Implementierung MatFlow Workflowanwendung für Machine Learning Experimente

Florian Küfner, Soeren Raymond, Alessandro Santospirito,
Lukas Wilhelm, Nils Wolters

Februar 2021

Inhaltsverzeichnis

1 Einleitung **3**

2 Aufteilung **4**
2.1 Geplante Aufteilung 4
2.2 Endgültiger Stand nach Implementierung 4

3 Entwurfsumentscheidungen **6**
3.1 Memento 6
3.2 JSON Converter 7
3.3 JSON Status Code 9
3.4 Authentifizierung 9
3.5 keys 9
3.6 Entfernen der Login und SignUp Klasse 9
3.7 hinzufügen der Schnittstelle zur Datenbank 9
3.8 Kommunikation von Dateien innerhalb der Serveranwendung 9
3.9 Änderungen im Workflow-Package 11
3.10 Database Paket Implementierungsänderungen 12
3.11 MySQL - Datenbankaufbau 13

4 Anhang **19**
4.1 Detailliertere Aufteilung 19

1 Einleitung

Diese Implementierungsheft stellt die Fortschritte der Implementierungsphase und die darin getroffenen Entwurfsänderungen und -entscheidungen dar.

2 Aufteilung

2.1 Geplante Aufteilung

Im Folgenden ist eine Grafik zu sehen, die einen Überblick über die Implementierung des MatFlow Systems gibt. Eine detailliertere Ansicht über die Aufgabenteile findet man im Anhang.

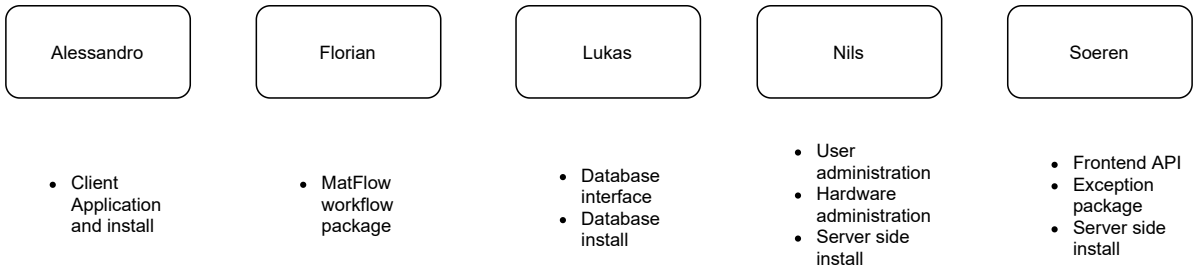

Abbildung 1: Aufteilung

2.2 Endgültiger Stand nach Implementierung

Frontend

- **Jede** der anzuwählenden Seiten ist implementiert
- Alle benötigten Datenklassen sind voll funktionsfähig implementiert

Frotend API

- **Jede** der zur Verfügung stehenden Methoden ist bis auf die keys voll funktionsfähig implementiert
- Alle Unit Tests sind geschrieben

User Administration

- **Jede** der zur Verfügung stehenden Methoden ist voll funktionsfähig implementiert
- Alle Unit Tests sind geschrieben

Hardware Administration

- **Vile** der zur Verfügung stehenden Methoden ist voll funktionsfähig implementiert

Workflow Package

- **Jede** der zur Verfügung stehenden Methoden ist voll funktionsfähig implementiert

- Alle Unit Tests sind geschrieben

Exception package

- **Jede** der zur Verfügung stehenden Methoden ist voll funktionsfähig implementiert
- Alle unittests sind geschrieben.

Database package

- Der Großteil der zur Verfügung stehenden Methoden ist voll funktionsfähig implementiert

Zu implementieren bleibt noch

- Die request Klasse BackendCommunicator und Unit Tests im Frontend
- Die WorkflowData Schnittstelle im Database package
- Der alternierende Operator für airflow
- Die containerweise Ausführung der Serverapplikation im Backend (Nomad Job File)

3 Entwurfsumentscheidungen

3.1 Memento

The memento pattern is used to set a view to the default state. Every view has its default state that is created at the start of the view. When a user makes an input, the view and its state changes. Whenever a user sends a request to the backend-server, the view should be reset, so that no mistakes are made on future requests. The memento pattern comes in place and sets the default state of the view and the view displays it accordingly. In the future it would be possible to safe inputs and load them as the view's new state.

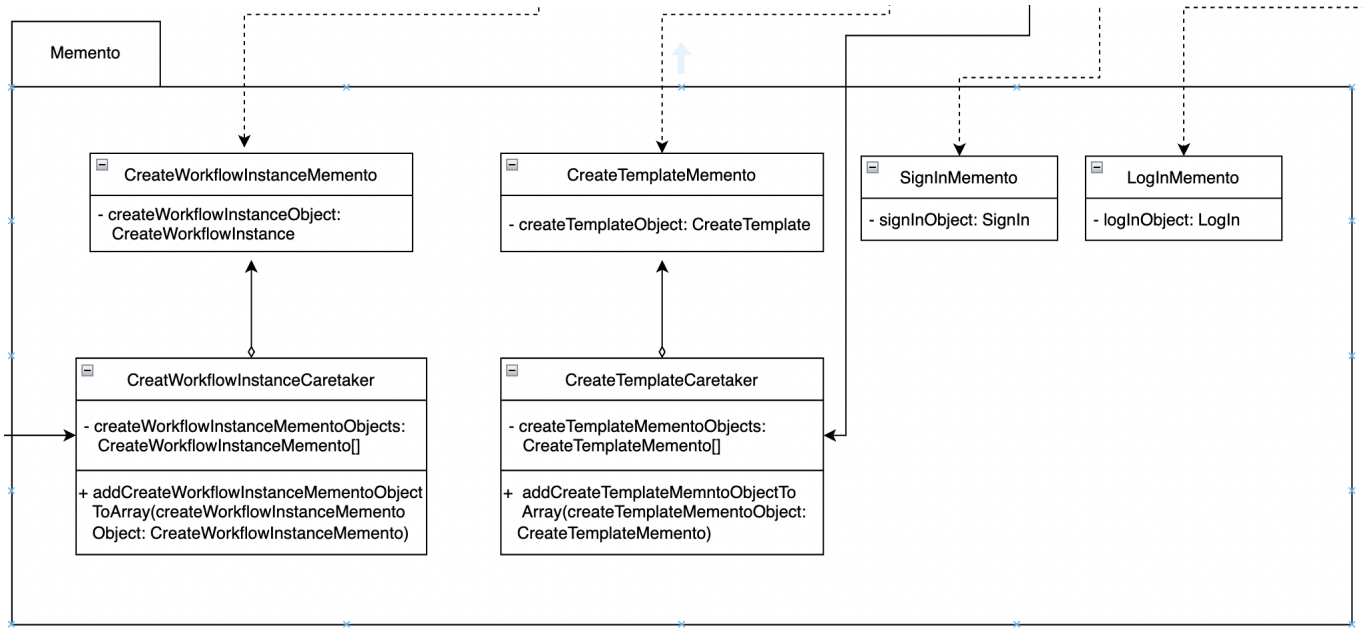

Abbildung 2: Memento package

3.2 JSON Converter

3.2.1 Frontend

A controler handels every request to convert a JSON-object into a class object. The controler takes in a string with the classname and the JSON-object, calls the method create...ObjectFromJSON in the corresponding class and returns the created object.

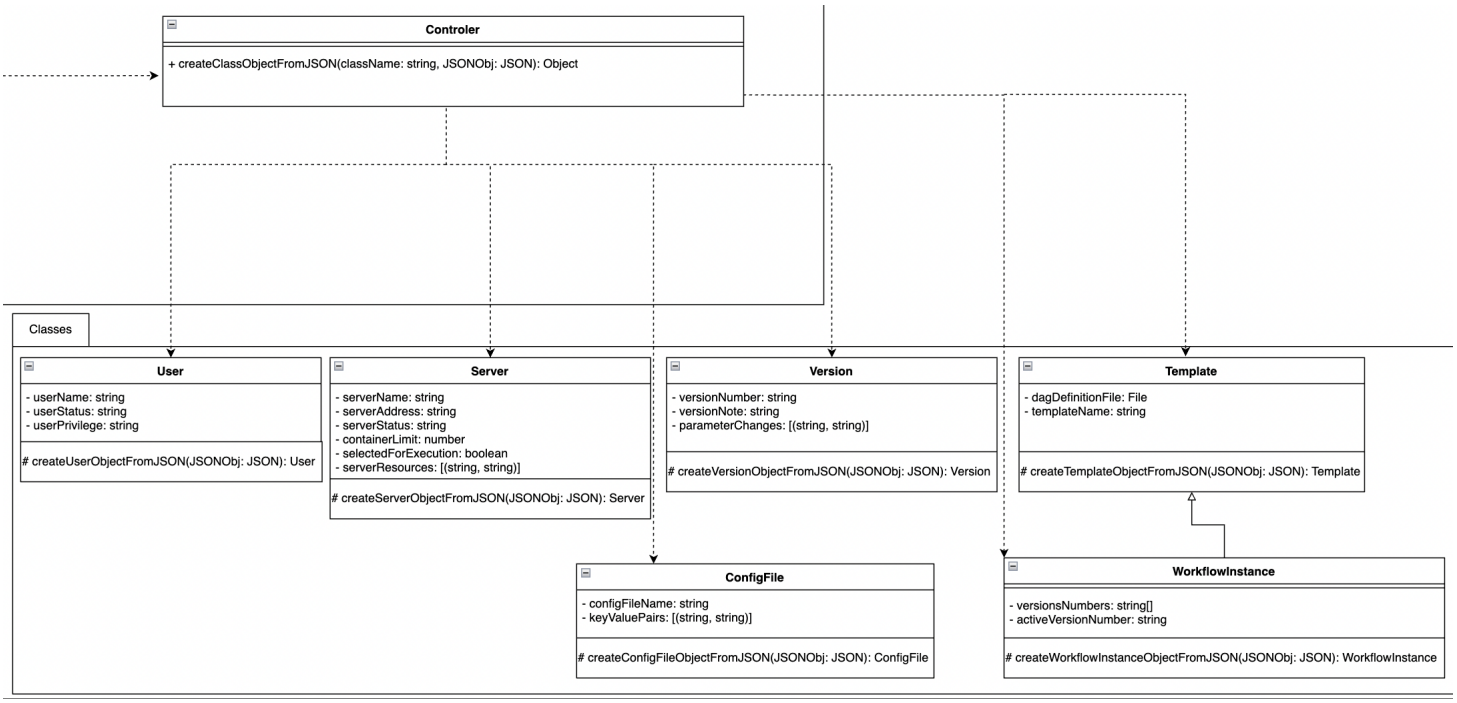

Abbildung 3: Controler package

3.2.2 API

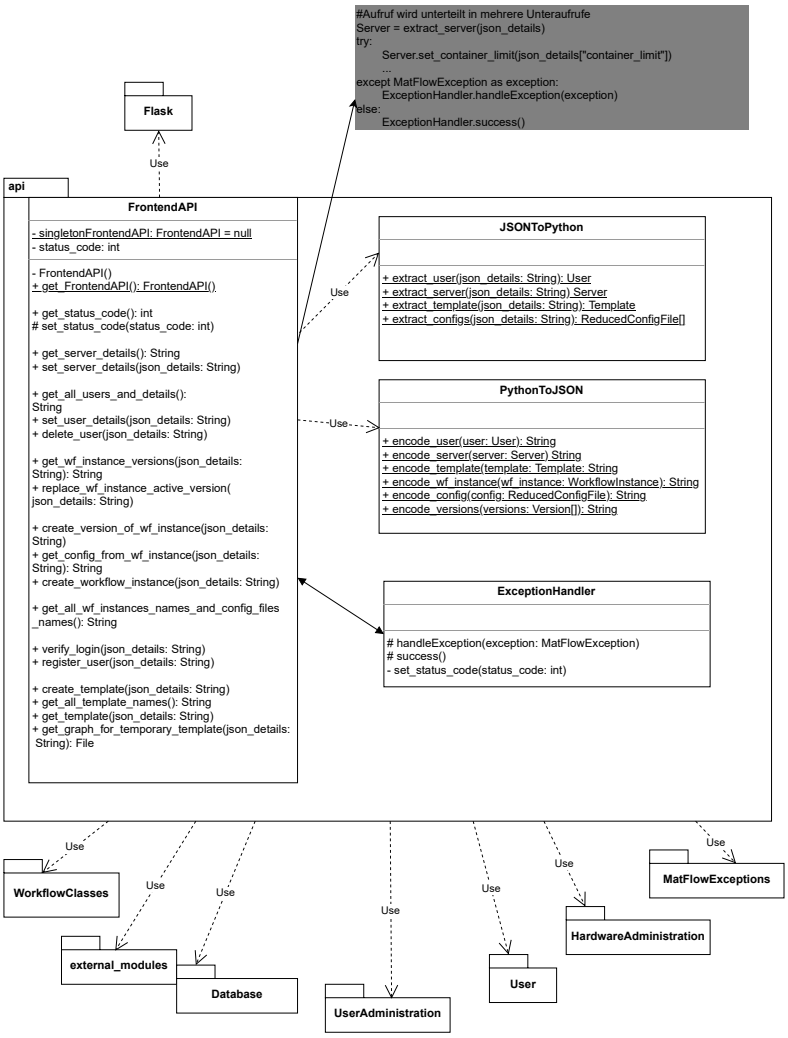

Abbildung 4: API package

Im Backend sind auch analog zum Frontend zwei JSON Converter(jewils von und zu) vorhanden. Diese Entwurfsentscheidung wurde aber verworfen, da sie das Kapselungsprinzip verletzt. Von nun an sind alle in den Converter vorhandenen Methoden direkt als Klassen- (von JSON) oder Objektmethoden (zu JSON) direkt in den jeweiligen Klassen implementiert. Als Beispiel: encode_template(template: Template) und extract_template(json_details: String): Template sind nun als encode_template(self): str und classmethod extract_template: Template in Template vorhanden.

3.3 JSON Status Code

Wie man auch in der API sieht, ist die Bereitstellung eines Status Codes bei einem request an die API nicht eindeutig dem Entwurf zu entnehmen, deswegen hier noch einmal eine saubere Erläuterung: Es wird bei einer Anfrage an die API diese Anfrage durchgeführt. Je nachdem ob sie nicht erfolgreich war (MatFlowException geworfen, spezieller Status Code vorhanden) oder doch erfolgreich war (success Status Code und eventuell Daten vorhanden) wird in der Klasse ExceptionHandler die Antwort in json gebaut und an den Client zurück geschickt. Der Client bekommt also **immer** eine Antwort mit einem json response body, indem sich der status code befindet. Es handelt sich hier also nur um eine Anlehnung an die bereits etablierten HTTP Status Codes.

3.4 Authentifizierung

Im Frontend kann man theoretisch unabhängig der User Privilegien mittels URL Manipulation auf alle Seiten zugreifen. Um dies zu verhindern, wird im Backernd ein einzigartiges Cookie erstellt, wenn sich der Benutzer einloggt. Bei jeder Anfrage aus dem Frontend wird dieser Cookie dann mitgeliefert und verhindert unerlaubten Zugriff.

3.5 keys

Zur Kapselung der JSON Keys wurde im FrontendAPI package eine eigenes Modul eingeführt.

3.6 Entfernen der Login und SignUp Klasse

Während der Implementierung der UserAdministration, ist schnell aufgefallen, dass sowohl Login- als auch SignUp keine benötigten Klassen sind. Stattdessen ist die Logik von Login und SignUp nun in der UserController Klasse beinhaltet.

In dem Diagramm wurden die 2 Redundanten Klassen entfernt und neue Methoden hinzugefügt. Des Weiteren ist die Schnittstelle zu Airflow nun genauer definiert worden.

3.7 hinzufügen der Schnittstelle zur Datenbank

Der Entwurf der HardwareAdministration konnte gut umgesetzt werden. Jedoch musste eine Schnittstelle zur Datenbank hinzugefügt werden, da der/die Server nicht über Airflow reguliert werden konnten.

In dem Diagramm wurden ein paar Methoden und die Schnittstelle zur Datenbank hinzugefügt.

3.8 Kommunikation von Dateien innerhalb der Serveranwendung

Alle Attribute, Parameter und Rückgaben, die im Entwurfsheft mit dem Typ „File“ gekennzeichnet waren, haben wir jetzt als Path implementiert. Wobei wir die Klasse Path aus dem Modul pathlib der Python-Standardbibliothek importieren.

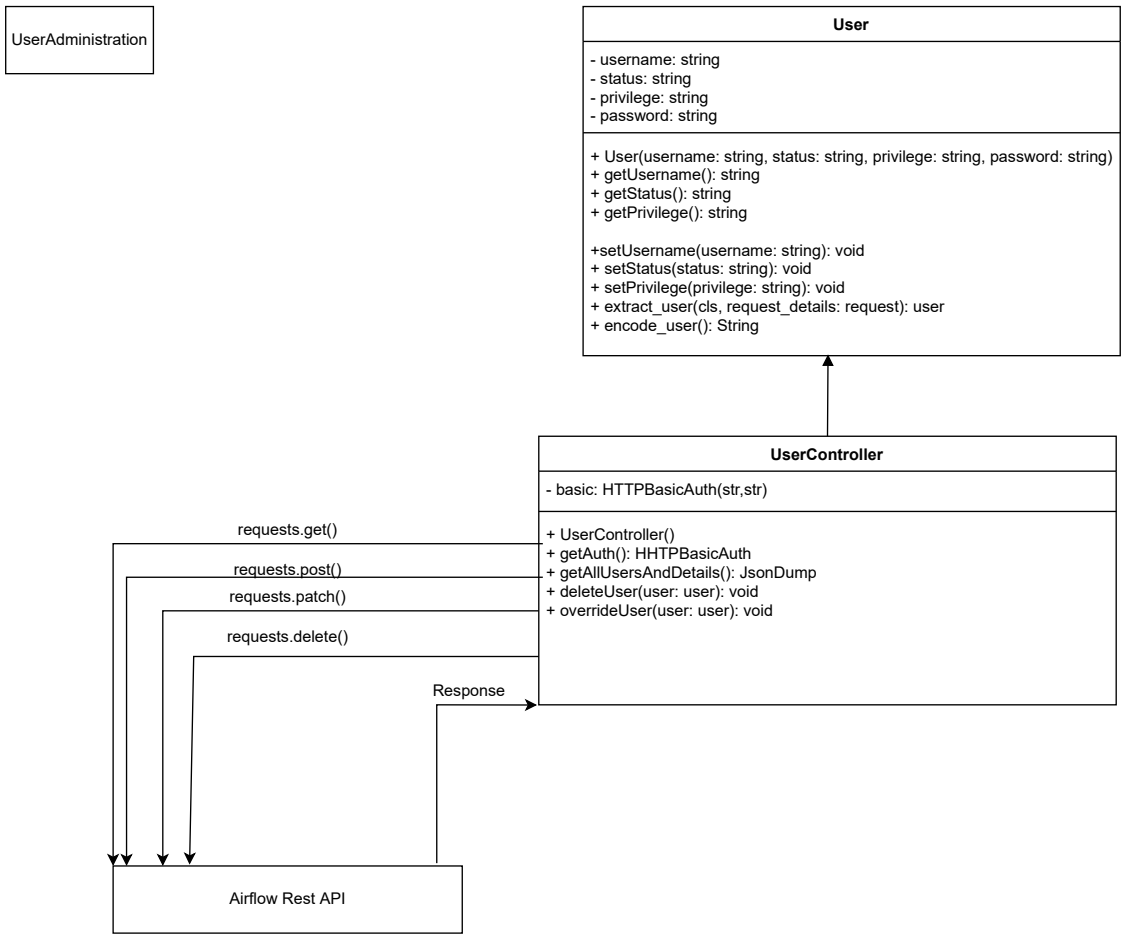

Abbildung 5: Die überarbeitete Version der UserAdministration

Im Folgenden soll die Handhabung der Dateien unter Zuhilfenahme von Pfaden erklärt werden. Dateien, die durch die API die oberste Schicht der Serverandwendung erreichen werden zunächst in einem temporären Ordner zwischengespeichert. Der entsprechende Pfad im temporären Ordner wird entweder durch Aufruf eines Konstruktors in einem Objekt abgelegt oder unmittelbar an die Schnittstelle des Workflow-Packages übergeben. Dort wird der Datei dann ein langfristiger Speicherort im Dateisystem zugewiesen. Die Löschung der Dateien im temporären Ordner obliegt dem Empfänger.

Muss umgekehrt eine Datei vom Workflow-Package an die Schnittstelle der Serveranwendung gealngen, so wird der Pfad des langfristigen Speicherorts verwendet. Dort kann die Datei ausgelesen werden, gelöscht wird sie anschließend natürlich nicht.

Im Entwurfsheft haben wir uns keine Gedanken über die Ordnerstruktur der auf dem Server gespeicherten Dateien gemacht, deshalb reichen wir nun einen entsprechenden Entwurf nach.

Zum einen gibt es den Ordner „templates“, der unmittelbar alle Dag-definition-files

der bisher erstellten Templates enthält. Der Name der Datei stimmt dabei stets mit dem Namen des Templates überein.

Daneben gibt es zudem noch den Ordner „workflow_instances“. Dieser enthält für jede erstellte Workflow-Instanz einen Unterordner, der den eindeutigen Namen der Instanz trägt.

In jedem dieser Unterordner befinden sich mindestens die Dag-Definition-File der Instanz und der Ordner „current_conf“, der in seiner Struktur stehts dem ursprünglich Konfigurationsordner enspricht. Insbesondere können auch Dateien mit einer Endung darin liegen, die nicht „.conf“ ist. Die „.conf“-Dateien sind dabei immer auf dem Stand der aktuellen Version der Workflowinstanz.

Ebenfalls fester Bestandteil ist der Ordner „1“, der die initiale Version der Instanz representiert. In ihm liegen alle „.conf“-Dateien in ihrer ursprünglichen Fassung. Zusätzlich gibt es noch für jede neu erstellte Version einen Ordner, in dem stets nur die in dieser Version geänderten „.conf “-Dateien liegen.

3.9 Änderungen im Workflow-Package

Ersetzung von Arrays durch Listen

An allen Stellen in denen im Entwurf ein Array vorgesehen war. Haben wir in der Implementierung Listen verwendet. Der wichtigste Grund dafür ist, dass Python die Verwendung von Listen schlicht besser unterstützt. Die Syntax die Python für den Umgang mit Listen anbietet erhöht dabei auch die Lesbarkeit des Programmcodes. Abstriche in der Performanz sind nicht zu erwarten, weil davon auszugehen ist, dass die Anzahl der Listenelemente sich in moderaten Bereichen bewegt.

Rückgabetyp von get_names_of_workflows_and_config_files in WorkflowManager

Das Informationsbedürfnis, das diese Methode abdecken soll sind für alle Workflow-Instanzen der Name und die Namen aller „.conf“-Dateien der Instanz. Das ursprüngliche Format war, für jede Instanz eine Liste von Strings, die zunächst den Namen der Instanz gefolgt von den Namen der Dateien enthalten sollte. Beim Implementieren kamen wir dann aber auf die wesentlich elegantere Lösung Dictionarys zu verwenden. Dabei sind die Namen der Instanzen Schlüssel und die Liste Namen der Dateien die zugehörigen Werte.

Umbenennung der Klasse Version in WorkflowVersion

Durch die Umbenennung ist klar, dass es die Workflow-Instanzen sind, die versioniert werden.

Verwefen der statischen Methode from_string in VersionNumber

Die Methode entpuppte sich als überflüssig, da sie sich zur stark mit dem Konstruktor überschnitt. Dieser hat als einzigen Eingabeparameter auch einen String. Die Prüfung ob der String dem Format einer Versionsnummer entspricht kann ebenfalls direkt im Konstruktor erfolgen.

En- und Decode-Methoden in den Datenklassen

Die Begründung und Details zu dieser Änderung finden sich in 3.2.2.

Neue Methode get_frontend_version in DatabaseVersion
In der neue Methode berechnen die Versionen mit ihren veränderten Dateien unter Hinzugabe der vorherigen Dateien, die Unterschiede zur Vorgänger-Version. Ursprünglich lag diese Funktionalität in der WorkflowManager-Klasse. Da das DatabaseVersion-Objekt aber selbst alle Informationen besitzt um die Berechnung durchzuführen, wäre es eine Missachtung des Kapselungsprinzips sie auf einer höheren Ebene anzusiedeln.

Verwefen der Attribute active_version und version_numbers in der Klasse WorkflowInstance
Die Klasse WorkflowInstance kommt nur bei der Erstellung von Instanzen zum Einsatz. Da zu diesem Zeitpunkt aber immer nur die Version „1“ existiert, kamen die oben genannten Attribute nie zum Einsatz.

Neue Methode activate_instance in WorkflowInstance
Sehr ähnlich wie bei der Methode get_frontend_version handelt es sich auch hier um Funktionalität die vom WorkflowManager in eine tiefere Ebene verschoben wurde. Die Methode ist dafür verantwortlich in der Dag-Definition-File den Namen der Instanz als dag_id einzutragen und die File in den „dags“-Ordner von Airflow zu kopieren. Die Methode ist bisher noch nicht implementiert.

3.10 Database Paket Implementierungsänderungen

TemplateData Die Klasse TemplateData des Database Pakets wurde entfernt da es keine Einträge in der Datenbank gibt die sie verwalten muss3.11.

DatabaseTable Unterscheidung zwischen einzelnen Werten und Mehreren für get(..) Funktion. Dadurch entstehen get_one(..) und get_multiple(..).

Neue Funktion **check_for(query: str): bool** um zu sehen ob ein Eintrag von der angegebenen query existiert.

Neue Funktion **setup_database()** die beim Aufrufen die Datenbank aufbaut

WorkflowData Hauptsächlich Typisierung und Namensgebung

3.11 MySQL - Datenbankaufbau

Auf 9 ist das Überarbeitete ER-Diagramm der Datenbank zu sehen.

Nach langen Überlegungen sind wir zu dem Schluss gekommen, dass es sinnvoller ist für die Dateispeicherung statt LONGBLOBS und weitere Parameter, nur Pfade in die Datenbank einzuspeichern, die auf die gewünschte Datei zeigen. Zum einen gibt es dadurch weniger rohe Daten die zwischen (manchen) Funktionen kommuniziert werden müssen, aber auch weniger Latenz beim schreiben oder lesen der Dateien aus der Datenbank.

Demzufolge ist es uns möglich Workflow Templates in einem einzigen Ordner zu speichern der alle DAG's beinhaltet die nach ihren Template Namen benannt sind. Dieser Ordner ist leicht zu erreichen und darüber zu iterieren. Die Tabelle Workflow_Template3.11 ist dementsprechend nicht mehr benötigt, da der Ordner als eigene Tabelle fungieren kann. Sie bleibt trotzdem bis auf weiteres in der Datenbank um bei eventuellen Komplikationen schneller wechseln zu können.

Weitere Änderungen beinhalten die Vereinfachung einer Workflow-Version in der Datenbank. Statt zwei Schlüssel 'wfName' und 'version' bekommt jede Version einen eindeutigen Integer Bezeichner. Dieser muss zwar für manche Zugriffe auf andere Tabellen zuerst von Version geholt werden, erhöht aber das Verständnis der Datenbankstruktur und bringt eine Erleichterung der MySQL-Befehle.

Die Verbindung von ResultFile wurde außerdem zu Version geändert. Es besteht keinen Grund das komplexe System der Speicherung von überlappenden .conf Dateien auch für die (höchstwahrscheinlich) immer unterschiedlichen Ergebnisse unterschiedlicher Versionen zu nutzen. Ein einfacher Fremdschlüssel der auf die entsprechende Workflow-Version zeigt reicht aus.

Dementsprechend sind Folgende Tabellen in der Datenbank vorhanden:

Server		
ip	varchar(50)	*key*
name	varchar(255)	*key*

WorkflowTemplate		
template_ID	int	*key*; *Auto_Increment*
name	varchar(255)	*notNull*
dag	varchar(255)	*notNull*

Workflow		
name	varchar(255)	*key*
dag	varchar(255)	*notNull*

FolderFile		
filesID	int	*key*; *Auto_Increment*
wfname	varchar(255)	*notNull*; name from Workflow
file	varchar(255)	*notNull*

Version		
ID	int	*key*; *Auto_Increment*
wfName	varchar(255)	name from Workflow
version	varchar(127)	*notNull*
note	varchar(1000)	

ActiveVersion		
wfName	varchar(255)	*key*; name from Workflow
version	varchar(127)	from Version

VersionFile		
versionID	int	*key*; ID from Version
filename	varchar(255)	*key*
confKey	int	*notNull*; from ConfFiles

ConfFile		
confKey	int	*key*; *Auto_Increment*
file	varchar(255)	*notNull*

ResultFile		
versionID	int	*key*; ID from Version
filesID	int	*key*; *Auto_Increment*
file	varchar(255)	*notNull*

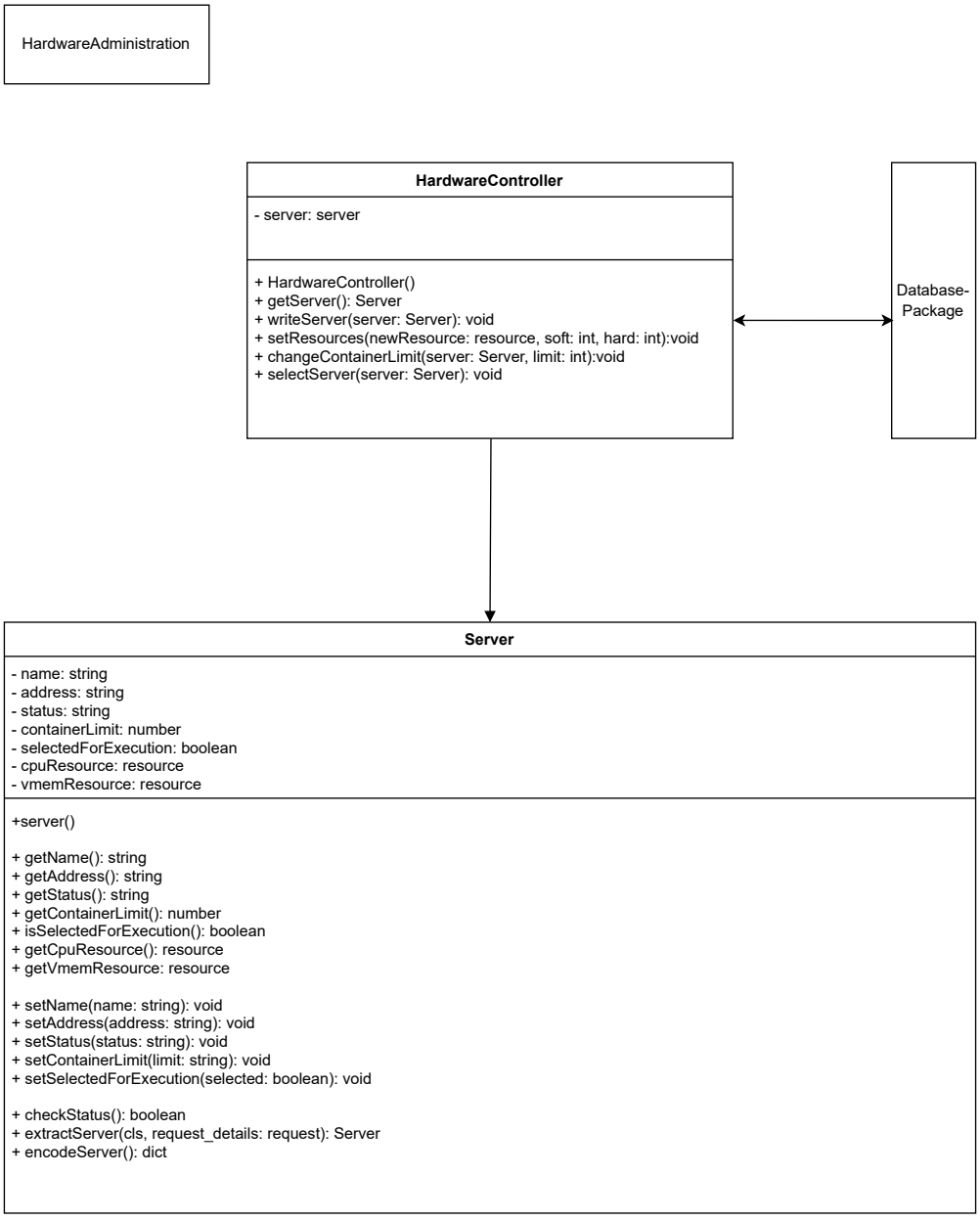

Abbildung 6: Die überarbeitete Version der HardwareAdministration

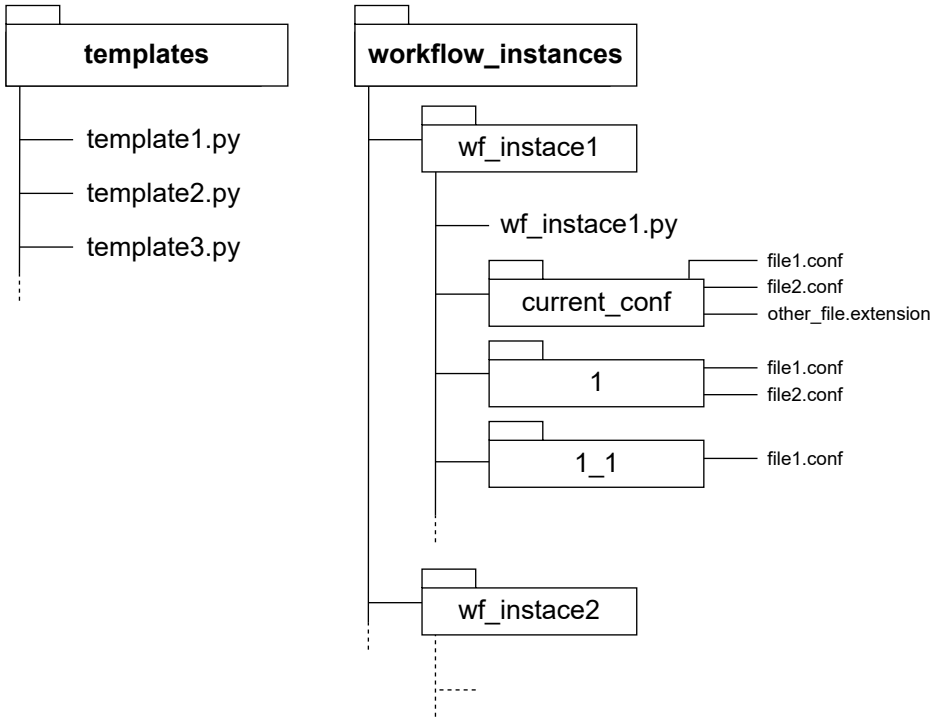

Abbildung 7: Eine graphische Veranschaulichung der Ordnerstruktur

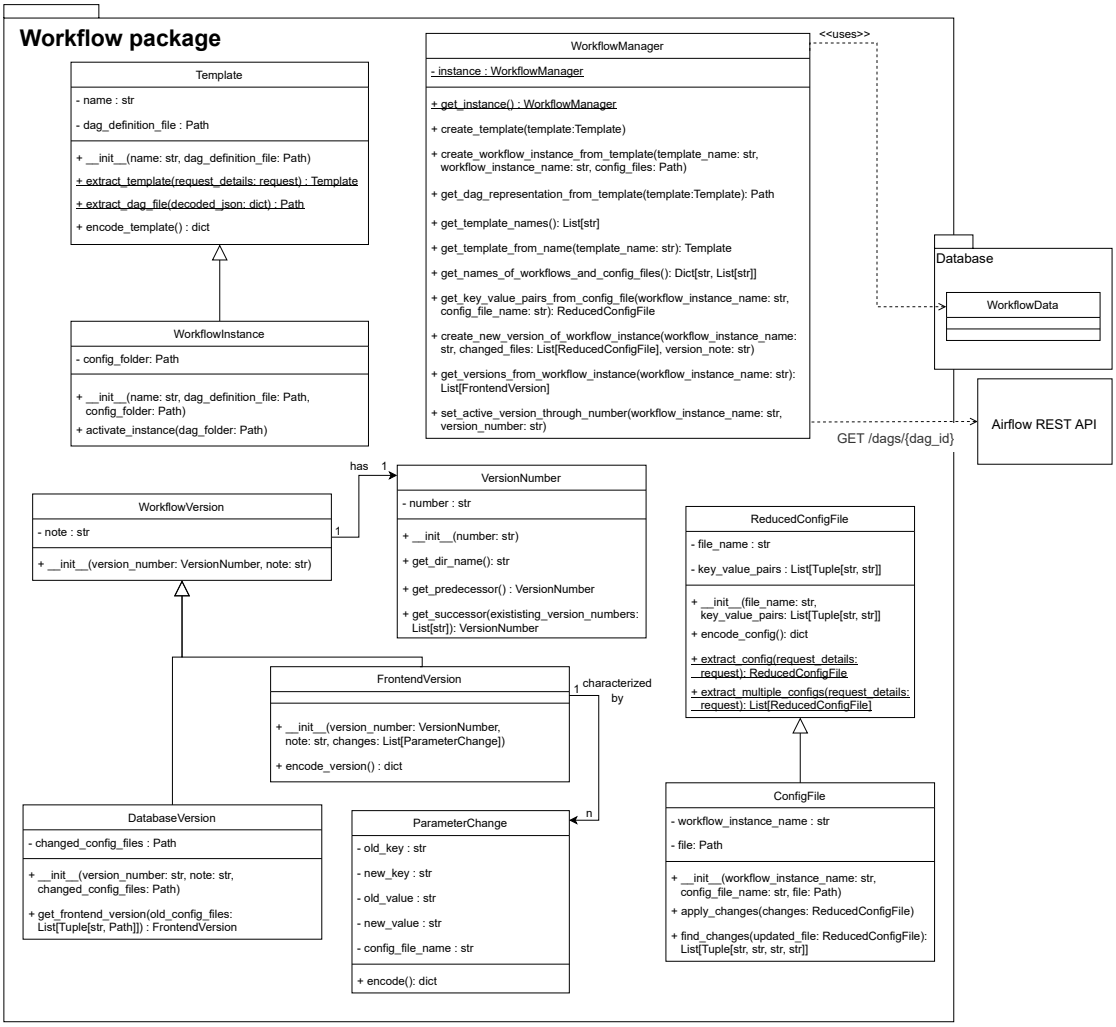

Abbildung 8: Das überarbeitete Klassendiagramm des Workflow Packages

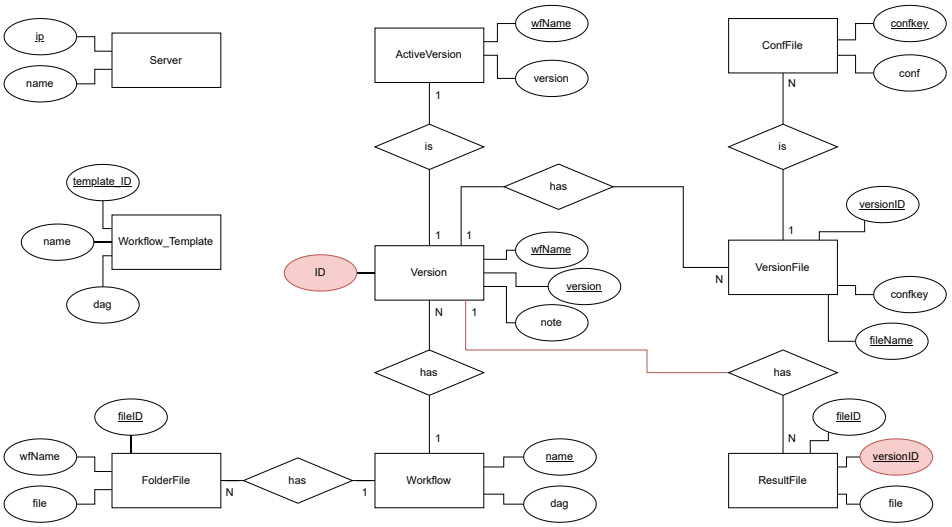

Abbildung 9: ER-Diagramm der Datenbank. Änderungen in rot

4 Anhang

4.1 Detailliertere Aufteilung

Man sieht im Folgenden die exportierten Issues aus Gitlab:

Issue ID	URL	Title	State	Description	Author	Author Usernam	Assignee	Assignee Userna	Confidential	Locked	Due Date	Created At (UTC	Updated At (UTC	Closed At (UTC)	Milestone	Weight	Labels	Time Estimate	Time Spent	Epic ID	Epic Title
1	https://git.scc.kit.	Exceptions schreiben	Closed	Alle MatFlowExceptions, die im System geworfen werden sollen	udhbh	udhbh	udhbh	udhbh	No	No	2022-01-20	2022-01-14 14:0	2022-01-20 09:0	2022-01-20 09:0				0	0		
2	https://git.scc.kit.	FrontendAPI	Closed	Jegliche Kommunikation zwischen Client Applikation und Server	udhbh	udhbh	udhbh	udhbh	No	No	2022-01-20	2022-01-14 14:3	2022-01-20 17:3	2022-01-20 17:3				0	0		
3	https://git.scc.kit.	Get FrontendAPI up and running	Open	API mit Flask auf Server laufen lassen	udhbh	udhbh	udhbh	udhbh	No	No	2022-01-25	2022-01-14 14:3	2022-01-20 17:3					0	0		
4	https://git.scc.kit.	MySQL auf Server laufen lassen	Open		udhbh	udhbh	udhbh, ulfth	udhbh, ulfth	No	No	2022-01-23	2022-01-14 14:4	2022-01-20 17:3					0	0		
5	https://git.scc.kit.	Backend App	Open	Backend Application verpacken (u.a. mit Docker, Nomad)	udhbh	udhbh	udhbh, ulfth	udhbh, ulfth	No	No	2022-02-01	2022-01-14 14:4	2022-01-20 17:3					0	0		
6	https://git.scc.kit.	/M60/: getVersionsFromWorkflowInstance()- and setActiveVersionThroughNumber()-method in WorkflowManager	Open	- in both cases forward the request to the database	uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-16 10:4	2022-01-20 17:3					0	0		
7	https://git.scc.kit.	backend functionality for creating templates /M10/	Closed	Methods - createTemplate() - getDagRepresentationFromTemplate() in class WorkflowManager - Implementation of the class Template	uczrb	uczrb	uczrb	uczrb	No	No	2022-01-20	2022-01-17 10:2	2022-01-20 17:3	2022-01-20 10:0				0	0		
8	https://git.scc.kit.	/M20/: createWorkflowInstanceFromTemplate()-method in WorkflowManager	Open	- arrange the given input inside a WorkflowInstance object - forward request to the database	uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-17 10:3	2022-01-20 17:3					0	0		
9	https://git.scc.kit.	/FA40/: operator for tasks alternating dependencies (used for tgds-models)	Open		uczrb	uczrb	uczrb	uczrb	No	No	2022-02-01	2022-01-17 10:4	2022-01-20 17:3					0	0		
10	https://git.scc.kit.	Install vujs and dependencies	Closed	Installing vuejs, vuetify, vuex	unjha	unjha	unjha, udhbh	unjha, udhbh	No	No	2022-01-12	2022-01-17 11:3	2022-01-20 17:3	2022-01-17 16:5				0	0		
11	https://git.scc.kit.	Connection between MySQL and python Code	Closed	maybe with mysql connector	upqzk	upqzk	upqzk	upqzk	No	No	2022-01-20	2022-01-17 15:3	2022-01-20 17:3	2022-01-20 12:1				0	0		
12	https://git.scc.kit.	Dummy-Data for frontend	Closed	In order to design the frontend features it's necessary to create dummy data that are placeholder for the backend data	unjha	unjha	unjha	unjha	No	No	2022-01-20	2022-01-17 16:5	2022-01-20 17:3	2022-01-17 18:2				0	0		
13	https://git.scc.kit.	Frontend: connect BackendServerCommunicator with dummy data	Closed	Connect the data from folder 'DummyData' with the corresponding functions in class 'BackendServerCommunicator'	unjha	unjha	unjha	unjha	No	No	2022-01-20	2022-01-17 18:2	2022-01-20 17:3	2022-01-17 18:5				0	0		
14	https://git.scc.kit.	Frontend: integrate vuex in the 'Model'-folder	Closed	Vuex has to be integrated in every Class of 'Model'-folder	unjha	unjha	unjha	unjha	No	No	2022-01-20	2022-01-17 18:5	2022-01-20 17:3	2022-01-18 10:3				0	0		
15	https://git.scc.kit.	frontend: Integrate Text editing	Open	The dag-definition file should be editable in the frontend-application	unjha	unjha	unjha	unjha	No	No	2022-02-01	2022-01-19 12:4	2022-01-20 17:3					0	0		
16	https://git.scc.kit.	frontend: File transfer and integration	Open	Eventhough most classes are implemented and dummy-data are created by now, there is no integration of files. The dag-definition file has to be integrated for template creation.	unjha	unjha	unjha, udhbh	unjha, udhbh	No	No	2022-02-01	2022-01-19 12:4	2022-01-20 17:3					0	0		
17	https://git.scc.kit.	frontend: Design frontend with vuetify	Open	The frontend application is designed with vuetify. The files containing vuetify are not implemented	unjha	unjha	unjha	unjha	No	No	2022-02-01	2022-01-19 12:5	2022-01-20 17:3					0	0		
18	https://git.scc.kit.	frontend: Messages	Open	For most requests done in the frontend-application an approval, error, connection loss message should appear	unjha	unjha	unjha	unjha	No	No	2022-02-01	2022-01-19 12:5	2022-01-20 17:2					0	0		
19	https://git.scc.kit.	frontend: Connect frontend to the frontend-api	Open	frontend-data is received from the frontend-api	unjha	unjha	unjha, udhbh	unjha, udhbh	No	No	2022-02-01	2022-01-19 12:5	2022-01-20 17:2					0	0		
20	https://git.scc.kit.	frontend: JSON-typescript and typescript-JSON converter implementation	Open	Implementing the JSON-typescript, typescript-JSON converter	unjha	unjha	unjha	unjha	No	No	2022-02-01	2022-01-19 12:5	2022-01-20 17:2					0	0		
21	https://git.scc.kit.	frontend: Make vue with typescript compatible	Closed	Ts is not supported in vue2 by default, but a requierement from the university is to use typescript. The files in folder classes and model are all made with typescript	unjha	unjha	unjha	unjha	No	No	2022-01-20	2022-01-19 15:5	2022-01-20 17:3	2022-01-19 17:3				0	0		
22	https://git.scc.kit.	frontend: Remove store component and disable eslintOnSave	Closed		unjha	unjha	unjha	unjha	No	No	2022-01-20	2022-01-19 15:5	2022-01-20 17:3	2022-01-19 16:0				0	0		
23	https://git.scc.kit.	frontend: fix issue with 'Material Icons'	Closed	vuetify can't use 'Material Icons' at the moment	unjha	unjha	unjha	unjha	No	No	2022-01-20	2022-01-19 17:4	2022-01-20 17:3	2022-01-19 18:3				0	0		
24	https://git.scc.kit.	FrontendAPI	Closed	Hardware calls	udhbh	udhbh	udhbh	udhbh	No	No	2022-01-20	2022-01-20 09:3	2022-01-20 17:3	2022-01-20 09:3				0	0		
25	https://git.scc.kit.	FrontendAPI	Closed	User calls	udhbh	udhbh	udhbh	udhbh	No	No	2022-01-20	2022-01-20 09:3	2022-01-20 17:3	2022-01-20 09:3				0	0		
26	https://git.scc.kit.	FrontendAPI	Closed	Workflow Calls	udhbh	udhbh			No	No	2022-01-20	2022-01-20 09:3	2022-01-20 17:3	2022-01-20 09:3				0	0		
27	https://git.scc.kit.	/M10/: Implementation of class Template	Open	- file management via paths - ensure that no templates are created from invalid dag-definition-files	uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-20 09:5	2022-01-20 17:3					0	0		
28	https://git.scc.kit.	/M10/: createTemplate()-method in WorkflowManager	Open		uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-20 09:5	2022-01-20 17:2					0	0		
29	https://git.scc.kit.	/M10/: getDagRepresentationFromTemplate()-method in WorkflowManager	Open	- forwards request to the Airflow system by creating a temporary dag - extract an return the given image	uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-20 09:5	2022-01-20 17:2					0	0		
30	https://git.scc.kit.	/M20/: getTemplateNames()- and getTemplateFromName()-methods in WorkflowManager	Open		uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-20 10:0	2022-01-20 17:2					0	0		
31	https://git.scc.kit.	/M20/: Implementation of the class WorkflowInstance	Open	- mainly the functionality inside the constructor of the class	uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-20 10:0	2022-01-20 17:2					0	0		
32	https://git.scc.kit.	/M60/: implementation of classes ReducedConfigFile and ConfigFile	Closed		uczrb	uczrb	uczrb	uczrb	No	No	2022-01-20	2022-01-20 10:1	2022-01-20 17:3	2022-01-20 10:3				0	0		
33	https://git.scc.kit.	/M60/: implementation of class Version and Subclasses	Open	subclasses are - FrontendVersion - DatabaseVersion	uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-20 10:1	2022-01-20 17:2					0	0		
34	https://git.scc.kit.	/M60/: getNamesOfWorkflowsAndConfigFiles()- and getKeyValuePairsFromConfigFile()-method in WorkflowManager	Open	- forward information request to database - bring results in the right format	uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-20 10:1	2022-01-20 17:2					0	0		
35	https://git.scc.kit.	/M60/: createNewVersionOfWorkflowInstance()-method in WorkflowManager	Open	- write the given changes into actual files - build version-object that is accepted by the database interface	uczrb	uczrb	uczrb	uczrb	No	No	2022-01-24	2022-01-20 10:2	2022-01-20 17:2					0	0		
36	https://git.scc.kit.	Set file saving method for database	Open		upqzk	upqzk	upqzk	upqzk	No	No	2022-01-25	2022-01-20 12:1	2022-01-20 17:2					0	0		
37	https://git.scc.kit.	Database package implementation	Open	Functions of DatabaseTable first, other classes later	upqzk	upqzk	upqzk	upqzk	No	No	2022-01-20	2022-01-20 12:1	2022-01-20 17:2					0	0		
38	https://git.scc.kit.	MySQL database table setups	Open	mysql queries for one-time setup uppon starting without the established tables	upqzk	upqzk	upqzk	upqzk	No	No	2022-01-25	2022-01-20 12:1	2022-01-20 17:2					0	0		
39	https://git.scc.kit.	Database package tests	Open	Unclear if TDD is best approach	upqzk	upqzk	upqzk	upqzk	No	No	2022-02-01	2022-01-20 12:2	2022-01-20 17:2					0	0		
40	https://git.scc.kit.	creating Hardware- und User-Administration package	Open		ulfth	ulfth	ulfth	ulfth	No	No	2022-01-20	2022-01-20 13:2	2022-01-20 17:2					0	0		
41	https://git.scc.kit.	implementing User and UserController class	Open		ulfth	ulfth	ulfth	ulfth	No	No	2022-02-01	2022-01-20 13:2	2022-01-20 17:2					0	0		
42	https://git.scc.kit.	User-Administration airflow rest api calls implemented	Open		ulfth	ulfth	ulfth	ulfth	No	No	2022-02-01	2022-01-20 13:2	2022-01-20 17:1					0	0		
43	https://git.scc.kit.	implementing User-Administration Unittests	Open		ulfth	ulfth	ulfth	ulfth	No	No	2022-02-01	2022-01-20 13:2	2022-01-20 17:1					0	0		
44	https://git.scc.kit.	Server and ServerController class implementation	Open		ulfth	ulfth	ulfth	ulfth	No	No	2022-01-23	2022-01-20 13:2	2022-01-20 17:1					0	0		
45	https://git.scc.kit.	implementing Hardware-Administration Unittests	Open		ulfth	ulfth	ulfth	ulfth	No	No	2022-02-01	2022-01-20 13:2	2022-01-20 17:1					0	0		

Abbildung 10: Aufteilung