

# **Entwurf MatFlow Workflowanwendung für Machine Learning Experimente**

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

Dezember 2021

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
<b>2</b>	<b>Systemüberblick</b>	<b>4</b>
<b>3</b>	<b>Kommunikation</b>	<b>5</b>
<b>4</b>	<b>Komponentenbeschreibung</b>	<b>6</b>
4.1	Client . . . . .	6
4.2	Server . . . . .	7
<b>5</b>	<b>Paketbeschreibung</b>	<b>8</b>
5.1	Client Application . . . . .	8
5.2	Server Application . . . . .	8
5.3	Database . . . . .	14
<b>6</b>	<b>Abläufe</b>	<b>16</b>
<b>7</b>	<b>Datenbank</b>	<b>17</b>
7.1	Datenbankaufbau . . . . .	17

## 1 Einleitung

## 2 Systemüberblick

### 3 Kommunikation

## **4 Komponentenbeschreibung**

### **4.1 Client**

## 4.2 Server

## 5 Paketbeschreibung

### 5.1 Client Application

### 5.2 Server Application

#### 5.2.1 Workflow Package

**Class: `WorkflowManager`** This class provides a singleton object that receives all requests addressed to this package. This class is also communicating with the API of Airflow as well as with the Database package.

#### Methods

**`getInstance():WorkflowManager`** Returns the singleton object if already existing otherwise it calls the private constructor

**`createTemplate(template:Template):void`** Causes the creation of a new template entry in the database. The new template can be used for creating workflow instances afterwards.

#### Parameters

- `template`: A `Template` object that provides all the necessary components for creating a new workflow template

#### Exceptions

- `DoubleTemplateNameException` Gets thrown if the user tries to use an already existing template name
- `InvalidDagFileException` Gets thrown if the user hands over an invalid dag-definition-file

**`createWorkflowInstanceFromTemplate(templateName:String, workflowInstanceName:String, configFiles:File[]):void`** Causes the instantiation of a workflow instance under the use of a workflow template.

#### Parameters

- `templateName`: The identifier of the template that is used for instantiation
- `workflowInstanceName`: The name of the new workflow instance
- `configFiles`: Contains all the files needed for the execution of the workflow



## Exceptions

- `DoubleWorkflowInstanceNameException` Gets thrown if the user tries to use an already existing workflow instance name
- `EmptyConfigFolderException` Gets thrown if the user hands over an empty config-folder

**`getTemplateNames():String[]`** Makes database request and returns the names of all templates in the system

**`getTemplateFromName(templateName:String):Template`** Forwards the request for the named template to the database and returns result

## Parameters

- `templateName`: The identifier of the desired template

**`getNamesOfWorkflowsAndConfigFiles():String[][]`** Forwards the request to the database and returns the names of all workflow instances in combination with the associated config-files

**`getKeyValuePairsFromConfigFile(workflowInstanceName:String,configFileName:String):String[][]`**  
Requests the desired config-file from the current version of the named workflow instance from the database. Afterwards changes the format from a file into an array of key-value-pairs and returns the result

## Parameters

- `workflowInstanceName`: The identifier of the workflow instance
- `configFileName`: The name of the desired config file

**`createNewVersionOfWorkflowInstance(workflowInstanceName:String,changedFiles:ReducedConfigFile,versionNote:String):void`** Causes the creation of a new version of the workflow instance in the database. The predecessor of the new version is the active version of the instance.

## Parameters

- `workflowInstanceName`: The identifier of the workflow instance of which a new version is created
- `changedFiles`: The config-files that were changed in comparison to the predecessor version in key-value-pair representation
- `versionNote` Note about the new version given by the user

### **getVersionsFromWorkflowInstance(workflowInstanceName:string):FrontendVersion[]**

Requests information about all the versions of the given workflow instance from the database. Afterwards calculates the difference to the predecessor for every version and returns that information in combination with the version numbers and version notes

#### **Parameters**

- workflowInstanceName: The identifier of the given workflow instance

### **setActiveVersionThroughNumber(workflowInstanceName:String, versionNumber:String):void**

Changes the active version of a workflow instance in the database

#### **Parameters**

- workflowInstanceName: The name of the workflow instance of which the active version shall be changed
- versionNumber: The number of the new active version

#### **Exceptions**

- WorkflowRunningException Gets thrown if the workflow instance is running. In that case the active version can not be changed

**Class: Template** This class represents a workflow template. It contains the identifying name of the template as well as a dag-definition-file.

#### **Constructor**

### **Template(name:String, dagDefinitionFile:File):Template**

#### **Parameters**

- name: The name of the new template
- dagDefinitionFile: The file which defines the behavior of workflows instantiated of this template

#### **Exceptions**

- DoubleTemplateNameException Gets thrown if the database already contains a template with the given name
- InvalidDagFileException Gets thrown if the handed dag-definition-file is invalid

**Class: WorkflowInstance** This class represents a workflow template. It contains the identifying name of the template as well as a dag-definition-file.

#### Constructor

**WorkflowInstance(name:String, dagDefinitionFile:File, configFolder:File[]):WorkflowInstance**

#### Parameters

- name: The identifying name of the workflow instance
- dagDefinitionFile: The file which defines the behavior of the workflow instance
- configFolder: An array of files that is needed for executing the workflow

#### Exceptions

- DoubleWorkflowInstanceNameException Gets thrown if the workflow instance name is already given away to another workflow instance
- EmptyConfigFolderException Gets thrown if the handed configFolder-array is empty

**Class: ParameterChange** This class represents the change of a key-value pair. It contains the old aswell as the new version of both the key aswell as the value. Furthermore it holds the name of the file the change was made in.

#### Constructor

**ParameterChange(oldKey:String, newKey:String, oldValue:String, newValue:String, configFile:String):ParameterChange**

**Class: ReducedConfigFile** This class holds all information to represent a config-file in the frontend.

#### Constructor

**ReducedConfigFile(fileName:String, keyValuePairs:(String,String)[[]])**

#### Parameters

- fileName: The name of the config-file through which the config-file can be identified among other config files of the workflow instance
- keyValuePairs: The contents of the file in key-value-pair representation

**Class: Version** This class represents a version of a workflow instance. It's main task is to switch between the two different representations of versions one in the frontend, one in the database.

#### **Constructor**

**Version(versionNumber:VersionNumber, note:String):Version**

#### **Parameters**

- versionNumber: Number that identifies the new Version
- note: Note from user that can be used for documenting the version

**Class: FrontendVersion** This class inherits from Version and is specialized to satisfy the need for information of the client application

#### **Constructor**

**FrontendVersion(versionNumber:VersionNumber, note:String, changes:ParameterChange[]):FrontendVersion**

#### **Parameters**

- versionNumber: Number that identifies the new Version
- note: Note from user that can be used for documenting the version
- changes: Contains the difference to the predecessor version in form of the changed key-value-pairs

**Class: DatabaseVersion** This class inherits from Version and is specialized to fit the way versions are managed in the database

#### **Constructor**

**DatabaseVersion(versionNumber:VersionNumber, note:String, changedConfigFiles:File[]):DatabaseVersion**

#### **Parameters**

- versionNumber: Number that identifies the new Version
- note: Note from user that can be used for documenting the version
- changedConfigFiles: Contains all the config-files that were changed in this version

**Class: VersionNumber** This class represents string expressions that are valid version numbers

### Methods

**static fromString(number:String):VersionNumber** This static method converts a string of correct syntax into a VersionNumber by calling the private constructor.

### Parameters

- number: String that has to fit the regular expression of a version number

**getSuccessor(workflowInstance:String):VersionNumber** This method calculates the smallest free successor of the VersionNumber for the given workflow instance

### Parameters

- workflowInstance: Identifier of the workflow instance to which the new number corresponds to

## 5.2.2 API Schnittstelle

**Class: FrontendAPI** Diese Klasse ist der Dreh- und Angelpunkt der ganzen Anwendung. Alle Anfragen von der Client Applikation werden an diese API, entworfen im Bequemlichkeitsmuster der **Fassade**, (im JSON Format) geschickt. Die Entscheidung FrontendAPI als **Singleton** zu entwerfen besteht darin, dass die API eine global ansprechbare Instanz sein soll, die auf dem Server an einem bestimmten Port laufen soll. Wie dies geschieht sind Implementierungsdetails, bei denen das eingebundene Framework **Flask** hilft, muss aber in die Entwurfsentscheidung einbezogen werden.

### Constructor

**Version(versionNumber:VersionNumber, note:String):Version**

### Parameters

- versionNumber: Number that identifies the new Version
- note: TODO

## 5.3 Database

### 5.3.1 WorkflowData

Diese Klasse ist die Schnittstelle für alle Klassen die auf den Datenbankeinträgen für Workflows arbeiten wollen. Sie erstellt für jeden Zugriff den passenden Datenbankbefehl und konvertiert die Ergebnisse in das gewünschte Format des Aufrufers. Sie wirft bei fehlerhaften Werten eine **MatFlowException**.

**createWorkflowInstanceFromTemplate(tName: String, wfName: String, confFold: Folder):void** Create a new instance of a workflow by using the dag-file of a Template, with the Version set to 1. Search confDirect for every .conf-file and add them into the Database. Set this Version as active Version.

- **tName** name of Template of which the dag should be taken from
- **wfName** name of the new Workflow
- **confFold** the folder where all data of the Workflow is saved into

**getNamesOfWorkflowsAndConfigFiles():String[][]** Return all Workflow names and the names of their corresponding config files. The returning value is a list of lists of Strings where an inner list has the form [<Workflow Name>, <config File1 Name>, <config File2 Name>, ...]

**createNewVersionOfWorkflowInstance(wfName:String, newVersion:DatabaseVersion, oldVersionNr:String):void** Create a new Version of an existing Workflow with changed config Files. Search which config Files changed and set those in the new Version.

- **wfName** name of the Workflow
- **newVersion** new Version identifier
- **oldVersionNr** identifier of Version the new one is based on

**getDatabaseVersionsOfWorkflowInstance(wfName:String):DatabaseVersion[]** Return all Versions of a Workflow Instance as DatabaseVersion Objects in a list. Asks the Database for the data and constructs every Object.

- **wfName** name of the Workflow

**setActiveVersionThroughNumber(wfName:String, version:String):void** Set the previous Version as inactive and a new Version as active

- **wfName** name of the Workflow
- **version** version to be set active

**getVersionNumbersOfWorkflowInstance(wfName:String):String[]** Return a sorted list of Strings of all Versions of a Workflow.

- **wfName** name of Workflow which versions should be listed

### 5.3.2 TemplateData

Diese Klasse ist die Schnittstelle für alle Klassen die auf den Datenbankeinträgen für Workflow\_Template arbeiten wollen. Sie erstellt für jeden Zugriff den passenden Datenbankbefehl und konvertiert die Ergebnisse in das gewünschte Format des Aufrufers. Sie wirft bei fehlerhaften Werten eine **MatFlowException**.

**createTemplate(template:Template):void** Read Values from template and convert them into a sql query. Throw error if name already exists.

- **template** Template object to convert

**getTemplateNames():String[]** Return a list of all existing Template names. List is empty when no Template exists.

**getTemplateByNames(name:String):Template** Return Template that is asked for. Throw error if Template doesn't exist.

- **name** name/identifier of the wanted Template

### 5.3.3 DatabaseTable

DatabaseTable ist die einzige Klasse die direkt mit der Datenbank kommuniziert. Ihre Funktion ist hauptsächlich MySQL Befehle entgegennimmt, diese an die Datenbank weiterzugeben und die dadurch erhaltenen Antworten zurückgibt. Im Fall eines Fehlers, wie einem leeren Ergebnis oder eines ungültigen Befehls wird eine **MatFlowException** geworfen.

**set(create: String):String** Create new entry on Database. Try to execute create on Database and return value.

- **create** a sql query to create an entry

**delete(del: String):String** Delete an existing entry. Try to execute del on Database and return value.

- **del** a sql query to delete an entry

**modify(change: String):String** Modify an existing entry. Try to execute change on Database and return value.

- **change** a sql query to change values

## 6 Abläufe



## 7 Datenbank

### 7.1 Datenbankaufbau

Die Datenbank beinhaltet folgende Tabellen:

<b>Workflow_Template</b>		
name	String	<i>key; notNull</i>
dag	.py -File	<i>notNull</i>

<b>Workflow</b>		
name	String	<i>key; notNull</i>
files	String	<i>notNull</i>
dag	.py -File	<i>notNull</i>

<b>Version</b>		
wfName	String	<i>key; notNull; name from Workflow</i>
version	String	<i>key; notNull</i>
note	String	

<b>ActiveVersion</b>		
wfName	String	<i>key; notNull; name from Workflow</i>
version	String	<i>notNull; from Version</i>

<b>VersionFile</b>		
wfName	String	<i>key; notNull; name from Workflow</i>
version	String	<i>key; notNull; from Version</i>
confKey	String	<i>key; notNull; from ConfFiles</i>

<b>ConfFile</b>		
confKey	String	<i>key; notNull</i>
file	.conf -File	<i>notNull</i>

**Notiz** Die dag Datei hätte als Schlüssel zwischen Workflow\_Template und Workflow fungieren können, ist aber redundant für alle Operationen auf einem Workflow. Deshalb wurde entschieden die Datei bei einer Workflowerstellung direkt zu kopieren.