Project Management SaaS (PMSaaS)

Documentation

1 INDEX

2 START OF THE PROJECT

Project Management SaaS, short PMSaaS, is an easy-to-use tool to keep track of your projects. The requirements where discussed with the project leaders of the Service Engineering team in a workshop at the start of the project. The main project was planned from June to October 2016. The presentation at the end of October gave new inputs and ideas for optimizations. Between then and mid of April, these optimizations were implemented. During the IPA (Individuelle Praktische Arbeit) from 19.04.2017 – 03.05.2017 another new function will be implemented.

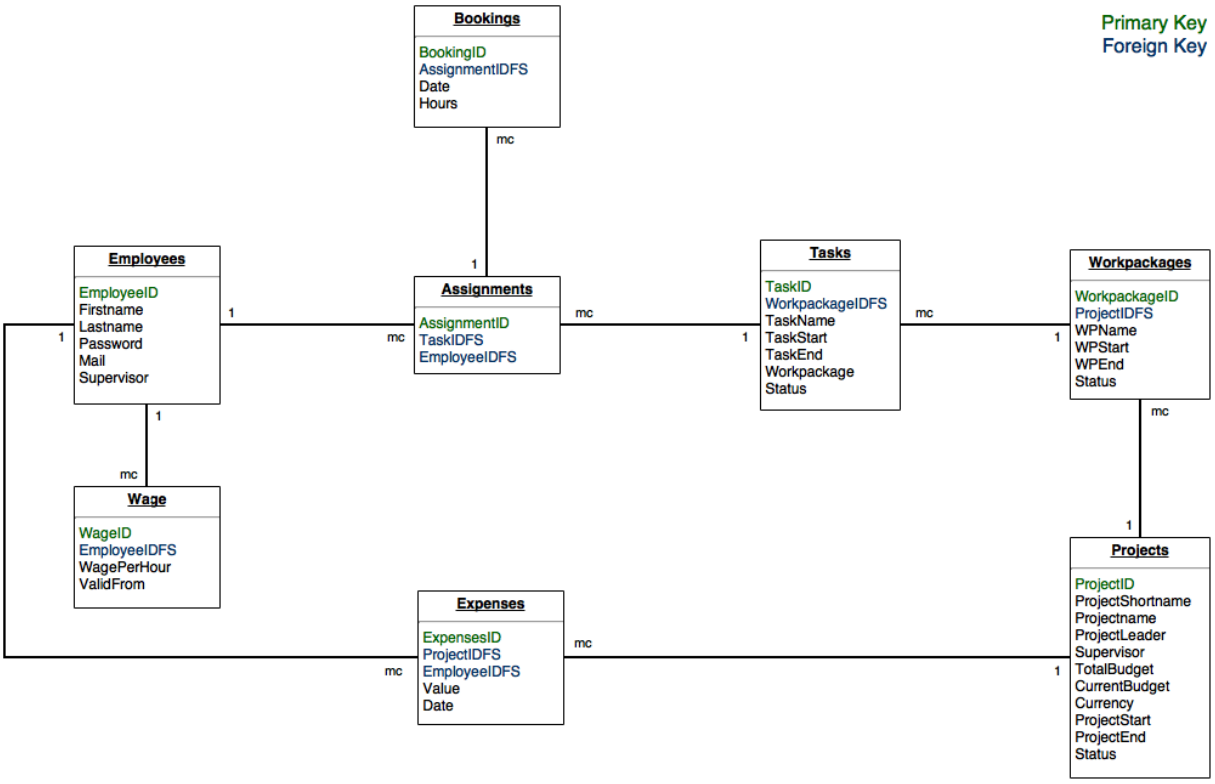
3 REQUIREMENTS

All requirements that resulted from the workshop with the project leaders are here listed:

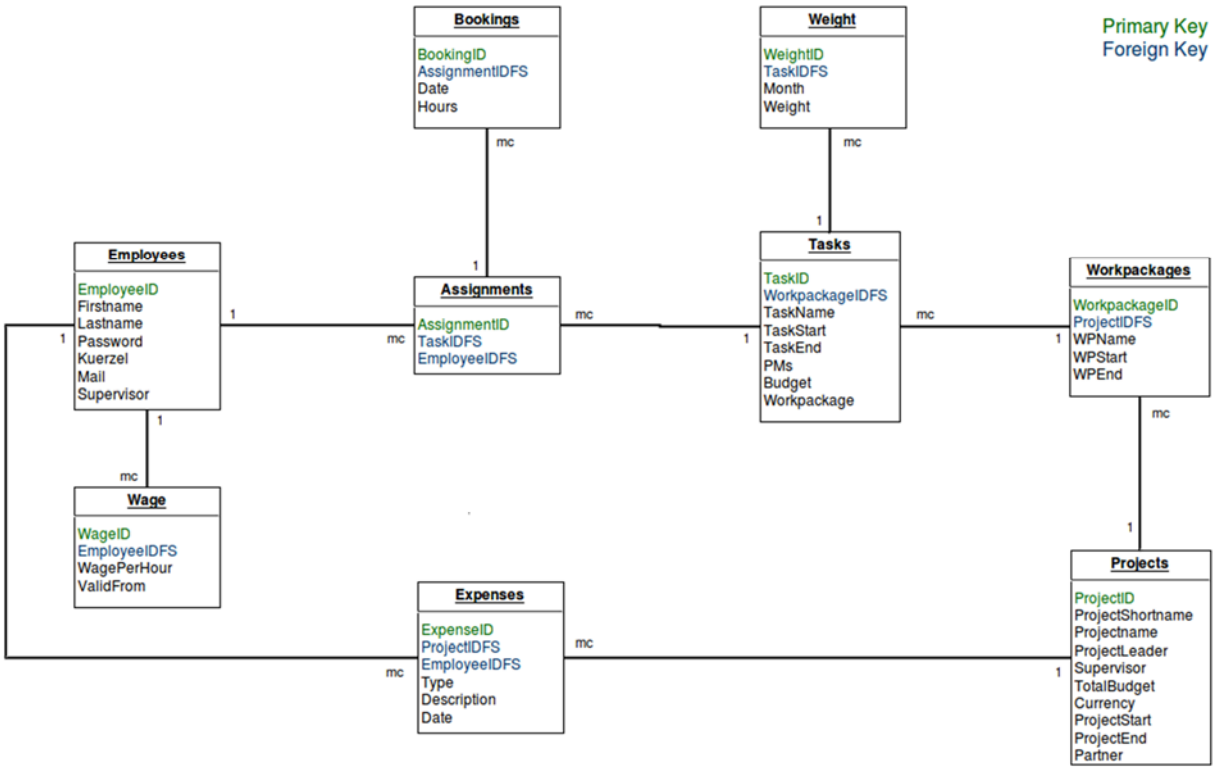
1. Project setup
    a. Enter tasks, start, end, PMs
    b. Map task to work package / task account in ZEUS
    c. Add people to the project
2. Monthly effort tracking
    a. Enter who has billed how much on each task
    b. (optional) import hours from CSV → apply consumed effort to task
3. Planning
    a. Assign people to task (hours)
4. Reporting
    a. Periodic → effort spent from date to date broken down per month

4 DATABASE

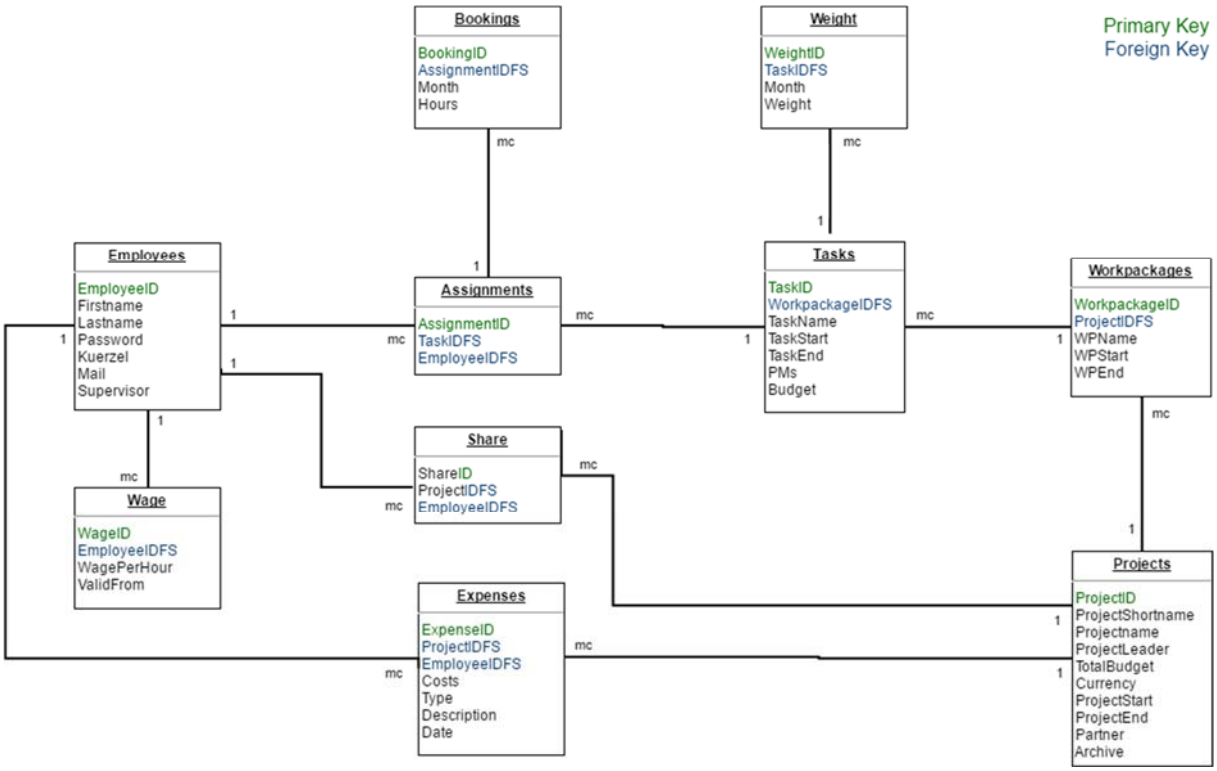
The database was designed at the start of the project but a few changes were made during the implementation of PMSaaS. The most significant changes were adding the two tables Weight and Share.



Entity Relationship Model 1, 13.06.2016



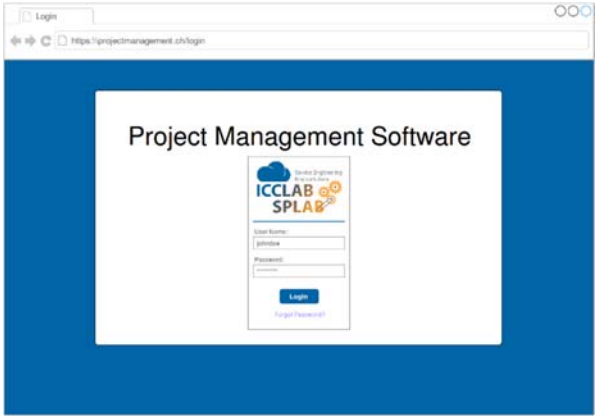
Entity Relationship Model 2, 31.10.2016



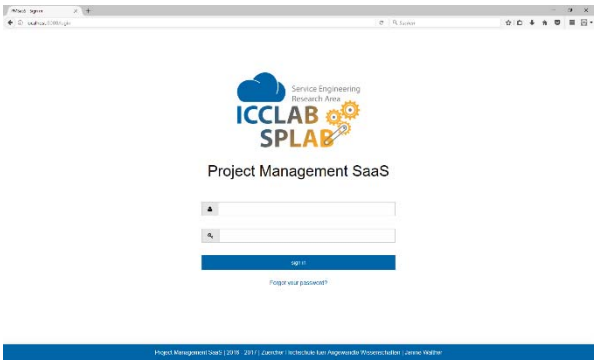
Entity Relationship Model 3, 07.11.2016

5 GUI MOCK-UPS & CURRENT DESIGN

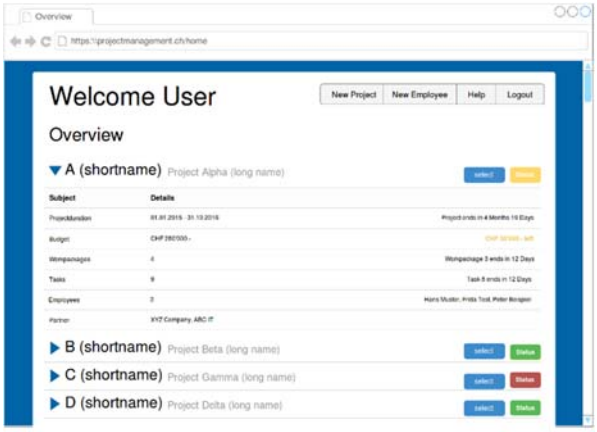
In the initial phase of the project, different mock-ups were made. However, the design of PMSaaS changed during the development. Here is a comparison of the mock-ups and the current design.



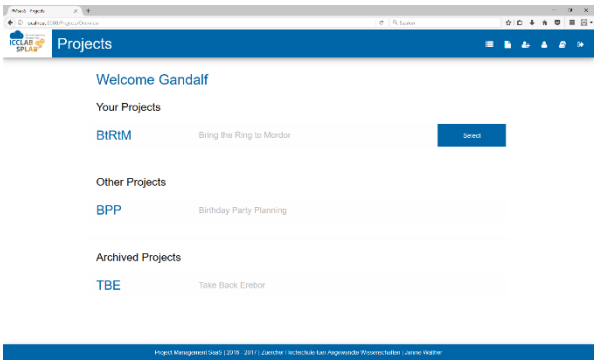
Mock-Up 1 Login



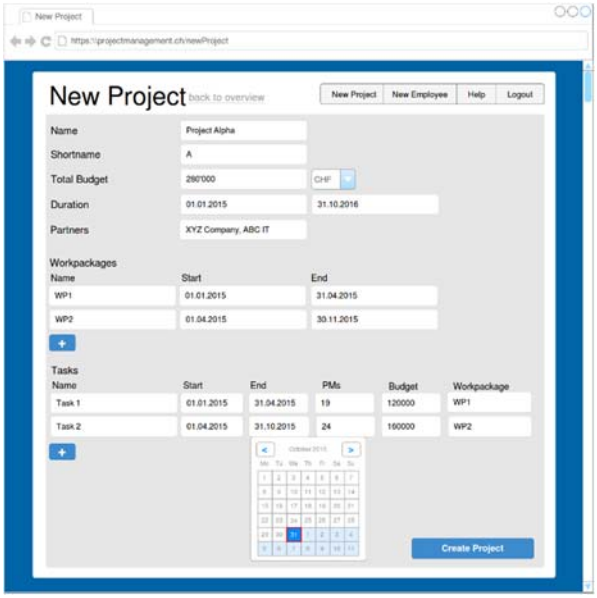
Screenshot 1 Login



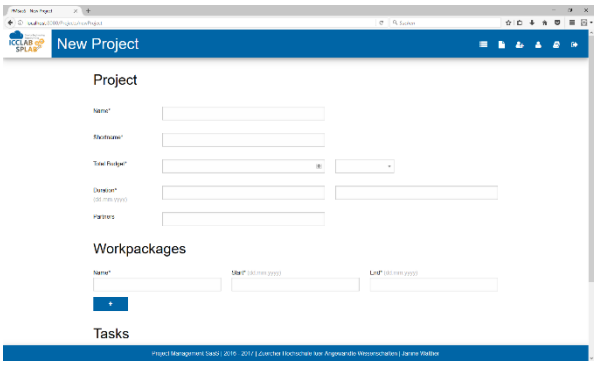
Mock-Up 2 Overview (home screen)



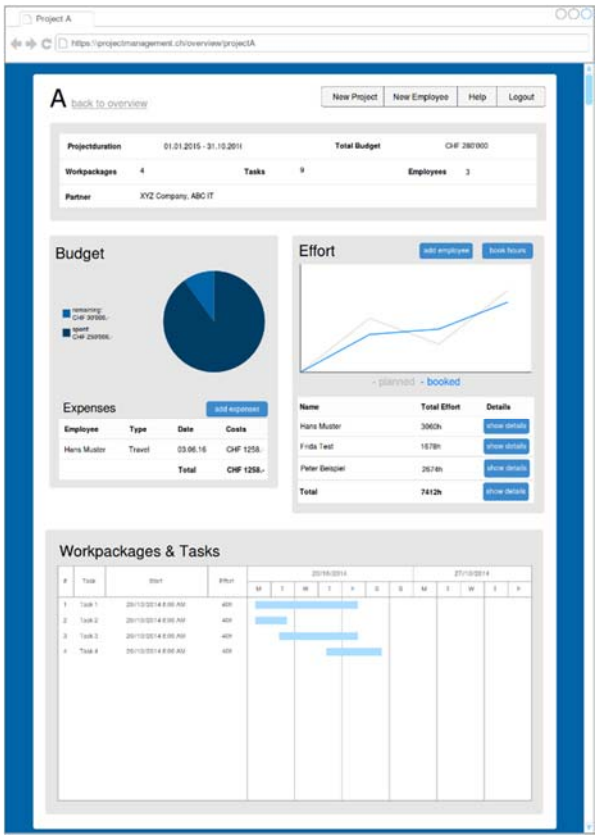
Screenshot 2 Overview (home screen)



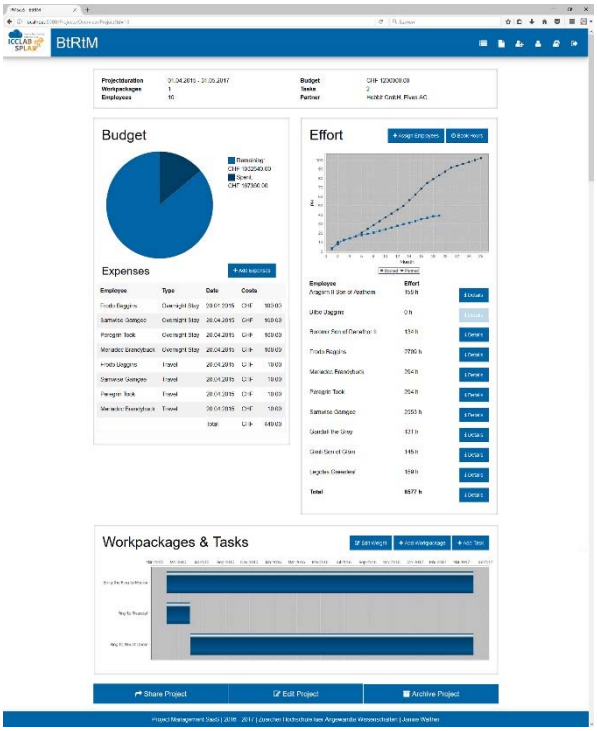
Mock-Up 3 New Project



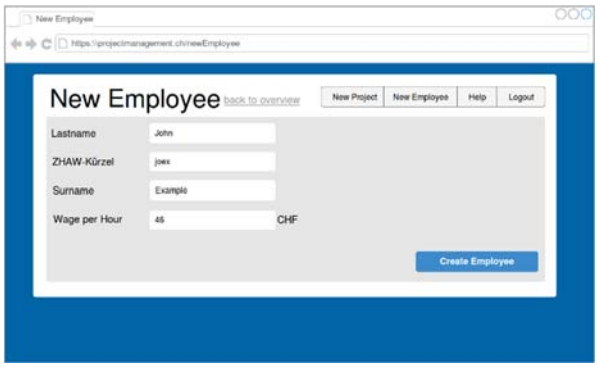
Screenshot 3 New Project



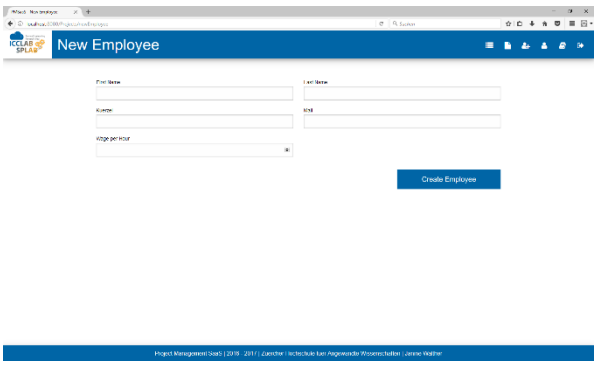
Mock-Up 4 Project Overview



Screenshot 4 Project Overview



Mock-Up 5 New Employee



Screenshot 5 New Employee

As you can see, there are a few differences between the mock-ups and the current design. On the left top, you can see a logo which can be changed by the admin. The design in the mock-ups was a bit confusing, the new one is clearer. PMSaaS consist of a lot more pages, but there was not for all a mock-up made.

6 PAGE STRUCTURE

Here is a structure of all pages PMSaaS consists:

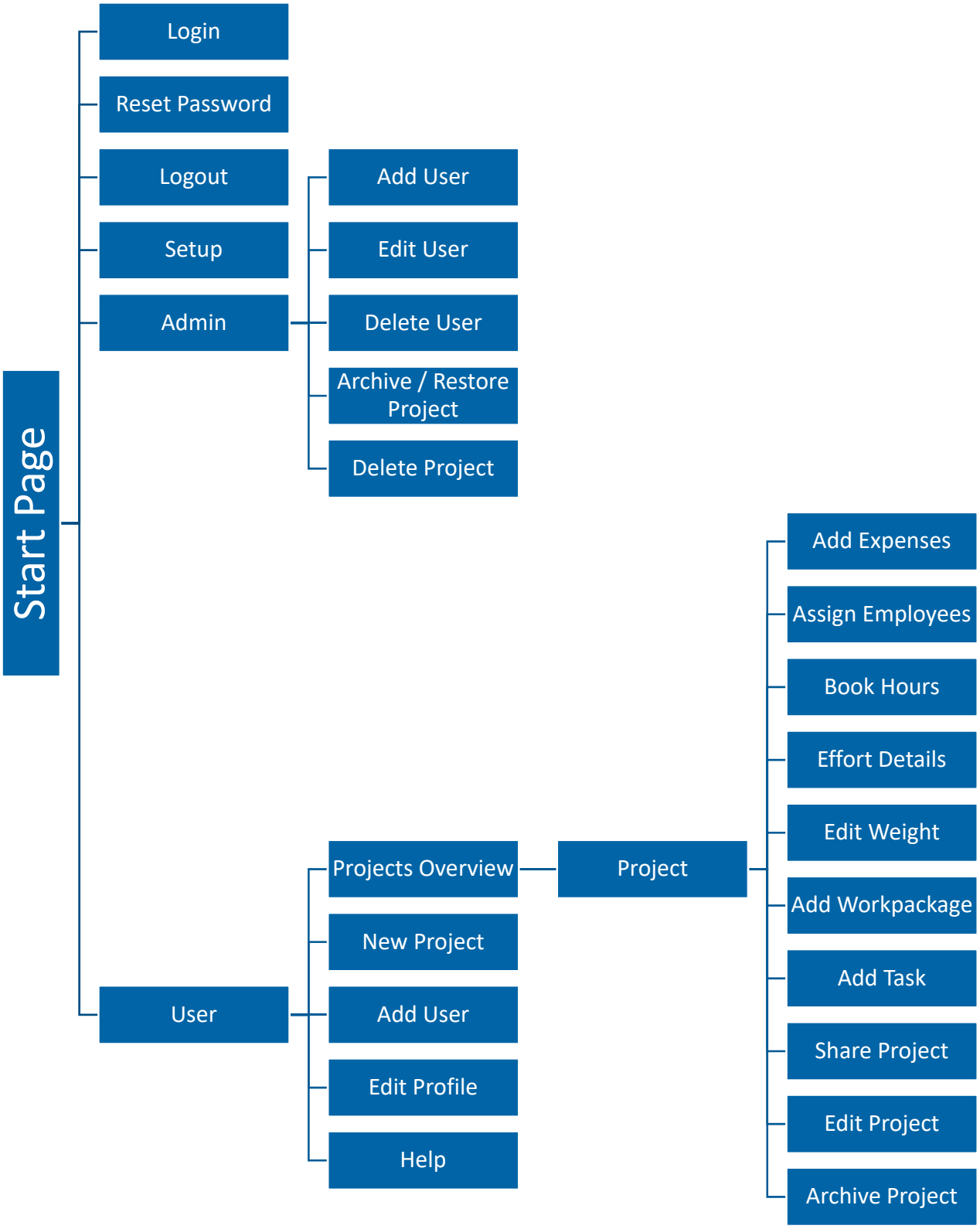


Figure 1 Structure PMSaaS

7 IMPLEMENTATION

This chapter will describe all classes that are used in PMSaaS ordered by package. All classes, that are not in the package ch.zhaw.init.walj.projectmanagement.util or its sub packages, are servlets that have only a doGet() or/and a doPost() method. Because of this reason, the methods have no own description.

7.1 ch.zhaw.init.walj.projectmanagement

7.1.1 Login

Page where users and the admin can log in to PMSaaS. Users can login with their kuerzel or e-mail address. The login page compares the entered data with the database and creates a session for the user if the login was correct.

7.1.2 Logout

If the user or admin clicked on the logout button, he will be forwarded to the logout page. The page destroys the current session and shows a link to go to the login page.

7.1.3 ResetPassword

ResetPassword creates a new password and sends it to the user as an e-mail. The password will be generated by the class PasswordGenerator and then sent to the user. After this, it will be encrypted and saved in the database.

7.2 ch.zhaw.init.walj.projectmanagement.admin

7.2.1 AdminFilter

This class is a filter that makes sure that only the admin can access the admin pages. If nobody is logged in it will redirect to the Login page. If another user, who is not the admin, is logged in, it redirects to the AccessDenied page.

7.2.2 Setup

The Setup page will be shown when there is no database found by PMSaaS. I this case, the user must declare where the server with MySQL running on it is located (URL + port), how the database should be named and what the login credentials are. In addition to that, the mail server must be declared and from what e-mail address users should get notifications. Last step is to enter the admin's e-mail address and choose a password. The class setup will then create the database with all its tables, write all information in a config file and create the admin user. The admin will receive an e-mail to confirm that PMSaaS was successfully installed and can be used now.

7.3 ch.zhaw.init.walj.projectmanagement.admin.properties

7.3.1 AddEmployee

The user can add an employee here. After entering all information, the page will create a new user in the database with a generated and encrypted password. The new user will receive an e-mail with the entered data and his password. If everything was successful, the page will return a message with the data of the employee.

7.3.2 AdminProperties

This is the first page the admin will see after the login page. Here he can change the logo and see all employees and projects in a list. The admin can choose an employee and edit or delete it. Projects can be edited, deleted, or archived / restored.

7.3.2.1 Change Logo

If the admin likes to change the logo he must upload 2 PNG files. One for the larger logo that is used on the start, login and logout page. The smaller one is used on all other pages. It is implemented like this so the admin can use 2 different versions of the logo.

7.3.3 ArchiveProject

Every project has an archive flag in the database. The ArchiveProject page changes this flag from 0 (not archived) to 1 (archived). The site will show a success or error message.

7.3.4 DeleteEmployee

The admin can delete employees if they don't have any projects or expenses or are assigned to some tasks. The admin will get a notification if the employee could not be deleted with a list of all reasons. If deleting the employee was successful, a message will inform the admin.

7.3.5 DeleteProject

Only the admin has the permission to delete projects. If a project is deleted, there is no possibility to restore it. The project will be deleted with all its work packages, tasks, expenses and assignments. A success or error message will be shown to the user.

7.3.6 EditEmployee

Employees can be edited by the admin. A form will be shown where he can edit name, kuerzel, e-mail and wage per hour. If the employee was successfully updated, a message with all new information will be shown, if an error occurred, PMSaaS shows an error message.

7.3.7 RestoreProject

The admin can restore archived projects. The RestoreProject page will restore an archived project and show an error or success message, depending on the project could be restored or not.

7.4 ch.zhaw.init.walj.projectmanagement.errorpages

7.4.1 AccessDenied

The AccessDenied page will be shown to a user who tried to access a page without the needed permission. It simply shows a message that the user is not allowed to visit the requested site.

7.4.2 ProjectNotFound

This site shows an error if someone tried to reach a project that does not exist (anymore).

7.5 ch.zhaw.init.walj.projectmanagement.user

7.5.1 EffortOverview

The EffortOverview site will give its user an overview about the effort in a project. This overview can be with all assigned employees included or with only one of them. If there is a get parameter employeeID, the page will only show the effort of this employee. A LineChart will visualize the hours of effort for the specific employee in each month. If there is no employee specified, the line chart will compare the planned (PMs) and booked effort.

7.5.2 Help

This page is for assisting the user in using PMSaaS. It's a helpful description with all essential functions of the tool.

7.5.3 Overview

The overview page is the first page a user sees. There are 3 lists: own projects, projects other people shared with me and own projects that I had archived. The lists are made with foundation accordion (look here for more information about that http://foundation.zurb.com/sites/docs/kitchen-sink.html#accordion). Overview will get all needed projects from the database and arrange them in the list. Every own project has a button to get to its ProjectOverview.

7.5.4 Profile

Profile creates a form with the user's data like name, kuerzel and e-mail. Further, the user can change his password here. The class Profile will update the information in the database and shows an error or success message.

7.5.5 ProjectOverview

The ProjectOverview class provides most of the usability of PMSaaS. It shows for ever project 4 panels:

- a short statistic with duration of the project, total budget, number of work packages, tasks and employees and the names of all partners
- a budget panel with a PieChart that compares the spent and remaining budget and a list of all expenses
- an effort panel with a LineChart that compares planned and booked effort and a list with all employees and the amount of their booked hours
- a panel with a Gantt chart that shows all work packages and tasks

Below these panels are three buttons to share, edit or archive the project.
While the first panel has no possibility of interaction, the others implement many different functions.

- Budget
    - AddExpense
- Effort
    - AssignEmployee
    - BookHours
    - Details employee (EffortOverview)
    - Details project (EffortOverview)
- Workpackages & Tasks
    - EditWeight
    - AddWorkpackage
    - AddTask

The ProjectOverview class gets all data from the database, creates the charts and fills the data into these panels.

7.6 ch.zhaw.init.walj.projectmanagement.user.add

7.6.1 AddEmployee

Every user can add new employees if the kuerzel and e-mail are not used yet. After fill in the information, the class will call the PasswordGenerator to generate a new password and send an e-mail with all data (including the password) to the new user.

7.6.2 AddExpense

The AddExpense class creates a form to record expenses. The expense types are defined in a separate Enum class. AddExpense gets all options from ExpenseTypes and writes them into a select field. The currency of the costs field is defined by the currency of the project. The expense will be added to the database.

7.6.3 AddProject

AddProject can create a new project with work packages and tasks in the database. The form consists of 3 parts:

1. Project information (name, duration, budget, etc.)
2. Workpackages
3. Tasks

The parts work packages and tasks each have a button to add more fields. Every click on one of these buttons calls a JavaScript function that adds a new group of fields under the existing ones.
An instance of DateFormatter controls if all dates are possible (start date before end date, etc.). If everything was ok, the project and its work packages and tasks will be added to the database.

7.6.4 AddTask

If needed, the project leader can add new tasks to his projects. This is nearly the same function as in AddProject. An instance of DateFormatter controls if all dates are possible (start date before end date, etc.). If everything was ok, the task will be added to the database.

7.6.5 AddWorkpackage

If needed, the project leader can add new work packages to his projects. This is nearly the same function as in AddProject. An instance of DateFormatter controls if all dates are possible (start date before end date, etc.). If everything was ok, the work package will be added to the database.

7.6.6 AssignEmployee

To book hours, an employee must be assigned. On the AssignEmployee page the user must choose an employee out of a list with all employees. With a click on the Choose Task button, the page ChooseTask will be opened. After choosing the task the user will be redirected back to AssignEmployee where the new assignment will be added to the database.

7.6.7 BookHours

BookHours lets the user choose from a list of employees, like in AssignEmployee. After choosing the employee it will redirect to 7.6.9ChooseTaskToBookHours where you can choose for which tasks you want to book hours. After fill in all hours it will redirect back to BookHours where the bookings will be written into the database.

7.6.8 ChooseTask

This page will be called from AssignEmployee to choose the task where the employee should be assigned to. It creates a list with all tasks of the project, to whom the employee is not assigned yet.

The user can choose more than one task, as multiple options are allowed. After choosing the task, ChooseTask will redirect back to AssignEmployee.

7.6.9 ChooseTaskToBookHours

After choosing the employee in BookHours, ChooseTaskToBookHours will be called. Like in ChooseTask, the user will see a list where he can choose the task where he wants to book hours. He can choose more than one task, as multiple options are allowed. A form will be created where the user can book hours for every task he chose. When all hours are filled in, the data will be sent to BookHours, where it will be written into the database.

7.7 ch.zhaw.init.walj.projectmanagement.user.delete

7.7.1 ArchiveProject

For better finding the current projects, users can archive completed projects. ArchiveProject will change the flag in the database from 0 (not archived) to 1 (archived).

7.7.2 DeleteEffort

DeleteEffort will delete the chosen booking in the database and write a success or error message in the browser.

7.7.3 DeleteExpense

The chosen expense will be deleted from the database and a success or error message will be shown.

7.7.4 DeleteTask

Users can delete tasks after being warned that with the task, all bookings and assignments will be deleted too. DeleteTask will delete the chosen task and print a success or error message.

7.7.5 DeleteWorkpackage

Like in DeleteTask, the user will be warned that all tasks and bookings of a work package will be deleted too. The chosen work package will then be deleted and a success or error message will be shown to the user.

7.8 ch.zhaw.init.walj.projectmanagement.user.edit

7.8.1 Edit

The edit page is the starting point for editing and deleting various parts of the project. Edit consists of the following parts:

- Project
- Workpackages
- Tasks
- Expenses
- Effort

If you edit one of the entries, Edit will send the edited data to another page that will update the database.

7.8.2 EditEffort

Updates the effort with the given data in the database and returns a success or error message.

7.8.3 EditExpense
Updates the expense with the given data in the database and returns a success or error message.

7.8.4 EditProject
Updates the project with the given data in the database and returns a success or error message.

7.8.5 EditTask
Updates the task with the given data in the database and returns a success or error message.

7.8.6 EditWeight
Creates new entries or updates existing ones in the weight table of the database.

7.8.7 EditWorkpackage
Updates the work package with the given data in the database and returns a success or error message.

7.9 ch.zhaw.init.walj.projectmanagement.user.share

7.9.1 ShareProject
ShareProject will first create a list with all employees, with whom the project is not shared yet. The user can then decide with which of the listed employees he likes to share his project. He can choose more than one, as multiple options are allowed. For the chosen employees will be an entry in the share table of the database created. They will see the project now in their overview under “Other Projects”.

7.10 ch.zhaw.init.walj.projectmanagement.util

7.10.1 DataBaseAccess

Name	Type	Description	public/private
url	String	the url of the MySQL database	private
dbname	String	the name of the MySQL database	private
username	String	name of the user of the MySQL database	private
password	String	password of the MySQL user	private

Table 1 Variables DataBaseAccess

Name	Description	public / private	return
DataBaseAccess(String path)	Constructor, reads .config file and initializes the variables url, dbname, username and password	public	-
getUrl()	returns the content of url	public	url
getDbname()	returns the content of dbname	public	dbname
getUsername()	returns the content of username	public	username
getPassword()	returns the content of password	public	password

Table 2 Methods DataBaseAccess

7.10.2 DBConnection

Name	Type	Description	public/private
conn	Connection	Connection to the database	private
st	PreparedStatement	Statement for MySQL queries	private
res	ResultSet	variable to save the result of a query	private
noConnection	boolean	true if DBConnection could no connect to Database	private

Table 3 Variables DBConnection

Name	Description	public / private	return
DBConnection(String path)	Constructor, creates a connection to the database and initializes noConnection	public	-
getProject(int pID)	gets the project with the given ID from the Database and creates a new Project object	public	Project object
getProjects(int id, boolean archive)	creates a list with all projects where the given user is project leader. If archive is true, only archived projects, else only not archived projects will be in the list	public	list of projects
getWorkpackages(int id)	creates a result set with all work packages of the project	private	result set with all work packages
getTasks(int id)	creates a result set with all tasks of the project	private	result set with all tasks
getEmployee(int id)	creates an employee object from the employee with the given ID	public	Employee object
getSharedEmployees(int projectID)	creates a list with all employees with whom the project is shared	public	list of employees
getSharedProjects(int id)	get all projects that are shared with the given employee	public	list of projects
getAllEmployees()	creates a list with all employees (without administrator)	public	list of employees
getAssignedTasks(int employee)	get all tasks the given employee is assigned to	public	list of task IDs
getAssignments(int taskID)	get all assignments to the given task	public	list of assignments
getAssignment(int employeeID, int taskID)	get the assignment with the given employee and task	public	assignment object
getExpenses(int id)	list with the IDs of all expenses of the given employee	public	list of expense IDs
getBookings (Assignment assignment)	get all bookings to a specific assignment	public	list of bookings
getUsedBudget(Project project)	calculates the used budget of a project	public	used budget
getRemainingBudget(Project project)	calculates the remaining budget of a project	public	remaining budget
findUser(String user, String password)	tries to find a user in the database	public	employee object or null
findUser(String user)	tries to find a user in the database	public	employee object or null

newProject(String pName, String pShortname, int pLeader, String pBudget, String pCurrency, String pStart, String pEnd, String pPartners)	creates a new project in the database	public	ID of the new project
newWorkpackage(int projectIDFS, String wpName, String wpStart, String wpEnd)	creates a new work package in the database	public	-
newTask(int projectID, String wpName, String taskName, String taskStart, String taskEnd, String taskPM, String taskBudget)	creates a new task in the database	public	-
newTask(int wpID, String taskName, String taskStart, String taskEnd, String taskPM, String taskBudget)	creates a new task in the database	public	-
newEmployee(int employeeID, String firstname, String lastname, String kuerzel, String mail, int wage)	creates a new employee and wage in the database, password will be generated	public	employee object
newEmployee(int employeeID, String firstname, String lastname, String kuerzel, String mail, String password, int wage)	creates a new employee and wage in the database, password is known	public	employee object
newExpense(int projectID, int employeeID, double costs, String type, String description, String date)	creates new expense in the database	public	-
newAssignment(int taskID, int employeeID)	creates new assignment in the database	public	-
newBooking(int assignment, int month, double hours)	creates new booking in the database	public	-
newWage(int userID, double wage, String date)	creates new wage in the database	public	-
newWeight(int taskID, int month, double weight)	creates new weight in the database	public	-
newShare(int projectID, int employeeID)	creates new share in the database	public	-
updateProject(int id, String name, String shortname, double budget, String currency, String start, String end, String partners)	updates the project with the given data	public	-
updateWorkpackage(int id, String name, String start, String end)	updates the work package with the given data	public	-

updateTask(int id, String name, String start, String end, int pm, double budget, int wp)	updates the task with the given data	public	-
updateExpense(int id, int employee, double costs, String type, String description, String date)	updates the expense with the given data	public	-
updateEffort(int id, String month, String hours)	updates the effort with the given data	public	-
updateUser(int userID, String firstname, String lastname, String kuerzel, String mail)	updates the user with the given data	public	-
updatePassword(int userID, String password)	updates the password of the user	public	-
updateWeight(int taskID, int month, double weight)	updates a weight of a task and month	public	-
archiveProject(int projectID)	sets the archive flag of a project to 0	public	-
restoreProject(int projectID)	sets the archive flag of a project to 1	public	-
deleteWorkpackage(int workpackageID)	deletes a work package (and all its tasks, assignments and bookings)	public	-
deleteTask(int taskID)	deletes a task (and all its assignments and bookings)	public	-
deleteExpense(int expenseID)	deletes an expense	public	-
deleteEffort(int effortID)	deletes an effort entry	public	-
deleteEmployee(int id)	deletes an employee	public	-
deleteProject(int projectID)	deletes a project with all its work packages, tasks, assignments, bookings, expenses and shares	public	-

Table 4 Methods DBConnection

7.10.3 Effort

Name	Type	Description	public/private
tasks	ArrayList<Task>	ArrayList that can be filled with tasks	private
con	DBConnection	Connection to MySQL database	private final

Table 5 Variables Effort

Name	Description	public / private	return
Effort (ArrayList<Task> tasks, String path)	Constructor, initializes tasks and creates a connection to the database	public	-
getBookings ()	creates an ArrayList with all assignments to the tasks and then creates an ArrayList with all bookings to the assignments	public	ArrayList of bookings
getBookings (int employeeID)	creates an ArrayList with all bookings of a specific employee in these tasks	public	ArrayList of Bookings
getPlannedEffort (double month)	calculates the planned effort for the given month	public	planned effort

getBookedEffort (double month)	get the booked effort for a specific month from all tasks	public	booked effort
getEffortPerEmployee(int employee)	get the effort of a specific employee in this project	public	effort in hours
getBookedEffortPerMonth(double month, int employee)	get the booked effort of a specific employee within the given month	public	effort in hours

Table 6 Methods Effort

7.10.4 ExpenseTypes

ExpenseTypes is an Enum which contains all possible expense types of PMSaaS.

Variable	Output
TRAVEL	Travel
OVERNIGHT_STAY	Overnight Stay
MEALS	Meals
OFFICE_SUPPLIES	Office Supplies
EVENTS	Events

Table 7 Variables ExpenseTypes

7.10.5 HTMLFooter

Name	Type	Description	public/private
instance	HTMLFooter	an instance of the HTMLFooter	private static

Table 8 Variables HTMLFooter

Name	Description	public / private	return
printFooter(boolean linkToTop)	creates the footer as a string, if linkToTop is true with a link to go to the top of the page	public	String of the footer
getInstance()	if instance is not initialized yet, getInstance() initializes it and returns the instance	public static	instance of HTMLFooter

Table 9 Methods HTMLFooter

7.10.6 HTMLHeader

Name	Type	Description	public/private
instance	HTMLHeader	an instance of the HTMLHeader	private static

Table 10 Variales HTMLHeader

Name	Description	public / private	return
printHeader(…)	creates the header as a string, following attributes are possible:	public	String of the footer
	String tabTitle, String path		
	String tabTitle, String path, String title, String script		
	String tabTitle, String path, String script		
	String tabTitle, String path, String title, String script, String link, boolean admin		
	String tabTitle, String path, String title, String script, String link, boolean admin, boolean logout		
getInstance()	if instance is not initialized yet, getInstance() initializes it and returns the instance	public static	instance of HTMLFooter

Table 11 Methods HTMLHeader

7.10.7 LoginFilter

LoginFilter is an implementation of the Java Servlet Filter class. It checks at every page call if the user is logged in. If not it redirects to the Login page. The LoginFilter also checks if the user is an administrator or a normal user and redirects to the right page.

7.10.8 Mail

Name	Type	Description	public/private
mailFrom	String	address that will be shown as the sender of the e-mail	private static
host	String	URL of the mail server	private static
message	MimeMessage	the mail that will be sent	private
user	Employee	the recipient	private

Table 12 Variables Mail

Name	Description	public / private	return
Mail (Employee user, String path)	constructor, reads mailconfig and initializes the variables	public	-
sendWelcomeMail()	sends a welcome mail to a new user with its data	public	-
sendNewPassword()	sends a mail with a newly generated password	public	-
sendInitialSetupMail()	sends a mail to the admin as a confirmation that PMSaaS was successfully initialized	public	-

Table 13 Methods Mail

7.11 ch.zhaw.init.walj.projectmanagement.util.chart

7.11.1 GanttChart

Name	Type	Description	public/private
project	Project	the project of which a GanttChart should be created	private final
workpackages	ArrayList<Workpackage>	list of workpackages	private final
tasks	ArrayList<Task>	list of tasks	private final
path	String	location of PMSaaS	private final
nbrOfObjects	int	needed to calculate height of GanttChart, number of work packages and tasks that are added in the dataset	private

Table 14 Variables GanttChart

Name	Description	public / private	return
GanttChart(Project project, String path)	constructor, initializes the variables	public	-
createDataset()	creates dataset with all workpackages and tasks of the project	private	dataset
createChart()	creates a Gantt chart with the data from the dataset and saves it as a JPEG file	public	-

Table 15 Methods GanttChart

7.11.2 LineChart

Name	Type	Description	public/private
project	Project	the project of which a LineChart should be created	private final
path	String	location of PMSaaS	private final
tasks	ArrayList<Task>	list of tasks	private

Table 16 Variables LineChart

Name	Description	public / private	return
LineChart(Project project, String path)	constructor, initializes the variables	public	-
createDataset()	creates dataset with the booked and planned effort of the project	private	dataset
createDataset(int employeeID)	creates dataset with the booked effort of the given employee	private	dataset
createChart()	creates a line chart with the data from the dataset and saves it as a JPEG file	public	-
createChart(int employeeID)	creates a line chart with the data from the dataset and saves it as a JPEG file	public	-

Table 17 Methods LineChart

7.11.3 PieChart

Name	Type	Description	public/private
project	Project	the project of which a PieChart should be created	private final
con	DBConnection	connection to the database	private final
path	String	location of PMSaaS	private final

Table 188 Variables PieChart

Name	Description	public / private	return
PieChart(Project project, String path)	constructor, initializes the variables	public	-
createChart()	creates a pie chart with the spent and remaining budget and saves it as a JPEG file	public	-

Table 199 Methods PieChart

7.12 ch.zhaw.init.walj.projectmanagement.util.dbclasses

7.12.1 Assignment

Name	Type	Description	public/private
id	int	the ID of the assignment	private final
taskID	int	the ID of the task where the assignment belongs to	private final
employeeID	int	the ID of the employee who is assigned to the task	private final

Table 20 Variables Assignment

Name	Description	public / private	return
Assignment(int id, int taskID, int employeeID)	constructor, initializes the variables	public	-
getID()	returns id	public	id

getTaskID()	returns taskID	public	taskID
getEmployeeID()	returns employeeID	public	employeeID

Table 21 Methods Assignment

7.12.2 Booking

Name	Type	Description	public/private
bookingID	int	the ID of the booking	private final
month	int	number of the month of the booking	private final
hours	double	number of hours	private final
taskID	int	ID of the task where the hours were booked	private final
employeeID	int	ID of the employee that booked the hours	private final

Table 22 Variables Booking

Name	Description	public / private	return
Booking(int bookingID, int assignmentID, int month, double hours, int taskID, int employeeID)	constructor, initializes the variables	public	-
getID()	returns bookingID	public	bookingID
getMonth()	returns month	public	month
getHours()	returns hours	public	hours
getTaskID	returns taskID	public	taskID
getEmployeeID	returns employeeID	public	employeeID

Table 23 Methods Booking

7.12.3 Employee

Name	Type	Description	public/private
id	int	the ID of the employee	private final
firstname	String	first name of the employee	private final
lastname	String	last name of the employee	private final
kuerzel	String	kuerzel of the employee	private final
mail	String	e-mail address of the employee	private final
password	String	password of the employee	private
wage	double	wage of the employee	private

Table 24 Variables Employee

Name	Description	public / private	return
Employee(int id, String firstname, String lastname, String kuerzel, String mail, String password, int wage, int supervisor)	constructor, initializes the variables	public	-
getID()	returns id	public	id
getFirstName()	returns firstname	public	firstname
getLastName()	returns lastname	public	lastname
getName()	returns firstname lastname	public	firstname + lastname

getFullName()	returns lastname, firstname	public	lastname + firstname
getKuerzel()	returns kuerzel	public	kuerzel
getMail()	returns mail	public	mail
getPassword()	returns password	public	password
setNewPassword(String password)	sets new password	public	-
getWage()	returns wage	public	wage
setWage(double wage)	sets new wage	public	-

Table 25 Methods Employee

7.12.4 Expense

Name	Type	Description	public/private
id	int	the ID of the expense	private final
projectID	int	ID of the project the expense belongs to	private final
employeeID	int	ID of the employee who booked the expense	private final
costs	double	costs of the expense	private final
type	String	type of the expense	private final
description	String	additional information about the expense	private final
date	String	date of the expense	private final

Table 26 Variables Expense

Name	Description	public / private	return
Expense (int id, int projectID, int employeeID, double costs, String type, String description, String date)	constructor, initializes the variables	public	-
getID()	returns id	public	id
getEmployeeID()	returns employeeID	public	employeeID
getCosts()	returns costs	public	costs
getType()	returns type	public	type
getDescription()	returns description	public	description
getDate()	returns date	public	date

Table 27 Methods Expense

7.12.5 Project

Name	Type	Description	public/private
id	int	the ID of the project	private final
shortname	String	short name of the project	private final
name	String	name of the project	private final
leader	int	ID of the project leader	private final
start	String	start date of the project	private final
end	String	end date of the project	private final
currency	String	currency of the project (CHF or EUR)	private final
budget	double	total budget of the project	private final
tasks	ArrayList<Task>	list with all tasks of the project	private
workpackages	ArrayList<Workpackage>	list with all work packages of the task	private
employees	ArrayList<Employee>	list with all employees assigned to the project	private

expenses	ArrayList<Expense>	list with all expenses of the project	private
partner	String	project partners	private final

Table 28 Variables Project

Name	Description	public / private	return
Project(int id, String name, String shortname,int leader, String start, String end, String currency, double budget, String partner)	constructor, initializes the variables	public	-
addWorkpackage(Workpackage wp)	adds a workpackage to workpackages	public	-
getID()	returns id	public	id
getShortname()	returns shortname	public	shortname
getName()	returns name	public	name
getLeader()	returns leader	public	leader
getStart()	returns start	public	start
getEnd()	returns end	public	end
getDuration()	retuns start – end	public	start – end
getNumberOfMonths()	returns the number of months of the project duration	public	number of months
getBudget()	returns budget	public	budget
getPartners()	returns partner	public	partner
addEmployees()	adds all assigned employees to the project	public	-
getEmployees()	returns employees	public	employees
getEmployee(int id)	returns the employee with the given ID or null if not found	public	employee or null
nbrOfEmployees()	returns size of employees	public	size of emplyoees
getCurrency()	returns currency	public	currency
getWorkpackages()	returns list of work packages	public	workpackages
nbrOfWorkpackages()	returns size of workpackages	public	size of workpackages
getTasks()	returns tasks	public	tasks
getTask(int id)	returns task with the given ID or null	public	task of null
nbrOfTasks()	returns size of tasks	public	size of tasks
addExpense(Expense expense)	adds the expense to expenses	public	-
getExpenses()	returns expenses	public	expenses
getTotalExpenses()	returns total costs of all expenses	public	total of all expenses
getWorkpackage(int taskWP)	returns work package with the given ID or null if not found	public	work package

Table 29 Methods Project

7.12.6 Task

Name	Type	Description	public/private
id	int	the ID of the task	private final
workpackageID	int	ID of the work package the task belongs to	private

workpackageName	String	name of the work package the task belongs to	private
name	String	name of the task	private final
start	String	start date of the task	private final
projectStart	String	start date of the project	private final
end	String	end date of the task	private final
pms	int	number of person months	private final
budget	double	budget of the task	private final
employees	ArrayList<Employee>	list with all employees assigned to the task	private final
weights	ArrayList<Weight>	list with all weights of the task	private

Table 30 Variables Task

Name	Description	public / private	return
Task(int id, int workpackageID, String name, String start, String projectStart, String end, int pms, double budget, ArrayList<Weight> weights	constructor, initializes the variables, used when the ID of the work package is known	public	-
Task(int id, String workpackageName, String name, String start, String projectStart, String end, int pms, double budget, ArrayList<Weight> weights)	constructor, initializes the variables, used when only the name of the work package is known	public	-
addEmployee(Employee employee)	adds the given employee to employees	public	-
getID()	returns id	public	id
getWorkpackageID()	returns workpackageID	public	workpackageID
getName()	returns name	public	name
getStart()	returns start	public	start
getStartAsDate()	returns start as a Date object	public	Date object
getEnd()	returns end	public	end
getEndAsDate()	returns end as a Date object	public	Date object
getPMs()	returns pms	public	pms
getBudget()	returns budget	public	budget
getEmployees()	returns employees	public	employees
nbrOfEmployees()	returns size of employees	public	size of employees
getStartMonth()	returns number of the month where the task starts	public	number of start month
getEndMonth()	returns number of the month where the task ends	public	number of end month
getNumberOfMonths()	returns the number of months of the task duration	public	number of months

getPMsPerMonth()	returns the calculated person months for every month	public	PMs per month
getWeight()	returns weights	public	weights
getWeight(double month)	returns the weight of the given month	public	weight of month

Table 31 Methods Task

7.12.7 Weight

Name	Type	Description	public/private
id	int	the ID of the weight	private final
taskIDFS	int	ID of the task the weight belongs to	private final
month	int	number of the month the weight belongs to	private final
weight	double	weight	private final

Table 32 Variables Weight

Name	Description	public / private	return
Weight (int id, int taskIDFS, int month, double weight)	constructor, initializes the variables	public	-
getMonth()	returns month	public	month
getWeight()	returns weight	public	weight

Table 33 Methods Weight

7.12.8 Workpackage

Name	Type	Description	public/private
id	int	the ID of the work package	private final
name	String	name of the work package	private final
start	String	start date of the work package	private final
end	String	end date of the work package	private final
tasks	ArrayList<Task>	list with all tasks belonging to the work package	private final
employees	ArrayList<Employee>	list of all employees assigned to the work package	private final

Table 34 Variables Workpackage

Name	Description	public / private	return
Workpackage(int id, String name, String start, String end)	constructor, initializes the variables	public	-
getID()	returns id	public	id
getName()	returns name	public	name
getStart()	returns start	public	start
getStartAsDate()	returns start as a Date object	public	Date object
getEnd()	returns end	public	end
getEndAsDate()	returns end as a Date object	public	Date object
addTask(Task task)	adds given task to tasks	public	-
nbrOfTasks()	returns size of tasks	public	size of tasks
getTasks()	returns tasks	public	tasks
addEmployees()	adds the employees of the tasks to emplyees	public	-

nbrOdEmployees()	returns size of employees	public	size of employees
getEmployees()	returns employees	public	employees

Table 35 Methods Workpackage

7.13 ch.zhaw.init.walj.projectmanagement.util.format

7.13.1 DateFormatter

Name	Type	Description	public/private
instance	DateFormatter	an instance of DateFormatter	private static

Table 36 Variables DateFormatter

Name	Description	public / private	return
getInstance()	returns an instance of DateFormatter	public static	instance
formatDate(String unformattedDate)	converts 'YYYY-MM-DD' to 'DD.MM.YYYY'	public	formatted date as String
formatDateForDB(String unformattedDate)	converts 'DD.MM.YYYY' to 'YYYY-MM-DD'	public	formatted date as String
formatDateForDB(Date unformattedDate)	formats a Date object to a string with format 'YYYY-MM-DD'	public	formatted date as String
getMonthsBetween(String start, String end)	calculates number of months between start and end	public	number of months
getDaysBetween(Date startDate, String end)	calculates number of days between startDate and end	public	number of days
getMonths(Date start, int nbrOfMonths)	Writes a defined number of months beginning with the start date in an array. Every month as '01.09.2016' and 'September 2016'	public	2-dimensional Array with all months
getMonthStrings(Date start, int nbrOfMonths)	get names of months (like 'September 2016')	public	String array with all months
stringToDate(String dateString, String formatString)	formats a string with the given format to a Date object	public	Date object
checkDate(String firstDate, String secondDate, String formatString)	Checks if firstDate is before or the same as secondDate.	public	true or false

Table 37 Methods DateFormatter

7.13.2 NumberFormatter

Name	Type	Description	public/private
instance	NumberFormatter	an instance of NumberFormatter	private static

Table 38 Variables NumberFormatter

Name	Description	public / private	return

getInstance()	returns an instance of NumberFormatter	public static	instance
formatDouble(double number)	returns given number as String, rounded to two decimal places	public	number as String
formatHours(double number)	returns given number as String, rounded to zero decimal places	public	number as String

Table 39 Methods NumberFormatter

7.14 ch.zhaw.init.walj.projectmanagement.util.password

7.14.1 PasswordGenerator

Name	Type	Description	public/private
instance	PasswordGenerator	an instance of PasswordGenerator	private static

Table 40 Variables PasswordGenerator

Name	Description	public / private	return
getNewPassword()	generates a new password with 8 random characters	public	password
getInstance()	returns instance of PasswordGenerator	public static	instance

Table 41 Methods PasswordGenerator

7.14.2 PasswordService

Name	Type	Description	public/private
instance	PasswordService	an instance of PasswordService	private static

Table 42 Varibles PasswordService

Name	Description	public / private	return
encrypt(String password)	encrypts the given password	public	password
getInstance()	returns instance of PasswordService	public static	instance

Table 43 Methods PasswordService