

IPA Bericht

Webtool zur Arbeitszeiterfassung

**Zürich, Donnerstag, 27. Mai 2021**

**Lars Gächter**

Inhaltsverzeichnis

[Teil 1 Umfeld und Ablauf 4](#_Toc73049187)

[1.1. Ausgangslage 4](#_Toc73049188)

[1.2. Detaillierte Aufgabenstellung 4](#_Toc73049189)

[1.3. Projektaufbauorganisation 7](#_Toc73049190)

[1.4. Projektmanagement Methode 7](#_Toc73049191)

[1.5. Mittel und Methoden 7](#_Toc73049192)

[1.6. Vorkenntnisse 7](#_Toc73049193)

[1.7. Vorarbeiten 7](#_Toc73049194)

[1.8. Neue Lerninhalte 8](#_Toc73049195)

[1.9. Arbeiten in den letzten 6 Monaten 8](#_Toc73049196)

[1.10. Dokumentablage 8](#_Toc73049197)

[1.11. Zeitplan 8](#_Toc73049198)

[1.12. Arbeitsprotokoll 10](#_Toc73049199)

[1.12.1. Tagesprotokoll vom 12.5.2021 10](#_Toc73049200)

[1.12.2. Tagesprotokoll vom 14.5.2021 12](#_Toc73049201)

[1.12.3. Tagesprotokoll vom 17.5.2021 14](#_Toc73049202)

[1.12.4. Tagesprotokoll vom 18.5.2021 15](#_Toc73049203)

[1.12.5. Tagesprotokoll vom 19.5.2021 16](#_Toc73049204)

[1.12.6. Tagesprotokoll vom 20.5.2021 17](#_Toc73049205)

[1.12.7. Tagesprotokoll vom 21.5.2021 18](#_Toc73049206)

[1.12.8. Tagesprotokoll vom 25.5.2021 19](#_Toc73049207)

[1.12.9. Tagesprotokoll vom 26.5.2021 20](#_Toc73049208)

[1.12.10. Tagesprotokoll vom 27.5.2021 21](#_Toc73049209)

[1.12.11. Tagesprotokoll vom 28.5.2021 22](#_Toc73049210)

[Teil 2 Projekt-Dokumentation 23](#_Toc73049211)

[2.1. Kurzfassung 23](#_Toc73049212)

[2.2. Informieren 24](#_Toc73049213)

[2.3. Planen 29](#_Toc73049214)

[2.3.1. Testkonzept 36](#_Toc73049215)

[2.4. Entscheiden 37](#_Toc73049216)

[2.5. Realisieren 37](#_Toc73049217)

[2.6. Kontrollieren 39](#_Toc73049218)

[2.6.1. Testprotokoll 44](#_Toc73049219)

[2.7. Auswerten 48](#_Toc73049220)

[2.7.1. Reflexion 48](#_Toc73049221)

[2.7.2. Schlusswort 52](#_Toc73049222)

[2.8. Glossar 53](#_Toc73049223)

[2.9. Quellenverzeichnis 53](#_Toc73049224)

[2.10. Literaturverzeichnis 53](#_Toc73049225)

[Teil 3 Anhang 54](#_Toc73049226)

[3.1. Abbildungsverzeichnis 231](#_Toc73049227)

[3.2. Tabellenverzeichnis 231](#_Toc73049228)

[3.3. Listing des Programmcodes 231](#_Toc73049229)

# Umfeld und Ablauf

## Ausgangslage

Die Rafisa Informatik bietet seit diesem Jahr ein betreutes Wohnen an. Die BetreuerInnen für das Wohnen absolvieren mehrmals über den Tag verteilt ihre Arbeitseinsätze. Diesen Fall deckt die Applikation die die Rafisa für die Zeiterfassung nutzt nicht ab. Da die Applikation "monolithisch" strukturiert ist und der Ursprüngliche Entwickler mitlerweile pensioniert ist die Applikation nur schwer erweiterbar. Deshalb wünscht sich die Geschäftsleitung eine Applikation mit der die WohnbetreuerInnen ihre Arbeitszeiten dokumentieren können.

## Detaillierte Aufgabenstellung

Code-Konventionen:

- Namensgebung analog zu Java. Klassen sind 'upper camel case', Methoden und Variabeln sind 'lower camel case' und Konstanten 'upper case underscore seperated'.

-Zugriff auf Variabeln nur über Methoden

- Authentifizierung: Benutzer müssen sich über das LDAP der Rafisa authentifizieren können.

- Eingeloggte Benutzer können die Zeiterfassung starten falls sie nicht aktiv ist und beenden falls sie aktiv ist

- Pro Tag können mehrere Arbeitseinträge pro Benutzer erfasst werden.

- Am Tagesende müssen Aufgrund der Einträge das jeweilige Tages-, Wochen- und Monatssaldo berechnet werden.

Dokumentation:

Zur Planung soll ein Use-Case Diagramm Verwendet werden.

Damit das Programm später erweiterbar ist soll auch ein ERM und ein Aktivitätsdiagramm erstellt werden.

Eingabemasken:

Die Eingabemasken sind analog zu denen in der regulären Zeiterfassung zu halten.

Loginmaske:

-Ein Textfeld für den Benutzernamen.

-Ein Passwortfeld für das Passwort

-Einen Button zum Absenden des Formulars

Zeiterfassungsmaske:

-Einen Button mit der Beschriftung "Zeiterfassung starten" falls die Zeiterfassung nicht gestartet ist.

-Einen Buttom mit der Beschriftung "Zeiterfassung stoppen" falls die Zeiterfassung gestartet ist.

Übersichtsmaske:

-Eine Tabelle mit den Daten des Ausgewählten Monats. Die Einträge sollen in chronologischer Reihenfolge dargestellt werden. Titel der Tabelle soll der ausgewählte Monat und die Jahreszahl sein und als Standardwert soll der derzeitige Monat gesetzt sein.

Links soll zum vorherigen Monat navigiert werden können, rechts zum folgenden. Wenn der aktuelle Monat ausgewählt ist soll das nach vorne Navigieren deaktiviert und ausgegraut sein.

-Die Einträge sollen für jeden Arbeitseinsatz folgende Attribute auflisten: Datum des Eintrages, Startzeit des Eintrages, Endzeit des Eintrages, Soll- und Istarbeitszeit des dazugehörigen Tages sowie die Differenz dieser zwei Werte und die Zeitsaldi für die jeweilige Woche, Monat und Total. Daneben noch jeweils ein Feld für allfällige Textkommentare. Neben jedem Eintrag soll ein klickbares Symbol sein das beim anklicken die Maske zur Korrektur des Eintrages aufruft.

-Unter den Einträgen der Arbeitszeit soll eine Zeile das Totalwerte dieses Monats angezeigt werden.

Korrekturmaske:

Die Korrekturmaske soll erneut alle Attribute eines Eintrages auflisten. Neben jedem Eintrag soll ein Feld zur Korrektur des Wertes vorhanden sein. Falls in einem oder mehreren Feldern Korrekturen vorgenommen werden soll der Kommentar ein Pflichtfeld sein. In einer dritten Spalte soll jeweils eine kurze Erklärung des jeweiligen Attributes stehen. Die Beschreibung kann 1:1 aus der vorhandenen Zeiterfassung kopiert werden. Zusätzlich soll angezeigt werden wann und von welchem Benutzer zuletzt eine Korrektur für diesen Eintrag vorgenommen worden ist.

Im Falle einer Korrektur ist ein Textkommentar zwingend.

Tägliche verarbeitung:

Am Ende jeden Tages soll das Saldo ausgerechnet werden. Arbeitsstunden an Sonn- und Feiertagen werden 1.5-fach gezählt.

Fehlerbehandlung

- Ist keinen Kontakt zum LDAP möglich, soll eine Fehlermeldung erscheinen die den Benutzer auf diesen Umstand aufmerksam macht.

- Bei Erhalt von Fehlerhaften Daten, wird eine Fehlermeldung angezeigt die auf die Inkompatibilität der

Daten aufmerksam macht.

- Bei internen PHP-Fehlermeldungen sollen diese nicht angezeigt werden. Werden solche

Fehlermeldungen generiert, muss der Programmablauf abgebrochen werden und zur „Einloggen“ Maske navigiert

werden.

Tests

Die Aufgabe umfasst die Installation eines Webauftrittes der funktionell ist. Das Ganze soll ausgiebig getestet werden.

- Dazu ist ein Testkonzept zu erstellen.

- Die Tests sind gemäss Konzept durchführen.

Mindestens folgende Testszenarien müssen durchgeführt werden:

- Korrekte Funktion der Masken.

- Fehler abfangen.

- Fehlermeldungen nach Vorgabe anzeigen.

## Projektaufbauorganisation

Graphical user interface, application

Description automatically generated

Abbildung : Baumdiagramm

## Projektmanagement Methode

## Mittel und Methoden

-Windows oder Linux-PC mit XAMP oder LAMP

PHPStorm

PHP/Symfony/Twig/Composer

HTML/CSS/Dart/Sass

Javascript/NodeJS

Git

MongoDB

LDAP

## Vorkenntnisse

PHP

HTML/CSS/Dart/Sass

Javascript/NodeJS

Git

## Vorarbeiten

Kennenlernen der LDAP authentifizierung.

Einrichten einer virtualisierten Arbeitsumgebung.

Installation von MongoDB

Evaluation der Anforderungen an die Zeiterfassung

## Neue Lerninhalte

LDAP api

MongoDB

## Arbeiten in den letzten 6 Monaten

-Webapplikation zum Ausführen und korrigieren von Mathematiktests

-Grafische Darstellung von "Nested Sets" in Kuchendiagrammen

Dafür wurden folgende Technologien verwendet:

PHPStorm

Git

VScode

CSS

javascript

## Dokumentablage

**Projekt Initialisierung**

* Ich mache jeden Tag einen Git Commit oder mehrere Git Commits.
* GitLab Repository erstellen und Vorgesetzten als Maintainer aufgenommen.
* Gitignore erstellt, um generierte Dateien von Composer, npm, PhpStorm und Autoloader im Git zu ignorieren.
* Ordnerverzeichnis “doc” für die Dokumentation angelegt und darin die Dokumentation als Word und das Gantt als Excel erstellt.
* Zeitplan anfangen zu erstellen.
* Dokumentation angefangen zu erstellen.

## Zeitplan



Abbildung : Projektzeitplan

## Arbeitsprotokoll

### Tagesprotokoll vom 12.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Das Analysieren der Aufgabenstellung und Zusatzinformationen einholen oder Ergänzungen zur Aufgabenstellung.  Wenn möglich alle Tätigkeiten festlegen.  Zeitplan zu den erfassten Tätigkeiten erstellen.  Besuchstag 1. Expertengespräch | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 09:00- 10:00 | Projekt Initialisierung | Hat alles funktioniert. | Mein Fachvorgesetzter Stefan Kuhn hat mir geholfen einen Namen für das Gitlab Repository festzulegen und wie Ihn als Maintainer vom Repository einzuladen. |
| 10:00-12:00 | Analysieren der Aufgabenstellung  Einlesen verstehen und in anderen Worten zusammenfassen. | Hat alles funktioniert. | Keine. |
| 13:00-13:30 | Analysieren der Aufgabenstellung  Einlesen verstehen und in anderen Worten zusammenfassen. | Hat alles funktioniert. | Keine. |
| 13:30-14:30 | Ergänzungen zur Aufgabenstellung.  Nachfolgende Informationen in der Dokumentation erwähnt,   * Abmeldung / Angemeldet * Masken-Navigation * Übersicht-Navigation * Mobiltauglichkeit * Übersicht Monatstotal * Benutzer Pflichtzeit * Korrekturmaske im Detail | Hat alles funktioniert. | Keine. |
| 14:30-16:00 | Zeitplan erstellen |  |  |
| 16:00-17:00 | Experten Gespräch |  |  |
| 17:00-18:00 | Zeitplan erstellen und Zusatzinformationen einholen |  | Mit Stefan Kuhn besprochen. |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| 9 Stunden | | 9 Stunden | |
|  | | | |
| **Projektfortschritt** | | | |
| I von IPERKA fertig  Analysieren der Aufgabenstellung, abgeschlossen  Zusatzinformationen einholen oder Ergänzungen zur Aufgabenstellung, abgeschlossen.  Zeitplan erstellen, angefangen. | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
| GitLab und Stefan Kuhn für die Erstellung vom Git Repository.  IPA Aufgabenstellung von PkOrg.  Screenshots von Masken der derzeitigen Zeiterfassung von Stefan Kuhn oder durch intern. Stiftung. | | | |
|  | | | |
| **Tagesreflexion** | | | |
| Ich habe heute gelernt, dass ich bei zwei Einladungen via Teams auf die erste Einladung am ersten Termintag im Teams abrufen sollte und nicht auf irgendeinen Link wie den zweiten Link für den zweiten Termintag.  Ich habe heute gelernt, wie man in GitLab zu einem neu erstellten Git Repository einen Maintainer hinzufügt.  Mir ist es heute gut bei der Informierung-Phase gegangen und war im Zeitplan. | | | |

Tabelle : Tagesprotokoll vom 12.5.2021

### Tagesprotokoll vom 14.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Die Projektaufbauorganisation in Form von einem Baumdiagramm erstellt. | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 08:30-  09:00 | Zeitplan überarbeitet | Ich konnte am meisten selbständig den Zeitplan überarbeiten | Hatte Hilfe durch Stefan Kuhn |
| 09:00-  12:00 | Use-Cases, ERM Diagramm erstellt und mit dem UML Klassen Diagramm angefangen | Hat alles funktioniert. | Keine. |
| 13:15-14:45 | UML Klassen Diagramm und Aktivitätsdiagramm erstellt, Kurzfassung schreiben. | Hat alles funktioniert. | Kurze Repetition mit Stefan Kuhn, was eine Kurzfassung ist. |
| 14:45-16:00 | Manuelles Testkonzept und PHPUnit Tests |  |  |
| 16:00-17:30 | Besprechung meiner IPA mit Ausbildner und nachträglich Notizen machen. | Ich weiss nun mehr was ich in meiner IPA zu beachten habe. | Feedback von Egil Rüefli, auf meine heutigen erarbeiteten Elemente |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| 8 Stunden | | 17 Stunden | |
|  | | | |
| **Projektfortschritt** | | | |
| P von IPERKA fertig | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
| Durch die Informierung vom Vortag und Hilfe von Stefan Kuhn und Egil Rüefli. | | | |
| nach feedback von Egil und Stefan  Kunde Monika Schmid im Baumdiagramm aufführen  Zeitplan noch in textfrom besser verfeinern  ERM überarbeiten, Aktivitätsdiagramm verfeinern  UML Klassendiagramm überarbeiten  ERM Datenbank Eigenschaften von Tabelle und in MongoDB Felder immer lower-camel-case wie die variablen in PHP  Glossar alphabetisch, alle Begriffe oder in anderen Worten Schlüsselwörter als Titel und darunter ein kurzer Text erfassen welcher wie zum Beispiel Google erklärt Abkürzung ausschreiben  Inhaltsverzeichnis quellen [1]... online quellen angeben  Testkonzept genauer beschreiben was bei den PHPUnit testen möchte und was bei den manuellen tests im frontend gemacht/getestet werden sollte  Arbeitsjournal ist wichtig erfolge und misserfolge genau zu benennen und das die zeiten im zeitplan übreinstimmen  Wenn ein link nicht mehr geht, waybackmachine benutzen oder darauf hinverweisen | | | |
| **Tagesreflexion** | | | |
| Ich habe heute viel von Stefan Kuhn über die Kurzfassung gelernt und habe ein Feedback von Egil Rüefli auf den Stand meiner Dokumentation und Planung bekommen.  Mir ist es heute gut bei der Informierung-Phase gegangen und war etwas ausserhalb vom Zeitplan. | | | |

Tabelle : Tagesprotokoll vom 9.5.2017

### Tagesprotokoll vom 17.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Vorentscheidungen treffen, ERM lower camel case statt snake case, Projektaufbauorganisation Kundin hinzufügen, mit Code realisierung von Masken und Architektur beginnen. Schlüsselwörter für das Glossar notieren. | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 08:10-12:00 | ERM und Projektaufbauorganisation angepasst, ein paar entscheindungen getroffen | Die Entscheidungen welche mir beim planen eingefallen sind, konnte ich gut abwägen. | Es bleiben Entscheidungen offen welche ert bei der Realsierung zum fallen kommen. |
| 13:00-17:30 | Composer und npm mit den für das Projekt nötigen Packages eingerichtet, PHP Klassen schreiben, MongoDB einrichten, LDAP einrichten, Views und Masken erstellen | Hat alles reibungslos funktioniert. | Keine. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| Stunden 9 Minuten 20 | | Stunden 26 Minuten 20 | |
|  | | | |
| **Projektfortschritt** | | | |
| E von IPERKA fertig und mit R angefangen | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
| Aus eigener Dokumentation. | | | |
|  | | | |
| **Tagesreflexion** | | | |
| Habe heute mir überlegt welche Entscheidungen für das Projekt wichtig sein können und diese notiert. | | | |

Tabelle : Tagesprotokoll vom 17.5.2021

### Tagesprotokoll vom 18.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Mit der Persistenz-Logik anfangen wie die Daten ans Frontend und Backend bei den Benutzern weitergegeben werden. | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 09:15-11:45 | snake case zu lower camel case von Twig Variablen und MongoDB Keys anpassen vom Benutzer. | Die Applikation funktionierte danach weiterhin gleich. | Nimmt etwas Zeit in Anspruch. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| Stunden 2 Minuten 30 | | Stunden 28 Minuten 50 | |
|  | | | |
| **Projektfortschritt** | | | |
| R von IPERKA fortgeführt | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
| Aus eigener Dokumentation. | | | |
|  | | | |
| **Tagesreflexion** | | | |
| Habe heute Morgen alle Keys wie Variablen in lower camel case umgeschrieben. | | | |

Tabelle : Tagesprotokoll vom 18.5.2021

### Tagesprotokoll vom 19.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Client Session konfigurierbar machen und auf 8 Stunden stellen. Die Persistenz-Logik im Aktivitätsdiagram für die Korrektur-Maske verfeinern. | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 09:00-11:00 | Session in Umgebungsvariablen aufgenommen. Benutzer ist 8 Stunden nach seiner letzten Anmeldung nicht mehr angemeldet. | Hat alles bestens funktioniert | Keine |
| 11:00-12.00 | Persistenz-Logik im Aktivitätsdiagram | Speicherung ohne Berechnung auseinandergehalten | Mit was beginnen und Aufbau vom Diagramm |
| 13:00-18:00 | Persistenz-Logik im Aktivitätsdiagram | Verarbeitung besser verfeinert | Verfeinern einzelner Input Verarbeitungen |
|  |  |  |  |
|  |  |  |  |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| Stunden 8 Minuten 0 | | Stunden 36 Minuten 50 | |
|  | | | |
| **Projektfortschritt** | | | |
| R von IPERKA fortgeführt  Session Handling abgeschlossen und im groben die Speicherung und Berechnung in der Korrektur-Maske im Aktivitätsdiagram festgehalten. | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
| Keine | | | |
|  | | | |
| **Tagesreflexion** | | | |
| Habe heute am Aktivitätsdiagram mehr Zustände für die Berechnung-Kaskadierung herausgefunden. | | | |

Tabelle : Tagesprotokoll vom 19.5.2021

### Tagesprotokoll vom 20.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Die Persistenz-Logik im Aktivitätsdiagram für die Korrektur-Maske fertig machen. Beginen mit implementieren von der Persistenz-Logik für die Korrektur-Maske oder Testdaten in der MongoDB erzeugen. | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 09:00-09:45 | Dokumentieren |  |  |
| 09:45-12:00 | Persistenz-Logik im Aktivitätsdiagram |  |  |
| 13:00-13:45 | Persistenz-Logik im Aktivitätsdiagram |  |  |
| 13:45-18:00 | Testdaten erfassen |  |  |
|  |  |  |  |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| Stunden 8 Minuten 0 | | Stunden 44 Minuten 50 | |
|  | | | |
| **Projektfortschritt** | | | |
| R von IPERKA fortgeführt | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
| Keine | | | |
|  | | | |
| **Tagesreflexion** | | | |
| Ich habe heute einen Weg gefunden schneller korrekte Testdaten zu generieren mit wenigen manuellen Anpassungen in der Datenbank. | | | |

Tabelle : Tagesprotokoll vom 20.5.2021

### Tagesprotokoll vom 21.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Monatliche Pflichtzeit von jedem Arbeitstag berechnen. Weitere Testdaten exportieren uind importieren. Die beiden Methoden isSunday und isSaturdy Logikfehler beheben. Da DateTime nicht grosser als 24 Stunden unterstzt das es wie ein tages Objekt ausgelegt, eine Methode welche zum Beispiel aus dem String “64:23” ein associative Array [“hour” => 64, “minute” => 23] zurückgibt. Da meherer Tage rauskommen werden diese anzahl Tage mit 24 Stunden multiplizieren, das Total in MongoDB statt DateTime speichern. Performance von der Übersicht testen, da diese am meisten Daten aus MongoDB bezieht. | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 08:05-12:00 |  |  |  |
| 13:00 - -17:45 |  |  |  |
| 17:45-18:00 | Performance von der Übersicht testen und optimieren. | Bei 1 Monat Eintrag von 5 Sekunden Ladezeit auf 100 Millisekunden Ladezeit. | Hilfe bei Performance Optimierung in PHP von Marco Selenati |
|  |  |  |  |
|  |  |  |  |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| Stunden 9 Minuten 0 | | Stunden 53 Minuten 50 | |
|  | | | |
| **Projektfortschritt** | | | |
| R von IPERKA fortgeführt | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
| Keine | | | |
|  | | | |
| **Tagesreflexion** | | | |
| Habe heute gelernt wie man performanteren Code schreiben kann und wie man die Stelle findet im Code wo ein Performance Problem vorliegt. | | | |

Tabelle : Tagesprotokoll vom 21.5.2021

### Tagesprotokoll vom 25.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Cron Job realisieren, dabei müssen die Zeiten von gestern mit den heutigen zusammengerechnet werden, wenn der heutige Tag nicht der erste Tag vom Monat, Woche oder im Gesamten ist. | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 08:05-12:00 | Cron Job realisieren | Unterscheidung von aktiven Benutzern und inaktiven Benutzern, welche die Zeiterfassung gestartet oder nicht gestartet haben. | Die Realisierung |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| Stunden 3 Minuten 55 | | Stunden 57 Minuten 45 | |
|  | | | |
| **Projektfortschritt** | | | |
| R von IPERKA fortgeführt | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
| Keine | | | |
|  | | | |
| **Tagesreflexion** | | | |
| Ich habe heute die Unterscheidung von Benutzern Zeiterfassende, nicht Zeiterfassende und allen Benutzer für den Cron Job implementiert. | | | |

Tabelle : Tagesprotokoll vom 25.5.2021

### Tagesprotokoll vom 26.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Beim Cron Job die Berechnung von gesamter und wöchentlicher Abweichung. Checkliste durchgehen. Durchführung der Abnahme. Diagramme, unter anderem Systeme verfeinern und mit den Unit Tests beginnen. | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 08:00-10:30 | Die Berechnung vom Cron Job weitergeführt. Nach DateTime yesterday und substr(string $string , int $offset , int|null $length = null): string Funktion recherchiert. | Hat alles mit Recherche zusammen funktioniert. | Musste noch berücksichtigen das sich zwei negative Zeiten zusammenaddieren und die Vorzeichen Operatoren von der Zeit getrennt und bei der Speicherung wieder zusammengefügt werden. |
| 10:30-11:30 | Checkliste durchgehen |  |  |
| 11:30-12:00 | UAT Tests vorbereiten |  |  |
| 13:00-17:00 | Cron Job realisieren und Abnahme | Das Berechnen vom heutigen Tag, der monatlichen Pflichtzeit, die monatlichen Präsenzzeit und Abweichung der monatlichen Zeit.  Habe ein Testkonzept 1 für die erste Abnahme geschrieben. | Die Realisierung, bei der Abnahme war ich nicht beim ersten Anlauf gut vorbereitet |
|  |  |  |  |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| Stunden 8 Minuten 0 | | Stunden 65 Minuten 45 | |
|  | | | |
| **Projektfortschritt** | | | |
| K von IPERKA fortgeführt | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
|  | | | |
|  | | | |
| **Tagesreflexion** | | | |
| Habe heute gelernt wie man ein System Diagramm erzeugt. | | | |

Tabelle : Tagesprotokoll vom 26.5.2021

### Tagesprotokoll vom 27.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Unit Tests durchführen, Tagesprotokoll nachführen. Dokumentieren. Besuchstag 2. Expertengespräch | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
| 08:00 – 12:00 | Korrekturen von der Abnahme |  |  |
| 13:00 – 13:30 | Weiterführung der Abnahme mit Testkonzept 2 |  |  |
| 13:30 – 14:30 | PHPUnit erstellen, durchführen und auswerten |  |  |
| 14:30 - | Dokumentieren |  |  |
|  |  |  |  |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| Stunden | | Stunden | |
|  | | | |
| **Projektfortschritt** | | | |
| K und A von IPERKA fortgeführt | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
|  | | | |
|  | | | |
| **Tagesreflexion** | | | |
|  | | | |

Tabelle : Tagesprotokoll vom 27.5.2021

### Tagesprotokoll vom 28.5.2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tagesziele** | | | |
| Bericht fertig dokumentieren. Glossar ergänzen. Bericht grammatikalische Korrektur. Alle Verzeichnisse aktualisieren. Abbildungsverzeichnis nachführen. Alle Tabellen und Diagrammen in Bericht einfügen. | | | |
|  | | | |
| **Zeit** | **Erledigte Arbeiten** | **Erfolge** | **Schwierigkeiten** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | |
| **Arbeitszeit Tagestotal** | | **Arbeitszeit Total** | |
| Stunden | | Stunden | |
|  | | | |
| **Projektfortschritt** | | | |
| K und A von IPERKA fortgeführt | | | |
|  | | | |
| **Informationsbeschaffung** | | | |
|  | | | |
|  | | | |
| **Tagesreflexion** | | | |
|  | | | |

Tabelle : Tagesprotokoll vom 28.5.2021

# Projekt-Dokumentation

## Kurzfassung

**Ausgangsituation**

Es gibt derzeit eine Zeiterfassung mit welcher die Betreuung ihre Zeit erfasst. Die Frau Monika Schmid welche für das Betreute Wohnen zuständig ist, hat jeden Tag alle Ihre mehreren Arbeitsstunden am Tag manuell zusammenzurechnen und leitet dessen Totalzeit an den Entwickler und Ausbildner Herr Valentin Marolf weiter, welcher das derzeitige Zeiterfassungssystem betreut, dieser muss dann diese Arbeitszeit von Frau Monika Schmid nachträglich im System erfassen. Da entsteht ein Problem, weil es viele manuelle Vorgänge der Benutzer braucht um Ihre Arbeitsstunden korrekt zu erfassen. Ein weiteres Problem ist das der ehemaligen Entwickler der derzeitigen Zeiterfassung schon lange pensioniert ist und sich die Applikation auf altem Stand haltet und darum schlecht zu erweitern ist.

Die Betreuung wünscht sich zudem noch das die Zeiterfassung auf den mobilen Geräten bedienbar ist und den Feiertagen und den nicht Feiertagen an Samstagen für die Berechnung von 1.5-fachen Zeit berücksichtigt werden.

**Umsetzung**

Die Umsetzung erfolgt auch in PHP, schliesslich war die bisherige Zeiterfassung auch in PHP und eine Webapplikation. Die Authentifizierung des Benutzers erfolgt auch mit dem LDAP der Rafisa, nur wird hierfür eine Login Maske und kein Popup erstellt. Die Speicherung erfolgt in der Datenbank MongoDB und nicht in einer SQL rationalen Datenbank. Da in MongoDB die Dokumentenstruktur von Daten viel Dynamischer gespeichert werden können, wie das es möglich sein sollte, dass beliebig viele Einträgen von Arbeitsstunden am selben Tag gespeichert werden sollten. Genauso ist es einfacher in MongoDB die Daten von Objekten unterschiedlicher Form abzuspeichern und diese dann nach dem Benutzer der neuen Applikation zu filtern.

Um die neue Zeiterfassung auch für mobile Geräten tauglich zu machen, wird diese responsive in HTML und CSS mit dem Framework Bootstrap in SCSS umgesetzt.

**Ergebnis**

Eine neue und erweiterbare Zeiterfassung, auf dem neusten Code-Stand von PHP, welche die gleichen Anforderungen der derzeitigen Zeiterfassung erfüllt und beliebig viele Arbeitseinsätze am Tag korrekt zusammenrechnet. Die Verrechnung von Arbeitsstunden an Feiertagen oder Sonntagen auf die Betreuung zugeschnitten und auf mobilen Geräten für das starten und beenden der Zeiterfassung bedienbar ist.

## Informieren

**Vorarbeiten**

Für die Feiertage hat Herr Valentin Marolf bereits die Geschäftsleitung dazu aufgefordert die Samstage auf Feiertage miteinzuschliessen. Anschliessen hat er mir einen API Aufruf ein JSON vom derzeitigen Zeiterfassung aufbereitet, welches in der Realisierung genauer in Betracht gezogen wird. Damit ist es möglich nur in einem von zwei Zeiterfassungen die Feiertage einzutragen und die neue Zeiterfassung weiss somit automatisch Bescheid, wenn ein Feiertag ist oder nicht.

**Analysieren der Aufgabenstellung**

Das einlesen der Detaillierte Aufgabenstellung meiner IPA.

**Titel der Arbeit**

Dieser Titel wurde als Titel vom IPA Bericht aufgenommen.

**Ausgangslage**

Seit 2021 bietet die Rafusa Informatik ein betreutes Wohnen an. Mehrere Arbeitseinsätze werden gestückelt über den Tag von der Betreuung abgeleistet. Die bisherige Zeiterfassung von der Rafisa deckt diesen Fall nicht ab und ist schwer erweiterbar. Aus diesem Grund will die Geschäftsleitung eine Applikation, mit welcher die Betreuung ihre Arbeitszeit erfassen und auswerten kann.

Detaillierte Aufgabenstellung

Klassen sollten wie in Java Programmierstil entsprechen, also Anfang groß und jedes Wort im selben wieder groß angefangen werden wie zum Beispiel für eine Rafisa Internal Organisation Mongo Collection Klasse heißt der Name wie folgt “RIOMongoCollection”.

Auf die Eigenschaften eines Objekts sollten via Methoden und nicht direkt zugergriffen werden.

Für das Login wird die Authentifizierung via ein LDAP von der Rafisa realisiert werden.

Nur Benutzer welche eingeloggt sind können die Zeiterfassung starten oder beenden, wenn diese bereits aktiv ist beenden und wenn inaktiv starten.

Es sind am Tag beliebig viele Arbeitseinträge für jeden Benutzer möglich.

Am Ende von jedem Tag müssen von jeweiligem Tag die Tages-, Wochen- und Monatssaldo berechnet werden.

Zur Dokumentation in der Planung sollte ein Use-Case Diagramm konzipiert werden. Für spätere möglich anfallenden Erweiterungen, sollte ein ERM und ein Aktivitätsdiagramm erstellt werden.

Bei der Eingabemaske vom Login sollte diese gleich dem Aufbau der bisherigen Zeiterfassung entsprechen. Zu den Elementen gehören, ein Textfeld für den Benutzername, ein Passwortfeld für das Passwort und ein Absenden Button für das Anmeldeformular.

Bei der Eingabemaske von der Zeiterfassung gibt es den nicht gestarteten und den gestarteten Zustand der Zeiterfassung des jeweiligen angemeldeten Benutzers. Dabei hat als Element derselbe Button im jeweiligen Zustand die Beschriftung “Zeiterfassung starten” oder “Zeiterfassung stoppen”.

Bei der Eingabemaske von der Übersicht sollte eine Tabelle vom selektierten Monat abgebildet werden. Die Arbeitseinträge sollten in aufeinanderfolgende Reihenfolge erscheinen. Tabellentitel sollte der jeweilige Monat und die Jahreszahl sein, die Weiterleitung durch die Navigation zur Übersicht ist immer der momentane Monat als Standardwert gesetzt. Links davon sollte zum letzten angrenzenden Monat und rechts zum nächsten Monat, wenn nicht in der Zukunftsform nicht ausgegraut und navigiert werden können. Die Arbeitseinträge sollten folgende Spalten mit Eigenschaften aufgezeigt werden: Datum des Eintrages, Startzeit des Eintrages, Endzeit des Eintrages, Soll- und Ist-Arbeitszeit des dazugehörigen Tages sowie die Differenz dieser zwei Werte und die Zeitsaldi für die jeweilige Woche, Monat und Total. Daneben noch jeweils ein Feld für allfällige Textkommentare. Bei jedem Arbeitseintrag sollte ein klickbares Symbol stehen, welches als Navigierung/Aufruf zur Korrektur vom jeweiligen Arbeitseintrag dienen sollte. Unter allen Arbeitseinträgen kommt eine Zeile hin, welche die Totalwerte dieses Monats abgebildet werden.

Bei der Eingabemaske von der Korrektur, sollten nochmals alle Eigenschaften eines Arbeitseintrages aufgezeigt werden. Neben jeder Eigenschaft (erste Spalte) sollte ein Korrekturfeld (zweite Spalte) vorhanden sein. Falls eine Korrektur oder mehrere Korrekturen anfallen wird der Kommentar zu einem Pflichtfeld. In der dritten Spalte sollte eine Erklärung zur jeweiligen Eigenschaft stehen, welche eins zu eins aus der bisherigen Zeiterfassung übernommen werden kann. Nach jeder Korrektur, sollte die letzte Korrektur mit welchem Benutzer und Datum sowie Uhrzeit angezeigt werden.

Täglichen Verarbeitung

Wird am Ende von jedem Tag der Saldo ausgerechnet. Die Arbeitsstunden an Sonn- und Feiertagen werden 1.5-fach berechnet.

Fehlerbehandlung

Der Kontakt zum LDAP nicht möglich, der Benutzer wird mit einer Fehlermeldung informieren.

Anfragen mit Fehlerhaften Date, wird der Benutzer mit einer Fehlermeldung informieren, welche ihn auf die Inkompatibilität der Daten hinweist.

PHP-Fehlermeldungen werden nicht angezeigt, der Programm-Fluss wird gestoppt und der Benutzer wird zur Login-Maske weitergeleitet.

Tests

Testkonzept erstellen

Tests gemäss Konzept durchführen

Folgende Testszenarien müssen mindestens berücksichtigt werden.

Korrekte Funktion der Masken

Fehler abfangen

Fehlermeldungen nach Vorgaben anzeigen

**Zusatzinformationen einholen**

Die Absenzen nur die Differenz von Präsenzzeit und Pflichtzeit beeinflusst und nicht die Pflichtzeit selbst.

Und das die korrigierte Gesamtsaldo auf die Künftige Einträge beeinflusst solange in der Zukunft keine weitere Gesamtsaldo Korrektur anfallen, dazwischen bilden sich dann Lücken welche dann von der letzten Gesamtsaldo Korrektur überschrieben werden. Vom Tag mit Gesamtsaldo Korrektur hat alles Rückwirkende keinen Einfluss auf das Gesamtsaldo in der Zukunft.

Die letzte Gesamtsaldo Korrektur Beeinflusst das Gesamtsaldo in der Zukunft.

Gesamtsaldo Korrektur ignoriert aller Arbeitsbeginn und Arbeitsende vom selben Tag und berücksichtigen erst den vom Folgetag an alles weitere.

**Ergänzungen zur Aufgabenstellung**

Abmeldung / Angemeldet

* Nur auf einem Gerät auf einen Benutzer gleichzeitig angemeldet sein.
* Die Option Abmeldung wird direkt nach der Anmeldung in der Zeiterfassungsmaske angezeigt.

Masken-Navigation

* Eine Masken-Navigation, welche zur Pflichtzeit, Übersicht und zur Zeiterfassung führt.

Übersicht-Navigation

* Statt Monat in der Zukunft ausgegraut nichts anzeigen.

Mobiltauglichkeit

* Monika Schmid von der Betreuung möchte gerne, dass die Webapplikation auf den mobilen Geräten für das starten und beenden der Zeiterfassung, tauglich ist.
* Design Responsive mit /Bootstrap CSS Framework/ für Mobile Bedienung der Betreuung, nur für Login- und Zeiterfassungsmaske.

Übersicht Monatstotal

* Monatstotal in der Übersichtsmaske im Detail.
* Wochentage sind in der Spalte Datum auch ersichtlich.
* Unten ein Bild mit allen Attributen für die Übersicht, welche 1:1 für die neue Applikation übernommen werden kann.

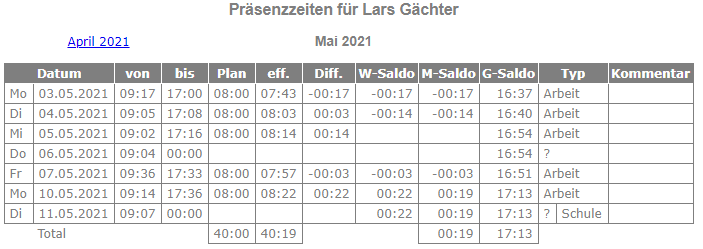
[](https://intern.stiftung-ifa.ch/betrieb/mitarbeiter/praesenzzeit_anzeige.php?popup=j&datum=11.05.2021)

Abbildung : Übersicht

Quelle: Screenshot von Lars Gächter Link <https://intern.stiftung-ifa.ch/betrieb/mitarbeiter/praesenzzeit_anzeige.php?popup=j&datum=11.05.2021>

Benutzer Pflichtzeit

* Weitere Maske mit der Anpassung der Pflichtzeit, welche ab dem nächsten Folgetag gesetzt sein wird.

Korrekturmaske im Detail

* Datum ist nicht veränderbar.
* Arbeitsbeginn ist veränderbar, darf nicht grösser als Arbeitsende sein.
* Arbeitsende ist veränderbar, darf nicht kleiner als Arbeitsbeginn sein.
* Arbeitsbeginn und Arbeitsende, können gleichgross sein, dürfen nicht andere Arbeitsbeginn und Arbeitsende am selben Tag überschneiden, kann nicht negativ ausfallen.
* Pflichtzeit ist veränderbar, kann nicht negativ ausfallen.
* Zeitgutschrift ist veränderbar und standartmässig 0 Stunden und 0 Minuten, kann nicht negativ ausfallen.
* Präsenzzeit ist veränderbar und überschreibt / ignoriert die Ausrechnung aus Arbeitsbeginn und Arbeitsende, kann nicht negativ ausfallen.
* Geleistete Arbeitszeit wird vorläufig genau wie die Präsenzzeit behandelt ohne Pausen-Abzug.
* Abweichung ist nicht veränderbar, kann negativ ausfallen.
* Abwesend?, kann die Pflichtzeit auf 0 setzen in der Differenz Rechnung der eigentlichen Präsenzzeit
* Vormittags, kann die Pflichtzeit halbieren in der Differenz Rechnung der eigentlichen Präsenzzeit
* Nachmittags, kann die Pflichtzeit halbieren in der Differenz Rechnung der eigentlichen Präsenzzeit
* Abwesend?, Vormittags und Nachmittags haben folgende Optionen
  + (nicht extern), ist immer standardmässig gesetzt
  + Ergänzungskurs, kann die Pflichtzeit halbieren oder auf 0 setzen
  + Telearbeit
  + Zügeln, kann die Pflichtzeit halbieren oder auf 0 setzen
  + Abwesend
  + Kompensation
  + offizieller Feiertag, kann die Pflichtzeit halbieren oder auf 0 setzen
  + Ferien, kann die Pflichtzeit halbieren oder auf 0 setzen
  + Betriebsferien, kann die Pflichtzeit halbieren oder auf 0 setzen
  + Bonus
  + Krank, kann die Pflichtzeit halbieren oder auf 0 setzen
  + Arzt, kann die Pflichtzeit halbieren oder auf 0 setzen
  + Unfall, kann die Pflichtzeit halbieren oder auf 0 setzen
  + Militär, kann die Pflichtzeit halbieren oder auf 0 setzen
  + Besprechung
  + †, kann die Pflichtzeit halbieren oder auf 0 setzen
* Wochensaldo ist nicht veränderbar, kann negativ ausfallen.
* Monatssaldo ist nicht veränderbar, kann negativ ausfallen.
* Gesamtsaldo ist veränderbar, kann negativ ausfallen.
* Kommentar ist veränderbar und ist bei jeglicher Korrektur oder Veränderung ein Plicht-Feld.

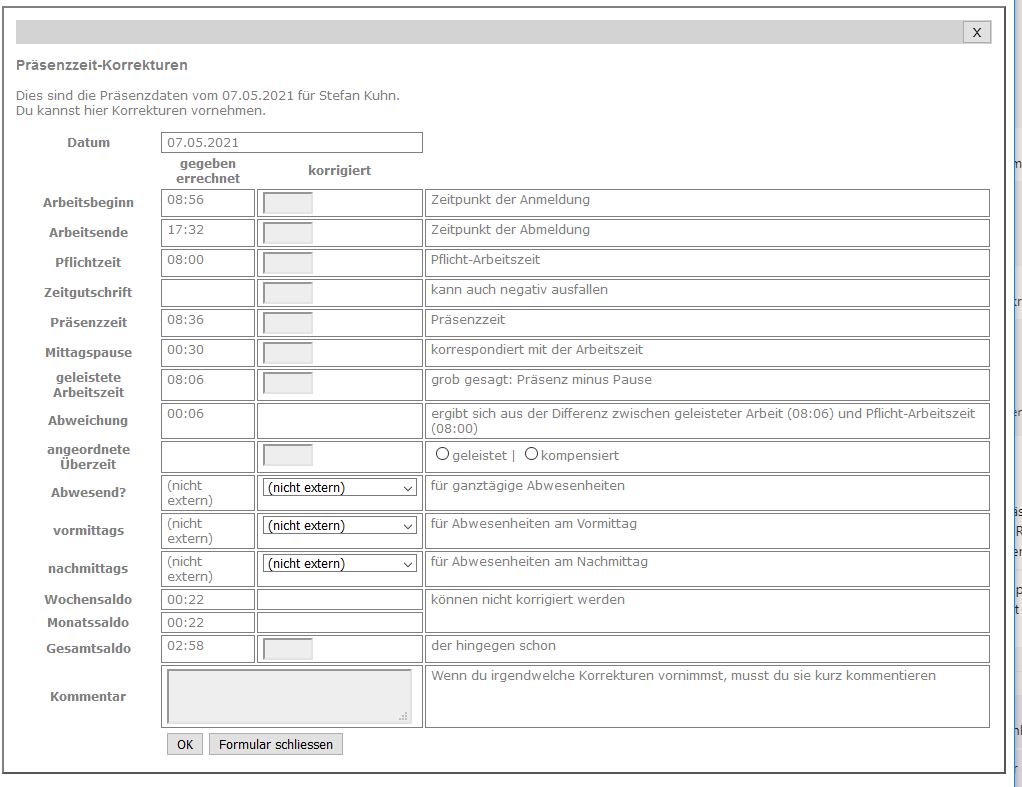


Abbildung : Korrektur

Quelle: Screenshot von Stefan Kuhn

* Nach jeder Salo beeinflussbaren Korrektur muss von diesem Datum an, jedes und jedes danach folgende Datum bis mit gestern neu berechnet und aktualisiert werden.
* Die Eingabefelder Mittagspause und angeordnete Überzeit wie im Sceenshot sind ausgeschlossen.

## Planen

Tätigkeiten festlegen für den Zeitplan

Projektmanagement-Methode: IPERKA

System Diagramm

Graphical user interface, application

Description automatically generated with medium confidence

Abbildung 5: System Diagramm

Use Cases

* Anmeldung/Abmeldung
* Zeiterfassung starten und beenden
* Pflichtzeit anpassen
* Tageskorrekturen

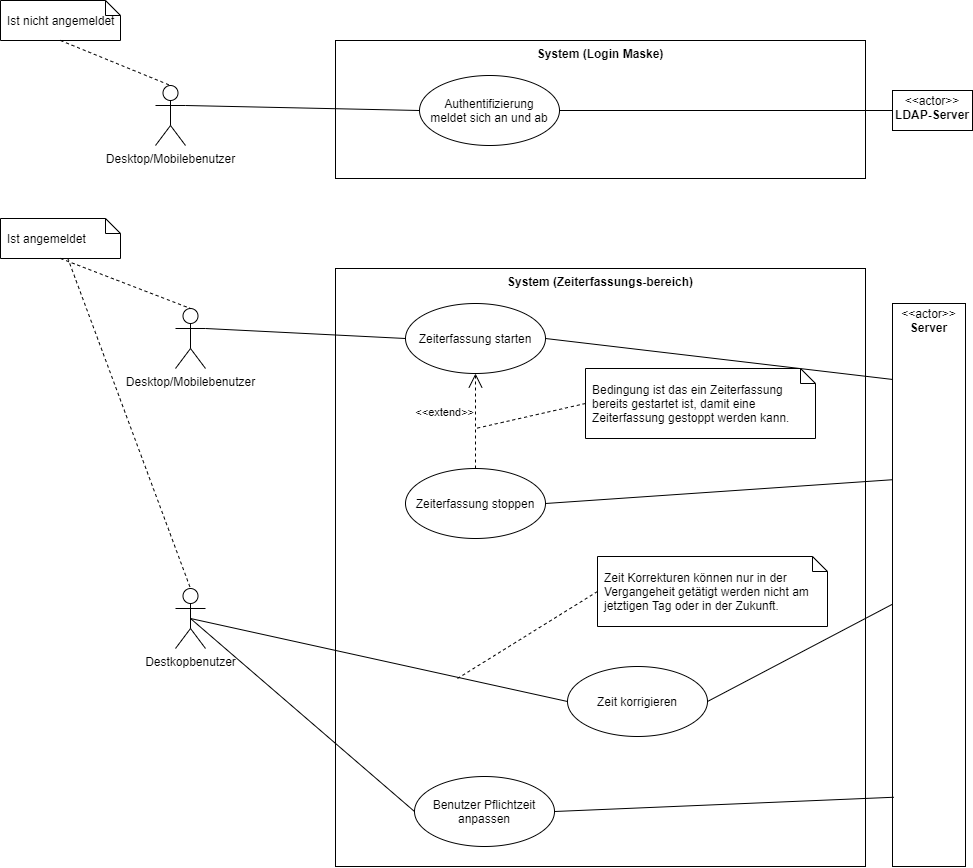


Abbildung : Use Cases

Aktivitäten Diagramm

Login

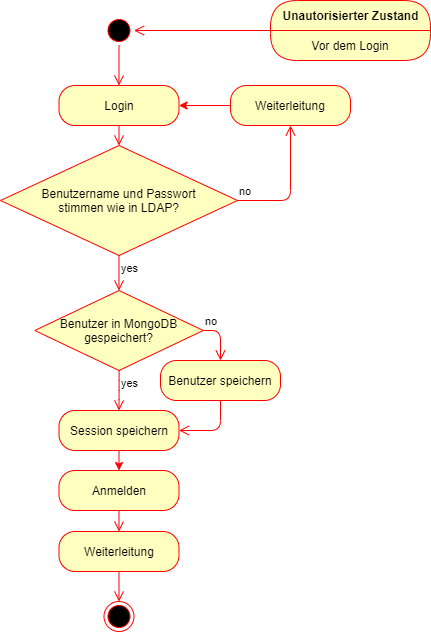


Abbildung : Aktivitäten Diagramm (Login)

Generelle Korrektur in der Zeiterfassung

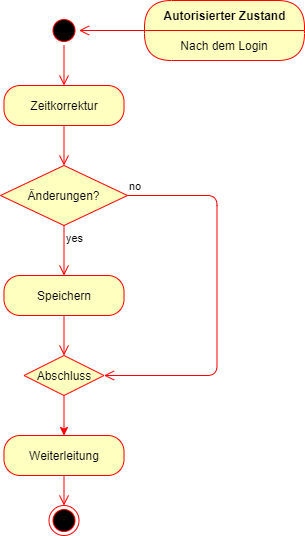


Abbildung 8: Generelle Korrektur

Detaillierte Korrekturen in der Zeiterfassung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Detaillierte Korrekturen

ERM

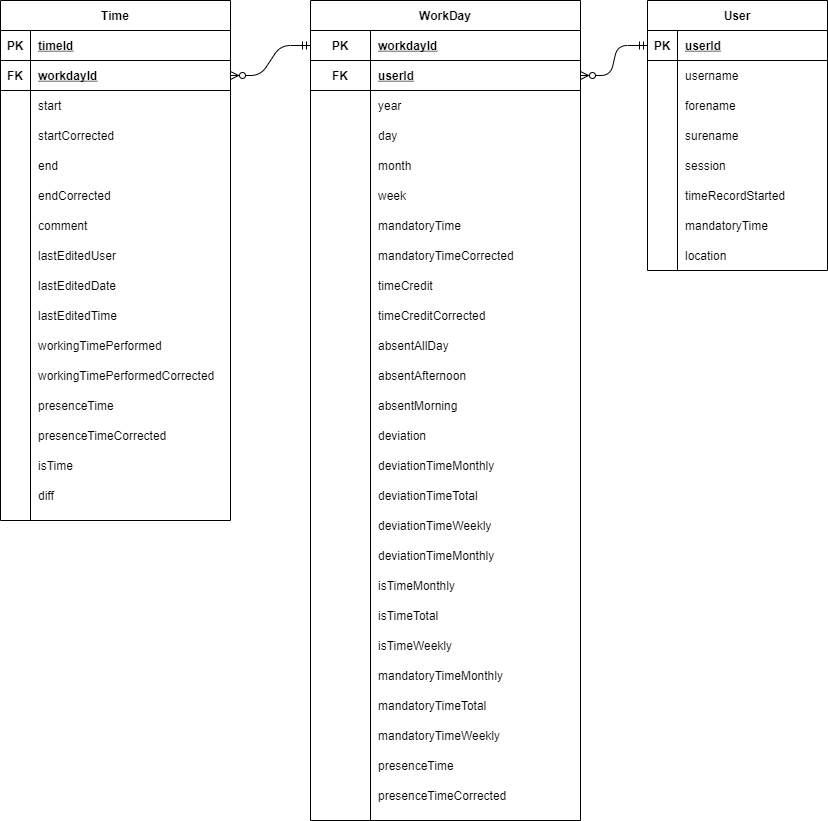


Abbildung 10: ERM

UML Klassen

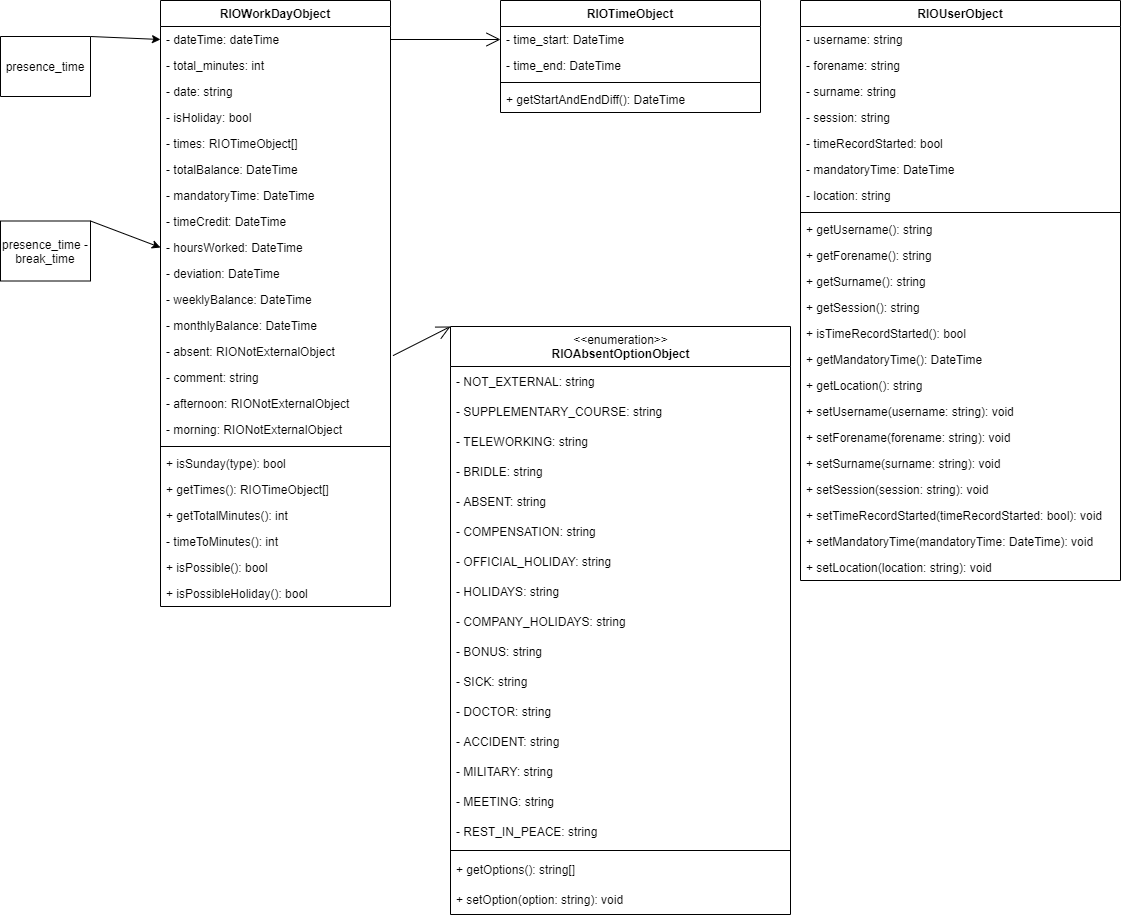


Abbildung 11: UML (manuell)

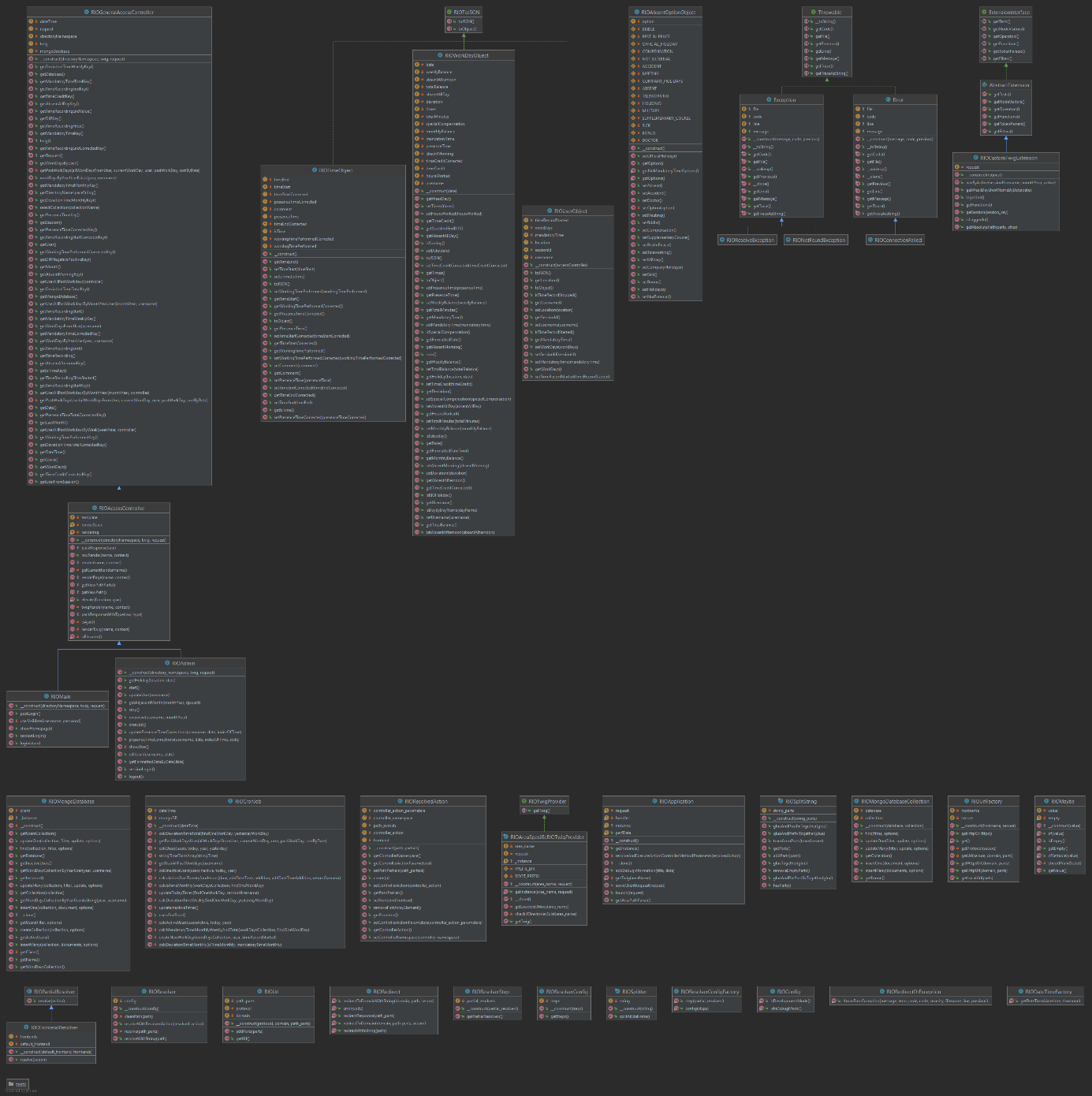


Abbildung 12: UML (PhpStorm)

### Testkonzept

Ich werde auf einem Windows Rechner auf welchem PHPStorm mit PHP läuft die Backend Tests durchführen, dafür benutze ich PHPUnit.

Auf einem Rechner auf dem das OS (Betriebssystem) Windows installiert ist, werden in der IDE (Integrierte Entwicklungsumgebung) PHPStorm die Backend Tests mit PHPUnit im Git Projekt durchgeführt.

Die Manuellen Tests werden von mir Lars Gächter als Entwickler und der Kundin Monika Schmid bei der Freigabe am 26.05.2021 durchgeführt. Diese Manuellen Tests werden in einer virtuellen Maschine namens VirtualBox durchgeführt. Dabei laufen 2 Server und 1 Client im selben Netzwerk, welche miteinander kommunizieren können. Auf dem einen Server läuft nur die Datenbank namens LDAP und auf dem anderen Server läuft die Datenbank namens MongoDB, die Webservern PHP und Apache und für die Projekt-Aktualisierung das Git. Auf dem Client wird das Frontend via dem Google Chrome Browser aufgerufen, um die Mobile Anwendung im Test auch miteinzuschliessen, wird im Google Chrome der Entwickler Modus aktiviert und die Mobile Ansicht abgerufen.

Testdaten

Ein Mitarbeiter mit Arbeitseinträge vom ganzen April bis jetzt im Mai.

## Entscheiden

Entscheidungsmatrix

Zeitklasse

Zeitzone von DateTime festlegen

Curl PHP oder Symfony

MngoDB VS SQL Datenbank

Kontroller Bereiche für nicht- und angemeldeter Benutzer

## Realisieren

\*.sh Datei ausführbar machen. <https://askubuntu.com/questions/122428/how-to-run-sh-file>

Ordner rekursiv entfernen. <https://askubuntu.com/questions/217893/how-to-delete-a-non-empty-directory-in-terminal>

Entfernen von Dateien ausser einer Datei. <https://www.tecmint.com/delete-all-files-in-directory-except-one-few-file-extensions/>

**Admin-Handbuch**

Die Installationsanleitung ist im Anhang mit dem Git Repository.

Das Inhaltsverzeichnis befindet sich im Ordner «arbeitszeitserfassung\_wohnen» und heisst README.md. Dabei habe ich die Entwicklung und Produktion der Applikation in zwei weitere verlinkten \*.md Dateien unterteilt.

Intallation vom Projekt

Klonen vom Git-Repository

git clone https://gitlab.rafisa.org/l.gaechter/arbeitszeitserfassung\_wohnen.git

Einrichten der Umgebungsvariablen

cp .env.example .env

Installieren von Composer

chmod +x install\_composer.sh

./install\_composer.sh

php composer.phar i

Installieren der npm Packages, generiert einen node\_modules Ordner im arbeitszeitserfassung\_wohnen Ordner.

npm init

npm i bootstrap

npm i sass

npm i

SCSS zu komprimierten CSS parsen und JS zu komprimierten JS

Für Produktion (wandelt einmal um)

npm run css:prod

npm run js:prod

Für Entwicklung (überwacht vorlaufend Veränderungen und wandelt um)

npm run css:dev

npm run js:dev

Apache und Berechtigung

sudo chown -R www-data arbeitszeitserfassung\_wohnen

**Performance Optimierung**

Kurzbeschreibung

Nach dem alle Arbeitstage inklusive des heutigen Tags aus der Datenbank gezogen wurden, musste der der heutige Tag aus der liste entfernt werden um nicht in der Übersicht zu erscheinen. Dabei wir über eine nach Datum sortierte Liste von Arbeitstagen iteriert.

Performance Problem

Bei der Iteration wurden Mal 2 Instanzen gebildet und viele diverse Methoden aufgerufen. Die Zeit beträgt bei über 30 Tagen Einträge rund 5 Sekunden und ist inakzeptabel.

Lösung

Vor der Iteration nötige Instanzen gebildet und in der Iteration ein Array Objekt mit Datum Key mit dem Datum welches vor der Iteration gesetzt wird als String vergleichen. Dies hat viele Instanzen und Methoden Aufrufe eliminiert und Zeit gespart. Die Zeit beträgt bei über 30 Tagen Einträge rund 100 Millisekunden und ist in einem akzeptablen bereich.

**Recherchen**

Datum vom nächsten Tag [1]

Datum von gestern [2]

Zum Beispiel +07:35 zu + und 07:35 aufteilen [3]

substr(‘+07:35’,0,0);

substr(‘+07:35’,1);

Architektur entwerfen

MVC

Datenbanken

MongoDB

* Datenbank
* Sammlungen (Collection)
* Dokumentenaufbau

Geo-Location/Zeitserver

RIODateTimeFactory

Logging

Persistenzlogik

Validationsklassen

Buisnesslogik

Fehlerbehandlung

## Kontrollieren

Präsentations-Masken

Login-Maske

Desktop

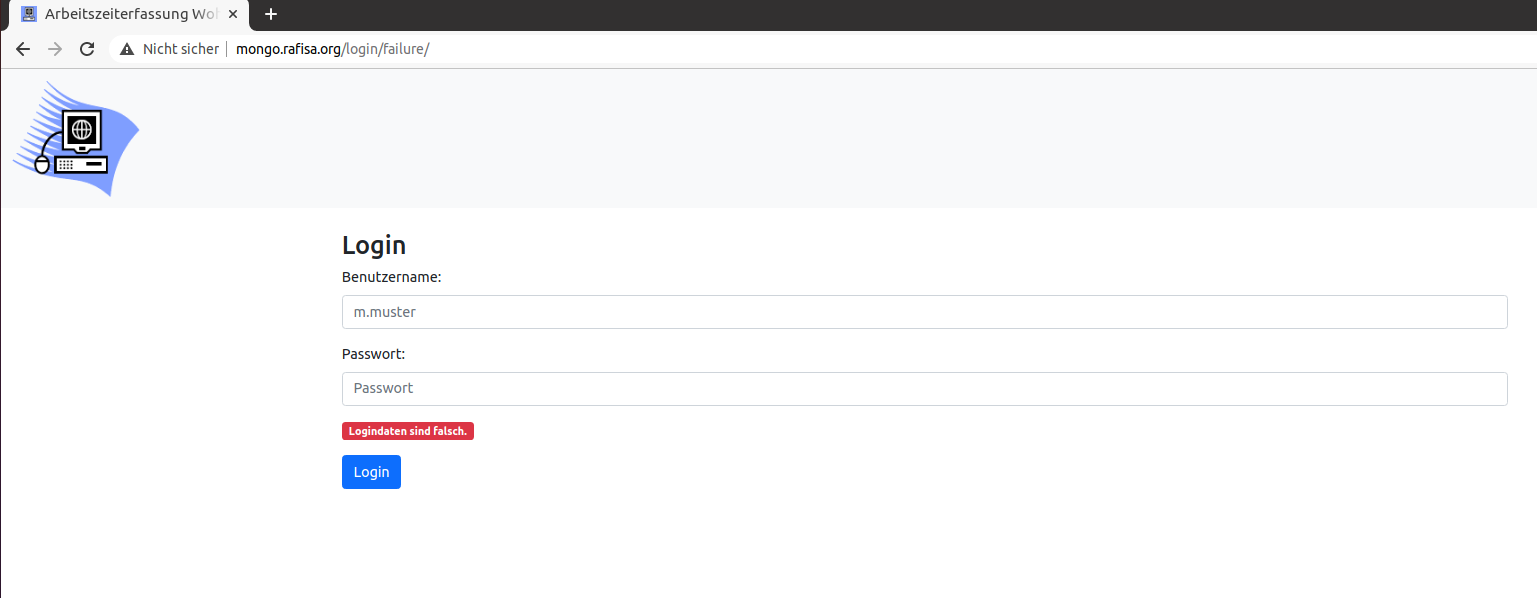


Abbildung : Login-Maske (Desktop)

Mobile

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Login-Maske (Mobile)

Zeiterfassungsmaske (Logout)

Desktop

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Zeiterfassungsmaske (Desktop)

Mobile

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Zeiterfassungsmaske (Mobile)

Benutzer Pflichtzeit

Desktop

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Benutzer Pflichtzeit (Desktop)

Mobile

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Benutzer Pflichtzeit (Mobile)

Übersichtsmaske

Desktop

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Abbildung : Übersichtsmaske (oben)

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Abbildung : Übersichtsmaske (unten)

Benutzer Tageskorrektur

Desktop

Ein Bild, das Text, Screenshot, computer enthält.

Automatisch generierte Beschreibung

Abbildung : Tageskorrektur

### Testprotokoll

Testkonzept 1

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Abbildung : Testkonzept 1 (Seite 1)

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Abbildung : Testkonzept 1 (Seite 2)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Testkonzept 1 (Seite 3)

Testkonzept 2

## Auswerten

### Reflexion

PHPUnit Tests für die Auswertung

|  |  |  |
| --- | --- | --- |
| ID / Bezeichnung | 1 | Ist Samstag Test |
| Beschreibung | Mit einem künftigen oder vergangenen Datum, welches ein Samstag ist wir getestet. | |
| Testvoraussetzung | Ein DateTime Objekt ist instanziiert und die isSaturday Methode wir aufgerufen. | |
| Testschritte  (Schritt / erwartetes Resultat) | Laden der Umgebungsvariablen, ein WorkDay Object mit einem Samstag Datum erstellen und das WorkDay Objekt fragen ob ein Samstag gesetzt ist. | |
| Erwartetes Ergebnis | OK | |
| Erhaltenes Ergebnis | OK | |

|  |  |
| --- | --- |
| Tester | Lars Gächter |
| Datum Testdurchführung | 27.05.2021 |
| Fehlerbeschreibung | Keine |

Code:

*/\*\*  
 \** ***@throws*** *\Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface  
 \*/*public function testIsSaturday(): void  
{  
 (new Dotenv())->bootEnv(dirname(*\_\_DIR\_\_*).'\..\..\.env');  
 $dateTime = RIODateTimeFactory::*getDateTime*("2021-05-22");  
 $workDay = new RIOWorkDayObject($dateTime);  
 $result = $workDay->isSaturday();  
 $this->assertEquals(true,$result,"Test has failed!");  
}

|  |  |  |
| --- | --- | --- |
| ID / Bezeichnung | 2 | Ist Sonntag Test |
| Beschreibung | Mit einem künftigen oder vergangenen Datum, welches ein Sonntag ist wir getestet. | |
| Testvoraussetzung | Ein DateTime Objekt ist instanziiert und die isSunday Methode wir aufgerufen. | |
| Testschritte  (Schritt / erwartetes Resultat) | Laden der Umgebungsvariablen, ein WorkDay Object mit einem Sonntag Datum erstellen und das WorkDay Objekt fragen ob ein Sonntag gesetzt ist. | |
| Erwartetes Ergebnis | OK | |
| Erhaltenes Ergebnis | OK | |
| Tester | Lars Gächter | |
| Datum Testdurchführung | 27.05.2021 | |
| Fehlerbeschreibung | Keine | |

Code:

*/\*\*  
 \** ***@throws*** *\Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface  
 \*/*public function testIsSunday(): void  
{  
 (new Dotenv())->bootEnv(dirname(*\_\_DIR\_\_*).'\..\..\.env');  
 $dateTime = RIODateTimeFactory::*getDateTime*("2021-05-23");  
 $workDay = new RIOWorkDayObject($dateTime);  
 $result = $workDay->isSunday();  
 $this->assertEquals(true,$result,"Test has failed!");  
}

|  |  |  |
| --- | --- | --- |
| ID / Bezeichnung | 3 | Ist Feiertag Test |
| Beschreibung | Mit einem künftigen oder vergangenen Datum, welches ein Feiertag ist wir getestet. | |
| Testvoraussetzung | Ein DateTime Objekt ist instanziiert und die isRIOHoliday Methode wir aufgerufen. | |
| Testschritte  (Schritt / erwartetes Resultat) | Laden der Umgebungsvariablen, ein WorkDay Object mit einem Feiertag Datum erstellen und das WorkDay Objekt fragen ob ein Feiertag gesetzt ist. | |
| Erwartetes Ergebnis | OK | |
| Erhaltenes Ergebnis | OK | |

|  |  |
| --- | --- |
| Tester | Lars Gächter |
| Datum Testdurchführung | 27.05.2021 |
| Fehlerbeschreibung | Keine |

Code:

*/\*\*  
 \* Test  
 \*  
 \** ***@throws*** *\PHPUnit\Framework\ExpectationFailedException|\Exception  
 \** ***@throws*** *\Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface  
 \*/*public function testIsHoliday(): void  
{  
 (new Dotenv())->bootEnv(dirname(*\_\_DIR\_\_*).'\..\..\.env');  
 $dateTime = RIODateTimeFactory::*getDateTime*("2021-05-24");  
 $workDay = new RIOWorkDayObject($dateTime);  
 $result = $workDay->isRIOHoliday();  
 $this->assertEquals(true,$result,"Test has failed!");  
}

|  |  |  |
| --- | --- | --- |
| ID / Bezeichnung | 4 | Ist heute Pflichtzeit Test |
| Beschreibung | Mit dem heutigen Datum wird getestet.  Ist der Test rot ist keine Pflichtzeit, ist der Test grün gilt eine Pflichtzeit. | |
| Testvoraussetzung | Ein DateTime Objekt ist instanziiert und die isRIOHoliday oder isSunday oder isSaturday Methoden werden aufgerufen. | |
| Testschritte  (Schritt / erwartetes Resultat) | Laden der Umgebungsvariablen, ein WorkDay Object mit dem heutigen Datum (siehe Datum Testdurchführung) erstellen und das WorkDay Objekt fragen ob ein Feiertag, ein Sonntag oder ein Samstag ist gesetzt ist. | |
| Erwartetes Ergebnis | OK | |
| Erhaltenes Ergebnis | OK | |

|  |  |
| --- | --- |
| Tester | Lars Gächter |
| Datum Testdurchführung | 27.05.2021 |
| Fehlerbeschreibung | Keine |

Code:

*/\*\*  
 \** ***@throws*** *\Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface  
 \*/*public function testMandatoryTimeToday(): void  
{  
 (new Dotenv())->bootEnv(dirname(*\_\_DIR\_\_*).'\..\..\.env');  
 $workDay = new RIOWorkDayObject();  
 $result = $workDay->isRIOHoliday() ||  
 $workDay->isSunday() ||  
 $workDay->isSaturday();  
 $this->assertEquals(false,$result,"Test has failed!");  
}

Meldung der Resultate:

Testing started at 14:43 ...

PHPUnit 9.5.4 by Sebastian Bergmann and contributors.

Time: 00:00.476, Memory: 6.00 MB

OK (4 tests, 4 assertions)

Process finished with exit code 0

### Schlusswort

## Glossar

Composer

CSS

Curl

HTML

IPERKA-Methode

* Besteht aus 6 abgekürzten Buchstaben welchen folgenden Schritten beschreiben. Das Informieren, das Planen, das Entscheiden, das Realisieren, das Kontrollieren und das Auswerten.

JSON

LDAP

npm

MongoDB

MVC

Responsive Design

Sass

* Ausgeschrieben Syntactically Awesome Style Sheets und ist eine Stylesheet-Sprache, welche CSS mit Kontrollstrukturen vereint. Unterstützt die Syntax von SCSS, welcher zu CSS kompiliert wird.

Symfony

Twig

Ubuntu

UTF-8

## Quellenverzeichnis

Stackoverflow kommentare günder Haken

|  |  |
| --- | --- |
| [1] | J. Conde, «stackoverflow,» 7 1 2014. [Online]. Available: https://stackoverflow.com/questions/14460518/php-get-tomorrows-date-from-date?answertab=active#tab-top. [Zugriff am 25 5 2021]. |
| [2] | «php.net,» [Online]. Available: https://www.php.net/manual/en/datetime.formats.relative.php. [Zugriff am 26 05 2021]. |
| [3] | «php.net,» [Online]. Available: https://www.php.net/manual/en/function.substr.php. |
| [4] | «Composer,» 17 05 2021. [Online]. Available: https://getcomposer.org/download/. |

## Literaturverzeichnis

Php.net nicht Kommentare sondern Fakten

# Anhang

**.env.example**

# Application

HOSTNAME=mongo.rafisa.org

API\_HOSTNAME=stiftung-ifa.ch

LDAP\_HOST="10.0.0.10"

LDAP\_SEARCH\_ROOT="ou=People,dc=rafisa,dc=org"

LDAP\_PORT=389

LDAP\_RDN="uid"

HTTPS=false

API\_HTTPS=true

APP\_NAME="Arbeitszeiterfassung Wohnen"

DEFAULT\_AREA\_NAME="riomain"

EMPTY\_URL\_CONTROLLER="showHomepage"

# 8 hour

SESSION\_LIFE\_TIME=28800

# development mode

DEVELOPMENT\_MODE=true

DEBUG=true

MAINTENANCE=false

LAUNCH\_YEAR=2021

# MongoDB

MONGODB="mongodb://localhost:27017"

DB\_NAME="time"

**ldap\_import/add\_content.ldif**

dn: ou=People,dc=rafisa,dc=org

objectClass: organizationalUnit

ou: People

dn: ou=Groups,dc=rafisa,dc=org

objectClass: organizationalUnit

ou: Groups

dn: cn=Administration,ou=Groups,dc=rafisa,dc=org

objectClass: posixGroup

cn: Administration

gidNumber: 5000

dn: uid=l.gaechter,ou=People,dc=rafisa,dc=org

objectClass: inetOrgPerson

objectClass: posixAccount

objectClass: shadowAccount

uid: Lars

sn: Gaechter

givenName: Lars

cn: Lars Gaechter

displayName: Lars Gaechter

uidNumber: 10000

gidNumber: 5000

userPassword: {CRYPT}x

gecos: Lars Gaechter

loginShell: /bin/bash

homeDirectory: /home/l.gaechter

dn: uid=m.schmid,ou=People,dc=rafisa,dc=org

objectClass: inetOrgPerson

objectClass: posixAccount

objectClass: shadowAccount

uid: Monika

sn: Schmid

givenName: Monika

cn: Monika Schmid

displayName: Monika Schmid

uidNumber: 11000

gidNumber: 5000

userPassword: {CRYPT}x

gecos: Monika Schmid

loginShell: /bin/bash

homeDirectory: /home/m.schmid

**public/index.php**

<?php

declare(strict\_types=1);

use Symfony\Component\Dotenv\Dotenv;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Component\HttpFoundation\Session\Session;

use Symfony\Component\HttpFoundation\Session\Storage\Handler\NativeFileSessionHandler;

use Symfony\Component\HttpFoundation\Session\Storage\NativeSessionStorage;

use Symfony\Component\Stopwatch\Stopwatch;

include\_once \_\_DIR\_\_.'/../source/autoload.php';

(new Dotenv())->bootEnv(dirname(\_\_DIR\_\_).'/.env');

error\_reporting(E\_ALL);

mb\_internal\_encoding('UTF-8');

mb\_http\_output('UTF-8');

mb\_regex\_encoding('UTF-8');

if("false" === $\_ENV["DEVELOPMENT\_MODE"]) {

error\_reporting(0);

ini\_set('display\_errors', "0");

}

if("true" === $\_ENV["DEVELOPMENT\_MODE"]) {

error\_reporting(E\_ALL);

ini\_set('display\_errors', "1");

}

$stopwatch = new Stopwatch(true);

$stopwatch->start('server\_time');

$api = RIOApplication::getInstance();

$request = Request::createFromGlobals();

if("true" === $\_ENV["MAINTENANCE"]) {

$maintenanceResponse = new Response();

$maintenanceResponse->setStatusCode(Response::HTTP\_SERVICE\_UNAVAILABLE)->setContent("Site under maintenance!")->send();

die();

}

$stopwatchSession = new Stopwatch(true);

$stopwatchSession->start('session');

if (!$request->hasSession()) {

$session = new Session(

new NativeSessionStorage(

[],

new NativeFileSessionHandler()

)

);

$request->setSession($session);

}

$request->getSession()->start();

if($\_ENV["SESSION\_LIFE\_TIME"] === $request->getSession()->getMetadataBag()->getLifetime()) {

if($\_ENV["SESSION\_LIFE\_TIME"] >= $request->getSession()->getMetadataBag()->getLastUsed() - $request->getSession()->getMetadataBag()->getCreated()) {

// Session has still lifetime

} else {

// Session has no lifetime

$request->getSession()->invalidate();

}

}

/\*\*

\* Force close the session

\* You cant use the session to store things unless you open it again

\* We start the session to get the id from it.

\*/

$request->getSession()->save();

$event = $stopwatchSession->stop('session');

RIOApplication::$perfData['session'] = [

'time' => $event->getDuration(),

];

try {

$response = RIOApplication::getInstance()->launch($request);

} catch (RIOConnectionFailed $connectionFailed) {

if (RIOConfig::isInDebugMode()) {

throw new Error("The database connection could not be established.", 0, $connectionFailed);

} else {

$response = RIORedirect::error(503);

}

}

$event = $stopwatch->stop('server\_time');

if (RIOConfig::isInDebugMode()) {

$response->headers->add([

'perf-server-duration' => $event->getDuration().' ms',

'perf-max-memory' => $event->getMemory().' bytes',

]);

}

$response->send();

**public/error.php**

<?php

$codes = [

'403/' => ['403 Forbidden', 'Kein zugriffsrecht'],

'404/' => ['404 Not Found', 'Die seite wurde nicht gefunden'],

'500/' => ['500 Internal Server Error', 'Die verarbeitung der seiten anfrage konnte nicht erfüllt werden'],

'503/' => ['503 Service Unavailable', 'Ein dienst welchen diese seite brauch ist nicht verfügbar'],

];

if (isset($\_GET['status'])) {

$status = $\_GET['status'];

if (isset($codes[$status])) {

$title = $codes[$status][0];

$message = $codes[$status][1];

} else {

$title = '?';

$message = 'Keine nachricht vorhanden für dieses Problem.';

}

} else {

$title = 'Kein error';

$message = 'Diese Seite wurde ohne ein vorheriges problem aufgerufen.';

}

echo

"

<!DOCTYPE html>

<html lang='de'>

<head>

<meta name='viewport' content='width=device-width, initial-scale=1.0'>

<meta charset='UTF-8'>

<meta name='description' content='Error'>

<link rel=\"stylesheet\" href=\"/css/main.css\">

<link rel=\"shortcut icon\" type=\"image/x-icon\" href=\"favicon.ico\" sizes='16x16'>

<title>$title</title>

</head>

<body>

<div style='max-width: 500px; margin: auto; background: white;'>

<h1 style='font-size: 36px;'>$title</h1>

<p style='font-size: 36px;'>$message</p>

</div>

</body>

</html>

";

**source/autoload.php**

<?php

namespace source;

// autoload rafisa internal organisation

include\_once \_\_DIR\_\_ . '/RIOAutoloader.php';

// autoload composer packages

$autoloader\_dir = dirname(\_\_DIR\_\_) . '/vendor/autoload.php';

$found\_autoloader\_file = @include\_once $autoloader\_dir;

if (!$found\_autoloader\_file) {

echo

$autoloader\_dir . "</br>" .

"<p>The autoloader file doesn't exist in the upon directory</p>",

'<p>Read the README.md in the root directory of this repository.</p>';

die();

}

include\_once \_\_DIR\_\_ . '/functions.php';

$env\_file = dirname(\_\_DIR\_\_) . '/.env';

$env\_exists = file\_exists($env\_file);

if (!$env\_exists) {

echo

$env\_file . "</br>" .

"<p>The environment file doesn't exist in the upon directory</p>",

'<p>Read the README.md in the root directory of this repository.</p>';

die();

}

**source/functions.php**

<?php

namespace source;

function getAbsolutePath(array $parts = [], string $after = ""): string

{

$path = "http";

$path .= "true" === $\_ENV['HTTPS'] ? "s" : "";

$path .= "://".$\_ENV['HOSTNAME']."/". getPathPartsToPath($parts) .$after;

return $path;

}

function getAPIAbsolutePath(array $parts = [], string $after = ""): string

{

$path = "http";

$path .= "true" === $\_ENV['API\_HTTPS'] ? "s" : "";

$path .= "://".$\_ENV['API\_HOSTNAME']."/". getPathPartsToPath($parts) .$after;

return $path;

}

function getPathPartsToPath(array $parts = []): string

{

$path = '';

foreach ($parts as $part) {

$path .= $part;

$path .= '/';

}

return $path;

}

**source/RIOAutoloader.php**

<?php

namespace source;

use RecursiveDirectoryIterator;

use RecursiveIteratorIterator;

/\*\*

\* Klasse um einen Autoloader zu erzeugen

\*/

RIOAutoloader::register();

class RIOAutoloader {

private static ?RIOAutoloader $autoloader = null;

private static string $filePath;

private static string $root;

public function \_\_construct() {

spl\_autoload\_register(array($this, '\_\_autoload'));

self::$root = \_\_DIR\_\_;

self::$filePath = self::$root . '/RIOClasses.php';

$this->initClassFile();

include(self::$filePath);

}

/\*

\* Erzeugt einen Autoloader, wenn es noch keinen gibt.

\*/

public static function register(): void

{

if (self::$autoloader === null) {

self::$autoloader = new RIOAutoloader();

}

}

/\*\*

\* Wenn man eine Klasse verwenden will und PHP sie nicht findet wird

\* <code>function \_\_autoload($className)</code> aufgerufen um die

\* Klasse zu inkludieren. Da die Namen der Klassen mit den Dateinamen

\* übereinstimmen, ist die Logik dahinter sehr einfach gehalten.

\*

\* @param string $className Der Name der Klasse die gesucht werden soll.

\*/

function \_\_autoload(string $className): void

{

if (isset($GLOBALS[$className])) {

if (file\_exists($GLOBALS[$className])) {

include\_once($GLOBALS[$className]);

return;

} else {

unlink(self::$filePath);

$this->initClassFile();

}

}

/\*

\* <code>RecursiveDirectoryIterator</code> enthält einen Array der

\* unter anderem den Dateipfad enthält.

\* <code>RecursiveIteratorIterator</code> geht durch diesen Pfad

\*

\* Hauptverzeichnis des Projektes auf dem Server

\* <code>$\_SERVER['DOCUMENT\_ROOT']</code>

\*/

$rii = new RecursiveIteratorIterator(new RecursiveDirectoryIterator(self::$root));

foreach ($rii as $file) {

/\*

\* Wenn <code>$file</code> kein Verzeichnis ist

\*/

if (!$file->isDir()) {

/\*

\* Wenn <code>$file</code> (ohne .php) denselben Inhalt wie

\* <code>$className</code> hat wird <code>$file</code> inkludiert.

\*

\*/

if ($file->getBaseName('.php') === $className) {

$this->saveToFile($className, $file);

include\_once($file);

}

}

}

}

private function saveToFile(string $className, string $classPath): void

{

file\_put\_contents(self::$filePath, '$GLOBALS[\'' . $className . '\']=\'' . $classPath . '\';' . PHP\_EOL, FILE\_APPEND | LOCK\_EX);

}

private function initClassFile(): void

{

if (!file\_exists(self::$filePath)) {

file\_put\_contents(self::$filePath, '<?php' . PHP\_EOL, FILE\_APPEND | LOCK\_EX);

}

}

}

**source/classes/tests/RIOWorkDayObjectTest.php**

<?php

namespace source\classes\tests;

use Exception;

use PHPUnit\Framework\ExpectationFailedException;

use PHPUnit\Framework\TestCase;

use RIODateTimeFactory;

use RIOWorkDayObject;

use Symfony\Component\Dotenv\Dotenv;

use Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface;

include\_once \_\_DIR\_\_.'\..\..\RIOAutoloader.php';

include\_once \_\_DIR\_\_.'\..\..\functions.php';

class RIOWorkDayObjectTest extends TestCase

{

/\*\*

\* @throws TransportExceptionInterface

\*/

public function testIsSaturday(): void

{

(new Dotenv())->bootEnv(dirname(\_\_DIR\_\_).'\..\..\.env');

$dateTime = RIODateTimeFactory::getDateTime("2021-05-22");

$workDay = new RIOWorkDayObject($dateTime);

$result = $workDay->isSaturday();

$this->assertEquals(true,$result,"Test has failed!");

}

/\*\*

\* @throws TransportExceptionInterface

\*/

public function testIsSunday(): void

{

(new Dotenv())->bootEnv(dirname(\_\_DIR\_\_).'\..\..\.env');

$dateTime = RIODateTimeFactory::getDateTime("2021-05-23");

$workDay = new RIOWorkDayObject($dateTime);

$result = $workDay->isSunday();

$this->assertEquals(true,$result,"Test has failed!");

}

/\*\*

\* Test

\*

\* @throws ExpectationFailedException|Exception

\* @throws TransportExceptionInterface

\*/

public function testIsHoliday(): void

{

(new Dotenv())->bootEnv(dirname(\_\_DIR\_\_).'\..\..\.env');

$dateTime = RIODateTimeFactory::getDateTime("2021-05-24");

$workDay = new RIOWorkDayObject($dateTime);

$result = $workDay->isRIOHoliday();

$this->assertEquals(true,$result,"Test has failed!");

}

/\*\*

\* @throws TransportExceptionInterface

\*/

public function testMandatoryTimeToday(): void

{

(new Dotenv())->bootEnv(dirname(\_\_DIR\_\_).'\..\..\.env');

$workDay = new RIOWorkDayObject();

$result = $workDay->isRIOHoliday() ||

$workDay->isSunday() ||

$workDay->isSaturday();

$this->assertEquals(false,$result,"Test has failed!");

}

}

**source/classes/RIOAbsentOptionObject.php**

<?php

/\*\*

\* Bei //0 sind bei ganzen Tagen 00:00 Pflichtzeit und bei halben Tagen Pflichtzeit/2

\*

\* Class RIOAbsentOptionObject

\*/

class RIOAbsentOptionObject

{

private const NOT\_EXTERNAL = "(nicht extern)";

private const SUPPLEMENTARY\_COURSE = "Ergänzungskurs"; //0

private const TELEWORKING = "Telearbeit";

private const BRIDLE = "zügeln"; //0

private const ABSENT = "abwesend";

private const COMPENSATION = "Kompensation";

private const OFFICIAL\_HOLIDAY = "offizieller Feiertag"; //0 isholiday

private const HOLIDAYS = "Ferien"; //0

private const COMPANY\_HOLIDAYS = "Betriebsferien"; //0

private const BONUS = "Bonus";

private const SICK = "krank"; //0

private const DOCTOR = "Arzt"; //0

private const ACCIDENT = "Unfall"; //0

private const MILITARY = "Militär"; //0

private const MEETING = "Besprechung";

private const REST\_IN\_PEACE = "†"; //0

private string $option;

/\*\*

\* RIOAbsentOptionObject constructor.

\*/

public function \_\_construct()

{

$this->setNotExternal();

}

static function getOptions(): array

{

return [

self::NOT\_EXTERNAL,

self::SUPPLEMENTARY\_COURSE,

self::TELEWORKING,

self::BRIDLE,

self::ABSENT,

self::COMPENSATION,

self::OFFICIAL\_HOLIDAY,

self::HOLIDAYS,

self::COMPANY\_HOLIDAYS,

self::BONUS,

self::SICK,

self::DOCTOR,

self::ACCIDENT,

self::MILITARY,

self::MEETING,

self::REST\_IN\_PEACE,

];

}

static function getNoMandatoryTimeOptions(): array

{

return [

self::SUPPLEMENTARY\_COURSE,

self::BRIDLE,

self::OFFICIAL\_HOLIDAY,

self::HOLIDAYS,

self::COMPANY\_HOLIDAYS,

self::SICK,

self::DOCTOR,

self::ACCIDENT,

self::MILITARY,

self::REST\_IN\_PEACE

];

}

/\*\*

\* @return string

\*/

public function getOption(): string

{

return $this->option;

}

/\*\*

\* @param string $option

\*/

private function setOption(string $option): void

{

$this->option = $option;

}

public function setNotExternal(): void

{

$this->setOption(self::NOT\_EXTERNAL);

}

public function setSupplementaryCourse(): void

{

$this->setOption(self::SUPPLEMENTARY\_COURSE);

}

public function setTeleworking(): void

{

$this->setOption(self::TELEWORKING);

}

public function setBridle(): void

{

$this->setOption(self::BRIDLE);

}

public function setAbsent(): void

{

$this->setOption(self::ABSENT);

}

public function setCompensation(): void

{

$this->setOption(self::COMPENSATION);

}

public function setOfficialHoliday(): void

{

$this->setOption(self::OFFICIAL\_HOLIDAY);

}

public function setHolidays(): void

{

$this->setOption(self::HOLIDAYS);

}

public function setCompanyHolidays(): void

{

$this->setOption(self::COMPANY\_HOLIDAYS);

}

public function setBonus(): void

{

$this->setOption(self::BONUS);

}

public function setSick(): void

{

$this->setOption(self::SICK);

}

public function setDoctor(): void

{

$this->setOption(self::DOCTOR);

}

public function setAccident(): void

{

$this->setOption(self::ACCIDENT);

}

public function setMilitary(): void

{

$this->setOption(self::MILITARY);

}

public function setMeeting(): void

{

$this->setOption(self::MEETING);

}

public function setRestInPeace(): void

{

$this->setOption(self::REST\_IN\_PEACE);

}

}

**source/classes/RIOAccessController.php**

<?php

use Symfony\Component\HttpFoundation\JsonResponse;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

use Twig\Environment;

class RIOAccessController extends RIOGeneralAccessController

{

private static array $renderStack = [];

private string $template = "templates";

private static bool $rendering = false;

public function \_\_construct(

string $directoryNamespace,

Environment $twig,

Request $request = null

) {

parent::\_\_construct($directoryNamespace, $twig, $request);

}

public static function elevate(callable $function, bool $ajax = false)

{

self::$renderStack[] = [

'rendering' => self::$rendering,

'ajax' => $ajax,

];

self::$rendering = false;

call\_user\_func($function);

self::$rendering = self::getCurrentRenderFrame()['rendering'];

array\_pop(self::$renderStack);

}

private static function getCurrentRenderFrame()

{

return self::$renderStack[count(self::$renderStack) - 1];

}

protected function renderPage(string $name, array $context = []): Response

{

return $this->render(

strtolower($this->getViewPath()).$name,

array\_merge(

[

'title' => $\_ENV['APP\_NAME'],

],

$context

)

);

}

protected function render(string $name, array $context = []): Response

{

$raw = $this->rawRender($name, $context);

return $this->packResponse($raw);

}

protected function getViewPath(): string

{

$cleanParts = $this->getViewPathParts();

return 'modules/'.implode('/', $cleanParts).'/';

}

public function rawRender(string $name, array $context = []): string

{

$context = array\_merge($context, [

'view\_base' => strtolower($this->getViewPath()),

'view\_base\_parts' => $this->getViewPathParts(),

'access\_controller' => $this,

]);

if (true !== self::$rendering && false === $this->isAjax()) {

// template is set by the area

$context = array\_merge($context, [

'template' => $this->template,

]);

}

return $this->renderTwig($name, $context);

}

private function renderTwig(string $name, array $context = []): string

{

if (false === self::$rendering) {

self::$rendering = true;

$response = $this->twigRender($name, $context);

self::$rendering = false;

} else {

$response = $this->twigRender($name, $context);

}

return $response;

}

private function twigRender(string $name, array $context = []): string

{

return $this->twig()->render(

$name,

$context

);

}

protected function packResponse(string $raw): Response

{

return $this->packResponseWithType($raw, 'html');

}

protected function packResponseWithType(string $raw, string $type): Response

{

if ($this->isAjax() && false === self::$rendering) {

return new JsonResponse([

'content' => $raw,

'type' => $type,

]);

} else {

return new Response($raw);

}

}

/\*\*

\* @return string[]

\*/

protected function getViewPathParts(): array

{

$dirSpace = $this->getDirectoryNamespaceString();

$parts = explode('\\', $dirSpace);

return array\_filter(

$parts,

function ($value) {

return

('' !== $value) &&

('modules' !== $value);

}

);

}

private function isAjax(): bool

{

if (self::isElevated()) {

return self::getCurrentRenderFrame()['ajax'];

} else {

$request = $this->getRequest();

return (!$request->isMethod(Request::METHOD\_GET)) && $request->headers->contains('X-Requested-With', 'XMLHttpRequest');

}

}

private static function isElevated(): bool

{

return 0 !== count(self::$renderStack);

}

}

**source/classes/RIOAdmin.php**

<?php

use MongoDB\Model\BSONArray;

use MongoDB\Model\BSONDocument;

use Symfony\Component\HttpFoundation\RedirectResponse;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Contracts\HttpClient\Exception\ClientExceptionInterface;

use Symfony\Contracts\HttpClient\Exception\RedirectionExceptionInterface;

use Symfony\Contracts\HttpClient\Exception\ServerExceptionInterface;

use Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface;

use Twig\Environment;

/\*\*

\* Class RIOAdmin

\* User must be in logged in state, otherwise user can't change other users time record

\* This area RIOAdmin should be used for editing other users past or feature data

\* Like other users view overview, editing mandatory time and change there presence in the past

\*/

class RIOAdmin extends RIOAccessController

{

public function \_\_construct(

string $directory\_namespace,

Environment $twig,

Request $request

) {

parent::\_\_construct($directory\_namespace, $twig, $request);

}

/\*\*

\* @throws Exception

\*/

public function cronJob(): Response

{

$cronJob = new RIOCronJob(RIODateTimeFactory::getDateTime());

$cronJob->usersEndStart();

}

/\*\*

\* Delete current user authentication with saved session id and username in mongodb

\*

\* @return Response

\* @throws Exception

\*/

public function logout(): Response

{

$user = new RIOUserObject($this);

$this->getUsers()->updateOne(

[ "sessionUsername" => $user->getUsername(), "sessionId" => $user->getSessionId() ],

[

'$set' => [ 'sessionId' => '' ]

]

);

$this->getSession()->invalidate();

return RIORedirect::redirectResponse();

}

/\*\*

\* @param string $location

\* @param string $date

\* @return Response

\* @throws ClientExceptionInterface

\* @throws RedirectionExceptionInterface

\* @throws ServerExceptionInterface

\* @throws TransportExceptionInterface

\*/

public function getHoliday(string $location, string $date): Response

{

$workDayObject = new RIOWorkDayObject();

return new Response($workDayObject->getHoliday($location, $date)->getContent());

}

/\*\*

\* Tries to login user by current session if saved

\*

\* @return Response

\* @throws Exception

\*/

public function sessionLogin(): Response

{

return $this->showUser();

}

/\*\*

\* @throws Exception

\*/

private function showUser(): Response

{

$user = new RIOUserObject($this);

$workday = new RIOWorkDayObject();

$monthYear = $workday->getDate()->format("m.Y");

$customTwigExtension = new RIOCustomTwigExtension($this->getRequest());

return $this->renderPage(

"user\_home.twig",

array\_merge\_recursive(

$customTwigExtension->navByActive($user->getUsername(), $monthYear, "user\_home"),

[

"timeRecordStarted" => $user->isTimeRecordStarted(),

"day" => $workday->getWeekDay(),

"date" => $workday->getFormattedDate(),

'displayUsername' => $this->getUsers()->findOne($this->getUserFromSession())["displayUsername"],

'monthYear' => $monthYear,

'sessionUsername' => $user->getUsername()

]

)

);

}

/\*\*

\* @throws Exception

\*/

public function updateUser(string $username): RedirectResponse

{

/\*\* @var BSONDocument $user \*/

$user = $this->getUsers()->findOne(['sessionUsername' => $username]);

$request = $this->getRequest();

$mandatoryTime = $request->get("mandatoryTime");

if(null !== $mandatoryTime && '' !== $mandatoryTime) {

$user->offsetSet("mandatoryTime",RIODateTimeFactory::getDateTime($mandatoryTime));

/\*\* @var DateTime $updatedMandatoryTime \*/

$updatedMandatoryTime = $user->offsetGet("mandatoryTime");

$this->getUsers()->findOneAndUpdate(

["sessionUsername" => $user->offsetGet("sessionUsername")],

['$set' => ['mandatoryTime' => $updatedMandatoryTime->format("H:i")]]

);

return RIORedirect::redirectResponse(['rioadmin', 'edituser', $username, "success"]);

}

return RIORedirect::redirectResponse(['rioadmin', 'edituser', $username, "failure"]);

}

/\*\*

\* @param string $username

\* @param string $state

\* @return RedirectResponse|Response

\*/

public function editUser(string $username, string $state): RedirectResponse|Response

{

/\*\* @var BSONDocument $user \*/

$user = $this->getUsers()->findOne(['sessionUsername' => $username]);

$workday = new RIOWorkDayObject();

$monthYear = $workday->getDate()->format("m.Y");

$customTwigExtension = new RIOCustomTwigExtension($this->getRequest());

if(null !== $state) {

$stateArray = ['state' => $state];

} else {

$stateArray = [];

}

return $this->renderPage(

"edit\_user.twig",

array\_merge(

$customTwigExtension->navByActive($user->offsetGet("sessionUsername"), $monthYear, "edit\_user"),

[

"mandatoryTime" => $user->offsetGet("mandatoryTime"),

'monthYear' => $monthYear,

'sessionUsername' => $user->offsetGet("sessionUsername")

],

$stateArray

)

);

}

/\*\*

\* Means: date like 21.04.2021 \* index of workday \* index of workday single time with start and end timestamp

\*

\* @param string $username

\* @param string $date

\* @param string $indexOfTime

\* @param string $state

\* @return RedirectResponse|Response

\* @throws Exception

\*/

public function presenceTimeCorrections(string $username, string $date, string $indexOfTime, string $state): RedirectResponse|Response

{

/\*\* @var BSONDocument $user \*/

$user = $this->getUsers()->findOne(['sessionUsername' => $username]);

if(null === $user && RIOConfig::isInDebugMode()) {

throw new Error("User with this username ".$username." doesn't exist.");

}

$workDays = $this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime($date)->format("Y"),$username);

if(null === $workDays && RIOConfig::isInDebugMode()) {

throw new Error("User ".$username." has not workday collection for the year ".RIODateTimeFactory::getDateTime($date)->format("Y"));

}

$givenTime = ["date" => $date];

/\*\* @var BSONDocument $workDay \*/

$workDay = $workDays->findOne($givenTime);

/\*\* @var BSONArray $times \*/

$times = $workDay->offsetGet("time");

/\*\* @var BSONDocument $time \*/

$time = $times->offsetGet($indexOfTime);

$workdayObject = new RIOWorkDayObject();

$monthYear = $workdayObject->getDate()->format("m.Y");

$customTwigExtension = new RIOCustomTwigExtension($this->getRequest());

if(null !== $state) {

$stateArray = ['state' => $state];

} else {

$stateArray = [];

}

return $this->renderPage(

"presence\_time\_corrections.twig",

array\_merge(

$customTwigExtension->navByActive($username, $monthYear),

[

"date" => $date,

"time" => $time,

"timeStart" => '' === $time->offsetGet("startCorrected") ? $time->offsetGet("start") : $time->offsetGet("startCorrected"),

"timeStartCorrected" => $time->offsetGet("startCorrected"),

"timeEnd" => '' === $time->offsetGet("endCorrected") ? $time->offsetGet("end") : $time->offsetGet("endCorrected"),

"timeEndCorrected" => $time->offsetGet("endCorrected"),

'displayUsername' => $user->offsetGet("displayUsername"),

'surnameUsername' => $user->offsetGet("surnameUsername"),

'presenceTime' => '' === $time->offsetGet("presenceTimeCorrected") ? $time->offsetGet("presenceTime") : $time->offsetGet("presenceTimeCorrected"),

'presenceTimeCorrected' => $time->offsetGet("presenceTimeCorrected"),

'absentOptions' => RIOAbsentOptionObject::getOptions(),

'absentAllDay' => $workDay->offsetGet("absentAllDay"),

'absentAfternoon' => $workDay->offsetGet("absentAfternoon"),

'absentMorning' => $workDay->offsetGet("absentMorning"),

'comment' => $time->offsetGet("comment"),

"mandatoryTime" => $workDay->offsetGet("mandatoryTime"),

'workingTimePerformedCorrected' => $time->offsetGet("workingTimePerformedCorrected"),

'workingTimePerformed' => '' === $time->offsetGet("workingTimePerformedCorrected") ? $time->offsetGet("workingTimePerformed") : $time->offsetGet("workingTimePerformedCorrected"),

'presenceTimeTotal' => $workDay->offsetGet("presenceTime"),

$this->getDeviationTimeTotalKey() => $workDay->offsetGet("deviationTimeTotal"),

$this->getDeviationTimeMonthlyKey() => $workDay->offsetGet("deviationTimeMonthly"),

$this->getDeviationTimeWeeklyKey() => $workDay->offsetGet("deviationTimeWeekly"),

$this->getDeviationTimeTotalCorrectedKey() => $workDay->offsetGet("deviationTimeTotalCorrected"),

'deviation' => $workDay->offsetGet("deviation"),

'deviationNegativeOrPositiveOrZero' => $workDay->offsetGet("deviationNegativeOrPositiveOrZero"),

'timeCredit' => $workDay->offsetGet("timeCredit"),

'timeCreditCorrected' => $workDay->offsetGet("timeCreditCorrected"),

'monthYear' => $monthYear,

'usernameDateTimeIndex' => [$username, $date, $indexOfTime],

'lastEditedUser' => $time->offsetGet("lastEditedUser"),

'lastEditedDate' => $time->offsetGet("lastEditedDate"),

'lastEditedTime' => $time->offsetGet("lastEditedTime")

],

$stateArray

)

);

}

/\*\*

\* @param string $username

\* @param string $monthYear

\* @return RedirectResponse|Response

\* @throws Exception

\*/

public function overview(string $username, string $monthYear): RedirectResponse|Response

{

/\*\* @var BSONDocument $user \*/

$user = $this->getUsers()->findOne(['sessionUsername' => $username]);

$previousMonthYear = $this->getAdjacentMonth($monthYear);

if([] !== $this->getUserAllPastWorkdaysByMonthYearUser($previousMonthYear, $username)) {

$previousMonthYearName = $this->getFormattedDateByDate(RIODateTimeFactory::getDateTime("01.".$previousMonthYear));

} else {

$previousMonthYearName = '';

$previousMonthYear = '';

}

$nextMonthYear = $this->getAdjacentMonth($monthYear, "next");

if('' !== $nextMonthYear) {

$nextMonthYearName = $this->getFormattedDateByDate(RIODateTimeFactory::getDateTime("01.".$nextMonthYear));

} else {

$nextMonthYearName = '';

$nextMonthYear = '';

}

$customTwigExtension = new RIOCustomTwigExtension($this->getRequest());

$currentMonthName = $this->getFormattedDateByDate(RIODateTimeFactory::getDateTime("01.".$monthYear));

$allWorkDaysFromUserPast = $this->getUserAllPastWorkdaysByMonthYearUser($monthYear, $username);

$navByActive = $customTwigExtension->navByActive($user->offsetGet("sessionUsername"), $monthYear, "overview");

$displayUsername = $user->offsetGet("displayUsername");

$surnameUsername = $user->offsetGet("surnameUsername");

$sessionUsername = $user->offsetGet("sessionUsername");

$context = [

"nav" => $navByActive["nav"],

"allWorkDaysFromUserPast" => $allWorkDaysFromUserPast,

"previousMonthName" => $previousMonthYearName,

"nextMonthName" => $nextMonthYearName,

"currentMonthName" => $currentMonthName,

"previousMonth" => $previousMonthYear,

"nextMonth" => $nextMonthYear,

'displayUsername' => $displayUsername,

'surnameUsername' => $surnameUsername,

'sessionUsername' => $sessionUsername

];

return $this->renderPage(

"overview.twig",

$context

);

}

/\*\*

\* @param string $username

\* @param string $date

\* @param string $indexOfTime

\* @return RedirectResponse|Response

\* @throws Exception

\*/

public function updatePresenceTimeCorrections(string $username, string $date, string $indexOfTime): RedirectResponse|Response

{

// By default the form hasn't changed compared to database data

$formHasChanged = false;

// Get all from requests

$request = $this->getRequest();

$comment = $request->get("comment");

$absentAllDay = $request->get($this->getAbsentAllDayKey());

$absentMorning = $request->get($this->getAbsentMorningKey());

$absentAfternoon = $request->get($this->getAbsentAfternoonKey());

$mandatoryTimeCorrected = $request->get($this->getMandatoryTimeCorrectedKey());

$startCorrected = $request->get($this->getTimeRecordingStartCorrectedKey());

$endCorrected = $request->get($this->getTimeRecordingEndCorrectedKey());

$timeCreditCorrected = $request->get($this->getTimeCreditCorrectedKey());

$presenceTimeCorrected = $request->get($this->getPresenceTimeCorrectedKey());

$workingTimePerformedCorrected = $request->get($this->getWorkingTimePerformedCorrectedKey());

$deviationTimeTotal = $request->get($this->getDeviationTimeTotalKey());

if(RIOConfig::isInDebugMode()) {

// Input validation

if(RIODateTimeFactory::getDateTime($startCorrected) > RIODateTimeFactory::getDateTime($endCorrected)) {

throw new Error("Start time cannot be after end time or end time before start time.");

}

if(false === array\_search($absentAllDay, RIOAbsentOptionObject::getOptions(), true)) {

throw new Error("The option ".$absentAllDay." doesn't exist for ".$this->getAbsentAllDayKey());

}

if(false === array\_search($absentMorning, RIOAbsentOptionObject::getOptions(), true)) {

throw new Error("The option ".$absentMorning." doesn't exist for ".$this->getAbsentMorningKey());

}

if(false === array\_search($absentAfternoon, RIOAbsentOptionObject::getOptions(), true)) {

throw new Error("The option ".$absentAfternoon." doesn't exist for ".$this->getAbsentAfternoonKey());

}

}

// Get database objects for this request

$workDays = $this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime($date)->format("Y"),$username);

$givenTime = ["date" => $date];

/\*\* @var BSONDocument $workDay \*/

$workDay = $workDays->findOne($givenTime);

/\*\* @var BSONArray $times \*/

$times = $workDay->offsetGet("time");

/\*\* @var BSONDocument $time \*/

$time = $times->offsetGet($indexOfTime);

// Check if request data is different then database data

if ($comment !== $time->offsetGet("comment")) {

$time->offsetSet("comment", $comment);

$formHasChanged = true;

}

if ($absentAllDay !== $workDay->offsetGet($this->getAbsentAllDayKey())) {

$workDay->offsetSet($this->getAbsentAllDayKey(), $absentAllDay);

$formHasChanged = true;

}

if ($absentMorning !== $workDay->offsetGet($this->getAbsentMorningKey())) {

$workDay->offsetSet($this->getAbsentMorningKey(), $absentMorning);

$formHasChanged = true;

}

if ($absentAfternoon !== $workDay->offsetGet($this->getAbsentAfternoonKey())) {

$workDay->offsetSet($this->getAbsentAfternoonKey(), $absentAfternoon);

$formHasChanged = true;

}

if ($startCorrected !== $time->offsetGet($this->getTimeRecordingStartCorrectedKey())) {

$time->offsetSet($this->getTimeRecordingStartCorrectedKey(), $startCorrected);

$formHasChanged = true;

}

if ($endCorrected !== $time->offsetGet($this->getTimeRecordingEndCorrectedKey())) {

$time->offsetSet($this->getTimeRecordingEndCorrectedKey(), $endCorrected);

$formHasChanged = true;

}

if ($mandatoryTimeCorrected !== $workDay->offsetGet($this->getMandatoryTimeCorrectedKey())) {

$workDay->offsetSet($this->getMandatoryTimeCorrectedKey(), $mandatoryTimeCorrected);

$formHasChanged = true;

}

if(

false !== array\_search($absentMorning, RIOAbsentOptionObject::getNoMandatoryTimeOptions(), true) ||

false !== array\_search($absentAfternoon, RIOAbsentOptionObject::getNoMandatoryTimeOptions(), true)

) {

// Minus half of mandatory time

$mandatoryTimeDateTime = RIODateTimeFactory::getDateTime($workDay->offsetGet($this->getMandatoryTimeKey()));

$newHours = (int) $mandatoryTimeDateTime->format("H") / 2;

$newMinutes = (int) $mandatoryTimeDateTime->format("i") / 2;

$newMandatoryTimeDateTime = RIODateTimeFactory::getDateTime($newHours.':'.$newMinutes);

$workDay->offsetSet($this->getMandatoryTimeKey(), $newMandatoryTimeDateTime->format("H:i"));

}

if(

(

false !== array\_search($absentMorning, RIOAbsentOptionObject::getNoMandatoryTimeOptions(), true) &&

false !== array\_search($absentAfternoon, RIOAbsentOptionObject::getNoMandatoryTimeOptions(), true)

) ||

false !== array\_search($absentAllDay, RIOAbsentOptionObject::getNoMandatoryTimeOptions(), true)

) {

// No mandatory time

$workDay->offsetSet($this->getMandatoryTimeKey(), "00:00");

}

if (true === $formHasChanged) {

// The form has changed

$user = $this->getUsers()->findOne($this->getUserFromSession());

$timestamp = RIODateTimeFactory::getDateTime();

$time->offsetSet("lastEditedUser", $user->offsetGet("displayUsername") . ' ' . $user->offsetGet("surnameUsername"));

$time->offsetSet("lastEditedDate", $timestamp->format("d.m.Y"));

$time->offsetSet("lastEditedTime", $timestamp->format("H:i"));

$times->offsetSet($indexOfTime, $time);

$workDays->findOneAndUpdate(

$givenTime,

[

'$set' => [

'time' => $times,

$this->getAbsentAllDayKey() => $workDay->offsetGet($this->getAbsentAllDayKey()),

$this->getAbsentMorningKey() => $workDay->offsetGet($this->getAbsentMorningKey()),

$this->getAbsentAfternoonKey() => $workDay->offsetGet($this->getAbsentAfternoonKey()),

$this->getMandatoryTimeCorrectedKey() => $workDay->offsetGet($this->getMandatoryTimeCorrectedKey()),

$this->getMandatoryTimeKey() => $workDay->offsetGet($this->getMandatoryTimeKey())

]

]

);

}

$workday = new RIOWorkDayObject();

$workday->setDate(RIODateTimeFactory::getDateTime($date));

$monthYear = $workday->getDate()->format("m.Y");

return RIORedirect::redirectResponse(["rioadmin", "overview", $username, $monthYear]);

}

public function getFormattedDateByDate(DateTime $date): string

{

$month = $date->format("m");

$year = $date->format("Y");

$monthNames = [

"Januar",

"Februar",

"März",

"April",

"Mai",

"Juni",

"Juli",

"August",

"September",

"Oktober",

"November",

"Dezember"

];

$monthName = $monthNames[(int) $month-1];

return $monthName." ".$year;

}

/\*\*

\* @param string $monthYear

\* @param string $djacent

\* @return string

\* @throws Exception

\*/

public function getAdjacentMonth(string $monthYear, string $djacent = "previous"): string

{

$date = RIODateTimeFactory::getDateTime("01.".$monthYear);

$djacentMonth = null;

if("previous" === $djacent) {

$djacentMonth = (int)$date->format("m")-1;

if(13 === $djacentMonth) {

$djacentMonth = 12;

$year = (int)$date->format("Y")-1;

} else {

$year = $date->format("Y");

}

}

if("next" === $djacent) {

$djacentMonth = (int)$date->format("m")+1;

if(13 === $djacentMonth) {

$djacentMonth = 1;

$year = (int)$date->format("Y")+1;

} else {

$year = $date->format("Y");

}

}

if(RIOConfig::isInDebugMode()) {

if(null === $djacentMonth || null === $year) {

throw new Error("Wrong adjacent argument can be previous or next");

}

}

$currentDate = RIODateTimeFactory::getDateTime();

$date->setDate($year, $djacentMonth, 1);

if($date > $currentDate) {

return '';

}

return $date->format("m.Y");

}

/\*\*

\* Check if theres an work day exist for the session user

\* if not then insert new work day with new start time

\* if yes then update existing work day with mew start time

\*

\* @return RedirectResponse|Response

\* @throws Exception|TransportExceptionInterface

\*/

public function start(): RedirectResponse|Response

{

$user = new RIOUserObject($this);

if($user->isTimeRecordStopped()){

/\*\* @var BSONDocument $findOneWorkDay \*/

$findOneWorkDay = $this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime()->format("Y"),$user->getUsername())->findOne(

$this->getDate()

);

if(null === $findOneWorkDay) {

$this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime()->format("Y"),$user->getUsername())->insertOne(

$this->getTimeRecording()

);

} else {

// Existing workday to update on one user

$findOneWorkDay->offsetGet("time")[] = $this->getTimeRecordingStart();

$this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime()->format("Y"),$user->getUsername())->findOneAndUpdate(

$this->getTimeRecordingFilter(),

['$set' => [ 'time' => $findOneWorkDay->offsetGet("time") ]]

);

}

$this->getUsers()->updateOne(

[ "sessionUsername" => $user->getUsername(), 'timeRecordStarted' => false ],

['$set' => [ 'timeRecordStarted' => true ]]

);

}

return RIORedirect::redirectResponse(["rioadmin", "sessionLogin"]);

}

/\*\*

\* @return RedirectResponse|Response

\* @throws Exception

\*/

public function stop(): RedirectResponse|Response

{

$user = new RIOUserObject($this);

if($user->isTimeRecordStarted()) {

/\*\* @var BSONDocument $findOneWorkDay \*/

$findOneWorkDay = $this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime()->format("Y"),$user->getUsername())->findOne($this->getDate());

/\*\* @var BSONArray $times \*/

$times = $findOneWorkDay->offsetGet("time");

/\*\*\* @var BSONArray $time \*/

foreach ($times as $time) {

$endTime = $this->getTimeRecordingEndValue();

if(false === $time->offsetExists($this->getTimeRecordingEndKey())) {

$time->offsetSet($this->getTimeRecordingEndKey(), $endTime);

}

if(false === $time->offsetExists($this->getTimeRecordingEndCorrectedKey())) {

$time->offsetSet($this->getTimeRecordingEndCorrectedKey(), '');

}

$timeObject = new RIOTimeObject();

$timeObject->setTimeStart(RIODateTimeFactory::getDateTime($time->offsetGet("start")));

$timeObject->setTimeEnd(RIODateTimeFactory::getDateTime($endTime));

$origin = $timeObject->getTimeStart();

$target = $timeObject->getTimeEnd();

$interval = $origin->diff($target);

$presenceTime = RIODateTimeFactory::getDateTime($interval->h.':'.$interval->i);

if(false === $time->offsetExists($this->getWorkingTimePerformedKey())) {

// TODO: workingTimePerformed take temporary the same value like presenceTime, actual calculation workingTimePerformed = presenceTime - breaks

$time->offsetSet($this->getWorkingTimePerformedKey(), $presenceTime->format("H:i"));

}

if(false === $time->offsetExists($this->getWorkingTimePerformedCorrectedKey())) {

$time->offsetSet($this->getWorkingTimePerformedCorrectedKey(), '');

}

if(false === $time->offsetExists($this->getPresenceTimeKey())) {

$time->offsetSet($this->getPresenceTimeKey(), $presenceTime->format("H:i"));

}

if(false === $time->offsetExists($this->getPresenceTimeCorrectedKey())) {

$time->offsetSet($this->getPresenceTimeCorrectedKey(), '');

}

}

$start = "00:00";

$startTime = RIODateTimeFactory::getDateTime($start);

$presenceTimeTotal = RIODateTimeFactory::getDateTime($start);

$times = $findOneWorkDay->offsetGet("time");

/\*\* @var BSONDocument $time \*/

foreach ($times as $time) {

if('' !== $time->offsetGet($this->getPresenceTimeCorrectedKey())) {

$addableTime = $time->offsetGet($this->getPresenceTimeCorrectedKey());

} else {

if('' !== $time->offsetGet($this->getTimeRecordingStartCorrectedKey())) {

$addableTimeStart = $time->offsetGet($this->getTimeRecordingStartCorrectedKey());

} else {

$addableTimeStart = $time->offsetGet($this->getTimeRecordingStartKey());

}

if('' !== $time->offsetGet($this->getTimeRecordingEndCorrectedKey())) {

$addableTimeEnd = $time->offsetGet($this->getTimeRecordingEndCorrectedKey());

} else {

$addableTimeEnd = $time->offsetGet($this->getTimeRecordingEndKey());

}

$start = RIODateTimeFactory::getDateTime($addableTimeStart);

$end = RIODateTimeFactory::getDateTime($addableTimeEnd);

$startEndDiff = RIODateTimeFactory::getDateTime();

/\*\* @var DateInterval|false $diff \*/

$diff = $start->diff($end);

$startEndDiff->setTime($diff->h, $diff->i);

if($startEndDiff->format("H:i") === $time->offsetGet($this->getPresenceTimeKey())) {

$addableTime = $time->offsetGet($this->getPresenceTimeKey());

} else {

$addableTime = $startEndDiff->format("H:i");

}

}

$presenceTime = RIODateTimeFactory::getDateTime($addableTime);

$presenceTimeTotal->add($startTime->diff($presenceTime));

if(false === $time->offsetExists($this->getIsTimeKey())) {

$time->offsetSet($this->getIsTimeKey(), $presenceTimeTotal->format("H:i"));

}

if(false === $time->offsetExists($this->getDiffKey())) {

$isTimeMandatoryTimeTimeDiff = RIODateTimeFactory::getDateTime();

$mandatoryTime = $findOneWorkDay->offsetGet($this->getMandatoryTimeKey());

$mandatoryTimeTime = RIODateTimeFactory::getDateTime($mandatoryTime);

$isTime = RIODateTimeFactory::getDateTime($time->offsetGet($this->getIsTimeKey()));

$diffIsTimeMandatoryTimeTime = $mandatoryTimeTime->diff($isTime);

$isTimeMandatoryTimeTimeDiff->setTime($diffIsTimeMandatoryTimeTime->h, $diffIsTimeMandatoryTimeTime->i);

$time->offsetSet($this->getDiffKey(), $isTimeMandatoryTimeTimeDiff->format("H:i"));

$isTimeNegativeOrPositiveOrZero = '';

if($isTime->format("H:i") === $mandatoryTime) {

$isTimeNegativeOrPositiveOrZero .= ' ';

}

if($isTime->format("H:i") > $mandatoryTime) {

$isTimeNegativeOrPositiveOrZero .= '+';

}

if($isTime->format("H:i") < $mandatoryTime) {

$isTimeNegativeOrPositiveOrZero .= '-';

}

if(false === $time->offsetExists($this->getDiffNegativePositiveKey())) {

$time->offsetSet($this->getDiffNegativePositiveKey(), $isTimeNegativeOrPositiveOrZero);

}

}

}

$mandatoryTime = RIODateTimeFactory::getDateTime($findOneWorkDay->offsetGet($this->getMandatoryTimeKey()));

$deviationDiff = $mandatoryTime->diff($presenceTimeTotal);

$deviation = RIODateTimeFactory::getDateTime();

$deviation->setTime($deviationDiff->h, $deviationDiff->i);

$deviationNegativeOrPositiveOrZero = '';

if($presenceTimeTotal->format("H:i") === $findOneWorkDay->offsetGet($this->getMandatoryTimeKey())) {

$deviationNegativeOrPositiveOrZero .= ' ';

}

if($presenceTimeTotal->format("H:i") > $findOneWorkDay->offsetGet($this->getMandatoryTimeKey())) {

$deviationNegativeOrPositiveOrZero .= '+';

}

if($presenceTimeTotal->format("H:i") < $findOneWorkDay->offsetGet($this->getMandatoryTimeKey())) {

$deviationNegativeOrPositiveOrZero .= '-';

}

$userMonth = $this->getUserAllPastWorkdaysByMonthYear($findOneWorkDay->offsetGet("monthYear"), $this);

$userTotal = $this->getUserAllPastWorkdays($this);

$userWeek = $this->getUserAllPastWorkdaysByWeek($findOneWorkDay->offsetGet("weekYear"), $this);

$isTimeMonthly = RIODateTimeFactory::getDateTime();

$isTimeMonthly->setTime(0, 0);

$mandatoryTimeMonthly = RIODateTimeFactory::getDateTime();

$mandatoryTimeMonthly->setTime(0, 0);

$deviationTimeMonthly = RIODateTimeFactory::getDateTime();

$deviationTimeMonthly->setTime(0, 0);

/\*\* @var BSONDocument $day \*/

foreach ($userMonth as $day) {

$dayTime = RIODateTimeFactory::getDateTime($day->offsetGet("presenceTime"));

$dayMandatoryTime = RIODateTimeFactory::getDateTime($day->offsetGet("mandatoryTime"));

$isTimeMonthly->add($startTime->diff($dayTime));

$mandatoryTimeMonthly->add($startTime->diff($dayMandatoryTime));

}

$deviationTimeMonthly->add($mandatoryTimeMonthly->diff($isTimeMonthly));

$isTimeTotal = RIODateTimeFactory::getDateTime();

$isTimeTotal->setTime(0, 0);

$mandatoryTimeTotal = RIODateTimeFactory::getDateTime();

$mandatoryTimeTotal->setTime(0, 0);

$deviationTimeTotal = RIODateTimeFactory::getDateTime();

$deviationTimeTotal->setTime(0, 0);

foreach ($userTotal as $day) {

$dayTime = RIODateTimeFactory::getDateTime($day->offsetGet("presenceTime"));

$dayMandatoryTime = RIODateTimeFactory::getDateTime($day->offsetGet("mandatoryTime"));

$isTimeTotal->add($startTime->diff($dayTime));

$mandatoryTimeTotal->add($startTime->diff($dayMandatoryTime));

}

$deviationTimeTotal->add($mandatoryTimeTotal->diff($isTimeTotal));

$isTimeWeekly = RIODateTimeFactory::getDateTime();

$isTimeWeekly->setTime(0, 0);

$mandatoryTimeWeekly = RIODateTimeFactory::getDateTime();

$mandatoryTimeWeekly->setTime(0, 0);

$deviationTimeWeekly = RIODateTimeFactory::getDateTime();

$deviationTimeWeekly->setTime(0, 0);

foreach ($userWeek as $day) {

$dayTime = RIODateTimeFactory::getDateTime($day->offsetGet("presenceTime"));

$dayMandatoryTime = RIODateTimeFactory::getDateTime($day->offsetGet("mandatoryTime"));

$isTimeWeekly->add($startTime->diff($dayTime));

$mandatoryTimeWeekly->add($startTime->diff($dayMandatoryTime));

}

$deviationTimeWeekly->add($mandatoryTimeWeekly->diff($isTimeWeekly));

$this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime()->format("Y"),$user->getUsername())->findOneAndUpdate(

$this->getTimeRecordingFilterStarted(),

['$set' =>

[

'presenceTime' => $presenceTimeTotal->format("H:i"),

'presenceTimeCorrected' => '',

'time' => $findOneWorkDay->offsetGet("time"),

'deviation' => $deviation->format("H:i"),

'deviationNegativeOrPositiveOrZero' => $deviationNegativeOrPositiveOrZero,

'isTimeMonthly' => $isTimeMonthly->format("H:i"),

'isTimeTotal' => $isTimeTotal->format("H:i"),

'isTimeWeekly' => $isTimeWeekly->format("H:i"),

// for nightly cronjob

$this->getMandatoryTimeMonthlyKey() => $mandatoryTimeMonthly->format("H:i"),

//$this->getMandatoryTimeMonthlyKey() => $final,

$this->getMandatoryTimeTotalKey() => $mandatoryTimeTotal->format("H:i"),

$this->getMandatoryTimeWeeklyKey() => $mandatoryTimeWeekly->format("H:i"),

// for nightly cronjob

$this->getDeviationTimeMonthlyKey() => $deviationTimeMonthly->format("H:i"),

$this->getDeviationTimeTotalKey() => $deviationTimeTotal->format("H:i"),

//$this->getDeviationTimeMonthlyKey() => $lastDayThisMonth->offsetGet($this->getDeviationTimeMonthlyKey()),

//$this->getDeviationTimeTotalKey() => $lastDayThisMonth->offsetGet($this->getDeviationTimeTotalKey()),

$this->getDeviationTimeWeeklyKey() => $mandatoryTimeWeekly->format("H:i"),

$this->getDeviationTimeTotalCorrectedKey() => '',

]

]

);

$this->getUsers()->findOneAndUpdate(

["sessionUsername" => $user->getUsername(), 'timeRecordStarted' => true],

['$set' => ['timeRecordStarted' => false]]

);

}

return RIORedirect::redirectResponse(["rioadmin", "sessionLogin"]);

}

}

**source/classes/RIOApplication.php**

<?php

declare(strict\_types=1);

use Symfony\Component\HttpFoundation\RedirectResponse;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

use Twig\Environment;

use Whoops\Handler\PrettyPageHandler;

use Whoops\Run;

/\*\*

\* For each client request there is only one instance of this class

\*

\* Class RIOApplication

\*/

class RIOApplication

{

/\*\*

\* instance

\*

\* @var null|RIOApplication

\*/

protected static ?RIOApplication $instance = null;

public static array $perfData = [];

private static PrettyPageHandler $handler;

private static Request $request;

/\*\*

\* Singleton

\* RIOApplication constructor.

\*/

protected function \_\_construct() {

}

protected function \_\_clone(): void {}

public static function getInstance(): self

{

if (null === self::$instance)

{

self::$instance = new self;

}

return self::$instance;

}

public function launch(Request $request): RedirectResponse|Response

{

if(RIOConfig::isDevelopmentMode()) {

$run = new Run();

if (RIOConfig::isInDebugMode()) {

self::$handler = new PrettyPageHandler();

$run->pushHandler(self::$handler);

assert\_options(ASSERT\_ACTIVE, true);

}

$run->register();

}

return $this->serveClientRequest($request);

}

private function serveClientRequest(Request $request): RedirectResponse|Response

{

try {

self::$request = $request;

$resolvedAction = $this->getAreaPathParser();

return $this->resolveAndExecuteActionControllerMethodParameter($resolvedAction);

} catch (RIONotFoundException $e) {

if (RIOConfig::isInDebugMode()) {

throw new Error("The page doesn't exist or is offline", 0, $e);

} else {

return RIORedirect::error(404);

}

} catch (ReflectionException | \Whoops\Exception\ErrorException) {

}

return RIORedirect::redirectWithString('');

}

/\*\*

\* @param RIOResolvedAction $resolvedAction

\* @return Response|null

\* @throws RIONotFoundException

\* @throws ReflectionException

\* @throws \Whoops\Exception\ErrorException

\*/

private function resolveAndExecuteActionControllerMethodParameter(RIOResolvedAction $resolvedAction): RIORedirect|Response

{

$request = self::$request;

$class = ucfirst($this->getAreaPathParser()->getFrontend()->getValue());

if(isset($class)) {

if(isset($this->getAreaPathParser()->getPathPartials()[0])) {

$method = $this->getAreaPathParser()->getPathPartials()[0];

}

$i = 1;

$args = [];

/\*\* @var string[] $pathPartials \*/

$pathPartials = [$this->getAreaPathParser()->getPathPartials()];

foreach ($pathPartials as $pathPartial) {

if($i >= 1) {

$args[] = $pathPartial;

}

$i++;

}

unset($args[0][0]);

$userRequestedParameters = $args[0];

} else {

// Controller not isset

return RIORedirect::error(404);

}

if("Rioadmin" === $class) {

$customTwigExtension = new RIOCustomTwigExtension($request);

$isNotLoggedIn = !$customTwigExtension->isLoggedIn();

if($isNotLoggedIn) {

return RIORedirect::redirectResponse();

}

}

$twig = self::getTwig($resolvedAction->getFrontend()->getValue());

$className = '';

if("Rioadmin" === $class) {

$className .= "RIOAdmin";

/\*\* @var RIOMain|RIOAdmin $instance \*/

$instance = new $className($className, $twig, $request);

} elseif("Riomain" === $class) {

$className .= "RIOMain";

/\*\* @var RIOMain|RIOAdmin $instance \*/

$instance = new $className($className, $twig, $request);

} else {

/\*\* @var RIOMain|RIOAdmin $instance \*/

$instance = new $class($class, $twig, $request);

}

if(false === RIOMaybe::ofSettable($class)->isEmpty()) {

if(!class\_exists($class)) {

return RIORedirectOrException::throwErrorException("Controller class called '".$class."' doesn't exist");

}

if(false === RIOMaybe::ofSettable($method)->isEmpty()) {

if(!method\_exists($instance, $method)) {

return RIORedirectOrException::throwErrorException("Method called '".$method."' doesn't exist in controller class called '".$class."'");

}

$methodSignature = new ReflectionMethod($instance, $method);

$numberOfUserRequestedParameters = count($userRequestedParameters);

$numberOfRequiredParameters = $methodSignature->getNumberOfRequiredParameters();

/\*\* @var ReflectionParameter $parameters \*/

$parameters = $methodSignature->getParameters();

// This is to make error message more comprehensive for developers

$paramNames = [];

foreach ($parameters as $parameter) {

$paramNames[] = $parameter->getName();

}

$nameParam = '';

$x = 0;

foreach ($paramNames as $paramName) {

if($x > 0) {

$nameParam .= ', ';

}

$nameParam .= $paramName;

$x++;

}

if($numberOfRequiredParameters !== $numberOfUserRequestedParameters) {

return RIORedirectOrException::throwErrorException(

"User requested ".

$numberOfUserRequestedParameters.

" parameter/s for controller ".

$class.

" with method ".

$method.

", which actually required ".$numberOfRequiredParameters.

" parameter/s called ".

$nameParam.

"."

);

}

$userParamIteration = 0;

$allParametersWithRequiredTypes = [];

foreach ($parameters as $parameter) {

$paramValueUser = $this->getAreaPathParser()->getPathPartials()[$userParamIteration+1];

$paramTypeUser = gettype($paramValueUser);

$paramTypeSystem = $parameter->getType()->getName();

if($paramTypeUser !== $paramTypeSystem) {

// User value its type isn't the same type as required

if(!settype($paramValueUser, $paramTypeSystem)) {

if(RIOConfig::isInDebugMode()) {

throw new \Whoops\Exception\ErrorException(

"The name of the requested parameter at position ".

$userParamIteration.

" === ".

$parameter->getPosition().

" is ".

$parameter->getName().

" We had to set value to a different type with settype(value\_with\_wrong\_type, actual\_type\_we\_need) which somehow failed to change. The type of requested value user parameter value is ".

$paramValueUser.

" and type of ".

$paramTypeUser.

" and the required type of this parameter is a/n ".

$paramTypeSystem

);

}

}

}

$allParametersWithRequiredTypes[] = $paramValueUser;

$userParamIteration++;

}

if(0 === $numberOfRequiredParameters && 0 === $numberOfUserRequestedParameters) {

// No parameter required or optional or zero possible parameter, user requested with 0 parameter

$response = call\_user\_func(

[

$instance

, $this->getAreaPathParser()->getPathPartials()[0]

]

);

}

if(1 === $numberOfRequiredParameters && 1 === $numberOfUserRequestedParameters) {

// One parameter required and user has requested with one parameter

$response = call\_user\_func(

[

$instance

, $this->getAreaPathParser()->getPathPartials()[0]

]

, $allParametersWithRequiredTypes[0]

);

}

if(1 < $numberOfRequiredParameters) {

// Multiple parameters required and user has requested with multiple parameters

$response = call\_user\_func\_array(

[

$instance

, $this->getAreaPathParser()->getPathPartials()[0]

]

, $allParametersWithRequiredTypes

);

}

if(0 === $numberOfRequiredParameters) {

$response = call\_user\_func(

[

$instance

, $this->getAreaPathParser()->getPathPartials()[0]

]

);

}

if(false === isset($response)) {

return RIORedirectOrException::throwErrorException("Couldn't create valid controller method response for number of parameters may required", 500);

}

return $response;

}

}

return call\_user\_func([$instance, $\_ENV['EMPTY\_URL\_CONTROLLER']]);

}

private static function getTwig(string $areaName): Environment

{

/\*\* @var RIOTwigProvider $twigProvider \*/

$twigProvider = RIOAreaSpecificRIOTwigProvider::getInstance($areaName, self::$request);

return $twigProvider->getTwig();

}

/\*\*

\* @throws RIONotFoundException

\*/

public function getAreaPathParser(): RIOResolvedAction

{

$path = urldecode($\_SERVER['REQUEST\_URI']);

$pathConfig = RIOResolverConfigFactory::config([

RIOResolverConfigFactory::step([new RIOFrontendResolver($\_ENV['DEFAULT\_AREA\_NAME'], [

'riomain',

'rioadmin',

])]),

]);

$pathResolver = new RIOResolver($pathConfig);

$resolvedAction = $pathResolver->resolveWithString($path);

RIOApplication::addDebugInformation('Uri', [

'frontend' => $resolvedAction->getFrontend(),

'controller namespace' => $resolvedAction->getControllerNamespace(),

'controller action' => $resolvedAction->getControllerAction(),

'controller action parameters' => $resolvedAction->getControllerActionParameters(),

]);

return $resolvedAction;

}

public static function addDebugInformation(string $title, array $data): void

{

if (RIOConfig::isInDebugMode()) {

self::$handler->addDataTable($title, $data);

}

}

}

**source/classes/RIOAreaSpecificRIOTwigProvider.php**

<?php

declare(strict\_types=1);

use Symfony\Component\HttpFoundation\Request;

use Twig\Environment;

use Twig\Extension\DebugExtension;

use Twig\Loader\ChainLoader;

use Twig\Loader\FilesystemLoader;

final class RIOAreaSpecificRIOTwigProvider implements RIOTwigProvider

{

/\*\*

\* instance

\*

\* @var null|RIOAreaSpecificRIOTwigProvider

\*/

protected static ?RIOAreaSpecificRIOTwigProvider $\_instance = null;

private string $area\_name;

private const ROOT\_PREFIX = '';

private const PREFIX\_DIR = self::ROOT\_PREFIX.'templates/';

private Request $request;

protected function \_\_construct(string $area\_name, Request $request)

{

$this->area\_name = $area\_name;

$this->request = $request;

}

protected function \_\_clone(): void {}

public static function getInstance(string $area\_name, Request $request): self

{

if (null === self::$\_instance)

{

self::$\_instance = new self($area\_name, $request);

}

return self::$\_instance;

}

public function getTwig(): Environment

{

$debug\_mode\_enabled = RIOConfig::isInDebugMode();

$area\_name = $this->area\_name;

$area\_inheritance = [

'email' => 'riomain',

];

if ($debug\_mode\_enabled) {

$this->checkIfDirectoriesExist($area\_name);

}

$loaders = [];

$loaders = array\_merge(

$loaders,

$this->getLoadersOfArea($area\_name)

);

$current\_area\_name = $area\_name;

while (isset($area\_inheritance[$current\_area\_name])) {

$current\_area\_name = $area\_inheritance[$current\_area\_name];

if ($debug\_mode\_enabled) {

$this->checkIfDirectoriesExist($current\_area\_name);

}

$loaders = array\_merge(

$loaders,

$this->getLoadersOfArea($current\_area\_name)

);

}

$loaders[] = new FilesystemLoader(self::ROOT\_PREFIX.'views', dirname(\_\_DIR\_\_));

$chain = new ChainLoader($loaders);

$twig\_arguments = [

'debug' => $debug\_mode\_enabled,

'cache' => \_\_DIR\_\_ . '/../../cache/twig',

];

$twig = new Environment($chain, $twig\_arguments);

if ($debug\_mode\_enabled) {

$twig->enableAutoReload();

} else {

$twig->disableAutoReload();

}

$twig->addExtension(new RIOCustomTwigExtension($this->request));

if ($debug\_mode\_enabled) {

$twig->addExtension(new DebugExtension());

}

return $twig;

}

private function getLoadersOfArea(string $area\_name): array

{

return [

new FilesystemLoader(self::PREFIX\_DIR."$area\_name", dirname(\_\_DIR\_\_)),

];

}

private function checkIfDirectoriesExist(string $area\_name): void

{

if(RIOConfig::isInDebugMode()) {

if (!is\_dir(

dirname(\_\_DIR\_\_) . '/'

)) {

throw new Error("The twig template folder \"$area\_name/\" is missing. "."The path \"/source/templates/$area\_name/\" should exist.");

}

}

}

}

**source/classes/RIOConfig.php**

<?php

declare(strict\_types=1);

class RIOConfig

{

public static function isInDebugMode(): bool

{

return "true" === $\_ENV['DEBUG'];

}

public static function isDevelopmentMode(): bool

{

return "true" === $\_ENV['DEVELOPMENT\_MODE'];

}

}

**source/classes/RIOConnectionFailed.php**

<?php

declare(strict\_types=1);

/\*\*

\* If the database connection has failed

\* Class RIOConnectionFailed

\*/

class RIOConnectionFailed extends Error

{

}

**source/classes/RIOCronJob.php**

<?php

use MongoDB\Collection;

use MongoDB\Model\BSONDocument;

use Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface;

/\*\*

\* Nightly cronjob which calc today with last day or only today

\*

\* Class RIOCronJob

\*/

class RIOCronJob

{

/\*\*

\* Have to update data in database

\* @var RIOMongoDatabase|null

\*/

private ?RIOMongoDatabase $mongoDB;

private DateTime $dateTime;

/\*\*

\* RIOCronJob constructor.

\*/

public function \_\_construct(DateTime $dateTime)

{

$this->mongoDB = RIOMongoDatabase::getInstance();

$this->dateTime = $dateTime;

}

/\*\*

\* Day ends at 23:59 and starts at 00:00

\*

\* @throws Exception

\*/

public function usersEndStart(): void

{

// The current year

$year = $this->dateTime->format("Y");

// User which haven't or have time record started

$users = $this->mongoDB->getUsers();

// Inactive user which haven't time record started

$usersInactive = $this->mongoDB->getInactiveUsers();

// Active user which haven time record started

$usersActive = $this->mongoDB->getActiveUsers();

$today = ['date' => $this->dateTime->format("d.m.Y")];

$yesterday = ['date' => RIODateTimeFactory::getDateTime('yesterday')->format("d.m.Y")];

// Update active users

//$this->calcActiveUsers($usersActive, $today, $year);

// Update users

$this->calcUsers($users, $today, $year, $yesterday);

// Update inactive users

//$this->calcInactiveUsers($usersInactive, $today, $year);

}

/\*\*

\* @param BSONDocument[] $users

\* @param array $today

\* @param string $year

\* @param array $yesterday

\* @throws Exception

\*/

private function calcUsers(array $users, array $today, string $year, array $yesterday): void

{

foreach ($users as $user) {

$workDaysCollection = $this->mongoDB->getWorkDaysCollectionByYearUser($year, $user->offsetGet("sessionUsername"));

/\*\* @var BSONDocument $yesterdayWorkDay \*/

$yesterdayWorkDay = $workDaysCollection->findOne($yesterday);

/\*\* @var BSONDocument $findOneWorkDay \*/

$findOneWorkDay = $workDaysCollection->findOne($today);

// Get all mandatory time calc

$timeMonthlyWeeklyAndTotal = $this->calcMandatoryTimeMonthlyWeeklyAndTotal($workDaysCollection, $findOneWorkDay);

// Calc mandatory time monthly

$findOneWorkDay->offsetSet("mandatoryTimeMonthly", $timeMonthlyWeeklyAndTotal["mandatoryTimeMonthly"]);

// Calc mandatory time weekly

$findOneWorkDay->offsetSet("mandatoryTimeWeekly", $timeMonthlyWeeklyAndTotal["mandatoryTimeWeekly"]);

// Calc mandatory time total

$findOneWorkDay->offsetSet("mandatoryTimeTotal", $timeMonthlyWeeklyAndTotal["mandatoryTimeTotal"]);

$isTimeMonthly = $this->calcIsTimeMonthly($workDaysCollection, $findOneWorkDay);

$deviationTimeMonthly = $this->calcDeviationTimeMonthly($isTimeMonthly, $findOneWorkDay->offsetGet("mandatoryTimeMonthly"));

$deviationTimeTotal = $this->calcDeviationTimeTotal($findOneWorkDay, $yesterdayWorkDay);

$deviationTimeWeekly = $this->calcDeviationTimeWeekly($findOneWorkDay, $yesterdayWorkDay);

$updateFutureImplementation = [

'mandatoryTimeWeekly' => $findOneWorkDay->offsetGet("mandatoryTimeWeekly"),

'mandatoryTimeTotal' => $findOneWorkDay->offsetGet("mandatoryTimeTotal")

];

$update = [

'mandatoryTimeMonthly' => $findOneWorkDay->offsetGet("mandatoryTimeMonthly"),

'isTimeMonthly' => $isTimeMonthly,

'deviationTimeMonthly' => $deviationTimeMonthly,

'deviationTimeTotal' => $deviationTimeTotal,

'deviationTimeWeekly' => $deviationTimeWeekly

];

echo "<pre>";

var\_dump($update);

echo "</pre>";

die();

// Save today

$workDaysCollection->updateOne(

[

'date' => $this->dateTime->format("d.m.Y")

],

[

'$set' => $update

]

);

}

}

private function calcInactiveUsers($usersInactive, $today, $year): void

{

foreach ($usersInactive as $userInactive) {

$timeRecordStarted = false;

$workDaysCollection = $this->mongoDB->getWorkDaysCollectionByYearUser($year, $userInactive->offsetGet("sessionUsername"));

/\*\* @var BSONDocument $findOneWorkDay \*/

$findOneWorkDay = $workDaysCollection->findOne($today);

// Check if times is still empty because of active user without start record

if([] === $findOneWorkDay->offsetGet("time")) {

// Insert 0 minute workday entry for inactive user

$stopTimes = $this->updateInactiveTimes();

} else {

$stopTimes = $findOneWorkDay->offsetGet("time");

}

// Save today

$workDaysCollection->updateOne(

[

'date' => $this->dateTime->format("d.m.Y")

],

[

'$set' => $stopTimes

]

);

// Save tomorrow, create new work day for the next day

$this->createNewWorkDay($workDaysCollection, $userInactive, $timeRecordStarted);

}

}

private function calcActiveUsers($usersActive, $today, $year): void

{

foreach ($usersActive as $userActive) {

$timeRecordStarted = true;

$workDaysCollection = $this->mongoDB->getWorkDaysCollectionByYearUser($year, $userActive->offsetGet("sessionUsername"));

/\*\* @var BSONDocument $findOneWorkDay \*/

$findOneWorkDay = $workDaysCollection->findOne($today);

// Check if times is still empty because of active user without start record

if([] === $findOneWorkDay->offsetGet("time")) {

$timeRecordStarted = false;

// Set active user inactive because of 24 hour inactivity

$this->mongoDB->getUsersCollection()->findOneAndUpdate(

["sessionUsername" => $userActive->offsetGet("sessionUsername"), 'timeRecordStarted' => $userActive->offsetGet("timeRecordStarted")],

['$set' => ['timeRecordStarted' => $timeRecordStarted]]

);

// Insert 0 minute workday entry for active user

$stopTimes = $this->updateInactiveTimes();

} else {

// All active users stop time record for today

$stopTimes = $this->updateTodayTimes($findOneWorkDay, $userActive->offsetGet("sessionUsername"));

}

// Save today

$workDaysCollection->updateOne(

[

'date' => $this->dateTime->format("d.m.Y")

],

[

'$set' => $stopTimes

]

);

// Save tomorrow, create new work day for the next day, all active users start time record for the next day

$this->createNewWorkDay($workDaysCollection, $userActive, $timeRecordStarted);

}

}

/\*\*

\* Set 0 hour and 0 minute entry for user current work day was empty

\*

\* @return array

\*/

private function updateInactiveTimes(): array

{

$timeStart =

[

'start' => '00:00',

'startCorrected' => '',

'comment' => '',

'lastEditedUser' => '',

'lastEditedDate' => '',

'lastEditedTime' => ''

];

$timeEnd = [

];

return [

"time" => array\_merge(

$timeStart,

$timeEnd

)

];

}

/\*\*

\* Stop time, calc time

\*

\* @param BSONDocument $findOneWorkDay

\* @param string $sessionUsername

\* @return array

\* @throws Exception

\*/

private function updateTodayTimes(BSONDocument $findOneWorkDay, string $sessionUsername): array

{

$times = $findOneWorkDay->offsetGet("time");

$end = "23:59";

$endDateTime = $this->dateTime->format("H:i");

if($end === $endDateTime) {

$endTime = $endDateTime;

} else {

$endTime = $end;

}

$start = "00:00";

$startTime = RIODateTimeFactory::getDateTime($start);

$presenceTimeTotal = RIODateTimeFactory::getDateTime($start);

foreach ($times as $time) {

if(false === $time->offsetExists("end")) {

$time->offsetSet("end", $endTime);

}

if(false === $time->offsetExists("endCorrected")) {

$time->offsetSet("endCorrected", '');

}

$addableTimeStart = $time->offsetGet("start");

$addableTimeEnd = $endTime;

$start = RIODateTimeFactory::getDateTime($addableTimeStart);

$end = RIODateTimeFactory::getDateTime($addableTimeEnd);

$startEndDiff = RIODateTimeFactory::getDateTime();

/\*\* @var DateInterval|false $diff \*/

$diff = $start->diff($end);

$startEndDiff->setTime($diff->h, $diff->i);

if($startEndDiff->format("H:i") === $time->offsetGet("presenceTime")) {

$addableTime = $time->offsetGet("presenceTime");

} else {

$addableTime = $startEndDiff->format("H:i");

}

}

if(null === $addableTime) {

$addableTime = "00:00";

}

$presenceTime = RIODateTimeFactory::getDateTime($addableTime);

$presenceTimeTotal->add($startTime->diff($presenceTime));

$mandatoryTime = RIODateTimeFactory::getDateTime($findOneWorkDay->offsetGet("mandatoryTime"));

$deviationDiff = $mandatoryTime->diff($presenceTimeTotal);

$deviation = RIODateTimeFactory::getDateTime();

$deviation->setTime($deviationDiff->h, $deviationDiff->i);

$deviationNegativeOrPositiveOrZero = '';

if($presenceTimeTotal->format("H:i") === $findOneWorkDay->offsetGet("mandatoryTime")) {

$deviationNegativeOrPositiveOrZero .= ' ';

}

if($presenceTimeTotal->format("H:i") > $findOneWorkDay->offsetGet("mandatoryTime")) {

$deviationNegativeOrPositiveOrZero .= '+';

}

if($presenceTimeTotal->format("H:i") < $findOneWorkDay->offsetGet("mandatoryTime")) {

$deviationNegativeOrPositiveOrZero .= '-';

}

//$userMonth = getUserAllPastWorkdaysByMonthYearUser($findOneWorkDay->offsetGet("monthYear"), "l.gaechter");

$userTotal = $this->getUserAllPastWorkdays($sessionUsername);

//$userWeek = getUserAllPastWorkdaysByWeek($findOneWorkDay->offsetGet("weekYear"), "l.gaechter");

/\*$isTimeMonthly = RIODateTimeFactory::getDateTime();

$isTimeMonthly->setTime(0, 0);

$mandatoryTimeMonthly = RIODateTimeFactory::getDateTime();

$mandatoryTimeMonthly->setTime(0, 0);

$deviationTimeMonthly = RIODateTimeFactory::getDateTime();

$deviationTimeMonthly->setTime(0, 0);

foreach ($userMonth as $day) {

$dayTime = RIODateTimeFactory::getDateTime($day->offsetGet("presenceTime"));

$dayMandatoryTime = RIODateTimeFactory::getDateTime($day->offsetGet("mandatoryTime"));

$isTimeMonthly->add($startTime->diff($dayTime));

$mandatoryTimeMonthly->add($startTime->diff($dayMandatoryTime));

}

$deviationTimeMonthly->add($mandatoryTimeMonthly->diff($isTimeMonthly));\*/

$isTimeTotal = RIODateTimeFactory::getDateTime();

$isTimeTotal->setTime(0, 0);

$mandatoryTimeTotal = RIODateTimeFactory::getDateTime();

$mandatoryTimeTotal->setTime(0, 0);

$deviationTimeTotal = RIODateTimeFactory::getDateTime();

$deviationTimeTotal->setTime(0, 0);

foreach ($userTotal as $day) {

$dayTime = RIODateTimeFactory::getDateTime($day->offsetGet("presenceTime"));

$dayMandatoryTime = RIODateTimeFactory::getDateTime($day->offsetGet("mandatoryTime"));

$isTimeTotal->add($startTime->diff($dayTime));

$mandatoryTimeTotal->add($startTime->diff($dayMandatoryTime));

}

$deviationTimeTotal->add($mandatoryTimeTotal->diff($isTimeTotal));

/\*$isTimeWeekly = RIODateTimeFactory::getDateTime();

$isTimeWeekly->setTime(0, 0);

$mandatoryTimeWeekly = RIODateTimeFactory::getDateTime();

$mandatoryTimeWeekly->setTime(0, 0);

$deviationTimeWeekly = RIODateTimeFactory::getDateTime();

$deviationTimeWeekly->setTime(0, 0);

foreach ($userWeek as $day) {

$dayTime = RIODateTimeFactory::getDateTime($day->offsetGet("presenceTime"));

$dayMandatoryTime = RIODateTimeFactory::getDateTime($day->offsetGet("mandatoryTime"));

$isTimeWeekly->add($startTime->diff($dayTime));

$mandatoryTimeWeekly->add($startTime->diff($dayMandatoryTime));

}

$deviationTimeWeekly->add($mandatoryTimeWeekly->diff($isTimeWeekly));\*/

return [

"time" => $times,

'presenceTime' => $presenceTimeTotal->format("H:i"),

'presenceTimeCorrected' => '',

'deviation' => $deviation->format("H:i"),

'deviationNegativeOrPositiveOrZero' => $deviationNegativeOrPositiveOrZero,

'isTimeMonthly' => '',

'isTimeTotal' => $isTimeTotal->format("H:i"),

'isTimeWeekly' => '',

"deviationTimeMonthly" => '',

"deviationTimeTotal" => $deviationTimeTotal->format("H:i"),

"deviationTimeWeekly" => '',

"deviationTimeTotalCorrected" => ''

];

}

/\*\*

\* @throws Exception

\* @throws TransportExceptionInterface

\*/

private function createNewWorkDay(Collection $workDaysCollection, BSONDocument $user, bool $timeRecordStarted): void

{

$mandatoryTime = RIODateTimeFactory::getDateTime($user->offsetGet("mandatoryTime"));

$workDay = new RIOWorkDayObject();

$tomorrow = RIODateTimeFactory::getDateTime('tomorrow');

if($workDay->isRIOHoliday() || $workDay->isSunday() || $workDay->isSaturday()) {

$workDay->getAbsentAllDay()->setOfficialHoliday();

$workDay->getAbsentMorning()->setOfficialHoliday();

$workDay->getAbsentAfternoon()->setOfficialHoliday();

$mandatoryTime->setTime(0,0);

}

if(true === $timeRecordStarted) {

$times = [

[

'start' => '00:00',

'startCorrected' => '',

'comment' => '',

'lastEditedUser' => '',

'lastEditedDate' => '',

'lastEditedTime' => ''

]

];

} else {

$times = [];

}

$workDaysCollection->insertOne(

[

'date' => $tomorrow->format("d.m.Y"),

'monthYear' => $tomorrow->format("m.Y"),

'weekYear' => $tomorrow->format("W.Y"),

'month' => $tomorrow->format("m"),

'week' => $tomorrow->format("W"),

'mandatoryTime' => $mandatoryTime->format("H:i"),

'mandatoryTimeCorrected' => '',

'timeCredit' => '00:00',

'timeCreditCorrected' => '',

'absentAllDay' => $workDay->getAbsentAllDay()->getOption(),

'absentAfternoon' => $workDay->getAbsentAfternoon()->getOption(),

'absentMorning' => $workDay->getAbsentMorning()->getOption(),

'time' => $times

]

);

}

/\*\*

\* @throws Exception

\*/

private function calcMandatoryTimeMonthlyWeeklyAndTotal(Collection $workDaysCollection, BSONDocument $findOneWorkDay): array

{

$isFirstDayInMonth = false;

$isFirstDayInWeek = false;

/\*\* @var BSONDocument[] $findWorkDaysFromThisMonth \*/

$findWorkDaysFromThisMonth = $workDaysCollection->find(["month" => $this->dateTime->format("m")])->toArray();

$findWorkDaysFromThisWeek = $workDaysCollection->find(["week" => $this->dateTime->format("W")])->toArray();

// Check if there is past day in this month

if(count($findWorkDaysFromThisMonth) >= 2) {

// There is minimum one day before in this month

$mandatoryTimeMonthly = $findWorkDaysFromThisMonth[count($findWorkDaysFromThisMonth)-2]->offsetGet("mandatoryTimeMonthly");

} else {

// First day in this month

$mandatoryTimeMonthly = "00:00";

$isFirstDayInMonth = true;

}

$mandatoryTime = $findOneWorkDay->offsetGet('mandatoryTime');

$mandatoryTimeCorrected = $findOneWorkDay->offsetGet('mandatoryTimeCorrected');

$time = $this->stringTimeToIntArray($mandatoryTimeMonthly);

// Is last day a corrected mandatory time set

if('' !== $mandatoryTimeCorrected) {

$mandatoryTimeCorrectedDateTime = RIODateTimeFactory::getDateTime($mandatoryTimeCorrected);

$finalMandatoryTimeMonthly = $this->calculationOverTwentyfourHours($time, $mandatoryTimeCorrectedDateTime);

} else {

$mandatoryTimeDateTime = RIODateTimeFactory::getDateTime($mandatoryTime);

$finalMandatoryTimeMonthly = $this->calculationOverTwentyfourHours($time, $mandatoryTimeDateTime);

}

$finalMandatoryTimeWeekly = "";

$finalMandatoryTimeTotal = "";

return [

"mandatoryTimeMonthly" => $finalMandatoryTimeMonthly,

"mandatoryTimeWeekly" => $finalMandatoryTimeWeekly,

"mandatoryTimeTotal" => $finalMandatoryTimeTotal

];

}

/\*\*

\* @param string $stringTime

\* @return int[]

\*/

private function stringTimeToIntArray(string $stringTime): array

{

$array = explode(':', $stringTime);

return [

"hour" => $array[0],

"minute" => $array[1]

];

}

/\*\*

\* Only for monthly calculation usage

\*

\* @param array $time add of sub time

\* @param \DateTime $addDateTime initial, time can be positiv or negative

\* @param string $addition + or - time

\* @param string $addDateTimeAddition initial time is + or -

\* @param bool $returnOperator example true +08:45, false 08:45 or true -08:45, false 08:45

\* @return string

\* @throws Exception

\*/

private function calculationOverTwentyfourHours(array $time, DateTime $addDateTime, string $addition = "", string $addDateTimeAddition = "", bool $returnOperator = false): string

{

$operator = "";

$dateTime = RIODateTimeFactory::getDateTime();

if("+" === $addition || "" === $addition) {

$subtraction = false;

}

if("-" === $addition) {

$subtraction = true;

}

$firstDateTime = RIODateTimeFactory::getDateTime("00:00");

$newDateTime = RIODateTimeFactory::getDateTime();

$nowDayNumber = (int)$dateTime->format("d");

$nowYearNumber = (int)$dateTime->format("Y");

$firstDayNumber = (int)$addDateTime->format("d");

$firstYearNumber = (int)$addDateTime->format("Y");

$addNumberOfDays = $firstDayNumber - $nowDayNumber;

$addNumberOfYears = $firstYearNumber - $nowYearNumber;

$dateTime->setTime($time["hour"], $time["minute"]);

$afterDayNumber = (int)$dateTime->format("d");

$numberOfDays = $afterDayNumber - $nowDayNumber;

$hoursOfDays = $numberOfDays\*24;

$restHour = (int)$dateTime->format("H");

$totalHour = $hoursOfDays + $restHour;

$hourFinal = $totalHour <= 9 ? '0'.$totalHour : $totalHour;

$restMinute = (int)$dateTime->format("i");

$minuteFinal = $restMinute <= 9 ? '0'.$restMinute : $restMinute;

$newDateTime->setTime($hourFinal, $minuteFinal);

$interval = new DateInterval("P".$addNumberOfYears."Y".$addNumberOfDays."DT".$addDateTime->format("H")."H".$addDateTime->format("i")."M");

if(false === $subtraction) {

$newDateTime->add($interval);

} else {

if($addition === $addDateTimeAddition) {

$operator .= "-";

$newDateTime->add($interval);

} else {

$newDateTime->sub($interval);

}

}

$finalInterval = $firstDateTime->diff($newDateTime);

$hour = ($finalInterval->d\*24)+$finalInterval->h <= 9 ? '0'.($finalInterval->d\*24)+$finalInterval->h : ($finalInterval->d\*24)+$finalInterval->h;

$minute = $finalInterval->i <= 9 ? '0'.$finalInterval->i : $finalInterval->i;

if(true === $returnOperator) {

return $operator.$hour.':'.$minute;

}

return $hour.':'.$minute;

}

/\*\*

\* Every workday of an user is unique

\*

\* @param string $username

\* @return array

\* @throws Exception

\*/

private function getUserAllPastWorkdays(string $username): array

{

$currentWorkDay = new RIOWorkDayObject();

/\*\* @var BSONDocument $user \*/

$user = $this->mongoDB->getUsersCollection()->findOne(['sessionUsername' => $username]);

$allWorkDaysFromUser = $this->mongoDB->getWorkDaysCollection()->find(["sessionUsername" => $username])->toArray();

return $this->getPastWorkDaysUser($allWorkDaysFromUser, $currentWorkDay, $user);

}

/\*\*

\* @param array $allWorkDaysFromUser

\* @param \RIOWorkDayObject $currentWorkDay

\* @param BSONDocument $user

\* @param bool $pastWorkDay

\* @param bool $sortByDate

\* @return array

\* @throws Exception

\*/

private function getPastWorkDaysUser(array $allWorkDaysFromUser, RIOWorkDayObject $currentWorkDay, BSONDocument $user, bool $pastWorkDay = true, bool $sortByDate = true): array

{

$allWorkDaysFromUserPast = [];

$currentWorkDayString = $currentWorkDay->getDate()->format("d.m.Y");

foreach ($allWorkDaysFromUser as $oneWorkDayFromUser) {

if(true === $pastWorkDay) {

if($oneWorkDayFromUser->offsetGet("date") !== $currentWorkDayString) {

$allWorkDaysFromUserPast[] = $oneWorkDayFromUser;

}

} else {

$allWorkDaysFromUserPast[] = $oneWorkDayFromUser;

}

}

if(true === $sortByDate) {

usort($allWorkDaysFromUserPast, function($a, $b) {

return RIODateTimeFactory::getDateTime($a['date']) <=> RIODateTimeFactory::getDateTime($b['date']);

});

}

$oneWorkDayFromUserPastIndexed = [];

$i = 0;

foreach ($allWorkDaysFromUserPast as $oneWorkDayFromUserPast) {

$oneWorkDayFromUserPast["presenceTimeCorrections"] = [$user->offsetGet("sessionUsername"), $oneWorkDayFromUserPast["date"]];

$oneWorkDayFromUserPastIndexed[] = $oneWorkDayFromUserPast;

$i++;

}

return $oneWorkDayFromUserPastIndexed;

}

private function calcIsTimeMonthly(Collection $workDaysCollection, BSONDocument $findOneWorkDay): string

{

$isFirstDayInMonth = false;

/\*\* @var BSONDocument[] $findWorkDaysFromThisMonth \*/

$findWorkDaysFromThisMonth = $workDaysCollection->find(["month" => $this->dateTime->format("m")])->toArray();

// Check if there is past day in this month

if(count($findWorkDaysFromThisMonth) >= 2) {

// There is minimum one day before in this month

$isTimeMonthly = $findWorkDaysFromThisMonth[count($findWorkDaysFromThisMonth)-2]->offsetGet("isTimeMonthly");

} else {

// First day in this month

$isTimeMonthly = "00:00";

$isFirstDayInMonth = true;

}

$presenceTime = $findOneWorkDay->offsetGet("presenceTime");

$presenceTimeCorrected = $findOneWorkDay->offsetGet("presenceTimeCorrected");

$time = $this->stringTimeToIntArray($isTimeMonthly);

if('' !== $presenceTimeCorrected) {

$presenceTimeCorrectedDateTime = RIODateTimeFactory::getDateTime($presenceTimeCorrected);

$finalPresenceTimeMonthly = $this->calculationOverTwentyfourHours($time, $presenceTimeCorrectedDateTime);

} else {

$presenceTimeDateTime = RIODateTimeFactory::getDateTime($presenceTime);

$finalPresenceTimeMonthly = $this->calculationOverTwentyfourHours($time, $presenceTimeDateTime);

}

return $finalPresenceTimeMonthly;

}

private function calcDeviationTimeMonthly(string $isTimeMonthly, mixed $mandatoryTimeMonthly): string

{

$positiveNegative = "";

$isTimeMonthlyDateTime = RIODateTimeFactory::getDateTime();

$isTimeMonthlyArray = $this->stringTimeToIntArray($isTimeMonthly);

$isTimeMonthlyDateTime->setTime($isTimeMonthlyArray["hour"], $isTimeMonthlyArray["minute"]);

$mandatoryTimeMonthlyDateTime = RIODateTimeFactory::getDateTime();

$mandatoryTimeMonthlyArray = $this->stringTimeToIntArray($mandatoryTimeMonthly);

$mandatoryTimeMonthlyDateTime->setTime($mandatoryTimeMonthlyArray["hour"], $mandatoryTimeMonthlyArray["minute"]);

if($isTimeMonthlyDateTime > $mandatoryTimeMonthlyDateTime) {

$positiveNegative .= "+";

} else {

$positiveNegative .= "-";

}

$h = $mandatoryTimeMonthlyDateTime->diff($isTimeMonthlyDateTime)->h;

$m = $mandatoryTimeMonthlyDateTime->diff($isTimeMonthlyDateTime)->i;

$hourFinal = $h <= 9 ? '0'.$h : $h;

$minuteFinal = $m <= 9 ? '0'.$m : $m;

return $positiveNegative.$hourFinal.':'.$minuteFinal;

}

/\*\*

\* @throws Exception

\*/

private function calcDeviationTimeTotal(BSONDocument $findOneWorkDay, BSONDocument $yesterdayWorkDay): string

{

$additionDeviationTimeTotal = $yesterdayWorkDay->offsetGet("deviationTimeTotal");

$addition = substr($additionDeviationTimeTotal,0,1);

$deviationTimeTotal = substr($additionDeviationTimeTotal,1);

$deviationTimeTotalArray = $this->stringTimeToIntArray($deviationTimeTotal);

$addTime = $this->stringTimeToIntArray($findOneWorkDay->offsetGet("deviation"));

$addTimeAddition = $findOneWorkDay->offsetGet("deviationNegativeOrPositiveOrZero");

$addDateTime = RIODateTimeFactory::getDateTime();

$addDateTime->setTime($addTime["hour"],$addTime["minute"]);

return $this->calculationOverTwentyfourHours($deviationTimeTotalArray, $addDateTime, $addTimeAddition);

}

/\*\*

\* If yesterday and today are in same week, then add them together else start at 00:00

\*

\* @param BSONDocument $findOneWorkDay Today

\* @param BSONDocument $yesterdayWorkDay Yesterday

\* @return string

\* @throws Exception

\*/

private function calcDeviationTimeWeekly(BSONDocument $findOneWorkDay, BSONDocument $yesterdayWorkDay): string

{

$deviation = $findOneWorkDay->offsetGet("deviation");

$addTime = $this->stringTimeToIntArray($deviation);

$addTimeAddition = $findOneWorkDay->offsetGet("deviationNegativeOrPositiveOrZero");

if($findOneWorkDay->offsetGet("week") === $yesterdayWorkDay->offsetGet("week")) {

$additionDeviationTimeWeekly = $yesterdayWorkDay->offsetGet("deviationTimeWeekly");

$addition = substr($additionDeviationTimeWeekly,0,1);

$deviationTimeWeekly = substr($additionDeviationTimeWeekly,1);

$deviationTimeWeeklyArray = $this->stringTimeToIntArray($deviationTimeWeekly);

$addDateTime = RIODateTimeFactory::getDateTime();

$addDateTime->setTime($addTime["hour"],$addTime["minute"]);

return $this->calculationOverTwentyfourHours($deviationTimeWeeklyArray, $addDateTime, $addTimeAddition, $addition, true);

} else {

return $addTimeAddition.$deviation;

}

}

}

**source/classes/RIOCustomTwigExtension.php**

<?php

use Symfony\Component\HttpFoundation\Request;

use Twig\Extension\AbstractExtension;

use Twig\TwigFunction;

use function source\getAbsolutePath;

class RIOCustomTwigExtension extends AbstractExtension

{

private Request $request;

public function \_\_construct(Request $request) {

$this->request = $request;

}

public function getFunctions(): array

{

return [

new TwigFunction('getAbsolutePath', [$this, 'getAbsolutePath']),

new TwigFunction('getSession', [$this, 'getSession']),

new TwigFunction('getWeekDayShortNameByDate', [$this, 'getWeekDayShortNameByDate']),

new TwigFunction('isLoggedIn', [$this, 'isLoggedIn']),

new TwigFunction('logoLink', [$this, 'logoLink'])

];

}

/\*\*

\* @param string $sessionUsername

\* @param string $monthYear

\* @param string $active

\* @return array

\*/

public function navByActive(string $sessionUsername, string $monthYear, string $active = ''): array

{

return [

"nav" => [

[

"name" => "Zeiterfassung",

"active" => "user\_home" === $active,

"link" => $this->getAbsolutePath(["rioadmin","sessionLogin"]),

"mobile" => true

],

[

"name" => "Benutzer",

"active" => "edit\_user" === $active,

"link" => $this->getAbsolutePath(["rioadmin","editUser", $sessionUsername, "unchanged"]),

"mobile" => true

],

[

"name" => "Übersicht",

"active" => "overview" === $active,

"link" => $this->getAbsolutePath(["rioadmin","overview", $sessionUsername, $monthYear]),

"mobile" => false

]

]

];

}

public function logoLink(): string

{

if($this->isLoggedIn()) {

return $this->getAbsolutePath(["sessionLogin"]);

}

return $this->getAbsolutePath();

}

public function isLoggedIn(): bool

{

$databaseCollection = new RIOMongoDatabaseCollection(RIOMongoDatabase::getInstance()->getDatabase(), "user");

$collection = $databaseCollection->getCollection();

$session = $this->request->getSession();

$userFind = $collection->findOne(

["sessionId" => $session->getId()]

);

if(null !== $userFind) {

$metadataBag = $session->getMetadataBag();

if($metadataBag->getLifetime() >= $metadataBag->getLastUsed() - $metadataBag->getCreated()) {

return true;

} else {

$collection->updateOne(

[ "sessionId" => $session->getId() ],

[

'$set' => [ 'sessionId' => '' ]

]

);

$this->request->getSession()->invalidate();

}

}

return false;

}

/\*\*

\* @param string $date

\* @return string

\* @throws Exception

\*/

public function getWeekDayShortNameByDate(string $date): string

{

$givenDate = RIODateTimeFactory::getDateTime($date);

$dayNames = [

'Mo',

'Di',

'Mi',

'Do',

'Fr',

'Sa',

'So'

];

return $dayNames[(int) $givenDate->format('N')-1];

}

public function getSession(string $session\_key = null): ?string

{

return $this->request->getSession()->get($session\_key);

}

/\*\*

\* @param string $after

\* @param array $parts

\* @return string

\*/

public function getAbsolutePath(array $parts = [], string $after = ""): string

{

return getAbsolutePath($parts, $after);

}

}

**source/classes/RIODateTimeFactory.php**

<?php

declare(strict\_types=1);

class RIODateTimeFactory

{

/\*\*

\* @param string $datetime

\* @param DateTimeZone|null $timezone

\* @return DateTime

\* @throws Exception

\*/

static function getDateTime(string $datetime = 'now', DateTimeZone $timezone = null): DateTime

{

return new DateTime($datetime, (null === $timezone) ? new DateTimeZone("Europe/Zurich") : $timezone);

}

}

**source/classes/RIOFrontendResolver.php**

<?php

declare(strict\_types=1);

class RIOFrontendResolver extends RIOPartialResolver

{

private string $default\_frontend;

/\*\*

\* @var string[]

\*/

private array $frontends;

/\*\*

\* @param string[] $frontends

\*/

public function \_\_construct(string $default\_frontend, array $frontends)

{

$this->default\_frontend = $default\_frontend;

$this->frontends = $frontends;

}

public function resolve(RIOResolvedAction $action): RIOResolvedAction

{

$possible\_frontend = RIOMaybe::ofSettable($action->getPathPartials()[0]);

if (!$possible\_frontend->isEmpty()) {

foreach ($this->frontends as $frontend) {

if ($frontend == $possible\_frontend->getValue()) {

// The frontend exists

$action->removeFirstArrayElement();

$action->setFrontend($possible\_frontend);

return $action;

}

}

}

// The first part of the parts isn't the area name

$action->setFrontend(RIOMaybe::of($this->default\_frontend));

return $action;

}

}

**source/classes/RIOGeneralAccessController.php**

<?php

use MongoDB\Collection;

use MongoDB\Database;

use MongoDB\Model\BSONDocument;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Session\SessionInterface;

use Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface;

use Twig\Environment;

class RIOGeneralAccessController

{

private Environment $twig;

private Request $request;

private string $directoryNamespace;

private RIOMongoDatabase $mongoDatabase;

private DateTime $dateTime;

public function \_\_construct(string $directoryNamespace, Environment $twig, Request $request)

{

$this->twig = $twig;

$this->request = $request;

$this->directoryNamespace = $directoryNamespace;

$this->mongoDatabase = RIOMongoDatabase::getInstance();

$this->dateTime = RIODateTimeFactory::getDateTime();

}

/\*\*

\* @return DateTime

\*/

public function getDateTime(): DateTime

{

return $this->dateTime;

}

public function getSession(): SessionInterface

{

return $this->getRequest()->getSession();

}

/\*\*

\* Current session user with current timestamp

\*

\* @return array

\* @throws Exception

\* @throws TransportExceptionInterface

\*/

public function getTimeRecording(): array

{

$mandatoryTime = RIODateTimeFactory::getDateTime($this->getUser()["mandatoryTime"]);

$workDay = new RIOWorkDayObject();

if($workDay->isRIOHoliday() || $workDay->isSunday() || $workDay->isSaturday()) {

$workDay->getAbsentAllDay()->setOfficialHoliday();

$workDay->getAbsentMorning()->setOfficialHoliday();

$workDay->getAbsentAfternoon()->setOfficialHoliday();

$mandatoryTime->setTime(0,0);

}

return [

'date' => $this->dateTime->format("d.m.Y"),

'monthYear' => $this->dateTime->format("m.Y"),

'weekYear' => $this->dateTime->format("W.Y"),

'month' => $this->dateTime->format("m"),

'week' => $this->dateTime->format("W"),

$this->getMandatoryTimeKey() => $mandatoryTime->format("H:i"),

$this->getMandatoryTimeCorrectedKey() => '',

$this->getTimeCreditKey() => '00:00',

$this->getTimeCreditCorrectedKey() => '',

'absentAllDay' => $workDay->getAbsentAllDay()->getOption(),

'absentAfternoon' => $workDay->getAbsentAfternoon()->getOption(),

'absentMorning' => $workDay->getAbsentMorning()->getOption(),

'time' => [

[

'start' => $this->dateTime->format("H:i"),

'startCorrected' => '',

'comment' => '',

'lastEditedUser' => '',

'lastEditedDate' => '',

'lastEditedTime' => ''

]

]

];

}

public function getTimeRecordingFilter(): array

{

return [

'date' => $this->dateTime->format("d.m.Y")

];

}

public function getTimeRecordingFilterStarted(): array

{

return [

'date' => $this->dateTime->format("d.m.Y")

];

}

public function getTimeRecordingStart(): BSONDocument

{

return new BSONDocument([

'start' => $this->dateTime->format("H:i"),

'startCorrected' => '',

'comment' => '',

'lastEditedUser' => '',

'lastEditedDate' => '',

'lastEditedTime' => ''

]);

}

public function getTimeRecordingEnd(): array

{

return [

$this->getTimeRecordingEndKey() => $this->getTimeRecordingEndValue()

];

}

public function getAbsentAllDayKey(): string

{

return "absentAllDay";

}

public function getAbsentMorningKey(): string

{

return "absentMorning";

}

public function getAbsentAfternoonKey(): string

{

return "absentAfternoon";

}

public function getTimeRecordingStartKey(): string

{

return 'start';

}

public function getTimeRecordingStartCorrectedKey(): string

{

return 'startCorrected';

}

public function getTimeRecordingEndKey(): string

{

return 'end';

}

public function getTimeRecordingEndCorrectedKey(): string

{

return 'endCorrected';

}

public function getPresenceTimeKey(): string

{

return 'presenceTime';

}

public function getTimeCreditKey(): string

{

return "timeCredit";

}

public function getTimeCreditCorrectedKey(): string

{

return "timeCreditCorrected";

}

public function getIsTimeKey(): string

{

return 'isTime';

}

public function getDiffKey(): string

{

return 'diff';

}

public function getDiffNegativePositiveKey(): string

{

return 'diffNegativePositive';

}

public function getMandatoryTimeKey(): string

{

return 'mandatoryTime';

}

public function getMandatoryTimeCorrectedKey(): string

{

return 'mandatoryTimeCorrected';

}

public function getMandatoryTimeMonthlyKey(): string

{

return "mandatoryTimeMonthly";

}

public function getMandatoryTimeTotalKey(): string

{

return "mandatoryTimeTotal";

}

public function getMandatoryTimeWeeklyKey(): string

{

return "mandatoryTimeWeekly";

}

public function getDeviationTimeMonthlyKey(): string

{

return "deviationTimeMonthly";

}

public function getDeviationTimeTotalKey(): string

{

return "deviationTimeTotal";

}

public function getDeviationTimeWeeklyKey(): string

{

return "deviationTimeWeekly";

}

public function getDeviationTimeTotalCorrectedKey(): string

{

return "deviationTimeTotalCorrected";

}

public function getWorkingTimePerformedKey(): string

{

return 'workingTimePerformed';

}

public function getWorkingTimePerformedCorrectedKey(): string

{

return 'workingTimePerformedCorrected';

}

public function getPresenceTimeCorrectedKey(): string

{

return 'presenceTimeCorrected';

}

public function getPresenceTimeTotalCorrectedKey(): string

{

return "presenceTimeTotalCorrected";

}

public function getTimeRecordingEndValue(): string

{

return $this->dateTime->format("H:i");

}

public function getDate(): array

{

return [

'date' => $this->dateTime->format("d.m.Y")

];

}

public function getLastMonth(): array

{

$month = ((int)$this->dateTime->format("m"))-1;

$monthString = (string)$month;

$monthFinal = $month <= 9 ? '0'.$monthString : $monthString;

return [

"month" => $monthFinal

];

}

public function getMonth(): array

{

return [

"month" => $this->dateTime->format("m")

];

}

/\*\*

\* Every workday of an user is unique

\*

\* @param RIOMain|RIOAdmin $controller

\* @return array

\* @throws Exception

\*/

public function getUserAllPastWorkdays(RIOMain|RIOAdmin $controller): array

{

$user = new RIOUserObject($controller);

$currentWorkDay = new RIOWorkDayObject();

$allWorkDaysFromUser = $this->getWorkDays()->find(["sessionUsername" => $user->getUsername()])->toArray();

return $this->getPastWorkDays($allWorkDaysFromUser, $currentWorkDay, $user);

}

/\*\*

\* Month of a year is unique

\*

\* @param string $monthYear

\* @param string $username

\* @return array

\* @throws Exception

\*/

public function getUserAllPastWorkdaysByMonthYearUser(string $monthYear, string $username): array

{

/\*\* @var BSONDocument $user \*/

$user = $this->getUsers()->findOne(['sessionUsername' => $username]);

$currentWorkDay = new RIOWorkDayObject();

/\*\* @var BSONDocument[]|[] $allWorkDaysFromUser \*/

$allWorkDaysFromUser = $this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime("01.".$monthYear)->format("Y"),$username)->find(["monthYear" => $monthYear])->toArray();

return $this->getPastWorkDaysUser($allWorkDaysFromUser, $currentWorkDay, $user);

}

/\*\*

\* Month of a year is unique

\*

\* @param string $monthYear

\* @param RIOMain|RIOAdmin $controller

\* @return array

\* @throws Exception

\*/

public function getUserAllPastWorkdaysByMonthYear(string $monthYear, RIOMain|RIOAdmin $controller): array

{

$user = new RIOUserObject($controller);

$currentWorkDay = new RIOWorkDayObject();

$allWorkDaysFromUser = $this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime("01.".$monthYear)->format("Y"),$user->getUsername())->find(["sessionUsername" => $user->getUsername(), "monthYear" => $monthYear])->toArray();

return $this->getPastWorkDays($allWorkDaysFromUser, $currentWorkDay, $user);

}

/\*\*

\* Week of a year is unique

\*

\* @param string $weekYear

\* @param RIOMain|RIOAdmin $controller

\* @return array

\* @throws Exception

\*/

public function getUserAllPastWorkdaysByWeek(string $weekYear, RIOMain|RIOAdmin $controller): array

{

$user = new RIOUserObject($controller);

$currentWorkDay = new RIOWorkDayObject();

$allWorkDaysFromUser = $this->getWorkDaysByYearUser(RIODateTimeFactory::getDateTime("01.01.".explode('.',$weekYear)[1])->format("Y"),$user->getUsername())->find(["sessionUsername" => $user->getUsername(), "weekYear" => $weekYear])->toArray();

return $this->getPastWorkDays($allWorkDaysFromUser, $currentWorkDay, $user);

}

/\*\*

\* @param BSONDocument[] $allWorkDaysFromUser

\* @param RIOWorkDayObject $currentWorkDay

\* @param BSONDocument $user

\* @param bool $pastWorkDay

\* @param bool $sortByDate

\* @return array

\* @throws Exception

\*/

private function getPastWorkDaysUser(array $allWorkDaysFromUser, RIOWorkDayObject $currentWorkDay, BSONDocument $user, bool $pastWorkDay = true, bool $sortByDate = true): array

{

$allWorkDaysFromUserPast = [];

$currentWorkDayString = $currentWorkDay->getDate()->format("d.m.Y");

foreach ($allWorkDaysFromUser as $oneWorkDayFromUser) {

if(true === $pastWorkDay) {

if($oneWorkDayFromUser->offsetGet("date") !== $currentWorkDayString) {

$allWorkDaysFromUserPast[] = $oneWorkDayFromUser;

}

} else {

$allWorkDaysFromUserPast[] = $oneWorkDayFromUser;

}

}

if(true === $sortByDate) {

usort($allWorkDaysFromUserPast, function($a, $b) {

return RIODateTimeFactory::getDateTime($a['date']) <=> RIODateTimeFactory::getDateTime($b['date']);

});

}

$oneWorkDayFromUserPastIndexed = [];

$i = 0;

foreach ($allWorkDaysFromUserPast as $oneWorkDayFromUserPast) {

$oneWorkDayFromUserPast["presenceTimeCorrections"] = [$user->offsetGet("sessionUsername"), $oneWorkDayFromUserPast["date"]];

$oneWorkDayFromUserPastIndexed[] = $oneWorkDayFromUserPast;

$i++;

}

return $oneWorkDayFromUserPastIndexed;

}

private function getPastWorkDays(array $allWorkDaysFromUser, RIOWorkDayObject $currentWorkDay, RIOUserObject $user, bool $pastWorkDay = true, bool $sortByDate = true): array

{

$allWorkDaysFromUserPast = [];

$currentWorkDayString = $currentWorkDay->getDate()->format("d.m.Y");

foreach ($allWorkDaysFromUser as $oneWorkDayFromUser) {

if(true === $pastWorkDay) {

if($oneWorkDayFromUser->offsetGet("date") !== $currentWorkDayString) {

$allWorkDaysFromUserPast[] = $oneWorkDayFromUser;

}

} else {

$allWorkDaysFromUserPast[] = $oneWorkDayFromUser;

}

}

if(true === $sortByDate) {

usort($allWorkDaysFromUserPast, function($a, $b) {

return RIODateTimeFactory::getDateTime($a['date']) <=> RIODateTimeFactory::getDateTime($b['date']);

});

}

$oneWorkDayFromUserPastIndexed = [];

$i = 0;

foreach ($allWorkDaysFromUserPast as $oneWorkDayFromUserPast) {

$OneWorkDayFromUserPast["presenceTimeCorrections"] = [$user->getUsername(), $oneWorkDayFromUserPast["date"]];

$oneWorkDayFromUserPastIndexed[] = $oneWorkDayFromUserPast;

$i++;

}

return $oneWorkDayFromUserPastIndexed;

}

/\*\*

\* Current session user with session id

\* @return array

\*/

public function getUser(): array

{

$username = $this->getSession()->get("username");

$sessionId = $this->getSession()->getId();

$timeRecordStarted = false;

$findUser = $this->getUsers()->findOne([ "sessionId" => $sessionId ]);

if(null === $username) {

if(null !== $findUser) {

if($findUser->offsetExists("sessionUsername")) {

$username = $findUser->offsetGet("sessionUsername");

} else {

$username = '';

}

} else {

$username = '';

}

}

if(null !== $findUser){

$timeRecordStarted = [

'timeRecordStarted' => $this->getUsers()->findOne(

[ "sessionUsername" => $username ]

)["timeRecordStarted"]

];

$timeRecordStarted = $timeRecordStarted["timeRecordStarted"];

}

$user = [

'sessionUsername' => $username,

'sessionId' => null === $sessionId ? '' : $sessionId,

'timeRecordStarted' => $timeRecordStarted,

];

if(null !== $findUser) {

$user = array\_merge(

$user,

[

'mandatoryTime' => $findUser["mandatoryTime"]

]

);

}

return $user;

}

/\*\*

\* @return array

\*/

public function getUserFromSession(): array

{

$sessionId = $this->getSession()->getId();

return [

'sessionId' => null === $sessionId ? '' : $sessionId

];

}

public function getUsers(): Collection

{

return $this->getMongoDatabase()->getUsersCollection();

}

public function getWorkDays(): Collection

{

return $this->getMongoDatabase()->getWorkDaysCollection();

}

/\*\*

\* @return Collection[]

\* @throws Exception

\*/

public function getWorkDaysFromUser(string $username): array

{

$currentYear = RIODateTimeFactory::getDateTime()->format("Y");

$collections = [];

for($startYear = $\_ENV["LAUNCH\_YEAR"]; $startYear <= $currentYear; $startYear++) {

if($this->workDaysByYearUserExists($startYear, $username)) {

$collections[] = $this->getWorkDaysByYearUser($startYear, $username);

}

}

return $collections;

}

public function workDaysByYearUserExists(string $year, string $username): bool

{

return 0 !== $this->getMongoDatabase()->getWorkDaysCollectionByYearUser($year, $username)->countDocuments();

}

public function getWorkDaysByYearUser(string $year, string $username): Collection

{

return $this->getMongoDatabase()->getWorkDaysCollectionByYearUser($year, $username);

}

/\*\*

\* @throws Exception

\*/

public function getWorkDaysByUser(): array

{

$collections = $this->getWorkDaysFromUser($this->getUser()['sessionUsername']);

$collectionsArray = [];

foreach ($collections as $collection) {

$collectionsArray[] = $collection->find(

[

"sessionUsername" => $this->getUser()['sessionUsername']

]

)->toArray();

}

return $collectionsArray;

}

final protected function twig(): Environment

{

return $this->twig;

}

public function getDirectoryNamespaceString(): string

{

if(isset($this->directoryNamespace)) {

return $this->directoryNamespace;

}

return "";

}

final protected function getRequest(): Request

{

return $this->request;

}

/\*\*

\* @return RIOMongoDatabase

\*/

public function getMongoDatabase(): RIOMongoDatabase

{

return $this->mongoDatabase;

}

public function getDatabase(): Database

{

return $this->getMongoDatabase()->getDatabase();

}

public function selectCollection($collectionName): Collection

{

return $this->mongoDatabase->getDatabase()->selectCollection($collectionName);

}

}

**source/classes/RIOMain.php**

<?php

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

use Twig\Environment;

use function source\getAbsolutePath;

/\*\*

\* Class RIOMain

\* User can be logged in or out state

\* This area RIOMain should be used for editing user by it's user session

\* Like user start and stop time record or login and logout user

\*/

class RIOMain extends RIOAccessController

{

public function \_\_construct(

string $directoryNamespace,

Environment $twig,

Request $request

) {

parent::\_\_construct($directoryNamespace, $twig, $request);

}

/\*\*

\* show home or do auto login if already an user is logged in

\*

\* @return Response

\*/

public function showHomepage(): Response

{

return RIORedirect::redirectResponse(["login", "unchanged"]);

}

public function login(string $state): Response

{

if(null !== $state) {

$stateArray = ['state' => $state];

} else {

$stateArray = [];

}

return $this->renderPage(

"home.twig",

array\_merge(

[

'action' => getAbsolutePath(["postlogin"])

],

$stateArray

)

);

}

/\*\*

\* Tries to login user by current session if saved

\*

\* @return Response

\*/

public function sessionLogin(): Response

{

$customTwigExtension = new RIOCustomTwigExtension($this->getRequest());

if($customTwigExtension->isLoggedIn()) {

return RIORedirect::redirectResponse(["rioadmin", "sessionLogin"]);

}

return RIORedirect::redirectResponse(["login", "unchanged"]);

}

/\*\*

\* Check if given user and password exists in LDAP

\* create new MongoDB user if not exists or just insert new session id

\*

\* @param string $username

\* @param string $password

\* @return Response|null

\* @throws \Exception

\*/

private function userValidate(string $username, string $password): ?Response

{

/\*\* @var resource $ldap \*/

$ldap = ldap\_connect($\_ENV["LDAP\_HOST"], $\_ENV["LDAP\_PORT"]);

ldap\_set\_option($ldap, LDAP\_OPT\_PROTOCOL\_VERSION, 3);

// TODO: uncatchable connection error on follow line below

$search = ldap\_search($ldap, $\_ENV["LDAP\_SEARCH\_ROOT"], '(' . $\_ENV["LDAP\_RDN"] . '=' . $username . ')');

$results = ldap\_get\_entries($ldap, $search);

$dn = $results[0]['dn'];

$displayUsername = $results[0]['uid'][0];

$sessionUsername = $results[0]['uid'][1];

$surnameUsername = $results[0]["sn"][0];

$session = $this->getSession();

$sessionId = $session->getId();

$maybeObject = [

'sessionUsername' => $sessionUsername,

'displayUsername' => $displayUsername,

'surnameUsername' => $surnameUsername

];

$maybeAuthObject = array\_merge(

$maybeObject,

['sessionId' => $sessionId]

);

$user = new RIOUserObject();

$authObjectNoTime = array\_merge(

$maybeAuthObject,

[

// New created user has by default no time record

'timeRecordStarted' => false,

"mandatoryTime" => $user->getMandatoryTime()->format("H:i"),

"location" => $user->getLocation()

]

);

$auth\_find = $this->getUsers()->findOne(

$maybeObject

);

try {

$bind = ldap\_bind($ldap, $dn, $password);

} catch (Exception $e) {

if(RIOConfig::isInDebugMode()) {

throw new Exception($e->getMessage(). ", dn = ".$dn.", password = ".$password);

} else {

// Username was correct but password was wrong

return RIORedirect::redirectResponse(["login", "failure"]);

}

}

if ($bind) {

ldap\_unbind($ldap);

if("0" === (string)$this->getSession()->getMetadataBag()->getLifetime()) {

$this->getSession()->getMetadataBag()->stampNew($\_ENV["SESSION\_LIFE\_TIME"]);

}

if(null === $auth\_find) {

$this->getUsers()->insertOne(

$authObjectNoTime

);

} else {

$this->getUsers()->updateOne(

$maybeObject,

[

// Update new sessionId from client

'$set' => [ 'sessionId' => $sessionId ]

]

);

}

return RIORedirect::redirectResponse(["rioadmin", "sessionLogin"]);

} else {

// Username or password was wrong

return RIORedirect::redirectResponse(["login", "failure"]);

}

}

/\*\*

\* Tries to login user by post, usually called by a form

\*

\* @return Response

\* @throws Exception

\*/

public function postLogin(): Response

{

$request = $this->getRequest();

$usernamePost = $request->get("username");

$passwordPost = $request->get("password");

if (null !== $usernamePost && null !== $passwordPost) {

$request->getSession()->set("username", $usernamePost);

return $this->userValidate($usernamePost, $passwordPost);

}

return RIORedirect::error(500);

}

}

**source/classes/RIOMaybe.php**

<?php

declare(strict\_types=1);

class RIOMaybe

{

private mixed $value;

private static ?RIOMaybe $empty = null;

protected function \_\_construct($value)

{

$this->value = $value;

$this->checkIfValidState();

}

public static function getEmpty(): RIOMaybe

{

if (null === self::$empty) {

self::$empty = new class() extends RIOMaybe {

public function \_\_construct()

{

parent::\_\_construct('EmptyRIOMaybe');

}

public function getValue(): void

{

if(RIOConfig::isInDebugMode()) {

throw new Error('Tried to getValue from empty RIOMaybe!');

}

}

};

}

return self::$empty;

}

public function getValue()

{

$this->checkIfValidState();

return $this->value;

}

public function isEmpty(): bool

{

return $this === self::$empty;

}

/\*\*

\* @param $value

\* @return RIOMaybe

\*/

public static function ofSettable(&$value): RIOMaybe

{

if (isset($value) && !empty($value)) {

return self::of($value);

}

return self::getEmpty();

}

public static function of($value): RIOMaybe

{

return new RIOMaybe($value);

}

private function checkIfValidState(): void

{

if(RIOConfig::isInDebugMode()) {

if (!isset($this->value)) {

throw new Error('RIOMaybe state is invalid');

}

}

}

}

**source/classes/RIOMongoDatabase.php**

<?php

declare(strict\_types=1);

use MongoDB\Client;

use MongoDB\Collection;

use MongoDB\Database;

use MongoDB\Driver\Cursor;

use MongoDB\InsertManyResult;

use MongoDB\InsertOneResult;

use MongoDB\Model\BSONDocument;

use MongoDB\UpdateResult;

use Symfony\Component\HttpFoundation\RedirectResponse;

/\*\*

\* Manage like a facade a single database with any number of collection/s

\*

\* Class RIOMongoDatabase

\*/

class RIOMongoDatabase

{

/\*\*

\* instance

\*

\* @var null|RIOMongoDatabase

\*/

protected static ?RIOMongoDatabase $\_instance = null;

private Client $client;

/\*\*

\* Singleton

\* RIOMongoDatabase constructor.

\*/

private function \_\_construct()

{

$this->client = $this->getClient();

}

protected function \_\_clone(): void {}

public static function getInstance(): self

{

if (null === self::$\_instance)

{

self::$\_instance = new self;

}

return self::$\_instance;

}

public function getName(): string

{

return $\_ENV["DB\_NAME"];

}

private function getClient(): Client|RedirectResponse

{

if(!isset($this->client)) {

$client = new Client($\_ENV["MONGODB"]);

try {

$databaseInfoIterator = $client->listDatabases();

} catch (RIOConnectionFailed $connectionFailed) {

if (RIOConfig::isInDebugMode()) {

throw new Error("The database connection could not be established.", 0, $connectionFailed);

} else {

return RIORedirect::error(503);

}

}

return $client;

}

return $this->client;

}

public function getDatabase(): Database

{

return $this->client->{$this->getName()};

}

public function getCollection(string $collection): Collection

{

return (new RIOMongoDatabaseCollection($this->getDatabase(), $collection))->getCollection();

}

public function createCollection(string $collection, array $options = []): array|object

{

return $this->getDatabase()->createCollection($collection, $options);

}

public function find(string $collection, $filter = [], array $options = []): Cursor

{

return $this->getCollection($collection)->find($filter, $options);

}

public function insertMany(string $collection, array $documents, array $options = []): InsertManyResult

{

return $this->getCollection($collection)->insertMany($documents, $options);

}

public function insertOne(string $collection, mixed $document, array $options = []): InsertOneResult

{

return $this->getCollection($collection)->insertOne($document, $options);

}

public function updateMany(string $collection, mixed $filter, mixed $update, array $options = []): UpdateResult

{

return $this->getCollection($collection)->updateMany($filter, $update, $options);

}

public function updateOne(string $collection, mixed $filter, mixed $update, array $options = []): UpdateResult

{

return $this->getCollection($collection)->updateOne($filter, $update, $options);

}

/\*\*

\* @return BSONDocument[]

\*/

public function getActiveUsers(): array

{

return $this->getUsers(["time\_record\_started" => true]);

}

/\*\*

\* @return BSONDocument[]

\*/

public function getInactiveUsers(): array

{

return $this->getUsers(["time\_record\_started" => false]);

}

/\*\*

\* @param array $filter

\* @param array $options

\* @return BSONDocument[]

\*/

public function getUsers(array $filter = [], array $options = []): array

{

return $this->getUsersCollection()->find($filter)->toArray();

}

public function getWorkDaysCollectionByYearUser(string $year, string $username): Collection

{

return $this->getDatabase()->selectCollection($this->getWorkDaysCollectionByYearUserAsString($year,$username));

}

public function getWorkDaysCollectionByYearUserAsString(string $year, string $username): string

{

return "work\_day\_".$year.'\_'.$username;

}

public function getUsersCollection(): Collection

{

return $this->getDatabase()->selectCollection("user");

}

public function getWorkDaysCollection(): Collection

{

return $this->getDatabase()->selectCollection("work\_day");

}

}

**source/classes/RIOMongoDatabaseCollection.php**

<?php

declare(strict\_types=1);

use MongoDB\Collection;

use MongoDB\Driver\Cursor;

use MongoDB\InsertManyResult;

use MongoDB\InsertOneResult;

use MongoDB\UpdateResult;

use MongoDB\Database;

class RIOMongoDatabaseCollection

{

private Database $database;

private string $collection;

/\*\*

\* RIOMongoDatabaseCollection constructor.

\* @param Database $database

\* @param string $collection

\*/

public function \_\_construct(Database $database, string $collection)

{

$this->database = $database;

$this->collection = $collection;

}

public function getCollection(): Collection

{

return $this->database->{$this->getName()};

}

public function getName(): string

{

return $this->collection;

}

public function find($filter = [], array $options = []): Cursor

{

return $this->getCollection()->find($filter, $options);

}

public function insertMany(array $documents, array $options = []): InsertManyResult

{

return $this->getCollection()->insertMany($documents, $options);

}

public function insertOne(array|object $document, array $options = []): InsertOneResult

{

return $$this->getCollection()->insertOne($document, $options);

}

public function updateMany(array|object $filter, array|object $update, array $options = []): UpdateResult

{

return $this->getCollection()->updateMany($filter, $update, $options);

}

public function updateOne(array|object $filter, array|object $update, array $options = []): UpdateResult

{

return $this->getCollection()->updateOne($filter, $update, $options);

}

}

**source/classes/RIONotFoundException.php**

<?php

declare(strict\_types=1);

class RIONotFoundException extends Exception

{

}

**source/classes/RIOPartialResolver.php**

<?php

declare(strict\_types=1);

abstract class RIOPartialResolver

{

/\*\*

\* @param RIOResolvedAction $action

\* @return RIOResolvedAction

\* @throws RIOResolveException

\*/

abstract public function resolve(RIOResolvedAction $action): RIOResolvedAction;

}

**source/classes/RIORedirect.php**

<?php

declare(strict\_types=1);

use Symfony\Component\HttpFoundation\RedirectResponse;

class RIORedirect

{

/\*\*

\* @param int $code

\* @return RedirectResponse

\*/

public static function error(int $code): RedirectResponse

{

return RIORedirect::redirectResponse(["error.php?status=$code"]);

}

/\*\*

\* RIORedirect constructor.

\* @param array $path\_parts

\* @return RedirectResponse

\*/

public static function redirectResponse(array $path\_parts = ['']): RedirectResponse

{

return self::redirectToDomain($\_ENV['HOSTNAME'], $path\_parts, "true" === $\_ENV['HTTPS']);

}

/\*\*

\* @param string $path

\* @return RedirectResponse

\*/

public static function redirectWithString(string $path): RedirectResponse

{

return self::redirectToDomainWithString($\_ENV['HOSTNAME'], $path, "true" === $\_ENV['HTTPS']);

}

/\*\*

\* @param string $domain

\* @param array $path\_parts

\* @param bool $secure

\* @return RedirectResponse

\*/

public static function redirectToDomain(string $domain, array $path\_parts, bool $secure = true): RedirectResponse

{

$url\_factory = new RIOUrlFactory($domain, $secure);

$url = $url\_factory->getLocalUrl($path\_parts);

return new RedirectResponse($url->getUrl());

}

/\*\*

\* @param string $domain

\* @param string $path

\* @param bool $secure

\* @return RedirectResponse

\*/

public static function redirectToDomainWithString(string $domain, string $path, bool $secure = true): RedirectResponse

{

$url\_factory = new RIOUrlFactory($domain, $secure);

$protocol = $url\_factory->getHttpOrHttps();

return new RedirectResponse("$protocol://$domain/$path");

}

}

**source/classes/RIORedirectOrException.php**

<?php

use Symfony\Component\HttpFoundation\RedirectResponse;

class RIORedirectOrException

{

/\*\*

\* @param string $message

\* @param int $error\_code

\* @param int $code

\* @param int $severity

\* @param string $filename

\* @param int $line

\* @param null $previous

\* @return \Symfony\Component\HttpFoundation\RedirectResponse

\* @throws \Whoops\Exception\ErrorException

\*/

public static function throwErrorException(

string $message = "",

int $error\_code = 404,

int $code = 0,

int $severity = 1,

string $filename = \_\_FILE\_\_,

int $line = \_\_LINE\_\_,

$previous = null

): RedirectResponse

{

if(RIOConfig::isInDebugMode()) {

throw new \Whoops\Exception\ErrorException(

$message,

$code,

$severity,

$filename,

$line,

$previous

);

}

return RIORedirect::error($error\_code);

}

}

**source/classes/RIOResolvedAction.php**

<?php

declare(strict\_types=1);

class RIOResolvedAction

{

private array $path\_partials;

private RIOMaybe $controller\_namespace;

private RIOMaybe $controller\_action;

private RIOMaybe $frontend;

private RIOMaybe $controller\_action\_parameters;

public function \_\_construct(array $path\_partials)

{

$this->path\_partials = $path\_partials;

$this->controller\_namespace = RIOMaybe::getEmpty();

$this->controller\_action = RIOMaybe::getEmpty();

$this->controller\_action\_parameters = RIOMaybe::getEmpty();

$this->frontend = RIOMaybe::getEmpty();

}

public static function create(self $o): self

{

$obj = new self($o->getPathPartials());

$obj->setControllerNamespace(clone $o->getControllerNamespace());

$obj->setControllerAction(clone $o->getControllerAction());

$obj->setControllerActionParameters(clone $o->getControllerActionParameters());

$obj->setFrontend(clone $o->getFrontend());

return $obj;

}

public function removeFirstArrayElement(): void

{

if (!isset($this->path\_partials[0])) {

if(RIOConfig::isInDebugMode()) {

throw new Error("Trying to remove path partial which doesn't exist");

}

}

unset($this->path\_partials[0]);

$this->path\_partials = array\_values($this->path\_partials);

}

public function getPathPartials(): array

{

return $this->path\_partials;

}

public function setPathPartials(array $path\_partials): void

{

$this->path\_partials = $path\_partials;

}

public function getControllerNamespace(): RIOMaybe

{

return $this->controller\_namespace;

}

public function setControllerNamespace(RIOMaybe $controller\_namespace): void

{

$this->controller\_namespace = $controller\_namespace;

}

public function getControllerAction(): RIOMaybe

{

return $this->controller\_action;

}

public function setControllerAction(RIOMaybe $controller\_action): void

{

$this->controller\_action = $controller\_action;

}

public function getControllerActionParameters(): RIOMaybe

{

return $this->controller\_action\_parameters;

}

public function setControllerActionParameters(RIOMaybe $controller\_action\_parameters): void

{

$this->controller\_action\_parameters = $controller\_action\_parameters;

}

public function getFrontend(): RIOMaybe

{

return $this->frontend;

}

public function setFrontend(RIOMaybe $frontend): void

{

$this->frontend = $frontend;

}

}

**source/classes/RIOResolveException.php**

<?php

declare(strict\_types=1);

class RIOResolveException extends Exception

{

}

**source/classes/RIOResolver.php**

<?php

declare(strict\_types=1);

use Symfony\Component\HttpFoundation\RedirectResponse;

class RIOResolver

{

private RIOResolverConfig $config;

public function \_\_construct(RIOResolverConfig $config)

{

$this->config = $config;

}

/\*\*

\* @throws RIONotFoundException

\*/

public function resolveWithString(string $path): RIOResolvedAction

{

$path = $this->cleanPath($path);

$splitter = new RIOSplitter($path);

$cleaned\_parts = $splitter->splitAt('/')->removeEmptyParts();

$path\_parts = $cleaned\_parts->getParts();

return $this->resolve($path\_parts);

}

/\*\*

\* @throws RIONotFoundException

\*/

public function resolveWithResolvedAction(RIOResolvedAction $resolved\_action): RIOResolvedAction | RedirectResponse

{

/\*\* @var RIOResolveException|null $lastError \*/

$lastError = null;

foreach ($this->config->getSteps() as $step) {

$solved = false;

$lastSuccessfulAction = RIOResolvedAction::create($resolved\_action);

foreach ($step->getPartialResolvers() as $partialResolver) {

$last\_successful\_action\_copy = RIOResolvedAction::create(

$lastSuccessfulAction

);

try {

$resolved\_action = $partialResolver->resolve($last\_successful\_action\_copy);

$solved = true;

break;

} catch (RIOResolveException $e) {

$lastError = $e;

}

}

if (!$solved) {

// 404

if(RIOConfig::isInDebugMode()) {

throw new RIONotFoundException("RIOResolver didn't find it", 0, $lastError);

}

return RIORedirect::error(404);

}

}

return $resolved\_action;

}

/\*\*

\* @param string[] $path\_parts

\* @return RIOResolvedAction

\* @throws RIONotFoundException

\*/

private function resolve(array $path\_parts): RIOResolvedAction

{

$resolved\_action = new RIOResolvedAction($path\_parts);

return $this->resolveWithResolvedAction($resolved\_action);

}

private function cleanPath(string $path): string

{

// Remove get parameters

$getRemoved = explode('?', $path, 2);

return $getRemoved[0];

}

}

**source/classes/RIOResolverConfig.php**

<?php

declare(strict\_types=1);

class RIOResolverConfig

{

/\*\*

\* @var RIOResolverStep[]

\*/

private array $steps;

/\*\*

\* @param RIOResolverStep[] $steps

\*/

public function \_\_construct(array $steps)

{

$this->steps = $steps;

}

/\*\*

\* @return RIOResolverStep[]

\*/

public function getSteps(): array

{

return $this->steps;

}

}

**source/classes/RIOResolverConfigFactory.php**

<?php

declare(strict\_types=1);

class RIOResolverConfigFactory

{

/\*\*

\* @param RIOResolverStep[] $steps

\* @return RIOResolverConfig

\*/

public static function config(array $steps): RIOResolverConfig

{

return new RIOResolverConfig($steps);

}

/\*\*

\* @param RIOPartialResolver[] $partial\_resolvers

\* @return RIOResolverStep

\*/

public static function step(array $partial\_resolvers): RIOResolverStep

{

return new RIOResolverStep($partial\_resolvers);

}

}

**source/classes/RIOResolverStep.php**

<?php

declare(strict\_types=1);

class RIOResolverStep

{

/\*\*

\* @var RIOPartialResolver[]

\*/

private array $partial\_resolvers;

/\*\*

\* @param RIOPartialResolver[] $partial\_resolvers

\*/

public function \_\_construct(array $partial\_resolvers)

{

$this->partial\_resolvers = $partial\_resolvers;

}

/\*\*

\* @return RIOPartialResolver[]

\*/

public function getPartialResolvers(): array

{

return $this->partial\_resolvers;

}

}

**source/classes/RIOSplitString.php**

<?php

declare(strict\_types=1);

final class RIOSplitString

{

private array $string\_parts;

public function \_\_construct(array $string\_parts)

{

$this->string\_parts = $string\_parts;

}

public function removeEmptyParts(): RIOSplitString

{

return new RIOSplitString(

array\_filter($this->string\_parts, function ($part) {

return '' !== $part;

})

);

}

public function transformParts(callable $transformer): RIOSplitString

{

$transformed\_parts = [];

foreach ($this->string\_parts as $string\_part) {

$transformed\_parts[] = call\_user\_func($transformer, $string\_part);

}

return new RIOSplitString($transformed\_parts);

}

public function glueTogether(string $glue): string

{

return implode($glue, $this->string\_parts);

}

public function glueAndPrefixTogether(string $glue): string

{

return $glue.implode($glue, $this->string\_parts);

}

public function glueAndPostfixTogether(string $glue): string

{

return implode($glue, $this->string\_parts).$glue;

}

public function glueAndPrePostfixTogether(string $glue): string

{

return $glue.implode($glue, $this->string\_parts).$glue;

}

public function addParts(array $parts): RIOSplitString

{

return new RIOSplitString(array\_merge($this->string\_parts, $parts));

}

public function hasParts(): bool

{

return count($this->string\_parts) !== 0;

}

public function getParts(): array

{

return array\_values($this->string\_parts);

}

}

**source/classes/RIOSplitter.php**

<?php

declare(strict\_types=1);

final class RIOSplitter

{

private string $string;

public function \_\_construct(string $string)

{

$this->string = $string;

}

public function splitAt(string $delimiter): RIOSplitString

{

return new RIOSplitString(explode($delimiter, $this->string));

}

}

**source/classes/RIOTimeObject.php**

<?php

declare(strict\_types=1);

class RIOTimeObject implements RIOToJSON

{

private DateTime $timeEnd;

private DateTime $workingTimePerformedCorrected;

private DateTime $workingTimePerformed;

private DateTime $timeEndCorrected;

private DateTime $timeStart;

private DateTime $timeStartCorrected;

private DateTime $presenceTime;

private DateTime $presenceTimeCorrected;

private DateTime $isTime;

private string $comment;

/\*\*

\* RIOTimeObject constructor.

\* @throws Exception

\*/

public function \_\_construct()

{

$this->timeStart = RIODateTimeFactory::getDateTime();

$this->timeStartCorrected = RIODateTimeFactory::getDateTime();

$this->timeEnd = RIODateTimeFactory::getDateTime();

$this->timeEndCorrected = RIODateTimeFactory::getDateTime();

$this->presenceTime = RIODateTimeFactory::getDateTime();

$this->presenceTimeCorrected = RIODateTimeFactory::getDateTime();

$this->comment = '';

$this->workingTimePerformedCorrected = RIODateTimeFactory::getDateTime();

$this->workingTimePerformed = RIODateTimeFactory::getDateTime();

$this->isTime = RIODateTimeFactory::getDateTime();

}

public function toJSON(): string

{

$array = [

"start" => $this->timeStart,

"start\_corrected" => $this->timeStartCorrected,

"end" => $this->timeEnd,

"end\_corrected" => $this->timeEndCorrected,

"presence\_time" => $this->presenceTime,

"presence\_time\_corrected" => $this->presenceTimeCorrected,

"comment" => $this->comment,

"is\_time" => $this->isTime

];

return json\_encode($array);

}

/\*\*

\* @return DateTime

\*/

public function getIsTime(): DateTime

{

return $this->isTime;

}

/\*\*

\* @param DateTime $isTime

\*/

public function setIsTime(DateTime $isTime): void

{

$this->isTime = $isTime;

}

public function toObject(): self

{

return new RIOTimeObject();

}

/\*\*

\* @return DateTime

\*/

public function getTimeEnd(): DateTime

{

return $this->timeEnd;

}

/\*\*

\* @param DateTime $timeEnd

\*/

public function setTimeEnd(DateTime $timeEnd): void

{

$this->timeEnd = $timeEnd;

}

/\*\*

\* @return DateTime

\*/

public function getTimeStart(): DateTime

{

return $this->timeStart;

}

/\*\*

\* @param DateTime $timeStart

\*/

public function setTimeStart(DateTime $timeStart): void

{

$this->timeStart = $timeStart;

}

/\*\*

\* @return \DateTime

\*/

public function getTimeEndCorrected(): DateTime

{

return $this->timeEndCorrected;

}

/\*\*

\* @param \DateTime $timeEndCorrected

\*/

public function setTimeEndCorrected(DateTime $timeEndCorrected): void

{

$this->timeEndCorrected = $timeEndCorrected;

}

/\*\*

\* @return \DateTime

\*/

public function getTimeStartCorrected(): DateTime

{

return $this->timeStartCorrected;

}

/\*\*

\* @param \DateTime $timeStartCorrected

\*/

public function setTimeStartCorrected(DateTime $timeStartCorrected): void

{

$this->timeStartCorrected = $timeStartCorrected;

}

/\*\*

\* @return \DateTime

\*/

public function getPresenceTime(): DateTime

{

return $this->presenceTime;

}

/\*\*

\* @param \DateTime $presenceTime

\*/

public function setPresenceTime(DateTime $presenceTime): void

{

$this->presenceTime = $presenceTime;

}

/\*\*

\* @return \DateTime

\*/

public function getPresenceTimeCorrected(): DateTime

{

return $this->presenceTimeCorrected;

}

/\*\*

\* @param \DateTime $presenceTimeCorrected

\*/

public function setPresenceTimeCorrected(DateTime $presenceTimeCorrected): void

{

$this->presenceTimeCorrected = $presenceTimeCorrected;

}

/\*\*

\* @return string

\*/

public function getComment(): string

{

return $this->comment;

}

/\*\*

\* @param string $comment

\*/

public function setComment(string $comment): void

{

$this->comment = $comment;

}

/\*\*

\* @return DateTime

\*/

public function getWorkingTimePerformedCorrected(): DateTime

{

return $this->workingTimePerformedCorrected;

}

/\*\*

\* @param DateTime $workingTimePerformedCorrected

\*/

public function setWorkingTimePerformedCorrected(DateTime $workingTimePerformedCorrected): void

{

$this->workingTimePerformedCorrected = $workingTimePerformedCorrected;

}

/\*\*

\* @return DateTime

\*/

public function getWorkingTimePerformed(): DateTime

{

return $this->workingTimePerformed;

}

/\*\*

\* @param DateTime $workingTimePerformed

\*/

public function setWorkingTimePerformed(DateTime $workingTimePerformed): void

{

$this->workingTimePerformed = $workingTimePerformed;

}

}

**source/classes/RIOToJSON.php**

<?php

declare(strict\_types=1);

interface RIOToJSON

{

public function toJSON(): string;

public function toObject(): object;

}

**source/classes/RIOTwigProvider.php**

<?php

declare(strict\_types=1);

use Twig\Environment;

interface RIOTwigProvider

{

public function getTwig(): Environment;

}

**source/classes/RIOUrl.php**

<?php

/\*\*

\* Allows for easy url generation.

\*

\* With this class you can avoid directly manipulating a string url.

\*/

class RIOUrl

{

private string $domain;

private string $protocol;

private RIOSplitString $path\_parts;

/\*\*

\* A link to a website.

\*

\* RIOUrl constructor.

\* @param string $protocol

\* @param string $domain

\* @param RIOSplitString $path\_parts

\*/

public function \_\_construct(string $protocol, string $domain, RIOSplitString $path\_parts)

{

$this->domain = $domain;

$this->protocol = $protocol;

$this->path\_parts = $path\_parts;

}

public function getUrl(): string

{

$safe\_parts = $this->path\_parts->transformParts(function ($part) {

$encoded\_part = urlencode($part);

// We still have get parameters sometimes in the url

// So we need to use the real characters for get parameters here

// This breaks cases were it shouldn't be a get parameter

$encoded\_part = str\_replace('%3F', '?', $encoded\_part);

return str\_replace('%3D', '=', $encoded\_part);

});

$parts = $safe\_parts->glueTogether('/');

if($safe\_parts->hasParts()) {

return "$this->protocol://$this->domain/$parts/";

} else {

return "$this->protocol://$this->domain/";

}

}

/\*\*

\* @param string[] $parts

\*/

public function addParts(array $parts): void

{

$this->path\_parts = $this->path\_parts->addParts($parts);

}

}

**source/classes/RIOUrlFactory.php**

<?php

declare(strict\_types=1);

class RIOUrlFactory

{

private string $hostname;

private bool $secure;

public function \_\_construct(

string $hostname,

bool $secure

) {

$this->hostname = $hostname;

$this->secure = $secure;

}

public static function get(): RIOUrlFactory

{

$domain = $\_ENV['HOSTNAME'];

$secure = $\_ENV['HTTPS'];

return new RIOUrlFactory(

$domain,

$secure

);

}

public function getLocalUrl(array $parts): RIOUrl

{

$split\_string = new RIOSplitString($parts);

return $this->getUrl($this->secure, $this->hostname, $split\_string);

}

public function getHttpUrl(string $domain, array $parts): RIOUrl

{

$split\_string = new RIOSplitString($parts);

return $this->getUrl(false, $domain, $split\_string);

}

public function getHttpsUrl(string $domain, array $parts): RIOUrl

{

$split\_string = new RIOSplitString($parts);

return $this->getUrl(true, $domain, $split\_string);

}

private function getUrl(bool $secure, string $domain, RIOSplitString $parts): RIOUrl

{

$protocol = $this->getProtocol($secure);

return new RIOUrl($protocol, $domain, $parts);

}

public function getHttpOrHttps(): string

{

return $this->getProtocol($this->secure);

}

private function getProtocol(bool $secure): string

{

$return = 'http';

if (true === $secure) {

$return .= 's';

}

return $return;

}

}

**source/classes/RIOUserObject.php**

<?php

class RIOUserObject implements RIOToJSON

{

private string $username;

private string $sessionId;

private bool $timeRecordStarted;

private array $workDays;

private DateTime $mandatoryTime;

private string $location;

/\*\*

\* RIOUserObject constructor.

\* @throws Exception

\*/

public function \_\_construct(RIOAccessController $accessController = null)

{

if(null !== $accessController) {

$user = $accessController->getUser();

$this->timeRecordStarted = $user["timeRecordStarted"];

$this->sessionId = $user["sessionId"];

$this->username = $user["sessionUsername"];

$this->workDays = $accessController->getWorkDaysByUser();

if(false === array\_key\_exists("mandatoryTime", $user)){

$this->mandatoryTime = RIODateTimeFactory::getDateTime();

// New created user has by default 8 hour and 0 minute mandatory time

$this->mandatoryTime->setTime(8,0);

} else {

$this->mandatoryTime = RIODateTimeFactory::getDateTime($user["mandatoryTime"]);

}

} else {

$this->mandatoryTime = RIODateTimeFactory::getDateTime();

$this->mandatoryTime->setTime(8,0);

}

// New created user has by default location dietikon

$this->location = "dietikon";

}

public function toJSON(): string

{

$array = [

"sessionUsername" => $this->username,

"displayUsername" => '',

"surnameUsername" => '',

"sessionId" => $this->sessionId,

"timeRecordStarted" => $this->timeRecordStarted,

"mandatoryTime" => $this->mandatoryTime,

"location" => $this->location

];

return json\_encode($array);

}

/\*\*

\* @return string

\*/

public function getLocation(): string

{

return $this->location;

}

/\*\*

\* @param string $location

\*/

public function setLocation(string $location): void

{

$this->location = $location;

}

/\*\*

\* @return DateTime

\*/

public function getMandatoryTime(): DateTime

{

return $this->mandatoryTime;

}

/\*\*

\* @param DateTime $mandatoryTime

\*/

public function setMandatoryTime(DateTime $mandatoryTime): void

{

$this->mandatoryTime = $mandatoryTime;

}

/\*\*

\* @return array

\*/

public function getWorkDays(): array

{

return $this->workDays;

}

/\*\*

\* @param array $workDays

\*/

public function setWorkDays(array $workDays): void

{

$this->workDays = $workDays;

}

/\*\*

\* @return string

\*/

public function getUsername(): string

{

return $this->username;

}

/\*\*

\* @param string $username

\*/

public function setUsername(string $username): void

{

$this->username = $username;

}

/\*\*

\* @return string

\*/

public function getSessionId(): string

{

return $this->sessionId;

}

/\*\*

\* @param string $sessionId

\*/

public function setSessionId(string $sessionId): void

{

$this->sessionId = $sessionId;

}

/\*\*

\* @return bool

\*/

public function isTimeRecordStarted(): bool

{

return $this->timeRecordStarted;

}

public function isTimeRecordStopped(): bool

{

return !$this->isTimeRecordStarted();

}

/\*\*

\* @param bool $timeRecordStarted

\*/

public function setTimeRecordStarted(bool $timeRecordStarted): void

{

$this->timeRecordStarted = $timeRecordStarted;

}

public function toObject(): self

{

return new RIOUserObject();

}

}

**source/classes/RIOWorkDayObject.php**

<?php

use Symfony\Component\HttpClient\CurlHttpClient;

use Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface;

use Symfony\Contracts\HttpClient\ResponseInterface;

use function source\getAPIAbsolutePath;

class RIOWorkDayObject implements RIOToJSON

{

private string $username;

private DateTime $presenceTime;

private RIOAbsentOptionObject $absentAllDay;

private RIOAbsentOptionObject $absentAfternoon;

private DateTime $date;

private DateTime $deviation;

private DateTime $hoursWorked;

private bool $specialCompensation;

private DateTime $mandatoryTime;

private DateTime $monthlyBalance;

private RIOAbsentOptionObject $absentMorning;

/\*\*

\* The actual minutes an user has worked at one day as an single int

\*

\* @var int

\*/

private int $totalMinutes;

private DateTime $timeCredit;

private DateTime $timeCreditCorrected;

private DateTime $totalBalance;

private DateTime $weeklyBalance;

/\*\*

\* @var RIOTimeObject[]

\*/

private array $times;

/\*\*

\* RIOWorkDayObject constructor.

\* @param \DateTime|null $date

\* @throws \Symfony\Contracts\HttpClient\Exception\TransportExceptionInterface

\* @throws \Exception

\*/

public function \_\_construct(DateTime $date = null)

{

$this->username = '';

$this->absentAllDay = new RIOAbsentOptionObject();

$this->absentAfternoon = new RIOAbsentOptionObject();

if(null === $date) {

$this->date = RIODateTimeFactory::getDateTime();

} else {

$this->date = $date;

}

$this->deviation = RIODateTimeFactory::getDateTime();

$this->hoursWorked = RIODateTimeFactory::getDateTime();

$this->specialCompensation = $this->isRIOHoliday() || $this->isSunday() || $this->isSaturday();

$this->mandatoryTime = RIODateTimeFactory::getDateTime();

if(true === $this->specialCompensation) {

$this->mandatoryTime->setTime(0,0);

} else {

$this->mandatoryTime->setTime(8,0);

}

$this->monthlyBalance = RIODateTimeFactory::getDateTime();

$this->absentMorning = new RIOAbsentOptionObject();

$this->totalMinutes = 0;

$this->timeCredit = RIODateTimeFactory::getDateTime();

$this->timeCredit->setTime(0,0);

$this->totalBalance = RIODateTimeFactory::getDateTime();

$this->weeklyBalance = RIODateTimeFactory::getDateTime();

$this->times = [];

$this->presenceTime = RIODateTimeFactory::getDateTime();

}

public function toJSON(): string

{

$array = [

"sessionUsername" => $this->username,

"date" => $this->date,

"mandatoryTime" => $this->mandatoryTime,

"time" => $this->times,

"presenceTime" => $this->presenceTime

];

return json\_encode($array);

}

/\*\*

\* @return DateTime

\*/

public function getTimeCreditCorrected(): DateTime

{

return $this->timeCreditCorrected;

}

/\*\*

\* @param DateTime $timeCreditCorrected

\*/

public function setTimeCreditCorrected(DateTime $timeCreditCorrected): void

{

$this->timeCreditCorrected = $timeCreditCorrected;

}

/\*\*

\* @return string

\*/

public function getUsername(): string

{

return $this->username;

}

/\*\*

\* @param string $username

\*/

public function setUsername(string $username): void

{

$this->username = $username;

}

/\*\*

\* @return DateInterval[]

\*/

public function getStartAndEndDiff(): array

{

$diff\_list = [];

foreach ($this->times as $time) {

$origin = $time->getTimeStart();

$target = $time->getTimeEnd();

$interval = $origin->diff($target);

$diff\_list[] = $interval;

}

return $diff\_list;

}

/\*\*

\* This will be a cron job

\*

\*/

public function sum(): void

{

$totalList = $this->getStartAndEndDiff();

$totalMinutes = 0;

foreach ($totalList as $total) {

$minutes = $total->m;

$hours = $total->h;

$hoursInMinutes = $hours \* 60;

$totalMinutes = $totalMinutes + $minutes + $hoursInMinutes;

}

$this->totalMinutes = $totalMinutes;

}

/\*\*

\* @return bool

\* @throws TransportExceptionInterface

\*/

public function isRIOHoliday(): bool

{

$user = new RIOUserObject();

$date = $this->getDate();

$response = $this->getHoliday($user->getLocation(), $date->format("Y-m-d"));

return '' !== json\_decode($response->getContent(),true)["holiday"];

}

public function getHoliday(string $location, string $date): ResponseInterface

{

$client = new CurlHttpClient();

return $client->request(

"GET",

getAPIAbsolutePath(

[

"api"

],

"getHoliday.php?location=".$location."&date=".$date

),

[

"verify\_peer" => false,

"verify\_host" => false

]

);

}

public function isDayByDayName(string $dayName): bool

{

if($this->date->format("D") === $dayName) {

return true;

} else {

return false;

}

}

public function isSunday(): bool

{

return $this->isDayByDayName("Sun");

}

public function isSaturday(): bool

{

return $this->isDayByDayName("Sat");

}

public function getWeekDay(): string

{

$dayNames = [

'Montag',

'Dienstag',

'Mittwoch',

'Donnerstag',

'Freitag',

'Samstag',

'Sonntag'

];

return $dayNames[(int) $this->date->format('N')-1];

}

public function getFormattedDate(): string

{

$date = $this->date;

$day = $date->format("d");

$month = $date->format("m");

$year = $date->format("Y");

$monthNames = [

"Januar",

"Februar",

"März",

"April",

"Mai",

"Juni",

"Juli",

"August",

"September",

"Oktober",

"November",

"Dezember"

];

$monthName = $monthNames[(int) $month-1];

return $day.". ".$monthName." ".$year;

}

public function getFormattedDateTwo(): string

{

$date = $this->date;

return $date->format("d.m.Y");

}

/\*\*

\* @return RIOAbsentOptionObject

\*/

public function getAbsentAllDay(): RIOAbsentOptionObject

{

return $this->absentAllDay;

}

/\*\*

\* @param RIOAbsentOptionObject $absentAllDay

\*/

public function setAbsentAllDay(RIOAbsentOptionObject $absentAllDay): void

{

$this->absentAllDay = $absentAllDay;

}

/\*\*

\* @return RIOAbsentOptionObject

\*/

public function getAbsentAfternoon(): RIOAbsentOptionObject

{

return $this->absentAfternoon;

}

/\*\*

\* @param RIOAbsentOptionObject $absentAfternoon

\*/

public function setAbsentAfternoon(RIOAbsentOptionObject $absentAfternoon): void

{

$this->absentAfternoon = $absentAfternoon;

}

/\*\*

\* @return RIOAbsentOptionObject

\*/

public function getAbsentMorning(): RIOAbsentOptionObject

{

return $this->absentMorning;

}

/\*\*

\* @param RIOAbsentOptionObject $absentMorning

\*/

public function setAbsentMorning(RIOAbsentOptionObject $absentMorning): void

{

$this->absentMorning = $absentMorning;

}

/\*\*

\* @return DateTime

\*/

public function getDate(): DateTime

{

return $this->date;

}

/\*\*

\* @param DateTime $date

\*/

public function setDate(DateTime $date): void

{

$this->date = $date;

}

/\*\*

\* @return DateTime

\*/

public function getDeviation(): DateTime

{

return $this->deviation;

}

/\*\*

\* @param DateTime $deviation

\*/

public function setDeviation(DateTime $deviation): void

{

$this->deviation = $deviation;

}

/\*\*

\* @return DateTime

\*/

public function getHoursWorked(): DateTime

{

return $this->hoursWorked;

}

/\*\*

\* @param DateTime $hoursWorked

\*/

public function setHoursWorked(DateTime $hoursWorked): void

{

$this->hoursWorked = $hoursWorked;

}

/\*\*

\* @return bool

\*/

public function isSpecialCompensation(): bool

{

return $this->specialCompensation;

}

/\*\*

\* @param bool $specialCompensation

\*/

public function setSpecialCompensation(bool $specialCompensation): void

{

$this->specialCompensation = $specialCompensation;

}

/\*\*

\* @return DateTime

\*/

public function getMandatoryTime(): DateTime

{

return $this->mandatoryTime;

}

/\*\*

\* @param DateTime $mandatoryTime

\*/

public function setMandatoryTime(DateTime $mandatoryTime): void

{

$this->mandatoryTime = $mandatoryTime;

}

/\*\*

\* @return DateTime

\*/

public function getMonthlyBalance(): DateTime

{

return $this->monthlyBalance;

}

/\*\*

\* @param DateTime $monthlyBalance

\*/

public function setMonthlyBalance(DateTime $monthlyBalance): void

{

$this->monthlyBalance = $monthlyBalance;

}

/\*\*

\* @return int

\*/

public function getTotalMinutes(): int

{

return $this->totalMinutes;

}

/\*\*

\* @param int $totalMinutes

\*/

public function setTotalMinutes(int $totalMinutes): void

{

$this->totalMinutes = $totalMinutes;

}

/\*\*

\* @return DateTime

\*/

public function getTimeCredit(): DateTime

{

return $this->timeCredit;

}

/\*\*

\* @param DateTime $timeCredit

\*/

public function setTimeCredit(DateTime $timeCredit): void

{

$this->timeCredit = $timeCredit;

}

/\*\*

\* @return DateTime

\*/

public function getTotalBalance(): DateTime

{

return $this->totalBalance;

}

/\*\*

\* @param DateTime $totalBalance

\*/

public function setTotalBalance(DateTime $totalBalance): void

{

$this->totalBalance = $totalBalance;

}

/\*\*

\* @return DateTime

\*/

public function getWeeklyBalance(): DateTime

{

return $this->weeklyBalance;

}

/\*\*

\* @param DateTime $weeklyBalance

\*/

public function setWeeklyBalance(DateTime $weeklyBalance): void

{

$this->weeklyBalance = $weeklyBalance;

}

/\*\*

\* @return RIOTimeObject[]

\*/

public function getTimes(): array

{

return $this->times;

}

/\*\*

\* @param RIOTimeObject[] $times

\*/

public function setTimes(array $times): void

{

$this->times = $times;

}

public function toObject(): self

{

return new RIOWorkDayObject();

}

/\*\*

\* @return DateTime

\*/

public function getPresenceTime(): DateTime

{

return $this->presenceTime;

}

/\*\*

\* @param DateTime $presenceTime

\*/

public function setPresenceTime(DateTime $presenceTime): void

{

$this->presenceTime = $presenceTime;

}

}

**source/js/\_main.js**

import '@popperjs/core';

import 'bootstrap';

**source/scss/\_custom.scss**

// Bootstrap override

//$table-cell-padding-y: .0rem !default;

// Custom

td, th {

vertical-align: middle;

}

**source/scss/\_main.scss**

@import "custom";

@import "../../node\_modules/bootstrap/scss/bootstrap";

**source/templates/rioadmin/modules/rioadmin/edit\_user.twig**

{% extends "base.twig" %}

{% block content %}

<h3>Benutzer</h3>

<form action="{{ getAbsolutePath(['rioadmin','updateUser',sessionUsername]) }}" method="post">

<div class="mb-3">

<label for="mandatoryTime" class="form-label">Pflichtzeit pro Tag</label>

<input type="time" class="form-control" id="mandatoryTime" name="mandatoryTime" value="{{ mandatoryTime }}" aria-describedby="mandatoryTimeHelp">

<div id="mandatoryTimeHelp" class="form-text">Von Montag bis Freitag ausgenommen von Samstag, Sonntag und Feiertag. Diese Änderung wird erst am nächsten Tag in Kraft treten.</div>

</div>

{% if state is same as("success") %}

<div class="mb-3">

<span class="badge bg-success">Änderung wurde erfolgreich übernommen.</span>

</div>

{% endif %}

{% if state is same as("failure") %}

<div class="mb-3">

<span class="badge bg-danger">Änderung wurde nicht erfolgreich übernommen.</span>

</div>

{% endif %}

<div>

<button type="submit" class="btn btn-primary">OK</button>

</div>

</form>

{% endblock %}

**source/templates/rioadmin/modules/rioadmin/overview.twig**

{% extends "base.twig" %}

{% macro link(oneWorkDayFromUserPast, key) %}

<a href="{{ getAbsolutePath(['rioadmin','presenceTimeCorrections']|merge(oneWorkDayFromUserPast.presenceTimeCorrections)|merge([key])) ~ 'unchanged' }}"><img src="{{ getAbsolutePath(['img']) ~ 'properties.gif' }}" alt="Details" title="bearbeiten"></a>

{% endmacro %}

{% block content %}

{% import \_self as overview %}

{# @var MongoDB\Model\BSONArray[] all\_work\_days\_from\_user\_past #}

<h3>Übersicht</h3>

<div>

{{ [] is same as(allWorkDaysFromUserPast) ? "Es hat keine Einträge" }}

</div>

<div class="d-flex justify-content-center">

<div class="p-2">

<p>Präsenzzeiten für {{ displayUsername ~ ' ' ~ surnameUsername }}</p>

</div>

</div>

<div class="d-flex justify-content-around">

<div class="p-2">

{% if previousMonth is not same as('') %}

<a href="{{ getAbsolutePath(['rioadmin','overview',sessionUsername,previousMonth]) }}" class="d-none d-md-inline">{{ previousMonthName }}</a>

{% endif %}

</div>

<div class="p-2">

<p>{{ currentMonthName }}</p>

</div>

<div class="p-2">

{% if nextMonth is not same as('') %}

<a href="{{ getAbsolutePath(['rioadmin','overview',sessionUsername,nextMonth]) }}" class="d-none d-md-inline">{{ nextMonthName }}</a>

{% endif %}

</div>

</div>

<table class="table table-bordered">

<thead>

<tr>

<th scope="col"></th>

<th scope="col" colspan="2">Datum</th>

<th scope="col">Startzeit</th>

<th scope="col">Endzeit</th>

<th scope="col">Soll</th>

<th scope="col">Ist</th>

<th scope="col">Differenz</th>

<th scope="col">Zeitsaldi (Woche)</th>

<th scope="col">Zeitsaldi (Monat)</th>

<th scope="col">Zeitsaldi (Total)</th>

<th scope="col">Kommentar</th>

</tr>

</thead>

<tbody>

{# @var MongoDB\Model\BSONArray oneWorkDayFromUserPast #}

{% for oneWorkDayFromUserPast in allWorkDaysFromUserPast %}

{% set length = oneWorkDayFromUserPast.offsetGet("time")|length %}

{% set printMandatoryTime = false %}

{% set printDate = false %}

{# @var MongoDB\Model\BSONDocument time #}

{% for key,time in oneWorkDayFromUserPast.offsetGet("time") %}

<tr>

<td>{{ overview.link(oneWorkDayFromUserPast, key) }}</td>

{% if false is same as(printDate) %}

<th scope="row" rowspan="{{ length }}">{{ getWeekDayShortNameByDate(oneWorkDayFromUserPast.date) }}</th>

<th scope="row" rowspan="{{ length }}">{{ oneWorkDayFromUserPast.date }}</th>

{% endif %}

<td>{{ '' is same as(time.offsetGet("startCorrected")) ? time.offsetGet("start") : time.offsetGet("startCorrected") }}</td>

<td>{{ '' is same as(time.offsetGet("endCorrected")) ? time.offsetGet("end") : time.offsetGet("endCorrected") }}</td>

{% if false is same as(printMandatoryTime) %}

<td rowspan="{{ length }}">{{ oneWorkDayFromUserPast.offsetGet("mandatoryTime") }}</td>

{% set printMandatoryTime = true %}

{% endif %}

<td>{{ time.offsetGet("isTime") }}</td>

<td>{{ time.offsetGet("diffNegativePositive")~time.offsetGet("diff") }}</td>

{% if false is same as(printDate) %}

<td rowspan="{{ length }}">{{ oneWorkDayFromUserPast.offsetGet("deviationTimeWeekly") }}</td>

<td rowspan="{{ length }}">{{ oneWorkDayFromUserPast.offsetGet("deviationTimeMonthly") }}</td>

<td rowspan="{{ length }}">{{ oneWorkDayFromUserPast.offsetGet("deviationTimeTotal") }}</td>

{% set printDate = true %}

{% endif %}

<td>{{ time.offsetGet("comment") }}</td>

</tr>

{% endfor %}

{% if loop.last %}

<tr class="border-bottom-0">

<td class="border-0"></td>

<td class="border-0"></td>

<th scope="row" class="border-0">Total</th>

<td class="border-0"></td>

<td class="border-0"></td>

<td class="border-bottom">{{ oneWorkDayFromUserPast.offsetGet("mandatoryTimeMonthly") }}</td>

<td class="border-bottom">{{ oneWorkDayFromUserPast.offsetGet("isTimeMonthly") }}</td>

<td class="border-0"></td>

<td class="border-0"></td>

<td class="border-bottom">{{ oneWorkDayFromUserPast.offsetGet("deviationTimeMonthly") }}</td>

<td class="border-bottom">{{ oneWorkDayFromUserPast.offsetGet("deviationTimeTotal") }}</td>

<td class="border-0"></td>

</tr>

{% endif %}

{% endfor %}

</tbody>

</table>

{% endblock %}

**source/templates/rioadmin/modules/rioadmin/presence\_time\_corrections.twig**

{% extends "base.twig" %}

{% block content %}

<h3>Korrektur</h3>

<p>Dies sind die Präsenzdaten vom {{ date }} für {{ displayUsername ~ ' ' ~ surnameUsername }}.</p>

<p>Du kannst hier Korrekturen vornehmen.</p>

<form action="{{ getAbsolutePath(['rioadmin','updatePresenceTimeCorrections']|merge(usernameDateTimeIndex)) }}" method="post">

<div class="mb-3">

<label for="date" class="form-label">Datum: </label>

<input readonly id="date" name="date" class="form-control" value="{{ date }}">

</div>

<table class="table">

<thead>

<tr>

<th scope="col"></th>

<th scope="col">gegeben errechnet</th>

<th scope="col">korrigiert</th>

<th scope="col"></th>

</tr>

</thead>

<tbody>

<tr>

<th scope="row">Arbeitsbeginn</th>

<td>{{ timeStart }}</td>

<td>

<label for="startCorrected" class="form-label d-inline"></label>

<input type="time" class="form-control" id="startCorrected" name="startCorrected" value="{{ timeStartCorrected }}">

</td>

<td>Zeitpunkt der Anmeldung</td>

</tr>

<tr>

<th scope="row">Arbeitsende</th>

<td>{{ timeEnd }}</td>

<td>

<label for="endCorrected" class="form-label d-inline"></label>

<input type="time" class="form-control" id="endCorrected" name="endCorrected" value="{{ timeEndCorrected }}">

</td>

<td>Zeitpunkt der Abmeldung</td>

</tr>

<tr>

<th scope="row">Pflichtzeit</th>

<td>{{ mandatoryTime }}</td>

<td>

<label for="mandatoryTimeCorrected" class="form-label d-inline"></label>

<input class="form-control" id="mandatoryTimeCorrected" name="mandatoryTimeCorrected">

</td>

<td>Pflicht-Arbeitszeit</td>

</tr>

<tr>

<th scope="row">Zeitgutschrift</th>

<td>{{ timeCredit }}</td>

<td>

<label for="timeCreditCorrected" class="form-label d-inline"></label>

<input class="form-control" id="timeCreditCorrected" name="timeCreditCorrected" value="{{ timeCreditCorrected }}">

</td>

<td>kann auch negativ ausfallen</td>

</tr>

<tr>

<th scope="row">Präsenzzeit</th>

<td>{{ presenceTime }}</td>

<td>

<label for="presenceTimeCorrected" class="form-label d-inline"></label>

<input class="form-control" id="presenceTimeCorrected" name="presenceTimeCorrected">

</td>

<td>Präsenzzeit</td>

</tr>

<tr>

<th scope="row">geleistete Arbeitszeit</th>

<td>{{ presenceTimeTotal }}</td>

<td>

<label for="workingTimePerformedCorrected" class="form-label d-inline"></label>

<input class="form-control" id="workingTimePerformedCorrected" name="workingTimePerformedCorrected" value="{{ workingTimePerformedCorrected }}">

</td>

<td>grob gesagt: Präsenz minus Pause</td>

</tr>

<tr>

<th scope="row">Abweichung</th>

<td>{{ deviationNegativeOrPositiveOrZero ~ deviation }}</td>

<td></td>

<td>ergibt sich aus der Differenz zwischen geleisteter Arbeit ({{ presenceTimeTotal }}) und Pflicht-Arbeitszeit ({{ mandatoryTime }})</td>

</tr>

<tr>

<th scope="row">Abwesend?</th>

<td><div><label for="absentAllDay"></label>{{ absentAllDay }}</div></td>

<td>

<select class="form-select" id="absentAllDay" name="absentAllDay" aria-label="absentAllDay">

{% for option in absentOptions %}

<option value="{{ option }}" {{ option is same as(absentAllDay) ? "selected" }}>{{ option }}</option>

{% endfor %}

</select>

</td>

<td>für ganztägige Abwesenheiten</td>

</tr>

<tr>

<th scope="row">vormittags</th>

<td><div><label for="absentMorning"></label>{{ absentMorning }}</div></td>

<td>

<select class="form-select" id="absentMorning" name="absentMorning" aria-label="absentMorning">

{% for option in absentOptions %}

<option value="{{ option }}" {{ option is same as(absentMorning) ? "selected" }}>{{ option }}</option>

{% endfor %}

</select>

</td>

<td>für Abwesenheiten am Vormittag</td>

</tr>

<tr>

<th scope="row">nachmittags</th>

<td><div><label for="absentAfternoon"></label>{{ absentAfternoon }}</div></td>

<td>

<select class="form-select" id="absentAfternoon" name="absentAfternoon" aria-label="absentAfternoon">

{% for option in absentOptions %}

<option value="{{ option }}" {{ option is same as(absentAfternoon) ? "selected" }}>{{ option }}</option>

{% endfor %}

</select>

</td>

<td>für Abwesenheiten am Nachmittag</td>

</tr>

<tr>

<th scope="row">Wochensaldo</th>

<td>{{ deviationTimeWeekly }}</td>

<td></td>

<td rowspan="2">können nicht korrigiert werden</td>

</tr>

<tr>

<th scope="row">Monatssaldo</th>

<td>{{ deviationTimeMonthly }}</td>

<td></td>

</tr>

<tr>

<th scope="row">Gesamtsaldo</th>

<td>{{ deviationTimeTotal }}</td>

<td>

<label for="deviationTimeTotalCorrected" class="form-label d-inline"></label>

<input class="form-control" id="deviationTimeTotalCorrected" name="deviationTimeTotalCorrected" value="{{ deviationTimeTotalCorrected }}">

</td>

<td>der hingegen schon</td>

</tr>

<tr>

<th scope="row">Kommentar</th>

<td colspan="2">

<div class="mb-3">

<label for="comment" class="form-label d-inline"></label>

<textarea class="form-control" id="comment" name="comment" rows="3">{{ comment }}</textarea>

</div>

</td>

<td>Wenn du irgendwelche Korrekturen vornimmst, musst du sie kurz kommentieren</td>

</tr>

</tbody>

</table>

{% if lastEditedUser is not same as('') or lastEditedDate is not same as('') or lastEditedTime is not same as('') %}

<p>Wurde von {{ lastEditedUser }} zuletzt am {{ lastEditedDate }} um {{ lastEditedTime }} Uhr korrigiert.</p>

{% endif %}

{% if state is same as("failure") %}

<div class="mb-3">

<span class="badge bg-danger">Änderung wurde nicht erfolgreich übernommen.</span>

</div>

{% endif %}

<div>

<button type="submit" class="btn btn-primary">OK</button>

</div>

</form>

{% endblock %}

**source/templates/rioadmin/modules/rioadmin/user\_home.twig**

{% extends "base.twig" %}

{% block content %}

<h3>Zeiterfassung</h3>

<p>Willkommen, {{ displayUsername }}, bei der Administration von Rafisa</p>

<p>Heute ist {{ day }}, der {{ date }}</p>

<form action="{{ getAbsolutePath(["rioadmin", "logout"]) }}">

<div class="my-3">

<button type="submit" class="btn btn-primary">Abmelden</button>

</div>

</form>

{% if true is same as(timeRecordStarted) %}

<form action="{{ getAbsolutePath(["rioadmin", "stop"]) }}">

<div class="mb-3">

<button type="submit" class="btn btn-primary">Zeiterfassung stoppen</button>

</div>

</form>

{% else %}

<form action="{{ getAbsolutePath(["rioadmin", "start"]) }}">

<div class="mb-3">

<button type="submit" class="btn btn-primary">Zeiterfassung starten</button>

</div>

</form>

{% endif %}

{% endblock %}

**source/templates/riomain/modules/riomain/home.twig**

{% extends "base.twig" %}

{% block content %}

{{ show\_form }}

<h3>Login</h3>

<form action="{{ action }}" method="post">

<div class="mb-3">

<label for="username" class="form-label">Benutzername:</label>

<input id="username" name="username" type="text" placeholder="m.muster" class="form-control" required>

</div>

<div class="mb-3">

<label for="password" class="form-label">Passwort:</label>

<input id="password" name="password" type="password" placeholder="Passwort" class="form-control" required>

</div>

{% if state is same as("failure") %}

<div class="mb-3">

<span class="badge bg-danger">Logindaten sind falsch.</span>

</div>

{% endif %}

<div>

<button type="submit" class="btn btn-primary">Login</button>

</div>

</form>

{% endblock %}

**source/views/base.twig**

<!DOCTYPE html>

<html lang="de">

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta charset="UTF-8">

<meta name="description" content="Zeiterfassungssystem">

<title>{{ title }}</title>

<link rel="shortcut icon" type="image/x-icon" href="{{ getAbsolutePath([], "favicon.ico") }}" sizes="16x16">

<link id="theme" rel="stylesheet" type="text/css" href="{{ getAbsolutePath(["css"]) ~ "main.css" }}" />

{% block head %}{% endblock %}

</head>

<body>

<header>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="container-fluid">

<a class="navbar-brand" href="{{ logoLink() }}">

<img class="img-fluid" src="{{ getAbsolutePath(["img"], "logo.gif") }}" alt="logo">

</a>

{% if isLoggedIn() is same as(true) %}

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

{% for item in nav %}

<li class="nav-item">

<a href="{{ item.link }}" class="{{ item.mobile is same as(false) ? 'd-none d-md-flex ' : '' }}nav-link{{ item.active is same as(true) ? ' active' : '' }}"{{ item.active is same as(true) ? ' aria-current="page"' : '' }}>{{ item.name }}</a>

</li>

{% endfor %}

</ul>

</div>

{% endif %}

</div>

</nav>

</header>

<div class="container-fluid pt-4">

<div class="container">

<div class="row justify-content-center">

<div class="col-12">

{% block content %}{% endblock %}

</div>

</div>

</div>

</div>

<script src="{{ getAbsolutePath(["js"]) ~ "main.js" }}"></script>

</body>

</html>

## Abbildungsverzeichnis

[Abbildung 1: Baumdiagramm 7](#_Toc73049152)

[Abbildung 2: Projektzeitplan 9](#_Toc73049153)

[Abbildung 3: Übersicht 27](#_Toc73049154)

[Abbildung 4: Korrektur 28](#_Toc73049155)

[Abbildung 5: System Diagramm 29](#_Toc73049156)

[Abbildung 6: Use Cases 30](#_Toc73049157)

[Abbildung 7: Aktivitäten Diagramm (Login) 31](#_Toc73049158)

[Abbildung 8: Generelle Korrektur 32](#_Toc73049159)

[Abbildung 9: Detaillierte Korrekturen 33](#_Toc73049160)

[Abbildung 10: ERM 34](#_Toc73049161)

[Abbildung 11: UML (manuell) 35](#_Toc73049162)

[Abbildung 12: UML (PhpStorm) 36](#_Toc73049163)

[Abbildung 13: Login-Maske (Desktop) 40](#_Toc73049164)

[Abbildung 14: Login-Maske (Mobile) 40](#_Toc73049165)

[Abbildung 15: Zeiterfassungsmaske (Desktop) 41](#_Toc73049166)

[Abbildung 16: Zeiterfassungsmaske (Mobile) 41](#_Toc73049167)

[Abbildung 17: Benutzer Pflichtzeit (Desktop) 42](#_Toc73049168)

[Abbildung 18: Benutzer Pflichtzeit (Mobile) 42](#_Toc73049169)

[Abbildung 19: Übersichtsmaske (oben) 43](#_Toc73049170)

[Abbildung 20: Übersichtsmaske (unten) 43](#_Toc73049171)

[Abbildung 21: Tageskorrektur 44](#_Toc73049172)

[Abbildung 22: Testkonzept 1 (Seite 1) 45](#_Toc73049173)

[Abbildung 23: Testkonzept 1 (Seite 2) 46](#_Toc73049174)

[Abbildung 24: Testkonzept 1 (Seite 3) 47](#_Toc73049175)

## Tabellenverzeichnis

[Tabelle 2: Tagesprotokoll vom 12.5.2021 11](#_Toc73049176)

[Tabelle 4: Tagesprotokoll vom 9.5.2017 13](#_Toc73049177)

[Tabelle 5: Tagesprotokoll vom 17.5.2021 14](#_Toc73049178)

[Tabelle 6: Tagesprotokoll vom 18.5.2021 15](#_Toc73049179)

[Tabelle 7: Tagesprotokoll vom 19.5.2021 16](#_Toc73049180)

[Tabelle 8: Tagesprotokoll vom 20.5.2021 17](#_Toc73049181)

[Tabelle 9: Tagesprotokoll vom 21.5.2021 18](#_Toc73049182)

[Tabelle 11: Tagesprotokoll vom 25.5.2021 19](#_Toc73049183)

[Tabelle 12: Tagesprotokoll vom 26.5.2021 20](#_Toc73049184)

[Tabelle 12: Tagesprotokoll vom 27.5.2021 21](#_Toc73049185)

[Tabelle 12: Tagesprotokoll vom 28.5.2021 22](#_Toc73049186)

## Listing des Programmcodes