Erweiterung SAP Backoffice um Groovy Perspektive

Tabea Bolliger

16.04.2024

MERKLE

Inhaltsverzeichnis

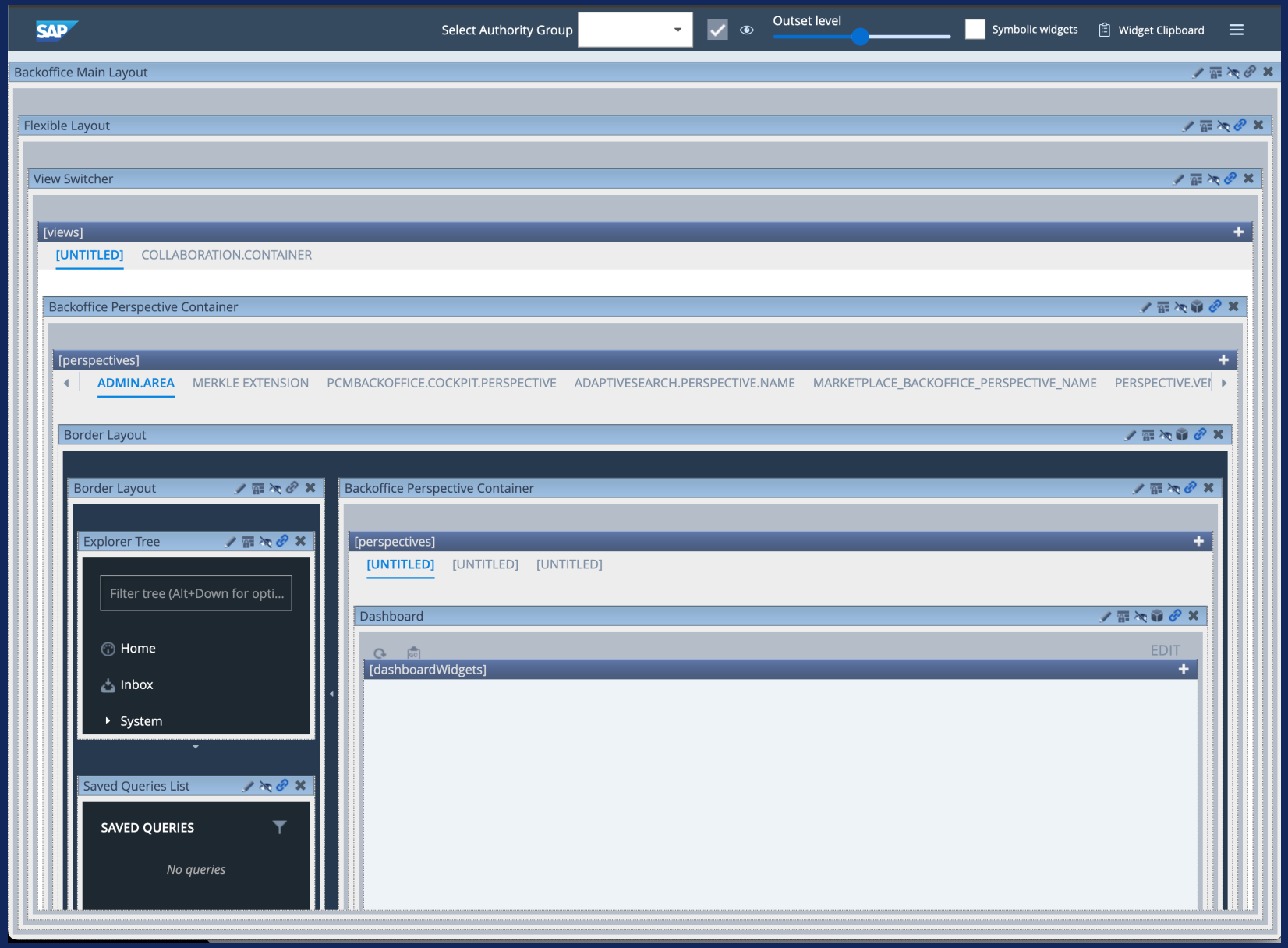
Überblick

Ausgangslage

- Das Backoffice Modul und Administrationsansicht
- Groovy Skripte ausführen
- Mühsam und Zeitaufwändig

Das SAP Backoffice

- Ansichten
- Orchestrator Modus
- cockpit-config.xml



Ziel der Arbeit

- Neue Ansicht generieren
- Neue Benutzergruppe und neuer Benutzer
- Skripte selber schreiben oder Skripte hochladen


IPERKA
Stage 6
Stage 5
Stage 4
Stage 3
Stage 2
Stage 1
START
01
02
03
04
05
06

IPERKA

Informieren

Planen

Entscheiden

Realisieren

Kontrollieren

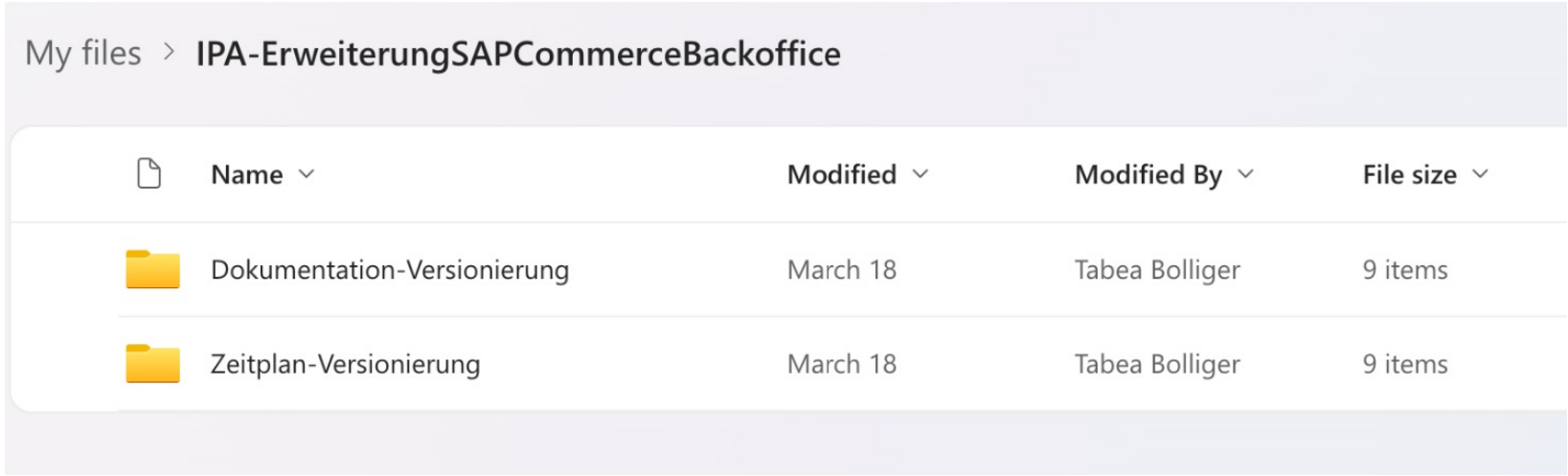
Auswerten

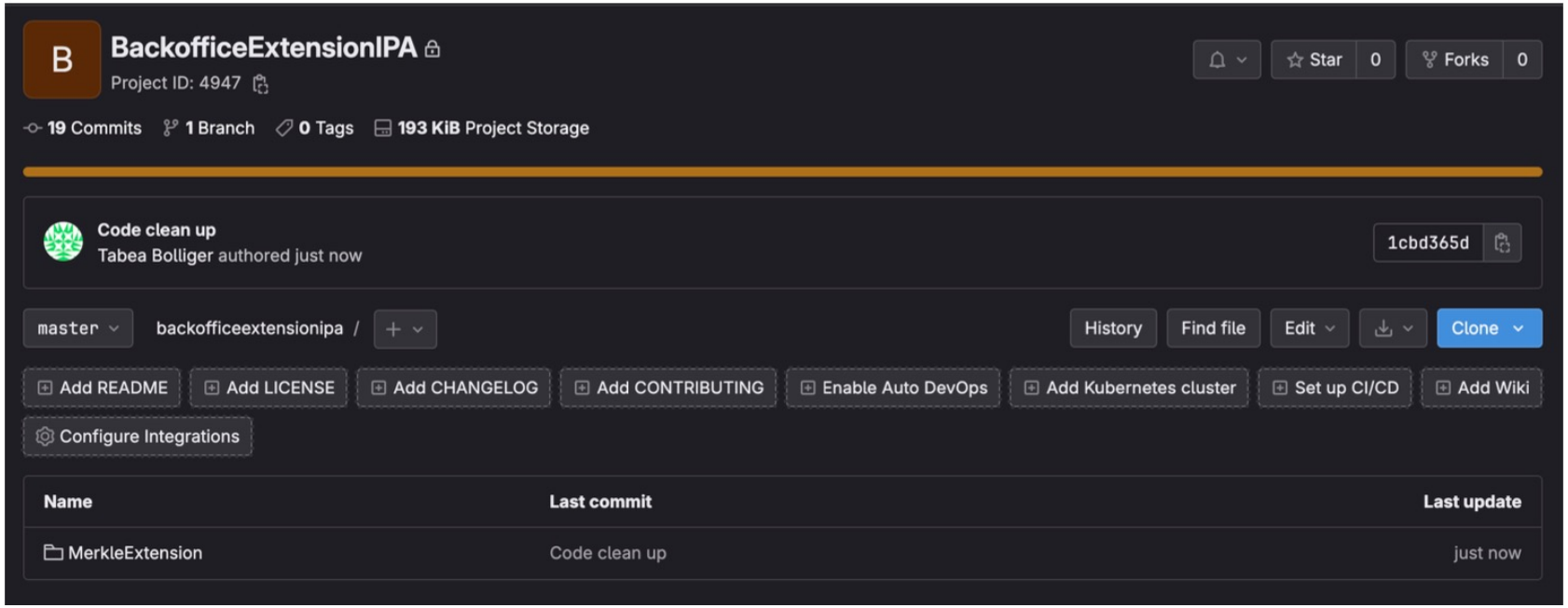
Planen

- Verwaltung meiner Dateien
- Anwendungsfalldiagramme
- Aktivitätsdiagramme
- Systemdiagramme

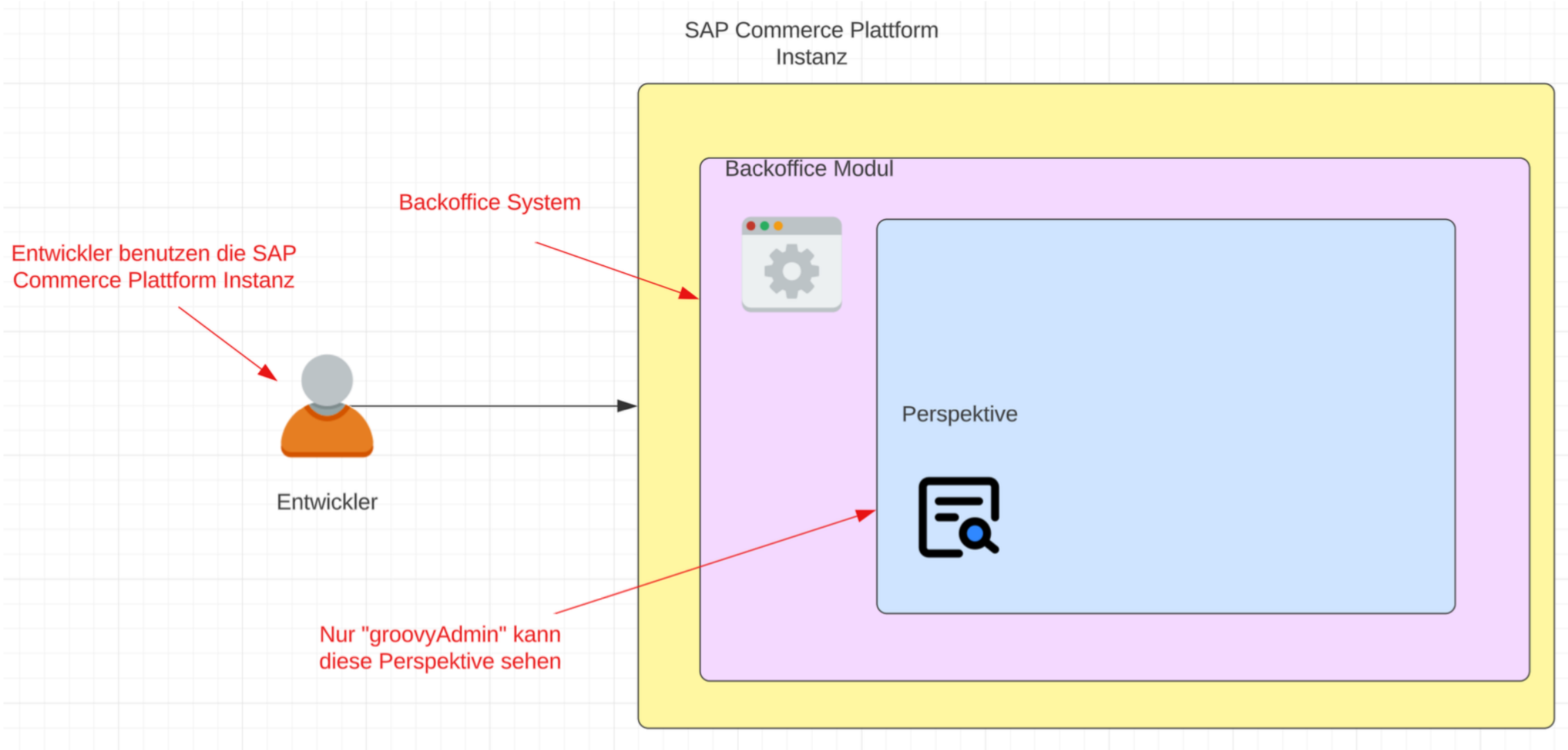
Verwaltung meiner Dateien

- OneDrive
- GitLab





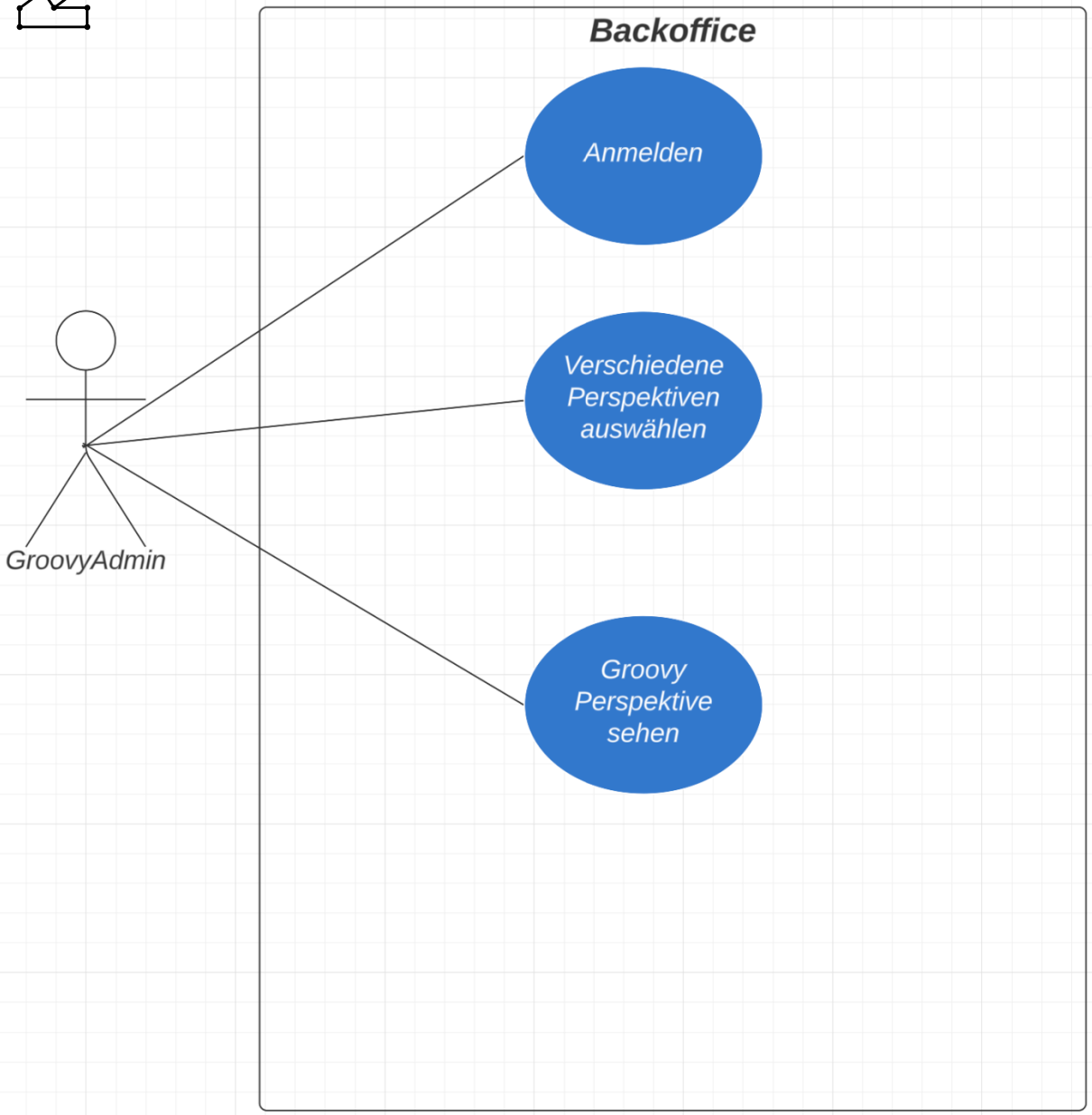
Systemdiagramm



Use Case

Anwendungsfalldiagramm

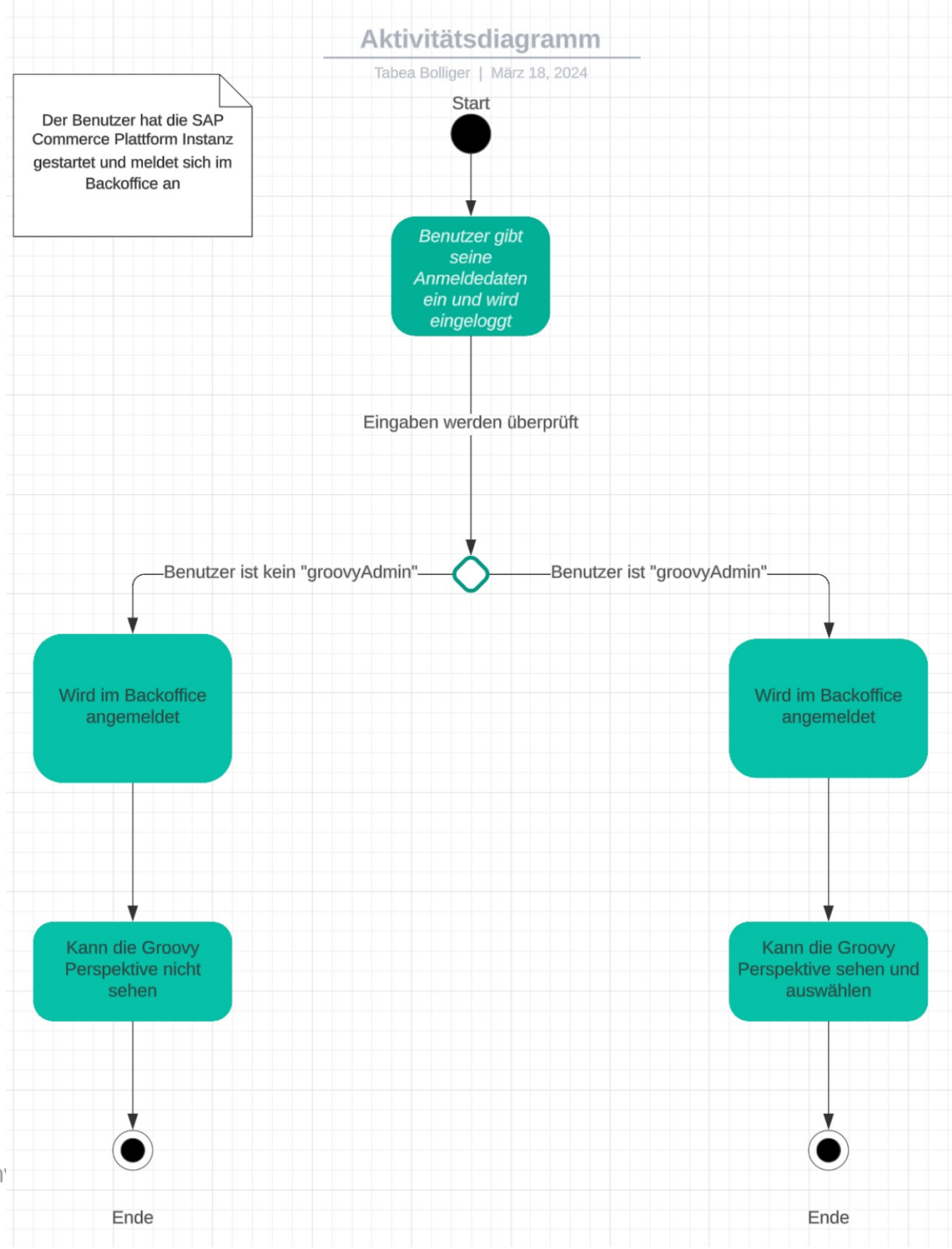
Tabea Bolliger | März 18 2024



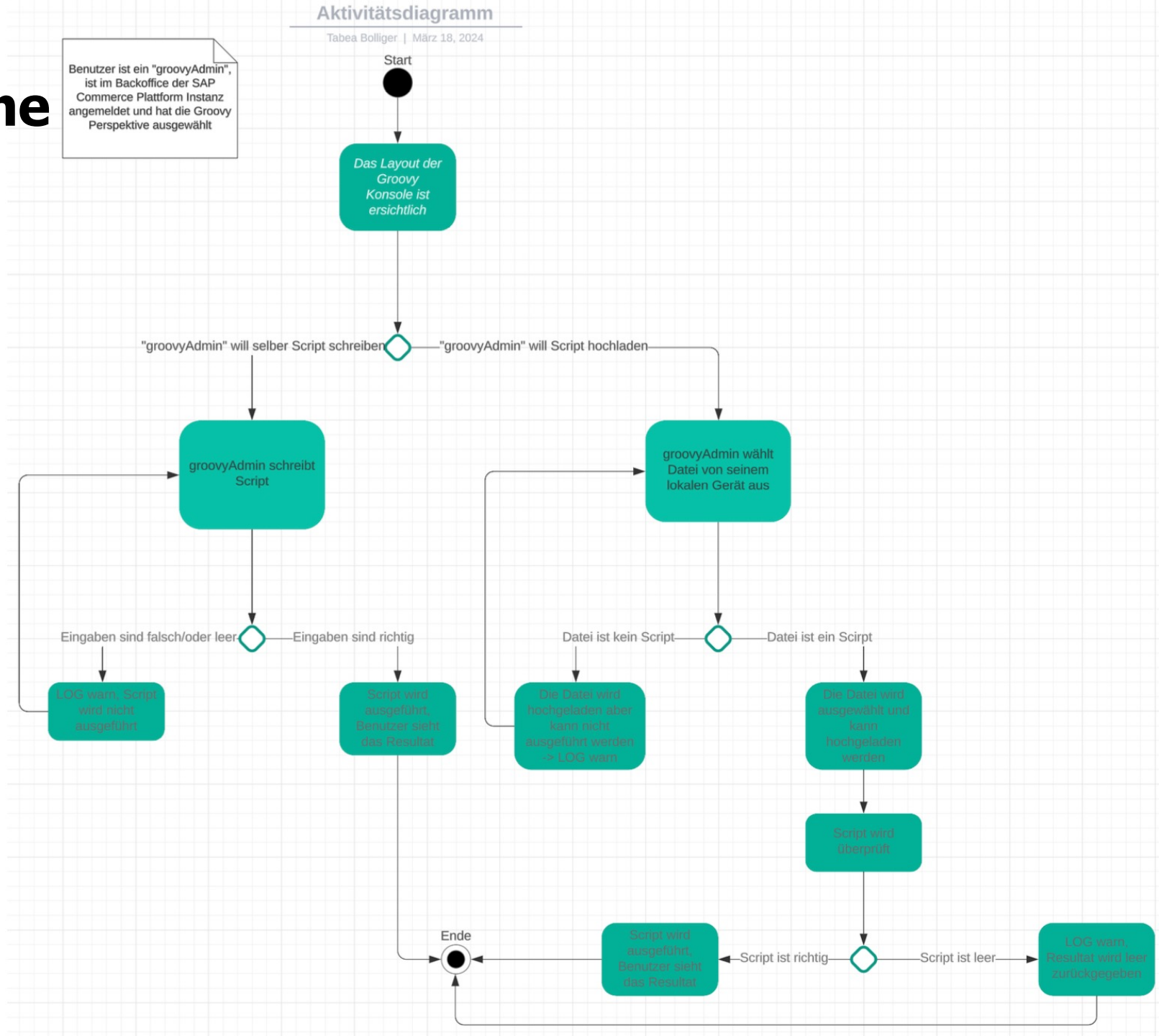
Anwendungsfalldiagramm

Tabea Bolliger | März 18 2024

Aktivitätsdiagramme



Aktivitätsdiagramme



IPERKA

Informieren

Planen

Entscheiden

Realisieren

Kontrollieren

Auswerten

Entscheiden

Werkzeuge für meine Arbeit

Werkzeuge für meine Arbeit

- Lucidchart

Werkzeuge für meine Arbeit

- SAP Commerce Plattform
- Ant extgen
- Scripting Languages Service

Werkzeuge für meine Arbeit

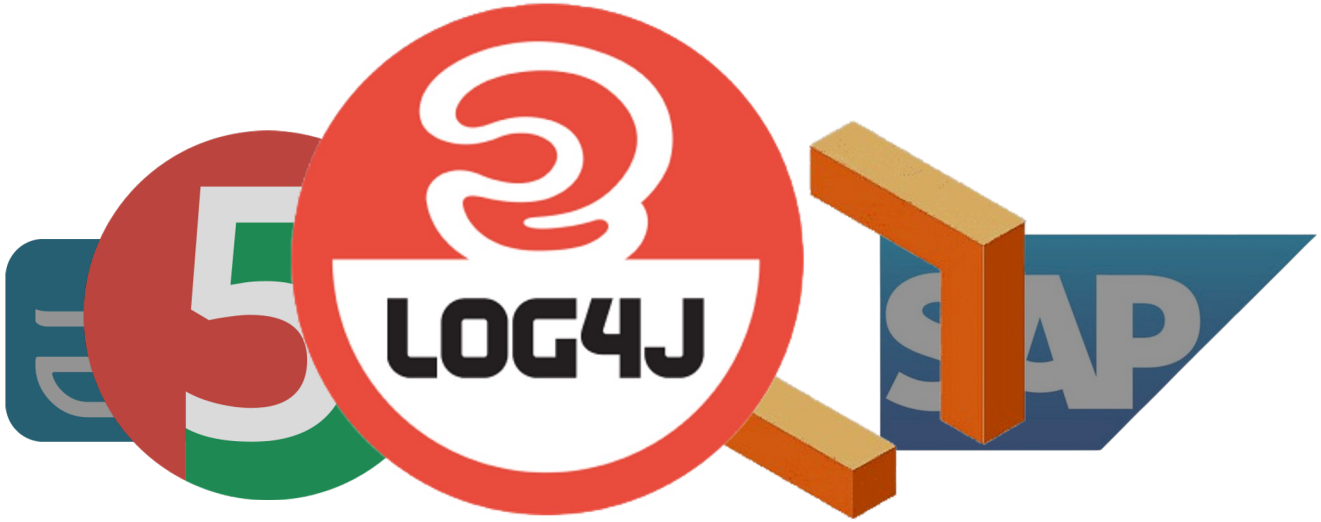
- ZK

Werkzeuge für meine Arbeit

- JUnit

Werkzeuge für meine Arbeit

- Log4j



IPERKA

Informieren

Planen

Entscheiden

Realisieren

Kontrollieren

Auswerten

Realisieren

- Benutzer
- Extensions und Perspektiven
- Widgets
- GroovyController
- GroovyViewModel
- Attachmentfile

Benutzer

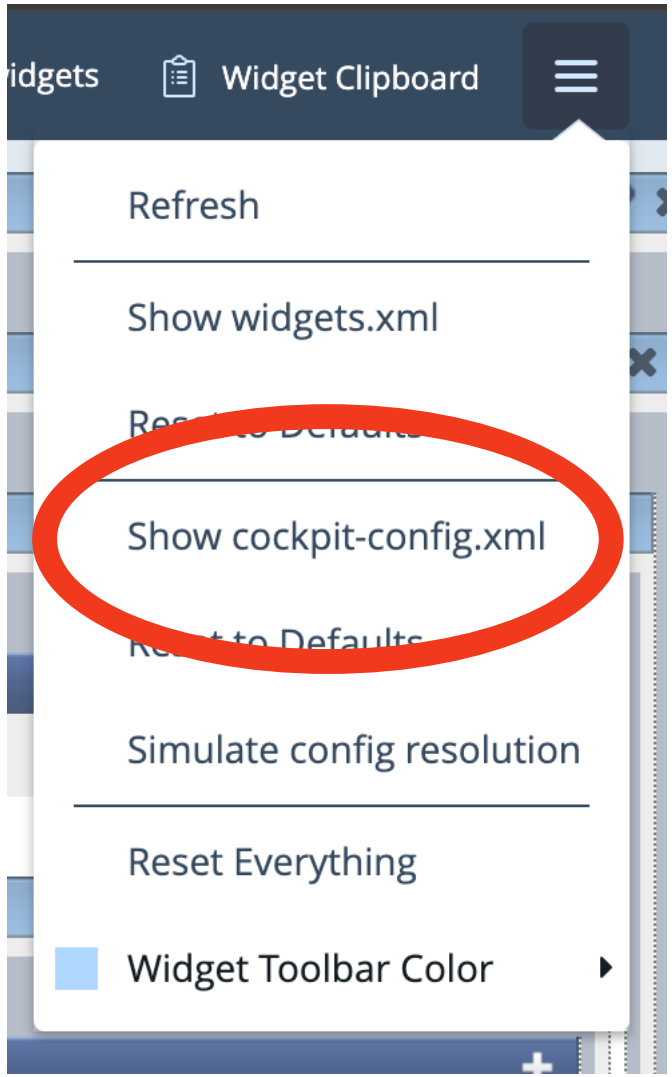
Backoffice Rollen

Benutzer

Kunden

Mitarbeiter

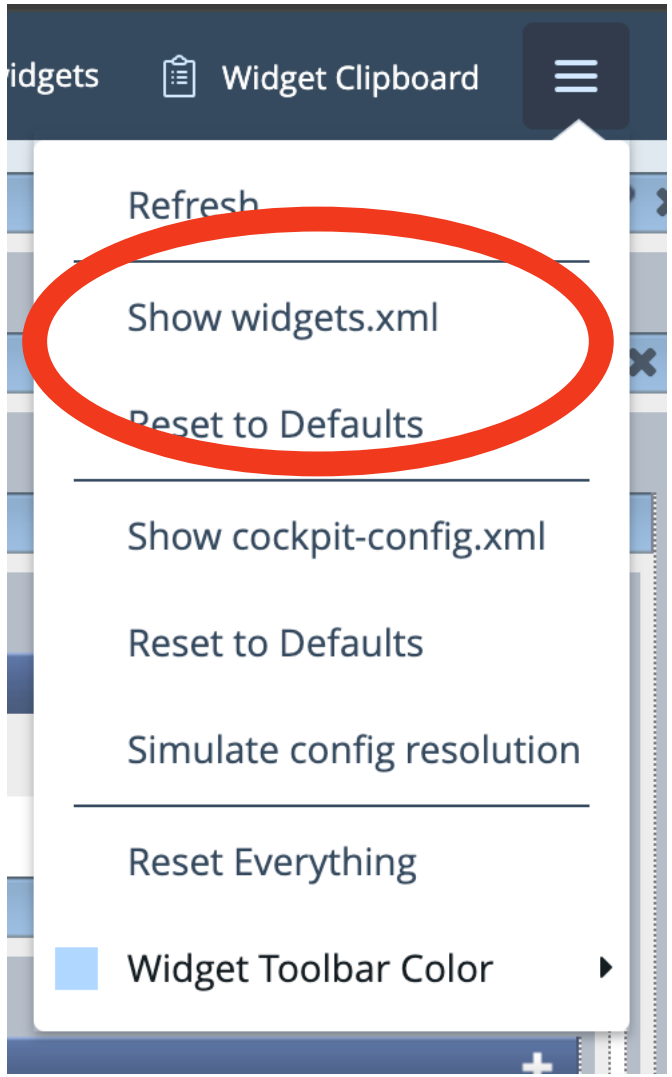
Extensions und Perspektiven




```
<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>
```

Extensions und Perspektiven




```
<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>
```

Widgets

```
<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>
```

Widgets

```
<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>
```

Widgets

```
<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.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>
```

GroovyController

- Skript durch Eingabe
- Skript durch Datei
- Skript Ausführung

```
private Textbox groovyInput;
private Listbox listbox;
private AttachmentFile attachmentFile;

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

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

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

public void executeScript(String content){...}
```

GroovyViewModel

- ZKFiddle
- onUpload
- attachedFile

```
private List<AttachmentFile> attachmentFileList;
private String dosPath;
private String uploadString;
private Long maxUploadedFileSize;


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

@Command("onFileUpload")
public void onFileUpload(@ContextParam(ContextType.BIND_CONTEXT) BindContext ctx) {...}

public void attachedFile(Media[] mediaArray) {...}
```

AttachmentFile

- ZKFiddle
- Getter und Setter

```
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;
```

IPERKA

Informieren

Planen

Entscheiden

Realisieren

Kontrollieren

Auswerten

Kontrollieren

Testing in der GroovyControllerTest Klasse

GroovyControllerTest

- Mocking
- Test Level

GroovyControllerTest

```
@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( content: "return '\"hi\"'");

}
```

GroovyControllerTest

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

    when(String.valueOf(attachmentFile.getUploadedfile())).thenReturn(String.valueOf(new File( pathname: "test.groovy")));
    when(reader.lines()).thenReturn(Stream.of( t: "return '\"Groovy Rocks!\"'"));

    groovyController.executeScriptFromFileUpload();

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



}
```

IPERKA

Informieren

Planen

Entscheiden

Realisieren

Kontrollieren

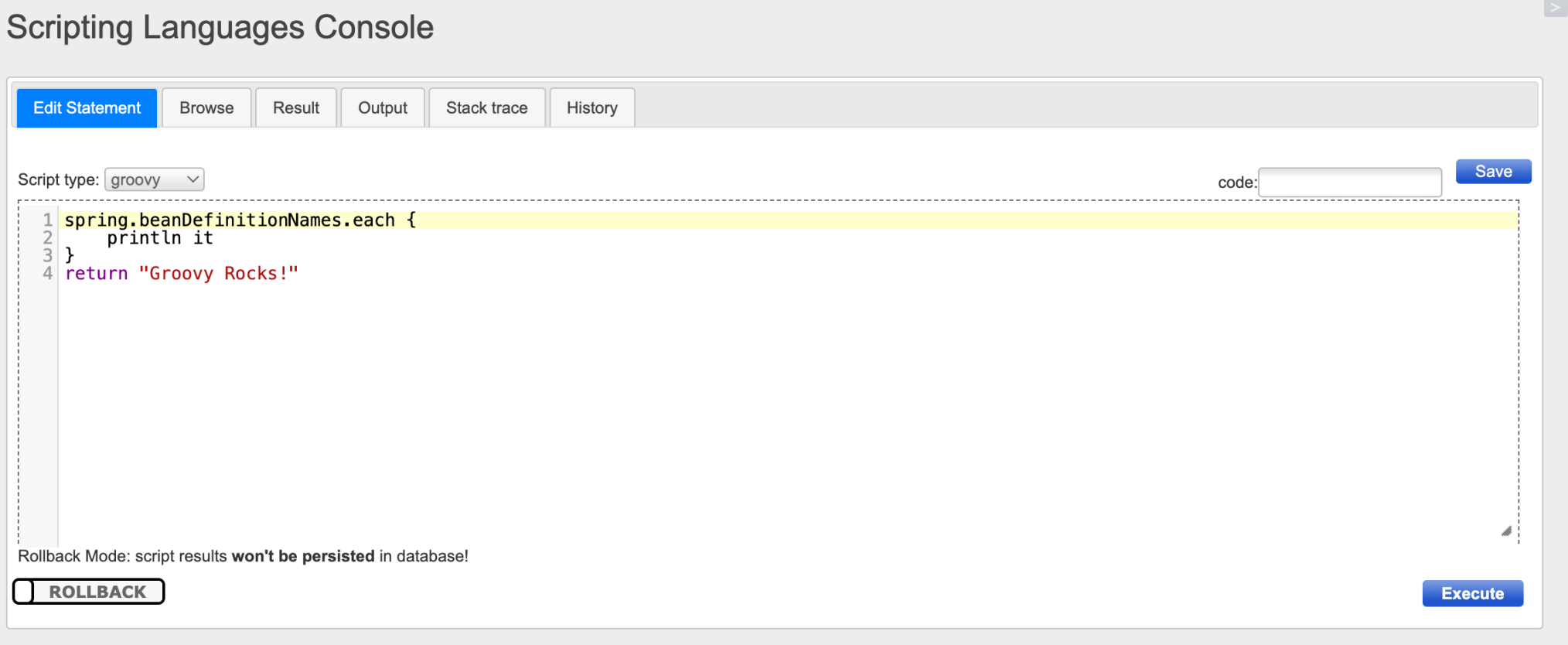
Auswerten

Auswerten

- Vergleich
- LOGGER Resultat in der Konsole
- Fehlerhafte Anforderungen
- Fazit

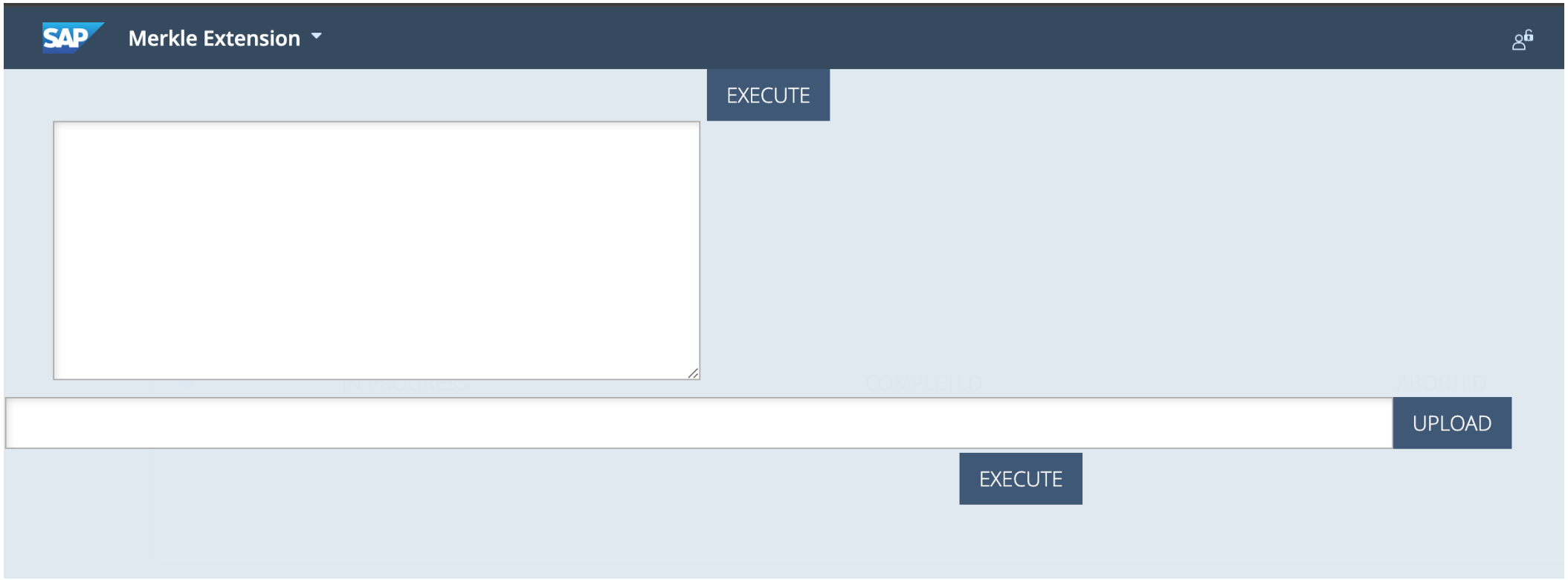
Vergleich

HAC



Vergleich

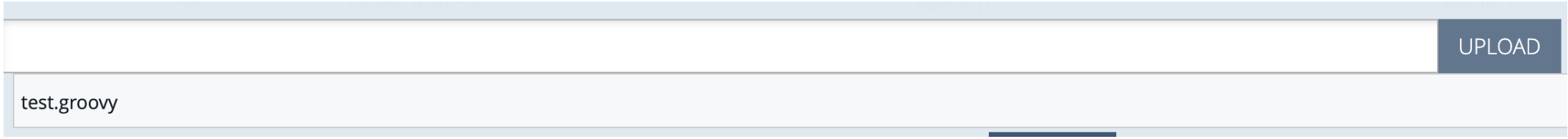
Backoffice



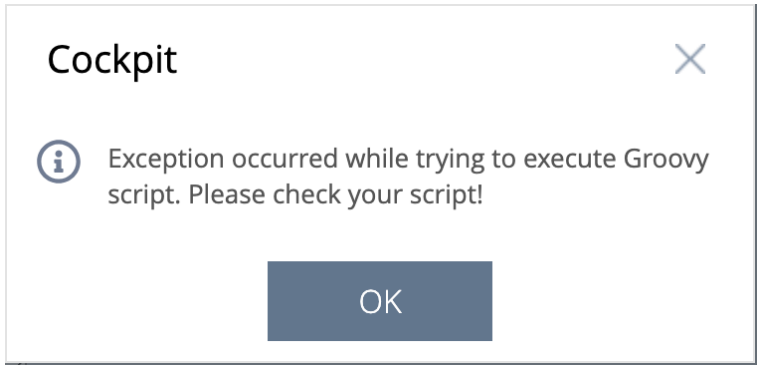
LOG Resultat in der Konsole

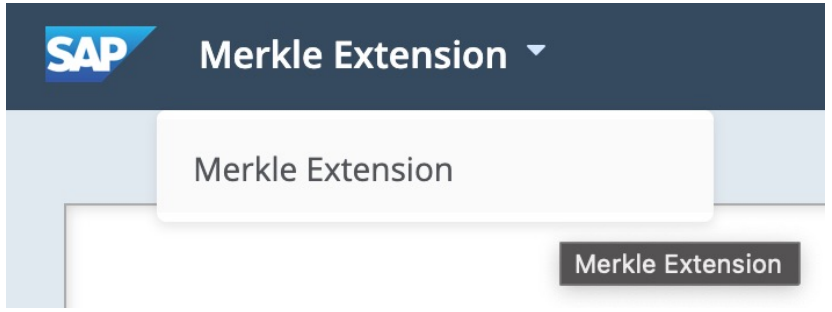
```
INFO  [hybrisHTTP40] [MERKLE_GROOVY_CONSOLE_LOGGER] Input is being checked
INFO  [hybrisHTTP40] [MERKLE_GROOVY_CONSOLE_LOGGER] Script was successful
```

```
WARN  [hybrisHTTP11] [MERKLE_GROOVY_CONSOLE_LOGGER] Execption occured while trying to execute Groovy Script from Upload
```



Fehlerhafte Anforderungen





Fazit

- Dokumentation
- Zeitplan
- Testing
- Design

DANKE FÜR EURE AUFMERKSAMKEIT!

Tabea Bolliger
Lernende Informatiker/in Applikationsentwicklung EFZ
tabea.bolliger@merkle.com

Demonstration der neuen Backoffice Erweiterung