ChronoCommand
Webbasierte Zeiterfassung

Entwurf

Jannis Friedmann
David Kuhmann
Maria Schmid
Xiaoming Wang
Jan Zenkner

18. Januar 2016

Inhaltsverzeichnis

1 Klassen **1**
1.1 Model . . . 3
1.2 Control . . . 16
1.3 View . . . 23
1.4 Entwurfsmuster . . . 28

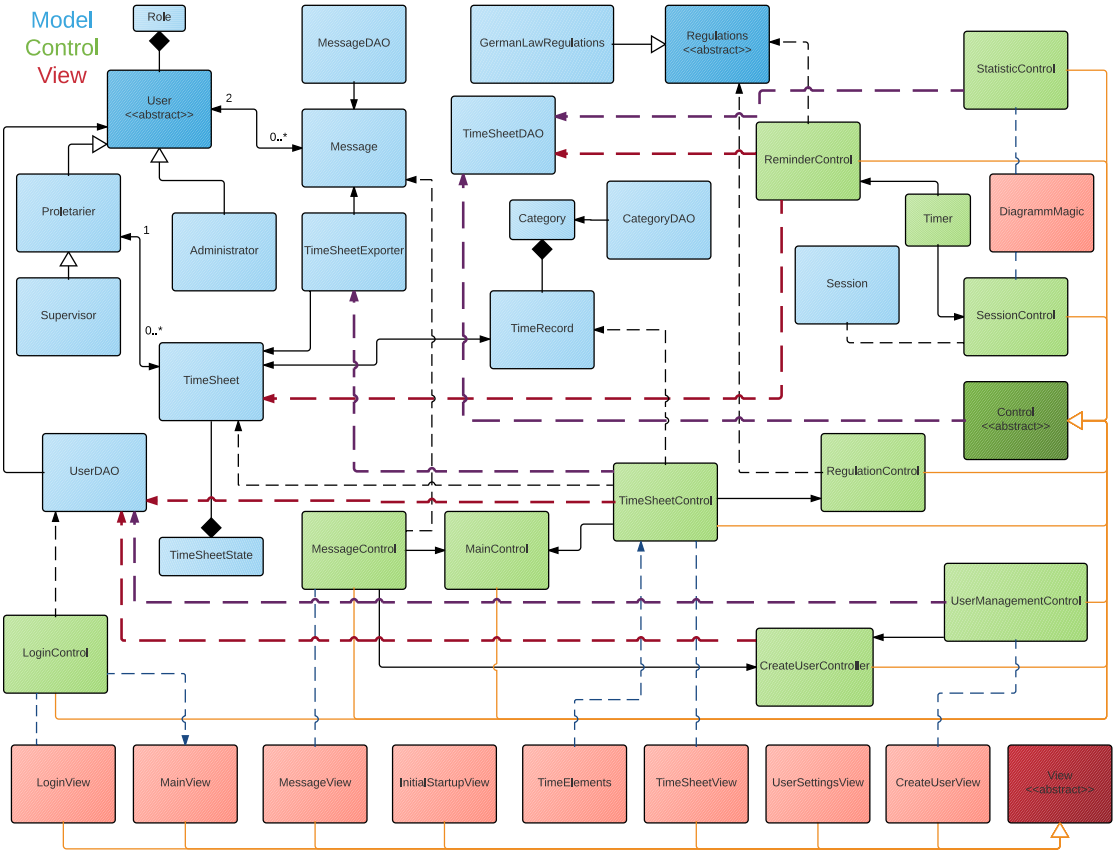
2 Sequenzen **29**
2.1 Neue Zeiterfassung . . . 29
2.2 Login Sequenz . . . 30
2.3 Alle Stundenzettel der Gruppenmitglieder einsehen . . . 31
2.4 Aktuellen Stundenzettel einsehen . . . 32
2.5 Stundenzettel abgeben . . . 33
2.6 Benutzer anlegen . . . 34
2.7 Nachrichten verschicken . . . 35
2.8 Administrator exportiert alle Stundenzettel . . . 36
2.9 Alle Stundenzettel mit Zustand unlocked prüfen . . . 37
2.10 Prüfung eines Stundenzettels auf Rechtskonformität . . . 38

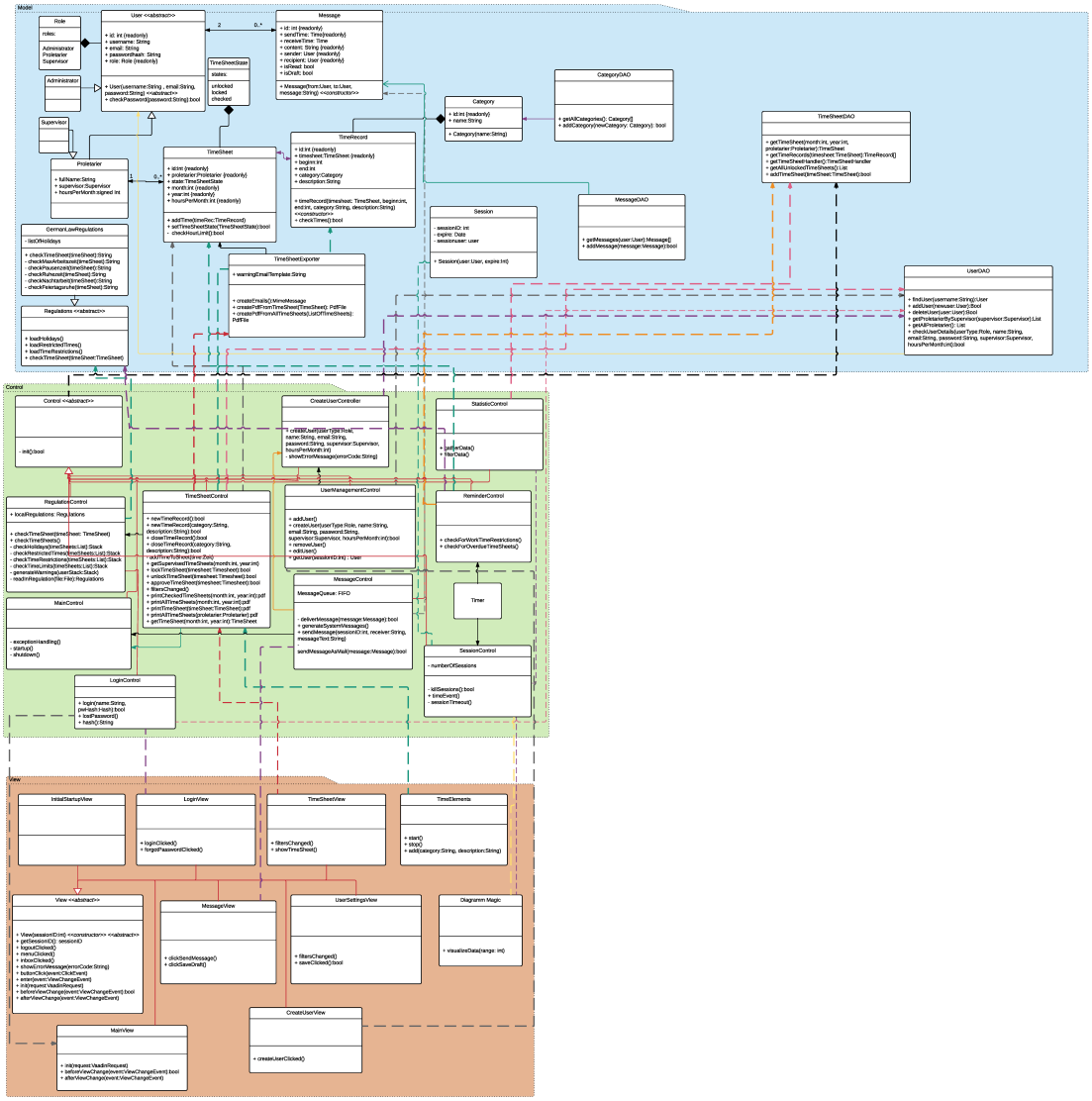
3 Datenbankdesign **40**
3.1 ER Diagramm . . . 40

4 Dateiformate **40**
4.1 Dateiformat zur Definition von gesetzlichen Feiertagen . . . 40
4.2 Datenbankanbindung . . . 40
4.3 Authentifizierung via LDAP . . . 40

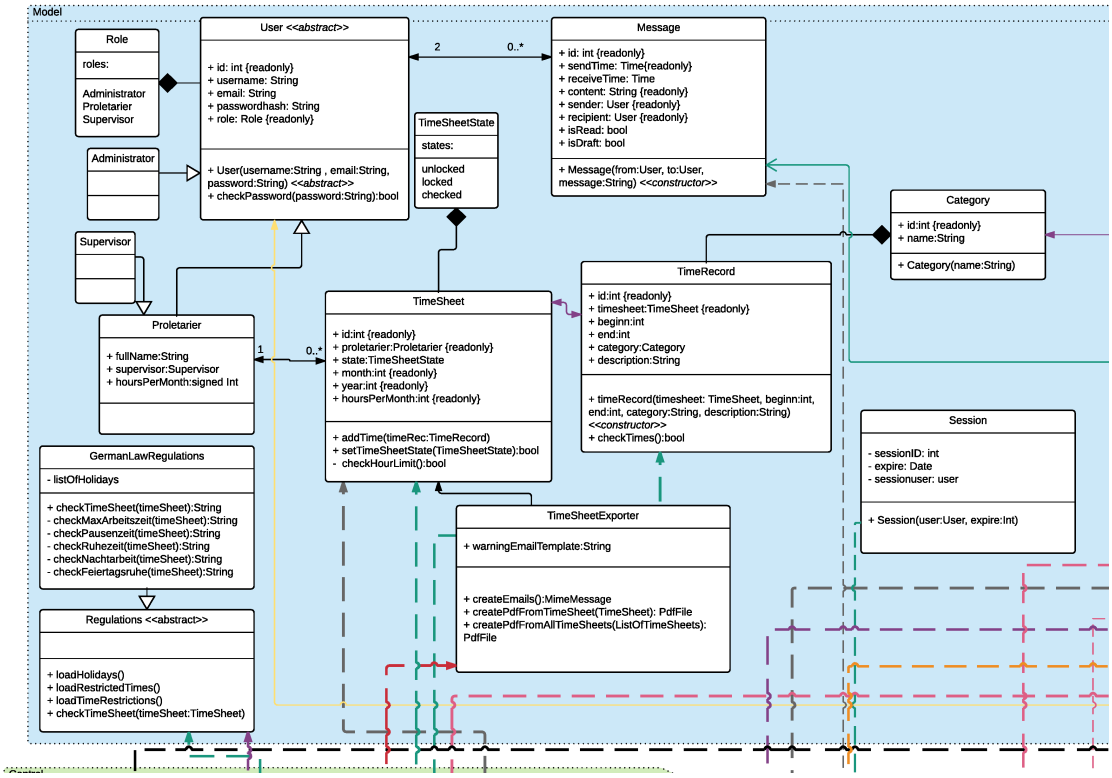
5 Glossar **41**

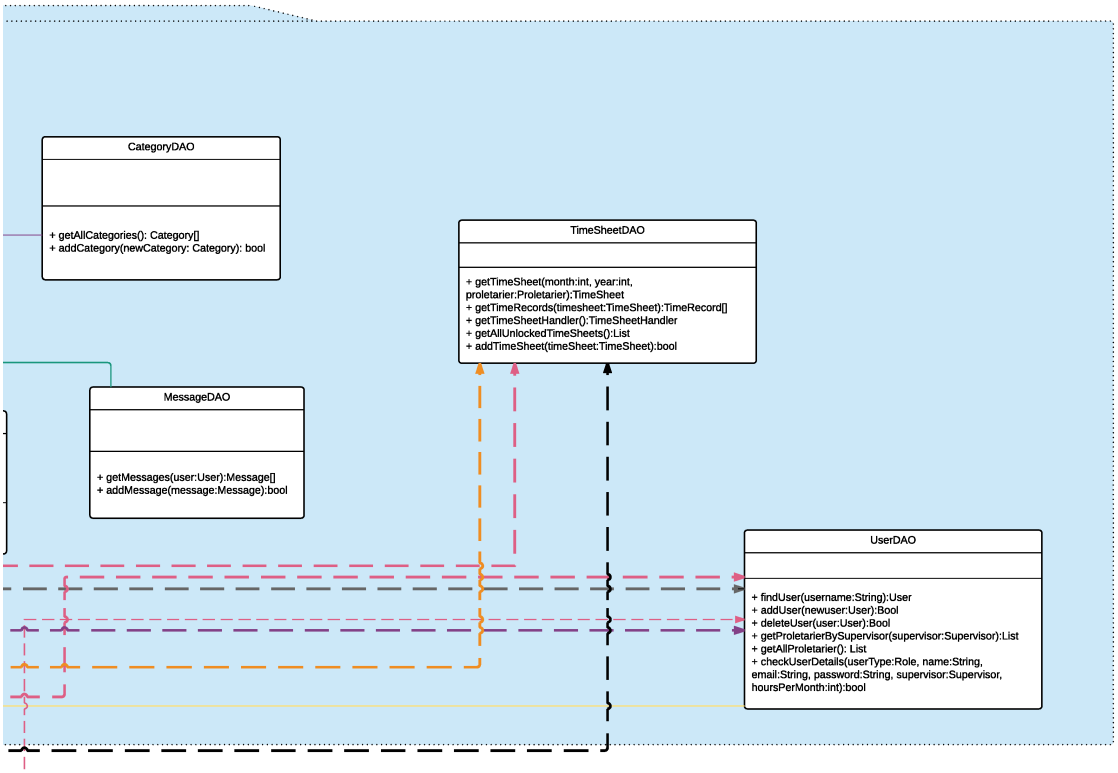
1 Klassen


Model
Control
View
Role
User
<<abstract>>
2
0..*
MessageDAO
Message
GermanLawRegulations
Regulations
<<abstract>>
StatisticControl
TimeSheetDAO
ReminderControl
Proletarier
1
Administrator
TimeSheetExporter
Category
CategoryDAO
Timer
DiagrammMagic
Supervisor
0..*
TimeRecord
Session
SessionControl
TimeSheet
Control
<<abstract>>
UserDAO
RegulationControl
TimeSheetControl
TimeSheetState
MessageControl
MainControl
UserManagementControl
LoginControl
CreateUserController
LoginView
MainView
MessageView
InitialStartupView
TimeElements
TimeSheetView
UserSettingsView
CreateUserView
View
<<abstract>>

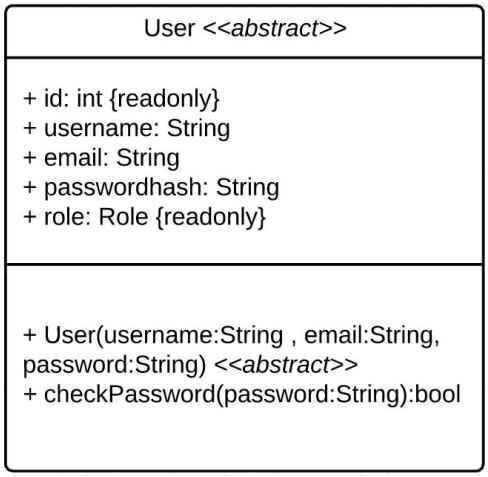

Model
Role
roles:
Administrator
Proletarier
Supervisor
User <<abstract>>
+ id: int (readonly)
+ username: String
+ email: String
+ passwordHash: String
+ role: Role (readonly)
+ User(username:String , email:String, password:String) <<abstract>>
+ checkPassword(password:String):bool
2
0..*
Message
+ id: int (readonly)
+ sendTime: Time(readonly)
+ receiveTime: Time
+ content: String (readonly)
+ sender: User (readonly)
+ recipient: User (readonly)
+ isRead: bool
+ isDraft: bool
+ Message(from:User, to:User, message:String) <<constructor>>
TimeSheetState
states:
unlocked
locked
checked
Administrator
Supervisor
Proletarier
+ fullName:String
+ supervisor:Supervisor
+ hoursPerMonth:signed int
1
0..n
TimeSheet
+ id:int (readonly)
+ proletarier:Proletarier (readonly)
+ state:TimeSheetState
+ month:int (readonly)
+ year:int (readonly)
+ hoursPerMonth:int (readonly)
+ addTime(timeRec:TimeRecord)
+ setTimeSheetState(TimeSheetState):bool
- checkHourLimit():bool
TimeRecord
+ id:int (readonly)
+ timesheet:TimeSheet (readonly)
+ begin:int
+ end:int
+ category:Category
+ description:String
+ timeRecord(timesheet: TimeSheet, begin:int, end:int, category:String, description:String) <<constructor>>
+ checkTime():bool
Category
+ id:int (readonly)
+ name:String
+ Category(name:String)
CategoryDAO
+ getAllCategories():Category[]
+ addCategory(newCategory: Category): bool
TimeSheetDAO
+ getTimeSheet(month:int, year:int, proletarier:Proletarier):TimeSheet
+ getTimeRecords(timesheet:TimeSheet):TimeRecord[]
+ getTimeSheetHandle():TimeSheetHandler
+ getAllUnlockedTimeSheets():List
+ addTimeSheet(timeSheet:TimeSheet):bool
Session
- sessionID: int
- expire: Date
- sessionuser: user
+ Session(user:User, expire:Int)
MessageDAO
+ getMessages(user:User):Message[]
+ addMessage(message:Message):bool
GermanLawRegulations
- listOfHolidays
+ checkTimeSheet(timeSheet):String
- checkRestzeitbasist(timeSheet):String
- checkPausenzeit(timeSheet): String
- checkRuhezeit(timeSheet):String
- checkNachtarbeit(timeSheet):String
- checkFeiertagsruhe(timeSheet):String
Regulations <<abstract>>
+ loadHolidays()
+ loadRestzeitTimes()
+ loadTimeRestrictions()
+ checkTimeSheet(timeSheet:TimeSheet)
TimeSheetExporter
+ warningEmailTemplate:String
+ createEmails():MimeMessage
+ createPdfFromTimeSheet(TimeSheet): PdfFile
+ createPdfFromAllTimeSheets(List(TimeSheets): PdfFile
UserDAO
+ findUser(username:String):User
+ addUser(newuser:User):Bool
+ deleteUser(user:User):bool
+ getProletarierBySupervisor(supervisor:Supervisor):List
+ getAllProletarier():List
+ checkUserDetails(userType:Role, name:String, email:String, password:String, supervisor:Supervisor, hoursPerMonth:int):bool
Control
Control <<abstract>>
- init():bool
CreateUserController
+ createUserPressed(userType:Role, name:String, email:String, password:String, supervisor:Supervisor, hoursPerMonth:int)
- showErrorMessage(errorCode:String)
StatisticControl
+ getRawData()
+ filterData()
RegulationControl
+ localRegulations: Regulations
+ checkTimeSheet(timeSheet: TimeSheet)
+ checkTimeSheets()
- checkHolidays(timeSheets:List):Stack
- checkRestzeit(TimeSheetSheets:List):Stack
- checkTimeRestrictions(timeSheets:List):Stack
- checkTimeLimits(timeSheets:List):Stack
- getRestzeitTimes(userStats:Stack)
- readFromRegulationFile(File):Regulations
TimeSheetControl
+ newTimeRecord():bool
+ addNewTimeRecord(category:String, description:String):bool
+ closeTimeRecord():bool
+ startTimeRecord(category:String, description:String):bool
+ updateTimeRecord(time:Date)
+ getSupervisedTimeSheets(month:int, year:int)
+ lockTimeSheet(timesheet:TimeSheet):bool
+ unlockTimeSheet(timesheet: Timesheet):bool
+ approveTimeSheet(timesheet:Timesheet):bool
+ filterChanged()
+ printCheckedTimeSheets(month:int, year:int):pdf
+ printAllTimeSheets(month:int, year:int):pdf
+ printTimeSheet(timesheet:TimeSheet):pdf
+ printAllTimeSheets(proletarier:Proletarier):pdf
+ getTimeSheet(month:int, year:int):TimeSheet
UserManagementControl
+ addUser()
+ createUser(userType:Role, name:String, email:String, password:String, supervisor:Supervisor, hoursPerMonth:int):bool
+ removeUser()
+ editUser()
+ getUserBy(emailOrUsr) : User
ReminderControl
+ checkForWorkTimeRestrictions()
+ checkForOverdueTimeSheets()
Timer
MessageControl
MessageQueue: FIFO
- deliverMessage(message:Message):bool
+ generateSystemMessages()
+ sendMessage(sessionID:int, receiver:String, messageText:String)
- sendMessageAsMail(message:Message):bool
SessionControl
- numberOfSessions
- killSessions():bool
+ timeEvent()
- sessionTimeout()
MainControl
- exceptionHandling()
- startup()
- shutdown()
LoginControl
+ login(name:String, pwHash:Hash):bool
+ lostPassword()
+ hash():String
View
InitialStartupView
LoginView
+ loginClicked()
+ forgotPasswordClicked()
TimeSheetView
+ filterChanged()
+ showTimeSheet()
TimeElements
+ start()
+ stop()
+ add(category:String, description:String)
View <<abstract>>
+ View(sessionID:int) <<constructor>> <<abstract>>
+ getSessionID(): sessionID
+ logoutClicked()
+ menuClicked()
+ inboxClicked()
+ showErrorMessage(errorCode:String)
+ buttonClick(event:ClickEvent)
+ enter(event:ViewChangeEvent)
+ init(request:VaadinRequest)
+ beforeViewChange(event:ViewChangeEvent):bool
+ afterViewChange(event:ViewChangeEvent)
MessageView
+ clickSendMessage()
+ clickSaveDraft()
UserSettingsView
+ filterChanged()
+ saveClicked():bool
Diagramm Magic
+ visualizeData(range: int)
MainView
+ init(request:VaadinRequest)
+ beforeViewChange(event:ViewChangeEvent):bool
+ afterViewChange(event:ViewChangeEvent)
CreateUserView
+ createUserClicked()

1.1 Model


Model
Role
roles:
Administrator
Proletarier
Supervisor
User <<abstract>>
+ id: int {readonly}
+ username: String
+ email: String
+ passwordhash: String
+ role: Role {readonly}
+ User(username:String , email:String, password:String) <<abstract>>
+ checkPassword(password:String):bool
2
0..*
Message
+ id: int {readonly}
+ sendTime: Time{readonly}
+ receiveTime: Time
+ content: String {readonly}
+ sender: User {readonly}
+ recipient: User {readonly}
+ isRead: bool
+ isDraft: bool
+ Message(from:User, to:User, message:String) <<constructor>>
TimeSheetState
states:
unlocked
locked
checked
Administrator
Supervisor
Category
+ id:int {readonly}
+ name:String
+ Category(name:String)
TimeRecord
+ id:int {readonly}
+ timesheet:TimeSheet {readonly}
+ beginn:int
+ end:int
+ category:Category
+ description:String
+ timeRecord(timesheet: TimeSheet, beginn:int, end:int, category:String, description:String) <<constructor>>
+ checkTimes():bool
Proletarier
+ fullName:String
+ supervisor:Supervisor
+ hoursPerMonth:signed Int
1
0..*
TimeSheet
+ id:int {readonly}
+ proletarier:Proletarier {readonly}
+ state:TimeSheetState
+ month:int {readonly}
+ year:int {readonly}
+ hoursPerMonth:int {readonly}
+ addTime(timeRec:TimeRecord)
+ setTimeSheetState(TimeSheetState):bool
- checkHourLimit():bool
Session
- sessionID: int
- expire: Date
- sessionuser: user
+ Session(user:User, expire:Int)
GermanLawRegulations
- listOfHolidays
+ checkTimeSheet(timeSheet):String
- checkMaxArbeitszeit(timeSheet):String
- checkPausenzeit(timeSheet):String
- checkRuhezeit(timeSheet):String
- checkNachtarbeit(timeSheet):String
- checkFeiertagsruhe(timeSheet):String
TimeSheetExporter
+ warningEmailTemplate:String
+ createEmails():MimeMessage
+ createPdfFromTimeSheet(TimeSheet): PdfFile
+ createPdfFromAllTimeSheets(ListOfTimeSheets): PdfFile
Regulations <<abstract>>
+ loadHolidays()
+ loadRestrictedTimes()
+ loadTimeRestrictions()
+ checkTimeSheet(timeSheet:TimeSheet)


CategoryDAO
+ getAllCategories(): Category[]
+ addCategory(newCategory: Category): bool
TimeSheetDAO
+ getTimeSheet(month:int, year:int, proletarier:Proletarier):TimeSheet
+ getTimeRecords(timesheet:TimeSheet):TimeRecord[]
+ getTimeSheetHandler():TimeSheetHandler
+ getAllUnlockedTimeSheets():List
+ addTimeSheet(timeSheet:TimeSheet):bool
MessageDAO
+ getMessages(user:User):Message[]
+ addMessage(message:Message):bool
UserDAO
+ findUser(username:String):User
+ addUser(newuser:User):Bool
+ deleteUser(user:User):Bool
+ getProletarierBySupervisor(supervisor:Supervisor):List
+ getAllProletarier(): List
+ checkUserDetails(userType:Role, name:String, email:String, password:String, supervisor:Supervisor, hoursPerMonth:int):bool

- **`User`**



Grundform eines Benutzers, stellt grundlegende Daten und Funktionen für spezialisierte Benutzer bereit. Es gibt folgende Spezialisierungen: Administrator, Proletarier und Supervisor.

Attribute:

- `id:int`
  Autogenerierte, von der Datenbank vergebene ID. Kann nicht verändert werden.
- `username:String`
  Benutzername der das Einloggen vereinfachen soll. Kann nur vom User eingesehen und geändert werden.
- `email:String`
  Eine Email-Adresse des Users. Wird genutzt um den User zu kontaktieren. Kann nur vom User oder einem Administrator eingesehen und geändert werden.
- `passwordhash:String`
  Ein kryptographischer, gesalzener Hash des Passworts. Kann nur vom User oder einem Administrator geändert und von niemandem eingesehen werden.
- `role:Role`
  Die Rolle des Users. Kann nur implizit vom Programm zwischen User und Supervisor geändert werden, je nachdem ob der User irgendwo als Supervisor eingetragen ist und selbst einen Supervisor hat. Kann vom User und Administratoren eingesehen werden.

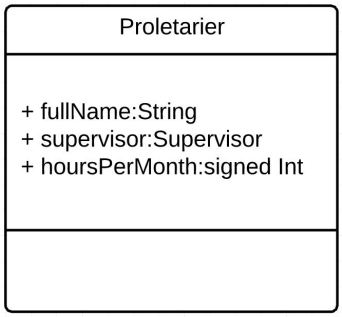
Methoden:

- `User(username:String , email:String, password:Hash)`

- `checkPassword(password:Hash):bool`
  Gibt true zurück wenn das Passwort korrekt ist sonst false.

- **`Administrator`**
  Stellt Daten für und über den Administrator bereit. Der Administrator verwaltet die anderen User, kann jedoch selbst keinen Stundenzettel führen.

- **`Proletarier`**



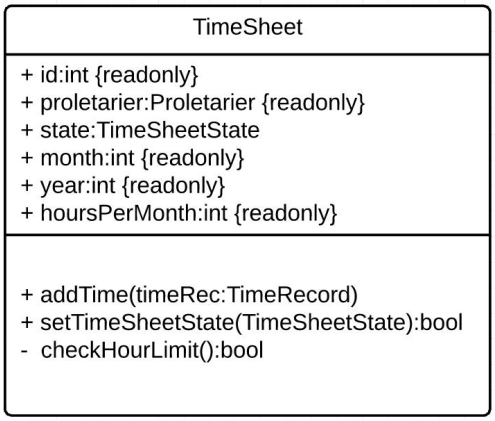
Stellt Daten für und über den Proletarier bereit, darüber hinaus hat der Proletarier eine Verbindung zu dem mit ihm assozierten Zeiterfassungen und Stundenzetteln.

Attribute:
Alle Attribute können nur vom Proletarier, dem entsprechenden Supervisor oder einem Administrator eingesehen werden.

- `fullName:String`
  Der volle Name der später auf dem Stundenzettel steht. Kann nur vom Proletarier oder einem Administrator geändert werden.
- `supervisor:Supervisor`
  Der Supervisor des Proletariers. Kann nur von einem Administrator geändert werden.
- `hoursPerMonth:int`
  Die Stunden die pro Monat gearbeitet werden müssen. Kann nur von einem Administrator geändert werden.

- **`Supervisor`**
  Erweiterung des Proletarier um Gruppen von Usern zu verwalten. Hat sich selbst als Supervisor.

- `TimeSheet`



Speichert Zeiterfassungen. Methoden zur Validierung und Sicherstellung der Unveränderlichkeit sind vorhanden.

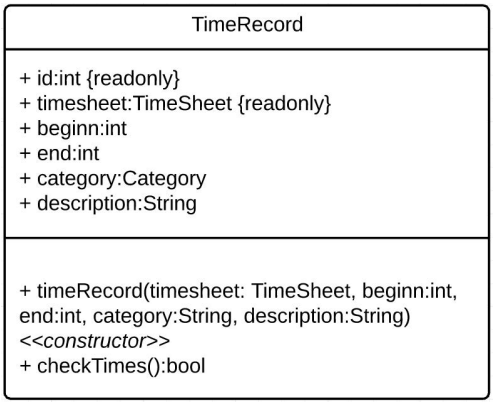
Attribute:

- `id:int`
  Autogenerierte, von der Datenbank vergebene ID. Kann nicht verändert werden.
- `proletarier:Proletarier`
  Gibt an, zu welchem Proletarier dieser Stundenzettel gehört. Wird beim Erstellen gesetzt und kann nicht geändert werden.
- `state:TimeSheetState`
  Gibt den aktuellen Status des Stundenzettels an. Mögliche Zustände sind: unlocked, locked, checked. Kann vom Proletarier oder dessen Supervisor von unlocked zu locked geändert werden. Kann nur vom zugehörigen Supervisor von locked zu unlocked oder checked verändert werden. Kann nur von Administratoren von checked in etwas anderes geändert werden.
- `month:int`
  Gibt den Monat an, für den der Stundenzettel gilt. Wird beim Erstellen gesetzt und kann nicht geändert werden.
- `year:int`
  Gibt das Jahr an, für das der Stundenzettel gilt. Wird beim Erstellen gesetzt und kann nicht geändert werden.
- `hoursPerMonth:int`
  Gibt an, wieviele Stunden in diesem Monat gearbeitet werden müssen. Wird beim Erstellen gesetzt und kann nicht geändert werden.

Methoden:

- `addTime(timeRec:TimeRecord)`
- `setTimeSheetState(TimeSheetState):bool`
- `checkHourLimit():bool`

- **`TimeRecord`**



Datenhaltung, der Zeiterfassung. *Attribute:*

- `id:int`
  Autogenerierte, von der Datenbank vergebene ID. Kann nicht verändert werden.
- `timesheet:TimeSheet`
  Gibt an, zu welchem Stundenzettel diese Zeiterfassung gehört. Wird beim Erstellen gesetzt und kann nicht geändert werden.
- `beginn:int`
  Gibt an, zu welchem Zeitpunkt die Zeiterfassung gestartet wurde. Die Zeit wird als Unix time gespeichert und auf Minuten gerundet. Kann nur vom Proletarier verändert werden. Kann nur vom Proletarier, dem entsprechenden Supervisor und Administratoren eingesehen werden.
- `end:int`
  Gibt an, zu welchem Zeitpunkt die Zeiterfassung beendet wurde. Die Zeit wird als Unix time gespeichert und auf Minuten gerundet. Kann nur vom Proletarier verändert werden. Kann nur vom Proletarier, dem entsprechenden Supervisor und Administratoren eingesehen werden.
- `category:Category`
  Gibt an, in welche Kategorie die Tätigkeit fällt. Kann nur vom Proletarier ver-

ändert werden. Kann nur vom Proletarier, dem entsprechenden Supervisor und Administratoren eingesehen werden.

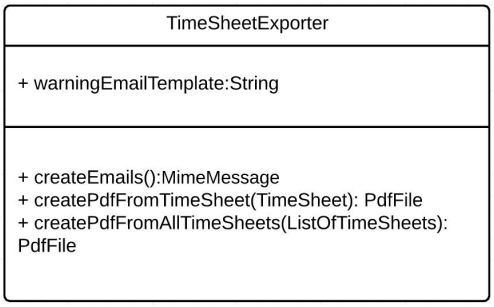
- `description:String`
  Gibt eine kurze Beschreibung der Tätigkeit an. Kann nur vom Proletarier verändert werden. Kann nur vom Proletarier, dem entsprechenden Supervisor und Administratoren eingesehen werden.

Methoden:

- `timeRecord(timesheet: TimeSheetbeginn:int, end:int, category:String, description:String)`

- `checkTimes():bool`

• TimeSheetExporter



Attribute:

- `warningEmailTemplate:String`
  Die Vorlage für die Email die verschickt wird, wenn Proletarier ihren Stundenzettel des letzten Monats noch nicht fertiggestellt haben. Kann nur von Administratoren eingesehen und geändert werden.

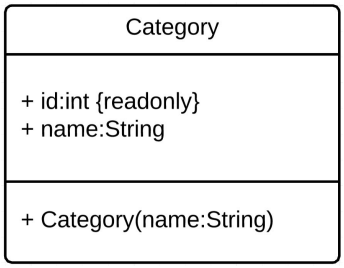
Methoden:

- `setTimeSheetState(TimeSheet, TimeSheetState):bool`

- `createPdfFromTimeSheet(TimeSheet): PdfFile`

- `createPdfFromAllTimeSheets(ListOfTimeSheets): PdfFile`

- **`Category`**



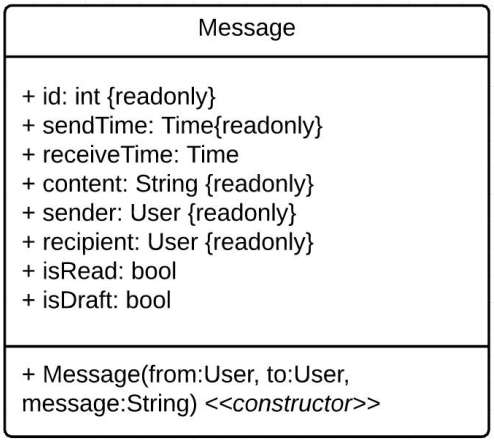
Kategorie, die für die Zeiterfassung benötigt wird *Attribute:*

- `id:int`
  Autogenerierte, von der Datenbank vergebene ID. Kann nicht verändert werden.
- `name:String`
  Der Name einer Kategorie. Kann von allen eingesehen, jedoch nur von Administratoren geändert werden.

Methoden:

- `Category(name:String)`

- **`Message`**



Nachrichten und damit verbundene Metadaten können mit dieser Klasse erfasst werden. *Attribute:*

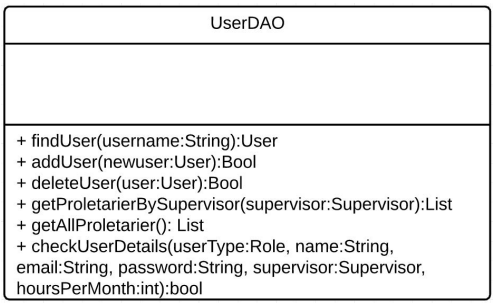
- `id:int`
  Autogenerierte, von der Datenbank vergebene ID. Kann nicht verändert werden.

- `sendTime:int`
  Die Uhrzeit zu der die Nachricht gesendet wurde. Die Zeit wird als Unix time gespeichert und auf Minuten gerundet. Wird beim Erstellen gesetzt und kann nicht geändert werden.
- `receiveTime:int`
  Die Uhrzeit zu der die Nachricht vom Empfänger gelesen wurde. Die Zeit wird als Unix time gespeichert und auf Minuten gerundet. Kann nur vom empfangenden Proletarier eingesehen und geändert werden.
- `content:String`
  Der Inhalt der Nachricht. Wird beim Erstellen gesetzt und kann nicht geändert werden.
- `sender:User`
  Der User der die Nachricht schickt oder in dessen Namen die Nachricht geschickt wird. Dieses Feld kann leer sein, falls das System die Nachricht geschickt hat. Wird beim Erstellen gesetzt und kann nicht geändert werden.
- `recipient:User`
  Der Empfänger der Nachricht. Wird beim Erstellen gesetzt und kann nicht geändert werden.
- `isRead:bool`
  Ob der Empfänger die Nachricht bereits gelesen hat. Kann nur vom empfangenden Proletarier eingesehen und geändert werden.

Methoden:

- `Message(from:User, to:User, message:String)`

- **`UserDAO`**

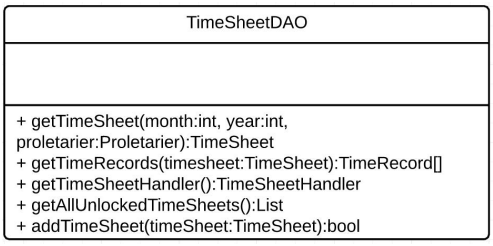


Listet alle vorhanden User auf. Enthält Methoden um User hinzuzufügen, zu löschen oder zu verändern. Stellt darüber hinaus sicher, dass die alle Einträge einzigartig sind. *Methoden:*

- `findUser(username:String):User`

- `addUser(newEntity:User):Bool`
  Gibt true zurück wenn die User erstellt werden konnte. False bei Fehler.
- `deleteUser(User:User):Bool`
  Gibt true zurück wenn die User gelöscht werden konnte. False bei Fehler.
- `checkCredentials(userName:String, pwHash:Hash):Bool`
  Gibt true zurück wenn Username und Passwort korrekt sind, sonst false.
- `getProletarierBySupervisor(supervisor:Supervisor):List`
- `getAllProletarier(): List`
- `checkUserDetails(userType:Role, name:String, email:String, password:String, supervisor:Supervisor, hoursPerMonth:int):bool`

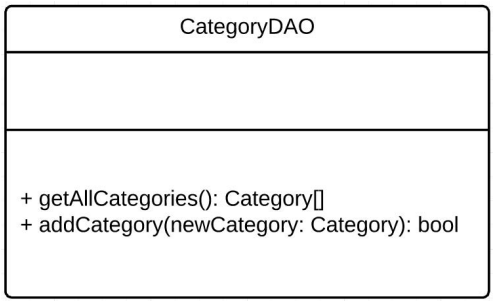
- **`TimeSheetDAO`**



Listet alle TimeSheets. *Methoden:*

- `getTimeSheet(month:int, year:int, proletarier:Proletarier):TimeSheet`
- `getTimeRecords(timesheet:TimeSheet):TimeRecord[]`
- `getTimeSheetHandler():TimeSheetHandler`
- `getAllUnlockedTimeSheets():List`
- `addTimeSheet(timeSheet:TimeSheet):bool`

- `CategorieDAO`

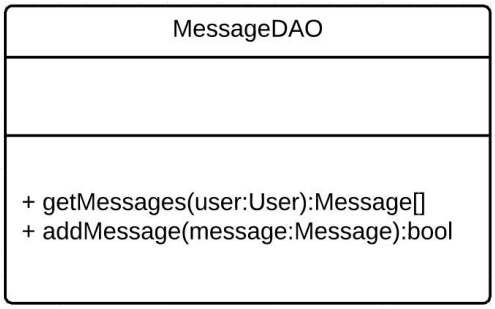


Liste aller verfügbaren Kategorien. *Methoden:*

  - `getAllCategories(): Category[]`

  - `addCategory(newCategory: Category): bool`

- `MessageDAO`

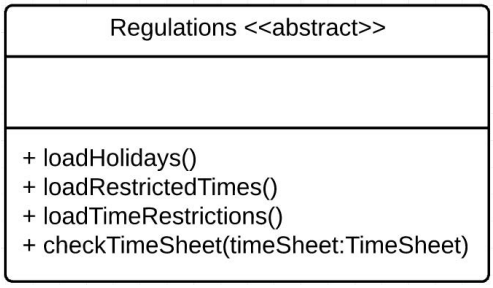


Nachrichten und damit Verbundene Metadaten können mit dieser Klasse erfasst werden. *Methoden:*

  - `getMessages(user:User):Message[]`

  - `addMessage(message:Message):bool`

- **`Regulations`**



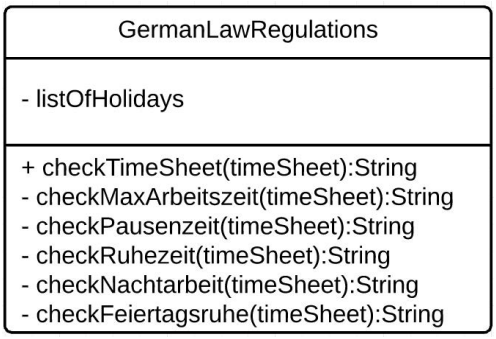
Lädt die gesetzlichen Regularien aus einer Datei und stellt diese für andere Klassen bereit. *Methoden:*

  - `loadHolidays()`

  - `loadRestrictedTimes()`

  - `loadTimeRestrictions()`

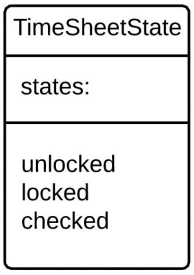
- **`GermanLawRegulations`**



Lädt die gesetzlichen Regularien aus einer Datei und stellt diese für andere Klassen bereit.
Attribute:

  - `listOfHolidays`

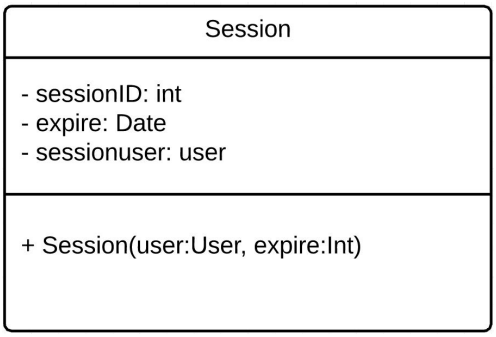
- **`TimeSheetState`**



Zustände, die beschreiben, ob ein Stundenzettel verändert werden darf, bzw bereits überprüft wurde. Mögliche Zustände:

  - `unlocked`
    Der Time Sheet kann bearbeitet werden.
  - `locked`
    der Time Sheet ist abgeben worden und es deshalb gegen bearbeitung gesperrt.
  - `checked`
    Der Time Sheet wurde vom Supervisor überprüft.

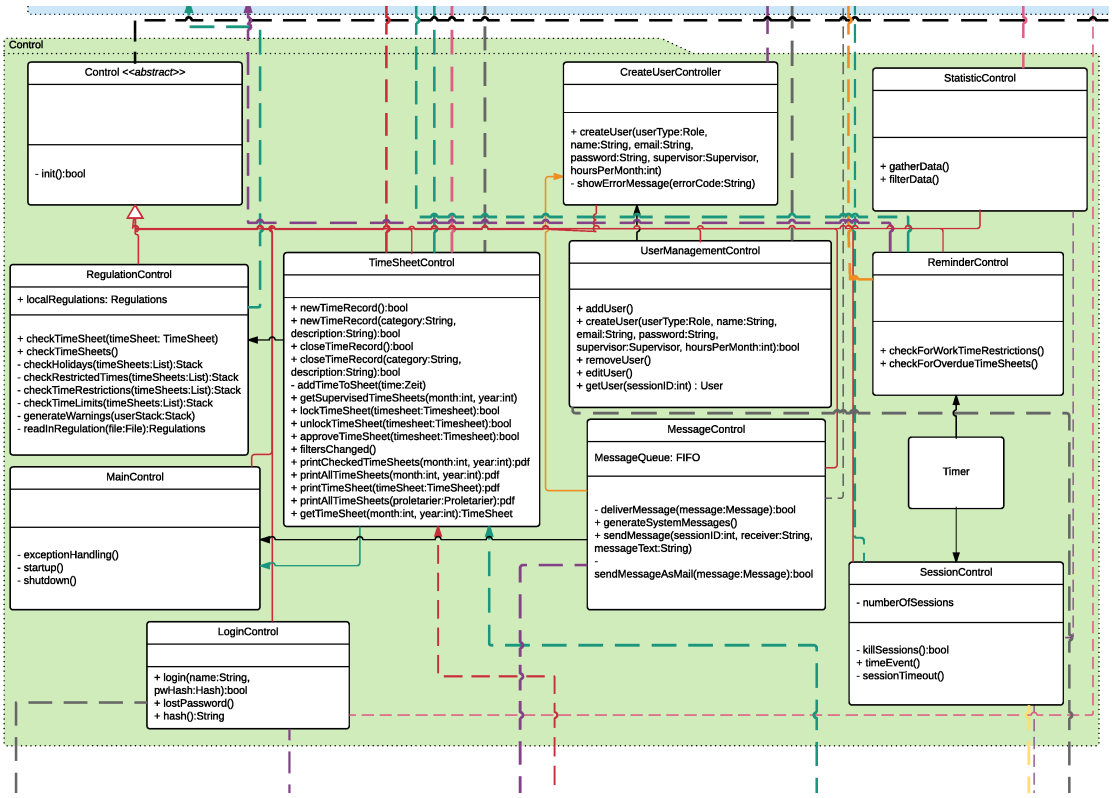
- **`Session`**



Daten die mit einer Login Session assoziert sind(Proletarier, Ablaufdatum, ...) werden in dieser Klasse gespeichert.

  - `Session(User:User, expire:Int)`

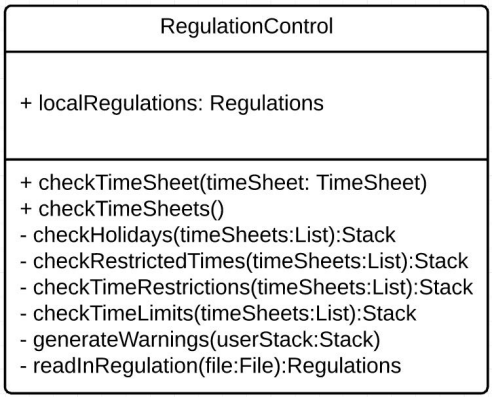
1.2 Control


Control
Control <<abstract>>
- init():bool
CreateUserController
+ createUser(userType:Role, name:String, email:String, password:String, supervisor:Supervisor, hoursPerMonth:int)
- showErrorMessage(errorCode:String)
StatisticControl
+ gatherData()
+ filterData()
RegulationControl
+ localRegulations: Regulations
+ checkTimeSheet(timeSheet: TimeSheet)
+ checkTimeSheets()
- checkHolidays(timeSheets:List):Stack
- checkRestrictedTimes(timeSheets:List):Stack
- checkTimeRestrictions(timeSheets:List):Stack
- checkTimeLimits(timeSheets:List):Stack
- generateWarnings(userStack:Stack)
- readInRegulation(file:File):Regulations
TimeSheetControl
+ newTimeRecord():bool
+ newTimeRecord(category:String, description:String):bool
+ closeTimeRecord():bool
+ closeTimeRecord(category:String, description:String):bool
- addTimeToSheet(time:Zeit)
+ getSupervisedTimeSheets(month:int, year:int)
+ lockTimeSheet(timesheet:Timesheet):bool
+ unlockTimeSheet(timesheet:Timesheet):bool
+ approveTimeSheet(timesheet:Timesheet):bool
+ filtersChanged()
+ printCheckedTimeSheets(month:int, year:int):pdf
+ printAllTimeSheets(month:int, year:int):pdf
+ printTimeSheet(timeSheet:TimeSheet):pdf
+ printAllTimeSheets(proletarier:Proletarier):pdf
+ getTimeSheet(month:int, year:int):TimeSheet
UserManagementControl
+ addUser()
+ createUser(userType:Role, name:String, email:String, password:String, supervisor:Supervisor, hoursPerMonth:int):bool
+ removeUser()
+ editUser()
+ getUser(sessionID:int) : User
ReminderControl
+ checkForWorkTimeRestrictions()
+ checkForOverdueTimeSheets()
MessageControl
MessageQueue: FIFO
- deliverMessage(message:Message):bool
+ generateSystemMessages()
+ sendMessage(sessionID:int, receiver:String, messageText:String)
- sendMessageAsMail(message:Message):bool
Timer
MainControl
- exceptionHandling()
- startup()
- shutdown()
SessionControl
- numberOfSessions
- killSessions():bool
+ timeEvent()
- sessionTimeout()
LoginControl
+ login(name:String, pwHash:Hash):bool
+ lostPassword()
+ hash():String

- `Control`

- Jede von Control erbende Klasse implementiert das Einzelstück-Entwurfsmuster, da die Control-Klassen keinen Zustand benötigen.
  Methoden:
  - init():bool Das Initialisierungsablauf jeder Klasse

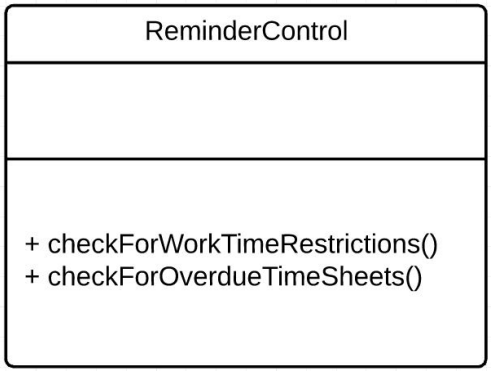
  Erbende Klassen:
  - `RegulationControl`



Überprüft Daten auf Gesetzeskonformität, leitet ebenfalls notwendige Schritte ein.
Methoden:

* `checkRegulations():bool`
  Es werden erfasste Zeiten auf ihre konformität geprüft.
* `checkHolidays(timeSheets:List):Stack`
  Es werden geprüft, ob an Feiertagen gearbeitet wird.
* `checkRestrictedTimes(timeSheets:List):Stack`
* `checkTimeRestrictions(timeSheets:List):Stack`
* `checkTimeLimits(timeSheets:List):Stack`
* `generateWarnings()`
  Erzeugt Messages über Verletzungen der Regularien.
* `readInRegulation(file:File):Regulations`

- `ReminderControl`

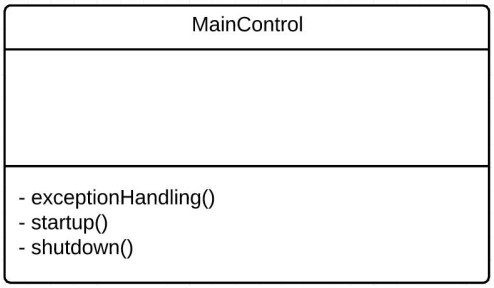


Überprüft in regelmäßigen abständen, ob Erinnerungen versendet werden müssen.
Methoden:

* `checkForWorkTimeRestrictions()`

* `checkForOverdueTimeSheets()`

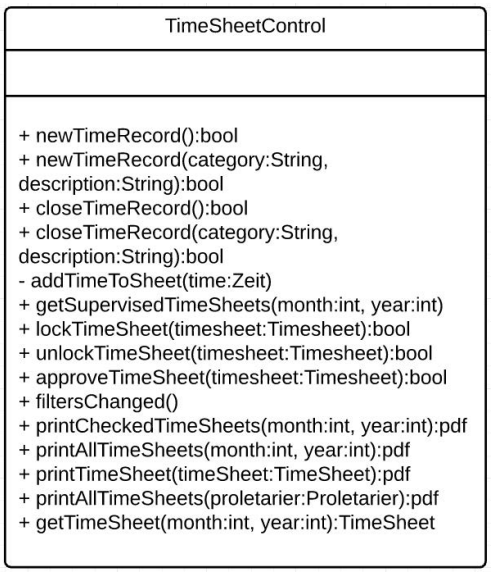
- `MainControl`



Kontrolliert den Hauptablauf des Programms.
Methoden:

* `exceptionHandling()`
  Im Falle von kritischen Excepetions, wird der Dienst gepeichert und runtergefahren.
* `startup()`
  Startroutine des Servers.
* `shutdown()`
  Abschaltverhalten des Servers.

- TimeSheetControl



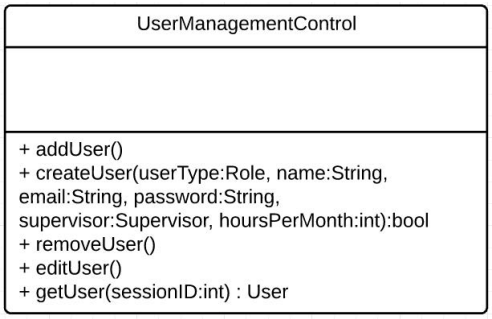
Regelt die Erstellung von Stundendaten für den Stundenzettel, darüber hinaus wird auch die Abarbeitung eines fertigen Stundenzettels geregelt.
Methoden:

* `newTimeRecord():bool`
  Starte eine neue Zeiterfassung OHNE Angabe von Kategorien bzw. Tätigkeiten.
* `newTimeRecord(category:String, description:String):bool`
  Starte eine neue Zeiterfassung MIT Angabe von Kategorien bzw. Tätigkeiten.
* `closeTimeRecord():bool`
  Beende eine Zeiterfassung OHNE Angabe von Kategorien bzw. Tätigkeiten.
* `closeTimeRecord(category:String, description:String):bool`
  Beende eine Zeiterfassung MIT Angabe von Kategorien bzw. Tätigkeiten.
* `addTimeToSheet(time:Zeit)`
  Füge erfasste Zeit in Stundenzettel hinzu.
* `getSupervisedTimeSheets(month:int, year:int)`
  Listet Stundenzettel von einem bestimmten Monat von einem bestimmten Jahr.
* `lockTimeSheet(month:int, year:int):bool`
  Markiere Stundenzettel von einem beestimmten Monat von einem bestimmten Jahr als geprüft und mache sie schreibgeschützt.

* `filtersChanged()`

* `printLockedTimeSheets(month:int, year:int)`
  Drucke alle geprüfte Stundenzettel von einem bestimmten Monat von einem bestimmten Jahr aus.
* `getTimeSheet(month:int, year:int):TimeSheet`
  Listet alle Stundenzettel von einem bestimmten Monat von einem bestimmten Jahr.

- **`UserManagementControl`**



Managed das hinzufügen, löschen und verändern von Benutzern.
Methoden:

* `addUser()`

* `createUser(userType:Role, supervisor: Supervisor)`

* `removeUser()`

* `editUser()`

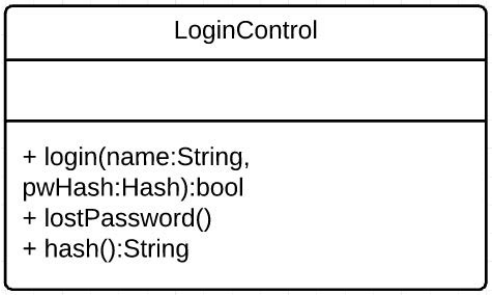
* `getUser(sessionID:int): User`

- **`LoginControl`**

Überwacht das korrekte Einloggen von Benutzern.
Methoden:

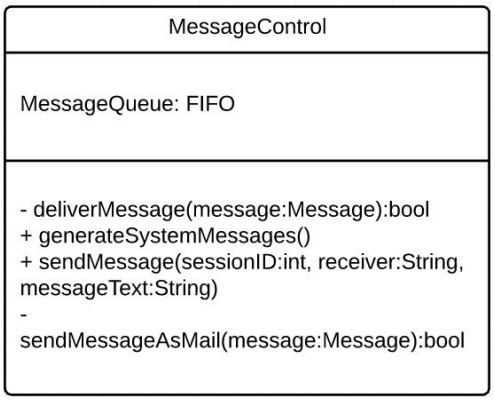
* `login(name:String, pwHash:Hash):bool`

* `lostPassword()`



* `hash():String`

- `MessageControl`



Stellt Nachrichten zwischen Benutzern (und dem System) zu.
Methoden:

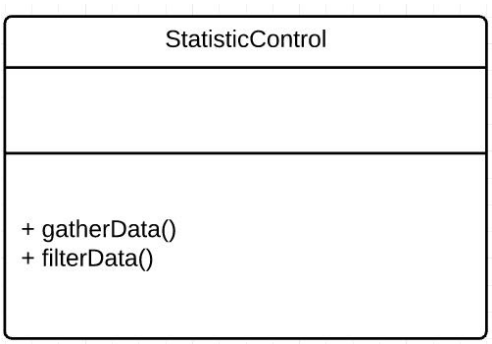
* `deliverMessage(message:Message):bool`

* `generateSystemMessages()`

* `sendMessage()`

* `sendMessageAsMail(message:Message):bool`

- `StatisticControl`

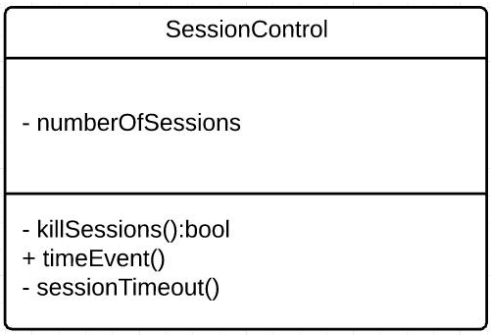


Sammelt und generiert Statistiken über die erfassten Stundendaten.
Methoden:

  * `gatherData()`

  * `filterData()`

- `SessionControl`



Kontrolliert offene Sessions und terminiert diese nach Ablauf eines gesetzen Zeitraums.

  * `killSessions():bool`

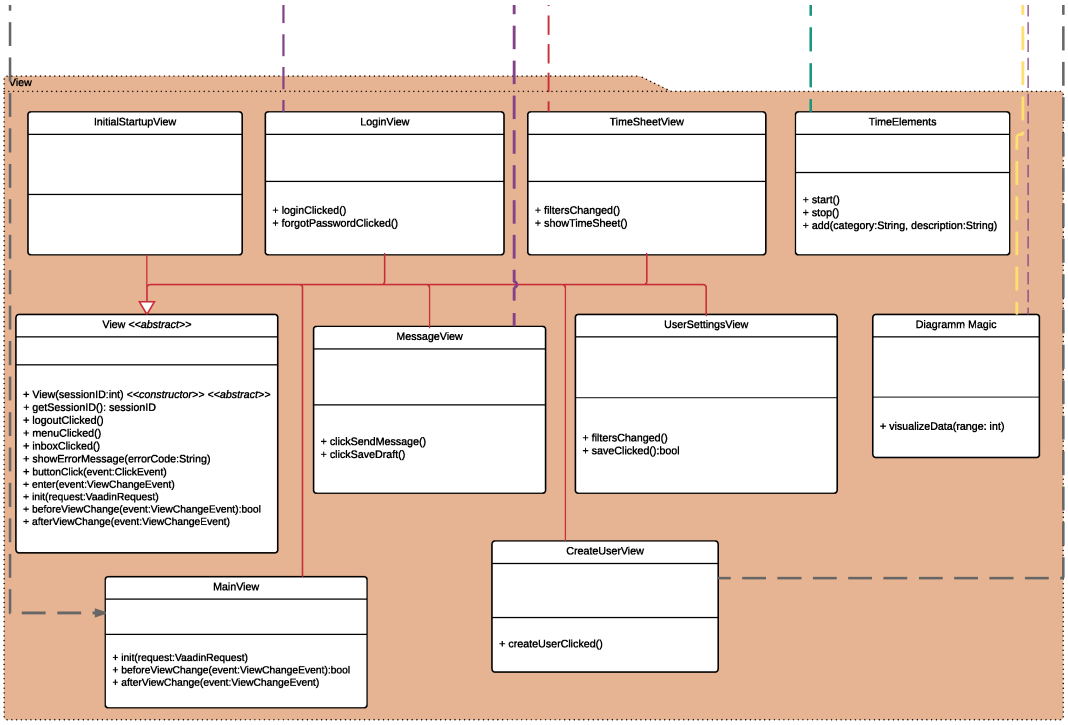
  * `timeEvent()`

  * `sessionTimeout()`

• `Timer`

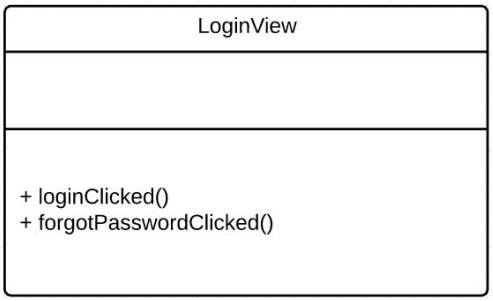
–

1.3 View


View
InitialStartupView
LoginView
+ loginClicked()
+ forgotPasswordClicked()
TimeSheetView
+ filtersChanged()
+ showTimeSheet()
TimeElements
+ start()
+ stop()
+ add(category:String, description:String)
View <<abstract>>
+ View(sessionID:int) <<constructor>> <<abstract>>
+ getSessionID(): sessionID
+ logoutClicked()
+ menuClicked()
+ inboxClicked()
+ showErrorMessage(errorCode:String)
+ buttonClick(event:ClickEvent)
+ enter(event:ViewChangeEvent)
+ init(request:VaadinRequest)
+ beforeViewChange(event:ViewChangeEvent):bool
+ afterViewChange(event:ViewChangeEvent)
MessageView
+ clickSendMessage()
+ clickSaveDraft()
UserSettingsView
+ filtersChanged()
+ saveClicked():bool
Diagramm Magic
+ visualizeData(range: int)
CreateUserView
+ createUserClicked()
MainView
+ init(request:VaadinRequest)
+ beforeViewChange(event:ViewChangeEvent):bool
+ afterViewChange(event:ViewChangeEvent)

- `View`
  Erbende Klassen:
  - `InitialStartupView`

  - `LoginView`

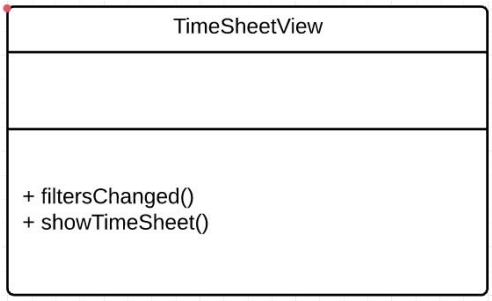


    Die Klasse LoginView stellt die Anmeldeseite bzw. die Seite “Passwort vergessenäuf dem Display dar.

    * `loginPressed()`

    * `forgotPasswordClicked()`

  - `TimeSheetView`

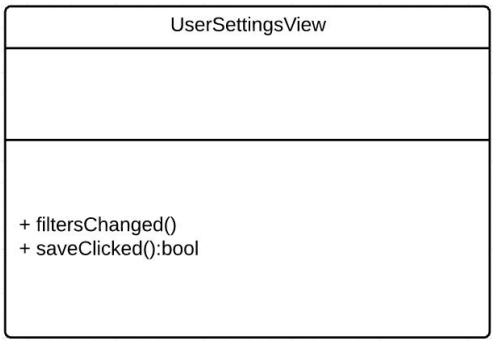


    Die Klasse TimeSheetView stellt die Anzeigen von Stundenzettel dar. Wenn der Filter geändert wird, wird die entsprechende Anzeige ebenfaslls umgesetzt.

    * `filtersChanged()`

    * `showTimeSheet()`

- `UserSettingsView`

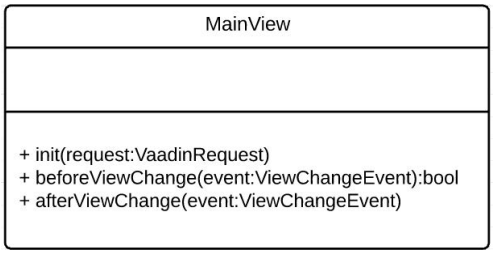


Die Klasse UserSettingsView stellt die Anzeigen bzw. Bearbeitungsseiten von Users dar. Wenn der Filter an User geändert wird, werden entsprechenden User nach der neuen Filterkategorie angezeigt. Nach dem Drücken von SSpeichern"wird ebenfalls eine entsprechende Seite auf dem Diaplay dargestellt.

  * `filtersChanged()`

  * `saveClicked():bool`

- `MainView`



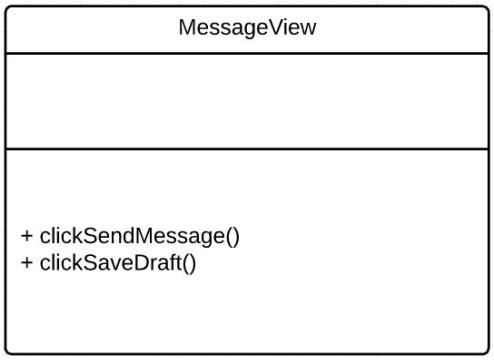
  * `LoadView(sessionID:int)`

- `MessageView`

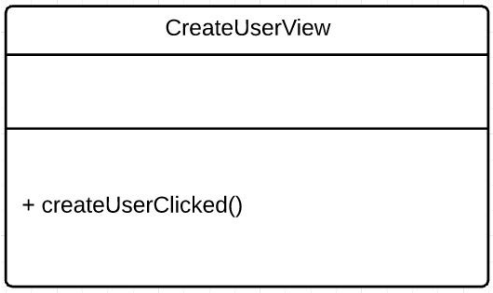
Die Klasse MessageView stellt die Nachrichtenempfangs- verfassungs- bzw. absendeseiten dar.

  * `clickSendMessage()`

  * `clickSaveDraft()`



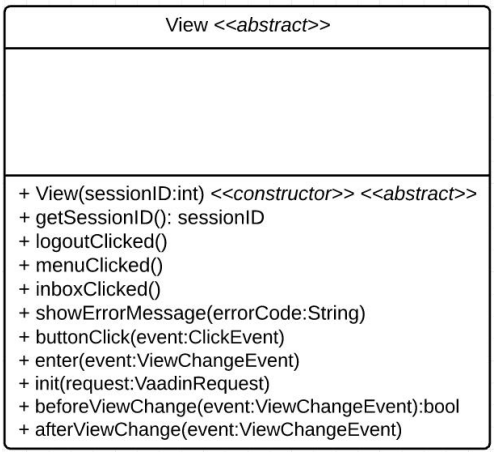
- `CreateUserView`



Die Klasse CreateUserView stellt die Seite beim Anlegen vom neuen Benutzer dar.

  * `createUserClicked()`

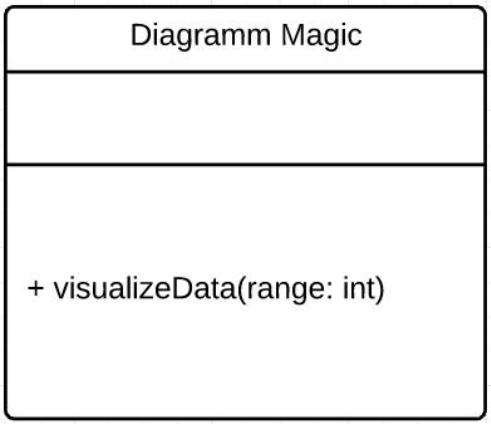
- `View<<abstract>>`



Die abstrakte Klasse fasst die Eigenschaften aller anzuzeigenden GUI-Elemente zusammmen.

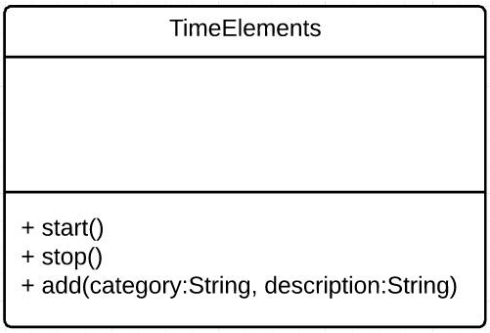
- `View(sessionID:int)`
- `getSessionID(): sessionID`
- `logoutClicked()`
- `menuClicked()`
- `inboxClicked()`
- `showErrorMessage(errorCode:String)`
- `buttonClick(event:ClickEvent)`
- `enter(event:ViewChangeEvent)`
- `init(request:VaadinRequest)`
- `beforeViewChange(event:ViewChangeEvent):bool`
- `afterViewChange(event:ViewChangeEvent)`

- **Diagramm Magic**



- `visualizeData(range: int)`
  Zeigt die Daten aus der StatisticControl an.

- **`TimeElements`**



  - `start()`

  - `stop()`

  - `add(category:String, description:String)`

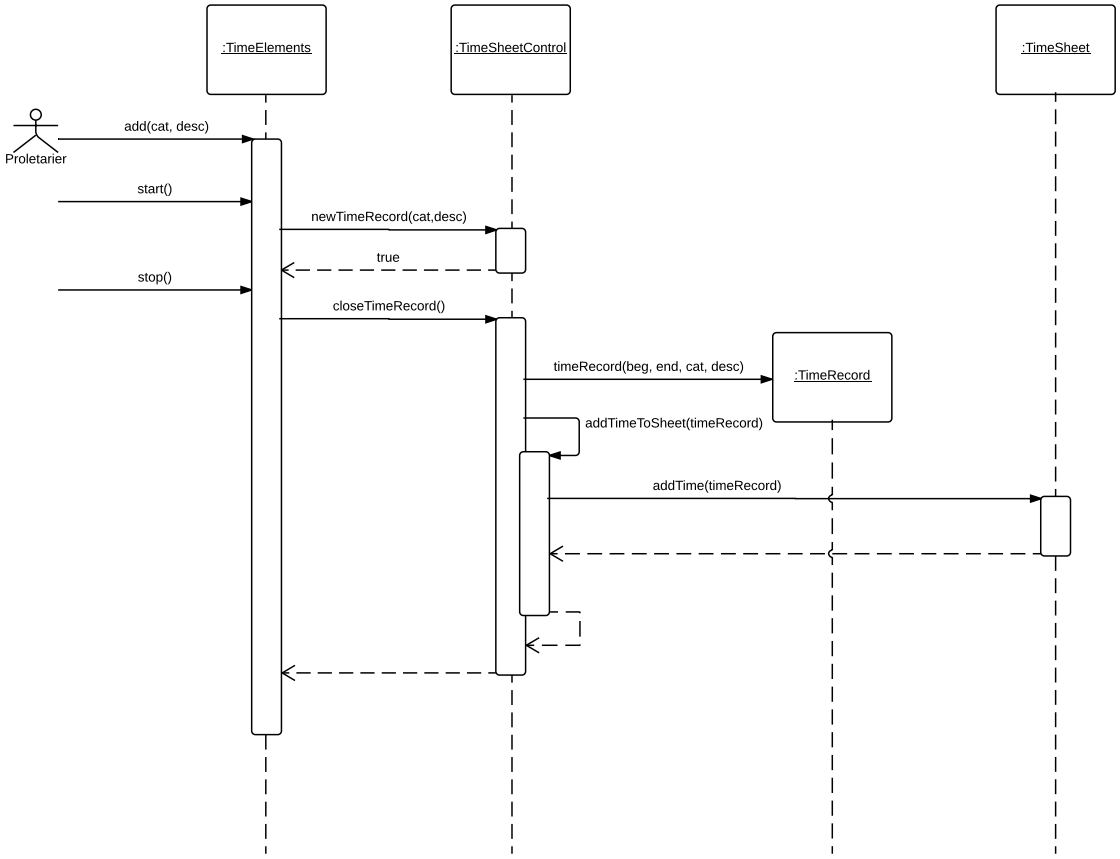
1.4 Entwurfsmuster

- Die Klasse GermanLawRegulations implementiert zusammen mit Regulations und RegulationControl das Strategie-Entwurfsmuster. Dies macht es einfacher, Gesetze von anderen Ländern zu implementieren und zu benutzen.
- Alle auf DAO endenden Klassen implementieren das Fabrik-Entwurfsmuster, um auf in der Datenbank abgelegte Klassen zuzugreifen. Damit wird der Zugriff auf die Datenbank von den anderen Klassen abgekapselt.
- Jede der von Control erbenden Klassen implementiert das Singleton-Entwurfsmuster, da die Klassen keine Zustände haben und nicht bei jeder Benutzung neu initialisiert werden müssen.
- Jede der von View erbenden Klassen bildet zusammen mit View das Schablenenmethode-Entwurfsmuster, da das Vaadin-Framework dies so vorgibt.

2 Sequenzen

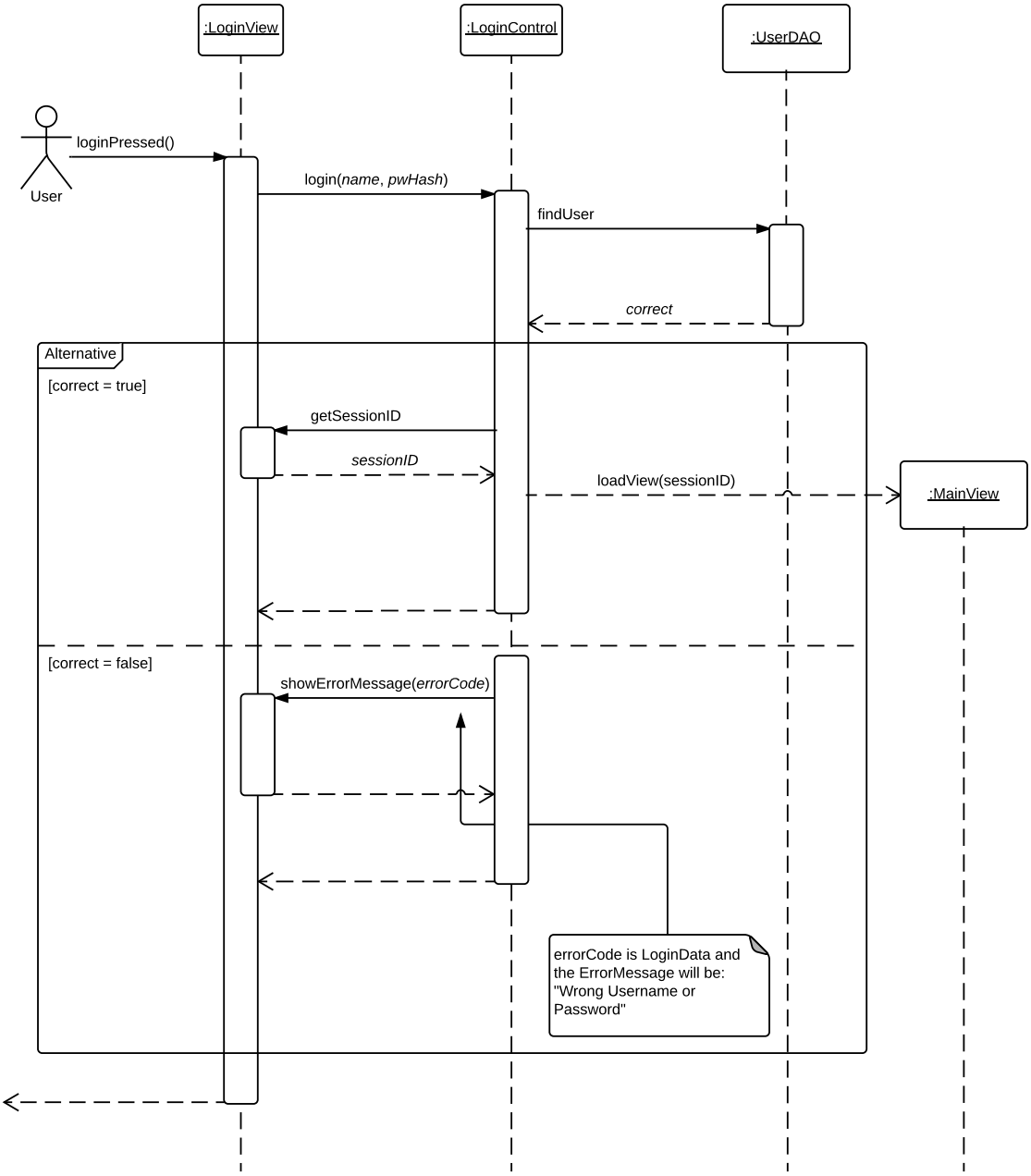
2.1 Neue Zeiterfassung

Eine neue Zeiterfassung wird hinzugefügt. Dies kann entweder durch Übergabe von Start- und Endzeit oder durch das Starten und spätere Stoppen eines Timers geschehen. Falls der Stundenzettel für den entsprechenden Monat noch nicht existiert wird er erstellt. Nur Proletarier und Supervisor dürfen diese Aktion durchführen. Außerdem können nur Zeiterfassungen für Monate hinzugefügt werden, für die der Stundenzettel unlocked ist oder noch nicht existiert.



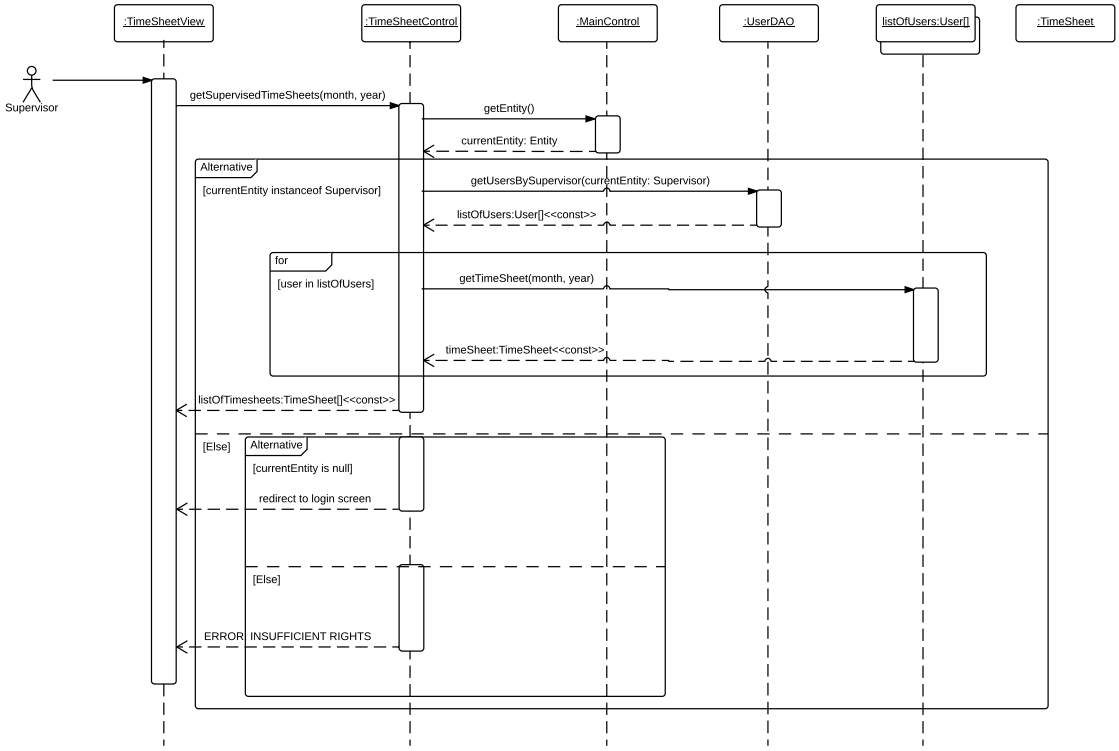
2.2 Login Sequenz

Ein User loggt sich ein. Falls als Authentifizierung LDAP statt Local ausgewählt ist, wird die Anfrage von LoginView stattdessen an den LDAP-Server weitergeleitet. Nur nicht eingeloggte User können diese Aktion durchführen. Für diese ist dies die einzige erlaubte Aktion.



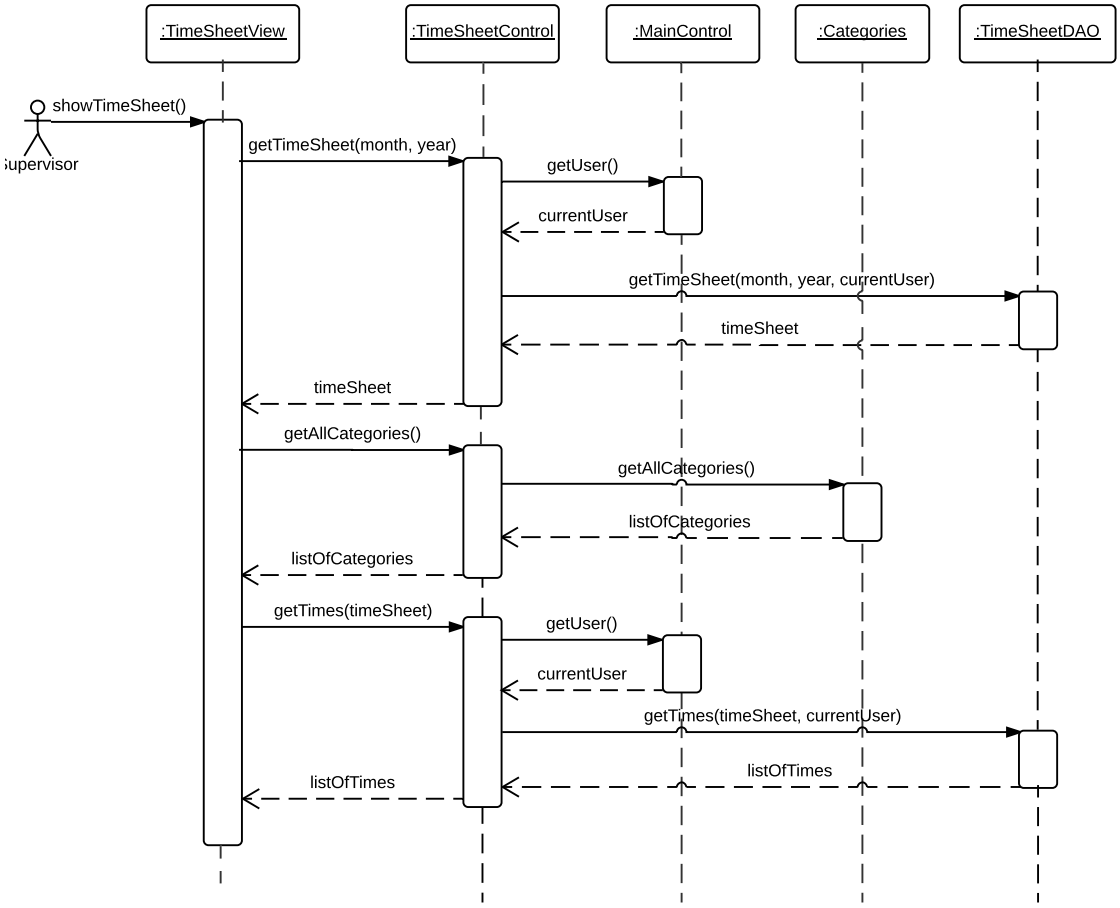
2.3 Alle Stundenzettel der Gruppenmitglieder einsehen

Ein Supervisor lässt sich die Stundenzettel eines beliebigen Monats von allen von ihm betreuten Proletariern anzeigen. Supervisor können sich die Stundenzettel eines Monats von allen oder alle Stundenzettel eines von ihnen betreuten Proletariers anzeigen lassen. Administratoren können sich die Stundenzettel eines Monats aller Proletarier, nur aller von einem Supervisor betreuten Proletarier oder alle Stundenzettel eines Proletariers anzeigen lassen. Proletarier können diese Aktion nicht durchführen.



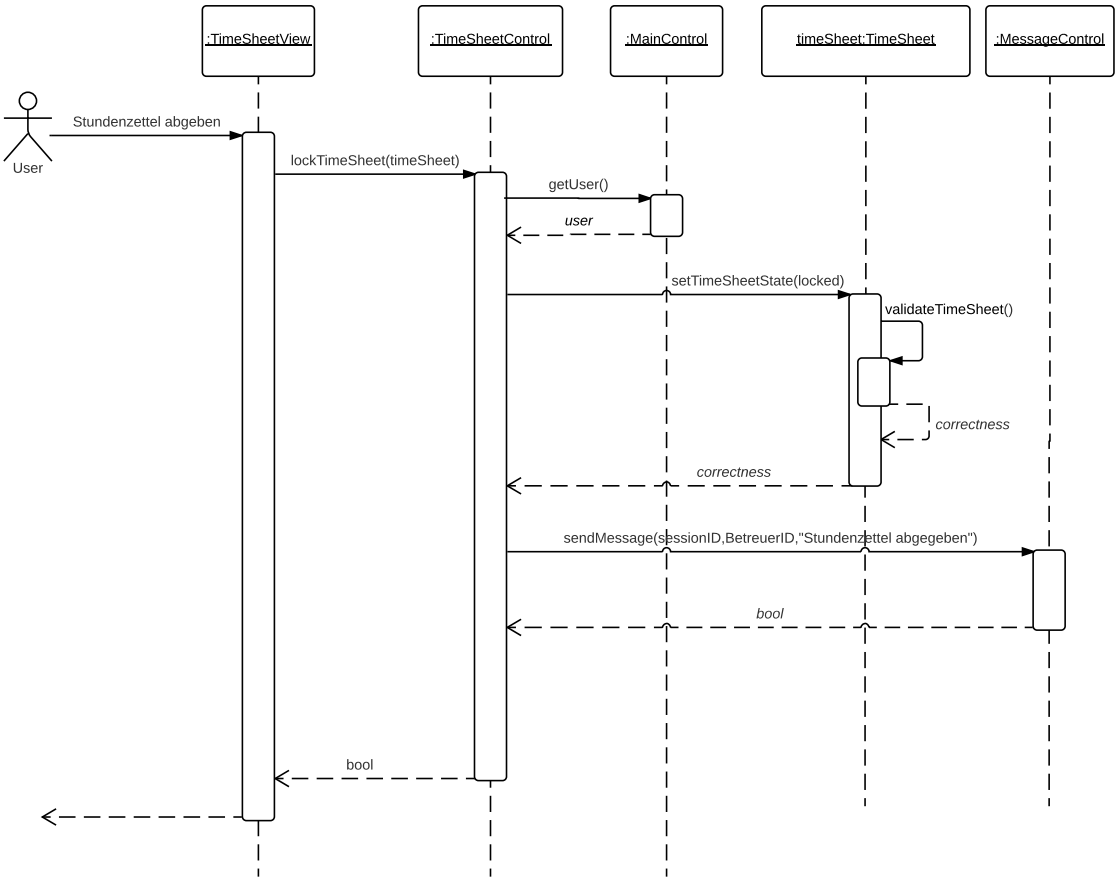
2.4 Aktuellen Stundenzettel einsehen

Ein Supervisor lässt sich den eigenen Stundenzettel des aktuellen Monats und die dazugehörigen Zeiterfassungen anzeigen. Diese Aktion kann für einen beliebigen Monat durchgeführt werden, solange ein Stundenzettel dafür existiert. Diese Aktion kann von Proletariern und Supervisorn durchgeführt werden.



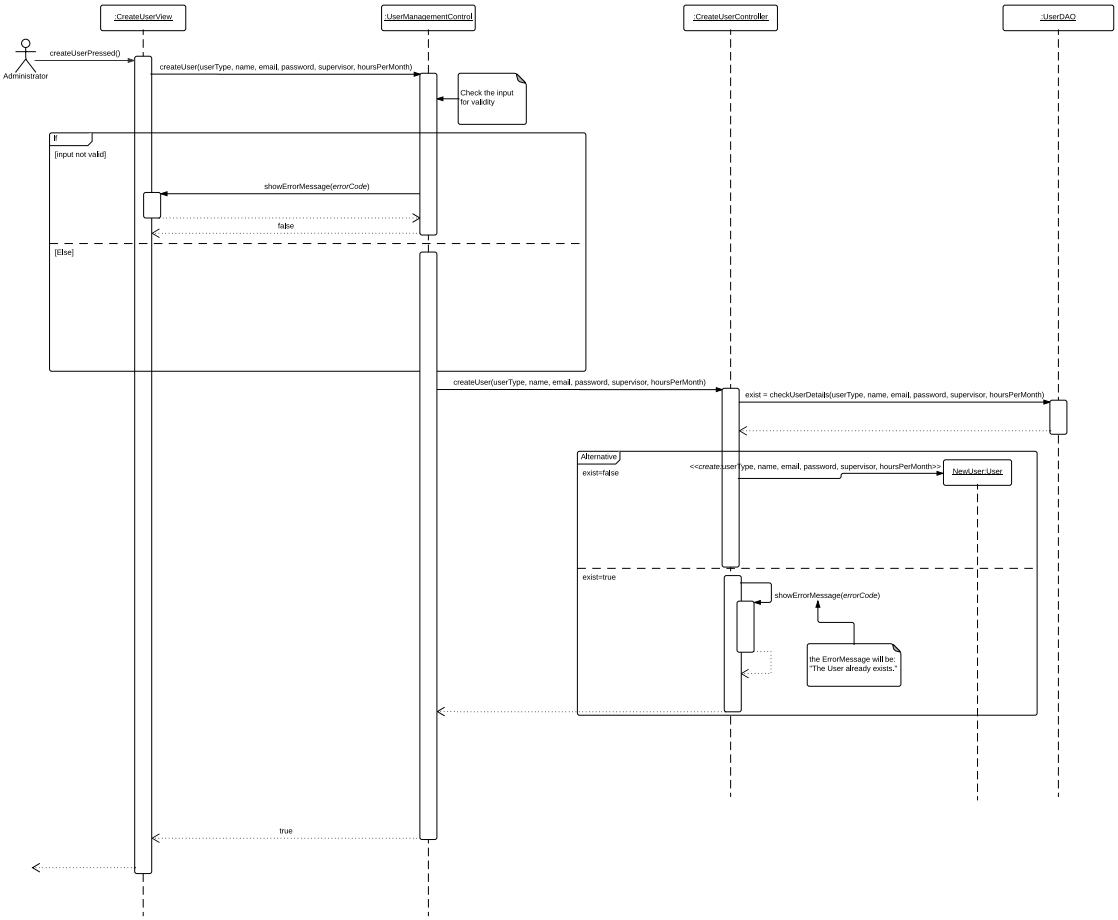
2.5 Stundenzettel abgeben

Ein Proletarier gibt seinen fertigen Stundenzettel ab. Dabei wird der Stundenzettel auf Fehler oder ähnliches überprüft und dann auf locked gesetzt. Dabei wird der Betreuer per Nachricht darüber informiert, damit dieser den Stundenzettel überprüfen kann. Diese Aktion kann von Proletariern und Supervisern ausgeführt werden. Superviser können außerdem Stundenzettel der von ihnen betreuten Proletarier von locked auf unlocked bzw. checked setzen.



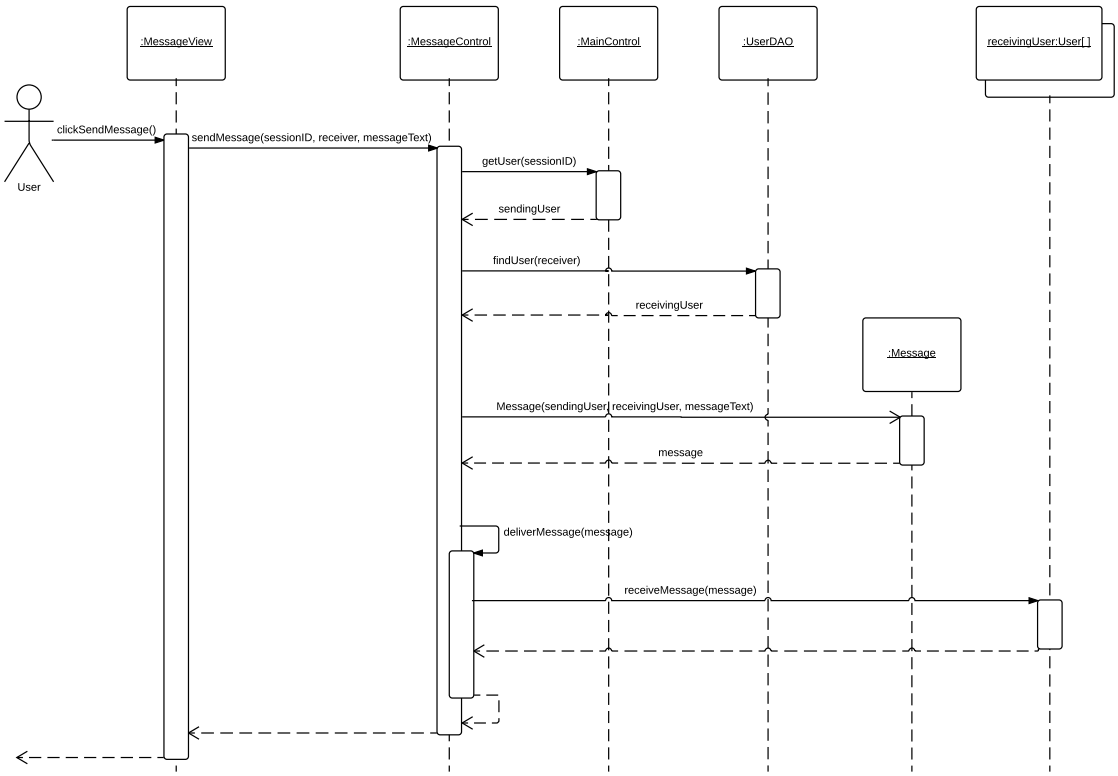
2.6 Benutzer anlegen

Ein neuer User wird angelegt. Falls LDAP zur Authentifizierung genutzt wird, muss kein Passwort angegeben werden. Falls ein Administrator angelegt wird, muss name, supervisor und hoursPerMonth nicht angegeben werden. Falls ein Supervisor angelegt wird, muss supervisor nicht angegeben werden. Diese Aktion kann nur von Administratoren ausgeführt werden.



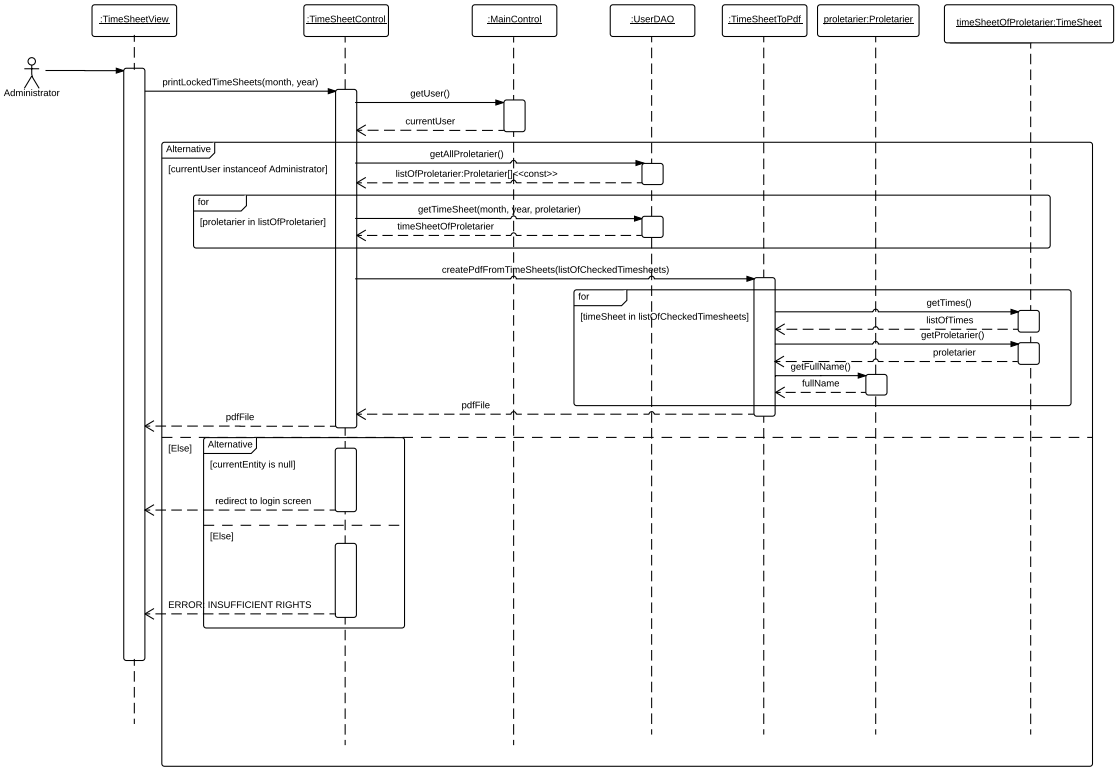
2.7 Nachrichten verschicken

Eine Nachricht wird von einem User an einen anderen User verschickt.



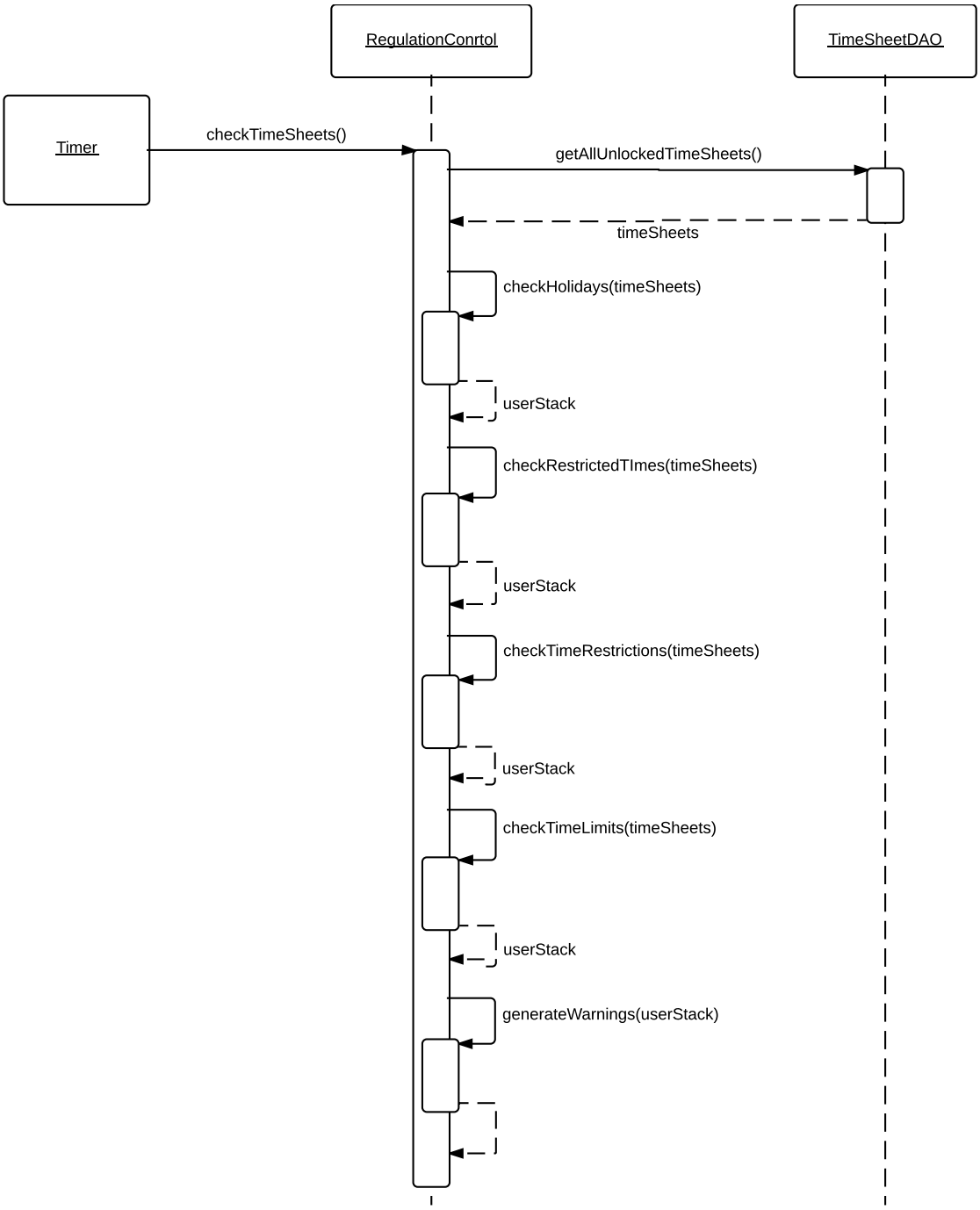
2.8 Administrator exportiert alle Stundenzettel

Ein Administrator exportiert alle Stundenzettel eines Monats als ein zusammenhängendes pdf die den Status checked haben. Administratoren können zudem alle Stundenzettel eines Monats unabhängig vom Status, einen oder alle Stundenzettel eines Proletariers exportieren. Proletarier können einen oder alle ihre Stundenzettel exportieren. Supervisor können von ihren betreuten Proletariern alle Stundenzettel eines Monats die checked sind oder unabhängig von ihrem status exportieren. Zudem können sie einen oder alle Stundenzettel eines von ihnen betreuten Proletariers exportieren.



2.9 Alle Stundenzettel mit Zustand unlocked prüfen

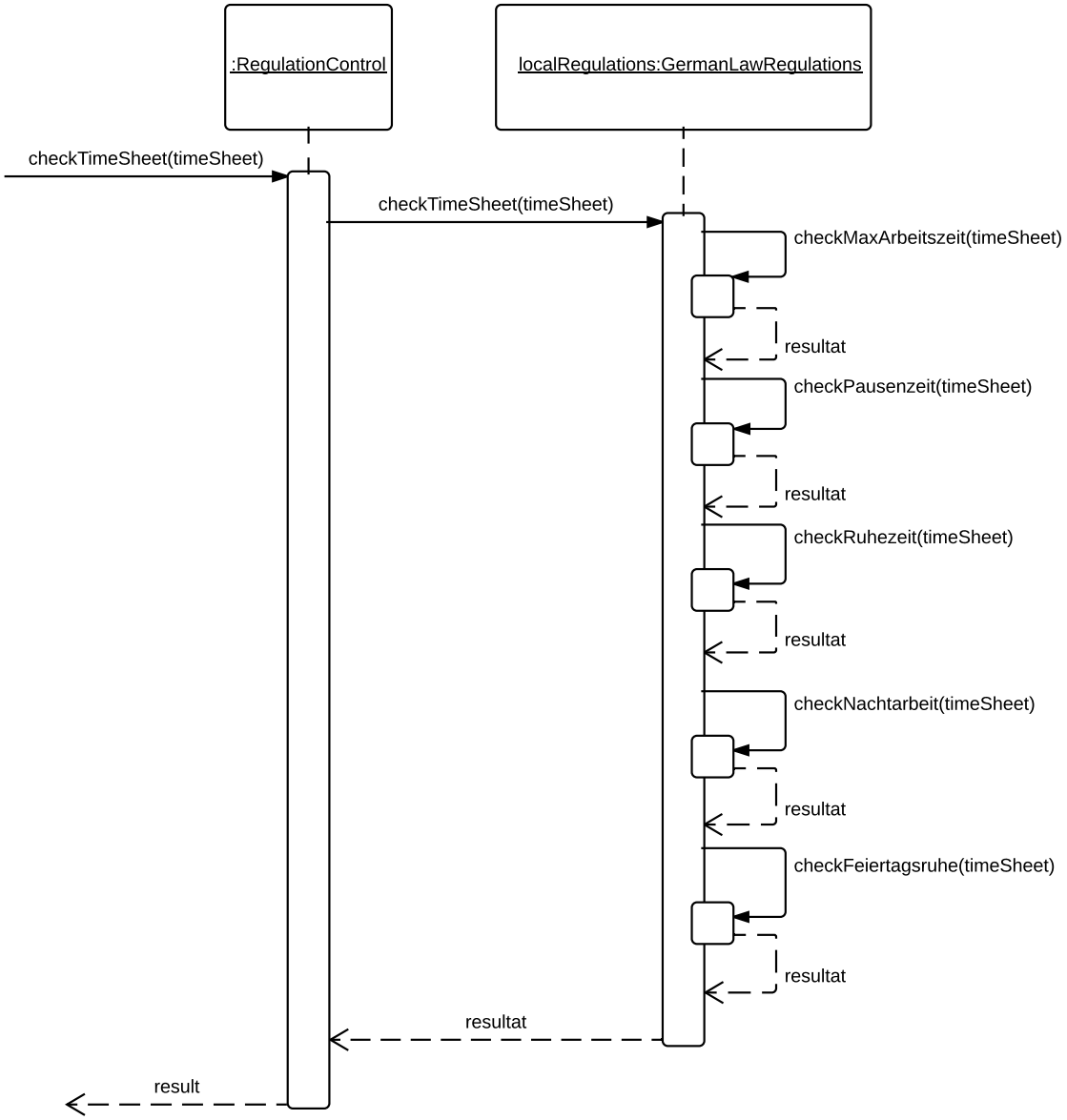
Alle Stundenzettel mit dem Zustand unlocked werden auf inkonsistenzen geprüft, zum Beispiel ob der Proletarier gewarnt werden muss, dass der Stundenzettel abgegeben werden muss.



2.10 Prüfung eines Stundenzettels auf Rechtskonformität

Ein Stundenzettel wird auf Rechtskonformität geprüft. Dies beinhaltet die folgenden gesetzlichen Regelungen:

- Die gesetzlich maximale Arbeitszeit wird nicht überschritten (§3 ArbZG).
- Die gesetzliche Pausenzeiten werden eingehalten (§4 ArbZG).
- Die Ruhezeiten wurden eingehalten (§5 ArbZG).
- Es hat keine Nachtarbeit stattgefunden (§6 ArbZG).
- Die Sonn- und Feiertagsruhe wurde eingehalten (§9 ArbZG). Dabei werden nur die Feiertage in Baden-Württemberg beachtet.


:RegulationControl
localRegulations:GermanLawRegulations
checkTimeSheet(timeSheet)
checkTimeSheet(timeSheet)
checkMaxArbeitszeit(timeSheet)
resultat
checkPausenzeit(timeSheet)
resultat
checkRuhezeit(timeSheet)
resultat
checkNachtarbeit(timeSheet)
resultat
checkFeiertagsruhe(timeSheet)
resultat
resultat
result

3 Datenbankdesign

3.1 ER Diagramm

4 Dateiformate

4.1 Dateiformat zur Definition von gesetzlichen Feiertagen

In einer Datei „Feiertage.dat“ können die gesetzlichen Feiertage des entsprechenden Bundeslandes konfiguriert werden. Ausgeliefert wird das Programm mit den gesetzlichen Feiertagen für Baden-Württemberg. Wenn die Datei geändert wurde, muss das Programm danach neu kompiliert werden, da im Quellcode eine Prüfsumme der Datei hinterlegt ist um Manipulation zu erschweren. Die Datei ist ASCII-kodiert.
In jeder Zeile steht der Name eines Feiertags gefolgt von einem Doppelpunkt und dem Datum. Das Datum kann auf zwei Arten hinterlegt werden:

- Das Datum im Format TT.MM., also zum Beispiel 05.01.
- Als Anzahl der Tage vor oder nach dem Osterdatum, im Format +N bzw -N, also zum Beispiel +60

4.2 Datenbankanbindung

In der Datei „hibernate.cfg.xml“ wird die Datenbankanbindung mittels der Variablen connection.driver_class, connection.url, connection.username und connection.password definiert. Diese Datei muss Die Syntax der Datei entspricht den Vorgaben von Hibernate ORM 5.0.

4.3 Authentifizierung via LDAP

Falls die Authentifizierung via LDAP erwünscht ist, muss die Anbindung der „Shiro.ini“ konfiguriert werden. Die Syntax der Datei entspricht den Vorgaben von Shiro 1.2.4.

5 Glossar

DAO Data Access Object, eine Klasse mit der auf Klassen in der Datenbank zugegriffen werden kann.

User Eine Person die sich einloggen kann.