen, da dies spezifisch für die Java Programmiersprache entwickelt wurde. Log4j werde ich für das Logging benutzen, damit ich jederzeit sehen kann welche Schritte die Applikation ausführt und wo es allenfalls zu Fehlermeldungen kommt.

⁴ [Codeacademy: MVC](#)

⁵ [SAP Help: Über SAP Commerce](#)

⁶ [SAP Help: Über Recipes](#)

⁷ [Groovy Lang: Über Groovy](#)

Realisieren

Das Realisieren ist die vierte Phase der IPERKA-Methode

Projektumgebung

Zu Beginn erstelle ich unter meinem Benutzer einen neuen Ordner im Finder und nenne diesen «abschluss». Danach füge ich den Ordner «BackofficeExtensionIPA» hinzu, in diesem Ordner werde ich für die nächsten 6 Tage arbeiten und mein Projekt aufbauen.

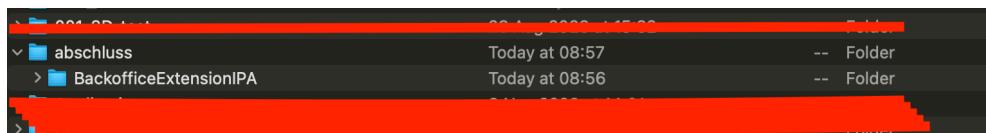


Abbildung 16: Ordner des Abschluss Projekts

SAP Commerce Version herunterladen

Ich wechsle nun zu meinem Chrome Browser und navigiere zu unserem GitLab, ich logge mich ein und navigiere in der Seitenlist zu «Gruppen». Ich suche dann hier nach der «SAP CX Solutions» Gruppe und wähle dort unter «Deploy» die «Package Registry» Option.

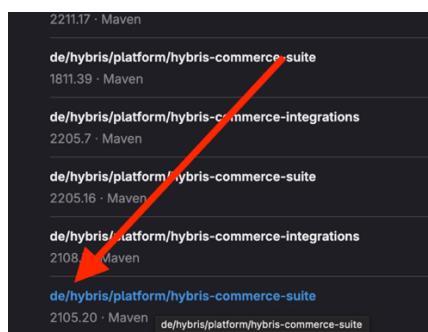


Abbildung 17: SAP Commerce Versionen

Ich bekomme nun eine Liste von verschiedenen SAP Commerce Versionen, aber ich werde hier die 2105.20 Version wählen. Somit habe ich die SAP Commerce Suite mit Backoffice und Hybris Administration Console. Unter «Assets» kann ich dann den «.zip» Ordner herunterladen und in meinen «BackofficeExtensionIPA» Ordner verschieben. Ich muss diesen SAP-Ordner dann auch entzepfen, damit ich später meine eigenen Dateien erstellen kann und auf die verfügbaren Java Klassen im Projekt zugreifen kann.

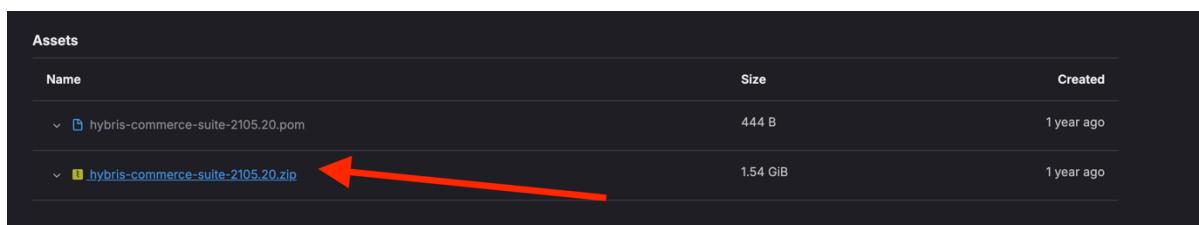


Abbildung 18: ".zip" Datei der benutzten SAP Commerce Version

Shell Skript setjavahome

Sobald dieser Ordner entzipped ist, kann ich durch Rechtsklick auf den Ordner "BackofficeExtensionIPA" ein neues Terminalfenster im gleichen Verzeichnis öffnen. Das Terminal ist die Shell von Mac und ermöglicht es uns, Befehle in ihrer Befehlszeile einzugeben. Ich werde zunächst den Befehl "touch setjavahome.sh" im Terminal verwenden und so ein neues Shell Skript mit dem Namen "setjavahome" erstellen. Dieses Skript werde ich verwenden, um die Variable "JAVA_HOME" zu setzen.

```
tbolli01@CH-C02D1540MD6R BackofficeExtensionIPA % touch setjavahome.sh
```

Abbildung 19: touch Befehl in der CLI

Nun werde ich mit dem zweiten Befehl «vim setjavahome.sh» mit Hilfe von «vim» mein Shell Skript im Terminal öffnen und bearbeiten. Um mit vim eine Datei bearbeiten zu können muss man auf seiner Tastatur zuerst «i» drücken. Damit die Veränderungen auch nicht verloren gehen drücke ich am Schluss die «esc» Taste und gebe «:x!» ein, dies wird meine Änderungen speichern und die Datei wieder schliessen.

Abbildung 20: Inhalt des "setiayahome.sh" Skript

Wir können jetzt vergleichen, ich habe das Skript noch nicht ausgeführt und rufe mit dem Befehl «echo» die Variable «JAVA_HOME» ab.

```
[tbolli01@CH-C02D1540MD6R BackofficeExtensionIPA % echo $JAVA_HOME  
tbolli01@CH-C02D1540MD6R BackofficeExtensionIPA % ]
```

Abbildung 31: echo Befehl in der CLI

Ich bekomme etwas Leeres zurück, also führe ich mein Shell Skript aus, wenn ich danach nochmals «JAVA_HOME» abrufe, bekomme ich den Pfad zu meinem JDK und kann nun mit dem nächsten Schritt der Projektaufsetzung fortfahren.

```
[tbolli01@CH-C02D1540MD6R BackofficeExtensionIPA % . ./setjavahome.sh  
[tbolli01@CH-C02D1540MD6R BackofficeExtensionIPA % echo $JAVA_HOME  
/Users/tbolli01/Library/Java/JavaVirtualMachines/openjdk-11.jdk/Contents/Home  
tbolli01@CH-C02D1540MD6R BackofficeExtensionIPA % █
```

Abbildung 22: erneuter echo Befehl in der CLI

SAP Commerce Instanz - Installer

```
[tbolli01@CH-C02D1540MD6R BackofficeExtensionIPA % cd hybris-commerce-suite-2105.20/installer/]
```

Abbildung 23: Navigation in den installer Ordner

Nachdem ich die Variable "JAVA_HOME" gesetzt habe, kann ich in das Verzeichnis innerhalb von hybris-commerce-suit-2105.20 navigieren. Hier führe ich das Skript install.sh -l aus, um eine Liste aller verfügbaren Recipes zu erhalten.

Für dieses Projekt werde ich das Recipe "cx_china" verwenden. Ich beginne mit der Ausführung dieser Zeile in meinem Terminal ". /install.sh -r cx_china setup -A initAdminPassword=nimda". Dadurch wird die Plattform eingerichtet, die ich verwenden werde, und die Variable „initAdminPassword“ wird auf "nimda" gesetzt. Dieser Build dauerte nur ein paar Sekunden, und ich kann damit fortfahren, die Plattform mit dem Befehl ". /install.sh -r cx_china initialize -A initAdminPassword=nimda" zu initialisieren. Dieser Build wird länger dauern, etwa eine halbe Stunde, aber sobald er abgeschlossen ist, kann ich die Plattform mit dem Befehl ". /install.sh -r cx_china start" starten. Ich muss diesen letzten Befehl jedes Mal verwenden, wenn ich meine Plattform starten und das Backoffice sehen möchte.

```
[tbolli01@CH-C02D1540MD6R installer % ./install.sh -l
Available recipes:
Recipe "cx_china_b2b"
Description:
=====
Ext-gen B2B Accelerator, based on yacceleratorstorefront, with b2bacceleratoraddon, commerceorgaddon, customerticketing and promotions engine.
Recipe "cx_old_occ_v1"
Description:
=====
CX Core
B2C & B2B Accelerators with OMS (formerly b2c_b2b_acc_oms). For Kyma integration + ApiRegistry, event sending is turned off by default by apiregistryservices.events.exporting=false property. Optionally and before initialization, deployment.api.endpoint property should be set to a server url reachable by kyma instead of https://localhost:9002.
Platform Setup:
  1. Setup platform using command ./install.sh -r cx setup -A initAdminPassword=[password]
  2. Initialize platform using command ./install.sh -r cx initialize -A initAdminPassword=[password]
  3. Start platform using command ./install.sh -r cx start
Recipe "cx_china"
Description:
=====
This recipe provides everything you need to install SAP Hybris Commerce B2C accelerator and marketplace for China with F lash Buy, Alipay Integration, Baidu Map Integration, Wechat Pay Integration and Customer Coupon addons.

Required Configurations:
* For features that require a Baidu API key (such as the Store Locator, which uses Baidu Maps), you need to set the "bai duApiKey"
* For features which related to Alipay Integration and Wechat Pay Integration, please refer to https://help.hybris.com/
* For Kyma integration + ApiRegistry, optionally, ccv2.services.api.url.0 property should be set to a server url reachable by kyma instead of https://localhost:9002.

Platform Setup:
  1. Setup platform using command ./install.sh -r cx_china setup -A initAdminPassword=[password]
  2. Initialize platform using command ./install.sh -r cx_china initialize -A initAdminPassword=[password]
```

Abbildung 24: Liste der verfügbaren Recipes

Projektaufsetzung in der IDE

Da die Plattform nun gestartet ist, kann ich mein Projekt in IntelliJ IDEA einrichten. Ich werde die IDE öffnen und im Taskleistenmenü Datei > Öffnen wählen. Ich navigiere zu meinem Ordner "BackofficeExtensionIPA" und fahre fort zum Verzeichnis hybris-commerce-suite-2105.20 > hybris > bin.

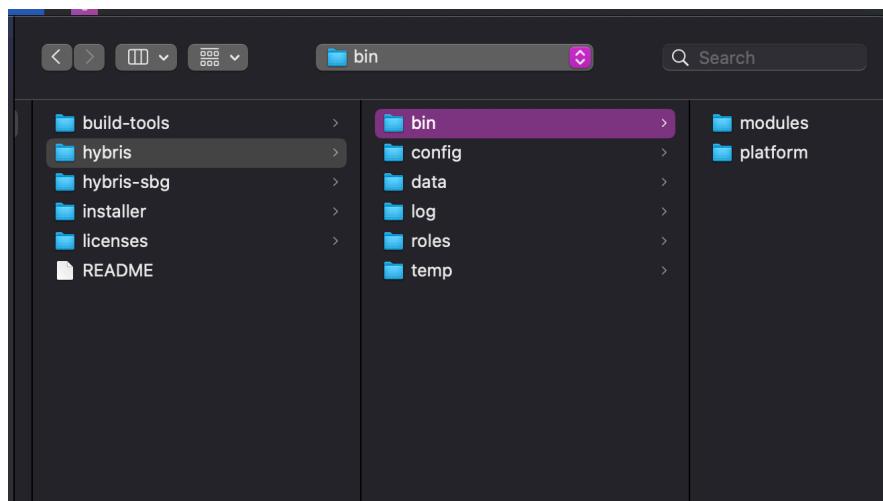


Abbildung 25: bin Verzeichnis

Nachdem ich das "bin" Verzeichnis ausgewählt habe, werde ich gefragt, wie ich mein Projekt importieren möchte. Hier wähle ich "SAP Commerce", um alle relevanten Daten für mein Projekt zu erhalten. Danach kann ich den Projektnamen ändern, hier wähle ich denselben Namen wie meinen "BackofficeExtensionIPA" Ordner.

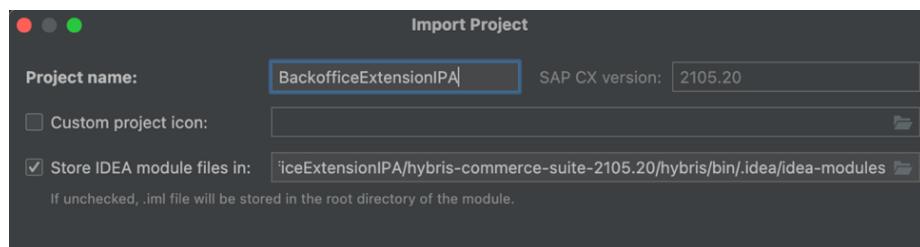


Abbildung 26: Projekt in IntelliJ IDEA importieren

Ich werde fortfahren und ein Projekt SDK auswählen müssen. Hier werde ich dasselbe wählen, dass ich meiner "JAVA_HOME" Variable zugewiesen habe, also Java 11. Als nächstes werde ich aufgefordert, auszuwählen, welche SAP-Commerce-Projekte importiert werden sollen. Ich werde hier nichts ändern und die Standardeinstellungen verwenden. Danach wird mein Projekt importiert und eingerichtet.

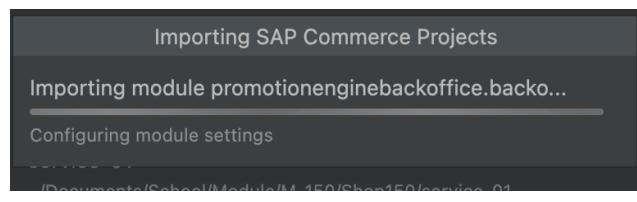


Abbildung 27: Import Prozess

Schliesslich kann ich die Projektstruktur sehen. Derzeit ist sie noch leer, aber im nächsten Schritt werde ich erklären, wie ich die neue Extension erstellt habe, die meiner Struktur hier hinzugefügt wird.

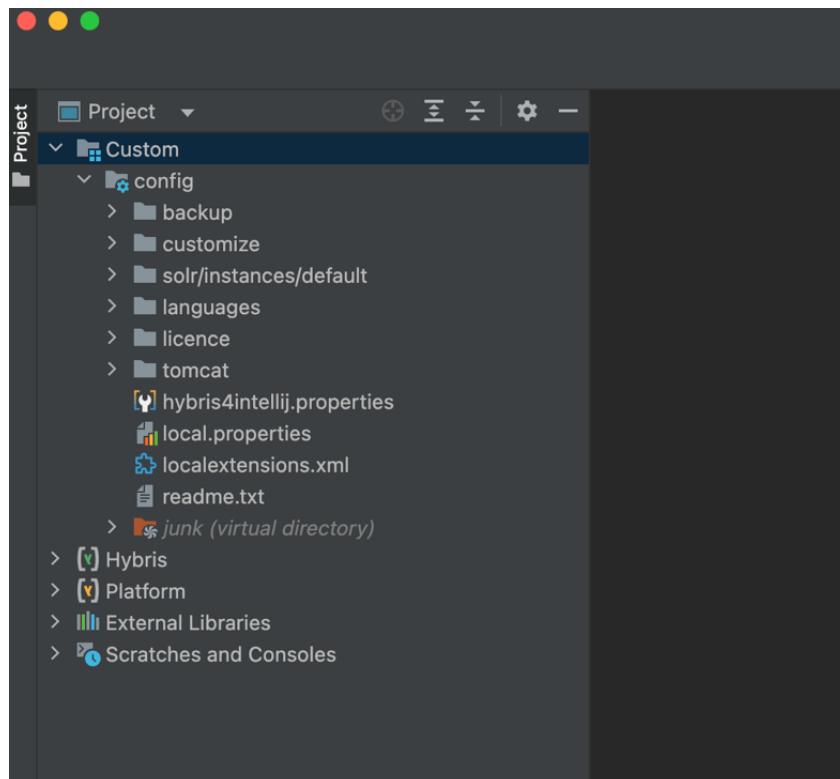


Abbildung 28: Struktur des Projekts in IntelliJ IDEA

Extension und Perspektive

Ich halte die Instanz an und navigiere mit dem Befehl "cd .." einen Ordner zurück. Dann werde ich vom Verzeichnis "hybris-commerce-suite-2105.20" in das Verzeichnis "platform" wechseln, indem ich den Befehl "cd hybris/bin/platform" verwende. Sobald ich in diesem Verzeichnis bin, werde ich das Skript "setantenv.sh" ausführen. Dadurch wird meine Ant Umgebung eingestellt und die Variable "ANT_HOME" gesetzt. Die Verwendung dieses Skripts ermöglicht es mir, alle benötigten "ant" Befehle zu verwenden, die ich benötige, um die Extension zu generieren und mein Projekt neu zu erstellen, wenn ich bestimmte Elemente oder Attribute in meinem Code ändere.⁸

```
tbolli010CH-C02D1540MD6R platform % ./setantenv.sh
Apache Ant(TM) version 1.10.11 compiled on July 10 2021
ant home: /Users/tbolli01/abschluss/BackofficeExtensionIPA/hybris-commerce-suite-2105.20/hybris/bin/platform/apache-ant
ant opts: -Xmx2g -Dfile.encoding=UTF-8
tbolli010CH-C02D1540MD6R platform %
```

Abbildung 29: Ausführung des "setantenv.sh" Skript

Nachdem ich die Variable "ANT_HOME" mit dem Skript gesetzt habe, kann ich im selben "platform"-Verzeichnis den Befehl "ant extgen" verwenden. Für die Generierung meiner benutzerdefinierten Extension habe ich dem Tutorial des offiziellen SAP Help Portals gefolgt. Wenn ich den Befehl "ant extgen" aufgerufen habe, wurde ich gefragt, welches Template ich für meine Extension verwenden möchte. Hier habe ich das "ybackoffice" gewählt, das üblicherweise für die Generierung neuer Backoffice Extension verwendet wird. Anschliessend wurde ich aufgefordert, meine neue Extension zu benennen; ich habe den Namen "Merkle Extension" gewählt. Ich drückte erneut Enter und musste den Namen für mein Paket wählen; ich habe "org.merkle" verwendet. Es wurde mich gefragt, ob ich meine Extension als SASS registrieren möchte, ob ich ein Beispiel Widget verwenden möchte und ob ich ein Beispiel Stylesheet möchte; für alle diese drei Fragen antwortete ich mit Nein.⁹ Meine Extension wurde dann generiert. In IntelliJ IDEA öffnete ich die Datei "localextensions.xml" und fügte meine neu generierte Extension hinzu:

```
<extension name='MerkleExtension'>
```

Danach musste ich mein Projekt mit "ant clean all" neu erstellen und startete den Anwendungsserver mit dem Befehl "./install.sh -r cx_china start". Nun konnte ich meinen "custom" Ordner im "bin" Verzeichnis sehen, und meine Projektstruktur in IntelliJ IDEA enthielt nun die generierte Extension.

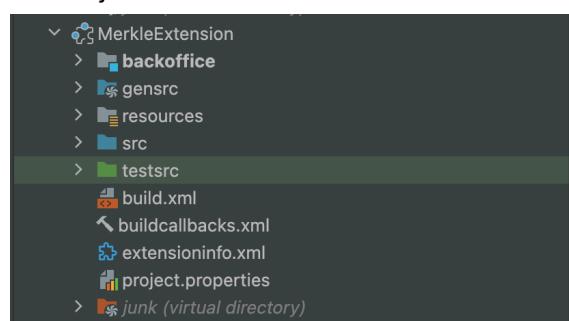


Abbildung 30: Neue Projektstruktur in der IDE

⁸ [SAP Help: Backoffice Extension erstellen](#)

⁹ [SAP Help: Neue Backoffice Extension erstellen](#)

Ich öffnete nun meinen Browser und gab die URL "localhost:9002/backoffice" ein. Dadurch wurde ich zum SAP-Backoffice-Anmeldebildschirm weitergeleitet. Hier werde ich zunächst den Benutzer "admin" verwenden und mich einloggen. Sobald ich im Backoffice eingeloggt war, konnte ich das Administration Cockpit sehen, es ist die Standardansicht für den Benutzer "admin". Auf meiner Tastatur drückte ich dann F4, um in den Orchestrator Modus zu wechseln. Dies ist im Grunde ein Modus, der es mir ermöglicht, die laufende Backoffice Ansicht zu bearbeiten. Durch diesen Modus kann ich Widgets für meine neue Perspektive hinzufügen, obwohl ich meine Änderungen aus dem Orchestrator Modus immer noch in die entsprechenden Dateien in meinem Projekt einfügen muss, das in IntelliJ IDEA geöffnet ist.

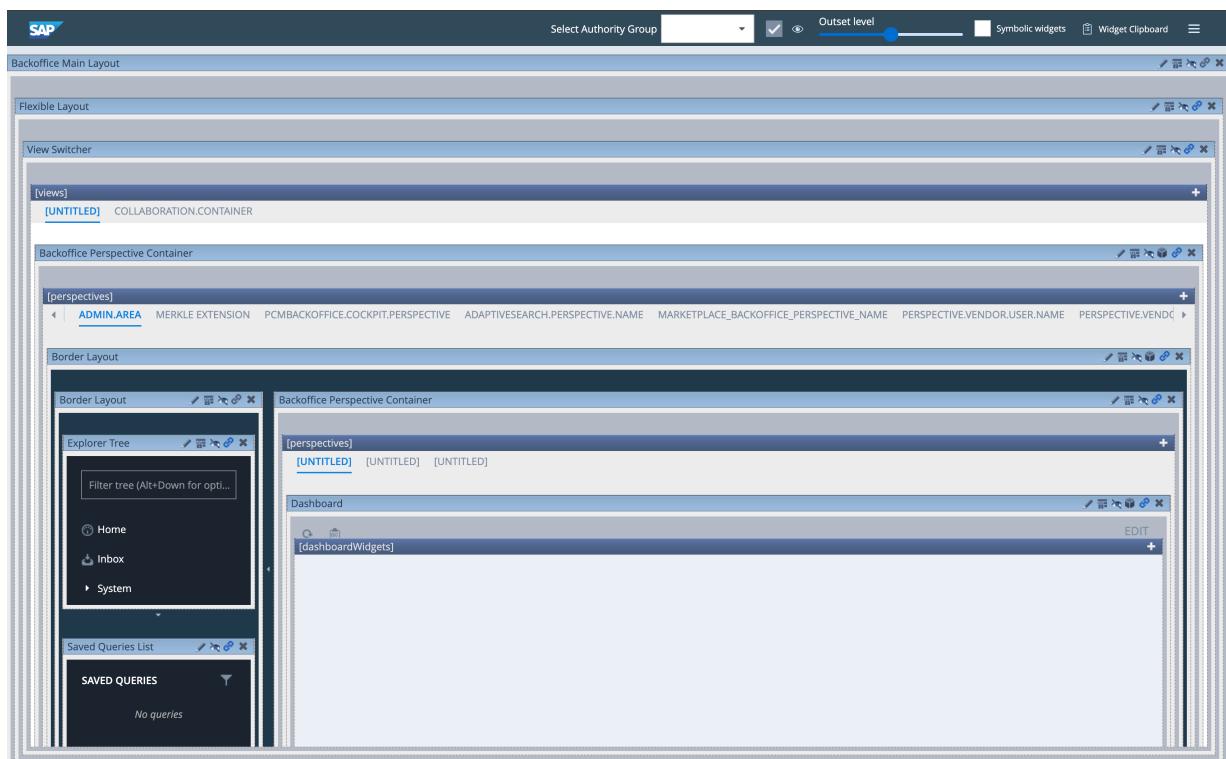


Abbildung 31: Backoffice Ansicht im Orchestrator Modus

Ich verwende die bereits generierten leeren Dateien in meinem Ressourcenverzeichnis namens "MerkleExtension-backoffice-config.xml" und "MerkleExtension-backoffice-widgets.xml". Wie ich bereits erwähnt habe, habe ich den Orchestrator Modus verwendet, um meine Perspektive hinzuzufügen. Durch die Änderungen, die ich im Orchestrator Modus vorgenommen habe, wird die Datei "cockpit-config.xml" bearbeitet. Diese Datei ist im Wesentlichen die gesamte zusammengeführte UI Konfiguration des Backoffice und wird als "Media" Item gespeichert. Der Grund, warum ich meine eigene "config.xml"-Datei hinzufügen muss, liegt darin, dass die "cockpit-config" nur die UI Konfiguration auf meinem eigenen Laptop ändert. Natürlich verwende ich daher die Datei "MerkleExtension-backoffice-config.xml", um genau die gleichen Änderungen hinzuzufügen, sodass jeder mit meinem Code die gleiche Backoffice Ansicht sehen kann. Bei "MerkleExtension-backoffice-widgets.xml" ist es dasselbe: Hier werden alle meine Widget Konfigurationen der "backoffice-widgets.xml" Datei hinzugefügt, wann immer ich etwas im Orchestrator Modus ändere.¹⁰

```
<?xml version="1.0" encoding="UTF-8"?>
<widgets xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.hybris.com/schema/cockpitng/widgets.xsd">

  <widget-extension widgetId="backofficeMainSlot" name="MerkleExtension">
    <widget id="groovy-perspective" widgetDefinitionId="org.merkle.widgets.groovykonsole" slotId="perspectives"
      template="false" title="Merkle Extension" access="groovybackofficemanager">
    </widget>
  </widget-extension>

</widgets>
```

Abbildung 32: MerkleExtension-backoffice-widgets.xml

Als nächstes, nachdem ich den Orchestrator Modus verlassen habe, bin ich zurück zur Perspektive Administration Cockpit gewechselt und habe im Explorer Tree zu "User" navigiert. Jetzt kann ich einen Kunden, Mitarbeiter oder eine Gruppe auswählen. Für diesen Schritt habe ich zunächst eine Backoffice Rolle erstellt und dann einen neuen Mitarbeiter "groovyAdmin" angelegt. Ich habe die Backoffice Rolle dem neuen Benutzer hinzugefügt und die Backoffice Rolle als Ansichtsberechtigung mit erneuter Hilfe des Orchestrator Modus hinzugefügt.¹¹



Abbildung 33: Restriktionen der Perspektive im Orchestrator Modus setzen

¹⁰ [SAP Help: UI Konfiguration](#)

¹¹ [SAP Community: Backoffice Restriktionen von Widgets](#)

Meine Konfiguration der Backoffice Rolle war jedoch unvollständig, daher war mein Backoffice nach dem builden meines Projekts und dem Start der Anwendung plötzlich defekt, und ich konnte das „view-switcher“ Widgets und keine anderen Perspektiven mehr sehen. Ich habe daraufhin mit meinem Berufsbildner, Janes Thomas das Problem besprochen und er hat mir geholfen. Wir haben zunächst versucht das fehlerhafte Backoffice im Orchestrator Modus zu sehen. Wir bemerkten, dass alle Perspektiven noch hier waren. Anschliessend haben wir im Browser in den Inspektor Modus gewechselt und unter „Application“ die Cookies sowie den lokalen Speicher des Browsers gelöscht, gefolgt von dem Versuch, die cockpit-config Datei im Orchestrator Modus zurückzusetzen.

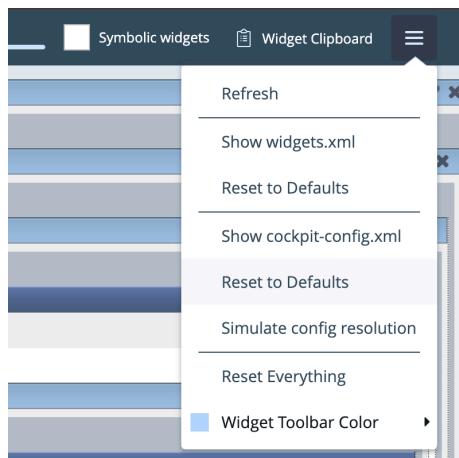


Abbildung 34: cockpit-config.xml Datei im Orchestrator Modus zurücksetzen

Das Problem bestand jedoch weiterhin, weshalb wir dann meinen Code überprüften. Wir untersuchten die Backoffice Rolle und den Benutzer, welchen ich erstellt habe und wir bemerkten, dass meine Backoffice Rolle einige Konfigurationen fehlte. Über einem ImpEx Import in der HAC haben wir diesen Fehler dann behoben. Diese ImpEx Datei habe ich schlussendlich nochmals überarbeitet, mehr dazu im «ImpEx» Abschnitt der Dokumentation.



```
user.impex 143 B
1 | INSERT_UPDATE User;UID[unique=true];groups(uid)[mode=append];
2 | ;groovyAdmin;backofficeadministratorrole, groovyBackofficeRole
3 |
```

Abbildung 35: Veraltetes ImpEx mit Benutzerberechtigung

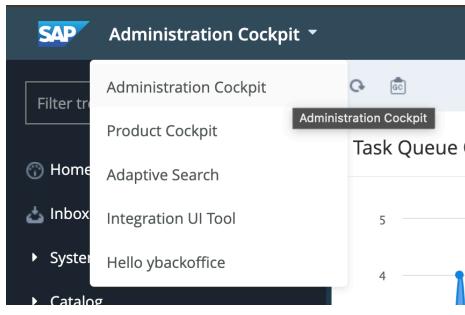


Abbildung 36: perspective-switcher als admin

Mit meinem Berufsbildner habe ich mich wieder als „groovyAdmin“ im Backoffice angemeldet. Mit diesem Benutzer konnte ich meine erstellte Perspektive sehen aber keine weitere Ansicht des Backoffice. Als wir uns wieder als normalen „admin“ Benutzer anmeldete, bestand die fehlerhafte Ansicht immer noch. Ich konnte die Standardperspektiven des „admin“ Nutzers immer noch nicht sehen. Diese leere Ansicht wies auf ein potenzielles Problem mit der „cockpit-config“ Datei oder mit der Konfiguration des „admin“ Benutzers, welches verhindert, dass das Backoffice ordnungsgemäß angezeigt wird.

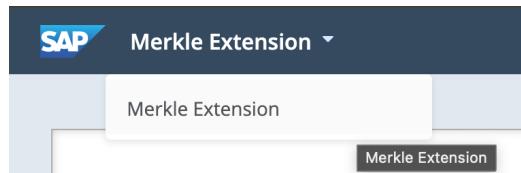


Abbildung 37: perspective-switcher als groovyAdmin

Da ich die Groovy Konsole mit meinem “groovyAdmin” Benutzer sehen konnte, beschlossen wir, dass ich die Arbeit einfach mit dem erstellten Benutzer fortsetzen würde. Ausserdem habe ich mein “perspective-view” Widget aktualisiert, um sicherzustellen, dass es genau dem Tutorial entspricht, dem ich zuvor gefolgt bin ¹².

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--
Copyright (c) 2020 SAP SE or an SAP affiliate company. All rights reserved
-->
<config xmlns="http://www.hybris.com/cockpit/config">
    <context principal="groovybackofficemanagergroup" component="perspective-view-switcher" module="MerkleExtension">
        <y:view-switcher xmlns:y="http://www.hybris.com/cockpitng/config/viewSwitcher">
            <y:authority name="groovybackofficemanager">
                <y:view id="groovy-perspective"/>
            </y:authority>
        </y:view-switcher>
    </context>
</config>
```

Abbildung 38: MerkleExtension-backoffice-config.xml

¹² [SAP Community: Neue Backoffice Perspektive erstellen](#)

Wie im Code zusehen ist, folgt dieses «perspective-view-switcher» Widget dem Prinzip der Backoffice Rolle «groovybackofficemanagergroup», dies bedeutet, dass die Konfigurationseinstellung dieses Widgets ausschliesslich für die Benutzer der Backofice Rolle gelten. Die Einstellungen der «view-switcher» ordnet die Perspektive der Benutzerrolle zu und definiert mit dem Element «authority» die Berechtigung für die Benutzer im Zusammenhang mit dem Widget. Wenn also die Konfigurationen fehlerfrei implementiert werden, können nur Mitarbeiter Benutzer der Gruppe «groovybackofficemanagergroup» meine Groovy Konsole im Backoffice sehen.¹³

Groovy Konsole

Ich begann damit, die beiden neuen Dateien namens "definition.xml" und "groovyKonsole.zul" in meinen Widgets Ordner hinzuzufügen, der sich im Verzeichnis backoffice > resources > widgets meiner Extension befindet, indem ich den Befehl CMD+N in der IntelliJ IDE verwendete.

In der Datei definition.xml fügte ich die folgende Konfiguration hinzu:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<widget-definition id="org.merkle.widgets.groovykonsole" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.hybris.com/schema/cockpitng/widget-definition.xsd">

    <name>Groovy Konsole</name>
    <description>Groovy Konsole für Skripte</description>
    <defaultTitle>Groovy Konsole</defaultTitle>
    <author>Tabea Bolliger</author>
    <version>1.0</version>
    <view src="groovyKonsole.zul"/>

    <controller class="org.merkle.controller.GroovyController"/>

</widget-definition>
```

Abbildung 39: definition.xml

Hier definiere ich mein Widget, das ich in meiner Groovy Konsole verwendet habe. Ich gebe dem Widget einen Namen, eine Beschreibung, einen Titel, einen Autor, eine Version und füge die Quelldatei für die Konfiguration meiner Ansicht hinzu. Ausserdem muss ich eine Verbindung zu meiner Controller Klasse hinzufügen, die die gesamte Logik für meine Groovy Konsole enthält.

¹³ [SAP Community: Neue Backoffice Perspektive erstellen](#)

In der "groovyKonsole.zul" Datei habe ich dann die verschiedenen Frontend Komponenten für meine Groovy Konsole erstellt. Ich habe ein Textfeld hinzugefügt, das der Benutzer verwenden kann, um sein eigenes Skript zu schreiben, und einen Knopf. Beide Komponenten haben eine eindeutige ID, die später in der Controller Klasse verwendet wird.¹⁴

```
<widget xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.zkoss.org/2005/zul">
    <viewModel @id('vm') @init('org.merkle.model.GroovyFileModel')>
        <div>
            <hlayout>
                <div style="width:100%;">
                    <hlayout id="GroovyWrapper" style="height: 250px; width: 100%;">
                        <textbox rows="5" cols="40" id="groovyInput" style="width: 500px; height: 200px; margin-top:40px; margin-left:40px;" />
                    </hlayout>
                </div>
                <button id="executeBtn" label="Execute"/>
            </hlayout>
        </div>
    </viewModel>
</widget>
```

Abbildung 40: groovyKonsole.zul

In meiner "local.properties" Datei habe ich auch die folgende Konfiguration hinzugefügt. Diese Konfiguration ermöglichte es mir, die „.zul“ Datei zu bearbeiten und die Resultate direkt im Backoffice zusehen, ohne dass ich die Applikation neu aufstarten muss.¹⁵

```
backoffice.cockpitng.additionalResourceLoader.enabled=true
backoffice.cockpitng.uifactory.cache.enabled=false
backoffice.cockpitng.widgetclassloader.resourcecache.enabled=false
backoffice.cockpitng.resourceloader.resourcecache.enabled=false
```

Abbildung 41: Zusätzliche Konfigurationen in local.properties

Dann setzte ich meine Arbeit fort, indem ich meine "GroovyController" Klasse für die Logik meiner Konsole erstellte. Ich hatte auch die neue Model Klasse "GroovyFileModel" hinzugefügt, werde sie jedoch erst später für die Logik meiner Skript Upload Funktionen verwenden. Vorläufig hatte ich nur die neue Methode "executeScriptFromString" in meiner Controller Klasse hinzugefügt.

¹⁴ [ZKOSS: Textbox Element](#)

¹⁵ [SAP Help: Ein Widget erstellen](#)

Doch als ich mein Projekt einmal neu builden wollte, konnte ich plötzlich nicht mehr auf das Backoffice zugreifen. Nachdem ich meinen Berufsbildner um Hilfe bat, fanden wir schnell einen Grund heraus. Offenbar ist der Server für meine Anwendung eher langsam beim Herunterfahren. Obwohl mir im Terminal mitgeteilt wurde, dass der Server heruntergefahren worden sei, dauerte es trotzdem eine Weile, bis der eigentliche Prozess beendet war. Ich hatte meine Anwendung immer zu schnell neu gestartet, was den Prozess, der versuchte herunterzufahren, durcheinanderbrachte und den neu gestarteten Prozess beeinträchtigte. Wir haben dieses Problem schnell behoben, indem wir den "Activity Monitor" geöffnet haben, den ich nun benutzt habe, um zu überprüfen, ob mein Server ordnungsgemäß heruntergefahren war oder ob ich ihn manchmal erzwingen musste.



Abbildung 42: Force Stop von Java Prozess im Activity Monitor

Nachdem ich immer noch mit dem Problem der Backoffice Ansicht mit dem "admin" Benutzer zu kämpfen hatte, entschied ich mich, meine Umgebung neu aufzusetzen. Da ich mit den vorherigen Arbeitspaketen schnell vorankam, konnte ich dies ohne grösseren Zeitverlust durchführen.

Zuerst lösche ich den Ordner "MerkleExtension" und generierte die Extension mit dem Befehl "ant extgen" neu. Danach entfernte ich den Inhalt des neuen "MerkleExtension" Ordners erneut und öffnete ihn in IntelliJ. Anschliessend führte ich ein "git pull" durch und überprüfte meine Daten. Als Vorsichtsmassnahme erstellte ich die Backoffice Rollen und Autoritäten neu. Danach erstellte ich den Benutzer "groovyAdmin" erneut.

Durch Navigieren zu "Multimedia" > "Medien" im Backoffice konnte ich die Dateien "widgets.xml" und "cockpit.xml" herunterladen und überprüfen, ob meine Konfigurationen ordnungsgemäss hinzugefügt wurden.

Trotz der neuen Aufsetzung der Extension bestand das Problem, ich konnte zwar, als «admin» Benutzer die normale Backoffice Ansicht sehen, jedoch konnte ich mit dem «groovyAdmin» immer noch nur meine erstellte Groovy Konsole Ansicht sehen. Aus Neugier fragte ich meinen Berufsbildner am zweit letzten Tag der IPA, ob er meinen hochgeladenen Code auf GitLab auf seinem Laptop austesten könnte. Wir stellten fest, dass er als «groovyAdmin» die gleichen Ansichten wie der «admin» Benutzer sah und meine erstellte Ansicht. Wir haben unsere Konfigurations Dateien («backoffice-config.xml» und «backoffice-widgets.xml») verglichen, wir fanden aber keine Abweichungen. Wieso genau die Ansicht auf meinem Gerät noch fehlerhaft ist, ist mir bis jetzt unbekannt. Ich vermute, dass das Problem an meinem Gerät liegt und nicht an meinem Code.

executeScriptFromString Methode

In dieser Methode wird auf ein Klick-Event des Knopfs "executeBtn" reagiert, indem ein Groovy Skript ausgeführt wird, das aus dem Eingabefeld "groovyInput" stammt. Der Versuch, das Skript auszuführen, wird in einem Try-Catch umschlossen. Dadurch ist es möglich, auf allfällige Ausnahmen während der Ausführung des Skripts zu reagieren.

Zu Beginn hatte ich Probleme mit dieser Methode, weil meine Anwendung das verwendete "ScriptingLanguagesService" Bean nicht erkannte. Ich bat erneut meinen Berufsbildner um Hilfe, und wir fügten diese Zeile zu meinem Code hinzu.

```
scriptExecutable = ((ScriptingLanguagesService) Registry.getApplicationContext().getBean( s: "defaultScriptingLanguagesService"))
    .getExecutableByContent(new SimpleScriptContent( engineName: "groovy", content));
```

Abbildung 43: Registration von dem ScriptingLanguagesService Bean

- Registry.getApplicationContext(): Zugriff auf den Anwendungskontext, um alle Beans in der Anwendung abzurufen.
- .getBean(""): Ruft ein spezifisches Bean mit dem angegebenen Namen ab.
- (ScriptingLanguagesService): Eine Typumwandlung, um das zurückgegebene Bean als ScriptingLanguagesService zuzuweisen.
- .getExecutableByContent: Ermöglicht das Abrufen des ausführbaren Skripts anhand des Inhalts.
- new SimpleScriptContent("groovy", content): Erstellt ein neues Objekt vom Typ SimpleScriptContent, wobei "groovy" das Skriptformat angibt und „content“ den Inhalt des Skripts enthält.

Nachdem das Problem mit dem Bean behoben war, habe ich die Methode "executeScript" erstellt, da ich den "ScriptingLanguagesService" nicht nur in der Methode "executeScriptFromString", sondern auch in der Methode für den Datei Upload benötigte. Doppelter Code mit derselben Logik wäre nicht effizient gewesen, daher habe ich die Logik für die eigentliche Skriptausführung in dieser neuen Methode hinzugefügt.

Mit diesen Änderungen sah mein endgültiger Code für die Methode wie folgt aus:

```

@ViewEvent(componentID = "executeBtn", eventName = Events.ON_CLICK)
public void executeScriptFromString() throws RuntimeException{

    try{
        LOG.info("Input is being checked");
        if (StringUtils.isEmpty(groovyInput.getValue())){
            LOG.warn("Groovy String Input equals null");
        } else {
            String content = groovyInput.getValue().toString();
            executeScript(content);
        }
    } catch (RuntimeException e){
        Messagebox.show("Exception occurred while trying to execute Groovy script. Please check your script!");
        LOG.warn("Exception occurred while trying to execute Groovy script");
    }
}
    
```

Abbildung 44: executeScriptFromString Methode im Controller

Innerhalb des Try-Blocks wird zuerst überprüft, ob die Eingabe des Benutzers leer ist, indem die Methode «StringUtils.isEmpty()» aus der Apache Commons Bibliothek verwendet wird. Wenn die Eingabe nicht leer ist, wird das Skript ausgeführt, indem die Methode «executeScript(content)» aufgerufen wird.

Falls während der Ausführung des Skripts eine Ausnahme vom Typ `RuntimeException` auftritt, wird ein entsprechendes Pop-up Fenster mit der Meldung "Exception occurred while trying to execute Groovy script. Please check your script!" angezeigt. Zusätzlich wird eine Warnung mit dem Logger protokolliert, um das Auftreten der Ausnahme zu dokumentieren.

executeScriptFromFileUpload Methode

Für diese Funktion konnte ich im Internet ein gutes Beispiel von der Online Plattform ZK Fiddle benutzen. Auf dieser Plattform gibt es viele verschiedene Beispiele, die von Benutzern erstellt worden sind und welche man direkt in Echtzeit austesten kann.

Ich hatte dieses Tutorial¹⁶ von ZK Fiddle verwendet und die neue Model Klasse "AttachmentFile" erstellt. Ich ging den Code durch, löschte die Segmente, von denen ich dachte, dass sie unnötig waren, und fügte die benötigten Methoden meiner Klasse "GroovyFileModel" hinzu.

¹⁶ [ZKFiddle: Beispiel zu einem File Uploader](#)

Damit der Uploader tatsächlich in meiner Groovy Konsole zu sehen ist, musste ich auch die Datei "groovyKonsole.zul" aktualisieren, indem ich folgenden Code hinzufügte.

```

<groupbox>
  <textbox width="75%" name="dosPath" tabindex="0"
    value="@bind(vm.dosPath)" maxLength="255" />
  <button id="btnUpload" label="Upload"
    onUpload="@command('onFileUpload') upload=@load(vm.uploadString)"
    mold="trendy" />

  <div style="padding-left: 10px; padding-right: 10px;">
    <listbox id="listbox" model="@load(vm.attachment fileList)"
      visible="@load(not empty vm.attachment fileList)" sclass="listbox">
      <listhead>
        <listheader width="93%"></listheader>
        <listheader width="7%"></listheader>
      </listhead>
      <template name="model" var="var">
        <listitem>
          <listcell style="cursor:auto">
            <label tooltiptext="@load(var.fileName)"
              value="@load(var.fileName)" maxlength="65">
            </label>
          </listcell>

          </listitem>
        </template>
      </listbox>
    </div>
  </groupbox>

  <button style="float:right; margin-right:600px;" id="executeScript">
    Execute
  </button>

</widget>
```

Abbildung 45: groovyKonsole.zul

Ich erstellte hier einen Knopf mit der Beschriftung „Upload“, dieser sollte dazu dienen, dass die Benutzer später damit ihre Datei hochladen könnten. Als nächstes eine „listbox“, welche alle hochgeladenen Dateien auflisten sollte und zuletzt einen weiteren separaten Knopf für das Ausführen des hochgeladenen Skripts.

Danach begann ich damit, die Logik für den Upload in meinen Controller einzufügen. Auch damit hatte ich am Anfang meine Schwierigkeiten. Ich versuchte, GroovyShell für die Ausführung meines Skripts zu verwenden, aber nachdem ich meinen Fortschritt mit meinem Berufsbildner besprochen hatte, gab er mir den Tipp, dass ich die Schritte in meiner "executeScriptFromString" Methode ähnlich befolgen könnte.

Am Ende sah mein Code nun so aus:

```
@ViewEvent(componentID = "executeScript", eventName = Events.ON_CLICK)
public void executeScriptFromFileUpload() throws RuntimeException{

    if (CollectionUtils.isEmpty(listbox.getItems())){
        LOG.warn("There is no uploaded file");
    } else {

        try {
            AttachmentFile attachmentFile = (AttachmentFile) listbox.getModel(); //cannot cast listmodellist (listbox) to attachmentfile
            String file = attachmentFile.getUploadedfile().toString();
            BufferedReader reader = new BufferedReader(new FileReader(file));
            String allLines = reader.lines()
                .collect(Collectors.joining(System.lineSeparator()));
            //collectors joining adds all lines into one single string object. linesSeparator -> space
            reader.close();
            executeScript(allLines);

        } catch (RuntimeException | IOException e) {
            Messagebox.show("Exception occurred while trying to execute Groovy script. Please check your script!");
            LOG.warn("Exception occurred while trying to execute Groovy Script from Upload");
        }
    }
}
```

Abbildung 46: executeScriptFromFileUpload Methode im Controller

In dieser Methode wird auf ein Klick-Event des Knopfs "executeScript" reagiert, indem ein Groovy-Skript ausgeführt wird, das aus einer hochgeladenen Datei stammt. Zuerst wird überprüft, ob eine Datei im Listbox Element hochgeladen wurde. Wenn keine Datei vorhanden ist, wird eine Warnung protokolliert, um den Benutzer darüber zu informieren.

In der ersten Zeile des Try-Catch-Blocks wird das Listbox Element verwendet, um das AttachmentFile Model zu erhalten. Das Problem bestand darin, dass mein GroovyFileModel mit der ZUL-Datei verbunden war, der Code jedoch nicht wusste, woher er die Daten beziehen sollte. Deshalb habe ich listbox.getModel verwendet, um das AttachmentFile Model zu erhalten. Innerhalb des Try-Blocks wird versucht, die hochgeladene Datei zu öffnen und deren Inhalt zu lesen. Auch hier wird eine Ausnahme vom Typ `RuntimeException` oder `IOException` auftreten, falls es zu Komplikationen kommt, es wird mit dem Logger eine Warnung im Terminal protokolliert und zusätzlich ein Pop-up Fenster angezeigt.

Während des Codierens stieß ich auf das Problem, dass die Anwendung zwar erkennt, dass eine Datei hochgeladen wurde, jedoch nicht weiß, woher sie die zusätzlichen Aktionen für diese Datei beziehen soll. Dies liegt daran, dass die Methode, «executeScriptFromFileUpload», zur Ausführung des hochgeladenen Skripts in der Controller Klasse liegt, während die Methode zum Abrufen der hochgeladenen Datei in einer separaten Model Klasse liegt.

executeScript Methode

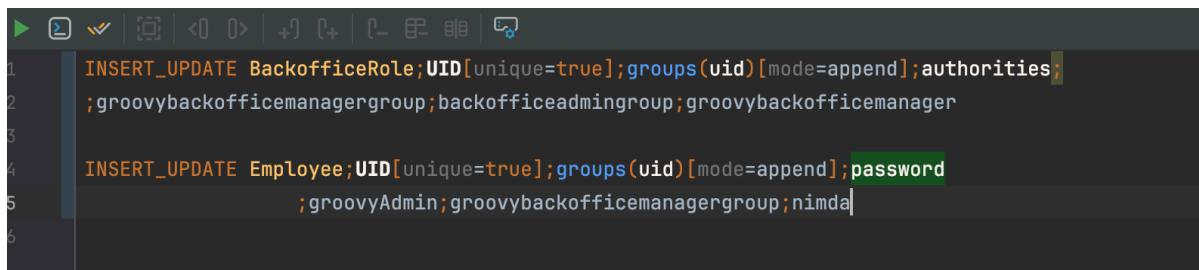
```
public void executeScript(String content){
    scriptExecutable = ((ScriptingLanguagesService) Registry.getApplicationContext().getBean("defaultScriptingLanguagesService"))
        .getExecutableByContent(new SimpleScriptContent(engineName: "groovy", content));
    scriptResult = scriptExecutable.execute();
    Messagebox.show(String.valueOf(scriptResult.getScriptResult()));
    LOG.info("Script was successful");
}
```

Abbildung 47: executeScript Methode im Controller

Diese Methode wird sowohl von „executeScriptFromString“ als auch von „executeScriptFromFileUpload“ verwendet. Sie dient als gemeinsame Funktion für beide Methoden. Das eingegebene oder hochgeladene Skript wird mit Hilfe des „ScriptingLanguagesService“ abgerufen. Mit der Methode „execute()“ wird das Skript, des Objekts „scriptExecutable“ ausgeführt.¹⁷ Das Ergebnis wird dann in „scriptResult“ gespeichert und durch ein Pop-Up Fenster im Backoffice angezeigt.

ImpEx

Zuletzt habe ich das ImpEx erarbeitet, hier musste ich mit INSERT_UPDATE eine neue Backoffice Rolle mit den Attributen "groups" und "authorities" erstellen. Dann habe ich einen neuen Mitarbeiter "groovyAdmin" erstellt, der Teil der von mir erstellten Backoffice Rolle ist. Sobald ein Benutzer dieses Impex im HAC importiert, kann er sich als "groovyAdmin" im Backoffice anmelden und meine benutzerdefinierte Extension verwenden.



```
1  INSERT_UPDATE BackofficeRole;UID[unique=true];groups(uid)[mode=append];authorities;
2 ;groovybackofficemanagergroup;backofficeadmingroup;groovybackofficemanager
3
4  INSERT_UPDATE Employee;UID[unique=true];groups(uid)[mode=append];password
5 ;groovyAdmin;groovybackofficemanagergroup;nimda
6
```

Abbildung 48: Finalisierte ImpEx Datei von Benutzer und Benutzergruppen

Ich habe der Backoffice Rolle die UID «groovybackofficemanagergroup» gegeben, die Gruppe «backofficeadmingroup» hinzugefügt damit der Benutzer auch das Backoffice ansehen kann und zuletzt die Autorität «groovybackofficemanager» zugewiesen. Das gleiche habe ich dann mit dem Mitarbeiter Benutzer gemacht, welcher die UID «groovyAdmin» hatte und das Passwort «nimda».

¹⁷ [Stackoverflow: Groovy Skript mit Java ausführen](#)

Kontrollieren

Das Kontrollieren ist die fünfte Phase der IPERKA-Methode.

Testprotokoll

Testfall	Resultat	Datum	Tester	Bemerkung	Unterschrift
1	Orange	04.04.2024	T.Bolliger	Der admin Benutzer sieht keine Merkle Extension aber der groovyAdmin sieht keine andere Extension als die Merkle Extension. Dies wurde bereits im Realisierungsteil der Dokumentation erklärt.	T.Bolliger
2	Green	04.04.2024	T.Bolliger	Benutzer kann die Perspektive sehen und Eingabefeld betätigen	T.Bolliger
3	Green	04.04.2024	T.Bolliger	Skript wird ausgeführt	T.Bolliger
4	Green	04.04.2024	T.Bolliger	Datei kann hochgeladen werden	T.Bolliger
5	Red	04.04.2024	T.Bolliger	Die Datei wird nicht ausgeführt, es gibt eine Exception	T.Bolliger
6	Orange	04.04.2024	T.Bolliger	Es wird eine MessageBox angezeigt von dem Resultat und eine LOG-Nachricht im Terminal abgebildet	T.Bolliger

Tabelle 26: Testergebnisse

Testfall 1

Dieser Test konnte ich durchführen in dem ich mich in das Backoffice zuerst als «admin» Benutzer anmeldete und danach mich nochmals als «groovyAdmin» Benutzer anmeldete.

Testfall 2

Dieser Test konnte ich durchführen in dem ich mich als «groovyAdmin» Benutzer angemeldet habe und dann in der Groovy Konsole Perspektive alles mögliche in das Textfeld der Konsole eingeben konnte.

Testfall 3

Dieser Test konnte ich als JUnit ausführen, die Eingabe des Benutzers wurde hier von dem Test registriert. Jedoch endet mein Test in der Exception von meinem Try-Catch in der «executeScriptFromString» Methode. Ich wenn ich aber das gleiche Skript manuell im Backoffice eingebe, bekomme ich ein positives Resultat.

Testfall 4

Die Methode für diese Funktion befindet sich in der Model Klasse und ist mit weiteren komplexen Methoden verknüpft. Da ich mich auf das Testing auf die Controller Klasse konzentriert habe, habe ich diesen Aufwand vermieden, alle Methoden der Model Klasse in einem Unit Test abzudecken. Stattdessen habe ich mich für einen schnellen manuellen Test im Backoffice entschieden.

Testfall 5

Dieser Test ist sowohl als Unit- als auch als manueller Test fehlgeschlagen. Das Problem liegt in meinem Code. Wie auch in der Methode 'executeScriptFromString' habe ich die Funktion meiner Methode 'executeScriptFromFileUpload' in einem Try-Catch implementiert. Den Fehler habe ich bereits im Realisierungsteil meiner Dokumentation erklärt. Mir war dieses Problem vor dem Testen bekannt, aber ich konnte es nicht beheben, ohne möglicherweise ein anderes Problem zu verursachen. Nach weiterer Überlegung habe ich festgestellt, dass es definitiv zu viel Zeit kosten würde, um den Fehler zu beheben, weshalb ich beschlossen habe, meine anderen Aufgaben zu priorisieren.

Testfall 6

Dieser Testfall ist ebenfalls als orange gekennzeichnet. Der Grund dafür war, dass ich so stark in die Hauptfunktionen meiner Konsole vertieft war, dass ich vergessen habe, ein solches Ergebnisfeld zu erstellen und hinzuzufügen. Durch Zufall bin ich dann auf die sogenannte "MessageBox" gestossen und habe dies als schnelle und alternative Lösung verwendet.

Logging Protokoll

Logfall	Resultat	Datum	Tester	Bemerkung	Unterschrift
1		04.04.2024	T.Bolliger	Log.info("Input is being checked")	T.Bolliger
2		04.04.2024	T.Bolliger	Log.info("Script was successful")	T.Bolliger
3		04.04.2024	T.Bolliger	Log.warn("Groovy String Input equals null")	T.Bolliger
4		04.04.2024	T.Bolliger	FEHLT	T.Bolliger
5		04.04.2024	T.Bolliger	Log.warn("There is no uploaded file")	T.Bolliger
6		04.04.2024	T.Bolliger	Log.info("Script was successful")	T.Bolliger
7		04.04.2024	T.Bolliger	FEHLT	T.Bolliger
8		04.04.2024	T.Bolliger	Log.warn("Exception occurred while trying to execute Groovy Script from Upload") / Log.warn("Exception occurred while trying to execute Groovy script")	T.Bolliger

Tabelle 27: Loggingergebnisse

Logging Resultat

Hier befinden sich Ausschnitte der verschiedenen Log Resultate. Die komplette Log Datei wird sich im Anhang befinden. Ich habe die Log Datei aus dem «tomcat» Verzeichnis in bin/config Ordner. Diese Log Datei beinhaltet den ganzen Prozess der Applikation, ich habe die Datei für diesen Abschnitt gekürzt und nur die Resultate meines Loggers festgehalten. Die gesamte Log Datei wird sich dann im Anhang befinden.

```
INFO | jvm 1 | main | 2024/04/08 13:29:32.467 | INFO [hybrisHTTP15]
[MERKLE_GROOVY_CONSOLE_LOGGER] Input is being checked
INFO | jvm 1 | main | 2024/04/08 13:29:32.571 | INFO [hybrisHTTP15]
[MERKLE_GROOVY_CONSOLE_LOGGER] Script was successful
INFO | jvm 1 | main | 2024/04/08 13:29:44.388 | INFO [hybrisHTTP25]
[MERKLE_GROOVY_CONSOLE_LOGGER] Input is being checked
INFO | jvm 1 | main | 2024/04/08 13:29:44.388 | WARN [hybrisHTTP25]
[MERKLE_GROOVY_CONSOLE_LOGGER] Groovy String Input equals null
INFO | jvm 1 | main | 2024/04/08 13:29:53.617 | INFO [hybrisHTTP28]
[MERKLE_GROOVY_CONSOLE_LOGGER] Input is being checked
INFO | jvm 1 | main | 2024/04/08 13:29:53.719 | INFO [hybrisHTTP28]
[MERKLE_GROOVY_CONSOLE_LOGGER] Script was successful
INFO | jvm 1 | main | 2024/04/08 13:30:04.541 | INFO [hybrisHTTP40]
[MERKLE_GROOVY_CONSOLE_LOGGER] Input is being checked
INFO | jvm 1 | main | 2024/04/08 13:30:04.742 | WARN [hybrisHTTP40]
[MERKLE_GROOVY_CONSOLE_LOGGER] Exception occurred while trying to execute Groovy
script
INFO | jvm 1 | main | 2024/04/08 13:30:17.576 | WARN [hybrisHTTP7]
[MERKLE_GROOVY_CONSOLE_LOGGER] There is no uploaded file
INFO | jvm 1 | main | 2024/04/08 13:30:27.134 | WARN [hybrisHTTP8]
[MERKLE_GROOVY_CONSOLE_LOGGER] Execption occured while trying to execute Groovy Script
from Upload
INFO | jvm 1 | main | 2024/04/08 13:30:40.769 | WARN [hybrisHTTP15]
[MERKLE_GROOVY_CONSOLE_LOGGER] Execption occured while trying to execute Groovy Script
from Upload
INFO | jvm 1 | main | 2024/04/08 13:30:54.209 | WARN [hybrisHTTP10]
[MERKLE_GROOVY_CONSOLE_LOGGER] Execption occured while trying to execute Groovy Script
from Upload
INFO | jvm 1 | main | 2024/04/08 13:31:15.269 | WARN [hybrisHTTP32]
[MERKLE_GROOVY_CONSOLE_LOGGER] There is no uploaded file
INFO | jvm 1 | main | 2024/04/08 13:31:27.205 | INFO [hybrisHTTP13]
[MERKLE_GROOVY_CONSOLE_LOGGER] Input is being checked
INFO | jvm 1 | main | 2024/04/08 13:31:27.309 | WARN [hybrisHTTP13]
[MERKLE_GROOVY_CONSOLE_LOGGER] Exception occurred while trying to execute Groovy
script
```

Auswerten

Das Auswerten ist die sechste Phase der IPERKA-Methode

Reflexion

Ich empfand die IPA als eine eher knifflige Arbeit, da ich nie einen Probefeldlauf hatte. Trotzdem konnte ich zuvor einige Tipps zur Dokumentation von meinen Kollegen sammeln. Glücklicherweise hat mir meine Kollegin die Grundstruktur ihrer Dokumentation geschickt, was mir viel Zeit eingespart hat. Auch beim Zeitplan habe ich mir Gedanken gemacht und versucht, genügend Zeit für die Programmierung einzuplanen. Ich konnte diesen Zeitplan auch mit meinem Berufsbildner besprechen. Da ich schon in der Schule öfters Arbeiten dokumentieren musste, habe ich auch eingeplant, dass ich täglich meine Vorgehensweise stichwortartig festhalten werde. Dies hat mir im Endspurt dann sehr geholfen, bessere Sätze zu formulieren. Während des ersten Besuchstermins konnte ich meine notierten Fragen stellen, und der Hauptexperte hat sie alle beantwortet. Von meinem Zeitplan und meiner Dokumentation habe ich täglich Versionen in OneDrive hochgeladen. Zu Beginn hatte ich etwas Mühe mit den Diagrammen, was in der Schule nicht so wirklich meine Stärke war. Trotzdem habe ich diese so gut wie möglich erarbeitet und konnte meinen täglichen Fortschritt immer mit meinem Berufsbildner besprechen. Während der Realisierungsphase kam ich anfangs gut voran, jedoch kamen später viele Probleme auf. Glücklicherweise konnte ich diese schnell beheben und verlor nicht allzu viel Zeit. Die Erstellung der Extension war für mich einfach zu verstehen, ich verdanke dies aber auch den vielen Anleitungen im SAP Help Portal. Im Code hatte ich oft Fehler, die ich zuerst nicht verstand, jedoch klickte es sofort jedes Mal, wenn mein Berufsbildner mir auf die Sprünge half. Durch das Finden von Beispielen für einen File Uploader im Internet konnte ich auch bei diesem Arbeitspaket etwas Zeit sparen. Ich muss aber zugeben, dass dieses Arbeitspaket auch das schwierigste für mich war. In den letzten paar Tagen wollte ich die Fehler beibehalten und mich mehr auf die Dokumentation fokussieren, da diese auch der wichtigste Teil meiner Arbeit war. Trotzdem kam mir meine Sturheit dann im Weg und ich versuchte immer noch, den fehlerhaften Code zu korrigieren und weiter an der Dokumentation zu schreiben, jedes Mal, wenn mein Projekt neu gebaut werden musste. Letztendlich konnte ich den Fehler leider nicht beheben und ich schrieb nun meine Tests und schloss langsam, mit viel Stress, meine Dokumentation ab. Mit meiner Leistung bin ich zwar zufrieden, jedoch wäre ich am Ende sehr wahrscheinlich in keinem Stress verfallen, wenn ich mich nur früher auf die Dokumentation konzentriert hätte.

Glossar

Bezeichnung	Beschreibung
Ant	Ist ein, von Apache entwickeltes, Werkzeug welches auf Java basiert. Ant ermöglicht das automatisierte Erzeugen von Computerprogrammen.
Backoffice	Es ist eine Backend Benutzer Ansicht für Entwickler. Im Backoffice können Daten der SAP-Plattform einfach manipuliert werden.
Bean	Ist ein Java Objekt, welches durch Spring verwaltet und instanziert werden.
Build	Der Prozess, um das Projekt sauber neu aufzubauen und die Änderungen im Code zu adaptieren.
CLI	Steht für «Command Line Interface», es ist eine Befehlszeilenschnittstelle
Commit	Kann man sich als Snapshots oder Meilensteine von dem Fortschritt seines Projektes auf Git vorstellen
Controller	Verbindet das Model mit der View. Nimmt die Benutzeraktionen entgegen und wertet diese aus.
Cookies	Kleine Textdateien, welche das Verhalten eines Nutzers im Webbrowser speichern
Explorer Tree	Ein Navigations Widget mit der Auflistung von verschiedenen Kategorien und Subkategorien.
IDE	Steht für Integrated Development Environment. Wird von Entwickler benutzt, um Applikationen zu programmieren.
ImpEx	Sind Kommagetrennte Dateien (CSV) welche den Import und Export von SAP-Daten vereinfachen.
Inspektor Modus	Ein Modus in welchem die Elemente der Webseite sichtbar machen.

Tabelle 28: Glossar

Bezeichnung	Beschreibung
Item	Ein Element in der SAP Commerce Platform, es repräsentiert eine einzelne Variable
Klick-Event	Ein Ereignis, welches abgespielt wird, wenn der Benutzer etwas klickt.
Logging	Automatische Erstellung der Logdatei
Model	Logische Struktur der Daten, diese Klasse sollte keine Informationen über die Benutzeroberfläche beinhalten
MVC	Steht für Model View Control es deutet auf das Entwurfsmuster zur Unterteilung der Software hin.
SDK	Software Development Kit
SASS	Syntactically Awesome Style Sheets. Basiert auf der styling Sprache CSS.
UID	Steht für unique identifier. Es ist ein Kennzeichen, welches nur von einem bestimmten Objekt benutzt wird, keine anderes hat das gleiche Kennzeichen.
UI	Steht für User Interface. Es ist die Ansicht, die wir als Benutzer sehen
UNIX	Eim Betriebssystem für Computer.
Widget	Frontend Komponente, welche benutzt wird, um die Backoffice Ansicht zu stylen
ZUL	Eine Webdatei, welche mit dem ZK-Framework erstellt wurde.

Tabelle 29: Glossar

Quellenverzeichnis

Bexio.com: IPERKA Methode – zuletzt besucht: 08.04.2024

<https://www.bexio.com/de-CH/blog/view/iperka-methode>

richard-seidl.com: Teststufen – zuletzt besucht: 08.04.2024

<https://www.richard-seidl.com/teststufen/>

tutorialspoint.com: log4j Logging Levels – zuletzt besucht: 08.04.2024

https://www.tutorialspoint.com/log4j/log4j_logging_levels.htm

help.sap.com: SAP Commerce – zuletzt besucht: 08.04.2024

https://help.sap.com/docs/SAP_COMMERCER?locale=en-US

help.sap.com: Recipes – zuletzt besucht: 08.04.2024

https://help.sap.com/docs/SAP_S4HANA_ON-PREMISE/1b4bdbde3faf42e7a1d477e7691646eb/aadf50664d5a4a55a58d2a7d48c66fb7.html

groovy-lang.org: About Groovy – zuletzt besucht: 08.04.2024

<https://groovy-lang.org/>

help.sap.com: Creating a Custom Backoffice Extension – zuletzt besucht: 08.04.2024

https://help.sap.com/docs/SAP_COMMERCE_CLOUD_PUBLIC_CLOUD/9b5366ff6eb34df5be29881ff55f97d2/8bd2f59e86691014859ba68181c14710.html

help.sap.com: Creating a New Extension – zuletzt besucht: 08.04.2024

https://help.sap.com/docs/SAP_COMMERCE_CLOUD_PUBLIC_CLOUD/20125f0eca6340dba918bd a360e3cd8a/b96270b86691014b1c3bbfd7556f0ed.html?locale=en-US

help.sap.com: UI Configuration – zuletzt besucht: 08.04.2024

https://help.sap.com/docs/SAP_COMMERCE/5c9ea0c629214e42b727bf08800d8dfa/31b4bcacf3e548efacc8f9ced9641a1a.html

community.sap.com: Leveraging Backoffice Roles: Effective Restriction of Backoffice Access in SAP Commerce Cloud – zuletzt besucht: 08.04.2024

<https://community.sap.com/t5/crm-and-cx-blogs-by-sap/leveraging-backoffice-roles-effective-restriction-of-backoffice-access-in/ba-p/13630435>

community.sap.com: Create a new Backoffice Perspective – zuletzt besucht: 08.04.2024

<https://community.sap.com/t5/crm-and-cx-blogs-by-sap/create-a-new-backoffice-perspective/ba-p/13523712>

zkoss.org: Textbox – zuletzt besucht: 08.04.2024

https://www.zkoss.org/wiki/ZK_Component_Reference/Input/Textbox

help.sap.com: Creating a Widget – zuletzt besucht: 08.04.2024

https://help.sap.com/docs/SAP_COMMERCE/5c9ea0c629214e42b727bf08800d8dfa/8bd4366a86691014b818f8dd642790a8.html

zkfiddle.org: ZK File Upload Example – zuletzt besucht: 08.04.2024

<https://zkfiddle.org/sample/1p66ij3/1-ZK-File-Upload-Exmaple>

stackoverflow.com: Possibilities to execute a groovy script with Hybris in java code – zuletzt besucht: 08.04.2024

<https://stackoverflow.com/questions/72265405/possibilities-to-execute-a-groovy-script-with-hybris-in-java-code-no-hac>

KI-Werkzeuge

chat.openai.com – zuletzt besucht: 08.04.2024

Prompt: «kannst du diesen text etwas flüssiger formulieren: Mich hat das Problem der Backoffice Ansicht mit dem «admin» Benutzer immer noch genervt und so beschloss ich meine Umgebung nochmals aufzusetzen. Dadurch dass ich mit den vorherigen Arbeitspaketen zügig voran kam, konnte ich dies ohne grossen Zeitkonsum durchführen.

Ich habe den Ordner "MerkleExtension" gelöscht und die Extension mit dem Befehl "ant extgen" neu generiert. Anschliessend habe ich den Inhalt des neuen "MerkleExtension"-Ordners erneut gelöscht und ihn dann in IntelliJ geöffnet. Danach habe ich ein "git pull" durchgeführt und meine Daten überprüft. Als Vorsichtsmassnahme habe ich die Backoffice-Rollen und Autoritäten sauber neu erstellt. Danach habe ich den Benutzer "groovyAdmin" erneut erstellt.

Durch das Navigieren zu "Multimedia" > "Medien" im Backoffice konnte ich die Dateien "widgets.xml" und "cockpit.xml" herunterladen und überprüfen, ob meine Konfigurationen ordnungsgemäss hinzugefügt wurden.»

chat.openai.com – zuletzt besucht: 08.04.2024

Prompt: «kannst du diesen selbstgeschriebenen text auf deutsch schreiben: I had used this tutorial from zk fiddle (<https://zkfiddle.org/sample/1p66ij3/1-ZK-File-Upload-Exmaple>) and created the new model class “AttachmentFile”, I went over the code and deleted the segments I thought were unnecessary and added the needed methods into my “GroovyFileModel” class.

For the uploader to actually be seen in my groovy console I had to update the “groovyKonsole.zul” file aswell by adding this code.»

chat.openai.com – zuletzt besucht: 08.04.2024

Prompt: «macht dieser text sinn: Trotz der neuen Aufsetzung der Extension bestand das Problem, ich konnte zwar, als «admin» Benutzer die normale Backoffice Ansicht sehen, jedoch konnte ich mit dem «groovyAdmin» immer noch nur meine erstellte Groovy Konsole Ansicht sehen. Aus Neugier fragte ich meinen Berufsbildner am zweit letzten Tag der IPA, ob er meinen hochgeladenen Code auf GitLab auf seinem Laptop austesten könnte. Wir stellten fest, dass er als «groovyAdmin» die gleichen Ansichten wie der «admin» Benutzer sah und meine erstellte Ansicht. Wir haben unsere Konfigurations Dateien («backoffice-config.xml» und “backoffice-widgets.xml”) verglichen, wir fanden aber keine Abweichungen. Wieso genau die Ansicht auf meinem Gerät noch fehlerhaft ist, ist mir bis jetzt unbekannt. Ich vermute, dass das Problem an meinem Gerät liegt und nicht an meinem Code.»

chat.openai.com – zuletzt besucht: 08.04.2024

Prompt: «kannst du diesen text korrigieren: Wie im Code zusehen ist, folgt dieses «perspective-view-switcher» Widget dem Prinzip der Backoffice Rolle «groovybackofficemanagergroup», dies bedeutet, dass die Konfigurationseinstellung dieses Widgets ausschliesslich für die Benutzer der Backoffice Rolle gelten. Die Einstellungen der «view-switcher» ordnet die Perspektive der Benutzerrolle zu und definiert mit dem Element «authority» die Berechtigung für die Benutzer im Zusammenhang mit dem Widget. Wenn also die Konfigurationen fehlerfrei implementiert werden, können nur Mitarbeiter Benutzer der Gruppe «groovybackofficemanagergroup» meine Groovy Konsole im Backoffice sehen.»

chat.openai.com – zuletzt besucht: 08.04.2024

Prompt: «kannst du das alphabetisch sortieren: CLI, UNIX, Commit, Bean, ImpEx, IDE, ZUL, Logging, MVC, build, Ant, UI, Item, Explorer-tree, Backoffice, Cookies, Widget, Controller, Model, Klick-Event, UID, SDK, SASS, Inspektor Modus»

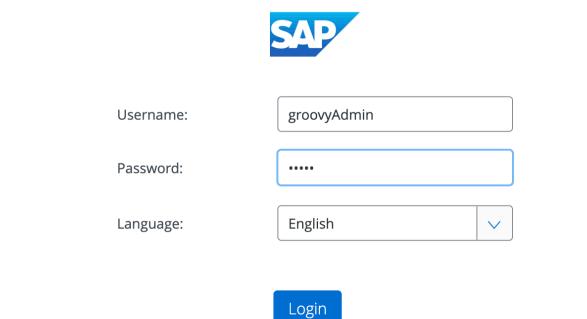
chat.openai.com – zuletzt besucht: 08.04.2024

Prompt: «if "sass" is a "css" preprocessor, does that mean that sass is based on css or does that mean that css is based on sass»

Anhang

Benutzeranleitung

Als allererstes ist es wichtig, dass der Benutzer bereits als «groovyAdmin» angemeldet ist. Damit er das Backoffice sehen kann und die Benutzeranleitung befolgen kann.



The image shows the SAP Backoffice login screen. It features the SAP logo at the top. Below it are three input fields: 'Username' containing 'groovyAdmin', 'Password' containing '*****', and 'Language' set to 'English'. A blue 'Login' button is located at the bottom.

Abbildung 49: Anmeldefenster des Backoffice

Im ersten Schritt werden wir zuerst auf die «Merkle Extension» Perspektive wechseln.



Abbildung 50: Groovy Konsole in der Merkle Extension

Wenn dies erfolgreich getätigt wurde, dann sollten wir die Groovy Konsole Ansicht mit dem Eingabefeld und den verschiedenen Knöpfen «Execute» und «Upload» sehen.

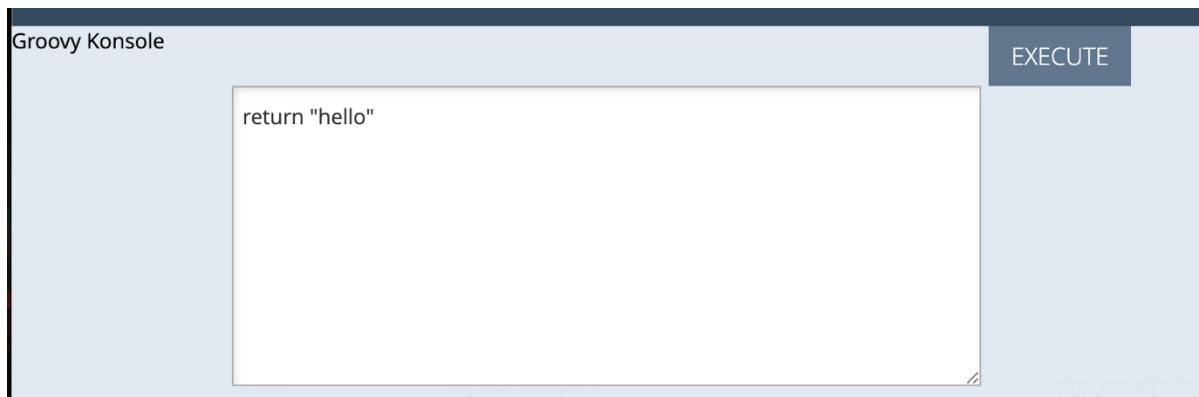


Abbildung 51: Eingabefeld der Groovy Konsole

Im folgenden Schritt können wir dann unser Groovy Skript in dem Eingabefeld schreiben und auf den «Execute» Knopf drücken. Wir bekommen dann eine sogenannte «MessageBox», ein kleines Popup Fenster, welches das Resultat des Skriptes anzeigen.

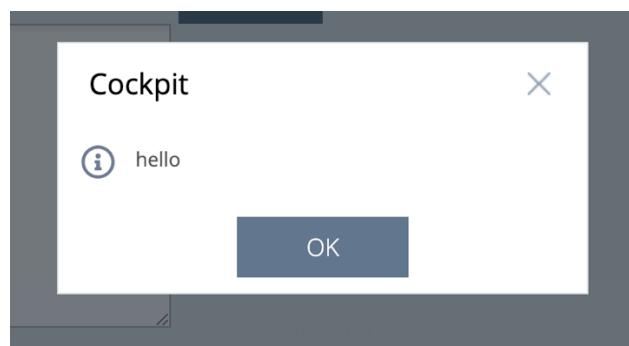


Abbildung 52: Pop-Up Fenster mit dem Resultat des Skripts

Eine weitere Möglichkeit, ist das Skript von unserem lokalen Gerät in die Groovy Konsole hochzuladen, hierfür können wir den «Upload» Knopf drücken und eine Datei aus unserem Gerät auswählen. Hier ist es wichtig zu wissen, dass wir Dateien von mehreren Arten hochladen können aber nur die Groovy Skript Dateien ausführen können. Auch hier wird die «MessageBox» das Resultat des Skriptes uns mitteilen.



Abbildung 53: Datei die in der Konsole hochgeladen wurde

Quellcode

In diesem Abschnitt werden sämtliche Projektspezifische Dateien und deren Inhalte aufgelistet. Ich habe hier keine Seitenumbrüche benutzt, damit die Dokumentation nicht riesengross wird.

GroovyController.java

```
package org.merkle.controller;

import com.hybris.cockpitng.annotations.ViewEvent;
import com.hybris.cockpitng.util.DefaultWidgetController;
import de.hybris.platform.core.Registry;
import de.hybris.platform.scripting.engine.ScriptExecutable;
import de.hybris.platform.scripting.engine.ScriptExecutionResult;
import de.hybris.platform.scripting.engine.ScriptingLanguagesService;
import de.hybris.platform.scripting.engine.content.ScriptContent;
import de.hybris.platform.scripting.engine.content.impl.SimpleScriptContent;
import de.hybris.platform.scripting.engine.exception.ScriptExecutionException;
import org.merkle.model.AttachmentFile;
import org.merkle.model.GroovyFileModel;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.util.CollectionUtils;
import org.springframework.util.StringUtils;
import org.zkoss.zk.ui.event.Events;
import org.zkoss.zul.*;

import java.io.*;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

import org.apache.log4j.Logger;

@Controller
public class GroovyController extends DefaultWidgetController {

    private ScriptingLanguagesService scriptService;
    private ScriptExecutable scriptExecutable;
    private ScriptContent scriptContent;
    private ScriptExecutionResult scriptResult;
    private Textbox groovyInput;
    private Listbox listbox;
    private AttachmentFile attachmentFile;

    protected static final Logger LOG =
Logger.getLogger("MERKLE_GROOVY_CONSOLE_LOGGER");

    @ViewEvent(componentID = "executeBtn", eventName = Events.ON_CLICK)
    public void executeScriptFromString() throws RuntimeException{

```

```

try{
    LOG.info("Input is being checked");
    if (StringUtils.isEmpty(groovyInput.getValue())){
        LOG.warn("Groovy String Input equals null");
    } else {
        String content = groovyInput.getValue().toString();
        executeScript(content);
    }
}

} catch (RuntimeException e){
    Messagebox.show("Exception occurred while trying to execute Groovy script. Please check
your script!");
    LOG.warn("Exception occurred while trying to execute Groovy script");
}

}

@ViewEvent(componentID = "executeScript", eventName = Events.ON_CLICK)
public void executeScriptFromFileUpload() throws RuntimeException{

if (CollectionUtils.isEmpty(listbox.getItems())){
    LOG.warn("There is no uploaded file");
} else {

try {
    AttachmentFile attachmentFile = (AttachmentFile) listbox.getModel(); //cannot cast
listmodellist (listbox) to attachmentfile
    String file = attachmentFile.getUploadedfile().toString();
    BufferedReader reader = new BufferedReader(new FileReader(file));
    String allLines = reader.lines()
        .collect(Collectors.joining(System.lineSeparator()));
    //collectors joining adds all lines into one single string object. linesSeperator -> space
    reader.close();
    executeScript(allLines);
}

} catch (RuntimeException | IOException e) {
    Messagebox.show("Exception occurred while trying to execute Groovy script. Please check
your script!");
    LOG.warn("Execption occured while trying to execute Groovy Script from Upload");
}
}
}

public void executeScript(String content){
    scriptExecutable = ((ScriptingLanguagesService)
Registry.getApplicationContext().getBean("defaultScriptingLanguagesService"))
    .getExecutableByContent(new SimpleScriptContent("groovy", content));
}

```

```

scriptResult = scriptExecutable.execute();
Messagebox.show(String.valueOf(scriptResult.getScriptResult()));
LOG.info("Script was successful");
}

public ScriptingLanguagesService getScriptService() {
    return scriptService;
}

public void setScriptService(ScriptingLanguagesService scriptService) {
    this.scriptService = scriptService;
}

public ScriptExecutable getScriptExecutable() {
    return scriptExecutable;
}

public void setScriptExecutable(ScriptExecutable scriptExecutable) {
    this.scriptExecutable = scriptExecutable;
}

public ScriptContent getScriptContent() {
    return scriptContent;
}

public void setScriptContent(ScriptContent scriptContent) {
    this.scriptContent = scriptContent;
}

public ScriptExecutionResult getScriptResult() {
    return scriptResult;
}

public void setScriptResult(ScriptExecutionResult scriptResult) {
    this.scriptResult = scriptResult;
}

public Textbox getGroovyInput() {
    return groovyInput;
}

public void setGroovyInput(Textbox groovyInput) {
    this.groovyInput = groovyInput;
}

public Listbox getListbox() {
    return listbox;
}

```

```

public void setListbox(Listbox listbox) {
    this.listbox = listbox;
}

public AttachmentFile getAttachmentFile() {
    return attachmentFile;
}

public void setAttachmentFile(AttachmentFile attachmentFile) {
    this.attachmentFile = attachmentFile;
}

}
  
```

GroovyViewModel.java

```
package org.merkle.model;
```

```

import javax.activation.MimetypesFileTypeMap;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import org.apache.commons.io.FilenameUtils;
import org.apache.commons.io.input.ReaderInputStream;
import org.zkoss.bind.BindContext;
import org.zkoss.bind.BindUtils;
import org.zkoss.bind.annotation.AfterCompose;
import org.zkoss.bind.annotation.Command;
import org.zkoss.bind.annotation.ContextParam;
import org.zkoss.bind.annotation.ContextType;
import org.zkoss.util.media.Media;
import org.zkoss.zk.ui.Component;
import org.zkoss.zk.ui.event.UploadEvent;
import org.zkoss.zk.ui.select.Selectors;
  
```

```

public class GroovyFileModel {
    private List<AttachmentFile> attachmentFileList;
    private String dosPath;
    private String uploadString;
    private Long maxUploadedFileSize;
  
```

@AfterCompose

```

public void afterCompose(@ContextParam(ContextType.VIEW) Component view) {

    Selectors.wireComponents(view, this, false);
    maxUploadedFileSize = 2097152L;
    uploadString = "true,multiple=false,maxsize=" + maxUploadedFileSize;
}
  
```

```

@Command("onFileUpload")
public void onFileUpload(@ContextParam(ContextType.BIND_CONTEXT) BindContext ctx) {
    UploadEvent upEvent = null;
    Object objUploadEvent = ctx.getTriggerEvent();
    upEvent = (UploadEvent) objUploadEvent;
    if (upEvent != null) {
        attachedFile(upEvent.getMedias());
    }
}

public void attachedFile(Media[] mediaArray) {

    AttachmentFile attachmentFile = null;
    String subString = ".";

    if (mediaArray.length > 0) {
        for (Media media : mediaArray) {

            InputStream inputStream = null;
            attachmentFile = new AttachmentFile();

            try {
                inputStream = media.getStreamData();
            } catch (Exception e1) {
                try {
                    inputStream = new ReaderInputStream(media.getReaderData());
                } catch (Exception e) {
                }
            }

            if (inputStream != null) {
                String mediaFormat = FilenameUtilsgetExtension(media.getName().toLowerCase());

                /**
                 * String mediaContent = media.getContentType(); String mediaFormat =
                media.getFormat();
                */
                attachmentFile.setUploadedfile(inputStream);
                if (attachmentFile.getUploadFileSize() != null && attachmentFile.getUploadFileSize() > 0)
                {
                    attachmentFile.setFileName(media.getName() + " (" +
                    attachmentFile.getUploadFileSize() + ")");
                } else {
                    attachmentFile.setFileName(media.getName());
                }
            }
        }
    }
}

```

```

attachmentFile.setFileFormat(mediaFormat);
attachmentFile.setAttachmentFileSuffixUpload(mediaFormat);

attachmentFile.setBeforeExtension(media.getName().substring(0,
media.getName().lastIndexOf(subString)));

attachmentFile.setAfterExtension(media.getName().substring(media.getName().lastIndexOf(subStrin
g) + 1, media.getName().length()));

attachmentFile.setFileNameToUpload(new File(attachmentFile.getBeforeExtension() +
"." + attachmentFile.getAfterExtension()));

// In case of N/W save Path, file name will get the timestamp if file with the same name
already exists.

if (attachmentFileList == null) {
    attachmentFileList = new ArrayList<AttachmentFile>();
}

attachmentFileList.add(attachmentFile);
}
BindUtils.postNotifyChange(null, null, this, "attachmentFileList");
}
}
}

public Long getMaxUploadedFileSize() {
    return maxUploadedFileSize;
}

public void setMaxUploadedFileSize(Long maxUploadedFileSize) {
    this.maxUploadedFileSize = maxUploadedFileSize;
}

public List<AttachmentFile> getAttachmentFileList() {
    return attachmentFileList;
}

public void setAttachmentFileList(List<AttachmentFile> attachmentFileList) {
    this.attachmentFileList = attachmentFileList;
}

public String getDosPath() {
    return dosPath;
}

public void setDosPath(String dosPath) {
    this.dosPath = dosPath;
}

```

```

public String getUploadString() {
    return uploadString;
}

public void setUploadString(String uploadString) {
    this.uploadString = uploadString;
}

}
  
```

AttachmentFile.java

```

package org.merkle.model;

import java.io.File;
import java.io.InputStream;
import java.io.Serializable;

public class AttachmentFile {

    private String fileName;
    //comment from creator:
    // Made inputStream transient as they are machine dependent and cannot be
    // shared.

    //transient -> means variable doesnt really need getter and setter. removed transient to see if i can
    get getter and setter now
    private InputStream uploadedfile;
    private String beforeExtension;
    private String afterExtension;
    private File fileNameToUpload;
    private String attachmentContentTypeUpload;
    private String fileFormat;
    private Integer uploadFileSize;
    private String dosPath;
    private String attachmentFileSuffixUpload;
    private String uploadName;

    private String fileSavePath;
    private String folderType;

    public String getFolderType() {
        return folderType;
    }

    public void setFolderType(String folderType) {
  
```

```
    this.folderType = folderType;
}

public String getFileSavePath() {
    return fileSavePath;
}

public void setFileSavePath(String fileSavePath) {
    this.fileSavePath = fileSavePath;
}

public String getUploadName() {
    return uploadName;
}

public void setUploadName(String uploadName) {
    this.uploadName = uploadName;
}

public String getFileName() {
    return fileName;
}

public void setFileName(String fileName) {
    this.fileName = fileName;
}

public InputStream getUploadedfile() {
    return uploadedfile;
}

public void setUploadedfile(InputStream uploadedfile) {
    this.uploadedfile = uploadedfile;
}

public String getBeforeExtension() {
    return beforeExtension;
}

public void setBeforeExtension(String beforeExtension) {
    this.beforeExtension = beforeExtension;
}

public String getAfterExtension() {
    return afterExtension;
}

public void setAfterExtension(String afterExtension) {
    this.afterExtension = afterExtension;
}
```

```
}

public File getFileNameToUpload() {
    return fileNameToUpload;
}

public void setFileNameToUpload(File fileNameToUpload) {
    this.fileNameToUpload = fileNameToUpload;
}

public String getAttachmentContentTypeUpload() {
    return attachmentContentTypeUpload;
}

public void setAttachmentContentTypeUpload(String attachmentContentTypeUpload) {
    this.attachmentContentTypeUpload = attachmentContentTypeUpload;
}

public String getFileFormat() {
    return fileFormat;
}

public void setFileFormat(String fileFormat) {
    this.fileFormat = fileFormat;
}

public Integer getUploadFileSize() {
    return uploadFileSize;
}

public void setUploadFileSize(Integer uploadFileSize) {
    this.uploadFileSize = uploadFileSize;
}

public String getDosPath() {
    return dosPath;
}

public void setDosPath(String dosPath) {
    this.dosPath = dosPath;
}

public String getAttachmentFileSuffixUpload() {
    return attachmentFileSuffixUpload;
}

public void setAttachmentFileSuffixUpload(String attachmentFileSuffixUpload) {
    this.attachmentFileSuffixUpload = attachmentFileSuffixUpload;
}
```

```
}
```

```
}
```

definition.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<widget-definition id="org.merkle.widgets.groovykonsole"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.hybris.com/schema/cockpitng/widget-
definition.xsd">

  <name>Groovy Konsole</name>
  <description>Groovy Konsole für Skripte</description>
  <defaultTitle>Groovy Konsole</defaultTitle>
  <author>Tabea Bolliger</author>
  <version>1.0</version>
  <view src="groovyKonsole.zul"/>

  <controller class="org.merkle.controller.GroovyController"/>

</widget-definition>
```

groovyKonsole.zul

```
<widget xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.zkoss.org/2005/zul">
  viewModel="@id('vm') @init('org.merkle.model.GroovyFileModel')"

  <div>
    <hlayout>
      <div style="width:100%;">
        <hlayout id="GroovyWrapper" style="height: 250px; width: 100%;">
          <textbox rows="5" cols="40" id="groovyInput" style="width: 500px; height: 200px;
margin-top:40px;
margin-left:40px;" />
        </hlayout>
      </div>
      <button id="executeBtn" label="Execute"/>
    </hlayout>
  </div>

  <groupbox>
    <textbox width="75%" name="dosPath" tabindex="0"
      value="@bind(vm.dosPath)" maxlength="255" />
    <button id="btnUpload" label="Upload"
      onUpload="@command('onFileUpload')" upload="@load(vm.uploadString)" />
  </groupbox>
</widget>
```

```

mold="trendy" />

<div style="padding-left: 10px;padding-right: 10px;">
  <listbox id="listbox" model="@load(vm.attachmentFileList)"
    visible="@load(not empty vm.attachmentFileList)" sclass="listbox">
    <listhead>
      <listheader width="93%"></listheader>
      <listheader width="7%"></listheader>
    </listhead>
    <template name="model" var="var">
      <listitem>
        <listcell style="cursor:auto">

          <label tooltiptext="@load(var.fileName)"
            value="@load(var.fileName)" maxlength="65">
          </label>
        </listcell>

      </listitem>
    </template>
  </listbox>
</div>
</groupbox>

<button style="float:right; margin-right:600px;" id="executeScript">
  Execute
</button>

</widget>

```

MerkleExtension-backoffice-config.xml

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--
Copyright (c) 2020 SAP SE or an SAP affiliate company. All rights reserved
-->
<config xmlns="http://www.hybris.com/cockpit/config">
  <context principal="groovybackofficemanagergroup" component="perspective-view-switcher"
  module="MerkleExtension">
    <y:view-switcher xmlns:y="http://www.hybris.com/cockpitng/config/viewSwitcher">
      <y:authority name="groovybackofficemanager">
        <y:view id="groovy-perspective"/>
      </y:authority>
    </y:view-switcher>
  </context>

</config>

```

MerkleExtension-backoffice-widgets.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<widgets xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.hybris.com/schema/cockpitng/widgets.xsd">

  <widget-extension widgetId="backofficeMainSlot" name="MerkleExtension">
    <widget id="groovy-perspective" widgetDefinitionId="org.merkle.widgets.groovykonsole"
      slotId="perspectives"
        template="false" title="Merkle Extension" access="groovybackofficemanager">
        </widget>
    </widget-extension>

  </widgets>
```

local.properties

```
#Generated by hybris installer
#Thu Mar 28 10:11:56 CET 2024
\#mykey=\#myvalue
\#hac.webroot=/hac
googleApiKey=
website.electronics.http=http://electronics.local:9001/yacceleratorstorefront
website.electronics.https=https://electronics.local:9002/yacceleratorstorefront
website.apparel-de.http=http://apparel-de.local:9001/yacceleratorstorefront
website.apparel-de.https=https://apparel-de.local:9002/yacceleratorstorefront
website.apparel-uk.http=http://apparel-uk.local:9001/yacceleratorstorefront
website.apparel-uk.https=https://apparel-uk.local:9002/yacceleratorstorefront
website.powertools.http=http://powertools.local:9001/yb2bacceleratorstorefront
website.powertools.https=https://powertools.local:9002/yb2bacceleratorstorefront
yb2bacceleratorstorefront.illegalrequirementstest.excluded=true
occ.rewrite.overlapping.paths.enabled=true
backoffice.solr.search.index.autoinit=false
initialpassword.admin=nimda
media.default.storage.location.hash.salt=f51f428186194e87869d2f6d7309c7e4f
backoffice.cockpitng.additionalResourceLoader.enabled=true
backoffice.cockpitng.uifactory.cache.enabled=false
backoffice.cockpitng.widgetclassloader.resourcecache.enabled=false
backoffice.cockpitng.resourceloader.resourcecache.enabled=false
```

localextensions.xml

```
<hybrisconfig xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:noNamespaceSchemaLocation='..//bin/platform/resources/schemas/extensions.xsd'>
  <extensions>
    <path autoload="true" dir="${HYBRIS_BIN_DIR}/modules/cockpit-applications"/>
    <path dir="${HYBRIS_BIN_DIR}"/>
    <extension name='MerkleExtension'/>
  </extensions>
</hybrisconfig>
```

```
<extension name='adaptivesearch' />
<extension name='adaptivesearchbackoffice' />
<extension name='adaptivesearchfacades' />
<extension name='adaptivesearchsamplesaddon' />
<extension name='adaptivesearchsolr' />
<extension name='adaptivesearchwebservices' />
<extension name='apiregistrybackoffice' />
<extension name='apiregistryservices' />
<extension name='assistedservicefacades' />
<extension name='assistedservicepromotionaddon' />
<extension name='assistedservicepromotionfacades' />
<extension name='assistedserviceservices' />
<extension name='assistedservicestorefront' />
<extension name='assistedservicewebservices' />
<extension name='commerceorgsamplesaddon' />
<extension name='electronicsstore' />
<extension name='wishlist' />
<extension name='pcmbackofficesamplesaddon' />
<extension name='classificationgroupsservices' />
<extension name='pcmbackoffice' />
<extension name='backoffice' />
<extension name='platformbackoffice' />
<extension name='ybackoffice' />
<extension name='yacceleratorbackoffice' />
<extension name='yacceleratorcore' />
<extension name='yacceleratorfacades' />
<extension name='yacceleratorfulfilmentprocess' />
<extension name='yacceleratorinitialdata' />
<extension name='yacceleratorstorefront' />
<extension name='yaddon' />
<extension name='basecommerce' />
<extension name='basecommercebackoffice' />
<extension name='customerreview' />
<extension name='payment' />
<extension name='promotions' />
<extension name='promotionsbackoffice' />
<extension name='voucher' />
<extension name='voucherbackoffice' />
<extension name='profileservices' />
<extension name='chinesecommercewebservicescommons' />
<extension name='chineseproductsharingaddon' />
<extension name='chinesegetStoreAddon' />
<extension name='chineseStoreservices' />
<extension name='verticalnavigationaddon' />
<extension name='chineseAddressAddon' />
<extension name='chineseAddressBackoffice' />
<extension name='chineseAddressFacades' />
<extension name='chineseAddressOCC' />
<extension name='chineseAddressServices' />
```

```
<extension name='chineseppspalipaysamplesaddon' />
<extension name='chineseppspalipaysbackoffice' />
<extension name='chineseppspalipayservices' />
<extension name='chineselogisticaddon' />
<extension name='chineselogisticbackoffice' />
<extension name='chineselogisticfacades' />
<extension name='chineselogisticocc' />
<extension name='chineselogisticservices' />
<extension name='chineseppaymentaddon' />
<extension name='chineseppaymentfacades' />
<extension name='chineseppaymentmock' />
<extension name='chineseppaymentocc' />
<extension name='chineseppaymentservices' />
<extension name='chinesepprofileaddon' />
<extension name='chinesepprofilefacades' />
<extension name='chinesepprofileocc' />
<extension name='chinesepprofileservices' />
<extension name='chinesetaxinvoiceaddon' />
<extension name='chinesetaxinvoicebackoffice' />
<extension name='chinesetaxinvocefacades' />
<extension name='chinesetaxinvoiceocc' />
<extension name='chinesetaxinvoiceservices' />
<extension name='chineseppspwechatpaymentaddon' />
<extension name='chineseppspwechatpaysamplesaddon' />
<extension name='chineseppspwechatpaysbackoffice' />
<extension name='chineseppspwechatpayservices' />
<extension name='cockpit' />
<extension name='commercefacades' />
<extension name='commerceservices' />
<extension name='commerceservicesbackoffice' />
<extension name='commercwebservices' />
<extension name='commercwebservicescommons' />
<extension name='commercwebservicestests' />
<extension name='consignmenttrackingaddon' />
<extension name='consignmenttrackingbackoffice' />
<extension name='consignmenttrackingfacades' />
<extension name='consignmenttrackingmock' />
<extension name='consignmenttrackingocc' />
<extension name='consignmenttrackingservices' />
<extension name='acceleratorbackoffice' />
<extension name='acceleratorcms' />
<extension name='acceleratorfacades' />
<extension name='acceleratorocc' />
<extension name='acceleratorservices' />
<extension name='acceleratorstorefrontcommons' />
<extension name='addonsupport' />
<extension name='couponbackoffice' />
<extension name='couponfacades' />
<extension name='couponservices' />
```

```

<extension name='couponwebservices' />
<extension name='customercouponaddon' />
<extension name='customercouponbackoffice' />
<extension name='customercouponfacades' />
<extension name='customercouponocc' />
<extension name='customercouponsamplesaddon' />
<extension name='customercouponservices' />
<extension name='customerinterestsaddon' />
<extension name='customerinterestsfacades' />
<extension name='customerinterestsocc' />
<extension name='customerinterestsservices' />
<extension name='customersupportbackoffice' />
<extension name='customerticketingaddon' />
<extension name='customerticketingfacades' />
<extension name='customerticketingocc' />
<extension name='customerticketingocctests' />
<extension name='ticketsystem' />
<extension name='ticketsystembackoffice' />
<extension name='inboundservices' />
<extension name='integrationbackoffice' />
<extension name='integrationbackofficetest' />
<extension name='integrationmonitoringbackoffice' />
<extension name='integrationservices' />
<extension name='odata2services' />
<extension name='odata2webservices' />
<extension name='odata2webservicesfeaturetests' />
<extension name='outboundservices' />
<extension name='outboundsync' />
<extension name='outboundsyncbackoffice' />
<extension name='webhookbackoffice' />
<extension name='webhookservices' />
<extension name='marketplaceaddon' />
<extension name='marketplacebackoffice' />
<extension name='marketplacebackofficesamplesaddon' />
<extension name='marketplacefacades' />
<extension name='marketplaceocc' />
<extension name='marketplacepromotionenginesamplesaddon' />
<extension name='marketplaceservices' />
<extension name='marketplacestore' />
<extension name='yacceleratormarketplaceintegration' />
<extension name='messagecentercsfacades' />
<extension name='messagecentercsocc' />
<extension name='messagecentercsservices' />
<extension name='notificationaddon' />
<extension name='notificationfacades' />
<extension name='notificationocc' />
<extension name='notificationservices' />
<extension name='npmancillary' />
<extension name='ordermanagementfacades' />

```

```
<extension name='orderselfserviceaddon' />
<extension name='personalizationcms' />
<extension name='personalizationcmsbackoffice' />
<extension name='personalizationcmsweb' />
<extension name='personalizationfacades' />
<extension name='personalizationsampleddataaddon' />
<extension name='personalizationservices' />
<extension name='personalizationservicesbackoffice' />
<extension name='personalizationsmartedit' />
<extension name='personalizationwebservices' />
<extension name='previewpersonalizationweb' />
<extension name='personalizationpromotions' />
<extension name='personalizationpromotionspromotionsbackoffice' />
<extension name='personalizationpromotionssampleddataaddon' />
<extension name='personalizationpromotionssmartedit' />
<extension name='personalizationpromotionsweb' />
<extension name='personalizationsearch' />
<extension name='personalizationsearchbackoffice' />
<extension name='personalizationsearchsamplesaddon' />
<extension name='personalizationsearchsmartedit' />
<extension name='personalizationsearchweb' />
<extension name='adminapi' />
<extension name='auditreportservices' />
<extension name='deltadetection' />
<extension name='embeddedserver' />
<extension name='groovynature' />
<extension name='springintegrationlibs' />
<extension name='tomcatembeddedserver' />
<extension name='promotionenginebackoffice' />
<extension name='promotionenginesamplesaddon' />
<extension name='promotionengineservices' />
<extension name='droolsruleengineservices' />
<extension name='ordercalculation' />
<extension name='rulebuilderbackoffice' />
<extension name='ruledefinitions' />
<extension name='ruleengine' />
<extension name='ruleenginebackoffice' />
<extension name='ruleengineservices' />
<extension name='kymaintegrationbackoffice' />
<extension name='kymaintegrationservices' />
<extension name='backofficesolrsearch' />
<extension name='solrfacetsearch' />
<extension name='solrfacetsearchbackoffice' />
<extension name='solrserver' />
<extension name='backofficesearchservices' />
<extension name='searchbackoffice' />
<extension name='searchprovidercssearchbackoffice' />
<extension name='searchprovidercssearchservices' />
<extension name='searchservices' />
```

```

<extension name='selectivecartaddon' />
<extension name='selectivecartfacades' />
<extension name='selectivecartservices' />
<extension name='cmssmartedit' />
<extension name='cmssmarteditwebservices' />
<extension name='smartedit' />
<extension name='smarteditaddon' />
<extension name='smartedittools' />
<extension name='smarteditwebservices' />
<extension name='stocknotificationaddon' />
<extension name='stocknotificationfacades' />
<extension name='stocknotificationservices' />
<extension name='textfieldconfiguratortemplateaddon' />
<extension name='textfieldconfiguratortemplatebackoffice' />
<extension name='textfieldconfiguratortemplatefacades' />
<extension name='textfieldconfiguratortemplateservices' />
<extension name='timedaccesspromotionengineaddon' />
<extension name='timedaccesspromotionenginebackoffice' />
<extension name='timedaccesspromotionenginefacades' />
<extension name='timedaccesspromotionengineocc' />
<extension name='timedaccesspromotionenginesamplesaddon' />
<extension name='timedaccesspromotionengineservices' />
<extension name='cms2' />
<extension name='cms2lib' />
<extension name='cmsbackoffice' />
<extension name='cmsfacades' />
<extension name='cmsocc' />
<extension name='cmswebservices' />
<extension name='cmscockpit' />
<extension name='previewwebservices' />
<extension name='permissionsfacades' />
<extension name='permissionswebservices' />
<extension name='webservicescommons' />
</extensions>
</hybrisconfig>

```

GroovyControllerTest.java

```

package org.merkle;

import de.hybris.bootstrap.annotations.UnitTest;
import de.hybris.platform.core.Registry;
import de.hybris.platform.scripting.engine.ScriptExecutable;
import de.hybris.platform.scripting.engine.ScriptExecutionResult;
import de.hybris.platform.scripting.engine.ScriptingLanguagesService;
import de.hybris.platform.scripting.engine.content.ScriptContent;
import org.junit.Before;
import org.junit.Test;
import static org.mockito.Mockito.*;

```

```
import org.merkle.controller.GroovyController;
import org.merkle.model.AttachmentFile;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
import org.zkoss.zul.ListModelList;
import org.zkoss.zul.Listbox;
import org.zkoss.zul.Messagebox;
import org.zkoss.zul.Textbox;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.util.stream.Stream;

@UnitTest
public class GroovyControllerTest {

    @Mock
    Textbox groovyInput;

    @Mock
    Listbox listbox;

    @Mock
    private ScriptingLanguagesService scriptingLanguagesServiceMock;

    @Mock
    private ScriptExecutable scriptExecutableMock;

    @Mock
    private ScriptExecutionResult scriptResultMock;

    @Mock
    private AttachmentFile attachmentFile;

    @Mock
    private BufferedReader reader;

    @Mock
    private ListModelList<AttachmentFile> listModelList;

    @Mock
    private Messagebox messageboxMock;

    @InjectMocks
    private GroovyController groovyController;

    @Before
}
```

```
public void setUp(){
    MockitoAnnotations.initMocks(this);
}

@Test
public void testExecuteFromString() throws RuntimeException{

when(scriptingLanguagesServiceMock.getExecutableByContent(any(ScriptContent.class))).thenReturn(scriptExecutableMock);
    when(scriptExecutableMock.execute()).thenReturn(scriptResultMock);

    String testcontent = "return \"hi\"";
    when(groovyInput.getValue()).thenReturn(testcontent);
    groovyController.executeScriptFromString();

    verify(groovyController).executeScript("return \"hi\"");
}

}

@Test
public void testExecuteFromFileUpload() throws RuntimeException, IOException {

    when(String.valueOf(attachmentFile.getUploadedfile())).thenReturn(String.valueOf(new File("test.groovy")));
    when(reader.lines()).thenReturn(Stream.of("return \"Groovy Rocks!\""));

    groovyController.executeScriptFromFileUpload();

    verify(groovyController).executeScript("return \"Groovy Rocks!\"");
    verify(reader).close();

}

}

}
```

git commits

```
commit 1cbd365de5ec7b9b7a055e716ff00aff3084486f
Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>
Date: Mon Apr 8 13:09:54 2024 +0200
```

Code clean up

```
commit 6a08300e6f719efec5ed2283747bc0d9edb128d
Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>
```

Date: Thu Apr 4 16:12:58 2024 +0200

Arbeitspaket Testing: Testing durchgeführt.

commit bf9816045d970ea820324ca18b14b3dc0451b7a8

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Wed Apr 3 19:21:43 2024 +0200

Arbeitspaket Skript Uploader: Skript Uploader Logik angepasst

commit 207da429a56267074555d703b531ad98e7e31a0e

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Wed Apr 3 19:20:37 2024 +0200

Arbeitspaket Testumgebung: Testing Klasse erstellt

commit e44e8ed3d2447a113e3a255ea2768db35f01960a

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Wed Apr 3 19:19:49 2024 +0200

Arbeitspaket Benutzergruppe mit Impex: Impex aktualisiert

commit 08a06b4aa086345db03430a5fb8df7f6f12318a8

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Wed Apr 3 12:45:22 2024 +0200

LOGs hinzugefügt

commit 42b8938dfd6ec1d75e7abf8bc91db37036033525

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Tue Apr 2 18:40:34 2024 +0200

Arbeitspaket Groovy Konsole: Fehler behoben und abgeschlossen

commit aadb62210691ceace5e800b846ee53eb81633b2c

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Tue Apr 2 18:26:37 2024 +0200

Arbeitspaket Skript Uploader: Logik für den Skript Uploader aktualisiert. Drag and Drop option entfernt.

commit 8f66972d40c9d9120e5c35d4ad74fa23e5a4be83

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Tue Apr 2 09:38:49 2024 +0200

Arbeitspaket Groovy Konsole Logik & Skript Uploader: weiter gearbeitet an der Logik für die Konsole und den Skript Uploader

commit ee28b4806d774ede33625e7576851ca26fcab002

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Thu Mar 28 16:29:11 2024 +0100

Arbeitspaket Groovy Konsole Logik & Skript Uploader Logik: drag and drop für dateien ist jetzt möglich.

commit 88c0799de60841a65513fa825bd7b74900d09c49

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Thu Mar 28 12:35:02 2024 +0100

Arbeitspaket Logik für Groovy Konsole: an der logik weiter gearbeitet und backoffice problem gelöst

commit b282adf7681421b8950e851e819a88643f68739f

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Thu Mar 28 08:16:03 2024 +0100

Arbeitspaket Groovy Konsole Logik: an der logik der konsole weitergearbeitet

commit e5758fe32ddc24fd954873d4959f7a9492224ada

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Wed Mar 27 18:39:25 2024 +0100

Arbeitspaket Groovy Konsole Logik: mit der Arbeit im GroovyController weiter gearbeitet, noch nicht fertig.

commit 0dc0070b484b11c2e4447cf1e54603959e5c068

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Wed Mar 27 12:47:56 2024 +0100

Arbeitspakte Groovy Konsole Design und Groovy Konsole Logik: design für skript eingabe fertig, logik für die skript eingabe angefangen

commit f45ec86d3b3af26bbc92c807fdabee8b4c9ba9786

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Wed Mar 27 12:45:36 2024 +0100

Arbeitspaket Benutzerberechtigung Perspektive : config aktualisiert

commit 5e3f226f0739b4c034591e57c2b278fb4ef69be1

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Mon Mar 25 17:49:33 2024 +0100

Arbeitspakete Benutzerberechtigung Perspektive und Groovy Konsole Design : weiter gearbeitet an Perspektive Berechtigung und Konsole Design

commit 6996ea03362c53643457362fef1140fca958bb0c

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Mon Mar 25 12:03:20 2024 +0100

Arbeitspaket Benutzerberechtigung Perspektive : Neue Benutzergruppe groovyAdmin erstellt und benutzerberechtigung angefangen

commit 7a9845a1ea26de97d7adbc51a0f300974f69bfd4

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Wed Mar 20 17:11:41 2024 +0100

Arbeitspaket Perspektive: Widgets hinzugefügt

commit cf76185e7f26c108ba068061f6f4402c920273a3

Author: Tabea Bolliger <tabea.bolliger@emea.merkleinc.com>

Date: Wed Mar 20 15:26:33 2024 +0100

Arbeitspaket Extension: Neue Extension erstell

console-20240408.log

Diese Datei werde ich als separaten Anhang hochladen.